UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PAULA LUANA OLIVEIRA MIRANDA

CLASSIFICAÇÃO DE ARRITMIA CARDÍACA COM APRENDIZADO DE MÁQUINA AUTOMATIZADO (AUTOML)

QUIXADÁ

2023

PAULA LUANA OLIVEIRA MIRANDA

CLASSIFICAÇÃO DE ARRITMIA CARDÍACA COM APRENDIZADO DE MÁQUINA AUTOMATIZADO (AUTOML)

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Regis Pires Magalhães

QUIXADÁ

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M645c Miranda, Paula Luana Oliveira.
Classificação de arritmia cardíaca com aprendizado de máquina automatizado (AutoML) / Paula Luana Oliveira Miranda. – 2023.
67 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2023.
Orientação: Prof. Dr. Regis Pires Magalhães.

1. Eletrocardiograma. 2. Aprendizado de máquina automatizado. 3. Técnica de reamostragem. 4. AutoGluon. I. Título.

CDD 004

PAULA LUANA OLIVEIRA MIRANDA

CLASSIFICAÇÃO DE ARRITMIA CARDÍACA COM APRENDIZADO DE MÁQUINA AUTOMATIZADO (AUTOML)

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em: 12/12/2023

BANCA EXAMINADORA

Prof. Dr. Regis Pires Magalhães (Orientador)
Universidade Federal do Ceará (UFC)

Profa. Dra. Lívia Almada Cruz
Universidade Federal do Ceará - UFC

Prof. Dr. Paulo de Tarso Guerra Oliveira
Universidade Federal do Ceará - UFC

À Deus por me permitir chegar até aqui e estar comigo em tudo. À minha família por serem meus maiores incentivadores na jornada que é a vida.

AGRADECIMENTOS

Quando penso em agradecer, a primeira coisa que vem a minha mente é agradecer por conseguir escrever esse trabalho. Esse trabalho representa que venci os sintomas da endometriose que me paralisou por tanto tempo. Esse trabalho significa que encontrei forças onde achei que não tinha e consegui me reerguer. Nessa etapa, nem consigo lembrar dos sofrimentos da graduação, só lembro que quase perdi meu diploma por uma doença que trancou minha graduação, me fez sair de estágio e me fez chorar por 1 ano. Precisei de 1 ano para me reerguer, 1 ano para voltar a me sentir uma jovem de 24 anos. Fico feliz por escrever. Fico feliz por superar.

Agradeço à Deus, por ser meu sustento e nunca deixar eu me sentir só. Agradeço ao meu marido, Paulo Miranda, por estar comigo entre os altos e baixos da vida, por fazer parecer tudo ser possível e por tornar minha graduação mais leve. A minha mãe, agradeço por acreditar imensamente nos meus sonhos desde quando eu estava tentando aprender a falar "liquidificador"; sem ela eu não seria nem metade do que sou hoje. Agradeço ao meu pai por persistir que eu sempre continuasse estudando e buscando melhorar. Ao meu irmão, agradeço por ser amor, aconchego e luz na minha vida. Aos meus avós paternos, Jaci e Orlando, agradeço por me darem o mundo e sempre fazerem eu acreditar que sou a garota mais especial que existe. Às minhas tias, Renata e Roberta: obrigada por me fazerem me sentir sortuda por ter vocês, seu amor e carinho são essenciais na minha vida.

Dentro da Universidade, agradeço ao PACCE por me acolher desde que ingressei, graças ao projeto pude me desenvolver mais. Aos professores, agradeço ao Paulo de Tarso por ter me ajudado sempre que precisei, a professora Lívia Almada por ser encorajamento e acolhida, ao meu grande orientador Régis Pires, agradeço por tanto aprendizado e compreensão. À Universidade Federal do Ceará (UFC) - campus Quixadá, por ser minha casa por todos esses anos, não imagino como teria sido a minha vida sem ter estudado nesse lugar incrível.

"Daqui a 20 anos você estará mais decepcionado pelas coisas que você não fez, do que pelas que fez. Então, jogue fora suas amarras, navegue para longe do porto seguro, pegue os ventos em suas velas. Explore, sonhe, descubra." (Mark Twain)

RESUMO

As doenças cardiovasculares são as que mais matam no mundo, sendo a arritmia cardíaca uma das maiores causadoras. O exame capaz de detectar a arritmia cardíaca é o eletrocardiograma, um exame que contém ruídos e é de difícil interpretação. Em vista disso, a fim de contribuir com pesquisas que facilitem o processo de interpretação de exames, a Inteligência artificial possui uma vasta quantidade de trabalhos relacionados a esse objetivo, onde os pesquisadores têm desenvolvido estratégias que tornam os dados mais legíveis para algoritmos e analisado diferentes estruturas de modelos que lidem melhor com dados de eletrocardiograma. Obtendo o mesmo objetivo, o presente trabalho implementa e analisa: tamanhos de segmentações sobre os dados do exame Eletrocardiograma (ECG); aplicação de técnica de remoção de ruído da biblioteca *Neurokit*; Técnica de Sobreamostragem Sintética para Classes Minoritárias (SMOTE) e técnica Amostragem Sintética Adaptativa para Classes Minoritárias (ADASYN); e utiliza aprendizado de máquina automatizado tendo em vista a busca por melhores hiperparâmetros que se adéquem a algoritmos sem que seja estabelecido testes manualmente. Ademais, o presente trabalho faz um levantamento sobre as variações de hiperparâmetros gerados para os modelos pela ferramenta de aprendizado de máquina automatizado, AutoGluon. Com as abordagens implementadas esse trabalho conclui que para a maioria dos algoritmos utilizados o uso de dados com segmentação 360 permite que o modelo alcance um maior desempenho quando comparado a dados segmentados em 2000. Modelos de aprendizado de máquina alcançaram melhores desempenhos quando classificaram dados que tiveram a técnica de remoção de ruído aplicado. E por fim, o *Autogluon* gerou mais variações para modelos de aprendizado profundo, mas apesar disso o algoritmo de aprendizado de máquina *Light Gradient Boosting Machine* (LightGBM) também obteve um alto desempenho na fase de teste. Sendo assim, os algoritmos que alcançaram melhores desempenhos foram os redes neurais e o LightGBM.

Palavras-chave: eletrocardiograma; aprendizado de máquina automatizado; técnica de reamostragem; AutoGluon.

ABSTRACT

Cardiovascular diseases are the biggest killers in the world, with cardiac arrhythmia being one of the biggest causes. The test capable of detecting cardiac arrhythmia is the electrocardiogram, a test that contains noise and is difficult to interpret. In view of this, in order to contribute to research that facilitates the process of interpreting exams, Artificial Intelligence has a vast amount of work related to this objective, where researchers have developed strategies that make data more readable for algorithms and analyzed different model structures that better handle electrocardiogram data. Achieving the same objective, the present work implements and analyzes: segmentation sizes on Electrocardiogram (ECG) exam data; application of noise removal technique from the Neurokit library; Synthetic Oversampling Technique for Minority Classes (SMOTE) and Adaptive Synthetic Sampling Technique for Minority Classes (ADASYN); and uses automated machine learning to search for better hyperparameters that fit algorithms without manually establishing tests. Furthermore, the present work surveys the variations in hyperparameters generated for the models by the automated machine learning tool, AutoGluon. With the approaches implemented, this work concludes that for most of the algorithms used, the use of data with 360 segmentation allows the model to achieve greater performance when compared to data segmented in 2000. Machine learning models achieved better performances when classifying data that had the noise removal technique applied. Finally, Autogluon generated more variations for deep learning models, but despite this, the Light Gradient Boosting Machine (LightGBM) machine learning algorithm also achieved high performance in the testing phase. Therefore, the algorithms that achieved the best performance were neural networks and LightGBM.

Keywords: electrocardiogram; automated machine learning; resampling technique; AutoGluon.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AC	Arritmia Cardíaca
ACI	Análise de Componentes Independentes
AD	Árvore de Decisão
ADASYN	Amostragem Sintética Adaptativa para Classes Minoritárias
AM	Aprendizado de Máquina
AP	Aprendizado Profundo
AutoML	Aprendizagem de Máquina Automatizado
CNN	Rede Neural Convolucional
DCV	Doenças cardiovasculares
ECG	Eletrocardiograma
Extra Trees	*Extremely Randomized Trees*
GB	*Gradient boosting*
GRU	*Gated Recurrent Unit*
KNN	*K-nearest neighbors*
LightGBM	*Light Gradient Boosting Machine*
LSTM	Rede de Memória de Longo Prazo e Curto Prazo
MIT-BIH	*Massashusetts Institute of Technology - Boston Beth Israel Hospital*
MSE	Erro Médio Quadrático
PTB	*Physikalisch-Technische Bundesanstalt*
RF	Floresta Aleatória
RNA	Redes Neurais Artificiais
RNN	Redes Neurais Recorrentes
SMOTE	Técnica de Sobreamostragem Sintética para Classes Minoritárias
SVM	Máquina de Vetores de Suporte
XGB	*Extreme Gradient Boosting*

SUMÁRIO

1 INTRODUÇÃO

Segundo a Organização Mundial da Saúde (2021), as Doenças cardiovasculares (DCV) são as que mais matam no Brasil e no mundo. Apenas em 2016 cerca de 17,9 milhões de pessoas vieram a óbito por DCV, sendo 31% das causas de mortes em nível global. Entre esses, 85% foram de ataques cardíacos e acidentes vasculares cerebrais. Somente no Brasil houveram em torno de 230 mil mortes por DCV em 2021 (Sociedade Brasileira de Cardiologia, 2021).

O Brasil possui aproximadamente 14 milhões de pessoas que possuem DCV, o que tornou-se ainda mais preocupante no cenário pandêmico causado pelo vírus da covid-19, já que o vírus é capaz de agravar DCV (Sociedade Brasileira de Cardiologia, 2021). Como consequência desse cenário, Brant *et al.* (2020) constou em seus estudos um aumento de 132% de mortes por DCV no Brasil durante a pandemia em 2020.

Apesar dos dados alarmantes, segundo Melo (2022) a maioria das doenças que afetam o coração podem ser prevenidas através de mudanças de hábitos, como manter uma dieta saudável, praticar atividades físicas, evitar o uso de drogas e realizar exames periódicos para possibilitar o diagnóstico precoce. Sendo assim, é evidente que os cuidados com o coração são fundamentais, ademais, o coração é um órgão vital, seu bom funcionamento permite que a circulação sanguínea transporte oxigênio para as células e para todos os sistemas do corpo humano, fazendo com que o corpo tenha um bom funcionamento.

Vale mencionar que algumas das DCV são mais comuns no Brasil, como a insuficiência cardíaca, pressão alta, miocardite e entre elas a Arritmia Cardíaca (AC). A AC é a doença que pode caracterizar a condição do infarto do miocárdio, conhecido também como ataque cardíaco, essa condição foi indicada por Stevens *et al.* (2018) como a que contém o maior custo financeiro e segundo Feldman e Goldwasser (2004) o processo de analisar o exame responsável por detectar a condição é de difícil compreensão. Tendo em vista esse contexto, o presente trabalho contribui com estratégias de pré-processamento sobre dados de ECG e modelos com hiperparâmetros configurados capazes de classificar a AC.

O ECG possibilita a constatação do infarto do miocárdio, conseguindo detectar o ritmo do coração através do uso de eletrodos fixados na pele. O ECG facilita a avaliação da atividade elétrica do coração, nele são apresentados dados como ondas, intervalos e segmentos. Apesar de o exame conter fácil manuseio, a análise dedutiva é a base fundamental na interpretação do ECG e requer o conhecimento do significado do processo de ativação do coração (FELDMAN; GOLDWASSER, 2004).

A fim de contribuir com o processo de classificação das DCV e colaborar com a prevenção, a Inteligência Artificial possui uma vasta quantidade de trabalhos relacionados ao tema (AL'AREF *et al.*, 2019). Os trabalhos englobam Aprendizado de Máquina (AM) e Aprendizado Profundo (AP), esses em sua maioria tem como objetivos analisar técnicas de pré-processamento dos dados de ECG, implementar modelos que obtenham um alto percentual de acertos na classificação, analisar e aperfeiçoar o desempenho dos algoritmos (MARINHO, 2021).

Esses objetivos não são fáceis de alcançar, considerando que o ECG pode obter dados de sinais irregulares que acontecem por falhas durante a captura. Outro fator considerado é o volume de dados e sua representação.

Sendo assim, além desse trabalho almejar contribuir com a prevenção de DCV a partir da classificação da AC com os dados do ECG, ele propõe-se a colaborar com a literatura por meio do contraste feito com trabalhos referentes a temática e fornecendo informações de análises de técnicas que podem possibilitar melhores classificações com AM e AP.

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho tem como intuito realizar a classificação da arritmia cardíaca (AC) propondo-se a comparar métodos e técnicas utilizados no processo de classificação de DCV.

1.1.2 Objetivos específicos

- Analisar diferentes segmentações dos dados do ECG a fim de identificar um melhor uso dos dados para a tarefa de classificação.
- Testar diferentes técnicas de reamostragem dos dados.
- Gerar modelos de AM e AP através do Aprendizagem de Máquina Automatizado (AutoML).
- Avaliar os modelos fazendo um comparativo com as segmentações e pré-processamentos.

1.2 Organização

A estruturação do trabalho nos capítulos posteriores é realizada da seguinte maneira: o Capítulo 2 detalha os conceitos que fundamentam o trabalho; o Capítulo 3 descreve resumidamente o que os trabalhos relacionados destacam, esclarece as diferenças com o presente trabalho e apresenta um quadro comparativo de técnicas utilizadas por cada um; o Capítulo 4 apresenta o processo seguido para a construção deste trabalho e as decisões tomadas durantes esse percurso; já os resultados são retratados no Capítulo 5; por fim, no Capítulo 6 será apresentado as conclusões e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

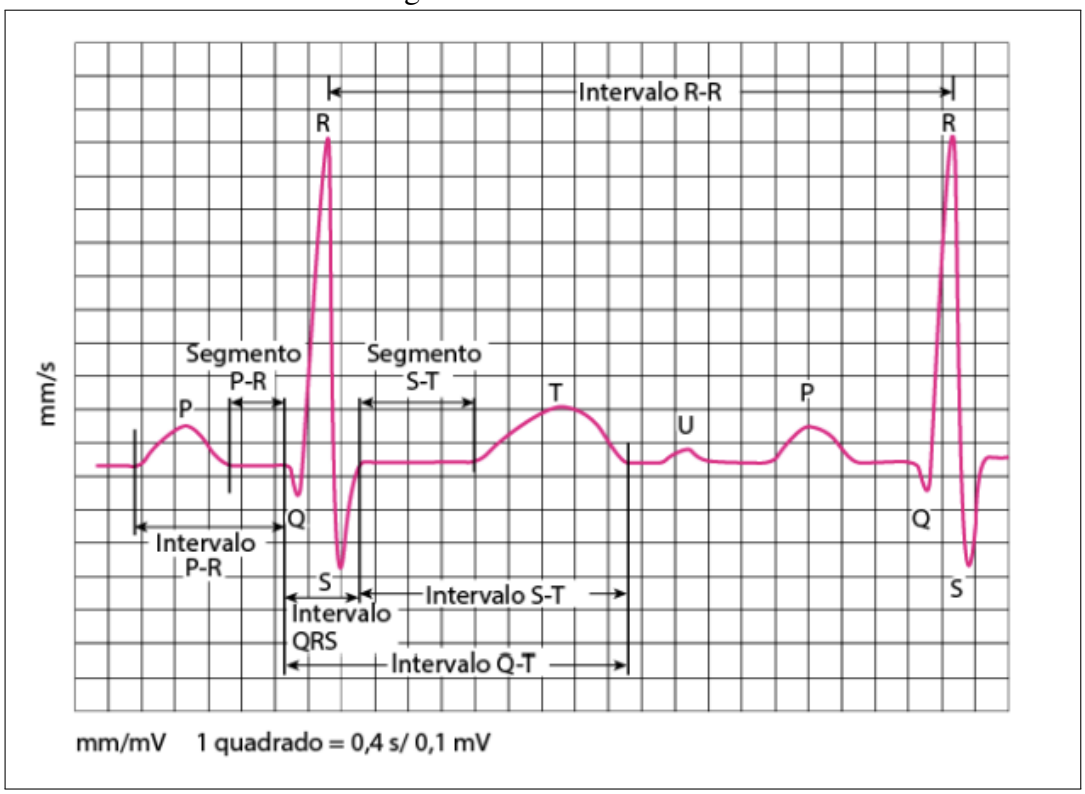
Este capítulo retrata os conceitos fundamentais do que está sendo desenvolvido neste trabalho. Sendo assim, são apresentados conceitos referentes ao eletrocardiograma (ECG), aprendizado de máquina (AM), aprendizado profundo (AP) e métricas avaliativas.

