Universidade de São Paulo
Escola de Engenharia de São Carlos

Daniel Fagundes Oliveira

A Classificação de Faltas em Linhas de Transmissão com Alta Penetração de Fontes Eólicas Interfaceadas por Inversores: uma abordagem via Machine Learning

São Carlos

2025

Daniel Fagundes Oliveira

A Classificação de Faltas em Linhas de Transmissão com Alta Penetração de Fontes Eólicas Interfaceadas por Inversores: uma abordagem via Machine Learning

Monografia apresentada ao de Curso de Engenharia Elétrica com Ênfase em Sistemas de Energia e Automação, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Mário Oleskovicz

São Carlos

2025

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Oliveira, Daniel
O48a A Classificação de Faltas em Linhas de Transmissão com Alta Penetração de Fontes Eólicas Interfaceadas por Inversores: uma abordagem via Machine Learning / Daniel Oliveira; orientador Mário Oleskovicz. São Carlos, 2025.

Monografia (Graduação em Engenharia Elétrica com ênfase em Sistemas de Energia e Automação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2025.

1. Classificação de faltas. 2. Machine learning. 3. PyCaret. 4. Transformada Discreta de Fourier. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Daniel Fagundes Oliveira

Título: "A Classificação de Faltas em Linhas de Transmissão com Alta Penetração de Fontes Eólicas Interfaceadas por Inversores: uma abordagem via Machine Learning"

Trabalho de Conclusão de Curso defendido e aprovado em 25/11/2025, com NOTA 9,0(nove, zero), pela Comissão Julgadora:

Prof. Associado Mario Oleskovicz - Orientador SEL/EESC/USP

Prof. Associado José Carlos de Melo Vieira Júnior - SEL/EESC/USP

Dr. Moisés Junior Batista Borges Davi - EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Professor Associado José Carlos de Melo Vieira Júnior

DEDICATÓRIA

Aos meus familiares, amigos e mestres que fizeram parte da minha formação pessoal e profissional

AGRADECIMENTOS

Agradeço ao Professor Doutor Mário Oleskovicz pela oportunidade oferecida em 2022, que resultou em três anos de cooperação, sendo este período fundamental para o meu desenvolvimento acadêmico e profissional.

A todos os professores e funcionários que passaram pela minha trajetória, deixo aqui registrada a minha imensa gratidão por aceitarem o desafio de lecionar e serem uma influência positiva na vida das pessoas.

Aos meus pais, Washington Oliveira e Lucélia Oliveira, por sempre estarem presentes e tornarem realidade o meu sonho de me formar como Engenheiro pela Universidade de São Paulo.

À minha companheira Cristiane Pires, por ser um dos grandes incentivos para o meu desenvolvimento intelectual e estar comigo nos momentos difíceis e também nos de celebração.

Por fim, agradeço aos meus amigos de graduação do ano de 2021, cujo apoio foi fundamental em diversos momentos de desafios impostos pela graduação e pelos momentos de descontração que tornaram a trajetória até aqui mais leve.

Resumo

Oliveira, D. **A Classificação de Faltas em Linhas de Transmissão com Alta Penetração de Fontes Eólicas Interfaceadas por Inversores: uma abordagem via Machine Learning** - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2025

A crescente integração de Fontes Eólicas Interfaceadas por Inversores (FEIIs) em linhas de transmissão impõe desafios significativos aos métodos convencionais de classificação de faltas, uma vez que as contribuições de curto-circuito dessas fontes são determinadas pelas estratégias de controle empregadas em seus inversores. Por isso, as contribuições para faltas das FEIIs diferem significativamente das de geradores convencionais, o que desafia métodos de classificação convencionais. Neste contexto, o Aprendizado de Máquina (AM) surge como uma abordagem promissora para superar essas limitações. Este trabalho avaliou sete classificadores quanto à sua capacidade de generalização frente a diferentes parâmetros de curtos-circuitos e terminais de medição. Os modelos foram treinados e testados com características extraídas por meio da Transformada Discreta de Fourier (TDF) de sinais de tensão e corrente obtidos em simulações computacionais no *software* PSCAD. A biblioteca PyCaret foi utilizada para automatizar o treinamento e a comparação sistemática dos classificadores. Os resultados demonstraram que o algoritmo Random Forest apresentou o melhor desempenho entre os métodos avaliados neste trabalho, com alta acurácia na maioria dos cenários, além de notável robustez à presença de ruído e capacidade de generalização para parâmetros de falta não observados durante o treinamento.

Palavras-chave: Classificação de faltas. Machine learning. PyCaret. Transformada Discreta de Fourier.

Abstract

Oliveira, D. **Exploring the Influence of Measurement Terminal and Fault Parameters on Fault Classification in Transmission Lines with High Penetration of IBWRs. A Machine Learning Approach** – Escola de Engenharia de São Carlos, University of São Paulo, São Carlos, 2025

The increasing integration of Inverter-Based Wind Resources (IBWRs) into transmission lines poses significant challenges for conventional fault classification methods, as the short-circuit contributions of these sources depend on the control strategies employed in their inverters. Therefore, the fault contributions from IBWRs differ significantly from those of conventional generators, challenging conventional classification methods. In this context, Machine Learning (ML) emerges as a promising approach to overcome these limitations. This work evaluated seven classifiers for their generalization performance across different short-circuit parameters and measurement terminals. The models were trained and tested with features extracted via the Discrete Fourier Transform (DFT) from voltage and current signals obtained in computational simulations using the PSCAD *software*. The PyCaret library was used to automate classifier training and systematic comparison. The results demonstrated that the Random Forest algorithm performed best among the methods evaluated in this work, achieving high accuracy across most scenarios and showing notable robustness to noise and generalization to fault parameters not observed during training.

Keywords: Fault classification. Machine learning. PyCaret. Discrete Fourier Transform.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AD	Árvores de Decisão
AM	Aprendizado de Máquina
CA	Corrente Alternada
CC	Corrente Contínua
DFT	*Discrete Fourier Transform*
DT	*Decision Trees*
DTFT	*Discret Time Fourier Transform*
ET	*Extra Trees*
FEIIs	Fontes Eólicas Interfaceadas por Inversores
GBC	*Gradient Boosting Classifier*
GFC	Geradores Full-Converter
GIDA	Geradores de Indução Duplamente Alimentados
HHT	*Hilbert-Huang Transform*
k-NN	*k-Nearest Neighbors*
LightGBM	*Light Gradient Boosting Machine*
LR	*Logistic Regression*
ML	*Machine Learning*
OCDE	Organização para a Cooperação e Desenvolvimento Econômico
OIE	Oferta Interna de Energia
IoT	*Internet of Things*
PSCAD	*Power Systems Computer Aided Design*
RF	*Random Forest*
RNAs	Redes Neurais Artificiais
STFT	*Short-Time Fourier Transform*
SVM	*Support Vector Machine*
TDF	Transformada Discreta de Fourier
TDW	Transformada Discreta de Wavelet
TF	Transformada de Fourier

Conteúdo

1 Introdução

Marcada por sua elevada taxa de renovabilidade, a matriz energética brasileira apresenta um perfil de sustentabilidade singular no panorama mundial. Conforme [1], o Brasil atingiu, em 2024, o marco histórico de 50,0% de sua Oferta Interna de Energia (OIE) suprida por fontes renováveis, um patamar significativamente superior à média global (14,3% em 2022) e aos países da OCDE (Organização para a Cooperação e Desenvolvimento Econômico) (13,2% em 2023). No mesmo ano, a OIE totalizou 322 Mtep (milhões de toneladas equivalentes de petróleo).

O diferencial da matriz brasileira reside na contribuição de fontes de baixo carbono, como a biomassa da cana (16,7%) e a hidráulica (11,6%). Conforme indicado na Figura 1, as fontes renováveis têm apresentado uma clara tendência de crescimento, enquanto a participação de fontes não renováveis, como petróleo e derivados e gás natural, tende a reduzir, caracterizando a mudança estrutural da matriz. A participação da energia eólica na OIE, por exemplo, saltou de 0,6% em 2015 para 2,9% em 2024, sendo um dos principais fatores da transição gradativa da matriz energética brasileira. Em termos absolutos, a oferta de energia eólica atingiu 9,3 Mtep em 2024, registrando um aumento significativo de 12,4% em relação ao ano anterior.

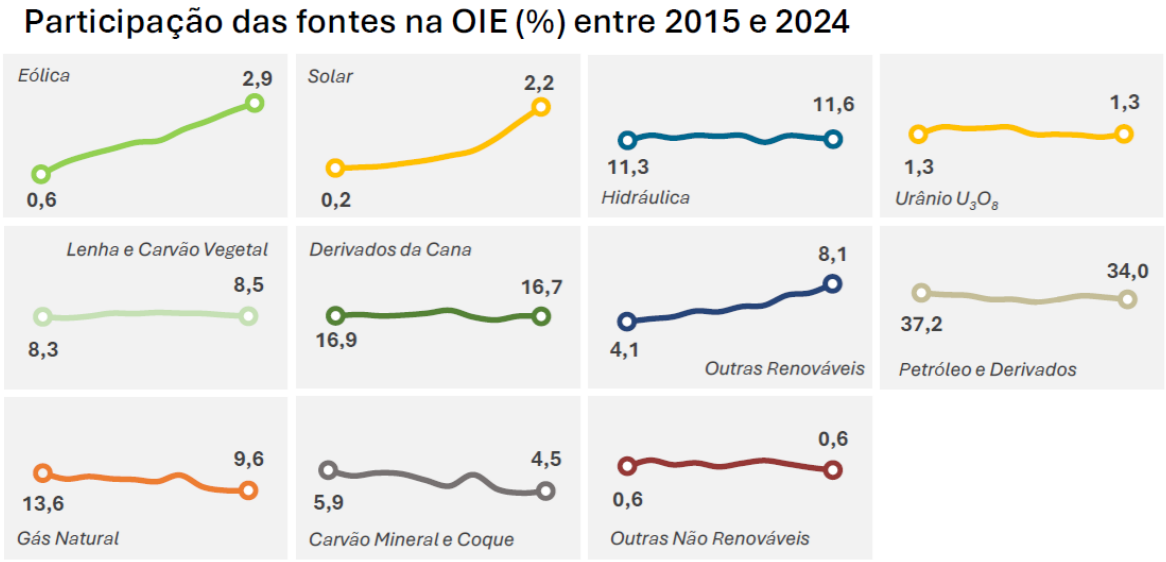

Figura 1: Contribuição de cada fonte na OIE brasileira na última década. Fonte: [1].

A relevância da fonte eólica é ressaltada ao analisar a matriz elétrica, que é consideravelmente mais limpa, atingindo 88,2% de participação de fontes renováveis em 2024. Neste setor, a energia eólica consolidou-se como um pilar estratégico. A Figura 2 confirma o salto das gerações eólica e solar na última década: a participação conjunta dessas fontes na geração total de eletricidade alcançou 23,7% em 2024, contra 7,2% em 2015. Esse crescimento foi crucial para a segurança energética do país, uma vez que a energia eólica atua como um excelente complemento à geração hidráulica. Ao operar em regimes de ventos muitas vezes opostos aos períodos de seca dos reservatórios, a energia eólica confere maior robustez ao Sistema Interligado Nacional (SIN), reduzindo a dependência do acionamento de termelétricas, que têm menor eficiência e emitem mais poluentes.

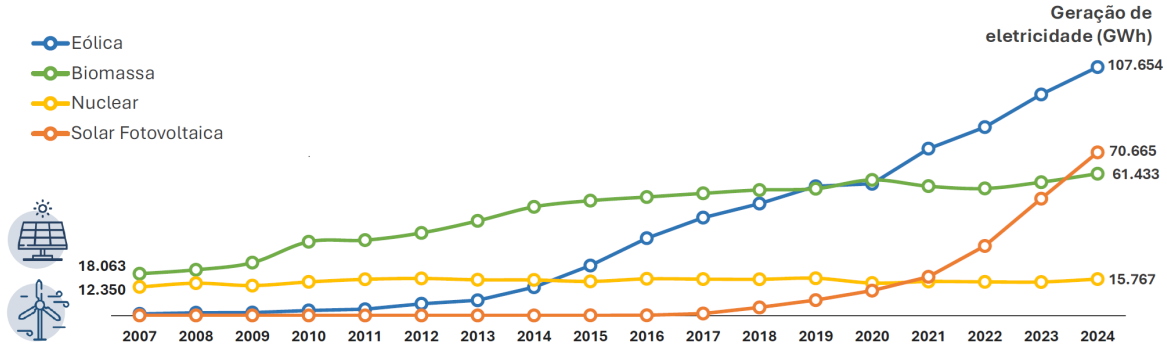

Figura 2: Geração de eletricidade proveniente de fontes renováveis no Brasil entre 2007 a 2025. Fonte: [1].

Em 2024, a geração eólica atingiu 107,7 TWh, representando um crescimento de 12,4% em relação a 2023, e sua potência instalada chegou a 29.550 MW, com uma expansão de 3% no ano. A energia eólica lidera entre as fontes renováveis não-hídricas e, juntamente com a solar fotovoltaica, responde por quase um quarto de toda a eletricidade gerada no país. A intensa renovabilidade da matriz elétrica, fortemente impulsionada pela energia eólica, resulta em um setor de baixíssimas emissões de carbono: em 2024, foram emitidos apenas 59,9 kg equivalentes de CO_2 por MWh gerado, um valor muito inferior ao de países como os EUA, a China e as nações europeias. A expansão contínua da energia eólica consolida-se, portanto, como um dos pilares centrais para que o Brasil mantenha sua liderança na transição energética global e cumpra seus compromissos climáticos.

1.1 Classificação de faltas em sistemas com alta penetração de FEIIs

A energia é o que move a sociedade moderna. Destarte, é de suma importância possibilitar que ela seja fornecida de forma estável, eficaz e sustentável. Neste contexto, ao longo das últimas décadas, devido à necessidade de redução do consumo de combustíveis fósseis e a busca por fontes de energia renováveis, observou-se uma crescente inserção de Fontes Eólicas Interfaceadas por Inversores (FEIIs) na rede convencional de transmissão de energia de diversos países ao redor do mundo [5], destacando-se duas topologias em especial: as FEIIs do tipo III, ou Geradores de Indução Duplamente Alimentados (GIDA), e as FEIIs de tipo IV, ou Geradores *Full-Converter* (GFC).

Inicialmente, quando a maior parte da geração disponível era proveniente de fontes convencionais, como usinas hidroelétricas, termoelétricas e nucleares, as FEIIs eram rapidamente desconectadas da rede principal em situações de perturbação. No entanto, com a crescente penetração das FEIIs nos sistemas elétricos, como mostra [6] no Brasil, a desconexão dessas fontes em caso de quaisquer distúrbios na rede tornou-se uma situação crítica para a estabilidade do sistema. Assim, foram adotados mundialmente os requisitos denominados *Fault Ride-Through* para que as FEIIs se mantivessem conectadas à rede primária, a depender de sua tensão terminal, mesmo sob perturbações [7]. Com a adoção destes requisitos, a avaliação das contribuições para os curtos-circuitos (faltas) provenientes das FEIIs passou a ser o foco de vários pesquisadores, por apresentarem características atípicas, determinadas pelos controles empregados nos inversores [8].

A classificação de faltas é um passo intermediário empregado por sistemas de proteção para detectar as fases em curto-circuito, a fim de garantir que apenas as fases necessárias sejam isoladas do sistema. Porém, diante deste contexto, observa-se que as FEIIs propõem desafios aos sistemas tradicionais de classificação, já que estes se baseiam nas características das contribuições para faltas de gerações síncronas convencionais, agora fortemente impactadas pela presença de FEIIs no sistema [9].

Assim, visando superar tais desafios, em [10], os autores foram pioneiros na utilização de métodos inteligentes para a classificação de faltas em sistemas com alta penetração de FEIIs. Entretanto, embora evidenciem o potencial desses métodos na classificação de

faltas em sistemas com FEIIs, apenas quatro métodos inteligentes foram avaliados, a saber: Árvores de Decisão (AD), Suport Vectors Machine (SVM), k-Nearest Neighbors (k-NN) e *Ensemble Trees*. Além disso, este estudo não aborda a capacidade de generalização desses métodos em relação aos parâmetros de falta e a presença de ruídos nos sinais, características importantes que testam a viabilidade de aplicação dos modelos em campo.

