Marcos Monteiro Junior

Geração de *pool* de classificadores buscando diversidade nos espaços de decisão e complexidade do problema

Curitiba - PR, Brasil
2021

Marcos Monteiro Junior

Geração de *pool* de classificadores buscando diversidade nos espaços de decisão e complexidade do problema

Tese apresentada ao Programa de Pós Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Doutor em Informática

Pontifícia Universidade Católica do Paraná - PUCPR
Programa de Pós-Graduação em Informática - PPGIa

Orientador: Alceu de Souza Britto Junior
Coorientador: Robert Sabourin

Curitiba - PR, Brasil
2021

Agradecimentos

Agradecer é um dos atos mais nobres, antes de ser uma atitude elogiosa, é um ato de reconhecimento. A Gratidão nos torna, na aprendizagem dos dias, cada vez mais existentes em relação aos outros. Agradeço primeiramente a Deus de maneira indizível pela vida e por ter me concedido a permissão para realizar está pesquisa.

A minha família, "o obrigado" será sempre uma palavra perene principalmente aos meus pais, por todo apoio e incentivo para o meu crescimento pessoal e profissional. Ao meu irmão Lucas e sua esposa Marina que a longa data presenciam minha trajetória acadêmica e muitas vezes souberam compreender a minha ausência, o meu isolamento e os momentos de reclusão.

Ao meu filho Miguel por ser meu companheiro e minha motivação para minhas realizações.

À minha amiga Yara que participou das últimas etapas desse trabalho, dando apoio e motivação.

Ao meu orientador Professor Dr. Alceu de Souza Britto Jr. pela orientação e incentivo na realização desse trabalho.

Ao Professor Dr. Robert Sabourin pela co-orientação, ensinamentos e sugestões.

Ao Professor Dr. Jean Paul Barddal pelas sugestões no desenvolvimento desse trabalho.

Aos Professores e aos colegas do PPGIa.

À CAPES pela concessão da Bolsa de Doutorado.

À todos que acompanharam o percurso e contribuíram para a realização desse trabalho.

Por fim, aos não mencionados, mas que guardo na memória, Muito Obrigado !!!

Resumo

A classificação é uma tarefa do aprendizado de máquina utilizada para classificar objetos, imagens, músicas, entre outras aplicações. A tarefa de classificação iniciou com algoritmos que geram modelos preditivos por meio da observação de um conjunto de dados rotulados. Um conjunto de modelos criados por estes algoritmos forma o *pool* de classificadores. O uso do *pool* de classificadores é uma estratégia bastante utilizada, pois pode aumentar o desempenho da classificação. A ideia desta estratégia é que os erros de classificação sejam minimizados através do uso de múltiplos classificadores ao invés de um único classificador. Existem diversas formas de gerar o conjunto de classificadores, uma delas é utilizar as meta-características do problema. A dificuldade de um problema de classificação é descrita como meta-características extraídas de um conjunto de dados que retratam a dificuldade de classificação. Outras meta-características são as medidas de diversidade que avaliam o quão diferentes são os modelos de aprendizado. Neste contexto, este trabalho propõe um método para gerar um conjunto de classificadores, cujo diferencial está no uso de medidas que avaliam a complexidade do problema e diversidade quanto à decisão dos classificadores. O método proposto está dividido em duas etapas. Na primeira foram utilizadas as medidas de complexidade para avaliar o comportamento de cada métrica nos problemas de classificação. As métricas que apresentam maior dispersão em diferentes amostragens dos dados, participam do segundo estágio do método. Já no segundo estágio do método, é utilizado um algoritmo evolutivo modificado que explora a dispersão dos subproblemas no espaço de complexidade e de decisão, para então gerar o *pool* de classificadores. Por fim, os modelos gerados foram utilizados em métodos de seleção dinâmica e combinação de classificadores. Foi aplicado um protocolo experimental envolvendo 28 bases de dados provindos de diferentes repositórios. Foram realizados ao todo 2240 experimentos somente para o método proposto. Em alguns deles, os resultados foram superiores em até 20 bases de dados das 28 testadas, isto é, 71,4%, quando comparado a outros métodos com a mesma proposta disponíveis na literatura.

Palavras-chaves: Geração de *pool* de classificadores; complexidade do problema; sistemas de múltiplos classificadores.

Abstract

Classification is a machine learning task used to classify objects, images, music, among other applications. The classification task started with algorithms that generate predictive models by observing a set of labeled data. A set of models created by these algorithms form the *pool* of classifiers. A pool of classifiers is a widely used strategy, as it can increase classification performance. The idea of this strategy is that classification errors are minimized through the use of multiple classifiers instead of a single classifier. There are several ways to generate classifiers, one of which is to use the problem's meta-characteristics. The difficulty of a classification problem is described as meta-characteristics drawn from a dataset that portrays the difficulty of classification. Other meta-features are diversity measures that assess how different learning models are. In this context, this work proposes a method to generate a set of classifiers, whose differential lies in using measures that assess the complexity of the problem and diversity regarding classifiers' decisions. The method of this thesis was divided into two steps. In the first, complexity measures were used to assess the behavior of each metric in classification problems. Metrics with more dispersion participate in the second stage of the method. In the second stage of the method, a modified evolutionary algorithm explores the dispersion of sub-problems in the space of complexity and decision to generate the pool of classifiers. Finally, the generated models were applied in dynamic selection and classifier combination methods. An experimental protocol was applied involving 28 databases from different repositories. A total of 2240 experiments were carried out only for the proposed method. In some of them, the results were superior in up to 20 databases out of the 28 tested, that is, 71.4% when compared to other methods.

Keywords: Generation of pool of classifiers; problem complexity; multiple classifier systems.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ACC	Acurácia.
AGM	Algoritmos genéticos multiobjetivo.
AGs	Algoritmos genéticos.
AM	Aprendizado de máquina.
CL	Conjunto local.
CP	Curva de Pareto.
DD	Diversidade no espaço de decisão.
F4	Eficiência coletiva dos atributos.
F3	Eficiência máxima de um atributo.
KEEL	Extração de conhecimento com base no repositório de aprendizagem evolutiva, do inglês *Knowledge extraction based on evolutionary learning repository*.
FR	Floresta rotativa.
T1	Fração de esferas de cobertura máxima.
N1	Fração de pontos na região de fronteira.
GP	Geração de *pool*.
MC	Medidas de complexidade.
META-DES	Meta-aprendizado para seleção dinâmica de conjuntos de classificadores, do inglês *Meta learning for dynamic ensemble selection*.
L3	Não-linearidade de um classificador linear.
N4	Não-linearidade do classificador KNN em dados sintéticos.
NSGA, NSGAII	Ordenação não dominada em algoritmos genéticos, do inglês *Non dominated Sorting in Genetic Algorithms*.
F1	Razão máxima do discriminante de Fisher.
N2	Relação dos vizinhos mais próximos intra/extra classes.

SC	Seleção de classificadores.
SDC	Seleção dinâmica de classificadores.
SDE	Seleção dinâmica de ensembles de classificadores.
SD	Seleção dinâmica.
SMC	Sistemas de múltiplos classificadores.
L1	Soma das distâncias do erro do classificador linear.
N3	Taxa de erro do classificador dos vizinhos mais próximos.
L2	Taxa de erro do classificador linear.
F1v	Vetor direcional da razão máxima do discriminante de Fisher.
F2	Volume da região de sobreposição.
VM	Voto majoritário.
RP	*Ranking* de Pareto.
LCA	Acurácia local da classe, do inglês *Local class accuracy.*
OLA	Acurácia local geral, do inglês *Overall local accuracy.*
SPEA, SPEAII	Algoritmo evolutivo de força de Pareto, do inglês *Strength Pareto Evolutionary Algorithm.*
PCA	Análises dos compontes principais, do inglês *Principal component analysis.*
MST	Árvore de abrangência mínima do inglês *Minimum spanning tree*
LSCAvg	Cardinalidade média do conjunto local, do inglês *local set average cardinality measure LSCAvg.*
RRC	Classificador de referência aleatório, do inglês *Randomized reference classifier.*
LKC	Coleção de dados médicos reais, do inglês *Ludmila Kuncheva Collection of real medical data.*
MCB	Comportamento de multiplos classificadores, do inglês *Multiple classifier behaviour.*
kDN	Desacordo dos vizinhos de K, do inglês *K-Disagreeing Neighbors.*

GPCDa	Geração de *Pool* de Classificadores fundamentado na Diversidade de dois níveis - acurácia.
GPCDd	Geração de *Pool* de Classificadores fundamentado na Diversidade de dois níveis - dispersão.
GPCD	Geração de *Pool* de Classificadores fundamentado na Diversidade de dois níveis.
KNORA-E	K oráculos mais próximos - Eliminação, do inglês *KNORA Eliminate.*
KNORA-U	K oráculos mais próximos - União, do inglês *KNORA Union.*
KNORA	K oráculos mais próximos, do inglês *k-nearest-oracle.*
SVM	Máquina de vetores de suporte, do inglês *Suport vector machine.*
ELENA	O aprendizado aprimorado para o projeto de arquiteturas neurais evolutivas, do inglês *The enhanced learning for evolutive neural architectures project.*
UCI	Repositório de aprendizado de máquina Uc irvine, do inglês *Uc irvine machine learning repository.*
RS	Subespaço aleatório, do inglês *Random subspace.*
VEGA	Vetor de validação de algoritmo genético, do inglês *Vector evaluated genect algorithm.*

Lista de símbolos

ACC_c	Acurácia do classificador C.
τ	Amostra de treino.
f	Atributos.
ce	Centroide médio ou ponto central.
C	Classificador.
CoC	Coeficiente de correlação.
ω	Combinação dos classificadores.
C_{ov}	Complexidade da amostra de validação.
C_{os}	Complexidade dop subproblema S.
EoC^*	Conjunto de classificadores reduzidos.
EoC^*	Conjunto de classificadores retornados dos métodos SDE.
C_c	Conjunto de classificadores.
Cc	Conjunto de classificadores.
L	Contador de repetições.
P_{tcop}	Cópia de uma população (P_t).
$\mathcal{D}$	Dispersão global.
$Dist$	Distância.
$\mathcal{B}$	Distribuição uniforme.
ϵ	Erro de predição do classificador.
A	Estrutura para armazenar os indivíduos do AGM.
v	Famílias de medidas de complexidade.
$\mathcal{F}$	Fitness.
$\pounds$	Frente de Pareto.
G	Gerações.

η	Importância do classificador.
b	Indicador de função.
N_d	Indivíduos não dominados.
I	Indivíduos.
x'	Instância sintética.
x	Instância.
x^*	Instância de teste.
M_{ce}	Matriz de centroides.
θ	Matriz de complexidade e diversidade para cada subproblema.
$Vdist$	Matriz de distância.
Δ	Matriz do atributos para uma classe específica.
B	Matriz dos atributos entre as classes.
W	Matriz dos atributos no próprio rótulo.
o	Medida de complexidade específica.
MC	Medida de complexidade.
MD	Medida de diversidade.
λ	Meta-característica.
Vf	Meta-espaço de características.
c	Níveis de confiança.
Z	Normalização.
m	Número de atributos.
r	Número de classificadores e subamostras.
n_o	Número de exemplos do problema que estão na região de sobreposição para um atributo.
g	Número de frentes de Pareto.
$g*$	Número de gerações.

u	Número de instâncias de um subproblema.
T^*	Número de iterações.
n	Número de par de exemplos (x_i, y_i).
s	Número de subpopulações.
β	Número não negativo.
Re	Número real.
O	Objetivos do AGM.
w	Peso das instâncias.
h_w	Peso para o classificador de base.
Q	População de pais e filhos.
P'	População final.
P^*	População formada pelos filhos.
P_t	População.
Φ	Predição do classificador.
D	Problema de classificação.
xp	Proporção de exemplos.
Qs	Q-estatística (do inglês q-statístic).
R'	Ranking da população de filhos.
R	Ranking da população.
r_f	Razão dos atributos discriminantes.
δ	Região de competência do classificador.
ϑ	Região difícil para classificação de acordo com uma métrica.
Φ	Resultado da predição do classificador/predição do modelo.
h	Resultado do classificador linear.
y^*	Rótulo da instância de teste.
y	Rótulo ou classe.

Os	Soluções dos objetivos.
P_s	Subpopulações de um AGM.
S	Subproblema.
v'	Total de famílias de complexidade.
or	Total de medidas de acordo com o total de subproblemas.
o'	Total de medidas dentro de um família.
z	Total de objetivos.
y^*	Total de rótulos.
vl	Valor aleatório.
Vf	Vetor de atributos.
Vy	Vetor de rótulos.
d	Vetor direcional.
ne	Vizinho mais próximo da instância de classe diferente dessa instância.
k	Vizinhos da instância de teste.

Sumário

1 Introdução

O aprendizado de máquina (AM) tem auxiliado na tomada de decisão em diferentes áreas do conhecimento. Uma tarefa importante de AM é a classificação, a qual está presente no cotidiano das pessoas. Tarefas que parecem ser simples para uma pessoa podem ser complexas para um computador como, por exemplo, classificar em diferentes categorias ou classes. A tarefa de classificação também está presente em outras aplicações, tais como: o auxílio de diagnóstico médico, biometria, sistemas de segurança, leitura automática de documentos, entre outras.

A classificação teve início por meio de modelos monolíticos, isto é, sistemas que consistem em apenas um classificador. Com o avanço das técnicas de classificação, notou-se que Sistemas de Múltiplos Classificadores (SMC) trazem benefícios em relação aos sistemas monolíticos, por permitirem a combinação de modelos com competências em diferentes regiões do problema (WOŹNIAK; GRAÑA; CORCHADO, 2014; ROKACH, 2010; REN; ZHANG; SUGANTHAN, 2016).

Os SMCs podem ter três estágios. O primeiro deles é a geração de classificadores, seguido pela seleção (opcional) e, por último, a integração. Na geração cria-se diferentes classificadores com o objetivo de formar um conjunto de classificadores (*pool* de classificadores) complementares, diversos e com diferentes competências. O segundo estágio seleciona os classificadores do conjunto. Essa seleção pode ser estática ou dinâmica. Na seleção estática, os classificadores são selecionados no decorrer da fase de treinamento. No processo dinâmico, a seleção de classificadores ocorre na fase de teste. Na estratégia de seleção dinâmica, um classificador ou um conjunto de classificadores é escolhido no *pool* para rotular cada instância de teste. A fase da integração representa o terceiro estágio, e é responsável pela fusão da decisão de cada classificador selecionado. É importante salientar que a fase de seleção é facultativa, podendo todos os classificadores gerados serem combinados. Outrossim, a fase de integração somente faz-se necessária quando mais de um classificador é considerado para gerar a decisão final do sistema sobre um determinado padrão de teste.

A geração de conjunto de classificadores desempenha um papel importante para o sucesso de um SMC. De acordo com Dietterich (2000), um *pool* de classificadores é formado por diferentes modelos, cujas decisões individuais são combinadas de alguma forma para classificar novos exemplos de um conjunto de teste. O *pool* de classificadores, ou *pool* de especialistas pode ser categorizado de duas maneiras: heterogêneo ou homogêneo. Na primeira estratégia, o conjunto de treinamento é utilizado para treinar modelos utilizando classificadores de base diferentes. Na segunda abordagem, os especialistas são formados com o mesmo classificador de base. Contudo, os exemplos do problema que formam o conjunto de treinamento são amostrados de diferentes maneiras de modo a originar modelos

diversificados. Outra maneira de se obter um *pool* homogêneo é variando os parâmetros do classificador de base.

Na abordagem homogênea encontram-se métodos onde as instâncias do problema de classificação são agrupadas de alguma forma para originar novas subamostras. Entre esses métodos, destacam-se o *Bagging* (BREIMAN, 1996) e o *AdaBoost* (FREUND; SCHAPIRE, 1997). Outros métodos de GP criam subamostras a partir da variação do número de atributos da base de treinamento como, por exemplo, o *Random Subspace* (HO, 1998). Outras técnicas que podem utilizar os atributos ou as instâncias para a geração das subamostras são *Random Forest* (BREIMAN, 2001) e *Rotation Forest* (RODRIGUEZ; KUNCHEVA; ALONSO, 2006).

Todos os métodos de geração homogênea citados acima têm ao menos um objetivo em comum: a criação de subamostras com diversidade. Claramente não há ganho de acurácia em um conjunto de classificadores que é composto por elementos idênticos. Assim, se houver diversidade entre os classificadores para serem combinados ou escolhidos por algum método de seleção dinâmica, pode-se esperar um aumento de precisão (SANTANA et al., 2006). Logo, um ponto de partida para a criação de uma nova técnica de GP é encontrar uma maneira de se obter subconjuntos de amostras representando subproblemas diferentes e consequentemente classificadores diversos.

Uma maneira de se obter diversidade em um *pool* de especialistas é utilizar meta-características do problema de classificação. Metas características são um conjunto de medidas usadas para extrair informações relevantes de um dado problema. Estudos indicam que a dificuldade do problema de classificação, ou complexidade do problema, é uma informação importante que pode ser aplicada em diferentes áreas do aprendizado de máquina. As medidas de complexidade (MC) extraem as metas características que mensuram a dificuldade de classificação. As pesquisas dos autores Macià, Orriols-Puig e Bernadó-Mansilla (2010), Cano (2013), Roy et al. (2016), Brun et al. (2018), Garcia, Carvalho e Lorena (2015), Ho e Basu (2002), Cruz, Sabourin e Cavalcanti (2017), Luengo e Herrera (2015), Lorena et al. (2012), Mollineda, Sánchez e Sotoca (2005) demonstram a influência da complexidade no desempenho de classificadores.

Primeiramente, as MCs foram divididas em três grupos pelos autores Orriols-Puig, Macì e Ho (2010). São eles: sobreposição, separabilidade, e geometria espacial de um problema de classificação. Uma segunda caracterização proposta pelos autores Lorena et al. (2019) organiza as medidas de complexidade em seis grupos conforme o cálculo de cada medida. Os grupos são definidos como medidas de sobreposição, linearidade, vizinhança, redes, dimensionalidade e balanceamento de classes.

Cada grupo de MC apresenta formas diferentes de calcular e extrair as propriedades de complexidade de um problema. Medidas de sobreposição mensuram quão próximos e sobrepostos estão os atributos de uma base de dados. As medidas de vizinhança, de modo geral, avaliam as fronteiras e as bordas entre classes diferentes. As medidas de linearidade

mensuram a dificuldade de gerar um modelo linear. Medidas de redes aplicam conceitos de redes e grafos para representar o problema de classificação. Em outra categoria, as medidas de dimensionamento avaliam a densidade dos atributos de um problema de classificação, isto é, regiões com muita concentração de atributos (características com valores muito próximos). Por fim, as medidas de balanceamento de classe representam a diferença entre o número de exemplos de uma classe em relação à outra.

Em Brun et al. (2018), durante a construção de um novo método de seleção dinâmica de classificadores, os autores relataram a importância de se treinar os classificadores do *pool* com base em subconjuntos de dados que representam diferentes níveis de dificuldade em termos de classificação. Inspirado nesta observação, o presente trabalho explora o uso de MC como mecanismo de geração de diversidade. Neste contexto, o desafio está em identificar as MCs mais adequadas para compor um novo método de geração de *pool* de classificadores (GP), bem como investigar a possibilidade de combinação entre a diversidade gerada no espaço de complexidade do problema e a diversidade referente ao espaço de decisão dos classificadores.

O desenvolvimento desta pesquisa consiste em criar um conjunto de classificadores que cubra adequadamente o espaço de complexidade do problema e a diversidade no espaço de decisão. Cada problema de classificação é representado por um ou mais grupos de medidas de complexidade. O método é composto por duas etapas: a primeira com a análise da dispersão das medidas de complexidade; a segunda etapa representa a geração do *pool*. Esta etapa subdivide-se em duas, responsáveis por analisar cada estágio na fase de geração.

Na análise da dispersão, o comportamento das medidas de complexidade é avaliado considerando diferentes subconjuntos de dados de treinamento do problema em questão. As medidas que apresentarem maior dispersão participam da segunda etapa do método. Na geração do *pool*, o conjunto de classificadores é construído com uso das medidas que foram obtidas como resultado na fase de análise da dispersão. Ainda na segunda etapa, utiliza-se uma medida de diversidade no espaço de decisão dos classificadores. Um algoritmo evolutivo, adaptado para o método proposto, se encarrega de de organizar os dados de treinamento do problema em subconjuntos com diferentes níveis de dificuldade.

Com a geração do *pool* seguindo os critérios propostos nesta pesquisa, busca-se melhorar a acurácia em métodos de seleção dinâmica e na combinação dos modelos. Um protocolo experimental, baseado em vinte replicações, foi elaborado para a avaliar e validar o método proposto. Tal abordagem consiste em vinte e oito bases de dados que provêm de diferentes repositórios estudados da literatura. Os resultados obtidos demonstram que o *pool* de classificadores gerados com uso das medidas de complexidade do problema e diversidade no espaço de decisão melhora a classificação em termos de acurácia. Dos experimentos realizados, o método proposto obteve resultados superiores em 20 das 28 bases de dados, quando comparado a outros métodos de GP.

1.1 Objetivos

O objetivo geral deste trabalho consiste em desenvolver um método de geração de *pools* de classificadores baseados em diversidade no espaço de decisão e na complexidade do problema que permita melhorar o desempenho da tarefa de classificação. Para tal, medidas de complexidade combinadas com medidas de diversidade em nível de decisão dos classificadores são utilizadas para compor subconjuntos de amostras com diferentes níveis de dificuldade e diversidade. Para isso, torna-se necessário:

- Identificar as medidas de complexidade mais adequadas para a geração dos subproblemas com níveis diferentes de dificuldades a partir do problema original.
- Definir uma estratégia para criação do *pool* de classificadores.
- Avaliar a acurácia do *pool* gerado em diferentes problemas de classificação.
- Medir o impacto do *pool* gerado com o uso de medidas de complexidade comparado a outros métodos de geração de *pool*, quando aplicados em diferentes abordagens para seleção dinâmica e na combinação de classificadores.

1.2 Hipóteses

Neste trabalho quatro hipóteses foram levantadas.

Hipótese 1: A escolha das medidas de complexidade adequadas para representar a dificuldade dos dados varia de acordo com o problema de classificação.

Hipótese 2: A combinação da diversidade obtida no espaço de complexidade do problema com a diversidade no espaço de decisão dos classificadores contribui positivamente, em termos de acurácia, na geração de *pool* de classificadores.

Hipótese 3: A amostragem baseada na dificuldade do problema, ou seja, orientada por medidas de complexidade permite gerar classificadores diversos que, quando combinados, melhoram o desempenho da tarefa de classificação.

Hipótese 4: O *pool* composto por classificadores treinados em subproblemas, representando diferentes níveis de dificuldade, melhora o desempenho dos sistemas de seleção dinâmica de classificadores.

Na primeira hipótese, as MCs podem contribuir na geração do *pool* de maneira diferente para cada problema de classificação, ou seja, as MCs se comportam de maneira diferente para cada problema de classificação. A segunda hipótese avalia a contribuição da diversidade no espaço de decisão referente ao desempenho em termos de acurácia dos classificadores quando combinada com as MCs. Na hipótese seguinte, a diversidade dos classificadores do *pool* melhora a acurácia na combinação de classificadores como, por exemplo, no voto majoritário. Na quarta e última hipótese, espera-se que o conjunto de

especialistas gerado auxilie positivamente em relação à acurácia nos métodos de seleção dinâmica de classificadores, quando comparado a outros métodos da literatura.

1.3 Contribuições

As contribuições deste trabalho estão listadas abaixo:

- Um novo método para geração de *pool* de classificadores com base na dificuldade e diversidade do problema.
- Avaliação das medidas de complexidade como ferramenta para geração de *pool* de classificadores.
- Definição de critérios para escolha das medidas de complexidade mais adequadas para o problema.
- Avaliação do impacto na acurácia do *pool* gerado com as medidas de complexidade e diversidade nos métodos de seleção e combinação de classificadores.

Trabalho Publicado: MONTEIRO JUNIOR, M. ; BRITTO JR, A. S. ; BARDDAL, J. P. ; OLIVEIRA, L. E. S. ; SABOURIN, R. . Classifier Pool Generation based on a Two-level Diversity Approach. In: International Conference on Pattern Recognition (ICPR), 2020, Milão. Proc. of the International Conference on Pattern Recognition (ICPR). Milão: IAPR, 2020. v. 8.

1.4 Visão geral do documento

Após a introdução realizada no Capítulo 1, apresenta-se na sequência o Capítulo 2, onde se encontra a fundamentação teórica, como a caracterização das medidas de complexidade e diversidade, algoritmos evolutivos, e os métodos de seleção dinâmica de classificadores. No Capítulo 3 é apresentado o estado da arte com trabalhos relacionados à geração de conjuntos de classificadores. No capítulo seguinte (4) é demonstrado o método proposto e cada etapa para geração do *pool* de classificadores. Os experimentos e análise dos resultados alcançados são apresentados no quinto capítulo (5). Por último, o Capítulo 6 discorre sobre a conclusão desta pesquisa, e a continuidade do trabalho em pesquisas subsequentes.

2 Fundamentação Teórica

Dentre os estudos do aprendizado supervisionado encontram-se formas de mensurar a dificuldade de um problema de classificação a partir dos dados de treinamento disponíveis. Essas medidas podem fornecer informações pertinentes antes, durante e depois da criação dos classificadores. Outros conceitos relevantes para o presente trabalho são relacionados à seleção dinâmica de classificadores. Tais métodos são aplicados como forma de validar e avaliar o método proposto.

Sendo assim, este capítulo aborda os conceitos de medidas de complexidade, suas descrições e caracterizações. Também discorre sobre as medidas de diversidade de decisão dos classificadores e ainda discute as diferentes implementações de algoritmos genéticos. Dentro dos sistemas de múltiplos classificadores são apresentados alguns dos principais métodos de seleção dinâmica de classificadores. Estes assuntos são essenciais para o entendimento das técnicas aplicadas no método, na análise e na validação deste trabalho.

2.1 Medidas de complexidade (MC)

De acordo com Ho, Basu e Law (2006), as medidas de complexidade permitem caracterizar a dificuldade de um problema de classificação. Em seu trabalho Lorena et al. (2019) realizaram uma ampla revisão sobre as medidas de complexidade onde propuseram uma divisão das mesmas em seis famílias: sobreposição de características, medidas de linearidade, vizinhança, redes, dimensionalidade e balanceamento de classes. Com os resultados encontrados nos testes preliminares e outras particularidades dos problemas de classificação utilizados nos experimentos (discutidos no Capítulo 5) optou-se pelo uso de um subgrupo dessas medidas. Nos próximos tópicos serão apresentados os descritores de complexidade.

A notação utilizada nas próximas seções é baseada na seguinte situação: Dado um problema de classificação D, (ou parte dele), contendo n pares de exemplos (x_i, y_i), onde $x_i = (x_i f_j, \ldots x_i f_m)$ e $y_i \in \{1, 2 \ldots y^*\}$. Isto é, cada exemplo x_i contêm m atributos (f) e um rótulo y_i de um total de y^* classes.

2.1.1 Medidas de Sobreposição

As medidas de sobreposição, comumente, analisam a sobreposição de atributos dos exemplos do problema em diferentes classes, isto é, realiza o cálculo para encontrar a distância entre os atributos das instâncias do problema.

Muitos algoritmos de classificação utilizam, em seu aprendizado, os atributos do problema para definir a qual classe pertence cada instância. Diante dessa perspectiva é

interessante mensurar a sobreposição dos atributos.

Razão Máxima do Discriminante de Fisher (F1) - A métrica F1 é responsável por encontrar a distância entre os centroides de duas classes distintas, e se baseia na média e no desvio padrão de cada atributo (ORRIOLS-PUIG; MACÌ; HO, 2010). A Equação 2.1 demonstra a Razão Máxima do Discriminante de Fisher de forma geral (LORENA et al., 2019).

$$F1 = \frac{1}{1 + max_{j=1}^{m}\, r_{f_j}} \tag{2.1}$$

em que, r_{f_j} é uma razão que representa os valores discriminantes para cada característica e f_j significa cada caraterística contida em x_i. Os valores discriminantes são atributos que conseguem distinguir as classes do problema, como por exemplo, atributos binários.

A medida F1 é aplicada para conjuntos de dados com dois rótulos (problemas binários). Entretanto Mollineda, Sánchez e Sotoca (2005) propuseram uma equação onde é possível calcular r_{f_j} para conjuntos de dados com múltiplas classes. A Equação 2.2 demonstra como calcular o valor de r_{f_j}, sendo que, n_{c_j} é o número de exemplos na classe y_j, $\mu_{y_j}^{f_i}$ denota a média do atributo f_i sobre os exemplos da classe y_j, o μ^{f_i} é a média de cada característica f_i em todas as classes do problema. Finalmente $x_{l_i}^{j}$ representa o valor individual (l) do atributo f_i para um exemplo da classe y_j.

$$r_{f_i} = \frac{\sum_{j=1}^{c} c_j (\mu_{y_j}^{f_i} - \mu^{f_i})^2}{\sum_{j=1}^{c} \sum_{l=1}^{c_j} (x_{li}^{j} - \mu_{y_j}^{f_i})^2} \tag{2.2}$$

Mediante a fórmula F1 obtém-se os valores de sobreposição de atributos de classes diferentes. Resultados que tendem ao infinito indicam que existe pelo menos um atributo que consegue distinguir instâncias de diferentes rótulos.

Utiliza-se a métrica F1 em situações em que os atributos das instâncias de uma determinada classe estão distribuídos perpendicularmente às instâncias de outras classes (LORENA et al., 2019), ou seja, em hiperplanos perpendiculares [1]. Em situações em que os atributos das instâncias de classes diferentes estão em outro hiperplano, utiliza-se a métrica equivalente F1v.

Vetor direcional da Razão Máxima do Discriminante de Fisher (F1v) - A medida F1v é utilizada para determinar o vetor de projeção que separa duas classes de acordo com a equação de Fisher. Aplica-se nos casos em que os atributos de um problema não estão perpendiculares entre exemplos de rótulos diferentes. Malina (2001) analisou hiperplanos distintos de problemas binários e não binários propondo a Equação 2.3 para encontrar a distância entre os centroides de classes diferentes.

$$F1v = \frac{d^S B d}{d^S W d} \tag{2.3}$$

[1] Generalização do plano em diferentes números de dimensões. (WEISSTEIN et al., 2007)

em que, d é um vetor direcional que armazena os dados que são projetados para maximizar a dispersão das classes, S é um subproblema, B é a matriz dos atributos entre as classes e W é a matriz dos atributos no próprio rótulo. A Equação 2.4 exemplifica o cálculo de d, onde ce_{y_j} é o centroide (vetor médio) da classe y_j, e W^{-1} é a matriz inversa de W. Os centroides são comparados par a par, nas próximas Equações (2.4, 2.5, 2.6), para representar classes distintas foram utilizados os valores de j como 1 e 2 (y_1 e y_2).

$$d = W^{-1}(ce_{y1} - ce_{y2}) \tag{2.4}$$

O cálculo dos fatores B e W estão descritos nas Equações 2.5 e 2.6 respectivamente. Na Equação 2.6, xp_{y_j} denota a proporção de exemplos da classe y_j e Δ_{y_j} representa a matriz dos atributos para classe y_j.

$$B = (ce_{y1} - ce_{y2})(ce_{y1} - ce_{y2})^S \tag{2.5}$$

$$W = xp_{y_1}\Delta_{y_1} + xp_{y_2}\Delta_{y_2} \tag{2.6}$$

Assim como F1, F1v demonstra a sobreposição dos atributos de diferentes classes, porém em diferentes planos (LORENA et al., 2019). As medidas F1 e F1v retornam valores que partem de zero até $+\infty$. Resultados próximos a zero representam problemas mais complexos.

Volume da região de sobreposição (F2) - Segundo Ho e Basu (2002) e Lorena et al. (2019), a medida F2 também mensura a distância e verifica o quão sobrepostos estão os valores de um atributo entre duas classes diferentes. A medida F2 é dada pelo mínimo e máximo de cada atributo de mesmo rótulo. O intervalo de sobreposição é calculado normalizando os atributos de ambas as classes e os valores obtidos são multiplicados. A determinação da métrica F2 é obtida pela Equação 2.7 que relaciona os fatores $overlap(f_j)$ e $range(f_j)$.

$$F2 = \prod_{j}^{m} \frac{overlap(f_j)}{range(f_j)} = \prod_{j}^{m} \frac{max\left\{0, minmax(f_j) - maxmin(f_j)\right\}}{maxmax(f_j) - minmin(f_j)} \tag{2.7}$$

sendo que:

- m é o número de atributos.
- f é o atributo.
- $minmax(f_j) = min(max(f_j^{y_1}), max(f_j^{y_2}))$.
- $maxmin(f_j) = max(min(f_j^{y_1}), min(f_j^{y_2}))$.
- $maxmax(f_j) = max(max(f_j^{y_1}), max(f_j^{y_2}))$.
- $minmin(f_j) = min(min(f_j^{y_1}), min(f_j^{y_2}))$.

A demonstração da Equação 2.7, leva em consideração duas classes y_1 e y_2 ($y_j \in \{1, 2\}$). Os valores de $max(f_j^{y_i})$ e $min(f_j^{y_i})$ são os máximos e mínimos dos atributos da classe y_i, respectivamente. Se um atributo não estiver em sobreposição, o valor de

F2 será zero. Na Figura 1 encontra-se a área de sobreposição determinada pela medida F2. A mesma figura representa um problema de dois atributos e duas classes, onde f significa os atributos dos exemplos do *dataset*. Sendo destacado na Figura 1 a área onde os atributos se encontram sobrepostos. Nesta área encontram-se os atributos f_1 e f_2 nas classes triângulo e círculo, nota-se que não é possível definir claramente a qual classe pertence cada característica, pois os atributos se encontram sobrepostos em duas classes diferentes.

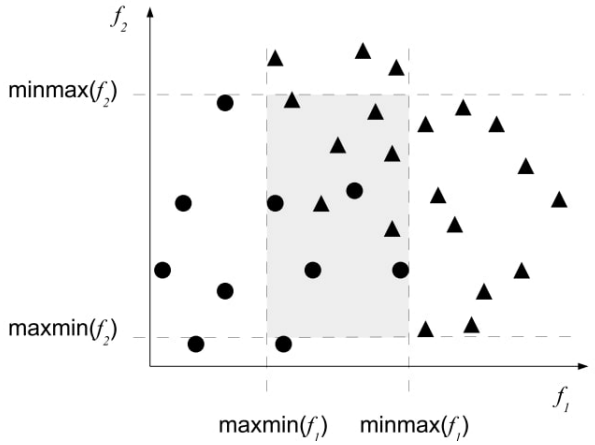


Figura 1 – Região de sobreposição determinada pela métrica F2. Os atributos f_1 e f_2 se encontram sobrepostos, pois não é possível determinar a qual classe eles pertencem (círculo ou triângulo).

Fonte: (LORENA et al., 2019).

Diferente das métricas F1 e F1v, o resultado de F2 varia entre zero e um. Valores próximos a um denotam problemas considerados mais difíceis, ou seja, uma correlação direta entre o valor encontrado com a complexidade.

Eficiência máxima de um atributo (F3) - Diferente das medidas anteriores, o cálculo da eficiência máxima de um atributo verifica se há sobreposição entre instâncias de classes diferentes para cada atributo e não do conjunto de característica na totalidade. Se houver sobreposição, as classes são consideradas ambíguas nesta região. Dessa forma, o percentual que cada característica contribui para separação de duas classes é o resultado da medida F3.

De acordo com Orriols-Puig, Macì e Ho (2010) o cálculo de F3 é feito seguindo a heurística: para cada atributo, considera-se uma região de sobreposição (onde o valor do atributo é o mesmo ou muito próximo para classes distintas), o resultado de F3 será a razão entre o número de instâncias do conjunto que não estão nesta região, sobre o número total de instâncias. O valor máximo encontrado será o valor de F3.

A Equação 2.8 representa a heurística de F3, onde m é o número de atributos, n_o representa o número de exemplos do problema que estão na região de sobreposição para um atributo qualquer e n é número total de par de exemplos (x, y). O fator $n_o(f_i)$

corresponde à região de sobreposição para o atributo f_i, essa região é calculada de acordo com a Equação 2.9. Na Equação 2.9 o membro x_{ji} representa o exemplo x_j que está a região de sobreposição do atributo f_i. Concluindo, $maxmin(f_i)$ e $minmax(f_i)$ possuem a mesma definição descrita em F2 (Seção 2.1.1). A variável b é um indicador de função, seu valor será um se o exemplo estiver em uma região de sobreposição ou zero se não estiver, isto é, incrementa a soma ou não (LORENA et al., 2019).

$$F3 = \min_{j=1}^{m} \frac{n_o(f_i)}{n} \tag{2.8}$$

$$n_o(f_i) = \sum_{j=1}^{n} b(x_{ji} > maxmin(f_i) \wedge x_{ji} < minmax(f_i)) \tag{2.9}$$

Os valores obtidos de F3 são normalizados no intervalo de zero a um. Os resultados de F3 possuem uma correlação inversa à complexidade, ou seja, valores maiores de F3 significam problemas considerados mais fáceis (menos complexos).

Eficiência coletiva dos atributos (F4) - Na eficiência coletiva dos atributos, é utilizado o conjunto de atributos mais discriminante de cada classe, diferenciando-se assim da métrica F3. Para o cálculo de F4, separa-se todos os exemplos que um atributo consegue distinguir. Dos exemplos que sobraram, encontra-se o atributo que mais distingue as classes do problema. O procedimento é repetido até que todos os exemplos tenham sido classificados ou que não haja mais atributos para tentar fazer a distinção entre as classes. F4 retornará a proporção de exemplos que foram discriminados pelos atributos (ORRIOLS-PUIG; MACÌ; HO, 2010; LORENA et al., 2019).

A Equação 2.10 determina o valor de F4, onde D é o *dataset*, L é um contador de repetições entre [1, m]. O fator $n_o(f_{min})$ é o resultado parcial da métrica F3 (Seção 2.1.1) em diferentes repetições. Resultados maiores, próximos ou iguais a um, de F4 representam problemas menos complexos.

$$F4 = \frac{n_o(f_{min}(D_L))}{n} \tag{2.10}$$

2.1.2 Medidas de Vizinhança

As medidas de vizinhança usam a distância entre exemplos, e/ou características, para demonstrar o comportamento das regiões de fronteira no espaço entre diferentes rótulos, além da separabilidade dessas classes, isto é, o quão difícil é separá-las. As regiões de borda (fronteira) entre classes, são locais de divergência entre classificadores e as medidas desta seção mensuram essas regiões. Assim como as métricas de sobreposição, as medidas de vizinhança só aceitam problemas com atributos numéricos. As MCs que fazem parte desses grupos são: N1, N2, N3, N4, T1 e LSCAvg.

Fração de Pontos na Região de Fronteira (N1) A medida N1 define a porcentagem de exemplos que está próxima da linha de fronteira entre diferentes rótulos do

problema. Para encontrar os limites entre as bordas, a métrica N1 aplica uma Árvore de Cobertura Mínima ACM (*Minimum Spanning Tree*) (GRAHAM; HELL, 1985). Esse algoritmo gera um grafo conectando todos os elementos do conjunto de forma que a soma das conexões seja a menor possível. As arestas dessa árvore estão conectadas em duas classes diferentes representam os exemplos que estão na linha de fronteira (HO; BASU, 2002; LORENA et al., 2019; LILEIKYTE; TELKSNYS, 2011).

Esse método é sensível a *outliers*, gerando árvores diferentes a cada nova iteração e, em alguns casos, poderá gerar árvores complexas para problemas que são linearmente separáveis, elevando o custo computacional. A Figura 2 representa a conexão entre instâncias de duas classes geradas pela ACM onde aparecem os nós simbolizando as classes do problema (cada cor é uma classe). As arestas conectadas em nós de classes diferentes (representadas por conexões vermelhas), determinam a fronteira entre esses rótulos, já as arestas azuis representam a ligação entre instâncias de mesma classe.

A Equação 2.11 define a soma das arestas que estão na região de fronteira, n denota o número de exemplos do problema, x_i e x_j são instâncias do problema, b é um indicador de função, assim como na medida F3, ele representa os valores entre zero e um. As variáveis y_i e y_j determinam os rótulos de cada instância.

Os resultados obtidos da métrica N1 situam-se no intervalo entre zero e um. Esses resultados possuem uma correlação direta com a complexidade, ou seja, valores maiores de N1 significam uma complexidade maior.

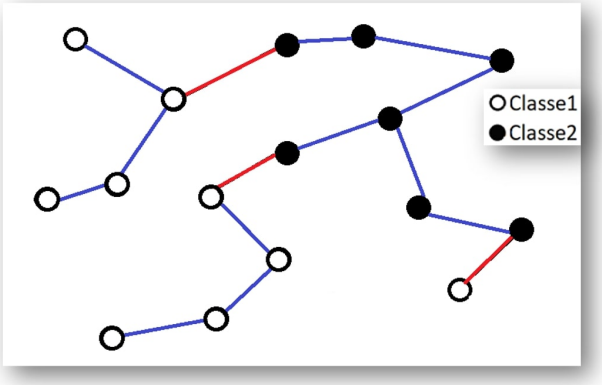


Figura 2 – Exemplo de uma ACM. As classes são representadas pelos círculos. Destacam-se as restas ligando classes diferentes do problema (arestas vermelhas).

Fonte: Adaptado de (HO; BASU, 2002).

$$N1 = \frac{1}{n}\sum_{i=1}^{n} b((x_i, x_j) \in ACM \wedge y_i \neq y_j) \tag{2.11}$$

Relação dos vizinhos mais próximos intra/extra classes (N2) - O descritor N2 avalia a interseção de duas classes por meio da distância entre instâncias de classes diferentes do problema. O cálculo de N2 consiste na soma das distâncias de apenas um vizinho mais próximo de uma instância na sua classe, pela razão da soma das distâncias do

vizinho mais próximo da outra classe (HO; BASU, 2002; BARELLA et al., 2018; LORENA et al., 2019). Esses passos são feitos para todas as instâncias do problema.

A Equação 2.12 corresponde ao cálculo da medida N2, onde x_i representa cada instância i do problema, $intraDist$ representa a distância do vizinho mais próximo do exemplo x_i na mesma classe e $interDist$ representa a distância do vizinho mais próximo que pertence a uma classe diferente da sua. Por último, n representa o total de instâncias e rótulos do problema. A forma de calcular a distância pode variar, no entanto, é mais comum o uso da Distância Euclidiana.

$$N2 = \frac{\sum_{i=0}^{n} intraDist(x_i)}{\sum_{i=0}^{n} interDist(x_i)} \tag{2.12}$$

A métrica N2 possui valores que variam de zero até $+\infty$. Valores menores de N2 representam baixa complexidade do problema, pois a soma das distâncias de todos os exemplos de uma classe será menor que soma das distâncias dos exemplos de outra classe.

Taxa de erro do classificador dos vizinhos mais próximos (N3) - A taxa de erro do classificador dos vizinhos mais próximos, refere-se à taxa de erro do classificador 1NN estimada por *leave-one-out*, para um vizinho da instância de teste (HERNÁNDEZ-REYES; CARRASCO-OCHOA; MARTÍNEZ-TRINIDAD, 2005; LORENA et al., 2019; MOLLINEDA; SÁNCHEZ; SOTOCA, 2005). Resultados de N3, próximos ou iguais a um, indicam que grande parte dos elementos do conjunto estão próximos da região de fronteira, isto significa um problema mais complexo. O cálculo de N3 está representado na Equação 2.13, em que NN é o vizinho mais próximo da predição do classificador para o exemplo x_i usando todas as instâncias de treinamento (LORENA et al., 2019).

$$N3 = \frac{\sum_{i=1}^{n} b(NN(x_i) \neq y_i)}{n} \tag{2.13}$$

Não-linearidade do classificador KNN em dados sintéticos (N4) - A medida N4 retorna a taxa de erro do classificador KNN. Para tanto, a métrica N4 usa um problema sintético criado a partir de um conjunto de dados do *dataset*. Essa medida utiliza um conjunto de treinamento para a criação de um conjunto de teste sintético. Esse conjunto é criado por meio da interpolação entre pares escolhidos aleatoriamente dentro de uma mesma classe, conforme abaixo descrito (HO; BASU, 2002; ORRIOLS-PUIG; MACÌ; HO, 2010).

