Aryel Soares Loureiro

Previsão de Preços da Soja no Paraná: estudo comparativo entre modelos SARIMA e LSTM

Rio Grande, Rio Grande do Sul, Brasil

Dezembro, 2023

Aryel Soares Loureiro

Previsão de Preços da Soja no Paraná: estudo comparativo entre modelos SARIMA e LSTM

Trabalho de Conclusão de Curso, Matemática Aplicada Bacharelado, submetido por Aryel Soares Loureiro junto ao Instituto de Matemática, Estatística e Física da Universidade Federal do Rio Grande.

Universidade Federal do Rio Grande - FURG

Instituto de Matemática, Estatística e Física - IMEF

Curso de Matemática Aplicada Bacharelado

Orientador: Prof[a] Doutora Raquel da Fontoura Nicolette

Rio Grande, Rio Grande do Sul, Brasil

Dezembro, 2023

Aryel Soares Loureiro

Previsão de Preços da Soja no Paraná: estudo comparativo entre modelos SARIMA e LSTM

Trabalho de Conclusão de Curso, Matemática Aplicada Bacharelado, submetido por Aryel Soares Loureiro junto ao Instituto de Matemática, Estatística e Física da Universidade Federal do Rio Grande.

Trabalho aprovado em 19 de dezembro de 2023.

gov.br Documento assinado digitalmente
RAQUEL DA FONTOURA NICOLETTE
Data: 15/01/2024 16:58:18-0300
Verifique em https://validar.iti.gov.br

Prof[a] Doutora Raquel da Fontoura Nicolette
(Orientadora - FURG)

gov.br Documento assinado digitalmente
PAUL GERHARD KINAS
Data: 15/01/2024 17:52:01-0300
Verifique em https://validar.iti.gov.br

Prof. Doutor Paul G. Kinas
(IMEF - FURG)

gov.br Documento assinado digitalmente
ADILSON DA SILVA NUNES
Data: 19/01/2024 13:15:17-0300
Verifique em https://validar.iti.gov.br

Prof. Doutor Adilson da Silva Nunes
(IMEF - FURG)

gov.br Documento assinado digitalmente
RODRIGO DA ROCHA GONCALVES
Data: 16/01/2024 18:19:52-0300
Verifique em https://validar.iti.gov.br

Prof. Doutor Rodrigo da Rocha Gonçalves
ICEAC - FURG

Rio Grande, Rio Grande do Sul, Brasil
Dezembro, 2023

Este trabalho é dedicado à ciência de dados, um campo dinâmico que se destaca pela sua capacidade de extrair insights valiosos a partir de dados complexos. Em especial, busca-se honrar o compromisso com a excelência na análise preditiva, evidenciado pela comparação entre o modelo econométrico SARIMA e a sofisticada rede neural LSTM na previsão do preço da soja no estado do Paraná. Que esta dedicação inspire futuras investigações e contribuições inovadores no vasto universo da ciência de dados, impulsionando avanços significativos para compreensão e antecipação de padrões em fenômenos econômicos tão cruciais, que exercem influência sobre commodities de mercados agrícolas.

Agradecimentos

Agradeço minha família por me acolher durante toda a graduação.

À orientadora Raquel por concordar na proposta da monografia e me auxiliar nas correções durante o desenvolvimento do trabalho.

À FURG, em especial ao IMEF, pela experiência vivida no âmbito acadêmico.

À Data Science Academy (DSA) pelos cursos gratuitos de programação em Python e introdução à engenharia de dados que foram fundamentais na resolução do problema proposto.

Resumo

Brasil está entre os países que mais produzem insumos agrícolas no mundo. Dentre eles, a soja é a cultura mais exportada, tendo um impacto significativo no PIB (Produto Interno Bruto) brasileiro. Um dos estados que mais produzem soja é o Paraná, e o preço por uma saca em dólar varia conforme o tempo. A previsão do preço é importante para tomar decisões na produção desse insumo agrícola e analisar a presença de fatores externos, permitindo assim melhorar a administração do cultivo do insumo no estado e aumentar o ganho na exportação. Diante dessas informações, o foco desse trabalho foi prever o preço em dólar de uma saca de soja no estado do Paraná para o ano 2023 com base nos dados mensais de janeiro de 2012 à dezembro de 2022. Os dados do preço da soja foram obtidos do Centro de Estudos Avançados em Economia Aplicada (CEPEA) e foram analisados através da linguagem de programação Python no Jupyter Notebook IDE, transformando-os em uma série temporal. Para isso foram estimados o modelo estatístico SARIMA(0,1,1)(0,0,1) e uma rede neural LSTM com três camadas para determinar a melhor previsão possível com base em métricas de desempenho. Ambos modelos demonstraram queda do preço nos primeiros meses de 2023 e volatilidade abaixo de 2022, além de conseguirem previsões adequadas para janeiro e fevereiro de 2023. Ao comparar os modelos a rede neural se tornou mais adequada para previsão por apresentar melhores métricas de desempenho. Também foi realizada uma comparação entre os modelos com base na previsão parcial (primeiro semestre) em relação aos dados reais. Nesse caso a rede neural continuou mostrando ser mais adequada para previsão com a exceção de janeiro de 2023, no qual o modelo estatístico obteve melhor resultado.

Palavras-chaves: Soja, Séries Temporais, Econometria, Redes Neurais.

Abstract

Brazil is among the countries that produce the most agricultural inputs in the world. Among them, soybeans are the most exported crop, having a significant impact on the Brazilian Gross Domestic Product (GDP). One of the states that produces the most soybeans is Paraná, and the price per bushel in dollars varies over time. Price forecasting is important for making decisions in the production of this agricultural input and analyzing the presence of external factors, thus improving the management of crop cultivation in the state and increasing earnings from exports. In light of this information, the focus of this work was to predict the price in dollars of a bushel of soybeans in the state of Paraná for the year 2023 based on monthly data from January 2012 to December 2022. Soybean price data were obtained from the Center for Advanced Studies in Applied Economics (CEPEA) and were analyzed using the Python programming language in the Jupyter Notebook IDE, transforming them into a time series. For this purpose, the SARIMA(0,1,1)(0,0,1) statistic model and a three-layer LSTM neural network were fitted to determine the best possible forecast based on performance metrics. Both models showed a decrease in price in the first months of 2023, with an average close to 32 dollars and volatility lower than in 2022. When comparing the models, the neural network proved to be more suitable for forecasting due to its better performance metrics. A comparison was also made between the models based on the partial forecast (first half) in relation to the real data. In this case, the neural network continued to be more suitable for forecasting, with the exception of January 2023, in which the statistical model obtained better results.

Key-words: Soy, Time Series, Econometrics, Neural Networks.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

Adam Estimativa de Momento Adaptativo (*Adaptive Moment Estimation*)

ADF Dickey-Fuller Aumentado (*Augmented Dickey-Fuller*)

AIC Critério de Informação de Akaike (*Akaike Information Criterion*)

AR Modelo Autorregressivo (*Autoregressive*)

ARIMA Modelo Autorregressivo integrado de Médias Móveis (*Autoregressive Integrated Moving Average*)

ARMA Modelo Autorregressivo de Médias Móveis (*Autoregressive Moving Average*)

BIC Critério de Informação de Schwarz (*Bayesian Information Criterion*)

BPTT Retropropagação Através do Tempo (*Backpropagation Through Time*)

CEPEA Centro de Estudos Avançados em Economia Aplicada

CONAB Companhia Nacional do Abastecimento

EAM Erro Absoluto Médio

ETL Extrair, Transformar e Carregar (*Extract, Transform, Load*)

HQIC Critério de Informação de Hannan-Quinn (*Hannan-Quinn Information Criterion*)

IBGE Instituto Brasileiro de Geografia e Estatística

IDE Ambiente de Desenvolvimento Integrado (*Integrated Development Environment*)

LSTM Memória de Curto-Longo Prazo (*Long Short-Term Memory*)

MA Modelo de Médias Móveis (*Moving Average*)

PIB Produto Interno Bruto

REQM Raiz do Erro Quadrático Médio

RMSProp Propagação da Raiz Quadrática Média (*Root Mean Square Propagation*)

RNN Rede Neural Recorrente (*Recurrent Neural Networks*)

SARIMA Modelo Autorregressivo integrado de Médias Móveis com Sazonalidade (*Seasonal Autoregressive Integrated Moving Average*)

SGD Gradiente Descendente Estocástico (*Stochastic Gradient Descent*)

Sumário

1 Introdução

A soja é uma cultura agrícola de grande importância econômica e comercial para o Brasil, sendo um dos principais insumos cultivados em grande escala em diversas regiões, especialmente no Centro-Oeste e Sul. Ela é importante devida à alta demanda mundial, tanto para a produção de alimentos quanto à produção de biocombustível. Além disso, a soja é relativamente resistente a condições climáticas adversas, o que a torna uma opção interessante para os agricultores brasileiros, que muitas vezes enfrentam secas e outros problemas climáticos. Ela também possui boa adaptação a diferentes tipos de solo e é bastante produtiva, o que torna uma produção rentável para os agricultores (HIRAKURI; LAZZAROTTO, 2014).

A importância econômica da soja no Brasil é atribuída a diversos fatores. O país é atualmente o maior exportador de soja no mundo, o que implica em uma parte significativa do Produto Interno Bruto (PIB) brasileiro, sendo responsável por gerar empregos e renda em todo o país. Outros fatores incluem demanda mundial, clima favorável, infraestrutura e tecnologia agrícola.

A região sul do Brasil é uma das que mais produz soja em seus estados, dentre eles o Paraná. Através das informações da Companhia Nacional do Abastecimento (CONAB), esse estado representa o segundo maior produtor de soja do país, respondendo por 14% da safra brasileira em 2022. Atrás apenas do Mato Grosso do Sul, Paraná possui grande impacto na balança econômica agrícola do Brasil.

Neste contexto, é interessante analisar como o preço da soja se comporta no Paraná. Como o preço é influenciado pela inflação, uma possível alternativa é analisar o preço em dólar ao invés da moeda local sem se preocupar com ajuste monetário (MANKIW, 2017). Nesse caso, o preço da soja é influenciado pela inflação dos Estados Unidos (EUA), podendo acarretar em uma possível queda futura no preço da soja produzida nacionalmente levando em consideração a inflação americana atual.

O comportamento do preço pode ser analisado através da análise exploratória de dados com o objetivo de obter insights e entender melhor as características dos dados no início de cada mês, em conformidade com GRUS (2018), pois a análise exploratória de dados é de suma importância para a ciência de dados. É possível interpretar os dados como uma série temporal devida à variação temporal do preço, exigindo também uma análise comportamental da série.

Série temporal é uma coleção de dados observados em intervalos de tempo regulares ao longo de um período contínuo (BOX; JENKINS, 1971). Ela pode ser obtida através de um conjunto de dados que são modificados temporalmente, como é o caso do preço da

soja. A série temporal possui propriedades como tendência, sazonalidade, ciclos, ruído, dependência temporal e estacionariedade. O conhecimento dessas propriedades é de suma importância para interpretar corretamente os dados. A previsão de preço é uma das finalidades de uma série temporal, de acordo com MORETTIN e TOLOI (2018), sendo útil na tomada de decisões de curto e longo prazo. É possível analisar o comportamento dos dados de modo a extrair o máximo de informações úteis utilizando modelos univariados ou multivariados.

Modelos estatísticos, também chamados de econométricos quando aplicados a dados no contexto econômico, podem ser utilizados para analisar efeitos de fatores externos sobre a produção de soja no Brasil, como preços governamentais, políticas internacionais e mudanças no clima. Esta análise pode ajudar a identificar oportunidades e riscos para a produção agrícola, permitindo que aumente a assertividade de decisões de agricultores e outros agentes do setor. De modo geral, a análise de séries temporais aplicada à produção de soja no Paraná pode fornecer informações importantes para a gestão macroeconômica do país (ZIEGLER et. al., 2020).

Para BOX e JENKINS (1971), um dos modelos estatísticos mais populares é o ARIMA (Autoregressive Integrated Moving Average), sendo uma extensão de um modelo mais simples ARMA (Autoregressive Moving Average) com ordem de integração caso a série temporal não seja estacionária em nível. Esse modelo é bastante popular na previsão de diversas séries temporais. Além de modelos estatísticos, existem modelos obtidos de redes neurais que também possuem a mesma proposta de previsão de séries temporais, mas com uma abordagem diferente.

As redes neurais são uma técnica de aprendizado de máquina que podem ser aplicadas na previsão de séries temporais com base nos dados históricos, assim como variáveis externas que podem impactar no resultado final. Uma das principais vantagens de redes neurais é a capacidade de identificar dados complexos e de grande volume, permitindo assim trabalhar com grande quantidade de dados e garantindo previsões mais precisas e confiáveis (GREFF et al., 2016).

Uma das classes de redes neurais artificiais projetadas para lidar com dados sequenciais é chamada Rede Neural Recorrente (RNN), capaz de modelar dependências em dados sequenciais. Existe uma extensão para essa classe chamada Memória de Curto-Longo Prazo (LSTM) que serve como um modelo de aprendizado profundo que também é usado para lidar com dados sequenciais, como na previsão dos preços do mercado de energia elétrica (SANTOS, 2019), e na predição de palavras em uma sequência de texto (GREFF et al., 2016). Assim como modelos estatísticos, modelos de rede neural também são influenciados por propriedades de séries temporais, podendo impactar na previsão dos dados. Sendo assim, comparar esses modelos que possuem diferentes abordagens com a mesma proposta é interessante para a busca do melhor ajuste, bem como predições mais

próximas aos valores observados.

1.1 Objetivos

Objetivos Gerais

Ajustar a série de dados do preço da soja a um modelo de séries temporais e uma rede neural LSTM.

Objetivos Específicos

1. Analisar o comportamento e as características da série temporal.
2. Determinar o melhor modelo SARIMA para previsão do preço da soja.
3. Desenvolver uma rede neural para previsão do preço da soja.
4. Avaliar comparativamente as duas modelagens para previsão do preço da soja.

1.2 Estrutura da Monografia

A monografia está estruturada em seis capítulos, conforme segue.

Capítulo 1 – Introdução: Apresenta todos os aspectos inicias do trabalho, assim como sua motivação, justificativa e propósito na previsão do preço da soja.

Capítulo 2 – Fundamentação Econômica: Introduz alguns conceitos e fatores econômicos que podem impactar no preço da soja influenciando na projeção de curto prazo.

Capítulo 3 - Fundamentação Matemática: Apresenta as definições e conceitos matemáticos que são necessários para este estudo como análise exploratória de dados, conceitos de séries temporais e modelos estatísticos.

Capítulo 4 – Fundamentação Computacional: Aborda conceitos computacionais usados em séries temporais.

Capítulo 5 – Aplicação e avaliação dos modelos: Desenvolve toda a projeção do preço da soja com base em duas modelagens diferentes.

Capítulo 6 – Considerações finais.

2 Fundamentação Econômica

Esse capítulo introduz conceitos econômicos importantes para o entendimento da evolução dos preços da soja ao longo dos anos. Saber o contexto dos dados é fundamental para analisar melhor o comportamento deles ao longo do tempo.

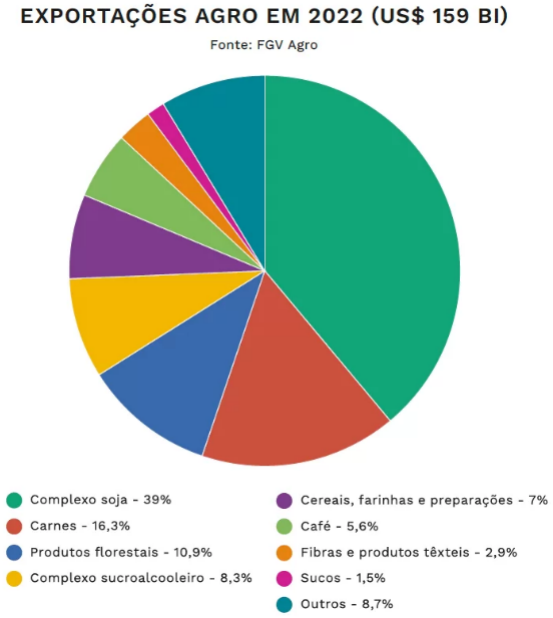
2.1 Análise do Mercado de Commodities Agrícolas

O Brasil é um dos maiores produtores e exportadores mundiais de commodities agrícolas. A diversidade de culturas se deve à extensa área de terras cultiváveis e às condições climáticas favoráveis.

2.1.1 Exportação de Commodities

O país é um grande exportador de commodities agrícolas, fornecendo alimentos e matérias-primas para outros países em todo o mundo. Dentre eles, a China é um importante parceiro comercial que importa grande quantidade de soja brasileira. A exportação é importante para o país porque impacta positivamente no PIB brasileiro. Uma das contribuições positivas para o PIB é o superávit comercial que ocorre quando o total de exportações é superior ao de importações, isto é, maior geração de riqueza para o país (MANKIW, 2017).

Figura 1 – Exportações das commodities agrícolas em 2022



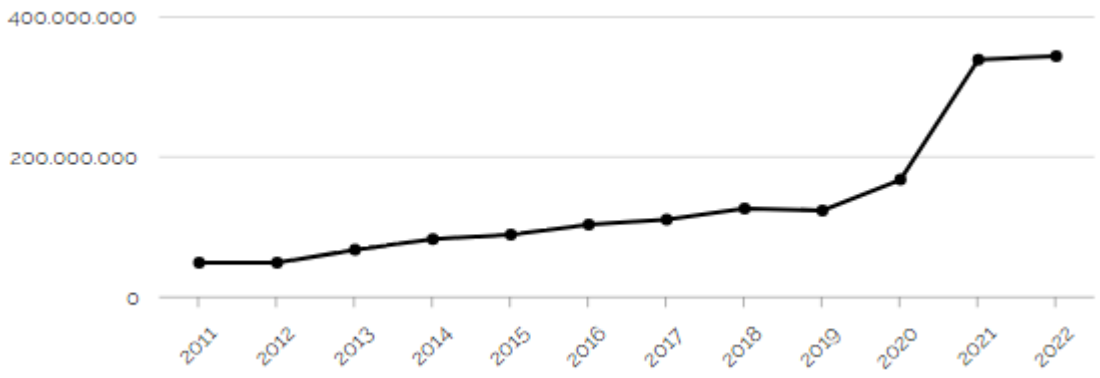
Fonte: CENTRO DE ESTUDOS DO AGRONEGÓCIO, 2023

Brasil obteve cerca de 159 bilhões de dólares com exportações de commodities agrícolas em 2022. As três categorias mais vendidas foram complexo soja, carnes e produtos florestais. Note que a soma de vendas das carnes e produtos florestais não chega próximo do número de vendas do complexo soja. Isso é um dos indicativos de que a soja é a cultura mais exportada do país. O complexo soja é um conjunto de produtos e atividades relacionadas à soja, sejam eles cultivo, processamento e outros, desempenhando um papel importante na segurança alimentar, produção de alimentos para animais e indústria de alimentos no geral.

2.1.2 Produção da Soja

O setor agrícola brasileiro tem adotado tecnologias modernas, como cultivos geneticamente modificados e práticas de agricultura de precisão, para aumentar a produtividade e a eficiência de culturas. Nesse cenário a produção de soja no Brasil cresceu bastante nos últimos anos. É possível analisar o indicador de produção anual de soja no Brasil através do Instituto Brasileiro de Geografia e Estatística (IBGE), abrangendo grandes regiões, unidades da federação, mesorregiões, microrregiões e municípios.

Figura 2 – Evolução anual da soja no Brasil



Fonte: PESQUISA AGRÍCOLA MUNICIPAL, 2023

A produção de soja no Brasil aumenta a cada ano de acordo com a Figura 2. Houve uma pequena queda no ano de 2019 e um aumento significativo em 2021. Esse aumento pode estar relacionado com a valorização da soja impulsionada pela pandemia de coronavirus, ocasionando em aumento do preço de uma safra em dólar. Pode ser mais relevante trabalhar com o preço da soja em dólar para a análise e previsão de preços, uma vez que a moeda brasileira não é tão relevante no mercado internacional quanto à americana. O impacto de variáveis externas como inflação e taxa de câmbio é que, em termos monetários, é necessário deflacionar a série.

2.1.3 Paraná

O Paraná é um estado localizado no sul do Brasil, conhecido por sua significativa contribuição para o setor agrícola e pelo desenvolvimento econômico. Sua história remonta à colonização europeia, com a presença inicial de portugueses, espanhóis e, posteriormente, imigrantes de diversas origens, como italianos, alemães e poloneses (TEXUGO, 2011). As diferentes regiões no estado do Paraná possuem relevância em vários aspectos, incluindo econômico, social, cultural e geográfico.

Figura 3 – Mapa das regiões do estado

Fonte: TEXUGO, 2011

No contexto econômico, o Paraná destaca-se como um dos principais produtores agrícolas do país. Suas condições climáticas favoráveis, aliadas à tecnologia agrícola moderna, resultaram em uma produção expressiva de commodities agrícolas, como soja, milho, trigo, entre outros. A diversificação da produção agrícola contribuiu para a resiliência econômica do estado diante das oscilações nos mercados internacionais (HIRAKURI; LAZZAROTTO, 2014).

A importância do Paraná para o mercado de soja é evidenciada pelo papel significativo que desempenha na produção nacional, influenciando os preços e as exportações do Brasil. A competitividade do estado no cenário internacional contribui para o fortalecimento da economia brasileira, uma vez que as exportações da soja desempenham um papel crucial no comércio exterior do país. Nesse contexto, uma previsão considerada adequada para o preço da soja no Paraná é de duas casas decimais com base na natureza dos dados, com erro abaixo da amplitude do intervalo de confiança para cada modelo.

2.2 Fenômenos Econômicos

O preço da soja pode ser influenciado por diversos fatores macroeconômicos como inflação, taxa de câmbio, política monetária, comércio global, clima e diversos outros. Alguns eventos atuais que podem afetar o preço da soja em dólar são inflação, pandemia, guerra e clima.

Inflação

De acordo com MANKIW (2017), a inflação é um fenômeno econômico causado pelo aumento na oferta da moeda não acompanhado por aumento proporcional de bens e serviços. Essa causa é tipicamente relacionada à impressão da moeda local de um país, em outras palavras, o governo imprimindo dinheiro para financiar suas despesas e, por consequência, prejudicando o valor monetário da moeda local. A inflação é um dos grandes eventos globais que afetam as cadeias produtivas, elevando custos como transporte e matérias primas.

Pandemia de COVID-19

Durante a pandemia do vírus SARS-CoV-2, chamado de COVID-19, governos imprimiram dinheiro desenfreadamente para arcar com os gastos públicos em combate ao vírus. Esse evento causou inflação alta em diversos países e, por consequência, impactou negativamente a economia global, causando aumento no preço de bens e serviços devido à desvalorização das moedas de troca. O preço da soja pode ter sofrido impacto da inflação e subido significativamente durante esse evento.

Guerra entre Rússia e Ucrânia

O mercado de commodities agrícolas pode influenciar o preço da soja. Em tempos de guerra e tensão geopolítica, os países envolvidos podem impor restrições (embargos) à exportação de produtos agrícolas, dentre eles a soja. Flutuações cambiais e mudanças na rota de transporte também são fatores que podem ocasionar em aumento de preço.

