UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Júlio Oliveira Schmidt

UTILIZAÇÃO DE TÉCNICAS DE DEEP LEARNING PARA PREDIÇÃO DE DESBALANCEAMENTO DE MASSA EM ROTORES DE TURBINAS EÓLICAS

Santa Maria, RS
2022

UTILIZAÇÃO DE TÉCNICAS DE DEEP LEARNING PARA PREDIÇÃO DE DESBALANCEAMENTO DE MASSA EM ROTORES DE TURBINAS EÓLICAS

Trabalho de conclusão de curso, apresentado ao Curso de Engenharia Elétrica, da Universidade Federal de Santa Maria (UFSM, RS) – Campus Santa Maria, como requisito parcial para obtenção do título de **Engenheiro Eletricista**.

Orientador: Prof. Dr. Daniel Fernando Tello Gamarra

Santa Maria, RS
2022

UTILIZAÇÃO DE TÉCNICAS DE DEEP LEARNING PARA PREDIÇÃO DE DESBALANCEAMENTO DE MASSA EM ROTORES DE TURBINAS EÓLICAS

Trabalho de conclusão de curso, apresentado ao Curso de Engenharia Elétrica, da Universidade Federal de Santa Maria (UFSM, RS) – Campus Santa Maria, como requisito parcial para obtenção do título de **Engenheiro Eletricista**.

Aprovado em 15 de agosto de 2022:

_____

Prof. Daniel Fernando Tello Gamarra (UFSM)
(Presidente/Orientador)

_____

Prof. Frederico Menine Schaf (UFSM)

_____

Prof. Cleo Zanella Billa (FURG)

Santa Maria, RS
2022

DEDICATÓRIA

Aos meus pais, Arlei e Mara Rosany, o amor de vocês me trouxe força nos meus momentos mais difíceis, tenho muita sorte de tê-los em minha vida, amo vocês mais que tudo.

E à minha vó Vicentina (in memorian), que me amou até se último instante de vida, te amo para sempre.

RESUMO

UTILIZAÇÃO DE TÉCNICAS DE DEEP LEARNING PARA PREDIÇÃO DE DESBALANCEAMENTO DE MASSA EM ROTORES DE TURBINAS EÓLICAS

AUTOR: Júlio Oliveira Schmidt
ORIENTADOR: Daniel Fernando Tello Gamarra

Uma fonte de energia limpa, renovável e permanentemente abundante, a energia elétrica eólica vem sendo cada vez mais buscada em todo o mundo, tanto no ramo acadêmico quanto no ramo empresarial, por seu baixíssimo impacto ambiental, tem se tornado cada vez mais uma excelente opção de produção de energia alternativa aos modos convencionais de geração energética. O Brasil possui um enorme potencial de expansão graças à sua geografia muito propícia ao aproveitamento dos ventos, na região Nordeste, por exemplo, tem capacidade de produzir até 75GW de energia eólica, equivalente à metade de toda energia produzida pelo país inteiro. Segundo a Abeeólica, o Brasil abastece 11,5% da matriz energética total do país, chegando à 20% em dias de alta produção. Apesar de toda as suas vantagens, ainda deixa muito a desejar em relação à sua eficiência, muitas pesquisas vem sido realizadas impulsionadas pelo grande aumento do interesse nesse tipo de produção energética. Nesse trabalho busco acrescentar à essas pesquisas realizadas no ramo um estudo desenvolvido com inteligência artificial usando duas redes neurais diferentes: uma rede neural convolucional (CNN) e uma memória de curto e longo prazo (LSTM), para uma averiguação no desbalanceamento de massa de lâminas de rotores de aerogeradores. O intuito final desse trabalho é trazer uma maneira capaz de possibilitar uma classificação categórica e competente de desbalaceamento de massa na velocidade do rotor, para o melhor funcionamento do sistema, agindo de forma mais precisa, constante e menos nociva à operação.

Palavras-chave: Redes Neurais Convolucionais, Turbinas Eólicas, Detecção de Falhas, LSTM

ABSTRACT

USE OF DEEP LEARNING TECHNIQUES TO PREDICT MASS IMBALANCE IN WIND TURBINE ROTORS

AUTHOR: Júlio Oliveira Schmidt
ADVISOR: Daniel Fernando Tello Gamarra

A source of clean, renewable and permanently abundant energy, wind electric energy has been increasingly sought after all over the world, both in academia and in the business sector, due to its very low environmental impact, it has become increasingly an excellent option. alternative energy production to conventional modes of energy generation. Brazil has an enormous potential for expansion thanks to its geography that is very favorable to the use of winds, in the Northeast region, for example, it has the capacity to produce up to 75GW of wind energy, equivalent to half of all energy produced by the entire country. According to Abeeólica, Brazil supplies 11.5% of the country's total energy matrix, reaching 20% on days of high production. Despite all its advantages, it still leaves a lot to be desired in relation to its efficiency, many researches have been carried out driven by the great increase in interest in this type of energy production. In this work I seek to add to these researches carried out in the field a study developed with artificial intelligence using two different neural networks: a convolutional neural network (CNN) and a short and long term memory (LSTM), for an investigation of the blade mass imbalance. of wind turbine rotors. The final purpose of this work is to bring a way capable of enabling a categorical and competent classification of mass unbalance in the rotor speed, for the better functioning of the system, acting in a more precise, constant and less harmful way to the operation.

Keywords: Convolutional Neural Network, Wind Energy, Failure detection, Recurrent Plot, Markov Transformation

LISTA DE FIGURAS

LISTA DE TABELAS

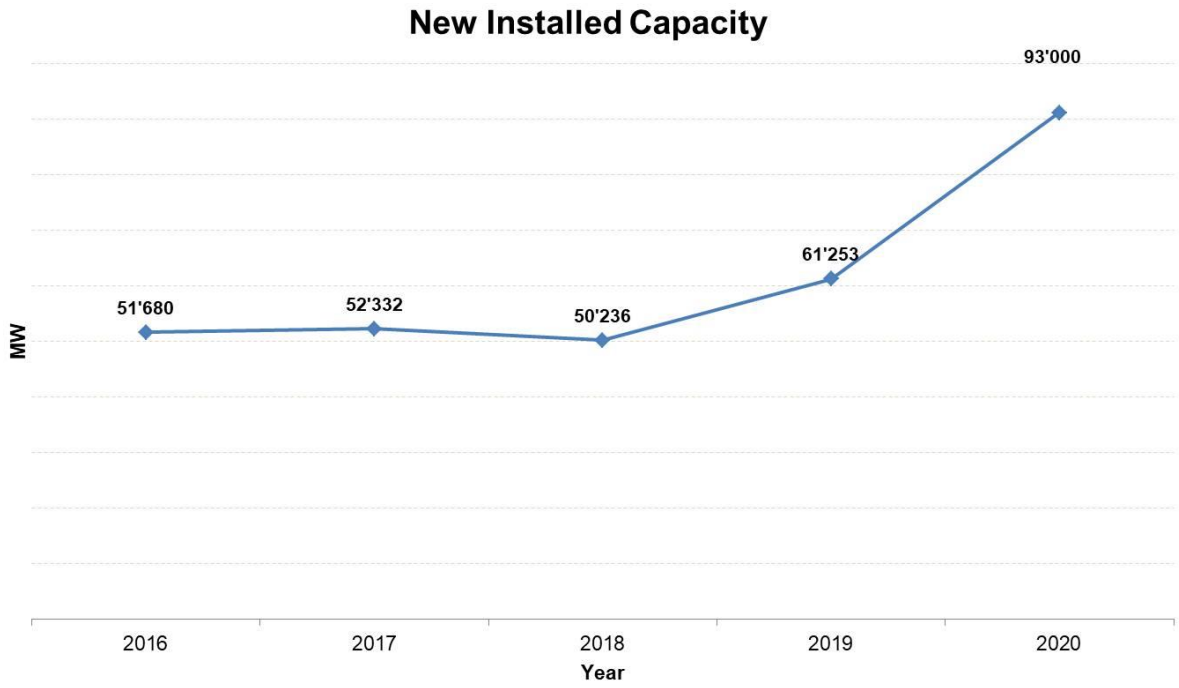
SUMÁRIO

1. INTRODUÇÃO

1.1 MOTIVAÇÃO

2020 foi mais um ano recorde para as energias renováveis, apesar da pandemia de Covid-19 e do aumento dos custos das matérias-primas em todo o mundo, segundo a Agência Internacional de Energia (AIE). Cerca de 290 GW de nova capacidade de geração de energia renovável, principalmente na forma de turbinas eólicas e painéis solares, foram instalados em todo o mundo este ano, batendo o recorde anterior do ano de 2020. De acordo com as tendências atuais, a capacidade de geração de energia renovável excederá a de combustíveis fósseis e energia nuclear combinadas até 2026.

Figura 1 – Evolução da Capacidade de energia eólica mundial



Fonte: World Wind Energy Association (2021).

O Brasil tem apresentado uma forte contribuição energética nesse aspecto, a produção de energia eólica aumenta gradativamente no país, principalmente no Nordeste brasileiro, o setor acumulou recorde de geração de energia. Isso tem ajudado a reduzir as consequências da crise hídrica que o país enfrenta atualmente — devido às chuvas secas — e também a estimular a recuperação econômica, gerando empregos.

O Brasil ocupa o sétimo lugar no ranking global do Global Wind Energy Council (GWEC) com 19 GW de capacidade instalada. Para se ter uma ideia do crescimento, o setor

tinha menos de 1 GW instalado no país há uma década e atualmente é a segunda maior fonte de energia do país, respondendo por 10% da matriz elétrica.

De acordo com a Associação Brasileira de Energia Eólica (ABEEólica) , a expectativa é que até 2024 o Brasil tenha pelo menos 30 GW de capacidade eólica instalada, considerando apenas os leilões já realizados e os contratos assinados. Novos leilões adicionarão mais capacidade instalada nos próximos anos.

Figura 2 – Evolução da capacidade instalada em MW no Brasil, de 2005 a 2024.

Fonte: ANEEL (2021). Imagem cortada do relatório de julho de 2021.

Com a rápida expansão da energia eólica, uma importante tarefa de manutenção e controle de qualidade dos aerogeradores deve ser assegurada. As turbinas eólicas requerem reparos com bastante frequência, e isso pode ser bastante caro (Carroll J, 2016), observaram que são necessários em média, 6,2 pequenos reparos, 1,1 grandes reparos e 0,43 grandes substituições por turbina por ano.

Os mecanismos de falha de turbinas eólicas dependem das condições climáticas locais, tipos de carregamento e efeitos ambientais. Baixas temperaturas aumentam a fragilidade dos componentes, enquanto variações de temperatura podem causar fadiga térmica. Raios, gelo e ventos fortes podem aumentar drasticamente a taxa de falhas das turbinas eólicas.

Espera-se que as altas variações das condições climáticas, típicas das regiões indianas, tenham um forte efeito na confiabilidade das turbinas eólicas. Altas temperaturas e umidade

no sul da Índia, fortes variações de temperatura no norte da Índia, poeira no Rajastan e padrões climáticos de monções podem atuar como fatores adicionais para a degradação das turbinas eólicas.

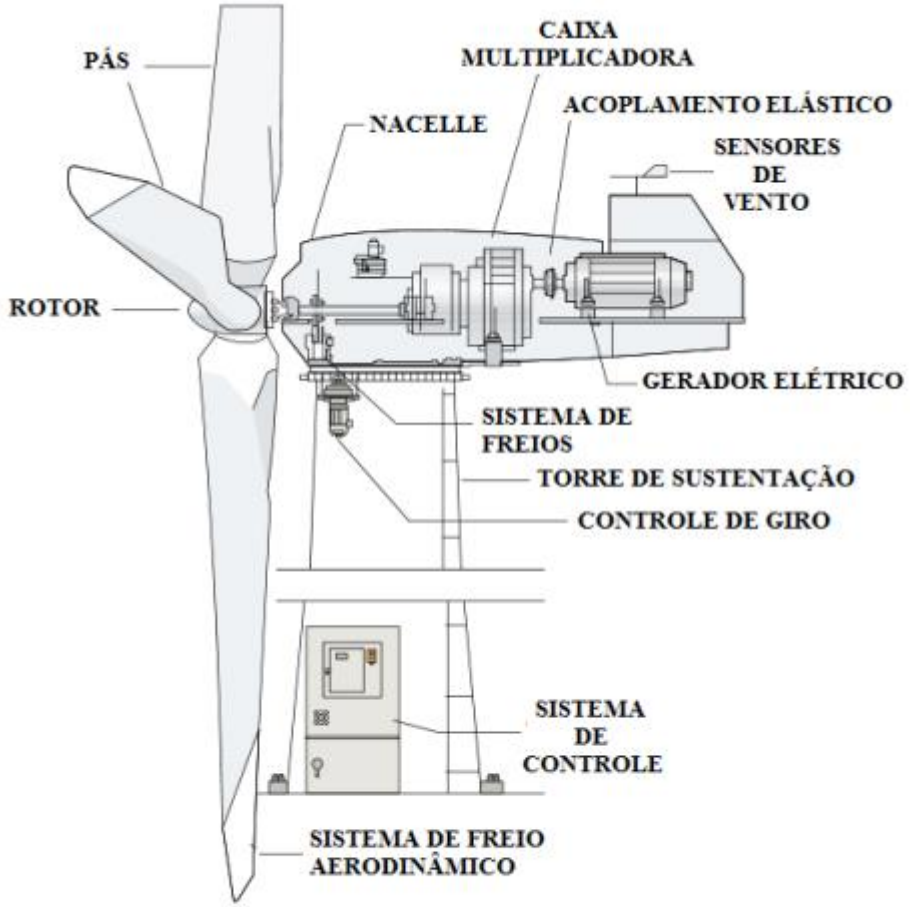
Os mecanismos de danos típicos observados nas pás de turbinas eólicas incluem descolamento da pele/adesivo, falha da junta adesiva, descolamento do sanduíche, delaminação, divisão ao longo das fibras e rachaduras no gelcoat. Em (Robinson CME 2013), as falhas da lâmina são classificadas em falha de conexão de raiz, flambagem ou separação estrutural catastrófica, borda de ataque, borda de fuga ou outra separação de ligação, dano por raio, erosão e falha no dispositivo aerodinâmico externo. As pás das turbinas eólicas consistem em laminados, espumas, revestimentos, estruturas sanduíche e camadas adesivas. Com vista a esses elementos, os mecanismos de danos das pás de turbinas eólicas podem ser classificados como: (a) danos na superfície (degradação do revestimento), (b) danos na resina polimérica e/ou na interface e (c) danos nos elementos estruturais (com fibras estruturais quebradas) . Os danos na superfície podem ser causados por erosão (erosão pela chuva, areia e granizo) ou impactos de pequenos objetos. Reduz o desempenho aerodinâmico das pás e a geração de energia. Não impede o funcionamento da turbina eólica, mas os defeitos crescem, se desenvolvem e levam a danos estruturais.

Logo, podemos notar que deve haver uma preocupação com o funcionamento das turbinas, com as melhorias do seu sistema de monitoramento sendo constante. Á partir deste conceito que vem a ideia desse artigo, trazer um algoritmo classificatório referente aos desbalanços de massa, sem a necessidade de sensores adicionais. Espera-se que os resultados apresentados nesse trabalho, se provem úteis à resolução do problema de desbalanço, ajudando assim, o sistema detectar falhas mais facilmente e realizar manutenções mais precisas.

1.2 AEROGERADORES E INTELIGENCIA ARTIFICIAL NA SUA MANUTENÇÃO

O aerogerador é um item essencial em qualquer turbina eólica e o responsável por converter a força mecânica do eixo em energia elétrica de corrente alternada. Os seus dispositivos de mais destaques são: o Rotor, as pás, e o gerador elétrico. Todos esses dispositivos estão ilustrados na figura 3.

Figura 3 – Modelo de Aerogerador



Fonte: (CBEE/UFPE, 2000).

Fonte: Portal Solar. Disponivel em: https://www.portalsolar.com.br/energia-solar-x-energia-eolica-precos.html (Acessado em Maio/2022)

Devido ao seu grande tamanho e estar localizado em regiões de climas hostis, com o passar do tempo muitos dispositivos passam a manifestar defeitos ou desgastes, de acordo com dois estudos da Deutsche WindGuard de 2013 e 2015, os custos de manutenção e reparação aumentam 40% após dez anos de operação, de 10,5 para 14,7 €/MWh e, portanto,

representam 44% - 55% dos custos operacionais totais. Com o propósito de reduzir as despesas e melhorar a eficácia das manutenções, pesquisas sobre reconhecimento de falhas aderindo a métodos de inteligência artificial estão gradativamente sendo mais utilizados.

Em um estudo recente, tivemos o uso de dados de 31 turbinas em Taiwan para adotar procedimentos de Machine Learning e assim, apresentando resultados acima de 90% de precisão na identificação de desacertos nessas turbinas (HSU et al, 2020).

