

Universidade Federal de Juiz de Fora
Engenharia Elétrica - Habilitação em Sistemas Eletrônicos
Trabalho de conclusão de curso

Paulo Henrique Pereira Careli

Classificação de Distúrbios de Qualidade de Energia com Machine Learning

Juiz de Fora

2021

Paulo Henrique Pereira Careli

Classificação de Distúrbios de Qualidade de Energia com Machine Learning

Monografia apresentada a Coordenação do Curso de Engenharia Elétrica da Universidade Federal de Juiz de Fora, Habilitação em Sistemas Eletrônicos, como requisito para aprovação na disciplina CEL046 - Trabalho Final de Curso.

Orientador: Leandro Rodrigues Manso Silva

Juiz de Fora

2021

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Careli, Paulo.

Classificação de Distúrbios de Qualidade de Energia com Machine Learning / Paulo Henrique Pereira Careli. – 2021.
45 f. : il.

Orientador: Leandro Rodrigues Manso Silva

Conclusão de curso – Universidade Federal de Juiz de Fora, Engenharia Elétrica - Habilitação em Sistemas Eletrônicos. Trabalho de conclusão de curso , 2021.

1. Qualidade de Energia Elétrica. 2. AI. 3. Machine Learning. I. Manso, Leandro, orient. II. Título.

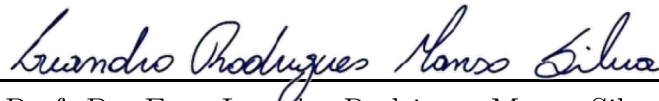
Paulo Henrique Pereira Careli

Classificação de Distúrbios de Qualidade de Energia com Machine Learning

Monografia apresentada a Coordenação do Curso de Engenharia Elétrica da Universidade Federal de Juiz de Fora, Habilitação em Sistemas Eletrônicos, como requisito para aprovação na disciplina CEL046 - Trabalho Final de Curso.

Aprovada em: 16 de Setembro de 2021

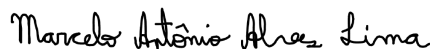
BANCA EXAMINADORA



Prof. Dr. Eng. Leandro Rodrigues Manso Silva
Orientador
Universidade Federal de Juiz de Fora



Prof. Dr. Eng. Eder Barboza Kapisch
Universidade Federal de Juiz de Fora



Prof. D. Sc. Marcelo Antônio Alves Lima
Universidade Federal de Juiz de Fora

AGRADECIMENTOS

Agradeço primeiramente aos meus pais Aloncio e Elizabeth que sempre estiveram presentes, me ajudaram e apoiaram, tornando possível assim, a realização deste sonho. Ao meu irmão Carlos, que sempre esteve do meu lado em tudo.

Aos meus amigos por todo o tempo de amizade, pelos conhecimentos, experiências e momentos compartilhados.

Ao meu orientador Leandro Manso pela oportunidade, ajuda, paciência e troca de conhecimento durante a elaboração deste trabalho.

Agradeço a todos aqueles que de alguma forma fizeram parte desta jornada, deixando sempre um novo aprendizado.

Agradeço à Universidade Federal de Juiz de Fora e aos meus professores pela incrível oportunidade de poder começar e terminar o curso de Engenharia Elétrica nesta querida instituição.

“O tempo é um tecido invisível em que se pode bordar tudo..”

- Machado de Assis

RESUMO

A energia elétrica é amplamente utilizada na atualidade e a cada dia que passa, torna-se mais importante conseguir efetuar a distribuição da mesma com qualidade. Este trabalho de Conclusão de Curso tem como foco gerar e comparar modelos de aprendizado de máquina que sejam capazes de classificar distúrbios de qualidade de energia. Serão simulados sinais contendo 14 tipos de distúrbios, a fim de montar um conjunto de dados e exportá-lo. Utilizando Python, será feito um processamento dos sinais deste conjunto, seguido da extração de 9 características, montando assim, um novo conjunto de dados que será utilizado para a classificação. Este novo conjunto será utilizado para o treinamento de 8 modelos de aprendizado de máquina diferentes. Logo após, será feita a comparação e avaliação dos mesmos.

Palavras-chave: Qualidade de Energia Elétrica. Distúrbios. Python. Inteligência Artificial. Aprendizado de Máquinas.

ABSTRACT

Electric energy is widely used nowadays, because of that, it's quality distribution becomes more and more important as time passes by. This final paper focuses on generating and comparing machine learning models that are able to classify power quality disturbances. Signals containing 14 types of disturbances are going to be simulated, and, a dataset with those signals will be exported. With Python, a signal processing technique will be applied to the signals, followed by the extraction of 9 features of each one, creating a new dataset that will be used for classification. This new dataset is going to be used to train 8 different machine learning models. After that, they will be compared and evaluated.

Keywords: Electric Power Quality. Disturbances. Python. Artificial Inteligence. Machine Learning.

LISTA DE ILUSTRAÇÕES

Figura 1 – Fluxograma do processo da classificação	17
Figura 2 – Sinal senoidal (esquerda) e sinal com distúrbio do tipo Swell (direita) .	19
Figura 3 – Sinal com distúrbio do tipo Sag (esquerda) e sinal com distúrbio do tipo Interrupção (direita)	19
Figura 4 – Sinal com distúrbio do tipo Flutuação de tensão (esquerda) e sinal com distúrbio do tipo Transitório oscilatório (direita)	20
Figura 5 – Sinal com distúrbio do tipo Notch (esquerda) e sinal com distúrbio do tipo Spike (direita)	20
Figura 6 – Sinal com distúrbio do tipo Harmônico (esquerda) e sinal com distúrbio do tipo Flicker com harmônico (direita)	21
Figura 7 – Sinal com distúrbio do tipo Sag com harmônico (esquerda) e sinal com distúrbio do tipo Swell com harmônico (direita)	21
Figura 8 – Sinal com distúrbio do tipo Interrupção com harmônico (esquerda) e sinal com distúrbio do tipo Transitório oscilatório com harmônico (direita)	22
Figura 9 – Exemplo - Transformada de Stockwell em um sinal	23
Figura 10 – Exemplo - Matriz de confusão	28
Figura 11 – Decision Tree - Matriz de Confusão	31
Figura 12 – Decision Tree - Relatório em tabela	32
Figura 13 – Resumo - Comparação entre os modelos	33
Figura 14 – Resultados - XGBoost	34
Figura 15 – Resultados - SVM	35
Figura 16 – Resultados - KNN	36
Figura 17 – Resultados - Naive Bayes	37
Figura 18 – Resultados - Decision Tree	38
Figura 19 – Resultados - Bagged Decision Tree	39
Figura 20 – Resultados - Random Forest	40
Figura 21 – Resultados - Gradient Boosting Tree	41

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
ML	Machine Learning
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

SUMÁRIO

1	INTRODUÇÃO	10
1.1	MOTIVAÇÃO	10
1.2	OBJETIVO	11
1.3	ESTRUTURA DO TRABALHO	11
2	REVISÃO DAS TÉCNICAS UTILIZADAS NO TRABALHO	12
2.1	ALGORITMOS DE APRENDIZADO DE MÁQUINA	12
2.2	CLASSIFICADORES	13
2.3	CONJUNTO DE DADOS	14
2.4	CARACTERÍSTICAS	14
2.4.1	TRANSFORMADA DE STOCKWELL	15
2.5	TREINAMENTO E RESULTADOS	15
2.6	CONCLUSÕES PARCIAIS	16
3	ALGORITMOS E IMPLEMENTAÇÃO	17
3.1	GERAÇÃO DE SINAIS	18
3.1.1	ALGORITMO PARA GERAR O CONJUNTO DE DADOS DE SINAIS	22
3.1.2	EXTRAÇÃO DE CARACTERÍSTICAS	22
3.2	ALGORITMO PARA EXTRAIR AS CARACTERÍSTICAS	25
3.3	ALGORITMOS DE CLASSIFICAÇÃO	26
3.4	CONCLUSÕES PARCIAIS	27
4	RESULTADOS	28
4.1	MÉTRICAS	28
4.2	RESULTADO DOS MODELOS	30
4.3	CONCLUSÕES PARCIAIS	42
5	CONCLUSÃO	43
5.1	TRABALHOS FUTUROS	43
	REFERÊNCIAS	44

1 INTRODUÇÃO

Com o avanço da civilização, o consumo e a necessidade da energia elétrica vem aumentando cada vez mais. Sua utilização está em todo lugar, desde uma simples lâmpada em casa até a alimentação de grandes indústrias. Para que seja possível atender à demanda de diversos setores de forma eficiente, é necessário manter a qualidade da energia, de forma a não ocasionar uma queda de desempenho significativa na carga.

Entretanto, devido à natureza de distribuição e consumo de energia elétrica, pode-se haver variações durante a sua distribuição conforme as cargas instaladas, os transformadores ou até mesmo na produção da mesma. Com isso, podemos ter distúrbios na rede elétrica, acarretando um mal funcionamento das cargas instaladas além de diversos outros problemas.

Os distúrbios na qualidade energia podem ocorrer por diversos problemas, e, muitas vezes, não é simples identificá-los. Entretanto, podemos utilizar algumas técnicas para facilitar esta tarefa. Uma solução para facilitar a identificação destes distúrbios, é utilizar a tecnologia, algo que está sempre em constante ascensão. Hoje, ela está presente no nosso dia a dia em vários locais e de diversas formas, onde, uma delas, a inteligência artificial, vem se expandindo e ganhando cada vez mais espaço na sociedade, devido ao avanço de outros itens, que possibilitam a sua evolução.

A inteligência artificial [1] teve origem a mais de 50 anos e desde então está em processo de constante aprimoramento, chegando hoje, ao ponto de conseguir realizar inúmeras tarefas, tendo, assim, diversos sub-campos e áreas de aplicações. Um destes sub-campos, é o aprendizado de máquinas (do inglês, *Machine Learning*) que, como o próprio nome já diz, utiliza de máquinas para resolver diversos problemas, fazendo com que ela aprenda sobre os dados passados para a mesma.

1.1 MOTIVAÇÃO

A tecnologia vem avançando cada dia mais, e com isso, a necessidade de energia elétrica também cresce e se torna cada vez mais essencial para o mundo. Sendo assim, é necessário fornecê-la com qualidade.

Para conseguir realizar esta tarefa, é necessário fazer testes e monitorar a sua situação a fim de diminuir o máximo possível os distúrbios presentes na forma de onda do sinal de tensão. Uma maneira de ajudar, é utilizar a tecnologia para facilitar a identificação dos mesmos.

Com este crescimento, temos máquinas cada vez mais potentes, capazes de realizar cálculos muito complexos, e, com isso, técnicas baseadas em aprendizagem de máquina para realizar a classificação dos distúrbios de qualidade de energia so tornam cada vez

mais atrativas.

1.2 OBJETIVO

O objetivo deste trabalho é aplicar técnicas de aprendizagem de máquina para analisar as amostras de sinais elétricos, extrair e selecionar características, além de gerar e comparar modelos capazes de classificar alguns dos distúrbios da qualidade de energia dado um número de amostras de sinais. Para isso, será gerado uma base de dados contendo os sinais com os distúrbios de forma aleatória, após, será aplicado um processamento dos sinais, através da Transformada de Stockwell seguido da extração de características (em inglês, *features*) e, finalmente, a classificação dos distúrbios. Para gerar e exportar os sinais, foi utilizado o software MATLAB, as demais tarefas, foram realizadas em Python.

1.3 ESTRUTURA DO TRABALHO

O atual trabalho, além do capítulo de introdução, foi subdividido em quatro capítulos, como descrito a seguir:

No capítulo 2 será feita uma descrição dos classificadores e da estratégia utilizada para treinar, gerar e validar os modelos.

No capítulo 3 será explicitado como foi gerado a base de dados utilizada para o treinamento, mostrando as técnicas utilizadas. Também será feita uma descrição dos algoritmos utilizados neste trabalho.

No capítulo 4 será apresentado os resultados finais obtidos no trabalho desenvolvido.