2.1 Eletrocardiograma

O coração é o órgão responsável por bombear o sangue e fazer o transporte de oxigênio pelo corpo. Esse órgão é constituído por quatro câmaras, que são: átrio direito, átrio esquerdo, ventrículo direito e ventrículo esquerdo. O lado direito é responsável por levar sangue para os pulmões, onde é feito a oxigenação do sangue. Já o lado esquerdo é responsável por bombear o sangue para o resto do corpo (SANTOS, 2023).

O eletrocardiograma consiste em impulsos elétricos do coração capturados por alguns eletrodos localizado nos membros e nas paredes torácicas (SANTOS, 2023). Na Figura 1 pode-se visualizar como é a forma de uma onda de um eletrocardiograma.

Figura 1 – Sinal de um eletrocardiograma



Fonte: Santos (2023)

Como é possível observar, o sinal tem vários intervalos, picos e vales. Sendo eles:

- Onda P: representa a contração do músculo e a ejeção de sangue para o ventrículo direito.
- Intervalo PR: trata-se de um retardo fisiológico do atrioventricular. Esse intervalo serve para evitar que o átrio seja contraído junto como o ventrículo.
- Complexo QRS: o complexo é a despolarização ventricular, no qual se tem uma deflexão negativa, seguida de uma positiva e finalizando com outra negativa. Tem uma duração normal entre 0,07 e 0,10 segundos.
- Segmento ST: significa o período de despolarização ventricular.
- Onda T: indica a repolarização dos ventrículos.
- Intervalo QT: representa o período entre a despolarização ventricular até a repolarização.
- Onda U: geralmente essa onda aparece em pessoas com algum problema como isquemia e hipomagnesemia.
- Intervalo RR: distância entre duas ondas consecutivas.

2.2 Aprendizado de máquina

O aprendizado de máquina (AM) faz a utilização da estatística e computação para extrair conhecimento dos dados. Algoritmos relacionados a essa área detectam padrões com relativa facilidade, seu propósito de utilização está em permitir que a análise de grandes volumes de dados não dependa frequentemente de intervenção humana. Essas análises permitem que dados sejam classificados e valores sejam previstos (DHAR, 2013).

O AM está presente no cotidiano através da recomendação de filmes, anúncios, detecção de fraudes em compras, etc. Fazer uma boa utilização dos dados tem sido crucial para o gerenciamento de negócios. Isso pode ser percebido na pesquisa da International Data Corporation (2021) que indica um gasto global de US$ 215,7 bilhões em 2021 com soluções relacionadas a área, o valor referido apresenta um aumento de 10,01% comparado aos gastos de 2020.

Os algoritmos utilizados no desenvolvimento das soluções podem ser caracterizadas com aprendizagens diferentes, como aprendizado não supervisionado, aprendizado por reforço e o aprendizado supervisionado. De acordo com Russell (2010), o aprendizado não supervisionado é quando o algoritmo pode aprender com dados de entrada, já no aprendizado por reforço, os algoritmos aprendem por atividades repetitivas resultando em recompensas ou punições, enquanto no aprendizado supervisionado a aprendizagem acontece com dados de entrada e saída.

Visto as definições de aprendizagem apresentadas, este trabalho fará o uso de téc-

nicas relacionadas a aprendizagem supervisionada, considerando pretenso o uso de dados que contenham entrada e saída a serem observados. Sabendo que, para analisar os padrões de arritmia cardíaca (AC) do eletrocardiograma (ECG), os sinais capturados pelo exame são os dados de entrada e o diagnóstico é o dado de saída.

2.3 Pré-processamento dos dados

Dados sempre podem ter inconsistências, valores nulos, valores discrepantes, informações que não contribuem, etc. Passar dados inconsistentes diretamente para um algoritmo podem ter como consequências erros durante a execução ou classificações erradas. Realizar o pré-processamento de dados consiste em aplicar técnicas de redução que podem ocasionar no melhor aproveitamento do algoritmo sobre os dados (GARCÍA *et al.*, 2015). Essa seção explica sobre as técnicas de pré-processamento utilizadas nesse trabalho.

2.3.1 Padronização de dados com escore Z

Como mencionado em 2.3, os dados na maioria das vezes não podem ser imediatamente inseridos em um algoritmo. Em dados de ECG por exemplo, a amplitude dos valores dos sinais pode variar para cada marca de equipamento. Lidar com dados muito discrepantes um do outro pode fazer com que um algoritmo defina pesos diferentes para atributos e isso influencie negativamente no processo de classificação (KRANZUSCH *et al.*, 2020).

O escore Z mantém os dados em uma escala entre -1 e 1 e retrata a relação de um dado sobre a média de dados. Na equação 2.1 é apresentado o cálculo do Z escore, onde Xi é o dado atual, μ é a média dos dados e o θ é o desvio padrão. A média indica para onde a maioria dos dados tende e já o desvio padrão representa a distância do valor de Xi em relação à média. Se o valor de Z tender a 0, indica que o Xi tem um valor próximo à média, se tender a 1 então Xi tem um desvio padrão acima da média e para -1 é quando o desvio padrão é abaixo da média (ABDI, 2007).

$$Z = \frac{Xi - \mu}{\theta} = \frac{Xi - \mu}{\frac{\sum (Xi-\mu)^2}{n}} \tag{2.1}$$

Quadro 1 – Distribuição das classes

Dados	Classe: 0	Classe: 1	Classe: 2	Classe: 3	Classe: 4
Sem técnica de reamostragem	56258	6053	5441	1910	5347
Técnica de reamostragem SMOTE	56258	56258	56258	56258	56258
Técnica de reamostragem ADASYN	56258	6053	5441	56201	5347

2.3.2 Reamostragem de dados

Segundo García *et al.* (2015), a reamostragem é uma técnica essencial em casos que o conjunto de dados possui grandes discrepâncias sobre a quantidade de dados por classe. Nesse cenário, classes minoritárias são dados atípicos e quando modelos recebem esses dados como entrada podem ter dificuldades de classificar corretamente.

O SMOTE e a de ADASYN são técnicas abordadas nesse trabalho, pois ambas lidam com as classes minoritárias. O SMOTE gera dados sintéticos para as classes minoritárias, onde para cada classe ele escolhe aleatoriamente vizinhos mais próximos. Para cada vizinho selecionado, novos dados sintéticos são gerados alternando características entre os dados sintéticos e os vizinhos selecionados (CHAWLA *et al.*, 2002).

Já o ADASYN ao invés de selecionar aleatoriamente os vizinhos como o SMOTE, ele considera a densidade local de dados para ajustar a quantidade de dados sintéticos. Mais especificamente, a quantidade de dados sintéticos depende da densidade de exemplos na vizinhança do ponto atual. Nesse sentido, regiões com menor densidade de dados pertencentes a classes minoritárias têm maiores dados sintéticos gerados (HE *et al.*, 2008).

Um exemplo de classes rotuladas de 0 a 4 pode ser observado no Quadro 1, onde é apresentado a quantidade de instâncias por classe sem técnica de reamostragem e com as técnicas de reamostragem.

2.3.3 Remoção de ruído com filtro Neurokit

A remoção de ruído é uma fase de pré-processamento importante para legibilidade dos dados e para facilitar o aprendizado de modelos sobre os dados. O filtro utilizado no presente trabalho é *Neurokit* pertencente à biblioteca Neurokit2, responsável por limpar os sinais de eletrocardiograma e melhorar a precisão na detecção de pico. O filtro têm sido amplamente utilizado por pesquisadores e médicos por resultar em uma melhor qualidade do exame e ser de fácil uso. Ademais, o filtro *Neurokit* consiste em um filtro passa-alto *Butterworth* de 0,5 Hz (ordem = 5), seguido de filtragem *powerline* (MAKOWSKI *et al.*, 2021).

2.4 Aprendizado supervisionado

Na aprendizagem supervisionada, os algoritmos dependem de dados que possuem a informação da variável alvo, denominada como rótulo, que pode conter categorias ou valores numéricos. O tipo do rótulo influencia na definição do problema, já que para problemas de classificação o objetivo é prever uma categoria e problemas de regressão têm a intenção de prever dados numéricos.

A classificação é definida por Rezende (2003) como um problema que tem a finalidade de gerar um classificador que rotule corretamente a classe de novos dados. Sendo assim, o problema abordado nesse trabalho é definido como problema de classificação, considerando pretenso o uso do exame ECG, onde os sinais capturados pelo exame são os dados de entrada e os rótulos são os diagnósticos. Portanto, neste trabalho são apresentados algoritmos classificatórios.

2.4.1 Árvore de decisão

A Árvore de Decisão (AD) é um dos algoritmos de aprendizado supervisionado mais utilizados em AM por possuir uma estrutura de fácil compreensão, não precisar de muito preparo prévio dos dados e por poder realizar atividades de classificação e regressão (GÉRON, 2019).

O algoritmo estabelece uma estrutura de árvore através da identificação de padrões apresentados pelos atributos e rótulos entregues a ele. A árvore é composta por um nó raiz, nós internos, nós folhas e ramos. Nessa composição são construídas condições capazes de retornar decisões que resultam na classificação de um novo dado.

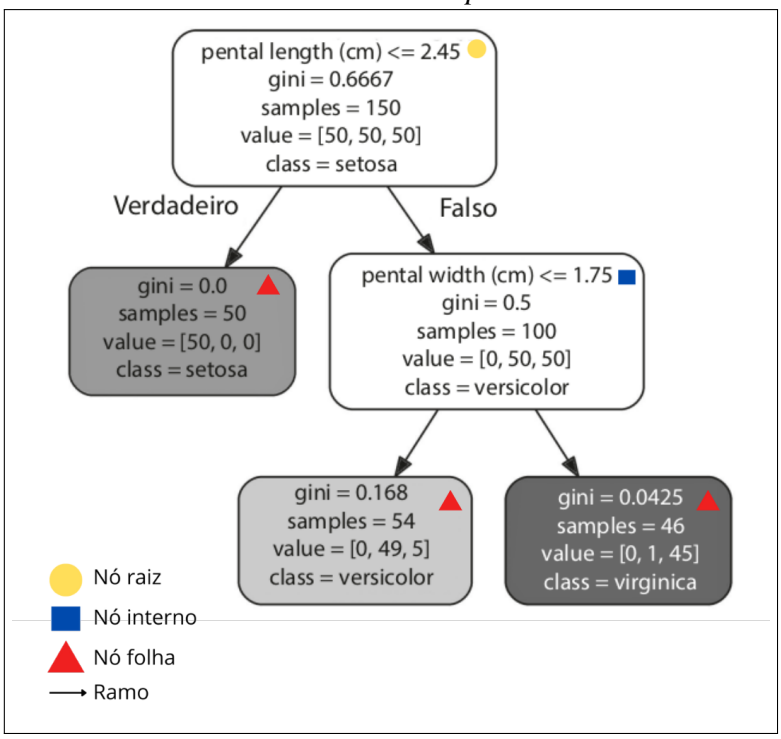
A estrutura da árvore é apresentada na Figura 2, representando o problema de classificação tradicional *Íris species*[1]. O conjunto de dados *Íris species* contém dados sobre as características da flor íris, sendo eles, o comprimento da sépala, largura da sépala, comprimento da pétala, largura da pétala e o rótulo categórico que pode ser do tipo virgínica, setosa ou versicolor. Por meio desses dados a AD faz inferências que relacionam os atributos com os rótulos.

Ao estabelecer parâmetros e os dados para o algoritmo de AD, é gerado um modelo. Um modelo contém informações aprendidas pelo algoritmo e através dele podem ser feitas predições. Na Figura 2 as condições estabelecidas pelo modelo gerado são representadas dentro do nó, se a condição de um nó for verdadeira para um determinado dado ele passa pelo ramo do

[1] https://www.kaggle.com/datasets/uciml/iris

lado esquerdo, se não, passa pelo ramo do lado direito. Um dado passa por essas condições até que chegue em um nó folha que determina a que classe ele pertence.

Figura 2 – Árvore de decisão da base de dados *Íris species*



Fonte: Adaptada de Géron (2019).

Apesar da facilidade no uso da AD, a abordagem contém a desvantagem do sobreajuste. Um modelo contém sobreajuste quando é apresentado um bom desempenho com dados de treino e um desempenho ruim com dados de teste. Diante desta desvantagem e a fim de aproveitar a boa estrutura da AD, a técnica de aprendizado *ensemble* consegue resolver esse problema, já que usa a classificação de um conjunto de modelos. Por isso, esse trabalho aborda essa técnica.

2.4.2 Aprendizado em conjunto

De acordo com Géron (2019), a técnica de aprendizado em conjunto é a agregação das predições de um conjunto de modelos. Considerando que, modelos de diferentes algoritmos podem apresentar sobreajuste e subajuste, os métodos da técnica de aprendizado em conjunto resolvem esse problema usando diferentes modelos de AM. Os métodos mais utilizados dessa técnica são *bagging* e *boosting*.

A técnica de *bagging* treina vários modelos de AD de forma independente, o que significa que os modelos não acessam informações uns sobre os outros. Além disso, cada modelo

treina com uma subamostra aleatória do conjunto de dados. Após o treino de cada modelo, a classificação da técnica é feita por um processo de votação entre os modelos, onde a classe mais votada para um dado X é definida como a predição final.

Já a técnica de *boosting* treina os modelos sequencialmente, permitindo que cada modelo aprenda com os erros do anterior. Com essa estratégia, cada modelo define pesos sobre as instâncias e as instâncias classificadas mais incorretamente possuem pesos maiores, consequentemente fazendo com que o próximo modelo se dedique mais a corrigir o erro do anterior. Por fim, a classificação é uma votação ponderada onde a votação dos modelos que tiveram melhores resultados têm mais peso.

2.4.3 Floresta aleatória

Conforme Pedregosa *et al.* (2011), a Floresta Aleatória (RF) é um algoritmo que aplica o método de *bagging*. Esse método obtém resultados de modelos de um mesmo algoritmo que utilizam subamostras aleatórias do conjunto de dados e considera a classificação majoritária desses modelos para determinar a predição. Precisamente, o algoritmo gera vários modelos de AD que introduzem a aleatoriedade. A vantagem de sua utilização está na diversidade das árvores construídas que evitam o sobreajuste, por consequência obtém um bom modelo frequentemente.

Apesar de evitar o sobreajuste, a floresta aleatória é rápida para treinar, mas lenta para fazer predições, isso acontece devido à quantidade de árvores geradas. A solução viável para esse problema é descobrir os hiperparâmetros que podem reduzir a quantidade de árvores sem reduzir a precisão (GÉRON, 2019).

2.4.4 Extremely Randomized Trees

O algoritmo *Extremely Randomized Trees* (Extra Trees) utiliza a técnica de *bagging* que treina de forma independente vários modelos de AD. Seu diferencial dos outros algoritmos de *bagging* é que introduz a aleatoriedade desde as subamostras de dados do conjunto por modelo até os pontos de divisão para cada nó da árvore. A estratégia de aleatoriedade do Extra Trees evita o sobreajuste e tem uma velocidade maior no processo de treino quando comparado ao algoritmo RF.

2.4.5 Gradient boosting

No aprendizado em conjunto, enquanto o método de *bagging* treina os modelos de forma independente, o método de *boosting* gera modelos que aprendem com os erros dos outros. O algoritmo explicado nessa seção faz a utilização do método de *boosting*.

Segundo a descrição de Pedregosa *et al.* (2011), para que o *Gradient boosting* (GB) utilize de modelos que aprendam com os erros dos outros, os modelos são treinados sequencialmente, identificando erros com a aplicação de gradientes na função de perda. A métrica de função de perda retorna a qualidade dos coeficientes utilizados nos modelos para realizar a classificação.

2.4.6 Extreme gradient boosting

O *Extreme Gradient Boosting* (XGB) é um algoritmo melhor computacionalmente quando comparado com o GB e assim como o GB faz a utilização do método de *boosting*. O algoritmo XGB é considerado flexível devido à quantidade de hiperparâmetros que podem ser alterados, além disso, seu diferencial quando comparado com outros algoritmos que utilizam os gradientes está na penalização inteligente de árvores, a redução proporcional dos nós folha e o uso do método de otimização reforço de Newton (CHANG *et al.*, 2018).

2.4.7 CatBoost

O *CatBoost* utiliza da técnica de *boosting*, onde os modelos aprendem uns com os outros através da atribuição de pesos as instâncias. Seu funcionamento acontece como XGB tendo como diferenciais o suporte a recursos categóricos, estratégia para valores ausentes, treino dos modelos mais rápidos e otimização de hiperparâmetros próprios.

