

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE
SANTA CATARINA CAMPUS FLORIANÓPOLIS DEPARTAMENTO ACADÊMICO
DE METAL-MECÂNICA BACHARELADO EM ENGENHARIA MECATRÔNICA

LUCAS SCARABELOT DA LUZ

**APLICAÇÃO DE ALGORITMOS DE LSTM PARA A PREDIÇÃO DO CONSUMO
DE ENERGIA ELÉTRICA NA CIDADE DE FLORIANÓPOLIS**

Florianópolis, julho de 2022

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA
CATARINA CAMPUS FLORIANÓPOLIS DEPARTAMENTO ACADÊMICO DE
METAL-MECÂNICA BACHARELADO EM ENGENHARIA MECATRÔNICA**

LUCAS SCARABELOT DA LUZ

**APLICAÇÃO DE ALGORITMOS DE LSTM PARA A PREDIÇÃO DO CONSUMO
DE ENERGIA ELÉTRICA NA CIDADE DE FLORIANÓPOLIS**

Trabalho de Conclusão de Curso submetido
ao Instituto Federal de Educação, Ciência e
Tecnologia de Santa Catarina como parte dos
requisitos para obtenção do título de
Engenheira

Orientador:
Prof. Gregory Chagas Da Costa Gomes

Florianópolis, 2022

Ficha de identificação da obra elaborada pelo autor.

Scarabelot da Luz, Lucas

Aplicação De Algoritmos De LSTM Para A Predição Do
Consumo De Energia Elétrica Na Cidade De Florianópolis / Lucas
Scarabelot da Luz; orientação de Gregory Chagas
da Costa Gomes. - Florianópolis, SC, 2023.
50 p.

Trabalho de Conclusão de Curso (TCC) - Instituto Federal
de Santa Catarina, Câmpus Florianópolis. Bacharelado
em Engenharia Mecatrônica. Departamento
Acadêmico de Metal Mecânica.
Inclui Referências.

1. Machine Learning. 2. Forecasting. 3. Séries temporais.
4. Long short-term memory. 5. Redes neurais.
I. Chagas da Costa Gomes, Gregory. II. Instituto Federal
de Santa Catarina. III. Aplicação De Algoritmos
De LSTM Para A Predição Do Consumo De Energia Elétrica
Na Cidade De Florianópolis.

FLORIANÓPOLIS, 2022

**APLICAÇÃO DE ALGORITMOS DE LSTM PARA A PREDIÇÃO DO CONSUMO
DE ENERGIA ELÉTRICA NA CIDADE DE FLORIANÓPOLIS**

LUCAS SCARABELOT DA LUZ

Este trabalho foi julgado adequado para obtenção do título de bacharel em engenharia mecatrônica e aprovado na sua forma final pela banca examinadora do Curso Superior de Engenharia Mecatrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 12 de dezembro, 2022.

Banca Examinadora:

 Documento assinado digitalmente
GREGORY CHAGAS DA COSTA GOMES\;gregory.c
Data: 02/03/2023 13:27:28-0300
CPF: 073.008.969-01
Verifique as assinaturas em <https://v.ifsc.edu.br>

Gregory Chagas Da Costa Gomes

Documento assinado digitalmente
 MAURICIO EDGAR STIVANELLO
Data: 23/02/2023 20:20:21-0300
Verifique em <https://verificador.iti.br>

Maurício Edgar Stivanello

 Documento assinado digitalmente
MARIO DE NORONHA NETO
Data: 24/02/2023 08:25:08-0300
CPF: 003.859.519-22
Verifique as assinaturas em <https://v.ifsc.edu.br>

Mario de Noronha Neto

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, José e Luzia, que sempre fizeram questão de me proporcionar as melhores condições para que eu pudesse buscar uma educação de qualidade, e em nenhum momento tiveram dúvida alguma do meu potencial.

A todos os colegas que conheci ao longo destes anos, que se mostraram fundamentais no meu progresso pessoal e profissional.

Também gostaria de agradecer a cada professor e funcionário do IFSC pelo empenho e dedicação para que todos os alunos pudessem ter uma experiência de ensino incrível e engrandecedora.

Ao professor Gregory Chagas Da Costa Gomes dedico um agradecimento especial por sempre estar disposto e entusiasmado para compartilhar seus conhecimentos e me orientar nesta última parte da caminhada na graduação.

RESUMO

Considerando o grande aumento na demanda de tecnologias envolvendo inteligência artificial e *Machine Learning* (ML) que se utilizam de arquiteturas baseadas em *deep learning*, como *Long short-term memory* (LSTM), este trabalho tem como principal objetivo fundamentar a teoria que justifica o uso destas tecnologias desde os fundamentos do ML, exemplos de utilização, definição de séries temporais e previsão de dados, até o treinamento e avaliação de modelos, de forma didática e introdutória utilizando como meio uma plataforma online interativa sendo ela o Kaggle, para predição de consumo de energia elétrica no município de Florianópolis, fazendo uso de uma base de dados obtida através da CELESC (Centrais Elétricas de Santa Catarina S.A).

Palavras-Chave: *Machine Learning*. *Forecasting*. Séries temporais. Long short-term memory. RNN. Redes neurais. Jupyter Notebook

ABSTRACT

Considering the increase in demand for technologies involving artificial intelligence and Machine Learning (ML) that use architectures based on deep learning, such as Long short-term memory (LSTM), this undergraduate thesis has as main objective to support the theory that justifies the use of these technologies from the fundamentals of ML, examples of use, definition of time series and data forecast, to the training and evaluation of models, in a didactic and introductory way using an interactive online platform (Jupyter/Kaggle) for electricity consumption prediction in the city of Florianópolis, using a database obtained through CELESC (Centrais Elétricas de Santa Catarina S.A).

Palavras-Chave: *Machine Learning. Forecasting.* Time series. Long short-term memory. RNN. Neural networks. Jupyter Notebook

LISTA DE FIGURAS

Figure 1: Diferença nas abordagens entre algoritmos tradicionais e de aprendizado de máquina	17
Figure 2: Exemplo de caso de regressão	18
Figure 3: Multilayer perceptron	20
Figure 4: Rede neural recorrente	21
Figure 5: Problema da dissipação do gradiente	21
Figure 6: Bloco de memória LSTM	22
Figure 7: Exemplo de uma rede LSTM	23
Figure 8: Exemplos de séries temporais	24
Figure 9: Consumo residencial de energia elétrica no Brasil entre 1979 e 2022	25
Figure 10: Produção de colheitadeiras no Brasil entre 2000 e 2022	26
Figure 11: Página inicial do Kaggle	27
Figure 12: Kaggle notebook	28
Figure 13: Visualização da planilha	29
Figure 14: Ambiente do Jupyter Notebook	31
Figure 15: Tratamento de dados	32
Figure 16: Dados após tratamento inicial	32
Figure 17: Tratamento de dados	33
Figure 18: Consumo de energia de classe comercial em Florianópolis de 1994 a 2022	34
Figure 19: Gráficos de tendência, sazonalidade e resíduos	34
Figure 20: Janelamento de dados	35
Figure 21: Função de Janelamento de Dados	36
Figure 22: Janelamento de dados	36
Figure 23: Divisão dos conjuntos de dados	37
Figure 24: Pacotes Keras	37
Figure 25: Construção do modelo LSTM	38
Figure 26: Desempenho do modelo 1 no <i>training set</i>	41
Figure 27: Desempenho do modelo 1 no <i>validation set</i>	41
Figure 28: Desempenho do modelo 1 no <i>test set</i>	42

Figure 29: Desempenho do modelo 2 no <i>training set</i>	42
Figure 30: Desempenho do modelo 2 no <i>validation set</i>	43
Figure 31: Desempenho do modelo 2 no <i>test set</i>	43
Figure 32: Desempenho do modelo 3 no <i>training set</i>	44
Figure 33: Desempenho do modelo 3 no <i>validation set</i>	44
Figure 34: Desempenho do modelo 3 no <i>test set</i>	45

LISTA DE TABELAS

Tabela 1: Exemplo de uma rede LSTM.....	40
---	----

SUMÁRIO

1 INTRODUÇÃO	15
1.1 Objetivo Geral	15
1.2 Objetivos Específicos	16
2 FUNDAMENTAÇÃO TEÓRICA.....	17
2.2 Tipos de aprendizado de máquina	18
2.2.1 Aprendizado supervisionado.....	18
2.2.2 Redes Neurais.....	19
2.2.2.1 Arquitetura LSTM	22
2.4 Fundamentos de séries temporais.....	24
2.5 Kaggle	26
3 METODOLOGIA.....	29
3.1 Base de dados utilizado.....	29
3.2 Jupyter Notebook.....	30
3.3 Tratamento dos dados	31
3.2 Análise exploratória	34
3.3 Janelamento de dados	35
3.4 Construção e treinamento do modelo	37
4 RESULTADOS	39
4.1 Treinamento.....	39
4.2 Resultados obtidos.....	39
4.3 Avaliação dos resultados.....	44
5 CONSIDERAÇÕES FINAIS.....	46

1 INTRODUÇÃO

Com o crescimento urbano, também cresce a necessidade de constante fornecimento de energia elétrica, sendo assim, a previsão correta deste consumo se torna vital para países em desenvolvimento, tendo em vista que se superestimado, pode causar gastos desnecessários devido sua capacidade total não ser utilizada, e caso subestimado, pode levar a falhas no abastecimento (KAYTEZ ET AL.,2015).

Há diversas maneiras de se realizar estas previsões, e com o poder computacional cada vez mais acessível, é possível perceber cada vez mais métodos que se utilizam de inteligência artificial e aprendizado de máquina para descrever os comportamentos no consumo de energia.

Diversos trabalhos mostram a construção de modelos de predição se utilizando de diversas técnicas. Em Kaytez (2015), os autores optaram por utilizar modelos de regressão, redes neurais e máquina de vetores de suporte para realizar suas predições.

Observando a relevância do tópico de utilização de métodos de aprendizado de máquina e a importância da previsão de dados de consumo de energia, foi escolhida a arquitetura *LSTM (Long short-term memory)* para construção e avaliação de um modelo de predição. A arquitetura LSTM se mostra bastante eficaz na previsão de dados baseado em séries temporais (GRAVES, 2012).

Para isto será feito uma revisão da fundamentação teórica abordando temas como aprendizado de máquina, métodos utilizados para predição de dados, redes neurais recorrentes, arquitetura LSTM, parâmetros de avaliação e séries temporais e suas características.

Portanto este trabalho tem como finalidade a documentação da construção e avaliação de um modelo para previsão de dados de consumo de energia de classe comercial no município de Florianópolis entre os anos de 1994 à 2022.

1.1 Objetivo Geral

Aplicar algoritmos de redes neurais recorrentes para previsão de dados de consumo de energia no município de Florianópolis, consequentemente avaliando o modelo e organizando o desenvolvimento em uma plataforma virtual interativa para fins didáticos e introdutórios sobre o tema.

1.2 Objetivos Específicos

- a) Obter e tratar através de bibliotecas como NumPy e Pandas, uma base de dados contemplando o consumo de energia elétrica no município de Florianópolis
- b) Fazer a análise destes dados e validar a possibilidade de serem utilizados para treinar modelos de predição através de redes neurais recorrentes
- c) Validar a efetividade dos modelos obtidos através de métricas já estabelecidas
- d) Documentar todo o processo em um Jupyter Notebook com intuito de servir como base introdutória sobre séries temporais, análise de dados e predição
- e) Exportar o documento gerado no Jupyter Notebook para a plataforma Kaggle, de forma a facilitar o compartilhamento e interatividade com o código desenvolvido

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo abordará a fundamentação teórica em que o trabalho foi baseado e justificado, tais como conceitos de aprendizado de máquina (*machine learning*), ciência de dados e séries temporais.

