DIOGO STEINKE DECONTO

CLASSIFICAÇÃO AUTOMÁTICA DE PARTITURAS POR NÍVEL DE DIFICULDADE

Curitiba - PR, Brasil

2021

DIOGO STEINKE DECONTO

CLASSIFICAÇÃO AUTOMÁTICA DE PARTITURAS POR NÍVEL DE DIFICULDADE

Projeto de Dissertação de Mestrado apresentado ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de mestre em Informática.

Pontifícia Universidade Católica do Paraná - PUCPR

Programa de Pós-Graduação em Informática - PPGIa

Orientador: CARLOS NASCIMENTO SILLA JUNIOR

Curitiba - PR, Brasil

2021

DIOGO STEINKE DECONTO
CLASSIFICAÇÃO AUTOMÁTICA DE PARTITURAS POR NÍVEL DE DIFICULDADE
DIOGO STEINKE DECONTO. – Curitiba - PR, Brasil, 2021-
78 p. : il. ; 30 cm.

Orientador: CARLOS NASCIMENTO SILLA JUNIOR

Dissertação – Mestrado
Pontifícia Universidade Católica do Paraná - PUCPR
Programa de Pós-Graduação em Informática - PPGIa, 2021.

1. Classificação de partituras musicais. 2. Música. 3. Nível de dificuldade. I. Carlos Nascimento Silla Junior.

II. Pontifícia Universidade Católica III. Programa de Pós-Graduação em Informática IV. Título

DIOGO STEINKE DECONTO

CLASSIFICAÇÃO AUTOMÁTICA DE PARTITURAS POR NÍVEL DE DIFICULDADE

Projeto de Dissertação de Mestrado apresentado ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de mestre em Informática.

Trabalho aprovado. Curitiba - PR, Brasil, 29 de janeiro de 2021:

CARLOS NASCIMENTO SILLA JUNIOR
Orientador(a)

Professor(a)
Júlio César Nievola

Professor(a)
Joezer de Souza Mendonça

Professor
Rodrigo Schramm

Curitiba - PR, Brasil
2021

Agradecimentos

A elaboração deste trabalho não teria sido possível sem a colaboração, estímulo e empenho de diversas pessoas. Gostaria de expressar toda a minha gratidão e apreço a todos aqueles que, direta ou indiretamente, contribuíram para que esta tarefa se tornasse realidade. Quero manifestar os meus sinceros agradecimentos.

Primeiramente gostaria de agradecer a minha mãe e meu pai, que desde criança foram os grandes incentivadores dos meus estudos, me apoiando em todas as fases do ensino até o presente momento. Também preciso agradecer a todas minhas irmãs (quatro), que sempre estiveram ao meu lado, escutando as dificuldades e me ajudando em momentos de grande dificuldade.

Os meus mais sinceros agradecimentos ao meu orientador Carlos Nascimento Silla Junior, entendendo em todos os momentos que, por motivos de trabalho ou saúde, eu não pude comparecer em algumas reuniões, mas sempre esteve disponível e me ajudou da melhor forma possível para que fosse possível a conclusão este trabalho.

Gostaria também de agradecer aos professores da banca final, que gentilmente aceitaram o convite para serem examinadores e puderam ceder um pouco do seu tempo. Obrigado!

E por último, mas não menos importante, a todos os meus colegas de trabalho, sendo eles coordenadores, professores e alunos, que sempre estiveram ao meu lado me apoiando, que por muitas vezes estavam passando pelas mesmas circunstâncias no mestrado, doutorado ou faculdade, mas nunca me deixaram esmorecer.

“Nós só podemos ver um pouco do futuro,
mas o suficiente para perceber que há muito a fazer.“
Alan Mathison Turing

Resumo

Atualmente, existem muitas músicas e partituras disponíveis na Internet. No entanto, identificar uma partitura apropriada para o estudo da música que seja compatível com o nível e o gosto específicos dos alunos de música é uma tarefa muito desafiadora, mesmo para professores de música. Existem várias razões para isso, como a quantidade de versões diferentes da mesma partitura e ainda cada uma contendo seu próprio nível de dificuldade. Definir a dificuldade de cada partitura de maneira manual é uma tarefa muito complexa, pois requer muito conhecimento teórico e domínio de um determinado instrumento musical. Por esse motivo, este trabalho trata da tarefa de classificar partituras automaticamente por nível de dificuldade. A abordagem proposta neste trabalho é usar os livros tradicionais de ensino de música de três diferentes instrumentos musicais, violino, violão e piano como padrão para criar cada nível de dificuldade. Cada uma das partituras de cada livro musical é convertida para um formato digital, a partir deste formato é possível extrair características de alto nível da música e também extrair características da imagem da partitura. Essas características são usadas como entrada para vários classificadores de aprendizado de máquina, que são treinados e testados. Os resultados obtidos mostram que para piano, considerando três níveis de dificuldade, fácil, médio e díficil os resultados variam de 60% a 83% de acurácia. Para violão, considerando apenas dois níveis de dificuldade, o nível fácil e o médio, a taxa de acerto varia de 54% a 84% e para violino, considerando também três níveis de dificuldade, a acurácia varia entre 59% e 85%. Estes resultados representam quanto cada classificador acerta em cada nível de dificuldade. Esses resultados representam o acerto do classificador em classificar cada partitura no nível correto.

Palavras-chave: Classificação de partituras musicais, música, nível de dificuldade.

Abstract

Currently, there are many songs and scores available on the Internet. However, identifying an appropriate score for the study of music that is compatible with the specific level and taste of music students is a very challenging task, even for music teachers. There are several reasons for this, such as the number of different versions of the same score and each containing its own level of difficulty. Defining the difficulty of each score manually is a very complex task, as it requires a lot of theoretical knowledge and mastery of a particular musical instrument. For this reason, this task of sorting scores automatically by level of difficulty. The approach proposed in this work is to use the traditional music teaching books of three different musical instruments, violin, guitar and piano as a pattern to create each level of difficulty. Each of the scores of each musical book is converted to a digital format, from this format it is possible to extract high-level characteristics from the music and also extra characteristics from the image of the score. These characteristics are used as input to various machine learning classifiers, which are trained and tested. The results found show that for piano, according to three levels of difficulty, easy, medium and difficult, the results vary from 60% to 83% accuracy. For guitar, considering only two levels of difficulty, the easy and medium level, the hit rate varies from 54% to 84% and for violin, also considering three levels of difficulty, the accuracy varies between 59% and 85%. These results represent how much each classifier hits at each difficulty level. These results represent the correctness of the classifier in classifying each score at the correct level.

Keywords: Classification of music scores, music, difficult level.

Lista de ilustrações

Figura 1 – Twinkle, Twinkle, Little Star - Variação A. 22
Figura 2 – Twinkle, Twinkle, Little Star - Variação B. 23
Figura 3 – Twinkle, Twinkle, Little Star - Variação C. 23
Figura 4 – Twinkle, Twinkle, Little Star - Variação D. 23
Figura 5 – Possíveis fases de um Sistema de Múltiplos Classificadores 34
Figura 6 – Exemplo de texturas, com a extração de suas respectivas imagens e histograma LBP 37
Figura 7 – Ilustração da Extração do LBP 38
Figura 8 – Exemplo de uma partitura musical. 38
Figura 9 – Exemplo da estrutura de um arquivo no formato MusicXML. 39
Figura 10 – Exemplo da estrutura de um arquivo no formato ARFF. 40
Figura 11 – Processo metodológico da pesquisa 48
Figura 12 – Readequação dos níveis de dificuldade para o método Suzuki 49
Figura 13 – Readequação dos níveis de dificuldade para o método Leila Fletcher . . 49
Figura 14 – Readequação dos níveis de dificuldade para o método Henrique Pinto . 50
Figura 15 – Exemplo da partitura musical L'Avalanche 1 51
Figura 16 – Exemplo da estrutura do arquivo XML gerado pelo jSymbolic para a música da Figura 1 51
Figura 17 – Todas as características disponibilizadas pelo jSymbolic 53
Figura 18 – Exemplo da junção entre dois vetores de características diferentes . . . 54
Figura 19 – Bases e dados e classificadores utilizados na pesquisa 56
Figura 20 – Matriz da base para violino com o classificador de Rede Neural 59
Figura 21 – Matriz da base para violão com o classificador de Rede Neural 59
Figura 22 – Matriz da base para piano com o classificador de Rede Neural 59

Lista de tabelas

Tabela 1 – Configuração do classificador J48, no software Weka. 27
Tabela 2 – Configuração do classificador IBk, no software Weka. 28
Tabela 3 – Configuração do classificador MultiLayer Perceptron, no software Weka. 30
Tabela 4 – Configuração do classificador Naive Bayes, no software Weka. 31
Tabela 5 – Configuração do classificador SMO, no software Weka. 33
Tabela 6 – Configuração do classificador Random Forest, no software Weka. . . . 35
Tabela 7 – Tabela comparativa entre os trabalhos relacionados 45
Tabela 8 – Resultado geral da acurácia das classificações por instrumento musical, com violino, piano e violão 58
Tabela 9 – Taxa de acerto por nível de dificuldade do formato MIDI 60
Tabela 10 – Taxa de acerto por nível de dificuldade do formato MIDI e da imagem digital 62
Tabela 11 – Resultado da comparação entre a taxa de acerto do formato MIDI e Imagem Digital 63
Tabela 12 – Resultado geral da acurácia das classificações por instrumento musical, com violino, piano, violão e uma base com todos os intrumentos musicais 64
Tabela 13 – Treino com piano e teste com violino, para o formato MIDI, imagem digital e a fusão entre o MIDI e imagem digital 65
Tabela 14 – Resultado geral da acurácia das classificações com o teste e treino invertido 66
Tabela 15 – Matriz de confusão do classificador C4.5 para abordagem em formato MIDI 75
Tabela 16 – Matriz de confusão do classificador C4.5 para abordagem da imagem digital 75
Tabela 17 – Matriz de confusão do classificador C4.5 para abordagem da fusão do formato MIDI com a imagem digital 75
Tabela 18 – Matriz de confusão do classificador Naive Bayes para abordagem em formato MIDI 75
Tabela 19 – Matriz de confusão do classificador Naive Bayes para abordagem da imagem digital 76
Tabela 20 – Matriz de confusão do classificador Naive Bayes para abordagem da fusão do formato MIDI com a imagem digital 76
Tabela 21 – Matriz de confusão do classificador Random Forest para abordagem em formato MIDI 76
Tabela 22 – Matriz de confusão do classificador Random Forest para abordagem da imagem digital 76

Tabela 23 – Matriz de confusão do classificador Random Forest para abordagem da fusão do formato MIDI com a imagem digital 76
Tabela 24 – Matriz de confusão do classificador SVM para abordagem em formato MIDI 76
Tabela 25 – Matriz de confusão do classificador SVM para abordagem da imagem digital 77
Tabela 26 – Matriz de confusão do classificador SVM para abordagem da fusão do formato MIDI com a imagem digital 77
Tabela 27 – Matriz de confusão do classificador KNN para abordagem em formato MIDI 77
Tabela 28 – Matriz de confusão do classificador KNN para abordagem da imagem digital 77
Tabela 29 – Matriz de confusão do classificador KNN para abordagem da fusão do formato MIDI com a imagem digital 77
Tabela 30 – Matriz de confusão do classificador Rede Neural para abordagem em formato MIDI 77
Tabela 31 – Matriz de confusão do classificador Rede Neural para abordagem da imagem digital 78
Tabela 32 – Matriz de confusão do classificador Rede Neural para abordagem da fusão do formato MIDI com a imagem digital 78

Lista de abreviaturas e siglas

ARFF	Attribute-Relation File Format
DBNs	Deep Belief Networks
IJCNN	International Joint Conference on Neural Networks
KNN	K-Nearest Neighbour
LBP	Local Binary Pattern
MIDI	Musical Instrument Digital Interface
MLP	Multilayer Perceptron
MIR	Music Information Retrieval
NB	Naive Bayes
PDF	Portable Document Format
RF	Random Forest
RNAs	Redes Neurais Artificiais
SMC	Sistema de Múltiplos Classificadores
SVM	Support Vector Machine
XML	eXtensible Markup Language

Sumário

1 INTRODUÇÃO 21
1.1 Definição do problema 21
1.2 Objetivos 23
1.3 Questões de pesquisa 24
1.4 Estrutura do trabalho 24

2 FUNDAMENTAÇÃO TEÓRICA 25
2.1 Teoria musical 25
2.1.1 Método Suzuki 25
2.1.2 Método Leila Fletcher 25
2.1.3 Método Henrique Pinto 26
2.2 Teoria da classificação e seus classificadores 26
2.2.1 Classificador baseado em árvore de decisão 26
2.2.2 Classificador baseado em instâncias 28
2.2.3 Classificador baseado em redes neurais 29
2.2.4 Classificador baseado em funções probabilísticas 31
2.2.5 Classificador baseado em sistemas de máquina de suporte 32
2.2.6 Sistema de Múltiplos Classificadores - *Ensembles* 33
2.2.7 Classificador baseado em sistemas de múltiplos classificadores 34
2.2.8 Early Fusion 36
2.3 Descritor de características 36
2.3.1 LBP - Local Binary Pattern 36
2.4 Tecnologias 37
2.4.1 MusicXML 37
2.4.2 MIDI 39
2.4.3 ARFF 40
2.4.4 jSymbolic 40
2.4.5 WEKA 40
2.4.6 Python 41

3 ESTADO DA ARTE 43

4 METODOLOGIA DE PESQUISA 47
4.1 Definição dos níveis de dificuldade e rotulação da base de dados . . 47
4.2 Extração das características 50
4.2.1 Extração das características do arquivo musical 50

4.2.1.1 Características 52
4.2.2 Extração das características da imagem 52
4.2.3 Junção dos vetores de música e imagem - Early Fusion 52
4.3 Classificação **54**

5 RESULTADOS E DISCUSSÕES **57**
5.1 Classificacão automática de partituras por nível de dificuldade utilizando características simbólicas de alto nível **57**
5.2 Formato MIDI Vs. Imagem digital **61**
5.3 Fusão entre as bases de dados de instrumentos diferentes **63**
5.4 Treino e teste em bases com intrumentos musicais diferentes **65**

6 CONCLUSÃO **67**

REFERÊNCIAS **69**

APÊNDICES **73**

APÊNDICE A – MATRIZES DE CONFUSÃO DO MÉTODO SUZUKI **75**

1 Introdução

(GROUP et al., 2002) Com o passar do tempo, a criação de novas tecnologias no campo da música passou de uma conveniência para uma necessidade. Desde então, o computador é uma parte essencial utilizada na manipulação da música. O MIDI (Musical Instrument Digital Interface) (GUÉRIN, 2008), desde sua criação em meados dos anos 80, foi um protocolo que teve um grande impacto na tecnologia musical, permitindo grandes feitos no que diz respeito a digitalização da música e a comunicação entre diferentes instrumentos eletrônicos, sendo ainda uma ferramenta muito utilizada por sua eficiência (ANDERTON, 2014). Os avanços tecnológicos proporcionaram também a criação de muitas ferramentas para dar assistência principalmente na construção, manutenção e preservação das partituras. Nesse contexto, diversos softwares, dispositivos e formatos de arquivos foram desenvolvidos e tem sido muito utilizado para ajudar no processo de integração da música com o computador. Alguns desses formatos auxiliam na edição das partituras, como o MusicXML (GOOD et al., 2001). No aprendizado de instrumentos musicais também existem aplicações capazes de apoiar a prática dos instrumentos e da música de forma que facilite e estimule seu aprendizado (LIN; LIU, 2006), porém métodos tradicionais como o método Suzuki (SUZUKI, 2015a), Leila Fletcher (FLETCHER, 1993c) e Henrique Pinto (PINTO, 2008a) ainda continuam sendo adotados por sua credibilidade com a comunidade musical (Suzuki International, 2018).

A música é uma parte importante da arte e está presente no cotidiano da sociedade. Ela é considerada uma forma primordial de expressão criativa dos sentimentos e até de comunicação social. É muito importante que ela continue a ser preservada e ensinada desde cedo, pois ela ajuda no desenvolvimento intelectual e psíquico dos indivíduos (ČRNČEC; WILSON; PRIOR, 2006). O desenvolvimento de tecnologias que colaborem no ensino da música ou que facilitem sua introdução no ensino é um estímulo para que ela continue a ser apreciada e explorada de formas que possam beneficiar no contexto educacional e criativo. O método desenvolvido neste trabalho visa a melhoria de qualidade no desempenho de atividades que podem ser trabalhosas se feitas por seres humanos, mas que quando desempenhadas por máquinas poderiam ser muito mais rápidas e efetivas. O método pretende ajudar professores e alunos de música na definição de níveis para estudo, diminuindo a necessidade da definição manual de níveis de dificuldade.

1.1 Definição do problema

Estudantes de música querem e deveriam progredir mais em seu conhecimento no instrumento ao mesmo tempo em que ampliam seu repertorio musical. No entanto, a

escolha de músicas que façam parte do seu nível atual de conhecimento musical não é tão simples assim. O processo de seleção de músicas no aprendizado de um instrumento é feito manualmente e exige um amplo conhecimento musical em um instrumento específico. Por esse motivo, é complexo selecionar músicas que possam compor um método de estudo e que possam servir de apoio para os alunos. Para os educadores de música, atribuir níveis às músicas de forma manual é trabalhoso e exige muito tempo, estudo e dedicação, alguns estudos já foram propostos como alternativa a esse problema (GOUYON et al., 2012). Por outro lado, esses trabalhos focam apenas em músicas para piano e são voltados para as dificuldades encontradas no desempenho deste instrumento.

Diversas partituras estão disponíveis on-line, mas muitas vezes podem ser encontradas em várias versões ou variações diferentes da mesma música, como nas diversas variações da música "Twinkle, Twinkle, Little Star", apresentada nas Figuras 1, 2, 3 e 4 do método de ensino para violino Suzuzi 1(SUZUKI, 2015a). Essa variedade de opções pode acabar dificultando na identificação da versão ou do nível adequado da música. Um outro problema e que sem a ajuda de um professor, os alunos iniciantes de música podem ter muita dificuldade em definir quais são músicas que eles devem estudar se não estiverem seguindo um método de ensino específico. Isso acaba limitando, não somente a quantidade de músicas que eles podem aprender inicialmente, como também a quantidade de gêneros musicais, pelo fato de que os métodos mais utilizados não tem uma variedade grande de gêneros musicais.

