UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

MÁQUINA DE VETORES DE SUPORTE E TÉCNICAS ESTATÍSTICAS PARA PREVISÃO E CLASSIFICAÇÃO DE FALHA DE TURBINAS EÓLICAS

TRABALHO DE CONCLUSÃO DE CURSO

Kauê Augusto Delazzeri

Santa Maria, RS

2020

Kauê Augusto Delazzeri

MÁQUINA DE VETORES DE SUPORTE E TÉCNICAS ESTATÍSTICAS PARA PREVISÃO E CLASSIFICAÇÃO DE FALHA DE TURBINAS EÓLICAS

Monografia apresentada ao Curso em Engenharia de Controle e Automação na Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Engenharia de Controle e Automação**

Orientador: Prof. Dr. Daniel Fernando Tello Gamarra

Santa Maria, RS

2020
Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Engenharia de Controle e Automação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Conclusão de Curso

MÁQUINA DE VETORES DE SUPORTE E TÉCNICAS ESTATÍSTICAS PARA PREVISÃO E CLASSIFICAÇÃO DE FALHA DE TURBINAS EÓLICAS

elaborado por
Kauê Augusto Delazzeri

como requisito parcial para obtenção do grau de
Bacharel em Engenharia de Controle e Automação

COMISSÃO EXAMINADORA:

Daniel Fernando Tello Gamarra, Dr. Eng. (UFSM)
(Presidente/Orientador)

Claiton Moro Franchi, Dr. Eng. (UFSM)

Humberto Pinheiro, Ph.D. (UFSM)

RESUMO

MÁQUINA DE VETORES DE SUPORTE E TÉCNICAS ESTATÍSTICAS PARA PREVISÃO E CLASSIFICAÇÃO DE FALHA DE TURBINAS EÓLICAS

AUTOR: KAUÊ AUGUSTO DELAZZERI
ORIENTADOR: DANIEL FERNANDO TELLO GAMARRA

A energia eólica é dominante entre a geração de energias renováveis no Brasil, devido ao alto crescimento nos últimos anos, há uma necessidade de desenvolver técnicas precisas de diagnóstico de falhas em aerogeradores, para diminuir o tempo de inatividade devido a defeitos ou manutenções desnecessárias. A maioria das técnicas tradicionais de identificação de falha não são capazes de detectar defeitos causados por assimetria/desequilíbrio de massa ou ângulo do sistema de passo nas pás, avarias comuns em aerogeradores. Métodos como a análise de vibrações, que é competente para o reconhecimento desses tipos de condições, exige a instalação de sensores extras em locais que pode ser de difícil acesso. Uma nova abordagem seria o diagnóstico baseado em sinais elétricos dos geradores da turbina eólica, pois, se mostram uma opção mais confiável e econômica, por não necessitar a instalação de sensores de vibração. Para isto, uma estrutura com TurbSim/FAST/Simulink foi utilizada para simular sinais elétricos gerados a partir de uma turbina eólica de 1,5 MW para diferentes cenários de influxo de vento, e parâmetros de desequilíbrio de massa nas pás e ângulos de passo. Algoritmos de aprendizado de máquina, como SVM, e ferramentas estatísticas, como PCA, LDA, MVPE e FFT, foram aplicados nos dados de simulação para a identificação de falhas. Na identificação de condições de falha foi obtido, no melhor dos casos, acurácia de 100,00% com sinais elétricos do gerador de turbinas eólicas.

Palavras chaves: Aprendizado de máquina, detecção de falhas em aerogeradores, previsão de falha em turbinas eólicas, SVM

ABSTRACT

SUPPORT VECTOR MACHINE AND STATISTICAL TECHNIQUES FOR WIND TURBINE FAULT PREDICTION

AUTHOR: KAUÊ AUGUSTO DELAZZERI
ADVISOR: DANIEL FERNANDO TELLO GAMARRA

Wind energy is dominant technology among the generation of renewable energies in Brazil and due to the high growth in recent years there is a need to develop accurate fault diagnosis techniques in wind turbines to reduce downtime due to defects or unnecessary maintenance. Most traditional fault identification techniques are not able to detect defects caused by asymmetry / unbalance in mass or pitch angle on the blades, common faults in wind turbines. Methods such as vibration analysis, which is competent, require the installation of extra sensors in places that may be difficult to access. A new approach would be the diagnosis based on electrical signals from the wind turbine generators, as they prove to be a more reliable and economical option as they do not require the installation of vibration sensors. So, a structure with TurbSim / FAST / Simulink was used to simulate electrical signals generated from a 1.5 MW wind turbine for different scenarios of wind influx and mass unbalance parameters in the blades and pitch angles, machine learning algorithms and statistical tools such as PCA, LDA, EPVM and FFT were applied to the simulation data for fault identification. In the identification of failure conditions, at best, 100.00% accuracy was obtained with electrical signals from the wind turbine generator.

Keywords: Machine learning, wind turbines fault detection, wind turbines failure predict, SVM.

LISTA DE FIGURAS

Figura 1.1 - Participação de energias renováveis na nova capacidade de geração de energia instalada em 2019 9
Figura 1.2 - Matriz energética brasileira 10
Figura 1.3 - Projeção da Energia eólica no Brasil 11
Figura 1.4 - Turbina Eólica 13
Figura 2.1 - PCA aplicado a dados 19
Figura 2.2 - Comparação entre PCA e LDA 21
Figura 2.3 - Comparação entre PCA e LDA 23
Figura 2.4 - id x iq 25
Figura 2.5 - Pontos a serem determinados para cálculo de EMD 28
Figura 2.6 - EMD aplicada a função Seno 29
Figura 2.7 - Função (em vermelho) gerada pela SVM de margens suaves 30
Figura 2.8 - Classificação em aprendizado supervisionado 31
Figura 2.9 - Diagrama contínuo de "Um contra todos" na classificação de 3 classes 32
Figura 2.10 - Classificações possíveis pela SVM de margens rígidas 32
Figura 2.11 - Margens de separação 33
Figura 2.12 - Distância entre margens 34
Figura 2.13 - Exemplo categorização de dados por SVM 35
Figura 2.14 - Exemplo de aplicação de gamma nos dados 36
Figura 2.15 - Escolha de hiperplanos 37
Figura 2.16 - Exemplo de Grid Search para SVM 38
Figura 2.17 - Fluxograma do algoritmo de pesquisa em grade 39
Figura 3.1 - Exemplo de um aerogerador utilizado nas simulações (com 3 pás) 42
Figura 3.2 - Malha Turbisim 43
Figura 3.3 - Esquemático geração de dados 44
Figura 3.4 - Framework da plataforma de geração de dados 44
Figura 3.5 - 175 combinações possíveis de parâmetros 47
Figura 3.6 - Parâmetros que formam o banco de dados 48
Figura 3.7 – Geração de dados 49
Figura 3.8 - Geração de dados 49
Figura 3.9 - Geração de dados 50

Figura 3.10 - Dados de simulação 50
Figura 3.11 - Condições de funcionamento simuladas 51
Figura 3.12 - "Tradução" das condições de funcionamento para o algoritmo 52
Figura 3.13 - Plotagem das correntes do gerador da TE 53
Figura 3.14 - Corrente de gerador no domínio do tempo com funcionamento normal 54
Figura 3.15 - Corrente Ia de gerador no domínio do tempo com condição propensa a falha – 105% de massa em uma pá 54
Figura 3.16 - FFT em Ia normal 55
Figura 3.17 - FFT em Ia com desequilíbrio 56
Figura 3.18 - Fluxograma de Malik adaptado 57
Figura 3.19 - Gerador elétrico e Motor de ângulo de passo de um aerogerador 59
Figura 3.20 - Fluxograma dos algoritmos 60
Figura 3.21 - Correntes trifásicas do gerador 61
Figura 4.1 - Primeira componente da análise, representa mais de 90% da variância total dos dados 63
Figura 4.2 - Gráfico da representação da variância por componente 64
Figura 4.3 - Acurácia x PCA 64
Figura 4.4 - Valores singulares por componente da PCA 65
Figura 4.5 - Acurácia PCA x LDA 66
Figura 4.6 - Matriz de confusão 68
Figura 4.7 - id e iq 71
Figura 4.8 - ip 71
Figura 4.9 - IP no domínio da frequência 72
Figura 4.10 – Comparação do impacto das estatísticas 74
Figura 4.11 - Matriz de confusão 75
Figura 4.12 - Comparação de banco de dados sem e com "limitação" de parâmetros 77
Figura 4.13 - Classificação de dados de Baixa X Alta turbulência 77
Figura 4.14 - Exemplo de corrente no domínio do tempo com gerador ligado 78
Figura 4.15 - Exemplo de corrente no domínio do tempo com gerador desligado 79
Figura 4.16 - Classificação de dados de Baixa X Alta velocidade de vento 79

LISTA DE TABELAS

Tabela 3.1 - Dados do aerogerador simulado 45
Tabela 3.2 - Parâmetros das simulações de funcionamento dos aerogeradores 45
Tabela 3.3 - Demonstração do desvio padrão 46
Tabela 3.4 - Parâmetros das simulações de funcionamento dos aerogeradores 46
Tabela 3.5 – Parâmetros das simulações de funcionamento dos aerogeradores 46
Tabela 3.6 - Amostra banco de dados da corrente do aerogerador 53
Tabela 3.7 - Parâmetros para Kernel 58
Tabela 3.8 - Parâmetros para C 58
Tabela 3.9 - Parâmetros para Gamma 58
Tabela 3.10 - Parâmetros para Kernel 62
Tabela 3.11 - Parâmetros para C 62
Tabela 3.12 - Parâmetros para Gamma 62
Tabela 4.1 - Acurácia com e sem LDA 66
Tabela 4.2 - Parâmetros escolhidos pela pesquisa em grade 67
Tabela 4.3 - Scores da SVM 69
Tabela 4.4 - Descrição de valores estatísticos 73
Tabela 4.5 - Parâmetros escolhidos pela pesquisa em grade 74
Tabela 4.6 - Score da SVM 76

SUMÁRIO

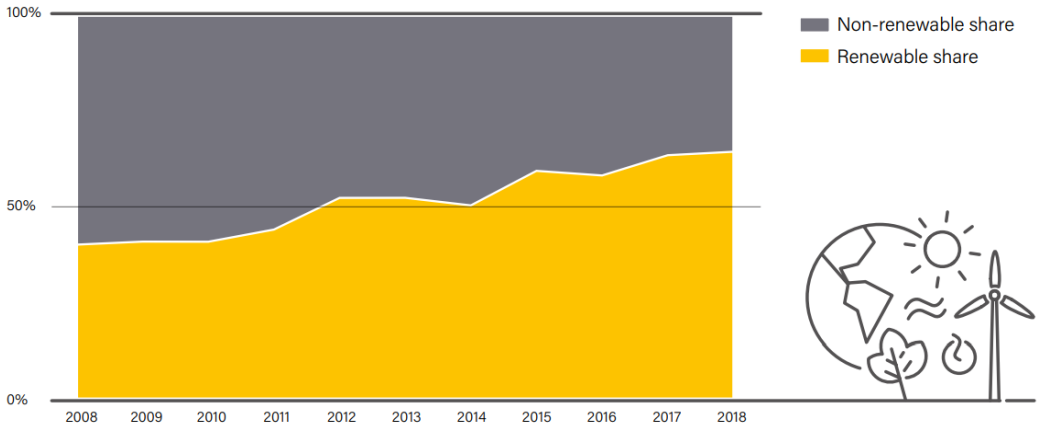
1. INTRODUÇÃO 9
1.2 TRABALHOS RELACIONADOS 14
1.3 OBJETIVOS 16
1.4 ESCOPO DO TRABALHO 17
2. REVISÃO BIBLIOGRÁFICA 18
2.1 TÉCNICAS DE PRÉ-PROCESSAMENTO 18
2.2 MÁQUINA DE VETORES DE SUPORTE 29
3. MATERIAIS E MÉTODOS 41
3.1 PYTHON 41
3.2 DADOS DE SIMULAÇÃO 41
4. APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS 63
4.1 MALIK ADAPTADO 63
4.2 KANDUKURI ADAPTADO 70
5. CONCLUSÃO 81
REFERÊNCIAS 83
6. APÊNDICES 89
6.1 APENDICE A 89
6.2 APENDICE B 93

1. INTRODUÇÃO

As energias renováveis (ER) são provenientes de ciclos naturais como do sol, vento, chuva, marés e energia geotérmica, por isso, são praticamente inesgotáveis. Não alteram o equilíbrio do planeta e se configuram como um conjunto de fontes de energia que podem ser chamadas de não-convencionais, ou seja, aquelas não baseadas nos combustíveis fósseis e hidroelétricas.

As ERs (como a energia eólica, de biomassa e a solar) são formas de energia que se regeneram de uma forma cíclica em uma escala de tempo reduzida quando comparada com energias tradicionais (PACHECO, 2006). Hoje as energias renováveis já representam 25% de toda energia consumida no mundo e são responsáveis por 65% das novas energias criadas para suprir o consumo mundial (IEA, 2019), como vemos na Figura 1.1:

Figura 1.1 - Participação de energias renováveis na nova capacidade de geração de energia instalada em 2019



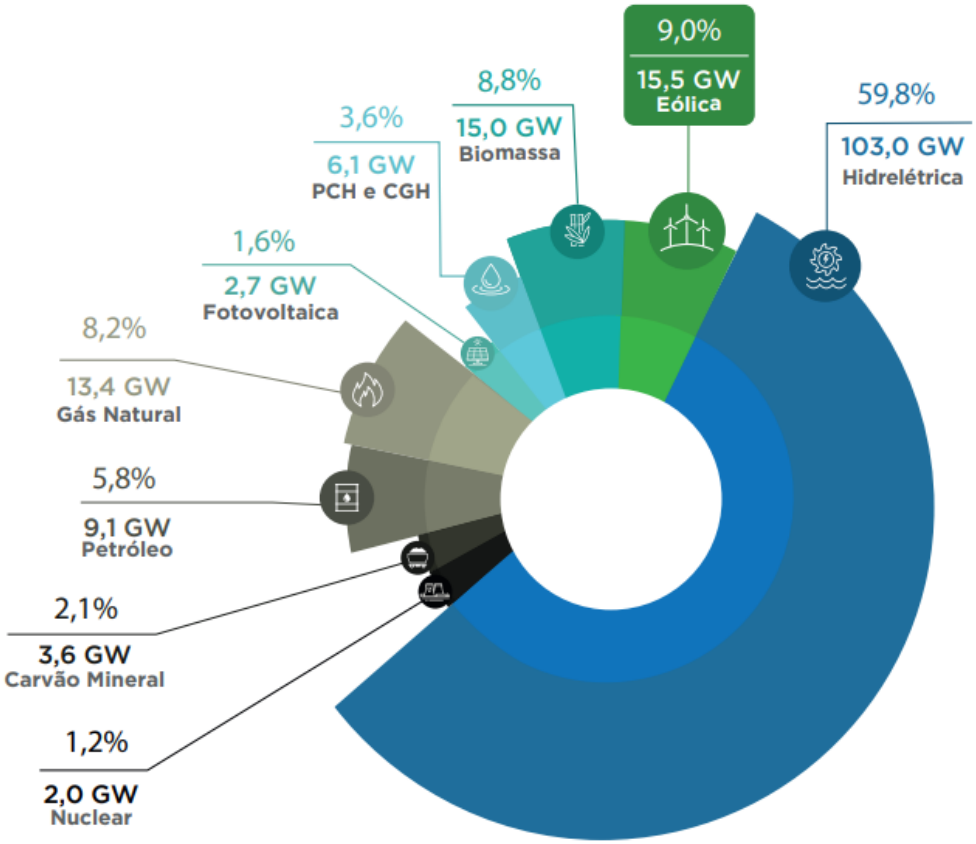
Fonte: (IEA, 2019)

1.1.1 Energia eólica

Energia eólica é a transformação da energia do vento em energia útil. Surge como alternativa à queima de combustíveis fósseis, é abundante, renovável, amplamente distribuída, limpa, não produz emissões de gases de efeito estufa durante a operação, não consome água e ocupa pouca área de terra. São efeitos que, sobre o meio ambiente, são menos problemáticos que os das fontes de combustíveis fósseis (FTHENAKIS, 2009).

De acordo com dados publicados em Abril de 2020 pela Associação Brasileira de Energia Eólica (ABEEólica) essa energia totaliza 15,5 GW de capacidade instalada, alcançando 9,0% da matriz energética total no Brasil, como vemos na Figura 1.2 (ABEEÓLICA, 2020):

Figura 1.2 - Matriz energética brasileira



Fonte: ABEEólica[1]

[1] Disponível em: http://abeeolica.org.br/wp-content/uploads/2020/04/Infovento-15_PT.pdf. Acesso em: 1 jun. 2020

Figura 1.3 - Projeção da Energia eólica no Brasil

Fonte: ABEEólica[2]

Entre os países que mais cresceram na capacidade de geração da energia, está o Brasil (IEA, 2019), cujo, apesar da grande distância para os primeiros colocados, traz como vantagens os recursos naturais, a extensão do território e o tamanho do litoral, que, juntamente com os dados promissores, mostram que a energia eólica no país favorece um cenário oportuno a médio e longo prazo (VICHI, 2009).

1.1.2 Aerogerador

O objeto de estudo do presente trabalho denomina-se aerogerador, equipamento no qual deseja-se detectar as condições de falha. O aerogerador é responsável pela conversão da energia dos ventos em energia elétrica, seus principais componentes são:

- Rotor: Compreende as pás e o cubo (*hub*), onde, as mesmas são fixadas. O eixo do aerogerador pode ser horizontal ou vertical.

[2] Disponível em: http://abeeolica.org.br/wp-content/uploads/2020/04/Infovento-15_PT.pdf. Acesso em: 1 jun. 2020

- Pás: As pás são perfis aerodinâmicos geralmente feitas com um material leve e resistente.
- Gerador: Instalado no interior da nacele, converte a energia mecânica de rotação das pás em energia elétrica.

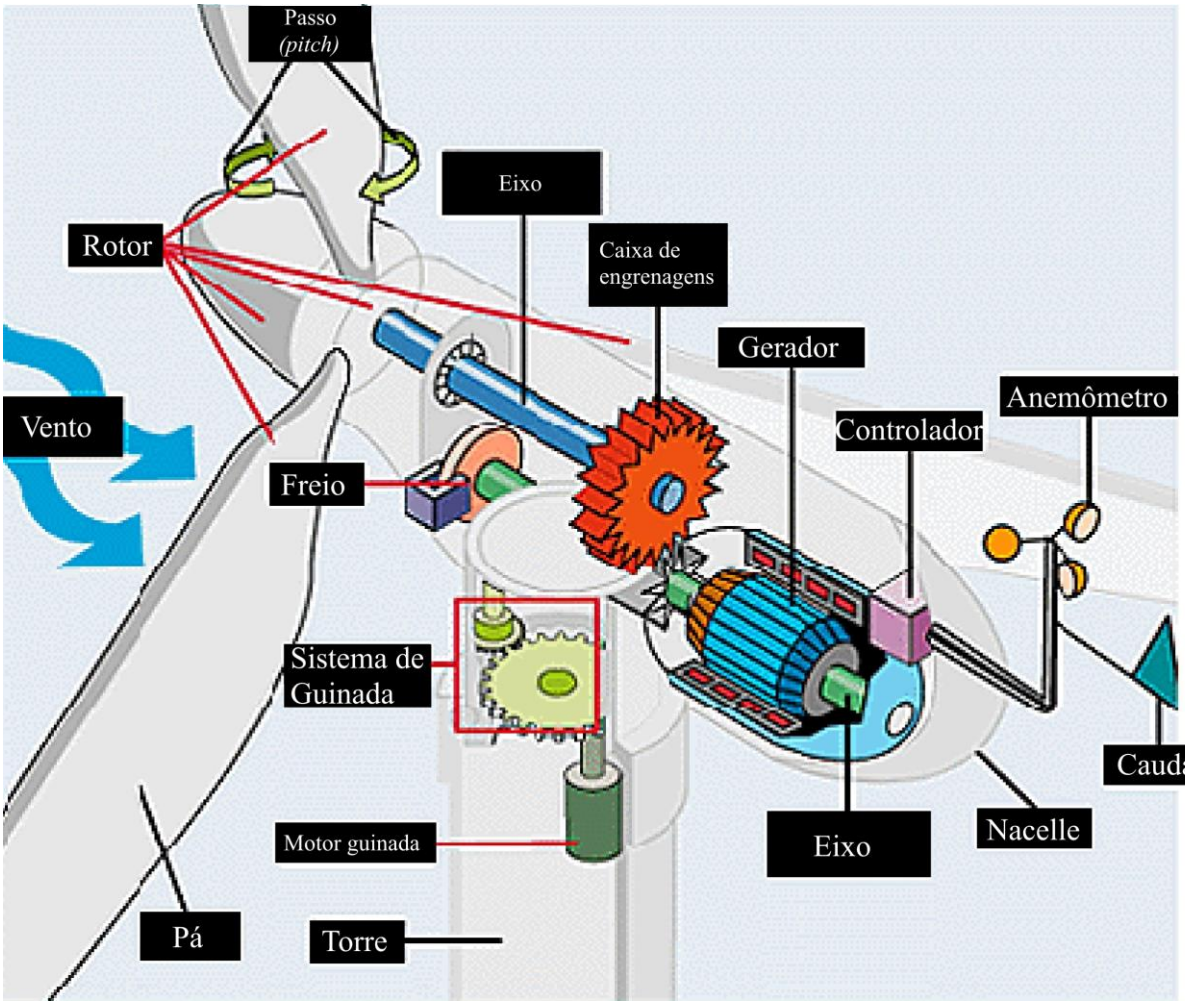
Além desses, o aerogerador também possui outras partes (ATLANTIC ENERGIAS RENOVAVEIS, 2019), conforme Figura 1.4 e também descrito abaixo:

- Anemômetro: Instrumento meteorológico localizado na parte superior e externa da nacele, com a função de medir a velocidade instantânea do vento local.
- Biruta/Cauda (sensor de direção): Também conhecida como *windvane*, é um item meteorológico posicionado do lado do anemômetro, cuja finalidade é medir a direção instantânea do vento incidente.
- Torre: Estrutura responsável por fornecer sustentação e posicionamento do rotor e nacele. As torres podem ser cônicas (de aço ou concreto) ou treliçadas (aço galvanizado).
- Nacele: Componente que fica no topo da torre do aerogerador. Em seu interior, estão abrigados a caixa de multiplicação, o gerador, o transformador, entre outros.
- Sistema de guinada: Componente responsável pela orientação do rotor da turbina eólica (TE) em direção ao vento

Algumas TE podem conter uma caixa de engrenagens (*gearbox*), as turbinas simuladas no presente trabalho não possuem tal componente, porém, a mesma está indicada na Figura 1.4 abaixo:

- Caixa de engrenagens (transmissão): Também conhecida como caixa de multiplicação, é responsável por aumentar a rotação proveniente do rotor.

Figura 1.4 - Turbina Eólica



Fonte: Revista AdNormas[3] adaptado

Conforme (RIBRANT, 2006) as TE têm o tempo de inatividade entre 0.595% e 2.705% do tempo no período de um ano, devido a manutenções preventivas ou corretivas, isso significa até 237 horas sem geração. Conforme (HYERS, R. W. et al., 2006) 60% das falhas nas TE envolvem o sistema de controle elétrico, caixa de engrenagens, sistema de guinada (*Yaw system*), gerador elétrico, conexão com a rede ou pás.

As falhas devido aos desequilíbrios – por exemplo nas pás - formam a maior parte de todas as avarias (RIBRANT, 2006), e são geradas devido aos erros de construção ou

[3] Disponível em: https://fspanero.files.wordpress.com/2009/12/pca.jpg?w=450&h=218 Acesso em: 20 nov. 2019

manufatura, congelamento, degradação do tempo, desgaste ou fadiga. Uma pá com diferença de massa em relação as demais pode gerar rachaduras e quebras (SALCH, S. A., 2011), resultando em um colapso da TE.

A assimetria aerodinâmica – falha no sistema de passo (*Pitch system*) - contribui para 15% do total de falhas e 20% do tempo de inatividade (GAYO, J. B., 2011) (WILKINSON, M. et al., 2010). Essa condição anormal de funcionamento ocorre devido a erros no mecanismo de controle e cisalhamento do vento (uma variação rápida de corrente no vento).

Logo, devido ao grande impacto no funcionamento das TE, essas duas falhas serão aprofundadas ao longo desse trabalho.

Conforme (YANG, WENXIAN et al., 2014) a observação do funcionamento de TEs pela maioria dos métodos tradicionais, como as análises: com contador de partículas, de qualidade do óleo das caixas de engrenagem, de vibração torcional, ultrassônica, com termopares (análise da temperatura), via método de pulsos de choque e SCADA, não conseguem prever a falha em consequência do desequilíbrio de massa nas pás, por isso, não são relevantes para esse estudo. As análises: do torque do eixo, de termografia, de emissões acústicas e de vibrações acústicas, que conseguem identificar condições anormais nas pás. Seus custos variam conforme a acurácia da medição, resolução, ambiente da TE, porém, todas exigem o uso ou instalação de sensores extras para seus diagnósticos. Fato que exige investimento, aumenta custo de projeto, diminui a confiabilidade e aumenta a manutenção.

Até pouco tempo, uma alternativa era a análise de assinatura de corrente elétrica (AACE) do gerador, que (BENBOUZID, M. E. H., 2000) e (BENBOUZID, M., KLIMAN, G., 2003) forneceram boas revisões. Hoje, as novas técnicas de aprendizado de máquina estão sendo utilizadas para identificação de padrões e tomada de decisão a partir das AACE (NANDI, S.; TOLIYAT, H. A.; LI, X., 2005), permitindo a detecção de falhas sem custos adicionais de instrumentação e de forma não invasiva.

1.2 TRABALHOS RELACIONADOS

As novas técnicas valendo-se do Aprendizado de Máquina, para identificação de padrões, já veem sendo utilizadas na detecção de falha em turbinas eólicas. Em (WENYI, L. et al., 2013), (JIANWU, Z. et al., 2013) e (JUN-HYUN, S.; YUN-SEONG, L; JIN-O, K, 2014) aplica-se Máquina de vetores de suporte (SVM) com métodos matemáticos e demonstra-se resultados que essas metodologias detectam falhas nos aerogeradores com sucesso. No trabalho

de (RUI, Z. et al., 2016) emprega-se uma variação de SVM para detectar falhas elétricas em turbinas eólicas e obtém-se resultados promissores onde pode-se prever o gatilho da falha com 18 horas de antecedência. O SVM também é comparado com outras técnicas, (SANTOS, P. et al., 2015) mostra que o SVM pode obter melhores resultados de precisão, tempo de treinamento e tempo de ajuste superior a redes neurais artificiais para detecção de falhas mecânicas em turbinas eólicas. A combinação do SVM com turbinas eólicas gerou diversos trabalhos, alguns com restrições específicas como (AN, X. et al., 2011) no diagnóstico de falha em rolamentos com método do quadrado mínimo do SVM, (LEAHY, K. et al., 2016) com aplicação do SVM nos dados de TE para previsão de manutenção, (HU, C. et al., 2016) detectando falha da caixa de engrenagens com SVM, (SAARI, J. et al., 2019) classificando as falhas de rolamento pertencentes a turbinas eólicas com apenas uma classe no SVM. A otimização de parâmetros do SVM, conforme (CHUNHUA, Z.; HAIJING, D.; XIANYOU, Z., 2015), é feita para a otimização do diagnóstico de falhas em caixa de engrenagens de aerogeradores, ou de acordo com (LU, D.; QIAO, W., 2013) na extração de características das amostras para inserção no SVM.