Em outra pesquisa, foram usados dados SCADA e metodologias de Recurrent Neural Network para supervisionar a situação de operação do sistema e a inteireza de seus elementos (SUN;SUN, 2018).

Com os mesmos valores de desbalanceamento usados nesta pesquisa, foi feito um prognóstico de desbalanceamento de rotor utilizando Máquina de vetores de suporte (SVM) e Power Spectral Density (HUBNER. 2021).

Esta pesquisa se une a esses empenhos empregando metodologias inventivas de pré-processamento de dados com o objetivo de uma melhor precisão na detecção de falhas de turbinas eólicas.

1.3 OBJETIVOS DO TRABALHO

Para apresentar os resultados obtidos durante o estudo, devemos antes de tudo explicar o propósito com o estudo objetivado. Esta parte busca transparecer o objetivo geral e objetivos específicos que foram explorados durante o trabalho.

1.3.1 Objetivo Geral

O objetivo geral é trazer soluções a respeito do desbalanço de massa em Turbinas Eólicas.

1.3.2 Objetivos Específicos

Tendo em vista o objetivo geral apresentado anteriormente, definimos como objetivos específicos do projeto:

- Utilizar o cálculo do Power Spectral Density para o pré-processamento do banco de dados utilizado;
- Utilizar algoritmo SMOTE para resolver o desbalanço do numero de amostras do banco de dados;

- Converter dados em imagem utilizando as técnicas de Recurrent Plot e Markov Transformation;
- Realizar extração de características para usar como complemento do banco de dados visando melhorar a acurácia;
- Aplicar as imagens a Redes Neurais convolucionais;
- Aplicar os dados a rede LSTM.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado em seis capítulos. Neste segmento está apresentada a divisão do estudo dentro desses capítulos, evidenciando o que cada um deles discorre. Até este momento discutimos os tópicos relativos ao capítulo 1 do trabalho, no qual foi realizada a introdução discutindo a motivação para esse estudo, a exposição dos estudos alusivos e o estabelecimento dos objetivos pretendidos.

No capítulo 2 serão retratados os instrumentos usados durante o processo de realização do trabalho, incluindo o algoritmo de classificação, o algoritmo de pré-processamento, o banco de dados e seu desenvolvimento.

Durante do capítulo 3 serão discutidos os argumentos da pesquisa e o emprego de certas apurações numéricas. No capítulo 4 será discutido o tratamento preterido para a análise dos dados, além de apresentar como serão realizadas as relações comparativas de resultados obtidos com os índices classificatórios. No capítulo 5 serão discutidos os resultados obtidos do estudo e no capítulo 6 será feita a conclusão.

Nos capítulos finais haverá os apêndices do trabalho, com alguns dos algoritmos implementados durante o processo de desenvolvimento.

2. REFERENCIAL TEÓRICO

Ao longo deste capítulo será realizado um desenvolvimento teórico de todos os materiais utilizados. Ele será divido em 3 seções, sendo elas: Power Spectral Density, Banco de dados Desbalanceado e Redes Neurais.

2.1 POWER SPECTRAL DENSITY

O Power Spectral Density foi essencial durante a pesquisa pelo fato de que, como observaremos no Capítulo 3, o banco de dados usado é formado por 240000 features de sinais temporais. Para fazer o diagnóstico de falhas com tamanha informação seria necessário um amplo empenho computacional, foram precisos instrumentos para pré-processar esse sinal.

O PSD, de acordo com sua definição técnica, é a variação de energia que ocorre dentro de um sinal vibracional, medida como frequência por unidade de massa. Em outras palavras, para cada frequência, a função de densidade espectral mostra se a energia presente é maior ou menor. A densidade de potência espectral é, portanto, uma função matemática da frequência ou do comprimento de onda. Na representação de frequência, tem a dimensão potência · tempo (por exemplo, em unidades de watts / hertz ou dBm /Hz). Na representação do comprimento de onda, tem a dimensão potência / comprimento . A integral da densidade de potência espectral sobre todas as frequências ou comprimentos de onda dá a potência total da radiação ou sinal. Em termos numéricos, o PSD é a energia do sinal, dividida pelo tempo e depois normatizada por cada frequência. Tanto a energia total de um sinal no tempo quanto a energia total do sinal na frequência podem ser extraídas pela mesma expressão:

$$E = \int_{-\infty}^{\infty} |x(t)|^2 \, dt = \int_{-\infty}^{\infty} |X(f)|^2 \, df \qquad (1)$$

Aderindo frequência angular a equação de energia, obtemos:

$$E = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X(\omega)|^2 \, d\omega \qquad (2)$$

Logo, pela relação de tempo e potência, apresentada anteriormente, podemos concluir que:

$$P = \frac{1}{2\pi} \int_{-\infty}^{\infty} \lim_{T \to \infty} \frac{|X(\omega)|^2}{T} \, d\omega \qquad (3)$$

O PSD é definido como o valor de dentro da integral, tanto na energia, quanto na potência, essa definição é válida, sendo assim:

$$PSD = \lim_{T \to \infty} \frac{|X(\omega)|^2}{T} \qquad (4)$$

2.2 BANCO DE DADOS DESBALANCEADO

Dados desequilibrados referem-se àqueles tipos de conjuntos de dados em que a classe-alvo tem uma distribuição desigual de observações, ou seja, um rótulo de classe tem um número muito alto de observações e o outro tem um número muito baixo de observações.

Por essa diferença nos tamanhos dos dados, suas classificações também ficam desequilibradas, representando um desafio para a modelagem preditiva, pois a maioria dos algoritmos de aprendizado de máquina usados para classificação foram projetados com base na suposição de um número igual de exemplos para cada classe. Isso resulta em modelos com desempenho preditivo ruim, especificamente para a classe minoritária. Isso é um problema porque, normalmente, a classe minoritária é mais importante e, portanto, o problema é mais sensível a erros de classificação para a classe minoritária do que para a classe majoritária.

Por exemplo, podemos coletar medidas de flores e ter 80 exemplos de uma espécie de flor e 20 exemplos de uma segunda espécie de flor, e apenas esses exemplos compõem nosso conjunto de dados de treinamento. Isso representa um exemplo de um problema de classificação desequilibrada. Um desequilíbrio ocorre quando uma ou mais classes têm proporções muito baixas nos dados de treinamento em comparação com as outras classes (— Página 419, Modelagem Preditiva Aplicada , 2013.).

Nossa situação não é tão seriamente difusiva quanto os vários exemplos que poderiam ter sido mencionados, mas de qualquer forma, a classe majoritária tem aproximadamente três vezes maior que a classe minoritária, como mostrado na figura 4. Todavia, essa divergência não é exclusiva somente entre duas classes, mas sim, entre todas. Totalizando, há seis classes que são minimamente duas vezes maiores que as cinco demais. Esse desbalanceamento de tamanho de classes é capaz de trazer vários empecilhos para um classificador alimentado por esses dados.

Figura 4 – Comparação do Tamanho das Classes

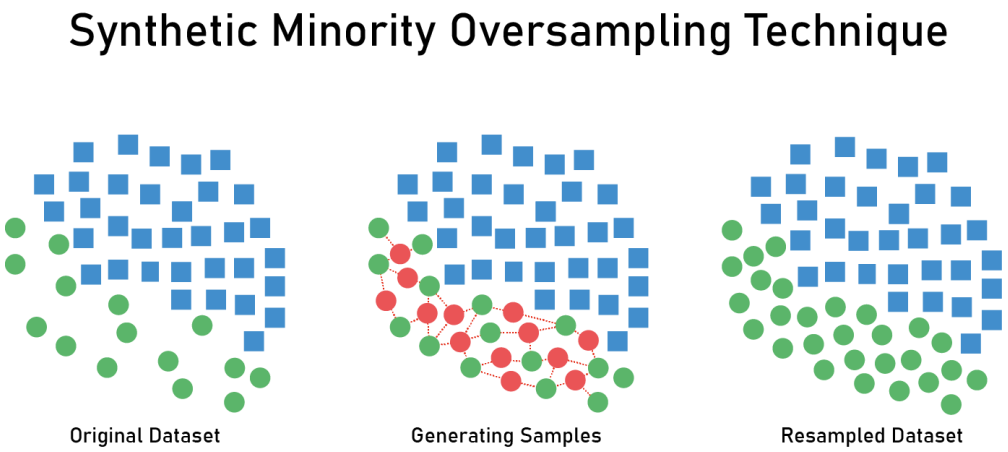
Fonte: Lucas França Aires.

.

2.3 SMOTE

Proposto em (N, V CHAWLA et al, 2002), o SMOTE funciona selecionando exemplos próximos no espaço de recursos, desenhando uma linha entre os exemplos no espaço de recursos e desenhando uma nova amostra em um ponto ao longo dessa linha. Especificamente, um exemplo aleatório da classe minoritária é escolhido primeiro. Então *k* dos vizinhos mais próximos para esse exemplo são encontrados (tipicamente *k=5*). Um vizinho selecionado aleatoriamente é escolhido e um exemplo sintético é criado em um ponto selecionado aleatoriamente entre os dois exemplos no espaço de características. Na figura 5 podemos encontrar uma representação visual desse processo.

Figura 5 - SMOTE



Fonte: Analytics Vidhya Website, Autor: Zaki Jefferson. Disponivel em: https://medium.com/analytics-vidhya/bank-data-smote-b5cb01a5e0a2

Em outras palavras, para resolver o problema com a classificação desequilibrada de que há muito poucos exemplos da classe minoritária para que um modelo aprenda efetivamente o limite de decisão, o SMOTE então, utiliza uma técnica chamada em oversampling, que consiste em superamostrar os exemplos na classe minoritária. Isso pode ser alcançado simplesmente duplicando exemplos da classe minoritária no conjunto de dados de treinamento antes de ajustar um modelo equilibrando a distribuição de classes, mas sem fornecer informação adicional alguma ao modelo.

2.4 RECURRENT PLOT

No ramo das redes neurais, é comum usarmos representações gráficas para o melhor aprendizado do algoritmo e baseado em (WANG, Z.; 2015), codificamos os nossos dados unidimensionais em imagens bidimensionais para alimentarmos nossa rede neural convolucional. E uma das técnicas utilizadas para realizar essa conversão foi o Recurrent plot ou, traduzindo, gráfico de recorrência.

Um gráfico de recorrência (RP) é uma técnica avançada de análise de dados não lineares. É uma visualização (ou um gráfico) de uma matriz quadrada, na qual os elementos da matriz correspondem àqueles tempos em que um estado de um sistema dinâmico se repete (colunas e linhas correspondem então a um certo par de tempos). Tecnicamente, o RP revela todos os momentos em que a trajetória do espaço de fase do sistema dinâmico visita aproximadamente à mesma área no espaço de fase.

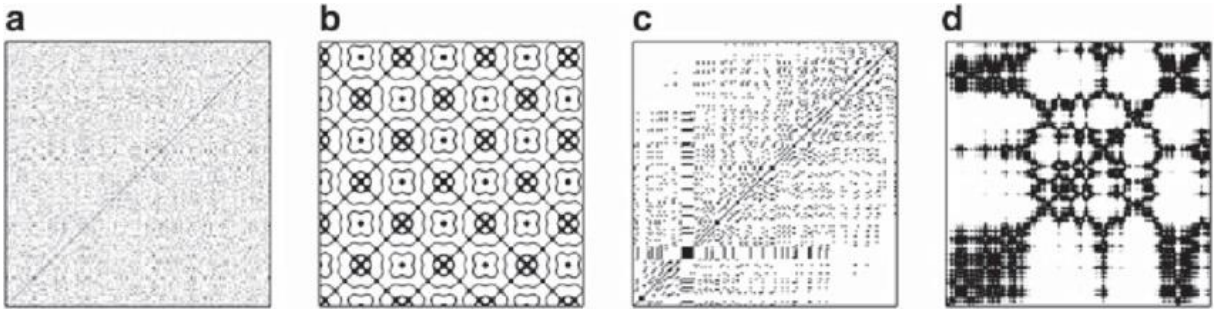
Apresentado em (ECKMANN et ai.1987) o gráfico de recorrência pode visualizar a recorrência de estados ($\vec{x}(i)$) em um espaço de fase. Normalmente, um espaço de fase não possui uma dimensão que permita sua representação. Espaços de fase de dimensão superior só podem ser visualizados por projeção nos subespaços bi ou tridimensionais. No entanto, a ferramenta de Eckmann nos permite investigar a m trajetória do espaço de fase tridimensional através de uma representação bidimensional de suas recorrências. Tal recorrência de um estado no tempo e em um momento diferente j é marcado dentro de uma matriz quadrada bidimensional com pontos de uns e zeros (pontos pretos e brancos no gráfico), onde ambos os eixos são eixos de tempo. Isso pode ser expresso matematicamente como:

$$R(i,j) = \Theta\left(\varepsilon - \left|\left|\vec{x}(i) - \vec{x}(j)\right|\right|\right) \qquad i,j = 1,\dots,N, \tag{5}$$

Onde ε é uma distância limite, || . || é uma norma, Θ a função de Heaviside e N o número de estados considerados.

Após esse cálculo, o RP nos trará um padrão característico dos dados que se aplicado a pequenos sinais é chamado de textura e se aplicado a grandes sinais é chamado de tipologia. A tipologia oferece uma impressão global do sinal que pode ser caracterizada como homogênea, periódica, à deriva é desordenada/interrompida.

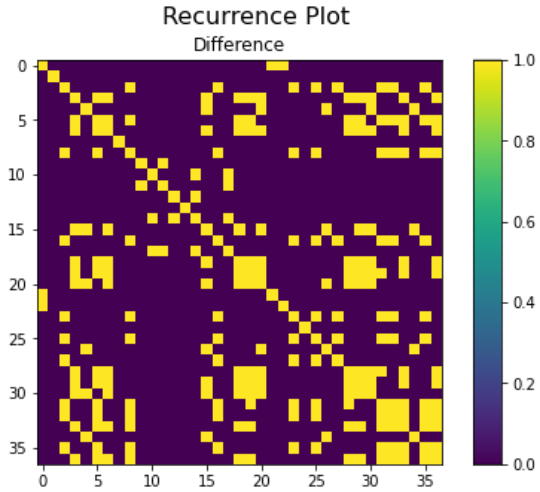
Figura 6 - Tipologia característica do gráfico de ocorrência: (A) homogêneo (ruído uniformemente distribuído), (B) periódico (oscilações harmônicas superposicionadas), (C) deriva (mapa logístico corrompido com um termo linear crescente) e (D) interrompido.



Fonte: Research Gate, Autor: Charles L. Webber, Disponível: https://www.researchgate.net/figure/Characteristic-typology-of-Recurrent-plots-a-homogeneous-uniformly-distributed_fig1_265003618, Acessado em: 04/06/2022 .

No nosso estudo, essa técnica foi aplicada aos sinais do banco de dados do domínio da frequência com a inclusão de alguns dados estatísticos e a imagem gerada pela equação pode ser observada na figura 7 abaixo:

Figura 7 - Imagem gerada pelo Recurrent Plot



Fonte: Autor.

Como pode ser observado, os valores estão ordenados da esquerda para a direita e de cima para baixo, ou seja, o primeiro valor está localizado no canto superior esquerdo e o

último valor está no canto inferior direito. Para cada um desses valores é estipulado um tom de acordo com uma paleta de cores que varia os tons de acordo com valores de 0 até 1. Na figura 7 exemplar acima, vemos que com a interferência dos dados estatísticos aos conjunto de dados, a imagem criada pelo Recurrent Plot segue o padrão de derivação da figura 6.

2.5 MARKOV TRANSITION FIELDS

Markov Transition Fields (MTF) é outra técnica de visualização para destacar o comportamento de séries temporais, como o Recurrent Plot. Começamos colocando cada valor na série temporal em quantis, nesses quantis cada valor é equivalente a um bin. Por exemplo, se usarmos quartis (4 bins), os menores 25% dos valores definiriam os limites do primeiro quartil, os segundos menores 25% dos valores definiriam os limites do segundo quartil, etc. Assim, podemos pensar em cada bin como um 'estado'.

