GRADUAÇÃO EM ENGENHARIA ELÉTRICA

LUAN FÁBIO MARINHO GALINDO

TRABALHO DE CONCLUSÃO DE CURSO

PREDIÇÃO DE DEMANDA DE ENERGIA ELÉTRICA UTILIZANDO ALGORITMOS DE APRENDIZADO DE MÁQUINA

Campina Grande, Paraíba, Brasil
2023

LUAN FÁBIO MARINHO GALINDO

PREDIÇÃO DE DEMANDA DE ENERGIA ELÉTRICA UTILIZANDO ALGORITMOS DE APRENDIZADO DE MÁQUINA

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Eletrotécnica

Professor Célio Anésio da Silva, D.Sc.
Orientador

Campina Grande, Paraíba, Brasil
2023

LUAN FÁBIO MARINHO GALINDO

PREDIÇÃO DE DEMANDA DE ENERGIA ELÉTRICA UTILIZANDO ALGORITMOS DE APRENDIZADO DE MÁQUINA

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Eletrotécnica

Aprovado em / /

Professor Jalberth Fernandes de Araújo, D. Sc.
Universidade Federal de Campina Grande
Avaliador

Professor Célio Anésio da Silva, D. Sc.
Universidade Federal de Campina Grande
Orientador

Dedico este trabalho a minha família e amigos, que me apoiaram nos momentos mais difíceis dessa trajetória, em particular, *in memoriam*, ao meu avô Francisco Galindo de Souza, que sonhou com o final dessa etapa de minha vida provavelmente mais do que eu.

AGRADECIMENTOS

Agradeço a minha mãe Luciene e minha avó Lourdes, em primeiro lugar, pelo incansável apoio e esforço para que eu pudesse chegar até aqui. Nada me deu mais forças do que os conselhos de vocês.

Agradeço também à minha avó Jovenice, ao meu avô Francisco Jacó, ao meu bisavô Cícero, *in memoriam*, e aos meu irmãos João Lucas e Lara, pelo encorajamento e pelo carinho que vocês sempre me deram.

Agradeço também ao restante da minha família e aos meus amigos, que com toda consideração e suporte, sempre me incentivaram a perseverar durante essa árdua jornada.

Enfim, agradeço a todos que de alguma forma, passaram pela minha vida e contribuíram para que eu me tornasse o homem que sou hoje.

“O que é essa breve vida mortal, senão a busca incessável por um legado?”

A Casa do Dragão. (Adaptado).

RESUMO

Este trabalho visa utilizar técnicas de Ciência dos Dados e algoritmos de Aprendizagem de Máquina para prever a demanda de energia elétrica no Brasil de janeiro a março de 2023. Técnicas de predição de demanda de energia elétrica são fundamentais para a logística de funcionamento do Sistema Elétrico Nacional, pois assim a gestão desses recursos é feita de forma mais eficiente e atenua desperdícios. Diante do exposto, este trabalho tem como objetivos específicos coletar e tratar uma base de dados para o uso no treinamento, realizar o próprio e em seguida analisar os resultados. Para isso, usar-se-á a linguagem *Python*, em conjunto com a Interface de Programação de Aplicações (API, em inglês) *Tensor Flow*, na plataforma de desenvolvimento Anaconda e no Ambiente de Desenvolvimento Integrado (*IDE*, em inglês) *Spyder*. A base de dados utilizada será obtida pelo Operador Nacional do Sistema Elétrico (ONS), pela Câmara de Comercialização de Energia Elétrica (CCEE) e pelo *Yahoo! Finances*. Por fim, este trabalho demonstrará o potencial do uso de técnicas de Aprendizagem de Máquina para a previsão de demanda de energia elétrica, evidenciando como o uso de dados históricos contribui para uma melhor gestão do Sistema Elétrico Nacional.

Palavras-chave: Ciência dos Dados. Aprendizagem de Máquina, Sistema Elétrico Nacional, *Python*, *Tensor Flow*, ONS, CCEE.

ABSTRACT

This work aims to use Data Science techniques and Machine Learning algorithms to predict the electricity demand in Brazil from January to March 2023. Electricity demand prediction techniques are fundamental for the operational logistics of the National Electric System, in a way that the management of those resources are handled in a more efficiently way and reduces waste. In view of the above, this work has the specific objectives of collecting and processing a database for use in training, carrying out the training itself and then analyze the results. For this, the Python programming language will be used, together with the Application Programming Interface (API) Tensor Flow, in the Anaconda development platform with the Integrated Development Environment (IDE) Spyder. The database used will be obtained by the National Electric System Operator (ONS), by the Electric Energy Commercialization Chamber (CCEE) and by Yahoo! Finances. Finally, this work will demonstrate the potential of using Machine Learning techniques to forecast electricity demand, showing how the use of historical data contributes to a better management of the National Electric System.

Keywords: Data Science, Machine Learning, National Electric System, Python, Tensor Flow, ONS, CCEE.

LISTA DE ILUSTRAÇÕES

Figura 1 – Rede Neural Multicamadas 17
Figura 2 – Exemplo ilustrativo do comportamento de uma RNA 18
Figura 3 – Exemplo de *dataset* e sua codifição 20
Figura 4 – Tensor de 3 eixos 25
Figura 5 – Composição de vetores que influenciam a demanda 26
Figura 6 – Demanda do conjunto de treino 30
Figura 7 – Demanda do conjunto de teste 32
Figura 8 – CMNEE do conjunto de treino 33
Figura 9 – CMNEE do conjunto de teste 34
Figura 10 – Conjunto de treinamento do IBOV 35
Figura 11 – *Pairplot* dos conjuntos de treino 36
Figura 12 – Evolução do algoritmo 37
Figura 13 – Resultados do treinamento 38
Figura 14 – Histograma da quantidade de erros 40
Figura 15 – Histograma dos erros percentuais 41

LISTA DE TABELAS

Tabela 1 - Especificações de *hardware* utilizado 28
Tabela 2 – Especificações de *software* utilizado 29
Tabela 3 – Amostra dos dados de treinamento da Demanda sem tratamento 31
Tabela 4 – Amostra dos dados de treinamento da Demanda tratados 31
Tabela 5 – Amostra do conjunto de dados de treinamento do CMNEE 32
Tabela 6 – Amostra do conjunto de dados de treinamento do CMNEE tratado 33
Tabela 7 – Amostra das pontuações de treino do IBOV 34
Tabela 8 – Amostra das pontuações de treino do IBOV tratadas 35
Tabela 9 – Resultados em MW 39
Tabela 10 – Erros de previsão em MW 40
Tabela 11 – Erro percentual de previsão 42
Tabela 12 – Estatísticas dos erros percentuais de previsão 42

LISTA DE ABREVIATURAS E SIGLAS

ONS	Operador Nacional do Sistema Elétrico
SIN	Sistema Interligado Nacional
ML	*Machine Learning*
CCEE	Câmara de Comercialização de Energia Elétrica
KDE	*Kernel Density Estimation*
RNA	Rede Neural Artificial
EPE	Empresa de Pesquisa Energética
INMET	Instituto Nacional de Meteorologia
ANEEL	Agência Nacional de Energia Elétrica
CMNEE	Custo Médio Nacional de Energia Elétrica
IBOV	Índice BOVESPA
IA	Inteligência Artificial
API	Interface de Programação de Aplicações
IDE	Ambiente de Desenvolvimento Integrado

LISTA DE SÍMBOLOS

$f(x)$	Notação utilizada para representar uma função dependente da variável x
$i\ x\ j$	Notação para representar as dimensões de uma matriz com *i* linhas e *j* colunas

SUMÁRIO

1 Introdução 14
1.1 Objetivos 15
1.1.1 Objetivos específicos 15
2 Fundamentação Teórica 16
2.1 Algoritmos de aprendizagem de máquina e seu uso em predição de dados 16
2.2 Bibliotecas utilizadas 18
2.2.1 OS – *Miscellaneous operating system interfaces* v1.2.0 18
2.2.2 Pandas v1.3.5 19
2.2.3 *Matplot* v3.5.3 22
2.2.4 *Numpy* v1.21.5 e *Seaborn* v0.12.2 23
2.2.5 TensorFlow v2.10.0 24
3 Metodologia 26
3.1 Descrição e obtenção dos dados utilizados 27
3.2 Seleção dos hiperparâmetros do modelo 28
3.3 Especificações do *hardware* e *software* utilizados 28
4 Análise dos resultados 30
4.1 Apresentação e análise dos resultados obtidos 37
4.2 Limitações e sugestões para trabalhos futuros 43
5 Considerações Finais 44
Referências 45
Glossário 47
Apêndice A – Código de tratamento da demanda 48
Apêndice B – Código de tratamento dos custos 51
Apêndice C – Código de tratamento do índice IBOV 54
Apêndice D – Algoritmo DNN 56

1 INTRODUÇÃO

A capacidade energética é um dos principais indicadores de desenvolvimento socioeconômico de um país. Sendo assim, para uma gestão da energia elétrica eficiente, é fundamental que parâmetros como qualidade de energia (tensão, corrente e frequência próximos dos níveis nominais), estabilidade do sistema, velocidade e planejamento da expansão da rede e faturamento atrativo estejam em conformidade com os padrões exigidos pelos órgãos reguladores. Com isso em mente, é necessário que os dados relacionados com operação do sistema sejam analisados para garantir que as decisões tomadas pela administração estejam em alinhamento as necessidades do sistema.

Nesse contexto, surge a necessidade de prever a demanda de energia elétrica para atingir padrões de confiabilidade do sistema aceitáveis. De fato, de 2020 a 2026 é previsto um aumento de carga de 20% no Sistema Interligado Nacional (SIN) (ONS, 2021). Ademais, visto que a matriz energética brasileira é predominantemente hídrica (ONS, 2023), o país depende fortemente de condições climáticas relacionadas a chuvas, o que acrescenta a importância de estudos de previsibilidade de demanda elétrica. Com efeito, financeiramente, a previsão de demanda a longo prazo é necessária para a precificação e contratação de energia, enquanto no curto prazo é essencial para que o sistema trabalhe dentro de limites contratados, evitando assim, onerosas multas. Em verdade, exige-se que o erro de previsão esteja abaixo da tolerância de 5% (RUAS, BRAGATTO, *et al.*).

Sob essa óptica, a previsão de tendências de mercado e de perfil dos consumidores, tanto pequenos quanto grandes, permite um dimensionamento mais acurado para os setores críticos no sistema. Somando-se a isso, o mapeamento de fatores externos pode contribuir para a identificação de pontos críticos de estabilidade, prevendo, assim, situações nas quais contramedidas de controle farar-se-ão necessárias (FERREIRA, 2006).

Diante do exposto, o emprego de métodos de previsão usando Aprendizagem de Máquina (ML, *Machine Learning*) mostra-se como uma alternativa atrativa, devido a popularização crescente do uso desses algoritmos em séries temporais (RUAS, BRAGATTO, *et al.*). Além disso, a capacidade dos algoritmos de ML de se adaptarem aos casos expostos, com capacidade de aprender padrões não óbvios e realizar análise de

correlações e interdependências de forma contínua possibilita resultados melhores conforme o algoritmo é empregado (CARVALHO DE ALMEIDA e LIMA PASSARI, 2006).

1.1 OBJETIVOS

Este trabalho tem como objetivo geral o desenvolvimento de um algoritmo de ML capaz de prever a demanda de energia elétrica no SIN.

Para isso, dividir-se-á o trabalho em duas partes, igualmente críticas: obtenção e tratamento dos dados necessários, e em seguida treinamento do modelo.

1.1.1 OBJETIVOS ESPECÍFICOS

i. Obtenção de *datasets* em órgãos como ONS, Câmara de Comercialização de Energia Elétrica (CCEE) e na plataforma *Yahoo! Finances;*
ii. Realizar o tratamento dos dados, e fazer análises estatísticas com o intuito de procurar por padrões de comportamento;
iii. Realizar o treinamento do modelo por meio da biblioteca *TensorFlow*;

2 FUNDAMENTAÇÃO TEÓRICA

Pode-se entender a Ciência dos Dados como a área do conhecimento que busca extrair conhecimento a partir de dados desorganizados (GRUS, 2016). Atualmente, é uma das áreas mais promissoras na área da tecnologia, dado a popularização de dispositivos que geram e consomem dados. Em verdade, o Cientista de Dados é um dos profissionais mais requisitados em 2023 (LE WAGON, 2023).

A Ciência dos Dados é multidisciplinar, envolvendo conhecimentos na área de estatística, programação, matemática, negócios, entre outras. Dessa forma, ela possui diversas aplicações em múltiplas áreas, como marketing (por anúncios direcionados), saúde (por telemetria de dados fisiológicos), esportes (por dados de desempenho), *et cetera*. Assim, por uso de análise de dados, é possível identificar padrões e tendências nos dados, e com isso, realizar previsões e tomar decisões com base nesses resultados.