Finalizando, no capítulo 5, será apresentado a conclusão do trabalho e suas propostas para melhorias.

2 REVISÃO DAS TÉCNICAS UTILIZADAS NO TRABALHO

Como visto na introdução, este trabalho busca gerar modelos capazes de identificar distúrbios característicos de qualidade de energia em sinais elétricos. Para isso, os modelos devem ser capazes de realizar a classificação dos sinais.

A classificação é o ato, neste caso, de atribuir uma classe para um conjunto de valores de entrada. Essa classificação pode ser binária (0 ou 1) ou multiclasse (mais que 2 classes). Por exemplo, classificar uma bola como cheia ou vazia (binário), ou como azul, verde ou amarelo (multiclasse). Este trabalho utilizou da classificação multiclasse.

O objetivo deste capítulo é abordar sobre os classificadores e estratégias utilizadas para obter os resultados apresentados no capítulo 4, fazendo uma abordagem geral do que foi feito.

2.1 ALGORITMOS DE APRENDIZADO DE MÁQUINA

Os algoritmos de aprendizagem de máquina [2] estão divididos entre supervisionados, não supervisionados e aprendizado por reforço. Onde, os supervisionados são divididos em algoritmos de classificação e regressão, os não supervisionados estão divididos em algoritmos de agrupamento de dados, classificação e redução de dimensionalidade e os de aprendizado por reforço são algoritmos focados em classificação e para reação ao que está acontecendo.

Os algoritmos supervisionados buscam prever uma variável dado um conjunto de características (em inglês, *features*). Elas são utilizadas como um rótulo (em inglês, *label*) de determinado dado na hora do treinamento. Eles são muito utilizados para prever um dado, utilizando um modelo treinamento com um conjunto de dados (em inglês, *dataset*) rotulado.

Os algoritmos não supervisionados, ao contrário do anterior, não tem um alvo para prever, sendo assim utilizados para separar os dados e criar grupos menores onde cada qual compartilha características, ele é muito utilizado para agrupar clientes em diferentes grupos para um estudo específico.

Os algoritmos de aprendizado por reforço, são algoritmos treinados através da tentativa e erro. São expostos em algum ambiente com determinadas regras e objetivos, e a cada nova tentativa, ele tenta chegar mais perto do resultado proposto. São muito utilizados para criar uma inteligência artificial capaz de jogar um jogo, por exemplo.

Os algoritmos de classificação são focados em prever um dado discreto, dado um número de categorias e/ou classes. (Ex: Verdadeiro ou falso)

Os algoritmos de regressão são focados em prever um dado contínuo de valor. (Ex:

Preço de um item).

Os algoritmos de agrupamento de dados são focados em descobrir padrões nos dados e classificar-los em determinados grupos.

Rótulo (em inglês, *Label*): É o rótulo dado a um conjunto de dados, visando o treinamento supervisionado.

Base de dados (em inglês, *Dataset*): É a base de dados utilizada para extrair características, treinar ou testar modelos de aprendizado de máquinas.

Característica (em inglês, *Feature*): É uma característica que pode ser utilizada para análise. Por exemplo: Nome, Idade, Cor, Altura.

A seguir serão apresentados os classificadores utilizados para gerar os modelos.

2.2 CLASSIFICADORES

KNN (K Nearest Neighbor): É um algoritmo de aprendizado de máquina supervisionado que determina a distância do vizinho mais próximo treinando amostras de acordo com a variável K para conseguir classificar um novo item.

Para encontrar a distância de um ponto à outro, podemos utilizar medidas como por exemplo: Distância Euclidiana, distância de Hamming, distância de Manhattan e distância de Markowski. Este algoritmo consegue resolver tanto problemas de classificação quanto de regressão.

Naive Bayes: É um algoritmo classificador probabilístico baseado no “Teorema de Bayes”, criado por Thomas Bayes assumindo independência entre os previsores. Este algoritmo assume que a presença de uma determinada característica em uma classe não está relacionada com a presença de nenhuma outra característica.

SVM (Support Vector Machine): É um algoritmo supervisionado que pode ser utilizado para classificação ou regressão. O objetivo deste algoritmo é achar um hiperplano em um espaço com N-dimensões (onde N é o número de características) que consegue classificar distintamente os dados.

Decision tree: O decision tree é um algoritmo supervisionado que pode ser utilizado para classificação ou regressão. Como o nome já diz, é um algoritmo que pode ser visualizado como um modelo de decisões baseado em uma árvore.

Bagged Decision Tree: É um algoritmo baseado no Decision Tree, onde a ideia é diminuir a variância do mesmo. Neste algoritmo, são criados vários subconjuntos de dados das amostras de treinamentos escolhidas aleatoriamente, criando assim, vários modelos. A média da previsão de todos esses modelos é mais robusta que apenas uma árvore de decisão.

Gradient Boosting Tree: É um algoritmo que utiliza a junção de duas técnicas, o Boosting e o Gradient Descent.

O Boosting é uma técnica de conjuntos (assim como o Decision Tree) que visa criar uma coleção de previsores. Ele treina várias árvores consecutivas e a cada uma, busca-se, resolvê-la para o erro da árvore anterior.

O Gradient Descent é um algoritmo que pode otimizar qualquer função de perda diferenciável. Onde um conjunto de árvores é criado um por um e somados sequencialmente e a próxima árvores tenta recuperar a perda, através da diferença entre o valor atual e o previsto.

Random Forest: É um algoritmo baseado no algoritmo Decision Tree e é uma extensão do Bagged Decision Tree. Este, além de criar vários subconjuntos de dados escolhidos aleatoriamente das amostras de treinamento, também escolhe aleatoriamente as características em vez de utilizar todas elas para criar novas árvores, criando assim, várias árvores aleatórias.

XGBoost (eXtreme Gradient Boosting): É um algoritmo baseado no Gradient Boosting Tree com o foco na velocidade de execução e na performance do modelo.

2.3 CONJUNTO DE DADOS

Na seção anterior, vimos os classificadores que foram utilizados para a resolução do problema. Para conseguir realizar a classificação dos distúrbios, é necessário prover conjunto de dados contendo itens que se relacionam com os distúrbios. Para isso, foi simulado e gerado um conjunto de dados de forma aleatória, contendo os sinais com os distúrbios.

Esse conjunto de dados é utilizado para extrair características e treinar os classificadores, visando gerar modelos capazes de classificar corretamente sinais com uma alta taxa de acerto.

2.4 CARACTERÍSTICAS

Para que os classificadores obtenham um bom resultado, é necessário tratar o conjunto de dados utilizado, de forma a facilitar o entendimento do mesmo, procurando

sempre, por pontos chaves. Para isso, foi criado um novo conjunto de dados, contendo características extraídas do conjunto original. Esse novo conjunto de dados busca prover as principais características presentes nos distúrbios, facilitando assim, o aprendizado dos classificadores e favorecendo seu bom desempenho.

As características, conforme visto anteriormente, são dados que pode ser utilizados para análise. Os modelos de aprendizagem de máquina utilizam-nas para o aprendizado, na etapa de treinamento, e posteriormente para a realização da classificação. Neste trabalho, para extraí-las do conjunto de dados contendo os sinais, foi utilizado a transformada de Stockwell cuja a função é realizar o processamento dos sinais, fazendo uma análise de tempo-frequência, tornando assim, possível o cálculo de características conhecidas, como por exemplo, a média, o valor máximo e mínimo.

2.4.1 TRANSFORMADA DE STOCKWELL

A transformada de Stockwell [3] [4] (conhecida também por transformada S) é uma representação de tempo-frequência, conhecida pelas suas propriedades de fase espectral local.

A transformada de Stockwell foi escolhida para ser aplicada no nosso conjunto de dados de sinais pela sua resolução de tempo-frequência e alta imunidade à ruídos.

Após a utilização da transformada de Stockwell nos dados gerados, obtemos uma matriz complexa contendo os dados de tempo-frequência dos sinais. A partir desta matriz, foi possível realizar os cálculos e extrair as características utilizadas para o treinamento dos nossos modelos.

2.5 TREINAMENTO E RESULTADOS

O treinamento é necessário para ensinar os nossos modelos como prever os dados corretamente utilizando um conjunto de dados. É importante ressaltar que, quanto melhor for este conjunto de dados, melhor será o resultado do nosso modelo. Para isso, é interessante tratar os dados, de forma a facilitar o treinamento e a gerar um resultado melhor, visto que a qualidade dos dados é tão importante quanto os algoritmos.

Existem diversas técnicas para tratar estes dados, e com isso, elas são escolhidas após a análise do conjunto de dados utilizado. É importante observar também, o quão interessante pode ser ter um conjunto de dados balanceado, caso queira classificar todos os tipos com uma eficiência parecida, por exemplo, caso o conjunto possua 95% dos dados de um tipo A e 5% de um tipo B, ele pode ter problemas para identificar os do tipo B devido à falta de informação.

Durante o treinamento, a quantidade de dados também influência no resultado, onde, modelos mais complexos tendem a necessitar de mais dados para conseguir um

melhor resultado quando comparados os mais simples. Conjuntos com muitos rótulos também necessitam de mais dados.

O treinamento possui parâmetros, que visam controlar certas propriedades deste processo e do modelo resultante. Estes parâmetros variam de acordo com o algoritmo. Entretanto, alguns dos mais comuns são, por exemplo, quando o algoritmo deve parar o treinamento (critério de parada), quantas vezes o algoritmo deve iterar sobre aqueles dados, se o algoritmo deve utilizar os dados de forma aleatória, entre outros. Os algoritmos utilizam um valor padrão para estes parâmetros caso não alterados.

Com o modelo gerado, é necessário fazer testes, utilizar o bom senso humano é uma das melhores maneiras para ver se o

Após treinar e gerar os modelos, é necessário fazer testes e avaliar os mesmos. Os modelos gerados podem não ter bons resultados, ou até mesmo estarem viciados devido à diversos fatores. Para verificar a qualidade dos modelos gerados e se eles atendem ao objetivo necessário, são utilizadas métricas em conjunto com o olhar crítico do ser humano.

2.6 CONCLUSÕES PARCIAIS

Este capítulo descreveu os classificadores utilizados para a classificação dos distúrbios e apresentou uma visão geral das técnicas utilizadas para extrair características dos sinais. Assim, tornando-se possível gerar o conjunto de dados utilizado para o treinamento e predição dos modelos, obtendo desta forma, o resultado utilizado para classificar os distúrbios.

3 ALGORITMOS E IMPLEMENTAÇÃO

Este capítulo contempla a implementação da solução proposta, mostrando como foi gerado os conjuntos de dados, os códigos utilizados e os modelos. Todos os códigos, modelos e conjunto de dados gerados e utilizados estão disponíveis no repositório “power-quality-disturbances-with-machine-learning”[5] hospedado no Github.

