RODOLFO MIRANDA PEREIRA

DEALING WITH IMBALANCENESS IN HIERARCHICAL CLASSIFICATION PROBLEMS THROUGH DATA RESAMPLING

Thesis presented to the Graduate Program in Informatics (PPGIa) of the Pontifícia Universidade Católica do Paraná (PUCPR) as partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Curitiba

2020

RODOLFO MIRANDA PEREIRA

DEALING WITH IMBALANCENESS IN HIERARCHICAL CLASSIFICATION PROBLEMS THROUGH DATA RESAMPLING

Thesis presented to the Graduate Program in Informatics (PPGIa) of the Pontifícia Universidade Católica do Paraná (PUCPR) as partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Concentration Area: Artificial Intelligence

Supervisor: Prof. Dr. Carlos Nascimento Silla Jr.
Co-supervisor: Prof. Dr. Yandre Maldonado e Gomes da Costa

Curitiba

2020

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Luci Eduarda Wielganczuk – CRB 9/1118

P436d
2020

Pereira, Rodolfo Miranda
Dealing with imbalanceness in hierachical classification problems through data resampling / Rodolfo Miranda Pereira ; supervisor: Carlos Nascimento Silla Jr. ; co-supervisor: Yandre Maldonado e Gomes da Costa. – 2020.
292 f. : il. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2020
Bibliografia: f. 256-275

1. Informática. 2. Algoritmos. 3. Processamento eletrônico de dados. I. Silla Júnior, Carlos Nascimento. II. Costa, Yandre Maldonado e Gomes da III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

CDD 22. ed. – 004

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

DECLARAÇÃO

Declaro para os devidos fins que o aluno **RODOLFO MIRANDA PEREIRA**, defendeu sua tese de Doutorado intitulada **"Dealing with Imbalanceness in Hierarchical Classification Problems Through Data Resampling"**, na área de concentração Ciência da Computação, no dia 03 de agosto de 2020, no qual foi aprovado.

Declaro ainda que foram feitas todas as alterações solicitadas pela Banca Examinadora, cumprindo todas as normas de formatação definidas pelo Programa.

Por ser verdade, firmo a presente declaração.

Curitiba, 17 de novembro de 2020.

Prof. Dr. Emerson Cabrera Paraiso
Coordenador do Programa de Pós-Graduação em Informática
Pontifícia Universidade Católica do Paraná

To my beloved wife, Cláudia.

Acknowledgements

Fist of all, I would like to thank my wife Cláudia Trevizan for all the support and patience that she had during the long development of this Doctoral Research. She was always my inspiration and wall of comfort during the hard days and nights.

I thank my supervisors Carlos Silla Jr and Yandre Costa for guiding me during the studies that lead me to this Thesis. I'm happy to have the honor to work with such good researchers as you. In the end, each reunion during the daybreak was worthy. In special, I thank Yandre Costa for showing me an open door when I was almost giving up.

I thank Júlio Nievola, Deborah Carvalho, Luiz Merschmann and Andre Leon de Carvalho for their insightful comments during the Thesis defense, which have increased the quality of this document.

I thank the Pontifícia Universidade Católica do Paraná (PUCPR) for the opportunity and structure for the studies. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

I would also like to thank all my friends from PPGIA that were somehow together with me during this four years of studies. In special, I thank João Pedro, Lucas Martiniano and Zacarias Curi for being such good friends during this phase. I'm grateful for meeting you guys in PPGIA, this is for sure one of the most important things that I've accomplished in this Doctoral Research.

I thank my friends from IFPR, Alessandra Zavala, Ana Carolina Carvalho, Andrius Roque, Eduardo Tieppo, Gledson Bianconi, Guilherme Gasparotto and Ronan Assumpção, and all my friends from my hometown, Alan Clappis, Ana Leticia, Bruno Godoy, Daniel Rosseto, Danilo Bueno, Débora França, Débora Sandi, Derick Gobetti, Erick Klein, Felippe Fernandes, Germano Paiola, Helen Pascoeto, Leonardo Hayashi, Leonardo Martins, Leonardo Saiki, Lucas Maldonado, Lucas Perassoli, Luiz Henrique Davantel, Guilherme Shiba, Maicon Michelon, Maurílio Campano, Rafael Aguiar, Ruan Liberato, Thales Chicoski and Yago Legal for supporting me somehow during the development of this project. In special, I thank Bruno, Leonardo and Yago for letting me stay at their home in the first year of the Doctoral Research, I really appreciate your efforts to make me comfortable.

I also thank all the support of my in-laws, in special Bianca Trevizan, Felipe Amadeo, Vera Trevizan and Pedro Segovia. You are also part of my life and have helped me somehow in this journey.

Last and not less important, I thank my parents Clarice Ruiz Miranda and José Lopes Pereira, my grandmother Nair Ruiz and my brother and sister, Guilherme Pereira and Karen Pereira, for sending me positive thoughts and supporting me during the last four years. Everything that I've become and achieved in my life is part of you.

Abstract

The process of classifying unlabeled and unseen data based on patterns of previously known samples is one of the main interests in Machine Learning. In this context, a lot of classification problems are naturally imbalanced, which means that some classes are less frequent then others, and even though many classification algorithms tend to benefit the most frequent classes during the training process, in scenarios such as disease identification, the main interest is exactly to maximize the identification rate of the less frequent ones. Although the data resampling algorithms are the most common and widely used solutions, they were not well defined for the hierarchical classification scenario. Given that many important real-world problems are casted as hierarchical, this is an open research gap. Thus, the main objective of this Doctoral Research is to study and propose resampling approaches to handle the class imbalanceness issue for hierarchical classification problems. In order to achieve this objetive, four directions were investigated: (i) The use of well-known binary and multi-label resampling methods in hierarchical datasets with single and multiple paths, respectively; (ii) A label path conversion strategy to deal with the imbalanceness in hierarchical multi-label datasets with multi-label resampling techniques; (iii) The design of classification schemas to use binary resampling algorithms with the different local classificaiton approaches; (iv) The proposal of global resampling algorithms that are able to identify the majority/minority label paths in hierarchical datasets and deal with them as a whole, considering their different characteristics such as number of paths and depth of prediction. During the investigation of the second and third directions, we have also proposed novel metrics to measure imbalanceness in hierarchical datasets in local and global perspectives. Concerning the fourth direction, we have proposed three novel resampling algorithms: Hierarchical Random Oversampling (HROS); Hierarchical Random Undersampling (HRUS); and Hierarchical Synthetic Oversampling Technique (HSMOTE). Moreover, while studying the challenges involved in dealing with multi-label imbalanced data, we have also developed a novel multi-label resampling algorithm named Multi-Label Tomek Link (MLTL). All the proposed approaches were experimentally analyzed in well-known hierarchical datasets, with different characteristics and, supported by statistical analysis, were able to somehow improve the classification results. In order to show the impacts of the contributions of this Thesis in a real world scenario, we have investigated the imbalanceness issue in the COVID-19 identification in chest x-ray images problem. The proposed resampling approaches were able to improve the COVID-19 identification rate in most scenarios. This was the first work to analyze this problem with a hierarchical taxonomy, considering the biological relationships between the pneumonia pathogens.

Keywords: Hierarchical Classification, Imbalanced Learning, Class Imbalanceness, Resampling Algorithms.

Resumo

O processo de classificação de dados não rotulados com base em padrões de amostras previamente conhecidas é uma das principais tarefas da Aprendizagem de Máquina. Neste sentido, muitos problemas de classificação são naturalmente desbalanceados, o que significa que algumas classes são menos frequentes do que outras, e embora muitos algoritmos de classificação tendam a beneficiar as classes mais frequentes durante o processo de treinamento, em cenários como detecção de doenças, o principal interesse é exatamente maximizar a taxa de identificação das classes menos frequentes. Apesar dos algoritmos de *resampling* serem as soluções mais amplamente utilizadas, elas não foram bem definidas no cenário de classificação hierárquica. Dado que muitos problemas importantes do mundo real são considerados hierárquicos, esta é uma lacuna de pesquisa. Assim, o objetivo principal deste trabalho é estudar e propor abordagens de *resampling* para lidar com o desbalanceamento de classes em problemas de classificação hierárquicos. Para atingir este objetivo, quatro direções foram investigadas: (i) O uso de métodos de *resampling* binários e multi-rótulos em conjuntos de dados hierárquicos com caminhos únicos e múltiplos, respectivamente; (ii) Uma estratégia de conversão de rótulos hierárquicos para utilização de técnicas de *resampling* multi-rótulo; (iii) O desenvolvimento de esquemas de classificação para o uso de algoritmos de *resampling* binários em abordagens hierárquicas locais; (iv) Algoritmos de *resampling* globais capazes de identificar os rótulos majoritários/minoritários em conjuntos de dados hierárquicos e tratá-los como um todo, considerando suas diferentes características como número de rótulos e profundidade de predição. Durante as investigações da segunda e terceira direções, também foram propostas novas métricas para medir o desbalanceamento em conjuntos de dados hierárquicos em perspectivas locais e globais. Em relação à quarta direção, foram propostos três novos algoritmos: *Hierarchical Random Oversampling* (HROS); *Hierarchical Random Undersampling* (HRUS); e *Hierarchical Synthetic Oversampling Technique* (HSMOTE). Além disso, ao estudar os desafios envolvendo desbalanceamento multi-rótulo, foi desenvolvido um novo algoritmo de *resampling* chamado *Multi-Label Tomek Link* (MLTL). As abordagens propostas foram analisadas em uma grande variedade de bases de dados hierárquicos e, apoiados por análises estatísticas, conseguiram melhorar os resultados da classificação. Para mostrar os impactos das contribuições desta Tese em um cenário real, o problema do desbalanceamento na identificação de COVID-19 em imagens de raios-X de tórax foi investigado. As abordagens propostas foram capazes de melhorar a taxa de identificação do COVID-19 na maioria dos cenários. Este foi o primeiro trabalho a analisar este problema com uma taxonomia hierárquica, considerando as relações biológicas entre os patógenos da pneumonia.

Palavras-Chave: Classificação Hierárquica, Aprendizado Desbalanceado, Desbalanceamento entre Classes, Algoritmos de Reamostragem.

List of Figures

Figure 1 – Inductive Learning Hierarchy. 39
Figure 2 – Example of kNN classification algorithm. 40
Figure 3 – Example of Decision Tree classification algorithm (TAN; STEINBACH; KUMAR, 2005). 41
Figure 4 – Example of Random Forest classification schema. 42
Figure 5 – Example of SVM classification algorithm (MEYER, 2004). 43
Figure 6 – Categorization of multi-label learning evaluation measures (adapted from Madjarov et al. (2012)). 46
Figure 7 – Examples of hierarchical organization (adapted from Silla Jr & Freitas (2011)). 50
Figure 8 – Classifiers approaches - circles represent classes and each dashed rectangle encloses the classes predicted the classifier (adapted from Silla Jr & Freitas (2011)). 53
Figure 9 – The positive/negative policies of the local classifiers per node approach. Blue nodes represent the positive labels, red the negative and white the labels that will not be used. 54
Figure 10 – Label combination process of the HMC-LP method (CERRI; CARVALHO, 2010). 56
Figure 11 – Label decomposition process of the HMC-CT method (CERRI; CARVALHO, 2010). 58
Figure 12 – Example of hierarchical precision and recall measure. The real labels are marked in gray. 63
Figure 13 – Example of hierarchical precision and recall measure. The predicted labels are marked in pink. 63
Figure 14 – Graphical representation of the H-Loss function (CERRI et al., 2015). . 65
Figure 15 – Different classes distribution in a binary labeled dataset. 72
Figure 16 – Example of the binarization techniques for a 3-class problem (FERNÁNDEZ et al., 2013). 73
Figure 17 – Example of Undersampling technique. 74
Figure 18 – Classic Tomek Link Identification. 75
Figure 19 – Tomek Link's types of use. 76
Figure 20 – Example of NearMiss undersampling technique. 77
Figure 21 – Example of CNEN undersampling technique. 78
Figure 22 – Example of ENN/RENN undersampling techniques. 80
Figure 23 – Example of CC undersampling technique. 81
Figure 24 – Example of NCL undersampling method. 82

Figure 25 – Example of All-KNN undersampling method. 83
Figure 26 – Example of SMOTE interpolation to create the synthetic samples (RAMENTOL et al., 2012). 85
Figure 27 – Example of SMOTE oversampling technique. 87
Figure 28 – Example of SMOTE-Borderline oversampling techniques. 88
Figure 29 – Example of Safe-Level SMOTE oversampling technique. 90
Figure 30 – Example of SPIDER oversampling method. 93
Figure 31 – Example of SPIDER-2 oversampling method. 93
Figure 32 – Example of ADASYN technique oversampling. 94
Figure 33 – Example of the SMOTE + ENN hybrid technique. 96
Figure 34 – Example of the SMOTE + Tomek Link hybrid technique. 97
Figure 35 – Ten most frequent and ten least frequent labels in some datasets (adapted from Herrera et al. (2016)). The frequency scale is individually adjusted to show better the relevance of labels in each MLD, instead of being common to all plots. 100
Figure 36 – Concurrence among the labels in Yeast Dataset (CHARTE et al., 2017). 101
Figure 37 – The Multi-Label Tomek Link (in this illustration, each color refers to one of the labels assigned to the instance). 115
Figure 38 – Graphical Overview of the Multi-Label Tomek Link. 116
Figure 39 – Resampling schema for the application of binary resampling algorithms in Hierarchical problems with single paths. 127
Figure 40 – Resampling schema for the application of multi-label resampling algorithms in Hierarchical problems with multiple paths. 128
Figure 41 – Example of Label Paths in a Hierarchical Taxonomy. 135
Figure 42 – Schema of the Proposed Resampling Approach. 136
Figure 43 – Example of HMD↔ML Conversion. 137
Figure 44 – Longest Paths Identification. 138
Figure 45 – Results Comparison Graphics. 142
Figure 46 – An example of a hierarchical classification dataset subdivided into train and test. 149
Figure 47 – The general classification schema for the LCN approach. 150
Figure 48 – Example of resampling schema for hierarchical datasets using LCN with two different policies. 151
Figure 49 – The general classification schema using the LCPN approach. 152
Figure 50 – Example of resampling schema for hierarchical datasets using LCPN with the two different policies. 153
Figure 51 – The general classification schema using the LCL approach. 154
Figure 52 – Example of resampling schema for hierarchical datasets using LCL. . . 154

Figure 53 – Mean Imbalance Ratio for Local Classifiers per Node (IR_{LCN}) for each dataset before (original dataset) and after resampling. 160
Figure 54 – Mean Imbalance Ratio for Local Classifiers per Parent Node (IR_{LCPN}) for each dataset before and after resampling. 161
Figure 55 – Mean Imbalance Ratio for Local Classifiers per Level (IR_{LCL}) for each dataset before and after resampling. 162
Figure 56 – Charts with the best F-Score results for all techniques in each dataset. 163
Figure 57 – An example of the main issue when creating samples in leaf nodes. The numbers on the top left of the nodes symbolize the number of samples belonging to each node. 173
Figure 58 – An example of application of the HROS-PD in a dataset with 85 instances. The nodes marked with a circled dashed are being processed at the certain step and the red nodes represent the label paths belonging to the minority set. 174
Figure 59 – Mean Imbalance Ratios for the Datasets. 177
Figure 60 – An example of the main issue when creating samples in leaf nodes. The numbers on the top left of the nodes symbolize the number of samples belonging to each node. 189
Figure 61 – An example of the application of an oversampling method in a dataset with 85 instances. The nodes marked with a circled dashed are being processed at the certain step and the red nodes represent the label paths belonging to the minority set. 190
Figure 62 – The hierarchical class structure of pneumonia caused by micro-organisms. 214
Figure 63 – A general classification schema for the COVID-19 identification in CXR images. While the blue lines represents the early fusion connections, the pink lines are used for the late fusion and results without fusion. 221
Figure 64 – The baseline resampling schema for the COVID-19 identification in CXR images. 227
Figure 65 – The resampling schema with local classifiers for the COVID-19 identification in CXR images. 228
Figure 66 – The classification schema with global resampling for the COVID-19 identification in CXR images. 229
Figure 67 – RYDLS-20 image samples. 230
Figure 68 – F-scores per label in the best macro-avg scenario using the baseline approach. 234
Figure 69 – F-scores per label in the best COVID-19 identification scenario using the baseline approach. 235
Figure 70 – Confusion Matrix in the best macro-avg scenario using the baseline approach. 235

Figure 71 – Confusion Matrix in the best COVID-19 identification scenario using the baseline approach. 236
Figure 72 – F-scores per label in the best macro-avg scenario using the local classifiers with Resampling approaches. 237
Figure 73 – F-scores per label in the best COVID-19 identification scenario using the local classifiers with Resampling approaches. 238
Figure 74 – Confusion Matrix in the best macro-avg scenario using the local classifiers with Resampling approaches. 238
Figure 75 – Confusion Matrix in the best COVID-19 identification scenario using the local classifiers with Resampling approaches. 239
Figure 76 – F-scores per label in the best macro-avg and COVID-19 identification scenarios using the global resampling approach. 241
Figure 77 – Confusion Matrix in the best macro-avg and COVID-19 identification scenarios using the global resampling approach. 242
Figure 78 – Examples of samples with “COVID-19” label that were predicted as “Normal”. 248
Figure 79 – Different examples of CXR with “normal” lungs. 248

List of Tables

Table 1 – Papers developed during this Doctoral Research. 33
Table 2 – Papers that were collaboratively developed during this Doctoral Research. 34
Table 3 – Novel datasets proposed in this Thesis. 35
Table 4 – Summary of the main media reports concerning our work. 36
Table 5 – Feature vector example. 38
Table 6 – Categorization of multi-label classification methods. 45
Table 7 – Example of differences between labelsets. 112
Table 8 – Hypothetical Dataset D1. 114
Table 9 – Hypothetical Dataset D2. 114
Table 10 – Hypothetical Dataset D3. 115
Table 11 – Characteristics of the datasets used in the experiments. 117
Table 12 – Parameter settings of the classifiers used in this work. 118
Table 13 – MeanIR of the datasets before/after apply the resampling methods. . . 119
Table 14 – Experimental results for the CAL500 dataset. 120
Table 15 – Experimental results for the Emotions dataset. 120
Table 16 – Experimental results for the Enron dataset. 120
Table 17 – Experimental results for the FMA dataset. 121
Table 18 – Experimental results for the Medical dataset. 121
Table 19 – Experimental results for the Scene dataset. 121
Table 20 – Experimental results for the Yeast dataset. 122
Table 21 – Wilcoxon statistical tests for F-score results. 122
Table 22 – Ranking for the classification algorithms using the Friedman test. 123
Table 23 – Time complexity of the multi-label resampling algorithms. 124
Table 24 – Hierarchical Datasets with Single and Multiple Paths. 129
Table 25 – Clus-HMC execution parameters. 130
Table 26 – Results for the hierarchical classification datasets with single paths before and after applying binary resampling. 130
Table 27 – Results for the hierarchical classification datasets with multiple paths before and after applying multi-label resampling (Part 1). 131
Table 28 – Results for the hierarchical classification datasets with multiple paths before and after applying multi-label resampling (Part 2). 131
Table 29 – Wilcoxon test for the hierarchical datasets with single paths. 132
Table 30 – Wilcoxon test for the hierarchical datasets with multiple paths. 133
Table 31 – Clus-HMC execution parameters. 140
Table 32 – Imbalanceness features of FMA before/after resampling. 140
Table 33 – Experimental results for the proposed resampling approach. 141

Table 34 – General review of the datasets used in the experiments. 155
Table 35 – Parameter settings of the Classification algorithms. 158
Table 36 – The imbalance ratios after each step from Figure 6. Numbers in bold are above the mean imbalance ratio - *HMeanIR*. 174
Table 37 – Datasets with Full Depth Hierarchical Classification Problems. 177
Table 38 – Datasets with Partial Depth Hierarchical Classification Problems. 177
Table 39 – Clus-HMC execution parameters. 179
Table 40 – Results for the full depth hierarchical datasets with single paths. 179
Table 41 – Results for the full depth hierarchical datasets with multiple paths. . . . 180
Table 42 – Results for the partial depth hierarchical datasets with single paths. . . 180
Table 43 – Results for the partial depth hierarchical datasets with multiple paths (Part 1). 181
Table 44 – Results for the partial depth hierarchical datasets with multiple paths (Part 2). 181
Table 45 – *P*-values of the Wilcoxon signed-rank statistical test for the Full Depth Random Resampling Algorithms. 182
Table 46 – *P*-values of the Wilcoxon signed-rank statistical test for the Partial Depth Random Resampling Algorithms. 182
Table 47 – Example of Label Paths Combinations for a partial depth problem with single paths considering the minority label path “A/B/C” and five neighbors. 192
Table 48 – Example of Label Paths Combinations for a FD problem with multiple paths considering the minority label path “A/B/C” five neighbors. . . . 193
Table 49 – Example of Label Paths Combinations for a PD problem with multiple paths considering the minority label path “D/F” five neighbors. 194
Table 50 – Hierarchical Classification Datasets with FD Problems. Datasets marked with (*) were originally proposed as flat datasets and were adapted to a hierarchical taxonomy. 201
Table 51 – Hierarchical Classification Datasets with PD Problems. Datasets marked with (*) and (**) were originally proposed as flat datasets and were adapted to a hierarchical taxonomy. 202
Table 52 – Clus-HMC execution parameters. 203
Table 53 – AUPRC classification results for the datasets with FD and single paths before and after applying HSMOTE in the training set. 204
Table 54 – AUPRC classification results for the datasets with FD and multiple paths before and after applying HSMOTE in the training set. 204
Table 55 – AUPRC classification results for the datasets with PD and single paths before and after applying HSMOTE in the training set. 205

Table 56 – AUPRC classification results for the datasets with PD and multiple paths before and after applying HSMOTE in the training set. 205
Table 57 – HMeanIR values before and after applying HSMOTE in the training datasets. 206
Table 58 – P-values of the Wilcoxon signed-rank statistical test for the best results with HSMOTE. 207
Table 59 – The best classification results before and after applying HSMOTE in each training set and its variances against the original results. 208
Table 60 – Ranking of the results for the strategies used in the FD with SP datasets. 208
Table 61 – Ranking of the results for the strategies used in the FD with MP datasets. 209
Table 62 – Ranking of the results for the strategies used in the PD with SP datasets. 209
Table 63 – Ranking of the results for the strategies used in the PD with MP datasets. 209
Table 64 – Summary of the works described in this section. 220
Table 65 – Features dimensions and main parameters. 225
Table 66 – RYDLS-20 main characteristics. 229
Table 67 – RYDLS-20 samples distribution for the hierarchical scenario. 231
Table 68 – Random Forest parameter settings. 231
Table 69 – Clus-HMC parameter settings. 232
Table 70 – Best macro-avg results for each prediction schema in the Baseline approach. 233
Table 71 – Best COVID-19 results for each prediction schema in the Baseline approach. 234
Table 72 – IRLP measure results for the baseline experiments. 236
Table 73 – Best macro-avg results for each prediction schema in the local classifiers with Resampling approaches. 237
Table 74 – Best COVID-19 results for each prediction schema in the local classifiers with Resampling approaches. 237
Table 75 – IR_{LCN} measure results for the local classifiers per node experiments. . . 239
Table 76 – IR_{LCPN} measure results for the local classifiers per parent node experiments. 240
Table 77 – IR_{LCL} measure results for the local classifiers per parent node experiments. 240
Table 78 – Best macro-avg results for each prediction schema in the global hierarchical resampling approach. 240
Table 79 – Best COVID-19 identification results for each prediction schema in the global hierarchical resampling approach. 241
Table 80 – IRLP measure results for the global hierarchical resampling experiments. 242
Table 81 – Ranking of the results per feature set in all classification scenarios. . . . 243
Table 82 – Wilcoxon test results for the baseline approach. 244
Table 83 – Wilcoxon test results for the local classifiers approach. 245

Table 84 – Wilcoxon test results for the global hierarchical resampling approach. The configuration values for the random algorithms represent the resize rate in %, while the values for the HSMOTE represent the number of neighbors used in the sample section stage. 245
Table 85 – Ranking of results per resampling method in the baseline classification schema. 246
Table 86 – Ranking of results per resampling method in the local classifiers schema. 246
Table 87 – Ranking of results per resampling method in the global resampling schema. 246
Table 88 – F-Score results for the proposed approaches in the Cell-cycle dataset. . . 277
Table 89 – F-Score results for the proposed approaches in the Eisen dataset. 278
Table 90 – F-Score results for the proposed approaches in the Exp dataset. 278
Table 91 – F-Score results for the proposed approaches in the FMA MFCC dataset. 279
Table 92 – F-Score results for the proposed approaches in the Gasch-1 dataset. . . . 279
Table 93 – F-Score results for the proposed approaches in the CLEF dataset. 280
Table 94 – F-Score results for the proposed approaches in the DMOZ-2010 dataset. 280
Table 95 – F-Score results for the proposed approaches in the LSHTC-small dataset. 281
Table 96 – F-Score results for the Top-Down (TD) approaches in all datasets. . . . 281
Table 97 – F-Score results for the Flat-ML approach in all datasets. 281
Table 98 – F-Score results for the Flat-MLRS approach with all datasets. 282
Table 99 – F-Score results for the Clus-HMC approach with all datasets. 282
Table 100 – F-Score results for the HierCost approach with all datasets. 282
Table 101 – Wilcoxon statistical tests for F-score results in the Flat Multi-Label scenarios. 282
Table 102 – Wilcoxon statistical tests for F-score results in the resampling for the Local Classifiers per Node approach. 283
Table 103 – Wilcoxon statistical tests for F-score results in the resampling for the Local Classifiers per Parent Node approach. 283
Table 104 – Wilcoxon statistical tests for F-score results in the resampling for the Local Classifiers per Level approach. 284
Table 105 – Average ranking of the classification results in the resampling for the Local Classifiers per Node approach. 284
Table 106 – Average ranking of the classification results in the resampling for the Local Classifiers per Parent Node approach. 284
Table 107 – Average ranking of the classification results in the resampling for the Local Classifiers per Level approach. 284
Table 108 – Average ranking of the classification results in the resampling for the all the Local Classifiers approaches. 284
Table 109 – Post-Hoc Mannwhitney Test comparing the flat with the local classifier approaches. 284

Table 110 – Pearson Correlation Statistical Test for the $MeanIR_{LCN}$ measure and the classification results. . . . 285
Table 111 – Pearson Correlation Statistical Test for the $MeanIR_{LCPN}$ measure and the classification results. . . . 285
Table 112 – Pearson Correlation Statistical Test for the $MeanIR_{LCL}$ measure and the classification results. . . . 285
Table 113 – Wilcoxon statistical tests comparing the best F-score results from the proposed approaches (LCN) *versus* the best results for each global classification approach considering all datasets. . . . 285
Table 114 – Top-10 macro-avg f-score using the baseline approach. . . . 286
Table 115 – Top-10 COVID-19 f-score using the baseline approach. . . . 287
Table 116 – Best macro-avg f-score per feature set using the baseline approach. . . . 287
Table 117 – Best COVID-19 f-score per feature set using the baseline approach. . . . 287
Table 118 – Top-10 macro-avg f-score with the early fusion technique using the baseline approach. . . . 287
Table 119 – Top-10 COVID-19 f-score with the early fusion technique using the baseline approach. . . . 288
Table 120 – Top-3 macro-avg f-score with the late fusion technique using the baseline approach. . . . 288
Table 121 – Top-3 COVID-19 f-score with the late fusion technique using the baseline approach. . . . 288
Table 122 – Top-10 macro-avg f-score using the local classifiers approaches. . . . 289
Table 123 – Top-10 COVID-19 f-score using the local classifiers approaches. . . . 289
Table 124 – Best macro f-score per feature set using the local classifiers approaches. 289
Table 125 – Best COVID-19 f-score per feature set using the local classifiers approaches. 289
Table 126 – Top-10 macro-avg f-score with the early fusion technique using the local classifiers approaches. . . . 290
Table 127 – Top-10 COVID-19 f-score with the early fusion technique using the local classifiers approaches. . . . 290
Table 128 – Top-10 macro-avg f-score the global resampling approach. . . . 291
Table 129 – Top-10 Covid-19 f-score using the global resampling approach. . . . 291
Table 130 – Best macro-avg f-score per feature set using the global resampling approach. 291
Table 131 – Best COVID-19 f-score per feature set using the global resampling approach. 291
Table 132 – Top-10 macro-avg f-score with the early fusion technique using the global resampling approach. . . . 292
Table 133 – Top-10 COVID-19 f-score with the early fusion technique using the global resampling approach. . . . 292
Table 134 – Top-3 macro-avg f-score with the late fusion technique using the global resampling approach. . . . 292

Table 135 – Top-3 Covid-19 f-score with the late fusion technique using the global resampling approach. 292

List of abbreviations and acronyms

ADASYN	Adaptive Synthetic Sampling Approach for Imbalanced Learning
AHD	Adjusted Hamming Distance
AI	Artificial Intelligence
AUC	Area Under Curve
AUPRC	Area Under the Precision-Recall Curve
AUROC	Area Under the ROC Curve
BR	Binary Relevance
BR-kNN	Binary Relevance k-Nearest Neighbors
BSIF	Binarized Statistical Image Features
CART	Classification and Regression Tree
CBR	Case-Based Reasoning
CCH	Classifier Chain
CC	Cluster Centroids
CLR	Calibrated Label Ranking
CNEN	Condensed Nearest Neighbors
CNN	Convolutional Neural Networks
COVID-19	Coronavirus Disease 2019
CT	Computed Tomography
CVIR	Coefficient of Variation of IRLbl
CXR	Chest X-Ray
DAG	Directed Acyclic Graph
DT	Decision Trees
DWT	Discrete Wavelet Transform
EBP	Eliptical Binary Pattern

EF	Early Fusion
ENN	Edited Nearest Neighbor
EQP	Elongated Quinary Pattern
FD	Full Depth labeling
Flat-MLRS	Flat Classification with Multi-Label Resampling
FMA	Free Music Archive
FPR	False Positive Rate
GC	Global Classifier
HC	Hierarchical Classification
HD	Hamming Distance
HierCost	Hierarchical Cost Sensitive Classification
H-Loss	Hierarchical Loss Function
HMC	Hierarchical Multi-Label Classification
HMC-BR	Hierarchical Multi-Label Binary Relevance
HMC-CT	Hierarchical Multi-Label Cross-Training
HMC-LP	Hierarchical Multi-Label Label-Powerset
HMD	Hierarchical Multi-Label Dataset
HMeanIR	Hierarchical Mean Imbalance Ratio
HOMER	Hierarchy of Multi-label Classifiers
HROS	Hierarchical Random Oversampling
HROS-FD	Random Oversampling for Full Depth Hierarchical Classification Problems
HROS-PD	Random Oversampling for Partial Depth Hierarchical Classification Problems
HRUS	Hierarchical Random Undersampling
HRUS-FD	Random Undersampling for Full Depth Hierarchical Classification Problems

HRUS-PD	Random Undersampling for Partial Depth Hierarchical Classification Problems
HSMOTE	Hierarchical Synthetic Oversampling Technique
HVDM	Heterogeneous Value Distance Metric
ICD-10	International Statistical Classification of Diseases and Related Health Problems
ICU	Intensive Care Unit
IR	Imbalance Ratio
IR_{LCL}	Imbalance Ratio for Local Classifiers per Level
IR_{LCN}	Imbalance Ratio for Local Classifiers per Node
IR_{LCPN}	Imbalance Ratio for Local Classifiers per Parent Node
IRLbl	Imbalance Ratio per Label
IRLP	Imbalance Ratio per Label Path
kNN	k-Nearest Neighbours
LBP	Local Binary Pattern
LC	Local Classifiers
LCard	Label Cardinality
LCL	Local Classifiers per Level
LCN	Local Classifiers per Node
LCPN	Local Classifiers per Parent Node
LDen	Label Density
LDiv	Label Diversity
LDN	Local Directional Number
LETRIST	Locally Encoded Transform Feature Histogram
LF	Late Fusion
LP	Label Path
LPS	Label Powerset

LPQ	Local Phase Quantization
LPROS	Label Powerset Random Oversampling
LPRUS	Label Powerset Random Undersampling
LTP	Local Ternary Pattern
MeanIR	Mean Imbalance Ratio
MERS	Middle East Respiratory Syndrome
ML	Machine Learning
MLC	Multi-Label Classification
MLD	Multi-Label Dataset
MLeNN	Multi-Label edited Nearest Neighbor
MLkNN	Multi-Label k-Nearest Neighbors
MLNP	Mandatory Leaf-Node Prediction
MLROS	Multi-Label Random Oversampling
MLRUS	Multi-Label Random Undersampling
MLSMOTE	Multi-Label Synthetic Minority Oversampling Technique
MLTL	Multi-Label Tomek Link
MPL	Multiple Path of Labels
MPP	Multiple Path Prediction
NCL	Neighbourhood Cleaning Rule
NCP	Novel Coronavirus Pneumonia
NM	NearMiss
NMLNP	Non-Mandatory Leaf-Node Prediction
NNR	Nearest Neighbor Rule
O-A-A	One-Against-All
O-A-O	One-Against-One
OBIF	Oriented Basic Image Features

PCT	Predictive Cluster Tree
PD	Partial Depth labeling
PLDen	Proportion of Label Diversity
PROD	Product Rule
RAkEL	Random k-LabelSets
REMEDIAL	Resampling by decoupling highly imbalanced labels
REMEDIAL-HwR	REMEDIAL Hybridization with Resampling
RENN	Repeated Edited Nearest Neighbor
RF	Random Forest
ROC	Receiver Operating Characteristics
ROS	Random Oversampling
RUS	Random Undersampling
SARS	Severe acute respiratory syndrome
SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus 2
SCUMBLE	Score of ConcUrrence among iMBalanced LabEls
SMOTE	Synthetic Minority Oversampling Technique
SPIDER	Selective Preprocessing of Imbalanced Data
SPL	Single Path of Labels
SPP	Single Path Prediction
SUM	Sum Rule
SVM	Support Vector Machines
TD	Top-Down
TH	Threshold
TL	Tomek Links
TPR	True Positive Rate
VDM	Value Difference Metric

VOTE	Voting Rule
WFT	Wavelet Frame Transform
WPT	Wavelet Packet Transform

Contents

1 INTRODUCTION . . . 28
1.1 Problem . . . 29
1.2 Objectives . . . 30
1.3 Hypothesis Statements . . . 31
1.4 Outcomes, Contributions and Repercussions . . . 32
1.4.1 Scientific Contributions . . . 32
1.4.2 Technical Contributions . . . 34
1.4.3 Social and Media Impact . . . 35
1.5 Text Organization . . . 35

2 MACHINE LEARNING . . . 37
2.1 Single-Label Classification . . . 39
2.1.1 Bayesian Classifiers . . . 40
2.1.2 K-Nearest Neighbors . . . 40
2.1.3 Decision Trees . . . 41
2.1.4 Support Vector Machines . . . 42
2.2 Multi-Label Classification . . . 43
2.2.1 Database Indicators . . . 43
2.2.2 Classification Algorithms . . . 44
2.2.3 Evaluation Measures . . . 46
2.2.3.1 Example-Based Metrics . . . 47
2.2.3.2 Ranking-Based Metrics . . . 47
2.2.3.3 Label-Based Metrics . . . 48
2.3 Hierarchical Classification . . . 49
2.3.1 Definitions . . . 50
2.3.2 Classification Algorithms . . . 52
2.3.3 Evaluation Metrics . . . 62
2.3.3.1 Hierarchy-Based Evaluation Metrics . . . 63
2.3.3.2 Distance-Based Evaluation Metrics . . . 66
2.4 Final Considerations . . . 68

3 THE IMBALANCENESS FACTOR . . . 70
3.1 Evaluating the Classification Results in Imbalanced Scenarios . . . 70
3.2 Dealing with Imbalanceness . . . 71
3.3 Classic Resampling Techniques . . . 73
3.3.1 Undersampling Algorithms . . . 74

3.3.2 Oversampling Algorithms . . . 84
3.3.3 Hybrid Resampling Techniques . . . 94
3.4 Measuring Imbalanceness in Multi-label Datasets . . . 97
3.4.1 Imbalance Ratio per Label . . . 98
3.4.2 Mean Imbalance Ratio . . . 98
3.4.3 Coefficient of Variation of IRLbl . . . 98
3.4.4 Score of Concurrence among Imbalanced Labels . . . 99
3.4.5 Visually Exploration of the Imbalanceness . . . 99
3.5 Facing Imbalanced Multi-Label Scenarios . . . 100
3.6 Multi-Label Resampling Techniques . . . 101
3.6.1 Random Algorithms . . . 102
3.6.2 MLeNN . . . 103
3.6.3 MLSMOTE . . . 104
3.6.4 REMEDIAL . . . 107
3.6.5 REMEDIAL-HwR . . . 109
3.7 Final Considerations . . . 109

4 THE MULTI-LABEL TOMEK LINK . . . 110
4.1 The Proposed Method . . . 110
4.2 Experimental Analysis . . . 116
4.2.1 The Datasets . . . 116
4.2.2 Algorithms and Parameters . . . 117
4.2.3 Results and Discussion . . . 118
4.3 The Imb-Mulan Framework . . . 124
4.4 Final Considerations . . . 125

5 USING FLAT RESAMPLING TECHNIQUES IN IMBALANCED HIERARCHICAL DATASETS . . . 126
5.1 Binary Resampling in Hierarchical Datasets with Single Paths . . . 126
5.2 Multi-Label Resampling in Hierarchical Datasets with Multiple Paths 127
5.3 Experimental Setup . . . 128
5.3.1 The Datasets . . . 128
5.3.2 Algorithm and Parameters . . . 129
5.4 Results . . . 130
5.5 Discussion . . . 131
5.6 Final Considerations . . . 133

6 A LABEL PATH CONVERSION STRATEGY FOR IMBALANCED HIERARCHICAL DATASETS . . . 134
6.1 Measuring Imbalanceness in HMDs . . . 134

6.2 Using Multi-Label Techniques to Deal with Imbalanceness 135
6.2.1 Hierarchical to Multi-Label Conversion 137
6.2.2 Multi-Label to Hierarchical Conversion 137
6.2.3 Approach Limitations 138
6.3 Experimental Evaluation and Discussions 139
6.3.1 Results 139
6.3.2 Analysis and Discussion 140
6.4 Final Considerations 143

7 DEALING WITH IMBALANCED HIERARCHICAL DATASETS ON LOCAL CLASSIFICATION APPROACHES 144
7.1 Measuring the Imbalanceness with Local Perspectives 144
7.1.1 Imbalance Metrics for the LCN Approach 146
7.1.2 Imbalance Metrics for the LCPN Ppproach 147
7.1.3 Imbalance Metrics for the LCL Approach 147
7.2 Proposed Approaches 148
7.2.1 Resampling Using the LCN Approach 149
7.2.2 Resampling Using the LCPN Approach 150
7.2.3 Resampling Using the LCL Approach 152
7.3 Experimental Protocol and Results 155
7.3.1 The Datasets 155
7.3.2 Proposed Approaches 156
7.3.3 Baseline Approaches 156
7.3.4 State-of-the-art Approaches 157
7.3.5 Parameters and Configurations 157
7.3.6 Hierarchical Local Imbalanceness Metrics Results 158
7.3.7 Classification Results 159
7.4 Analysis and Discussion 162
7.5 Final Considerations 167

8 GLOBAL APPROACHES: THE HIERARCHICAL RANDOM RESAMPLING ALGORITHMS 169
8.1 The Proposed Random Resampling Algorithms 169
8.1.1 Finding the Majority and Minority Classes 170
8.1.2 Resampling Full Depth Hierarchical Classification Problems 172
8.1.3 Resampling Partial Depth Hierarchical Classification Problems 173
8.2 Experimental Protocol and Results 176
8.2.1 The Datasets 176
8.2.2 Classification Algorithm and Parameters 178
8.2.3 Experimental Setup 178

8.2.4 Results . . . 179
8.3 Analysis and Discussion . . . 181
8.4 Final Considerations . . . 183

9 GLOBAL APPROACHES: THE HIERARCHICAL SYNTHETIC OVER-SAMPLING ALGORITHM . . . 185
9.1 The Synthetic Oversampling Techniques . . . 185
9.2 The Proposed HSMOTE Technique . . . 186
9.2.1 Minority Instances Selection . . . 187
9.2.2 Dealing with different kinds of hierarchical classification problems . . . 188
9.2.3 Nearest neighbor search . . . 190
9.2.4 Feature set generation . . . 190
9.2.5 Synthetic labelset production . . . 190
9.2.6 HSMOTE Pseudocode . . . 193
9.2.6.1 How HSMOTE deals with Full Depth problems with Single Paths . . . 196
9.2.6.2 How HSMOTE deals with Full Depth problems with Multiple Paths . . . 199
9.2.6.3 How HSMOTE deals with Partial Depth problems with Single Paths . . . 199
9.2.6.4 How HSMOTE deals with Partial Depth problems with Multiple Paths . . . 200
9.3 Experimental Protocol and Results . . . 201
9.3.1 The Datasets . . . 201
9.3.2 Classification Algorithm and Parameters . . . 202
9.3.3 Results . . . 203
9.4 Analysis and Discussion . . . 205
9.5 Final Considerations . . . 210

10 A CASE STUDY OF IMBALANCENESS IN COVID-19 IDENTIFICATION IN CHEST X-RAY IMAGES . . . 211
10.1 COVID-19 Pandemic and the Pneumonia Disease . . . 213
10.2 Hierarchical Structure of the Problem . . . 214
10.3 Related Works . . . 215
10.4 General Classification Schema . . . 219
10.4.1 Feature Extraction (Phase 1) . . . 221
10.4.2 Early Fusion (Phase 2) . . . 225
10.4.3 Late Fusion (Phase 5) . . . 225
10.5 Hierarchical Classification with Flat Resampling . . . 226
10.6 Local Classifiers with Resampling . . . 226
10.7 Resampling with Global Algorithms . . . 227
10.8 Experimental Setup . . . 227
10.8.1 The Database . . . 228
10.8.2 Algorithms, Parameters and Metrics . . . 231

10.8.3 Evaluation Metric . . . 232
10.9 Experimental Results . . . 233
10.9.1 Baseline Results . . . 233
10.9.2 Local Classifiers with Resampling . . . 236
10.9.3 Global Hierarchical Resampling . . . 240
10.10 Discussions . . . 242
10.11 Final Considerations . . . 249

11 CONCLUSIONS . . . 251
11.1 Concluding Remarks . . . 251
11.2 Research Gaps and Future Work Directions . . . 254

BIBLIOGRAPHY . . . 256

APPENDIX 276

APPENDIX A – EXPERIMENTAL RESULTS OF LOCAL CLASSIFIERS WITH RESAMPLING . . . 277
A.1 Classification Results . . . 277
A.2 Statistical Tests . . . 282

APPENDIX B – EXPERIMENTAL RESULTS OF THE COVID-19 IDENTIFICATION STUDY CASE . . . 286
B.1 Baseline Results . . . 286
B.1.1 No Fusion . . . 286
B.1.2 Early Fusion . . . 287
B.1.3 Late Fusion . . . 288
B.2 Local Classifiers with Resampling . . . 288
B.2.1 No Fusion . . . 289
B.2.2 Early Fusion . . . 290
B.3 Global Hierarchical Resampling . . . 290
B.3.1 No Fusion . . . 291
B.3.2 Early Fusion . . . 292
B.3.3 Late Fusion . . . 292

Chapter 1 Introduction

Machine Learning is an area of Computer Science that has evolved from the study of Pattern Recognition and the theory of Computational Learning in Artificial Intelligence. The term "Machine Learning" was created by Arthur L. Samuel in 1959 and defined as "the field of study that gives computers the ability to learn without being explicitly programmed" (SAMUEL, 1988). The Machine Learning tasks are typically classified into three categories, depending on whether or not there is a "learning signal" available to the learning system: a) Supervised learning, when the learning algorithms are presented with samples and their outputs; b) Unsupervised learning, in which samples without their outputs are given to the learning algorithms; c) Semi-supervised learning, which combines a small amount of labeled data with a large amount of unlabeled data in the learning algorithms.

One of the most important tasks in Machine Learning is the supervised learning classification. This subarea explores the study and construction of algorithms that can learn from labeled input data and make predictions. Such algorithms operate by constructing a model from these sample inputs to make predictions or decisions guided by data rather than simply following inflexible and static programmed instructions. In other words, the goal is to train a computational model using a set of labeled data, obtaining a classifier able to label new, never seen before, unlabeled samples.

A large amount of research in the Machine Learning community has focused on flat classification problems. By flat we are referring to binary, multi-class or multi-label classification problems in which there are no relationships between the classes. First, the difference between a binary and multi-class is that in binary problems a sample can only have two possible classes, whilst in a multi-class classification problem an instance can have in principle any number of classes. Second, in binary and multi-class problems a sample is associated with only one label, whereas in multi-label classification problems each of the instances can be associated to more than one class (TSOUMAKAS; KATAKIS, 2006).

Nevertheless, according to Silla Jr & Freitas (2011), many important real-world classification problems are naturally cast as hierarchical, in which the predicted classes are organized into a class hierarchy, such as Text Categorization (KOLLER; SAHAMI, 1997; CHAKRABARTI et al., 1998; LI; KUANG; LING, 2012), Protein Function Prediction

(TIPTON, 1994; COSTA et al., 2008; OTERO; FREITAS; JOHNSON, 2009; LIN et al., 2013), Music Genre Classification (BURRED; LERCH, 2003; DECORO; BARUTCUOGLU; FIEBRINK, 2007; SILLA JR; FREITAS, 2009b; ARIYARATNE; ZHANG, 2012), Biological Sequence Classification (SZALKAI; GROLMUSZ; HANCOCK, 2018), Image classification (BINDER; KAWANABE; BREFELD, 2009; DIMITROVSKI et al., 2011a), Credit Card Fraud detection (CHAN; STOLFO, 1998) Pneumonia identification (PEREIRA et al., 2020) and so on.

Hierarchical Classification (HC) is as a type of classification problem where the output of the learning algorithm is defined over a particular class taxonomy. While the authors Wu, Zhang & Honavar (2005) consider the taxonomy as a structured concept tree hierarchy, Silla Jr & Freitas (2011) describe that the label taxonomy in a HC problem can also be structured as a Directed Acyclic Graph (DAG), which means that a certain label node have more than one parent node in the label path. Furthermore, a Hierarchical Multi-Label Classification (HMC) problem is a variant where the instances may be associated with multiple label paths at the same time. There are different types of hierarchical classification problems, mainly regarding the number of label paths per samples and the depth of the prediction in the label hierarchy. This variety of types of problems makes the hierarchical classification a challenging research area.

In the last years, in the hierarchical classification research area, a lot of studies and effort have been made in proposing new and different methods for classification such as in (FAGNI; SEBASTIANI, 2007; PUNERA; GHOSH, 2008; SILLA JR; FREITAS, 2009b; DIMITROVSKI et al., 2012; LIN et al., 2013; ROY et al., 2015; MCNAMARA et al., 2015; KHOWAJA; YAHYA; LEE, 2017; BONIFAZI; CAPOBIANCO; SERRANTI, 2018). However, there are other well-known issues in flat classification problems, such as class imbalanceness, that has not been addressed and studied for hierarchical problems so far.

1.1 Problem

Class imbalanceness is a problem where the total number of samples from some classes is far less than the total number of instances from other classes. Usually the classifiers are focused in the minimization of the global error rate and thus, when dealing with imbalance data, the algorithms tend to benefit the most frequent classes, also known as the Majority Classes. Most evaluation metrics are not appropriately affected by the incorrect classification of classes with few samples. However, depending on the problem, the main interest of the task could be on properly labeling these rare patterns. It is important no observe that the problem is not necessarily the imbalanceness, but the impact that the imbalanceness can cause in the learning phases of the classification algorithms.

A lot of researchers face these imbalanced class distribution issues, mostly when working with real world datasets such as medical image classification (PEREIRA et al., 2020), detection and classification of acoustic scenes and events (MESAROS et al., 2018), credit card fraud detection (KUMAR et al., 2015) and so on.

In order to deal with the imbalanceness problem in the flat classification context, i.e., in binary, multi-class and multi-label problems, several methods have been proposed in the literature such as cost-sensitive solutions, data resampling and hybrid techniques. According to Haixiang et al. (2017) the resampling techniques, which can be further sub-categorized into oversampling and undersampling, are the most common and widely used solutions, since they present, in general, the most promising results. While oversampling balances the dataset by creating new instances for the minority classes, undersampling is aimed at the removal of existent samples from the majority classes.

Although dataset imbalanceness is a well-known problem in the machine learning community, as shown in Branco, Torgo & Ribeiro (2016), there are few works in the literature studying this issue in the context of hierarchical classification (CESA-BIANCHI; RE; VALENTINI, 2012; CHEN; DUAN; HU, 2012; LI; JU; ZOU, 2016; NAIK; RANGWALA, 2016; XU; GENG, 2019). Even though, these studies do not directly address the imbalance problem with resampling methods.

1.2 Objectives

In this work, the overall objective is to analyze and propose methods to deal with imbalanceness in the hierarchical classification scenarios. We are concerned in how the imbalanceness in a dataset can affect the classification results and, in addition, how to pre-process the dataset by using resampling techniques in order to minimize the imbalance issues. In order to meet the general objective of this work, we outline the following specific objectives:

1. Propose novel resampling methods to deal with imbalanced datasets in the flat classification scenario.
2. Investigate the impacts of binary/multi-class and multi-label resampling methods on hierarchical datasets.
3. Propose metrics to measure the imbalanceness issues in the different types of hierarchical classification problems.
4. Propose techniques to deal with imbalanceness in hierarchical classification problems considering the different classification approaches.

5. Propose and investigate the use of the novel resampling measures and approaches in a real world hierarchical classification case study.

Objective 1 allows us to deeply investigate and understand the basics machine learning and the existing resampling methods used in flat classification scenarios. Objective 2 gives experimental baselines for further comparison with the novel hierarchical methods. Objective 3 concerns the formalization of the imbalanceness problem in the different types of hierarchical datasets, as well as formulas to assist the identification of this issue in this type of dataset. Objective 4 allows the most important contributions of this work, i.e., the proposal of different techniques to deal with the imbalanceness in the different hierarchical classification problems. Finally, Objective 5 allows us to investigate the impact of the contributions from this Thesis in a real world hierarchical classification problem.

1.3 Hypothesis Statements

Hypothesis 1. *The imbalanceness of a hierarchical dataset can be measure by using metrics that consider different local and global perspectives, i.e., perspectives that compute the number of samples per node, per parent node, per level and considering the hierarchy as a whole.*

Hypothesis 2. *The classification results for a hierarchical dataset may be improved by reducing its imbalanceness using the existing binary, multi-class and multi-label resampling approaches with global classification approaches.*

Hypothesis 3. *The classification results for a hierarchical dataset may be improved by using existing binary and multi-class resampling techniques to reduce its imbalanceness in local perspectives with local flat classification approaches.*

Hypothesis 4. *The classification results for a hierarchical dataset may be improved by reducing its imbalanceness using novel global resampling techniques developed to consider the labels relationships in the hierarchy.*

All Hypothesis are somehow related. In order to verify Hypothesis 2, 3, and 4 it is necessary to have specific metrics that measure the degree of imbalanceness in hierarchical classification problems, which is analyzed in Hypothesis 1. Hence, we hypothesize that it is possible to use techniques to pre-process the hierarchical classification dataset in order to reduce its imbalanceness, and this reduction will improve the hierarchical classification overall result.

A fundamental aspect of all the hypothesis is that we have to test the proposed techniques in different hierarchical classification datasets, evaluating their effectiveness across different application domains.

1.4 Outcomes, Contributions and Repercussions

The development of this Thesis lead to outcomes for the research community and the society. In addition to the expected scientific contributions, which were already published or are currently under evaluation in peer-reviewed venues, some technical contributions were also made freely available to the research community. Furthermore, one of the scientific contributions of this Thesis attracted the attention of the local, national and international media, given its potential social impact. In order to describe these outcomes in more detail, this section is subdivided into three subsections: (i) Scientific Contributions; (ii) Technical Contributions; and (iii) Social Impact and Media Recognition.

1.4.1 Scientific Contributions

Through this work we have contributed to the current understanding of dataset imbalanceness for hierarchical classification problems. We developed novel metrics and methods to tackle the imbalanceness issue in both flat and mainly in hierarchical classification contexts. Concretely, we also provided new datasets and empirical evidences of the advantages of using the proposed techniques to handle the class imbalanceness. These empirical evidences include statistical tests comparing state-of-the-art baselines to the methods presented in this work, which are experiments developed to validate the contributions of all the proposed metrics and techniques.

As almost all Chapters of this thesis generated research papers, they were published or submitted to peer-reviewed venues. In Table 1 we present the journal and conference papers that were directly developed during this Doctoral Research, while in Table 2 shows the journal and conference papers that were collaboratively developed. In this context, we make explicit what were the learning step and contribution of the paper to the project and the research community. The tables have information about the venue in which the paper was published (or currently under revision), the venue (conference or journal name), year of publication or submission and the impact factor (in case of journal) or h5-index (in case of conference) of the venue. It is important to mention that, even thought some papers from the tables are not described in the Chapters of this Thesis, they were important steps towards the construction of the knowledge path necessary for the execution of this Doctoral Research.

Table 1 – Papers developed during this Doctoral Research.

Paper	*Learning Step and Contribution*	*Venue*	*Impact*	*Reference*
A multi-label approach for the Tomek Link undersampling algorithm	A Novel Multi-Label Resampling	Neurocomputing	IF: 4.44	(PEREIRA; COSTA; SILLA JR, 2020)
Handling imbalanceness in hierarchical classification problems using local classifiers approaches	Second proposal towards Hierarchical Classification of Imbalanced Data	Data Mining and Knowledge Discovery	IF: 2.63	*Currently Under the Third Round of Reviews*
Towards Hierarchical Classification of Imbalance Data	Third proposal towards Hierarchical Classification of Imbalanced Data (Random Resampling Methods)	Information Sciences	IF: 5.91	*Currently Under the First Round of Reviews*
HSMOTE: Dealing with imbalanceness in hierarchical classification problems using synthetic samples	Fourth proposal towards Hierarchical Classification of Imbalanced Data	Knowledge Based Systems	IF: 5.92	*Currently Under the Second Round of Reviews*
COVID-19 identification in chest X-ray images on flat and hierarchical classification scenarios	Case Study Analysis of Thesis Application	Computer Methods and Programs in Biomedicine	IF: 3.63	(PEREIRA et al., 2020)
Using simplified chords sequences to classify songs genres	Basics of Machine Learning Application	IEEE International Conference on Multimedia and Expo	H5i: 30	(PEREIRA; SILLA JR, 2017)
Representation Learning vs. Handcrafted Features for Music Genre Classification	Advanced Machine Learning Application	International Joint Conference on Neural Networks	H5i: 46	(PEREIRA et al., 2019)
Dealing with Imbalanceness in Hierarchical Multi-Label Datasets Using Multi-Label Resampling Techniques	First proposal towards Hierarchical Classification of Imbalanced Data	IEEE International Conference on Tools with Artificial Intelligence	H5i: 19	(PEREIRA; COSTA; SILLA JR, 2018)

Table 2 – Papers that were collaboratively developed during this Doctoral Research.

Paper	*Learning Step and Contribution*	*Venue*	*Impact*	*Reference*
A multimodal approach for multi-label movie genre classification	Basics of Multi-Label Resampling	Multimedia Tools and Application	IF: 2.31	(MANGOLIN et al., 2020)
A Resampling Approach for Imbalanceness on Music Genre Classification using Spectrograms	Binary and Multi-Class Resampling	The Florida Artificial Intelligence Research Society Conference	H5i: 16	(VALERIO et al., 2018)

1.4.2 Technical Contributions

Most of the codes and scripts used in the experiments performed in this Thesis are freely available for download. Considering that there are few freely available implementations to deal with resampling or hierarchical classification in the literature, this can be considered an important technical contribution of this work.

In order to understand the technicality of the state-of-the-art resampling approaches, during the preliminary studies of this Thesis, a multi-label imbalance learning framework, named Imb-Mulan[1], was developed. This framework is an extension to the well-known Mulan framework (TSOUMAKAS et al., 2011) and contains the state-of-the-art multi-label resampling approaches, including the one proposed in this Thesis.

As three novel hierarchical resampling algorithms were proposed in this Thesis, in order to provide machine learning researchers with implementations of these techniques, we proposed the hierarchical-imblearn[2], an Open-source Hierarchical Imbalanced Learning Framework.

Beyond the frameworks, we proposed many novel datasets in order to investigate the effects of the proposed classification and resampling approaches. Since the search for datasets is always a challenging task for researchers, specially when dealing with hierarchically organized data, these datasets can also be considered an important technical contribution of this thesis. In Table 3 we present a brief summary of the datasets proposed in this work and where to find them.

[1] https://github.com/rodolfomp123/imb-mulan
[2] https://github.com/rodolfomp123/hierarchical-imblearn

Table 3 – Novel datasets proposed in this Thesis.

Dataset	*Type of Classification*	*Domain*	*Link for Download*
RYDLS-20	Single-Label and Hierarchical	Medical	https://bit.ly/rydls-20
P-TMDB	Multi-Label	Movie	https://bit.ly/p-tmdb
FMA90k		Music	https://bit.ly/fma-90k
FMA-SL	Single-Label		https://bit.ly/fma-sl
BRMD			https://bit.ly/brmdb
Hier-CAL500 Hier-Emotions Hier-FMA-MFCC Hier-FMA-SL-LBP Hier-FMA-SL-SSD	Hierarchical	Music	https://bit.ly/h-imb-db
Hier-Enron		Text	
Hier-Birds		Animal	

1.4.3 Social and Media Impact

This Doctoral Research was partially developed during the breakthrough of the COVID-19 and its pandemic. Considering this context, in order to give a case study of the methods proposed in this work in a real world scenario, we developed a method to identify COVID-19 and other pneumonia pathogens in Chest X-Ray (CXR) images. Our work was one of the first studies published in the literature (PEREIRA et al., 2020) that proposed a method to deal with the COVID-19 identification task considering factors such as data imbalance and the hierarchical relationship between the pneumonia pathogens. Given the importance of the topic and the timely publication of this contribution, it has attracted the attention of researchers, society, and media vehicles.

In order to give an overview of the repercussions, in Table 4 we present a summary of the main reports concerning our work in the TV, Radio and Magazines. It is worth mentioning that the report from the *Agencia Brasil* was republished by over a 100 online news agency websites, including *Valor Econômico*, *Época* and *Istoé*.

1.5 Text Organization

The remaining of this work is organized as follows: Chapter 2 presents a literature review concerning machine learning, being subdivided into Single, Multi-Label and Hierarchical Classification, while Chapter 3 describes the imbalanceness issue as well as related work. Chapter 4 shows the Multi-Label Tomek Link, the first contribution of this

Table 4 – Summary of the main media reports concerning our work.

Report Name	*Repercussion Level*	*Venue*	*Media Type*	*Reference*
AI Assist for Spotting COVID-19 in X-Rays	International	Physics	Magazine	(WRIGHT, 2020)
Coronavírus: Pesquisadores do Paraná criam método para diagnóstico com raio-x	National	Gazeta do Povo	Magazine	(FELIX, 2020)
Estudo possibilita diagnóstico da COVID-19 via Raio-X	National	CAPES	Youtube Channel	(JELLER, 2020)
Estudo brasileiro identifica COVID-19 por Raio-X com 90% de eficácia	National	CNN Brasil	TV	(BRONZE, 2020)
Novo Sistema de Diagnóstico por Raio-X	National	JovemPan News	Radio	(PEREIRA, 2020)
Radiografia Inteligente	National	SuperAcesso	Magazine	(SILLA JR, 2020d)
Universidades desenvolvem apoio a diagnóstico de covid-19 com raio-x	National	Agencia Brasil*	Magazine	(VALENTE, 2020)
COVID-19: estudo desenvolve diagnóstico por Raio-X	State	CBN Curitiba	Radio	(SILLA JR, 2020a)
Estudo com Inteligência Artificial para diagnóstico da COVID-19 em Raio-X	State	Transamérica	Radio	(SILLA JR, 2020b)
Imagens de Raio-X no diagnóstico da COVID-19	State	RPC Parana	TV	(SILLA JR, 2020c)
Pesquisadores do Paraná estudam método para diagnosticar COVID-19 com Raio-X	Local	CBN Maringá	Radio	(COSTA, 2020)
Pesquisadores maringaenses fazem estudo que pode ajudar a identificar o COVID-19	Local	RIC Maringá	TV	(GERHARD, 2020)
Pesquisadores se unem para agilizar diagnóstico da COVID-19	Local	RPC Maringá	TV	(GUZZONI, 2020)

* This report was republished by over a 100 online news agency websites.

work in terms of resampling methods. Chapter 5 presents the main baseline approach for this work, which is grounded in the use of binary/multi-class and multi-label resampling algorithms in the different hierarchical classification problems. Chapter 6 presents the proposal of two novel metrics to measure the imbalanceness in hierarchical datasets and a label path conversion method in order to deal with this imbalanceness by using multi-label resampling techniques. In Chapter 7 we present an approach to deal with imbalanceness in hierarchical datasets on local classifiers perspectives, using the binary/multi-class resampling algorithms, as well as imbalanceness measures for each one of these local perspectives. Chapter 8 shows the first hierarchical resampling solutions able to handle the data as whole, the Hierarchical Random Overesampling (HROS) and Undersampling (HRUS) algorithms. In Chapter 9 we present the Hierarchical Synthetic Oversampling Technique (HSMOTE). Chapter 10 shows a case study of imbalanceness in a real world problem, which is the COVID-19 identification in chest x-ray images. Finally, in Chapter 11, we describe the concluding remarks and future works of this Doctoral Research.

Chapter 2 Machine Learning

The development of machines able to learn by experience has been, for a long time, objective of technical and philosophical discussion. The technical aspect of the discussion, in specific, has receive a lot of attention since the beginning of the electronic computers era.

Given that there are problems that cannot be resolved by classic programming methods, because it is not possible to algorithmically model these problems, reliable machine learning systems becomes an important topic. An example of this is the classification of musical genres, in which given a series of previous identified songs' patterns, a learning system is able to recognize unseen songs based on its patterns. The solution of this kind of task is to build a system able to learn from previous experiences.

According to Mitchell (1997), a computer program is said learning from a experience E with respect of a task class T and performance measure P, if its performance in the tasks in T, according to the measure P, improves with the experience E.

There is an overlap in the literature concerning the terms "Machine Learning", "Pattern Recognition" and "Data Mining". These terms are often used to refer to a same Artificial Intelligence (AI) subarea. However, there are some differences between these areas. While the term Pattern Recognition is related to statistical analysis of data patterns (FUKUNAGA, 2013), Data Mining is more related to the data preparation and the kinds of knowledge discovery processes (ZHANG; ZHANG; YANG, 2003). Moreover, according to Monard & Baranauskas (2003), the Machine Learning is a subarea of AI which its main objective is to develop computational techniques over learning and designing intelligent systems capable to obtain knowledge in an automatic way. Within the Machine Learning there is a variety of learning paradigms. According to Monard & Baranauskas (2003), most of the existing approaches can be fitted in one of these paradigms:

- Symbolic: The learning system attempt to build symbolic representations of a concept by analysing existing examples. In general, the symbolic representations are expressed as logical expression, rules, decision trees or web semantic.
- Statistic: The learning system uses statistical models to find a good approximation of the induced concept. One of the most well-known statistical methods is the

Bayesian learning, which uses a probabilistic model based on previous knowledge of the problem.

- Example-based: This type of system uses previous examples of well-known classes to classify a new example based on their similarity. The most common algorithms in this paradigm are the Case-Based Reasoning (CBR) and the k-Nearest Neighbours (kNN).
- Connectionist: This kind of learning system is based on highly interconnected units, such as Neural Networks.
- Genetic: The learning system consists of a population of classification elements competing to make the prediction. The elements that present better performances are maintained and the weak ones are discarded. The main idea is that the "strong" elements are maintained and may proliferate, producing promising variations of itselves.

There are a lot of machine learning algorithms and each one can perform differently depending on the problem. Thus, there is no single algorithm capable of obtaining the best result in all situations, and it is important to understand the power and limitations of each one of them. This concept was handled by Wolpert & Macready (1997) and became widely known as the "No Free Lunch Theorem".

According to Monard & Baranauskas (2003), induction is the form of logical inference that makes it possible to obtain generic conclusions through a set of examples and inductive learning can be divided between supervised and unsupervised. Usually the examples used in inductive learning are composed of a feature vector containing the attributes of each example.

Table 5 shows an example of feature vector with three samples, each one of them containing three attributes (nominal an numeric) and their labels (A, B and C).

Table 5 – Feature vector example.

#Sample	**Attribute 1**	**Attribute 2**	**Attribute 3**	**Label**
1	No	Big	200	A
2	Yes	Medium	100	B
3	No	Small	150	C

In supervised learning the algorithm, also known as the inductor, receives a set of training samples. Usually these samples are composed of a feature vector and the label/class associated to them. In this case, the objective of the inductor algorithm is to build a classifier, based on these labeled samples, able to label a new unlabeled and unseen sample. In synthesis, this kind of learning method is used when the labels are already known by the user.

In unsupervised learning, the samples are usually unlabeled feature vectors. Thus, without knowing the label of each sample, the inductor is concerned to group the samples in some way, forming clusters or clusters. In this case, since the samples are not labeled, it is necessary to make an analysis to identify the meaning of each group based on the problem context. In synthesis, this type of learning is used when the user is not aware of possibly existing labels.

Furthermore, semi-supervised learning can be considered as a halfway between supervised and unsupervised learning. Besides unlabeled data, the learner is provided with some supervision information, but not necessarily for all examples in the dataset (CHAPELLE; SCHOLKOPF; ZIEN, 2009).

Figure 1 presents the hierarchy of inductive learning, in which in the second level we have the separation between supervised, unsupervised and semi-supervised models, and in the third level we have some examples of models commonly used. As the focus of this Thesis is the "Classification" field, is it circled with a dashed red square in Figure 1. In the next sections, we will present some details concerning the Classification research field.

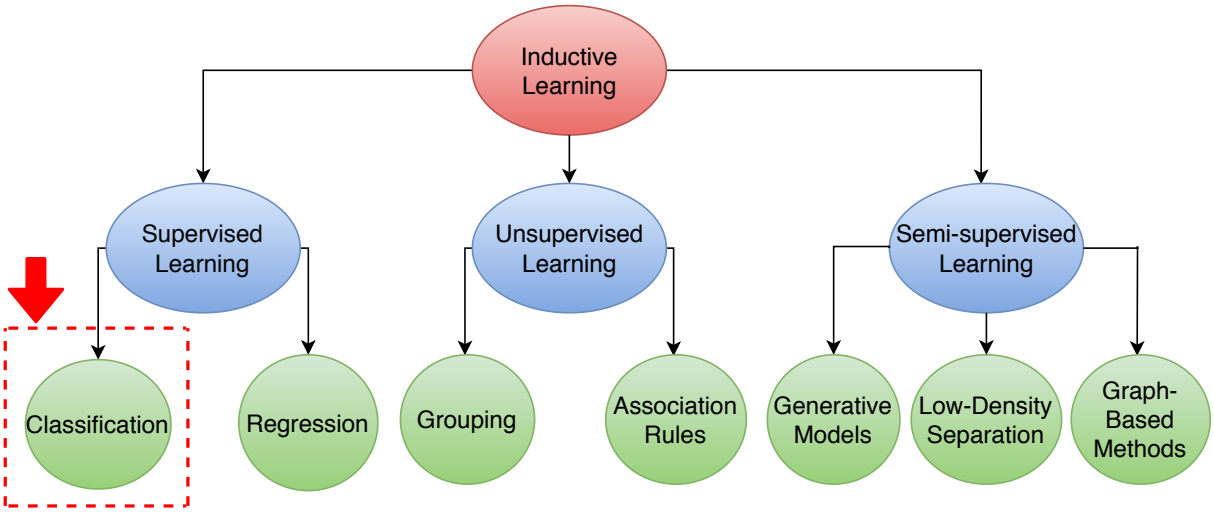


Figure 1 – Inductive Learning Hierarchy.

2.1 Single-Label Classification

We refer as single-label classification problem the ones where the samples are associated with only one single class. If there is only two possible class labels in the problem it is called a binary classification task, otherwise the problem is know as a multi-class task. Since the focus of this work is to deal with prediction tasks, in the following we briefly explain some of the single-label predictive techniques that have been widely used in the machine learning community.

2.1.1 Bayesian Classifiers

Bayesian classifiers use the Bayesian Decision Theory, presented in Duda, Hart & Stork (2001), in which the class conditional probability densities and prior probabilities are estimated from a training set. The classification is achieved through the assignment of each sample to the class of a maximum posteriori probability given by Formulas 2.1 and 2.2, where y specifies the class labels, and x is a given sample.

$$Y_{MAP} = argmax_i P(y_i|x) \tag{2.1}$$

$$P(y_i|x) = \frac{P(y_i)P(x|y_i)}{P(x)} \tag{2.2}$$

The Naive Bayes classifier assumes that the features in each class are independent and normally distributed, thus, the product rule can be applied for the estimation of the conditional probabilities.

2.1.2 K-Nearest Neighbors

The K-Nearest Neighbor (kNN) is an algorithm that uses the labels of the k nearest samples provided by the training set to predict the label of a testing sample.

According to Tan, Steinbach & Kumar (2005) this kind of algorithm is specially focused in two points: (1) A metric to calculate the distance between the samples and (2) a value for k, which is the number of nearest neighbors to search for. In general, the distance metric is calculaed with the Euclidean Distance, nevertheless, other measures can be applied as shown in Kotsifakos et al. (2013).

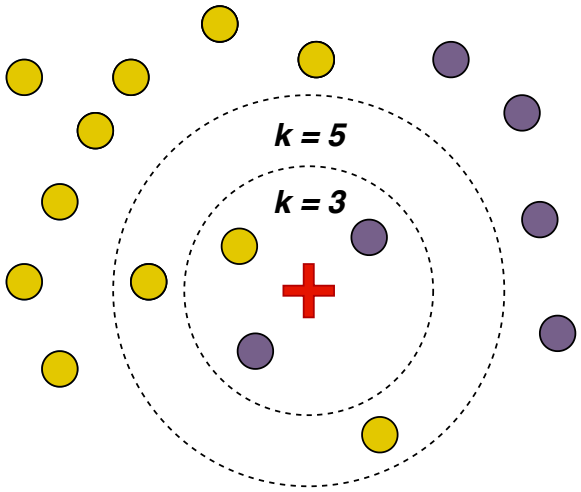


Figure 2 – Example of kNN classification algorithm.

Figure 2 presents an example of the application of k-NN technique, with $k = 3$ and 5, to classify a sample represented as a "red cross" in the dataset. The dash dots shows which instances from the dataset would be consider to choose the label.

2.1.3 Decision Trees

A decision tree is an approach that generates a visual model of the learning procedure, where each non-leaf node is labeled with an attribute and each leaf node contains a decision. One of the most famous decision trees is the C4.5, proposed in Quinlan (1993). The C4.5 algorithm finds the attribute that more effectively divide the samples into subsets tending to a category or another. The standardized information gain is the partitioning criterion and the attribute with the highest information gain is chosen to make the decision. The C4.5 algorithm then repeats the previous step in the smaller partitions and so on, until there is no data left to divide.

Figure 3 shows an example of classification using a Decision Tree for the problem of tax evasion identification. We may observe that the training dataset is composed by 10 samples and the features values are used as internal nodes to generate the Decision Tree model.

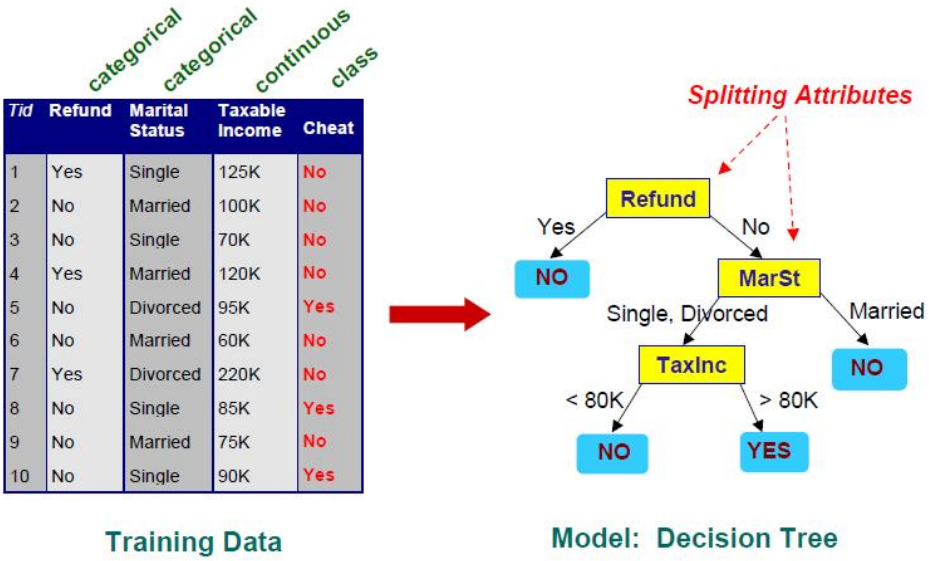


Figure 3 – Example of Decision Tree classification algorithm (TAN; STEINBACH; KUMAR, 2005).

Another well used decision tree technique is the Random Forest (BREIMAN, 2001). This method is a combination of decision trees in a way that each tree is dependent of values from a random vector with the same distribution for all trees in the forest. As the number of trees in the forest increases, the generalization error tends to converge to a limit.

Figure 4 presents a classification schema for the Random Forest algorithm. We may observe that, for this example, four random trees are generated in order to give the final decision for the instance X.

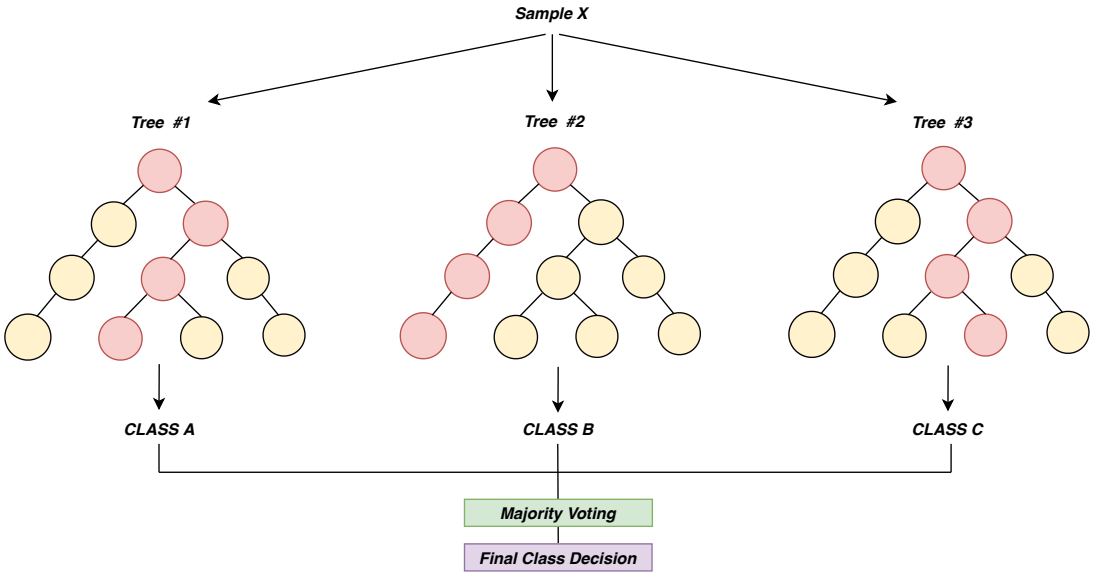


Figure 4 – Example of Random Forest classification schema.

2.1.4 Support Vector Machines

The Support Vector Machines (SVM) technique, first proposed in (BOSER; GUYON; VAPNIK, 1992) and then published in (CRISTIANINI; SHAWE-TAYLOR, 2000), is a binary classifier based on the maximum-margin principle. This technique has solid bases in the statistical learning theory proposed by (VAPNIK; CHERVONENKIS, 1971).

The SVM algorithm uses labeled samples from a training set to find the optimal hyperplane that best separates the samples from two classes. The hyperplane is calculated so that the distance between the nearest samples from each class and the hyperplane is maximized. Such samples are referred to as supporting vectors and represent the closest samples to the hyperplane with labels.

According to (MEYER, 2004), a SVM classifier may be roughly sketched as follows:

- Class separation: The optimal separating hyperplane between the two classes, which maximizes the margin between the classes' closest samples.
- Overlapping classes: Data points on the "wrong" side of the discriminant margin are weighted down to reduce their influence (known as "soft margin").
- Nonlinearity: If a linear separator is not found, some samples are projected into an higher-dimensional space where these samples effectively become linearly separable.

- Problem solution: The task is formulated as a quadratic optimization problem.

Figure 5 shows an example of SVM classifier used to classify two classes of samples. We may observe that there is a margin in the hyperplane, in which it would correctly separate the samples.

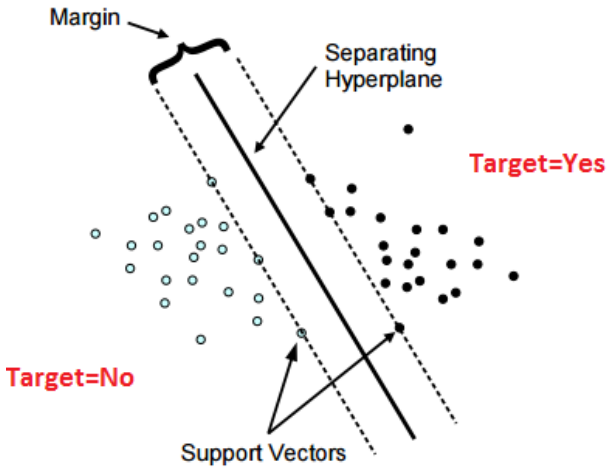


Figure 5 – Example of SVM classification algorithm (MEYER, 2004).

2.2 Multi-Label Classification

Traditional classification problems are concerned with learning from a set of instances that are associated to a single-label l of a disjoint set of labels L, where the cardinality of the set is greater than 1. If the cardinality of the set of labels is equal to 2 ($|L| = 2$), then the learning problem is called binary classification. If the cardinality of the set is greater than 2 ($|L| > 2$), then it is a multi-class classification problem. In multi-label classification each instance will be associated to a set of labels $Y \subseteq L$ instead of a single-label $l \in L$ (TSOUMAKAS; KATAKIS, 2006).

In the next subsection we present fundamental concepts concerning the multi-label classification problem, which includes database indicators, main classification algorithms, and evaluation metrics.

2.2.1 Database Indicators

According to Zhang & Zhou (2014) to characterize the properties of any Multi-Label Database (MLD), there are several useful multi-label indicators that can be utilized. Let D be the MLD, m the total number of labels and Y the set of Labels with Y_i being the i-th label of the set, the most natural way to measure the degree of "multi-labeledness" are:

- Label Cardinality: The average number of labels per example.

$$LCard(D) = \frac{1}{m}\sum_{i=1}^{m}|Y_i| \tag{2.3}$$

- Label Density: Normalization of label cardinality by the number of possible labels in the label space.

$$LDen(D) = \frac{1}{|Y|} \times LCard(D) \tag{2.4}$$

- Label Diversity: The number of distinct label sets appeared in the dataset.

$$LDiv(D) = |\{Y \mid \exists\, x : (x, Y) \in D\}| \tag{2.5}$$

- Proportion of Label Diversity: Label diversity normalized by the number of examples, indicating the proportion of distinct label sets.

$$PLDen(D) = \frac{1}{|D|} \times LDiv(D) \tag{2.6}$$

2.2.2 Classification Algorithms

According to (TSOUMAKAS; KATAKIS, 2006), we can group the existing methods for multi-label classification into two main categories: (1) problem transformation methods; and (2) algorithm adaptation methods.

The problem transformation methods are those in which the multi-label classification problem is transformed either into one or more single-label classification or regression problems, for both of which there exists a huge bibliography of learning algorithms.

On the other hand, we call algorithm adaptation methods, those techniques that extend specific learning algorithms in order to handle multi-label data directly.

Table 6 presents the categorization of the multi-label classification methods, which will be described in the following subsections.

Binary Relevance

The Binary Relevance (BR) technique builds independent binary classifiers for each label (λ_i). Each classifier maps the original dataset to a single binary label with values λ_i, $\sim \lambda_i$. The classification of a new instance is given by the concatenation of the labels λ_i that are produced by the classifiers (BRINKER; FÜRNKRANZ; HÜLLERMEIER, 2006).

Label Powerset

The Label Powerset (LPS) was first introduced by Boutell et al. (2004). This method transforms the multi-label dataset into a multi-class dataset by using the labelset

Table 6 – Categorization of multi-label classification methods.

Problem Transformations	Binary Relevance
	Classifier Chain
	Calibrated Label Ranking
	HOMER
	Label Powerset
	RAkEL
Algorithms Adaptation	ADABoost.MH
	BRkNN
	MLkNN

of each instance as class identifier. Any multi-class classifier can be used, transforming back the predicted class into a labelset.

Classifier Chain

According to Read et al. (2009), in the Classifier Chain (CCH), one classifier h_i is trained for each label similarly to a scoring function f_i. Given a new instance x to be classified, the model h_1 predicts y_1, i.e., the relevance of λ_1 for x. Then, another classifier h_2 predicts the relevance of λ_2, taking x plus the predicted value $y_1 \in 0, 1$ as an input. Proceeding in this way, h_i predicts y_i using $y_1 ... y_{i-1}$ as additional input information.

Calibrated Label Ranking

The Calibrated Label Ranking (CLR) was originally proposed in Fürnkranz et al. (2008), and its main idea is to add an additional label to the original label set which is interpreted as a "neutral element". This label calibrates a ranking by splitting it into a positive and a negative part. By extending conventional label ranking approaches, this algorithm provides a means to represent and learn bipartite partitions of alternatives and, thereby, combines multi-class classification and label ranking.

Hierarchy of Multi-label Classifiers

According to Tsoumakas, Katakis & Vlahavas (2008), the Hierarchy of Multi-label Classifiers (HOMER) follows the divide-and-conquer paradigm of algorithm design. The main idea is the transformation of a multi-label classification task with a large set of labels L into a tree-shaped hierarchy of simpler multi-label classification tasks, each one dealing with a small number $k << |L|$ of labels.

Random k-LabelSets

In the Random k-LabelSets (RAkEL) algorithm, first introduced by Tsoumakas,

Katakis & Vlahavas (2011), each ensemble member constructs an LPS classifier based on a randomly chosen subset of k labels. These subsets are referred to as k-labelsets. The classification of a new instance is achieved by thresholding the average of the binary decisions of each model for each label.

kNN-Based Algorithms

The Multi-Label k-Nearest Neighbors (MLkNN) (ZHANG; ZHI-HUA, 2007) is an adaptation of the kNN lazy learning algorithm for multi-label data. In essence, MLkNN uses the kNN algorithm independently for each label l: It finds the k nearest examples to the test instance and considers those that are labeled at least with l as positive and the rest as negative.

On the other hand, the Binary Relevance k-Nearest Neighbors (BRkNN) determines the k-nearest neighbors of a query and calculates the confidences of each label based on the label sets of the neighbor queries. In this algorithm, the neighbors are determined by the Problem Transformation of Binary Relevance (SPYROMITROS; TSOUMAKAS; VLAHAVAS, 2008).

2.2.3 Evaluation Measures

The evaluation of methods that learn from multi-label data requires metrics that differ from those employed for single-label data. According to Madjarov et al. (2012), the multi-label evaluation metrics can generally be categorized into three groups: Example-Based, Ranking-Based and Label-Based metrics. Figure 6 presents a categorization of the evaluation measures used in this work.

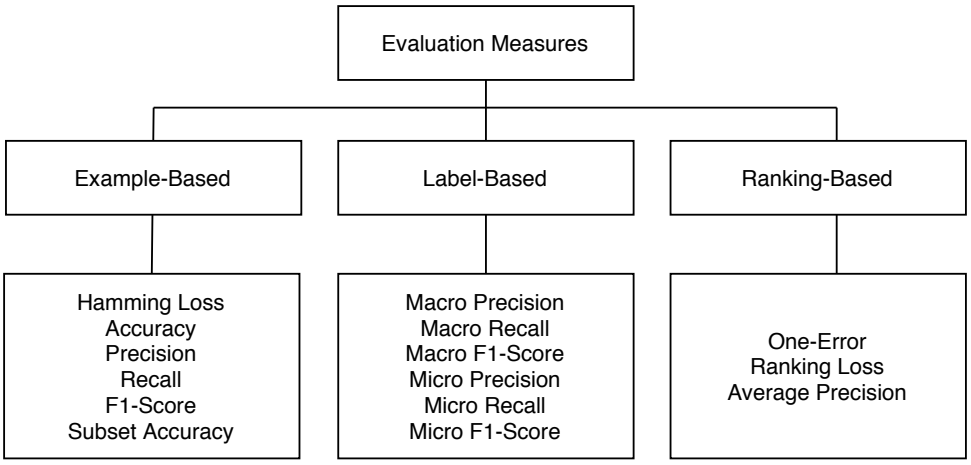


Figure 6 – Categorization of multi-label learning evaluation measures (adapted from Madjarov et al. (2012)).

For the definition of the Example-Based and Ranking-Based metrics let $(x_i, Y_i), i =$

$1..m$ be a set of instances of a dataset, where $Y_i \in L$ is the set of labels associated with an instance and $L = \lambda_j : j = 1..q$ is the set of all labels. Given an instance x_i, the resulting set of labels predicted by a multi-label classifier is denoted by Z_i.

2.2.3.1 Example-Based Metrics

The Example-based measures evaluate the performance on each test example separately, and then return the mean value across the test set.

- Subset Accuracy: The ratio of labels predicted for a sample that exactly match with the corresponding set of labels.

$$\frac{1}{m}\sum_{i=1}^{m}|Y_i = Z_i| \tag{2.7}$$

- Accuracy: The proportion of label values correctly classified of the total number (predicted and actual) of labels for that instance averaged over all instances.

$$\frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{2.8}$$

- Precision: The proportion of labels correctly classified of the predicted positive labels, averaged over all instances.

$$\frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap Z_i|}{|Z_i|} \tag{2.9}$$

- Recall: The fraction of predicted correct labels of the actual labels.

$$\frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap Z_i|}{|Y_i|} \tag{2.10}$$

- F-Measure: Harmonic mean that combines Precision and Recall.

$$\frac{(\beta^2+1)Precision \times Recall}{\beta^2 Precision + Recall} \tag{2.11}$$

2.2.3.2 Ranking-Based Metrics

Compare the predicted ranking of the labels with the ground truth ranking.

- Average Precision: Evaluates the average fraction of labels ranked above a particular label $\lambda \in Y_i$ which actually are in Y_i.

$$\frac{1}{m}\sum_{i=1}^{m}\frac{1}{|Y_i|}\sum_{\lambda \in Y_i}^{|Y|}\frac{\left|\lambda' \in Y_i : r_i(\lambda') \leq r_i(\lambda)\right|}{r_i(\lambda)} \tag{2.12}$$

- Hamming Loss: The percentage of incorrect labels predicted in relation to the total number of labels.

$$\frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \Delta Z_i|}{M} \tag{2.13}$$

  where M is the total number of labels and Δ is the symmetric difference between two sets, which is, in set theory, equivalent to the XOR operator in Boolean logic.

- One-Error: Evaluates how many times the highest ranked label is not in the set of relevant labels of the instance. Let $r_i(\lambda)$ be the rank predicted for a label λ by the classification method, where for the most relevant label is given the rank 1 and the less relevant receives the rank M.

$$\frac{1}{m}\sum_{i=1}^{m}\delta(\arg\min r_i(\lambda)), \lambda \in L \tag{2.14}$$

2.2.3.3 Label-Based Metrics

These measures evaluate the performance on each class label separately, and then return the macro or micro-averaged value across all class labels. The measures are based on a generalization of single label measures for many classes C_i where tp_i, fp_i, fn_i, and tn_i are true positive, false positive, false negative and true negative counts for C_i respectively. According to Sokolova & Lapalme (2009), macro-averaging treats all classes equally while micro-averaging favors classes with more samples.

- Micro-Averaged Precision: Agreement of the data class labels with those of a classifiers if calculated from sums of per-text decisions.

$$\frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l} tp_i + fp_i} \tag{2.15}$$

- Micro-Averaged Recall: Effectiveness of a classifier to identify class labels if calculated from sums of per-text decisions.

$$\frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l} tp_i + fn_i} \tag{2.16}$$

- Micro-Averaged F-Measure: Relations between data's positive labels and those given by a classifier based on sums of per-text decisions.

$$\frac{(\beta^2+1)\mu Precision \times \mu Recall}{\beta^2 \mu Precision + \mu Recall} \tag{2.17}$$

- Macro-Averaged Precision: An average per-class agreement of the data class labels with those of a classifiers.

$$\frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i+fp_i}}{l} \tag{2.18}$$

- Macro-Averaged Recall: An average per-class effectiveness of a classifier to identify class labels.

$$\frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i}}{l} \tag{2.19}$$

- Macro-Averaged F-Measure: Relations between data's positive labels and those given by a classifier based on a per-class average.

$$\frac{(\beta^2 + 1) MPrecision \times MRecall}{\beta^2 MPrecision + MRecall} \tag{2.20}$$

- Area Under the ROC Curve (AUROC): The Receiver Operating Characteristics (ROC) describes the classifiers' performance across the entire range of error costs by plotting the true positive rate (TPR) and the false positive rate (FPR) for each possible threshold. Furthermore, the Area Under the Curve (AUC) metric is given by computing the exact area under the ROC curve. In a multi-label problem, the TPs and FPs can be calculated individually by label and then joined with a micro-averaged technique to calculate TPR and FPR. In mathematical terms we can define micro TPR and FPR as:

$$micro - TPR = \frac{\sum_{i=1}^{|L|} tp_i}{\sum_{i=1}^{|L|} p_i} \tag{2.21}$$

$$micro - FPR = \frac{\sum_{i=1}^{|L|} fp_i}{\sum_{i=1}^{|L|} n_i} \tag{2.22}$$

  where L is the set of labels, tp_i and fp_i are the number of True Positives and False Positives for label i, p_i is the number of instances with label i and n_i is the number of instances without label i.