A interpolação é feita retirando aleatoriamente dois exemplos do conjunto de treinamento da mesma classe. Após, cria-se um exemplo sintético entre as duas instâncias e o processo se repete até que todos os exemplos sejam combinados. O conjunto sintético servirá como teste para o classificador KNN. A Equação 2.14 define o cálculo dessa MC, onde Inp representa os números de interpolações realizadas, b o resultado do indicador de função, que pode ser zero ou um, $resKNN$ representa a predição do algoritmo KNN para

a instância sintética x'_i e y'_i é o rótulo do exemplo sintético (LORENA et al., 2019).

$$N4 = \frac{1}{Inp} \sum_{i=1}^{Inp} b(resKNN(x'_i) \neq y'_i) \tag{2.14}$$

A taxa de erro do classificador KNN será o valor da métrica N4, cujo resultado é normalizado entre zero e um. Resultados elevados da medida N4 significam problemas de classificação com alta complexidade.

Fração de esferas de cobertura máxima (T1) - A fração de esferas de cobertura máxima (T1) corresponde ao número de esferas necessárias para envolver cada classe. T1 consiste em criar esferas, cujo centro está em uma instância de uma classe escolhida aleatoriamente. O raio dessas circunferências será incrementado até que sua borda seja tocada por alguma instância de outra classe. Após todas as instâncias serem usadas para criação das esferas, eliminam-se as esferas que estão completamente sobrepostas por outras maiores. O valor de T1 é dado pelo número de esferas para envolver uma classe, dividido pelo total de instâncias do problema (LORENA et al., 2019; HO; BASU, 2002; HO; BASU; LAW, 2006).

Essa métrica possui valores diferentes para cada replicação, isso ocorre porque a instância inicial é escolhida ao acaso, logo o número e o tamanho das circunferências podem variar. A Figura 3 representa a linha de fronteira entre duas classes estabelecidas por T1, a cor das esferas representa as classes do problema. Ainda na Figura 3, o resultado da medida T1 será o total de esferas (12) divido pelo total de instâncias n. A medida T1 possui resultados entre zero e um, logo quanto maior o valor de T1, maior será a complexidade do problema.

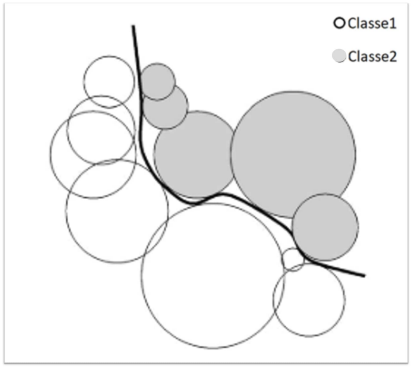


Figura 3 – Esferas de T1 para cada instância. Em destaque a linha de fronteira entre as esferas de cores diferentes. Cada cor representa uma classe do problema de classificação.

Fonte: (HO; BASU, 2002).

Cardinalidade média do conjunto local (do inglês *local set average cardinality measure LSCAvg*) - A última métrica pertencente a esse grupo é a cardinalidade média do conjunto local. De acordo com Leyva, González e Perez (2014), o conjunto local

de um exemplo é definido como um conjunto de instâncias cuja distância dos vizinhos de x_i é menor do que a distância dele para a instância de outra classe. O tamanho do conjunto será determinado de acordo com a menor distância *dist* entre os exemplos x_i e x_j. A Equação 2.15 define o conjunto local (CL), em que $ne(x_i)$ é o vizinho mais próximo da instância da outra classe.

$$CL(x_i) = \{|x_j| dist(x_i, x_j) < dist(x_i, ne(x_i))\} \tag{2.15}$$

A média dos grupos, formados por CL, corresponde ao valor de LSCAvg. Resultados, próximos a zero, encontrados no cálculo de LSCAvg representam problemas mais complexos.

2.1.3 Medidas de linearidade

Medidas de linearidade mensuram o grau de dificuldade que um problema apresenta para ser separado linearmente. Nessas medidas são utilizados classificadores lineares. Os autores Orriols-Puig, Macì e Ho (2010) e Lorena et al. (2019) utilizam o classificador *Support Vector Machine* (SVM) (CRISTIANINI; SHAWE-TAYLOR et al., 2000) para determinar as grandezas de linearidade.

Soma das distâncias do erro do classificador linear (L1) - A métrica L1 consiste na soma das distâncias entre exemplos rotulados incorretamente de acordo com o hiperplano traçado pelo classificador. A soma das distâncias desses exemplos, dividido pelo total de instâncias (n) representa o valor de L1 (HO; BASU, 2002; LORENA et al., 2019). A Equação 2.16 descreve como é calculada a L1, em que $dist(h_i, \epsilon_i)$ é uma função que retorna distância entre o hiperplano (h) e as classes classificadas incorretamente (ϵ) pelo classificador linear.

$$L1 = \frac{1}{n} \sum_{i=1}^{n} dist(h_i, \epsilon_i) \tag{2.16}$$

Taxa de erro do classificador linear (L2) - A métrica L2 é similar a medida L1, difere-se por não utilizar a distância das instâncias rotuladas incorretamente até o hiperplano traçado pelo classificador. L2 utiliza somente a taxa de erro do classificador linear (HO; BASU, 2002).

A Equação 2.17 define a L2, onde h é a representação da predição do classificador linear, x_i é o exemplo cujo rótulo é y_i. O resultado da classificação e a comparação com o rótulo real retorna um valor binário representado por b, sendo que 1 são as instâncias classificadas corretamente e 0 incorretamente. Finalizando, o resultado de L2 será a razão da soma das instâncias classificadas incorretamente, pelo total das instâncias (n). A medida L2, que também é normalizada entre zero e um e possui resultado diretamente proporcional à complexidade.

$$L2 = \frac{\sum_{i=1}^{n} b(h(x_i) \neq y_i)}{n} \tag{2.17}$$

Não-linearidade de um classificador linear (L3) - Assim, como a métrica N4 (Seção 2.1.2), L3 utiliza um subconjunto sintético derivado de um problema de classificação. O subconjunto sintético é formado pela interpolação das amostras de treino. Esse conjunto será utilizado como teste para o classificador linear (HO; BASU; LAW, 2006).

O valor final de L3 será o resultado da medida L1 (Seção 2.1.3), porém utilizando o conjunto de teste sintético. Da mesma maneira que as outras medidas de linearidade, o resultado gerado pela métrica L3, possui uma correlação direta com a complexidade.

A Tabela 1 apresenta um resumo das métricas apresentadas, onde a primeira coluna descreve o nome da medida, a segunda é a família a qual ela pertence. Na terceira coluna é apresentado o intervalo do resultado obtido do cálculo da MC, e por último a relação entre a complexidade e o valor resultante. Na coluna complexidade, "↓" significa que a complexidade é inversamente proporcional ao resultado da MC, diferentemente das medidas onde o símbolo é "↑" que significa resultados diretamente proporcionais.

Tabela 1 – Resumo das medidas de complexidade apresentadas na seção

Nome	**Família**	**Intervalo de variação**	**Proporção à complexidade**
F1	Sobreposição	0 até $+\infty$	↓
F1v	Sobreposição	0 até $+\infty$	↓
F2	Sobreposição	0 até 1	↑
F3	Sobreposição	0 até 1	↓
F4	Sobreposição	0 até 1	↓
N1	Vizinhança	0 até 1	↑
N2	Vizinhança	0 até $+\infty$	↑
N3	Vizinhança	0 até 1	↑
N4	Vizinhança	0 até 1	↑
T1	Vizinhança	0 até 1	↑
LSCAvg	Vizinhança	$\frac{1}{x_n}$ até 1	↓
L1	Linearidade	0 até 1	↑
L2	Linearidade	0 até 1	↑
L3	Linearidade	0 até 1	↑

2.2 Medidas de diversidade

As medidas de diversidade não possuem uma definição amplamente aceita na literatura (CAVALCANTI et al., 2016). Muitas medidas de diversidade mensuram o quanto dois ou mais classificadores diferem entre si. Logo, pode-se calcular as diferenças entre todos os classificadores do *pool*.

Para calcular algumas dessas medidas, é necessário entender a combinação do total de rótulos preditos por dois classificadores para um conjunto de exemplos de teste, apresentado na Tabela 2. A predição dos classificadores é representada pelo símbolo Φ. Na Tabela 2 N é um vetor de rótulos, em que N^0 representa uma instância predita

incorretamente e N^1 são as instâncias preditas corretamente (KUNCHEVA; WHITAKER, 2003).

Tabela 2 – Representação da combinação do total de rótulos preditos por dois classificadores

	Φ_i correta (1)	Φ_i incorreta (0)
Φ_j correta (1)	N^{11}	N^{10}
Φ_j incorreta (0)	N^{01}	N^{00}

Fonte: Adaptado de (KUNCHEVA; WHITAKER, 2003)

Q-estatística (Qs)- A medida Q-estatística é definida de acordo com a Equação 2.18. O resultado dessa medida está entre -1 e 1, onde 0 significa que os dois classificadores são independentes, ou seja, são especialistas em regiões diferentes do problema. O resultado 1 quer dizer que ambos os classificadores possuem resultados demasiadamente próximos, logo podem ser considerados iguais. Resultados iguais a -1 definem predições diferentes para os dois, isto é, os classificadores divergem nas suas predições (YULE, 1900).

$$Qs_{(\Phi_i,\Phi_j)} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}} \tag{2.18}$$

Coeficiente de Correlação (CoC)- Outra métrica é a Coeficiente de Correlação para dois classificadores, definida pela Equação 2.19. A definição dos seus resultados é similar aos da Q-estatística (KUNCHEVA et al., 2003).

$$CoC_{(\Phi_i,\Phi_j)} = \frac{N^{11}N^{00} - N^{01}N^{10}}{\sqrt{(N^{11} + N^{10})(N^{01} + N^{00})(N^{11} + N^{01})(N^{10} + N^{00})}} \tag{2.19}$$

Medida de desacordo (do inglês *Disagreement Measure* DM) – Esta medida é a proporção de exemplos com diferentes predições realizadas por Φ_i i e Φ_j. Seu valor é calculado pela Equação 2.20. Seu valor varia de 0 a 1, com valores mais altos indicando mais diversidade (SKALAK, 1996).

$$MD_{(\Phi_i,\Phi_j)} = \frac{N^{01}N^{10}}{N^{00} + N^{01} + N^{10} + N^{11}} \tag{2.20}$$

Dupla Falta (Df) – É a última medida apresentada nessa seção. Essa métrica calcula a proporção de todos os exemplos classificados incorretamente por dois modelos. O valor da Df é calculado de acordo com a Equação 2.21, onde Φ_n é o total de instâncias preditas. Quanto menor o resultado obtido por esta medida, mais diferentes são os classificadores (GIACINTO; ROLI, 2001a).

$$Df_{(\Phi_i,\Phi_j)} = \frac{N^{00}}{N^{00} + N^{01} + N^{10} + N^{11}} \tag{2.21}$$

As medidas de diversidade representam uma análise do comportamento e criação dos modelos. Estas medidas mensuram a diversidade par-a-par entre os modelos. A Tabela

3 resume o comportamento das medidas de diversidade, em que, a diversidade entre os classificadores será maior se o resultado do cálculo da medida for menor ("↓"), ou diretamente proporcional quando for maior ("↑").

Tabela 3 – Resumo das medidas de diversidade pareadas.

Nome	**Intervalo de variação**	**Proporção à diversidade**
Q-estatística	-1 até 1	↓
Coeficiente de Correlação	-1 até 1	↓
Medida de desacordo	0 até 1	↑
Dupla Falta	0 até 1	↓

Existem outras maneiras de calcular a diversidade entre os classificadores, como o grupo de medidas de diversidade não pareadas. Neste caso, as medidas levam em consideração todo o conjunto de classificadores. Destacam-se as medidas de Entropia, *Kohavi-Wolpert variance* (KOHAVI; WOLPERT et al., 1996) e a concordância entre os classificadores (LOONEY, 1988).

2.3 Algoritmos Evolutivos

Nessa seção serão abordados os conceitos sobre algoritmos evolutivos, com foco principal em Algoritmos Genéticos e Algoritmos Genéticos Multiobjetivos. Serão relatadas algumas aplicações e trabalhos relacionados.

De acordo com Eiben e Smith (2015), os algoritmos evolutivos seguem, em sua maioria, o seguinte conceito derivado da natureza: dada a população de indivíduos dentro de um ambiente com poucos recursos, logo existirá uma disputa entre membros da população por esses recursos. Essa competição causará a seleção natural dos indivíduos mais fortes. Os indivíduos com melhor aptidão serão os mais aptos à reprodução, por consequência transmitirão suas qualidades para as próximas gerações. Esta ideia foi inspirada por (DARWIN, 2009), onde são relatados os conceitos de evolução das espécies.

2.3.1 Algoritmos Genéticos

O exemplo mais conhecido de algoritmos evolutivos são os algoritmos genéticos (AGs). Os AGs seguem os princípios descritos no início desta seção. E têm como objetivo transmitir as qualidades dos indivíduos para as futuras gerações. Os AGs utilizam operadores como: cruzamento, mutação, seleção e uma função de avaliação chamada *fitness* ou função objetivo.

Konak, Coit e Smith (2006) salientam que, o operador principal é o cruzamento, no qual ocorre a troca de genes entre dois indivíduos. Nos AGs, dois indivíduos chamados "pais", são escolhidos de forma aleatória na população para a troca de características,

ou seja, os cromossomos. Para atingir uma boa aptidão entre os indivíduos, Lacerda e Carvalho (1999) recomendam uma taxa de cruzamento entre 60% e 90%.

A mutação ocorre quando um, ou mais, genes são modificados nos cromossomos de um indivíduo, trazendo alterações que contribuem ou não para um membro da sociedade. O papel fundamental da mutação é inserir a diversidade na população aumentando a probabilidade de um indivíduo aprimorar suas habilidades individuais sem depender da herança genética dos seus pais. Entretanto, essa alteração causada pela mutação deve ser aplicada em uma pequena parcela da população. Caso a taxa mutação seja alta é possível que os indivíduos gerados não tenham semelhança com seus pais (ROSA; LUZ, 2009). Segundo Lacerda e Carvalho (1999) a taxa de mutação é interessante estar entre 0,5% e 5%. Após a etapa de cruzamento surge uma nova população formada pelos filhos. Essa nova população é conhecida como *offspring*, ou descendentes.

Em um algoritmo genético deve-se definir os critérios para a seleção dos membros mais aptos da população. A função de *fitness* é responsável por avaliar os atributos dos indivíduos. Normalmente, a função objetivo é utilizada para maximizar uma habilidade, mas também pode ser aplicada para outras finalidades como, por exemplo, diferenciar os indivíduos mediante seus genes.

O último operador dos AGs é a seleção. Há diversas maneiras de fazer a seleção dos indivíduos na população. As principais técnicas de seleção são: aleatória, seleção dos melhores, dos piores, torneio, roleta, entre outras. Uma das variáveis a ser considerada nesses métodos é a quantidade de indivíduos escolhidos. Esse número varia de acordo com a aplicação do AG.

A seleção aleatória seleciona os indivíduos sem considerar os resultados da função de *fitness*, diferentemente das demais estratégias citadas anteriormente. A abordagem da seleção dos melhores, forma um *ranking* dos melhores indivíduos de acordo com função objetivo. As entidades selecionadas, que irão compor a nova geração, estão nas primeiras posições.

Ao contrário da técnica anterior, a seleção dos piores, escolhe os últimos classificados do *ranking*. As funções de torneio e roleta unem a aleatoriedade com o modelo de *ranking*. O objetivo dessas duas técnicas é evitar a perda de diversidade na sociedade. Outro objetivo é o de impedir que membros com habilidades secundárias sejam descartados. Na função torneio, são selecionados aleatoriamente indivíduos da população, dentro deste grupo são escolhidos os melhores. Neste caso, os indivíduos com pouca habilidade podem ser escolhidos para compor a amostra formando um *ranking* que, não necessariamente, engloba os melhores membros da população.

Já no método roleta, os membros da sociedade recebem pesos, de acordo com o resultado do *fitness*, que aumentam suas probabilidades de escolha. Contudo, essa técnica não garante que somente os melhores sejam escolhidos, contribuindo para os integrantes da população menos favorecidos permaneçam nas próximas gerações.

Para organizar os operadores de um AG, deve-se seguir as etapas descritas por Holland e Goldberg (1989) e que estão demonstradas no fluxograma da Figura 4. Onde expõe que, normalmente um AG é descrito como um *loop* principal, em que, são realizadas todas as operações descritas anteriormente. Na Figura 4, i representa um contador de gerações G e r^* o número de gerações, P_t denota o conjunto de indivíduos, P^* a população formada pelos filhos e P' como a população final. Os indivíduos são representados por I, u_n é o número de filhos que cada geração deve criar, R é um vetor que contém o resultado do *ranking* encontrado pela função objetivo ($\mathcal{F}$) e R' o *ranking* formado pelos filhos. Por último, Q representa a união entre pais e filhos.

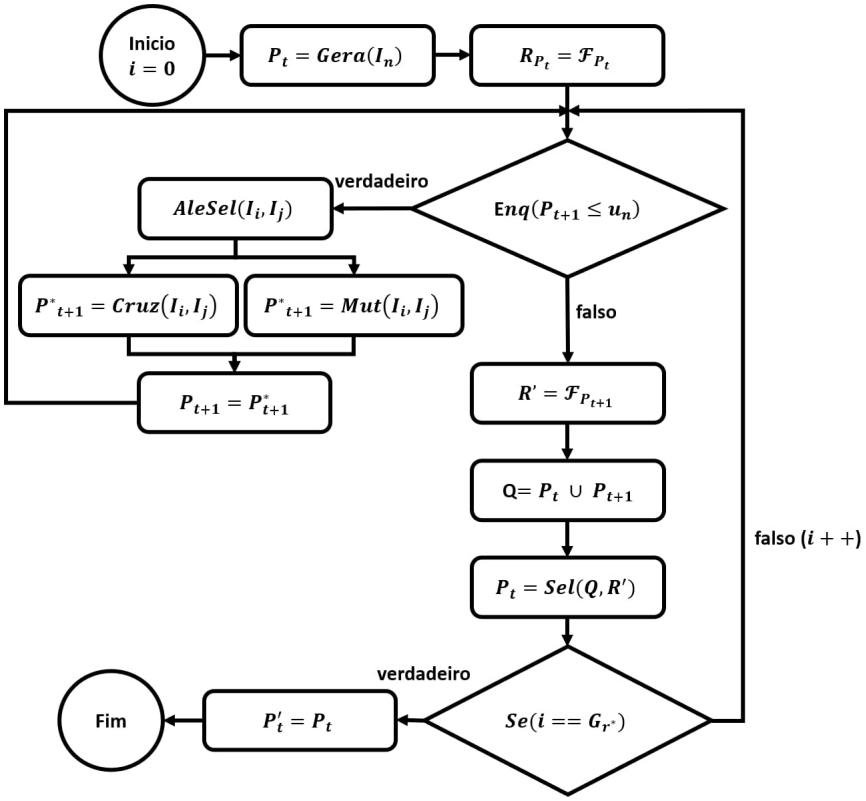


Figura 4 – Representação de um algoritmo genético.

Fonte: O autor.

As funções do algoritmo presente na Figura 4 são: *Gera* função responsável por originar os indivíduos na primeira geração. O tamanho inicial da população pode variar de acordo com a aplicação; *AleSel* é função que seleciona pares de indivíduos da população aleatoriamente; $Cruz$ e Mut são as funções de cruzamento e mutação respectivamente; *Sel* seleciona os melhores membros do grupo Q, onde Q representa o *offspring*. O resultado da função *Sel* poderá conter pais e filhos na próxima geração.

O algoritmo AG começa com a geração da população inicial, seguido de sua avaliação pelo *Fitness*. No próximo passo, inicia–se o laço de cruzamento e mutação até que o número de filhos seja atingido. O resultado desse *loop* é a criação de uma nova população. Novamente, os membros da população formada pelos filhos são avaliados pela

função objetivo, posteriormente as duas populações são unidas. Os melhores indivíduos são selecionados para compor a população final. Caso o número de gerações não seja atingido, o algoritmo retorna para a etapa de cruzamento e mutação.

O elitismo é um operador que nem todas as variações dos AGs possuem. Esse operador é utilizado para manter os melhores cromossomos na população. O melhor membro da população segue para próxima geração independentemente do método de seleção. Se a população for grande, mais do que um elemento pode ser submetido ao elitismo.

Uma das limitações dos AGs tradicionais é a otimização de apenas um objetivo. Em algumas situações, existe a necessidade de otimizar mais de um objetivo de maneira simultânea. Ocorre na maioria das vezes, que os objetivos não se comportam de maneira similar, ou seja, um objetivo pode ser linear crescente e o outro decrescente, ou até mesmo seguirem outro tipo de convergência. Essas diferenças tornam quase impossível a otimização de mais de um objetivo pelos métodos de seleção e *fitness* apresentados nesta seção. Os algoritmos genéticos multiobjetivos utilizam outras técnicas de seleção de indivíduos para ponderar as diferenças entre os objetivos da função de *fitness*.

2.3.2 Algoritmos Genéticos Multiobjetivos

Essa seção trata dos algoritmos genéticos multiobjetivos. Essa categoria de algoritmos evolutivos se distingue basicamente dos AGs tradicionais na função objetivo e na função de seleção de indivíduos.

Pode-se aludir que Algoritmos Genéticos Multiobjetivos (AGM) seguem os mesmos princípios dos AGs. Os passos descritos por (HOLLAND; GOLDBERG, 1989) podem ser aplicados para representar os AGMs, assim como as funções apresentadas na Seção 2.3.1.

Para solucionar o problema das funções com multiobjetivos, descritos na subseção 2.3.1, utilizam-se, geralmente, a Curva de Pareto (CP). A CP tem dois objetivos: minimizar a distância ótima para a Curva de Pareto e maximizar a diversidade das soluções geradas (ZITZLER; LAUMANNS; THIELE, 2001). Muitos algoritmos de seleção usam o critério de dominância, ou seja, determinam os indivíduos dominados e não dominados. Os tópicos a seguir explicam as características necessárias para um indivíduo ser não dominado para duas soluções onde, os objetivos são representados por O, o símbolo Os_i são as soluções i encontradas para cada objetivo O_j sendo que, j representa cada objetivo ($j \in \{1,2, \ldots, z\}$) e por último, z é o total de objetivos.

- As soluções de Os_1 não podem ser piores que as de Os_2 para todos os objetivos, ou $O_j(Os_1)$ ser melhor que $O_j(Os_2)$ para todo objetivo j.
- As soluções de Os_1 são estritamente melhores que as de Os_2 em pelo menos um objetivo, ou $O_j(Os_1)$ é melhor que $O_j(Os_2)$ para pelo menos um objetivo (GHOSH; DAS, 2008).

Os indivíduos que atendem os critérios acima descritos são considerados não dominados. Com base nesses indivíduos é possível aplicar diferentes técnicas de seleção das entidades da população baseados na Curva de Pareto.

O primeiro algoritmo de seleção de indivíduos em um AGM presente na literatura é *Vector Evaluated GA* (VEGA), proposto por Schaffer (1985). No algoritmo VEGA, a população (P_t) é dividida aleatoriamente em r subpopulações de tamanhos iguais. P_s representa as subpopulações. Cada subpopulação P_{s_i}, onde $i \in \{1, 2, 3, 4....r\}$ é avaliada pela função objetivo ($\mathcal{F}_j$), em que $\mathcal{F}_j = P_s(I_{lj})$ e I representa cada indivíduo i em j funções objetivas. Os indivíduos são selecionados de acordo com um valor proporcional definido na execução do algoritmo. Ao final de cada geração, os indivíduos se tornam especialistas em um objetivo, esses indivíduos passarão pelo cruzamento e mutação gerando um *offspring* híbrido. Esse método é bastante simples, haja visto que, utiliza um *fitness* para cada objetivo, além disso, não utiliza o conceito de não dominados.

O VEGA dispõe de funções objetivos individuais para conseguir garantir a diversidade dos membros da população, para então cruzar os indivíduos que convergiram para objetivos diferentes. Outros métodos de seleção de indivíduos inspirados no VEGA foram desenvolvidos depois como: *Strength Pareto Evolutionary Algorithm* (SPEA) (ZITZLER; THIELE, 1999), antecessor do SPEAII (ZITZLER; LAUMANNS; THIELE, 2001), *Non dominated Sorting in Genetic Algorithms* (NSGA) (SRINIVAS; DEB, 1994), NSGAII (DEB et al., 2002), entre outros. Nos próximos tópicos é descrito o funcionamento desses algoritmos.

Algoritmo Evolutivo de Força de Pareto (do inglês *Strength Pareto Evolutionary Algorithm* SPEA, SPEAII) - Os algoritmos (SPEA) (ZITZLER; THIELE, 1999) e SPEAII (ZITZLER; LAUMANNS; THIELE, 2001) foram os primeiros algoritmos a inserir o conceito de elitismo nos AGMs. O método consiste em separar as populações em dois blocos: bloco interno e bloco externo. A população inicial é colocada no bloco interno, o bloco externo permanece vazio na primeira iteração.

Na etapa seguinte, os membros da população passam pela avaliação do *fitness*. E todos os indivíduos não dominados são copiados para o conjunto externo (elite). Se o tamanho do conjunto externo exceder um limite predefinido, os membros do bloco serão excluídos por uma técnica de *clustering*, que preservará as características da frente não dominada. Posteriormente, aplicadas as funções de cruzamento e mutação, a nova população é avaliada nos critérios de dominância e será colocada no conjunto interno.

Os algoritmos SPEA e SPEAII se diferenciam apenas na última etapa. O SPEA une as populações (conjunto externo e interno) com o emprego do algoritmo de seleção por torneio (Seção 2.3.1). No SPEAII, a densidade dos blocos (interno e externo) é mensurada por meio dos vizinhos mais próximos. Os indivíduos presentes na região com menor densidade serão selecionados para a próxima geração.

Ordenação não dominada em Algoritmos Genéticos (do inglês *Non do-*

***minated Sorting in Genetic Algorithms* NSGA, NSGAII)** - Os métodos de seleção de indivíduos (SRINIVAS; DEB, 1994) (NSGA) e o (NSGAII) (DEB et al., 2002), utilizam a técnica de *Ranking* de Pareto (RP) a partir dos conceitos de dominados e não dominados. O método RP foi descrito por Goldberg (1989) e segue o Algoritmo 1:

Algoritmo 1: Frente de Pareto (GOLDBERG, 1989)

Entrada : P_t população de indivíduos
Saída : $\pounds$ frente de Pareto

```
1  i = 1
2  P* = P_t
3  enquanto i ≠ g faça:
4      Aid_i j = N_d não dominados de P*_i
5      se Aid_i j == ∅ então:
6          para cada O_si ∈ P*_ti faça:
7              R_i = O_si
8              N_d_ij = Rk_i
9              i=i + 1
10             j=j + 1
11         fim
12     senão
13         i=i + 1
14         j=j + 1
15     fim
16     £ = N_d
17 fim
18 retorne £
```

Nesse algoritmo, i é um contador de frentes, P_t é a representação da população e P_{tcop} é uma cópia de P_t. O símbolo $\pounds$ denota a frente de Pareto, ou seja, os melhores indivíduos posicionados no *ranking*. As soluções não dominadas para os objetivos são representadas por Os, o número frentes é denotado por $\pounds^*$ e o *ranking* por R. O algoritmo começa atribuindo 1 ao contador, realiza a cópia da população e entra em *loop* (linha 3). Durante o *loop*, A_i recebe os indivíduos não dominados, N_d. Se por acaso não se encontre membros não dominados é feito um *ranking* dos melhores resultados dos objetivos. Diante dessa possibilidade, $\pounds$ recebe todos os indivíduos não dominados de todas as frentes (linha 16).

O algoritmo NSGA utiliza diferentes frentes (*fronts*) de Pareto, conceito de elitismo e o critério de dominância para distinguir os melhores indivíduos e classificá-los em diferentes categorias. Indivíduos da primeira frente, $\pounds$, são melhores que os indivíduos da segunda frente ($\pounds + 1$).

O NSGAII possui dois critérios que o diferencia do NSGA, são eles: *Fast Non-Dominated Sorting* e a *Crowding Distance*. O algoritmo *Fast Non-Dominated Sorting*, compara os indivíduos em pares, com o conceito de dominância para criar as frentes, já a *Crowding Distance* é utilizada como um método de diversidade.

A *Crowding Distance* é um método presente no NSGAII, cujo objetivo é garantir a melhor dispersão dos indivíduos ao longo da CP. O algoritmo calcula um ponto central (ce) entre as soluções dos objetivos para cada indivíduo. A partir do ce, encontram-se as extremidades $(ce-1)$ e $(ce+1)$. Dentro desses limites, são empregados os indivíduos na Curva de Pareto. A Figura 5 representa a *Crowding Distance* para dois objetivos (O_1 e O_2), a área criada pelo algoritmo é chamada de *Cuboid*, os círculos iguais representam os indivíduos do mesmo *front*. Ainda na Figura 5, é possível observar onde foi colocado a nova solução (ce), ela se encontra no centro de *Cuboid*, nesse caso, em um espaço de apenas duas dimensões.

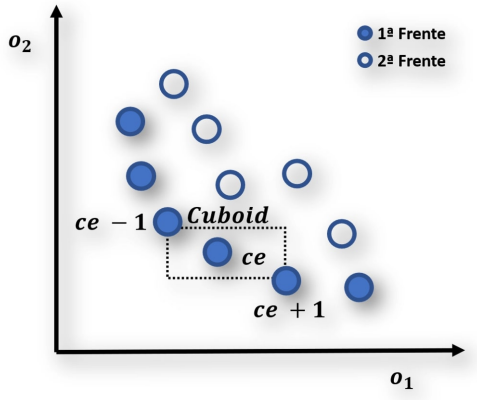


Figura 5 – Representação do algoritmo *Crowding-distance*. A solução ce é colocada entre duas outras soluções muito próximo ao centro da curva de Pareto.

Fonte: Adaptado (DEB et al., 2002).

A distância de cada um dos indivíduos em relação ao ponto médio, para diferentes objetivos, é dada pela Equação 2.22. A distância do $j-$ésimo indivíduo I da população P_t, em relação ao objetivo O_j, é representada por $dist_{P_t^{O_j}}$. Os valores de cada objetivo correspondem aos indivíduos das posições $j+1$ e $j-1$ do conjunto P_t. A representação do $O-$ésimo objetivo na frente de Pareto é $£O^{(P_t^{O_{j+1}})}$ e $£O^{(P_t^{O_{j-1}})}$ logo, a distância será a razão da diferença das frentes ($£O$), pela diferença do máximo e mínimo ($£_O^{max} - £_O^{min}$), melhores e piores indivíduos, pertencentes ao $O-$ésimo objetivo.

$$dist_{P_t^{O_j}} = dist_{P_t^{O_j}} + \frac{£O^{(P_t^{O_{j+1}})} - £O^{(P_t^{O_{j-1}})}}{£_O^{max} - £_O^{min}} \tag{2.22}$$

Destarte, o NSGAII mantém a diversidade e os critérios de dominância entre indivíduos de uma população.

Outros métodos - O resumo dos métodos apresentados e outros métodos de seleção de indivíduos encontrado na literatura, estão presentes na Tabela 4.

Tabela 4 – Algoritmos de seleção de AGM

Algoritmo	Elitismo	Diversidade	Autor
VEGA	Não	Não	(SCHAFFER, 1985)
MOGA	Não	Por grupo baseado no fitness	(FONSECA; FLEMING, 1993)
WBGA	Não	Por grupo com pesos	(HAJELA; LIN, 1992)
NPGA	Não	Por grupo e seleção por torneio	(HORN; NAFPLIOTIS; GOLDBERG, 1994)
RWGA	Sim	Randômico por pesos	(MURATA; ISHIBUCHI, 1995)
PESA	Sim	Baseado em densidade	(CORNE; KNOWLES; OATES, 2000)
PAES	Sim	Baseado em densidade entre filhos e pais	(LU; YEN, 2003)
NSGA	Não	Por grupo baseado no fitness	(SRINIVAS; DEB, 1994)
NSGAII	Sim	*Crowding distance*	(DEB et al., 2002)
NSGA3	Sim	Métricas de hyper-voume	(JAIN; DEB, 2014)
SPEA	Sim	Cluster e truncamento da população	(ZITZLER; THIELE, 1999)
SPEAII	Sim	Densidade baseada nos vizinhos mais próximos	(ZITZLER; LAUMANNS; THIELE, 2001)

Na próxima seção, serão abordados os métodos de seleção dinâmica de classificadores. Esses métodos fazem parte da validação da geração de *pool* proposta neste trabalho. Um bom desempenho na seleção dinâmica de classificadores demonstra, na maioria das vezes, que o conjunto de modelos foi bem construído.

2.4 Algoritmos de seleção dinâmica de classificadores

De acordo com Britto, Sabourin e Oliveira (2014), um classificador monolítico dificilmente consegue gerar um modelo capaz de representar todo o problema de classificação. Diante disto, os Sistemas de Múltiplos Classificadores (SMCs) criam um conjunto de classificadores com diferentes competências e especialidades.

A seleção de classificadores (SC) é a segunda etapa dos SMCs, sendo a primeira formada pela geração de classificadores, a qual será abordada no próximo capítulo, e a terceira denominada de integração (BRITTO; SABOURIN; OLIVEIRA, 2014). A seleção de classificadores, apesar de ser uma etapa opcional, entrega muitas vezes, melhores resultados quando comparada com a combinação dos classificadores.

As SCs podem ser divididas em duas categorias: seleção estática e seleção dinâmica. A seleção estática de classificadores é realizada durante o treinamento. O conjunto de classificadores será composto com os de melhor *score* (CRUZ; SABOURIN; CAVALCANTI, 2017). Referente aos métodos de seleção estática, o *score* pode variar de acordo com a estratégia, sendo uma delas a gulosa (complementariedade, concorrência e a minimização da distância da margem, as quais são baseadas em acurácia) (PARTRIDGE; KRZANOWSKI, 1997).

A Seleção Dinâmica (SD), pode ser dividida em: Seleção Dinâmica de Classificadores (SDC) e Seleção Dinâmica de *Ensembles* de Classificadores (SDE). Ambos os métodos

ocorrem durante a fase de teste. Na SDC, para cada exemplo de teste, é escolhido o classificador que possua um valor mínimo de competência (o cálculo da competência de um classificador depende de cada método de seleção) previamente estipulado (CRUZ; SABOURIN; CAVALCANTI, 2017). Analogamente, um método de SDE escolhe um subgrupo de classificadores dentro do *pool* de classificadores, que posteriormente serão combinados. Nos tópicos subsequentes, serão apresentadas algumas técnicas de SDC e SDE.

Acurácia Local Geral (OLA) - Esse método de SDC foi proposto pelos autores Woods, Kegelmeyer e Bowyer (1997). O *Overall Local Accuracy* (OLA) determina a competência de cada classificador do *pool* na região definida pela vizinhança do exemplo de teste em uma base de validação. O método inicia com uma instância de teste x^*. Encontra-se então um subconjunto de instâncias a partir de um conjunto de validação, com os k vizinhos da instância de teste. Os classificadores do *pool* devem rotular todos os exemplos desse conjunto, o classificador que rotular corretamente o maior número de instâncias, será o modelo escolhido. A Equação 2.23 representa a estimativa da competência.

$$\delta_{i,j} = \frac{1}{k} \sum_{k=1}^{k} \Phi(y_j^* \mid {x^*}_k, C_i) \tag{2.23}$$

onde k corresponde ao número de vizinhos da instância em análise, x^* representa a instância de teste, enquanto y^* é o rótulo a ser predito pelo classificador C_i e Φ é o resultado da predição do classificador. O símbolo δ representa a região de competência.

Acurácia local da Classe (LCA) - O *Local Class Accuracy* (LCA), da mesma forma que o OLA, utiliza o conceito dos k vizinhos mais próximos para determinar a região de competência de cada classificador. Nesse método, o modelo atribui a um exemplo de teste, x^*, um rótulo, y^*, obtendo um conjunto de instâncias dos k vizinhos da amostra de teste. Na etapa seguinte, cada classificador irá rotular todo conjunto dos k vizinhos do exemplo de teste. Por concluinte, calcula-se a razão entre os rótulos classificados como y^* corretamente, pelo total de rótulos classificados como y^*, mesmo que incorretamente. O classificador que apresentar a melhor relação entre acertos e erros será escolhido (WOODS; KEGELMEYER; BOWYER, 1997).

A Equação 2.24 representa o método LCA, onde Φ é o número de vizinhos rotulados como y_j pelo classificador C_i. O fator y_i representa as classes corretas do conjunto de instâncias formado pelos k vizinhos. Finalmente, δ irá estimar a competência de cada classificador.

$$\delta_{i,j} = \frac{\sum_{{x^*}_k \in y_i} \Phi(y_j \mid {x^*}_k, C_i)}{\sum_{k=1}^{k} \Phi(y_j \mid {x^*}_k, C_i)} \tag{2.24}$$

A Figura 6 representa o LCA para $k = 7$ e um conjunto de 3 classificadores. Quando um classificador classifica incorretamente uma instância, com o mesmo rótulo predito, isso é considerado um erro. No caso de um modelo classificar incorreta ou corretamente uma

instância, cujo rótulo é diferente do predito, o resultado será ignorado. Diante disto, são considerados acertos quando o rótulo predito é o mesmo do exemplo de teste. Nessa figura o classificador C_1 é o mais competente.

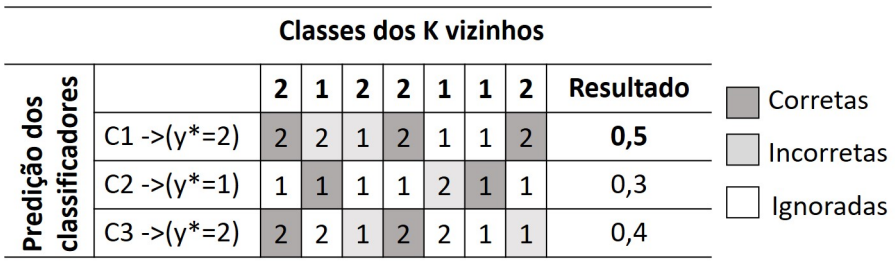


Figura 6 – Representação do algoritmo LCA para 3 classificadores e $k = 7$.

Fonte: O autor.

Seleção de classificador baseado em Rank (modificado) - O Rank (modificado) (WOODS; KEGELMEYER; BOWYER, 1997) é um método que avalia o nível de competência de cada classificador individualmente com o mesmo conceito dos vizinhos mais próximos. A competência de cada classificador de base é calculada por meio do número consecutivo de amostras classificadas corretamente, partindo do vizinho mais próximo do exemplo de teste (x^*) até k vizinhos. Nesse sentido o modelo com maior número de exemplos classificados corretamente na sequência em que os vizinhos mais próximos foram organizados, será considerado o mais competente.

Outros métodos - Outros métodos de seleção dinâmica de classificadores são citados abaixo:

A Priori, esse método utiliza a probabilidade de classificação correta e a distância Euclidiana dos vizinhos do exemplo de teste (GIACINTO; ROLI, 1999).

A Posteriori, utiliza a probabilidade de classificação correta, todavia somente da mesma classe do exemplo de teste e a distância Euclidiana dos vizinhos do exemplo de teste (GIACINTO; ROLI, 1999).

DS-MCB, avalia o nível de competência de cada classificador levando em consideração a precisão local na região de competência. A região de competência é definida pelo algoritmo KNN (GIACINTO; ROLI, 2001b).

DS-MLA, semelhante à técnica LCA. Entretanto, a saída de cada classificador é ponderada pela distância entre o exemplo de teste e a região de competência para as estimativas dos classificadores (SMITS, 2002).

K Oráculos mais próximos (KNORA) - Similar aos métodos LCA, OLA e Rank, que utilizam os k vizinhos para determinar a região de competência, o método KNORA (*K-Nearest-Oracle*) se diferencia dos demais por utilizar o conceito de oráculo. A estratégia consiste em encontrar os modelos mais aptos para classificar o exemplo de teste através de um conjunto de validação (KUNCHEVA; RODRIGUEZ et al., 2007).

O método proposto pelos autores Ko, Sabourin e Jr. (2008), seleciona um conjunto de classificadores de acordo com o resultado da classificação dos k vizinhos da instância de teste no conjunto de validação. A combinação desses classificadores, por padrão, fica a cargo do voto majoritário simples. Abaixo serão abordadas as variações do método KNORA: KNORA *Eliminate* (KNORA-E), e KNORA *Union* (KNORA-U).

O KNORA-E localiza os k exemplos mais próximos da amostra de teste (vizinhos) no conjunto de validação, gerando um subproblema. O algoritmo busca os modelos capazes de classificar corretamente todo esse conjunto. Caso os classificadores não consigam classificar todas as amostras, o valor de k é decrementado em uma unidade, o processo repete-se até que sejam classificadas todas as amostras por pelo menos um classificador. O Algoritmo 2 descreve o funcionamento do KNORA-E.

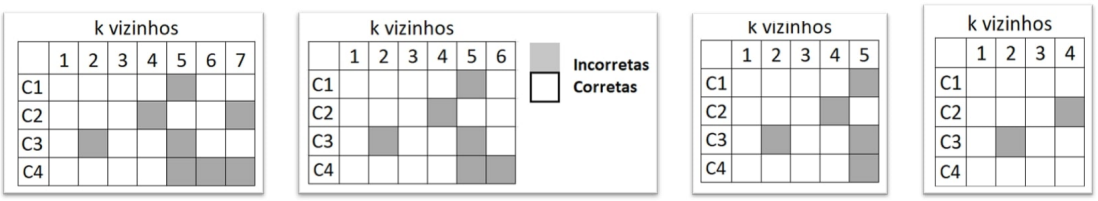
Algoritmo 2: KNORA-E (KO; SABOURIN; JR., 2008)

Entrada : C_c é o conjunto de classificadores
vf meta espaço de treinamento
x^* é a amostra de teste
k é o número de vizinhos

Saída : EoC^*, conjunto de classificadores para amostra x^*

```
enquanto k>0 faça:
    δ = os exemplos dos k vizinhos mais próximos da amostra x* em vf
    para cada C_i ∈ C_c faça:
        se C_i classifica corretamente todas as amostras de δ então:
            EoC* = EoC* ∪ C_i
        fim
    fim
    se EoC* == ∅ então:
        k=k − 1
    fim
    senão
        break
    fim
fim
se EoC* == ∅ então:
    C_i = o classificador que reconheça corretamente mais amostras em delta.
    EoC* = os classificadores capazes de reconhecer a mesma quantidade de
     amostras C_i
fim
retorne EoC*
```

A Figura 7 demonstra o KNORA-E, onde estão representados quatro classificadores



(a) K=7 (b) K=6 (c) K=5 (d) K=4

Figura 7 – Representação do algoritmo KNORA-E para 4 classificadores e $k = 7$ em (a), 6 em (b), 5 em (c) e 4 em (d).

Fonte: O autor.

e um conjunto de $k = 7$ vizinhos da instância de teste. No primeiro passo (Figura 7 (a)), os classificadores fazem a previsão dos 7 exemplos, como nenhum especialista consegue acertar todos os exemplos formados pelos k vizinhos, então k é decrementado em uma unidade. O processo continua nas Figuras 7 (b) e (c), até que os classificadores C_1 e C_4 classificam corretamente todos os exemplos obtidos por k, onde nesse caso, $k = 4$ (Figura 7 (d)). O KNORA-U funciona de maneira semelhante ao KNORA-E. O método busca classificadores que classifiquem corretamente os k vizinhos da instância de teste x^*, para cada acerto, o classificador receberá um voto. Aqueles que acertarem n instâncias terão n votos para a fase de combinação. Dessa maneira, os classificadores que possuírem mais votos terão maiores oportunidades de participar do conjunto final de classificadores. O Algoritmo 3 representa o método KNORA-U.

Algoritmo 3: KNORA-U (KO; SABOURIN; JR., 2008)

Entrada : C_c é o Conjunto de classificadores
vf meta espaço de características
x^* é a amostra de teste
Saída : EoC^* como o conjunto de classificadores para amostra x^*

```
δ = os exemplos dos k vizinhos mais próximos da amostra x* em vf
para cada δ_i ∈ δ faça:
    para cada C_j ∈ C_c faça:
        se C_j classifica corretamente δ_i então:
            EoC* = EoC* ⊎ C_j
        fim
    fim
fim
retorne EoC*
```

Os autores Ko, Sabourin e Jr. (2008) concluíram que os métodos KNORA-U e KNORA-E obtiveram bons resultados quando comparados aos métodos de SDC, apesar da incerteza do comportamento do oráculo na definição da região de competência.