Consta abaixo um fluxograma mostrando como foi decidida e implementada esta solução:

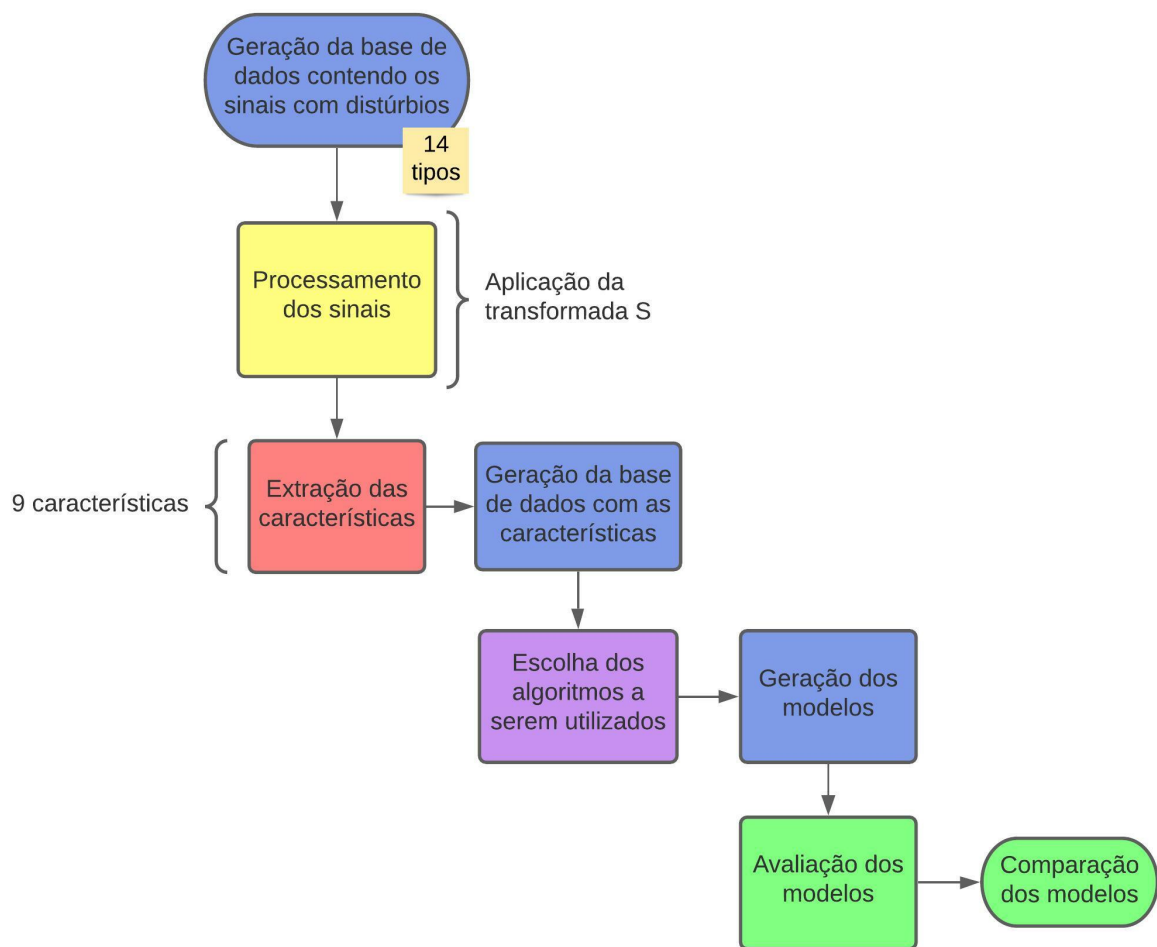


Figura 1 – Fluxograma do processo da classificação

Fonte: Autor

Os próximos tópicos deste capítulo irão abordar os tópicos conforme o fluxograma da Figura 1. Entretanto, os tópicos “Avaliação dos modelos” e “Comparação dos modelos” serão contemplados no capítulo 4.

3.1 GERAÇÃO DE SINAIS

Para a realização deste estudo, são necessárias amostras de sinais contendo distúrbios. Assim, foi utilizado o software MATLAB para simular e gerar estes sinais com seus respectivos distúrbios, visto a dificuldade de obtenção de um banco de dados de uma rede elétrica real. Através deste software, foram utilizados códigos baseados nos providos pelo professor Leandro Manso, desenvolvidos pelos autores: Cristiano A. G. Marques, Lucas R. Freitas e Lucas Oliveira. Adequando-os conforme necessário, para gerar os devidos sinais.

Os sinais com os distúrbios escolhidos foram:

Sinais Gerados:

- Normal/Sem Distúrbio.
- Swell
- Sag
- Interrupção (em inglês, *Interruption*)
- Flutuação de Tensão (em inglês, *Flicker*)
- Transitório Oscilatório (em inglês, *Oscillatory transient*)
- Notch
- Spike
- Harmônicos (em inglês, *Harmonics*)
- Flicker com harmônicos (em inglês, *Flicker and harmonics*)
- Sag com harmônicos (em inglês, *Sag and harmonics*)
- Swell com harmônicos (em inglês, *Swell and harmonics*)
- Interrupção com harmônicos (em inglês, *Interruption and harmonics*)
- Transitório oscilatório com harmônicos (em inglês, *Oscillatory transient and harmonics*)

Para um melhor entendimento e visualização dos devidos sinais, foram gerados os gráficos abaixo (Figuras 2 a 8) exemplificando os mesmos citados acima utilizando o código “plot_signals.m”.

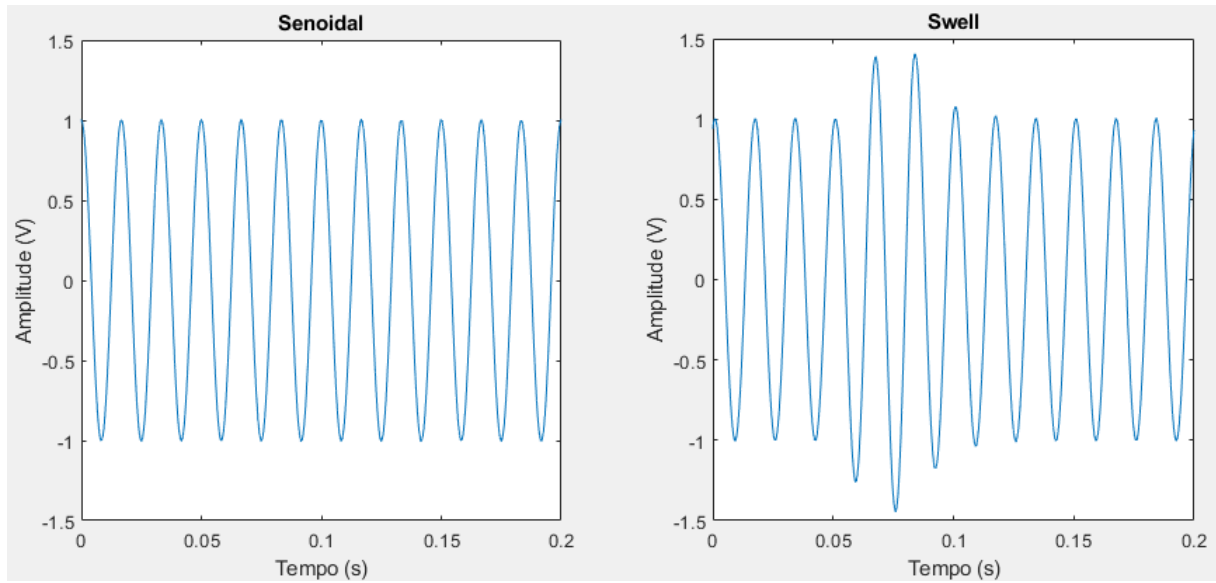


Figura 2 – Sinal senoidal (esquerda) e sinal com distúrbio do tipo Swell (direita)

Fonte: Autor

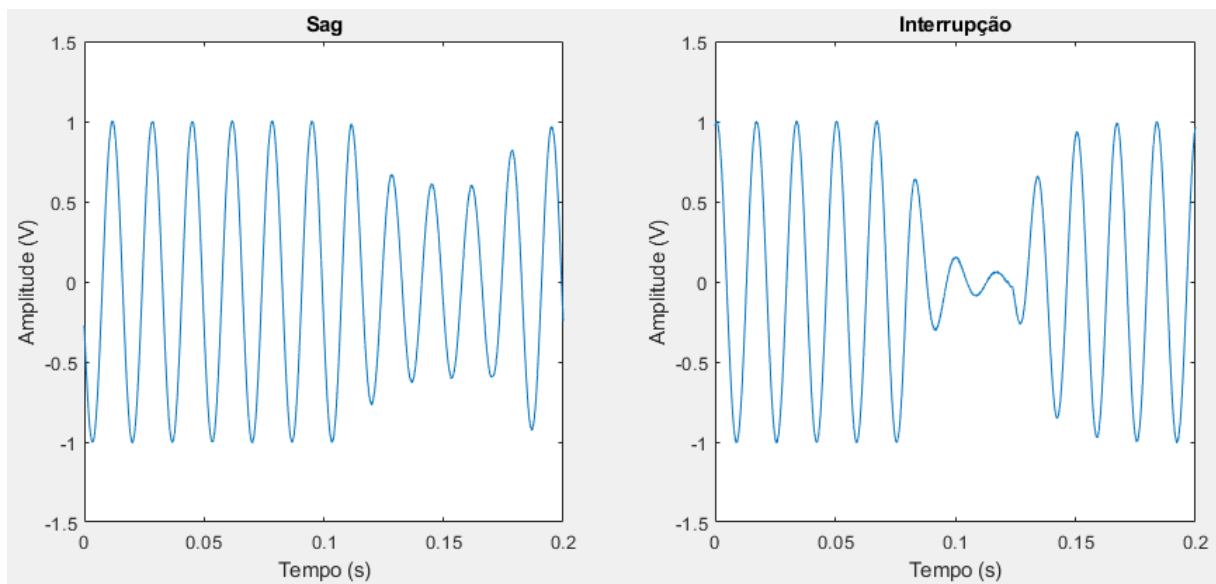


Figura 3 – Sinal com distúrbio do tipo Sag (esquerda) e sinal com distúrbio do tipo Interrupção (direita)

Fonte: Autor

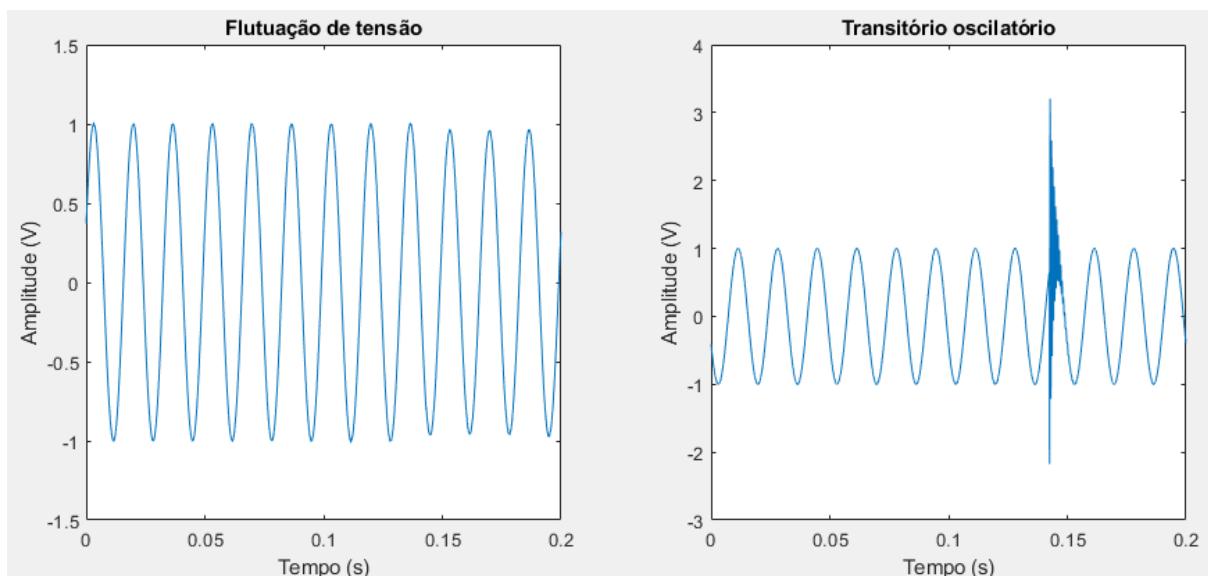


Figura 4 – Sinal com distúrbio do tipo Flutuação de tensão (esquerda) e sinal com distúrbio do tipo Transitório oscilatório (direita)