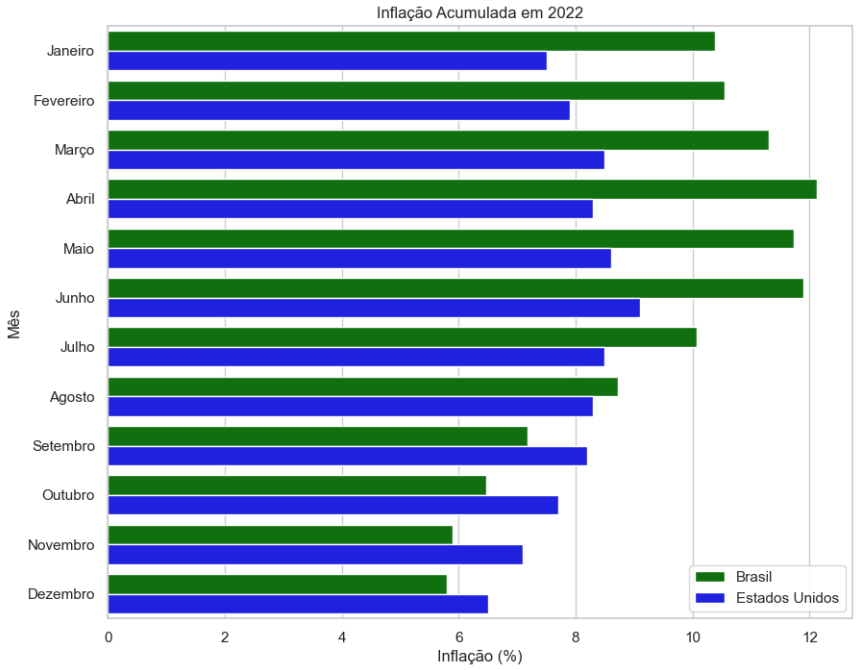
Mudanças Climáticas

A produção de soja pode sofrer impacto significativo de acordo com o clima da região. Se o clima estiver instável a produção poderá diminuir com o tempo, influenciando no aumento da demanda por alimentos. Como a soja é uma cultura sensível às mudanças climáticas, o preço pode estar correlacionado com a produção dessa cultura que é afetado por vários fatores como volatilidade dos mercados globais e das políticas e regulamentações ambientais (ZIEGLER et. al., 2020).

2.3 Inflação Americana

A inflação no Brasil e Estados Unidos podem diferir em vários aspectos. Uma boa razão para trabalhar com preços em dólar ao invés da moeda local (real) é a natureza internacional e global das commodities, juntamente com a referência internacional que o dólar desempenha no mercado. O mercado global da soja é influenciado por fatores econômicos e climáticos em diversos países. O uso do dólar ajuda a contextualizar os preços da soja em um cenário internacional mais amplo, incluindo a influência de fatores globais na oferta e demanda. A inflação americana tende a ser mais baixa do que a brasileira com base nas observações ao longo dos anos (MANKIW, 2017).

Figura 4 – Inflação acumulada em 2022



Fonte: Elaborado pelo Autor

Ao analisar a Figura 4 que representa o índice de Preços ao Consumidor Amplo em relação ao Brasil e o índice de Preços ao Consumidor em relação aos Estados Unidos, cujos dados foram obtidos do site Investing.com, é observável que a inflação brasileira é maior que a americana no começo de 2022, porém tende a cair até o final do ano. Isso pode ser um indicativo que o governo anterior conseguiu manusear a inflação durante a pandemia em comparação com o governo americano. No cenário americano o dólar está sofrendo de inflação elevada desde a pandemia, impactando na economia global por ser uma das principais moedas do mundo. Evidências empíricas indicam que o preço da soja é influenciado pela inflação, em especial a inflação americana quando se trabalha com o valor monetário em dólar.

3 Fundamentação Matemática

Esse capítulo traz um breve resumo da fundamentação matemática em relação ao modelo estatístico e rede neural, demonstrando os cálculos mais relevantes durante a resolução do problema.

3.1 Análise Exploratória de Dados

A análise exploratória de dados é uma abordagem fundamental na estatística e na ciência de dados que visa resumir, visualizar e entender os dados antes de aplicar técnicas estatísticas mais avançadas ou modelos preditivos (McKINNEY, 2018). Ela é uma etapa crítica no processo de análise de dados, pois ajuda a orientar etapas subsequentes, como modelagem estatística, aprendizado de máquina ou outras análises quantitativas. A estatística está dividida em três áreas que se complementam, cada uma com sua própria finalidade.

- Estatística Descritiva: Resumir e apresentar os dados de uma maneira que seja informativa e fácil de entender, ajudando na análise exploratória e na comunicação de informações sobre os dados.
- Probabilidade: Modelar eventos incertos e auxiliar na tomada de decisões baseadas em informações probabilísticas.
- Estatística Inferencial: Fazer inferências sobre uma população com base em uma amostra, avaliar a significância de resultados e testar hipóteses.

Em relação à estatística descritiva, ela envolve o uso de várias medidas estatísticas, gráficos e tabelas para resumir e descrever as principais características de um conjunto de dados, incluindo a centralidade (média, mediana, moda), a dispersão (variância, desvio padrão), a posição relativa (amplitude, quartil), a forma da distribuição e outras propriedades dos dados (McKINNEY, 2018).

3.1.1 Tendência Central e Dispersão

Tendência central é a noção de onde os dados estão centrados e a dispersão se refere à medida de espalhamento dos dados (GRUS, 2018).

Média Aritmética

A média aritmética é a soma de todos os valores em um conjunto de dados dividida pelo número de valores (GRUS, 2018). É a medida de tendência central mais comumente usada na estatística.

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.1}$$

Onde:

x_i - Dados da amostra;

n - Tamanho da amostra.

Mediana

A mediana é o valor do meio de um conjunto de dados quando eles estão organizados em ordem crescente ou decrescente (GRUS, 2018). Isso significa que metade dos valores estão abaixo da mediana e metade estão acima dela. Para calcular a mediana, os dados devem ser primeiramente ordenados. Se o número de observações for ímpar, a mediana é o valor do meio. Caso contrário, isto é, se for par, a mediana é a média dos valores do meio.

Moda

A moda é o valor(es) que ocorre(m) com maior frequência em um conjunto de dados (GRUS, 2018). Ela é determinada contando a frequência de cada valor nos dados e identificando o(s) valor(es) com a maior frequência.

Variância

A variância é uma medida estatística que quantifica a dispersão ou variabilidade dos valores em um conjunto de dados (MORETTIN; TOLOI, 2018). Ela mede o grau em que os valores individuais em um conjunto de dados se afastam da média aritmética desses valores.

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x}_i)^2 \tag{3.2}$$

Onde x_i são os dados da amostra, $\overline{x}$ a média amostral e n o tamanho da amostra.

Desvio Padrão

O desvio padrão é a raiz quadrada da variância (MORETTIN; TOLOI, 2018). O desvio padrão pode ser usado para avaliar a variabilidade (volatilidade) nos preços de produtos e bens de consumo ao longo do tempo. Isso é importante para entender a estabilidade dos preços e seu impacto na inflação e no poder de compra dos consumidores.

3.1.2 Posição Relativa

A posição relativa refere-se à localização de um valor específico dentro de um conjunto de dados em relação a outros valores (GRUS, 2018). É uma maneira de entender como um valor se compara a outros valores no mesmo conjunto de dados.

Amplitude

A amplitude é uma medida de posição relativa que representa a diferença entre o maior e menor valor de um conjunto de dados (GRUS, 2018). Ela fornece uma indicação de quão amplamente os valores estão distribuídos no conjunto de dados. Quanto maior for a amplitude, maior a variação dos valores.

$$AT = x_{max} - x_{min} \tag{3.3}$$

Quartis

Os quartis são medidas estatísticas que dividem um conjunto de dados ordenados em quatro partes iguais, representando, assim, os valores que dividem os dados em quartos (MORETTIN; TOLOI, 2018). Eles são úteis para entender a distribuição e a dispersão dos dados. Os três quartis mais comuns são Q_1, Q_2 e Q_3.

- O primeiro quartil (Q_1) é definido como o número intermediário entre o menor número e a mediana do conjunto de dados. Também é conhecido como quartil inferior, pois 25% dos dados estão abaixo desse ponto.
- O segundo quartil (Q_2) é a mediana de um conjunto de dados. Sendo assim, 50% dos dados estão abaixo desse ponto.
- O terceiro quartil (Q_3) é o valor intermediário entre o maior número e a mediana do conjunto de dados. É conhecido como quartil superior, pois 75% dos dados estão abaixo desse ponto.

Quando a cardinalidade (tamanho) do conjunto de dados é ímpar, a fórmula para calcular os quartis pode ser definida como:

$$Q_i = x_{\frac{n+1}{2}}, i = 1, 2, 3 \tag{3.4}$$

Caso a cardinalidade seja par, então a fórmula é ajustada para:

$$Q_i = \frac{x_{min(\frac{i(n+1)}{4})} + x_{max(\frac{i(n+1)}{4})}}{2}, i = 1, 2, 3 \tag{3.5}$$

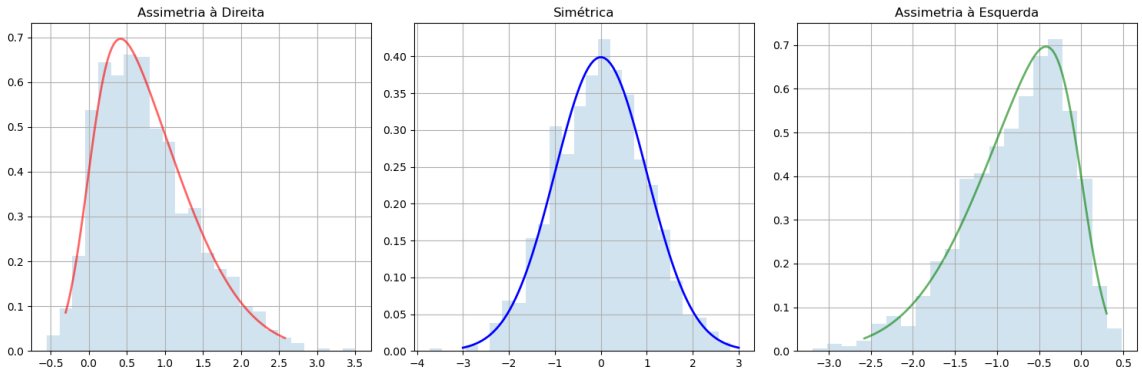
3.1.3 Assimetria e Curtose

Assimetria e curtose são medidas de forma da distribuição de dados. Elas fornecem informações sobre a forma de distribuição de probabilidade de um conjunto de dados, ou seja, como os valores estão distribuídos em relação à média e à variabilidade (MORETTIN; TOLOI, 2018). Os coeficientes analisados são baseados no teste de Jarque-Bera.

Coeficiente de Assimetria

O coeficiente de assimetria mede a falta de simetria em uma distribuição (MORETTIN; TOLOI, 2018). Uma distribuição simétrica tem uma assimetria igual a zero, o que significa que metade da esquerda da distribuição é um espelho da metade direita. Quando a assimetria é positiva, a cauda direita é mais longa do que a cauda esquerda, e a distribuição é considerada positivamente assimétrica. Quando a assimetria é negativa, a cauda esquerda é mais longa que a cauda direita, e a distribuição é considerada negativamente assimétrica.

Figura 5 – Simetria de distribuição



Fonte: Elaborado pelo autor

As medidas de tendência central estão relacionadas com a simetria de distribuição. De acordo com a Figura 5, a média (M_e), mediana (M_d) e moda (M_o) seguem uma relação de equivalência da seguinte forma:

- Assimétrica à direita: $M_o \leq M_d \leq M_e$.
- Simétrica: $M_e = M_o = M_e$.
- Assimétrica à esquerda: $M_e \leq M_d \leq M_o$.

O coeficiente de assimetria é dado por:

$$A_s = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{(x_i-\overline{x})}{s}\right)^3}{\left(\frac{1}{n}\sum_{i=1}^{n}\left(\frac{(x_i-\overline{x})}{s}\right)^2\right)^{3/2}} \tag{3.6}$$

Onde:

x_i - Dados da amostra;

$\overline{x}$ - Média amostral;

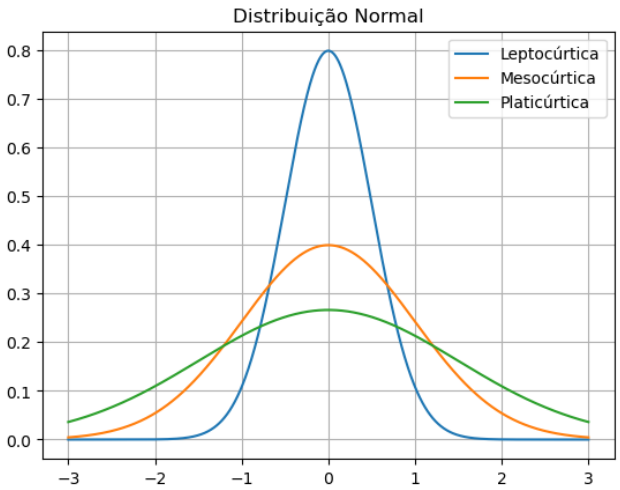
s - Desvio padrão amostral;

n - Tamanho da amostra.

Coeficiente de Curtose

O coeficiente de curtose é uma medida estatística que quantifica a forma da distribuição dos dados, em particular, o grau de achatamento ou picos na distribuição em comparação com uma distribuição normal (MORETTIN; TOLOI, 2018). Um valor de curtose maior que 3 indica uma distribuição mais pontiaguda (leptocúrtica), enquanto um valor menor que 3 indica uma distribuição mais achatada (platicúrtica) em relação à distribuição normal, que tem uma curtose de 3 (mesocúrtica).

Figura 6 – Classificação do coeficiente de curtose



Fonte: Elaborado pelo autor

O coeficiente de curtose é dado por:

$$K = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{(x_i-\overline{x})}{s}\right)^4}{\left(\frac{1}{n}\sum_{i=1}^{n}\left(\frac{(x_i-\overline{x})}{s}\right)^2\right)^2} \tag{3.7}$$

Onde:

x_i - Dados da amostra;

$\overline{x}$ - Média amostral;

s - Desvio padrão amostral;

n - Tamanho da amostra.

3.2 Séries Temporais

Uma série temporal é uma sequência de dados observados ao longo do tempo (BOX; JENKINS, 1971). Esses dados podem representar medições, observações ou eventos que ocorrem em momentos distintos e são registrados em ordem cronológica. As técnicas de séries temporais incluem métodos estatísticos como modelos de regressão, redes neurais, entre outros. Essas análises são usadas para entender o comportamento temporal dos dados e fazer previsões com base em observações passadas. A lei de formação para séries temporais geralmente envolve componentes determinísticos e/ou estocásticos que descrevem como a série temporal é gerada e evolui ao longo do tempo. Essas séries temporais podem ser compostas por duas partes principais:

- Componente Determinístico: A parte determinística da série temporal segue um padrão específico que pode ser modelado de maneira previsível. Exemplos de componentes determinísticos incluem tendência, sazonalidade e ciclos.
- Componente Estocástico: A parte estocástica da série temporal é aleatória e não segue um padrão previsível. Esse componente representa a incerteza e a variabilidade inerentes à série temporal. Pode incluir ruído aleatório, perturbações não explicadas por componentes determinísticos e outros fatores imprevisíveis.

Um exemplo de série temporal com componente aleatório é:

$$\Delta^1 Z_t = -\theta\epsilon_{t-1} - \Theta\epsilon_{t-1S} + \epsilon_t \tag{3.8}$$

Onde:

$\Delta^1 Z_t$ - Série temporal na primeira diferença;

θ - Coeficiente de média móvel;

Θ - Coeficiente de média móvel sazonal;

ϵ_t - Resíduo.

A primeira parte da série temporal $-\theta\epsilon_{t-1} - \Theta\epsilon_{t-1}$ se refere ao componente determinístico da série enquanto que a segunda parte ϵ_t o componente estocástico.

3.3 Processo Estocástico

Processo estocástico é um conceito fundamental na teoria das probabilidades e na estatística, que descreve a evolução de variáveis aleatórias ao longo do tempo (BOX; JENKINS, 1971). É uma generalização de uma série temporal que incorpora incerteza ou aleatoriedade na evolução dos dados. Os processos estocásticos são frequentemente usados para modelar sistemas que evoluem ao longo do tempo, nos quais as observações em momentos diferentes são influenciadas por fatores estocásticos.

Definição

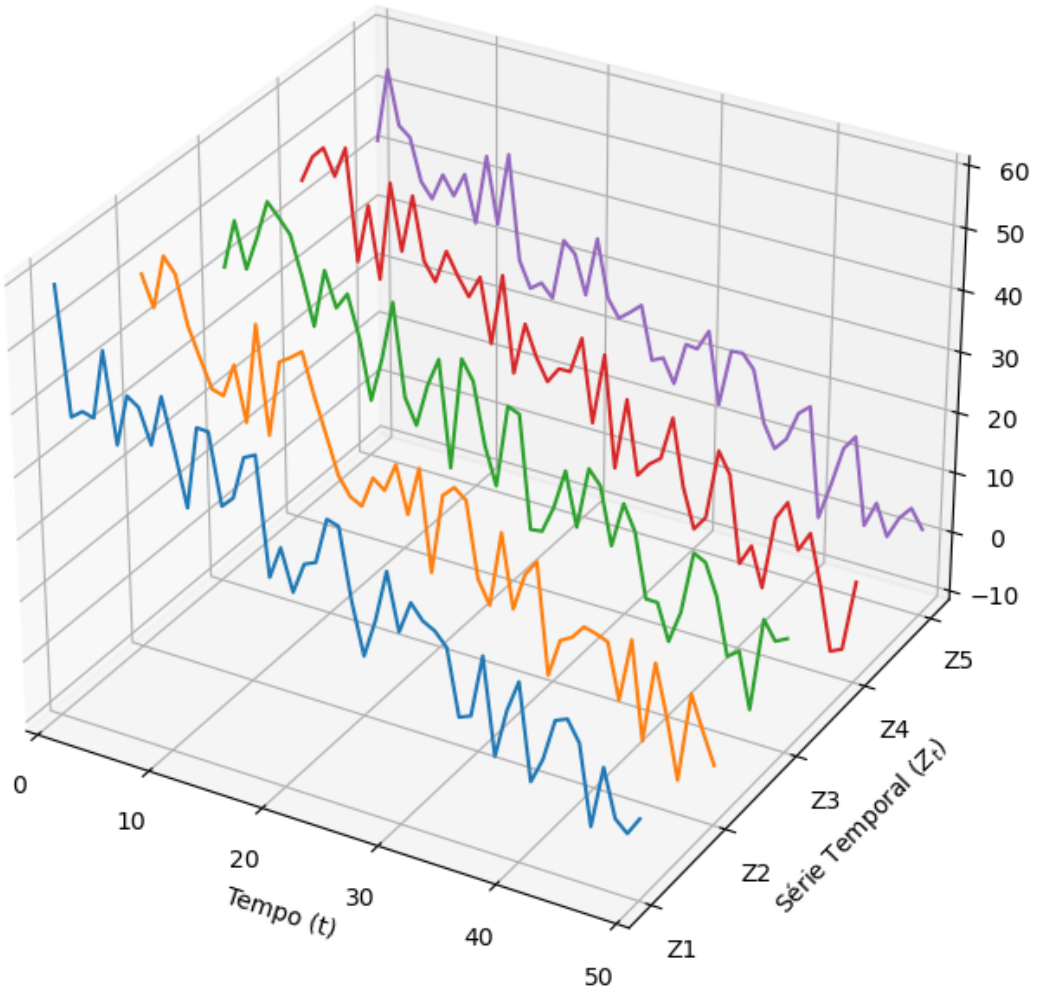
De acordo com BOX e JENKINS (1971), um processo estocástico é uma família de variáveis aleatórias $\{X_t, t \in T\}$ definidas em um espaço de probabilidade comum $\{\Omega, \mathcal{F}, P\}$ onde:

Ω - Conjunto de resultados possíveis. Representa o espaço de todos os resultados possíveis em um experimento ou fenômeno aleatório. É o conjunto universal que abrange todas as possíveis observações ou resultados de interesse. Cada resultado específico é um elemento de Ω;

$\mathcal{F}$ - σ-Álgebra de eventos, que é um conjunto de subconjuntos de ω. É uma família de conjuntos que contém eventos para medir ou analisar no espaço amostral;

P - Medida de probabilidade que atribui probabilidades a eventos em F. É uma medida que descreve quão provável é a ocorrência de eventos em $\mathcal{F}$.

Figura 7 – Exemplo de processo estocástico



Fonte: Elaborado pelo autor

As funções plotadas na Figura 7 representam uma série temporal cuja parte determinística é linearmente decrescente ao longo do tempo e a parte estocástica segue uma distribuição uniforme. É observável que os dados da série temporal são diferentes cada vez que o gráfico dessa série é gerado por causa do processo estocástico, implicando em oscilações aleatórias ao longo do tempo mesmo que a parte determinística seja semelhante. Isso mostra que a parte estocástica da série é importante na análise e desenvolvimento de modelos para previsão dos dados. A análise do processo estocástico faz parte da construção de modelos regressivos para poder fazer previsões adequadas, minimizando o impacto causado pela aleatoriedade dos dados.

3.4 Componentes de Séries Temporais

Os componentes de séries temporais representam as diferentes fontes de variação que podem estar presentes em uma série temporal, sendo eles tendência, sazonalidade e resíduo (MORETTIN; TOLOI, 2018). Compreender esses componentes é fundamental na análise de séries temporais, pois ajuda a decompor a série em partes mais facilmente interpretáveis e a modelar melhor seu comportamento ao longo do tempo. A decomposição aditiva e multiplicativa são duas abordagens comuns usadas na análise de séries temporais para separar uma série temporal em seus componentes principais: tendência, sazonalidade e resíduo. Essas abordagens são úteis para entender os padrões subjacentes em uma série temporal e podem ajudar na previsão de futuros valores.

Decomposição Aditiva

A decomposição aditiva é apropriada quando as variações absolutas são importantes e a amplitude da sazonalidade não está relacionada ao nível da série (MORETTIN; TOLOI, 2018).

$$Z_t = T_t + S_t + R_t \tag{3.9}$$

Onde:

Z_t - Valor da série temporal no momento t;

T_t - Tendência no momento t;

S_t - Sazonalidade no momento t;

R_t - Resíduo no momento t.

Decomposição Multiplicativa

A decomposição multiplicativa é apropriada quando as variações percentuais são mais significativas e a amplitude da sazonalidade varia com o nível da série (MORETTIN; TOLOI, 2018).

$$Z_t = T_t \cdot S_t \cdot R_t \tag{3.10}$$

Z_t - Valor da série temporal no momento t;

T_t - Tendência no momento t;

S_t - Sazonalidade no momento t;

R_t - Resíduo no momento t.

É importante entender as características dos seus dados e as implicações da escolha da técnica para obter uma decomposição precisa e significativa dos componentes.

3.4.1 Tendência

A tendência em séries temporais refere-se a um padrão de mudança gradual ou direção consistente observada ao longo do tempo, sendo um componente fundamental em muitas séries temporais podendo revelar informações importantes sobre o comportamento de uma variável ao longo do tempo (BOX; JENKINS, 1971).

3.4.1.1 Método dos Mínimos Quadrados Ordinários

O método dos mínimos quadrados ordinários é uma técnica estatística usada para estimar os parâmetros de um modelo de regressão linear (BOX; JENKINS, 1971). Ele é amplamente utilizado na análise estatística e econometria para encontrar a linha de melhor ajuste que minimiza a soma dos quadrados das diferenças entre os valores observados e os valores previstos para o modelo. Esse método também pode ser aplicado em regressão polinomial, ajustando uma curva polinomial aos dados em vez de uma linha reta, o que permite modelar relações não lineares entre as variáveis independentes e dependentes. O modelo de regressão polinomial de grau m pode ser definido como:

$$z = a_0 + a_1 t + a_2 t^2 + \cdots + a_m t^m + \epsilon \tag{3.11}$$

Onde:

z - Modelo estimado;

m - Valor de maior grau do polinômio;

t - Tendência;

a - Estimadores da regressão;

ϵ - Resíduo.