Existem outros diversos trabalhos com o mesmo objetivo geral, porém, metodologias diferentes. Entre eles (VIDAL, Y. et al; 2015) que estuda a detecção de falhas em turbinas eólicas de 4.8 MW valendo-se de um algoritmo baseado em uma estrutura estendida e reconfigurada do filtro de Kalman. O artigo conclui que a aplicação de um sistema de detecção e diagnóstico de falhas aumenta a confiabilidade dos aerogeradores, esse monitoramento possibilita a melhora de performance, minimiza cargas mecânicas e aumenta eficiência.

Tendo em consideração o objetivo de diminuir os custos e tempo de reparo, melhorar funcionamento e aumentar tempo de duração de um gerador eólico (BERTRAND K., 2014) cria um modelo estocástico com o propósito de identificar a opção mais econômica a curto, médio e longo prazo. Os dados são referentes a pequenos geradores (600kW) e a solução não contempla nenhum tipo de algoritmo de inteligência artificial ou aprendizagem de máquina. (BERTRAND K., 2014) conclui que para apenas uma turbina eólica a manutenção corretiva é a que apresenta maior custo/benefício, porém, o uso de sistemas de monitoramento online se torna mais benéfico com o aumento do custo da eletricidade, aumento da capacidade da turbina e quão mais remoto estiver o aerogerador.

A utilização de rede neurais pode se dar conforme (BANGALORE, P.; 2015) para detecção de falhas em componentes de um pequeno gerador, onde são treinadas com dados de um ano inteiro de funcionamento de aerogeradores. O autor conclui que o método detecta

anormalidades semanas antes de acontecerem as falhas, possibilitando planejamento para a manutenção.

Peng em (PENG, Q.; XIANDONG, M.; PHILIP, C.; 2017) utiliza camadas ocultas de neurónios para o treino do aprendizado de máquina. Conclui-se que as novas técnicas de aprendizado de máquina melhoram em tempo de treino e aumento da eficiência em identificação de falhas.

Também é necessário avaliar-se estudos que apesar de não terem o mesmo objetivo geral do trabalho atual, utilizam-se de métodos e metodologias que serão aplicadas nesse trabalho. Trabalhos como (ROSSETTI, D.; 2016) trazem à tona a melhora dos resultados de classificação do SVM quando combinado com análise de componentes principais (PCA), reafirmando a eficácia da metodologia. Bons resultados com a aplicação da Decomposição em modo empírico (EMD) juntamente com a PCA e SVM também é reforçada por (LAHMIRI S. 2012) e (BEIBEI M.; 2017) em detecção de patologias de retina e diagnóstico de falhas em inversores de três níveis, respectivamente.

1.3 OBJETIVOS

Neste capítulo serão apresentados os objetivos que o autor pretende atingir ao final do trabalho, tão importante quanto a discussão dos problemas é a quantificação de soluções em metas palpáveis e mensuráveis para uma métrica de êxito do estudo.

1.3.1 Objetivo geral

O corrente trabalho tem como finalidade propor uma melhora no diagnóstico e classificação de condições anormais de funcionamento de TE. Busca-se o objetivo comparando (MALIK H., 2016) e (KANDUKURI, S.; 2019), propondo melhoras e adaptações nos processos.

1.3.2 Objetivos específicos

Melhorar processos e acurácia dos algoritmos de AM dos artigos em comparação, utilizando-se de ferramentas similares aos artigos originais:

- Aplicar a análise de componentes principais (PCA) para extração de dados mais relevantes da turbina eólica
- Adicionar o método de análise discriminante linear (LDA) sintetizando amostras expressivas para o classificador
- Utilizar pesquisa em grade (Grid Search) para otimizar busca dos parâmetros ótimos do classificador
- Empregar o classificador máquina de vetor de suporte (SVM)

- Aplicar a análise com módulo do vetor de Park estendido (MVPE) e *FFT* para extração de componentes mais significativos da turbina eólica
- Utilizar pesquisa em grade (Grid Search) para otimizar busca dos parâmetros ótimos do classificador
- Empregar o classificador máquina de vetor de suporte (SVM)
- Transformar para apenas um estágio o algoritmo original

1.4 ESCOPO DO TRABALHO

Esse documento apresenta o desenvolvimento das soluções propostas pelo autor para o problema descrito. Inicialmente este trabalho expõe um capítulo introdutório com informações gerais sobre a área abordada e definição do problema. O capítulo 2 compreende a revisão da literatura afim de mostrar abordagem de outros autores, fundamentar os conhecimentos para futuras elucidações e estabelecer uma compreensão preambular. O capítulo 3 apresenta as soluções propostas, subdividas entre os materiais que foram utilizados, bem como os métodos adotados para resolver o problema envolvido. No capítulo 4 são mostrados os resultados obtidos, sendo discutidos e analisados os valores encontrados. No capítulo 5 são apresentadas as conclusões do trabalho.

2. REVISÃO BIBLIOGRÁFICA

Esta seção é dedicada à exposição dos estudos necessários para a compreensão teórica, conceitos técnicos envolvidos no projeto e justificação das escolhas executadas ao longo do trabalho. Consiste na elucidação dos conceitos básicos para entendimento das soluções propostas são revisados.

2.1 TÉCNICAS DE PRÉ-PROCESSAMENTO

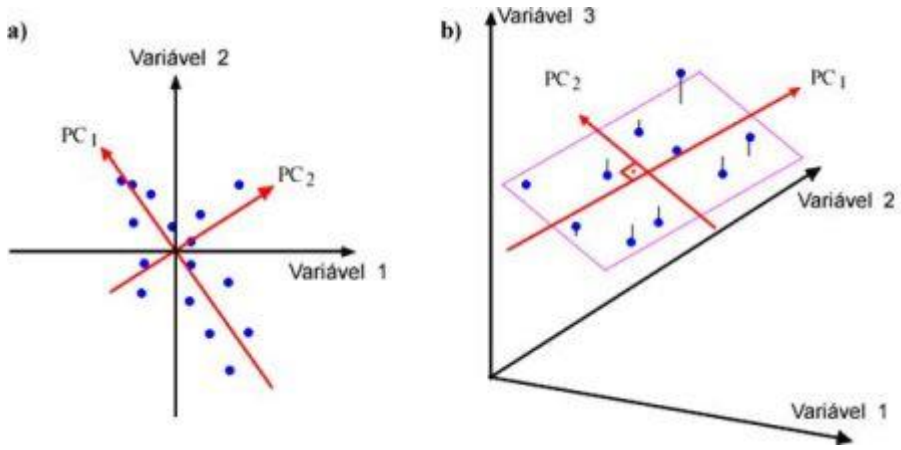
Nesse trabalho utiliza-se um vasto banco de dados de simulações com um grande número de variáveis, as simulações somam mais de 40Gb. Assim, é favorável considerar a aplicação de uma ou mais técnicas que reduzam a quantidade de dados mantendo as informações relevantes, com objetivo de diminuir o custo computacional mantendo a qualidade do classificador.

2.1.1 Análise De Componentes Principais

A Análise de Componentes Principais, do inglês *Principal component analysis* (PCA), é um procedimento matemático para análise exploratória de dados, os resultados têm como objetivo transformar um conjunto de valores variáveis linearmente correlacionados num conjunto linearmente não correlacionado (JOLLIFFE, I.; 2011). A redução de dimensão é umas das abordagens mais populares para remover redundância e irrelevância dos dados (TANG, J.; ALELYANI, S.; LIU, H., 2014), como ruídos.

Assim, revela-se as componentes que tem a maior variância (distância de valores do valor médio do vetor) possível, ou seja, responsável pelo máximo de variabilidade nos dados, como vemos na Figura 2.1, onde quando os dados são projetados sobre a componente principal 1 - Indicada pela flecha vermelha de nome PC1- mostram a maior variância possível. As outras componentes são sempre ortogonais a primeira e podemos observa-las – indicadas como PC1 e PC2 – na Figura 2.1:

Figura 2.1 - PCA aplicado a dados



Fonte: FSPANERO[4]

Conforme (PEARSON, K.; 1901) e (PEDREGOSA et al., 2011) o número de componentes principais após aplicação da PCA é sempre menor ou igual ao número de variáveis originais, e também, menor ou igual ao valor de amostras ou características. Devido a suas propriedades o PCA será utilizado para diminuição de dados, por conseguinte espera-se uma diminuição no tempo de processamento mantendo-se as informações relevantes. Para o entendimento sobre PCA necessita-se da compreensão de álgebra linear e estatística, conforme vemos nos passos para a implementação abaixo a primeira etapa consiste na normalização das variáveis:

$$X_{\text{médio}} = \frac{\sum_{i=1}^{n} x_i}{n} \quad (1)$$

Sendo:

$\bar{x}$ – Correspondente ao valor médio do conjunto de dados

s – Desvio padrão da amostra

Damos sequencia no cálculo considerando o vetor **x** sendo o vetor já normalizado. O PCA consiste fundamentalmente na rotação de eixos, sempre buscando alinhamento com a máxima variação dos dados.

Sendo assim descrevemos:

[4] Disponível em: https://galaxydatatech.com/wp-content/uploads/2018/07/PCA-Principal-Component-Analysis.png. Acesso em: 8 ago. 2019

$$\varepsilon = R\ \mathrm{x_i} \tag{2}$$

Sendo:

R – Matriz de rotação

ε – Novas coordenadas de referência

$\mathrm{x_i}$ – Vetor normalizado que representa as coordenadas iniciais

Para maximizar a variância nas novas coordenadas de referência, obtém-se:

$$\varepsilon_1 = \arg[\max[var(\varepsilon)]] = \arg[\max[var(r_1\ X)]] \tag{3}$$

Sendo:

r_1 – Vetor da matriz de rotação que gera maior variação

Conforme (JOHNSON; WICHERN et al., 2002) pode-se descrever a equação da variância de uma combinação linear como:

$$var(r_1 X) = r_1 \sum {r_1}^T \tag{4}$$

Utilizando multiplicador de Lagrange para o problema de otimização e realizando procedimentos matemáticos chega-se a um problema de autovalor e autovetor:

$$\max[var(r_1 X)] = \alpha \tag{5}$$

Sendo:

α – Multiplicador de Lagrange

Isto é, o maior autovalor da matriz de covariância corresponde a máxima variância, que refere-se a um autovetor da matriz de rotação. Após obter-se o primeiro autovalor e autovetor correspondente a maior variância, a busca das demais componentes principais é feita de forma análoga, com a adição de uma regra, que o próximo vetor seja ortogonal ao anterior.

Conforme desenvolvimento acima, são auferidas as covariâncias entre as variáveis, cálculo descrito por:

$$cov(x,y) = \frac{\sum_{i=1}^{n}(x_i - x_{médio}) * (y_i - y_{médio})}{n-1} \tag{6}$$

$$cov(x,z) = \frac{\sum_{i=1}^{n}(x_i - x_{médio}) * (z_i - z_{médio})}{n-1} \tag{7}$$

Sendo:

$cov(x, y)$ – Covariância entre os vetores x e y

$cov(x, z)$ – Covariância entre os vetores x e z

Monta-se então a matriz de covariância para um caso qualquer de dimensão $\Re^3$ da seguinte forma:

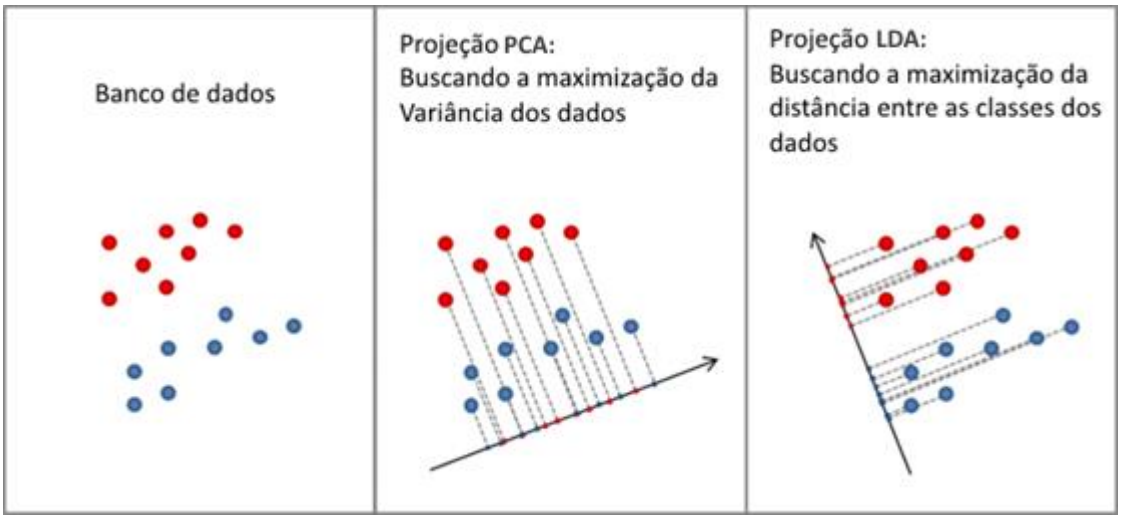
$$C = \begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{bmatrix} \quad (8)$$

Obtém-se os autovetores da matriz de covariância, onde os vetores mais significantes correspondem ao autovalor com maior magnitude (SMITH, L.; 2002).

2.1.2 Análise De Discriminante Linear

Análise discriminante linear, do inglês *Linear discriminant analysis* (LDA), é uma generalização do discriminante linear de Fisher, empregada nas mais diversas áreas que envolvem dados, desde estatística, reconhecimento de padrões e imagens (YE;JANARDAN; LI, 2005). É utilizada como um classificador linear ou, mais comumente, para redução de dimensionalidade antes da classificação (WELLING M., 2005) e seus resultados se dão conforme Figura 2.2:

Figura 2.2 - Comparação entre PCA e LDA



Fonte: Id Tools[5] (Adaptado)

De acordo com a Figura 2.2 nota-se a diferença entre a aplicação de duas técnicas estatísticas. Com a LDA tem-se a maximização de distância entre diferentes classes e minimização da variância entre uma mesma classe. Consequência disso são melhores resultados com algoritmos de aprendizado de máquina (FERNANDES S., 2013) em comparação apenas com aplicação de PCA (BOUZALMAT, A. 2014.

A ideia proposta pela LDA consiste em um princípio semelhante a PCA, porém, para a Análise Discriminante Linear de Fisher busca-se uma matriz de rotação $\boldsymbol{r_1}$ que irá maximizar a distância de dispersão entre diferentes classes e, ao mesmo tempo, minimizar a distância de dispersão na mesma classe. Logo:

$$\begin{cases} \max[disp(r_1(A_1 - A_2))] \\ \min[disp(r_1(A_1, A_2))] \end{cases} \tag{9}$$

Sendo:

A_1– Conjunto de amostras pertencente a classe 1

A_2– Conjunto de amostras pertencente a classe 2

$$\begin{cases} disp(r_1(A_1 - A_2))] = r_1^T S_B r_1 \\ disp(r_1(A_1, A_2))] = r_1^T S_W r_1 \end{cases} \tag{10}$$

Onde:

$$S_B = \sum_j (\mu_e - \bar{x}) * (\mu_e - \bar{x})^T \tag{11}$$

$$S_W = \sum_e \sum_i^e (x_i - \mu_e) * (x_i - \mu_e)^T \tag{12}$$

Sendo:

μ_e – Média de toda matriz de uma classe

$\bar{x}$ – Média total das amostras

x_i – Amostras em cada classe

[5] Disponível em: https://www.idtools.com.au/wp/wp-content/uploads/2018/11/PCAvsLDA-1024x467.png. Acesso em: 8 ago. 2019

Sabendo que S_B representa a matriz de dispersão entre as classes, logo, deve-se maximizar seu valor. Enquanto S_W representa a matriz de dispersão entre as amostras de uma mesma classe, logo deve-se minimizar seu valor.

Para resolver esse problema de otimização, se faz uso de manipulações matemáticas estruturadas em um problema de otimização de Lagrange, representada por:

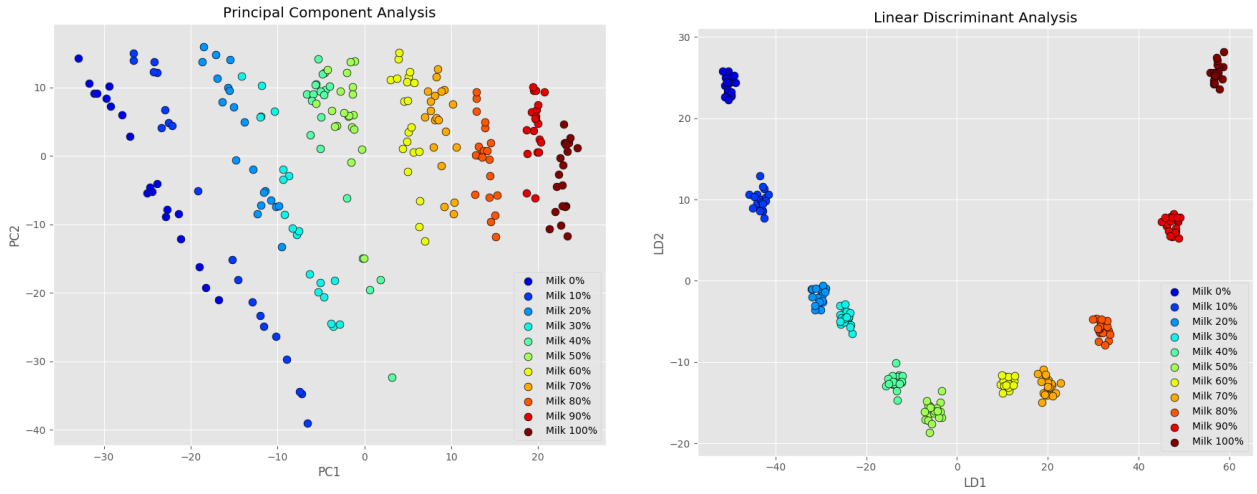
$$L_p = -\frac{1}{2} r_1{}^T S_B r_1 + \frac{1}{2}\lambda(r_1{}^T S_W r_1 - 1) \tag{13}$$

Conforme condições de Lagrange, a derivada da Equação (11) em relação a r_1 deve ser zero, obtemos:

$$S_B S_W{}^{-1} r_1 = \lambda r_1 \tag{14}$$

Chega-se assim aos dados finais conforme (BALAKRISHNAMA S.;1998), percebe-se que se trata de um problema de autovalores e autovetores. Abaixo vemos que o LDA pode melhor representar os dados por classe como na Figura 2.3:

Figura 2.3 - Comparação entre PCA e LDA



Fonte: Id Tools[6]

[6] Disponível em: https://www.idtools.com.au/classification-nir-spectra-linear-discriminant-analysis-python/. Acesso em: 8 ago. 2019

2.1.3 Módulo Dos Vetores De Park's Estendido

O Módulo dos vetores de Parks's Estendido (MVPE) é uma transformação algébrica que rotaciona o *frame* de referência de um vetor com três elementos afim de simplificar análise. A função principal transforma as variáveis de corrente de fase (i_a, i_b, i_c) em duas componentes (CRUZ, S., CARDOSO, A., 2000) (i_d, i_q) sem a perda de informações pela Equação (30):

$$i_d = \sqrt{\frac{2}{3}} i_a - \sqrt{\frac{1}{6}} i_b - \sqrt{\frac{1}{6}} i_c \tag{15}$$

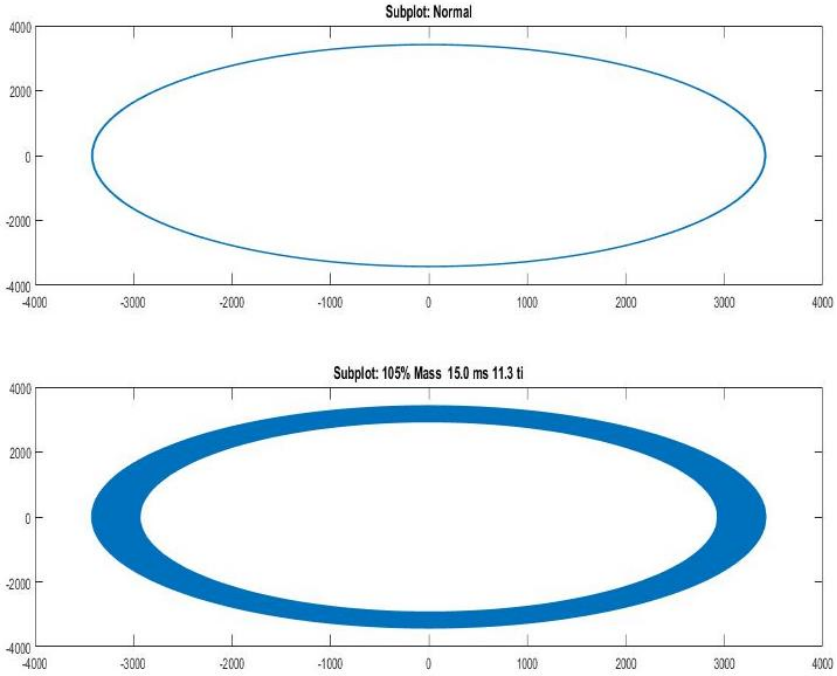
$$i_q = \sqrt{\frac{1}{2}} i_b - \sqrt{\frac{1}{2}} i_c \tag{16}$$

A representação de i_d x i_q na forma de curva de Lissajou's, sendo circular e centralizada na origem das coordenadas com o sistema de equações i_d x i_q, apresenta duas situações, onde, conforme (SILVA, 2008) faz com que componentes características de avarias apareçam com destaque.

Quando as correntes são provenientes de motores elétricos em bom e normal funcionamento, exibem-se um módulo constante, representado pela plotagem superior na Figura 2.4.

Sob condições anormais, como por exemplo desequilíbrio no rotor do gerador, nota-se a presença de bandas laterais a frequência fundamental, evidentes na plotagem inferior da Figura 2.4 abaixo:

Figura 2.4 - id x iq



Fonte: Autor

Para análises futuras utilizaremos o módulo da soma dos vetores i_d e i_q, para isso, aplica-se a transformação direta em quadratura zero conforme Equação (33):

$$i_p = \left| i_d + j i_q \right| \tag{17}$$

Conforme (ZAREI, J., POSHTAN, J, 2006) somente o uso da transformação de Park's não é bom o suficiente para o diagnóstico de falhas, primeiramente por que não é obvio que o modelo/padrão é único para os diferentes tipos de falha, a classificação se torna difícil se considerarmos problemas práticos e ruídos. Por isso, utiliza-se o espectro do módulo do vetor de Park's além da transformada de Fourier em i_p.

2.1.4 Transformada Rápida De Fourier

Derivada da Transformada discreta de Fourier a Transformada rápida de Fourier - do inglês *Fast Fourier Transform*, ou FFT - é uma análise que converte um sinal de seu domínio original, como o domínio do tempo, para sua representação no domínio de frequência e vice-

versa. Assim, qualquer função periódica de tempo pode ser representada por uma Série de Fourier como uma soma infinita de termos em seno e cosseno (RAO, 2009) tornando-se um método muito eficiente que reordena os cálculos dos coeficientes de uma Transformada Discreta de Fourier (DFT).

É chamada de transformada rápida de Fourier devido ao seu algoritmo de computação muito rápido (COOLEY, J., 1965) sua distinção da FFT se dá por realizar uma avaliação da DFT com o menor esforço computacional, ao invés de realizar o cálculo da DFT diretamente pela definição (HANLY, 2016) por isso a transformada de Fourier pode ser descrita pela equação (30):

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\, W_N^{ux} \tag{18}$$

Sendo:

N – Número total de amostras

u – Número atual da amostra no plano do tempo

x – Número atual da amostra no plano da frequência

$f(x)$ – Função no domínio do tempo

$F(u)$ – Função no domínio da frequência

Onde:

$$W_N^{ux} = e^{\frac{-j2\pi ux}{N}} \tag{19}$$

Assume-se que que o número de amostras tenha base 2, conforme (VAN LOAN, C., 1992), portanto $N = 2^n$ onde n é um inteiro positivo. N pode ser escrito como $N = 2M$ onde M é um inteiro positivo. Separa-se em duas partes os coeficientes pares e ímpares, e se tem a Equação (32):

$$F(u) = \frac{1}{2}\left[\frac{1}{M}\sum_{x=0}^{M-1} f(2x)\, W_M^{u(2x)} + \frac{1}{M}\sum_{x=0}^{M-1} f(2x+1)\, W_M^{u(2x+1)}\right] \tag{20}$$

Chegando assim em uma recombinação final que nos fornece a Equação (33):

$$F(u+M) = F_{par}(u) - F_{impar}(u) W_{2M}^{u} \tag{21}$$

Para cálculos de FFT usaremos a biblioteca "*Numpy*" na função "*fftn*", ou seja, cálculos computacionais de n-dimensões da transformada rápida de Fourier.

2.1.5 Decomposição em modo empírico

Do inglês *Empirical Mode Decomposition* (EMD), a decomposição em modo empírico é a desmembração de um sinal em vários componentes, forma-se um número pequeno de componentes, quase ortogonais com o sinal original, descritos como funções do modo intrínseco (IMF). O EMD tem o mesmo objetivo de Fourier ou Wavelets, porém, não faz suposições sobre a composição do sinal, usa-se a interpolação *spline* entre máximos e mínimos para traçar sucessivamente as IMFs. Cada IMF será um único oscilador periódico, mas não poderá ser previsto antes de ser empiricamente observado a partir do sinal (RILLING G., FLANDRIN P., GONCALVES P., 2003).

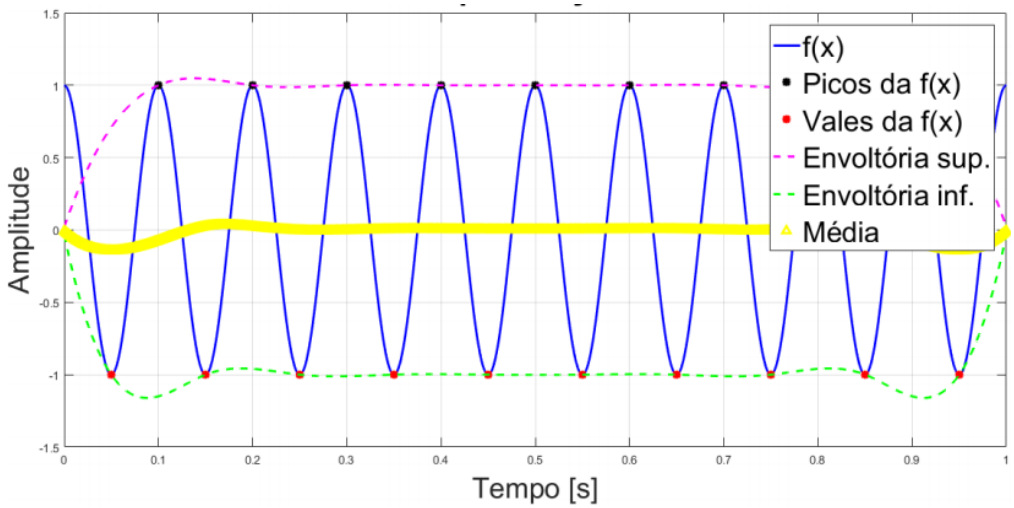
Podemos descrever matematicamente a EMD conforme (MALIK, H.; 2016) onde no processo de aplicação de EMD e seleção das IMFS, há duas condições que devem ser satisfeitas na derivação do IMF:

- O número de extremos e o número de cruzamentos de zero devem ser iguais ou no máximo diferentes por um;
- O valor médio do "envelope" definido pelos mínimos e máximos locais deve ser zero (ou próximo de zero).