A análise de dados é realizada por técnicas como mineração de dados, aprendizagem de máquina e análise estatística (GRUS, 2016). Na engenharia elétrica, o uso dessas técnicas pode contribuir consideravelmente para melhorar o desempenho e eficiência de sistemas elétricos.

2.1 ALGORITMOS DE APRENDIZAGEM DE MÁQUINA E SEU USO EM PREDIÇÃO DE DADOS

Desde o surgimento da matemática o ser humano busca modelar o processo de funcionamento da natureza por meio de equações. E a partir disso, áreas do conhecimento que existem atualmente, como física, química, estatística e engenharia, se desenvolveram. Nesse contexto, graças ao avanço de poder computacional disponível atualmente, surgiu-se a ideia de entender padrões por meio dos modelos computacionais das Redes Neurais.

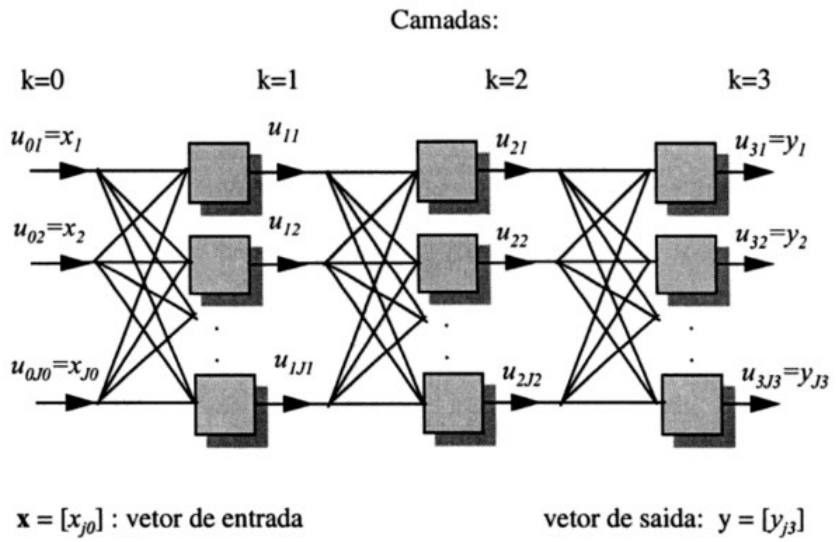
No campo da álgebra, entende-se como função a relação entre um conjunto de entrada de dados e um conjunto de saída de dados. Simbolicamente:

$$x \rightarrow f(x) \rightarrow y \tag{1}$$

O entendimento de como $f(x)$ pode ser descrito analiticamente nem sempre é possível, especialmente quando se trata de um problema extremamente complexo. Dessa maneira, assimilar $f(x)$ com base em métodos numéricos emerge como alternativa plausível de ser empregada, sendo isso o cerne do funcionamento dos algoritmos de ML.

Nesse contexto, o desenvolvimento das Redes Neurais Artificiais (RNA) baseou-se no comportamento do neurônio humano, que fundamenta seu funcionamento em estímulos de entrada chamados conexões sinápticas que resultam em uma única saída, sendo esta repassada para todos os outros neurônios convenientes. Ao longo do tempo, abordagens evoluíram essa ideia, dado origem a rede Perceptron, que é uma rede que possui neurônios dispostos em várias camadas, onde cada camada é responsável por um peso a ser repassado para a próxima (KOVÁCS, 2006), conforme ilustra a Figura 1.

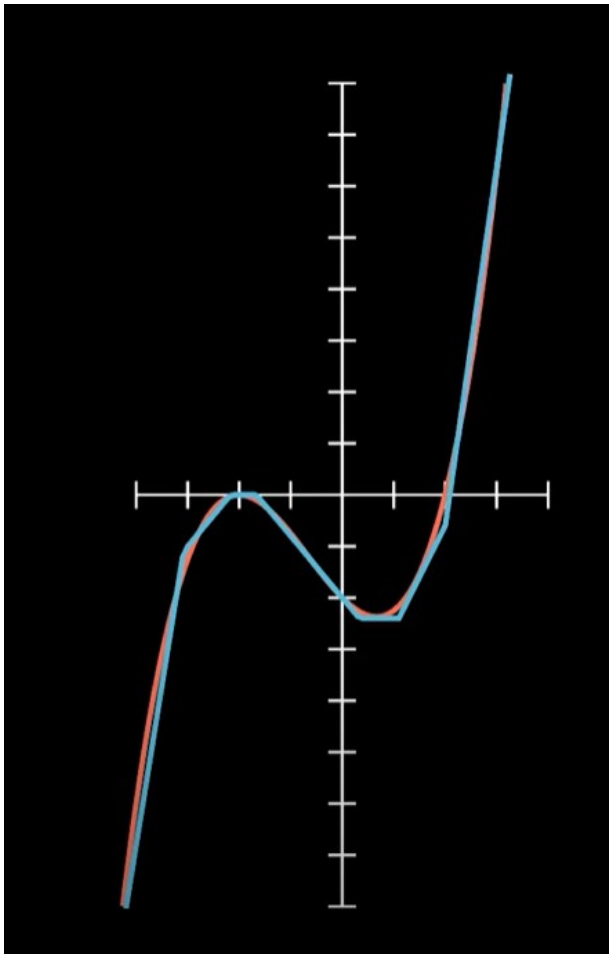
Figura 1 – Rede Neural Multicamadas



Fonte: (KOVÁCS, 2006)

Com isso, para que ocorra a excitação de cada neurônio individualmente, necessita-se de uma função de ativação, responsável por atribuir o peso que cada conexão sináptica irá exercer para influenciar na saída do neurônio. É por meio dessa função de ativação que o algoritmo converge para ser capaz de prever a saída da função sem ser capaz de descrevê-la analiticamente. A Figura 2 exemplifica o comportamento de uma RNA ao tentar prever os resultados de $f(x) = x^3 - x$, onde a linha laranja representa a função exata e a linha azul representa a aproximação do algoritmo.

Figura 2 – Exemplo ilustrativo do comportamento de uma RNA



Fonte: (ROBINSON, 2022)

Com base nisso e às custas de elevado poder computacional, as RNA's são capazes de prever funções consideravelmente complexas, sem a necessidade de ser capaz de saber exatamente a lei de formação dessas funções.

2.2 BIBLIOTECAS UTILIZADAS

2.2.1 OS – *MISCELLANEOUS OPERATING SYSTEM INTERFACES* v1.2.0

Esse módulo fornece uma maneira simples de usar funcionalidades que são dependentes do sistema operacional (PYTHON SOFTWARE FOUNDATION, 2023). Para este trabalho, utilizou-se uma função de especificação de caminho de diretório, para que o compilador não encontre erros ou ambiguidades durante a execução do código e carregamento dos arquivos para processamento. A dita função é *os.chdir(path)*, com o

seu argumento (*path*) sendo o caminho do diretório onde se encontram os arquivos que serão manipulados. O caminho deve estar entre aspas simples, e entre cada pasta deve haver contrabarras duplas, pois contrabarra simples é um caractere de escape, e assim o compilador pode não entender o caminho e acusar erro de compilação. A seguir, um exemplo de aplicação usando a sintaxe da função:

```
os.chdir('C:\\pasta_1\\pasta_2\\...\\pasta_n\\diretório')
```

2.2.2 PANDAS V1.3.5

Uma das bibliotecas mais conhecidas e utilizadas em Ciência dos Dados. Com ela, é possível a importação e exportação em diferentes formatos, como CSV, SQL e outros. Além disso, com a Pandas é possível a manipulação de objetos do tipo *Dataframe*, que são tabelas bidimensionais semelhantes a planilhas do Excel™. A Pandas permite operações como filtragem de dados, seleção de colunas ou linhas, ordenação dos dados, análises estatísticas e manipulação do tipo de dado, e.g., convertendo *floats* em *int* ou *double* ou transformando datas do tipo DD/MM/YYYY de *string* para um formato possível de ser manipulado no *Dataframe* (NUMFOCUS, 2023). A seguir, apresentar-se-á as funções da Pandas utilizadas neste trabalho.

i. Ler CSV

Função responsável por carregar os *datasets*, tratados ou não, para o compilador possibilitando, assim, que seja iniciada a manipulação dos dados. Seu argumento possui diversas possibilidades de entrada, sendo utilizadas nesse trabalho os argumentos "*filepath_or_buffer*", "*sep*", "*encoding*", "*header*" e "*decimal*".

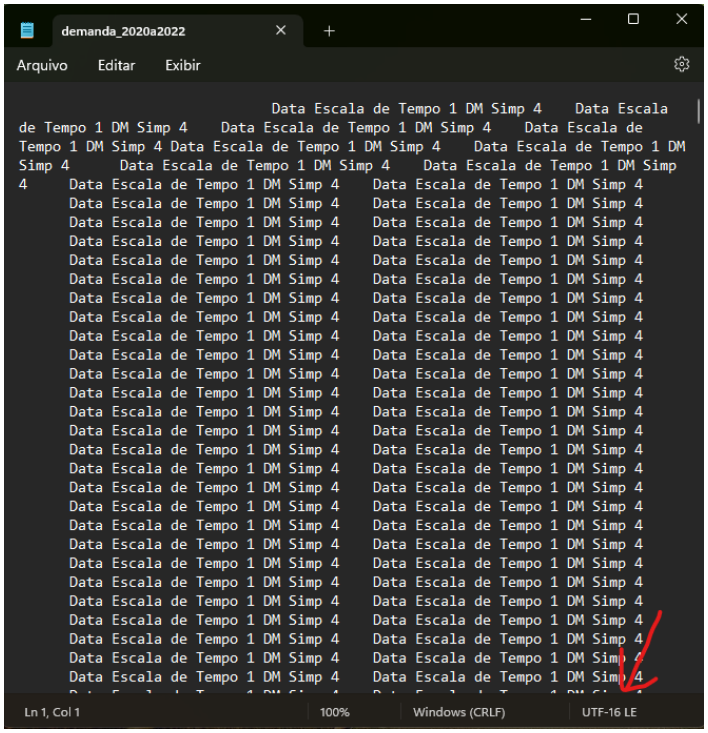
O argumento "*filepath_or_buffer*" é o nome do arquivo junto com sua extensão (e.g. arquivo.csv). Como utilizou-se o módulo OS para definir de início o diretório de trabalho do compilador, não é necessário inserir o local do arquivo a ser carregado, caso este esteja no diretório de trabalho. Caso contrário, é possível buscar o arquivo em outro diretório ou até mesmo na internet (NUMFOCUS, 2023).

O argumento "*sep*", que por *default* é atribuído a ",", é o delimitador para detectar a separação dos dados (NUMFOCUS, 2023). Por exemplo, há arquivos do tipo *.csv* nos quais a separação dos dados é feita por células, exigindo, assim, o uso do argumento *sep="\t"*; outros arquivos podem vir com separação via ";", motivando, então, o uso de

sep=";". Por esse motivo, é importante executar o arquivo antes de carregá-lo no compilador, pois assim evita-se erros difíceis de serem detectados. Isso porque o compilador carrega o arquivo (com o delimitador incorreto) e interpreta as propriedades dos dados de forma incoerente, gerando assim, erros na manipulação dos dados.

Sobre o argumento "*encoding*", que por *default* é configurado como *encoding="utf-8"*, refere-se à codificação do arquivo a ser carregado ou salvo (NUMFOCUS, 2023). Analogamente ao argumento *sep*, o uso incorreto da codificação gera erros na manipulação dos dados. Dessa forma, uma maneira de saber a codificação do arquivo é executando-o por meio do Bloco de Notas do Windows™, como mostrado na Figura 3, que ilustra a obtenção da codificação de um dos *datasets* utilizados no trabalho.

Figura 3 – Exemplo de *dataset* e sua codifição



Fonte: Autoria própria, 2023

No que tange o argumento "*header*", este define se a primeira linha do *Dataframe* deve ser usada como cabeçalho, por conter nomes das colunas (NUMFOCUS, 2023). Dessa maneira, usar essa função depende de uma análise preliminar do *Dataframe* antes de sua importação. Ademais, são possíveis algumas manipulações, como inserir uma lista de inteiros como os cabeçalhos a serem usados, o que não convém a este trabalho, sendo

aqui utilizado o argumento como "*header=None"* para que o cabeçalho do *dataset* seja ignorado devido a não-uniformidade dos *datasets* obtidos.

O argumento "*decimal*", por sua vez, é responsável por reconhecer valores decimais (e.g. na Europa, usa-se ',') (NUMFOCUS, 2023). Essa função também contribui para atenuar erros no tratamento dos dados, posto que nem todos os *datasets* usam o mesmo caractere para separar valores decimais.