Para esta pesquisa, vale adiantar que foi utilizada uma base de dados com oscilografias de um estudo prévio [4], que engloba diversos cenários de falhas em um sistema com topologia tipicamente empregada na interconexão de FEIIs à rede. Para processar tais oscilografias, utilizou-se a TDF, a fim de construir uma base de dados para treinamento e testes de sete modelos inteligentes disponibilizados pelo PyCaret. O PyCaret é uma ferramenta que vem ganhando popularidade entre pesquisadores por disponibilizar vários recursos que facilitam a introdução ao tema de Aprendizado de Máquina (AM), simplificando os processos de construção e avaliação de métodos inteligentes e automatizando o treinamento, a validação e os testes, com poucas linhas de código. Além desses pontos, para esta pesquisa, o PyCaret foi escolhido por conter outros métodos ainda não contemplados pelos trabalhos já publicados na literatura, como o *Randon Forest Classifier*, o *Extreme Gradient Boosting* e o *Logistic Regression*, entre outros.

1.2 Objetivos

Tendo em vista os desafios que as FEIIs impõem aos sistemas de classificação de faltas convencionais e a escassez de trabalhos que utilizam métodos inteligentes para a superação desses desafios, este projeto propõe-se a:

- Avaliar a influência dos parâmetros de falta na função de classificação.
- Avaliar a influência do ruído sobre a função de classificação.
- Avaliar a influência do terminal do provedor de sinais de tensão e corrente no processamento de dados via TDF na função de classificação.
- Comparar os resultados de diferentes métodos inteligentes utilizando a biblioteca PyCaret.

Ao fim do trabalho, deseja-se avaliar a capacidade de generalização dos métodos, analisando em quais situações a metodologia de processamento e classificação produziu os melhores resultados.

2 Revisão Bibliográfica

Com o intuito de buscar trabalhos relacionados ao tema deste projeto, foi realizada uma pesquisa na base de dados *Scopus*, buscando por títulos, resumos e palavras-chave com a seguinte combinação de termos: **(“fault diagnosis” OR “fault classification”) AND (“Transmission Line” OR “Transmission System”)**. O resultado da pesquisa forneceu 2.379 documentos. Porém, ao adicionar, na sequência, os termos **AND (“inverter-based” OR “inverter-interfaced” OR “converter-based”)**, observaram-se apenas 38 documentos como resultado. Por fim, para alcançar o escopo principal do trabalho proposto, o termo **AND (”Machine Learning”)** foi adicionado à busca. Obteve-se, como resultado, somente 10 documentos, o que mostra que poucos trabalhos têm sido até então reportados, considerando os sistemas de controle de geradores interfaceados por inversores e utilizando uma abordagem baseada em ML.

No que tange à classificação de faltas em sistemas convencionais, um dos primeiros métodos foi reapresentado em [11], que se baseia na comparação de fasores de correntes superpostos e na corrente da componente de sequência zero fundamental. Conforme os autores destacam, um dos pontos negativos deste método é a necessidade de definição de limiares fixos para sua aplicação, os quais são consideravelmente afetados por mudanças na configuração do sistema. Apesar de ser simples e eficiente em cenários tradicionais, essa abordagem depende fortemente de ajustes empíricos e apresenta dificuldades em condições operacionais variáveis.

Com o avanço do processamento digital de sinais, surgiram técnicas mais robustas para extração de informações transitórias. Em [12], os autores propõem o uso dos coeficientes de detalhe da Transformada Discreta de Wavelet (TDW) aplicados a sinais de corrente obtidos de um terminal da linha de transmissão. Já em [13], três técnicas distintas de processamento (Transformada Discreta de Fourier, Transformada de Hilbert e Transformada de Stockwell Ortonormal) são comparadas e combinadas para a criação de índices representativos do tipo de falta. De forma complementar, [14] também utiliza a TDW, mas baseia a classificação na diferença dos coeficientes de detalhe de primeiro nível obtidos em ambos os terminais da linha.

De acordo com a revisão apresentada em [15], os métodos de processamento digital

de sinais têm sido amplamente utilizados para a detecção e classificação de faltas, sendo a TDW, a *Short-Time Fourier Transform* (STFT) e a *Hilbert-Huang Transform* (HHT) as mais empregadas. Os autores destacam que essas técnicas apresentam boa capacidade de caracterização temporal e espectral, mas sua eficácia depende fortemente da escolha dos parâmetros, como a função base e o tamanho da janela de análise. Além disso, a necessidade de limiares fixos e a sensibilidade ao ruído permanecem limitações comuns, justificando a transição para abordagens baseadas em aprendizado de máquina.

Com o avanço das técnicas de inteligência artificial, as metodologias baseadas em ML passaram a ser aplicadas com sucesso na área de proteção e diagnóstico de sistemas elétricos. O aprendizado de máquina pode ser definido como um processo computacional que permite a um algoritmo aprender padrões a partir de dados e realizar predições, sem a necessidade de modelar explicitamente o comportamento físico do sistema. Em outras palavras, o modelo é treinado com exemplos e ajusta seus parâmetros para melhorar o desempenho na tarefa desejada [16].

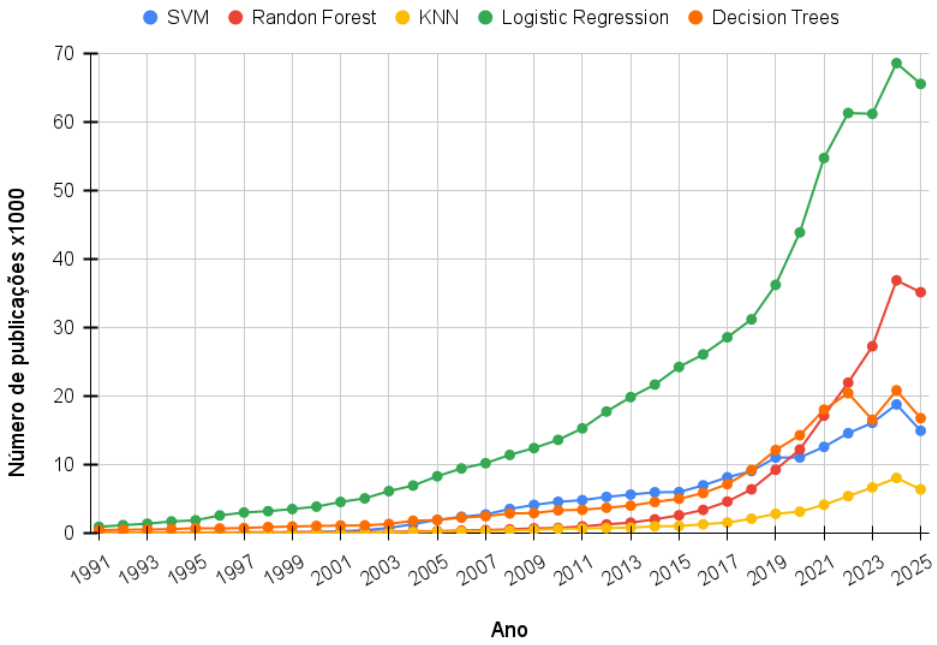

Figura 3: Número de publicações utilizando métodos diferentes de ML. Fonte: elaborado pelo autor.

Como mostra a Figura 3, o número de publicações que utilizam ML cresceu significativamente nas últimas duas décadas, com destaque para o uso dos algoritmos Logistic Regression (LR) e Random Forest (RF). Além disso, observa-se, conforme a Figura 4, o crescimento no uso de técnicas de aprendizado de máquina no contexto das fontes renováveis, reforçando a tendência de adoção dessas metodologias em estudos de sistemas com alta penetração de fontes eólicas.

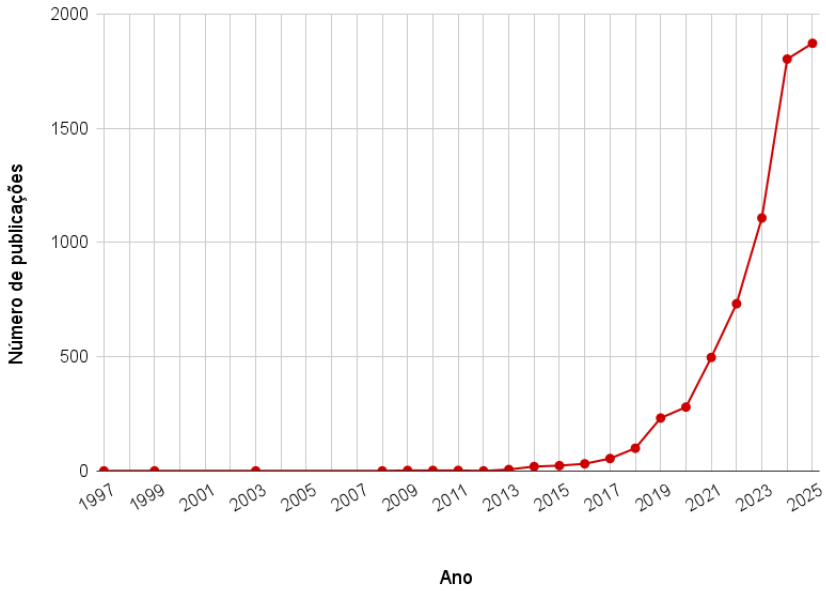

Figura 4: Número de publicações por ano em energia renovável utilizando ML. Fonte: Elaborado pelo autor.

No que tange à classificação de faltas, trabalhos recentes têm integrado o uso de ML a técnicas de processamento digital de sinais. Em [17], propõe-se o uso da Transformada de Stockwell Hiperbólica em conjunto com Redes Neurais Artificiais (RNA) para a classificação de faltas em sistemas de transmissão, alcançando bons resultados em termos de precisão e tempo de resposta. Em [18], os autores combinam a TDW com uma SVM, utilizando sinais de corrente provenientes de um único terminal da linha, reduzindo a necessidade de sincronização entre as medições. Mais recentemente, [4] aplica a TDW para a extração de características de sinais de tensão e corrente em sistemas com alta penetração de FEIIs, empregando cinco classificadores inteligentes, três baseados em AD e dois em regras de associação, com resultados promissores para a identificação de faltas.

De forma complementar, o estudo apresentado em [19] realiza uma avaliação sistemática de diferentes algoritmos de aprendizado de máquina aplicados à classificação de faltas em linhas de transmissão interconectadas a geradores baseados em inversores. Os autores comparam modelos como SVM, RNA e RF sob diferentes condições de operação e de ruído, destacando que o desempenho dos classificadores varia conforme as características dos sinais de entrada e a técnica de extração de atributos adotada. Os resultados apontam que modelos baseados em árvores apresentam maior estabilidade e interpretabilidade, enquanto RNAs e SVMs tendem a oferecer maior precisão em cenários com dados mais limpos e bem segmentados.

Em resumo, as FEIIs apresentam um comportamento dinâmico que se distingue substancialmente do das fontes síncronas convencionais. Seus inversores de potência limitam a corrente de curto-circuito, alterando significativamente sua composição harmônica e fasorial conforme a estratégia de controle adotada [20]. Esta característica compromete a eficácia dos métodos tradicionais de proteção, baseados em medições fasoriais.

Embora estudos recentes, como [4] e [19], demonstrem o potencial das técnicas de aprendizado de máquina na proteção de sistemas com alta penetração de FEIIs, lacunas de conhecimento relevantes ainda persistem na literatura. Dessa forma, são necessários trabalhos que abordem aspectos práticos da implementação dessas ferramentas em campo. Esses aspectos incluem: a capacidade de generalização para diferentes parâmetros de falta, a robustez dos classificadores diante da presença de ruídos nos sinais e a influência do terminal de medição. Paralelamente, também é necessário investigar um maior número de classificadores.

Este cenário justifica a proposta do presente trabalho de conclusão de curso, o qual se propõe a realizar uma análise conjunta dos principais fatores que influenciam o desempenho de classificadores na tarefa de classificação de faltas em sistemas com alta penetração de FEIIs, a saber: a influência da escolha do terminal de medição para a extração dos sinais de tensão e corrente, a influência dos parâmetros de falta e a presença de ruído nos sinais.

3 Fundamentos Teóricos

Para a condução deste trabalho, foram considerados os dois tipos de geradores mais comumente empregados em parques eólicos: os de Tipo 3 e 4. Na sequência, apresentam-se as principais características desses geradores. Além disso, apresentam-se os fundamentos básicos das técnicas de processamento, tanto convencionais quanto inteligentes, empregadas nesta pesquisa.

3.1 Geradores de Tipo 3, ou GIDA

Os geradores GIDA possuem enrolamentos trifásicos de corrente alternada (CA) tanto no estator quanto no rotor. O estator está conectado diretamente à rede elétrica, enquanto os enrolamentos trifásicos do rotor são ligados a um conversor de potência CA/CC por meio de anéis coletores, que fornecem a magnitude e a frequência variáveis à tensão do rotor. Outro conversor CC/CA é empregado para conectar o circuito do rotor à rede. Todo o circuito do gerador é conectado à rede por meio de um transformador elevador, como mostrado na Figura 5.

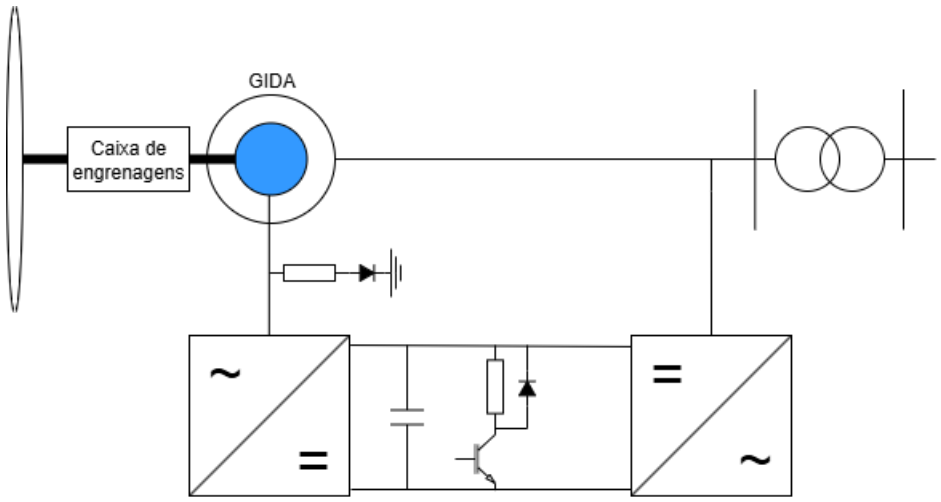

Figura 5: Topologia de um gerador de Tipo 3. Fonte: Adaptado de [2]

Quando a turbina eólica opera abaixo da velocidade síncrona (sub-síncrona), o campo magnético observado no estator é a soma da velocidade de rotação mecânica do rotor e da rotação aparente causada pela excitação CA aplicada. Neste modo de operação,

a potência de saída do estator alimenta a rede elétrica, enquanto o conversor fornece potência ao rotor.

Já quando a turbina opera acima da velocidade síncrona (super-síncrona), o campo magnético resultante no rotor gira no sentido oposto à rotação mecânica do próprio rotor. Neste caso, a potência entregue à rede é a soma da potência proveniente do estator e do rotor, por meio do conversor.

Existem dois blocos principais de controle nos geradores GIDA. O primeiro é o controle da velocidade de rotação da turbina eólica em conjunto com o controle de passo das pás, responsável pelo controle da potência ativa gerada. O segundo é o controle do conversor, que regula as potências ativa e reativa, ajustando a magnitude e o ângulo de fase da tensão do rotor. A potência reativa pode ser regulada diretamente ou utilizada para controlar a tensão terminal do gerador eólico.

A excitação de frequência variável no circuito do rotor permite a operação em uma ampla faixa de velocidades, e o fluxo de potência ativa no conversor pode ser bidirecional, dependendo se o gerador está operando acima ou abaixo da velocidade síncrona. Além das características de controle, essas características são responsáveis pela ampla utilização dos geradores do Tipo 3 em usinas eólicas devido ao seu modo de operação flexível e à eficiência [2].