2.4.8 LightGBM

Projetado para ser um algoritmo de alto desempenho, o LightGBM é um algoritmo de *boosting* que utiliza histogramas no treino dos modelos de AD, evitando que o algoritmo ordene os dados no decorrer do processo de treino, o que o torna mais eficiente em termos computacionais.

2.4.9 K-nearest neighbors

O *K-nearest neighbors* (KNN) é um algoritmo de aprendizado supervisionado. A ideia principal do algoritmo para encontrar um rótulo para um dado desconhecido é selecionar os k exemplos já rotulados que são mais parecidos com o dado analisado. No problema de classificação, o dado a ser rotulado irá receber o valor mais frequente entre os k selecionados. Esse algoritmo tem um treinamento muito simples e rápido, entretanto o custo para rotular um novo dado é relativamente alto (FERRERO, 2009).

Ao treinar o modelo, a quantidade de dados utilizado no treino pode ser grande e usar todos eles pode deixar o processo de rotulação lento. Para melhorar a eficiência, pode-se utilizar apenas os dados que melhor representa cada conjunto de dados. Assim, pode-se analisar menos candidatos ao escolher os k melhores (FERRERO, 2009).

Para calcular a similaridade entre os dados de treino e o dado que será rotulado, diversas medidas foram propostas para calcular distância e correlação. No contexto de dados numéricos, a distância pode ser aplicada para calcular a similaridade entre os dados. As distâncias mais utilizadas são a Manhattan e a Euclidiana (FERRERO, 2009).

A fórmula para calcular a distância entre dois dados P e Q de dimensão n usando a distância Manhattan pode ser vista na Equação 2.2 e a distância Euclidiana pode ser vista na Equação 2.3.

$$D_{Manhattan}(P,Q) = \sum_{i=1}^{n} |p_i - q_i| \tag{2.2}$$

$$D_{Euclidiana}(P,Q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{2.3}$$

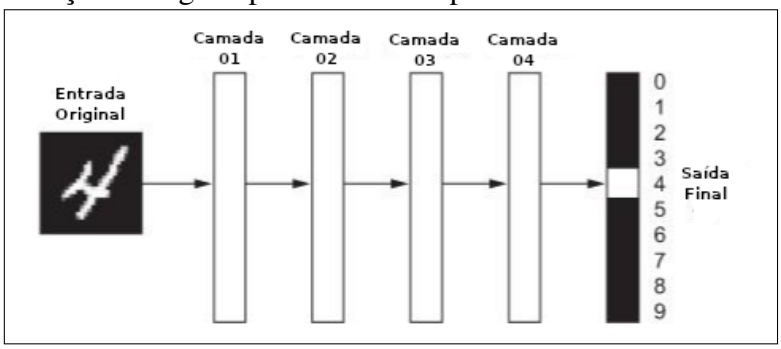
2.5 Aprendizado profundo

O aprendizado profundo (AP) é uma subárea do AM, em que se diferencia por ser baseado em Redes Neurais Artificiais (RNA). O termo profundo nessa subárea refere-se a forma em que o aprendizado é feito, através de muitas camadas, enquanto em outras abordagens a aprendizagem acontece em no máximo duas camadas, o que é considerado uma aprendizagem superficial (CHOLLET, 2021).

2.5.1 Redes neurais artificiais

O aprendizado em camadas é realizado por modelos de RNA. Uma representação sobre o aprendizado em camadas é apresentada na Figura 3, em que o dado nesse caso é a imagem do número quatro escrito à mão. Para ser feita a classificação esse dado passa por quatro camadas.

Figura 3 – Classificação de dígitos por rede neural profunda



Fonte: Adaptada de Chollet (2021).

A especificação do que as camadas fazem depende das definições dos pesos, da função de perda e do otimizador. Os pesos são valores numéricos passados para as camadas, inicialmente são valores aleatórios aplicados em transformações dos dados. Durante a etapa de treino sobre os dados, após a camada aplicar a transformação e obter predições, a função de perda calcula a distância entre o valor predito e o valor real, em seguida, utilizando o resultado da função de perda, o otimizador ajusta os pesos para a camada seguinte a fim de tornar as predições mais semelhantes com os valores reais (CHOLLET, 2021).

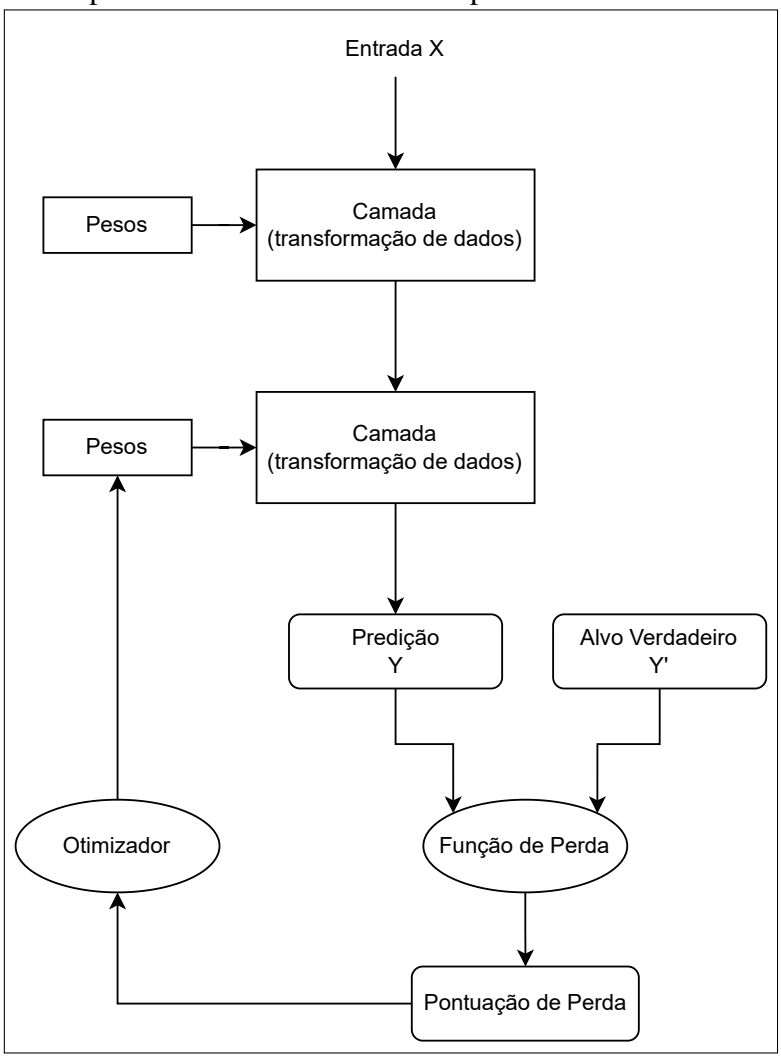
A Figura 4 representa a rede neural profunda, onde é apresentado como um laço de repetição o processo de ajuste dos pesos.

2.5.2 Redes Neurais Recorrentes

Redes Neurais Recorrentes (RNN) é um tipo de RNA que utiliza multicamadas e possui ligações de realimentação. O processo de realimentação funciona com as saídas das camadas podendo ser ligadas as entradas da própria camada ou das camadas anteriores. Com essa topologia de rede muitos problemas relacionados a dados sequenciais podem ser solucionados (FREIRE *et al.*, 2009).

Ao trabalhar com séries temporais é possível adicionar uma defasagem no tempo ao implementar a realimentação. Isso significa que os dados gerados nas etapas de treinamento

Figura 4 – Estrutura representacional de rede neural profunda



Fonte: Adaptada de Chollet (2021).

anteriores podem ser utilizados nas camadas seguintes. Isso ajuda a lidar com dados que cujo contexto é importante para a interpretação. (FREIRE *et al.*, 2009).

2.5.3 Rede de Memória de Longo Prazo e Curto Prazo

O algoritmo Rede de Memória de Longo Prazo e Curto Prazo (LSTM) foi criado em 1997 por Hochreiter e Schmidhuber (1997) com a proposta de armazenar informações em intervalos de tempo baseado no gradiente. Atualmente, o algoritmo é conhecido por fazer boas predições e classificações de séries temporais.

Hochreiter e Schmidhuber (1997) explica que as camadas de memória da LSTM guardam e manipulam os dados por meio de portões. Esses portões são os responsáveis por controlar a passagem dos dados por meio das camadas, eles são definidos como portão de entrada, portão de esquecimento e o portão de saída.

O portão de entrada adiciona dados as camadas, o portão de esquecimento decide qual informação pode ser descartada e o portão de saída seleciona as informações importantes para passar para a camada seguinte. Através dessas decisões, a LSTM salva informações para mais tarde, evitando que os sinais antigos desapareçam gradualmente durante o processamento (CHOLLET, 2021).

2.5.4 Gated Recurrent Unit

Gated Recurrent Unit (GRU) é uma rede neural semelhante a LSTM porém mais simples. Assim como a LSTM o GRU também utiliza de portões para controlar o fluxo dos dados. No entanto, ao invés de utilizar o portão de entrada e de esquecimento, ele utiliza o portão de atualização para controlar informações novas e antigas. Através dessa estrutura, tem-se um modelo mais simples, mas com um bom desempenho (MELLO, 2021).

2.5.5 Redes Neurais Convolucionais

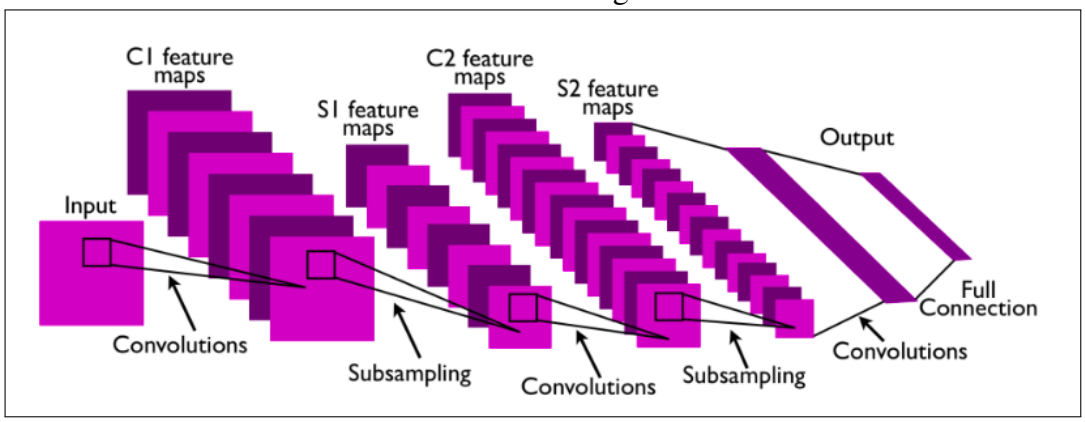
A Rede Neural Convolucional (CNN) é uma expansão das redes neurais tradicionais. Nesse algoritmo é adicionada algumas etapas de transformações de dados e por isso é muito utilizada para processamento de imagens. As etapas que compõem a CNN são: convolução, *pooling* e *Fully-Connected* (JURASZEK *et al.*, 2014).

As camadas de convolução são responsáveis por extrair características da imagem. Isso é feito a partir da operação de convolução entre a matriz de uma etapa e uma matriz de convolução chamada de *kernel*. Essa matriz *kernel* é ajustada automaticamente a cada iteração de treinamento do algoritmo (SILVA *et al.*, 2018).

Após cada camada de convolução é adicionada uma camada de *pooling*, onde é aplicada uma função de redução de matriz. Uma função muito utilizada na camada é a função *max*, no qual a cada submatriz de tamanho 2x2 pega-se o maior elemento, através dessa abordagem é gerado uma matriz com menor dimensão (JURASZEK *et al.*, 2014).

Por fim, após algumas etapas de convolução e *pooling*, tem-se as camadas *Fully-Connected*. Essas camadas têm o mesmo funcionamento de uma rede neural densa, no qual todos os neurônios de uma camada são ligados a todos os neurônios da camada subsequente(LECUN *et al.*, 2010). A Figura 5 apresenta a esttutura de uma CNN com dois estágios.

Figura 5 – Rede neural convolucional com dois estágios



Fonte: *Convolutional Networks and Applications in Vision* (LECUN *et al.*, 2010).

2.6 Aprendizado de máquina automatizado

Criado para otimizar o desenvolvimento de AM e AP, o AutoML tem sido um tema importante na indústria e no meio acadêmico. Isso decorre por ele lidar com a complexidade dos algoritmos, mantendo a facilidade em seu uso e gerando uma economia de tempo. Resumidamente, o AutoML automatiza as etapas comuns em um projeto, sendo elas: seleção de algoritmos, pré-processamento e escolha de hiperparâmetros. Ademais, além de realizar as etapas comuns, ele detecta anomalias e gera modelos que vão se ajustando com os dados, ou seja, ele vai obtendo hiperparâmetros que o ajudem a realizar melhores classificações para cada algoritmo (HE *et al.*, 2021).

2.6.1 Hiperparâmetros

Segundo Yang e Shami (2020), hiperparâmetro é uma característica de um modelo, onde o valor não pode ser estimado a partir de dados e para a descoberta de hiperparâmetros ideais na geração de um modelo são necessários testes e comparações. Os hiperparâmetros são comumente definidos com valores padrão em algoritmos de bibliotecas e fazem uso desses valores para realizar a classificação. Não necessariamente o hiperparâmetro definido por padrão é o ideal para gerar o melhor modelo.

Para Yang e Shami (2020), não utilizar hiperparâmetros que se adéquem melhor ao algoritmo e aos dados usados, ocasiona na redução da precisão do modelo. Sendo assim, os problemas citados na Seção 2.4.3 podem ser evitados ao identificar hiperparâmetros que gerem a melhor solução. Para isso existem abordagens diferentes para avaliar modelos de um mesmo algoritmo com diferentes hiperparâmetros.

Os algoritmos de otimização de hiperparâmetros mais comuns são o *randomized search* e *grid search*, onde em ambas técnicas é necessário definir uma lista de hiperparâmetros para que modelos sejam testados e avaliados (COMPARING. . . ,).

Em um cenário de otimização de recursos computacionais, um meio é utilizar o AutoML para encontrar esses hiperparâmetros. Nesse trabalho é utilizado a ferramenta *AutoGluon* que é capaz de adaptar automaticamente o processo de treino segundo a complexidade do problema e os recursos disponíveis. Dessa maneira, os algoritmos não ficam limitados a definições de hiperparâmetros de um usuário (AUTOGLUON,).

2.6.2 AutoGluon

A ferramenta *AutoGluon* é uma ferramenta de AutoML, que se diferencia de outras estruturas por se concentrar na seleção de modelos/hiperparâmetros, reunindo vários modelos e os empilhando em várias camadas. Experimentos revelaram que a combinação multicamadas de vários modelos faz um melhor uso do tempo de treino (ERICKSON *et al.*, 2020).

Ademais, foi feita uma avaliação extensa de plataformas AutoML, sendo elas: TPOT, H2O, *AutoWEKA*, *auto-sklearn*, *AutoGluon* e *Google AutoML Tables*. Por consequência, os testes em um conjunto de 50 tarefas de classificação e regressão do *Kaggle* e do *OpenML AutoML Benchmark* indicaram que o *AutoGluon* é mais rápido, mais robusto e muito mais preciso (**??**).

De forma geral, o *AutoGluon* realiza técnicas de pré-processamento para valores ausentes e para categorização de classes a fim de evitar erros no processo de classificação dos modelos; seleciona os modelos de forma automática tendo em vista o tipo de problema; realiza ajustes dos hiperparâmetros; cria de forma automática conjuntos de modelos; e permite o treino paralelo em várias máquinas.

2.7 Métricas de desempenho

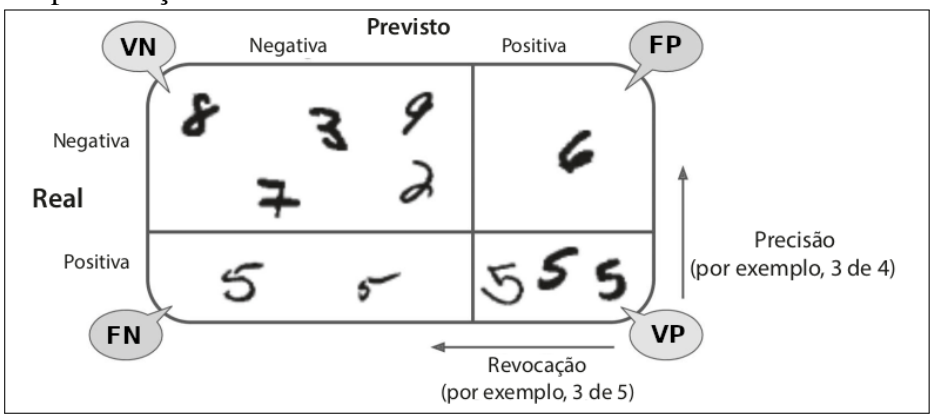
A utilização de métricas é importante para avaliar o desempenho dos modelos em realizar os seus objetivos. Para modelos classificatórios destacam-se as métricas de matriz de confusão, acurácia, precisão, revocação e F_1-score.