Fonte: Autor

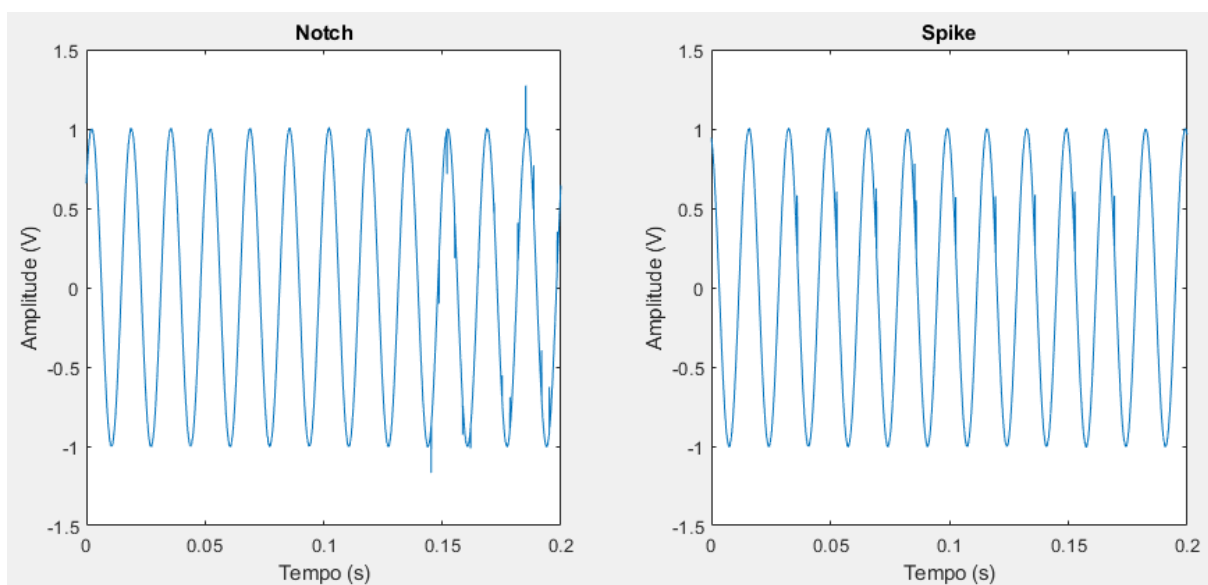


Figura 5 – Sinal com distúrbio do tipo Notch (esquerda) e sinal com distúrbio do tipo Spike (direita)

Fonte: Autor

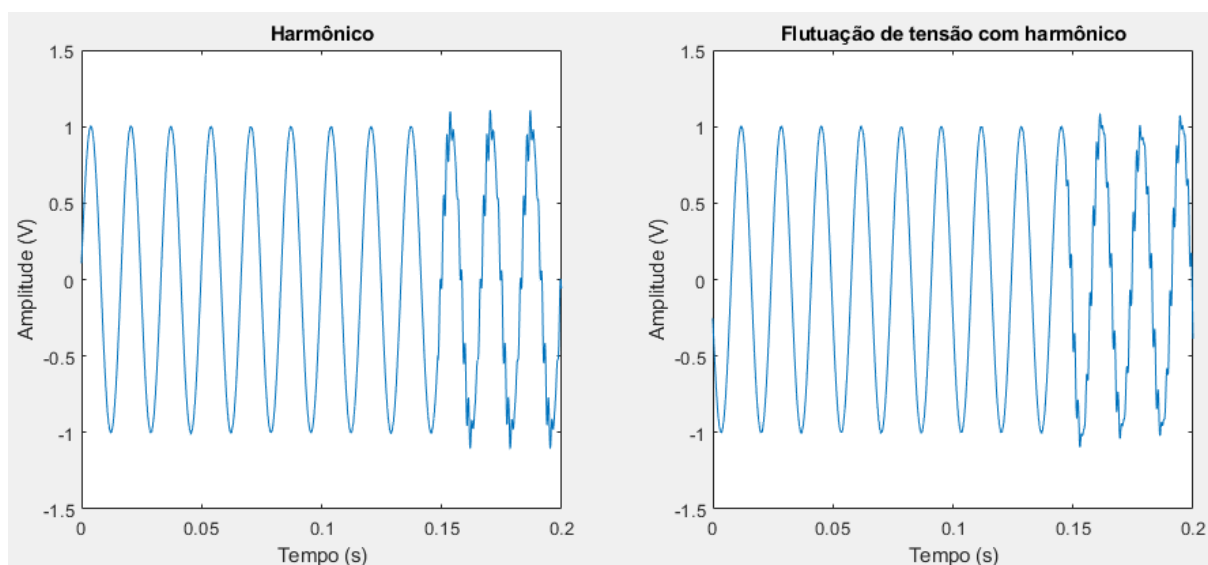


Figura 6 – Sinal com distúrbio do tipo Harmônico (esquerda) e sinal com distúrbio do tipo Flicker com harmônico (direita)

Fonte: Autor

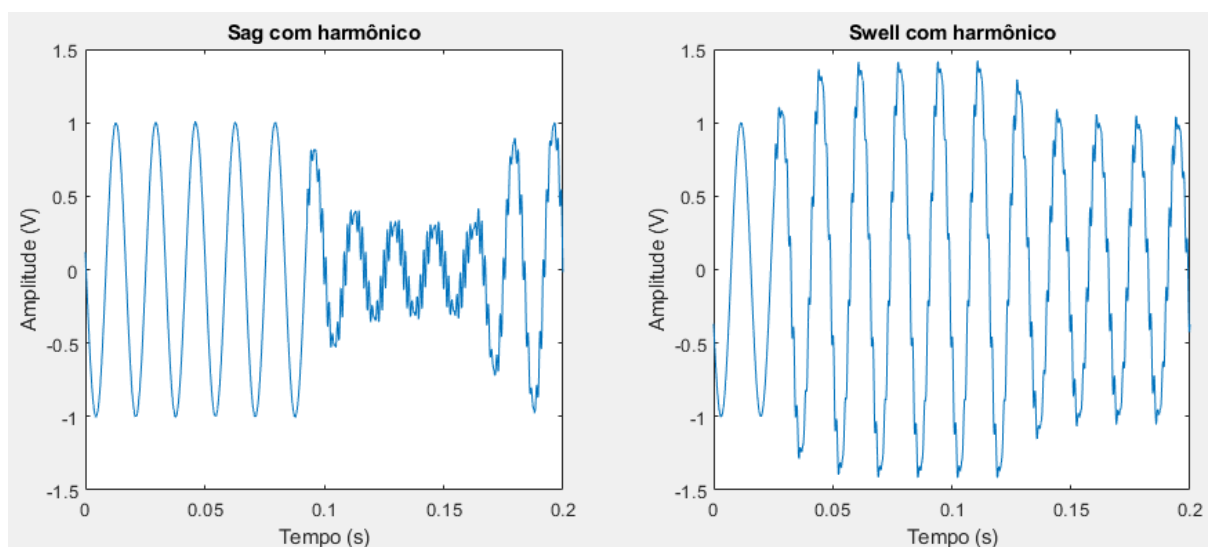


Figura 7 – Sinal com distúrbio do tipo Sag com harmônico (esquerda) e sinal com distúrbio do tipo Swell com harmônico (direita)

Fonte: Autor

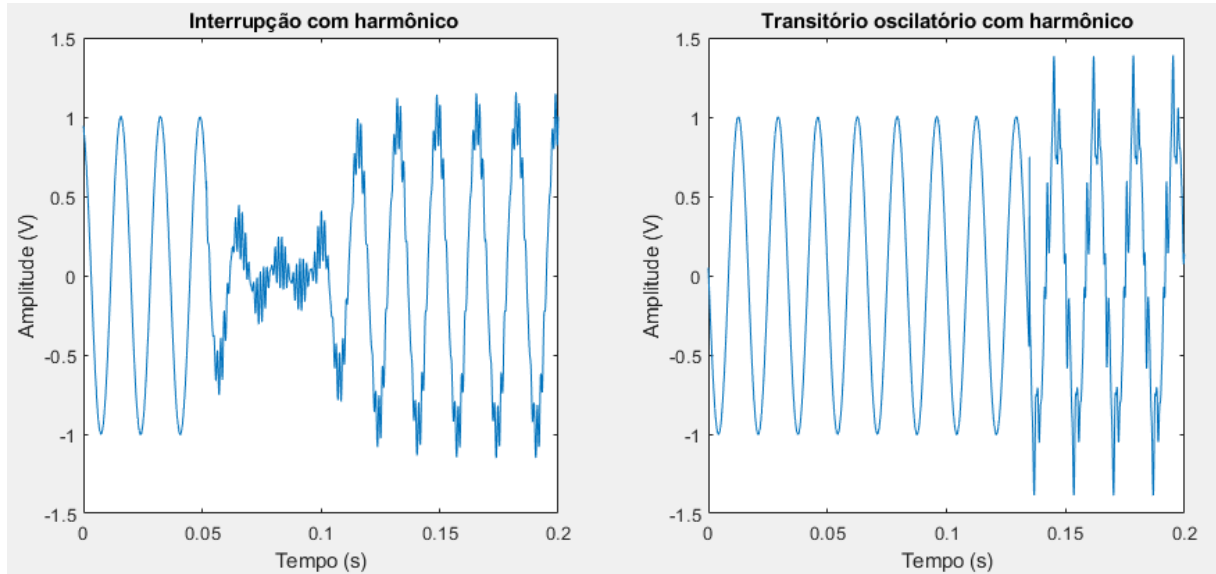


Figura 8 – Sinal com distúrbio do tipo Interrupção com harmônico (esquerda) e sinal com distúrbio do tipo Transitório oscilatório com harmônico (direita)

Fonte: Autor

A partir destes sinais, foi construído um conjunto de dados contendo 20.000 sinais, escolhidos aleatoriamente dentre os 14 tipos citados previamente com a frequência de amostragem de 15360, contendo 12 ciclos do componente fundamental de frequência igual a 60Hz e com relação sinal-ruído (em inglês, *signal to noise ratio (SNR)*) de valor 40.

Através do código “create_dataset.m”, foi gerado o conjunto de dados contendo os sinais aleatórios simulados. Cada sinal ocupa uma linha em formato de vetor do conjunto de dados exportado, onde a última posição de cada linha contém, o rótulo daquele sinal.

3.1.1 ALGORITMO PARA GERAR O CONJUNTO DE DADOS DE SINAIS

Como dito no capítulo anterior, foram utilizados códigos baseados nos providos pelo professor Manso, desenvolvidos pelos autores Cristiano A. G. Marques, Lucas R. Freitas e Lucas Oliveira. Estes códigos foram feitos no MATLAB.

Foram realizadas adaptações nos códigos providos e criado um novo código (create_dataset.m) que faz uso dos mesmos para gerar e exportar um conjunto de dados no formato csv, que será processado posteriormente em Python. Este código é composto de um loop, que cria sinais de forma aleatória, salva os mesmos em uma matriz e depois a exporta.

3.1.2 EXTRAÇÃO DE CARACTERÍSTICAS

Para aplicar as técnicas de Machine Learning, primeiramente, é necessário preparar o conjunto de dados de forma a melhorar e facilitar para o algoritmo, visando um melhor

resultado. Para isso, foi aplicado a transformada de Stockwell em cada sinal, resultando em uma matriz complexa. Como vimos no capítulo anterior, a transformada de Stockwell foi escolhida para ser aplicada no nosso conjunto de dados de sinais pela sua resolução de tempo-frequência e alta imunidade à ruídos. Estas técnicas foram baseadas no artigo referenciado [6].