- Area Under the Precision-Recall Curve (AUPRC): The AUPRC measure is computed as the area below the curve plotted with the precision (y-axis) and the recall values (x-axis) considering the different classifiers decision thresholds.

2.3 Hierarchical Classification

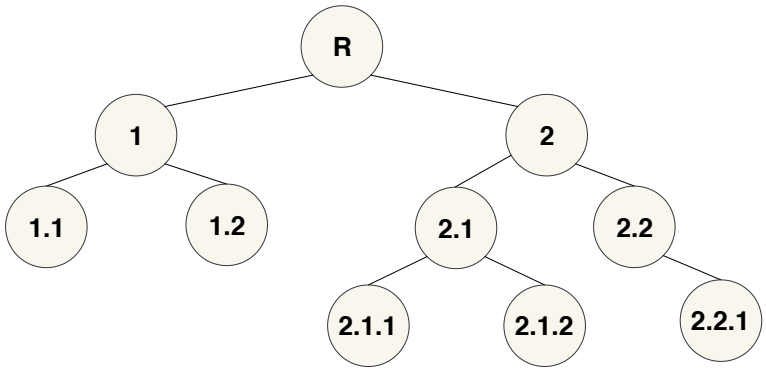
Hierarchical Classification (HC) can be seen as a particular type of classification problem, in which the output of the learning algorithm is defined over a specific class taxonomy. In Wu, Zhang & Honavar (2005) this taxonomy is considered a structured concept tree hierarchy defined over a partially order set $(C, \prec)$, where C is a finite set that enumerates all class concepts in the application domain, and the relation $\prec$ represents a "IS-A" relationship. In Silla Jr & Freitas (2011) is described that a HC label taxonomy can also form a Directed Acyclic Graph (DAG), meaning that a certain label node in

the hierarchy may be derived from more than one label path. By its turn, Hierarchical Multi-label Classification (HMC) problem is a variant where the instances may belong to multiple classes at the same time and these classes are organized in a hierarchy.

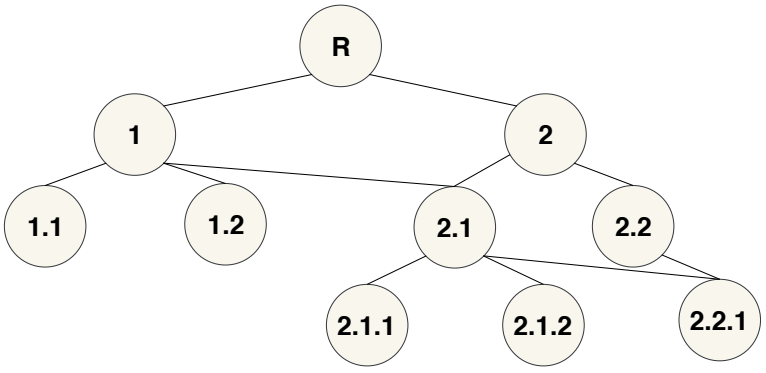
2.3.1 Definitions

According to Silla Jr & Freitas (2011) a hierarchical classification problem can be described as a 3-tuple $\langle \Upsilon, \Psi, \Phi \rangle$, where:

- Υ specifies the type of graph representing the hierarchical classes (nodes in the graph) and their interrelationships (edges in the graph). The possible values for this attribute are:
  - Tree (T): Indicates that the classes to be predicted are arranged into a tree structure (See Figure 7(a)).
  - DAG (D): Indicates that the classes to be predicted are arranged into a DAG (Direct Acyclic Graph) (See Figure 7(b)).



(a) Tree structure.



(b) DAG structure.

Figure 7 – Examples of hierarchical organization (adapted from Silla Jr & Freitas (2011)).

- Ψ indicates whether a data instance is allowed to have class labels associated with a single or multiple paths in the class hierarchy. This attribute can take one of the two values:
  - Single Path of Labels (SPL); or
  - Multiple Paths of Labels (MPL).
- Φ describes the label depth of the data instances:
  - Full Depth labeling (FD): Indicates that all instances have a full depth of labeling, i.e., every instance is labeled with classes at all levels, from the first level to the leaf level; or
  - Partial Depth labeling (PD): Indicates that at least one instance has a partial depth of labeling, i.e. the value of the class label at some level (typically the leaf level) is unknown.

Furthermore, according to Silla Jr & Freitas (2011) a hierarchical classification algorithm, can be described as a 4-tuple $\langle \Delta, \Xi, \Omega, \Theta \rangle$ where:

- Δ indicates whether or not the algorithm can predict labels in just one or multiple (more than one) different paths in the hierarchy. This attribute can take on two values, as follows:
  - Single Path Prediction (SPP): Indicates that the algorithm can assign to each data instance at most one path of predicted labels or
  - Multiple Path Prediction (MPP): Indicates that the algorithm can potentially assign to each data instance multiple paths of predicted labels.
- Ξ is the prediction depth of the algorithm. It can have two values:
  - Mandatory Leaf-Node Prediction (MLNP): Means the algorithm always assign leaf class(es); or
  - Non-mandatory Leaf-Node Prediction (NMLNP): Means the algorithm can assign classes at any level (including leaf classes).
- Ω is the taxonomy structure the algorithm can handle. It has two values:
  - Tree (T): Indicates that the classes to be predicted are arranged into a tree structure; or
  - DAG (D): Indicates that the classes to be predicted are arranged into a DAG (Direct Acyclic Graph).

- Θ is the categorization of the algorithm under the proposed taxonomy and has one of the following values:
  - Local Classifier per Node (LCN);
  - Local Classifier per Level (LCL);
  - Local Classifier per Parent Node (LCPN); or
  - Global Classifier (GC).

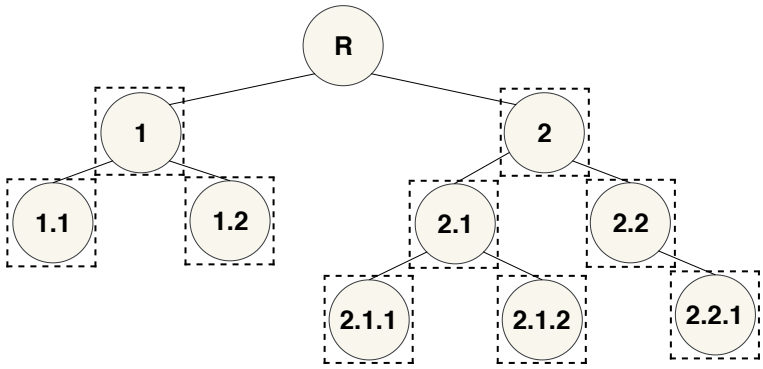
According to Silla Jr & Freitas (2011), specifically for the LCN approach, there are at least six different policies that can be used to define the set of positive and negative examples for training the binary classifiers per node:

- Exclusive (Figure 9(a)): Consider the true label node as positive and all others as negative;
- Less Exclusive (Figure 9(b)): Consider the true label node as positive and all nodes not descendants as negative (ignore the descendant nodes);
- Inclusive (Figure 9(c)): Consider the true label node and all descendants as positive and all other nodes as negatives, besides the ancestors, which are ignored;
- Less Inclusive (Figure 9(d)): Consider the true label and all descendants as positive and all others as negative;
- Siblings (Figure 9(e)): Consider the true label and all descendants as positive and the siblings and its descendants as negatives (ignore all others); and
- Exclusive Siblings (Figure 9(f)): Consider only the true label as positive and its sibling as negative (ignore all others).

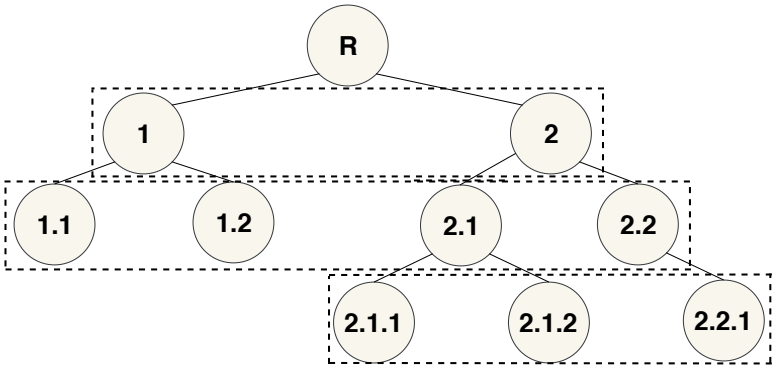
Moreover, for the LCPN approach, there are also two different policies in order to choose which samples will be used to train the multi-class classifier per parent node: Siblings or Exclusive Siblings. In this case, the difference is related to use all descendant samples or only the immediate child in order to build the multi-class classification model which is used to differ the nodes children.

2.3.2 Classification Algorithms

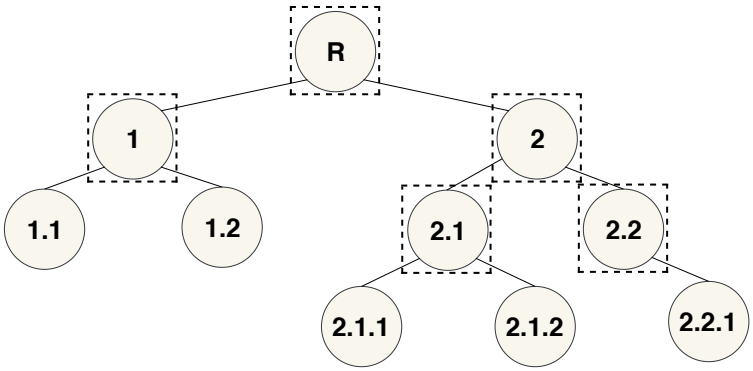
As stated in Silla Jr & Freitas (2011), there are three different strategies to perform hierarchical classification, which can be categorized into: Flat Classifiers; Local Classifiers; and Global Classifiers.



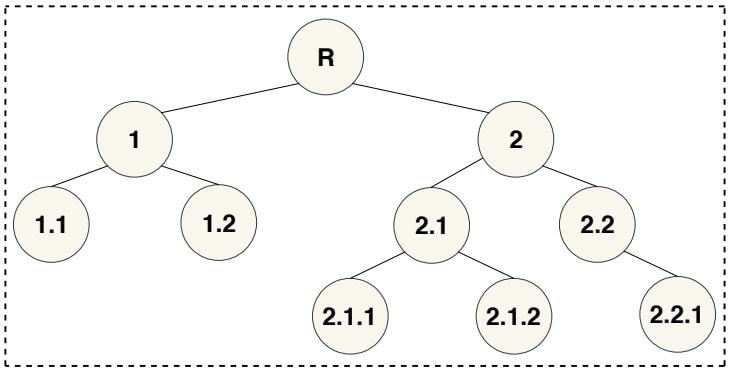
(a) Local Classifier per Node (LCN).



(b) Local Classifier per Level (LCL).

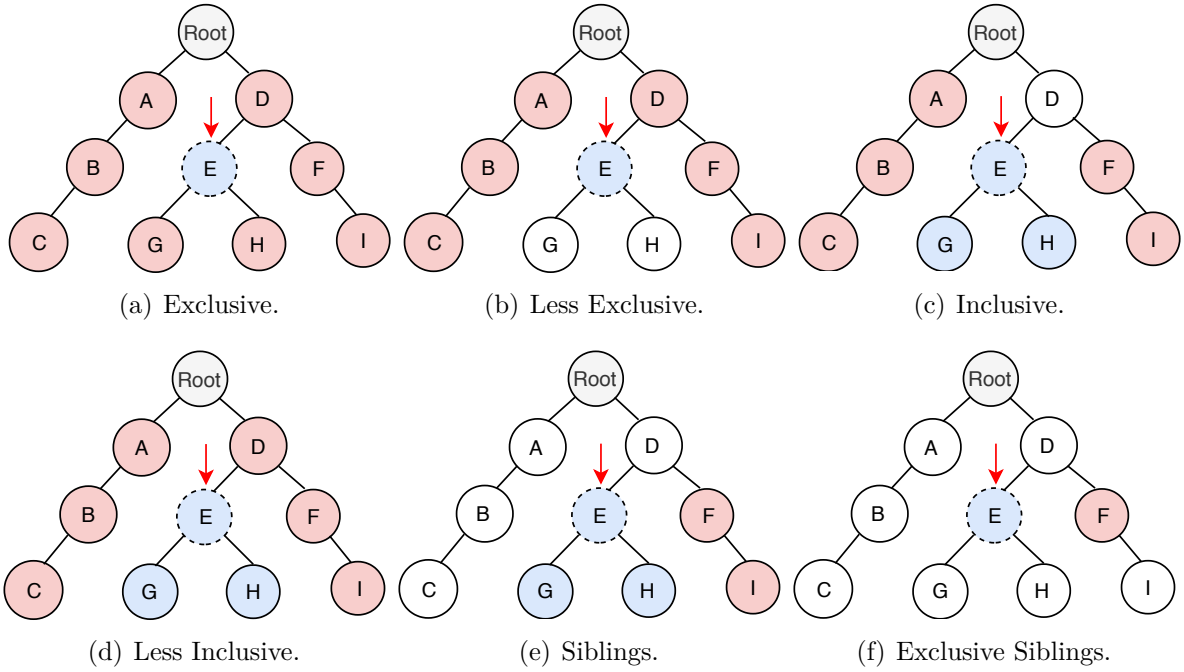


(c) Local Classifier per Parent Node (LCPN).



(d) Global Classifier (GC).

Figure 8 – Classifiers approaches - circles represent classes and each dashed rectangle encloses the classes predicted the classifier (adapted from Silla Jr & Freitas (2011)).



(a) Exclusive. (b) Less Exclusive. (c) Inclusive.

(d) Less Inclusive. (e) Siblings. (f) Exclusive Siblings.

Figure 9 – The positive/negative policies of the local classifiers per node approach. Blue nodes represent the positive labels, red the negative and white the labels that will not be used.

The flat classification approach is the simplest one and consists of completely ignoring the labels hierarchy, predicting only classes at the leaf nodes, using typical multi-class classification methods. This approach provides an indirect solution to the problem of hierarchical classification, since when a leaf class is assigned to an example, one can consider that all its ancestor classes are also implicitly assigned to that instance (SILLA JR; FREITAS, 2011).

On the other hand, the Local Classifiers (LC) approach can be sub-categorized based on how they use the local information and how they build their classifiers around it. There are three standard ways of using the local information:

- Local Classifier per Node (LCN): Consists of training one binary classifier for each node of the class hierarchy (except the root node). This type of classification was used in Eisner et al. (2005), Fagni & Sebastiani (2007), Ceci & Malerba (2007), Valentini (2009), Melo, Paulheim & Völker (2016), Nakano et al. (2017b), Zhang, Shah & Kakadiaris (2017). In Figure 8(a) we present an visual example of this type of classification;

- Local Classifier per Level (LCL): Consists of training one multi-class classifier for each level of the class hierarchy. This category of classsification was used in Cerri, Barros & Carvalho (2014), Bewley et al. (2015), Zeng et al. (2018), Alshamaa, Chehade & Honeine (2018). In Figure 8(b) we shown a visual example of this method;

- Local Classifier per Parent Node (LCPN): In this approach, for each parent node in the class hierarchy, a multi-class classifier (or a problem decomposition approach with binary classifiers) is trained to distinguish between its child nodes. This method was used in works such as Silla Jr & Freitas (2009b), Ramírez-Corona, Sucar & Morales (2016), Cerri et al. (2016), Nakano et al. (2017a), Costa et al. (2018). Figure 8(c) presents a visual example of this kind of technique.

In the Local Classifiers approach, conventional classification algorithms, such as Decision-Trees, Support Vector Machines and Neural Networks, are trained to produce an hierarchy of classifiers. On the other hand, the Global Classifiers approach (GC), also known as the big-bang approach, consists of using one global hierarchy classifier for hole the problem - See Figure 8(d) for example.

In the following we present a detailed description of some of the well-known approaches to perform HMC, such as: HMC Binary Relevance (LCN), HMC Label-Powerset (LCPN), HMC Cross-Training (LCL), Ant-Miner (GC), Association Rule-Based (GC), Decision Tree-Based (GC), Hierarchical Naive Bayes (GC) and Predictive Clustering Trees (GC).

HMC Binary Relevance

According to Cerri & Carvalho (2010), HMC Binary Relevance (HMC-BR) uses N classifiers in the induction phase, with N being the number of labels in the hierarchy, associating each classifier to one of the node labels. Then, the method trains these classifiers to solve N binary classification problems, using the One-Against-All approach. During the training phase, the label associated with node is considered as positive, while all the others are considered negatives.

Considering a HMC problem with five levels, which contains 2, 3, 4, 5 and 6 labels on each level, two classifiers will be trained in the first level, three in the second, four in the third, five in the fourth level and six in the fifth level. In this context, the j_{th} classifier considers the samples belonging to the j_{th} label as positives and all the other samples as negatives.

Algorithm 1 presents the pseudocode for the classification process of HMC-BR method. The pseudocode is recursive, and C is initially the set of the first level classes of the hierarchy.

Algorithm 1 HMC Binary Relevance (adapted from Cerri et al. (2015))

Input: x_i: Instance, C: Set of classes
Output: Classes

```
Classes ← empty set
for each c_j in C do
    if c_j not leaf node then then
        Children ← child classes of c_j in C
        Classes ← Classes ∪ {c_j} ∪ HMC-BR(x_i, Children)
    else
        Classes ← Classes ∪ {c_j}
    end if
end for
return Classes
```

HMC Label-Powerset

The HMC Label-Powerset (HMC-LP) approach, proposed by Cerri & Carvalho (2010), is a hierarchical adaptation of the multi-label classification method Label Powerset. HMC-LP combines all the labels assigned to the hierarchy, at a specific level, into a new and unique label.

Given two example of samples: One assigned to the labels A/D and A/E, and other sample assigned to B/F, B/G, C/H and C/I, in which these labels are hierarchically organized as in the left tree of Figure 10. The powerset combination of these labels would be a new hierarchical structure C_A/C_{DE} and C_{BC}/C_{FGHI}, respectively. In the example, C_{DE} is a new label formed by the combination of the labels D and E, while C_{FGHI} is a label formed by the combination of the labels F, G, H and I (CERRI; CARVALHO, 2010). Figure 10 presents the label combination process.

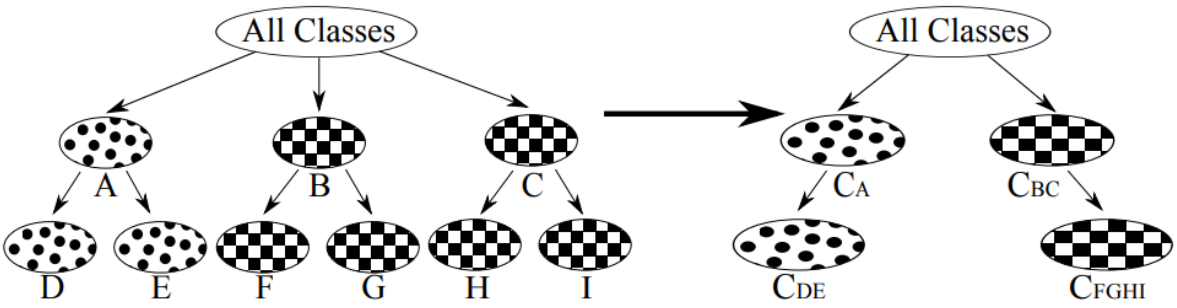


Figure 10 – Label combination process of the HMC-LP method (CERRI; CARVALHO, 2010).

According to Cerri & Carvalho (2010), after combining the labels, the original HMC problem is transformed into a hierarchical single-label problem, and a top-down approach can be employed, using multi-class classifiers for each level. In the end of the classification process, the original labels are recovered. Although the correlation between

the labels is considered, it can increase the total number of labels. The label combination procedure is presented in the Algorithm 2.

Algorithm 2 Label Combination of HMC-LP (adapted from (CERRI; CARVALHO, 2010))

Input: Y: Set of instances, C: Set of classes
Output: New classes

1: **for each** level j of the class hierarchy **do**
2: **for each** subset C_i of the set C, assigned to an sample y_i in level j **do**
3: Gets a new class $c_{i,j}$ for the example y_i from C_i
4: $NewClasses_{i,j} \leftarrow c_{i,j}$
5: **end for**
6: **end for**
7: **return** $NewClasses$

HMC Cross-Training

The HMC Cross-Training (HMC-CT), proposed by Cerri & Carvalho (2010), uses a label decomposition method. In this process the multi-label samples are decomposed into a set of single-label samples. For each sample, each possible label is considered as the positive label in sequence, using the multi-label data more than once during the training phase. For instance, if a given dataset has samples with labels A, B and C, when a classifier for the label A is trained, all the multi-label samples whose set of labels includes A become single-label samples for label A.

Figure 11 shows an example of the label decomposition process applied by the HMC-CT method in a dataset. It is important to observe that when a sample is assigned to more than one label, these labels are separated with slashes. Algorithm 3 shows the classification process of HMC-CT.

Algorithm 3 HMC Cross-Training (adapted from Cerri & Carvalho (2010))

Input: y: Sample, Cl: Set of classifiers
Output: Classes

1: Classes $\leftarrow$ empty set
2: **for each** classifier cl_i from Cl **do**
3: Predicts a class c_i for the example y using the classifier cl_i
4: **if** not the last hierarchical level then **then**
5: Gets the set Cl_i of children of cl_i trained with examples from class c_i
6: Classes $\leftarrow$ Classes $\cup$ $\{c_i\}$ $\cup$ Classify(y, Cl_i)
7: **else**
8: Classes $\leftarrow$ Classes $\cup$ c_i
9: **end if**
10: **end for**
11: **return** Classes

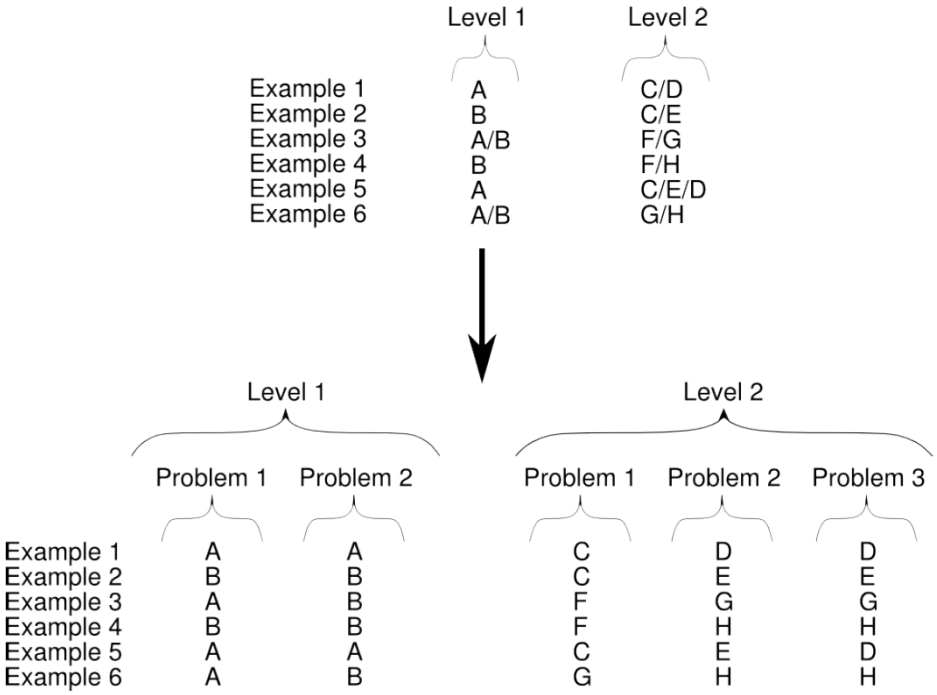


Figure 11 – Label decomposition process of the HMC-CT method (CERRI; CARVALHO, 2010).

Ant-Miner

In Otero, Freitas & Johnson (2009) is proposed the *h*Ant-Miner algorithm, the first hierarchical Ant-Miner classification approach. The *h*Ant-miner method is a type of swarm intelligence method based on the paradigm of ant colony optimization.

Algorithm 4 presents the pseudocode for the hAnt-miner algorithm. The pseudocode starts with an empty rule list and adds a new rule while the number of uncovered training examples is greater than a maximum parameter (specified by the user). In the iterations, the rules are created using an ant colony optimization procedure.

According to Otero, Freitas & Johnson (2009), *h*Ant-miner discovers a list of classification rules that can predict all classes from a class hierarchy, independently of their level, but has the limitation of not being able to cope with multi-label data. Moreover, in Otero, Freitas & Johnson (2010), the same authors presented an extension to the *h*Ant-miner algorithm, which overcomes this limitation.

Association Rule-Based

In Wang, Zhou & He (2001) an association rule mining algorithm is modified in order to deal with hierarchical document categorization. The main adaptation was to make the algorithm able to work with a set of labels instead of only a single label.

According to Wang, Zhou & He (2001), there are four steps to build the classifier.

Algorithm 4 *h*Ant-miner (adapted from Otero, Freitas & Johnson (2009))

Inputs: Training Examples
Output: Discovered Rule List

```
1: training_set ← all training examples
2: rule_list ← empty set
3: while (|training_set| > max_uncovered_examples) do
4:     rulebest ← empty set
5:     i ← 1
6:     repeat    ▷ use separate ant colonies for antecedent and consequent construction
7:         rule_current ← empty set
8:         for (j from 1 to colony_size) do
9:             rule_j ← CreateAntecedent() + CreateConsequent()
10:            Prune(rule j)                          ▷ applies a local search operator
11:            if Q(rule_j) > Q(rule_current) then
12:                rule_current ← rule_j              ▷ updates the reference to the best rule
13:            end if
14:            j ← j + 1
15:        end for
16:        UpdatePheromones(rule_current)
17:        if Q(rule_current) > Q(rule_best) then
18:            rule_best ← rule_current
19:        end if
20:        i ← i + 1
21:    until i ≥ max_number_iterations OR RuleConvergence()
22:    rule_list ← rule_list + rule_best
23:    training_set ← training_set - Covered(rule_best, training_set)
24: end while
25: return rule_list
```

First, association rules are created in the form $T \rightarrow CS$, where T is a set of terms and CS is a classset, that satisfy the minimum support (specified by the user) and minimum confidence. Second, the rules are ranked in order to determine the classification rule of a specific document. Third, the rules that incorrectly classify many training documents are removed. Lastly, the ranked list is remover to minimize the overall classification error.

Decision Tree-Based

In Clare & King (2003) is proposed a modified version of the classic C4.5 decision tree algorithm, named HC4.5, which was specifically developed to deal with hierarchy between the labels. There are few details available about how this algorithm is different from the standard C4.5. The main information the authors provide is referred to modifications that they have made in the entropy formula to consider a form of weighting.

The new entropy, proposed by Clare & King (2003) and presented in Formula 2.23, allows the leaves of the decision tree to be a set of labels. In this context, the classification

output for a new sample can be a set of labels, which is represented as a vector. The model induction is done in one step, generating one decision tree for the whole hierarchy.

$$entropy(S) = -\sum_{j=1}^{N}((p(c_j)\log_2 p(c_j)) + (q(c_j)\log_2 q(c_j)) - \alpha(c_j)\log_2 treesize(c_j)) \quad (2.23)$$

where:

- N = number of classes of the problem.
- p(c_j) = probability (relative frequency) of class c_j.
- $q(c_j) = 1 - p(c_j)$ = probability of not being member of class c_j.
- treesize(c_j) = 1 + number of descendant classes of class c_j (1 is added to represent c_j itself).
- $\alpha(c_j) = 0$, if $p(c_j) = 0$ (a user-defined constant (default = 1) otherwise).

The final output of the HC4.5 classifier, for a given sample y_i, is a vector of real values v_i. If the value of $v_{i,j}$ is above a given threshold l, the example is assigned to the class c_j.

Naive Bayes Classifier

In Silla Jr & Freitas (2009a) the authors extended the traditional flat Naive Bayes to deal with a hierarchical classification problem. According to the authors, this extension allows the algorithm to create a global-model that allows the prediction of any class in the hierarchical class structure instead of only classes at the leaf nodes of the class hierarchy. The authors also augment the global-model Naive Bayes by using a notion of "usefulness", which takes into account the depth of the prediction.

To calculate the probabilities for the Naive Bayes hierarchical classification, the authors adapted Clare's measure of "usefulness" (CLARE, 2003) by using a normalized usefulness value based on the position of each class level in the hierarchy, as presented in Equation 2.24.

$$usefulness(c_i) = 1 - (\frac{a(c_i)\log_2 treesize(c_i)}{max}) \quad (2.24)$$

where:

- treesize(c_i) = 1 + number of descendant classes of c_i (1 is added to represent c_i itself).

- $a(c_i) = 0$, if $p(c_i) = 0$; $a(c_i) =$ a user defined constant (default = 1) otherwise.
- max is the highest value obtained by computing $a(c_i) \log_2 treesize(c_i)$ and it is used to normalize all the other values into the range [0, 1].

Furthermore, in order to make the final classification decision, the proposed Naive Bayes has two options: (1) Assign the final class label with the maximum value of posterior probability (as shown in Formula 2.25); or (2) Assign the class label which maximizes the product of the posterior probability and usefulness (as shown in Formula 2.26).

$$classify(A) = argmax_{class} \prod_{i=1}^{n} (A_i = V_{ij}|Class) \times P(Class) \tag{2.25}$$

$$classify(A) = argmax_{class} \prod_{i=1}^{n} (A_i = V_{ij}|Class) \times P(Class) \times Usefulness(Class) \tag{2.26}$$

Predictive Clustering Trees

In Blockeel et al. (2006) and Vens et al. (2008) the authors present the Clus-HMC algorithm[1], which is based on Predictive Cluster Trees (PCTs). The key idea of Clus-HMC is to build a set of trees to predict a set of labels, instead of only one label. In order to do this, Clus-HMC transforms the classification output into a boolean vector corresponding to the possible labels. The algorithm also needs to consider a distance-based metric to calculate how similar the training samples are in the classification tree. Futhermore, the procedure used to construct the PCTs is similar to another decision tree algorithms, like Classification and Regression Trees (CART) or the classic C4.5.

The Euclidean Distance metric is usually the distance-based metric used in Clus-HMC. In Aleksovski, Kocev & Dzeroski (2009) the authors analyzed the use of other distance measures, namely the Jaccard distance, the SimGIC distance and the ImageClef distance. However, they concluded that there was no statistically significant difference between the distance metrics in their empirical experiments.

Moreover, in Dimitrovski et al. (2011a) the authors have proposed the use of ensembles approaches in the Clus-HMC algorithm, such as Bagging and Random Forests, and concluded that the use of these methods may improve the classification accuracy.

Algorithm 5 presents the pseudocode for the Clus-HMC approach. The main loop (lines 2-8) searches for the best acceptable attribute-value test for a node. If this test $t*$ can be found, then the method creates a new internal node labeled $t*$ and calls itself recursively to construct a sub-tree for each subset in the partition $P*$ induced by $t*$ on the training instances. Otherwise, if no acceptable test can be found, the algorithm creates a leaf node.

[1] Available for download at https://dtai.cs.kuleuven.be/clus/

Algorithm 5 Clus-HMC (adapted from Blockeel et al. (2006))

Input: T: The training set of instances
Output: The decision tree

(t*, h*, P*) ← (none, ∞, empty set)
for each possible test t **do**
    P ← partition induced by t on T
    h ← $\sum_{t_k \in P} \frac{|T_k|}{|T|} Var(T_k)$
    if (h < h*) and acceptable (t, P) **then**
        (t*, h*, P*) ← (t, h, P)
    end if
end for
if t* ≠ none **then**
    for each T_k ∈P* **do**
        $tree_k$ ← Clus-HMC(T_k)
    end for
    return node(t*, $\bigcup_k \{tree_k\}$)
else
    return leaf($\bar{v}$)
end if

2.3.3 Evaluation Metrics

According to Cerri et al. (2015), accuracy measures for flat classification problems are not appropriate for hierarchical multi-label problems. In addition to not consider the label hierarchy, the samples may simultaneously be assigned to more than one label, which makes the conventional accuracy metrics ignore that the difficulty of classification usually increases with the depth of the labels to be predicted.

In general, in hierarchical classification problems, labels from lower levels in the hierarchy (finer-grained labels) can be harder to predict than labels in higher levels (coarse-grained labels). The conventional metrics consider the misclassification costs independent of the location of labels in the hierarchy. Moreover, in multi-label problems, the measures do not consider that a sample may be classified with a subset of its true labels (CERRI et al., 2015).

Considering the previously described context, specific metrics for hierarchical multi-label classification problems have been proposed in the literature. According to Cerri et al. (2015), these metrics can be sub-categorized into: (i) Hierarchy-Based Evaluation measures; and (ii) Distance-Based evaluation measures. While the first ones are based only on the hierarchical label structure, the second ones consider the distance between the predicted and true label paths. The following sub-sections gives a general overview of these hierarchical evaluation measures.

2.3.3.1 Hierarchy-Based Evaluation Metrics

When evaluating a classifier, the hierarchy-based metrics consider the ancestors and descendants of the predicted labels. In the following, we present a brief review of the existing hierarchical-based evaluation measures.

Hierarchical Precision and Recall

Based on the classic precision and recall metric, Kiritchenko, Matwin & Famili (2004) proposed the hierarchical precision and hierarchical recall measures, in order to consider the hierarchical relationships between labels. Even though the metrics were proposed in 2004, they were only formally defined latter, in Kiritchenko, Matwin & Famili (2005).

According to Cerri et al. (2015), the hierarchical precision/recall consider that a sample belongs not only to its predicted labels but also to all its ancestor labels in the hierarchy. Given a sample (x_i, L_i'), in which x_i belongs to the space X of samples, L_i' is the set of predicted labels for x_i, and L_i is the set of true labels of x_i, the sets L_i and L_i' can be extended to contain their corresponding ancestor labels as $\widehat{L_i} = \bigcup_{l_k \in L_i} Ancestors(l_k)$ and $\widehat{L_i'} = \bigcup_{l_m \in L_i'} Ancestors(l_m)$, where $Ancestors(c_k)$ denotes the set of ancestors of class l_k.

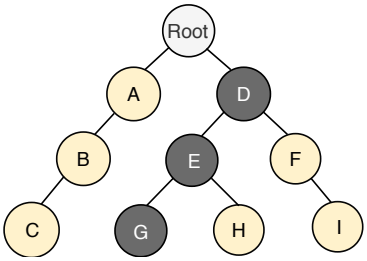


Figure 12 – Example of hierarchical precision and recall measure. The real labels are marked in gray.

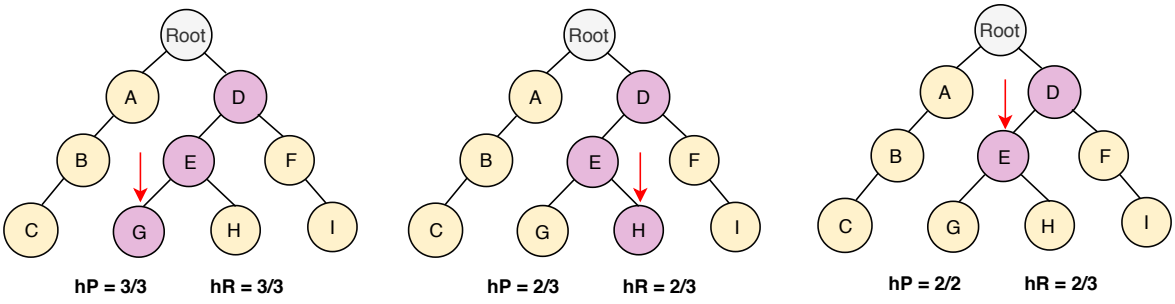


Figure 13 – Example of hierarchical precision and recall measure. The predicted labels are marked in pink.

Equations 2.27 and 2.28 shows the hierarchical precision (hP) and recall (hR) metrics, respectively. These measures count the correctly predicted labels, in conjunction with the number of correctly predicted ancestor labels (KIRITCHENKO; MATWIN; FAMILI, 2005). Figures 12 and 13 shows a visual example of how to calculate hP and hR. While Figure 12 presents the true labels, Figure 13 shows three possible label paths that were predicted for a given sample.

$$hP = \frac{\sum_i |\widehat{L_i} \bigcap \widehat{L_i'}|}{\sum_i |\widehat{L_i'}|} \tag{2.27}$$

$$hR = \frac{\sum_i |\widehat{L_i} \bigcap \widehat{L_i'}|}{\sum_i |\widehat{L_i}|} \tag{2.28}$$

As shown in Figure 13, all nodes from the root to the label pointed with the red arrow were assigned to the sample, indicating that the ancestor labels of the predicted label are also part of the label path. The values for hP and hR in the three different prediction scenarios are shown below each tree in Figure 13 (CERRI et al., 2015).

Hierarchical F-Measure

Equation 2.29 shows the hierarchical F-Measure metric (named $Hierarchical - F_\beta$), which combines hR and hP, previously presented. In the equation, β represents the importance assigned to the values of hP and hR. As the value of β increases, the weight assigned to the value of hR also increases. On the other hand, when the value of β decreases, the weight assigned to hP increases (CERRI et al., 2015).

$$Hierarchical - F_\beta = \frac{(\beta^2 + 1) \times hP \times hR}{\beta^2 \times hP + hR} \tag{2.29}$$

Hierarchical Loss Function

In Cesa-Bianchi, Gentile & Zaniboni (2006), the Hierarchical Loss Function, also known as H-Loss, was proposed. The main idea of this metric is that a misclassification in a label of the hierarchy should not penalize the sub-tree of this label. This means that, when a missclassification occurs in a label l_j', the additional errors in the sub-tree with root at l_j' should not be considered. For instance, if a given classifier misclassifies a music assigning it to the label "rock", this same classifier should not be penalized again by erroneously classifying it in the label "indie rock".

Lets consider that the true labels of a given sample x_i is any subset of the set L, which is composed of all the labels from the hierarchy. In this context, the subset will be represented as a vector (l_1, ...,$l_{|L|}$), where a label l_j belongs to the subset of labels of

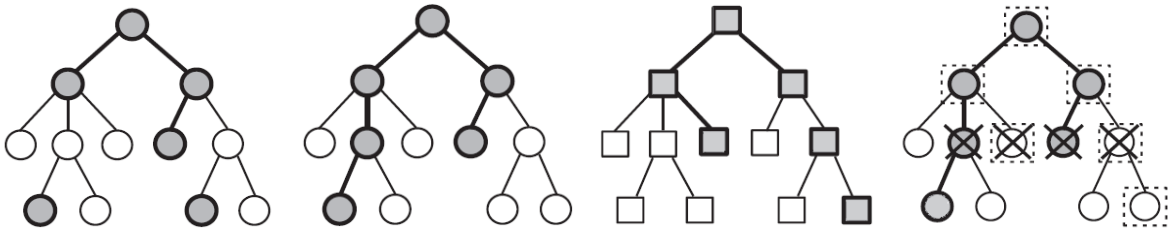
sample x_i if and only if $l_j = 1$. According to Cerri et al. (2015), before defining the H-Loss function, we must define two measures regarding the discrepancy between a multi-label prediction for $x_i(L^{'} = (l^{'}_1, ..., l^{'}_{|L|}))$, and the true set of labels of $x_i(L = (l_1, ..., l_{|L|}))$, for each sample: (i) The zero-one loss $(l_{0/1}(L, L^{'}))$, which is presented in Equation 2.30; (ii) The symmetric difference loss $(l_{\Delta}(L, L^{'}))$, which is defined in Equation 2.31. Based on these metrics, Cesa-Bianchi, Gentile & Zaniboni (2006) proposed the H-Loss function $(l_H(L, L^{'}))$, which is presented in Equation 2.32. In these three equations, 1. is a function used to return 1 if the provided equation is true and 0 otherwise.

$$l_{0/1}(L, L^{'}) = 1, \text{if } \exists_j \in \{1, ..., |L|\} : l_j \neq l^{'}_j \tag{2.30}$$

$$l_{\Delta}(L, L^{'}) = \sum_{j=1}^{|L|} 1\{l_j \neq l^{'}_j\} \tag{2.31}$$

$$l_H(L, L^{'}) = \sum_{j=1}^{|L|} 1\{l_j \neq l^{'}_j \wedge Ancestors(l_j) = Ancestors(l^{'}_j)\} \tag{2.32}$$

According to Cerri et al. (2015), the H-Loss metric leans on the idea that a hierarchical structure G can be considered as a forest of trees based on the set of labels belonging to the problem. Considering this, a multi-label classification $L^{'} \in \{0, 1\}^{|L|}$ respects the structure G only if $L^{'}$ is the union of one or more paths from G, where each path starts in a root and not necessarily ends in a leaf label. Thus, all the paths of G are examined (from root to a leaf) and, when a label $l^{'}_j$ is found, considering that in this case $l^{'}_j \neq l_j$, the value 1 is added to H-Loss and all predictions in the sub-trees rooted in this label $l^{'}_j$ are discarded.



(a) (b) (c) (d)

Figure 14 – Graphical representation of the H-Loss function (CERRI et al., 2015).

Figure 14, first proposed by Cesa-Bianchi, Gentile & Zaniboni (2006) and adapted in Cerri et al. (2015), presents the basic concepts of the H-Loss metric. In the hierarchy example, the round gray nodes represent the labels being predicted for a given sample, while the squared gray nodes represent the true labels of this sample. It is important to observe that in Figure 14(a), the labels predicted do not respect the hierarchical taxonomy

of G, since some parents labels were not predicted. On the other hand, in Figure 14(b), the hierarchical taxonomy is respected. Moreover, Figure 14(c) presents the true labels of the sample previously classified in Figure 14(b), and in Figure 14(d) is presented an example of the H-Loss metric calculation schema, in which the classifications shown in Figures 14(b) and 14(c) are considered. It can be noted that the nodes marked with "X" are the only ones considered when computing the H-Loss. In this example, the zero-one loss function will output the value 1, whilst the symmetric difference loss function will output 6, and, as a final results of H-Loss, we will have the value 4.

2.3.3.2 Distance-Based Evaluation Metrics

This kind of metric takes into account that labels that are closer in the hierarchy are usually similar to each other when compared to distant labels. Thus, classification errors in which close labels are mixed up, should lead to smaller errors rates when compared to confusions made between labels far from each other. In the following, we present a brief review of the existing distance-based evaluation measures.

Micro/Macro Distance-Based Hierarchical Precision and Recall

In Sun & Lim (2001) the micro and macro hierarchical precision and recall metrics are proposed. In general, these measures consider the distance between the true and the predicted labels. For the macro hierarchical precision and recall, first we have to compute the performance obtained in each label separately and then return the average of these values for each metric. By its turn, in the micro hierarchical precision and recall metrics, we compute the average of the performance obtained in each sample of a dataset.

It is necessary to define the contribution of the samples erroneously associated to that label for each one of these metrics and label. According to Cerri et al. (2015), the contribution is defined according to an acceptable distance (named as number of edges (Dis_Θ)) between a predicted and a true label, which must be higher than zero. For instance, when using the value $Dis_\Theta = 2$, the samples that are "slightly" misclassified give no contribution in the computation of the measures, while the samples that are "seriously" misclassified contribute negatively to the values of the metrics. Equations 2.33 and 2.34 define the contribution of a sample x_i to a label l_j, where $x_i.agd$ and $x_i.lbd$ are the predicted and true labels of x_i, respectively. Moreover, $Dis(l, l'_j)$ is the distance between a true label l and a predicted label l'_j.

- If x_i is a false negative:

$$Con(x_i, l'_j) = \sum_{l \in x_i.agd} (1.0 - \frac{Dis(l, l'_j)}{Dis_\Theta}) \tag{2.33}$$

- If x_i is a false positive:

$$Con(x_i, l_j^{'}) = \sum_{l \in x_i.lbd} (1.0 - \frac{Dis(l, l_j^{'})}{Dis_{\Theta}}) \tag{2.34}$$

Equation 2.35 defines the RCon(x_i, $l_j^{'}$), which is a refinement of the contribution of a sample x_i that restricts its values with a range between [-1, 1].

$$RCon(x_i, l_j^{'}) = min(1, max(-1, Con(x_i, l_j^{'})) \tag{2.35}$$

Equations 2.36 and 2.37 defines the total contribution of false positives (FP) ($FpCon_j$) and false negatives (FN) ($FnCon_j$), respectively.

$$FpCon_j = \sum_{x_i \in FP_j} RCon(x_i, l_j^{'}) \tag{2.36}$$

$$FnCon_j = \sum_{x_i \in FN_j} RCon(x_i, l_j^{'}) \tag{2.37}$$

Equations 2.38 and 2.39 define the hierarchical precision and recall for each label, which are calculated after computing the contributions of each sample.

$$Pr_j^{CD} = \frac{max(0, |TP_j| + FpCon_j + FnCon_j)}{|TP_j| + |FP_j| + FnCon_j} \tag{2.38}$$

$$Re_j^{CD} = \frac{max(0, |TP_j| + FpCon_j + FnCon_j)}{|TP_j| + |FN_j| + FpCon_j} \tag{2.39}$$

The values of hierarchical micro precision and recall are presented in equations 2.40 and 2.41, where m represents the number of labels. Depending on Dis_{Θ}, $FpCon_j$ and $FnCon_j$ can assume negative values. Thus, in order to not let their value get lower than zero, a max function is used into the numerators of Equations 2.40 and 2.41 (CERRI et al., 2015).

$$\widehat{Pr}^{\mu CD} = \frac{\sum_{j=1}^{m} max(0, |TP_j| + FpCon_j + FnCon_j)}{\sum_{j=1}^{m} |TP_j| + |FP_j| + FnCon_j} \tag{2.40}$$

$$\widehat{Re}^{\mu CD} = \frac{\sum_{j=1}^{m} max(0, |TP_j| + FpCon_j + FnCon_j)}{\sum_{j=1}^{m} |TP_j| + |FN_j| + FpCon_j} \tag{2.41}$$

Equations 2.42 and 2.43 presents the hierarchical macro precision/recall metrics, in which m is the number of labels.

$$\widehat{Pr}^{M_{CD}} = \frac{\sum_{j=1}^{m} Pr_j^{CD}}{m} \tag{2.42}$$

$$\widehat{Re}^{M_{CD}} = \frac{\sum_{j=1}^{m} Re_j^{CD}}{m} \tag{2.43}$$

Such as hP and hP metrics, the hierarchical micro/macro precision and recall metrics can also be combined to generate the $Hierarchical - F_\beta$ metric.

Calculating the Distances Between Classes

In order to determine the predictions of a given classifier, the micro/macro hierarchical precision/recall use the distances between two labels in the hierarchy. According to Cerri et al. (2015), for the purpose of calculate the distances, the methods are generally defined as a function of two components: (i) the number of edges between the predicted label and the true label and (ii) the depth of the predicted and true label in the hierarchy.

According to Cerri et al. (2015), the standard of the metrics is to consider the distance as the number of edges that separate the true and predicted labels. Besides, weights can be assigned to each edge of the labels hierarchy, making the misclassification between the predicted and true labels be the sum of the weights in the path.

Depending on the hierarchy structure of the problem, different approaches may be used to compute the paths between labels. When the structure is a tree, there is only one path between two labels. However, when dealing with a DAG structure, it is possible to have more then one path between two labels, depending on the number of father labels of each label. According to Cerri et al. (2015), in the final classification, it can be considered two interpretations of the labels hierarchy: (i) if a sample belongs to a label l_j, it belongs to all father labels of l_j; (ii) it belongs to at least one father label of l_j.

2.4 Final Considerations

In this chapter we presented a brief review concerning the machine learning context. Logical inference is one of the most used resources in the artificial intelligence to deal with knowledge. In this work, since we are studying the classification task in specific, we subdivided the logical inference into three main categories: Single-Label Classification, Multi-Label Classification and Hierarchical Classification.

It is worth to mention that, while studying the machine learning concepts, we have used the field known as Information Retrieval (IR) as a case study and we have also published contributions to this research field in the 18^{th} IEEE International Conference on Multimedia and Expo (ICME) (PEREIRA; SILLA JR, 2017), in the 2019 International Joint Conference on Neural Networks (IJCNN) (PEREIRA et al., 2019), in the Thirty-First International Florida Artificial Intelligence Research Society Conference (VALERIO et al.,

2018) and in the Multimedia Tools and Applications (MANGOLIN et al., 2020).

Regardless of label taxonomy, the imbalanceness problem is present in all kinds of datasets. In the single-label classification context, the problem is well-known and studied, whilst in the multi-label classification context is still under investigation. In the next chapter we present a general review concerning the imbalanceness factor in single and multi-label classification contexts.

Chapter 3 The Imbalanceness Factor

Usually the classifiers are focused in the minimization of the global error rate and thus, when dealing with imbalance data, the algorithms tend to benefit the most frequent classes. Most evaluation metrics may hide the incorrect classification of classes with few samples. However, depending on the scenario, the main interest of the task could be properly labelling these rare patterns. A lot of researchers face imbalanced class distribution issues, mostly when working with real world datasets such as medical image classification (ARIAS et al., 2016; BAI et al., 2019; ABDULRAZZAQ et al., 2019; PEREIRA et al., 2020), detection and classification of acoustic scenes and events (MESAROS et al., 2018), anomaly detection (AHMED; MAHMOOD; HU, 2016), credit card fraud detection (KUMAR et al., 2015) and so on.

In the following sections we present explanations concerning how to evaluate the classification results in a imbalanced dataset, metrics to measure the dataset imbalanceness, and the most well-known techniques to resampling datasets in single-label (binary class or multi-classes) and multi-label classification scenarios.

3.1 Evaluating the Classification Results in Imbalanced Scenarios

The performance of machine learning algorithms is typically evaluated using the classic predictive accuracy. However, this is not appropriate when the data is imbalanced and/or the costs of different errors vary markedly.

As an example, lets consider the classification of pixels in mammogram images as possibly cancerous, as studied in Woods et al. (1993). A mammography dataset may have 98% normal pixels and 2% abnormal pixels. The most simple approach, in which the majority class is guessed, will give an accuracy of 98%. However, the nature of the application requires a fairly high rate of correct detection precisely in the minority class, even allowing a small error rate in the majority class in order to achieve this. In this kind of situation, the predictive accuracy is surely not appropriate (CHAWLA et al., 2002).

There are some alternative metrics that can be used in these cases to predict the classification results considering the data imbalanceness. In general, these measures should present a balanced result between precision and recall scores, showing that the classifier is

not only correctly predicting a certain class X, but also not making mistakes by classifying Y samples as X.

Considering this context, according to Davis & Goadrich (2006), one of these measures is the Area Under the Precision-Recall Curve (AUPRC), which is more informative when there is a high-class imbalance in the dataset. Moreover, according to Goutte & Gaussier (2005), another well-known measure that can be used to consider the imbalanceness of the different labels is the F-Score.

3.2 Dealing with Imbalanceness

In order to answer this question, it is necessary to understand and define how the instances are arranged in the dataset. If the dataset is modeled as a binary classification problem, then the imbalanceness problem may be identified if one class, named majority class, has much more instances than the other class (the minority class). A binary dataset with a class proportion of 75/25, for example, may be considered an unbalanced dataset.

According to Fernández et al. (2013), a large number of approaches have been proposed to deal with imbalanceness in binary classification problems, which can be mainly sub-categorized into three groups:

- Data level solutions: Their objective is to re-balance the class distribution by creating and/or removing instances from the dataset to diminish the effect of the class imbalanceness, i.e., pre-process the dataset before the training phase (HART, 1968; CHAWLA et al., 2002; MANI; ZHANG, 2003; BATISTA; PRATI; MONARD, 2004; HAN; WANG; MAO, 2005; STEFANOWSKI; WILK, 2008; HE et al., 2008; BUNKHUMPORNPAT; SINAPIROMSARAN; LURSINSAP, 2009; NAPIERAŁA; STEFANOWSKI; WILK, 2010).

- Algorithmic level solutions: This type of solution modifies the classifier to reinforce the learning towards the minority class. Unlike the data-level solutions, the algorithmic solutions do not make any modifications in the class distribution and, although are classifier dependent, are adaptable to the dataset (ZADROZNY; ELKAN, 2001; BARANDELA et al., 2003; DIAMANTINI; POTENA, 2009; GARCÍA-PEDRAJAS et al., 2012; CIESLAK et al., 2012).

- Cost-sensitive solutions: In these solutions, instead of correctly or incorrectly classifying each sample, each class is associated with a misclassification cost. Thus, instead of optimizing the accuracy, the task is to minimize the total misclassification cost (DOMINGOS, 1999; TING, 2002; ZADROZNY; LANGFORD; ABE, 2003; SUN et al., 2007; ZHAO, 2008; ZHOU; LIU, 2010).

As the most well-known and used techniques to solve the imbalanceness issue and the focus of our research, the resampling methods are the main object of study in the next sections.

The resampling methods can be subdivided in two categories: Oversampling and Undersampling, which are used to adjust the class distribution of a dataset, i.e., the ratio between the different classes in the dataset. While in an undersampling method some instances from the majority class are removed in order to balance the samples distribution, in an oversampling technique, some instances from the minority class are duplicated or synthetically created in order to balance the distribution.

In a certain way, the resampling techniques (oversampling and undersampling) can be considered opposite and roughly equivalent, since both methods use a bias to select more samples from one class than another.

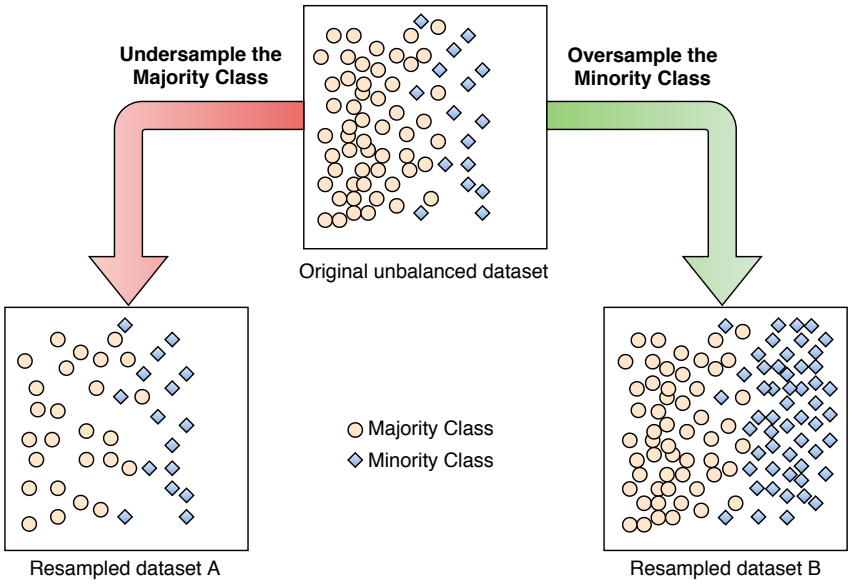


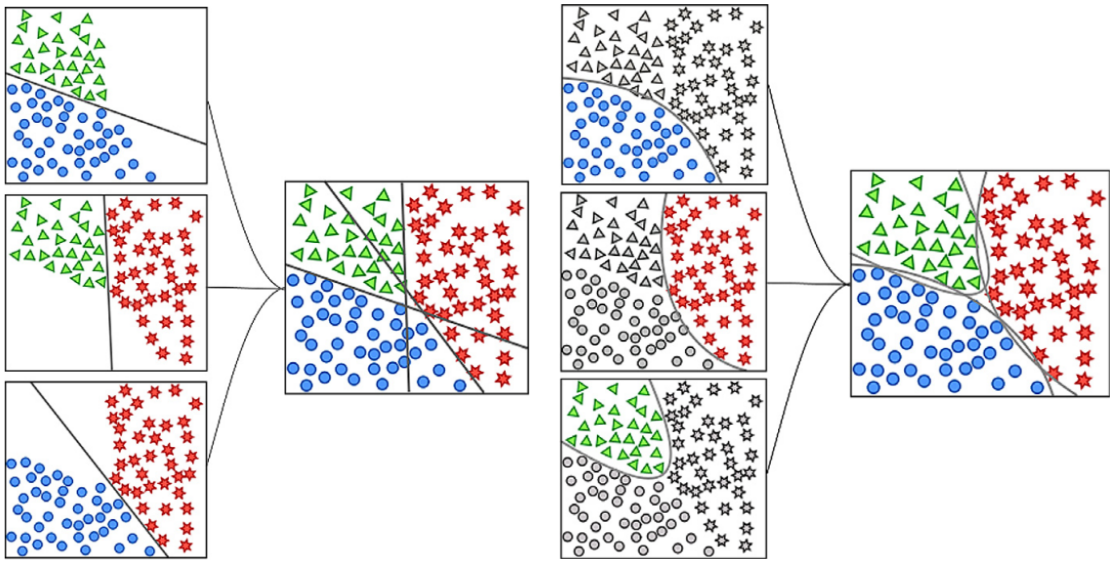
Figure 15 – Different classes distribution in a binary labeled dataset.

Figure 15 shows three different classes distribution in a typical binary dataset: (1) The original unbalanced dataset; (2) The resulting dataset after applying a undersampling over the majority class (Resampled dataset A); (3) The resulting dataset after applying a oversampling over the minority class (Resampled dataset B).

Altought the resampling solutions were first defined and implemented in binary class distribution scenarios, we may apply them to multi-class imbalanceness problems as well. According to Wang & Yao (2012), in order to apply these solutions in multi-class scenarios, most attention in the literature was devoted to class decomposition, i.e., the conversion of a multi-class problem into a set of binary class sub-problems. Fernández et al. (2013) describes two common decomposing schemas: The One-*versus*-One and the One-*versus*-All approaches.

The One-*versus*-One (O-V-O) technique, also known as One-Against-One (O-A-O) and first proposed in Hastie & Tibshirani (1998), tries to train a classifier for each possible pair of classes, ignoring the examples that do not belong to the related classes. When classifying instances, a query is submitted to all binary models, and the predictions of these models are combined into an overall classification. An example of this binarization technique is depicted in Figure 16(a).

On the order hand, the One-*versus*-All (O-V-A) approach, also known as One-Against-All (O-A-A) and introduced by Rifkin & Klautau (2004), builds a single classifier for each of the classes of the problem, considering the examples of the current class to be positives and the remaining instances negatives. An example of this binarization technique is depicted in Figure 16(b).



(a) One-*versus*-One approach. (b) One-*versus*-All approach.

Figure 16 – Example of the binarization techniques for a 3-class problem (FERNÁNDEZ et al., 2013).

3.3 Classic Resampling Techniques

As stated in the previous section, the single-label or classic resampling techniques can be further subcategorized into three groups: (1) Undersampling methods that create a subset of the original dataset by eliminating some of the examples of the majority class; (2) Oversampling methods that create a superset of the original data-set by replicating some of the examples of the minority class or creating new ones from the original minority class instances; (3) Hybrid methods that combine the two previous methods, eliminating some of the examples before or after resampling, in order to reduce *overfitting*.

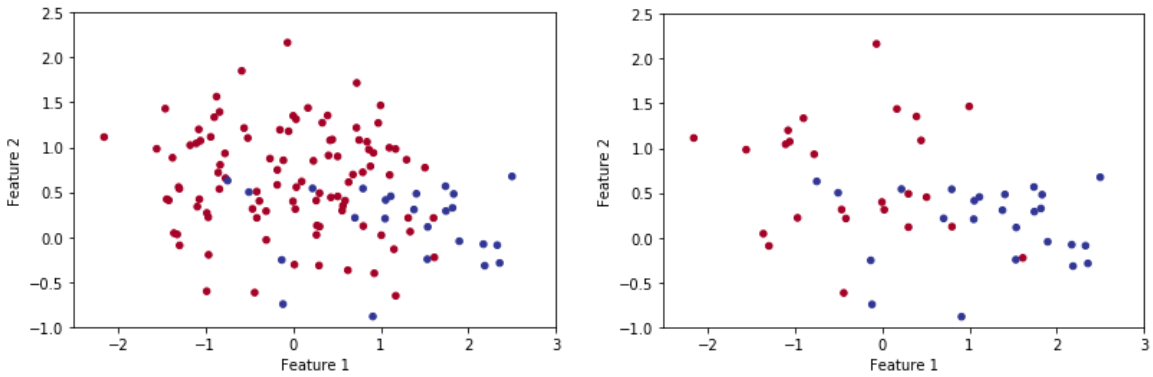
The next subsections are aimed to describe methods belonging to these categories, with figurative examples and pseudocodes. The figure examples represent graphical examples of the samples from a certain dataset before and after applying the given resampling method. The example dataset is composed of two labels (blue and red) and two features (represented by axis y and x).

3.3.1 Undersampling Algorithms

The previous described, the following methods create a subset of the original dataset by eliminating samples from the majority class.

Random Undersampling

The Random Undersampling (RUS) is the most common undersampling techniques in the literature and was proposed in Batista, Prati & Monard (2004). Usually considered as baseline results, its main idea is to balance the dataset class distribution by randomly removing samples from the majority class. While Figure 17 shows a graphical example of dataset before and after applying the RUS method, Algorithm 6 presents the pseudocode for RUS. In the pseudocode, the Dataset D is duplicated to D' and P percent of the samples belonging to the majority class are randomly removed.



(a) Original Dataset. (b) After applying RUS.

Figure 17 – Example of Undersampling technique.

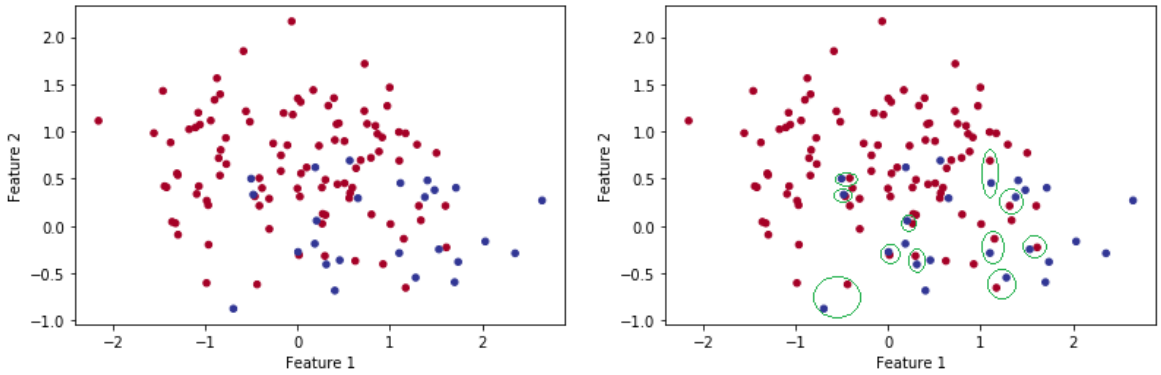
Tomek Links

Given two samples E_i and E_j belonging to different classes, lets consider $d(E_i, E_j)$ the distance between them. A pair (E_i, E_j) is called a Tomek Link if there is no other sample E_l, such that $d(E_i, E_l) < d(E_i, E_j)$ or $d(E_j, E_l) < d(E_i, E_j)$ (BATISTA; PRATI; MONARD, 2004). Figure 18 shows a graphical example of Tomek Link identification for a dataset with a binary classification problem, in which the Tomek Link pairs are identified with green circles.

Algorithm 6 Random Undersampling (adapted from Batista, Prati & Monard (2004))

Inputs: D: Dataset to resample, P: Percentage of samples to delete, C_{maj}: Majority Class
Output: D': The resampled dataset

1: $D' \leftarrow$ copyOf(D)
2: $s_{maj} \leftarrow$ samples from C_{maj}
3: samplesToDelete $\leftarrow (P/100) \times |s_{maj}|$
4: **while** samplesToDelete > 0 **do**
5: $\quad x \leftarrow$ random(s_{maj})
6: $\quad D' \leftarrow D' - x$
7: $\quad s_{maj} \leftarrow s_{maj} - x$
8: $\quad$ samplesToDelete $\leftarrow$ samplesToDelete - 1
9: **end while**
10: **return** D'



(a) Original Dataset. (b) Tomek Links.

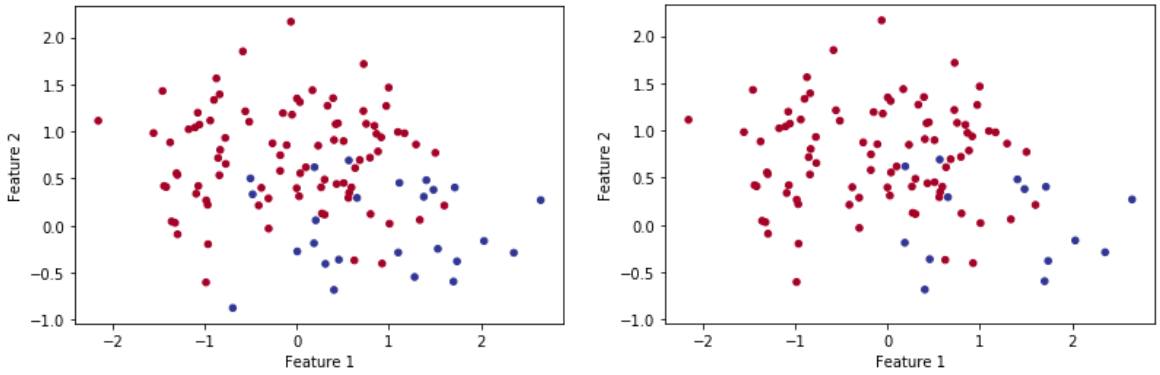
Figure 18 – Classic Tomek Link Identification.

According to Batista, Prati & Monard (2004) Tomek Links could be used as an undersampling technique or as a cleaning post-process step. If used as the first one, only the samples from the majority class identified as Tomek Link pairs are removed, otherwise both samples are removed. Figure 19 shows the datasets resulted from the application of Tomek Link as a Undersampling approach (Figure 19(a)) or as a cleaning method (Figure 19(b)), both over the same dataset from Figure 18. Algorithm 7 shows the Tomek Link undersampling technique. In the pseudocode, the set o Tomek Link samples (identified as *TL*) is identified and removed from the dataset *D*, generating the resampled dataset *D'*.

The reason to use Tomek Link as a post-process clean step leans on the fact that, after applying an oversampling method, frequently the classes groups are not well defined. Some samples from the majority class may be invading the minority class space.

NearMiss

Mani & Zhang (2003) proposed the use of techniques named NearMiss, which has three different variants: 1, 2 and 3. All versions perform undersampling of samples in the



(a) Undersampling. (b) Cleaning.

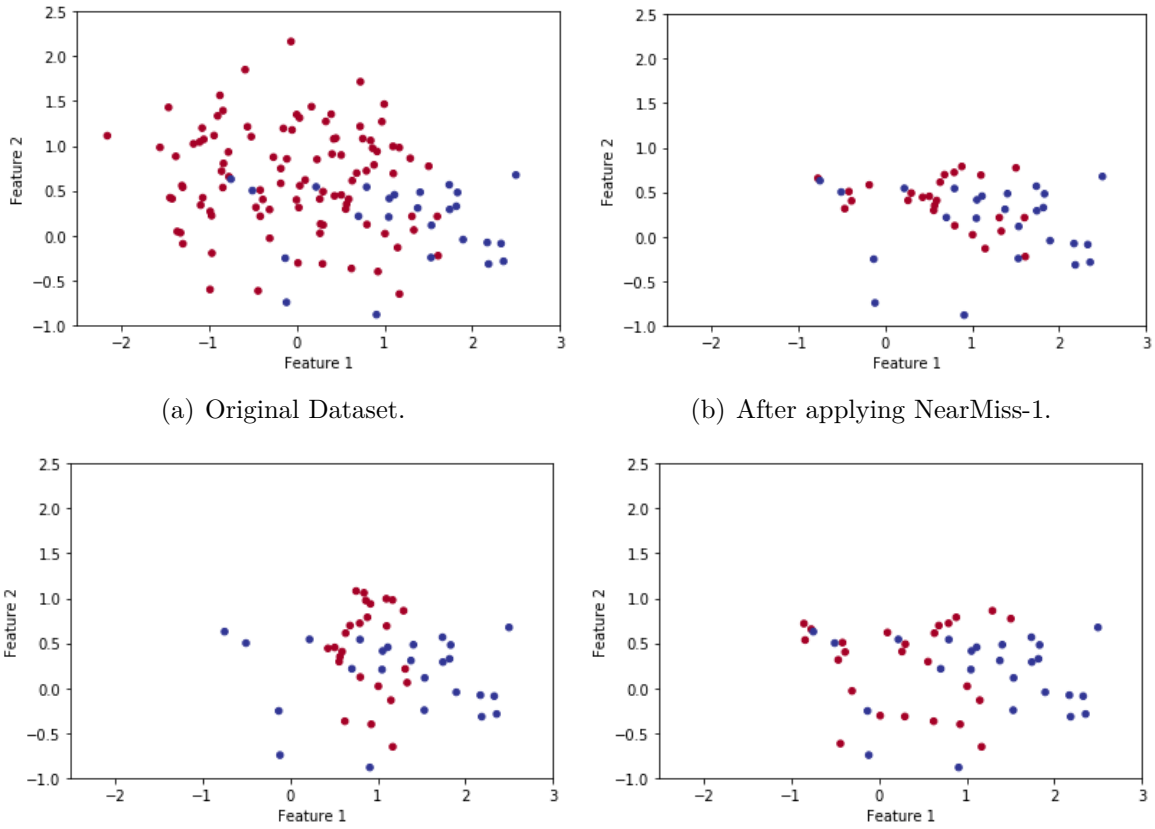
Figure 19 – Tomek Link's types of use.

Algorithm 7 Tomek Link Undersampling (adapted from Batista, Prati & Monard (2004))

Inputs: D: Dataset to resample, C_{maj}: Majority Class
Output: $D^{'}$: Resampled dataset

$TL \leftarrow$ empty set of samples
checkedSamples $\leftarrow$ empty set of samples
$s_{maj} \leftarrow$ samples from C_{maj}
for each sample s_i **in** s_{maj} **do**
    if (s_i **in** checkedSamples) **then**
        continue
    end if
    $NN \leftarrow$ NEARESTNEIGHBOR(s_i)
    checkedSamples $\leftarrow$ checkedSamples $\cup$ s_i
    if (NN[class] $\neq$ s_i[class]) **then**
        $TL \leftarrow TL \cup s_i$
    end if
end for
$D^{'} \leftarrow$ empty dataset
for each sample s_j **in** D **do**
    if (s_j **not in** TL) **then**
        $D' \leftarrow D' \cup s_j$
    end if
end for
return $D^{'}$

majority class based on their distance to other instances from the same class.



(a) Original Dataset.

(b) After applying NearMiss-1.

(c) After applying NearMiss-2.

(d) After applying NearMiss-3.

Figure 20 – Example of NearMiss undersampling technique.

In NearMiss-1, the samples retained from the majority class are those in which their mean distance to their k-nearest neighbors in the minority class is the lowest. On the other hand, NearMiss-2 keeps the samples from the majority class whose mean distance to the k-farthest instances in the minority class is lowest. The last variant, NearMiss-3, selects the k-nearest neighbors from the majority class for every sample in the minority class. While in NearMiss-1 and NearMiss-2 k is a tunable hyperparameter, in NearMiss-3, k is directly used to control the undersampling ratio.

Figure 20 shows an example of the use of the three NearMiss techniques in a unbalanced dataset with binary classes. We may observe that all NearMiss techniques cause huge samples reductions in the original dataset.

Moreover, Algorithm 8 presents a pseudocode for the NearMiss techniques. The three versions are grouped in the same pseudocode, since the V parameter is used as an input to choose between the versions. In the Algorithm, the function sel_dist_based is used to select the samples which will remain in the dataset D based on a vector of distances ($dist_vec$), which is calculated for each sample in the majority class.

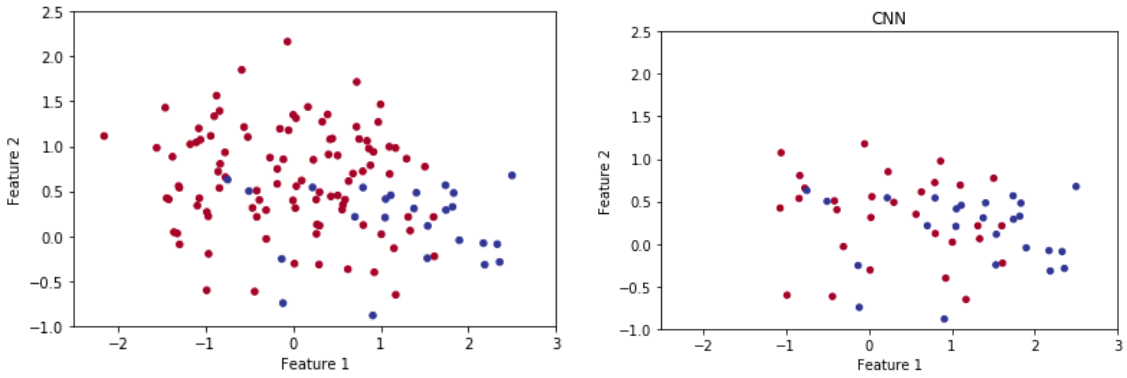
Algorithm 8 NearMiss Undersampling (adapted from Mani & Zhang (2003))

Inputs: D: Dataset to resample, C_{min}: The minority class, C_{maj}: The majority class, k: The number of Neighbors, V: Algorithm version (1, 2 or 3)
Output: $D^{'}$: Resampled dataset

$s_{maj} \leftarrow$ samples from C_{maj}
n_samp_maj $\leftarrow$ number of samples in s_{maj}
$s_{min} \leftarrow$ samples from C_{min}
n_samp_min $\leftarrow$ number of samples in s_{min}
if V $= 1$ **then**
    dist_vec $\leftarrow$ KNearestNeighbors(s_{maj}, k)
    $D^{'} \leftarrow$ sel_dist_based(D, dist_vec, n_samp_maj, C_{maj}, "nearest")
else
    if V $= 2$ **then**
        dist_vec, $samp_{select} \leftarrow$ KFarthestNeighbors(s_{min}, k)
        $D^{'} \leftarrow$ sel_dist_based($samp_{select}$, dist_vec, n_samp_maj, C_{maj}, "farthest")
    else
        if V $= 3$ **then**
            dist_vec $\leftarrow$ KNearestNeighbors(s_{maj}, k, n_samp_min)
            $D^{'} \leftarrow$ sel_dist_based(D, dist_vec, n_samp_maj, C_{maj}, "nearest")
        end if
    end if
end if
return $D^{'}$

Condensed Nearest Neighbor

The Condensed Nearest Neighbor (CNEN) was firstly proposed in Hart (1968). Its main objective is to undersample the dataset by choosing a subset of the training set much more representative then the full dataset. This technique guarantees that all instances in the training set will have the same classification with CNEN and the original dataset, and that the new training set will be no larger than the original.



(a) Original Dataset. (b) After applying CNEN.

Figure 21 – Example of CNEN undersampling technique.

Figure 20 presents an example of application of CNEN method over an unbalanced

dataset, while Algorithm 9 presents a pseudocode for the technique. In the pseudocode, the set of CNEN samples are obtained from the dataset D and the resampled dataset D' is generated by coping the samples which are not present in the CNEN set.

Undersampling with CNEN can be slower if compared with other methods, once it requires a lot of passages over the training data. Due to the randomness envolved in the selection of the points in each iteration, the selected subset can vary significantly.

Algorithm 9 Condensed Nearest Neighbor Undersampling (adapted from Hart (1968))

Inputs: D: Dataset to resample
Output: $D^{'}$: Resampled dataset

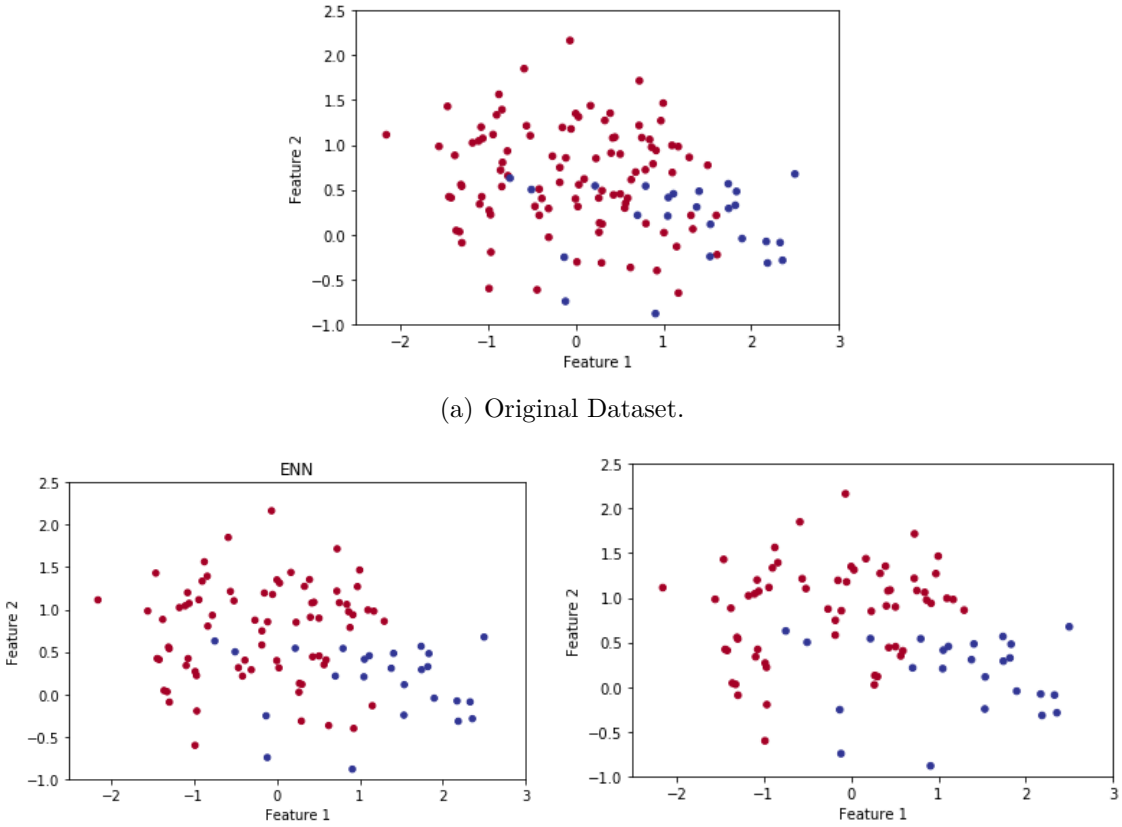
$x \leftarrow$ a random instance from D
CNEN $\leftarrow \{x\}$
additions $\leftarrow$ true
while additions is true **do**
    additions $\leftarrow$ false
    for each sample s_i in D **do**
        classify s_i with samples in CNEN
        if s_i is incorrectly classified **then**
            CNEN $\leftarrow$ CNEN $\cup\ s_i$
            additions $\leftarrow$ true
        end if
    end for
end while
$D^{'} \leftarrow$ empty set of samples
for each sample s_j in D **do**
    if s_j **not in** CNEN **then**
        $D^{'} \leftarrow D^{'} \cup s_j$
    end if
end for
return $D^{'}$

Edited Nearest Neighbor

The Edited Nearest Neighbor (ENN) was proposed in Wilson (1972), the undersampling of the majority class is made by removing the instances in which its label differs from the most of its nearest neighbors.

There is a extension of the ENN named Repeated Edited Nearest Neighbours (RENN), presented in Tomek (1976). In this algorithm the ENN is applied successively until ENN cannot remove anymore instances. Figure 22 shows examples of a binary dataset before and after applying the ENN and RENN techniques.

Algorithm 10 shows ENN method. In this pseudocode, the variable “kind_selection” represents the strategy to use in order to exclude samples. If "all" is chosen, all neighbours will have to agree with the samples of interest to not be excluded. However, if “mode” is



(a) Original Dataset.

(b) After applying ENN.

(c) After applying RENN.

Figure 22 – Example of ENN/RENN undersampling techniques.

selected, the majority vote of the neighbours will be used in order to exclude a sample.

Cluster Centroids

Proposed by Yen & Lee (2009), Cluster Centroids (CC) perform undersampling by generating centroids based on clustering methods. This method that undersamples the majority class by replacing a cluster of majority samples by the cluster centroid of a KMeans algorithm. This algorithm keeps N majority samples by fitting the KMeans algorithm with N cluster to the majority class and using the coordinates of the N cluster centroids as the new majority samples.

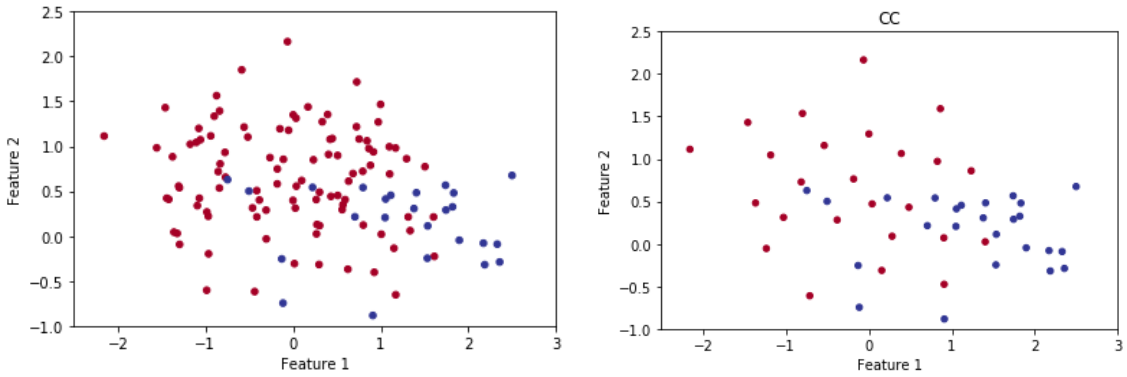
While Figure 23 presents a visual example of the application of CC in a unbalanced dataset, Algorithm 11 shows the pseudocode for the CC technique. The pseudocode uses a *KMeans* function to select the clusters centers in the dataset D and a *generate_sample* function to create the new resampled dataset D' based on these clusters.

Algorithm 10 Edited Nearest Neighbor Undersampling (adapted from Tomek (1976))

Inputs: D: Dataset to resample, C_{maj}: The majority class, sel_kind: Strategy used to exclude the samples, k: Number of neighbors

Output: $D^{'}$: Resampled dataset

$D^{'} \leftarrow$ empty set of samples
$s_{maj} \leftarrow$ samples from C_{maj}
for each sample s_i **in** s_{maj} **do**
    neighbors $\leftarrow$ kNearestNeighbors(D, s_i, k)
    if sel_kind = "mode" **then**
        neigh_labels $\leftarrow$ getLabelSet(neighbors)
        freq_label $\leftarrow$ mostFrequentValue(neigh_labels)
        selected_samples $\leftarrow$ selectSamples(D, neighbors, freq_label)
    else
        if sel_kind = "all" **then**
            ref_label $\leftarrow$ s_i[class]
            selected_samples $\leftarrow$ selectSamples(D, neighbors, ref_label)
        end if
    end if
    $D^{'} \leftarrow D^{'} \cup$ selected_samples
end for
return $D^{'}$



(a) Original Dataset. (b) After applying CC.

Figure 23 – Example of CC undersampling technique.

Algorithm 11 Cluster Centroids Undersampling (adapted from Yen & Lee (2009))

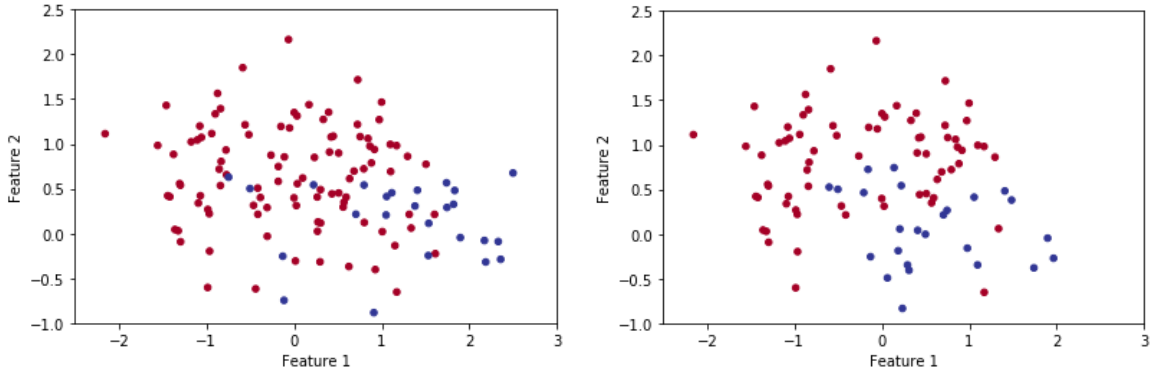
Inputs: D: Dataset to resample, C_{maj}: The majority class

Output: $D^{'}$: Resampled dataset

$D^{'} \leftarrow$ empty set of samples
n_samples $\leftarrow$ number of samples in C_{maj}
cluster_centers $\leftarrow$ KMeans(n_samples)
$D^{'} \leftarrow$ generate_sample(D, cluster_centers, C_{maj})
return $D^{'}$

Neighbourhood Cleaning Rule

The Neighbourhood Cleaning Rule (NCL) (LAURIKKALA, 2001) can be considered a modification of ENN, in which the role of data cleaning is increased. The NCL consists of two phases. The first one is focused in identifying the noisy data from the majority class and removing them by using the ENN method. The second consists into removing samples from the majority class which are misclassified by their k-nearest neighbors.



(a) Original Dataset. (b) After applying NCL.

Figure 24 – Example of NCL undersampling method.

Figure 24 presents a visual example of the application of Neighbourhood Cleaning Rule in a unbalanced dataset, and Algorithm 12 shows the pseudocode for the Neighbourhood Cleaning Rule technique. In the pseudocode, the two phases are represented with the set of samples *P1* and *P2*, which obtain the samples to remove from the dataset D following the previously described reasoning.

All-kNN

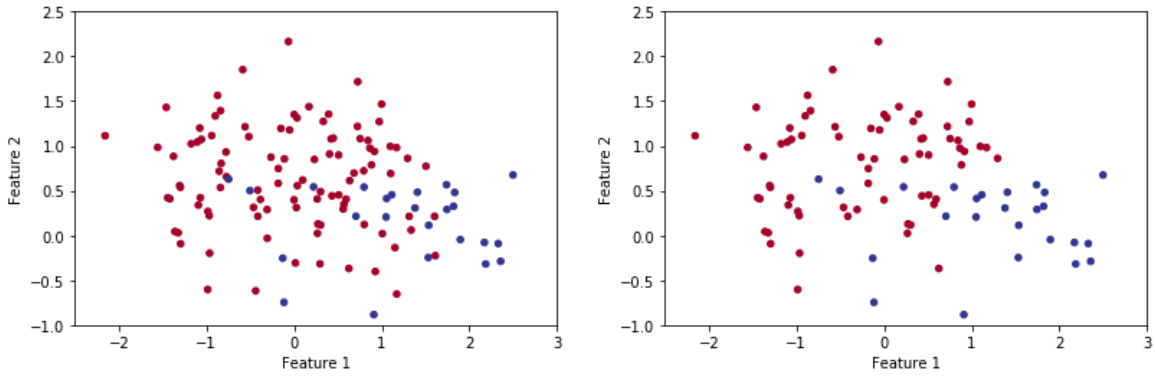
All-kNN was proposed in Tomek (1976), and it can also be used as an undersampling method. The idea is to delete the instances from the majority classes if a k-NN classifier misclassifies it. The technique proposes the use with a k parameter, which will be used in the algorithm to build different neighborhoods ranging from 1 to k neighbors.

While Figure 25 presents a visual example of the application of All-KNN in a unbalanced dataset, Algorithm 13 shows the a pseudocode for the All-KNN method. In the pseudocode, the set of samples misclassified by its k-nearest neighbors are grouped in the set R, which is used to filter the dataset D into D' later in the algorithm.

Algorithm 12 Neighbourhood Cleaning Rule Undersampling (adapted from (LAURIKKALA, 2001))

Inputs: D: Dataset to resample, C_{maj}: The majority class, C_{min}: The minority class, k: The number of neighbors
Output: $D^{'}$: Resampled dataset

$s_{maj} \leftarrow$ samples from C_{maj}
$s_{min} \leftarrow$ samples from C_{min}
$P1 \leftarrow$ EditedNearestNeighbors(s_{maj})
$P2 \leftarrow$ empty set of samples
for each class C_i **in** s_{maj} **do**
    for each $x \in C_i$ **do**
        $y \leftarrow$ kNearestNeighbors(x, k)
        misclas $\leftarrow$ samples y that missclasifies x
        if (misclas not empty) **and** ($|C_i| \geq 0.5 \times |s_{min}|$) **then**
            $P2 \leftarrow x \cup P2$
        end if
    end for
end for
$D^{'} \leftarrow D - (P1 \cup P2)$
return $D^{'}$



(a) Original Dataset. (b) After apply AKNN.

Figure 25 – Example of All-KNN undersampling method.

Algorithm 13 All-KNN Undersampling (adapted from Tomek (1976))

Inputs: D: Dataset to resample, NN: The number of neighbors, C_{maj}: The majority class
Output: $D^{'}$: Resampled dataset

```
R, D' ← empty set of samples
s_maj ← samples from C_maj
for each sample s_i in s_maj do
    for k ← 1 to NN do
        neighbors ← kNearestNeighbors(s_i, k)
        if s_i[class] ≠ mostFreqLabel(neighbors) then
            R ← R ∪ s_i
        end if
    end for
end for
for each sample s_j in D do
    if s_j not in R then
        D' ← D' ∪ s_j
    end if
end for
return D'
```

3.3.2 Oversampling Algorithms

The following methods create new sets of samples from the original dataset by duplicating instances from the minority classes or creating new synthetic samples.

Random Oversampling

Random Oversampling (ROS) is the most common oversampling techniques in the literature and was proposed in Batista, Prati & Monard (2004). Usually considered as baseline results, the main idea is to balance the dataset class distribution by randomly duplicating instances from the minority class.

Algorithm 14 shows the pseudocode for ROS technique. For this resampling algorithm, we do not present a visual example of the feature space modifications. Since the method just duplicate samples from the minority class, these samples are overlapped in the feature space, thus they cannot be viewed in the visual example.

It is important to observe that ROS method may increase the probability of overfitting in the classification, since it produces exact copies of samples from the minority classes. The problem may happen if the classifier is induced to learn only from the duplicated instances.