2.7.1 Matriz de confusão

A matriz de confusão é utilizada como base para a acurácia, precisão, revocação e F_1-score. Consiste em uma matriz, onde cada linha representa uma classe real, enquanto as colunas representam as classes preditas (GÉRON, 2019). A Figura 6 mostra uma matriz de confusão para o problema de números escritos à mão mencionado na Seção 2.5.1, nessa matriz é representada a classificação da classe 5 e é destacado classificações consideradas verdadeiros negativos (VN), verdadeiros positivos (VP), falsos negativos (FN) e falsos positivos (FP).

Para esse problema em específico há uma classificação binária onde é determinado como verdadeiro negativo quando foi predita como não pertencente a classe 5 e isso é verdade. O verdadeiro positivo é determinado quando a predição indica que aquele dado é da classe 5 e isso é verdade. Já o falso negativo é quando o dado foi classificado como diferente de 5 e não é. Por fim, o falso positivo é o dado que foi rotulado como da classe 5, mas não é da classe 5.

Figura 6 – Representação da matriz de confusão

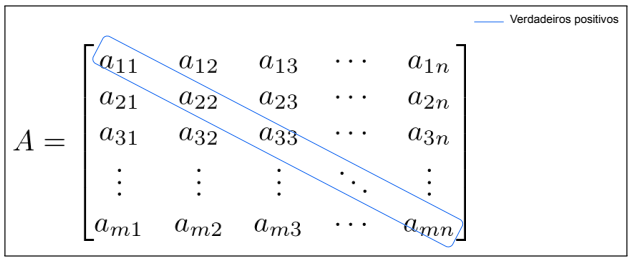


Fonte: Adaptada de Géron (2019).

A Figura 7 apresenta uma matriz genérica que facilita o entendimento sobre a matriz de confusão multiclasse. Na matriz de confusão não há verdadeiro negativo, considerando que o problema não é mais binário. Os verdeiros positivos sempre estão na diagonal principal, dessa forma para a classificação de 1 a m, a quantidade dos verdadeiros positivos de 1 estão na posição que o índice da linha é igual ao da coluna. Os valores que estão na mesma coluna, mas que possuem índices de linhas diferentes são os falsos positivos, já os elementos que possuem o mesmo índice da linha e índices de colunas diferenças são falsos negativos.

Com a explicação da matriz genérica da Figura 7, é possível compreender a matriz de confusão sobre um problema de classificação multiclasse rotuladas de 1 a 5. Os Capítulos 2.7.1.1, 2.7.1.2, 2.7.1.3 e 2.7.1.4 utilizam dos dados da matriz de confusão da Figura 8 com

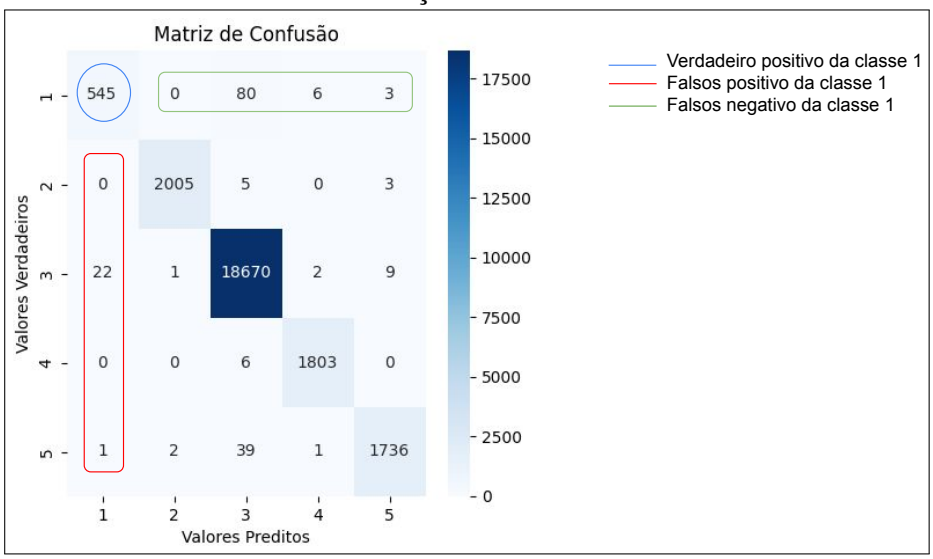
Figura 7 – Matriz genérica



Fonte: Elaborada pela autora.

intuito de ilustrar o cálculo de métricas avaliativas para classificações multiclasse.

Figura 8 – Matriz de confusão de classificação multiclasse



Fonte: Elaborada pela autora.

2.7.1.1 Acurácia

A acurácia é uma métrica que evidencia os acertos, independente dos rótulos. Ela é calculada através da divisão entre a quantidade de dados classificados corretamente sobre a quantidade de dados previstos (GÉRON, 2019). A equação 2.4 apresenta o cálculo da acurácia sobre a matriz de confusão da Figura 6 e a equação 2.5 da Figura 8.

$$\text{Acurácia para problema binário} = \frac{VP+VN}{VP+VN+FP+FN} = \frac{3+5}{3+5+1+2} = 0,72 \tag{2.4}$$

$$\text{Acurácia para problema multiclasse} = \frac{VP}{VP+FN} = \frac{24759}{24759+180} = 0,9927 \tag{2.5}$$

2.7.1.2 *Precisão*

A precisão indica a proporção de dados pertencentes a uma classe que foram classificados corretamente. O cálculo da precisão é a divisão entre os classificados corretamente como pertencentes à classe sobre o total de classificações corretas (GÉRON, 2019). Para um problema de classificação binária, como do dígito 5 da Figura 6, a precisão pode ser calculada como está na equação 2.6. Para o problema multiclasse da Figura 8, é calculado a precisão do modelo sobre a classe 1 na equação 2.7.

$$\text{Precisão do dígito 5} = \frac{VP}{VP+FP} = \frac{3}{3+1} = 0,75. \tag{2.6}$$

$$\text{Precisão da classe 1} = \frac{VP}{VP+FP} = \frac{545}{545+23} = 0,9595. \tag{2.7}$$

2.7.1.3 *Revocação*

Diferente da precisão, a revocação dá ênfase para os erros por falso negativo como apresentado na função 2.8, em que a quantidade de rótulos de uma classe preditos corretamente é dividida pelo soma das quantidades entre os acertos e os erros (GÉRON, 2019). A equação 2.8 calcula a revocação da classe 1 do problema multiclasse apresentado na Figura 8.

$$\text{Revocação} = \frac{VP}{VP+FN} = \frac{545}{545+89} = 0,8596 \tag{2.8}$$

2.7.1.4 *F_1-score*

O *F_1-score* é calculado pela multiplicação de 2 pela fração que contém a multiplicação de precisão com a revocação sobre a soma da precisão com a revocação. Essa é uma média harmônica da precisão e revocação, nela o modelo terá o percentual alto se a revocação e a precisão forem altas (GÉRON, 2019). Considerando que os valores de precisão e revocação da classe 1 da Figura 8 foram calculados nas Seções 2.7.1.2 e 2.7.1.3, a equação 2.9 calcula o *F_1-score* da classe 1.

$$F_1\text{-score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Revocação}}{\text{Precisão + Revocação}} = 2 \cdot \frac{0,9595 \cdot 0,8596}{0,9595 + 0,8596} = 0,9068 \tag{2.9}$$

2.7.2 Intervalo de confiança

O Intervalo de confiança calcula os limites para um parâmetro populacional. Em AM, os limites são definidos sobre o desempenho de um modelo. No contexto de AM é importante calcular o intervalo de confiança para casos em que os percentuais de acertos de modelos são aproximados.

Sendo assim, obtendo intervalos de confiança sobre modelos, quando um modelo X não possui interseção entre os intervalos de confiança de outros modelos e o percentual de acerto do modelo X é o maior, pode-se definir X como melhor modelo.

A equação 2.10 apresenta o método de Wilson para intervalo de confiança, onde *erro* é o percentual de erro da classificação, *const* define a probabilidade escolhida, n é o tamanho da amostra classificada.

$$erro \pm const \cdot \sqrt{\frac{erro \cdot (1 - erro)}{n}} \tag{2.10}$$

2.7.3 Erro médio quadrático

O Erro Médio Quadrático (MSE) é utilizado para ter uma estimativa do quão diferente o valor estimado está do valor real. Assim, quando menor for o valor de MSE melhor é o desempenho do algoritmo. A equação 2.11 apresenta como o MSE pode ser calculado, onde n é o número total de observações, y_i é o valor observado na i-ésima observação e $\hat{y}_i$ é o valor predito ou estimado para a i-ésima observação.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.11}$$

3 TRABALHOS RELACIONADOS

Nesse capítulo são descritos os artigos que contém o mesmo contexto do trabalho proposto com soluções de classificação de doenças cardiovasculares (DCV). O capítulo faz contraste do trabalho proposto com os trabalho apresentado, pondo em evidência as diferentes técnicas de pré-processamento aplicadas sobre os dados e as técnicas sobre as estruturas dos algoritmos

3.1 Classificação de microclasses de ECG usando Rede Neural Convolucional

De acordo Wu *et al.* (2021), os algoritmos mais comuns na classificação de sinais de ECG são RNA, Máquina de Vetores de Suporte (SVM) e Análise de Componentes Independentes (ACI). Além disso, ele enfatiza que dado o grande volume de dados relacionados a ECG poucos trabalhos possuem classificação de microclasses.

Por isso, o objetivo de Wu *et al.* (2021) foi agregar para a literatura com uma abordagem utilizando uma CNN de 12 camadas para realizar a classificação de 5 subclasses do *Massashusetts Institute of Technology - Boston Beth Israel Hospital* (MIT-BIH). Para aplicar a abordagem mencionada, o trabalho realiza um pré-processamento sobre o conjunto de dados Arritmia MIT-BIH definindo a segmentação de dados como 2000, aplicando a padronização do escore Z e a remoção de ruído com o método da transformada de *wavelet*.

Wu *et al.* (2021) obteve a rede CNN com uma acurácia de 97,41%, sensibilidade de 97,05%, especificidade de 99,35% e taxa de previsão positiva de 97,21%, por fim, faz comparações precisas com modelos de RF e outras CNN.

Tendo em vista as informações apresentadas, o trabalho de Wu *et al.* (2021) é semelhante a algumas das estratégias abordadas no trabalho proposto, sendo eles: o método de transformação dos arquivos em dados pertencentes a um quadro de dados, modelos com lidam com a mesma segmentação de dados para microclasses. O diferencial deste trabalho com Wu *et al.* (2021) é sobre a quantidade de técnicas de pré-processamentos avaliadas, o uso da técnica de remoção de ruído da biblioteca Neurokit2[1] e por conter modelos de AM e AP.

[1] https://neuropsicologia.github.io/NeuroKit/

3.2 Análise e classificação de doenças cardíacas usando recursos de batimentos cardíacos e algoritmos de aprendizado de máquina

Alarsan e Younes (2019) propuseram a classificação de ECG usando aprendizado de máquina com base em recursos de ECG. Os autores realizam implementações usando *ML-libs* e a linguagem *Scala* no *framework Apache Spark*. Em seu trabalho reconhecem o desafio e a importância de classificar ECG por conter irregularidades nos sinais.

Sabendo que cada batimento é uma combinação produzida por diferentes tecidos cardíacos e que existem formas de onda que ocorrem de maneiras diferentes em cada pessoa, Alarsan e Younes (2019) utilizaram a transformada de *wavelet* para extrair recursos das ondas e utilizar nas entradas dos algoritmos.

Suas análises foram implementadas para um problema multiclasse e binário com as respectivas base de dados: Arritmia MIT-BIH e Arritmia Supraventricular MIT-BIH. Para o problema binário os modelos alcançaram um desempenho de 96,75% usando o algoritmo *Gradient Boosted Trees* e 97.98% usando RF. Para classificação multiclasse, apenas o algoritmo RF foi utilizado e o modelo alcançou uma acurácia de 98,03%.

O trabalho proposto se assemelha com o trabalho de Alarsan e Younes (2019) na geração de modelos da RF e por aplicar remoção de ruído, mas se diferencia sobre o tipo de filtro aplicado para remoção e gera modelos de RF que foram treinados para classificar duas estruturas do conjunto de dados Arritmia MIT-BIH: para 5 classes e para 16 classes.

3.3 Classificação de arritmia de ECG usando redes neurais recorrentes

De acordo com Singh *et al.* (2018), RNN e CNN são os campos mais interessantes do AP, mas em cenários que lidam com sinais de ECG a CNN corta os batimentos em pedaços de comprimento fixo, reduzindo o desempenho da classificação.

Por isso, Singh *et al.* (2018) gerou 3 modelos de RNN com estratégias diferentes para concluir sua análise. Os modelos possuem 3 camadas cada, possuem variações sobre a quantidade de neurônios utilizadas em cada camada e a função de perda definida como MSE. Os algoritmos de RNN para cada modelo variam, onde a primeira trata-se de uma RNN comum, a segunda uma RNN com GRU e a terceira uma LSTM.

As acurácias de seus modelos foram de 88,1% para a LSTM, para a RNN comum 85,4% e para a RNN com GRU foi de 82,5%. Segundo o autor, a complexidade de seus modelos

é menor do que as de algoritmos tradicionais de AM por não ter realizado pré-processamento. O autor conclui que para obter melhores resultados com os modelos é necessário o aumento de épocas nas RNN e realizando a classificação multiclasse.

O trabalho de Singh *et al.* (2018) é interessante pelos usos das RNN e como as estruturou e pode ser comparado com o trabalho proposto através dos modelos de RNA, com o diferencial de que o presente trabalho contém 18 combinações de estratégias de pré-processamento. Além de que, enquanto o presente trabalho realiza classificação multiclasse, Singh *et al.* (2018) adaptou o conjunto de dados Arritmia MIT-BIH para classificação binária.

3.4 Classificação de batimentos cardíacos ECG: uma representação profunda transferível

Em seu trabalho Kachuee *et al.* (2018) propôs uma estrutura capaz de representar o sinal de forma que as análises de ECG sejam transferíveis entre diferentes tarefas. Formaram essa estrutura com uma rede neural profunda que oferece uma grande capacidade de aprendizagem, mais precisamente relacionadas as características do sinal.

Kachuee *et al.* (2018) utilizou o conjunto de dados Arritmia MIT-BIH e *Physikalisch-Technische Bundesanstalt* (PTB) *Diagnostic ECG*. Os dados do conjunto de dados Arritmia *MIT-BIH* foram categorizados e direcionados para treino. Os do conjunto de dados PTB *Diagnostic ECG* foram selecionados apenas os que se tratavam sobre o infarto do miocárdio.

Apesar da classificação final ser o do conjunto de dados PTB *Diagnostic ECG* e do objetivo principal ser o aprendizado por transferência, é possível realizar comparações com o presente trabalho, já que Kachuee *et al.* (2018) também disponibiliza dados de desempenho do modelo de CNN apenas sobre o conjunto de dados Arritmia *MIT-BIH*. Isso é interessante em razão de que o modelo gerado foi criado para ter uma aprendizagem ampla de maneira que fosse aplicado para dois conjunto de dados.

3.5 Classificação do sinal de ECG usando técnicas de aprendizagem profunda com base no conjunto de dados PTB-XL

Śmigiel *et al.* (2021) apresenta informações que as doenças cardiovasculares continuam sendo a principal causa de mortalidade em todo o mundo e indica que uma das principais causas é a arritmia cardíaca. Além disso, enfatiza o ECG como um exame comum, confiável e não invasivo.

Os autores evidenciam que para uma interpretação manual do ECG é preciso habilidade e que interpretações errôneas podem ser decisivas nas vidas dos pacientes que precisam de um diagnóstico rápido para o tratamento da doença. Por isso, propõe o uso do aprendizado profundo para facilitar classificações de arritmia cardíaca.