Cumprindo os requisitos podemos aplicar EEMD nos dados conforme os seguintes passos (GACI, S., 2016):

Primeiramente determinar os extremos de máximo e mínimo do sinal, conectar os pontos extremos por interpolação *spline*, envolvendo os valores de alta e de baixa determinando suas médias locais e finalmente determinar a diferença entre essas médias de alta e baixa.

Figura 2.5 - Pontos a serem determinados para cálculo de EMD



Fonte: (FERREIRA, L.; PORSANI, M.; DA SILVA, M.; 2010)

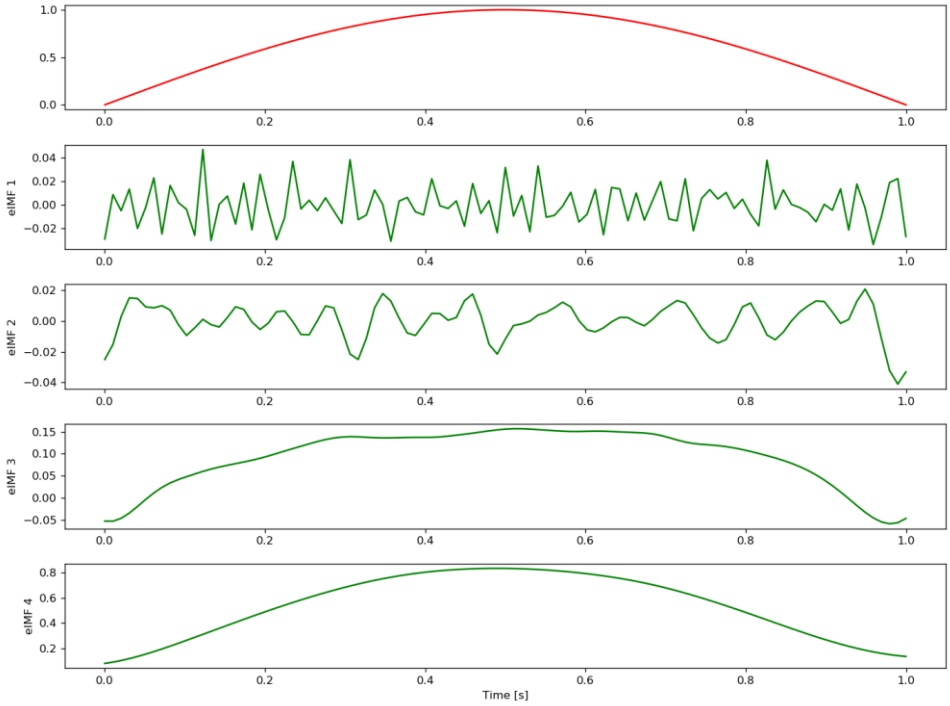
$$X(t) = \sum_{m=1}^{M-1} IMF_m^n(t) + r_M^n(t) \tag{22}$$

Sendo:

$r(t)$ – Residual

Ou seja, a soma das IMFs mais o residual é igual ao sinal original, conforme:

Figura 2.6 - EMD aplicada a função Seno



Fonte: Autor

2.2 MÁQUINA DE VETORES DE SUPORTE

Aprendizado de máquina é o estudo científico de algoritmos e modelos estatísticos que computadores usam para realizar uma tarefa específica sem usar instruções explícitas (PHIL,2013), confiando em padrões e inferência. É visto como um subconjunto da inteligência artificial (KOZA; J.; 1996) e está intimamente relacionado à estatística computacional, que se concentra em fazer previsões (FRIEDMAN, J.; 1998) por isso também é conhecido como análise preditiva. Considerando que a ação de aprender deve-se ao fato de vir a ter uma melhor compreensão de algo pela experiência, conclui-se que há várias maneiras desse processo ocorrer. Para tanto, pode ser categorizado e (MARSLAN, S.; 2014) classifica os diferentes métodos de aprendizado conforme:

- Supervisionado: Um conjunto de dados e seus rótulos são fornecidos, de forma a dar exemplos dos tipos de classe.
- Não supervisionado: Não há presença de um agente externo, logo, as respostas não são fornecidas, mas ao invés disso, o algoritmo tenta encontrar padrões e

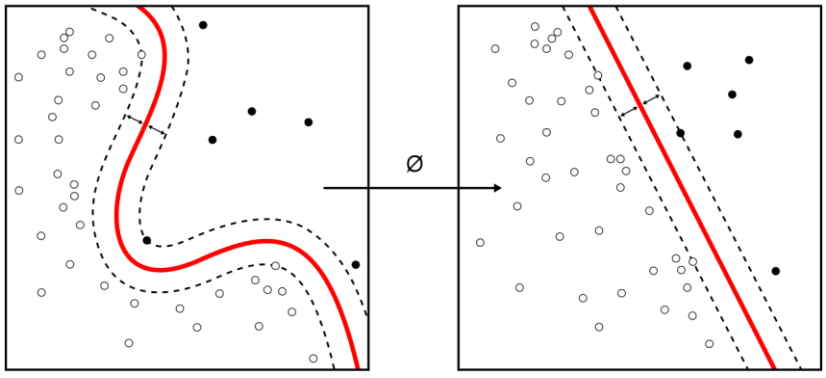
semelhanças entre as entradas de forma a agrupá-las em conjunto e categorizá-las.

Do inglês, *Support Vector Machines* (SVM), em tradução livre “Máquina de Vetores de Suporte”, é uma das técnicas de aprendizado de máquina (AM) amplamente utilizada para reconhecimento de padrões, apesar de novas teorias, ainda é muito utilizada devido a sua simplicidade de configuração e resultados ocasionalmente superiores a outras técnicas mais robustas de aprendizado de máquina, como por exemplo redes neurais artificiais (LORENA; CARVALHO, 2007).

Conforme (CHANG; LIN, 2011) SVM é um método popular de Aprendizado de Máquina e foi escolhido para aplicação nesse trabalho devido a sua simplicidade, com a convicção de que aliada a métodos estatísticos de pré-processamento traria bons resultados de classificação, além de ser amplamente documentada em pesquisas similares como (MALIK H., 2016), (KANDUKURI, S., 2019), (LAOUTI, N.; SHEIBAT-OTHMAN, N.; OTHMAN, S., 2011) e (SANTOS, P., 2015) que também fazem detecção de falhas em TE com SVM.

SVM tenta encontrar uma linha de separação, mais comumente chamada de hiperplano, entre dados de duas classes. Essa linha busca maximizar a distância entre os pontos mais próximos em relação a cada uma das classes, como mostra a Figura 2.7:

Figura 2.7 - Função (em vermelho) gerada pela SVM de margens suaves

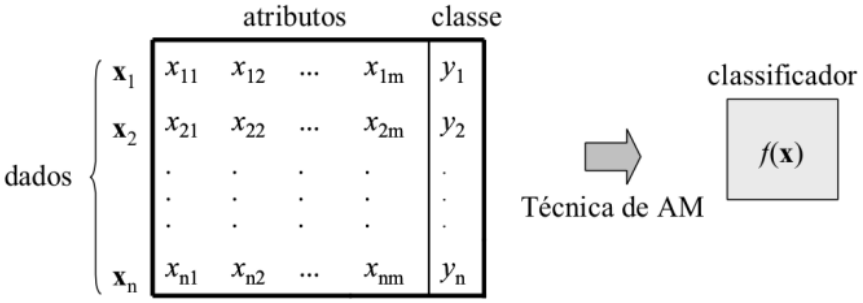


Fonte: Alisneaky, svg[7]

[7] Alisneaky, svg version by Zirguezi / CC BY-SA (https://creativecommons.org/licenses/by-sa/4.0). Acesso em: 16 jun. 2020

A SVM, por padrão e com treinamento supervisionado, tem como entrada um conjunto de dados e faz uma predição para cada entrada dada. Dado um conjunto de exemplos de treinamento, cada um contendo um marcador de uma das categorias possíveis, como mostrado na Figura 2.8, onde um algoritmo de treinamento da SVM constrói um classificador.

Figura 2.8 - Classificação em aprendizado supervisionado

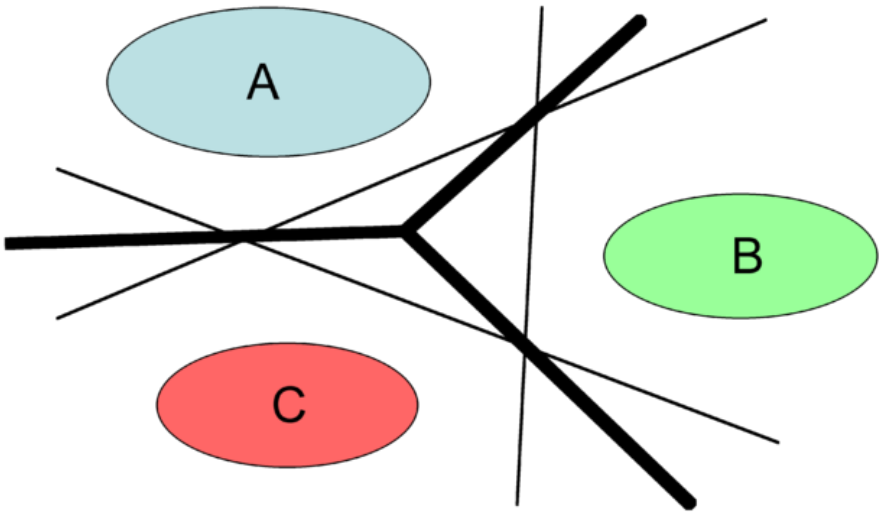


Fonte: (A. C. LORENA, 2007)

SVM é originalmente um classificador linear binário não probabilístico, mas com os avanços da tecnologia desenvolveu-se para problemas multiclasses (HSU; LIN, 2002). Existem duas opções para funcionamento da SVM multiclasses, '*one-versus-one*' ou '*one-versus-rest*', em tradução livre, "um-contra-um" e "um-contra-resto" respectivamente. Foi escolhido o método "um-contra-resto", pois, conforme (PEDREGOSA et al., 2011) o método "um-contra-um" é apenas interessante do ponto de vista teórico, raramente é usado na prática pois tem resultados de precisão pior e tem maior custo computacional.

A técnica "um-contra-resto" foi introduzida por (VAPNIK, V.; 1995) e até hoje é utilizada. Uma amostra só é classificada em alguma classe se for aceito pelo hiperplano dessa classe e se for rejeitado por todos as outras classificações, conforme a Figura 2.9 vemos a melhoria proposta por (VAPNIK, V.; 1998) com o uso de valores contínuos nas funções de decisão, destacada em negrito:

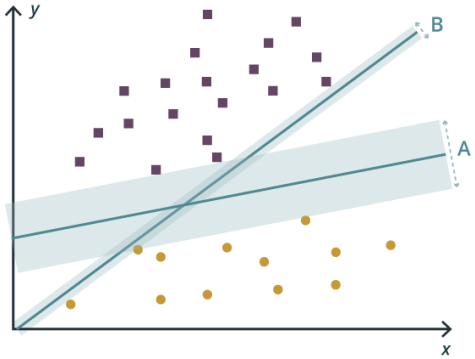
Figura 2.9 - Diagrama contínuo de "Um contra todos" na classificação de 3 classes



Fonte: (AISEN, B.; 2006)

Na imagem da Figura 2.10 temos duas hipóteses, de hiperplano de separação, que classificam corretamente os exemplos da amostra de treinamento, por isso, é necessário a escolha do classificador ótimo. Temos então a Teoria do Aprendizado Estático (TAE) que define equações matemáticas para ajudar na escolha entre por exemplo o classificador "A" ou "B".

Figura 2.10 - Classificações possíveis pela SVM de margens rígidas



Fonte: Harp SVM GitHub[8]

Para SVM lineares e com margens rígidas, sendo $S = (x_1, y_1), \dots, (x_m, y_m)$ um conjunto de dados de treinamento com x_i pertencente ao domínio dos números reais e y_i assumindo

[8] Disponível em:https://dsc-spidal.github.io/harp/docs/examples/svm/. Acesso em: 8 ago. 2019

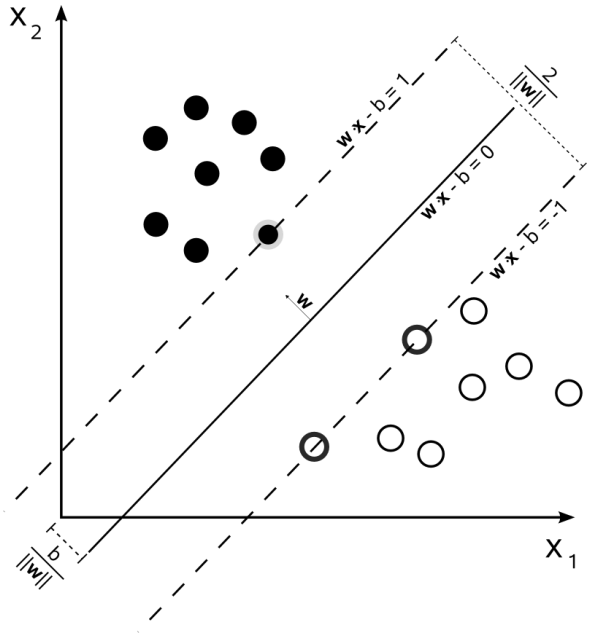
valores de -1, 0 ou 1. Esse conjunto é linearmente separável se existe um hiperplano (w, b) tal que $y_i = \text{f(x)} = \text{sign}(\text{w} \cdot \text{x} + \text{b})$ para todo i (SHALEV-SHWARTZ; BEN-DAVID, 2014). O classificador assume a forma:

$$\text{f(x)} = \text{sign}(\text{w} \cdot \text{x} + \text{b}) \tag{23}$$

Onde:

$$\text{sign(y)} \begin{cases} +1, \text{y} \geq 0 \\ \text{-1}, \text{y} < 0 \end{cases}$$

Figura 2.11 - Margens de separação

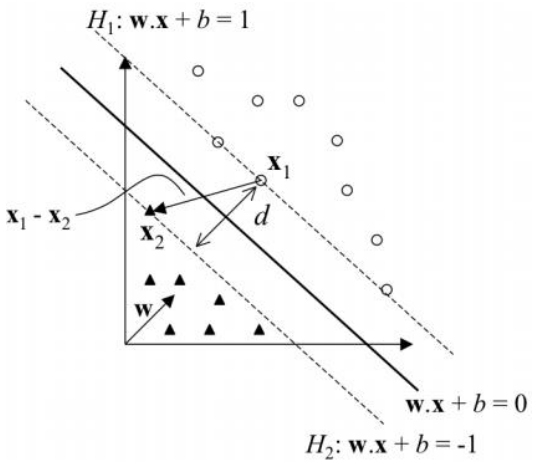


Fonte: Wikimedia[9]

Com a obtenção da Equação (23) sabemos a restrição entre os dados linearmente separáveis, faz-se necessário então o cálculo da expressão matemática que modela a distância d – que queremos maximizar – entre as margens.

[9] Disponível em: https://upload.wikimedia.org/wikipedia/commons/2/2a/Svm_max_sep_hyperplane_with _margin.png. Acesso em: 28 jan. 2020

Figura 2.12 - Distância entre margens



Fonte: (A. C. LORENA, 2007)

O vetor distância d na Figura 2.12 representa a distância entre dois vetores de suporte para o hiperplano de separação. Sabendo que $d = x_1 - x_2$ e fazendo as manipulações necessárias, conclui-se que para a maior margem do hiperplano, deve-se minimizar a Equação (24):

$$d = \frac{2}{\|w^*\|} \tag{24}$$

Minimizando a Equação (24) sob os padrões de uma função custo de Lagrange, conforme (SCHÖLKOPF et al., 2002), passa a ser representada por:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i(\{[w \cdot x_i] + b\} \cdot y_i - 1) \tag{25}$$

Sendo:

L – Lagrange

w - Vetor normal ao hiperplano escolhido

b - Distância do vetor normal do hiperplano a origem do sistema

α_i (i = 1...N) - Multiplicador de Lagrange

x_i – Atributo de número i

y_i – Classe referente ao atributo x_i

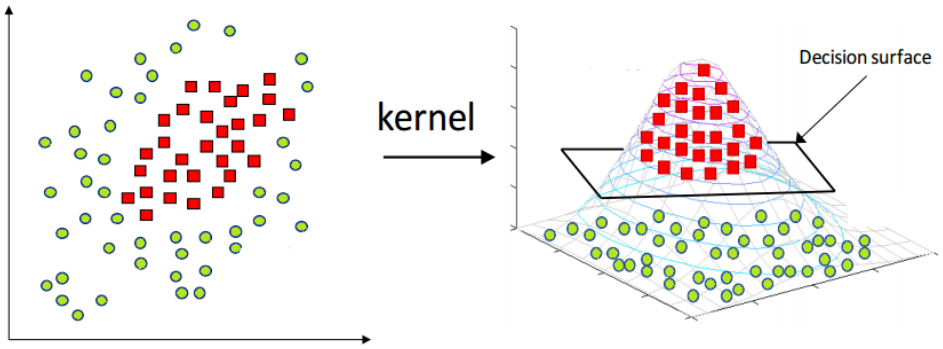
Com o resultado da Equação escolhe-se o classificador que gera um hiperplano de separação ótimo, conforme parâmetros de entrada. Conforme (V. N. VAPNIK, 1995) alguns princípios que devem ser seguidos para que a classificação seja feita com uma boa

generalização, promovendo uma alta taxa de acertos não só durante o treinamento, mas também durante os testes. Isso envolve a escolha de três principais parâmetros de entrada, e sua escolha correta evita excesso ou escassez de ajuste do hiperplano (B. KOMER, J. BERGSTRA, C. ELIASMITH, 2014).

2.2.1.1 Kernel

No algoritmo em *python*, utiliza-se a biblioteca *sklearn*, quando é utilizada a máquina de vetor de suporte, podemos definir o tipo de função *kernel* que aplica-se nos dados. O objetivo do *kernel* é melhorar a separação dos dados e, se necessário, um aumento de dimensões será feito. Como na Figura 2.13 onde a função *kernel* define o produto interno no espaço transformado, que mesmo após uma mudança de dimensão continua representado os dados iniciais.

Figura 2.13 - Exemplo categorização de dados por SVM



Fonte: Hackerearth[10]

Existem diversas funções *kernel*, cada uma modifica as características dos dados de forma a torná-lo separável, dentre elas podemos destacar as mais comuns:

Kernel da Função Linear:

$$K_{linear}(\mathrm{x},\mathrm{y}) = \sum_{i=1}^{d} x_i y_i \tag{26}$$

[10] Disponível em: https://www.bogotobogo.com/python/scikit-learn/images/svm2/. Acesso em: 8 ago. 2019

Kernel da Função Polinomial com grau "d":

$$K_{polinomial}(\mathrm{x},\mathrm{y}) = (1 + {x_i}^T y_i)^d \quad (27)$$

Kernel da Função sigmoide com parâmetros k e δ:

$$K_{sigmoid}(\mathrm{x},\mathrm{y}) = tanh(k{x_i}^T y_i - \delta)^d \quad (28)$$

Kernel da Função de Base Radial com parâmetro σ:

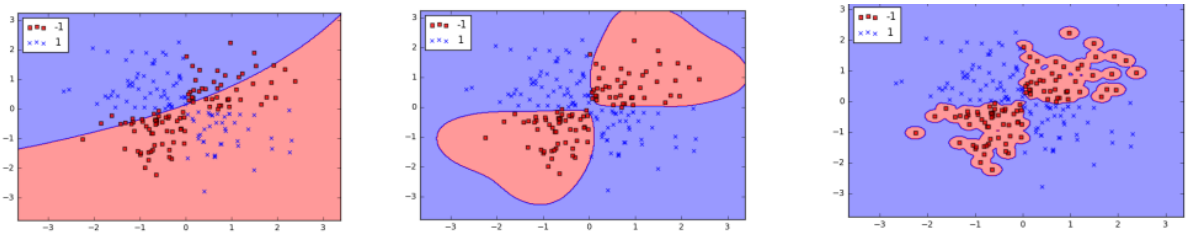
$$K_{RBF}(\mathrm{x},\mathrm{y}) = \exp(-\frac{(x_i - y_i)^2}{2\sigma^2}) \quad (29)$$

As funções *Kernel* acima serão incluídas nos testes de parâmetros afim de encontrar qual função melhor separa os dados.

2.2.1.2 Gamma

A variável *gamma* representa a generalização dos dados, quanto maior o valor de *gamma* menor será a generalização:

Figura 2.14 - Exemplo de aplicação de gamma nos dados



Fonte: Qinkkai's Blog[11]

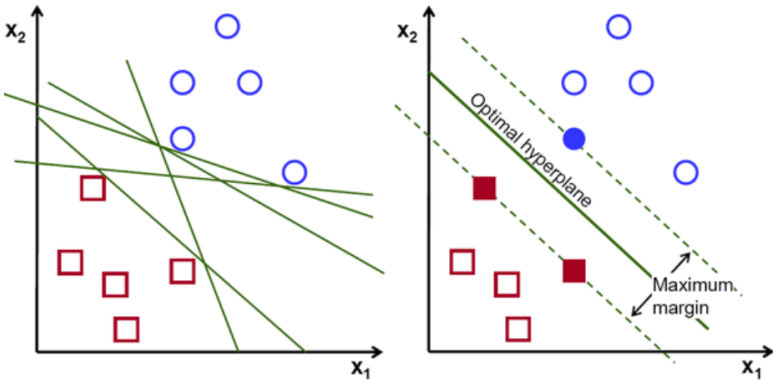
Como vemos na Figura 2.14 o gamma afeta diretamente na classificação entre os pontos vermelhos e azuis. O padrão do gamma é $\frac{1}{\text{Nºde características}}$.

[11] Disponível em: http://qingkaikong.blogspot.com/2016/12/machine-learning-8-support-vector.html. Acesso em: 8 ago. 2019

2.2.1.3 C

O parâmetro C afeta o tamanho da margem do hiperplano, ou seja, é uma variável de entrada do usuário para controle da distância ‘d’ da Equação 23 e Equação 24. Como observamos na Figura 2.15 a diferença da margem máxima de distância entre os hiperplanos plotados influencia diretamente na escolha do classificador.

Figura 2.15 - Escolha de hiperplanos



Fonte: Towards Data Science[12]

C é um parâmetro, que altera a troca de uma classificação correta de exemplos de treinamento por uma maximização da margem da função decisão.

Grandes valores de C determinam hiperplanos com a menor margem entre classes. Para pequenos valores de C o otimizador escolhe o hiperplano com a maior margem de separação entre as classes, mesmo se não for o classificador que teve mais categorizações corretas.

2.2.1.4 Divisão Treino/Teste

A Divisão Treino/Teste é a função responsável por dividir a matriz de dados em subconjuntos aleatórios de treino e teste mantendo sempre 70% do banco de dados para treino e 30% para testes afim de estabelecer uma confiança nos resultados. A função faz uma validação cruzada tornando a simulação mais confiável.

---

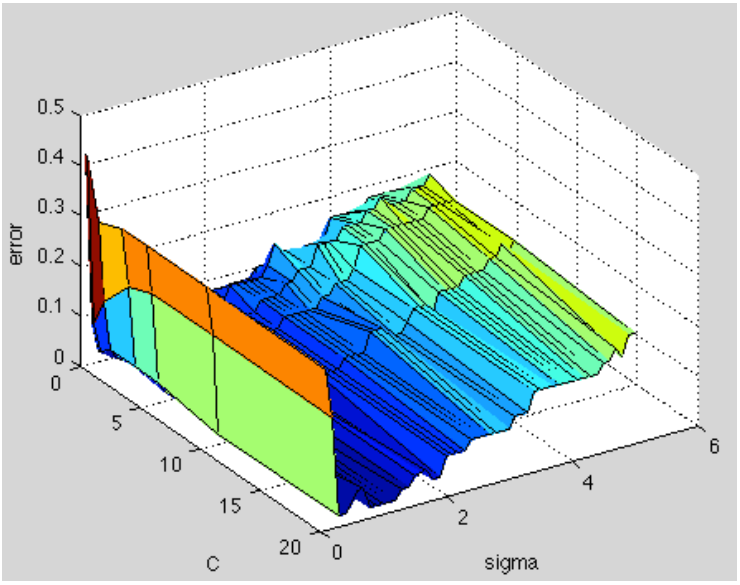
[12] Disponível em: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. Acesso em: 8 ago. 2019

Proveniente da biblioteca *sklearn*, a função de divisão dos dados foi aplicada em todos algoritmos apresentados no apêndice.

2.2.1.5 Pesquisa em grade (Grid Search)

A "Pesquisa em grade", do Inglês *Grid Search*, é um processo de exploração de dados para otimização das configurações de parâmetros a um dado modelo. Tem o propósito claro de encontrar os parâmetros ótimos, que sucedem nos melhores resultados de uma ou mais variáveis requisitadas. A Figura 2.16 exemplifica a correlação entre o erro e os parâmetros inseridos na SVM

Figura 2.16 - Exemplo de Grid Search para SVM



Fonte: Per Class[13]

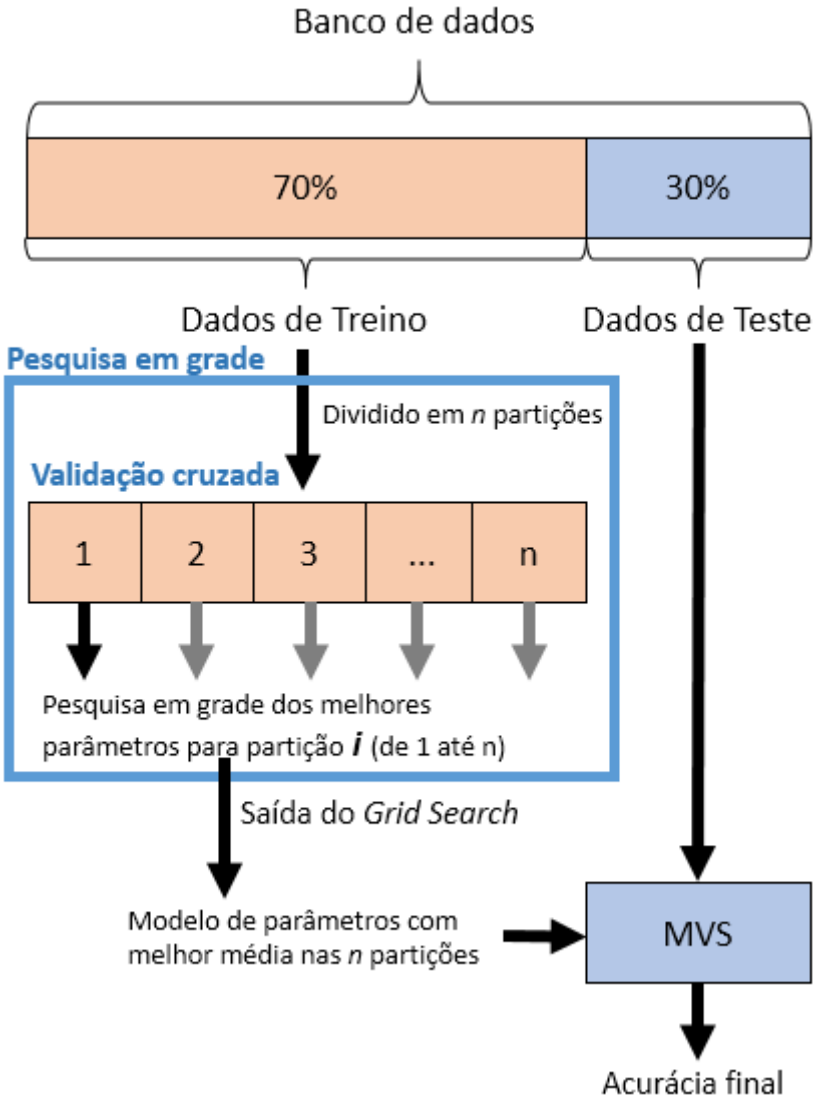
Esse trabalho busca a maximização da acurácia e pode ter um custo computacional grande (PEDREGOSA et al., 2011), pois irá construir equações para cada combinação de parâmetros.