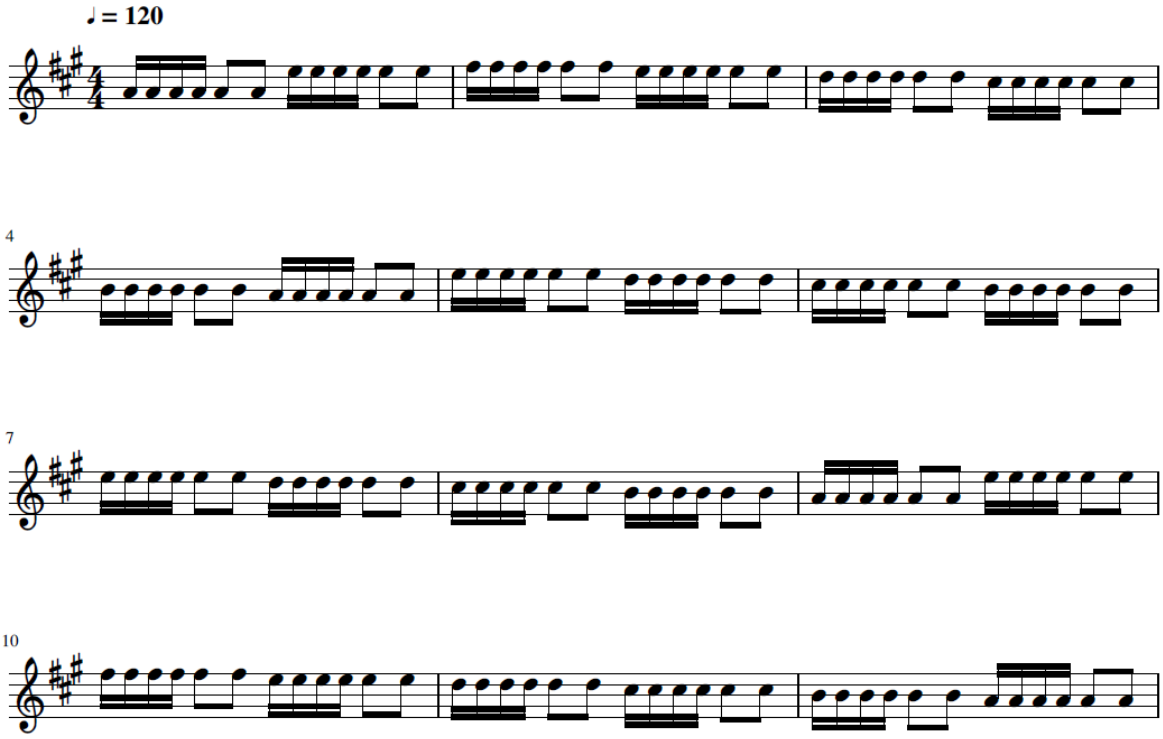

Figura 1 – Twinkle, Twinkle, Little Star - Variação A.
Fonte: (SUZUKI, 2015a)

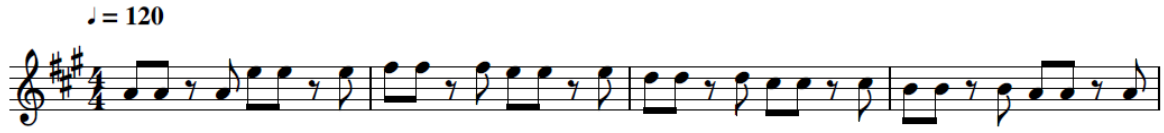

Figura 2 – Twinkle, Twinkle, Little Star - Variação B.
Fonte: (SUZUKI, 2015a)

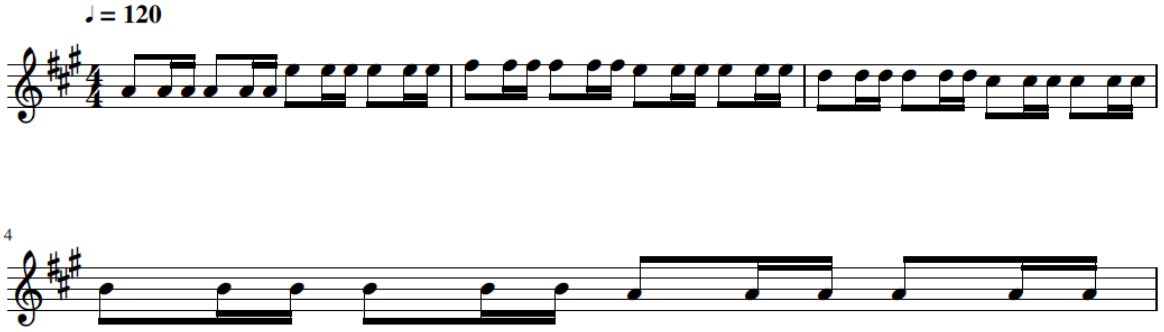

Figura 3 – Twinkle, Twinkle, Little Star - Variação C.
Fonte: (SUZUKI, 2015a)

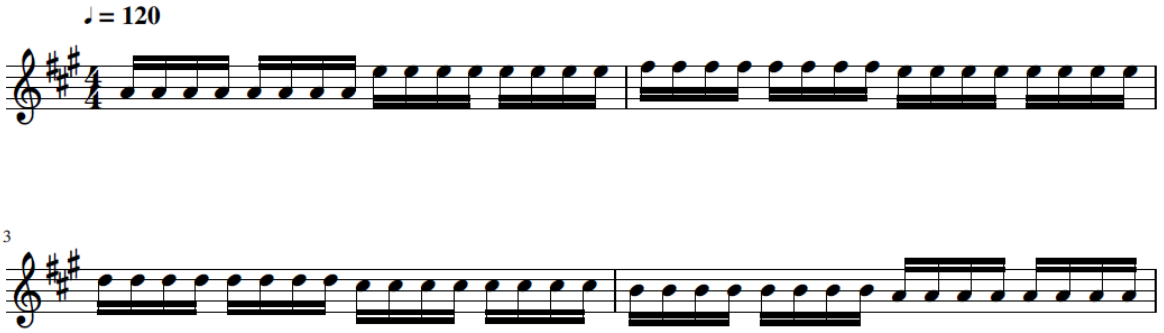

Figura 4 – Twinkle, Twinkle, Little Star - Variação D.
Fonte: (SUZUKI, 2015a)

1.2 Objetivos

O objetivo deste trabalho é desenvolver um método capaz de automatizar a organização de partituras musicais por níveis de dificuldade.

Os objetivos específicos deste trabalho são:

- Criar bases de dados com as partituras musicais de acordo com os métodos tradicionais Suzuki, Leila Fletcher e Henrique Pinto.
- Extrair características a partir do formato MIDI, para que possam ser utilizadas nos classificadores.
- Extrair características a partir da imagem da partitura em formato digital para que possam ser utilizadas nos classificadores.

- Classificar partituras musicais a partir de características musicais e da imagem em formato digital da partitura .

1.3 Questões de pesquisa

A execução do trabalho terá além de seu objetivo principal, questões a serem respondidas com base na classificação de partituras musicais pelo áudio e pela imagem. As questões a serem respondidas estão abaixo listadas:

- 1. É possível classificar partituras musicais por nível de dificuldade utilizando aprendizagem de máquina?
- 2. Existe diferença entre a utilização de características de alto nível simbólico do áudio da partitura musical, em relação a características da imagem da partitura musical para classificação automática por nível de dificuldade?
- 3. O que acontece com os resultados de classificação quando as bases dos diferentes instrumentos são unificadas em uma única base?
- 4. Qual é o impacto de treinar classificadores com a base de dados de um instrumento musical, por exemplo piano, e testar com a base de dados de outro instrumento musical, por exemplo violino?

1.4 Estrutura do trabalho

A presente dissertação de mestrado está organizada em seis capítulos. No Capítulo 2 é feita uma revisão sobre os métodos tradicionais do ensino da música escolhidos para esta pesquisa, falamos também sobre algoritmos de classificação e uma breve explicação sobre as tecnologias utilizadas nessa pesquisa. O Capítulo 3 apresenta o estado da arte do tema proposto. Os procedimentos metodológicos utilizados nesta pesquisa são apresentados no Capítulo 4. O Capítulo 5 apresenta os resultados finais e as discussões que esses resultados geram, junto com as questões de pesquisas a serem respondidas. Finalmente, o Capítulo 6 apresentamos uma abordagem final para o tema, junto com as próximas etapas dessa pesquisa.

2 Fundamentação Teórica

Neste capítulo são apresentadas as principais definições e conceitos necessários para o entendimento do método proposto e dos resultados obtidos. As seções a seguir apresentam informações sobre Teoria Musical, Classificação e as Tecnologias utilizadas nessa pesquisa. É importante salientar que os métodos escolhidos na Seção 2.1, foram selecionados com o apoio de um especialista em música. Cada método escolhido trata apenas de um instrumento musical, por isso o presente projeto limita-se apenas a classificação de partituras musicais com violino, violão e piano.

2.1 Teoria musical

2.1.1 Método Suzuki

O método Suzuki(Suzuki International, 2018) é um método de estudo de música instrumental que se apoia no princípio chamado abordagem língua-mãe em que crianças podem aprender música da mesma forma que aprendem sua língua nativa: ouvindo e repetindo. O violinista e educador Shinichi Suzuki, criador do método, acreditava que começar cedo é essencial, mas que nunca é muito tarde para começar. Através dos anos, sua filosofia transmitiu a milhares de pessoas, que o envolvimento dos pais e professores, lições em grupo e aprender a tocar antes de aprender a ler eram as chaves para um futuro promissor na música. O método Suzuki é um método muito conhecido e utilizado em todo o mundo como forma de aprendizado de diversos instrumentos musicais (Suzuki Americas,).

2.1.2 Método Leila Fletcher

O método Leila Fletcher é enumerado, não para representar as várias notas musicais que existem, mas como uma representação de um curso contínuo de educação musical. O método de piano foi projetado para atender aos requisitos de adolescentes e é projetado para permitir que o aluno tenha um progresso constante e desfrute de satisfações imediatas. O material utilizado no curso foi testado com experiência real no ensino de um grande número de alunos, e os resultados de seu uso são: maior interesse no estudo da música, melhor musicalidade e uma evasão menor de alunos que interrompem o estudo da música por perda de interesse ou desânimo. O método Leila Fletcher é dedicado a um objetivo quádruplo: o desenvolvimento da capacidade de ler música fluentemente e interpretá-la artisticamente, o estabelecimento de uma técnica de piano abrangente e sonora, a nutrição

do talento musical criativo e a promoção de uma experiência duradoura (FLETCHER, 1993c).

2.1.3 Método Henrique Pinto

O método Henrique Pinto tem como objetivo parcelar cada problema durante as várias etapas da iniciação violonística. A individualização de cada fase irá acentuar de uma forma positiva e concreta os primeiros reflexos. Sobre esta base irá ser construída toda a evolução Instrumental. A separação dos objetivos, individualizando os problemas, não quer dizer rigidez ou falta de relaxamento, mas sim, colocar os músculos de tal forma a facilitar seu trabalho durante a execução. Assim sendo, os alunos terão a movimentação muscular a favor do instrumento, portanto, o relaxamento será uma consequência natural. Quanto ao fator psicológico, com relação ao instrumento, remos pensar no vilão como uma extensão de nosso corpo, e não como um objeto estranho e incômodo (PINTO, 2008a).

2.2 Teoria da classificação e seus classificadores

A classificação é uma técnica de aprendizado de máquina no qual necessita de uma série de exemplos já classificados, que servirão de base para que um classificador de aprendizagem de máquina, a partir do treinamento com esses exemplos já rotulados, espera-se que o algoritmo possa classificar novas instâncias de acordo com os exemplos utilizados. Quando os métodos atuam perante supervisão fornecendo resultados para cada exemplo de treinamento, denominados classes de exemplo, a classificação é qualificada como supervisionada. As taxas de sucesso do aprendizado da máquina são determinadas a partir da descrição do conceito que é aprendido utilizando um conjunto independente de dados para teste, onde as classificações consideradas legítimas são conhecidas, mas não mostradas à máquina. Essas taxas medem a precisão na classificação que é basicamente o quão bem esse conceito foi aprendido (WITTEN, 2005).

2.2.1 Classificador baseado em árvore de decisão

É um algoritmo de classificação supervisionada utilizado para prever a classe de um objeto baseado no treinamento aprendido com os exemplos disponibilizados. O algoritmo monta uma estrutura de árvore utilizando os testes de atributos e a partir desta estrutura e possível classificar a amostra desconhecida sem necessariamente testar todos os valores e seus atributos. Uma estrutura de arvore é basicamente um conjunto de nós que possuem ramificações. Os testes que dividem os atributos determinam a sequência da classificação de uma nova amostra, definindo em qual classe a amostra se encaixa melhor (WITTEN, 2005). No projeto o classificador utilizado foi o j48 no software weka, os parâmetros e valores utilizados para o classificador podem ser observados na Tabela 1.

Parâmetro	Valor
batchSize	10
binarySlipts	False
collapseTree	True
confidenceFactor	0.25
debug	False
doNotCheckCapabilities	False
doNotMakeSplitPointActualValue	False
minNumObj	2
numDecimalPlaces	2
numFolds	3
reducedErrorPruning	False
saveInstanceData	False
seed	1
subtreeRaising	True
unpruned	False
useLaplace	False
useMDLcorrection	True

Tabela 1 – Configuração do classificador J48, no software Weka.
Fonte: Autoria própria

Abaixo segue a descrição de cada parâmetro:

- **batchSize:** O número preferencial de instâncias a serem processadas se a previsão.
- **binarySlipts:** Caso seja utilizado atributos nominais na construção da árvore.
- **collapseTree:** Determina se partes da árvore serão removidas.
- **confidenceFactor:** Informa o fator de conAança para a podagem da árvore.
- **debug:** Indica se informações adicionais serão impressas no console.
- **doNotCheckCapabilities:** Determina se os recursos do classificador não serão verificados antes de o classificador ser construído.
- **doNotMakeSplitPointActualValue:** Sinaliza se o ponto de divisão da árvore será um valor de dado real.
- **minNumObj:** O número mínimo de instâncias por folha.
- **numDecimalPlaces:** Quantidade de números após a vírgula.
- **numFolds:** Determina a quantidade de dados que será usada para a redução de erros durante o processo de poda.
- **reducedErrorPruning:** denota se a podagem de erro reduzido será aplicada ao invés da podagem C4.5

- **saveInstanceData:** Se deseja salvar os dados de treinamento para visualização.
- **seed:** Um número randômico usado para reduzir os erros gerados pela poda.
- **subtreeRaising:** Determina se deverão ser considerados as sub árvores, aumentando a operação durante a poda.
- **unpruned:** Aponta se a árvore será podada.
- **useLaplace:** Determina se a contagens de folhas será suavizadas com base no teorema de Laplace.
- **useMDLcorrection:** Indica se a correção MDL será aplicada quando encontradas divisões em atributos numéricos.

2.2.2 Classificador baseado em instâncias

O método KNN é considerado um dos métodos de classificação mais antigos e simples (COVER; HART, 1967). Apesar da sua simplicidade, esse método tem alcançado bom desempenho em diferentes cenários. O método KNN é um *lazy learning*. Um *lazy learning* simplesmente armazena os documentos de treino e realiza três etapas para classificar os documentos. Dado um documento de teste d, para classificá-lo o método KNN tradicionalmente realiza as seguintes atividades: 1. A distância entre o documento de cada um dos documentos de treino é calculada utilizando alguma medida de similaridade entre documentos, tal como a medida do cosseno. 2. Os k documentos de treino mais próximos, isto é, mais similares ao documento d são selecionados. 3. O documento d é classificado em determinada categoria de acordo com algum critério de agrupamento das categorias dos k documentos de treino selecionados na etapa anterior (SANTOS et al., 2009). No projeto o classificador utilizado foi o IBk no software weka, os parâmetros e valores utilizados para o classificador podem ser observados na Tabela 2.

Parâmetro	Valor
KNN	1
batchSize	100
crossValidate	False
debug	False
distanceWeighting	No distance weighting
doNotCheckCapabilities	False
meanSquared	False
nearestNeighbourSearchAlgorithm	LinearNNSearch
numDecimalPlaces	2
windowSize	0

Tabela 2 – Configuração do classificador IBk, no software Weka.
Fonte: Autoria própria

Abaixo segue a descrição de cada parâmetro:

- **KNN:** Número de vizinho a serem utilizados.
- **batchSize:** O número preferencial de instâncias a serem processadas se a previsão.
- **crossValidate:** Indicação sobre o uso do método de validação *hold-one-out* nas validações cruzadas, afim de eleger o melhor valor entre 1 e aquele especificado no parâmetro KNN.
- **debug:** Indica se informações adicionais serão impressas no console.
- **distanceWeighting:** Método matemático para cálculo do peso das distâncias.
- **doNotCheckCapabilities:** Determina se os recursos do classificador não serão verificados antes de o classificador ser construído.
- **meanSquared:** Utilização do erro quadrado médio ao invés do erro absoluto médio durante as validações cruzadas.
- **nearestNeighbourSearchAlgorithm:** Definição do algoritmo utilizado.
- **numDecimalPlaces:** Quantidade de números após a vírgula.
- **windowSize:** Montante máximo de instâncias na Ąla de treinamento.