Algorithm 14 Random Oversampling (adapted from Batista, Prati & Monard (2004))

Inputs: D: Dataset to resample, P: Percentage of samples to increase, C_{min}: Minority Class

Output: $D^{'}$: The resampled dataset

```
1: D' ← copyOf(D)
2: s_min ← samples from C_min
3: samplesToClone ← (P/100) × |s_min|
4: while samplesToClone > 0 do
5:     x ← random(s_min)
6:     newSample ← cloneSample(x)
7:     D' ← D' ∪ newSample
8:     samplesToClone ← samplesToClone - 1
9: end while
10: return D'
```

SMOTE

The Synthetic Minority Oversampling Technique (SMOTE) is one of the most important and used resampling methods from the literature (CHAWLA et al., 2002). The approach is inspired by a technique that proved successful in handwritten character recognition (HA; BUNKE, 1997). SMOTE proposes the creation of new synthetic samples from the minority classes by interpolation nearest instances.

According to the quantity of new samples to be created, some of the k nearest neighbors are randomly chosen. By default SMOTE uses five nearest neighbors. In Figure 26 we exemplify the interpolation process, considering that x_i is the selected point/sample, x_{i1} to x_{i4} are the nearest neighbors and r_{i1} to r_{i4} are the synthetic points/samples created by the randomly interpolation.

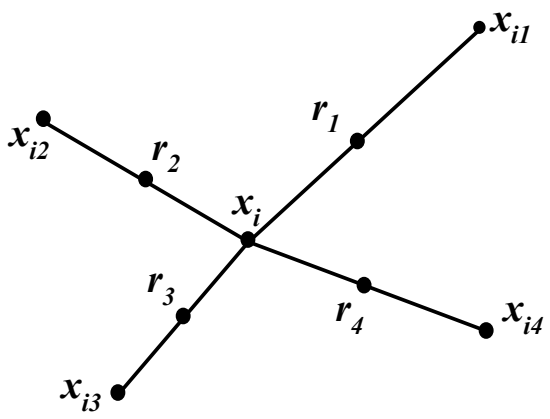


Figure 26 – Example of SMOTE interpolation to create the synthetic samples (RAMENTOL et al., 2012).

Synthetic samples are created using the following criteria: Calculate the difference

between the characteristics vector from the analyzed sample and its nearest neighbor. Multiply this difference by a random value between 0 and 1, add this result to the characteristic vector in analysis. This causes the selection of a new sample between the segment line of two specific characteristics. This approach forces the class decision region to become more widespread.

Lets consider a sample $S = (6, 2)$ and $N = (4, 3)$ as its nearest neighbor. In this example, S is the sample in which k nearest neighbors are being identified. Then, considering a function f_{ij} where i is the sample identifier (1 for sample S and 2 for sample N) and j is the feature identifier (1 for feature x and 2 for feature y), we can calculate the synthetic sample as:

$$\begin{cases} f_{11} = 6 \\ f_{21} = 4 \end{cases} \rightarrow f_{21} - f_{11} = -2 \tag{3.1}$$

$$\begin{cases} f_{12} = 2 \\ f_{22} = 3 \end{cases} \rightarrow f_{22} - f_{12} = 1 \tag{3.2}$$

Considering the calculations above and that rand(0-1) generates a random number between 0 and 1, the new sample will be generated as follows:

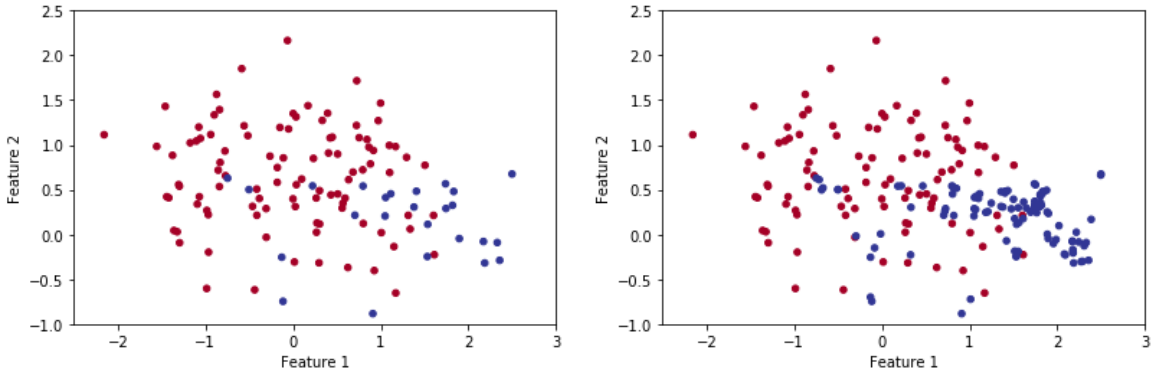
$$(f_1^{'}, f_2^{'}) = (6, 4) + rand(0 - 1) \times (-2, 1) \tag{3.3}$$

According to Chawla et al. (2002) the synthetic examples cause the classifier to create larger and less specific decision regions, rather than smaller and more specific regions. More general regions are now learned for the minority class samples rather than those being subsumed by the majority class samples around them. The effect is that decision trees usually makes a more robust generalization.

While Algorithms 15 and 16 presents the pseudocodes for SMOTE technique, Figure 27 presents a visual example of applying SMOTE over an unbalanced dataset.

SMOTE Borderline

In order to improve the prediction, often the classification algorithms try to learn the borderline of each class as much as possible during the training process. According to Han, Wang & Mao (2005), the examples on the borderline and the ones nearby are more likely to be misclassified than the ones far from the borderline. Considering this context, samples from the borderlines can be considered more important for the classification than the samples far from theses borderlines. Considering this issue, Han, Wang & Mao (2005)



(a) Original Dataset. (b) After applying SMOTE.

Figure 27 – Example of SMOTE oversampling technique.

Algorithm 15 SMOTE Oversampling (adapted from (CHAWLA et al., 2002))

Inputs: D: The dataset to resample, C_{min}: The minority class, P: Percentage of samples to increase, NN: The number of neighbors to consider

Output: D': The resampled dataset

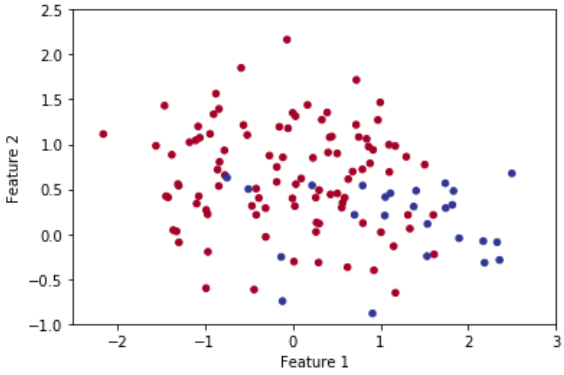
$N_{min} \leftarrow$ Number of samples from C_{min}
if $P < 100$ **then**
    Randomize the N_{min} samples from the minority class
    $N_{min} \leftarrow (P/100) \times N_{min}$
    $P \leftarrow 100$
end if
$NS \leftarrow$ (**int**)$(P/100)$
numattrs $\leftarrow$ Number of attributes
samples $\leftarrow$ Array with the original samples from the minority class
newindex $\leftarrow$ Counter of the number of synthetic samples generated (initialized with 0)
synthetics $\leftarrow$ Vector of synthetic samples
for $i \leftarrow 1$ **to** N_{min} **do**
    calculate NN nearest neighbors for i and save the index in nnarray
    Populate(NS, i, nnarray, samples, synthetics, newindex)
end for
$D' \leftarrow$ copyOf$(D) \cup$ synthetic
return D'

Algorithm 16 SMOTE-Populate (adapted from (CHAWLA et al., 2002))

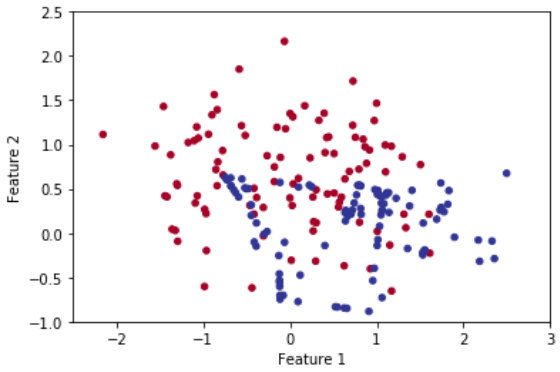
Inputs: D: The dataset to resample, NS: Number of sampes to increase, NN: The number of neighbors to consider, samples: Array of samples from minority class, i: Sample index, nnarray: A array of the neighbors indexes, synthetics: Array of synthetic samples created, newindex: Index for the new synthetic samples

while $NS \neq 0$ **do**
    Choose a random number between 1 and NN and name it *rand*.
    for attr $\leftarrow$ 1 **to** numattrs **do**
        diff $\leftarrow$ samples[nnarray[*rand*]][attr] - samples[i][attr]
        gap $\leftarrow$ random number between 0 and 1
        synthetics[newindex][attr] $\leftarrow$ sample[*i*][attr] + gap $\times$ diff
    end for
    newindex $\leftarrow$ newindex + 1
    $NS \leftarrow NS$ - 1
end while

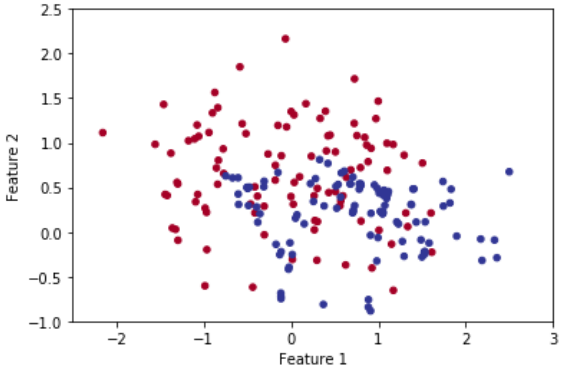
presented two adaptation to the classic SMOTE algorithm: SMOTE Borderline-1 and SMOTE Borderline-2. In these techniques, only the samples from the minority class that belongs to the borderlines of the feature space are oversampled by creating new synthetic samples.



(a) Original Dataset.



(b) After applying SMOTE Borderline-1.



(c) After applying SMOTE Borderline-2.

Figure 28 – Example of SMOTE-Borderline oversampling techniques.

A possible SMOTE Borderline-1 pseudocode is given in Algorithm 17. In this pseudocode, the *DANGER* set represents the samples identified in the borderline that may be used to generate the synthetic samples. SMOTE Borderline-2 not only generates synthetic examples from each example in DANGER and its nearest neighbors from minority class in P, but also does that from its nearest neighbor from majority class in N. The difference between it and its nearest negative neighbor is multiplied by a random number between 0 and 0.5, making the new synthetic sample closer to the minority class. Figure 28 presents a visual example of applying SMOTE Borderline (1 and 2) over an unbalanced dataset.

Algorithm 17 SMOTE Borderline-1 Oversampling (adapted from Han, Wang & Mao (2005))

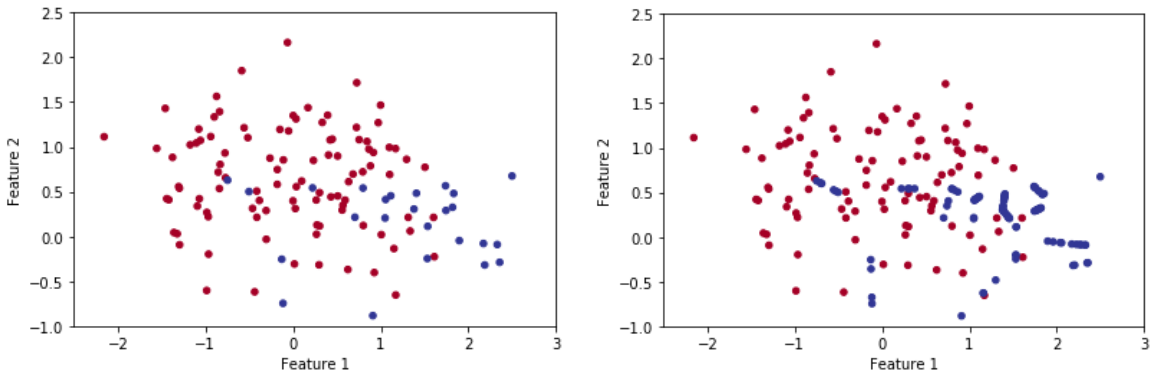
Inputs: D: The dataset to resample, C_{min}: The minority class, m: The number of neighbors to consider

Output: D': The resampled datasets

$s_{min} \leftarrow$ instances from the C_{min}
DANGER $\leftarrow$ empty set
for each p_i in s_{min} **do**
    $nn \leftarrow$ nearestNeighbors(p_i, m)
    $m^{'} \leftarrow$ Number of majority examples among nn $(0 \leq m^{'} \leq m)$.
    if $m^{'} = m$ **then**
        p_i is considered a noise and is not operated in the following steps
    end if
    if $0 \leq m^{'} < m/2$ **then**
        p_i is safe and do not needs to participate in the follows steps
    end if
    if $m/2 \leq m^{'} < m$ **then**
        p_i is considered to be easily misclassified
        DANGER $\leftarrow p_i$
    end if
end for
for each sample in DANGER **do**
    calculate its k nearest neighbors from s_{min}
    generate $dnum \times s$ synthetic samples from the data in DANGER
    for each p_i in s_{min} **do**
        randomly select s nearest neighbors from its k nearest neighbors in s_{min}.
        calculate the differences, $dif_j (j = 1, 2, ..., s)$ between $p_i^{'}$ and its s neighbors
        multiply dif_j by a random number $r_j (j = 1, 2, ..., s)$ between 0 and 1
        $D \leftarrow D \cup$ new synthetic sample generated between $p_i^{'}$ and its neighbors
    end for
end for
$D^{'} \leftarrow$ copyOf(D)
discard all changes in D
return $D^{'}$

Safe-Level SMOTE

According to Bunkhumpornpat, Sinapiromsaran & Lursinsap (2009), SMOTE faces the problem of over-generalization because it blindly generalizes the region of the minority class without considering the majority class. In order to solve this issue, Safe-Level SMOTE attribute to each instance of the minority class a level of security called *safe level* before generating synthetic instances. Each synthetic instance is positioned as close as possible to the highest level of security so that all synthetic instances are generated only in secure regions (BUNKHUMPORNPAT; SINAPIROMSARAN; LURSINSAP, 2009). Figure 29 presents a visual example of applying the Safe-Level SMOTE over an unbalanced dataset.



(a) Original Dataset. (b) After applying Safe-Level SMOTE.

Figure 29 – Example of Safe-Level SMOTE oversampling technique.

The Safe-Level-SMOTE method can be see in Algorithm 18. In the pseudocode, p represents the set of original minority class samples from D, n is the nearest neighbours of p, s is the synthetic sample from the minority class, sl_p and sl_n are safe level of p and safe level of n respectively, and $D^{'}$ is the resampled dataset D.

SPIDER

The Selective Preprocessing of Imbalanced Data (SPIDER) method was first introduced in Stefanowski & Wilk (2008). The idea is to use "internal characteristic" of the samples to drive their preprocessing. Two types of instances are distinguished: noisy and safe. Safe instances should be correctly classified by a constructed classifier, while noisy ones are likely to be misclassified and require special processing. The sample type is discovered by applying the Nearest Neighbor Rule (NNR) in conjunction with the Heterogeneous Value Distance Metric (HVDM). According to the authors, an instance is safe if it is correctly classified by its k-Nearest Neighbors, otherwise it is noisy. Figure 30 presents a visual example of the application of SPIDER in a unbalanced dataset.

In Algorithm 19 is presented the SPIDER technique. For this algorithm, the flags *safe* and *noisy* indicate appropriate types of samples. Moreover, the function

Algorithm 18 Safe-Level-SMOTE Oversampling (adapted from Bunkhumpornpat, Sinapiromsaran & Lursinsap (2009))

Inputs: D: The dataset to resample, k: The number of neighbors to conside
Output: D': The resampled dataset

```
D' ← empty set
for each positive sample p in D do
    compute the k nearest neighbours for p and randomly select one of them, call it n.
    sl_p ← the number of positive instances in k nearest neighbours for p in D
    sl_n ← the number of positive instances in k nearest neighbours for n in D
    if sl_n ≠ 0 then
        sl_ratio ← sl_p/sl_n
    else
        sl_ratio ← ∞
    end if
    if sl_ratio = ∞ and sl_p = 0 then
        does not generate positive synthetic instance
    else
        for atti = 1 to numattrs do
            if sl_ratio = ∞ and sl_p = 0 then
                gap ← 0
            end if
            if sl_ratio = 1 then
                gap ← randomize(0, 1)
            end if
            if sl_ratio > 1 then
                gap ← randomize(0, 1/sl_ratio)
            end if
            if (sl_ratio < 1 then
                gap ← randomize(1-sl_ratio, 1)
            end if
            dif ← n[atti] - p[atti]
            s[atti] ← p[atti] + gap × dif
        end for
        D' ← D'∪ {s}
    end if
end for
return D'
```

$classify\_knn(x, k)$ is used to retrieve the result of classifying the sample x using its k-Nearest Neighbors and $knn(x, k, c, f)$ is used for a set of these examples among k-Nearest Neighbors of x that belong to class c and are flagged as f.

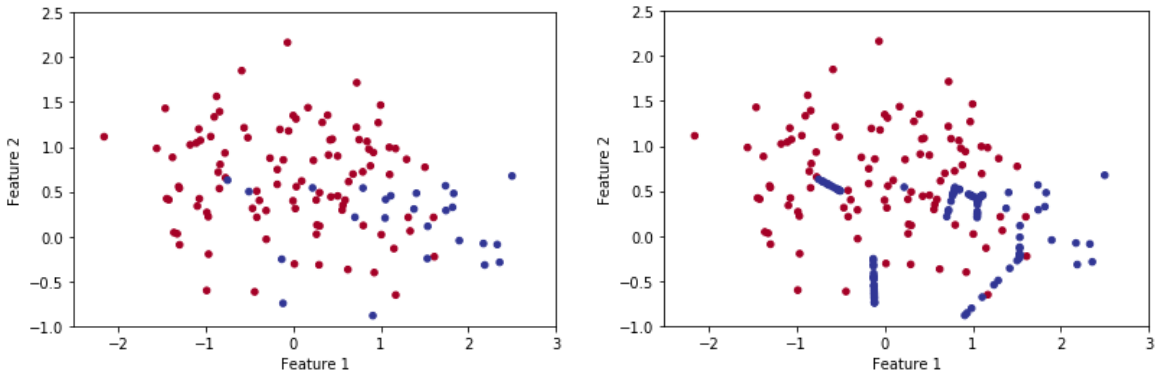
SPIDER-2

In Napierała, Stefanowski & Wilk (2010) proposed a modified version of the classic SPIDER technique, named SPIDER-2. This new version consisted of two phases

Algorithm 19 SPIDER Oversampling (adapted from Stefanowski & Wilk (2008))

Inputs: D: The dataset to resample, C_{min}: The minority class, C_{maj}: The majority class
Output: $D^{'}$: The resampled dataset

for each $x \in C_{maj} \cup C_{min}$ **do**
  if $classify\_knn(x, 3)$ is correct **then**
    flag x as safe
  else
    flag x as noisy
  end if
end for
$D \leftarrow$ all $y \in C_{maj}$ and flagged as *noisy*
if weak amplification **then**
  for each $x \in C_{min}$ and flagged as *noisy* **do**
    amplify x by creating its $\|knn(x, 3, C_{maj}, safe)\|$ copies
  end for
else
  if weak amplification and relabeling **then**
    for each $x \in C_{min}$ and flagged as *noisy* **do**
      amplify x by creating its $\|knn(x, 3, C_{maj}, safe)\|$ copies
    end for
    for each $x \in C_{min}$ and flagged as *noisy* **do**
      for each $y \in knn(x, 3, C_{maj}, noisy)$ **do**
        relabel y by changing its class from C_{maj} to C_{min}
        remove y from D
      end for
    end for
  else
    for each $x \in C_{min}$ and flagged as *safe* **do**
      amplify x by creating its $\|knn(x, 3, C_{maj}, safe)\|$ copies
    end for
    for each $x \in C_{min}$ and flagged as *noisy* **do**
      if classify $knn(x, 5)$ is correct **then**
        amplify x by creating its $\|knn(x, 3, C_{maj}, safe)\|$ copies
      else
        amplify x by creating its $\|knn(x, 5, C_{maj}, safe)\|$ copies
      end if
    end for
  end if
end if
remove all y $\in D$
$D^{'} \leftarrow$ copyOf(D)
discard all changes in D
return $D^{'}$



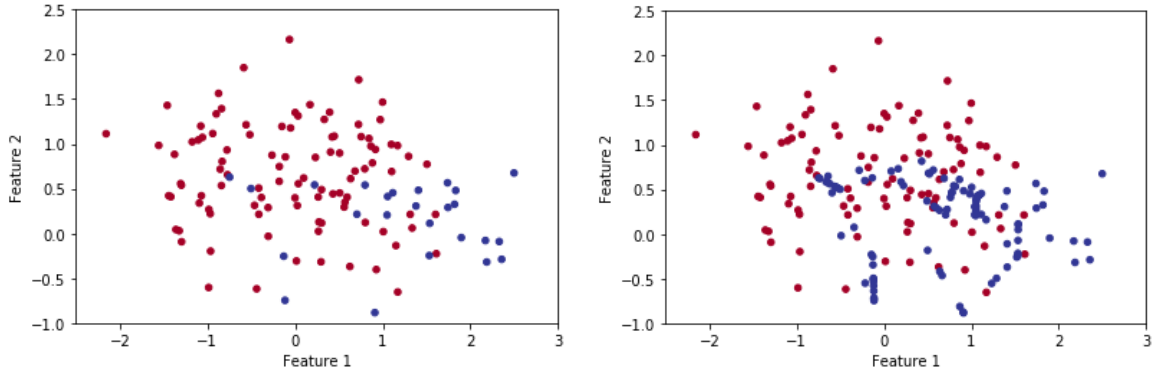
(a) Original Dataset. (b) After applying SPIDER.

Figure 30 – Example of SPIDER oversampling method.

corresponding to the preprocessing of C_{maj} (the majority class) and C_{min} (the minority class), respectively.

The first phase identifies the characteristics of C_{maj} samples and, depending on the label exchange option, can remove or change the label of the C_{maj} noisy samples (changing its classification to C_{min}). In the second phase, it identifies the characteristics of the C_{min} samples considering the changes introduced in the first phase. Then, noisy samples of C_{min} are amplified (by replicating them) according to a *ampl* option.

The two-phase structure is the main difference between SPIDER-2 and SPIDER, which first identifies the nature of the samples and then simultaneously processes C_{maj} and C_{min}. Figure 31 presents a visual example of the application of SPIDER-2 in a unbalanced dataset.



(a) Original Dataset. (b) After applying SPIDER-2.

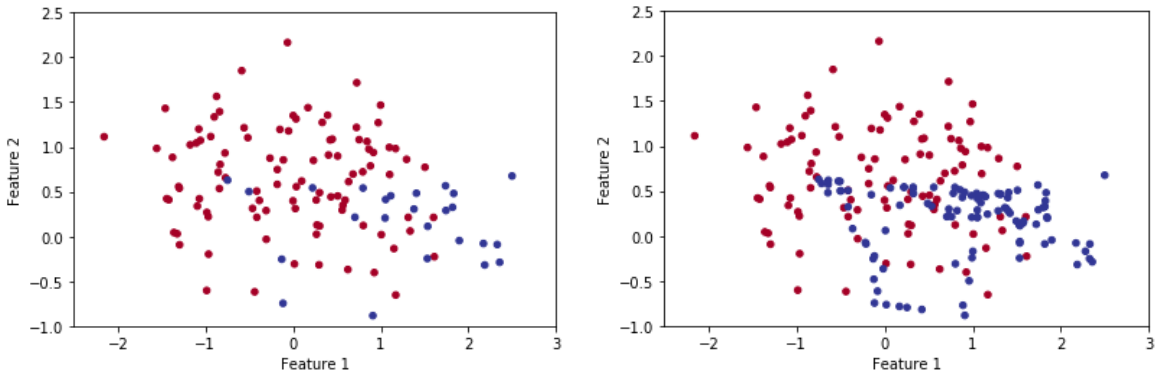
Figure 31 – Example of SPIDER-2 oversampling method.

Algorithm 20 shows the SPIDER-2 method step-by-step. In the pseudocode was used the following auxiliary functions: *correct(S, x, k)* – classifies example x using its k-nearest neighbors in set S and returns true or false for correct and incorrect classification respectively; *class(S, c)* – returns a subset of examples from S that belong to class c;

flagged(S, c, f) – returns a subset of examples from S that belong to class c and are flagged as f; *knn(S, x, k, c)* – identifies and returns these examples among the k-nearest neighbors of x in S that belong to class c; *amplify(S, x, k)* – amplifies example x by creating its n-copies and adding them to S.

ADASYN

Proposed by He et al. (2008), the Adaptive Synthetic Sampling Approach for Imbalanced Learning (ADASYN) method creates synthetic instances for the minority class adaptively. Its essential idea is to use a weighted distribution for different samples from the minority classes according to their level of learning difficulty. More synthetic data is generated for instances of the minority classes that are harder to learn compared to minority samples that are easier to learn.



(a) Original Dataset. (b) After applying ADASYN.

Figure 32 – Example of ADASYN technique oversampling.

Figure 32 shows a visual example of applying the ADASYN over an unbalanced dataset. Algorithm 21 shows ADASYN technique. In this algorithm, the training dataset is represented as D with m samples $x_i, y_i, i = 1, ..., m$, where x_i is an instance in the n-dimensional feature space X and $y_i \in Y = \{1, -1\}$ is the class identity label associated with x_i. Define m_s and m_l as the number of minority class examples and the number of majority class examples, respectively. Therefore, $m_s \leq m_l$ and $m_s + m_l = m$.

3.3.3 Hybrid Resampling Techniques

The following methods combine the application of the two previous methods (oversampling and undersampling), eliminating some minority class instances that have been expanded by oversampling methods to eliminate overfitting.

Algorithm 20 SPIDER-2 Oversampling (adapted from Napierała, Stefanowski & Wilk (2010))

Inputs: D: The dataset to resample, C_{min}: The minority class, k: The number of nearest neighbors, *relabel*: Relabeling option (yes, no), *ampl*: amplification option (no, weak, strong)

Output: D': The resampled dataset

$C_{maj} \leftarrow$ An artificial class combining all classes except C_{min}
for each $x \in$ class(D, C_{maj}) **do**
    if correct(D, x, k) **then**
        flag x as safe
    else
        flag x as not-safe
    end if
end for
$RS \leftarrow$ flagged(D, C_{maj}, not-safe)
if relabel **then**
    for each $y \in RS$ **do**
        change classification of y to C_{min}
        $SR \leftarrow SR \setminus \{y\}$
    end for
else
    $D \leftarrow D \setminus$RS
end if
for each $x \in$ class(D, C_{min}) **do**
    if correct(D, x, k) **then**
        flag x as safe
    else
        flag x as not-safe
    end if
end for
if *ampl* = weak **then**
    for each $x \in$ flagged(D, C_{min}, not-safe) **do**
        amplify(D, x, k)
    end for
else
    if *ampl* = strong **then**
        for each $x \in$ flagged(D, C_{min}, not-safe) **do**
            if correct(D, x, $k+2$) **then**
                amplify(D, x, k)
            else
                amplify(D, x, $k+2$)
            end if
        end for
    end if
end if
$D' \leftarrow$ copyOf(D)
discard all changes in D
return D'

Algorithm 21 ADASYN Oversampling (adapted from He et al. (2008))

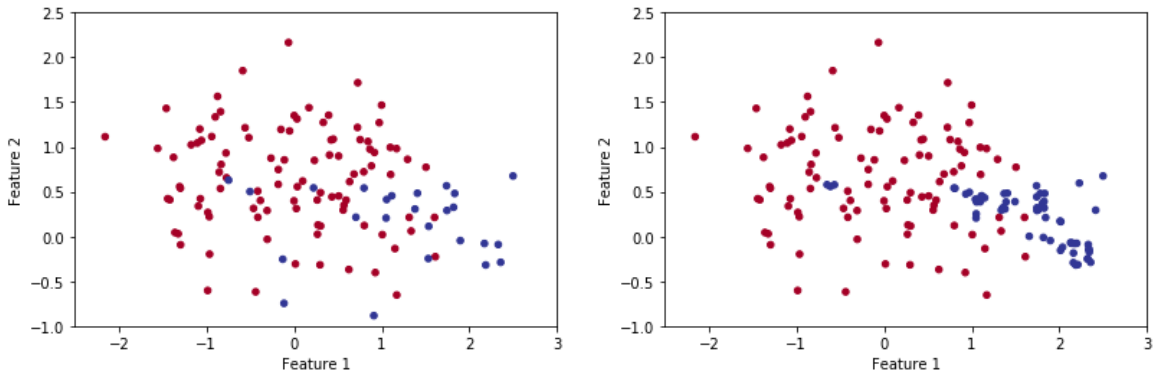
Inputs: D: The dataset to resample, C_{min}: The minority class, d_{th}: Threshold for the maximum tolerated degree of class imbalance ratio

Output: $D^{'}$: The resampled dataset

$d \leftarrow m_s/m_l$
if $d < d_{th}$ **then**
    G $\leftarrow (m_l - m_s) \times \beta$ ▷ total number of synthetic samples to be generated
    for each $x_i \in C_{min}$ **do**
        find k-nearest neighbors based on the Euclidean distance in n-dimensional space
        $\Delta_i \leftarrow$ number of samples in the knn of x_i which belongs to C_{min}
        $r_i \leftarrow \Delta_i/k$, where $i = 1, ..., m_s$
        Normalize r_i according to $\hat{r_i} = r_i / \sum_{i=1}^{m_s} r_i$
        $g_i \leftarrow \hat{r_i} \times G$ ▷ number of synthetic samples to be generated for each x_i
        for each $x_i \in C_{min}$ **do**
            for 1 to g_i **do**
                Randomly choose one minority data example, x_{zi}, from the knn for x_i.
                $\lambda \leftarrow$ a random number between 0 and 1
                $dif\_vector \leftarrow x_{zi} - x_i$
                $s_i \leftarrow x_i + (dif\_vector) \times \lambda$.
            end for
        end for
    end for
end if
$D^{'} \leftarrow$ copyOf(D)
discard all changes in D
return $D^{'}$

SMOTE + ENN

This combination of oversampling and undersampling techniques was first proposed by Batista, Prati & Monard (2004). The ENN technique is used after the application of SMOTE to remove samples from both classes (majority and minority) that are incorrectly classified by their three nearest neighbors.



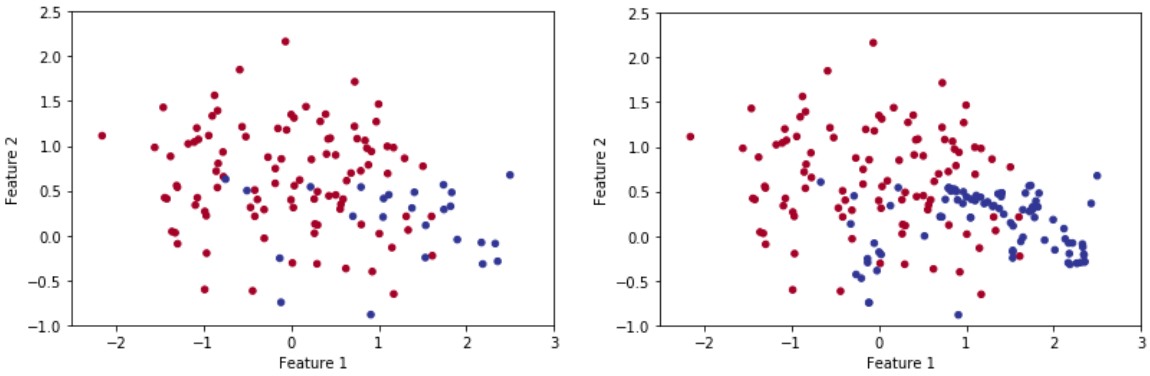
(a) Original Dataset. (b) After applying SMOTE and ENN.

Figure 33 – Example of the SMOTE + ENN hybrid technique.

Figure 3.3.3 shows a visual example of applying the ENN method after apply SMOTE technique over an unbalanced dataset.

SMOTE + Tomek Link

Also proposed in Batista, Prati & Monard (2004), the idea is to apply SMOTE in the original dataset, and then the Tomek Links are identified and removed (of all classes), producing a balanced and well-defined grouping of data.



(a) Original Dataset. (b) After applying SMOTE and Tomek Link.

Figure 34 – Example of the SMOTE + Tomek Link hybrid technique.

Figure 34 shows a visual example of applying the Tomek Link method after apply SMOTE technique over an unbalanced dataset.

3.4 Measuring Imbalanceness in Multi-label Datasets

The number of labels in an Multi-Label Datasets (MLD) can go from a few dozens to several thousands. Only a handful of them have less than ten labels. According to Herrera et al. (2016), despite the fact that most MLDs have a large set of labels, the average number of active labels per instance (their cardinality) seldom is above 5. As a general rule, the more labels there are in an MLD, the higher would be the likelihood of having imbalance problems.

Another important fact, easily deducible from the own MLDs nature, is that there is not a single majority label and a single minority one, but several of them in each group. This have different implications, affecting the way the imbalance level of an MLD can be measured or the behavior of resampling and classification methods.

In order to measure the "imbalanceness" of a multi-label database, the authors of (CHARTE et al., 2013) propose three metrics, which will be detailed explained in the following subsections.

3.4.1 Imbalance Ratio per Label

With D being a MLD with a set of labels Y, and Y_i the i-th label, the Imbalance Ratio per Label (IRLbl) is calculated for the label y as the ratio between the majority label and the label y. Formula 3.4 shows the mathematical representation of this measure.

The IRLbl value will be 1 for the most frequent label and a greater value for the remaining labels. The larger the IRLbl is, the higher is the imbalance level for the considered label.

$$IRLbl(y) = \frac{\max_{y' \in Y}(\sum_{i=1}^{|D|} h(y', Y_i))}{\sum_{i=1}^{|D|} h(y, Y_i)}$$
$$h(y, Y_i) = \begin{cases} 1, & y \in Y_i \\ 0, & y \notin Y_i \end{cases} \tag{3.4}$$

3.4.2 Mean Imbalance Ratio

The Mean Imbalance Ratio (MeanIR) will offer a value that represents the average level of imbalance in an MLD. The value 0 will show that there is no imbalance in the database. Formula 3.5 presents the mathematical terms for MeanIR.

$$MeanIR = \frac{1}{|Y|} \sum_{y=Y_1}^{Y_{|Y|}} IRLbl(y) \tag{3.5}$$

It must be taken into account that different label distributions can produce the same MeanIR value, hence the MeanIR measure should always be used jointly with the CVIR metric, which will be presented in the next subsection.

3.4.3 Coefficient of Variation of IRLbl

The Coefficient of Variation of IRLbl (CVIR) will indicate if the labels suffer from a similar level of imbalance or, on the contrary, there are big differences among them. The larger the CVIR value, the higher would be this difference, and the value 0 indicates that there is no difference between the imbalance levels of the labels. The CVIR measure is given by Formula 3.6.

$$CVIR = \frac{IRLbl\sigma}{MeanIR} \tag{3.6}$$

$$IRLbl\sigma = \sqrt{\sum_{y=Y_1}^{Y_{|Y|}} \frac{(IRLbl(y) - MeanIR)^2}{|Y| - 1}} \tag{3.7}$$

3.4.4 Score of Concurrence among Imbalanced Labels

The Score of ConcUrrence among iMBalanced LabEls (SCUMBLE) was conceived by Charte et al. (2014) in order to establish a way to discover "how difficult" would be to resampling a certain MLD. According to the authors, its main goal is to appraise the concurrence among imbalanced labels. On other words, SCUMBLE metric aims to quantify the imbalance variance among the labels present in each data sample.

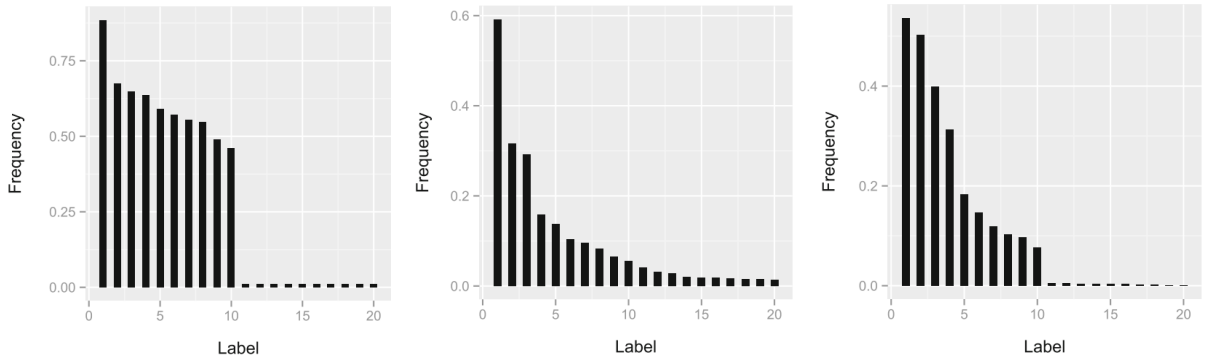
The SCUMBLE measure is calculated as presented in Formula 3.8, and it is based on the Atkinson index (ATKINSON, 1970) and the IRLbl measure (shown in Formula 3.4). The Atkinson index is calculated using incomes, we used the imbalance level of each label instead, taking each instance D_i in the MLD D as a population, and the active labels in D_i as the individuals. If the label l is present in the instance i then $IRLbl_{il} = IRLbl(l)$. On the contrary, $IRLbl_{il} = 0$. $\overline{IRLbl_i}$ stands for the average imbalance level of the labels appearing in instance i. The scores for every sample are averaged, obtaining the final SCUMBLE value.

$$SCUMBLE(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}[1 - \frac{1}{\overline{IRLbl_i}}](\prod_{l=1}^{|Y|} IRLbl_{il})^{(1/|Y|)}] \tag{3.8}$$

The SCUMBLE score will be in the range $[0, 1]$, in which a low value would denote an MLD with not much concurrence among the imbalanced labels, whereas a high one would evidence the opposite case.

3.4.5 Visually Exploration of the Imbalanceness

According to Herrera et al. (2016), besides the use of specific characterization metrics, one of the best approaches to analyze label imbalance in MLDs is to visually explore the data from the dataset. In Figure 35, the relative frequencies for the ten most frequent labels (left side) and the ten least frequent ones (right side) in three well-known MLDs (Cal500 (TURNBULL et al., 2008), TMC2007 (SRIVASTAVA; ZANE-ULMAN, 2005) and Enron (KLIMT; YANG, 2004a)) have been plotted. As we may observe, the difference between frequent and rare labels is immense. Even among the most frequent labels, there are significant disparities, with one or two labels having much more presence than the others. This pattern is common to many MLDs. Therefore, the imbalance problem is almost intrinsically linked to multi-label data.



(a) Cal500 Dataset. (b) TMC2007 Dataset. (c) Enron Dataset.

Figure 35 – Ten most frequent and ten least frequent labels in some datasets (adapted from Herrera et al. (2016)). The frequency scale is individually adjusted to show better the relevance of labels in each MLD, instead of being common to all plots.

The concurrence between the samples labels in MLDs can also be visually explored in some cases. Figures 36 the concurrence among the labels of Yeast Dataset Elisseeff & Weston (2002) is plotted. Each arc in the external circumference represents a label. The arc's amplitude is proportional to the frequency of the label, so small arcs are associated with minority labels, and analogously large arcs indicate majority labels. The width of the bands connecting arcs denote the number of samples in which each label pair appears together. We may observe that the samples associated to minority labels, such as Class14 and Class9, always appear together with one or more majority labels (CHARTE et al., 2017).

3.5 Facing Imbalanced Multi-Label Scenarios

On the basis of the specific characteristics associated with imbalanced MLDs, the design of algorithms capable of dealing with this problem is a challenge (HERRERA et al., 2016). In general, two main approaches have been followed in the literature to face the imbalanceness issue in multi-label scenarios:

- Classifier Adaptation: The idea is to adapt the classifier to take this aspect into consideration, for instance assigning weights to each label depending on its frequency. It is a solution tightly attached to the adjusted algorithm. Although it is not a general application approach, what can be seen as a disadvantage, the adaptation can strengthen the best point of a good classifier, something that a pre-processing method cannot do.

- Resampling Techniques: Consisting of the same core idea from the single-label

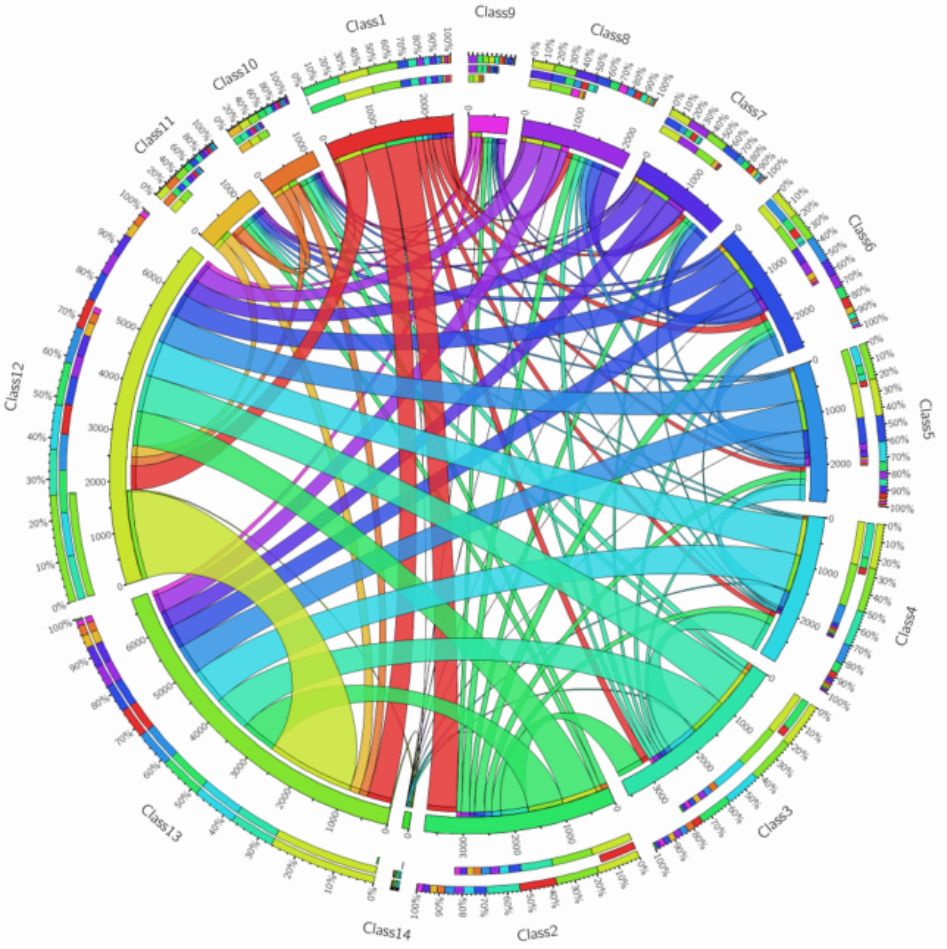


Figure 36 – Concurrence among the labels in Yeast Dataset (CHARTE et al., 2017).

resampling methods, these techniques are based on removing samples which belong to the majority label, adding samples associated with the minority label, or both actions at once.

As the focus of this work is the development of resampling techniques, in the next section we describe the most well-know algorithms in the multi-label resampling context.

3.6 Multi-Label Resampling Techniques

In the last years some papers have been published concerning solutions for imbalanceness in multi-label problems. For this task, the most common and successful studies are based on resampling techniques. In this section we present a brief review of the main multi-label resampling approaches, as well as their pseudocodes.

3.6.1 Random Algorithms

In Charte et al. (2013) two random resampling techniques are presented. LPROS (Label Powerset Random Oversampling) and LPRUS (Label Powerset Random Undersampling) evaluate the frequency of the full labelsets. While LPRUS removes instances from the most frequent labelsets, LPROS clones samples associated with the least frequent ones.

Algorithms 22 and 23 presents the pseudocodes for LPROS and LPRUS, respectively. Both algorithms take as input the percentage of samples to create/remove from the MLD. After computing the average number of samples sharing each labelset, a set of minority/majority bags are produced. The number of instances to create/delete is distributed among these minority/majority bags, randomly picking the data samples to duplicate/remove.

Algorithm 22 LPROS (adapted from Charte et al. (2013))

Inputs: D: The dataset with all original instances, P: Percentage of samples to duplicate
Output: D': The resampled dataset

samplesToClone $\leftarrow |D|/100 \times P$
labelsets $\leftarrow$ retriveLabelSets(D) ▷ Group samples according to their labelsets
for each i **from** 1 **to** |labelsets| **do**
    $labelSetBag_i \leftarrow$ samplesWithLabelset(i)
end for
meanSize $\leftarrow 1/|labelsets| \times (sum(i = 1, labelsets)|labelSetBag_i|)$
for each $labelSetBag_i$ **in** labelSetBag **do**
    if $|labelSetBag_i| <$ meanSize **then**
        $minBag_i \leftarrow labelSetBag_i$
    end if
end for
meanIncrement $\leftarrow$ samplesToClone/$|minBag|$
minBag $\leftarrow$ SortFromLargestToSmallest(minBag)
for each $minBag_i$ **in** minBag **do**
    $incrementBag_i \leftarrow$ max($|minBag_i|$ - meanSize, meanIncrement)
    remainder $\leftarrow$ meanIncrement - $incrementBag_i$
    $distributeAmongBags_{(j>i)}$(remainder)
    for each n **from** 1 **to** $incrementBag_i$ **do**
        $x \leftarrow$ random(features)
        createSample($minBag_i$, x)
    end for
end for
$D' \leftarrow$ copyOf(D)
discard all changes in D
return D'

The work of Charte et al. (2015a) also presents two different random resampling methods: Multi-Label Random Oversampling (MLROS) and Multi-Label Random Under-

Algorithm 23 LPRUS (adapted from Charte et al. (2013))

Inputs: D: The dataset with all original instances, P: Percentage of samples to delete
Output: D': The resampled dataset

samplesToDelete ← $|D|/100 \times P$
labelsets ← retriveLabelSets(D) ▷ Group samples according to their labelsets
for each i **from** 1 **to** |labelsets| **do**
    $labelSetBag_i$ ← samplesWithLabelset(i)
end for
meanSize ← $1/|labelsets| \times (sum(i = 1, labelsets)|labelSetBag_i|)$
for each $labelSetBag_i$ **in** labelSetBag **do**
    if $|labelSetBag_i|$ > meanSize **then**
        $majag_i$ ← $labelSetBag_i$
    end if
end for
meanReduction ← samplesToDelete/$|majBag|$
majBag ← SortFromSmallestToLargest(majBag)
for each $majBag_i$ **in** majBag **do**
    $reductionBag_i$ ← min($|majBag_i|$ - meanSize, meanReduction)
    remainder ← meanReduction - $reductionBag_i$
    $distributeAmongBags_{(j>i)}$(remainder)
    for each n **from** 1 **to** $reductionBag_i$ **do**
        x ← random(1, $|majBag_i|$)
        deleteSample($majBag_i$, x)
    end for
end for
D' ← copyOf(D)
discard all changes in D
return D'

sampling (MLRUS). However, unlike LPROS and LPRUS, both techniques are focused on the evaluation of the individual imbalance level per label, deleting instances linked to the majority labels (MLRUS) or cloning those associated with the minority ones (MLROS).

The pseudocodes for MLROS and MLRUS are respectively provided in Algorithms 24 and 25. Once the number of samples to clone/delete is computed, the IRLbl and MeanIR measures are used to get a bag with the instances in which each minority/majority label appears. The clones (or samples to delete) will be generated (or deleted) from these bags. A new sample is created from each minority bag (or deleted from each majority bag), reassessing their condition of minority/majority bags in each cycle.

3.6.2 MLeNN

Multi-Label edited Nearest Neighbor (MLeNN), proposed in Charte et al. (2014), is a heuristic undersampling procedure. The method was built considering the ENN (Edited Nearest-Neighbor) rule, which is a well-known data cleaning algorithm. It compares the

Algorithm 24 MLROS (adapted from Charte et al. (2015a))

Inputs: D: The dataset with all original instances, P: Percentage of samples to delete
Output: D': The resampled dataset

samplesToClone $\leftarrow |D|/100 \times P$
labelsets $\leftarrow$ retriveLabelSets(D) ▷ Group samples according to their labelsets
MeanIR $\leftarrow$ calculateMeanIR(D, $labelsets$)
for each label **in** $labelsets$ **do** $IRLbl_{label} \leftarrow$ calculateIRperLabel(D, label)
    if $IRLbl_{label} >$ MeanIR **then**
        $minBag_{i++} \leftarrow Bag_{label}$
    end if
end for
while samplesToClone > 0 **do** ▷ Clone a random sample from each minority bag
    for each $minBag_i$ **in** minBag **do**
        $x \leftarrow$ random(1, $|minBag_i|$)
        cloneSample($minBag_i$, x)
        if $IRLblminBag_i \leq$ MeanIR **then**
            minBag $\leftarrow minBag_i$
        end if
        samplesToClone $\leftarrow$ samplesToClone - 1
    end for
end while
$D' \leftarrow$ copyOf(D)
discard all changes in D
return D'

classes from each sample against the one from its k-Nearest Neighbors, in which k is usually three. Those samples whose classes differ from the classes of at least half of the neighbors are marked for removing.

Algorithm 26 presents the pseudocode for MLeNN. According to Charte et al. (2014), the algorithm adapts the ENN rule to the Multi-Label context introducing two key ideas: (1) A principle to chose the samples acting as candidates to be removed; and (2) A comparison operator to determine when the labelsets of two instances are considered to be different. Only the instances which do not contain any minority label are used as candidates.

The samples labelsets are compared with a metric based on the Hamming distance among labelsets, but only taking into account active labels. As a result, a value in the range of [0,1] is obtained. Applying a configurable threshold (parameter TH in Algorithm 26), the algorithm determines which of its neighbors will be considered as distinct.

3.6.3 MLSMOTE

The Multi-Label Synthetic Minority Oversampling Technique (MLSMOTE) was proposed by Charte et al. (2015b), and it is an adaptation of the original SMOTE

Algorithm 25 MLRUS (adapted from Charte et al. (2015a))

Inputs: D: The dataset with all original instances, P: Percentage of samples to delete
Output: $D^{'}$: The resampled dataset

samplesToDelete $\leftarrow |D|/100 \times P$
labelsets $\leftarrow$ retriveLabelSets(D) ▷ Group samples according to their labelsets
MeanIR $\leftarrow$ calculateMeanIR(D, $labelsets$)
for each label **in** $labelsets$ **do** $IRLbl_{label} \leftarrow$ calculateIRperLabel(D, label)
    if $IRLbl_{label} <$ MeanIR **then**
        $majBag_{i++} \leftarrow Bag_{label}$
    end if
end for
while samplesToDelete > 0 **do** ▷ Clone a random sample from each minority bag
    for each $majBag_i$ **in** majBag **do**
        $x \leftarrow$ random(1, $|majBag_i|$)
        deleteSample($majBag_i$, x)
        if $IRLblmajBag_i \geq$ MeanIR **then**
            majBag $\leftarrow majBag_i$
        end if
        samplesToDelete $\leftarrow$ samplesToDelete - 1
    end for
end while
$D^{'} \leftarrow$ copyOf(D)
discard all changes in D
return $D^{'}$

Algorithm 26 MLeNN (adapted from Charte et al. (2014))

Inputs: D: The dataset with all original instances, TH: The threshold, NN : Number of Neighbors to consider
Output: $D^{'}$: The resampled dataset

for each sample **in** D **do**
    for each label **in** getLabelset(D) **do**
        if IRLbl(label) $>$ MeanIR **then**
            Jump to next sample ▷ Preserve instance with minority labels
        end if
    end for
    numDifferences $\leftarrow 0$
    for each neighbor **in** nearestNeighbors(sample, NN) **do**
        if adjustedHammingDist(sample, neighbor) $> TH$ **then**
            numDifferences $\leftarrow$ numDifferences + 1
        end if
    end for
    if numDifferences $\geq NN/2$ **then**
        markForRemoving(sample)
    end if
end for
$D^{'} \leftarrow$ deleteAllMarkedSamples(D)
discard all changes in D
return $D^{'}$

(proposed by Chawla et al. (2002)) toward the creation of syntethic samples in MLDs. Rather than producing synthetic samples only from one minority class, the MLSMOTE method individually processes the set of samples in which the minority labels appears.

In practice, each one of the minority instances is used as seed for a new synthetic sample. The new samples need a set of synthetic features, as well as a synthetic labels, and, to solve this issue, MLSMOTE adopt four main strategies:

1. Minority instances selection: The algorithm assumes that most MLDs will have more than one minority label. Therefore, the criterion to know if a label l is minority takes into account if IRLbl(l) > MeanIR, i.e., the labels that has a frequency below the average frequency of all labels in the dataset.

2. Nearest neighbor search: Once a sample belonging to a minority label has been selected, MLSMOTE has to search its nearest neighbors. The size of the neighborhood is established by a parameter, and the process begins by obtaining the distances between the sample and the other instances from the same minority bag. In the algorithm, the distance between two samples is calculated aggregating the differences between their corresponding features. For numeric attributes the Euclidean Distance is used, while for the nominal ones is used the Value Difference Metric (VDM) (STANFILL; WALTZ, 1986).

3. Feature set generation: Having selected one of the neighbors, the set of features for the synthetic instance is obtained through a interpolation technique.

4. Synthetic labelset production: In order to generate the synthetic labelset three techniques may be used: Ranking, Union and Intersection. As statistically proved in Charte et al. (2015b), the most effective technique was ranking the labels in the labelsets. This method consists in counting the number of occurrences of each label in the reference sample and its neighbors, including in the synthetic labelset those present in half or more of the samples considered.

Algorithm 27 shows the pseudocode of MLSMOTE. The main body of the algorithm spans from line 4 to line 14, a loop that goes across all the labels in the dataset. For each label its IRLbl is obtained, and if it is above the MeanIR the label is considered as minority. In this case, all the samples in which the label appears are taken as seed instances, looking for their nearest neighbors and generating a synthetic sample.

Furthermore, Algorithm 28 shows the pseudocode for the function in charge of generating the new synthetic instances. The function uses as inputs the seed sample, its nearest neighbors, and a random neighbor (one of the previous neighbors) that will be taken as reference to interpolate the features values. As can be seen, the function is

basically divided in two parts: While the first one (lines 1-10) produces the feature values of the synthetic sample, the second one (lines 12–15) generates the synthetic labelset.

Algorithm 27 MLSMOTE (adapted from Charte et al. (2015b))

Inputs: D: The dataset with all original instances, NN : Number of Neighbors to consider
Output: D': The resampled dataset

1: $D' \leftarrow$ copyOf(D)
2: L $\leftarrow$ labelsInDataset(D) ▷ Full set of labels
3: MeanIR $\leftarrow$ calculateMeanIR(D, L)
4: **for each** label **in** L **do**
5: $IRLbl_{label} \leftarrow$ calculateIRperLabel(D, label)
6: **if** $IRLbl_{label} >$ MeanIR **then**
7: minBag $\leftarrow$ getAllInstancesOfLabel(label) ▷ Bags of minority labels samples
8: **for each** sample **in** minBag **do**
9: distances $\leftarrow$ calcDistance(sample, minBag)
10: sortSmallerToLargest(distances)
11: neighbors $\leftarrow$ getHeadItems(distances, NN) ▷ Neighbor set selection
12: refNeigh $\leftarrow$ getRandNeighbor(neighbors)
13: synthSample $\leftarrow$ newSample(sample, refNeigh, neighbors)
14: $D' \leftarrow D' \cup$ synthSample
15: **end for**
16: **end if**
17: **end for**
18: **return** D'

3.6.4 REMEDIAL

The Resampling by decoupling highly imbalanced labels (REMEDIAL), presented in Charte et al. (2015c), is a method designed to deal with MLDs having a high concurrence level between the labels. This method works both as an oversampling method and as an editing technique.

Algorithm 29 shows REMEDIAL pseudocode. This method is specifically designed to work with MLDs having a high SCUMBLE. Firstly, the samples in which minority and majority labels appear together are found. Then, for each instance in the previous set a new sample is generated by preserving the original features, but containing only the minority labels. The final step is to edit the original sample, removing the same minority labels from it.

According to Charte et al. (2015c), using REMEDIAL, the samples which can make harder the learning process are decoupled. Moreover, as the authors highlight, the method can also be used as a previous step to apply other resampling techniques.

Algorithm 28 Function: newSample

Inputs: sample: Sample of reference, refNeigh: Random neighbor of reference, neighbors: Nearest neighbors as configured in the main algorithm
Output: synthSmpl: The new synthetic sample

```
synthSmpl ← new Sample                                  ▷ Feature set assignment
for each feat in synthSmpl do
    if typeOf(feat) is numeric then
        diff ← refNeigh.feat - sample.feat
        offset ← diff × randInInterval(0, 1)
        value ← sample.feat + offset
    else
        value ← mostFreqVal(neighbors,feat)
    end if
    syntSmpl.feat ← value
end for
lblCounts ← counts(sample.labels)                       ▷ Label set assignment
lblCounts ← lblCounts + counts(neighbors.labels)
labels ← lblCounts > (k+1) / 2
synthSmpl.labels ← labels
return synthSmpl
```

Algorithm 29 REMEDIAL (adapted from Charte et al. (2015c))

Inputs: D: The dataset with all original instances, NN : Number of Neighbors to consider
Output: $D^{'}$: The resampled dataset

```
D' ← empty dataset
L ← labelsInDataset(D)                                  ▷ Full set of labels
for each label in L do                                  ▷ Calculate imbalance levels
    IRLbl_l ← calculateIRLbl(l)
end for
IRMean ← mean(IRLbl)
for each Ins_i in D do
    SCUMBLE_Ins_i ← calculateSCUMBLE(Ins_i)
end for
SCUMBLE ← mean(SCUMBLE_Ins_i)
for each instance i in D do
    if SCUMBLE_Ins_i > SCUMBLE then
        D'_i ← D_i                                      ▷ Clone the affected instance
        D_i[labelsIRLbl ≤ IRMean] ← 0                   ▷ Maintain minority labels
        D'_i[labelsIRLbl > IRMean] ← 0                  ▷ Maintain majority labels
        D ← D + D'_i
    end if
end for
D' ← copyOf(D)
discard all changes in D
return D'
```

3.6.5 REMEDIAL-HwR

In 2019, Charte et al. (CHARTE et al., 2019) proposed an improvement for the REMEDIAL technique named REMEDIAL Hybridization with Resampling (REMEDIAL-HwR). The main goal of the proposal is to hybridize REMEDIAL with well-known resampling methods described in the literature. In their experiments three different combinations with REMEDIAL were tested: MLROS (REMEDIAL-HwR-ROS), MLeNN (REMEDIAL-HwR-HUS) and MLSMOTE (REMEDIAL-HwR-SMT).

3.7 Final Considerations

In this chapter we presented a review concerning the imbalanceness issue in classification datasets. Imbalanceness is a well-known and studied problem in the binary and multi-class classification scenarios, with a large variety of solutions, nevertheless it may be noted that it is still an open problem in the multi-label context, being addressed in some studies in the recent years (CHARTE et al., 2015a; CHARTE et al., 2015b; CHARTE et al., 2017; CHARTE et al., 2019).

Whilst there are a lot of resampling alternatives for single-label datasets, there are only a few techniques in the multi-label context. Considering this issue, in the next chapter we present a preliminary contribution of this work: The Multi-Label Tomek Link (MLTL), an adaptation for the classic Tomek Link algorithm to deal with imbalanced multi-label data.

Chapter 4 The Multi-Label Tomek Link

As well as some multi-label resampling methods were inspired from classic techniques from the single-label scenario, such as MLSMOTE and MLeNN, there are resampling techniques that have not been adapted and analyzed in the multi-label context, which could lead to promising results. Among these methods, there is the Tomek Link.

The Tomek Links identification, first introduced by Tomek (1976), was proposed as an undersampling technique in Batista, Prati & Monard (2004). A pair of instances is a Tomek Link if they are nearest neighbors but belong to different classes. According to Batista, Prati & Monard (2004), if two samples form a Tomek Link, it means that either one of them is a noise or both are at the boundary of the clusters. The resampling can be made by removing only the samples belonging to the majority classes or removing all Tomek Links pairs from the dataset. If used as in the second case, Tomek Link is considered a post-process cleaning method.

In the following sections we present the Multi-Label Tomek Link (MLTL). Besides pseudocodes, we also present an experimental analysis for the proposed technique applied to seven well-known datasets from the literature. A paper describing the Multi-Label Tomek Link was published in the Neurocomputing journal (PEREIRA; COSTA; SILLA JR, 2020).

4.1 The Proposed Method

There are two main concerns when resampling a Multi-Label Dataset. First, single-label resampling methods usually consider that there are only one minority and one majority class in the dataset, which does not apply to MLDs. Second, classic resampling methods, such as Tomek Link, cannot be directly applied in MLDs, since they were developed to deal with instances that belong to only one class. Thus, a new MLD resampling approach has to be designed by taking into account the intrinsic characteristics of multi-label classification problems.

In order to attend the first consideration, we used the imbalance measures IRLbl and MeanIR. The criterion used to consider a label l as a majority label is that $IRLbl(l) < MeanIR$, i.e., if the frequency of l is above the average frequency of all labels, it is

considered a majority label.

The second and main concern was to adapt the identification of Tomek Links to deal with multiple labels. Tomek Links are defined based on the similarity between the features from two instances at the same time that have different labels. Thus, in the Multi-Label context, the problem is to define how different two labelsets are. In order to calculate this difference, MLTL uses a Hamming Distance between the labelsets.

The Hamming Distance may be calculated by counting how different two labelsets are. However, if the total number of labels in the dataset is large, the percentage of distance between two instances is usually low. To solve this issue Charte et al. (2014) suggested the use of an *Adjusted* Hamming Distance (AHD), which considers only the "active" labels from the labelsets. In this context, the "active labels" means the union of labels from the labelsets.

Given two vectors of binary numbers a and b of size N and considering *VecSum(z)* as the individual sum of the elements of a given vector z, we define the percentages of Hamming Distance ($HD_{\%}$) and *Adjusted* Hamming Distance ($AHD_{\%}$) in Equations 4.1 and 4.2, respectively. While Equation 4.1 is defined as the number of coefficients in which the vectors differ over all possible coefficients, Equation 4.2 is given as the fraction of coefficients in which the vectors differ considering only the ones which appears in at least one of the vectors.

$$HD_{\%}(a,b) = \frac{VecSum(XOR(a,b))}{N} \tag{4.1}$$

$$AHD_{\%}(a,b) = \frac{VecSum(XOR(a,b))}{VecSum(OR(a,b))} \tag{4.2}$$

Table 7 shows an example of the difference between two labelsets. The first line represents the labels from the dataset (numbered from 1 to 100), the second and third lines are the two labelsets being compared (the number 1 represents the presence of a label), and the fourth line shows the differences between the labelsets (the number 1 represents that there is a difference). In the example presented on 7, the labels from 6 to 99 are note present in the labelsets, thus they are represented with the value zero. We may observe that calculating the HD measure we obtain a distance of 3% (3/100), a very low value. On the other hand, using the AHD metric the distance is 50% (3/6), a higher and considerable value. As there are MLDs with many different labels, but usually with few active ones, it makes sense to use the AHD measure to evaluate the differences between labelsets.

Algorithms 30, 31 and 32 show the MLTL scheme of operation. Algorithm 30 (lines 2-6) handles the type of use chosen for MLTL, i.e., a cleaning technique or undersampling method, and creates the new resampled dataset based on the instances selected for removal.

Table 7 – Example of differences between labelsets.

Line n.° 1	#Label	1	2	3	4	5	...	100
Line n.° 2	Labelset 1	0	1	1	1	0	0	1
Line n.° 3	Labelset 2	1	1	1	0	1	0	1
Line n.° 4	*Difference*	*1*	*0*	*0*	*1*	*1*	*0*	*0*

Algorithm 31 identifies the Tomek Links instances in case of undersampling, i.e., creation of the majority bags of instances[2] (lines 1-9) and selects only the Tomek Link samples from these bags. On the other hand, Algorithm 32 identifies all Tomek Links from the dataset, irrespective of whether they are from the majority or the minority class.

Algorithm 30 Multi-Label Tomek Link

Inputs: D: Dataset to resample, TH: Threshold
Output: D': Resampled dataset

```
1: TL ← new empty list of instances
2: if (MLTL was chosen as a cleaning procedure) then
3:     TL ← cleaningMethod(D, TH)
4: else
5:     TL ← undersamplingMethod(D, TH)
6: end if
7: D' ← new empty dataset
8: for each sample in D do
9:     if (sample not in TL) then
10:        D' ← D' ∪ {sample}
11:    end if
12: end for
13: return D'
```

As we may observe in Algorithms 31 (lines 20-22) and 32 (lines 10-12), it is necessary to define a Threshold (TH) to consider two labelsets as "distinct" or not, since the *Adjusted* Hamming Distance generates a value in the range of [0,1]. We propose a metric to find a TH value considering the dataset imbalanceness. Equation 4.3 defines an indicator I, based on MeanIR measure. This equation determines the imbalance level of the dataset with a value between 0 and 1. Using I, Equation 4.4 shows how to determine TH considering three levels of imbalanceness: High ($I < 0.3$); Medium ($0.5 > I \geq 0.3$); and Low ($I \geq 0.5$). These intervals were empirically determined during the preliminary experiments.

$$I = \frac{1}{\sqrt{MeanIR}} \tag{4.3}$$

$$TH = \begin{cases} 0.5 & if \quad I \geq 0.5 \\ 0.3 & if \quad 0.5 > I \geq 0.3 \\ 0.15 & if \quad I < 0.3 \end{cases} \tag{4.4}$$

[2] Maj./min. bag of instances is a set of samples from a maj./min. class.

Algorithm 31 Undersampling Method

Inputs: D: Dataset to resample, TH: Threshold
Output: TL: Tomek Link instances

```
L ← labelsInDataset(D)
meanIR ← getMeanIR(D)
for each l in L do
    iRLBl ← getIRLbl(l)
    if (iRLBl < meanIR) then
        majBags[l] ← getInstances(l)
    end if
end for
TL ← empty list of instances
checkedSamples ← new empty list of instances
for each majBag in majBags do
    for each sample in majBag do
        if (sample in checkedSamples) then
            continue
        end if
        NN ← nearestNeighbor(sample)
        checkedSamples ← checkedSamples ∪ {sample}
        dist ← adjustedHammingDist(sample, NN)
        if (dist ≥ TH) then
            TL ← TL ∪ sample
        end if
    end for
end for
return TL
```

Algorithm 32 Cleaning Method

Inputs: D: Dataset to resample, TH: Threshold
Output: TL: Tomek Link instances

```
TL ← new empty list of instances
checkedSamples ← new empty list of instances
for each sample in D do
    if (sample in checkedSamples) then
        continue
    end if
    NN ← nearestNeighbor(sample)
    checkedSamples ← checkedSamples ∪ sample
    dist ← adjustedHammingDist(sample, NN)
    if (dist ≥ TH) then
        TL ← TL ∪ sample
    end if
end for
return TL
```

The use of I (shown in Equation 4.3) is important to define a value strictly between

0 and 1. However, using this value directly as the threshold may be a problem because it tends to be too small (if the dataset is very imbalanced) or too big (if the dataset is not imbalanced), leading to a cleaning threshold that can be too high or too low, which in practice (in some of our preliminary experiments) would either remove all the instances or leave the instances unmodified. Thus, we defined Equation 4.4 to "normalize" TH for three possibilities of imbalanceness: high, medium and low.

It is important to observe that TH may be manually defined and tested in the MLTL algorithm, which gives the user the freedom to test the best threshold for its problem. Nevertheless, the idea behind Equations 4.3 and 4.4 is to give the user an automatic way to define TH based on the imbalanceness level of the dataset, adapting this threshold to different datasets, which may even be used as a start value for further tests aimed at optimizing the threshold value (which is out of the scope of this contribution).

In the following we give a brief example of how to manually set the TH threshold considering the pre-established values obtained from Equations 4.3 and 4.4. Tables 8, 9 and 10 present three hypothetical datasets with different levels of imbalanceness: low (exemplified by dataset D1); medium (D2); and high (D3). Considering the MeanIR from the datasets, we can calculate the indicator I, from Equation 4.3, which will lead to 0.81, 0.41 and 0.09 values for D1, D2 and D3, respectively. With the indicator I, the researcher can use Equation 4.4 to define a starter value for TH threshold and then may test other values around it. For instance, for each hypothetical dataset we could test the following thresholds: D1 - TH starts with 0.5 and values between 0.35 and 0.75 may be tested; D2 - TH starts with 0.3 and values from 0.2 to 0.5 may be tested; D3 - TH starts with 0.15 and values from 0.05 to 0.35 may be tested.

Table 8 – Hypothetical Dataset D1.

	La	*Lb*	*Lc*
#Samples	100	120	190
IRLbl	1.90	1.58	1.00
MeanIR		1.49	

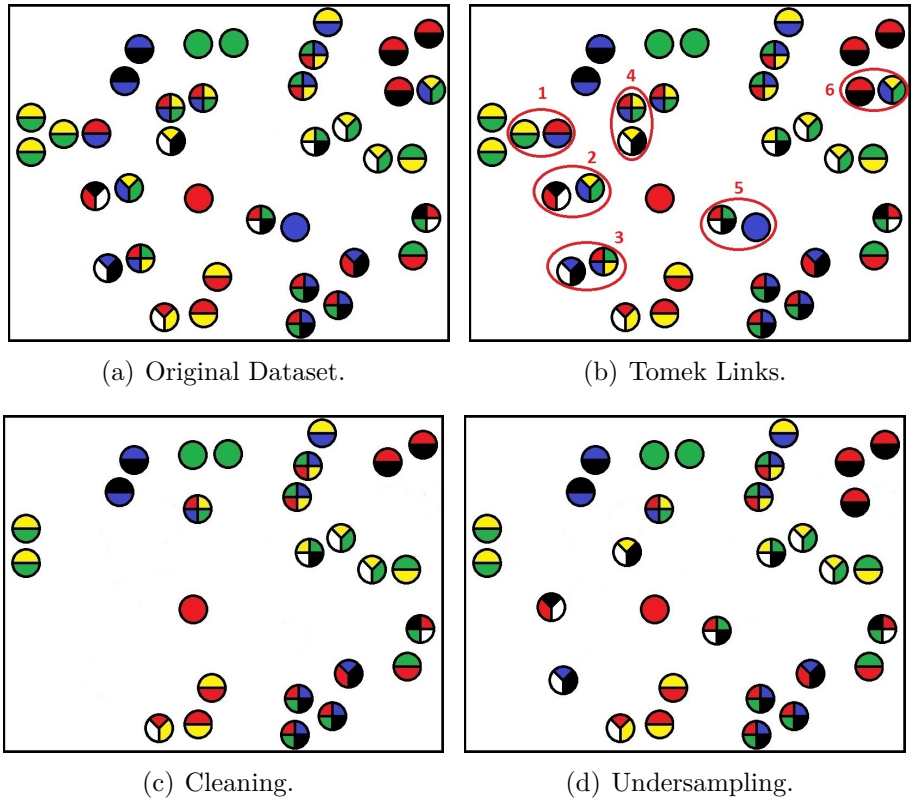
Table 9 – Hypothetical Dataset D2.

	La	*Lb*	*Lc*	*Ld*
#Samples	100	500	700	50
IRLbl	7.00	14.00	1.00	14.00
MeanIR		5.85		

In Figure 37 we present a graphical example of the Multi-Label Tomek Link identification for the labels in a dataset. Each label is then represented by one specific color: Green, Red, Yellow, Blue, Black and White.

Table 10 – Hypothetical Dataset D3.

	La	*Lb*	*Lc*	*Ld*	*Le*
#Samples	10	100	500	900	5000
IRLbl	500.00	50.00	10.00	5.50	1.00
MeanIR			113.33		



(a) Original Dataset. (b) Tomek Links.

(c) Cleaning. (d) Undersampling.

Figure 37 – The Multi-Label Tomek Link (in this illustration, each color refers to one of the labels assigned to the instance).

As we can see in Figure 37(a), the dataset is composed of 38 samples, which are labeled with 1 to 4 labels (each label is represented by a different color). Using Equation 3 to calculate the imbalance ratio per label (IRLbl) of these labels we will get the following values: Green - 1.00; Red - 1.05; Yellow - 1.16; Blue - 1.29; Black - 1.47; and White - 2.44. As the mean imbalance ratio (MeanIR) of these labels is 1.40, we consider the Black and White labels as belonging to the Minority Classes, since their IRLbl are greater than the MeanIR. All the other labels are considered as belonging to the Majority Classes.

Figure 37(b) shows which pairs of instances were identified as Tomek Links (numerated from 1 to 6). In this example, the identification is made by considering a Threshold (TH) of 0.5 when applying the *Adjusted* Hamming Distance in each instance and its nearest neighbor. Furthermore, in Figure 37 we also show the resulting datasets when using MLTL as a Cleaning Method (Figure 37(c)) and as an Undersampling Technique (Figure 37(d)). It is important to observe that, when using MLTL as undersampling technique, in the cases

where the Tomek Link pairs do not have a label belonging to the Minority Classes (e.g.: pair identified with number 1 in Figure 37(b)), MLTL chooses to remove both instances from the dataset.

In order to give a general overview of the proposed resampling method, Figure 38 presents a graphical abstract of the Multi-Label Tomek Link approach. In the schema, the labels *S1.X* and *S2.Y* represents the steps towards the construction of the final classification model. While *S1.X* shows the flow when MLTL is chosen as a Undersampling Method, *S2.Y* is used to represent the flow of MLTL as a Cleaning Step.

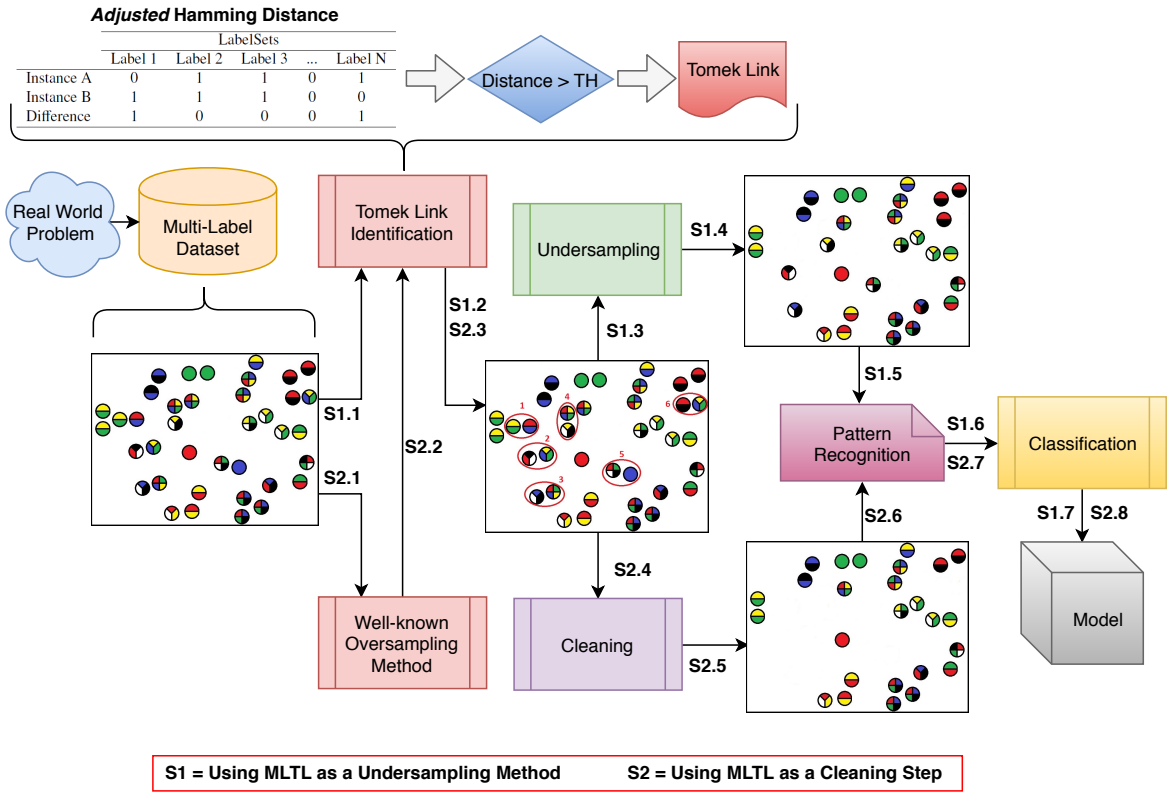


Figure 38 – Graphical Overview of the Multi-Label Tomek Link.

4.2 Experimental Analysis

This section shows the experimental evaluation of the Multi-Label Tomek Link algorithm. The following subsections present the datasets, parameters configuration, results and discussion related to our proposed method.

4.2.1 The Datasets

We have tested the proposed method over seven well-known multi-label datasets from the literature from different domains: Cal500 (TURNBULL et al., 2008), Emotions

(TROHIDIS et al., 2008), Enron (KLIMT; YANG, 2004a), FMA (DEFFERRARD et al., 2017), Medical (CRAMMER et al., 2007), Scene (BOUTELL et al., 2004), Yeast (ELISSEEFF; WESTON, 2002). These datasets were chosen since they are present in many multi-label classification works from the literature, which will enable future results comparison. Table 11 shows the main characteristics of these datasets. We may observe the variety of imbalanceness level in the datasets, which were chosen precisely in order to represent different levels of mean imbalance rate.

While Scene, Emotions and Yeast are the least imbalanced datasets, with a meanIR of 1.25, 1.48, and 7.20, respectively, FMA dataset is, by far, the most imbalanced one, with a meanIR of 1,403.81. Moreover, we may note that CAL500, Enron and Medical datasets also present a considerable imbalance level.

Table 11 – Characteristics of the datasets used in the experiments.

Dataset	*Domain*	*#Samples*	*#Labels*	*Cardinality*	*Density*	*MeanIR*	*MaxIR*
CAL500	Audio Tagging	502	174	26.044	0.150	20.58	88.8
Emotions	Music Sentiments	593	6	1.869	0.311	1.48	1.78
Enron	E-mail Analysis	1,702	53	3.378	0.064	73.95	913.00
FMA	Music Genres	90,410	160	2.4667	0.0153	1,403.81	21,897.00
Medical	Medical	978	45	1.245	0.028	89.50	266.00
Scene	Event Detection	2,407	6	1.074	0.179	1.25	1.46
Yeast	Biological	2,417	14	4.237	0.303	7.20	53.41

4.2.2 Algorithms and Parameters

For the classification task we used the eight previous presented algorithms: Binary Relevance (BR) (BRINKER; FÜRNKRANZ; HÜLLERMEIER, 2006), Label Powerset (LPS) (TSOUMAKAS; VLAHAVAS, 2007), Classifier Chains (CCH) (READ et al., 2009), Calibrated Label Ranking (CLR) (FÜRNKRANZ et al., 2008), Hierarchy of Multi-label Classifiers (HOMER) (TSOUMAKAS; KATAKIS; VLAHAVAS, 2008), Random k-labelsets (RAkEL) (TSOUMAKAS; KATAKIS; VLAHAVAS, 2011), MLkNN (ZHANG; ZHI-HUA, 2007) and BRkNN (SPYROMITROS; TSOUMAKAS; VLAHAVAS, 2008).

Table 12 shows the parameters used in the algorithms for the experiments, which were obtained using a grid search. The Problem Transformations algorithms were executed on SVM with a linear kernel and the complexity constant C equal to 1.

All experiments were conducted using five-fold cross-validation, applying the MLTL resampling technique only in the training folds. The Mulan[3] (TSOUMAKAS et al., 2011) open source library for multi-label learning was used to train and evaluate each of the classifiers.

[3] mulan.sourceforge.net

Table 12 – Parameter settings of the classifiers used in this work.

Algorithm	Parameters	
Binary Relevance	Base Classifier	SVM
BRkNN	Extension Type	None
	Neighbors	10
	Distance Function	Euclidean
Calibrated Label Ranking	Base Classifier	SVM
	Voting Mode	Standard
	Type of Output	Binary
Classifier Chain	Base Classifier	SVM
HOMER	Multi-Label Learner	Binary Relevance
	Base Classifier	SVM
	Number of Clusters	3
	Method	Balanced Clustering
Label Powerset	Base Classifier	SVM
MLkNN	Smooth	1
	Neighbors	10
	Distance Function	Euclidean
RAkEL	Multi-Label Learner	Binary Relevance
	Base Classifier	SVM
	Size of Subset	3
	Threshold	0.5
	Number of Models	314

Concerning the resampling algorithms, the resize rates were set to 25% (oversampling and undersampling). In MLSMOTE we used a ranking label combination and MLeNN used a threshold of 0.5, besides, both were set to work with five neighbors. In addition, the threshold parameter (TH) for the MLTL algorithm was calculated with Equations 4.3 and 4.4 for each one of the datasets.

4.2.3 Results and Discussion

In this section we present the discussion of the results achieved with the proposed MLTL method. In order to analyze the results from different perspectives, we split the results into five tables, which are divided according to the following criteria:

- MeanIR of the datasets before/after resampling (Table 13);
- Experimental results using the micro F-Score metric (Tables 14, 15, 16, 17, 18, 19 and 20);
- Wilcoxon statistical tests for the results considering the different resampling algorithms (Table 21); and
- Ranking of the classification algorithms using the Friedman test ranking (Table 22).

Following the same strategy proposed in Batista, Prati & Monard (2004) for the single-label scenario, besides testing the resampling algorithms by themselves, we have also tested the Tomek Link removal technique as a post-process cleaning step after applying the SMOTE resampling method. In our case, as we are dealing with a Multi-Label scenario, we have tested the use of the Multi-Label Tomek Link (MLTL), proposed in this work, after applying the Multi-Label SMOTE (MLSMOTE).

Table 13 – MeanIR of the datasets before/after apply the resampling methods.

Resampling Method	CAL500	Emotions	Enron	FMA	Medical	Scene	Yeast
Original (no resampling)	20.58	1.48	73.95	1,403.81	89.50	1.25	7.20
LPROS	15.25	1.35	47.23	582.83	50.32	1.21	4.86
LPRUS	17.82	1.44	50.64	689.90	55.45	1.23	7.01
MLROS	16.93	1.45	52.87	784.76	60.12	1.24	3.37
MLRUS	22.30	1.24	78.93	821.42	63.76	1.20	7.13
MLeNN	16.21	1.48	51.41	611.69	53.12	1.26	7.20
REMEDIAL	20.58	1.48	73.90	1,350.20	90.39	1.30	7.20
REMEDIAL-HwR-ROS	13.49	**1.15**	26.70	454.14	30.91	1.18	**3.13**
REMEDIAL-HwR-HUS	12.06	1.46	47.68	493.29	39.36	**1.15**	7.21
REMEDIAL-HwR-SMT	13.95	1.37	71.95	637.67	82.51	1.66	7.16
MLSMOTE	13.66	1.56	43.68	509.43	45.82	1.50	3.62
MLTL	*14.13*	*1.48*	*48.92*	*551.23*	*47.95*	*1.49*	*7.39*
MLSMOTE + MLTL	***5.21***	*1.57*	***25.84***	***437.72***	***28.34***	*1.52*	*3.66*

Lower imbalance ratios are highlighted in bold and the proposed methods results are in italic.

Table 13 shows the mean imbalance ratio (MeanIR) for the seven datasets after applying each one of the resampling techniques and the combination, hereafter referred to as MLSMOTE + MTLT. First, we may observe that as Emotions and Scene datasets already have low imbalanceness, none of the resampling methods changed much of their MeanIR. Second, we may note that REMEDIAL-HwR-ROS reduced Yeast imbalanceness from 7.20 to 3.13, followed by MLROS (3.37) and MLSMOTE (3.62). In relation to CAL500, Enron, FMA and Medical datasets, the combination of MLSMOTE with MLTL was by far the most successful resampling technique to reduce the mean imbalance ratio (MeanIR). It is important to observe that MLTL by itself was also effective, achieving satisfactory results in three of the datasets: CAL500, FMA and Medical.

The classification results are shown in Tables 14, 15, 16, 17, 18, 19 and 20. These results correspond to the micro-averaged F-Score, measured from the classification experiments after applying the resampling techniques. The values in italic represent the results for the proposed method, and the best results are highlighted in bold.

Concerning the evaluation metric, we choose to use F-score because it is still one of the most important and used metrics in the imbalanced learning literature, as we can see in works such as (CHARTE et al., 2014), (CHARTE et al., 2015a), (CHARTE et al., 2015b), (CHARTE et al., 2013) and (CHARTE et al., 2015c).

Observing the results from Tables 14, 15, 16, 17, 18, 19 and 20, three main questions are raised: (1) Which resampling methods improved the classification results? (2) Which

Table 14 – Experimental results for the CAL500 dataset.

Resampling	Classification Algorithm							
	BR	BRkNN	CCH	CLR	HOMER	LPS	MLkNN	RAkEL
Original (No Resampling)	0.3515	0.3091	0.3354	0.3541	0.3366	0.3354	0.3205	0.3354
LPROS	0.4823	0.3651	0.4821	0.4732	0.4712	0.4541	0.3615	0.4924
LPRUS	0.3625	0.3151	0.3522	0.3642	0.3656	0.3422	0.3312	0.3751
MLROS	0.4913	0.3625	0.5653	0.4545	0.5721	0.5653	0.3423	0.5413
MLRUS	0.3581	0.3042	0.3255	0.3106	0.3249	0.3255	0.3079	0.3255
REMEDIAL	0.3371	0.3167	0.3270	0.3077	0.2971	0.3126	0.3435	0.2951
REMEDIAL-HwR-ROS	0.3097	0.2676	0.2270	0.2537	0.2490	0.2451	0.2906	0.2503
REMEDIAL-HwR-HUS	0.2321	0.2331	0.1296	0.2137	0.1293	0.1084	0.2616	0.1293
REMEDIAL-HwR-SMT	0.2754	0.2425	0.1538	0.1565	0.1574	0.1499	0.2855	0.1542
MLeNN	0.3665	0.3169	0.3466	0.3393	0.3442	0.3466	0.3173	0.3466
MLSMOTE	0.3690	0.3268	0.3861	0.3772	0.3527	0.3858	0.3706	0.3839
MLTL	*0.3766*	*0.3518*	*0.3720*	*0.3528*	*0.3736*	*0.3720*	*0.3717*	*0.3720*
MLSMOTE + MLTL	*0.6303*	*0.4328*	*0.6123*	*0.6104*	*0.6002*	*0.6321*	*0.4526*	***0.6652***

Table 15 – Experimental results for the Emotions dataset.

Resampling	Classification Algorithm							
	BR	BRkNN	CCH	CLR	HOMER	LPS	MLkNN	RAkEL
Original (No Resampling)	0.5635	0.6123	0.5848	0.5885	0.5926	0.5407	0.6248	0.6210
LPROS	0.6116	0.6630	0.6114	0.6317	0.6216	0.6356	0.6795	**0.6814**
LPRUS	0.5593	0.5998	0.5638	0.5610	0.5503	0.5411	0.6188	0.5838
MLROS	0.6135	0.6504	0.6591	0.6276	0.6256	0.6639	0.6511	0.6395
MLRUS	0.5520	0.5660	0.5305	0.5808	0.5249	0.5196	0.5966	0.5846
REMEDIAL	0.3574	0.3638	0.4278	0.3734	0.4225	0.4015	0.3927	0.3250
REMEDIAL-HwR-ROS	0.5266	0.4554	0.4599	0.5144	0.5061	0.4772	0.4583	0.5111
REMEDIAL-HwR-HUS	0.3854	0.3625	0.3126	0.3510	0.3495	0.3193	0.3849	0.3490
REMEDIAL-HwR-SMT	0.3400	0.3132	0.3071	0.3047	0.3029	0.2972	0.3106	0.3056
MLeNN	0.5635	0.6423	0.5848	0.5885	0.5926	0.5407	0.6548	0.6210
MLSMOTE	0.5023	0.5116	0.4197	0.4265	0.4244	0.4073	0.5210	0.4265
MLTL	*0.5850*	*0.6450*	*0.6209*	*0.6209*	*0.6206*	*0.6209*	*0.6617*	*0.6409*
MLSMOTE + MLTL	*0.5361*	*0.5134*	*0.4208*	*0.4354*	*0.4262*	*0.4102*	*0.5375*	*0.4306*

Table 16 – Experimental results for the Enron dataset.

Resampling	Classification Algorithm							
	BR	BRkNN	CCH	CLR	HOMER	LPS	MLkNN	RAkEL
Original (No Resampling)	0.5496	0.2812	0.5378	0.5683	0.5270	0.4340	0.4670	0.5496
LPROS	0.6306	0.3277	0.6070	0.6408	0.6355	0.4959	0.5194	0.6306
LPRUS	0.5158	0.2753	0.4962	0.5414	0.4944	0.4182	0.4490	0.5158
MLROS	0.6678	0.4860	0.6640	0.6930	0.6626	0.5837	0.5932	0.6694
MLRUS	0.5259	0.2856	0.5147	0.5594	0.5417	0.4372	0.4806	0.5259
REMEDIAL	0.2135	0.1417	0.2465	0.2527	0.2333	0.2004	0.1010	0.1135
REMEDIAL-HwR-ROS	0.3043	0.1896	0.2828	0.2822	0.2925	0.2674	0.1394	0.2822
REMEDIAL-HwR-HUS	0.6244	0.5890	0.6742	0.4668	0.4912	0.6805	0.5896	0.6841
REMEDIAL-HwR-SMT	0.2564	0.0888	0.1849	0.1851	0.1907	0.1517	0.0733	0.1851
MLeNN	0.6489	0.5337	0.6438	0.4017	0.6259	0.5857	0.5621	0.6489
MLSMOTE	0.6052	0.3085	0.6041	0.6108	0.6012	0.4997	0.5108	0.6125
MLTL	*0.6494*	*0.5376*	*0.6520*	*0.4027*	*0.6290*	*0.5870*	*0.5677*	*0.6499*
MLSMOTE + MLTL	*0.6711*	*0.4841*	*0.6655*	***0.7012***	*0.6639*	*0.5886*	*0.5952*	*0.6704*

Table 17 – Experimental results for the FMA dataset.