Para o funcionamento do modelo de Markov, precisamos criar uma matriz de transição de estados e esses valores podem ser obtidos através da seguinte equação:

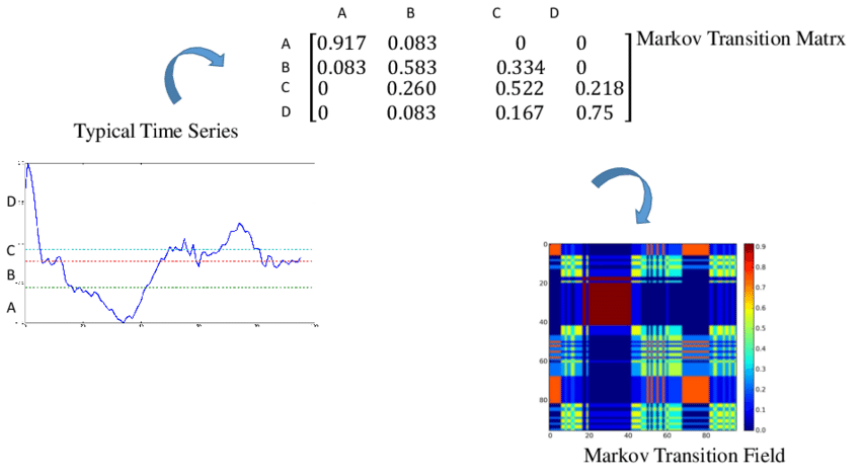
$$A_{ij} = P(s_t = j \mid s_{t-1} = i) \qquad (6)$$

Onde Aij é a probabilidade de transição do estado i para o estado j. Logo a matriz MTF definida ficaria como:

$$M_{kl} = A_{qkql} \qquad (7)$$

M_{kl} é a probabilidade de uma transição de um passo para o bin de x_k, para o bin de x_l, onde x_k e x_l, são dois pontos na série temporal no time step de k e l, respectivamente. Logo, q_k e q_l, correspondem aos quantis. Com a matriz de probabilidade de transição definida podemos gerar uma imagem através dessa matriz, como mostra o exemplo abaixo na figura 8:

Figura 8- Ilustração de um mapa de codificação de Markov Transition Field



Fonte: (WANG, Zhiguang. Spatially Encoding Temporal Correlations to ClassifyTemporal Data Using Convolutional Neural Networks)

2.6 MATRIZ DE CONFUSÃO

A pontuação F1 é uma métrica de aprendizado de máquina que pode ser usada em modelos de classificação, é muito útil quando se está lidando com problemas de classes desequilibradas como é o nosso caso. Essa métrica de aprendizado é uma melhoria proposta de duas métricas de desempenho mais simples. Portanto, antes de entrar nos detalhes da pontuação F1, vamos voltar atrás e fazer uma visão geral dessas métricas subjacentes à pontuação da F1.

Primeiramente, vamos aos conceitos de positivo e negativo, esses dois conceitos apresentados são duas classes opostas uma à outra, por exemplo, sobreviveu/não sobreviveu, câncer/não câncer, spam/não spam, etc. também pode ser dividido em gato/cachorro, macho/fêmea. Então, uma classe nós consideramos como positiva e outra como negativa. É arbitrário ou depende do objetivo do estudo para que você tome um como positivo e outro como negativo. Não há nenhum aspecto bom (positivo) ou ruim (negativo) nele.

Segundamente, vamos aos conceitos de verdadeiro e falso, quando há dados amostrais de alguma população e utilizamos modelagem através da qual podemos prever sua classe/rótulos. O “verdadeiro” representa os registros que o modelo conseguiu identificar sua classe, enquanto “falso” representa os registros que o modelo não conseguiu identificar.

Aplicando esses conceitos juntos ao aprendizado da máquina temos a seguinte divisão:

Verdadeiro positivo (TP): a previsão está correta e o valor real é positivo (ou seja, esse valor é um dos valores que o modelo estava tentando identificar. No caso de um modelo que busca identificar clientes que provavelmente comprarão o produto, esses dados apontam representa um potencial comprador).

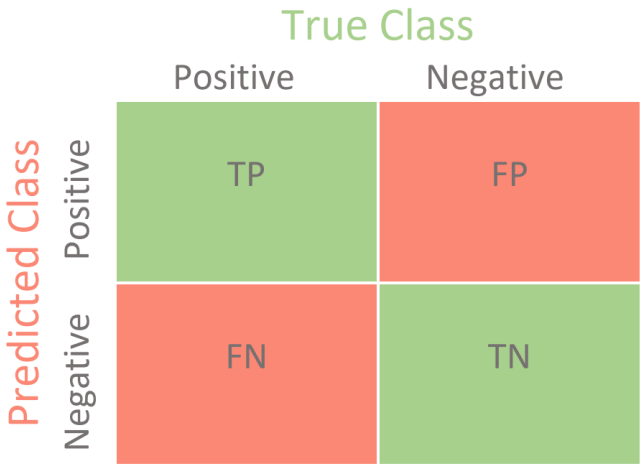
Falso positivo (FP): a previsão está errada e o valor real é positivo.

Verdadeiro negativo (TN): a previsão está correta e o valor real é negativo (ou seja, esse valor não é um dos valores que o modelo estava tentando identificar. No caso de um modelo que busca identificar clientes que provavelmente comprarão o produto, esses dados representam um cliente que não está interessado em comprar).

Falso negativo (FN): a previsão está errada e o valor real é negativo.

Uma matriz de confusão é uma tabela que é frequentemente usada para descrever o desempenho de um modelo de classificação em um conjunto de dados de teste para os quais os valores verdadeiros são conhecidos, essa matriz correlaciona em si os conceitos de TP, FP, FN e TN em uma demonstração gráfica quantitativa desses valores. Na figura 9 abaixo, podemos ver a estrutura geral de uma matriz de confusão:

Figura 9 – Modelo de matriz de confusão



Fonte: Precision and Recall, disponível em: <https://en.wikipedia.org/wiki/Precision_and_recall>

A partir da matriz de confusão, também podemos calcular a precisão, a acurácia, o recall e a pontuação F1 do modelo. Essas medidas ajudam a entender o desempenho do modelo. A partir das fórmulas, podemos calcular essas medidas e para o cálculo podemos utilizar a matriz de confusão.

Acurácia: é a razão dos registros que o modelo classificou corretamente sobre o número total de registros.

$$\frac{TP+TN}{TP+TN+FP+FN} = \frac{Nº\ de\ predições\ corretas}{Nº\ de\ todas\ predições\ feitas} \qquad (8)$$

Precisão: é a proporção dos positivos que são identificados corretamente pelo modelo sobre o total de registros positivos.

$$\frac{TP}{TP+FP} \qquad (9)$$

Recall: é uma medida de quantos casos positivos o classificador previu corretamente, sobre todos os casos positivos nos dados.

$$\frac{TP}{TP+FN} \qquad (10)$$

Pontuação F1: é uma medida que combina precisão e recall. É geralmente descrito como a média harmônica dos dois.

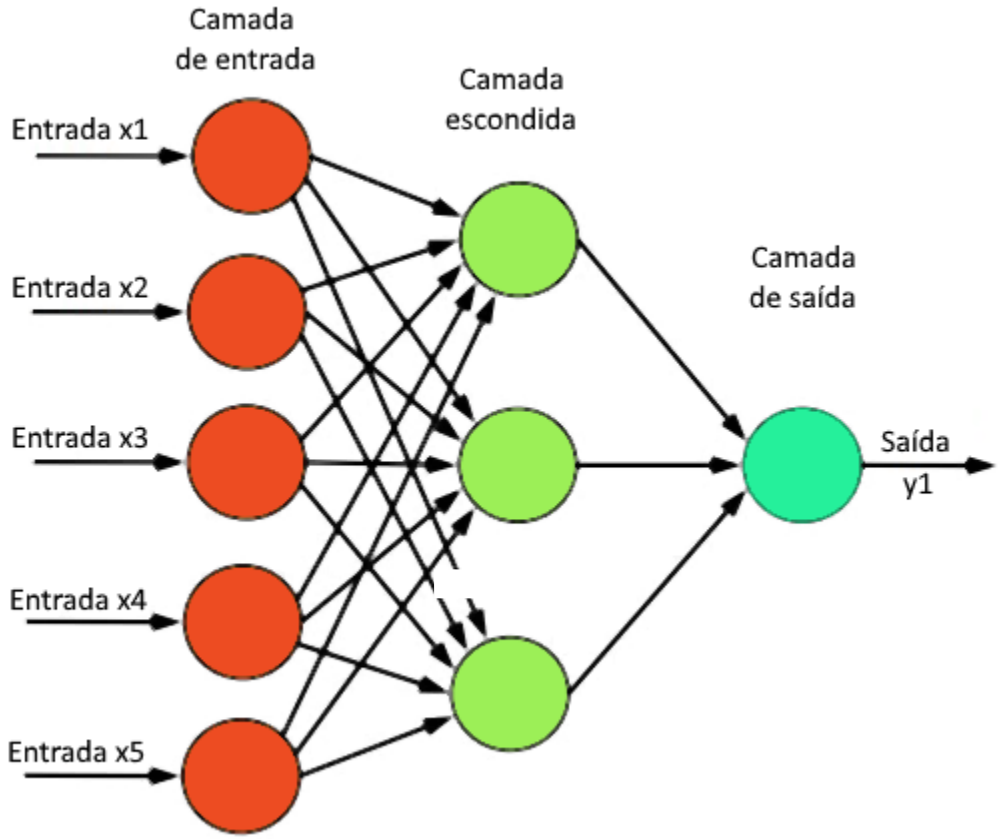
$$\frac{2\ .\ Precisão\ .Recall}{Precisão+Recall} \qquad (11)$$

2.7 REDES NEURAIS ARTIFICIAIS

As redes neurais artificiais (RNAs) são um subconjunto do aprendizado de máquina e estão no centro dos algoritmos de deep learning, são compostas por camadas de nós, contendo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada nó, ou neurônio artificial, conecta-se a outro e tem um peso e um limiar associados. Se a saída de qualquer nó individual estiver acima do valor limite especificado, esse nó é ativado, enviando dados para a próxima camada da rede. Caso contrário, nenhum dado é passado para a próxima camada da rede.

Pela complexidade do problema proposto a ser resolvido por este estudo, foi utilizada a aplicação de um mecanismo apropriado para sua resolução, como o Multilayer Percepcron, que consiste em uma classe totalmente conectada de rede neural artificial em que as conexões entre os nós não formam um ciclo. Ela consiste em três tipos de camadas - a camada de entrada, a camada de saída e a camada oculta, como podem ser observadas na figura 10 abaixo:

Figura 10 – Estrutura de uma Multilayer Percepcron



Fonte: Conflito Permanente, Disponível em: https://conflitopermanente.wordpress.com/2017/07/05/redes-neuronais-artificiais-rascunho/, Acessado em: 06/06/2022.

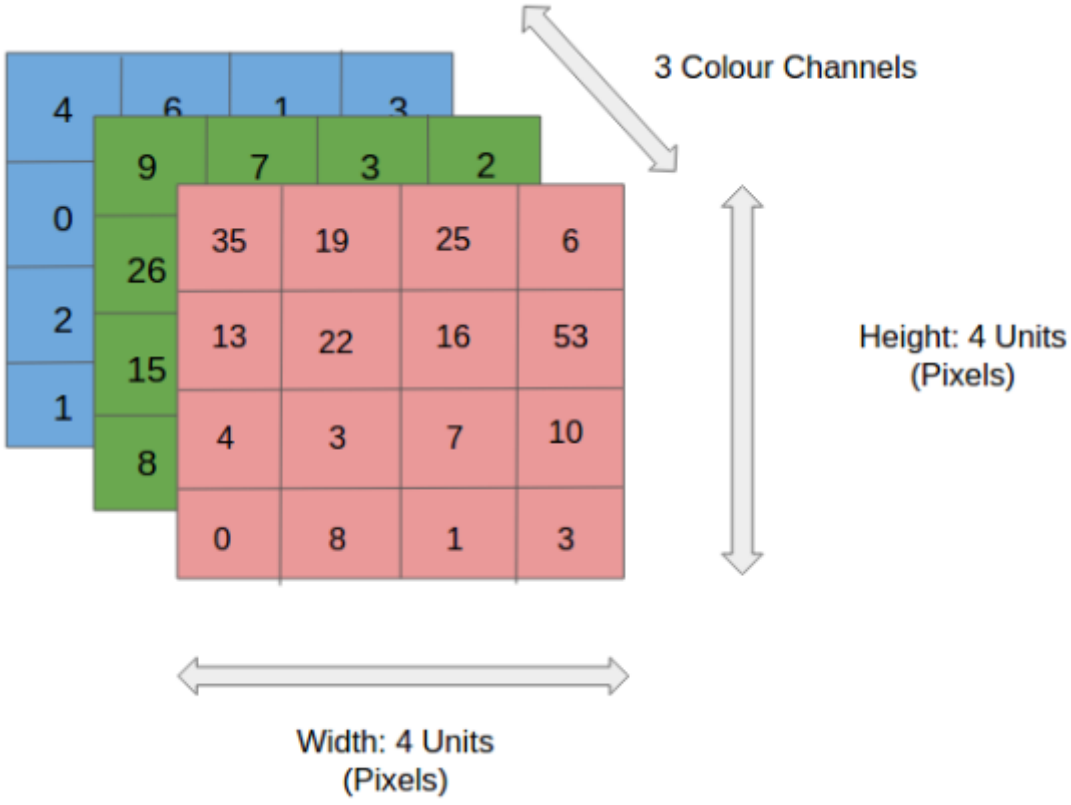
A camada de entrada recebe o sinal de entrada a ser processado. A tarefa necessária, como previsão e classificação, é realizada pela camada de saída. Um número arbitrário de camadas ocultas que são colocadas entre as camadas de entrada e saída são o verdadeiro mecanismo computacional do MLP. Semelhante a uma rede feed forward em um MLP, os dados fluem na direção direta da camada de entrada para a camada de saída. Os neurônios no MLP são treinados com o algoritmo de aprendizado de retropropagação. As MLPs são projetadas para aproximar qualquer função contínua e podem resolver problemas que não são linearmente separáveis. Os principais casos de uso do MLP são classificação de padrões, reconhecimento, previsão e aproximação.

2.8 Redes Neurais Convolucionais

Uma Rede Neural Convolucional (ConvNet/CNN) é um algoritmo de Deep Learning que pode receber uma imagem de entrada, atribuir importância (pesos e vieses aprendíveis) a

vários aspectos/objetos na imagem e ser capaz de diferenciar um do outro. A CNN é especializada no processamento de dados que possuem uma topologia semelhante a uma grade, como uma imagem. Uma imagem digital é uma representação binária de dados visuais. Ele contém uma série de pixels organizados em forma de grade que contém valores de pixel para denotar quão brilhante e qual cor cada pixel deve ser, uma representação desse processo se encontra na figura 11 abaixo, utilizando uma figura no modelo RGB (modelo de adição de cores no qual o vermelho , o verde e o azul são somados de várias maneiras para reproduzir uma ampla gama de cores) :

Figura 11 – Exemplo de uma imagem RGB (separada por 3 planos de cores)



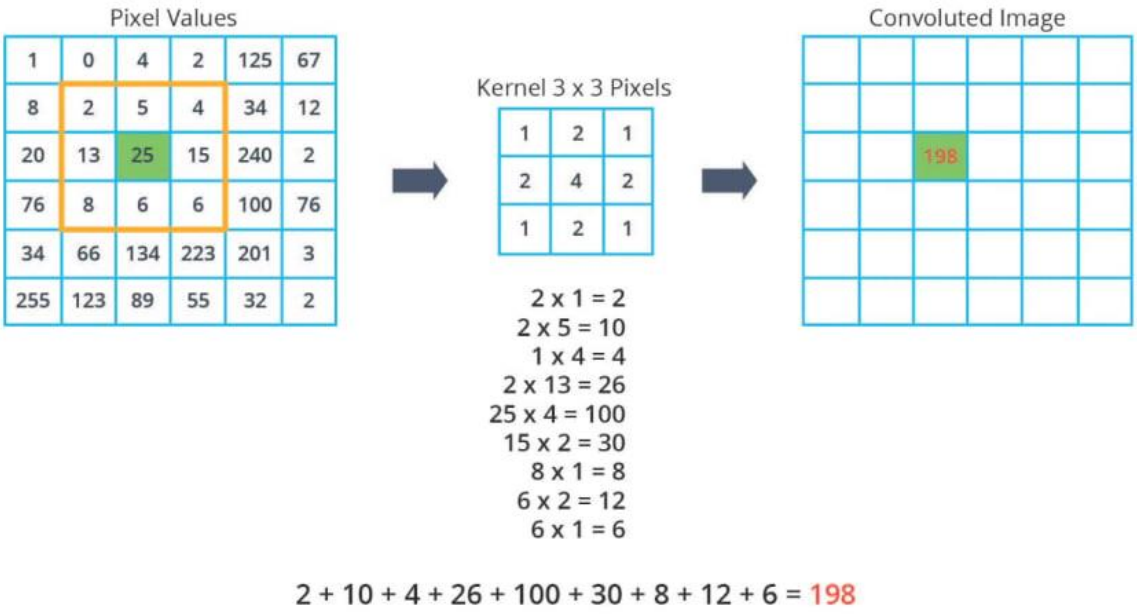
Fonte: MNIST–Handritten–Digit–Recognition–using-CNN, Disponível em: https://github.com/NehaCkumari/MNIST-Handwritten-Digit-Recognition-using-CNN, Acessado em: 06/06/2022.

Enquanto nos métodos tradicionais os filtros são projetados à mão, com treinamento suficiente, as CNNs têm a capacidade de aprender esses filtros/características. Em outras palavras, uma rede neural convolucional é apenas uma rede neural que usa convolução.