2.2.3 Classificador baseado em redes neurais

Em ciência da computação e campos relacionados, redes neurais artificiais (RNAs) são modelos computacionais inspirados pelo sistema nervoso central de um animal que são capazes de realizar o aprendizado de máquina bem como o reconhecimento de padrões. Redes neurais artificias geralmente são apresentadas como sistemas de neurônios interconectados, que podem computar valores de entradas, simulando o comportamento de redes neurais biológicas. Por exemplo, uma rede neural para o reconhecimento de escrita manual é definida por um conjunto de neurônios de entrada que podem ser ativados pelos pixels de uma imagem de entrada. Os dados adquiridos por essa ativação dos neurônios são então repassados, ponderadas e transformadas por uma função determinada pelo criador da rede, a outros neurônios. Este processo é repetido até que, finalmente, um neurônio de saída é ativado. Isso determina que caractere foi lido (BISHOP et al., 1995).No projeto o classificador utilizado foi o MultiLayer Perceptron no software weka, os parâmetros e valores utilizados para o classificador podem ser observados na Tabela 3.

Abaixo segue a descrição de cada parâmetro:

- **GUI:** Se será aberta uma interface visual.

Parâmetro	Valor
GUI	False
autoBuild	True
batchSize	100
debug	False
decay	False
doNotCheckCapabilities	False
hiddenLayers	a
learningRate	0.3
momentun	0.2
nominalToBinaryfilter	True
normalizeAttributes	True
normalizeNumericClass	True
numDecimalPlaces	2
reset	True
resume	False
seed	0
trainingTime	500
validationSetSize	0
validationTreshold	20

Tabela 3 – Configuração do classificador MultiLayer Perceptron, no software Weka.
Fonte: Autoria própria

- **autoBuild:** Determina se ocorrerá a adição e conexão das camadas escondidas da rede.
- **batchSize:** O número preferencial de instâncias a serem processadas se a previsão.
- **debug:** Indica se informações adicionais serão impressas no console.
- **decay:** Possibilita a diminuição da taxa de aprendizado do algoritmo, aplicando um cálculo que divide o valor inicial pelo número de épocas.
- **doNotCheckCapabilities:** Determina se os recursos do classificador não serão verificados antes de o classificador ser construído.
- **hiddenLayers:** Configuura a quantidade de camadas escondidas na rede. Quando igual a **a**, significa que o valor deve ser o somatório da quantidade de atributos com o número de classes divido por dois.
- **learningRate:** Define a taxa de aprendizado do algoritmo.
- **momentun:** Estipula o Momentum utilizado na atualização dos pesos da rede.
- **nominalToBinaryfilter:** Isso irá pré-processar as instâncias com o filtro Nominal-ToBinary. Isso pode ajudar a melhorar o desempenho se houver atributos nominais nos dados.

- **normalizeAttributes:** Determina se o valor dos atributos deverá passar por um filtro de normalização.
- **normalizeNumericClass:** Informa se classes numéricas serão submetidas ao filtro de normalização.
- **numDecimalPlaces:** Quantidade de números após a vírgula.
- **reset:** : Permite ao programa reiniciar o algoritmo, com um valor menor para a taxa de aprendizado, caso a classificação indicada pelo modelo seja diferente da real.
- **resume:** Define se o classificador pode continuar o treinamento após realizar o número solicitado de iterações.
- **seed:** Um número randômico usado na inicialização.
- **trainingTime:** Total de épocas, também chamadas de iterações, do algoritmo.
- **validationSetSize:** Especifica o tamanho, em porcentagem, do conjunto de validação.
- **validationTreshold:** Estabelece o montante de vezes possíveis que o erro da rede pode aumentar, antes da fase de treinamento do algoritmo ser Ąnalizada.

2.2.4 Classificador baseado em funções probabilísticas

O classificador Naive de Bayes é uma família de simples lassificadores probabilísticos"baseados na aplicação do teorema de Bayes, que parte do princípio da independência forte entre os recursos. Foi introduzido sob um nome diferente na comunidade no início dos anos 1960 e continua sendo um método popular para categorização de texto. O classificador Naive Bayes é altamente escalonável, exigindo um número de parâmetros lineares no número de variáveis (DUDA et al., 2001). No projeto o classificador utilizado foi o Naive Bayes no software weka, os parâmetros e valores utilizados para o classificador podem ser observados na Tabela 4.

Parâmetro	Valor
batchSize	100
debug	False
displayModelInOldformat	False
doNotCheckCapabilities	False
numDecimanlPlaces	2
userKernelEstimator	False
userSupervisedDiscretization	false

Tabela 4 – Configuração do classificador Naive Bayes, no software Weka.
Fonte: Autoria própria

Abaixo segue a descrição de cada parâmetro:

- **batchSize:** O número preferencial de instâncias a serem processadas se a previsão.
- **debug:** Indica se informações adicionais serão impressas no console.
- **displayModelInOldformat:** Utilizar a saída antiga do modelo.
- **doNotCheckCapabilities:** Determina se os recursos do classificador não serão verificados antes de o classificador ser construído.
- **numDecimalPlaces:** Quantidade de números após a vírgula.
- **userKernelEstimator:** Define a utilização, ou não, do estimador de kernel para os atributos ao invés da distribuição normal.
- **userSupervisedDiscretization:** determina se a discretização dos valores dos atributos ocorrerá.

2.2.5 Classificador baseado em sistemas de máquina de suporte

Uma máquina de vetores de suporte (SVM) é um conceito na ciência da computação para um conjunto de métodos do aprendizado supervisionado que analisam os dados e reconhecem padrões. O SVM toma como entrada um conjunto de dados e prediz, para cada entrada dada, qual de duas possíveis classes a entrada faz parte, o que faz do SVM um classificador não linear binário não probabilístico. O que um SVM faz é encontrar uma linha de separação, mais comumente chamada de hiperplano entre dados de duas classes. Essa linha busca maximizar a distância entre os pontos mais próximos em relação a cada uma das classes (LORENA; CARVALHO, 2007). No projeto o classificador utilizado foi o SMO no software weka, os parâmetros e valores utilizados para o classificador podem ser observados na Tabela 5.

Abaixo segue a descrição de cada parâmetro:

- **batchSize:** O número preferencial de instâncias a serem processadas se a previsão.
- **buildCalibrationModel:** Ajusta modelos de calibração aos outputs do SVM (para estimativas de probabilidade adequadas).
- **c:** O parâmetro de complexidade C.
- **calibrator:** O método de calibração a ser usado.
- **checksTurnedOff:** Desativa verificações demoradas.
- **debug:** Indica se informações adicionais serão impressas no console.

Parâmetro	Valor
batchSize	100
buildCalibrationModel	False
c	1.0
calibrator	Logistic
checksTurnedOff	False
debug	False
doNotCheckCapabilities	False
epislon	1.0E-12
filterType	Normalize training data
kernel	PolyKernel
numDecimanlPlaces	2
numFolds	-1
randomSeed	1
toleranceParameter	0.001

Tabela 5 – Configuração do classificador SMO, no software Weka.
Fonte: Autoria própria

- **doNotCheckCapabilities:** Determina se os recursos do classificador não serão verificados antes de o classificador ser construído.
- **epislon:** O epsilon para erro de arredondamento(não deve ser alterado).
- **filterType:** Determina como/se os dados serão transformados.
- **kernel:** O kernel a ser usado.
- **numDecimalPlaces:** Quantidade de números após a vírgula.
- **numFolds:** O número de dobras para validação cruzada usado para gerar dados de treinamento para modelos de calibração.
- **randomSeed:** : Semente de número aleatório para a validação cruzada.
- **toleranceParameter:** : O parâmetro de tolerância(não deve ser alterado).

2.2.6 Sistema de Múltiplos Classificadores - *Ensembles*

A classificação é uma tarefa fundamental no reconhecimento de padrões, que, embora os métodos disponíveis na literatura possam ser diferentes em muitos aspectos, os resultados mais recentes da pesquisa levam a uma conclusão comum, criar um classificador monolítico para cobrir toda a variedade dos problemas de reconhecimento de padrões é algo impraticável. Dado este problema, muitos pesquisadores focaram nos Sistemas de Múltiplos Classificadores (SMC) e, consequentemente, muitas novas soluções foram dedicadas a cada uma das três fases possíveis do SMC: **geração**, **seleção** e **integração** , conforme a Figura 5.

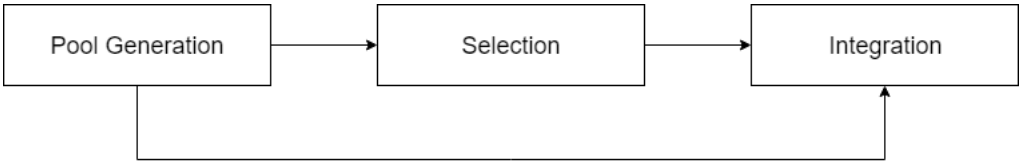

Figura 5 – Possíveis fases de um Sistema de Múltiplos Classificadores
Fonte: (JR; SABOURIN; OLIVEIRA, 2014)

A primeira fase é responsável pela geração de um *pool* de classificadores, um *pool* pode ser composto de classificadores homogêneos ou classificadores heterogêneos, que em ambos os casos, espera-se alguma diversidade. A ideia é gerar classificadores que cometam erros diferentes e, consequentemente, mostrem algum grau de complementaridade.

A segunda fase é responsável pela seleção, que por sua vez, pode ser estática ou dinâmica. Geralmente, a seleção é feita calculando a competência dos classificadores disponíveis no *pool*. Para esse fim, um processo de particionamento é comumente usado durante as fases de treinamento ou teste do SMC. Nesta parte, o espaço de características é dividido em diferentes partições e os classificadores mais competentes são levados a diante. Nos métodos de seleção estática, o particionamento é geralmente baseado em algoritmos evolucionários ou de cluster, e é executado durante a fase de treinamento. Em relação às medidas de competência, a literatura relata vários que consideram os classificadores individualmente ou em grupo. Vale ressaltar que, basicamente, as medidas individuais baseiam-se mais frequentemente na precisão do classificador. No entanto, as medidas são realizadas de maneiras diferentes. Por exemplo, pode-se encontrar medidas baseadas na exatidão pura, informação probabilística e entre outros.

A terceira fase de um SMC consiste em aplicar os classificadores selecionados para reconhecer um determinado padrão de teste, nos casos em que todos os classificadores são usados (sem seleção) ou quando um conjunto é selecionado, uma estratégia de fusão é necessária (JR; SABOURIN; OLIVEIRA, 2014).

2.2.7 Classificador baseado em sistemas de múltiplos classificadores

Random Forest é um algoritmo de aprendizado supervisionado, o algoritmo cria uma "floresta", que é um conjunto de árvores de decisão, mais frequentemente utilizadas com o método *bagging*. Uma grande vantagem do *random forest* é que ela pode ser usada para problemas de classificação e regressão. Random Forest tem quase os mesmos hiperparâmetros de uma árvore de um *bagging*. Em vez de procurar o recurso mais importante ao dividir um nó, ele procura o melhor recurso entre um subconjunto aleatório de recursos. Isso resulta em uma ampla diversidade que geralmente resulta em um modelo melhor. Portanto, no Random Forest, apenas um subconjunto aleatório dos recursos é levado em conta pelo algoritmo para dividir um nó. Você pode até tornar as árvores mais

aleatórias utilizando técnicas de aleatoriedade, ao invés de procurar o melhor conjunto ótimo possível (ENGLUND; VERIKAS, 2012).No projeto o classificador utilizado foi o Random Forest no software weka, os parâmetros e valores utilizados para o classificador podem ser observados na Tabela 6.

Parâmetro	Valor
bagSizePercent	100
batchSize	100
breakTiesRandomly	False
calcOutOfBag	False
computeAttributeImportance	False
debug	False
doNotCheckCapabilities	False
maxDepth	0
numDecimalPlaces	2
numExecutionSlots	1
numFeatures	0
numIterations	100
outputOutOfBagComplexityStatistics	False
printClassifiers	False
seed	1
StoreOutOfBagPredictions	False

Tabela 6 – Configuração do classificador Random Forest, no software Weka.
Fonte: Autoria própria

Abaixo segue a descrição de cada parâmetro:

- **bagSizePercent:** Tamanho de cada bag, como porcentagem do tamanho do conjunto de treinamento.
- **batchSize:** O número preferencial de instâncias a serem processadas se a previsão.
- **breakTiesRandomly:** Determina se os loops serão randomicamente quebrados quando muitos atributos trazem a mesmo resultado.
- **calcOutOfBag:** : Determina se o cálculo de erros “fora dos bags” estará presente.
- **computeAttributeImportance:** Calcule a importância do atributo por meio da diminuição média da impureza.
- **debug:** Indica se informações adicionais serão impressas no console.
- **doNotCheckCapabilities:** Determina se os recursos do classificador não serão verificados antes de o classificador ser construído.
- **maxDepth:** Profundidade máxima para checagem de atributos.

- **numDecimalPlaces:** Quantidade de números após a vírgula.
- **numExecutionSlots:** O número de slots de execução (threads) a serem usados na construção do conjunto.
- **numFeatures:** : Determina o randomicamente o número de atributos que serão utilizados pelo classificador.
- **numIterations:** O número de iterações a serem executadas.
- **seed:** Um número randômico usado para reduzir os erros gerados pela poda.
- **outputOutOfBagComplexityStatistics:** Determina se o Random Forest deve gerar estatísticas baseadas em complexidade quando a avaliação calcOutOfBag for executada.
- **printClassifiers:** Imprime os classificadores individuais na saída.
- **seed:** Número aleatório a ser utilizado pelo algoritmo.
- **StoreOutOfBagPredictions:** Determina se o algoritmo irá armazenar as classificações fora das "bags".

2.2.8 Early Fusion

A fusão inicial ou *early fusion* é realizada no nível das características. Nesse caso, os vetores de características de diferentes fontes são concatenados em um vetor de características único que será usado para classificação. Como esse vetor consiste em muitas características, o tempo de treinamento e classificação aumentará. No entanto, um vetor de recursos em grande escala em conjunto com métodos de aprendizado adequados pode levar a um desempenho muito melhor no final (EBERSBACH; HERMS; EIBL, 2017).

2.3 Descritor de características

2.3.1 LBP - Local Binary Pattern

Uma das formas utilizadas para classificar partituras por nível de dificuldade nessa pesquisa foi através da imagem de cada peça, para isso é necessário utilizar um algoritmo capaz extrair as características da mesma. O LBP (Local Binary Pattern) é um extrator que consiste em realizar cálculos matemáticos em cima de cada pixel da imagem, gerando uma nova imagem com padrões binários (AQUINO, 2017). O histograma dos padrões da imagem é comumente utilizados como vetor de características para a descrição de texturas. Isso é possível observar na Figura 6.

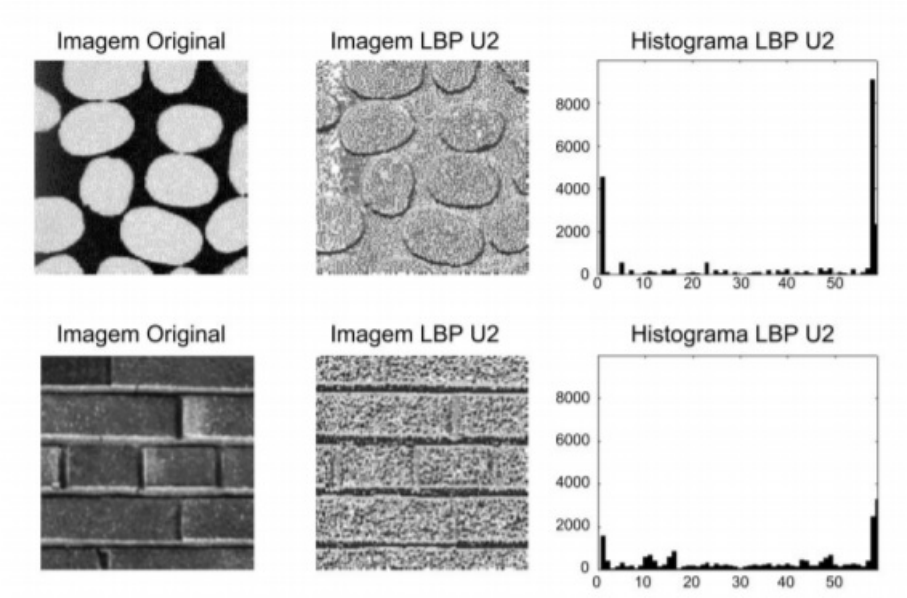

Figura 6 – Exemplo de texturas, com a extração de suas respectivas imagens e histograma LBP

Fonte: (BRODATZ, 1966)

A extração do LBP consiste em analisar os pixels da vizinhança com relação a um pixel central. Primeiro a imagem deve ser convertida para tons de cinza, após a conversão, o pixel vizinho que tiver a intensidade igual ou maior que o central, receberá o valor 1 e consequentemente, o pixel da vizinhança que tiver a menor intensidade que o pixel central, receberá o valor 0. Após o primeiro processo, deve ser associado aos pixels vizinhos os pesos na potência de 2, afim de formar um valor na base 10, com isso apenas os valores que anteriormente continham o valor binário 1, devem ser somados para obter um valor final. A Figura 7 explica melhor todo o processo da criação da nova imagem e posteriormente a geração do histograma, usando 8 pontos vizinhos e 1 pixel central.

2.4 Tecnologias

Nesta seção serão apresentadas as principais ferramentas e tecnologias utilizadas nessa pesquisa.