2.1 Fundamentos do aprendizado de máquina

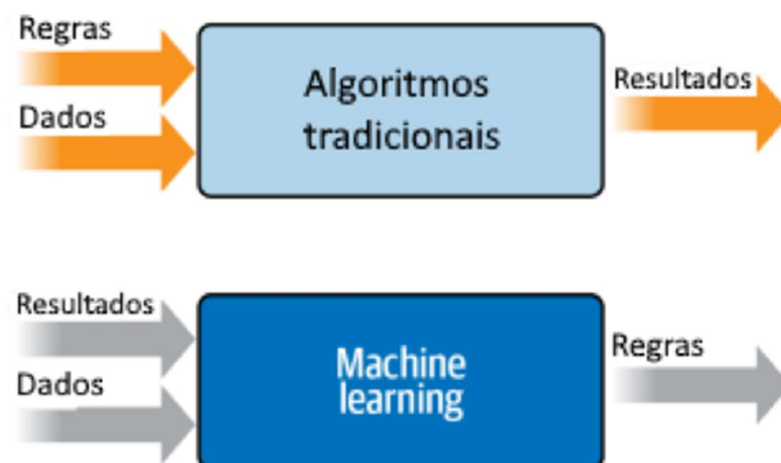
Aprendizado de máquina é definido como um conjunto de métodos que podem automaticamente detectar padrões em conjuntos de dados, e usar esses padrões para realizar previsões, e outros tipos de decisões (Murphy, 2012).

Outra definição, de acordo com Burkov (2019), seria o processo de solucionar um problema através de 1) coletar dados e 2) por meio de algoritmos construir modelos estatísticos baseados no conjunto de dados coletado.

De acordo com Moroney (2020) diferente de algoritmos comuns, onde as entradas são os dados e as regras que o computador deve seguir para chegar em determinado resultado, no ramo de aprendizado de máquina estes fatores são alterados, fazendo com que agora as entradas sejam os resultados e os dados, para assim determinarmos as regras que governarão o algoritmo.

A figura 1 a seguir ilustra a diferença das duas abordagens.

Figure 1: Diferença nas abordagens entre algoritmos tradicionais e de aprendizado de máquina



Fonte: Moroney, L. Adaptado (2020)

2.2 Tipos de aprendizado de máquina

Quando falamos sobre machine learning temos diferentes tipos de abordagens dependendo dos problemas a serem solucionados. As principais formas são: aprendizado supervisionado e aprendizado não supervisionado.

O problema em questão se trata de um caso de aprendizado supervisionado, que será mais bem descrito no próximo capítulo.

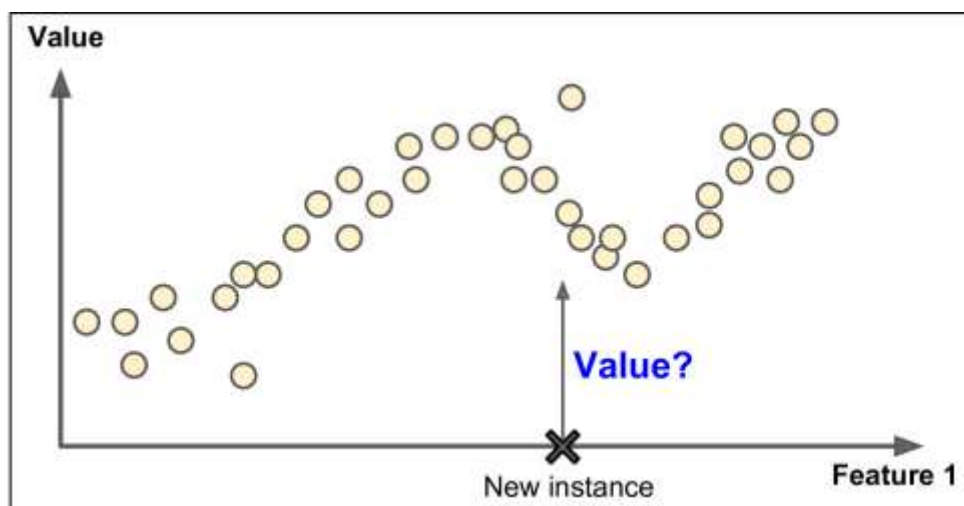
2.2.1 Aprendizado supervisionado

Segundo Murphy (2022) aprendizado supervisionado é a técnica mais comum e amplamente utilizada de aprendizado de máquina.

Neste caso, o objetivo é criar um modelo f a partir de entradas $x \in X$ para obter resultados $y \in Y$, os valores de x também são conhecidos como *features* (*senal de entrada para o modelo preditivo*) e os de y como *labels* (*rótulo*) (MURPHY, 2022).

Um problema pertinente nos casos de aprendizado supervisionado é a regressão, onde é feita uma predição de um valor numérico, tal como o preço de um carro, dado um conjunto de *features* (*dados de entrada do modelo preditivo*) como quilometragem, marca, ano, etc. (GÉRON, 2019). Na figura 2 abaixo, podemos ver um exemplo de caso de regressão, onde uma nova instancia será predita baseada na *feature 1*.

Figure 2: Exemplo de caso de regressão



Fonte: GÉRON (2019, p. 9)

Os tipos de aprendizados supervisionados mais comuns incluem:

- K-ésimo vizinho mais próximo
- Regressão Linear
- Regressão Logística
- Máquina de Vetores de Suporte
- Árvores de Decisão e Floresta Aleatória
- Redes Neurais

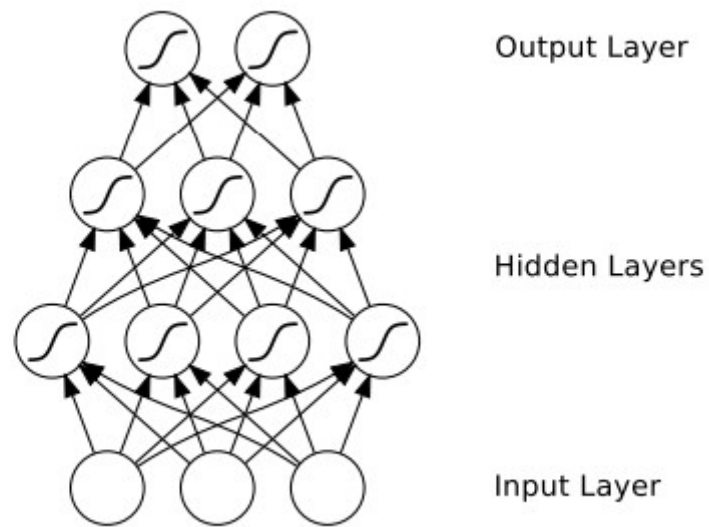
No tópico a seguir será abordado o tema de redes neurais.

2.2.2 Redes Neurais

Os pássaros nos inspiraram a voar, as plantas de bardana inspiraram o velcro e inúmeras outras invenções foram inspiradas pela natureza. Parece lógico, então, olhar para a arquitetura do cérebro para se inspirar em como construir uma máquina inteligente. Esta é a ideia-chave que provocou o nascimento das redes neurais artificiais (ANNs). (GÉRON, 2019, p. 277, tradução nossa)

De acordo com Géron (2019, p. 277) as redes neurais artificiais são o núcleo do que conhecemos por *deep learning*. Elas são versáteis, poderosas e escaláveis, as fazendo ideal para atacar problemas complexos de aprendizado de máquina. Uma distinção importante a ser fazer é a diferença entre redes neurais cíclicas e não-cíclicas. Redes neurais cíclicas são comumente referenciadas como RNN (*Recurrent Neural Networks*) e redes neurais não-cíclicas como FNN (*Feedforward Neural Networks*). Um exemplo bastante comum de FNN é o *perceptron* (ROSENBLATT, 1958) e *multilayer perceptron* (MLP) (BISHOP, 1995) como pode-se observar na figura 5.

Figure 3: Multilayer perceptron



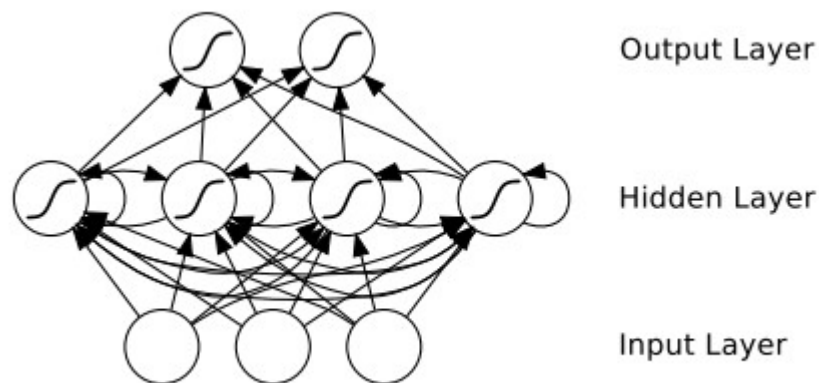
Fonte: GRAVES. A (2022, p. 13)

Na figura 3 podemos perceber a existência de três tipos de *layers* (camadas), os *input layers* (camadas de entrada), *hidden layers* (camadas ocultas) e *output layers* (camadas de saída).

- Input layers são a camada onde os dados são inseridos na rede neural
- Hidden layers são responsáveis, junto ao input layer, por gerar os coeficientes, ou pesos, das conexões feitas dentro da FNN.
- Output layers é a camada onde obtemos o resultado do modelo.

Observa-se a propagação partindo do *input layer*, para os *hidden layers* e consequentemente aos *output layers*, portanto a saída de um modelo MLP depende apenas dos dados de entrada da rede neural.

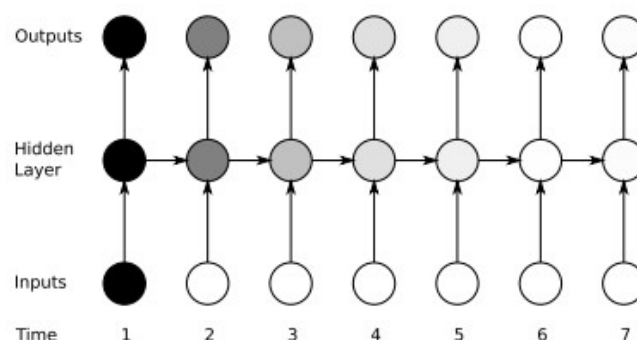
Nos casos de RNNs, o treinamento do modelo possui ciclos de *feedback* internos, permitindo conexões cíclicas entre as camadas. Uma grande variedade de RNNs foram propostas ao longo dos anos (GRAVES, 2012). Na figura 4 é possível visualizar a estrutura de uma rede neural recorrente.

Figure 4: Rede neural recorrente

Fonte: GRAVES. A (2012, p. 18)

Em uma arquitetura MLP as entradas apenas podem ser mapeadas diretamente aos *output layers*, enquanto as redes neurais recorrentes permitem a estrutura mapear o histórico completo das entradas para cada *output layers*. O ponto principal é que redes neurais recorrentes permitem uma “memória” de uma entrada anterior persistir e afetar o estado da rede neural, assim afetando o resultado de saída (GRAVES, 2012 p.19).

Um das grandes adversidades das RNNs é o problema da dissipação do gradiente, como exemplificado no diagrama a seguir:

Figure 5: Problema da dissipação do gradiente

Fonte: GRAVES. A (2012, p. 32)

A coloração indica a sensibilidade da rede neural as entradas em $t = 1$, é possível perceber que esta sensibilidade cai ao longo do tempo através do tempo, até que a rede neural “esquece” a primeira entrada. Assim os gradientes propagados

durante o treinamento chegam as camadas finais com valores insignificantes para o modelo.

Uma das principais arquiteturas propostas para solucionar este problema é o *LSTM* (*Long short-term memory*) que será abordado em detalhe no tópico a seguir.

2.2.2.1 Arquitetura LSTM

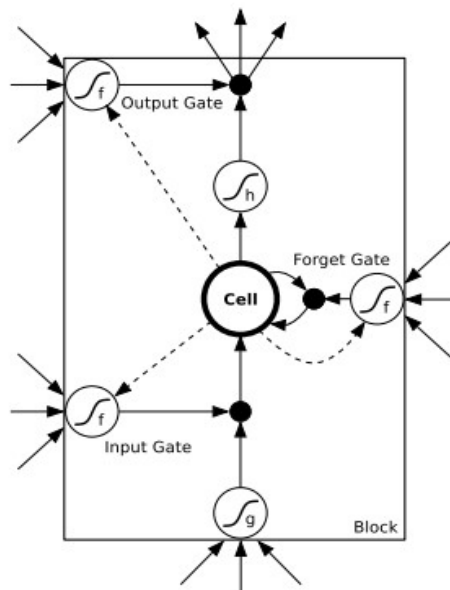
O LSTM surge como uma solução para o problema da dissipação do gradiente em 1997, no trabalho Long short-term memory (S Hochreiter, J Schmidhuber, 1997).