3.2 Geradores de Tipo 4, ou GFC

Os GFC também são geradores de velocidade variável, conectados à rede por meio de dois conversores: o primeiro, do lado do gerador (CA-CC), e o segundo, do lado da rede (CC-CA).

O gerador produz uma corrente alternada que varia em função da velocidade da turbina eólica, sendo retificada pelo primeiro conversor. Já o conversor do lado da rede (ou inversor) converte a corrente contínua (CC) em corrente alternada (CA) à frequência da rede, de 50 Hz ou 60 Hz. A topologia de um gerador eólico do Tipo 4 é apresentada na Figura 6.

Assim como em geradores do Tipo 3, a potência ativa é controlada para ajustar a velocidade da turbina eólica e as cargas mecânicas, enquanto a potência reativa é utilizada

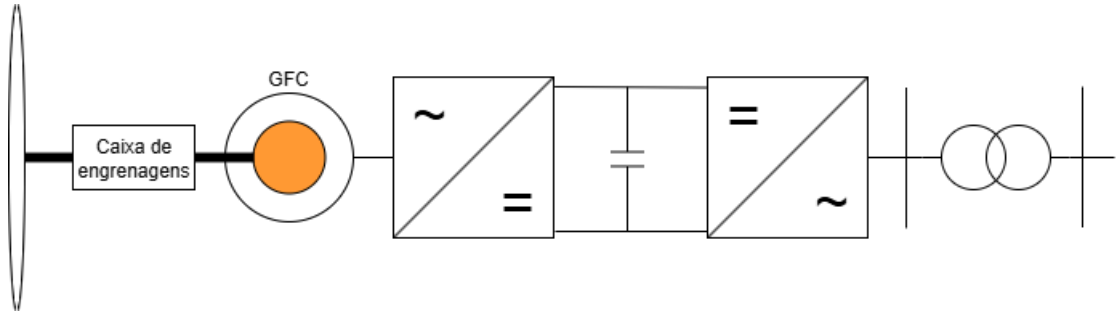

Figura 6: Topologia de um gerador de Tipo 4. Fonte: Adaptado de [2]

para regular a tensão. Porém, o conversor do gerador eólico Tipo 4 é dimensionado para operar com a potência nominal total da turbina eólica, enquanto os conversores de geradores de Tipo 3 operam na faixa de 20 a 30% da potência nominal total da turbina eólica [2].

3.3 Impactos das FEIIs em Métodos de Classificação de faltas

Segundo [20, 4], fontes convencionais são modeladas como fontes de tensão em série com impedâncias para a análise de faltas. Porém, as contribuições para as faltas de geradores de Tipo 3 e 4 podem ser significativamente diferentes, dependendo dos métodos de controle dos inversores, o que torna a análise mais complexa.

A contribuição de geradores GIDA durante faltas assume um caráter transitório, decrescente e potencialmente descontínuo. Este comportamento resulta da atuação dos circuitos de proteção *crowbar* ou *chopper*, que desviam as altas correntes de curto-circuito do rotor para fora do conversor, protegendo os dispositivos semicondutores contra sobreaquecimento e sobretensão no *link* CC. Já os geradores GFC limitam suas contribuições a até 1,2 p.u. [8], devido a restrições térmicas de seus componentes e apresentam comportamento variável conforme o controle utilizado. Este trabalho adota o Controle de Sequência Acoplado, no qual as contribuições para a falta possuem apenas componente de sequência positiva, mesmo em distúrbios assimétricos [20]. Entretanto, estudos recentes sugerem novos métodos de controle, como o Controle de Sequência Desacoplada, que permite corrente de sequência negativa em faltas assimétricas [20]. Vale lembrar também que vários países definiram códigos de rede que exigem, além da capacidade de *Fault*

Ride-Through, que as FEIIs forneçam determinados níveis de corrente reativa durante distúrbios, auxiliando no controle de tensão e frequência do sistema [4].

Além dos tipos de geradores empregados, outra fonte de complexidade na classificação de faltas em linhas de transmissão com alta penetração de FEIIs está relacionada aos terminais de medição dos sinais de tensão e corrente. Como mostrado na Figura 7, para uma falta do tipo AT, os sinais de corrente obtidos do lado das fontes eólicas apresentam uma assinatura harmônica com componentes de sequência zero expressivos, enquanto os sinais de corrente medidos pelo terminal remoto indicam que somente a fase faltante apresenta amplitude considerável, similar à das fontes convencionais.

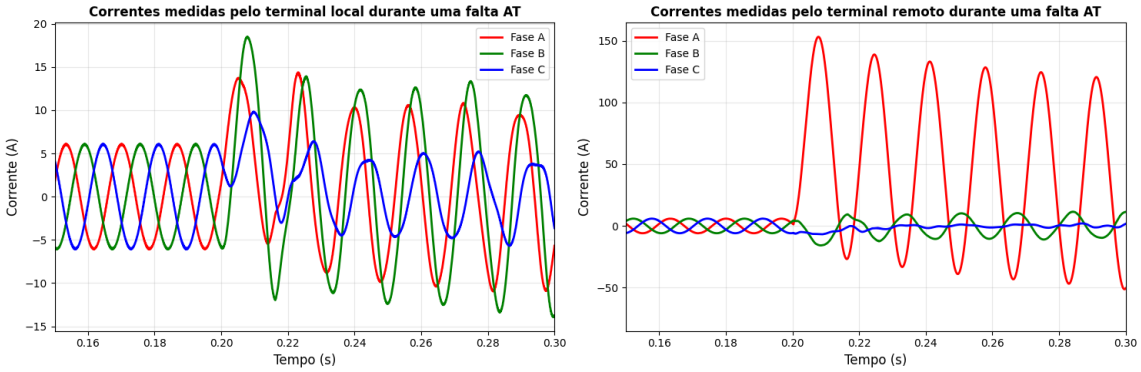

Figura 7: Sinais de corrente medidos em diferentes pontos do sistema durante uma falta do tipo AT. Fonte: elaborado pelo autor.

Outro exemplo é apresentado na Figura 8 para uma falta do tipo AB. Neste caso, do lado das FEIIs, observa-se uma contribuição de corrente trifásica equilibrada, devido à característica de supressão de corrente de sequência negativa, enquanto as medições feitas a partir do terminal remoto mostram que apenas as fases faltosas possuem amplitude expressiva.

Nesse contexto, métodos tradicionais de classificação de faltas, baseados nas características das correntes de falta produzidas por geradores síncronos, estão significativamente comprometidos devido às contribuições atípicas das FEIIs, determinadas pelos controles dos conversores [4].

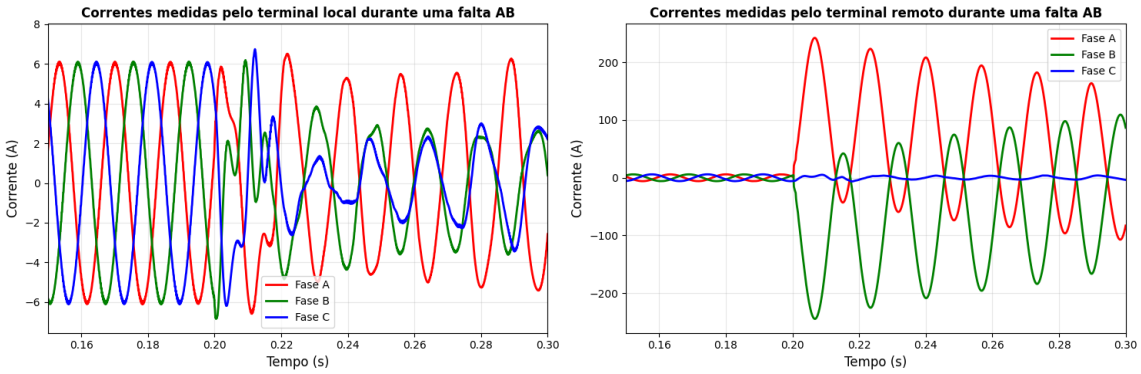

Figura 8: Sinais de corrente medidos em diferentes pontos do sistema durante uma falta do tipo AB. Fonte: elaborado pelo autor.

3.4 Transformada discreta de Fourier

A análise espectral é uma das ferramentas mais importantes no processamento de sinais, permitindo identificar e compreender os componentes de frequência presentes neles. Quando se trata de sinais discretos, obtidos por amostragem de sinais contínuos, a TDF tem sido, por muitos anos, a ferramenta mais utilizada na análise de sinais.

A versão clássica da Transformada de Fourier (TF) mostra que sinais contínuos no tempo podem ser descritos como a soma de uma série de funções seno e cosseno, formadas por diferentes combinações de amplitude, frequência e fase, conforme demonstra a Equação 1, que utiliza a forma exponencial das funções trigonométricas.

$$F(\omega) = Ff(t) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}\,dt \tag{1}$$

A *Discrete Time Fourier Transform* (DTFT) parte do princípio do Teorema da Amostragem, que estabelece que um sinal contínuo pode ser recuperado a partir de suas amostras sem perda de informação, desde que a taxa de amostragem seja pelo menos o dobro da maior frequência presente no sinal. Quando aplicada à DTFT em um sinal contínuo amostrado no tempo, o resultado é um espectro periódico no domínio da frequência, com cópias do espectro original espaçadas pela frequência de amostragem. Se a taxa de amostragem for insuficiente, ocorre o fenômeno de *aliasing*, no qual as réplicas se sobrepõem e distorcem a informação original no domínio da frequência, como mostra a Figura 9.

Enquanto a DTFT descreve o espectro de sinais discretos de comprimento infinito de

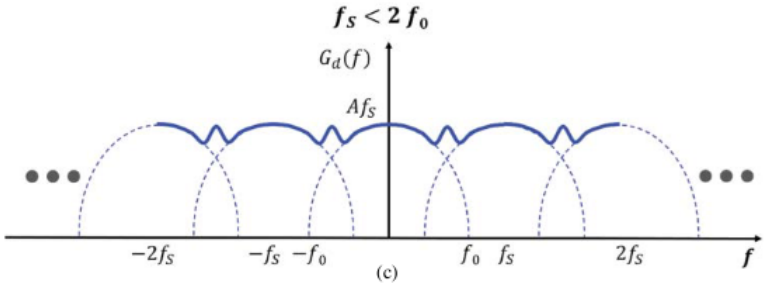

Figura 9: Sobreposição de espectros no domínio da frequência. Fonte: [3]

forma contínua no domínio da frequência, a TDF é válida para sinais de comprimento finito e produz uma representação espectral também discreta no domínio da frequência, transformando uma sequência de N valores no tempo em uma sequência de N valores complexos no domínio da frequência. Cada valor no espectro contém informações sobre a amplitude e a fase das componentes senoidais que compõem o sinal. Por ser discreta, a TDF também gera um espectro periódico, com período N.

Na prática, não é possível manipular digitalmente um sinal infinito. Por isso, a TDF é usada, pois ela amostra a DTFT em N pontos, igualmente espaçados, ao longo de um período do espectro. Esses pontos de frequência correspondem a múltiplos inteiros da frequência fundamental $\frac{2\pi}{N}$, onde N é o número de amostras no sinal. Este processo tem custo computacional reduzido, pois é realizado por meio de uma somatória finita, enquanto a DTFT exige integração.

Um exemplo prático da implementação da TDF é apresentado na Figura 10, para um sinal composto por dois cossenos com frequências de 20 e 60 Hz, com amplitudes de 5 e 2, respectivamente. É importante ressaltar que somente a parte positiva do espectro de frequência é representada, pois o resultado da aplicação da TDF gera espectros simétricos.

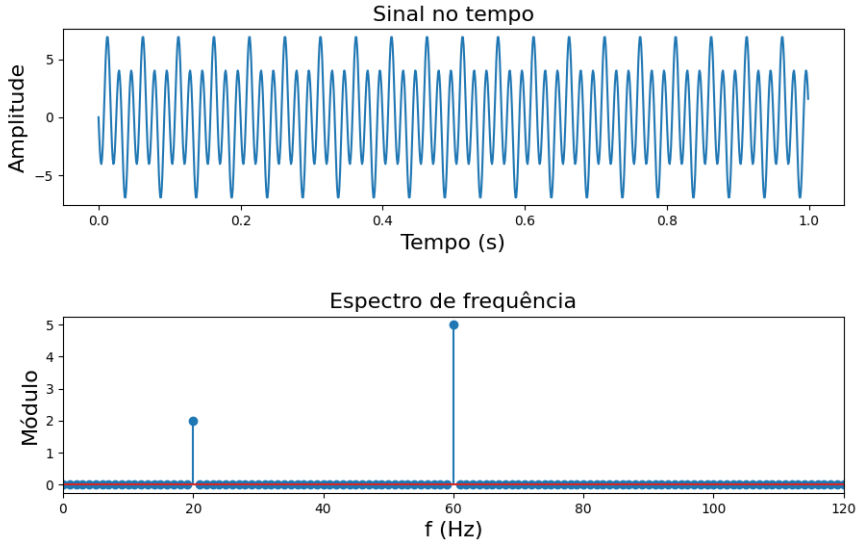

Figura 10: Exemplo de aplicação da TDF. Fonte: elaborado pelo autor.

3.5 Aprendizado de Máquina

O ML é um ramo da inteligência artificial que permite que sistemas computacionais aprendam padrões e tomem decisões com base em dados, sem necessidade de programação explícita. Esta técnica utiliza algoritmos capazes de prever resultados e de melhorar seu desempenho à medida que são retreinados com novos exemplos, o que é essencial para o desenvolvimento de tecnologias inteligentes.

Os modelos de classificação em ML são comparados e selecionados com base em um conjunto de características que equilibram o desempenho preditivo e a eficiência operacional. Estas são as quatro principais:

- **Custo Computacional:** Refere-se aos recursos (tempo, memória e poder de processamento) necessários para ajustar os parâmetros do modelo aos dados de treinamento. É medido em termos de complexidade temporal e é crucial para a escalabilidade.

- **Dimensionalidade:** Um modelo com alta capacidade de dimensionalidade é capaz de manter a robustez e o desempenho em conjuntos de dados com um número muito elevado de variáveis de entrada.

- **Latência:** É a velocidade com que o modelo gera uma classificação para uma nova amostra. Fator vital para aplicações em tempo real.
- **Interpretabilidade:** Descreve a facilidade de compreender o raciocínio subjacente à decisão de classificação do modelo.

Para a execução deste trabalho, foram considerados sete classificadores disponíveis na biblioteca PyCaret, a saber: Decision Tree (DT), Extra Trees (ET), Gradient Boosting (GBC), K-Nearest Neighbors (KNN), Light Gradient Boosting Machine (Light GBM), LR e RF. Estes modelos foram selecionados com base em um estudo anterior que testou duas metodologias de extração de características e analisou a influência do ruído na precisão dos classificadores. A abordagem revelou que a TDF, associada a este conjunto específico de modelos, mostrou-se promissora para a classificação de faltas em linhas de transmissão com alta penetração de FEIIs. Além disso, trabalhos recentes [4] mostram que o KNN e outros algoritmos baseados em DT são modelos promissores para a classificação de faltas em linhas de transmissão com alta penetração de FEIIs. A Tabela 1 apresenta as principais características desses classificadores e, em seguida, segue uma breve discussão sobre cada um.

Tabela 1: Principais características dos classificadores. Fonte: elaborado pelo autor.

Modelo	**Custo**	**Dimensionalidade**	**Latência**	**Interpretabilidade**
DT	Médio	Alta	Baixa	Alta
ET	Médio	Alta	Baixa	Média
GBC	Alto	Alta	Média	Baixa
k-NN	Baixo	Baixa	Alta	Alta
LightGBM	Baixo	Alta	Baixa	Baixa
LR	Baixo	Alta	Baixa	Alta
RF	Médio	Alta	Baixa	Média

3.5.1 *Decision Trees*

As árvores de decisão são amplamente utilizadas em estatística, mineração de dados e aprendizado de máquina por sua capacidade de representar, de forma intuitiva, o processo de tomada de decisão. Elas funcionam como modelos preditivos que utilizam uma estrutura hierárquica para relacionar observações de entrada (nas bifurcações) a resultados ou classes-alvo (nas folhas). Quando a variável de destino assume valores discretos, o modelo é chamado de árvore de classificação. Enquanto isso, para valores contínuos, denomina-se árvore de regressão. Essa abordagem é valorizada por sua simplicidade, interpretabilidade e aplicação em diversas áreas, como no diagnóstico médico, na avaliação de crédito e em sistemas de apoio à decisão.

