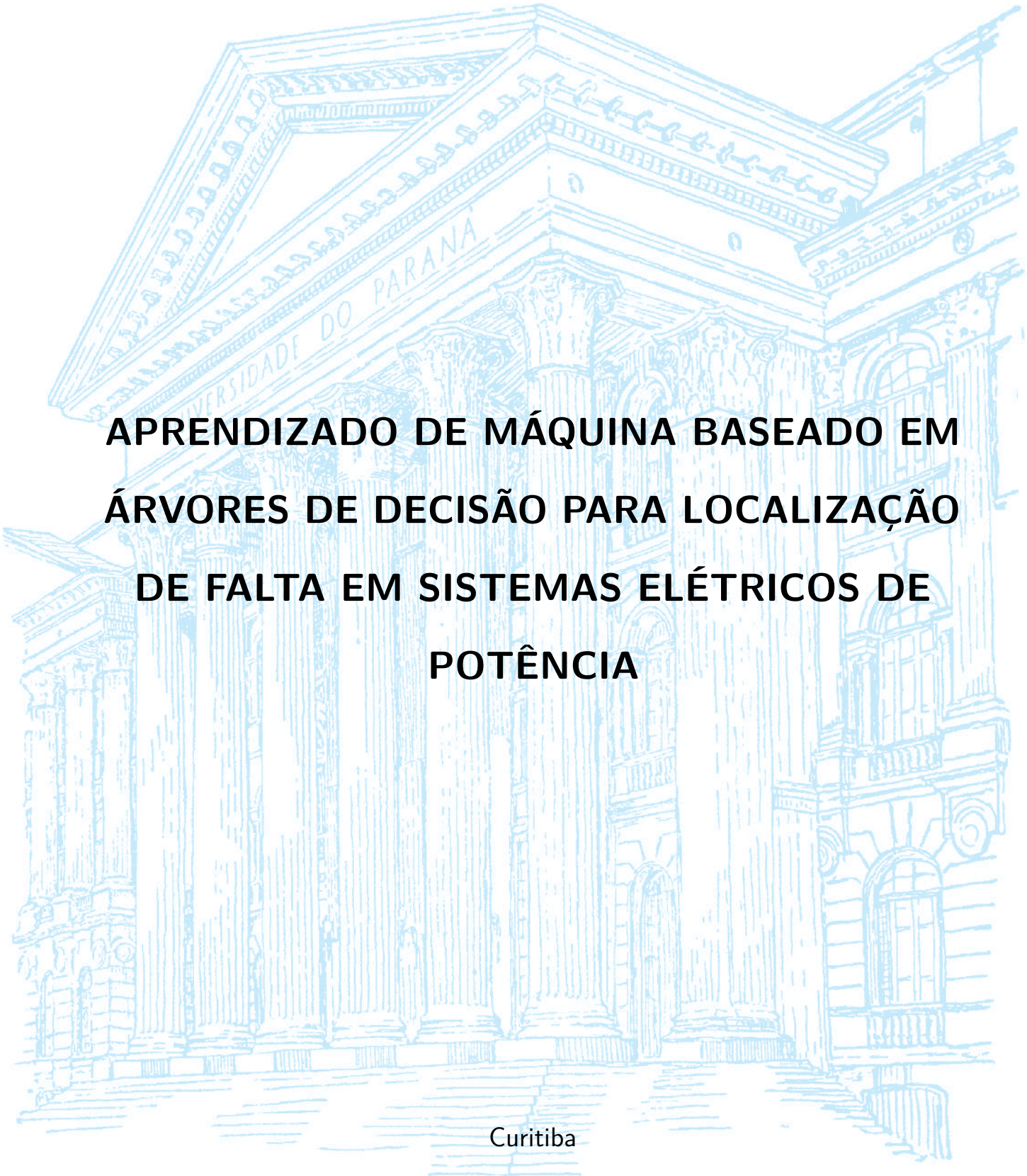
BRUNO GUENTHER



APRENDIZADO DE MÁQUINA BASEADO EM ÁRVORES DE DECISÃO PARA LOCALIZAÇÃO DE FALTA EM SISTEMAS ELÉTRICOS DE POTÊNCIA

Curitiba

2024

BRUNO GUENTHER

APRENDIZADO DE MÁQUINA BASEADO EM ÁRVORES DE DECISÃO PARA LOCALIZAÇÃO DE FALTA EM SISTEMAS ELÉTRICOS DE POTÊNCIA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica da Universidade Federal do Paraná, como requisito parcial à obtenção do bacharelado em Engenharia Elétrica com ênfase em Eletrotécnica.

Orientador: Prof. Dr. Ricardo Schumacher

Curitiba

2024

Agradecimentos

Gostaria de registrar aqui meus sinceros e calorosos agradecimentos para aqueles que me ensinaram sobre a vida e permitiram eu me tornar a pessoa que sou com as capacidades que tenho.

Aos meus pais Gerson e Solange e ao meu irmão Gabriel por serem a minha base, por terem me proporcionado o privilégio do acesso a educação de qualidade e por me ensinarem o que é amor desde criança.

A todos aqueles e aquelas que estiveram presentes em minha vida, dividiram momentos de alegrias e de tristezas e me mostraram que o melhor da vida é compartilhar o que ela te proporciona.

Um agradecimento a tudo que tive o prazer de aprender com o esporte. Esse é uma ferramenta de educação e foi através dele que aprendi que os maiores objetivos da vida só se atingirão ao colocar esforço, consistência e o coração em sua busca.

Por fim, um agradecimento ao meu professor orientador Dr. Ricardo Schumacher e a todos que estiveram presentes em minha jornada neste ano durante a construção deste trabalho, que me guiaram e apoiaram em construir não só um trabalho de conclusão de curso, mas também uma entrega que eu me orgulhe de ter feito.

Obrigado.

"A vontade de se preparar precisa ser maior que a vontade de vencer"

Bob Knight - Hall da Fama do Basquete

Resumo

A capacidade de medir, transferir e armazenar sincrofasores da rede elétrica por meio de PMUs (Unidades de Medição Fasorial) viabiliza uma ampla gama de estudos sobre o comportamento dinâmico dos sistemas de potência. A obtenção de dados sincronizados por GPS, com taxas de amostragem de até uma vez por ciclo, proporciona um nível de visibilidade anteriormente restrito a dispositivos especializados, cuja análise era limitada a janelas de tempo curtas e específicas. Essa nova abordagem amplia significativamente as possibilidades de monitoramento e análise em tempo real, contribuindo para uma maior compreensão e gerenciamento do sistema elétrico. Este trabalho tem como objetivo aplicar a dados simulados referentes ao Sistema Benchmark IEEE-39 Bus System, um modelo de aprendizagem de máquina que a partir de uma rotulação prévia, aprende o comportamento do sistema considerando faltas em diferentes barras e linhas e consequentemente seja capaz de prever, se a falta ocorre no território da concessionária A ou no território da concessionária B. Por conseguinte, o trabalho apresenta não só uma análise em cima das métricas de validação de modelos de aprendizagem de máquina de classificação, como também visa elucidar formas de utilizar técnicas de aprendizagem de máquina para se extrair informações valiosas do comportamento do SEP. Decorrente da vasta literatura e do amplo campo de pesquisa, este trabalho foca na utilização de modelos baseados em Árvores de Decisão, e estende a análise para modelos *Ensemble* que agregam diferentes Árvores visando um modelo mais robusto e de melhor resultado. Os resultados apresentados em 5 mostram que o desempenho do modelo foi suficiente em diferenciar o local de falta entre área interna e externa, alcançando uma acurácia de 90%. Além disso, foi demonstrado como otimização de hiperparâmetros e etapas de pós-processamento podem aumentar este desempenho para 92% e 97% respectivamente. Por fim, são demonstrado resultados que descrevem o SEP a partir da análise do modelo, validando por exemplo, o aumento deste desempenho a partir do número de PMU's fornecidas ao modelo, diferença de performance do modelo entre as faltas, entre outros resultados.

Palavras-Chave: Sistemas Elétricos de Potência. Localização de Falta. Aprendizagem de

Máquina. Árvores de Decisão. *Gradient Boosting*. Unidades de Medição Fasorial.

Abstract

The ability to measure, transfer, and store synchrophasor data from the electrical grid through PMUs (Phasor Measurement Units) enables a wide range of studies on the dynamic behavior of power systems. The acquisition of GPS-synchronized data, with sampling rates of up to once per cycle, provides a level of visibility previously restricted to specialized devices, whose analysis was limited to short and specific time windows. This approach significantly broadens the possibilities for real-time monitoring and analysis, contributing to a better understanding and management of the electrical system. This study aims to apply machine learning models to simulated data from the IEEE-39 Bus Benchmark System. Through prior labeling, the model learns the system's behavior by considering faults in different buses and lines, and is subsequently capable of predicting whether a fault occurs within the territory of concessionaire A or concessionaire B.

Therefore, this work not only presents an analysis of the validation metrics of machine learning classification models but also aims to highlight how machine learning techniques can be used to extract valuable insights into the behavior of the power system (SEP). Due to the vast literature and broad field of research, this study focuses on the use of decision tree-based models and extends the analysis to ensemble methods that aggregate different trees, aiming for a more robust model with better performance. The results presented in **??** show that the model's performance was sufficient to differentiate the fault location between internal and external areas, achieving an accuracy of 90%. Furthermore, it was demonstrated how hyperparameter optimization and post-processing steps can increase this performance to 92% and 97%, respectively. Finally, results are presented that describe the SEP based on the model analysis, validating, for instance, the improvement in performance with the increase in the number of PMUs provided to the model, the difference in model performance across faults, among other findings.

Keywords: Power Systems. Fault Location. Machine Learning. Decision Trees. Gradient Boosting. Phasor Measurement Units.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PMU	**P**hasor **M**easurement **U**nits
GPS	**G**lobal **P**ositioning **S**ystem
ELE	**E**stimação de **L**ocal de **E**vento
IEEE	**I**nstitute of **E**lectrical and **E**letronic **E**ngineers
AM	**A**mplitude **M**odulation
GOES	**G**eo**E**stationary **E**nviromental **S**atellite
EVT	**E**rro **V**etorial **T**otal
PDC	**P**hasor **D**ata **C**oncentrator
UTC	**C**oordinated **U**niversal **T**ime
SCADA	**S**upervisionary **C**ontrol and **D**ata **A**cquisition
EMS	**E**nergy **M**anagement **S**ystem
CRC	**C**yclic **R**edundancy **C**heck
SOC	**S**econd **O**f **C**entury
CE	**C**ross **E**ntropy
CART	**C**lassification and **R**egression **T**rees
XG	E**X**treme **G**radient
EFB	**E**xclusive **F**eature **B**undling
SMBO	**S**equential **M**odel **B**ased **O**ptimization
ROC	**R**eceiver **O**perating **C**haracteristic

Sumário

1 Introdução

Com o avanço tecnológico das últimas décadas, e com a digitalização estando cada vez mais presente na rotina da sociedade, a produção massiva de um volume de dados tornou-se algo comum. Como exemplo, estima-se que todo dia são produzidos 2.5 Quintilhões de bytes em dados nas redes sociais (SG Analytics, 2020) por isso o monitoramento e a análise de dados torna-se cada vez mais necessário.

Para isso, técnicas e tecnologias foram se desenvolvendo e sendo aperfeiçoadas cada vez mais, sendo uma entre elas, o Aprendizado de Máquina. De acordo com Arthur Samuel, o Aprendizado de Máquina é o campo de estudo que possibilita aos computadores a habilidade de aprender sem explicitamente serem programados. (GÉRON, 2019a)

No âmbito da Ciência de Dados, entende-se que não existe um algoritmo de aprendizado de máquina (ou modelo gerado pelo mesmo) capaz de superar todos os demais em todas as aplicações possíveis (STERKENBURG; GRUNWALD, 2022) dessa forma, considera-se como melhor, aquele que possui melhor desempenho empírico dada uma métrica de erro, que pode variar de problema para problema. Esse conceito ficou conhecido como “Teorema do Não Almoço Grátis”.

Dentre os diversos tipos de algoritmos de aprendizado de máquina disponíveis na literatura, aqueles baseados em árvores de decisão têm recebido atenção significativa por parte de diversos pesquisadores/desenvolvedores. Os principais motivos para isso são sua fácil interpretabilidade e sua capacidade de se ajustar a dados complexos. Além disso, Árvores de Decisão são constituintes dos chamados Métodos Ensemble, métodos que integram os algoritmos mais poderosos disponíveis atualmente (GÉRON, 2019b), e que serão objeto de estudo deste trabalho.

Nas engenharias, o aprendizado de máquina vem se tornando referência e material de estudo, decorrente principalmente do avanço da Indústria 4.0 e seus paradigmas como o uso de sensores e máquinas que coletem, armazenem e monitorem dados da sua produção.

Deste modo, o uso de técnicas que ajudem a interpretar os dados para discernir padrões complexos oferece um caminho de tomada de decisão inteligente na indústria (RAI *et al.*,).

Em Sistemas Elétricos de Potência, uma das estratégias para o monitoramento inteligente das condições de operação da rede é o uso das Unidades de Medição Fasorial (ou PMUs, do inglês, *Phasor Measurement Units*), que são instaladas em barras de um sistema e dessa forma fornecem medições sincronizadas via GPS dos fasores de corrente e tensão. Ainda que alguns desafios referente ao uso do equipamento precisem ser superados como mau funcionamento, comunicação do equipamento e seu custo (KIM; WHITE; SHIN, 2018), técnicas de processamento e análise de dados capazes de tratar e operar sob tais condições constituem, por exemplo, uma poderosa ferramenta para solução de problemas de Estimação de Local de Evento (ELE), atrelado a faltas elétricas na rede.

1.1 Objetivos

1.1.1 Objetivo Geral

O Objetivo geral deste trabalho consiste em desenvolver e avaliar a performance de um algoritmo de Aprendizado de Máquina baseado em árvores de decisão. Trata-se de um problema de classificação que visa diagnosticar o local de faltas elétricas em simulações feitas para o Sistema IEEE New England de 39 barras, através de um conjunto de dados extraídos por PMU's.

1.1.2 Objetivos Específicos

Os objetivos específicos deste trabalho incluem:

- Coletar, pré-processar e trazer uma visualização sobre os dados coletados a partir das Unidades de Medição Fasorial.
- Modelar através de algoritmos baseados em árvores o problema voltado a diagnosticar o local da falha entre a área A ou B do SEP.

- Comparar a performance de técnicas ensemble como *Bagging* ou *Boosting* com modelos menos robustos baseados em árvores de decisão.
- Desenvolver uma análise aprofundada em cima das métricas de classificação para entender o desempenho do modelo frente a problemática proposta.
- Estabelecer uma comparação direta entre o uso de Árvores de Decisão e Redes Neurais frente a problemática proposta.

1.2 Estrutura

Este trabalho está dividido em seis capítulos, o primeiro contém a introdução ao tema, bem como os objetivos e a estrutura. O segundo capítulo traz a fundamentação teórica, são apresentados dados que embasam o comportamento de faltas no SEP, bem como apresenta-se um dos principais objetos de estudo deste trabalho, as PMU's. Em seguida, aprofunda-se em conceitos sobre aprendizagem de máquina, sendo eles: Árvores de decisão e métodos Ensemble, otimização de hiperparâmetros e métricas de avaliação de desempenho de modelos.

O capítulo 3 faz uma revisão do artigo que motivou este trabalho, onde o mesmo problema é abordado com os mesmos dados, a partir de uma solução utilizando redes neurais.

O capítulo 4 apresenta materiais e métodos utilizados para a elaboração dos resultados atingidos no capítulo 5, sendo esse dividido em 9 resultados comentados separadamente.

O Capítulo 6 traz a conclusão e comenta-se sobre as recomendações do autor para seguir no campo de estudo do tópico apresentado neste trabalho, visando principalmente aproximar a solução de uma aplicação real e factível.

O Apêndice apresenta a teoria por trás da variação de tensão nos fasores da rede trifásica em momentos de falta, fenômeno que embasa este trabalho.

2 Fundamentação Teórica

2.1 Visão Geral de Faltas no Sistema Elétrico de Potência

Em Sistemas de Potência de grande extensão é extremamente comum a ocorrência de faltas, a maioria das vezes causadas por alguma interferência externa ao sistema, como um efeito natural de grande porte. Quando da ocorrência de curtos-circuitos, por exemplo, o sistema sai do ponto de operação inicial e tende a se acomodar em outro ponto de operação estável. Tais curtos podem também ser temporários, em que após um período de tempo, o curto se encerre por causas naturais e o sistema volte a operar no mesmo ponto de operação anterior ao momento da falta. Outro possível comportamento é a oscilação crescente ao longo do tempo (indicativo de instabilidade) dos fasores da rede. Essas súbitas mudanças de ponto de operação ocasiona um efeito de transitório eletromecânico[1] (CEPEL, 2024), que apresenta diferentes comportamentos dependendo do tipo do curto-circuito.

De fato, a obra "Curto-Circuito"(KINDERMANN, 1997) classifica os problemas de falta em 5 principais, sendo eles:

Problemas de Isolação: As tensões nos condutores do sistema são elevadas, consequentemente, rupturas para a terra ou entre cabos poderá ocorrer devido a desenho inadequado da isolação de equipamentos ou material de má qualidade.

Problemas Mecânicos: São os oriundos da natureza e que provocam ação mecânica no sistema elétrico, deslocando linhas de transmissão e estabelecendo um curto entre elas.

Problemas Elétricos: São os problemas elétricos intrínsecos da natureza ou os devidos à operação do sistema. Aqui os melhores exemplos são descargas atmosféricas e sobretensões no sistema.

[1] Esse transitório eletromecânico resulta, principalmente, da interação eletromecânica existente entre os geradores síncronos conectados ao sistema, quando estes são submetidos a uma perturbação passiva de afetar sua sincronia.

Problemas de Natureza Térmica: O aquecimento nos cabos e equipamentos do sistema, ocasionado por sobrecorrentes em consequência da sobrecarga no sistema.

Problemas de Manutenção: Problemas relacionados à falta de qualidade do material ou da falta de qualificação técnica adequada da mão de obra utilizada para realizar a manutenção dos equipamentos.

Atrelado aos problemas acima, a Tabela 1 mostra a distribuição de ocorrências de Curto Circuitos de acordo com cada um dos setores (partes) do sistema, quais sejam, Geração, Subestação ou Linhas de Transmissão.

Tabela 1 – Porcentagem de Curto-Circuito por Sistema do Setor Elétrico

Setor do Sistema Elétrico	Curto-Circuito
Geração	06%
Subestação	05%
Linhas de Transmissão	89%

Fonte: Curto-Circuito (KINDERMANN, 1997)

A Tabela 2 mostra uma relação de ocorrência em % para o Tipos de Curto-Circuito, Monofásico, Bifásico e Trifásico.

Tabela 2 – Porcentagem de ocorrência por Tipo de Curto

Tipos de Curtos-Circuitos	ocorrências em %
3ϕ	06%
2ϕ	15%
2ϕ-terra	16%
1ϕ	63%

Fonte: Curto-Circuito (KINDERMANN, 1997)

O planejamento e a operação de Sistemas Elétricos de Potência são realizados visando o equilíbrio do custo e da qualidade da geração, transmissão e distribuição da energia. Nesse sentido, as faltas não podem ser evitadas, tornando imprescindível o desenvolvimento de técnicas para localizá-las no menor tempo possível, a fim de minimizar danos irreparáveis ao sistema e aos seus usuários. Cumpre destacar, por exemplo, que, na ocorrência de curtos e faltas, o sistema pode experimentar grandes variações de corrente e tensão, que podem resultar em fortes efeitos térmicos e mecânicos ao sistema (MARTINS,

2021), para exemplificar essas variações, um estudo de afundamento de tensão em faltas monofásicas é feito no Apêndice A.

Ao se tratar de tecnologias modernas de monitoramento de sistemas elétricos, dados em tempo real acerca dos fasores de tais eventos podem ser capturados (medidos e processados) usando-se PMUs. A próxima seção tem como objetivo apresentar em detalhes esta tecnologia.

2.2 Unidades de Medição Fasorial

2.2.1 PMU's Como Ferramenta de Medição

Coletar medições de fasores sincronizados para Sistemas Elétricos de Potência é uma necessidade cada vez maior quando se trata de seu controle e monitoramento. A vantagem de realizar essa coleta é a possibilidade de analisar o sistema através de um *Snapshot*, onde é possível avaliar os fasores elétricos de todas as componentes do SEP para uma mesma referência de tempo. A partir de uma PMU, é possível se ter um registro no tempo de forma precisa das correntes, tensões da frequência do sistema (SINGH *et al.*, 2011).

As primeiras tentativas de realizar uma medição sincronizada se deram nos anos 80, tecnologias como modulação AM e sinais via satélite GOES (Geostationary Operational Environmental Satellite) foram utilizados (MARTINS, 2012). Tais tecnologias conseguiam atingir uma sincronização de 40μs, não alcançando um nível necessário de precisão para serem generalizáveis (PHADKE; THORP, 2008). Além disso, tais sistemas não conseguiam trabalhar com a defasagem em sistemas trifásicos.

Para se atingir uma precisão que pudesse tornar o sistema de medição generalizável, foram desenvolvidas as PMUs, sigla para Unidade de Medição Fasorial, que utiliza sinal de GPS como responsável pela sua sincronização no espaço e no tempo.

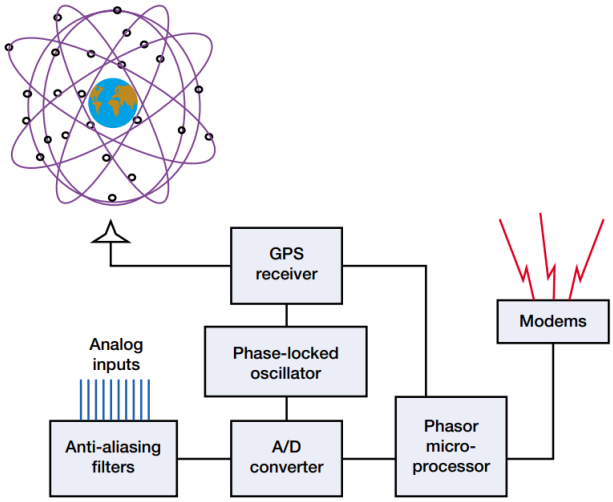
Em 2011 lançou-se a última revisão do protocolo IEEE C37.118.1 (IEEE. . . , 2011b) e IEE C37.118.2 (IEEE. . . , 2011a) sendo protocolos respectivamente responsáveis por

definir características de medição e de transmissão dos dados do sistema coletados pelas PMU's. As características descritas em 2.2.2 e 2.2.3 possuem embasamento tanto nos protocolos quanto em trabalhos relacionados citados ao longo do texto.

2.2.2 Características Construtivas de Uma PMU

Para entender o funcionamento de uma PMU, é interessante observar sua estrutura básica, conforme mostra a Figura 1.