Por fim, convém destacar a função *to_csv*, que é responsável por salvar um *Dataframe* no formato *.csv* (NUMFOCUS, 2023), onde os argumentos usados neste trabalho se resumiram ao nome do arquivo juntamente com *index=False*, para que o índice no arquivo salvo não seja impresso nos campos de índice, mas sim como linhas, por questão de boas práticas.

ii. *Drop*, *Reset index* e *Columns* (Colunas)

Drop, do inglês "deixar cair", remove rótulos (*labels*) especificados nas linhas ou colunas do *Dataframe* (NUMFOCUS, 2023). Essa função é bastante útil durante o momento de realizar a limpeza dos dados, possibilitando que diversos rótulos sejam excluídos de forma ágil, com o auxílio da função *range*, responsável por definir uma sequência de execução (PYTHON SOFTWARE FOUNDATION, 2023).

Reset index é responsável por reiniciar os índices do *Dataframe* para os valores padrão (NUMFOCUS, 2023). Essa função torna-se útil após diversas manipulações serem feitas no *Dataframe*, resultando em valores incoerentes nos índices (e. g. linha 9 após a 5). Por esse motivo, seu uso é uma questão de boa prática.

Columns (Colunas) é utilizada para renomear as colunas do *Dataframe* (NUMFOCUS, 2023), fazendo-se útil para que o analista possa realizar manipulações no *Dataframe* com maior confiabilidade acerca dos resultados esperados.

iii. *To datetime* e *As type*

To datetime é responsável por converter um escalar, *array*, *series*, ou *Dataframe* para um objeto de data manipulável pelo Pandas (NUMFOCUS, 2023). Utilizou-se essa função com o intuito de usar as datas de geração dos dados para o treinamento da Rede Neural, visando pesar a influência dos fatores sazonais de forma indireta na aprendizagem do algoritmo. Essa função possui como vantagem a possibilidade de formatar a data de

acordo com a preferência do usuário, *e.g.* DD/MM/YYYY ou MM/DD/YYYY (padrão americano), por uso do método *strftime*.

iv. *Iloc* e *apply*

A função *iloc* realiza a indexação baseada em inteiros para a seleção via posição (NUMFOCUS, 2023). Em outras palavras, realiza a localização dos objetos via a posição desses. Dessa forma, é usada em quando deseja-se realizar alterações em uma gama de objetos de forma ágil.

Sobre a função *apply*, esta realiza a aplicação de uma outra função sobre um eixo do *Dataframe*, podendo essa aplicação ser sobre uma linha ou coluna (NUMFOCUS, 2023). Seu uso é potencializado pelo *iloc*, durante a manipulação dos objetos de um *Dataframe*. Somado a isso, pode-se usar a expressão *lambda* para a criação de uma função anônima (PYTHON SOFTWARE FOUNDATION, 2023), aumentando a escalabilidade da operação a ser realizada.

v. *Concat* e *describe*

Acerca da função *concat*, esta realiza o concatenamento de objetos do Pandas ao longo de um eixo (NUMFOCUS, 2023). Desse modo, ao obter-se *datasets* diferentes com o mesmo tipo de objeto, é conveniente que o tratamento dos dados seja realizado em apenas um conjunto de objetos criado a partir de outros.

A função *describe* gera dados estatísticos descritivos de um *dataframe*, resumindo características como tendência central, dispersão e formato da distribuição dos dados (NUMFOCUS, 2023). Essa função é útil para analisar a contagem, média, desvio padrão, quartis[1] e valores de ponta (máximo e mínimo).

2.2.3 *MATPLOT* v3.5.3

Uma das bibliotecas mais robustas para a realização de *plots* em Python, capaz de criar vários tipos diferentes de gráficos (MATPLOTLIB, 2023). Ademais, é possível

[1] O quartil é uma forma de dividir um conjunto de dados em quatro partes iguais, representando então, uma divisão em partes de 25%, 50% 75% e 100% dos dados.

modificar o gráfico de modo a torná-lo mais compreensível ao usuário, por meio de funções como *title*, *labels*, *ticks*, *legend*, *grid*, *axes*, *lims*, entre outros.

i. *Title*, *labels* e *ticks*

Essas são funções bastante utilizadas para a etiquetação dos gráficos. Em verdade, a função *title* configura o título do gráfico, podendo ser definido o local do título, a fonte, e até mesmo a cor (MATPLOTLIB, 2023).

As funções de etiqueta são aplicáveis ao eixo das abcissas e das ordenadas, possuindo mesmas parametrizações, variando apenas sua sintaxe, sendo *xlabel* e *ylabel*, respectivamente, para renomear os eixos. Somado aos *ticks*, é possível controlar a forma como as etiquetas aparecem no gráfico, *e.g.*controlar a rotação das etiquetas e/ou o espaçamento entre elas. (MATPLOTLIB, 2023).

ii. *Legend*, *grid*, *axes* e *lims*

Para um maior aprimoramento do gráfico, utilizou-se a função *legend* para inserir legendas sobre os dados, particularmente quando o *plot* possui mais de um dado simultaneamente; a função *grid* insere uma grade quadriculada sobre o gráfico, podendo ser configurada com cor, estilo de linha e espessura. Somado a isso é possível realizar um *subplot* com uso da classe *axes*, aperfeiçoando o gráfico final com mais informações contextuais. Por fim, usa-se os *lims* que delimitam o gráfico, tanto nas abcissas quanto nas ordenadas (se desejável), para garantir que o intervalo mostrado seja o mais conveniente (MATPLOTLIB, 2023).

2.2.4 *NUMPY* V1.21.5 E *SEABORN* V0.12.2

Numpy é uma biblioteca em *Python* que provê objetos em vetores multidimensionais, possibilitando sua aplicação em objetos derivados como matrizes ou vetores mascarados. Além disso, é capaz de realizar com rapidez operações matemáticas, lógicas, manipulação de formas, classificação, transformada de Fourier discreta, análise estatística entre outras (NUMPY DEVELOPERS, 2023). Com isso, utilizou-se a biblioteca *Numpy* para a manipulação dos dados para análises estatísticas e modelagem de objetos, visando aumentar a eficiência do algoritmo ao lidar com operações complexas.

A respeito da *Seaborn*, esta é uma biblioteca de visualização de dados baseada na *Matplotlib*, que fornece interfaces de alto nível para o desenvolvimento de gráficos estatísticos com visuais atrativos e informativos (WASKOM, 2022). Dessa forma, o uso da *Seaborn* é essencial durante a análise exploratória dos dados, especialmente com funções como a *pairplot*, que possibilita a visualização das relações entre as variáveis em um *Dataframe*, por meio de uma matriz de gráficos. Além disso, é possível aprimorar o *pairplot* com uso de parâmetros que enfatizem uma análise de estimativa de densidade de probabilidade das variáveis, como o *Kernel Density Estimation* (KDE), que contribui para uma melhor compreensão dos dados através de identificação de padrões, correlações e tendências.

2.2.5 TensorFlow v2.10.0

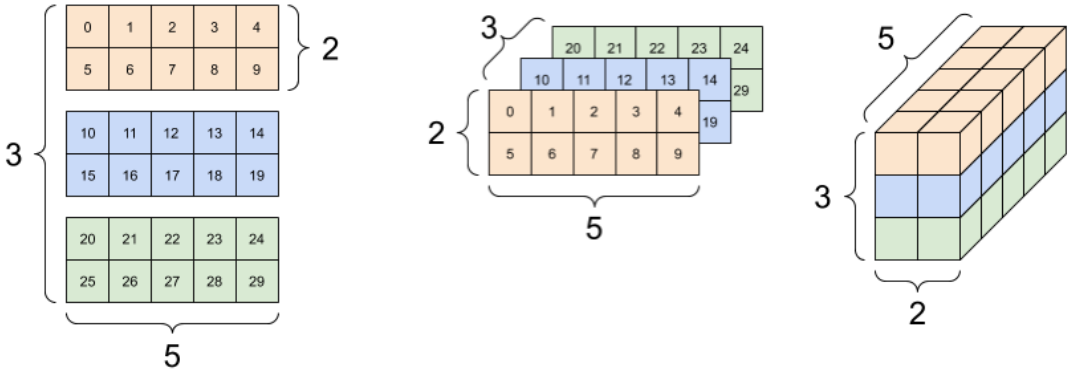
Oficialmente lançado como biblioteca de código aberto em 2015, o *TensorFlow* é uma ferramenta para desenvolvimento de algoritmos de Redes Neurais e *Deep Learning* desenvolvido pelo *Google*®. Por ser uma ferramenta bastante eficiente e acessível, o *TensorFlow* se tornou numerosamente utilizado tanto pelos desenvolvedores individuais como por empresas como Coca-cola®, Intel®, Airbus®, General Eletric®, Twitter® e outras (DIDÁTICA TECH, 2022).

Neste trabalho, utilizou-se funções do *TensorFlow* visando a criação de uma Rede Neural de Aprendizagem Profunda. Para isso, foram empregadas classes como *keras.Sequential* e *keras.layers.Normalization*, as quais possuem subfunções essenciais para a modelagem do algoritmo.

i. *keras.Sequential*

Essa classe permite criar algoritmos de ML camada por camada para o desenvolvimento de um modelo, onde cada camada possui exatamente um tensor de entrada e outro de saída. Um tensor é um *array* multidimensional com um tipo uniforme. Tensores são imutáveis por natureza, podendo apenas ser criados ou destruídos. É possível criar um modelo *Sequential* argumentando uma lista de camadas para o construtor (GOOGLE DEVELOPERS, 2023). A Figura 4 mostra diferentes formas de visualizar um Tensor de 3 eixos.

Figura 4 – Tensor de 3 eixos



Fonte: (GOOGLE DEVELOPERS, 2023)

Para a criação e configuração das camadas, utilizou-se neste trabalho a Unidade de Retificação Linear (ReLU, *Rectified Linear Unit*) como função de ativação. Apesar de seu nome, esta é uma função de ativação não-linear multicamadas, que possui como vantagens um menor custo computacional do treinamento, juntamente com não apresentar problemas de gradiente de convergência, permitindo que um maior número de camadas seja aplicado no treinamento (DEEPAI). Além disso, usou-se camadas densamente conectadas para emprego do método de aprendizagem profunda. Neste método, cada neurônio conecta-se com todos os outros neurônios das camadas anterior e posterior a ele, recebendo entradas e emitindo saídas (pesos), respectivamente.

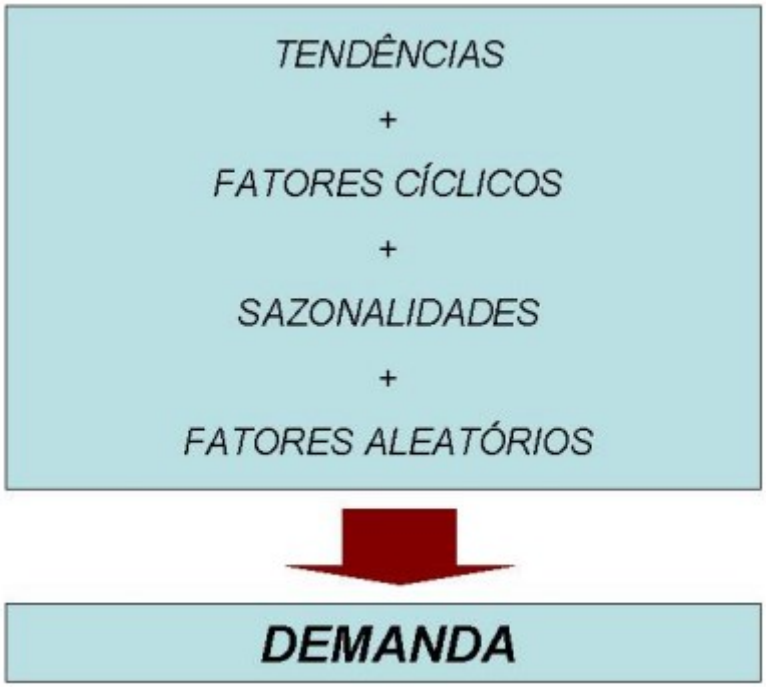
ii. *keras.layers.Normalization*

O estágio de pré-processamento dos dados é essencial para que o algoritmo seja desenvolvido com o mínimo esforço computacional, concomitantemente com o máximo desempenho. Nesse contexto, a função *keras.layers.Normalization* realiza a normalização das características (*features*) contínuas (GOOGLE DEVELOPERS, 2023); em outras palavras, os dados são escalados num intervalo de distribuição padronizado. A função *adapt* torna-se essencial para ajustar o conjunto de dados para a normalização, calculando os valores médios e de variância do *dataset*, permitindo assim, que os dados de entrada sejam normalizados em tempo de execução.

3 METODOLOGIA

Entender quais fatores podem influenciar na demanda de energia elétrica no Brasil é fundamental para que se possa obter dados que possuam correlação com seu comportamento. A Figura 5 mostra vetores que podem ser considerados nesse estudo.