2.4.1 MusicXML

MusicXML e um formato de código aberto para partituras musicais digitais. Ele se destaca por usar um formato simples e flexível baseado em XML. O XML é um formato utilizado pela sua eficiência, facilidade e flexibilidade. Na internet existem diversos outros formatos digitais que também fazem essa representação das partituras e que caracterizam mais valores semânticos da música, porém, esses formatos são pagos e específicos de seus criadores. O MusicXML aparece como uma possível solução neste caso, seu surgimento tem

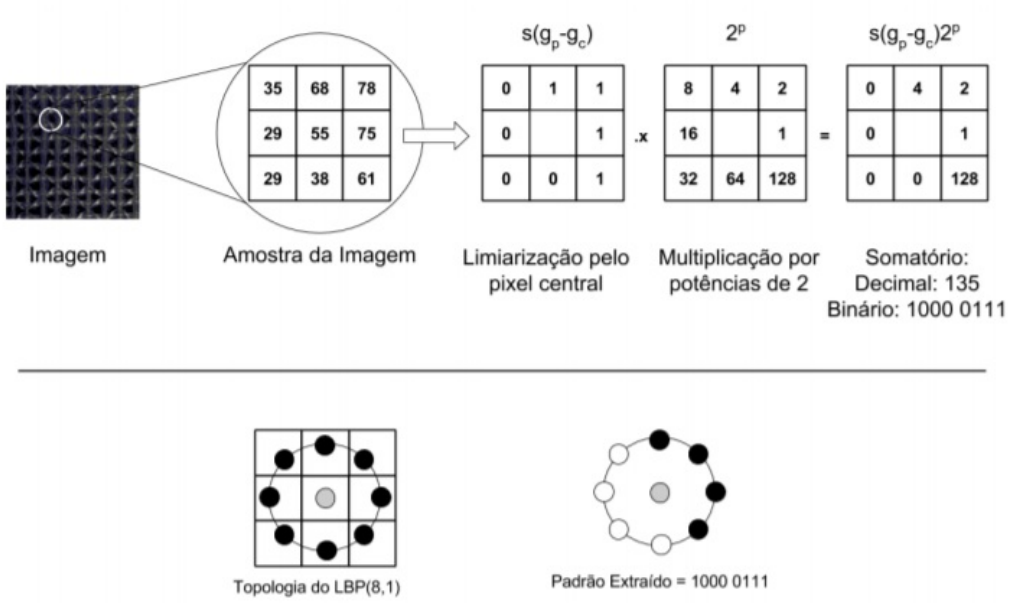

Figura 7 – Ilustração da Extração do LBP
Fonte: (AQUINO, 2017)

como objetivo estabelecer uma ferramenta on-line gratuita que unificaria os formatos de partituras digitais, se tornando um formato padrão (GOOD et al., 2001). Ele pode tanto ser manipulado com editores de XML, como em softwares criados especificamente para a edição de notação musical, como Finale e MuseScore (MAKEMUSIC,)(SHINN, 2013). O diferencial de usar esses softwares de notação musical é poder ler a partitura como se estivesse com o papel na mão, editar facilmente, reproduzir com *players* e imprimir sem a necessidade de entender a sintaxe de um documento XML.

Na Figura 8 podemos observar a visualização de uma partitura representando uma nota **Semibreve Dó baseada no tempo 4/4**. Para visualizar essa partitura, foi utilizado o software MuseScore que faz a leitura de arquivos MusicXML.

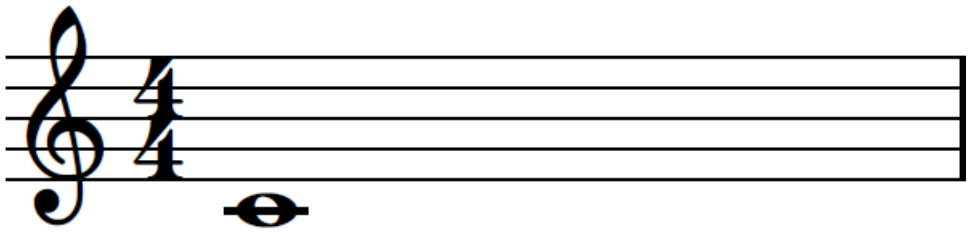

Figura 8 – Exemplo de uma partitura musical.
Fonte: Autoria própria

Na Figura 9 podemos observar a mesma partitura vista na Figura 8 mas no formato MusicXML.

```
<?xml version="1.0" encoding="UTF-8"?>                    → Declaração XML
<!DOCTYPE score-partwise PUBLIC
    "-//Recordare//DTD MusicXML 3.1 Partwise//EN"
    "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="3.1">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key>
          <fifths>0</fifths>
          </key>
        <time>
          <beats>4</beats>                                → Tempo compass 4/4
          <beat-type>4</beat-type
          </time>
        <clef>
          <sign>G</sign>                                  → Clave de Sol
          <line>2</line>
          </clef>
        </attributes>
      <note>
        <pitch>
          <step>C</step>                                  → Nota Dó
          <octave>4</octave>
          </pitch>
        <duration>4</duration>                            → Nota inteira (Semibreve)
        <type>whole</type>                                  4 tempos
        </note>
      </measure>
    </part>
  </score-partwise>
```

Figura 9 – Exemplo da estrutura de um arquivo no formato MusicXML.
Fonte: Autoria própria

2.4.2 MIDI

MIDI é uma interface ou linguagem padrão que foi desenvolvida para possibilitar a comunicação entre computadores e instrumentos musicais. Ele define como e quais informações são transmitidas, manipuladas ou executadas pelos dispositivos que se relacione com o seu padrão (GUÉRIN, 2008). Ele também é definido como protocolo que funciona por meio de controladores, sequenciadores e módulos de som. O arquivo MIDI, não retém som da mesma forma que outros formatos de áudio, na verdade ele guarda instruções de notas, tons e informações de como essa música deve ser tocada, a partir disso o *player* interpreta o arquivo e toca a música de acordo com as instruções contidas dentro do documento (MOOG, 1986).

2.4.3 ARFF

ARFF como demonstrado na Figura 10 é um formato de arquivo de texto que foi desenvolvido para ser utilizado no conceito de aprendizado de máquina. O ARFF descreve uma lista de instâncias que podem possuir diversos atributos. Ele funciona como uma matriz, onde cada linha de texto representa uma instância, como por exemplo uma música, e cada coluna representa um atributo dessa instância, por exemplo, um dos atributos de uma música poderia ser o formato melódico (GARNER et al., 1995).

Figura 10 – Exemplo da estrutura de um arquivo no formato ARFF.
Fonte: Autoria própria

2.4.4 jSymbolic

O *jSymbolic* é um software de código aberto desenvolvido em *JAVA*, destinado a extração de características de dados simbólicos de música. Diferentemente dos formatos de áudio que gravam sinais de ondas de som aproximadas, dados simbólicos são arquivos que retém intrinsecamente abstrações musicais de alto nível, por isso pode conter informações relevantes da música como por exemplo, tons e sequência de notas. O *jSymbolic* foi desenvolvido para ajudar na mineração de dados de música e pode ser utilizado na classificação automática contando com cerca de 160 características que podem ser extraídas de arquivos MIDI. Essas características foram baseadas a partir de trabalhos do estado da arte envolvendo teoria musical e MIR e se dividem em categorias como instrumentação, textura, ritmo, dinâmica, estatísticas de nota, melodia e acordes (MCKAY; FUJINAGA, 2006).

2.4.5 WEKA

O WEKA visa fornecer uma coleção abrangente de algoritmos de aprendizado de máquina e ferramentas para o pré-processamento de dados. Permite usuários experimentar e comparar diferentes algoritmos de aprendizagem de máquina em novos conjuntos de

dados. Sua arquitetura modular e extensível permite que processos de mineração de dados sejam construídos a partir da coleção de algoritmos de aprendizado básico e ferramentas fornecidas (HALL et al., 2009).

2.4.6 Python

O Python favorece a legibilidade do código-fonte, tornando a linguagem mais produtiva, a linguagem inclui diversas estruturas de alto nível (listas, dicionários, data/hora, complexos e outras) e uma vasta coleção de módulos prontos para uso, além de frameworks de terceiros que podem ser adicionados. Também inclui recursos encontrados em outras linguagens modernas, tais como geradores, introspecção, persistência, metaclasses e unidades de teste. Multiparadigma, a linguagem suporta programação modular e funcional, além da orientação a objetos. Mesmo os tipos básicos no Python são objetos. A linguagem é interpretada através de bytecode pela máquina virtual Python, tornando o código portável. Com isso é possível compilar aplicações em uma plataforma e rodar em outros sistemas ou executar direto do código-fonte. Python é um software de código aberto (com licença compatível com a General Public License [GPL], porém menos restritiva, permitindo que o Python seja inclusive incorporado em produtos proprietários). A especificação da linguagem é mantida pela Python Software Foundation (PSF). (BORGES, 2014).

3 Estado da Arte

Nesta seção será apresentado o que há de mais parecido com o que está sendo proposto nesta pesquisa. É importante salientar que existem apenas dois trabalhos que se aproximam daquilo que está sendo proposto.

O estudo de Chiu e Chen (2012) propôs reconhecer níveis de dificuldade de partituras criando uma nova abordagem de seleção de características baseadas em dificuldade. Eles analisaram a semântica do conteúdo musical simbólico de partituras para piano e aplicaram o conceito de regressão para estimar os níveis de dificuldade. Seu trabalho utilizou duas bases de dados reais, contendo 159 e 184 partituras musicais respectivamente, para a entrada de dados o formato utilizado foi o arquivo MIDI, sendo a partir desses arquivos que a extração de características foi feita. Especificamente nesse trabalho a extração de características é algo que de grande importância, pois não foi utilizada nenhuma biblioteca já existente, com isso autores foram a fundo na literatura sobre padrões que denotam dificuldades musicais e utilizaram algumas características que consideraram importantes, tendo que desenvolver mais algumas para classificar partituras de piano por nível de dificuldade. Todos os esses padrões e características foram implementados em uma aplicação própria, onde a partir dela foi possível extrair os dados para rodar nos classificadores. Antes da classificação foi utilizado um algoritmo para selecionar as cinco melhores características. Utilizando o conceito de regressão os resultados foram de 38,8% e 39,9% nas bases de 159 e 184 instâncias respectivamente (CHIU; CHEN, 2012). Os resultados representam a porcentagem de partituras classificadas no nível correto.

O Score Analyzer(SÉBASTIEN et al., 2012) é uma ferramenta que foi proposta com o intuito de ajudar no aprendizado eletrônico de partituras para piano. Essa ferramenta é basicamente um analisador de partituras musicais que utiliza o MusicXML para definir de forma automática níveis de dificuldade. Para desenvolver essa ferramenta os autores estudaram primeira a estrutura do MusicXML, para entender e definir quais eram as principais partes da partitura. A partir desse estudo foi definido quais eram os fatores que evidenciavam as dificuldades técnicas instrumentais e a partir disso criar parâmetros que caracterizassem estas dificuldades, sendo assim, puderam identificar sete critérios que definiram como mais relevantes e a partir disso extrair as características: velocidade de reprodução, dedilhado, deslocamento da mão, polifonia, harmonia, ritmo e extensão. Tudo o que é evidenciado pelo protótipo, como partes da partitura e cada nível de dificuldade, precisa, obrigatoriamente passar pela análise de um especialista na área. Para validar o método criado, aplicaram o conceito de Análise de Componentes Principais (PCA) em uma base com 50 partituras musicais do curso de piano clássico do conservatório de Música Francesa. Ao final cada música deveria ser classificada em fácil, médio, difícil e profissional.

É importante ressaltar que o presente estudo, precisa ser considerado novo, pois temos apenas duas pesquisas que se aproximam daquilo que está sendo proposto. Na Tabela 7, podemos observar as diferenças e similaridades entre os trabalhos relacionados.

Na pesquisa do (SÉBASTIEN et al., 2012) o foco é na criação de um protótipo para o auxílio no ensino da música, onde a ferramenta deve analisar uma partitura musical de piano e sugerir níveis de dificuldade para cada parte da partitura, após isso, um especialista em música deve validar a sugestão da aplicação. Em comparação com a pesquisa desenvolvida nessa dissertação, o método citado acima não utiliza aprendizagem de máquina e por isso teve que ser desenvolvido junto a um especialista, para validar todo o algoritmo de extração de característica, sendo válido apenas para piano, pois cada instrumento musical tem níveis e formas de dificuldades diferentes.

Finalizando a comparação entre os trabalhos relacionados, pois temos apenas duas pesquisa que se assemelham com esta, (CHIU; CHEN, 2012) é o que tem o processo metodológico mais próximo com o que está sendo adotado, pois ambos necessitam de partituras musicais rotuladas com os níveis de dificuldade e utilizam aprendizagem de máquina para classificar as partituras. Podemos notar a diferença no quesito da extração de características, onde o autor necessita entender de música para criar o seu próprio extrator, em contrapartida do presente trabalho, onde é utilizado bibliotecas e métodos de extração de características utilizados na literatura. No trabalho atual é utilizado mais de um instrumento musical e os resultados são consideravelmente superiores ao trabalho de 2012.

	Chiu, S.	Sébastien, V.	Trabalho atual
Instrumento musical	Apenas piano	Apenas piano	Piano Violão Violino
Arquivo de entrada	MIDI	MusicXML	MIDI
Extrator de características	Biblioteca própria	Biblioteca própria	jSmbolic e LBP
Bases de dados	159 e 184 partituras	50 partituras	Violão - 56 partituras Violino - 99 partituras Piano - 258 partituras
Níveis de dificuldade	Iniciante Fácil Médio Díficil	Fácil Médio Díficil Profissional	Fácil Médio Díficil
Aprendizagem de Máquina	Linear Regression Multiple Linear Regression Pace Regression Support Vector Regression	N/A	Decision Tree Random Forest Naive Bayes Support Vector Machine K-nearest Neighbor Neural Networks
Resultados	39,9% e 38,8% de acurácia	N/A	Entre 54% e 100% de acurária
Conhecimento musical	Sim	Sim	Não

Tabela 7 – Tabela comparativa entre os trabalhos relacionados
Fonte: Autoria própria

4 Metodologia de Pesquisa

Este capítulo apresenta os passos sobre a abordagem metodológica utilizada no desenvolvimento desta pesquisa. A estrutura da pesquisa é dividida em três fases: Definição dos níveis de dificuldade e rotulação da base de dados (Seção 4.1), Extração das características (Seção 4.2) e Classificação (Seção 4.3). Na Figura 11, temos um diagrama para melhorar a identificação de todos os passos utilizados no método que está sendo proposto.

4.1 Definição dos níveis de dificuldade e rotulação da base de dados

No desenvolvimento da pesquisa de classificação automática de partituras musicais por nível de dificuldade, foi necessário utilizar uma base de dados já existente, encontrada na internet, que é do método Suzuki para violino, e duas bases de dados foram criadas unicamente para esse trabalho, que são do método Henrique Pinto para violão e Leila Fletcher para piano. Para a classificação automática das partituras por nível de dificuldade, foi necessário adicionar detalhes em cada partitura que definem o contexto de cada nível de dificuldade. Isso é necessário para que cada classificador possa aprender o que cada nível representa e, posteriormente, onde cada partitura musical nova deve ser classificada. Identificando esse problema, foi necessário utilizar partituras já classificadas na construção das bases de dados, para servir como exemplo do que cada nível se assemelha. Neste trabalho foram utilizados três bases de dados, cada base de dado refere-se a um método tradicional de ensino da música, Suzuki para violino com 99 partituras, Henrique Pinto para violão com 56 partituras e Leila Fletcher para piano com 254 partituras.

Para que houvesse um padrão na classificação das partituras por níveis de dificuldade, pois cada método de ensino possui uma quantidade diferente de volumes, o que acarretaria em níveis diferentes de dificuldade, foi adotado, com orientação de um especialista, os níveis fácil, médio e difícil.

No método Suzuki foram utilizados os volumes 1 a 10 ((SUZUKI, 2015a), (SUZUKI, 2015c), (SUZUKI, 2015d), (SUZUKI, 2015e), (SUZUKI, 2015f), (SUZUKI, 2015g), (SUZUKI, 2015h), (SUZUKI, 2015i), (SUZUKI, 2015j), (SUZUKI, 2015b)) para a readequação dos níveis de dificuldade, foram agrupados os volume 1, 2 e 3 para que representassem o nível fácil, no nível médio agrupamos os volumes 4, 5, 6 e 7 e, por fim, os volumes 8, 9 e 10 representam o nível difícil, pode-se isso observar na Figura 13.