A análise e classificação no trabalho de Śmigiel *et al.* (2021) foi realizada com o conjunto de dados PTB-*XL* [2]. As arquiteturas de rede neural implementadas foram de CNN. As CNN que obtiveram melhor desempenho uma com recursos baseados em entropia e uma rede simples. A CNN com recursos baseados em entropia obteve os melhores resultados, no entanto, a rede que não utilizou dos recursos baseados em entropia não apresentou resultados inferiores e desempenhou uma maior eficiência computacional.

O trabalho de Śmigiel *et al.* (2021) se assemelha ao presente trabalho pelo uso de uso RNA e pelo seu conjunto de dados conter dados de arritmia, assim como o conjunto de dados utilizado no presente trabalho. As diferenças são sobre os conjuntos de dados que utilizam e os algoritmos de RNA.

3.6 Análise comparativa

As Seções 3.1, 3.2, 3.3 e 3.5 esclarecem as técnicas e objetivos dos artigos, ao final de cada seção foram descritas as contribuições para esse trabalho e as diferenças. O Quadro comparativo 2 torna visível o que foi utilizado em cada trabalho e as pretensões deste trabalho.

[2] https://physionet.org/content/ptb-xl/1.0.1/

Quadro 2 – Comparação entre os trabalhos relacionados e este trabalho.

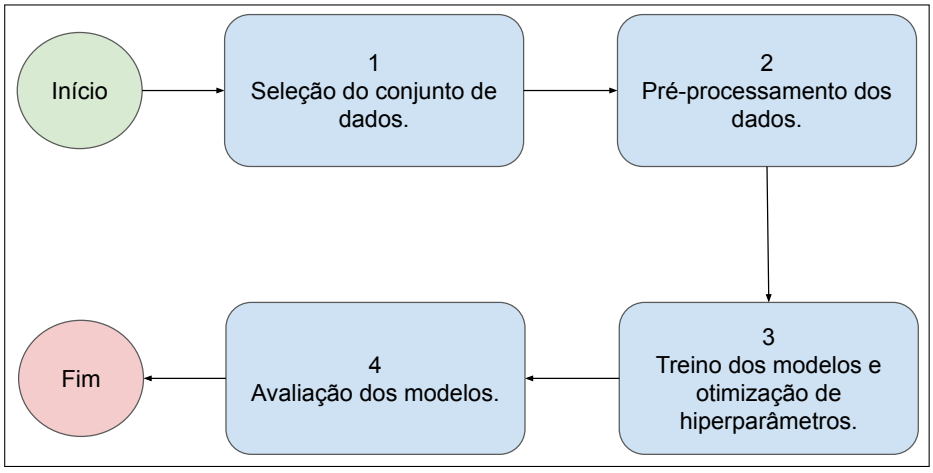
Trabalho	Classificação	Base de dados	Algoritmos
Wu *et al.* (2021)	Multiclasse (5 classes)	Arritmia MIT-BIH	CNN
Alarsan e Younes (2019)	Multiclasse (5 classes) e binária	Arritmia MIT-BIH e Arritmia Supraventricular MIT-BIH	RF e *Gradient Boosted Trees*
Singh *et al.* (2018)	Binária	Arritmia MIT-BIH	RNN, GRU e LSTM
Kachuee *et al.* (2018)	Multiclasse (5 superclasses)	PTB *Diagnostic ECG* e Arritmia MIT-BIH	CNN
Śmigiel *et al.* (2021)	Multiclasse (5 superclasses)	PTB-XL	CNN
Este trabalho	Multiclasse (5 classes) e (16 classes)	Arritmia MIT-BIH	KNN, RF, Extra Trees, XGB, LightGBM Catboost RNA do *Pytorch* RNA do FASTAI

Fonte: Elaborado pela autora.

4 PROCEDIMENTOS METODOLÓGICOS

O presente capítulo apresenta os passos a serem seguidos na construção deste trabalho, a Figura 9 exibe sequencialmente esses passos. Este capítulo segue a seguinte estrutura: a Seção 4.1 descreve o processo de seleção do conjunto de dados e esclarece o que é almejado no procedimento de análise; a Seção 4.2 apresenta justificativas sobre as decisões tomadas nas estratégias de pré-processamento; já a Seção 4.3 abrange o processo de treino e otimização dos hiperparâmetros utilizados e a Seção 4.4 discorre sobre o processo de avaliação dos modelos.

Figura 9 – Fluxuograma representacional dos procedimentos metodológicos



Fonte: Elaborada pela autora.

4.1 Seleção do conjunto de dados

Na produção de projetos relacionados a Ciência de Dados, os dados utilizados podem ser inventados, mas adquirir dados verdadeiros, limpá-los e transformá-los é um processo fundamental para obter conclusões sólidas (GRUS, 2016). Como apresentado no Capítulo 1, os dados selecionados para esse trabalho possuem relação com área da saúde, mais especificamente com a classificação de arritmia cardíaca.

A base de dados selecionada contém registros de ECG. O conjunto de dados chama-se Arritmia MIT-BIH [1]. Nesses dados coletados pelo Laboratório de Arritmias do Hospital Beth Israel, eles se certificaram de incluir batimentos normais e os que continham anormalidades. Os batimentos foram coletados através do ECG de homens entre 32 a 89 anos e mulheres de 23 a 89 anos (MIT-BIH. . . ,).

[1] https://physionet.org/content/mitdb/1.0.0/

O conjunto de dados Arritmia MIT-BIH é muito utilizado em AM para comparar o desempenho de algoritmos na classificação de batimentos cardíacos. Apesar de já conter muitas pesquisas relacionadas ao conjunto de dados, ele prossegue sendo utilizado em estudos que visam investigar técnicas que facilitem o processo de classificação do ECG, já que novas técnicas e abordagens surgem a cada dia em AM e AP, podendo gerar novos resultados e análises (MOODY; MARK, 2001).

Especificamente, o conjunto de dados Arritmia MIT-BIH possui 48 registros, 46 de pacientes diferentes e 2 do mesmo paciente, onde cada registro possui 30 minutos de coleta de um ECG. Esses dados foram selecionados a partir de uma base de 4.000 dados obtidos pelo Laboratório de Arritmia do Hospital Beth Israel entre 1975 e 1979. Desses 48 registros, 23 foram selecionados aleatoriamente e 25 foram selecionados de registros que continham arritmias ventriculares, juncionais e supraventriculares complexas e anormalidades de condução.

Esses dados foram coletados por meio de eletrodos posicionados no tórax, esses eletrodos foram nomeados como MLII, V1, V2, V4 E V5. Essa informação é relevante, pois cada registro possui informação de dois eletrodos, onde os dados coletados pelo eletrodo MLII aparece majoritariamente nos registros e os outros eletrodos aparecem variadamente.

Os dados são os sinais coletados pelos eletrodos, esses foram digitalizados a 360Hz em um canal com resolução de 11 bits com uma faixa de $10mV$. Ou seja, os dados de cada registro estão em milivolts e são números inteiros que variam de 0 a 2047. Com as informações apresentadas é possível especificar o que contém cada registro dos pacientes: três colunas, onde a primeira coluna contém os índices das amostras e as outras duas colunas são os sinais dos dois eletrodos.

Além dos registros, o conjunto de dados Arritmia MIT-BIH contém anotações referentes a cada registro, nessas anotações estão presentes as informações sobre os sinais, o tempo em que o batimento foi classificado e o índice da amostra referente.

O estudo afundo sobre o que se trata o conjunto de dados foi de extrema importância no trabalho proposto, considerando que possibilitou um melhor uso dos dados no processo de experimentação. Com essas informações foram feitas análises gráficas que são apresentadas na Seção 5.1.

4.2 Pré-processamento dos dados

Através das informações obtidas no processo de análise do conjunto de dados é que as etapas de pré-processamento são definidas, podendo ser elas: remoção de dados, aplicação de normalização, inferência de novos dados, etc. Um pré-processamento adequado facilita a interpretação dos modelos de AM e AP sobre o conjunto de dados, tendo como por consequência melhores classificações (GARCÍA *et al.*, 2015).

O pré-processamento feito neste trabalho sobre o conjunto de dados Arritmia MIT-BIH refere-se a transformação dos arquivos em um *DataFrame*, a normalização dos dados e remoção de ruídos. No processo da transformação dos arquivos em *DataFrame* é feita a combinação das informações de registros e de anotações, a fim de ter correlacionado um intervalo de coleta de sinais a um diagnóstico. Por conseguinte, é necessário a escolha sobre o tamanho do segmento dos dados, que o que define quantos sinais contém uma instância.

A fim de abranger a pesquisa desse trabalho são definidos segmentos de tamanho 360 e 2000 para o pré-processamento do conjunto de dados, ou seja, o presente trabalho lida com duas distribuições diferentes sobre o mesmo conjunto de dados. Quando o conjunto de dados é segmentado em 360, significa que cada instância do *DataFrame* possui 360 sinais elétricos do coração, o mesmo padrão acontece quando o conjunto de dados é segmentado em 2000, cada instância possui 2000 sinais elétricos. Sobre os sinais é importante destacar que possuem valores discrepantes quando se tratam de anormalidades detectadas e por isso, uma fase importante para o pré-processamento é utilizar a padronização do escore Z para centralizá-los em torno da média, que é uma fase etapa comum a ser seguida na utilização de dados de ECG.

Outra consideração sobre os sinais, é que eles contêm ruídos e isso pode atrapalhar a precisão da classificação. Por isso foi feita a implementação e análise sobre o filtro de remoção de ruído *Neurokit2* da biblioteca *Neurokit2*[2] que é uma biblioteca amplamente utilizada por pesquisadores que lidam com dados de ECG. O código 4.2.1 exibe as configurações feitas sobre a função *ecg_clean* do *Neurokit*, que aplica o filtro de remoção de ruído, onde *sampling_rate* foi definido como $360Hz$ já que é o mesmo do conjunto de dados e *method* como *neurokit*.

[2] https://neuropsychology.github.io/NeuroKit/

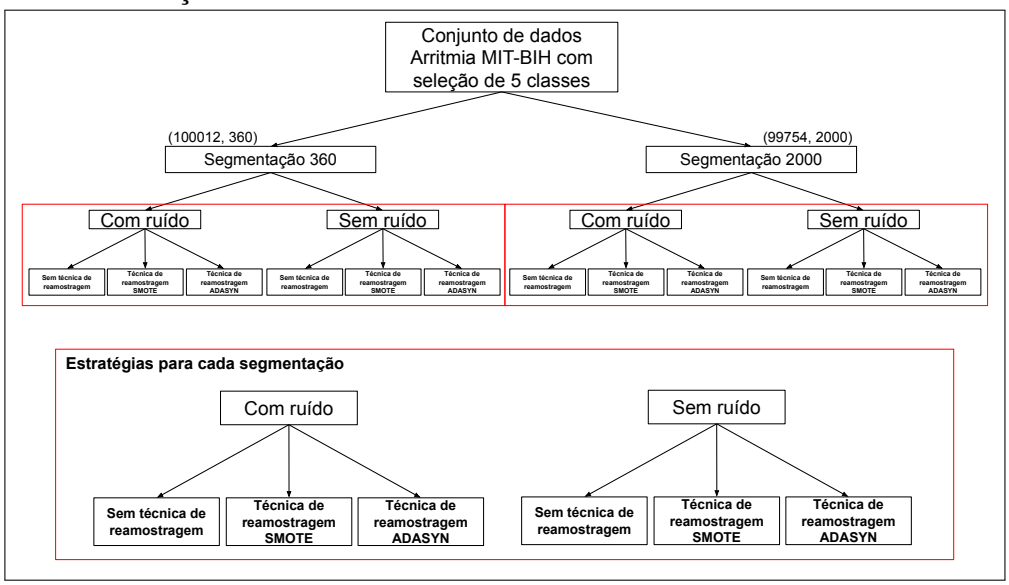
```
def denoise(ecg):
    #sampling_rate ( int ) - A frequência de amostragem do sinal (em Hz,
    #ou seja, amostras/segundo).
    data = nk.ecg_clean(ecg, sampling_rate=360, method="neurokit")
    return data
```

Listagem 4.2.1 – Método de remoção de ruído

Ademais, combinações sobre técnicas de reamostragem também foram implementadas sobre os dados de treino e sua distribuição sobre as classes em relação a cada técnica são apresentadas no Quadro 3. Mais detalhes do porquê as técnicas tiveram a quantidade de instâncias estabelecidas são explicadas na Seção 2.3.2.

Por fim, com todas as informações apresentadas, a Figura 10 e Figura 11 ilustram as diferentes estratégias de dados executadas no pré-processamento com objetivo de facilitar o entendimento sobre as metodologias desse trabalho, que em suma resultaram em 18 representações do conjunto de dados MIT-BIH.

Figura 10 – Estratégias de pré-processamento sobre o cojunto de dados Arritmia MIT-BIH para classificação de 5 classes



Fonte: Elaborada pela autora.

4.2.0.1 Classes selecionadas

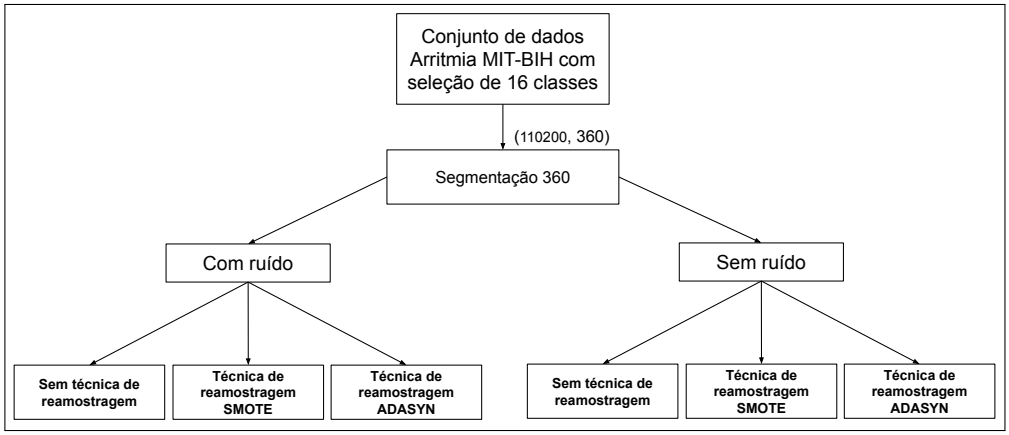
Como apresentado na Seção 4.1, os sinais de ECG possuem anotações. Essas anotações informam sobre o batimento, o ritmo e qualidade do sinal, ou seja, são específicas as anotações que classificam um problema de arritmia ou declara os batimentos como normais.

Quadro 3 – Pré-processamento sobre os dados de treino com reamostragem

	Dados de treino com segmentação 360 e 5 classes	Dados de treino com segmentação 360 e 16 classes	Dados de treino com segmentação 2000 e 5 classes
Sem técnica de reamostragem	'N': 56258 'L': 6053 'R': 5441 'V': 5347 'A': 1910	'N': 75011 'L': 8071 'R': 7255 'V': 7129 '/': 7023 'A': 2546 'f': 982 'F': 802 '!': 472 'j': 229 'x': 193 'a': 150 '|':132 'E': 106 'J': 83 'e': 16	'N': 56111 'L': 6040 'R': 5426 'V': 5335 'A': 1903
Com técnica de reamostragem SMOTE	'L': 56258 'N': 56258 'V': 56258 'R': 56258 'A': 56258	'N': 56258 'L': 56258 'R': 56258 'V': 56258 '/': 56258 'A': 56258 'f': 56258 'F': 56258 '!': 56258 'j': 56258 'x': 56258 'a': 56258 '|': 56258 'E': 56258 'J': 56258 'e': 56258	'L': 56111' N': 56111 'V': 56111 'R': 56111 'A': 56111
Com técnica de reamostragem ADASYN	'A': 56288 'N': 56258 'L': 6053 'R': 5441 'V': 5347	'e': 56262 'N': 56258 'L': 6053 'R': 5441 'V': 5347 '/': 5267 'A': 1909 'f': 737 'F': 601 '!': 354 'j': 172 'x': 145 'a': 113 '|': 99 'E': 80 'J': 62	'A': 56246 'N': 56111 'L': 6040 'R': 5426 'V': 5335

Fonte: Elaborado pela autora.

Figura 11 – Estratégias de pré-processamento sobre o cojunto de dados Arritmia MIT-BIH para classificação de 16 classes



Fonte: Elaborada pela autora.