A estrutura interna de uma árvore de decisão segue uma topologia hierárquica, como ilustrado na Figura 11. Esta estrutura é composta por três tipos de elementos: o Nó Raiz (o ponto de partida que contém o conjunto de dados completo), os Nós Internos (onde são realizados os testes nas variáveis de entrada, gerando ramificações) e os Nós Folha (os nós terminais que representam a previsão final ou a classe de saída). O processo preditivo segue um caminho recursivo e sequencial, iniciando na raiz e percorrendo os nós até atingir uma folha.

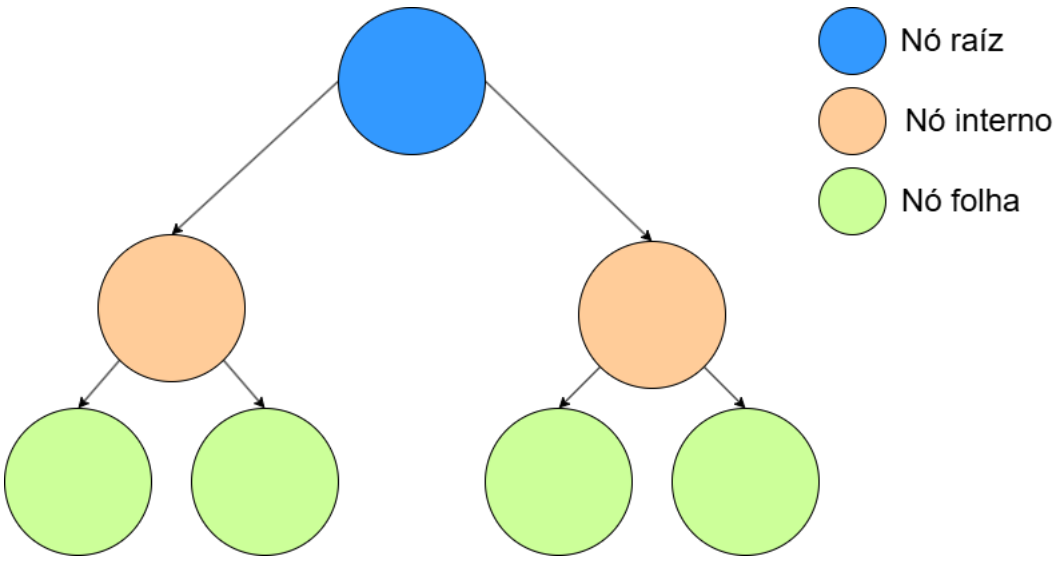

Figura 11: Topologia simplificada de uma *Decision Tree*. Fonte: elaborado pelo autor.

O processo de aprendizado das árvores de decisão é heurístico e baseado em uma busca

sequencial e ávida. A cada etapa, seleciona-se a melhor divisão possível, com base em métricas de impureza, como a entropia ou o índice de Gini.

- **Índice de Gini:** Mede a probabilidade de que uma amostra escolhida aleatoriamente seja classificada incorretamente, caso seja rotulada aleatoriamente com base na distribuição de classes no nó. O objetivo da divisão é minimizar o valor do índice de Gini na divisão resultante. É matematicamente definido como:

$$\text{Gini}(S) = 1 - \sum_{i=1}^{C} p_i^2 \tag{2}$$

- **Entropia:** Conceito adaptado da teoria termodinâmica para a Teoria da Informação, que mede o nível de desordem ou incerteza de um sistema. O algoritmo de treinamento busca maximizar o ganho de informação, que é a redução da entropia decorrente de uma divisão. Quanto mais homogêneo (puro) for um nó, menor será sua entropia. É expressa por:

$$\text{Entropy}(S) = -\sum_{i=1}^{C} p_i \log_2(p_i) \tag{3}$$

Onde:

- S: Nó para o qual a impureza é calculada.
- C: O número total de classes distintas presentes no conjunto S.
- p_i: A proporção de instâncias da classe i no conjunto S.

O critério de divisão que minimiza a impureza (Gini) ou maximiza o ganho (Entropia) é escolhido, e o processo se repete recursivamente até que os nós se tornem puros ou até que um critério de parada seja atingido. Essa busca não é retroativa, ou seja, não revisita decisões anteriores, mas tende a gerar modelos eficientes e robustos a ruídos nos dados. Além disso, as árvores podem ser expressas como conjuntos de regras do tipo “se-então”, o que facilita sua compreensão e explicação, consolidando-as como uma das técnicas mais populares e acessíveis no aprendizado supervisionado [21].

3.5.2 *Extra Trees*

O Extra Trees Classifier (Extremely Randomized Trees) é um algoritmo de aprendizado supervisionado que se baseia no conceito de métodos de conjunto (*ensemble learning*). Este modelo baseia-se em múltiplas árvores de decisão independentes e combina seus resultados para aprimorar o desempenho global da classificação. A principal característica desse método é a introdução de aleatoriedade adicional durante o processo de construção das árvores, o que o diferencia de modelos tradicionais, como o Random Forest. Em cada árvore do conjunto, é selecionado um subconjunto aleatório de variáveis e, diferentemente das árvores convencionais, que utilizam critérios como a entropia ou o índice de Gini para determinar o melhor ponto de divisão, o Extra Trees escolhe os pontos de divisão de forma totalmente aleatória no intervalo de valores das variáveis selecionadas.

Essa estratégia aumenta a diversidade entre as árvores do modelo, reduzindo a correlação entre elas e melhorando a capacidade de generalização. A predição final é obtida pela agregação das saídas individuais das árvores, de modo que a classe mais frequentemente prevista seja escolhida como resultado. Essa abordagem estocástica contribui para mitigar o *overfitting* que pode ocorrer em modelos de árvores de decisão isoladas, mantendo o equilíbrio entre precisão, robustez e interpretabilidade [22].

3.5.3 *Gradient Boosting*

O GBC constrói um modelo preditivo forte de forma sequencial, combinando vários modelos fracos, como árvores de decisão rasas. O processo começa com um modelo inicial simples que faz uma previsão básica sobre o conjunto de dados. A partir daí, o algoritmo calcula os erros desse modelo inicial, que, na verdade, correspondem aos gradientes negativos da função de perda.

Em seguida, uma nova árvore de decisão é treinada especificamente para prever esses erros. Essa nova árvore é, então, adicionada ao modelo cumulativo, mas com sua contribuição ponderada por uma taxa de aprendizado. Esse hiperparâmetro controla o quão rapidamente o modelo aprende a cada nova árvore, ajudando a evitar o *overfitting*.

Esse ciclo de cálculo de erros e de treinamento de uma nova árvore para corrigi-los se repete por um número definido de iterações. No final, a previsão do classificador é a soma

das previsões das árvores individuais, resultando em um modelo final preciso e robusto. A eficiência do algoritmo deve-se à sua capacidade de focar nos erros mais significativos a cada passo [23].

3.5.4 *K-Nearest Neighbors*

O KNN é um método de aprendizado de máquina supervisionado e não paramétrico, baseado em agrupamento, que faz suas previsões com base na proximidade dos novos dados em relação aos dados de treinamento existentes, em contraste com outros modelos que constroem uma função preditiva durante o treinamento [24].

Em tarefas de classificação, o KNN determina a classe de um novo dado com base nas amostras mais próximas no conjunto de treinamento. Para isso, calcula-se a distância entre a nova amostra e os dados existentes, geralmente utilizando a distância euclidiana quando as variáveis de entrada são contínuas. Após identificar os K vizinhos mais próximos, o algoritmo atribui à nova amostra a classe mais frequente entre eles.

O valor de K é um parâmetro crucial, pois influencia diretamente o desempenho do modelo: valores muito baixos podem tornar o classificador sensível a ruídos, enquanto valores muito altos podem gerar classificações imprecisas. Por isso, costuma-se testar diferentes valores de K para determinar o mais adequado.

Além da classificação, o KNN também pode ser utilizado em regressão, na qual o valor previsto é a média dos valores dos K vizinhos mais próximos. O método é especialmente útil em bases de dados em que as amostras se organizam em grupos bem definidos e é adequado para situações em que há pouco conhecimento prévio sobre os dados utilizados.

3.5.5 *Light Gradient Boosting Machine*

O LightGBM é um algoritmo baseado no método GBC, desenvolvido pela Microsoft em 2017. Assim como o GBC, ele constrói árvores de decisão de forma sequencial, nas quais cada árvore tenta corrigir os erros das anteriores, mas foi projetado para superar limitações de memória e de desempenho em grandes conjuntos de dados.

O LightGBM utiliza um método baseado em histogramas, discretizando os valores contínuos em *bins* e construindo histogramas que armazenam informações, como a soma

dos gradientes e o número de amostras. Essa abordagem reduz significativamente os custos computacionais e de armazenamento, pois os pontos de divisão são avaliados apenas entre os limites discretos dos bins. Além disso, emprega um crescimento *leaf-wise* com limite de profundidade, expandindo sempre a folha que reduz mais o erro, o que aumenta a eficiência do aprendizado [25].

Comparado ao GBC, o LightGBM apresenta maior eficiência e escalabilidade, permitindo lidar com bases de dados extensas e de alta dimensionalidade sem grandes impactos na memória ou no tempo de treinamento. Assim, otimiza-se o processo, tornando-o mais rápido e preciso, especialmente em cenários de grandes volumes de dados e alta complexidade.

3.5.6 *Logistic Regression*

O modelo de Regressão Logística é um algoritmo de classificação supervisionada utilizado para prever a probabilidade de um evento binário (por exemplo, sucesso/falha, sim/não) com base em um conjunto de variáveis independentes. Diferente da regressão linear, que prevê valores contínuos, a regressão logística transforma a saída em uma probabilidade entre 0 e 1 usando a função sigmoide (ou logística). A função sigmoide é definida como:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{4}$$

Onde z é a combinação linear das variáveis independentes ($z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$). Essa transformação garante que a previsão esteja sempre no intervalo [0, 1] e pode ser interpretada como uma probabilidade.

Para classificar uma nova amostra, o modelo calcula essa probabilidade e aplica um limiar de decisão (geralmente 0,5) para determinar a classe final. Por exemplo, se a probabilidade prevista for maior que 0,5, a classe é considerada 1; caso contrário, é 0. O modelo ajusta os coeficientes β_i durante o treinamento, usando a máxima verossimilhança, para encontrar valores que convertam os dados de entrada na classificação correta da amostra [26].

Além da classificação binária, a regressão logística pode ser estendida para multiclasse

(regressão logística multinomial) ou utilizada para prever probabilidades condicionais de eventos, sendo amplamente aplicada em áreas como medicina, finanças, marketing e ciência de dados.

3.5.7 *Random Forest*

O RF é um modelo de classificação que combina múltiplas árvores de decisão, em que cada árvore é criada a partir de um vetor aleatório, amostrado de forma independente, mas com a mesma distribuição para todas as árvores. Para classificar um novo dado, o modelo utiliza a categoria mais prevista pelas árvores que o compõem para determinar a classe à qual a observação pertence. O erro de generalização desta técnica converge para um limite à medida que o número de árvores aumenta, o que significa que o *overfitting* não se torna um problema ao adicionar mais árvores.

A aleatoriedade está presente em dois aspectos da construção do modelo. O primeiro mecanismo é o *Bagging (Bootstrap Aggregation)*, em que cada nova árvore é treinada em um novo conjunto de dados amostrado com reposição a partir do conjunto de treinamento original. O segundo mecanismo, e o mais crucial, é a Seleção Aleatória de Características, pela qual, em cada nó da árvore de decisão, apenas um pequeno subconjunto aleatório de variáveis de entrada é considerado para encontrar a melhor divisão. Essa dualidade de aleatoriedade tem como objetivo fundamental minimizar a correlação entre as árvores e manter a força (precisão) dos classificadores individuais, sendo esses os fatores determinantes da performance do classificador [27].

4 Metodologia

A seguir, são descritos o sistema elétrico considerado neste estudo, assim como as principais etapas metodológicas envolvidas no trabalho, como o pré-processamento e a inserção de ruídos nos sinais, além do treinamento e testes dos classificadores, utilizando a biblioteca PyCaret.

4.1 Descrição do sistema de teste e segmentação do banco de dados gerado por simulações computacionais

A pesquisa visou avaliar os principais tipos de FEIIs, e, para isso, foram considerados os geradores eólicos das duas topologias mais usuais: GFC e GIDA. Para ambos os tipos de geração, a usina eólica foi modelada para fornecer, em regime permanente, potência ativa de 220,5 MW e potência reativa de 0 var. Para a modelagem das unidades GIDA, o controle dos geradores foi ajustado conforme descrito em [28] e [29]. Já para a modelagem dos GFCs, o ajuste de controle foi realizado conforme descrito em [30] e em [28]. O diagrama unifilar do sistema considerado está apresentado na Figura 12, e os parâmetros empregados estão na Tabela 2. A simulação considerada foi por parâmetros distribuídos, considerando os parâmetros de sequência R_+, R_o, L_+, L_o, C_+, e C_o, sendo, respectivamente, os parâmetros resistivos, indutivos e capacitivos de sequência positiva e zero.

As simulações foram realizadas considerando o sistema de transmissão com a topologia definida na Figura 12, para o qual foram simulados curto-circuitos na linha 1-2, variando o tipo da falta (monofásica, envolvendo uma fase e o terra (AT, BT e CT), bifásica, envolvendo duas fases ou duas fases e o terra (AB, AC, BC, ABT, ACT, BCT) e trifásica, envolvendo as três fases (ABC)); a resistência entre fases (R_p igual a 0, 1, 1,5 e 2 Ω); a resistência entre fases e terra (R_y igual a 0, 25, 50 e 100 Ω); o ângulo de incidência (0, 45 e 90 graus); e a localização da falta (de 0% a 100% da linha 1-2 em passos de 10%, sendo 0% o ponto P1). Vale ressaltar que a base de dados utilizada nesta abordagem é composta pelas oscilografias de ambos os geradores (tipos III e IV).

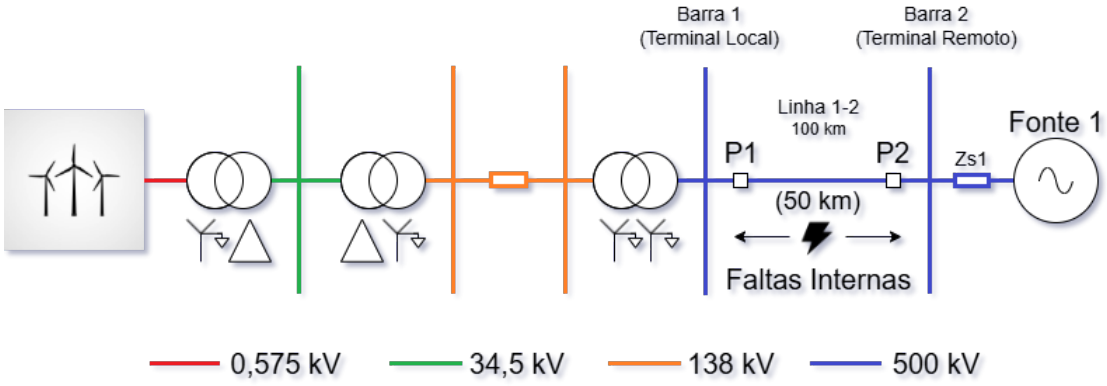

Figura 12: Topologia do sistema de teste. Fonte: Adaptado de [4].

Tabela 2: Parâmetros do sistema de teste. Fonte: Adaptado de [4].

Parâmetros	**Valores**
Fonte 1	$V_{s1} = 500\angle 0°$ kV $R_+/R_o = 0,984/3,447\ \Omega$ $L_+/L_o = 28,732/57,467$ mH
Transformador Dyn11 (34,5 – 0,575 kV)	1,75 MVA - Z = 6%
Transformador YNd1 (138 - 34,5 kV)	90 MVA - Z = 10%
Transformador YNyn0 (500 – 138 kV)	250 MVA - Z = 10%
Linha 1-2	$R_+/R_o = 0,017/0,331\ \Omega$/km $L_+/L_o = 0,839/2,382$ mH/km $C_+/C_o = 0,0137/0,0082\ \mu$F/km

Visando analisar a capacidade de generalização dos métodos inteligentes em relação aos parâmetros de falta, as bases de dados de treino e teste dos modelos foram segmentadas de modo que determinados valores das grandezas envolvidas nas simulações pertencessem somente a um único conjunto de dados. A Tabela 3 mostra como os dados de treinamento e teste foram organizados.