A arquitetura LSTM se utiliza de um conjunto de subredes recorrentemente conectadas, conhecidas como blocos de memória. Cada bloco contém uma ou mais células de memórias interconectadas e três portões (*gates*): a *input gate*, *output gate* e *forget gate*.

- *Forget gate*: Utilizada para remover as informações que não são mais úteis para o bloco de memória
- *Input gate*: Onde é feita a adição de informações úteis ao bloco de memória
- *Output gate*: Tem como função extrair as informações do bloco de memória

No diagrama 6 a seguir é possível ver a anatomia de um bloco de memória da arquitetura LSTM.

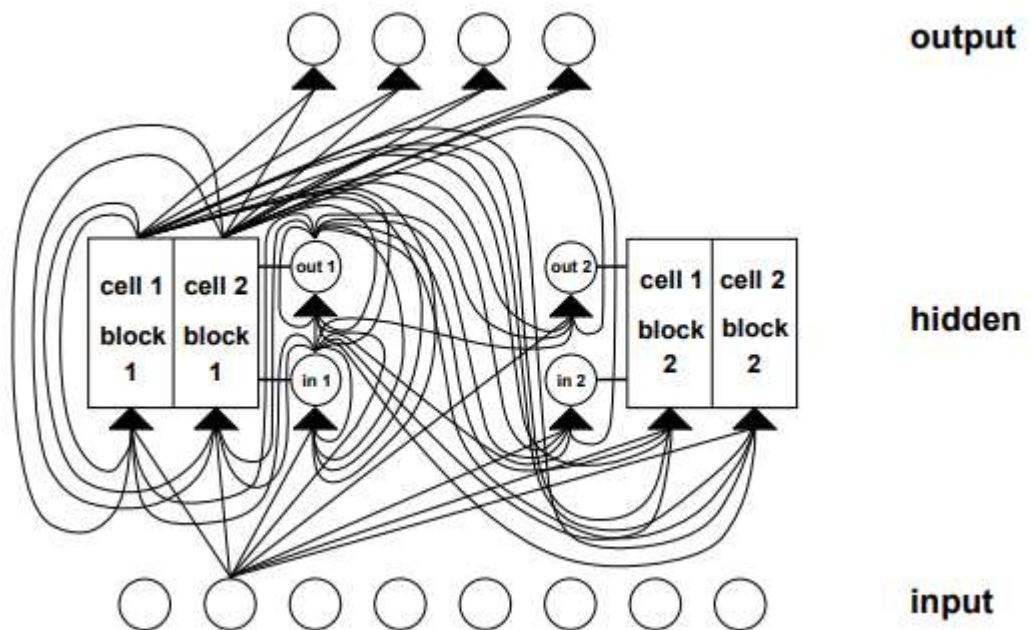
Figure 6: Bloco de memória LSTM



Fonte: GRAVES. A (2012, p. 33)

Abaixo observa-se um exemplo de uma rede LSTM contando com 8 *inputs*, 4 *outputs*, 2 blocos de memórias com 2 células cada.

Figure 7: Exemplo de uma rede LSTM



Fonte: Hochreiter, Y (1997, p. 9)

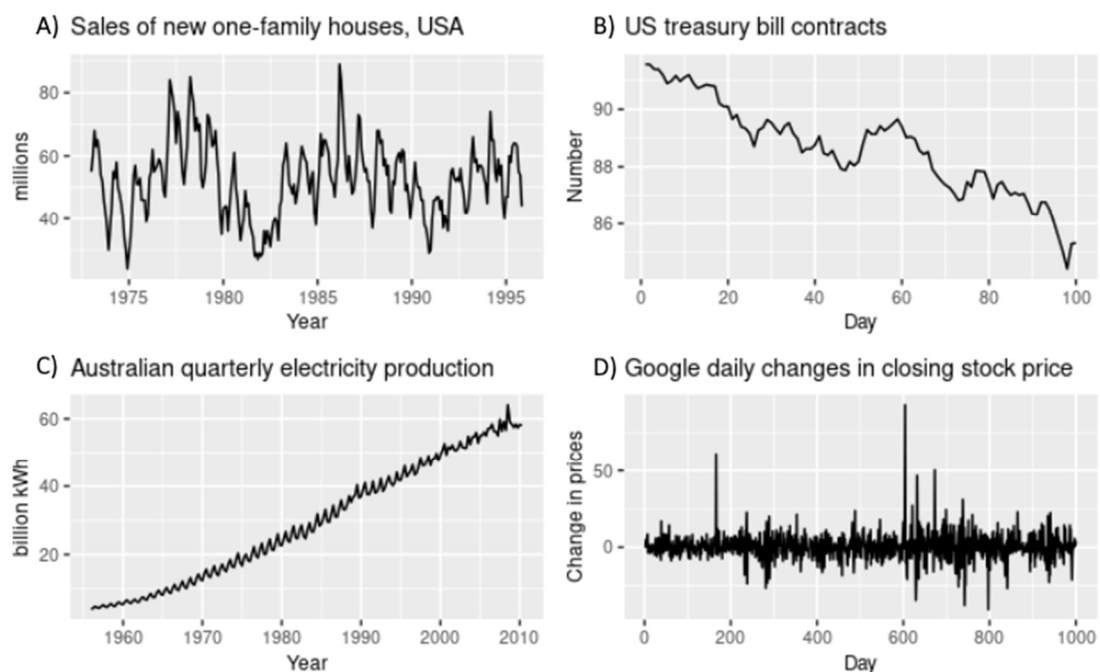
A arquitetura LSTM se provou bastante útil nos últimos anos sendo capaz de resolver diversos problemas reais, como a predição de demanda de energia elétrica (Daniel L. Et al. 2016), progressão de casos de COVID-19 (Ignacio Segovia-Dominguez. Et al. 2021).

No próximo tópico será discutido os fundamentos e características de séries temporais.

2.4 Fundamentos de séries temporais

De acordo com Box et al. (2016) séries temporais são definidas por uma coleção de observações feitas sequencialmente ao longo do tempo. Diversos fenômenos podem ser descritos na forma de uma série temporal, como a quantidade de energia produzida por um país ao longo dos anos, vendas de casas em um determinado período ou até mesmo as oscilações diárias no preço das ações do Google. (FIGURA 8).

Figure 8: Exemplos de séries temporais

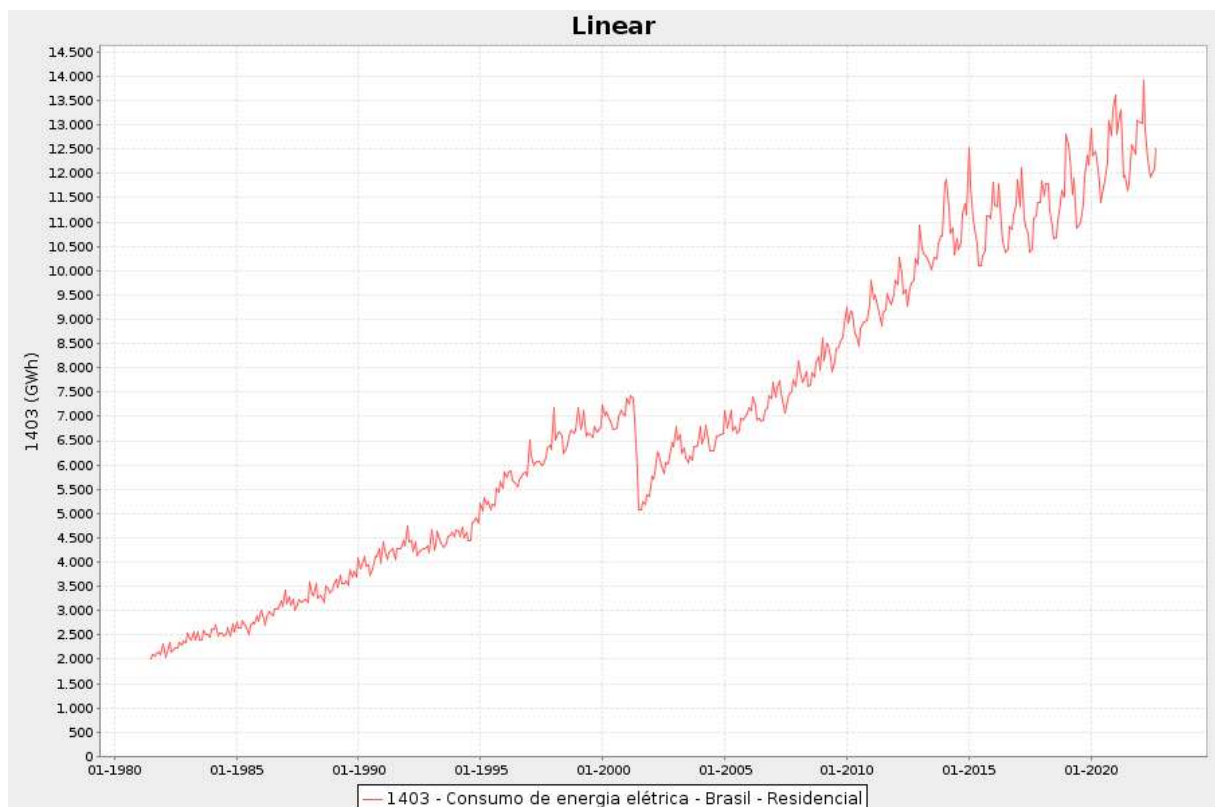


Fonte: HYNDMAN, R.J (2021)

Séries temporais podem apresentar três tipos de padrões, sendo eles **sazonalidade, tendência e ciclos**.

Sazonalidade: um padrão sazonal ocorre nos dados de uma série temporal quando ela é afetada por fatores sazonais, como um tempo específico, podendo ser o mês ou o dia distinto da semana. Temos como exemplo o consumo de energia no Brasil, que apresenta seus picos no final e início de cada ano, tendo como um dos fatores o aumento das temperaturas, por exemplo. Na figura 9 abaixo, é possível perceber que o gráfico apresenta forte sazonalidade anual.

Figure 9: Consumo residencial de energia elétrica no Brasil entre 1979 e 2022

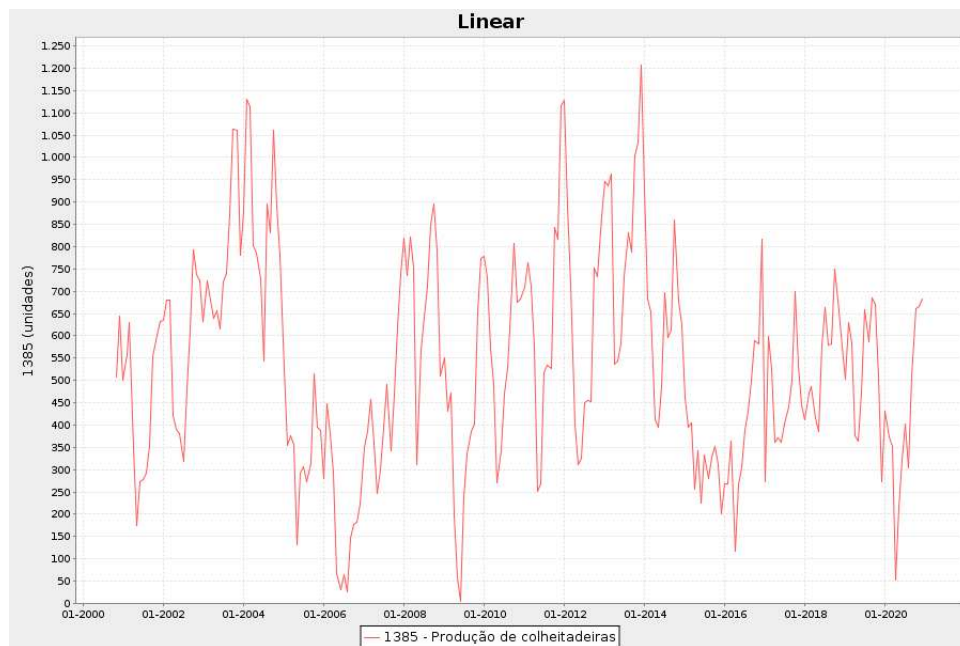


Fonte: Banco Central - Sistema Gerenciador de Séries Temporais (2022)

Tendência: dados apresentam uma tendência quando há uma variação contínua no longo prazo, podendo ser ela negativa ou positiva. O gráfico da figura 9 apresenta comportamentos claros de tendência positiva através do tempo.