Convolução é uma operação matemática que permite a fusão de dois conjuntos de informações. No caso da CNN, a convolução é aplicada aos dados de entrada para filtrar as informações e produzir um mapa de características. Esse filtro também é chamado de kernel,

ou detector de recursos, e suas dimensões podem ser, por exemplo, 3x3. Para realizar a convolução, o kernel percorre a imagem de entrada, fazendo a multiplicação da matriz elemento após elemento. O resultado para cada campo receptivo (a área onde ocorre a convolução) é anotado no mapa de características, como pode ser visto na figura 12 abaixo:

Figura 12 - Demonstração da atuação do Kernel / Filtro.



Fonte: Machine Learning – Convolution with color images, Disponível em: https://dev.to/sandeepbalachandran/machine-learning-convolution-with-color-images-2p41/, Acessado em: 06/06/2022.

O objetivo da CNN é reduzir as imagens para que sejam mais fáceis de processar sem perder recursos valiosos para uma previsão precisa. A arquitetura ConvNet tem três tipos de camadas: camada convolucional, camada de pool e camada totalmente conectada.

A camada convolucional é o bloco de construção central de uma CNN que faz a maior parte do trabalho pesado computacional. Os parâmetros da camada convolucional consistem em um conjunto de filtros/kernels. Cada kernel é pequeno espacialmente (ao longo da largura e da altura), mas se estende por toda a profundidade do volume de entrada. Por exemplo, um filtro típico em uma primeira camada de um ConvNet pode ter tamanho 5x5x3 (ou seja, 5 pixels de largura e altura e 3 porque as imagens têm profundidade 3, os canais de cor). Durante a passagem para frente, deslizamos (mais precisamente, convolvemos) cada filtro pela largura e altura do volume de entrada e calculamos os produtos escalares entre as entradas do filtro e a entrada em qualquer posição. À medida que deslizamos o kernel sobre a

largura e a altura do volume de entrada, produziremos um mapa de ativação bidimensional que fornece as respostas desse filtro em cada posição espacial. Intuitivamente, a rede aprenderá filtros que são ativados quando eles veem algum tipo de recurso visual, como uma borda de alguma orientação ou uma mancha de alguma cor na primeira camada, ou eventualmente padrões inteiros em forma de favo de mel ou roda em camadas superiores da rede. Agora, teremos um conjunto inteiro de filtros em cada camada convolucional (por exemplo, 12 filtros), e cada um deles produzirá um mapa de ativação bidimensional separado, funcionamento analogamente ao processo da figura 11 acima.

A camada de pooling é responsável por reduzir progressivamente o tamanho espacial da representação para reduzir a quantidade de parâmetros e computação na rede e, portanto, também controlar o overfitting e baratear o custo operacional. A camada de pooling opera independentemente em cada fatia de profundidade da entrada e a redimensiona espacialmente. Uma das formas mais comum de pooling é Max Pooling, que consiste em uma camada de pooling com filtros de tamanho 2x2 aplicados com um passo de 2 diminui cada fatia de profundidade na entrada por 2 ao longo da largura e da altura, descartando 75% das ativações. Cada operação Max, neste caso, levaria no máximo 4 números (pequena região 2x2 em alguma fatia de profundidade). o elemento mais significativo é obtido como entrada do mapa de recursos e é executado para fornecer a imagem de saída com dimensões bastante reduzidas, mantendo as informações essenciais.

A camada totalmente conectada é a última camada que determina a saída. A saída da camada de pooling é achatada em um vetor unidimensional e, em seguida, fornecida como entrada para a camada totalmente conectada. A camada de saída possui o mesmo número de neurônios que o número de categorias que tivemos em nosso problema de classificação, associando assim características a um rótulo específico. Após este processo ser conhecido como propagação de encaminhamento, a saída assim gerada é comparada com a produção real para geração de erros. O erro é então retropropagado para atualizar os filtros (pesos) e valores de polarização. Assim, um treinamento é concluído após esse ciclo de propagação de encaminhamento e retrocesso.

2.9 REDES DE MEMÓRIA DE CURTO PRAZO LONGO

Uma rede neural recorrente (RNN) é um tipo de rede neural artificial que usa dados sequenciais ou dados de séries temporais. Esses algoritmos de aprendizado profundo são comumente usados para problemas ordinais ou temporais, como tradução de idiomas,

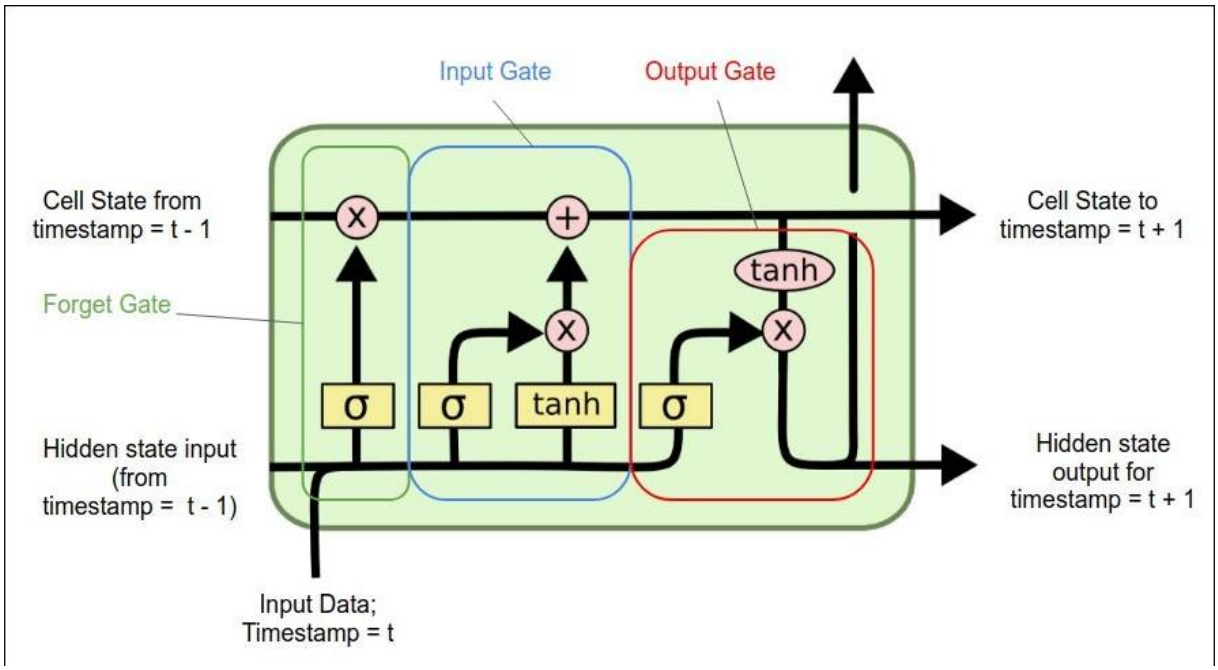
processamento de linguagem natural, reconhecimento de fala e legendagem de imagens. Assim como as redes neurais de feedforward e convolucionais (CNNs), as redes neurais recorrentes utilizam dados de treinamento para aprender. Eles se distinguem por sua "memória", pois obtêm informações de entradas anteriores para influenciar a entrada e a saída atuais. Enquanto as redes neurais de deep learning tradicionais assumem que as entradas e saídas são independentes umas das outras, a saída das redes neurais recorrentes depende dos elementos anteriores dentro da sequência.

Uma das RNNs mais famosas é a Rede de Memória de Curto Prazo Longo, ou como é comumente conhecida, LSTM. As redes LSTM foram projetadas especificamente para superar o problema de dependência de longo prazo enfrentado por redes neurais recorrentes RNNs (devido ao problema do gradiente de fuga). Os LSTMs têm conexões *de* feedback que os tornam diferentes das redes neurais de feed forward mais tradicionais. Essa propriedade permite que os LSTMs processem sequências inteiras de dados (por exemplo, séries temporais) sem tratar cada ponto na sequência de forma independente, mas retendo informações úteis sobre dados anteriores na sequência para ajudar no processamento de novos pontos de dados. Como resultado, os LSTMs são particularmente bons no processamento de sequências de dados, como texto, fala e séries temporais gerais.

A saída de um LSTM em um determinado momento depende de três coisas: A memória atual de longo prazo da rede - conhecida como estado da célula, a saída no momento anterior - conhecido como o estado oculto e os dados de entrada no passo de tempo atual.

Os LSTMs usam uma série de 'gates' que controlam como as informações em uma sequência de dados entram, são armazenadas e saem da rede. Existem três portas em um LSTM típico: forget gate, input gate e output gate. Essas portas podem ser consideradas filtros e são cada uma sua própria rede neural, na figura 13 abaixo há uma representação diagramática de um LSTM.

Figura 13 – Estrutura de LSTM.

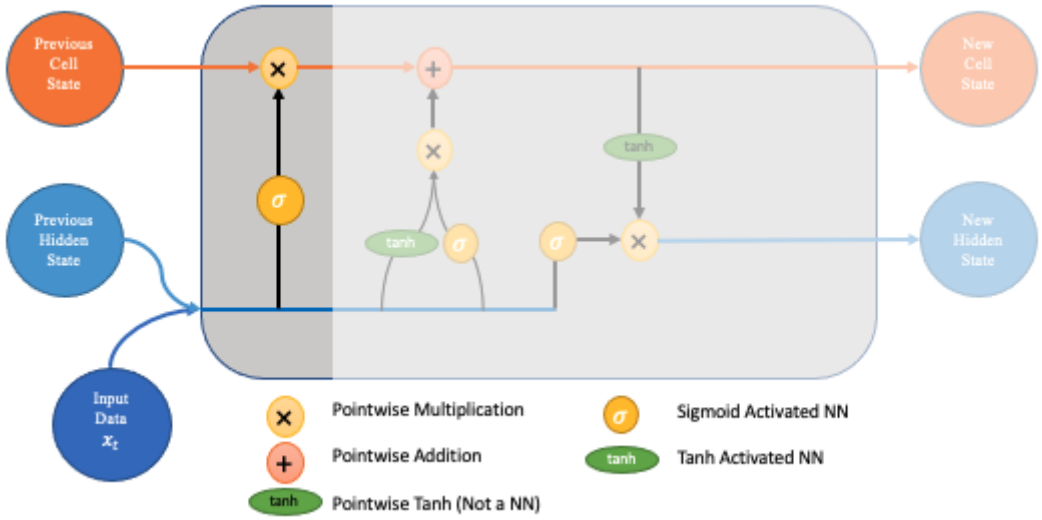


Fonte: Comprehensive Guide to LSTM & RNN, Disponível em: https://www.turing.com/kb/comprehensive-guide-to-lstm-rnn/, Acessado em: 08/06/2022.

Gates contém ativações sigmoides (σ). Uma ativação sigmóide é semelhante à ativação tanh. Em vez de comprimir valores entre -1 e 1, ele comprime valores entre 0 e 1. Isso é útil para atualizar ou esquecer dados porque qualquer número multiplicado por 0 é 0, fazendo com que os valores desapareçam ou sejam “esquecidos”. Qualquer número multiplicado por 1 é o mesmo valor, portanto, esse valor permanece o mesmo ou é “mantido”. A rede pode aprender quais dados não são importantes, portanto, podem ser esquecidos ou quais dados são importantes para manter.

Primeiro, temos forget gate. Este gate decide quais informações devem ser descartadas ou mantidas. As informações do estado oculto anterior e as informações da entrada atual são passadas pela função sigmóide. Os valores saem entre 0 e 1. Quanto mais próximo de 0 significa esquecer, e quanto mais próximo de 1 significa manter. A figura 14 traz uma ilustração do forget gate:

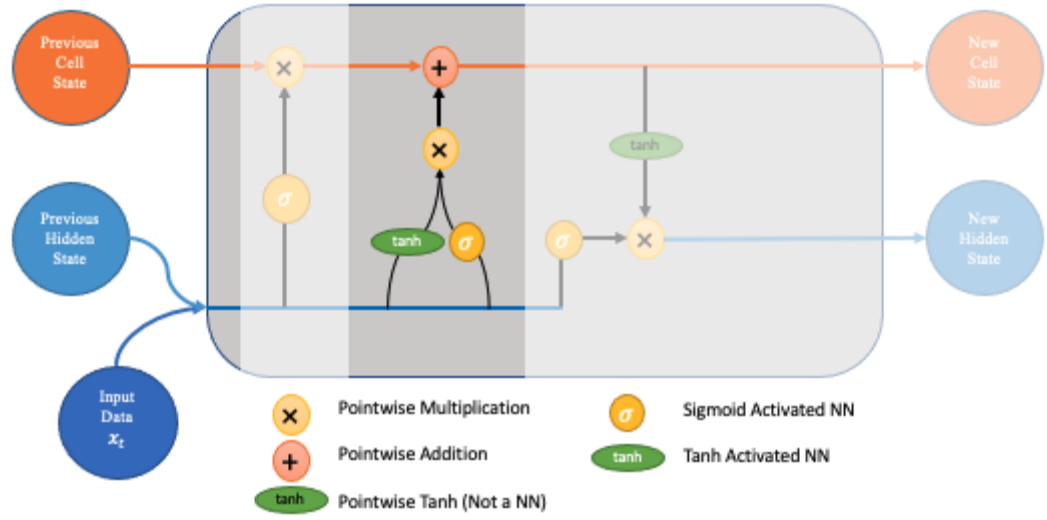
Figura 14 – Forget Gate



Fonte: LSTM Networks – A Detailed Explaination, Disponível em: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9/, Acessado em: 08/06/2022.

Para atualizar o estado da célula, temos o input gate. Primeiro, passamos o estado oculto anterior e a entrada atual para uma função sigmóide. Isso decide quais valores serão atualizados transformando os valores entre 0 e 1. 0 significa não importante e 1 significa importante. Assim, passa o estado oculto e a entrada atual para a função tanh para compactar valores entre -1 e 1 para ajudar a regular a rede. E por fim, multiplica a saída tanh pela saída sigmóide. A saída sigmóide decidirá qual informação é importante manter da saída tanh.

Figura 15 – Input gate

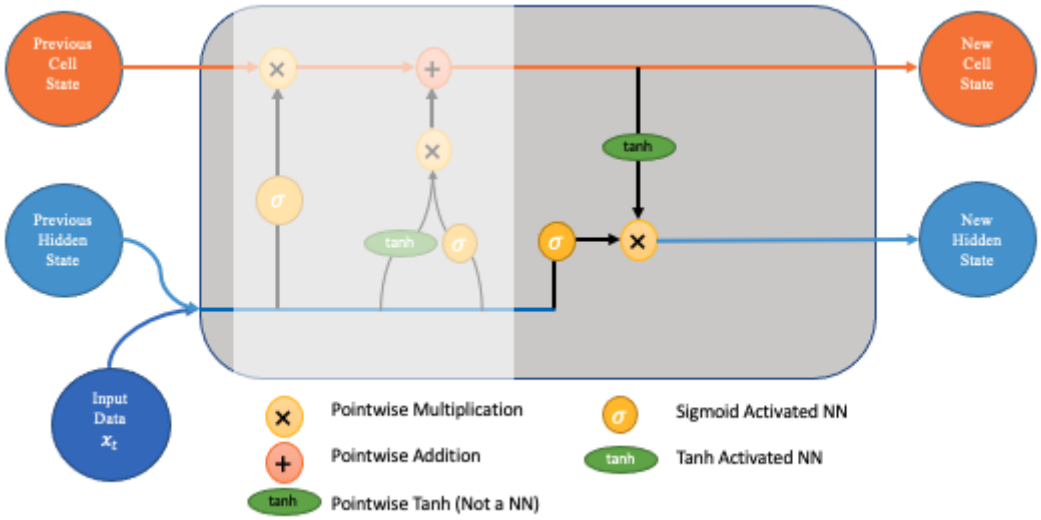


Fonte: LSTM Networks – A Detailed Explaination, Disponível em: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9/, Acessado em: 08/06/2022.

Após esse processo, teremos informações suficientes para calcular o estado da célula. Primeiro, o estado da célula é multiplicado pontualmente pelo vetor de esquecimento. Isso tem a possibilidade de descartar valores no estado da célula se for multiplicado por valores próximos de 0. Em seguida, pegamos a saída da porta de entrada e fazemos uma adição pontual que atualiza o estado da célula para novos valores que a rede neural considera relevantes. Isso nos dá nosso novo estado celular.

Por último a output gate decide qual deve ser o próximo estado oculto. O estado oculto contém informações sobre entradas anteriores e também é usado para previsões. Primeiro, passamos o estado oculto anterior e a entrada atual para uma função sigmóide. Em seguida, passamos o estado da célula recém-modificado para a função tanh. Multiplicamos a saída tanh pela saída sigmóide para decidir quais informações o estado oculto deve transportar. A saída é o estado oculto. O novo estado da célula e o novo oculto são então transportados para a próxima etapa de tempo.