Tabela 3: Segmentação de dados para a composição das bases de dados de treino e teste.

	Treino	**Teste**
Local	1, 3, 5, 7, 9 e 11	2, 4, 6, 8, 10, e 12
Rp	1e-6, 1 e 2 Ω	1.5 Ω
Ry	1e-6, 25 e 100 Ω	50 Ω
Ang. de Incidência	$0,1°$ e $90,1°$	$45,1°$

4.2 Pré-processamento via a TDF

A ferramenta de extração de características utilizada foi a TDF, por ser amplamente empregada em diversas áreas, ser mais simples e ter menor custo computacional. A Figura 13 detalha as etapas para a obtenção de características utilizando a TDF.

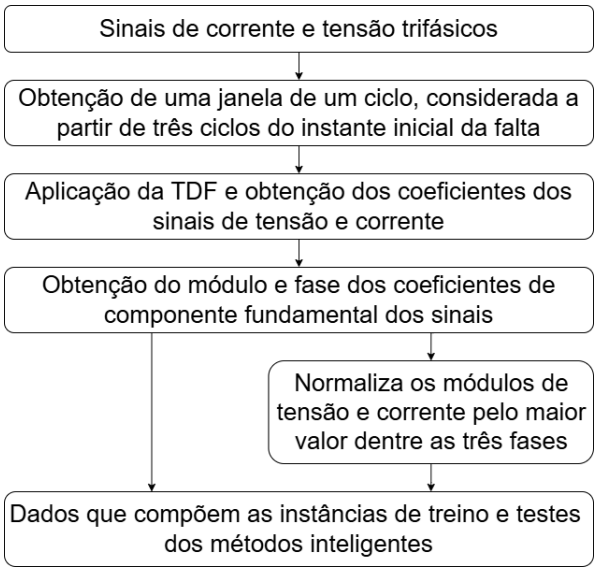

Figura 13: Extração de características via a TDF. Fonte: elaborado pelo autor.

Para a obtenção dos coeficientes dos sinais, foi utilizada a função ”fft.rfft()”da biblioteca ”scipy”, que retorna a primeira metade do vetor dos coeficientes da decomposição do sinal, já que este vetor é simetricamente espelhado, resultado da aplicação da TDF em um sinal amostrado no tempo. Em seguida, foi necessário normalizar o vetor pela metade do número de amostras e, ainda assim, dividir o primeiro coeficiente por dois. Com isso, obtém-se um vetor de números complexos, dos quais os módulos e fases dos fasores podem

ser obtidos aplicando, respectivamente, as funções ”np.abs()” e ”np.angle” da biblioteca ”numpy”.

4.3 Metodologia para a inserção de ruídos nos sinais

A fim de reproduzir uma situação típica de aplicação, onde os sinais estão constantemente submetidos a ruídos, foi utilizada uma metodologia para sobrepor os sinais obtidos pelas oscilografias a ruídos de diferentes intensidades, respeitando diferentes SNRs, do inglês *Signal to Noise Ratio*, que podem ser definidos como a razão entre a média da potência de um sinal ($\overline{Pot}_s$) e do ruído nele contido ($\overline{Pot}_r$). Ou seja:

$$SNR = \frac{\overline{Pot}_s}{\overline{Pot}_r} \tag{5}$$

Esta relação é mais comumente apresentada na escala de decibéis, definida como:

$$SNR_{dB} = 10Log(SNR) \tag{6}$$

Da Equação 6, é possível obter a média da potência do ruído a ser sobreposta no sinal, que é:

$$\overline{Pot}_r = \frac{\overline{Pot}_s}{10^{SNR_{dB}/10}} \tag{7}$$

Assim, podemos obter o valor RMS do ruído (RMS_r) a ser aplicado a um sinal, como sendo:

$$RMS_r = \sqrt{\overline{Pot}_r} \tag{8}$$

Com isso, pode-se utilizar a função ”np.random.normal()”para gerar números aleatórios com distribuição normal (gaussiana). A função tem três principais parâmetros:

- *loc* (média): Esse parâmetro define a média da distribuição normal, ou seja, o valor central em torno do qual os dados se distribuem.
- *scale* (desvio padrão): Este parâmetro define o desvio padrão da distribuição, que controla a dispersão dos valores em torno da média. É importante enfatizar que, neste caso, o desvio padrão será dado pelo valor calculado de RMS_r.

- *size* (forma ou quantidade de amostras): Esse parâmetro define o número de valores a serem gerados.

Um exemplo do resultado do processo de inserção de ruído é apresentado na Figura 14.

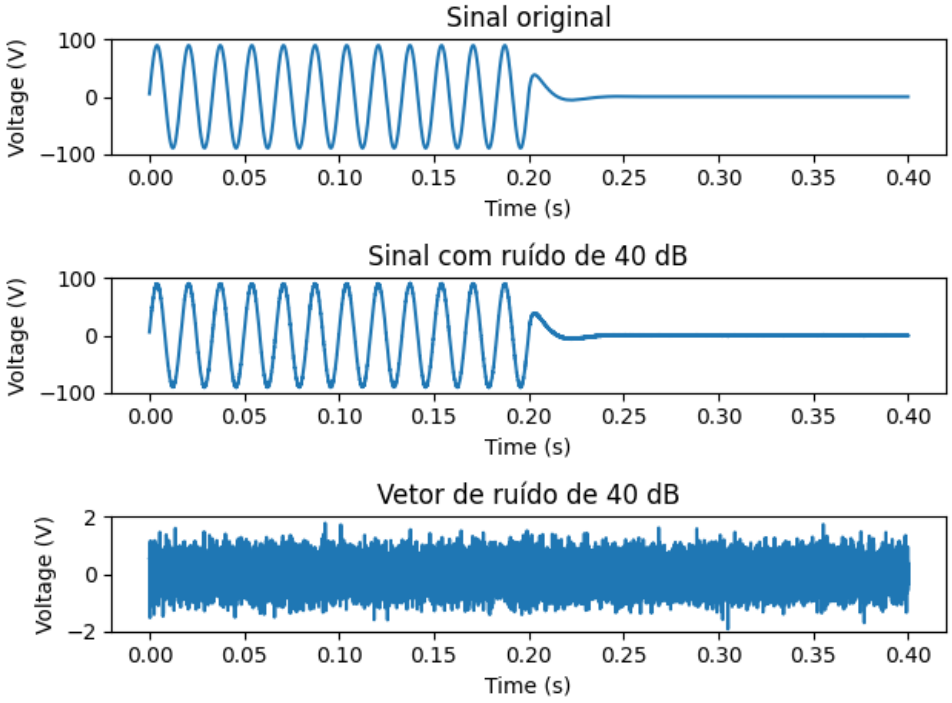

Figura 14: Gráficos de tensão obtidos antes e após a inserção de ruídos no sinal de tensão da fase "A", durante uma falta "AT". Fonte: elaborado pelo autor.

Os códigos utilizados para o pré-processamento dos dados via a TDF, com a inserção de ruídos, estão presentes no Apêndice.

4.4 A biblioteca PyCaret

PyCaret é uma biblioteca para *Python* que visa auxiliar na construção e na comparação de modelos de aprendizado de máquina. Primeiro, é necessário configurar o experimento, o que pode ser feito utilizando a *Functional API*, cuja função *setup* possui diversos parâmetros. A Figura 15 ilustra a chamada da função *setup* e seus atributos internos.

Os atributos da função *setup* são os seguintes:

- *index*: admite dois valores (*True e False*). Por padrão, o *index* é definido como *True*, considerando que o conjunto de testes será parte do conjunto de dados. Se forem utilizados arquivos diferentes para treinamento e teste, é necessário alterar o parâmetro para *False*.
- *data*: é a variável que receberá o conjunto de dados para treino;
- *train_size*: caso o usuário deseje utilizar o mesmo conjunto de dados para treino e teste, este parâmetro é utilizado para separar aleatoriamente uma porção do conjunto para treino e a outra para teste. Por exemplo, *train_size* = 0.7 equivale a 70% dos dados para treino e 30% para testes;
- *test_data*: é a variável que receberá o conjunto de dados para testes;
- *target*: indica qual coluna da tabela que o classificador deverá considerar no treino e nos testes.

Em seguida, para comparar diferentes classificadores, utiliza-se a função *compare_models*, cuja chamada e seus atributos são mostrados na Figura 16. Essa função treina e testa 14 classificadores disponíveis nativamente na biblioteca.

Os principais atributos da função *setup* são detalhados da seguinte forma:

- *n_select*: Define a quantidade de classificadores a serem retornados, em ordem do melhor para o pior.
- *sort*: ranqueia os classificadores com base em uma das métricas utilizadas pelo PyCaret;

```
from pycaret.classification import *
setup = setup(index = False, data = dados_treino , train_size = 1.0 , test_data = dados_teste, target = 'Classe')
```

Figura 15: Configuração da função *setup* do PyCaret. Fonte: elaborado pelo autor.

```
best_models = compare_models(n_select = 3, sort = 'Accuracy', cross_validation = False)
```

Figura 16: Instrução para treinar, testar e comparar modelos. Fonte: elaborado pelo autor.

- *cross_validation*: é um parâmetro que, por padrão, é configurado como *True*, e, para que os testes sejam realizados utilizando os dados de teste carregados, é necessário alterá-lo para False.

Após a execução do comando, o programa retorna uma lista com um ranking dos métodos utilizados com base na métrica selecionada no comando *compare_models*, podendo ser:

- Precisão (*Accuracy*): a precisão mede a proporção de predições corretas em relação ao total de predições. É uma métrica adequada quando as classes estão balanceadas. No entanto, pode ser enganosa em problemas com classes desbalanceadas.

$$ACC = TP/(TP + FP) \tag{9}$$

  Onde TP é o número de verdadeiros positivos e FP é o número de falsos positivos.

- Área sob a Curva ROC (AUC-ROC): a Curva ROC é uma representação gráfica que mostra a taxa de verdadeiros positivos (TPR ou Sensibilidade) em relação à taxa de falsos positivos (FPR) para diferentes valores de limiar de probabilidade. A Curva ROC pode ser criada calculando-se o TPR e o FPR para vários limiares de probabilidade, variando de 0 a 1. Quanto maior a área, melhor será o desempenho do modelo. O método de cálculo de AOC mais comum é a integração numérica. A AUC fornece uma medida útil do desempenho do modelo em relação à classificação binária

- Revocação (Recall ou Sensibilidade): é uma métrica de avaliação de desempenho usada em problemas de classificação, especialmente em situações em que a identificação dos verdadeiros positivos é fundamental. O recall é uma métrica importante em

problemas em que a detecção de exemplos positivos é crucial e onde a consequência de um falso negativo é significativa.

$$Recall = TP/(TP + FN) \tag{10}$$

Sendo FN o número de falsos negativos.

- F1-Score: o F1-Score é uma métrica de avaliação de desempenho que combina precisão (precision) e recall (sensibilidade) em um único número, sendo especialmente útil em problemas de classificação onde há um desequilíbrio entre as classes. O F1-Score fornece uma única métrica que equilibra a precisão e o recall. Ele é especialmente útil quando há um desequilíbrio entre as classes, pois não favorece nenhuma classe em particular.

$$F1 = (Prec \cdot Recall)/(Prec + Recall) \tag{11}$$

O valor do F1-Score varia de 0 (pior desempenho) a 1 (melhor desempenho), sendo um indicador geral de quão bem o modelo está funcionando em relação à classificação das classes positivas e negativas.

- Cohen's Kappa (Kappa): mede a concordância entre as classificações previstas por um modelo de AM e as classificações reais (rótulos verdadeiros). É especialmente útil quando se lida com problemas de classificação, onde se deseja avaliar o quão bem o modelo concorda com as categorias de classificação. O índice Kappa é particularmente útil quando se lida com problemas de classificação em que as classes estão desbalanceadas, pois leva em consideração a concordância esperada ao acaso, o que pode ajudar a evitar avaliações enganosas.

$$k = \frac{Po - Pe}{1 - Pe} \tag{12}$$

Onde Po é a proporção de casos em que o modelo e os rótulos verdadeiros concordam, e Pe é a proporção de concordância que seria esperada ao acaso. k>1 indica coerência perfeita entre as precisões e os rótulos verdadeiros. k=0 indica que o modelo está realizando tão bem quanto o esperado ao acaso, e k<0 indica que o modelo está performando pior do que o esperado ao acaso (discordância).

- Matthews Correlation Coefficient (MCC): é uma métrica de avaliação de desempenho amplamente utilizada em problemas de classificação binária. O MCC leva em consideração tanto verdadeiros positivos (*True Positives* - TP), verdadeiros negativos (*True Negatives* - TN), falsos positivos (*False Positives* - FP) quanto falsos negativos (*False Negatives* - FN) para calcular um único valor que reflete a qualidade geral das previsões de um modelo de AM.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{13}$$

  MCC = 1 indica uma concordância perfeita entre as previsões do modelo e os rótulos verdadeiros. MCC=0 indica que o modelo está desempenhando tão bem quanto o esperado pelo acaso. MCC=-1 indica uma discordância perfeita entre as previsões do modelo e os rótulos verdadeiros. O MCC é especialmente útil quando o desbalanceamento entre as classes é significativo ou quando o custo de cometer falsos positivos e falsos negativos é desigual. No entanto, é importante notar que o MCC é mais adequado para problemas de classificação binária e não se aplica diretamente a problemas de classificação multiclasse.

Durante a pesquisa, a única métrica considerada nos estudos foi a acurácia, dado que essa abordagem visa avaliar o desempenho global de diferentes modelos de classificação em um contexto de classificação multiclasse. Além disso, o banco de dados para treino e teste envolve todos os tipos de faltas, fazendo com que as classes estejam balanceadas entre si. Vale lembrar que sete modelos, dentre os quatorze disponíveis nativamente na biblioteca, foram considerados. Os modelos nativos disponíveis pelo PyCaret e suas siglas são apresentados na Tabela 4.

Tabela 4: Classificadores disponibilizados pelo PyCaret.

Sigla	**Classificador**
ET	*Extra Trees*
LightGBM	*Light Gradient Boosting Machine*
RF	*Random Forest*
GBC	*Gradient Boosting*
DT	*Decision Tree*
QDA	*Quadratic Discriminant Analysis*
KNN	*K-Nearest Neighbors*
LR	*Logistic Regression*
LDA	*Linear Discriminant Analysis*
NB	*Naive Bayes*
SVM	*Support Vector Machine*
Ada	*AdaBoost Classifier*
Dummy	*Dummy Classifier*
Ridge	*Ridge Classifier*

5 Resultados obtidos

Com o objetivo de avaliar a influência dos terminais de onde são obtidas as oscilografias utilizadas na extração de características sobre o desempenho dos classificadores, foram conduzidos nove testes distintos. Em cada teste, variaram-se os terminais responsáveis pela formação dos conjuntos de dados de treino e de teste, conforme a segmentação apresentada na Tabela 3. A configuração específica de cada teste é detalhada na Tabela 5. A seguir, são apresentados e discutidos a performance dos classificadores em cada teste e, na sequência, segue uma discussão geral dos resultados obtidos.

Tabela 5: Descrição dos testes realizados no trabalho.

Teste	**Descrição**
1	Treino e teste com medições em ambos os terminais
2	Treino com ambos os terminais e teste com o terminal local
3	Treino com ambos os terminais e teste com o terminal remoto
4	Treino com o terminal local e teste com ambos os terminais
5	Treino e teste com o terminal local
6	Treino com o terminal local e teste com o terminal remoto
7	Treino com o terminal remoto e teste com ambos os terminais
8	Treino com o terminal remoto e teste com o terminal local
9	Treino e teste com o terminal remoto

5.1 Teste 1: Treino e teste com medições em ambos os terminais

Os resultados obtidos no Teste 1 são apresentados na Figura 17. Neste caso, é possível observar que os modelos apresentaram um ótimo desempenho, com aproximadamente 100% de acurácia, com exceção dos Classificadores KNN e LR, que mantiveram uma taxa de acerto de 78% e 42%, respectivamente. Vale ressaltar que o modelo LR desempenhou melhor quando foi treinado e testado com sinais ruidosos, alcançando uma acurácia de 84%.