Meta-aprendizado para seleção dinâmica de conjuntos de classificadores

Figura 8 – Representação do algoritmo KNORA-U para 3 classificadores e $k = 7$.

Fonte: O autor.

(META-DES) - O método META-DES (CRUZ et al., 2015) é um *framework*, onde se executa um estágio de meta-treinamento, no qual são extraídas as meta-características de cada instância do conjunto de treino, as quais serão utilizadas para treinar um meta-classificador que será aplicado para determinar o nível de competência dos modelos do *pool*.

Os especialistas do *pool*, com nível de competência superior a um limiar predefinido serão selecionados para compor o conjunto de seleção dinâmica. Se nenhum classificador for selecionado, todo o conjunto será usado para a classificação (CRUZ et al., 2015).

O framework META-DES é segmentado em três fases: geração, meta-treinamento e generalização. Na fase de geração, um conjunto de classificadores é criado com os dados de treinamento. Nesta fase é utilizado o método Bagging (BREIMAN, 1996). No estágio do meta-treinamento, cinco meta-características são extraídas dos dados de treinamentos, e são usadas para treinar um meta-modelo. Durante a fase de generalização, as meta-características são extraídas das instâncias de teste, essas são utilizadas como entrada do meta-classificador. O meta-modelo estima se um classificador do *pool* é competente o suficiente para classificar a instância de teste.

A segunda etapa do método é responsável por criar o meta-modelo. Nesse processo são utilizadas as seguintes meta-características: classificação dos vizinhos, probabilidade posterior, acurácia local geral, classificação dos perfis de saída e a confiança do classificador. A Tabela 5 define os critérios para criação de cada meta-característica e seus paradigmas.

Tabela 5 – Meta-características utilizadas no META-DES.

Meta-característica	**Critério**	**Paradigma**
Classificação dos vizinhos	Acurácia local da região de competência	Acurácia do classificador em uma região
Probabilidade posterior	Extensão do consenso na região de competência	Consenso dos classificadores
Acurácia local geral	Acurácia geral da região de competência	Acurácia sobre a região
Classificação dos perfis de saída.	Acurácia no espaço de decisão	Modelos de decisão
Confiança do classificador	Grau de confiança das entradas de teste	Confiança dos classificadores

Fonte: adaptado (CRUZ et al., 2015).

A fase do meta-treinamento está representada no Algoritmo 4, onde τ representa o conjunto de treinamento original, λ representa a meta-característica de treinamento do

meta-classificador, para toda amostra $x_{j,treino} \in \tau_\lambda$.

Algoritmo 4: Meta-treinamento (CRUZ et al., 2015)

Entrada : τ_λ conjunto de treinamento
Saída : λ como meta-classificador

τ_λ^*=$\emptyset$
para cada $x_{j,treino} \in \tau_\lambda$ **faça:**
  compute o consenso para o *pool* Cons($x_{j,treino}\tau_\lambda$, C_c)
  se *Cons*($x_{j,treino}\tau_\lambda$, C_c)<$cons_{C_c}$ **então:**
    δ = a região de competência para $x_{j,treino}$ usando o τ_λ
    $\hat{x}_{j,treino}$ = compute a saída de $x_{j,treino}$
    y^{**} = saídas similares de φ_j de $\hat{y}_{j,treino}$ usando τ_λ **para cada** $C_i \in C_c$ **faça:**
      Ef_{ij} = extração das meta-características (δ_j, C_i, $x_{j,treino}$)
      se C_i *classifica corretamente* $x_{j,treino}$ **então:**
        $\alpha_{ij} = 1$ // C_i é competente para $x_{j,treino}$
      senão
        α_{ij}=0 // C_i não é competente para $x_{j,treino}$
      fim
      $\tau_\lambda^* = \tau_\lambda^* \cup V_{ij}$
    fim
  fim
fim
divida τ_λ^* em 25% para validação e 75% para treino
retorne *meta-classificador* λ

Dentre os métodos abordados, o META-DES foi na maioria dos casos de teste o melhor método de seleção de conjunto de classificadores. De acordo com os autores Cruz et al. (2015), a estratégia apresenta um bom desempenho para os conjuntos com poucos exemplos, visto que cada instância de treino gera uma quantidade grande de dados para criação do meta-classificador.

2.5 Considerações finais

Nesse capítulo foram apresentadas as principais métricas de complexidade, como elas funcionam e como são obtidos seus valores. Ainda foi discorrido sobre os algoritmos evolutivos, combinação de classificadores, os métodos de seleção dinâmica de classificadores (SDC) e a seleção dinâmica de *ensemble* de classificadores (SDE). Foram abordadas as medidas de complexidade, e foi verificado que elas podem apresentar inúmeras aplicações em AM. Entre as aplicações, pode-se destacar a extração do nível de dificuldade do

problema para identificar o seu domínio e suas particularidades.

Trabalhos como (LORENA et al., 2012) e (KAMATH; YEATMAN; ESCHRICH, 2009) aplicam as métricas de complexidade para conjunto de dados do genoma humano. Na pesquisa dos autores Lorena et al. (2012), as propriedades de densidade e balanceamento são extraídas e utilizadas para otimizar o problema e melhorar o desempenho do classificador SVM. No trabalho dos autores Kamath, Yeatman e Eschrich (2009), foram utilizadas as medidas de complexidade para explorar o limite máximo de acurácia preditiva que pode ser alcançado em um conjunto de dados.

Outra aplicação do uso de complexidade está no pré-processamento de um problema de classificação, como nos trabalhos (MOLLINEDA; SÁNCHEZ; SOTOCA, 2005), (SÁNCHEZ; MOLLINEDA; SOTOCA, 2007), (GARCIA; CARVALHO; LORENA, 2015) e (FRÉNAY; VERLEYSEN, 2014). Já na pesquisa dos autores Mollineda, Sánchez e Sotoca (2005) e Sánchez, Mollineda e Sotoca (2007), foram usadas as métricas de complexidade para pré-processar um *dataset*, com o objetivo de melhorar o desempenho do classificador KNN. Os autores Garcia, Carvalho e Lorena (2015) e Frénay e Verleysen (2014) aplicaram as medidas de complexidade para identificar atributos com ruído, ou seja, atributos que podem ser considerados *outliers* que dificultam a aprendizagem do classificador.

Entre a diversidade de aplicações da dificuldade do problema nos algoritmos de AM, pode-se destacar o uso em métodos de geração de classificadores e seleção de classificadores. Os autores Brun et al. (2018) apresentam em sua pesquisa, uma maneira de gerar um *pool* de classificadores com algoritmos de otimização e métricas de complexidade. No mesmo trabalho, a complexidade do problema de classificação é aplicada na extração da região de competência, para a seleção de um classificador no *pool*. Na pesquisa dos autores Flores, Gámez e Martínez (2014) e Luengo e Herrera (2015) as métricas de complexidade também são utilizadas para encontrar a região de competência dos classificadores. Para isso, ambos autores definem, em alguns passos e de forma dinâmica, o que é um comportamento "bom" ou "ruim" de um classificador. A cada iteração os métodos avaliam os subproblemas utilizando a acurácia e as medidas de complexidades.

Nos algoritmos evolutivos, foram descritos os algoritmos genéticos e as principais estruturas que os compõem. Destacam-se nesses métodos, a seleção de indivíduos, cujo funcionamento pode ser dividido em duas categorias: seleção baseada em apenas um objetivo; seleção com mais de um objetivo. A aplicação dos algoritmos genéticos na área de AM está presente na classificação de imagens, na evolução de classificadores, extração de características, entre outras. O trabalho dos autores Sun et al. (2020), propõem um método automático para melhorar a arquitetura de Redes Neurais Convulsionais (CNN), com algoritmos genéticos, para então utilizar na classificação de imagens. Li et al. (2011) utilizaram AG combinado com SVM para extração de características de imagens hiper espectrais. Outra aplicação do AG é o uso na geração de classificadores, como por exemplo, os autores Cavalcanti et al. (2016) que utilizam as medidas de diversidade combinadas por

um AG para gerar um conjunto de classificadores.

Finalmente, foram apresentados alguns métodos de seleção dinâmica de classificadores. A escolha de classificadores foi dividida em métodos de seleção de um único classificador e seleção de conjuntos de classificadores. O primeiro elege o melhor classificador do *pool*, de acordo com uma heurística, para cada instância de teste. No segundo, um grupo de especialistas são escolhidos e combinados para tentar obter o melhor resultado de classificação.

As técnicas descritas neste capítulo farão parte da composição e avaliação do método proposto para essa pesquisa. No capítulo seguinte serão discutidos os principais métodos para geração e avaliação de conjuntos classificadores.

3 Estado da arte

Os SMCs possuem algumas vantagens em relação aos classificadores monolíticos (classificador único). O trabalho de Kuncheva (2002) mostra que a combinação de classificadores é melhor, em termos de acurácia, quando comparada a performance de um único classificador.

Já Dietterich (2000) demonstra em seu trabalho três problemas encontrados em sistemas de classificação de um único classificador. O primeiro, chamado de estatístico, ocorre quando os dados de treinamento são escassos, o modelo tende a classificar as instâncias por aproximação, consequentemente cometendo mais erros. A segunda dificuldade encontrada é a computacional, muitos algoritmos de aprendizagem trabalham de forma otimizada em um determinado local do problema de classificação logo, podem classificar incorretamente as instâncias fora dessa região. Por último, Dietterich (2000) cita que dificilmente um único classificador consegue representar todo o problema de classificação. Todas essas situações são minimizadas com o uso da combinação de classificadores.

Na pesquisa dos autores Fernández-Delgado et al. (2014) foram realizados 21659 experimentos com 179 classificadores, incluindo métodos de geração de *pool* de classificadores e 121 bases de dados. Os pesquisadores observaram que os conjuntos de classificadores possuíam um melhor desempenho quando comparado a um único classificador.

Outro estudo empírico do uso de SMCs foi apresentado pelos autores Opitz e Maclin (1999), nele foi comprovado que o método *Bagging* (apresentado nas próximas seções) é melhor quando comparado a somente um modelo.

No que diz respeito à primeira etapa de um SMC (geração de *pool*), pode-se destacar duas estratégias principais: heterogênea e homogênea. Na estratégia heterogênea, o conjunto de especialistas é criado utilizando diferentes classificadores de base sob um mesmo conjunto de treinamento. Já na estratégia homogênea, um classificador de base é treinado com diferentes amostras de dados, ou através da variação dos parâmetros do algoritmo de aprendizado.

Grande parte das propostas existentes na literatura, seguem a abordagem homogênea. Os métodos, *Bagging* (BREIMAN, 1996), *AdaBoost* (FREUND; SCHAPIRE, 1996) e *Random Subspaces* (HO, 1998) são exemplos de geradores de *pools* homogêneos.

3.1 *Bagging*

O *Bootstrap Aggregating* (*Bagging*) (BREIMAN, 1996) é um algoritmo executado em duas etapas. Na primeira etapa é feita a construção dos subproblemas que formarão os modelos (quando o classificador de base é treinado) para a classificação dos dados. Esses subconjuntos são gerados por amostragens aleatórias e com reposição a partir de

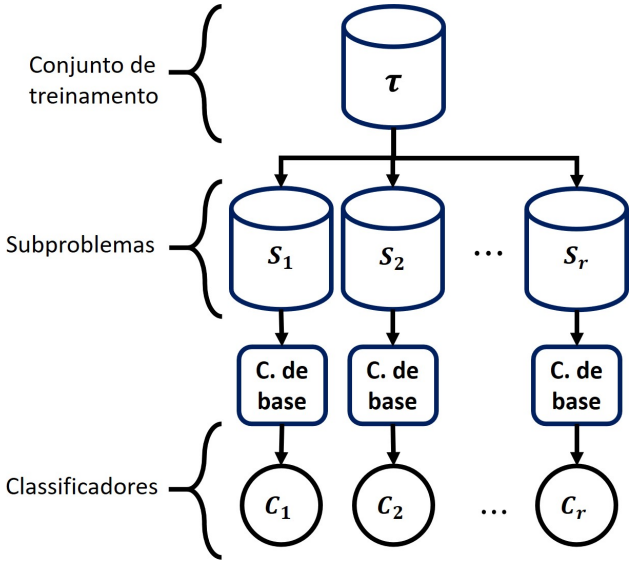


Figura 9 – Primeira etapa do *Bagging*. O símbolo τ representa a amostra de treinamento, S os subproblemas gerados aleatoriamente e com repetição, por fim C representa os classificadores treinados com as amostras S.

Fonte: O autor.

um conjunto de dados de treinamento. Sequencialmente, na segunda etapa do método, os classificadores são combinados com diferentes técnicas, tal como o voto majoritário (BREIMAN, 1996).

A Figura 9 representa a primeira etapa do *Bagging*. A notação utilizada na Figura 9 e no algoritmo 5 segue a seguinte situação: dada uma amostra de treino τ, que contêm n pares de exemplos (x_i, y_i) do problema, sendo que, x_i e y_i recebem a mesma definição apresentada no Capítulo 2. Com o conjunto τ, são criados S subproblemas contendo u pares de exemplos. Estes subproblemas são gerados por distribuições aleatórias e com reposição das instâncias de τ. De forma consecutiva, C representa os classificadores treinados com as amostras S, o número de classificadores é o mesmo de subamostras de treino representado por r.

O método pode ser observado no Algoritmo 5. Entre as linhas 1 e 4 o algoritmo entra na estrutura de repetição que irá construir r subproblemas e treinar o classificador de base. Na linha 5 é realizada a combinação dos classificadores, onde ω representa o voto majoritário para cada predição, Φ, do exemplo de teste x^*. Na última linha o algoritmo retorna a acurácia de todo conjunto de treinamento.

Algoritmo 5: Bagging (BREIMAN, 1996)

Entrada : τ é o conjunto de treinamento
u é o tamanho dos subproblemas
r é o número de subproblemas
y é o rótulo de cada instância de teste
x^* é a instância de teste.

Saída : ω combinação de classificadores

1 **para** *i=1 até r* **faça:**
2 S_i = construa a subamostra de tamanho u a partir do conjunto τ
3 C_i = construa um modelo a partir de S_i
4 **fim**
5 $\omega_{(x^*)} = maxcount\left[argmax\ \Phi_{C_i}(y|x^*)\right]$ /* retorna o resultado da combinação dos classificadores para cada instância x^*. */
6 **retorne** ω

3.2 *AdaBoost*

O *Adaptive Boosting* (*AdaBoost*) (FREUND; SCHAPIRE, 1996) é um algoritmo que utiliza a distribuição aleatória dos exemplos do conjunto de treinamento para gerar as subamostras. Esse método é realizado em dois estágios. No primeiro, são gerados os subproblemas com exemplos de treinamento distribuídos aleatoriamente e com repetição. Cada subproblema é gerado a partir de um grupo de treinamento τ, formado por n exemplos, tal como o *Bagging*. Esse processo se repete por T iterações. Contudo, diferente do *Bagging* a cada iteração as instâncias classificadas incorretamente pelos classificadores já gerados terão maior probabilidade de participar do treinamento do próximo classificador.

Na primeira iteração do algoritmo, cada instância do conjunto possui a mesma chance de ser sorteada. Entretanto, a partir da segunda iteração o classificador gerado é usado para avaliar a base de treinamento e cada instância terá um peso associado. Este peso é incrementado quando a instância é incorretamente classificada. Desta forma, na próxima iteração as instâncias com peso maior terão maior chance de serem selecionadas pelo processo de amostragem. Dessa maneira, o método prioriza as instâncias mais difíceis de serem classificadas para a construção do *pool*. Após o término do primeiro estágio é realizada a combinação dos classificadores criados.

A Figura 10 representa a primeira etapa do *AdaBoost*, onde τ representa as amostras de treinamento, o número de subproblemas e de classificadores é representado por r. Por último C representa o modelo treinado pelos subproblemas de τ.

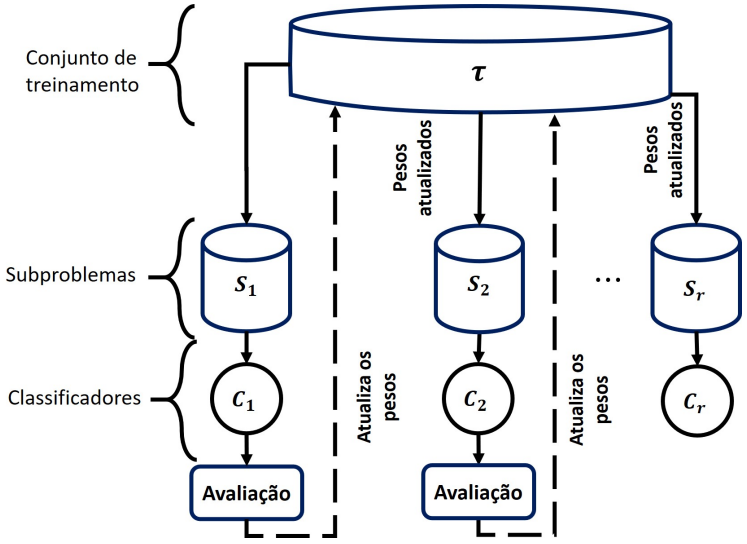


Figura 10 – Primeira etapa do *AdaBoost*. Depois de geradas as subamostras o classificador fornece peso para as instâncias classificadas incorretamente, para assim, aumentar as chances de participarem da geração do próximo subproblema.

Fonte: O autor.

O Algoritmo 6 demonstra como são executadas todas as etapas do *AdaBoost*. Na primeira linha, um contador de interações t é iniciado com 1. Em seguida, cria-se um subconjunto de dados S a partir das instâncias do conjunto de treinamento estratificado e com reposição. Na linha 3 são atribuídos pesos para todas às instâncias (Equação 3.1) formando um subconjunto w_t. Nesta equação, u representa o número de instâncias do subconjunto S. Vale salientar que, na primeira iteração todos os exemplos recebem o mesmo peso.

$$w_t = S_t(x_i)\frac{1}{u} \tag{3.1}$$

O algoritmo entra em *loop* (linha 4 - 11) iniciando na iteração $t = 1$ até T. Na linha 5 o classificador de base é treinado com a subamostra w_t. Na próxima linha, realiza-se o cálculo do erro cometido por esse classificador. O valor do erro (ϵ) é determinado pela Equação 3.2, sendo que Φ é a predição do classificador C_t para cada instância x_i.

$$\epsilon_t = \sum_{i=0}^{u} \Phi(C_t(x_i)) \neq y_i \tag{3.2}$$

No próximo passo do algoritmo, encontra-se a importância (η) do modelo de classificação por meio da Equação 3.3. A distribuição dos pesos atribuída aos exemplos de treinamento é atualizada de acordo com a Equação 3.4. O termo Z é o fator normalizador, que na primeira iteração assume o valor 1. Nas demais iterações, Z será atualizado seguindo a Equação 3.5.

$$\eta_t = \frac{1}{2} ln \left(\frac{1 - \epsilon_j}{\epsilon_j} \right) \tag{3.3}$$

$$w_{(t+1)}(i) = \frac{w_t(i)}{Z_t} * \begin{Bmatrix} e^{-\eta_t} & se & \Phi(C_t(x_i)) = y_i \\ e^{\eta_t} & se & \Phi(C_t(x_i)) \neq y_i \end{Bmatrix} \tag{3.4}$$

$$Z_t = \prod_{t=1}^{T} \left[2\sqrt{\epsilon_t(1-\epsilon_t)} \right] \tag{3.5}$$

Algoritmo 6: AdaBoost (FREUND; SCHAPIRE, 1996)

Entrada : τ é o conjunto de treinamento
T é o número de iterações
u é número de instâncias de cada subproblema
y é o rótulo de cada instância de teste
x^* é a instância de teste.

Saída : ω é combinação de classificadores

1 $t = 1$
2 S_t = construa a subamostra de tamanho u a partir do conjunto τ.
3 w_t= Atribua pesos as instâncias de S_t /* Equação 3.1 */
4 **Repita**
5 C_t = treine o classificador com w_t
6 ϵ_t = calcule o erro do classificador C_t /* Equação 3.2 */
7 η_t = calcule a importância do classificador C_t /* Equação 3.3 */
8 w_{t+1} = atualize os pesos de cada instância de w_t /* Equação 3.4 */
9 Z_t = atualize do fator normalizante Z /* Equação 3.5 */
10 $t = t + 1$
11 **até** $t = T$
12 $\omega_{(x^*)} = maxcount\ [argmax\Phi_{C_i}(y|x^*)]$ /* retorne o resultado da combinação do conjunto para cada instância de teste. */
13 **retorne** ω

3.3 *Random Subspaces*

O método *Random Subspaces* (RS) (HO, 1998) explora a diversidade dos dados de forma vertical, ou seja, criando diferentes subconjuntos de características (atributos). Sendo que, para cada par de exemplos (x_i, y_i), onde n é o número total, existe um conjunto de m atributos, tal que $x_i = \{x_i f_1, x_i f_2, ..., x_i f_m\}$. Portanto, o conjunto de treinamento descreve um vetor de $x_n f_m$-dimensões, representado por Vf. Para criação das subamostras do método, o valor de m' (subconjunto de m) deve ser menor que m, isto é, um vetor Vf'.

A Figura 11 representa a geração dos subproblemas do método RS em que, C denota os classificadores e r o número de classificadores. Os atributos do problema são representados por f, o número total por m e m' corresponde ao total de atributos de cada

subamostra S. Importante destacar que não se deve sortear os mesmos atributos para formar um classificador. Entretanto, classificadores distintos podem possuir atributos em comum.

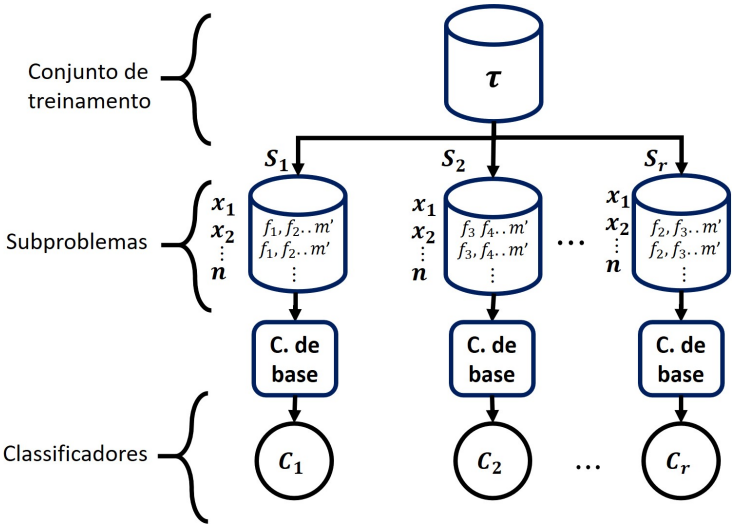


Figura 11 – Primeira etapa do *Random Subspaces*.

Fonte: O autor.

O Algoritmo 7 descreve os passos para gerar um conjunto de classificadores utilizando o RS. Na primeira linha do algoritmo, o *loop* é executado de i até o número de subamostras r. Geram-se os subproblemas S_i com os exemplos compostos pelo vetor de atributos Vf' (linha 2). O classificador C_i é construído com a subamostra $S_{i(Vf')}$, (linha 3). Por último, os classificadores são combinados por alguma técnica de fusão, neste caso, pelo Voto Majoritário.

Algoritmo 7: Random Subspace (HO, 1998)

Entrada : τ é o conjunto de treinamento
f são as característica
Vf vetor de características
r o número de subproblemas.

Saída : ω combinação de classificadores

1 **para** $i = 1$ *até* r **faça:**
2 $S_{i(Vf')}$ = construa a subamostra aleatória do conjunto $\tau_{(Vf)}$
3 C_i = construa o classificador $S_{i(Vf')}$
4 **fim**
5 $\omega_{(x^*)} = maxcount\left[argmax\ \Phi_{C_i}(y|x^*)\right]$ /* retorne o resultado da combinação do conjunto de classificadores para o conjunto de teste. */
6 **retorne** ω

Vale salientar, para problemas de classificação com poucos atributos o RS pode não ser apropriado, pois há poucas possibilidades de combinação, logo algoritmo gera muitos classificadores semelhantes.

3.4 *Random Forest*

O método Random Forest (BREIMAN, 2001) consiste em criar Árvores de decisões por meio de subconjuntos de dados aleatórios. Muito semelhante ao *Bagging*, o *Random Forest* utiliza os mesmos princípios de amostragem do *dataset*.

Esse método se diferencia do *Bagging*, pelo classificador de base, que é sempre árvore decisão, e pela amostragem com diferentes quantidades de atributos sendo esta opcional. Quando não se opta pela criação dos subproblemas a partir dos atributos do problema, a sequência executada no *Bagging* se repete no *Random Forest*. Caso contrário novas subamostras são criadas com diferentes números de característica, isto é, cada subamostra terá seu próprio número de atributos derivados do problema de classificação.

Assim como no RS, se um problema de classificação possuir poucos atributos e quando se utiliza o número variado de características das subamostras na geração, podem existir classificadores iguais.

3.5 *Wagging*

O *Wagging* (BAUER; KOHAVI, 1999) é um método que utiliza pesos nas instâncias para geração do conjunto de classificadores. Os pesos são adicionados às instâncias utilizando o ruído gaussiano. Esse tipo de ruído é construído de acordo com a Equação 3.6, onde $Gauss$ representa a probabilidade de uma variável aleatória gaussiana em função da variável vl, que representa um valor aleatório, μ é a média e σ o desvio padrão.

$$Gauss(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-} \frac{(vl-\mu)^2}{2\sigma^2} \tag{3.6}$$

O uso do ruído gaussiano pode gerar pesos próximos a zero, isso faz com que as instâncias da formação das subamostras possam ser retiradas do conjunto. Os autores Quinlan (1996) propuseram uma variação do método *AdaBoost*, que também foi utilizada no *Wagging*. Nesta mudança, os pesos dado às instâncias são fornecidos pela distribuição de Poisson.

A distribuição de Poisson é calculada de acordo com a Equação 3.7. Na equação, β é um número inteiro não negativo e Re é um número real igual ao valor esperado de ocorrências que acontecem num dado intervalo de tempo.

$$P(\beta, Re) = \frac{e^{-Re} Re^{\beta}}{\beta!} \tag{3.7}$$

Por último, os passos do *AdaBoost* (Seção 3.2) dão continuidade ao processo de criação do *pool* de classificadores.

3.6 *MultiBoosting*

Baseado na ideia do *Bagging*, *AdaBoost* e *Wagging*, o *MultiBoosting* é um método que reúne todas as técnicas. Em seu trabalho Webb (2000) realizou algumas análises nos métodos *Bagging* e *AdaBoost*, de modo que encontrou lacunas nessas técnicas. Duas análises se destacaram nesse trabalho: o erro gerado pelo viés e pela variância. O viés ao quadrado que é uma medida que calcula o erro da tendência de uma classificação, ou seja, o classificador tende a indicar sempre a mesma classe. Essa medida avalia o poder de generalização do conjunto de classificadores. Já a variância é uma medida do erro de desvios da tendência central (tendência de uma classificação), isto quer dizer, classificadores distantes da classificação correta (WEBB, 2000).

O viés e a variância são avaliados em relação a um conjunto de treinamento. Para tanto, é criada uma distribuição que contém todos os subconjuntos de treinamento possíveis e que tenham um domínio específico. Se o resultado das previsões de um conjunto de classificadores diferir, então essa diferença retratará o erro médio dos classificadores.

O estudo de Webb (2000) indicou o uso de análises do viés e da variância focando em fatores significativos na precisão dos classificadores. O método inicia criando um grupo de especialistas com a técnica do *Bagging*. Após criados os primeiros subproblemas, o método segue os mesmos passos do *AdaBoost* e do *Wagging*. Os pesos atribuídos as instâncias derivam da Equação de Poisson 3.7, descrita na variação do método *Wagging*.

Na segunda etapa, é avaliado o conjunto classificadores pelo erro do viés e da variância. Para terminar, o método utiliza esses resultados para alterar o tamanho do conjunto de classificadores. Então, os especialistas que não atenderam aos critérios mínimos de erro são retirados do conjunto. O tamanho final do *pool* de classificadores é fixo, e se em algum momento o número de modelos ficar menor que o definido, novos classificadores serão gerados.

De acordo com Webb (2000) o *Bagging* reduz principalmente a variância, enquanto o *AdaBoost* reduz o viés e a variância. O objetivo desse método é aprimorar o *AdaBoost* nesses quesitos. Com esse estudo o autor pôde concluir que a precisão dos classificadores melhorou em relação às outras estratégias descritas em seu trabalho.

3.7 *Rotation Forest*

Como o *Random Subspace*, a Floresta Rotativa (do inglês *Rotation Forest RF*) (RODRIGUEZ; KUNCHEVA; ALONSO, 2006) é um método de geração de classificadores baseado nas características do problema. A RF visa ser robusta aos ruídos de um problema de classificação.

O processo de construção de cada classificador da RF é feito da seguinte maneira: o espaço de atributos Vf é dividido em r subproblemas. Se o número de atributos é igual a m, então cada amostra terá o número de atributos $m' = \frac{r}{m}$.

Durante a execução de cada *fold*, um subconjunto aleatório de instâncias (x, y) é retirado do conjunto de treinamento original, τ, produzindo um grupo de treinamento τ'. Então, gerar-se-á, um subproblema a partir de τ' (S_{ij}), nessa etapa i retrata cada classificador que será criado e j cada *fold* de um total de fl. Em seguida, é aplicado o Análise de Componentes Principais, do inglês *Principal Component Analysis*, (PCA) (PEARSON, 1901) em cada subconjunto S_{ij}. Extrai-se os coeficientes do PCA, representados por Cp e armazenados em uma matriz chamada de rotação (Mat) descrita na Equação 3.8.

$$Mat = \begin{pmatrix} Cp_{ij} & 0 & ... & 0 \\ 0 & Cp_{ij} & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & ... & 0 & Cp_{ifl} \end{pmatrix} \tag{3.8}$$

onde i é índice do classificador do *pool*. A última etapa consiste na construção da matriz de rotação Mat', reorganizando as colunas da matriz de coeficientes (Mat) em relação à ordem original dos atributos. O Algoritmo 8 exemplifica os passos descritos anteriormente.

O método retrata uma melhora na acurácia de classificação em comparação com outros métodos de classificação e até outras abordagens de conjunto de classificadores. O algoritmo proposto foi comparado ao *Bagging*, *AdaBoost* e *Random Forest* e foi validado em 33 problemas de classificação do repositório UCI. Os resultados demonstram que o RF superou os três métodos com grande margem de acurácia.

Algoritmo 8: Rotation Forest (RODRIGUEZ; KUNCHEVA; ALONSO, 2006)

Entrada : r, τ, (x, y), Vf, onde:
r = número de subproblemas e número de classificadores do conjunto
τ = Conjunto de treinamento
y = rótulos dos exemplos
V_f = conjunto de atributos
Saída : C_c

1 **para** $i = 1$ *até* r **faça:**
2 τ'_{ij} = divida o conjunto de atributos Vf em fl *folds*
3 **para** $j = 1$ *até* r **faça:**
4 τ'_{ij} = selecione e elimine os subproblemas com uma determinada classe y, e mantenha o conjunto de treinamento restante.
5 S'_{ij} = forme um subproblema de τ'_{ij}
6 Mat = aplique o PCA em S'_{ij} e armazene os coeficientes principais Cp, j
7 **fim**
8 $Mat'_{ij} = Mat_{ij}$ de acordo com a ordem original dos atributos de τ.
9 C_i = construa o classificador com $\tau Mat'_{ij}$.
10 **fim**
11 **retorne** C_c

O Algoritmo 8 inicia com uma estrutura de repetição que parte de 1 até o numero de subproblemas desejado (linha 1). Na segunda linha o conjunto τ é dividido em fl *folds*.

No *loop* interno o algoritmo separa as instâncias de acordo com as classes do problema, forma-se um subproblema e aplica o PCA sobre S'_{ij}. Na linha 8 e 9 a matriz é organizada e então, os classificadores são treinados. Finalmente, o método retorna o conjunto de classificadores C_c.

3.8 *RotBoost*: Uma técnica para combinar Florestas Rotativas e *Boost*

O *RotBoost* é um método de geração de classificadores proposto por Zhang e Zhang (2008). O *RotBoost* une o *AdaBoost* com a Floresta Rotativa. A estratégia de geração desse método consiste em duas etapas, a geração dos classificadores e sua combinação.

Na primeira etapa, os subproblemas são construídos pelo *AdaBoost*. Cada subproblema construído irá constituir uma matriz de rotação. Com essa matriz de rotação, criam-se os modelos de classificação. Na segunda etapa é realizada a combinação desses classificadores pelo voto majoritário. Os resultados encontrados foram superiores ao RF e o *AdaBoost*.

3.9 Caracterização do Oráculo em um *pool* de classificadores para os algoritmos de seleção dinâmica de classificadores

O método proposto por Souza et al. (2017) apresenta outra abordagem para criação de um conjunto homogêneo de classificadores. Diferente dos métodos anteriores, que utilizam a variedade do problema para criar as subamostras, esse método utiliza o conceito de Oráculo.

O Oráculo foi definido por Kuncheva (2002) como um modelo abstrato o qual sempre se seleciona o classificador que prevê a classe correta, caso esse classificador exista. Este conceito é frequentemente utilizado para medir o desempenho do limite superior que se pode atingir em um método SD. Em outras palavras, um Oráculo de 100% significa que cada instância no conjunto de treinamento é classificada corretamente por pelo menos um dos classificadores do *pool*.

Ainda no trabalho de Souza et al. (2017), o *pool* é induzido a fornecer um Oráculo próximo ou igual a 100%. Para atingir esse objetivo, algumas técnicas foram utilizadas para selecionar as instâncias que participarão das subamostras. O Algoritmo 9 descreve o funcionamento do método.

Primeiramente, na linha 1 é criado um vetor vazio C_c, para armazenar o *pool*. No *loop* interno, entre as linhas 3 e 5, são calculados os centroides *ce*, para cada classe do problema e armazenados na matriz M_{ce} (matriz de centroides). Entre as linhas 6 e 11, método encontra o ponto mais distante, $Dist$, do centroide de uma classe para todas as

outras e armazena os resultados no vetor $Vdist$, por último encontra-se o ponto médio entre as matrizes de distância e armazena em Pm. O algoritmo segue, encontrando a normal de M_{ce}, cria-se um hiperplano entre as instâncias separadas pelo ponto médio Pm, com base no vetor de distância ($Vdist$) e a Normal encontrada. O maior número de instâncias separadas pelo hiperplano irá determinar o peso do hiperplano (Perceptron h_w).

Entre as linhas 11 e 15, as instâncias x_i são testadas nos classificadores, e se forem classificadas corretamente, irão participar na criação do modelo. Os exemplos classificados incorretamente voltaram para o *loop*, até que todos sejam classificados corretamente.

Algoritmo 9: Caracterização do Oráculo em sistemas de multi classificadores (SOUZA et al., 2017)

Entrada : τ conjunto de treinamento
Vc conjunto de classes do problema
Saída : C_c *pool* de classificadores

1 $C_c = \emptyset$
2 **enquanto** $\tau \neq \emptyset$ **faça:**
3 **para** $j = 1$ *até* ty **faça:**
4 M_{ce}= $ce(j)$ de y_j
5 **fim**
6 $Dist = maior(DistanciaPareada(Mce))$
7 $Vdist = Index(Dist)$
8 $Pm = Vdist(Mce(Vdist) + Mce(Vdist)) \div Vdist$
9 $Normal = (Mce(Vdist) - Mce(Vdist)) \div Dist$
10 $h_w = \text{Perceptron}(Normal, Pm)$
11 **para** $i = 1$ *até* m **faça:**
12 **se** *teste*(h_w, x_i)==y_i **então:**
13 $\tau = \tau - \{x_i\}$
14 **fim**
15 **fim**
16 $C_c = C_c \cup h_w$
17 **fim**
18 **retorne** C_c

Os resultados alcançados mostraram que, mesmo com o conjunto de classificadores com Oráculo próximo ou igual 100%, os métodos de seleção dinâmica de classificadores não conseguem atingir esse limite. A razão para isso se deve ao fato de que, embora o modelo Oráculo seja executado globalmente, as técnicas de DCS e DES usam apenas dados locais para selecionar o melhor classificador. Desse modo, o Oráculo não é recomendado como métrica de avaliação do *pool* (SOUZA et al., 2017).

3.10 Geração de *pool* local e dinâmico para Seleção Dinâmica de classificadores

O método de geração de *pool* local e dinâmico gera um subconjunto de classificadores em regiões de sobreposição de atributos. Essa estratégia proposta por Souza et al. (2018) é dividida em dois estágios principais: a fase estática e a dinâmica. Na primeira, cada exemplo do conjunto de treinamento é avaliado por uma métrica que exprime a dificuldade de se classificar uma instância x. Essa dificuldade é extraída pelas medidas conhecidas como *Instance Hardness*. Neste estudo a medida de *Instance Hardness* utilizada foi a Desacordo dos vizinhos de K, (do inglês *K-Disagreeing Neighbors* (KDN)) (SMITH; MARTINEZ; GIRAUD-CARRIER, 2014).

A kDN pode ser observada na Equação 3.9. A métrica mensura a sobreposição de cada instância, x, em um conjunto de treinamento τ. A medida do kDN representa a porcentagem de instâncias na vizinhança de um exemplo que não compartilham o mesmo rótulo que ela. Portanto, os valores maiores de kDN significam que a instância avaliada está em uma região de sobreposição ou não (SOUZA et al., 2018). Para mensurar o valor supracitados, a métrica utiliza o recurso dos k vizinhos mais próximos do algoritmo KNN. Na Equação 3.9, k representa o número de vizinhos, x as instâncias, y o rótulo de cada instância. Os rótulos dos vizinhos do exemplo x_j são comparados com a classe do exemplo x_i. Ainda na fase estática do método, o resultado encontrado da kDN será utilizado para definir as regiões onde será aplicada a construção do *pool*.

$$kDN(x_i, \tau, k) = \frac{\mid x_j : x_i \in kNN(x_i, \tau, k) \wedge y_j \neq y_i \mid}{k} \tag{3.9}$$

Na fase dinâmica, para cada instância de treinamento define-se a região de competência por algum método de seleção dinâmica. Caso a região encontrada estiver em um local considerado difícil pelo resultado da Equação 3.9 (valores altos é considerado uma região difícil, ou seja, uma região de sobreposição (SOUZA et al., 2018)), gera-se um *pool* com o método descrito na Seção 3.9. Caso a região seja considerada fácil de acordo com o resultado do kDN, utiliza-se um KNN simples para classificar o exemplo.

As Figuras 12 (a) e (b) definem todas as etapas desse trabalho, incluindo a fase de generalização (resultado). A Figura 12 (a) define a etapa estática, onde a kDN é aplicada no conjunto de treino τ, com o resultado obtido tem-se a estimativa de todas as regiões consideradas difíceis para a classificação (ϑ). Na Figura 12 (b), a primeira etapa define a região de competência por algum método de SDC ou SDE (δ). Dentro dessa região, o método avalia se o local é considerado difícil por ϑ, se verdadeiro um *pool* é gerado com o método definido na Seção 3.9. A combinação dos classificadores do *pool* ficará sob responsabilidade do voto majoritário. Se a região não for considerada difícil, o algoritmo aplica um KNN.

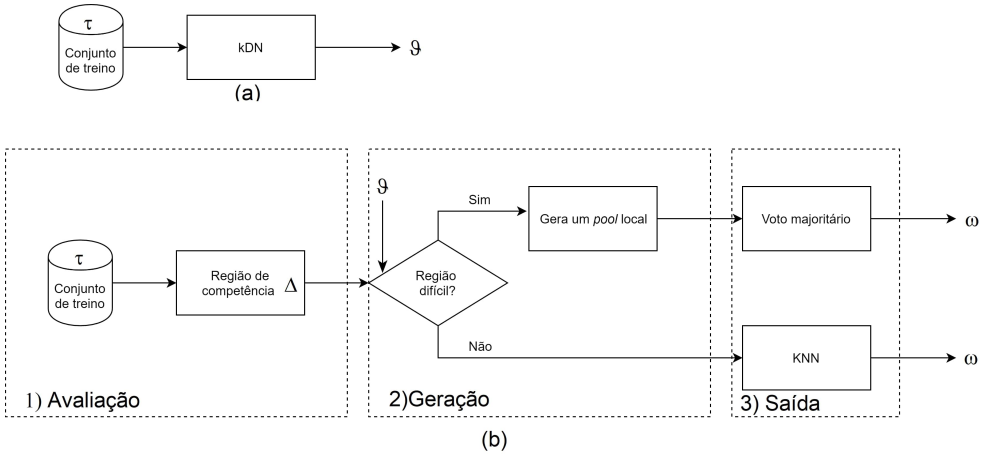


Figura 12 – Etapas para geração do *pool* dinâmico.

Fonte: Adaptado (SOUZA et al., 2018).

No estudo foram utilizados 20 problemas da literatura. O *pool* dinâmico foi comparado ao Bagging e aplicado nos métodos de seleção dinâmica de classificadores OLA, LCA, META-DES, KNORA-U (Seção 2.4), Comportamento de multiplos classificadores, (do inglês *Multiple Classifier Behaviour* (MCB)) (GIACINTO; ROLI; FUMERA, 2000) e Classificador de referência aleatório, (do inglês *Randomized Reference Classifier* (RRC)) (WOLOSZYNSKI; KURZYNSKI, 2011).

O desempenho dessa proposta demonstra que o uso de *pools* locais aumentou a taxa de acertos do *pool* global, para a maioria dos conjuntos de dados testados. O uso de tais *pools*, ajudou as técnicas SDC e SDE a selecionar os classificadores mais competentes para uma determinada instância de teste.

3.11 Poda do conjunto de classificadores baseada na Curva de Pareto: um algoritmo de poda que aprende a procurar *ensembles* otimizados

O trabalho Poda do conjunto de classificadores baseada na Curva de Pareto apresenta duas contribuições: o algoritmo de otimização de Pareto que é aplicado para calcular o tamanho ótimo do conjunto de classificadores durante o estágio de geração do *pool* e uma nova métrica de seleção dinâmica de classificadores.

O tamanho do conjunto de classificadores é geralmente fixo e o desempenho de um conjunto de classificadores depende muito da definição da região de competência (HU et al., 2019). Muitas vezes, um *pool* de classificadores de tamanho fixo pode não representar o problema por completo. Nesse sentido, os autores Hu et al. (2019) propõem a otimização da geração do *pool* de classificadores, baseada na Curva de Pareto. A estratégia consiste

em um mecanismo de busca global e busca local para definir o tamanho do conjunto de classificadores.

Na geração de classificadores é criado um *pool* inicial com métodos consolidados na literatura, podendo ser o *Bagging*, *AdaBoost*, entre outros. Após criado esse *pool*, verifica-se a competência global de cada classificador gerado com as subamostras do problema, a competência é obtida de acordo com um algoritmo KNN. Na próxima etapa, a região de competência de cada classificador é avaliada aplicando as instâncias do conjunto de validação. Usam-se os valores obtidos da região de competência para o algoritmo de otimização de Pareto, definindo assim, o tamanho do *pool*.

A principal diferença do método de seleção dinâmica analisada pelos autores e comparado a outros métodos é que a região de competência é extraída durante a fase de geração do *pool*. Outros métodos já apresentados na Seção anterior 2.4 utilizaram a fase de teste para encontrar a região de competência de cada instância.

A construção do *pool* com a Curva de Pareto mostrou-se ser uma alternativa viável, quando comparada com métodos que utilizam tamanho do *pool* fixo. Hu et al. (2019) apresentaram o tamanho ideal para cada *pool* de classificadores em diferentes métodos SD. Na avaliação do método foram utilizados 16 problemas de classificação em 6 algoritmos de SD.