Quadro 4 – Anotações de batimentos cardíacos e seus significados

Anotação de batimento	**Significado**
N	Batida normal
L	Batimento de bloqueio de ramo esquerdo
R	Batida de bloqueio de ramo direito
A	Batimento atrial prematuro
a	Batimento atrial prematuro aberrado
J	Batimento prematuro nodal (juncional)
S	Batimento prematuro supraventricular
V	Contração ventricular prematura
F	Fusão de batimento ventricular e normal
[	Início do flutter/fibrilação ventricular
!	Onda de vibração ventricular
]	Fim do flutter/fibrilação ventricular
e	Batimento de escape atrial
j	Batida de escape nodal (juncional)
E	Batida de escape ventricular
/	Batida ritmada
f	Fusão de ritmo normal e ritmo
x	Onda P não conduzida (APB bloqueada)
Q	Batida inclassificável
|	Artefato isolado semelhante a QRS

Fonte: Elaborado pela autora.

Para os batimentos existem 20 tipos de classificações no conjunto de dados Arritmia MIT-BIH, como são nomeados e seus respectivos significados são apresentados no Quadro 4.

Por ter essa grande variação entre as anotações de batimentos, os trabalhos relacionados ao MIT-BIH podem possuir quantidades diferentes de classes apesar de lidarem com o mesmo conjunto de dados e isso muda de acordo com os objetivos de cada trabalho. Por exemplo, o trabalho da Seção 3.1, Wu *et al.* (2021) utiliza de 5 rótulos para analisar a classificação, já na Seção 3.3, Singh *et al.* (2018) transforma o problema multiclasse em binário.

Nesse sentido, esse trabalho também possui a sua motivação sobre a escolha das

classes. Como apresentado na Seção de 4.2 o presente trabalho lida com segmentações de dados de tamanho 360 e 2000. Para a segmentação de 360 há dois conjunto de dados gerados do Arritmia MIT-BIH, onde um contém instâncias com 5 tipos de classes, sendo eles: N, L, R, A, V; já o outro contém instâncias com 16 tipos de classes: N, L, R, A, a, J, V, F, !, e, j, E, /, f , x, |. Para os dados que tiveram a segmentação como 2000 foi definido as 5 classes: N, L, R, A, V.

Especificando as escolhas das classes, no processo de experimentação das estratégias para os dados com segmentação 2000 não houve sucesso na execução da classificação de 16 classes por falhas no servidor disponível para execução, só foi houve sucesso na execução da classificação de das 5 classes selecionadas. Para os dados segmentados em 360 o processo de classificação funcionou normalmente para as 5 classes e 16 classes.

Sobre a quantidade das classes, o objetivo inicial era utilizar as 20 classes, no entanto, com o uso das técnicas de reamostragem SMOTE e ADASYN, existe a necessidade que cada classe tenha nos dados de treino pelo menos 6 amostras para realizar o balanceamento dos dados e isso não acontece na classe S,] e [. A classe Q também não foi utilizada por não possuir padrão, por de fato ser caracterizada como batimento desconhecido. Portanto, foram utilizadas 16 classes em um *DataFrame* que tiveram os dados segmentados em 360. Dessas 16, 5 foram selecionadas (N, L, R, A, V) por serem comumente utilizadas em trabalhos relacionados ao conjunto de dados Arritmia MIT-BIH, mantê-las facilitará o processo de comparação com outros trabalhos.

4.3 Treino dos modelos e otimização dos hiperparâmetros

Na Seção 1.1.2 foi mencionado o objetivo de gerar modelos de AM e AP utilizando AutoML e a seção atual detalha o porquê do objetivo e seu processo de implementação. Inicialmente, os trabalhos relacionados e suas análises, foram estudados e comparados, como diferencial o presente trabalho aplica uma maior quantidade de variações de hiperparâmetros.

Na Seção 2.6.1 é apresentado que os hiperparâmetros podem influenciar positivamente em modelos e que utilizá-los adequadamente tem como por consequências melhores classificações e um melhor aproveitamento do algoritmo. Com essas informações, a busca por como selecionar hiperparâmetros mais adequados para os algoritmos predispôs na escolha do AutoML para as tarefas de classificação deste trabalho. Já que o AutoML é considerado uma boa opção em casos de interesse em modelos complexos e computacionalmente caros com muitos hiperparâmetros (HUTTER *et al.*, 2019).

Na pesquisa comparativa de Erickson *et al.* (2020), estruturas de AutoML são

Quadro 5 – Hiperparâmetros utilizados no *TabularPredictor* e seus respectivos significados

Parâmetro	Significado	Valor
time_limit	Aproximadamente por quanto tempo fit() deve ser executado (tempo de relógio em segundos).	28800 e 7200
hyperparameter_tune_kwargs	Estratégia de ajuste de hiperparâmetros e kwargs	{'num_trials': 5,'scheduler' : 'local', 'searcher': 'auto'}
num_trials	Quantos testes de otimização de hiperparâmetros executar	5
scheduler	Qual agendador usar	local (Agendador local que agenda testes *First In, First Out*(FIFO))
searcher	Qual algoritmo de pesquisa usar	auto (Realiza busca de otimização bayesiana nos modelos NN_TORCH e FASTAI. Execute pesquisa aleatória em outros modelos)

Fonte: Elaborado pela autora.

testadas e analisadas, ele conclui que o *AutoGluon* Tabular é mais preciso do que estruturas mais populares e possui pré-processamento robusto, lida com dados heterogêneos, forma modelos complexos e bem avaliados.

Posto isso, o *AutoGluon* foi escolhido para gerar os modelos e encontrar melhores hiperparâmetros nesse trabalho, dessa forma é possível realizar as combinações de dados mencionadas na Seção 4.2 e gerar modelos diferentes que lidem com elas, possibilitando o alcance de uma maior quantidade de experimentações do que em um cenário que os modelos fossem testados e configurados manualmente.

O código 4.3.1 apresenta como a função *TabularPredictor* do *AutoGluon* foi implementada. A função *TabularPredictor* é ideal para o problema Arritmia MIT-BIH em razão de que os dados fornecidos são valores tabulares e são séries temporais, o qual são os tipos de dados que o *TabularPredictor* foi feito para lidar.

O Quadro 5 apresenta os hiperparâmetros utilizados e seus respectivos significados. No $time\_limit$, o tempo para modelos que classificaram 16 classes foi menor devido ao tempo de 4h e 8h se exceder ao ponto de não conseguir finalizar as execuções e o servidor perder a conexão. Para os modelos que classificaram 5 classes, foi definido 8h por ser o tempo mais alto que não acarretou em falha na conexão com servidor. Ademais, independente do problema de classificação, apesar do tempo definido, a fase de treino pode exigir um poder computacional maior devido à complexidade dos dados, a complexidade dos modelos e a busca por hiperparâmetros que resultem em melhores modelos não ter sido finalizada.

Sobre o $hyperparameter\_tune\_kwargs$, configurá-lo foi importante para poder acessar os dados de vários modelos sobre os mesmos algoritmos, já que por padrão o *AutoGluon* retorna apenas o melhor modelo de cada algoritmo. O crucial para esse resultado foi o parâmetro

scheduler que é o que permite acessar todos os modelos, ele significa que a busca de hiperparâmetros é executada localmente, o que apropriado para este trabalho considerando que outras configurações são destinadas a problemas de grande escala com recursos distribuídos.

Ademais, os outros parâmetros auxiliam o *scheduler*. O *num_trials* definido como 5 indica o número de buscas automatizada de hiperparâmetros. Já o parâmetro *searcher* definido como *auto* faz com que o *AutoGluon* fique responsável pela escolha do mecanismo da pesquisa tendo em vista os recursos passados para o método. Por meio disso, o *TabularPredictor* obteve muitas variações de modelos sobre os algoritmos: Redes neurais da biblioteca *FastAI*(NeuralNetFastAI), redes neurais da biblioteca *PyTorch* (NeuralNetTorch), *LightGBM*, *ExtraTrees*, RF, *XGBoost*, *CatBoost* e *KNeighbors*. Mais detalhes sobre os modelos são apresentados em 5.

```
predictor = TabularPredictor(label='label').fit(train_data, time_limit=28800,
hyperparameter_tune_kwargs = {'num_trials': 5,'scheduler' : 'local',
'searcher': 'auto'})
```

Listagem 4.3.1 – Implementação do *TabularPredictor*

4.4 Avaliação dos modelos

O método de avaliação dos modelos desse trabalho se refere a duas abordagens, já que muitas técnicas foram aplicadas sobre os dados e muitos modelos foram gerados. A primeira abordagem faz um comparativo dos modelos com as informações de pré-processamento, já a segunda abordagem evidência os comportamentos dos algoritmos em si.

Mais especificamente, na primeira abordagem cada modelo possui as métricas de desempenho apresentadas, sendo elas: acurácia, média da precisão, média da revocação e média do F_1-score. Com esses valores foi possível construir uma tabela comparativa dos modelos contendo as informações das estratégias de pré-processamento.

A segunda abordagem de avaliação, trata-se da análise dos modelos por algoritmo. Contendo informações sobre quais os melhores modelos por algoritmo, quais estratégias de pré-processamento de dados eles conseguiram percentuais de acertos maiores e como cada algoritmo tiveram seus hiperparâmetros variados.

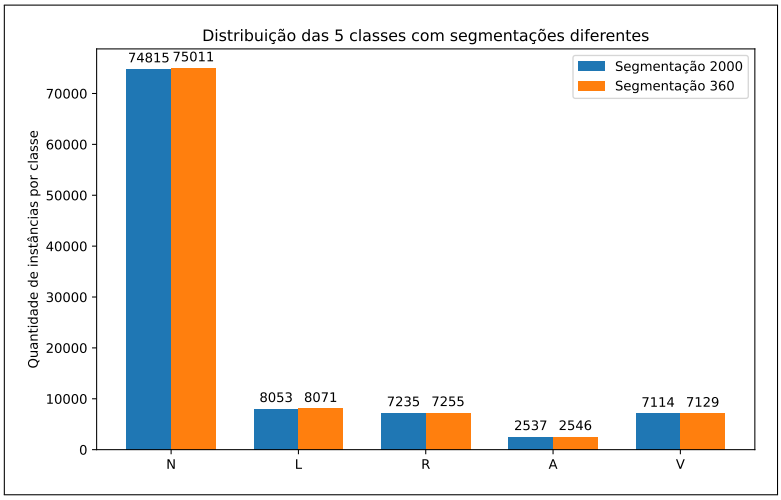
5 RESULTADOS

Este capítulo apresenta gráficos e conclusões sobre as experimentações realizadas com o conjunto de dados Arritmia MIT-BIH.

5.1 Análise visual do conjunto de dados

A presente seção tem a intenção de exibir as diferentes distribuições das classes do conjunto de dados com relação às decisões de pré-processamento e apresentar o comportamento dos dados por classe. Como mencionado na Seção 4.2, o presente trabalho lida com tamanhos de segmentações diferentes dos dados e com quantidade de classes diferentes. Uma comparação simples de se fazer são sobre as estruturas que contêm as mesmas classes, sendo assim, a Figura 12 mostra que apesar da diferença no tamanho das segmentações, os conjuntos de dados não adquiriram quantidades discrepantes de instâncias por classe.

Figura 12 – Histograma agrupado da distribuição dos dados para classificação de 5 classes com segmentações diferentes



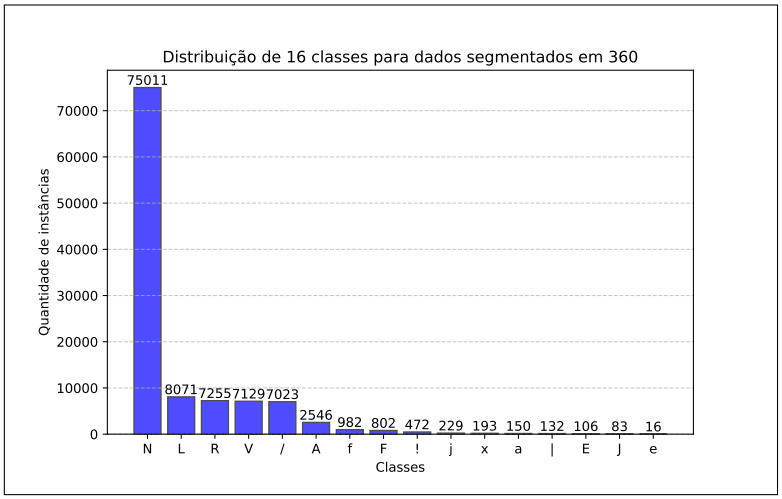
Fonte: Elaborada pela autora.

A Figura 13 apresenta a distribuição das 16 classes onde os dados foram segmentados a cada 360 sinais. Observando a figura, conclui-se que o conjunto de dados possui desbalanceamento entre as classes e que apesar de alguns dados terem sido coletados de pacientes em estado de emergência para garantir variedade de arritmias no conjunto de dados, ainda assim, os dados majoritários são de batimentos normais.

As classes de arritmia cardíaca além de estarem em menor quantidade no conjunto de dados, elas são difíceis de classificar pelas variações nos comportamentos das ondas e devido

aos ruídos. Os comportamentos das classes podem ser observados na Figura 14, onde são representados os dados de sinais ruidosos e dados que tiveram o filtro de remoção de ruído aplicado. Segundo as definições do filtro *Neurokit*, ele aplica mais alterações sobre sinais que possui variações lentas e esse ajuste pode ser observado no gráfico de batida de escape ventricular.

Figura 13 – Histograma de distribuição de 16 classes para dados segmentados em 360



Fonte: Elaborada pela autora.

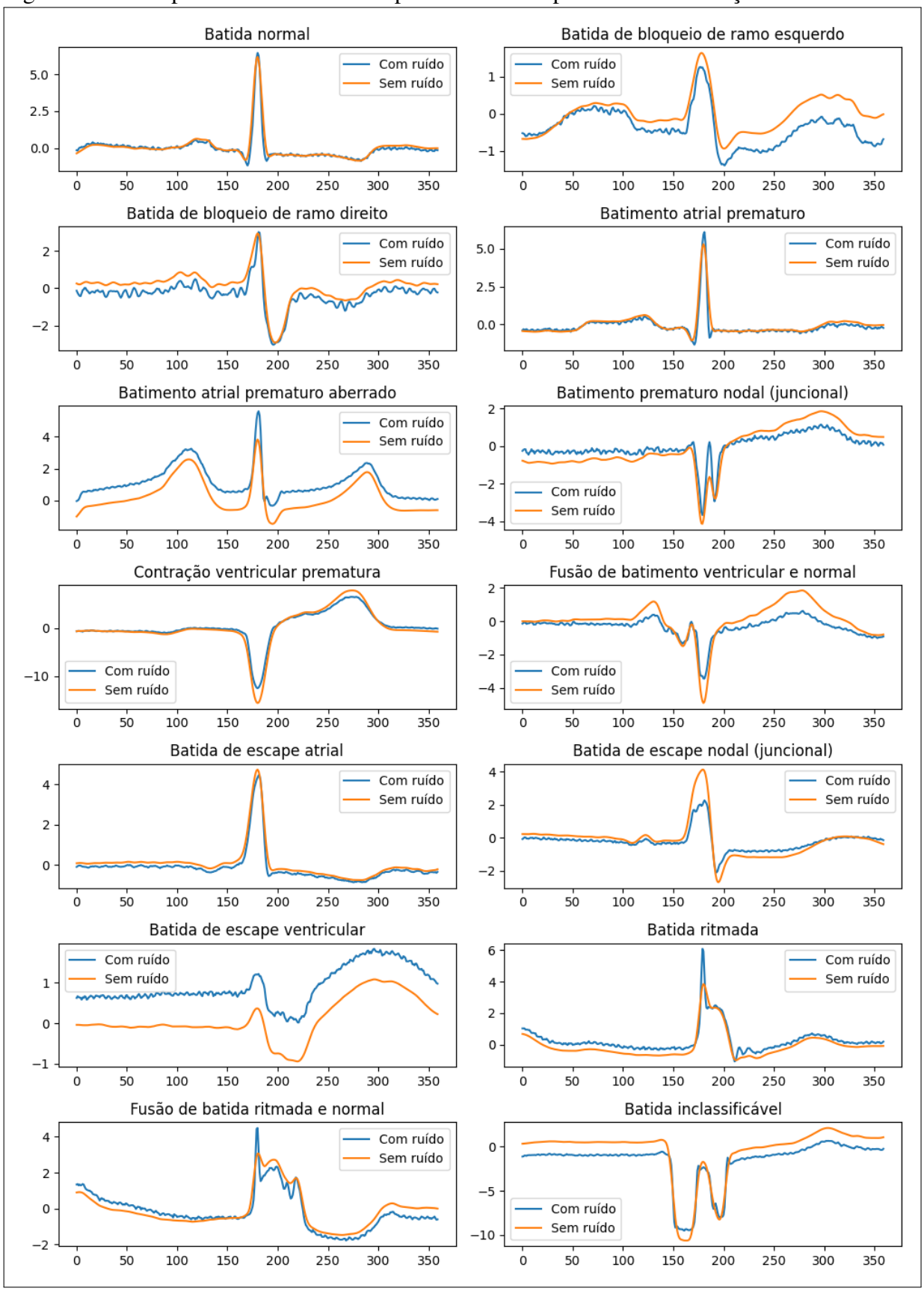
5.2 Comparação entre os modelos

As características deste trabalho que devem ser analisadas durante essa seção são: o tamanho da segmentação, a aplicação de filtro para remoção de ruído, as técnicas de reamostragem e os modelos por algoritmo. Sendo assim, a Seção 5.2.1 realiza análises de desempenhos sobre métricas aplicadas aos melhores modelos desenvolvidos neste trabalho e faz conclusões sobre eles. A Seção 5.2.2 faz um levantamento de informações sobre o comportamento dos modelos por algoritmo, fazendo conclusões em relação às estratégias de pré-processamento utilizadas.