Ciclo: um comportamento cíclico ocorre quando os dados apresentam variações em frequências inconstantes. Na figura 10 abaixo, é possível perceber que em intervalos de 1 a 3 anos, temos um comportamento cíclico de forte aumento na produção de colheitadeiras no Brasil, seguido consequentemente por uma forte queda.

Figure 10: Produção de colheitadeiras no Brasil entre 2000 e 2022



Fonte: Banco Central - Sistema Gerenciador de Séries Temporais (2022)

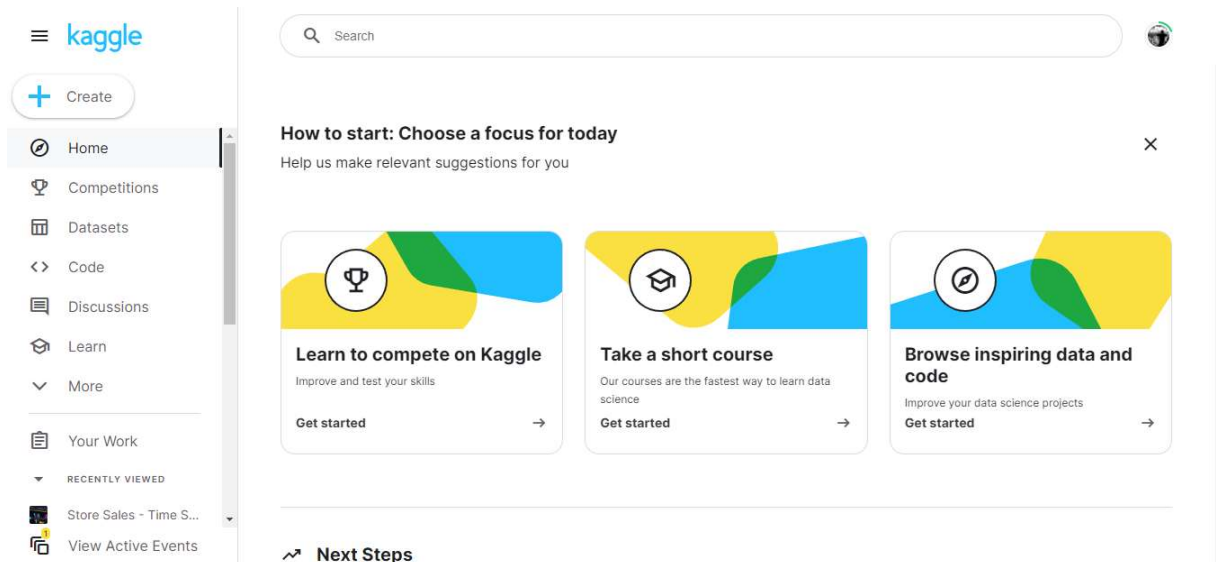
2.5 Kaggle

Neste capítulo será feita uma breve análise da plataforma Kaggle, onde o desenvolvimento do modelo de predição será documentado e compartilhado.

A plataforma Kaggle, fundada em 2010 por Anthony Goldbloom, é uma comunidade de desenvolvimento voltada para ciência de dados e aprendizado de

máquina, permitindo aos usuários compartilharem suas pesquisas e desenvolver seus modelos em um ambiente virtual *web-based*.

Figure 11: Página inicial do Kaggle



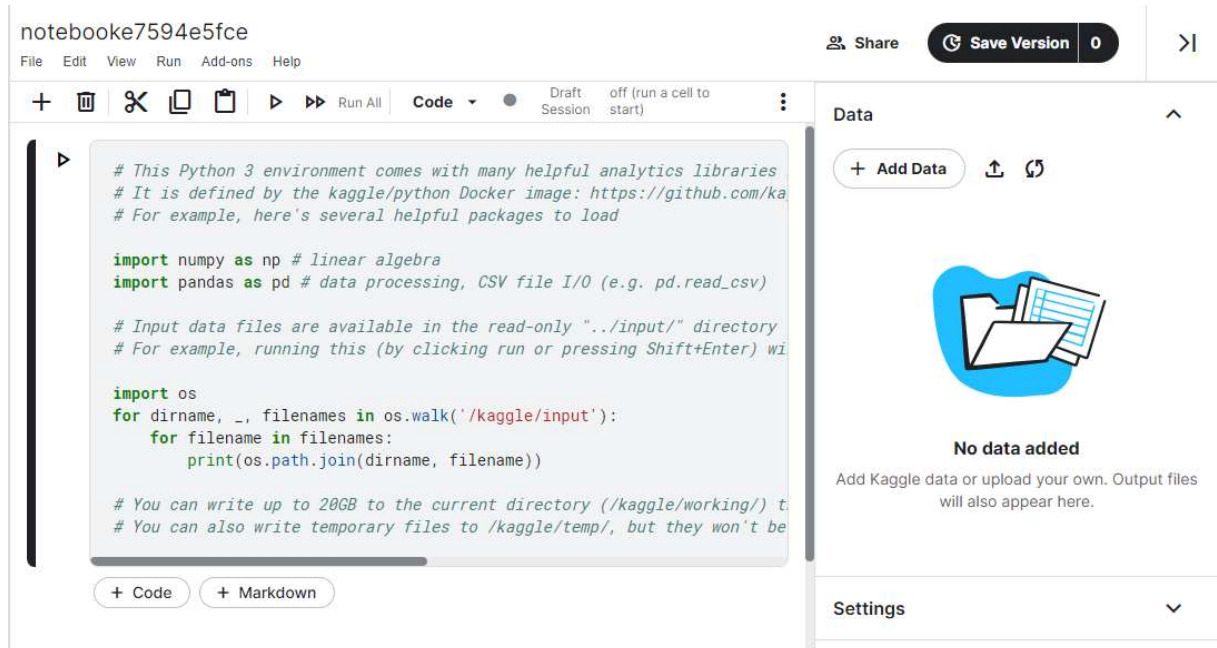
Fonte: Do Autor (2022)

Neste ambiente também são disponibilizados cursos correlatos a área de ciência de dados, grande quantidade de bases de dados, juntamente a competições onde os usuários podem participar.

Também são oferecidos os *Kaggle Notebooks*, similares ao *Jupyter Notebook* porém totalmente online, como pode-se observar na figura 12.

Toda a construção e documentação do processo de desenvolvimento será feita dentro de um *Kaggle Notebook* para posterior compartilhamento, para que sirva como porta de entrada para aprendizado de máquina, séries temporais e redes neurais recorrentes, especificamente o LSTM.

Figure 12: Kaggle notebook



The image shows a Kaggle notebook interface. The top bar displays the notebook ID "notebooke7594e5fce" and buttons for "Share", "Save Version", and "0". Below the top bar is a menu bar with "File", "Edit", "View", "Run", "Add-ons", and "Help". The main area contains a code cell with the following Python code:

```
# This Python 3 environment comes with many helpful analytics libraries
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list the files in the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) to
# You can also write temporary files to /kaggle/temp/, but they won't be
```

Below the code cell are buttons for "+ Code" and "+ Markdown". On the right side, there is a "Data" sidebar with a "+ Add Data" button, an icon of a folder with documents, and the text "No data added". Below this, it says "Add Kaggle data or upload your own. Output files will also appear here." At the bottom of the sidebar is a "Settings" button with a dropdown arrow.

Fonte: Do Autor (2022)

3 METODOLOGIA

Este trabalho será realizado através do método exploratório e quali-quantitativo, utilizando-se de uma forte fundamentação teórica, estudo da linguagem Python e suas ferramentas de aprendizado de máquina, juntamente ao estudo de séries temporais e predição de dados.

Nos tópicos a seguir será descrito o processo de obtenção da base de dados utilizado, tratamento dos dados, análise exploratória, construção de um modelo de previsão utilizando RNNs (Redes neurais recorrentes), mais especificamente LSTMs e exportação da documentação do desenvolvimento para o *Kaggle*.

3.1 Base de dados utilizado

A base de dados utilizada foi o de consumo de energia no estado de Santa Catarina. Contemplando um período de 28 anos, partindo de janeiro de 1994 a setembro de 2022. Os dados foram retirados da seção de dados de consumo do site da CELESC (Centrais Elétricas de Santa Catarina), possuindo duas planilhas, uma relacionada ao número de unidades consumidoras em Santa Catarina, e a outra que possui informações sobre o consumo e vieram no formato .xlsx.

Para utilização, antes foi necessário realizar um processo de tratamento e organização dos dados para possibilitar a análise exploratória, como também deixar os dados no formato correto para utilização das ferramentas de aprendizado de máquina disponibilizadas pelas bibliotecas TensorFlow, Keras e Scikit-learn. Na imagem 11 abaixo podemos ver como os dados vieram organizados.

Figure 13: Visualização da planilha

tipo	cod_muni	agência	núcleo	unidade	município	cod_classe	classe	jan/94	fev/94
Cativo	1101	ARFLO	Núcleo Grande Capital	Florianópolis (sede)	Florianópolis	1	Residencial	21.357,147	20.782,785
Cativo	1101	ARFLO	Núcleo Grande Capital	Florianópolis (sede)	Florianópolis	2	Industrial	1.698,383	1.641,103
Cativo	1101	ARFLO	Núcleo Grande Capital	Florianópolis (sede)	Florianópolis	3	Comercial	14.766,175	13.420,879
Cativo	1101	ARFLO	Núcleo Grande Capital	Florianópolis (sede)	Florianópolis	4	Rural	25,908	23,863
Cativo	1101	ARFLO	Núcleo Grande Capital	Florianópolis (sede)	Florianópolis	5	Poder Público	6.393,112	4.903,650
Cativo	1101	ARFLO	Núcleo Grande Capital	Florianópolis (sede)	Florianópolis	6	Iluminação Pública	2.513,127	2.568,735
Cativo	1101	ARFLO	Núcleo Grande Capital	Florianópolis (sede)	Florianópolis	7	Serviço Público	659,419	653,971
Cativo	1101	ARFLO	Núcleo Grande Capital	Florianópolis (sede)	Florianópolis	8	Próprio	419,640	404,616
Cativo	1101	ARFLO	Núcleo Grande Capital	Florianópolis (sede)	Florianópolis	9	Revenda	-	-
Cativo	1102	ARFLO	Núcleo Grande Capital	Florianópolis (sede)	São José	1	Residencial	7.482,370	6.657,775

Fonte: Do Autor (2022)

A frequência de obtenção de dados é mensal, possuindo *features* como:

- Tipo
- Código do município
- Agência
- Núcleo
- Unidade
- Município
- Código de classe
- Classe
- Datas referentes a coleta dos dados

A classe de dados utilizada para a construção do modelo será a comercial, contemplando apenas o município de Florianópolis. Para isto foi feito um tratamento, possibilitando esta filtragem dos dados, como também a preparação para utilização na construção do modelo preditivo LSTM.