Figura 5 – Composição de vetores que influenciam a demanda



Fonte: (FERREIRA, 2006)

Pode-se entender como tendência a direção de mudança gradual que ocorre em uma série temporal devido a fatores externos ou internos. Por exemplo, devido a popularização de dispositivos inteligentes, e a modernização das residências frente ao uso crescente de dispositivos eletrônicos, o consumo médio energético residencial tende a crescer, implicando então na necessidade de um maior planejamento e de mais infraestrutura para atender centros urbanos.

Os fatores cíclicos são comportamentos estocásticos que se se repetem ao longo de um período (FERREIRA, 2006), não necessariamente sendo esses períodos curtos. Fatores econômicos, por exemplo, como um elevado crescimento ou estagnação industrial, influenciam diretamente na demanda por energia elétrica e não necessariamente se alteram anualmente.

Fatores sazonais são um conjunto de fatores cíclicos que acontecem em períodos curtos, como intervalos de 1 ano. Em outras palavras, são fatores que causam flutuações previsíveis e regulares num sistema, como o frio do inverno que resulta em maior uso de chuveiros elétricos e/ou aquecedores, ou o calor do verão que implica em maior uso de aparelhos de ar-condicionado.

Por outro lado, os fatores aleatórios exercem influência imprevisível e incontrolável sobre um sistema elétrico. Dessa forma, sua influência só pode ser atenuada por estudos estocásticos e simulações por Análise de Contingências.

Assim, para a realização do trabalho, fez-se necessário a pesquisa dos *datasets* em fontes estratégicas, como ONS, Agência Nacional de Energia Elétrica (ANEEL), Empresa de Pesquisa Energética (EPE), Instituto Nacional de Meteorologia (INMET), CCEE, e *Yahoo! Finance*.

Essas fontes foram escolhidas por serem reconhecidas e confiáveis. No entanto, é importante ressaltar que nem sempre foi possível encontrar *datasets* que pudessem ser utilizados neste trabalho, pois nem todas essas fontes disponibilizavam seus dados ao público ou porque não era possível realizar o tratamento a tempo hábil desses, como no caso do INMET, que disponibiliza um massivo conjunto de dados meteorológicos que podem ser bastante úteis (INMET, 2023), mas que necessitam de bastante tratamento, o que escaparia do escopo deste trabalho. Apesar disso, os dados obtidos mostraram-se suficientes para os objetivos deste estudo.

3.1 DESCRIÇÃO E OBTENÇÃO DOS DADOS UTILIZADOS

Para a obtenção dos dados necessários, considerar-se-á os resultados semanais dos anos de 2020 a 2022 como período para treinamento, e o período de janeiro a março de 2023 como dados para teste do modelo. Esses intervalos foram selecionados de forma arbitrária, atentando-se ao fato de não selecionar conjuntos com dados insuficientes para o treinamento. Com base no que já foi discutido em (3), utilizou-se *datasets* com os valores de preços por megawatt hora (R$/MWh) obtido junto a CCEE (CCEE, 2023), a pontuação do índice IBOVESPA para simular o comportamento econômico geral do Brasil obtida junto ao *YAHOO! Finance* (YAHOO, 2023) e a demanda efetiva em megawatt (MW) do SIN, obtida via o ONS (ONS, 2023).

3.2 SELEÇÃO DOS HIPERPARÂMETROS DO MODELO

Os hiperparâmetros são características do algoritmo determinados pelo programador. Essas características dizem respeito a complexidade do modelo, como número de camadas, tipo de camada, taxa de aprendizagem, função de ativação, otimizador e épocas de treinamento. Existem alguns métodos para a configuração desses parâmetros (REIS, 2018), sendo neste trabalho utilizado o método manual.

Os parâmetros foram selecionados de modo a prevenir que ocorra *overfit* e *underfit*, que são fenômenos decorrentes do algoritmo saturar e ter seu coeficiente de aprendizado com o *dataset* comprometido, ou não conseguir aprender suficientemente de modo a realizar previsões com acurácia aceitável (GOOGLE DEVELOPERS, 2023). Além disso, o algoritmo pode consumir mais poder de processamento do que o necessário, tornando-se ineficiente do ponto de vista de consumo computacional, caso os parâmetros superestimem a complexidade do problema.

3.3 ESPECIFICAÇÕES DO *HARDWARE* E *SOFTWARE* UTILIZADOS

Outro fator importante para a execução do trabalho é o ambiente em que será empregado o algoritmo. Isso porque o treinamento exige um considerável poder de processamento do computador; além disso, incompatibilidade de *software*/*hardware* pode impedir que as bibliotecas sejam sequer executadas. A Tabela 1 apresenta especificações de *hardware* relevantes do ambiente utilizado.

Tabela 1 - Especificações de *hardware* utilizado

Memória RAM 24GB
Processador INTEL® Core i5 11400H @2.70GHz
Placa de vídeo dedicada NVIDIA GeForce RTX 3050

Fonte: Autoria própria, 2023

Com isso, é importante destacar que, caso não haja incompatibilidade de *software*, é possível executar o algoritmo em ambientes menos robustos, se atentando apenas ao fato de que o treinamento pode levar um tempo considerável até finalizar. Agora,

apresentar-se-á as especificações de *software* do ambiente de treinamento através da Tabela 2.

Tabela 2 – Especificações de *software* utilizado

Descrição	**Versão**
Sistema Operacional	Ubuntu 22.04.2 LTS
Ambiente de desenvolvimento	Spyder IDE v5.3.3
Interface Gráfica de Usuário	Anaconda Navigator v23.5.0
Linguagem de Programação	Python v3.7.16

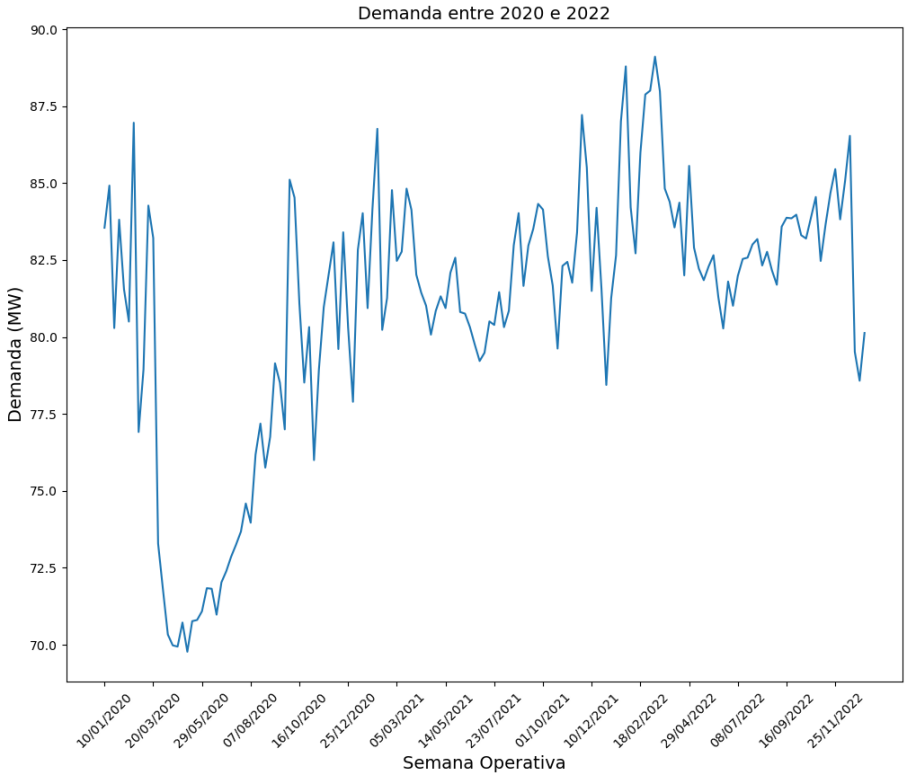
Fonte: Autoria própria, 2023

As versões das bibliotecas utilizadas foram destacadas em (2.2), juntamente com indagações sobre elas.

4 ANÁLISE DOS RESULTADOS

Destarte, apresentar-se-á os *datasets* utilizados em forma de gráficos e tabelas, e então serão feitas observações sobre esses dados. Assim, a Figura 6 exibe o comportamento da demanda utilizada para treino do algoritmo. Nota-se que sem uso de ferramentas de análise gráfica adequadas, a visualização de padrões e tendências torna-se dificultosa, permitindo-se apenas que seja observada uma tendência de convergência dos valores para 82,5MW.

Figura 6 – Demanda do conjunto de treino



Fonte: Autoria Própria, 2023

A Tabela 3 mostra um intervalo do estado do conjunto de dados antes do tratamento para o treinamento do algoritmo que, em sua totalidade, representa um conjunto de dados com dimensão de 159x162, o que explica porque o conjunto não pôde ser inserido aqui de maneira mais abrangente. Os dados descritos com o caractere cerquilha (#) podem ser considerados como ruído. Esse conjunto ajuda a enfatizar a importância de um tratamento de dados eficaz, pois muitas das entradas que se encontram

nesse *dataset* não são convenientes para o treinamento da RNA. Por ser um *dataset* em estado bruto, ainda não é possível perceber informações relevantes que possam ser extraídas para análise. Para uma melhor visualização dos dados, é recomendado carregar essa base de dados brutos no *Spyder*, e a partir daí começar a realizar manipulações e chegar a conclusões.

Tabela 3 – Amostra dos dados de treinamento da Demanda sem tratamento

					Data Escala de Tempo 1 DM Simp 4	Data Escala de Tempo 1 DM Simp 4
Subsistema	Data do Início da Semana Din Instante DM Simp 4	Texto Data Início da Semana Din Instante DM	Período Exibido DM Simp 4	Din Instante	4 de jan de 20	11 de jan de 20
SIN	########	a	Semana Operativa	##########	83.549	
SIN	########	a	Semana Operativa	##########		84.919

Fonte: (ONS, 2023) (Adaptado)

Na Tabela 4, tem-se um intervalo deste mesmo conjunto após sua filtragem, sendo o mesmo utilizado na confecção da Figura 6. Destaca-se que a dimensão desse conjunto se desvalorizou para 157x2, demonstrando que o conjunto original possuía um elevado número de elementos nulos para os objetivos deste trabalho, restando apenas a data de geração dos dados e o valor da demanda.

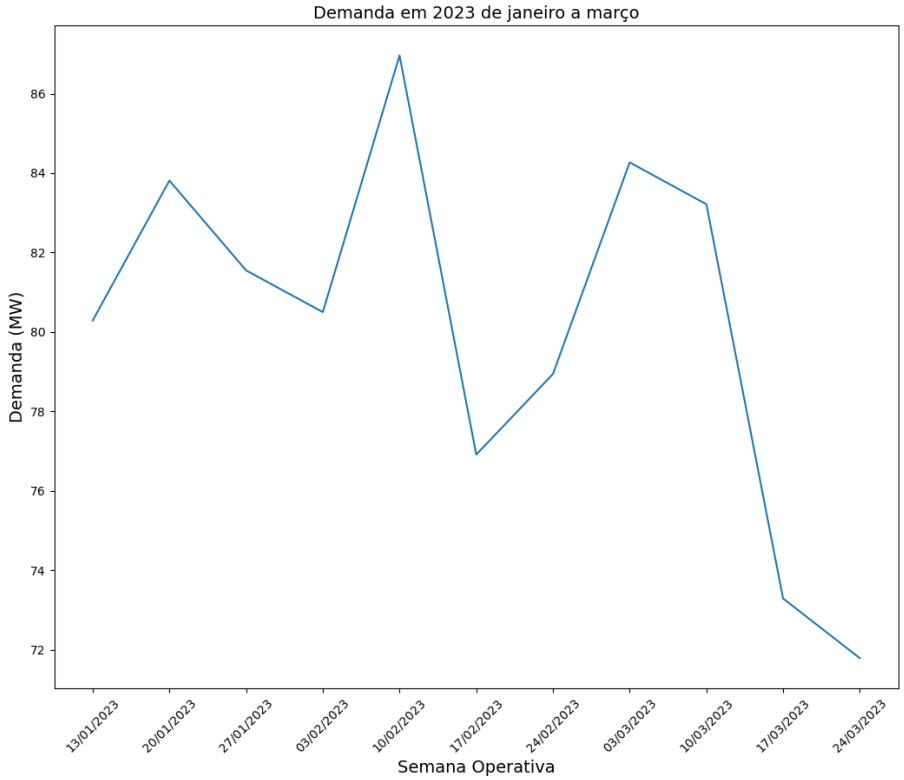
Tabela 4 – Amostra dos dados de treinamento da Demanda tratados

Data	Demanda (MW)
10/01/2020	83549
17/01/2020	84919
24/01/2020	80029
31/01/2020	83811

Fonte: Autoria própria, 2023 (Adaptado)

A Figura 7 ilustra o comportamento do conjunto de testes do algoritmo. Seu tratamento e obtenção deram-se de forma análoga ao dos dados demonstrados na Tabela 3 e Tabela 4.