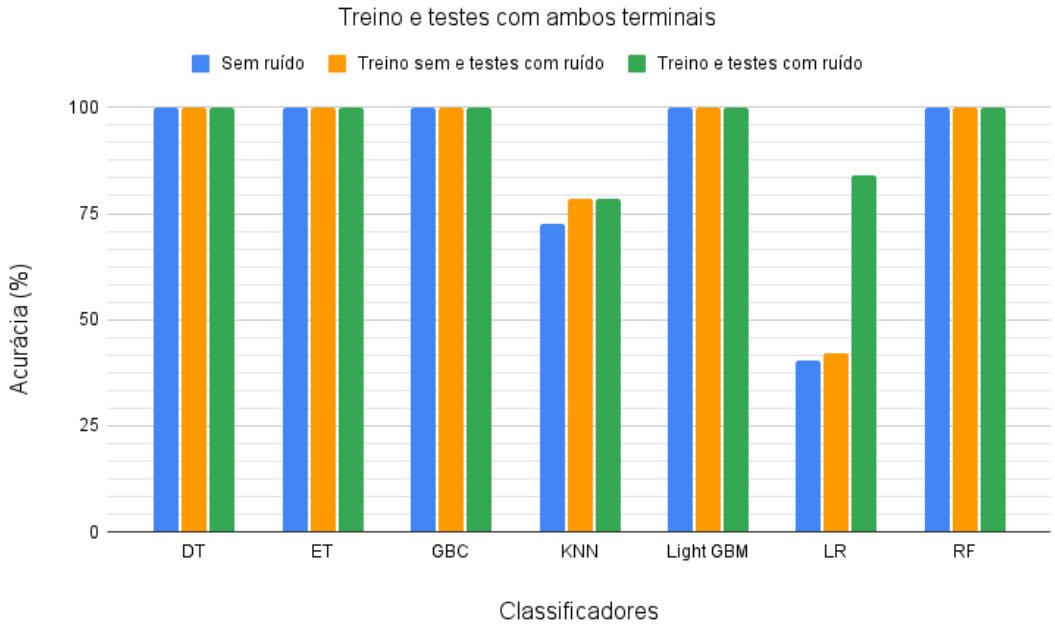

Figura 17: Resultados obtidos com ambos os terminais para treino e teste. Fonte: elaborado pelo autor

5.2 Teste 2: Treino com ambos os terminais e teste com o terminal local

De acordo com a Figura 18, os resultados obtidos utilizando ambos os terminais para treino e testando apenas com o terminal local não mostram alterações significativas na precisão dos classificadores quando comparados ao Teste 1. Porém, vale ressaltar que o desempenho do classificador LR caiu cerca de 20% quando treinado e testado com sinais sem a presença de ruído e, ao mesmo tempo, treinado com sinais sem ruído e testado com sinais contendo ruído.

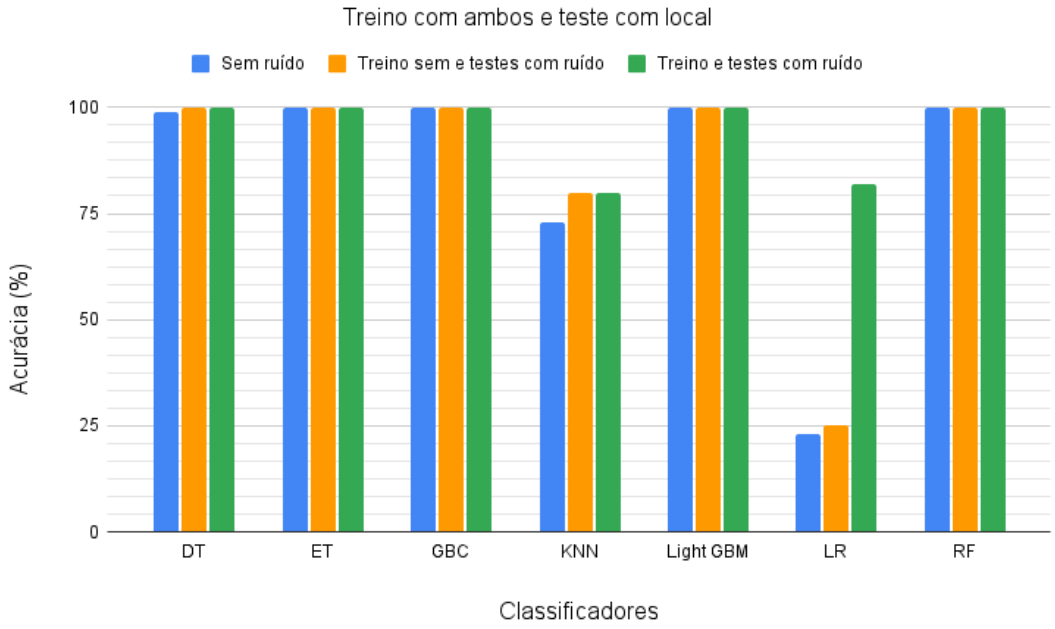

Figura 18: Resultados obtidos utilizando ambos os terminais para treino e o terminal local para teste. Fonte: elaborado pelo autor

5.3 Teste 3: Treino com ambos os terminais e teste com o terminal remoto

A última bateria de testes, utilizando ambos os terminais para o treino dos classificadores, apresentou níveis de desempenho semelhantes nos Testes 1 e 2, como mostrado na Figura 19. Os classificadores DT, ET, GBC, Light GBM e RF se mantiveram com altas taxas de acerto, enquanto KNN e LR se mantiveram com cerca de 80% de acurácia máxima.

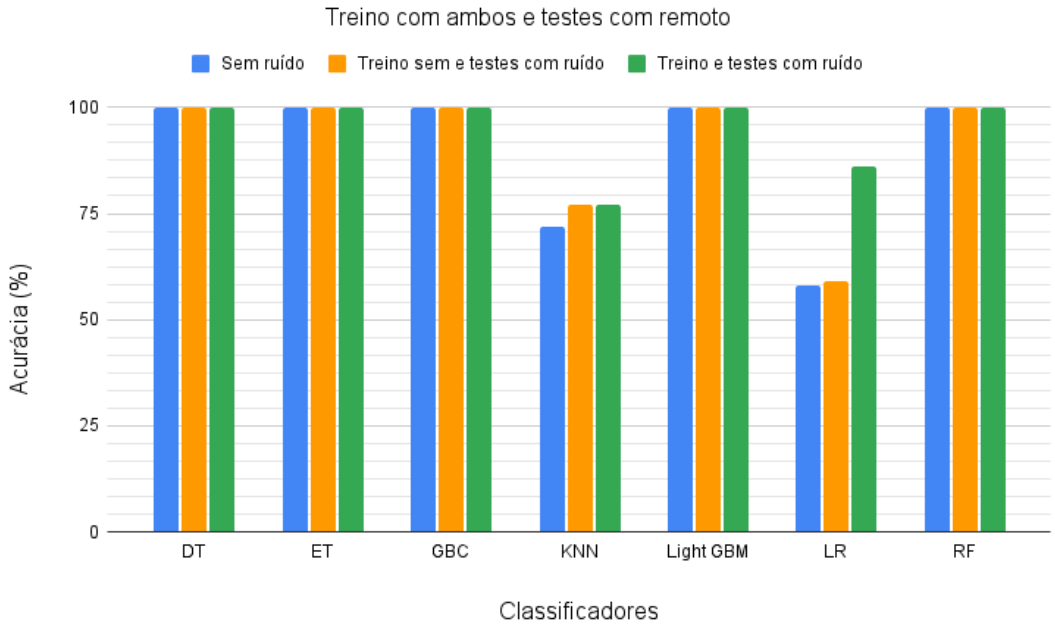

Figura 19: Resultados obtidos utilizando ambos terminais para treino e o terminal remoto para teste. Fonte: elaborado pelo autor.

5.4 Teste 4: Treino com o terminal local e teste com ambos os terminais

A Figura 20 mostra que o desempenho dos classificadores foi fortemente afetado quando treinados com dados do terminal local e testados com ambos os terminais. Os modelos DT, ET, GBC, Light GBM e RF que figuravam com taxas de acerto perto de 100% agora apresentam desempenho de cerca de 83%, 76%, 72%, 73% e 83%, respectivamente. O algoritmo KNN também perdeu 15% de acurácia, assim como a LR, com um decréscimo de 34%.

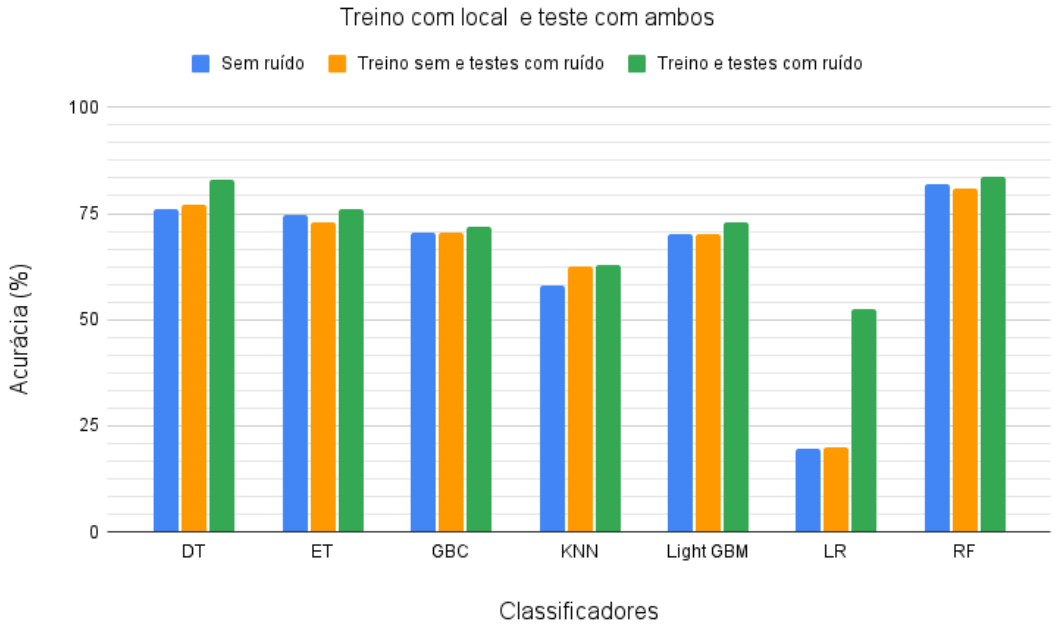

Figura 20: Resultados obtidos utilizando o terminal local para treino e ambos os terminais para teste. Fonte: elaborado pelo autor.

5.5 Teste 5: Treino e teste com o terminal local

Os resultados obtidos com o terminal local, para treino e teste, são apresentados na Figura 21. Como já era esperado, os classificadores voltaram a ter um bom desempenho, com DT, ET, GBC, Light GBM e RF apresentando acurácias de 100%, enquanto KNN e LR seguiram o mesmo comportamento dos testes anteriores, figurando com 80% e 84%, respectivamente.

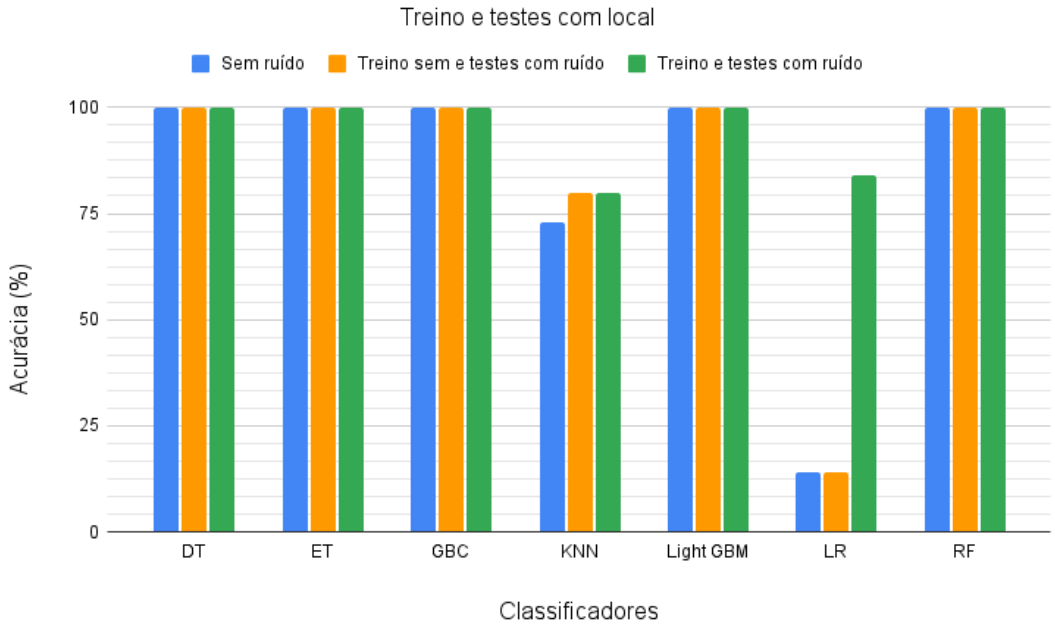

Figura 21: Resultados obtidos utilizando o terminal local para treino e teste. Fonte: elaborado pelo autor.

5.6 Teste 6: Treino com o terminal local e teste com o terminal remoto

Ao finalizar a bateria de testes no terminal local para o treinamento dos classificadores, é possível observar, conforme mostrado na Figura 22, que os classificadores são fortemente afetados pelos terminais a partir dos quais os sinais de tensão e corrente são extraídos. Os classificadores DT, ET, GBC e Light GBM obtiveram uma acurácia de 52%, 55%, 44% e 46%, respectivamente, enquanto KNN e LR obtiveram 46% e 26% de acurácia. O único classificador que ainda permaneceu com uma taxa de assertividade relativamente alta foi o RF, com 77%.

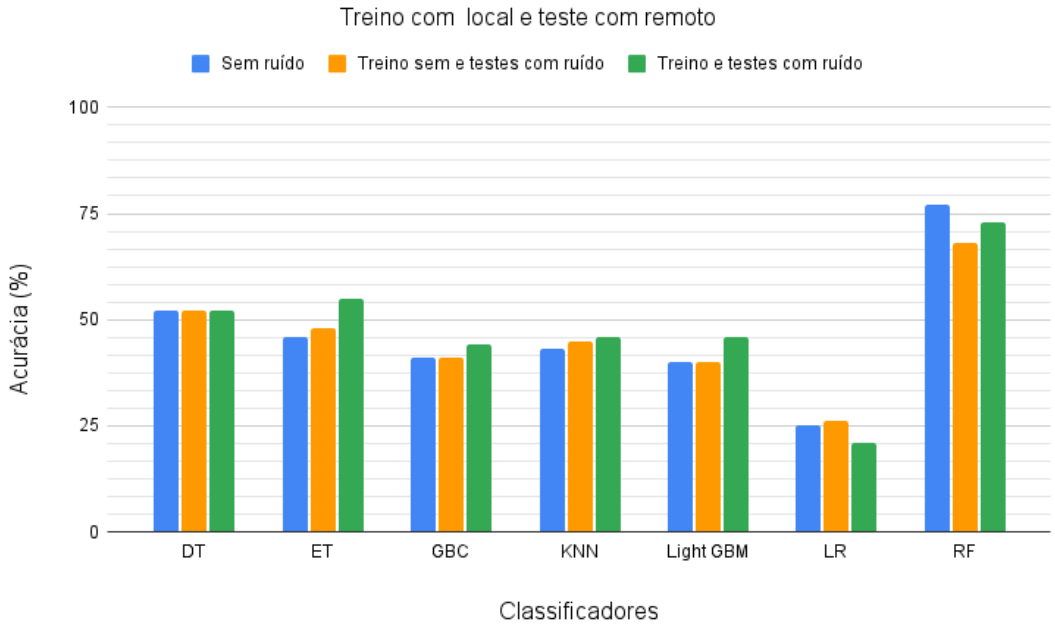

Figura 22: Resultados obtidos utilizando o terminal local para treino e terminal remoto para teste. Fonte: elaborado pelo autor.