O algoritmo consta nos apêndices, foi utilizada a função *GridSearch CV* e o fluxograma de funcionamento da função segue conforme a Figura 2.17:

---

[13] Disponível em: http://doc.perclass.com/perClass_Toolbox/guide/classifiers/svm.html. Acesso em: 8 ago. 2019

Figura 2.17 - Fluxograma do algoritmo de pesquisa em grade



Fonte: Autor

Os parâmetros obtidos da “Pesquisa em grade” são inseridos na SVM juntamente com os dados de teste. A “Acurácia final” é então obtida como indicador de validação dos parâmetros, pois os dados são independentes aos utilizados para treino do aprendizado de máquina. Baixa acurácia revelaria ajuste extremo (*overfit*) ou erro na escolha dos parâmetros.

2.2.1.6 Accuracy Score

Também da biblioteca *sklearn*, a função para cálculo da acurácia, do subconjunto de teste, faz uma comparação dos valores da previsão do SVM, com os rótulos da matriz verdade (matriz com as classes corretas das amostras), chegando assim a uma pontuação da precisão da classificação.

Outras métricas também foram adotadas com o intuito de melhorar a apresentação dos resultados obtidos, são elas:

$$\text{Acurácia} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{30}$$

Sendo:

TP – True Positive (Verdadeiro positivo)

TN – True Negative (Verdadeiro negativo)

FP – False Positive (Falso positivo)

FN – False Positive (Falso negativo)

Acurácia (Equação 30) é normalmente descrita em porcentagem, representando o número total de acertos dividido pelo número total de tentativas.

$$\text{Precisão} = \frac{\text{TP}}{\text{TP+FP}} \tag{31}$$

Precisão (Equação 31) pode ser definida como a média aritmética das precisões.

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{32}$$

Recall (Equação 32) é um indicador de casos relevantes que foram corretamente identificados, ou seja, refere-se a diagonal principal da matriz de confusão.

$$\text{F1-Score} = \frac{\text{2xRecallxPrecisão}}{\text{Recall+Precisão}} \tag{33}$$

A última métrica de avaliação dos algoritmos é a *F1-Score* (Equação 33), que é a média harmônica entre a precisão e o *recall*, e procura representar a qualidade do classificador até para casos onde o número de classes é desproporcional.

3. MATERIAIS E MÉTODOS

Neste capítulo explana-se a metodologia escolhida na execução do projeto, bem como os materiais, desde linguagens de programação, softwares, técnicas de pré-processamento e bibliotecas de algoritmo.

3.1 PYTHON

Lançada por Guido van Rossum em 1991. Atualmente possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos Python Software Foundation. Apesar de várias partes do python possuírem padrões e especificações formais, a linguagem como um todo não é formalmente especificada. A linguagem foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Prioriza a legibilidade do código sobre a velocidade ou expressividade, uma prova é a importância da indentação para o bom funcionamento do script, que só é compilado e executado quando está de acordo com as normas estabelecidas.

Para o presente projeto será utilizada a linguagem *Python*, pois combina uma sintaxe concisa e clara com os recursos poderosos de sua biblioteca padrão além dos módulos e frameworks desenvolvidos por terceiros. Hoje, ganhou notoriedade exponencial devido a participação da comunidade online, transformando-a em uma linguagem de construção comunitária. O python é uma linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipo dinâmica e forte. Foram utilizadas as seguintes bibliotecas: *Numpy*, *Pandas*, *Matplotlib*, *Sklearn* e *Scipy*.

3.2 DADOS DE SIMULAÇÃO

Sem ter à disposição dados reais de geradores eólicos faz-se necessário o uso de dados simulados, e é parte fundamental para a resolução do problema o entendimento dos dados. Como na Figura 3.1, de uma TE em alto mar, nesse trabalho utiliza-se aerogeradores que possuem 3 pás.

Figura 3.1 - Exemplo de um aerogerador utilizado nas simulações (com 3 pás)

Fonte: Sun Wind Energy[14] adaptado

As simulações foram executadas nas seguintes condições:

- Funcionamento com desequilíbrio de massa nas pás
- Funcionamento com desajuste de ângulo de passo das pás
- Condição normal de funcionamento

Para o funcionamento com desequilíbrio nas pás, simula-se uma das pás com massa diferente das restantes, representado em porcentagem com valores positivos ou negativos em relação a massa original.

Na performance com o desajuste de passo, simula-se uma das pás com ângulo de passo diferente das demais, a diferença é representada em graus (º) e com valores positivos. Apesar de existir um controlador de passo, suas ações são coletivas perante as pás, logo, não apresenta comportamentos corretivos em relação ao desequilíbrio unitário. A condição normal de funcionamento, como esperado, segue o comportamento ideal da TE.

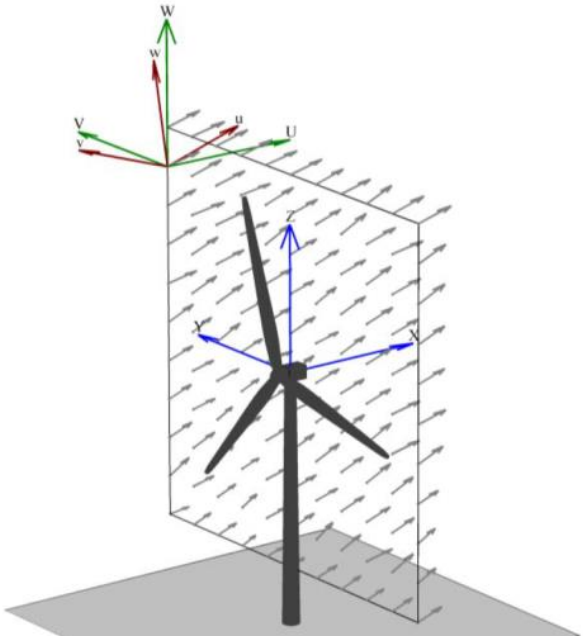
[14] Disponível em: https://www.sunwindenergy.com/sites/default/files/field/image/offshore_france-sgre_offshore_wind_turbine.jpg. Acesso em: 8 ago. 2019

3.2.1 Softwares de geração de dados de aerogeradores

Baseado na linguagem de programação Python criou-se uma interface para automatizar a conexão entre os programas e a troca de informações que o usuário insere para as diversas simulações.

O software da primeira etapa é o TurbSim, que é uma ferramenta de simulação de ventos estocásticos e turbulentos que tem como saída séries temporais das 3 componentes vetoriais da velocidade do vento, os parâmetros são os diferentes projetos e modelos das turbinas, escoamento do vento e turbulência, fatores que alteram a resposta das cargas mecânicas de aerogeradores.

Figura 3.2 - Malha Turbsim



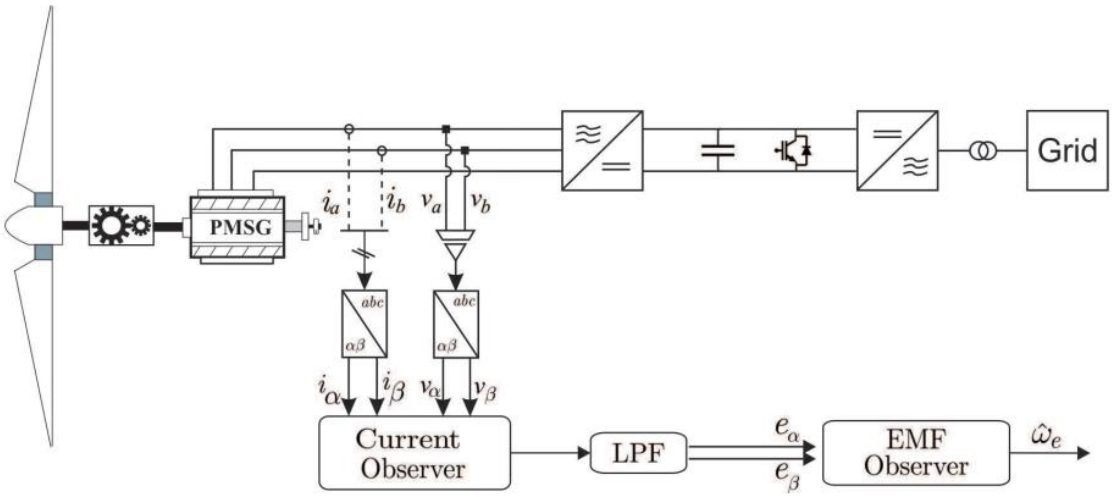
Fonte: (JONKMAN; K., 2006)

Após essa etapa insere-se os dados no FAST v8 aplicativo para simulação das respostas dinâmicas acopladas de uma turbina eólica, ou seja, permite a análise incluindo a aerodinâmica de controle, aerodinâmica de sistemas elétricos, aerodinâmica estruturais e hidrodinâmicas. Gerando assim dados de velocidade de rotação, cargas mecânicas e torque mecânico, variáveis que dão origem a um conjugado no rotor.

Assim, insere-se os dados obtidos até então no Simulink do Matlab, onde com aplicação em um gerador elétrico calcula-se as tensões e correntes do estator. Com dados da tensão e

corrente, determina-se o torque elétrico e a potência elétrica, que são realimentadas no FAST para uma melhor resposta de saída. O esquemático com o observador de corrente e de velocidade segue conforme Figura 3.3

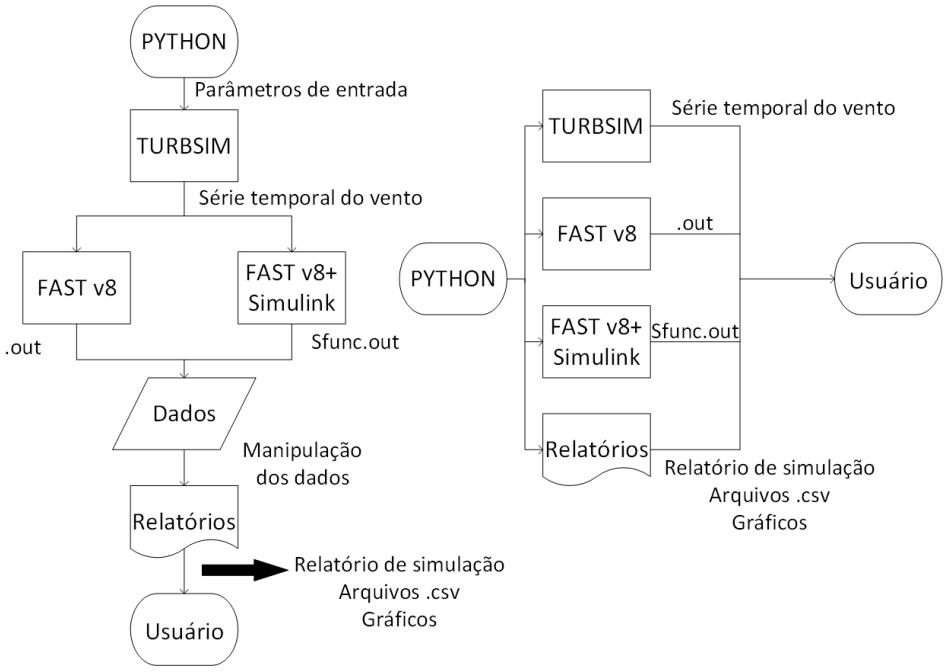
Figura 3.3 - Esquemático geração de dados



Fonte: (ROSA; L. D., 2019)

O fluxograma das etapas, para geração de sinal, é ilustrado na Figura 3.4:

Figura 3.4 - Framework da plataforma de geração de dados



Fonte: (ROSA; L. D., 2019)

Utiliza-se o conjunto de softwares relatados acima para gerar o banco de dados nas diversas condições de funcionamento do aerogerador, funcionando sempre em potência nominal e com os parâmetros de simulação da Tabela 3.1:

Tabela 3.1 - Dados do aerogerador simulado

Parâmetro	Valor
Potência Nominal	1,5 MW
Tipo de máquina elétrica	Gerador síncrono de imãs permanentes
Altura da nacele	84 m
Diâmetro do rotor	70 m
Orientação e configuração do rotor	Upwind, 3 pás
Rotação nominal	2,14 rad/s (20 RPM)
Torque nominal do rotor	736,79 kNm

Fonte: (ROSA; L. D., 2019) Adaptado

As configurações são similares à turbina 1,5*sle* da *General Electric* (GE), outros modelos foram escalonados para obtenção de suas características, assim, sabe-se que é um modelo que se aproxima de um aerogerador comercial real (ROSA; L. D., 2019). Os parâmetros de simulação foram inseridos conforme a Tabela 3.2:

Tabela 3.2 - Parâmetros das simulações de funcionamento dos aerogeradores

Parâmetro	Valores				
Turbulência	5.0	11.3	17.5	23.8	30.0

Fonte: Autor

A turbulência é inserida conforme valores de desvio padrão da amostra, ou seja, o vento mantém a média conforme valor de entrada em metros por segundo, porém, durante a simulação

os valores da incidência de vento desviam do padrão/média nas proporções da Tabela 3.2. Segue demonstração:

Tabela 3.3 - Demonstração do desvio padrão

Valores									Média	Desvio Padrão
2	10	20	36	12	34	97	0	5	24	30.3

Fonte: Autor

Tabela 3.4 - Parâmetros das simulações de funcionamento dos aerogeradores

Parâmetro	Valores				
Velocidade (m/s)	15.0	17.3	19.5	21.8	24.0

Fonte: Autor

Tabela 3.5 – Parâmetros das simulações de funcionamento dos aerogeradores

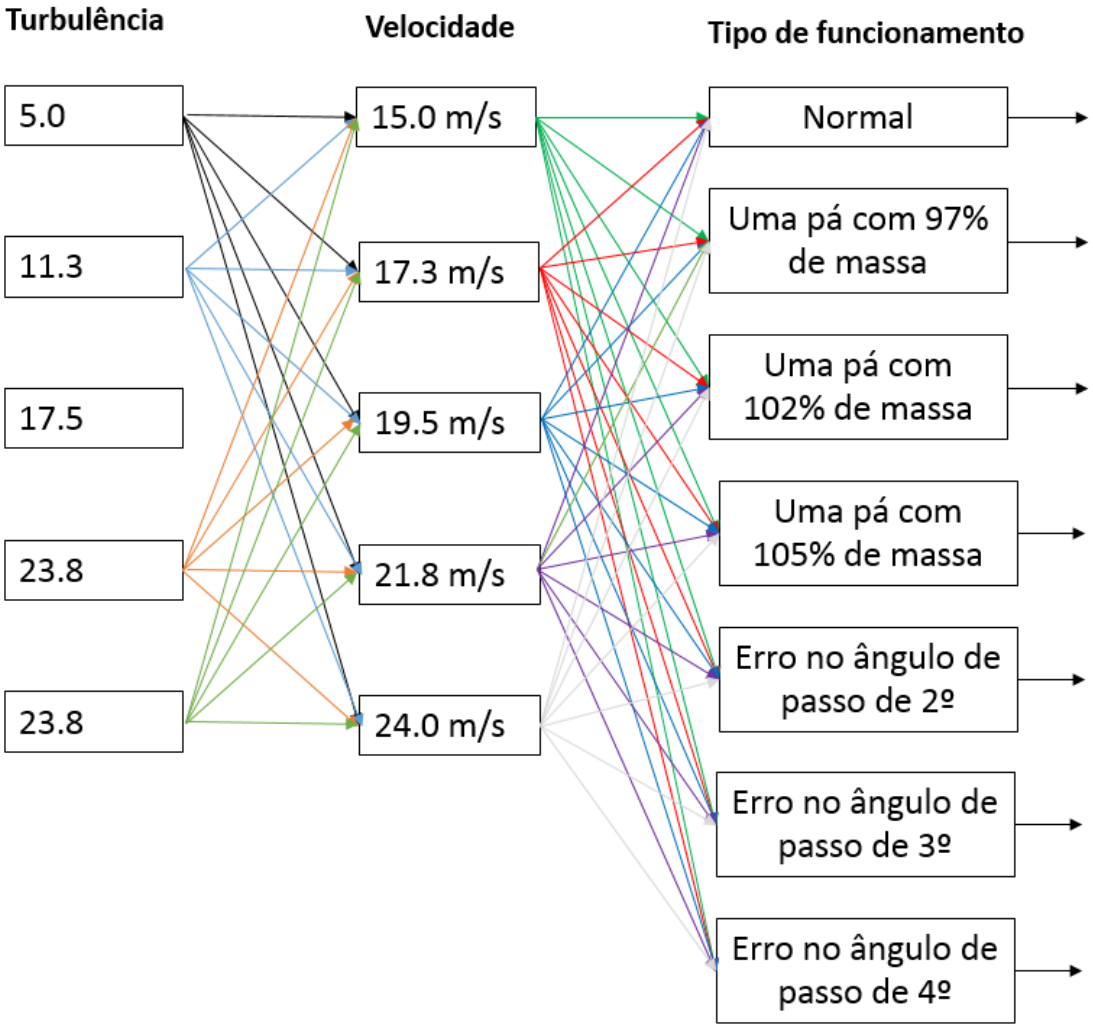
Parâmetro	Valores						
Tipo de funcionamento	Normal	Massa 97%	Massa 102%	Massa 105%	Pitch 2º	Pitch 3º	Pitch 4º

Fonte: Autor

Insere-se a turbulência como um ruído branco que mantém a velocidade média do vento constante, porém, com um desvio padrão para tornar os dados mais reais.

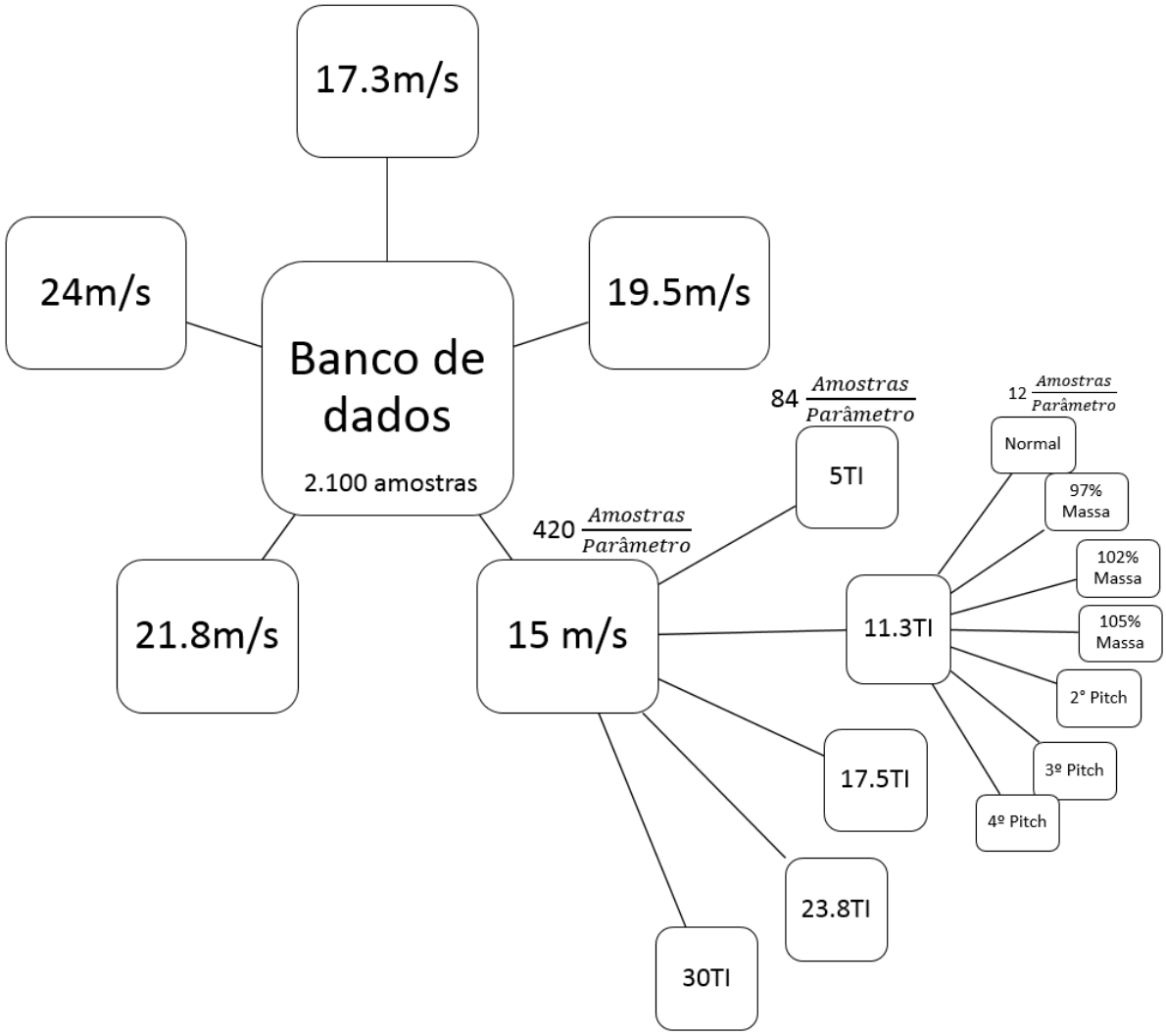
Assim, temos a disposição 5 valores de turbulência, simulado em cada uma das 5 velocidades, e em cada uma das 7 condições de funcionamento, totalizando 175 combinações. Cada uma dessas combinações foi simulada 12 vezes – visto que possui variáveis aleatórias como o ruído branco da turbulência do vento – e criou-se um banco de dados com 2.100 amostras diferentes, somando mais de 35h de funcionamento de aerogeradores nas mais diversas condições. As variáveis foram gravadas a uma frequência de 2.000Hz.

Figura 3.5 - 175 combinações possíveis de parâmetros



Fonte: Autor

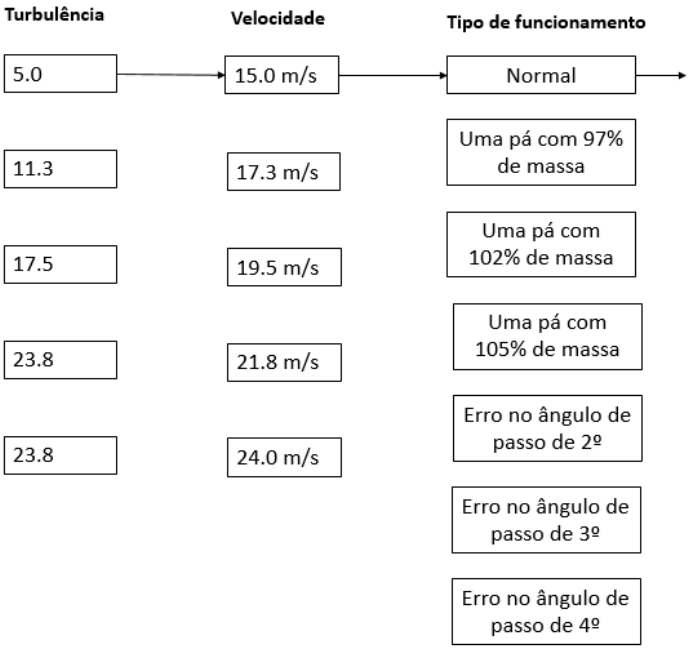
Figura 3.6 - Parâmetros que formam o banco de dados



Fonte: Autor

Dentre as 175 combinações de parâmetros temos a primeira conforme a Figura 3.7:

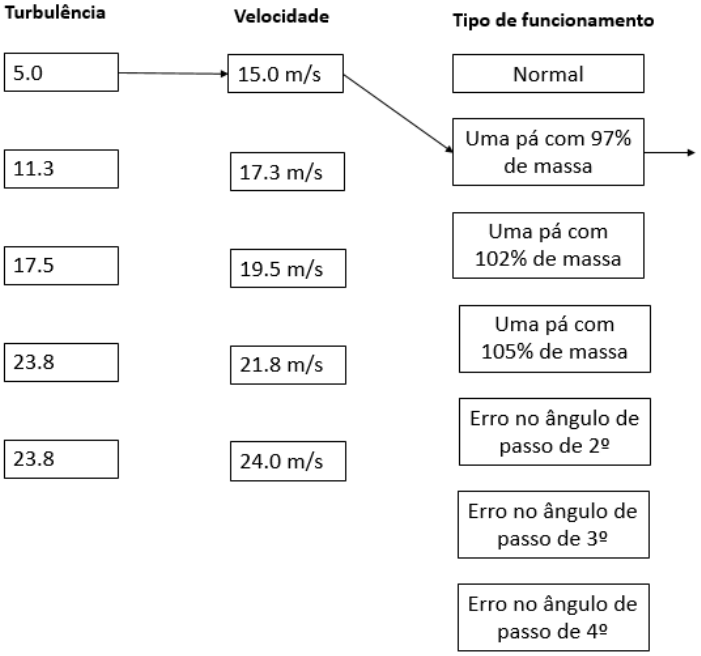
Figura 3.7 – Geração de dados



Fonte: Autor

Após a geração de 12 amostras com a turbulência, velocidade e tipo de funcionamento conforme a Figura 3.7, o algoritmo muda a geração para o próximo parâmetro de tipo de funcionamento, conforme Figura 3.8:

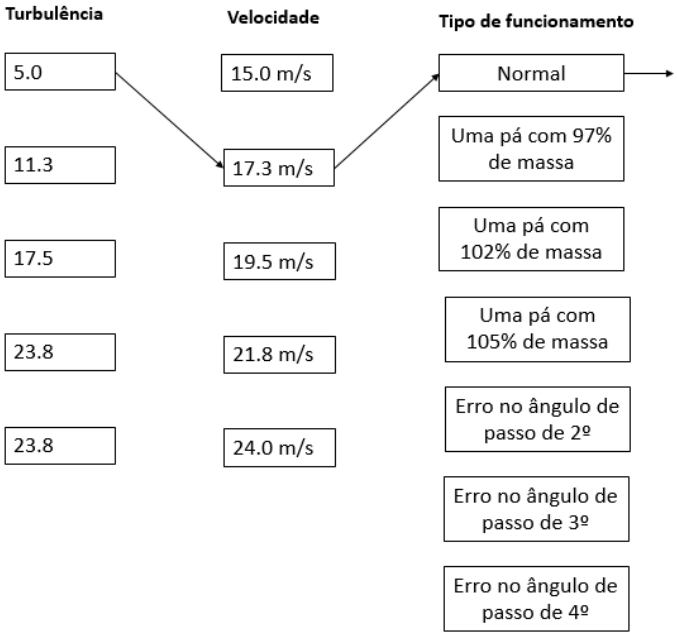
Figura 3.8 - Geração de dados



Fonte: Autor

Depois de gerar-se os dados com determinada turbulência e velocidade em todos os sete (7) tipos de funcionamento, a geração se dá na próxima velocidade determinada, conforme Figura 3.9:

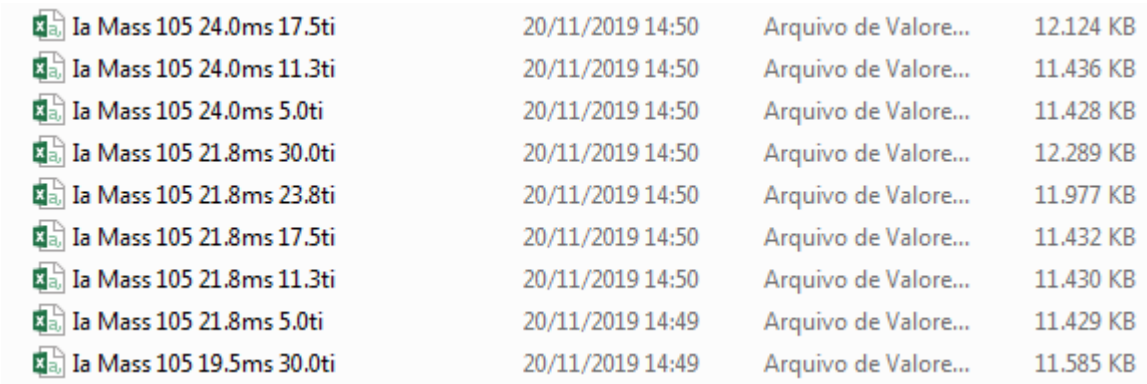
Figura 3.9 - Geração de dados



Fonte: Autor

Esse processo é repetido até que seja produzido dados de simulação com as 175 combinações de turbulência, velocidade e tipos de funcionamento. Originando em uma lista de amostras exemplificada na Figura 3.10:

Figura 3.10 - Dados de simulação



Fonte: Autor

Temos os arquivos conforme fórmula:

$$Variável_{obs.} + Tipo_{Func.} + Velocidade_{Vento} + Turbulência_{Vento}.csv \quad (34)$$

Sendo:

$Variável_{obs.}$ - Variável observada (Tensão, corrente, velocidade do rotor...)