A soma dos quadrados da regressão S_r para a regressão polinomial é da forma:

$$S_r = \sum_{i=1}^{n} (z_i - a_0 - a_1 t_i - a_2 t_i^2 - \cdots - a_m t_i^m)^2 \tag{3.12}$$

Para determinar os estimadores que minimizam Sr deriva-se a soma dos quadrados da regressão parcialmente em relação à cada estimador.

$$\frac{\partial S_r}{\partial a_j} = -2 \sum_{i=1}^{n} t_i^j (z_i - a_0 - a_1 t_i - a_2 t_i^2 - \cdots - a_m t_i^m) \tag{3.13}$$

Após isso cada derivada é igualada a zero para determinar os estimadores que minimizam S_r.

$$\left(\sum t_i^j\right) a_0 + \left(\sum t_i^{j+1}\right) a_1 + \left(\sum t_i^{j+2}\right) a_2 + \cdots + \left(\sum t_i^{j+m-2}\right) a_m = \sum t_i^j z_i \tag{3.14}$$

As equações obtidas podem ser representadas matricialmente da forma:

$$\begin{bmatrix} n & \sum t & \sum t^2 & \dots & \sum t^m \\ \sum t & \sum t^2 & \sum t^3 & \dots & \sum t^{m+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum t^m & \sum t^{m+1} & \sum t^{m+2} & \dots & \sum t^{2m} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} \sum z \\ \sum tz \\ \vdots \\ \sum t^m z \end{bmatrix} \tag{3.15}$$

Por fim, basta resolver o sistema definido pela equação matricial acima.

3.4.1.2 Propriedades e Pressupostos da Regressão Polinomial

Após determinar os estimadores que minimizam a soma dos quadrados da regressão, existem algumas propriedades e pressupostos que devem ser considerados para validar o modelo (BOX; JENKINS, 1971).

- Linearidade nos parâmetros: A relação entre as variáveis independentes e a variável dependente é linear nos parâmetros. Em um a regressão polinomial, isso significa que a relação linear nos coeficientes que multiplicam as diferentes potências do termo polinomial.
- Independência dos erros: Os resíduos da regressão são independentes entre si. Isso significa que o erro associado a uma observação não está relacionado aos erros das outras observações.
- Homocedasticidade: A variabilidade dos erros é constante em todos os níveis das variáveis independentes.
- Normalidade dos erros: Os erros da regressão seguem uma distribuição normal. Isso é importante para inferências estatísticas, como intervalos de confiança e testes de hipóteses.
- Ausência de multicolinearidade: É importante considerar a correlação entre variáveis independentes ao selecionar as variáveis a serem incluídas no modelo.
- Validade do modelo polinomial: Escolha adequada de um modelo polinomial para representar a relação entre as variáveis.
- Adequação do grau do polinômio: Além da validade do modelo polinomial, é importante escolher o grau apropriado do polinômio.

3.4.1.3 Média Móvel Simples

A média móvel é uma técnica comumente usada na análise de séries temporais para suavizar os dados e identificar tendências ou padrões subjacentes (MORETTIN; TOLOI, 2018). A média móvel envolve o cálculo de médias de um conjunto de valores em um determinado período de tempo, e a técnica é útil para remover o ruído aleatório e destacar padrões de longo prazo nos dados. Dado uma sequência m de elementos $P = (p_1, p_2, \ldots, p_m)$, a média móvel simples é calculada tomando a média aritmética dos valores em um período fixo.

$$\overline{P}_i = \frac{1}{n} \sum_{j=1}^{n} p_{i+j} \tag{3.16}$$

Onde:

$\overline{P}_i$ - Valor da média móvel simples no momento i;

p_{i+j} - Valores individuais da série temporal;

n - Número de observações.

3.4.1.4 Média Móvel Exponencial

A média móvel exponencial é uma técnica de suavização utilizada em análise de séries temporais para calcular uma média ponderada de valores passados (MORETTIN; TOLOI, 2018). Dado uma sequência m de elementos $P = (p_1, p_2, \ldots, p_m)$, a média móvel exponencial é dita como:

$$\overline{P}_i = \alpha \sum_{j=1}^{n-1} p_{i+j}(1-\alpha)^{j-1} + (1-\alpha)^{i+n} p_{i+n} \tag{3.17}$$

Onde:

$\overline{P}_i$ - Valor da média móvel exponencial no momento i;

P_{i+j} - Valores a serem suavizados;

n - Número de períodos a serem considerados na suavização;

α - Fator de suavização.

O cálculo referente ao fator de suavização é dado por:

$$\alpha = \frac{2}{s+1}, s \in \mathbb{N}^* \tag{3.18}$$

A saída do fator de suavização é um número entre 0 e 1. Quanto maior o valor de α, maior será o peso dado aos valores mais recentes da série temporal média móvel exponencial. Um valor baixo de α dará mais peso aos valores passados.

3.4.1.5 Desvio Padrão Móvel

O desvio padrão móvel é uma medida estatística que é usada em análise de séries temporais para avaliar a variabilidade dos dados em um determinado intervalo de tempo. É uma extensão do conceito de média móvel e é usado para fornecer informações sobre a dispersão dos valores em um conjunto de dados ao longo do tempo. A equação do desvio padrão móvel é análogo ao desvio padrão, porém aplicado à média móvel.

$$S_i = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(p_{i+j} - \overline{P}_i)^2} \tag{3.19}$$

Onde:

S_i - Valor do desvio padrão móvel no momento i;

p_{i+j} - Valores individuais da série temporal;

$\overline{P}_i$ - Média móvel simples no momento i;

n - Número de observações.

3.4.2 Sazonalidade

A sazonalidade em uma série temporal refere-se a padrões ou flutuações regulares e previsíveis que se repetem em intervalos fixos de tempo (MORETTIN; TOLOI, 2018). Esses padrões sazonais geralmente estão relacionados a fatores sazonais, como estação do ano, feriados, dias da semana ou outros ciclos recorrentes. A presença de sazonalidade em uma série temporal significa que os valores da série variam de acordo com esses ciclos previsíveis. É preciso tomar cuidado ao analisar a sazonalidade de uma série temporal para não se confundir com ciclos. Os ciclos em séries temporais se referem a variações de médio ou longo prazo que não são sazonais nem diretamente relacionadas a tendências lineares. Eles são padrões de oscilação que podem durar mais do que um ano e não têm uma periodicidade fixa ou previsível, ao contrário da sazonalidade.

3.4.3 Resíduo

O resíduo de uma série temporal é a parte não explicada ou não modelada da série temporal após a aplicação de um modelo estatístico (MORETTIN; TOLOI, 2018). Ele é importante na análise de séries temporais porque fornece informações sobre o quão bem o modelo se ajusta aos dados. Se os resíduos são pequenos e aleatórios, isso sugere que o modelo é capaz de capturar as principais tendências e padrões na série temporal. Em geral, um bom modelo de séries temporais deve ter resíduos que se aproximam de um ruído branco.

3.5 Ruído Branco

Ruído branco em séries temporais é uma fonte de variação aleatória e imprevisível que é adicionada à série temporal para representar a aleatoriedade ou os efeitos não sistêmicos que não podem ser explicados por nenhum modelo específico (BOX; JENKINS, 1971). A presença do ruído branco é importante para avaliar a precisão e a confiabilidade de modelos e previsões, uma vez que ajuda a explicar a variabilidade não sistemática dos dados. O conceito de ruído branco é dado aos resíduos de uma série temporal desde que satisfaçam as propriedades a seguir:

$$\begin{cases} E(\epsilon_t) = 0 \\ Var(\epsilon_t) = \sigma_\epsilon^2 \\ Cov(\epsilon_{t-\alpha}, \epsilon_{t-\beta}) = 0, \alpha \neq \beta \end{cases} \tag{3.20}$$

Para verificar se os resíduos de um modelo de regressão para uma série temporal seguem o conceito de ruído branco, são feitos testes estatísticos para cada propriedade para validar os resíduos os modelo. Nesse cenário são realizados os testes de Jarque-Bera para normalidade dos resíduos, teste de White para heterocedasticidade e teste de Ljung-Box para autocorrelação residual.

3.5.1 Normalidade dos Resíduos

3.5.1.1 Teste de Jarque-Bera

O teste de Jarque-Bera é usado para verificar a normalidade dos dados em duas estatísticas de momentos, a assimetria e curtose. O teste é uma ferramenta útil em análises estatísticas se é apropriado usar métodos que pressupõem normalidade dos dados (BOX; JENKINS, 1971). Se os dados não forem normalmente distribuídos, podem ser necessárias abordagens estatísticas alternativas. Em séries temporais é verificado a distribuição normal dos resíduos. A estatística do teste é calculada através da fórmula abaixo:

$$JB = \frac{n}{6}\left(A_s^2 + \frac{1}{4}(K-3)^2\right) \tag{3.21}$$

Onde A_s é o coeficiente de assimetria, K coeficiente de curtose e n tamanho da amostra. Já o teste de hipótese é da forma:

$$\begin{cases} H_0 : \text{Os dados seguem uma distribuição normal.} \\ H_a : \text{Os dados não seguem uma distribuição normal.} \end{cases} \tag{3.22}$$

3.5.2 Heterocedasticidade

A heterocedasticidade é um termo utilizado na estatística e econometria para descrever uma situação em que a variabilidade dos erros de um modelo estatístico não é constante em todos os níveis das variáveis independentes (STOA, 2015). Em um modelo de regressão, a heterocedasticidade ocorre quando a variabilidade dos erros é maior em algumas partes dos dados do que em outras. Isso pode ser problemático porque viola uma das premissas da regressão linear, que assume que os erros têm variâncias constantes, o que é conhecido como homocedasticidade.

3.5.2.1 Teste de White

O teste de White é um teste estatístico para verificar a presença de heterocedasticidade em um modelo de regressão (STOA, 2015). Por exemplo, admitindo um modelo com duas variáveis preditoras:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + u_i \tag{3.23}$$

O primeiro passo no teste é estimar o modelo inicial para obter os resíduos u_i. Após isso estimar a seguinte equação auxiliar:

$$R^2 \hat{u}_i^2 = \alpha_0 + \alpha_1 X_{i1} + \alpha_2 X_{i2} + \alpha_3 X_{i1}^2 + \alpha_4 X_{i2}^2 + \alpha_5 X_{i1} X_{i2} + v_i \tag{3.24}$$

Então fazer o teste de hipótese. A hipótese nula diz que o modelo é homocedástico (ideal) enquanto a hipótese alternativa diz que o modelo é heterocedástico (não ideal).

$$\begin{cases} H_0 : \alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \alpha_5 = 0 \\ H_a : \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5 \neq 0 \end{cases} \tag{3.25}$$

Analisar o seguinte critério estatístico:

$$nR^2 \sim X_k^2 \tag{3.26}$$

Se $nR^2 > X_c^2$, onde X_c^2 é algum valor crítico de uma distribuição Qui-Quadrada com k graus de liberdade, dado um nível de significância, rejeitar a hipótese de homocedasticidade. Caso o modelo seja heterocedástico, medidas corretivas podem ser tomadas como a utilização de técnicas de regressão robusta ou transformação dos dados.

3.5.3 Autocorrelação Residual

A autocorrelação dos resíduos refere-se à correlação entre os resíduos de um modelo estatístico em diferentes pontos no tempo ou em diferentes observações (BOX; PIERCE, 1970). Os resíduos são a diferença entre os valores observados e os valores previstos por um modelo, geralmente em séries temporais ou modelos de regressão. A detecção e tratamento da autocorrelação residual são importantes para garantir que as estimativas dos parâmetros do modelo sejam precisas e que as inferências feitas com base no modelo sejam confiáveis. A presença de autocorrelação não tratada pode levar a erros nas análises estatísticas e previsões imprecisas.

3.5.3.1 Teste de Ljung-Box

O teste de Ljung-Box é um teste estatístico usado para verificar a presença de autocorrelação nos resíduos de um modelo de séries temporais (BOX; PIERCE, 1970). A presença de autocorrelação nos resíduos indica que há padrões sistemáticos nas diferenças entre os valores observados e os valores previstos pelo modelo, o que pode sugerir que o modelo não captura completamente a estrutura temporal dos dados. O teste de hipótese é da forma:

$$\begin{cases} H_0 : \text{Os dados são independentemente distribuídos.} \\ H_a : \text{Os dados não são independentemente distribuídos.} \end{cases} \tag{3.27}$$

Em relação à estatística do teste:

$$Q = n(n+2)\sum_{k=1}^{h}\frac{\hat{p}_k^2}{n-k} \tag{3.28}$$

Onde:

k - Atraso (lag);

h - Total de atrasos;

$\hat{p}_k$ - Coeficientes de autocorrelação com k = 1, 2, ..., h;

n - Tamanho da amostra.

Dado um nível de significância α, a região crítica para rejeição da hipótese de aleatoriedade é:

$$Q > X^2_{1-\alpha,h} \tag{3.29}$$

Onde $X^2_{1-\alpha,h}$ é o $(1-\alpha)$ quartil da distribuição Qui-Quadrado com h graus de liberdade.

3.6 Estacionariedade

A estacionariedade de uma série temporal é uma propriedade fundamental em análise de séries temporais (MORETTIN; TOLOI, 2018). Uma série temporal é considerada estacionária quando suas propriedades estatísticas, média e variância, não mudam ao longo do tempo. Ela é importante porque simplifica a modelagem e a previsão de séries temporais. Em séries temporais estacionárias, os modelos estatísticas e as técnicas de previsão são mais aplicáveis, porque muitos métodos pressupõem que as propriedades estatísticas da série não mudam ao longo do tempo.

3.6.1 Teste de Dickey-Fuller Aumentado

O teste de Dickey-Fuller Aumentado é uma técnica estatística para determinar se a série temporal é estacionária (BOX; JENKINS, 1971). Ele é uma extensão do teste de Dickey-Fuller original e é amplamente usado na análise de séries temporais. O teste é projetado para testar a hipótese nula de que uma série temporal possui raiz unitária, o que a torna não estacionária. A equação do teste para modelo sem constante é dada por:

$$\Delta y_t = \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \delta_2 \Delta y_{t-2} + \cdots + \delta_p \Delta y_{t-p} + \epsilon_t \tag{3.30}$$

Onde:

Δy_t - Diferença de duas observações consecutivas no momento t;

γ - Raiz unitária;

δ_i - Coeficientes que multiplicam as diferenças passadas onde $i = 1, \ldots, p$;

p - Ordem de atraso do processo regressivo;

ϵ_t - Termo de erro independente em série no tempo t para $t = 2, \ldots, T$;

T - Tamanho da amostra.

O modelo pode ser ajustado para outras variações como presença de constante, tendência linear e tendência quadrática dependendo do contexto da análise. O teste de hipótese nesse caso é:

$$\begin{cases} H_0 : \gamma = 0 \\ H_a : \gamma \neq 0 \end{cases} \tag{3.31}$$

3.7 Função de Autocorrelação

A função de autocorrelação é usada para analisar a relação entre os valores de uma série temporal e seus próprios valores defasados em relação ao tempo (BOX; PIERCE, 1970). A análise da função de autocorrelação é uma etapa fundamental na modelagem e previsões de séries temporais, pois ajuda a entender a estrutura da série e a escolher o melhor modelo estatístico para descrever seu comportamento ao longo do tempo. O coeficiente de correlação entre r_t e r_{t-k} é chamado de autocorrelação de k-ésima ordem, ou seja, a autocorrelação na defasagem k (lag) é detonado por:

$$p_k = \frac{Cov(r_t, r_{t-k})}{\sqrt{Var(r_t, r_{t-k})}} = \frac{Cov(r_t, r_{t-k})}{Var(r_t)} = \frac{\gamma_k}{\gamma_0} \tag{3.32}$$

Um conjunto de autocorrelações p_k é chamado de função de autocorrelação de r_t. A autocorrelação amostral de k-ésima ordem de r_t pode ser definida como:

$$\hat{p}_k = \frac{\sum_{t-k+1}^{T}(r_t - \overline{r})(r_{t-k} - \overline{r})}{\sum_{t-1}^{T}(r_t - \overline{r})^2}, 0 \leq k \leq T-1 \tag{3.33}$$

Onde:

$\overline{r}$ - Média amostral;

T - Tamanho da amostra.

Uma autocorrelação positiva em um determinado lag indica que valores anteriores têm uma relação positiva com os valores atuais, enquanto uma autocorrelação negativa indica uma relação negativa. Uma autocorrelação próxima de zero indica que não há uma relação linear forte entre os valores em um determinado lag.

3.7.1 Função de Autocorrelação Parcial

A função de autocorrelação parcial tenta isolar a relação direta entre os valores atuais e passados, eliminando a influência dos valores em lags intermediários (BOX; PIERCE, 1970). Ela filtra correlações e mantêm apenas a correlação pura entre duas observações.

$$r_t = \phi_{j,1} r_{t-1} + \phi_{j,2} r_{t-2} + \cdots + \phi_{j,j} r_{t-j} + \epsilon_t \tag{3.34}$$

Onde:

$\phi_{j,n}$ - Coeficiente de autocorrelação parcial no lag j com $n = 1, 2, \ldots, j$;

ϵ_t - Erro aleatório no tempo t.

Os coeficientes de autocorrelação são analisados para ajustar modelos de média móvel enquanto coeficientes de autocorrelação parcial para modelos autorregressivos.

3.8 Metodologia Box & Jenkins

A metodologia BOX e JENKINS (1971), é uma abordagem amplamente usada para modelar e prever séries temporais. É altamente flexível e pode ser aplicada a uma ampla variedade de séries temporais, desde dados financeiros até previsão de vendas e séries climáticas. É especialmente útil quando dados exibem padrões complexos e não lineares. A metodologia para esse trabalho é da forma:

i) Identificação: Identificar o tipo de modelo autorregressivo integrado de médias móveis com sazonalidade (SARIMA) apropriado para a série temporal. Isso envolve a determinação dos componentes do modelo autorregressivo (AR) e de médias móveis (MA) e a ordem de diferenciação necessária para tornar a série estacionária. Por ter sazonalidade os parâmetros sazonais também são analisados.

ii) Estimação: Identificar os parâmetros do modelo SARIMA usando métodos estatísticos como a máxima verossimilhança. São analisados os valores adequados para os coeficientes do modelo.

iii) Diagnóstico: Verificar a adequação do modelo estimado. Isso inclui a análise dos resíduos do modelo para garantir que eles sejam ruído branco. Se os resíduos não forem satisfatórios é preciso ajustar o modelo e repetir as etapas anteriores.

Após verificar que o modelo apresenta ruído branco é possível fazer previsões para dados fora da amostra. Embora são seja obrigatório, a ausência de ruído branco no modelo pode afetar a precisão das previsões e a validade das inferências feitas a partir do modelo.

3.8.1 ARMA (p,q)

O modelo autorregressivo de médias móveis (ARMA) junta os modelos de autorregressão e de média móvel para ajuste de séries temporais estacionárias em nível (MORETTIN; TOLOI, 2018). O modelo AR(p) é baseado no conceito de autocorrelação e na ideia de que o valor atual de uma série temporal está linearmente relacionado com seus valores anteriores.

$$Z_t = \mu + \phi_1 Z_{t-1} + \phi_2 Z_{t-2} +, \ldots, + \phi_p Z_{t-p} + \epsilon_t \tag{3.35}$$

Onde:

μ - Média do processo;

ϕ_i - Coeficientes de autorregressão com $i = 1, 2, \ldots, p$;

ϵ_t - Resíduo.

Já o modelo MA(q) é modelado como uma combinação linear dos valores de erro anteriores, ponderados por coeficientes específicos.

$$Z_t = \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} -, \ldots, -\theta_q\epsilon_{t-q} \tag{3.36}$$

Onde:

θ_i - Coeficientes de média móvel com $i = 1, 2, \ldots, q$;

ϵ_t - Resíduo.

A média do processo AR(p) quando o modelo apresenta ruído branco é nula. Agrupando os modelos anteriores dá origem ao modelo ARMA(p,q) dado por:

$$Z_t = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} +, \ldots, +\phi_p Z_{t-p} + \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} -, \ldots, -\theta_q\epsilon_{t-q} \tag{3.37}$$

Onde:

ϕ_i - Coeficientes autorregressivos com $i = 1, 2, \ldots, p$;

θ_i - Coeficientes de média móvel com $i = 1, 2, \ldots, q$;

ϵ_t - Resíduo.

3.8.2 ARIMA (p,d,q)

O modelo autorregressivo integrado de médias móveis (ARIMA) possui ordem de integração para ajuste de séries temporais não estacionárias em nível. Caso a série seja estacionária em nível, o modelo funciona similarmente ao modelo ARMA (MORETTIN; TOLOI, 2018).

$$\Delta^d Z_t = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} +, \ldots, +\phi_p Z_{t-p} + \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} -, \ldots, -\theta_q\epsilon_{t-q} \tag{3.38}$$

Onde:

Δ^d - Ordem de integração com d diferenças;

ϕ_i - Coeficientes autorregressivos com $i = 1, 2, \ldots, p$;

θ_i - Coeficientes de média móvel com $i = 1, 2, \ldots, q$;

ϵ_t - Resíduo.

Note que o modelo é adequado quando na série temporal não há indícios de sazonalidade. Se tiver é preciso ajustar o modelo de modo a capturar a sazonalidade da série.

3.8.3 SARIMA (p,d,q)(P,D,Q)

De acordo com MORETTIN e TOLOI (2018), o modelo SARIMA é um caso geral dos modelos propostos por BOX e JENKINS (1971), para o ajuste de séries temporais estacionárias. Esse modelo é adequado quando há presença de componente sazonal nos dados, permitindo assim que o modelo consiga capturar a sazonalidade da série temporal ao longo do tempo. O modelo estimado é denominado SARIMA(p,d,q)(P,D,Q) dado por:

$$\begin{aligned}\nabla^D \Delta^d Z_t = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} + \cdots + \phi_p Z_{t-p} + \epsilon_t - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - \cdots - \theta_q \epsilon_{t-q} \\ + \Phi_1 Z_{t-1S} + \Phi_2 Z_{t-2S} + \cdots + \Phi_P Z_{t-P} - \Theta_1 \epsilon_{t-1S} - \Theta_2 \epsilon_{t-2S} - \cdots - \Theta_Q \epsilon_{t-Q}\end{aligned} \tag{3.39}$$

Onde:

∇^D - Ordem de integração sazonal com D diferenças;

Δ^d - Ordem de integração não sazonal com d diferenças;

ϕ_i - Coeficientes autorregressivos com $i = 1, 2, \ldots, p$;

θ_i - Coeficientes de média móvel com $i = 1, 2, \ldots, q$;

Φ_i - Coeficientes autorregressivos sazonais com $i = 1, 2, \ldots, P$;

Θ_i - Coeficientes de média móvel sazonal com $i = 1, 2, \ldots, Q$;

ϵ_t - Resíduo.

3.9 Critérios para Validação do Modelo

A validação de um modelo estatístico é uma etapa crítica no processo de modelagem, pois permite avaliar o desempenho do modelo e garantir sua capacidade de generalização para dados não observados. A comparação de modelos tem o objetivo de comparar modelos validados. Nesse cenário são usados critérios de informação para analisar qual modelo é mais adequado para previsão (SIN; WHITE, 1996).

3.9.1 Função de Verossimilhança

A função de verossimilhança é usada para estimar os parâmetros de um modelo estatístico com base em um conjunto de dados observados. Busca-se encontrar os valores dos parâmetros que maximizam a probabilidade de observar os dados em questão (BOX; JENKINS, 1971). Em modelos que apresentam ruído branco ela é baseada em uma distribuição normal. A função de distribuição normal é dada por:

$$f(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.40}$$

Onde:

x - Variável aleatória;

μ - Média populacional;

σ^2 - Variância populacional.

A função de verossimilhança é dada como o produto de cada função $f(x_i; \mu, \sigma^2)$ onde $i = 1, 2, \ldots, n$.

$$L(\mu, \sigma^2; x) = \prod_{i=1}^{n} f(x_i; \mu, \sigma^2) \tag{3.41}$$

Analisa-se a decomposição do produtório referente à equação anterior. Note que o primeiro termo é multiplicado repetidas vezes enquanto o expoente do segundo termo é somado consecutivamente.

$$L(\mu, \sigma^2; x) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(-\sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2}\right) \tag{3.42}$$

É comum utilizar a função em log para facilitar os cálculos. O logaritmo da função de verossimilhança é dado por:

$$l(\mu, \sigma^2; x) = -\frac{n}{2}\log(2\pi\sigma^2) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \tag{3.43}$$

A função de verossimilhança em log é utilizada nas equações dos critérios AIC, BIC e HQIC para determinar a máxima verossimilhança dos parâmetros de modelos.