3.12 Um *framework* para seleção de dinâmica de classificadores orientado pela complexidade do problema de classificação

O método proposto por Brun et al. (2018), o qual fornece inspiração para esse estudo, descreve o uso de medidas de complexidade para geração de um *pool* de classificadores. Além disso, o método faz uso das medidas de complexidade na criação de um método de seleção dinâmica para *pool* de classificadores. As medidas de complexidade F1 e N2 (Seção 2) e a acurácia são aplicadas na geração do *pool* de classificadores. As métricas citadas anteriormente e a acurácia são combinadas na forma de distância média F1 *versus* N2, o conjunto de classificadores é gerado com o uso de um Algoritmo Genético (AG).

O método inicia com a criação de um grupo de subamostras (S) de tamanho r a partir de um conjunto de treinamento τ. Cada amostra S_i é construída com u exemplos, distribuídos aleatoriamente de τ. Inicia-se o AG com os r subproblemas produzidos. Todas as amostras são submetidas à avaliação das medidas F1 e N2 ($S_i[F1_i, N2_i], ..., S_r[F1_r, N2_r]$). Os vetores criados com essas medidas consistem nos subproblemas que ocupam o espaço de complexidade e podem ser representados por C_{ori}, então calcula-se a distância média de cada subamostra com as demais. A função de *fitness* do AG é baseada nessa distância e na acurácia obtida a partir de um Perceptron treinado em cada subconjunto.

A Equação 3.10 demonstra a função *fitness* do AG nesse método, onde ACC_{C_i} é a acurácia de cada classificador formado pelas subamostras e $Dist_{S_i}$ representa a distância

normalizada de cada subproblema no espaço de complexidade perante as demais. A acurácia dos subproblemas é resultado do teste com uma amostra de validação. O resultado do *fitness* é a soma dos valores encontrados da distância e acurácia ($\mathcal{F}_{S_i}$). A seleção dos indivíduos da população utilizada no AG foi a roleta (Seção 2.3.1).

$$\mathcal{F}_{S_i} = ACC_{C_i} + Dist_{S_i} \tag{3.10}$$

A função de cruzamento do método consiste na troca de instâncias (x) entre dois subproblemas S_i e S_j são compostos por u pares de exemplos (x_i, y_i) e escolhidos aleatoriamente. O método se atenta em manter as amostras estratificadas, ou seja, (x_1, y_1) será trocada por (x_2, y_2) se e somente se $y_1 = y_2$.

O método ainda propõe a seleção dinâmica (do inglês *Dinamic Selection Over Complexity* (DSOC)) de classificadores com o uso das mesmas medidas. A seleção dinâmica de classificadores possui cinco etapas: (i) encontra-se a acurácia local (ACC) para cada instância de teste x^*; (ii) encontra-se os centroides (ce) de cada classe e a distância ($Dist_{ce}$) entre x^* e ce; (iii) para cada instância de teste, gera-se um subproblema com os vizinhos da base de validação e extrai suas complexidades (F1, N2, N4); (iv) realiza a diferença entre as complexidades de cada amostra $dif = C_{ovi} - C_{osi}$, onde C_{ovi} corresponde a complexidade da amostra de validação do subproblema i e C_{osi} representa a vizinhança da amostra da amostra i; (v) por fim, realiza-se a soma de todos os valores encontrados. O classificador que possuir o maior resultado encontrado ($Compe$) será escolhido. A Equação 3.11 representa a combinação dos três resultados.

$$Compe = ACC + Dist_{ce} + dif \tag{3.11}$$

Com o estudo desse artigo foi concluído pelos autores (BRUN et al., 2018) que em 165 de 180 experimentos (91,67%) a adoção do AG proposto para gerar o *pool* combinado com o método de seleção dinâmica, permitiu uma melhora na acurácia da classificação em relação a outros métodos de seleção dinâmica de classificadores com *pool* gerado pelo *Bagging* (BRUN et al., 2018).

3.13 Considerações finais

A Tabela 6 apresenta um resumo dos métodos abordados nesse capítulo. Pode-se observar que a grande parte dos trabalhos de geração de conjunto de classificadores exploram a abordagem da distribuição aleatória dos dados do problema. Diferentes metodologias são empregadas para gerar conjuntos de classificadores, partindo das mais simples como o *Bagging*, até métodos mais complexos, onde envolvem o uso de PCA com alto custo computacional (Floresta Rotativa). Também nota-se na tabela que poucos trabalhos exploram as meta características do problema de classificação, apenas os trabalhos de

(SOUZA et al., 2017), (SOUZA et al., 2018) (Oráculo e dificuldade da instância) e (BRUN et al., 2018) (complexidade do problema).

Tabela 6 – Trabalhos relacionados a criação de *pool* de classificadores.

Método	**Abordagem/objetivo**	**Autores**
Bagging	Amostragem horizontal	(BREIMAN, 1996)
AdaBoost	Amostragem horizontal	(FREUND; SCHAPIRE, 1996)
Random Forest	Amostragem vertical e horizontal	(BREIMAN, 2001)
Wagging	Amostragem horizontal	(BAUER; KOHAVI, 1999)
MultiBoosting	Amostragem horizontal	(WEBB, 2000)
Random Subspace	Amostragem vertical	(HO, 1998)
Floresta Rotativa	Amostragem vertical	(RODRIGUEZ; KUNCHEVA; ALONSO, 2006)
RotBoost	Amostragem vertical	(ZHANG; ZHANG, 2008)
Caracterização do Oráculo	Oráculo	(SOUZA et al., 2017)
Geração de *pool* dinâmico	Oráculo	(SOUZA et al., 2018)
Poda CP	Determinação do tamanho do *pool*	(HU et al., 2019)
DSOC	Amostragem, com medidas de complexidade	(BRUN et al., 2018)

Fonte: O autor.

Os métodos que envolvem a amostragem buscam, em sua maioria, gerar classificadores com diversidade entre as subamostras do problema por meio dos seus dados. A diversidade possui um papel fundamental na criação de conjuntos de classificadores (KUNCHEVA; WHITAKER, 2003). Parece razoável, se obter diversidade por meio das propriedades de complexidade do problema. Diante deste ponto de vista, o *pool* gerado com diferentes níveis de dificuldade poderá impactar na precisão global do conjunto de classificadores.

O método proposto no próximo capítulo, que compreende a proposta do presente trabalho, é inspirado e estende a estratégia elaborada Brun et al. (2018).

4 Método proposto

A estratégia de geração de *pool* homogênea descrita no Capítulo 3 explica diferentes maneiras de se obter subamostras de treinamento (ou subproblemas) diversificadas em diferentes regiões do problema de classificação.

Este capítulo apresenta um novo método de geração de *pool* de classificadores homogêneo que explora meta-características de complexidade do problema. O objetivo do método proposto consiste em maximizar a distância entre subproblemas (a partir de um problema de classificação) utilizando meta-características de complexidade e Diversidade no espaço de Decisão (DD). Com isso, espera-se obter classificadores diversos, com diferentes níveis de complexidade e que possam contribuir para um melhor desempenho, em termos de acurácia, dos métodos de SDC e SDE.

O método proposto é intitulado Geração de *Pool* de Classificadores fundamentado na Diversidade de dois níveis ($GPCD$), isto significa, um conjunto de classificadores com base na complexidade e na diversidade. O $GPCD$ é dividido nas seguintes etapas: (i) análise da complexidade dos dados do problema de classificação, seguido da (ii) geração do *pool* via algoritmo genético multiobjetivo.

A Figura 13 apresenta uma visão geral das duas fases do método. Na fase de análise da complexidade, o método utiliza um conjunto de treinamento para gerar novos subproblemas. Com essas subamostras são realizadas as análises da dispersão de cada medida de complexidade. As métricas que apresentarem a maior variação no espaço participarão da segunda fase do método.

Na segunda fase, um Algoritmo Genético Multiobjetivo modificado (AGM) recebe como entrada um conjunto de subproblemas como população inicial (a geração dos subproblemas é detalhada nas próximas seções). A evolução desses subproblemas é orientada pelos resultados obtidos das medidas de complexidade. Ainda nessa fase o método avalia cada geração de indivíduos de duas maneiras diferentes, originando duas variações do método principal, uma focada em diversidade e outra em acurácia. Na segunda fase, um Algoritmo Genético Multiobjetivo modificado (AGM) recebe como entrada um conjunto de subproblemas como população inicial. A evolução desses subproblemas é realizada por meio dos resultados obtidos das medidas de complexidade. Ainda nesta fase o método verifica cada geração de indivíduos de duas maneiras diferentes, originando duas variações do método principal.

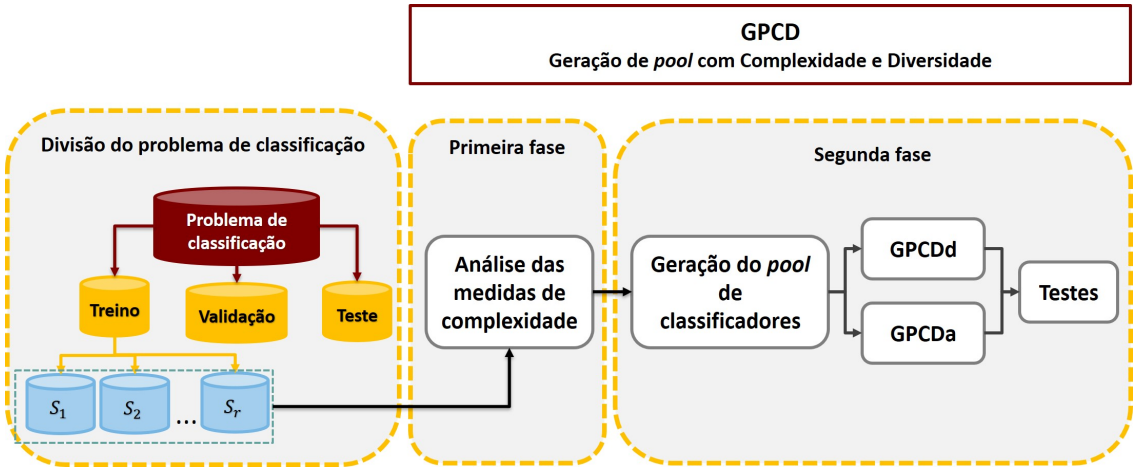


Figura 13 – Representação geral do método. O método se divide em duas etapas principais, e cada uma delas possui suas subdivisões.

Fonte: O autor.

4.1 Análise das medidas de complexidade

A primeira parte do $GPCD$ visa encontrar as métricas de complexidade que melhor se adaptam aos problemas de classificação. A Figura 14 apresenta um fluxograma com os passos utilizados nessa etapa. É possível observar duas divisões principais: **Cálculo das medidas de complexidade** e **Peso para as medidas de complexidade**. Tais divisões estão detalhadas nas Figuras 15 e 16, respectivamente.

No segundo passo do fluxograma da Figura 14, demonstra-se o conjunto de treinamento τ, formado a partir de um *dataset*. No terceiro processo (Figura 14), gera-se r subconjuntos $(S_1, S_2, ..., S_r)$ aleatoriamente com repetição e com estratificação, isto é, considera o balanceamento entre classes, estratégia essa utilizada no *Bagging* (BREIMAN, 1996).

Nessa fase, e em todas etapas do método que exigem aleatoriedade, foi utilizada a função pseudoaleatória uniforme *Mersenne Twister* cuja semente corresponde à hora corrente do sistema. A *Mersenne Twister* (MATSUMOTO; NISHIMURA, 1998) passou por diversos testes empíricos que comprovaram sua eficiência na geração de números pseudos-aleatórios (L'ECUYER; SIMARD, 2007).

No próximo passo (14.4) são calculadas as complexidades de todos os subproblemas. No quinto processo (14.5), encontra-se a dispersão para cada MC. Por último, aplica-se um peso para cada medida de complexidade (14.6). Todos esses processos se repetem por T^* iterações.

Cálculo das medidas de complexidade - A Figura 15 trata da obtenção e armazenamento das medidas de complexidade, sendo que S são os subproblemas, r número de subamostras (subproblemas), $MC^v_{o_i}$ são os resultados obtidos para cada medida o da

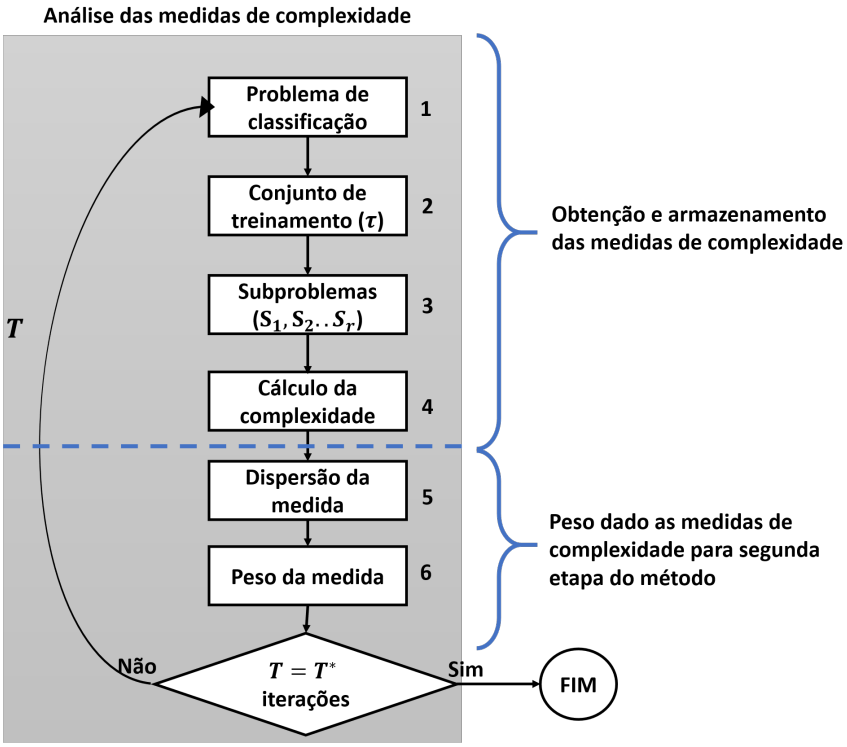


Figura 14 – Etapas da primeira fase do método. Pode-se dividir todos os processos em duas partes distintas: Levantamento das medidas de complexidade e o peso dado a cada uma delas.

Fonte: O autor.

família v do subproblema i. O símbolo o' significa o número total de medidas da família e v' o total de famílias.

Os resultados encontrados para cada medida são normalizados. A normalização é feita pelo método $MinMax$ representado pela Equação 4.1, onde MC_o consiste na medida de complexidade o da família v, MC_{min} e MC_{max} representam, respectivamente, o menor e o maior valor atingido pela medida de complexidade. O resultado de $MinMax_o$ é a MC normalizada entre zero e um.

$$MinMax_o^v = \frac{MC_o^v - MC_{o_{min}}^v}{MC_{o_{max}}^v - MC_{o_{min}}^v} \tag{4.1}$$

Na etapa seguinte, os resultados obtidos são armazenados na matriz $\wp_v$, em que v representa a família das medidas de complexidade. Cada matriz $\wp$ contêm os resultados das MCs dos r subproblemas e o' métricas de complexidade.

Peso para as medidas de complexidade - Após cada iteração, cada matriz $\wp$ armazena os resultados de todas as medidas de complexidade para cada subproblema de classificação. Com esses resultados calcula-se a dispersão das medidas a fim de encontrar as que possuem a maior variação. A ideia consiste em criar um *pool* de classificadores com diferentes níveis de complexidade, logo espera-se que as medidas com maior variação possam contribuir trazendo maior diversidade entre os classificadores. A dispersão da

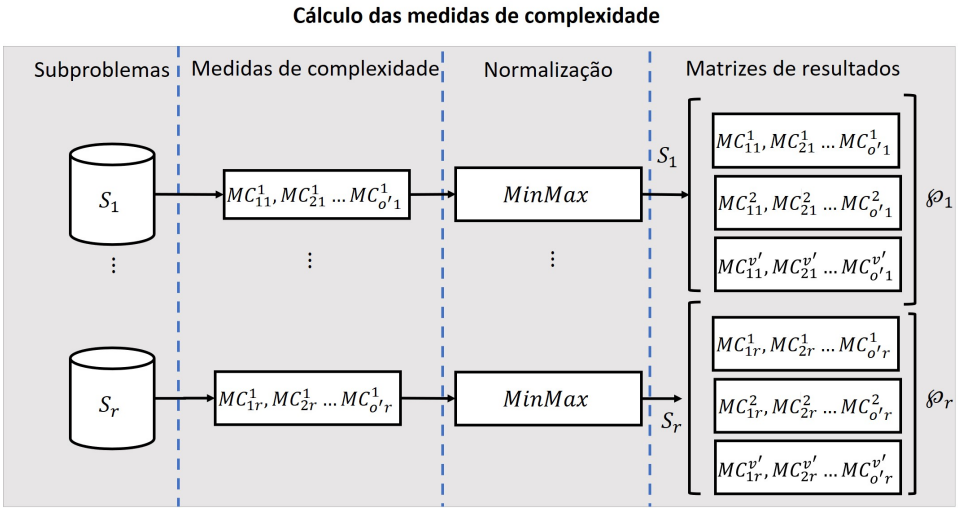


Figura 15 – Análise das medidas de complexidade, calcula-se a complexidade de todas os subproblemas, normaliza os valores entre zero e um. Por último o resultado é armazenado na matriz $\wp$.

Fonte: O autor.

complexidade é dada pela Equação 4.2. Ela representa o desvio padrão calculado para cada métrica, sendo que MC_i^j é a medida i (onde $i \in \{1, 2, 3, ..., o'\}$) do conjunto (família) j (em que $j \in \{1, 2, 3, ..., v'\}$) avaliado, μ representa a média e l o total de resultados da métrica para os r subproblemas.

$$\sigma_o = \sqrt{\frac{\sum_{l=1}^{r}(MC_{il}^j - \mu_{ij})^2}{r}} \tag{4.2}$$

A Figura 16 ilustra como é feito o esquema do voto. Com a matriz $\wp$, onde as colunas representam os valores das medidas e as linhas correspondem aos subproblemas, é calculada a dispersão de todas as métricas. O resultado é armazenado na lista de dispersão. Os valores de σ_o são os desvios-padrão do conjunto de MC_o correspondente à coluna da matriz, logo a lista σ terá tamanho igual o' (onde $o \in \{1, 2, 3, ..., o'\}$) e o número de listas será igual a v'.

As medidas que possuírem a maior dispersão em cada grupo recebem um peso maior. Em caso de empate utiliza-se a métrica que obteve os maiores valores de dispersão, ou seja, mesmo que em determinada geração ela não atinja o maior desvio padrão, utiliza-se os resultados com maior variação, em média, entre as iterações. Depois de T iterações, a medida que possuir o maior peso em cada família participará da segunda fase do método, importante destacar que a cada iteração o peso é incrementado para as medidas mais dispersas.

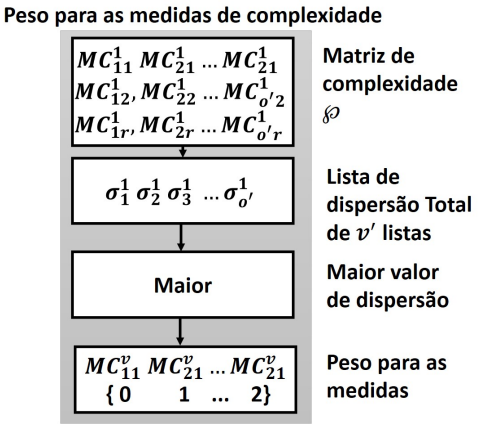


Figura 16 – Representação do esquema para atribuição de pesos para as medidas de complexidade de cada grupo.

Fonte: O autor.

O objetivo dessa fase é obter as grandezas com maior variação. Espera-se que quanto maior a oscilação de uma medida, maior será a possibilidade de se obter classificadores com diferentes níveis de complexidade. Os resultados dessa etapa serão discutidos no próximo capítulo.

4.2 Geração do Pool

Na segunda etapa intitulada "geração do *pool*", um Algoritmo Genético Multi-objetivo (AGM) aplica as medidas de complexidade encontradas na primeira etapa em cada problema de classificação. Uma nova medida denominada medida de diversidade também faz parte dessa etapa do método. O objetivo é formar um conjunto de especialistas que explora ambos os espaços (diversidade em termos de opinião dos classificadores e complexidade dos dados de treinamento) da melhor forma. O método tenta cobrir esses espaços de duas maneiras distintas. A primeira avalia a acurácia do *pool* em cada geração do AGM, para isto utiliza-se a combinação dos classificadores. Neste caso a geração com o melhor resultado será o conjunto de classificadores final. Essa variante do $GPCD$ é chamada de $GPCDa$.

A outra alternativa, é escolhida a geração com a distância média mais significativa entre os indivíduos (subproblemas), melhor dizendo, os indivíduos que estão mais dispersos no espaço, essa versão é chamada de $GPCDd$.

Para gerar as subamostras com diferentes níveis de complexidade e diversidade de opinião dos classificadores, foi criada uma estratégia em que, as amostras devem estar distantes umas das outras de acordo com suas meta-características. Com isso, espera-se obter um conjunto de classificadores com maior diversidade e consequentemente mais acurados.

A Figura 17 representa a segunda etapa do método. Primeiro, escolhe-se qual variação do método será executada. As principais entradas do método são: a população inicial gerada na etapa anterior ou o conjunto de treinamento para a então criação dos subproblemas, a variação do método ($GPCDa$ ou $GPCDd$) e as medidas de complexidade, isto é, as medidas e a família de medidas escolhidas na primeira etapa. Como resultado, tem-se a geração cuja população seja mais acurada ou mais dispersa para formar o *pool*.

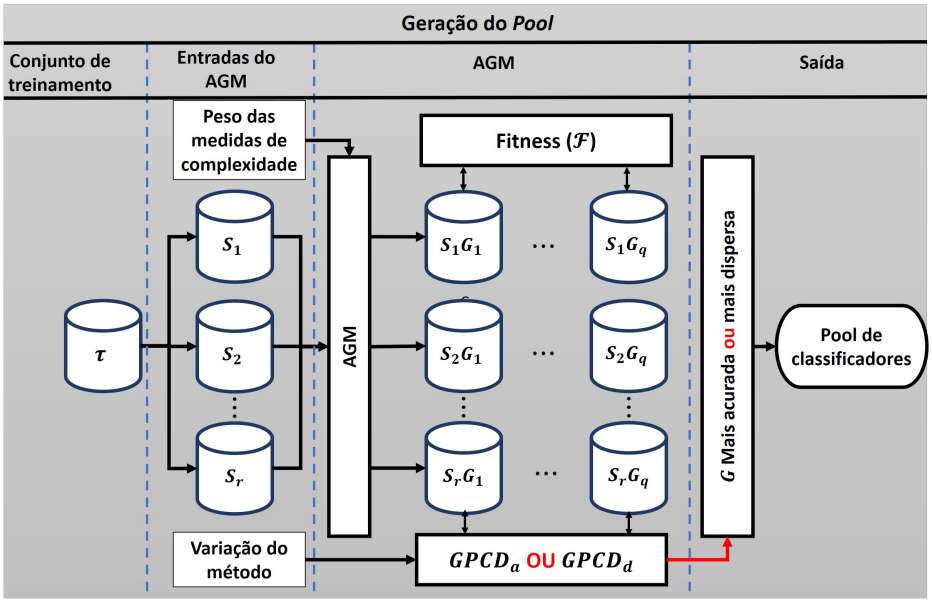


Figura 17 – Representação da segunda etapa do $GPCD$. A imagem retrata as duas possibilidades para executar o método.

Fonte: O autor.

Para uma melhor compreensão do método proposto ($GPCD$), a Tabela 7 explica a correlação entre os termos do AGM e do método $GPCD$.

Tabela 7 – Correlação entre os termos do AGM e o $GPCD$.

AGM	Equivalente
População	Conjunto de subproblemas
Indivíduo/Cromossomos	Conjunto de instâncias ou exemplos
Gene	Instância

Fonte: O autor.

4.2.1 Cruzamento

Uma das funções essenciais em um AGM é a função de cruzamento. Ela é responsável por gerar novos indivíduos para o alcance dos objetivos pré-estabelecidos. Para tanto, o operador de cruzamento agrupa os genes de pais para gerar um ou mais filhos com seus atributos.

A técnica escolhida para o cruzamento foi o "cruzamento de dois pontos". Esta escolha foi empírica, pois foi a que apresentou o melhor comportamento dos subproblemas para os objetivos desse método. O cruzamento presente no método consiste em escolher aleatoriamente dois indivíduos da população, em outras palavras, escolher dois subproblemas dentro do *pool*. O cruzamento gera um novo subconjunto de dados trocando parte das instâncias entre dois subproblemas (pais) S_1 e S_2 escolhidos aleatoriamente na população P_t. O Algoritmo 10 demonstra os passos desse processo.

Nas linhas 2 e 3, dois subconjuntos arbitrários de dados S_1 e S_2 são escolhidos na população P_t e o subconjunto resultante (filho) S^* é inicializado como uma estrutura vazia (linha 4). As variáveis que determinam os pontos de cruzamento (`im` e `fm`) são definidas, aleatoriamente, de acordo com uma distribuição uniforme representada por $\mathcal{B}$, sendo que u define o número de instâncias dos subproblemas. Contudo, o algoritmo preserva pelo menos dois representantes de cada classe do problema (linhas 5 - 6). Na estrutura de repetição as instâncias dos subproblemas S_1 e S_2 são trocadas e agrupadas para formar um novo subproblema (linhas 7 a 13). Esse modelo de cruzamento em AG específico para aprendizado de máquina foi desenvolvido por Macià, Orriols-Puig e Bernadó-Mansilla (2010) e foi utilizado por Brun et al. (2018) para construir novos subconjuntos de dados.

Algoritmo 10: Cruzamento

Entrada : $P_t, \mathcal{U}$
Saída : S^*

```
1  Function Cruzamento (P_t, B):
2      S_1 = B(P_t) /* selecione aleatoriamente um indivíduo em P_t */
3      S_2 = B(P_t \ {S_i}) /* selecione aleatoriamente um indivíduo em P_t */
4      S* = ∅
5      im = B(0, u)
6      fm = B(im, u)
7      para i = 1 até max(|S_1|) faça:
8          se i ≤ im ou i ≥ fm então:
9              S*[i] = S_1[i]
10         senão
11             S*[i] = S_2[i]
12         fim
13     fim
14 retorne S*
```

4.2.2 Mutação

Na mutação, um subproblema apresenta uma mudança em seus genes (instâncias). A função de mutação do método proposto, utiliza-se de um subproblema "doador" que fornece uma cópia da instância que será substituída no subproblema receptor. O algoritmo

verifica se o exemplo que será substituído pertence ao mesmo rótulo da instância fornecida pelo subproblema doador. Essa verificação mantém as amostras balanceadas.

A Figura 18 representa o processo de mutação, onde S_1 é um subconjunto que irá sofrer a mutação e S_3 é um subconjunto doador, ambos escolhidos aleatoriamente na população. No resultado obtido, ocorre a troca da instância x_7 pela cópia da instância x_2. O subproblema S_3 permanecerá inalterado. Entretanto, a amostra S_1 perde o exemplo x_7 e recebe x_2 no lugar, tornando-se "mutante".

Importante destacar que o processo de mutação permite repetição de instâncias, e de indivíduo tanto para doação como para repetição. O número máximo de mutações é equivalente ao número de filhos, contudo dentro de uma probabilidade para ocorrer pré-estabelecida e compartilhada com o cruzamento.

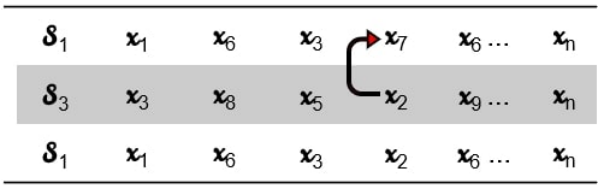


Figura 18 – Mutação, onde S_1 e S_3 representam os subproblemas, x_7 representa a instância que foi substituída por x_2.

Fonte: O autor.

4.2.3 Função objetivo

A função de *fitness* é responsável por orientar a convergência dos objetivos do método proposto. Neste trabalho a função de *fitness* possui três objetivos, o primeiro e o segundo são maximizar a distância média par a par para cada amostra no espaço da complexidade. Cada amostra é representada por dois valores de complexidade, sendo que cada medida pertence a uma família. As grandezas utilizadas nesse processo são originárias dos pesos obtidos na etapa anterior.

O terceiro objetivo é baseado na diversidade ao nível de decisão do classificador. O propósito é direcionar as amostras em dois espaços de meta-características, visando, maximizar a dispersão entre os classificadores em diferentes planos. Dando sequência ao algoritmo, são calculas as distâncias média par a par entre os valores correspondentes na mesma família de complexidade. Sendo assim cada subconjunto de dados está relacionado a v valores de distância, onde v significa as famílias de complexidade. A Equação 4.3 demonstra como se obtém distância média $Dist_{MC}$ para cada amostra S_i, em que l representa a métrica de complexidade (onde $l \in \{1, 2, 3, ..., o'\}$), k as famílias (em que $k \in \{1, 2, 3, ..., v'\}$) e r como o número total de subproblemas. Os resultados das distâncias médias são armazenados no vetor $Dist_{MC_k}$.

$$Dist_{MC_k}[S_i] = \frac{\sum_{j=1}^{r} Abs(MC_l^k[S_i] - MC_l^k[S_j])}{r-1} \tag{4.3}$$

A proposta do $GPCD$ é aplicar diferentes grandezas de complexidades combinadas com uma medida de diversidade e torná-las igualmente ponderadas no processo de otimização de algoritmos genéticos.

Assim como nas métricas de complexidade, a ideia principal é aumentar a diversidade em termos de opinião de complexidade. Para isto, aplica-se a Equação 4.4, que se refere ao terceiro objetivo do método, onde $Diver_{[C_i]}$ é um vetor que armazena a média da diversidade, para cada modelo, aplicada par a par com r especialistas do *pool*. Os classificadores C_i e C_j são treinados com as amostras S_i e S_j, respectivamente, do *ensemble* de classificadores. Ainda nessa equação, MD representa uma medida de diversidade e r o número total de classificadores. Para a coleta das opiniões dos classificadores para a estimação das diversidades foi utilizado o conjunto de validação.

$$Diver_{[C_i]} = \frac{\sum_{j=1}^{r} MD(C_i, C_j)}{r-1} \tag{4.4}$$

O Algoritmo 11 descreve todas as etapas da função objetivo aplicada no $GPCD$. A entrada para a função é a população P_t. No primeiro *loop* (linhas 2 a 7), as métricas selecionadas na primeira etapa representadas por MC_o^v (v representa o grupo da medida de complexidade e o a medida para determinado problema), são calculadas para cada subproblema S_i. Os resultados encontrados são armazenados na matriz de dimensões $\theta[v][r]$, onde as famílias da medida são as linhas e S_i as colunas.

Na sequencia, um modelo C_i é criado para cada S_i considerando um classificador de base previamente definido. Os modelos treinados são utilizados no cálculo da diversidade. Na segunda estrutura de repetição (linhas 8 a 14), o algoritmo calcula para cada S_j a dispersão $Dist\theta[v][S_j]$ usando a Equação 4.3. Finalmente, a diversidade $Diver$ é encontrada para cada modelo C_i utilizando a Equação 4.4 (linhas 15 a 17), os resultados do *fitness* são armazenados na matriz de dimensões $\mathcal{F}[v+1][r]$.

4.2.4 Algoritmo Genético Multiobjetivo (AGM)

Na execução da segunda etapa do método, o $GPCD$ possui duas variações, realizadas separadamente e ambas as alternativas visam encontrar a melhor geração de acordo com critérios diferentes. Durante a execução do AGM, certas métricas de complexidade podem atingir seu limite de dispersão, ou seja, os subproblemas começam a perder variabilidade de complexidade. Isto ocorre junto a perda de diversidade, pois os exemplos do problema que definem valores expressivos de complexidade, em relação a outros exemplos, são considerados genes "bons". Logo, essas instâncias começam a se sobressair e se repetir nos subproblemas das gerações seguintes. O AGM ignora o limite de uma métrica específica, visto que os outros objetivos ainda têm espaço para dispersão.

Com o acima exposto, foram definidas duas variações do método, onde o objetivo é selecionar a geração cuja dispersão ainda não esteja em seu limite.

Algoritmo 11: Fitness

Input : P_t
Output : $\mathcal{F}$

Function `Fitness`(P_t):
  para $o = 1$ *até* o' **faça:**
    para cada S_i ***em*** P_t **faça:**
      $\theta[v][S_i] = MC_o^v$ /* calcule a medida para S_i */
      $C_i[S_i] = S_i$ /* treine o classificador com S_i */
    fim
  fim
  para $v = 1$ *até* v' **faça:**
    para cada S_j ***em*** P_t **faça:**
      para cada i ***em*** $\theta[v]$ **faça:**
        $\mathcal{F}[i][v] = Dist(\theta[v][S_j])$ /* calcule a distância média (Equação 4.3) */
      fim
    fim
  fim
  para cada i ***em*** $[C]$ **faça:**
    $\mathcal{F}[i][v+1] = Diver([C_i])$ /* calcule a diversidade média para todos os classificadores (Equação 4.4) */
  fim
  retorne $\mathcal{F}$

Na variação, chamada de acurácia máxima ($GPCDa$), o critério para finalizar o processo de otimização é o número de gerações (g^*). Entretanto, o algoritmo avalia cada geração, juntamente com um conjunto de dados de validação, a **acurácia média do *pool***. Seleciona-se a geração com maior acurácia ao final das g^* gerações.

A outra variação, chamada dispersão máxima ($GPCDd$), seleciona a geração, considerando todos os objetivos em que os indivíduos estão mais distantes entre si. Da mesma forma que o $GPCDa$, o AGM realiza as g^* gerações. Entretanto, somente a que possuir a **maior dispersão** será escolhida para formar o *pool*.

No $GPCDd$ é mantido o descritor global $\mathcal{F}$, composto pelos objetivos do AGM. A seleção dos indivíduos da população será definida, considerando a geração com maior dispersão de acordo com os dados contidos nesse descritor. Esse espalhamento é calculado, conforme indicado na Equação 4.5, onde $\mathcal{D}$ é a dispersão de todos subproblemas na geração G, r é o número de subconjuntos e k é o objetivo O do AGM onde, $k = \{1, 2, 3..., z\}$.

$$\mathcal{D}(G) = \frac{\sum_{i=1}^{r} S_i \sqrt{\sum_{j=1}^{r} (S_i \mathcal{F}[O_k] - S_j \mathcal{F}[O_k])^2}}{r - 1} \tag{4.5}$$

O AGM cria, a partir do conjunto de treinamento, a primeira geração em que cada subconjunto (indivíduo) S_i é gerado aleatoriamente, com estratificação e reposição dos exemplos do problema. O AGM é executado por g^* gerações (G) com os operadores descritos nas subseções anteriores.

O Algoritmo 12 descreve como é feita a geração do *pool* de classificadores. A função começa com um contador de gerações, i, e segue na linha 3 com a geração da população inicial P_t a partir de um conjunto de subproblemas. Então, na linha 4, a população tem seu *fitness* calculado ($\mathcal{F}$) e é gerado um *ranking* R dos indivíduos, pelo NSGAII. A posteriori o algoritmo entra em *loop* que inicia-se em zero até que se satisfaça o número de gerações g^* (linha 5 a 33).

Na estrutura de repetição entre as linhas 6 e 16, são criados u_n novos indivíduos I. Esses indivíduos são escolhidos aleatoriamente para participarem do cruzamento ou da mutação, entretanto o peso atribuído pelo *ranking* (realizado pelo NSGAII) aumenta a probabilidade de escolha dos indivíduos mais bem posicionados. As funções de mutação e cruzamento descritas nas seções anteriores possuem os mesmos nomes presente na Figura 4. Essas funções são escolhidas aleatoriamente de acordo com o resultado de var, a probabilidade do sorteio dessa funções são valores preestabelecidos.

Ao fim da estrutura de repetição interna, a população formada pelos filhos é avaliada pela função objetivo e adicionadas a um novo *ranking* R' (linha 17). Na linha 18 os membros da prole são adicionados ao conjunto formado pelos pais e são condicionados na variável Q. No próximo passo os indivíduos são selecionados de acordo com os critérios do NSGAII e atribuídos a população P_t. Neste passo se forma uma nova geração G.

Um dos parâmetros de entrada do algoritmo é utilizado para diferenciar as abordagens $GPCDa$ e $GPCDd$ (Figura 17). Entre as linhas 20 e 24 são aplicados os $GPCDa$ ou $GPCDd$ e os resultados médios são atribuídos a Γ. Da linha 25 até a 31 compara-se e os resultados de Γ com a da geração atual. Se o resultado for melhor que a geração anterior troca-se a geração, senão permanece a mesma. Finalmente i é incrementado até atingir o valor de g^* (linha 32). O algoritmo retorna a melhor geração, de acordo com os critérios das variações $GPCDa$ e $GPCDd$.

Logo a geração do *pool* (AGM) é idêntica para ambas variações. A única diferença é que nem sempre a última geração do AGM será a escolhida, e sim a que atenda melhor os critérios de cada variação do $GPCD$.

Algoritmo 12: AGM baseado no NSGAII

Entrada : τ, g^*, u_n, variação do método
Saída : P'_t

Function `AGM`(τ, g^*, u_n, *variação do método*)**:**
  $i = 0$
  $P_t = \tau$ /* gere a população */
  $R = \mathcal{F}_{P_t}$ /* avalie a população pelo *fitness* */
  enquanto $i \neq g^*$ **faça:**
    enquanto $Pt + 1 \leq u_n$ **faça:**
      $I_k, I_j = AleSel(P_t)$ /* selecione 2 indivíduos na população */
      $var = Ale(cruz, mut)$ /* selecione a função para gerar um novo indivíduo */
      se $var ==$ *"cruz"* **então:**
        $P_t^* = \text{cruz}(I_k, I_j)$
      fim
      senão se $var ==$ *"mut"* **então:**
        $P_t^* = \text{mut}\ (I_k, I_j)$
      fim
      $P_{t+1} = P_{t+1} + P_t^*$
    fim
    $R' = \mathcal{F}_{P_{t+1}}$ /* avalie a nova população pelo *fitness* */
    $Q = P_t \cup P_{t+1}$ /* agrupe pais e filhos */
    $P_t = Sel(Q, (Rk', R))$ /* seleciones os indivíduos pelo NSGAII */
    se $GPCDa$ **então:**
      $\Gamma = GPCDa\ (P_t)$ /* valie a acurácia global por combinação */
    senão se $GPCDd$ **então:**
      $\Gamma = GPCDd\ (P_t)$ /* avalie a distância entre os indivíduos (Equação 4.5) */
    fim
    se $i = 0$ **então:**
      $P'_t = P(t + 1)$
      $\texttt{temp} = \Gamma$
    senão se $\texttt{temp} < \Gamma$ **então:**
      $P'_t = P_t$
      $\texttt{temp} = \Gamma$
    fim
    $i = i + 1$
  fim
retorne P'_t

4.3 Exemplo prático do método

Esta seção apresentada um exemplo prático do método. Foram executadas todas as etapas, passo a passo, com um protocolo experimental reduzido a fim de exemplificar numérica e graficamente cada um dos passos.

Base de dados - A base de dados utilizada nesse exemplo foi a Banana, representada na Figura 19, onde os pontos de cores diferentes representam as classes, os eixos `x` e `y` denotam os valores dos atributos do problema. Essa base é simples de ser representada graficamente, pois ela possui apenas 2 classes, 2 atributos e 2000 instâncias. Mais informações desse problema de classificação estão presentes no Capítulo 5.

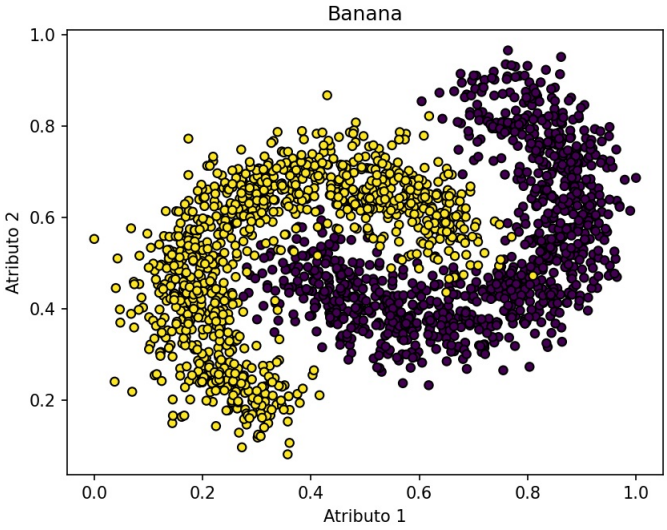


Figura 19 – Representação do *dataset* Banana. Pontos representam as diferentes classes do problema

Fonte: O autor.

Protocolo experimental - Foi realizado um protocolo experimental inspirado nos experimentos presentes no Capitulo 5. Entretanto, reduzido para uma mais fácil compreensão. O protocolo é composto por uma técnica de *holdout*, com 5 replicações, onde as instâncias do problema foram divididas em treino (50%), validação (25%) e teste (25%). O classificador de base utilizado foi o Perceptron com valor máximo de épocas igual a 100. Os demais atributos são os *defauls* do *scikit-learn 0.24.2* (PEDREGOSA et al., 2011).

A partir das informações apresentadas é possível dar sequencia na execução do método. Em resumo, a primeira etapa se subdivide nos seguintes passos:

- Geração dos subproblemas.
- Extração das medidas de complexidade.
- Normalização das medidas de complexidade.

- Cálculo da dispersão.
- Atribuir os pesos às medidas.

Geração dos subproblemas - No primeiro passo do método foram gerados 10 subproblemas com 50% das instâncias do conjunto de treinamento, isto é, 25% do problema original o que totaliza 500 exemplos para cada subproblema. A Figura 20 demonstra graficamente os 10 subproblemas. Pode-se observar pequenas diferenças entres as amostras, porém suficientes para um novo hiperplano do Perceptron.

Extração das medidas de complexidade - O segundo passo da primeira etapa do método é encontrar as complexidades de cada subproblema. As famílias de medidas utilizadas foram as de sobreposição e vizinhança. Os resultados estão presentes nas Tabelas 8 e 9, acompanhados dos valores máximos e mínimos de cada medida. Não foi utilizado arredondamento nessa etapa, pois os resultados, dentro de uma mesma medida, são muito próximos.

Tabela 8 – Valores das medidas de sobreposição para os 10 subproblemas do *dataset* Banana.