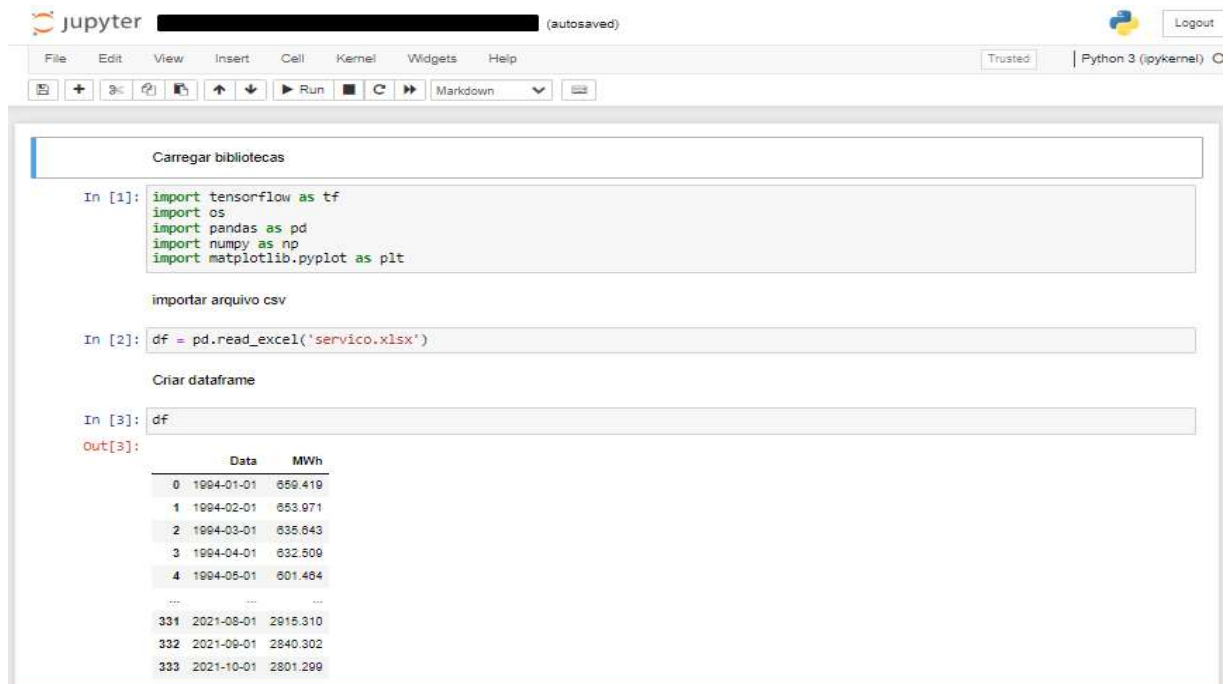
3.2 Jupyter Notebook

Para este tratamento, o ambiente utilizado foi o *Jupyter Notebook* uma plataforma *web-based* interativa de computação de dados, possibilitando um desenvolvimento mais ágil e visual.

O *Jupyter Notebook* foi desenvolvido pelo Projeto Jupyter, uma organização sem fins lucrativos com o fim de desenvolver *software* de código aberto com uma interface flexível e modular. Suporta hoje em dia mais de 40 linguagens de programação, no caso deste trabalho, utiliza-se a linguagem de programação Python.

Na Figura 14 é mostrado o ambiente do *Jupyter Notebook*.

Figure 14: Ambiente do Jupyter Notebook



Fonte: Do Autor (2022)

Como visto na Figura 14, o ambiente conta com células, que podem ser ambas de texto ou código, que podem ser rodadas individualmente, facilitando a parte de *debug* e visualização de gráficos e dados utilizados. No capítulo seguinte será descrito o processo de tratamento da base de dados utilizado dentro do ambiente Jupyter.

3.3 Tratamento dos dados

O primeiro passo no tratamento dos dados utilizado é a remoção de features que não são pertinentes a construção de um modelo de predição de séries temporais, eles sendo:

- tipo
- cod_muni
- agência
- núcleo
- unidade
- cod_classe

Para isto utiliza-se a biblioteca Pandas, que permite a criação de um *DataFrame*, uma estrutura de dados bidimensional, com colunas de valores potencialmente diferentes. Pode-se pensar em um *DataFrame* como uma planilha, ou um banco de dados SQL. Na figura 15 a seguir é possível visualizar o código utilizado para remover as *features* não utilizadas.

Figure 15: Tratamento de dados

```
#Limpeza e estruturação do DataFrame
#Database retirado de https://www.celesc.com.br/home/mercado-de-energia/dados-de-consumo em 18/07/2022

import pandas as pd
import numpy as np
pd.options.mode.chained_assignment = None

#Carrega a planilha de consumo para o DataFrame df
df = pd.read_excel("C:/Users/Lucas/Desktop/Lucas/TCC/TCC - LSTM/municipio_mensal - TCC.xlsx", "Consumo MWh")
#Filtra apenas os dados referentes ao município de Florianópolis
df = df.loc[df['município']=='Florianópolis']
#São removidas as colunas tipo, cod_muni, agência, núcleo, unidade, cod_classe e município
df = df.drop(columns=['tipo', 'cod_muni', 'agência', 'núcleo', 'unidade', 'cod_classe', 'município'])
```

Fonte: Do Autor (2022)

Após as operações acima, os dados se encontram na seguinte configuração:

Figure 16: Dados após tratamento inicial

	classe	1994-01-01 00:00:00	1994-02-01 00:00:00	1994-03-01 00:00:00	1994-04-01 00:00:00	1994-05-01 00:00:00	1994-06-01 00:00:00	1994-07-01 00:00:00	1994-08-01 00:00:00	1994-09-01 00:00:00	...	2021-03-01 00:00:00	2021-04-01 00:00:00	2021-05-01 00:00:00
0	Residencial	21357.147	20782.785	19920.442	19890.395	19409.539	20029.374	18449.182	17824.86	21078.597	...	83138.639	80099.15982	48354.88444
1	Industrial	1698.383	1641.103	1744.84	1797.511	1669.668	1829.881	1727.951	1545.273	1895.187	...	3517.388	3146.53	2663.232
2	Comercial	14766.175	13420.879	14324.136	12955.039	10762.104	10620.011	10209.671	9908.564	11228.961	...	35374.574	33479.453	27832.68
3	Rural	25.908	23.863	25.417	25.898	24.958	24.585	22.323	21.72	36.874	...	45.939	49.791	43.465
4	Poder Público	6393.112	4903.65	4499.429	5233.793	4214.206	4632.583	3856.017	4024.602	5044.499	...	7233.226	7427.578	6053.575
5	Iluminação Pública	2513.127	2568.735	3028.517	2626.243	2624.639	2647.882	2659.551	2670.446	2670.446	...	3089.328	3057.086	3374.065
6	Serviço Público	659.419	653.971	635.643	632.509	601.464	574.817	598.423	549.799	583.066	...	2970.119	3148.011	2891.073
7	Próprio	419.64	404.616	453.703	362.545	390.938	318.439	333.54	330.582	367.67	...	420.683	452.864	341.151
8	Revenda	...	...	...	...	...	...	...	...	...	...	...	...	...
2670	Comercial	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	819.0756	733.4418	676.15296

Fonte: Do Autor (2022)

É possível perceber que a forma em que os dados estão arranjados não é ideal, visto que cada data individual é tida como uma *feature*, e, também que os consumos por classe aparecem diversas vezes. Sendo assim é necessário transformar todas as datas em uma *feature* chamada “Data” e cada valor corresponder a um período de

coleção de dados de consumo, juntamente com somar os dados relativos a uma mesma classe de consumo, neste caso de aplicação, a classe “Comercial”.

Para realizar estas operações, foi utilizado a função *melt()*, *groupby()* e *sum()* da biblioteca Pandas, que permitem o “despivotamento” dos dados, utilizando como eixo a coluna classe, agrupamento de dados e a soma deste dados, como demonstrado na figura 17 a seguir.

Figure 17: Tratamento de dados

```
#Transformando as datas em uma única Feature
df_floripa = df.melt(id_vars=['classe'])
#Renomeando as colunas
df_floripa.rename(columns = {'variable':'Data', 'value':'Mwh'}, inplace = True)
#convertendo as datas para o formato datetime
df_floripa['Data']=pd.to_datetime(df_floripa['Data'])
#Filtrando apenas os dados referentes a classe comercial
df_com = df_floripa.loc[df_floripa['classe']=='Comercial']
#Retirando a coluna classe
df_com_sum.drop(columns=['classe'], axis = 1, inplace=True)
#Agrupando os dados referentes a mesma data e classe
df_com_sum = df_com.groupby(['Data']).sum()
```



Data	Mwh
1994-01-01	14766.175000
1994-02-01	13420.879000
1994-03-01	14324.136000
1994-04-01	12955.039000
1994-05-01	10762.104000
...	...
2022-05-01	41944.976901
2022-06-01	37886.855337
2022-07-01	38236.300856
2022-08-01	38969.017239
2022-09-01	37268.579756

[345 rows x 1 columns]

Fonte: Do Autor (2022)

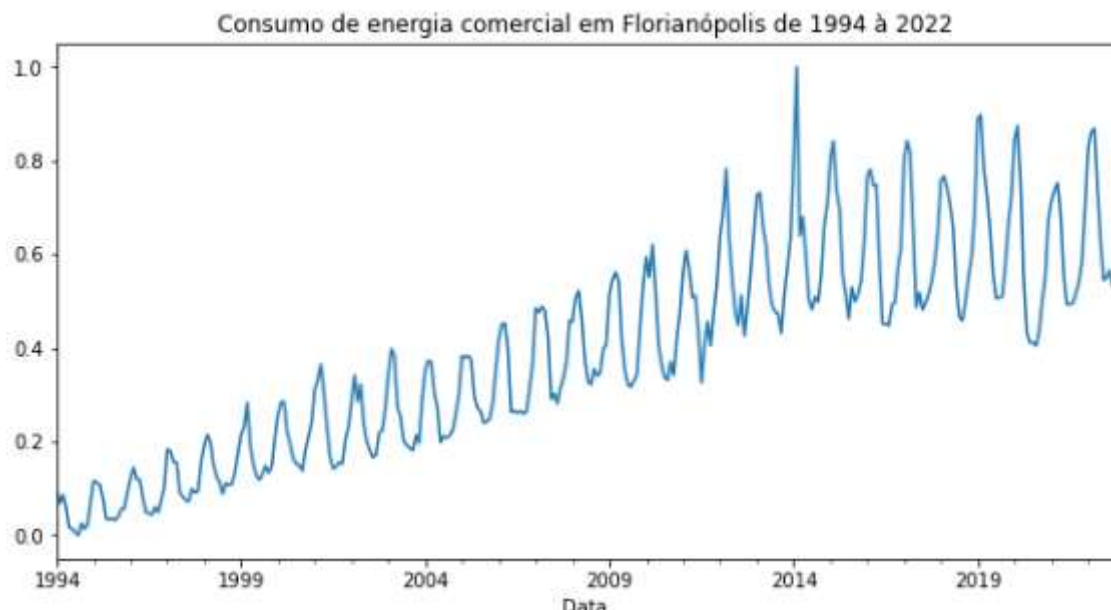
Por fim, os valores de consumo, em MWh serão normalizados através da função *MinMaxScaler()* da biblioteca *sklearn* assim mantendo os valores entre 0 e 1.

Com os dados estruturados, no próximo capítulo será abordado o tema da análise exploratória.

3.2 Análise exploratória

Para a análise exploratória, será realizada a plotagem dos gráficos de consumo pelo tempo, tendência, sazonalidade e gráfico de resíduos. Para isso foi utilizada a biblioteca *stats models*. Assim decompondo a série temporal, como mostrado nas figuras 18 e 19 a seguir.

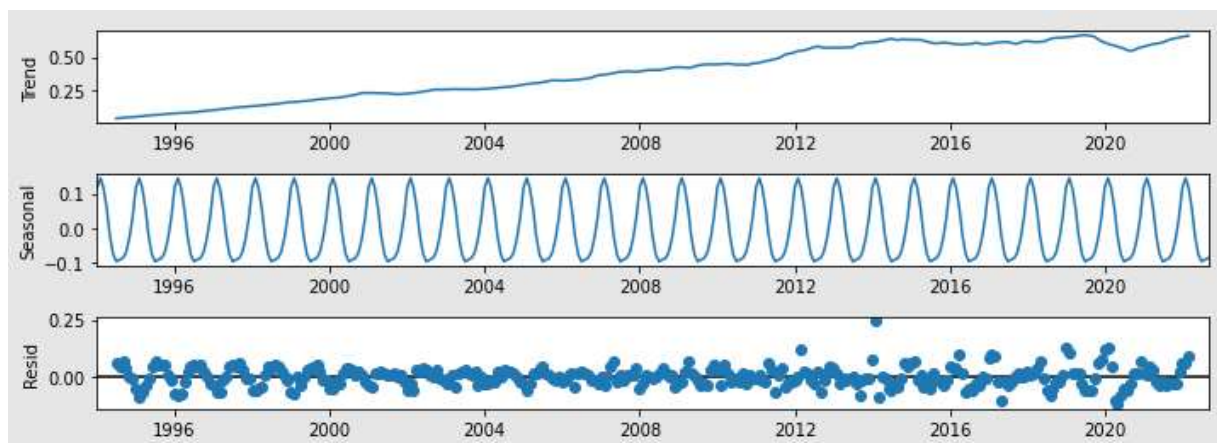
Figure 18: Consumo de energia de classe comercial em Florianópolis de 1994 a 2022



Fonte: Do Autor (2022)

Através do código a seguir, a decomposição da série é realizada e obtemos os seguintes gráficos:

Figure 19: Gráficos de tendência, sazonalidade e resíduos



Fonte: Do Autor (2022)

É possível perceber uma tendência positiva, juntamente a fortíssima sazonalidade por se tratar de dados de consumo de energia, com seus picos durante o período de verão, já que grande parte desta sazonalidade é provocada pela variação climática (BARDELIN, 2004).