Figura 16 – Output Gate



Fonte: LSTM Networks – A Detailed Explaination, Disponível em: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9/, Acessado em: 08/06/2022.

A ConvNet e LSTM, ambas as redes foram utilizadsa por esse estudo e foram construídas juntando todos esses layers apresentados, colocados em várias posições e com distintas contagens de layers de cada tipo. As redes estudadas são expostas mais detalhadamente no próximo capítulo, onde haverá uma concisa elucidação da metodologia e do enfoque do trabalho.

3. MATERIAIS E MÉTODOS

Esta seção busca apresentar os materiais empregados durante a pesquisa e os balanços computacionais do Power Spectral Density e das extrações de características aplicadas ao banco de dados. O capitulo está divido em 3 seções, são elas: Banco de Dados, Cálculo do Power Spectral Density aplicado ao sinal e Extração de Características.

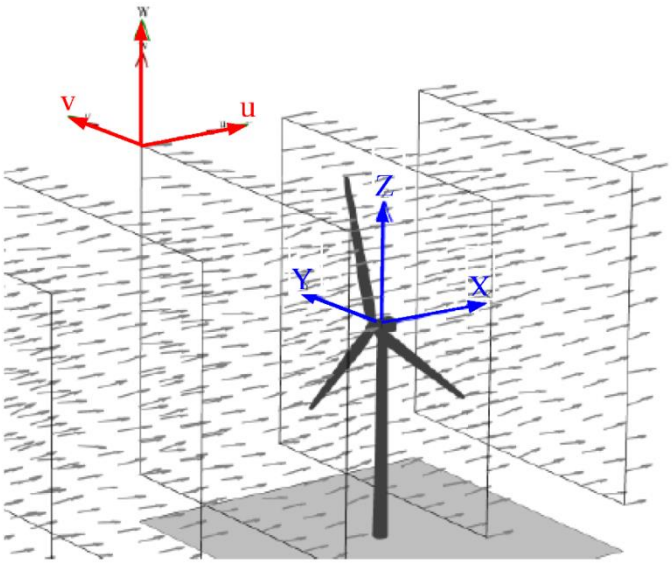
3.1 BANCO DE DADOS

Há duas formas de se produzir um banco de dados com velocidade do rotor de uma turbina eólica: através de um sensor em uma turbina ou através de simulações feitas por softwares. A alternativa inicial é extremamente mais cara de se conseguir. Além do custo do maquinário, o preço operacional ainda deve incluído nessa conta, que muito provavelmente, alcançará valores milionários. Porém, o uso de softwares para produção de dados se mostra uma maneira definitivamente mais barata para o desenvolvimento do banco de dados. Em vista disso, o banco de dados empregado neste estudo foi totalmente adquirido via simulações, utilizando principalmente dois programas: Turbsim e FastV8. O Banco de Dados foi elaborado em (ROSA, 2019) e em seguida será realizada uma concisa descrição do procedimento empregado.

3.1.1 Turbsim

O TurbSim é um simulador estocástico de ventos turbulentos, em outras palavras, à partir de um padrão aleatório definido pelo tempo, simula ventos que provavelmente seriam danosos a uma turbina eólica. Ele usa um modelo estatístico para simular numericamente séries temporais de vetores de velocidade do vento de três componentes em pontos em uma grade retangular vertical bidimensional que é fixa no espaço. A figura 17 exemplifica a série de ventos produzida.

Figura 17 – Representação Gráfica da Série temporal de vento do Turbsim



(conclusão)

Fonte: (JONKMAN, B.J; 2009)

3.1.2 Fast v8.

O Fast v8 é um programa que permite o usuário simular Turbinas Eólicas com diversas variações de parâmetros para a modelagem de eventos muito característicos e que se ajustem perfeitamente ao escopo escolhido pelo projetista. O software comporta aos pesquisadores gerar várias formas de modelos sendo eles: modelo das dinâmicas estruturais modelo aerodinâmico e modelo do sistema elétrico. Assim, para a obtenção do resultante final, todos esses modelos citados anteriormente são interligados. Na Tabela 1 estão os dados da Turbina simulada no Fastv8.

Tabela 1 – Parâmetros da Turbina Eólica Simulada no Fast v8

Parâmetro	Valor
Potência Nominal	1.5MW
Tipo de Maquina Elétrica	Gerador Síncrono de Îmas Permanentes
Altura do HUB	84m
Diametro do Rotor	70m
Orientação do Rotor	Upwind
Configuração do Rotor	3 pás
Velocidade Nominal	2.14 rad/s (20 RPM)
Torque nominal do rotor	736,79kNm

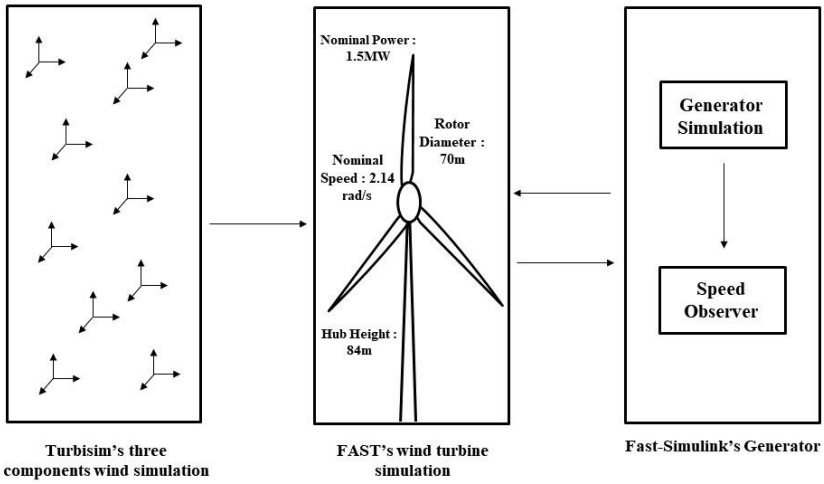
Fonte: (ROSA,2019)

Afora os programas Turbsim e Fast v8, o autor ainda emprega Matlab Simulink para concretizar a conexão do sistema e realizar a simulação do aerogerador de uma forma mais exata ao instituir uma interconexão com Fast-Simulink. Enfim, também são usadas algoritmos em Python para o desenvolvimento de uma interface gráfica para que quem estiver utilizando esse sistema possa alterar os parâmetros de modo mais intuitivo.

3.1.3 Geração do Banco de Dados

O banco de dados foi desenvolvido adotando o esquemático evidenciado na figura 18. Com a finalidade de produzir um banco de dados com modelos de pás de massa desbalanceada, foi feita uma alteração da densidade das pás a uma percentagem de 1%, ou seja, houve materiais com densidade de 95% até 105% da densidade original do material na simulação.

Figura 18 – Fluxograma de dados.



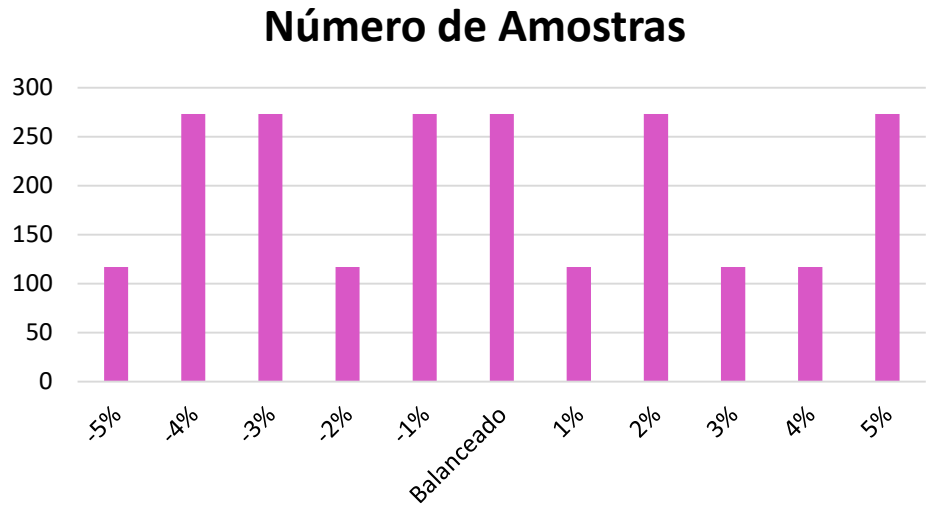
Fonte: (LUCAS, 2021).

Houve a efetivação de 390 simulações para cada classe de desbalanço, ou seja, cada comparação foi feita utilizando uma classe completamente balanceada, enquanto as outras mantinham seu desbalanceamento proposto. Por exemplo, a classe normal (balanceada) teve 273 simulações feitas, porém, a classe de 1% de desbalanço foi formada por 273 simulações de -1% de desbalanço e 117 simulações com +1% de desbalanço, assim, obtendo no total 390

amostras para a classe 1% geral. A união dos desbalanços de 1% em uma classe só, deve-se ao fato de que como o algoritmo classifica preferencialmente o desbalanço absoluto, a classificação de -1% e +1% podem ser relacionadas unitariamente.

O programa armazenou as estimações em uma frequência de 2kHz em um intervalo de tempo de 180 segundos por simulação, 60 segundos iniciais em regime transiente e os 120 segundos finais em regime permanente. O número total de simulações por cada desbalanço simulado se encontra na figura 19 abaixo, onde os valores armazenados se encontram somente no regime permanente.

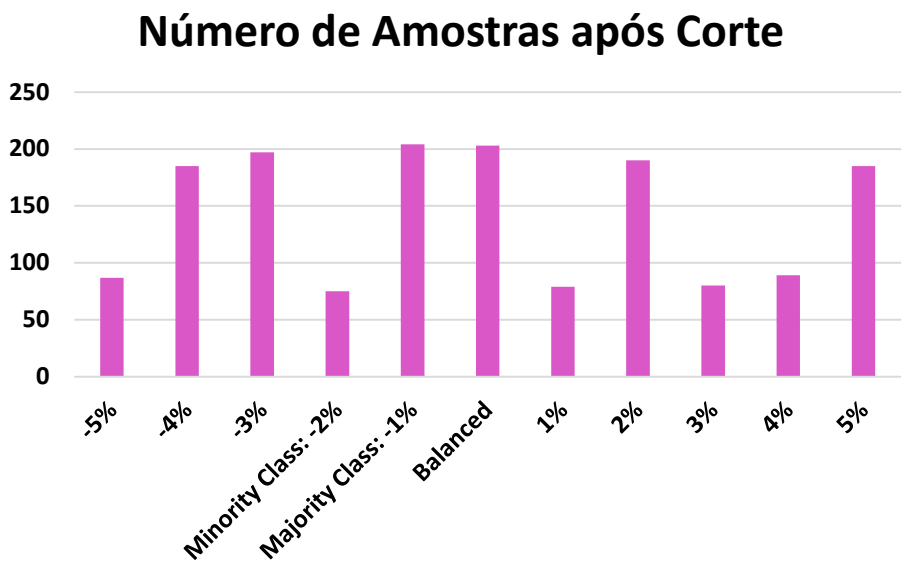
Figura 19 – Número de amostras por classe.



Fonte: Autor.

Depois da conclusão das simulações, foi estabelecido um limite de corte entre 390 rad/s e 330 rad/s. Este limite serve para remover amostras muito excessivas, como por exemplo, uma intensa ventania ou uma ampla inércia no vento. Assim, removendo esses dados, prevenimos que o algoritmo classificatório não coloque margem para casos tão extraordinários, o que poderia ocasionar erros nos casos mais realistas. Na figura 20 temos o número de amostras por classe após a aplicação desse limite de corte.

Figura 20 – Número de amostras por classe após o corte.



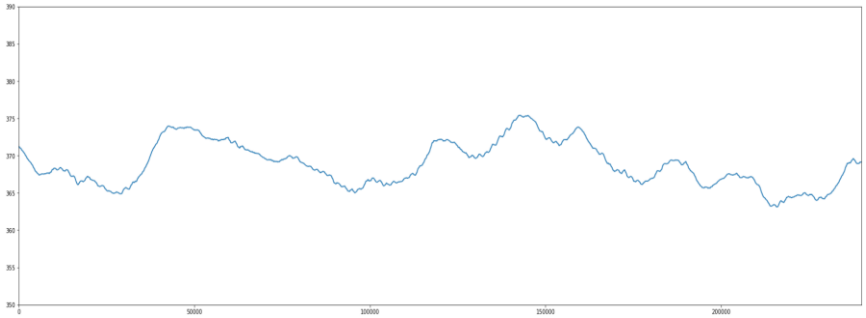
Fonte: Autor.

Por fim, o banco de dados temporal ficou constituído por 1574 amostras decompostas segundo a figura exibida acima.

3.2 CÁLCULO DO POWER SPECTRAL DENSITY APLICADO AO BANCO DE DADOS

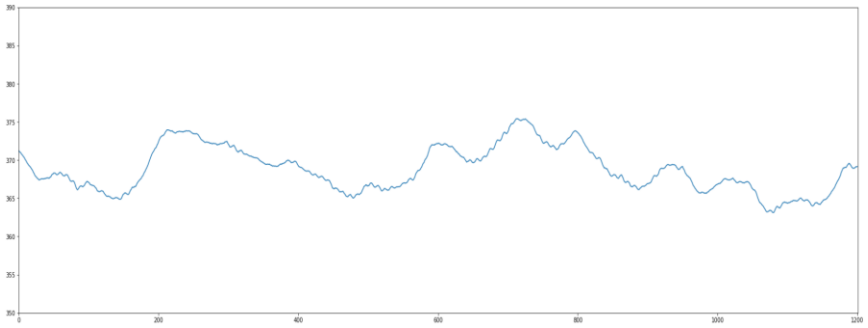
A equação do PSD apresentada na seção 2 foi posta em prática em todas as amostras originadas pelo software no processo anterior, empregando a metodologia de Welch (WELCH, 1967). Contudo, um downsampling foi aplicada nos dados do banco de dados, antes do processo Welch ser utilizado. Assim, a frequência do sinal temporal foi de 2kHz para 10Hz, e por consequência, a quantidade de amostragens foi reduzida de 240000 para 1200. Nas imagens 21 e 22, temos uma visualização da forma de onda do sinal original e da forma de onda do sinal depois do downsampling ser aplicado, respectivamente:

Figura 21 – Sinal Original no Tempo



Fonte: Autor.

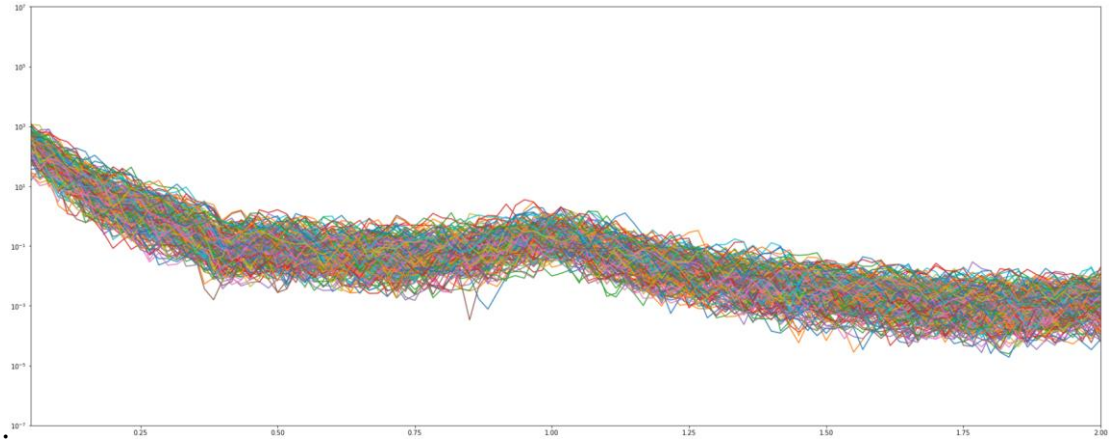
Figura 22 – Sinal no tempo após Downsampling



Fonte: Autor.