Resampling	Classification Algorithm							
	BR	BRkNN	CCH	CLR	HOMER	LPS	MLkNN	RAkEL
Original (No Resampling)	0.0514	0.1442	0.0348	0.0515	0.1827	0.1646	0.1375	0.0964
LPROS	0.1422	0.2317	0.3987	0.1454	0.3986	0.3987	0.2105	0.2022
LPRUS	0.0921	0.1001	0.2289	0.0912	0.2289	0.2289	0.0944	0.1765
MLROS	0.1153	0.2122	0.3375	0.1167	0.3512	0.3636	0.2003	0.2088
MLRUS	0.0833	0.1039	0.2366	0.0821	0.2359	0.2549	0.0860	0.1634
REMEDIAL	0.0213	0.0461	0.1301	0.0209	0.1308	0.1219	0.0449	0.0765
REMEDIAL-HwR-ROS	0.1034	0.2045	0.3149	0.0937	0.3241	0.3520	0.1852	0.1828
REMEDIAL-HwR-HUS	0.0934	0.1176	0.2301	0.0937	0.2367	0.2391	0.1042	0.1321
REMEDIAL-HwR-SMT	0.1182	0.1620	0.3318	0.1152	0.3509	0.3214	0.1739	0.1632
MLeNN	0.1012	0.1284	0.2512	0.1054	0.2519	0.2512	0.1190	0.1543
MLSMOTE	0.1345	0.1717	0.3609	0.1376	0.3713	0.3409	0.1868	0.1775
MLTL	*0.1430*	*0.1520*	*0.3933*	*0.1423*	*0.3873*	*0.3470*	*0.1752*	*0.1553*
MLSMOTE + MLTL	*0.1530*	*0.2540*	*0.4170*	*0.1487*	*0.4231*	***0.4233***	*0.1988*	*0.1989*

Table 18 – Experimental results for the Medical dataset.

Resampling	Classification Algorithm							
	BR	BRkNN	CCH	CLR	HOMER	LPS	MLkNN	RAkEL
Original (No Resampling)	0.8130	0.5753	0.8127	0.6194	0.7944	0.7603	0.6648	0.8132
LPROS	0.8760	0.5477	0.8743	0.8771	0.8586	0.8536	0.6744	0.8761
LPRUS	0.7857	0.5539	0.7881	0.6093	0.7586	0.7400	0.6301	0.7853
MLROS	0.8358	0.5331	0.8347	0.8320	0.8252	0.7947	0.6086	0.8354
MLRUS	0.8348	0.6424	0.8360	0.8351	0.8349	0.8028	0.7126	0.8345
REMEDIAL	0.6373	0.4073	0.6337	0.3234	0.6536	0.5787	0.4920	0.6370
REMEDIAL-HwR-ROS	0.6722	0.3480	0.5046	0.5064	0.5052	0.4896	0.4275	0.5064
REMEDIAL-HwR-HUS	0.8713	0.6762	0.7714	0.2933	0.4913	0.7737	0.7524	0.7841
REMEDIAL-HwR-SMT	0.6452	0.3287	0.4151	0.3253	0.4127	0.3818	0.4807	0.4114
MLeNN	0.8776	0.7113	0.8808	0.8673	0.8469	0.8522	0.7800	0.8774
MLSMOTE	0.8532	0.5351	0.8387	0.8209	0.8241	0.8392	0.6412	0.8546
MLTL	*0.8780*	*0.7125*	***0.8812***	*0.8671*	*0.8478*	*0.8555*	*0.7817*	*0.8798*
MLSMOTE + MLTL	*0.8762*	*0.7132*	*0.8798*	*0.8794*	*0.8612*	*0.8610*	*0.7822*	*0.8804*

Table 19 – Experimental results for the Scene dataset.

Resampling	Classification Algorithm							
	BR	BRkNN	CCH	CLR	HOMER	LPS	MLkNN	RAkEL
Original (No Resampling)	0.6237	0.6984	0.6018	0.6380	0.6911	0.5881	0.7300	0.6237
LPROS	0.7617	0.7741	0.7506	0.7619	0.8038	0.7306	**0.7989**	0.7617
LPRUS	0.6339	0.6783	0.6344	0.6395	0.6771	0.5728	0.7311	0.6339
MLROS	0.6500	0.6992	0.6197	0.6632	0.7077	0.6298	0.7451	0.6500
MLRUS	0.6363	0.6964	0.6919	0.6933	0.6904	0.6919	0.7413	0.6919
REMEDIAL	0.5648	0.6635	0.5524	0.5705	0.6144	0.5541	0.6948	0.5648
REMEDIAL-HwR-ROS	0.5897	0.6704	0.6376	0.6369	0.6365	0.6339	0.7061	0.6361
REMEDIAL-HwR-HUS	0.6554	0.7234	0.7176	0.7207	0.7180	0.7176	0.7593	0.7176
REMEDIAL-HwR-SMT	0.4327	0.4428	0.3772	0.3915	0.3891	0.3676	0.5343	0.3888
MLeNN	0.6415	0.7138	0.6282	0.6462	0.7042	0.6103	0.7502	0.6415
MLSMOTE	0.4994	0.5039	0.4474	0.4541	0.4529	0.4357	0.5821	0.4532
MLTL	*0.6751*	*0.7442*	*0.7102*	*0.7305*	*0.7692*	*0.7002*	*0.7554*	*0.7502*
MLSMOTE + MLTL	*0.5994*	*0.6817*	*0.6755*	*0.6740*	*0.6735*	*0.6727*	*0.7173*	*0.6735*

Table 20 – Experimental results for the Yeast dataset.

Resampling	Classification Algorithm							
	BR	BRkNN	CCH	CLR	HOMER	LPS	MLkNN	RAkEL
Original (No Resampling)	0.5807	0.6355	0.5526	0.6165	0.6019	0.5355	0.6504	0.5812
LPROS	0.6532	0.6436	**0.6848**	0.6698	0.6764	0.6722	0.6562	0.6721
LPRUS	0.5810	0.6221	0.5507	0.6130	0.6016	0.5302	0.6373	0.5823
MLROS	0.6822	0.6611	0.6708	0.6807	0.6822	0.6563	0.6702	0.6671
MLRUS	0.5671	0.6146	0.5402	0.6072	0.5863	0.5281	0.6386	0.5677
REMEDIAL	0.4560	0.4141	0.3463	0.4462	0.3806	0.3290	0.4871	0.4560
REMEDIAL-HwR-ROS	0.4483	0.3820	0.4194	0.4296	0.4255	0.4145	0.3391	0.3929
REMEDIAL-HwR-HUS	0.4935	0.4271	0.3825	0.3858	0.3802	0.3591	0.4932	0.3849
REMEDIAL-HwR-SMT	0.3740	0.3938	0.3808	0.3769	0.3766	0.3641	0.3842	0.3772
MLeNN	0.5842	0.6347	0.5405	0.6194	0.6017	0.5327	0.6485	0.5846
MLSMOTE	0.5717	0.5932	0.5738	0.5761	0.5723	0.5722	0.6099	0.5794
MLTL	*0.5834*	*0.6371*	*0.6016*	*0.6074*	*0.5998*	*0.6016*	*0.6497*	*0.6348*
MLSMOTE + MLTL	*0.5917*	*0.6053*	*0.5845*	*0.5880*	*0.5870*	*0.5841*	*0.6179*	*0.5903*

is the best resampling method? (3) Which is the best algorithm for classification?

In order to answer the first two questions with statistical significance, we applied the Wilcoxon test stating as hypothesis that the F-score is higher after using each resampling method. The z-score and p-value outputs for the test are shown in Table 21.

Table 21 – Wilcoxon statistical tests for F-score results.

	Z-score	*p-value*
LPROS	-2.790	0.0026
LPRUS	0.274	0.6078
MLeNN	-1.312	0.0947
MLROS	-3.035	0.0012
MLRUS	0.064	0.5255
MLSMOTE	1.030	0.8485
REMEDIAL	3.625	0.9999
REMEDIAL-HwR-ROS	3.212	0.9993
REMEDIAL-HwR-HUS	1.123	0.8693
REMEDIAL-HwR-SMT	5.156	1.0000
MLTL	*-2.398*	*0.0083*
MLSMOTE+MLTL	*-2.109*	*0.0175*

Considering the p-value threshold as 0.05, we are able to answer the first question: LPROS, MLROS, MLTL and MLSMOTE + MLTL algorithms significantly improved the classification results, since the z-score is negative and the p-value is below the threshold. On the other hand, the tests were not able to define a real improvement in the results with the algorithms LPRUS, MLRUS, MLSMOTE and REMEDIAL, as their z-score were positive and their p-values crossed the threshold. Regarding MLeNN, even though the wilcoxon test concluded that the null hypothesis is not rejected, there is not enough evidence to claim that the method did not increase the results, since its z-scores are negative.

Furthermore, through the wilcoxon test, we may also conclude that, in general, MLROS is the most effective resampling method, since it achieved the lowest p-value and

z-score. However, it is important to observe that in the four most imbalanced datasets (CAL500, Enron, FMA, Medical), the best result was achieved using MLTL, either by itself or in combination with MLSMOTE. This fact indicates that, although the answer for the second question should be MLROS (because of the wilcoxon test), MLTL obtained success in the most imbalanced datasets.

It may be asked why the combination of MLSMOTE and MLTL significantly outperforms either MLSMOTE or MLTL in three datasets (CAL500, Enron and FMA). The reason for that can be grounded in the fact that, after applying MLSMOTE, the new synthetic samples were introduced somewhere along the feature space that made the classes groups become overlapped, i.e., some samples from the majority classes were invading the minority classes spaces or vice versa. Then, when MLTL is applied as a cleaning method, some of these noises were removed, leading to a better model built by the classifier. This same phenomenon was also identified in the single-label classification scenario by Batista et al. (BATISTA; PRATI; MONARD, 2004).

To answer the third question, i.e., to define the best classifier, we used the statistical evaluation protocol proposed in Charte et al. (CHARTE et al., 2015b). Using this protocol, we calculate the ranking principle from the Friedman statistical test, in which the algorithms classification results are ranked and an average rank is calculated for each dataset. Table 22 presents the results for this test. We may observe that, while for Emotions, Yeast and Scene datasets the best classifier was MLkNN, for CAL500, Enron and Medical datasets the best was BR classifier, and for FMA dataset it was HOMER. Furthermore, in general, considering all datasets the best overall classifier was RAkEL.

Table 22 – Ranking for the classification algorithms using the Friedman test.

	BR	BRkNN	CCH	CLR	HOMER	LPS	MLkNN	RAkEL
Emotions	4,85	3,38	5,31	4,46	5,54	6,15	2,15	3,85
Yeast	4,77	3,46	5,38	3,69	4,62	7,08	2,38	4,38
Cal500	2,54	6,31	3,77	4,69	4,54	4,38	5,15	3,54
FMA	7,38	4,77	2,54	7,46	1,69	1,77	5,23	4,77
Enron	2,46	7,38	3,62	3,08	3,92	6,00	6,38	2,62
Medical	2,15	7,69	2,77	4,92	4,54	5,15	6,23	2,46
Scene	5,92	2,31	5,92	3,92	3,69	7,08	1,15	5,00
Average Rank	*4,30*	*5,04*	*4,19*	*4,60*	*4,08*	*5,37*	*4,10*	***3,80***

Table 23 shows the time complexity of the resampling algorithms using the "big O" notation (CORMEN et al., 2009), i.e., considering the superior limit for the execution time. In Table 23, the variable n stands for the number of samples in the dataset, x is the number of samples which will be increased/decreased by the method, y is the number of labels in the dataset, s is the number of different labelsets and f is the number of features in the dataset. It is important to note that the proposed method (MLTL) was analyzed in two different contexts: as an undersampling method and as a cleaning step. We may

observe that, the most computationally costly algorithms are LPROS and LPRUS, since they have quadratic time complexities. On the other hand, we may observe that MLTL, when used as a cleaning step, is the least computationally costly technique, being only dependent on the number of instances in the dataset.

Table 23 – Time complexity of the multi-label resampling algorithms.

Resampling Method	**Time Complexity**
LPROS (CHARTE et al., 2013)	$O(n^2 s \log s)$
LPRUS (CHARTE et al., 2013)	$O(n^2 s \log s)$
MLROS (CHARTE et al., 2015a)	$O(yn + x)$
MLRUS (CHARTE et al., 2015a)	$O(yn + x)$
MLeNN (CHARTE et al., 2014)	$O(yn)$
MLSMOTE (CHARTE et al., 2015b)	$O(ynf)$
REMEDIAL (CHARTE et al., 2015c)	$O(y + n)$
REMEDIALHwR-HUS (CHARTE et al., 2019)	$O(yn)$
REMEDIALHwR-ROS (CHARTE et al., 2019)	$O(yn + x)$
REMEDIALHwR-SMT (CHARTE et al., 2019)	$O(ynf)$
MLTL (Undersampling)	$O(yn + s)$
MLTL (Cleaning)	$O(n)$

We can also observe that our results corroborate with previous work in the field done by Charte et al. (CHARTE et al., 2014), who stated that undersampling methods should not be applied to MLDs which are not truly imbalanced. Hence, since for the most imbalanced datasets (CAL500, Enron, FMA and Medical) the best results were achieved by using MLTL, we may consider our technique as a promising approach.

We may note that it is difficult to define a criterion for the use of MLTL by itself or MLSMOTE in combination with MLTL in a certain dataset, since the success of both techniques depends on the intrinsic characteristics of the dataset, such as the distribution of the samples features along the feature space.

Moreover, analyzing the advantages and disadvantages of using MLTL or MLSMOTE + MLTL, we may consider the performance factor as a possible issue. By using only MLTL, the processing time spent to build the classification model may be lower than using MLSMOTE + MLTL, since MLSMOTE technique may increase the number of instances in the dataset. On the other hand, looking at Table 13, we may note that one advantage of MLSMOTE + MLTL is that, in general, it reduced the datasets mean imbalanceness better than using only MLTL.

4.3 The Imb-Mulan Framework

Is is important to note that, to the best of our knowledge, the Multi-Label Classification Frameworks from the literature do not provide implementations for all the

multi-label resampling algorithms, such as the official Mulan library. Thus, in order to compare our proposed method with other state-of-the-art methods, all resampling methods were implemented and are freely available for free download as a Mulan extension in the GitHub [4], which may also be considered as a contribution of this Chapter.

4.4 Final Considerations

MLTL is a Multi-Label approach to the classic Tomek Link resampling algorithm. This method may be used as an undersampling technique or a post-process cleaning step. For the first case, MLTL is anchored by a multi-label imbalanceness measure (MeanIR) to find the samples from the majority classes which will be resized. Besides, MLTL leans on the *Adjusted* Hamming Distance to calculate the difference between the nearest neighbors labelsets.

Moreover, we also proposed a formula to calculate the threshold (TH), which MLTL uses to define whether or not two instances have different labelsets. This formula is also based in the MeanIR metric and the more imbalanced is the dataset the lower TH is, i.e., more samples from the majority classes will be removed.

The experimental results with seven well-known datasets from the literature showed that MLTL is a competitive method, reducing the imbalanceness and beating all resampling techniques in four of the datasets. Therewithal, these datasets in which MLTL performed the best are the most imbalanced ones (higher MeanIR), showing that our technique is successful in a valuable scenario.

The proposal of MLTL was an important step in this thesis project, since it allowed us to better comprehend two scenarios: (1) The imbalance learning problem in the multi-label classification context; (2) How to organize and define a new method to deal with the imbalanceness problem.

In order to define and analyze the imbalanceness problem in hierarchical datasets we have to define baselines experiments using the existing resampling techniques. Keeping this in mind, in the following chapter we present the first studies towards the application of resampling techniques in hierarchical datasets.

[4] https://github.com/rodolfomp123/imb-mulan

Chapter 5 Using Flat Resampling Techniques in Imbalanced Hierarchical Datasets

Although dataset imbalanceness in a well-known problem in the machine learning community, there are few works in the literature that studied the imbalanceness issue in Hierarchical Datasets somehow. Thus, as stated before, the main objective of this Thesis is to investigate the imbalanceness problem in Hierachical Datasets and propose novel resampling solutions. However, before proposing novel solutions, we have first to define baselines experiments for further analysis and comparison. In this chapter we provide an experimental analysis concerning the application of the existing binary/multi-class and multi-label resampling algorithms into hierarchical datasets with single and multi-paths, respectively, in order to give baseline results for the further proposed techniques.

It is important to observe that there are no concerns regarding the depth of the prediction at this point, i.e., the resampling algorithms do not distinguish hierarchical classification problems with partial or full depth of prediction, since they deal with the label paths ignoring their hierarchical structure. It means that the resampling methods will totally ignore that two labels can be somehow related (Ex.: A/B and A/B/C).

5.1 Binary Resampling in Hierarchical Datasets with Single Paths

Figure 39 presents the resampling schema for the application of binary resampling algorithms in single paths hierarchical datasets. This schema represents a baseline for the application of resampling algorithms in hierarchical classification datasets with single paths.

In this resampling scenario, the imbalanced hierarchical problem is handled by applying binary resampling techniques in the training set. Here, each label path of the sample is considered as an whole individual label. Thus, in order to be able to apply these binary resampling algorithms, in Phase 1 we first use an O-A-A or O-A-O approach together with the binary resampling algorithm. Then, in Phase 2 we use the resampled training to train the hierarchical classifier, creating the learning model. In Phase 3 we predict the test set labels with the learned model and in Phase 4 we analyze the prediction results with a hierarchical measure. In the example from Figure 39, three new samples were created by the binary resampling algorithm: S_x, S_y and S_z.

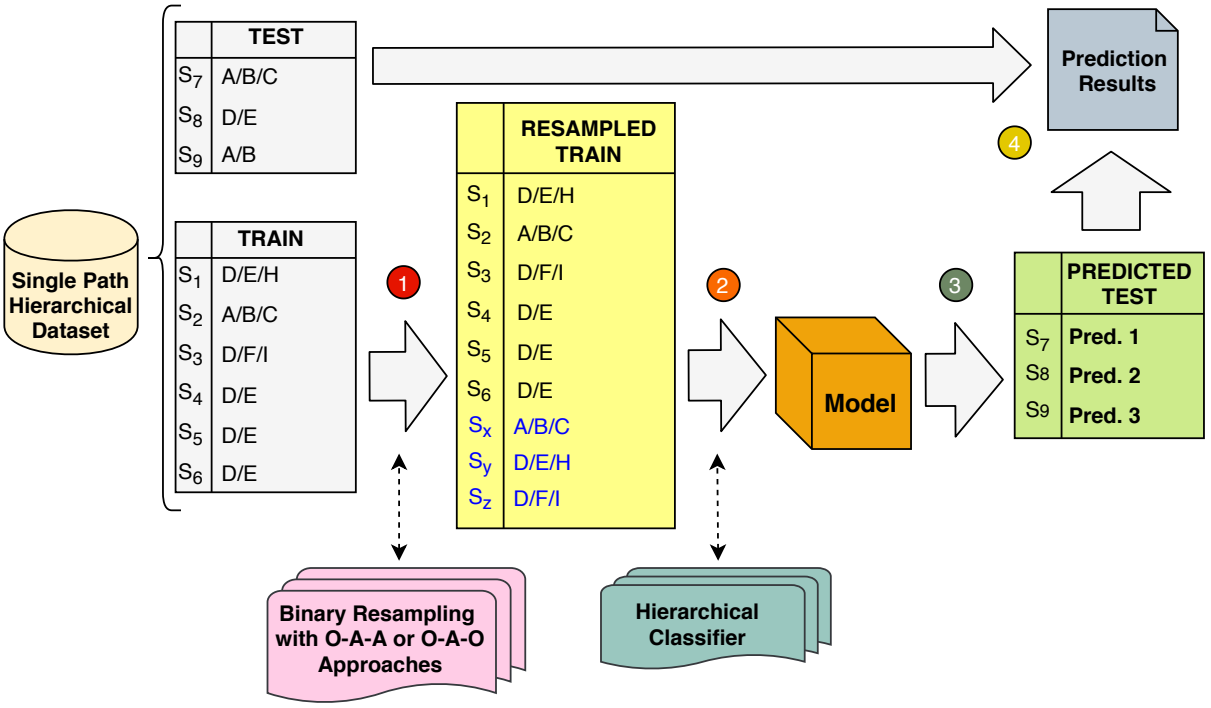


Figure 39 – Resampling schema for the application of binary resampling algorithms in Hierarchical problems with single paths.

5.2 Multi-Label Resampling in Hierarchical Datasets with Multiple Paths

Figure 40 shows the resampling schema used to apply multi-label resampling algorithms in hierarchical datasets with multiple paths. As well as in the previous schema, this resampling schema also represents a baseline for the application of resampling techniques in hierarchical classification problems, however considering only datasets with multiple paths.

In this scenario, the hierarchical problem is considered by the resampling algorithm as a multi-labelled classification problem. Here, each label path of the sample is considered as an whole individual label. Thus, in Phase 1 we apply the multi-label resampling algorithms in the training set. In Phase 2 we train the hierarchical classifier in the resampled training set, creating the learning model. In Phase 3 we predict the test set labels with the learned model and in Phase 4 we analyze the prediction results with a hierarchical measure. In the example from Figure 40, two new samples were created by the multi-label resampling method: S_x and S_y.

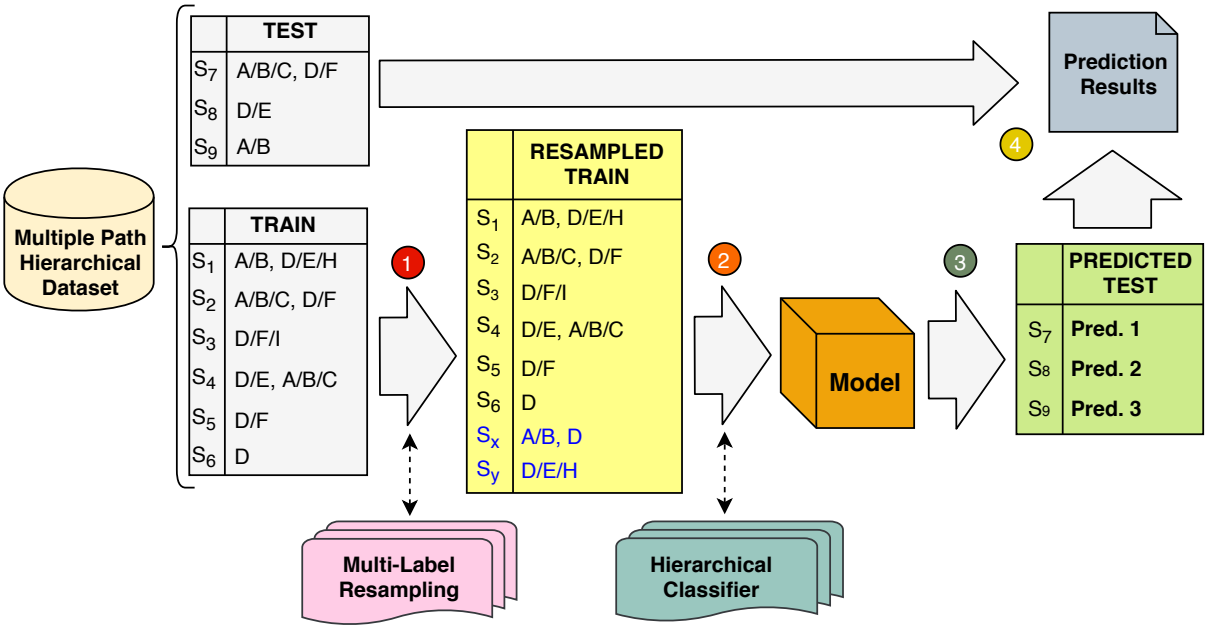


Figure 40 – Resampling schema for the application of multi-label resampling algorithms in Hierarchical problems with multiple paths.

5.3 Experimental Setup

In this section we present the datasets, algorithms and parameters, experimental setup and classification results regarding the proposed resampling algorithms.

5.3.1 The Datasets

In order to cover different aspects of hierarchical classification problems, we performed computational experiments in a total of 18 datasets: 11 with multiple paths and 7 with single paths.

Table 24 presents a detailed description of the datasets with Single and Multiple Paths used in the experimental analysis of this Chapter. The table is composed of the following columns (respectively): The name of the dataset; the multiplicity of paths in the dataset (single paths or multiple paths); the depth of the prediction of the problem (partial or full); the application domain; the number of samples in the training set; the number of the samples in the testing set; the number of attributes of the dataset; the number of labels/classes in the datasets. Looking at the datasets characteristics, we may observe that FMA-MFCC is the largest one, while Hglass is the smallest one. Furthermore, Hglass is the dataset with the lowest number of attributes, while Exp, Sequence and Diatoms datasets have the most.

Table 24 – Hierarchical Datasets with Single and Multiple Paths.

Name	Paths	Depth of Pred.	Domain	Train	Test	Attr.	Lbl	Reference
Cell-cycle	Multi	Partial	Biology	2484	1281	78	180	(CLARE; KING, 2003)
Church				2474	1281	24	180	
Derisi				2450	1275	62	180	
Eisen				1587	837	80	170	
Exp				2488	1291	544	180	
Gasch-1				2480	1284	174	180	
Gasch-2				2488	1291	53	180	
Phenotype				1009	582	64	168	
Sequence				2580	1339	437	180	
SPO				2437	1266	79	180	
FMA-MFCC	Single	Full	Audio	33259	14274	13	97	(DEFFERRARD et al., 2017)
FMA-SL-LBP				15331	7000	59	135	
Diatoms			Image	2065	1054	371	398	(DIMITROVSKI et al., 2012)
Actinopterygii			Biology	15705	6739	15	30	(PARMEZAN; SOUZA; BATISTA, 2018)
Diptera				15194	6528	33	29	
Instrument			Audio	6583	2836	30	46	
Hglass			Glass	144	70	9	11	(METZ et al., 2011)
ImCLEF07D			Image	10000	1006	80	24	(DIMITROVSKI et al., 2011a)

5.3.2 Algorithm and Parameters

For the hierarchical classification task we used the Clus-HMC framework. Clus-HMC was chosen because it is considered in the literature as the state-of-the-work hierarchical classification framework (CERRI; BARROS; CARVALHO, 2015; WEHRMANN; CERRI; BARROS, 2018; PEREIRA; GABRIEL; CERRI, 2019).

Clus-HMC is based on Predictive Cluster Trees (PCT) and generates a single Decision Tree (DT) considering an entire class hierarchy. In Clus-HMC, DTs are seen as a hierarchy of clusters where the root node contains all the training instances, while the remaining are recursively divided into smaller groups as the hierarchy is traversed towards the leaves. The classification is performed using a distance-based metric which calculates how similar an instance is to some tree.

The parameter configurations used in the algorithm were obtained after applying an extensive Grid Search, as proposed in Bergstra & Bengio (2012), and are reported in Table 25.

In the experiments, we have tested all the binary and multi-label resampling algorithms presented in Chapter 3 of this thesis. Moreover, the binary resampling algorithms were used with the O-A-A approach.

Table 25 – Clus-HMC execution parameters.

Parameter	Value
Type	Tree
ConvertToRules	No
HSeparator	"/"
FTest	[0.001, 0.005, 0.01, 0.05, 0.1, 0.125]
EnsembleMethod	RForest
Iterations	10
VotingType	Majority
EnsembleRandomDepth	No
SplitSampling	None
Heuristic	Default
PruningMethod	Default
CoveringMethod	Standard

5.4 Results

In order to evaluate the experimental results we have used the weighted-AUPRC measure. In all tables presented in this section, the values um bold represent the best result achieved for the given dataset.

Table 26 shows the experimental results for the hierarchical classification datasets with single paths before and after applying binary resampling. We may observe that Diptera was the only dataset in which the binary resampling method did not improved the classification results. Morever, we may also note that SMOTE+ENN was the most effective in improving the classification results in four out of the seven single paths datasets.

Table 26 – Results for the hierarchical classification datasets with single paths before and after applying binary resampling.

	FMA-SL-LBP	*Diatoms*	*Actinopterygii*	*Diptera*	*Instrument*	*Hglass*	*ImCLEF07D*
Original	0.3268	0.2582	0.7570	**0.6027**	0.7656	0.8145	0.7151
ROS	0.3278	0.2449	0.7407	0.5933	0.7529	0.8287	0.6924
SMOTE	0.3324	0.2630	0.7613	0.6014	0.7626	0.8389	0.7024
SMOTE-B1	0.3173	0.2596	0.7428	0.6012	0.7623	0.8324	0.7017
SMOTE-B2	0.3285	0.2613	0.7423	0.6003	0.7633	0.8228	0.7016
ADASYN	0.3144	0.2623	0.7433	0.5829	0.7624	0.8224	0.7028
RUS	0.2924	0.2433	0.7419	0.5724	0.7529	0.8024	0.6928
CC	0.3043	0.2436	0.7428	0.5833	0.7424	0.8014	0.7092
CNEN	0.3245	0.2328	0.7387	0.5925	0.7424	0.8276	0.7018
ENN	0.3234	0.2613	0.7498	0.5741	0.7529	0.8382	0.7016
RENN	0.3327	0.2640	0.7217	0.5833	0.7572	0.8382	0.7161
AllKNN	0.3148	0.2568	0.7388	0.5733	0.7372	0.8229	0.7014
NearMiss1	0.2722	0.2298	0.7383	0.5429	0.7176	0.7937	0.6724
NearMiss2	0.2890	0.2127	0.7314	0.5433	0.7226	0.7838	0.6882
NearMiss3	0.2789	0.2398	0.7228	0.5328	0.7228	0.7776	0.6874
TomekLinks	0.3283	0.2433	0.7424	0.5928	**0.7738**	0.8387	**0.7213**
SMOTE+ENN	**0.3328**	**0.2698**	**0.7626**	0.6023	0.7693	**0.8476**	0.7117
SMOTE+TL	0.3293	0.2617	0.7433	0.6005	0.7524	0.8279	0.7017

The classification results for the hierarchical classification datasets with multiple paths were divided into two tables. Thus, Tables 27 and 28 present the results before and after applying multi-label resampling. We may observe that, unlike the previous resampling context, in this resampling scenario the multi-label methods were not able to improve the classification results in five out of the eleven multiple paths datasets. Besides, we do not have one resampling method ahead of the other in terms of effectivity, since different techniques were able to improve the classification results in different datasets.

Table 27 – Results for the hierarchical classification datasets with multiple paths before and after applying multi-label resampling (Part 1).

	Cell-cycle	*Church*	*Derisi*	*Eisen*	*Expr*	*Gasch-1*
Original	0.1307	0.1222	0.1309	**0.1483**	**0.1606**	0.1544
LPROS	0.1232	0.1225	0.1213	0.1265	0.1473	**0.1584**
LPRUS	0.1117	0.1124	0.1134	0.1156	0.1499	0.1410
MLROS	0.1173	0.1211	0.1178	0.1133	0.1318	0.1467
MLRUS	0.1166	0.1138	0.1276	0.1276	0.1303	0.1418
MLeNN	0.1207	0.1297	0.1355	0.1265	0.1493	0.1418
REMEDIAL	0.1238	0.1198	0.1276	0.1195	0.1330	0.1430
MLSMOTE	0.1335	**0.1298**	0.1340	0.1419	0.1492	0.1576
MLTL	0.1325	0.1231	**0.1376**	0.1361	0.1497	0.1423
MLSMOTE+MLTL	**0.1355**	0.1237	0.1374	0.1448	0.1519	0.1434

Table 28 – Results for the hierarchical classification datasets with multiple paths before and after applying multi-label resampling (Part 2).

	Gasch-2	*Phenotype*	*Sequence*	*SPO*	*FMA-MFCC*
Original	**0.1410**	0.1256	**0.1683**	0.1342	**0.2803**
LPROS	0.1377	0.1150	0.1417	**0.1397**	0.2646
LPRUS	0.1343	0.1128	0.1481	0.1298	0.2569
MLROS	0.1330	0.1271	0.1483	0.1355	0.2570
MLRUS	0.1403	0.1224	0.1479	0.1316	0.2488
MLeNN	0.1365	0.1247	0.1472	0.1322	0.2645
REMEDIAL	0.1385	0.1233	0.1586	0.1336	0.2646
MLSMOTE	0.1353	**0.1264**	0.1593	0.1386	0.2728
MLTL	0.1344	0.1105	0.1583	0.1377	0.2733
MLSMOTE+MLTL	0.1339	0.1105	0.1672	0.1337	0.2615

5.5 Discussion

The discussion concerning the results presented in the previous section is grounded in these two questions: (i) Which binary resampling algorithms improved the classification results in the hierarchical datasets with single paths? (ii) Which multi-label resampling techniques improved the classification results in the hierarhical datasets with multiple paths? Both questions are answered with statistical significance in the following.

Which binary resampling algorithms improved the classification results in the hierarchical datasets with single paths?

In order to answer this question we have applied the Wilcoxon statistical test for each binary resampling algorithm stating that the classification results are different after applying each specific technique in the training sets. Table 29 presents the z-scores and p-values for the test results. Considering a threshold of 0.05 for the p-value, the only resampling technique that did changed the classification results with statistical significance was the combination of SMOTE with ENN.

Table 29 – Wilcoxon test for the hierarchical datasets with single paths.

	z-score	*p-value*
ROS	22.0	0.9119
SMOTE	9.0	0.1990
SMOTE-B1	20.0	0.8448
SMOTE-B2	18.0	0.7505
ADASYN	23.0	0.9359
RUS	28.0	0.9910
CC	28.0	0.9910
CNEN	25.0	0.9685
ENN	21.0	0.8816
RENN	16.0	0.6323
AllKNN	26.0	0.9787
NearMiss1	28.0	0.9910
NearMiss2	28.0	0.9910
NearMiss3	28.0	0.9910
TomekLinks	15.0	0.5671
SMOTE+ENN	3.0	0.0315
SMOTE+TL	17.5	0.7233

Which multi-label resampling algorithms improved the classification results in the hierarchical datasets with multiple paths?

This questions is also answered applying the Wilcoxon statistical test for each multi-label resampling algorithm stating that the classification results are different after applying each specific technique in the training sets. Table 30 shows the z-scores and p-values for the test results. Also considering a threshold of 0.05 for the p-value, we can observe that none of the resampling algorithms have statistically improved the classification results.

Table 30 – Wilcoxon test for the hierarchical datasets with multiple paths.

	z-score	*p-value*
LPROS	58.0	0.9869
LPRUS	66.0	0.9983
MLROS	61.0	0.9936
MLRUS	66.0	0.9983
MLeNN	57.0	0.9836
REMEDIAL	66.0	0.9983
MLSMOTE	42.0	0.7882
MLTL	55.0	0.9748
MLSMOTE+MLTL	52.0	0.9544

5.6 Final Considerations

In this Chapter we have presented the first approaches to deal with imbalanceness in hierarchical classification datasets. As we have different types of hierarchical datasets regarding the number of paths assigned to each sample, we have suggested two resampling schemas. The first one deals with the application of binary resampling algorithms using O-A-A or O-A-O approaches in the hierarchical datasets with single paths of labels. The second one is aimed in the application of multi-label resampling techniques in the hierarchical datasets with multiple paths of labels.

The resampling schemas presented in this Chapter are considered the baselines for the application of resampling algorithms in hierarchical classification, since we apply the existing resampling techniques in the hierarchical datasets with the only concern being the number of labels per paths in the dataset.

The experimental analysis with seven hierarchical classification datasets with single paths of labels and eleven with multiple paths of labels, supported by statistical significance, shown that the use of standard resampling approaches do not lead to improvements in the classification performance, mainly of the multiple paths problems. The bad performance of the schema may be explained by the fact that it does not considers the depth of the prediction of the problems, which is a crucial point in hierarchical classification problems.

In the next Chapters, we present specific approaches proposed in this Thesis in order to handle imbalanceness in hierarchical classification datasets considering different strategies and heuristics, which somehow consider the relationship between the label paths. In this context, the next Chapter present a label conversion strategy to deal with imbalanceness in hierarchical multi-label datasets with multi-label resampling algorithms.

Chapter 6 A Label Path Conversion Strategy for Imbalanced Hierarchical Datasets

In this chapter we present the first contributions towards the resampling of hierarchical datasets, which are:

- Two new metrics to measure imbalanceness in Hierarchical Datasets.
- A novel approach to deal with imbalanceness in hierarchical scenarios using Multi-Label Resampling Techniques.

The proposed approach is focused in the conversion of the hierarchical into a multi-label dataset, applying well-known multi-label resampling algorithms, and then converting the dataset back to its hierarchical taxonomy. The technique is detailed explained in the following sections.

It is important to observe that a paper describing the main findings of this Chapter were published at the 30^{th} IEEE International Conference on Tools with Artificial Intelligence (ICTAI) (PEREIRA; COSTA; SILLA JR, 2018).

6.1 Measuring Imbalanceness in HMDs

The measurement of imbalanceness in datasets, known as Imbalance Ratio (IR), is usually obtained by computing a ratio between the number of samples in the majority classes and the ones associated to the minority classes. A high IR leads to a highly imbalanced dataset (JAPKOWICZ; STEPHEN, 2002).

In the multi-label imbalanceness scenario, the authors of (CHARTE et al., 2013) proposed important metrics such as *IRLbl* and *MeanIR*. *IRLbl* measure calculates the IR for each label as a ratio between the frequency of the most common label in the labelsets and the frequency of the given label. Meanwhile, *MeanIR* measure the average level of imbalance in a Multi-Label Dataset (MLD) as the average *IRLbl*.

However, as shown in Figure 41, in Hierarchical Multi-Label Datasets (HMDs) the labels are represented as paths through the labels hierarchical taxonomy, instead of only sets of labels. Thus, we may define a IR measure for each label path. It is important to observe that in multi-label classification problems we consider as majority/minority

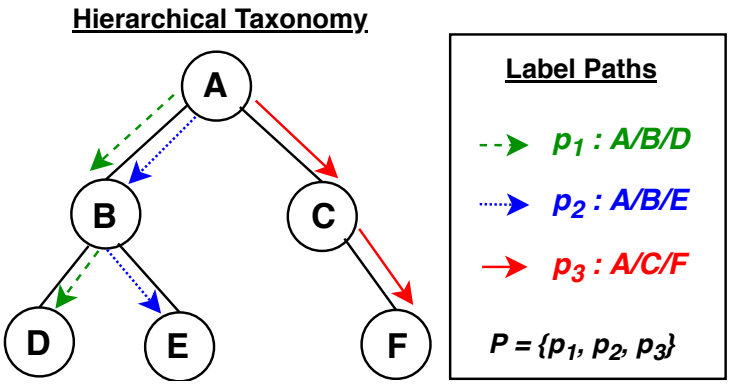


Figure 41 – Example of Label Paths in a Hierarchical Taxonomy.

labels the ones which appears the most/least in the samples. Nevertheless, in the following measures, we are proposing to consider as majority/minority the label paths, instead of individual labels.

Following the same line of reasoning of the authors from (CHARTE et al., 2013), in Formula 6.1 we define $IRLP(p)$, which represents the imbalance level of a certain Label Path p. In this context, P is the set of all possible Label Paths that has at least one occurrence in any samples, P_i is the *i-th* label path, and the dataset is represented as D.

$$IRLP(p) = \frac{\max\limits_{p' \in P}(\sum_{i=1}^{|D|} h(p', P_i))}{\sum_{i=1}^{|D|} h(p, P_i)}$$
$$h(p, P_i) = \begin{cases} 1, & p \in P_i \\ 0, & p \notin P_i \end{cases} \tag{6.1}$$

In Formula 6.1, the value is 1 for the most frequent Label Path, and a greater value for the others. The higher $IRLP$ is, the larger will be the imbalance level for the Label Path.

Formula 6.2 defines the Mean Imbalanceness of a Hierarchical Dataset ($HMeanIR$) based on the average between the imbalanceness per label path previously presented.

$$HMeanIR = \frac{1}{|P|} \sum_{p=P_1}^{P_{|P|}} IRLP(p) \tag{6.2}$$

6.2 Using Multi-Label Techniques to Deal with Imbalanceness

In order to give a general view, Figure 42 presents a visual schema of the proposed approach to tackle the imbalance problem in Hierarchical Multi-Label Datasets (HMD). As can be observed in Figure 42, the idea is mainly focused in three phases:

1. Convert the HMD into a Multi-Label Dataset (MLD);
2. Apply the well-known multi-label resampling algorithms; and
3. Convert the resampled ML dataset back to a HMD.

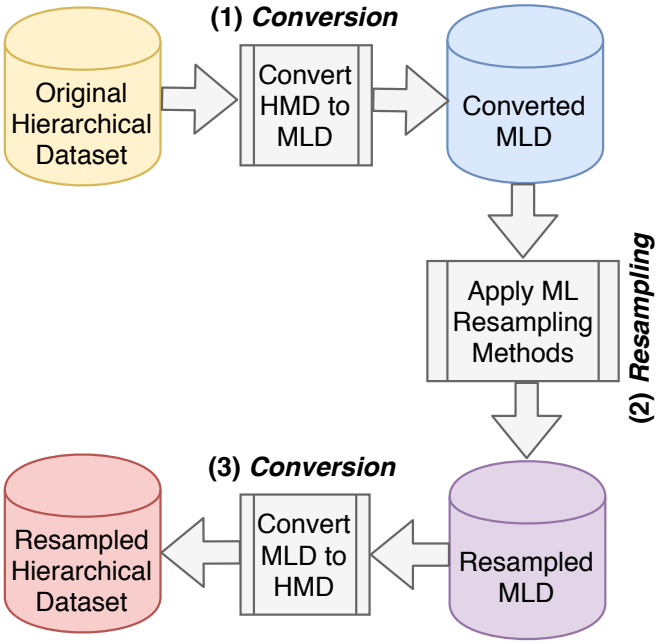


Figure 42 – Schema of the Proposed Resampling Approach.

Figure 43 shows an example of HMD↔MLD Conversion. In this example each S_n represents the n^{th} sample/instance from the dataset, the letters (A, B, C, D, E, F, and G) are the labels and the hierarchy between them is represented by "/". In the following subsections we give a detailed explanation regarding the HMD↔MLD conversion.

6.2.1 Hierarchical to Multi-Label Conversion

The first phase of Figure 42, i.e., the conversion from a HMD to a MLD, can be seen through the steps of Algorithm 33. The main idea is to group (for each instance) its labels paths into an unique labelset. For example, looking at Figure 43, we may note that instance S_1 has the hierarchy labels $\{A/B/E, A/B/D\}$ and it would be converted to a labelset $\{A, B, D, E\}$.

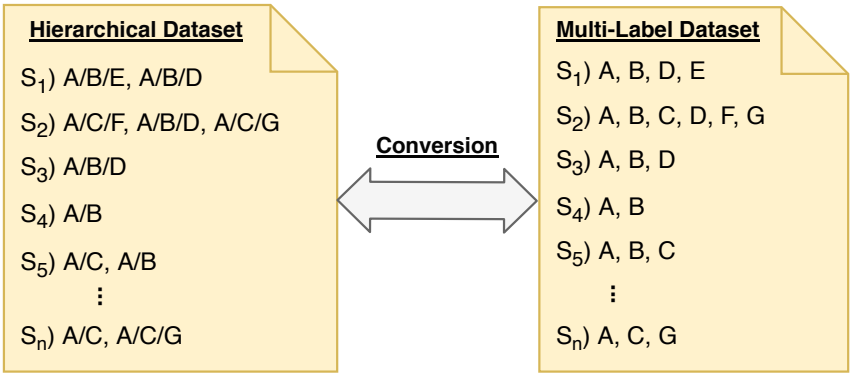


Figure 43 – Example of HMD↔ML Conversion.

Algorithm 33 HMD→MLD Conversion

Inputs: D: A Hierarchical Dataset
Output: D': A Multi-Label Dataset

1: $D' \leftarrow$ new empty list of instances
2: **for each** h_sample **in** D **do**
3: ml_sample$_{features} \leftarrow$ h_sample$_{features}$
4: labels $\leftarrow$ empty set
5: **for each** label_path **in** h_sample$_{labels}$ **do**
6: labels $\leftarrow$ labels $\bigcup$ label_path$_{labels}$
7: **end for**
8: ml_sample$_{labels} \leftarrow$ labels
9: $D' \leftarrow D' +$ ml_sample
10: **end for**
11: **return** D'

In details, Algorithm 33 works as follows: The main *for* loop (lines 2 to 10) pass through each one the instances (named *h_sample*) of the hierarchical dataset D. Then, in line 3, all features (excluding the labels) are cloned to the new sample (*ml_sample*), which will represents the converted sample. The internal *for* loop (lines 5 to 7) group each one of the individual labels in *h_sample* label paths, which are later set as *ml_sample* labels (line 8). Finally, in line 9, the converted sample is attached to the converted multi-label dataset D'.

6.2.2 Multi-Label to Hierarchical Conversion

The third phase of Figure 42, i.e., the conversion of the resampled MLD back to a HMD, is shown in Algorithm 34. The first important observation regarding this conversion is that the algorithm needs the label hierarchy as input (represented as H in Algorithm 34). The main idea is: For each label in the instance labelset, the algorithm will "walk through" the labels hierarchy, identifying the longest label path ending with the given label.

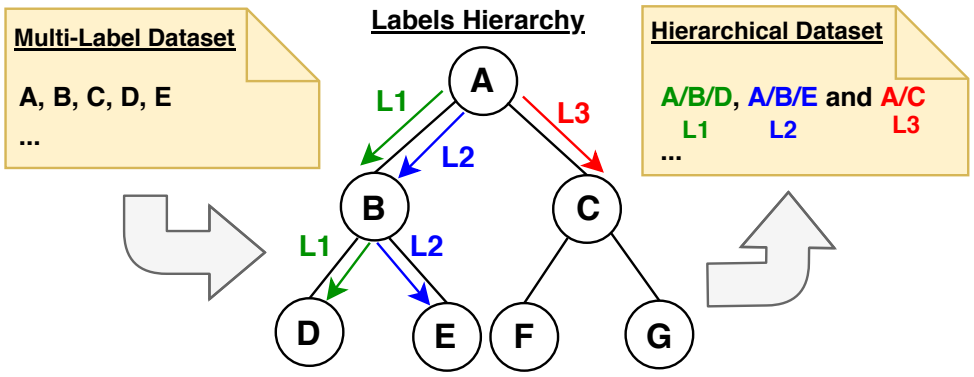


Figure 44 – Longest Paths Identification.

Figure 44 shows an example of the longest label path identification task (represented for line 6 in Algorithm 34). The paths are identified thought a pre-order traversal search in the labels hierarchy. In the example, LN shows the path to identify the Nth hierarchical label.

It is worth to highlight line 9 of Algorithm 34, which presents the *removeRepeatedPaths* function. The goal of this function is to remove the label paths already included in longer paths. For example, in Figure 44 the algorithm would create the following hierarchical labelset: $\{A, A/B, A/C, A/B/D, A/B/E\}$. However, we may note that the labels A and A/B are already contemplated in the label paths $A/B/D$, $A/B/E$. Thus, *removeRepeatedPaths* function is used to remove these labels.

Algorithm 34 MLD→HMD Conversion

Inputs: D: Multi-Label Dataset, H: Labels Hierarchy
Output: D': Hierarchical Dataset

1: $D' \leftarrow$ new empty list of instances
2: **for each** h_sample **in** D **do**
3: ml_sample$_{features} \leftarrow$ h_sample$_{features}$
4: labels $\leftarrow$ empty set
5: **for each** label **in** h_sample$_{labels}$ **do**
6: path $\leftarrow$ FINDLONGESTPATH(H, label)
7: labels $\leftarrow$ labels $\bigcup$ path
8: **end for**
9: labels $\leftarrow$ REMOVEREPEATEDPATHS(labels)
10: ml_sample$_{labels} \leftarrow$ labels
11: $D' \leftarrow D' +$ ml_sample
12: **end for**
13: **return** D'

6.2.3 Approach Limitations

It is important to observe that this proposed approach have two main limitations, which means that the conversions only work on hierarchical datasets in which: (i) There is the possibility of multiple paths, i.e., Hierarchical Multi-Label Datasets; (ii) The labels are disposed in a Tree taxonomy, i.e., will not work in datasets with Directed Acyclic Graph (DAG) taxonomy.

6.3 Experimental Evaluation and Discussions

In this section we present the dataset, parameters and configuration, results and discussion regarding the proposed method.

The dataset used in the experiments is the FMA-MFCC dataset, which is the same presented in Section 5.3.1. However, in the experiments presented in this Chapter, we have performed 10-fold cross-validation in the dataset, joining the training and test sets and subdividing them into folds. The features were extracted from the songs' audio signals and are one of the most well-known features for audio recognition tasks: The Mel Frequency Cepstral Coefficients (MFCC), which is a representation of the short-term power spectrum of a sound, based on a discrete cosine transform of the log power spectrum on a mel-scale frequency. More details can be found in (MERMELSTEIN, 1976).

The experiments were conducted using the Clus-HMC framework (TODOROVSKI; BLOCKEEL; DZEROSKI, 2002; BLOCKEEL et al., 2006; VENS et al., 2008). Clus is an algorithm to perform Hierarchical Multi-Label Classification based on predictive cluster trees. The main idea of the method is to build a set of classification trees to predict a set of classes, instead of only one class. To do this, the authors transform the classification output into a vector with Boolean components corresponding to the possible classes. They also need to take into account a distance-based metric to calculate how similar the training examples are in the classification tree. Clus-HMC was chosen for the task presented in this Chapter because it is considered the state-of-the-art hierarchical classification algorithm in some works from the literature (CERRI; BARROS; CARVALHO, 2015; WEHRMANN; CERRI; BARROS, 2018; PEREIRA; GABRIEL; CERRI, 2019).

Table 31 show the execution parameters of the Clus-HMC algorithm. These parameters were chosen after an exhaustive analysis in a set of experiments using the classifier in the dataset.

Table 31 – Clus-HMC execution parameters.

Parameter	*Value*
Model Testing Method	Cross-Validation
Number of Folds	10
Attributes Weights	Normalize
Model Minimal Weight	2
Tree Heuristic	Gain Ratio
Pruning Method	Default
Beam Size Penalty	0.1
Beam Width	10
Beam Max Size	Infinity
Constraints Syntactic	None
Constraints Max Size	Infinity
Constraints Max Error	Infinity
Constraints Max Depth	Infinity
WType	Exp Avg Parent Weight
WParam	0.75
Optimize Error Measure	Average AUROC
Classification Threshold	0.5
MEstimate	No

6.3.1 Results

Table 32 presents the values of the proposed imbalance measures for FMA before and after applying the proposed resampling approach for six different ML resampling techniques: LPROS, LPRUS, MLROS, MLRUS, MLSMOTE and MLeNN. In this Table, *MaxIRLP* represents the highest value of IRLP in the dataset. The best rate, i.e, the methods which achieved the lowest *HMeanIR* and *MaxIRLP* are highlighted in bold.

Table 32 – Imbalanceness features of FMA before/after resampling.

Dataset	*MaxIRLP*	*HMeanIR*
Original (No Resampling)	8,254	347.60
LPROS	10,291	**185.93**
LPRUS	**6,097**	262.14
MLROS	10,422	203.42
MLRUS	6,524	352.37
MLSMOTE	7,273	243.62
MLeNN	7,940	289.08

Table 33 shows the experimental results for the Averaged Area Under Receiver Operating Characteristic (AUROC) and the Weighted Averaged Area Under the Precision Recall Curve (AUPRC) metrics of the proposed approach using the same six ML resampling algorithms. The best result, i.e., the resampling technique which reached the best result using the proposed method, is highlighted in bold.

Table 33 – Experimental results for the proposed resampling approach.

	Avg. AUROC	*Avg. AUPRC (Weighted)*
Original (No Resampling)	0.5409	0.1898
LPROS	**0.7746**	**0.3842**
LPRUS	0.5375	0.1835
MLROS	0.7255	0.3550
MLRUS	0.5509	0.1910
MLSMOTE	0.6151	0.2587
MLeNN	0.5727	0.2018

6.3.2 Analysis and Discussion

In this section we give a detailed analysis concerning the experimental results performed in this work. In this discussion we mainly intend to answer three questions:

1. Which Multi-Label resampling technique was the most effective in reducing the imbalance on the HMD?
2. Which Multi-Label resampling technique achieved the best results in the HMD?
3. May the proposed *HMeanIR* measure the dataset imbalanceness?

The Imbalanceness Reduction

Firstly, looking at Table 32 we may observe that, with exception of MLRUS, all the ML resampling algorithms could reduce the dataset mean imbalanceness, calculated thought the proposed metric *HMeanIR*. Moreover, the techniques which reduced the imbalanceness the most were the oversampling methods. While LPROS decreased the dataset *HMeanIR* from 347.60 to 185.93, MLROS reduced it to 203.42, and MLSMOTE could decrease it to 243.62.

Secondly, we may note that LPRUS technique achieved the lowest result considering the *MaxIRLP* value. On the other hand, we may observe that the techniques which reduced the *HMeanIR* the most (LPROS and MLROS) were also the methods in which the *MaxIRLP* achieved the higher result. This apparently controversial result might be explained by the fact that, when the oversamplig methods create new instances to increase the number of samples in the minority classes, reducing its imbalanceness, it indirectly increases the imbalanceness of some label paths from the majority classes which were already imbalanced, generating high individual *IRLPs* for specific label Paths.

The Best Resampling Technique

The experimental results from Table 33 shown that using LPROS to resample the dataset before the classification phase improved the results from 0.5409 to 0.7746 (for Avg. AUROC measure) and from 0.1898 to 0.3842 (for Avg. AUPRC metric), i.e., LPROS could increase 0.2337 (AUROC) and 0.1844 (AUPRC) of the baseline result, proving to be the most promising resampling technique in this scenario.

In its turn, LPRUS was the unique technique that could not improve the classification results, decreasing the AUROC result (in relation to the original dataset) from 0.5409 to 0.5275 and AUPRC from 0.1898 to 0.1835. In fact, we may note that the best results were achieved by the oversampling techniques (LPROS, MLROS and MLSMOTE), which improved the classification results in at least 0.07 points.

The relation between *HMeanIR* and the imbalanceness

In order to analyze the relationship between *HMeanIR* and the dataset imbalanceness, we plotted *HMeanIR* rate and *AUROC / AUPRC* classification results side by side in Figure 45. Looking at the graphics, we may observe that there is a relationship between *HMeanIR* and the classification results. The three curves follow practically the same behavior for all resampling algorithms: When *HMeanIR* decreases, the results of AUROC and AUPRC increase.

Thus, the graphics suggest that: (1) The proposed *HMeanIR* metric can indeed measure the imbalanceness of the dataset; (2) The imbalanceness directly influences the classification results.

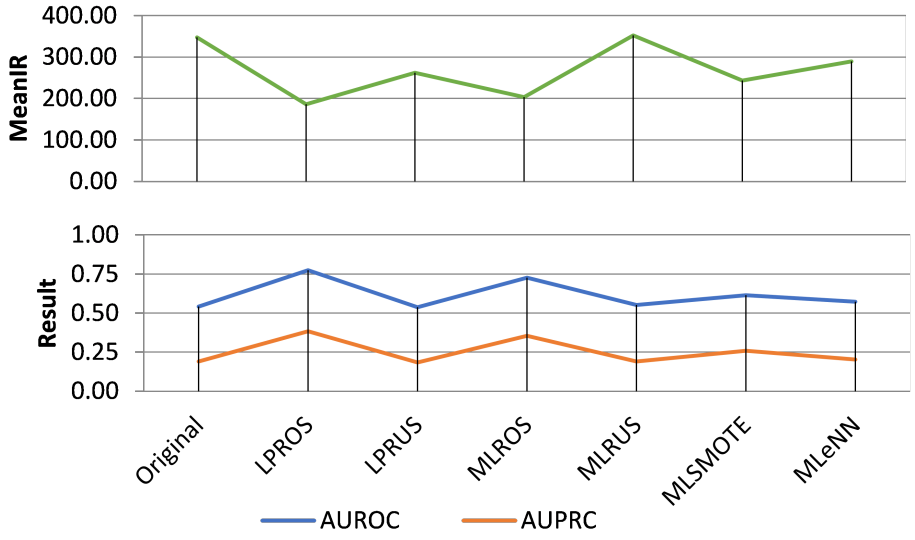


Figure 45 – Results Comparison Graphics.

6.4 Final Considerations

In this chapter we presented two metrics to evaluate how imbalanced a Hierarchical Multi-Label Dataset is. While the first metric (*IRLP*) is focused on calculating the imbalance level of a certain Label Path, the second metric (*HMeanIR*) calculates the general imbalance by averaging all the *IRLPs* from the dataset.

Moreover, we proposed a novel approach to deal with imbalanceness in Hierarchical Multi-Label Datasets. This method is aimed at converting HMD $\leftrightarrow$ MLD, in order to allow the use of well known Multi-Label resampling algorithms in the dataset to deal with the imbalanceness. The experimental results in one of the biggest hierarchical classification datasets from the literature showed promising results, in which its *HMeanIR* decreased from 347.60, using the original dataset, to 185.93, using the LPROS technique. Furthermore, the LPROS oversampling method increased the classification results from 0.5409 to 0.7746, i.e., an improvement of 0.2337 points using the Avg. AUROC metric, and from 0.1898 to 0.3842 using the Weighted Avg. AUPRC measure. Moreover, the lowest value of *MaxIRLPs* was achieved by LPRUS method.

Based on these results, there is still room for improvement by creating new strategies that deal specifically with the hierarchical taxonomy. In the next Chapter we present an approach to deal with imbalanced hierarchical classification datasets through the use of resampling algorithms during the training phase of different local classifiers approaches.

Chapter 7 Dealing with Imbalanced Hierarchical Datasets on Local Classification Approaches

Among the techniques to deal with hierarchical classification, the Local Classifiers are one of the most used approaches in the literature (SILLA JR; FREITAS, 2011). Taking these techniques into consideration, a study concerning the application of well-known resampling algorithms in these contexts was proposed and is presented in this chapter, which regards the following contributions:

- Three new metrics to measure imbalanceness in hierarchical classification datasets considering the different local classification scenarios.
- Novel approaches to deal with the imbalanceness in hierarchical classification problems using local classification approaches.
- A comprehensive set of experiments using the proposed approaches for eleven hierarchical classification datasets from the literature.

The contributions of this chapter are under evaluation for publication in the Data Mining and Knowledge Discovery Journal.

7.1 Measuring the Imbalanceness with Local Perspectives

In order to investigate if the dataset imbalanceness may be affecting the classifiers learning process, the calculation of a concrete imbalanceness level can be used. This measure is an useful information that can be used either to justify a classifier's bad performance or to help to choose a suitable classifier. Furthermore, this imbalance measure is commonly used by the resampling algorithms as a borderline information to decide which labels have to be tackled: the most imbalanced in case of an oversampling algorithm or the least imbalanced in case of an undersampling algorithm.

The measurement of imbalanceness in datasets is known as Imbalance Ratio (IR) and is usually obtained by computing a ratio between the number of samples in the majority classes and the ones associated to the minority classes. A high IR leads to a highly imbalanced dataset (JAPKOWICZ; STEPHEN, 2002).

In the binary and single-label classification scenarios, the dataset imbalaceness level can be computed by simply counting the number of samples belonging to each label of the dataset. While the most frequent label will be associated as the majority class, the less frequent is the minority class. However, in the multi-label scenarios, the possibility of association of more than one label per sample turns the IR calculation in a non-trivial task. Moreover, in the hierarchical scenario, besides the possibility of multiple labels per sample, their specific hierarchical taxonomy brings another challenge to the IR calculation task.

To solve the IR calculation issue in the Multi-Label scenario, Charte et al. (2013) proposed metrics such as Imbalance Ratio per Label (IRLbl) and Mean Imbalance Ratio (MeanIR). While the IRLbl measure calculates the IR for each label as a ratio between the frequency of the given label and the most common label in the labelsets, MeanIR is the average level of imbalance in a multi-label dataset as the average IRLbl.

The measures proposed by Charte et al. (2013) are fitted to work with multi-label dataset, but not hierarchically organized. To do so, in Pereira, Costa & Silla Jr (2018) we proposed a new metric, so-called Imbalance Ratio per Label Path (IRLP), in order to measure the imbalanceness in hierarchical datasets by adapting IRLbl to consider paths through the label's hierarchical taxonomy, instead of only sets of labels. In this case, the value 1 for IRLP represents for the most frequent Label Path. The higher the IRLP is, the larger the imbalance level for the given Label Path will be. Meanwhile, a HMeanIR was also proposed as the mean imbalance ratio for the hierarchical dataset.

Although the HMeanIR metric was experimentally tested and showed that is able to measure the imbalanceness of a hierarchical dataset, it may be considered as a global approach, since it measures the labels paths frequency in the dataset as a whole. In other words, it means that when using local classifiers to build the classification models per node or level (such as the ones presented in section 2.2), the IRLP metric will not be able to detect the imbalanceness in a local perspective. Given that the local classifiers approaches are one of the most used techniques to deal with hierarchical classification problems, it is necessary to define specific metrics to measure the imbalanceness in these local contexts. Thus, if we are able to identify when the imbalanceness issue is affecting the local classifiers, we may use the resampling approaches proposed in this work in order to tackle this problem.

Considering the previously described context, in the following subsections we present novel metrics that can be used to measure imbalanceness in hierarchical datasets taking into account the local imbalance information. The idea behind these measures is to create a mechanism that can summarize and quantify the imbalanceness in the subsets created in the training step of each local classifier, considering the different local approaches (LCN, LCPN or LCL) and policies (Fig. 9). Thus, we have defined three

different Imbalance Ratio equations: IR_{LCN}; IR_{LCPN}; and IR_{LCL}.

For all equations, let us consider D as the hierarchical classification dataset, p as the policy chosen to select the positive/negatives samples in order to build the local classification model, n as a node/label from the hierarchy, $|L|$ as the total number of nodes/labels from the hierarchy, S_j as the j^{th} instance of the dataset, n_i as the i^{th} node from the labels hierarchy, C_n as the set of immediate children of node n, C_{n_i} as the i^{th} immediate child of n, LV as the set of levels in the label hierarchy, and N_{lv} as the set of nodes of the level lv. Moreover, for all metrics the h formulas are used in order to identify if a certain sample S_j is labeled with the given label x when using the given local approach with that specific policy p.

7.1.1 Imbalance Metrics for the LCN Approach

The LCN approach is focused on building a classification model for each node on the labels hierarchy, which will decide whether or not a new sample belongs to that specific node. During the training step, the dataset is "binarized" using a chosen policy (as presented in Figure 9) and we may raise the following question: How imbalanced is each one of these binary subsets during the training phase? In the following we present a metric that can be used to measure these local imbalanceness. The LCN Imbalance Ratio for the node n with policy p, named IR_{LCN}, is defined as:

$$IR_{LCN}(n,p) = \frac{\max\limits_{x \in 0,1}(\sum_{j=1}^{|D|} h(S_j, x, p))}{\min\limits_{x \in 0,1}(\sum_{j=1}^{|D|} h(S_j, x, p))} \tag{7.1}$$

where:

$$h(S_j, x, p) = \begin{cases} 1 & \text{if } S_j \text{ is labeled with } x \text{ when using policy } p, \\ 0 & \text{otherwise.} \end{cases} \tag{7.2}$$

The goal of $IR_{LCN}(n,p)$ is to define the proportion of number of positive/negative samples that will be considered to build the binary classification model for the label node n when using the policy p. In other words, the higher $IR_{LCN}(n,p)$ is, the larger will be the imbalance level when building the classification model for label node n using policy p.

Furthermore, we may define the Mean Imbalance Ratio ($MeanIR_{LCN}$) when using the local classifiers per node approach with the policy p as the average between the $IR_{LCN}(n,p)$ for all label nodes:

$$MeanIR_{LCN}(p) = \frac{\sum_{i=1}^{|L|} IR_{LCN}(n_i, p)}{|L|} \tag{7.3}$$

7.1.2 Imbalance Metrics for the LCPN Ppproach

The LCPN approach builds a multi-class classification model for each parent node on the labels hierarchy, which will decide from which one of a node's children a new sample belongs to. During the training phase, the dataset is transformed into a series of multi-class subsets using a chosen policy (e.g: siblings or exclusive siblings) and we may raise the same question: How imbalanced is each one of these subsets? In the following we present a metric to measure these local imbalanceness. The LCPN Imbalance Ratio for the node n with policy p, named $IR_{LCPN}(n,p)$, is defined as:

$$IR_{LCPN}(n,p) = \frac{1}{|C_n|^2} \sum_{i=1}^{|C_n|} \frac{\sum_{j=1}^{|D|} h(S_j, C_n, p)}{\sum_{j=1}^{|D|} h(S_j, C_{n_i}, p)} \tag{7.4}$$

where:

$$h(S_j, x, p) = \begin{cases} 1 & \text{if p = sib. and } S_j \text{ is labeled with } x \text{ or a descendant of a label in } x, \\ 1 & \text{if p = exc. siblings and } S_j \text{ is labeled with a label } x \text{ (or in } x), \\ 0 & \text{otherwise.} \end{cases} \tag{7.5}$$

The goal of $IR_{LCPN}(n,p)$ is to define the proportions of numbers of samples labelled with children of node n when using the policy p that will be considered to build the multi-class classification model for node n.

The higher $IR_{LCPN}(n,p)$ is, the larger will be the imbalance level when building the classification model for label node n using policy p. It is important to observe that we may have two different policies to define the children of a node n in order to calculate the proportions: (1) Exclusive Siblings, which considers only the samples labelled with immediate children of node n; (2) Siblings, which considers samples labelled with the children of node n in addition to all its descendants.

Moreover, we may define the mean Imbalance Ratio when using the local classifiers per parent node approach with the policy p, named $MeanIR_{LCPN}$, as:

$$MeanIR_{LCPN}(p) = \frac{\sum_{i=1}^{|PN|} IR_{LCPN}(n_i, p)}{|PN|} \tag{7.6}$$

7.1.3 Imbalance Metrics for the LCL Approach

The LCL approach is aimed into building a multi-class classification model for each level of the labels hierarchy, which will associate a new sample to one label from

each hierarchy levels. Like the LCPN approach, during the training step, the dataset is transformed into a series of multi-class subsets and we may also raise the question: How imbalanced is each one of these subsets? In the following we present a metric to measure these local imbalanceness. The LCL Imbalance Ratio for the level lv, named IR_{LCL}, is defined as:

$$IR_{LCL}(lv) = \frac{1}{|N_{lv}|^2} \sum_{i=1}^{|N_{lv}|} \frac{\sum_{j=1}^{|D|} h(S_j, N_{lv})}{\sum_{j=1}^{|D|} h(S_j, n_i)} \tag{7.7}$$

where:

$$h(S_j, x) = \begin{cases} 1 & \text{if } S_j \text{ is labeled with label } x \text{ (or in } x\text{)}, \\ 0 & \text{otherwise.} \end{cases} \tag{7.8}$$

The idea of $IR_{LCL}(lv)$ is to define the proportions of number of samples labelled with the nodes in the level lv of the labels hierarchy that will be considered to build the multi-class classification model for level lv. The higher $IR_{LCL}(lv)$ is, the larger will be the imbalance level when building the classification model for level lv.

Therewithal, we may define the mean Imbalance Ratio when using the local classifiers per level approach with the policy p, named $MeanIR_{LCL}$, as:

$$MeanIR_{LCL} = \frac{\sum_{lv=1}^{|LV|} IR_{LCL}(lv)}{|LV|} \tag{7.9}$$

7.2 Proposed Approaches

In this Chapter we propose three different resampling approaches to couple with the three different strategies of hierarchical classification using local classifiers: The Local Classifier per Node; Local Classifier per Parent Node; and Local Classifier per Level.

We have visually designed resampling schemas for all the proposed approaches, which will be detailed in the following subsections. In all proposed schemas we are considering an arbitrary example dataset (presented in Figure 46). Furthermore, in all Figures, the blue colored samples exemplify the use of oversampling techniques, while red colors samples with a dashed line are used to exemplify the use of undersampling techniques.

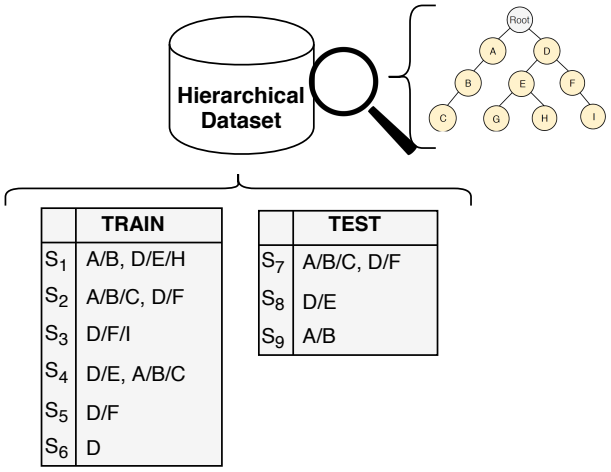


Figure 46 – An example of a hierarchical classification dataset subdivided into train and test.

7.2.1 Resampling Using the LCN Approach

The first resampling schema was designed for the Local Classifiers per Node approach (LCN), which according to (SILLA JR; FREITAS, 2011), is the most used hierarchical classification technique in the literature.

The main idea here is to resample each binarized dataset before building the classification model for each node. Figure 47 presents a general schema for the classification process using the proposed resampling step. In this figure, the orange cubes represent the classification models, which are built for each label node (M_x stands for the classification model of node x).

The classification schema is composed of three main steps: (1) Building one binary classifier per label node in the hierarchy; (2) Classifying the test dataset; (3) Measuring the classification results with a hierarchical measure. Even though these steps are already commonly used in order to classify a hierarchical dataset with the LCN approach, the first step is further subdivided into three substeps: (1.1) Applying a previously defined policy to choose the positive/negative samples when building the classification model for a given node n; (1.2) Applying a flat binary classification resampling algorithm in the binarized training dataset; (1.3) Using a flat single-label classification algorithm to build the classification model for node n. It is important to observe that the proposed approach is specifically embedded into the step 1.2, in which a binary resampling process is applied into the training dataset.

During the testing phase (step 2 of Figure 47), we use a top-down approach to predict the hierarchy of labels for a new sample, avoiding inconsistencies in class prediction at the different levels. It means that given an unknown sample, the idea is to walk down into the model tree predicting if the sample belongs to each label from the hierarchy. This

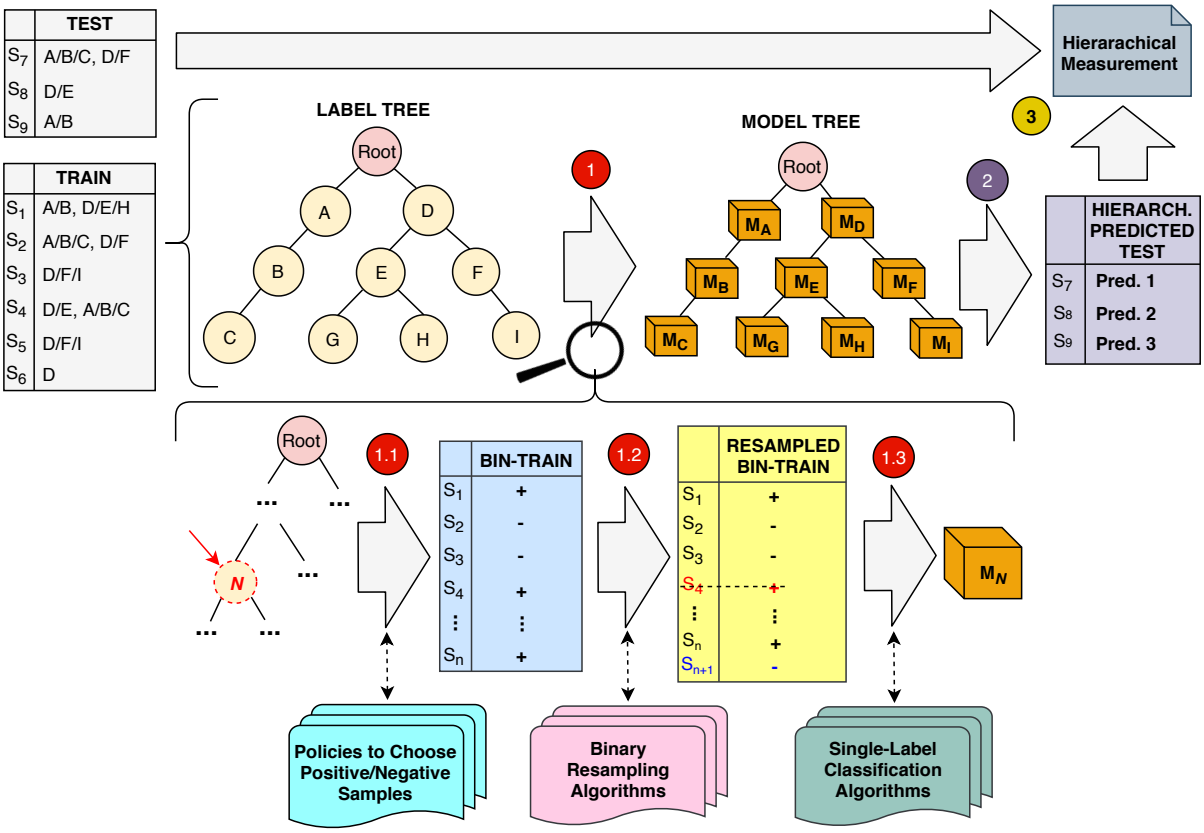


Figure 47 – The general classification schema for the LCN approach.

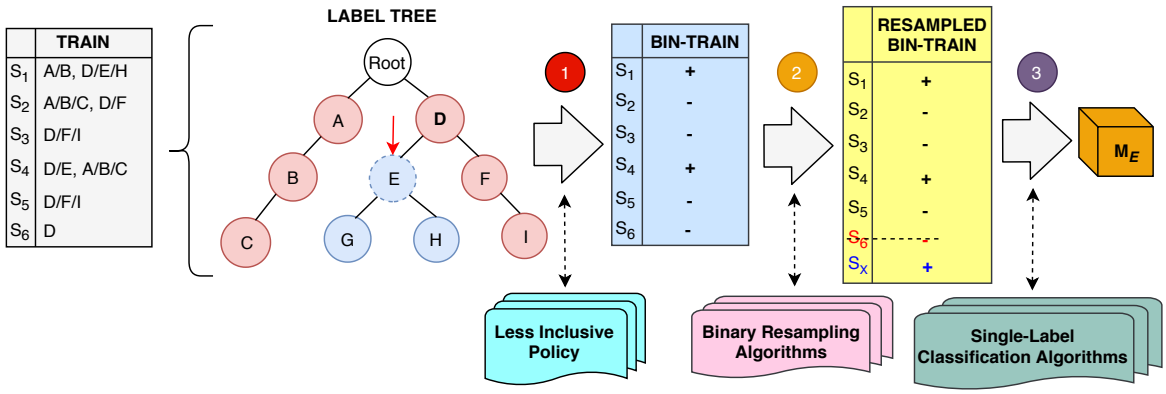
way, we have to use a threshold to define if we must consider a sample belonging to a certain label or not. It is important to note that we only keep moving down the next node of the model tree if the sample is labeled with the previous node.

In substep 1.1 of Figure 47 we may use six different policies to choose the positive/negative samples from the train dataset in order to build the classification model for a certain node n. To illustrate this process considering a certain policy, in Figure 48 we present two examples of the substeps 1.1, 1.2 and 1.3 for the Less Inclusive (Figure 48(a)) and Less Exclusive (Figure 48(b)) policies when applied to the node "E" of the example train dataset from Figure 46. It is important to observe that depending on the policy, some instances may not be used to train the model.

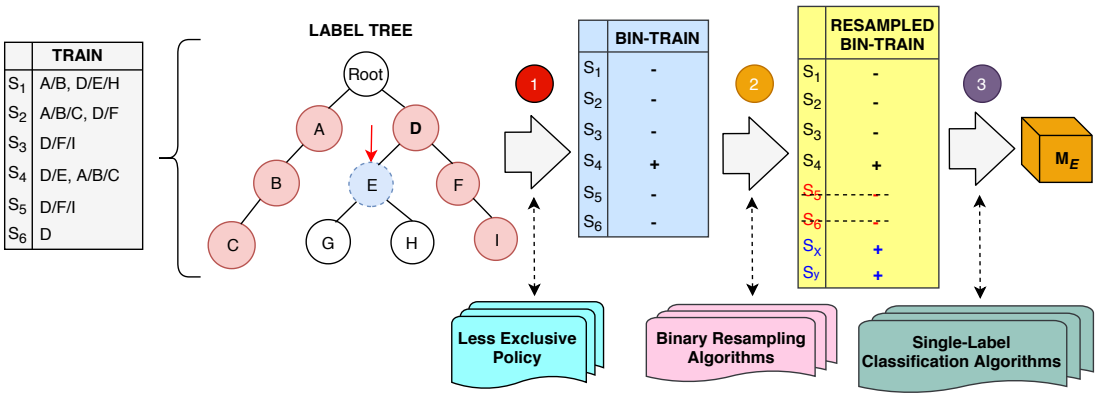
7.2.2 Resampling Using the LCPN Approach

This resampling schema is proposed to the Local Classifiers per Parent Node approach (LCPN), which consists into creating a multi-class classifier per parent node in order to distinguish between its child nodes.

The goal is to resample the multi-class datasets before building the classification



(a) Less Inclusive.



(b) Less Exclusive.

Figure 48 – Example of resampling schema for hierarchical datasets using LCN with two different policies.

model for each parent node. Figure 49 presents a general schema for the classification process using the proposed resampling step. In this figure, the orange cubes represent the classification models, which are built for each parent node of the label tree (M_{xy} stands for the classification model of node which is parent of x and y).

Similarly to LCN, the classification schema is also composed of three main steps: (1) Building a classification model for each parent node in the labels hierarchy; (2) Classifying the test dataset using a top-down approach; (3) Measuring the results with a hierarchical measure. Such as in LCN, there are three substeps in the model building phase, where in substep 1.1 a policy has to be chosen (in this scenario only siblings or exclusive siblings are allowed). The proposed resampling phase is also embedded into substep 1.2. It might be observed that, as the LCPN approach creates multi-class problems for each parent node and the classic resampling approaches are used to work in binary class problems, we have to apply an O-A-A or an O-A-O approach. These techniques decompose a multi-class classification problem into a series of binary sub-problems, so we can apply a binary resampling algorithm in each one of them. Finally, on substep 1.3, the parent node model is built considering a single-label classification algorithm.

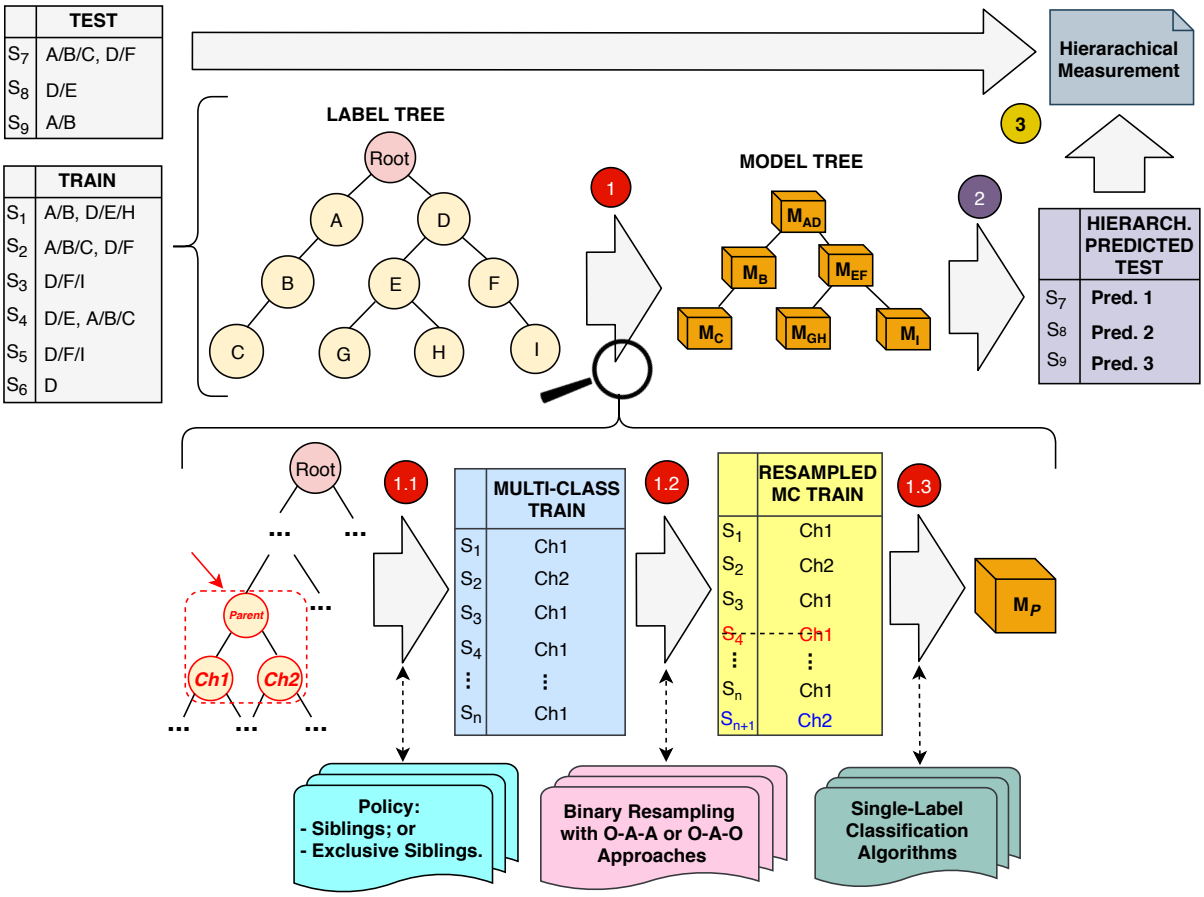


Figure 49 – The general classification schema using the LCPN approach.

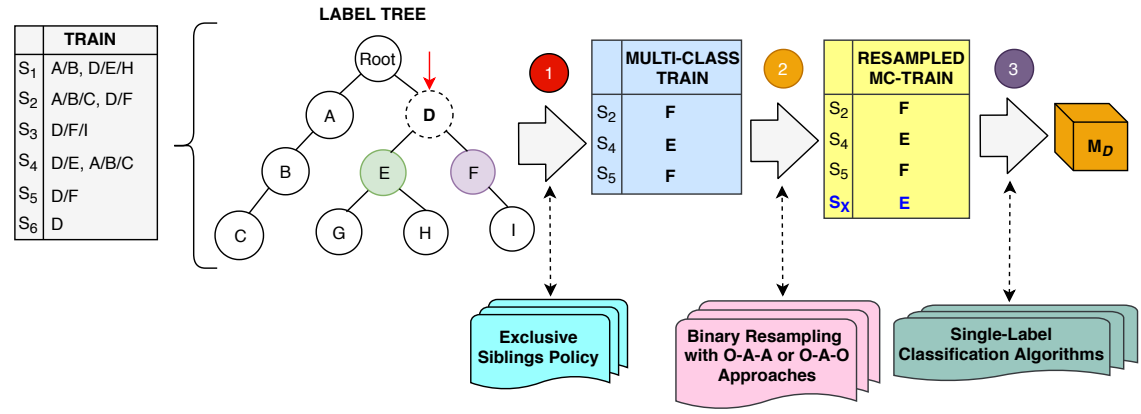
In substep 1.1 of Figure 49, differently to the LCN approach, we may use only two different policies to choose the samples from the train dataset in order to build the classification model for a certain parent node n. To illustrate this process considering the chosen policy, we present Figure 50, which shows examples of the substeps 1.1, 1.2 and 1.3 for the Exclusive Siblings (Figure 50(a)) and Siblings (Figure 50(b)) when applied to the parent node "D" from the example train dataset presented in Figure 46.

7.2.3 Resampling Using the LCL Approach

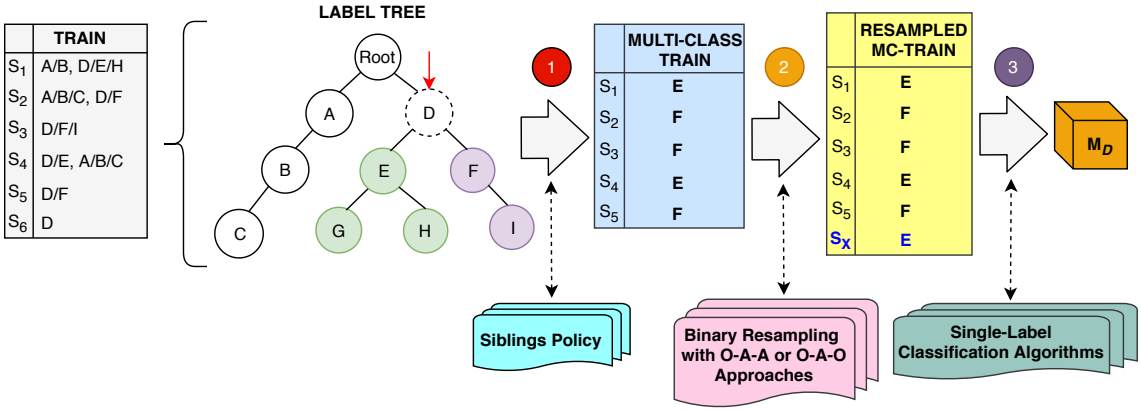
The third and last resampling schema is proposed to the Local Classifiers per Level approach (LCL), which consists into creating a multi-class classifier per level of the label's hierarchy.

The idea here is similar to the LCPN approach, in which we resample the multi-class datasets before building the classification model for each level. Figure 51 presents a general schema for the classification process using the proposed resampling step.

Similarly to the previously schemas, the classification is composed of three main



(a) Exclusive Siblings.



(b) Siblings.

Figure 50 – Example of resampling schema for hierarchical datasets using LCPN with the two different policies.

steps: (1) Building a classification model for each level in the labels hierarchy; (2) Classifying the test dataset; (3) Measuring the results with a hierarchical measure. The first important difference from the other approaches is that even using a top-down technique, the classification may predict labels with an inconsistency between the classes from different levels, which has to be removed later.

Step 1 is also subdivided into three substeps and, such as in LCN and LCPN approaches, the proposed resampling schema is embedded into substep 1.2. In this substep we have also an O-A-A or an O-A-O approach to decompose the multi-class classification problems per level into a series of binary sub-problems and then apply the classic binary resampling algorithms.

On the contrary of the LCN and LCPN approaches, we do not have different policies to apply on substep 1.1, since we must select all samples labelled with the labels from the level that we are building the classifier to. To exemplify the resampling process in LCL approaches, we present Figure 52, which shows an example of the substeps 1.1, 1.2 and 1.3 applied to the second level of the label hierarchy from the example train dataset presented in Figure 46.

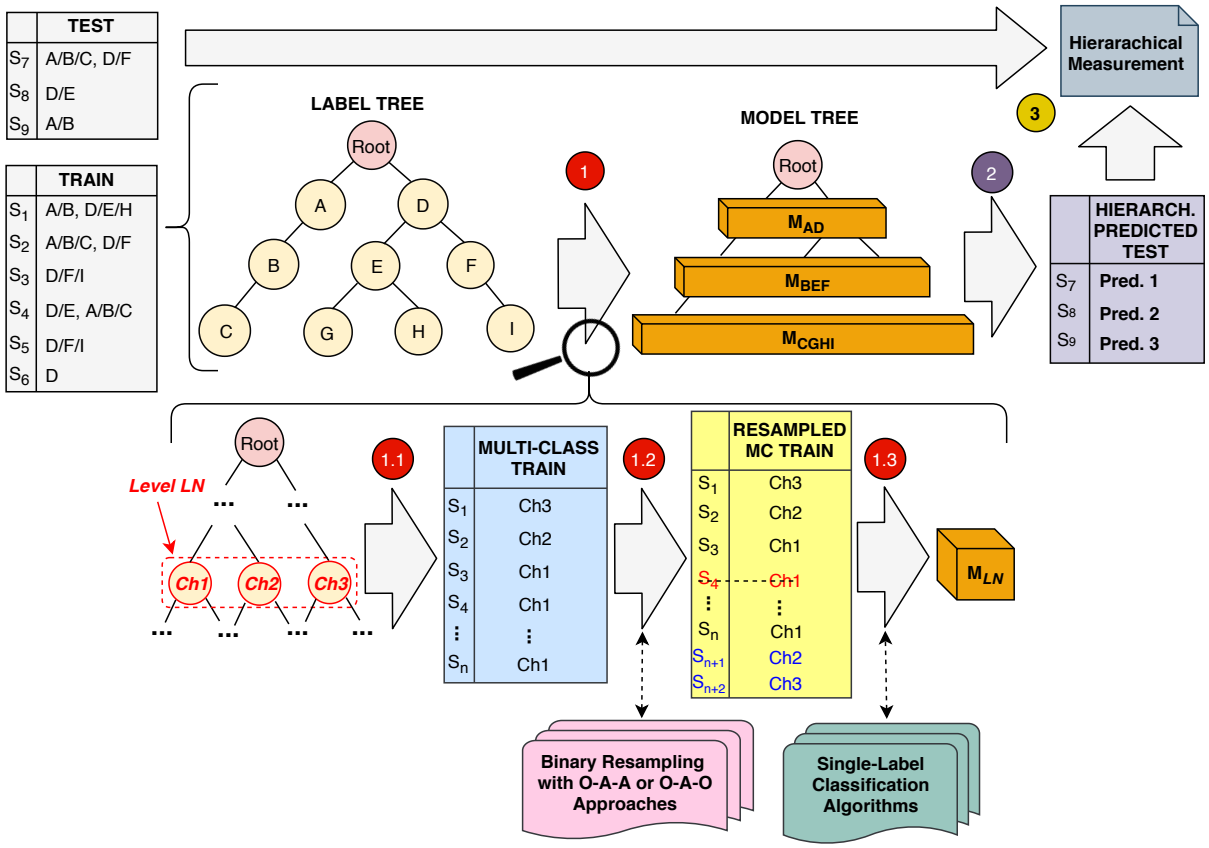


Figure 51 – The general classification schema using the LCL approach.