Figura 7 – Demanda do conjunto de teste



Fonte: Autoria própria, 2023

No caso da Figura 7, é possível notar um comportamento que indica uma tendência de queda da demanda elétrica em 2023, pois tanto os topos quanto os fundos do gráfico estão decrescendo com o evoluir do tempo.

A Tabela 5, analogamente a Tabela 3, mostra um intervalo amostral do conjunto de dados de treinamento referente ao Custo Médio Nacional de Energia Elétrica (CMNEE), em R$/MWh. O *dataset* se denomina assim porque determinou-se que seria utilizado o custo médio ao invés do custo por região. No entanto, não há nada que impeça que o algoritmo seja modificado de modo a aprender a prever este custo, bastando para isso apenas mudar a abordagem de modelagem dos *datasets* para refazer o treinamento.

Tabela 5 – Amostra do conjunto de dados de treinamento do CMNEE

DATA_FIM	SUDESTE	SUL	NORDESTE	NORTE
03/01/2020	290,33	290,33	290,33	290,33
10/01/2020	368,51	368,51	368,51	368,51
17/01/2020	268,32	268,32	268,32	268,32

Fonte: (CCEE, 2023) (Adaptado)

Apesar do que é apontado na Tabela 5, nem sempre os custos são idênticos em todas as regiões no mesmo intervalo de tempo, por isso fez-se necessário extrair a média aritmética desses valores para a obtenção do custo médio. A Tabela 6 mostra um intervalo deste *dataset* tratado. Convém mencionar que o conjunto completo possui dimensão de 163x2.

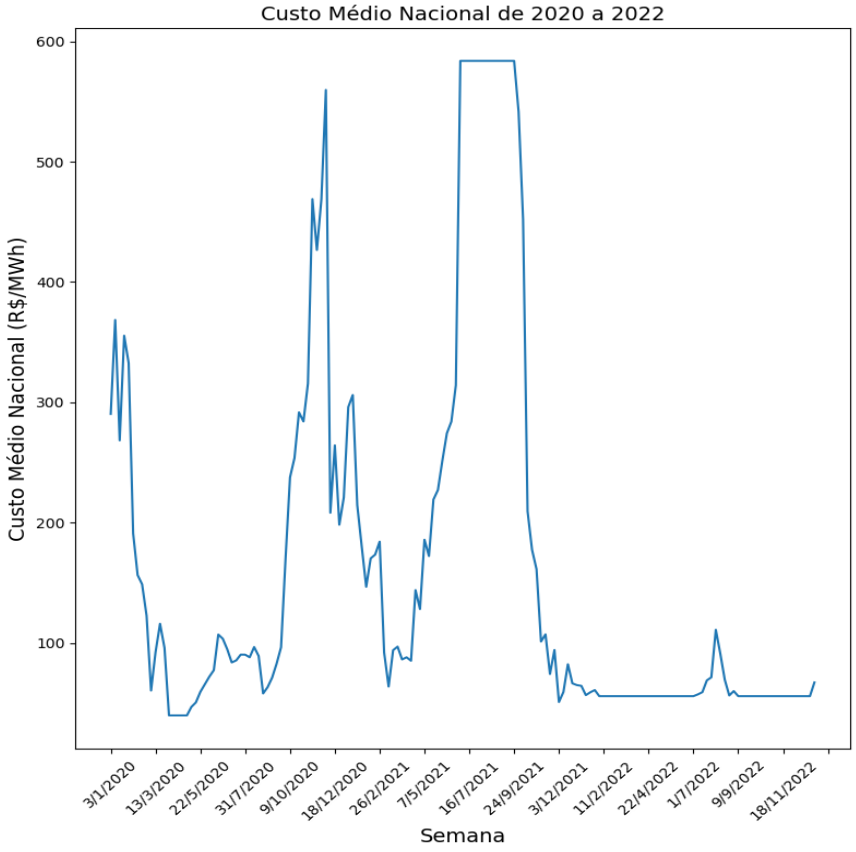
Tabela 6 – Amostra do conjunto de dados de treinamento do CMNEE tratado

Data	CMN (R$/MWh)
03/01/2020	290.33
10/01/2020	368.51
17/01/2020	268.32
24/01/2020	355.42

Fonte: Autoria própria, 2023 (Adaptado)

Com o conjunto originador da Tabela 6, foi construída a Figura 8.

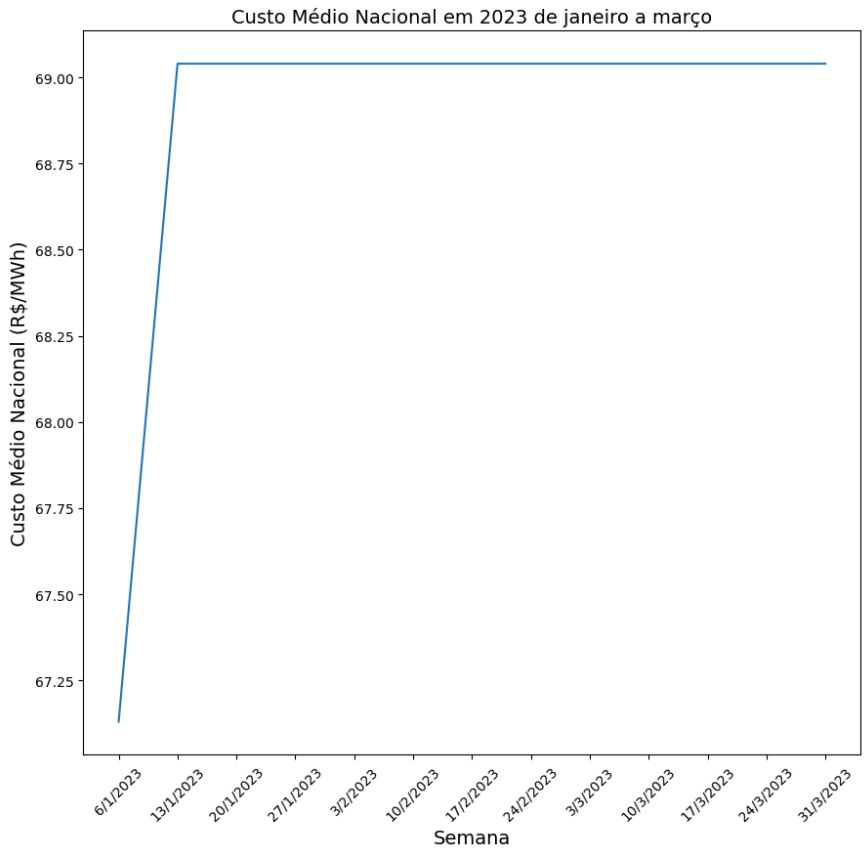
Figura 8 – CMNEE do conjunto de treino



Fonte: Autoria Própria, 2023

A Figura 9 mostra o comportamento do CMNEE do conjunto de testes. Seu tratamento deu-se de forma análoga ao do conjunto de treino, mostrado na Tabela 5 e na Tabela 6.

Figura 9 – CMNEE do conjunto de teste



Fonte: Autoria própria, 2023

A Tabela 7 apresenta uma amostra dos dados de treinamento das pontuações do Índice BOVESPA (IBOV). Para o objetivo do trabalho, foi escolhido arbitrariamente a pontuação de fechamento (*Close*) para treino e teste. Destaca-se que é possível utilizar mais parâmetros deste *dataset* no treino, necessitando apenas de alterações na modelagem dos dados.

Tabela 7 – Amostra das pontuações de treino do IBOV

Date	Open	High	Low	Close	Adj Close	Volume
2020-01-01	115652.0	118792.0	115649.0	116662.0	116662.0	23421300
2020-01-08	116667.0	117705.0	114952.0	117632.0	117632.0	27694200
2020-01-15	117632.0	118862.0	115961.0	117026.0	117026.0	24868400

Fonte: (YAHOO, 2023) (Adaptado)

A Tabela 8 apresenta uma amostra desse conjunto após a filtragem. Sua dimensão alterou de 157x7 para 157x2.

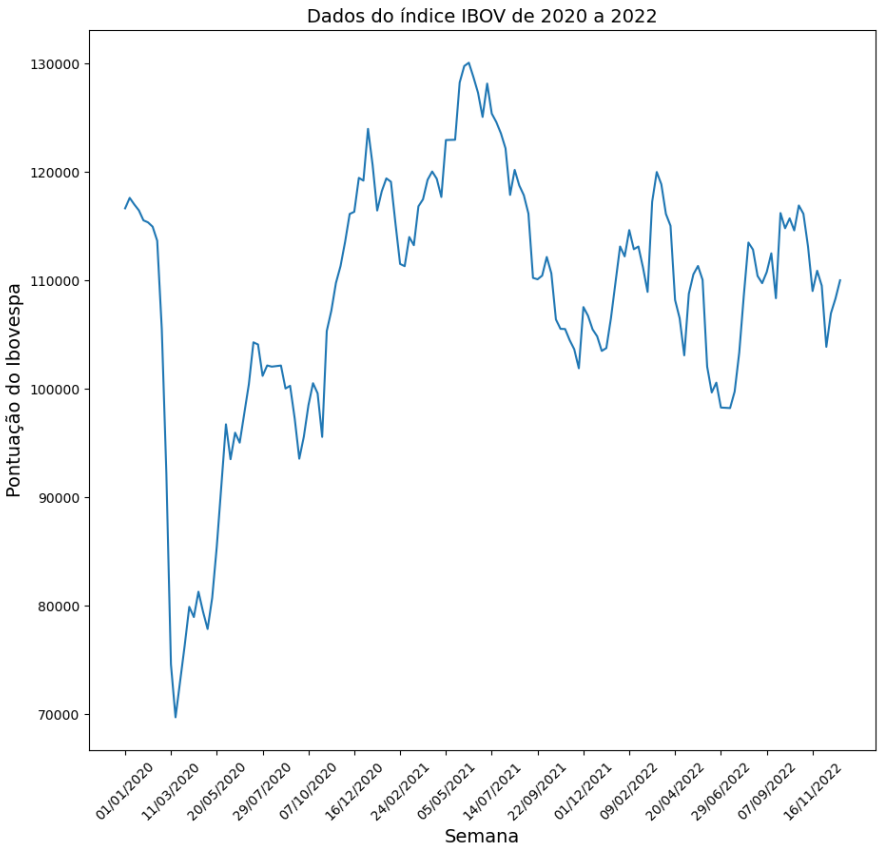
Tabela 8 – Amostra das pontuações de treino do IBOV tratadas

Data	Fechamento
01/01/2020	116662.0
08/01/2020	117632.0
15/01/2020	117026.0
22/01/2020	116479.0

Fonte: Autoria própria, 2023 (Adaptado)

Com o conjunto completo dos dados apresentados na Tabela 8, a Figura 10 foi confeccionada. É possível notar que esse conjunto apresenta tendência para convergir para 110000 pontos conforme o tempo cresce, dado que as cristas e vales diminuem suas amplitudes em torno dessa pontuação.