Figura 1 – Estrutura de uma PMU



Fonte: (HART *et al.*, 2001)

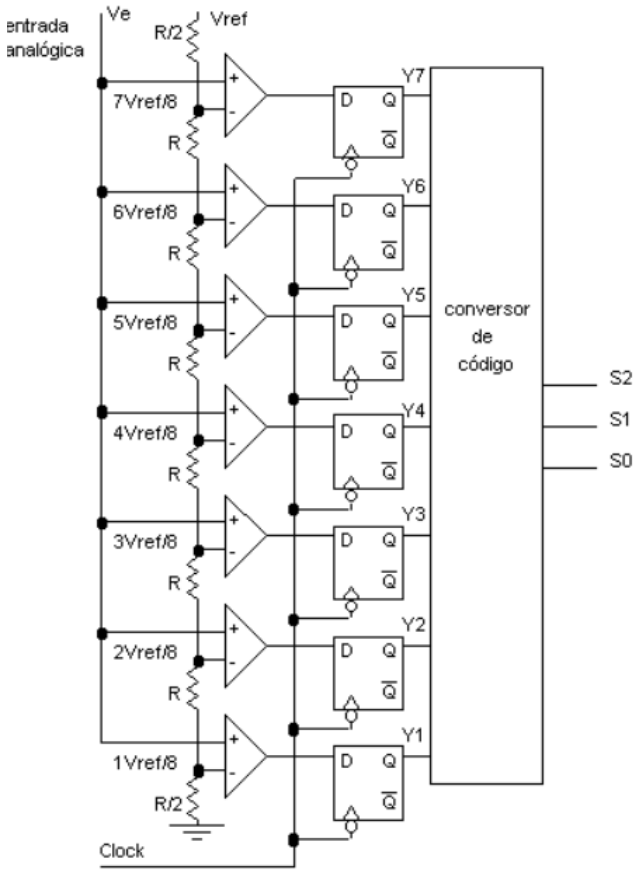
Inicialmente, o sinal analógico de tensão e corrente é medido e filtrado por um filtro Anti-aliasing, garantindo a coleta correta do sinal a partir de uma taxa de amostragem que respeite o Critério de Nyquist. Durante esse processo, o GPS sincroniza a aquisição dos dados. Então, ele envia constantemente um sinal de um pulso por segundo o qual será a base de tempo para todo o processo. Este trem de pulsos emitido tem uma precisão maior do que 1μs, possibilitando ser a referência de tempo para que o processo de aquisição dos dados seja sincronizado mesmo nas subestações geograficamente distantes. O receptor de sinal de GPS divide o sinal em intervalos menores para que haja um número maior de amostras de grandezas fasoriais no período de um segundo. Por exemplo, para uma frequência de 60 Hz, 12 aquisições por ciclo são suficientes para representar o sinal amostrado com precisão adequada (FERREIRA,).

Como o sinal coletado é analógico, para realizar seu armazenamento é necessário digitalizar esse sinal, para isso um conversor A/D é utilizado. A partir da resolução do conversor, o valor analógico é convertido para digital a partir de um circuito comparador, que pode variar de velocidade e preço a partir de sua tecnologia (FLOYD, 2007).

Um conversor A/D possui diversos circuitos comparadores dentro de sua topologia, quanto maior a quantidade, maior a resolução do conversor. Para realizar o processo de conversão, um circuito comparador possui um limiar, que quando processa um sinal analógico, define se a saída do comparador terá saída alta (Bit de valor 1) ou baixa (Bit de valor 0). Ao final da conversão, o sinal analógico terá sido convertido em um sinal digital representado por uma sequência de bits.

A Figura 2 exemplifica a topologia de um conversor A/D paralelo.

Figura 2 – Conversor A/D paralelo de 3 bits



É importante comentar que um sinal vindo da rede pode contar harmônicas indesejadas devido às flutuações nos valores de tensão e corrente, causada por cargas

capacitivas e indutivas, por isso, junto ao processo de conversão do sinal, uma técnica para filtrar o valor das harmônicas é a Transformada Discreta de Fourier, de forma que apenas a componente fundamental da frequência é obtida (NADUVATHUPARAMBIL; VALENTI; FELIACHI, 2002), A equação 2.1 representa esse processamento.

$$X = \frac{\sqrt{2}}{N} \sum_{k=1}^{N} x_k e^{-jk\left(\frac{2\pi}{N}\right)} \tag{2.1}$$

Sendo:

- X é o Fasor medido;
- N é o número de amostras medidas em um ciclo de onda;
- x_k são os valores das amostras retiradas da forma de onda.

Além disso, durante a etapa de medição, para garantir a precisão do sinal medido, testes que estimem o erro da amostra são executados (IEEE. . . , 2013). A equação 2.2 fundamenta este teste.

$$EVT = \sqrt{\frac{[x_r(n) - X_r]^2 - [x_i(n) - X_i]^2}{x_r - x_i}} \tag{2.2}$$

Sendo:

- $x_r(n)$ é a parte real do sincrofasor medido;
- $x_i(n)$ é a parte imaginária do sincrofasor medido;
- x_r é a parte real do sinal de entrada;
- x_i é a parte imaginária do sinal de entrada;
- n representa o índice da barra.

É exigido que o EVT seja de menos que 1% para operação em nível de conformidade tanto 0 quanto 1, as diferenças entre os níveis de conformidade podem ser visualizadas na Tabela 3.

Tabela 3 – Limite de Erro em medição de grandezas fasoriais

Característica	**Referência**	**Limites das grandezas de influência**	
		Nível 0	**Nível 1**
Frequência do Sinal	60 Hz	± 0,5 Hz	± 5 Hz
Magnitude do Sinal	100% da nominal	80 a 120% da nominal	10 a 120% da nominal
Ângulo de Fase	0 radianos	$\pm\pi$ radianos	$\pm\pi$ radianos
Distorção harmônica total	< 0,2%	1% para qualquer harmônica até 50^o	10% para qualquer harmônica até 50^o
Sinal de interferência fora da banda	< 0,2% da magnitude do sinal	1% da magnitude do sinal de entrada	10% da magnitude do sinal de entrada

2.2.3 Transmissão de Sincrofasores

Em seguida à aquisição, os dados devem ser transmitidos e alocados em uma PDC (Phasor Data Concentrator). A tecnologia responsável por garantir esta transmissão sofreu grandes evoluções com o passar dos anos, assim como a topologia da sua rede, podendo existir desde um cabeamento para sua realização, como linhas telefônicas ou cabos de fibra óptica, até processos de transmissão *wireless* (NADUVATHUPARAMBIL; VALENTI; FELIACHI, 2002). A seguir serão apresentados algumas exigências definidas por padrões pela IEEE para garantir a padronização dos protocolos de comunicação de uma PMU.

Em IEE C37.118.2 (IEEE. . . , 2011a) estabelece-se a necessidade da transmissão de dados sincronizados em UTC. Com precisão suficiente para atender os requisitos da tabela 3, ou seja, 0,01 radianos (0,57 graus) que representa 1% do EVT exige uma precisão de ± $26\mu s$ para sistemas de 60 Hz e ± $31\mu s$.

Em relação a taxa de amostragem, a PMU deve suportar a taxa como submúltiplos da frequência nominal da linha de transmissão. As taxas mínimas exigidas para sistemas de 60 Hz são, 10, 12, 15, 20, 30, 60 Frames de amostra por segundo.

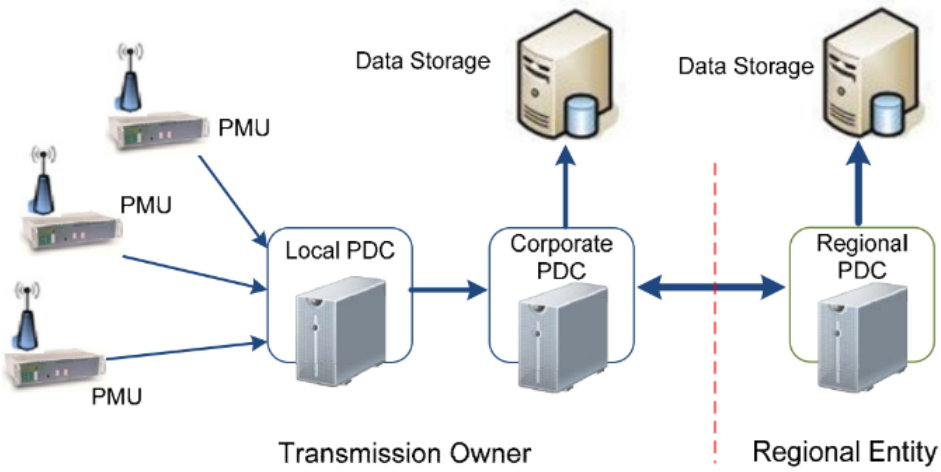
PDC's funcionam como nós em redes de comunicação, dependendo da topologia da rede de comunicação de um Sistema de Potência, ela pode executar diferentes funções:

- Verificações de qualidade nos dados de fasores e inserção de flags apropriados no fluxo de dados correlacionados.
- Verificações para flags de distúrbio e gravação de arquivos de dados para análise.

- Monitoramento do sistema de medição geral e exibição dos resultados.
- Saídas especializadas, como uma interface direta para um sistema SCADA ou EMS.

A Figura 3 apresenta um exemplo de rede de PDCs.

Figura 3 – Rede exemplar de PDCs



É possível analisar PDCs com diferentes funções, bem como uma comunicação estabelecida entre PDCs. Isso ocorre devido a necessidade do sistema e a preferência da topologia estabelecida pelos projetistas da solução apresentada. Ao se comunicar com diferentes PMU's, uma *Local PDC* faz a função de agregar e ordenar fasores no tempo, *Corporate PDCs* são responsáveis por uma verificação na qualidade e integridade dos dados, podendo também algumas vezes armazená-los diretamente, sem a necessidade de passar por uma *Regional PDC*, que é especialmente dedicada ao armazenamento de dados.

Por fim, ressalta-se informações sobre o protocolo de comunicação estabelecidos entre os nós da rede de comunicação e o formato das suas mensagens. Para transmissão dos dados entre os nós qualquer protocolo pode ser seguido, é indicado apenas a utilização de testes que garantam a perenidade do dado por toda a rede como o uso de CRC's (Cyclic Redundancy Check) e Syncwords.

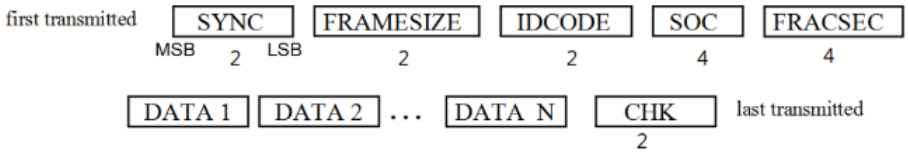
O formato da mensagem transmitida é codificada e configurada para que 4 tipos de mensagens possam ser transmitidos, são eles: dados, configuração, cabeçalho e comando.

- Mensagens de Dados são as medições feitas por uma PMU.

- A configuração é uma mensagem legível por máquina que descreve os tipos de dados, fatores de calibração e outros metadados para os dados que a PMU/PDC envia.
- As informações do cabeçalho são informações descritivas legíveis por humanos enviadas da PMU/PDC, mas fornecidas pelo usuário.
- Comandos são códigos legíveis por máquina enviados para a PMU/PDC para controle ou configuração.

No geral, diferentes configurações de mensagens podem ser adotadas, e a forma de decodificar sua informação normalmente é tabelada por usuários daquela rede, a Figura 4 exemplifica uma estrutura de mensagem.

Figura 4 – Exemplo de mensagem transmitida



Sendo:

- *SYNC* (Palavra de Sincronização): 2 bytes usados para sincronização e identificação do frame.
- *FRAMESIZE* (Tamanho do Quadro): 2 bytes que indicam o tamanho total da mensagem transmitida.
- *SOC* (Segundo do Século): 4 bytes que representam os segundos desde 1º de janeiro de 2000.
- *IDCODE*: 2 bytes que identificam a origem da mensagem como o identificador único de uma PMU ou PDC.
- *FRACSEC* (Segundos Fracionários): 4 bytes que incluem a fração de um segundo e um sinalizador de qualidade de tempo.

- *CHK* (Palavra de Verificação): Código CRC-CCITT que assegura a integridade dos dados transmitidos.

Em suma, PMU's são ferramentas úteis para visualizar e armazenar dados de Sistemas de Potência. O Protocolo IEE C37.118.1 padroniza aspectos importantes ao se construir uma PMU referente a medição dos dados, o protocolo IEE C37.118.2 padroniza aspectos importantes referente a transmissão dos dados. Para garantir que a informação chegue com qualidade ao humano, uma série de testes e equipamentos intermediários são utilizados. As PDC's são ferramentas essenciais na rede de transmissão de uma PMU, pois ela pode ser utilizada como um nó multifuncional na rede, sendo responsável por manipular e processar os dados, aplicar testes de qualidade e armazenar os dados em um disco rígido ou banco em nuvem.

2.3 Aprendizagem de Máquina

Essa seção, tem como objetivo familiarizar alguns conceitos de aprendizagem de máquina, como o tipo do aprendizado utilizado neste trabalho e como o modelo aprende, posteriormente será discutido em detalhes uma categoria de modelo chamada Árvore de Decisão.

2.3.1 Tipos de Aprendizado

A Aprendizagem de máquina pode ser categorizado em 3 amplas categorias (GÉRON, 2019c). São elas:

Aprendizado Supervisionado: É conhecida a variável que se quer prever e ela está disponível para ser utilizada no treinamento do modelo, podendo ela ser contínua ou categórica, categorizando o aprendizado supervisionado como de regressão e classificação, respectivamente.

Aprendizado Não Supervisionado: Não se tem uma variável que deseja-se ser prevista. Nesse tipo de aprendizado, o modelo faz, por exemplo, o trabalho de clusterização a partir de características semelhantes entre as amostras de dados.

Aprendizado por Reforço: Através de um sistema de recompensas dado ao modelo, um agente interage com o ambiente e, a partir dessas interações, aprende a tomar decisões que maximizem uma recompensa acumulada ao longo do tempo.

Para o presente estudo, será utilizado um modelo que prevê uma variável categórica binária, se encaixando como um modelo supervisionado de classificação. Decorrente deste fator, os conceitos a seguir serão apresentados com foco em problemas de classificação.

2.3.2 Função Custo

Para encontrar o modelo ótimo é preciso definir um critério, que se traduz matematicamente por meio de uma função que quantifica o erro entre os valores preditos (pelo modelo) e os observados. Ao minimizá-la, obtém-se os parâmetros estimados mais adequados segundo o critério adotado. Essa função é denominada de função custo (BISHOP, 2006).

Na equação 2.3 é apresentado um exemplo de uma função custo muito utilizada em problemas de classificação binária, chamada de Entropia Cruzada Binária:

$$CE_{(y,p)} = \begin{cases} -y\ln(p), & \text{se} \quad y = 1 \\ -(1-y)\ln(1-p), & \text{se} \quad y = 0 \end{cases} \tag{2.3}$$

Nessas equações, CE denota a entropia cruzada (do inglês *cross-entropy*). Como em um modelos de classificação binária o valor observado e predito podem ser apenas 0 ou 1, separam-se ambos cenários e tratamos eles individualmente. A variável p nos indica a probabilidade do valor predito daquela amostra ser y. Note-se que para encontrar uma expressão generalizável para a entropia cruzada binária os casos para $y = 1$ e $y = 0$ podem ser somados e uma média em relação a todo conjunto de amostras representará a expressão geral, esta expressão é apontada em 2.4.

$$CE_{(y,p)} = -\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \cdot ln(p^{(i)}) + (1-y^{(i)}) \cdot \ln(1-p^{(i)}) \tag{2.4}$$

Dessa forma, a função custo final é a soma do valor individual da função custo

para cada amostra, dividido pelo número de amostras, ou seja, a média. Quando o valor predito condiz com o valor observado, a função custo tende a zero, assim como, quando os valores preditos se diferem dos valores observados, a função custo diverge de 0. Assim, minimizar a função custo significa encontrar um modelo que busca aproximar os valores preditos dos valores observados.

2.3.3 Processo de Treinamento de um Modelo

É necessário possuir um algoritmo que a cada iteração e predição do modelo, atualize a função custo, juntamente com os parâmetros que definem essa predição. A seguir, será descrito possíveis comportamentos do modelo dependendo da forma como ele é treinado, e o algoritmo de treinamento será descrito mais adiante na seção 2.4, visto que esta terá foco em revisar os Algoritmos de Árvore de Decisão.

Ao se realizar o processo de Aprendizado Supervisionado, é necessário que o modelo tenha acesso a um conjunto de dados que no qual possa ser treinado, possuindo características que se relacionem com a variável que deseja-se prever (GÉRON, 2019c). Em dados tabulares as características podem ser descritas como as colunas de uma tabela, enquanto que as linhas representam uma amostra. Este conjunto de dados destinado ao aprendizado do modelo é chamado de Dados de Treinamento.

Na maioria dos casos, adquirir um conjunto de dados pode ser custoso e uma tarefa nada trivial, deste modo, durante o treinamento do modelo, ou seja, durante o processo de otimização de seus parâmetros e redução do valor da função custo, dependendo da quantidade e da qualidade das amostras, é possível que o modelo entre em dois possíveis estados que prejudicarão a tarefa a ser realizada: o de sobreajuste (ou, em inglês, *overfitting*) e o de subajuste (*underfitting*) (GHOJOGH; CROWLEY, 2023a).

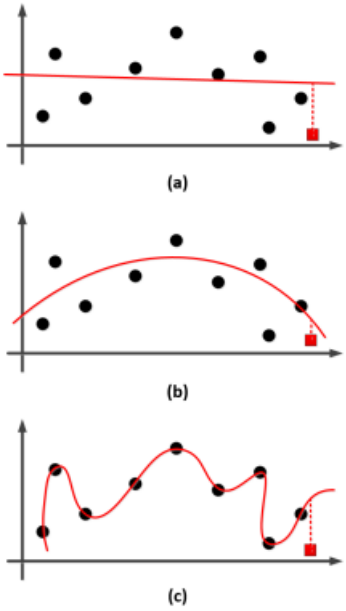
O estado de sobreajuste refere-se ao problema do modelo se hiper-ajustar aos dados de treinamento, se tornando bom demais em prever o resultado das amostras que lhe foi apresentado, porém pouco assertivo ao prever o resultado de amostras não antes vistas. De uma perspectiva da qualidade dos dados, um conjunto de dados com baixa diversidade (*data diversity*), ou seja, baixa capacidade de representar o fenômeno de forma generalizável

está diretamente relacionado com o sobreajuste (GONG; ZHONG; HU, 2019).

No fenômeno de subajuste o modelo não possui capacidade de se ajustar de forma que sua predição seja assertiva, seja para os dados de treinamento ou para os dados de validação. O subajuste está diretamente relacionado a incapacidade dos dados de treinamento de estabelecer uma relação com a variável a ser predita, dessa forma, o modelo não possui qualquer aprendizado com o dado visto no treinamento (GHOJOGH; CROWLEY, 2023b).

Na Figura 5 visualiza-se graficamente os fenômenos de subajuste (a), ajuste ideal (b) e sobreajuste(c).

Figura 5 – Subajuste, Ajuste ideal e Sobreajuste



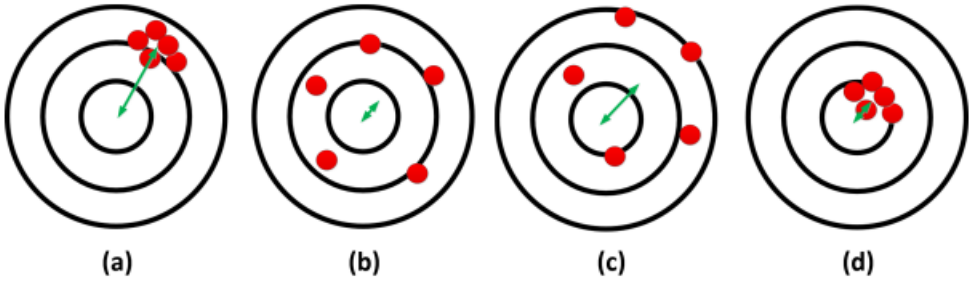
Fonte: (GHOJOGH; CROWLEY, 2023a)

Na Figura 5, as circunferências pretas representam os dados de treinamento, o quadrado vermelho uma nova amostra, não vista anteriormente pelo modelo. Conclui-se que modelos subajustados e sobreajustados tendem a performar pior em geral para novas amostras.

Complementando uma boa coleta de dados, também é possível trabalhar as problemáticas de sobreajuste e subajuste de outras formas. Para isso, descreve-se a seguir os conceitos de viés e variância em aprendizagem de máquina.

Pode-se entender o trabalho de treinamento de um modelo como achar o ponto ótimo entre a quantidade de viés que um modelo possui e a quantidade de variância (RASHIDI *et al.*, 2019). Viés representa a diferença entre o valor previsto e o valor observado, a variância mede a variabilidade do valor previsto frente uma variação nos dados de treinamento (PICHLER; HARTIG, 2023). Dessa forma, é possível classificar os resultados de um modelo em 4 quadrantes de comportamento a Figura 6 descreve-os.

Figura 6 – Desempenho de um modelo baseado em Viés e Variância



Fonte:(GHOJOGH; CROWLEY, 2023a)

- (a) Alto Viés e Baixa Variância - O modelo não se aproxima das predições ideais e uma variação nos dados de treino não causam variações significativas em sua predição
- (b) Baixo Viés e Alta Variância - O modelo tende a se aproximar das predições ideais e uma variação nos dados de treino causam variações significativas em sua previsão
- (c) Alto Viés e Alta Variância - O modelo não se aproxima das predições ideais e uma variação nos dados de treino causam variações significativas em sua previsão
- (d) Baixo Viés e Baixa Variância - O modelo se aproxima das predições ideais e uma variação nos dados de treino não causam variações significativas em sua previsão

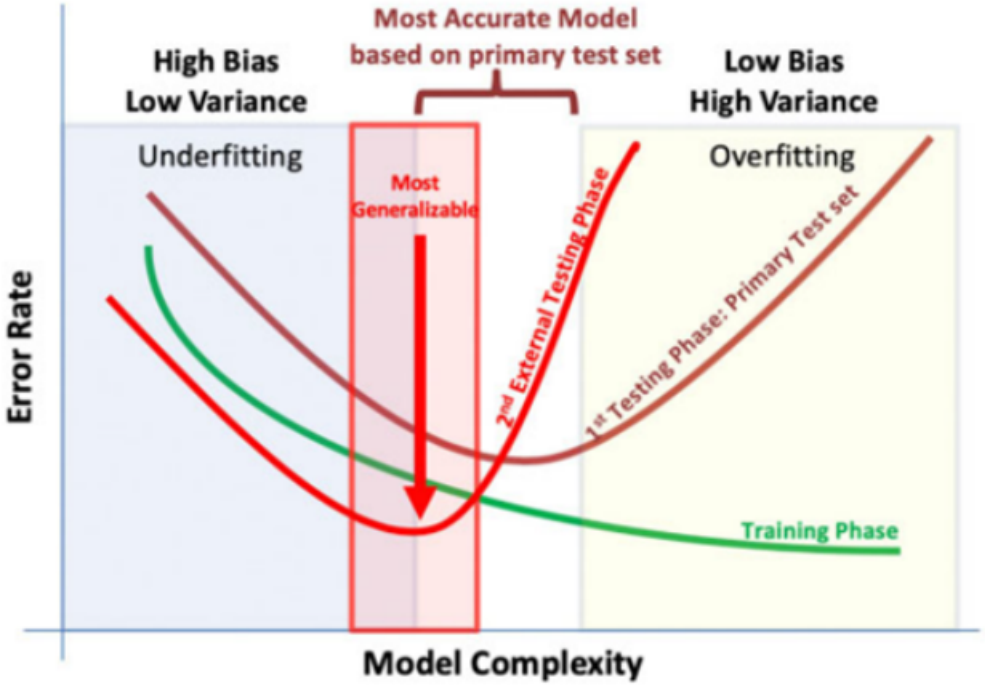
Através de tais descrições, é possível relacionar que um modelo subajustado tem o comportamento do caso (a) enquanto que um modelo sobreajustado possui o comportamento do caso (b).