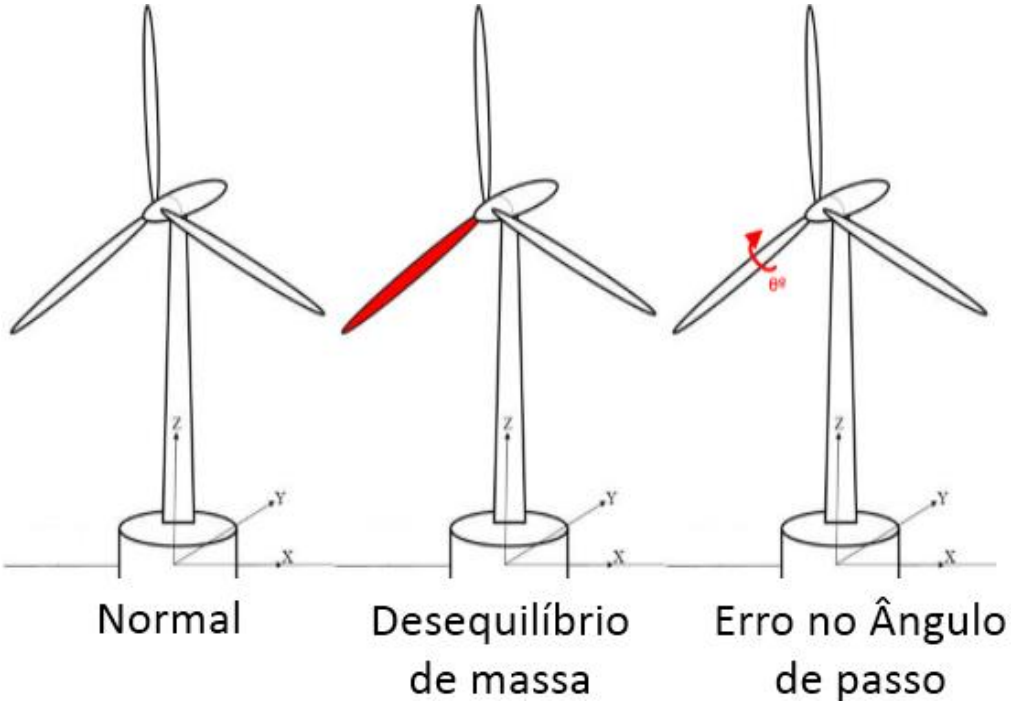
$Tipo_{Func.}$ - Tipo de funcionamento do aerogerador, entre as 7 opções (Normal, desequilíbrio de massa...)

$Velocidade_{Vento}$ - Velocidade de incidência do vento na TE (15.0 m/s, 17.3 m/s...)

$Turbulência_{Vento}$ - Desvio padrão do vento na TE (5.0, 17.5...)

Escolheu-se as velocidades e os tipos de funcionamento – e falhas - conforme (KANDUKURI, S.; 2016). Onde o desequilíbrio de massa é somente em uma das pás e tem o valor de -3%, +2% ou +5%, como na pá destacada na imagem central da Figura 3.11 e o erro do ângulo de pitch de 2º, 3º e 4º aparece como na imagem à direita da Figura 3.11 abaixo:

Figura 3.11 - Condições de funcionamento simuladas



Fonte: (SHIVAJI GANESAN T, 2016)[15] adaptado

[15] Disponível em: https://www.researchgate.net/figure/16-Schematic-of-floating-wind-turbine-and-discretized-panels-shallow-draft-barge-as_fig32_303663059. Acesso em: 8 ago. 2019

Os tipos de funcionamento possíveis da TE, para inserção no algoritmo, traduz-se de forma simplista, conforme:

Figura 3.12 - "Tradução" das condições de funcionamento para o algoritmo

Condição de funcionamento	Classe no algoritmo
Normal	100
Uma pá com 97% de massa	97
Uma pá com 102% de massa	102
Uma pá com 105% de massa	105
Erro no ângulo de passo de 2º	2
Erro no ângulo de passo de 3º	3
Erro no ângulo de passo de 4º	4

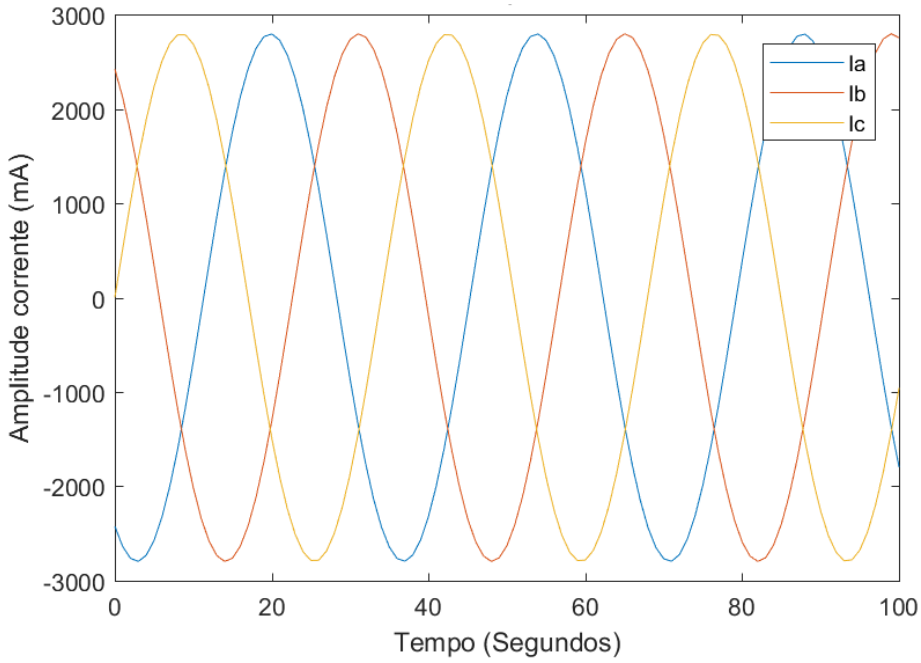
Fonte: Autor

3.2.2 Banco de dados utilizado

O banco de dados foi gerado com simulações do funcionamento de TE durante 120 segundos, onde somente os últimos 60 segundos foram utilizados, gerando portanto 120.000 *features*/características. Em cada amostra foram salvas as correntes das três fases (Ia, Ib e Ic), as três tensões trifásicas (Va, Vb e Vc), potência do gerador, velocidade real e velocidade estimada do rotor.

Conforme (MALIK, H.; MISHRA, S., 2015) sinais de corrente são mais confiáveis para monitoramento da condição da TE, então, como resultado, a abordagem de identificação de falha de desequilíbrio com essa técnica tem um enorme benefício econômico. Assim sendo, para esse trabalho, utiliza-se somente as correntes do gerador, demonstradas na plotagem abaixo:

Figura 3.13 - Plotagem das correntes do gerador da TE



Fonte: Autor

A medida das correntes é em miliAmpère, cada célula então representa o valor de uma corrente em um instante de tempo, conforme a representação dos dados segue na Tabela 3.6:

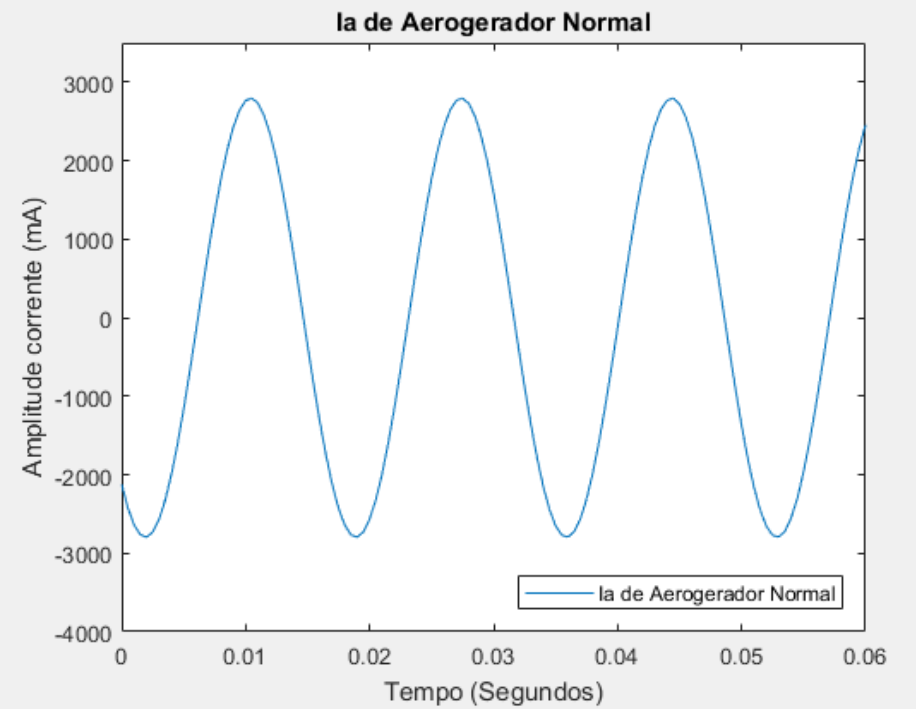
Tabela 3.6 - Amostra banco de dados da corrente do aerogerador

Amostra/Tempo(s)	0	0.001	...	119.99
Amostra 1	1942.3	2279.8	...	2539.3
Amostra 2	2665.8	2456.9	...	2158.8
...	...	...	...	...
Amostra 2100	2016.7	2335.6	...	2576.4

Fonte: Autor

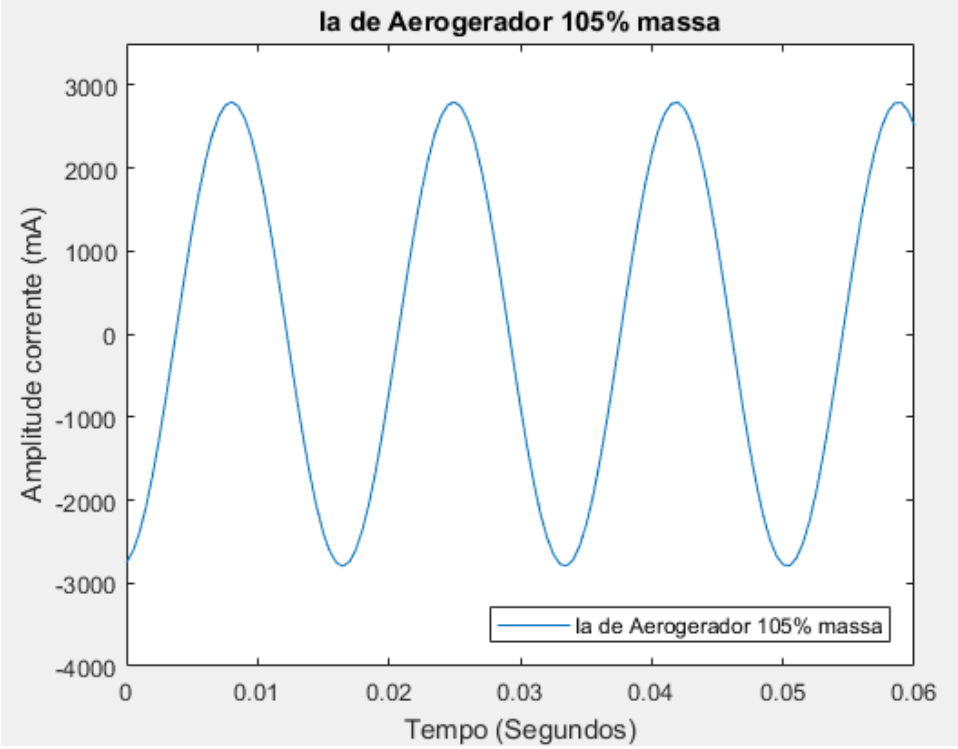
Assim, temos a plotagem de uma amostra da corrente do gerador elétrico no domínio do tempo conforme a Figura 3.14:

Figura 3.14 - Corrente de gerador no domínio do tempo com funcionamento normal



Fonte: Autor

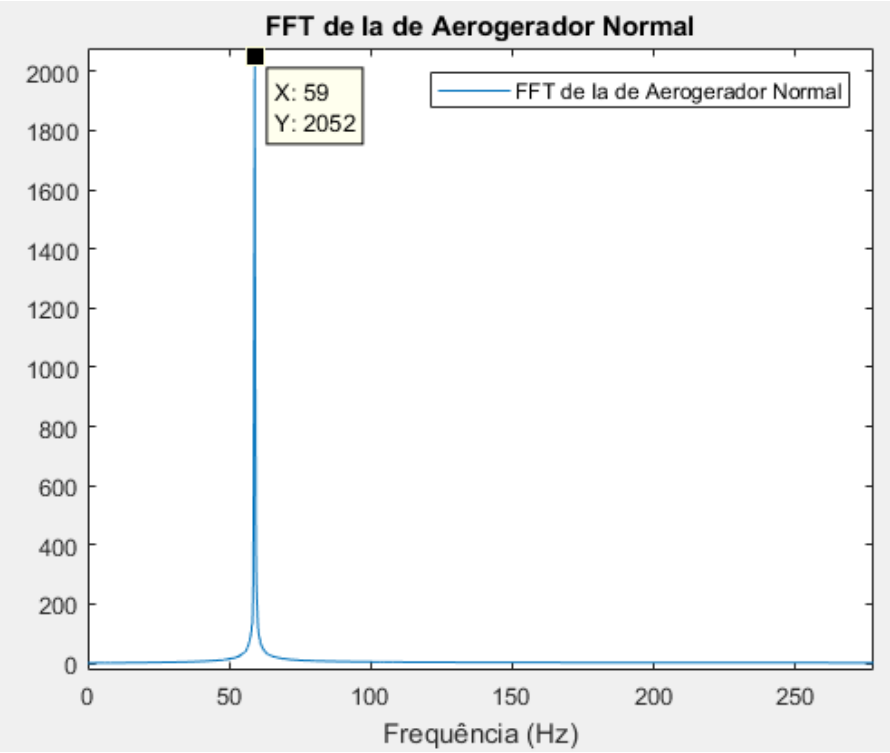
Figura 3.15 - Corrente Ia de gerador no domínio do tempo com condição propensa a falha – 105% de massa em uma pá



Fonte: Autor

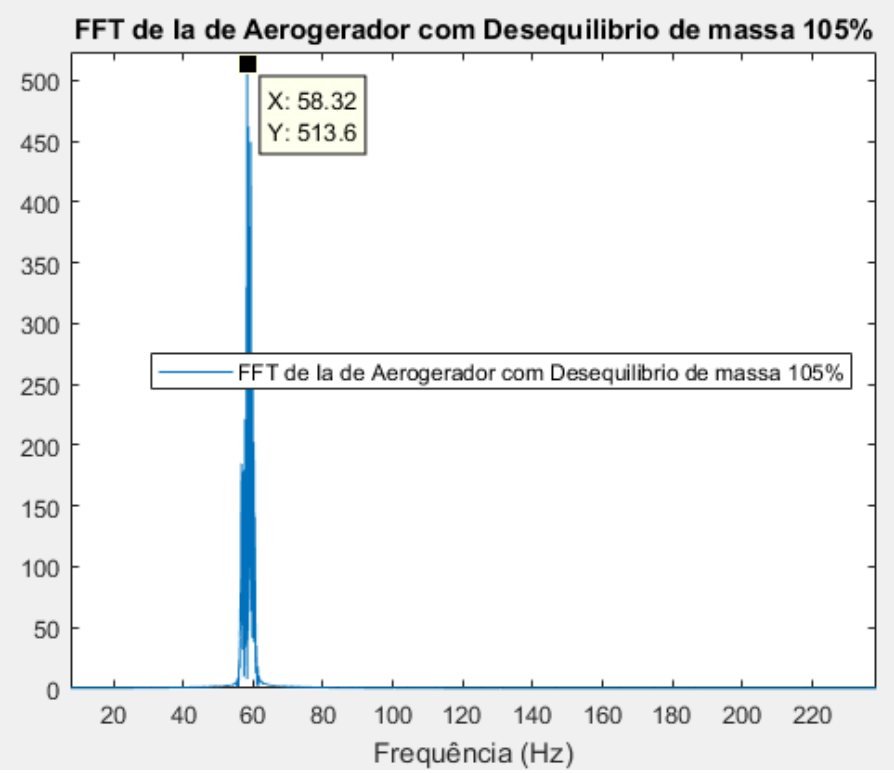
Nota-se que os dados das correntes no domínio do tempo têm diferenças imperceptíveis entre aerogeradores com funcionamento normal e anormal. Para verificação dos dados aplica-se FFT em uma amostra da corrente ‘Ia’ onde se espera uma componente principal próxima a frequência fundamental da rede, ou seja, próximo aos 60 Hz.

Figura 3.16 - FFT em Ia normal



Fonte: Autor

Figura 3.17 - FFT em Ia com desequilíbrio



Fonte: Autor

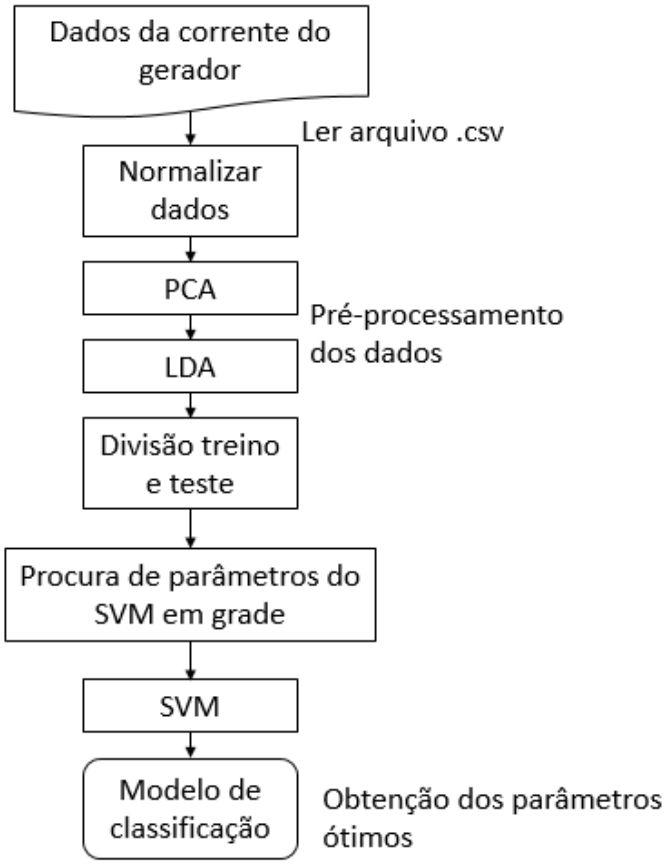
As diferenças entre a Figura 3.16 e Figura 3.17 na amplitude da corrente, pode ser justificada devido a valores de incidência de vento e turbulência diferentes. Porém, nota-se uma distância maior entre a frequência observada (58.32Hz) e esperada (60Hz) tratando-se da turbina eólica com desequilíbrio, um indicio, ainda que não conclusivo sobre as consequências de um funcionamento anormal.

3.2.3 Metodologia de Malik adaptada

Para o diagnóstico e classificação dos dados utiliza-se duas metodologias, a primeira é conforme o trabalho de Malik descrito em (MALIK H., 2016) e a segunda conforme ao trabalho de Kandukuri em (KANDUKURI S., 2019). Espera-se um aperfeiçoamento das técnicas utilizadas, além da produção de conhecimento. E devido a comparação de dois métodos que obtiveram resultados próximos a 100% de taxa de acertos em suas classificações, espera-se contribuir com a pesquisa em aprendizado de máquina aplicada a classificação do funcionamento de TE.

O método de Malik foi adaptado buscando melhorias e as técnicas de pré-processamento foram modificadas para melhorar eficiência. Para melhor ilustração do funcionamento dos algoritmos segue o fluxograma conforme Figura 3.18:

Figura 3.18 - Fluxograma de Malik adaptado



Fonte: Autor

Primeiramente utiliza-se a biblioteca *Pandas* para importação das 2.100 amostras (com 120.000 características cada) que serão utilizadas. Observa-se uma das correntes, a corrente "Ia" do gerador. Após isso normaliza-se os dados e emprega-se PCA e LDA.

Na análise de componentes principais definiu-se como objetivo extrair *n* componentes principais, tais que representassem 99,99% da variância total das características originais.

A LDA utilizou-se das 2.100 amostras e das componentes principais extraídas pela PCA e fez a discriminação linear conforme a classe de cada amostra. Assim, obtém-se as características que maximizam a distância entre classes diferentes entre todas amostras.

Então, divide-se as amostras entre treino e teste de forma aleatória. Os dados de treino – 70% das amostras – são inseridos na "pesquisa em grade" em busca dos parâmetros ótimos da SVM.

Na função *GridSearch CV* - responsável pela pesquisa em grade - insere-se os valores de *kernel*, *gamma* e C. Foram inseridos quatro tipos de *kernel*, sete valores para C e seis para *gamma*, totalizando 168 combinações. Os parâmetros são baseados em (PEDREGOSA et al., 2011) e seguem:

Tabela 3.7 - Parâmetros para Kernel

Parâmetro	Valores			
Kernel	RBF	Sigmoid	Polynomial	Linear

Fonte: Autor

Tabela 3.8 - Parâmetros para C

Parâmetro	Valores						
C	1	10	10^3	50^3	10^4	50^4	10^5

Fonte: Autor

Tabela 3.9 - Parâmetros para Gamma

Parâmetro	Valores					
Gamma	0.001	0.005	0.01	0.1	1	10

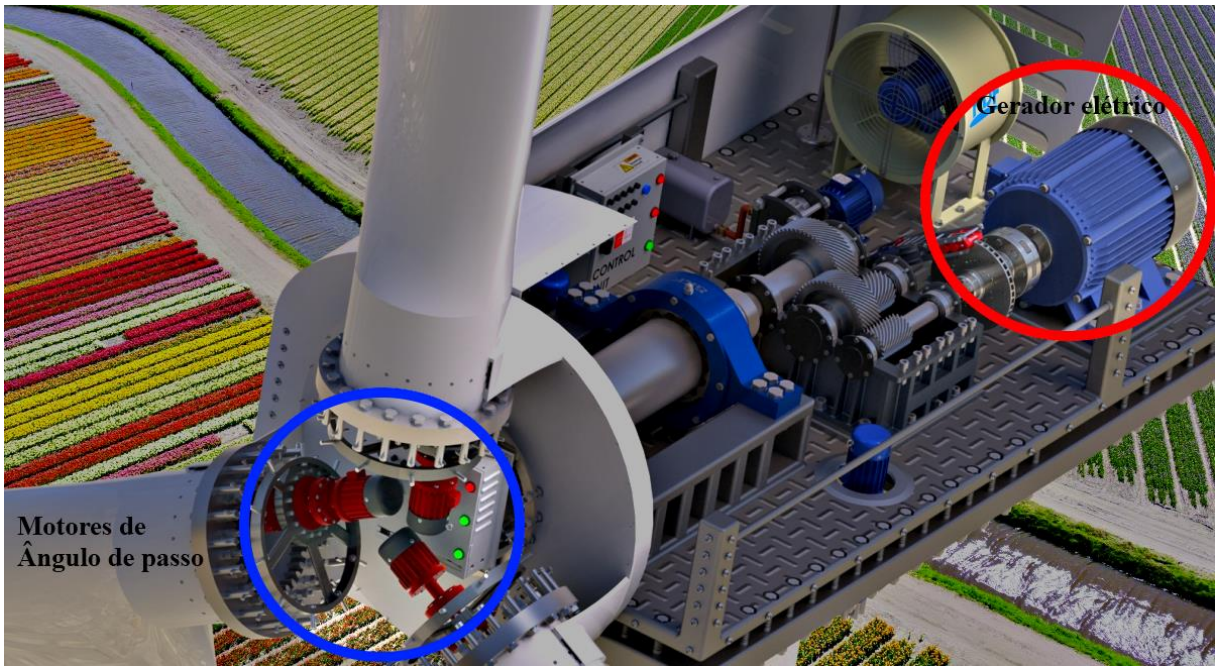
Fonte: Autor

É feito a busca pelos melhores parâmetros e gera-se um modelo de classificação ótimo, ou seja, obtém-se os parâmetros para a SVM gerar o hiperplano de separação ótimo. Esses valores são inseridos na SVM juntamente com os dados de teste para obtenção de indicadores finais sobre o classificador.