Podemos observar abaixo um sinal antes e depois de aplicar a transformada de Stockwell:

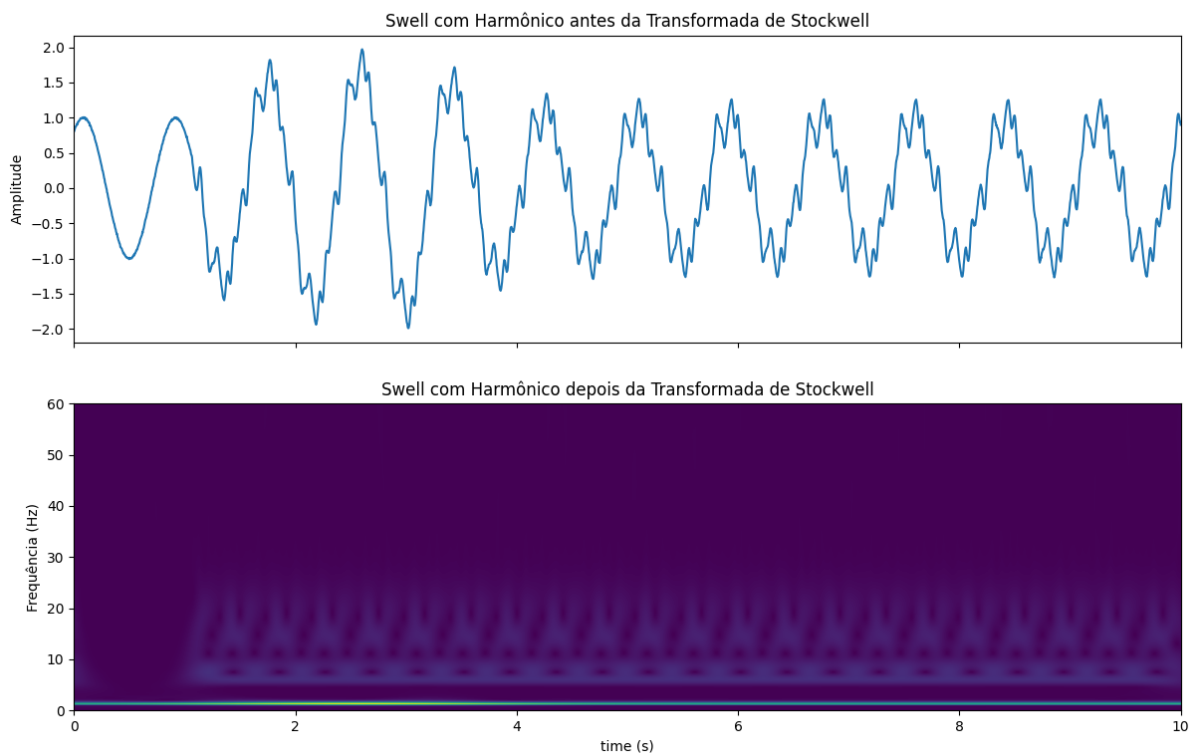


Figura 9 – Exemplo - Transformada de Stockwell em um sinal

Fonte: Autor

A partir da matriz obtida pela transformada de Stockwell, foram extraídas 9 características de cada sinal, e assim, criado um novo conjunto de dados. Este novo conjunto foi utilizado para realizar o treinamento dos classificadores, ele contém as características de cada sinal e seu rótulo.

Com essa extração, foi possível diminuir o conjunto de dados, pegando características importantes de cada sinal para serem utilizadas no treinamento. Foram escolhidos alguns parâmetros estatísticos bem populares para algumas das características como Máximo, mínimo, Valor médio, Desvio Padrão, Variância, Skewness e Kurtosis e outros mais específicos como Valor RMS e DER (Disturbance Energy Ratio). Todos os valores

foram calculados como sendo valores absolutos e calculados a partir da matriz obtida pela transformada S. As linhas da matriz foram definidas por “M” e as colunas por “N”.

Os cálculos para a definição das características são definidos como:

F1 - Máximo:

$$M = \max\{|A_{jn}|\} \quad (3.1)$$

F2 - Mínimo:

$$m = \min\{|A_{jn}|\} \quad (3.2)$$

F3 - Valor médio:

$$\mu = \frac{\sum_{j=1}^M \sum_{n=1}^N |A_{jn}|}{M \cdot N} \quad (3.3)$$

F4 - Valor RMS:

$$RMS = \sqrt{\frac{\sum_{j=1}^M \sum_{n=1}^N |A_{jn}|^2}{M \cdot N}} \quad (3.4)$$

F5 - DER (Disturbance Energy Ratio):

$$DER = \frac{\sum_{freq=15360Hz}^{freq=60Hz} RMS_j}{\sum_{freq=0Hz}^{freq=59Hz} RMS_j} \quad (3.5)$$

F6 - Desvio Padrão:

$$\sigma = \sqrt{\frac{\sum_{j=1}^M \sum_{n=1}^N (|A_{jn}| - \mu_j)^2}{(M-1)(N-1)}} \quad (3.6)$$

F7 - Variância:

$$\sigma^2 = \frac{\sum_{j=1}^M \sum_{n=1}^N (|A_{jn}| - \mu_j)^2}{(M-1)(N-1)} \quad (3.7)$$

F8 - Skewness:

$$SKEW_{\phi} = \frac{\sum_{j=1}^M \sum_{n=1}^N (|\phi_{jn}| - \mu_{(\phi)j})^3}{M \cdot N \cdot \sigma_{\phi}^3} \quad (3.8)$$

F9 - Kurtosis:

$$KURT = \frac{\sum_{j=1}^M \sum_{n=1}^N (|A_{jn}| - \mu_j)^4}{M \cdot N \cdot \sigma^4} \quad (3.9)$$

3.2 ALGORITMO PARA EXTRAIR AS CARACTERÍSTICAS

Após exportado o conjunto de dados iniciais pelo MATLAB, foi utilizado Python para as demais atividades. Com ele foi feito o processamento dos sinais utilizando a transformada de Stockwell, a extração de características, o treinamento, predições, validações e comparações entre os modelos de aprendizado de máquina.

No código para extração das características em Python “extract_features.py”, foi feito o processamento dos sinais utilizando a transformada de Stockwell e a extração de características. Foi utilizado a biblioteca “pandas” [7] para a manipulação dos conjuntos de dados neste trabalho, quando em Python. Utilizando essa biblioteca, os conjuntos de dados foram manipulados através de dataframes.

DataFrame: É uma estrutura de dados de duas dimensões com os dados em forma tabular (linhas e colunas), semelhante a um arquivo excel.

Começando a execução, o código “extract_features.py” faz a leitura do arquivo csv gerado previamente no MATLAB. Começa separando a última coluna que foi utilizada com o rótulo para o treinamento e configurando os parâmetros dos sinais utilizados, como por exemplo, as frequências mínimas e máximas. Após, foi construído um loop que itera sobre cada sinal contido no conjunto de dados, esse loop aplica a transformada S no sinal utilizando a biblioteca “stockwell” [8], calcula as características e as salva em uma matriz. Este loop pode demorar um pouco dependendo do número dos sinais e de amostras contidos no conjunto de dados utilizado, devido aos cálculos realizados. Para diminuir o tempo levado para realizar os cálculos, foi utilizado a biblioteca “Numba” [9]. Esta biblioteca transforma uma parte do código em um código de máquina rápido usando uma

infraestrutura de compilador feita em C++, possibilitando ainda, uma fácil paralelização do mesmo.

Para o cálculo das características, foram utilizadas funções embutidas da biblioteca “NumPy” [10]. Para algumas das características, foram criadas funções auxiliares contidas no código “parameters_calculator.py”. Como dito previamente, dependendo do número de sinais e amostras contidos no conjunto de dados, os cálculos para extração das características podem demorar um pouco, principalmente quando é necessário percorrer a matriz gerada pela transformada de Stockwell, como por exemplo, no cálculo de Skewness e Kurtosis. Visando melhorar a velocidade para um conjunto de dados maior, foi utilizada, quando possível, a biblioteca “numba”, que ajuda a reduzir o tempo de execução de funções quando executadas mais de uma vez. Com isso, foi possível diminuir o tempo de extração das características.

Após realizados os cálculos, o programa cria um conjunto de dados com os campos relacionados às devidas características, incluindo novamente, a coluna contendo os rótulos. Em seguida, é realizada a exportação do mesmo no formato .csv.

3.3 ALGORITMOS DE CLASSIFICAÇÃO

Com o novo conjunto de dados gerado, contendo as características, foi possível aplicar os algoritmos de aprendizagem de máquina para criar os respectivos modelos, capazes de fazer a classificação destes sinais.

Os algoritmos utilizados conforme dito previamente foram:

- KNN.
- Naive Bayes.
- SVM.
- Decision Tree.
- Bagged Decision Tree.
- Gradient Boosting Tree.
- Random Forest.
- XGBoost.

Foi criado um código para cada modelo escolhido. Cada código importa as bibliotecas necessárias para o modelo contemplado, carrega o conjunto de dados contendo as características e os devidos rótulos conforme vimos na seção anterior, divide os dados em treinamento e teste, escolhe o classificador, treina o mesmo com os dados importados e salva o modelo no formato “.pkl” para ser utilizado posteriormente. Os códigos também

contemplam a predição e os resultados obtidos com determinado modelo, entretanto, estes tópicos serão abordados no capítulo 4.

É interessante notar que, para um melhor resultado de alguns destes modelos, foram especificados alguns hyperparamêtros atrelados aos mesmos. Como por exemplo os hyperparâmetros “C” e “sigma” do SVM e o “K” no KNN. Os valores para os mesmos foram obtidos através de testes, visto que não é possível saber o melhor valor.

Hyperparâmetro (em inglês, *Hyperparameter*): São variáveis manipuláveis de configuração (parâmetros) ajustadas externamente ao modelo, que influenciam diretamente o processo de treinamento, podendo acarretar em um melhor aprendizado, e, conseqüentemente, em um resultado de predição superior. Os hyperparâmetros não podem ser estimados pelos dados disponíveis.

Abaixo seguem as bibliotecas utilizadas em Python durante todo o processo.

Bibliotecas utilizadas:

- **Scipy [11]:** Esta biblioteca contempla outras bibliotecas de grande importância para a computação matemática.
- **Numpy:** Biblioteca para a linguagem Python que contempla ferramentas muito úteis para a computação científica.
- **Pandas:** Biblioteca para a linguagem Python que contempla ferramentas muito úteis para a análise e manipulação de dados.
- **Matplotlib [12]:** Biblioteca que facilita a plotagem de gráficos, no caso, dos sinais.
- **Stockwell:** Biblioteca utilizada para realizar a transformada de Stockwell.
- **Numba:** Biblioteca utilizada para otimizar o código na máquina enquanto o mesmo está rodando.
- **Sklearn [13]:** Biblioca que contempla diversas ferramentas para machine learning em Python.

3.4 CONCLUSÕES PARCIAIS

Este capítulo descreveu como foi feito a simulação e geração dos sinais com distúrbios, a extração de características utilizando a transformada de Stockwell, o cálculo utilizado para cada uma das características. Também foi descrito como foi montado o conjunto de dados contendo as características extraídas, e finalmente, como foram criados e salvos os modelos de aprendizado de máquina.

4 RESULTADOS

Este capítulo tem o objetivo de analisar os resultados produzidos pelos modelos descritos no Capítulo 3 de forma a avaliá-los, verificar sua eficiência e re realizar uma comparação entre eles.

4.1 MÉTRICAS

Para mostrar o resultado, primeiro devemos entender como avaliar um modelo de aprendizado de máquinas. Nesta seção, serão abordadas as métricas que foram utilizadas para verificar a eficácia dos modelos.

Matriz de confusão (em inglês, *confusion matrix*): A matriz de confusão é uma métrica que mostra a frequência da ocorrência de TP, FP, TN ou FN em um formato de tabela, facilitando assim, verificar o que está ocorrendo no sistema, como por exemplo, quantos acertos está ocorrendo em cada classe. Cada linha desta tabela, representa instâncias de uma classe prevista e cada coluna simboliza instâncias da classe atual.

		Classificação Atual	
		T	N
Classificação Prevista	T	TP	FN
	N	FP	TN

Figura 10 – Exemplo - Matriz de confusão

Fonte: Autor

Onde temos que::

- **TP (True Positive):** O verdadeiro positivo acontece quando o modelo prevê corretamente a classe positiva.Exemplo: O copo está cheio e o modelo previu corretamente que ele está cheio.

- **TN (True Negative)**: O verdadeiro negativo acontece quando o modelo prevê corretamente a classe negativa. Exemplo: O copo está vazio e o modelo previu corretamente que ele está vazio.
- **FP (False Positive)**: O falso positivo acontece quando o modelo prevê incorretamente a classe positiva. Exemplo: O copo está vazio e o modelo previu incorretamente que ele está cheio.
- **FN (False Negative)**: O falso negativo acontece quando o modelo prevê incorretamente a classe negativa. Exemplo: O copo está cheio e o modelo previu incorretamente que ele está vazio.