3.9.2 Critério de Informação de Akaike (AIC)

O critério AIC é usado na estatística e em análise de modelos para comparar modelos estatísticos, levando em consideração o equilíbrio entre a capacidade de ajuste do modelo e a complexidade do modelo. Quanto menor o valor do critério de informação, melhor será a qualidade do ajuste (SIN; WHITE, 1996).

$$\begin{aligned} AIC(l, k) &= -2\sum_{i=1}^{n} \log f(x_i|\hat{\theta}) + 2k \\ &= -2\log L(\hat{\theta}) + 2k \end{aligned} \tag{3.44}$$

Onde:

$l = \log L(\hat{\theta})$ - Logaritmo da função de máxima verossimilhança;

k - Número de parâmetros do modelo.

Note que esse critério apresenta viés potencial em direção a modelos mais complexos quando há um número limitado de dados. Para isso existe o AICc (Critério de Informação de Akaike Corrigido) que é uma versão corrigida do AIC usado especialmente quando o número de observações (n) é relativamente pequeno em relação ao número de parâmetros no modelo (SIN; WHITE, 1996).

$$AICc(l,k,n) = -2\log L(\hat{\theta}) + 2k + \left(\frac{2k^2+2}{n-k-1}\right) \tag{3.45}$$

Onde:

$l = \log L(\hat{\theta})$ - Logaritmo da função de máxima verossimilhança;

k - Número de parâmetros do modelo.

n - Tamanho da amostra.

3.9.3 Critério de Informação de Schwarz (BIC)

O critério BIC é semelhante ao AIC e também ajuda a avaliar a qualidade de ajuste de um modelo estatísticos, considerando o equilíbrio entre o ajuste do modelo e sua complexidade. O critério BIC foi desenvolvido de acordo com princípios da inferência bayesiana. Diferente do critério anterior, ele penaliza modelos mais complexos de forma mais severa, sendo útil para evitar a seleção de modelos excessivamente complexos (SIN; WHITE, 1996).

$$\begin{aligned} BIC(l,k,n) &= -2\sum_{i=1}^{n}\log f(x_i|\hat{\theta}) + k\log n \\ &= -2\log L(\hat{\theta}) + k\log n \end{aligned} \tag{3.46}$$

Onde:

$l = \log L(\hat{\theta})$ - Logaritmo da função de máxima verossimilhança;

k - Número de parâmetros do modelo.

n - Tamanho da amostra.

3.9.4 Critério de Informação de Hannan-Quinn (HQIC)

Assim como os outros critérios de informação, de acordo com SIN e WHITE (1996), o critério HQIC é uma alternativa aos critérios anteriores, sendo uma extensão do AIC. A penalização do HQIC é mais moderada em comparação com BIC, sendo assim mais adequado caso o modelo não seja muito simples ou complexo.

$$HQIC(l, k, n) = -2k \log(\log n) - 2 \log(L(\hat{\theta})) \tag{3.47}$$

Onde:

$l = \log L(\hat{\theta})$ - Logaritmo da função de máxima verossimilhança;

k - Número de parâmetros do modelo.

n - Tamanho da amostra.

3.10 Gradiente Descendente Estocástico

O gradiente de uma função é um vetor que aponta na direção de máxima taxa de crescimento da função no ponto em que é avaliado.

$$\nabla f(x_1, x_2, \ldots, x_n) = \left\langle \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right\rangle \tag{3.48}$$

O gradiente descendente é um algoritmo de otimização usado para ajustar os parâmetros de um modelo de maneira a minimizar uma função objetiva (MURPHY, 2021). A função objetiva, também chamada função de custo ou função de perda, quantifica o quão bem o modelo está realizando uma tarefa específica, permitindo assim que o modelo faça previsões mais precisas. Seu conceito fundamental é de forma iterativa ajustar os parâmetros do modelo na direção em que a função objetiva está diminuindo mais rapidamente. O gradiente descendente estocástico (SGD) é uma variação do algoritmo de otimização do gradiente descendente que atualiza os parâmetros do modelo após cada exemplo de treinamento individual. Ele introduz estocasticidade no processo, o que pode ajudar a evitar mínimos locais e a escapar de platôs na função objetiva. Em conformidade com MURPHY (2021), o vetor de parâmetros do SGD é dado por:

$$Q(w) = \frac{1}{n} \sum_{i=1}^{n} Q_i(w) \tag{3.49}$$

$$w := w - \eta \nabla Q(w) = w - \frac{\eta}{n} \sum_{i=1}^{n} \nabla Q_i(w) \tag{3.50}$$

Onde:

$Q(w)$ - Função objetiva;

$\nabla Q(w)$ - Gradiente da função objetiva.

n - Tamanho do conjunto;

η - Taxa de aprendizado;

3.10.1 Método dos Momentos

O método dos momentos é usado para acelerar a convergência durante o processo de otimização. Então os parâmetros do modelo são definidos com o acréscimo do vetor de momento (SUTSKEVER, 2013).

$$\Delta w := \alpha \Delta w - \eta \nabla Q_i(w) \tag{3.51}$$

$$w := w - \eta \nabla Q_i(w) + \alpha \Delta w \tag{3.52}$$

Onde:

Δw - Vetor de momento;

$\nabla Q_i(w)$ - Gradiente da função objetiva.

α - Taxa de momento;

η - Taxa de aprendizado;

3.10.2 Propagação da Raiz Quadrática Média (RMSProp)

A propagação de raiz quadrática média é um algoritmo de otimização que também serve para minimizar a função objetiva. De acordo com BUSHAEV (2018), RMSProp é uma das variações do gradiente descendente que adaptam a taxa de aprendizado de maneira automática durante o processo de treinamento. O algoritmo foi projetado para superar algumas limitações tradicionais como a escolha de taxa de aprendizado fixa que pode levar a problemas de convergência lenta ou instabilidade.

$$v(w,t) := \gamma v(w,t-1) + (1-\gamma)(\nabla Q_i(w))^2 \tag{3.53}$$

$$w := w - \frac{\eta}{\sqrt{v(w,t)}} \nabla Q_i(w) \tag{3.54}$$

Onde:

$v(w,t)$ - Vetor de acumulação de quadrados dos gradientes;

$\gamma v(w,t-1)$ - Média móvel ponderada dos quadrados dos gradientes anteriores;

$(1-\gamma)(\nabla Q_i(w))^2$ - Quadrado do gradiente atual ponderado.

3.10.3 Estimativa de Momento Adaptativo (Adam)

A estimativa de momento adaptativo, também conhecido como Adam, é um algoritmo de otimização usado para treinar modelos em aprendizado de máquina e aprendizado profundo (KINGMA; BA, 2014). O Adam é uma melhoria em relação a outros algoritmos de otimização porque ele adapta dinamicamente as taxas de aprendizado durante o treinamento. Esse algoritmo combina as técnicas de estimação de momento e RMSProp. Os parâmetros são atualizados com base nos gradientes e nas médias móveis exponenciais $m_w^{(t+1)}$ e $v_w^{(t+1)}$.

$$m_w^{(t+1)} \longleftarrow \beta_1 m_w^{(t)} + (1-\beta_1)\nabla_w L^{(t)} \tag{3.55}$$

$$v_w^{(t+1)} \longleftarrow \beta_2 v_w^{(t)} + (1-\beta_2)(\nabla_w L^{(t)})^2 \tag{3.56}$$

Onde:

$\nabla_w L^{(t)}$ - Gradiente atual da função perda;

β_1 - Taxa de decaimento da primeira média móvel exponencial;

β_2 - Taxa de decaimento da segunda média móvel exponencial.

Agora são feitas correções de viés para as médias móveis exponenciais.

$$\hat{m}_w = \frac{m_w^{(t+1)}}{1-\beta_1^t} \tag{3.57}$$

$$\hat{v}_w = \frac{v_w^{(t+1)}}{1-\beta_2^t} \tag{3.58}$$

Feito isso o vetor de parâmetros do modelo na ordem de tempo $(t+1)$ após a atualização é da forma:

$$w^{(t+1)} \longleftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon} \tag{3.59}$$

Onde:

η - Taxa de aprendizado;

$\hat{m}_w$ - Correção de viés para a primeira média móvel exponencial;

$\hat{v}_w$ - Correção de viés para a segunda média móvel exponencial;

ϵ - Pequeno escalar.

3.11 Funções de Ativação

As funções de ativação são componentes fundamentais das redes neurais artificiais e de outros modelos de aprendizado de máquina. Elas são funções matemáticas que determinam a ativação ou saída de um neurônio com base nas entradas que ele recebe. As funções de ativação introduzem ausência de linearidade nas redes neurais, permitindo que elas capturem e aprendam relações complexas nos dados (HAYKIN, 2001).

Retificadora

A função ReLU (Unidade Linear Retificada) é uma das funções de ativação mais populares. Ela passa a entrada diretamente para entradas positivas e retorna zero para entradas negativas.

$$\phi(x) = \begin{cases} x, \, se \, x > 0 \\ 0, \, se \, x \leq 0 \end{cases} \tag{3.60}$$

$$\phi'(x) = \frac{\phi}{x} \tag{3.61}$$

Sigmoide

A função sigmoide mapeia a entrada para um valor entre 0 e 1. É frequentemente usada em neurônios de saída binária ou para representar probabilidades.

$$\phi(x) = \frac{1}{1 + e^{-x}} \tag{3.62}$$

$$\phi'(x) = \phi(x) - \phi^2(x) \tag{3.63}$$

Tangente Hiperbólica

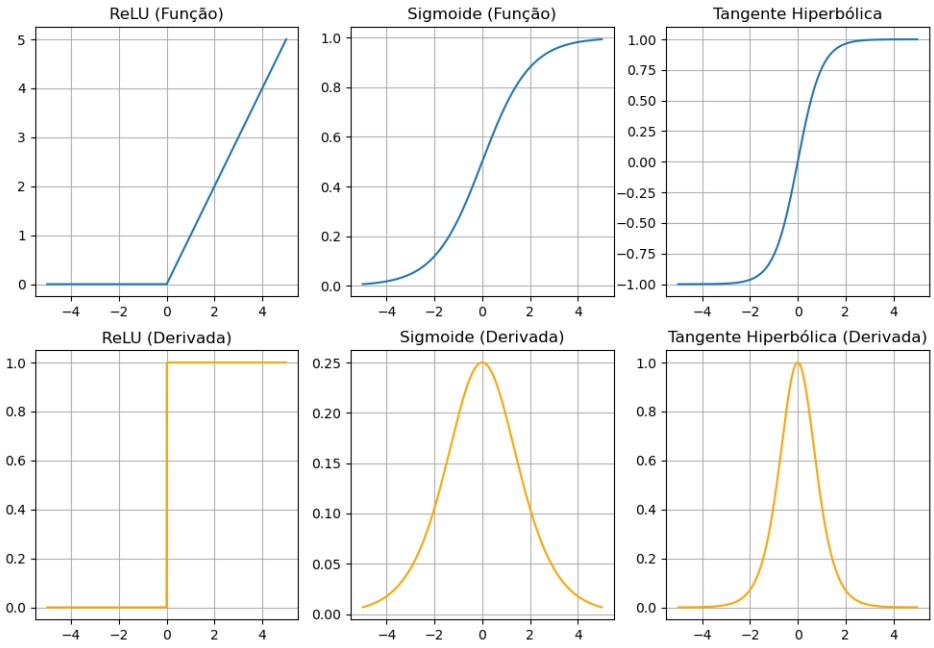
A função tangente hiperbólica mapeia a entrada para um valor entre -1 e 1. Ela é útil para camadas intermediárias de redes neurais.

$$\phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.64}$$

$$\phi'(x) = 1 - \phi^2(x) \tag{3.65}$$

A aplicação dessas funções dependem do contexto do problema. É comum, por exemplo, normalizar dados para trabalhar com as funções sigmoide e tangente hiperbólica. Em compensação, a função retificadora é mais simples e não requer, necessariamente, ajuste nos dados.

Figura 8 – Funções de ativação



Fonte: Elaborado pelo Autor

A função de ativação retificada é popular devido à simplicidade e eficiência computacional, sendo preferida quando se deseja lidar com problemas de regressão e classificação. A função sigmoide é usada em redes neurais mais antigas e também em unidades de saída de classificação binária, podendo ter o problema do desvanecimento do gradiente (gradiente muito pequeno). Por fim, a tangente hiperbólica também é usada em redes neurais profundas, porém está menos propensa ao problema de desvanecimento do gradiente em comparação à sigmoide devido à média dos valores de saída estarem próximos de zero. Isso significa que os dados são centrados em torno de zero, facilitando a propagação do gradiente em ambas as direções (positivo e negativo) durante o treinamento. Diferente da função de retificação, as funções sigmoide e tangente hiperbólica são mais sensíveis à dados próximos de zero. Isso pode ser facilmente analisado através de suas derivadas na Figura 8.

3.12 Intervalos de Confiança

Os intervalos de confiança são uma ferramente importante em estatística que ajudam a estimar um parâmetro desconhecido de uma população com base em uma amostra de dados (MORETTIN; TOLOI, 2018). Esses intervalos fornecem uma faixa de valores dentro da qual é razoável supor que o valor do parâmetro real esteja com uma certa probabilidade, geralmente expressa em termos de nível de confiança dados em percentual. Por padrão, é comum construir intervalos de confiança com nível de confiança de $\gamma = 1 - \alpha$

que, por consequência, apresenta nível de significância de $\alpha = 5\%$ que é interpretado como o seu complementar.

Nível de Confiança (γ)

O nível de confiança é um conceito estatístico que está relacionado à precisão das estimativas feitas a partir de dados amostrais. Ele representa a probabilidade de que uma estimativa contenha o valor real de um parâmetro da população, supondo que o mesmo processo de amostragem fosse realizado repetidas vezes.

Nível de Significância (α)

O nível de significância é um conceito estatístico que está relacionado à tomada de decisões em testes de hipóteses. Ele representa a probabilidade de cometer um erro de significância que ocorre quando se rejeita uma hipótese nula verdadeira.

3.12.1 Distribuição Normal

A distribuição normal é uma das distribuições de probabilidade mais importantes na estatística (MORETTIN; TOLOI, 2018). Ela é caracterizada por uma forma de sino, simétrica e unimodal, e é amplamente utilizada devido a suas propriedades matemáticas bem compreendidas. A distribuição normal é descrita por dois parâmetros: média (μ) e variância (σ).

$$X \sim N(\mu, \sigma^2) \tag{3.66}$$

O intervalo de confiança para dados estimados supondo uma distribuição normal pode ser definido como:

$$IC_{1-\alpha}(x_i) = \left(\hat{x}_i - Z_{1-\alpha/2} \cdot \epsilon, \hat{x}_i + Z_{1-\alpha/2} \cdot \epsilon\right) \tag{3.67}$$

Onde:

$\hat{x}_i$ - Dado previsto onde $i = 1, \ldots, n$;

α - Nível de significância;

$Z_{(1-\frac{\alpha}{2})}$ - Z crítico;

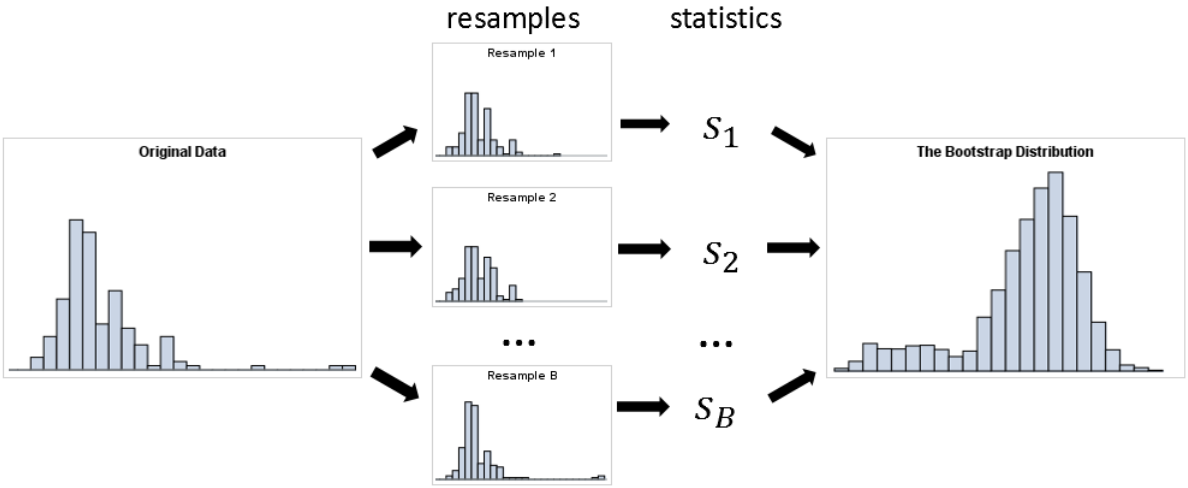
ϵ - Margem de erro.

A margem de erro é ajustada de acordo com o contexto do problema. Se o desvio padrão populacional for conhecido, a margem de erro é dada por $\epsilon = \sigma$. Caso o desvio padrão populacional não seja conhecido ou a amostra tenha tamanho acima de 30 elementos, então a margem de erro será da forma $\epsilon = \frac{\sigma}{\sqrt{n}}$, onde, nesse caso, σ é o desvio padrão amostral e n tamanho da amostra.

3.12.2 Bootstrap

O método de Bootstrap é uma técnica estatística que envolve a geração de múltiplas amostras (reamostragens) a partir de uma única amostra de dados original, com o objetivo de estimar a distribuição amostral de uma estatística de interesse (WICKLIN, 2018). Isso é particularmente útil quando é gerada uma amostra pequena ou quando deseja obter uma estimativa da variabilidade de uma estatística sem assumir que a população segue uma distribuição específica.

Figura 9 – Bootstrap por reamostragem



Fonte: WICKLIN, 2018

O processo de bootstrap envolve os seguintes passos:

- Coleta de dados: Se inicia com uma única amostra de dados, que pode ser pequena e representativa de uma população maior.
- Reamostragem: Gerar múltiplas amostras de dados a partir da amostra original. No caso de uma série temporal montar blocos de mesmo tamanho e então permutar aleatoriamente os dados de cada bloco para manter a dependência temporal.
- Estimativa: Faz as estimativas necessárias para cada subamostra.
- Análise dos resultados: Com as estimativas a partir das subamostras, obtêm-se uma estimativa da distribuição amostral da estatística de interesse.

3.13 Métricas de Desempenho

Métricas de desempenho são medidas quantitativas usadas para avaliar ou quantificar aspectos específicos de conjunto de dados. Elas são essenciais para a tomada de

decisão informadas, a avaliação de desempenho e a comparação de resultados em diferentes contextos. De acordo com SANTOS (2019), as métricas podem ser usadas em redes neurais para comparar dados previstos com os reais, assim como modelos estatísticos.

3.13.1 Erro Absoluto Médio

O EAM é uma métrica que calcula a média das diferenças absolutas entre as previsões de um modelo e os valores reais. Ela mede o erro médio em termos de magnitude ignorando a direção do erro (MORETTIN; TOLOI, 2018).

$$EAM = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.68}$$

Onde:

y_i - Valores observados;

$\hat{y}_i$ - Valores previstos;

n - Tamanho do conjunto.

3.13.2 Raiz do Erro Quadrático Médio

A REQM é uma métrica que calcula a raiz quadrada da média das diferenças quadráticas entre as previsões de um modelo e os valores reais. Ela faz com que os erros maiores tenham um impacto mais significativo na métrica (MORETTIN; TOLOI, 2018).

$$REQM = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.69}$$

Onde:

y_i - Valores observados;

$\hat{y}_i$ - Valores previstos;

n - Tamanho do conjunto.

3.13.3 Coeficiente de Determinação

O R^2 é uma métrica que fornece uma medida de qualidade da regressão de um modelo em relação à variabilidade dos dados. Ela varia de 0 a 1, onde 1 indica que o modelo explica perfeitamente a variabilidade dos dados e 0 indica que o modelo é inferior à média (MORETTIN; TOLOI, 2018).

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y}_i)^2} \tag{3.70}$$

Onde:

y_i - Valores observados;

$\hat{y}_i$ - Valores previstos;

$\overline{y}_i$ - Média dos valores observados;

n - Tamanho do conjunto.

3.13.4 U de Theil

A métrica U de Theil compara a variabilidade das previsões de um modelo com a variabilidade dos valores reais. Ela é usada para avaliar a precisão da previsão em relação à variabilidade natural dos dados. A métrica foi criada pelo econometrista holandês Henri Theil para avaliar modelos preditivos para séries temporais. Existem duas equações para calcular o valor de U de Theil, sendo elas $U1$ e $U2$. Para analisar a qualidade preditiva de intervalos de previsão é utilizada a segunda equação por ser mais sofisticada (BLIEMEL; MACKAY, 1973).

$$U1 = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - f_i)^2}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} y_i^2} + \sqrt{\frac{1}{n}\sum_{i=1}^{n} f_i^2}} \tag{3.71}$$

$$U2 = \sqrt{\frac{\frac{1}{n}\sum_{i=1}^{n-1}\left(\frac{f_{i+1} - y_{i+1}}{y_i}\right)^2}{\frac{1}{n}\sum_{i=1}^{n-1}\left(\frac{y_{i+1} - y_i}{y_i}\right)^2}} \tag{3.72}$$

Onde:

y_i - Valores observados;

f_i - Valores previstos;

n - Tamanho do conjunto.

A primeira equação é utilizada quando é desejável analisar modelos quantitativamente, similarmente à métricas como EAM e REQM, enquanto a segunda equação serve para analisar se modelos são superiores à previsão ingênua, ou seja, é feita uma análise qualitativa do modelo (BLIEMEL; MACKAY, 1973).

4 Fundamentação Computacional

Esse capítulo traz algumas informações a respeito da teoria computacional referente ao desenvolvimento dessa monografia.

4.1 Software Python

Para o desenvolvimento dessa monografia foi utilizada a linguagem de programação Python. O Python foi inicialmente idealizado por Guido van Rossum nos anos 80 como uma linguagem de programação de alto nível, intuitiva e fácil de ler. Sua criação foi inspirada no desejo de se ter uma linguagem poderosa e versátil, com ênfase na legibilidade do código. A primeira versão oficial, Python 0.9.0, foi lançada em 1991.

O Python faz uso de diversas bibliotecas para expandir sua funcionalidade básica. Essas bibliotecas oferecem conjuntos de ferramentas específicas para tarefas como manipulação de dados, criação de interfaces gráficas e desenvolvimento web e são fundamentais para a eficiência e flexibilidade no desenvolvimento analises com esta linguagem.

Para o desenvolvimento e análise dos modelos SARIMA e LSTM, foi instalado o pacote Anaconda, que possui diversas bibliotecas para trabalhar em projetos estatísticos, e o ambiente Jupyter Notebook para IDE. Foi necessária a instalação das bibliotecas Pmdarima e TensorFlow separadamente por não serem nativas no Anaconda. Elas foram fundamentais para o desenvolvimento do modelo estatístico e rede neural, respectivamente. A funcionalidade de cada biblioteca é dada por:

- Numpy: Suporte para arrays multidimensionais, funções matemáticas, álgebra linear, geração de números aleatórios, entre outras funcionalidades.
- Pandas: Estrutura de dados flexíveis e eficientes, como DataFrames, que facilitam a manipulação e análise de conjuntos de dados tabulares.
- Matplotlib: Variedade de gráficos e plotagens, permitindo a representação visual de dados de maneira clara e informativa.
- Seaborn: Simplifica a criação de gráficos estatísticos atraentes e informativos.
- Statsmodels: Estimativas e testes estatísticos.
- Pmdarima: Análise e modelagem de séries temporais.
- TensorFlow: Aprendizado de máquina e aprendizado profundo.