Sobreposição					
Medida / **Subprob.**	**F1**	**F1v**	**F2**	**F3**	**F4**
Banana 1	0.745388	0.170755	0.354543	0.498	0.448
Banana 2	0.750617	0.162203	0.367486	0.578	0.532
Banana 3	0.735824	0.161026	0.350168	0.528	0.470
Banana 4	0.743283	0.166007	0.331029	0.522	0.468
Banana 5	0.701954	0.130664	0.393988	0.508	0.472
Banana 6	0.732910	0.153400	0.389434	0.564	0.518
Banana 7	0.719842	0.145514	0.340163	0.462	0.414
Banana 8	0.752527	0.170897	0.377868	0.518	0.478
Banana 9	0.739116	0.150741	0.356420	0.494	0.462
Banana 10	0.747790	0.157010	0.385888	0.538	0.516
Máximo	**0.752527**	**0.170897**	**0.393988**	**0.578**	**0.532**
Mínimo	**0.701954**	**0.130664**	**0.331029**	**0.462**	**0.414**

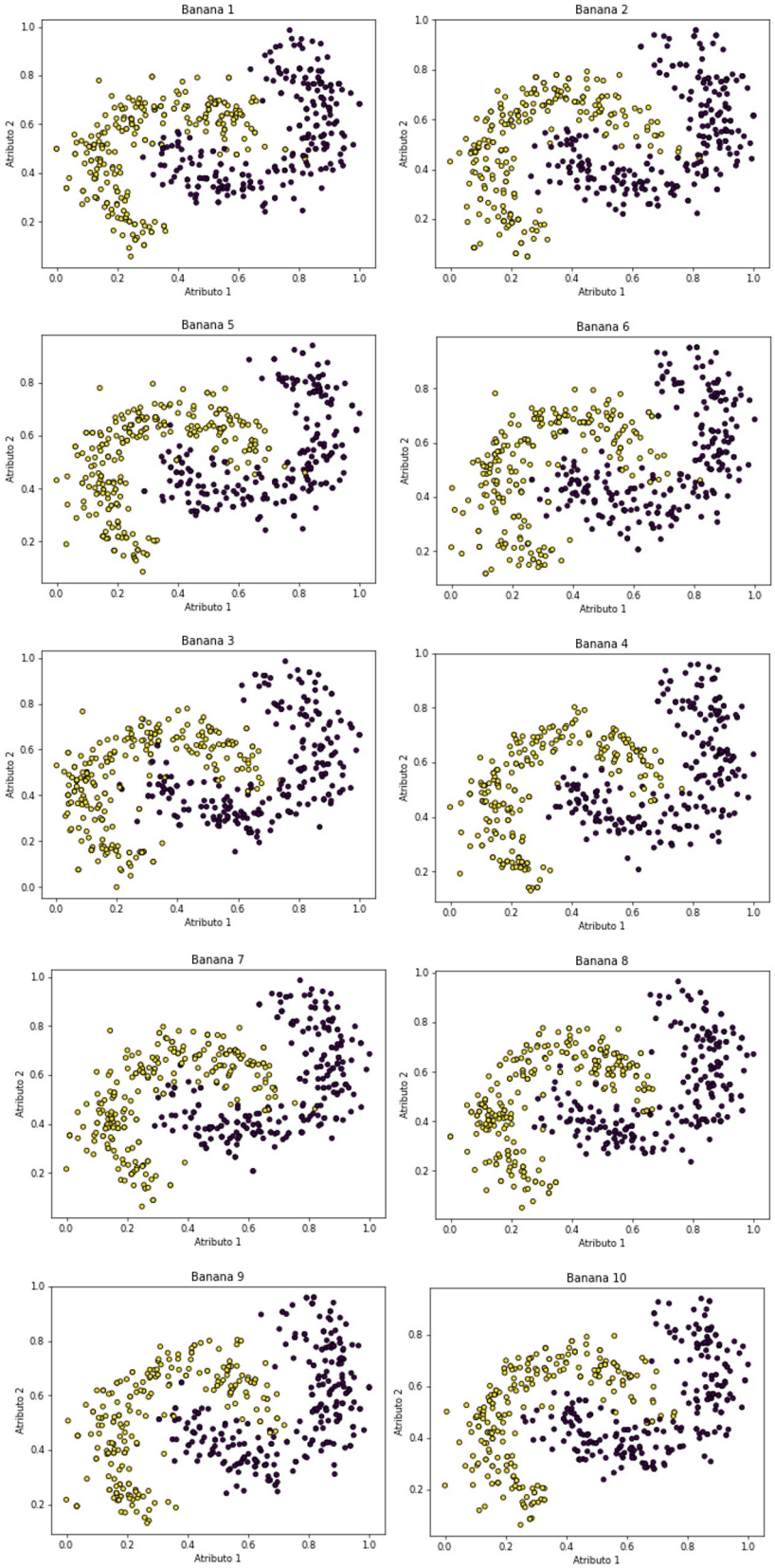


Figura 20 – Representação dos subproblemas originados do conjunto de treinamento do problema Banana.

Fonte: O autor.

Tabela 9 – Valores das medidas de vizinhança para os 10 subproblemas do *dataset* Banana.

	Vizinhança					
Medida / Subprob.	**N1**	**N2**	**N3**	**N4**	**T1**	**LSC**
Banana 1	0.042	0.087924	0.022	0.152	0.037037	0.901564
Banana 2	0.062	0.093354	0.020	0.14	0.026315	0.908892
Banana 3	0.056	0.083477	0.020	0.136	0.035714	0.910688
Banana 4	0.076	0.091016	0.024	0.146	0.028571	0.915856
Banana 5	0.046	0.096244	0.022	0.086	0.028571	0.904884
Banana 6	0.050	0.103359	0.020	0.142	0.034482	0.913864
Banana 7	0.044	0.091206	0.022	0.128	0.033333	0.904360
Banana 8	0.060	0.085617	0.018	0.152	0.038461	0.906624
Banana 9	0.054	0.092651	0.012	0.144	0.037037	0.915140
Banana 10	0.052	0.093536	0.03	0.128	0.034482	0.908016
Máximo	**0.076**	**0.103359**	**0.030**	**0.152**	**0.038461**	**0.915856**
Mínimo	**0.042**	**0.083477**	**0.012**	**0.086**	**0.026315**	**0.901564**

Normalização das medidas de complexidade - Neste passo as medidas de complexidade são normalizadas utilizando a Equação 4.1. As Tabelas 10 e 11 apresentam os resultados normalizados e com arredondamento das medidas, das duas famílias, para todos os 10 subproblemas.

Tabela 10 – Valores das medidas de sobreposição normalizados para 10 subproblemas do *dataset* Banana.

	Sobreposição				
Medida / Subprob.	**F1**	**F1v**	**F2**	**F3**	**F4**
Banana 1	0,86	1,00	0,38	0,32	0,29
Banana 2	0,97	0,79	0,58	1,00	1,00
Banana 3	0,67	0,76	0,31	0,57	0,48
Banana 4	0,82	0,88	0,00	0,52	0,46
Banana 5	0,00	0,00	1,00	0,40	0,50
Banana 6	0,62	0,57	0,93	0,88	0,89
Banana 7	0,36	0,37	0,15	0,00	0,00
Banana 8	1,00	1,00	0,75	0,49	0,55
Banana 9	0,74	0,50	0,41	0,28	0,41
Banana 10	0,91	0,66	0,88	0,66	0,87

Tabela 11 – Valores das medidas de vizinhança normalizados para 10 subproblemas do *dataset* Banana.

	Vizinhança					
Medida / Subprob.	**N1**	**N2**	**N3**	**N4**	**T1**	**LSC**
Banana 1	0,00	0,23	0,56	1,00	0,89	0,00
Banana 2	0,59	0,50	0,45	0,82	0,00	0,52
Banana 3	0,42	0,00	0,45	0,76	0,78	0,64
Banana 4	1,00	0,38	0,67	0,91	0,19	1,00
Banana 5	0,12	0,65	0,56	0,00	0,19	0,24
Banana 6	0,24	1,00	0,45	0,85	0,68	0,87
Banana 7	0,06	0,39	0,56	0,64	0,58	0,20
Banana 8	0,53	0,11	0,34	1,00	1,00	0,36
Banana 9	0,36	0,47	0,00	0,88	0,89	0,95
Banana 10	0,30	0,51	1,00	0,64	0,68	0,46

Cálculo da dispersão - Com os resultados das Tabelas 10 e 11 é possível calcular a dispersão (Equação 4.3) de cada medida, isto é, o quanto ela varia do valor médio.

Peso das medidas - Junto ao cálculo da dispersão, é atrelado um peso as medidas que possuírem o maior valor encontrado. Por fim, as Tabelas 12 e 13 apresentam a dispersão de cada medida e o voto dado as que obtiveram o maior valor para uma das 5 repetições, isto é, cada repetição terá uma nova tabela. Todo esses passos foram repetidos 5 vezes totalizando 50 subproblemas e 5 votos. Como resultado final (Tabela 14) as medidas F3 e T1 irão participar da segunda etapa do método.

Tabela 12 – Dispersão e peso atribuído as medidas de complexidade da família sobreposição.

Sobreposição					
	F1	**F1v**	**F2**	**F3**	**F4**
Dispersão	0,309345	0,309014	**0,343849**	0,291501	0,302847
Peso	0	0	1	0	0

Tabela 13 – Dispersão e peso atribuído as medidas de complexidade da família vizinhança.

Vizinhança						
	N1	**N2**	**N3**	**N4**	**T1**	**LSC**
Dispersão	0,296303	0,282300	0,252243	0,292346	**0,344570**	0,338697
Peso	0	0	0	0	**1**	0

Tabela 14 – Resultado final dos pesos, as medidas que F3 e T1 obtiveram mais votos. Logo essas participaram da segunda etapa.

Medidas	**F1**	**F1v**	**F2**	**F3**	**F4**	**N1**	**N2**	**N3**	**N4**	**T1**	**LSC**
Peso Final	1	0	1	**2**	1	0	0	1	1	**2**	1

A segunda etapa do método é composta, resumidamente, pelos seguintes passos:

- Escolha da variação do método ($GPCDa$, $GPCDd$).
- Geração dos subproblemas.
- Execução do AGM.
- Analise das gerações de acordo com a variação escolhida.
- *Pool* final.

Escolha da variação do método - Esta é uma etapa voltada à quem irá aplicar o método. A escolha da variação não influencia no processo de geração dos subproblemas utilizados para treinar o classificador, mas sim na escolha da melhor geração do AGM. A escolha específica de uma geração é justificada na Seção 4.2.4.

Geração dos subproblemas - Esse passo é semelhante ao da primeira etapa. Quando os subproblemas já foram gerados anteriormente, são utilizados nesse passo. Além de manter o padrão na construção das subamostras, diminui-se o tempo de processamento.

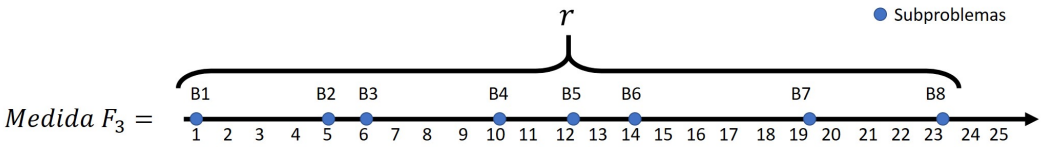


Figura 21 – Representação de um objetivo da função de *fitness*. Os círculos azuis representam os subproblemas e os valores da reta são os valores de complexidade de cada um deles.

Com isso a Figura 20, que exemplifica as subamostras geradas, pode ser utilizada para representar esse passo.

Execução do AGM - O AGM é responsável pela criação das subamostras que serão utilizadas para treinar o classificadores de base e formar o *pool*. Na Figura 21, observa-se um exemplo de um dos objetivos da função de *fitness*. A linha principal representa os valores (simbólicos) obtidos pela medida F3 para cada subproblema. Os círculos azuis denotam os subproblemas, que na figura, são representados pelas letras $B1$ até $B8$ (Banana1, Banana2, ..., Banana8) e finalmente, o símbolo r é o número total de subproblemas.

A Equação 4.6 apresenta como foi calculado um dos objetivos do *fitness* da subamostra $B1$ do exemplo referente a Figura 21. Na Equação 4.7 os valores correspondentes as complexidades de cada subproblema foram substituídos, e foi obtido o valor de $10,25$ unidades de medida.

$$Dist_{F3}B1 = \frac{Abs(B1_{F3} - B2_{F3}) + Abs(B1_{F3} - B3_{F3}) + ... + Abs(B1_{F3} - B8_{F3})}{r} \tag{4.6}$$

substituindo os valores, tem-se:

$$Dist_{F3}B1 = \frac{Abs(1-5) + Abs(1-6) + ... + Abs(1-23)}{8} \tag{4.7}$$

Esse procedimento é realizado para todas as subamostras. O segundo objetivo consiste do mesmo cálculo, no entanto para a medida da família de Vizinhança, que nesse exemplo é a T1. O último objetivo refere-se a Diversidade média par a par de cada subproblema. Utilizando a Figura 21 como exemplo, o terceiro objetivo é calculado de acordo com a Equação 4.8.

$$Diver_{[C_{B1}]}B1 = \frac{MD(C_{B1}C_{B2}) + MD(C_{B1}C_{B3}) + ... + MD(C_{B1}C_{B7}) + MD(C_{B1}C_{B8})}{r} \tag{4.8}$$

em que, MD é a medida de diversidade (Capítulo 2), C é o classificador treinado com as subamostras e r é o total de subproblemas.

Para representar as gerações do AGM, foram elaborados os gráficos da Figura 22 com base no resultado da função de *fitness*. Neste exemplo prático foram utilizadas 100

subamostras e apenas 5 gerações. Na Figura 22 os eixos `x` e `z` representam a distancia média entre os indivíduos nas famílias de sobreposição e vizinhança, já o eixo `y` representa a diversidade. A variação do método aplicada foi a $GPCDd$, ou seja, a geração mais dispersa nesse espaço tridimensional é a escolhida. As áreas dentro das marcações em vermelho nas figuras, representam a evolução e o distanciamento entre os subproblemas. Nesse caso, a geração 5 (G_5) foi escolhida para compor o *pool*, pois ela possui a maior distância média entre os indivíduos (valores encontrados estão descritos na legenda).

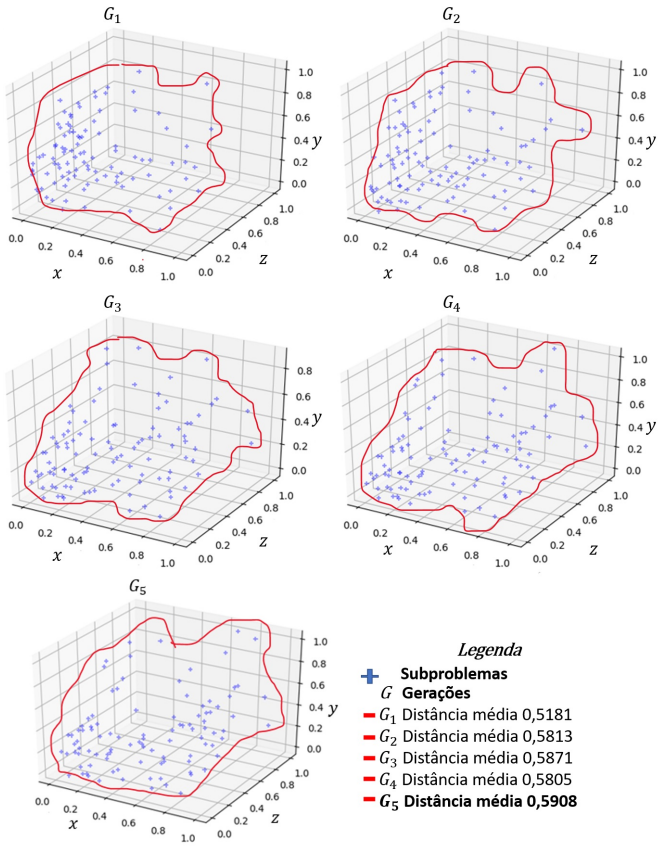


Figura 22 – Evolução dos subproblemas durante o AGM. Em destaque a uma representação em 2 dimensões da dispersão entre os subproblemas.

Fonte: O autor.

Pool final - O conjunto final de classificadores será formado de acordo com os subproblemas da geração escolhida. A diferença entre o $GPCDd$ e o $GPCDa$ está na escolha da geração com maior acurácia por combinação em relação a uma base de validação.

4.4 Considerações finais

A geração de subconjuntos para o treinamento de classificadores é fundamental para um bom desempenho dos métodos SDC e SDE. Neste capítulo foi apresentado um

novo método de geração de *pool* de classificadores baseado na complexidade dos dados e na diversidade a nível de classificação.

O método foi dividido em duas etapas, a primeira tem como objetivo selecionar uma ou mais medidas de complexidade que melhor se adaptam à proposta. Para isso, foi desenvolvido um esquema de pesos a partir das medidas que mais oscilam em valores. Na segunda etapa, foi desenvolvido um método que orienta a criação das subamostras de treinamento de forma ponderada a partir das medidas selecionadas na primeira etapa.

O próximo capítulo apresenta os experimentos pelos quais o método $GPCD$ e suas variações foram analisados. Pelos experimentos, são apresentadas algumas vantagens e limitações para cada uma das variações. O método também é comparado com outros métodos com o mesmo propósito encontrados na literatura.

5 Resultados Experimentais

O encadeamento deste capítulo apresenta o protocolo experimental e os resultados obtidos no decorrer da realização da pesquisa. Para tanto, as duas versões do $GPCD$ foram comparadas com outros quatro métodos de geração de conjunto de classificadores da literatura, sendo eles: Bagging, AdaBoost, Random Forest e DSOC.

Ao todo foram utilizadas 28 bases de dados nos experimentos. Os métodos de seleção dinâmica foram aplicados sobre as duas variações do $GPCD$ e os demais *pools* citados acima. Os resultados encontrados nos métodos de seleção dinâmica ou na combinação de classificadores variam de acordo com a diversidade encontrada entre os modelos de classificação, isto é, quanto menor a semelhança entre os classificadores, aumentam-se as chances de sucesso. Os métodos de SD utilizados para avaliar os conjuntos de classificadores foram: LCA, OLA e RANK KNORA-E KNORA-U, META-DES. O método de fusão de classificadores utilizado foi o Voto Majoritário (VM). Todos os classificadores treinados nos *pools* foram utilizados nos experimentos.

Visando avaliar o $GPCD$, foram executadas 20 replicações do mesmo e dos demais métodos de GPs já citados. Os resultados encontrados foram comparados com os testes estatísticos de *Friedman* e *Nemeniy*, cujo objetivo é detectar diferenças significativas entre os experimentos.

A apresentação dos resultados dos experimentos estão descritos em três seções sendo a primeira, a análise das medidas de complexidade (Seção 5.2). A segunda seção discorre sobre os resultados do conjunto de classificadores gerado pelo algoritmo evolutivo (Seção 5.3). E a terceira seção, a comparação do *pool* gerado pelo método proposto e os outros métodos de GPs da literatura (Seção 5.4).

5.1 Bases de Dados

O método $GPCD$ foi avaliado em 28 problemas de classificação da literatura disponíveis nos seguintes repositórios: *Machine Learning Repository* (UCI) (DUA; GRAFF, 2019), *Knowledge Extraction based on Evolutionary Learning Repository* (KEEL) (ALCALÁ-FDEZ et al., 2011), *Artificial datasets generated with the Matlab PRTools toolbox* (PRTools) (DUIN et al., 2004), STATLOG *project* (KING; FENG; SUTHERLAND, 1995), *Ludmila Kuncheva Collection of Real Medical Data* (LKC) (KUNCHEVA, 2004), *The enhanced learning for evolutive neural architectures project* (ELENA) (JUTTEN, 2002) e (VALENTINI, 2005). Essas bases de dados têm sido frequentemente utilizadas na literatura para mensurar o desempenho de métodos de Geração de *pool* e Seleção dinâmica de classificadores (SD) como por exemplo,os trabalhos de Souza et al. (2018), Brun et al. (2018), Cruz et al. (2015) e Pereira et al. (2018).

A Tabela 15 apresenta as características de cada problema utilizado nos experimentos. As bases de classificação Banana, Lithuanian e P2 são sintéticas, logo o número de instâncias pode ser definido no momento da criação da base. O tamanho da base, nesses casos, foi definido como 2000 (número de instâncias). Esse valor é o mesmo utilizado nos trabalhos de Brun et al. (2016), Brun et al. (2018) e Cruz et al. (2015), no caso dos *datasets* Banana e Lithuanian foi utilizada a ferramenta *PrTools* (DUIN et al., 2004) para criação dos mesmos, já para geração da base P2 foi utilizada a biblioteca *Deslib* (CRUZ et al., 2018, 2020). As demais bases de dados possuem os mesmos números de atributos, exemplos e número de classes, presentes nos seus respectivos repositórios.

Nenhum dos problemas de classificação, apresentados na Tabela 15, possuem um desbalanceamento expressivo entre as classes. Uma das condições para se encontrar a complexidade de um *dataset* é que o número de exemplos de cada classe seja maior ou igual a 2. Então, durante a geração ou evolução das subamostras, pode ocorrer a substituição de alguns exemplos de determinada classe, tornando o cálculo da complexidade impreciso.

Tabela 15 – Problemas utilizados nos experimentos

No.	**Problemas**	**Instâncias**	**Atributos**	**Classes**	Repositórios
01	Australian	690	14	2	UCI
02	Banana	2000	2	2	PRTools
03	Blood	748	4	2	UCI
04	CTG	2126	21	3	UCI
05	Diabetes	766	8	2	UCI
06	Faults	1941	27	7	UCI
07	German	1000	24	2	STATLOG
08	Haberman	306	3	2	UCI
09	Heart	270	10	2	STATLOG
10	ILPD	583	34	2	UCI
11	Ionosphere	351	16	2	UCI
12	Laryngeal1	213	16	2	LKC
13	Laryngeal3	353	2	3	LKC
14	Lithuanian	2000	2	2	PRTools
15	Liver	345	6	2	UCI
16	Mammo	830	5	2	KEEL
17	Monk	432	6	2	KEEL
18	Phoneme	5404	5	2	ELENA
19	P2	2000	2	2	(VALENTINI, 2005)
20	Segmentation	2310	19	7	UCI
21	Sonar	208	60	2	UCI
22	Thyroid	692	16	2	LKC
23	Vehicle	846	18	4	STATLOG
24	Vertebral	300	6	2	UCI
25	WBC	569	30	2	UCI
26	WDVG	5000	21	3	UCI
27	Weaning	302	17	2	LKC
28	Wine	178	13	3	UCI

Os problemas analisados foram divididos em subamostras de treino, validação e teste. Esta técnica de avaliação é conhecida como *holdout* e é amplamente utilizada na literatura. O subconjunto de treinamento contém 50% dos exemplos do problema, a validação e o teste, possuem 25% do restante da base de dados. Essas divisões foram realizadas aleatoriamente com estratificação. A cada nova iteração, o processo de divisão é executado novamente, gerando bases de treino, validação e teste distintas.

5.2 Resultados encontrados na etapa de análise das medidas de complexidade

A partir de um conjunto de treinamento, foram realizadas 100 amostragens ($r = 100$) com uma proporção de 50% das instâncias. Esses conjuntos foram utilizados para analisar a dispersão dos níveis de dificuldade de cada métrica de complexidade. Esse procedimento foi repetido por dez vezes, e a cada repetição foram geradas 100 amostras totalmente novas. Assim sendo, foram criadas 1000 amostras para cada problema de classificação.

Para facilitar o entendimento dos resultados e demonstrá-los graficamente foi utilizado o problema Lithuanian (Figura 23). A escolha desta base de dados se dá pela sua simplicidade de representação, pois possui apenas duas classes e dois atributos.

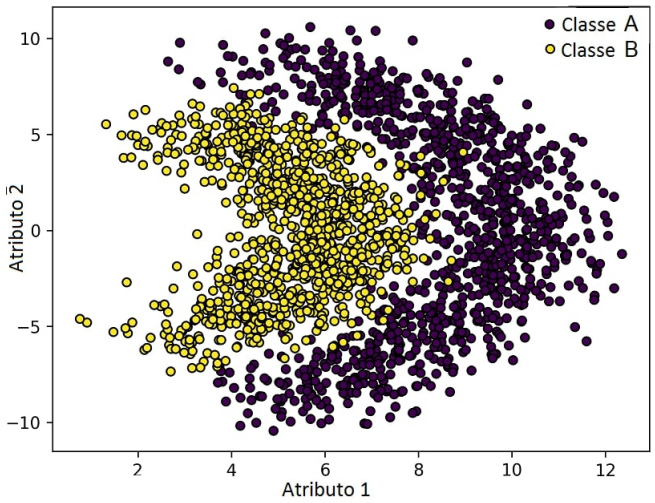


Figura 23 – Representação do *Dataset* Lithuanian.

Fonte: O autor.

Os subproblemas gerados foram mensurados pelas MCs dos grupos de sobreposição e vizinhança descritas no Capítulo 2. Essas medidas possuem uma complexidade em tempo de execução menor quando comparadas às medidas de dimensionalidade (LORENA et al., 2019). Como relatado anteriormente, os problemas propostos para o método não possuem um desbalanceamento entre classes significativo, por isso, foram descartadas as medidas de balanceamento. Finalmente, as métricas de linearidade não apresentaram uma dispersão muito elevada em testes anteriores, utilizando todos os problemas classificação.

A Tabela 16 mostra o resultado da dispersão do conjunto de dados Lithuanian, das grandezas dos grupos de sobreposição e de vizinhança, nas dez iterações da primeira etapa do método. As medidas selecionadas, nesse caso, foram F3 e T1. A Figura 24 mostra graficamente o comportamento das medidas ao longo das dez repetições da família de sobreposição, já a Figura 25 retrata o mesmo para as medidas de vizinhança.

Tabela 16 – Dispersão média do *Dataset* Lithuanian para as medidas de complexidade em 10 repetições.

Dispersão											
Família	**Sobreposição**					**Vizinhança**					
Medida / **Iteração**	**F1**	**F1v**	**F2**	**F3**	**F4**	**N1**	**N2**	**N3**	**N4**	**T1**	**LSC**
1	0,194	0,181	0,219	**0,234**	0,233	0,197	**0,208**	0,183	0,190	0,197	0,181
2	0,220	0,207	0,229	0,228	**0,236**	0,215	0,215	**0,242**	0,163	0,174	0,183
3	0,230	0,210	0,197	**0,238**	0,214	0,173	0,201	**0,215**	0,214	0,210	0,172
4	0,189	0,186	0,225	0,232	**0,237**	0,203	0,185	0,184	0,217	**0,221**	0,185
5	0,230	0,229	0,211	0,216	**0,248**	0,205	0,168	0,236	0,183	**0,245**	0,205
6	**0,243**	0,234	0,211	0,222	0,223	0,170	0,173	0,228	0,227	**0,246**	0,220
7	0,189	0,184	0,184	**0,206**	0,203	**0,217**	0,214	0,195	0,205	0,188	0,179
8	0,193	0,187	**0,204**	0,187	0,184	0,189	**0,220**	0,214	0,184	0,203	0,200
9	0,205	0,207	0,216	**0,238**	0,208	0,201	0,202	0,206	0,199	0,205	**0,211**
10	0,180	0,176	**0,225**	0,220	0,206	**0,221**	0,201	0,193	0,166	0,167	0,204
Votos	1	0	2	**4**	3	2	2	2	0	**3**	1

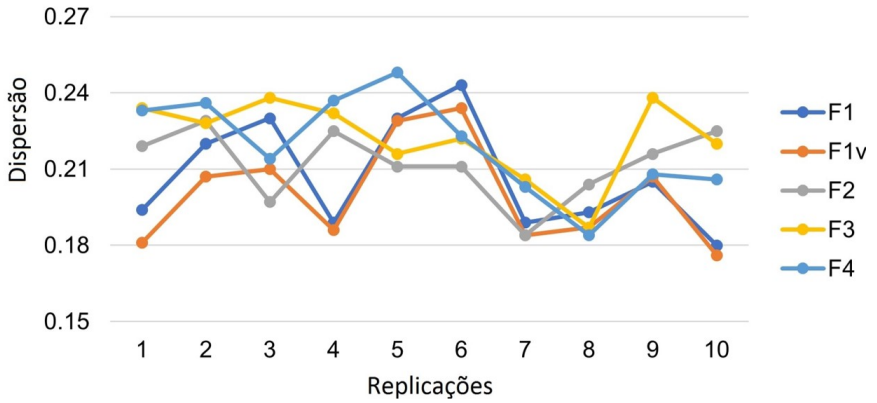


Figura 24 – Dispersão do problema Lithuanian considerando a família de medidas de sobreposição Sobreposição

Fonte: O autor.

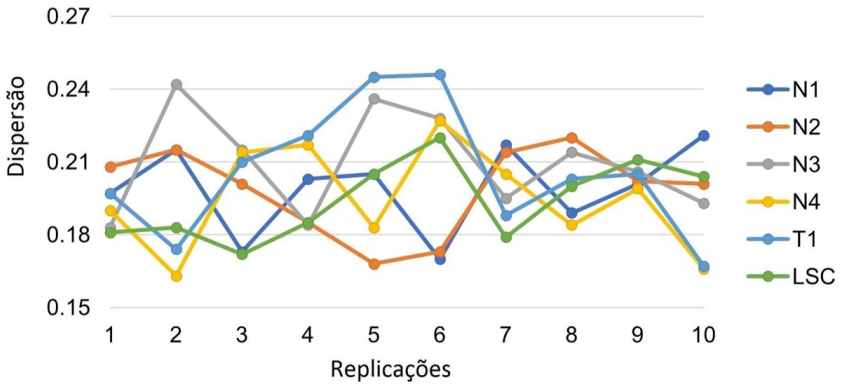


Figura 25 – Dispersão do problema Lithuanian considerando a família de medidas de Vizinhança

Fonte: O autor.

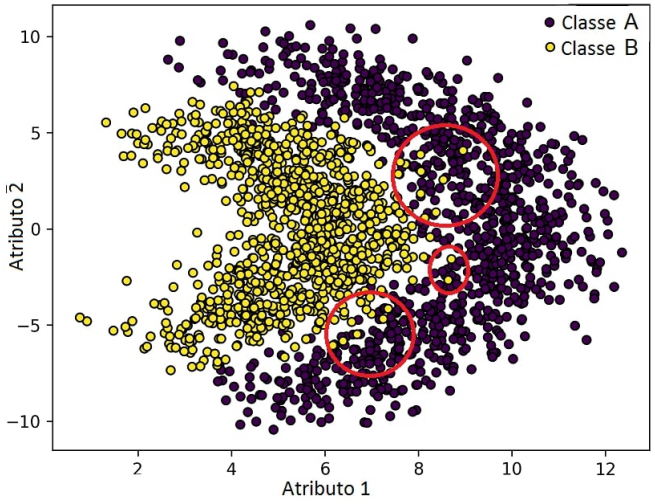


Figura 26 – Problema Lithuanian com destaque para sobreposição de atributos em vermelho.

Fonte: O autor.

Para o problema da Lithuanian, o valor de F3 (Seção 2.1.1) possui maior variação, pois os atributos desse *dataset*, dependendo das amostras, podem estar sobrepostos, conforme pode ser observado na Figura 26. Outrossim, a medida T1 pode assumir valores altos ou baixos de acordo com a instância em que se inicia o processo de cobertura pelas esferas conforme exposto na Seção 2.1.2. No entanto, o valor desta métrica está relacionado a quantidade de exemplos e a proximidade entre as classes, poucos exemplos favorecem a criação de esferas muito próximas, não alterando significativamente o resultado, da mesma maneira, classes muito distantes tendem a formar esferas que cobrem uma área grande e acabam sobrepondo as demais esferas.

Outro fator é a distribuição dos atributos das bases de dados, esses contribuem para a obtenção de mais variabilidade dessas medidas. Sendo assim, a Tabela 17 apresenta os resultados de todos *datasets* na primeira etapa do método. O peso atribuído às medidas foi realizado antes da geração do *pool* de classificadores e apenas uma vez, visto que são geradas 1000 subamostras, para atribuir o resultado do peso para cada medida.

Nota-se na Tabela 17 que não existe um padrão dos votos obtidos por cada base de dados para cada medida de complexidade. Na abordagem desta proposta, existe uma MC que melhor representa o problema de classificação de acordo com suas características. Essa análise confirma a Hipótese 1 onde os valores da métrica diferem de acordo com a distribuição das instâncias e atributos de cada *dataset*.

Tabela 17 – Tabela de votos de acordo com a dispersão de cada *dataset*, em cada métrica de complexidade

Problema	F1	F1v	F2	F3	F4	N1	N2	N3	N4	T1	LSC
Australian	**4**	3	1	2	0	0	0	2	**5**	1	2
Banana	0	2	3	**4**	1	1	0	3	0	**4**	2
Blood	2	0	**5**	3	0	**3**	1	1	2	1	2
CTG	**4**	3	2	1	0	1	**3**	2	1	2	1
Diabetes	1	1	2	**5**	1	1	3	1	0	1	**4**
Faults	0	**10**	0	0	0	1	1	**4**	1	1	2
German	0	0	**9**	1	0	3	1	0	**4**	1	1
Haberman	2	2	**4**	0	2	0	1	3	**4**	1	1
Heart	0	0	0	0	**10**	0	0	0	**6**	3	1
ILPD	1	3	0	**5**	1	**4**	2	2	1	0	1
Ionosphere	**7**	2	0	1	0	0	1	3	**5**	1	0
Laryngeal1	1	0	0	1	**8**	0	2	2	**4**	0	2
Laryngeal3	0	3	0	2	**5**	2	**4**	2	1	0	1
Lithuanian	1	0	2	**4**	3	2	2	2	0	**3**	1
Liver	1	5	2	0	2	2	1	**3**	2	2	0
Mammo	1	2	**7**	0	0	2	1	1	**3**	2	1
Monk	0	0	4	0	**6**	1	**4**	1	2	2	0
P2	2	**5**	3	0	0	1	2	**4**	2	0	1
Phoneme	1	**4**	1	2	2	1	3	**4**	1	0	1
Segmentation	0	**10**	0	0	0	0	2	**4**	1	2	1
Sonar	**5**	2	0	1	2	2	0	2	**5**	1	0
Thyroid	2	0	0	**5**	3	0	2	2	1	0	**5**
Vehicle	0	0	0	1	**9**	1	0	3	0	**4**	2
Vertebral	1	2	1	0	**6**	2	**4**	2	2	0	0
WBC	3	2	0	**5**	0	1	2	0	**3**	2	2
WDVG	**5**	0	0	2	3	3	1	2	0	**4**	0
Weaning	3	3	0	**4**	0	0	2	0	**6**	0	2
Wine	1	1	0	**8**	0	4	0	**5**	0	1	0
Média	1.7	2.3	1.6	2	2.3	1.4	1.6	2.1	2.2	1.4	1.3

5.3 Resultados encontrados na etapa de geração do *pool* de classificadores

Ao concluir a primeira etapa, onde foram escolhidas as medidas promissoras para cada problema de classificação, o método inicia a geração do *pool* de classificadores. Foi utilizado como base o algoritmo NSGAII para o AGM e a medida de diversidade escolhida foi a Falta Dupla (*Double Fault*) Df. O algoritmo NSGAII possui duas características importantes para o método proposto: (i) uma estratégia de preservação da elite, em outros termos, os melhores indivíduos seguem nas próximas gerações; (ii) a diversidade entre os membros da população por meio da *crowding distance*. Essas duas características do NSGAII mantêm uma boa distribuição dos indivíduos na frente de Pareto (SANTOS; SABOURIN; MAUPIN, 2009). O uso da medida de diversidade (MD) Falta Dupla é justificado de acordo com a pesquisa realizada pelos autores Ruta e Gabrys (2005), onde demonstraram bons resultados na construção GPs.

A Tabela 18 mostra a configuração AGM modificado para o $GPCD$ utilizado no método. Os valores de cruzamento e mutação foram decididos com base no trabalho dos autores Lacerda e Carvalho (1999), que recomendam uma taxa de cruzamento acima de 60%. O número de gerações foi definido empiricamente, partindo de 10 até 100, após 20 gerações o conjunto passou a perder diversidade, pois os atributos dos melhores indivíduos

foram replicados em vários subproblemas, tornando-os muito parecidos. Finalmente, o número de filhos também foi escolhido empiricamente, partido de 50 filhos até 500, não se notou melhora depois de 100 filhos, isto porque os problemas de classificação utilizados são relativamente pequenos em número de exemplos, logo era possível encontrar muitos filhos semelhantes.

Tabela 18 – Configuração do AGM utilizado no método

Configuração do AGM

Configuração	**Valor**
Probabilidade de cruzamento	90%
Probabilidade de mutação	10%
Número de gerações	20
Número de filhos por geração	100
Número de indivíduos selecionados	100

Os resultados para ambas as abordagens *GPCDa* e *GPCDd* foram obtidos com os classificadores de base Perceptron (ROSENBLATT, 1958) e Árvore de Decisão (BREIMAN et al., 1984). Ambos algoritmos foram utilizados com os parâmetros *defaults* das implementações da biblioteca *Scikit-learn* (PEDREGOSA et al., 2011). Para o Perceptron destacam-se os seguintes parâmetros: número máximo de interações igual a 1000 e aleatoriedade dos dados em cada época. Para o classificador de base Árvore de Decisão destacam-se os parâmetros: profundidade igual a máxima possível (descrição da biblioteca) e número mínimo de amostras por folha igual a 1.

O uso do classificador Perceptron justifica-se, pois quando utilizado em *pool* de classificadores, consegue representar problemas não linearmente separáveis (CRUZ; SABOURIN; CAVALCANTI, 2017). Já a Árvore de Decisão é considerada um classificador instável, de acordo com Skurichina e Duin (1998) classificadores construídos em conjuntos de subproblemas são tendenciosos e podem ter uma grande variação durante a classificação. Em outras palavras um classificador instável é sensível à mudança do conjunto de treinamento gerando classificadores diferentes para cada nova amostra.

A cada geração do *GPCDa* é executada a combinação pelo voto majoritário (VM). O VM consiste em combinar o voto de cada classificador do conjunto, de maneira a escolher a classe mais votada. O voto de cada especialista contabiliza um voto para o rótulo predito. Logo a classe que obtiver mais votos será atribuída ao exemplo de teste. Na variação do método *GPCDd*, a distância média entre os indivíduos é mensurada de acordo com a Equação 4.5.

Ao fim de cada variação do método, o algoritmo seleciona a geração com valor mais adequado, ou seja, no caso do *GPCDa* leva-se em conta a maior acurácia do conjunto dos classificadores pelo voto majoritário simples entre as gerações do AGM. No *GPCDd* utiliza-se a maior distância média entre todos os indivíduos do *pool*.

Para tornar os experimentos mais robustos, a segunda etapa do método foi replicada por 20 vezes. A primeira geração do algoritmo $GPCD$ consiste em amostras de 50% do conjunto de dados de treinamento. O resultado do cruzamento e da mutação gera os novos indivíduos que serão avaliados de acordo com os objetivos propostos do $GPCD$. Ao completar a geração de número 20 o método reinicia com a próxima iteração.

Ao todo, foram executadas 1600 gerações para cada base de dados. De todas as 1600, 800 foram para as variações do $GPCDa$ e $GPCDd$, sendo 400 para cada classificador de base Perceptron e Árvore de decisão. A Figura 27 foi criada para exemplificar como foram realizados os experimentos da segunda etapa do método, o losango representa as alternativas do $GPCD$ e r^* o número de gerações.

Por fim, depois das vinte gerações o $GPCD$ retornou quatro conjuntos de classificadores diferentes, chamados de $GPCDa_{pe}$, $GPCDa_{ad}$, $GPCDd_{pe}$ e $GPCDd_{ad}$, nos quais, os símbolos pe e ad representam os classificadores de base Perceptron e Árvore de Decisão respectivamente.

Com o objetivo de demonstrar o espalhamento entre os subproblemas na execução do AGM, foram calculadas as médias das distâncias entre cada indivíduo da primeira geração e da melhor geração do $GPCD$. Com o resultado obtido para cada iteração, calculou-se a média final. Esse resultado é apresentado na Tabela 19, os valores em destaque representam as situações onde o $GPCD$ obteve um espalhamento menor que a primeira geração. Nos parenteses estão os desvios padrão, neste caso foi utilizado 3 casas decimais no arredondamento, devido ao baixo desvio padrão. Observa-se na tabela que o número de ocasiões onde a dispersão foi menor que a primeira geração é muito pequeno quando comparado ao número total de experimentos.

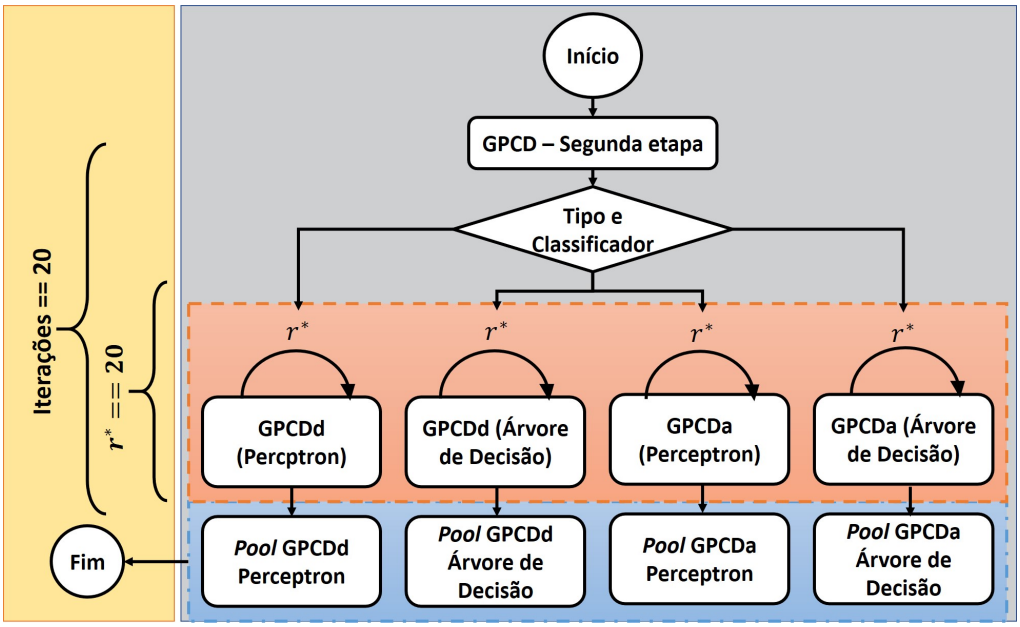


Figura 27 – Detalhes da execução dos experimentos da segunda etapa do GPCD

Fonte: O autor.

Tabela 19 – Comparativo das distâncias média entre os subproblemas de acordo com os valores do *fitness*. Na segunda coluna estão os subproblemas de primeira geração, nas demais colunas as variações dos $GPCD$. Em destaque estão as distâncias onde o $GPCD$ foi menor que os subproblemas de primeira geração.

Distância média entre os subproblemas					
		Perceptron		**Árvore de Decisão**	
Problema	**Primeira Geração**	*GPCDa*	*GPCDd*	*GPCDa*	*GPCDd*
Australian	0,429 (0,037)	0,463 (0,027)	0,43 (0,038)	0,475 (0,036)	0,494 (0,041)
Banana	0,452 (0,039)	0,453 (0,026)	0,454 (0,016)	0,497 (0,032)	0,494 (0,017)
Blood	0,538 (0,092)	0,665 (0,026)	0,6 (0,047)	**0,485 (0,028)**	0,545 (0,052)
CTG	0,444 (0,027)	0,531 (0,015)	0,497 (0,021)	**0,414 (0,017)**	0,503 (0,021)
Diabetes	0,397 (0,051)	0,509 (0,06)	0,559 (0,05)	0,398 (0,014)	0,464 (0,038)
Faults	0,45 (0,039)	0,463 (0,034)	0,51 (0,023)	0,512 (0,015)	0,518 (0,019)
German	0,439 (0,06)	0,46 (0,034)	0,554 (0,08)	0,483 (0,097)	0,53 (0,056)
Haberman	0,48 (0,049)	0,567 (0,031)	0,559 (0,039)	0,525 (0,022)	0,557 (0,025)
Heart	0,433 (0,041)	0,456 (0,031)	0,526 (0,074)	0,476 (0,047)	0,484 (0,039)
ILPD	0,449 (0,044)	0,544 (0,023)	0,513 (0,02)	0,475 (0,025)	0,541 (0,014)
Ionosphere	0,414 (0,035)	0,454 (0,023)	0,439 (0,027)	0,451 (0,019)	0,445 (0,035)
Laryngeal1	0,398 (0,032)	0,485 (0,038)	0,471 (0,039)	0,49 (0,046)	0,537 (0,053)
Laryngeal3	0,431 (0,034)	0,435 (0,02)	0,458 (0,021)	0,471 (0,033)	0,455 (0,02)
Lithuanian	0,506 (0,036)	**0,483 (0,037)**	0,508 (0,018)	0,529 (0,027)	0,542 (0,027)
Liver	0,462 (0,066)	0,586 (0,019)	0,561 (0,049)	0,515 (0,014)	0,561 (0,035)
Mammo	0,424 (0,044)	0,553 (0,02)	0,552 (0,04)	0,548 (0,023)	0,534 (0,029)
Monk	0,45 (0,033)	0,462 (0,027)	0,598 (0,033)	0,574 (0,028)	0,672 (0,02)
P2	0,42 (0,062)	0,46 (0,014)	0,545 (0,071)	0,477 (0,023)	0,482 (0,026)
Phoneme	0,446 (0,044)	0,502 (0,045)	0,545 (0,029)	0,533 (0,03)	0,536 (0,027)
Segmentation	0,434 (0,04)	0,463 (0,019)	0,501 (0,013)	0,492 (0,034)	0,54 (0,023)
Sonar	0,429 (0,066)	0,452 (0,043)	**0,425 (0,033)**	0,43 (0,031)	0,446 (0,053)
Thyroid	0,462 (0,049)	0,558 (0,052)	0,608 (0,032)	0,523 (0,023)	0,552 (0,022)
Vehicle	0,419 (0,039)	0,432 (0,053)	0,524 (0,026)	0,455 (0,017)	0,464 (0,046)
Vertebral	0,445 (0,047)	0,486 (0,035)	0,571 (0,022)	0,604 (0,042)	0,536 (0,017)
WBC	0,492 (0,044)	**0,488 (0,049)**	0,512 (0,04)	0,507 (0,039)	**0,476 (0,03)**
WDVG	0,454 (0,035)	0,516 (0,04)	0,537 (0,017)	0,469 (0,015)	0,589 (0,032)
Weaning	0,403 (0,036)	0,419 (0,042)	0,457 (0,025)	0,416 (0,036)	0,479 (0,043)
Wine	0,459 (0,048)	0,464 (0,02)	0,645 (0,048)	**0,417 (0,022)**	0,598 (0,059)

.