A relação proposta leva a entender que o ideal seria ter um modelo que possua Baixo Viés e Baixa Variância, porém, dentro do contexto de Aprendizagem de Máquina,

ambas variáveis propõem entre si uma relação de troca. Este é um fenômeno mais conhecido pela sua terminologia em inglês, *Bias-Variance Trade-Off*.

A Figura 7 demonstra essa relação de troca, onde ao se inserir uma quantidade de Viés, automaticamente se reduz a quantidade de Variância presente em um modelo. Além disso, a Figura relaciona ambos conceitos com o erro em eixo y e a complexidade do modelo em eixo x.

Figura 7 – Relação entre Viés, Variância, Sobreajuste, Subajuste, Erro e complexidade do modelo.



Fonte:(GHOJOGH; CROWLEY, 2023a)

Conclui-se que encontrar um modelo que performa bem, é encontrar um modelo generalista que aprenda a partir do conjunto de treino mas também generalize bem sua solução para um conjunto de dados não visto previamente, nomeado durante a modelagem como conjunto de dados de validação. Para isso, balancea-se os valores de viés e variância conhecendo o modelo e entendendo como a variação de seus hiperparâmetros aumentam ou diminuem sua complexidade, além de garantir que o conjunto de treino seja balanceado e diverso representando bem o comportamento das características e sua relação com a variável alvo.

2.4 Modelos Baseados em Árvores de Decisão

Esta seção tem como objetivo descrever com profundidade como funcionam algoritmos baseados em árvore de decisão visando o domínio do seu funcionamento, visto que esse é o tipo de modelo que será usado neste trabalho.

2.4.1 Estrutura e Funcionamento

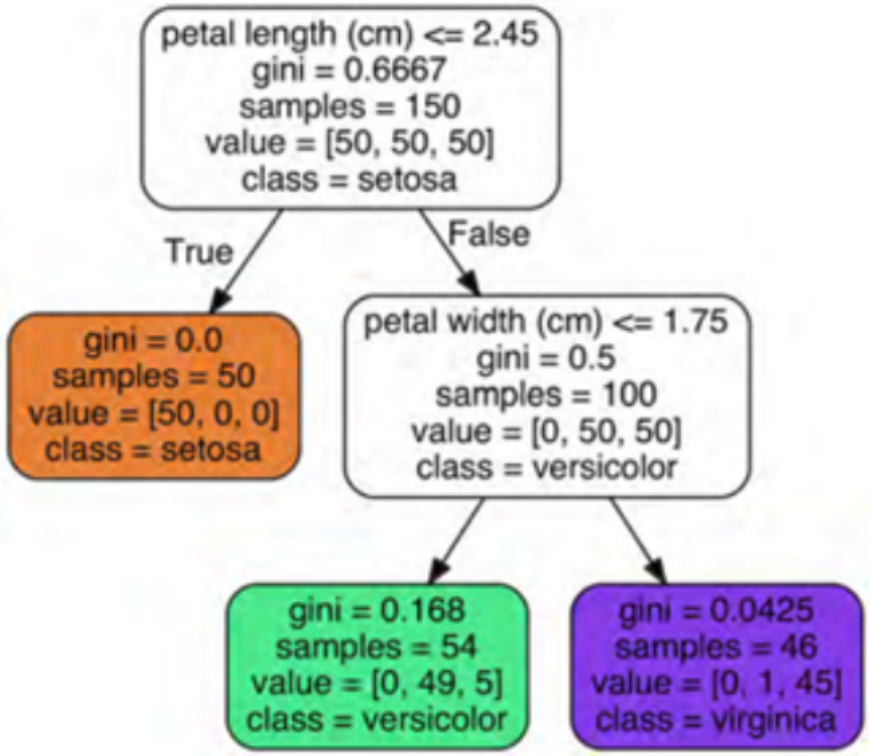
Árvores de decisão são algoritmos versáteis e poderosos, amplamente utilizados em tarefas de classificação binária, multiclasse e tarefas regressivas (GÉRON, 2019d), assim como, também possuem utilização em aprendizagem não supervisionada (LIU; TING; ZHOU, 2008) e aprendizagem por reforço (AWAYA; MA, 2023).

Para fazer sua predição, a árvore é organizada em nós que são atribuídos a uma característica do conjunto de dados, tenta-se então encontrar um valor para a característica que separa a variável alvo observada de forma ótima, criando dois novos nós que podem vir a ser testados por outra característica do conjunto de dados, até que um critério de parada seja atingido, os nós localizados ao final da árvore e que não sofrem uma separação são chamados de folhas.

A compreensão de uma árvore de decisão pode ser facilitada a partir do entendimento de um exemplo prático, para isso, será utilizado um exemplo da obra "Mãos à Obra: Aprendizado de Máquina com Scikit-Learn, Keras & TensorFlow: Conceitos, Ferramentas e Técnicas Para a Construção de Sistemas Inteligentes"de Aurélien Géron.

A Figura 8 representa um esquema de árvore de um problema vindo de um conjunto de dados famoso em estudos de Aprendizagem de Máquina, no qual se analisa através de dados da pétala de determinadas flores de íris a qual espécie ela pertence: *Setosa*, *Versicolor* ou *Virginica*.

Figura 8 – Exemplo de árvore de decisão



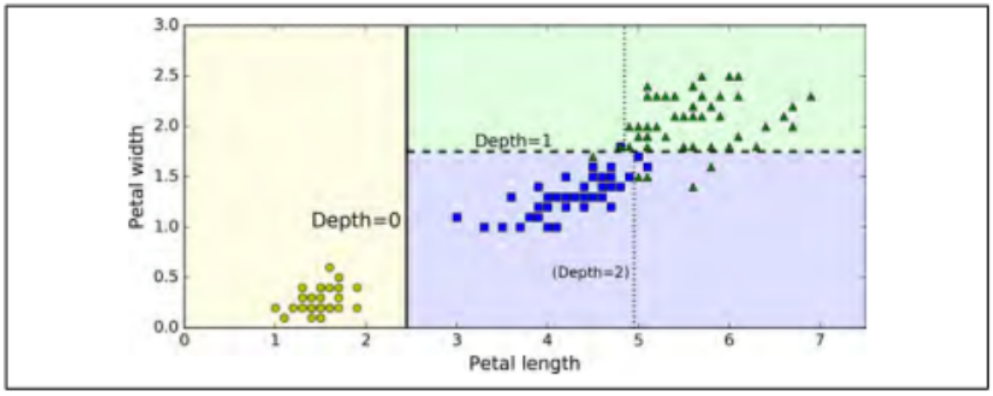
Fonte:(GÉRON, 2019e)

Na figura 8, a árvore de decisão pode ser traduzida como o seguinte raciocínio: "Se o comprimento da pétala(*length*) for menor que 2,45 centímetros, essa Iris é da espécie *Setosa*, caso contrário, avalia-se sua largura(*width*). Se sua largura for menor que 1,75 centímetros, essa Iris é da espécie *Versicolor*, se for maior, essa Iris é da classe *Virginica*". Também observamos que esse problema é um problema em que a variável a ser prevista está previamente balanceada, possuindo 46, 50 e 54 espécies de Iris, e que dessas 150, apenas 6 são classificadas errado, resultando em uma acurácia de 96%.

Como este é um problema de apenas 2 características, é possível visualizá-lo a partir de um gráfico cartesiano que mostra as fronteiras de decisão da árvore do exemplo. A Figura 9 traz esta visualização.

A Figura 9, além de reforçar a ideia de separação dada pela árvore, também é possível visualizar que o número de fronteiras de decisão para esse problema é igual ao tamanho da profundidade da árvore, indicado pelo hiperparâmetro *depth* podendo ser ajustável na hora da modelagem.

Figura 9 – Fronteiras de uma Árvore de Decisão de duas Características



Fonte:(GÉRON, 2019f)

2.4.2 Critérios de Separação e Treinamento

Entender o significado matemático de uma árvore de decisão está diretamente relacionado com entender seu critério de separação feito nos nós, ou seja, do seu método de *"splitting"*. Intuitivamente, pensa-se que a seleção de quais características causarão o *splitting* no modelo, é efetivamente aquela que melhor separa os dados em conjuntos puros, ou seja, que possuem uma menor quantidade de entropia.

Entropia é uma medida utilizada para calcular o ganho de informação no campo da teoria da informação, ela caracteriza a falta de homogeneidade dos dados em relação a sua classificação (GÉRON, 2019g). Por exemplo, dado um conjunto de dados S que pode ter c classes distintas, a entropia de S será dada a partir da equação 2.5.

$$Entropia(S) = \sum_{i=1}^{c} -p_i log_2 p_i \tag{2.5}$$

Em que p_i é a proporção de dados em que S que pertecem a classe i

Uma outra opção dentro da modelagem de árvores de decisão é o Coeficiente de Gini, que da mesma forma que a Entropia, calcula o grau de impureza do conjunto de dados após uma etapa de *splitting*, abaixo a equação 2.6 sendo sua representação.

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2 \tag{2.6}$$

É possível ver sua semelhança com a fórmula da entropia, na prática, pouca

diferença ocorre em uma modelagem visto que no final é alcançada uma árvore semelhante. A principal diferença é em relação ao balanceamento de árvore, enquanto o Coeficiente de Gini tem a tendência de produzir árvores em que uma classe é mais isolada em uma das raízes da árvore, entropia produz uma árvore mais balanceada (GÉRON, 2019g).

Em seguida é necessário escolher um algoritmo de otimização para a modelagem, dentre as árvores de decisão, diversos algoritmos podem ser usados, aqui será exemplificado o mais famoso deles. O CART (*Classification and Regression Trees*) é o algoritmo padrão utilizado na biblioteca Sk-Learn desenvolvida para Python, que suporta diversos algoritmos de modelagem. Por isso vamos usá-lo como exemplo (PEDREGOSA *et al.*, 2011a).

O Algoritmo CART nada mais é que um processo iterativo de tentativa e erro da aplicação da função escolhida como critério de separação (Entropia ou Coeficiente de Gini) para todas as características do conjunto de dados de forma a minimizar a função custo.

Dado um nó m, e seus dados sendo representados por Q_m, com n_m amostras, para cada nó, encontramos um par de valores $\theta = (j, t_m)$, sendo j a característica e t_m o limiar que separa os dados. Separamos então o conjunto a esquerda e a direita, representado pelas equações 2.7 e 2.8.

$$Q_m^{left}(\theta) = \{(x, y) \mid x_j \leq t_m\} \tag{2.7}$$

$$Q_m^{right}(\theta) = \frac{Q_m}{Q_m^{left}}(\theta) \tag{2.8}$$

Por fim, a função custo computada é a equação 2.9.

$$G(Q_m, \theta) = \frac{n_m^{left}}{n_m} H(Q_m^{left}(\theta)) + \frac{n_m^{right}}{n_m} H(Q_m^{right}(\theta)) \tag{2.9}$$

Sendo $H()$ a função impureza que assume a forma de Coeficiente de Gini ou Entropia. No fim, será obtido um conjunto de parâmetros de pares características e *thresholds* que minimizam a função custo computada (PEDREGOSA *et al.*, 2011a), conforme demonstrado na equação 2.10.

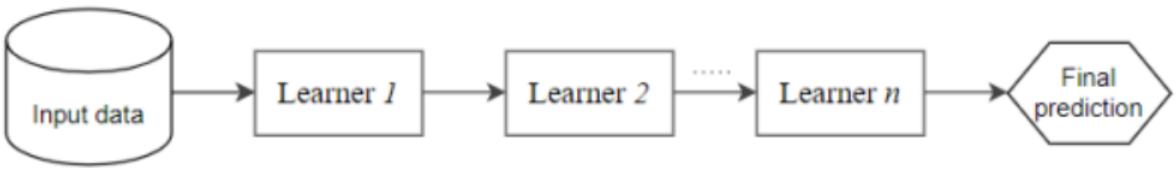
$$\theta^* = argmin_\theta G(Q_m, \theta) \tag{2.10}$$

2.5 Métodos Ensemble

Ensemble é o nome da técnica utilizada ao agregar dois ou mais modelos que sozinhos performam mal, mas quando utilizados em conjunto possuem um bom aumento na sua performance. Desta forma, para realizar esta técnica, é necessário estabelecer um critério que permitam os modelos interagirem entre si na hora de serem treinados, chamando este processo de Ensemble Learning. (PEDREGOSA *et al.*, 2011a)

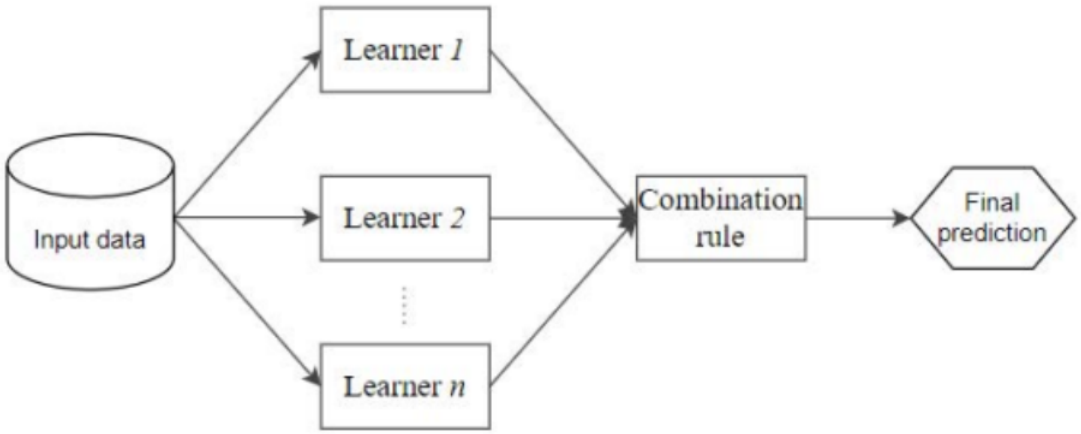
É possível agrupar o método de Ensemble Learning em 3 grandes grupos, Boosting, Bagging e Stacking (MIENYE; SUN, 2022b), na Figura 10 observa-se um diagrama representativo de Boosting e na Figura 11 de Bagging, em seguida suas descrições.

Figura 10 – Diagrama de um algoritmo Boosting



Fonte:(MIENYE; SUN, 2022b)

Figura 11 – Diagrama de um algoritmo Bagging



Fonte:(MIENYE; SUN, 2022b)

2.5.1 Boosting

Boosting é uma técnica que ordena modelos um pouco melhores que modelos aleatórios, chamados de aprendizes fracos, que são treinados iterativamente. Após o treinamento de um dos aprendizes fracos, ele realiza predições e na próxima iteração, outro modelo será treinado, porém com um parâmetro extra voltado para adicionar peso na correção dos erros do modelo anterior. O processo iterativo continua até um critério de parada ser atingido, sendo este critério, normalmente, o número de aprendizes fracos utilizados (MIENYE; SUN, 2022a).

Modelos Boosting tendem a reduzir mais viés do que a variância, podendo facilmente sobreajustar os dados, adicionando demasiada complexidade ao modelo, mesmo quando o conjunto de dados não necessita.

No âmbito das árvores de decisão, algoritmos Boosting são extremamente utilizados, árvores de decisão de apenas um nó fazem a função do aprendiz fraco. O algoritmo de treinamento Gradient Boosting, forma a base dos principais modelos de boosting utilizados hoje, ele utiliza o gradiente da função custo para nos apontar a direção para minimizar a função, e atualizamos os parâmetros através de um multiplicador chamado de taxa de aprendizagem, vejamos o passo a passo do seu treinamento (MIENYE; SUN, 2022a).

Dado um conjunto de treinamento S tal que:

$$S = (x_1, y_1), ..., (x_2, y_2), ..., (x_m, y_m)$$

E uma função custo L tal que:

$$L(y, F(x))$$

Inicializa-se o treinamento através de um modelo base com parâmetros arbitrariamente escolhidos, esses parâmetros são representados por F, como esta é a primeira iteração do algoritmo, utiliza-se F_0, segundo a equação 2.11.

$$F_0 = argmin_\alpha \sum_{i=1}^{N} L(y_i, \alpha) \tag{2.11}$$

Para cada iteração m, calcula-se o gradiente da função custo, que são as derivadas parciais da função, com respeito às variáveis independentes escolhidas na iteração anterior, a partir disso, obtém-se a magnitude dos erros de cada característica, esses erros são chamados de resíduos, e representam a diferença entre os valores reais e a previsão do modelo na iteração m (CARUANA *et al.*, 2004) esse cálculo é apresentado na equação 2.12.

$$r_{mi} = [\frac{\delta L(y_i, F(x))}{\delta F(x)}]_{F(x)=F_{m-1}(x)} \tag{2.12}$$

Treina-se um novo modelo base, as características desse treinamento agora serão as mesmas, porém a variável a ser prevista serão os erros estipulados no cálculo anterior, o objetivo deste passo é encontrar relações entre os dados de entrada e os erros calculados anteriormente.

Ao encontrar essa relação entre os erros e as características, tenta-se encontrar valores para as características que minimizem os erros que agora são previstos pelo modelo base e isso fica conhecido como contribuição do modelo base ao processo.

Após encontrar a contribuição, o modelo faz uma otimização do parâmetro da taxa de aprendizado (ρ), essa otimização é feita a partir do cálculo da função custo quando a

contribuição é adicionada ponderada por diferentes magnitudes. A taxa de aprendizado escolhida será aquela que melhor minimiza a função custo. A equação 2.13 representa essa etapa do algoritmo.

$$(\rho_m h_m(x)) = argmin_{p,h} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \rho h(x_i)) \tag{2.13}$$

Sendo:

- $argmin_{p,h} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i))$ é a função custo inicialmente calculada na iteração anterior, carregada para esta fase da iteração
- $\rho h(x_i)$ é a contribuição do modelo base ponderado pela taxa de aprendizado escolhida
- $(\rho_m h_m(x))$ é a contribuição ponderada pela taxa de aprendizado que irá atualizar os parâmetros para iteração subsequente

Por fim, o modelo é atualizado para a próxima iteração a partir da equação 2.14

$$F_m(x) = F_{m-1}(x) + \rho_m h_m(x) \tag{2.14}$$

A principal vantagem do Gradient Boosting é que, assim como outros algoritmos de boosting, ele pode aprender padrões complexos e não lineares a partir dos dados de entrada (MIENYE; SUN, 2022a), já que é treinado para corrigir os erros do modelo anterior. No entanto, um modelo construído com esse algoritmo tende a ter facilidade em se sobreajustar (NATEKIN; KNOLL, 2013). Esse algoritmo é ideal para aplicações com conjuntos de dados pequenos (JIANG *et al.*, 2020).

Hoje os modelos de Boosting detém o posto de modelos estado da arte na aplicação de árvores de decisão. A seguir será apresentado exemplos de modelos utilizados neste trabalho.

2.5.1.1 Extreme Gradient Boosting (XGB)

Publicado em 2016 por Chen e Guestrin (CHEN; GUESTRIN, 2016a), o XGBoost implementou um termo regularizador na função custo, sendo responsável por diminuir as chances de sobreajuste do modelo (CHEN; GUESTRIN, 2016b). A função custo do XGBoost pode ser representada com a equação 2.15. A equação 2.16 descreve o termo regularizador $\Omega(h_m)$.

$$L_M(F(x_i)) = \sum_{i=1}^{n} L(y_i, F(x_i)) + \sum_{m=1}^{M} \Omega(h_m) \tag{2.15}$$

$$\Omega(h_m) = \gamma T + \frac{1}{2}\lambda\|\omega\|^2 \tag{2.16}$$

- γ é um parâmetro de complexidade, ele funciona como um limiar, em que uma árvore só irá realizar um splitting em um dos seus nós, caso a redução na função custo seja maior que γ, ou seja, ele evita que árvores possuam vários nós, diminuindo sua complexidade.
- T é o número de nós indivisíveis (folhas), nas árvores.
- λ é um fator de regularização, que define a magnitude dessa regularização e ω a saída das folhas do modelo, que para um problema de classificação binária, seriam as probabilidades de uma amostra ser positiva para cada folha.

Em conclusão, XGBoost adiciona alguns parâmetros extras a modelagem que penalizam a construção de árvores complexas, reduzindo as chances de sobreajuste.

2.5.1.2 Light Gradient Boosting Machine (LightGBM)

Criado em 2017 por pesquisadores da Microsoft(KE *et al.*, 2017) sua principal melhoria para o XGBoost é seu aumento na velocidade de treinamento, mantendo uma acurácia extremamente parecida. Para isso, duas principais implementações são feitas.

A primeira delas, é o GOSS (Gradient Based One Sided Sampling), em que o algoritmo exclui exemplares do treinamento em que o valor do Gradiente é baixo, isso

porque, gradientes altos possuem maior ganho de informação para o modelo (WANG *et al.*, 2019).

A outra técnica é conhecida como EFB (*Exclusive Feature Bundling*), nessa técnica o algoritmo faz uma seleção automática de características, excluindo características que são mutuamente exclusivas, ou seja, carregam a mesma informação e características categóricas esparsas, que costuma-se carregar muitos valores 0 e poucos valores 1. Dessa forma, como as árvores possuem sua complexidade diretamente relacionada ao número de características do conjunto de dados, o treinamento é acelerado (RIBEIRO; dos Santos Coelho, 2020).

Dessa forma, LightGBM se mostra uma boa escolha para otimização do processo de treinamento, pois os estudos mostram que mesmo com algumas mudanças no algoritmo do XGBoost, no geral, pouca variabilidade nos resultados é observada.

2.5.1.3 Categorical Boosting (CatBoost)

Proposto em 2017 por Prokhorenkova (PROKHORENKOVA *et al.*, 2019), uma das vantagens deste algoritmo é a forma com que o modelo lida com dados categóricos. Essencialmente, esses dados são transformados em numéricos, sem que o usuário do modelo precise fazer isso na etapa de pré-processamento dos dados, sua biblioteca cobre uma gama de codificação de características.

A relevância desta etapa, está no fato que um modelo de aprendizagem de máquina não processa diretamente dados categóricos, é necessário que eles sejam numéricos de alguma forma. A forma com que se processa esses dados, pode ser extremamente relevante para a modelagem, podendo ser fator determinante na redução do sobreajuste (PROKHORENKOVA *et al.*, 2019).