O gráfico residual é que obtemos quando se retira os componentes sazonais e de tendência, assim sendo flutuações nos dados que os componentes citados não conseguem descrever.

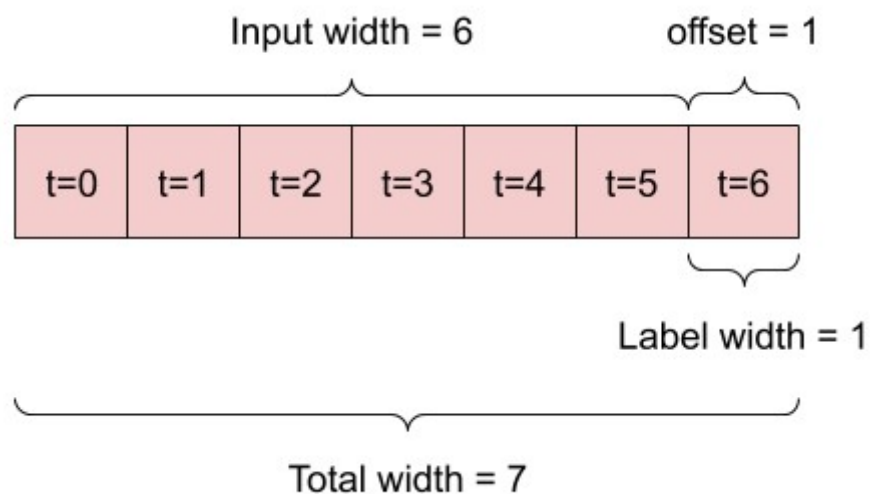
3.3 Janelamento de dados

O janelamento de dados permite que um conjunto de previsões com base em uma janela de amostras consecutivas dos dados seja feita. As principais características das janelas de entrada são:

- A largura (número de etapas de tempo) das janelas de entrada e rótulo.
- O deslocamento de tempo entre eles.
- Quais recursos são usados como entradas, rótulos ou ambos.

No exemplo abaixo, se considerarmos em meses, temos um janelamento que recebe 6 unidades de tempo de dados, tem um deslocamento de 6 e prevê 1 unidade de tempo no futuro.

Figure 20: Janelamento de dados



Fonte: TensorFlow Previsão de Séries Temporais (2022)

O janelamento utilizado para este estudo foi desenvolvido a partir da função a seguir, que recebe como argumentos o *DataFrame* utilizado como o tamanho da janela.

Figure 21: Função de Janelamento de Dados

```
def df_to_X_y(df, window_size):
    df_as_np = df.to_numpy()
    X = []
    y = []
    for i in range(len(df_as_np)-window_size):
        row = [[a] for a in df_as_np[i:i+window_size]]
        X.append(row)
        label = df_as_np[i+window_size]
        y.append(label)
    return np.array(X), np.array(y)
```

Fonte: Do Autor (2022)

Onde os *outputs* X1 e y1 representam a janela de dados utilizada para realizar uma predição, e a predição em si, respectivamente. Na figura 22 é possível visualizar o progresso do janelamento.

Figure 22: Janelamento de dados

X1 = [1, 2, 3, 4, 5, 6, 7, 8, 9] y1 = [10]

X1 = [2, 3, 4, 5, 6, 7, 8, 9, 10] y1 = [11]

Fonte: Do Autor (2022)

Com os dados devidamente tratados, e a função de janelamento definida, o próximo capítulo tratará da construção e treinamento de um modelo de predição utilizando arquitetura LSTM.

3.4 Construção e treinamento do modelo

Para realizar o treinamento do modelo, antes foi feita uma divisão dos dados em 3 partes, o *training set* (*conjunto de treinamento*), *validation set* (*conjunto de validação*) e *test set* (*conjunto de teste*), sendo caracterizados por:

- *Training set* sendo o conjunto utilizado para treinar o modelo, sendo composto por pares de valores de entrada e saída correspondentes
- *Validation set* utilizado para validar a performance do modelo em questão, também é utilizado como parâmetro para determinar quando o treinamento deve parar, em função de prevenir o overfitting (*sobreajuste*).
- *Test set* é o conjunto de dados que o modelo não tem acesso durante o treinamento, utilizado para comparar os dados que serão preditos com os valores reais.

O *dataset* utilizado conta com 345 entradas, datando de janeiro de 1994 a setembro de 2022, e a divisão foi realizada conforme a figura 23.

Figure 23: Divisão dos conjuntos de dados

```
X_train1, y_train1 = X1[:220], y1[:220]
X_val1, y_val1 = X1[220:300], y1[220:300]
X_test1, y_test1 = X1[300:], y1[300:]
X_train1.shape, y_train1.shape, X_val1.shape, y_val1.shape, X_test1.shape, y_test1.shape
((220, 12, 1), (220,), (80, 12, 1), (80,), (24, 12, 1), (24,))
```

Fonte: Do Autor (2022)

Após esta divisão dos dados é feita a construção do modelo, para isto utilizamos os pacotes da biblioteca *Keras* mostrados na figura 24 abaixo.

Figure 24: Pacotes Keras

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import *
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.losses import MeanSquaredError
from tensorflow.keras.metrics import RootMeanSquaredError, MeanAbsoluteError
from tensorflow.keras.optimizers import Adam
```

Fonte: Do Autor (2022)

O modelo é composto por um *input layer* com 9 entradas, referentes ao tamanho da nossa janela de dados, um *layer LSTM* com 128 células, dois *hidden layers*, ambos com 32 *nodes* e um *output layer* de dimensão 1, que será o valor da nossa predição, como descrito no código mostrado na figura 25 abaixo.

Figure 25: Construção do modelo LSTM

```
model1 = Sequential()
model1.add(InputLayer((WINDOW_SIZE, 1)))
model1.add(LSTM(128))
model1.add(Dense(32, 'relu'))
model1.add(Dense(32, 'relu'))
model1.add(Dense(1, 'linear'))