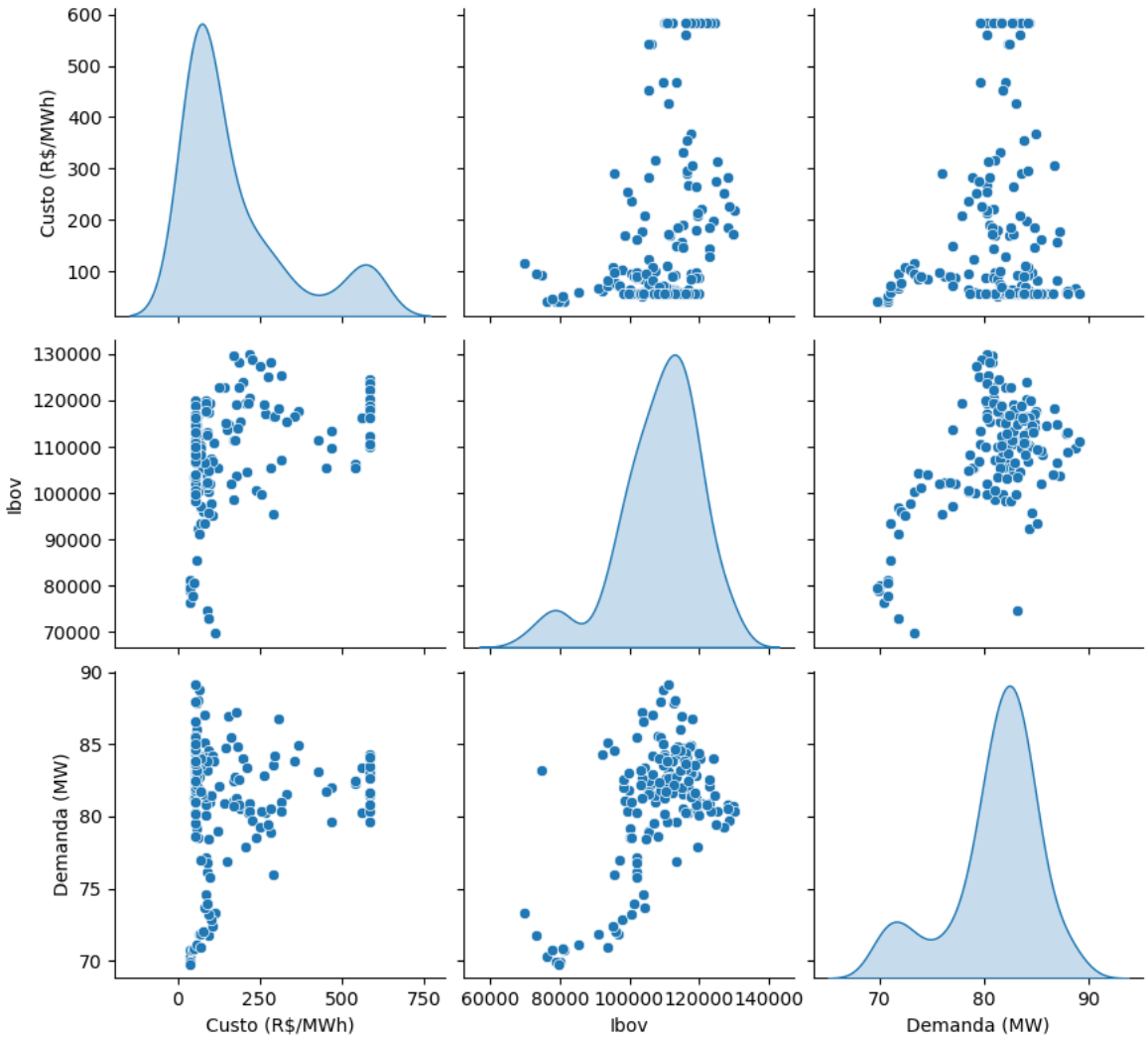
Figura 10 – Conjunto de treinamento do IBOV



Fonte: Autoria própria, 2023

Por fim, a Figura 11 apresenta um *pairplot* (seção 2.2.4), para que seja possível realizar observações acerca de como os dados de treino se comportam um com relação ao outro. A simetria diagonal dos gráficos permite que mesmas plotagens possam ser visualizadas sob ópticas diferentes.

Figura 11 – *Pairplot* dos conjuntos de treino



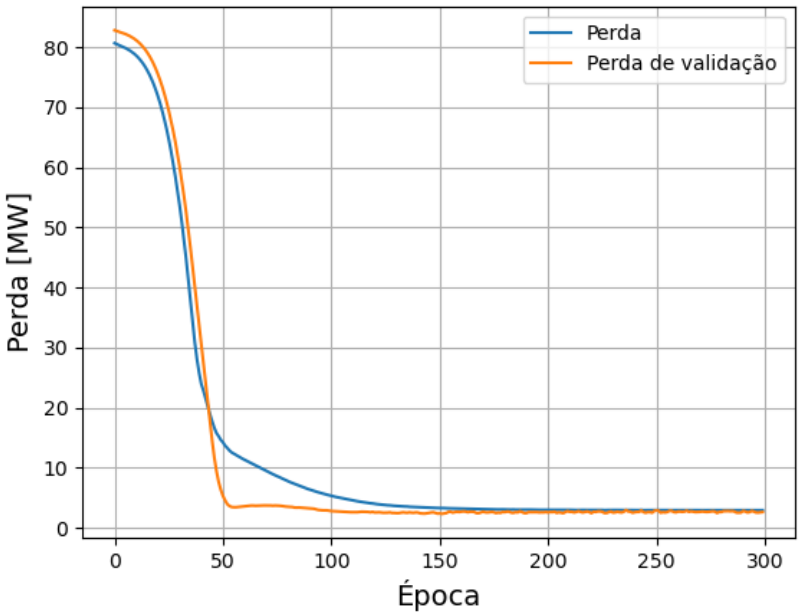
Fonte: Autoria própria, 2023

Percebe-se, pela Figura 11, que os dados possuem correlação entre si devido a concentração de pontos em regiões específicas. Assim, por exemplo, observa-se que a demanda e o IBOV possuem comportamentos similares conforme o custo varia. Além disso, é possível observar um adensamento de pontos entre o IBOV e a demanda, indicando alinhamento entre tendências dessas variáveis.

4.1 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS OBTIDOS

A Figura 12 demonstra como a taxa de aprendizagem do algoritmo evoluiu conforme ele se adaptou ao treinamento em cada época[2].

Figura 12 – Evolução do algoritmo



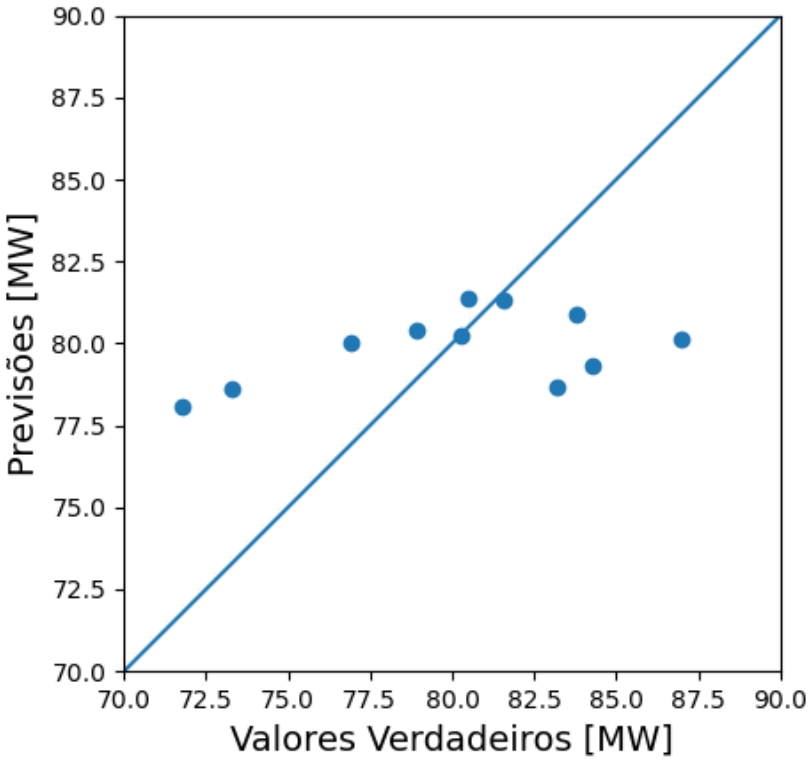
Fonte: Autoria própria, 2023

Com a Figura 12, é possível visualizar que os valores de perda, que representam a discrepância entre os dados de treinamento e os dados de teste[3], é continuamente decrescente até o algoritmo atingir a melhor taxa de aprendizado. Além disso, a perda por validação, que traduz as perdas dentre subintervalos não vistos durante o treinamento[4], também apresenta o mesmo comportamento. Isso demonstra que os fenômenos de *underfit* ou *overfit* descritos em (3.2) não ocorreram, assegurando então, a eficiência do algoritmo.

[2] Época se refere a cada sessão de treinamento do algoritmo, onde a rede neural recebe uma amostra do *dataset* e aprende seu comportamento para inferir um resultado
[3] este conjunto, neste caso, é um subconjunto dos dados de treinamento utilizados no processo de aprendizagem, não o conjunto dos dados mostrados na Figura 7.
[4] o conjunto de treinamento ainda se divide em um terceiro subconjunto para que o algoritmo valide suas previsões e faça uma aferição de como sua taxa de aprendizado deve convergir.

Com o treinamento concluído, obteve-se os resultados reproduzidos na Figura 13. Nela, os valores verdadeiros são retratados em função dos valores previstos, com uma reta de simetria para auxiliar a avaliar a magnitude dos erros dos resultados obtidos quando o algoritmo tentou prever os dados reproduzidos na Figura 7.

Figura 13 – Resultados do treinamento



Fonte: Autoria própria, 2023

Nota-se que o algoritmo converge para os resultados reais, de acordo com o esperado. Apesar do aparente espalhamento dos valores, deve-se observar que o intervalo de valores mostrado na figura é relativamente pequeno. Para uma análise mais detalhada dos erros, a Figura 14 exibe um histográfico dos erros, juntamente com a frequência com que cada um ocorre, enquanto a Tabela 9 exibe os dados previstos e verdadeiros usados na confecção da Figura 13.

Tabela 9 – Resultados em MW

Valores Previstos	Valores Verdadeiros
80.24	80.29
80.87	83.81
81.31	81.55
81.37	80.50
80.12	86.96
80.01	76.91
80.37	78.94
79.30	84.26
78.68	83.21
78.60	73.29
78.09	71.78

Fonte: Autoria própria, 2023

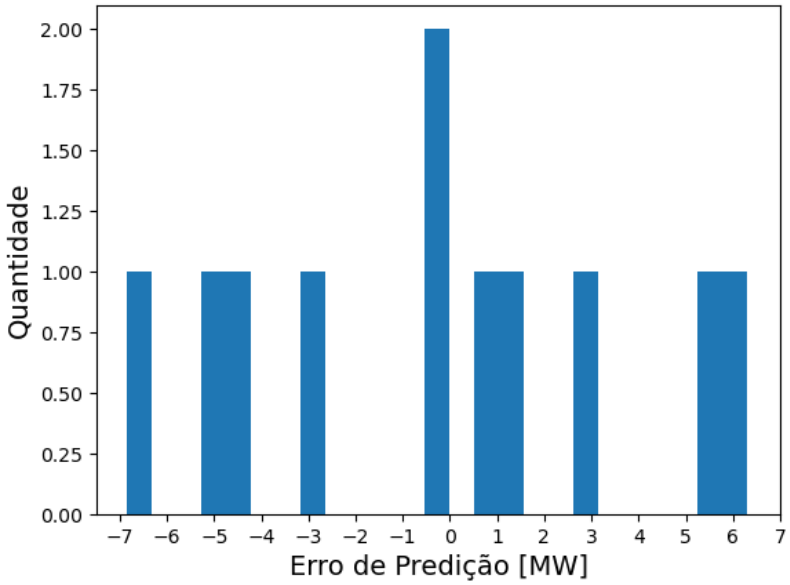
Os Valores Verdadeiros na Tabela 9 são o conjunto completo dos dados que originaram a Figura 7. Os dados da Tabela 9 originaram a Tabela 10, que exibe o valor de cada erro individual. Convém mencionar que o algoritmo apresentou erro médio absoluto de 3.3257MW, algo que pode ser considerado satisfatório, pois é um valor relativamente pequeno com relação a uma média de demanda de aproximadamente 80MW.

Tabela 10 – Erros de previsão em MW

Erro (MW)
-0.0493
-2.9317
-0.2393
0.8722
-6.8374
3.1036
1.4337
-4.9606
-4.5328
5.3149
6.3073

Fonte: Autoria própria, 2023

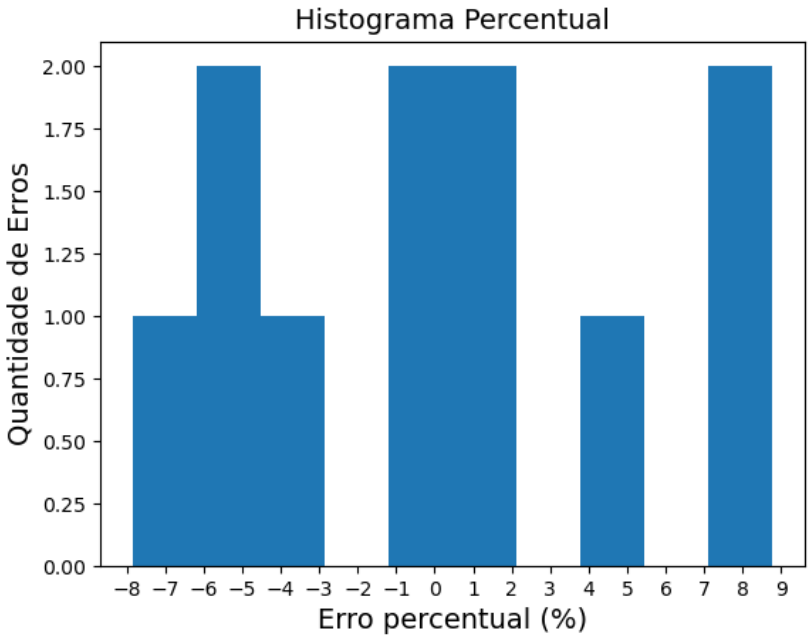
Figura 14 – Histograma da quantidade de erros



Fonte: Autoria própria, 2023

Pela Figura 14, percebe-se que a maioria dos erros do algoritmo foram sobrestimados, ou seja, a previsão da RNA previu valores maiores do que os valores reais, sendo isso observável também na Figura 13 e Tabela 10. Além disso, nenhum erro ultrapassou o valor de 7MW, indicando que houve uma alta taxa de precisão sob a óptica percentual. Com efeito, a Figura 15 exibe este fato de forma melhor observável.