Com a matriz, é fácil visualizar o resultado e buscar os erros de previsão, pois eles estarão fora da diagonal principal, que contém os valores VP e VN de cada classe.

Acurácia (em inglês, *Accuracy*): É a porcentagem das predições corretas para um teste. Ela nos retorna quantos dos exemplos foram realmente classificados corretamente independente da classe. É calculada dividindo o número de predições corretas pelo número total de predições.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.1)$$

Essas acurácias ainda são úteis para identificar outros pontos importantes do nosso modelo, como por exemplo, analisar se o nosso modelo está sofrendo de sobreajuste ou subajuste. Onde:

Sobreajuste (em inglês, *Overfitting*): Acontece quando um modelo aprende muito bem os dados, conhecendo excessivamente os detalhes e ruídos nos dados de treinamento impactando negativamente quando exposto a novos dados.

Subajuste (em inglês, *Underfitting*): Acontece quando um modelo não consegue ter um bom treinamento com os dados fornecidos, tendo, em muitas vezes, um desempenho ruim.

Relatório de classificação (em inglês, *Classification Report*): É a fração relevante dos exemplos, verdadeiros positivos, entre todos os exemplos que foram previstos para pertencerem a uma certa classe. Esta métrica pode ser definida também como a razão entre o número de itens classificados corretamente como verdadeiros positivos e o total de exemplos classificados como positivos.

$$Precision = \frac{TP}{TP + FP} \quad (4.2)$$

Revocação (em inglês, *Recall*): É a proporção dos verdadeiros positivos entre todas as observações que realmente são positivas em seu conjunto de dados. Esta métrica busca mostrar dentre todas as positivas, quantas o modelo conseguiu identificar como positiva.

$$Recall = \frac{TP}{TP + FN} \quad (4.3)$$

F1-Score ou F-measure: Esta métrica considera tanto a revocação quanto a precisão. Ela é a média harmônica entre estas duas outras métricas. Geralmente utilizada quando temos classes desbalanceadas. A melhor pontuação (score) é 1.0 e a pior é 0.0.

$$F1 - Score = \frac{2.(Precision.Recall)}{Precision + Recall} \quad (4.4)$$

É interessante observar, que, como a métrica F1-score depende tanto da revocação quanto da precisão, se o modelo obtiver um bom resultado nela, quer dizer que ele é um modelo capacitado para acertar suas predições pois tem a precisão alta e recuperar os exemplos da classe que possui interesse devido à alta revocação.

Suporte (em inglês, *Support*): É o número atual de ocorrências da classe no conjunto de dados especificado. Caso o suporte não esteja balanceado, ele pode estar indicando uma falha estrutural nas scores reportadas do classificador, também pode acusar a necessidade de rever o conjunto de dados.

4.2 RESULTADO DOS MODELOS

Após vistas as métricas que foram utilizadas, será apresentado a aplicação delas nos modelos gerados.

Para avaliar e comparar os modelos foi criado o código em python (“compare_models.py”) para facilitar a visualização e entendimento dos mesmos. Este código importa os modelos

previamente exportados, faz a predição e compara os resultados aplicando as métricas vistas na seção anterior.

Como exemplo, podemos ver os resultados do modelo "Decision Tree" abaixo:

Model: Decision_Tree

Confusion Matrix:

[424	0	0	0	0	0	0	1	0	0	0	2	0	0]
[0	379	19	0	0	0	0	0	0	0	0	0	0	0]
[0	30	377	0	0	0	1	0	0	0	0	0	0	0]
[0	0	0	395	0	0	0	5	0	0	0	0	0	0]
[0	0	0	0	407	0	0	0	18	0	0	0	0	0]
[0	0	0	0	0	438	0	0	0	0	0	0	0	0]
[0	0	3	0	0	0	432	0	1	0	0	0	0	1]
[0	0	0	4	0	0	0	407	0	0	0	0	0	0]
[0	2	0	0	19	0	0	0	407	0	0	0	0	0]
[0	0	0	0	0	0	0	0	0	455	4	0	0	0]
[0	0	0	0	0	0	0	0	0	1	450	0	0	1]
[0	0	0	0	0	0	0	0	0	0	0	455	0	0]
[0	0	0	0	2	0	0	0	1	0	0	0	389	0]
[0	0	0	0	0	0	0	0	0	0	1	0	0	469]]

Figura 11 – Decision Tree - Matriz de Confusão

Fonte: Autor

Na imagem acima, podemos ver a sua matriz de confusão, ela é um pouco maior do que a mostrada na seção anterior devido à quantidade de resultados possíveis, ou seja, por causa da classificação possuir 14 sinais diferentes. Com isso, é gerado uma matriz 14x14 que contempla, os TP, FP, TN ou FN conforme visto previamente.

Classification Report:

	precision	recall	f1-score	support
flicker	1.00	0.99	1.00	427
flicker_and_harmonic	0.92	0.95	0.94	398
harmonico	0.94	0.92	0.93	408
interruption	0.99	0.99	0.99	400
interruption_and_harmonic	0.95	0.96	0.95	425
notches	1.00	1.00	1.00	438
oscillatory_transient_and_harmonic	1.00	0.99	0.99	437
sag	0.99	0.99	0.99	411
sag_and_harmonic	0.95	0.95	0.95	428
senoidal	1.00	0.99	0.99	459
spikes	0.99	1.00	0.99	452
swell	1.00	1.00	1.00	455
swell_and_harmonic	1.00	0.99	1.00	392
transitorio_oscilatorio	1.00	1.00	1.00	470
accuracy			0.98	6000
macro avg	0.98	0.98	0.98	6000
weighted avg	0.98	0.98	0.98	6000

Accuracy score (training): 0.979
Accuracy score (validation): 0.981

Figura 12 – Decision Tree - Relatório em tabela

Fonte: Autor

Na Figura 12, temos o relatório da classificação. Este relatório contém métricas como a precisão, revocação, f1-score, suporte, acurácia e média de cada uma. Podemos observar também que os valores são diferentes para cada um dos 14 sinais, isso se dá devido à vários fatores, como por exemplo, o número de sinais de cada um existente no conjunto de dados utilizado, tipo do sinal e os resultados obtidos após a extração de características. Vale lembrar, que estes resultados em sua maioria são mostrados como porcentagem, onde 1.00 indica 100% e 0.00 indica 0%. Apenas o suporte é retornado como número total de ocorrências conforme visto na seção anterior.

Também temos a acurácia de treinamento e validação, elas são bem interessantes para termos uma ideia rápida do resultado do nosso modelo e também, podem ser indicadores de sobreajuste ou subajuste.

Como podemos ver nas duas imagens acima, os valores para o modelo “Decision Tree” foram satisfatórios, alcançando valores com uma média em torno de 98%, isso significa, que o modelo tem uma precisão de aproximadamente 98% para classificação

destes distúrbios.

Com isso em mente, será apresentado em sequência como os demais modelos se comportaram. Para facilitar uma primeira visualização destes resultados, foi criado a tabela abaixo, contendo a média dos valores de cada métrica para os modelos utilizados.

	Modelos	Accuracy Score	Precision Score	Recall Score	F1 Score
0	XGBoost	0.983	0.983	0.983	0.983
1	SVM	0.932	0.934	0.932	0.932
2	KNN	0.924	0.926	0.924	0.924
3	Naive_Bayes	0.821	0.847	0.821	0.823
4	Decision_Tree	0.981	0.981	0.981	0.981
5	Bagged_Decision_Tree	0.978	0.979	0.978	0.979
6	Random_Forest	0.986	0.986	0.986	0.986
7	Gradient_Boosting_Tree	0.934	0.934	0.934	0.934

Figura 13 – Resumo - Comparação entre os modelos

Fonte: Autor

Logo, podemos perceber que os modelos obtiveram resultados interessantes após o treinamento e predição, estando estes, em sua maioria, acima de 90%. Podemos ver também, que o modelo com maior média foi o “Random Forest“, que, por gerar diversas árvores aleatoriamente no seu algoritmo, tende a ter maior facilidade para a classificação de diversos rótulos diferentes. É possível ver também, que o modelo com a média mais baixa, foi o “Naive Bayes“, o que já era esperado, devido à dificuldade deste em classificar os dados utilizados. É importante lembrar que, estes valores tendem a variar caso seus hiperparâmetros sejam modificados, além de sofrerem leves alterações caso gerados novamente, devido às propriedades aleatórias presentes nos algoritmos.

Após esta visão geral, será apresentado abaixo os resultados mais completos de cada modelo, conforme visto com o modelo “Decision Tree“ acima.

```

Model: XGBoost

Confusion Matrix:
[[427  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0 363 33  0  0  0  1  0  1  0  0  0  0]
 [  0 16 391  0  0  0  0  0  1  0  0  0  0]
 [  0  0  0 396  0  0  0  4  0  0  0  0  0]
 [  0  0  0  0 406  0  0  0 19  0  0  0  0]
 [  0  0  0  0  0 438  0  0  0  0  0  0  0]
 [  0  0  1  0  0  0 436  0  0  0  0  0  0]
 [  0  0  0  3  0  0  0 408  0  0  0  0  0]
 [  0  4  1  0  9  0  0  0 414  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0 459  0  0  0]
 [  0  0  0  0  0  0  0  0  0  4 448  0  0]
 [  1  0  0  0  0  0  0  0  0  0  0 454  0]
 [  0  0  0  0  0  0  0  0  0  0  0  0 392]
 [  0  0  0  0  0  0  1  0  0  0  1  0 468]]

Classification Report:

```

						precision	recall	f1-score	support
					flicker	1.00	1.00	1.00	427
					flicker_and_harmonic	0.95	0.91	0.93	398
					harmonic	0.92	0.96	0.94	408
					interruption	0.99	0.99	0.99	400
					interruption_and_harmonic	0.98	0.96	0.97	425
					notches	1.00	1.00	1.00	438
					oscillatory_transient_and_harmonic	1.00	1.00	1.00	437
					sag	0.99	0.99	0.99	411
					sag_and_harmonic	0.95	0.97	0.96	428
					senoidal	0.99	1.00	1.00	459
					spikes	1.00	0.99	0.99	452
					swell	1.00	1.00	1.00	455
					swell_and_harmonic	1.00	1.00	1.00	392
					transitorio_oscilatorio	1.00	1.00	1.00	470
					accuracy			0.98	6000
					macro avg	0.98	0.98	0.98	6000
					weighted avg	0.98	0.98	0.98	6000

```

Accuracy score (training): 0.985
Accuracy score (validation): 0.983