5.2.1 Análise de desempenho

A Tabela 1 refere-se as métricas de desempenho dos modelos sobre os dados com segmentação 2000 que possuem 5 classes. Através dos valores das métricas exibidas é possível inferir que todos os modelos possuem interseção relacionado ao intervalo de confiança sobre a média do F_1-score. Sendo assim, todos os modelos obtiveram bons resultados e são considerados empatados entre si.

Figura 14 – Comportamentos das ondas por classe e comparativo com remoção de ruído



Fonte: Elaborada pela autora.

Tabela 1 – Análise de desempenho dos modelos que lidaram com dados de segmentação 2000 contendo 5 classes: N, L, R, A, V

Estrutura dos dados com segmentação 2000 de 5 classes	Melhores modelos por F1-score	Acurácia	Média da precisão	Média da revocação	Média do F1-score	Intervalo de confiança para F1-score
Com ruído e sem reamostragem	NeuralNetFastAI	0.9922	0,9921	0,9922	0,9922	(0,9911 - 0,9932)
Com ruído e com técnica SMOTE	LightGBM	0,9919	0,9918	0,9919	0,9918	(0,9906 - 0,9929)
Com ruído e com técnica ADASYN	NeuralNetFastAI	0,9919	0,9918	0,9919	0,9918	(0,9906 - 0,9929)
Com filtro e sem reamostragem	LightGBM	0,9943	0,9942	0,9943	0,9942	(0,9932 - 0,9951)
Com filtro e com técnica SMOTE	Floresta aleatória	0,9936	0,9936	0,9936	0,9936	(0,9926 - 0,9945)
Com filtro e com técnica ADASYN	LightGBM	0,9939	0,9939	0,9939	0,9939	(0,9929 - 0,9948)

Fonte: Elaborada pela autora.
Nota: *Os modelos não podem ser considerados um melhor que o outro devido a interseção entre os intervalos de confiança.

Tabela 2 – Análise de desempenho dos modelos que lidaram com dados sobre de estrutura de segmentação 360 contendo 5 classes: N, L, R, A, V

Estrutura dos dados com segmentação 360 de 5 classes	Melhores modelos por F1-score	Acurácia	Média da precisão	Média da revocação	Média do F1-score	Intervalo de confiança para F1-score
Com ruído e sem reamostragem	NeuralNetFastAI	0,9944	0,9943	0,9944	0,9943	(0,9934 - 0,9952)
Com ruído e com técnica SMOTE	NeuralNetTorch	0,9933	0,9933	0,9933	0,9933	(0,9923 - 0,9943)
Com ruído e com técnica ADASYN	NeuralNetFastAI	0,9939	0,9939	0,9939	0,9939	(0,9929 - 0,9949)
Com filtro e sem reamostragem	LightGBM	0,9940	0,9940	0,9940	0,9939	(0,9930 - 0,9949)
Com filtro e com técnica SMOTE	NeuralNetFastAI	0,9940	0,9941	0,9940	0,9940	(0,9931 - 0,9950)
Com filtro e com técnica ADASYN	NeuralNetFastAI	0,9939	0,9940	0,9939	0,9939	(0,9930 - 0,9949)

Fonte: Elaborada pela autora.
Nota: *Os modelos não podem ser considerados um melhor que o outro devido a interseção entre os intervalos de confiança.

O mesmo acontece para as métricas de desempenho da Tabela 2 dos modelos sobre os dados com segmentação 360 que possuem 5 classes, no entanto, sobre essa estrutura há mais semelhanças nos valores dos desempenhos obtidos e o intervalo de confiança é menor, ou seja, os modelos que lidaram com a segmentação 360 dos dados contêm estimativas mais precisas.

A tabela 3 apresenta mais diferenças nos valores das medidas de desempenho, isso decorre por ser mais difícil a classificação para 16 classes e pelo método *TabularPredictor* ter um $time\_limit$ de 2 horas. Ainda assim, apesar de possuir um $time\_limit$ inferior aos dos modelos que classificaram 5 classes, os modelos que lidaram com 16 classes alcançaram altos

Tabela 3 – Análise de desempenho dos modelos que lidaram com dados sobre de estrutura de segmentação 360 contendo 16 classes

Estrutura dos dados com segmentação 360 de 16 classes	Melhores modelos por F1-score	Acurácia	Média da precisão	Média da revocação	Média do F1-score	Intervalo de confiança para F1-score
Com ruído e sem reamostragem	**NeuralNetFastAI**	**0,9884**	**0,9880**	**0,9884**	**0,9881**	**(0,9868 - 0,9893)**
Com ruído e com técnica SMOTE	NeuralNetTorch	0,5777	0,8863	0,5772	0,6696	(0,6640 - 0,6751)
Com ruído e com técnica ADASYN	**NeuralNetFastAI**	**0,9892**	**0,9887**	**0,9892**	**0,9888**	**(0,9875 - 0,9900)**
Com filtro e sem reamostragem	**NeuralNetFastAI**	**0,9895**	**0,9892**	**0,9895**	**0,9891**	**(0,9878 - 0,9903)**
Com filtro e com técnica SMOTE	NeuralNetTorch	0,9768	0,9810	0,9768	0,9783	(0,9765 - 0,9800)
Com filtro e com técnica ADASYN	**NeuralNetFastAI**	**0,9888**	**0,9885**	**0,9888**	**0,9885**	**(0,9872 - 0,9897)**

Fonte: Elaborada pela autora.
Nota: *Os melhores modelos são os que não utilizaram a técnica SMOTE.
Nota: **Não foi possível realizar a classificação de 16 classes com dados segmentados em 2000 devido a falhas na execução.

desempenhos apesar do grande desbalanceamento entre os dados.

Para os modelos da Tabela 3, considerando o intervalo de confiança para o F_1-score, pode-se afirmar que os melhores modelos foram os que não utilizaram a técnica SMOTE. Isso é justificável tendo em vista que o $time\_limit$ teve um tempo inferior aos dos modelos que classificaram 5 classes como mencionado na Seção 4.3, tornando o treino dos modelos que lidaram com a técnica SMOTE mais difícil pela técnica igualar a quantidade de instâncias por classe, gerando um grande volume de dados. Ademais, os melhores modelos para a classificação de 16 classes são todos do algoritmo *NeuralNetFastA*.

Ademais, as Tabelas 4 e 5 são apresentados com o intuito de fazer análises das classificações dos modelos por classes. Vale ressaltar que apesar de possuírem classes em comum, a quantidade de instâncias por classe é diferente pela Tabela 4 retratar o melhor modelo por $F_1 - score$ da Tabela 2 e a Tabela retratar o desempenho do melhor modelo da Tabela 3, sendo assim, os modelos lidam com quantidade de classes diferentes. Apenas esses foram escolhidos a fim de analisar o comportamento dos modelos quando lidam com quantidade de classes diferentes.

Os modelos possuem dificuldades em comum para classificar as classes A e V, mas percebe-se que para o modelo que realiza a classificação de 16 classes os valores de desempenho são menores. Ademais, o modelo representado pela Tabela 5 apresenta mais diferenças sobre os desempenhos, isso decorre pela discrepância na quantidade de instâncias por classes e na maior

Tabela 4 – Métricas de desempenho focado em 5 classes do modelo LightGBM

Classes	Precisão	Revocação	F1-score	Total de instâncias
N	0,9938	0,9991	0,9965	18704
L	0,9985	0,9950	0,9968	2013
R	0,9983	0,9939	0,9961	1809
A	0,9844	0,8975	0,9389	634
V	0,9937	0,9775	0,9855	1779
Média	0,9938	0,9726	0,9828	24939
Média ponderada	0,9943	0,9943	0,9942	24939

Fonte: Elaborada pela autora.

Tabela 5 – Métricas de desempenho focado em 16 classes do modelo NeuralNetFastAI

Classes	Precisão	Revocação	F1-score	Total de instâncias
N	0,9924	0,9969	0,9947	18753
L	0,9970	0,9990	0,9980	2018
R	0,9994	0,9978	0,9986	1814
A	0,9570	0,9089	0,9324	637
a	0,8148	0,5946	0,6875	37
J	0,8696	0,9524	0,9091	21
V	0,9716	0,9781	0,9748	1782
F	0,9212	0,7562	0,8306	201
!	0,8780	0,9153	0,8963	118
e	0,5000	0,5000	0,5000	4
j	0,8980	0,7719	0,8302	57
E	1,0000	1,0000	1,0000	26
/	0,9960	0,9983	0,9972	1756
f	0,9833	0,9633	0,9732	245
x	0,9592	0,9792	0,9691	48
|	0,7143	0,3030	0,4255	33
Média	0,9032	0,8509	0,8698	27550
Média ponderada	0,9892	0,9896	0,9892	27550

Fonte: Elaborada pela autora.

quantidade de arritmias de difícil classificação.

5.2.2 Análise dos modelos por algoritmo

Este trabalho gerou 428 modelos através do AutoML utilizando as estratégias apresentadas na Seção 4.3 e com as abordagens de pré-processamento indicadas na Seção 4.2. Sendo assim, essa seção tem como objetivo extrair informações além dos melhores modelos, entender comportamentos por algoritmo e visualizar as variações dos hiperparâmetros.

Inicialmente, analisando os algoritmos que tiveram melhor desempenho para os modelos que lidaram com classificação de 5 classes: *LightGBM* e *NeuralNetFastAI*. Os 10 melhores alcançaram a média do F_1-score a partir de 0.9939, 9 deles usaram os dados com filtro de remoção de ruído aplicado. Apenas 2 utilizaram as técnicas de reamostragem 1 sendo o SMOTE e o outro sendo o ADASYN. Dos 10, 7 são do algoritmo *LightGBM* e 3 *NeuralNetFastAI*.

O *LightGBM* teve 95 modelos gerados para classificação de 5 classes, em um

ranking considerando o F_1-score, seus 10 primeiros modelos são todos que lidaram com dados que tiveram o filtro de remoção de ruído aplicado, apenas 2 utilizaram a técnica de reamostragem ADASYN, 7 foram segmentados com tamanho 2000.

Já o *NeuralNetFastAI* com 52 modelos, apresentou lidar melhor com dados de segmentação 360, onde de 10 dos seus melhores modelos 9 lidaram com a segmentação de 360. Ademais, não mostrou ter uma correlação direta sobre a remoção de filtro e técnica de reamostragem, já que os melhores modelos se misturaram entre o uso e não uso das técnicas, obtendo nenhuma quantidade de uso relevante.

Sobre os 10 melhores modelos que classificaram 16 classes, todos foram modelos dos algoritmos *NeuralNetFastAI* e *NeuralNetTorch*. Para classificações de 16 classes as redes neurais apresentaram melhor aceitação sobre o pré-processamento dos dados com filtro de remoção de ruído e com a técnica de reamostragem ADASYN. Os modelos *NeuralNetTorch* apresentaram ter mais correlação com o pré-processamento, de forma que, quando o modelo não utilizava um dado que teve o filtro aplicado, ele utilizava dados ruidosos com reamostragem ADASYN.

O Quadro 6 apresenta os melhores modelos por algoritmo. No total foram 428 modelos gerados, sendo eles: 124 do *LightGBM*, 76 do *NeuralNetTorch*, 72 do *NeuralNetFastAI*, 42 do *XGBoost*, 32 do *ExtraTrees*, 32 do RF, 26 do *CatBoost* e 24 do *KNeighbors*. Os melhores modelos resumem os comportamentos dos modelos por estratégia, no sentido de que, para os algoritmos de redes neurais, lidar com dados segmentados em 360 fizeram com que majoritariamente obtivessem melhores classificações. O *LightGBM* e o RF foram os únicos que obtiveram melhores classificações com dados segmentados em 2000. A técnica de reamostragem foi amplamente utilizada pelos melhores modelos de *ExtraTrees*, RF e *XGBoost*. O *CatBoost* obteve melhores resultados para dados segmentados em 360 e com filtro aplicado, não importando o uso e não uso de técnicas de reamostragem. Já o *KNeighbors* só manteve padrões de alto desempenho sobre a segmentação dos dados de tamanho 360.

5.3 Variações de hiperparâmetros obtidas pela ferramenta AutoGluon

Muitos modelos foram gerados por algoritmo através de configurações implementadas no método *TabularPredictor* da ferramenta *Autogluon* que permitiram uma busca maior sobre hiperparâmetros e mais tentativas na busca de melhores modelos, o que retornou uma grande quantidade de modelos. Os Quadros 7, 8, 9, 10, 11, 12 e 13 apresentam os hiperparâmetros utilizados por modelo e suas variações de valores.

Quadro 6 – Melhores modelos por algoritmo

Algoritmo	Tamanho do segmento	Filtro foi aplicado	Técnica de reamostragem	Média F1-score
NeuralNetFastAI	360	Não	None	0.9943
LightGBM	2000	Sim	None	0.9942
ExtraTrees	360	Sim	Smote	0.9937
Floresta aleatória	2000	Sim	Smote	0.9936
NeuralNetTorch	360	Não	None	0.9935
XGBoost	360	Sim	Smote	0.9933
CatBoost	360	Sim	None	0.9914
KNeighbors	360	Sim	None	0.9910

Fonte: Elaborado pela autora.

Quadro 7 – Variações dos hiperparâmetros do NeuralNetFastAI

NeuralNetFastAI		
Hiperparâmetro	**Variações**	**Quantidade de variações**
layers	[(500, 200), (500,), None, (200, 100), (50, 25), (200, 100, 50), (1000, 500, 200), (200,), (1000,), (500, 200, 100), (1000, 500)]	11
smoothing	[0.0]	1
epochs	Mínimo=5; Máximo=30	26
ps (linear layers dropout)	Mínimo=0.0198; Máximo=0.4916	42
early.stopping.patience	[20]	1
bs (batch size)	[256, 2048, 64, 512, 1024, 4096]	6
lr (maximum learning rate)	Mínimo=5.3878e-05; Máximo=0.0952	42
early.stopping.min_delta	[0.0001]	1
emb_drop	Mínimo=0.01624; Máximo=0.4914)	42

Fonte: Elaborado pela autora.

Os modelos que mais obtiveram valores variados em seus hiperparâmetros foram os de redes neurais artificiais. Neste trabalho as redes neurais artificiais obtiveram os melhores desempenhos aparacendo nos quadros de melhores modelos de cada estratégia, além disso, elas apresentaram lidar melhor com os dados originários com segmentação 360.

Outro algoritmo que obteve um ótimo desempenho foi o *LightGBM* de AM. Apesar de não ter tido muitas variações em seus hiperparâmetros o algoritmo apresentou um alto poder preditivo para os dados com segmentação 2000 que tiveram o filtro de remoção de ruído aplicado.

Os demais algoritmos não obtiveram grandes variações em seus hiperparâmetros, mas ainda assim atingiram classificações consideráveis. O *KNeighbors* por exemplo só obteve duas variações para o hiperparâmetro *weights*. Tendo isso em vista, com os dados apresentados e análises obtidas, conclui-se para o *AutoGluon* que com o conjunto de dados utilizado nesse trabalho ele não gerou grandes quantidades de hiperparâmetros para modelos de AM, mas gerou para AP. Sendo assim, sobre isso o que fica como questionamento é: a decisão da ferramenta de priorizar modelos de AP foi por obterem melhores classificações para dados de ECG.

Quadro 8 – Variações dos hiperparâmetros do LightGBM

LightGBM		
Hiperparâmetro	**Variações**	**Quantidade de variações**
num_leaves	[128, 74, 83, 52, 53, 31]	6
extra_trees	[True, None]	2
min_data_in_leaf	[3, 5, 14, 48, 20, 52]	6
learning_rate	Mínimo=0.0061; Máximo=0.1000)	6
feature_fraction	Mínimo=0.8243; Máximo=1.0)	6

Fonte: Elaborado pela autora.