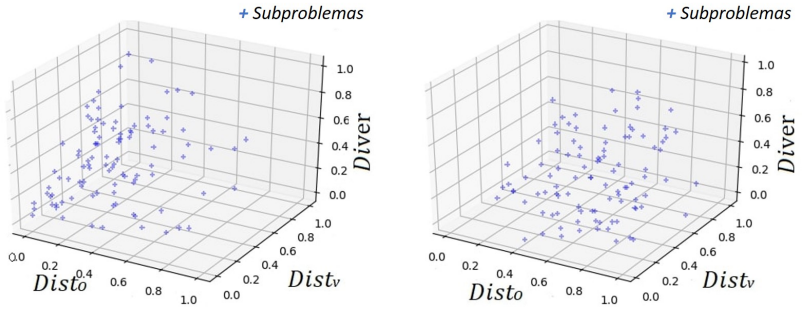
As melhores gerações, de acordo com os critérios desse trabalho, são apresentadas na Tabela 20. Nela são indicas as médias e o desvios padrão do índice da geração que possui a maior acurácia ($GPCDa$) e a que possui a maior dispersão ($GPCDd$), dentro das 20 iterações.

Nota-se que em $GPCDa_{pe}$, o *dataset* Banana atingiu sua maior dispersão dentro das vinte gerações, já na oitava geração em todas as iterações. O problema Banana teve variação significativa em sua complexidade, atingindo seu pico em poucas gerações, assim como o problema de classificação P2 que atingiu a sua acurácia máxima em apenas duas gerações.

Nas Figuras 28 e 29 pode-se observar a evolução dos indivíduos no processo de geração do *pool* realizada com os conjuntos de dados Lithuanian e P2, respectivamente. Os pontos dispostos nessas figuras representam os subconjuntos (indivíduos), o eixo `x` ($Dist_o$) significa a distância média par a par da medida de sobreposição, no eixo `z` a distância da grandeza de vizinhança ($Dist_v$), enquanto o eixo `y` ($Diver$) representa a diversidade (DF).

Tabela 20 – Resultados da média e o desvio padrão da geração em que o melhor desempenho foi alcançado.

Melhor geração (em média)				
Problema	$GPCDa_{pe}$	$GPCDa_{ad}$	$GPCDd_{pe}$	$GPCDd_{ad}$
Australian	7,6 (4,5)	12,5 (0,9)	10,5 (1,1)	14,6 (3,3)
Banana	2,9 (2,8)	6,6 (5,0)	8,0 (0,0)	7,8 (0,7)
Blood	3,5 (6,2)	9,0 (5,7)	6,5 (1,7)	11,8 (4,1)
CTG	19,1 (2,2)	10,1 (2,1)	9,2 (4,8)	11,0 (2,0)
Diabetes	11,4 (4,8)	17,5 (0,9)	12,5 (3,9)	12,0 (3,7)
Faults	6,3 (6,3)	10,6 (3,2)	9,5 (2,7)	13,6 (0,5)
German	9,9 (0,2)	16,1 (0,2)	10,6 (3,7)	12,8 (2,3)
Haberman	9,8 (1,7)	6,8 (4,1)	5,3 (0,7)	14,5 (2,2)
Heart	9,2 (2,5)	13,2 (4,9)	7,2 (3,2)	8,8 (1,6)
ILPD	13,9 (2,2)	12,8 (2,2)	13,4 (1,3)	17,4 (2,8)
Ionosphere	7,5 (2,2)	11,0 (3,9)	9,9 (3,3)	11,0 (3,9)
Laryngeal1	15,8 (2,2)	12,8 (1,3)	11,6 (4,5)	12,5 (5,0)
Laryngeal3	12,1 (1,8)	13,4 (6,7)	11,3 (3,5)	12,5 (2,3)
Lithuanian	12,8 (5,2)	15,0 (2,4)	7,5 (2,2)	11,9 (4,3)
Liver	18,6 (1,1)	6,5 (1,1)	11,4 (3,6)	11,1 (5,6)
Mammo	5,4 (4,3)	10,3 (3,1)	12,2 (1,9)	10,7 (4,0)
Monk	16,4 (0,8)	10,5 (1,5)	13,7 (0,7)	11,6 (4,4)
Phoneme	8,7 (4,4)	15,5 (5,0)	11,1 (5,2)	10,9 (3,8)
P2	2,0 (0,0)	11,8 (3,4)	9,0 (3,1)	10,9 (3,8)
Segmentation	17,1 (4,1)	7,1 (3,5)	7,0 (2,3)	12,0 (4,4)
Sonar	17,9 (3,3)	16,7 (2,2)	15,3 (4,9)	8,7 (3,1)
Thyroid	11,5 (7,0)	14,1 (0,9)	12,8 (0,9)	11,6 (1,1)
Vehicle	7,3 (4,7)	12,6 (4,3)	11,2 (3,2)	8,4 (4,1)
Vertebral	5,9 (4,2)	12,5 (1,7)	12,1 (3,1)	12,5 (1,7)
WBC	4,4 (0,5)	12,4 (1,5)	8,2 (2,6)	12,4 (1,5)
WDVG	9,2 (6,0)	11,3 (2,6)	11,6 (3,0)	11,3 (2,6)
Weaning	10,6 (1,6)	11,9 (1,6)	18,1 (4,8)	11,9 (1,6)
Wine	11,2 (0,9)	12,8 (1,9)	14,8 (5,6)	12,8 (1,9)
Média	10,3 (3)	11,9 (2,8)	10,7 (2,8)	11,9 (2,9)

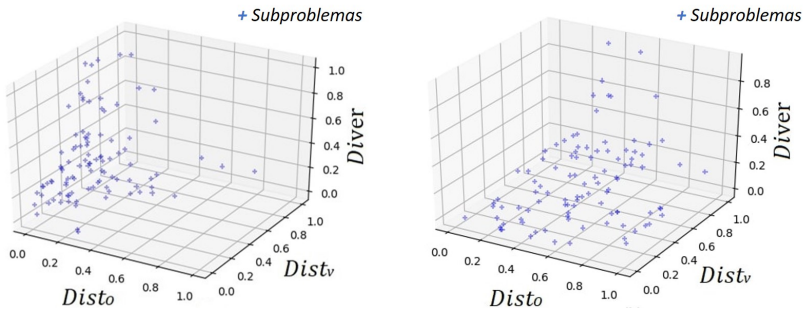


(a) Primeira geração, cuja distância média é de 0,5071

(b) Melhor geração, cuja distância média é de 0,553

Figura 28 – Representação da dispersão dos indivíduos da base de dados Lithuanian durante a execução do AGM. A Figura (a) demonstra a dispersão na primeira geração de indivíduos, a Figura (b) representa a melhor geração do $GPCD$.

Fonte: O autor.



(a) Primeira geração, cuja distância média é de 0,4976

(b) Melhor geração, cuja distância média é de 0,5935

Figura 29 – Representação da dispersão dos indivíduos da base de dados P2 durante a execução do AGM. A Figura (a) demonstra a dispersão na primeira geração de indivíduos, a Figura (b) representa a melhor geração do $GPCD$.

Fonte: O autor.

As Figuras 28a e 29a mostram a dispersão das mostras da primeira geração no espaço, quando os subconjuntos são gerados aleatoriamente, enquanto nas Figuras 28b e 29b uma visão geral dos indivíduos no espaço na geração escolhida pelo $GPCD$. A diferença entre elas é nítida, pois na geração escolhida (b) os subproblemas estão mais bem espalhados em relação ao observado nas figuras que representam a primeira geração (a), mostrando assim, que o método está distribuindo as amostras no espaço, na maioria das vezes como proposto nos objetivos dessa pesquisa.

5.4 Resultados das comparações com outros métodos da literatura

Primeiramente, os *pools* gerados pelo método $GPCD$ foram combinados pelo Voto Majoritário, e comparado com outros métodos de geração de conjuntos de classificadores da literatura. Os algoritmos *Bagging* e *AdaBoost* utilizam como combinação padrão o VM o que torna mais justa a comparação entre os métodos. Posteriormente, os métodos SDC e SDE foram aplicados sobre os conjuntos de especialistas. Todos os resultados encontrados foram comparados com o *Bagging, AdaBoost, Random Forest* (BREIMAN, 2001) e *DSOC* (BRUN et al., 2018).

5.4.1 Resultados da fusão de classificadores

As Tabelas 21 e 22 apresentam os resultados encontrados para a fusão dos classificadores. Os valores destacados nas tabelas são os melhores em termos de acurácia qunado comparados todos os metedos, os valores entre parênteses são os resultados dos desvios padrão. Sendo, a Tabela 21 para o classificador de base Perceptron e a Tabela 22 para a Árvore de Decisão.

Tabela 21 – Resultados da combinação de classificadores por Voto Majoritário usando Perceptron como o classificador base.

	Fusão por VM - Classificador Perceptron					
Problema	*GPCDa*	*GPCDd*	**Bagging**	**AdaBoost**	**R.F.**	**DSOC**
Australian	87,8 (2,2)	**88,1 (2,3)**	86,9 (2,1)	86,9 (2,2)	87,7 (2,0)	85,8(2,0)
Banana	84,7 (1,3)	83,5 (2,0)	84,6 (1,4)	83,9 (1,5)	**97,2 (0,8)**	83,5(1,9)
Blood	77,1 (3,3)	76,7 (1,9)	**77,6 (2,0)**	77,5 (1,3)	74,9 (2,8)	74,1(3,9)
CTG	88,9 (1,7)	84,7 (17,7)	81,1 (24,3)	81,1 (23,4)	**93,5 (1,0)**	80,8(24,2)
Diabetes	**76,9 (3,0)**	76,4 (2,7)	76,8 (2,8)	76,6 (2,7)	75,7 (3,3)	74,7(3,1)
Faults	70,7 (2,1)	70,5 (1,4)	69,4 (1,7)	69,5 (1,3)	**75,9 (1,6)**	67,6(2,2)
German	**77,0 (2,4)**	76,5 (3,0)	75,9 (2,3)	75,8 (2,6)	74,8 (2,5)	74,2(3,0)
Haberman	**76,8 (3,3)**	**76,8 (2,7)**	75,0 (2,3)	74,7 (2,0)	69,7 (3,7)	74,3(2,8)
Heart	**85,9 (3,9)**	85,4 (3,6)	82,5 (3,7)	81,5 (4,6)	80,0 (4,8)	82,3(2,7)
ILPD	70,6 (3,7)	**72,7 (3,3)**	71,4 (2,7)	71,2 (3,2)	70,4 (2,8)	69,4(3,1)
Ionosphere	89,9 (2,7)	91,4 (3,8)	89,2 (2,9)	87,0 (4,9)	**93,4 (2,9)**	88,5(3,4)
Laryngeal1	**86,0 (5,1)**	84,2 (3,7)	83,5 (3,7)	81,9 (4,8)	83,2 (3,9)	83,5(5,0)
Laryngeal3	**76,3 (3,2)**	74,8 (2,7)	72,7 (2,3)	68,7 (7,7)	72,3 (2,8)	71,9(3,4)
Lithuanian	81,6 (2,1)	82,9 (1,7)	82,8 (1,7)	82,4 (1,8)	**97,0 (0,6)**	82,0(1,8)
Liver	**70,5 (3,3)**	68,4 (3,8)	68,7 (4,2)	68,0 (3,9)	68,4 (4,0)	64,9(5,8)
Mammo	**84,0 (2,3)**	83,8 (2,5)	82,9 (2,1)	81,9 (2,4)	78,7 (2,0)	82,1(2,6)
Monk	85,7 (2,3)	83,4 (2,6)	78,9 (3,2)	78,0 (3,5)	**98,8 (1,6)**	80,0(3,7)
Phoneme	54,6 (4,1)	76,1 (1,4)	75,6 (1,1)	75,9 (0,7)	**89,1 (0,8)**	74,6(2,6)
P2	77,6 (1,1)	56,7 (4,4)	56,5 (4,6)	51,9 (3,9)	**93,7 (1,0)**	53,3(3,4)
Segmentation	90,8 (1,0)	91,0 (1,4)	91,0 (1,2)	91,5 (1,1)	**97,0 (1,0)**	90,4(1,0)
Sonar	**84,4 (6,2)**	81,7 (6,5)	77,7 (3,4)	72,3 (7,2)	78,1 (3,9)	75,9(3,5)
Thyroid	97,1 (1,0)	**97,3 (0,9)**	96,6 (1,2)	96,8 (1,3)	96,0 (0,8)	96,2(1,6)
Vehicle	75,5 (2,6)	**77,5 (2,4)**	75,2 (2,0)	75,4 (2,0)	74,5 (2,1)	73,6(2,4)
Vertebral	**89,1 (3,8)**	87,9 (3,6)	87,1 (3,4)	86,5 (3,3)	85,7 (3,2)	86,1(3,4)
WBC	98,0 (0,8)	**98,2 (1,0)**	97,1 (1,1)	95,1 (3,9)	95,5 (1,5)	96,9(1,0)
WDVG	**86,8 (1,0)**	**86,8 (0,9)**	86,5 (0,8)	86,7 (0,9)	85,0 (0,7)	85,6(0,9)
Weaning	85,3 (3,6)	84,9 (3,6)	82,5 (3,8)	81,8 (4,0)	**89,5 (2,8)**	81,7(3,8)
Wine	**98,4 (1,9)**	98,0 (2,3)	97,5 (2,3)	97,2 (2,6)	97,3 (2,8)	97,3(1,7)
Média	**82,4**	**82,0**	**80,8**	**79,9**	**84,8**	**79,7**

Tabela 22 – Resultados da combinação de classificadores por Voto Majoritário usando Árvore de Decisão como o classificador base.

	Fusão por VM - Árvore de Decisão					
Problema	*GPCDa*	*GPCDd*	**Bagging**	**AdaBoost**	**R.F.**	**DSOC**
Australian	**93,0(2,3)**	91,3(2,8)	87,6(1,6)	82,3(2,6)	87,7(2,0)	86,6(1,9)
Banana	**97,8(0,7)**	97,6(0,7)	97,0(0,8)	95,8(0,8)	97,2(0,8)	96,8(1,0)
Blood	**83,6(2,4)**	81,7(2,9)	77,4(3,2)	74,4(2,5)	74,9(2,7)	75,4(3,8)
CTG	**95,8(1,0)**	95,5(1,0)	93,1(1,1)	92,0(1,5)	93,5(1,0)	92,8(0,9)
Diabetes	**85,2(2,8)**	81,8(4,6)	75,8(2,8)	69,0(3,8)	75,7(3,3)	75,3(3,5)
Faults	83,6(5,0)	**84,6(2,9)**	75,4(1,7)	69,4(2,7)	75,9(1,6)	73,8(2,2)
German	**85,9(5,1)**	82,5(5,2)	74,9(2,7)	68,3(3,1)	74,8(2,5)	73,5(3,2)
Haberman	**80,5(6,1)**	79,5(5,2)	70,4(4,1)	71,4(4,5)	69,7(3,7)	71,6(4,2)
Heart	**90,1(4,1)**	88,4(5,2)	80,4(4,5)	73,5(5,1)	80,0(4,8)	79,3(6,0)
ILPD	81,6(5,6)	**81,7(4,8)**	69,9(3,2)	65,8(3,9)	70,4(2,8)	69,6(4,4)
Ionosphere	**96,2(1,5)**	95,9(1,9)	92,4(2,7)	87,4(4,5)	93,4(2,9)	91,8(3,0)
Laryngeal1	**91,5(4,6)**	88,4(5,1)	83,1(4,7)	77,8(4,7)	83,2(3,9)	82,0(5,1)
Laryngeal3	81,0(5,5)	**81,1(4,9)**	73,3(3,1)	63,2(4,6)	72,3(2,8)	71,6(2,9)
Lithuanian	**98,0(0,8)**	97,8(0,5)	97,0(0,6)	95,6(0,8)	97,0(0,6)	96,7(0,6)
Liver	**85,1(3,2)**	79,7(7,5)	69,4(3,7)	60,2(4,1)	68,4(4,0)	66,9(3,4)
Mammo	**85,4(4,1)**	84,4(3,9)	81,5(1,8)	55,9(4,5)	78,7(2,0)	79,7(1,8)
Monk	**100,0(0,0)**	99,9(0,4)	**100,0(0,2)**	**100,0(0,0)**	98,8(1,6)	99,6(1,3)
Phoneme	**96,8(1,2)**	95,5(1,2)	93,1(1,0)	91,7(1,7)	93,7(1,0)	87,1(0,9)
P2	91,9(2,6)	91,7(2,2)	88,0(0,9)	84,8(1,1)	89,1(0,8)	**92,4(1,3)**
Segmentation	**98,1(0,9)**	97,9(0,8)	96,3(0,9)	95,3(1,1)	97,0(1,0)	95,8(0,9)
Sonar	84,7(5,5)	**85,5(6,6)**	74,3(5,4)	69,4(6,8)	78,1(3,9)	72,6(6,1)
Thyroid	97,7(1,2)	**97,8(0,9)**	95,9(1,0)	95,1(0,9)	96,0(0,8)	95,7(1,0)
Vehicle	**86,1(3,4)**	85,2(5,1)	72,9(1,8)	68,3(2,5)	74,5(2,1)	72,3(3,0)
Vertebral	92,1(4,1)	**92,5(4,0)**	85,7(3,3)	83,4(3,3)	85,7(3,2)	84,1(3,7)
WBC	**97,0(1,7)**	96,6(1,9)	94,7(1,8)	92,3(1,9)	95,5(1,5)	94,4(2,0)
WDVG	**90,5(2,5)**	89,5(3,0)	84,3(0,9)	74,6(1,1)	85,0(0,7)	83,1(0,9)
Weaning	**93,7(3,4)**	90,1(3,0)	85,7(4,0)	77,1(5,5)	89,5(2,8)	83,6(3,9)
Wine	**99,3(1,0)**	96,4(2,2)	95,2(4,4)	89,4(5,9)	97,3(2,8)	94,2(4,2)
Média	**90,8**	**89,6**	**84,4**	**79,4**	**84,7**	**83,5**

Os valores observados nas Tabelas 21 e 22 demonstraram que a *GPCDa* obteve o melhor resultado geral na combinação pela regra do Voto Majoritário com o classificador de base Perceptron e Árvore de Decisão. Com o classificador Perceptron (Tabela 21) o *GPCDa* obteve 12 vitórias (42,8%), o segundo lugar ficou com o algoritmo *Random Forest* com 10 vitórias (35,7%). Com a Árvore de Decisão o *GPCDa* obteve um total de 21 vitórias, ou seja, 75,0% (Tabela 22). Importante destacar que todos os empates são contabilizados como vitória.

Os resultados encontrados neste experimento são justificáveis devido à escolha da geração, com maior acurácia global. Esta estratégia favorece o modelo de combinação por Voto Majoritário. Entretanto, o *GPCDd* obteve sete vitórias à frente dos métodos *Bagging*, *AdaBoost* e *DSOC* com o Perceptron como indutor e seis vitórias com Árvore de Decisão.

Com o Perceptron os métodos de geração de *pool Bagging, AdaBoost, DSOC* e *GPCD* sofrem desvantagens contra o algoritmo RF, pois ele sempre utiliza Árvore de Decisão como classificador de base, logo os resultados tendem a ser melhores por se tratar de um classificador mais instável.

5.4.2 Resultados com seleção dinâmica de classificadores

Assim como na combinação de classificadores, foram utilizados os métodos de seleção dinâmica sobre os *pools* gerados, o objetivo é comparar os resultados obtidos nesses métodos utilizando diferentes técnicas de geração de classificadores. Os métodos de SDC utilizados foram LCA, OLA, Rank, KNORA-E, KNORA-U e META-DES. O mesmo protocolo descrito na seção anterior foi utilizado nesses experimentos. Os resultados médios encontrados para o classificador Perceptron são apresentados na Tabela 23. O valores em destaque são as melhores médias, nos parênteses estão os desvios padrão. Por fim, os asteriscos demostram onde não houve diferença estatística significativa.

A técnica de SDC LCA, quando aplicada sobre os *pools* $GPCDa_{pe}$ e $GPCDd_{pe}$ (classificador de base Perceptron) proporcionou um número de vitórias em 32% dos *datasets*, ou seja, cada variação do $GPCD$ foi melhor em 10 problemas de classificação diferentes, isto quando comparados a outros métodos de GPs.

Para o OLA, o $GPCDa_{pe}$ obteve um ganho em 39,3% o que equivale a 11 dos 28 *datasets* testados e o $GPCDd_{pe}$ ganhou em 28,6% (8). Finalmente, no método de seleção Rank, o *pool* $GPCDa_{pe}$ obteve 35,7% (10 bases de dados) das vitórias quando comparados a outros métodos GPs. Já o $GPCDd_{pe}$ obteve 28,6% em vitórias. O método proposto foi o mais preciso no que tange valores absolutos em todos os casos quando comparado ao *Bagging, AdaBoost, DSOC* e melhor ou igual ao *Random Forest*.

A Tabela 24 mostra os resultados encontrados para os mesmos métodos de seleção dinâmica contudo, foi utilizado o indutor Árvore de Decisão. Pode-se observar que o $GPCD$ apresentou melhores resultados em todos os casos. Por exemplo, no método LCA, o $GPCDa_{ad}$ apresentou os melhores resultados em 42,9% dos problemas de classificação o que equivale a 12 problemas. Para a mesma técnica de SDC, o $GPCDd_{ad}$ obteve ganhos, em termos de acurácia, em 39% dos problemas testados.

Para o OLA, $GPCDa_{ad}$ obteve 64,3%, ou seja, 18 problemas sendo dois empates. Já o $GPCDd_{ad}$ ganhou em 35,7% (10) das bases de dados e obteve apenas um empate. Finalmente, usando o método de seleção de Rank, os melhores resultados foram obtidos pelo $GPCDa_{ad}$ com 64,3% (18) dos problemas, seguido por $GPCDd_{ad}$ com 35,7%.

Para finalizar os experimentos, foi realizada a comparação com as técnicas de seleção dinâmica de *ensemble* de classificadores, conforme os resultados dispostos nas Tabelas 25 e 26. Para o classificador Perceptron, o *pool* gerado por $GPCDd_{pe}$, quando utilizado pelo KNORA-E, ganhou em acurácia em 28,6%, isto é, em 8 dos 28 *datasets* utilizados. Já o Random Forest superou o $GPCD$, ganhando em 46,4% dos problemas (13). Quando empregado os *pools* no método KNORA-E o $GPCDa_{pe}$ obteve 25,0% e usando KNORA-U como método de seleção foi obtido como resultado 42,9% de vitórias o que equivale a 12 problemas. As soluções do $GPCDd_{pe}$ foram um pouco inferiores, para esses métodos de seleção. Por último, os resultados encontrados no método META-DES foram de 25,0% ($GPCDa_{pe}$) e 21,5% ($GPCDd_{pe}$) de todas bases de dados.

Tabela 23 – Comparação dos resultados entre métodos de geração de *pool* de classificadores, com classificador de base Perceptron, na seleção dinâmica de classificadores.

	LCA						**OLA**						**Rank**					
Prob.	*GPCDa*	*GPCDd*	Bagging	AdaBoost	R.F.	DSOC	*GPCDa*	*GPCDd*	Bagging	AdaBoost	R.F.	DSOC	*GPCDa*	*GPCDd*	Bagging	AdaBoost	R.F.	DSOC
01	**84,6(3,4)**	84,0(2,8)	82,9(2,7)	83,1(2,7)	82,3(3,0)	81,3(2,6)	83,0(2,2)	**83,3(2,5)**	82,9(1,9)	82,8(2,0)	79,0(1,9)	**83,3(3,2)**	81,1(2,9)	81,5(2,8)	81,4(2,7)	81,3(2,6)	79,0(2,1)	**81,9(3,8)**
02	95,7(1,0)	95,7(1,0)	95,7(0,9)	95,7(1,0)	**97,1(0,7)**	95,6(1,0)	**97,4(0,7)**	**97,4(0,6)**	**97,4(0,6)**	**97,4(0,7)**	96,5(0,6)	97,0(0,7)	**96,9(0,8)**	**96,9(0,7)**	**96,9(0,8)**	**96,9(0,8)**	96,5(0,7)	96,5(0,9)
03	68,9(3,9)	72,9(5,3)	70,4(5,3)	72,6(3,6)	**74,6(2,4)**	69,7(5,3)	75,2(3,3)	**75,7(3,0)**	74,8(2,5)	74,5(2,7)	72,1(3,8)	75,1(3,7)	70,0(4,2)	70,3(3,6)	69,8(3,7)	69,5(2,8)	**71,3(3,5)**	70,2(5,1)
04	86,8(1,8)	86,7(1,7)	86,2(1,9)	86,7(2,2)	**89,4(1,2)**	86,5(2,1)	89,3(1,2)	89,2(1,3)	89,0(1,4)	88,8(1,7)	**90,2(1,5)**	88,8(1,4)	89,3(1,4)	89,3(1,3)	88,9(1,6)	88,6(2,7)	**90,2(1,5)**	88,7(2,0)
05	69,9(3,3)	69,8(3,2)	**70,7(3,1)**	69,8(3,1)	68,1(2,6)	69,1(3,0)	**73,3(3,3)**	73,1(3,1)	72,4(2,6)	72,0(2,9)	68,6(2,3)	71,9(2,2)	**71,8(2,6)**	71,5(3,3)	71,5(2,6)	70,8(3,0)	68,5(3,0)	70,5(2,9)
06	64,5(1,9)	**65,0(2,4)**	62,8(2,4)	63,8(1,6)	64,8(1,9)	63,0(2,4)	67,4(1,5)	**68,7(1,5)**	67,7(1,9)	67,9(1,4)	66,4(1,8)	67,6(1,8)	67,2(1,8)	**68,1(1,6)**	67,4(2,1)	67,7(1,4)	66,2(2,4)	67,4(2,1)
07	**70,1(2,6)**	69,1(3,1)	69,1(2,2)	68,5(3,1)	67,3(2,6)	68,2(2,5)	70,4(2,1)	**71,3(2,6)**	71,2(2,3)	69,8(1,9)	67,0(3,0)	71,0(2,0)	69,4(1,9)	70,1(2,5)	**71,0(2,7)**	69,1(2,2)	66,6(3,1)	70,1(2,6)
08	62,0(7,4)	**72,8(3,7)**	69,0(7,8)	67,2(10,3)	70,7(4,1)	66,7(8,4)	69,5(4,4)	68,4(4,9)	69,7(4,5)	69,0(5,9)	66,2(5,2)	**69,8(5,2)**	66,0(5,1)	64,7(6,2)	65,7(5,2)	65,4(5,6)	65,8(5,3)	**66,5(5,9)**
09	80,1(5,5)	**80,6(5,2)**	75,7(7,1)	79,6(6,0)	72,5(5,9)	79,2(5,3)	**78,5(4,6)**	78,4(5,7)	77,5(4,5)	76,0(4,0)	71,9(5,2)	78,4(5,4)	**77,5(4,1)**	76,9(5,0)	77,1(4,5)	74,8(4,5)	72,0(5,0)	77,2(6,2)
10	**69,7(2,1)**	66,9(3,2)	66,8(4,8)	66,2(4,5)	67,9(4,2)	64,6(4,4)	**69,3(3,0)**	69,1(3,3)	69,0(3,5)	68,8(3,1)	67,3(3,7)	69,1(2,4)	67,7(3,8)	67,8(3,4)	68,0(2,8)	**68,2(3,0)**	67,5(3,7)	67,1(3,7)
11	**87,1(2,5)**	86,1(3,6)	85,4(3,9)	85,2(3,0)	84,7(4,0)	84,9(4,1)	**88,8(2,8)**	88,4(3,0)	85,9(2,7)	87,6(3,4)	87,1(3,5)	86,4(4,0)	**88,8(2,9)**	88,3(3,2)	86,0(2,8)	87,0(3,5)	87,2(3,5)	86,5(4,1)
12	**82,5(3,6)**	75,1(5,2)	77,6(6,8)	78,1(7,0)	79,8(4,6)	77,9(5,5)	81,3(3,8)	**81,4(4,7)**	80,6(3,7)	80,0(3,8)	77,3(4,9)	79,4(4,7)	81,3(3,9)	**81,4(3,8)**	78,6(3,5)	79,0(4,3)	76,3(4,6)	79,3(4,7)
13	67,6(4,4)	**70,5(4,9)**	68,1(4,4)	70,3(4,0)	67,8(4,7)	67,7(4,9)	67,3(4,2)	67,9(3,8)	68,0(4,8)	**69,2(4,1)**	63,7(4,9)	67,4(4,9)	64,3(5,2)	**66,6(4,3)**	64,1(5,8)	64,0(5,8)	63,4(5,1)	65,3(5,7)
14	93,4(1,2)	92,8(1,1)	92,7(1,0)	93,0(1,2)	**96,2(0,8)**	92,8(1,4)	96,4(0,6)	96,6(0,8)	**96,7(0,7)**	96,5(0,8)	95,8(0,9)	96,4(0,7)	95,7(0,7)	**95,9(0,8)**	**95,9(0,9)**	95,7(0,9)	95,7(0,9)	95,7(0,7)
15	57,9(5,5)	**58,0(4,6)**	55,3(5,3)	57,2(5,0)	55,0(5,6)	55,6(5,6)	66,6(5,0)	65,5(5,4)	**68,2(3,9)**	65,8(4,0)	63,1(4,6)	67,6(3,4)	66,2(4,3)	65,6(5,5)	**66,4(4,1)**	63,8(4,9)	61,7(4,4)	65,9(3,7)
16	74,1(5,4)	**80,4(2,8)**	79,7(3,6)	77,4(4,6)	79,5(2,3)	78,0(5,6)	**81,1(2,2)**	80,8(2,5)	80,8(2,6)	80,5(2,7)	78,4(2,9)	80,8(3,3)	77,1(2,7)	76,8(3,2)	77,4(2,4)	76,2(2,8)	**78,0(2,8)**	77,7(3,0)
17	72,8(2,9)	73,8(4,7)	72,8(4,3)	71,6(5,4)	**93,0(3,5)**	79,5(1,8)	87,6(3,2)	87,8(3,1)	85,7(3,6)	84,9(2,7)	**97,5(2,2)**	85,7(3,9)	87,8(3,4)	87,6(3,7)	85,5(3,8)	85,2(2,9)	**97,5(2,2)**	85,6(3,4)
18	78,8(1,6)	76,2(1,9)	75,7(2,4)	75,5(1,7)	**82,8(1,1)**	70,2(4,8)	**89,9(1,4)**	83,0(0,9)	82,8(0,6)	82,5(0,7)	85,0(0,9)	82,8(0,6)	**91,0(1,3)**	83,3(0,9)	83,0(0,8)	83,0(0,7)	85,0(0,8)	83,3(1,0)
19	76,5(2,4)	74,0(2,7)	73,4(3,0)	79,0(2,3)	**88,4(1,8)**	75,5(1,5)	82,4(0,8)	88,0(2,2)	88,1(1,7)	89,7(1,6)	**91,2(1,3)**	90,5(1,3)	82,8(0,8)	89,8(2,3)	90,1(1,7)	91,0(1,5)	91,2(1,3)	**91,8(1,2)**
20	88,5(1,4)	88,8(1,2)	88,7(1,2)	89,4(1,7)	**92,4(1,2)**	88,4(1,5)	92,6(1,2)	92,7(1,2)	92,8(1,4)	93,4(1,3)	**93,8(1,4)**	92,1(1,4)	92,8(1,3)	93,0(1,2)	93,0(1,1)	93,5(1,1)	**93,9(1,4)**	92,5(1,3)
21	73,2(6,9)	**73,3(7,4)**	66,9(6,8)	66,6(7,8)	63,7(7,1)	70,4(5,7)	77,7(6,5)	**78,6(6,9)**	75,1(6,2)	75,2(5,2)	70,6(7,0)	76,5(5,3)	77,5(7,2)	**79,6(6,1)**	74,8(5,5)	76,2(4,4)	70,8(7,0)	76,8(5,5)
22	94,4(1,4)	**95,5(1,6)**	95,0(1,9)	95,4(1,6)	94,6(1,2)	94,4(1,9)	96,1(1,3)	96,0(1,5)	95,7(1,7)	**96,2(1,6)**	94,5(1,4)	95,8(1,4)	95,6(1,6)	**95,7(1,8)**	95,1(1,7)	95,5(1,7)	94,4(1,5)	95,5(1,7)
23	68,1(2,3)	**69,1(3,0)**	67,1(2,5)	68,8(3,1)	64,3(3,6)	67,5(2,4)	74,0(2,0)	74,0(2,7)	73,1(2,0)	**74,5(3,1)**	67,5(2,0)	73,4(3,0)	**74,5(2,6)**	74,1(3,1)	73,0(2,0)	74,0(2,6)	67,4(1,8)	73,7(2,8)
24	81,5(4,2)	82,7(3,4)	81,7(4,6)	**83,3(4,2)**	78,2(4,6)	79,3(6,4)	**85,1(4,0)**	83,8(3,9)	83,8(4,3)	83,5(4,0)	81,1(4,0)	84,9(4,7)	**84,6(4,0)**	83,6(3,6)	84,3(5,0)	83,0(4,2)	81,3(4,2)	84,1(4,8)
25	**95,5(1,4)**	95,3(1,8)	95,0(1,4)	95,4(1,5)	91,3(2,3)	94,5(1,4)	96,3(1,1)	**96,6(1,3)**	96,0(1,3)	96,0(1,2)	92,5(2,2)	96,0(1,4)	96,2(1,1)	**96,7(1,3)**	95,9(1,3)	95,8(1,4)	92,4(2,2)	96,1(1,3)
26	**80,3(1,6)**	78,9(1,8)	79,2(2,1)	79,9(2,0)	74,1(1,2)	79,1(2,2)	**83,0(1,1)**	82,9(0,9)	82,4(1,2)	82,6(1,0)	73,0(1,2)	82,8(1,0)	**82,4(1,4)**	82,3(0,9)	82,0(0,9)	82,2(1,0)	72,9(1,2)	82,2(1,0)
27	**78,7(4,9)**	77,3(4,4)	75,4(5,0)	76,9(5,0)	74,7(7,5)	76,3(5,1)	**81,4(4,1)**	80,3(5,2)	79,7(4,5)	79,3(4,2)	77,8(3,4)	79,2(4,8)	**81,3(4,7)**	80,1(4,9)	79,9(4,7)	77,2(4,7)	78,0(3,0)	79,4(5,4)
28	**97,0(1,9)**	96,1(2,4)	94,2(4,3)	**97,0(2,7)**	90,6(5,2)	94,3(3,6)	**97,3(2,0)**	95,6(2,7)	95,0(2,2)	97,0(2,7)	89,9(4,8)	95,7(3,0)	**97,3(2,0)**	95,6(2,7)	95,0(2,2)	97,0(2,7)	89,9(4,8)	95,7(3,0)
Média	**78,6**	**78,8**	**77,6**	**78,3**	**78,8**	**77,5**	**82,1**	**81,9**	**81,5**	**81,5**	**79,5**	**81,6**	**81,2**	**81,2**	**80,6**	**80,4**	**79,3**	**80,8**

Tabela 24 – Comparação dos resultados entre métodos de geração de *pool* de classificadores, com classificador de base Árvore de Decisão, na seleção dinâmica de classificadores,

	LCA						**OLA**						**Rank**					
Prob.	*GPCDa*	*GPCDd*	Bagging	AdaBoost	R,F,	DSOC	*GPCDa*	*GPCDd*	Bagging	AdaBoost	R,F,	DSOC	*GPCDa*	*GPCDd*	Bagging	AdaBoost	R,F,	DSOC
01	85,4(2,1)	**86,1(2,8)**	84,5(1,9)	82,3(2,6)	82,3(3,0)	83,6(2,6)	**85,2(2,7)**	84,6(3,4)	82,4(3,1)	82,3(2,6)	79,0(1,9)	82,1(2,4)	**85,2(3,0)**	84,7(3,3)	82,1(2,7)	82,3(2,6)	79,0(2,1)	81,7(2,4)
02	97,0(0,8)	**97,1(0,6)**	96,9(0,7)	95,8(0,8)	**97,1(0,7)**	96,7(0,8)	**96,9(0,6)**	96,7(0,7)	96,5(0,8)	95,8(0,8)	96,5(0,6)	96,3(1,1)	**96,9(0,6)**	96,6(0,7)	96,4(0,8)	95,8(0,8)	96,5(0,7)	96,1(1,1)
03	**77,4(2,8)**	74,4(3,7)	74,5(2,4)	73,1(3,8)	74,6(2,4)	73,6(2,7)	**76,0(3,0)**	73,8(3,6)	72,6(2,8)	71,8(3,2)	72,1(3,8)	72,9(3,3)	**75,0(3,1)**	72,4(4,3)	70,3(3,9)	68,9(3,5)	71,3(3,5)	71,8(3,4)
04	**91,5(1,2)**	91,3(1,5)	89,6(1,7)	90,9(1,2)	89,4(1,2)	89,5(1,5)	**92,3(0,6)**	92,1(1,1)	91,2(1,2)	91,3(1,0)	90,2(1,5)	91,2(1,4)	**92,2(0,7)**	92,0(1,1)	91,1(1,1)	91,4(1,0)	90,2(1,5)	91,1(1,3)
05	**74,5(3,2)**	73,6(3,2)	69,8(3,8)	69,0(3,8)	68,1(2,6)	69,6(3,4)	**75,3(3,1)**	74,7(3,5)	69,5(2,9)	69,0(3,8)	68,6(2,3)	69,9(3,0)	**75,0(3,2)**	74,2(3,9)	69,4(2,5)	69,0(3,8)	68,5(3,0)	69,4(2,7)
06	69,4(2,9)	**69,9(2,1)**	65,3(2,5)	69,4(2,7)	64,8(1,9)	65,1(1,7)	72,8(3,1)	**73,6(2,9)**	67,9(2,2)	69,4(2,7)	66,4(1,8)	68,0(2,1)	72,6(3,5)	**73,3(2,8)**	67,5(1,8)	69,4(2,7)	66,2(2,4)	67,5(1,9)
07	71,2(3,3)	**71,9(4,4)**	67,5(2,6)	68,3(3,1)	67,3(2,6)	67,6(2,9)	**74,1(4,0)**	73,4(3,9)	67,7(2,6)	68,3(3,1)	67,0(3,0)	68,2(2,0)	**73,9(3,9)**	73,2(3,9)	67,5(2,6)	68,3(3,1)	66,6(3,1)	67,8(2,1)
08	69,8(5,6)	**72,1(3,6)**	71,1(4,8)	70,2(5,0)	70,7(4,1)	71,8(4,4)	**70,7(4,9)**	68,9(5,5)	65,5(5,5)	67,3(4,9)	66,2(5,2)	67,2(5,0)	**69,4(4,5)**	68,0(5,4)	64,5(4,7)	66,8(5,3)	65,8(5,3)	65,9(6,2)
09	74,3(3,8)	**79,9(6,1)**	73,8(4,0)	73,5(5,1)	72,5(5,9)	74,8(5,6)	78,1(3,8)	**79,3(5,9)**	75,5(3,7)	73,5(5,1)	71,9(5,2)	74,3(5,6)	77,6(3,8)	**78,5(5,5)**	74,7(3,9)	73,5(5,1)	72,0(5,0)	73,8(5,6)
10	69,4(4,3)	**69,9(3,9)**	66,8(4,3)	65,8(3,9)	67,9(4,2)	65,4(4,1)	70,5(4,6)	**72,1(3,7)**	65,8(3,7)	65,8(3,9)	67,3(3,7)	65,2(3,4)	69,6(4,5)	**72,1(4,0)**	65,5(3,1)	65,8(3,9)	67,5(3,7)	65,4(3,1)
11	86,5(3,2)	85,7(2,9)	84,3(2,8)	**87,4(4,5)**	84,7(4,0)	84,8(4,0)	89,9(2,3)	**90,0(1,9)**	87,9(3,0)	87,4(4,5)	87,1(3,5)	86,9(2,9)	89,9(2,4)	**90,0(1,9)**	88,3(3,1)	87,4(4,5)	87,2(3,5)	86,6(2,7)
12	**81,9(5,5)**	80,3(4,4)	80,1(4,8)	77,8(4,7)	79,8(4,6)	77,4(6,1)	**82,1(5,4)**	80,4(4,7)	77,3(5,4)	77,8(4,7)	77,3(4,9)	76,3(6,1)	**82,6(5,6)**	80,5(4,6)	77,5(6,2)	77,8(4,7)	76,3(4,6)	76,7(6,2)
13	72,0(4,1)	**72,2(4,8)**	66,0(4,1)	63,2(4,6)	67,8(4,7)	68,5(4,5)	69,3(5,8)	**72,4(4,2)**	63,4(5,6)	63,2(4,6)	63,7(4,9)	64,9(3,9)	69,0(6,1)	**72,0(4,8)**	62,0(5,6)	63,2(4,6)	63,4(5,1)	64,0(4,6)
14	95,9(1,0)	**96,6(0,9)**	96,1(0,8)	95,6(0,8)	96,2(0,8)	95,8(1,0)	96,1(0,8)	**96,5(0,8)**	96,3(0,9)	95,6(0,8)	95,8(0,9)	96,0(0,7)	95,9(0,8)	**96,4(0,8)**	96,1(0,9)	95,6(0,8)	95,7(0,9)	95,8(0,5)
15	**63,0(3,9)**	60,3(7,8)	55,6(5,7)	60,2(4,1)	55,0(5,6)	55,5(4,0)	**70,2(4,1)**	68,4(5,2)	61,3(4,3)	60,2(4,1)	63,1(4,6)	62,6(4,6)	**69,7(4,5)**	68,1(5,5)	60,8(4,9)	60,2(4,1)	61,7(4,4)	62,8(4,3)
16	**82,0(2,9)**	81,0(2,7)	80,0(2,3)	74,6(3,7)	79,5(2,3)	79,2(3,6)	**81,0(3,6)**	80,2(3,7)	77,2(3,4)	77,6(2,7)	78,4(2,9)	78,2(2,6)	**80,3(3,2)**	79,2(3,7)	76,0(3,4)	74,5(2,9)	78,0(2,8)	77,8(2,1)
17	99,9(0,4)	99,0(1,5)	98,2(2,4)	**100,0(0)**	93,0(3,5)	97,5(3,4)	**100,0(0)**	99,5(0,7)	99,2(1,4)	**100,0(0)**	97,5(2,2)	99,5(1,0)	**100,0(0)**	99,5(0,7)	99,2(1,4)	**100,0(0)**	97,5(2,2)	99,5(1,0)
18	90,3(1,5)	89,6(1,1)	88,2(1,3)	**91,7(1,7)**	88,4(1,8)	81,5(1,0)	**93,3(0,9)**	92,7(1,1)	91,5(1,0)	91,7(1,7)	91,2(1,3)	84,5(1,1)	**93,2(0,9)**	92,8(1,1)	91,6(1,0)	91,7(1,7)	91,2(1,3)	84,6(1,1)
19	84,2(2,0)	84,4(1,7)	81,9(1,1)	84,8(1,1)	82,8(1,1)	**87,8(1,5)**	86,6(1,5)	86,9(1,5)	84,6(0,9)	84,8(1,1)	85,0(0,9)	**92,0(1,2)**	86,6(1,5)	86,9(1,4)	84,8(0,9)	84,8(1,1)	85,0(0,8)	**92,1(1,2)**
20	94,7(1,2)	94,5(1,0)	93,4(0,9)	**95,3(1,1)**	92,4(1,2)	93,6(1,0)	**95,6(1,0)**	**95,6(0,9)**	95,2(1,2)	95,3(1,1)	93,8(1,4)	94,9(1,2)	**95,6(1,0)**	**95,6(0,9)**	95,1(1,2)	95,3(1,1)	93,9(1,4)	94,9(1,2)
21	70,0(7,4)	**70,8(6,4)**	64,9(8,4)	69,4(6,8)	63,7(7,1)	61,7(7,5)	74,7(7,2)	**75,8(5,0)**	71,1(6,3)	69,4(6,8)	70,6(7,0)	72,0(5,4)	74,5(6,8)	**75,7(5,0)**	71,4(6,3)	69,4(6,8)	70,8(7,0)	72,4(4,4)
22	**95,6(1,6)**	94,1(1,7)	95,3(1,2)	95,1(0,9)	94,6(1,2)	94,5(1,4)	95,4(1,3)	**95,8(1,8)**	94,8(1,6)	95,1(0,9)	94,5(1,4)	94,5(1,4)	95,3(1,3)	**95,8(1,7)**	94,8(1,6)	95,1(0,9)	94,4(1,5)	94,3(1,4)
23	**70,9(2,8)**	68,1(3,8)	64,2(3,9)	68,3(2,5)	64,3(3,6)	65,4(3,7)	**74,7(3,6)**	73,5(3,2)	68,4(2,4)	68,3(2,5)	67,5(2,0)	70,0(4,3)	**75,0(3,2)**	73,0(3,2)	68,5(3,5)	68,3(2,5)	67,4(1,8)	70,2(4,3)
24	**83,6(3,4)**	82,4(4,2)	79,8(4,4)	83,4(3,3)	78,2(4,6)	79,9(4,0)	**85,5(3,1)**	84,5(4,0)	82,8(3,3)	83,4(3,3)	81,1(4,0)	83,5(3,6)	**86,1(2,6)**	84,4(4,0)	82,1(4,0)	83,4(3,3)	81,3(4,2)	82,9(3,1)
25	**94,4(1,9)**	92,2(1,7)	92,1(1,9)	92,3(1,9)	91,3(2,3)	92,0(2,2)	94,5(2,4)	**94,6(1,5)**	93,0(2,0)	92,3(1,9)	92,5(2,2)	92,2(2,4)	**94,6(2,4)**	**94,6(1,6)**	93,0(2,1)	92,3(1,9)	92,4(2,2)	92,3(2,3)
26	**79,3(1,8)**	78,0(2,2)	74,9(1,3)	74,6(1,1)	74,1(1,2)	75,2(1,1)	**78,8(1,7)**	77,8(2,7)	74,4(1,3)	74,6(1,1)	73,0(1,2)	73,9(0,9)	**78,7(1,8)**	77,7(2,7)	74,4(1,2)	74,6(1,1)	72,9(1,2)	73,7(1,0)
27	78,9(6,2)	**79,8(5,7)**	73,4(6,3)	77,1(5,5)	74,7(7,5)	74,9(6,7)	**82,9(4,7)**	82,8(4,0)	77,9(4,6)	77,1(5,5)	77,8(3,4)	78,1(4,2)	82,7(4,4)	**82,9(3,8)**	78,0(4,6)	77,1(5,5)	78,0(3,0)	76,7(4,7)
28	**94,2(3,6)**	93,4(3,5)	91,7(4,7)	89,4(5,9)	90,6(5,2)	90,1(5,1)	**93,6(4,0)**	92,4(3,2)	91,5(4,5)	89,4(5,9)	89,9(4,8)	91,2(4,6)	**93,6(4,0)**	92,4(3,2)	91,5(4,5)	89,4(5,9)	89,9(4,8)	91,2(4,6)
Média	**82,0**	**81,7**	**79,3**	**79,9**	**78,8**	**79,0**	**83,4**	**83,2**	**80,1**	**79,9**	**79,5**	**80,2**	**83,2**	**82,9**	**79,8**	**79,7**	**79,3**	**80,0**

Tabela 25 – Comparação dos resultados entre os métodos de geração de *pool* de classificadores, com classificador de base Perceptron, na seleção *ensemble* de classificadores.