No método Leila Fletcher foram utilizados os volumes 1 a 6 ((FLETCHER, 1993c), (FLETCHER, 1993f), (FLETCHER, 1993e), (FLETCHER, 1993b), (FLETCHER, 1993a)

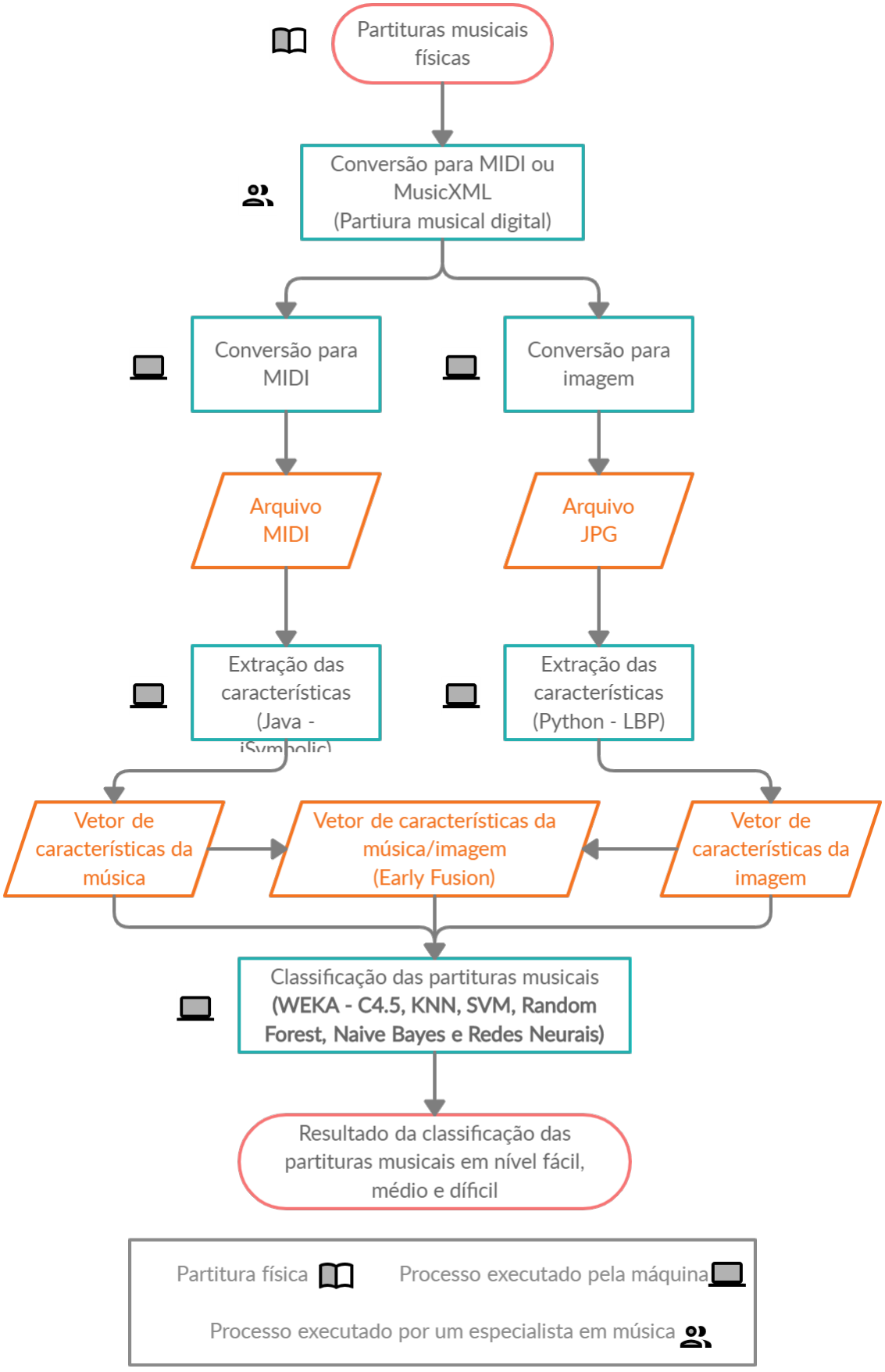

Figura 11 – Processo metodológico da pesquisa
Fonte: Autoria própria

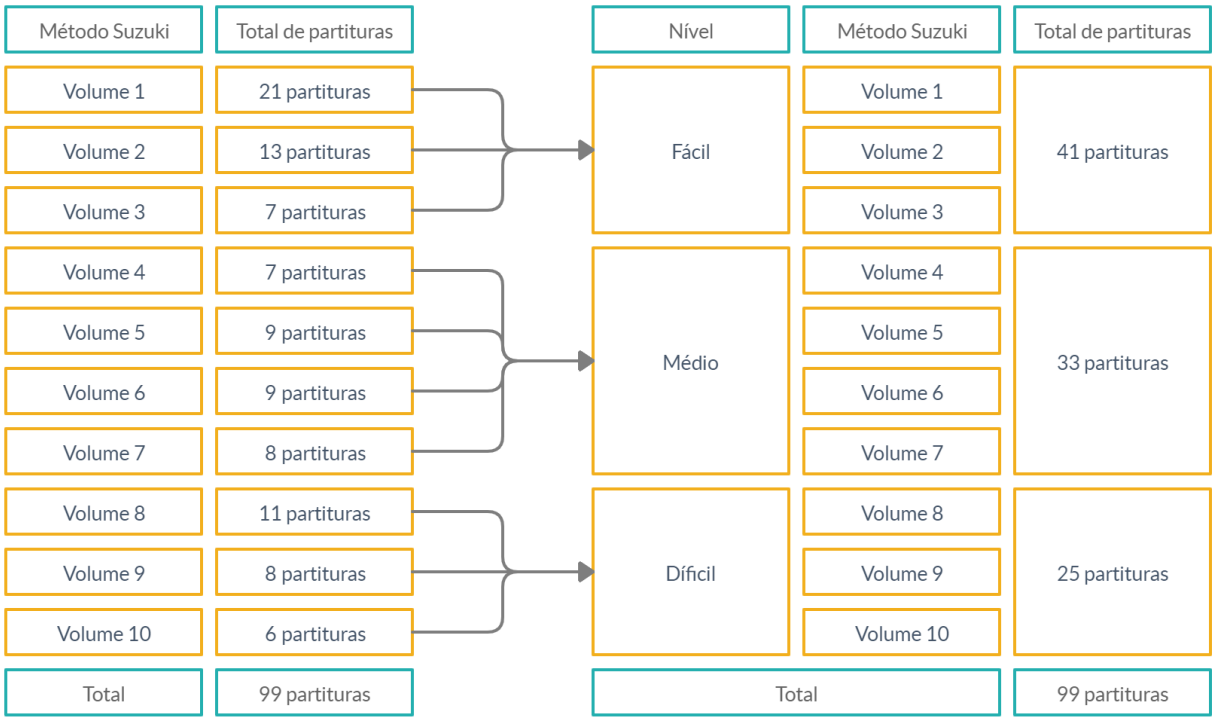

Figura 12 – Readequação dos níveis de dificuldade para o método Suzuki
Fonte: Autoria própria

e (FLETCHER, 1993d)), para a readequação dos níveis de dificuldade, foram agrupados os volume 1 e 2 para que representassem o nível fácil, no nível médio agrupamos os volumes 3 e 4 e, por fim, os volumes 5, 6 representam o nível difícil, pode-se isso observar na Figura 13.

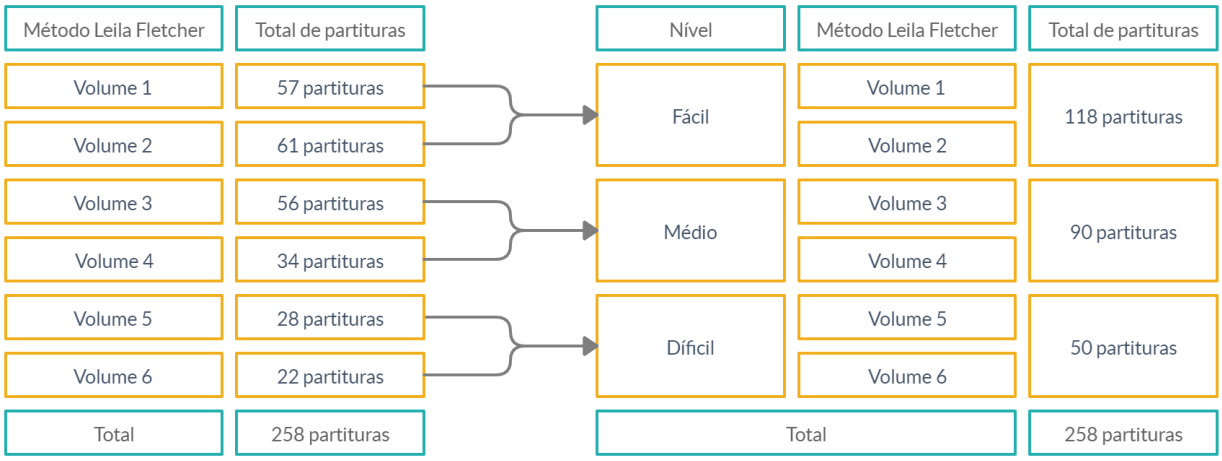

Figura 13 – Readequação dos níveis de dificuldade para o método Leila Fletcher
Fonte: Autoria própria

No método Henrique Pinto foram utilizados os volumes 1 e 2 ((PINTO, 2008a) e (PINTO, 2008b)), para a readequação dos níveis de dificuldade, foi considerado o volume 1 para que representasse o nível fácil e, o nível médio foi representado pelo volume 2, pode-se

isso observar na Figura 14.

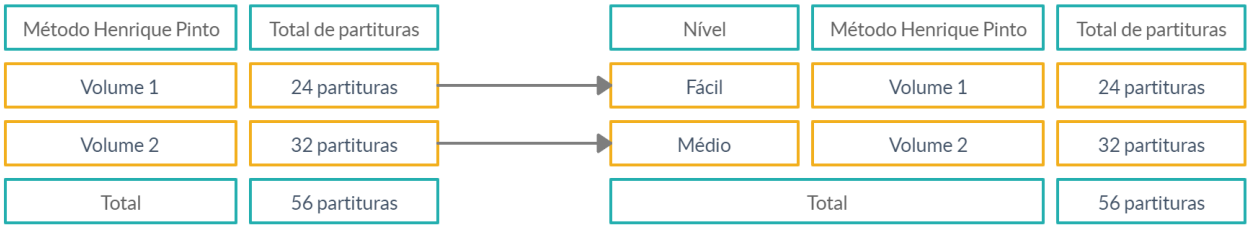

Figura 14 – Readequação dos níveis de dificuldade para o método Henrique Pinto
Fonte: Autoria própria

4.2 Extração das características

A extração das características acontece após a conversão das partituras musicais do formato físico para dois formatos. O primeiro formato que utilizamos é o formato MIDI, um formato musical que será utilizado para extrair as características simbólicas de alto nível da música. A segunda forma de extração acontece através da própria imagem da partitura, que pode ser gerada por diversos softwares de leitura do MusicXML ou do MIDI, essa imagem é totalmente limpa, pois ela contém apenas símbolos da partitura musical. Conforme a Figura 15 podemos ver um exemplo da imagem que será utilizada na extração das características e um exemplo de uma imagem escaneada direto do livro, que não será utilizada no trabalho, pois contém diversos fatores que podem atrapalhar na classificação das partituras, como níveis de luminosidade diferentes, tons de envelhecimento e etc. O trabalho propõe a extração das características de duas formas diferentes, conforme as Seções 4.2.1 e 4.2.2.

4.2.1 Extração das características do arquivo musical

Para extrair as características musicais, foi utilizada a biblioteca em java chamada *jSymbolic* dentro de uma aplicação própria. A biblioteca necessita do arquivo em formato MIDI, por isso foi necessário converter as partituras físicas para formato digital, a partir dos arquivos de entrada, são extraídas através da biblioteca, por processo interno, 1228 características de alto nível simbólico de valores numéricos de cada partitura musical, além de uma classe que indica o nível de dificuldade em fácil, médio e difícil.

O jSymbolic por sua vez, lê os arquivos de entrada e extrai as características para o formato XML com uma própria formatação, devido a flexibilidade deste tipo de formato, como podemos observar na Figura 16. Após a geração do arquivo XML pela biblioteca jSymbolic, é necessário ler este arquivo e converter para um formato que os classificadores consigam entender, neste projeto o arquivo final foi convertido para ARFF.

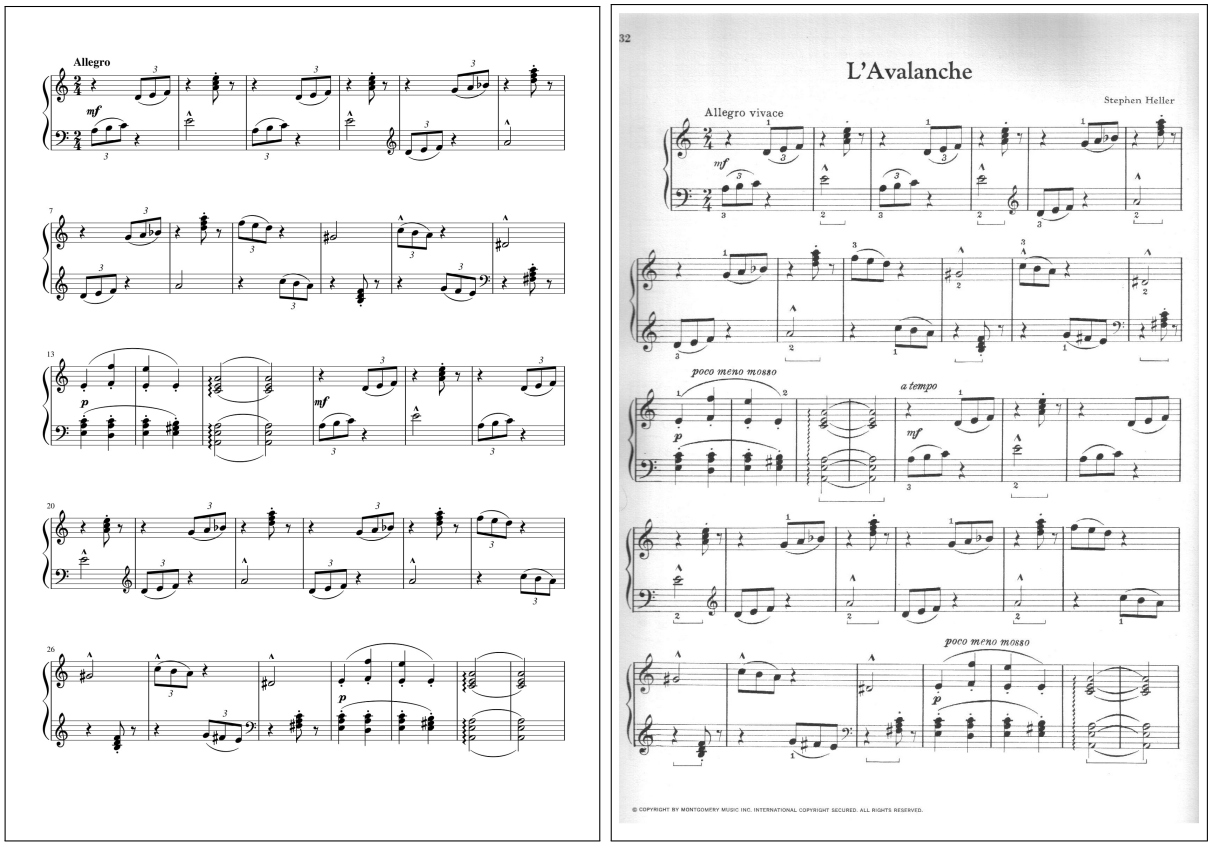

Figura 15 – Exemplo da partitura musical L'Avalanche 1
Fonte: (SUZUKI, 2015c)

```
<?xml version="1.0"?>                                   Declaração ARFF

<feature_vector_file>                                   Música em formato MIDI
   <comments></comments>
    <data_set>
        <data_set_id>C:\Users\223240969-vol1_1-1-twinkle_twinkle_little_star_var-a.mid</data_set_id>
        <feature>
            <name>Prevalence of Most Common Pitch</name>        Dado simbólico
            <v>2,5E-1</v>                               Valor do dado simbólico
        </feature>
        <feature>
            <name>Prevalence of Most Common Pitch Class</name>
            <v>2,5E-1</v>
        </feature>
        <feature>
            <name>Relative Prevalence of Top Pitches</name>
            <v>6,667E-1</v>
        </feature>
```

Figura 16 – Exemplo da estrutura do arquivo XML gerado pelo jSymbolic para a música da Figura 1

Fonte: Autoria própria

4.2.1.1 Características

É importante ressaltar que nessa pesquisa não foi utilizado nenhum algoritmo para realizar a seleção das melhores características que definem o nível de dificuldade e também, não foi utilizado nenhum especialista da área de música para indicar quais características mais impactariam na escolha do níveis de dificuldade, portanto, para esse experimento foram utilizadas todas as características que o extrator disponibiliza. Segundo (MCKAY; FUJINAGA, 2006),as características extraídas estão agrupadas em instrumentação, textura, ritmo, estatística da nota, melodia, dinâmica e acordes e podem ser observadas em sua totalidade em (MCKAY, 2018). Na Figura 17 podemos observar todas as características que o *jSymbolic* disponibiliza na versão 2.0.

4.2.2 Extração das características da imagem

Para extrair as características da imagem da partitura musical, foi utilizado o LBP (Local Binary Pattern) (AHONEN; HADID; PIETIKAINEN, 2006), que é um algoritmo que visa analisar a textura da imagem. O LBP tenta extrair característica estruturais imagem, através de padrões binários locais. Isso acontece baseando-se na vizinhança dos pixels da imagem. Devido à sua simplicidade computacional esse algoritmo pode ser implementado em qualquer linguagem de computador, neste trabalho o algoritmo foi implementado em Python.

O formato de entrada para o extrator é o JPG, por isso foi necessário mais um processo, para converter o MusicXML para JPG. A partir da entrada dos arquivos, são extraídas 827 características de valores numéricos da imagem de cada partitura musical, além de uma classe que indica o seu nível de dificuldade em fácil, médio e difícil. Esses valores junto com a classe de cada partitura musical, são agrupados e ao final formam um arquivo ARFF para a entrada nos classificadores. É importante lembrar que apenas a imagem digital foi utilizada nessa trabalho, a imagem que foi gerada após a transcrição das partituras digitais em softwares de notação musical.

4.2.3 Junção dos vetores de música e imagem - Early Fusion

Com intuito de avançarmos ainda mais na pesquisa e tentar sanar a dúvidas inicias sobre a classificação automática de partituras musicais por nível de dificuldade, foi utilizada a técnica de *Early Fusion* para a fusão entre os vetores de características da música e da imagem da partitura, afim de complementar informações faltantes em algum dos vetores (MIDI e imagem). Para isso, após a geração dos dois vetores inicias, vetor das características do MIDI e o vetor das características da imagem digital, uma fusão foi realizada, a fusão é a junção do vetor MIDI com o vetor da imagem digital, como podemos observar na Figura 18.