Figura 15 – Histograma dos erros percentuais



Fonte: Autoria própria, 2023

Os erros percentuais exibidos na Figura 15 foram calculados usando uma métrica que se baseia em valor por valor, não com relação a um valor fixo. Em outas palavras, cada porcentagem representa o erro de cada valor previsto com relação a seu par verdadeiro, conforme a Tabela 11. Assim, percebe-se que minoria dos valores previstos pelo algoritmo seriam danosos financeiramente conforme citado em (1), dado que apenas esses elementos ultrapassaram um erro superior a +5%. Nesse contexto, as previsões sobrestimadas do algoritmo podem servir como uma vantagem estratégica, dado que esse excesso de energia pode ser negociado com outros atores, possibilitando maiores lucros operacionais.

Tabela 11 – Erro percentual de previsão

Erro (%)
-0.0614
-3.4981
-0.2935
1.0835
-7.8626
4.0351
1.8161
-5.8867
-5.447
7.2518
8.7858

Fonte: Autoria própria, 2023

A Tabela 12 apresenta dados estatísticos dos erros percentuais registrados na Tabela 11. Com isso, percebe-se que pelo menos 75% dos erros são menores que 3%, provando a eficiência do algoritmo. O erro apresenta um desvio padrão relativamente elevado por causa da variação dos dados em si, não por questão de ineficiência. Por fim, a média dos erros apresenta valor quase nulo por se tratar de uma média aritmética, não absoluta, e isso indica que o algoritmo converge para erro nulo quando for prever uma quantidade elevada de dados.

Tabela 12 – Estatísticas dos erros percentuais de previsão

	Erro (%)
Quantidade de valores	11
Média	-0.0069
Desvio Padrão	5.3759
Mínimo	-7.8626
25%	-4.4725
50%	-0.0614
75%	2.9256
Máximo	8.7858

Fonte: Autoria própria, 2023

4.2 LIMITAÇÕES E SUGESTÕES PARA TRABALHOS FUTUROS

- Melhorar o método de seleção dos hiperparâmetros
    - Comparar métodos como Busca em Grade (*Grid Search*), Busca Aleatória (*Random Search*), e Otimização Bayesiana (*Bayesian Optimization*) (REIS, 2018)
- Tentar obter uma maior base de dados para treinamento
    - Obter dados climáticos para usar no treinamento; podem ser úteis devido à presença elevada da matriz hídrica no Brasil. Além disso, podem ajudar a entender a demanda por questões como tempo quente ou frio; ensolarado ou não;
        - Tratar os dados do INMET pode ser um desafio, conforme citado em (3).
- Utilizar datas como parte dos dados de treinamento
    - Isso pode contribuir para o aprendizado amplificando a capacidade da RNA de prever fatores periódicos no tempo.
    - Não foi possível usar as datas no treino deste trabalho; essas acabaram sendo utilizadas na confecção das figuras.
        - O não uso das datas no treino é uma limitação do *TensorFlow* que não foi possível contornar.
- Desenvolver um modelo de previsão contínua
    - Evoluir o trabalho de modo a possibilitar o algoritmo a receber dados de forma contínua para obter uma prevenção de resultados em tempo real
- Estipular as previsões para um período maior
    - Com o modelo desenvolvido, aumentar as previsões para intervalos de tempo futuros *e.g.*, até 2025.

5 CONSIDERAÇÕES FINAIS

A inteligência artificial está se mostrando como uma das áreas mais importantes da tecnologia atualmente, e se tornando um importante vetor de desenvolvimento para o futuro próximo, especialmente para a Engenharia Elétrica.

Com efeito, o matemático londrino Clive Humby, especialista em Ciência dos Dados, afirmou:

> Dados são o novo petróleo. É valioso, mas se não tratado não pode ser realmente usado (...) então os dados devem ser tratados, analisados para que possam ter valor. (Traduzido)[5]

Sob essa óptica, reforça-se a importância de estudar, obter e tratar dados para um fim benéfico, como o objetivo principal deste trabalho, que com sucesso desenvolveu uma RNA com alta eficiência de previsão da demanda de energia elétrica no Brasil, apesar dos desafios encontrados com a obtenção e tratamento dos dados. Outro indicativo do potencial do campo de Inteligência Artificial é a valorização trilionária que empresas de tecnologias atingiram somente no primeiro trimestre de 2023 (WANG e POPINA, 2023), que demonstra o quanto essa área irá receber em investimentos no futuro próximo.

[5] *Data is the new oil. It's valuable, but if unrefined it cannot really be used (...) so must data be broken down, analyzed for it to have value. (HUMBY, C.)*

REFERÊNCIAS

CARVALHO DE ALMEIDA, F.; LIMA PASSARI, A. F. Previsão de vendas no varejo por meio de redes neurais. **Revista de Administração - RAUSP**, São Paulo, Brasil, 41, n. 3, Setembro 2006. 17. Disponivel em: <https://www.redalyc.org/pdf/2234/223417413004.pdf>. Acesso em: 12 Maio 2023.

CCEE. CCEE - Painel de Preços. **Painel de Preços**, 2023. Disponivel em: <https://www.ccee.org.br/web/guest/precos/painel-precos>. Acesso em: 7 Abril 2023.

DEEPAI. DeepAI. **Data Science & AI Glossary**. Disponivel em: <https://deepai.org/definitions>. Acesso em: 20 Maio 2023.

DIDÁTICA TECH. Didática Tech. **O que é TensorFlow? Para que serve?**, 2022. Disponivel em: <https://didatica.tech/o-que-e-tensorflow-para-que-serve/>. Acesso em: 20 Maio 2023.

FERREIRA, R. V. **Previsão de demanda: um estudo de caso para o Sistema Interligado Nacional**. UFMG. [S.l.], p. 142. 2006.

GOOGLE DEVELOPER DOCS. Regressão: Preveja o consumo de combustível. **TensorFlow**, 2022. Disponivel em: <https://www.tensorflow.org/tutorials/keras/regression?hl=pt-br>. Acesso em: 3 Maio 2023.

GOOGLE DEVELOPER DOCS. Basic regression: Predict fuel efficiency. **TensorFlow**, 2023. Disponivel em: <https://www.tensorflow.org/tutorials/keras/regression?hl=en>. Acesso em: 13 Maio 2023.

GOOGLE DEVELOPERS. TensorFlow. **TensorFlow v2.12.0**, 2023. Disponivel em: <https://www.tensorflow.org/api_docs/python/tf>. Acesso em: 20 Maio 2023.

GRUS, J. **Data Science do Zero**. 1. ed. Rio de Janeiro: Alta Books, v. 1, 2016.

INMET. Instituto Nacional de Meteorologia. **Dados Históricos Anuais**, 2023. Disponivel em: <https://portal.inmet.gov.br/dadoshistoricos>. Acesso em: 29 Março 2023.

KOVÁCS, Z. L. **Redes neurais artificiais:** fundamentos e aplicações. 4. ed. São Paulo: Livraria da Física, v. 1, 2006.

LE WAGON. Cientista de dados: a profissão que mais cresce no Brasil. **Le Wagon**, 2023. Disponivel em: <https://blog.lewagon.com/pt-br/news/cientista-de-dados-a-profissao-que-mais-cresce-no-brasil/>. Acesso em: 13 Maio 2023.

MATPLOTLIB. matplotlib. **matplotlib Users guide**, 2023. Disponivel em: <https://matplotlib.org/stable/users/index>. Acesso em: 17 Maio 2023.

NUMFOCUS. Pandas. **Pandas User Guide**, 2023. Disponivel em: <https://pandas.pydata.org/docs/user_guide/index.html>. Acesso em: 13 Maio 2023.

NUMPY DEVELOPERS. Numpy. **NumPy documentation v1.24**, 2023. Disponivel em: <https://numpy.org/doc/1.24/index.html>. Acesso em: 17 Maio 2023.

ONS. Plano da Operação Elétrica de Médio Prazo do SIN, p. 69, 2021. Disponivel em: <https://www.ons.org.br/AcervoDigitalDocumentosEPublicacoes/Sumario%20Executivo_PARPEL_2021.pdf>. Acesso em: 12 Maio 2023.

ONS. Geração de Energia. **ONS**, 12 Maio 2023. Disponivel em: <https://tableau.ons.org.br/t/ONS_Publico/views/GeraodeEnergia/HistricoGeraodeEnergia?%3Aembed=y&%3Adisplay_count=n&%3AshowAppBanner=false&%3AshowVizHome=n&%3Aorigin=viz_share_link>. Acesso em: 12 Maio 2023.

ONS. Resultados da Operação. **Demanda Máxima**, 2023. Disponivel em: <https://www.ons.org.br/Paginas/resultados-da-operacao/historico-da-operacao/demanda_maxima.aspx>. Acesso em: 29 Março 2023.

PYTHON SOFTWARE FOUNDATION. The Python Standard Library. **Python**, 2023. Disponivel em: <https://docs.python.org/3/library/>. Acesso em: 13 Maio 2023.

REIS, C. H. **Otimização de Hiperparâmetros em Redes Neurais Profundas**. Universidade Federal de Itajubá - UNIFEI. Itajubá, p. 72. 2018.

ROBINSON, M. Why Neural Networks can learn (almost) anything. **YouTube**, 2022. Disponivel em: <https://www.youtube.com/watch?v=0QczhVg5HaI>. Acesso em: 21 Maio 2023.

RUAS, G. I. S. et al. **Previsão de Demanda de Energia Elétrica Utilizando Redes Neurais Artificiais e Support Vector Regression**. Departamento de Engenharia Elétrica – Universidade Federal do Paraná (UFPR). Curitiba – PR – Brazil, p. 10.

WANG, L.; POPINA, E. Bloomberg Línea. **Bloomberg Línea**, 2023. Disponivel em: <https://www.bloomberglinea.com.br/2023/05/31/techs-sobem-44-no-ano-com-febre-da-ia-e-levam-mercado-a-repensar-estrategia/>. Acesso em: 03 Junho 2023.

WASKOM, M. seaborn. **seaborn:** statistical data visualization, 2022. Disponivel em: <https://seaborn.pydata.org/>. Acesso em: 20 Maio 2023.

YAHOO. IBOVESPA(^BVSP). **Yahoo! Finance**, 2023. Disponivel em: <https://finance.yahoo.com/quote/%5EBVSP/history?period1=1672531200&period2=1680220800&interval=1wk&filter=history&frequency=1wk&includeAdjustedClose=true>. Acesso em: 7 Abril 2023.