3.2.4 Metodologia de Kandukuri adaptada

O método de Kandukuri foi adaptado devido as divergências no propósito do autor. A técnica de Kandukuri consiste no monitoramento dos motores de controle de ângulo de passo, observado em azul na Figura 3.19, a procura de 3 tipos de falhas: no estator, rotor ou rolamentos. Enquanto nesse trabalho tem-se o objetivo de encontrar desequilíbrio de massa nas pás ou desajuste de ângulo de passo, monitorando as correntes do gerador principal, observado em vermelho na Figura 3.19 abaixo:

Figura 3.19 - Gerador elétrico e Motor de ângulo de passo de um aerogerador

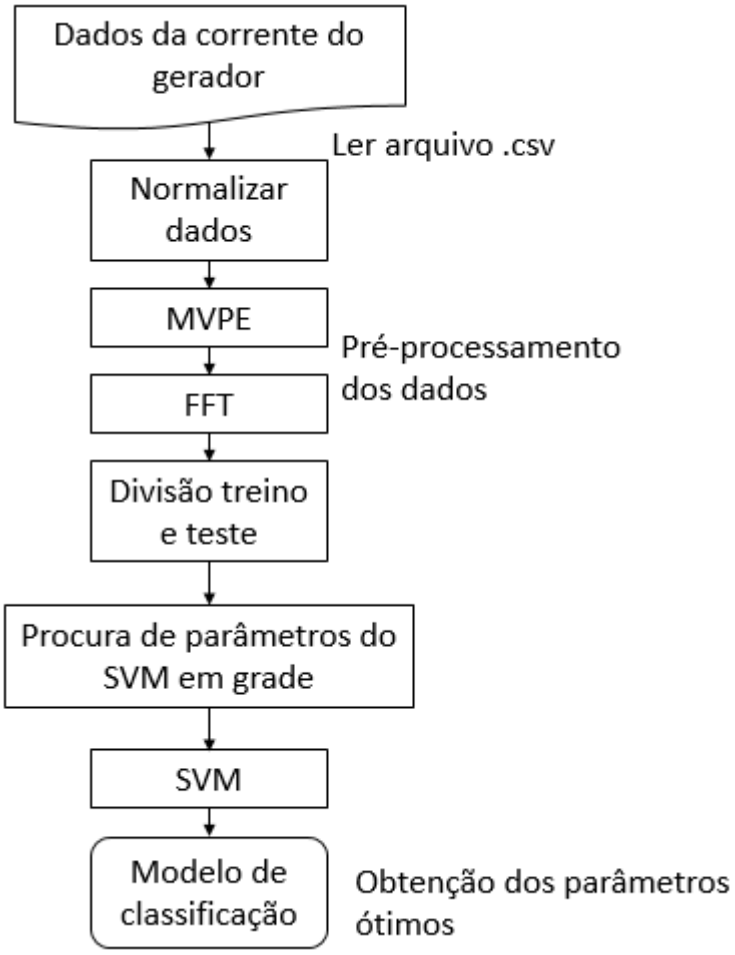


Fonte: GrabCAD[16] adaptado

Apesar do aerogerador da Figura 3.19 não ser igual a TE utilizada nesse estudo, pela figura mostrar a existência de uma caixa de engrenagens, a ilustração é pertinente e clara para sinalizar a localização do objeto de estudo de Kandukuri, em azul, e o objeto fonte de dados nesse estudo, o gerador elétrico. Para melhor ilustração do funcionamento dos algoritmos segue o fluxograma conforme Figura 3.20:

[16] Disponível em: https://grabcad.com/library/wind-turbine-100. Acesso em: 6 fev. 2020

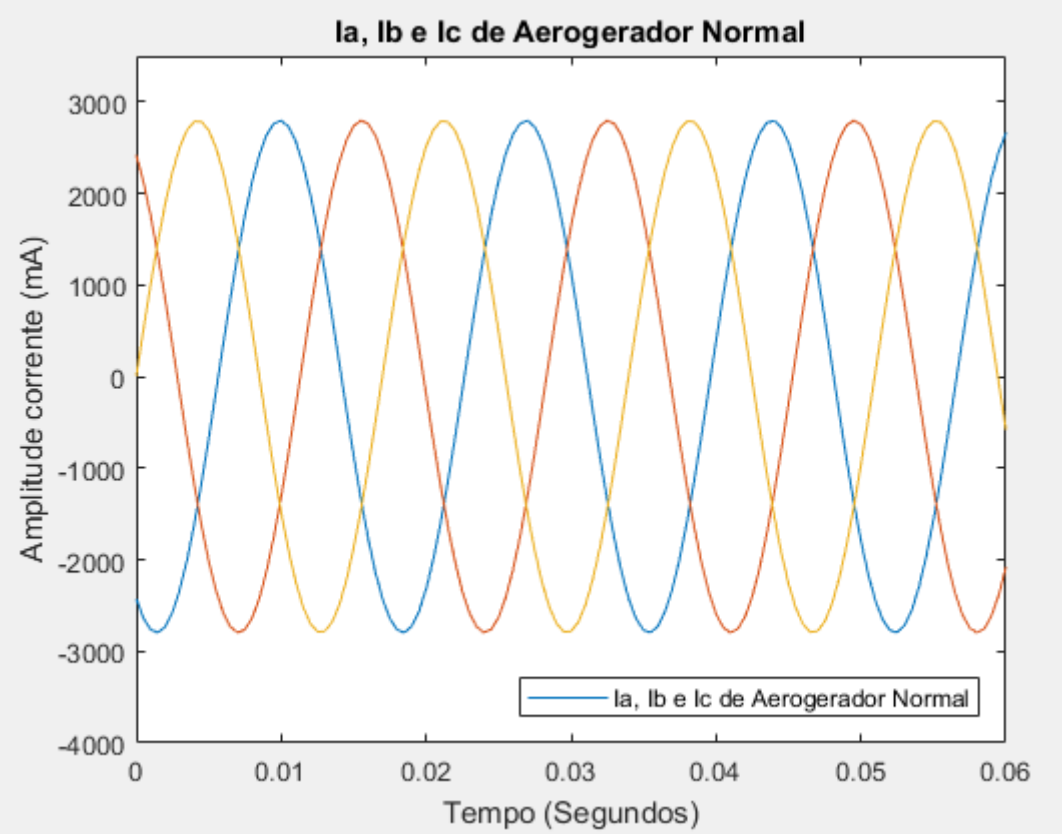
Figura 3.20 - Fluxograma dos algoritmos



Fonte: Autor

Utiliza-se a biblioteca *Pandas* para importação dos dados. Para técnica de Kandukuri é necessária uma captura instantânea (*snapshot*) de cinco segundos das três correntes do gerador, utiliza-se as correntes Ia, Ib e Ic. Portanto, somam-se assim três variáveis, contendo 10.000 características, para cada uma das 2.100 amostras.

Figura 3.21 - Correntes trifásicas do gerador



Fonte: Autor

Após isso normaliza-se os dados, afim de diminuir o tempo de processamento, e aplica-se MVPE e FFT. Via MVPE obtém-se somente uma variável por amostra, chamada de ‘ip’, a qual conforme (KANDUKURI, S., 2019) contém mais informações do que somente uma corrente. Adota-se então FFT onde ‘ip’ é representada no domínio da frequência por ‘IP’.

Então, as amostras são divididas entre treino e teste. Os dados de treino – 70% das amostras – são inseridos na “pesquisa em grade” juntamente insere-se os valores de *kernel*, *gamma* e C. Análoga a técnica de Malik, foram inseridos quatro tipos de *kernel*, sete valores para C e seis para *gamma*, totalizando 168 combinações. Os parâmetros são baseados em (PEDREGOSA et al., 2011) e seguem:

Tabela 3.10 - Parâmetros para Kernel

Parâmetro	Valores			
Kernel	RBF	Sigmoid	Polynomial	Linear

Fonte: Autor

Tabela 3.11 - Parâmetros para C

Parâmetro	Valores						
C	1	10	10^3	50^3	10^4	50^4	10^5

Fonte: Autor

Tabela 3.12 - Parâmetros para Gamma

Parâmetro	Valores					
Gamma	0.001	0.005	0.01	0.1	1	10

Fonte: Autor

A busca pelos parâmetros é feita em cada uma das partições da validação cruzada e o melhor classificador é o que obter a melhor média de acurácia em todas as partições

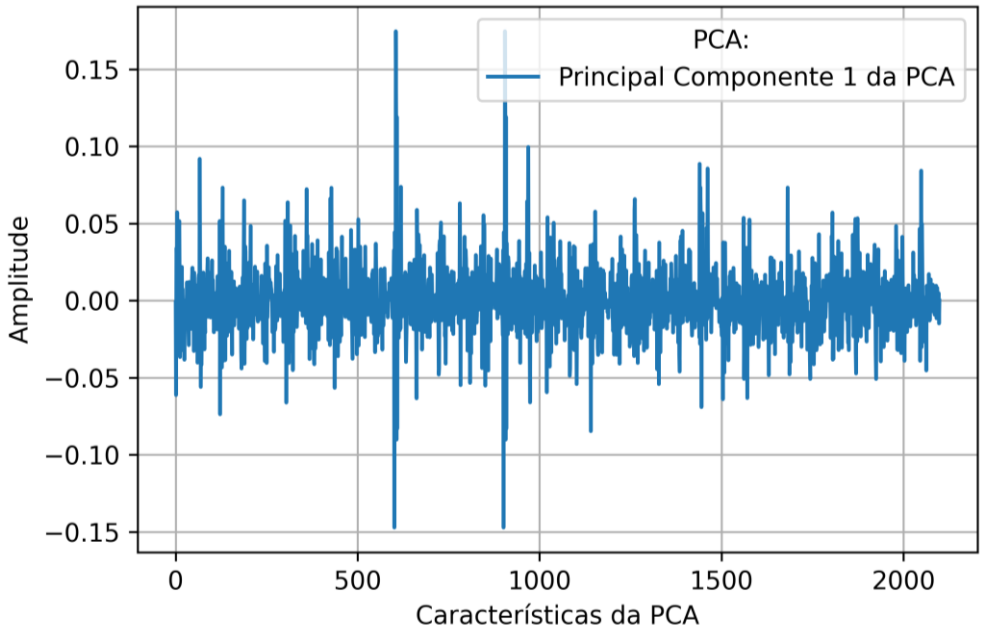
4. APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

Este capítulo apresenta os resultados das soluções encontradas para resolver o problema proposto.

4.1 MALIK ADAPTADO

Na PCA extraiu-se componentes que representassem 99,99% da variância total das características originais. Devido ao alto custo computacional, proporcional ao tamanho do banco de dados, calculou-se a PCA em um servidor com 25Gb de memória RAM e 16 núcleos, onde obteve-se aproximadamente 2.000 componentes que atingiram os requisitos de variância.

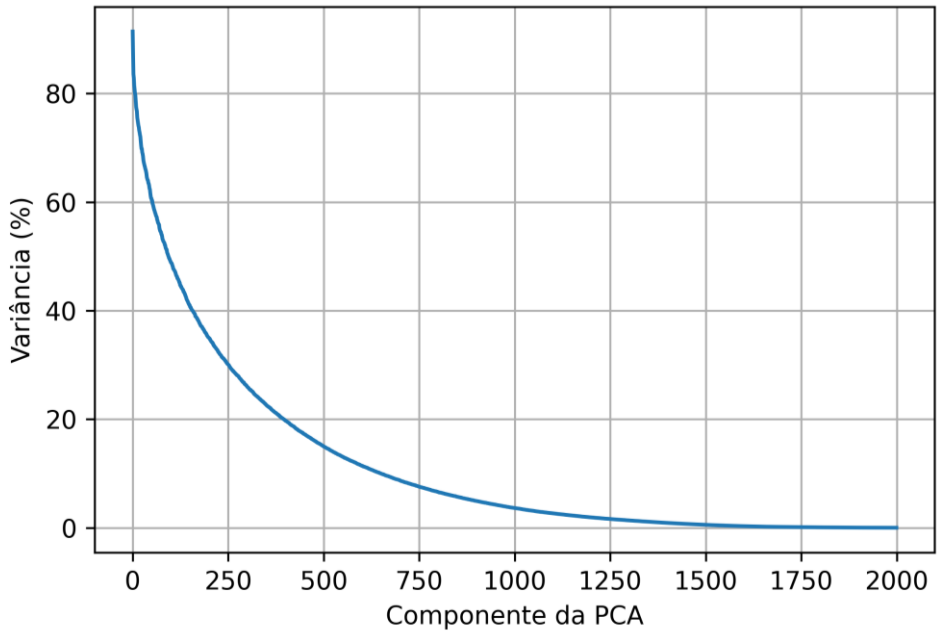
Figura 4.1 - Primeira componente da análise, representa mais de 90% da variância total dos dados



Fonte: Autor

As 2.000 componentes principais elegidas, apesar de expressar quase a totalidade da variância dos dados originais, têm um consumo de Gigabytes próximo a apenas 1,66% das amostras iniciais.

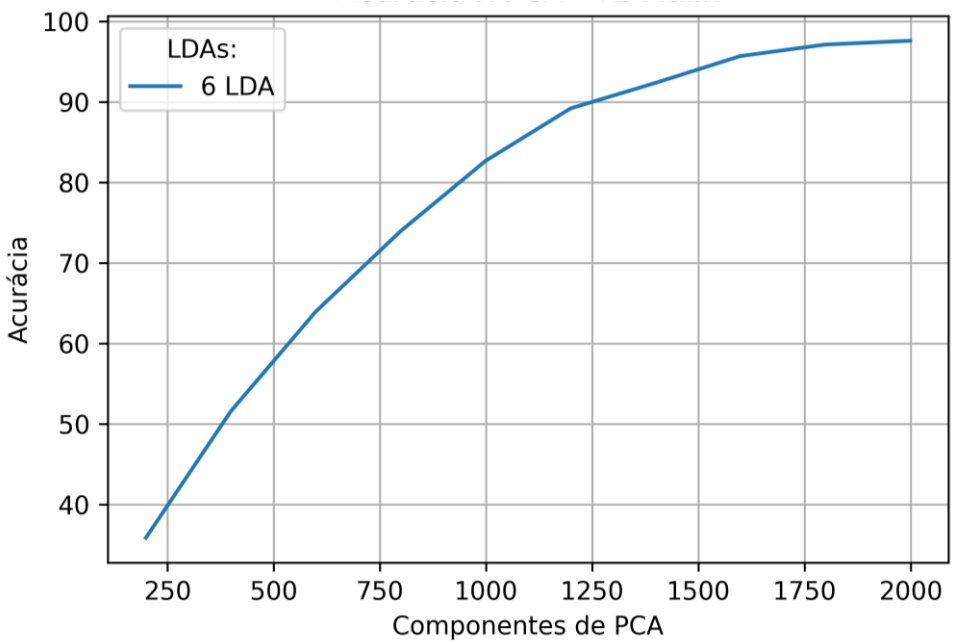
Figura 4.2 - Gráfico da representação da variância por componente



Fonte: Autor

Assim, a redução para tal número de componentes se justifica também pela relação direta com a acurácia, conforme Figura 4.3:

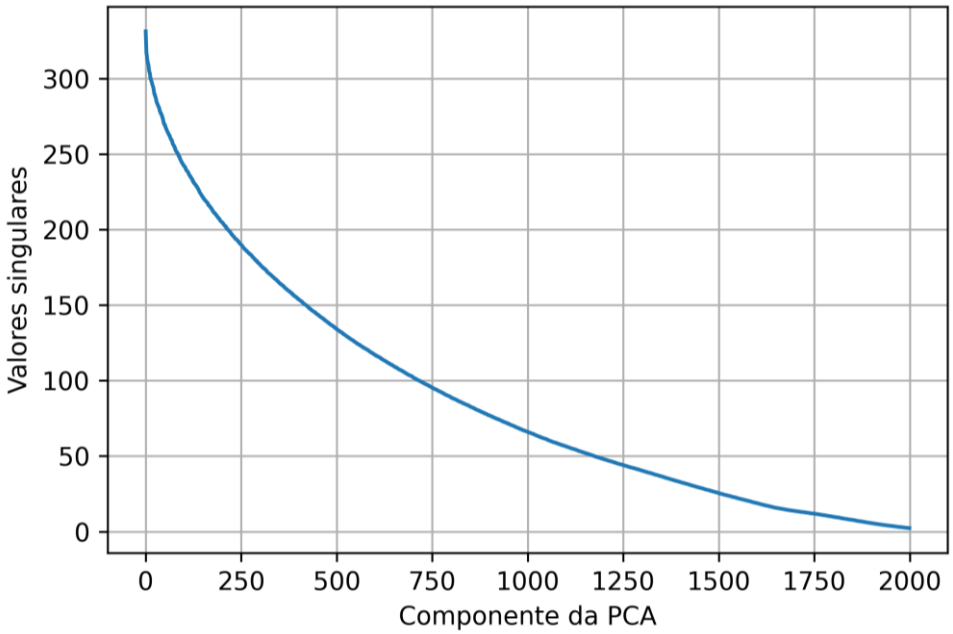
Figura 4.3 - Acurácia x PCA



Fonte: Autor

A curva da acurácia apresenta-se como inversamente dependente a função da variância em função do número de componentes, porém, conforme o número de componentes cresce notamos que essa dependência diminui. No final, a acurácia mostra-se uma relação inversamente proporcional ao número de valores singulares por componente, conforme:

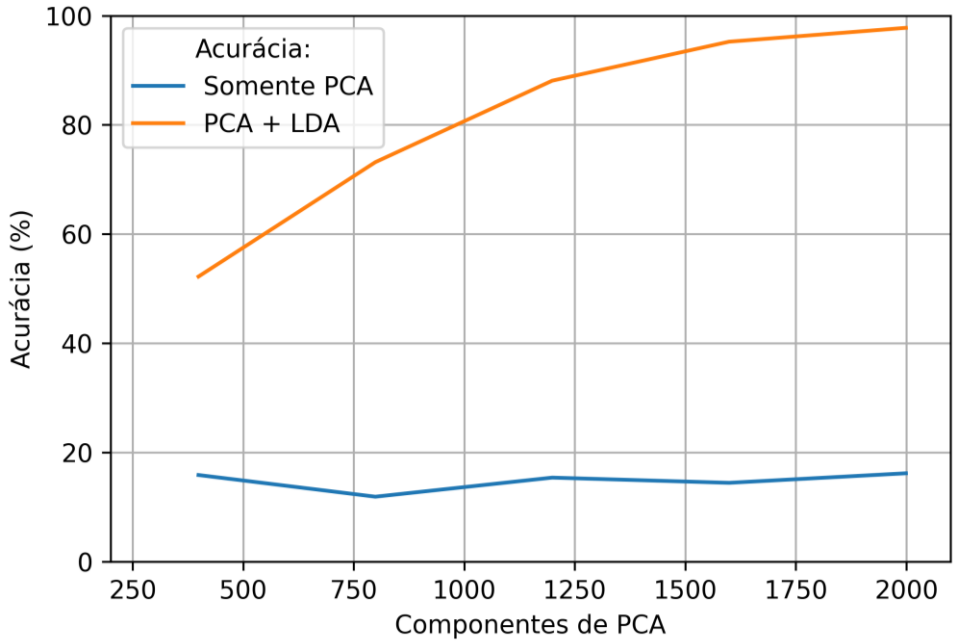
Figura 4.4 - Valores singulares por componente da PCA



Fonte: Autor

Assim, definiu-se o valor de 2.000 componentes para padronizar o número de características/*features*. A LDA utilizou-se das 2.100 amostras e suas 2.000 componentes principais (características) e fez a discriminação linear conforme suas classes, obtendo-se as seis características que maximizam a distância entre classes diferentes.

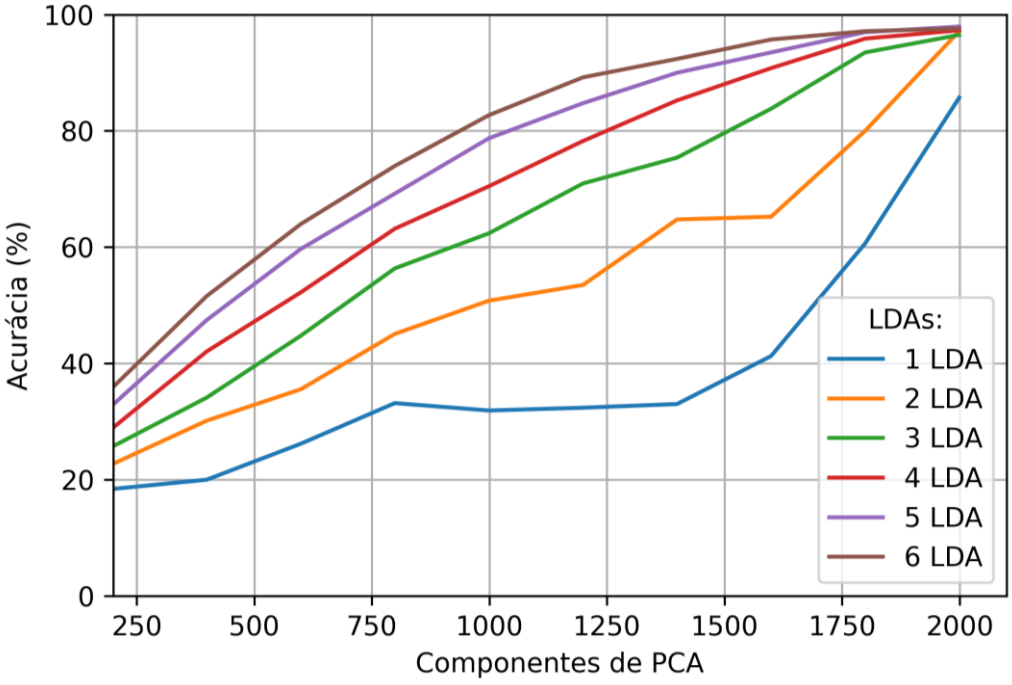
Tabela 4.1 - Acurácia com e sem LDA



Fonte: Autor

Os valores dos parâmetros mostram-se adequados, visto que busca-se uma maximização da acurácia e demais valores que representam a precisão e eficiência do algoritmo. Conforme a Figura 4.5 expondo a relação entre acurácia e métodos de pré-processamento.

Figura 4.5 - Acurácia PCA x LDA



Fonte: Autor

Os dados obtidos pelo LDA insere-se na pesquisa em grade, com validação cruzada dos dados de treino em 10 partições. Das 2.100 amostras, 1.440 utiliza-se para treino – aproximadamente 70% - logo, 144 amostras de classes aleatórias por partição.

O tempo para encontrar os melhores parâmetros foi de 2 minutos e 12 segundos. E obteve-se os parâmetros ótimos conforme a Tabela 4.2:

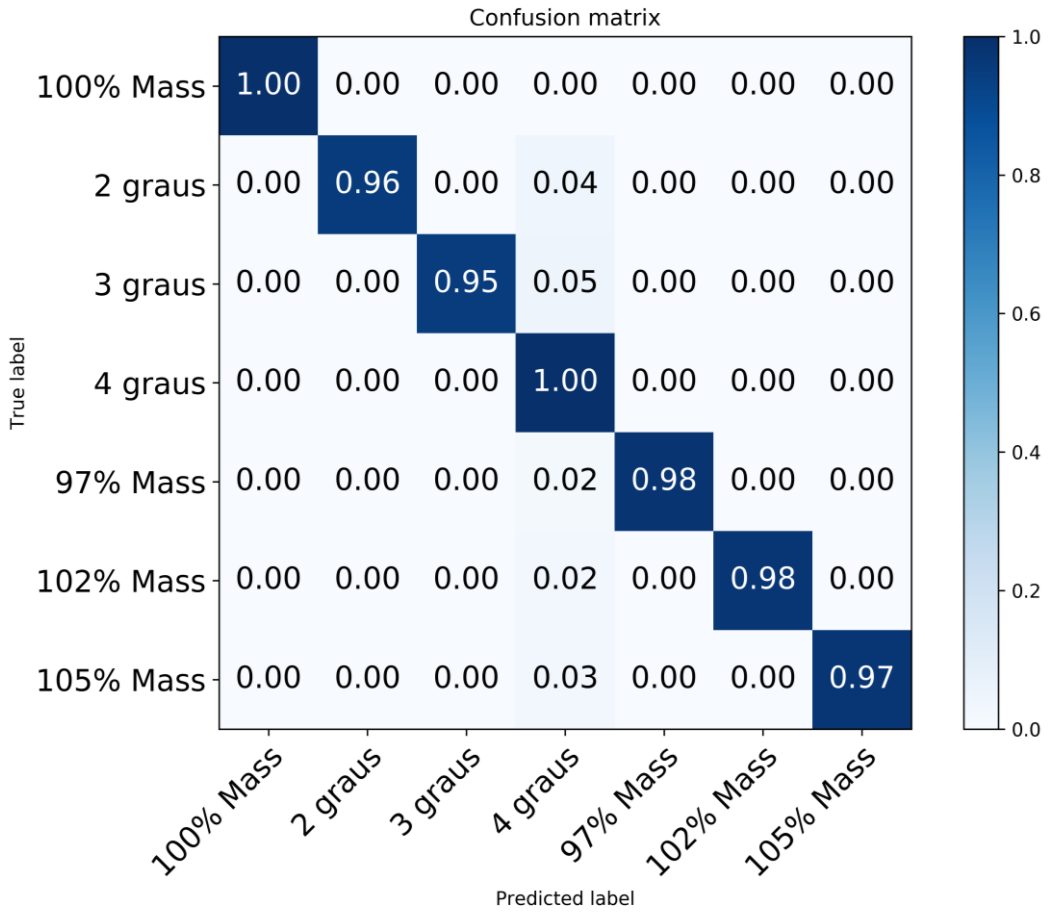
Tabela 4.2 - Parâmetros escolhidos pela pesquisa em grade

Kernel	C	Gamma	Grau	Função decisão
RBF	0.1	0.005	3	Um contra todos

Fonte: Autor

Aplica-se então os parâmetros obtidos pela pesquisa em grade na SVM, juntamente com os dados de teste – representando 30% dos dados originais – que não interferiram na pesquisa em grade, portanto, expõem com fidelidade se o classificador tem êxito. O resultado é a matriz de confusão abaixo:

Figura 4.6 - Matriz de confusão



Fonte: Autor

Observa-se uma acurácia alta, o classificador acertou 618 das 630 amostras de teste, obtendo-se uma taxa de acertos de 97.63%. Porém, outros indicadores devem ser auferidos para verificar se a categorização realmente teve sucesso, conforme Tabela 4.3:

Tabela 4.3 - Scores da SVM

Classe	Precisão	Recall	F1-Score	Nº de amostras testadas
2	1.00	1.00	1.00	102
3	1.00	0.96	0.98	91
4	1.00	0.95	0.97	100
97	0.84	1.00	0.91	80
100	1.00	0.98	0.99	96
102	1.00	0.98	0.99	85
105	1.00	0.97	0.99	79

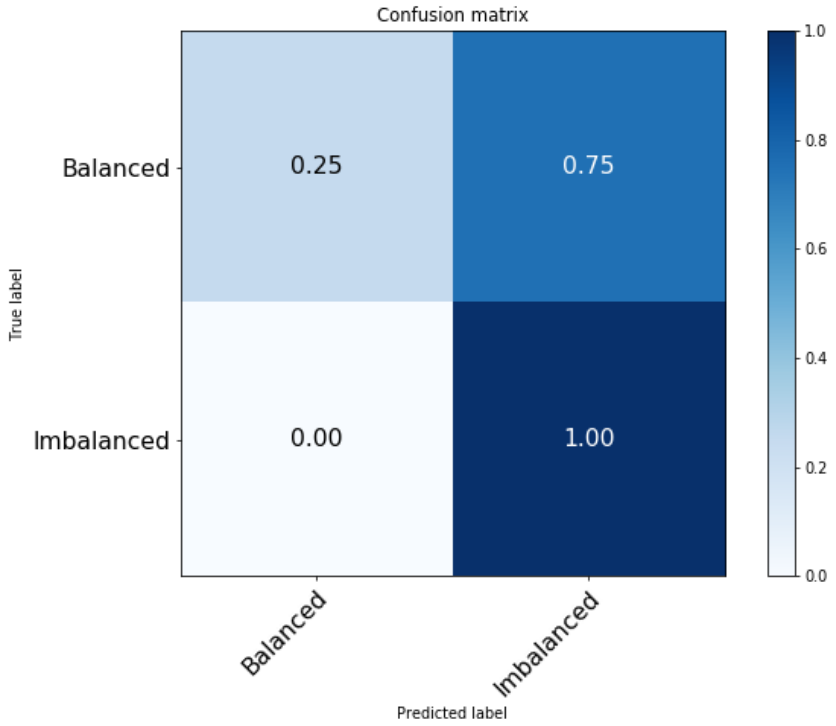
Fonte: Autor

Comparando os resultados desse estudo com (MALIK, H., 2016), é valido observar que no estudo de 2016 utilizaram-se dez variáveis para cada amostra, (MALIK, H., 2016) também fez uso de uma rede neural para classificação de apenas duas classes (funcionamento normal ou anormal), ou seja, houve um uso de dados por amostra 10 (dez) vezes maior, aplicação de um classificador mais robusto e com maior custo computacional, além de não haver a classificação quanto ao tipo ou gravidade da falha.

Nesse estudo obteve-se uma acurácia menor, porém, utilizou-se apenas a corrente do gerador - um banco de dados dez (10) vezes menor – além de valer-se de uma técnica de aprendizado de máquina simples e fazer uma classificação entre seis (6) condições de desequilíbrio.