2.5.2 Bagging

Proposto por Breiman em 1996 (BREIMAN, 1996) o método chamado de Bootstrap Aggregating tem como premissa combinar a predição de diversos aprendizes fracos de um mesmo modelo treinados em subconjuntos aleatoriamente particionados. Ou seja, um

conjunto de dados dividido aleatoriamente em n partições, serão utilizados n aprendizes fracos para a modelagem.

Uma das maiores vantagens de se utilizar esta técnica, é conseguir reduzir a variância de uma modelagem sem aumentar seu viés, ou seja, se torna uma técnica extremamente poderosa quando utilizada com aprendizes fracos de alta variância. Desta forma, utilizar esta técnica com aprendizes fracos de árvores de decisão é extremamente vantajoso, visto que pequenas variações nos dados podem modificar consideravelmente predições feitas por uma árvore (MIENYE; SUN, 2022b).

Além disso, esses algoritmos podem ter baixo custo computacional, visto que modelos simples são treinados em pequenas partições dos dados (ALELYANI, 2021). Algumas desvantagens do algoritmo são a baixa interpretabilidade do modelo, uma vez que diversas árvores são obtidas paralelamente, é difícil dizer o que fez aquela árvore prever uma determinada classe, além de as amostras serem obtidas aleatoriamente, possuindo a chance de algumas amostras não serem utilizadas em nenhum sobconjunto. Na sequência, descreve-se um pouco sobre o algoritmo de Bagging que implementa árvores de decisão, chamado de Floresta Aleatória (*Random Forest*)

2.5.2.1 Florestas Aleatórias (Random Forest)

A ideia de se utilizar árvores de decisões em paralelo para uma tomada de decisão foi proposta inicialmente por Ho (HO, 1995) porém a ideia de utilizar subconjuntos aleatórios foi implementada apenas posteriormente, aprimorando este modelo uma vez que essa técnica reduz a correlação entre as árvores utilizadas (BREIMAN, 2001).

Como mencionado anteriormente, árvores de decisão são algoritmos que tendem a sobreajustar devida sua alta carga de variância, por isso, nas Florestas Aleatórias, as árvores são agregadas a partir de voto majoritário, ou seja, no momento de realizar uma predição, todos os aprendizes fracos realizam uma predição individualmente, e a classe com maior número de predições é definida como a predição final da Floresta, a Equação 2.17 representa o formato matemático desse agrupamento.

$$H_T(x) = argmax_j \sum_{k=1}^{K} I(h_k(x) = j) \; for \; j = 1, ..., C \tag{2.17}$$

Sendo:

- $H_T(x)$ é a classe final prevista
- K é o número de árvores utilizadas no modelo
- j é a classe prevista entre as árvores
- I é uma função indicadora que conta o número de predições para uma classe

No geral, este algoritmo performa bem quando o conjunto de dados é largo e possui diversas características, porém, quanto maior o dataset maior o número de árvores é necessário para o algoritmo performar bem, muitas árvores podem promover uma baixa velocidade na hora de se realizar uma predição (MIENYE; SUN, 2022b).

2.6 Otimização de Hiperparâmetros

Algoritmos de classificação dificilmente podem ser analisados sem considerar seus hiperparâmetros, isso porque estes algoritmos podem ter suas métricas de desempenho crucialmente afetadas ao variarmos seus hiperparâmetros. A partir disto, diversas técnicas de otimização foram desenvolvidas com o objetivo de aprimorar a busca pelo mínimo local feita na função custo (XIA *et al.*, 2017). O processo de otimização de hiperparâmetros pode ser definido como uma *black art*, já que este é avaliado a partir de desempenhos subjetivos do problema e da tentativa e erro (BERGSTRA; BENGIO, 2012).

2.6.1 Parâmetros e Hiperparâmetros

Para total compreensão do processo de otimização de hiperparâmetros, propõem-se nessa subseção entender a diferença entre Parâmetros e Hiperparâmetros.

O objetivo final de um problema de aprendizagem de máquina é encontrar uma função que mapeie o problema no formato $y = G(x, \theta)$, sendo y a saída do modelo, x sua

entrada e G a função caracterizada pelo vetor de parâmetros θ. Este vetor são parâmetros que são ajustados diretamente através da iteração de um algoritmo de treinamento, no caso das árvores o CART é usado, descrito em 2.3.3. Exemplos para esses parâmetros são os *thresholds* responsáveis por fazer a separação dos nós da árvore, como descrito anteriormente.

Hiperparâmetros são definidos como parâmetros em que seu valor não é alterado a partir dos dados de treinamento do problema (XIA *et al.*, 2017), possuem esse nome por serem considerados "alto nível"e por precisar do ajuste direto do usuário.

A seguir, são apresentados os principais hiperparâmetros característicos de Boosting, visto que no trabalho estes foram submetidos ao processo de otimização.

- Número de Estimadores (n_estimators): Número de árvores (aprendizes fracos) que são construídos ao longo do treinamento e que são agregadas entre si.
- Máxima Profundidade (max_depth): Controla a profundidade em que as árvores são construídas.
- Peso mínimo de Folha (min_data_in_leaf): Controla o número de amostras necessário para uma folha existir.
- Ganho mínimo de separação (min_split_gain): Ganho de informação mínimo para uma separação ser feita em um nó da árvore.
- Máximo de Folhas: (max_leaves): Número máximo de folhas permitidas por cada árvore.

Além disso, outros hiperparâmetros comuns de modelos de aprendizagem de máquina também fazem parte do escopo de algoritmos Boosting, como taxa de aprendizado variável, termos regularizadores e critérios de *early stopping*.

2.6.2 Busca Bayesiana

Técnicas tradicionais de otimização como a Busca em Grade (*Grid Search*) e a Busca Aleatória (*Random Search*), apesar de serem extremamente utilizadas no campo da aprendizagem de máquina, podem não ser viáveis em algoritmos de Boosting. As buscas citadas fazem um processo de tentativa e erro dentre um intervalo de hiperparâmetros definidos pelo usuário. Quando o intervalo é extenso, dentre vários hiperparâmetros, os recursos computacionais utilizados para o treinamento podem ser insuficientes (XIA *et al.*, 2017).

Para isso, será descrito a seguir uma técnica estado da arte para otimização de hiperparâmetros, a Busca Bayesiana (*Bayesian Search*) voltado ao âmbito da aprendizagem de máquina.

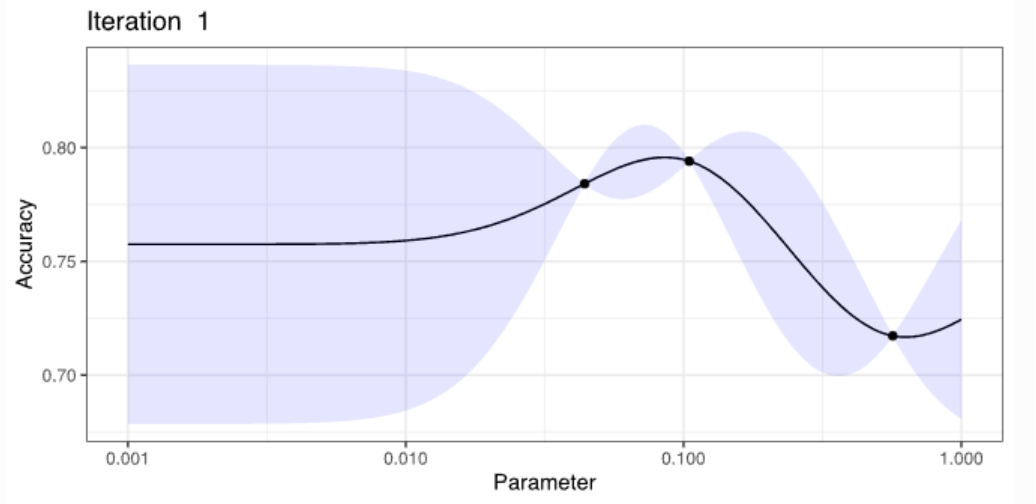
O diferencial deste método frente a outros métodos de busca está em guardar informação histórica das iterações durante o processo e utilizar isso para acelerar a convergência para um mínimo local ou global da função custo (AKIBA *et al.*, 2019). Em problemas complexos, não é possível computar com eficiência o comportamento da função custo, por isso essa técnica envolve modelar uma aproximação através de uma função probabilística, que seja mais simples de se otimizar e usar os hiperparâmetros encontrados para esta função como tentativa de busca na função custo original.

Com o avanço técnico na área de otimização, diversos algoritmos usam da lógica bayesiana para se beneficiar computacionalmente (HUTTER; HOOS; LEYTON-BROWN, 2011), neste trabalho será abordado o SMBO (Sequential Model-Based Optimization), visto é que o algoritmo principal aplicado pela ferramenta de otimização usada neste trabalho. Além disso, o exemplo a seguir toma como base (KHUN,).

Dada uma função objetivo $y = f(x)$, mas que se desconhece seu comportamento exato ao longo de todo o intervalo, e um único hiperparâmetro x, calcula-se $f(x_1)$, $f(x_2)$, $f(x_3)$. Em seguida, a partir dos valores encontrados utiliza-se de uma função probabilística chamada Processo Gaussiano $P(y \mid x)$ para aproximar o comportamento de $f(x)$. Esta técnica retorna uma função substituta à função objetivo, possuindo um grau de incerteza

a partir dos valores $f(x)$ computados. A figura 12 descreve este processo.

Figura 12 – Função substituta dada por um Processo Gaussiano

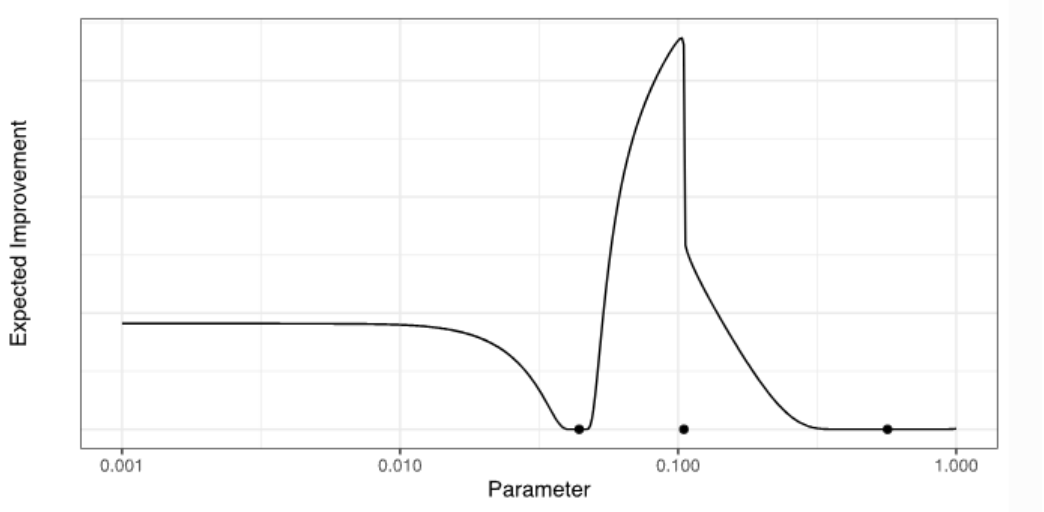


Fonte:(KHUN,)

É possível ver que o grau de incerteza próximo aos pontos computados tende a 0, e conforme se afasta este grau aumenta proporcionalmente.

A próxima etapa do algoritmo se dá por calcular a função de aquisição, neste exemplo utiliza-se a *Expected Improvement*, seu objetivo é encontrar a região do espaço onde dado x_n, $f(x_n)$ terá mais chance de se aproximar do seu valor alvo, sendo aqui, a maximização. A figura 13 representa esta etapa.

Figura 13 – Função de Aquisição



Fonte:(KHUN,)

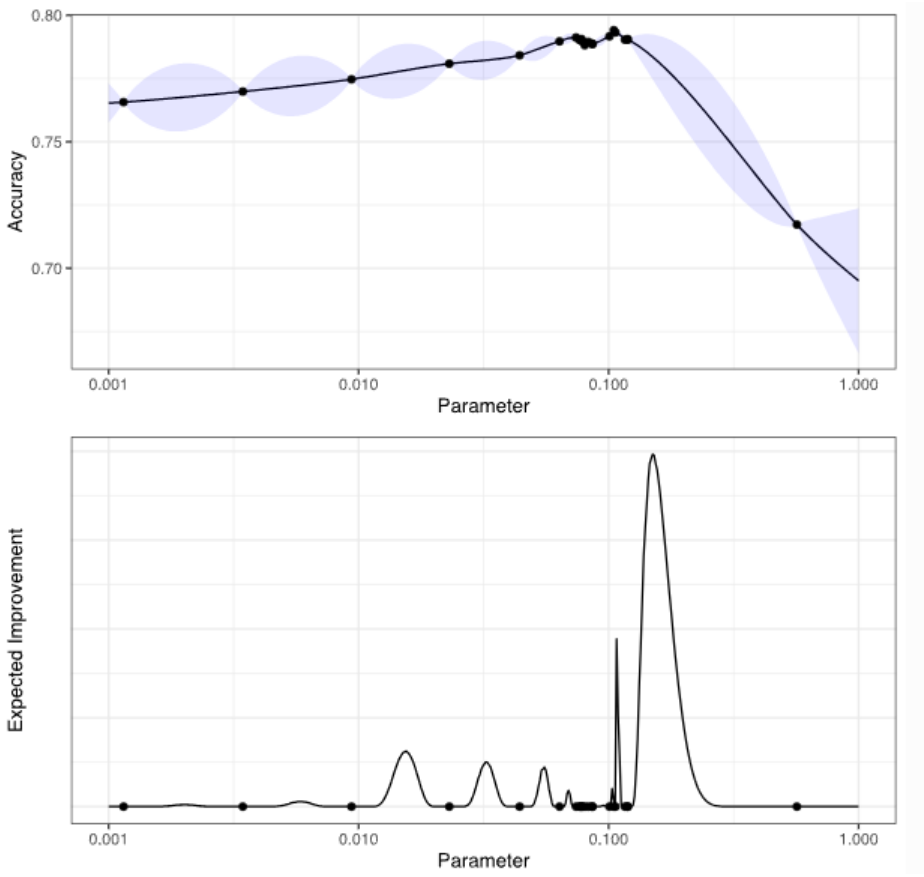
O valor da função aquisição tende a ser alto para uma região entre dois pontos que se aproximam de um máximo, assim como, a medida que o grau de incerteza dado pelo Processo Gaussiano aumenta ela também tende a aumentar.

Por fim, calcula-se com base nas regiões de maior *Expected Improvement* o próximo x a ser utilizado para se calcular $f(x)$. A figura 14 mostra o comportamento das funções substituta e de aquisição para 20 pontos de exemplo.

Em um processo iterativo, a melhor forma de convergir seria encontrando o x em que a função de aquisição é máxima, visto ser uma função conhecida pelo algoritmo, e utilizando esse valor de x como próximo hiperparâmetro a ser testado na função objetivo real.

Ao longo da seção buscou-se contextualizar a importância de métodos de busca dentro do contexto de aprendizagem de máquina, principalmente a busca bayesiana dado

Figura 14 – Função substituta e função de aquisição para 20 pontos



Fonte:(KHUN,)

sua posição de algoritmo estado da arte. O intuito principal foi trazer uma visualização gráfica do funcionamento do algoritmo SMBO para entendimento da aplicação do framework Optuna, utilizado em 5.2.

2.7 Métricas de Desempenho para Modelos de Classificação

Um modelo de classificação binário prevê a probabilidade de um evento ter o rótulo 0 ou 1, e um valor de *threshold* arredonda essa probabilidade para a variável predita. Ou seja, se o modelo retorna a probabilidade que em uma determinada previsão ele terá rótulo 1, avalia-se se aquela probabilidade está acima do valor de *threshold*, se estiver, a previsão final do modelo é 1, se não estiver a previsão final é 0 (LIU *et al.*, 2014).

A partir deste fator e dos acertos e erros apresentados pelo modelo, é possível analisar uma série de métricas que indicam o quanto uma modelagem de fato desempenha

em cima do problema proposto. A seguir serão apresentadas as principais.

2.7.1 Acurácia

A acurácia, também chamada de taxa de acerto, mede a quantidade de previsões corretas, sejam elas 0 ou 1 dada pelo modelo em relação ao total de amostras disponíveis. Sua definição matemática é dada na equação 2.18.

$$Acc = \frac{TP + TN}{N} \tag{2.18}$$

- TP (*True Positives*) - Quantidade de amostras que observou-se Verdadeiro e o modelo previu Verdadeiro.
- TN (*True Negatives*) - Quantidade de amostras que observou-se Falso e o modelo previu Falso (Verdadeiros Negativos).
- N - Quantidade total de Previsões feitas.

A acurácia é a métrica mais simples e de mais fácil interpretabilidade para problemas de classificação. Porém ela não ressalta os tipos de erro que o modelo comete, por isso ela pode não ser suficiente para entender a performance do seu modelo (GÉRON, 2019h). Além disso, ela pode ser extremamente tendenciosa para ocorrência de dados desbalanceados, ou seja, que uma das classes está muito mais presente que a outra no conjunto de dados.

2.7.2 Precisão

A precisão é uma medida que calcula a eficiência do modelo em identificar casos positivos (classe 1), isso porque ela nos dá a quantidade de verdadeiros positivos encontrados dividido pela soma de verdadeiros positivos com falsos positivos (PEDREGOSA *et al.*, 2011b), A equação 2.19 descreve esta métrica:

$$Precision = \frac{TP}{TP + FP} \tag{2.19}$$

- TP (*True Positives*) - Quantidade de amostras que observou-se Verdadeiro e o modelo previu Verdadeiro.
- FP (*False Positives*) - Quantidade de amostras que observou-se Falso e o modelo previu Verdadeiro.

Ao tentar maximizar essa medida, obtém-se um modelo que se preocupa em prever classe 1 com mais segurança, evitando casos onde ele prevê 1 quando deveria ser 0. Essa métrica é útil quando erros de falso positivo tem um grande impacto no problema, e devem ser evitados.

2.7.3 Revocação

Análogo a precisão, a revocação nos diz a eficiência do modelo em evitar casos de falso negativo, isso porque ela mede a quantidade de verdadeiros positivo encontrados dividido pela soma de verdadeiros positivo com falso negativo. A equação 2.20 a representa abaixo:

$$Recall = \frac{TP}{TP + FN} \tag{2.20}$$

- TP (*True Positives*) - Quantidade de amostras que observou-se Verdadeiro e o modelo previu Verdadeiro.
- FN (*False Negatives*) - Quantidade de amostras que observou-se Verdadeiro e o modelo previu Negativo.

Por mais de não se ter acesso direto à taxa de verdadeiro negativo, ao maximizar essa métrica levamos a taxa de falsos negativo para 0, garantindo que todas as previsões de 0 estejam corretas, dessa forma, evita-se prever a classe 0 para uma amostra que possui classe 1.

2.7.4 F1 - Score

Com a importância em entender os diferentes pesos entre os erros que um modelo pode cometer é pertinente avaliar a precisão e a revocação a partir de uma única métrica, o Score F1 trabalha com a média harmônica de ambos (GÉRON, 2019i), a equação 2.21 representa essa métrica.

$$F_1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{TP}{TP + \frac{FN+FP}{2}} \tag{2.21}$$

Na prática, o F1-Score carrega a informação do modelo ser balanceado nas métricas de Revocação e Precisão, pois a média harmônica garante que caso ocorra um desbalanceamento entre ambas métricas, o F1-Score tenda a 0.

Nesta seção até aqui foram apresentados métricas numéricas que representam a desempenho do modelo. A seguir serão apresentadas representações gráficas que complementam as análises de desempenho.

2.7.5 Matriz de Confusão

Uma das principais ferramentas para avaliar a performance do modelo de forma gráfica é a Matriz de Confusão. Ao se avaliar os possíveis resultados reais em um eixo e os preditos em outro, criam-se 4 quadrantes de ocorrências. Verdadeiro Positivo, Verdadeiro Negativo, Falso Positivo e Falso Negativo (GÉRON, 2019j). A figura 15 mostra um exemplar.

Dessa forma, analisa-se que os valores da diagonal principal da matriz de confusão representam os acertos do modelo e a diagonal secundária os erros de previsão.

2.7.6 Curva Precisão-Revocação

Mesmo que seja possível controlar um dos tipos de erros causados pelo modelo a partir da variação do *threshold* de decisão, não é possível diminuir o número de erros totais (GÉRON, 2019k). Esse fenômeno é conhecido como *Tradeoff* Precisão-Revocação.

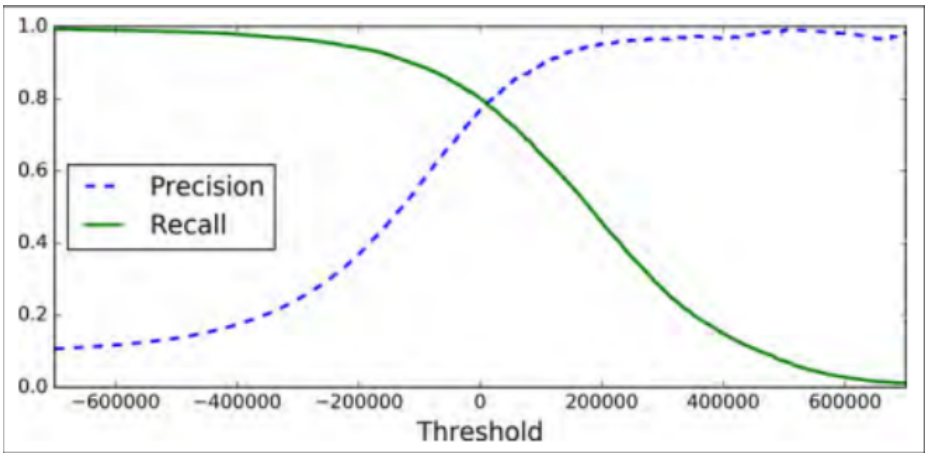
Figura 15 – Matriz de Confusão

predicted class	*actual class* positive	negative
positive	**TP** True Positives (*a*)	**FP** False Positives (*b*)
negative	**FN** False Negatives (*c*)	**TN** True Negatives (*d*)

Fonte:(LOVELL *et al.*, 2022)

Este Tradeoff diz que a precisão e a Revocação são variáveis inversamente proporcionais, e ao variar o *threshold* com o intuito de aumentar um maximizar, obrigatoriamente minimiza o outro. A Figura 16 representa esta relação.