```

Figura 14 – Resultados - XGBoost

Fonte: Autor

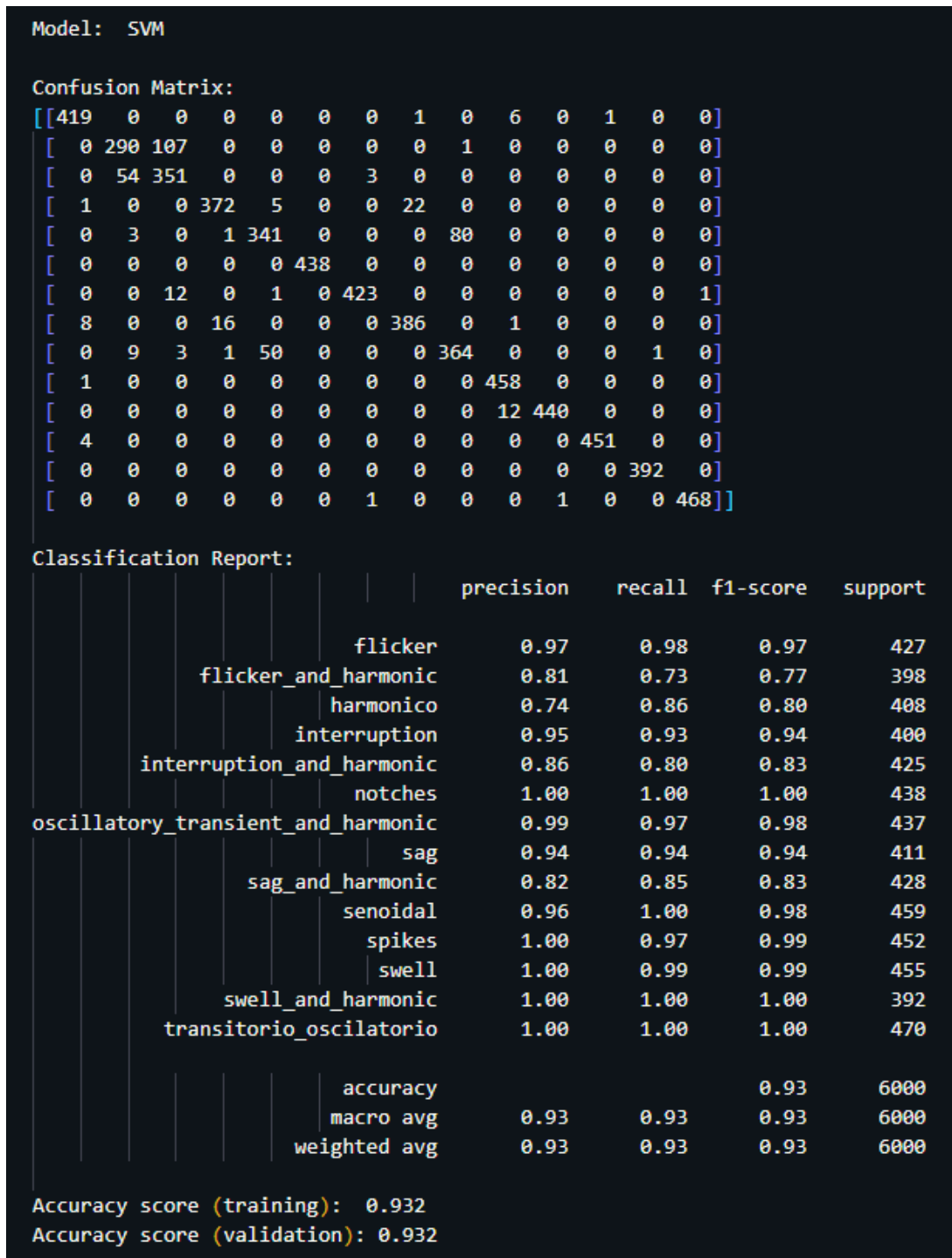


Figura 15 – Resultados - SVM

Fonte: Autor

```

Model: KNN

Confusion Matrix:
[[406  0  0  0  0  0  5  0 15  0  1  0  0]
 [  0 317 81  0  0  0  0  0  0  0  0  0  0]
 [  0 41 366  0  0  0  0  0  1  0  0  0  0]
 [  2  0  0 380  1  0  0 17  0  0  0  0  0]
 [  0  2  0  6 376  0  0  2 39  0  0  0  0]
 [  0  0  0  0  0 438  0  0  0  0  0  0  0]
 [  0 12  7  0  0  0 378  0  0  0  0  0 40]
 [  4  0  0 15  1  0  0 385  0  6  0  0  0]
 [  0 18 13  2 35  0  0  6 354  0  0  0  0]
 [  2  0  0  0  0  0  0  0  0 457  0  0  0]
 [  0  0  0  0  0 15  0  0  0 13 424  0  0]
 [  7  0  0  0  0  0  0  0  0  0  0 448  0]
 [  0  0  0  0  0  0  0  0  0  0  0 10 382]
 [  0  0  0  0  0  0 18  0  0  2 14  0  0 436]]

Classification Report:

```

	precision	recall	f1-score	support
flicker	0.96	0.95	0.96	427
flicker_and_harmonic	0.81	0.80	0.80	398
harmonic	0.78	0.90	0.84	408
interruption	0.94	0.95	0.95	400
interruption_and_harmonic	0.91	0.88	0.90	425
notches	0.97	1.00	0.98	438
oscillatory_transient_and_harmonic	0.95	0.86	0.91	437
sag	0.93	0.94	0.93	411
sag_and_harmonic	0.90	0.83	0.86	428
senoidal	0.93	1.00	0.96	459
spikes	0.97	0.94	0.95	452
swell	0.98	0.98	0.98	455
swell_and_harmonic	1.00	0.97	0.99	392
transitorio_oscilatorio	0.92	0.93	0.92	470
accuracy			0.92	6000
macro avg	0.92	0.92	0.92	6000
weighted avg	0.93	0.92	0.92	6000

```

Accuracy score (training): 0.920
Accuracy score (validation): 0.924

```

Figura 16 – Resultados - KNN

Fonte: Autor

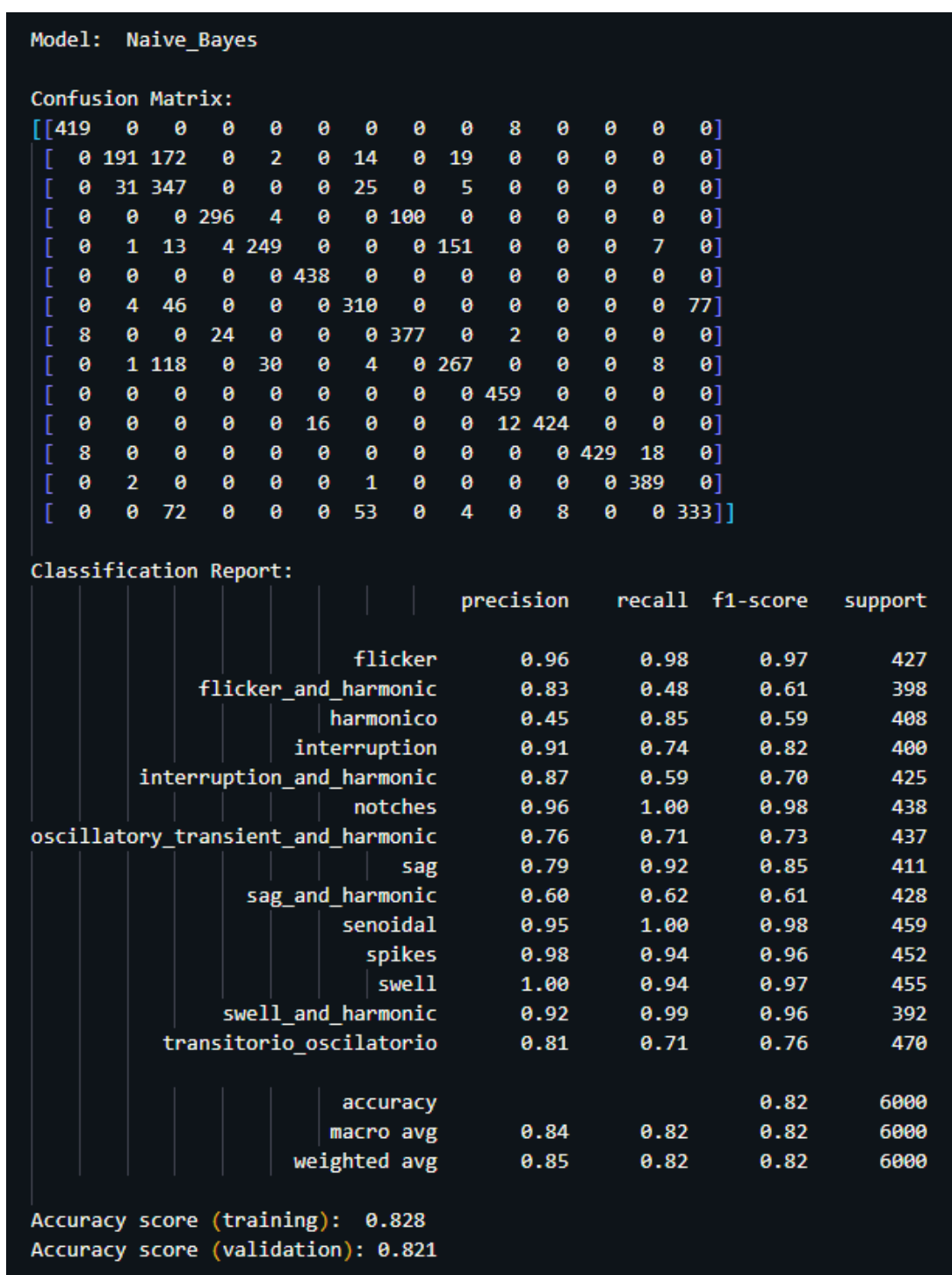


Figura 17 – Resultados - Naive Bayes

Fonte: Autor

Model: Decision_Tree

Confusion Matrix:

```
[[424  0  0  0  0  0  0  1  0  0  0  2  0  0]
 [  0 379 19  0  0  0  0  0  0  0  0  0  0  0]
 [  0  30 377  0  0  0  1  0  0  0  0  0  0  0]
 [  0  0  0 395  0  0  0  5  0  0  0  0  0  0]
 [  0  0  0  0 407  0  0  0 18  0  0  0  0  0]
 [  0  0  0  0  0 438  0  0  0  0  0  0  0  0]
 [  0  0  3  0  0  0 432  0  1  0  0  0  0  1]
 [  0  0  0  4  0  0  0 407  0  0  0  0  0  0]
 [  0  2  0  0 19  0  0  0 407  0  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0 455  4  0  0  0]
 [  0  0  0  0  0  0  0  0  0  0 1 450  0  0  1]
 [  0  0  0  0  0  0  0  0  0  0  0 455  0  0]
 [  0  0  0  0  2  0  0  0  1  0  0  0 389  0]
 [  0  0  0  0  0  0  0  0  0  0  1  0  0 469]]
```

Classification Report:

								precision	recall	f1-score	support
							flicker	1.00	0.99	1.00	427
							flicker_and_harmonic	0.92	0.95	0.94	398
							harmonic	0.94	0.92	0.93	408
							interruption	0.99	0.99	0.99	400
							interruption_and_harmonic	0.95	0.96	0.95	425
							notches	1.00	1.00	1.00	438
							oscillatory_transient_and_harmonic	1.00	0.99	0.99	437
							sag	0.99	0.99	0.99	411
							sag_and_harmonic	0.95	0.95	0.95	428
							senoidal	1.00	0.99	0.99	459
							spikes	0.99	1.00	0.99	452
							swell	1.00	1.00	1.00	455
							swell_and_harmonic	1.00	0.99	1.00	392
							transitorio_oscilatorio	1.00	1.00	1.00	470
							accuracy			0.98	6000
							macro avg	0.98	0.98	0.98	6000
							weighted avg	0.98	0.98	0.98	6000

Accuracy score (training): 0.979

Accuracy score (validation): 0.981

Figura 18 – Resultados - Decision Tree

Fonte: Autor

Model: Bagged_Decision_Tree

Confusion Matrix:

```
[[426  0  0  0  0  0  0  1  0  0  0  0  0  0]
 [  0 370 27  0  0  0  1  0  0  0  0  0  0  0]
 [  0  31 375  0  0  0  2  0  0  0  0  0  0  0]
 [  0  0  0 399  0  0  0  1  0  0  0  0  0  0]
 [  0  0  0  0 406  0  0  0 19  0  0  0  0  0]
 [  0  0  0  0  0 438  0  0  0  0  0  0  0  0]
 [  0  1  0  0  0  0 436  0  0  0  0  0  0  0]
 [  3  0  0  3  0  0  0 405  0  0  0  0  0  0]
 [  0  6  1  0 18  0  0  0 403  0  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0 459  0  0  0  0]
 [  0  0  0  0  0  1  0  0  0  7 443  0  0  1]
 [  3  0  0  0  0  0  0  0  0  0  0 452  0  0]
 [  0  0  0  0  0  0  0  0  1  0  0  0 391  0]
 [  0  0  0  0  0  0  0  0  0  0  2  0  0 468]]
```