OVERALL PITCH STATISTICS
Basic Pitch Histogram
Pitch Class Histogram
Folded Fifths Pitch Class Histogram
Prevalence of Most Common Pitch
Prevalence of Most Common Pitch Class
Relative Prevalence of Top Pitches
Relative Prevalence of Top Pitch Classes
Interval Between Most Prevalent Pitches
Interval Between Most Prevalent Pitch Classes
Number of Common Pitches
Pitch Variety
Pitch Class Variety
Range
Most Common Pitch
Mean Pitch
Importance of Bass Register
Importance of Middle Register
Importance of High Register
Most Common Pitch Class
Dominant Spread
Strong Tonal Centres
Major or Minor
Glissando Prevalence
Average Range of Glissandos
Vibrato Prevalence
Microtone Prevalence

MELODIC INTERVALS
Melodic Interval Histogram
Most Common Melodic Interval
Mean Melodic Interval
Number of Common Melodic Intervals
Distance Between Most Prevalent Melodic Intervals
Prevalence of Most Common Melodic Interval
Relative Prevalence of Most Common Melodic Intervals
Amount of Arpeggiation
Repeated Notes
Chromatic Motion
Stepwise Motion
Melodic Thirds
Melocid Perfect Fourths
Melodic Tritones
Melodic Fifths
Melodic Sixths
Melodic Sevenths
Melodic Octaves
Melodic Large Intervals
Minor Major Melodic Third Ratio
Melodic Embellishments
Direction of Melodic Motion
Average Length of Melodic Arcs
Average Interval Spanned by Melodic Arcs
Melodic Pitch Variety

CHORDS AND VERTICAL INTERVALS
Vertical Interval Histogram
Wrapped Vertical Interval Histogram
Chord Type Histogram
Average Number of Simultaneous Pitch Classes
Variability of Number of Simultaneous Pitch Classes
Average Number of Simultaneous Pitches
Variability of Number of Simultaneous Pitches
Most Common Vertical Interval
Second Most Common Vertical Interval
Distance Between Two Most Common Vertical Intervals
Prevalence of Most Common Vertical
Prevalence of Second Most Common Vertical Interval
Prevalence Ratio of Two Most Common Vertical Intervals
Vertical Unisons
Vertical Minor Seconds
Vertical Thirds
Vertical Tritones
Vertical Perfect Fourths
Vertical Perfect Fifths
Vertical Sixths
Vertical Sevenths
Vertical Octaves
Perfect Vertical Intervals
Vertical Dissonance Ratio
Vertical Minor Third Prevalence
Vertical Major Third Prevalence
Chord Duration
Partial Chords
Standard Triads
Diminished and Augmented Triads
Dominant Seventh Chords
Seventh Chords
Non-Standard
Complex Chords
Minor Major Triad Ratio

RHYTHM
Beat Histogram
Strongest Rhythmic Pulse
Second Strongest Rhythmic Pulse
Harmonicity of Two Strongest Rhythmic Pulses
Strength of Strongest Rhythmic Pulse
Strength of Second Strongest Rhythmic Pulse
Strength Ratio of Two Strongest Rhythmic
Combined Strength of Two Strongest Rhythmic Pulses
Number of Strong Rhythmic Pulses
R-10 Number of Moderate Rhythmic Pulses
Number of Relatively Strong Rhythmic Pulses
Rhythmic Looseness
Polyrhythms
Rhythmic Variability
Note Density
Note Density Variability
Average Note Duration
Variability of Note Durations
Maximum Note Duration
Minimum Note Duration
Amount of Staccato
Average Time Between Attacks
Variability of Time Between Attacks
Average Time Between Attacks for Each Voice
Average Variability of Time Between Attacks for Each Voice
Complete Rests
Longest Complete Rest
Average Rest Fraction Per Voice
Variability Across Voices of Total Rests Per Voice
Initial Tempo
Compound or Simple Meter
Triple Meter
Quintuple Meter
Metrical Diversity
Duration

INSTRUMENTATION
Pitched Instruments Present
Unpitched Instruments Present
Note Prevalence of Pitched Instruments
Note Prevalence of Unpitched Instruments
Time Prevalence of Pitched Instruments
Variability of Note Prevalence of Pitched Instruments
Variability of Note Prevalence of Unpitched Instruments
Number of Pitched Instruments
Number of Unpitched Instruments
Percussion Instrument Prevalence
String Keyboard Prevalence
Acoustic Guitar Prevalence
Electric Guitar Prevalence
Violin Prevalence
Saxophone Prevalence
Brass Prevalence
Woodwinds Prevalence
Orchestral Strings Prevalence
String Ensemble Prevalence
Electric Instrument Prevalence

TEXTURE
Maximum Number of Independent Voices
Average Number of Independent Voices
Variability of Number of Independent
Voice Equality – Number of Notes
Voice Equality – Note Duration
Voice Equality – Dynamics
Voice Equality – Melodic Leaps
Voice Equality – Range
Importance of Loudest Voice
Relative Range of Loudest Voice
Relative Range Isolation of Loudest Voice
Relative Range of Highest Line
Relative Note Density of Highest Line
Relative Note Durations of Lowest Line
Relative Size of Melodic Intervals in Lowest Line
Voice Overlap
Voice Separation
Variability of Voice Separation
Parallel Motion
Similar Motion
Contrary Motion
Oblique
Parallel Fifths
Parallel Octaves

DYNAMICS
Dynamic Range
Variation of Dynamics
Variation of Dynamics in Each Voice
Average Note to Note Change in Dynamics

Figura 17 – Todas as características disponibilizadas pelo jSymbolic
Fonte: Autoria própria

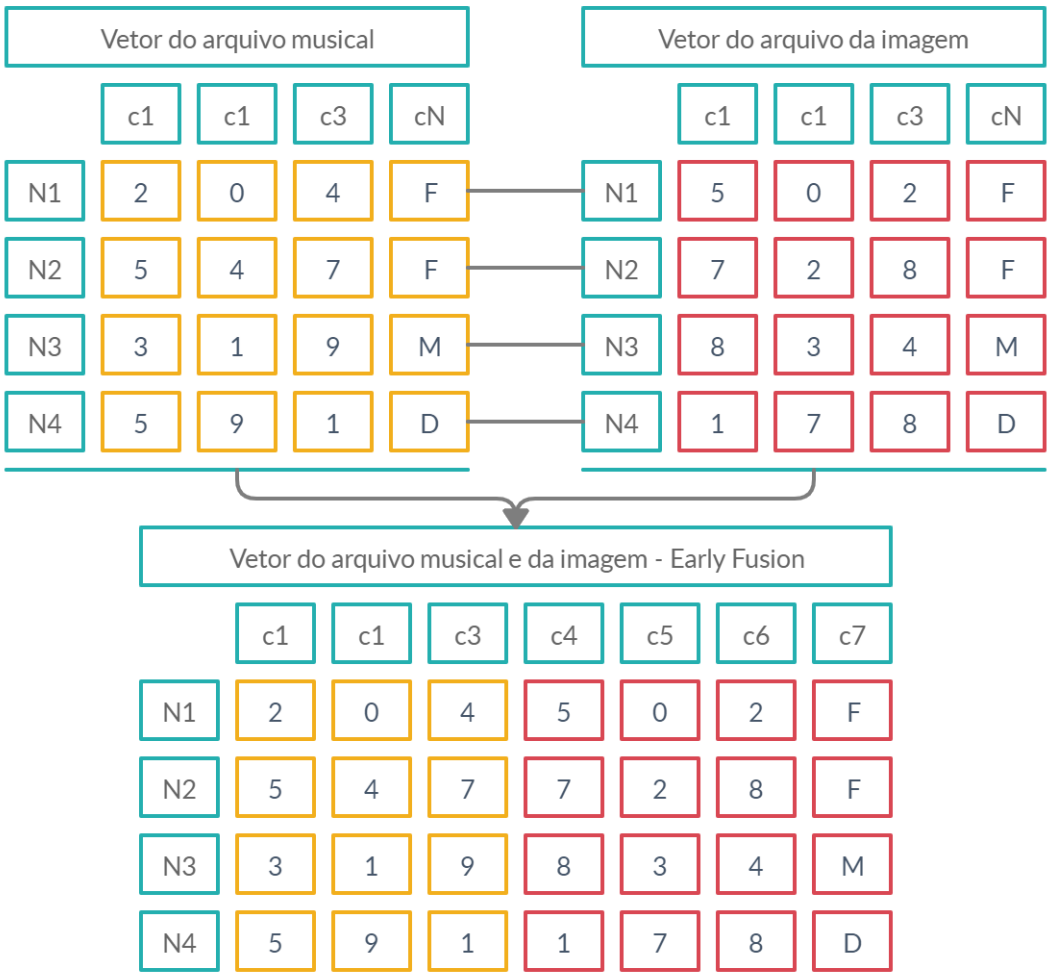

Figura 18 – Exemplo da junção entre dois vetores de características diferentes
Fonte: Autoria própria

4.3 Classificação

Para classificar uma partitura musical em seu nível de dificuldade foi utilizado o WEKA, que contém os classificadores já apresentados na Seção 2.2. Após a extração das características musicais e da imagem de cada partitura é gerado um arquivo para cada tipo de característica no formato ARFF.

Com o arquivo ARFF gerado, é possível utilizar o WEKA para treinar e classificar as partituras musicais em seus níveis de dificuldade. Nenhum dos parâmetros dos classificadores foram alterados, ou seja, todo o método foi testado com as configurações padrões de cada classificador dentro do WEKA. Para o particionamento dos dados em conjuntos de treinamento e teste, empregamos o método *leave-one-out* (CAWLEY; TALBOT, 2003), que é uma aplicação de *n-fold*, que faz uma subdivisão do conjunto de dados em partes iguais. Cada subconjunto é usado para testes e os outros para treinamento e isso é repetido várias vezes. No *leave-one-out*, o número definido em n representa o número de instâncias usadas. Neste experimento foram utilizadas no *leave-one-out* a mesma quantidade de

partituras que temos em cada método, ou seja, cada algoritmo foi treinado noventa e noves no método suzuki, duzentas e cinquenta e oito vezes no método Leila Fletcher e cinquenta e seis vezes para o método do Henrique Pinto.

No intuito de evoluir ao máximo no campo da pesquisa, alguns experimentos foram realizados, afim de responder as questões de pesquisa definidas para esse trabalho. A Figura 19 exemplifica a quantidade de bases de dados, formas de extração das características e classificadores utilizados nessa pesquisa. Abaixo detalhes da pesquisa:

- Quatro bases de dados: Suzuki para violino, Henrique Pinto para violão, Leila Fletcher para piano e uma base com todas as partituras de todos de todos os instrumentos musicais.
- Três vetores de características: vetor de características do arquivo MIDI, vetor de características da imagem digital e o vetor de características entre a fusão do vetor MIDI e da imagem digital.
- Seis classificadores: C4.5, Random Forest, KNN, Naive Bayes, SVM e Rede Neural.

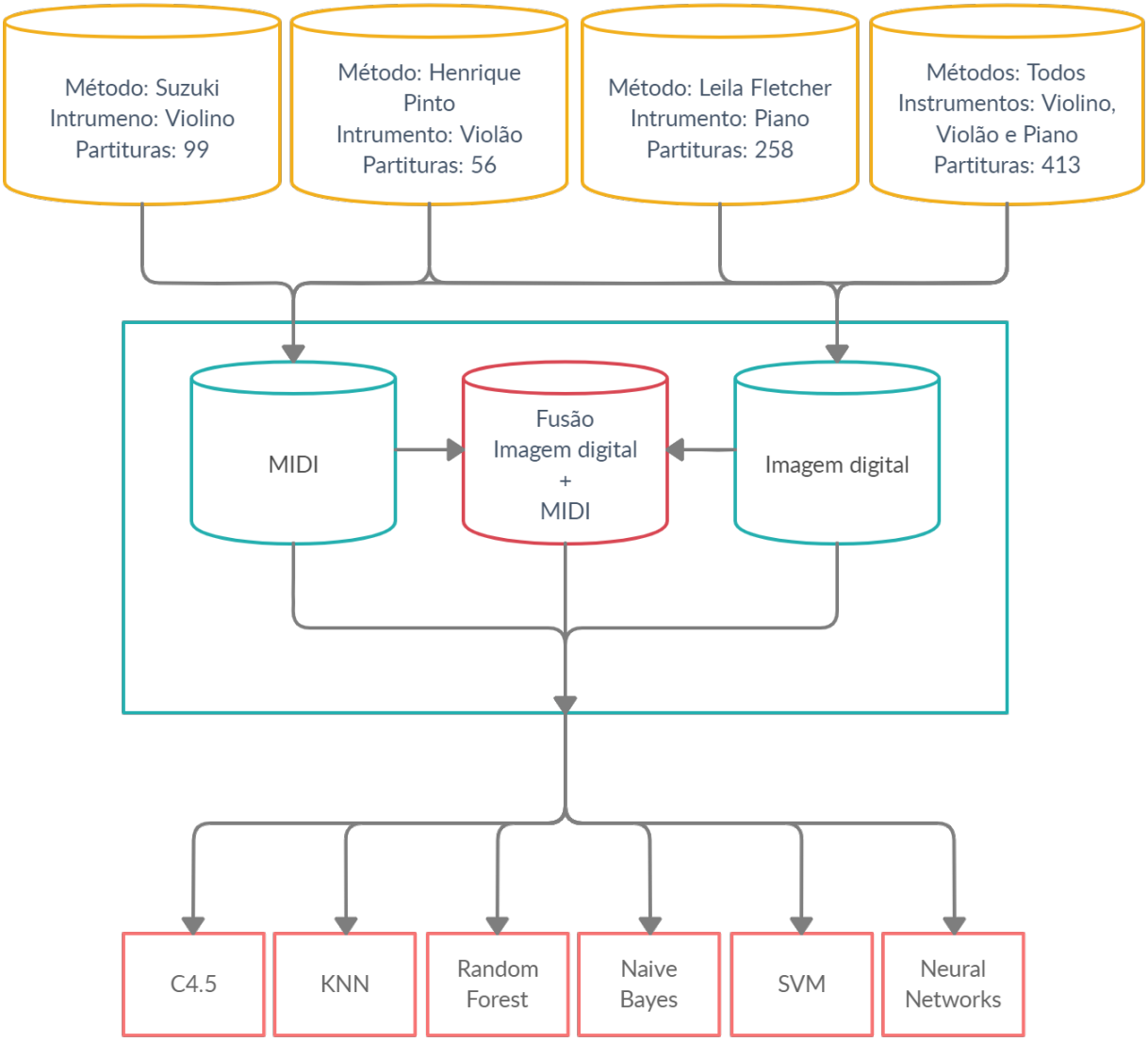

Figura 19 – Bases e dados e classificadores utilizados na pesquisa
Fonte: Autoria própria

5 Resultados e Discussões

Neste capítulo são apresentados os experimentos realizados ao longo do desenvolvimento do trabalho. É importante ressaltar que para evitar qualquer tipo de tendência e aumentar o campo de pesquisa desse trabalho três bases de dados com três instrumentos musicais diferentes foram utilizadas, Suzuki para violino, Henrique Pinto para violão e Leila Fletcher para piano.

Os experimentos foram realizados com os classificadores apresentados na Seção 2.2, que são o C4.5, Rede Neural, Naive Bayes, Random Forest, SVM e KNN, usando as bases de dados apresentadas na Seção 4.1, com a configuração apresentada na Seção 4.3.

Nesta seção, estamos interessados em responder as seguintes perguntas:

- 1. É possível classificar partituras musicais por nível de dificuldade utilizando aprendizagem de máquina?
- 2. Existe diferença entre a utilização de dados de alto nível simbólico do áudio da partitura musical, em relação a dados da imagem da partitura musical?
- 3. O que acontece com os resultados de classificação quando as bases dos diferentes instrumentos são unificadas em uma única base?
- 4. Qual é o impacto de treinar classificadores com a base de dados de um instrumento musical, por exemplo piano, e testar com a base de dados de outro instrumento musical, por exemplo violino?

Para responder as perguntas 1 e 2 será utilizada a Tabela 8, que apresenta o resultado geral da classificação das bases de violino, violão e piano, utilizando as características do formato MIDI, da imagem digital e da fusão entre o formato MIDI e a imagem digital.

Os resultados para o classificador de Rede Neural que não foram apresentados nesse documento, ainda estão sendo executados e se encontram com um "-". No documento final, junto com as correções apresentadas na banca, os resultados já estarão disponíveis.

5.1 Classificacão automática de partituras por nível de dificuldade utilizando características simbólicas de alto nível

Nesta seção, estamos interessados em responder a seguinte pergunta: É possível classificar partituras musicais por nível de dificuldade utilizando aprendizagem de máquina?

Instrumento	Classificadores	MIDI	Imagem Digital	MIDI + Imagem Digital
VIOLINO	C4.5	74%	64%	74%
	Naive Bayes	74%	59%	71%
	Random Forest	83%	66%	68%
	SVM	83%	70%	80%
	KNN	77%	69%	72%
	Rede Neural	**85%**	**73%**	**82%**
PIANO	C4.5	74%	70%	69%
	Naive Bayes	74%	70%	75%
	Random Forest	81%	**74%**	75%
	SVM	81%	72%	**77%**
	KNN	70%	60%	72%
	Rede Neural	**83%**	67%	-
VIOLÃO	C4.5	79%	54%	75%
	Naive Bayes	75%	70%	68%
	Random Forest	77%	70%	70%
	SVM	77%	**73%**	**77%**
	KNN	61%	70%	71%
	Rede Neural	**84%**	**73%**	68%

Tabela 8 – Resultado geral da acurácia das classificações por instrumento musical, com violino, piano e violão

Fonte: Autoria própria

Para responder essa pergunta, iremos analisar apenas a primeira coluna da Tabela 8, que é referente a classificação das partituras musicais utilizando as características do formato MIDI, que são as características consideradas de alto nível simbólico.

Vão ser utilizadas as bases Suzuki para violino, Leila Fletcher para piano e Henrique Pinto para violão, com respectivamente, 99, 56 e 258 partituras musicais, apresentadas na Seção 4.2.1.

Nesta seção será analisada somente a primeira coluna (MIDI) de resultados da Tabela 8. Nesta coluna é possível observar que o classificador que melhor classifica partituras musicais por nível de dificuldade, utilizando as características do formato MIDI como entrada, é o classificador de Rede Neural. É possível chegar a essa conclusão, pois a Rede Neural é o classificador que tem a melhor taxa de acerto nas 3 bases de dados. Sendo assim, na base de dados de violino a acurácia foi de 85%, para piano temos uma taxa de acerto de 83% e por fim, 84% foi a taxa de acerto na base de dados de violão.

Através da Figura 20 podemos observar que na base para violino utilizando Rede Neural, que é o melhor classificador entre todos os utilizados, a maior quantidade de erros acontece no nível intermediário. Já Figura 22 podemos observar que na base para violão com o mesmo classificador, a maior quantidade de erros acontece no último nível, que

Fácil	Médio	Díficil	
38	3	0	Fácil
3	27	3	Médio
1	5	19	Díficil

(a) Midi

Fácil	Médio	Díficil	
35	6	0	Fácil
9	20	4	Médio
1	6	18	Díficil

(b) Imagem Digital

Fácil	Médio	Díficil	
36	4	1	Fácil
3	27	3	Médio
1	5	19	Díficil

(c) Midi + Imagem Digital

Figura 20 – Matriz da base para violino com o classificador de Rede Neural

Fácil	Médio	
19	5	Fácil
4	28	Médio

(a) Midi

Fácil	Médio	
14	10	Fácil
5	27	Médio

(b) Imagem Digital

Fácil	Médio	
14	10	Fácil
8	24	Médio

(c) Midi + Imagem Digital

Figura 21 – Matriz da base para violão com o classificador de Rede Neural

Fácil	Médio	Díficil	
109	9	0	Fácil
5	70	15	Médio
1	14	35	Díficil

(a) Midi

Fácil	Médio	Díficil	
102	14	2	Fácil
22	43	25	Médio
1	21	28	Díficil

(b) Imagem Digital

Figura 22 – Matriz da base para piano com o classificador de Rede Neural

também é o intermediário, pois essa base contém apenas dois níveis.