Figura 16 – *Tradeoff* Precisão e Revocação



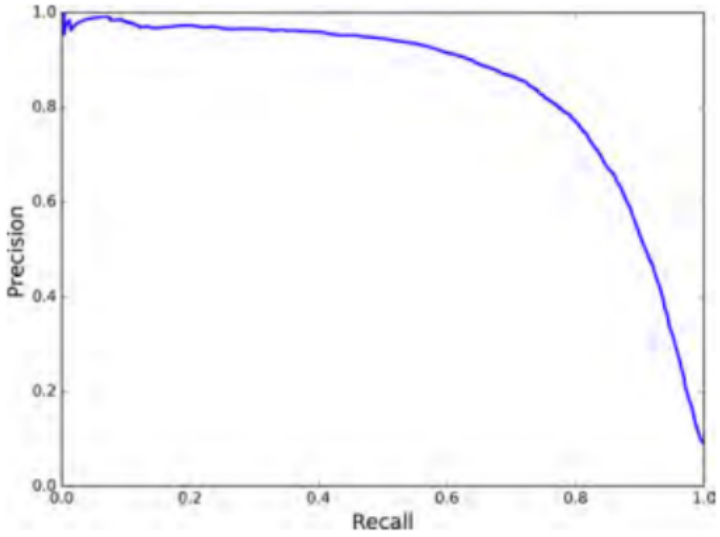
Fonte:(GÉRON, 2019i)

A relação entre a Precisão e a Revocação também pode ser visualizada através da Curva Precisão-Revocação, esta visualização facilita o processo de decisão de *threshold*, já que é possível compreender a relação de ambas variáveis para todos os valores do limiar de decisão, a Figura 17 exemplifica a curva.

2.7.7 Curva ROC

Na Prática, a curva ROC desempenha um papel extremamente parecido com a Curva PR, a diferença é que ao invés de relacionar a Precisão e Revocação, aqui avalia-se

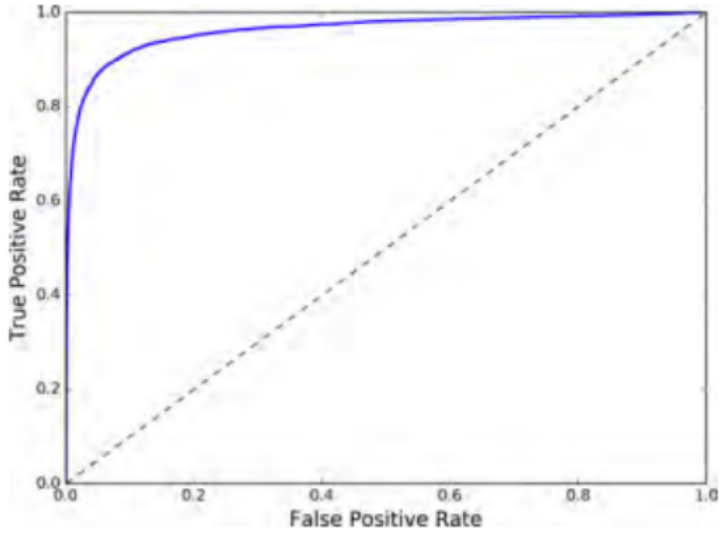
Figura 17 – Curva Precisão Revocação



Fonte:(GÉRON, 2019i)

a taxa de verdadeiros positivo (Revocação) com a taxa de falso positivo, a Figura 18 a exemplifica.

Figura 18 – Curva ROC



Fonte:(GÉRON, 2019l)

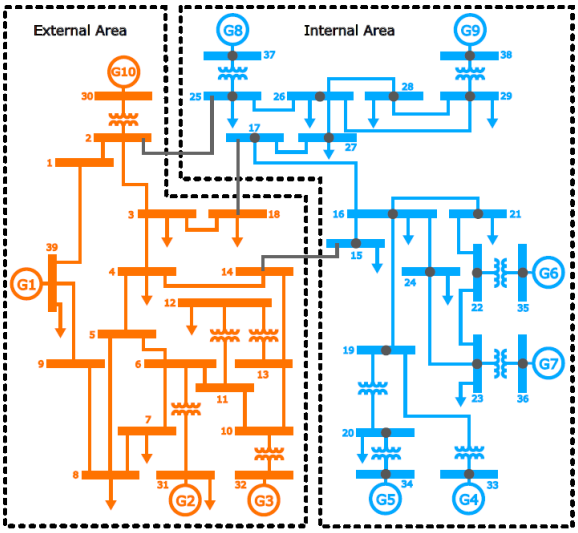
Na Figura 18, entende-se como modelo ideal um modelo localizado no ponto (0,1) onde se minimiza a taxa de Falso Positivo e maximiza a taxa de Verdadeiro Positivo. A curva ROC é preferível às demais representações gráficas porque é mais fácil entender os conceitos de sobreajuste e subajuste através dela. Porque para um modelo aleatório que não obedece qualquer critério ao realizar sua predição terá sua curva próximo a linha pontilhada, indicando um modelo subajustado.

3 Localização de Falta utilizando Redes Neurais

Este capítulo tem como objetivo destacar um artigo desenvolvido previamente e que foi usado como fundamental base teórica e prática. Como comentado nos objetivos específicos, o trabalho aqui desenvolvido analisa o desempenho dos modelos de Árvore de Decisão frente as Redes Neurais para o problema de localização de falta no Sistema de Potência. Para fator comparativo, os resultados obtidos através das Redes Neurais em *Accurate Internal/External Area Under Fault Classifier applying Neural Networks to PMU Data* (SCHUMACHER *et al.*, 2023) serão destacados nesse capítulo. Assim como este trabalho, o artigo mencionado tem como objetivo analisar que se um Sistema de Potência está sob condições de falta, as variações causadas no seus fasores medidos através de PMU's carregam informação suficiente para deduzir qual é a área afetada do Sistema, através da aplicação de Redes Neurais.

Para isto, utilizou-se o sistema de barras IEEE-39, e dividiu-se em duas áreas, apelidadas de Área Interna (*Internal Area*) e Área Externa (*External Area*). a Figura 19 representa esta divisão.

Figura 19 – Área Externa e Área Interna para IEEE 39 barras



Fonte:(SCHUMACHER *et al.*, 2023)

Para treinar um modelo de aprendizado profundo, o trabalho considerou que os fasores do sistema são medidos através de PMU's nas suas barras, e que são sincronizados e ordenados no tempo, de forma que seja possível avaliar graficamente seus comportamentos.

Para coleta de dados, utilizando o software ANATEM, o sistema foi simulado sob a condição de diferentes faltas em diferentes barras do sistema, a Tabela 4 registra o número de simulações feitas para cada tipo de falta.

Tabela 4 – Simulação de Eventos de falta para IEEE 39 barras

Evento	**Abreviação**	**Número de Simulações**
Curto Monofásico em Barra	APCC	39
Curto Circuito Trifásico em Linha CA	APCL	31
Abertura total de Linha	ABCI	43
Curto Circuito Trifásico em Barra	APCB	39
Desligamento de Barra	DBCA	39

Em seguida, foi utilizado o MATLAB para o processo de Engenharia de Características (*Feature Engineering*). Foram selecionados os fasores de frequência e tensão em cada uma das barras e as correntes em cada uma das linhas. Utilizando 3 conjuntos de dados diferentes, fasores em formato polar, fasores em formato retangular e a diferença entre os fasores no instante de tempo t e $t-1$ em formato retangular. As equações 3.1, 3.2 e 3.3 representam respectivamente esses conjuntos.

$$\vec{x} = [V \;\; \theta_V \;\; I \;\; \phi_I \;\; f] \tag{3.1}$$

$$\vec{x} = [V^{real} \;\; V^{imag} \;\; I^{real} \;\; I^{imag} \;\; f] \tag{3.2}$$

$$\vec{x} = [\Delta V^{real} \;\; \Delta V^{imag} \;\; \Delta I^{real} \;\; \Delta I^{imag} \;\; \Delta f] \tag{3.3}$$

O processo de treinamento do modelo foi feito utilizando 100 amostras de cada simulação de falta. O conjunto total foi dividido em 80% amostras de treinamento e 20% amostras de validação. O conjunto de validação foi utilizado para encontrar as métricas de

performance. É importante ressaltar que a falta é aplicada no instante de tempo $t = 0$, dessa forma, garante-se que as amostras selecionadas para o treinamento representam o transitório da rede durante uma falta. Além disso, foram considerados apenas os fasores das PMU's posicionadas na área interna.

Por fim, os modelos foram treinados 20 épocas de treinamento, o otimizador Adam (KINGMA; BA, 2017), e uma taxa de aprendizado $\alpha = 0.001$, sem regularização, pré-processamento através de uma normalização por Z-score e a seleção dos hiperparâmetros da rede foram adquiridas através de um processo exaustivo de treinamento.

A tabela 5, mostra um aumento de desempenho da classificação uma vez que uma maior quantidade de amostras são fornecidas e com maior diversidade de faltas, avaliado pela métrica Acurácia em validação cruzada.

Tabela 5 – Acurácia em validação cruzada por diversidade do conjunto de treinamento

Eventos do treinamento	**Acurácia**
APCC	$82.40\% \pm 8.85$
APCC+APCL	$86.32\% \pm 5.23$
APCC+APCL+ABCI	$83.23\% \pm 5.00$
APCC+APCL+ABCI+APCB	$87.73\% \pm 3.23$
APCC+APCL+ABCI+APCB+DBCA	$88.24\% \pm 3.42$

É importante ressaltar que a tabela 5 adota os fasores em seu formato polar, como representa a equação 3.1, a tabela 6 compara o resultado da classificação para os diferentes formato dos fasores. Demonstrando a superioridade em que o formato retangular tem de carregar informação ao modelo, principalmente levando em consideração o seu valor em delta.

Tabela 6 – Acurácia por formato do Fasor

Formato dos fasores	**Acurácia**
Polar	$88.24\% \pm 3.42$
Retangular	$91.12\% \pm 3.05$
Δ Retangular	$92.00\% \pm 2.50$

O artigo propôs um modelo de rede neural para concessionárias de Geração e Transmissão (G&T) que detecta falhas usando dados fasorias coletados por PMUs, os

dados fornecidos ao modelo são apenas da área interna e atingiu 92% de acurácia na validação cruzada ao incluir características temporais de diferentes eventos de falta.

4 Materiais e Métodos

Neste capítulo serão apresentados os materiais e métodos necessários para o desenvolvimento da proposta de trabalho. Dentre os materiais constam os equipamentos utilizados como recursos computacionais, linguagens de programação e suas bibliotecas e softwares auxiliares. Quanto a metodologia destaca-se o passo a passo utilizado para realização da leitura do arquivo que contém as simulações integralmente com Python e também os pontos de atenção tomados durante o processo de desenvolvimento do modelo de aprendizagem de máquina.

4.1 Materiais

4.1.1 Recursos de Hardware

Durante o desenvolvimento de todo o trabalho, foi utilizado um computador comercial, com processador AMD Ryzen 5 1600 @ 3.20 GHz, contendo 6 Núcleos e memória RAM 16 GB. Sistema operacional Windows 10, versão 22H2.

4.1.2 Python

Python é a principal linguagem utilizada para abordar a problemática neste trabalho. Considerada a linguagem mais utilizada no mundo, Python é uma linguagem de código aberto, de alto nível, eficiente e fácil interpretação, disponível em Windows, MacOS e Linux. Sua principal vantagem é a vasta gama de bibliotecas que a tornam extremamente versátil. No tópico deste trabalho, detém-se diversas bibliotecas que auxiliam na manipulação de dados, construção de gráficos e implementação de APIs para algoritmos de aprendizagem de máquina. O pacote utilizado no desenvolvimento deste trabalho é o 3.12.1, e as bibliotecas foram instaladas através do seu gerenciador de pacotes chamado de pip.

4.1.3 Bibliotecas

A Sub-seção a seguir destaca as bibliotecas utilizadas neste trabalho, bem como sua função dentro do desenvolvimento do mesmo.

4.1.3.1 Processamento de Dados

- Scipy: Biblioteca matemática que possui módulos para aplicações estatísticas, otimização, Álgebra Linear, processamento de sinais, dentre outros (VIRTANEN *et al.*, 2020). Neste trabalho utilizou-se o módulo "io"para leituras de arquivos .mat em um vetor multidimensional.

- Numpy: Biblioteca especializada em trabalhar com vetores e operações entre vetores de forma extremamente eficiente (HARRIS *et al.*, 2020). Neste trabalho, foi utilizada para manipular vetores de alta dimensionalidade que armazenavam dados das simulações desenvolvidas.

- Pandas: Biblioteca voltada para trabalhar com conjunto de dados em formato relacional e tabular, de forma rápida e eficiente, sua eficiência se dá principalmente por carregar os dados em RAM. (TEAM, 2020) Neste trabalho foi utilizada para manipulação de dados, engenharia de características e versionamento de tabelas por ter a capacidade de processar arquivos .txt e .csv.

4.1.3.2 Visualização de Dados

- Matplotlib: Biblioteca para desenvolvimento de gráficos estáticos e animados, em conjunto com visualizações matemáticas interativas. Os gráficos utilizados neste trabalho foram todos gerados a partir de sua API chamada de pyplot.

4.1.3.3 Aprendizagem de Máquina

- Scikit-learn: Biblioteca para diversas aplicações de aprendizado de máquina, utilização de algoritmos, cálculo de métricas e processamento de dados (PEDREGOSA *et al.*, 2011c). Neste trabalho foi utilizado para a aplicação dos Algoritmos de Árvore de

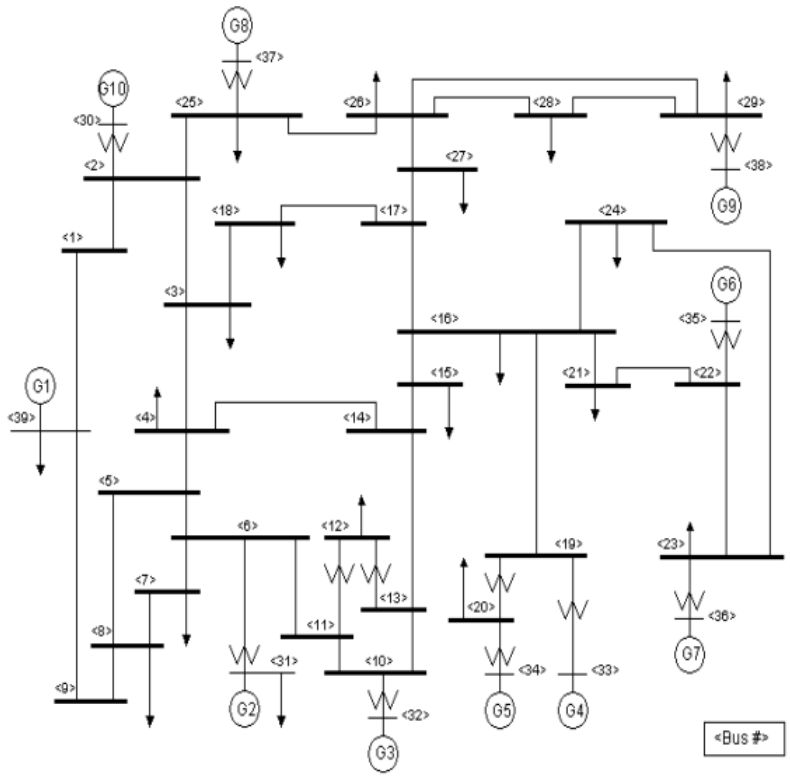
Decisão e Florestas Aleatórias. Também foi utilizada para realizar o cálculo de métricas de validação de modelos.

- XGBoost: Biblioteca voltada para utilização em alto nível do algoritmo Extreme Gradient Boosting. Foi utilizada para este fim no trabalho.
- LightGBM: Biblioteca voltada para utilização em alto nível do algoritmo Light Gradient Boosting Machine. Foi utilizada para este fim no trabalho.
- CatBoost: Biblioteca voltada para utilização em alto nível do algoritmo Categorical Boosting. Foi utilizada para este fim no trabalho.
- Optuna: Biblioteca voltada para aplicar algoritmos de otimização (AKIBA *et al.*, 2019). Foi utilizada para otimização dos hiperparâmetros dos algoritmos de aprendizagem de máquina.

4.1.4 Sistemas de Potência

- IEEE-39 Barras. É um sistema considerado como *benchmark* para diversos testes de aplicações em sistemas elétricos de potência, a Figura 20 representa sua topologia. Neste trabalho foi utilizado como objeto de estudo e validação para aplicação do algoritmo de aprendizado de máquina.

Figura 20 – IEEE-39 Bus System



Fonte:(SEREM; LETTING; MUNDA, 2021)

- Anatem. O programa é um aplicativo para simulações dinâmicas no domínio do tempo, visando a análise não linear de transitórios eletromecânicos de sistemas de potência de grande porte, compreendendo os períodos de estabilidade transitória e dinâmica (CEPEL, 2024). Os dados de corrente, tensão e frequência analisados neste trabalho foram gerados através de simulações no Anatem.

- IEEE Std C37.118. Protocolo desenvolvido pelo IEEE, com sua última atualização em 2011 que tem o objetivo de padronizar e regularizar as condições de medição de fasores sincronizados da rede, bem como, a comunicação de diferentes dispositivos que trabalham com sincrofasores. As informações apresentadas em 2.2 tomam como base este protocolo.

4.2 Métodos

4.2.1 Motivação

O desenvolvimento do presente trabalho tem como objetivo ser um estudo sequencial ao artigo apresentado em 3, para garantir a comparabilidade da análise buscou-se utilizar o conjunto de dados coletados pela mesma simulação, por isso, a descrição do conjunto de dados se encontra na seção mencionada. A seguir, serão apresentados resultados e análises que complementam a capacidade de se obter informações valiosas de Sistemas de Potência e seu funcionamento a partir do uso de aprendizagem de máquina em fasores coletados por PMU's.

4.2.2 Leitura e Análise de Dados utilizando Python

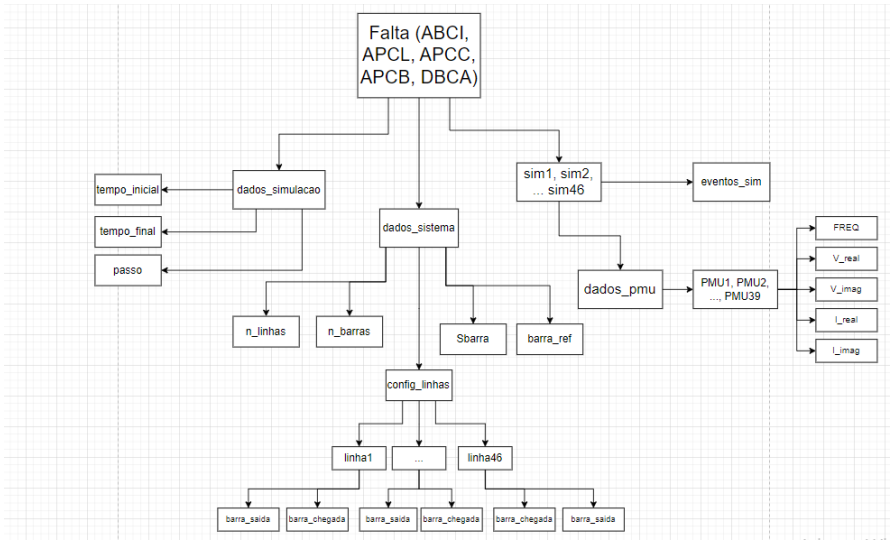
Para o trabalho, optou-se centralizar todo seu desenvolvimento em Python, visto sua eficiência em trabalhos que usam processamento e análise de dados, além de ser código aberto. Por isto, o foco inicial do trabalho foi em processar os arquivos .mat exportados pelo Anatem para garantir sua manipulação 100% dentro do Python.

Como Python é uma linguagem orientada a objetos, A leitura do arquivo de simulação foi realizada utilizando a biblioteca Scipy, nela, importamos a classe "Scipy.io"e utilizamos o método ".loadmat"que converte a estrutura de dados .mat em um vetor multidimensional aninhado. Este vetor é simplificado em formato de diagrama na figura 21.

Para acessar os dados dentro do vetor multidimensional, foi desenvolvida um script que utiliza um endereçamento simples. Por exemplo, utilizando a Figura 21 como base é possível acessar a frequência da PMU da barra 1, da simulação 1 para uma falta do tipo ABCI através do seguinte endereçamento: "['ABCI', 'sim1', 'dadosPMU', 'PMU1', 'FREQ']".

Este enderençamento retorna um vetor de formato (2003, 1). Sendo 2003 amostras da frequência para essa simulação. Desta forma, é possível acessar qualquer variável de qualquer simulação, PMU e falta do estudo.

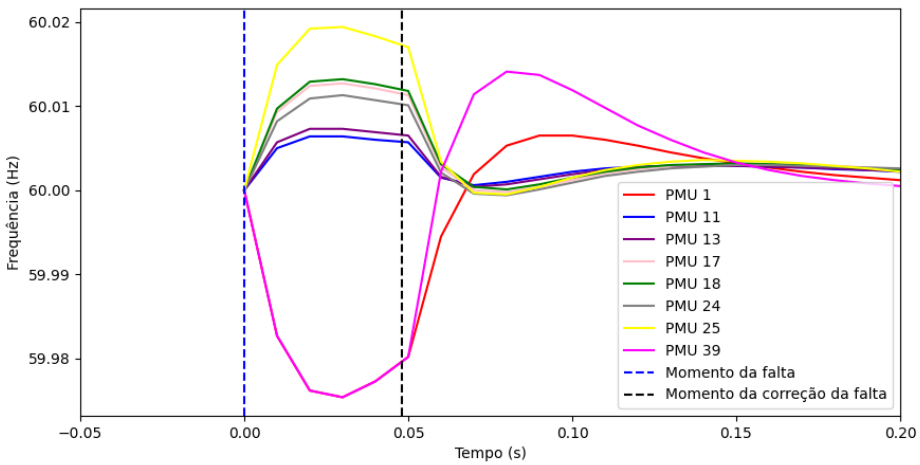
Figura 21 – Estrutura de Dados de simulação



Fonte: O Autor

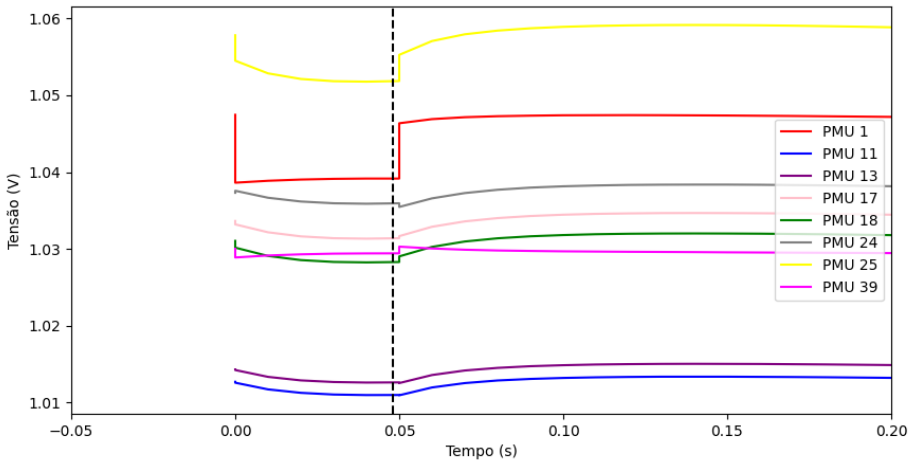
Para entender o comportamento das variáveis elétricas alocadas no sistema, utiliza-se de ferramentas gráficas, buscou-se ter uma visualização do comportamento da frequência, tensão e corrente em cada um dos tipos de faltas. Visualiza-se barras adjacentes a falta, barras relativamente afastadas da falta e barras totalmente afastadas da falta. desta forma, as figuras 22 e 23 trazem um exemplo de comportamento de diferentes PMU's durante uma falta na PMU 1.