5.7 Teste 7: Treino com o terminal remoto e teste com ambos os terminais

O primeiro teste, utilizando o terminal remoto para treino dos classificadores, é apresentado na figura 23. Neste caso, é possível observar que o desempenho dos métodos DT, ET, GBC, Light GBM e RF melhorou, se comparado com o teste anterior, o que já era esperado, com acurácias de 68, 78, 76, 78,5 e 79%, respectivamente. Enquanto KNN e LR figuraram com 38,5 e 58% de acurácia.

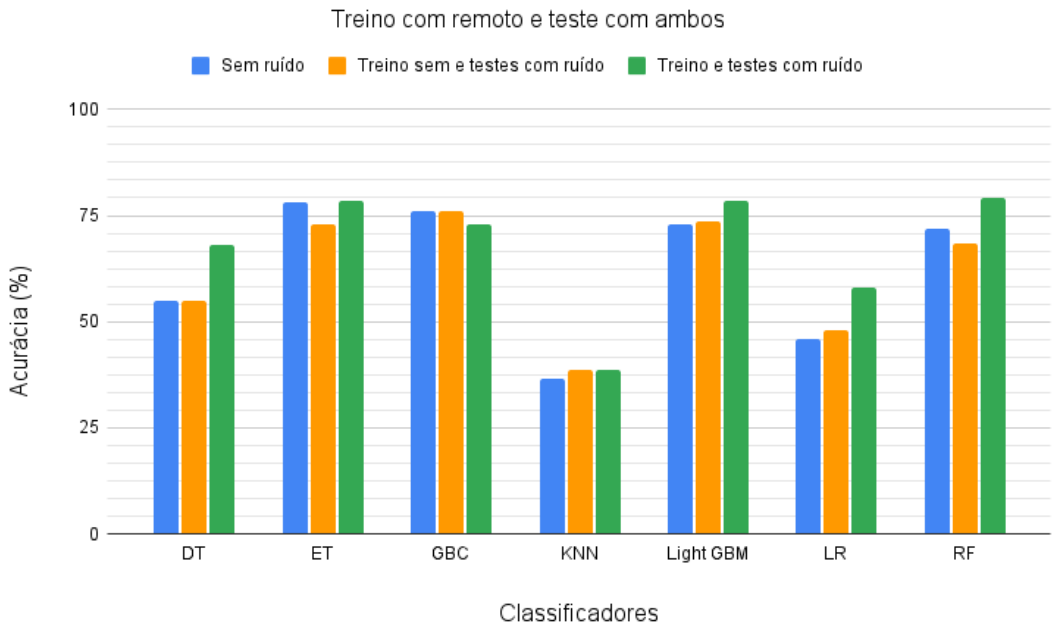

Figura 23: Resultados obtidos utilizando o terminal remoto para treino e ambos os terminais para teste. Fonte: elaborado pelo autor.

5.8 Teste 8: Treino com o terminal remoto e teste com o terminal local

Os resultados obtidos utilizando o terminal remoto para treino e o local para teste são apresentados. LightGBM e RF figuraram com 33%, 70%, 55%, 58%, 34% e 43% de acurácia, enquanto LR caiu para 34% na Figura 24. Ao analisar o desempenho dos classificadores, é possível deduzir que essa combinação configura o pior cenário, tanto para o treinamento quanto para os testes dos classificadores avaliados. Os classificadores DT, ET, GBC e Light GBM e RF figuraram com 33, 70, 55, 58, 34 e 43% de acurácia, enquanto LR caiu para 34%. Vale ressaltar que, neste caso, o classificador KNN não apresentou desempenho em nenhuma das abordagens, independentemente do nível de ruído inserido no sinal.

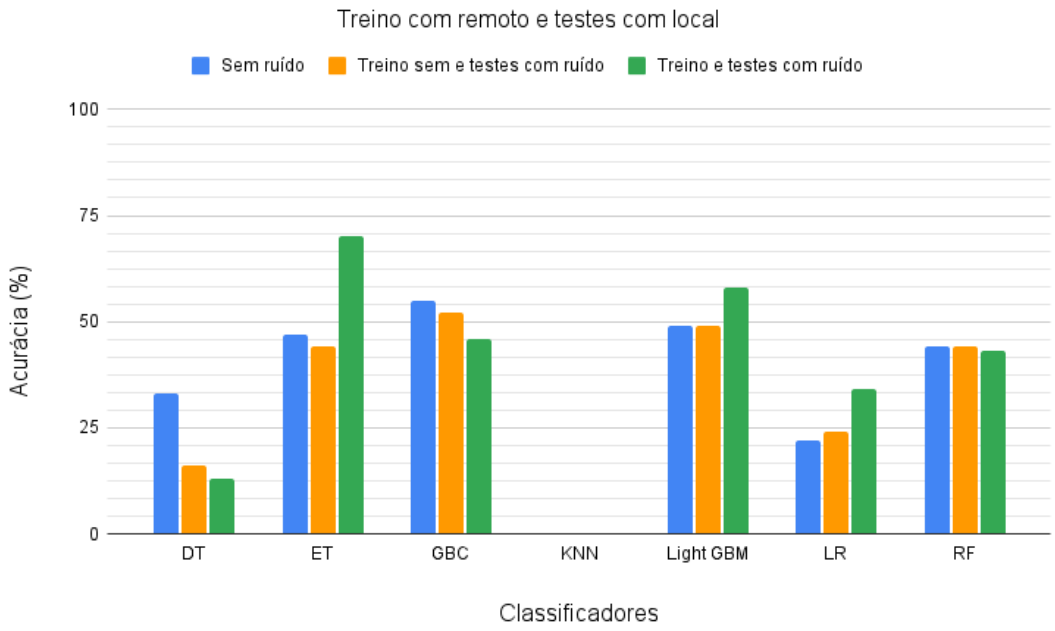

Figura 24: Resultados obtidos utilizando o terminal remoto para treino e o terminal local para teste. Fonte: elaborado pelo autor.

5.9 Teste 9: Treino e teste com o terminal remoto

Por fim, os resultados obtidos com o terminal remoto para treino e teste são apresentados na Figura 25. Como já era esperado, os classificadores voltaram a apresentar desempenhos semelhantes aos dos melhores casos, uma vez que a influência dos terminais foi removida. Neste teste, DT, ET, GBC, Light GBM e RF alcançaram novamente 100% de desempenho, enquanto KNN e LR alcançaram 77% e 82% de acurácia, respectivamente.

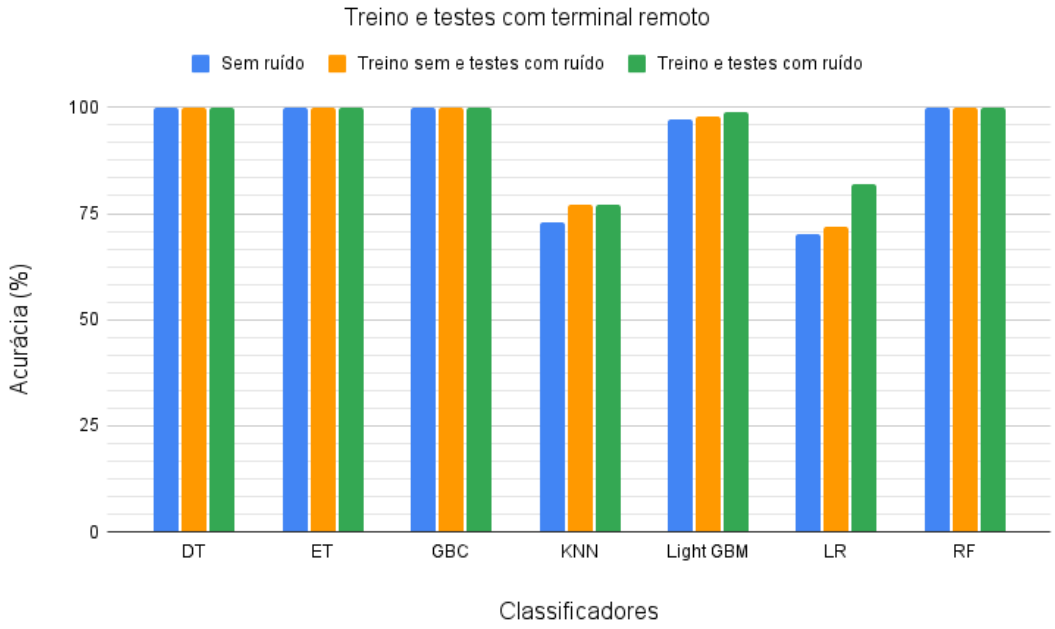

Figura 25: Resultados obtidos utilizando o terminal remoto para treino e teste. Fonte: elaborado pelo autor.

5.10 Análise Geral dos Classificadores

Em resumo, a Tabela 6 apresenta os percentuais máximos e mínimos de acerto de cada classificador para cada condição de teste avaliada. De acordo com os testes realizados, observa-se que a maioria dos métodos avaliados, quando treinados com características de ambos os terminais, é capaz de aprender tanto as contribuições das FEIIs (terminal local) quanto as dos geradores convencionais (terminal remoto) para a classificação das faltas. No entanto, esse comportamento não se mantém quando os classificadores são testados com características provenientes de um terminal que não está presente no conjunto de treinamento. Isso ocorre devido às diferenças nas contribuições de corrente de falta entre FEIIs e fontes convencionais, conforme ilustrado nas Figuras 7 e 8. Durante o treinamento, os classificadores ajustam seus hiperparâmetros para reconhecer os padrões característicos de cada tipo de fonte, tornando-se menos eficazes quando expostos a padrões distintos dos que constam nos dados de treinamento.

Tabela 6: Tabela resumo da acurácia obtida em cada teste (valores em porcentagem). Fonte: elaborado pelo autor.

Teste	DT	ET	GBC	KNN	Light GBM	LR	RF
1	Max: 100	Max: 100	Max: 100	Max: 78,5	Max: 100	Max: 84	Max: 100
	Min: 100	Min: 100	Min: 100	Min: 72,5	Min: 100	Min: 40,5	Min: 100
2	Max: 100	Max: 100	Max: 100	Max: 80	Max: 100	Max: 84	Max: 100
	Min: 99	Min: 100	Min: 100	Min: 73	Min: 100	Min: 40,5	Min: 100
3	Max: 100	Max: 100	Max: 100	Max: 77	Max: 100	Max: 86	Max: 100
	Min: 100	Min: 100	Min: 100	Min: 72	Min: 100	Min: 58	Min: 100
4	Max: 83	Max: 76	Max: 72	Max: 63	Max: 73	Max: 52,5	Max: 83,5
	Min: 76	Min: 73	Min: 70,5	Min: 58	Min: 70	Min: 19,5	Min: 82
5	Max: 100	Max: 100	Max: 100	Max: 80	Max: 100	Max: 84	Max: 100
	Min: 100	Min: 100	Min: 100	Min: 73	Min: 100	Min: 14	Min: 100
6	Max: 52	Max: 55	Max: 46	Max: 46	Max: 46	Max: 25	Max: 77
	Min: 52	Min: 46	Min: 41	Min: 43	Min: 40	Min: 21	Min: 68
7	Max: 68	Max: 78,5	Max: 76	Max: 38,5	Max: 78,5	Max: 58	Max: 79
	Min: 55	Min: 73	Min: 73	Min: 36,5	Min: 73	Min: 46	Min: 72
8	Max: 33	Max: 70	Max: 55	Max: -	Max: 58	Max: 34	Max: 44
	Min: 13	Min: 47	Min: 46	Min: -	Min: 49	Min: 22	Min: 43
9	Max: 100	Max: 100	Max: 100	Max: 77	Max: 99	Max: 82	Max: 100
	Min: 100	Min: 100	Min: 100	Min: 73	Min: 97	Min: 70	Min: 100

6 Conclusão

O presente trabalho investigou o desenvolvimento de uma ferramenta inteligente para a classificação de faltas em linhas de transmissão com alta penetração de FEIIs. Diante dos desafios impostos pela variação dos terminais de obtenção dos sinais de tensão e corrente, os resultados obtidos utilizando a TDF demonstram a viabilidade da utilização de classificadores baseados em árvore de decisão, que exibiram desempenho robusto e notável capacidade de generalização em relação aos parâmetros de falta e ao ruído presentes nos sinais.

Conclui-se que, caso haja infraestrutura para captação simultânea e processamento das medições dos dois terminais, é possível que apenas uma ferramenta inteligente seja treinada. Caso essa infraestrutura de comunicação não esteja disponível, recomenda-se o treinamento de uma ferramenta inteligente em cada terminal. Esta abordagem não apenas se alinha com a realidade operacional da maioria dos sistemas de proteção, que possuem medições primárias apenas em seus próprios terminais, mas também, conforme evidenciado pelos testes, assegura a aplicabilidade e a eficácia do modelo, sem depender de dados provenientes do terminal remoto, frequentemente indisponíveis ou com custos de comunicação proibitivos.

Dentre as técnicas avaliadas que utilizam a TDF para a extração de características, os algoritmos DT, ET, GBC e Light GBM destacaram-se positivamente. Estes modelos combinam alta precisão com características operacionais vantajosas, como redução do custo computacional, baixa latência, baixa dimensionalidade e nível satisfatório de interpretabilidade.

Contudo, para uma implementação em campo, o classificador RF destaca-se ligeiramente como a escolha mais promissora e robusta, uma vez que esse algoritmo não apenas compartilha as mesmas características de custo computacional, latência, dimensionalidade e interpretabilidade, mas também apresenta desempenho igual ou superior ao dos demais em todos os cenários avaliados, como evidenciado pelo teste 6. Esta estabilidade e confiabilidade são atributos críticos para um sistema de proteção, tornando o **Random Forest** o classificador mais recomendado para uma futura implementação prática.

Apesar das contribuições deste trabalho, os autores reconhecem que ainda há campos a

serem explorados para a incorporação de ferramentas inteligentes nos sistemas de proteção. Logo, como linhas de pesquisa futuras, pode-se destacar:

- **Validação do modelo Random Forest em ambiente de simulação em tempo real (*Hardware-in-the-Loop*)**: Esta validação pode ser implementada por meio da integração do classificador treinado a um simulador em tempo real, no qual o algoritmo seria executado em um dispositivo embarcado dedicado. O sistema receberia sinais de tensão e corrente gerados em tempo real pela simulação do sistema elétrico, processando-os continuamente para avaliar não apenas a precisão, mas também o tempo de resposta e a robustez computacional do modelo em condições que simulam a operação real.

- **Avaliação do desempenho do algoritmo em sistemas com diferentes topologias**: Esta análise envolveria a criação de múltiplos cenários de simulação que consideram diferentes topologias de sistemas. Para cada configuração, seriam gerados conjuntos de dados abrangentes por meio de simulações de faltas com variação sistemática dos parâmetros (localização, resistência, ângulo de incidência), permitindo quantificar a capacidade de generalização do classificador diante de mudanças estruturais no sistema.

- **Investigação de classificadores alternativos com diferentes técnicas de extração de características**: Um estudo comparativo poderia ser conduzido entre a abordagem TDF utilizada neste trabalho e métodos baseados em TDW, aplicando ambos a diversos algoritmos de classificação. Os testes deveriam incluir explicitamente cenários com ruídos aditivos e variações nos terminais de medição, permitindo uma avaliação abrangente do desempenho relativo das diferentes combinações característica-classificador.

- **Avaliação dos classificadores em diferentes pontos de operação dos geradores**: Por fim, uma possível abordagem para futuras pesquisas seria considerar os diferentes pontos de operação dos geradores eólicos, a fim de avaliar a capacidade de generalização dos classificadores diante de distintos perfis de geração.

Referências

[1] EMPRESA DE PESQUISA ENERGÉTICA. **Relatório Síntese 2025**, 2025.

[2] E. GURSOY. **Representation of Variable Speed Wind Turbine Generators for Short Circuit Analysis**. *IEEE Electrical Power and Energy Conference*, 2011.

[3] EMILIANO RESENDE MARTINS. ***Essentials of Signals and Systems***. John Wiley Sons, 2023.