GLOSSÁRIO

Features	Propriedade mensurável ou característica de um *dataset*
Labels	Saída ou alvo que um modelo está sendo treinado para prever
Quartil	Valor divisor de um *dataset* em 25% de seu tamanho
array	Estrutura de dados com uma coletânea de elementos
Tensor	Objeto que representa um *array* numérico multidimensional
dataset	Conjunto de dados
Hiperparâmetro	Configuração de comportamento de algoritmo em ML

APÊNDICE A – CÓDIGO DE TRATAMENTO DA DEMANDA

```
# -*- coding: utf-8 -*-
"""
Created on Wed Mar 29 10:52:28 2023

@author: luanfabiomg
"""

#%%
import os
import pandas as pd
import matplotlib.pyplot as plt

#%%
#setando o diretório de trabalho
os.chdir('H:\\data\\demanda')
#Backslash \ is an escape character, so you have to use \\

demanda_treino = pd.read_csv("demanda_2020a2022.csv",
                             sep="\t", encoding= 'utf-16',
                             header=None)

demanda_teste = pd.read_csv("demanda_2023.csv", sep="\t",
                            encoding='utf-16', header=None)

#%%
#eliminando a primeira linha
demanda_treino.drop(0, inplace=True)

#eliminando os valores na das colunas e criando uma nova coluna
demanda_treino["Demanda (MW)"] = ...
demanda_treino.iloc[1:, 5:].apply(lambda x: x.dropna().values,
                                  axis=1)

#eliminando as colunas com valores na
demanda_treino.drop(range(5,162), axis=1, inplace=True)

#%%
#eliminando demais colunas que não serão usadas
demanda_treino.drop(range(2,4), axis=1, inplace=True)
demanda_treino.drop(0, axis=1, inplace=True)
demanda_treino.drop(4, axis=1, inplace=True)
```

```
#eliminando a primeira linha
demanda_treino.drop(1, inplace=True)

#resetando os índices das linhas
demanda_treino.reset_index(drop=True, inplace=True)

#renomeando as colunas
demanda_treino.columns=["Data", "Demanda (MW)"]

#%%
#convertendo os dados para float
demanda_treino['Demanda (MW)'] = ...
demanda_treino['Demanda (MW)'].astype(float)

#convertendo as datas para um formato reconhecido pelo pandas
demanda_treino['Data'] = ...
pd.to_datetime(demanda_treino['Data'],
               format='%d/%m/%Y')

demanda_treino['Data'] = ...
demanda_treino['Data'].dt.strftime('%d/%m/%Y')

#%%
#plot exploratório
plt.figure()
plt.title("Demanda entre 2020 e 2022")
plt.plot(demanda_treino['Data'],
         demanda_treino['Demanda (MW)'])
plt.xlabel('Semana Operativa')
plt.xticks(range(0, len(demanda_treino), 10), rotation=45)
plt.ylabel('Demanda (MW)')

#%%
#realizando os mesmos procedimentos para os dados de teste
demanda_teste.drop(0, inplace=True)
demanda_teste["Demanda (MW)"] = ...
demanda_teste.iloc[1:, 5:].apply(lambda x: x.dropna().values,
                                 axis=1)

demanda_teste.drop(range(4,16), axis=1, inplace=True)
demanda_teste.drop(range(2,4), axis=1, inplace=True)

demanda_teste.drop(0, axis=1, inplace=True)
demanda_teste.drop(1, inplace=True)

demanda_teste['Demanda (MW)'] = ...
demanda_treino['Demanda (MW)'].astype(float)
```

```
demanda_teste.columns=["Data", "Demanda (MW)"]

demanda_teste['Data'] = pd.to_datetime(demanda_teste['Data'],
                                       format='%d/%m/%Y')
demanda_teste['Data'] = ...
demanda_teste['Data'].dt.strftime('%d/%m/%Y')

plt.plot(demanda_teste['Data'], demanda_teste['Demanda (MW)'])
plt.xlabel('Semana Operativa')
plt.xticks(rotation=45)
plt.ylabel('Demanda (MW)')

#%%
#salvando os arquivos tratados
demanda_treino.to_csv('demanda_treino.csv', index=False)
demanda_teste.to_csv('demanda_teste.csv', index=False)
```

APÊNDICE B – CÓDIGO DE TRATAMENTO DOS CUSTOS

```
# -*- coding: utf-8 -*-"""
Created on Wed Apr 12 09:49:26 2023

@author: luang
"""

#%%
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

#%%
#diretório de trabalho
os.chdir('H:\\data\\custos')

#dados sem tratamento
 custo1 = pd.read_csv("1 preco_semanal JAN a JUN 2020.csv",
                      sep=";", encoding='utf-8', decimal=',')

custo2 = pd.read_csv("2 preco_semanal JUN a DEZ 2020.csv",
                     sep=";", encoding='utf-8', decimal=',')

custo3 = pd.read_csv("3 preco_semanal DEZ 2020 a MAI 2021.csv",
                     sep=";",encoding='utf-8', decimal=',')

custo4 = pd.read_csv("4 preco_semanal MAI A OUT 2021.csv",
                     sep=";", encoding='utf-8', decimal=',')

custo5 = pd.read_csv("5 preco_semanal OUT 2021 a ABR 2022.csv",
                     sep=";",encoding='utf-8', decimal=',')

custo6 = pd.read_csv("6 preco_semanal ABR a OUT 2022.csv",
                     sep=";",encoding='utf-8', decimal=',')

custo7 = pd.read_csv("7 preco_semanal OUT a DEZ 2022.csv",
                    sep=";", encoding='utf-8', decimal=',')

custo_teste = pd.read_csv("preco_semanal JAN a MAR 2023.csv",
                          sep=";", encoding='utf-8', decimal=',')
```

```
#%%
#concatenando os dados
custo_treino = pd.concat([custo1, custo2, custo3, custo4,
                          custo5, custo6, custo7],
                         ignore_index=True, levels=None)

#%%
#tomando o custo médio nacional
custo_treino['CMN (R$/MWh)'] = ...
np.mean(custo_treino.loc[:, "SUDESTE":"NORTE"], axis=1)

custo_teste['CMN (R$/MWh)'] = ...
np.mean(custo_teste.loc[:, "SUDESTE":"NORTE"], axis=1)

#%%
#eliminando as colunas desnecessárias
custo_treino.drop(['ANO', 'MES', 'SEMANA', 'DATA_INICIO',
                   'SUDESTE', 'SUL', 'NORDESTE', 'NORTE'],
                   axis=1, inplace=True)

custo_teste.drop(['ANO', 'MES', 'SEMANA', 'DATA_INICIO',
                  'SUDESTE', 'SUL', 'NORDESTE',
                  'NORTE'], axis=1, inplace=True)
#%%
#renomeando as colunas
custo_treino.columns=["Data", "CMN (R$/MWh)"]
custo_teste.columns=["Data", "CMN (R$/MWh)"]

#%%
#plot exploratório
plt.plot(custo_treino['Data'], custo_treino['CMN (R$/MWh)'])
plt.xlabel('Semana')
plt.xticks(range(0, len(custo_treino), 10), rotation=45)
plt.ylabel('Custo Médio Nacional (R$/MWh)')

#%%
#formatando as datas
custo_treino['Data'] = pd.to_datetime(custo_treino['Data'],
                        format='%d/%m/%Y')
custo_treino['Data'] = ...
custo_treino['Data'].dt.strftime('%d/%m/%Y')

custo_teste['Data'] = pd.to_datetime(custo_teste['Data'],
                      format='%d/%m/%Y')

custo_teste['Data'] = ...
custo_teste['Data'].dt.strftime('%d/%m/%Y')
```

```
#%%
#por fim, salvando os dados filtrados
custo_treino.to_csv('custo_treino.csv', index=False)
custo_teste.to_csv('custo_teste.csv', index=False)
```

APÊNDICE C – CÓDIGO DE TRATAMENTO DO ÍNDICE IBOV

```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 12 16:24:46 2023
@author: luang
"""

#%%
import os
import pandas as pd
import matplotlib.pyplot as plt

#%%
#diretório de trabalho
os.chdir('H:\\data\\ibovespa')

#dados sem tratamento
ibov_treino = pd.read_csv("^BVSP 2020 a 2022.csv", sep=",",
                          encoding='utf-8')
ibov_teste = pd.read_csv("^BVSP 2023.csv", sep=",",
                         encoding='utf-8')

#%%
#eliminando as colunas desnecessárias
ibov_treino.drop(['Open', 'High', 'Low', 'Adj Close',
                  'Volume'],axis=1, inplace=True)

ibov_teste.drop(['Open', 'High', 'Low', 'Adj Close', 'Volume'],
                axis=1, inplace=True)

#%%
#renomeando as colunas
ibov_treino.columns=["Data", "Fechamento"]
ibov_teste.columns=["Data", "Fechamento"]

#%%
#formatando as datas
ibov_treino['Data'] = pd.to_datetime(ibov_treino['Data'],
                                     format= '%Y/%m/%d')
ibov_treino['Data'] = ...
ibov_treino['Data'].dt.strftime('%d/%m/%Y')
```

```
ibov_teste['Data'] = pd.to_datetime(ibov_teste['Data'],
                                    format= '%Y/%m/%d')

ibov_teste['Data'] = ...
ibov_teste['Data'].dt.strftime('%d/%m/%Y')

#%%
#plot exploratório
plt.plot(ibov_treino['Data'], ibov_treino['Fechamento'])
plt.xlabel('Semana')
plt.xticks(range(0, len(ibov_treino), 10), rotation=45)
plt.ylabel('Pontuação do Ibovespa')

#%%
#por fim, salvando os dados filtrados
ibov_treino.to_csv('ibov_treino.csv', index=False)
ibov_teste.to_csv('ibov_teste.csv', index=False)
```

APÊNDICE D – ALGORITMO DNN

```
# -*- coding: utf-8 -*-
"""
Created on Wed May  3 10:15:15 2023

@author: luang
"""

#%% bibliotecas
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers

#%% datasets e diretório de trabalho
os.chdir('//home//algoritmo')

custo_treino = pd.read_csv("custo_treino.csv", sep=",",
                           encoding='utf-8')
custo_teste = pd.read_csv("custo_teste.csv", sep=",",
                          encoding='utf-8')
ibov_treino = pd.read_csv("ibov_treino.csv", sep=",",
                          encoding='utf-8')
ibov_teste = pd.read_csv("ibov_teste.csv", sep=",",
                         encoding='utf-8')
demanda_treino = pd.read_csv("demanda_treino.csv", sep=",",
                             encoding='utf-8')
demanda_teste = pd.read_csv("demanda_teste.csv", sep=",",
                            encoding='utf-8')

#%% arranjo dos conjuntos de treinamento e teste
train_dataset = pd.concat([custo_treino, ibov_treino,
                           demanda_treino],axis=1,
                           ignore_index=True)

train_dataset.dropna(axis=0, how='any', inplace=True)

test_dataset = pd.concat([custo_teste, ibov_teste,
                          demanda_teste],axis=1,
                          ignore_index=True)
```

```
test_dataset.dropna(axis=0, how='any', inplace=True)

#%% removendo as datas e dividindo o dataset
train_dataset.drop(columns=[0, 2, 4], inplace=True)

train_dataset.columns = ["Custo (R$/MWh)", "Ibov",
                         "Demanda (MW)"]

test_dataset.drop(columns=[0, 2, 4], inplace=True)

test_dataset.columns = ["Custo (R$/MWh)", "Ibov",
                        "Demanda (MW)"]

#%% inspeção dos dados e análise estatística
sns.pairplot(train_dataset, diag_kind="kde")

train_stats = train_dataset.describe()

#remoção dos alvos
train_stats.pop("Demanda (MW)")
train_stats = train_stats.transpose()

#%% separação das etiquetas (labels)
#criando datasets para não modificar os originais
train_features = train_dataset.copy()
test_features = test_dataset.copy()

train_labels = train_features.pop('Demanda (MW)')
test_labels = test_features.pop('Demanda (MW)')

# os dados serão normalizados por questão de boas práticas,
# e para aumentar a eficiência do algoritmo
norm = tf.keras.layers.Normalization(axis=-1)

norm.adapt(np.array(train_features))

#%% construção do modelo de aprendizagem profunda (DNN)
def build_and_compile_model(normalizer):
   model = keras.Sequential([
       normalizer,
       layers.Dense(64,activation='relu'),
       layers.Dense(64, activation='relu'),
       layers.Dense(1)
  ])

  model.compile(loss='mean_absolute_error',
                optimizer=tf.keras.optimizers.Adam(0.001))
  return model
```

```
#%% treinamento do modelo
dnn_model = build_and_compile_model(norm)
dnn_model.summary()

history = dnn_model.fit(train_features,
                        train_labels,validation_split=0.2,
                        verbose=0, epochs=300)

#%% gráfico temporal
def plot_loss(history):
    plt.figure()
    plt.plot(history.history['loss'], label='Erro')
    plt.plot(history.history['val_loss'],
             label='Erro de validação')
    plt.xlabel('Época')
    plt.ylabel('Erro [MW]')
    plt.legend()
    plt.grid(True)

plot_loss(history)

#%% teste do modelo
test_results = {}
test_results['Modelo_DNN'] = dnn_model.evaluate(test_features,
                                                test_labels,
                                                verbose=0)

#%% predição da demanda (MW) usando o conjunto de teste
test_predictions = ...
dnn_model.predict(test_features).flatten()

plt.figure()
a = plt.axes(aspect='equal')
plt.scatter(test_labels, test_predictions)
plt.xlabel('Valores Verdadeiros [MW]')
plt.ylabel('Previsões [MW]')
lims = [70, 90]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)

#%% distribuição de erros
error = test_predictions - test_labels

plt.figure()
plt.hist(error, bins = 25)
plt.xlabel("Erro de Predição [MW]")
_ = plt.ylabel("Quantidade")
plt.xticks(np.arange(-7, 8, 1))
```

```
#%% erros percentuais
error_p = (error/test_labels)*100

plt.figure()
plt.hist(error_p, bins=10)
plt.title('Erro de Previsão')
plt.ylabel('Erro percentual (%)')
plt.xticks(labels=[])

error_p_stats = error_p.describe()
error_stats = error.describe()

#%% salvando o modelo
dnn_model.save('Modelo_DNN')

#%% carregar o modelo
reloaded = tf.keras.models.load_model('Modelo_DNN')

test_results['reloaded'] = ...
reloaded.evaluate(test_features, test_labels, verbose=0)

pd.DataFrame(test_results,
             index=['Erro Médio Absoluto [MW]']).T
```