	Knora-E						**Knora-U**						**Meta-Des**					
Prob.	GPCDa	*GPCDd*	Bagging	AdaBoost	R.F.	DSOC	*GPCDa*	*GPCDd*	Bagging	AdaBoost	R.F.	DSOC	*GPCDa*	*GPCDd*	Bagging	AdaBoost	R.F.	DSOC
01	82,3(3,1)	82,3(2,8)	82,2(3,1)	82,4(3,2)	**84,0(2,3)**	82,8(3,6)	87,5(2,3)	**88,1(2,0)**	86,9(2,3)	86,8(1,9)	88,0(2,0)	85,9(1,9)	86,2(2,7)	86,0(2,6)	86,2(2,5)	84,0(3,3)	**88,0(2,5)**	84,7(2,2)
02	96,9(0,8)	96,9(0,6)	96,9(0,7)	97,0(0,8)	**97,1(0,8)**	96,6(0,9)	96,1(1,0)	96,2(0,9)	96,2(1,0)	96,2(1,0)	**97,3(0,8)**	95,6(0,8)	94,4(1,1)	94,9(1,4)	93,3(1,2)	93,7(1,6)	**97,2(0,8)**	92,7(2,0)
03	70,3(4,1)	70,1(3,3)	70,2(3,8)	69,9(3,5)	**72,1(3,4)**	70,7(4,5)	78,3(2,4)	78,0(1,7)	**78,7(1,6)**	78,5(1,6)	75,1(2,1)	76,8(2,4)	76,0(1,1)	**76,2(1,1)**	76,0(0,9)	76,1(0,9)	75,8(1,4)	76,0(0,9)
04	89,8(1,5)	90,2(1,3)	89,5(1,6)	89,1(2,1)	**93,5(0,9)**	89,4(1,8)	89,5(1,0)	89,7(1,2)	89,0(2,1)	88,7(3,0)	**93,7(1,0)**	89,0(1,9)	89,4(1,2)	89,6(1,2)	88,8(1,1)	88,4(1,3)	**93,9(1,0)**	88,7(1,3)
05	**73,1(2,4)**	72,3(2,5)	72,5(2,7)	71,8(2,9)	71,7(3,0)	71,6(2,7)	**77,0(2,7)**	**77,0(2,8)**	76,9(2,7)	76,2(2,8)	75,8(2,8)	76,0(3,0)	74,5(3,3)	74,4(2,6)	73,7(2,5)	73,1(2,4)	**74,8(3,0)**	74,1(2,0)
06	68,6(2,0)	**69,9(1,4)**	68,9(1,7)	69,1(1,7)	69,8(1,8)	68,9(1,6)	72,3(1,6)	72,4(1,3)	71,7(1,5)	72,1(1,3)	**76,3(1,7)**	69,6(1,7)	70,8(1,3)	70,8(1,9)	70,3(1,4)	70,1(1,5)	**76,5(1,9)**	69,2(2,0)
07	71,0(1,5)	71,5(2,2)	**72,4(2,3)**	70,3(2,0)	71,1(1,8)	71,0(2,9)	**76,9(2,4)**	76,7(2,3)	76,1(2,4)	75,2(1,8)	75,3(2,4)	74,8(2,7)	73,0(2,0)	**73,8(1,8)**	72,9(1,7)	71,6(1,4)	72,9(1,8)	72,4(1,6)
08	67,2(4,8)	66,4(5,6)	65,6(6,1)	66,1(4,5)	**68,2(4,7)**	67,2(5,8)	**76,4(3,3)**	75,7(2,2)	75,1(2,1)	74,9(2,0)	70,5(3,7)	75,0(2,7)	73,4(1,6)	73,5(1,7)	**74,0(1,5)**	73,7(1,6)	72,0(2,6)	73,8(2,1)
09	**80,9(4,6)**	78,9(4,3)	79,4(4,2)	76,3(5,0)	76,0(5,3)	79,0(4,4)	**85,5(4,2)**	85,1(3,4)	82,8(3,8)	83,0(4,6)	80,1(5,3)	82,6(2,6)	**84,6(3,6)**	83,1(3,9)	82,9(4,0)	79,3(3,9)	79,9(3,9)	81,9(3,5)
10	67,9(3,0)	68,8(2,8)	68,9(2,3)	68,7(3,1)	**69,6(3,5)**	68,0(3,2)	70,9(2,4)	**72,4(2,1)**	71,0(2,4)	70,3(2,7)	71,0(3,2)	69,8(2,6)	70,7(2,2)	70,1(2,1)	70,7(1,6)	70,7(1,5)	**70,8(1,9)**	70,7(1,9)
11	88,8(3,7)	90,0(3,3)	88,4(3,6)	85,9(3,3)	**91,9(3,4)**	88,5(3,6)	90,7(3,1)	91,5(3,8)	89,2(2,8)	88,2(3,5)	**93,4(2,7)**	89,0(3,2)	90,3(2,7)	89,8(3,2)	88,8(3,0)	87,1(3,3)	**93,4(2,9)**	88,9(2,9)
12	**82,5(3,9)**	**82,5(4,4)**	79,9(3,7)	78,7(5,2)	79,7(4,4)	82,4(4,3)	**85,6(4,9)**	84,1(3,7)	83,0(3,9)	83,6(4,5)	82,8(4,5)	84,1(4,9)	**84,5(4,4)**	84,3(5,6)	83,0(5,5)	80,9(5,2)	83,1(4,4)	82,8(4,7)
13	66,5(4,6)	**68,5(4,5)**	66,7(4,0)	66,3(5,4)	67,7(5,2)	67,0(4,8)	75,1(3,0)	**75,9(2,6)**	73,5(2,1)	72,7(2,8)	73,0(2,8)	73,0(3,6)	71,2(3,0)	**72,1(3,2)**	71,1(2,8)	70,9(3,4)	70,2(4,1)	70,1(2,2)
14	95,8(0,7)	96,0(0,7)	96,0(0,7)	95,9(0,9)	**96,7(0,8)**	95,8(0,8)	95,0(1,2)	94,8(1,1)	95,1(1,1)	95,2(0,9)	**97,0(0,6)**	94,4(1,4)	93,9(1,4)	94,1(1,0)	94,2(1,1)	94,5(1,1)	**97,0(0,6)**	93,6(1,3)
15	65,6(4,9)	66,2(4,6)	**67,0(4,1)**	63,6(4,9)	63,3(4,0)	**67,0(4,2)**	**72,1(3,8)**	70,1(5,2)	70,1(3,5)	68,1(4,4)	69,0(4,6)	67,6(4,6)	67,7(4,8)	**68,2(4,1)**	67,2(4,7)	64,7(4,9)	68,0(4,4)	66,5(4,4)
16	77,4(2,8)	77,6(2,9)	77,2(2,7)	76,9(2,7)	**79,1(2,6)**	78,0(2,8)	**83,3(2,3)**	**83,3(2,3)**	82,9(1,9)	82,1(2,3)	79,3(2,0)	82,5(2,4)	79,9(3,3)	79,6(3,3)	79,7(3,0)	79,1(3,1)	79,2(2,3)	**80,1(3,5)**
17	89,4(3,1)	89,5(2,9)	88,9(3,1)	87,1(3,4)	**99,6(0,8)**	88,1(2,6)	87,3(1,9)	84,6(2,7)	81,8(3,5)	82,9(3,4)	**99,0(1,4)**	82,0(3,5)	94,4(3,2)	94,1(2,9)	91,5(4,5)	89,9(3,5)	**99,6(0,9)**	90,2(3,0)
18	**91,1(1,3)**	83,5(0,8)	83,5(0,9)	83,4(0,7)	88,3(0,8)	83,7(0,8)	**89,5(1,3)**	81,9(0,9)	81,8(1,0)	81,6(0,9)	**89,5(0,7)**	80,9(1,1)	88,7(2,2)	82,4(0,9)	82,5(0,9)	82,5(0,9)	**90,2(0,8)**	82,0(0,8)
19	83,1(0,8)	89,5(2,2)	90,5(1,6)	90,8(1,6)	**93,7(0,8)**	91,8(1,1)	81,7(0,9)	85,8(2,3)	87,2(2,8)	90,3(1,2)	**94,0(0,9)**	89,0(1,6)	82,4(1,0)	86,8(2,6)	86,8(2,5)	88,7(2,1)	**94,6(0,9)**	88,9(2,5)
20	94,0(1,1)	94,2(1,0)	94,2(1,0)	94,5(0,9)	**97,5(0,8)**	93,8(1,1)	91,9(1,0)	92,4(1,4)	92,3(1,0)	92,9(1,2)	**97,1(1,0)**	91,8(1,0)	94,0(1,0)	94,3(0,9)	94,0(1,1)	94,3(0,9)	**97,6(0,8)**	93,8(1,0)
21	**83,2(6,3)**	82,2(4,5)	78,5(3,5)	77,3(5,5)	77,9(5,0)	78,7(5,2)	**84,8(6,3)**	82,1(6,4)	78,1(3,0)	74,8(4,1)	78,7(4,0)	76,5(3,6)	**84,9(5,3)**	82,7(6,0)	79,7(3,7)	76,6(5,7)	80,2(3,9)	78,8(3,8)
22	96,2(1,6)	**96,3(1,5)**	95,9(1,8)	95,6(1,5)	96,2(0,8)	96,0(1,5)	97,2(1,0)	**97,4(0,9)**	97,0(1,0)	97,0(1,1)	96,1(0,8)	96,6(1,3)	**97,0(0,9)**	96,7(0,8)	96,4(0,9)	96,4(1,0)	95,5(1,3)	96,4(0,9)
23	75,5(2,5)	**75,8(2,6)**	**75,8(2,2)**	74,8(2,2)	71,0(2,1)	74,8(2,4)	76,5(2,9)	**77,7(2,1)**	76,0(2,1)	76,8(2,6)	74,0(2,2)	75,1(2,4)	76,2(1,9)	**76,6(2,4)**	75,9(2,3)	75,9(2,7)	74,5(1,6)	74,9(2,7)
24	**85,2(3,9)**	85,1(3,5)	84,8(4,3)	83,1(4,4)	84,1(4,1)	84,9(3,8)	**88,0(4,2)**	87,1(3,5)	86,5(4,0)	86,0(3,5)	85,8(3,0)	86,1(4,1)	**88,0(3,0)**	87,3(3,0)	86,7(3,8)	86,1(3,3)	85,1(4,3)	86,7(3,6)
25	97,4(1,0)	**97,7(1,2)**	97,1(0,9)	96,3(1,3)	95,4(1,6)	97,1(1,1)	98,1(0,7)	**98,2(1,0)**	97,1(1,2)	96,7(1,3)	95,5(1,5)	97,0(1,0)	97,2(1,2)	**97,5(1,6)**	96,9(1,3)	96,0(1,3)	95,7(1,4)	96,7(1,4)
26	83,7(1,0)	**83,8(1,0)**	83,4(0,8)	83,6(1,0)	80,3(0,8)	83,3(1,2)	**86,8(0,9)**	**86,8(1,0)**	86,4(0,8)	86,6(0,8)	85,1(0,8)	85,8(0,9)	**86,1(0,9)**	85,9(0,9)	85,9(0,8)	85,8(0,7)	84,8(0,9)	85,6(0,8)
27	82,9(3,8)	81,9(4,7)	82,1(4,6)	75,9(5,5)	**84,9(3,2)**	80,5(4,5)	85,2(3,8)	85,5(3,7)	83,2(3,6)	82,5(4,5)	**89,7(2,7)**	81,8(3,6)	85,3(3,4)	84,4(4,1)	82,7(3,4)	81,1(4,8)	**89,1(3,6)**	80,8(4,4)
28	**98,4(1,9)**	97,3(2,4)	97,5(2,0)	97,2(2,6)	96,6(2,8)	97,2(1,6)	**98,5(1,8)**	98,0(2,3)	97,5(2,1)	97,2(2,6)	97,2(3,0)	97,4(1,7)	**98,4(1,9)**	98,2(2,2)	98,0(2,0)	97,3(2,2)	97,2(2,7)	97,2(2,0)
Média	**82,3**	**82,3**	**81,9**	**81,0**	**82,8**	**81,9**	**84,9**	**84,6**	**83,7**	**83,5**	**85,0**	**83,1**	**84,0**	**83,8**	**83,2**	**82,4**	**84,9**	**82,8**

Tabela 26 – Comparação dos resultados entre os métodos de geração de *pool* de classificadores, com classificador de base Árvore de Decisão, na seleção *ensemble* de classificadores.

	Knora-E						**Knora-U**						**Meta-Des**					
Prob.	GPCDa	*GPCDd*	Bagging	AdaBoost	R.F.	DSOC	*GPCDa*	*GPCDd*	Bagging	AdaBoost	R.F.	DSOC	*GPCDa*	*GPCDd*	Bagging	AdaBoost	R.F.	DSOC
01	**88,6(2,7)**	88,1(2,9)	84,9(3,0)	82,3(2,6)	84,0(2,3)	84,2(2,2)	**92,4(2,5)**	91,1(3,0)	87,6(2,1)	82,3(2,6)	88,0(2,0)	86,8(1,9)	**91,7(2,4)**	90,4(2,5)	86,6(2,8)	82,3(2,6)	87,9(2,5)	85,8(2,0)
02	**97,6(0,7)**	97,5(0,6)	97,2(0,7)	95,8(0,8)	97,1(0,8)	96,9(0,7)	**97,9(0,7)**	97,7(0,7)	97,1(0,9)	95,8(0,8)	97,3(0,8)	97,0(0,9)	**97,9(0,7)**	97,6(0,7)	97,1(0,8)	95,8(0,8)	97,2(0,8)	96,9(0,9)
03	**75,8(3,4)**	75,0(4,1)	71,8(4,4)	69,8(3,7)	72,1(3,4)	72,8(3,5)	**82,7(2,1)**	80,8(2,7)	77,4(2,6)	76,1(2,2)	75,1(2,1)	76,3(3,5)	**79,1(1,8)**	78,8(2,1)	75,6(1,1)	76,0(1,0)	75,8(1,4)	75,7(1,3)
04	**94,7(1,2)**	94,6(1,1)	93,0(0,9)	91,9(1,4)	93,5(0,9)	92,9(0,9)	**95,7(1,0)**	95,4(1,0)	93,3(1,0)	92,0(1,5)	93,7(1,0)	92,9(0,8)	**95,6(0,9)**	95,3(0,8)	93,4(1,0)	92,0(1,5)	93,9(1,0)	93,1(0,9)
05	**79,0(2,7)**	77,0(4,1)	71,7(2,9)	69,0(3,8)	71,7(3,0)	72,0(3,0)	**85,0(2,8)**	82,1(4,5)	76,5(2,6)	69,0(3,8)	75,8(2,8)	75,6(3,2)	**84,8(2,8)**	81,5(5,1)	75,0(2,4)	69,0(3,8)	74,8(3,0)	74,3(3,0)
06	76,6(3,6)	**77,2(2,8)**	71,0(1,8)	69,4(2,7)	69,8(1,8)	69,9(1,7)	83,6(4,8)	**84,5(2,6)**	75,9(1,6)	69,4(2,7)	76,3(1,7)	74,5(2,0)	84,0(4,2)	**84,6(2,9)**	76,2(1,7)	69,4(2,7)	76,5(1,9)	74,6(1,8)
07	**79,5(3,7)**	77,8(4,4)	70,8(2,1)	68,3(3,1)	71,1(1,8)	71,0(2,3)	**85,5(5,1)**	81,8(5,3)	75,3(2,5)	68,3(3,1)	75,3(2,4)	74,0(3,2)	**85,1(4,7)**	81,0(5,2)	73,1(2,8)	69,7(1,7)	72,9(1,8)	72,4(2,3)
08	**73,2(5,0)**	71,6(4,8)	66,9(4,5)	69,4(4,9)	68,2(4,7)	68,0(5,3)	**80,3(5,9)**	79,9(4,9)	71,5(3,5)	71,8(4,3)	70,5(3,7)	71,6(4,5)	**78,4(5,3)**	77,4(4,8)	72,0(3,6)	73,2(1,3)	72,0(2,6)	71,8(2,8)
09	**82,2(4,5)**	80,2(4,1)	78,3(4,6)	73,5(5,1)	76,0(5,3)	75,6(4,7)	**89,8(3,8)**	87,8(4,7)	80,5(4,6)	73,5(5,1)	80,1(5,3)	79,7(5,7)	**88,2(4,1)**	85,1(4,5)	79,4(3,7)	73,5(5,1)	79,9(3,9)	78,3(5,7)
10	73,1(4,6)	**76,1(4,0)**	67,7(3,1)	65,8(3,9)	69,6(3,5)	67,4(3,5)	81,4(5,1)	**81,7(4,9)**	70,5(3,1)	65,8(3,9)	71,0(3,2)	69,7(3,3)	78,8(4,6)	**80,5(4,1)**	70,8(1,6)	71,2(0)	70,8(1,9)	70,3(2,1)
11	**95,7(2,0)**	94,9(2,5)	91,1(3,5)	87,4(4,5)	91,9(3,4)	90,4(3,3)	**96,2(1,5)**	96,0(2,0)	92,6(2,7)	87,4(4,5)	93,4(2,7)	92,0(2,9)	**96,4(1,6)**	96,0(1,9)	92,7(3,0)	87,4(4,5)	93,4(2,9)	92,2(2,2)
12	**85,5(5,5)**	84,1(3,8)	80,9(5,2)	77,8(4,7)	79,7(4,4)	79,5(6,1)	**91,0(4,9)**	88,1(5,8)	82,9(4,3)	77,8(4,7)	82,8(4,5)	82,7(4,8)	**90,7(5,3)**	87,7(5,3)	82,2(4,6)	77,8(4,7)	83,1(4,4)	81,8(5,7)
13	72,6(5,3)	**74,5(5,7)**	66,5(4,5)	63,2(4,6)	67,7(5,2)	66,6(4,0)	80,5(5,2)	**81,1(4,3)**	73,2(3,4)	63,2(4,6)	73,0(2,8)	71,1(3,2)	78,5(6,2)	**79,0(5,9)**	70,2(4,4)	67,9(3,9)	70,2(4,1)	69,0(3,8)
14	**97,1(0,6)**	**97,1(0,7)**	96,6(0,7)	95,6(0,8)	96,7(0,8)	96,5(0,8)	**97,9(0,7)**	97,7(0,5)	97,2(0,6)	95,6(0,8)	97,0(0,6)	96,8(0,6)	**97,7(0,7)**	97,5(0,6)	97,0(0,7)	95,6(0,8)	97,0(0,6)	96,8(0,6)
15	**73,7(4,5)**	70,7(5,5)	63,4(3,9)	60,2(4,1)	63,3(4,0)	64,8(5,0)	**84,5(3,5)**	79,1(7,1)	70,1(4,0)	60,2(4,1)	69,0(4,6)	68,3(3,7)	**83,7(4,1)**	78,7(6,1)	67,9(4,6)	59,8(4,1)	68,0(4,4)	66,4(5,2)
16	**81,7(3,0)**	80,1(2,9)	78,9(2,8)	76,4(3,3)	79,1(2,6)	79,6(2,6)	**85,1(4,0)**	84,3(3,4)	81,5(2,1)	74,5(4,5)	79,3(2,0)	80,3(2,0)	**83,2(2,9)**	82,2(3,2)	80,4(2,8)	79,3(2,1)	79,2(2,3)	79,9(2,9)
17	**100,0(0)**	**100,0(0)**	**100,0(0)**	**100,0(0)**	99,6(0,8)	93,5(1,3)	**100,0(0)**	**100,0(0)**	**100,0(0)**	**100,0(0)**	99,0(1,4)	99,6(1,1)	**100,0(0)**	**100,0(0)**	**100,0(0)**	**100,0(0)**	99,6(0,9)	**100,0(0)**
18	95,3(1,1)	95,0(1,0)	93,4(1,0)	91,7(1,7)	93,7(0,8)	**99,8(0,8)**	**96,9(1,3)**	95,9(1,1)	93,6(1,0)	91,7(1,7)	94,0(0,9)	93,0(1,2)	**96,5(1,1)**	96,0(1,0)	94,5(1,0)	91,7(1,7)	94,6(0,9)	88,7(0,8)
19	**90,2(2,1)**	**90,2(1,5)**	87,4(0,8)	84,8(1,1)	88,3(0,8)	86,8(0,7)	**92,1(2,5)**	92,0(1,9)	88,5(0,9)	84,8(1,1)	89,5(0,7)	87,5(0,8)	92,3(2,1)	92,6(1,5)	89,7(0,7)	84,8(1,1)	90,2(0,8)	**94,0(1,2)**
20	**98,0(1,0)**	97,9(0,8)	96,8(0,8)	95,3(1,1)	97,5(0,8)	96,4(0,9)	**98,1(0,9)**	97,9(0,9)	96,4(0,9)	95,3(1,1)	97,1(1,0)	96,0(1,0)	**98,3(1,1)**	98,2(0,8)	97,0(0,7)	95,3(1,1)	97,6(0,8)	96,6(1,0)
21	80,3(4,8)	**81,9(4,9)**	75,7(4,9)	69,4(6,8)	77,9(5,0)	75,2(4,6)	84,3(5,2)	**85,8(6,6)**	75,7(5,6)	69,4(6,8)	78,7(4,0)	73,2(5,8)	83,8(5,4)	**85,5(6,6)**	78,3(3,7)	69,4(6,8)	80,2(3,9)	76,1(4,5)
22	96,7(1,4)	**97,1(1,1)**	95,6(1,1)	95,1(0,9)	96,2(0,8)	95,4(1,3)	97,6(1,3)	**97,8(1,0)**	95,9(1,1)	95,1(0,9)	96,1(0,8)	95,8(1,0)	97,6(1,4)	**97,9(1,2)**	95,4(1,0)	94,7(1,4)	95,5(1,3)	95,5(0,9)
23	**79,5(3,4)**	77,5(3,9)	70,9(3,0)	68,3(2,5)	71,0(2,1)	72,1(3,8)	**85,6(3,1)**	84,9(5,1)	73,3(1,8)	68,3(2,5)	74,0(2,2)	73,2(3,2)	**84,7(3,0)**	83,7(6,0)	73,9(2,2)	68,3(2,5)	74,5(1,6)	73,4(3,0)
24	**88,0(3,7)**	87,1(2,7)	83,2(4,3)	83,4(3,3)	84,1(4,1)	83,7(3,1)	92,1(3,9)	**92,3(4,1)**	85,8(3,3)	83,4(3,3)	85,8(3,0)	84,5(3,5)	**92,1(4,4)**	91,8(4,1)	84,9(3,2)	83,4(3,3)	85,1(4,3)	84,1(3,8)
25	**97,0(1,2)**	96,2(1,4)	94,8(1,3)	92,3(1,9)	95,4(1,6)	94,8(1,2)	**97,4(1,8)**	96,6(1,7)	95,0(1,7)	92,3(1,9)	95,5(1,5)	94,5(2,0)	**97,4(1,7)**	96,8(1,2)	95,0(1,5)	92,3(1,9)	95,7(1,4)	94,5(1,7)
26	**86,3(2,2)**	85,2(3,1)	80,0(1,1)	74,6(1,1)	80,3(0,8)	78,7(1,0)	**90,5(2,5)**	89,6(2,9)	84,4(0,8)	74,6(1,1)	85,1(0,8)	83,3(0,9)	**90,3(2,6)**	89,5(2,9)	83,9(0,8)	74,6(1,1)	84,8(0,9)	82,9(0,8)
27	**88,9(4,2)**	86,9(4,3)	82,0(4,3)	77,1(5,5)	84,9(3,2)	79,3(3,8)	**93,5(3,3)**	90,3(3,1)	86,5(4,0)	77,1(5,5)	89,7(2,7)	83,7(3,5)	**93,3(3,1)**	92,3(3,5)	86,6(3,6)	77,1(5,5)	89,1(3,6)	84,1(4,1)
28	**98,6(2,0)**	96,4(2,7)	96,2(2,8)	89,4(5,9)	96,6(2,8)	95,5(3,4)	**99,4(1,0)**	96,5(2,5)	95,7(4,1)	89,4(5,9)	97,2(3,0)	94,9(3,8)	**99,1(1,3)**	96,2(2,7)	96,0(4,2)	89,4(5,9)	97,2(2,7)	94,7(4,0)
Média	**86,6**	**86,0**	**82,4**	**79,9**	**82,8**	**82,1**	**90,6**	**89,6**	**84,8**	**80,1**	**85,0**	**83,9**	**90,0**	**89,1**	**84,5**	**80,7**	**84,9**	**83,7**

Na próxima seção são discutidos e analisados os resultados encontrados nos testes realizados. Também são apresentados a sumarização e analise estatística.

5.5 Discussões e analises estatísticas

Os resultados encontrados na seção anterior foram analisados e comparados estatisticamente e submetidos a testes estatísticos para mensurar a diferença entre os métodos existentes na literatura e o método proposto com suas variantes.

A Figura 30 mostra um resumo das soluções encontradas para o Voto Majoritário em gráficos de "ganho-empate-perda" (G-E-P) comparando os métodos de modo pareado. A análise de ganho-empate-perda nos permite calcular o nível crítico (Nc). O nível crítico é representado pelo número de vitórias mais metade do número de empates, esse deve ser maior que Nc (Equação 5.1), onde $N_{exp} = 28$ (número de experimentos), $Z = 2,57$, $Nc = \{20,8\}$ para o nível de significância $c = \{0,01\}$, respectivamente.

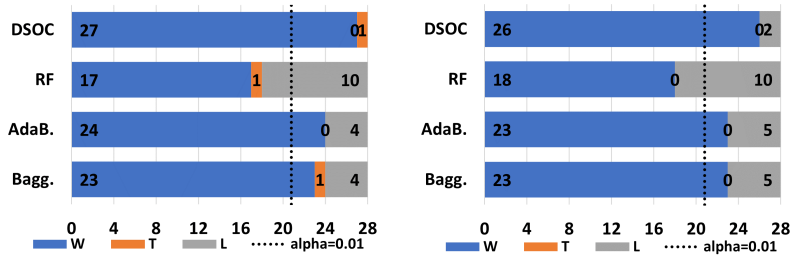
$$Nc = \frac{N_{exp}}{2} + Z_c \frac{\sqrt{N_{exp}}}{2} \tag{5.1}$$

Pode-se observar, considerando os 28 problemas de classificação e o Perceptron como indutor de base, quando comparado a Bagging, AdaBoost e RF, o $GPCDa$ obteve 90 vitórias (80,3 %), dois empates e 20 derrotas em um total de 112 comparações. Os resultados são ainda mais promissores quando utilizada a árvore de decisão como classificador de base. Neste caso, o $GPCDa$ obteve 109 vitórias (97,3 %), dois empates e uma derrota. Também pode-se observar na Figura 30, que o desempenho do $GPCDd$ foi um pouco inferior. O $GPCDd$ com Perceptron obteve 91 vitórias (81,2 %), três empates e 18 derrotas em um total de 112 comparações. Da mesma forma, os resultados são mais promissores quando utilizada a árvore de decisão como classificador de base. Neste caso, o $GPCDd$ obteve 108 vitórias (96,4 %) e quatro derrotas.

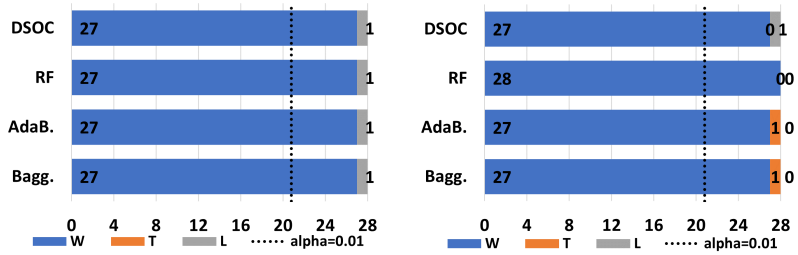
Uma análise estatística baseada nos testes de Friedman e Nemenyi (DEMSAR, 2006) com p-valor = 0,01 confirmou uma diferença significativa entre os métodos comparados. As Figuras 31 e 32 apresentam os rankings produzidos por meio do teste post-hoc de Nemenyi.

De maneira geral, pode-se observar que o método proposto é uma estratégia promissora para gerar um pool de classificadores complementares. A ideia de treinar os classificadores em subconjuntos de dados que representam subproblemas com diferentes níveis de dificuldade contribuiu para melhorar a precisão do pool. Além disso, um indutor de base mais instável melhorou significativamente os resultados de ambas as variantes do método proposto.

Como já mencionado, espera-se que o pool gerado possa contribuir para melhorar o desempenho do SMC com base em métodos de seleção dinâmica. Com isso em mente, o

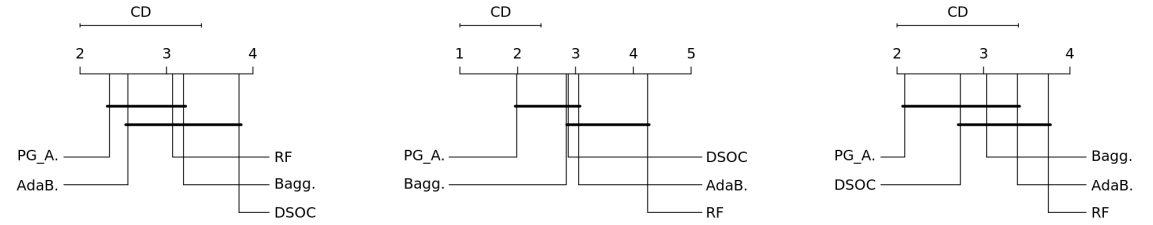


(a) $GPCDd$ - Perceptron (b) $GPCDa$ - Perceptron

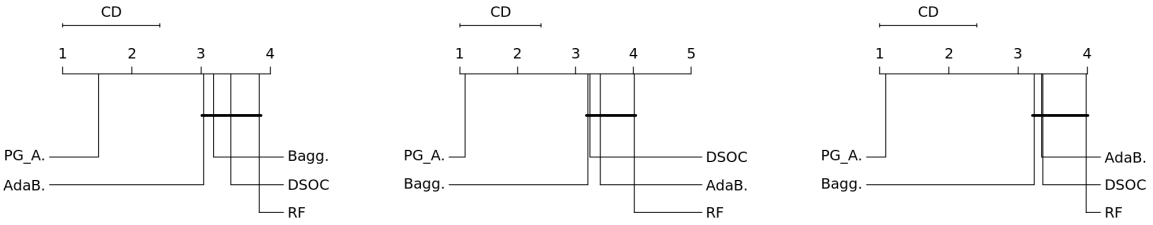


(c) $GPCDd$ - Decison tree (d) $GPCDa$ - Decison tree

Figura 30 – Ganho, empate e perda das variantes do método ($GPCDa$, $GPCDd$) quando comparado com Bagging, AdaBoost, Random Forest (RF) e DSOC. Perceptron e árvore de decisão como classificadores de base. Fusão baseada em Voto Majoritário.



(a) LCA - Perceptron (b) OLA - Perceptron (c) RANK - Perceptron

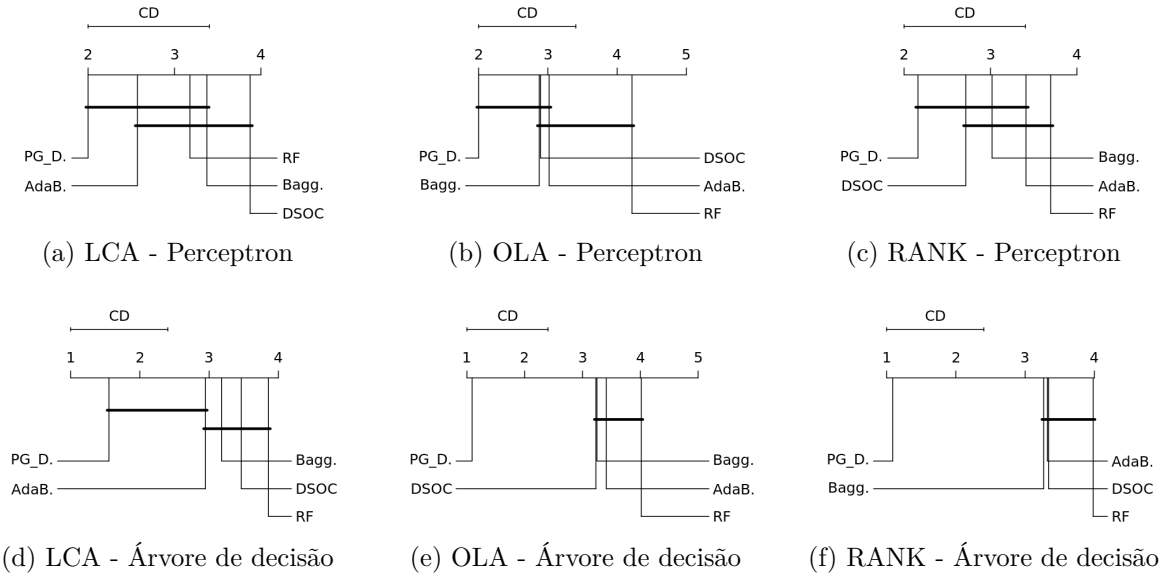


(d) LCA - Árvore de decisão (e) OLA - Árvore de decisão (f) RANK - Árvore de decisão

Figura 31 – Análise de Nemenyi comparando $GPCDa$ a outros métodos de geração de pool em diferentes abordagens SDCs.

método proposto foi avaliado no contexto de seleção dinâmica de classificadores (SDC) e seleção dinâmica de conjuntos de classificadores (SDE).

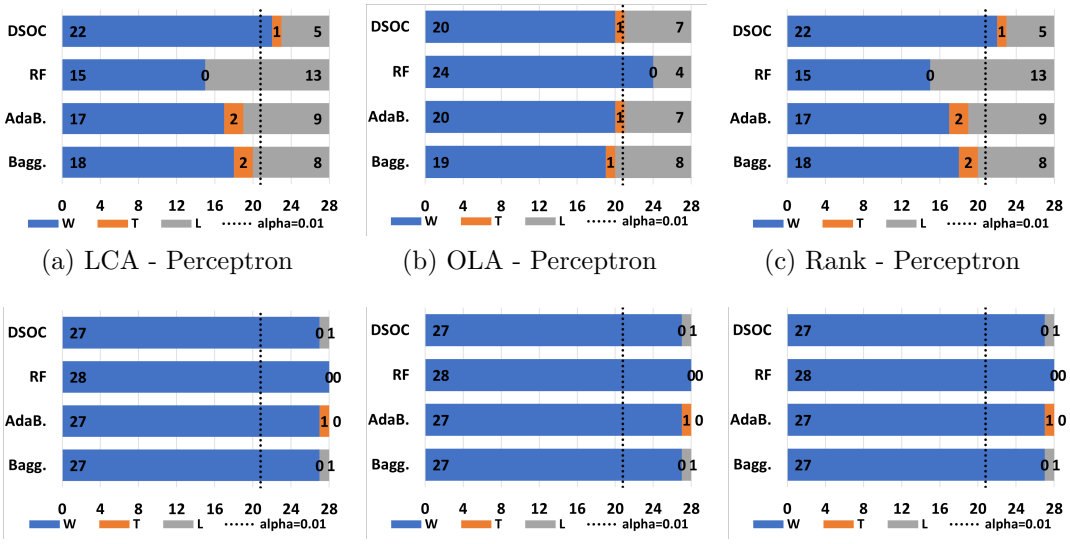
Dados os pools gerados, foram aplicados três métodos de seleção dinâmica de classificadores considerando um protocolo experimental com 20 repetições. Os resultados médios encontrados utilizando o Perceptron e Árvore de Decisão como indutores de base



(a) LCA - Perceptron (b) OLA - Perceptron (c) RANK - Perceptron

(d) LCA - Árvore de decisão (e) OLA - Árvore de decisão (f) RANK - Árvore de decisão

Figura 32 – Análise de Nemenyi comparando *GPCDd* a outros métodos de geração de pool em diferentes abordagens SDC



(a) LCA - Perceptron (b) OLA - Perceptron (c) Rank - Perceptron

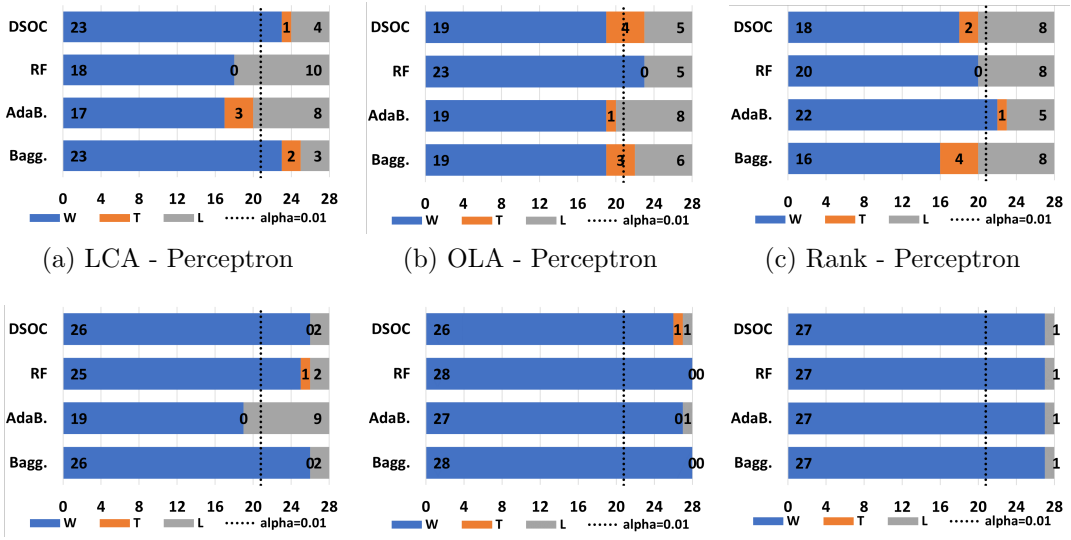
(d) LCA - Árvore de decisão (e) OLA - Árvore de decisão (f) Rank - Árvore de decisão

Figura 33 – A variante *GPCDa* considerando ambos os indutores de base (Perceptron e Árvore de Decisão) em comparação com outros métodos de geração de pools ao considerar métodos SDC.

foram resumidos nas Figuras 33 e 34 respectivamente.

Esses gráficos mostram o desempenho das duas variantes do método proposto em termos de G-E-P ao usar Perceptron e Árvore de decisão como indutores de base. Como esperado, o uso de Árvore de decisão é a melhor opção. Além disso, o *GPCDa* voltou a apresentar um melhor desempenho. Isso significa que a diversidade por si só não é suficiente para contribuir com o desempenho dos métodos SDCs.

Em resumo, foram realizados 336 experimentos considerando métodos SDCs para



(a) LCA - Perceptron (b) OLA - Perceptron (c) Rank - Perceptron

(d) LCA - Árvore de decisão (e) OLA - Árvore de decisão (f) Rank - Árvore de decisão

Figura 34 – A variante *GPCDd* considerando ambos os indutores de base (Perceptron e Árvore de Decisão) em comparação com outros métodos de geração de pools ao considerar métodos SDC.

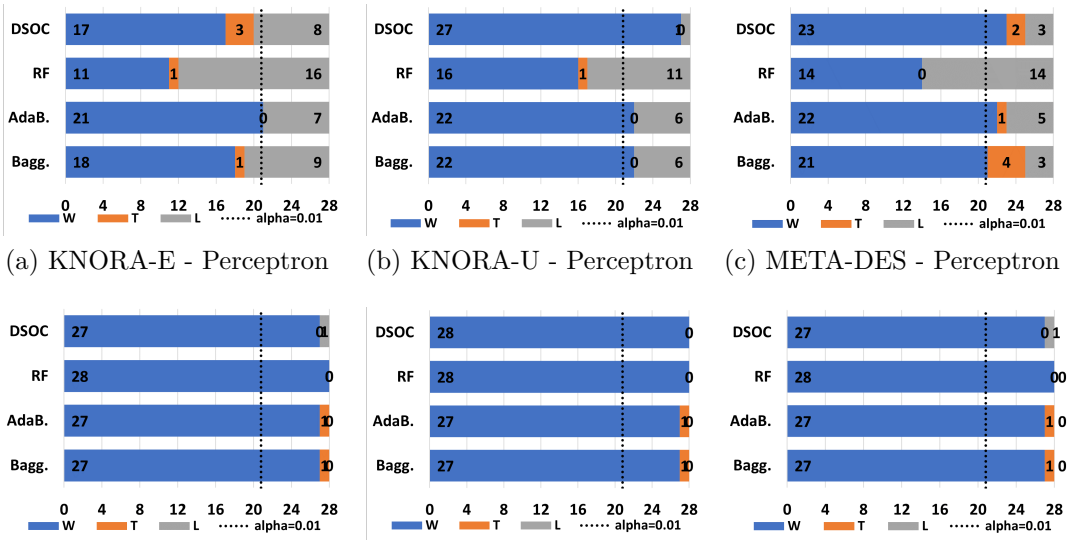
Tabela 27 – Métodos da literatura e os parâmetros usados.