Salienta-se que a retirada do método EMD deve-se aos resultados de classificação ruins, a seleção de IMFs principais não alterou a quantidade de dados, logo, não diminuiu o custo computacional, e trouxe perdas de acurácia relevantes, conforme:

Figura 4.7 - Acurácia de classificação com uso de EMD



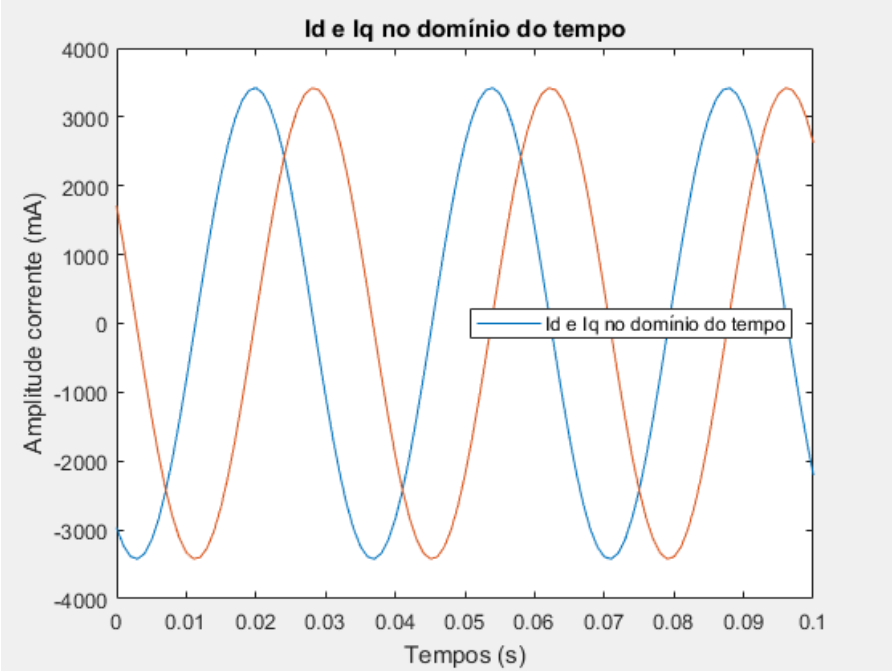
Fonte: Autor

Com acurácia média próximo a 40.0% substitui-se a técnica EMD pela LDA, conforme apresentado nos capítulos anteriores e com resultados demonstrados no decorrer dessa secção.

4.2 KANDUKURI ADAPTADO

Dos dados das correntes trifásicas, vide Figura 3.21, utiliza-se uma captura instantânea de 5 segundos, ou seja, toma-se 10.000 características como dado inicial para aplicação da metodologia, conforme (KANDUKURI, S., 2019). Esses dados iniciais são transformados, com o cálculo do MVPE, onde primeiramente obtém-se ‘Id’ e ’Iq’ que representam as correntes iniciais.

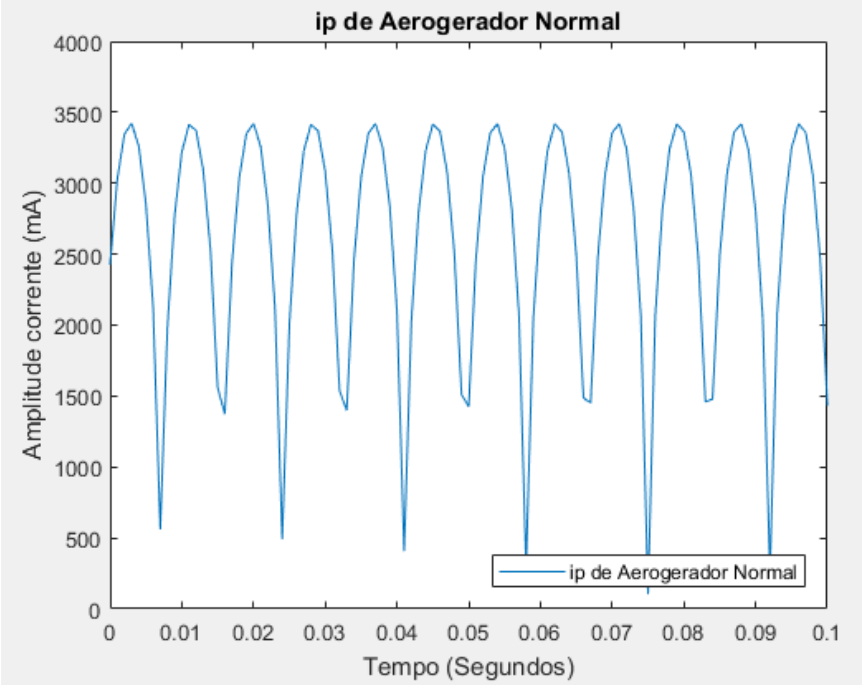
Figura 4.8 - id e iq



Fonte: Autor

Continua-se o cálculo do MVPE até obter-se ‘ip’, que representa as correntes trifásicas em apenas uma componente, conforme Figura 4.9:

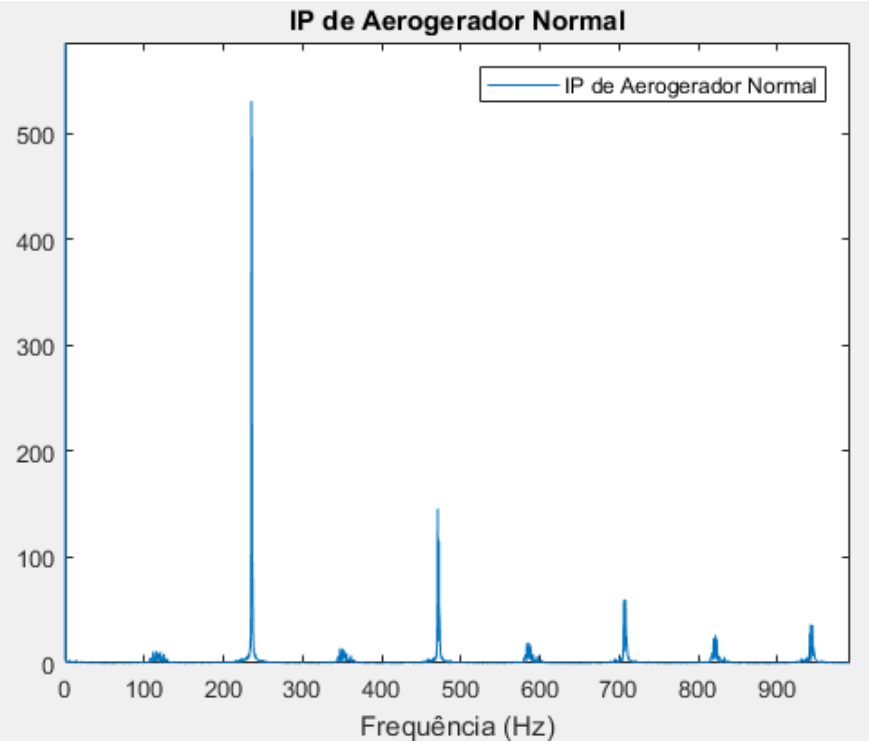
Figura 4.9 - ip



Fonte: Autor

Aplica-se FFT em ‘ip’, onde as 10.000 características no domínio do tempo são representadas em 5.000 frequências, conforme Figura 4.10:

Figura 4.10 - IP no domínio da frequência



Fonte: Autor

‘IP’ representa as componentes das correntes do domínio da frequência. Para possibilitar uma melhor classificação, insere-se 14 dados estatísticos que representem cada amostra, valores como média, desvio padrão, variância, entropia e outros, utilizando-se a biblioteca *SciPy 1.0*, conforme:

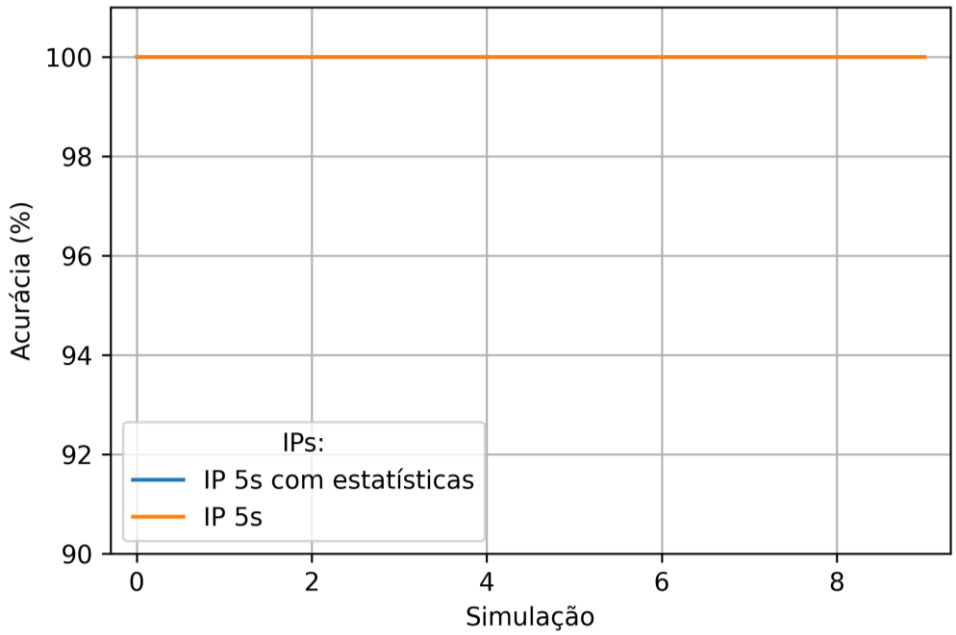
Tabela 4.4 - Descrição de valores estatísticos

Estatística	Descrição
Tmean	Média ponderada
hmean	Média harmônica
Gmean	Média Geométrica
Tstd	Desvio Padrão ponderado
Gstd	Desvio padrão geométrico
Median Absolute deviation	Desvio absoluto médio
Skew	Assimetria
Kurtosis	Kurosis Fisher
Variation	Coeficiente de variação
Kstatvar	Retorna um estimador imparcial da variância da estatística k
Tvar	Variância ponderada
Tsem	Erro padrão aparado a média
Sem	Erro padrão da média
Entropy	Entropia de uma distribuição para determinados valores de probabilidade

Fonte: Autor

Adiciona-se as 14 estatísticas como últimas características/*features* das amostras, as fórmulas utilizadas para cálculo, além do detalhamento da biblioteca, seguem (VIRTANEN, PAULI et al., 2020). Espera-se uma melhora na classificação, conforme (KANDUKURI, S., 2019):

Figura 4.11 – Comparação do impacto das estatísticas



Fonte: Autor

A adição de características estatísticas se mostrou desprezível. Então, os valores de 'IP' são inseridos na função *GridSearch CV*. Das 2.100 amostras, 1.440 foram utilizadas para treino – aproximadamente 70% - logo, cada uma das 10 partições continha 144 amostras de classes aleatórias. O tempo para a pesquisa em grade encontrar os melhores parâmetros foi de 700 minutos e 30 segundos, ou seja, mais de 11 horas. E os parâmetros ótimos seguem a Tabela 4.5:

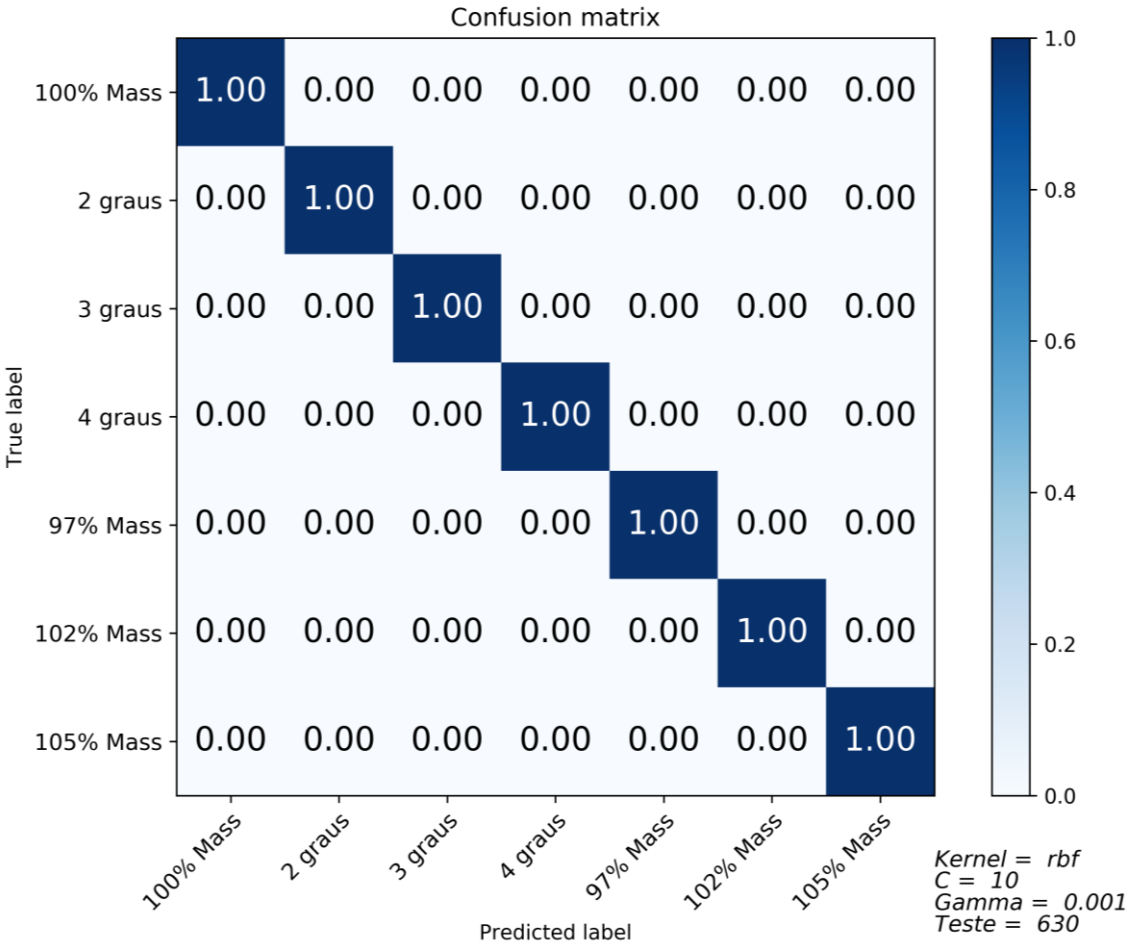
Tabela 4.5 - Parâmetros escolhidos pela pesquisa em grade

Kernel	C	Gamma	Grau	Forma da função decisão
RBF	10	0.001	3	Um contra todos

Fonte: Autor

Aplicam-se então os parâmetros obtidos pela pesquisa em grade na SVM, juntamente com dados de entrada o banco de dados de teste. Como os dados de teste não interferiram na pesquisa em grade, expõem a acurácia como dados novos. O resultado é a matriz de confusão da abaixo:

Figura 4.12 - Matriz de confusão



Fonte: Autor

Observa-se uma acurácia alta, o classificador acertou 630 das 630 amostras de teste, uma taxa de acertos de 100%. Porém, deve-se auferir outros indicadores para verificar se a categorização realmente teve sucesso, conforme Tabela 4.6:

Tabela 4.6 - Score da SVM

Classe	Precisão	Recall	F1-Score	Nº de amostras testadas
2	1.00	1.00	1.00	100
3	1.00	1.00	1.00	93
4	1.00	1.00	1.00	73
97	1.00	1.00	1.00	100
100	1.00	1.00	1.00	81
102	1.00	1.00	1.00	92
105	1.00	1.00	1.00	91

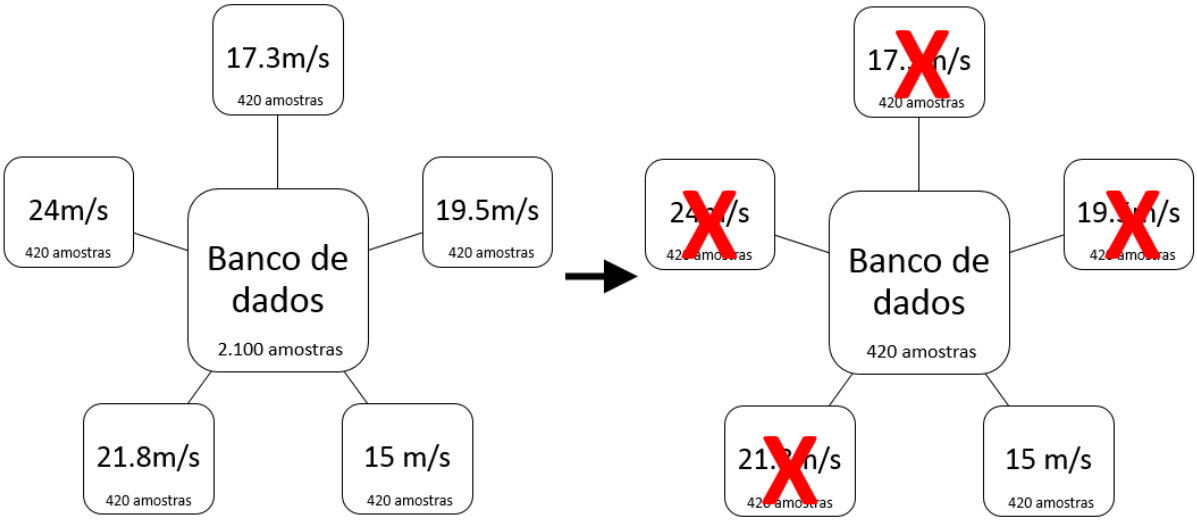
Fonte: Autor

Os demais indicadores corroboram com a acurácia preliminar, onde o classificador programado revela-se ótimo para os dados utilizados.

Para ponderar o peso dos parâmetros utilizados na geração dos ventos, é feita uma classificação de dados isolando algumas variáveis, como turbulência e velocidade. As simulações são repetidas diversas vezes com um randomização padronizada para possibilitar a quantificação dos resultados. Apesar de tratar-se dos mesmos dados, quando nos limitamos a analisar um parâmetro, consequentemente, faz-se uso de um subconjunto de dados menor do que o conjunto total, assim, tem-se um número menor de amostras.

Por exemplo, a utilização de somente simulações apenas com vento de incidência a 15 m/s ocasiona em dados de treinamento cinco vezes menor, conforme:

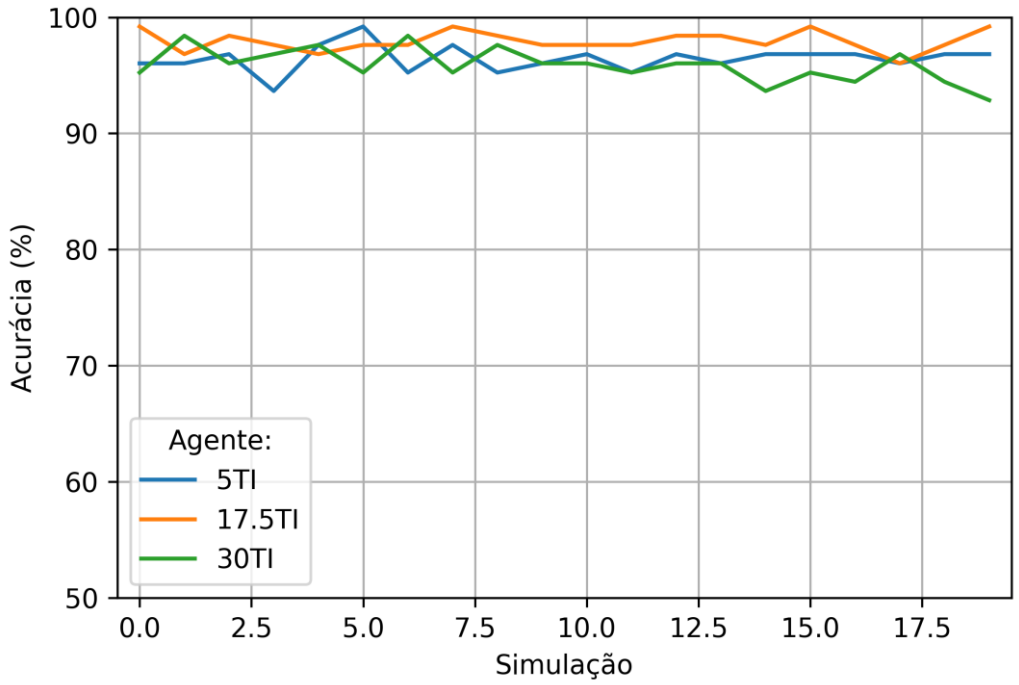
Figura 4.13 - Comparação de banco de dados sem e com "limitação" de parâmetros



Fonte: Autor

Primeiramente, se faz a classificação com dados de funcionamento do aerogerador com incidência de vento com baixa turbulência, após, com incidência de vento com alta turbulência.

Figura 4.14 - Classificação de dados de Baixa X Alta turbulência

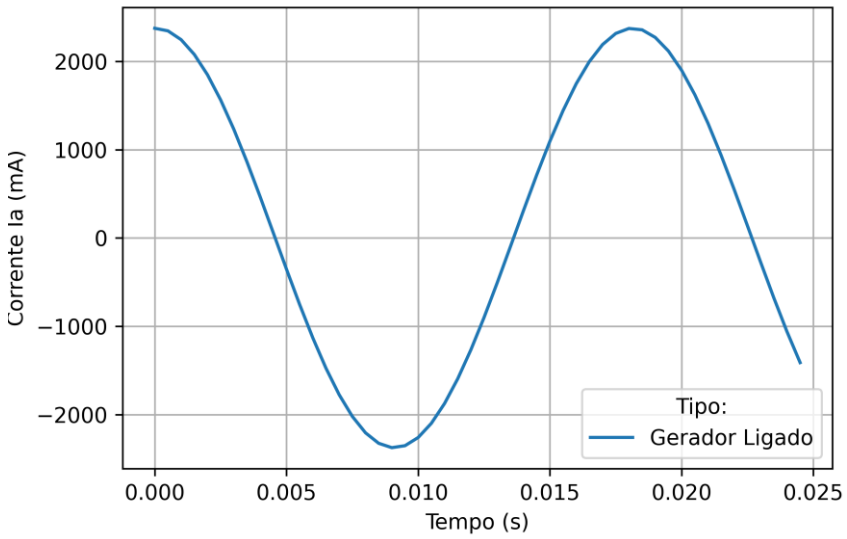


Fonte: Autor

Primeiramente observa-se uma queda média na acurácia, natural devido a menor quantidade de dados, conforme (RAKOTOMAMONJY, A.; 2003) o “sucesso” de um algoritmo de aprendizado de máquina é fortemente afetado pela qualidade dos dados. Nas simulações apenas com uma categoria de dados proporciona-se, para o SVM, uma quantidade menor de amostras para treino, o que afeta a qualidade da classificação, *Yu et al* em (YU, H.; VAIDYA, J.; JIANG, X.; 2006) expõem que a acurácia de um classificador SVM depende crucialmente em ter acesso ao conjunto de dados correto.

Nas vinte simulações percebe-se uma leve tendência na melhor classificação proporcionalmente ao aumento da turbulência.

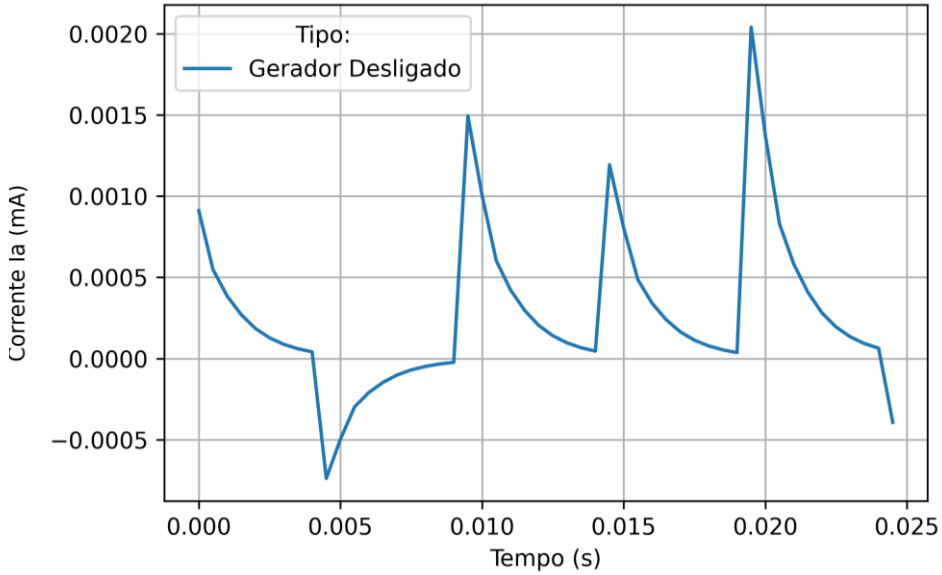
Figura 4.15 - Exemplo de corrente no domínio do tempo com gerador ligado



Fonte: Autor

Porém, é notável a queda de acurácia em altas turbulências, esse fenômeno pode estar relacionado ao fato de que em algumas simulações quando o aerogerador atinge velocidades muito altas (causadas por turbulências intensas) o gerador é desligado durante alguns instantes.

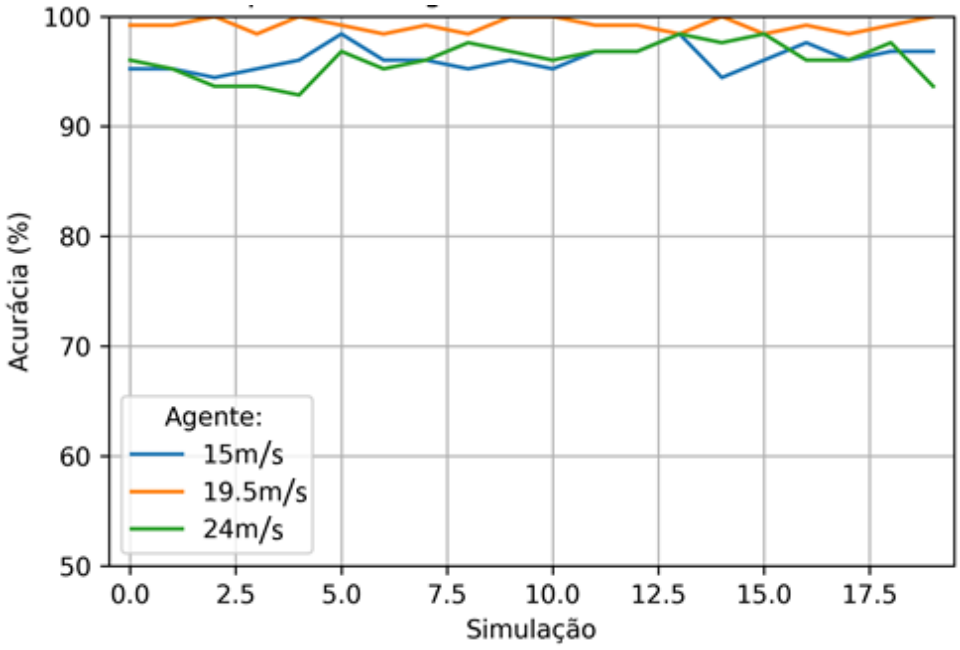
Figura 4.16 - Exemplo de corrente no domínio do tempo com gerador desligado



Fonte: Autor

O classificador demonstra lidar melhor com essa adversidade quando lhe é concedido um número maior de amostras. O processo de classificação se repete, simulando o funcionamento do aerogerador somente com incidência de vento com baixa (15 m/s), média (19.5 m/s) e alta velocidade (24 m/s).