Classification Report:

								precision	recall	f1-score	support
							flicker	0.99	1.00	0.99	427
							flicker_and_harmonic	0.91	0.93	0.92	398
							harmonic	0.93	0.92	0.92	408
							interruption	0.99	1.00	1.00	400
							interruption_and_harmonic	0.96	0.96	0.96	425
							notches	1.00	1.00	1.00	438
							oscillatory_transient_and_harmonic	0.99	1.00	1.00	437
							sag	1.00	0.99	0.99	411
							sag_and_harmonic	0.95	0.94	0.95	428
							senoidal	0.98	1.00	0.99	459
							spikes	1.00	0.98	0.99	452
							swell	1.00	0.99	1.00	455
							swell_and_harmonic	1.00	1.00	1.00	392
							transitorio_oscilatorio	1.00	1.00	1.00	470
							accuracy			0.98	6000
							macro avg	0.98	0.98	0.98	6000
							weighted avg	0.98	0.98	0.98	6000

Accuracy score (training): 0.982

Accuracy score (validation): 0.979

Figura 19 – Resultados - Bagged Decision Tree

Fonte: Autor

```

Model: Random_Forest

Confusion Matrix:
[[427  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0 370 26  0  0  0  1  0  1  0  0  0  0]
 [  0  10 398  0  0  0  0  0  0  0  0  0  0]
 [  0  0  0 400  0  0  0  0  0  0  0  0  0]
 [  0  0  0  0 414  0  0  0 11  0  0  0  0]
 [  0  0  0  0  0 438  0  0  0  0  0  0  0]
 [  0  1  0  0  0  0 436  0  0  0  0  0  0]
 [  1  0  0  1  0  0  0 409  0  0  0  0  0]
 [  0  4  1  0 17  0  0  0 406  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0 459  0  0  0]
 [  0  0  0  0  0  0  0  0  0  7 445  0  0]
 [  1  0  0  0  0  0  0  0  0  0  0 454  0]
 [  0  0  0  0  1  0  0  0  0  0  0  0 391]
 [  0  0  0  0  0  0  0  0  0  0  1  0 469]]

Classification Report:

```

	precision	recall	f1-score	support
flicker	1.00	1.00	1.00	427
flicker_and_harmonic	0.96	0.93	0.95	398
harmonico	0.94	0.98	0.96	408
interruption	1.00	1.00	1.00	400
interruption_and_harmonic	0.96	0.97	0.97	425
notches	1.00	1.00	1.00	438
oscillatory_transient_and_harmonic	1.00	1.00	1.00	437
sag	1.00	1.00	1.00	411
sag_and_harmonic	0.97	0.95	0.96	428
senoidal	0.98	1.00	0.99	459
spikes	1.00	0.98	0.99	452
swell	1.00	1.00	1.00	455
swell_and_harmonic	1.00	1.00	1.00	392
transitorio_oscilatorio	1.00	1.00	1.00	470
accuracy			0.99	6000
macro avg	0.99	0.99	0.99	6000
weighted avg	0.99	0.99	0.99	6000

```

Accuracy score (training): 0.988
Accuracy score (validation): 0.986

```

Figura 20 – Resultados - Random Forest

Fonte: Autor

```

Model: Gradient_Boosting_Tree

Confusion Matrix:
[[427  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0 301 86  0  0  0  2  0  9  0  0  0  0]
 [  0 59 342  0  0  0  2  0  5  0  0  0  0]
 [  0  0  0 389  1  0  0 10  0  0  0  0  0]
 [  0  2  1  5 341  0  0  0 76  0  0  0  0]
 [  0  0  0  0  0 438  0  0  0  0  0  0  0]
 [  0  3  5  0  0  0 419  0  1  0  0  0  9]
 [  4  0  0  6  0  0  0 401  0  0  0  0  0]
 [  0 16 15  0 50  0  3  0 343  0  0  0  1]
 [  0  0  0  0  0  0  0  0  0 458  1  0  0]
 [  0  0  0  0  0  0  0  0  0 12 439  0  0]
 [  3  0  0  0  0  0  0  0  0  0  0 452  0]
 [  0  0  0  0  1  0  0  0  0  0  0  0 391]
 [  0  0  0  0  0  0  4  0  0  0  4  0 462]]

Classification Report:

```

						precision	recall	f1-score	support
					flicker	0.98	1.00	0.99	427
					flicker_and_harmonic	0.79	0.76	0.77	398
					harmonic	0.76	0.84	0.80	408
					interruption	0.97	0.97	0.97	400
					interruption_and_harmonic	0.87	0.80	0.83	425
					notches	1.00	1.00	1.00	438
					oscillatory_transient_and_harmonic	0.97	0.96	0.97	437
					sag	0.98	0.98	0.98	411
					sag_and_harmonic	0.79	0.80	0.80	428
					senoidal	0.97	1.00	0.99	459
					spikes	0.99	0.97	0.98	452
					swell	1.00	0.99	1.00	455
					swell_and_harmonic	1.00	1.00	1.00	392
					transitorio_oscilatorio	0.98	0.98	0.98	470
					accuracy			0.93	6000
					macro avg	0.93	0.93	0.93	6000
					weighted avg	0.93	0.93	0.93	6000

```

Accuracy score (training): 0.934
Accuracy score (validation): 0.934

```

Figura 21 – Resultados - Gradient Boosting Tree

Fonte: Autor

Após verificar todas as imagens, podemos ver que o número de amostras de cada tipo de sinal foi parecido através do suporte, mostrando que o treinamento foi feito utilizando dados balanceados.

Podemos observar a precisão, revocação, f1 score e acurácia de cada modelo para

verificar a eficácia deles. Com isso, podemos observar no geral, os modelos obtiveram bons resultados, sendo que, dentre todos eles, apenas o modelo “Naive Bayes”, conforme visto previamente, possui um resultado abaixo dos demais. O que levou este modelo à um resultado inferior, foi a sua suposição de preditores independentes, que dificultou a classificação dos sinais contendo harmônicos conforme podemos ver nos seus resultados.

Em contra partida, o modelo com melhores resultados foi o Random Forest, devido à sua estratégia, que gera árvores de decisão aleatórias e utiliza da combinação de diversos classificadores para prover a solução para o devido problema.

Podemos ver também que, em uma comparação geral, os modelos tendem a ter mais dificuldade em classificar corretamente os distúrbios do tipo “harmônico” ou derivados. Podemos ver que nos algoritmos SVM, KNN, Naive Bayes e gradient boosting tree, isto ficou mais evidente.

Com os modelos gerados e visto os resultados dos mesmos, foram feitos alguns testes, com novos sinais, onde eles foram classificados corretamente.

Após estes testes, foi proposto pelo professor Leandro Manso, tentar classificar alguns sinais extraídos de uma rede elétrica com distúrbios, fornecidos por ele. O teste não foi concluído, pois estes sinais possuíam certos parâmetros diferentes dos sinais simulados. Para uma correta classificação dos mesmos, seria necessário um conjunto de dados grande o suficiente para o treinamento de novos modelos. Entretanto, não foi possível adquirir este conjunto de dados e este teste proposto para trabalhos futuros.

4.3 CONCLUSÕES PARCIAIS

Este capítulo descreveu as métricas utilizadas para a avaliação dos modelos gerados para classificar os distúrbios, seguido dos resultados obtidos com cada modelo ao fazer a classificação. Foi possível observar, que os modelos gerados se mostraram bem efetivos, apresentando, em sua maioria, uma precisão acima de 90% na classificação.

5 CONCLUSÃO

Após um estudo do problema e criação de uma estratégia para resolução do mesmo, foi possível criar modelos de aprendizado de máquina capazes de classificar os distúrbios de qualidade de energia. Para isto, foi empregado diversos conhecimentos, abrangendo as áreas de potência, processamento de sinais e inteligência artificial.

Foi possível verificar também, a diferença entre os modelos, onde os decorrentes de árvores de decisões, obtiveram em geral, melhores resultados. Em especial, o Random Forest obteve o valores superiores devido à sua estratégia robusta de gerar diversas árvores de decisão aleatórias e utiliza da combinação de diversos classificadores para prover a solução para o devido problema. Enquanto o Naive Bayes, obteve valores inferiores quando comparado aos outros modelos, devido à sua suposição de preditores independentes que dificultou a classificação dos sinais contendo harmônicos.

Com isso, este trabalho atingiu os objetivos propostos no capítulo 1, ao conseguir gerar modelos com uma taxa de acerto aproximada acima de 90% no geral, capazes assim, de classificar os distúrbios de qualidade de energia.

5.1 TRABALHOS FUTUROS

Para trabalhos futuros, propõe-se:

- Aplicar o processo para um conjunto de dados extraídos de uma rede elétrica real e utilizar os mesmos para treinar os modelos. Foi feita uma tentativa de classificação de alguns sinais obtidos desta forma, repassados pelo professor Leandro Manso. Entretanto, não haviam sinais suficientes para criar um conjunto de dados e, devido à diferença dos parâmetros entre esses sinais e os simulados, foi constatado a necessidade do treinamento dos modelos com um conjunto de dados daqueles sinais.
- Adicionar novos distúrbios ao processo, buscando conseguir classificar mais tipos de distúrbios.
- Criar uma interface ou hospedar os modelos de forma a facilitar o uso dos modelos gerados.

REFERÊNCIAS

- [1] Anyoha, Rockwell. The History of Artificial Intelligence. SITN, 2021. Disponível em: <https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/>. Acesso em: 02 de set. de 2021.
- [2] RAY, Sunil. Commonly used Machine Learning Algorithms (with Python and R Codes). Analytics Vidhya, 2021. Disponível em: <https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/>. Acesso em: 02 de set. de 2021.
- [3] Stockwell, Robert Glenn, Lalu Mansinha, and R. P. Lowe. “Localization of the complex spectrum: the S transform.” *IEEE transactions on signal processing* 44.4 (1996): 998-1001.
- [4] Stockwell, R. G. “Why use the S-transform.” *Pseudo-differential operators: partial differential equations and time-frequency analysis* 52 (2007): 279-309.
- [5] CARELI, Paulo Henrique Pereira. power-quality-disturbance-with-machine-learning. Github, 2021. Disponível em: <https://github.com/PauloCareli/power-quality-disturbances-with-machine-learning/>. Acesso em: 29 de ago. de 2021.
- [6] Bravo-Rodríguez, Juan Carlos, Francisco J. Torres, and María D. Borrás. “Hybrid machine learning models for classifying power quality disturbances: A comparative study.” *Energies* 13.11 (2020): 2761.
- [7] pandas. pandas, 2021. Disponível em: <https://pandas.pydata.org/>. Página inicial. Acesso em: 29 de ago. de 2021.
- [8] SATRIANO, Claudio. Stockwell. GitHub, 2021. Disponível em: <https://github.com/claودیsf/stockwell>. Acesso em: 29 de ago. de 2021.
- [9] Numba: A High Performance Python Compiler. Numba, 2021. Página inicial. Disponível em: <http://numba.pydata.org>. Acesso em: 29 de ago. de 2021.
- [10] NumPy. NumPy, 2021. Disponível em: <https://numpy.org/doc/stable/index.html>. Página inicial. Acesso em: 29 de ago. de 2021.
- [11] SciPy. SciPy, 2021. Disponível em: <https://www.scipy.org/>. Página inicial. Acesso em: 29 de ago. de 2021.
- [12] matplotlib. matplotlib, 2021. Página inicial. Disponível em: <https://matplotlib.org/>. Acesso em: 29 de ago. de 2021.
- [13] Scikit-learn Machine Learning in Python. Scikit-learn, 2021. Página inicial. Disponível em: <https://scikit-learn.org/stable/index.html>. Acesso em: 29 de ago. de 2021.
- [14] Classification Visualizers. Scikit Yellowbrick, 2021. Disponível em: <https://www.scikit-yb.org/en/latest/api/classifier/index.html>. Acesso em: 29 de ago. de 2021.

- [15] Find, install and publish Python packages with the Python Package Index. PyPI, 2021. Página inicial. Disponível em: <<https://pypi.org/>>. Acesso em: 29 de ago. de 2021.
- [16] Towards data science. Towards data science, 2021. Página inicial. Disponível em: <<https://towardsdatascience.com/>>. Acesso em: 29 de ago. de 2021.