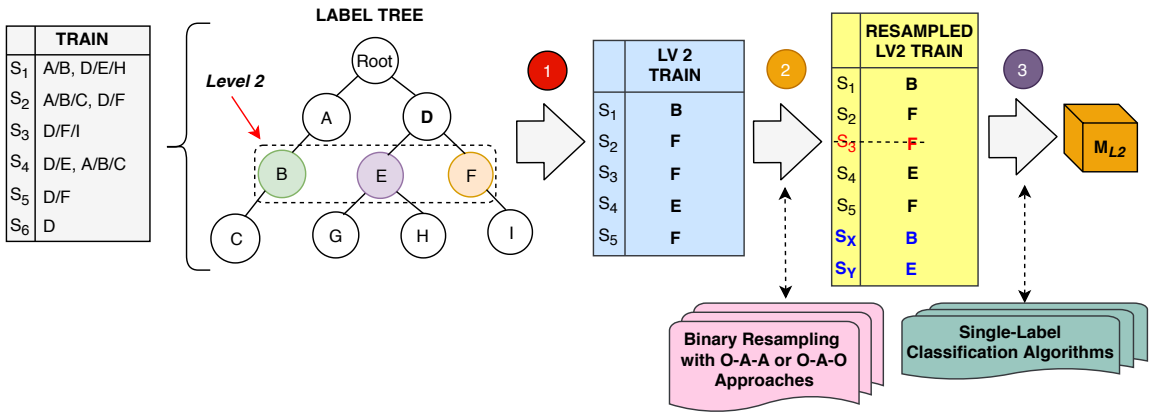


Figure 52 – Example of resampling schema for hierarchical datasets using LCL.

7.3 Experimental Protocol and Results

In this section we present the dataset, algorithms and parameters, imbalanceness measures, classification results and discussion regarding the experimental analysis of the proposed approaches presented in this work.

7.3.1 The Datasets

In the computational experiments, we used a total of eight datasets from four different domains: Biology, Music Information Retrieval, Image Classification and Text Categorization.

Table 34 shows a general review of these datasets. We may observe that DMOZ-2010 dataset is the largest one, while Eisen is the smallest one. Furthermore, DMOZ-2010 is by far the dataset with the largest number of attributes and labels.

Table 34 – General review of the datasets used in the experiments.

Name	Domain	#Samples	#Attr.	Labels	Depth	Reference
Cell-cycle	biological	1,711	78	180	4	(RUEPP et al., 2004)
Eisen	biological	1,163	80	170	4	(RUEPP et al., 2004)
Exp	biological	1,688	544	180	4	(RUEPP et al., 2004)
FMA-MFCC	music	90,393	13	161	4	(DEFFERRARD et al., 2017)
Gasch-1	biological	1,660	174	180	4	(RUEPP et al., 2004)
CLEF	image	10,000	80	97	3	(DIMITROVSKI et al., 2011b)
DMOZ-2010	text	128,710	381,580	12,294	5	(PARTALAS et al., 2015)
LSHTC-small	text	6,323	51,033	2,388	5	(PARTALAS et al., 2015)

The biological datasets, published in Ruepp et al. (2004) and frequently named as FunCat Datasets[5], describes a set of *Saccharomyces cerevisiae* fungus and were used in the experiments in order to predict the functional class of yeast, in which the classes were taken from the Munich Information Center for Protein Sequences (MIPS) functional catalog.

The musical dataset[6], firstly introduced by Defferrard et al. (2017), is composed of ninety thousand songs extracted from the Free Music Archive (FMA) repository, having thirteen attributes referred to the Mel-Frequency Cepstral Coefficients (MFCC) labeled with 161 genres.

We have used three datasets from the large scale hierarchical classification domain. The image dataset (CLEF)[7] consists of medical X-ray images from the ImageCLEF2009

[5] Available at http://sites.labic.icmc.usp.br/jeanmetz/datasets.html
[6] Available at https://github.com/mdeff/fma
[7] Available at https://www.imageclef.org/2009/medanno

annotation task. The DMOZ-2010 and LSHTC-small[8] are document datasets released from the Large Scale Hierarchical Text Classification challenges.

7.3.2 Proposed Approaches

For our proposed approaches presented in Section X, each approach (LCN, LCPN and LCL) are employed using Random Forest as the base classifier with the following binary resampling algorithms: Random Oversampling (ROS), Synthetic Minority Oversampling Technique (SMOTE), SMOTE Borderline-1 (SMOTE-B1), SMOTE Borderline-2 (SMOTE-B2), Adaptive Synthetic Resampling (ADASYN), Random Undersampling (RUS), Cluster Centroids (CC), Condensed Nearest Neighbors (CNEN), Edited Nearest Neighbors (ENN), Repeated Edited Nearest Neighbors (RENN), All k-Nearest Neighbors (AllKNN), NearMiss-1 (NM1), NearMiss-2 (NM2), NearMiss-3 (NM3), TomekLinks (TL), apply ENN after SMOTE (SMOTE+ENN) and apply TL after SMOTE (SMOTE+TL). Since we are dealing with multiple path prediction problems, it should be noted that for the testing phase of the LCPN and LCL approaches, we have adapted the classic top-down approach (DUMAIS; CHEN, 2000) to consider as output classes, all the classes that have a probability higher than a given threshold. In the reported experiments the value of 0.2 was used.

7.3.3 Baseline Approaches

In this Chapter we employ thirteen different approaches as baseline for the proposed methods:

- The first baseline approaches, named Top-Down (TD) are exactly the same local hierarchical classification approaches with the only different of not using any resampling method. Therefore, each local approach has its own baseline TD approach, hereafter named as TD-LCN, TD-LCPN and TD-LCL.
- We also employ a flat classification approach (Flat-ML) that uses the RAKEL (TSOUMAKAS; VLAHAVAS, 2007) multi-label classification approach with binary relevance.
- The last baseline approach, the Flat Classification with Multi-Label Resampling (Flat-MLRS) uses Flat-ML with the following multi-label resampling algorithms: Label PowerSet Random Oversampling (LPROS), Label PowerSet Random Undersampling (LPRUS), Multi-Label Random Oversampling (MLROS), Multi-Label Random Undersampling (MLRUS), Multi-Label Edited Nearest Neighbors (MLeNN),

[8] Available at http://lshtc.iit.demokritos.gr/

Resampling Multi-label Datasets by Decoupling highly Imbalanced Labels (REMEDIAL), Multi-label SMOTE (MLSMOTE), Multi-label Tomek Link (MLTL) and apply MLTL after MLSMOTE (MLSMOTE+MLTL).

7.3.4 State-of-the-art Approaches

In order to evaluate the performance of the proposed methods with respect to the state of the art, we have chosen to employ the following approaches:

- Clus-HMC: As presented in subsection 2.2, this is a global classifier based on Predictive Clustering Trees. This approach is used because represents the state-of-the-art hierarchical classification ensemble approach in the literature (NAKANO; LIETAERT; VENS, 2019).
- HMC↔ML: As presented in section 3, this is an indirect solution to the hierarchical imbalanceness issue, proposed earlier in (PEREIRA; COSTA; SILLA JR, 2018), which converts the hierarchical dataset to a multi-label dataset in order to apply well-known multi-label resampling algorithms and after resampling converts the dataset to a hierarchical dataset again. This method was used in conjuntion with Clus-HMC and all the multi-label resampling approaches also used in the Flat-MLRS approach.
- HierCost[9]: Presented in subsection 2.2, it is a cost-sensitive approach for the large scale hierarchical classification problem and can be considered as another state-of-the-art solution (NAIK; RANGWALA, 2018). This is a particularly interesting approach since it uses a cost-sensitive solution according to the label distribution,therefore dealing with the imbalanceness issue in a different approach.

7.3.5 Parameters and Configurations

The parameter configurations used in the classification algorithm were obtained after applying a Grid Search, as proposed in (BERGSTRA; BENGIO, 2012), and are reported in Table 35.

All classification experiments were conducted using a five-fold cross-validation scheme over the exact same folds. Moreover, the Random Forest and RAkEL algorithms were executed using the implementations from the scikit-learn library[10]. For the binary

[9] Available at https://cs.gmu.edu/~mlbio/HierCost/
[10] Available at http://scikit-learn.org/ and http://scikit.ml/

Table 35 – Parameter settings of the Classification algorithms.

Algorithm	Parameter	Value
Random Forest	Max Depth	None
	Max Leaf Nodes	None
	Number of Estimators	50
	Bootstrap	True
	Criterion	Entropy
	Min Samples Leaf	1
	Max Features	3
	Min Samples Split	2
RAkEL	Multi-Label Learner	Binary Relevance
	Base Classifier	SVM
	Size of Subset	3
	Threshold	0.5
	Number of Models	314
Clus-HMC	Type	Tree
	ConvertToRules	No
	FTest	[0.001, 0.005, 0.01, 0.05, 0.1, 0.125]
	EnsembleMethod	RForest
	Iterations	10
	VotingType	Majority
	EnsembleRandomDepth	No
HierCost	Cost Type	Exponetiated Tree Distance
	Imbalance	Yes
	Regularization Parameter	1

resampling tasks we used the implementations of imbalance-learn library[11], while for the multi-label resampling tasks we used the implementation of Imb-Mulan.

Concerning the configurations for the resampling algorithms, the resize rates were set to 25% (oversampling and undersampling), which is a common and well-tested value in the imbalance learning community (HAIXIANG et al., 2017). The Multi-class resampling experiments were made through an O-A-A design. In MLSMOTE we used a ranking label combination and in MLeNN we used a threshold of 0.5. To complete, all the resampling algorithms that used a nearest neighbors technique in their internal logic were set to work with three neighbors, which is also a common pattern in the resampling community.

7.3.6 Hierarchical Local Imbalanceness Metrics Results

In this section we present the results for the imbalanceness metrics proposed in section 2 of this work. In order to support a better visualization, the results of the proposed imbalanceness metrics are summarized in charts, in which x-axis represents the resampling algorithms and y-axis represents the IR metrics. These measures can help to comprehend the classification results, since the difficulty in the learning process may be associated with the dataset imbalanceness.

[11] Available at https://github.com/scikit-learn-contrib/imbalanced-learn

In Figure 53 we show eight charts, one for each dataset, plotting the IR_{LCN} values before (original dataset) and after applying each resampling algorithm using the proposed approach to perform LCN. Each chart plots the six different policies to select the samples in different formats and colors. We may observe that the exclusive siblings policy generates the lowers IRs, while the exclusive policies tends to generate the higher IRs.

On a similar way, Figure 54 shows the charts for the eight datasets in relation to the values of IR_{LCPN} metric before (original) and after applying each resampling algorithm using the proposed approach to perform LCPN. The charts also shows the different IR_{LCPN} values that are generated with the two different policies (Exclusive and Exclusive Siblings).

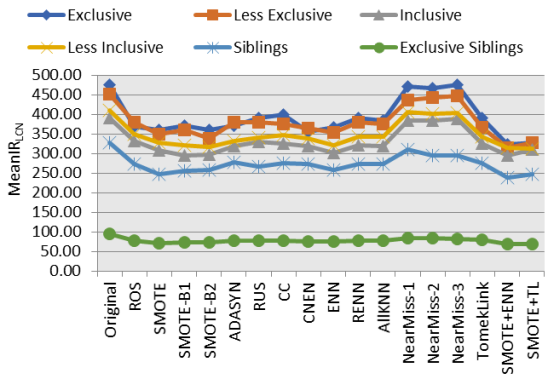
Finally, in Figure 55 we show the eight charts for the IR_{LCL} measures before (original) and after apply the resampling algorithms using the proposed approach to perform LCL.

Looking at the charts, it is important to observe the variations of the measures results when using the different resampling approaches. One can notice that, in general, the resampling algorithms involving the SMOTE techniques (SMOTE, SMOTE-B1, SMOTE-B2, SMOTE+ENN and SMOTE+TL) were the most efficient in reducing the imbalanceness of the hierarchical datasets. Moreover, we may also observe that the different policies can directly impact the imbalancenes factor of the datasets. For instance, the Exclusive Siblings policy seems to generate the lower imbalanceness in the datasets, which is logical since it uses the lowest number of samples to build the training sets during the local classification.

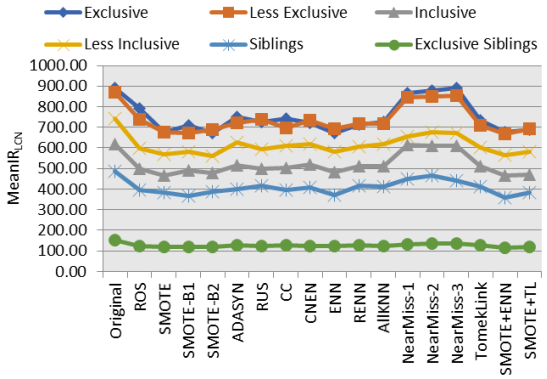
7.3.7 Classification Results

Following the same reasoning of the subsection 6.3, in this section we will only present a summarized version of the best results achieved with each one of the proposed approaches and related techniques. All the detailed results are presented in the Appendix A of this Thesis.

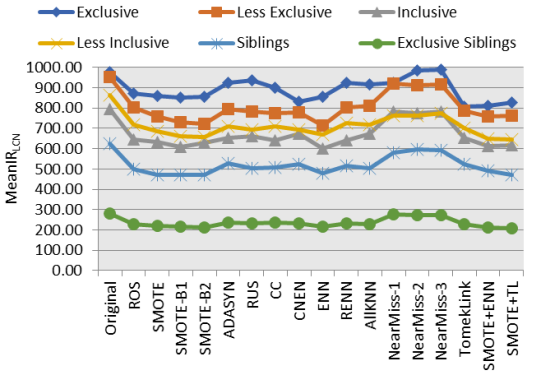
It is important to observe that all classification results were measured with the Hierarchical F-Score metric, as suggested in the study of Sokolova, Japkowicz & Szpakowicz (2006). The Figure 56 shows the best F-Score results for each technique and dataset. The information bellow the method's name and inside the brackets represents the configuration used to achieve the best result. We may observe that our proposed approaches outperformed the related techniques in half of the datasets: cell-cycle, FMA MFCC, Gasch-1 and CLEF.



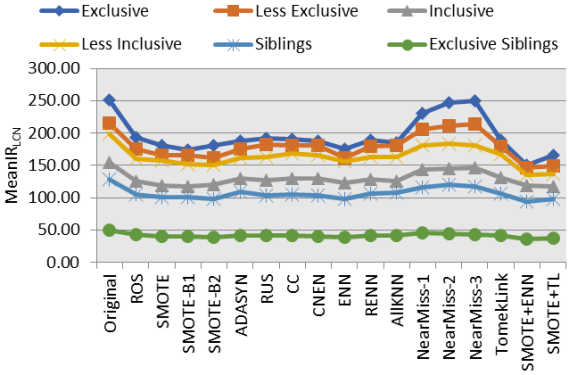
(a) Cell-cycle.



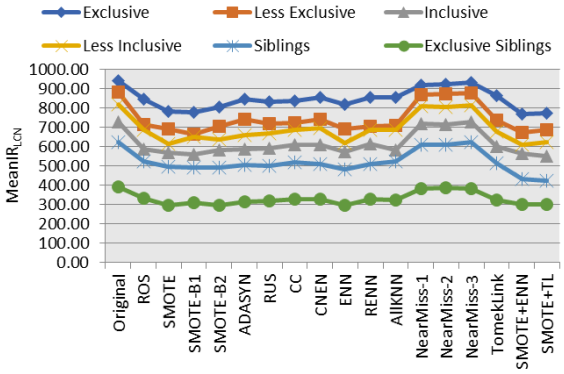
(b) Eisen.



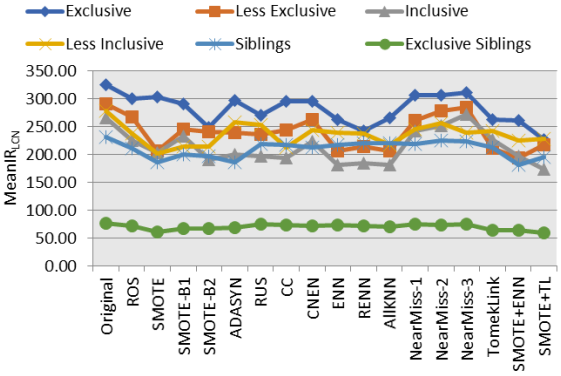
(c) Exp.



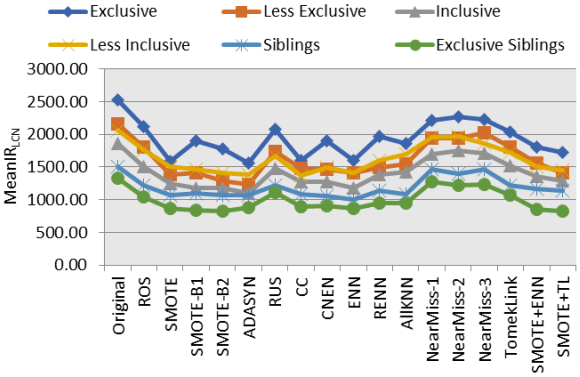
(d) FMA MFCC.



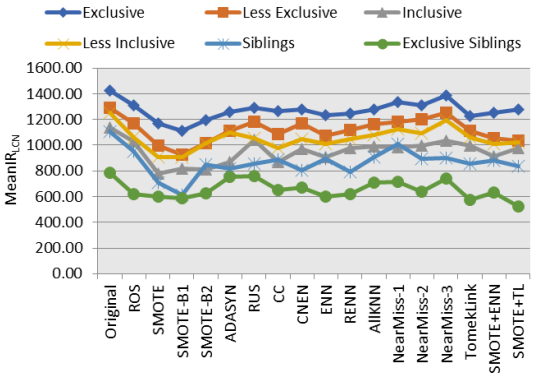
(e) Gasch-1.



(f) CLEF.

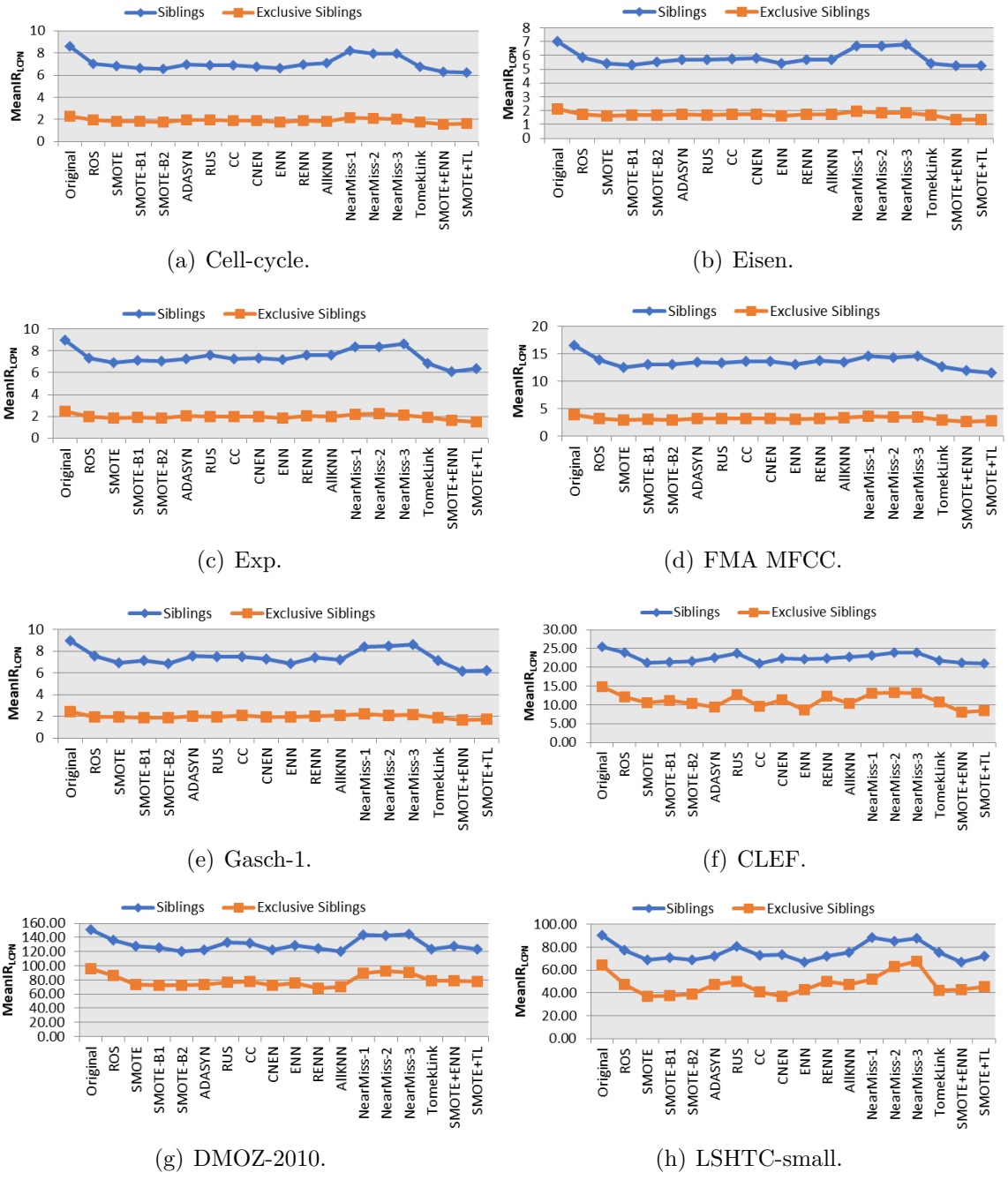


(g) DMOZ-2010.



(h) LSHTC-small.

Figure 53 – Mean Imbalance Ratio for Local Classifiers per Node (IR_{LCN}) for each dataset before (original dataset) and after resampling.



(a) Cell-cycle.

(b) Eisen.

(c) Exp.

(d) FMA MFCC.

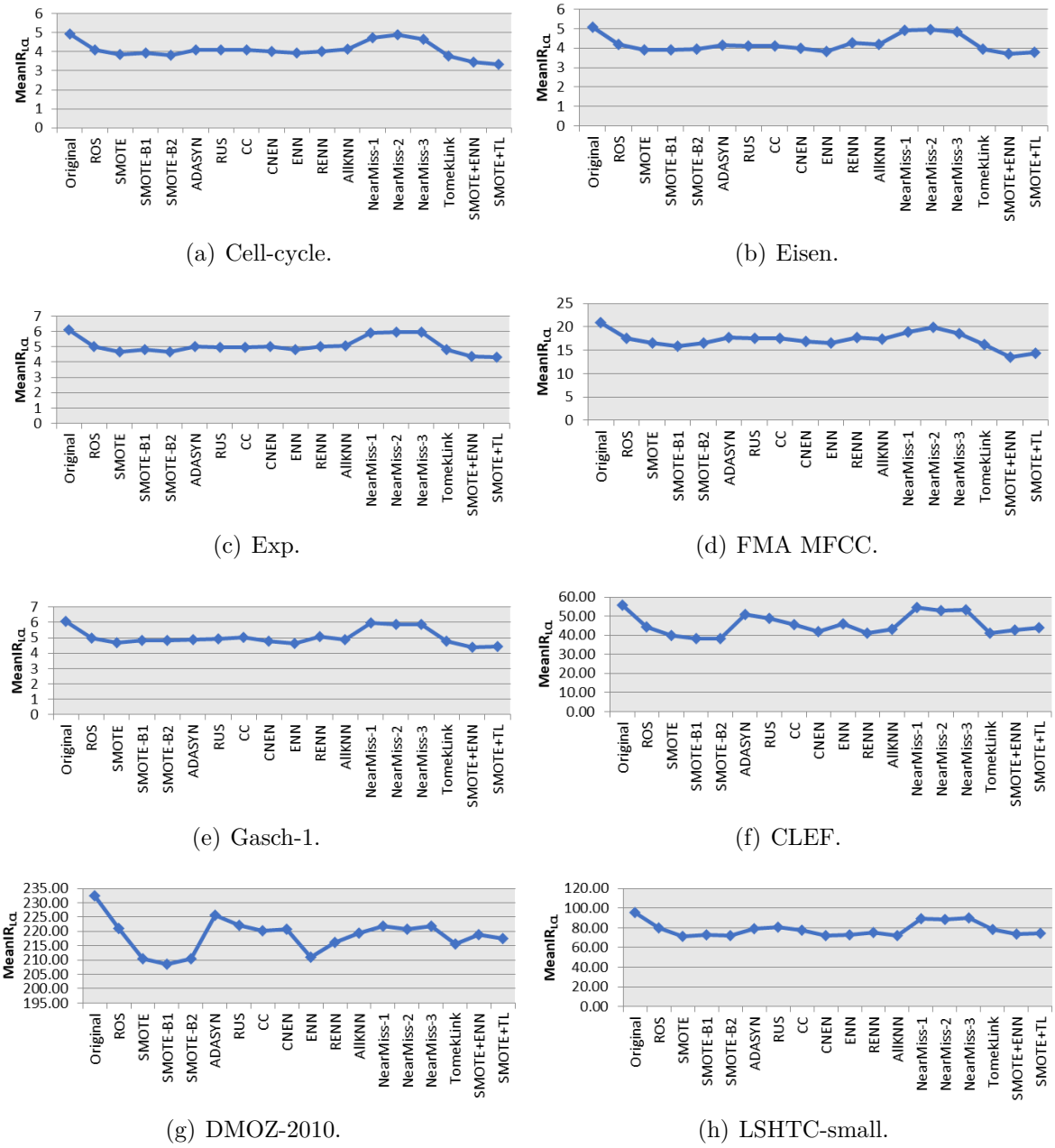
(e) Gasch-1.

(f) CLEF.

(g) DMOZ-2010.

(h) LSHTC-small.

Figure 54 – Mean Imbalance Ratio for Local Classifiers per Parent Node (IR_{LCPN}) for each dataset before and after resampling.



(a) Cell-cycle.

(b) Eisen.

(c) Exp.

(d) FMA MFCC.

(e) Gasch-1.

(f) CLEF.

(g) DMOZ-2010.

(h) LSHTC-small.

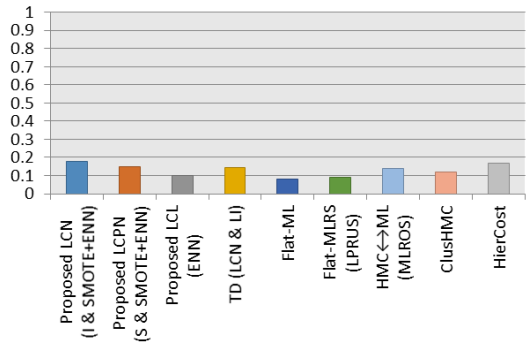
Figure 55 – Mean Imbalance Ratio for Local Classifiers per Level (IR_{LCL}) for each dataset before and after resampling.

7.4 Analysis and Discussion

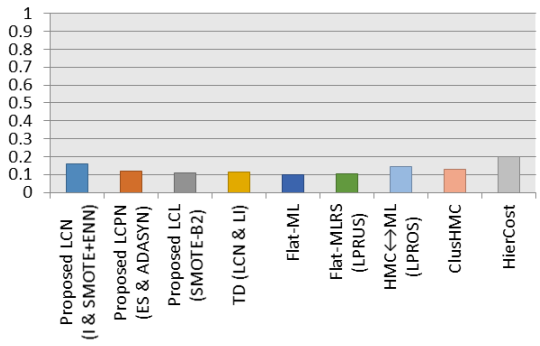
The analysis and discussion concerning the experimental results of the approaches and metrics proposed in this work is grounded in seven questions:

1. Which resampling techniques improved the classification results in each one of the local classifiers approaches?

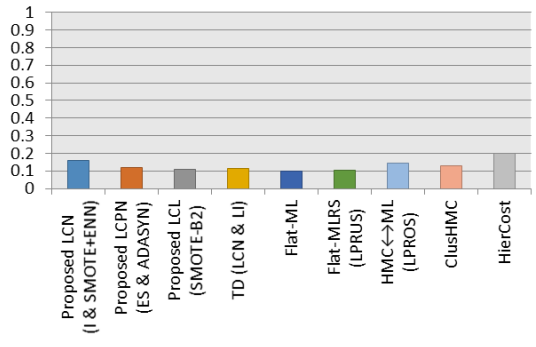
2. Which resampling technique is the best in each local classification scenario?



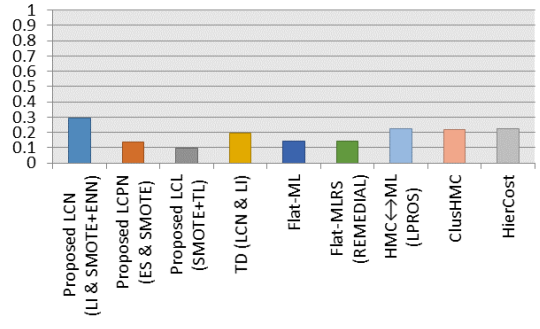
(a) Cell-cycle.



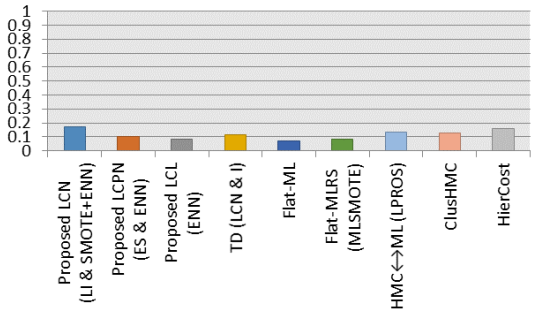
(b) Eisen.



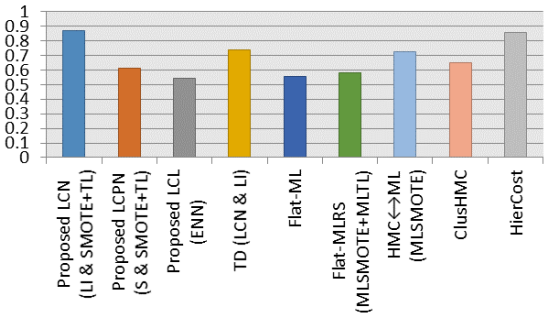
(c) Exp.



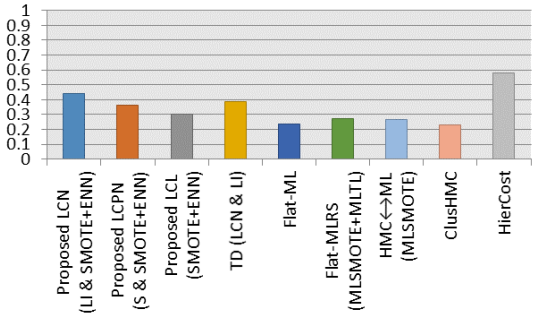
(d) FMA MFCC.



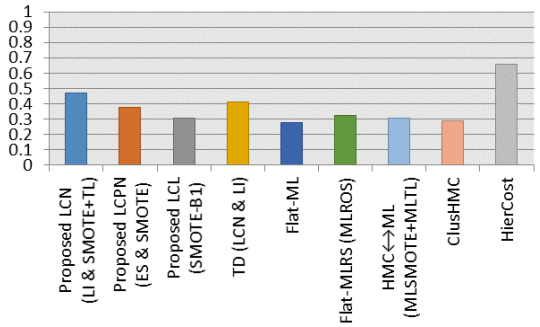
(e) Gasch-1.



(f) CLEF.



(g) DMOZ-2010.



(h) LSHTC-small.

Figure 56 – Charts with the best F-Score results for all techniques in each dataset.

3. Are the local classifiers with resampling approaches (proposed in this work) able to surpass the results obtained by the flat classification using multi-label resampling algorithms baselines (Flat-MLRS)?

4. Are the proposed imbalanceness metrics measure the datasets imbalanceness in each local classification scenario?

5. Are the proposed resampling approaches able to deal with imbalanceness in large scale hierarchical datasets?

6. Can the proposed approaches enhance the local classifiers in order to outperform a global classification approach?

In the following sections we presents the responses to each one of the questions raised before with statistical significance.

Which resampling techniques improved the classification results in each one of the local classifiers approaches?

To answer this question we have also applied the Wilcoxon test in the classification results for each one of the local classifiers approaches, i.e., we applied the test crossing the results of the classic TD approach (Table 96 from the Appendix A) against the results with each local classifier approach (considering each given policy). The results for this test are presented in Table 102 (Local Classifiers per Node), Table 103 (Local Classifiers per Parent Node) and Table 104 (Local Classifiers per Level), which are also in the Appendix A of this Thesis. We must highlight that, even though none of the approaches statistically improved the classification results, SMOTE, SMOTE-B1, SMOTE-B2 and the hybrid techniques (SMOTE+ENN and SMOTE+TL) achieved interesting results in all tests, regardless the local classifiers approach, since they all reached a negative *z-score* and a *p*-value close to the threshold. On the other hand, we may also observe that NM1, NM2 and NM3 are the worst techniques, since their *z-scores* are positive and p-value above the threshold in all tests. Moreover, the other resampling techniques seem to improve the classification results with some of the policies, but not all of them.

Which resampling technique is the best in each local classification scenario?

To answer the second question, i.e., define the best classifier, we used the statistical evaluation protocol proposed in (CHARTE et al., 2015b). Using this protocol, we calculate the ranking of the classification results using resampling algorithms based on the Friedman statistical test. In other words, the algorithms performances are ranked (from first to last) and an average rank is calculated for each dataset and approach. Then a general average is computed for each approach. While Tables 105, 106 and 107 presents the results of this

test for the resampling schemas in LCN, LCPN and LCL approaches, respectively, Table 108 shows the average rank for all local classifiers approaches. To avoid visualization issues, all of these Tables are presented in the Appendix A of this Thesis. In average, the SMOTE and hybrid techniques are the most effective, since SMOTE + ENN is the best ranked method (3.06), followed by SMOTE (4.67), SMOTE + TL (5.06) and then SMOTE-B1 (5.91). On the other hand, NM2 (15.84), NM3 (15.43), NM1 (15.15), RUS (11.66) and AllKNN (11.44) are the less effective techniques. It is also important to note that there is a slight difference among the ranking of the algorithms in each local approach. For example, the second best resampling technique in the LCN scenario is SMOTE + TL, however in LCPN and LCL it is SMOTE by itself. Moreover, we may note that RENN and TL are much more effective in the LCL scenario than in LCN and LCPN.

It may be asked why the combination of SMOTE with ENN or TL significantly outperforms either SMOTE, ENN or TL by their selves. The reason for that can be grounded in the fact that, after applying SMOTE, the new synthetic samples were introduced somewhere along the feature space that made the classes groups become overlapped, i.e., some samples from the majority classes were invading the minority classes spaces or vice versa. Then, when ENN or TL is applied, some of these noises were removed, leading to a better model built by the classifier. This phenomenon was already identified in the experiments proposed by (BATISTA; PRATI; MONARD, 2004).

Are the local classifiers with resampling approaches able to surpass the results obtained by the flat classification using multi-label resampling algorithms baselines?

To answer the third question we applied the Kruskal Wallis Statistical Test in order to compare Table 98 to Tables 88-95. We stated as null hypothesis that there is no difference between the results with the flat resampling approach (Flat-MLRS) and all the proposed local approaches with resampling (LCN, LCPN and LCL). The Kruskal Wallis test resulted in a value of 15336.5 and a p-value of 0, which means that we reject the null hypothesis and, thus, there are differences between the results (considering a significance of 0.05). Since the results are statistically different, we may apply the Post-Hoc Mannwhitney test to find which pairs of results tables are different. The results of the Post-Hoc Mannwhitney test are shown in Table 109 (Appendix A), which shows the comparison between the flat resampling results and the local strategies. Considering a threshold of 0.005, the test confirms that there are significant differences between the flat and all local classifiers approaches, except for LCL. Besides, the low values indicate that the flat resampling approach is outperformed by LCN and LCPN approaches.

Are the proposed imbalanceness metrics measure the datasets imbalanceness in each local classification scenario?

The answer to this question is grounded in the Pearson Correlation Statistical Test. This test is used to define whether or not there is a correlation between two sequences of variables. In this context, we applied the Pearson test comparing the metrics calculated using each approach with the classification results achieved in the same approach. Tables 110, 111 and 112 (Appendix A) presents the results for the statistical test considering all scenarios (metric *versus* classification results). While the first column represents a value between -1 and 1, which indicates the correlation between the two sequences (a negative value shows an inverse correlation and a positive otherwise), the second column (*p-value*) indicates the reliability of the correlation: the less is the *p-value*, more reliable the correlation is. It is important to state that, according to (MUKAKA, 2012), we may associate the following correlations to the value of ρ:

- $\rho >= 0.9 \rightarrow$ very strong correlation;
- $0.7 >= \rho < 0.9 \rightarrow$ strong correlation;
- $0.5 >= \rho < 0.7 \rightarrow$ medium correlation;
- $0.3 >= \rho < 0.5 \rightarrow$ week correlation; and
- $\rho < 0.3 \rightarrow$ despicable correlation.

Taking the above information into account, we may calculate that in the LCN scenario, from a total of 48 Pearson tests, 19 of them (40%) indicated a strong correlation between the classification results and the imbalance ratio (IR) measures, 22 a medium correlation (46%), 6 a week correlation (12%) and 1 a despicable correlation (2%). Moreover, considering the tests in the LCPN scenario, from a total of 16 Pearson tests, 4 indicated a strong correlation (25%), 9 a medium correlation (56%) and 3 a week correlation (19%). Finally, in the LCL context, from a total of 8 Pearson tests, 3 indicated a strong correlation (38%), 3 a medium correlation (38%), 1 a week correlation (12%) and 1 a despicable correlation (12%). We may note that all correlations are inverse, i.e., when the IR measure is high, in general, the classification results are low. Thus, due to the fact that most of the Pearson tests indicated a strong or medium correlation between the IR measures and the classification results and, in general, the tests are reliable (presented low *p-values*), we may affirm that the proposed metrics, i.e., $MeanIR_{LCN}$, $MeanIR_{LCPN}$, $MeanIR_{LCL}$, can indeed measure the datasets imbalanceness in the different local classification perspectives.

Are the proposed resampling approaches able to deal with imbalanceness in large scale hierarchical datasets?

To answer the fifth question, we have to look specifically at Tables 91, 93, 93, 95. These Tables shows the summary of the results for the Large Scale Hierarchical Datasets experimentally tested in this work. It can be observed that our approach beat all the other techniques in two of these four datasets: FMA MFCC and CLEF. These datasets have a common characteristic: both have a considerable number of samples (90,393 and 10,000, respectively). However, these same datasets do not have a large number of attributes and labels, such as DMOZ-2010 and LSHTC-small. This fact leads us to the conclusion that our approach may not be suitable to solve imbalanceness in hierarchical datasets with large number of attributes and labels.

Can the proposed approaches enhance the local classifiers in order to outperform a global classification approach?

To answer this question we applied again the Wilcoxon Statistical Test, comparing the best F-Score results obtained with each global classification approach (HMC↔ML, Clus-HMC and HierCost) with the best F-Score result obtained with the proposed approaches for all datasets. The results for this test are presented in Table 113, which is also in Appendix A. Considering a threshold of 0.1, we could observe that the best proposed approach significantly improved the results when compared to Clus-HMC. Furthermore, when the best proposed approach is compared to the HMC↔ML method, although it is not statically higher, it did got a negative z-score and a p-value really close to the threshold (0.1038), indicating that it could statically improve the results in other scenarios. Finally, when comparing the best proposed approach to the HierCost technique, we concluded that it did not statically improved the results, since the z-score is positive.

7.5 Final Considerations

This chapter presented novel resampling schemas to handle imbalanceness in hierarchical classification problems when using the local classification approaches. We proposed three different methods to deal with each one of the local classification approaches: Local Classifiers per Node (LCN); Local Classifiers per Parent Node (LCPN); and Local Classifiers per Level (LCL). The proposed techniques handle the imbalanceness by applying classic binary resampling methods locally in the training datasets before building each individual classification model. Besides, in order to experimentally test and compare the proposed approaches, we also presented results for baselines, related works and state-of-the-art approaches from the literature.

Moreover, we proposed novel metrics to calculate the average imbalanceness (MeanIR) in the three different local classification contexts: $MeanIR_{LCN}$, $MeanIR_{LCPN}$, $MeanIR_{LCL}$. These measures are mainly based in the average proportions between the number of instances labeled with the label nodes.

The experimental results and statistical analysis with eight well-known datasets from the biological, musical, text and image domains showed that the proposed resampling schemas may indeed increase the classification results for all datasets when compared to the original classification scenarios. Furthermore, using the Pearson statistical test we were able to identify an inverse relation between the proposed imbalance measures and the classification results, showing that the proposed metrics are able to measure the hierarchical datasets imbalanceness.

Concerning the large scale hierarchical classification datasets, we could observe that our proposed techniques were not able to deal with the imbalanceness and improve the results of the datasets with large number of attributes and labels, although it could improve the results of two large datasets that do not have a large number of attributes/labels.

After investigate the imbalanceness issue in hierarchical classification problems considering local perspectives, the next step is to analyze the problem in a global fashion-way. In the next Chapter we present the first contributions of this Doctoral Research towards the resampling of hierarchical classification datasets as a whole. The Hierarchical Random Oversampling (HROS) and Hierarchical Random Undersampling (HRUS) algorithms are able to pre-process a hierarchical dataset considering the labels hierarchy and the different possibilities of depth among the label paths.

Chapter 8 Global Approaches: The Hierarchical Random Resampling Algorithms

So far, it was presented in this thesis alternative approaches to deal with the imbalanceness in hierarchical classification problems with existing resampling techinques, i.e., binary/multi-class and multi-label resampling algorithms. In this chapter we present the first studies towards the development of a global resampling approach for hierarchical classification datasets which considers the labels hierarchy as a whole.

Dealing with the imbalanceness issue in hierarchical classification datasets is a challenging task, because there are different kinds of hierarchical classification problems, which have different types of properties, such as the type of labels taxonomy, the depth of the prediction and the number of paths associated to each sample (SILLA JR; FREITAS, 2011). Given this context, the contributions of this chapter are three-fold:

- A novel approach to find sets of majority/minority label paths in a hierarchical classification dataset;
- Novel resampling algorithms for hierarchical classification problems with partial and full depth label prediction; and
- A diversified set of hierarchical classification datasets allowing testbeds for the research community.

It is worth mentioning that the contributions of this chapter were submitted for publication and are under review in the Information Sciences journal.

8.1 The Proposed Random Resampling Algorithms

Although we are only concerned with hierarchical classification problems in which the labels are organized in a tree-based taxonomy, in order to design the novel resampling algorithms, we have considered other two different variants described in Silla Jr & Freitas (2011): number of paths (defined in the literature as Ψ) and depth of the paths (defined in the literature as Φ). We propose two groups of resampling algorithms considering the depth of the labels in the hierarchical classification problem. Each group is composed by an Oversampling and an Undersampling algorithm.

8.1.1 Finding the Majority and Minority Classes

When resampling a dataset, we first have to define the target samples of the resampling process. When we are applying an oversampling algorithm, the samples belonging to the minority classes have to be increased, and when applying an undersampling algorithm, the instances from to the majority classes are decreased.

In binary and multi-class datasets, the majority/minority classes can be identified by considering the most/less frequent labels among the samples. Moreover, for the identification of majority and minority labels in multi-label datasets, Charte et al. (2013) defined and suggested the use of the IRLbl and MeanIR imbalance measures. Their idea was to consider the labels with an imbalance ratio (IRLbl) below the average imbalance ratio (MeanIR) as belonging to the set of majority classes (named as majority bag), otherwise the classes belong to the minority bag.

Thus, before proposing a new resampling algorithm for hierarchical datasets, we first have to establish a mechanism to identify the majority and minority classes considering the class hierarchy. In hierarchical problems the classes are represented by label paths in the tree taxonomy instead of individual labels, we used the imbalance ratio per label path (IRLP) and hierarchical mean imbalance ratio (HMeanIR), earlier presented in Section 6.1 and published in Pereira, Costa & Silla Jr (2018). Our idea here is to find the majority and minority paths in the dataset based on their imbalance ratio, which were calculated with Formulas 6.1 and 6.2.

Algorithm 35 shows the pseudocode for the imbalance ratios calculations. Furthermore, the pseudocode of the proposed methods to retrieve the set of majority and minority label paths in hierarchical datasets are presented in Algorithms 36 and 37, respectively.

Algorithm 35 Pseudocode for Imbalance Ratios Calculations

Inputs: D: The hierarchical dataset
Output: IRLP: The imbalance ratio per label paths in D
HMeanIR: The average imbalance ratio in D

```
labelPaths ← label paths from dataset D
for each path in labelPaths do
    countDict[path] ← number of samples in D laballed with path
end for
maxCount ← max number of labelPaths in countDict
IRLP ← empty dictionary
for each path in labelPaths do
    pathCount ← countDict[path]
    IRLP[path] ← maxCount/pathCount
end for
hMeanIR ← (∑_{i=1}^{|labelPaths|} IRLP[path_i])/|labelPaths|
return IRLP, hMeanIR
```

Algorithm 35 receives as input the hierarchical dataset (D) and returns two outputs: The imbalance ratio per label path (IRLP) and the average imbalance ratio (HMeanIR). The looping from lines 1-3 counts the number of samples belonging to each label path from dataset D. It is important to state that we are considering all samples that are labeled with the given *path*, for example, if the *path* is "A/B", samples labeled with "A/B/C" and "A/B/D" are taken into account. In lines 5-10 the imbalance ratio per label path is calculated and then in line 11 the average imbalance ratio is obtained.

Algorithms 36 and 37 receive the dataset D as input and outputs the set of majority (MajPaths) or minority (MinPaths) paths. First, in line 1 both algorithms use Algorithm 35 to get the imbalance ratios and then, in lines 2-7, the algorithms make a looping over the label paths, filtering the ones whose IRLbl are below hMeanIR (majority paths) or above hMeanIR (minority paths).

Algorithm 36 Pseudocode for Retrieving Majority Paths

Inputs: D: The hierarchical dataset
Output: $MajPaths$: The label paths from the set of majority paths

```
1: IRLP, hMeanIR ← Calculate Imbalance Ratios
2: majPaths ← empty list
3: for each path in labelPaths do
4:     if IRLblP[path] < hMeanIR then
5:         append path into majPaths
6:     end if
7: end for
8: return majPaths
```

Algorithm 37 Pseudocode for Retrieving Minority Paths

Inputs: D: The hierarchical dataset
Output: $MinPaths$: The label paths from the set of minority paths

```
1: IRLP, hMeanIR ← Calculate Imbalance Ratios
2: minPaths ← empty list
3: for each path in labelPaths do
4:     if IRLblP[path] > hMeanIR then
5:         append path into minPaths
6:     end if
7: end for
8: return minPaths
```

Given the target samples, which can be obtained with Algorithms 35, 36 and 37, we are able to oversample or undersample the hierarchical dataset. In the following subsections, we present details of the novel resampling algorithms. In order to contemplate the different kinds of hierarchical problems, we proposed two different oversampling/undersampling algorithms, considering the depth of label paths in the dataset (based on the Φ definition - Full or Partial Depth), one oversampling/undersampling for each type.

8.1.2 Resampling Full Depth Hierarchical Classification Problems

In this kind of problem, the instances may be associated with one or more label paths, but always with full depth in the label tree. Considering this, we propose the Random Oversampling/Undersampling for Full Depth Hierarchical Classification Problems (HROS-FD/HRUS-FD). The pseudocodes of these methods are presented in Algorithms 38 and 39, respectively.

Both algorithms are very similar and receive the dataset to oversample/undersample (represented as D) and the percentage of samples to increase/decrease (S) and outputs the resampled Dataset (D'). The main idea of the algorithms is to obtain the set of majority/minority label paths and randomly remove/create samples from/for these paths until the number of removed/created samples reach the percentage determined by the S parameter. Since the procedure followed by HROS-FD and HRUS-FD is analogous, in the following we will only describe HROS-FD in details.

Algorithm 38 Pseudocode for HROS-FD

Inputs: D: The hierarchical dataset, S: Percentage of samples to increase
Output: D': An oversampled dataset

```
1: samplesToCreate ← |D| × S
2: minPaths ← getMinorityPaths(D)
3: maxIncrease ← samplesToCreate / |minPaths|
4: meanSize ← calculate the average number of samples per labelPaths
5: for labelPath in minPaths do
6:     numSamples ← samplesWithLabelPath(D, labelPath)
7:     increased ← 0
8:     while increased < maxIncrease and numSamples < meanSize do
9:         D' ← randomly duplicate sample labeled with labelPath
10:        numSamples ← numSamples + 1
11:        increased ← increased + 1
12:    end while
13: end for
14: return D'
```

First, we calculate the exact number of samples to be created (line 1) and then retrieve the set of minority paths using Algorithm 37 (line 2). In order to force a distribution of the number of samples to be increased among the minority label paths, in line 3 a maximum increase per label path is calculated. In line 4 the average number of samples per label path is obtained to establish another limit in the sample's duplication distribution. Into the algorithm's main loop (lines 5-12) the samples from each minority path are randomly duplicated until reach the maximum increase, which is determinate between the division between the total number of samples to increase and the number of minority paths, or until reach the mean size of the label paths in the dataset.

Algorithm 39 Pseudocode for HRUS-FD

Inputs: D: The hierarchical dataset, S: Percentage of samples to decrease
Output: D': An undersampled dataset

```
samplesToRemove ← |D| × S
maxPaths ← getMajorityPaths(D)
maxDecrease ← samplesToRemove / |maxPaths|
meanSize ← calculate the average number of samples per labelPaths
for labelPath in minPaths do
    numSamples ← samplesWithLabelPath(D, labelPath)
    decrease ← 0
    while decrease < maxDecrease and numSamples > meanSize do
        D' ← randomly remove sample labeled with labelPath
        numSamples ← numSamples - 1
        decrease ← decrease + 1
    end while
end for
return D'
```

8.1.3 Resampling Partial Depth Hierarchical Classification Problems

In this type of hierarchical classification problems, the instances may be associated with one or more label paths with full or partial depth in the label tree. The challenge of resampling this kind of data is that when creating or removing samples from children nodes, the number of samples from parent label nodes will be indirectly increased or removed. This problem does not affect full depth hierarchical classification problems because there are no samples labeled exclusively with internal nodes. Figure 57 presents an example of this issue for a given training set with 6 samples ($S_1 ... S_6$). We simulated the duplication of 3 samples labeled with node H to shown how it impacts on the internal nodes. We may observe that when we created these samples for node H, we indirectly created samples for nodes E and D.

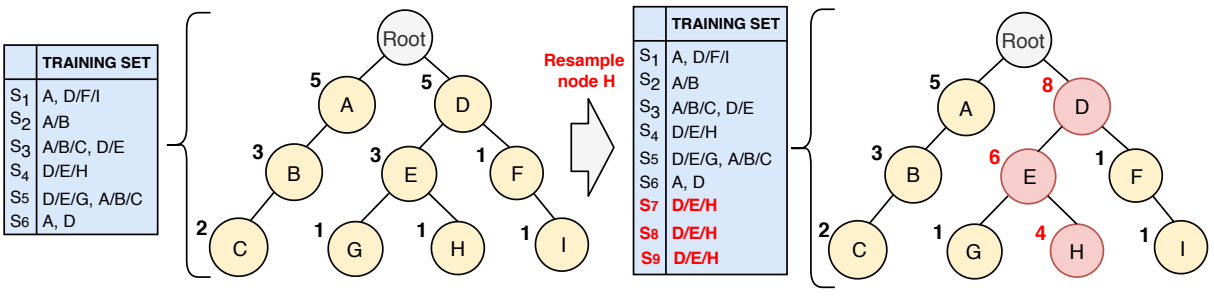


Figure 57 – An example of the main issue when creating samples in leaf nodes. The numbers on the top left of the nodes symbolize the number of samples belonging to each node.

In order to deal with this issue, we propose a technique to process the instances

in a "bottom-up order", recalculating the Majority/Minority Paths in each loop of the resampling process. Figure 58 shows a visual example of the proposed method. For this specific example the resample process takes a total of 3 steps. The example dataset is composed of 85 samples and a label tree with 9 nodes. We simulated the application of an oversampling method with an increase rate of 15%, i.e., 12 samples.

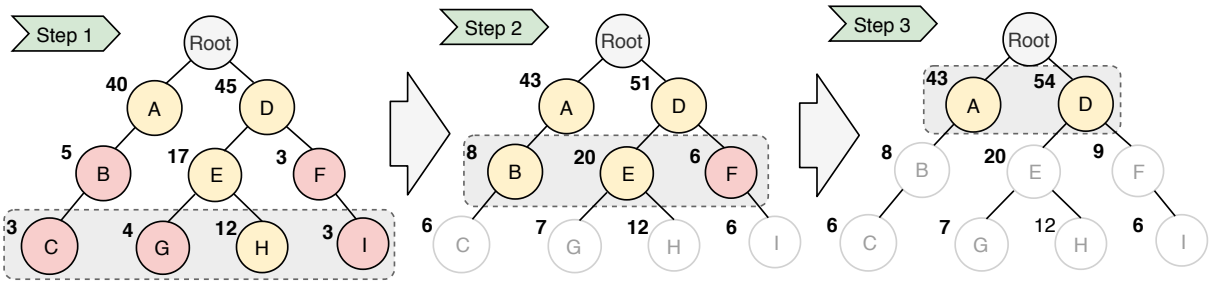


Figure 58 – An example of application of the HROS-PD in a dataset with 85 instances. The nodes marked with a circled dashed are being processed at the certain step and the red nodes represent the label paths belonging to the minority set.

Table 36 shows the variation of the imbalance ratio (IRLP) during the resampling steps. The numbers in bold are above the HMeanIR, which is 8.45. In the first step, the proposed method process nodes C, G and I, since they belong to the set of minority paths, randomly duplicating 3 samples from each of these label paths. The number of samples to be duplicate is calculated by dividing the 12 - 15% of increase rate - to the total of minority label paths in the dataset, which is 5 for this example. In step 2 only node F is resampled. It is important to observe that in step 1 node B belonged to the set of minority paths (with an IRLP of 11.25), however, since node C was resampled, node B was also indirectly resampled (IRLP changed to 7.29) and thus does not belong to the minority set anymore (HMeanIR is 8.45). In the third and last step, no nodes are resampled.

Table 36 – The imbalance ratios after each step from Figure 6. Numbers in bold are above the mean imbalance ratio - *HMeanIR*.

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
Step 1	1.13	**11.25**	**15.00**	1.00	2.65	**15.00**	**11.25**	3.75	**15.00**
Step 2	1.19	7.29	**8.50**	1.00	2.55	**8.50**	7.29	4.25	**8.50**
Step 3	1.26	7.71	**9.00**	1.00	2.70	6.00	7.71	4.50	**9.00**

Considering the previous example, we propose the Random Over/Under-sampling for Partial Depth Hierarchical Classification Problems (HROS-PD/HRUS-PD). The pseudocodes of these methods are presented in Algorithms 40 and 41, respectively. Since the procedure followed by HROS-PD and HRUS-PD is similar, we will present a detailed explanation concerning HROS-PD and then will highlight the main differences in relation to HRUS-PD.

Algorithm 40 Pseudocode for HROS-PD

Inputs: D: The hierarchical dataset, S: Percentage of samples to increase
Output: D': An oversampled dataset

```
1: samplesToCreate ← |D| × S
2: labelTree ← retrieve label tree from D
3: minPaths, hMeanIR ← getMinorityPaths(D)
4: meanSize ← calculate the average number of samples per labelPaths
5: maxIncrease ← samplesToCreate / |minPaths|
6: while labelTree is not empty do
7:     for each leafNode in labelTree do
8:         if leafNode in minPaths then
9:             numSamples ← samplesWithLabelPath(D, leafNode)
10:            increased ← 0
11:            while increased < maxIncrease and numSamples < meanSize do
12:                D' ← randomly duplicate sample (lastly labeled with leafNode)
13:                numSamples ← numSamples + 1
14:                increased ← increased + 1
15:            end while
16:        end if
17:        remove leafNode from labelTree
18:    end for
19:    minPaths ← getMinorityPaths(D, hMeanIR)
20: end while
21: return D'
```

The first five lines of the algorithm represent the preparation phase and are similar to HROS-FD, with only a few differences. The first difference is on line 2, in which the dataset label tree is obtained. The second difference is when calculating the set of minority paths, in which the mean imbalancenss (hMeanIR) also has to be obtained by the algorithm. The main process of the algorithm is made thought the *while* looping from lines 6 to 20. The idea is to resample each leaf node that belongs to the set of minority paths, re-calculating the set of minority paths in each looping step. The duplicating process is also similar to HROS-FD. An important difference is that in line 12, in which a sample is randomly duplicated, the chosen sample is preferably lastly labeled with the target leaf node, i.e., if the algorithm is resampling an internal label path $x/y/z$, the sample chosen to be duplicated has also to be labeled with $x/y/z$, save exceptions in which there are no samples under this circumstance. This procedure avoids that the duplication of samples from internal nodes affects children nodes already processed by the algorithm. Another important issue in the proposed algorithm is that when re-calculating the set of minority paths (line 19) the hMeanIR first obtained from the dataset (on line 3) has to be used as the threshold for the selection of the minority label paths.

Algorithm 41 Pseudocode for HRUS-PD

Inputs: D: The hierarchical dataset, S: Percentage of samples to decrease
Output: D': An undersampled dataset

```
samplesToRemove ← |D| × S
labelTree ← retrieve label tree from D
maxPaths, hMeanIR ← getMajorityPaths(D)
meanSize ← calculate the average number of samples per labelPaths
maxDecrease ← samplesToRemove / |maxPaths|
while labelTree is not empty do
    for each leafNode in labelTree do
        if leafNode in maxPaths then
            numSamples ← samplesWithLabelPath(D, leafNode)
            decreased ← 0
            while decreased < maxDecrease and numSamples > meanSize do
                randomly remove sample (preferably lastly labeled with leafNode)
                decreased ← decreased + 1
                numSamples ← numSamples - 1
            end while
        end if
    end for
    remove leafNode from labelTree
    maxPaths ← getMajorityPaths(D, hMeanIR)
end while
return D'
```

8.2 Experimental Protocol and Results

In this section we present the datasets, algorithms and parameters, experimental setup and classification results regarding the proposed resampling algorithms.

8.2.1 The Datasets

In order to cover the different aspects of hierarchical classification problems, we performed computational experiments in a total of 23 datasets: 9 with full depth problems and 14 with partial depth problems. As some of these datasets were adapted and hence are somehow novel, presenting characteristics concerning the different taxonomies of hierarchical classification problems as defined by Silla Jr & Freitas (2011), they form a testbed for hierarchical classification researchers and, thus, may be also considered a contribution of this Chapter. It is important to state that all datasets, algorithms implementations and detailed results can be obtained in this link[12].

Tables 37 and 38 present a detailed description of the datasets with Full Depth problems and Partial Depth problems, respectively. The datasets marked with (*) and

[12] https://sites.google.com/view/hierarchical-imblearn/

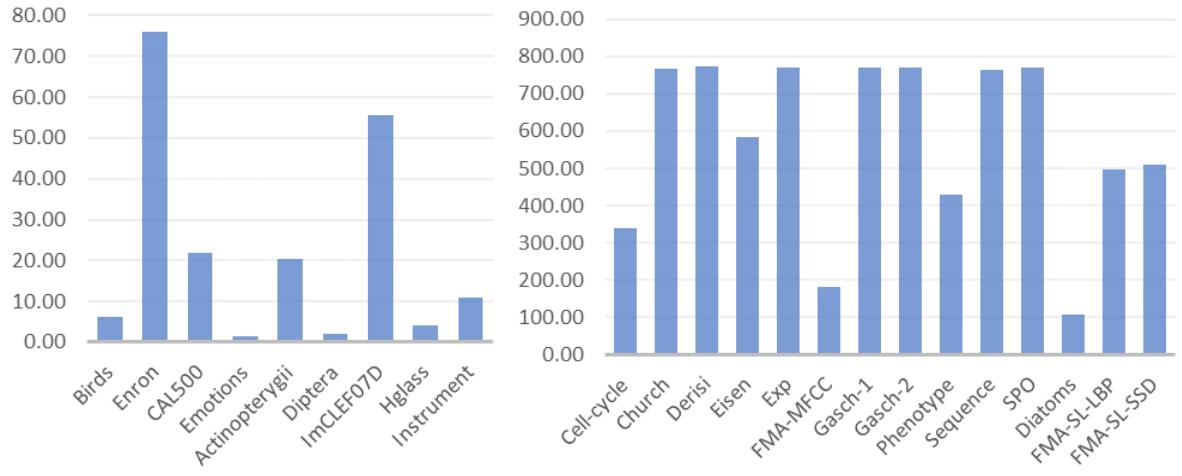
(**) are somehow novel. The datasets marked with (*) were originally proposed as flat multi-label classification datasets in the literature and in this work were adapted to a hierarchical taxonomy. The datasets marked with (**) were extracted as single-label subsets from the original multi-label dataset and were also adapted to a hierarchical taxonomy. The hierarchical adaptations were manually made considering the intrinsic characteristics of the each problem domain. For instance, in the Enron dataset, which is composed of e-mail subjects classification examples, the subjects were hierarchically organized according to the subjects.

Table 37 – Datasets with Full Depth Hierarchical Classification Problems.

Name	Paths	Domain	Train	Test	Attr.	Labels	Labels p/ level	Reference
Enron*	Multiple	Text	988	660	1001	57	3, 40, 14	(KLIMT; YANG, 2004b)
CAL500*		Music	351	151	68	164	7, 76, 78, 3	(TURNBULL et al., 2007)
Emotions*			392	203	72	9	3, 6	(TROHIDIS et al., 2008)
Birds*			272	79	260	49	13, 17, 19	(BRIGGS et al., 2013)
Actinopterygii	Single	Biology	15705	6739	15	30	2, 6, 12, 15	(PARMEZAN; SOUZA; BATISTA, 2018)
Diptera			15194	6528	33	29	4, 6, 9, 10	
Instrument		Audio	6583	2836	30	46	5, 10, 31	
Hglass		Glass	144	70	9	11	2, 3, 5, 1	(METZ et al., 2011)
ImCLEF07D		Image	10000	1006	80	24	4, 9, 11	(DIMITROVSKI et al., 2011a)

Table 38 – Datasets with Partial Depth Hierarchical Classification Problems.

Name	Paths	Domain	Train	Test	Attr.	Labels	Labels p/ level	Reference
Cell-cycle	Multiple	Biology	2484	1281	78	180	4, 22, 70, 84	(CLARE; KING, 2003)
Church			2474	1281	24	180	4, 22, 70, 84	
Derisi			2450	1275	62	180	4, 22, 70, 84	
Eisen			1587	837	80	170	4, 22, 66, 78	
Exp			2488	1291	544	180	4, 22, 70, 84	
Gasch-1			2480	1284	174	180	4, 22, 70, 84	
Gasch-2			2488	1291	53	180	4, 22, 70, 84	
Phenotype			1009	582	64	168	4, 22, 66, 76	
Sequence			2580	1339	437	180	4, 22, 70, 84	
SPO			2437	1266	79	180	4, 22, 70, 84	
FMA-MFCC*	Single	Music	33259	14274	13	97	12, 66, 19	(DEFFERRARD et al., 2017)
FMA-SLLBP**			15331	7000	59	135	25, 91, 19, 1	
FMA-SLSSD**			15631	6700	161	135	25, 91, 19, 1	
Diatoms		Image	2065	1054	371	398	82, 313, 3	(DIMITROVSKI et al., 2012)



(a) Full Depth. (b) Partial Depth.

Figure 59 – Mean Imbalance Ratios for the Datasets.

Looking at the datasets characteristics we may observe that FMA-MFCC is by far the largest one, while Hglass is the smallest one. Furthermore, Hglass is the dataset with the lowest number of attributes, while Enron dataset has the most.

Figure 59 presents graphics of the Hierarchical Mean Imbalance Ratio (HMeanIR) for the datasets used in the experimental analysis. We may observe that the datasets present a large variety of imbalanceness. The full depth classification problems datasets are much less imbalanced than the datasets with partial depth, with Emotions dataset reaching a HMeanIR of only 1.49. On the other hand, Church, Derisi, Exp, Gasch-1, Gasch-2, Sequence and SPO are the most unbalanced datasets, with HMeanIR close to 800.00.

It is important to observe that all datasets presented in Tables 37 and 38 have labels in a Tree taxonomy, since we are not dealing with Directed Acyclic Graphs taxonomies in this work.

8.2.2 Classification Algorithm and Parameters

For the hierarchical classification task we used the Clus-HMC framework. Clus-HMC was chosen because it is considered in the literature as the state-of-the-work hierarchical classification framework (CERRI; BARROS; CARVALHO, 2015; WEHRMANN; CERRI; BARROS, 2018; PEREIRA; GABRIEL; CERRI, 2019).

Clus-HMC is based on Predictive Cluster Trees (PCT) and generates a single Decision Tree (DT) considering an entire class hierarchy. In Clus-HMC, DTs are seen as a hierarchy of clusters where the root node contains all the training instances, while the remaining are recursively divided into smaller groups as the hierarchy is traversed towards the leaves. The classification is performed using a distance-based metric which calculates how similar an instance is to some tree.

The parameter configurations used in the algorithm are reported in Table 39.

8.2.3 Experimental Setup

In this work we used the *weighted* Area Under the Precision-Recall Curve (AUPRC) metric (details presented in subsection 2.2.3.3) to measure the classification results and we have tested seven increase/decrease rates commonly used by researchers in the resampling tasks: 5%, 10%, 15%, 20%, 25%, 30% and 35%.

The results presented in the tables of subsection 8.2.4 are the average of 10 executions of Clus-HMC after re-applying the proposed resampling algorithms in the training sets.

Table 39 – Clus-HMC execution parameters.

Parameter	Value
Type	Tree
ConvertToRules	No
HSeparator	"/"
FTest	[0.001, 0.005, 0.01, 0.05, 0.1, 0.125]
EnsembleMethod	RForest
Iterations	10
VotingType	Majority
EnsembleRandomDepth	No
SplitSampling	None
Heuristic	Default
PruningMethod	Default

8.2.4 Results

Table 40 presents the classification results before and after applying HROS-FD and HRUS-FD algorithms in the training sets of the full depth hierarchical datasets with single paths prediction. In this table we may highlight Hglass dataset, which was the most benefited from the resampling algorithms and being able to reach an AUPRC of 0.9394 after applying HROS-FD with 10% of samples increase (0.1249 more than the original dataset). On the other hand, Diptera dataset showed to be the least affected by the resampling algorithms, with the best result achieving only an increasement of 0.0079.

Table 40 – Results for the full depth hierarchical datasets with single paths.

		Actinopterygii	Diptera	ImCLEF07D	Hglass	Instrument
	Original	0.7570	0.6027	0.7151	0.8145	0.7656
HROS-FD	5%	0.7746	0.6003	0.7041	0.8394	0.7698
	10%	**0.7869**	0.6042	0.7292	**0.9394**	**0.7873**
	15%	0.7658	0.6094	0.7042	0.8880	0.7692
	20%	0.7530	0.5977	0.7069	0.8759	0.7584
	25%	0.7505	0.5969	0.6927	0.8485	0.7625
	30%	0.7502	0.5863	0.6835	0.8706	0.7602
	35%	0.7571	0.5953	0.6941	0.8646	0.7499
HRUS-FD	5%	0.7680	0.6086	0.7185	0.8222	0.7726
	10%	0.7541	0.6026	**0.7210**	0.7944	0.7633
	15%	0.7512	0.6033	0.7182	0.8910	0.7569
	20%	0.7516	0.6089	0.6937	0.8161	0.7222
	25%	0.7520	**0.6106**	0.6882	0.8214	0.7280
	30%	0.7397	0.6082	0.6880	0.7783	0.7152
	35%	0.7312	0.6042	0.6623	0.7868	0.7090

Table 41 shows the classification results for the datasets with full depth prediction and multiple paths. We may observe that, in general, the resampling algorithms were not very effective, with the best result being reached after applying HROS-FD with 10% of increase in the Emotions dataset, in which the results were 0.0436 greater than using the original dataset.

Table 41 – Results for the full depth hierarchical datasets with multiple paths.

		Birds	Enron	CAL500	Emotions
	Original	0.4536	0.5693	0.4942	0.6843
HROS-FD	5%	0.4327	0.5720	0.4874	0.7114
	10%	0.4498	0.5990	0.4938	**0.7279**
	15%	0.4333	**0.5789**	0.5043	0.6951
	20%	0.4454	0.5656	**0.5174**	0.6980
	25%	0.4459	0.5516	0.4925	0.6888
	30%	0.4475	0.5578	0.4942	0.6931
	35%	0.4444	0.5628	0.4874	0.7101
HRUS-FD	5%	0.4308	0.5672	0.4874	0.6943
	10%	0.4224	0.5690	0.4945	0.7012
	15%	**0.4645**	0.5630	0.4941	0.6922
	20%	0.4301	0.5635	0.4945	0.6954
	25%	0.4365	0.5676	0.4874	0.6821
	30%	0.4106	0.5585	0.4945	0.6856
	35%	0.3952	0.5623	0.4947	0.6890

Table 42 shows the classification results before and after applying HROS-PD and HRUS-PD in the partial depth prediction hierarchical datasets with single paths. In these results, a highlight is that the best results with HROS-PD overperformed all HRUS-PD results. Another consideration that can be made is how bad performed was HRUS-PD on Diatoms dataset, decreasing the classification results for all resampling rates.

Table 42 – Results for the partial depth hierarchical datasets with single paths.

		Diatoms	FMA-SL-LBP	FMA-SL-SSD
	Original	0.2582	0.3268	0.2700
HROS-PD	5%	0.2445	0.3173	0.2717
	10%	0.2646	0.3204	0.2776
	15%	0.2659	**0.3484**	**0.2918**
	20%	**0.2887**	0.3307	0.2754
	25%	0.2728	0.3277	0.2760
	30%	0.2544	0.3107	0.2668
	35%	0.2623	0.3159	0.2633
HRUS-PD	5%	0.2243	0.3271	0.2721
	10%	0.2401	0.3293	0.2726
	15%	0.2431	0.3200	0.2663
	20%	0.2393	0.3371	0.2716
	25%	0.2206	0.3376	0.2697
	30%	0.2333	0.3238	0.2659
	35%	0.2223	0.3248	0.2717

Tables 43 and 44 presents the classification results for the datasets with partial depth and multiple paths prediction. We may note in these tables that while Cell-cycle was the dataset most benefited by the resampling methods, reaching an increasement of 0.0429 using HROS-PD with 10% of resampling rate, SPO dataset was the least affected

by the resampling algorithms, achieving a maximum increasement of only 0.0034.

Table 43 – Results for the partial depth hierarchical datasets with multiple paths (Part 1).

		Cell-cycle	Church	Derisi	Eisen	Exp	FMA-MFCC
	Original	0.1307	0.1222	0.1309	0.1483	0.1606	0.2803
HROS-PD	5%	0.1654	0.1347	0.1404	0.1590	0.1672	0.2890
	10%	**0.1736**	**0.1352**	**0.1586**	0.1719	0.1585	0.2628
	15%	0.1632	0.1304	0.1425	0.1603	**0.1753**	0.2592
	20%	0.1594	0.1264	0.1450	0.1513	0.1660	0.2588
	25%	0.1415	0.1279	0.1366	0.1557	0.1660	0.2572
	30%	0.1424	0.1230	0.1261	0.1642	0.1526	0.2533
	35%	0.1465	0.1204	0.1264	0.1625	0.1600	0.2560
HRUS-PD	5%	0.1458	0.1217	0.1330	0.1589	0.1634	0.2908
	10%	0.1425	0.1287	0.1307	0.1578	0.1636	0.2783
	15%	0.1471	0.1221	0.1312	**0.1797**	0.1553	**0.3022**
	20%	0.1481	0.1264	0.1310	0.1486	0.1642	0.2820
	25%	0.1422	0.1217	0.1311	0.1465	0.1585	0.2910
	30%	0.1446	0.1304	0.1307	0.1473	0.1597	0.2812
	35%	0.1411	0.1254	0.1229	0.1501	0.1486	0.2817

Table 44 – Results for the partial depth hierarchical datasets with multiple paths (Part 2).

		Gasch-1	Gasch-2	Phenotype	Sequence	SPO
	Original	0.1544	0.1410	0.1256	0.1683	0.1342
HROS-PD	5%	0.1590	0.1492	0.1318	0.1655	0.1357
	10%	**0.1868**	**0.1654**	0.1303	0.1679	0.1359
	15%	0.1733	0.1549	0.1272	0.1598	0.1277
	20%	0.1636	0.1469	0.1237	0.1637	0.1265
	25%	0.1581	0.1466	0.1256	0.1761	**0.1376**
	30%	0.1596	0.1415	0.1294	**0.1886**	0.1264
	35%	0.1611	0.1480	0.1297	0.1597	0.1317
HRUS-PD	5%	0.1560	0.1411	0.1240	0.1665	0.1344
	10%	0.1585	0.1540	0.1260	0.1674	0.1347
	15%	0.1510	0.1399	0.1276	0.1675	0.1348
	20%	0.1569	0.1388	0.1344	0.1626	0.1337
	25%	0.1496	0.1301	0.1326	0.1620	0.1343
	30%	0.1521	0.1410	**0.1370**	0.1626	0.1346
	35%	0.1451	0.1368	0.1255	0.1587	0.1352

8.3 Analysis and Discussion

Observing the previously described results, four main questions are raised: (i) May the proposed resampling algorithms increase the classification results? (ii) Which one is the most/least effective resampling algorithm? (iii) Does the increase/decrease rate influence in the classification results? (iv) Does the dataset HMeanIR influence the resampling and classification results?

In order to answer the first question with statistical significance, we applied the Wilcoxon Statistical Test, stating as hypothesis that the *weighted*-AUPRC of the classification is different after using the resampling methods. The test was applied for each one of the seven different increase/decrease rates and the *p-values* outputs for the Full Depth (HROS-FD and HRUS-FD) and Partial Depth algorithms (HROS-PD and HRUS-PD) are shown in Tables 45 and 46, respectively. Considering the threshold as 0.05, we may observe that in a certain way all resampling algorithms improved the results in particular scenarios. While HROS-FD statistically improved the classification results of the single path datasets when using a 10% of increase rate, HRUS-FD improved the results of the single path datasets with a decrease rate of 5%. Moreover HROS-PD significantly improved the classification results for the datasets with multiple paths when using 5% and 10% of increase rate, and HRUS-PD improved the results also for the multiple paths datasets with a decrease rate of 10%. Interestingly, it is possible to note that while the FD resampling algorithms performed better with single paths datasets, the PD methods were better in the multiple path classification problems.

Table 45 – *P*-values of the Wilcoxon signed-rank statistical test for the Full Depth Random Resampling Algorithms.

	HROS-FD		HRUS-FD	
(%)	Single Path	Multiple Path	Single Path	Multiple Path
5	0.3452	1.0000	**0.0431**	0.4652
10	**0.0431**	0.4652	0.3452	0.7150
15	0.3452	0.7150	0.8927	0.4652
20	0.5002	0.4652	0.3452	0.7150
25	0.5002	0.2733	0.5002	*0.0679*
30	0.5002	0.5930	*0.0796*	0.4652
35	0.6858	0.7150	*0.0796*	0.4652

Numbers in bold are below the threshold and in italic are close to the threshold.

Table 46 – *P*-values of the Wilcoxon signed-rank statistical test for the Partial Depth Random Resampling Algorithms.

	HROS-PD		HRUS-PD	
(%)	Single Path	Multiple Path	Single Path	Multiple Path
5	0.2850	**0.0058**	1.0000	*0.0828*
10	0.2850	**0.0409**	1.0000	**0.0409**
15	0.1088	0.1307	0.1088	0.4769
20	0.1088	0.3281	1.0000	0.1549
25	0.1088	*0.0743*	0.5930	0.8589
30	0.1088	0.6566	0.1088	0.5751
35	0.2850	0.7897	0.2850	0.4236

Numbers in bold are below the threshold and in italic are close to the threshold.

Moreover, although the Wilcoxon tests were not able to identify a significant improvement at threshold 0.05, we cannot affirm that HRUS-FD (using a decrease of 25%,

30% and 35%), HROS-PD (using an increase rate of 25%) and HRUS-PD (using a decrease rate of 5%), certainly did not improved the results, since their *p-values* are so close to the threshold. In fact, if we observe the classification results in Tables 40, 41, 42, 43 and 44, we may note some improvements in terms of AUPRC for these cases.

To answer the second question, we may also analyze Tables 45 and 46. We may note that HROS-PD was the only one that statistically overperformed the original classification results when using two different rates: 5% and 10%. Furthermore, HROS-PD was also close to the threshold when using 25% of increase rate. And, finally, although in the single path datasets HROS-PD did not statistically overperformed the original results using any increase rate, we may observe that in four of the seven p-values the value was 0.1088, i.e., only 0.05 above the threshold. In contrast, we may note that HRUS-PD was the only one that got exact 1.0 of p-value for three decreasing rates (5%, 10% and 15%), which shows a sign to be the less effective resampling method. Moreover, HROS-FD also poorly performed for all scenarios, with exception of single paths with 10% of increasing rate.

The third question may be answered by looking at the different ranges of *p*-values presented in Tables 45 and 46. It is notable that the post-resampled results that overperformed the original classification results were obtained with increase/decrease rates of 5% and 10%, despite of resampling algorithm or type of dataset. Thus, we can indeed affirm that the increase/decrease rate does influence the classification results.

The answer to the fourth and last question is first grounded at Figure 59. We may note that Enron, IMCLEF07D, Church, Derisi, Exp, Gasch-1, Gasch-2, Sequence and SPO are the most imbalanced datasets regarding HMeanIR, in contrast with Birds, Emotions, Diptera, Hglass and Diatoms, which are the least imbalanced ones. However, if we look at Tables 40, 41, 42, 43 and 44, it can be observed that the most imbalanced datasets were not necessarily the most benefited by the resampling algorithms. As a counter-example, let's consider the Hglass datasets, which has HMeanIR of only 4.04. In this case, the result with the original datasets (0.8145) had the largest improvements (0.9394) between all datasets (after applying HROS-FD with an increase of 10%). Therefore, although the hierarchical dataset imbalanceness does influence the resampling and classification results, we may not conclude that this relation is necessarily inverse, i.e., when the imbalanceness is low, the resampling algorithms will not really impact in the classification results.

8.4 Final Considerations

In this Chapter we proposed novel global random resampling methods to handle imbalanceness in hierarchical classification problems. The proposed methods can be subdivided in four algorithms: Random Oversampling and Undersampling for Hierarchical

Datasets with Full Depth Prediction (HROS-FD and HRUS-FD); and Random Oversampling and Undersampling for Hierarchical Datasets with Partial Depth Prediction (HROS-PD and HRUS-PD). These algorithms are, to the best of our knowledge, the first ones to deal with imbalanceness in hierarchical datasets handling and resampling the data in a direct-way.

In order to retrieve the set of majority/minority label paths in a given hierarchical dataset, we have used the Imbalance Ratio per Label Path (IRLP) and Hierarchical Mean Imbalance Ratio (HMeanIR) metrics, defined in Chapter 6. While the Full Depth resampling algorithms deal with the data in unique looping though the majority/minority label paths, the Partial Depth algorithms handle the data walking through the label tree in a bottom-up order, resampling their instances and re-calculating the IRLPs to retrieve the updated sets of majority/minority paths.

Experimental results with twenty three datasets ranging a different broad of characteristics, such as partial/full depth prediction and single/multiple paths, showed that the proposed algorithms can statically improve the classification results, in relation to the *weighted*-AUPRC, for all scenarios.

In the next Chapter we present the Hierarchical Synthetic Oversampling Technique (HSMOTE), the first heuristic global resampling method for hierarchical classification datasets. HSMOTE uses the same minority samples selection approach proposed in this Chapter in order to obtain the set of samples that will be processed by the algorithm.

Chapter 9 Global Approaches: The Hierarchical Synthetic Oversampling Algorithm

As state in Chapter 8, dealing with imbalanceness in hierarchical classification problems is a challenging task, since there are different kinds of hierarchically organized problems, with different types of properties, such as the labels taxonomy, the depth of the prediction and the number of paths associated to each example (SILLA JR; FREITAS, 2011). Moreover, in Chapter 8, we have presented HROS and HRUS, the first resampling methods that are able to deal with a hierarchical classification dataset as a whole. However, these methods aim to rebalance the dataset by randomly creating or removing samples from the dataset.

The main contribution of this Chapter is an adapted version of the SMOTE algorithm, that is able to tackle the imbalanceness problem in hierarchical classification datasets. The adapted version, named Hierarchical Synthetic Oversampling Technique (HSMOTE), was designed considering the different properties of hierarchical classification datasets regarding the depth of the prediction (partial or full depth) and the number of label paths associated with each sample (single paths or multiple paths).

The contributions of this chapter were submitted to the journal Knowledge-Based Systems and are under review for publication.

9.1 The Synthetic Oversampling Techniques

Among all the resampling techniques, the Synthetic Oversampling Technique (SMOTE), first proposed in Chawla et al. (2002), is by far the most important and well-known resampling method in the literature (KRAWCZYK, 2016). The SMOTE technique was proposed both in the binary/multi-class and multi-label scenarios. In the multi-label context, the proposed algorithm is named Multi-Label Synthetic Oversampling Technique (MLSMOTE) and was proposed by Charte et al. (2015b).

As shown in Chapter 3, SMOTE/MLSMOTE proposes the creation of new synthetic samples for the minority class(es) by interpolation of nearest instances. After selecting a sample from the minority class, two of its k nearest neighbors are randomly chosen and one new sample will be created in the direction of each one.

The main differences between SMOTE and MLSMOTE are: (i) the minority samples

selection and (ii) the labels interpolation. In the classic SMOTE, the selection of minority samples is quite natural, since we only have to rank the number of samples per label in the dataset and choose the least present one. Moreover, in classic SMOTE, the labels interpolation do not exist, inasmuch we are dealing with samples which are always labeled with the same labels. However, in MLSMOTE, the minority labels have to be selected based in a pre-defined metric, which measures the least frequent labels among all the label sets in the dataset. Furthermore, when creating a new sample, MLSMOTE interpolate the neighbors' labelsets by using Ranking, Union and Intersection techniques.

Other well-known variations of SMOTE are the SMOTE Borderline-1 and SMOTE Borderline-2, proposed by Han, Wang & Mao (2005). In these variations, the algorithm creates synthetic samples taking into account the borderlines between the classes, forcing the synthetic process to avoid the creation of samples close to these areas.

In Sharma et al. (2018) the authors have proposed another SMOTE solution specifically for extreme classification scenarios named Sampling With the Majority (SWIM). Their method uses the information inherent in the majority class to synthesize minority class data. To do so, the algorithm generates synthetic data that is at the same *Mahalanbois* distance from the majority class as the known minority instances.

An important observation to state here is that SMOTE algorithm has already been proposed in the binary, multi-class, multi-label and extreme classification scenarios. Thus, in this Chapter, we propose an adaptation of the SMOTE algorithms in order to deal with the specificities of the hierarchical classification problems.

9.2 The Proposed HSMOTE Technique

The SMOTE and MLSMOTE algorithms were designed to work with data organized in a flat taxonomy, i.e., without hierarchical relationship between the labels. However, in the hierarchical classification scenario, instead of simply producing synthetic samples for single or multiple classes, the resampling algorithm has to deal with the labels hierarchy, i.e., when creating a synthetic sample for a certain label "A/B/C", for example, the method has to be aware that it will automatically create a new sample to the class A/B as well, since they belong to the same path.

As defined in Silla Jr & Freitas (2011), there are different types of hierarchical classification problems that have to be considered by the resampling algorithm: number of paths (defined in the literature as Ψ) and depth of the paths (defined in the literature as Φ). Besides, as in all oversampling algorithms, we have to define and select the set of samples belonging to the minority classes. And finally, as in the SMOTE technique and its adaptions, we also have to define how to select the nearest neighbors, how to generate the

synthetic feature set and how to combine the labelsets. Thus, there are five main aspects to solve:

1. Minority instances selection: A criterion to define and select which label paths belong to the minority set of paths has to be established.
2. Different kinds of hierarchical problems and relationship between the labels: The resampling process has to be investigated in each hierarchical classification scenario (full or partial depth prediction and single or multiple paths) and a mechanism to deal with the labels hierarchy (mainly in partial depth problems) has to be defined.
3. Nearest neighbor search: Given an instance that belongs to a minority label path, the algorithm has to search its nearest neighbors which will be used to generate the synthetic sample.
4. Feature set generation: After selecting the neighbors, the set of features for the synthetic sample is obtained through interpolation techniques.
5. Synthetic labelset production: Since we have different kinds of hierarchical classification problems, the production of synthetic path(s) also depends on the type of the problem.

In the following subsections we detail each one of the previously mentioned aspects.

9.2.1 Minority Instances Selection

When oversampling a dataset, the first task is to define the target samples of this resampling process. When we are applying an oversampling algorithm, such as HSMOTE, the samples belonging to the minority classes have to be defined and selected. To do so in the binary and multi-class contexts (such as in the classic SMOTE), the majority/minority classes can be identified by considering the most/less frequent labels among the samples. Moreover, for the identification of majority and minority labels in multi-label datasets (as in MLSMOTE), Charte et al. (2013) defined and suggested the use of the IRLbl and MeanIR imbalance measures. Their idea was to consider the labels with an imbalance ratio (IRLbl) below the average imbalance ratio (MeanIR) as belonging to the set of majority classes (named as majority bag), otherwise the classes belong to the minority bag.

Thus, before proposing an oversampling method for hierarchical datasets, we first have to establish a mechanism to identify the minority classes considering the class hierarchy. As in hierarchical problems the classes are represented by label paths in the tree taxonomy instead of individual labels, we used the imbalance ratio per label path (IRLP) and hierarchical mean imbalance ratio (HMeanIR), proposed in Pereira, Costa &

Silla Jr (2018) and presented in Chapter 6 of this Thesis. Our idea here is to find the set of minority paths in the dataset based on their imbalance ratio, which can be calculated with Equations 9.1 and 9.2.

As presented in Chapter 6, Formula 9.1 defines the imbalance ratio for a certain label path p as $IRLP(p)$, where p is the set of all possible Label Paths that have at least one occurrence, P_i is the *i-th* label path, and the dataset is represented as D.

$$IRLP(p) = \frac{\max\limits_{p' \in P}(\sum_{i=1}^{|D|} h(p', P_i))}{\sum_{i=1}^{|D|} h(p, P_i)}$$

$$h(p, P_i) = \begin{cases} 1, & p \in P_i \\ 0, & p \notin P_i \end{cases} \tag{9.1}$$

Furthermore, in Pereira, Costa & Silla Jr (2018) Equation 9.2 is also defined in order to retrieve the mean imbalanceness of a hierarchical dataset (named as *HMeanIR*) based on the average of the imbalanceness per label path, previously presented by $IRLP$.

$$HMeanIR = \frac{1}{|P|} \sum_{p=P_1}^{P_{|P|}} IRLP(p) \tag{9.2}$$

Based on Equations 1 and 2 we are able to retrieve the minority paths of a given dataset. In this context, the set of minority paths will contain the label paths with an $IRLP$ above the HMeanIR.

9.2.2 Dealing with different kinds of hierarchical classification problems

Although in this work we are only concerned with hierarchical classification problems with labels organized in a tree-based taxonomy, in order to design an oversampling method to hierarchical datasets, we have to consider two other problems variants, as described in Silla Jr & Freitas (2011): number of paths (defined in the literature as Ψ) and depth of the paths (defined in the literature as Φ).

In full depth classification problems, the samples may be associated with one or more label paths, but always with full depth in the label tree. As in this type of problem there are no samples labeled with sub-paths of paths (for example, some samples labeled with "A/B" and others with "A/B/C"), the resampling process is similar to the classic SMOTE (when dealing with a single path prediction problem) and to MLSMOTE (when dealing with a multiple path prediction problem).

In partial depth classification problems, the samples may also be associated with one or more label paths, however these paths can achieve full or partial depth in the label tree hierarchy. One of the main challenges of resampling this kind of samples is that, when creating samples from children labels, the number of samples belonging to parent labels nodes will be also indirectly increased. This issue does not affect full depth problems, since there are no instances labeled exclusively with internal nodes labels. In Figure 60 we show an example of this issue for an example training set with 6 samples ($S_1 ... S_6$). In this case, we simulated the creation of 3 samples labeled with node H to show how it can impact in the internal label nodes. We may note that, when we created these samples for node H, we indirectly created samples for nodes E and D.

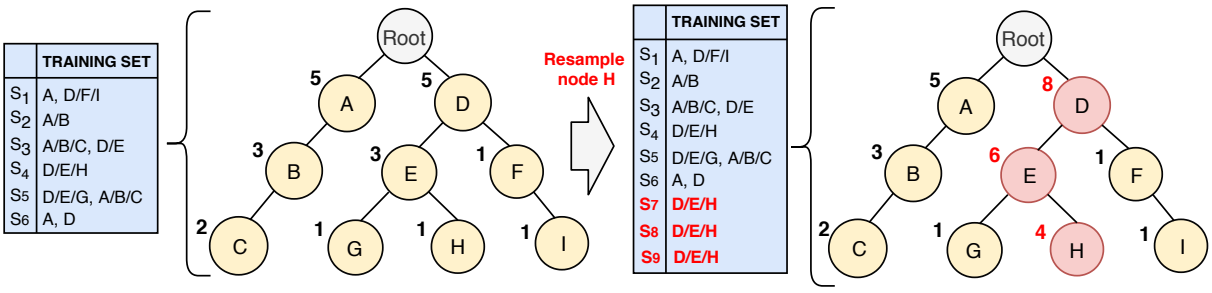


Figure 60 – An example of the main issue when creating samples in leaf nodes. The numbers on the top left of the nodes symbolize the number of samples belonging to each node.

In order to deal with the previously presented issue, we used a method to process the instances in a "bottom-up order", in which we recalculate the set of minority paths in each loop during the resampling process. In Figure 61 we present a visual example of this "bottom-up order" process. In the example, the dataset is composed of eighty five instances and a label tree with ten nodes. Besides, since we have a label tree with depth of three, the resample process occurs in three steps. We simulated the application of an oversampling method (such as HSMOTE). We may observe that initially we had five label paths belonging to the set of minority paths: "A/B", "A/B/C", "D/E/G", "D/F" and "D/F/I". However, after applying the first looping of the resampling, which considers the labels with depth 3 ("A/B/C", "D/E/G" and "D/F/I"), the label path "A/B" do not belongs to the set of minority paths anymore. This happens because we have created samples to "A/B/C" and, indirectly for "A/B", making it achieve an IRLP higher than HMeanIR.