[4] M. J. B. B. DAVI, M. OLESKOVICZ, and F. LOPES. **Exploring the Potential of a Machine Learning-Based Methodology for Fault Classification in Inverter-Based Resource Interconnection Lines**. *Electric Power Systems Research*, 2023.

[5] M. LISERRE, T. SAUTER, and J. HUNG. **Future Energy Systems**. *IEEE Industrial Electronics Magazine*, pages 18–37, 2010.

[6] EPE. **Balanço Energético Nacional 2021: Ano base 2020**, 2021.

[7] P. PIYA, M. EBRAHIMI, M. KARIMI-GHARTEMANI, and S. A. KHAJEHODDIN. **Fault ride-through capability of voltage-controlled inverters**. *IEEE Transactions on Industrial Electronics*, 65:7933–7943, 2018.

[8] IEEE-PES. **Impact of Inverter Based Generation on Bulk Power System Dynamics and Short-Circuit Performance**. *Report No. PES-TR68*, 2018.

[9] IEEE. **Fault current contributions from wind plants**. In *2015 68th Annual Conference for Protective Relay Engineers*, pages 137–227, 2015.

[10] K. AI KHARUSI, A. K.; EI HAFFAR, and M. MESBAH. **Fault Detection and Classification in Transmission Lines Connected to Inverter-Based Generators Using Machine Learning**. *MDPI*, 2022.

[11] D. V. COURY, M. OLESKOVICZ, and R. GIOVANINI. ***Proteção Digital de Sistemas Elétricos de Potência: dos relés eletromecânicos aos micro-***

processados inteligentes, page 378. Universidade de São Paulo (USP-EESC), São Carlos-SP, 2011.

[12] ALOK KUMAR, ADITYA, SAURAV RAJ, AMAN KUMAR SWARNKAR, KUNDAN BARNWAL, and SUDIPTA DEBNATH. **A Single Ended Wavelet Based Fault Classification Scheme in Transmission Line**. In *2018 IEEE Applied Signal Processing Conference (ASPCON)*, pages 29–33, 2018.

[13] D. GUPTA, O. P. MAHELA, and S. ALI. **Voltage Based Transmission Line Protection Algorithm Using Signal Processing Techniques**. In *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, pages 1–6, 2020.

[14] V. P. GOLI and S. DAS. **A transient current based DG Interconnected transmission system protection scheme using Wavelet analysis**. In *2020 IEEE Kansas Power and Energy Conference (KPEC)*, pages 1–6, 2020.

[15] H. AGARWAL, R. S. TIWARI, and J. P. SHARMA. **A Review of Signal Processing Techniques for Detection and Classification of Faults in Transmission Lines**. *3rd International Conference on Power Electronics and IoT Applications in Renewable Energy and its Control*, 2024.

[16] D. et al. SIHOMBING. **Design and Analysis of Automated Machine Learning (AutoML) in PowerBI Application Using PyCaret**. *International Conference of Science and Information Technology in Smart Administration (ICSINTESA)*, 2022.

[17] HSUEH-HSIEN CHANG, NGUYEN VIET LINH, and WEI-JEN LEE. **A Novel Nonintrusive Fault Identification for Power Transmission Networks Using Power-Spectrum-Based Hyperbolic S-Transform—Part I: Fault Classification**. *IEEE Transactions on Industry Applications*, 54(6):5700–5710, 2018.

[18] M. COBAN and S. S. TECZAN. **Detection and classification of short-circuit faults on a transmission line using current signal**. *Bulletin of the Polish Academy of Sciences: Technical Sciences*, 69, 2021.

[19] T. M. O. A. CUNHA, M. J. B. B. DAVI, and M. OLESKOVICZ. **Enhancing Power Grid Reliability: Evaluating Machine Learning Algorithms for Fault Classification in Inverter-Based Generators Interconnection Lines**. *International Conference on Harmonics and Quality of Power (ICHQP)*, 2024.

[20] THOMAS KAUFFMANN, ULAS KARAAGAC, ILHAN KOCAR, SIMON JENSEN, EVANGELOS FARANTATOS, ABOUTALEB HADDADI, and JEAN MAHSEREDJIAN. **Short-Circuit Model for Type-IV Wind Turbine Generators With Decoupled Sequence Control**. *IEEE Transactions on Power Delivery*, 34(5):1998–2007, 2019.

[21] K. POLAT and S. GÜNES. **The Effect to Diagnostic Accuracy of Decision Tree Classifier of Fuzzy and k-NN Based Weighted Pre-Processing Methods to Diagnosis of Erythemato-Squamous Diseases**. *Science Direct*, 2006.

[22] S. S. HUSSAIN and S. S. H. ZAIDI. **Robust Electric Motor Fault Classification with Extra Trees Classifier on Comprehensive Dataset**. *IEEE*, 2024.

[23] S. et al. EMAMI. **Multi-Task Gradient Boosting**. *18th International Conference*, 2023.

[24] K. et al. TAUNK. **A Brief Review of Nearest Neighbor Algorithm for Learning and Classification**. *IEEE*, 2019.

[25] M. et al. HAJIHOSSEINLOU. **A Novel Scheme for Mapping of MVT-Type Pb–Zn Prospectivity: LightGBM, a Highly Efficient Gradient Boosting Decision Tree Machine Learning Algorithm**. *Natural Resources Research, Vol. 32*, 2023.

[26] G. M. FITZMAURICE and N. M. LAIRD. **Binary Response Models and Logistic Regression**. *Elsevier*, 2015.

[27] L. BREIDMAN. **Random Forests**. *Machine Learning, 45, 5-32*, 2001.

[28] N. et al. MILLER. **Dynamic Modeling of GE 1.5 and 3.6 MW Wind Turbine-Generators for Stability Simulations**. *2003 IEEE Power Engineering Society General Meeting*, 2003.

[29] R. et al. GAGNON. **Large-Scale Real-Time Simulation of Wind Power Plants into Hydro-Québec Power System**. 2010.

[30] O. TREMBLAY, R. GAGNON, and M. FECTEAU. **Real-Time Simulation of a Fully Detailed Type-IV Wind Turbine**. 2013.

Apêndice 01 - Código implementado em Python para extração de características com inserção de ruídos via TDF

```
import numpy as np
import pandas as pd
import comtrade
from scipy.fft import rfft, rfftfreq


# criando uma tabela que relaciona cada simulação com um tipo de falta
with open('oscilografias\Casos.txt', 'r') as file:
    casos = file.read()
casos = casos.split()
casos = casos[6:] # excluindo a primeira linha (nome das colunas) lembrando
    que indice run = 0, indice tipo = 2
tipos = ['No Fault', 'AT', 'BT', 'CT', 'ABT', 'ACT', 'BCT', 'ABCT', 'AB', '
   AC', 'BC', 'ABC'] # o índice de cada elemento segue a convenção
   utilizada pelo PSCad para caregorizar as faltas
tabela_casos = []

## ================================================================
## ================================================================

# criando uma tabela com os tempos de faltas para cada tipo de gerador
tempos = pd.read_csv('oscilografias\Tempos_de_Falta.csv', delimiter = ',')
tempos = tempos.values.tolist()
tabela_tempos = []

## ================================================================
## ================================================================

for linha in tempos:
    tabela_tempos.append(linha)

vet = np.arange(0, len(casos),6)
for i in vet:
    linha = casos[i:i+6]
```

```
    linha[2] = tipos[int(float(linha[2]))]
    tabela_casos.append(linha)

# preparando para ler os arquivos
run_min = 1
run_max = 5280
lista_aux = list(range(run_min, run_max+1))
lista_run = []

# retirar os zeros do nome da pasta de cada Run
for numero in lista_aux:
    string_formatada = str(numero).zfill(5)
    lista_run.append(string_formatada)


ruido = [40, 50, 60]
terminal = [0, 6]
lista_tipo = ['3','4']

for ruido_dB in ruido:
    dados_csv = [['Modulo_Va', 'Fase_Va', 'Modulo_Vb', 'Fase_Vb', '
   Modulo_Vc', 'Fase_Vc',
                  'Modulo_Ia', 'Fase_Ia', 'Modulo_Ib', 'Fase_Ib', '
   Modulo_Ic', 'Fase_Ic',
                  'NModulo_Va', 'NModulo_Vb', 'NModulo_Vc', 'Nmodulo_Ia', '
   Nmodulo_Ib', 'Nmodulo_Ic', 'Classe']]

    for term in terminal:
        for tipo in lista_tipo:
            for run in lista_run:
                print('ruido:', ruido_dB, 'Tipo:', tipo, 'Run:', lista_aux[
   lista_run.index(run)], 'Classe:',
                      tabela_casos[lista_aux[lista_run.index(run)] - 1][2])

                caminho = "oscilografias\Type" + tipo + "\Rank_00001\Run_"
   + run + "\T4"
```

```
            cfg = caminho + ".cfg"
            dat = caminho + ".dat"

            rec = comtrade.load(cfg, dat)

            media_ruido = 0

            # Lendo e inserindo ruído ao sinal. 0 a 5 - local, 6 a 11 -
 remoto
            Va = np.array(rec.analog[0 + term])
            pot_Va = np.mean(Va ** 2)
            potencia_ruido = pot_Va / (10 ** (ruido_dB / 10))
            ruido_Va = np.random.normal(media_ruido, np.sqrt(
potencia_ruido), len(Va))
            Va = Va + ruido_Va

            Vb = np.array(rec.analog[1 + term])
            pot_Vb = np.mean(Vb ** 2)
            potencia_ruido = pot_Vb / (10 ** (ruido_dB / 10))
            ruido_Vb = np.random.normal(media_ruido, np.sqrt(
potencia_ruido), len(Vb))
            Vb = Vb + ruido_Vb

            Vc = np.array(rec.analog[2 + term])
            pot_Vc = np.mean(Vc ** 2)
            potencia_ruido = pot_Vc / (10 ** (ruido_dB / 10))
            ruido_Vc = np.random.normal(media_ruido, np.sqrt(
potencia_ruido), len(Vc))
            Vc = Vc + ruido_Vc

            Ia = np.array(rec.analog[3 + term])
            pot_Ia = np.mean(Ia ** 2)
            potencia_ruido = pot_Ia / (10 ** (ruido_dB / 10))
            ruido_Ia = np.random.normal(media_ruido, np.sqrt(
potencia_ruido), len(Ia))
```

```
            Ia = Ia + ruido_Ia

            Ib = np.array(rec.analog[4 + term])
            pot_Ib = np.mean(Ib ** 2)
            potencia_ruido = pot_Ib / (10 ** (ruido_dB / 10))
            ruido_Ib = np.random.normal(media_ruido, np.sqrt(
potencia_ruido), len(Ib))
            Ib = Ib + ruido_Ib

            Ic = np.array(rec.analog[5 + term])
            pot_Ic = np.mean(Ic ** 2)
            potencia_ruido = pot_Ic / (10 ** (ruido_dB / 10))
            ruido_Ic = np.random.normal(media_ruido, np.sqrt(
potencia_ruido), len(Ic))
            Ic = Ic + ruido_Ic

            Ns = len(Va)  # número de amostras
            Fs = 50000  # frequencia de amostragem
            Ts = 1 / Fs  # periodo de amostragem

            t_plot = list(np.arange(0, Ns / Fs, Ts))  # cria um vetor
de tempo para tomar como referencia e realizar os plots
            t_falta = tabela_tempos[lista_aux[lista_run.index(run)] -
1][lista_tipo.index(tipo) + 1]

            t_ciclo = 1 / 60  # tempo demandado por um ciclo em s
            n_ciclo = int(t_ciclo / Ts)  # número de índices
necessários para percorrer 1 ciclo

            # acha no vetor de tempo o índice relativo ao instante da
falta com menor erro possível
            dif1 = 1
            for t in t_plot:
                dif2 = abs(t_falta - t)
                if dif2 <= dif1:
                    n_falta = t_plot.index(t)
```

```
                    dif1 = dif2

            # obtendo 1 ciclo dos sinais após uma janela de 1ms após a
detecção da falta
            Va = Va[n_falta + 3*n_ciclo:n_falta + 4*n_ciclo]
            Vb = Vb[n_falta + 3*n_ciclo:n_falta + 4*n_ciclo]
            Vc = Vc[n_falta + 3*n_ciclo:n_falta + 4*n_ciclo]

            Ia = Ia[n_falta + 3*n_ciclo:n_falta + 4*n_ciclo]
            Ib = Ib[n_falta + 3*n_ciclo:n_falta + 4*n_ciclo]
            Ic = Ic[n_falta + 3*n_ciclo:n_falta + 4*n_ciclo]

            t_plot = t_plot[n_falta + 3*n_ciclo:n_falta + 4*n_ciclo]  #
 reajusta o vetor de tempo

            Ns = len(Va)  # atualizando o número de amostra para o
sinal obtido

            # Aplicando a FFT aos sinais e normalizando. (rfft retorna
somente a primeira metade do vetor, ja que ele é simetrico)
            coef_Va = rfft(Va)
            modulo_Va = np.abs(coef_Va)
            modulo_Va = modulo_Va / (Ns / 2)
            modulo_Va[0] = modulo_Va[0] / 2
            fase_Va = np.angle(coef_Va, deg=True)

            coef_Vb = rfft(Vb)
            modulo_Vb = np.abs(coef_Vb)
            modulo_Vb = modulo_Vb / (Ns / 2)
            modulo_Vb[0] = modulo_Vb[0] / 2
            fase_Vb = np.angle(coef_Vb, deg=True)

            coef_Vc = rfft(Vc)
            modulo_Vc = np.abs(coef_Vc)
            modulo_Vc = modulo_Vc / (Ns / 2)
            modulo_Vc[0] = modulo_Vc[0] / 2
```

```
fase_Vc = np.angle(coef_Vc, deg=True)

coef_Ia = rfft(Ia)
modulo_Ia = np.abs(coef_Ia)
modulo_Ia = modulo_Ia / (Ns / 2)
modulo_Ia[0] = modulo_Ia[0] / 2
fase_Ia = np.angle(coef_Ia, deg=True)

coef_Ib = rfft(Ib)
modulo_Ib = np.abs(coef_Ib)
modulo_Ib = modulo_Ib / (Ns / 2)
modulo_Ib[0] = modulo_Ib[0] / 2
fase_Ib = np.angle(coef_Ib, deg=True)

coef_Ic = rfft(Ic)
modulo_Ic = np.abs(coef_Ic)
modulo_Ic = modulo_Ic / (Ns / 2)
modulo_Ic[0] = modulo_Ic[0] / 2
fase_Ic = np.angle(coef_Ic, deg=True)

f = list(rfftfreq(len(Va), Ts))
n_fund = 1

modulo_Va = modulo_Va[n_fund]
modulo_Vb = modulo_Vb[n_fund]
modulo_Vc = modulo_Vc[n_fund]
fase_Vb = fase_Vb[n_fund]
fase_Va = fase_Va[n_fund]
fase_Vc = fase_Vc[n_fund]

modulo_Ia = modulo_Ia[n_fund]
modulo_Ib = modulo_Ib[n_fund]
modulo_Ic = modulo_Ic[n_fund]
fase_Ia = fase_Ia[n_fund]
fase_Ib = fase_Ib[n_fund]
fase_Ic = fase_Ic[n_fund]
```

```
            v_max = max(modulo_Va, modulo_Vb, modulo_Vc)
            i_max = max(modulo_Ia, modulo_Ib, modulo_Ic)

            dados = [modulo_Va, fase_Va, modulo_Vb, fase_Vb, modulo_Vc,
 fase_Vc, modulo_Ia, fase_Ia, modulo_Ib,
                     fase_Ib, modulo_Ic, fase_Ic, modulo_Va / v_max,
modulo_Vb / v_max, modulo_Vc / v_max,
                     modulo_Ia / i_max, modulo_Ib / i_max, modulo_Ic /
i_max, tabela_casos[lista_run.index(run)][2]]

            dados_csv.append(dados)

 dados_csv = pd.DataFrame(dados_csv)
 caminho_dados = 'C:/Users/faold/OneDrive/Área de Trabalho/TDF_Ruido_' +
 str(ruido_dB) + 'dB.csv'
 dados_csv.to_csv(caminho_dados, index = False, header = False)
```