Figura 4.17 - Classificação de dados de Baixa X Alta velocidade de vento



Fonte: Autor

Também nota-se uma leve tendência, a melhor classificação com ventos de média velocidade. A justificativa pode ser a mesma encontrada nas altas turbulências.

5. CONCLUSÃO

O trabalho atingiu o objetivo geral de implementar um algoritmo capaz de realizar o diagnóstico e a classificação da condição de funcionamento de acordo com sua severidade em aerogeradores. Os objetivos específicos foram integralmente concluídos. Obteve-se uma menor taxa de acertos na classificação com a adaptação do algoritmo de (MALIK, H., 2016) porém, houve uma maior diversificação na classificação e melhora nos processos. Aplicação do PCA com LDA foi consumada com êxito, com a demonstração da influência dos dois métodos em uma classificação ótima. Já os algoritmos desenvolvidos conforme (KANDUKURI, S., 2019) mostraram um grande avanço, apesar de a grande adaptação na aplicação, os resultados finais com a análise do módulo de parks estendido, a transformada rápida de *Fourier*, e a adaptação para apenas um estágio resultaram na taxa máxima de acertos na classificação de funcionamento das turbinas eólicas, ou seja 100,00%. Os resultados corroboram para a afirmação de que os métodos foram compreendidos, o emprego do classificador máquina de vetor de suporte se mostrou extremamente eficaz e útil, e as técnicas foram bem aplicadas pelo autor.

Comparando o algoritmo adaptado de (MALIK, H., 2016) – do subcapítulo 4.1 - com a adaptação de (KANDUKURI, S., 2019) – do subcapítulo 4.2 - podemos pontuar seus prós e contras. No quesito de tempo de processamento o algoritmo de ‘Malik adaptado’ obteve uma ampla vantagem, o uso de PCA e LDA para redução de dados se mostrou extremamente eficiente e impactou diretamente a pesquisa em grade, etapa que é exponencialmente afetada pelo tamanho dos dados. Ao contrário, o algoritmo de ‘Kandukuri adaptado’ inicia a etapa de pesquisa em grade com aproximadamente 800 vezes mais características/*features*, tomando um tempo maior para processamento. Já para comparação a quantidade de processos o classificador de ‘Kandukuri adaptado’ tem etapas que têm saídas que podem ser facilmente conferidas visualmente, o que aumenta a confiança durante o processo, além de usar ferramentas matemáticas tradicionais como a FFT, que torna processo mais atraente devido a sua ampla aplicação em diversas áreas da engenharia. Seu processo também expõe suas melhoras nesse trabalho devido ao aumento em todos indicadores de qualidade de classificação, como acurácia, precisão, recall, entre outros.

Para trabalhos futuros existe a necessidade de um banco de dados com uma maior variação de desequilíbrios, a fim de estudar os limites de classificação correta para as técnicas

utilizadas e desenvolvidas. O aumento dos tipos de funcionamento também se faz interessante, acrescentar ao banco de dados simulações com falhas no sistema de controle elétrico, caixa de engrenagens, sistema de guinada (Yaw system), gerador elétrico ou conexão com a rede poderia mostrar qual é o limiar de detecção de falha para a análise de correntes utilizada nesse trabalho.

Testes com dados reais são de suma importância para validação dos algoritmos desenvolvidos nesse estudo, além da criação de um novo algoritmo com treinamento não supervisionado, com o objetivo de fazer o diagnóstico de falhas em aerogeradores em tempo real e com aplicação comercial/industrial.

REFERÊNCIAS

ABEEólica: **Boletim de fevereiro de Geração Eólica no Brasil 2020**, Disponível em < http://abeeolica.org.br/wp-content/uploads/2020/04/Infovento-15_PT.pdf> Acesso em: 1 jun. 2020

RIBRANT, J.; **Reliability performance and maintenance – a survey of failures in wind power systems**. Master's thesis, KTH School of Electrical Engineering, 2006

PACHECO F.; **Energias Renováveis: breves conceitos**, Conjuntura e Planejamento, Salvador: SEI, Out 2006

Global Energy & CO2 Status Report; **Online Annual Report** Disponível em: < https://www.iea.org/geco/renewables/> Acesso em 28 abr. 2019

FTHENAKIS, V.; KIM, H. C.; **Land use and electricity generation: A life-cycle analysis**. Renewable and Sustainable Energy Reviews. 2009

KOZA, JOHN R.; BENNETT, FORREST H.; ANDRE, DAVID; KEANE, MARTIN A.; **Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming. Artificial Intelligence in Design** 1996. Springer, Dordrecht.

BISHOP, C. M.; **Pattern Recognition and Machine Learning**, Springer. 2006

FRIEDMAN, JEROME H.; **Data Mining and Statistics: What's the connection?** Computing Science and Statistics, 1998.

LORENA, A. C.; CARVALHO, A. C.; **Uma Introdução às Support Vector Machines**, Centro de Matemática, Computação e Cognição, Universidade do ABC, 2007

V. N. VAPNIK; **The nature of Statistical learning theory**. Springer-Verlag, New York, 1995

LIMA, C.: "**Comitê de Máquinas: Uma abordagem unificada empregando máquinas de veotr de suporte**", 2004 FEEC/Unicamp

KOMER B., BERGSTRA J., ELIASMITH C.: "**Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn**" - ICML, 2014

SMITH, L.; **A tutorial on Principal Components Analysis**. Univ. of Georgia. 2002

S. BALAKRISHNAMA, A. GANAPATHIRAJU; "**Linear Discriminant Analysis - A Brief Tutorial"** - Institute for Signal and Information Processing Department of Electrical and Computer Engineering, Mississippi State University. 1998

GACI, S.; **“A new ensemble empirical mode decomposition (EEMD) denoising method for seismic signals”** - Algerian Institute of Petroleum (IAP), Boumerdès, Algeria.2016

JOLLIFFE I.; **“Principal Component Analysis”.** - International Encyclopedia of Statistical Science. Springer, Berlin, Heidelberg. 2011.

WELLING M.; **“Fisher linear discriminant analysis”** - Department of Computer Science, University of Toronto, 2005

RILLING G., FLANDRIN P., GONCALVES P.; **“On empirical mode decomposition and its algorithms”** - IEEE-EURASIP. 2003

WU Z., HUANG N.; **“Ensemble empirical mode decomposition: a noise-assisted data analysis method Advances in adaptive data analysis”**, IEEE-EURASIP .2009

ATLANTIC ENERGIAS RENOVÁVEIS. **Conheça as partes do aerogerador**. 2019. Disponível em: < http://atlanticenergias.com.br/conheca-as-partes-do-aerogerador//> Acesso em: 20 out. 2019.

SIMON, PHIL; **Too Big to Ignore: The Business Case for Big Data**. [S.l.]: Wiley, 2013

KOHAVI, R.; PROVOST, F.; **Glossary of terms. Machine Learning,** Stanford Artificial Intelligence Laboratory, 1998.

PEARSON, K.; **"On Lines and Planes of Closest Fit to Systems of Points in Space".** Philosophical Magazine, 1901.

VAN LOAN, C.; **"Computational frameworks for the fast Fourier transform"** - 1992

COOLEY, J. W.; **“An algorithm for the machine calculation of complex Fourier series,”** - Math. Comput. 19: 297-301, 1965.

S. CRUZ AND A. J. M. CARDOSO, **“Rotor cage fault diagnosis in threephase induction motors by extended Park’s vector approach”** Elect. Mach. Power Syst., 2000.

ZAREI, J., POSHTAN, J.; **“An advanced Park's vectors approach for bearing fault detection”** - 2006 IEEE International Conference

BOUZALMAT, A., KHARROUBI, J., ZARGHILI, R.; “**Comparative study of PCA, ICA, LDA using SVM classifier”** - Journal of emerging technologies in web intelligence, vol. 6, no. 1, 2014

HANLY, S. **Vibration Analysis: FFT, PSD and Spectrogram Basics.** 2016. Acessado em: <http://blog.mide.com/vibration-analysis-fft-psd-and-spectrogram>. Acesso em: 24 de Novembro de 2019.

RAO, S. **Vibrações mecânicas**. Pearson Prentice Hall, 2009.

SCHÖLKOPF, B. et al. **Learning with kernels: support vector machines, regularization, optimization, and beyond**. [S.l.]: MIT press, 2002.

Silva, J.G.B. **Aplicação da Análise de Componentes Principais no Diagnostico de Defeitos em Rolamentos Através da Assinatura Elétrica de Motores de Indução**. Itajubá: Universidade Federal de Itajubá, 2008. 98p. Dissertação Mestrado

AISEN, B.; **A Comparison of Multiclass SVM Methods**, MIT – 2006

Pedregosa et al., **Scikit-learn: Machine Learning in Python**, JMLR 12, pp. 2825-2830, 2011.

C. A. Walford, **"Wind turbine reliability: understanding and minimizing wind turbine operation and maintenance costs**" Sandia National Laboratories, Rep. SAND2006-1100, Mar. 2006.

JOHNSON, R.; WICHERN, D. et al. – "**Applied Multivariate Statistical Analysis"**, Upper Saddle River, New Jersey, 2002

YE, Jieping; JANARDAN, Ravi; LI, Qi. **Two-dimensional linear discriminant analysis.** In: Advances in neural information processing systems. 2005. p. 1569-1576.

MARSLAND, Stephen. **Machine learning: an algorithmic perspective**. Chapman and Hall/CRC, 2014.

BENNETT, Kristin P.; DEMIRIZ, Ayhan. **Semi-supervised support vector machines**. In: Advances in Neural Information processing systems. 1999. p. 368-374

CHANG, C.-C.; LIN, C.-J. **Libsvm: A library for support vector machines**. ACM transactions on intelligent systems and technology (TIST), Acm, v. 2, n. 3, p. 27, 2011.

HSU, C.-W.; LIN, C.-J. **A comparison of methods for multi-class support vector machines**. IEEE Transactions on Neural Networks, v. 13, n. 2, p. 415–425, March 2002.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding machine learning : from theory to algorithms**. [s.n.], 2014. ISBN 9781107057135 1107057132. Disponível em: <http://www.worldcat.org/search?qt=worldcat_org_all&q=9781107057135>.

KANDUKURI, SURYA TEJA et al. **Fault diagnostics for electrically operated pitch systems in offshore wind turbines**. In: Journal of Physics: Conference Series. IOP Publishing, 2016. p. 052005.

KANDUKURI, SURYA TEJA et al. A Two-Stage Fault Detection and Classification Scheme for Electrical Pitch Drives in Offshore Wind Farms Using Support Vector Machine. **IEEE Transactions on Industry Applications**, v. 55, n. 5, p. 5109-5118, 2019.

HYERS, R. W. et al. **Condition monitoring and prognosis of utility scale wind turbines**. Energy materials, v. 1, n. 3, p. 187-203, 2006.

GAYO J. B., "**Final publishable summary of results of project ReliaWind**," Tech. Rep. FP7-Energy-2007-1-RTD, 2011

WILKINSON M. et al., "**Methodology and results of the reliawind reliability field study**," in Proc. Eur. Wind Energy Conf., Warsaw, Poland, pp. 20–23, Apr. 2010

SALEH S. A. e MOLONEY C. R., "**Development and testing of wavelet packet transform-based detector for ice accretion on wind turbines**" in Proc. IEEE Digital Signal Process. Workshop IEEE Signal Process. Education Workshop, Jan. 2011, pp. 72–77.

YANG, WENXIAN et al. **Wind turbine condition monitoring: technical and commercial challenges**. Wind Energy, v. 17, n. 5, p. 673-693, 2014.

TANG, J.; ALELYANI, S.; LIU, H.. **Feature selection for classification: A review**. Data classification: Algorithms and applications, p. 37, 2014.

LAOUTI, N.; SHEIBAT-OTHMAN, N.; OTHMAN, S.. **Support vector machines for fault detection in wind turbines**. IFAC Proceedings Volumes, v. 44, n. 1, p. 7067-7072, 2011.

SANTOS, P. et al. **An SVM-based solution for fault detection in wind turbines**. Sensors, v. 15, n. 3, p. 5627-5648, 2015.

BENBOUZID, M. H. E..; KLIMAN, G. **What stator current processing-based technique to use for induction motor rotor faults diagnosis?** IEEE Transactions on Energy Conversion, v. 18, n. 2, p. 238-244, 2003.

BENBOUZID, M. E. H. **A review of induction motors signature analysis as a medium for faults detection**. IEEE transactions on industrial electronics, v. 47, n. 5, p. 984-993, 2000.

NANDI, S.; TOLIYAT, H. A.; LI, X.. **Condition monitoring and fault diagnosis of electrical motors—A review**. IEEE transactions on energy conversion, v. 20, n. 4, p. 719-729, 2005.

JONKMAN, B. J.; KILCHER, L. **TurbSim user's guide**. [S.l.], 2006

KERRES, B.; FISCHER, K.; MADLENER, R.. Economic evaluation of maintenance strategies for wind turbines: a stochastic analysis. **IET renewable power generation**, v. 9, n. 7, p. 766-774, 2015.

VIRTANEN, PAULI et al. **SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature methods**, v. 17, n. 3, p. 261-272, 2020.

MALIK, H.; MISHRA, S. **Application of probabilistic neural network in fault diagnosis of wind turbine using FAST, TurbSim and Simulink**. Procedia Computer Science, v. 58, p. 186-193, 2015.

MALIK, H.; MISHRA, S. **Artificial neural network and empirical mode decomposition based imbalance fault diagnosis of wind turbine using TurbSim, FAST and Simulink**. IET Renewable Power Generation, v. 11, n. 6, p. 889-902, 2016.

YU, H.; VAIDYA, J.; JIANG, X. **Privacy-preserving svm classification on vertically partitioned data**. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Berlin, Heidelberg, 2006. p. 647-656.

RAKOTOMAMONJY, A. **Variable selection using SVM-based criteria**. Journal of machine learning research, v. 3, n. Mar, p. 1357-1370, 2003.

WENYI, Liu et al. **Wind turbine fault diagnosis method based on diagonal spectrum and clustering binary tree SVM**. Renewable Energy, v. 50, p. 1-6, 2013.

ZENG, Jianwu et al. **Wind turbine fault detection and isolation using support vector machine and a residual-based method**. In: 2013 American Control Conference. IEEE, 2013. p. 3661-3666.

SHIN, Jun-Hyun; LEE, Yun-Seong; KIM, Jin-O. **Fault prediction of wind turbine by using the SVM method.** In: 2014 International Conference on Information Science, Electronics and Electrical Engineering. IEEE, 2014. p. 1923-1926.

ZHAO, Rui et al. **Wind turbine fault prediction using soft label SVM**. In: 2016 23rd International conference on pattern recognition (ICPR). IEEE, 2016. p. 3192-3197.

RILLING, Gabriel et al. **On empirical mode decomposition and its algorithms**. In: IEEE-EURASIP workshop on nonlinear signal and image processing. NSIP-03, Grado (I), 2003. p. 8-11.

GACI, Said. **A new ensemble empirical mode decomposition (EEMD) denoising method for seismic signals**. Energy Procedia, v. 97, n. November, p. 84-91, 2016.

FERREIRA, Luiz Eduardo Soares; PORSANI, Milton José; DA SILVA, Michelângelo Gomes. **Aplicação do método de decomposição em modos empíricos na atenuação do ruído de rolamento**. In: IV Simpósio Brasileiro de Geofísica. European Association of Geoscientists & Engineers, 2010. p. cp-197-00035.

AN, Xueli et al. **Bearing fault diagnosis based on ITD and LS-SVM for wind turbine**. Electric Power Automation Equipment, v. 31, n. 9, p. 10-13, 2011.

LEAHY, Kevin et al. **Diagnosing wind turbine faults using machine learning techniques applied to operational data**. In: 2016 IEEE International Conference on Prognostics and Health Management (ICPHM). IEEE, 2016. p. 1-8.

HU, Chun-zhi et al. **On the use of EEMD and SVM based approach for bearing fault diagnosis of wind turbine gearbox**. In: 2016 Chinese Control and Decision Conference (CCDC). IEEE, 2016. p. 3472-3477.

SAARI, Juhamatti et al. **Detection and identification of windmill bearing faults using a one-class support vector machine (SVM)**. Measurement, v. 137, p. 287-301, 2019.

CHUNHUA, Zhao; HAIJING, Dong; XIANYOU, Zhong. **SVM parmeter optimization in fault diagnosis for wind power gear box**. China Mechanical Engineering, v. 26, n. 16, p. 2222-2225, 2015.

LU, Dingguo; QIAO, Wei. **Adaptive feature extraction and SVM classification for real-time fault diagnosis of drivetrain gearboxes**. In: 2013 IEEE Energy Conversion Congress and Exposition. IEEE, 2013. p. 3934-3940.

6. APÊNDICES

6.1 APENDICE A

Algoritmo de Kandukuri adaptado:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.preprocessing import MinMaxScaler

def plot_confusion_matrix(y_true, y_pred,classes,
                          normalize=True,
                          title=None,
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if not title:
        if normalize:
            title = 'Normalized confusion matrix'
        else:
            title = 'Confusion matrix, without normalization'

    # Compute confusion matrix
    cm = confusion_matrix(y_true, y_pred)
```

```
#classes = classes[unique_labels(y_true, y_pred)]
# Only use the labels that appear in the data
if normalize:
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    print("Normalized confusion matrix")
else:
    print('Confusion matrix, without normalization')

print(cm)

fig, ax = plt.subplots()
im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
ax.figure.colorbar(im, ax=ax)
# We want to show all ticks...
ax.set(xticks=np.arange(cm.shape[1]),
       yticks=np.arange(cm.shape[0]),
       # ... and label them with the respective list entries
       xticklabels=classes, yticklabels=classes,
       title=title,
       ylabel='True label',
       xlabel='Predicted label')

# Rotate the tick labels and set their alignment.
plt.setp(ax.get_xticklabels(), rotation=45, ha="right",fontsize=16,
         rotation_mode="anchor")

# Loop over data dimensions and create text annotations.
fmt = '.2f' if normalize else 'd'
thresh = cm.max() / 2.
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(j, i, format(cm[i, j], fmt),
                ha="center", va="center", fontsize=16,
```

```
                color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
    return ax

x=pd.read_csv(IP 5s Resumido com stats.csv',index_col=0, sep=',').T
print('Dados lidos')
x = x.to_numpy()
y = np.ones((2100), dtype=np.int32)
y[   :300] = 100      #100
y[300:600] = 2     #
y[600:900] = 3     #
y[900:1200] = 4        #
y[1200:1500] = 97      #100
y[1500:1800] = 102     #
y[1800:] = 105
print('Classes adicionadas')

scaler = MinMaxScaler()  # Default behavior is to scale to [0,1]
x = scaler.fit_transform(x)
print('Dados Normalizados')
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=12)
#DIVIDE DADOS
print('Dados divididos')
#del x, y
print('x e y deletados')
# defining parameter range
param_grid = {'C': [10,1,1e3, 5e3, 1e4, 5e4, 1e5, 5e5],
              'gamma': [0.001, 0.005, 0.01, 0.1,1,10],
              'kernel': ['rbf','sigmoid','poly']}
#param_grid = {'C': [10,1,1e3, 5e3, 1e4, 5e4, 1e5],'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01,
0.1,1,10], }
grid = GridSearchCV(SVC(),param_grid, cv=10,verbose=1,return_train_score=True)
print('Grid Search')
```

```
grid.fit(X_train, y_train)
print('Grid fit')
y_pred = grid.predict(X_test)
# print best parameter after tuning
print(grid.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)

##pickle.dump(A, open('NOME DO ARQUIVO.sav', 'wb'))
print(classification_report(y_test, y_pred))
acertos = accuracy_score(y_test, y_pred, normalize=True, sample_weight=None)
T_test = len(y_test)
estimator = grid.best_estimator_
C = grid.best_estimator_.C
gamma = grid.best_estimator_.gamma
kernel =  grid.best_estimator_.kernel
print(acertos,'%')

class_names = ['100% Mass','2 graus','3 graus','4 graus','97% Mass','102% Mass','105% Mass']
plot_confusion_matrix(y_test, y_pred,classes = class_names,title='Confusion matrix')
t1 = ("Kernel =  "+str(kernel))
t2 = ("C =  "+str(C))
t3 = ("Gamma =  "+str(gamma))
t4 = ("Teste =  "+str(T_test))
plt.text(6.5, 7, t1, fontsize=11, style='oblique', ha='left', va='top', wrap=True)
plt.text(6.5, 7.2, t2, fontsize=11, style='oblique', ha='left', va='top', wrap=True)
plt.text(6.5, 7.4, t3, fontsize=11, style='oblique', ha='left', va='top', wrap=True)
plt.text(6.5, 7.6, t4, fontsize=11, style='oblique', ha='left', va='top', wrap=True)
plt.rcParams['figure.figsize'] = (9,7)
#plt.tight_layout()
plt.tick_params(axis = 'y', labelsize = 11)
plt.tick_params(axis = 'x', labelsize = 11)
```

```
plt.savefig(str(Acuracia)+' IP-C '+str(C)+'-G '+str(gamma)+'-'+str(kernel)+'.png', dpi = 470)
```

6.2 APENDICE B

Algoritmo de Malik adaptado:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import confusion_matrix, classification_report

def plot_confusion_matrix(y_true, y_pred,classes,
                          normalize=True,
                          title=None,
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if not title:
        if normalize:
            title = 'Normalized confusion matrix'
        else:
            title = 'Confusion matrix, without normalization'
```

```
# Compute confusion matrix
cm = confusion_matrix(y_true, y_pred)
#classes = classes[unique_labels(y_true, y_pred)]
# Only use the labels that appear in the data
if normalize:
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    print("Normalized confusion matrix")
else:
    print('Confusion matrix, without normalization')

print(cm)

fig, ax = plt.subplots()
im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
ax.figure.colorbar(im, ax=ax)
# We want to show all ticks...
ax.set(xticks=np.arange(cm.shape[1]),
       yticks=np.arange(cm.shape[0]),
       # ... and label them with the respective list entries
       xticklabels=classes, yticklabels=classes,
       title=title,
       ylabel='True label',
       xlabel='Predicted label')

# Rotate the tick labels and set their alignment.
plt.setp(ax.get_xticklabels(), rotation=45, ha="right",fontsize=16,
         rotation_mode="anchor")

# Loop over data dimensions and create text annotations.
fmt = '.2f' if normalize else 'd'
thresh = cm.max() / 2.
for i in range(cm.shape[0]):
```

```
        for j in range(cm.shape[1]):
            ax.text(j, i, format(cm[i, j], fmt),
                    ha="center", va="center", fontsize=16,
                    color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
    return ax

data1 = pd.read_csv(r'Ia_Dados_Tempo_Normal.csv',index_col=0, sep=',').T
data2 = pd.read_csv(r'Ia_Dados_Tempo_2 graus.csv',index_col=0, sep=',').T
data3 = pd.read_csv(r'Ia_Dados_Tempo_3 graus.csv',index_col=0, sep=',').T
data4 = pd.read_csv(r'Ia_Dados_Tempo_4 graus.csv',index_col=0, sep=',').T
data5 = pd.read_csv(r'Ia_Dados_Tempo_97% Imbalanced.csv',index_col=0, sep=',').T
data6 = pd.read_csv(r'Ia_Dados_Tempo_102% Imbalanced.csv',index_col=0, sep=',').T
data7 = pd.read_csv(r'Ia_Dados_Tempo_105% Imbalanced.csv',index_col=0, sep=',').T
print('Dados lidos')
frames = [data1,data2,data3,data4,data5,data6,data7]
datatotal = pd.concat(frames)
print('Dados contatenados')
del data1,data2,data3,data4,data5,data6,data7,frames
print('Dados não utilizados deletados')
datatotal.insert(119999, "Flag", "Any")
print('Coluna inserida')
datatotal.iloc[    : 301, 119999] = 100
datatotal.iloc[ 301: 602, 119999] = 2
datatotal.iloc[ 602: 903, 119999] = 3
datatotal.iloc[ 903:1204, 119999] = 4
datatotal.iloc[1204:1505, 119999] = 97
datatotal.iloc[1505:1806, 119999] = 102
datatotal.iloc[1806:2107, 119999] = 105
print('Classes adicionadas')
x = datatotal.iloc[:,:100000]
y = datatotal.iloc[:,119999]
del datatotal
```

```
print('Dados não utilizados deletados 2')
scaler = MinMaxScaler()  # Default behavior is to scale to [0,1]
x = scaler.fit_transform(x)
print('Dados Normalizados')
PCA1 = len(x[0])
print('Tamanho de X: ',PCA1)
x = PCA(0.999999).fit_transform(x)
PCA1 = len(x[0])
print('Tamanho de X_PCA: ',PCA1)

#SAVE PCA
##x = pd.DataFrame(x)
##x.to_csv('Ia PCA - '+str(PCA1)+'.csv')
##print('Ia PCA Salvo')
##x = x.to_numpy()

lda = LinearDiscriminantAnalysis(n_components=6)
LDA1 = lda.n_components
x = lda.fit(x, y).transform(x)
print('LDA feito')
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=19) #DIVIDE DADOS
del x,y
print('Dados divididos')
#param_grid = {'C': [10,1,1e3, 5e3, 1e4, 5e4, 1e5],'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1,1,10], }
param_grid = {'C': [0.001, 0.005, 0.01, 0.1,1,10],
              'gamma': [0.00001,0.0001,0.001, 0.005, 0.01, 0.1,],
              'kernel': ['rbf','poly','linear']}
grid = GridSearchCV(SVC(),param_grid,n_jobs=-1, cv=5,verbose=1,return_train_score=True)
print('Grid Search feito')
grid = grid.fit(X_train, y_train)
```

```
print('SVM feito')
y_pred = grid.predict(X_test)
# print best parameter after tuning
print(grid.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)

##pickle.dump(A, open('NOME DO ARQUIVO.sav', 'wb'))
print(classification_report(y_test, y_pred))

acertos = accuracy_score(y_test, y_pred, normalize=True, sample_weight=None)
T_test = len(y_test)

C1 = grid.best_estimator_.C
gamma1 = grid.best_estimator_.gamma
kernel1 = grid.best_estimator_.kernel

print(acertos,'%')

class_names = ['100% Mass','2 graus','3 graus','4 graus','97% Mass','102% Mass','105% Mass']
plot_confusion_matrix(y_test, y_pred,classes = class_names,title='Confusion matrix')
t1 = ("Kernel =  "+str(kernel1))
t2 = ("C =  "+str(C1))
t3 = ("Gamma =  "+str(gamma1))
t4 = ("PCA =  "+str(PCA1))
t5 = ("LDA =  "+str(LDA1))
t6 = ("Teste =  "+str(T_test))
plt.text(6.5, 7, t1, fontsize=11, style='oblique', ha='left', va='top', wrap=True)
plt.text(6.5, 7.2, t2, fontsize=11, style='oblique', ha='left', va='top', wrap=True)
plt.text(6.5, 7.4, t3, fontsize=11, style='oblique', ha='left', va='top', wrap=True)
plt.text(6.5, 7.6, t4, fontsize=11, style='oblique', ha='left', va='top', wrap=True)
plt.text(6.5, 7.8, t5, fontsize=11, style='oblique', ha='left', va='top', wrap=True)
```

```
plt.text(6.5, 8, t6, fontsize=11, style='oblique', ha='left', va='top', wrap=True)
plt.rcParams['figure.figsize'] = (9,7)
plt.tick_params(axis = 'y', labelsize = 16)
plt.tick_params(axis = 'x', labelsize = 16)
plt.savefig(str(Acuracia)+' MALIK '+str(PCA1)+' PCA '+str(LDA1)+' LDA-C '+str(C1)+'-G
'+str(gamma1)+'-'+str(kernel1)+'.png', dpi = 470)
```