Moreover, in order to deal with the Ψ variants, i.e., hierarchical classification problems with single and multiple paths, we have designed different types of label combinations for the synthetic samples creation process, which are detailed explained in subsection 9.2.5.

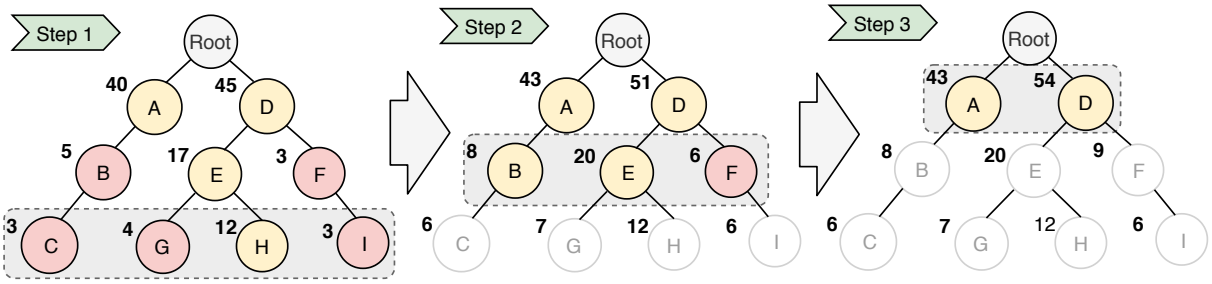


Figure 61 – An example of the application of an oversampling method in a dataset with 85 instances. The nodes marked with a circled dashed are being processed at the certain step and the red nodes represent the label paths belonging to the minority set.

9.2.3 Nearest neighbor search

The solution to this issue leans on the same strategy used in SMOTE and MLSMOTE algorithms. After selecting a sample from the set of minority label paths, we select a set of its k nearest neighbors, in which k is an entry parameter of the HSMOTE algorithm. These neighbors will be used in the interpolation phase to create the synthetic sample. In order to execute the nearest neighbors' process, we have to obtain the distances between the selected sample and its neighbors. As well as in SMOTE and MLSMOTE techniques, for the numeric features, the well-known Euclidean Distance metric is used to calculate the distances between the samples, while for the nominal features the Value Difference Metric (VDM) is used (STANFILL; WALTZ, 1986; COST; SALZBERG, 1993).

9.2.4 Feature set generation

This issue is also solved with a strategy similar to SMOTE and MLSMOTE algorithms. After selecting the two samples that will be used to generate the synthetic sample, i.e., a sample from the minority set of label paths and a reference nearest neighbor (which is randomly selected from the k nearest neighbors), an interpolation technique is used to generate the feature set of the synthetic sample. When a given feature is nominal, instead of interpolating the values, the method selects the most frequent values among the sample and all its nearest neighbors as the value for the synthetic sample.

9.2.5 Synthetic labelset production

Regarding the association of label(s) to the synthetic instance, in the classic SMOTE, as it deals with binary/multi-class problems, the label of the sample selected from the minority set is cloned to the synthetic sample. In MLSMOTE, as it handle multi-label

problems, Charte et al. (2015b) proposed the use of three label combinations techniques with the neighbors labels to solve the issue producing a new labelset: Intersection, Union and Ranking. As HSMOTE deals with hierarchical classification problems, which can has single or multiple paths (defined as Ψ) and be either full or partial depth (defined as Φ), we designed the following possibilities for the label(s) generation, according to the problem's taxonomy:

- Single Path Problems:
  - Full Depth:
    - Clone: The label path of the seed sample, i.e., the instance selected from the set of minority paths, is cloned to the synthetic sample.
  - Partial Depth:
    - Clone: The same as in FD, i.e., the label path of the seed sample is cloned.
    - Longest Common Path: The longest common path among the neighbors is chosen as the label path for the synthetic sample.
- Multiple Path Problems (FD or PD):
  - Union: All label paths that appear in the reference instance or any of its neighbors are used as the synthetic labelset.
  - Intersection: The label paths that appear in the reference instance and the neighbors are used as the synthetic labelset.
  - Ranking: We count the number of occurrences of each label path in the reference sample and its neighbors and those which are present in half or more of the instances are considered as labelset of the synthetic sample.

In Table 47 we present a visual example of the label combination process for a partial depth problem with single paths, i.e., the Clone and Longest Common Paths combinations. In the table, $Neigh_i$ is the i_{th} neighbor of the Selected Sample (which belongs to the minority set of label paths). Moreover, the example from Table 47 considers the use of five neighbors. As in the clone method we only copy the same label path from the selected sample, the combination is "A/B/C". Moreover, the Longest Common Path between the selected sample and the neighbors is "A/B", since it is present in all the samples' label paths. It is worth to mention that the Clone combination method is exactly the same for the full depth problems with single paths.

Even though in the multi-label classification scenario the possibilities of union, intersection and ranking criterion were already introduced by Charte et al. (2015b) as combinations for the MLSMOTE technique, it is important to observe that when we are

Table 47 – Example of Label Paths Combinations for a partial depth problem with single paths considering the minority label path "A/B/C" and five neighbors.

	Label Paths
Selected Sample	*A/B/C*
$Neigh_1$	A/B
$Neigh_2$	A/B/C
$Neigh_3$	A/B
$Neigh_4$	A/B/C
$Neigh_5$	A/B
	Combinations
Clone	A/B/C
Longest Common Path	A/B

dealing with hierarchical classification problems with partial depth, we also have to keep in mind this partial depth of the label paths during the combination analysis.

In order to visually explain how considering partial depth label paths can affect the multiple paths combinations, we present Tables 48 and 49, which show examples of label paths combination using the Union, Intersection and Ranking criteria for FD and PD problems, respectively. In both tables, LP_i represents the i_{th} label paths of the sample and $Neigh_i$ the i_{th} neighbor of the Selected Sample.

Table 48 shows an example of label combination for a given FD problem in which the selected sample had label paths "A/B/C" and "D/F/I" and the minority path was "A/B/C". We may observe that the combinations are somehow similar to MLSMOTE. Using the union combination process we choose all the label paths present in the samples (selected samples plus neighbors), which lead us to all label paths ("A/B/C", "D/F/I", "D/E/H" and "D/E/G"). In the intersection process we have only the label paths present in all samples (in this case, "A/B/C"). Moreover, during the ranking combination process, the most frequent label paths are computed as "A/B/C" - 6, "D/E/H" - 4, "D/F/I" - 2, and "D/E/G" - 2, thus, we choose "A/B/C" and "D/E/H" as the label paths for the synthetic sample (half of the top ranked).

However, in Table 49, which presents the combination for a PD problem in which the selected sample had the paths "A/B/C" and "D/F" and the minority path was "D/F", we have to handle the following issues in each label combination criteria:

- Union: The combination of a partial depth path and its full depth path results in the full depth path, since the partial depth path belongs to the full path. As example, we can note in Table 49 that the union of the label paths resulted in all the full label paths, so the union of "A/B/C" and "A/B" lead to "A/B/C", as well as the union of "D/F/I" and "D/F" lead to "D/F/I", and so on.

Table 48 – Example of Label Paths Combinations for a FD problem with multiple paths considering the minority label path "A/B/C" five neighbors.

	Label Paths			
	LP_1	LP_2	LP_3	LP_4
Selected Sample	*A/B/C*	*D/F/I*		
$Neigh_1$	A/B/C	D/F/I		
$Neigh_2$	A/B/C	D/E/G	D/E/H	
$Neigh_3$	D/E/H	A/B/C		
$Neigh_4$	D/E/H	A/B/C		
$Neigh_5$	D/E/H	D/E/G	A/B/C	
	Combinations			
Union	A/B/C	D/F/I	D/E/H	D/E/G
Intersection	A/B/C			
Ranking	A/B/C	D/E/H		

- Intersection: The combination of two label paths can lead to a common partial depth path between them. As example, in Table 49 we can note that the intersection between the six instances (selected sample plus the five neighbors) resulted in two partial depth paths: "D/F" and "A/B". In this case, it should be noted that the intersection of "A/B/C" and "A/B" lead to "A/B", as well as the intersection of "D/F/I" and "D/F" lead to "D/F".

- Ranking: All label paths present in the samples (partial and full depth) are ranked according to their frequency and, during this ranking step we have to take into account when a partial depth path is present in a full depth path. In the example of Table 49, the rankings are computed as "A/B" - 6, "D/F" - 6, "A/B/C" - 3, "D/F/I" - 2, "D/E/H" - 2 and "D/E/G" - 1, thus, we would choose "A/B", "A/B/C" and "D/F" as the top ranked label paths. However, when joining the top label paths, we have to use the same reasoning of the union process, i.e., as "A/B" belongs to "A/B/C", the output label paths are only "D/F" and "A/B/C".

It is important to observe that when using the intersection strategy in multiple path problems, the label sets combination may lead to an empty set. In theses cases, the solution adopted is to copy the label set from the selected sample.

9.2.6 HSMOTE Pseudocode

Considering all the aspects previously discussed, we have designed the HSMOTE algorithm through a series of nine pseudocodes:

- Algorithm 42: Presents the main procedure of the HSMOTE method.

Table 49 – Example of Label Paths Combinations for a PD problem with multiple paths considering the minority label path "D/F" five neighbors.

	Label Paths			
	LP_1	LP_2	LP_3	LP_4
Selected Sample	*A/B/C*	*D/F*		
$Neigh_1$	A/B	D/F		
$Neigh_2$	A/B	D/F	D/E/H	
$Neigh_3$	D/F/I	D/E/G	A/B	
$Neigh_4$	D/F/I	A/B/C		
$Neigh_5$	D/F	D/E/H	A/B/C	
	Combinations			
Union	A/B/C	D/F/I	D/E/H	D/E/G
Intersection	D/F	A/B		
Ranking	D/F	A/B/C		

- Algorithms 43 and 44: Present the oversampling procedure for the full depth and the partial depth hierarchical classification problems, respectively.
- Algorithm 45: Shows the calculations for the hierarchical mean imbalance Ratio ($HMeanIR$).
- Algorithm 46: Shows the calculations for the imbalance ratio for a given label path ($IRLP$).
- Algorithm 47: Shows the hierarchical resample internal procedure.
- Algorithm 48: Shows the synthetic samples creation procedure.
- Algorithms 49 and 50: Present the procedures to create the label paths for single path and multiple paths problems, respectively.

The main procedure of HSMOTE (Algorithm 42) has the following inputs:

- D - the hierarchical training set to be processed.
- *TypeOfPred* - the depth of the prediction in the hierarchical classification problem.
- *TypeOfLP* - the multiplicity of paths in the prediction.
- *LComb* - the type of label combination used to generate the synthetic samples.
- k - the number of neighbors to use.

With exception of the D and k parameters, which represent, respectively, the hierarchical training set and the number of neighbors used to build the synthetic sample,

Algorithm 42 HSMOTE main procedure

Inputs:

```
D: The hierarchical training set
depthPred: Full depth or partial depth
pathType: Single path or multiple path
lcomb: clone, longest common, ranking, union or intersection
k: Number of nearest neighbors
```

```
if depthPred = "full depth" then
    hsmote_fd(D, pathType, lcomb, k)
else if depthPred = "partial depth" then
    hsmote_pd(D, pathType, lcomb, k)
end if
```

Algorithm 43 Full Depth Oversampling

```
function hsmote_fd(D, pathType, lcomb, k)
    HMeanIR ← calculateHMeanIR(D)
    P ← labelPathsInDataset(D)
    for each labelPath in P do
        IRLP_labelPath ← calculateIRLP(D, labelPath)
        if IRLP_labelPath > HMeanIR then
            hierarchical_resample(D, labelPath, k, pathType, lcomb)
        end if
    end for
end function
```

Algorithm 44 Partial Depth Oversampling

```
function hsmote_pd(D, pathType, lcomb, k)
    labelTree ← retrieve label tree from D
    HMeanIR ← calculateHMeanIR(D)
    while labelTree is not empty do
        leafNodesSet ← getLeafNodes(labelTree)
        for each node in leafNodesSet do
            labelPath ← getPath(node)
            IRLP_labelPath ← calculateIRLP(D, labelPath)
            if IRLP_labelPath > HMeanIR then
                hierarchical_resample(D, labelPath, k, pathType, lcomb)
            end if
        end for
        remove leafNodesSet from labelTree
        HMeanIR ← calculateHMeanIR(D)
    end while
end function
```

Algorithm 45 Calculating the average imbalance ratio of the dataset

```
32: function calculateHMeanIR(D)
33:     P ← labelPathsInDataset(D)
34:     IRLP ← empty dictionary
35:     for each labelPath in P do
36:         IRLP[labelPath] ← calculateIRLP(D, labelPath)
37:     end for
38:     hMeanIR ← sum(IRLP)/|IRLP|
39:     return hMeanIR
40: end function
```

Algorithm 46 Calculating the imbalance ratio for a label path

```
41: function calculateIRLP(D, lp)
42:     countDict ← empty dictionary
43:     P ← labelPathsInDataset(D)
44:     for each labelPath in P do
45:         countDict[labelPath] ← numOfSamples(D, labelPath)
46:     end for
47:     maxCount ← max(countDict)
48:     IRLP_lp ← maxCount/countDict[lp]
49:     return IRLP_lp
50: end function
```

Algorithm 47 Hierarchical Resample for a given minority Label Path

```
51: function hierarchical_resample(D, labelPath, k, pathType, lcomb)
52:     minBag ← getAllInstancesOfLabelPath(labelPath)
53:     for each sample in minBag do
54:         neighs ← getKNearestNeighbors(sample, minBag, k)
55:         refNeigh ← getRandNeighbor(neighs)
56:         synth ← newSample(sample, refNeigh, neighs, pathType, lcomb)
57:         D ← D + synth
58:     end for
59: end function
```

the inputs of the HSMOTE algorithm are directly related to the type of hierarchical classification problem being resampled.

In the following we have divided the pseudocode explanation into four subsections, which explain how HSMOTE works when dealing with each specific type of hierarchical classification problem: (i) Full Depth with Single Paths; (ii) Full Depth with Multiple Paths; (iii) Partial Depth with Single Paths; and (iv) Partial Depth with Multiple Paths.

9.2.6.1 How HSMOTE deals with Full Depth problems with Single Paths

Lines 1-5 of the HSMOTE main procedure (Algorithm 42) are aimed to choose the kind of problem regarding the depth of the prediction. In this case, *depthPred* parameters

Algorithm 48 Creating new synthetic sample

```
60: function newSample(sample, refNeigh, neighbors, pathType, lcomb)
61:     synth ← new Sample
62:     for each feat in synthSmpl do
63:         if typeOf(feat) is numeric then
64:             diff ← refNeigh.feat - sample.feat
65:             offset ← diff * randomize(0,1)
66:             value ← sample.feat + offset
67:         else
68:             value ← mostFrequentValue(neighbors, feat)
69:         end if
70:         synth.feat ← value
71:     end for
72:     if pathType = "single path" then
73:        synth.labelPath ← buildSinglePath(sample, neighbors, lcomb)
74:     else if pathType = "mutiple path" then
75:        synth.labelPaths ← buildMultiPaths(sample, neighbors, lcomb)
76:     end if
77:     return synth
78: end function
```

Algorithm 49 Building the label path for single path problems

```
79: function buildSinglePath(sample, neighbors, lcomb)
80:     if lcomb = "clone" then
81:         return sample.labelPath
82:     else if lcomb = "longest common" then
83:         return getLongestCommonPath(sample, neighbors)
84:     end if
85: end function
```

Algorithm 50 Building the set of label paths for multiple paths problems

```
86: function buildMultiPaths(sample, neighbors, lcomb)
87:     if lcomb = "ranking" then
88:         rankedPaths ← rankMostFrequentPaths(sample, neighbors)
89:         labelPaths ← getHalf(rankedPaths)
90:     else if lcomb = "union" then
91:         labelPaths ← union(sample, neighbors)
92:     else if lcomb = "intersection" then
93:         labelPaths ← intersection(sample, neighbors)
94:     end if
95:     return labelPaths
96: end function
```

needs to be set as "full depth" and $hsmote\_fd$ function is called (represented by Algorithm 43). The first task of $hsmote\_fd$, is to calculate the mean hierarchical imbalance ratio of the training set by using the *calculateHMeanIR* function (Algorithm 45).

In order to calculate HMeanIR, Algorithm 45 retrieves all label paths in the training set D and named it P (line 33). Thus, in lines 34-37, Algorithm 45 calculates the individual IRLP for each label path in P by calling the *calculateIRLP* function (presented in Algorithm 46) and then calculates the average of all these IRLPs (line 38). By its turn, Algorithm 46 calculates the imbalance ratio for a given label path (lp) by counting the number of samples belonging to each label path in the training dataset (Lines 42-46). It is important to state that we are considering all samples that are labeled with the given *path*, for example, if the *path* is "A/B", samples labeled with "A/B/C" and "A/B/D" are taken into account. Then, in line 47 the algorithm gets the maximum number of samples of a label path in the dataset ($maxCount$) and in line 49 the $IRLP$ for the label path lp is calculated using the $maxCount$.

Going back to line 9 of Algorithm 43, after retrieving the HMeanIR value, the algorithm retrieves the label paths in the dataset (line 8) and, in lines 9-14, makes a loop for each one of these label paths, calculating their IRLP and verifying if they belong to the minority set by checking if their IRLP is above the average (HMeanIR). If a given label path belongs to the minority set, the algorithm calls the $hierarchical\_resample$ function (Algorithm 47) in line 12, which will, in fact, create the new synthetic samples for the given label path. It is important to observe that HSMOTE will create one synthetic sample for each sample labeled with a minority label path.

In line 52 of Algorithm 47, the minority bag of instances (*minBag*), which contains all the samples labeled with the minority label path is retrieved. Moreover, in the loop from lines 53-58, for each sample from the *minBag*, a new synthetic sample is created. To do so, in line 54 the algorithm obtains the k-nearest neighbors of the selected sample in relation to the other samples from the *minBag*. It should be noted that these distances are calculated ignoring the label paths, i.e., considering only the feature values. Besides, the value of k is an input parameter of HSMOTE. In line 56 a reference neighbour (*refNeigh*) is randomly selected, and will be used in the interpolation step. In line 56 the *newSample* function is called (presented in Algorithm 48) to create the synthetic sample considering its neighbors.

In line 61 of Algorithm 48 the new Synthetic sample is instantiated as *synth*. Then, in the looping from lines 62-71, each feature of *synth*, with exception of the label path, is created. At each step, if the given feature is numeric, the new feature is generated by interpolating the features from the selected sample and the *refNeigh*. However, if the feature is nominal, the most frequent value among all neighbors is chosen for the synthetic sample. In lines 72-76 the synthetic label path is generated. As we are dealing with a single

path problem, the *pathType* parameter should be set to "single path" and, thus, line 73 will be executed in order to call the *buildSinglePath* function (Algorithm 49), which will create the label path for the synthetic sample *synth*.

In Algorithm 49, when dealing with full depth hierarchical classification problems with single paths of labels, the only possible label combination (represented by *lcomb* parameter) is to clone the same label path from the selected sample to the synthetic instance, since the selected sample and its neighbors are all labeled with the same label path, which belongs to the minority set. Thus, in line 81 the label path from the selected sample is returned.

Back to algorithm 48, the synthetic sample is fully created and, in line 77, it is returned to Algorithm 47. By its turn, in line 57 of Algorithm 47 the new synthetic sample is added to the training set and Algorithm 43 continues the looping process until there are no more samples belonging to minority label paths to process.

9.2.6.2 How HSMOTE deals with Full Depth problems with Multiple Paths

When dealing with full depth problems with multiple paths, in Algorithm 42 the *depthPred* parameter needs to be set as "full depth" and the *pathType* parameter should be set to "multiple path". By doing this, the main difference between HSMOTE's flow in relation to the detailed explanation presented in section 9.2.6.1 is the synthetic label paths generation. This difference begins in lines 72-76 of Algorithm 48. As the *pathType* is set as "multiple path", *buildMultiPaths* function (Algorithm 50) is called in line 76.

In Algorithm 50, the *lcomb* parameter is used to decide which kind of label combination should be used to combine the label paths from the selected sample and the neighbors in order to generate the synthetic set of label paths. The ranking, union and intersection criteria can be chosen as input of HSMOTE for this kind of hierarchical classification problem. Using the ranking criteria, in line 88, the most frequent paths are ranked and, in line 89, the half (plus 1 if dealing with an odd number of label paths) is selected as the synthetic set of label paths. When using the union or intersection criteria, the union or intersection between the samples label paths are computed. These combination strategies are exemplified in Table 48.

9.2.6.3 How HSMOTE deals with Partial Depth problems with Single Paths

To deal with this kind of hierarchical classification problem, in Algorithm 42, the *depthPred* and *pathType* parameters should be set as "partial depth" and "single path", respectively. Therefore, in line 4 of Algorithm 42, the *hsmote_pd* function (presented in Algorithm 44) is called in order to handle the partial depth problem, which is the main

flow difference between sections 9.2.6.1 and 9.2.6.2.

The first step of Algorithm 44, in lines 17, is to retrieve the tree representing the label paths of the training set D. In line 18, the HMeanIR value is calculated following almost the same flow as presented in section 9.2.6.1, with the only difference being regarding the *labelPathsInDataset* fuction (used in line 33 of Algorithm 45 and line 43 of Algorithm 46). As we have a partial depth problem, this function has to take into account these "partial" label paths. In the loop from line 19-30, the label tree is walked in a bottom-up order. In line 20, the set of leaf nodes is retrieved. Then, in the loop from line 21 to 26, each node from the set of leaf nodes is processed, retrieving its full path (line 22), calculating its IRLP (line 23) and checking if it belongs to the set of minority label paths. When the leaf node label path belongs to the set of minority label paths, in line 25 the *hierarchical_resample* function (Algorithm 47) is called, which will work almost in the same way as presented section 9.2.6.1. After creating the synthetic samples for the leaf nodes belonging to the minority set, the algorithm removes this set of leaf nodes from the label tree (line 28), recalculating the HMeanIR of the training set (line 29). It is important to highlight the importance of this bottom-up process for the partial depth problems, recalculating the HMeanIR at each step, since the imbalanceness of the internal nodes may change during the resampling process.

The other difference when dealing with partial depth problems is regarding the creation of the label path for the synthetic sample. As we are dealing with a single path problem, function *buildSinglePath* (Algorithm 49) is called from line 73 of Algorithm 48. However, differently from the process presented in section 9.2.6.1, the label paths from the neighbors may be slightly different, since some neighbors label paths may be deeper than others into the label tree hierarchy. Thus, besides the "clone" label combination, *lcomb* parameter we may be also set as "longest common path", as the example shown in Table 47.

9.2.6.4 How HSMOTE deals with Partial Depth problems with Multiple Paths

Finally, to deal with partial depth problem with multiple paths, in Algorithm 42, the *depthPred* and *pathType* parameters should be set as "partial depth" and "multiple paths". As well as in section 9.2.6.3, which also presents a partial depth flow for HSMOTE, in line 4 of Algorithm 42, the *hsmote_pd* function (presented in Algorithm 44) is called. The main difference here is related to the label combination process, which happens inside the *buildMultiPaths* function (Algorithm 50). Although there are no visual differences in the pseudocode if compared to the full depth with multiple paths problems flow (section 9.2.6.2), it should be noted that these label combinations, i.e., the ranking, union and intersection, has to take into account the issues presented in Table 49. This means that the functions of lines 88, 91 and 93 has to be aware of subpaths of paths during the

combinations.

9.3 Experimental Protocol and Results

In this section we present the datasets, algorithms and parameters, experimental setup and classification results regarding the proposed resampling algorithms.

9.3.1 The Datasets

In order to cover the different aspects of hierarchical classification problems, we performed computational experiments in a total of 23 datasets: 9 with full depth of prediction and 14 with partial depth of prediction. It is important to state that all datasets, algorithms implementations and detailed results can be obtained in this link[12].

Tables 50 and 51 present a detailed description of the datasets with FD problems and PD problems, respectively. The datasets marked with (*) were originally proposed as flat multi-label classification datasets in the literature and in this work were adapted to a hierarchical taxonomy. The datasets marked with (**) were extracted as single-label subsets from the original multi-label dataset and were also adapted to a hierarchical taxonomy. The hierarchical adaptations were manually made considering the intrinsic characteristics of the each problem domain. For instance, in the Enron dataset, which is composed of e-mail subjects classification examples, the subjects were hierarchically organized according to the subjects.

Table 50 – Hierarchical Classification Datasets with FD Problems. Datasets marked with (*) were originally proposed as flat datasets and were adapted to a hierarchical taxonomy.

Paths	Domain	Name	Train	Test	Attr.	Lbls	HMeanIR	Reference
Multiple	Text	Enron*	988	660	1001	57	84.30	(KLIMT; YANG, 2004b)
	Music	CAL500*	351	151	68	164	22.06	(TURNBULL et al., 2007)
		Emotions*	392	203	72	9	1.48	(TROHIDIS et al., 2008)
		Birds*	272	79	260	49	5.41	(BRIGGS et al., 2013)
Single	Biology	Actinopterygii	15705	6739	15	30	21.52	
		Diptera	15194	6528	33	29	2.05	(PARMEZAN; SOUZA; BATISTA, 2018)
	Audio	Instrument	6583	2836	30	46	11.28	
	Glass	Hglass	144	70	9	11	3.91	(METZ et al., 2011)
	Image	ImCLEF07D	10000	1006	80	24	55.97	(DIMITROVSKI et al., 2011a)

Looking at the datasets characteristics we may observe that FMA-MFCC is by far the largest one, while Hglass is the smallest one. Furthermore, Hglass is the dataset with the lowest number of attributes, while Enron dataset has the most. We can also observe the Hierarchical Mean Imbalance Ratio (HMeanIR) for the datasets. We may

[12] https://sites.google.com/view/hierarchical-imblearn/

Table 51 – Hierarchical Classification Datasets with PD Problems. Datasets marked with (*) and (**) were originally proposed as flat datasets and were adapted to a hierarchical taxonomy.

Paths	Domain	Name	Train	Test	Attr.	Lbls	HMeanIR	Reference
Multiple	Biology	Cell-cycle	2484	1281	78	180	410.69	(CLARE; KING, 2003)
		Church	2474	1281	24	180	1022.80	
		Derisi	2450	1275	62	180	1038.41	
		Eisen	1587	837	80	170	727.79	
		Exp	2488	1291	544	180	1025.75	
		Gasch-1	2480	1284	174	180	1024.47	
		Gasch-2	2488	1291	53	180	1025.75	
		Phenotype	1009	582	64	168	587.17	
		Sequence	2580	1339	437	180	1016.30	
		SPO	2437	1266	79	180	1027.81	
Single	Music	FMA-MFCC*	33259	14274	13	97	179.97	(DEFFERRARD et al., 2017)
		FMA-SLLBP**	15331	7000	59	135	539.07	
		FMA-SLSSD**	15631	6700	161	135	539.07	
	Image	Diatoms	2065	1054	371	398	118.20	(DIMITROVSKI et al., 2012)

note that the datasets present a large variety of imbalanceness. Furthermore, the FD datasets are less imbalanced than the PD datasets, with Emotions dataset, for example, reaching a HMeanIR of 1.48. On the other hand, Church, Derisi, Exp, Gasch-1, Gasch-2, Sequence and SPO are the most imbalanced datasets, with HMeanIR higher than 1000. It is important to inform that the HMeanIR values presented in Tables 50 and 51 were calculated considering the full dataset, i.e., with all sample from both training and testing sets.

It is important to observe that all datasets presented in Tables 50 and 51 have labels in a Tree taxonomy, since we are not dealing with Directed Acyclic Graphs taxonomies in this work.

9.3.2 Classification Algorithm and Parameters

For the hierarchical classification task we used the Clus-HMC framework. Clus-HMC was chosen because it is considered in the literature as the state-of-the-work hierarchical classification framework (CERRI; BARROS; CARVALHO, 2015; WEHRMANN; CERRI; BARROS, 2018; PEREIRA; GABRIEL; CERRI, 2019).

Clus-HMC is based on Predictive Cluster Trees (PCT) and generates a single Decision Tree (DT) considering an entire class hierarchy. In Clus-HMC, DTs are seen as a hierarchy of clusters where the root node contains all the training instances, while the remaining are recursively divided into smaller groups as the hierarchy is traversed towards the leaves. The classification is performed using a distance-based metric which calculates how similar an instance is to some tree.

The parameter configurations used in the algorithm are reported in Table 52.

Table 52 – Clus-HMC execution parameters.

Parameter	Value
Type	Tree
ConvertToRules	No
HSeparator	"/"
FTest	[0.001, 0.005, 0.01, 0.05, 0.1, 0.125]
EnsembleMethod	RForest
Iterations	10
VotingType	Majority
EnsembleRandomDepth	No
SplitSampling	None
Heuristic	Default
PruningMethod	Default
CoveringMethod	Standard

In these experiments, we have used the *weighted* Area Under the Precision-Recall Curve (AUPRC) metric to measure the classification results. According to Davis & Goadrich (2006), the Precision-Recall curve is more informative when there is a high-class imbalance in the data. Furthermore, we have used the holdout technique, with an approximated proportion of 70/30 between training and testing.

The results presented in Tables 53, 54, 55 and 56 are the average of ten executions of Clus-HMC before and after applying the proposed resampling algorithms in the training sets. Moreover, we have tested the values of three and five as the number of neighbors in HSMOTE (represented by the k parameter in the algorithm).

9.3.3 Results

In the following we present Clus-HMC classification results with the AUPRC measure for all the experimentally tested datasets. In all the results tables, the values in bold represent the best results achieved among the original training set and the resampled ones. We have divided the results into four tables, according to the type of hierarchical classification problem:

- Full Depth with Single Paths (Table 53);
- Full Depth with Multiple Paths (Table 54);
- Partial Depth with Single Paths (Table 55); and
- Partial Depth with Multiple Paths (Table 56).

Table 53 presents the AUPRC results for the FD with Single Paths problems. One can note that, since the datasets of this table are single labeled paths with full depth of

prediction concerning the tree hierarchy, the only possible label combination is to clone the same label path of the selected sample. In this prediction context, from the five tested datasets, the only one in which the classification model did not benefit from the resampling was the actinopterygii.

Table 53 – AUPRC classification results for the datasets with FD and single paths before and after applying HSMOTE in the training set.

Dataset	Before	After	
		k = 3	k = 5
actinopterygii	**0.7570**	0.7514	0.7484
diptera	0.6027	0.6039	**0.6164**
ImCLEF07D	0.7151	**0.7333**	0.7086
Hglass	0.8145	0.8779	**0.8950**
instrument	0.7656	**0.7898**	0.7750

Table 54 shows the AUPRC results for the FD with Multiple Paths problems. In this classification scenario, as we are dealing with multi-labeled problems, there are three possible label combinations: Ranking - "R"; Union - "U"; and Intersection - "I". We may note that, for this specific classification scenario, HSMOTE increased the results for all four datasets. Moreover, we can also observe that all these best results were obtained with five neighbors during the interpolation phase (k = 5).

Table 54 – AUPRC classification results for the datasets with FD and multiple paths before and after applying HSMOTE in the training set.

Dataset	Before	After					
		k = 3			k = 5		
		R	U	I	R	U	I
Birds	0.4536	0.4562	0.4419	0.4329	0.4206	0.4480	**0.4708**
Enron	0.5693	0.5676	0.5713	0.5699	**0.5865**	0.5561	0.5709
CAL500	0.4942	0.4918	0.5038	0.4950	**0.5259**	0.4864	0.4955
Emotions	0.6843	0.6770	0.7013	0.6635	**0.7040**	0.6717	0.6777

Table 55 shows the AUPRC results for the PD with Single Paths problems. In this classification context, there are two possible label combinations: Clone - "C" and Longest Common Path - "LC". As well as in the previous classification scenario, in this context all datasets were benefited by HSMOTE during the training step. However, in this case, there was no unanimity between the best results in terms of k value and Label Combination.

Finally, Table 56 presents the AUPRC results for the PD with Multiple Paths problems. It can be noticed that, as it is also a multi-labeled problem, there are three types of label combination: Ranking - "R"; Union - "U"; and Intersection - "I". In this scenario, the HSMOTE algorithm improved the classification results for all the eleven datasets. Furthermore, with exception of the *Union*, the best results were obtained with different values for the k parameter and different types of label combinations.

Table 55 – AUPRC classification results for the datasets with PD and single paths before and after applying HSMOTE in the training set.

Dataset	Before	After			
		k = 3		k = 5	
		C	LC	C	LC
Diatoms	0.2582	**0.2720**	0.2581	0.2552	0.2524
FMA-SL-LBP	0.3268	0.3306	0.3283	0.3269	**0.3493**
FMA-SL-SSD	0.2700	0.2663	0.2803	0.2692	**0.2855**

Table 56 – AUPRC classification results for the datasets with PD and multiple paths before and after applying HSMOTE in the training set.

Dataset	Before	After					
		k = 3			k = 5		
		R	U	I	R	U	I
Cell-cycle	0.1307	0.1391	0.1406	0.1480	**0.1513**	0.1341	0.1491
Church	0.1222	0.1203	0.1186	0.1214	0.1203	0.1167	**0.1226**
Derisi	0.1309	0.1221	0.1331	**0.1354**	0.1284	0.1277	0.1341
Eisen	0.1483	**0.1826**	0.1599	0.1722	0.1616	0.1572	0.1697
Exp	0.1606	**0.1797**	0.1562	0.1723	0.1659	0.1478	0.1650
FMA-MFCC	0.2803	0.2802	0.2793	0.2809	**0.2822**	0.2798	0.2804
Gasch-1	0.1544	0.1731	0.1625	0.1669	**0.1896**	0.1581	0.1675
Gasch-2	0.1410	0.1470	0.1446	0.1615	**0.1710**	0.1509	0.1461
Phenotype	0.1256	0.1259	0.1215	0.1249	0.1290	0.1254	**0.1299**
Sequence	0.1683	0.1597	0.1378	0.1623	0.1625	0.1270	**0.1731**
SPO	0.1342	0.1276	0.1191	0.1434	0.1221	0.1155	**0.1440**

9.4 Analysis and Discussion

Observing the previously described results, four main questions are raised: (i) Is HSMOTE able to decrease the HMeanIR of the datasets? (ii) Can the HSMOTE resampling algorithm increase the classification results? (iii) Which datasets were the most/least benefited by the HSMOTE algorithm in the classification? (iv) Which ones are the best/worst label combination strategies and number of neighbors parameter?

Each one of these questions are answered in the following subsections.

The impact of HSMOTE in the HMeanIR

In order to investigate this task, in Table 57 we present the HMeanIR measures of training sets used in the experiments before and after applying the HSMOTE technique. It is important to cite that, as we have tested different parameters in the experiments (such as k neighbors and label combinations), in order to calculate the "after" HMeanIR value, we have considered the resampled training sets which obtained the best classification

results.

We can observe that HSMOTE was able to reduce the mean imbalance ratio in eighteen out of the twenty-three datasets, with a percentage of decrement (represented as %Δ) between ≈14% and ≈43%. Thus, we can affirm that, in general, HSMOTE has a positive impact in the HMeanIR reduction.

Table 57 – HMeanIR values before and after applying HSMOTE in the training datasets.

Depth of Prediciton	Labedness	Datasets	hmeanir-hsmote		
			Before	After	%Δ
FD	Single Path	actinopterygii	20.28	11.67	-42.45
		diptera	2.07	1.62	-21.49
		ImCLEF07D	55.64	33.81	-39.23
		Hglass	4.04	2.40	-40.64
		instrument	10.99	6.89	-37.29
	Multiple Path	Birds	6.21	4.04	-34.99
		Enron	75.96	53.36	-29.75
		CAL500	21.78	17.52	-19.57
		Emotions	1.49	1.56	5.11
PD	Single Path	Diatoms	107.85	82.46	-23.54
		FMA-SL-LBP	496.51	389.21	-21.61
		FMA-SL-SSD	508.74	398.51	-21.67
	Multiple Path	Cell-cycle	339.55	421.48	24.13
		Church	768.00	865.21	12.66
		Derisi	773.01	873.28	12.97
		Eisen	582.73	445.75	-23.51
		Exp	769.76	517.55	-32.77
		FMA-MFCC	181.01	104.54	-42.25
		Gasch-1	769.83	545.49	-29.14
		Gasch-2	769.76	570.93	-25.83
		Phenotype	430.64	367.83	-14.58
		Sequence	765.21	535.44	-30.03
		SPO	769.96	882.74	14.65

The impact of HSMOTE in the classification results

In order to investigate the impact of HSMOTE with statistical significance, we have applied the Wilcoxon Statistical Test, stating as hypothesis that the *weighted*-AUPRC of the classification is higher after using the HSMOTE resampling method in the training sets. The test was applied for the classification results in the original training sets against the best results achieved in each dataset with the HSMOTE algorithm. The *p-values* outputs are shown in Table 58. Considering the threshold as 0.10, we can conclude that HSMOTE is statically able to improve the classification results when applied in the training sets in all hierarchical classification contexts, i.e., Single and Multiple Paths with FD or PD.

Table 58 – *P*-values of the Wilcoxon signed-rank statistical test for the best results with HSMOTE.

Depth of Prediction	labeldness	*p-value*
FD	Single Path	0.0398
	Multiple Path	0.0544
PD	Single Path	0.0328
	Multiple Path	0.0017

The most and least benefited datasets

In order to accomplish this task, which leans on the datasets most/least benefited by HSMOTE, we have presented in Table 59 the results before applying the resampling and the best result achieved in each dataset after applying HSMOTE in the training set. Moreover, the table also presents two points of view for this task: (i) the percentage of change (represented by $\%\Delta$) and (ii) the raw difference in the results.

Considering the first criteria, i.e., the percentage of change in the results, we have Eisen (18.76%) and Gasch-1 (18.57%) as the datasets most benefited by the proposed technique. Nevertheless, if we consider the difference between the raw results as the criteria, Hglass dataset is by far the most benefited, with an increment of 0.0804 in the results.

Finally, looking at the least benefited datasets, independent of criteria, actinopterygii was the worst one with a negative changing percentage of 0.74% and a decrement of 0.0056 in the raw value.

The best/worst label combination and number of neighbors strategies

To investigate this issue, i.e., define the best and worst label combination and number of neighbors strategies, we have used the statistical evaluation protocol proposed in Charte et al. (2015b). In this protocol, we calculate the ranking of the classification results after the resampling using the different parameters (label combinations and number of neighbors) based on the Friedman statistical test. In other words, the performances of the different parameters are ranked (from first to last) and an average rank is calculated for each dataset and type of problem. Tables 60, 61, 62 and 63 present the results of this test for the FD with Single Path, FD with Multiple Paths, PD with Single Paths and PD with Multiple Paths, respectively. In these tables, the strategies are grouped together and "C-3", for example, stands for the use of the Clone ("C") label combination strategy in conjunction with k = 3 for the neighborhood selection parameter.

It is important to observe that in the FD with SP scenario, we have to clone the label path of the selected sample and, thus, there is only the k parameter as variant, which in our experiments was tested as 3 or 5. In this context, the statistical test presented in Table 60 indicated that the use of 3 neighbors (represented by C-3) usually generates

Table 59 – The best classification results before and after applying HSMOTE in each training set and its variances against the original results.

Depth of Prediciton	Labedness	Datasets	AUPRC			
			Before	After	%Δ	Diff.
FD	Single Paths	actinopterygii	0.7570	0.7514	-0.74	-0.0056
		diptera	0.6027	0.6164	2.21	0.0136
		ImCLEF07D	0.7151	0.7333	2.48	0.0182
		Hglass	0.8145	0.8950	8.99	0.0804
		instrument	0.7656	0.7898	3.06	0.0241
	Multiple Paths	Birds	0.4536	0.4708	3.65	0.0172
		Enron	0.5693	0.5865	2.93	0.0172
		CAL500	0.4942	0.5259	6.02	0.0317
		Emotions	0.6843	0.7040	2.80	0.0197
PD	Single Paths	Diatoms	0.2582	0.2720	5.05	0.0137
		FMA-SL-LBP	0.3268	0.3493	6.45	0.0225
		FMA-SL-SSD	0.2700	0.2855	5.45	0.0156
	Multiple Paths	Cell-cycle	0.1307	0.1513	13.59	0.0206
		Church	0.1222	0.1226	0.37	0.0005
		Derisi	0.1309	0.1354	3.32	0.0045
		Eisen	0.1483	0.1826	18.76	0.0342
		Exp	0.1606	0.1797	10.65	0.0191
		FMA-MFCC	0.2803	0.2822	0.67	0.0019
		Gasch-1	0.1544	0.1896	18.57	0.0352
		Gasch-2	0.1410	0.1710	17.53	0.0300
		Phenotype	0.1256	0.1299	3.33	0.0043
		Sequence	0.1683	0.1731	2.76	0.0048
		SPO	0.1342	0.1440	6.78	0.0098

a better classification result, since it got an average ranking of 1.4 considering all five datasets.

Table 60 – Ranking of the results for the strategies used in the FD with SP datasets.

Dataset	Strategy Ranking	
	C-3	C-5
actinopterygii	1	2
diptera	2	1
ImCLEF07D	1	2
Hglass	2	1
instrument	1	2
Avg.	***1.4***	*1.6*

In the FD with MP datasets, we may note through the average rankings presented in Table 61 that, in general, the best classification results were obtained with the Ranking label combination when using 5 neighbors (represented by R-5), since it got an average ranking of 2.25.

Table 61 – Ranking of the results for the strategies used in the FD with MP datasets.

Dataset	Strategy Ranking					
	R-3	U-3	I-3	R-5	U-5	I-5
Birds	2	4	5	6	3	1
Enron	5	2	4	1	6	3
CAL500	5	2	4	1	6	3
Emotions	4	2	6	1	5	3
Avg.	*4*	*2.5*	*4.75*	***2.25***	*5*	*2.5*

Furthermore, looking at Table 62, we may note that, with an average raking of 2, the best strategy for the PD with SP datasets was using the Longest Common Path as label combination with 5 neighbors (represented by LC-5).

Table 62 – Ranking of the results for the strategies used in the PD with SP datasets.

Dataset	Strategy Ranking			
	C-3	LC-3	C-3	LC-5
Diatoms	1	2	3	4
FMA-SL-LBP	2	3	4	1
FMA-SL-SSD	4	2	3	1
Avg.	*2.33*	*2.33*	*3.33*	***2***

Considering the PD with MP datasets, we may observe in Table 63 that two out of the six different strategies got the same average ranking value (2.36): R-5 and I-5. Thus, the ranking and intersection strategies generated the best classification results for this kind of hierarchical classification problem.

Table 63 – Ranking of the results for the strategies used in the PD with MP datasets.

Dataset	Strategy Ranking					
	R-3	U-3	I-3	R-5	U-5	I-5
Cell-cycle	5	4	3	1	6	2
Church	4	5	2	3	6	1
Derisi	6	3	1	4	5	2
Eisen	1	5	2	4	6	3
Exp	1	5	2	3	6	4
FMA-MFCC	4	6	2	1	5	3
Gasch-1	2	5	4	1	6	3
Gasch-2	4	6	2	1	3	5
Phenotype	3	6	5	2	4	1
Sequence	4	5	3	2	6	1
SPO	3	5	2	4	6	1
Avg.	*3.36*	*5*	*2.55*	***2.36***	*5.36*	***2.36***

Finally, as a last analysis in this context, we may observe that among all datasets and strategies, with the exception of the FD with SP, the best classification results were obtained with 5 neighbors in the neighborhood selection phase.

9.5 Final Considerations

Usually, imbalanced class distribution in datasets imposes a difficult task for many classification algorithms. Resampling the training set towards a more balanced class distribution is one of the most common and effective ways to address this issue. Although this issue has been widely studied in the literature, the authors usually focus on flat classification contexts, i.e., binary/multi-class and multi-label scenarios, ignoring problems where there is a hierarchical relationship between the classes. As far as we known, no works in the literature have investigated the proposal of resampling algorithms capable of pre-processing a hierarchical classification ataset as a whole. Considering the lack of studies in the literature, in this work we proposed the Hierarchical Synthetic Oversampling Technique (HSMOTE).

The HSMOTE technique is an adaptation of the classic SMOTE and MLSMOTE oversampling algorithms tailored to deal with hierarchical classification problems. The proposed resampling algorithms is composed of eight functions, in which the different types of hierarchical classification problems are handled based on given parameters, such as depth of the prediction in the label tree and labeldness of samples paths (single and multiple paths). In order to find the minority sets of paths, we use the HMeanIR measure, pre-processing the samples which have an IRLP above the HMeanIR. Then, depending on the dataset characteristics, the minority label paths are pre-processed in a bottom-up approach (in PD problems) or a leaf-node only way (in FD problems).

In order to create the synthetic set of label paths, we have proposed different label combinations possibilities, according to the dataset characteristics: (i) FD with SP - Cloning; (ii) FD with MP - Ranking, Union or Intersection; (iii) PD with SP - Cloning or Longest Common Path; (iv) PD with MP - Ranking, Union or Intersection, considering the match of sub-paths in the combinations.

Experimental results with twenty three datasets with different characteristics, such as partial/full depth prediction and single/multiple paths, showed that the proposed HSMOTE resampling algorithm can statically improve the classification results, in relation to the *weighted*-AUPRC, in all hierarchical classification scenarios.

In order to experimentally test and compare the contributions of this Chapter and the previously ones, in the next Chapter we present a main case study concerning an important hierarchical medical application: Identification of pneumonia pathogens, such as the SARS-COV-2, in x-ray chest images.

Chapter 10 A Case Study of Imbalanceness in COVID-19 Identification in Chest X-ray Images

The most recent novel coronavirus, officially named Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2), causes the Coronavirus Disease 2019 (COVID-19) (ORGANIZATION, 2020). The COVID-19 can cause illness to the respiratory system, fever and cough and in some extreme cases can lead to severe pneumonia (GUAN et al., 2020). Pneumonia is an infection that causes inflammation primarily in the lungs' air sacs responsible for the oxygen exchange (GUAN et al., 2020).

Pneumonia can be caused by other pathogens besides SARS-CoV-2, such as bacterias, fungi and other viruses. Several characteristics can influence its severity: weak or impaired immune system, chronic diseases like asthma or bronchitis, elderly people and smoking. The treatment depends on the organism responsible for the infection, but usually requires antibiotics, cough medicine, fever reducer and pain reliever. Depending on the symptoms, the patient may need to be hospitalized; in severe cases the patient must be admitted into an intensive care unit (ICU) to use a mechanical ventilator to help breathing (MUSHER; THORNER, 2014).

The COVID-19 pandemic can be considered severe due to its high transmissibility and seriousness (TOLKSDORF et al., 2020). The impact in the healthcare system is also high due to the amount of people that needs ICU admission and mechanical ventilators for long periods (GRASSELLI; PESENTI; CECCONI, 2020). In this scenario, early diagnosis is crucial for correct treatment to possibly reduce the stress in the healthcare system. In this context, artificial intelligence (AI) based solutions can provide a cheap and accurate diagnosis for COVID-19 and other types of pneumonia.

The standard image diagnosis tests for pneumonia are chest X-ray (CXR) and computed tomography (CT) scan. The CXR is the primary radiographic exam to evaluate pneumonia, but it not as precise as the CT scan and has higher misdiagnosis rates. Nevertheless, the CXR is still useful because it is cheaper, faster, expose the patient to less radiation and is more widespread than CT scan (SELF et al., 2013). The task of pneumonia identification is not easy, the professional reviewing the CXR needs to look for white patches in the lungs, the white patches are the lungs' air sacs filled with pus or water. However, these white patches can also be confused with tuberculosis or bronchitis, for example.

In this Chapter, we aim to explore the identification of different types of pneumonia caused by multiple pathogens using only CXR images. Despite the CT scan being the gold standard for the pneumonia diagnosis, we focused only on CXR images due to its reduced cost, fast result and its general availability, since the CT scan machines are still scarse and costly. Specifically, we considered pneumonia caused by viruses (COVID-19, SARS, MERS and Varicella), bacteria (Streptococcus) and fungi (Pneumocystis). Moreover, because of the recent pandemic which is ravaging the world, the main focus of this Case Study is to identify the pneumonia caused by the COVID-19 among the pneumonia caused by other pathogens and healthy lungs, and our main goal is to reach the best possible identification rate for the COVID-19. To support that, we have taken into account two perspectives in the results' evaluation: first, we considered all classes mentioned above and summarized the results using a macro-avg f-score; second, we considered only the COVID-19 class and summarized the results using f-score.

In order to achieve our objectives, we composed a database, named RYDLS-20, using CXR images from the open source GitHub repository shared by Dr. Joseph Cohen (COHEN; MORRISON; DAO, 2020), images from the Radiopedia encyclopedia[13] and healthy CXR images from the NIH dataset, also known as Chest X-ray14 (WANG et al., 2017). The distribution of the classes reflects a real world scenario in which healthy cases are the majority, followed by viral pneumonia, bacterial and fungi pneumonia being the least frequent, in this order. The RYDLS-20 database, which is presented in Section 10.8.1, was made freely available also as a contribution of this Chapter.

We address the problem as a hierarchical classification problem, since we can structure the different kinds of pneumonia based on the kind of pathogens that caused it. Furthermore, we are dealing with a naturally imbalanced problem, since some types of pneumonia are much more likely than others, and even pneumonia itself is less frequent than healthy cases. Thus, the use of prediction schemas that does not take into account this imbalance issue usually leads to bad performances. Besides, the main focus of this task is to identify the COVID-19 pneumonia among pneumonia caused by other pathogens, so we are aiming to increase the prediction scores for a class belonging to the set of minority labels. In this context, the use of resampling algorithms may increase the identification rate for the main target label of the task (COVID-19).

By analyzing CXR images, we could observe that texture is one of the main visual attributes present in those images. So, we decided to extract features from CXR images by exploring some popular texture descriptors, and also a pre-trained Convolutional Neural Networks (CNN) model, not to neglect the power of representation learning approaches. Thus, for the flat classification, using the extracted features, we applied some well-known multi-class classification algorithms. In parallel, we also applied a hierarchical classification

[13] https://radiopaedia.org/articles/pneumonia

approach on the same set of extracted features. It is worth mentioning that we also tried a pure deep learning (end-to-end CNN) approach, however the results were very bad, probably due to the small sample size and the class imbalance.

It is worth mentioning that some of the contributions from this Chapter were published in the Computer Methods and Programs in Biomedicine Journal (PEREIRA et al., 2020).

10.1 COVID-19 Pandemic and the Pneumonia Disease

The COVID-19 outbreak was first reported in Wuhan, China at the end of 2019, it spread quickly around the World in a matter of months. The evidence points to an exponential growth in the number of cases, as of right now there are more than 2 million confirmed cases worldwide (ORGANIZATION, 2020).

The epidemiological characteristics of COVID-19 are still under heavy investigation. The evidence so far shows that approximately 80% of patients are in mild conditions (some are even asymptomatic) and 20% are in serious or critical conditions. Moreover, around 10% need to be admitted into an ICU unit to use mechanical ventilators. The fatality rate seems to be 2%, but some specialists estimated it to be lower around 0.5% (SURVEILLANCES, 2020; REMUZZI; REMUZZI, 2020). The ICU admission is the main concern since there are a limited number of units available.

One of the main complications caused by COVID-19 is pneumonia. Pneumonia is an infection of the portion of the lung responsible for the gas transfer (the alveoli, alveolar ducts and respiratory bronchioles), called pulmonary parenchyma, that can be caused by different organisms, such as viruses, bacteria or fungi. Pneumonia cannot be classified as a single disease, but rather as a group of different infections with different characteristics (MACKENZIE, 2016).

Given that pneumonia is considered a group of diseases, the diagnosis for each one is also different. However, radiologic images, such as CXR and CT scan, are commonly used as one of the first exams to diagnose pneumonia of any kind. This happens because all kinds of pneumonia causes inflammation in the lungs, and that is captured by the radiologic images (O'GRADY et al., 2014).

Both CXR and CT scan are radiologic images that can be used aiming at the identification of pneumonia inflammation. CT scan is considered the gold standard over CXR since it is more precise. However, it has some drawbacks: it is more expensive, slower to be obtained and to an extent still rare (SELF et al., 2013). Some CT scan machines can cost up to millions of dollars, and X-ray machines cost roughly ten times less than that. So, there are still reasons to use CXR images to diagnosis pneumonia.

Following that, pneumonia detection in CXR images can be difficult even for experienced radiologists. The inflammation appear as white patches in the lungs, they can be vague, overlapped with other diseases (asthma for example) and can even be confused with benign patches. In this context, artificial intelligence solutions can be very useful to aid the diagnosis.

10.2 Hierarchical Structure of the Problem

The problem of identifying types of pneumonia based on features extracted from CXR images can be naturally casted as multi-class classification problem, since we have one label associated to each sample. However, if we look at this same problem from another perspective, we may conclude that there is a hierarchy between the pathogens that causes pneumonia.

Figure 62 shows how the types of pneumonia caused by different micro-organisms can be hierarchically organized. We may observe that there is a total of fourteen labels, in which seven are leaf nodes, i.e., which are the actual type of pneumonia. There are pneumonia caused by micro-organisms Acelullar/Virus and Cellular. By its turn, the Acelullar/Virus pneumonia can be subdivided into Coronavirus and Varicella, and the Celullar into Bacteria/Streptococcus and Fungus/Pneumocystis. Furthermore, the Coronavirus can be further subcategorized into COVID-19, SARS and MERS. This hierarchy is based on the structure developed in 10th revision of the International Statistical Classification of Diseases and Related Health Problems (ICD-10) (ASSOCIATION, 2019).

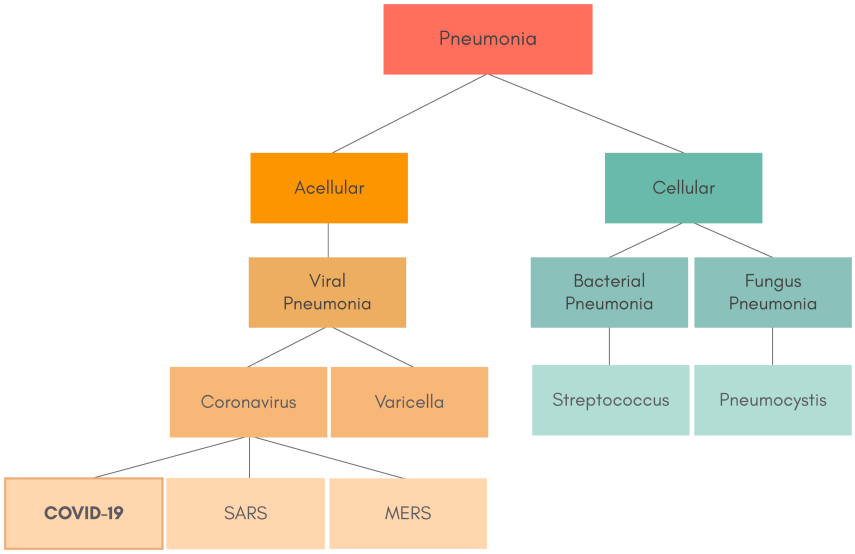


Figure 62 – The hierarchical class structure of pneumonia caused by micro-organisms.

Furthermore, following the 3-tuple definitions proposed in Silla Jr & Freitas (2011), we can define this hierarchical problem as (T, SPL, FD), i.e., the problem is proposed over

a Tree (T) hierarchy with Single Paths Labels (SPL) and Full Depth prediction (FD).

10.3 Related Works

In this section, we describe some remarkable works presented in the literature which address one of the following topics, and that have directly influenced the development of this work: texture descriptors in medical images, pneumonia detection in CXR images, and COVID-19 pneumonia detection on CXR or CT scan images using artificial intelligence. The works will be described predominantly in chronological order, and we try to highlight the main facts related to each of them, such as: the feature extraction step, whether it is performed using handcrafted features or automated feature learning, the classification model, the database used in the experiments, the type of image used in the experiments (CXR or CT), and the types of pneumonia investigated (COVID-19 among others).

The first work we are going to describe was presented by Nanni, Lumini & Brahnam (2010) in 2010. In that work, the authors compared a series of handcrafted texture descriptors derived from the Local Binary Pattern (LBP), considering their use specifically in medical applications. The variants of LBP evaluated were Local Ternary Pattern (LTP), Eliptical Binary Pattern (EBP), and Elongated Quinary Pattern (EQP). These descriptors were evaluated on three different medical applications: i) pain expression detection, starting from facial images of the COPE database (BRAHNAM; NANNI; SEXTON, 2007), taken from 26 neonates categorized into five different classes (i.e. rest, cry, air stimulus, friction, and pain); ii) Cell phenotype image classification using the 2D–Hela dataset (CHEBIRA et al., 2007), a dataset composed of 862 single-cell images distributed into ten different classes (i.e. Actin Filaments, Endsome, Endoplasmic Reticulum, Golgi Giantin, Golgi GPP, Lysosome, Microtubules, Mitochondria, Nucleolus, and Nucleus); and iii) Pap smear (Papanikolaou test) aiming to diagnose cervical cancer. They used a dataset composed of 917 images collected at the Herlev University Hospital (JANTZEN et al., 2005) using digital camera and microscope. The images were labeled according to seven different classes, being three of them related to normal states, and four of them related to abnormal states. After a comprehensive set of experiments using the Support Vector Machine (SVM) classifier, the authors verified that the EQP descriptor, or ensembles created using variations of EQP performed better for all the addressed tasks. For this reason, we decided to investigate the performance of EQP in the experiments accomplished in this work.

Still in 2010, Parveen & Sathik (2011) addressed pneumonia detection in CXR images. The authors suggested that the feature extraction could be properly made, at that time, by using Discrete Wavelet Transform (DWT), Wavelet Frame Transform (WFT), or Wavelet Packet Transform (WPT), followed by the use of fuzzy c-means classification learning algorithm. Looking backward, we can easily note that the feature extraction was

still strongly coupled up to the handcrafted perspective. However, the efforts made by the authors aiming at find useful descriptors to properly capture the information about different kinds of lung infections is worth mentioning.

Scalco & Rizzo (2017) performed texture analysis of medical images for radiotherapy applications. In this sense, the authors applied texture analysis as a mathematical technique to describe the grey-level patterns of an image aiming at characterize tumour heterogeneity. They also carried out a review of the literature on texture analysis on the context of radiotherapy, particularly focusing on tumour and tissue response to radiations. In conclusion, the authors point out that texture analysis may help to improve the characterization of intratumour heterogeneity, favouring the prediction of the clinical outcome. Some important open issues concerning the interpretation of radiological images have been raised in that work. Among these issues, we can highlight the lack of a proper biological interpretation of the models that could predict the tissue response to radiation.

Although the COVID-19 outbreak is a quite recent event, it has been attracting a lot of attention from the society and also from the image analysis research community in particular, in view of the urgency of this matter. In this sense, Zhou et al. (2020) have just published a study describing a deep learning model for differentiating novel coronavirus pneumonia (NCP) and influenza pneumonia in chest computed tomography (CT) images. The work is one of the pioneering works that has brought to light some scientific evidences concerning the challenging pandemic which has been dramatically affecting the world. By this way, it can be taken as an important reference, mainly if we take into account that the study was developed by scientists from the country from where the outbreak has emerged.

The first point to be highlighted regarding that work is that, differently from the study presented in this work, Zhou et al. (2020) adopted CT images in their study. It is particularly important to emphasize this difference, because in one hand, CT images are much better than CXR images due to its better capacity to show details from the pulmonary infection. On the other hand, CXR images are much cheaper and can be obtained in much less time, as already pointed in the introduction of this Chapter.

In the experimental protocol, Zhou et al. (2020) composed the training set using CT images scanned from 35 confirmed NCP patients, enrolled with 1,138 suspected patients. Among these images, 361 images from confirmed viral pneumonia patients were included, being 156 of them influenza pneumonia patients. In summary, the study showed that most of the NCP lesions (96.6%) are larger than 1 cm, and for 76.6% of the lesions the intensity was below -500 Hu[14], showing that these lesions have less consolidation than those provoked by influenza, whose nodes size ranges from 5 to 10 mm. Regarding the classification results, the deep model created obtained a rate above 0.93 for distinguishing between NCP and influenza considering the AUROC metric.

[14] Hounsfield unit, more details can be found in https://en.wikipedia.org/wiki/Hounsfield_scale

The authors also handle the transferability problem, aiming to avoid that the well-trained deep learning model performs poorly on data from unseen sources. In this way, Zhou et al. (2020) proposed a novel training schema, which they call Trinary schema. By this way, the model is supposed to better learn device independent features. The Trinary schema performed better than the Plain schema with specialists regarding the device-independence and consistence, achieving a f-score of 0.847, while the Plain schema obtained 0.774, specialists 0.785, and residents 0.644.

Li et al. (2020) also addressed COVID-19 identification on chest CT images using artificial intelligence techniques. For this purpose, the authors used a database composed of CT images collected from COVID-19 patients, other viral pneumonia patients, and also from patients not diagnosed with pneumonia. The images were provided by six Chinese hospitals, and the created database is composed of 2,969 images on the training set, being 400 from COVID-19 patients, 1,396 from other viral pneumonia patients, and 1,173 from non-pneumonia patients. In addition, it was created an independent test set with images from 68 COVID-19 patients, 155 other viral pneumonia patients, and 130 non-pneumonia patients, totaling 353 CT images.

A 3D deep learning model, which the authors call COVNet, was created using the ResNet-50 (HE et al., 2016) as a backbone. The model is fed by a series of CT slices and generates a classification prediction for the CT image considering the following three classes: COVID-19, other viral pneumonia, and non-pneumonia. After experimentation, the authors reported an AUROC value of 0.96 obtained for COVID-19, and 0.95 for other viral pneumonia.

Narin, Kaya & Pamuk (2020) evaluated COVID-19 detection on CXR images using three different deep neural network models (i.e. ResNet50, Inception-V3, and Inception-ResNetV2). The dataset was composed using images taken from the open source GitHub repository shared by Dr. Joseph Cohen (COHEN; MORRISON; DAO, 2020), in which we can find CXR radiographs of individuals with the following kinds of pneumonia: COVID-19, Middle East respiratory syndrome (MERS), severe acute respiratory syndrome (SARS). In addition, fifty normal CXR images were selected from Kaggle repository called "Chest X-Ray Images (Pneumonia)"[15]. The results were obtained using 5-fold cross validation, and they are as follows: 98% of accuracy using the ResNet50 model, 97% of accuracy using the Inception-V3 model, and 87% of accuracy for Inception-ResNetV2.

Gozes et al. (2020) addressed COVID-19 detection and patient monitoring using deep learning models on CT images. By patient monitoring, the authors mean the evolution of the disease in each patient over time using a 3D volume, generating what they call "Corona score". The authors claim that the work is the first one developed to detect, characterize and track the progression of COVID-19. The study was developed using

[15] https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

images taken from 157 patients, from China and USA. The authors make use of robust 2D and 3D deep learning models, they also modified and adapted existing AI models, combining the results with clinical understanding. The classification results, aiming at differentiate coronavirus images vs. non-coronavirus images obtained 0.996 of AUROC. Gozes et al. also claim that they successfully performed quantification and tracking of the disease burden.

Wang & Wong (2020) created the COVID-Net, an open source deep neural network specially created aiming to detect COVID-19 on chest radiography images. To accomplish that, the authors curated the COVIDx, a dataset created exclusively to support the COVID-Net experimentation. The dataset is composed of 16,756 chest radiography images from 13,645 different patients taken from two distinct repositories.

The initial network design prototype was created based on human-driven design principles and best practices, combined with machine-driven design exploration to produce the network architecture. The authors claim that the developed model obtained a good trade off between accuracy and computational complexity. In terms of recognition performance, they obtained 92.4% of accuracy for the COVIDx test dataset as a whole. They also reported the following sensitivity rate for each kind of infection/non-infection image: 95% for "normal" patients, 91% for non-COVID-19 infection, and 80% for COVID-19 infection. More details regarding the results, the created model, and the dataset can be found in Wang & Wong (2020).

Khan, Shah & Bhat (2020) designed the CoroNet, a Convolutional Neural Network (CNN) for detection of COVID-19 from CXR images. The CNN model is based in Xception (Extreme Inception) and contains 71 layers trained on the ImageNet dataset. The authors also developed a balanced dataset to support and test their neural network configuration, which is composed of 310 normal, 330 bacterial, 327 viral and 284 COVID-19 resized CXR images. According to the authors, the proposed CoroNet achieved an average accuracy of 0.87 and a f-score of 0.93 for the COVID-19 identification. We can highlight the following main differences between their work and ours: (i) Their dataset do not consider an unbalanced realistic scenario, thus they do not use resampling techniques; (ii) Their dataset have only four classes and it is not publicly available for download; (iii) They did not use both handcraft and representation learning features. (iv) They did not investigate a hierarchical classification approach.

Ozturk et al. (2020) proposed a deep model for early detection of COVID-19 cases using X-ray images. The author accomplished the classification in both binary (COVID vs. No-findings) and multi-class (COVID vs. No-Findings vs. Pneumonia) modes. The created model achieved an accuracy of 98.08% for binary classes and 87.02% for multi-class cases. The model setup was built with the DarkNet and named as YOLO classifier, a object detection system. The authors made the codes available and they claim that they can be

used to create a tool to assist radiologists in validating their initial screening.

It is important to mention that as the identification of COVID-19 in CXR images is a hot topic nowadays due to the growing pandemic, it is unfeasible to represent the real state-of-the-art for this task, since there are new works emerging every day. However, we may observe that most of these works are aiming to investigate configurations for Deep Neural Networks, which is already somehow different from our proposal.

Table 64 presents a summary of the studies described in this section, focusing on their most important characteristics. The main purpose of this table is to provide a practical way to find some important information regarding those works at a glance.

10.4 General Classification Schema

As aforementioned in this Chapter, we focus on exploring data from CXR images considering different feature extraction methods to classify the different types of pneumonia and, consequently, identify COVID-19 among pneumonias caused by other micro-organisms. Thus, we chose specific approaches that could lead us to obtain the best benefit in terms of the classification performance for this specific class.

The first step towards the analysis of the resampling effects into this specific hierarchical classification problem is to define the baseline classification schema. For this purpose, we employed the flat resampling algorithms in the dataset (Chapter 5). Moreover, we have tested two of the hierarchical resampling approaches proposed in this thesis: (i) Resampling with Local Classifiers Approach (Chapter 7); (ii) Global Hierarchical Resampling Algorithms (Chapters 8 and 9). The only approach that cannot be used in this classification problem is the label path conversion (Chapter 6), since it can only be applied in hierarchical problems with multiple paths prediction.

To better understand the general idea, Figure 63 shows a general overview of the classification schema, considering: The feature extraction process (Phase 1), the Early Fusion technique (Phase 2), the data resampling (Phase 3), the classification (Phase 4) and Late Fusion technique (Phase 5). It should be noted the reasoning behind this naming schema is as follows: Phases 1 and 2 are the same though all configurations, while Phases 3, 4 and 5 may change according to the resampling approach (that is why they are presented in dashed lines in the Figure 63. In the next Sections, Phases 3 and 4 are described in details for each specific resampling approach.

It is important to inform that we did not performed a pre-processing step aiming to standardize the images before the feature extraction. Thus, we are dealing with CXR images with different sizes. More information concerning the dataset are described in Section 10.8.1. In the following subsections we describe in details Phases 1, 2 and 5

Table 64 – Summary of the works described in this section.

Reference	Image Type	Database/applications	Computational/ML* techniques
Nanni, Lumini & Brahnam (2010)	Neonatal facial, fluorescence microscope and smear cells images	Three databases: Neonatal facial images, 2D-HeLa dataset and Pap smear datasets	LBP, LPQ, EQP, LTP, EBP, ILBP CSLBP and SVM
Parveen & Sathik (2011)	CXR	Pneumonia detection	DWT, WFT, WPT and fuzzy C-means clustering
Scalco & Rizzo (2017)	CT, PET and MR	Tumour heterogeneity characterization	Grey-level histogram, GLCM, NGTDM, GLRLM and GLSZM
Zhou et al. (2020)	CT	NCP/influenza differentiation images from 1,138 suspected patients, being 361 viral pneumonia, 35 confirmed NCP and 156 confirmed influenza	YOLOv3, VGGNet and AlexNet
Li et al. (2020)	CT	2,969 images obtained in Chinese hospitals 400 NCP images 1,396 other viral pneumonia and 1,173 non-pneumonia	COVNet deep learning model based on ResNet-50
Narin, Kaya & Pamuk (2020)	CXR	Non-pneumonia and pneumonia images including SARS, MERS and NCP Dr. Joseph Cohen github repository	ResNet50, Inception-V3 and InceptionResNetV2
Gozes et al. (2020)	CT	NCP detection and analysis using images taken from 157 patients	2D and 3D deep learning models, and other AI models
Wang & Wong (2020)	CXR	NCP detection using 16,756 images taken from 13,645 patients	COVID-Net a deep neural network created to detect NCP
Khan, Shah & Bhat (2020)	CXR	NCP detection using 1,251 images from four classes	CoroNet a CNN created to detect NCP
Ozturk et al. (2020)	CXR	NCP detection using 500 pneumonia images and 500 non-pneumonia images	DarkNet and YOLO

* Machine Learning

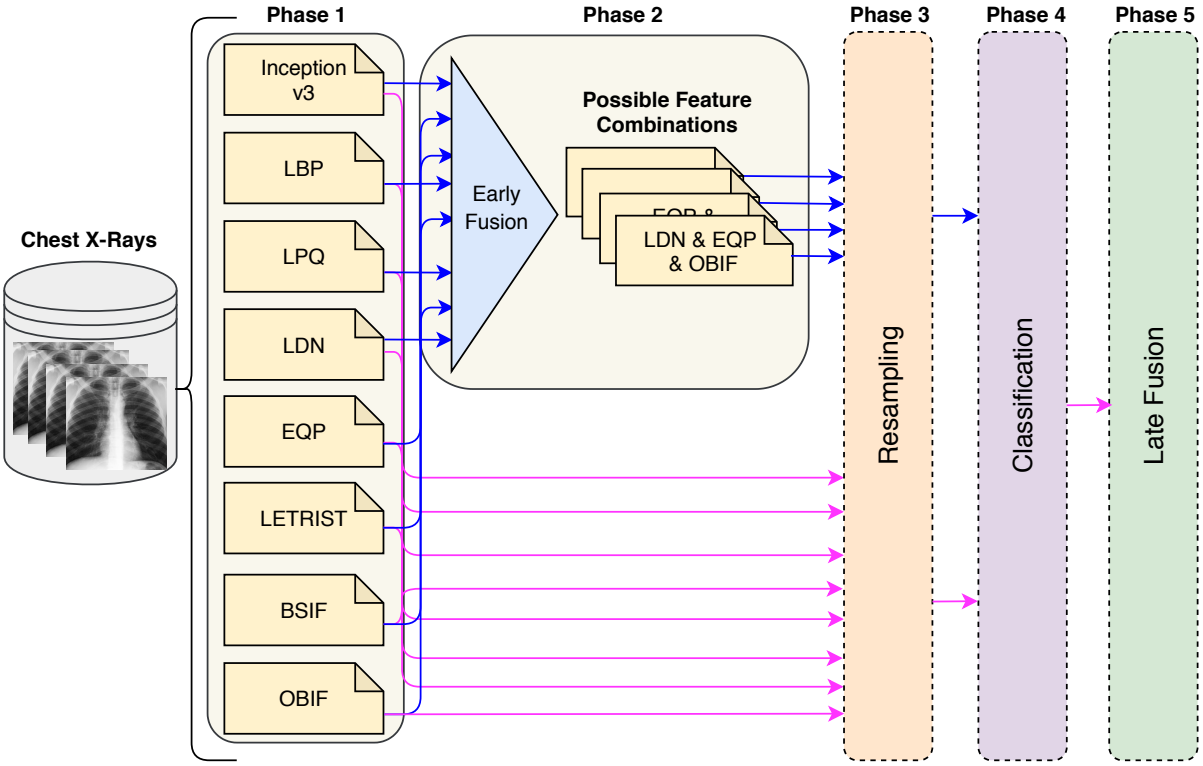


Figure 63 – A general classification schema for the COVID-19 identification in CXR images. While the blue lines represents the early fusion connections, the pink lines are used for the late fusion and results without fusion.

presented in Figure 63.

10.4.1 Feature Extraction (Phase 1)

By analyzing the CXR images, we can observe that texture is probably the main visual attribute that we can find in these images. Thus, we have explored some of the most well-successful texture descriptors described in the literature, including one automated feature learning approach.

In this section, we briefly describe the texture descriptors used in this work. The texture descriptors selected were chosen either because they presented good performance in general applications, or specifically in medical image analysis related applications.

Local Binary Pattern (LBP)

Presented by Ojala, Pietikäinen & Harwood (1996), LBP is a powerful texture descriptor, successfully experimented in several different applications which involve texture classification. LBP is found by calculating a binary pattern for the local neighborhood for each pixel of the image. The pattern has one position to each neighbor i involved in the

calculation, and it is calculated by subtracting the values of gray intensity of the central pixel (g_c) from the gray intensity of each neighbor (g_i) to get a distance d, such a way that if d is greater or equal to zero, that position in the binary pattern assumes the value 1, otherwise it assumes 0, like shown in equation 10.1.

$$d = \begin{cases} 1, & if\ g_i - g_c \geq 0 \\ 0, & otherwise. \end{cases} \tag{10.1}$$

The final texture descriptor for the image corresponds to a histogram which counts the occurrences of the binary patterns along the image. The number of possible binary patterns varies according to the setup previously defined to get pixels from the neighborhood, considering the neighborhood topology and number of neighbors for example. Regardless these details, this texture descriptor has been presenting impressive results on several different application domains for more than one decade (BERTOLINI et al., 2013; COSTA et al., 2012; FILHO et al., 2014). The details about how the patterns can be defined, and how we can set the parameters of the neighborhood can be found in (OJALA; PIETIKÄINEN; HARWOOD, 1996).

Elongated Quinary Patterns (EQP)

Elongated Quinary Pattern (EQP) (NANNI; LUMINI; BRAHNAM, 2010) is basically a variation of LBP and LTP descriptors. The great difference from EQP to LBP and LTP descriptors is that the EQP uses a quinary pattern, not binary or ternary encoding, like LBP and LTP respectively. From a given grayscale image, let us denote $\mathbf{x}$ as the central pixel using a given topology, and $\mathbf{u}$ as the gray value of the neighboring pixels. In the EQP descriptor, we can assume five different values, instead of two or three, as proposed in the LBP and LTP respectively. Thus, in quinary encoding the difference d is encoded using five values according to two thresholds τ_1 and τ_2, as described in Equation 10.2.

$$d = \begin{cases} 2, & \mathbf{u} \geq \mathbf{x} + \tau \\ 1, & \mathbf{x} + \tau_1 \leq \mathbf{u} < \mathbf{x} + \tau_2 \\ 0, & \mathbf{x} - \tau_1 \geq \mathbf{u} < \mathbf{x} + \tau_1 \\ -1, & \mathbf{x} - \tau_2 \leq \mathbf{u} < \mathbf{x} - \tau_1 \\ -2, & otherwise. \end{cases} \tag{10.2}$$

The creators of EQP evaluated different topology patterns for the neighborhood of the central pixel, instead of considering only the circular neighborhood, as originally done for LBP and LTP. The elliptical topology showed better results. Moreover, the EQP

proved to be a robust descriptor for different medical image problems, presenting results superior to other descriptors.

Local Directional Number (LDN)

LDN was originally experimented for the assessment of the face recognition task (RIVERA; CASTILLO; CHAE, 2013). This technique tries to capture the texture structure, by encoding the information about the texture direction in a compact way. In this sense, this descriptor is supposed to be more discriminative than other methods. The directional information is obtained by using a compass mask and the prominent direction indices are used to encode this information. Sign-which is also used to distinguish between similar pattern with different intensity transitions. More details about this descriptor can be find in (RIVERA; CASTILLO; CHAE, 2013).

Locally Encoded Transform Feature Histogram (LETRIST)

The LETRIST descriptor, proposed by Song et al. (2018) aims to present a simple and efficient texture descriptor. The authors describe some important characteristics that all texture descriptor must have: Discriminative, Invariant, Intense, Low-dimensional and Efficient. Basically, the histogram which corresponds to the LETRIST descriptor is based on the across feature and scale space of the image. The authors describe three main steps for generating the representation histogram. In the first step, from multiple input image scales and using Directional Gaussian Derivative Filters, extremum responses are computed. These extremum responses are used to feed linear and non-linear operators to quantitatively construct a set of transform features. This characterizes structures of local textures and their correlations with the input image. In the second step, the quantization is performed using various levels or binary threshold schemas. This step aims at greater robustness in terms of changes in lighting and rotation. Finally, in the third step, a joint cross-scale coding schema is carried out. In this way, it is possible to add discrete texture codes in a compact histogram representation. The authors describe LETRIST as a robust descriptor for different texture classification tasks (SONG et al., 2018). We can also point out that it is robust to Gaussian noise, changes in scale, rotation and lighting. In this way, LETRIST becomes a very interesting texture descriptor.

Binarized Statistical Image Features (BSIF)

Proposed by Kannala & Rahtu (2012), the BSIF texture descriptor was initially proposed for texture classification particularly on face recognition tasks. The BSIF descriptor is based both on LBP and LPQ descriptors. However, the authors emphasize that BSIF uses a schema based on statistics of natural images and not on heuristics, such as the descriptors LBP and LPQ. That is, from a small set of samples of natural images, the

descriptor learns a fixed set of filters using Independent Component Analysis (ICA). For the generation of BSIF descriptors, the value of each pixel from an input image $M \times N$ is transformed into a binary string. In this work, the feature vectors generated using BSIF have 56 dimensions. An interesting study evaluating the robustness of 27 descriptors in palmprint recognition (IDRSSI; RUICHEK, 2020) describes that the BSIF descriptor was among the Top-3 best descriptors evaluated.

Local Phase Quantization (LPQ)

Proposed by Ojansivu & Heikkilä (2008), LPQ was originally proposed aiming to provide a good texture description for noised images, affected by blur. However, surprisingly LPQ has shown to be quite effective also to describe the textural content even for images not affected by blur. This descriptor is constructed by taking the coefficients that reveal the blur intensity of the image. It is done by using the phase of 2D Short Term Fourier Transform (STFT) over a window with a previously defined size, which is slid over the image. The mathematical details regarding the LPQ can be obtained in Ojansivu & Heikkilä (2008).

Oriented Basic Image Features (OBIF)

The BIF descriptor was originally designed for texture classification (CROSIER; GRIFFIN, 2010), but it also performs well in other tasks (NEWELL; GRIFFIN, 2014). Gattal et al. (2018) proposed an extension of the BIF descriptor. The main idea is to categorize each location in the image into one of seven possible local symmetry classes. These types of local classes are the following: flat, slope, dark rotational, light rotational, dark line on light, light line on dark or saddle-like. To categorize each part of the image, six Derivative-of-Gaussian filters are used, which is determined by the α parameter. The parameter ε classifies the location probability as flat. The feature vector generated through the OBIF descriptor has 23 dimensions. The orientations were quantified at four levels ($n = 4$). Gattal et al. (2018) and Newell & Griffin (2014) propose a change in the OBIF descriptor aiming to improve its performance. In this sense, the rationale is that from two different OBIF descriptors (using different parameters σ and ε), it is possible to produce OBIF column features with $(5n + 2)^2$ dimensions. Thus, the number of dimensions was increased from 23 to 484.

Automatically Learned Features with Inception-V3

In the non-handcrafted scenario, we used the Inception-V3 (SZEGEDY et al., 2016) to perform feature extraction. This architecture proved to be more robust than other deep architectures, presenting low error rates in the ILSVRC-2012 challenge[16]. It also also

[16] http://image-net.org/challenges/LSVRC/2012/

presented results better than previous architectures, such as GoogleLeNet (SZEGEDY et al., 2015), PReLU (HE et al., 2016), and VGG (SIMONYAN; ZISSERMAN, 2014). We have used zero padding to fill the images and keep their size in the standard. After the training of the Inception-V3, we used the 2,048 weights values of the penultimate layer of the net as feature vector. Before extracting the features, we applied transfer learning using the weights of an Inception-V3 trained on the IMAGENET Dataset (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

All codes employed in this Chapter can be found through links in their respective papers. In Table 65 we describe the dimensions of the feature vectors and also the values set to their main parameters, aiming to facilitate the reproducibility by other researchers.

Table 65 – Features dimensions and main parameters.

Feature	*Parameters*	*Dimensions*
LBP	$LBP_{8,2}$	59
EQP	$loci = ellipse$	256
LDN	$mSize = 3$; $mask = kirsch$; $\sigma = 0.5$	56
LETRIST	$sigmaSet = 1, 2, 4$; $noNoise$	413
BSIF	$filter = ICAtextureFilters\text{-}11\times11\text{-}8bit$	256
LPQ	$winSize = 7$	256
OBIF	$\alpha = 2, 4$; $\varepsilon = 0.001$	484
Inception-V3	*default parameters*	2,048

10.4.2 Early Fusion (Phase 2)

This fusion technique was first used in Snoek, Worring & Smeulders (2005) and its main idea is to group the different features as a unique set of features to feed the learner. Thus, the method generates a unique dataset with all the chosen characteristics together. In our method, as we are using eight different features, we have decided to use 2×2 and 3×3 combinations, which lead us to a total of eighty four different feature sets.

10.4.3 Late Fusion (Phase 5)

In opposition to early fusion strategies, the Late Fusion technique combines the output of the learners (SNOEK; WORRING; SMEULDERS, 2005). In general, this combination is achieved by calculating a only prediction involving all the predicted scores.

According to Kittler *et al.* (KITTLER et al., 1998), the Late Fusion may achieve promising results in scenarios in which there is complementarity between the outputs. In these cases, the classifiers do not make the same misclassification and thus, when combined, they can help each other to give the best label prediction.

Among the most used fusion strategies, we can highlight the rules introduced by Kittler *et al.*(KITTLER et al., 1998) and which were used in this work:

- Sum rule (SUM): Corresponds to the sum of the predictions probabilities provided by each classifier for each label.
- Product rule (PROD): Corresponds to the product between the predictions probabilities provided by each classifier for each label.
- Voting rule (VOTE): We contabilize the votes of the classifiers in the each possible label (considering the higher probability prediction) and choose the label with the most votes.

Another aspect regarding the predictions integration is the criteria adopted to select the classifiers that will be used in the fusion. In this sense, we have tested the Top-N and Best-per-Feature fusion criteria. The Top-N consists of selecting the N tested scenarios (feature + resampling) with the best overall performance. The Best-on-Feature consists of using the best results for each feature. In our method, we have used N=5 in the Top-N approach. In the Best-on-Feature we have tested the combination 2×3, 3×3, 4×4 and 5×5 of the best result per feature.

10.5 Hierarchical Classification with Flat Resampling

The Hierarchical Classification with Flat Resampling (proposed in Chapter 5) can be considered as the baseline schema for the task proposed in this Chapter. The details for this schema applied in the proposed task are presented in Figure 64.

In Figure 64, besides the two first Phases explained in the previous section (feature extraction - Phase 1 and early fusion - Phase 2), we also have the resampling step (Phase 3), in which we use binary resampling algorithms with the O-A-A approach. In Phases 4.1 and 4.2 we use a hierarchical learner to generate the predictions for the early fusion and late fusion techniques.

10.6 Local Classifiers with Resampling

The details for the Hierarchical Classification using Local Classifiers with Resampling (proposed in Chapter 7) are presented in Figure 65.

The first two Phases are the same as described in the previous classification schema (baseline). Phase 3 presents the classification step, in which the results are generated for

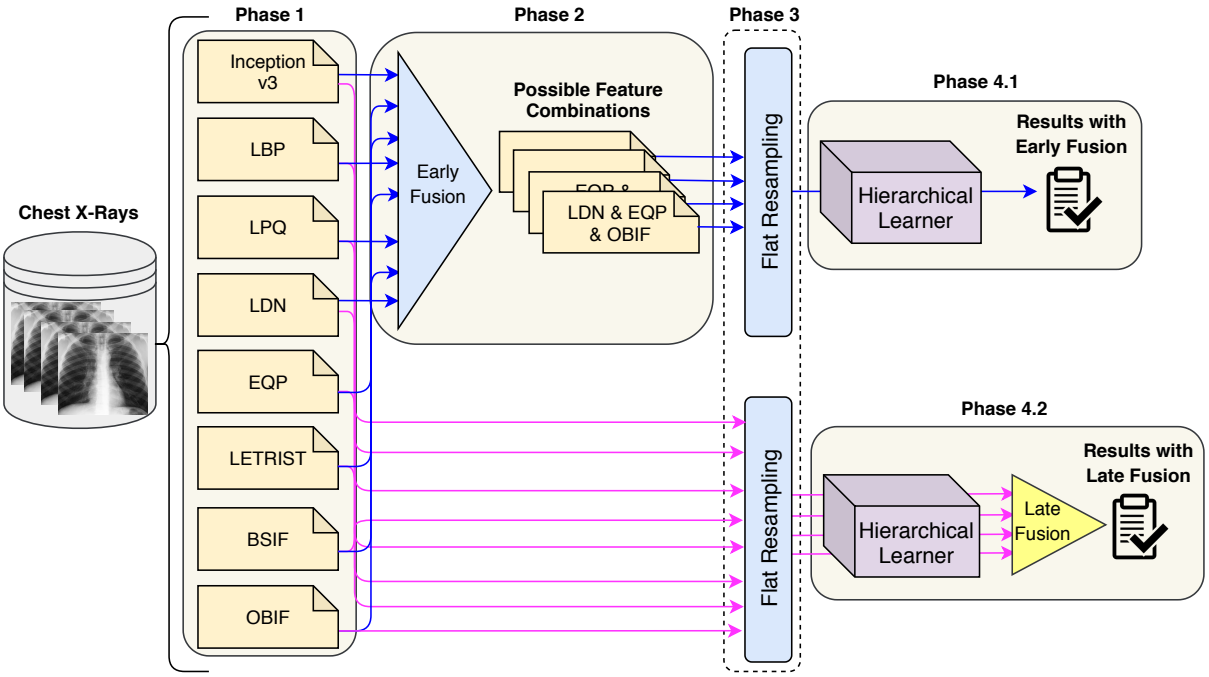


Figure 64 – The baseline resampling schema for the COVID-19 identification in CXR images.

the early and late fusion techniques. Phase 3 is subdivided into the three local classifiers approach: Local Classifiers per Parent Node (Phase 3.1); Local Classifiers per Node (Phase 3.2); and Local Classifiers per Level (Phase 3.3). As described in Chapter 6, the resampling proposed for the local classifiers is embedded in the model building step for each local node/level. It is important to observe that, in this classification approach, we did not performed the late fusion technique into the predictions.

10.7 Resampling with Global Algorithms

The details of the hierarchical classification schema using the Global Hierarchical Resampling algorithms are presented in Figure 65. This schema is similar to the one presented in the baseline approach, however, instead of using flat resampling algorithms, we use the Hierarchical Random Undersampling/Oversampling (HROS/HRUS) and the Hierarchical Synthetic Oversampling Technique (HSMOTE), presented in Chapters 8 and 9.

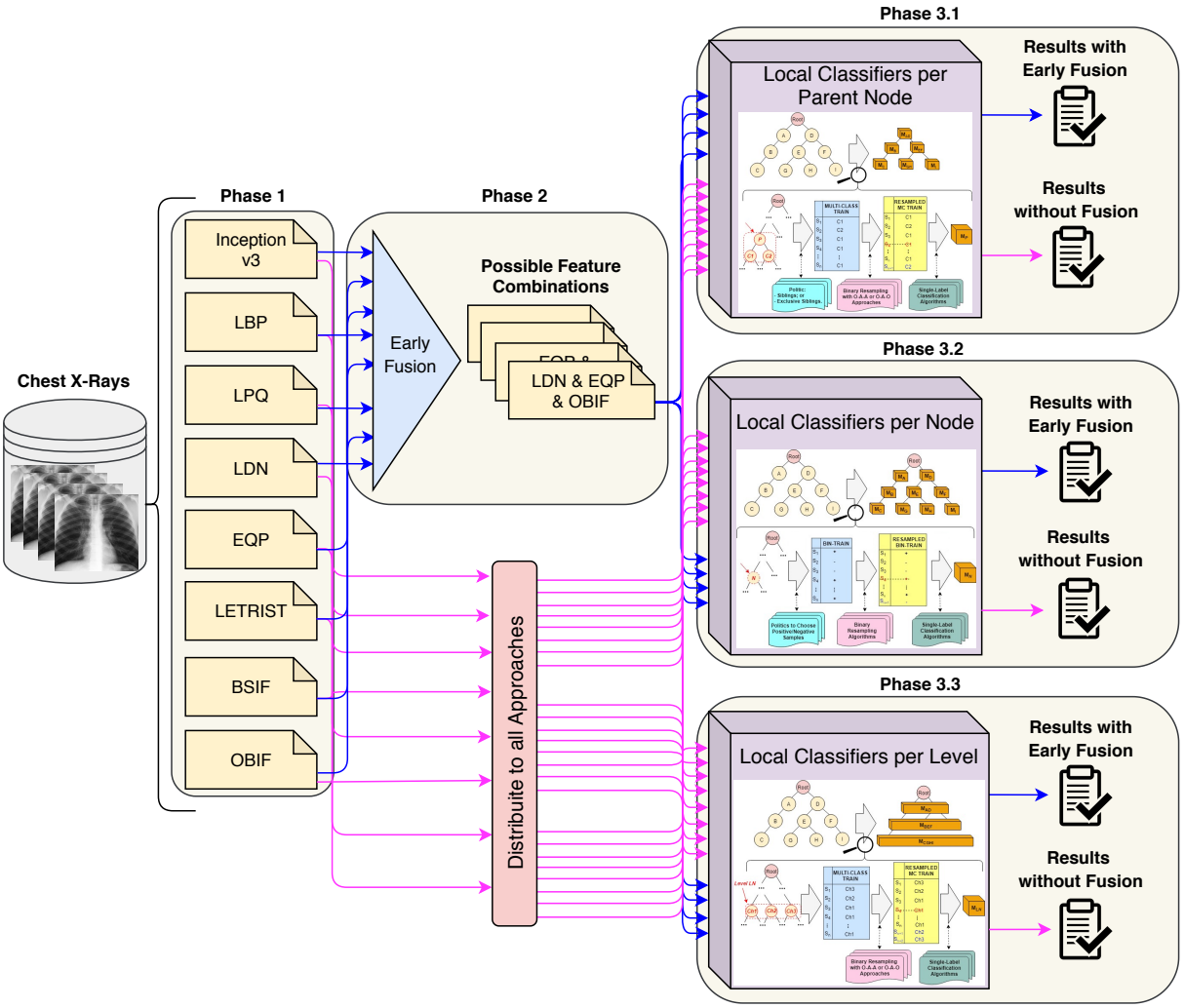


Figure 65 – The resampling schema with local classifiers for the COVID-19 identification in CXR images.