É importante ressaltar que na base de dados para violão, apenas dois níveis de dificuldade foram considerados, e na base de dados de violino e piano três níveis de dificuldade foram considerados, na Seção 4.1 podemos observar a readequação dos níveis de dificuldade.

Outra observação importante, é de como os classificadores acertam de forma parecida independente da base de dados, utilizando as características do formato MIDI. A exceção é o classificador KNN, que tem taxas de acerto mais distantes.

Outro fator importante e que deve ser considerado para a classificação de partituras musicais por níveis de dificuldade, é de como cada classificador se comporta dentro de cada nível de dificuldade. Podemos observar os resultados na Tabela 9, que mostra a acurácia de cada classificador para cada nível de dificuldade, utilizando apenas o formato de entrada MIDI.

Para analisar como é o comportamento de cada classificador, iremos analisar cada instrumento musical de forma individual, começado com a base do método Suzuki para violino. Se analisarmos os classificadores junto aos níveis de dificuldade, podemos ver que os resultados variam bastante entre cada nível de dificuldade e classificador, mas existe um padrão que se repete na maioria dos classificadores. A taxa de acerto é na maioria das vezes superior para o nível fácil, comparado aos outros dois níveis de dificuldade. Podemos observar isso com mais clareza na última linha do instrumento violino, onde

Instrumento	Classificadores	Fácil	Médio	Difícil
VIOLINO	C4.5	90%	67%	56%
	Naive Bayes	76%	58%	**92%**
	Random Forest	85%	**91%**	68%
	SVM	88%	79%	80%
	KNN	**95%**	70%	56%
	Rede Neural	93%	82%	76%
	Média	88%	74%	71%
PIANO	C4.5	86%	63%	66%
	Naive Bayes	86%	59%	**72%**
	Random Forest	**92%**	**79%**	58%
	SVM	91%	78%	66%
	KNN	84%	63%	48%
	Rede Neural	**92%**	78%	70%
	Média	89%	70%	63%
VIOLÃO	C4.5	75%	75%	N/A
	Naive Bayes	67%	81%	N/A
	Random Forest	63%	**88%**	N/A
	SVM	75%	78%	N/A
	KNN	63%	59%	N/A
	Rede Neural	**79%**	**88%**	N/A
	Média	70%	78%	N/A

Tabela 9 – Taxa de acerto por nível de dificuldade do formato MIDI
Fonte: Autoria própria

temos a média da taxa de acerto para cada nível de dificuldade. Logo podemos dizer que os classificadores acertam mais para o nível fácil, do que para o médio e difícil. Um dos fatores que pode estar interferindo para que isso aconteça, é que para o nível fácil existem 41 partituras, para o nível médio temos 33 partituras e para o nível difícil, contamos com 25 partituras, totalizando 99 partituras musicais. Isso deve ser analisado com mais clareza, mas os classificadores devem estar aprendendo melhor para o primeiro nível do que para os outros.

Para piano, podemos observar o mesmo padrão. Nesse caso, de forma mais clara ainda, pois cinco dos seis classificadores tem sempre a taxa de acerto mais alto no nível fácil, depois no nível médio e por fim, no nível difícil, isso só não acontece no classificador C4.5, onde a taxa de acerto é maior no fácil, depois no difícil e por último no nível médio.

Analisando o último instrumento musical, o violão, o padrão analisado nos dois instrumentos musicais acima, não se repete, onde 4 dos 6 classificadores tem uma acurácia maior no nível médio. Para o C4.5 temos um empate na taxa de acerto em 75% e para o KNN a acurácia é maior no primeiro nível de dificuldade.

Por fim, os resultados que se encontram em negrito na Tabela 9, são os classificadores

que obtiveram a melhor taxa de acerto, para cada instrumento musical e também por nível dificuldade. Para o nível difícil, Naive Bayes teve a melhor performance, tanto para violino quanto para piano. Para o nível médio o Random Forest teve o melhor resultado em todos as bases de dados. Para o nível fácil, as três bases de dados tiveram classificadores diferentes, sendo eles, o KNN para violino, Random Forest e Rede Neural para piano e Rede Neural para o violão.

5.2 Formato MIDI Vs. Imagem digital

Iremos responder nessa seção, a seguinte pergunta: Existe diferença entre a utilização de dados de alto nível simbólico do áudio da partitura musical, em relação a dados da imagem da partitura musical? Para responder essa pergunta, iremos analisar a primeira e a segunda coluna da Tabela 8, que são referentes a classificação das partituras musicais utilizando as características do formato MIDI e da imagem digital, respectivamente.

Como pudemos observar na Seção 5.1, a primeira coluna refere-se a classificação de partituras musicais através da característica do formato MIDI. Nessa seção iremos analisar, se gerar a imagem da partitura a partir do formato MIDI ou do MusicXML, utilizando um programa de notação musical é realmente necessário.

Para analisarmos se as características do MIDI ou da imagem digital tem resultados melhores para classificar as partituras por nível de dificuldade, é importante recapitular sobre o que se trata a imagem digital. A imagem digital é a imagem da partitura gerada a partir de arquivos no formato MIDI ou MusicXML, que são utilizados como entrada em programas de notação musical, como por exemplo, o MuseScore, ou seja, quando a partitura é transcrita para MIDI ou MusicXML, também é possível gerar uma imagem da partitura, essa é uma imagem que consideramos limpa, pois contém elementos que apenas compõe a partitura, não contendo todos os elementos de um livro físico, como explicações e exemplos e também sombras e níveis de amarelamento como os livros físicos apresentam.

Ao observarmos o resultado após a classificação das partituras, comparando as colunas ”MIDI” e da ”Imagem Digital” da Tabela 8, podemos observar que as características do formato MIDI tem um resultado melhor do que as características da imagem digital, pois apenas em um dos experimentos na base de dados para violão, com o classificador KNN, a imagem digital teve um resultado superior ao formato MIDI, sendo de 70% de taxa de acerto para a imagem digital, contra 61% do formato MIDI.

Olhando para a Tabela 10 e analisando apenas o resultado que estão marcados em negrito, pois determinam a melhor taxa de acerto para aquele nível de dificuldade e comparando o formato MIDI com a imagem digital, é possível observar que para o violino, na coluna MIDI, o classificador KNN tem uma taxa de acerto 10% superior ao melhor classificador do nível fácil para a coluna da imagem digital, sendo a Rede Neural com 85%

		MIDI				Imagem digital		
Instrumento	Classificadores	Fácil	Médio	Difícil		Fácil	Médio	Difícil
VIOLINO	C4.5	90%	67%	56%		83%	45%	56%
	Naive Bayes	76%	58%	**92%**		68%	52%	52%
	Random Forest	85%	**91%**	68%		80%	58%	56%
	SVM	88%	79%	80%		78%	**64%**	68%
	KNN	**95%**	70%	56%		80%	**64%**	56%
	Rede Neural	93%	82%	76%		**85%**	61%	**72%**
	Média	88%	74%	71%		79%	57%	60%
PIANO	C4.5	86%	63%	66%		87%	58%	52%
	Naive Bayes	86%	59%	**72%**		87%	49%	**68%**
	Random Forest	**92%**	**79%**	58%		88%	**61%**	64%
	SVM	91%	78%	66%		**90%**	60%	54%
	KNN	84%	63%	48%		77%	37%	64%
	Rede Neural	**92%**	78%	70%		86%	48%	56%
	Média	89%	70%	63%		86%	52%	60%
VIOLÃO	C4.5	75%	75%	N/A		50%	56%	-
	Naive Bayes	67%	81%	N/A		46%	**88%**	-
	Random Forest	63%	**88%**	N/A		46%	**88%**	-
	SVM	75%	78%	N/A		**58%**	84%	-
	KNN	63%	59%	N/A		54%	81%	-
	Rede Neural	**79%**	**88%**	N/A		**58%**	84%	-
	Média	70%	78%	N/A		52%	80%	-

Tabela 10 – Taxa de acerto por nível de dificuldade do formato MIDI e da imagem digital
Fonte: Autoria própria

de acurácia. Isso também se repete para os níveis médio e difícil, onde o classificador com a melhor taxa de acerto do MIDI é superior aos melhores classificadores do nível médio e difícil da imagem digital.

Para as bases de piano e violão, em todos os níveis de dificuldade, a taxa de acerto do formato MIDI é superior ou igual aos melhores classificadores da imagem digital.

Sendo assim, podemos concluir que utilizar o formato MIDI e extrair as características utilizando a biblioteca jSymbolic, obtém-se resultados melhores do que gerar a imagem digital da partitura, a partir do MIDI ou MusicXML e utilizar o LBP para extrair as características da imagem. Podemos observar o resultado compilado na Tabela 11, onde em apenas um dos casos o a imagem digital tem uma taxa de acerto superior ao formato MIDI.

Instrumento	Classificadores	MIDI	Imagem Digital	MIDI Vs. Imagem Digital
VIOLINO	C4.5	74%	64%	MIDI
	Naive Bayes	74%	59%	MIDI
	Random Forest	83%	66%	MIDI
	SVM	83%	70%	MIDI
	KNN	77%	69%	MIDI
	Rede Neural	85%	73%	MIDI
PIANO	C4.5	74%	70%	MIDI
	Naive Bayes	74%	70%	MIDI
	Random Forest	81%	74%	MIDI
	SVM	81%	72%	MIDI
	KNN	70%	60%	MIDI
	Rede Neural	83%	67%	MIDI
VIOLÃO	C4.5	79%	54%	MIDI
	Naive Bayes	75%	70%	MIDI
	Random Forest	77%	70%	MIDI
	SVM	77%	73%	MIDI
	KNN	61%	70%	Imagem Digital
	Rede Neural	84%	73%	MIDI

Tabela 11 – Resultado da comparação entre a taxa de acerto do formato MIDI e Imagem Digital

Fonte: Autoria própria

5.3 Fusão entre as bases de dados de instrumentos diferentes

Nesta seção, estamos interessados em responder a seguinte pergunta: O que acontece com os resultados de classificação quando as bases dos diferentes instrumentos são unificadas em uma única base?

Para responder essa questão de pesquisa vamos analisar a Tabela 12, que é a extensão da Tabela 8. Nessa tabela adicionamos o resultado das classificações da base onde foram unidos todos os instrumentos musicais. Essa base conta com 413 partituras musicais, sendo 99 do método Suzuki para violino, 56 do método Henrique Pinto para piano e 258 do método Leila Fletcher para piano, considerando o nível fácil, médio e difícil. No total 183 partituras estão rotuladas para o nível fácil, 155 para o nível médio e 75 para o nível difícil.

Ao analisarmos os resultados da base com todos os instrumentos musicais, em comparação com a base de dados de instrumentos únicos, é possível perceber que a unificação das bases de dados, obtém um resultado inferior em 2 das 3 abordagens proposta nessa pesquisa. A única abordagem onde o resultado maior é do que as demais é abordagem, é quando usamos a fusão do formato MIDI com a imagem digital, contemplando a acurácia de 93%.

Instrumento	Classificadores	MIDI	Imagem Digital	MIDI + Imagem Digital
VIOLINO	C4.5	74%	64%	74%
	Naive Bayes	74%	59%	71%
	Random Forest	83%	66%	68%
	SVM	83%	70%	80%
	KNN	77%	69%	72%
	Rede Neural	**85%**	**73%**	**82%**
PIANO	C4.5	74%	70%	69%
	Naive Bayes	74%	70%	75%
	Random Forest	81%	**74%**	75%
	SVM	81%	72%	**77%**
	KNN	70%	60%	72%
	Rede Neural	**83%**	67%	-
VIOLÃO	C4.5	79%	54%	75%
	Naive Bayes	75%	70%	68%
	Random Forest	77%	70%	70%
	SVM	77%	**73%**	**77%**
	KNN	61%	70%	71%
	Rede Neural	**84%**	**73%**	68%
VIOLINO PIANO VIOLÃO	C4.5	70%	61%	85%
	Naive Bayes	67%	63%	58%
	Random Forest	78%	**71%**	**93%**
	SVM	**79%**	70%	82%
	KNN	70%	63%	91%
	Rede Neural	-	-	-

Tabela 12 – Resultado geral da acurácia das classificações por instrumento musical, com violino, piano, violão e uma base com todos os intrumentos musicais
Fonte: Autoria própria

Observando a Seção 4.1, onde os métodos de ensino musical possuem diferentes níveis de dificuldade, por se tratar de diferentes instrumentos musicais e, no método que está sendo proposto haja uma readequação dos níveis de dificuldade para padronizarmos os níveis de dificuldade em fácil, médio e difícil, não significa que o nível fácil em piano possui as mesmas características do nível fácil em violão ou em violino e isso também acontece para os demais níveis de dificuldade.

Pelos motivos citados acima, os classificadores possivelmente não entendem as questões específicas cada instrumento musical e as taxas de acerto são menores do que as taxas com as bases musicais de instrumentos únicos.

5.4 Treino e teste em bases com intrumentos musicais diferentes

Iremos responder nessa seção, a seguinte pergunta: Qual é o impacto de treinar classificadores com a base de dados de um instrumento musical, por exemplo piano, e testar com a base de dados de outro instrumento musical, por exemplo violino?

Afim de realizar algo comum em aprendizagem de máquina, que é o teste e treino de classificadores de bases diferentes, realizamos um experimento treinando os classificadores com a base de dados da Leila Fletcher para piano e testamos com a base de dados Suzuki para violino. Comumente os testes e treinos são feitos com bases de dados parecidas, ou seja, o treino é realizado com uma base de dados para violino e o teste também é feito com uma base de dados para violino. Por não termos disponível duas bases de dados com o mesmo instrumento, o experimento foi realizado com duas bases de instrumentos diferentes. Com isso também é possível comprovar o que foi discutido na Seção 5.3.

	MIDI	Imagem digital	MIDI + Imagem digital
J48	52%	33%	**61%**
NB	36%	44%	46%
RF	**61%**	38%	43%
SMO	41%	**48%**	44%
KNN	**61%**	37%	43%

Tabela 13 – Treino com piano e teste com violino, para o formato MIDI, imagem digital e a fusão entre o MIDI e imagem digital
Fonte: Autoria própria

Como podemos observar na Tabela 13, somente algumas abordagens foram escolhidas para esse experimento final, o formato MIDI, a imagem digital e a fusão do formato MIDI com a imagem digital. O treino foi realizado utilizando 258 partituras musicais e o teste foi realizado com 99 partituras musicais.

É possível observar que taxa de acerto dos diferentes classificadores e abordagens propostas nesse experimento, ficam abaixo dos experimentos realizados anteriormente, onde a melhor taxa de acerto é de 61%.

Olhando para a Tabela 14, notamos que apenas três classificadores obtêm os melhores resultados, C4.5, Random Forest e o SVM. É possível observar que na maioria dos experimentos há sempre um resultado com a taxa de acerto baixa, na coluna do formato MIDI com o classificador Random Forest chega a 0% de acerto.

Para cada abordagem o padrão dos resultados para os níveis de dificuldade, muda totalmente. Se analisarmos o formato MIDI, temos o melhores resultados no nível médio, depois no nível fácil e por último no nível difícil. Para a abordagem da imagem digital, o melhor resultado é no nível difícil, depois no nível fácil e por último o nível médio. E na última abordagem desse experimento, a fusão do formato MIDI com o a imagem digital, o

	MIDI			Imagem digital			MIDI + Img. Digital		
Classificadores	Fácil	Médio	Difícil	Fácil	Médio	Difícil	Fácil	Médio	Difícil
C4.5	61%	15%	**84%**	15%	**48%**	44%	**66%**	58%	56%
NB	7%	64%	48%	46%	6%	92%	44%	21%	84%
RF	**90%**	70%	0%	41%	18%	60%	44%	9%	**91%**
SVM	22%	**91%**	12%	**51%**	12%	**92%**	37%	**76%**	16%
KNN	88%	58%	16%	34%	9%	80%	49%	39%	40%
Média	54%	60%	32%	37%	19%	74%	48%	41%	57%

Tabela 14 – Resultado geral da acurácia das classificações com o teste e treino invertido
Fonte: Autoria própria

melhor resultado se encontra no nível difícil, depois nível médio e por fim, o nível fácil.

Em comparação com experimentos já discutidos acima, onde em grande parte dos resultados um padrão se repetia, aqui não é possível identificar qual é o nível de dificuldade que os classificadores conseguem aprender melhor. Com isso é possível analisar que a interação entre bases musicais de diferentes instrumentos musicais, não melhora o resultado para a classificação automática de partitura por nível de dificuldade.

6 Conclusão

Ao analisarmos esse cenário, conseguiremos concluir qual é a melhor abordagem para a classificação automática de partituras musicais por nível de dificuldade.

Neste trabalho propusemos um método para a classificação automática de partituras musicais por níveis de dificuldade. O método é baseado no uso de materiais didáticos tradicionais de música como padrão. Na pesquisa, utilizamos o método Suzuki para o violino com 99 partituras, Henrique Pinto para violão com 56 partituras e Leila Fletcher para piano com 258 partituras. Cada método tradicional de ensino possui uma quantidade diferente de livros musicais, que refletem nos níveis de dificuldade, o método Suzuki possui 10 livros, o método Henrique Pinto possui apenas 2 livros e por fim, o método Leila Fletcher possui 6 livros. Para padronizar o método proposto, condensamos os livros, para que tivéssemos apenas níveis 3 níveis, o fácil, o médio e o difícil. Cada partitura musical de cada volume foi convertida para arquivos musicXML (partitura digital), que por sua vez foram convertidos para arquivo musical (MIDI) e arquivo visual (JPG), é importante ressaltar, que o arquivo visual contém uma versão, a versão da imagem digital da partitura, reproduzido por software de notação musical, ou seja, é uma imagem limpa da partitura, a imagem original do livro, contendo a sombras da foto, níveis de envelhecimento diferentes nos livros e tantas outras características de uma foto tirada de um livro física, não foi utilizada nessa pesquisa.