Quadro 9 – Variações dos hiperparâmetro do ExtraTrees

ExtraTrees		
Hiperparâmetro	**Variações**	**Quantidade de variações**
max_leaf_nodes	[15000]	1
n_jobs	[-1]	1
criterion	['gini', 'entropy']	2
bootstrap	[True]	1
random_state	[0]	1
n_estimators	[300]	1

Fonte: Elaborado pela autora.

Quadro 10 – Variações dos hiperparâmetro do NeuralNetTorch

NeuralNetTorch		
Hiperparâmetro	**Variações**	**Quantidade de variações**
proc.impute_strategy	['most_frequent', 'mean', 'median']	3
use_ngram_features	[False]	1
embedding_size_factor	[0.5, 1.3, 0.7, 1.1, 1.0, 0.6, 1.2, 1.4, 0.9, 1.5, 0.8]	11
hidden_size	[128, 512, 256]	3
y_range_extend	[0.05]	1
use_batchnorm	[False, True]	2
num_layers	[2, 3, 4]	3
proc.embed_min_categories	[3, 100, 4, 1000, 10]	5
y_range	[None]	1
proc.skew_threshold	[0.9, 0.8, 0.99, 0.999, 1.0, 0.5, 0.2, 0.3, 100.0, 10.0]	10
activation	['relu', 'elu']	2
embed_exponent	[0.56]	1
loss_function	['auto']	1
epochs_wo_improve	[20]	1
num_epochs	[500]	1
learning_rate	Mínimo=0.0001; Máximo=0.0208;	43
proc.max_category_levels	[100, 1000, 200, 10, 300, 10000, 400, 20, 500]	9
max_embedding_dim	[100]	1
optimizer	['adam']	1
max_batch_size	[512]	1
weight_decay	Mínimo=1.3209e-12; Máximo=0.09426;	43
dropout_prob	[0.4, 0.5, 0.1, 0.2, 0.3, 0.0]	6

Fonte: Elaborado pela autora.

Quadro 11 – Variações dos hiperparâmetros do Floresta aleatória (RF)

Floresta aleatória		
Hiperparâmetro	**Variações**	**Quantidade de variações**
max_leaf_nodes	[15000]	1
n_jobs	[-1]	1
criterion	['gini', 'entropy']	2
bootstrap	[True]	1
random_state	[0]	1
n_estimators	[300]	2

Fonte: Elaborado pela autora.

Quadro 12 – Variações dos hiperparâmetros do XGBoost

XGBoost		
Hiperparâmetro	**Variações**	**Quantidade de variações**
n_jobs	[-1]	1
objective	['multi:softmax']	1
learning_rate	Mínimo=0.0238; Máximo=0.1341;	5
proc.max_category_levels	[100]	1
num_class	[5]	1
colsample_bytree	Mínimo=0.6917; Máximo=1.0;	5
min_child_weight	[1, 3, 4, 5]	4
max_depth	[8, 10, 3, 6]	4
booster	['gbtree']	1
n_estimators	[10000]	1

Fonte: Elaborado pela autora.

Quadro 13 – Variações dos hiperparâmetros do *CatBoost*

CatBoost		
Hiperparâmetro	**Variações**	**Quantidade de variações**
eval_metric	['Accuracy']	1
l2_leaf_reg	[3.3713784729000733, 3, 3.1795327319875875]	3
iterations	[10000]	1
random_seed	[0]	1
allow_writing_files	[False]	1
depth	[8, 5, 6]	3
learning_rate	[0.02386109712430462, 0.11259896076097302, 0.05]	3

Fonte: Elaborado pela autora.

6 CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho teve como intuito propor uma solução para classificação de arritmia cardíaca utilizando aprendizado de máquina automatizado com abordagens que visam um melhor uso dos hiperparâmetros dos algoritmos e um melhor uso dos dados de ECG. Isso foi possível por meio da ferramenta AutoGluon com o método de predição tabular para dados tabulares e séries temporais. Por meio da configuração do método de predição tabular, 428 modelos puderam ser gerados utilizando estratégias de pré-processamento diferentes sobre os dados do ECG.

Em vista disso, os modelos lidaram com combinações das seguintes estratégias: dados com segmentação de 360 e 2000; dados com ruído e com filtro de remoção de ruído; dados sem técnica de reamostragem, com técnica de reamostragem SMOTE e com técnica de reamostragem ADASYN. Sendo assim, foram feitas classificações para 5 classes com dados segmentados em 360 e 2000 e classificações para 16 classes com dados segmentados em 360.

Por meio do uso das estratégias de pré-processamento mencionadas e da geração de modelos pela ferramenta *AutoGluon* que faz uma busca por melhores hiperparâmetros de forma automatizada; o presente trabalho obteve resultados tão eficientes quanto os dos trabalhos relacionados. Sendo assim, os modelos de maior percentual sobre o $F_1 - score$ foram do algoritmo de rede neural da biblioteca *FASTAI* com 99,43% para classificação de 5 classes e 98,91% para classificação de 16 classes. Os códigos utilizados estão disponíveis em um repositório público no GitHub[1].

Ademais, apesar da eficiência obtida, as dificuldades encontradas no decorrer deste trabalho foram quanto ao desempenho do servidor utilizado que em todas as abordagens houve execuções lentas e para uma estrutura de classificação com segmentação de dados de 2000 que utiliza 16 classes não houve sucesso tendo várias quedas no servidor. Além disso, devido essas limitações, não foi possível gerar uma quantidade suficiente de modelos que classificaram 16 classes e tiveram dados com a técnica SMOTE aplicada, por consequência não foi possível fazer uma análise mais a fundo sobre o impacto da técnica sobre a quantidade de classes desbalanceadas. Outro ponto a considerar, foi o uso dos batimentos normais que não permitiu uma análise mais aprofundada sobre revocação e precisão das classes de arritmia devido ao grande desbalanceamento quando comparado a dados de batimentos normais.

Mas ainda assim, foi possível desenvolver os modelos e realizar as experimentações

[1] https://github.com/PaulaLuana2/arrhythmia-classification

nas limitações apresentadas. Sobre todos os modelos gerados pode-se inferir que os pertencentes à área de aprendizado de máquina obtiveram melhores resultados com dados tendo sido aplicado filtro para remoção de ruído. Já sobre os tamanhos das segmentações e uso das estratégias de reamostragem, resultaram em diferentes comportamentos. Para o aprendizado profundo, os modelos apresentaram lidar melhor com os dados originários quando classificaram dados segmentados em 360 que tinham 5 classes, mas para classificação de 16 classes tiveram melhores desempenhos quando tinham dados em que foram aplicadas técnicas de pré-processamento. Além disso, modelos de aprendizado profundo tiveram maiores variações de hiperparâmetros quando comparados a modelos de aprendizado de máquina.

À vista disso, pretende-se para trabalhos futuros utilizar um servidor com maior poder computacional, permitindo o uso de outros conjuntos de dados maiores relacionados ao ECG recorrendo apenas a classes de arritmias; fazer um estudo específico para modelos de aprendizado profundo, a fim de analisar estratégias de uso de dados de ECG voltados para redes neurais e analisar o uso das estruturas de redes neurais para a ferramenta *AutoGluon*.

REFERÊNCIAS

ABDI, H. Z-scores. **Encyclopedia of measurement and statistics**, Sage Thousand Oaks, CA, v. 3, p. 1055–1058, 2007.

ALARSAN, F. I.; YOUNES, M. Analysis and classification of heart diseases using heartbeat features and machine learning algorithms. **Journal of big data**, Springer, v. 6, n. 1, p. 1–15, 2019.

AL'AREF, S. J.; ANCHOUCHE, K.; SINGH, G.; SLOMKA, P. J.; KOLLI, K. K.; KUMAR, A.; PANDEY, M.; MALIAKAL, G.; ROSENDAEL, A. R. V.; BEECY, A. N. *et al.* Clinical applications of machine learning in cardiovascular disease and its relevance to cardiac imaging. **European heart journal**, Oxford University Press, v. 40, n. 24, p. 1975–1986, 2019.

AUTOGLUON: Automl for text, image, and tabular data. Disponível em: https://auto.gluon.ai/0.3.1/index.html. Acesso em: 11 de nov. 2023.

BRANT, L. C. C.; NASCIMENTO, B. R.; TEIXEIRA, R. A.; LOPES, M. A. C. Q.; MALTA, D. C.; OLIVEIRA, G. M. M.; RIBEIRO, A. L. P. Excess of cardiovascular deaths during the covid-19 pandemic in brazilian capital cities. **Heart**, BMJ Publishing Group Ltd and British Cardiovascular Society, v. 106, n. 24, p. 1898–1905, 2020.

CHANG, Y.-C.; CHANG, K.-H.; WU, G.-J. Application of extreme gradient boosting trees in the construction of credit risk assessment models for financial institutions. **Applied Soft Computing**, Elsevier, v. 73, p. 914–920, 2018.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. Smote: synthetic minority over-sampling technique. **Journal of artificial intelligence research**, v. 16, p. 321–357, 2002.

CHOLLET, F. **Deep learning with Python**. [S.l.]: Simon and Schuster, 2021.

COMPARING randomized search and grid search for hyperparameter estimation. Disponível em: https://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html. Acesso em: 11 de nov. 2023.

DHAR, V. Data science and forecasting. **ACM Communication**, ACM New York, NY, USA, v. 56, n. 12, p. 64–73, 2013.

ERICKSON, N.; MUELLER, J.; SHIRKOV, A.; ZHANG, H.; LARROY, P.; LI, M.; SMOLA, A. Autogluon-tabular: Robust and accurate automl for structured data. **arXiv preprint arXiv:2003.06505**, 2020.

FELDMAN, J.; GOLDWASSER, G. P. Eletrocardiograma: recomendações para a sua interpretação. **Revista da SOCERJ**, v. 17, n. 4, p. 251–256, 2004.

FERRERO, C. A. **Algoritmo kNN para previsão de dados temporais**: funções de previsão e critérios de seleção de vizinhos próximos aplicados a variáveis ambientais em limnologia. Tese (Doutorado) — Universidade de São Paulo, 2009.

FREIRE, A. L.; JR, J. M. P. de M.; BARRETO, G. A. Redes neurais recorrentes para predição recursiva de séries temporais caóticas: um estudo comparativo. **IX CBRN**, p. 25–28, 2009.

GARCÍA, S.; LUENGO, J.; HERRERA, F. **Data preprocessing in data mining**. [S.l.]: Springer, 2015. v. 72.

GÉRON, A. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow**: Concepts, tools, and techniques to build intelligent systems. [S.l.]: "O'Reilly Media, Inc.", 2019.

GRUS, J. **Data science do zero**. [S.l.: s.n.], 2016.

HE, H.; BAI, Y.; GARCIA, E. A.; LI, S. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: IEEE. **2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)**. [S.l.], 2008. p. 1322–1328.

HE, X.; ZHAO, K.; CHU, X. Automl: A survey of the state-of-the-art. **Knowledge-Based Systems**, Elsevier, v. 212, p. 106622, 2021.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J. **Automated machine learning**: methods, systems, challenges. [S.l.]: Springer Nature, 2019.

International Data Corporation. **Global Spending on Big Data and Analytics Solutions Will Reach $215.7 Billion in 2021, According to a New IDC Spending Guide**. 2021. Disponível em: https://www.idc.com/getdoc.jsp?containerId=prUS48165721. Acesso em: 21 de jul. 2023.

JURASZEK, G. D. *et al.* **Reconhecimento de produtos por imagem utilizando palavras visuais e redes neurais convolucionais**. Universidade do Estado de Santa Catarina, 2014.

KACHUEE, M.; FAZELI, S.; SARRAFZADEH, M. Ecg heartbeat classification: A deep transferable representation. In: IEEE. **2018 IEEE international conference on healthcare informatics (ICHI)**. [S.l.], 2018. p. 443–444.

KRANZUSCH, R.; SIEPEN, F. A. D.; WIESEMANN, S.; ZANGE, L.; JEUTHE, S.; SILVA, T. Ferreira da; KUEHNE, T.; PIESKE, B.; TILLMANNS, C.; FRIEDRICH, M. G. *et al.* Z-score mapping for standardized analysis and reporting of cardiovascular magnetic resonance modified look-locker inversion recovery (molli) t1 data: Normal behavior and validation in patients with amyloidosis. **Journal of Cardiovascular Magnetic Resonance**, Springer, v. 22, p. 1–10, 2020.

LECUN, Y.; KAVUKCUOGLU, K.; FARABET, C. Convolutional networks and applications in vision. In: IEEE. **Proceedings of 2010 IEEE international symposium on circuits and systems**. [S.l.], 2010. p. 253–256.

MAKOWSKI, D.; PHAM, T.; LAU, Z. J.; BRAMMER, J. C.; LESPINASSE, F.; PHAM, H.; SCHöLZEL, C.; CHEN, S. H. A. NeuroKit2: A python toolbox for neurophysiological signal processing. **Behavior Research Methods**, Springer Science and Business Media LLC, v. 53, n. 4, p. 1689–1696, feb 2021.

MARINHO, A. S. **Classificação automática da qualidade de sinal de ECG com redes neurais convolucionais**. Crateús: [s.n.], 2021.

MELLO, T. R. d. **Comparativo entre redes neurais recorrentes GRU e LSTM para a predição de instrumentos financeiros**. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2021.

MELO, E. S. **Cuide do Seu Coração**: Estratégias para redução do risco cardiovascular em pessoas que vivem com hiv. [S.l.]: Bibliomundi, 2022.

MIT-BIH Arrhythmia Database. Disponível em: https://www.physionet.org/content/mitdb/1.0.0/. Acesso em: 19 de nov. 2023.

MOODY, G. B.; MARK, R. G. The impact of the mit-bih arrhythmia database. **IEEE Engineering in Medicine and Biology Magazine**, IEEE, v. 20, n. 3, p. 45–50, 2001.

Organização Mundial da Saúde. **Cardiovascular diseases (CVDs)**. 2021. Https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds). Acesso em: 21 de jul. 2023.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

REZENDE, S. O. **Sistemas inteligentes**: fundamentos e aplicações. [S.l.]: Editora Manole Ltda, 2003.

RUSSELL, S. J. **Artificial intelligence a modern approach**. [S.l.]: Pearson Education, Inc., 2010.

SANTOS, P. H. M. d. S. C. d. **Aplicação de aprendizado de máquina em exames de ECG-Holter para a detecção de dispositivos cardíacos implantáveis e arritmias.** Universidade Federal de São Paulo, 2023.

SILVA, I. R.; SOUZA, R. G. de; SILVA, G.; OLIVEIRA, C. S. de; CAVALCANTI, L. H.; BEZERRA, R. S.; FAGUNDES, R. A. d. A.; SANTOS, W. P. dos. Utilização de redes convolucionais para classificação e diagnóstico da doença de alzheimer. In: SIMPÓSIO DE INOVAÇÃO EM ENGENHARIA BIOMÉDICA, 2. **Anais [...]**. [S.l.], 2018. p. 73–76.

SINGH, S.; PANDEY, S. K.; PAWAR, U.; JANGHEL, R. R. Classification of ecg arrhythmia using recurrent neural networks. **Procedia computer science**, Elsevier, v. 132, p. 1290–1297, 2018.

ŚMIGIEL, S.; PAŁCZYŃSKI, K.; LEDZIŃSKI, D. Ecg signal classification using deep learning techniques based on the ptb-xl dataset. **Entropy**, MDPI, v. 23, n. 9, p. 1121, 2021.

Sociedade Brasileira de Cardiologia. **Aumenta o número de mortes por doenças cardiovasculares no primeiro semestre de 2021**. 2021. Disponível em: https://www.portal.cardiol.br/post/aumenta-o-número-de-mortes-por-doenças-cardiovasculares-no-primeiro-semestre-de-2021. Acesso em: 21 de jul. 2023.

STEVENS, B.; PEZZULLO, L.; VERDIAN, L.; TOMLINSON, J.; GEORGE, A.; BACAL, F. Os custos das doenças cardíacas no brasil. **Arquivos Brasileiros de Cardiologia**, SciELO Brasil, v. 111, p. 29–36, 2018.

WU, M.; LU, Y.; YANG, W.; WONG, S. Y. A study on arrhythmia via ecg signal classification using the convolutional neural network. **Frontiers in computational neuroscience**, Frontiers Media SA, v. 14, p. 564015, 2021.

YANG, L.; SHAMI, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. **Neurocomputing**, Elsevier, v. 415, p. 295–316, 2020.