Figura 22 – Comportamento da Frequência em diferentes barras do sistema para a falta ABCI



Fonte: O Autor

Figura 23 – Comportamento da Tensão em diferentes barras do sistema para a falta ABCI



Fonte: O Autor

Nas figuras, as PMU's que medem uma variação de maior intensidade são as PMU's 1 e 39, dessa forma, analisando a topologia do sistema, é possível deduzir uma relação direta entre a proximidade do local de falta e a intensidade da variação da falta. Todavia, por mais que a correlação possa existir, extrair parâmetros que possam representar o local da falha para diferentes faltas, não é trivial. Por isso, uma modelagem de aprendizagem de máquina é extremamente válida para coleta desses parâmetros em um modelo estatístico.

4.2.3 Modelagem

A etapa da construção de um modelo baseado em árvores utilizou dos 5 modelos descritos em 2.4, para realização desta tarefa foram utilizadas as bibliotecas 4.1.3.3, tornando a aplicação de tais algoritmos bastante intuitiva.

A etapa de modelagem foi aplicada utilizando o conjunto de dados em formato retangular e o conjunto de dados em formato retangular Δ. O formato polar foi despriorizado devido a sua perda de desempenho durante a modelagem.

Para garantir um score que represente um modelo generalizável, utilizou-se do método de validação cruzada em que 80% dos dados são utilizados para treinamento do modelo e 20% para sua validação. As amostras do conjunto de dados são divididas em simulações, em que cada simulação possui 100 amostras, para garantir que no processo de

validação cruzada uma mesma simulação fosse alocada tanto nos dados de treinamento quanto nos dados de validação, criou-se uma função de validação cruzada própria, evitando o processo de vazamento de dados de validação.

Esta função escolhe aleatoriamente simulações de cada tipo de falta dentro da proporção 80/20, a tabela 7 demonstra essa proporção referente ao número total de cada simulação.

Tabela 7 – Divisão de Simulações de treino e validação

Tipo de Falta	**Simulações de Treinamento**	**Simulações de Validação**
APCC	29	8
APCL	31	8
ABCI	34	9
APCB	31	8
DBCA	31	8

Com essa prática, a métrica de validação do modelo acaba sendo a média simples da métrica de cada modelo de cada iteração da validação cruzada. Dessa forma exige-se que sejam treinados n modelos para uma validação cruzada de tamanho n o custo computacional pode ser bem alto, adotou então $n = 30$ validações para cada modelo.

Em seguida buscou-se executar um processo de otimização dos modelos ensemble, visto que esses apresentam melhores resultados que árvores simples, para isso foi utilizado o framework Optuna, que aplica a otimização por Busca Bayesiana. O custo computacional da otimização pode ser extremamente elevado, visto que a ideia é retreinar o modelo várias vezes a partir de pequenas mudanças nos seus hiperparâmetros. Quando esta técnica é aplicada com a validação cruzada atingimos um custo de $m.n$ sendo n o número de validações cruzadas e m o número de treinamentos feitos no processo de otimização.

Decorrente deste fator, reduziu-se o número de validações cruzadas para 5 por iteração da otimização. E a quantidade de iterações da otimização foi adotada com base no tempo de treinamento do modelo, seguindo a Tabela 8.

em 5 são abordados o número de iterações alcançadas pelo processo de otimização.

Durante a etapa de modelagem, não foram utilizados as colunas de número da

Tabela 8 – Divisão de Simulações de treino e validação

Modelo	**Número Máximo de Buscas Bayesianas**
XGBoost	250
LightGBM	250
CatBoost	80

simulação e tipo da falta, pois em uma aplicação real estas não seriam informações acessíveis para um modelo ligado a rede. Porém, para promover análises sobre o desempenho do modelo, estas foram preservadas e adicionadas aos conjuntos de validação após a previsão.

5 Resultados Numéricos

Este capítulo tem como principal foco trazer os resultados atingidos durante a etapa de modelagem, assim como possíveis análises a serem feitas com base no embasamento teórico apresentado em 2.7. Essas análises tem como objetivo elucidar ações e percepções a serem tomadas frente tais métricas de desempenho do modelo.

5.1 Resultado 1 - Acurácia e F1-Score

Dentro do pré-processamento, características que relacionem temporalmente diferentes amostras como implementação de características *lags* e a média móvel de partições do conjunto de dados foram aplicadas mas ocasionaram uma piora no desempenho dos modelos.

A tabela 9 traz o resultado dos modelos em seu formato padrão estabelecido pela sua biblioteca, sem alteração nos hiperparâmetros por parte do usuário, no conjunto retangular.

Tabela 9 – Resultados no conjunto de dados retangular

Modelo	**Acurácia**	**F1-Score**	**Tempo de Treinamento (s)**
Árvore de Decisão	$76,75 \pm 4,5\%$	$76,58 \pm 5,0\%$	$8,40$
Floresta Aleatória	$85,55 \pm 4,2\%$	$85,78 \pm 4,0\%$	$46,63$
XGBoost	$87,27 \pm 4,3\%$	$87,26 \pm 4,1\%$	$2,82$
LightGBM	$87,35 \pm 4,5\%$	$87,35 \pm 4,5\%$	1.35
CatBoost	$87,92 \pm 4,2\%$	$87,80 \pm 4,2\%$	$31,31$

A tabela 10 traz o resultado dos modelos em seu formato padrão estabelecido pela sua biblioteca, sem alteração nos hiperparâmetros por parte do usuário, no conjunto retangular Δ.

Tabela 10 – Resultados no conjunto de dados retangular Δ

Modelo	**Acurácia**	**F1-Score**	**Tempo de Treinamento (s)**
Árvore de Decisão	$82,39 \pm 3,0\%$	$81,34 \pm 3,5\%$	$7,24$
Floresta Aleatória	$89,16 \pm 3,5\%$	$88,58 \pm 4,1\%$	$41,59$
XGBoost	$90,12 \pm 3,2\%$	$89,53 \pm 3,7\%$	$2,75$
LightGBM	$89,94 \pm 3,1\%$	$89,36 \pm 3,6\%$	1.2
CatBoost	$90,49 \pm 3,1\%$	$89,86 \pm 3,7\%$	$30,15$

Os resultados são condizentes com a teoria ao mostrar que modelos ensemble tem melhor desempenho que árvores sozinhas. Além disso, foi analisado o desempenho dos modelos ensemble nos próprios dados de treino, como forma de analisar a quantidade de viés e variância do modelo. Todos os modelos tiveram um desempenho de 99% de acurácia nos dados de treino, isto mostra que o modelo possui uma maior quantidade de variância decorrente da sua complexidade. Por isso, conclui-se que uma forma de aumentar o desempenho do modelo deve levar em consideração estratégias que vão além de trabalhar em cima de sua configuração, como etapas de pós-processamento ou aumentar o conjunto de dados disponível para treinamento.

5.2 Resultado 2 - Otimização de hiperparâmetros com Busca Bayesiana

A tabela 11 mostra os resultados da aplicação da Busca Bayesiana, decorrente do custo computacional optou-se por restringir o processo apenas aos modelos de Boosting para este estudo.

Tabela 11 – Otimização de hiperparâmetros no conjunto retangular Δ

Modelo	**F1-Score**	**Tempo de Otimização (min)**	**Número de buscas**
XGBoost	$87,93 \pm 3,5\%$	20:05	245
CatBoost	$90,10 \pm 2,3\%$	164:15	123
LightGBM	$91,91 \pm 2,42\%$	20:01	237

É possível concluir que para este processo, apenas o LightGBM foi o modelo com desempenho positivo na otimização, já que melhorou a acurácia geral do modelo. É importante ressaltar que o processo de busca possui restrições computacionais, isto é, foi necessário pré-definir um espaço de busca que não aumentasse a complexidade do modelo além do que a máquina pode processar em tempo viável, isto estabelece uma relação direta entre o poder computacional e o desempenho de modelos durante o processo de sua construção.

Os resultados gráficos apresentados nas seções a seguir foram executados utilizando diferentes modelos ensemble. Além disso foi selecionado uma amostra aleatória dentre as diferentes amostras de validações cruzadas nas etapas anteriores.

5.3 Resultado 3 - Desempenho do modelo por tipo de falta

Ao adicionar o tipo de falha às previsões sem que de fato o modelo fosse treinado por isso possibilita realizar uma análise de desempenho por tipo de falha. A figura 24 mostra a % de erro por tipo de falta e a figura 25 consequentemente mostra a performance do modelo por tipo de falta.

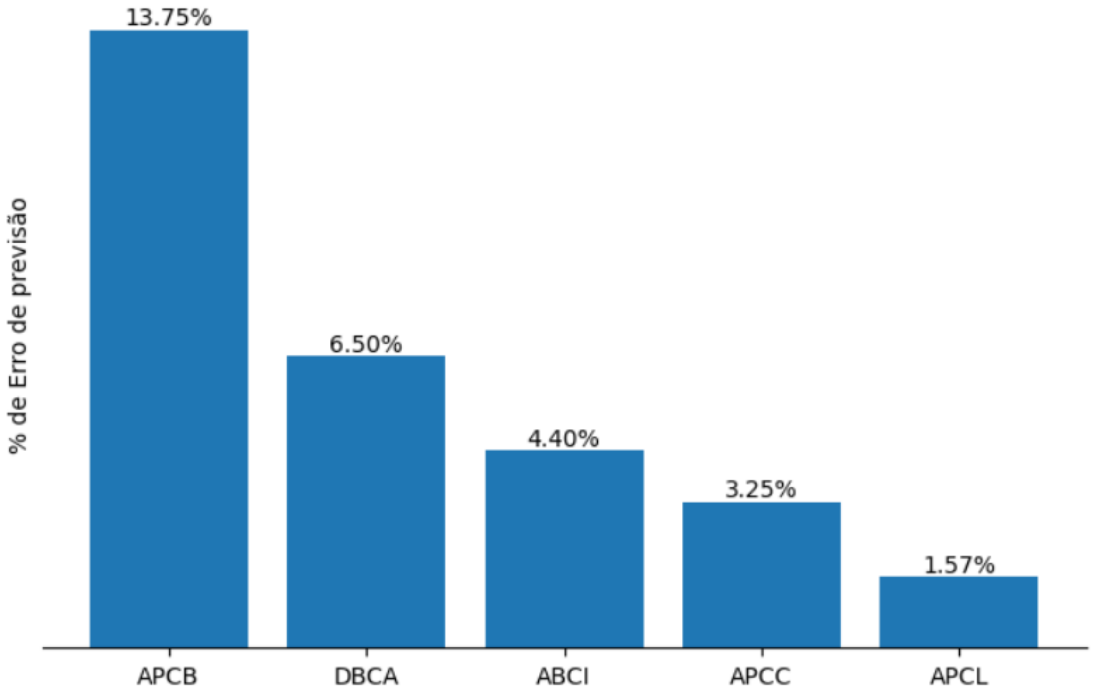
A modelagem do resultado 3 possui um F1-Score de aproximadamente 94%. Ambas figuras mostram de forma clara como o tipo de falha impacta na capacidade de previsão do modelo, e que o modelo não performa igualmente para todas as falhas.

5.4 Resultado 4 - Desempenho por quantidade de PMU

Apesar de ser consolidado no mercado, montar uma rede de monitoramento utilizando PMU's pode ser muito custoso, decorrente disto, buscou-se realizar um estudo de performance do modelo a partir do número de PMUs implementadas no sistema.

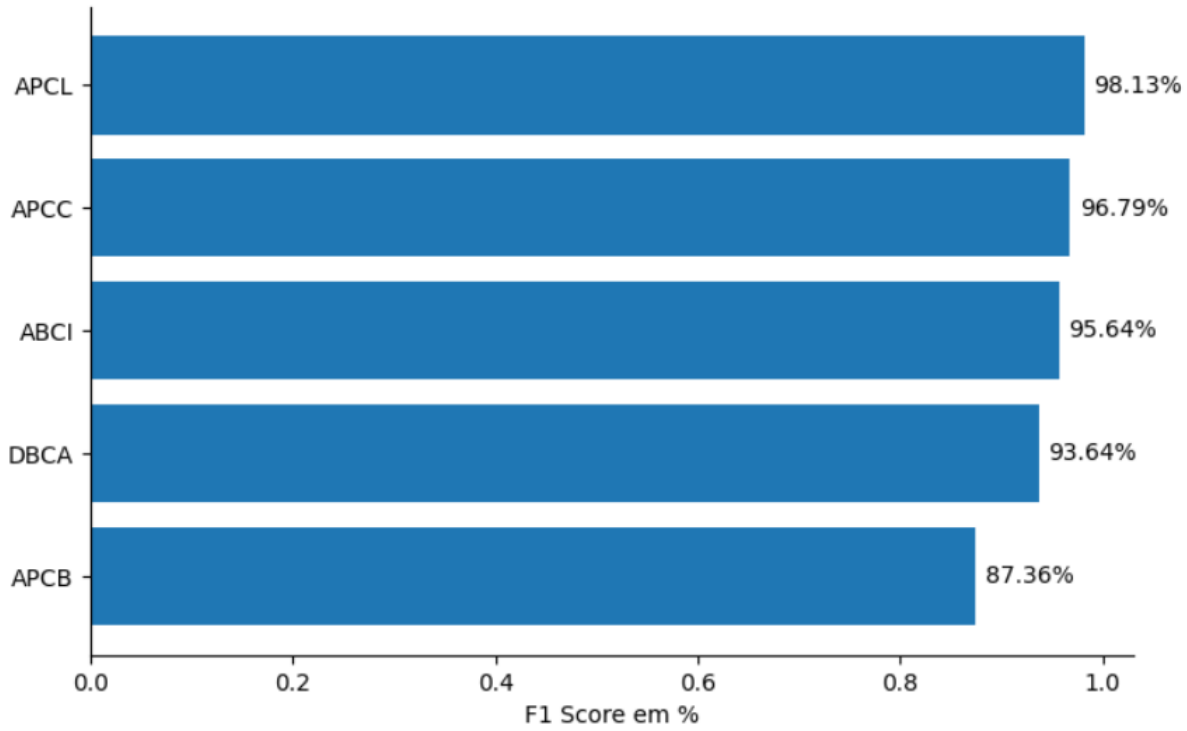
Para isso, buscou-se selecionar aleatoriamente PMU's da área interna e treinar o modelo com base em suas características, aumentando gradativamente o número de PMU's disponíveis. Para cada grupo numérico de PMU's treinaram-se 20 modelos diferentes e a

Figura 24 – Erro por tipo de Falta



Fonte: O Autor

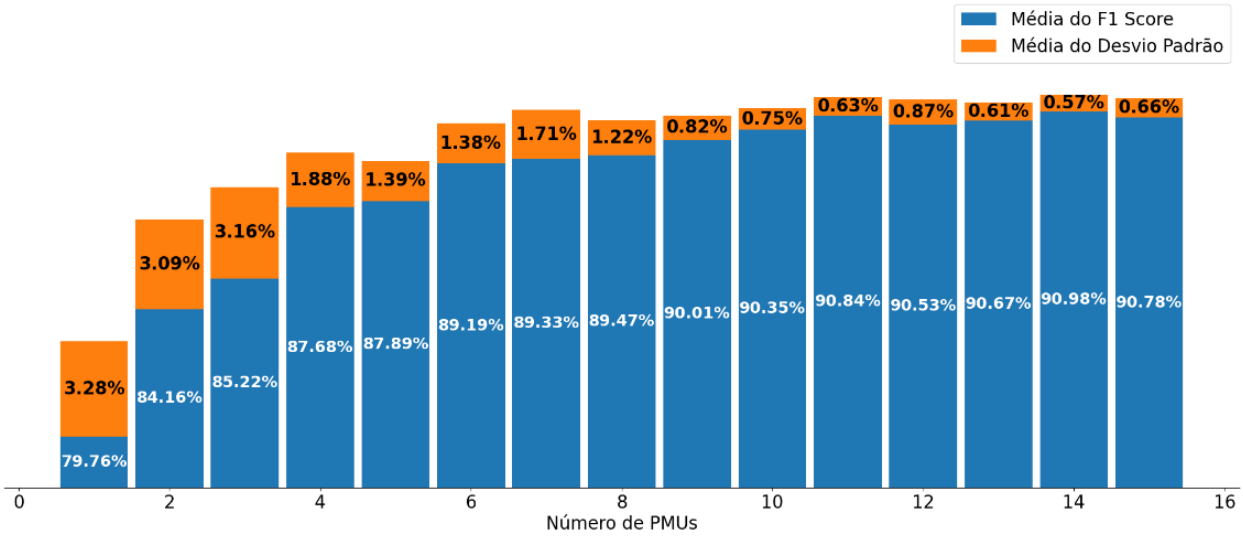
Figura 25 – F1-Score por tipo de falta



Fonte: O Autor

média dos seus desempenhos foram calculados, a Figura 26 demonstra os resultados.

Figura 26 – Desempenho do modelo por número de PMU's



Fonte: O Autor

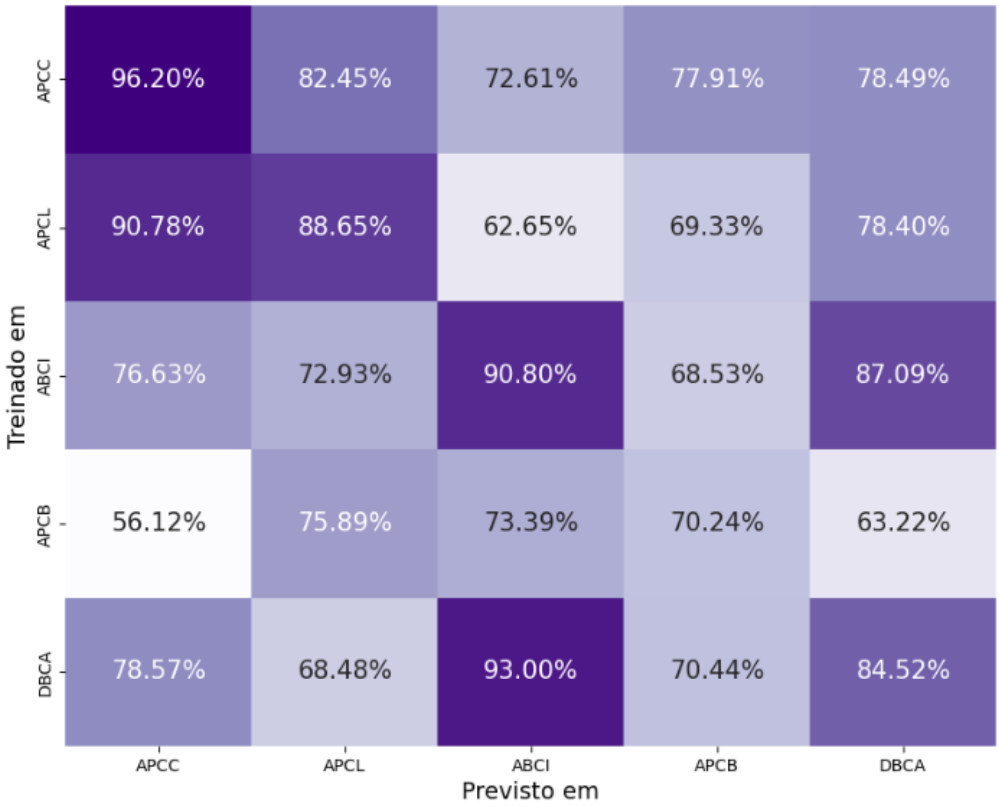
Observa-se que somente uma PMU já é o suficiente para entregar $79.76 \pm 3.28\%$ de F1-Score, isso representa aproximadamente 88% da capacidade total da modelagem. Além disso, observa-se que mesmo com o aumento de desempenho de modelo conforme o aumento de PMUs disponíveis, este acréscimo tende a diminuir, praticamente se estabilizando a partir de 9 PMU's.

Com estas informações, destaca-se a importância do trabalho de modelagem e de conhecer diferentes modelos de diferentes níveis de complexidades, visto que modelos Ensemble performam em média melhor com somente acesso a 1 PMU, em comparação com Árvores de Decisão acessando todas as PMU's.

5.5 Resultado 5 - Desempenho em faltas não previamente treinadas

Esta seção tem como objetivo trazer resultados sobre quanto eventos de uma falta são capazes de descrever outras faltas que ocorrem no sistema. Para isso, foi treinado diferentes modelos para cada falta do sistema, depois, foram submetidos dados de faltas que o modelo não viu no treinamento para avaliar seu desempenho. A figura 27 resume estes resultados.

Figura 27 – Desempenho em faltas não previamente treinadas



Fonte: O Autor

Os resultados mostram que enquanto as faltas APCC, APCL e ABCI são necessárias para prever seus próprios eventos a falta APCB performou um pouco melhor em prever a localização de faltas APCL e ABCI, enquanto a falta DBCA performou melhor em prever faltas ABCI.

Este resultado se mostra relevante visto que em um contexto não simulado, algumas faltas são mais abundantes no sistema que outras, e isso pode desbalancear o conjunto de dados e impossibilitar a coleta abundante de amostras de determinadas faltas.

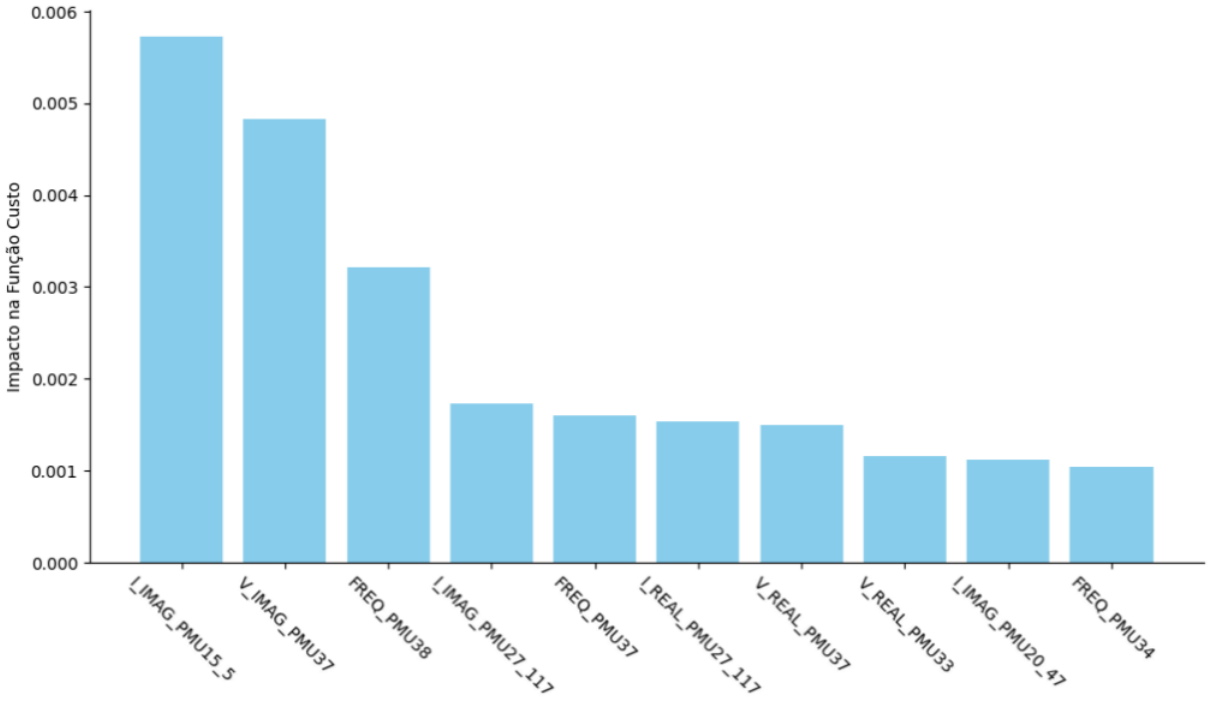
5.6 Resultado 6 - Importância de Características

O processo de interpretação de importância de características é um campo de estudo vasto dentro da área de Aprendizagem de Máquina, isso porque um dos principais problemas que modelos robustos sofrem hoje é a dificuldade em trazer as razões pelas quais um modelo tomou determinada previsão, os chamados modelos *black-box*. Por isso,

saber o peso de cada característica é uma das possibilidades na hora de entender tópicos relacionados a esse tema.