A partir dos arquivos digitais, foram utilizadas técnicas para extrair características da música e da imagem. Para a imagem utilizamos o descritor LBP para extrair as características e no arquivo de áudio, foi utilizada a biblioteca jSymbolic. Nossos experimentos foram realizados utilizando seis classificadores, que são: C4.5, Redes Neurais, Naive Bayes, Random Forest, SVM e KNN.

Com base nos resultados das colunas ”MIDI” e ”MIDI + Img. DIgital” da Tabela 12, que são as abordagens com melhores taxas de acerto, é possível analisar que as melhores acurácias acontecem sempre na abordagem do arquivo MIDI, exceto quando unimos todas as bases, tendo a melhor acurária na abordagem da fusão. Para todas as bases de instrumentos únicos o classicador que obtém o melhor resultado é a Rede Neural, obtendo 85% de acurácia para violino, 83% para piano, 84% para violão. Por fim, o melhor resultado nessa pesquisa, acontece quando todas as bases são unificadas em um banco de dados e utilizamos a fusão entre o MIDI e a imagem digital, obtendo 93% de taxa de acerto.

Por fim, essa pesquisa concluiu que:

- Os resultados obtidos mostraram que esta é uma tarefa desafiadora, uma vez que o

melhor resultado obtido na fase inicial da pesquisa foi de 85% com o classificador Rede Neural na base Suzuki para violino, com as características do arquivo MIDI. Posteriormente, novas bases foram integradas a pesquisa junto com novas formas de extrair as características dos arquivos. Com isso, a pesquisa demonstrou que classificar automaticamente partituras musicais por nível de dificuldade é possível e pode ser realizada com êxito, tendo em vista que obteve-se resultados promissores em todas as bases de dados.

- A abordagem da fusão entre do formato MIDI foi a que teve melhor taxa de acerto na maioria das bases musicais, mas essa abordagem só é possível se a transcrição da partitura física para o formato digital for realizado por um especialista, tendo em conta que isso leva, tempo e conhecimento músical.

- A junção de bases com instrumentos musicais diferentes, produz resultados inferiores aos resultados das bases individuais, ou seja, na produção de uma plataforma para a classificação automática de partituras musicais, os modelos para classificação seriam individuais para cada instrumento.

A principal limitação dessa pesquisa está justamente na criação das bases de dados, pois partimos do princípio que para a classificação automática de partituras, precisamos de bases rotuladas, utilizando como exemplo métodos de ensino que já estão consolidados no meio musical. Para que novas pesquisas fossem executadas com instrumentos musicais diferentes, seria necessário novos livros de ensino musical, e para cada um deles uma transcrição. A transcrição como já citamos nesse trabalho, leva tempo, dinheiro e conhecimento musical.

Por fim, as próximas etapas dessa pesquisa, seria em cima de novas bases com instrumentos musicais diferentes, pois seria possível comprovar se o método proposto nessa pesquisa se aplica em outros cenários. A obtenção de bases com instrumentos musicais já utilizados na pesquisa, também ajudaria a comprovar todos os experimentos realizados até então.

Outro passo importante, seria a criação uma plataforma que classifica e organiza partituras musicais automaticamente por nível de dificuldade e instrumento.

Referências

AHONEN, Timo; HADID, Abdenour; PIETIKAINEN, Matti. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, IEEE, n. 12, p. 2037–2041, 2006. Citado na página 52.

ANDERTON, Craig. *Midi reloaded. Keyboard Magazine*. [S.l.: s.n.], 2014. Citado na página 21.

AQUINO, JÔNATAS HOLANDA NOGUEIRA DE. ExtraÇÃo de caracterÍsticas de imagens para classificaÇÃo da qualidade de couro caprino usando padrÃo binÁrio local. 2017. Citado 2 vezes nas páginas 36 e 38.

BISHOP, Christopher M et al. *Neural networks for pattern recognition*. [S.l.]: Oxford university press, 1995. Citado na página 29.

BORGES, Luiz Eduardo. *Python para desenvolvedores: aborda Python 3.3*. [S.l.]: Novatec Editora, 2014. Citado na página 41.

BRODATZ, Phil. *Textures: a photographic album for artists and designers*. [S.l.]: Dover Pubns, 1966. Citado na página 37.

CAWLEY, Gavin C; TALBOT, Nicola LC. Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers. *Pattern Recognition*, Elsevier, v. 36, n. 11, p. 2585–2592, 2003. Citado na página 54.

CHIU, S.; CHEN, M. A study on difficulty level recognition of piano sheet music. In: *2012 IEEE International Symposium on Multimedia*. [S.l.: s.n.], 2012. p. 17–23. Citado 2 vezes nas páginas 43 e 44.

COVER, Thomas; HART, Peter. Nearest neighbor pattern classification. *IEEE transactions on information theory*, IEEE, v. 13, n. 1, p. 21–27, 1967. Citado na página 28.

ČRNČEC, Rudi; WILSON, Sarah J; PRIOR, Margot. The cognitive and academic benefits of music to children: Facts and fiction. *Educational Psychology*, Taylor & Francis, v. 26, n. 4, p. 579–594, 2006. Citado na página 21.

DUDA, Richard O; HART, Peter E; STORK, David G et al. Pattern classification. *International Journal of Computational Intelligence and Applications*, IMPERIAL COLLEGE PRESS, v. 1, p. 335–339, 2001. Citado na página 31.

EBERSBACH, Mike; HERMS, Robert; EIBL, Maximilian. Fusion methods for icd10 code classification of death certificates in multilingual corpora. In: *CLEF (Working Notes)*. [S.l.: s.n.], 2017. Citado na página 36.

ENGLUND, Cristofer; VERIKAS, Antanas. A novel approach to estimate proximity in a random forest: An exploratory study. *Expert systems with applications*, Elsevier, v. 39, n. 17, p. 13046–13050, 2012. Citado na página 35.

FLETCHER, Leila. *The Leila Fletcher Piano Course - Book Five*. [S.l.]: Montgomery Publishing, 1993. Citado na página 47.

FLETCHER, Leila. *The Leila Fletcher Piano Course - Book Four*. [S.l.]: Montgomery Publishing, 1993. Citado na página 47.

FLETCHER, Leila. *The Leila Fletcher Piano Course - Book One*. [S.l.]: Montgomery Publishing, 1993. Citado 3 vezes nas páginas 21, 26 e 47.

FLETCHER, Leila. *The Leila Fletcher Piano Course - Book Six*. [S.l.]: Montgomery Publishing, 1993. Citado na página 49.

FLETCHER, Leila. *The Leila Fletcher Piano Course - Book Three*. [S.l.]: Montgomery Publishing, 1993. Citado na página 47.

FLETCHER, Leila. *The Leila Fletcher Piano Course - Book Two*. [S.l.]: Montgomery Publishing, 1993. Citado na página 47.

GARNER, Stephen R et al. Weka: The waikato environment for knowledge analysis. In: *Proceedings of the New Zealand computer science research students conference*. [S.l.: s.n.], 1995. p. 57–64. Citado na página 40.

GOOD, Michael et al. Musicxml: An internet-friendly format for sheet music. In: *XML Conference and Expo*. [S.l.: s.n.], 2001. p. 3–4. Citado 2 vezes nas páginas 21 e 38.

GOUYON, F.; HERRERA, Perfecto; MARTINS, L. G.; MÜLLER, M. *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*. Porto: FEUP Edições, 2012. ISBN 978-972-752-144-9. Disponível em: <http://ismir2012.ismir.net/event/978-972-752-144-9.pdf>. Citado na página 22.

GROUP, The REACT; CHYLACK, Leo T; BROWN, Nicholas P; BRON, Anthony; HURST, Mark; KÖPCKE, Wolfgang; THIEN, Uta; SCHALCH, Wolfgang. The roche european american cataract trial (react): a randomized clinical trial to investigate the efficacy of an oral antioxidant micronutrient mixture to slow progression of age-related cataract. *Ophthalmic epidemiology*, Taylor & Francis, v. 9, n. 1, p. 49–80, 2002. Citado na página 21.

GUÉRIN, Robert. *MIDI Power! The Comprehensive Guide*. [S.l.]: Alfred Music, 2008. (2 edição). ISBN 978-1598630848. Citado 2 vezes nas páginas 21 e 39.

HALL, Mark; FRANK, Eibe; HOLMES, Geoffrey; PFAHRINGER, Bernhard; REUTEMANN, Peter; WITTEN, Ian H. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, ACM, v. 11, n. 1, p. 10–18, 2009. Citado na página 41.

JR, Alceu S Britto; SABOURIN, Robert; OLIVEIRA, Luiz ES. Dynamic selection of classifiers—a comprehensive review. *Pattern Recognition*, Elsevier, v. 47, n. 11, p. 3665–3680, 2014. Citado na página 34.

LIN, Chih-Chun; LIU, Damon Shing-Min. An intelligent virtual piano tutor. In: *Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications*. New York, NY, USA: ACM, 2006. (VRCIA '06), p. 353–356. ISBN 1-59593-324-7. Disponível em: <http://doi.acm.org/10.1145/1128923.1128986>. Citado na página 21.

LORENA, Ana Carolina; CARVALHO, André CPLF de. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007. Citado na página 32.

MAKEMUSIC. *Finale: The World Standard in Music Notation Software*. <http://www.finalemusic.com/>. Accessed: 2019-01-15. Citado na página 38.

MCKAY, Cory. *jSymbolic Feature Explanation*. 2018. <http://jmir.sourceforge.net/manuals/jSymbolic_manual/home.html>. Accessed: 2020-02-19. Citado na página 52.

MCKAY, Cory; FUJINAGA, Ichiro. jsymbolic: A feature extractor for midi files. In: *ICMC*. [S.l.: s.n.], 2006. Citado 2 vezes nas páginas 40 e 52.

MOOG, Robert A. Midi: musical instrument digital interface. *Journal of the Audio Engineering Society*, Audio Engineering Society, v. 34, n. 5, p. 394–404, 1986. Citado na página 39.

PINTO, Henrique. *Iniciação ao Violão - Volume I*. [S.l.]: Editora Ricordi, 2008. Citado 3 vezes nas páginas 21, 26 e 49.

PINTO, Henrique. *Iniciação ao Violão - Volume II*. [S.l.]: Editora Ricordi, 2008. Citado na página 49.

SANTOS, FERNANDO CHAGAS et al. *Variações do método kNN e suas aplicações na classificação automática de textos*. Tese (Doutorado) — Dissertação de Mestrado, Programa de Pós-Graduação do Instituto de . . . , 2009. Citado na página 28.

SÉBASTIEN, Véronique; RALAMBONDRAINY, Henri; SÉBASTIEN, Olivier; CONRUYT, Noël. Score analyzer: automatically determining scores difficulty level for instrumental e-learning. In: *13th International Society for Music Information Retrieval Conference (ISMIR 2012)*. [S.l.: s.n.], 2012. p. 571–576. Citado 2 vezes nas páginas 43 e 44.

SHINN, Maxwell. *Instant MuseScore*. [S.l.]: Packt Publishing Ltd, 2013. Citado na página 38.

Suzuki Americas, International Association of the Americas. *About the Suzuki Method*. <https://suzukiassociation.org/about/suzuki-method/>. Accessed: 2019-01-15. Citado na página 25.

Suzuki International, Association. *The Suzuki Method*. 2018. <http://internationalsuzuki.org/method.htm>. Accessed: 2018-09-27. Citado 2 vezes nas páginas 21 e 25.

SUZUKI, Shinichi. *Suzuki Violin School-Volume 1: Violin Part*. [S.l.]: Alfred Music, 2015. Citado 4 vezes nas páginas 21, 22, 23 e 47.

SUZUKI, Shinichi. *Suzuki Violin School-Volume 10: Violin Part*. [S.l.]: Alfred Music, 2015. Citado na página 47.

SUZUKI, Shinichi. *Suzuki Violin School-Volume 2: Violin Part*. [S.l.]: Alfred Music, 2015. Citado 2 vezes nas páginas 47 e 51.

SUZUKI, Shinichi. *Suzuki Violin School-Volume 3: Violin Part*. [S.l.]: Alfred Music, 2015. Citado na página 47.

SUZUKI, Shinichi. *Suzuki Violin School-Volume 4: Violin Part*. [S.l.]: Alfred Music, 2015. Citado na página 47.

SUZUKI, Shinichi. *Suzuki Violin School-Volume 5: Violin Part*. [S.l.]: Alfred Music, 2015. Citado na página 47.

SUZUKI, Shinichi. *Suzuki Violin School-Volume 6: Violin Part*. [S.l.]: Alfred Music, 2015. Citado na página 47.

SUZUKI, Shinichi. *Suzuki Violin School-Volume 7: Violin Part*. [S.l.]: Alfred Music, 2015. Citado na página 47.

SUZUKI, Shinichi. *Suzuki Violin School-Volume 8: Violin Part*. [S.l.]: Alfred Music, 2015. Citado na página 47.

SUZUKI, Shinichi. *Suzuki Violin School-Volume 9: Violin Part*. [S.l.]: Alfred Music, 2015. Citado na página 47.

WITTEN, Ian H. *Data Mining: Practical machine learning tools and techniques*. [S.l.: s.n.], 2005. ISBN 978-0-12-804291-5. Citado na página 26.

Apêndices

APÊNDICE A – Matrizes de confusão do método Suzuki

Fácil	Médio	Díficil	
38	3	0	Fácil
3	**27**	3	Médio
1	5	**19**	Díficil

Tabela 15 – Matriz de confusão do classificador C4.5 para abordagem em formato MIDI
Fonte: Autoria própria

Fácil	Médio	Díficil	
35	66	0	Fácil
9	**20**	4	Médio
1	6	**18**	Díficil

Tabela 16 – Matriz de confusão do classificador C4.5 para abordagem da imagem digital
Fonte: Autoria própria

Fácil	Médio	Díficil	
36	4	1	Fácil
3	**27**	3	Médio
1	5	**19**	Díficil

Tabela 17 – Matriz de confusão do classificador C4.5 para abordagem da fusão do formato MIDI com a imagem digital
Fonte: Autoria própria

Fácil	Médio	Díficil	
31	10	0	Fácil
3	**19**	14	Médio
0	2	**23**	Díficil

Tabela 18 – Matriz de confusão do classificador Naive Bayes para abordagem em formato MIDI
Fonte: Autoria própria

Fácil	Médio	Díficil	
28	11	2	Fácil
3	**17**	13	Médio
2	10	**13**	Díficil

Tabela 19 – Matriz de confusão do classificador Naive Bayes para abordagem da imagem digital

Fonte: Autoria própria

Fácil	Médio	Díficil	
31	8	2	Fácil
1	**19**	13	Médio
1	3	**21**	Díficil

Tabela 20 – Matriz de confusão do classificador Naive Bayes para abordagem da fusão do formato MIDI com a imagem digital

Fonte: Autoria própria

Fácil	Médio	Díficil	
35	6	0	Fácil
3	**30**	0	Médio
0	8	**17**	Díficil

Tabela 21 – Matriz de confusão do classificador Random Forest para abordagem em formato MIDI

Fonte: Autoria própria

Fácil	Médio	Díficil	
33	7	1	Fácil
6	**19**	8	Médio
2	9	**14**	Díficil

Tabela 22 – Matriz de confusão do classificador Random Forest para abordagem da imagem digital

Fonte: Autoria própria

Fácil	Médio	Díficil	
33	8	0	Fácil
6	**24**	3	Médio
1	3	**21**	Díficil

Tabela 23 – Matriz de confusão do classificador Random Forest para abordagem da fusão do formato MIDI com a imagem digital

Fonte: Autoria própria

Fácil	Médio	Díficil	
36	5	0	Fácil
5	**26**	2	Médio
0	5	**20**	Díficil

Tabela 24 – Matriz de confusão do classificador SVM para abordagem em formato MIDI

Fonte: Autoria própria

Fácil	Médio	Díficil	
32	8	1	Fácil
9	**21**	3	Médio
1	7	**17**	Díficil

Tabela 25 – Matriz de confusão do classificador SVM para abordagem da imagem digital
Fonte: Autoria própria

Fácil	Médio	Díficil	
35	5	1	Fácil
5	**27**	1	Médio
2	5	**18**	Díficil

Tabela 26 – Matriz de confusão do classificador SVM para abordagem da fusão do formato MIDI com a imagem digital
Fonte: Autoria própria

Fácil	Médio	Díficil	
39	2	0	Fácil
6	**23**	4	Médio
2	9	**14**	Díficil

Tabela 27 – Matriz de confusão do classificador KNN para abordagem em formato MIDI
Fonte: Autoria própria

Fácil	Médio	Díficil	
33	6	2	Fácil
6	**21**	6	Médio
5	7	**14**	Díficil

Tabela 28 – Matriz de confusão do classificador KNN para abordagem da imagem digital
Fonte: Autoria própria

Fácil	Médio	Díficil	
38	3	0	Fácil
3	**27**	3	Médio
1	5	**19**	Díficil

Tabela 29 – Matriz de confusão do classificador KNN para abordagem da fusão do formato MIDI com a imagem digital
Fonte: Autoria própria

Fácil	Médio	Díficil	
35	66	0	Fácil
9	**20**	4	Médio
1	6	**18**	Díficil

Tabela 30 – Matriz de confusão do classificador Rede Neural para abordagem em formato MIDI
Fonte: Autoria própria

Fácil	Médio	Díficil	
35	6	0	Fácil
4	**27**	2	Médio
0	4	**21**	Díficil

Tabela 31 – Matriz de confusão do classificador Rede Neural para abordagem da imagem digital

Fonte: Autoria própria

Fácil	Médio	Díficil	
36	4	1	Fácil
3	**27**	3	Médio
1	5	**19**	Díficil

Tabela 32 – Matriz de confusão do classificador Rede Neural para abordagem da fusão do formato MIDI com a imagem digital

Fonte: Autoria própria