10.8 Experimental Setup

In this section we present the proposed database, algorithms, parameters and metrics used in this Chapter. It is important to observe that the database, as well as the experimental scripts used in this work are all freely available for download[17].

10.8.1 The Database

Table 66 presented the main characteristics of the proposed database, which was named RYDLS-20. As it can be noted, the database is composed of 1,144 CXR images,

[17] https://drive.google.com/open?id=1J9I-pPtPfLRGHJ42or4pKO2QASHzLkkj

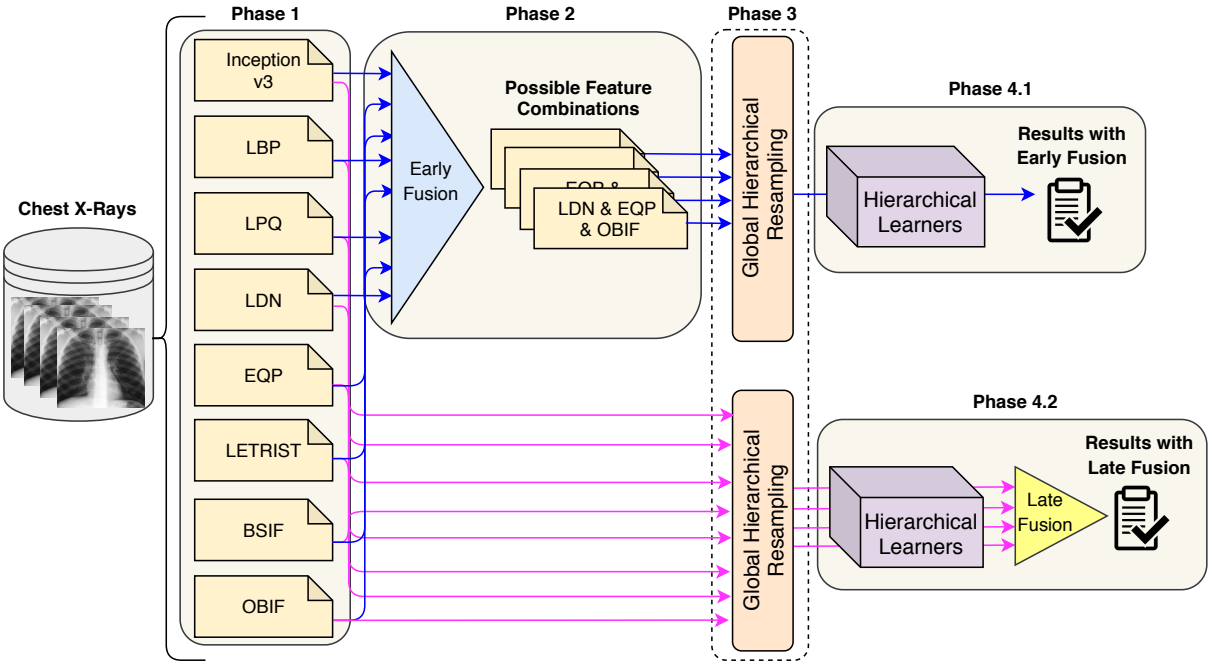


Figure 66 – The classification schema with global resampling for the COVID-19 identification in CXR images.

with a 70/30 percentage of split between train/test. Moreover, there are seven labels, which can be further hierarchically organized into fourteen label paths.

Table 66 – RYDLS-20 main characteristics.

Characteristic	*Quantity*
Samples	1,144
Train	802
Test	342
Label Nodes	14
Label Paths	7

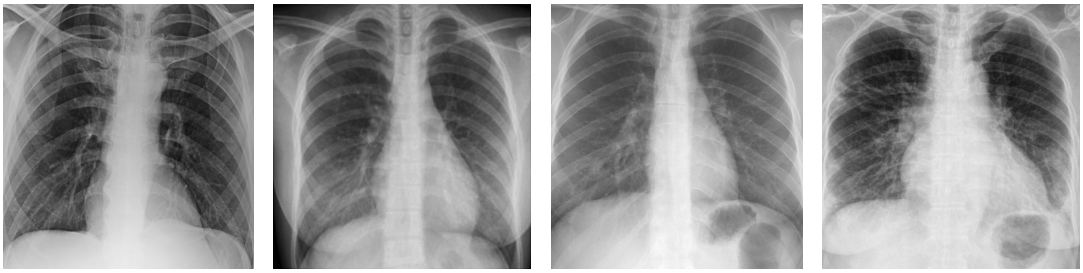
The CXR images have different sized and were obtained from three different sources:

- COVID-19, SARS, Pneumocystis and Streptococcus images were obtained in the open source GitHub repository shared by Dr. Joseph Cohen (COHEN; MORRISON; DAO, 2020).
- Varicella and MERS images were obtained from the Radiopedia encyclopedia[18].
- The Normal lung images were all obtained from NIH dataset, also known as Chest X-ray14 (WANG et al., 2017).

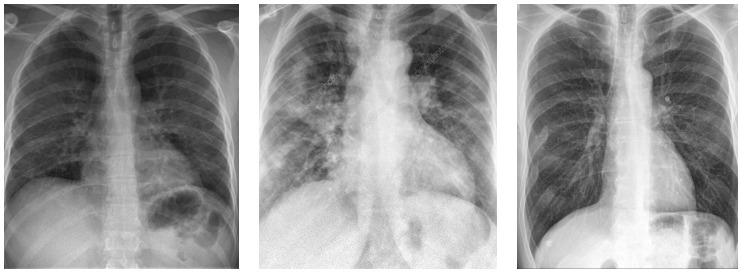
[18] https://radiopaedia.org/articles/pneumonia

Figure 67 presents image examples for each class retrieved from the RYDLS-20 database. It is worth to mentioning that we have no further information concerning the CXR images with regards to CXR machine used to take the image, as well as the origin, age and ethnicity of the people whose these images belong to.

Another important aspect concerning the database is that we have manually cut the edges of the images in order to avoid the recognition of undesirable patterns. In order to confirm the importance of this preprocessing step, we must cite the work of Maguolo & Nanni (2020), in which the authors have made a critical evaluation regarding the combination of different databases for the COVID-19 automatic detection from CXR. In their work, they have cut off the lungs from the x-ray images and have experimentally proved that a classifier can identify from which database the images came from. Thus, the authors highlight that joining different databases may add bias to the classification results, since the classifiers may be recognizing patterns from the origin database and not from the lung injuries. However, as we have manually cut the images edges in RYDLS-20, we have minimized this issue in our experiments.



(a) Normal. (b) COVID-19. (c) SARS. (d) MERS.



(e) Pneumocystis. (f) Streptococcus (g) Varicella.

Figure 67 – RYDLS-20 image samples.

Table 67 presents RYDLS-20 samples distribution for the label paths in the dataset. Though the table, it is possible to observe the major imbalanceness of the dataset between the pneumonia labels. In most cases, we have seven–nine samples in the training set and only three samples in the test dataset, which makes the learning process much more difficult than in a balanced context.

It is important to reinforce the statement that RYDLS-20 labels distribution reflect a real world scenario in which healthy cases are much more frequent (majority class),

Table 67 – RYDLS-20 samples distribution for the hierarchical scenario.

Label Path	*#Samples*	*#Train*	*#Test*
Normal	1,000	700	300
Pneumonia/Acellular/Viral/Coronavirus/COVID-19	90	63	27
Pneumonia/Acellular/Viral/Coronavirus/MERS	10	7	3
Pneumonia/Acellular/Viral/Coronavirus/SARS	11	8	3
Pneumonia/Acellular/Viral/Varicella	10	7	3
Pneumonia/Celullar/Bacterial/Streptococcus	12	9	3
Pneumonia/Celullar/Fungus/Pneumocystis	11	8	3

followed by viral pneumonia (mainly caused by COVID-19), bacterial and fungi pneumonia being the least frequent, in this order.

The experiments were conducted with a 70/30 split between training and test, which means that we are using the holdout validation technique. It may be asked why do not use a cross-validation schema, since it brings robustness to the experimental results. The answer to this question is mainly grounded in the size of the database. As shown in this section, we are dealing with a highly imbalanced database, in which half of the labels have between 10-12 samples. Thus, if we use a 10-fold or 5-fold cross-validation (the most recommended values by the research community), most labels will have only one or two examples into each fold. This division would have a high impact in the testing phase, which may lead to misleading results regarding the evaluation measures.

10.8.2 Algorithms, Parameters and Metrics

In this subsection we present the main information concerning the algorithms, parameters and metrics used in the experiments of this work.

As the Local Classifiers approaches uses flat classifiers (binary and multi-class) to distinguish among the label nodes, we have to define which algorithm will be used in this task. Thus, in order to perform this task, we have used the Random Forests (RF) algorithm. Table 68 reports the parameter used in this classification algorithm.

Table 68 – Random Forest parameter settings.

Parameters	*Value*
Number of Trees	10
Class Weight	Balance
Criterion	Gini
Splitter	Best
Min Samples Leaf	10
Min Samples Split	20
Max Leaf Nodes	None
Max Depth	10

For the hierarchical classification task we used the Clus-HMC framework. The

parameter configurations used in the Clus-HMC algorithm are presented in Table 69.

Table 69 – Clus-HMC parameter settings.

Parameter	*Value*
Type	Tree
ConvertToRules	No
FTest	[0.001, 0.005, 0.01, 0.05, 0.1, 0.125]
EnsembleMethod	RForest
Iterations	10
VotingType	Majority
EnsembleRandomDepth	No
SplitSampling	None
Heuristic	Default
PruningMethod	Default

Moreover, as we are dealing with a full depth hierarchical classification problem, the LCN approach was executed with a "Less Inclusive" policy (Section 2.3.1), since there are no samples labelled with internal nodes. Following this same reasoning, the LCPN approach was executed with the "Siblings" policy (Section 2.3.1).

Regarding the resampling methods, for both hierarchical classification scenarios in which we have used binary resampling algorithms (baseline and local classifiers), we have tested a total of 16 methods, considering oversampling, undersampling and hybrid approaches. However, we present in Results sections (10.9) only the methods that somehow improved the individual classification results: ADASYN, SMOTE, SMOTE-B1, SMOTE-B2, AllKNN, ENN, RENN, Tomek Links (TL) and SMOTE+TL. All this binary resampling method were tested with their default parameters in the imbalanced-learn framework[19].

In the global resampling scenarios, we have tested the increase/decrease rates of 5%, 10%, 15%, 20% and 25% for HROS and HRUS. Moreover, in the HSMOTE algorithm, we have tested 3 and 5 neighbors as the k value.

10.8.3 Evaluation Metric

In order to analyze the performance of the experimental results, the f-score measure was chosen. Moreover, in order to analyze the general classification performance, we have chosen the macro-avg evaluation, which makes an averaging calculation by class. This is a crucial point in the experimental setup, since evaluation measures such as accuracy may neglect the real performance of the learners for the imbalanced classes, which in our case is the main objective. Following this reasoning, the use of a metric that can really consider the imbalanceness of the different labels is necessary, and, according to Goutte & Gaussier (2005), f-score is a good alternative to deal with this issue.

[19] Available at https://imbalanced-learn.readthedocs.io/en/stable/api.html

As well known in the machine learning community, f-score is the harmonic average between precision and recall calculations. Moreover, we have used the macro-avg f-score evaluation in order to calculate the mean f-score between the classes and not the samples. It is important to observe that we have used the same f-score measure in both multi-class and hierarchical classification scenarios.

10.9 Experimental Results

As we have experimentally tested three hierarchical classification scenarios, we have sub-divided the experimental results into three subsections: (i) Baseline (Flat resampling); (ii) Local Classifiers with Resampling; (iii) Global hierarchical resampling.

We must highlight that we present the results considering two perspectives:

- The general macro-avg f-score for the evaluated scenarios, i.e., the average f-score for all classes in the classification task; and
- The f-score obtained specifically for the COVID-19 label, given that this is our main interest here.

Due to the large number of experimental results, in this Chapter we present only the best results achieved in each prediction schema. A detailed version of the results is shown in the Appendix B.

10.9.1 Baseline Results

Table 70 presents the best macro-avg f-score results for each prediction schema in the baseline approach. It can be noted that the best macro-avg f-score was achieved in the early fusion prediction schema without the use of resampling algorithms. Moreover, it is possible to observe that SMOTE resampling technique (and its variations) appears in all the other best classification scenarios.

Table 70 – Best macro-avg results for each prediction schema in the Baseline approach.

Prediction Schema	***Feature***	***Resampling***	***F-Score***
Individual	LPQ	SMOTE-B1	0.4428
Early Fusion	BSIF & EQP & OBIF	-	0.4996
Late Fusion (Top-5)	LETRIST and LPQ	SMOTE-B1/B2	0.4428
Late Fusion (Top-Features)	BSIF, EQP and LPQ	SMOTE-B1/B2	0.4428

Table 71 presents the best f-score results for the COVID-19 identification in each prediction schema in the baseline approach. It can be noticed that the best result was also

achieved in the early fusion prediction schema. Besides, SMOTE resampling technique appears in most of the classification schemas.

Table 71 – Best COVID-19 results for each prediction schema in the Baseline approach.

Prediction Schema	***Feature***	***Resampling***	***F-Score***
Individual	OBIF	ENN	0.7463
Early Fusion	LETRIST & OBIF	SMOTE-B1	0.7931
Late Fusion (Top-5)	BSIF, EQP and OBIF	ADASYN and SMOTE-B1/B2	0.7547
Late Fusion (Top-Features)	BSIF, EQP and OBIF	SMOTE-B1/B2	0.7869

Figure 68 presents a chart of the individual f-score results per label for the best macro-avg case scenario, which was achieved in the early fusion schema with BSIF, EQP and OBIF features without the application of resampling algorithms. We may observe that the classifier achieved very different performances for each type of pneumonia, including zero f-scores for the "Varicella" and "Streptococcus" labels, However, we may note that it reached interesting and close performances for the Pneumonia caused by the different Coronavirus pathogens, i.e., COVID-19, SARS and MERS.

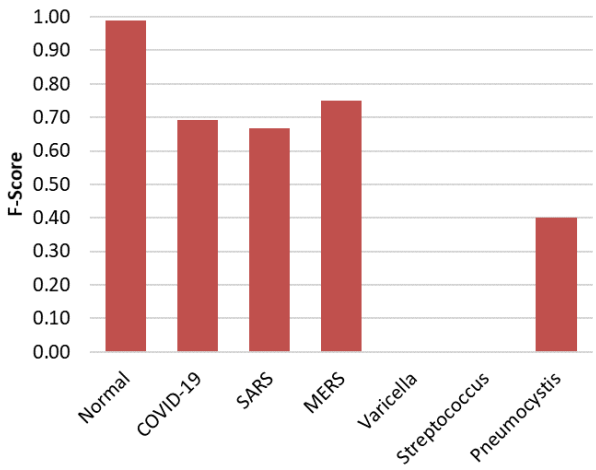


Figure 68 – F-scores per label in the best macro-avg scenario using the baseline approach.

Figure 69 presents a chart of the individual f-score results per label for the best COVID-19 identification scenario, which was also achieved in the early fusion schema, but with LETRIST and OBIF features and the SMOTE-B1 resampling algorithm. It may be noticed that, as well as in the macro-avg scenario, the classifier also achieved zero f-scores for the "Varicella" and "Streptococcus" labels. Furthermore, as it achieve a better performance for the COVID-19, the classifier reduced the prediction results for the SARS, MERS and Pneumocystis labels.

Figure 70 shows the confusion matrix for the same macro-avg best case scenario in the baseline schema as presented in Figure 68 (Early Fusion of BSIF, EQP and OBIF).

Figure 71 shows the confusion matrix for the same best case scenario for the COVID-19 identification in the baseline schema as presented in Figure 69 (Early Fusion of LETRIST and OBIF with SMOTE-B1).

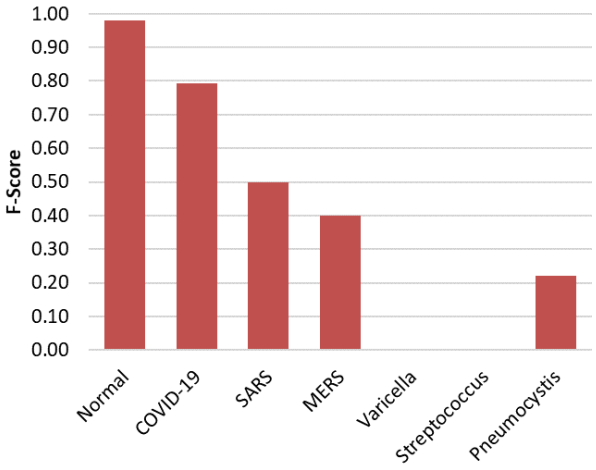


Figure 69 – F-scores per label in the best COVID-19 identification scenario using the baseline approach.

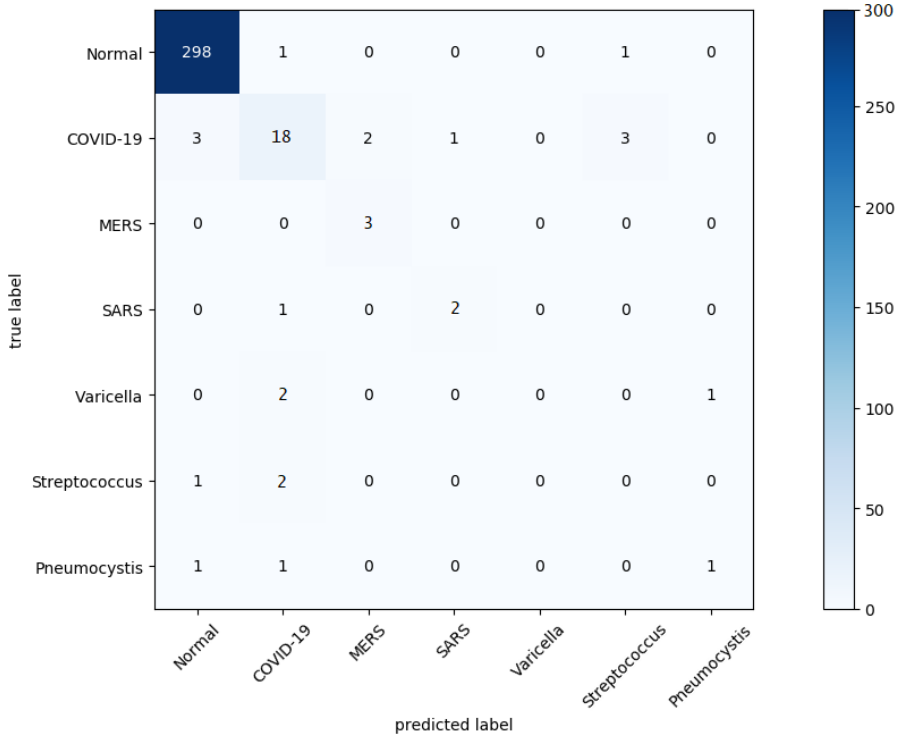


Figure 70 – Confusion Matrix in the best macro-avg scenario using the baseline approach.

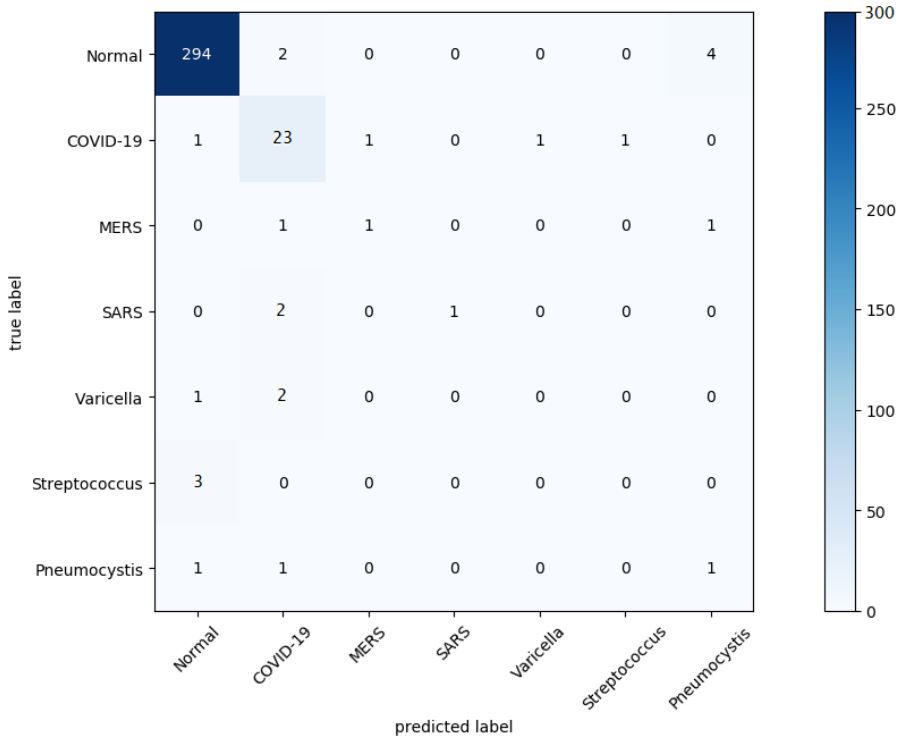


Figure 71 – Confusion Matrix in the best COVID-19 identification scenario using the baseline approach.

Table 72 shows the IRLP measure results (Section 6.1) for the baseline datasets with different feature sets. It can be noted that the original dataset has an IRLP of 66.31 and five of the resampling algorithms achieved the best imbalance ratio value (52.27): SMOTE, SMOTE-B1, SMOTE-B2, ADASYN and SMOTE+TL.

Table 72 – IRLP measure results for the baseline experiments.

	Original	SMOTE	SMOTE B1	SMOTE B2	ADASYN	ENN	RENN	AllKNN	TL	SMOTE + TL
BSIF	66.41	52.27	52.27	52.27	52.34	64.14	63.67	64.05	66.22	52.27
EQP	66.41	52.27	52.27	52.27	52.27	64.99	64.90	64.99	66.32	52.27
INCEPTION	66.41	52.27	66.41	66.41	52.27	62.15	61.30	61.68	66.03	52.27
LBP	66.41	52.27	52.27	52.27	52.27	64.99	64.71	64.71	66.41	52.27
LDN	66.41	52.27	52.27	52.27	52.27	64.80	64.80	64.80	66.13	52.27
LETRIST	66.41	52.27	52.27	52.27	52.34	63.95	63.19	63.38	65.94	52.27
LPQ	66.41	52.27	52.27	52.27	52.27	64.52	64.24	64.24	66.22	52.27
OBIF	66.41	52.27	52.27	52.27	52.27	63.19	62.25	62.34	66.13	52.27

10.9.2 Local Classifiers with Resampling

Table 73 presents the best macro-avg f-score results for each prediction schema in the local classifiers with resampling approaches. The best macro-avg f-score was achieved with an early fusion technique over the LDN, LETRIST and LPQ features after applying the SMOTE-B2 resampling technique. Besides, LCPN was used to achieve the best results in both classification schemas. It is important to observe that, with a f-score of 0.6642, the

classification scenario obtained the best macro-avg rate for the classification task among all experiments from this Chapter.

Table 73 – Best macro-avg results for each prediction schema in the local classifiers with Resampling approaches.

Prediction Schema	***Feature***	***Resampling***	***Approach***	***F-Score***
Individual	LETRIST	SMOTE-B2	LCPN	0.5332
Early Fusion	LDN & LETRIST & LPQ	SMOTE-B2	LCPN	0.6643

Table 74 presents the best COVID-19 identification f-scores for each prediction schema in the local classifiers with resampling approaches. As well as in the macro-avg scenario, the best f-score was also achieved with the early fusion schema classified with an LCPN approach, however using the LETRIST, EQP and OBIF features with the SMOTE-B2 algorithm. It is worth to mention that, with a f-score of 0.9333, this classification scenario obtained the best rate for the COVID-19 identification among all experiments from this Chapter.

Table 74 – Best COVID-19 results for each prediction schema in the local classifiers with Resampling approaches.

Prediction Schema	***Feature***	***Resampling***	***Approach***	***F-Score***
Individual	LETRIST	SMOTE	LCPN	0.8936
Early Fusion	LETRIST & EQP & OBIF	SMOTE-B2	LCPN	0.9333

Figure 72 shows a chart of the individual f-score results per label for the best macro-avg case scenario, which was obtained in the early fusion schema with LDN, LETRIST and LPQ features after applying the SMOTE-B2 resampling technique and LCPN. Even though "Pneumocystis" label achieved a zero f-score, it is interesting to observe that MERS reached a perfect score of 1.

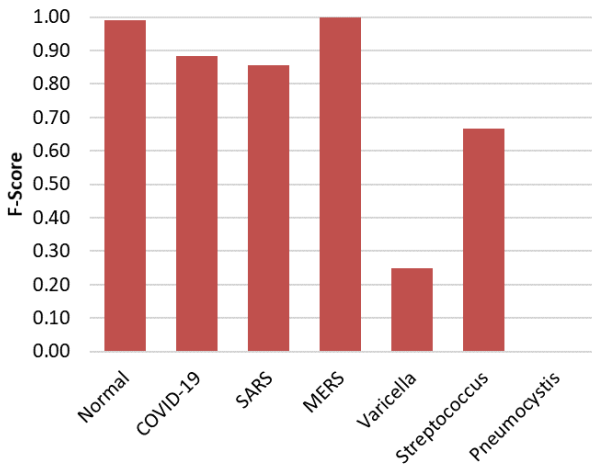


Figure 72 – F-scores per label in the best macro-avg scenario using the local classifiers with Resampling approaches.

Figure 73 presents a chart of the individual f-score results per label for the best COVID-19 identification case scenario, which was obtained in the early fusion schema with

LETRIST, EQP and OBIF features with the SMOTE-B2 algorithm and LCPN approach. Moreover, such as in the previous scenario, "Pneumocystis" achieved a zero f-score.

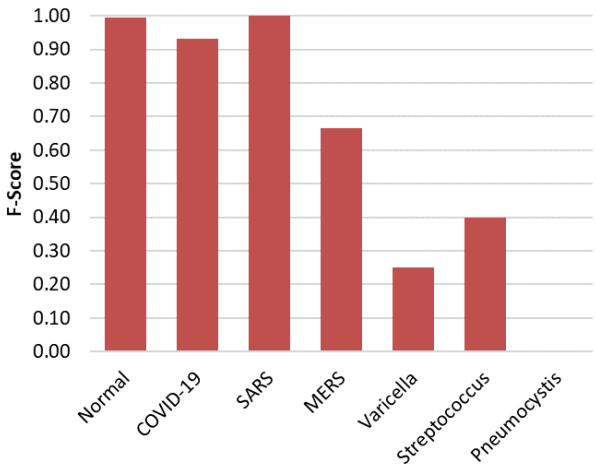


Figure 73 – F-scores per label in the best COVID-19 identification scenario using the local classifiers with Resampling approaches.

Figure 74 shows the confusion matrix for the same best macro-avg case scenario in the local classifiers with resampling schema as presented in Figure 72 (Early Fusion of LDN, LETRIST and LPQ with SMOTE-B2 and LCPN approach).

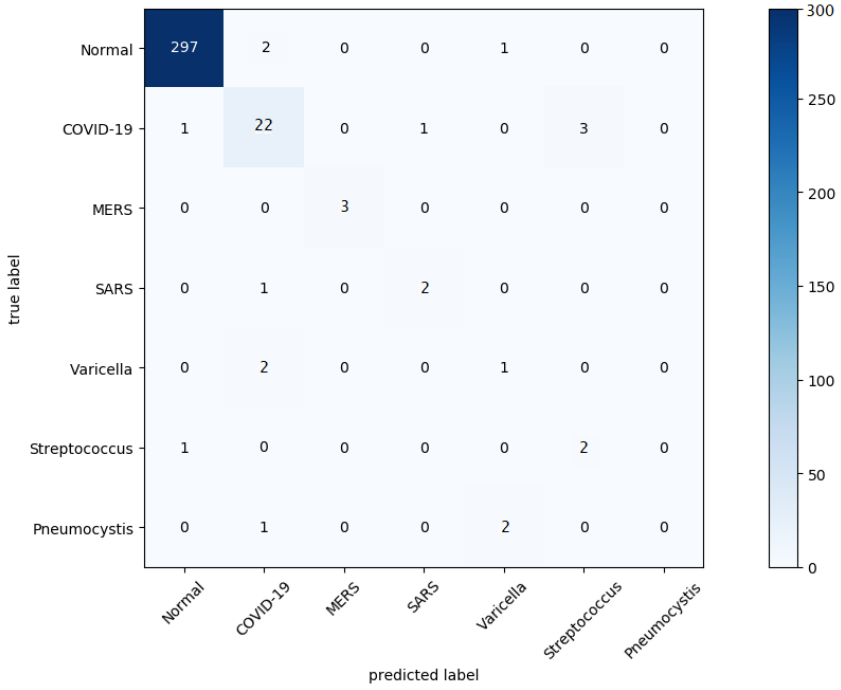


Figure 74 – Confusion Matrix in the best macro-avg scenario using the local classifiers with Resampling approaches.

Figure 75 shows the confusion matrix for the same best case scenario for the COVID-19 identification in the local classifiers with resampling schema as presented in Figure 73 (Early Fusion of LETRIST, EQP and OBIF with SMOTE-B2 using the LCPN approach).

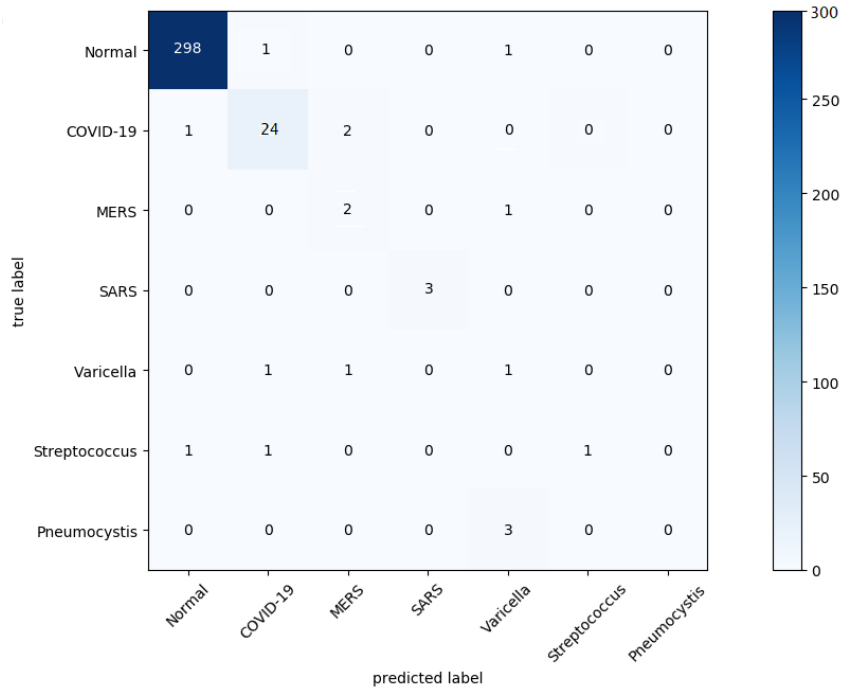


Figure 75 – Confusion Matrix in the best COVID-19 identification scenario using the local classifiers with Resampling approaches.

Table 75 presents the IR_{LCN} measure results for the local classifiers per node approach (7.1.1) in the different feature sets. It can be observed that the original dataset has an IR_{LCN} of 6.60 and the best imbalance ratio value (3.64) was obtained by the LETRIST feature set with the SMOTE+TL resampling method.

Table 75 – IR_{LCN} measure results for the local classifiers per node experiments.

	Original	SMOTE	SMOTE B1	SMOTE B2	ADASYN	ENN	RENN	AllKNN	TL	SMOTE + TL
BSIF	6.60	4.46	4.46	3.99	3.94	4.39	4.04	4.01	4.37	4.01
EQP	6.60	4.91	4.99	5.27	3.74	5.07	5.38	5.17	4.19	4.05
INCEPTION	6.60	4.86	4.02	3.99	5.47	4.72	5.34	4.93	5.07	5.09
LBP	6.60	3.89	4.70	4.19	4.37	5.11	5.47	5.37	5.60	5.46
LDN	6.60	5.51	5.03	5.36	5.49	4.03	5.01	4.41	4.58	5.33
LETRIST	6.60	4.01	3.69	4.62	3.80	4.18	4.14	4.00	3.84	3.64
LPQ	6.60	3.77	3.76	5.22	3.67	5.26	5.28	4.53	3.81	5.38
OBIF	6.60	4.49	5.06	4.17	4.72	5.24	4.40	4.52	4.35	4.98

Table 76 shows the IR_{LCPN} measure results for the local classifiers per parent node (7.1.2) with the different feature sets. We can be note that the original dataset has an IR_{LCPN} of 6.36 and the best imbalance ratio value (3.41) was obtained by the LBP feature with the RENN resampling method.

Table 77 presents the IR_{LCL} measure results for the local classifiers per level (7.1.3) with different feature sets. We can note that the original dataset has an IR_{LCL} of 35.96 and the best imbalance ratio value (27.39) was obtained by the BSIF feature with the ADASYN resampling method.

Table 76 – IR_{LCPN} measure results for the local classifiers per parent node experiments.

	Original	SMOTE	SMOTE B1	SMOTE B2	ADASYN	ENN	RENN	AllKNN	TL	SMOTE + TL
BSIF	6.36	4.63	4.27	4.46	3.76	4.99	4.57	5.14	4.62	3.58
EQP	6.36	4.21	4.80	4.74	4.58	3.82	3.49	5.17	4.92	3.90
INCEPTION	6.36	3.89	4.61	4.43	3.80	3.76	4.40	4.38	5.22	4.11
LBP	6.36	4.03	5.21	4.87	4.59	4.01	3.41	4.16	4.31	5.20
LDN	6.36	4.79	3.50	4.88	3.62	4.18	5.24	3.66	4.96	4.90
LETRIST	6.36	3.96	3.96	3.96	5.26	5.19	3.84	4.16	4.44	4.15
LPQ	6.36	3.96	3.53	4.51	3.59	4.50	4.14	4.54	4.78	3.87
OBIF	6.36	4.26	3.40	4.04	4.30	4.44	4.31	3.42	4.67	4.44

Table 77 – IR_{LCL} measure results for the local classifiers per parent node experiments.

	Original	SMOTE	SMOTE B1	SMOTE B2	ADASYN	ENN	RENN	AllKNN	TL	SMOTE + TL
BSIF	35.96	30.55	29.40	28.13	27.39	28.84	27.24	29.78	30.14	29.77
EQP	35.96	31.08	29.09	29.82	29.83	31.02	29.92	31.14	31.67	32.88
INCEPTION	35.96	33.19	33.17	31.38	29.39	30.18	30.18	30.14	29.22	28.44
LBP	35.96	30.25	29.24	28.14	29.14	32.38	31.37	33.13	28.73	27.78
LDN	35.96	31.61	32.90	29.84	29.15	32.90	30.52	29.91	29.82	30.37
LETRIST	35.96	30.69	31.86	32.89	31.36	30.91	32.27	31.00	29.14	32.29
LPQ	35.96	29.32	30.56	30.44	29.50	31.03	31.02	31.87	29.67	30.68
OBIF	35.96	30.60	31.01	29.10	31.50	32.23	32.73	32.24	32.23	31.63

10.9.3 Global Hierarchical Resampling

Table 78 shows the best macro-avg f-score results for each prediction schema in the global hierarchical resampling approach. It is important to mention that, in all tables of this subsection, the values after the resampling algorithms represent the configuration used for the resampling, i.e., the resize rate (for the random algorithms) and the number of neighbors (for the HSMOTE technique). As in all the previous schemas, the best macro-avg f-score was obtained with an early fusion technique. This result was achieved with the combination of EQP and LPQ features after applying the HROS resampling technique with an increase rate of 5%.

Table 78 – Best macro-avg results for each prediction schema in the global hierarchical resampling approach.

Prediction Schema	***Feature***	***Resampling***	***F-Score***
Individual	LDN	HROS-15	0.4662
Early Fusion	EQP & LPQ	HROS-5	0.5524
Late Fusion (Top-5)	BSIF and LETRIST	HRUS-5 and HSMOTE-3	0.4272
Late Fusion (Top-Features)	BSIF, EQP and LDN	HRUS-5, HROS-20 and HROS-10	0.4548

Table 79 presents the best COVID-19 identification f-score for each prediction schema in the global hierarchical resampling approach. The best score was obtained in exactly the same scenario as the macro-avg f-score with an early fusion of EQP and LPQ features after applying the HROS resampling technique with an increase rate of 5%.

Table 79 – Best COVID-19 identification results for each prediction schema in the global hierarchical resampling approach.

Prediction Schema	***Feature***	***Resampling***	***F-Score***
Individual	OBIF	HROS-5	0.7692
Early Fusion	EQP & LPQ	HROS-5	0.8235
Late Fusion (Top-5)	EQP, LBP, and OBIF	HRUS-10 and HROS-5	0.7692
Late Fusion (Top-Features)	BSIF, EQP and LBP	HRUS-5, HROS-20 and HROS-5	0.7692

Figure 76 shows a chart of the f-score results per label for the best macro-avg and COVID-19 identification scenarios, which was obtained in the early fusion schema with EQP and LPQ features and HROS-5 algorithm. It can be noted that, in this classification scenario, the "SARS" label achieved the perfect f-score of 1, while the "Streptococcus" label reached the a zero f-score.

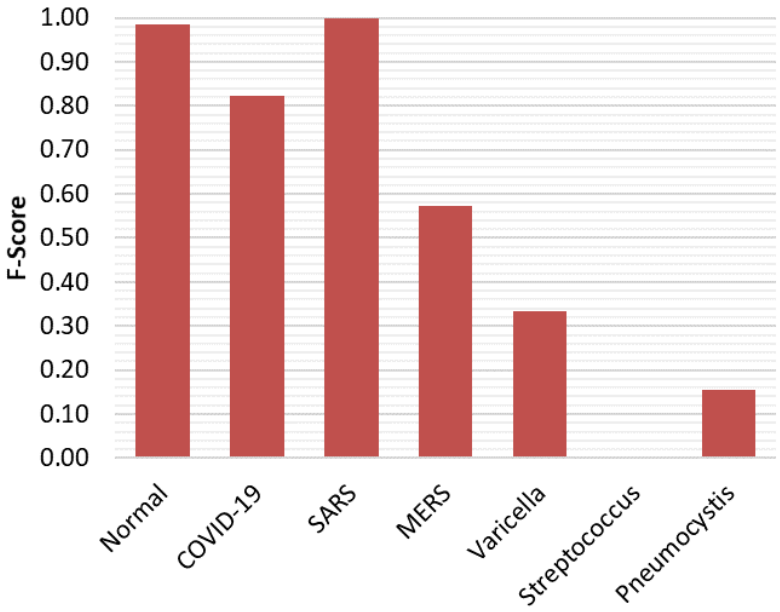


Figure 76 – F-scores per label in the best macro-avg and COVID-19 identification scenarios using the global resampling approach.

Figure 77 shows the confusion matrix for the best case scenario (both macro-avg and COVID-19 identification) in the global resampling approach as presented in Figure 76 (Early Fusion of EQP and LPQ features after applying the HROS-5).

Table 80 presents the IRLP measure results for the global hierarchical resampling with the different feature sets. We can observe that the original dataset has an IRLP of 66.41 and the best imbalance ratio value (11.98) was achieved by the HROS resampling method with 25% of increasing rate.

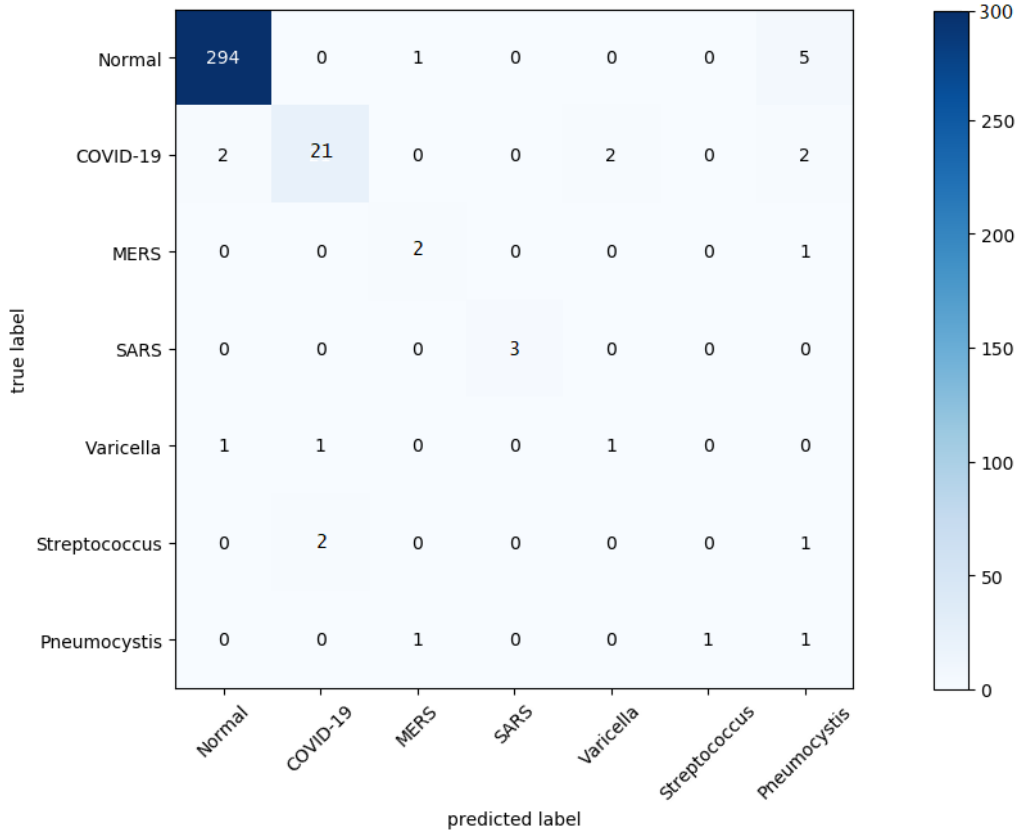


Figure 77 – Confusion Matrix in the best macro-avg and COVID-19 identification scenarios using the global resampling approach.

Table 80 – IRLP measure results for the global hierarchical resampling experiments.

	Original	HROS					HRUS					HSMOTE
		5%	10%	15%	20%	25%	5%	10%	15%	20%	25%	
BSIF	66.41	31.55	21.91	16.98	13.99	11.98	64.21	60.37	50.8	45.6	41.63	34.07
EQP	66.41	31.55	21.91	16.98	13.99	11.98	64.21	60.37	50.8	45.6	41.63	34.15
INCEPTION	66.41	31.55	21.91	16.98	13.99	11.98	64.21	60.37	50.8	45.6	41.63	32.34
LBP	66.41	31.55	21.91	16.98	13.99	11.98	64.21	60.37	50.8	45.6	41.63	33.45
LDN	66.41	31.55	21.91	16.98	13.99	11.98	64.21	60.37	50.8	45.6	41.63	31.25
LETRIST	66.41	31.55	21.91	16.98	13.99	11.98	64.21	60.37	50.8	45.6	41.63	32.45
LPQ	66.41	31.55	21.91	16.98	13.99	11.98	64.21	60.37	50.8	45.6	41.63	32.45
OBIF	66.41	31.55	21.91	16.98	13.99	11.98	64.21	60.37	50.8	45.6	41.63	31.35

10.10 Discussions

Aiming to evaluate the obtained results from different points of view, we guide our discussion in this research answering the following questions:

- Which feature representation provided the best results?
- Did the resampling approaches proposed in this Doctoral Research improve the classification results?
- Which resampling algorithms improved the results the most in each classification schema?

- Did the fusion strategies contribute to improve the results?
- Which kinds of labels are easier/harder to predict?
- Which labels were confused in the best case scenario for each classification schema?
- What may be happening in the misclassified cases?

In the following subsections, we answer each question taking into account the statistical significance, when it is suitable.

Which feature representation provided the best results?

To answer this question, i.e., define the best feature representation, we have used the statistical protocol proposed in Charte et al. (2015b) (already used in some discussions of the previous Chapters). As we have twelve different measurement contexts, i.e., the general macro-avg f-scores results and the COVID-19 identification f-scores for the baseline, local classifiers and global approaches, we have performed this protocol twelve times, one in each classification context and measurement. The test results in relation to the ranking of the features for the macro-avg f-Score and the COVID-19 identification f-scores are shown in Table 81. It is worth mentioning that, since this test is composed of rankings, which ranges from the first to the last, the lower the ranking score is, the better is the performance.

We can observe that, LETRIST was the best ranked feature set in the baseline classification schema for both scenarios: COVID-19 identification and macro-avg f-score. In the local classifiers scenario, the best ranked feature set for the COVID-19 identification was tied between LETRIST and OBIF, while for the macro-avg f-score, the best ranked feature set was LETRIST. Finally, in the global resampling schema, the best feature set for the COVID-19 identification was OBIF and for the macro-avg f-score was LETRIST. Moreoever, analyzing the overall average ranking, we have, by far, LETRIST as the best feature set considering all classification schemas and evaluation scenarios.

Table 81 – Ranking of the results per feature set in all classification scenarios.

	Baseline		Local Classifiers		Global Resampling		*Overall*
Resampling	COVID-19	Macro-Avg	COVID-19	Macro-Avg	COVID-19	Macro-Avg	*Avg. Rank.*
BSIF	3.33	3.67	6.50	7.50	6.33	3.33	*5.25*
INCEPTION	7.33	7.33	4.50	7.00	7.00	6.00	*6.54*
LBP	6.33	5.33	5.50	4.50	5.33	4.67	*5.42*
LDN	5.33	3.33	6.50	2.00	2.67	4.33	*4.29*
LETRIST	1.33	2.67	3.50	1.00	4.33	2.33	***2.13***
EQP	4.67	3.67	5.00	6.50	2.33	3.67	*4.96*
LPQ	3.67	3.33	6.00	2.50	3.33	2.67	*3.88*
OBIF	1.67	4.33	3.50	3.50	1.67	6.33	*3.25*

Did the resampling approaches proposed in this Doctoral Research improve the classification results?

The first important observation is that, as stated in the Results section (10.9.2), the best nominal macro-avg (0.66) and COVID-19 identification (0.93) results were obtained using one of the proposed resampling approaches, which was the local classifiers with resampling using the LCPN classification schema with the SMOTE-B2 resampling method.

In order to make a deeper analysis regarding the possible improvement of the classification results by each particular proposed approach, we performed the Wilcoxon Statistical Test stating as hypothesis that the classification results improve with the resampling algorithms in each proposed approach (Baseline, Local and Global). In order to calculate the test scores, we have grouped all the results in each classification approach per resampling method. Thus, Tables 82, 83 and 84 presents the p-values scores for the baseline, local classifiers and global hierarchical resampling approaches, respectively.

Analyzing Table 82 and considering a threshold of 0.1, we can observe that only the resampling algorithms that involves the use of the SMOTE method, i.e., SMOTE, SMOTE+TL, SMOTE-B1 and SMOTE-B2, could improve the classification results in baseline resampling schema, since their p-values are below the threshold.

Table 82 – Wilcoxon test results for the baseline approach.

Resampling	*p-value*
ADASYN	0.7123
AllKNN	0.3897
ENN	0.5000
RENN	0.6103
SMOTE	0.0250
SMOTE+TL	0.0344
SMOTE-B1	0.0059
SMOTE-B2	0.0059
TL	0.6425

Furthermore, analyzing Table 83 and also considering a threshold of 0.1, we may observe that, besides the SMOTE related methods, ENN and TL methods also statistically improved the classification results. It is worth to mention that we are considering the three local approaches: LCN, LCPN and LCL.

Finally, looking at Table 84 and still considering the threshold of 0.1, we may observe that HROS method could statistically improve the classification results using three different resize rates (10, 15 and 20), while HRUS only improved the results in two of the resize rates (5 and 10). Moreover, HSMOTE method could improve the results with two different number of neighbors (3 and 5).

Table 83 – Wilcoxon test results for the local classifiers approach.

Resampling	*p-value*
ADASYN	0.1760
AllKNN	0.6792
ENN	0.0604
RENN	0.3208
SMOTE	0.0247
SMOTE+TL	0.0604
SMOTE-B1	0.0002
SMOTE-B2	0.0193
TL	0.0991

Table 84 – Wilcoxon test results for the global hierarchical resampling approach. The configuration values for the random algorithms represent the resize rate in %, while the values for the HSMOTE represent the number of neighbors used in the sample section stage.

Resampling	Parameter Configuration	*p-value*
HROS	5	0.1038
	10	0.0059
	15	0.0250
	20	0.0125
	25	0.1038
HRUS	5	0.0086
	10	0.0059
	15	0.9914
	20	0.9941
	25	0.9941
HSMOTE	3	0.0464
	5	0.0125

Which resampling algorithms improved the results the most in each classification schema?

To answer this question, we have used the same statistical protocol from the first questions. However, in this case, we have ranked the resampling methods for each classification schema, considering the resampling methods used in each case scenario.

Table 85 presents the ranking results for the statistical test in the baseline classification schema. We may observe that for both measuring scenarios, i.e., COVID-19 identification and macro-avg f-score, SMOTE-B1 resampling method achieved by far the best average ranking (overall ranking of 1.33), while ADASYN and AllKNN were the worst resampling algorithms with overall average ranking of 7.50 and 7.17, respectively.

Table 86 shows the ranking results for the statistical test in the local classifiers schema. For the COVID-19 identification measurement the best ranked resampling algorithm was tied between SMOTE and TL (3.5), while for the macro-avg measurement scenario the best ranked resampling method was SMOTE-B2. Moreover, SMOTE-B2

Table 85 – Ranking of results per resampling method in the baseline classification schema.

Resampling	COVID-19	Macro-Avg	*Overall Avg. Rank*
ADASYN	6.67	8.33	*7.50*
AllKNN	6.67	7.67	*7.17*
ENN	3.67	5.67	*4.67*
RENN	6.33	6.67	*6.50*
SMOTE	4.33	2.67	*3.50*
SMOTE-B1	1.67	1.00	***1.33***
SMOTE-B2	5.33	3.00	*4.17*
TL	5.00	4.33	*4.67*
SMOTE+TL	5.00	5.67	*5.33*

obtained the best overall average ranking (3.50), while RENN and TL were tied as the worst resampling methods with an overall average rankings of 6.25.

Table 86 – Ranking of results per resampling method in the local classifiers schema.

Resampling	COVID-19	Macro-Avg	*Overall Avg. Rank*
ADASYN	6.00	2.50	*4.25*
AllKNN	4.50	6.00	*5.25*
ENN	5.50	5.50	*5.50*
RENN	6.50	6.00	*6.25*
SMOTE	3.50	4.00	*3.75*
SMOTE-B1	5.00	5.00	*5.00*
SMOTE-B2	6.00	1.00	***3.50***
TL	3.50	9.00	*6.25*
SMOTE+TL	4.50	6.00	*5.25*

Table 87 presents the ranking results for the statistical test in the global resampling schema. We may note that in both measurement scenarios, HROS algorithm obtained the best average ranking.

Table 87 – Ranking of results per resampling method in the global resampling schema.

Resampling	COVID-19	Macro-Avg	*Overall Avg. Rank*
HROS	1.67	1.00	***1.33***
HRUS	2.00	2.00	*2.00*
HSMOTE	2.00	2.00	*2.00*

Did the fusion strategies contribute to improve the results?

Yes, the early fusion strategy was particularly important to provide better results for the COVID-19 identification and macro-avg f-score results in all classification approaches. Besides, the early fusion technique also provided the best f-scores among all experiments.

However, it is worth to mention that, in general, the late fusion technique was not able to increase the classification results in all approaches.

Which kinds of labels are easier/harder to predict?

The answer to this question is grounded in the charts of the individual f-scores per labels presented in the Results section (10.9). Analyzing all classification and measurement scenarios, the fist easily observation is that "Normal" label has the remarkably best f-scores results compared to the pneumonia labels. In this case, this performance leans on the fact that "Normal" label is by far the most frequent in the database.

Considering the "Coronavirus" labels, i.e., "COVID-19", "SARS" and "MERS", their individual f-score results have a large variety among the different classification scenarios. In the baseline experiments, the 'MERS" label achieved slightly better results in the macro-avg measurement scenario, while in the COVID-19 identification scenario it achieved the worst result. In the local classifiers experiments, "MERS" and "SARS" achieved a score of 1 in the macro-avg and the COVID-19 measurement scenarios, respectively. In the global resampling scenario, "SARS" also achieved a f-score of 1.

Furthermore, another important observation is concerning the "Varicella", "Streptrococcus" and "Pneumocystis" labels, which were the hardest ones to predict in general. In the baseline experiments "Varicella" and "Streptrococcus" labels achieved zero f-scores in both measurement scenarios. In the local classifiers experiments "Pneumocystis" label reached a zero f-score, while in the global resampling experiments "Streptococcus" got zero f-scores.

Which labels were confused in the best case scenario for each classification schema?

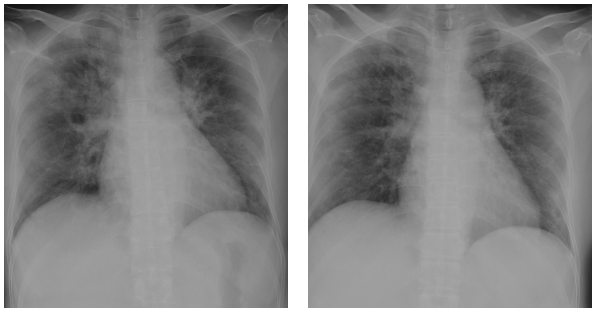
The answer to this question is made by analyzing the confusion matrices in each classification approach. Thus, looking at the confusion matrix for the best case in the baseline experiments (Figures 70 and 71), we may observe that in both measurement scenarios, a variety of "COVID-19" samples were incorrectly predicted as all the other labels. Among these, it is important to observe that "Varicella", "Streptococcus" and "Pneumocystis" were all confused with "Normal" and "COVID-19" labels.

In relation to the confusion matrices for the local classifiers approach experiments (Figures 74 and 75), we may observe that in the COVID-19 identification measurement scenario, the "COVID-19" samples were only confused with "Normal" and "MERS", while only two "Normal" samples were incorrectly labeled with "COVID-19" and "Streptococcus". Moreover, in the macro-avg scenario, there were some "COVID-19" samples misclassified as a variety of other labels.

Finally, in the confusion matrix for the global hierarchical resampling experiments (Figure 77), we may note that few "COVID-19" samples were misclassified as "Normal" and "MERS", while all "Pneumocystis" samples were incorrectly classified as "Varicella".

What may be happening in the misclassified cases?

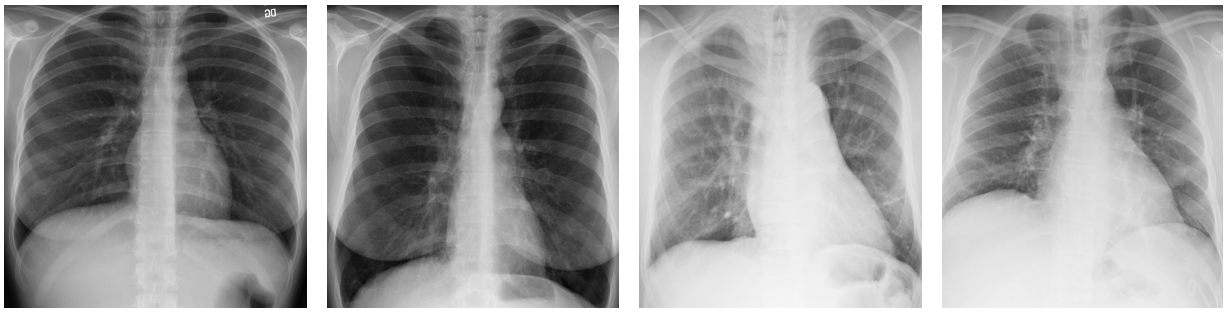
This is probably the most tricky question in this discussion so far, since CXR images are not always the medical standard to diagnose pneumonia pathogens. Figure 78 presents two examples of samples with the "COVID-19" label that were misclassified as "Normal" in the best case scenario of the multi-class classification approach. In these examples, it is really difficult to identify what could make the learner recognize "Normal" patterns instead of "COVID-19". However, in the following we present other examples that can bring some thoughts into the light of this issue.



(a) Example 1. (b) Example 2.

Figure 78 – Examples of samples with "COVID-19" label that were predicted as "Normal".

In order to give a direction regarding what may be happening in some of the misclassified CXR cases, we present in Figure 79 four examples of CXR images from "normal" lungs, which were also extracted from RYDLS-20.



(a) Normal Example 1. (b) Normal Example 2. (c) Normal Example 3. (d) Normal Example 4.

Figure 79 – Different examples of CXR with "normal" lungs.

When we think about a "normal" CXR, i.e., a CXR of a person without pneumonia, we may immediately think in a CXR similar to Figures 79(a) and 79(b), in which the

image is pretty clear and we may see that there are no white spots whatsoever. However, in practice, due to variate factors such as the patient lung characteristics, type of x-ray scan machine, and even due to the protocol followed by the professional radiologist which operates the scan machine, there might be visual variations between the x-rays captured in this different circumstances that still belongs to the same class. Considering this context, it is common to observe in the database samples from people without pneumonia but with CXR images similar to the ones presented in Figures 79(c) and 79(d). Thus, given as example the four samples from Figure 79, which belong to the same class but present such different characteristics, it is comprehensive that an artificial intelligence system, which is based on similarity patterns, can make recognition mistakes for these images.

10.11 Final Considerations

As the COVID-19 pandemic spreads around the world, the number of cases keeps growing exponentially. Finding a method that can help in the diagnosis of this disease in people, using a cheap and fast method, is fundamental to avoid overwhelming the healthcare system. In this context, the use of machine learning techniques to identify the pneumonia disease in CXR images has been proposed in the literature and may help in this diagnosis. However, when we are dealing with images taken from patients stricken of pneumonia caused by different types of pathogens and we are trying to predict a cause of a specific pneumonia (in this case, COVID-19), the problem turns into an even more challenging task.

In a real world context, we have many more people unaffected by pneumonia than affected. In addition, there is a natural imbalance between the number of people stricken by pneumonia caused by different pathogens. And due to the COVID-19 outbreak, it is hard to estimate the precise imbalance between these numbers. Considering this realistic context, in this Chapter we have presented a case study of the resampling schemas proposed in this Doctoral research aiming to classify pneumonia caused by different pathogens in CXR images, and also to identify COVID-19 among them. In order to apply hierarchical classification into this application domain, we have developed a tree structured taxonomy in which the pathogens that causes pneumonia are hierarchically organized according to their biological characteristics.

In this Chapter we have tested three different resampling schemas proposed in this Doctoral Research: (i) The hierarchical classification with flat resampling (baseline); (ii) The local classifiers with resampling (iii) The hierarchical global resampling. All these schemas were experimentally tested with eight different feature sets, which were extracted from the x-ray images and tested as individual and combined in an early fusion designs. Besides, the prediction outputs are also tested individually and in a late fusion design.

We have also proposed the RYDLS-20 database. The database is composed of 1,144 CXR images from seven classes: Normal lungs and lungs affected by COVID-19, MERS, SARS, Varicella, Streptococcus and Pneumocystis. The dataset is highly imbalanced, with 1,000 images being from people unaffected by pneumonia, 90 of people affected by COVID-19 and the rest almost equally divided among the other pathogens. The CXR images were obtained from three different sources: Dr. Josepth Cohen GitHub repository; Radiopedia; and NIH dataset.

The local classifiers schema achieved a macro-avg f-score of 0.66 using a LCPN approach with the early fusion of LDN, LETRIST and LPQ features resampled with SMOTE-B2. Furthermore, the local classifiers with LCPN was also able to achieve a f-score of 0.93 for the COVID-19 identification using the early fusion combination of LETRIST, EQP and OBIF features resampled with SMOTE-B2.

Even though this proposal does not provide a definitive COVID-19 diagnosis, and this is not the purpose of this work, the good identification rate achieved for COVID-19 can be quite useful to help the screening of patients in emergency medical support services, that have been severely affected by the pandemic breakthrough.

Chapter 11 Conclusions

Appreciating a better organization of this Chapter we have subdivided it in two sections: (i) The final considerations of this Doctoral Research; and (ii) The research gaps and future work directions after the development of this work.

11.1 Concluding Remarks

Many important real-world classification problems are naturally cast as hierarchical, in which the predicted labels are organized into a hierarchy, such as protein function prediction, text categorization, sound signal classification, biological sequence classification, bioacoustic animal species identification, credit card fraud detection, medical image classification and so on. On the other hand, the imbalanceness issue affects many of these problems, specially in critical application domains where the less frequent classes are the main objective of the prediction, such as in credit card fraud detection and medical image classification. Although the imbalanceness issue is a well-known problem in the machine learning community, there were few works in the literature proposing ways to deal with it for hierarchical classification problems. In this context, this Doctoral Research presented contributions to the machine learning community mainly focused in how to deal with imbalanceness in hierarchical classification datasets.

In **Chapters 2 and 3** we presented an extensive backbone background concerning the topics of Machine Learning (Single-Label, Multi-Label and Hierarchical Classifications) and Imbalance Learning (Metrics and Resampling algorithms). During the studies of these topics we made four contributions related to the Music Information Retrieval and Movie Genre Classification research fields.

In **Chapter 4**, we presented the Multi-Label Tomek Link (MLTL), which adapts the classic Tomek Link algorithm to deal with multi-label samples by using the adjusted hamming distance metric to compare the samples sets of labels. Experimental results in seven well-known datasets showed that MLTL is a competitive technique if compared to other multi-label resampling methods from the literature. Another important contribution presented in **Chapter 4** is the Imb-Mulan framework, an extension to the well-known Mulan framework implementing multi-label resampling algorithms. These contributions

were essential to give an overview regarding the design and development of novel resampling algorithms, which was important to confirm **Hypothesis 4** in **Chapters 8 and 9**.

In **Chapter 5** we presented the first analysis towards the classification of imbalanced hierarchical classification datasets using resampling algorithms, which represents the baseline experiments for this task. We have proposed the use of binary/multi-class and multi-label resampling algorithms in hierarchical classification datasets with single paths and multiple paths, respectively. The experimental analysis over eighteen datasets, supported by statistical analysis, shown that well-known resampling algorithms can partially improve the classification results of hierarchical classification datasets without further adaptations. The contributions of this Chapter partially corroborate with **Hypothesis 2**, showing that even thought the pre-exiting resampling techniques may improve the results, this improvement is not statistically enough and, thus, there is room for improvement for novel and more specific techniques.

Then, in **Chapter 6**, two metrics to evaluate how imbalanced a Hierarchical Multi-Label Dataset is proposed. While the first metric (IRLP) is focused on calculating the imbalance level of a certain Label Path, the second metric (MeanIR) calculates the general imbalance by averaging all the IRLPs from the dataset. The metrics are a direct contribution to corroborate with **Hypothesis 1**. Moreover, In **Chapter 6**, a novel approach to deal with imbalanceness in Hierarchical Multi-Label Datasets was also proposed. This method is based on the conversion of HMD $\leftrightarrow$ MLD, in order to apply well-known Multi-Label resampling algorithms in the dataset to deal with the imbalanceness. The experimental results in one of the biggest hierarchical classification datasets from the literature, i.e., the Free Music Archive (FMA), shown promising results. The label path conversion also corroborate with **Hypothesis 2**, showing that the existing resampling approaches may be used in other ways to reduce the imbalanceness hierarchical dataset.

Chapter 7 presented an approach to deal with different imbalanced hierarchical classification datasets for different local classification approaches, in which binary/multi-class resampling algorithms are intrinsically used inside the training step of the local classification approaches. Besides the classification approaches, we have also proposed different imbalanceness measures considering each of the local classification approaches, which helped to corroborate with **Hypothesis 1**. An extensive experimental analysis with eight datasets from the literature ranging from different contexts and characteristics supported by statistical analysis shown that the proposed measures can measure the imbalanceness in these specific conditions and the proposed resampling approaches can indeed improve the classification results. The findings of **Chapter 7** directly corroborates with **Hypothesis 3**.

In **Chapter 8** we proposed the first advances towards the proposal of global resampling algorithms that are able to handle hierarchical classification datasets as a

whole: the Hierarchical Random Oversampling (HROS) and Undersampling (HRUS). In these algorithms we have designed a way to select the samples from the majority/minority label paths by using the imbalanceness measures proposed in **Chapter 6**. We have also considered the different depth of prediction in the hierarchical classification problems to design these random resampling algorithms. The experimental results in twenty-three datasets with different hierarchical characteristics, supported by statistical analysis, have shown that the proposed global resampling algorithms can improve the classification results. Furthermore, we have also proposed a testbed of datasets for the research community for further studies in the hierarchical classification field.

Chapter 9 presented the Hierarchical Synthetic Oversampling Technique (HSMOTE), the first heuristic global resampling agorithm for hierarchical classification datasets. The proposed oversampling algorithm is an adapted version of the SMOTE and MLSMOTE methods that is able to handle different hierarchical classification datasets. The experimental results with the testbed presented in **Chapter 8**, supported by statistical analysis, comproved that the proposed HSMOTE method can improve the classification results. The algorithmic solutions presented in **Chapters 8 and 9** directly corroborate with **Hypothesis 4**, showing that novel hierarchical resampling methods can indeed improve the classification results.

Finally, in **Chapter 10** we presented the main case study of this Thesis. Using the knowledge acquired during the development of this work we were able to identify an important contribution for a real world problem currently affecting our society: The identification of pneumonia pathogens, such as the SARS-COV-2, in X-ray chest images. Firstly, we have identified that this problem can be also cast as a hierarchical classification problem, since there are biological relationships between the pathogens that causes pneumonia. Secondly, this is a classification problem that suffers from several imbalanceness, since there is a natural imbalanceness between the number of people with healthy lungs and with lungs affected by pneumonia caused by specific types of pathogens, such as COVID-19. We have employed all the resampling approaches proposed in this Thesis, with the exception of the HMD $\leftrightarrow$ MLD conversion, since the problem is not multi-label. The findings of **Chapter 10** definitively corroborates with **Hypothesis 1, 2, 3 and 4**, considering all aspects of this Thesis and its contributions, since the experimental results have show that the proposed resampling approaches improve the baseline results in all classification scenarios.

After analyzing this extensive scenario towards the hierarchical classification of imbalanced data it may be asked what kind of resampling approach should be used when dealing with a new problem. The answer to this question is not straightforward, but we can give some hints based on the findings of this Doctoral Research. First, the use of the flat resampling techniques directly in the hierarchical datasets, mainly studied in **Chapters**

5 and 6, were the least effective approaches when compared with the techniques from **Chapters 7, 8 and 9**. This fact is acceptable and understandable, since they are the baseline for the hierarchy-based approaches. Secondly, the use of the local classifiers with flat resampling, proposed in **Chapter 6**, achieved interesting results in the experiments of the Chapter but also provided the best classification results in the pneumonia classification analysis, presented in **Chapter 10**. Hence, the local classifiers with resampling approaches should be experimentally analyzed when dealing with a new imbalanced hierarchical dataset. Finally, although the global resampling techniques (HROS, HRUS and HSMOTE), proposed in **Chapters 8 and 9**, did not provide the best classification results in the case study of **Chapter 10**, they have shown promising results (proved with statistical analysis) for other hierarchical classification datasets in the Chapters that they were presented and, thus, should also be experimentally tested in a new imbalanced hierarchical dataset.

We must highlight that the contributions regarding the hierarchical resampling approaches were only experimentally tested in datasets with tree taxonomies. Thus, although some of these contributions might work in hierarchical classification datasets with DAG taxonomies, as we have not considered them when designing the protocol of the proposed approaches, we have no further information regarding the performance of the proposed approaches when applied to hierarchical classification problems which DAG-structured class taxonomy.

11.2 Research Gaps and Future Work Directions

After the development of this Doctoral Research some gaps were open and the following future work directions can be considered as further investigations in this context:

- Investigate the use of local classification approaches, i.e., LCN, LCPN and LCL, in the following hierarchical classification scenarios:
  - After resampling the hierarchical classification datasets with binary/multi-class resampling algorithms (in single path problems) and multi-label resampling method (in multiple path problems).
  - After converting Hierarchical Multi-Label Datasets with the HMD $\leftrightarrow$ MLD conversion, presented in Chapter 6.
  - After resampling the hierarchical classification datasets with HROS and HRUS (proposed in Chapter 8) and the HSMOTE (proposed in Chapter 9).
- Investigate the use of other global classification approaches after resampling the hierarchical classification datasets with HROS, HRUS and HSMOTE.

- Propose hierarchical adaptations for the multi-label imbalanceness measures CVIR and SCUMBLE.
- Propose an adapted version of the Tomek Link resampling algorithm for hierarchical classification problems, which will allow the use of an hybrid resampling approach combining the HSMOTE with the proposed Hierarchical Tomek Link.
- Investigate the results of applying a hybrid resampling technique using REMEDIAL-HwR-MLSMOTE with MLTL.
- Investigate the impact of using the One-Against-One decomposition technique in the multi-class experiments.
- Propose novel and/or adapted versions of heuristic resampling methods for hierarchical problems considering the majority/minority samples selection approaches for hierarchical classification datasets proposed in this Thesis.
- Propose a formula/equation to calculate and suggest an increasement/decreasement rate for a given imbalanced dataset (hierarchical or not).
- Investigate the impact of the resampling methods in the label paths visibility.
- Investigate the impact of missing data in the heuristic resampling algorithms.
- Propose hierarchical imbalanceness solutions regarding cost-sensitive solutions an hybrid techniques.
- Investigate the impact of all the proposed approaches in hierarchical classification problems with DAG taxonomies.

Bibliography

ABDULRAZZAQ, M. M.; YASEEN, I. F. T.; NOAH, S. A.; FADHIL, M. A.; ASHOUR, M. U. XMIAR: X-ray medical image annotation and retrieval. In: *Proceedings of the Science and Information Conference*. Las Vegas, USA: SAI, 2019. p. 638–651. Referenced in page(s) 70.

AHMED, M.; MAHMOOD, A. N.; HU, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, Elsevier, v. 60, p. 19–31, 2016. Referenced in page(s) 70.

ALEKSOVSKI, D.; KOCEV, D.; DZEROSKI, S. Evaluation of distance measures for hierarchical multilabel classification in functional genomics. In: *Proceedings of the Workshop on Learning from Multi-Label*. Bled, Slovenia: ECML, 2009. p. 5–16. Referenced in page(s) 61.

ALSHAMAA, D.; CHEHADE, F. M.; HONEINE, P. A hierarchical classification method using belief functions. *Signal Processing*, Elsevier, v. 148, p. 68–77, 2018. Referenced in page(s) 54.

ARIAS, J.; MARTINEZ-GOMEZ, J.; GAMEZ, J. A.; HERRERA, A. G. S. de; MÜLLER, H. Medical image modality classification using discrete bayesian networks. *Computer Vision and Image Understanding*, Elsevier, v. 151, p. 61–71, 2016. Referenced in page(s) 70.

ARIYARATNE, H. B.; ZHANG, D. A novel automatic hierachical approach to music genre classification. In: *Proceedings of the IEEE International Conference on Multimedia and Expo Workshops*. Melbourne, Australia: IEEE, 2012. p. 564–569. Referenced in page(s) 29.

ASSOCIATION, A. M. *ICD-10-CM 2020 The Complete Official Codebook*. USA: American Medical Association, 2019. Referenced in page(s) 214.

ATKINSON, A. B. On the measurement of inequality. *Journal of Economic Theory*, Elsevier, v. 2, n. 3, p. 244–263, 1970. Referenced in page(s) 99.

BAI, J.; JIANG, H.; LI, S.; MA, X. NHL pathological image classification based on hierarchical local information and googlenet-based representations. *BioMed Research International*, Hindawi, v. 19, n. 3, 2019. Referenced in page(s) 70.

BARANDELA, R.; SÁNCHEZ, J. S.; GARCIA, V.; RANGEL, E. Strategies for learning in class imbalance problems. *Pattern Recognition*, Pergamon, v. 36, n. 3, p. 849–851, 2003. Referenced in page(s) 71.

BATISTA, G.; PRATI, R.; MONARD, M. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, v. 6, n. 1, p. 20–29, 2004. Referenced 12 time(s) in page(s) 71, 74, 75, 76, 84, 85, 96, 97, 110, 119, 123, and 165.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, v. 13, n. Feb, p. 281–305, 2012. Referenced 2 time(s) in page(s) 129 and 157.

BERTOLINI, D.; OLIVEIRA, L. S.; JUSTINO, E.; SABOURIN, R. Texture-based descriptors for writer identification and verification. *Expert Systems with Applications*, Elsevier, v. 40, n. 6, p. 2069–2080, 2013. Referenced in page(s) 222.

BEWLEY, M. S.; NOURANI-VATANI, N.; RAO, D.; DOUILLARD, B.; PIZARRO, O.; WILLIAMS, S. B. Hierarchical classification in auv imagery. In: *Proceedings of the International Conference on Field and Service Robotics*. Chamonix, France: Springer, 2015. p. 3–16. Referenced in page(s) 54.

BINDER, A.; KAWANABE, M.; BREFELD, U. Efficient classification of images with taxonomies. In: *Proceedings of the Asian Conference on Computer Vision*. Xian, China: Springer, 2009. p. 351–362. Referenced in page(s) 29.

BLOCKEEL, H.; SCHIETGAT, L.; STRUYF, J.; DŽEROSKI, S.; CLARE, A. Decision trees for hierarchical multilabel classification: A case study in functional genomics. In: *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*. Berlin, Germany: Springer, 2006. p. 18–29. Referenced 3 time(s) in page(s) 61, 62, and 139.

BONIFAZI, G.; CAPOBIANCO, G.; SERRANTI, S. A hierarchical classification approach for recognition of low-density (ldpe) and high-density polyethylene (hdpe) in mixed plastic waste based on short-wave infrared (swir) hyperspectral imaging. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, Elsevier, v. 198, p. 115–122, 2018. Referenced in page(s) 29.

BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: *Proceedings of the Annual Workshop on Computational Learning Theory*. Pittsburgh, USA: ACM, 1992. p. 144–152. Referenced in page(s) 42.

BOUTELL, M. R.; LUO, J.; SHEN, X.; BROWN, C. M. Learning multi-label scene classification. *Pattern Recognition*, v. 37, n. 9, p. 1757–1771, 2004. Referenced 2 time(s) in page(s) 44 and 117.

BRAHNAM, S.; NANNI, L.; SEXTON, R. Introduction to neonatal facial pain detection using common and advanced face classification techniques. Springer, p. 225–253, 2007. Referenced in page(s) 215.

BRANCO, P.; TORGO, L.; RIBEIRO, R. P. A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys*, ACM, v. 49, n. 2, p. 31, 2016. Referenced in page(s) 30.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Referenced in page(s) 41.

BRIGGS, F.; HUANG, Y.; RAICH, R.; EFTAXIAS, K.; LEI, Z.; CUKIERSKI, W.; HADLEY, S. F.; HADLEY, A.; BETTS, M.; FERN, X. Z. The 9th annual mlsp competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In: *Proceedings of the IEEE International Workshop on Machine*

Learning for Signal Processing. Southampton, United Kingdom: IEEE, 2013. p. 1–8. Referenced 2 time(s) in page(s) 177 and 201.

BRINKER, K.; FÜRNKRANZ, J.; HÜLLERMEIER, E. A unified model for multilabel classification and ranking. In: *Proceedings of the European Conference on Artificial Intelligence*. Riva del Garda, Italy: AEPIA, 2006. p. 489–493. Referenced 2 time(s) in page(s) 44 and 117.

BRONZE, G. Estudo brasileiro identifica COVID-19 por raio-x com 90% de eficácia. *Saúde*, CNN Brasil, 2020. Available from Internet: https://bit.ly/cnnbrasil-covid19. Referenced in page(s) 36.

BUNKHUMPORNPAT, C.; SINAPIROMSARAN, K.; LURSINSAP, C. Safe-level-smote: Safe-level-synthetic minority oversampling technique for handling the class imbalanced problem. In: *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Bangkok, Thailand: ACM, 2009. p. 475–482. Referenced 3 time(s) in page(s) 71, 90, and 91.

BURRED, J. J.; LERCH, A. A hierarchical approach to automatic musical genre classification. In: *Proceedings of the International Conference on Digital Audio Effects*. London, United Kingdom: DAFx, 2003. p. 8–11. Referenced in page(s) 29.

CECI, M.; MALERBA, D. Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, Springer, v. 28, n. 1, p. 37–78, 2007. Referenced in page(s) 54.

CERRI, R.; BARROS, R. C.; CARVALHO, A. C. D. Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences*, Elsevier, v. 80, n. 1, p. 39–56, 2014. Referenced in page(s) 54.

CERRI, R.; BARROS, R. C.; CARVALHO, A. C. de. Hierarchical classification of gene ontology-based protein functions with neural networks. In: *Proceedings of the International Joint Conference on Neural Networks*. Killarney, Ireland: IEEE, 2015. p. 1–8. Referenced 4 time(s) in page(s) 129, 139, 178, and 202.

CERRI, R.; BARROS, R. C.; CARVALHO, A. C. de; JIN, Y. Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC Bioinformatics*, BioMed Central, v. 17, n. 1, p. 373, 2016. Referenced in page(s) 55.

CERRI, R.; CARVALHO, A. C. P. F. de. New top-down methods using svms for hierarchical multilabel classification problems. In: *Proceedings of the International Joint Conference on Neural Networks*. Barcelona, Spain: IEEE, 2010. p. 1–8. Referenced 5 time(s) in page(s) 7, 55, 56, 57, and 58.

CERRI, R.; PAPPA, G. L.; CARVALHO, A. C. P.; FREITAS, A. A. An extensive evaluation of decision tree–based hierarchical multilabel classification methods and performance measures. *Computational Intelligence*, Wiley Online Library, v. 31, n. 1, p. 1–46, 2015. Referenced 9 time(s) in page(s) 7, 56, 62, 63, 64, 65, 66, 67, and 68.

CESA-BIANCHI, N.; GENTILE, C.; ZANIBONI, L. Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research*, v. 7, n. Jan, p. 31–54, 2006. Referenced 2 time(s) in page(s) 64 and 65.

CESA-BIANCHI, N.; RE, M.; VALENTINI, G. Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference. *Machine Learning*, Springer, v. 88, n. 1-2, p. 209–241, 2012. Referenced in page(s) 30.

CHAKRABARTI, S.; DOM, B.; AGRAWAL, R.; RAGHAVAN, P. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB Journal*, v. 7, n. 3, p. 163–178, 1998. Referenced in page(s) 28.

CHAN, P. K.; STOLFO, S. J. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, USA: ACM, 1998. p. 164–168. Referenced in page(s) 29.

CHAPELLE, O.; SCHOLKOPF, B.; ZIEN, A. Semi-supervised learning. *IEEE Transactions on Neural Networks*, IEEE, v. 20, n. 3, p. 542–542, 2009. Referenced in page(s) 39.

CHARTE, F.; RIVAS, A. J. R.; JESUS, M. del; HERRERA, F. MLeNN: a first approach to heuristic multilabel undersampling. In: *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning*. Salamanca, Spain: Springer, 2014. p. 1–9. Referenced 6 time(s) in page(s) 103, 104, 105, 111, 119, and 124.

CHARTE, F.; RIVERA, A.; JESUS, M. del; HERRERA, F. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, v. 163, p. 3–16, 2015. Referenced 6 time(s) in page(s) 102, 104, 105, 109, 119, and 124.

CHARTE, F.; RIVERA, A.; JESUS, M. del; HERRERA, F. MLSMOTE: Approaching imbalanced multilabel learning through synthetic instance generation. *Knowledge-Based Systems*, v. 89, p. 385–397, 2015. Referenced 12 time(s) in page(s) 104, 106, 107, 109, 119, 123, 124, 164, 185, 191, 207, and 243.

CHARTE, F.; RIVERA, A.; JESUS, M. del; HERRERA, F. Resampling multilabel datasets by decoupling highly imbalanced labels. In: *Proceedings of the International Conference on Hybrid Artificial Intelligence Systems*. Bilbao, Spain: Springer, 2015. p. 489–501. Referenced 4 time(s) in page(s) 107, 108, 119, and 124.

CHARTE, F.; RIVERA, A.; JESUS, M. del; HERRERA, F. Remedial-hwr: Tackling multilabel imbalance through label decoupling and data resampling hybridization. *Neurocomputing*, v. 326–327, n. 31, p. 110–122, 2019. Referenced 2 time(s) in page(s) 109 and 124.

CHARTE, F.; RIVERA, A.; JESUS, M. J. del; HERRERA, F. A first approach to deal with imbalance in multi-label datasets. In: *Proceedings of the International Conference on Hybrid Artificial Intelligence Systems*. Salamanca, Spain: Springer, 2013. p. 150–160. Referenced 10 time(s) in page(s) 97, 102, 103, 119, 124, 134, 135, 145, 170, and 187.

CHARTE, F.; RIVERA, A.; JESUS, M. J. del; HERRERA, F. Concurrence among imbalanced labels and its influence on multilabel resampling algorithms. In: *Proceedings of the International Conference on Hybrid Artificial Intelligence Systems*. Salamanca, Spain: Springer, 2014. p. 110–121. Referenced in page(s) 99.

CHARTE, F.; RIVERA, A. J.; JESUS, M. J. del; HERRERA, F. Dealing with difficult minority labels in imbalanced mutilabel data sets. *Neurocomputing*, Elsevier, 2017. Referenced 4 time(s) in page(s) 8, 100, 101, and 109.

CHAWLA, N.; BOWYER, K.; HALL, L.; KEGELMEYER, P. Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, v. 16, p. 321–357, 2002. Referenced 8 time(s) in page(s) 70, 71, 85, 86, 87, 88, 106, and 185.

CHEBIRA, A.; BARBOTIN, Y.; JACKSON, C.; MERRYMAN, T.; SRINIVASA, G.; MURPHY, R. F.; KOVAČEVIĆ, J. A multiresolution approach to automated classification of protein subcellular location images. *BMC Bioinformatics*, Springer, v. 8, n. 1, p. 210, 2007. Referenced in page(s) 215.

CHEN, B.; DUAN, L.; HU, J. Composite kernel based svm for hierarchical multi-label gene function classification. In: *Proceedings of the International Joint Conference on Neural Networks*. Brisbane, Australia: IEEE, 2012. p. 1–6. Referenced in page(s) 30.

CIESLAK, D. A.; HOENS, T. R.; CHAWLA, N. V.; KEGELMEYER, W. P. Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery*, Springer, v. 24, n. 1, p. 136–158, 2012. Referenced in page(s) 71.

CLARE, A. Machine learning and data mining for yeast functional genomics. *Aberystwyth: The University of Wales (Ph.D. Thesis)*, 2003. Referenced in page(s) 60.

CLARE, A.; KING, R. D. Predicting gene function in saccharomyces cerevisiae. *Bioinformatics*, Oxford University Press, v. 19, n. 2, p. 42–49, 2003. Referenced 4 time(s) in page(s) 59, 129, 177, and 202.

COHEN, J. P.; MORRISON, P.; DAO, L. COVID-19 image data collection. *arXiv 2003.11597*, 2020. Referenced 3 time(s) in page(s) 212, 217, and 229.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Introduction to Algorithms*. 3. ed. London, United Kingdom: MIT Press, 2009. Referenced in page(s) 123.

COST, S.; SALZBERG, S. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine learning*, Springer, v. 10, n. 1, p. 57–78, 1993. Referenced in page(s) 190.

COSTA, E. P.; LORENA, A. C.; CARVALHO, A. C.; FREITAS, A. A. Top-down hierarchical ensembles of classifiers for predicting g-protein-coupled-receptor functions. In: *Proceedings of the Brazilian Symposium on Bioinformatics*. Santo André, Brazil: SBC, 2008. p. 35–46. Referenced in page(s) 29.

COSTA, H.; GALVÃO, L.; MERSCHMANN, L.; SOUZA, M. A vns algorithm for feature selection in hierarchical classification context. *Electronic Notes in Discrete Mathematics*, Elsevier, v. 66, p. 79–86, 2018. Referenced in page(s) 55.

COSTA, Y. M. G. Pesquisadores do paraná estudam método para diagnosticar COVID-19 com raios x. *CBN Maringá: Ciência e Tecnologia*, CBN, 2020. Available from Internet: https://bit.ly/cbn-mga-covid19. Referenced in page(s) 36.

COSTA, Y. M. G.; OLIVEIRA, L.; KOERICH, A. L.; GOUYON, F.; MARTINS, J. Music genre classification using LBP textural features. *Signal Processing*, v. 92, n. 11, p. 2723–2737, 2012. Referenced in page(s) 222.

CRAMMER, K.; DREDZE, M.; GANCHEV, K.; TALUKDAR, P. P.; CARROLL, S. Automatic code assignment to medical text. In: *Proceedings of the Workshop on Biological, Translational, and Clinical Language Processing*. Prague, Czech Republic: ACM, 2007. p. 129–136. Referenced in page(s) 117.

CRISTIANINI, N.; SHAWE-TAYLOR, J. *An introduction to support vector machines and other kernel-based learning methods*. 1. ed. London, United Kingdom: Cambridge University Press, 2000. 204 p. Referenced in page(s) 42.

CROSIER, M.; GRIFFIN, L. D. Using basic image features for texture classification. *International Journal of Computer Vision*, Kluwer Academic Publishers, USA, v. 88, n. 3, p. 447–460, 2010. Referenced in page(s) 224.

DAVIS, J.; GOADRICH, M. The relationship between precision-recall and roc curves. In: *Proceedings of the International Conference on Machine Learning*. Pittsburgh, USA: ACM, 2006. p. 233–240. Referenced 2 time(s) in page(s) 71 and 203.

DECORO, C.; BARUTCUOGLU, Z.; FIEBRINK, R. Bayesian aggregation for hierarchical genre classification. In: *Proceedings of the International Society for Music Information Retrieval Conference*. Kobe, Japan: ISMIR, 2007. p. 77–80. Referenced in page(s) 29.

DEFFERRARD, M.; BENZI, K.; VANDERGHEYNST, P.; BRESSON, X. FMA: A dataset for music analysis. In: *Proceedings of the International Society for Music Information Retrieval Conference*. Suzhou, China: ISMIR, 2017. Referenced 5 time(s) in page(s) 117, 129, 155, 177, and 202.

DIAMANTINI, C.; POTENA, D. Bayes vector quantizer for class-imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 21, n. 5, p. 638–651, 2009. Referenced in page(s) 71.

DIMITROVSKI, I.; KOCEV, D.; LOSKOVSKA, S.; DŽEROSKI, S. Hierarchical annotation of medical images. *Pattern Recognition*, Elsevier, v. 44, n. 10-11, p. 2436–2449, 2011. Referenced 5 time(s) in page(s) 29, 61, 129, 177, and 201.

DIMITROVSKI, I.; KOCEV, D.; LOSKOVSKA, S.; DZEROSKI, S. Hierarchical annotation of medical images. *Pattern Recognition*, v. 44, n. 10, p. 2436–2449, 2011. Referenced in page(s) 155.

DIMITROVSKI, I.; KOCEV, D.; LOSKOVSKA, S.; DŽEROSKI, S. Hierarchical classification of diatom images using ensembles of predictive clustering trees. *Ecological Informatics*, Elsevier, v. 7, n. 1, p. 19–29, 2012. Referenced 4 time(s) in page(s) 29, 129, 177, and 202.

DOMINGOS, P. Metacost: A general method for making classifiers cost-sensitive. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Paris, France: ACM, 1999. p. 155–164. Referenced in page(s) 71.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern classification*. New York: John Wiley & Sons Inc, 2001. Referenced in page(s) 40.

DUMAIS, S.; CHEN, H. Hierarchical classification of web content. In: *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. Athens, Greece: ACM, 2000. p. 256–263. Referenced in page(s) 156.

EISNER, R.; POULIN, B.; SZAFRON, D.; LU, P.; GREINER, R. Improving protein function prediction using the hierarchical structure of the gene ontology. In: *Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*. La Jolla, USA: IEEE, 2005. p. 1–10. Referenced in page(s) 54.

ELISSEEFF, A.; WESTON, J. A kernel method for multi-labelled classification. *Advances in Neural Information Processing Systems*, Springer, v. 15, n. 1, p. 681–687, 2002. Referenced 2 time(s) in page(s) 100 and 117.

FAGNI, T.; SEBASTIANI, F. On the selection of negative examples for hierarchical text categorization. In: *Proceedings of the Language & Technology Conference*. Poznan, Poland: ACM, 2007. p. 24–28. Referenced 2 time(s) in page(s) 29 and 54.

FELIX, R. Coronavírus: Pesquisadores do paraná criam método para diagnóstico com raio-x. *Inovação*, Gazeta do Povo, 2020. Available from Internet: https://bit.ly/gazeta-covid19. Referenced in page(s) 36.

FERNÁNDEZ, A.; LÓPEZ, V.; GALAR, M.; JESUS, M. J. D.; HERRERA, F. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-Based Systems*, Elsevier, v. 42, p. 97–110, 2013. Referenced 4 time(s) in page(s) 7, 71, 72, and 73.

FILHO, P. L. P.; OLIVEIRA, L. S.; NISGOSKI, S.; BRITTO JR, A. S. Forest species recognition using macroscopic images. *Machine Vision and Applications*, Springer, v. 25, n. 4, p. 1019–1031, 2014. Referenced in page(s) 222.

FUKUNAGA, K. *Introduction to Statistical Pattern Recognition*. 2. ed. Netherlands: Academic Press, 2013. Referenced in page(s) 37.