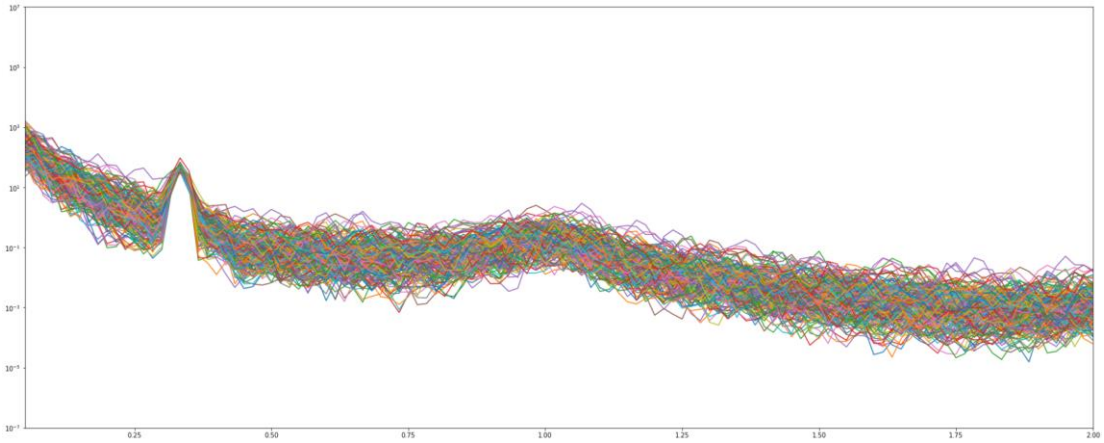
Posteriormente ao downsampling, foi aplicado o PSD no sinal. Logo podemos notar uma boa alteração em cada classe de sinal analisada: na frequência próxima a 0.33Hz ocorrem mudanças na densidade de potência alterando com o desbalanço de massa das pás. Nas figuras 23 e 24 podemos notar essas mudanças claramente ao nos atentarmos às frequências próximas da frequência mencionada:

Figura 23 – PSD de uma amostra de Turbina sem desbalanceamento



Fonte: Autor.

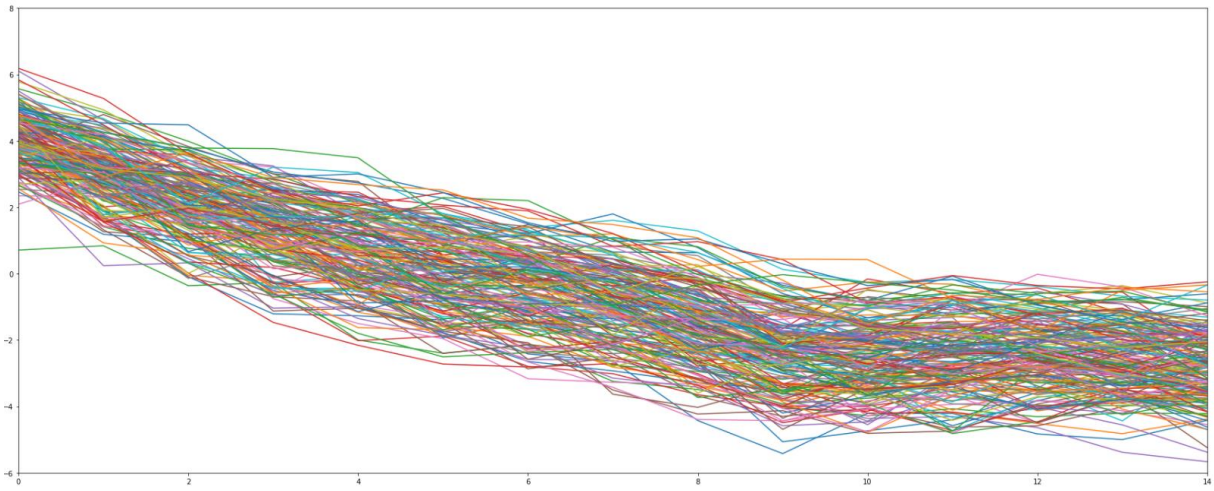
Figura 24 – PSD de uma amostra de Turbina com 5% de desbalanceamento



Fonte: Autor.

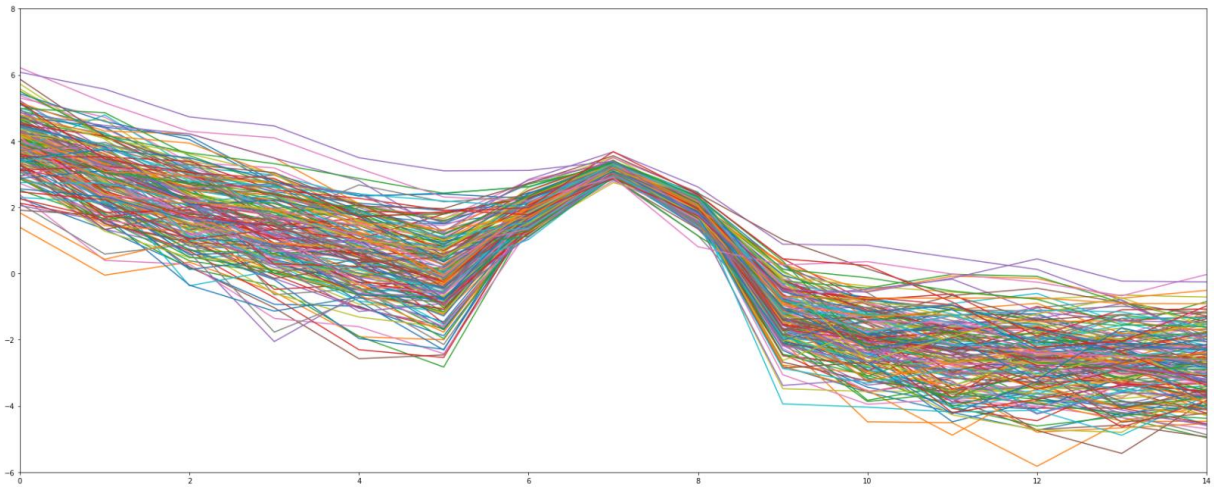
Essa deformação no espectro deixa o trabalho de classificar as amostragens com desbalanços se torne mais simples. Pelo fato da delimitação dessa deformação ser bem acentuada, não é preciso utilizar todas as estimações de frequência para o treinamento do nosso código, assim, os valores empregados consistiram em um fragmento de 15 dados de frequência que estão próximos à deformação, obtendo, portanto uma eficácia mais alta. Nas imagens 25 e 26 há uma representação do fragmento da classe sem desbalanceamento e da classe com 5% de desbalanceamento, respectivamente.

Figura 25 – Recorte normalizado ao redor da frequência de 0.33Hz do PSD da classe sem desbalanceamento



Fonte: Autor.

Figura 26 – Recorte normalizado ao redor da frequência de 0.33Hz do PSD da classe com 5% desbalanceamento



Fonte: Autor.

No espaço de frequência utilizado para representação acima, os valores se encontram normalizados, pois desta forma, prevenimos a intromissão de uma alta amplitude de potência em um ponto particular da frequência.

3.3 EXTRAÇÃO DE CARACTERISTICAS

Está gradativamente mais corriqueiro que um banco de dados fique demasiadamente volumoso demais e, portanto, mais complexo de ser avaliado pelo algoritmo. Isso se energiza neste estudo, porque o método utilizado, no qual adentraremos mais a fundo a seguir, solicita um enorme empenho computacional.

Assim, buscando aprimorar a precisão do algoritmo, contudo sem deixar que se torne irrealizável o processamento, foi introduzido ao código dados de características temporais, retirados à partir do banco de dados original e dados de características do PSD, à partir das amostras normalizadas do fragmento de frequências do PSD, esses dados foram criados por (LUCAS, 2021) e reutilizados na pesquisa presente.

3.3.1 Dados extraídos com características do Tempo

Os dados complementares são características de tempo, calculadas por meio de parâmetros estatísticos dos dados originais. Ao todo, são 11 parâmetros: Valor Máximo, Curtose, Desvio médio absoluto, Desvio mediano absoluto, Desvio padrão, Variância,

Energia Total, Média aparada, Valor pico a pico, Média e Valor Mínimo (LUCAS, 2021). Para a amostra ‘1’ do banco de dados, os valores encontrados estão expostos na tabela 2.

Tabela 2 – Características do Tempo para a amostra 1 do banco de dados.

Parâmetro	Valor
Valor Máximo	3.81390000e+02
Curtose	-1.82088986e-02
Desvio Médio Absoluto	2.62000000e+00
Desvio Mediano Absoluto	3.31151934e+00
Desvio Padrão	4.27513733e+00
Variância	1.82767992e+01
Energia Total	1.36627940e+05
Média Aparada	3.69469110e+02
Valor pico a pico	2.13100000e+01
Média	3.69568950e+02
Valor Mínimo	3.60080000e+02

Fonte: (LUCAS, 2021).

3.3.2 Dados extraídos com de características do PSD

Os dados extraídos do PSD foram gerados de forma muito parecida com os dados anteriores, porém as características estatísticas utilizadas nesse caso, são referentes ao campo da frequência. Ao todo, foram 11 parâmetros, sendo elas: Curtose, Desvio mediano absoluto, Desvio médio absoluto, Desvio Padrão, Variância, Energia total, Média aparada, Valor pico a pico, Média, Valor RMS e Zero cross (LUCAS, 2021). Para a amostra ‘1’ do banco de dados, os valores encontrados estão expostos na tabela 3.

Tabela 3 – Características do PSD para a amostra 1 do banco de dados.

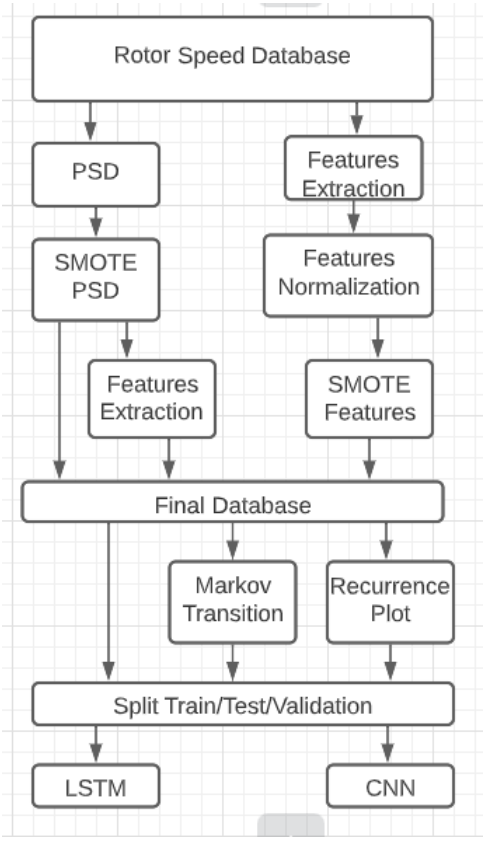
Parâmetro	**Valor**
Curtose	-1.36620153
Desvio Mediano Absoluto	2.8208194
Desvio Médio Absoluto	2.3242139
Desvio Padrão	2.74286074
Variância	7.52328506
Energia Total	8.33816268
Média Aparada	-0.89248167
Valor pico a pico	7.85386496
Média	-0.87209658
Valor RMS	2.78967474
Zero cross	3

Fonte: (LUCAS, 2021).

4. ABORDAGEM PROPOSTA

Nesta seção é descrita a metodologia utilizada para introduzir os dados produzidos às redes neurais usadas neste trabalho. Na figura 27 é ilustrado um fluxograma com todos os passos da metodologia. Haverá também uma concisa elucidação de como os testes foram concretizados para os mesmos serem expostos no capítulo seguinte.

Figura 27 – Fluxograma do projeto.



Fonte: Autor.

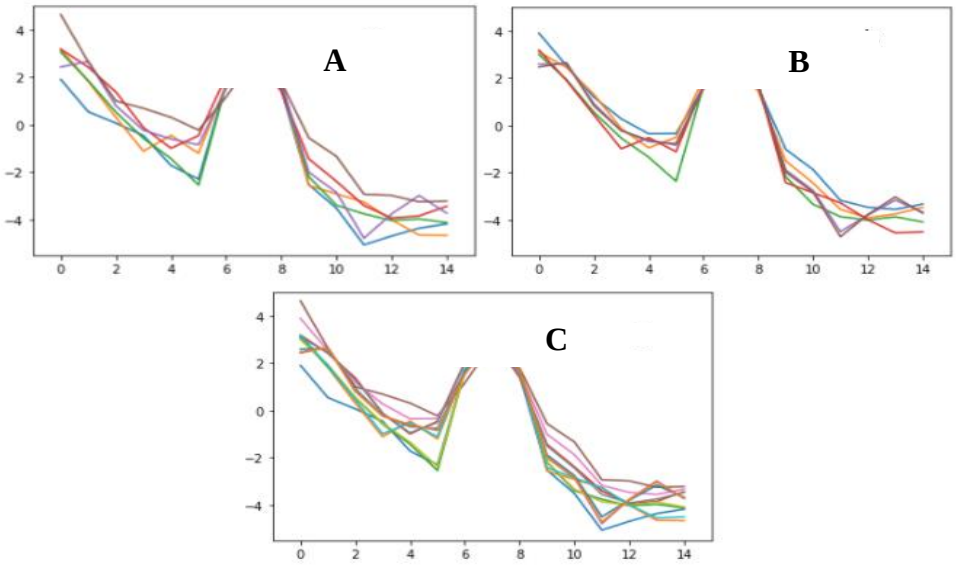
4.1 FLUXO DE PROJETO

Inicialmente, foi usada a biblioteca Pandas para transportar o banco de dados constituído de 240001 features em 2223 amostragens. Valores esses que são relacionados à velocidade do rotor através do tempo. Depois de esse método ser concretizado, houve a retirada de amostras que escapavam dos limites determinados, sobrando 1574 amostras decompostas em 11 classes. Assim, foi feita a extração de características inicial.

Depois da extração de características, foi computado o PSD de cada amostra, procedendo em 1574 PSD’s decompostos em 11 classes. Para equilibrar as classes foi feita

geração de amostras sinópticas a partir do procedimento SMOTE, gerando assim, 2448 amostras espalhadas em 11 classes. Levando em conta o foco do trabalho que é a computação do desbalanço absoluto, não ponderando se é acima ou abaixo de uma classe balanceada, as classes utilizadas no experimento foram então ajuntadas pelo valor absoluto de seu desbalanceamento. Ficando assim, por exemplo, a classe de 1 formada pelos valores da classe de -1% e da classe de 1%, a classe de 2 formada pelos valores da classe de -2% e da classe de 2% e assim sucessivamente. Com união de tais classes formou-se um dataset constituído por 6 classes, 5 classes desbalanceadas e 1 uma classe balanceada. O funcionamento do SMOTE nos dados está ilustrado na figura 28.

Figura 28 – A) Os Dados Originais; b) Os valores produzidos pelo SMOTE e c) O banco de dados com todas as amostras reunidas.

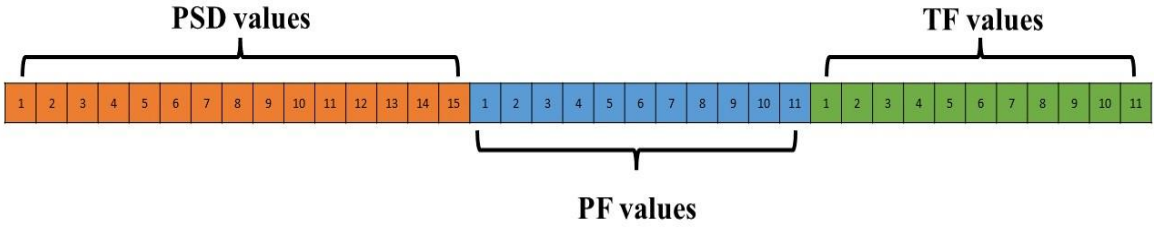


Fonte: Autor.

.

Assim foi criado um banco de dados constituído por 15 dados do PSD, extraídos da deformação do sinal em torno da frequência 0.33Hz, 11 dados de características extraídos do tempo e mais 11 dados de características extraídos do PSD. Podemos observar na figura 29 como ficaram distribuídos os valores no banco de dados.

Figura 29 – Banco de dados formado pelos valores do PSD, das características do PSD (PF Values) e das características do tempo (TF Values).

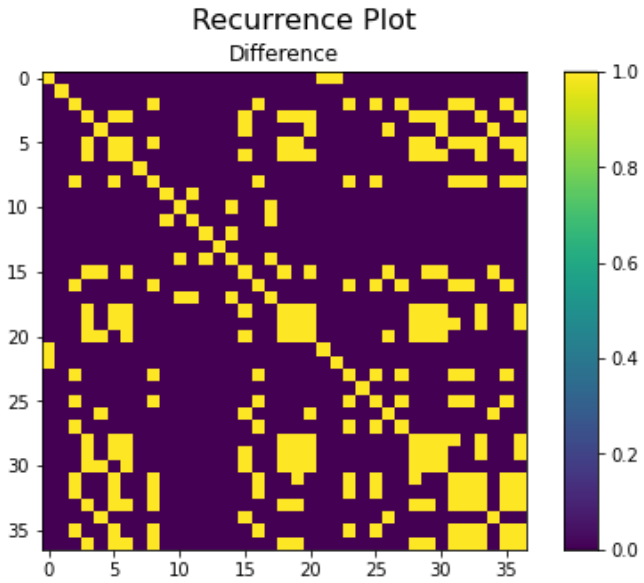


Fonte: (LUCAS, 2021).