4.2 Engenharia de Dados

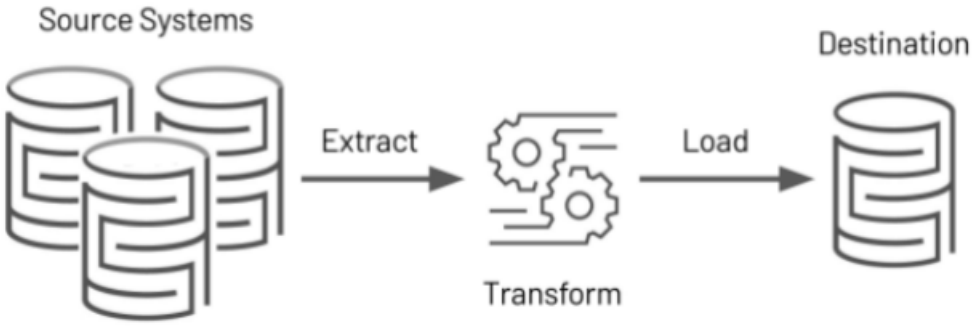
De acordo com BABU e SUMA (2022), a engenharia de dados é uma disciplina que se concentra em coletar, armazenar, processar e disponibilizar dados para análise e tomada de decisões usando pipelines como mecanismo de entrada e saída de dados. Pipeline de dados é um conjunto de processos automatizados e interconectados que permitem a transferência de dados de uma fonte para um destino, passando por várias etapas de processamento ao longo do caminho. Um dos pipelines básicos em engenharia de dados é o pipeline ETL. Ele funciona como uma coletânea de procedimentos para transferir dados de fontes para uma base de dados (BABU; SUMA, 2022). Esse processo se concentra em:

1. Extração: Extrair dados de fonte de dados, podendo elas ser homogêneas ou heterogêneas. As fontes são fundamentais para a criação de pipeline de dados.

2. Transformação: Processar e transformar dados para atender aos requisitos específicos de análise ou armazenamento. Nessa etapa costuma incluir limpeza de dados, agregação, filtragem e outros processos para tratamento de dados.

3. Carga: Carregar dados a um destino onde podem ser acessados e consultados para análise. Essa etapa tende a ser mais flexível com linguagens de programação porque permitem que o carregamento possa ser feito diretamente no ambiente IDE.

Figura 10 – Processo ETL



Fonte: BABU e SUMA, 2022

4.3 Aprendizado de Máquina

O aprendizado de máquina (machine learning) é um subcampo da inteligência artificial que se concentra no desenvolvimento de algoritmos e modelos que permitem que os sistemas de computador aprendam e melhorem seu desempenho em tarefas específicas a partir de dados, sem serem explicitamente programados para essas tarefas (GOODFELLOW; BENGIO; COURVILLE, 2016). Em vez de seguir instruções específicas em programação, os sistemas de aprendizado de máquina utilizam técnicas estatísticas para reconhecer padrões nos dados e fazer previsões ou tomar decisões com base nesses padrões. Os três tipos principais de aprendizado de máquina são:

- Aprendizado Supervisionado: O algoritmo é treinado com um conjunto de dados de treinamento que inclui pares de entrada e saída. O objetivo é aprender uma função que mapeie as entradas para as saídas correspondentes. Uma vez treinado, o modelo é capaz de fazer previsões ou classificações em novos dados com base no que aprendeu durante o treinamento.
- Aprendizado Não-Supervisionado: O algoritmo é treinado com um conjunto de dados que não inclui rótulo de saída. Seu objetivo é encontrar estrutura nos dados, identificando padrões, grupos ou relações ocultas entre os pontos de dados.
- Aprendizado por Reforço: Um agente interage com o ambiente e toma decisões para maximizar uma recompensa cumulativa. O agente aprende a melhorar seu desempenho através da tentativa e erro, recebendo feedback na forma de recompensas ou punições. O objetivo é encontrar uma política de ação que maximize a recompensa ao longo do tempo.

4.3.1 Aprendizado Profundo

O aprendizado profundo (deep learning) é uma subárea do aprendizado de máquina que se concentra no uso de redes neurais artificiais para aprender e realizar tarefas complexas (GOODFELLOW; BENGIO; COURVILLE, 2016). As redes neurais artificiais são modelos computacionais inspirados no funcionamento do cérebro humano, compostos por camadas de unidades interconectadas chamadas neurônios. A característica distintiva do aprendizado profundo é o uso de arquiteturas de redes neurais profundas, que consistem em várias camadas. Essas redes profundas têm a capacidade de aprender automaticamente representações hierárquicas e abstratas dos dados, o que as torna especialmente eficazes para lidar com tarefas complexas e grande volumes de dados. As redes neurais profundas são treinadas usando o aprendizado supervisionado, em que o treinamento envolve ajustar os pesos das conexões entre os neurônios para minimizar a diferença entre as previsões do modelo e as saídas desejadas.

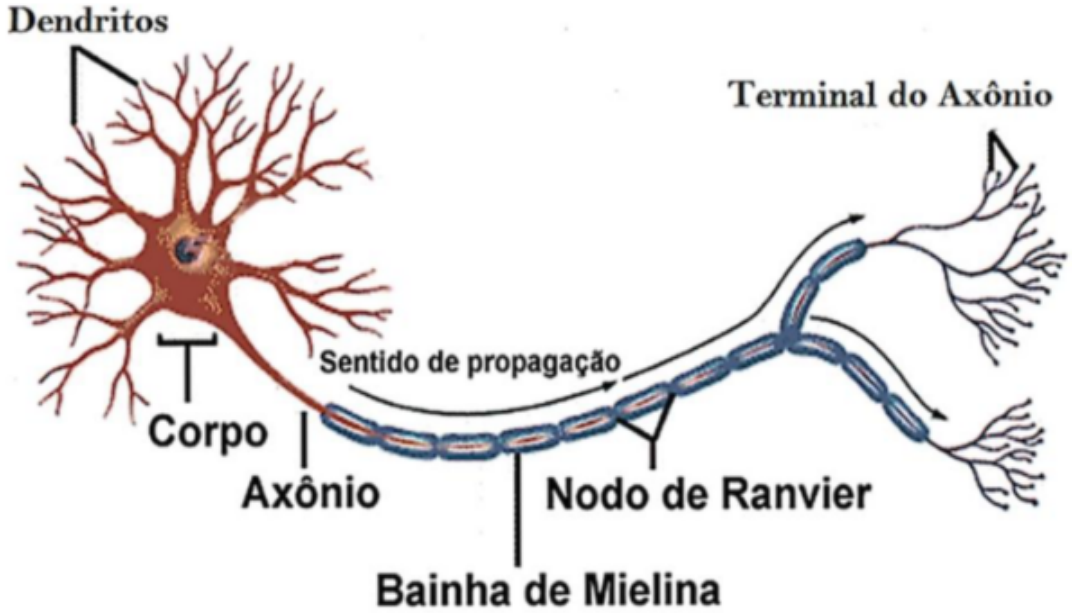
4.4 Perceptron

O perceptron, também chamado de neurônio artificial, é um tipo de algoritmo de aprendizado supervisionado de uma única camada (HAYKIN, 2001). É uma forma simplificada de uma rede neural artificial e serve como bloco de construção fundamental para arquiteturas mais complexas de redes neurais. O perceptron é especificamente projetado para realizar tarefas de classificação binária. Seu treinamento envolve ajustar os pesos com base nos erros de classificação. O algoritmo busca minimizar a diferença entre as previsões do perceptron e as saídas desejadas no conjunto de treinamento.

4.4.1 Neurônio Artificial

A ideia por trás do perceptron é simular de forma simplificada o funcionamento de um neurônio humano (HAYKIN, 2001). O perceptron é um tipo de modelo de rede neural que foi desenvolvido com base na estrutura e no funcionamento dos neurônios biológicos que são unidades fundamentais do sistema nervoso.

Figura 11 – Neurônio biológico

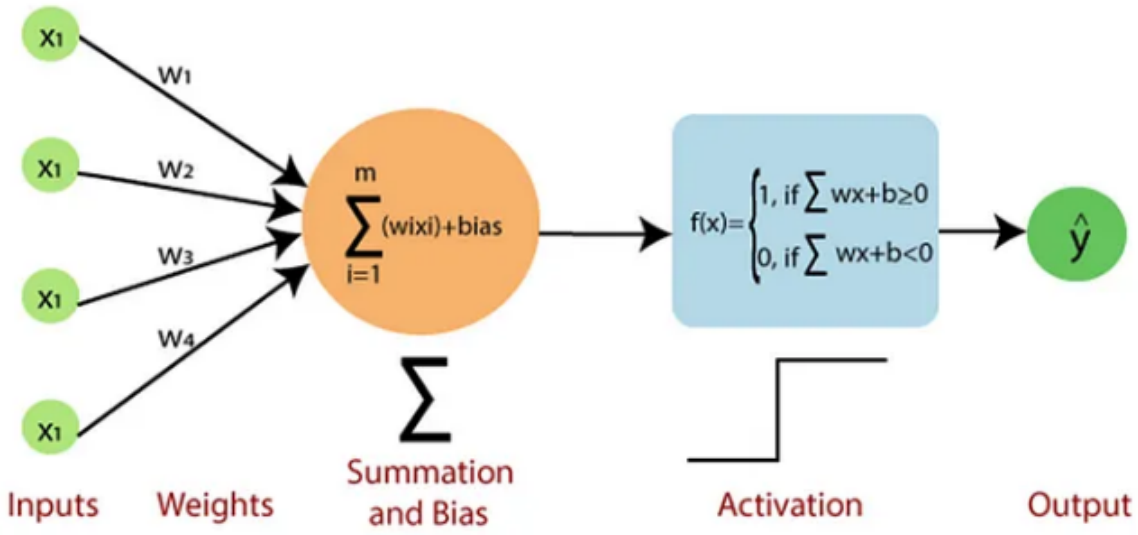


Fonte: BEAR e CONNORS, 2008

O neurônio biológico recebe sinais de entrada através de dendritos, processa esses sinais no corpo celular e, se a soma desses atingir um certo limiar, o neurônio dispara um impulso elétrico ao longo do axônio para transmitir um sinal de saída (BEAR e CONNORS, 2008). O perceptron é uma tentativa de modelar esse processo. Ele recebe múltiplas entradas ponderadas, soma essas entradas ponderadas, e se a soma ultrapassar um determinado limiar, o perceptron produz uma saída (HAYKIN, 2001). No entanto, é importante notar que o perceptron é uma simplificação significativa do funcionamento dos

neurônios biológicos. Neurônios reais têm uma complexidade muito maior, com características como sinapses variáveis, potenciais de ações mais complexas, e muitas interações com outros neurônios.

Figura 12 – Perceptron



Fonte: KILIÇ, 2023

Em relação ao perceptron apresentado na Figura 12, as etapas são dadas por:

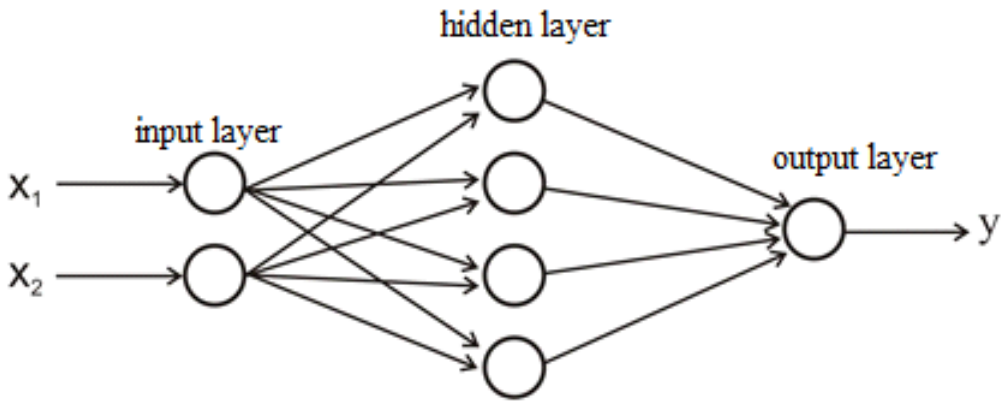
- Inputs (entradas): Representam os valores de entrada fornecidos ao perceptron. Cada entrada é multiplicada pelo peso correspondente durante o processo.
- Weights (pesos): Cada entrada tem um peso associado. Os pesos são parâmetros ajustáveis que indicam a importância relativa de cada
- Summation (soma ponderada): A soma ponderada é obtida somando-se os produtos das entradas pelos pesos.
- Bias (viés): O viés é um termo adicional adicionado à soma ponderada antes de ser passada por uma função de ativação. Introduz uma certa flexibilidade à operação do perceptron, permitindo que ele gere uma saída mesmo quando todas as entradas são nulas.
- Threshold Activation (Ativação limiar): A função de ativação decide se o perceptron deve gerar uma saída com base na soma ponderada mais o viés. A função de ativação padrão é de limiar, isto é, função degrau que retorna 1 se a entrada por maior que um de limiar e 0 caso contrário.
- Output (saída): A saída do perceptron é determinada pela função de ativação aplicada à soma ponderada mais o viés.

Esse tipo de algoritmo tem limitações e só pode aprender a separar conjuntos linearmente separáveis, ou seja, conjuntos de dados que podem ser divididos por uma linha reta em um espaço multidimensional. Para lidar com conjuntos de dados mais complexos, foram desenvolvidas arquiteturas de redes neurais mais profundas e sofisticadas, como é o caso das redes neurais recorrentes para dados sequenciais.

4.4.2 Algoritmo de Retropropagação Através do Tempo

O algoritmo de retropropagação, também conhecido como backpropagation, é um método utilizado para treinar redes neurais artificiais, ajustando os pesos das conexões entre os neurônios com base no erro da saída em comparação com a saída desejada (SUTSKEVER, 2013). Esse processo é fundamental para o aprendizado supervisionado em redes neurais. A retropropagação utiliza o gradiente descendente como técnica de otimização para minimizar a função perda ou erro da rede.

Figura 13 – Algoritmo de retropropagação



Fonte: YILDIZ, 2015

A retropropagação através do tempo é uma extensão da retropropagação em RNNs para lidar com dados sequenciais (SUTSKEVER, 2013). Nas RNNs, a retropropagação é estendida ao longo do tempo para considerar a dependência temporal das sequências. O BPTT desenrola a rede ao longo do tempo, criando uma versão "esticada" da RNN para que a retropropagação possa ocorrer em sequência, tratando cada passo de tempo como uma camada adicional. Seu objetivo é treinar a rede para aprender padrões temporais e dependência em sequências temporais.

4.4.3 Camadas

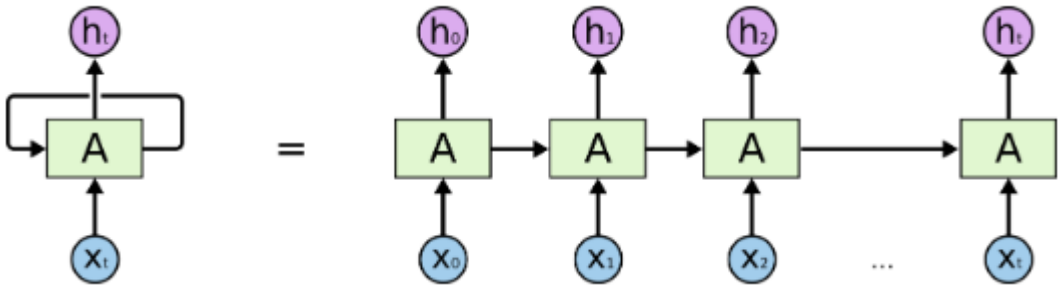
As camadas em redes neurais referem-se aos diferentes níveis de processamento ou transformação pelos quais os dados passam ao serem propagados pela rede. Cada camada é composta por um conjunto de unidades chamadas neurônios (perceptrons). Essas camadas são organizadas sequencialmente para formar a arquitetura da rede neural (GOODFELLOW; BENGIO; COURVILLE, 2016). Cada tipo de camada desempenha uma função específica na extração e transformação de características dos dados. No caso de dados sequenciais, as principais categorias de camadas incluem:

- Camada de Entrada: Primeira camada da rede neural. Cada neurônio nesta camada representa uma característica de entrada e não realiza processamento algum, ou seja, apenas recebe as entradas e as transmite para a próxima camada.
- Camadas Ocultas: Camadas intermediárias entre a camada de entrada e saída. Cada neurônio em uma camada oculta recebe entradas, realiza operações ponderadas e aplica uma função de ativação, sendo responsáveis por aprender representações mais complexas e abstratas dos dados.
- Camada de Saída: A camada de saída é a última camada da rede neural. Geralmente fornece as saídas finais do modelo, em que o número de neurônios nesta camada é determinado pelo tipo de tarefa que a rede está realizando.
- Camadas Densas: As camadas densas são aquelas em que cada neurônio recebe entradas de todos os neurônios da camada anterior. Elas são responsáveis por aprender relações complexas nos dados.
- Camadas Recorrentes: Usadas em redes neurais recorrentes para lidar com dados sequenciais. Permitem que as informações sejam propagadas ao longo de sequências, mantendo uma espécie de "memória" temporal.

4.5 Redes Neurais Recorrentes (RNN)

As redes neurais recorrentes são um tipo de arquitetura de rede neural projetada para lidar com dados sequenciais ou temporais (KILIÇ, 2023). Ao contrário de redes neurais tradicionais, as RNNs possuem conexões retroativas, o que permite que elas mantenham um estado interno ou memória. Isso as torna particularmente eficazes para lidar com sequência de dados como, por exemplo, séries temporais.

Figura 14 – Rede neural recorrente



Fonte: OLAH, 2015

Redes neurais recorrentes podem ser interpretadas como loops. Embora seja uma representação simplificada e conceitual, ele reflete a ideia central por trás do funcionamento das RNNs, que é a capacidade de processar sequências temporais mantendo um estado interno que é atualizado em cada passo de tempo (OLAH, 2015).

4.5.1 Problema do Gradiente

O problema do gradiente em RNNs refere-se a dificuldades encontradas durante o treinamento dessas redes devido à propagação de gradientes ao longo de dependências temporais. Especificamente, o problema do gradiente pode se manifestar de duas maneiras principais: desvanecimento do gradiente e explosão do gradiente (BROWNLEE, 2020).

Desvanecimento do Gradiente

O desvanecimento do gradiente ocorre quando os gradientes das funções de perda em relação aos pesos da rede se tornam muito pequenos à medida que são propagados de volta no tempo durante o treinamento. Em RNNs, onde a retropropagação é estendida ao longo do tempo, o gradiente pode diminuir exponencialmente à medida que se move para trás nas dependências temporais. Isso resulta em atualizações de peso insignificantes ou quase nulo, o que dificuldade a aprendizagem de dependências temporais de longo prazo.

Explosão do Gradiente

A explosão do gradiente é o oposto do desvanecimento. Ela ocorre quando os gradientes se tornam muito grandes à medida que são propagados de volta no tempo. Isso pode levar a atualização extremamente grandes, o que pode fazer com que os pesos cresçam exponencialmente durante o treinamento. A explosão do gradiente pode levar a problemas de convergência, instabilidade numérica e dificuldade em treinar a rede de maneira eficaz.

4.5.2 Época

Uma época (epoch) refere-se a uma única passagem completa por todo o conjunto de treinamento durante o processo de treinamento de um modelo. Durante uma época, o modelo faz previsões para cada exemplo no conjunto de treinamento de um modelo, calcula a perda (diferença entre as previsões e os rótulos reais), e atualiza os pesos dos parâmetros do modelo com base nessas previsões e na função de perda. De acordo com BROWNLEE (2020), o objetivo do treinamento é ajustar esses pesos de forma que o modelo possa fazer previsões mais precisas e generalizar bem para dados não vistos. O número ideal de épocas pode variar dependendo do problema, do tamanho do conjunto de dados e da complexidade do modelo.

4.5.3 Sobreajuste

O sobreajuste (overfitting) é um fenômeno em que o modelo se ajusta excessivamente aos dados de treinamento, capturando padrões específicos desse conjunto de dados, mas falhando em generalizar para dados não vistos (BROWNLEE, 2020). Isso ocorre quando a capacidade do modelo é muito alta em relação à complexidade do problema, resultando em um ajuste excessivo aos detalhes de situações incomuns dos dados de treinamento. Em contexto de RNNs, o overfitting pode ser especialmente desafiador devido à natureza das dependências temporais em sequências.

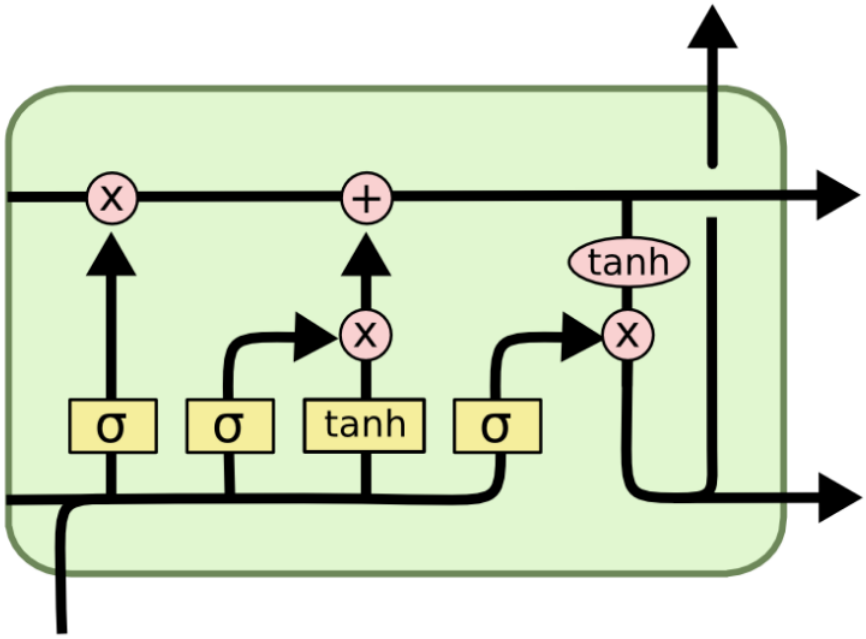
4.6 Memórias de Curto-Longo Prazo (LSTM)

As memórias de curto-longo prazo são um tipo de arquitetura específica de unidade recorrente usada em redes neurais, especialmente projetadas para lidar com problemas em que é crucial considerar dependências temporais de longo prazo (GREFF, 2016). As LSTMs foram introduzidas para superar as limitações das unidades recorrentes tradicionais, que muitas vezes têm dificuldade em lidar com gradiente durante o treinamento em sequências temporais longas. Elas são amplamente utilizadas em tarefas que envolvem dados sequenciais, como séries temporais, devido à sua capacidade de aprender e lembrar dependências temporais de longo prazo.

4.6.1 Célula de Estado (Kernel)

As LSTMs possuem uma "célula de estado" interna, que atua como uma espécie de memória de longo prazo (BROWNLEE, 2020). Essa célula é projetada para armazenar informações relevantes por longos períodos, permitindo que a LSTM aprenda dependências temporais mais duradouras.

Figura 15 – Célula de estado



Fonte: OLAH, 2015

Para controlar o fluxo de informações na célula de estado, as LSTMs utilizam três portões principais (BROWNLEE, 2020):

- Portão de Entrada (Input Gate): Controla quanto da nova informação deve ser adicionada à célula de estado. A operação do portão de entrada é controlada por uma função de ativação, geralmente a função sigmoide. Essa função produz valores entre 0 e 1, agindo como um "interruptor" que determina quanto da nova informação anterior que deve ser adicionado à célula de estado.

- Portão de Esquecimento (Forget Gate): Regula quanto da informação atual na célula de estado deve ser esquecido. Ele utiliza a função de ativação sigmoide para produzir valores entre 0 e 1, indicando a quantidade de informação anterior que deve ser mantida (próxima de 1) ou esquecida (próxima de 0), e a informação esquecida é calculada com a tangente hiperbólica para escalonar os valores entre -1 e 1.