Dentro da biblioteca Catboost, existe um suporte completo a cálculo de importância de características a partir de diferentes métodos. O método escolhido para esta análise é o nomeado *"LossFunctionChange"* que simplesmente compara a função custo com e sem aquela característica, utilizando de aproximações estatísticas para não precisar retreinar o modelo diversas vezes o resultado obtido é descrito na figura 28.

Figura 28 – Impacto da função custo em fasores



Fonte: O Autor

Tirar conclusões a partir desse resultado é um processo delicado, o que é possível afirmar é que as informações que mais prejudicariam a performance do modelo utilizado caso fossem faltantes seriam as 10 listadas na figura, porém, não é possível afirmar que essas são as 10 características que melhor descrevem a variável alvo, porque essa afirmação pode variar de modelo para modelo, e exige análise estatísticas mais robustas.

Porém, essa informação é importante quando se trata de custos operacionais, uma vez que tenhamos uma lista de características e quais são as mais importantes para o modelo performar, é possível trabalhar a eficiência do modelo versus o custo operacional para

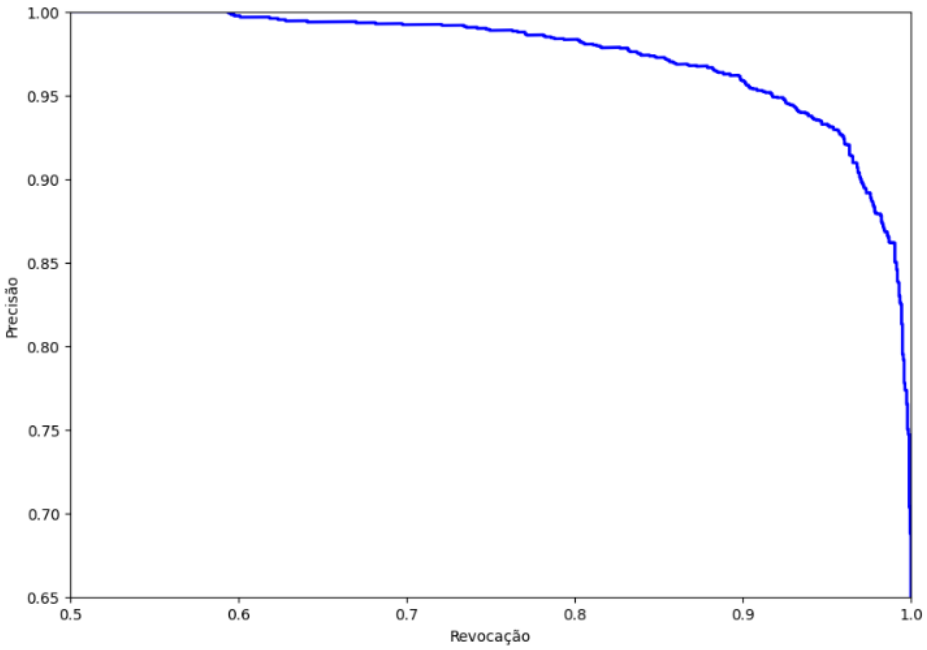
manter as medições daquela característica a nível mais granular que apenas PMUs, porém esse processo deve ser feito com bastante embasamento e atenção, visto que características podem ser relacionadas entre si, entrando em casos de multicolinearidades, em que uma informação é computada de forma repetida em características.

5.7 Resultado 7 - Análise de Tipo de Erro

Ao realizar uma previsão o modelo retorna uma probabilidade de uma predição ser verdadeira (Classe 1), no caso deste trabalho, Área Interna. Dessa forma, é estabelecido um limiar (Valor Padrão 0.5) que define se aquela amostra será classificada como verdadeira ou falsa.

Não é possível aumentar o desempenho do modelo variando o limiar de decisão, porém é possível controlar o tipo de erro causado. A curva Precisão-Revocação da modelagem é mostrado na figura 29.

Figura 29 – Curva Precisão-Revocação



Fonte: O Autor

A curva nos mostra que para se obter uma precisão perfeita, isto é, não possuir erros de Falso Positivo, o modelo precisaria ter uma revocação de aproximadamente 0.6. Para

avaliar a utilidade prática dessa ferramenta precisa-se primeiro entender a implicabilidade de negócio dos erros atingidos.

Erros de Falso Positivo - No problema apresentado, Classe 1 significa a falta ser dentro da concessionária de acesso aos dados de PMU, isto é, hipoteticamente, a concessionária que está aplicando e monitorando a solução de Machine Learning. Um erro Falso Positivo significa que a classe foi prevista como 1 quando deveria ser 0, ou seja, foi previsto que o problema é interno quando na verdade é externo.

Erros de Falso Negativo - Dado a descrição acima, Um erro Falso Negativo significa que a classe foi prevista como 0 quando deveria ser 1, ou seja, foi previsto que o problema é externo quando na verdade é interno.

Pode-se pensar que as implicações em ter um erro Falso Positivo sejam o falso alarme em resolver o problema da falta por parte da concessionária que aplica o modelo, isso pode envolver checar mais afundo através de outros métodos e medidas a causa daquele problema, isto é, alocar um grupo de trabalhadores da concessionária que despenderiam tempo investigando o problema e no fim descobririam que a falta foi na concessionária vizinha.

Pode-se pensar que as implicações em ter um erro Falso Negativo sejam o não alarme em resolver o problema de falta por parte da concessionária, isto é, a falsa impressão da não necessidade de manutenção do sistema, pois a partir desse método, as faltas seriam imperceptíveis.

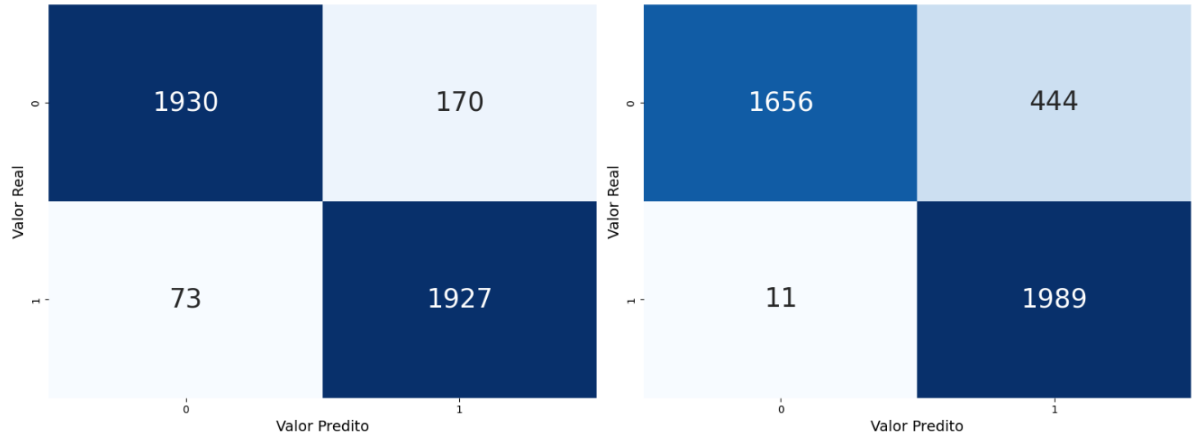
Dado a análise descrita acima, supõem-se para esta análise que possuir um erro de Falso Negativo seja mais problemático que um erro de Falso Positivo, desta forma a métrica que a ser maximizada seria a Revocação, pois isso acarreta em minimização dos erros de Falso Negativo.

Ao manipular o limiar, chega-se a conclusão que para uma Revocação de 98% a maior precisão que pode ser alcançada é 87%, para isto, basta o modelo prever 28% de chance da falta ocorrer na área interna que já é o suficiente para amostra ser Classe 1.

A Figura 30 mostra a comparação das matrizes de confusão antes e depois do

ajuste do limiar.

Figura 30 – Comparação de Matriz de confusão ao variar o limiar



Fonte: O Autor

A alteração no limiar de decisão ocasionou uma diminuição de 62 erros de Falso Negativo, mas também ocasionou um aumento de 274 erros de Falsos Positivos.

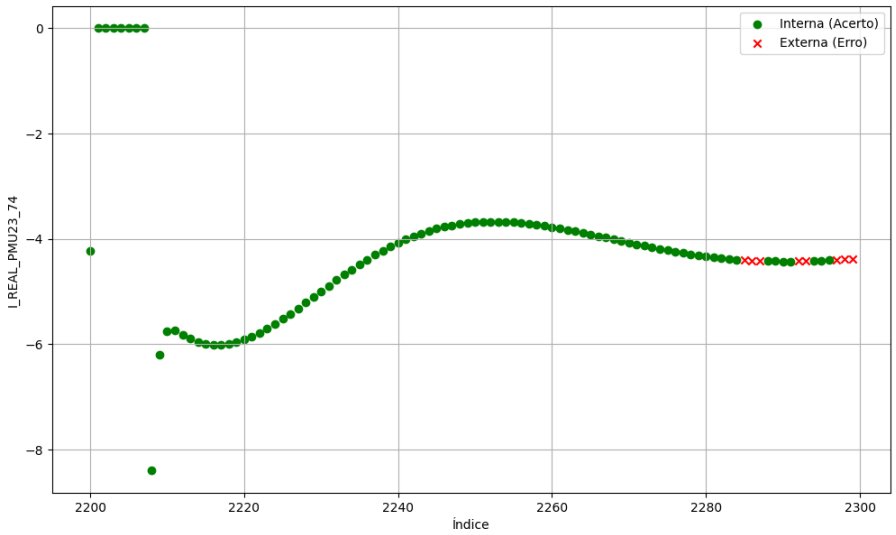
5.8 Resultado 8 - Agregando previsões para aumento de desempenho

Em modelagem de aprendizagem de máquina, etapas de pós-processamento também podem ser aplicadas para aumento de desempenho e extração de informação relevante para o problema proposto.

No contexto deste trabalho, entende-se que uma mesma simulação não pode ter faltas Internas e Externas ao mesmo tempo, pois ao configurar as simulações utilizadas, limitou-se a uma falta por vez.

Porém, não foi apresentada nenhuma informação referente a separação das simulações ao treinamento do modelo, justamente por isso não ser aplicável em uma solução real. A figura 31 representa a parte real de um fasor de corrente na PMU23 em uma das simulações feitas.

Figura 31 – Acertos e erros em referente a fasor de corrente na PMU23



Fonte: O Autor

Com a falta aplicada no instante 0, a corrente apresenta um comportamento anômalo, chegando a atingir o 0 em algumas amostras, ao final da simulação, repara-se que o modelo fica incerto do local da falta, o que não pode ocorrer.

A partir desta premissa, decide-se utilizar a moda das previsões como forma de classificar a simulação toda com apenas uma classe, seja Interna ou Externa. E isto foi utilizado como hiperparâmetro da modelagem, ou seja, buscou-se encontrar o melhor valor n de número de amostras coletadas que maximiza a acurácia quando utiliza-se da moda para classificar uma simulação como Classe 1 ou Classe 0.

A busca foi feita para os seguintes valores de n, representados na lista na equação 5.1.

$$n = [1, 3, 5, 7, 9, 25, 55, 75, 125, 175, 225] \tag{5.1}$$

A tabela 12 representa a acurácia do modelo ao passar pelo pós-processamento e o número de amostras mínimo necessário para atingir essa acurácia.

Tabela 12 – Acurácia da modelagem após pós-processamento

Modelo	**Acurácia**	**número de amostras mínimo**
Árvore de Decisão	$90,24\%$	3
LightGBM	$97,56\%$	150
XGBoost	$97,56\%$	125
Floresta Aleatória	$97,56\%$	125
CatBoost	$97,56\%$	5

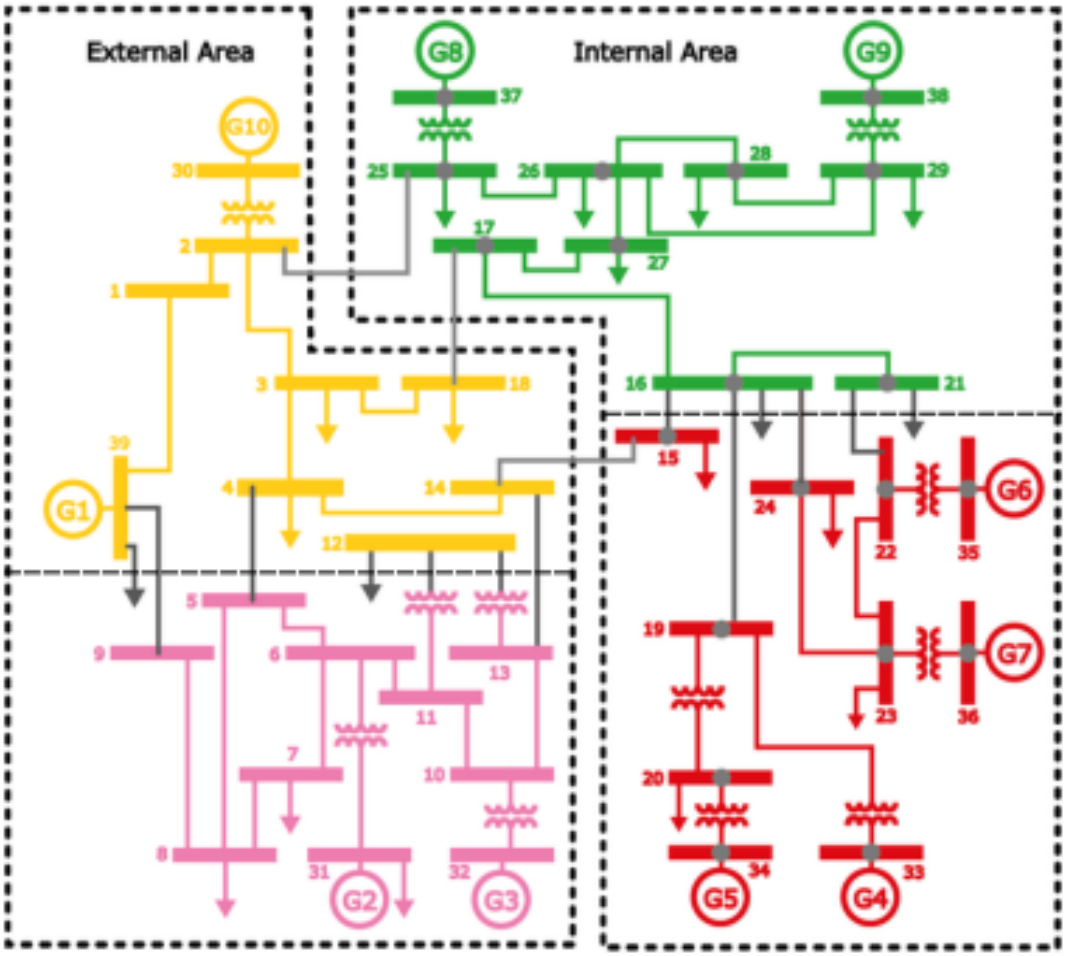
O resultado aponta que todos os modelos ensemble obtiveram a mesma acurácia, analisando em detalhes, todas as simulações foram classificadas corretamente exceto uma, uma simulação APCC na barra 26. Ademais, o modelo Catboost conseguiu atingir este resultado com impressionantes 5 amostras, enquanto que os outros modelos atingiram este valor em 125 a 150 amostras, este valor é importante porque quanto menor o número de amostras necessário, maior a velocidade de classificação do sistema. Por fim, destaca-se também o resultado da Árvore de Decisão, que apesar de sua simplicidade, com essa técnica classificou corretamente 90,24% das simulações.

Em uma aplicação real, não seria trivial ter uma identificação de eventos para segregar o comportamento da rede em falta como é feito neste trabalho utilizando simulações, para isso, técnicas de detecção de anomalia, que identifiquem quando o sistema entra em estado de falta e consequentemente começar a classificar amostras seria necessário.

5.9 Resultado 9 - Escalonando a solução: dividindo o sistema em 4 áreas

Este resultado tem como objetivo demonstrar a fácil escalabilidade que a solução pode tomar uma vez que se tenha acesso a dados rotulados. Aqui, foi feita a redivisão do sistema, dividindo as áreas Internas e Externas em: Interno Norte, Interno Sul, Externo Norte e Externo Sul. Isso é possível de ser feito apenas manipulando em código os resultados simulados e coletados pelas PMU's, para entender a redivisão compara-se a figura 19 com a figura 32.

Figura 32 – Divisão do Sistema em 4 áreas



Fonte: O Autor

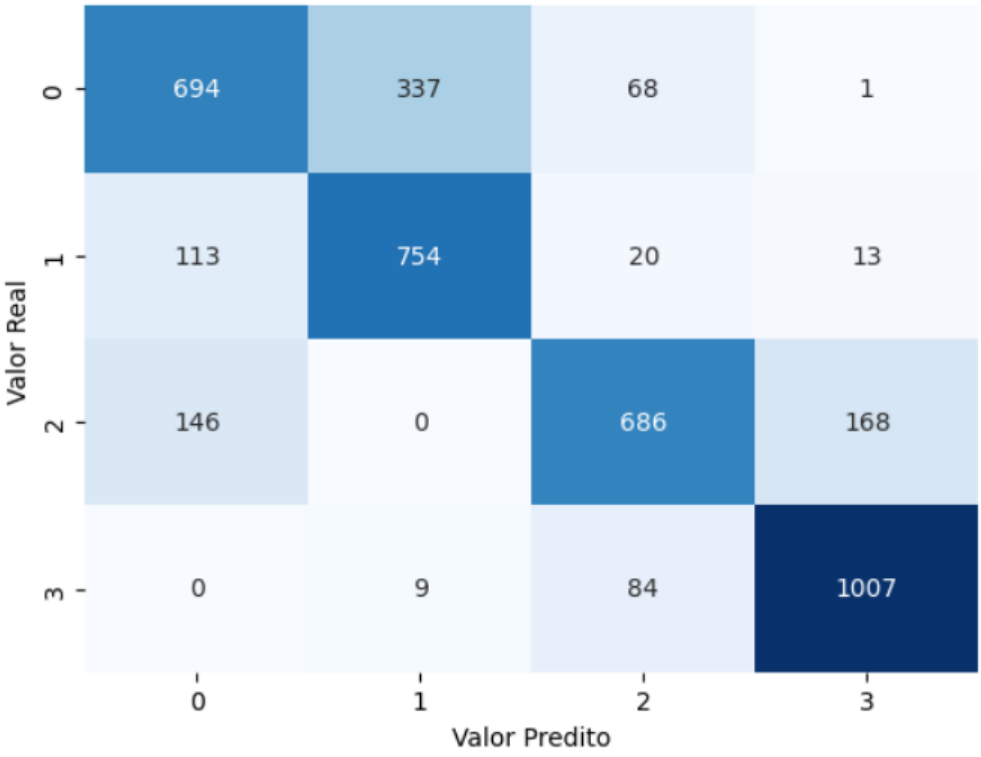
O mapeamento de cores auxilia a identificar as regiões de separação, mantendo a integridade da separação original. Na modelagem original, as linhas que ligam as regiões externas e internas foram desconsideradas como passíveis de sofrerem faltas. Caso este processo fosse feito nesta nova divisão, muitas linhas do sistema ficariam submetidas a mesma condição, reduzindo consideravelmente a quantidade de dados disponibilizada ao modelo. Por isso, adotou-se o critério de quando uma dessas linhas sofre a falta, a classificação da localização é padronizada para de onde a linha parte, e não de onde a linha chega. Esta classificação toma como base os dados fornecidos pelo próprio IEEE-39.

Em seguida o modelo foi retreinado, com o seguinte mapeamento de classes.

- Classe 0 - Interno Norte
- Classe 1 - Interno Sul
- Classe 2 - Externo Norte
- Classe 3 - Externo Sul

O resultado é visualizado a partir da figura 33.

Figura 33 – Matriz de Confusão - Sistema 4 Áreas



Fonte: O Autor

O tempo de treinamento para este problema sofreu um aumento para 109.95 segundos, isto é 3 vezes mais que o seu treinamento no problema binário. Suas métricas de Acurácia e F1-Score ambas detém um valor de 76%, comparado a um novo modelo aleatório de que possivelmente teria 25% em sua métrica.

Além disso, é possível ver uma tendência maior do modelo em errar as previsões referentes as divisões Norte e Sul, mas a incidência de erro é muito menor em erros Externo e Interno.

É importante ressaltar que ao criar novas classes para o problema, o processo de modelagem muda consideravelmente, isto porque, a função custo abordada até agora é exclusivamente para problemas de classificação binários. Além disso, deve-se considerar um maior imbalanceamento das classes a serem previstas, além do aumento do custo computacional em resolver problemas desse tipo.

6 Conclusões e Trabalhos Futuros

A partir dos resultados demonstrados neste trabalho é possível concluir que o uso de Árvores de Decisão para a previsão do local de falta em sistemas de potência pode trazer uma alta taxa de acerto ao separar o sistema em duas áreas. Além disso, foi demonstrado a possibilidade de usar os mesmos algoritmos, para uma classificação multiclasse, podendo separar o sistema em mais áreas e a localização da falta ser mais precisa geograficamente. Técnicas de pré-processamento e pós-processamento podem ser benéficas para auxiliar o modelo na classificação da falta, demonstrando desempenho positivo ao se utilizar da moda das previsões de uma falta para aumentar a qualidade do modelo.

Ainda, é possível relacionar o desempenho do modelo com diversas características do sistema de potência, é possível visualizar a taxa de acerto do modelo por falta, assim como quanto cada tipo de falta beneficia o modelo em sua previsão ao estar presente nos dados de treinamento. É também possível estabelecer com as PMU's e a performance do modelo, como uma análise de custo de investimento ao instalar PMU's x ganho de performance do modelo, visto que o modelo possui uma maior taxa de acerto quando observa mais fasores da rede elétrica porém essa relação não é linear, também sendo possível identificar o impacto de cada fasor na minimização da função custo do modelo.