Método	Parâmetros	Biblioteca/Versão	Ref.
Perceptron	max_iter=100, tol=1.0	scikit-learn 0.24.2	(PEDREGOSA et al., 2011)
Árvore de decisão	Padrão	scikit-learn 0.24.2	(PEDREGOSA et al., 2011)
BaggingClassifier	n_estimators=100, max_sample=0.5	scikit-learn 0.24.2	(PEDREGOSA et al., 2011)
AdaBoostClassifier	n_estimators=100, random_state=86	scikit-learn 0.24.2	(PEDREGOSA et al., 2011)
RandomForestClassifier	n_estimators=100, random_state=86	scikit-learn 0.24.2	(PEDREGOSA et al., 2011)
Métodos SDC	Padrão	DESlib 0.3	(CRUZ et al., 2018, 2020)
Métodos SDE	Padrão	DESlib 0.3	(CRUZ et al., 2018, 2020)
NSGAII	Padrão	deap 1.3.1	(FORTIN et al., 2012)
Medidas de complexidade	Padrão	ECoL 0.3.0	(LORENA et al., 2019)

cada variante proposta. Em geral, considerando o melhor classificador de base (Árvore de decisão), o *GPCDa* (Figura 33) obteve 315 vitórias (93,7 %), três empates e 18 derrotas, enquanto os melhores resultados com *GPCDd* (Figura 34) teve 314 vitórias (93,4 %), três empates e 19 derrotas. Em todos os experimentos, os métodos SDC foram executados usando os parâmetros *default* da biblioteca utilizada nos testes, conforme demonstrado na Tabela 27 (tamanho do pool = 100 e número de vizinhos = 7), mas usando um pool diferente gerado por Bagging, AdaBoost, RF, *GPCDa* ou *GPCDd*.

O impacto mais significativo foi observado nas abordagens que consideram a *GPCDa* dos classificadores para realizar a seleção dinâmica. Os métodos OLA e Rank para ambas as variantes (*GPCDa* e *GPCDd*) tiveram 109 (97,3 %) vitórias, um empate e duas derrotas.

Em outro conjunto de experimentos, foi avaliado o impacto do método proposto em SMC baseado em SDE. Os resultados médios considerando Perceptron e Árvore de Decisão como indutores de base foram resumidos e relacionados nas Figuras 35 e 36. Esses gráficos mostram o desempenho das duas variantes do método proposto em G-E-P.

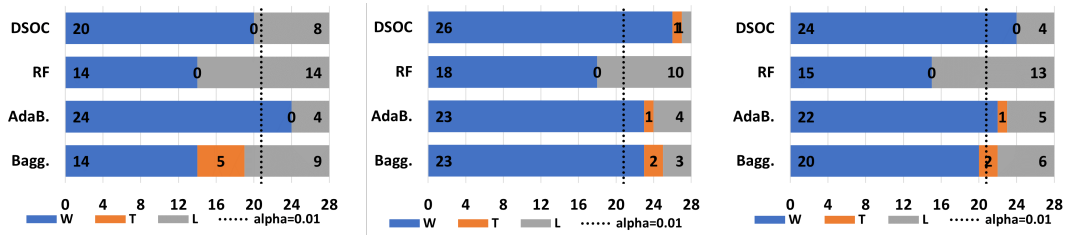


(a) KNORA-E - Perceptron (b) KNORA-U - Perceptron (c) META-DES - Perceptron

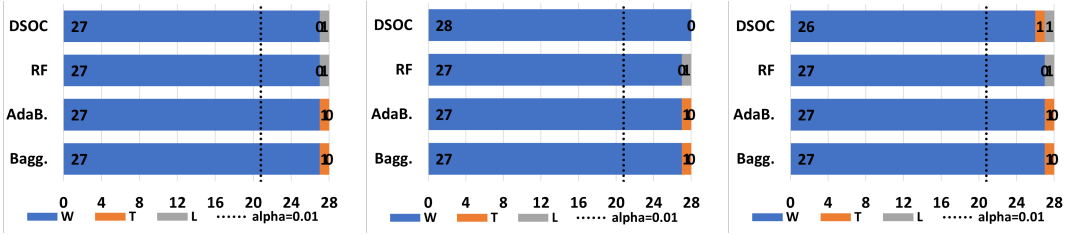
(d) KNORA-E - Árvore de decisão (e) KNORA-U - Árvore de decisão (f) META-DES - Árvore de decisão

Figura 35 – G-E-P da variante *GPCDa* contra outros métodos de geração de pool ao considerar os métodos DES.

Os métodos SDE foram executados usando os mesmos parâmetros (tamanho do pool = 100 e o números de vizinhos = 7), mas com diferentes pools gerados por Bagging, AdaBoost, RF, *GPCDa* ou *GPCDd*. Finalmente, *GPCDa* usando uma árvore de decisão é novamente a opção melhor.



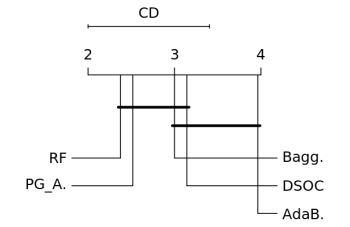
(a) KNORA-E - Perceptron (b) KNORA-U - Perceptron (c) META-DES - Perceptron



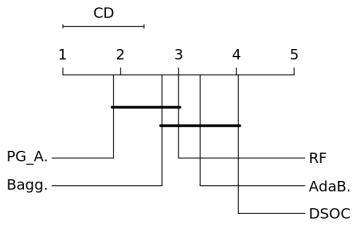
(d) KNORA-E - Árvore de decisão (e) KNORA-U - Árvore de decisão (f) META-DES - Árvore de decisão

Figura 36 – G-E-P da variante *GPCDd* contra outros métodos de geração de pool ao considerar os métodos DES.

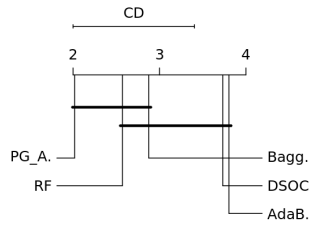
Uma análise estatística baseada nos testes de Friedman e Nemenyi (DEMSAR, 2006) com p-valor = 0,01 confirmou uma diferença significativa entre os métodos comparados. As



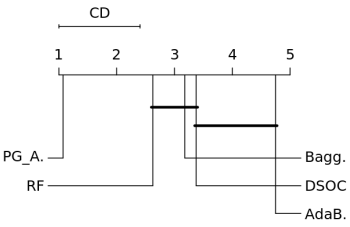
(a) KNORA-E - Perceptron



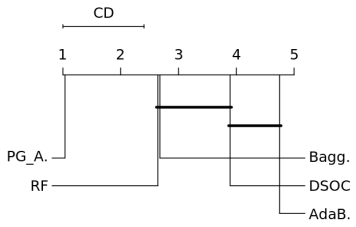
(b) KNORA-U - Perceptron



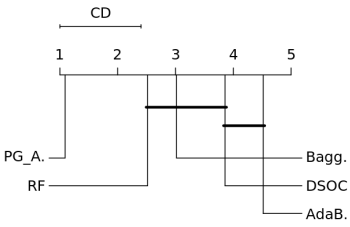
(c) META-DES- Perceptron



(d) KNORA-E - Árvore de decisão

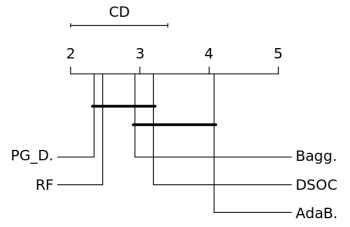


(e) KNORA-U - Árvore de decisão

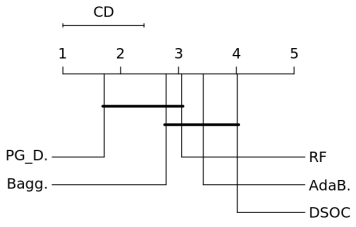


(f) META-DES- Árvore de decisão

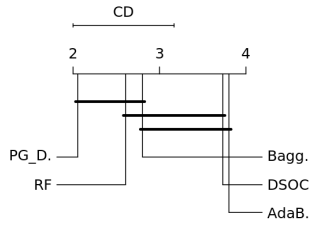
Figura 37 – Analise de Nemenyi da variante *GPCDa* contra outros métodos de geração de pool ao considerar os métodos DES.



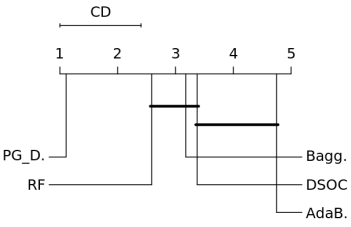
(a) KNORA-E - Perceptron



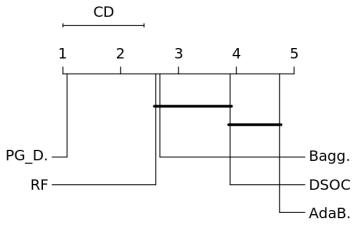
(b) KNORA-U - Perceptron



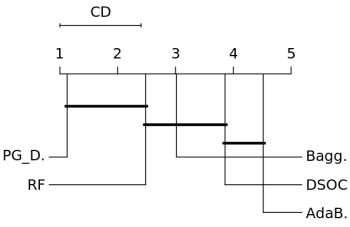
(c) META-DES - Perceptron



(d) KNORA-E - Árvore de decisão



(e) KNORA-U - Árvore de decisão



(f) META-DES - Árvore de decisão

Figura 38 – Analise de Nemenyi da variante *GPCDd* contra outros métodos de geração de pool ao considerar os métodos DES.

figuras 37 e 38 apresentam os rankings produzidos por meio do teste post-hoc de Nemenyi.

Pode-se observar um impacto significativo da variante *GPCDa* em todos os métodos SDE avaliados. O mais importante foi o observado no KNORA-U, onde foram observadas 109 (97,3%) vitórias, dois empates e uma derrota em 112 experimentos.

O método proposto usando a árvore de decisão como classificador base gerou pools superiores a todos os concorrentes. Os melhores resultados foram alcançados com a

abordagem $GPCDa$.

Com os resultados apresentados pode-se discutir as hipóteses 1, 2, 3, 4 deste trabalho, abaixo elencadas:

Hipótese 1: A escolha das medidas de complexidade é dependente do problema de classificação.

Hipótese 2: A combinação da diversidade obtida no espaço de complexidade do problema com a diversidade no espaço de decisão dos classificadores contribui positivamente na geração de *pool* de classificadores.

Hipótese 3: A amostragem baseada na dificuldade do problema, ou seja, orientada por medidas de complexidade permite gerar classificadores diversos, que quando combinados, melhoram o desempenho da tarefa de classificação.

Hipótese 4: O *pool* composto por classificadores treinados em subproblemas, representando diferentes níveis de dificuldade, melhora o desempenho dos sistemas de seleção dinâmica de classificadores.

Como demonstrado na primeira fase do método $GPCD$, cada problema de classificação possui características que os tornam únicos, consequentemente as MCs apresentaram resultados diferentes. O cálculo de cada medida é o retrato da disposição dos atributos e/ou instâncias no espaço. Em virtude disso, a Hipótese 1 foi validada.

Os resultados da segunda fase demostram que a combinação entre as medidas MCs com a MD contribuem positivamente para a tarefa de classificação. Os resultados apresentaram melhora em termos de acurácia em diversos problemas quando comparados a outros métodos da literatura. Portanto, assim como na primeira hipótese, a Hipótese 2 também se confirmou.

A Hipótese 3 também foi validada, pois mesmo com a medida Df na criação das amostras para compor o *pool*, as medidas de complexidade tiveram um papel fundamental. As Figuras 28 e 29 demonstraram as amostras em duas situações, na primeira (28a e 29a) as amostras estão concentradas no espaço. O método inicia com amostras criadas aleatoriamente, logo o resultado das figuras na situação (a) é muito próximo ao *Bagging*, os demais resultados podem ser observados na Tabela 19. Na combinação de classificadores o método $GPCDa_{pe}$ obteve ótimos resultados se igualando ao RF, mesmo com classificadores de bases diferentes (*Random Forest* sempre utiliza Árvore de Decisão). Já com o mesmo classificador de base para todas as técnicas GPs o $GPCDa_{ad}$ obteve 71% das vitórias.

O mesmo raciocínio da Hipótese 3 se aplica para a Hipótese 4 onde os Métodos SDEs apresentam melhor desempenho quando os classificadores são diversos. Se o *pool* apresentar diversidade, os SDEs conseguem selecionar mais de um modelo para rotular a instância de teste. Caso o *pool* não seja diverso, as técnicas SDEs acabam por selecionar poucos classificadores, como acontece no KNORA-E, diminuindo as chances das predições dos classificadores estarem corretas.

Para finalizar, com os resultados deste capítulo pode-se afirmar que os experimentos

realizados comprovam a eficácia do $GPCD$ quando comparados, em termos de acurácia e ao grau de efetividade, em relação a outros métodos testados nesta pesquisa.

6 Conclusão

Sabe-se que um *pool* de classificadores diverso, na maioria das vezes, aumenta a precisão em Sistemas de Múltiplos Classificadores, pois a probabilidade de se encontrar um classificador no conjunto com competência para rotular corretamente a instância de teste é maior. O grande desafio é criar um conjunto de classificadores capaz de cobrir adequadamente todo o problema de classificação, formando modelos complementares e ao mesmo tempo especialistas em uma região do problema.

Neste contexto, este trabalho apresentou uma nova abordagem para a geração de conjuntos de classificadores, baseando-se em medidas de complexidade (MC) e diversidade a nível de decisão dos modelos de classificação. O uso de meta-características de complexidade do problema é algo bastante difundido no Aprendizado de Máquina, torná-las uma ferramenta na construção de classificadores foi um dos principais objetivos dessa proposta.

Para cumprir esse objetivo, foi criado um método que utiliza as MCs junto a uma métrica de diversidade igualmente ponderadas em algoritmo de otimização. Buscou-se criar classificadores com níveis de complexidade diferentes e distribuí-los nesse espaço. A primeira hipótese levantada foi a possibilidade de uma MC representar mais adequadamente um problema de classificação conforme a metodologia desta proposta.

A partir da primeira hipótese, foi realizada uma análise das medidas que mais oscilavam (em valores) em um mesmo problema. Nesta análise foram criadas várias subamostras da base de dados e assim verificar o comportamento de cada métrica num mesmo *dataset*. Com isso, foi confirmado que a distribuição das instâncias e atributos de cada base de dados a torna única, logo o comportamento de cada MC também varia de acordo com essas características, validando a primeira hipótese.

A primeira análise seguiu como base para a geração do conjunto de classificadores e para o levantamento das demais hipóteses. A segunda hipótese leva em consideração o uso de uma segunda medida, a de diversidade na decisão dos classificadores. Como um dos principais objetivos foi distribuir os subproblemas no espaço, diferenciando-os uns dos outros, observou-se que havia a possibilidade de adicionar mais uma métrica. Assim, o uso de uma medida de diversidade contribuiu ainda mais para o espalhamento dos subproblemas no espaço.

As duas últimas hipóteses foram voltadas para avaliação do método proposto, sendo uma delas para as contribuições na combinação dos classificadores e a outra na seleção dinâmica de classificadores. Para confirmar ou refutar essas duas hipóteses, foi elaborado um protocolo experimental robusto, contendo 28 problemas de classificação e 20 replicações. A construção das subamostras ficou por conta de um algoritmo evolutivo, que além disto foi responsável por distribuir essas subamostras no espaço de complexidade e diversidade.

As medidas de complexidade escolhidas para direcionar a construção do *pool* foram as pertencentes as famílias de sobreposição de atributos e as medidas de vizinhança. Essas medidas foram selecionadas, entre outros motivos, devido as características dos problemas de classificação testados. Apesar dessas bases de dados possuírem características bem específicas, nenhuma delas contém um desbalanceamento de classes acentuado, o que tornam as métricas dessas famílias uma boa opção para o propósito do método.

Ao todo foram realizados 2240 experimentos envolvendo apenas o método proposto. O *GPCD* (Geração de *Pool* de Classificadores fundamentado na Diversidade de dois níveis), foi elaborado com duas variações, cujos objetivos foram encontrar a melhor geração de classificadores dentro das replicações do algoritmo evolutivo. Cada variação criou dois conjuntos de classificadores para cada um dos 28 problemas de classificação, utilizando 2 classificadores de base diferentes.

O *GPCD* foi melhor ou igual a outros métodos de geração de classificadores em muitos casos. Os conjuntos de classificadores gerados pelo *GPCD* trouxeram ganhos significativos aos métodos de seleção dinâmica de um único classificador e na seleção dinâmica de conjuntos de classificadores. Isso significa que os modelos gerados na primeira etapa de SMC possuem um papel fundamental no que diz respeito a acurácia final dos métodos de seleção dinâmica.

Os testes estatísticos comparativos demonstraram que o *GPCD* além de possuir um bom resultado no geral, se difere significativamente com melhores resultados ou se iguala as outras técnicas em diversas situações. O número de vitórias do *GPCD* em comparação aos outros métodos de geração de *pool* chegou, em algumas situações, a mais de 70% das 28 bases de dados testadas. Dessa maneira foi possível confirmar as duas últimas hipóteses deste trabalho.

A Geração de *pool* de classificadores proposta neste trabalho complementa os demais métodos da literatura, quanto a aplicabilidade no contexto do Conjunto de Classificadores, Aprendizado de Máquina e Reconhecimento de Padrões. Portanto o *GPCD* é uma alternativa viável e promissora, com a possibilidade de novas variações e melhorias de forma geral.

6.1 Trabalhos Futuros

Este trabalho propôs um novo método de geração de conjunto de classificadores que utiliza como base a complexidade do problema e a diversidade a nível de decisão dos classificadores.

Neste sentido, como sugestão de trabalho futuro está a aplicação de outras medidas de complexidade e de diversidade. No que tange a métrica de diversidade, uma que avalie o conjunto de classificadores por completo e não pareado como no método apresentado, por exemplo o coeficiente Kappa (GALTON, 1892).

Outra alternativa para a continuação desta pesquisa seria tornar a primeira etapa do método dinâmica. Visto que no $GPCD$ as medidas de complexidade são encontradas na primeira etapa do método e permanecem durante toda a execução da segunda etapa. Entretanto a disposição das instâncias do problema pode variar durante o método e consequentemente pode alterar a medida de complexidade escolhida. Por último, há de se investigar o desempenho do método proposto para bases de dados desbalanceadas.

Referências

ALCALÁ-FDEZ, Jesús; FERNÁNDEZ, Alberto; LUENGO, Julián; DERRAC, Joaquín; GARCÍA, Salvador; SÁNCHEZ, Luciano; HERRERA, Francisco. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Comp.*, Citeseer, v. 17, 2011. Citado na página 89.

BARELLA, Victor H; GARCIA, Luís PF; SOUTO, Marcilio P de; LORENA, Ana C; CARVALHO, Andre De. Data complexity measures for imbalanced classification tasks. In: IEEE. *Int. Joint Conf. on Neural Networks (IJCNN)*. [S.l.], 2018. p. 1–8. Citado na página 32.

BAUER, Eric; KOHAVI, Ron. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, Springer, v. 36, n. 1-2, p. 105–139, 1999. Citado 2 vezes nas páginas 59 e 68.

BREIMAN, Leo. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996. Citado 7 vezes nas páginas 22, 49, 53, 54, 55, 68 e 70.

BREIMAN, Leo. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Citado 4 vezes nas páginas 22, 59, 68 e 99.

BREIMAN, Leo; FRIEDMAN, Jerome; STONE, Charles J; OLSHEN, Richard A. *Classification and regression trees*. [S.l.]: CRC press, 1984. Citado na página 95.

BRITTO, Alceu S.; SABOURIN, Robert; OLIVEIRA, Luiz E.S. Dynamic selection of classifiers—A comprehensive review. *Pattern Recognit.*, Pergamon, v. 47, n. 11, p. 3665–3680, nov 2014. ISSN 0031-3203. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0031320314001885>. Citado na página 44.

BRUN, André L; JR., Alceu S. Britto; OLIVEIRA, Luiz S; ENEMBRECK, Fabricio; SABOURIN, Robert. Contribution of data complexity features on dynamic classifier selection. *Proc. Int. Jt. Conf. Neural Networks*, v. 2016-Octob, n. July, p. 4396–4403, 2016. Citado na página 90.

BRUN, André L; JR., Alceu S. Britto; OLIVEIRA, Luiz S; ENEMBRECK, Fabricio; SABOURIN, Robert. A framework for dynamic classifier selection oriented by the classification problem difficulty. *Pattern Recognition*, Elsevier, v. 76, p. 175–190, 2018. Citado 10 vezes nas páginas 22, 23, 51, 66, 67, 68, 75, 89, 90 e 99.

CANO, José-Ramón. Analysis of data complexity measures for classification. *Expert Syst. Appl.*, v. 40, 2013. Citado na página 22.

CAVALCANTI, George DC; OLIVEIRA, Luiz S; MOURA, Thiago JM; CARVALHO, Guilherme V. Combining diversity measures for ensemble pruning. *Pattern Recognition Letters*, Elsevier, v. 74, p. 38–45, 2016. Citado 2 vezes nas páginas 35 e 51.

CORNE, David W; KNOWLES, Joshua D; OATES, Martin J. The pareto envelope-based selection algorithm for multiobjective optimization. In: SPRINGER. *International*

conference on parallel problem solving from nature. [S.l.], 2000. p. 839–848. Citado na página 44.

CRISTIANINI, Nello; SHAWE-TAYLOR, John et al. *An introduction to support vector machines and other kernel-based learning methods*. [S.l.]: Cambridge university press, 2000. Citado na página 34.

CRUZ, Rafael M.O.; SABOURIN, Robert; CAVALCANTI, George D.C. META-DES.Oracle: Meta-learning and feature selection for dynamic ensemble selection. *Inf. Fusion*, Elsevier B.V., v. 38, p. 84–103, 2017. ISSN 15662535. Citado 4 vezes nas páginas 22, 44, 45 e 95.

CRUZ, Rafael M.O.; SABOURIN, Robert; CAVALCANTI, George D.C.; Ing Ren, Tsang. META-DES: A dynamic ensemble selection framework using meta-learning. *Pattern Recognit.*, Elsevier, v. 48, n. 5, p. 1925–1935, 2015. ISSN 00313203. Citado 4 vezes nas páginas 49, 50, 89 e 90.

CRUZ, Rafael M. O.; HAFEMANN, Luiz G.; SABOURIN, Robert; CAVALCANTI, George D. C. DESlib: A Dynamic ensemble selection library in Python. *arXiv preprint arXiv:1802.04967*, 2018, 2020. Citado 2 vezes nas páginas 90 e 110.

DARWIN, Charles. *A origem das espécies*. [S.l.]: Lelo & Irmão, 2009. Citado na página 37.

DEB, Kalyanmoy; PRATAP, Amrit; AGARWAL, Sameer; MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, v. 6, n. 2, p. 182–197, 2002. Citado 4 vezes nas páginas 41, 42, 43 e 44.

DEMSAR, Janez. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, JMLR. org, v. 7, p. 1–30, 2006. Citado 2 vezes nas páginas 107 e 111.

DIETTERICH, Thomas G. Ensemble methods in machine learning. In: SPRINGER. *International workshop on multiple classifier systems*. [S.l.], 2000. p. 1–15. Citado 2 vezes nas páginas 21 e 53.

DUA, Dheeru; GRAFF, Casey. *UCI Machine Learning Repository*. 2019. Disponível em: <http://archive.ics.uci.edu/ml>. Citado na página 89.

DUIN, R.P.W.; P., Juszczak; P., Paclik; E., Pekalska; D., De Ridder; D.M.J., Tax; S., Verzakov. Prtools, a matlab toolbox for pattern recog. 2004. Disponível em: <http://www.prtools.org>. Citado 2 vezes nas páginas 89 e 90.

EIBEN, Agoston E.; SMITH, Jim E. What Is an Evolutionary Algorithm? In: . [S.l.: s.n.], 2015. ISBN 978-3-642-07285-7. Citado na página 37.

FERNÁNDEZ-DELGADO, Manuel; CERNADAS, Eva; BARRO, Senén; AMORIM, Dinani. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, JMLR. org, v. 15, n. 1, p. 3133–3181, 2014. Citado na página 53.

FLORES, Julia M.; GáMEZ, José A.; MARTíNEZ, Ana M. Domains of competence of the semi-naive bayesian network classifiers. *Information Sciences*, v. 260, p. 120–148, 2014. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025513007226>. Citado na página 51.

FONSECA, Carlos M.; FLEMING, Peter J. Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization. In: *Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. p. 416–423. ISBN 1-55860-299-2. Citado na página 44.

FORTIN, Félix-Antoine; De Rainville, François-Michel; GARDNER, Marc-André; PARIZEAU, Marc; GAGNÉ, Christian. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, v. 13, p. 2171–2175, jul 2012. Citado na página 110.

FRÉNAY, Benoît; VERLEYSEN, Michel. Classification in the presence of label noise: A survey. *IEEE Trans. Neural Networks Learn. Syst.*, 2014. ISSN 21622388. Citado na página 51.

FREUND, Y; SCHAPIRE, R. E. Experiments with a New Boosting Algorithm. *Int. Conf. Mach. Learn.*, p. 148–156, 1996. Citado 4 vezes nas páginas 53, 55, 57 e 68.

FREUND, Yoav; SCHAPIRE, Robert E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, Elsevier, v. 55, n. 1, p. 119–139, 1997. Citado na página 22.

GALTON, Francis. *Finger prints*. [S.l.]: Macmillan and Company, 1892. Citado na página 116.

GARCIA, Luís P.F.; CARVALHO, André C.P.L.F. de; LORENA, Ana C. Effect of label noise in the complexity of classification problems. *Neurocomputing*, 2015. ISSN 18728286. Citado 2 vezes nas páginas 22 e 51.

GHOSH, Ashish; DAS, Mrinal Kanti. Non-dominated Rank based Sorting Genetic Algorithms. *Fundam. Informaticae*, v. 83, p. 231–252, 2008. Disponível em: <https://www.isical.ac.in/{~}ash/Mrinal-fi-08.p>. Citado na página 40.

GIACINTO, Giorgio; ROLI, Fabio. Methods for dynamic classifier selection. In: IEEE. *Proceedings 10th International Conference on Image Analysis and Processing*. [S.l.], 1999. p. 659–664. Citado na página 46.

GIACINTO, Giorgio; ROLI, Fabio. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, v. 19, n. 9, p. 699–707, 2001. ISSN 0262-8856. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0262885601000452>. Citado na página 36.

GIACINTO, Giorgio; ROLI, Fabio. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, v. 19, n. 9-10, p. 699–707, 2001. ISSN 02628856. Citado na página 46.

GIACINTO, Giorgio; ROLI, Fabio; FUMERA, Giorgio. Selection of classifiers based on multiple classifier behaviour. In: SPRINGER. *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. [S.l.], 2000. p. 87–93. Citado na página 65.

GOLDBERG, David E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201157675. Citado na página 42.

GRAHAM, Ronald L; HELL, Pavol. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, IEEE, v. 7, n. 1, p. 43–57, 1985. Citado na página 31.

HAJELA, Prabhat; LIN, C-Y. Genetic search strategies in multicriterion optimal design. *Structural optimization*, Springer, v. 4, n. 2, p. 99–107, 1992. Citado na página 44.

HERNÁNDEZ-REYES, Edith; CARRASCO-OCHOA, J. A.; MARTÍNEZ-TRINIDAD, J. Fco. Classifier Selection Based on Data Complexity Measures. n. 1, p. 586–592, 2005. Citado na página 32.

HO, Tin Kam. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 20, n. 8, p. 832–844, 1998. ISSN 01628828. Citado 5 vezes nas páginas 22, 53, 57, 58 e 68.

HO, Tin Kam; BASU, Mitra. Complexity measures of supervised classification problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 24, n. 3, p. 289–300, 2002. ISSN 01628828. Citado 6 vezes nas páginas 22, 28, 31, 32, 33 e 34.

HO, Tin Kam; BASU, Mitra; LAW, Martin Hiu. Measures of geometrical complexity in classification problems. In: *Data complexity in pat.recog.* [S.l.]: Springer, 2006. p. 1–23. Citado 3 vezes nas páginas 26, 33 e 35.

HOLLAND, J.H.; GOLDBERG, D. Genetic algorithms in search, optimization and machine learning. *Massachusetts: Addison-Wesley*, 1989. Citado 2 vezes nas páginas 39 e 40.

HORN, Jeffrey; NAFPLIOTIS, Nicholas; GOLDBERG, David E. A niched pareto genetic algorithm for multiobjective optimization. In: *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. [S.l.: s.n.], 1994. v. 1, p. 82–87. Citado na página 44.

HU, Ruihan; ZHOU, Songbin; LIU, Yisen; TANG, Zhiri. Margin-Based Pareto Ensemble Pruning : An Ensemble Pruning Algorithm That Learns to Search Optimized Ensembles. v. 2019, 2019. Citado 3 vezes nas páginas 65, 66 e 68.

JAIN, Himanshu; DEB, Kalyanmoy. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach. *IEEE Trans. Evol. Comput.*, IEEE, v. 18, n. 4, p. 602–622, 2014. ISSN 1089-778X. Disponível em: <http://ieeexplore.ieee.org/document/6595567/>. Citado na página 44.

JUTTEN, Christian. *The enhanced learning for evolutive neural architectures project*. 2002. Citado na página 89.

KAMATH, Vidya; YEATMAN, Timothy J.; ESCHRICH, Steven A. Toward a measure of classification complexity in gene expression signatures. In: . [S.l.: s.n.], 2009. Citado na página 51.

KING, Ross D.; FENG, Cao; SUTHERLAND, Alistair. Statlog: comparison of classification algorithms on large real-world problems. *Applied Art. Intelligence an Int. Journal*, Taylor & Francis, v. 9, n. 3, p. 289–333, 1995. Citado na página 89.

KO, Albert H.R.; SABOURIN, Robert; JR., Alceu S. Britto. From dynamic classifier selection to dynamic ensemble selection. *Pattern Recognition*, v. 41, n. 5, p. 1718 – 1731, 2008. ISSN 0031-3203. Citado 2 vezes nas páginas 47 e 48.

KOHAVI, Ron; WOLPERT, David H et al. Bias plus variance decomposition for zero-one loss functions. In: *ICML*. [S.l.: s.n.], 1996. v. 96, p. 275–83. Citado na página 37.

KONAK, Abdullah; COIT, David W; SMITH, Alice E. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, Elsevier, v. 91, n. 9, p. 992–1007, 2006. Citado na página 37.

KUNCHEVA, L. Ludmila kuncheva collection lkc. *Available: http://pages.bangor.ac.uk/mas00a/activi*, 2004. Citado na página 89.

KUNCHEVA, Ludmila I. A theoretical study on six classifier fusion strategies. *IEEE Trans. Pattern Anal. Mach. Intell.*, IEEE, v. 24, p. 281–286, 2002. Citado 2 vezes nas páginas 53 e 62.

KUNCHEVA, Ludmila I; RODRIGUEZ, Juan José et al. Classifier ensembles with a random linear oracle. *IEEE Trans. Knowl. Data Eng.*, v. 19, n. 4, p. 500–508, 2007. Citado na página 46.

KUNCHEVA, Ludmila I; WHITAKER, Christopher J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, Springer, v. 51, n. 2, p. 181–207, 2003. Citado 2 vezes nas páginas 36 e 68.

KUNCHEVA, Ludmila I; WHITAKER, Christopher J; SHIPP, Catherine A; DUIN, Robert PW. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis & Applications*, Springer, v. 6, n. 1, p. 22–31, 2003. Citado na página 36.

LACERDA, Estéfane GM de; CARVALHO, ACPLF De. Introdução aos algoritmos genéticos. *Sistemas inteligentes: aplicaçoes a recursos hıdricos e ciências ambientais*, v. 1, p. 99–148, 1999. Citado 2 vezes nas páginas 38 e 94.

L'ECUYER, Pierre; SIMARD, Richard. Testu01: Ac library for empirical testing of random number generators. *ACM Transactions on Mathematical Software (TOMS)*, ACM, v. 33, n. 4, p. 22, 2007. Citado na página 70.

LEYVA, Enrique; GONZÁLEZ, Antonio; PEREZ, Raul. A set of complexity measures designed for applying meta-learning to instance selection. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 27, n. 2, p. 354–367, 2014. Citado na página 33.

LI, Shijin; WU, Hao; WAN, Dingsheng; ZHU, Jiali. An effective feature selection method for hyperspectral image classification based on genetic algorithm and support vector machine. *Knowledge-Based Systems*, v. 24, n. 1, p. 40–48, 2011. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705110001097>. Citado na página 51.

LILEIKYTE, Rasa; TELKSNYS, Laimutis. Quality estimation methodology of speech recognition features/snekos signalu atpazinimo pozymiu kokybes matas. *Elektronika ir Elektrotechnika*, Kaunas University of Technology, Faculty of Telecommunications and Electronics, p. 113–117, 2011. Citado na página 31.

LOONEY, Stephen W. A statistical technique for comparing the accuracies of several classifiers. *Pattern Recognition Letters*, Elsevier, v. 8, n. 1, p. 5–9, 1988. Citado na página 37.

LORENA, Ana C.; COSTA, Ivan G.; SPOLAÔR, Newton; De Souto, Marcilio C.P. Analysis of complexity indices for classification problems: Cancer gene expression data. *Neurocomputing*, 2012. ISSN 18728286. Citado 2 vezes nas páginas 22 e 51.

LORENA, Ana C.; GARCIA, Luís P. F.; LEHMANN, Jens; SOUTO, Marcilio C. P.; HO, Tin Kam. How complex is your classification problem? a survey on measuring classification complexity. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 52, n. 5, set. 2019. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3347711>. Citado 12 vezes nas páginas 22, 26, 27, 28, 29, 30, 31, 32, 33, 34, 91 e 110.

LU, Haiming; YEN, Gary G. Rank-density-based multiobjective genetic algorithm and benchmark test function study. *IEEE Transactions on Evolutionary Computation*, v. 7, n. 4, p. 325–343, Aug 2003. ISSN 1089-778X. Citado na página 44.

LUENGO, Julián; HERRERA, Francisco. An automatic extraction method of the domains of competence for learning classifiers using data complexity measures. *Knowl Inf Syst*, v. 42, p. 147–180, 2015. Disponível em: <http://sci2s.ugr.es/DC-automatic-method.> Citado 2 vezes nas páginas 22 e 51.

MACIà, Núria; ORRIOLS-PUIG, Albert; BERNADó-MANSILLA, Ester. In search of targeted-complexity problems. In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: Association for Computing Machinery, 2010. (GECCO '10), p. 1055–1062. ISBN 9781450300728. Disponível em: <https://doi.org/10.1145/1830483.1830674>. Citado 2 vezes nas páginas 22 e 75.

MALINA, Witold. Two-parameter fisher criterion. *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, v. 31, n. 4, p. 629–636, Aug 2001. ISSN 1083-4419. Citado na página 27.

MATSUMOTO, Makoto; NISHIMURA, Takuji. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, ACM, v. 8, n. 1, p. 3–30, 1998. Citado na página 70.

MOLLINEDA, Ramón A; SÁNCHEZ, J. Salvador; SOTOCA, José M. *Data Characterization for Effective Prototype Selection*. [S.l.], 2005. 27–34 p. Disponível em: <http://sci2s.ugr.es/keel/pdf/specific/congreso/LNCS05bMollineda.pdf>. Citado 4 vezes nas páginas 22, 27, 32 e 51.

MURATA, Tadahiko; ISHIBUCHI, Hisao. Moga: Multi-objective genetic algorithms. In: *IEEE international conference on evolutionary computation*. [S.l.: s.n.], 1995. v. 1, p. 289–294. Citado na página 44.

OPITZ, David; MACLIN, Richard. Popular ensemble methods: An empirical study. *Journal of art. intelligence research*, v. 11, p. 169–198, 1999. Citado na página 53.

ORRIOLS-PUIG, Albert; MACÌ, Núria; HO, Tin Kam. Documentation for the Data Complexity Library in C++. *Universitat Ramon Llull, La Salle*, v. 196, p. 1–40, 2010. Citado 6 vezes nas páginas 22, 27, 29, 30, 32 e 34.

PARTRIDGE, Derek; KRZANOWSKI, Wojtek. Software diversity: practical statistics for its measurement and exploitation. *Information and software technology*, Elsevier, v. 39, n. 10, p. 707–717, 1997. Citado na página 44.

PEARSON, Karl. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, Taylor & Francis, v. 2, n. 11, p. 559–572, 1901. Citado na página 61.

PEDREGOSA, Fabian; VAROQUAUX, Gaël; GRAMFORT, Alexandre; MICHEL, Vincent; THIRION, Bertrand; GRISEL, Olivier; BLONDEL, Mathieu; PRETTENHOFER, Peter; WEISS, Ron; DUBOURG, Vincent et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado 3 vezes nas páginas 81, 95 e 110.

PEREIRA, Marcelo; BRITTO, Alceu; OLIVEIRA, Luiz; SABOURIN, Robert. Dynamic ensemble selection by k-nearest local oracles with discrimination index. In: IEEE. *2018 IEEE 30th International conference on tools with artificial intelligence (ICTAI)*. [S.l.], 2018. p. 765–771. Citado na página 89.

QUINLAN, Ross J. Bagging, boosting, and c4.5. In: *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*. [S.l.]: AAAI Press, 1996. p. 725–730. Citado na página 59.

REN, Ye; ZHANG, Le; SUGANTHAN, Ponnuthurai N. Ensemble classification and regression-recent developments, applications and future directions. *IEEE Computational intelligence magazine*, IEEE, v. 11, n. 1, p. 41–53, 2016. Citado na página 21.

RODRIGUEZ, Juan J.; KUNCHEVA, Ludmila I.; ALONSO, Carlos J. Rotation forest: A new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 28, n. 10, p. 1619–1630, 2006. Citado 4 vezes nas páginas 22, 60, 61 e 68.

ROKACH, Lior. Ensemble-based classifiers. *Artificial Intelligence Review*, Springer, v. 33, n. 1-2, p. 1–39, 2010. Citado na página 21.

ROSA, Thatiane O.; LUZ, Hellen S. Conceitos básicos de algoritmos genéticos: Teoria e prática. *XI Encontro de Estudantes de Informática do Tocantins*, p. 27–37, 2009. Citado na página 38.

ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 95.

ROY, Anandarup; CRUZ, Rafael M.O.; SABOURIN, Robert; CAVALCANTI, George D.C. Meta-learning recommendation of default size of classifier pool for META-DES. *Neurocomputing*, v. 216, p. 351–362, 2016. Citado na página 22.

RUTA, Dymitr; GABRYS, Bogdan. Classifier selection for majority voting. *Information fusion*, Elsevier, v. 6, n. 1, p. 63–81, 2005. Citado na página 94.

SÁNCHEZ, José Salvador; MOLLINEDA, Ramón Alberto; SOTOCA, José Martínez. An analysis of how training data complexity affects the nearest neighbor classifiers. *Pattern Anal. Appl.*, Springer-Verlag, v. 10, n. 3, p. 189–201, jul 2007. ISSN 1433-7541. Disponível em: <http://link.springer.com/10.1007/s10044-007-0061-2>. Citado na página 51.

SANTANA, Alixandre; SOARES, Rodrigo G.F.; CANUTO, Anne M.P.; SOUTO, Marcilio C.P. de. A dynamic classifier selection method to build ensembles using accuracy and diversity. In: *2006 Ninth Brazilian Symposium on Neural Networks (SBRN'06)*. [S.l.: s.n.], 2006. p. 36–41. Citado na página 22.

SANTOS, Eulanda M. Dos; SABOURIN, Robert; MAUPIN, Patrick. Overfitting cautious selection of classifier ensembles with genetic algorithms. *Information Fusion*, v. 10, n. 2, p. 150–162, 2009. ISSN 1566-2535. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1566253508000730>. Citado na página 94.

SCHAFFER, James. D. Multiple objective optimization with vector evaluated genetic algorithms. In: *ICGA*. [S.l.: s.n.], 1985. Citado 2 vezes nas páginas 41 e 44.

SKALAK, David B. The sources of increased accuracy for two proposed boosting algorithms. In: CITESEER. *Proc. American Association for Artificial Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop*. [S.l.], 1996. v. 1129, p. 1133. Citado na página 36.

SKURICHINA, Marina; DUIN, Robert P.W. Bagging for linear classifiers. *Pattern Recognition*, v. 31, n. 7, p. 909–930, 1998. ISSN 0031-3203. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0031320397001106>. Citado na página 95.

SMITH, Michael R; MARTINEZ, Tony; GIRAUD-CARRIER, Christophe. An instance level analysis of data complexity. *Machine learning*, Springer, v. 95, n. 2, p. 225–256, 2014. Citado na página 64.

SMITS, Paul C. Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection. *IEEE Transactions on Geoscience and Remote Sensing*, IEEE, v. 40, n. 4, p. 801–813, 2002. Citado na página 46.

SOUZA, Mariana A; CAVALCANTI, George DC; CRUZ, Rafael MO; SABOURIN, Robert. On the characterization of the oracle for dynamic classifier selection. In: IEEE. *Int. Joint Conf. on Neural Networks (IJCNN)*. [S.l.], 2017. p. 332–339. Citado 3 vezes nas páginas 62, 63 e 68.

SOUZA, Mariana A.; CAVALCANTI, George D. C.; CRUZ, Rafael M. O.; SABOURIN, Robert. Online local pool generation for dynamic classifier selection: an extended version. *CoRR*, abs/1809.01628, 2018. Disponível em: <http://arxiv.org/abs/1809.01628>. Citado 4 vezes nas páginas 64, 65, 68 e 89.

SRINIVAS, Nidamarthi; DEB, Kalyanmoy. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Computation*, MIT Press, v. 2, n. 3, p. 221–248, 1994. Citado 3 vezes nas páginas 41, 42 e 44.

SUN, Yanan; XUE, Bing; ZHANG, Mengjie; YEN, Gary G; LV, Jiancheng. Automatically designing cnn architectures using the genetic algorithm for image classification. *IEEE Transactions on Cybernetics*, v. 50, n. 9, p. 3840–3854, 2020. Citado na página 51.

VALENTINI, Giorgio. An experimental bias-variance analysis of svm ensembles based on resampling techniques. *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, IEEE, v. 35, n. 6, p. 1252–1271, 2005. Citado 2 vezes nas páginas 89 e 90.

WEBB, Geoffrey I. Multiboosting: A technique for combining boosting and wagging. *Machine learning*, Springer, v. 40, n. 2, p. 159–196, 2000. Citado 2 vezes nas páginas 60 e 68.

WEISSTEIN, Eric et al. *Wolfram mathworld*. 2007. Citado na página 27.

WOLOSZYNSKI, Tomasz; KURZYNSKI, Marek. A probabilistic model of classifier competence for dynamic ensemble selection. *Pattern Recognition*, Elsevier, v. 44, n. 10-11, p. 2656–2668, 2011. Citado na página 65.

WOODS, Kevin; KEGELMEYER, W. Philip; BOWYER, Kevin. Combination of multiple classifiers using local accuracy estimates. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 19, n. 4, p. 405–410, 1997. Citado 2 vezes nas páginas 45 e 46.

WOŹNIAK, Michał; GRAÑA, Manuel; CORCHADO, Emilio. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, Elsevier, v. 16, p. 3–17, 2014. Citado na página 21.

YULE, George Udny. Vii. on the association of attributes in statistics: with illustrations from the material of the childhood society, &c. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, The Royal Society London, v. 194, n. 252-261, p. 257–319, 1900. Citado na página 36.

ZHANG, Chun Xia; ZHANG, Jiang She. RotBoost: A technique for combining Rotation Forest and AdaBoost. *Pattern Recognit. Lett.*, v. 29, n. 10, p. 1524–1536, 2008. ISSN 01678655. Citado 2 vezes nas páginas 62 e 68.

ZITZLER, Eckart; LAUMANNS, Marco; THIELE, Lothar. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische, v. 103, 2001. Citado 3 vezes nas páginas 40, 41 e 44.

ZITZLER, Eckart; THIELE, Lothar. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, IEEE, v. 3, n. 4, p. 257–271, 1999. Citado 2 vezes nas páginas 41 e 44.