model1.summary()
```

Fonte: Do Autor (2022)

No próximo capítulo serão discutidos os resultados obtidos durante este estudo.

4 RESULTADOS

Neste capítulo será abordado o tema do treinamento, juntamente aos resultados obtidos após o treinamento do modelo de predição LSTM.

4.1 Treinamento

Para o treinamento do modelo, os seguintes parâmetros foram adotados:

- Número de épocas = 200
- Parâmetro para função loss sendo o *Mean Average Error* (Erro médio)
- Otimizador *Adam* disponível no pacote *Keras*
- *Learning rate* (Taxa de aprendizado) de 10^{-4}

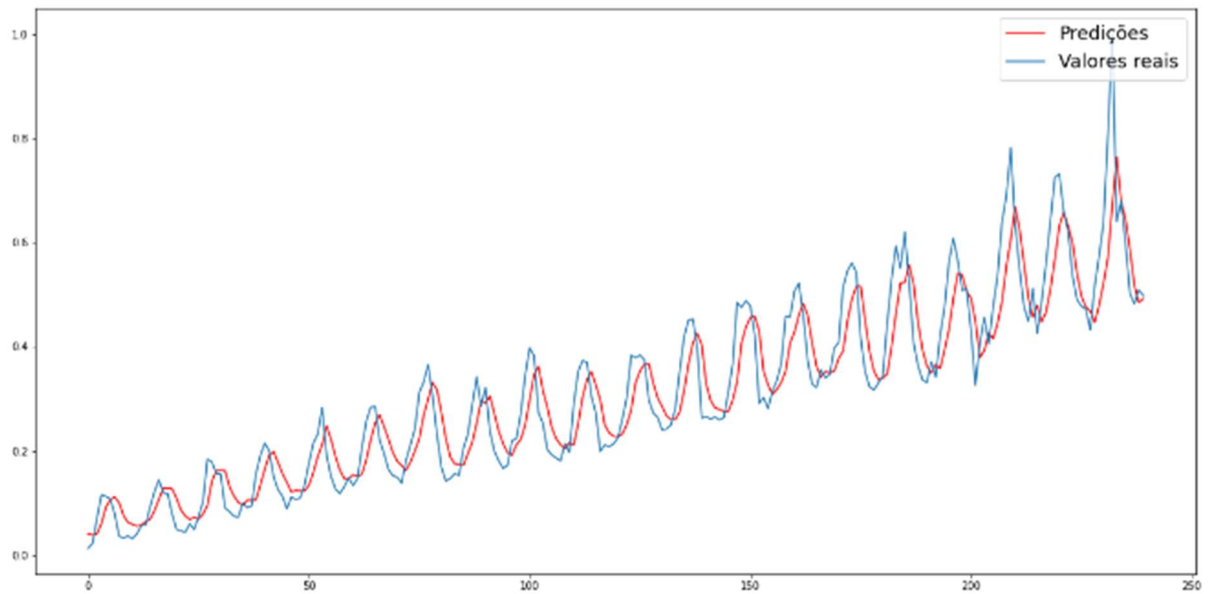
Além destes parâmetros, foram realizados 3 treinamentos com arquiteturas diferentes, sendo elas:

- M1_LSTM_64c_3h_32n_1out: Uma camada LSTM com 64 células, três *hidden layers* com 32 *nodes* e um *output layer* de dimensão 1
- M2_LSTM_128c_3h_64n_1out: Uma camada LSTM com 128 células, três *hidden layers* com 64 *nodes* e um *output layer* de dimensão 1
- M1_LSTM_256c_3h_128n_1out: Uma camada LSTM com 256 células, três *hidden layers* com 128 *nodes* e um *output layer* de dimensão 1

Para avaliar a performance destes diferentes modelos, nos próximos capítulos serão abordadas as métricas de avaliação e gráficos dos resultados obtidos.

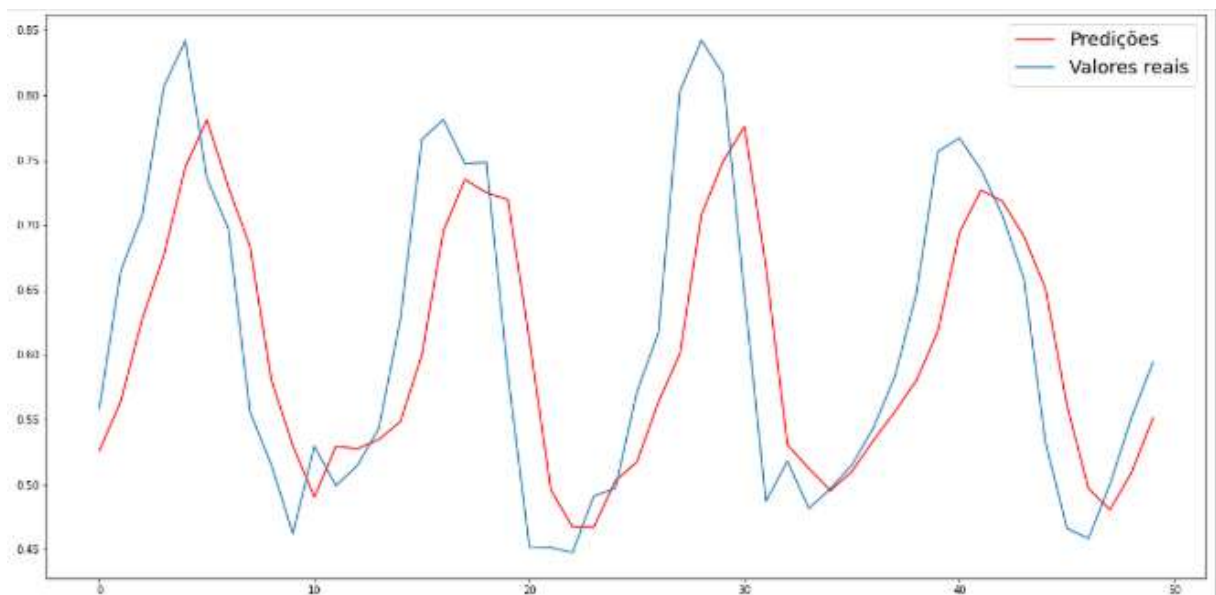
4.2 Resultados obtidos

Após o treinamento dos modelos, este será aplicado no *training set* e *validation set* para gerar as predições, como observado nas figuras 26, 27, 28, 29, 30, 31, 32, 33 e 34 a seguir, nas quais estão representados os diferentes modelos aplicados aos conjuntos de *training set*, *validation set* e *test set*. Na figura 26 abaixo está representado o desempenho do modelo 1.

Figura 26: Desempenho do modelo 1 no *training set*

Fonte: Do Autor (2022)

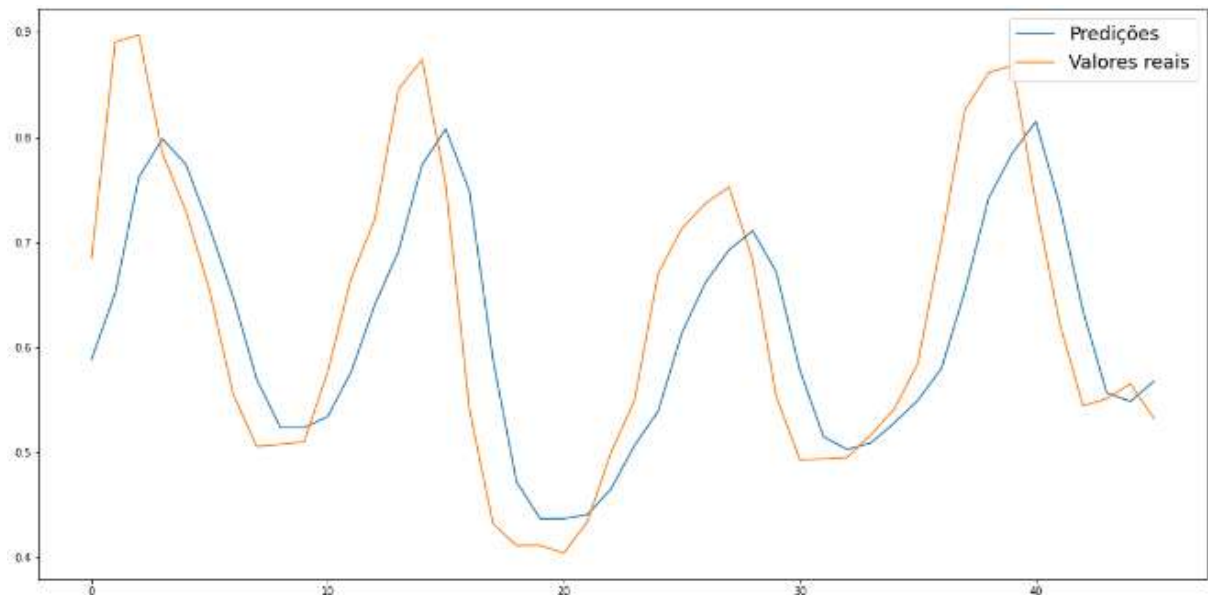
Pode-se perceber visualmente que o modelo não atinge as expectativas esperadas. Agora observando o comportamento no *validation set* na figura 27, não se percebe significativa melhora.

Figura 27: Desempenho do modelo 1 no *validation set*

Fonte: Do Autor (2022)

Por último, é reproduzido o gráfico de comparação de valores preditos pelo modelo com os valores reais da base de dados na figura abaixo.

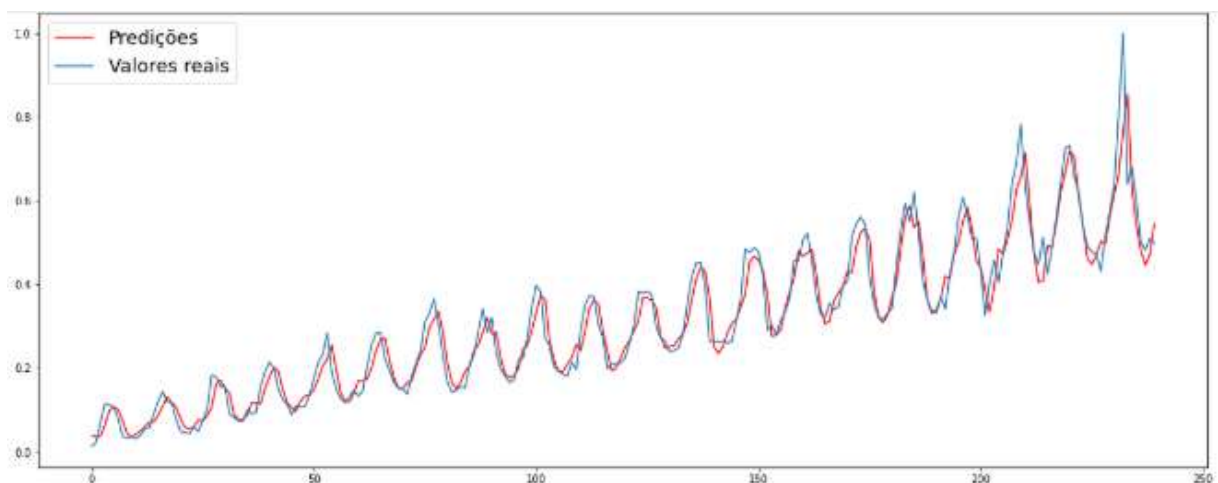
Figura 28: Desempenho do modelo 1 no *test set*



Fonte: Do Autor (2022)

Em relação ao modelo 2, foram obtidos os seguintes gráficos, na imagem 29, é possível perceber relativa melhora nos valores preditos.

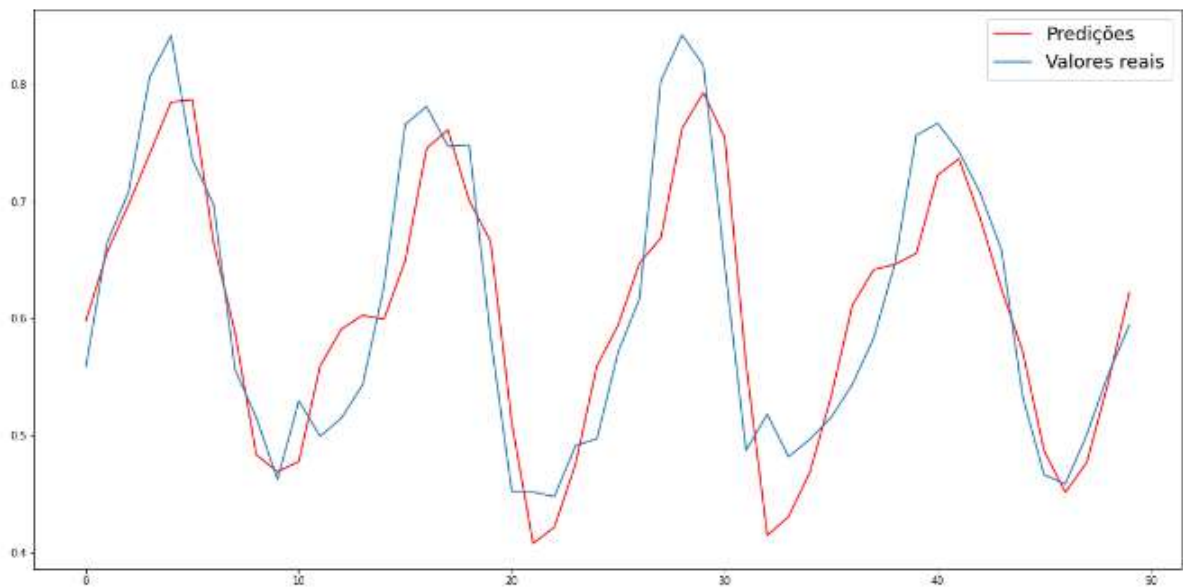
Figura 29: Desempenho do modelo 2 no *training set*



Fonte: Do Autor (2022)

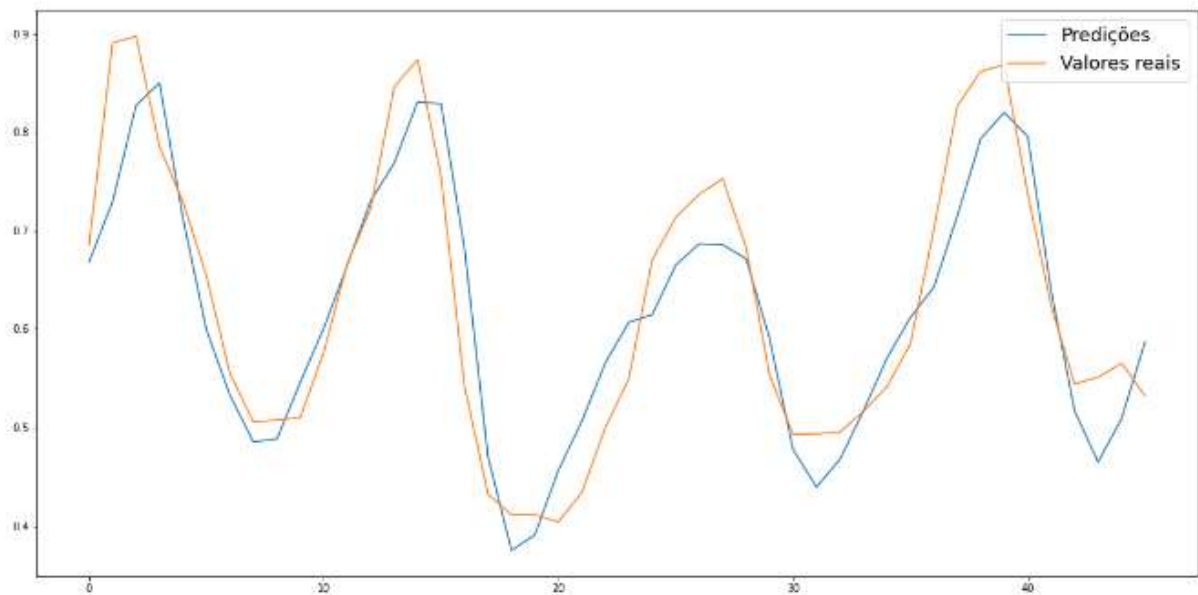
Também é possível perceber visualmente na figura abaixo, que os modelos 1 e 2 tem o desempenho bastante parecido no *validation set*.

Figura 30: Desempenho do modelo 2 no *validation set*



Fonte: Do Autor (2022)

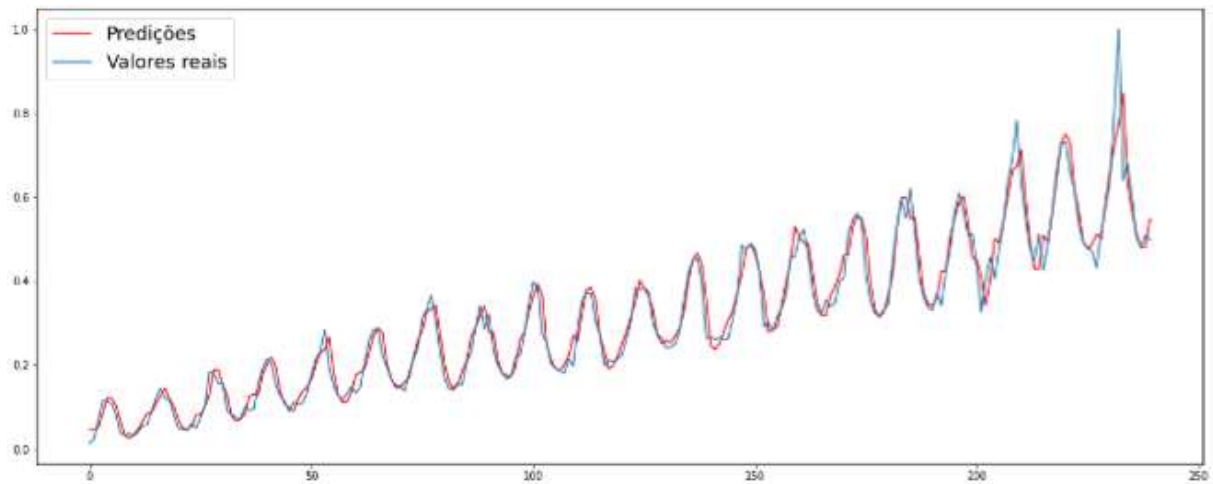
Figura 31: Desempenho do modelo 2 no *test set*



Fonte: Do Autor (2022)

O último modelo aplicado à base de dados teve como resultados os seguintes gráficos, iniciando pelo desempenho em relação ao *training set*. Visualmente se percebe melhoras significativas na comparação dos modelos.

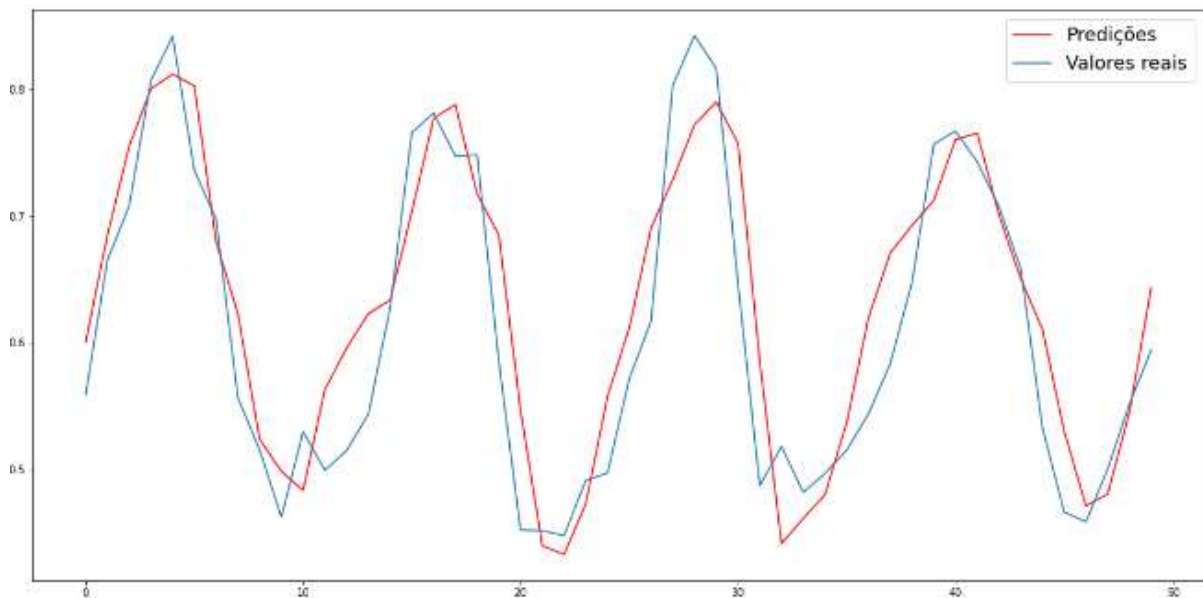
Figura 32: Desempenho do modelo 3 no *training set*



Fonte: Do Autor (2022)

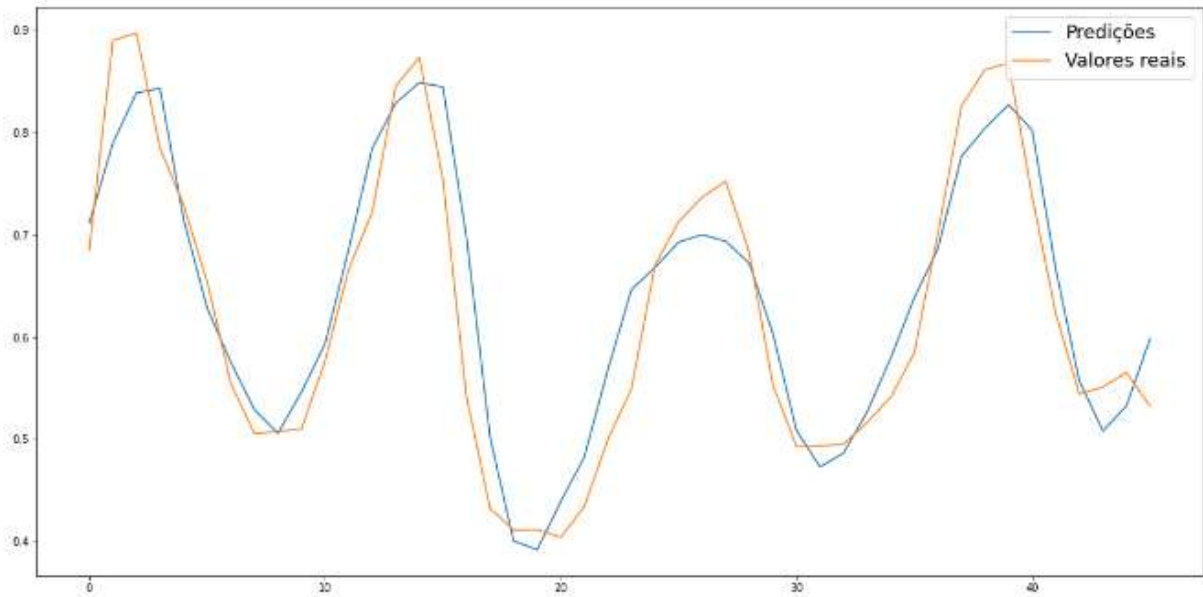
Nos gráficos 33 e 34, não é possível perceber melhoras ou pioras significativas em relação aos modelos anteriores. No próximo capítulo será abordada a avaliação destes resultados utilizando métricas específicas para predição de dados.

Figura 33: Desempenho do modelo 3 no *validation set*



Fonte: Do Autor (2022)

Figura 34: Desempenho do modelo 3 no *test set*



Fonte: Do Autor (2022)

4.3 Avaliação dos resultados

Para se verificar a avaliação dos resultados, será feito uma análise utilizando as métricas *MAE* (Erro médio), *MSE* (Erro quadrático médio) e *MAPE* (Erro médio percentual), que são definidos da seguinte maneira, sendo e_t o erro da predição, definido por $e_t = y_t - f_t$, com y_t sendo o valor verdadeiro e f_t o valor da predição.

$$\text{MAE: } \frac{1}{n} \sum_{t=1}^n |e_t| \quad (1)$$

$$\text{MSE: } \frac{1}{n} \sum_{t=1}^n e_t^2 \quad (2)$$

$$\text{MAPE: } \frac{1}{n} \sum_{t=1}^n \left| \frac{e_t}{y_t} \right| \times 100 \quad (3)$$

A performance das diferentes arquiteturas testadas do nosso modelo é apresentada na tabela 1.

Tabela 1: Performance do modelo

Método	Conjunto	MAE	MSE	MAPE
M1_LSTM_64c_3h_32n_1out	Treino	0.050	0.003	22.77%
	Validação	0.064	0.007	10.49%
	Teste	0.073	0.008	11.39%
M2_LSTM_128c_3h_64n_1out	Treino	0.031	0.002	13.22%
	Validação	0.048	0.003	7.44%
	Teste	0.053	0.003	7.66%
M1_LSTM_256c_3h_128n_1out	Treino	0.026	0.001	11.68%
	Validação	0.044	0.003	7.62%
	Teste	0.040	0.002	6.59%

Fonte: Do Autor