4.1.1 Rede Neural Convolucional

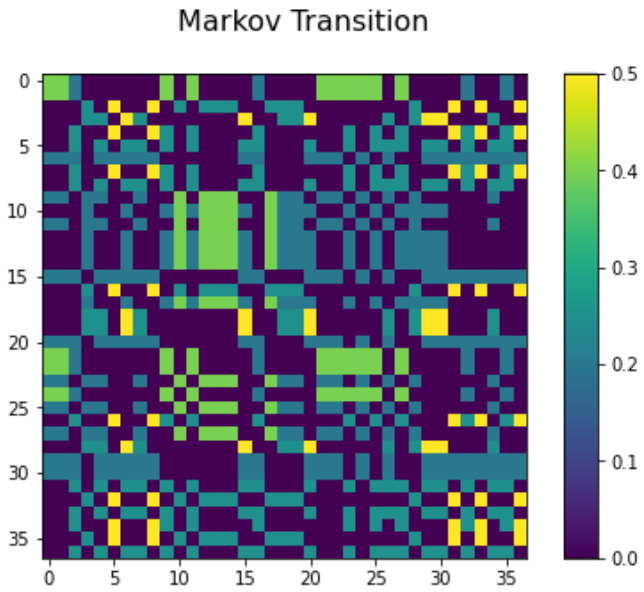
Para a utilização eficiente da CNN, esse dataset formado por 2448 amostras com 37 features é alterado em 2448 imagens de proporções 37x37 pelos algoritmos de Recurrent Plot e Markov Transition. Abaixo nas figuras 30 e 31, podemos ver uma representação final como essas imagens geradas ficam quando sobrepostas pelas 6 classes juntas.

Figura 30 – Recurrent Plot aplicado a 6 classes.



Fonte: Autor.

Figura 31 – Markov Transition Field aplicado a 6 classes

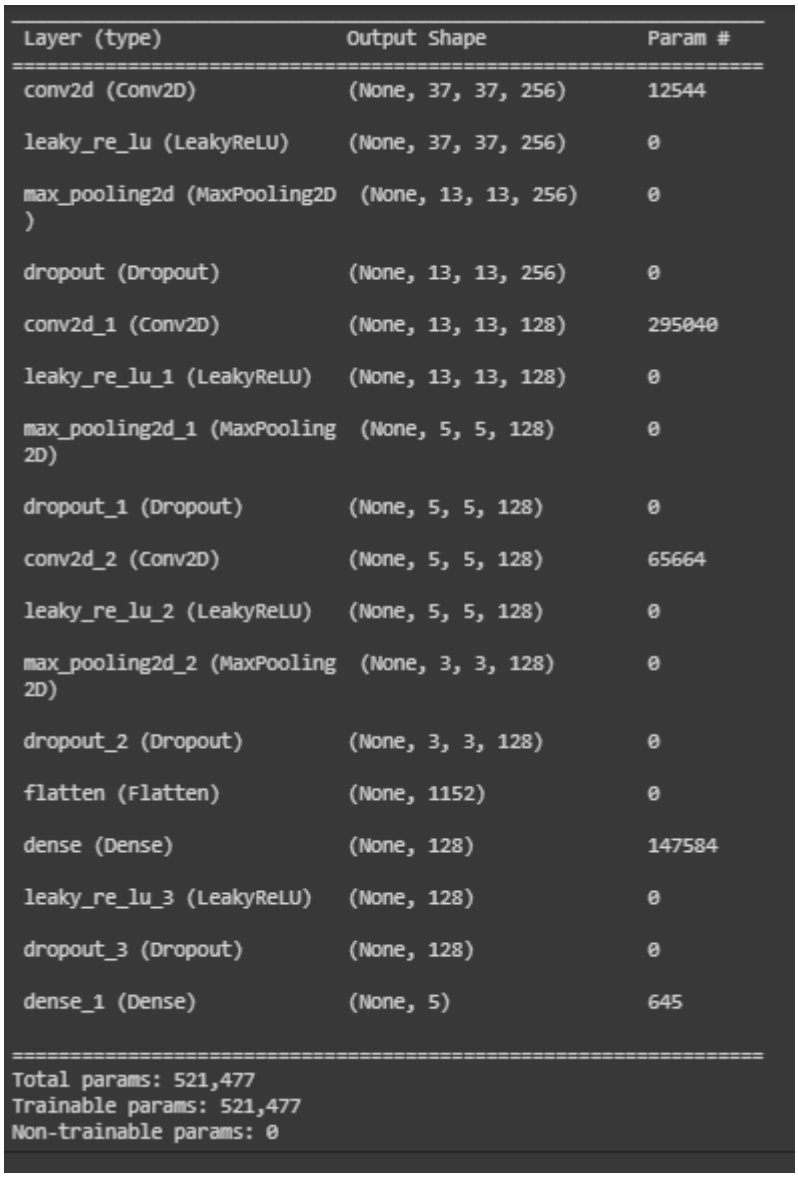


Fonte: Autor.

Essas imagens são decompostas em três sets: set de treinamento, set de teste e set de validação. O set de treinamento é formado por 68% dos valores, o set de validação é formado por 12% dos valores e o set de teste é formado por 20% dos valores. Essas percentualidades são apropriadas para o episódio de 6 classes em utilização, mas observaremos mais a frente que, a medida que o número de classes diminui, surge a necessidade de uma adequação à essa proporção para haver resultados mais precisos.

A CNN utilizada no estudo é formada por 3 layers convolucionais, 3 layers de Max Pooling, 4 layers de Dropout, 4 layers de ativação e dois layers Fully-connected, uma rede tal qual foi utilizada em (LUCAS, 2021). Na imagem 32, trazemos como ficou a rede convolucional no teste para 6 classes, vale ressaltar, que a rede sofre alteração de acordo com o número de classes na camada Dense quando os testes são realizados com menos classes envolvidas.

Figura 32 – Modelo de Rede Neural Convolucional



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 37, 37, 256)	12544
leaky_re_lu (LeakyReLU)	(None, 37, 37, 256)	0
max_pooling2d (MaxPooling2D)	(None, 13, 13, 256)	0
dropout (Dropout)	(None, 13, 13, 256)	0
conv2d_1 (Conv2D)	(None, 13, 13, 128)	295040
leaky_re_lu_1 (LeakyReLU)	(None, 13, 13, 128)	0
max_pooling2d_1 (MaxPooling 2D)	(None, 5, 5, 128)	0
dropout_1 (Dropout)	(None, 5, 5, 128)	0
conv2d_2 (Conv2D)	(None, 5, 5, 128)	65664
leaky_re_lu_2 (LeakyReLU)	(None, 5, 5, 128)	0
max_pooling2d_2 (MaxPooling 2D)	(None, 3, 3, 128)	0
dropout_2 (Dropout)	(None, 3, 3, 128)	0
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 128)	147584
leaky_re_lu_3 (LeakyReLU)	(None, 128)	0
dropout_3 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 5)	645

Total params: 521,477
Trainable params: 521,477
Non-trainable params: 0

Fonte: Autor.

Junto com a CNN, foram utilizados 9 classificadores comparativos. São eles: Nearest Neighbours, Linear SVM, RBF SVM, Decision Tree, Random Forest, Multi-layer Perceptron Neural Network, Ada Boost, Gaussian Naive Bayes e QDA (LUCAS, 2021). A proporção de treino e teste utilizados foi de 80% e 20%, respectivamente. Muitos tiveram melhor acurácia usando somente os valores do dataset do PSD, mas houve outros classificadores que se mostraram mais acurados com a adição dos dados de características. Na seção a seguir veremos melhor essas acurácias.

4.1.2 LSTM

No caso da rede LSTM, não foi necessária a utilização de imagens, porém as amostras passaram por reshape, primeiramente para 1 dimensão, somando as duas dimensões iniciais, logo após, foi introduzido um time step como segunda dimensão dos dados. Esse time step foi estabelecido em 15 em todos os testes, por neste valor apresentou os melhores resultados. Assim, com duas dimensões estabelecidas foi adicionada mais uma terceira dimensão no valor de 1, para facilitação no processo do algoritmo.

Depois dos dados estarem devidamente preparados, passou-se então a rede neural LSTM, que foi constituída por 2 layers de LSTM (uma com 150 neurônios e outra com 96 neurônios) e 2 layers de dropout, como podemos observar na figura 33, o estágio final da rede quando utilizada com 6 classes em teste.

Figura 33 – Modelo da rede LSTM

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_4 (LSTM)               (16516, 15, 150)          91200