- Portão de Saída (Output Gate): Determina quanto da informação na célula de estado deve ser exposto como saída. Assim como os outros portões, o portão de saída também utiliza a função sigmoide para decidir quanto da informação na célula de estado deve ser exposto, mas a saída da célula de estado passa pela tangente hiperbólica para escalonar os valores entre -1 e 1.

5 Aplicação e Avaliação dos Modelos

Esse capítulo apresenta todo o desenvolvimento da construção e análise comparativa dos modelos, incluindo informações a respeito da origem e finalidade dos dados.

5.1 Preparação dos Dados

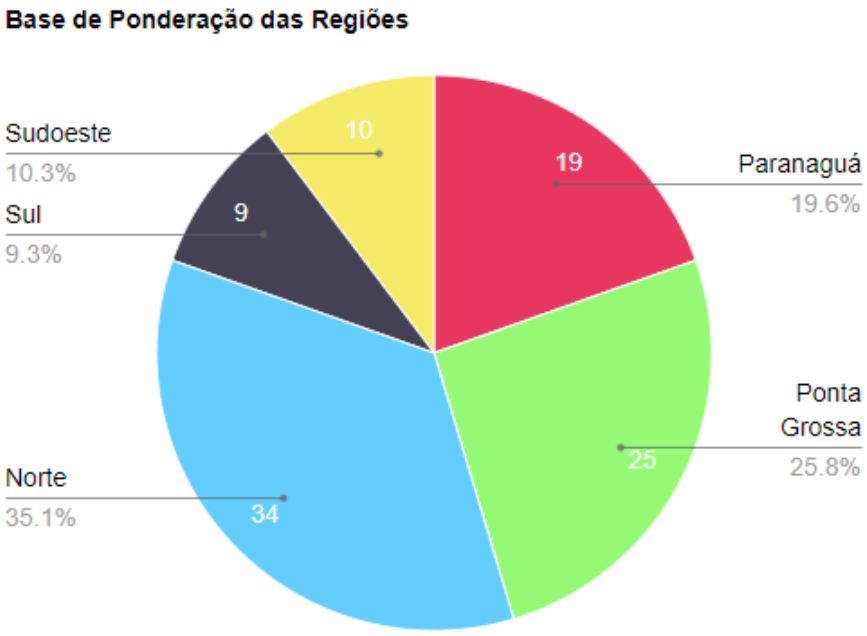
Fonte dos Dados

Os dados do preço da soja foram obtidos do Centro de Estudos Avançados em Economia Aplicada (CEPEA). Foi coletada uma amostra do preço da soja no estado do Paraná de janeiro de 2012 até dezembro de 2022 para dados mensais baseados no primeiro dia de cada mês, seguindo uma amostragem sistemática secundária, obtidos em uma planilha Excel.

Indicador da Soja

Os dados são baseados em um indicador que apresenta preços tanto em reais quanto dólares da soja, disponíveis desde agosto de 1997 seguindo uma periodicidade diária. A unidade de medida da soja é de uma saca (60kg). A base de ponderação das regiões é representada na Figura 16. Por fim, a entrega é referente à negociações no mercado físico de lotes.

Figura 16 – Base de ponderação das regiões



Fonte: Elaborado pelo autor

Processo ETL

Os dados foram extraídos da planilha e carregados com a biblioteca Pandas do Python. Os dados não ficaram adequados para uso no dataframe e, sendo assim, é preciso fazer alterações para que o dataframe seja convertido para uma série temporal.

Tabela 1 – Dados extraídos do Excel

	Soja	**Unnamed: 1**	**Unnamed: 2**
0	Nota	por saca de 60 kg, descontado o Prazo de Pagam...	NaN
1	Fonte	Cepea	NaN
2	Data	À Vista R$	À Vista US$
3	01/2012	46,80	26,18
4	02/2012	47,06	27,41
...	...	...	...
130	08/2022	181,86	35,34
131	09/2022	181,72	34,74
132	10/2022	179,71	34,24
133	11/2022	182,44	34,55
134	12/2022	177,11	34,03

O primeiro passo é fatiar o dataframe da quarta linha até a final, removendo as três primeiras linhas no processo. Essas linhas são removidas porque são irrelevantes para análise. Após isso, remover a segunda coluna que envolve o preço da soja em reais. Alterar o nome das coluna do cabeçalho para data e preço, respectivamente. Substituir as vírgulas por ponto e converter os dados do preço para decimal. Por fim, converter o índice do dataframe para série temporal mensal iniciando no primeiro mês. Ao fazer a filtragem, limpeza e transformação dos dados, a série temporal já está pronta para análise.

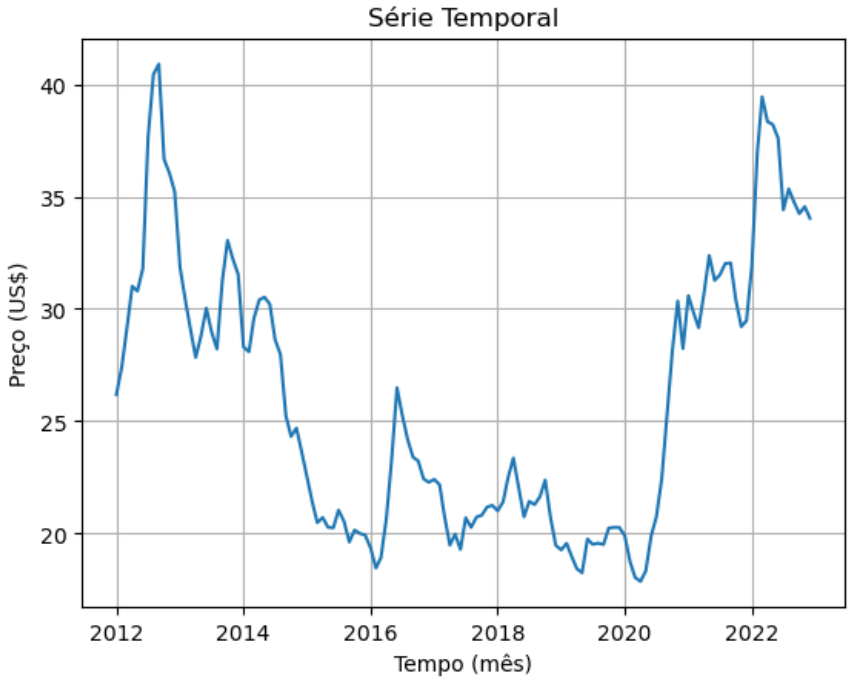
Tabela 2 – Dados da série temporal

Data	**Preço**
2012-01-01	26.18
2012-02-01	27.41
2012-03-01	29.11
2012-04-01	31.01
2012-05-01	30.79
...	...
2022-08-01	35.34
2022-09-01	34.74
2022-10-01	34.24
2022-11-01	34.55
2022-12-01	34.03

5.2 Análise da Série Temporal

Agora é possível fazer análise da série temporal usando os dados do dataframe criado anteriormente. Uma boa prática inicial é dar uma olhada no gráfico da série temporal.

Figura 17 – Série temporal



Fonte: Elaborado pelo autor

5.2.1 Estatística Descritiva

As informações descritivas da série temporal podem ser calculadas através da análise exploratória de dados, bastante útil para ter uma visão geral dos dados.

Tabela 3 – Análise exploratória dos preços da soja em dólar por saca

Estatística	**Dados**
Tamanho da Amostra	132
Média Aritmética	25.91
Desvio Padrão	6.11
Valor Mínimo	17.87
Primeiro Quartil	20.53
Segundo Quartil	23.92
Terceiro Quartil	30.54
Valor Máximo	40.90

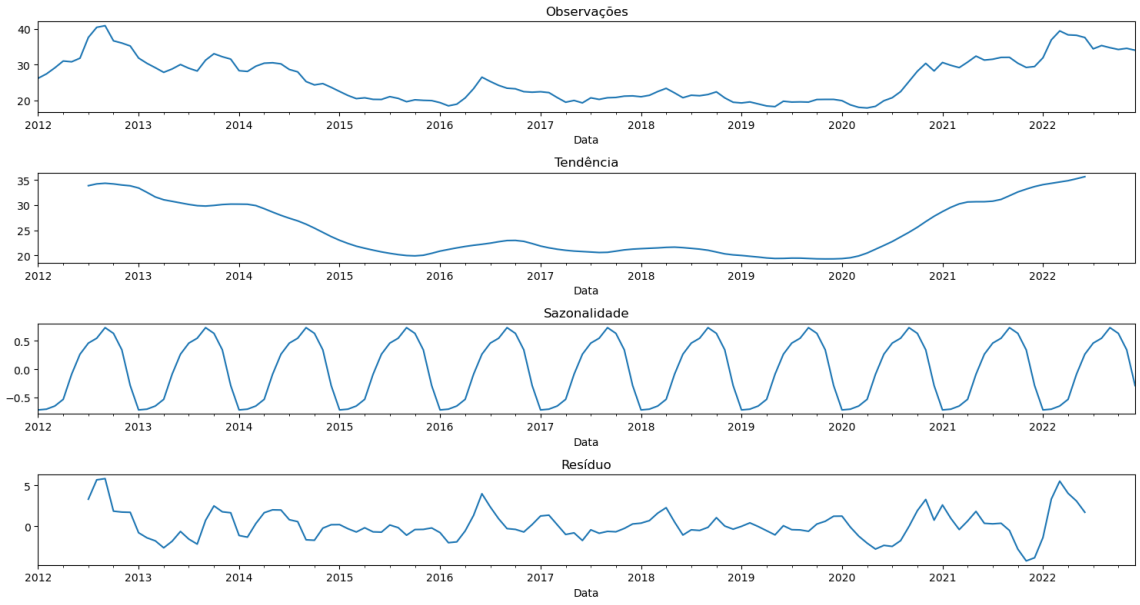
Informações complementares podem ser extraídas da estatística descritiva.

- O tamanho da amostra indica uma série temporal mensal com 11 anos.
- A média está próxima à mediana (segundo quartil), sugerindo uma distribuição aproximadamente simétrica dos preços.
- O desvio padrão indica uma variabilidade moderada do preço em torno da média.
- A amplitude dos preços é de aproximadamente 23 dólares.
- A dispersão interquartil dos preços é de aproximadamente 10 dólares.

5.2.2 Decomposição

Após analisar a estatística descritiva da série, o próximo passo é analisar suas características. Para isso é feita uma decomposição aditiva da série temporal.

Figura 18 – Decomposição aditiva da série temporal



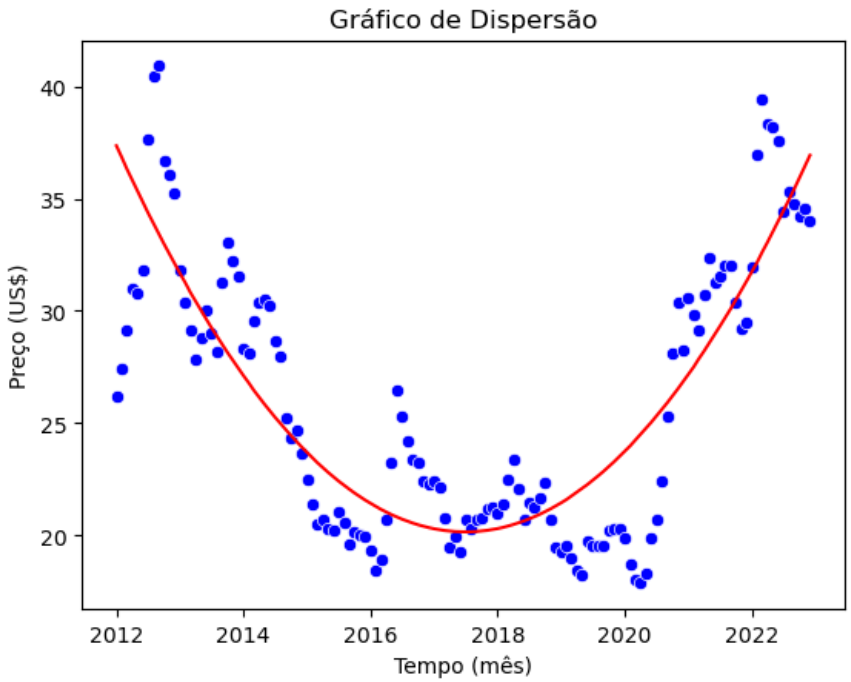
Fonte: Elaborado pelo autor

A figura 18 mostra a decomposição da série temporal em tendência, sazonalidade e resíduo, além dos valores observados ao longo do tempo. Observe que a tendência parece cair com o tempo até um certo ano, diminuir sua oscilação em um certo intervalo e então subir. Já no caso do componente sazonal é possível enxergar curvas que se repetem ao longo do tempo. Esse comportamento pode estar relacionado a ciclos sazonais. Será feita uma análise mais técnica para cada decomposição de modo a estudar melhor as propriedades dessa série temporal.

5.2.2.1 Tendência

A tendência permite analisar o comportamento da série ao longo do tempo.

Figura 19 – Dispersão dos dados



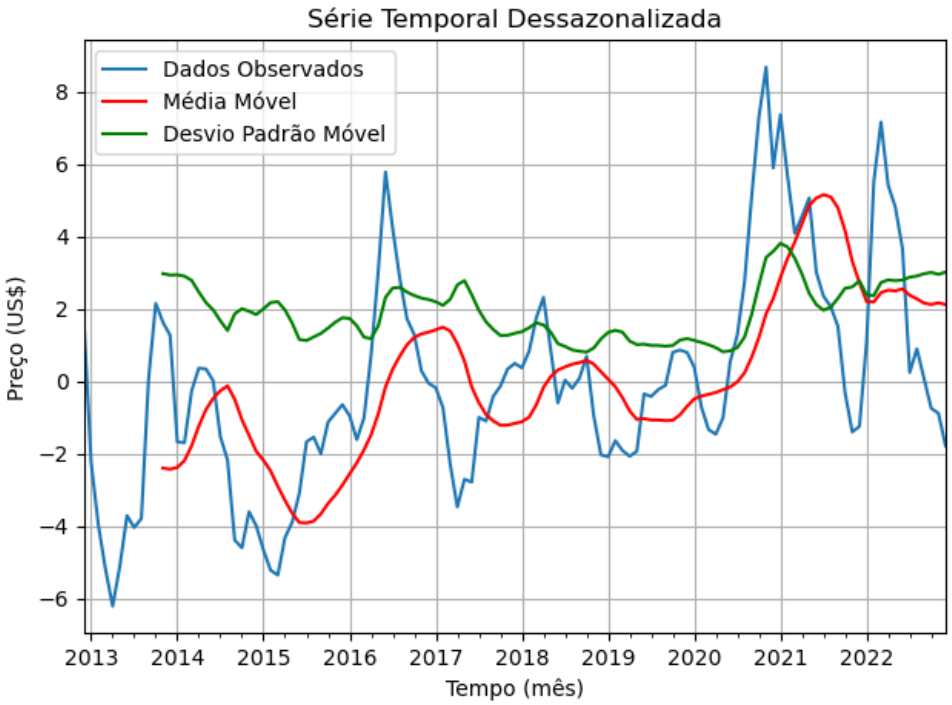
Fonte: Elaborado pelo autor

A série temporal em questão aparenta ter uma tendência quadrática. Os dados tendem a decrescer inicialmente, oscilar em um curto período e então tendem a aumentar com o tempo. Esse comportamento dá origem à uma parábola convexa. O modelo quadrático faz parte do teste de estacionariedade, porém não é adequado abordá-lo nesse estudo por causa dos dados serem parte de uma amostra e não população. Sendo assim, o ideal será remover a tendência da série temporal.

5.2.2.2 Sazonalidade

Analisando graficamente o componente sazonal da Figura 18 é possível identificar periodicidade sazonal anual na série temporal. Isso implica que a sazonalidade está relacionada a eventos que acontecem anualmente. Para verificar se, de fato, a série apresenta sazonalidade, outras abordagens podem ser feitas como dessazonalizar a série temporal e analisar a volatilidade anual do preço. Elas permitem análises mais aprofundadas na detecção de ciclos sazonais, aumentando assim a probabilidade da série temporal ter sazonalidade anual.

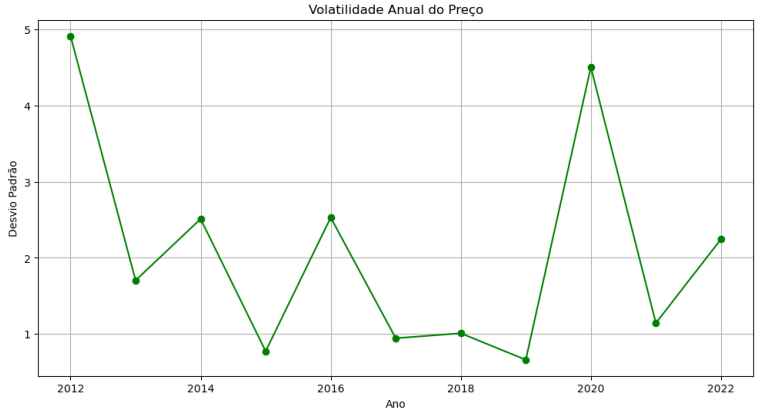
Figura 20 – Série temporal dessazonalizada



Fonte: Elaborado pelo autor

Um comportamento comum em séries sazonais é a oscilação do desvio padrão móvel ser menor que a média móvel, que por sua vez ser menor que a da série. Esse é um indicativo de sazonalidade na série temporal.

Figura 21 – Volatilidade anual do preço

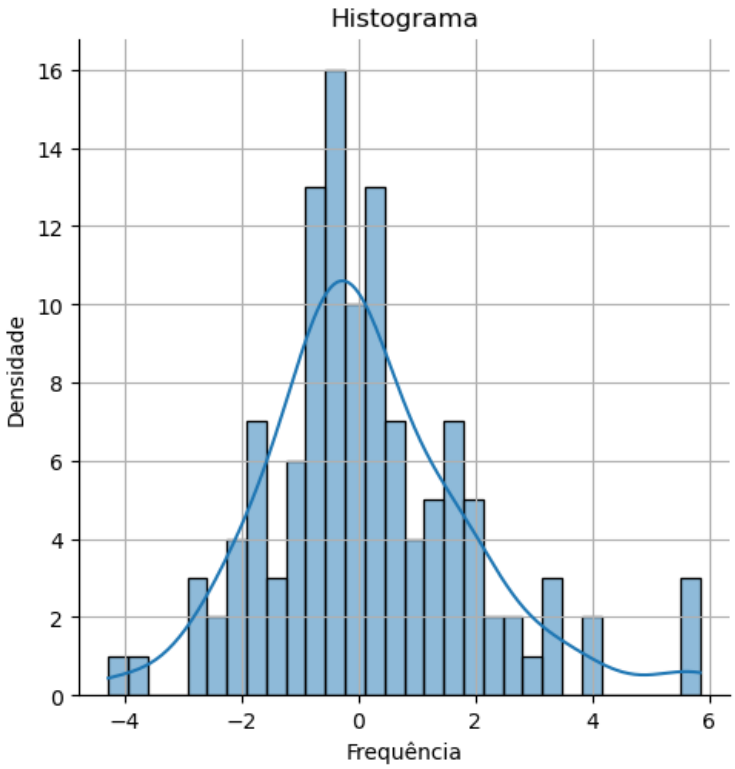


Fonte: Elaborado pelo autor

Ao analisar o desvio padrão anual da série temporal observa-se um comportamento de queda e subida continuamente. Através desse comportamento é esperado que a volatilidade, terminologia comum em econometria, caia no ano de 2023 seguindo o padrão volátil da série temporal. Esse é outro indicativo de sazonalidade na série temporal.

5.2.2.3 Resíduo

Figura 22 – Histograma dos resíduos



Fonte: Elaborado pelo autor

É importante verificar se a série temporal apresenta resíduos normalmente distribuídos. Para isso uma das opções é utilizar o teste de Jarque-Bera.

Tabela 4 – Teste de Jarque-Bera para os resíduos da série temporal

Série Temporal	**Estatística de Teste**	**Valor-p**	**Assimetria**	**Curtose**
Não Transformada	18.25	0.0001	0.71	4.28
Transformada	3.18	0.2035	0.38	3.25

Os resíduos da série temporal não são normalmente distribuídos com base no teste de Jarque-Bera. Uma das formas de corrigir esse problema é transformar a série temporal em log (logaritmo natural). Essa técnica reduz a dispersão dos resíduos e passa a trabalhar com variação do preço ao invés do preço nominal. Isso é bastante comum no desenvolvimento de modelos estatísticos para problemas com contexto econômico. Observa-se também que os resíduos da série temporal apresentam uma distribuição assimétrica à direita e leptocúrtica em relação ao grau de achatamento da curva, cujas mesmas propriedades são atribuídas à série temporal transformada. Com base no teste, o problema de ausência de normalidade nos resíduos da série temporal é corrigido ao fazer a transformação. Esse processo é importante no desenvolvimento do modelo SARIMA.

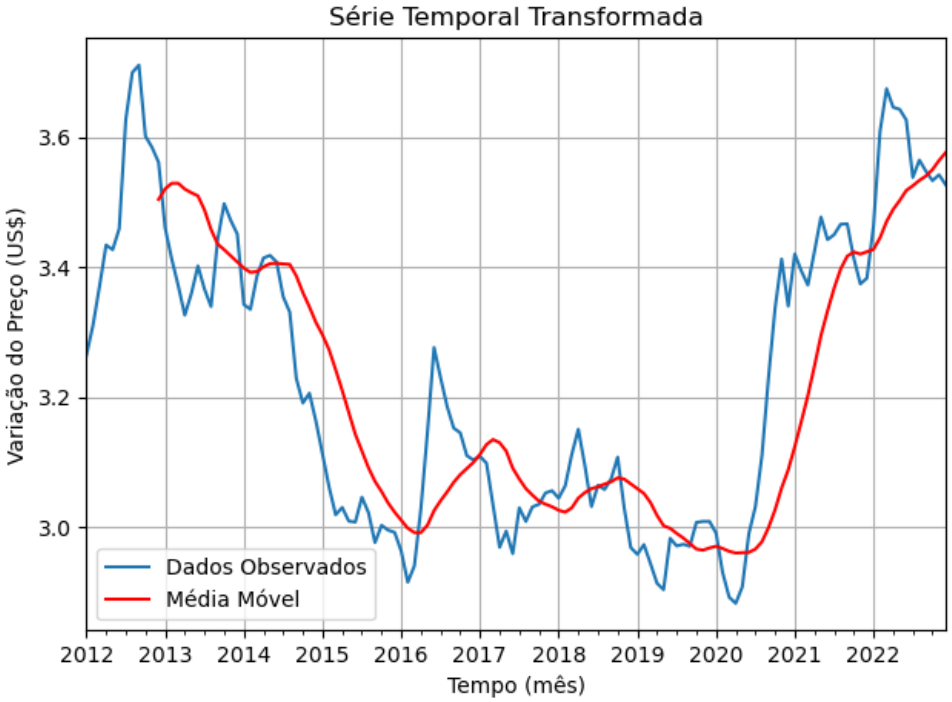
5.3 Construção do Modelo SARIMA

No desenvolvimento do modelo SARIMA é preciso determinar os termos autorregressivos e de média móvel não sazonais e sazonais, além de verificar quando a série temporal e seu componente sazonal são estacionários.

5.3.1 Estacionariedade

Para construir o modelo SARIMA a série é primeiramente transformada em log.

Figura 23 – Série temporal em log



Fonte: Elaborado pelo autor

Agora é feito o teste de Dickey-Fuller aumentado para verificar se a série é estacionária em nível. São analisados os modelos sem constante, com constante e com constante e tendência.