FÜRNKRANZ, J.; HÜLLERMEIER, E.; MENCÍA, E. L.; BRINKER, K. Multilabel classification via calibrated label ranking. *Machine Learning*, v. 73, n. 2, p. 133–153, 2008. Referenced 2 time(s) in page(s) 45 and 117.

GARCÍA-PEDRAJAS, N.; PÉREZ-RODRÍGUEZ, J.; GARCÍA-PEDRAJAS, M.; ORTIZ-BOYER, D.; FYFE, C. Class imbalance methods for translation initiation site recognition in dna sequences. *Knowledge-Based Systems*, Elsevier, v. 25, n. 1, p. 22–34, 2012. Referenced in page(s) 71.

GATTAL, A.; DJEDDI, C.; SIDDIQI, I.; CHIBANI, Y. Gender classification from offline multi-script handwriting images using oriented basic image features (oBIFs). *Expert Systems with Applications*, v. 99, p. 155 – 167, 2018. Referenced in page(s) 224.

GERHARD, B. Pesquisadores maringaenses fazem estudo que pode ajudar a identificar o COVID-19. *Balanço Geral Maringá*, RIC TV, 2020. Available from Internet: https://bit.ly/ric-covid19. Referenced in page(s) 36.

GOUTTE, C.; GAUSSIER, E. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In: *Proceedings of the European Conference on Information Retrieval*. Santiago de Compostela, Spain: ACM, 2005. p. 345–359. Referenced 2 time(s) in page(s) 71 and 232.

GOZES, O.; FRID-ADAR, M.; GREENSPAN, H.; BROWNING, P. D.; ZHANG, H.; JI, W.; BERNHEIM, A.; SIEGEL, E. Rapid AI development cycle for the coronavirus (COVID-19) pandemic: Initial results for automated detection & patient monitoring using deep learning CT image analysis. *arXiv preprint arXiv:2003.05037*, 2020. Referenced 2 time(s) in page(s) 217 and 220.

GRASSELLI, G.; PESENTI, A.; CECCONI, M. Critical care utilization for the COVID-19 outbreak in lombardy, italy: early experience and forecast during an emergency response. *JAMA*, 2020. Referenced in page(s) 211.

GUAN, W.-j.; NI, Z.-y.; HU, Y.; LIANG, W.-h.; OU, C.-q.; HE, J.-x.; LIU, L.; SHAN, H.; LEI, C.-l.; HUI, D. S. C. Clinical characteristics of coronavirus disease 2019 in china. *New England Journal of Medicine*, Mass Medical Society, 2020. Referenced in page(s) 211.

GUZZONI, J. Pesquisadores se unem para agilizar diagnóstico da COVID-19. *Boa Noite Paraná*, RPC TV, 2020. Available from Internet: https://bit.ly/rpc-mga-covid19. Referenced in page(s) 36.

HA, T. M.; BUNKE, H. Off-line, handwritten numeral recognition by perturbation method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 19, n. 5, p. 535–539, 1997. Referenced in page(s) 85.

HAIXIANG, G.; YIJING, L.; SHANG, J.; MINGYUN, G.; YUANYUE, H.; BING, G. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, Elsevier, v. 73, p. 220–239, 2017. Referenced 2 time(s) in page(s) 30 and 158.

HAN, H.; WANG, W.-Y.; MAO, B.-H. Borderline-smote: A new oversampling method in imbalanced datasets learning. In: *Proceedings of the International Conference on Intelligent Computing*. Hefei, China: Springer, 2005. p. 878–887. Referenced 4 time(s) in page(s) 71, 86, 89, and 186.

HART, P. The condensed nearest neighbor rule (corresp.). *IEEE Transactions on Information Theory*, Citeseer, v. 14, n. 3, p. 515–516, 1968. Referenced 3 time(s) in page(s) 71, 78, and 79.

HASTIE, T.; TIBSHIRANI, R. Classification by pairwise coupling. *Advances in Neural Information Processing Systems*, Springer, v. 11, n. 1, p. 507–513, 1998. Referenced in page(s) 73.

HE, H.; BAI, Y.; GARCIA, E. A.; LI, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In: *Proceedings of the IEEE International Joint Conference Neural Networks*. Hong Kong: IEEE, 2008. p. 1322–1328. Referenced 3 time(s) in page(s) 71, 94, and 96.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, USA: IEEE, 2016. p. 770–778. Referenced 2 time(s) in page(s) 217 and 225.

HERRERA, F.; CHARTE, F.; RIVERA, A. J.; JESUS, M. J. D. *Multilabel Classification: Problem Analysis, Metrics and Techniques*. Granada, Spain: Springer, 2016. Referenced 4 time(s) in page(s) 8, 97, 99, and 100.

IDRSSI, A. E.; RUICHEK, Y. Palmprint recognition using state-of-the-art local texture descriptors: A comparative study. *IET Biometrics*, IET, 2020. Referenced in page(s) 224.

JANTZEN, J.; NORUP, J.; DOUNIAS, G.; BJERREGAARD, B. Pap-smear benchmark data for pattern classification. *Nature Inspired Smart Information Systems*, p. 1–9, 2005. Referenced in page(s) 215.

JAPKOWICZ, N.; STEPHEN, S. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, v. 6, n. 5, p. 429–449, 2002. Referenced 2 time(s) in page(s) 134 and 144.

JELLER, A. Estudo possibilita diagnóstico da COVID-19 via raio-x. *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*, CAPES, 2020. Available from Internet: https://bit.ly/capes-covid19. Referenced in page(s) 36.

KANNALA, J.; RAHTU, E. BSIF: Binarized statistical image features. In: *Proceedings of the International Conference on Pattern Recognition*. Tsukuba, Japan: ACM, 2012. p. 1363–1366. Referenced in page(s) 223.

KHAN, A. I.; SHAH, J. L.; BHAT, M. Coronet: A deep neural network for detection and diagnosis of COVID-19 from chest x-ray images. *arXiv preprint arXiv:2004.04931*, 2020. Referenced 2 time(s) in page(s) 218 and 220.

KHOWAJA, S. A.; YAHYA, B. N.; LEE, S.-L. Hierarchical classification method based on selective learning of slacked hierarchy for activity recognition systems. *Expert Systems with Applications*, Elsevier, v. 88, p. 165–177, 2017. Referenced in page(s) 29.

KIRITCHENKO, S.; MATWIN, S.; FAMILI, F. Hierarchical text categorization as a tool of associating genes with gene ontology codes. In: *Proceedings of the European Workshop on Data Mining and Text Mining in Bioinformatics*. Pisa, Italy: ACM, 2004. p. 30–34. Referenced in page(s) 63.

KIRITCHENKO, S.; MATWIN, S.; FAMILI, F. Functional annotation of genes using hierarchical text categorization. In: *Proceedings of the ACL Workshop on Linking Biological Literature*. Detroit, USA: ACL, 2005. Referenced 2 time(s) in page(s) 63 and 64.

KITTLER, J.; HATEF, M.; DUIN, R. P. W.; MATAS, J. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 20, n. 3, p. 226–239, 1998. Referenced 2 time(s) in page(s) 225 and 226.

KLIMT, B.; YANG, Y. The enron corpus: A new dataset for email classification research. In: *Proceedings of the European Conference on Machine Learning*. Pisa, Italy: ECML, 2004. p. 217–226. Referenced 2 time(s) in page(s) 99 and 117.

KLIMT, B.; YANG, Y. Introducing the enron corpus. In: *Proceedings of the Conference on Email and Anti-Spam*. Mountain View, California, USA: ACM, 2004. p. 1–2. Referenced 2 time(s) in page(s) 177 and 201.

KOLLER, D.; SAHAMI, M. Hierarchically classifying documents using very few words. In: *Proceedings of the International Conference on Machine Learning*. San Francisco, USA: ACM, 1997. Referenced in page(s) 28.

KOTSIFAKOS, A.; KOTSIFAKOS, E. E.; PAPAPETROU, P.; ATHITSOS, V. Genre classification of symbolic music with smbgt. In: *Proceedings of the International Conference on Pervasive Technologies Related to Assistive Environments*. Island of Rhodes, Greece: ACM, 2013. p. 1–7. Referenced in page(s) 40.

KRAWCZYK, B. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, Springer, v. 5, n. 4, p. 221–232, 2016. Referenced in page(s) 185.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. Springer, p. 1097–1105, 2012. Referenced in page(s) 225.

KUMAR, S.; ROWLEY, H. A.; WANG, X.; RODRIGUES, J. J. M. *Hierarchical classification in credit card data extraction*. [S.l.]: Google Patents, 2015. US Patent 9,213,907. Referenced 2 time(s) in page(s) 30 and 70.

LAURIKKALA, J. Improving identification of difficult small classes by balancing class distribution. In: *Proceedings of the Conference on Artificial Intelligence in Medicine in Europe*. Cascais, Portugal: Springer, 2001. p. 63–66. Referenced 2 time(s) in page(s) 82 and 83.

LI, D.; JU, Y.; ZOU, Q. Protein folds prediction with hierarchical structured SVM. *Current Proteomics*, Bentham Science Publishers, v. 13, n. 2, p. 79–85, 2016. Referenced in page(s) 30.

LI, L.; QIN, L.; XU, Z.; YIN, Y.; WANG, X.; KONG, B.; BAI, J.; LU, Y.; FANG, Z.; SONG, Q. Artificial intelligence distinguishes COVID-19 from community acquired pneumonia on chest ct. *Radiology*, Radiological Society of North America, p. 200905, 2020. Referenced 2 time(s) in page(s) 217 and 220.

LI, X.; KUANG, D.; LING, C. X. Active learning for hierarchical text classification. In: *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Kuala Lumpur, Malaysia: ACM, 2012. p. 14–25. Referenced in page(s) 28.

LIN, C.; ZOU, Y.; QIN, J.; LIU, X.; JIANG, Y.; KE, C.; ZOU, Q. Hierarchical classification of protein folds using a novel ensemble classifier. *PLOS One Journal*, v. 8, n. 2, p. e56499, 2013. Referenced in page(s) 29.

MACKENZIE, G. The definition and classification of pneumonia. *Pneumonia*, Springer, v. 8, n. 1, p. 14, 2016. Referenced in page(s) 213.

MADJAROV, G.; KOCEV, D.; GJORGJEVIKJ, D.; DŽEROSKI, S. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, v. 45, n. 9, p. 3084–3104, 2012. Referenced 2 time(s) in page(s) 7 and 46.

MAGUOLO, G.; NANNI, L. A critic evaluation of methods for COVID-19 automatic detection from x-ray images. *arXiv preprint arXiv:2004.12823*, 2020. Referenced in page(s) 230.

MANGOLIN, R. B.; PEREIRA, R. M.; BRITTO JR, A. S.; SILLA JR, C. N.; FELTRIM, V. D.; GONCALVES, D. B.; COSTA, Y. M. G. A multimodal approach for multi-label movie genre classification. *Multimedia Tools and Applications*, Springer, v. 79, n. 43, p. 1–30, 2020. Referenced 2 time(s) in page(s) 34 and 69.

MANI, I.; ZHANG, I. knn approach to unbalanced data distributions: a case study involving information extraction. In: *Proceedings of Workshop on Learning from Imbalanced Datasets*. Washington DC, USA: ACM, 2003. v. 126. Referenced 3 time(s) in page(s) 71, 75, and 78.

MCNAMARA, D. S.; CROSSLEY, S. A.; ROSCOE, R. D.; ALLEN, L. K.; DAI, J. A hierarchical classification approach to automated essay scoring. *Assessing Writing*, v. 23, p. 35–59, 2015. Referenced in page(s) 29.

MELO, A.; PAULHEIM, H.; VÖLKER, J. Type prediction in rdf knowledge bases using hierarchical multilabel classification. In: *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*. Nimes, France: ACM, 2016. p. 14. Referenced in page(s) 54.

MERMELSTEIN, P. Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence*, v. 116, p. 374–388, 1976. Referenced in page(s) 139.

MESAROS, A.; HEITTOLA, T.; BENETOS, E.; FOSTER, P.; LAGRANGE, M.; VIRTANEN, T.; PLUMBLEY, M. Detection and classification of acoustic scenes and events. *IEEE Transactions on Audio, Speech, and Language Processing*, v. 26, n. 2, p. 379–393, 2018. Referenced 2 time(s) in page(s) 30 and 70.

METZ, J.; FREITAS, A. A.; MONARD, M. C.; CHERMAN, E. A. A study on the selection of local training sets for hierarchical classification tasks. *Proceedings of the Encontro Nacional de Inteligncia Artificial*, Natal, Brazil, p. 572–583, 2011. Referenced 3 time(s) in page(s) 129, 177, and 201.

MEYER, D. Support vector machines: The interface to libsvm in package e1071. Citeseer, 2004. Referenced 3 time(s) in page(s) 7, 42, and 43.

MITCHELL, T. M. *Machine Learning*. Burr Ridge, USA: McGraw Hill, 1997. 870–877 p. Referenced in page(s) 37.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas Inteligentes Fundamentos e Aplicações*, v. 1, n. 1, p. 32, 2003. Referenced 2 time(s) in page(s) 37 and 38.

MUKAKA, M. M. A guide to appropriate use of correlation coefficient in medical research. *Malawi Medical Journal*, Medical Association of Malawi, v. 24, n. 3, p. 69–71, 2012. Referenced in page(s) 166.

MUSHER, D. M.; THORNER, A. R. Community-acquired pneumonia. *New England Journal of Medicine*, Mass Medical Soc, v. 371, n. 17, p. 1619–1628, 2014. Referenced in page(s) 211.

NAIK, A.; RANGWALA, H. Large-scale hierarchical classification with rare categories and inconsistencies. *AI Matters*, ACM, v. 2, n. 3, p. 27–29, 2016. Referenced in page(s) 30.

NAIK, A.; RANGWALA, H. *Large Scale Hierarchical Classification: State of the Art*. USA: Springer, 2018. Referenced in page(s) 157.

NAKANO, F. K.; LIETAERT, M.; VENS, C. Machine learning for discovering missing or wrong protein function annotations. *BMC Bioinformatics*, v. 20, n. 1, p. 485, 2019. Referenced in page(s) 157.

NAKANO, F. K.; PINTO, W. J.; PAPPA, G. L.; CERRI, R. Top-down strategies for hierarchical classification of transposable elements with neural networks. In: *Proceedings of the International Joint Conference on Neural Networks*. Anchorage, USA: IEEE, 2017. p. 2539–2546. Referenced in page(s) 55.

NAKANO, F. kenji; MASTELINI, S. M.; BARBON, S.; CERRI, R. Stacking methods for hierarchical classification. In: *Proceedings of the IEEE International Conference on Machine Learning and Applications*. Cancun, Mexico: IEEE, 2017. p. 289–296. Referenced in page(s) 54.

NANNI, L.; LUMINI, A.; BRAHNAM, S. Local binary patterns variants as texture descriptors for medical image analysis. *Artificial Intelligence in Medicine*, Elsevier, v. 49, n. 2, p. 117–125, 2010. Referenced 3 time(s) in page(s) 215, 220, and 222.

NAPIERAŁA, K.; STEFANOWSKI, J.; WILK, S. Learning from imbalanced data in presence of noisy and borderline examples. In: *Proceedings of the International Conference on Rough Sets and Current Trends in Computing*. Warsaw, Poland: Springer, 2010. p. 158–167. Referenced 3 time(s) in page(s) 71, 91, and 95.

NARIN, A.; KAYA, C.; PAMUK, Z. Automatic detection of coronavirus disease (COVID-19) using x-ray images and deep convolutional neural networks. *arXiv preprint arXiv:2003.10849*, 2020. Referenced 2 time(s) in page(s) 217 and 220.

NEWELL, A. J.; GRIFFIN, L. D. Writer identification using oriented basic image features and the delta encoding. *Pattern Recognition*, v. 47, n. 6, p. 2255 – 2265, 2014. Referenced in page(s) 224.

OJALA, T.; PIETIKÄINEN, M.; HARWOOD, D. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, Elsevier, v. 29, n. 1, p. 51–59, 1996. Referenced 2 time(s) in page(s) 221 and 222.

OJANSIVU, V.; HEIKKILÄ, J. Blur insensitive texture classification using local phase quantization. In: *Proceedings of the International Conference on Image and Signal Processing*. Herbourg-Octeville, France: Springer, 2008. p. 236–243. Referenced in page(s) 224.

ORGANIZATION, W. H. Coronavirus disease 2019 (COVID-19): situation report, 72. World Health Organization, 2020. Referenced 2 time(s) in page(s) 211 and 213.

OTERO, F. E.; FREITAS, A. A.; JOHNSON, C. G. A hierarchical classification ant colony algorithm for predicting gene ontology terms. In: *Proceedings of the European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*. Tubingen, Germany: Springer, 2009. p. 68–79. Referenced 3 time(s) in page(s) 29, 58, and 59.

OTERO, F. E.; FREITAS, A. A.; JOHNSON, C. G. A hierarchical multi-label classification ant colony algorithm for protein function prediction. *Memetic Computing*, Springer, v. 2, n. 3, p. 165–181, 2010. Referenced in page(s) 58.

OZTURK, T.; TALO, M.; YILDIRIM, E. A.; BALOGLU, U. B.; YILDIRIM, O.; ACHARYA, U. R. Automated detection of COVID-19 cases using deep neural networks with x-ray images. *Computers in Biology and Medicine*, Elsevier, p. 103792, 2020. Referenced 2 time(s) in page(s) 218 and 220.

O'GRADY, K.-A. F.; TORZILLO, P. J.; FRAWLEY, K.; CHANG, A. B. The radiological diagnosis of pneumonia in children. *Pneumonia*, BioMed Central, v. 5, n. 1, p. 38, 2014. Referenced in page(s) 213.

PARMEZAN, A. R. S.; SOUZA, V. M.; BATISTA, G. E. Towards hierarchical classification of data streams. In: *Proceedings of the Iberoamerican Congress on Pattern Recognition*. Madrid, Spain: Springer, 2018. p. 314–322. Referenced 3 time(s) in page(s) 129, 177, and 201.

PARTALAS, I.; KOSMOPOULOS, A.; BASKIOTIS, N.; ARTIÈRES, T.; PALIOURAS, G.; GAUSSIER, É.; ANDROUTSOPOULOS, I.; AMINI, M.; GALLINARI, P. LSHTC: A benchmark for large-scale text classification. *CoRR*, abs/1503.08581, 2015. Referenced in page(s) 155.

PARVEEN, N.; SATHIK, M. M. Detection of pneumonia in chest x-ray images. *Journal of X-Ray Science and Technology*, IOS Press, v. 19, n. 4, p. 423–428, 2011. Referenced 2 time(s) in page(s) 215 and 220.

PEREIRA, G. T.; GABRIEL, P. H.; CERRI, R. Hierarchical classification of transposable elements with a weighted genetic algorithm. In: *Proceedings of the EPIA Conference on Artificial Intelligence*. Vila Real, Portugal: Springer, 2019. p. 737–749. Referenced 4 time(s) in page(s) 129, 139, 178, and 202.

PEREIRA, R. M. Novo sistema de diagnóstico por raio-x. *PanNews SP*, JovemPan, 2020. Available from Internet: https://bit.ly/jovempan-covid19. Referenced in page(s) 36.

PEREIRA, R. M.; BERTOLINI, D.; TEIXEIRA, L. O.; SILLA JR, C. N.; COSTA, Y. M. G. COVID-19 identification in chest x-ray images on flat and hierarchical classification scenarios. *Computer Methods and Programs in Biomedicine*, Elsevier, v. 194, p. 1–28, 2020. Referenced 6 time(s) in page(s) 29, 30, 33, 35, 70, and 213.

PEREIRA, R. M.; COSTA, Y. M. G.; AGUIAR, R. L.; BRITTO JR, A. S.; OLIVEIRA, L. E. S.; SILLA JR, C. N. Representation learning vs. handcrafted features for music genre classification. In: *Proceedings of the International Joint Conference on Neural Networks*. Budapest, Hungary: IEEE, 2019. p. 1–8. Referenced 2 time(s) in page(s) 33 and 68.

PEREIRA, R. M.; COSTA, Y. M. G.; SILLA JR, C. N. Dealing with imbalanceness in hierarchical multi-label datasets using multi-label resampling techniques. In: *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*. Volos, Greece: IEEE, 2018. p. 818–824. Referenced 6 time(s) in page(s) 33, 134, 145, 157, 170, and 188.

PEREIRA, R. M.; COSTA, Y. M. G.; SILLA JR, C. N. MLTL: A multi-label approach for the tomek link undersampling algorithm. *Neurocomputing*, Elsevier, v. 383, p. 95–105, 2020. Referenced 2 time(s) in page(s) 33 and 110.

PEREIRA, R. M.; SILLA JR, C. N. Using simplified chords sequences to classify songs genres. In: *Proceedings of IEEE International Conference on Multimedia and Expo*. Hong Kong: IEEE, 2017. p. 1446–1451. Referenced 2 time(s) in page(s) 33 and 68.

PUNERA, K.; GHOSH, J. Enhanced hierarchical classification via isotonic smoothing. In: *Proceedings of the International Conference on World Wide Web*. Beijing, China: ACM, 2008. p. 151–160. Referenced in page(s) 29.

QUINLAN, J. R. C4.5: Programs for machine learning. Morgan Kaufmann, p. 629, 1993. Referenced in page(s) 41.

RAMENTOL, E.; CABALLERO, Y.; BELLO, R.; HERRERA, F. Smote-rsb*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory. *Knowledge and Information Systems*, Springer, v. 33, n. 2, p. 245–265, 2012. Referenced 2 time(s) in page(s) 8 and 85.

RAMÍREZ-CORONA, M.; SUCAR, L. E.; MORALES, E. F. Hierarchical multilabel classification based on path evaluation. *International Journal of Approximate Reasoning*, Elsevier, v. 68, p. 179–193, 2016. Referenced in page(s) 55.

READ, J.; PFAHRINGER, B.; HOLMES, G.; FRANK, E. Classifier chains for multi-label classification. In: *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Bled, Slovenia: Springer, 2009. p. 254–269. Referenced 2 time(s) in page(s) 45 and 117.

REMUZZI, A.; REMUZZI, G. COVID-19 and italy: what next? *The Lancet*, Elsevier, 2020. Referenced in page(s) 213.

RIFKIN, R.; KLAUTAU, A. In defense of one-vs-all classification. *Journal of Machine Learning Research*, v. 5, n. Jan, p. 101–141, 2004. Referenced in page(s) 73.

RIVERA, A. R.; CASTILLO, J. R.; CHAE, O. O. Local directional number pattern for face analysis: Face and expression recognition. *IEEE Transactions on Image Processing*, v. 22, n. 5, p. 1740–1752, 2013. Referenced in page(s) 223.

ROY, S. B.; TEREDESAI, A.; ZOLFAGHAR, K.; LIU, R.; HAZEL, D.; NEWMAN, S.; MARINEZ, A. Dynamic hierarchical classification for patient risk-of-readmission. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Sydney, Australia: ACM, 2015. p. 1691–1700. Referenced in page(s) 29.

RUEPP, A.; ZOLLNER, A.; MAIER, D.; ALBERMANN, K.; HANI, J.; MOKREJS, M.; TETKO, I.; GÜLDENER, U.; MANNHAUPT, G.; MÜNSTERKÖTTER, M. The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research*, Oxford University Press, v. 32, n. 18, p. 5539–5545, 2004. Referenced in page(s) 155.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. Springer, p. 366–400, 1988. Referenced in page(s) 28.

SCALCO, E.; RIZZO, G. Texture analysis of medical images for radiotherapy applications. *The British Journal of Radiology*, The British Institute of Radiology., v. 90, n. 1070, p. 20160642, 2017. Referenced 2 time(s) in page(s) 216 and 220.

SELF, W. H.; COURTNEY, D. M.; MCNAUGHTON, C. D.; WUNDERINK, R. G.; KLINE, J. A. High discordance of chest x-ray and computed tomography for detection of pulmonary opacities in ed patients: implications for diagnosing pneumonia. *The American journal of emergency medicine*, Elsevier, v. 31, n. 2, p. 401–405, 2013. Referenced 2 time(s) in page(s) 211 and 213.

SHARMA, S.; BELLINGER, C.; KRAWCZYK, B.; ZAIANE, O.; JAPKOWICZ, N. Synthetic oversampling with the majority class: A new perspective on handling extreme imbalance. In: *Proceedings of the IEEE International Conference on Data Mining*. Singapore: IEEE, 2018. p. 447–456. Referenced in page(s) 186.

SILLA JR, C. N. COVID-19: estudo desenvolve diagnóstico por raio-x. *CBN Curitiba*, CBN, 2020. Available from Internet: https://bit.ly/cbn-covid19. Referenced in page(s) 36.

SILLA JR, C. N. Estudo com inteligência artificial para diagnóstico da COVID-19 em raio-x. *Light News*, Transamerica Light, 2020. Available from Internet: https://bit.ly/lightnews-covid19. Referenced in page(s) 36.

SILLA JR, C. N. Imagens de raio-x no diagnóstico da COVID-19. *Meio Dia Paraná*, RPC TV, 2020. Available from Internet: https://bit.ly/rpc-covid19. Referenced in page(s) 36.

SILLA JR, C. N. Radiografia inteligente. *Com as armas do conhecimento*, SuperAcesso, 2020. Available from Internet: https://bit.ly/3r12JLq. Referenced in page(s) 36.

SILLA JR, C. N.; FREITAS, A. A. A global-model naive bayes approach to the hierarchical prediction of protein functions. In: *Proceedings of the IEEE International Conference on Data Mining*. Miami, USA: IEEE, 2009. p. 992–997. Referenced in page(s) 60.

SILLA JR, C. N.; FREITAS, A. A. Novel top-down approaches for hierarchical classification and their application to automatic music genre classification. In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*. San Antonio, USA: IEEE, 2009. p. 3499–3504. Referenced 2 time(s) in page(s) 29 and 55.

SILLA JR, C. N.; FREITAS, A. A. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, v. 22, n. 1-2, p. 31–72, 2011. Referenced 17 time(s) in page(s) 7, 28, 29, 49, 50, 51, 52, 53, 54, 144, 149, 169, 176, 185, 186, 188, and 214.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Referenced in page(s) 225.

SNOEK, C. G. M.; WORRING, M.; SMEULDERS, A. W. M. Early versus late fusion in semantic video analysis. In: *Proceedings of the ACM International Conference on Multimedia*. Singapore: ACM, 2005. p. 399–402. Referenced in page(s) 225.

SOKOLOVA, M.; JAPKOWICZ, N.; SZPAKOWICZ, S. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In: *Australasian Joint Conference on Artificial Intelligence*. Hobart, Australia: ACM, 2006. p. 1015–1021. Referenced in page(s) 159.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, v. 45, n. 4, p. 427–437, 2009. Referenced in page(s) 48.

SONG, T.; LI, H.; MENG, F.; WU, Q.; CAI, J. LETRIST: Locally encoded transform feature histogram for rotation-invariant texture classification. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 28, n. 7, p. 1565–1579, 2018. Referenced in page(s) 223.

SPYROMITROS, E.; TSOUMAKAS, G.; VLAHAVAS, I. An empirical study of lazy multilabel classification algorithms. In: *Proceedings of the Hellenic Conference on Artificial Intelligence*. Syros, Greece: Springer, 2008. p. 401–406. Referenced 2 time(s) in page(s) 46 and 117.

SRIVASTAVA, A. N.; ZANE-ULMAN, B. Discovering recurring anomalies in text reports regarding complex space systems. In: *Proceedings of the IEEE Aerospace Conference*. Big Sky, MT, USA: IEEE, 2005. p. 3853–3862. Referenced in page(s) 99.

STANFILL, C.; WALTZ, D. Toward memory-based reasoning. *Communications of the ACM*, ACM, v. 29, n. 12, p. 1213–1228, 1986. Referenced 2 time(s) in page(s) 106 and 190.

STEFANOWSKI, J.; WILK, S. Selective pre-processing of imbalanced data for improving classification performance. In: *Proceedings of the International Conference on Data Warehousing and Knowledge Discovery*. Turin, Italy: Springer, 2008. p. 283–292. Referenced 3 time(s) in page(s) 71, 90, and 92.

SUN, A.; LIM, E.-P. Hierarchical text classification and evaluation. In: *Proceedings of the IEEE International Conference on Data Mining*. San Jose, CA, USA: IEEE, 2001. p. 521–528. Referenced in page(s) 66.

SUN, Y.; KAMEL, M. S.; WONG, A. K.; WANG, Y. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, Elsevier, v. 40, n. 12, p. 3358–3378, 2007. Referenced in page(s) 71.

SURVEILLANCES, V. The epidemiological characteristics of an outbreak of 2019 novel coronavirus diseases (COVID-19) in china. *China CDC Weekly*, v. 8, n. 2, p. 113–122, 2020. Referenced in page(s) 213.

SZALKAI, B.; GROLMUSZ, V.; HANCOCK, J. Seclaf: A webserver and deep neural network design tool for hierarchical biological sequence classification. *Bioinformatics*, v. 1, p. 3, 2018. Referenced in page(s) 29.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Boston, USA: IEEE, 2015. p. 1–9. Referenced in page(s) 225.

SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J.; WOJNA, Z. Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, USA: IEEE, 2016. p. 2818–2826. Referenced in page(s) 224.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to Data Mining*. 1. ed. India: Pearson, 2005. 769 p. Referenced 3 time(s) in page(s) 7, 40, and 41.

TING, K. M. An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 14, n. 3, p. 659–665, 2002. Referenced in page(s) 71.

TIPTON, K. F. Nomenclature committee of the international union of biochemistry and molecular biology. *European Journal of Biochemistry*, v. 223, n. 1, p. 1, 1994. Referenced in page(s) 29.

TODOROVSKI, L.; BLOCKEEL, H.; DZEROSKI, S. Ranking with predictive clustering trees. In: *Proceedings of the European Conference on Machine Learning*. Helsinki, Finland: ECML, 2002. p. 444–455. Referenced in page(s) 139.

TOLKSDORF, K.; BUDA, S.; SCHULER, E.; WIELER, L. H.; HAAS, W. Influenza-associated pneumonia as reference to assess seriousness of coronavirus disease (COVID-19). *Eurosurveillance*, v. 25, n. 11, 2020. Referenced in page(s) 211.

TOMEK, I. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, n. 6, p. 448–452, 1976. Referenced 5 time(s) in page(s) 79, 81, 82, 84, and 110.

TROHIDIS, K.; TSOUMAKAS, G.; KALLIRIS, G.; VLAHAVAS, I. Multi-label classification of music into emotions. In: *Proceedings of The International Society for Music Information Retrieval Conference*. Philadelphia, USA: ISMIR, 2008. p. 325–330. Referenced 3 time(s) in page(s) 117, 177, and 201.

TSOUMAKAS, G.; KATAKIS, I. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, v. 3, n. 3, 2006. Referenced 3 time(s) in page(s) 28, 43, and 44.

TSOUMAKAS, G.; KATAKIS, I.; VLAHAVAS, I. Effective and efficient multilabel classification in domains with large number of labels. In: *Proceedings of the Workshop on Mining Multidimensional Data*. Montpellier, France: ACM, 2008. p. 30–44. Referenced 2 time(s) in page(s) 45 and 117.

TSOUMAKAS, G.; KATAKIS, I.; VLAHAVAS, I. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, v. 23, n. 7, p. 1079–1089, 2011. Referenced 2 time(s) in page(s) 46 and 117.

TSOUMAKAS, G.; VLAHAVAS, I. Random k-labelsets: An ensemble method for multilabel classification. *Machine Learning: ECML 2007*, p. 406–417, 2007. Referenced 2 time(s) in page(s) 117 and 156.

TSOUMAKAS, G.; XIOUFIS, E.; VILCEK, J.; VLAHAVAS, I. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, v. 12, p. 2411–2414, July 2011. Referenced 2 time(s) in page(s) 34 and 117.

TURNBULL, D.; BARRINGTON, L.; TORRES, D.; LANCKRIET, G. Towards musical query-by-semantic-description using the CAL500 dataset. In: *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Amsterdam, The Netherlands: ACM, 2007. p. 439–446. Referenced 2 time(s) in page(s) 177 and 201.

TURNBULL, D.; BARRINGTON, L.; TORRES, D.; LANCKRIET, G. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech, and Language Processing*, v. 16, n. 2, p. 467–476, 2008. Referenced 2 time(s) in page(s) 99 and 116.

VALENTE, J. Universidades desenvolvem apoio a diagnóstico de COVID-19 com raio-x. *Saúde*, Agencia Brasil, 2020. Available from Internet: https://bit.ly/agenciabrasil-covid19. Referenced in page(s) 36.

VALENTINI, G. True path rule hierarchical ensembles. In: *Proceedings of the International Workshop on Multiple Classifier Systems*. Reykjavik, Iceland: Springer, 2009. p. 232–241. Referenced in page(s) 54.

VALERIO, V. D.; PEREIRA, R. M.; COSTA, Y. M. G.; BERTOINI, D.; SILLA JR, C. N. A resampling approach for imbalanceness on music genre classification using spectrograms. In: *Proceedings of the International Florida Artificial Intelligence Conference*. Melbourne, USA: AAAI, 2018. Referenced 2 time(s) in page(s) 34 and 69.

VAPNIK, V. N.; CHERVONENKIS, A. Y. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, v. 16, n. 2, p. 283–305, 1971. Referenced in page(s) 42.

VENS, C.; STRUYF, J.; SCHIETGAT, L.; DŽEROSKI, S.; BLOCKEEL, H. Decision trees for hierarchical multi-label classification. *Journal of Machine Learning*, v. 73, n. 2, p. 185, 2008. Referenced 2 time(s) in page(s) 61 and 139.

WANG, K.; ZHOU, S.; HE, Y. Hierarchical classification of real life documents. In: *Proceedings of the SIAM International Conference on Data Mining*. Chicago, USA: SIAM, 2001. p. 1–16. Referenced in page(s) 58.

WANG, L.; WONG, A. COVID-Net: A tailored deep convolutional neural network design for detection of COVID-19 cases from chest radiography images. *arXiv preprint arXiv:2003.09871*, 2020. Referenced 2 time(s) in page(s) 218 and 220.

WANG, S.; YAO, X. Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 42, n. 4, p. 1119–1130, 2012. Referenced in page(s) 72.

WANG, X.; PENG, Y.; LU, L.; LU, Z.; BAGHERI, M.; SUMMERS, R. M. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Honolulu, USA: IEEE, 2017. p. 2097–2106. Referenced 2 time(s) in page(s) 212 and 229.

WEHRMANN, J.; CERRI, R.; BARROS, R. Hierarchical multi-label classification networks. In: *Proceedings of the International Conference on Machine Learning*. Stockholm, Sweden: ACM, 2018. p. 5225–5234. Referenced 4 time(s) in page(s) 129, 139, 178, and 202.

WILSON, D. L. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, IEEE, n. 3, p. 408–421, 1972. Referenced in page(s) 79.

WOLPERT, D. H.; MACREADY, W. G. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 1, n. 1, p. 67–82, 1997. Referenced in page(s) 38.

WOODS, K. S.; DOSS, C. C.; BOWYER, K. W.; SOLKA, J. L.; PRIEBE, C. E.; JR, W. P. K. Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography. *International Journal of Pattern Recognition and Artificial Intelligence*, World Scientific, v. 7, n. 06, p. 1417–1436, 1993. Referenced in page(s) 70.

WRIGHT, K. An ai assist for spotting COVID-19 in x-rays. *Physics*, American Physical Society, v. 13, n. 73, 2020. Available from Internet: https://bit.ly/physics-covid19. Referenced in page(s) 36.

WU, F.; ZHANG, J.; HONAVAR, V. Learning classifiers using hierarchically structured class taxonomies. In: *Proceedings of the International Symposium on Abstraction, Reformulation, and Approximation*. Airth Castle, Scotland, UK: Springer, 2005. p. 313–320. Referenced 2 time(s) in page(s) 29 and 49.

XU, C.; GENG, X. Hierarchical classification based on label distribution learning. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Honolulu, USA: AAAI, 2019. p. 5533–5540. Referenced in page(s) 30.

YEN, S.-J.; LEE, Y.-S. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, Elsevier, v. 36, n. 3, p. 5718–5727, 2009. Referenced 2 time(s) in page(s) 80 and 81.

ZADROZNY, B.; ELKAN, C. Learning and making decisions when costs and probabilities are both unknown. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, USA: ACM, 2001. p. 204–213. Referenced in page(s) 71.

ZADROZNY, B.; LANGFORD, J.; ABE, N. Cost-sensitive learning by cost-proportionate example weighting. In: *Proceedings of the IEEE International Conference on Data Mining*. Melbourne, FL, USA: IEEE, 2003. p. 435–442. Referenced in page(s) 71.

ZENG, X.; YANG, C.; TU, C.; LIU, Z.; SUN, M. Chinese liwc lexicon expansion via hierarchical classification of word embeddings with sememe attention. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. New Orleans, USA: AAAI, 2018. Referenced in page(s) 54.

ZHANG, L.; SHAH, S.; KAKADIARIS, I. Hierarchical multi-label classification using fully associative ensemble learning. *Pattern Recognition*, Elsevier, v. 70, p. 89–103, 2017. Referenced in page(s) 54.

ZHANG, M.-L.; ZHI-HUA, Z. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, v. 40, n. 7, p. 2038–2048, 2007. Referenced 2 time(s) in page(s) 46 and 117.

ZHANG, M.-L.; ZHOU, Z.-H. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, v. 26, n. 8, p. 1819–1837, 2014. Referenced in page(s) 43.

ZHANG, S.; ZHANG, C.; YANG, Q. Data preparation for data mining. *Applied Artificial Intelligence*, Taylor & Francis, v. 17, n. 5-6, p. 375–381, 2003. Referenced in page(s) 37.

ZHAO, H. Instance weighting versus threshold adjusting for cost-sensitive classification. *Knowledge and Information Systems*, Springer, v. 15, n. 3, p. 321–334, 2008. Referenced in page(s) 71.

ZHOU, M.; CHEN, Y.; WANG, D.; XU, Y.; YAO, W.; HUANG, J.; JIN, X.; PAN, Z.; TAN, J.; WANG, L. Improved deep learning model for differentiating novel coronavirus pneumonia and influenza pneumonia. *medRxiv*, Cold Spring Harbor Laboratory Press, 2020. Referenced 3 time(s) in page(s) 216, 217, and 220.

ZHOU, Z.-H.; LIU, X.-Y. On multi-class cost-sensitive learning. *Computational Intelligence*, Wiley Online Library, v. 26, n. 3, p. 232–257, 2010. Referenced in page(s) 71.

Appendix

APPENDIX

A Experimental Results of Local Classifiers with Resampling

In the appendix we present all the Tables of classification and metrics results generated in the experiments of Chapter 7, which were summarized into charts in the Chapter. Besides the raw results we also present here the Tables of the statistics, which were applied over the results in order to give statistical background in the responses of the Analysis and Discussion section. We have subdivided this Appendix in two Sections: (i) Classification Results; and (ii) Statistical Tests.

A.1 Classification Results

In this sections we present the tables with the classification results of the main experiments proposed in Chapter 7.

Table 88 – F-Score results for the proposed approaches in the Cell-cycle dataset.

	Proposed Approach with LCN						Proposed Approach with LCPN		Proposed Approach with LCL
Resampling Algorithm	ES	E	I	LE	LI	S	ES	S	
ROS	0.1372	0.1298	0.1409	0.1183	0.1181	0.1190	0.1074	0.1118	0.0872
SMOTE	0.1542	0.1397	0.1417	0.1422	0.1445	0.1357	0.1147	0.1328	0.0989
SMOTE-B1	0.1427	0.1323	0.1449	0.1453	0.1352	0.1403	0.1106	0.1155	0.0857
SMOTE-B2	0.1413	0.1298	0.1487	0.1416	0.1310	0.1337	0.1013	0.1155	0.0975
ADASYN	0.1532	0.1033	0.1512	0.1424	0.1427	0.1388	0.0759	0.1303	0.0552
RUS	0.1365	0.1017	0.1422	0.1418	0.1112	0.1072	0.0812	0.1143	0.0692
CC	0.1551	0.1058	0.1476	0.1382	0.1067	0.1065	0.0804	0.1307	0.0638
CNEN	0.1613	0.1173	0.1515	0.1324	0.1312	0.1287	0.0924	0.1367	0.0740
ENN	0.1548	0.1359	0.1591	0.1401	0.1329	0.1190	0.1097	0.1264	0.1015
RENN	0.1512	0.1212	0.1507	0.1389	0.1317	0.1324	0.0922	0.1302	0.0768
AllKNN	0.1489	0.0947	0.1480	0.1120	0.1294	0.1313	0.0659	0.1198	0.0631
NM1	0.0966	0.0822	0.0948	0.0928	0.0985	0.0998	0.0536	0.0766	0.0335
NM2	0.0919	0.0898	0.0875	0.0987	0.0999	0.0971	0.0691	0.0646	0.0590
NM3	0.0910	0.0869	0.1025	0.0948	0.1002	0.0928	0.0660	0.0672	0.0513
TL	0.1420	0.1353	0.1460	0.1124	0.0826	0.1084	0.1066	0.1199	0.0961
SMOTE+ENN	0.1708	0.1342	0.1791	0.1438	0.1361	0.1433	0.1135	0.1476	0.1004
SMOTE+TL	0.1609	0.1302	0.1679	0.1342	0.1406	0.1324	0.1045	0.1355	0.0931

Table 89 – F-Score results for the proposed approaches in the Eisen dataset.

	Proposed Approach with LCN						Proposed Approach with LCPN		Proposed Approach with LCL
Resampling Algorithm	ES	E	I	LE	LI	S	ES	S	
ROS	0.0976	0.1039	0.1069	0.1044	0.1212	0.0935	0.0816	0.0754	0.0669
SMOTE	0.1151	0.1379	0.1433	0.1393	0.1520	0.1174	0.1156	0.0935	0.1003
SMOTE-B1	0.1090	0.1433	0.1327	0.1418	0.1561	0.1085	0.1173	0.0830	0.0987
SMOTE-B2	0.1079	0.1402	0.1414	0.1444	0.1438	0.1133	0.1107	0.0824	0.1094
ADASYN	0.1152	0.1407	0.1406	0.1449	0.1458	0.1099	0.1190	0.0937	0.1002
RUS	0.0961	0.1119	0.1209	0.1156	0.1183	0.1010	0.0912	0.0709	0.0768
CC	0.1171	0.1164	0.1232	0.1124	0.1294	0.1113	0.0941	0.0944	0.0682
CNEN	0.1211	0.1213	0.1376	0.1219	0.1445	0.1130	0.0959	0.1011	0.0746
ENN	0.1209	0.1310	0.1311	0.1395	0.1415	0.1124	0.1108	0.0996	0.0900
RENN	0.1212	0.1213	0.1286	0.1250	0.1338	0.1088	0.1004	0.0956	0.0821
AllKNN	0.1201	0.1042	0.1145	0.1129	0.1395	0.0993	0.0828	0.0901	0.0711
NM1	0.0965	0.0824	0.0919	0.0889	0.1021	0.0790	0.0600	0.0763	0.0448
NM2	0.0898	0.0876	0.0973	0.0890	0.1018	0.0772	0.0668	0.0607	0.0547
NM3	0.0868	0.0887	0.0920	0.0849	0.1060	0.0764	0.0589	0.0640	0.0584
TL	0.1011	0.0992	0.1146	0.0928	0.0878	0.1017	0.0724	0.0810	0.0598
SMOTE+ENN	0.1291	0.1118	0.1573	0.1483	0.1477	0.1138	0.0836	0.1031	0.0748
SMOTE+TL	0.1248	0.1111	0.1499	0.1389	0.1522	0.1224	0.0818	0.0960	0.0679

Table 90 – F-Score results for the proposed approaches in the Exp dataset.

	Proposed Approach with LCN						Proposed Approach with LCPN		Proposed Approach with LCL
Resampling Algorithm	ES	E	I	LE	LI	S	ES	S	
ROS	0.1280	0.1187	0.1325	0.1074	0.1118	0.1087	0.0892	0.1009	0.0741
SMOTE	0.1324	0.1372	0.1424	0.1059	0.1491	0.1120	0.1110	0.1061	0.0903
SMOTE-B1	0.1321	0.1365	0.1385	0.1090	0.1523	0.1114	0.1153	0.1061	0.0892
SMOTE-B2	0.1313	0.1313	0.1349	0.1070	0.1421	0.1098	0.1032	0.1056	0.0832
ADASYN	0.1336	0.1275	0.1413	0.1046	0.1442	0.1129	0.1030	0.1122	0.0793
RUS	0.1241	0.1210	0.1257	0.0986	0.1151	0.1073	0.0994	0.1023	0.0842
CC	0.1371	0.1223	0.1498	0.1099	0.1221	0.1012	0.0953	0.1135	0.0816
CNEN	0.1412	0.1407	0.1454	0.1125	0.1313	0.1038	0.1127	0.1147	0.1020
ENN	0.1379	0.1428	0.1425	0.1120	0.1403	0.1176	0.1206	0.1095	0.1075
RENN	0.1311	0.1384	0.1329	0.1098	0.1314	0.1035	0.1132	0.1092	0.1068
AllKNN	0.1202	0.1290	0.1356	0.1022	0.1368	0.0931	0.1084	0.0993	0.0862
NM1	0.1057	0.0975	0.0904	0.0882	0.1001	0.0828	0.0729	0.0794	0.0507
NM2	0.0873	0.0916	0.0889	0.0868	0.1009	0.0837	0.0658	0.0670	0.0487
NM3	0.0851	0.0914	0.0885	0.0857	0.1047	0.0893	0.0641	0.0620	0.0512
TL	0.1301	0.1178	0.1303	0.0999	0.1035	0.1094	0.0885	0.1086	0.0877
SMOTE+ENN	0.1420	0.1391	0.1678	0.1229	0.1412	0.1184	0.1099	0.1159	0.1007
SMOTE+TL	0.1308	0.1322	0.1537	0.1199	0.1503	0.1179	0.1078	0.1099	0.0904

Table 91 – F-Score results for the proposed approaches in the FMA MFCC dataset.

	Proposed Approach with LCN						Proposed Approach with LCPN		Proposed Approach with LCL
Resampling Algorithm	ES	E	I	LE	LI	S	ES	S	
ROS	0.1313	0.1508	0.1868	0.1335	0.2590	0.1573	0.1218	0.1023	0.0767
SMOTE	0.1412	0.1609	0.2067	0.1537	0.2265	0.1723	0.1391	0.1114	0.0862
SMOTE-B1	0.1301	0.1561	0.2094	0.1411	0.2198	0.1989	0.1288	0.1096	0.0810
SMOTE-B2	0.1290	0.1489	0.1929	0.1706	0.1771	0.1824	0.1277	0.1019	0.0811
ADASYN	0.1312	0.1433	0.1923	0.1573	0.2297	0.1612	0.1184	0.1104	0.0804
RUS	0.1272	0.1312	0.1637	0.1506	0.1637	0.1463	0.1074	0.1006	0.0760
CC	0.1322	0.1399	0.1886	0.1574	0.1633	0.1389	0.1169	0.1027	0.0831
CNEN	0.1418	0.1463	0.2037	0.1524	0.1437	0.1483	0.1205	0.1158	0.0702
ENN	0.1299	0.1497	0.2173	0.1533	0.2529	0.2187	0.1278	0.1034	0.0629
RENN	0.1219	0.1512	0.1878	0.1447	0.2393	0.1987	0.1274	0.0993	0.0817
AllKNN	0.1189	0.1372	0.1740	0.1298	0.2317	0.1683	0.1092	0.0907	0.0641
NM1	0.0978	0.0874	0.1174	0.1072	0.1334	0.1114	0.0617	0.0739	0.0484
NM2	0.0890	0.0961	0.1124	0.1024	0.1214	0.1073	0.0698	0.0677	0.0507
NM3	0.0877	0.0915	0.1030	0.1018	0.1376	0.1048	0.0651	0.0621	0.0421
TL	0.1312	0.1489	0.1433	0.1194	0.2190	0.1289	0.1272	0.1103	0.0837
SMOTE+ENN	0.1511	0.1533	0.2373	0.1536	0.2924	0.2249	0.1274	0.1266	0.0865
SMOTE+TL	0.1413	0.1444	0.2133	0.1673	0.2791	0.2359	0.1207	0.1208	0.0974

Table 92 – F-Score results for the proposed approaches in the Gasch-1 dataset.

	Proposed Approach with LCN						Proposed Approach with LCPN		Proposed Approach with LCL
Resampling Algorithm	ES	E	I	LE	LI	S	ES	S	
ROS	0.1092	0.1033	0.1228	0.1091	0.1440	0.1114	0.0791	0.0801	0.0650
SMOTE	0.1132	0.1087	0.1283	0.1128	0.1685	0.1172	0.0855	0.0867	0.0746
SMOTE-B1	0.1098	0.1063	0.1250	0.1077	0.1614	0.1224	0.0830	0.0805	0.0637
SMOTE-B2	0.1090	0.1090	0.1227	0.1130	0.1587	0.1187	0.0882	0.0805	0.0614
ADASYN	0.1122	0.1112	0.1338	0.1129	0.1656	0.1167	0.0900	0.0844	0.0662
RUS	0.1071	0.1077	0.1118	0.1078	0.1341	0.1014	0.0868	0.0781	0.0619
CC	0.1113	0.1113	0.1299	0.1099	0.1380	0.1017	0.0858	0.0844	0.0679
CNEN	0.1212	0.1214	0.1324	0.1129	0.1282	0.1117	0.0975	0.0990	0.0822
ENN	0.1157	0.1232	0.1374	0.1187	0.1498	0.1174	0.1025	0.0944	0.0853
RENN	0.1118	0.1222	0.1389	0.1139	0.1551	0.1098	0.0978	0.0858	0.0769
AllKNN	0.1109	0.1085	0.1289	0.1033	0.1449	0.1133	0.0884	0.0843	0.0675
NM1	0.1071	0.0822	0.0924	0.0918	0.1022	0.0913	0.0561	0.0846	0.0334
NM2	0.0919	0.0823	0.0917	0.0889	0.1025	0.0899	0.0578	0.0621	0.0459
NM3	0.0901	0.0828	0.0938	0.0878	0.1046	0.0896	0.0597	0.0681	0.0410
TL	0.1109	0.1023	0.1168	0.1109	0.1043	0.1082	0.0788	0.0835	0.0542
SMOTE+ENN	0.1272	0.1274	0.1546	0.1219	0.1718	0.1276	0.0987	0.1037	0.0851
SMOTE+TL	0.1110	0.1177	0.1462	0.1224	0.1654	0.1294	0.0911	0.0820	0.0751

Table 93 – F-Score results for the proposed approaches in the CLEF dataset.

	Proposed Approach with LCN						Proposed Approach with LCPN		Proposed Approach with LCL
Resampling Algorithm	ES	E	I	LE	LI	S	ES	S	
ROS	0.6902	0.6618	0.7395	0.5799	0.7577	0.6163	0.5711	0.5716	0.4931
SMOTE	0.7153	0.6948	0.7590	0.5953	0.8070	0.6322	0.5905	0.5956	0.5059
SMOTE-B1	0.7147	0.6871	0.7591	0.5880	0.8056	0.6228	0.5868	0.5921	0.5216
SMOTE-B2	0.7192	0.6731	0.7547	0.5858	0.8030	0.6287	0.5800	0.5849	0.5071
ADASYN	0.6733	0.6660	0.7151	0.5553	0.7521	0.6163	0.5483	0.5547	0.4769
RUS	0.6815	0.6551	0.7283	0.5690	0.7437	0.5995	0.5614	0.5684	0.4743
CC	0.6889	0.6550	0.7238	0.5838	0.7616	0.6144	0.5827	0.5893	0.4976
CNEN	0.6912	0.6769	0.7446	0.6042	0.7782	0.6172	0.5987	0.6060	0.5123
ENN	0.6960	0.6800	0.7362	0.6005	0.7790	0.6302	0.5921	0.6009	0.5409
RENN	0.6795	0.6510	0.7206	0.5810	0.7569	0.6021	0.5795	0.5859	0.5337
AllKNN	0.6870	0.6560	0.7240	0.5946	0.7517	0.6252	0.5868	0.5940	0.5020
NM1	0.6704	0.6567	0.7192	0.5424	0.7324	0.5965	0.5370	0.5416	0.4494
NM2	0.6759	0.6508	0.7086	0.5432	0.7313	0.5913	0.5351	0.5367	0.4499
NM3	0.6639	0.6500	0.7138	0.5593	0.7276	0.5782	0.5588	0.5685	0.4437
TL	0.7094	0.6718	0.7478	0.5993	0.7538	0.6238	0.5967	0.5974	0.5259
SMOTE+ENN	0.7186	0.6921	0.7520	0.6008	0.7801	0.6372	0.5965	0.6017	0.5093
SMOTE+TL	0.7376	0.7029	0.7725	0.6119	0.8687	0.6548	0.6085	0.6123	0.5239

Table 94 – F-Score results for the proposed approaches in the DMOZ-2010 dataset.

	Proposed Approach with LCN						Proposed Approach with LCPN		Proposed Approach with LCL
Resampling Algorithm	ES	E	I	LE	LI	S	ES	S	
ROS	0.3299	0.3100	0.3504	0.3139	0.3980	0.3230	0.3211	0.3310	0.2468
SMOTE	0.3501	0.3318	0.3652	0.3352	0.4139	0.3436	0.3392	0.3394	0.2472
SMOTE-B1	0.3513	0.3379	0.3660	0.3327	0.4189	0.3442	0.3377	0.3450	0.2517
SMOTE-B2	0.3551	0.3356	0.3646	0.3297	0.4167	0.3311	0.3312	0.3401	0.2687
ADASYN	0.3108	0.3090	0.3272	0.2801	0.3868	0.3006	0.2831	0.2899	0.2026
RUS	0.3378	0.3232	0.3385	0.3049	0.3932	0.3204	0.3069	0.3129	0.2132
CC	0.3424	0.3240	0.3412	0.3183	0.3999	0.3220	0.3216	0.3238	0.2369
CNEN	0.3494	0.3190	0.3470	0.3086	0.4018	0.3127	0.3131	0.3221	0.2698
ENN	0.3455	0.3183	0.3435	0.3065	0.4015	0.3165	0.3138	0.3214	0.2323
RENN	0.3378	0.3162	0.3369	0.2976	0.3937	0.3180	0.3074	0.3152	0.2236
AllKNN	0.3305	0.3162	0.3232	0.2882	0.3871	0.3173	0.2951	0.3024	0.2035
NM1	0.3125	0.3053	0.3167	0.2807	0.3754	0.2778	0.2842	0.2859	0.2188
NM2	0.3063	0.2914	0.3339	0.2818	0.3745	0.2760	0.2826	0.2863	0.1891
NM3	0.3113	0.2928	0.3208	0.2783	0.3785	0.2894	0.2799	0.2800	0.2138
TL	0.3404	0.3271	0.3410	0.3246	0.4003	0.3243	0.3311	0.3391	0.2690
SMOTE+ENN	0.3501	0.3512	0.3898	0.3415	0.4387	0.3369	0.3504	0.3602	0.3019
SMOTE+TL	0.3670	0.3425	0.3693	0.3273	0.4210	0.3130	0.3358	0.3397	0.2779

Table 95 – F-Score results for the proposed approaches in the LSHTC-small dataset.

	Proposed Approach with LCN						Proposed Approach with LCPN		Proposed Approach with LCL
Resampling Algorithm	ES	E	I	LE	LI	S	ES	S	
ROS	0.3596	0.3335	0.3775	0.3510	0.4180	0.3537	0.3513	0.3544	0.2542
SMOTE	0.3713	0.3651	0.3868	0.3741	0.4471	0.3642	0.3788	0.3822	0.2917
SMOTE-B1	0.3805	0.3652	0.4027	0.3560	0.4549	0.3764	0.3639	0.3709	0.3074
SMOTE-B2	0.3920	0.3681	0.4013	0.3529	0.4367	0.3663	0.3546	0.3613	0.2612
ADASYN	0.3480	0.3463	0.3500	0.3099	0.4099	0.3369	0.3181	0.3240	0.2617
RUS	0.3715	0.3593	0.3654	0.3430	0.4187	0.3544	0.3517	0.3542	0.2566
CC	0.3643	0.3612	0.3702	0.3416	0.4357	0.3573	0.3467	0.3540	0.2761
CNEN	0.3713	0.3457	0.3691	0.3485	0.4245	0.3461	0.3573	0.3655	0.2858
ENN	0.3713	0.3447	0.3822	0.3309	0.4339	0.3369	0.3364	0.3445	0.2592
RENN	0.3618	0.3556	0.3616	0.3338	0.4325	0.3394	0.3348	0.3407	0.2845
AllKNN	0.3555	0.3449	0.3587	0.3086	0.4169	0.3493	0.3174	0.3185	0.2483
NM1	0.3378	0.3445	0.3397	0.3179	0.4051	0.3157	0.3204	0.3269	0.2607
NM2	0.3372	0.3200	0.3637	0.3163	0.4007	0.3013	0.3186	0.3281	0.2217
NM3	0.3384	0.3280	0.3525	0.3145	0.4128	0.3105	0.3229	0.3238	0.2634
TL	0.3720	0.3582	0.3633	0.3514	0.4245	0.3479	0.3574	0.3639	0.3055
SMOTE+ENN	0.3862	0.3816	0.4208	0.3658	0.4559	0.3529	0.3748	0.3725	0.2805
SMOTE+TL	0.3997	0.3722	0.4006	0.3493	0.4732	0.3480	0.3558	0.3603	0.2576

Table 96 – F-Score results for the Top-Down (TD) approaches in all datasets.

	LCN						LCPN		LCL
Dataset	ES	E	I	LE	LI	S	ES	S	
Cell-cycle	0.1405	0.1374	0.1459	0.1152	0.1448	0.1206	0.1105	0.1148	0.0836
Eisen	0.0979	0.0932	0.1141	0.0937	0.1142	0.1024	0.0649	0.0736	0.0547
Exp	0.1271	0.1176	0.1312	0.1060	0.1157	0.1087	0.0934	0.1054	0.0802
FMA MFCC	0.1290	0.1489	0.1723	0.1160	0.1950	0.1378	0.1246	0.1059	0.0676
Gasch-1	0.1086	0.1021	0.1150	0.1088	0.1094	0.1089	0.0742	0.0828	0.0662
CLEF	0.6812	0.6698	0.7306	0.5602	0.7411	0.6133	0.5539	0.5591	0.4758
DMOZ-2010	0.3183	0.3067	0.3363	0.2879	0.3849	0.2954	0.2883	0.2982	0.2482
LSHTC-small	0.3468	0.3270	0.3702	0.3163	0.4112	0.3252	0.3224	0.3306	0.2533

Table 97 – F-Score results for the Flat-ML approach in all datasets.

Dataset	F-Score
Cell-cycle	0.0781
Eisen	0.0971
Exp	0.0444
FMA MFCC	0.1408
Gasch-1	0.0697
CLEF	0.5572
DMOZ-2010	0.2350
LSHTC-small	0.2774

Table 98 – F-Score results for the Flat-MLRS approach with all datasets.

	LPROS	LPRUS	MLROS	MLRUS	MLeNN	REMEDIAL	MLSMOTE	MLTL	MLSMOTE+MLTL
Cell-cycle	0.0796	0.0888	0.0792	0.0834	0.0776	0.0792	0.0794	0.0787	0.0779
Eisen	0.0966	0.1015	0.0970	0.0806	0.0923	0.0969	0.0980	0.0831	0.0918
Exp	0.0427	0.0423	0.0402	0.0406	0.0379	0.0406	0.0431	0.0390	0.0449
FMA MFCC	0.1408	0.1385	0.1413	0.1319	0.1407	0.1413	0.1399	0.1405	0.1392
Gasch-1	0.0695	0.0730	0.0697	0.0728	0.0671	0.0695	0.0871	0.0580	0.0636
CLEF	0.5670	0.5664	0.5549	0.5571	0.5727	0.5546	0.5786	0.5791	0.5798
DMOZ-2010	0.2624	0.2623	0.2713	0.2689	0.2673	0.2525	0.2623	0.2673	0.2739
LSHTC-small	0.3082	0.2743	0.3249	0.2722	0.3047	0.3047	0.3128	0.3034	0.3182

Table 99 – F-Score results for the Clus-HMC approach with all datasets.

Dataset	F-Score
Cell-cycle	0.1208
Eisen	0.1271
Exp	0.1418
FMA MFCC	0.2196
Gasch-1	0.1254
CLEF	0.6496
DMOZ-2010	0.2300
LSHTC-small	0.2897

Table 100 – F-Score results for the HierCost approach with all datasets.

Dataset	F-Score
Cell-cycle	0.1676
Eisen	0.1973
Exp	0.1766
FMA MFCC	0.2236
Gasch-1	0.1587
CLEF	0.8569
DMOZ-2010	0.5824
LSHTC-small	0.6558

A.2 Statistical Tests

In this sections we present the results for the statistical tests applied into the classification results (presented before) in order to answer the questions raised in the discussion of Chapter 7.

Table 101 – Wilcoxon statistical tests for F-score results in the Flat Multi-Label scenarios.

	Z-score	p-value
LPROS	0.0530	0.4791
LPRUS	0.0000	0.5000
MLROS	0.0525	0.4791
MLRUS	0.2100	0.4168
MLeNN	0.2100	0.4168
REMEDIAL	0.2100	0.4168
MLSMOTE	-0.2100	0.4168
MLTL	0.1050	0.4582
MLSMOTE+MLTL	0.1050	0.4582

Table 102 – Wilcoxon statistical tests for F-score results in the resampling for the Local Classifiers per Node approach.

	Exclusive Siblings		Exclusive		Inclusive		Less Exclusive		Less Inclusive		Siblings	
	Z-score	p-value	Z-score	p-value	Z-score	p-value	Z-score	p-value	Z-score	p-value	Z-score	p-value
ROS	-0.210	0.417	-0.315	0.376	-0.210	0.417	-0.525	0.300	-0.420	0.337	-0.158	0.437
SMOTE	-0.840	0.200	-0.840	0.200	-0.525	0.300	-0.945	0.172	-1.050	0.147	-0.840	0.200
SMOTE-B1	-0.735	0.231	-0.735	0.231	-0.525	0.300	-1.050	0.147	-1.050	0.147	-0.840	0.200
SMOTE-B2	-0.578	0.282	-0.683	0.247	-0.630	0.264	-1.050	0.147	-0.735	0.231	-0.840	0.200
ADASYN	-0.420	0.337	-0.315	0.376	-0.420	0.337	-0.525	0.300	-0.735	0.231	-0.840	0.200
RUS	0.105	0.458	0.105	0.458	0.210	0.417	-0.630	0.264	-0.210	0.417	0.525	0.300
CC	-0.630	0.264	-0.105	0.458	-0.473	0.318	-0.945	0.172	-0.210	0.417	0.210	0.417
CNEN	-0.945	0.172	-0.525	0.300	-0.630	0.264	-1.050	0.147	-0.420	0.337	-0.525	0.300
ENN	-0.630	0.264	-0.735	0.231	-0.630	0.264	-1.260	0.104	-0.735	0.231	-0.630	0.264
RENN	-0.315	0.376	-0.630	0.264	-0.420	0.337	-1.050	0.147	-0.735	0.231	-0.315	0.376
AllKNN	-0.210	0.417	0.105	0.458	-0.105	0.458	-0.105	0.458	-0.735	0.231	-0.105	0.458
NM1	1.260	0.104	1.260	0.104	1.260	0.104	1.155	0.124	1.155	0.124	1.155	0.124
NM2	1.470	0.071	1.365	0.086	1.470	0.071	1.208	0.114	1.155	0.124	1.365	0.086
NM3	1.470	0.071	1.365	0.086	1.470	0.071	1.260	0.104	1.050	0.147	1.365	0.086
TL	-0.525	0.300	-0.368	0.357	0.000	0.500	-0.210	0.417	0.630	0.264	0.210	0.417
SMOTE+ENN	-1.260	0.104	-0.735	0.231	-1.260	0.104	-1.575	0.058	-0.945	0.172	-1.050	0.147
SMOTE+TL	-0.840	0.200	-0.525	0.300	-1.050	0.147	-1.575	0.058	-1.050	0.147	-0.945	0.172

Table 103 – Wilcoxon statistical tests for F-score results in the resampling for the Local Classifiers per Parent Node approach.

	Exclusive Siblings		Siblings	
	Z-score	p-value	Z-score	p-value
ROS	-0.210	0.417	0.000	0.500
SMOTE	-0.945	0.172	-0.735	0.231
SMOTE-B1	-0.945	0.172	-0.630	0.264
SMOTE-B2	-0.735	0.231	-0.210	0.417
ADASYN	-0.105	0.458	-0.315	0.376
RUS	-0.105	0.458	0.210	0.417
CC	-0.315	0.376	-0.420	0.337
CNEN	-0.525	0.300	-0.735	0.231
ENN	-0.840	0.200	-0.420	0.337
RENN	-0.630	0.264	-0.420	0.337
AllKNN	0.105	0.458	-0.105	0.458
NM1	1.365	0.086	0.840	0.200
NM2	1.050	0.147	1.470	0.071
NM3	1.050	0.147	1.365	0.086
TL	-0.315	0.376	-0.630	0.264
SMOTE+ENN	-0.735	0.231	-0.945	0.172
SMOTE+TL	-0.420	0.337	-0.735	0.231

Table 104 – Wilcoxon statistical tests for F-score results in the resampling for the Local Classifiers per Level approach.

	Z-score	p-value
ROS	-0.210	0.417
SMOTE	-1.155	0.124
SMOTE-B1	-0.945	0.172
SMOTE-B2	-0.945	0.172
ADASYN	-0.263	0.396
RUS	-0.210	0.417
CC	-0.315	0.376
CNEN	-0.840	0.200
ENN	-0.945	0.172
RENN	-0.735	0.231
AllKNN	0.105	0.458
NM1	1.365	0.086
NM2	1.418	0.078
NM3	1.260	0.104
TL	-0.630	0.264
SMOTE+ENN	-1.365	0.086
SMOTE+TL	-1.155	0.124

Table 105 – Average ranking of the classification results in the resampling for the Local Classifiers per Node approach.

	ROS	SMOTE	SMOTE B1	SMOTE B2	ADASYN	RUS	CC	CNEN	ENN	RENN	All KNN	NM1	NM2	NM3	TL	SMOTE + ENN	SMOTE + TL
ES	11.25	5.50	7.38	7.63	9.63	12.13	6.88	4.38	6.00	9.25	11.00	14.88	16.00	16.50	9.00	1.88	3.63
E	11.13	4.00	4.63	6.00	9.38	10.88	9.38	7.25	5.50	7.63	11.75	15.38	16.00	16.00	9.38	3.25	5.50
I	11.63	5.88	6.13	6.63	8.88	12.63	7.63	6.00	5.25	8.50	11.13	15.75	16.38	15.75	11.38	1.63	1.88
LE	10.25	5.25	6.25	5.13	9.25	10.38	8.25	7.00	6.50	8.75	12.63	15.38	15.25	16.25	9.88	2.25	4.38
LI	10.00	3.50	3.75	6.38	7.75	12.25	10.25	9.13	6.63	8.38	10.13	16.00	16.25	14.63	12.88	3.00	2.13
S	9.63	4.00	4.38	4.63	8.50	11.00	10.38	9.75	7.00	9.63	9.38	15.38	16.25	16.38	9.88	2.63	4.25
Avg. Rank	*10.65*	*4.69*	*5.42*	*6.06*	*8.90*	*11.54*	*8.79*	*7.25*	*6.15*	*8.69*	*11.00*	*15.46*	*16.02*	*15.92*	*10.40*	*2.44*	*3.63*

Table 106 – Average ranking of the classification results in the resampling for the Local Classifiers per Parent Node approach.

	ROS	SMOTE	SMOTE B1	SMOTE B2	ADASYN	RUS	CC	CNEN	ENN	RENN	All KNN	NM1	NM2	NM3	TL	SMOTE + ENN	SMOTE + TL
ES	10.13	3.75	4.25	7.00	10.88	10.88	10.00	6.38	4.75	7.63	11.50	15.75	15.50	15.50	8.50	4.13	6.50
S	11.50	5.13	7.88	9.75	9.13	12.75	7.25	3.25	6.63	8.75	11.63	13.75	16.00	16.00	7.63	1.38	4.63
Avg. Rank	*10.81*	*4.44*	*6.06*	*8.38*	*10.00*	*11.81*	*8.63*	*4.81*	*5.69*	*8.19*	*11.56*	*14.75*	*15.75*	*15.75*	*8.06*	*2.75*	*5.56*

Table 107 – Average ranking of the classification results in the resampling for the Local Classifiers per Level approach.

	ROS	SMOTE	SMOTE B1	SMOTE B2	ADASYN	RUS	CC	CNEN	ENN	RENN	All KNN	NM1	NM2	NM3	TL	SMOTE + ENN	SMOTE + TL
Avg. Rank	*11.13*	*4.88*	*6.25*	*7.38*	*10.88*	*11.63*	*9.25*	*6.25*	*5.63*	*5.63*	*11.75*	*15.25*	*15.75*	*14.63*	*6.75*	*4.00*	*6.00*

Table 108 – Average ranking of the classification results in the resampling for the all the Local Classifiers approaches.

	ROS	SMOTE	SMOTE B1	SMOTE B2	ADASYN	RUS	CC	CNEN	ENN	RENN	All KNN	NM1	NM2	NM3	TL	SMOTE + ENN	SMOTE + TL
LCN	10.65	4.69	5.42	6.06	8.90	11.54	8.79	7.25	6.15	8.69	11.00	15.46	16.02	15.92	10.40	2.44	3.63
LCPN	10.81	4.44	6.06	8.38	10.00	11.81	8.63	4.81	5.69	8.19	11.56	14.75	15.75	15.75	8.06	2.75	5.56
LCL	11.13	4.88	6.25	7.38	10.88	11.63	9.25	6.25	5.63	5.63	11.75	15.25	15.75	14.63	6.75	4.00	6.00
Avg. Rank	*10.86*	*4.67*	*5.91*	*7.27*	*9.92*	*11.66*	*8.89*	*6.10*	*5.82*	*7.50*	*11.44*	*15.15*	*15.84*	*15.43*	*8.40*	*3.06*	*5.06*

Table 109 – Post-Hoc Mannwhitney Test comparing the flat with the local classifier approaches.

LCN w/ ES	LCN w/ E	LCN w/ I	LCN w/ LE	LCN w/ LI	LCN w/ S	LCPN w/ ES	LCPN w/ S	LCL
3.12×10^{-22}	3.23×10^{-18}	2.34×10^{-23}	1.14×10^{-21}	4.21×10^{-25}	3.70×10^{-22}	1.18×10^{-03}	3.70×10^{-04}	1.00

Table 110 – Pearson Correlation Statistical Test for the $MeanIR_{LCN}$ measure and the classification results.

	Excl. Siblings		Exclusive		Inclusive		Less Exclusive		Less Inclusive		Siblings	
	ρ	*p-value*	ρ	*p-value*	ρ	*p-value*	ρ	*p-value*	ρ	*p-value*	ρ	*p-value*
CellCycle	-0.60	0.0086	-0.59	0.0106	-0.81	0.0000	-0.22	0.3850	-0.48	0.0457	-0.61	0.0070
Eisen	-0.61	0.0068	-0.77	0.0002	-0.80	0.0001	-0.82	0.0000	-0.60	0.0083	-0.69	0.0014
Exp	-0.74	0.0005	-0.68	0.0021	-0.74	0.0004	-0.69	0.0016	-0.65	0.0033	-0.66	0.0032
FMA MFCC	-0.56	0.0159	-0.73	0.0006	-0.85	0.0000	-0.84	0.0000	-0.69	0.0017	-0.76	0.0003
Gasch-1	-0.66	0.0027	-0.72	0.0008	-0.81	0.0000	-0.79	0.0001	-0.86	0.0000	-0.85	0.0000
CLEF	-0.87	0.0000	-0.32	0.1931	-0.34	0.1607	-0.68	0.0018	-0.55	0.0177	-0.65	0.0037
DMOZ-2010	-0.76	0.0001	-0.57	0.0143	-0.48	0.0423	-0.48	0.0397	-0.67	0.0023	-0.72	0.0006
LSHTC-small	-0.65	0.0035	-0.68	0.0020	-0.44	0.0646	-0.68	0.0016	-0.70	0.0011	-0.59	0.0094

Table 111 – Pearson Correlation Statistical Test for the $MeanIR_{LCPN}$ measure and the classification results.

	Exclusive Siblings		Siblings	
	ρ	*p-value*	ρ	*p-value*
Cell-cycle	-0.49	0.0373	-0.75	0.0003
Eisen	-0.41	0.0883	-0.70	0.0011
Exp	-0.58	0.0125	-0.68	0.0018
FMA MFCC	-0.56	0.0159	-0.57	0.0142
Gasch-1	-0.61	0.0077	-0.59	0.0106
CLEF	-0.68	0.0018	-0.72	0.0008
DMOZ-2010	-0.48	0.0461	-0.57	0.013
LSHTC-small	-0.73	0.0006	-0.63	0.005

Table 112 – Pearson Correlation Statistical Test for the $MeanIR_{LCL}$ measure and the classification results.

	ρ	*p-value*
Cell-cycle	-0.68	0.0020
Eisen	-0.65	0.0035
Exp	-0.74	0.0004
FMA MFCC	-0.67	.0021
Gasch-1	-0.73	.0005
CLEF	-0.81	0.0000
DMOZ-2010	-0.29	0.2418
LSHTC-small	-0.47	0.0504

Table 113 – Wilcoxon statistical tests comparing the best F-score results from the proposed approaches (LCN) *versus* the best results for each global classification approach considering all datasets.

	z-score	*p-value*
vs. HMC↔ML	-1.260	0.1038
vs. Clus-HMC	-1.365	0.0861
vs. HierCost	0.210	0.4168

APPENDIX

B Experimental Results of the COVID-19 Identification Study Case

The following tables presents a detailed version of the experimental results achieved in the COVID-19 Identification Study Case presented in Chapter 10 of this Thesis. This Appendix is subdivided into three Sections: (i) Baseline Results; (ii) Local Classifiers with Resampling Results; and (iii) Global Hierarchical Resampling Results.

B.1 Baseline Results

In this section we present the detailed results of the baseline experiments, i.e., the experiments with the Hierarchical Classification with Flat Resampling algorithms. The section is further subdivided into three subsections, which presents the results for each classification schema: (i) No Fusion; (ii) Early Fusion; (iii) Late Fusion.

B.1.1 No Fusion

Table 114 – Top-10 macro-avg f-score using the baseline approach.

Ord.	*Feature*	*Resampling*	*F-Score*
1	LPQ	SMOTE-B1	0.4428
2	LDN	SMOTE-B2	0.3817
3	LETRIST	SMOTE	0.3802
4	LETRIST	SMOTE+TL	0.3802
5	LPQ	SMOTE	0.3768
6	LPQ	ROS	0.3768
7	LPQ	SMOTE+TL	0.3768
8	LETRIST	ROS	0.3749
9	LETRIST	SMOTE-B2	0.3735
10	LETRIST	SMOTE-B2	0.3595

Table 115 – Top-10 COVID-19 f-score using the baseline approach.

Ord.	*Feature*	*Resampling*	*F-Score*
1	OBIF	ENN	0.7463
2	OBIF	-	0.7241
3	LETRIST	ENN	0.7213
4	LDN	SMOTE-B1	0.7200
5	OBIF	ADASYN	0.7143
6	OBIF	SMOTE-B2	0.6984
7	OBIF	SMOTE-B1	0.6800
8	LDN	SMOTE	0.6792
9	LDN	SMOTE+TL	0.6792
10	LDN	ENN	0.6780

Table 116 – Best macro-avg f-score per feature set using the baseline approach.

Feature	*Resampling*	*F-Score*
BSIF	ADASYN	0.2921
EQP	ROS	0.3341
INCEPTION	SMOTE	0.2488
LBP	CNEN	0.2937
LDN	SMOTE-B2	0.3817
LETIRST	SMOTE	0.3802
LPQ	SMOTE-B1	0.4428
OBIF	SMOTE-B1	0.2898

Table 117 – Best COVID-19 f-score per feature set using the baseline approach.

Feature	*Resampling*	*F-Score*
BSIF	ROS	0.5862
EQP	-	0.6129
INCEPTION	ENN	0.3214
LBP	ENN	0.5574
LDN	SMOTE-B1	0.7200
LETIRST	ENN	0.7213
LPQ	SMOTE-B1	0.6545
OBIF	ENN	0.7463

B.1.2 Early Fusion

Table 118 – Top-10 macro-avg f-score with the early fusion technique using the baseline approach.

Ord.	*Combined Features*	*Resampling*	*F-Score*
1	BSIF & EQP & OBIF	-	0.4996
2	LETRIST & EQP & OBIF	-	0.4733
3	BSIF & OBIF	-	0.4484
4	BSIF & LDN	-	0.4450
5	INCEPTION & LETRIST & EQP	-	0.4150
6	EQP & LPQ	SMOTE-B1	0.4150
7	LBP & BSIF	SMOTE	0.4143
8	LBP & BSIF	SMOTE-B1	0.4143
9	LBP & BSIF	SMOTE+TL	0.4143
10	LETRIST & OBIF	-	0.4136

Table 119 – Top-10 COVID-19 f-score with the early fusion technique using the baseline approach.

Ord.	*Combined Features*	*Resampling*	*F-Score*
1	LETRIST & OBIF	SMOTE-B1	0.7931
2	BSIF & LETRIST & LPQ	-	0.7407
3	EQP & LPQ	TL	0.7333
4	EQP & LPQ	-	0.7333
5	INCEPTION & LETRIST & EQP	-	0.7308
6	BSIF & EQP & LPQ	-	0.7302
7	INCEPTION & EQP & OBIF	-	0.7273
8	LDN & OBIF	-	0.7241
9	LETRIST & LPQ & OBIF	-	0.7170
10	LBP & EQP & LPQ	-	0.7143

B.1.3 Late Fusion

Table 120 – Top-3 macro-avg f-score with the late fusion technique using the baseline approach.

Choice Strategy	*Combination*	*Fusion Strategy*	*F-Score*
Top-5 best of all	(LETRIST, SMOTE-B2), (LPQ, SMOTE-B1)	PROD	0.4428
	(LETRIST, SMOTE-B2), (LPQ, SMOTE-B1)	SUM	0.4428
	(LETRIST, SMOTE-B2), (LPQ, SMOTE-B1)	VOTE	0.3950
Top per feature	(BSIF, SMOTE-B1), (EQP, SMOTE-B2), (LPQ, SMOTE-B1)	PROD	0.4428
	(BSIF, SMOTE-B1), (EQP, SMOTE-B2), (LPQ, SMOTE-B1)	SUM	0.4428
	(BSIF, SMOTE-B1), (EQP, SMOTE-B2), (LPQ, SMOTE-B1)	PROD	0.4428

Table 121 – Top-3 COVID-19 f-score with the late fusion technique using the baseline approach.

Choice Strategy	*Combination*	*Fusion Strategy*	*F-Score*
Top-5 best of all	(BSIF, ADASYN), (EQP, SMOTE-B2), (LPQ, SMOTE-B1)	VOTE	0.7547
	(BSIF, ADASYN), (EQP, SMOTE-B2), (OBIF, SMOTE-B1)	VOTE	0.7547
	(BSIF, ADASYN), (INCEPTION, SMOTE), (OBIF, SMOTE-B1)	VOTE	0.7547
Top per feature	(BSIF, SMOTE-B1), (EQP, SMOTE-B2), (LPQ, SMOTE-B1)	VOTE	0.7869
	(BSIF, SMOTE-B1), (EQP, SMOTE-B2)	VOTE	0.7869
	(BSIF, SMOTE-B1), (INCEPTION, ENN), (EQP, SMOTE-B2)	VOTE	0.7869

B.2 Local Classifiers with Resampling

In this section we present the detailed results of the Local Classifiers with Resampling experiments, i.e., the experiments with the Hierarchical Classification using Local Classifiers (LCN, LCPN and LCL) with Resampling. The section is further subdivided into two subsections, which shows the results for each classification schema tested in this context: (i) No Fusion; (ii) Early Fusion.

B.2.1 No Fusion

Table 122 – Top-10 macro-avg f-score using the local classifiers approaches.

Ord.	*Feature*	*Resampling*	*Approach*	*F-Score*
1	LETRIST	SMOTE-B2	LCPN	0.5332
2	OBIF	SMOTE	LCPN	0.5065
3	LDN	SMOTE-B2	LCPN	0.5027
4	LPQ	SMOTE-B2	LCPN	0.4974
5	LPQ	ADASYN	LCN	0.4748
6	LBP	AllKNN	LCPN	0.4712
7	LDN	SMOTE-B2	LCN	0.4696
8	LBP	RENN	LCPN	0.4674
9	LETRIST	SMOTE	LCPN	0.4650
10	LPQ	AllKNN	LCPN	0.4515

Table 123 – Top-10 COVID-19 f-score using the local classifiers approaches.

Ord.	*Feature*	*Resampling*	*Approach*	*F-Score*
1	LETRIST	SMOTE	LCPN	0.8936
2	LDN	SMOTE-B2	LCPN	0.8889
3	OBIF	-	LCN	0.8846
4	OBIF	TL	LCN	0.8846
5	OBIF	ENN	LCPN	0.8837
6	LETRIST	SMOTE+TL	LCPN	0.8750
7	OBIF	ENN	LCN	0.8621
8	OBIF	RENN	LCN	0.8475
9	OBIF	RENN	LCPN	0.8462
10	OBIF	TL	LCPN	0.8462

Table 124 – Best macro f-score per feature set using the local classifiers approaches.

Feature	*Resampling*	*Approach*	*F-Score*
BSIF	ADASYN	LCPN	0.4071
EQP	ENN	LCPN	0.4468
INCEPTION	original	LCPN	0.2438
LBP	AllKNN	LCPN	0.4712
LDN	SMOTE-B2	LCPN	0.5027
LETRIST	SMOTE-B2	LCPN	0.5332
LPQ	SMOTE-B2	LCPN	0.4974
OBIF	SMOTE	LCPN	0.5065

Table 125 – Best COVID-19 f-score per feature set using the local classifiers approaches.

Feature	*Resampling*	*Approach*	*F-Score*
BSIF	TL	LCPN	0.8077
EQP	ENN	LCPN	0.8077
INCEPTION	original	LCPN	0.7500
LBP	RENN	LCPN	0.8372
LDN	SMOTE-B2	LCPN	0.8889
LETRIST	SMOTE	LCPN	0.8936
LPQ	AllKNN	LCPN	0.8400
OBIF	original	LCN	0.8846

B.2.2 Early Fusion

Table 126 – Top-10 macro-avg f-score with the early fusion technique using the local classifiers approaches.

Ord.	*Approach*	*Combined Features*	*Resampling*	*F-Score*
1	LCPN	LDN & LETRIST & LPQ	SMOTE-B2	0.6643
2	LCPN	LBP & LETRIST	ADASYN	0.6423
3	LCPN	LETRIST & OBIF	SMOTE-B2	0.6277
4	LCPN	LDN & LPQ & OBIF	SMOTE-B2	0.6258
5	LCPN	INCEPTION & LPQ & OBIF	SMOTE-B2	0.6246
6	LCPN	LBP & LETRIST & EQP	SMOTE-B2	0.6217
7	LCPN	BSIF & LDN & LETRIST	SMOTE-B2	0.6190
8	LCPN	LETRIST & OBIF	SMOTE-B1	0.6188
9	LCPN	LDN & LETRIST & EQP	SMOTE-B2	0.6111
10	LCPN	LETRIST & EQP & OBIF	SMOTE-B2	0.6067

Table 127 – Top-10 COVID-19 f-score with the early fusion technique using the local classifiers approaches.

Ord.	*Approach*	*Combined Features*	*Resampling*	*F-Score*
1	LCPN	LETRIST & EQP & OBIF	SMOTE-B2	0.9333
2	LCPN	LBP & OBIF	RENN	0.9231
3	LCPN	LBP & LDN & OBIF	SMOTE-B2	0.9189
4	LCPN	LBP & LETRIST	ADASYN	0.9167
5	LCPN	LDN & LETRIST& OBIF	SMOTE+TL	0.9091
6	LCPN	INCEPTION & OBIF	SMOTE	0.9057
7	LCPN	BSIF & LDN & OBIF	SMOTE-B2	0.9048
8	LCPN	LETRIST & EQP	RENN	0.9048
9	LCPN	LETRIST & OBIF	SMOTE-B1	0.9048
10	LCPN	BSIF & LDN & LETRIST	SMOTE-B2	0.9020

B.3 Global Hierarchical Resampling

In this section we shows the detailed results of the experiments using the Global Hierarchical Classification with Global Resampling Algorithms. The section is also further subdivided into three subsections, which presents the results for each classification schema: (i) No Fusion; (ii) Early Fusion; (iii) Late Fusion.

B.3.1 No Fusion

Table 128 – Top-10 macro-avg f-score the global resampling approach.

Ord.	*Feature*	*Resampling*	*F-Score*
1	LDN	HROS-15	0.4662
2	LDN	HROS-10	0.4548
3	LPQ	HROS-5	0.4419
4	LPQ	HROS-10	0.4326
5	LETRIST	HSMOTE-3	0.4272
6	LETRIST	HROS-10	0.4225
7	LDN	HSMOTE-3	0.4212
8	LDN	HROS-20	0.4179
9	LPQ	HSMOTE-5	0.4139
10	BSIF	HROS-10	0.4132

Table 129 – Top-10 Covid-19 f-score using the global resampling approach.

Ord.	*Feature*	*Resampling*	*F-Score*
1	OBIF	HROS-5	0.7692
2	LETRIST	HROS-10	0.7458
3	OBIF	HROS-15	0.7308
4	LPQ	HRUS-5	0.7273
5	OBIF	HRUS-5	0.7241
6	LDN	HRUS-5	0.7213
7	LDN	HSMOTE-3	0.7059
8	LDN	HROS-15	0.6977
9	LETRIST	HROS-20	0.6786
10	LETRIST	HROS-5	0.6765

Table 130 – Best macro-avg f-score per feature set using the global resampling approach.

Feature	*Resampling*	*F-Score*
BSIF	HROS-10	0.4132
EQP	HROS-20	0.4069
INCEPTION	HRUS-10	0.3260
LBP	HROS-20	0.3997
LDN	HROS-15	0.4662
LETRIST	HSMOTE-3	0.4272
LPQ	HROS-5	0.4419
OBIF	HROS-5	0.3626

Table 131 – Best COVID-19 f-score per feature set using the global resampling approach.

Feature	*Resampling*	*F-Score*
BSIF	HRUS-5	0.5926
EQP	HRUS-10	0.6383
INCEPTION	HSMOTE-5	0.3509
LBP	HRUS-10	0.5778
LDN	HRUS-5	0.7213
LETRIST	HR0S-10	0.7458
LPQ	HRUS-5	0.7273
OBIF	HROS-5	0.7692

B.3.2 Early Fusion

Table 132 – Top-10 macro-avg f-score with the early fusion technique using the global resampling approach.

Ord.	*Combined Features*	*Resampling*	*F-Score*
1	EQP & LPQ	HROS-5	0.5524
2	INCEPTION & LETRIST & EQP	HSMOTE-5	0.5110
3	INCEPTION & LPQ	HROS-15	0.5017
4	BSIF & EQP & OBIF	-	0.4996
5	LBP & LPQ & OBIF	HRUS-5	0.4827
6	LETRIST & EQP & OBIF	-	0.4733
7	BSIF & LDN & OBIF	HROS-10	0.4728
8	LBP & LPQ & OBIF	HROS-15	0.4712
9	LDN & LETRIST & LPQ	HROS-15	0.4705
10	BSIF & LETRIST & LPQ	HRUS-5	0.4597

Table 133 – Top-10 COVID-19 f-score with the early fusion technique using the global resampling approach.

Ord.	*Combined Features*	*Resampling*	*F-Score*
1	EQP & LPQ	HROS-5	0.8235
2	LDN & OBIF	HRUS-10	0.8136
3	INCEPTION & EQP & LPQ	HRUS-5	0.8070
4	LETRIST & LPQ	HSMOTE-5	0.8000
5	LETRIST & OBIF	-	0.7931
6	LETRIST & LPQ	HRUS-5	0.7931
7	BSIF & LETRIST & LPQ	HRUS-5	0.7925
8	LDN & EQP & OBIF	HROS-15	0.7755
9	BSIF & EQP & LPQ	HRUS-5	0.7719
10	BSIF & EQP & OBIF	HROS-10	0.7692

B.3.3 Late Fusion

Table 134 – Top-3 macro-avg f-score with the late fusion technique using the global resampling approach.

Choice Strategy	*Combination*	*Fusion Strategy*	*F-Score*
Top-5 best of all	(BSIF, HRUS-5), (LETRIST, HSMOTE-3)	PROD	0.4272
	(BSIF, HRUS-5), (LETRIST, HSMOTE-3)	SUM	0.4272
	(LBP, HROS-20), (LETRIST, HSMOTE-3)	PROD	0.4272
Top-1 per feature	(BSIF, HRUS-5), (EQP, HROS-20), (LBP, HROS-20), (LDN, HROS-10)	SUM	0.4548
	(BSIF, HRUS-5), (EQP, HROS-20), (LDN, HROS-10)	PROD	0.4548
	(BSIF, HRUS-5), (EQP, HROS-20), (LDN, HROS-10)	SUM	0.4548

Table 135 – Top-3 Covid-19 f-score with the late fusion technique using the global resampling approach.

Choice Strategy	*Combination*	*Fusion Strategy*	*F-Score*
Top-5 best of all	(EQP, HRUS-10), (LBP, HRUS-10), (OBIF, HROS-5)	PROD	0.7692
	(EQP, HRUS-10), (LBP, HRUS-10), (OBIF, HROS-5)	SUM	0.7692
	(EQP, HRUS-10), (LBP, HRUS-10), (OBIF, HROS-5)	VOTE	0.7692
Top-1 per feature	(BSIF, HRUS-5), (EQP, HROS-20), (LBP, HROS-20), (OBIF, HROS-5)	PROD	0.7692
	(BSIF, HRUS-5), (EQP, HROS-20), (LBP, HROS-20), (OBIF, HROS-5)	SUM	0.7692
	(BSIF, HRUS-5), (EQP, HROS-20), (LBP, HROS-20), (OBIF, HROS-5)	VOTE	0.7692