Com estes resultados é possível perceber que ao aumentar a quantidade de células LSTM, juntamente aos *nodes* do *hidden layer*, os resultados de *MAPE* diminuem, juntamente aos valores de *MAE* e *MSE*, fazendo o modelo 3 ser o mais adequado para a aplicação de predição.

5 CONSIDERAÇÕES FINAIS

Esse trabalho teve como objetivo desenvolver um modelo de predição se utilizando de métodos de aprendizado de máquina, mais especificamente redes neurais recorrentes utilizando a arquitetura LSTM para a predição de consumo de energia elétrica de classe comercial no município de Florianópolis a partir da obtenção de uma base de dados, tratamento, construção e aplicação de modelos utilizando-se da linguagem de programação Python e da biblioteca TensorFlow.

Foram construídos 3 modelos distintos com arquiteturas diferentes para aplicação e comparação, após isso foram feitas análises exploratórias e análise das métricas *MAE*, *MSE* E *MAPE* para avaliar o desempenho dos modelos tendo como resposta que o modelo com mais células LSTM e número de *nodes* nos *hidden layers* da rede neural foi o mais satisfatório na função de predição de dados.

Também se destaca o trabalho realizado para criação de um *Kaggle Notebook*, disponível no endereço <https://www.kaggle.com/code/pinducas/previs-o-de-consumo-de-energia-utilizando-lstm>, que tem como função distribuição do código desenvolvido e encorajamento na iniciação de projetos relacionados a predição de séries temporais utilizando redes neurais.

Para trabalhos futuros, a título de curiosidade e construção em cima do tema, é encorajado o uso de base de dados maiores, como também maior diversidade na arquitetura da rede neural, buscando resultados ainda mais satisfatórios.

REFERÊNCIAS

BURKOV, A. **The Hundred-Page Machine Learning Book**. 2019.

MORONEY, L. **AI and Machine Learning for Coders**. Gravenstein Highway North, Sebastopol, 2020.

MURPHY, K. P. **The Machine Learning: A Probabilistic Perspective**. Palo Alto, Califórnia: MIT Press, 2012.

GÉRON, A. **Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow** Gravenstein Highway North, Sebastopol, CA: O'Reilly Media, 2019.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence A Modern Approach**. 2ª edição. Upper Saddle River, New Jersey: Pearson, 2019.

BOX, G. **Time Series Analysis Forecasting and Control**. 5ª edição. Hoboken, New Jersey: John Wiley & Sons, Inc, 2016.

EHLERS, R. **Análise de séries temporais**.

Universidade Federal do Paraná, 2019. Disponível em: https://scholar.google.com/scholar_lookup?title=An%C3%A1lise%20de%20s%C3%A9ries%20temporais.%20Laborat%C3%B3rio%20de%20Estat%C3%ADstica%20e%20Geoinforma%C3%A7%C3%A3o&publication_year=2007&author=R.S.%20Ehlers.

Acesso em: 26 set. 2022.

HYNDMAN, R.J; ATHANASOPOULOS, G. **Forecasting: Principles and Practice**. 3ª edição, OTexts: Melbourne, Austrália. <https://otexts.com/fpp3/>. Acesso em: 26 set. 2022.

Banco Central - Sistema Gerenciador de Séries Temporais, 2022. Disponível em: <https://www3.bcb.gov.br/sgspub/consultarvalores/consultarValoresSeries.do?method=consultarGraficoPorId&hdOidSeriesSelecionadas=1403>. Acesso em: 10 nov. 2022.

MURPHY, K.P. **Probabilistic Machine Learning An Introduction**. 1ª edição. Cambridge, Massachusetts: The MIT Press, 2022.

ROSENBLATT, F. **THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN**, 1958. Disponível em:

<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=65a83117cbcc4e6eb7c6ac5be8e61195dc84b9fc>. Acesso em: 20 nov. 2022.

BISHOP, C.M. **Neural Networks for Pattern Recognition**. 1ª edição. Department of Computer Science and Applied Mathematics Aston University Birmingham, UK, 1995.

GRAVES, A. **Supervised Sequence Labelling with Recurrent Neural Networks**. Springer, 2012.

KAYTEZ, F.; TAPLAMACIOGLU, C.; CAM, E. &HARDALAC, F. **Forecasting electricity consumption: A comparison of regression analysis, neural networks and least squares support vector machines**. International Journal of Electrical Power & Energy Systems, 2015.

Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. **LONG SHORT-TERM MEMORY** IEEE Press, 1997.

Daniel L. Marino, Kasun Amarasinghe, Milos Manic. **Building Energy Load Forecasting using Deep Neural Networks** IEEE Press, 2016.

Ignacio Segovia-Dominguez, Zhiwei Zhen, Rishabh Wagh, Huikyo Lee, and Yulia R. Gel **TLife-LSTM: Forecasting Future COVID-19 Progression with Topological Signatures of Atmospheric Conditions**, The University of Texas at Dallas, Richardson, TX 75080, USA, 2021.

BARDELIN C. **Os efeitos do Racionamento de Energia Elétrica ocorrido no Brasil em 2001 e 2002 com ênfase no Consumo de Energia Elétrica**, Universidade de São Paulo 2004.

TensorFlow Previsão de Séries Temporais, 2022. Disponível em: https://www.tensorflow.org/tutorials/structured_data/time_series#setup. Acesso em: 10 nov. 2022.