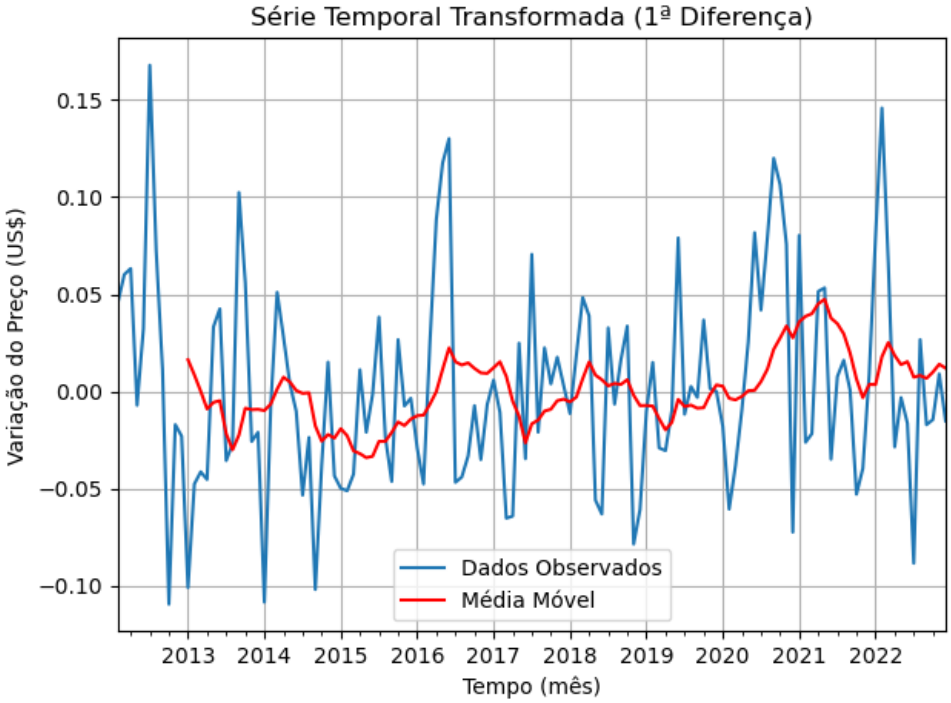
Tabela 5 – Teste ADF em nível

Modelo	Estatística do Teste	Valor-p	AIC	BIC
Sem Constante	0.0819	0.7109	-386.53	**-380.99**
Com Constante	-1.7534	0.4038	-387.10	-378.79
Com Constante e Tendência	-1.6963	0.7525	**-387.34**	-376.26

Observa-se que os critérios de informação divergem em relação ao modelo mais adequado. Independentemente do modelo escolhido para esse cenário, a série temporal

não é estacionária em nível com base no valor-p. Agora a série em log é diferenciada para tentar torná-la estacionária.

Figura 24 – Série temporal na primeira diferença



Fonte: Elaborado pelo autor

Novamente é realizado o teste ADF para verificar a presença de estacionariedade na série, nesse caso na primeira diferença.

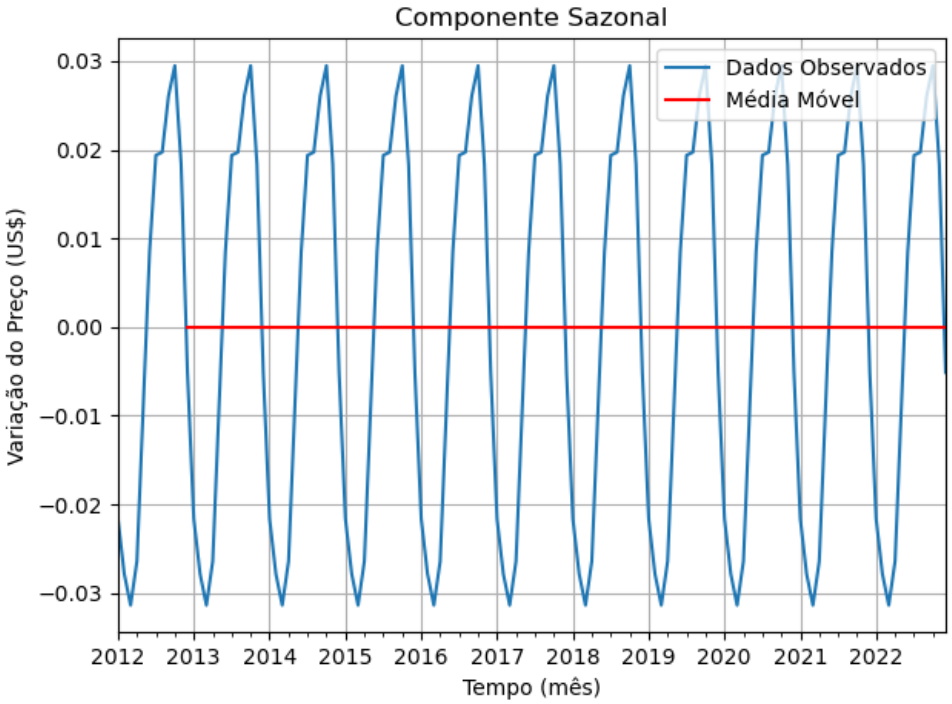
Tabela 6 – Teste ADF na primeira diferença

Modelo	Estatística do Teste	Valor-p	AIC	BIC
Sem Constante	-7.7249	$139 \cdot 10^{-9}$	**-384.51**	**-381.75**
Com Constante	-7.6980	$136 \cdot 10^{-9}$	-382.56	-377.04
Com Constante e Tendência	-7.7268	$306 \cdot 10^{-9}$	-381.81	-373.52

Ambos os critérios de informação AIC e BIC indicam o modelo sem constante como o mais adequado, o qual apresenta valor-p abaixo de 0.05. Esse resultado indica que a série temporal é estacionária na primeira diferença sem constante.

Ainda é necessário verificar se o componente sazonal é estacionário.

Figura 25 – Componente sazonal

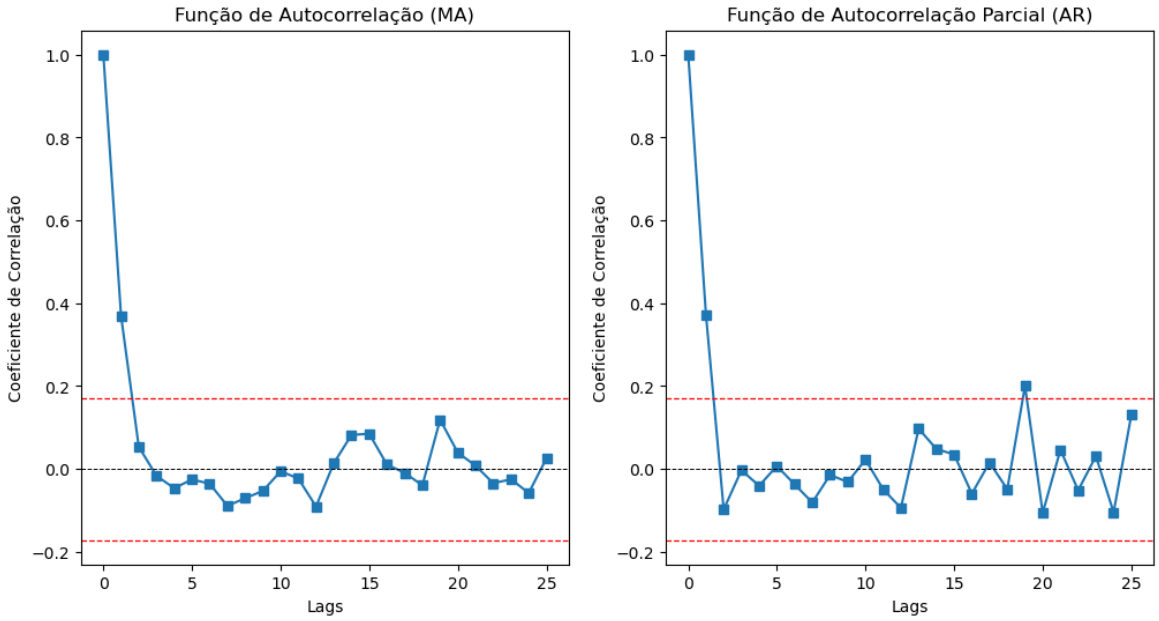


Fonte: Elaborado pelo autor

É visível que o componente sazonal é estacionário em nível sem constante, logo não há necessidade de fazer o teste ADF sazonal. Observe que os dados observados oscilam na reta formada pela variação de preço nula. As propriedades estatísticas do componente sazonal não variam com o tempo, como por exemplo medidas de tendência central e dispersão. Isso pode ser visto na representação da média móvel que é sempre nula com o tempo. Sendo assim, já são informadas as ordens de integração não sazonais e sazonais para o modelo. O próximo passo é analisar as funções de autocorrelação para determinar a ordem dos termos autorregressivos e de média móvel. Os lags que tiverem coeficientes significativos em sequência serão levados em consideração como termos máximos autorregressivos e de média móvel. Para isso são analisadas as funções de autocorrelação em relação aos termos de média móvel e as funções de autocorrelações parciais em relação aos termos autorregressivos.

5.3.2 Modelagem SARIMA

Figura 26 – Funções de autocorrelação



Fonte: Elaborado pelo autor

Observa-se que os dois primeiros lags das funções de autocorrelação estão fora do intervalo de confiança. Isso é um indicativo para determinar o modelo SARIMA mais adequado com até 2 termos autorregressivos e de média móvel. O mesmo será levado em consideração para os termos sazonais. Portanto, o melhor modelo ARIMA é estimado com base nesses resultados através dos menores critérios de informação AIC, BIC e HQIC.

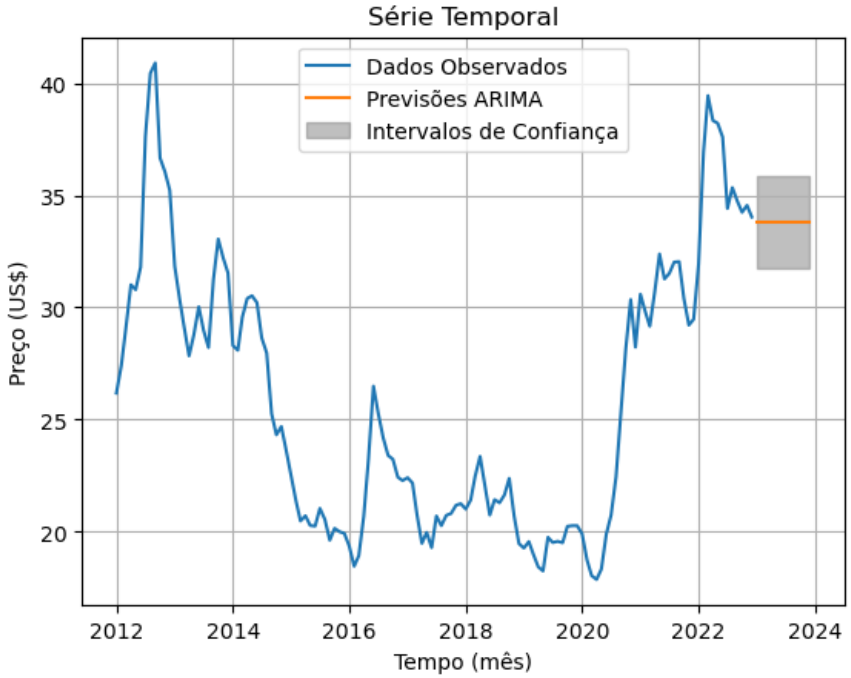
Tabela 7 – Estimação do melhor modelo ARIMA

Modelo	**AIC**	**BIC**	**HQIC**
ARIMA(0,1,0)(1,0,1)[12]	-397.985	-389.359	-394.480
ARIMA(0,1,0)(0,0,0)[12]	-400.127	-397.252	-398.958
ARIMA(1,1,0)(1,0,0)[12]	-417.826	-409.200	-414.321
ARIMA(0,1,1)(0,0,1)[12]	**-418.743**	-410.118	-415.238
ARIMA(0,1,1)(0,0,0)[12]	-418.028	**-412.277**	**-415.691**
ARIMA(0,1,1)(1,0,0)[12]	-418.058	-409.433	-414.553
ARIMA(1,1,0)(0,0,0)[12]	-418.480	-411.635	-415.049

O melhor modelo com base nos critérios de informação BIC e HQIC é dado por ARIMA(0,1,1)(0,0,0)[12] enquanto que o critério AIC seleciona o mesmo modelo com adição de um termo de média móvel sazonal, onde 12 representa o número de defasagens (lags). O modelo escolhido para previsão será dado por ARIMA(0,1,1).

Com base nas análises desse modelo é visto que não serve para previsão da série temporal conforme ilustrado no gráfico abaixo.

Figura 27 – Previsão do modelo ARIMA



Fonte: Elaborado pelo autor

Os dados previstos do modelo ARIMA formam uma reta horizontal com valores repetidos de 33.789529. Esse evento pode ser causado pela evidência de sazonalidade na série temporal no qual o modelo não conseguiu capturar. Sendo assim, é estimado o melhor modelo SARIMA através dos menores critérios de informação retirando os modelos sem termos sazonais.

Tabela 8 – Estimação do melhor modelo SARIMA

Modelo	AIC	BIC	HQIC
ARIMA(0,1,0)(1,0,1)[12]	-397.985	-389.359	-394.480
ARIMA(1,1,0)(1,0,0)[12]	-417.826	-409.200	-414.321
ARIMA(0,1,1)(0,0,1)[12]	**-418.743**	**-410.118**	**-415.238**
ARIMA(0,1,1)(1,0,0)[12]	-418.058	-409.433	-414.553

O melhor modelo estimado é ARIMA(0,1,1)(0,0,1)[12] com base nos critérios de informação. Note que os modelos que apresentaram intercepto ou ausência de termos sazonais foram ignorados na estimação com base na análise da série temporal. Essa filtragem é importante para evitar modelos inadequados para previsão.

É importante analisar os resultados após determinar o melhor modelo para verificar se é adequado para previsão.

Figura 28 – Sumário do modelo SARIMA

Dep. Variable:	Preço	**No. Observations:**	132
Model:	ARIMA(0, 1, 1)x(0, 0, 1, 12)	**Log Likelihood**	212.372
Date:	Sun, 03 Sep 2023	**AIC**	-418.743
Time:	23:53:36	**BIC**	-410.118
Sample:	01-01-2012	**HQIC**	-415.238
	- 12-01-2022		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	0.3918	0.077	5.117	0.000	0.242	0.542
ma.S.L12	-0.1821	0.085	-2.134	0.033	-0.349	-0.015
sigma2	0.0023	0.000	8.595	0.000	0.002	0.003

Ljung-Box (L1) (Q):	0.10	**Jarque-Bera (JB):**	2.30
Prob(Q):	0.76	**Prob(JB):**	0.32
Heteroskedasticity (H):	1.03	**Skew:**	0.29
Prob(H) (two-sided):	0.91	**Kurtosis:**	3.28

Fonte: Elaborado pelo autor

O modelo estimado apresenta três estimadores, sendo eles o termo de média móvel, termo de média móvel sazonal e o resíduo. As médias móveis são analisadas com base no teste Z, na qual todas apresentam p-valor abaixo de 0,05, ou seja, são estatisticamente significativas. O resultado é similar para a variância do resíduo que está estatisticamente próxima de ser nula. Em relação aos testes realizados para o resíduo do modelo, há evidências de que os resíduos são normalmente distribuídos, homocedásticos e descorrelacionados com base nos testes de Jarque-Bera, White e Ljung-Box, respectivamente. Com os três testes satisfeitos é visível que o resíduo do modelo SARIMA estimado seja um ruído branco. Isso é fundamental para garantir a integridade do modelo.

Equação matemática do modelo estimado:

$$\Delta^1 \hat{Z}_t = -0.3918\epsilon_{t-1} + 0.1821\epsilon_{t-1s} + \epsilon_t \tag{5.1}$$

5.3.3 Intervalo de Confiança SARIMA

O intervalo de confiança para o modelo SARIMA(0,1,1)(0,0,1) pode ser calculado com base na distribuição normal com nível de confiança de 95%. A variância do modelo é estimada levando em consideração que os resíduos do modelo são ruído branco. Sendo assim, primeiramente é aplicada a variância de ambos os lados da igualdade.

$$Var(\Delta \hat{Z}_t) = Var(\theta \epsilon_{t-1} + \Theta \epsilon_{t-1s} + \epsilon_t) \tag{5.2}$$

A variância do modelo é dada por $sigma^2$. Note que é possível desmembrar a variância do lado direito da igualdade em uma soma de variâncias. Os coeficientes, por serem constantes, podem ser calculados como produtos fora da variância de modo que estejam ao quadrado.

$$\sigma^2 = \theta^2 \cdot Var(\epsilon_{t-1}) + \Theta^2 \cdot Var(\epsilon_{t-1s}) + Var(\epsilon_t) \tag{5.3}$$

A variância do resíduo é dada por σ_R^2 independentemente da sua defasagem por causa da independência temporal.

$$\sigma^2 = \theta^2 \cdot \sigma_R^2 + \Theta^2 \cdot \sigma_R^2 + \sigma_R^2 \tag{5.4}$$

Por fim, a variância do modelo estimado é da forma:

$$\sigma^2 = \sigma_R^2(1 + \theta^2 + \Theta^2) \tag{5.5}$$

Como o modelo segue uma distribuição normal, a equação do intervalo de confiança para as previsões SARIMA é:

$$IC_{1-\alpha}(x_i) = \left(\hat{x}_i - Z_{1-\alpha/2} \cdot \sqrt{\sigma_R^2(1 + \theta^2 + \Theta^2)}, \hat{x}_i + Z_{1-\alpha/2} \cdot \sqrt{\sigma_R^2(1 + \theta^2 + \Theta^2)}\right) \tag{5.6}$$

Onde:

$\hat{x}_i$ - Dado previsto para $i = 1, 2, \ldots, 12$;

α - Nível de significância;

$Z_{1-\alpha/2}$ - Valor crítico Z;

σ_R^2 - Variância residual;

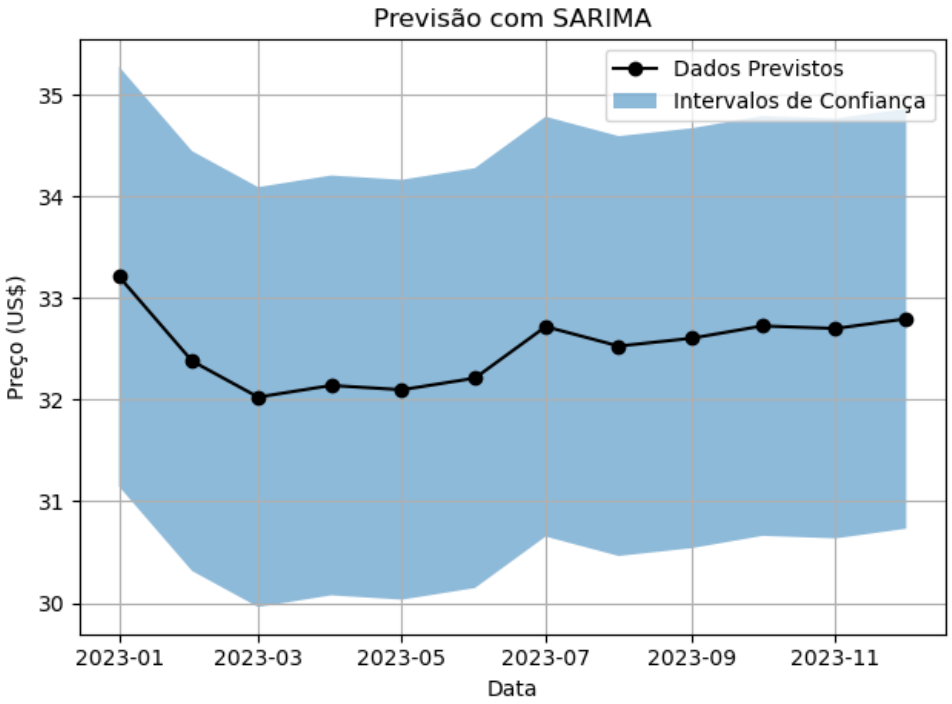
θ - Média móvel não sazonal;

Θ - Média móvel sazonal.

5.3.4 Previsão do Modelo

Com o modelo pronto é possível fazer a previsão para os próximos 12 meses.

Figura 29 – Previsão com SARIMA



Fonte: Elaborado pelo autor

Tabela 9 – Resultados da previsão com SARIMA para 2023

Mês	**Previsão**	**Limite Inferior (95%)**	**Limite Superior (95%)**
Janeiro	33.21	31.15	35.28
Fevereiro	32.38	30.32	34.45
Março	32.03	29.96	34.09
Abril	32.14	30.08	34.21
Maio	32.10	30.03	34.16
Junho	32.21	30.15	34.28
Julho	32.72	30.65	34.78
Agosto	32.53	30.46	34.59
Setembro	32.60	30.54	34.67
Outubro	32.73	30.66	34.79
Novembro	32.70	30.64	34.77
Dezembro	32.80	30.73	34.86

A amplitude do intervalo de confiança é de 4.13 dólares. É observável queda no preço nos primeiros meses de 2023 e pequenas oscilações ao longo do tempo.

5.4 Construção do Modelo LSTM

A abordagem de uma rede neural é diferente de uma regressão. Por exemplo, diferente do modelo SARIMA, não é necessário transformar a série temporal em log para modelar a rede neural.

5.4.1 Dados de Treinamento

O primeiro passo no desenvolvimento da rede neural é preparar os dados da série temporal para uso em um modelo de aprendizado de máquina. Para isso foram criados vetores de sequência de dados para entrada e saída. Cada vetor contém uma etapa de tempo consecutiva da série temporal, e o valor correspondente da próxima etapa de tempo é armazenado no vetor de saída. Após isso é adicionada mais uma dimensão ao vetor de entrada com a etapa de tempo. Seja m o tamanho da amostra ajustado dado por $m = n - 12$, onde n representa o tamanho da amostra, o vetor de entradas para o modelo é definido como:

$$\vec{X} = \begin{bmatrix} x_1 & x_2 & \dots & x_{12} \\ x_2 & x_3 & \dots & x_{13} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m-12} & x_{m-11} & \dots & x_m \end{bmatrix}_{mX12} \tag{5.7}$$

Onde:

x_i - Elemento da amostra onde i representa sua posição.

m - Tamanho da amostra ajustado.

O vetor de saída é similar ao conjunto de dados, porém perde os 12 primeiros dados no processo que faz parte da captura da sazonalidade da série temporal. Ele é definido como:

$$\vec{Y} = \begin{bmatrix} y_{13} & y_{14} & \dots & y_n \end{bmatrix}_{1Xm} \tag{5.8}$$

Onde:

y_i - Elemento da amostra onde i representa sua posição.

m - Tamanho da amostra ajustado.

A rede neural em questão é projetada para dados sequenciais. Note que os índices da série temporal não fazem parte do processo, porém podem ser reutilizados nas previsões da rede neural.

5.4.1.1 Sumário do Modelo

O sumário do modelo LSTM é importante para analisar as camadas e o número de parâmetros referentes à arquitetura do modelo.