 dropout_4 (Dropout)         (16516, 15, 150)          0

 lstm_5 (LSTM)               (16516, 96)               94848

 flatten_2 (Flatten)         (16516, 96)               0

 dense_4 (Dense)             (16516, 128)              12416

 leaky_re_lu_2 (LeakyReLU)   (16516, 128)              0

 dropout_5 (Dropout)         (16516, 128)              0

 dense_5 (Dense)             (16516, 3)                387

=================================================================
Total params: 198,851
Trainable params: 198,851
Non-trainable params: 0
_________________________________________________________________
```

Fonte: Autor.

Da mesma forma que ocorreu com a rede CNN, poderemos observar mais para frente que tiveram haver pequenas alterações na rede quando o número de classes foi alterado.

Por fim, além da LSTM, foram utilizadas métricas de aprendizado de máquina para a comprovação da eficiência da rede, métricas essas como: Pontuação F1, Acurácia, Precisão e Recall.

5. RESULTADOS

Os resultados serão decompostos pela quantidade de classes utilizadas e cada uma dessas subdivisões terão 2 tipos de processamento para comparação. Com a rede em CNN foram usados dados formados pelo PSD e dados pelo PSD, adicionando os dados de características do PSD e de características do tempo. Para os dados retirados da rede LSTM, foram usados dados formados pelo PSD, dividindo os resultados em teste e treino, com as métricas de avaliação adicionadas.

5.1 RESULTADOS OBTIDOS COM A CNN

5.1.1 3 classes: 5% de desbalanço, 3% de desbalanço e balanceado

Com 3 classes conseguimos obter o melhor desempenho dos algoritmos implementados. Foram feitas pequenas modificações na rede convolucional da figura 30, para conseguirmos encaixar perfeitamente os dados agregados de 3 classses na CNN empregada, mantendo a divisão de 68% dos dados repartidos em treino, 12% em validação e 20% em testes.

A metodologia utilizada nos testes foi primeiramente aplicar Recurrent Plot para geração de imagens com os dados extraídos somente do PSD. Logo após, é inserido no algoritmo os dados de características tanto do tempo quanto da frequência. Então, substituímos Recurrent Plot por Markov Transitition e novamente inserimos, primeiramente, somente os dados extraídos do PSD e, por fim, acrescemos os dados de características do tempo e da frequência. Pelo número reduzido de somente 3 classes em uso, todos os algoritmos tiveram um desempenho superior a 94% de precisão para os dados do PSD. Acrescemos a tabela da CNN, dados comparativos complementares utilizando Gramian Angular Metrics, algoritmo esse, utilizado em outras pesquisas referentes ao mesmo assunto aqui estudado. As acurácias geradas pelas CNN’s e pelos atuadores classificatórios estão apresentadas na tabela 4 e 5, respectivamente:

Tabela 4 - Precisões atingidas pela CNN

CNN	Acurácia
Dados PSD + RP	77,94%
Dados PSD + características + RP	97,96%
Dados PSD + MT	72,55%
Dados PSD + características + MT	90,61%
Dados PSD + GAM	70,38%
Dados PSD + características + GAM	96,00%

Fonte: Autor

Tabela 5 - Precisões resultantes dos testes desenvolvido para 3 classes.

Classificador	Dados PSD + RP	Dados PSD + Caracterísitcas + RP	Dados PSD + MT	Dados PSD + Caracterísitcas + MT
Linear SVM	99,02%	98,78%	99,51%	98,78%
RBF SVM	99,02%	98,78%	97,05%	98,78%
MLP Neural Network	97,06%	99,18%	98,04%	99,18%
Naïve Bayes	99,02%	99,18%	99,02%	99,18%
Decision Tree	99,51%	97,55%	99,02%	97,55%
Nearest Neighboor	96,08%	96,73%	97,06%	96,73%
QDA	99,02%	99,59%	99,02%	99,59%
Ada Boost	99,02%	96,33%	92,65%	96,33%
Random Forest	94,61%	90,61%	96,33%	90,61%

Fonte: Autor

Pelo fato do número de classes utilizadas ser de apenas 3, os classificadores puderam atingir uma precisão muito elevada. O maior desempenho atingido foi de 99,59% de precisão, que foi alcançado pelo classificador QDA com os dados do PSD, mais de características do tempo e da frequência, tanto utilizando Recurrent Plot quanto Markov Transition. Já a CNN apresentou uma grande diferença de desempenho em com a adição dos dados de características do tempo e da frequência, chegando até 97,96%. Na tabela 6 colocamos uma junção dos melhores resultados atingidos e ficou assim:

Tabela 4 - Melhores resultados para 3 classes

Classificador	Acurácia
Linear SVM e Decision Tree	99,51%
MLP Neural Network e Naïve Bayes	99,18%
Nearest Neighboors	97,06%
Decision Tree	97,95%
Random Forest	94,61%
RBF SVM e AdaBoost	99,02%
QDA	99,59%
Rede Neural Convolucional	97,96%

Fonte: Autor

5.1.2 4 classes: 5% de desbalanço, 3% de desbalanço, 1% de desbalanço e balanceado

Como era de se esperar, com 4 classes tivemos um desempenho do algoritmo um pouco mais baixo que o desempenho obtido utilizando 3 classes, porém ainda conseguimos obter acurácias bem satisfatórias, muitas acima de 94%. Foi mantida a divisão de 68% dos dados em treino, 12% em validação e 20% em testes. Adotando a mesma metodologia do teste com 3 classes de desbalanceamento, obtivemos os resultados da CNN na tabela 7 e 8 abaixo:

Tabela 7 - Precisões atingidas pela CNN

CNN	Acurácia
Dados PSD + RP	65,03%
Dados PSD + características + RP	90,21%
Dados PSD + MT	57,34%
Dados PSD + características + MT	74,01%
Dados PSD + GAM	55,40%
Dados PSD + características + GAM	89,60%

Fonte: Autor

Tabela 8 - Precisões resultantes dos testes desenvolvido para 4 classes.

Classificador	**Dados PSD + RP**	**Dados PSD + Caracterísitcas + RP**	**Dados PSD + MT**	**Dados PSD + Caracterísitcas + MT**
Linear SVM	93,01%	94,50%	93,01%	94,50%
RBF SVM	91,26%	91,44%	90,21%	92,12%
MLP Neural Network	92,31%	94,19%	89,16%	94,11%
Naïve Bayes	86,36%	88,38%	86,01%	88,47%
Decision Tree	94,06%	91,74%	93,36%	92,42%
Nearest Neighboor	86,71%	82,67%	85,66%	82,57%
QDA	94,06%	93,58%	89,51%	93,58%
Ada Boost	92,31%	68,20%	85,31%	69,15%
Random Forest	88,46%	77,68%	93,01%	78,21%

Fonte: Autor

Apesar da queda nas precisões obtidas, conseguimos retirar bons desempenhos classificatórios do algoritmo. O maior desempenho atingido foi de 94,50% de precisão, que foi alcançado pelo classificador Linear SVM com os dados do PSD, mais de características do tempo e da frequência, tanto utilizando Recurrent Plot quanto Markov Transition. A CNN teve uma resposta muito positiva em relação a adição de dados de características ao algoritmo, obtendo uma diferença de mais de 25% no caso do Recurrent Plot, em relação aos dados extraídos somente do PSD. Na tabela 9, novamente, temos uma amarração dos melhores resultados atingidos e ficou assim:

Tabela 5 - Melhores resultados para 4 classes

Classificador	**Acurácia**
Linear SVM	94,50%
RBF SVM	92,12%
MLP Neural Network	94,19%
Naïve Bayes	88,47%
Decision Tree	94,06%
Nearest Neighboor	86,71%
QDA	94,06%
Ada Boost	89,51%
Random Forest	88,46%
Rede Neural Convolucional	90,21%

Fonte: Autor

5.1.3 5 classes: 5% de desbalanço, 3% de desbalanço, 2% de desbalanço, 1% de desbalanço e balanceado

Com a adição de mais uma classe no algoritmo, tivemos mais uma queda nas precisões se compararmos com o desempenho obtido com 4 classes, contudo conseguimos atingir algumas precisões elevadas, em torno de 90%. A divisão dos dados não foi alterada, de 68% em treino, 12% em validação e 20% em testes. Os resultados extraídos estão nas tabelas 10 e 11:

Tabela 10 – Precisões atingidas pela CNN

CNN	Acurácia
Dados PSD + RP	59,51%
Dados PSD + características + RP	89,46%
Dados PSD + MT	48,37%
Dados PSD + características + MT	73,28%
Dados PSD + GAM	53,30%
Dados PSD + características + GAM	85,50%

Fonte: Autor

Tabela 11 - Precisões resultantes dos testes desenvolvido para 4 classes.

Classificador	Dados PSD + RP	Dados PSD + Caracterísitcas + RP	Dados PSD + MT	Dados PSD + Caracterísitcas + MT
Linear SVM	91,30%	91,42%	90,22%	91,42%
RBF SVM	85,87%	86,27%	86,41%	86,27%
MLP Neural Network	87,50%	87,50%	86,14%	87,50%
Naïve Bayes	82,34%	81,86%	81,52%	81,86%
Decision Tree	89,95%	87,25%	91,58%	87,25%
Nearest Neighboor	82,61%	80,15%	84,78%	80,15%
QDA	93,21%	90,44%	92,66%	90,44%
Ada Boost	71,47%	69,61%	70,83%	69,61%
Random Forest	75,00%	70,83%	76,09%	70,83%

Fonte: Autor

Novamente o maior desempenho atingido foi de QDA, chegou a 93,21% com os dados do PSD, mais de características do tempo e da frequência, tanto utilizando Recurrent Plot. Na tabela 12, mais uma vez, temos uma vinculação com os resultados mais altos e ficou assim:

Tabela 6 - Melhores resultados para 5 classes

Classificador	Acurácia
Linear SVM	91,42%
RBF SVM	86,41%
MLP Neural Network	87,50%
Naïve Bayes	82,34%
Decision Tree	91,58%
Nearest Neighboor	84,78%
QDA	93,21%
Ada Boost	71,47%
Random Forest	76,09%
Rede Neural Convolucional	89,46%

Fonte: Autor

5.1.4 6 classes: 5% de desbalanço, 4% de desbalanço, 3% de desbalanço, 2% de desbalanço, 1% de desbalanço e balanceado

Por fim, utilizamos todas as 6 classes de desbalanceamento no algoritmo e por consequência, tivemos as menores acurácias como esperado. A divisão dos dados não foi alterada, de 68% em treino, 12% em validação e 20% em testes. Seguindo os mesmos passos usados anteriormente tivemos os resultados retirados da CNN na tabela 13 e 14:

Tabela 13 - Precisões atingidas pela CNN

CNN	Acurácia
Dados PSD + RP	28,95%
Dados PSD + características + RP	87,35%
Dados PSD + MT	44,77%
Dados PSD + características + MT	69,80%
Dados PSD + GAM	40,67%
Dados PSD + características + GAM	82,80%

Fonte: Autor

Tabela 14 - Precisões resultantes dos testes desenvolvido para 6 classes.

Classificador	**Dados PSD + RP**	**Dados PSD + Caracterísitcas + RP**	**Dados PSD + MT**	**Dados PSD + Caracterísitcas + MT**
Linear SVM	88,86%	89,59%	88,20%	89,59%
RBF SVM	83,07%	83,87%	81,07%	83,88%
MLP Neural Network	85,08%	88,86%	83,96%	88,36%
Naïve Bayes	75,50%	70,00%	73,50%	70,00%
Decision Tree	85,97%	86,32%	83,80%	86,33%
Nearest Neighboor	77,73%	75,30%	79,51%	75,31%
QDA	87,97%	87,75%	88,20%	87,75%
Ada Boost	58,57%	58,59%	58,57%	58,57%
Random Forest	64,08%	64,09%	71,94%	64,08%

Fonte: Autor

Mais uma vez a maior performance alcançada foi de Linear SVM, 89,59% com os dados do PSD, mais de características do tempo e da frequência, tanto utilizando Recurrent Plot quanto Markov Transition. Na tabela 15, novamente, unimos as maiores precisões atingidas para os testes realizados com 6 classes:

Tabela 7 - Melhores resultados para 6 classes

Classificador	**Acurácia**
Linear SVM	89,59%
RBF SVM	83,88%
MLP Neural Network	88,36%
Naïve Bayes	75,50%
Decision Tree	86,33%
Nearest Neighboor	79,51%
QDA	88,20%
Ada Boost	58,57%
Random Forest	71,94%
Rede Neural Convolucional	87,35%

Fonte: Autor

5.2 RESULTADOS OBTIDOS COM A LSTM

Para a utilização da rede LSTM tivemos uma aplicação direta dos dados à rede, ou seja, não utilizamos algoritmos de geração de imagens. Os testes foram realizados a partir da divisão dos dados em teste e treino, 30% e 70%, respectivamente.

5.2.1 Dados de treino

O algoritmo da rede LSTM foi aplicado nas classes de desbalanceamento na mesma metodologia de divisão utilizada pela rede CNN, ou seja, primeiramente foram utilizadas 2 classes de desbalanceamento (5% de desbalanceamento e a classe balanceada), 3 classes de desbalanceamento (5% de desbalanceamento, 3% de desbalanceamento e a classe balanceada), depois foram usadas 4 classes de desbalanceamento (5% de desbalanceamento, 3% de desbalanceamento, 1% de desbalanceamento e a classe balanceada) e assim por diante. Os resultados obtidos pela rede neural seguem na tabela 16 abaixo:

Tabela 8 - Precisões obtidas pela LSTM por classes desbalanceadas

Número de classes	LSTM
2 classes	95,02%
3 classes	94,23%
4 classes	92,23%
5 classes	90,34%
6 classes	88,85%

Fonte: Autor

Logo após aplicamos as métricas de aprendizado de máquina em cada conjunto de classes de desbalanceadas utilizadas acima. Os resultados obtidos pelas métricas se encontram na tabela 17 abaixo:

Tabela 9 - Precisões obtidas pela LSTM por classes desbalanceadas

Nº de classes	Acurácia	F1 Score	Precisão	Recall
2 classes	93,20%	92,99%	93,19%	92,83%
3 classes	92,75%	92,90%	92,99%	92,66%
4 classes	91,64%	91,95%	92,06%	91,92%
5 classes	90,79%	90,84%	90,80%	90,90%
6 classes	88,98%	89,22%	89,17%	89,30%

Fonte: Autor

5.2.2 DADOS DE TESTE

Para realizarmos uma comparação com os resultados obtidos anteriormente, utilizamos os dados de teste, que correspondem a 30% do total dos dados utilizados. Naturalmente, a precisão obtida pela rede será reduzida, pelo fato do número de dados ser menor. Os resultados finais gerados pela rede neural seguem na tabela 18 abaixo:

Tabela 10 - Precisões obtidas pela LSTM por classes desbalanceadas

Número de classes	**LSTM**
2 classes	90,24%
3 classes	83,59%
4 classes	81,33%
5 classes	79,77%
6 classes	77,41%

Fonte: Autor

Logo após aplicamos as métricas de aprendizado de máquina em cada conjunto de classes de desbalanceadas utilizadas acima. Os resultados obtidos pelas métricas se encontram na tabela 19 abaixo:

Tabela 11 - Precisões obtidas pela LSTM por classes desbalanceadas

Nº de classes	**Acurácia**	**F1 Score**	**Precisão**	**Recall**
2 classes	91,25%	91,30%	91,15%	91,66%
3 classes	90,71%	91,08%	90,99%	91,23%
4 classes	89,30%	89,60%	89,51%	89,70%
5 classes	88,58%	88,95%	89,09%	88,84%
6 classes	86,18%	86,24%	87,01%	86,99%

Fonte: Autor

6. CONCLUSÃO

O objetivo proposto pelo trabalho era de apresentar um algoritmo classificador que conseguisse realizar o diagnóstico de desbalanceamento de massa em pás de turbinas eólicas. Foi também proposto que a abordagem ao problema fosse feita de maneira inovativa, fugindo de trabalhos anteriores com o mesmo banco de dados.

Pode-se dizer que ambos os objetivos foram concretizados ao longo do estudo. O levantamento de dados e resultados foi exaustivo e minucioso, para que o panorama de acurácia fosse apresentado da maneira mais completa possível. Com a abordagem adotada, grande parte dos algoritmos convencionais conseguiu obter bons resultados na classificação dos dados.

Mesmo assim, a Rede Neural Convolucional projetada conseguiu se destacar dentre os classificadores. A CNN foi o classificador que obteve melhores resultados de acurácia, isoladamente, para os cenários de 4 classes, 5 classes e 6 classes. Ela ainda foi, de maneira conjunta com Linear SVM, o melhor classificador para 3 classes.

O resultado de 91,84% é bem expressivo, considerando uma classificação entre 6 classes de desbalanço de massa. Se analisarmos a matriz de confusão, percebemos que a classe mais crítica foi a de 1%, o que, em situações reais, é um desbalanceamento extremamente difícil de ser diagnosticado e muitas vezes é ignorado.

Este resultado pode ser atribuído a duas técnicas utilizadas ao longo desse estudo: SMOTE e a extração de características. O SMOTE demostrou-se relevante para todos os classificadores testados, melhorando os resultados de acurácia para todos os cenários. A extração de características se mostrou extremamente relevante para a CNN. A concatenação dos parâmetros estatísticos calculados com os dados do PSD fez com que a CNN se transformasse de um dos piores classificadores naquele que obteve os melhores resultados. Logo, demonstra-se que o êxito em alcançar os objetivos específicos propostos foi essencial nos resultados positivos do estudo.

Para trabalhos futuros, seria interessante confirmar a metodologia apresentada perante dados obtidos de uma turbina eólica real, para dessa maneira confirmar a CNN como uma excelente ferramenta de diagnóstico. Além disso, seria interessante explorar outros tipos de desbalanceamento, como apresentado em pesquisas anteriores, onde o ângulo das pás também foi levado em consideração.

Por fim, ainda em trabalhos futuros, a metodologia ainda poderia ser testada e aplicada a diferentes modelos de turbinas eólicas, com o intuito de analisar como os impactos do desbalanceamento iriam se manifestar em sistemas mais ou menos robustos.

REFERÊNCIAS

ROSA, L. D. da. **Desenvolvimento de interface para simulação de aerogeradores**. 2019.

KUSNICK, J.; ADAMS, D. E.; GRIFFITH, D. T. **Wind turbine rotor imbalance detection using nacelle and blade measurements.** Wind Energy, Wiley Online Library, v. 18, n. 2, p. 267–276, 2015.

Faruk Ugranlı, Engin Karatepe. **Optimal wind turbine sizing to minimize energy loss.** International Journal of Electrical Power & Energy Systems, Volume 53, 2013, Pages 656-663, ISSN 0142-0615.

Polinder, H., de Haan, S. W. H., Dubois, M. R., & (Han) Slootweg, J. G. (2005). **Basic Operation Principles and Electrical Conversion Systems of Wind Turbines**. EPE Journal, 15(4), 43–50.

HEIER, S. **Grid integration of wind energy: onshore and offshore conversion systems.** [S.l.]: John Wiley & Sons, 2014

WANG, J. et al. **An integrated fault diagnosis and prognosis approach for predictive maintenance of wind turbine bearing with limited samples**. Renewable Energy, Elsevier, v. 145, p. 642–650, 2020.

JR, L. M. **Repair of wind turbine blades: Review of methods and related computational mechanics problems.** Renewable energy, Elsevier, 2019.

J. Hsu, Y. Wang, K. Lin, M. Chen and J. H. Hsu. **Wind Turbine Fault Diagnosis and Predictive Maintenance Through Statistical Process Control and Machine Learning.** in IEEE Access, vol. 8, pp. 23427-23439, 2020.

Zexian Sun, Hexu Sun, **Health Status Assessment for Wind Turbine with Recurrent Neural Networks**. Mathematical Problems in Engineering, vol. 2018, Article ID 6972481, 16 pages, 2018.

Qiang Gao, Xinhong Wu, Junhui Guo, Hongqing Zhou, Wei Ruan. **Machine-Learning-Based Intelligent Mechanical Fault Detection and Diagnosis of Wind Turbines**. Mathematical Problems in Engineering, vol. 2021, Article ID 9915084, 11 pages, 2021.

G.R. Hübner, H. Pinheiro, C.E. de Souza, C.M. Franchi, L.D. da Rosa, J.P. Dias.**Detection of mass imbalance in the rotor of wind turbines using Support Vector Machine.** Renewable Energy, Volume 170, 2021, Pages 49-59, ISSN 0960-1481.

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.(a)

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. (b)

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. (a)

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. (b)

Youngworth, R. N., Gallagher, B. B., & Stamper, B. L. (2005). **An overview of power spectral density (PSD) calculations.** Optical Manufacturing and Testing VI.

WELCH, P. **The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms.** IEEE Transactions on audio and electroacoustics, IEEE, v. 15, n. 2, p. 70–73, 1967.

Haibo He, & Garcia, E. A. (2009). **Learning from Imbalanced Data.** IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284.

N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer. **SMOTE: Synthetic Minority Over-sampling Technique.** Journal Of Artificial Intelligence Research, Volume 16, pages 321-357, 2002

O. Gecgel, S. Ekwaro-Osire, J. P. Dias, A. Serwadda, F. M. Alemayehu and A. Nispel. **Gearbox Fault Diagnostics Using Deep Learning with Simulated Data.** 2019 IEEE International Conference on Prognostics and Health Management (ICPHM), 2019, pp. 1-8.

Zhiguang Wang, Tim Oates, **Imaging Time-Series to Improve Classification and Imputation.** CoRR, abs/1506.00327, 2015. <http://arxiv.org/abs/1506.00327>.

Yegnanarayana, B. **Artificial neural networks for pattern recognition**. Sadhana 19, 189–238 (1994).

Ramchoun, H., Idrissi, M. A. J., Ghanou, Y., & Ettaouil, M. (2016). **Multilayer Perceptron: Architecture Optimization and Training**. *Int. J. Interact. Multim. Artif. Intell.*, 4(1), 26-30.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). **Understanding of a convolutional neural network.** 2017 International Conference on Engineering and Technology (ICET).

Apêndice A – Código Python para aplicação do Recurrent Plot e Markov Transition utilizando Google Colab:

```
!pip install pyts

import matplotlib.pyplot as plt

from mpl_toolkits.axes_grid1 import ImageGrid

from pyts.image import Recurrent Plot

from pyts.image import MarkovTransitionField

from pyts.datasets import load_gunpoint

import matplotlib.pyplot as plt
```

```
#Para utilização do Recurrent Plot
rp  = RecurrentPlot(threshold=’point’, percentage=20)
X_rp = rp.fit_transform(x)

#Para utilização do Markov Transition
mtf  = MarkovTransitionField(image_size=15)
X_mtf = mtf.fit_transform(x)



# Plot das imagens geradas
fig = plt.figure(figsize=(26, 13))
grid = ImageGrid(fig, 111,
                 nrows_ncols=(1, 1),
                 axes_pad=0.15,
                 share_all=True,
                 cbar_location="right",
                 cbar_mode="single",
                 cbar_size="7%",
                 cbar_pad=0.3,
                 )
#Para utilização do Recurrent Plot
images = [ X_rp[1]]
#Para utilização do Markov Transition
images = [ X_mtf[1]]

titles = [‘’]
for image, title, ax in zip(images, titles, grid):
    im = ax.imshow(image)
    ax.set_title(title, fontdict={'fontsize': 30})
ax.cax.colorbar(im)
ax.cax.toggle_label(True)
plt.suptitle(‘’, y=0.98, fontsize=16)
plt.show()
```

Apêndice B – Código Python para a implementação da Rede Neural Convolucional utilizando o Google Colab:

```
import keras
from keras.models import Sequential,Input,Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.advanced_activations import LeakyReLU

batch_size = 64
epochs = 110
num_classes = nClasses

fashion_model = Sequential()
fashion_model.add(Conv2D(256,kernel_size=(4,4),strides=(1,1),
activation='linear',padding='same',input_shape=(37,37,3)))
fashion_model.add(LeakyReLU(alpha=0.2))
fashion_model.add(MaxPooling2D((3, 3),padding='same'))
fashion_model.add(Dropout(0.3))
fashion_model.add(Conv2D(128,kernel_size=(3,3),strides=(1,1),
activation='linear',padding='same'))
fashion_model.add(LeakyReLU(alpha=0.2))
fashion_model.add(MaxPooling2D(pool_size=(3, 3),padding='same'))
fashion_model.add(Dropout(0.3))
fashion_model.add(Conv2D(128,kernel_size=(2,2),strides=(1,1),
activation='linear',padding='same'))
fashion_model.add(LeakyReLU(alpha=0.2))
fashion_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
fashion_model.add(Dropout(0.3))
fashion_model.add(Flatten())
fashion_model.add(Dense(128, activation='linear'))
fashion_model.add(LeakyReLU(alpha=0.2))
fashion_model.add(Dropout(0.3))
fashion_model.add(Dense(num_classes, activation='softmax'))
```

Apêndice C – Código Python para a implementação da LSTM utilizando o Google Colab:

```
import keras
from keras.layers import Dense, Dropout, Flatten
from keras.layers import LSTM

batch_size = 64
epochs = 110
num_classes = nClasses

model = Sequential()
model.add(LSTM(150, return_sequence=True))
model.add(Dropout(0.3))
model.add(LSTM(96))
model.add(Flatten())
model.add(Dense(128, activation='linear'))
model.add(LeakyReLu(alpha=0.2))
model.add(Dropout(0.3))
model.add(Dense(num_classes, activation='softmax'))
```