Por fim, também foi apresentado uma análise de Tipo de Erro, onde se mostra possível variar o limiar de decisão do modelo para que o usuário controle a taxa de erro Falso Positivo e Falso Negativo, como esperado, isso não aumenta a performance geral do modelo, porém é uma análise válida ao estabelecer que um tipo de Erro pode ser mais prejudicial que outro.

Ao comparar o desempenho do modelo desenvolvido neste trabalho com o desempenho de Redes Neurais apresentado em 3, a acurácia de ambos se mostra similar no processo de validação cruzada, por isso, neste trabalho conclui-se que para se analisar o benefício de um modelo sob o outro, devem ser avaliadas características que vão além de métricas de performance, como por exemplo, interpretabilidade do modelo, suporte em

suas respectivas bibliotecas com funções complementares de análise, custo computacional, dentre outros.

A utilização de aprendizagem de máquina vem sendo objeto de estudo crescente com o passar dos anos, para dar segmento no estudo apresentado neste trabalho, sugere-se principalmente aproximar a modelagem de um caso de um Sistema Elétrico real, diversas características de um sistema elétrico não foram consideradas nesse trabalho e podem influenciar diretamente no impacto desta solução. Por exemplo: Dificuldade na obtenção de dados de qualidade e a capacidade de faltas ocorrerem simultaneamente.

No âmbito de aprendizagem de máquina, é possível explorar a capacidade de um modelo identificar quando o momento de uma falta a partir de um detector de anomalias, isso restringiria o número de amostras dada ao modelo de classificação de local, viabilizando o trabalho de pós processamento. Também sugere-se a possibilidade de realizar uma análise de interpretabilidade do modelo mais profunda, com o objetivo de um maior entendimento da relação do SEP com o modelo, utilizando de, por exemplo, valores SHAP para diagnosticar as previsões.

Referências

AKIBA, T.; SANO, S.; YANASE, T.; OHTA, T.; KOYAMA, M. *Optuna: A Next-generation Hyperparameter Optimization Framework*. 2019. Disponível em: <https://arxiv.org/abs/1907.10902>. Cited 2 times in pages 44 e 60.

ALELYANI, S. Stable bagging feature selection on medical data. *Journal of Big Data*, v. 8, 01 2021. Cited in page 41.

AWAYA, N.; MA, L. *Unsupervised tree boosting for learning probability distributions*. 2023. Disponível em: <https://arxiv.org/abs/2101.11083>. Cited in page 31.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, JMLR.org, v. 13, p. 281–305, 2012. Cited in page 42.

BISHOP, C. M. *Pattern Recognition and Machine Learning*. New York: Springer, 2006. 41 p. Cited in page 26.

BOLLEN, M.; SOCIETY, I. I. A.; SOCIETY, I. P. E.; SOCIETY, I. P. E. *Understanding Power Quality Problems: Voltage Sags and Interruptions*. Wiley, 2000. (IEEE Press Series on Power Engineering (Was Power Systems Engineering), Series Editor Ser: Paul M. Anderson Series). ISBN 9780780347137. Disponível em: <https://books.google.com.br/books?id=Je0eAQAAIAAJ>. Cited 2 times in pages 92 e 93.

BREIMAN, L. Bagging predictors. *Machine Learning*, v. 24, n. 2, p. 123–140, Aug 1996. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/BF00058655>. Cited in page 40.

BREIMAN, L. Random forests. *Machine Learning*, v. 45, n. 1, p. 5–32, Oct 2001. ISSN 1573-0565. Disponível em: <https://doi.org/10.1023/A:1010933404324>. Cited in page 41.

CARUANA, R.; NICULESCU-MIZIL, A.; CREW, G.; KSIKES, A. Ensemble selection from libraries of models. In: *Proceedings of the Twenty-First International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2004. (ICML '04), p. 18. ISBN 1581138385. Disponível em: <https://doi.org/10.1145/1015330.1015432>. Cited in page 37.

CEPEL. *Manual Anatem*. [S.l.], 2024. Disponível em: <https://see.cepel.br/manual/anatem/index.html>. Cited 2 times in pages 16 e 61.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. v. 11, p. 785–794. Disponível em: <http://dx.doi.org/10.1145/2939672.2939785>. Cited in page 39.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. (KDD '16, v. 11), p. 785–794. Disponível em: <http://dx.doi.org/10.1145/2939672.2939785>. Cited in page 39.

FERREIRA, E. C. *Sensores e Condicionamento de Sinais*. São Paulo: Departamento de Eletrônica e Microeletrônica - Unicamp. Disponível em: <https://www.dsif.fee.unicamp.br/~elnatan/ie327/IE327.pdf>. Cited in page 19.

FLOYD, T. *Sistemas Digitais:Fundamentos e Aplicações*. São Paulo: Bookman, 2007. Cited in page 20.

GÉRON, A. *Mãos á Obra: Aprendizado de Máquina com Scikit-Learn, Keras & Tensorflow: Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes*. 2. ed. [S.l.]: O'Reilly, 2019. 3 p. Cited in page 13.

GÉRON, A. *Mãos á Obra: Aprendizado de Máquina com Scikit-Learn, Keras & Tensorflow: Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes*. 2. ed. [S.l.]: O'Reilly, 2019. 137 p. Cited in page 13.

GÉRON, A. *Mãos á Obra: Aprendizado de Máquina com Scikit-Learn, Keras & Tensorflow: Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes*. 2. ed. [S.l.]: O'Reilly, 2019. 7-9 p. Cited 2 times in pages 25 e 27.

GÉRON, A. *Mãos á Obra: Aprendizado de Máquina com Scikit-Learn, Keras & Tensorflow: Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes*. 2. ed. [S.l.]: O'Reilly, 2019. 137-138 p. Cited in page 31.

GÉRON, A. *Mãos á Obra: Aprendizado de Máquina com Scikit-Learn, Keras & Tensorflow: Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes*. 2. ed. [S.l.]: O'Reilly, 2019. 138 p. Cited in page 32.

GÉRON, A. *Mãos á Obra: Aprendizado de Máquina com Scikit-Learn, Keras & Tensorflow: Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes*. 2. ed. [S.l.]: O'Reilly, 2019. 139 p. Cited in page 33.

GÉRON, A. *Mãos á Obra: Aprendizado de Máquina com Scikit-Learn, Keras & Tensorflow: Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes*. 2. ed. [S.l.]: O'Reilly, 2019. 142 p. Cited 2 times in pages 33 e 34.

GÉRON, A. *Mãos á Obra: Aprendizado de Máquina com Scikit-Learn, Keras & Tensorflow: Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes*. 2. ed. [S.l.]: O'Reilly, 2019. 72 p. Cited in page 48.

GÉRON, A. *Mãos á Obra: Aprendizado de Máquina com Scikit-Learn, Keras & Tensorflow: Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes*. 2. ed. [S.l.]: O'Reilly, 2019. 75 p. Cited 3 times in pages 50, 51 e 52.

GÉRON, A. *Mãos á Obra: Aprendizado de Máquina com Scikit-Learn, Keras & Tensorflow: Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes*. 2. ed. [S.l.]: O'Reilly, 2019. 75 p. Cited in page 50.

GÉRON, A. *Mãos á Obra: Aprendizado de Máquina com Scikit-Learn, Keras & Tensorflow: Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes*. 2. ed. [S.l.]: O'Reilly, 2019. 75-77 p. Cited in page 50.

GÉRON, A. *Mãos á Obra: Aprendizado de Máquina com Scikit-Learn, Keras & Tensorflow: Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes*. 2. ed. [S.l.]: O'Reilly, 2019. 75-77 p. Cited in page 52.

GHOJOGH, B.; CROWLEY, M. *The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial*. 2023. Disponível em: <https://arxiv.org/abs/1905.12787>. Cited 4 times in pages 27, 28, 29 e 30.

GHOJOGH, B.; CROWLEY, M. *The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial*. 2023. Disponível em: <https://arxiv.org/abs/1905.12787>. Cited in page 28.

GONG, Z.; ZHONG, P.; HU, W. Diversity in machine learning. *IEEE Access*, v. 7, p. 64323–64350, 2019. Cited in page 28.

HARRIS, C. R.; MILLMAN, K. J.; WALT, S. J. van der; GOMMERS, R.; VIRTANEN, P.; COURNAPEAU, D.; WIESER, E.; TAYLOR, J.; BERG, S.; SMITH, N. J.; KERN, R.; PICUS, M.; HOYER, S.; KERKWIJK, M. H. van; BRETT, M.; HALDANE, A.; RÍO, J. F. del; WIEBE, M.; PETERSON, P.; GÉRARD-MARCHANT, P.; SHEPPARD, K.; REDDY, T.; WECKESSER, W.; ABBASI, H.; GOHLKE, C.; OLIPHANT, T. E. Array programming with NumPy. *Nature*, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>. Cited in page 59.

HART, D. G.; UY, D.; GHARPURE, V.; NOVOSEL, D.; KARLSSON, D.; KABA, M. *PMUs – A new approach to power network monitoring*. ABB Group, 2001. Disponível em: <https://library.e.abb.com/public/a6880679d199bb63c1256ddd00346c1e/58-61%20M800.pdf>. Cited in page 19.

HO, T. K. Random decision forests. In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. [S.l.: s.n.], 1995. v. 1, p. 278–282 vol.1. Cited in page 41.

HUTTER, F.; HOOS, H. H.; LEYTON-BROWN, K. Sequential model-based optimization for general algorithm configuration. In: COELLO, C. A. C. (Ed.). *Learning and Intelligent Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 507–523. ISBN 978-3-642-25566-3. Cited in page 44.

IEEE Approved Draft Guide for Synchronization, Calibration, Testing, and Installation of Phasor Measurement Units (PMU) for Power System Protection and Control. *IEEE PC37.242/D11, October 2012*, p. 1–125, 2013. Cited in page 21.

IEEE Standard for Synchrophasor Data Transfer for Power Systems. *IEEE Std C37.118.2-2011 (Revision of IEEE Std C37.118-2005)*, p. 1–53, 2011. Cited 2 times in pages 18 e 22.

IEEE Standard for Synchrophasor Measurements for Power Systems. *IEEE Std C37.118.1-2011 (Revision of IEEE Std C37.118-2005)*, p. 1–61, 2011. Cited in page 18.

JIANG, J.; WANG, R.; WANG, M.; GAO, K.; NGUYEN, D. D.; WEI, G.-W. Boosting tree-assisted multitask deep learning for small scientific datasets. *Journal of Chemical Information and Modeling*, v. 60, n. 3, p. 1235–1244, 2020. PMID: 31977216. Disponível em: <https://doi.org/10.1021/acs.jcim.9b01184>. Cited in page 38.

KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. Curran Associates, Inc.,

v. 30, 2017. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf>. Cited in page 39.

KHUN, M. Disponível em: <https://tune.tidymodels.org/articles/acquisition_functions.html>. Cited 4 times in pages 44, 45, 46 e 47.

KIM, D. I.; WHITE, A.; SHIN, Y. J. Pmu-based event localization technique for wide-area power system. Yonsei University, Seul, South Korea, v. 33, n. 6, 2018. Disponível em: <https://ieeexplore.ieee.org/document/8334283>. Cited in page 14.

KINDERMANN, G. *Curto-Circuito*. 2. ed. [S.l.]: Sagra Luzzato, 1997. 138-140 p. Cited 2 times in pages 16 e 17.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2017. Disponível em: <https://arxiv.org/abs/1412.6980>. Cited in page 56.

LIU, F. T.; TING, K. M.; ZHOU, Z.-H. Isolation forest. In: *2008 Eighth IEEE International Conference on Data Mining*. [S.l.: s.n.], 2008. p. 413–422. Cited in page 31.

LIU, Y.; ZHOU, Y.; WEN, S.; TANG, C. A strategy on selecting performance metrics for classifier evaluation. *International Journal of Mobile Computing and Multimedia Communications*, v. 6, p. 20–35, 10 2014. Cited in page 47.

LOVELL, D.; MILLER, D.; CAPRA, J.; BRADLEY, A. *Never mind the metrics – what about the uncertainty? Visualising confusion matrix metric distributions*. 2022. Disponível em: <https://arxiv.org/abs/2206.02157>. Cited in page 51.

MARTINS, D. de B. Análise de curto-circuito probabilístico em sistemas de distribuição de energia elétrica com geração distribuída. Escola de Engenharia de São Carlos, São Paulo, 2021. Cited in page 18.

MARTINS, R. da S. Apresentação do sistema de medição fasorial sincronizada e abordagem de sua implantação no estimador de estado. Rio de Janeiro, 2012. Disponível em: <https://pantheon.ufrj.br/bitstream/11422/9013/1/monopoli10005247.pdf>. Cited in page 18.

MIENYE, D.; SUN, Y. A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access*, PP, p. 1–1, 09 2022. Cited 2 times in pages 36 e 38.

MIENYE, I. D.; SUN, Y. A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access*, v. 10, p. 99129–99149, 2022. Cited 4 times in pages 35, 36, 41 e 42.

NADUVATHUPARAMBIL, B.; VALENTI, M.; FELIACHI, A. Communication delays in wide area measurement systems. In: *Proceedings of the Thirty-Fourth Southeastern Symposium on System Theory (Cat. No.02EX540)*. [S.l.: s.n.], 2002. p. 118–122. Cited 2 times in pages 21 e 22.

NATEKIN, A.; KNOLL, A. Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, v. 7, 2013. ISSN 1662-5218. Disponível em: <https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2013.00021>. Cited in page 38.

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. Disponível em: <https://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart>. Cited 2 times in pages 34 e 35.

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. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html>. Cited in page 48.

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. Cited in page 59.

PHADKE, A.; THORP, J. Synchronized phasor measurements and their applications. Springer, Virginia, 2008. Cited in page 18.

PICHLER, M.; HARTIG, F. Machine learning and deep learning—a review for ecologists. *Methods in Ecology and Evolution*, v. 14, 02 2023. Cited in page 29.

PROKHORENKOVA, L.; GUSEV, G.; VOROBEV, A.; DOROGUSH, A. V.; GULIN, A. *CatBoost: unbiased boosting with categorical features*. 2019. Disponível em: <https://arxiv.org/abs/1706.09516>. Cited in page 40.

RAI, R.; TIWARI, M. K.; IVANOV, D.; DOLGUI, A. Machine learning in manufacturing and industry 4.0 applications. International Journal of Production Research. Disponível em: <https://doi.org/10.1080/00207543.2021.1956675>. Cited in page 14.

RASHIDI, H. H.; TRAN, N. K.; BETTS, E. V.; HOWELL, L. P.; GREEN, R. Artificial intelligence and machine learning in pathology: The present landscape of supervised methods. *Academic Pathology*, v. 6, p. 2374289519873088, 2019. ISSN 2374-2895. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2374289521001573>. Cited in page 29.

RIBEIRO, M. H. D. M.; dos Santos Coelho, L. Ensemble approach based on bagging, boosting and stacking for short-term prediction in agribusiness time series. *Applied Soft Computing*, v. 86, p. 105837, 2020. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494619306180>. Cited in page 40.

SCHUMACHER, R.; AOKI, A. R.; OLIVEIRA, G. H. C.; KUIAVA, R. Accurate internal/external area under fault classifier applying neural networks to pmu data. *International Journal of Electrical Power and Energy Systems*, Curitiba, Paraná, Brazil, 2023. Preprint submitted, in process of publication. Cited in page 54.

SEREM, N.; LETTING, L.; MUNDA, J. Voltage profile analysis on a grid with power injection from a wind farm. *Energies*, p. 7530, 11 2021. Cited in page 61.

SG Analytics. *2.5 Quintillion Bytes of Data Generated Everyday - Top Data Science Trends 2020*. 2020. Cited in page 13.

SINGH, B.; SHARMA, N.; TIWARI, A.; VERMA, K.; SINGH, S. Applications of phasor measurement units (pmus) in electric power system networks incorporated with facts controllers. International Journal of Engineering, v. 3, p. 64–82, 2011. Cited in page 18.

STERKENBURG, T. F.; GRUNWALD, P. D. The no-free-lunch theorems of supervised learning. Cornell University, New York, 2022. Disponível em: <https://arxiv.org/abs/2202.04513>. Cited in page 13.

TEAM, T. pandas development. *pandas-dev/pandas: Pandas*. Zenodo, 2020. Disponível em: <https://doi.org/10.5281/zenodo.3509134>. Cited in page 59.

VIRTANEN, P.; GOMMERS, R.; OLIPHANT, T. E.; HABERLAND, M.; REDDY, T.; COURNAPEAU, D.; BUROVSKI, E.; PETERSON, P.; WECKESSER, W.; BRIGHT, J.; van der Walt, S. J.; BRETT, M.; WILSON, J.; MILLMAN, K. J.; MAYOROV, N.; NELSON, A. R. J.; JONES, E.; KERN, R.; LARSON, E.; CAREY, C. J.; POLAT, İ.; FENG, Y.; MOORE, E. W.; VanderPlas, J.; LAXALDE, D.; PERKTOLD, J.; CIMRMAN, R.; HENRIKSEN, I.; QUINTERO, E. A.; HARRIS, C. R.; ARCHIBALD, A. M.; RIBEIRO, A. H.; PEDREGOSA, F.; van Mulbregt, P.; SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, v. 17, p. 261–272, 2020. Cited in page 59.

WANG, R.; LIU, Y.; YE, X.; TANG, Q.; GOU, J.; HUANG, M.; WEN, Y. Power system transient stability assessment based on bayesian optimized lightgbm. In: *2019 IEEE 3rd Conference on Energy Internet and Energy System Integration (EI2)*. [S.l.: s.n.], 2019. p. 263–268. Cited in page 40.

XIA, Y.; LIU, C.; LI, Y.; LIU, N. A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring. *Expert Systems with Applications*, v. 78, p. 225–241, 2017. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417417301008>. Cited 3 times in pages 42, 43 e 44.

Apêndices

APÊNDICE A – Análise de Afundamento de Tensão em faltas monofásicas

Este apêndice tem como objetivo descrever o comportamento do SEP em faltas, utilizando de exemplo a falta Monofásica-Terra, em que o mecanismo de Desligamento Monofásico é utilizado (*Single-Phase Tripping*). A fundamentação a seguir é baseada na seção 3.6.1 da obra *Understanding Power Quality Problems* do autor Math Bollen.

A.1 Tensão durante a falta

Durante o desligamento monofásico, a impedância entre a fase da falta e o terra faz com que a tensão durante a falta seja muito próxima de 0, desta forma, o equacionamento A.1 representa o comportamento do sistema, com a sendo a fase da falta.

$$\begin{aligned} V_a &= 0 \\ V_b &= (-\frac{1}{2} - \frac{1}{2}j\sqrt{3})E \\ V_c &= (-\frac{1}{2} + \frac{1}{2}j\sqrt{3})E \end{aligned} \tag{A.1}$$

Sendo E a magnitude da tensão antes do evento de falta. Considerando a tensão E como sendo a base *pu* do sistema, o equacionamento A.2 representa seu formato.

$$\begin{aligned} V_a &= 0 \\ V_b &= (-\frac{1}{2} - \frac{1}{2}j\sqrt{3}) \\ V_c &= (-\frac{1}{2} + \frac{1}{2}j\sqrt{3}) \end{aligned} \tag{A.2}$$

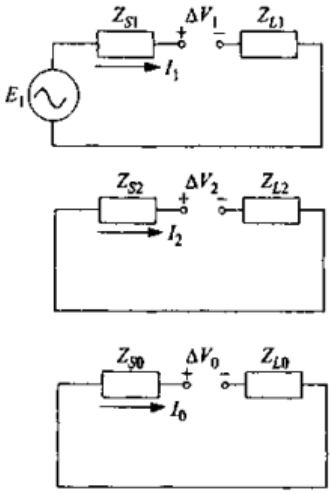
Esse comportamento é especialmente utilizado em redes de alta tensão, onde o efeito do desbalanceamento das tensões pode ser mitigado. Para redes de média e baixa

tensão, equipamentos são frequentemente conectados a rede à partir de um transformador delta-estrela, neste caso, experimentam o valor em *pu* da tensão fase-fase no nível da média tensão.

A.2 Tensão após a falta

Quando a falta se extingue, a falta vai de um curto circuito para um circuito aberto. Com o caminho de baixa impedância mitigado, a tensão não tende mais a zero e agora depende das impedâncias equivalentes da fonte e da carga da linha de transmissão. Uma maneira de calcular esta tensão é a partir do método das componentes simétricas, que representa as componentes do sistema desbalanceado como a soma das componentes de 3 sistemas balanceados. A figura 34 ilustra estes sistemas.

Figura 34 – Componentes simétricas equivalentes



Fonte: (BOLLEN *et al.*, 2000)

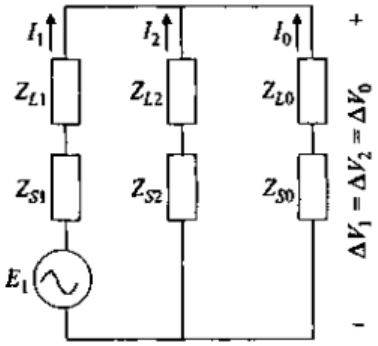
Z_{S1}, Z_{S2}, Z_{S0} são as componentes positiva, negativa e zero da impedância da fonte, Z_{L1}, Z_{L2}, Z_{L0} são as componentes positiva, negativa e zero da impedância da carga, ΔV_1, ΔV_2,ΔV_0 são as quedas de tensões em cada uma das componentes. $E_1 = 1$.

Para uma falta monofásica, as seguintes considerações são feitas para realizar o cálculo das componentes simétricas, sendo a a fase da falta, obtém-se então do comportamento do sistema $\Delta V_b = 0$, $\Delta V_c = 0$, $I_a = 0$. A equação A.3 representa essas considerações em formato de componentes simétricas

$$\begin{aligned} \Delta V_1 &= \Delta V_2 \\ \Delta V_1 &= \Delta V_0 \\ I_1 + I_2 + I_0 &= 0 \end{aligned} \tag{A.3}$$

Por fim, o curto pode ser representado com a topologia da figura 35 e a queda de tensão total ΔV_a pode ser representada pela equação A.4.

Figura 35 – Falta monofásica representada por componentes simétricas



Fonte: (BOLLEN *et al.*, 2000)

$$\Delta V_a = \Delta V_1 + \Delta V_2 + \Delta V_0 = \frac{3}{1 + \frac{Z_{L1}+Z_{S1}}{Z_{L0}+Z_{S0}} + \frac{Z_{L1}+Z_{S1}}{Z_{L2}+Z_{S2}}} \tag{A.4}$$

Através da análise de tensões durante e após a falta, é possível compreender como o comportamento do sistema se altera devido à presença ou remoção da baixa impedância causada pela falta. O uso de componentes simétricas permite modelar e calcular a distribuição de tensões e correntes durante a falta, fornecendo uma base sólida para a análise de sistemas desbalanceados.