Figura 30 – Sumário do modelo LSTM

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 12, 64)            16896

 lstm_1 (LSTM)               (None, 64)                33024

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 49,985
Trainable params: 49,985
Non-trainable params: 0
_________________________________________________________________
```

Fonte: Elaborado pelo autor

Foi escolhida uma forma de entrada de tamanho 12 para tentar capturar a sazonalidade da série temporal durante o treinamento dos dados. Após isso a rede neural LSTM foi projetada com três camadas totalizando quase 50 mil parâmetros. Note que modelos sequencias de aprendizado de máquina aprendem padrões complexos dos dados que não são captados por modelos regressivos. Por trabalhar com alta quantidade de parâmetros, os modelos sequenciais não apresentam equações matemáticas para estimação dos dados.

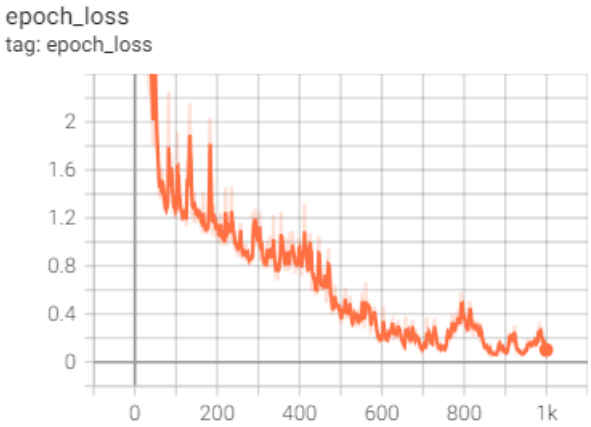
5.4.1.2 Função de Previsão

Os dados previstos do modelo são obtidos através de uma função de previsão. Matematicamente, essa função realiza previsões sequenciais usando o modelo LSTM para extrapolar valores futuros da série temporal. Ela utiliza um loop para iterar sobre as previsões e atualiza a entrada para cada iteração. A lógica do loop é projetada para manter uma janela deslizante de dados de entrada para alimentar o modelo. Posteriromente a função retorna os dados em um formato que é semelhante a uma série temporal.

5.4.2 Modelagem LSTM

O modelo aprendeu o comportamento dos dados através da passagem completa pelo conjunto de dados de treinamento durante seu treinamento, ajustando seus parâmetros por época. Essa passagem foi repetida mil vezes de modo a diminuir a discrepância entre as previsões do modelo e os valores reais com base na perda (no erro quadrático médio). A alta quantidade de épocas é necessária devido ao tamanho da amostra ser muito pequeno para o modelo aprender mais eficientemente. Além disso, foi utilizado o algoritmo de otimização Adam para ajustar iterativamente os pesos das redes com base nos gradientes calculados, com taxa de aprendizado de 0.1%.

Figura 31 – Perdas por época



Fonte: Elaborado pelo autor

A perda foi diminuindo gradativamente conforme as épocas passam. As oscilações mostradas na Figura 31 podem indicar a presença de overfitting no modelo em certos intervalos, especialmente nas épocas iniciais. Ainda assim, a discrepância continuou caindo até atingir 0.0798 que foi o resultado da última época. A condição de discrepância máxima para o modelo estimado foi determinada como 0.1, ou seja, o modelo conseguiu aprender os dados conforme o necessário.

Modelagem das Camadas

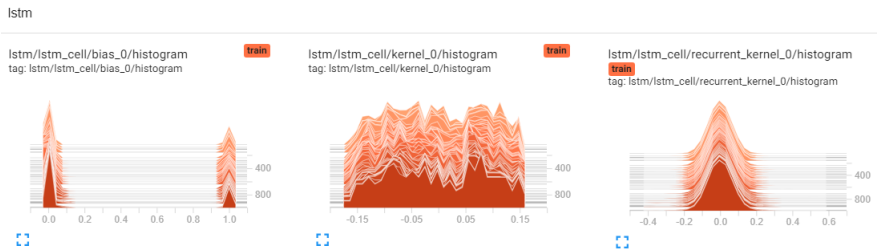
1. A camada LSTM inicial possui 64 neurônios com retorno de sequência de dados utilizando função retificadora para ativação.

2. A segunda camada LSTM também possui 64 neurônios utilizando função retificadora para ativação.

3. A última camada do modelo é uma camada densa com apenas um neurônio sem função de ativação.

Histogramas da Primeira Camada

Figura 32 – Primeira camada LSTM

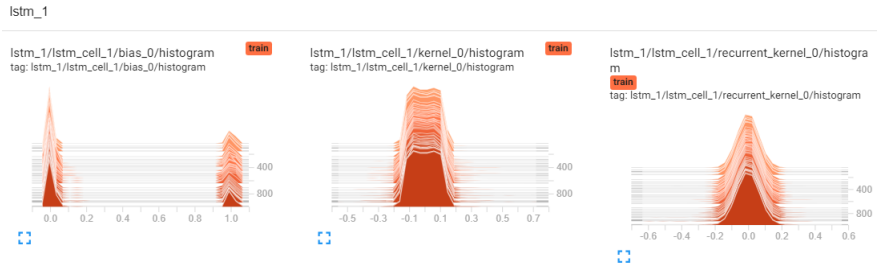


Fonte: Elaborado pelo autor

O viés segue distribuições de média 0 e 1, com ênfase na primeira. O kernel não apresenta um comportamento padrão em sua distribuição cujos valores dos tensores são limitados pelo intervalo $(-0.20, 0.20)$. Por fim o kernel recorrente segue uma distribuição normal dos dados.

Histogramas da Segunda Camada

Figura 33 – Segunda camada LSTM

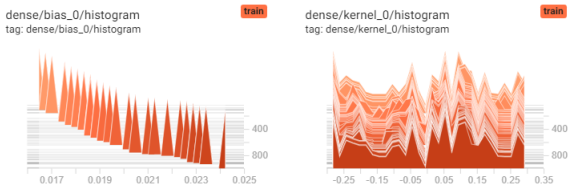


Fonte: Elaborado pelo autor

O viés e o núcleo recorrente seguem comportamentos similares à primeira camada. Já o núcleo da camada apresenta uma distribuição similar à normal com expansão da área no intervalo $(-0.1, 0.1)$

Histogramas da Terceira Camada

Figura 34 – Camada densa



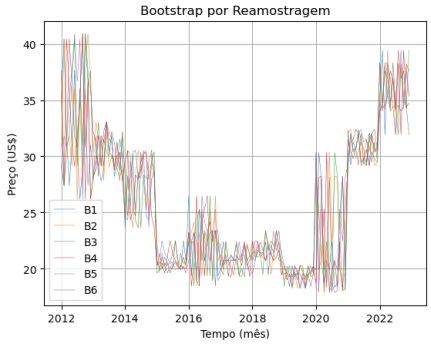
Fonte: Elaborado pelo autor

O viés aumenta vagarosamente ao longo das épocas e o kernel não segue um comportamento padrão com valores dos tensores limitados pelo intervalo $(-0.35, 0.40)$.

5.4.3 Intervalo de Confiança LSTM

Para montar o intervalo de confiança da rede neural são utilizadas técnicas de bootstrap por reamostragem. Para isso oram criadas 6 subamostras de mesmo tamanho da amostra original. Em cada subamostra foram divididos grupos consecutivos de 12 dados e realizou-se permutação aleatória dos dados em cada bloco. Esse processo é importante para manter a dependência temporal dos dados, onde o tamanho dos blocos foi decidido de acordo com evidência de sazonalidade da série temporal.

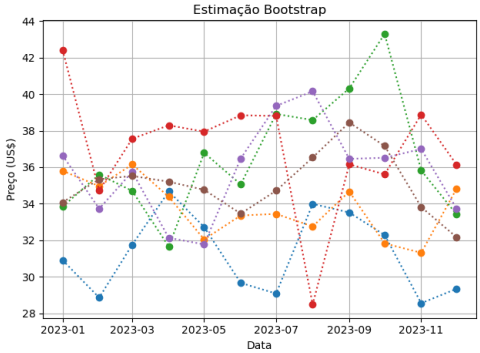
Figura 35 – Técnica de bootstrap por reamostragem



Fonte: Elaborado pelo autor

Os conjuntos $B1, B2, \ldots, B6$ são exemplos de subamostras após o processo de reamostragem. O próximo passo é realizar previsões para os próximos 12 dados fora da amostra usando o modelo LSTM para cada subamostra.

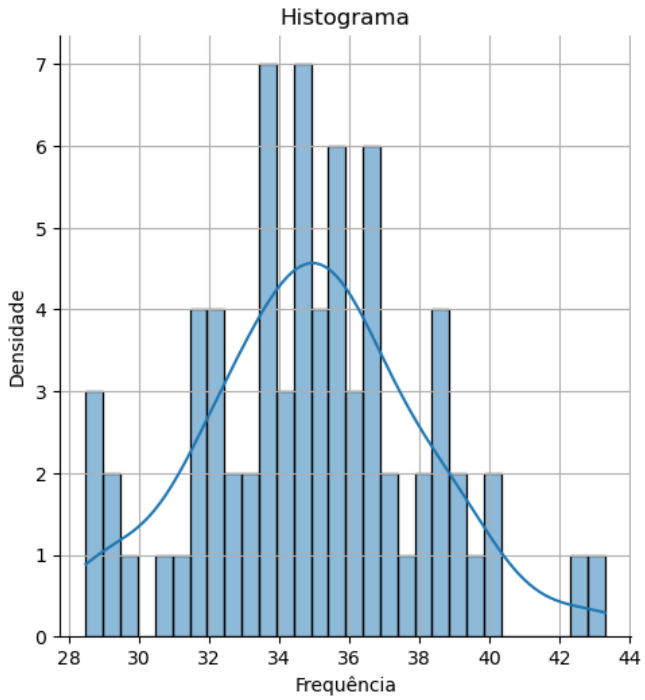
Figura 36 – Estimativas das amostras bootstrap



Fonte: Elaborado pelo autor

Os dados previstos de cada subamostra são importantes para construir um conjunto bootstrap que agrupa todos os dados previstos pelo modelo. Esse conjunto conterá 72 dados que serão úteis para construir o intervalo de confiança com base na distribuição dos dados com nível de confiança de 95%.

Figura 37 – Histograma do conjunto das amostras bootstrap



Fonte: Elaborado pelo autor

Os dados do conjunto bootstrap aparentam seguir uma distribuição normal. Para validar essa análise basta realizar o teste de Jarque-Bera.

Tabela 10 – Teste de Jarque-Bera para o conjunto das amostras bootstrap

Estatística de Teste	**Valor-p**	**Assimetria**	**Curtose**
0.1989	0.9053	0.1214	3.0856

Com base no valor-p acima de 0.05 há fortes indícios de que os dados, de fato, seguem uma distribuição normal. Esse resultado é importante para calcular o intervalo de confiança da rede neural usando a distribuição normal cuja equação é dada por:

$$IC_{1-\alpha}(x_i) = \left(\hat{x}_i - Z_{1-\alpha/2} \cdot \frac{\sigma}{\sqrt{n}}, \hat{x}_i + Z_{1-\alpha/2} \cdot \frac{\sigma}{\sqrt{n}}\right) \tag{5.9}$$

Onde:

$\hat{x}_i$ - Dado previsto onde $i = 1, 2, \ldots, 12$;

α - Nível de significância;

$Z_{1-\alpha/2}$ - Teste crítico Z;

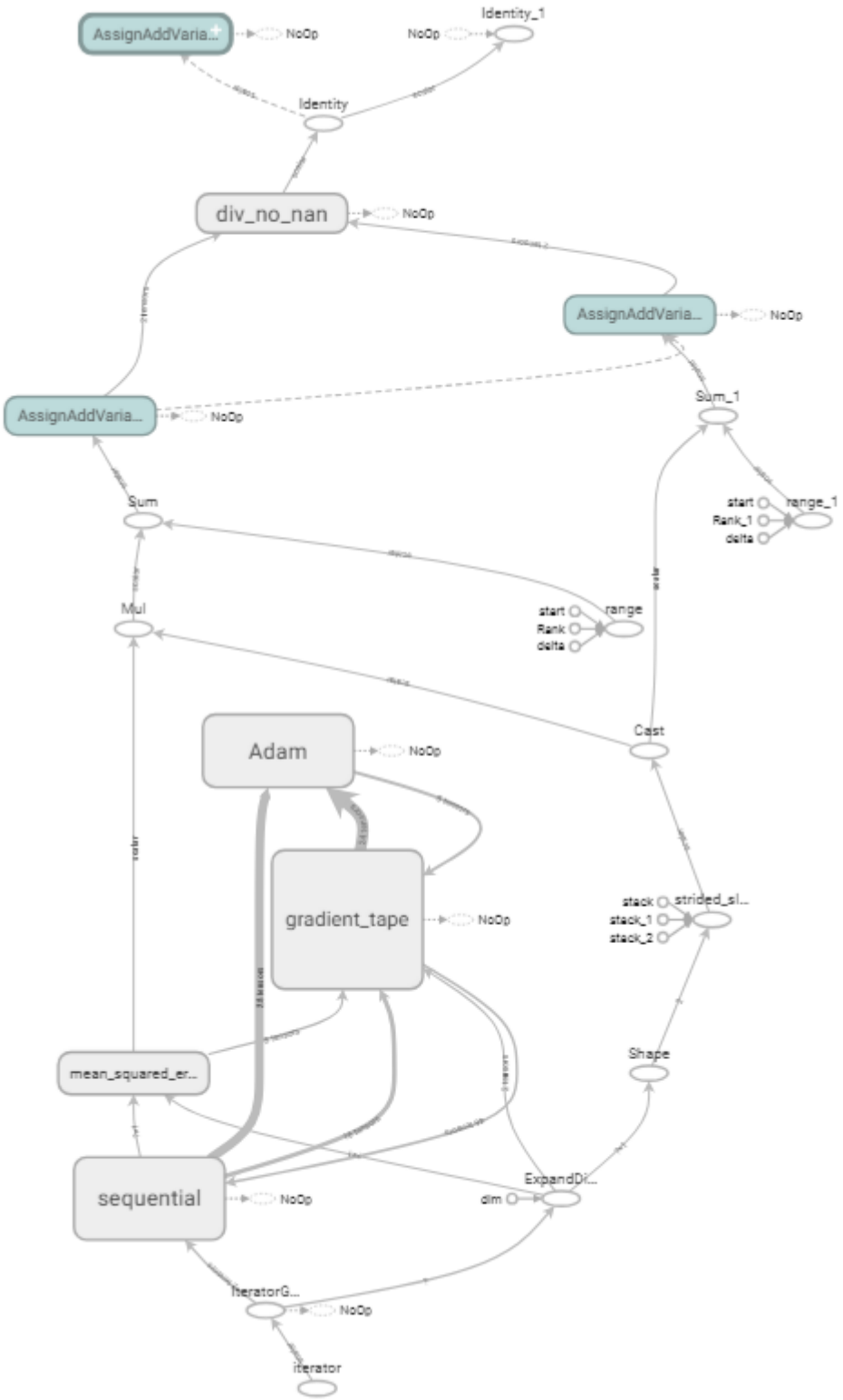
σ - Desvio padrão amostral do conjunto bootstrap;

n - Tamanho do conjunto de previsão.

5.4.4 Arquitetura da Rede Neural

A arquitetura representa o algoritmo da rede neural com todas as etapas.

Figura 38 – Arquitetura da rede neural

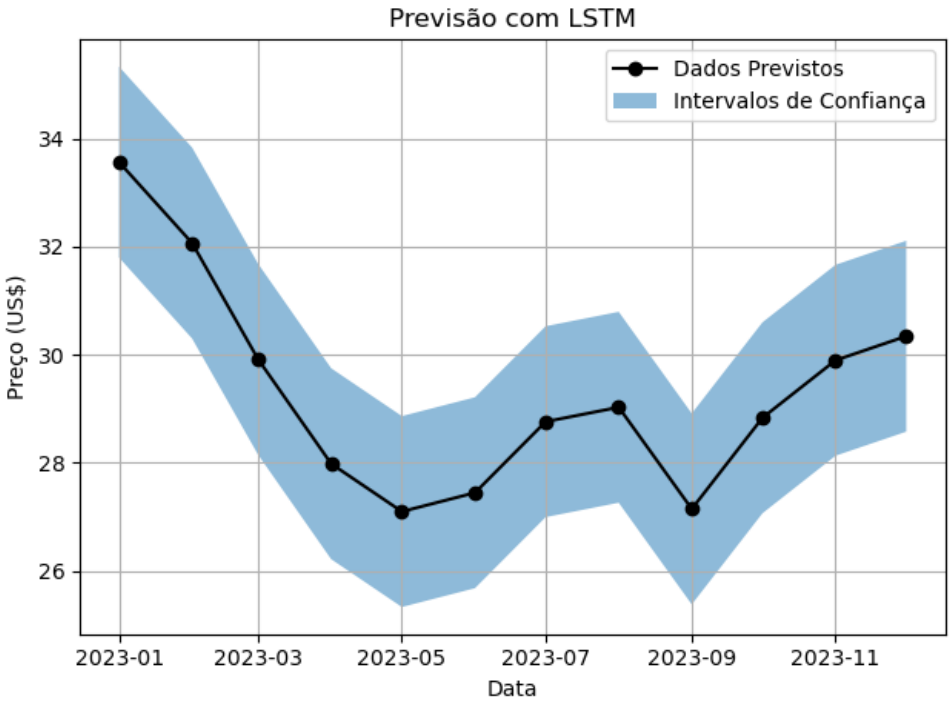


Fonte: Elaborado pelo autor

5.4.5 Previsão do Modelo

Com a rede neural pronta é possível fazer previsões para os próximos 12 meses.

Figura 39 – Previsão com LSTM



Fonte: Elaborado pelo autor

Tabela 11 – Resultados da previsão com LSTM para 2023

Mês	**Previsão**	**Limite Inferior (95%)**	**Limite Superior (95%)**
Janeiro	33.57	31.80	35.33
Fevereiro	32.07	30.30	33.83
Março	29.92	28.15	31.68
Abril	27.98	26.22	29.75
Maio	27.10	25.33	28.86
Junho	27.45	25.68	29.21
Julho	28.76	27.00	30.53
Agosto	29.03	27.26	30.79
Setembro	27.15	25.38	28.91
Outubro	28.84	27.07	30.60
Novembro	29.90	28.13	31.66
Dezembro	30.35	28.58	32.11

A amplitude do intervalo de confiança para o modelo é de 3.53 dólares. Os preços previstos tendem a ser menores se comparados com o modelo anterior.

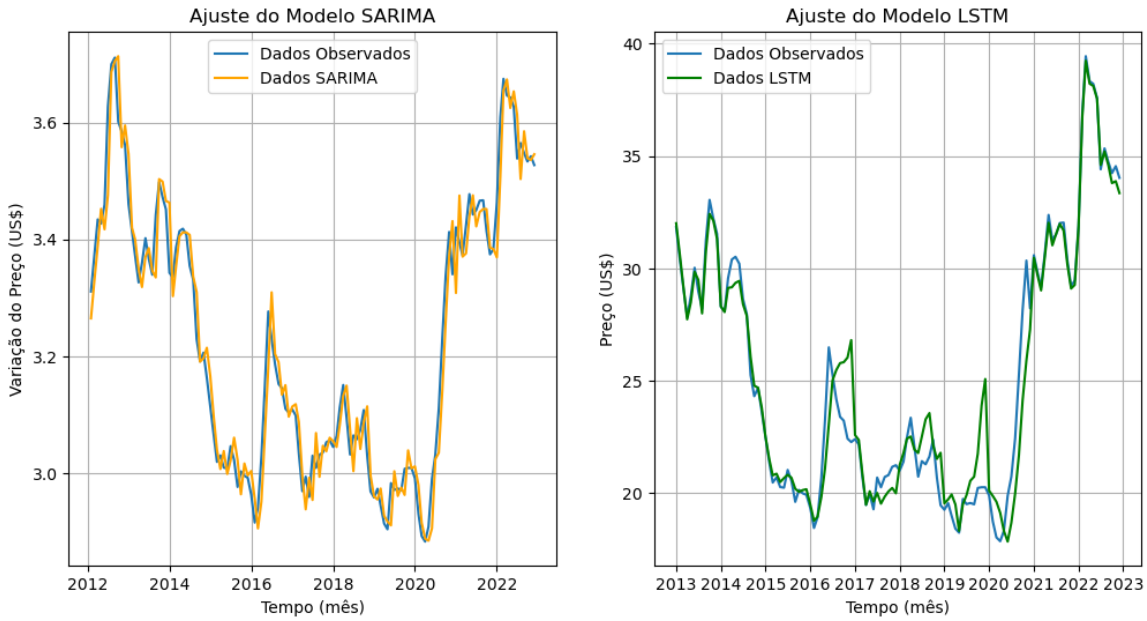
5.5 Análise Comparativa

Agora é possível obter insights dos modelos através da análise comparativa. É fundamental analisar o ajuste e previsão de cada modelo de modo a extrair informações relevantes para a tomada de decisão.

5.5.1 Ajuste dos Modelos

Ajustar os modelos aos dados da série temporal é de suma importância para previsões. Se o modelo não for bem ajustado não será adequado para previsões tanto dentro quanto fora da amostra.

Figura 40 – Ajuste dos modelos

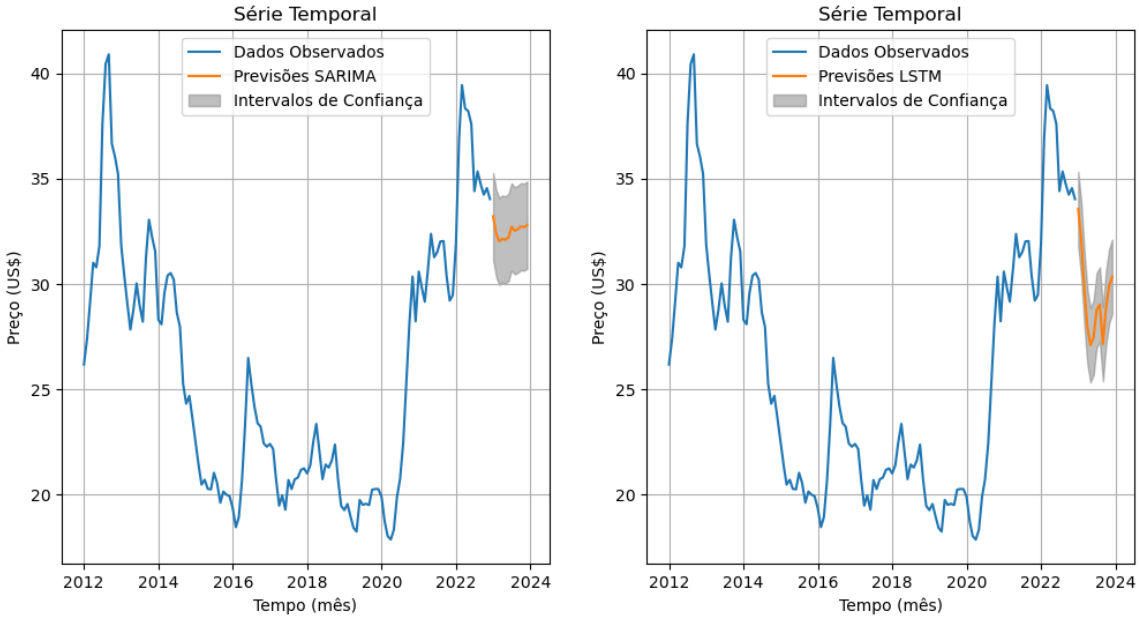


Fonte: Elaborado pelo autor

Nota-se que os ajustes foram decentes para ambos os modelos. O ajuste do modelo SARIMA segue a escala em log por ser desenvolvido na série temporal transformada e não apresenta o primeiro mês de 2012 por ter sido diferenciado uma vez, perdendo assim a primeira informação da série. Já no modelo LSTM os dados iniciam a partir de 2013, perdendo assim o primeiro ano da série por causa do vetor de entradas de tamanho 12 para a rede neural capturar a sazonalidade da série temporal. Analisando os ajustes é possível concluir que os modelos conseguiram se ajustar bem aos dados da série. A rede neural teve dificuldade em ajustar os dados no intervalo entre 2019 e 2020. Também é visível que a rede neural aparenta ter um ajuste melhor no ano de 2022 comparada ao modelo estatístico, impactando assim nas métricas de desempenho analisadas posteriormente.

5.5.2 Análise das Previsões

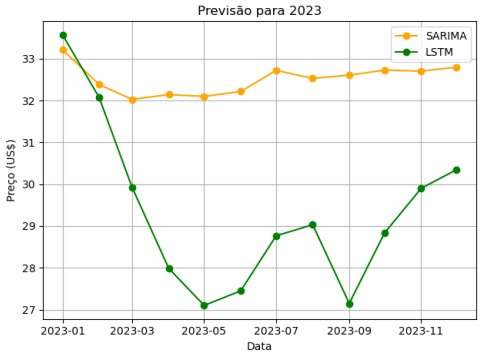
Figura 41 – Previsões para 2023



Fonte: Elaborado pelo autor

É observável como a previsão da rede neural é mais volátil comparada ao modelo estatístico, assim como a amplitude dos intervalos de confiança. Esse comportamento pode ser indicativo de que o modelo LSTM aprendeu comportamentos complexos dos dados nos quais o SARIMA não captou. O modelo estatístico aparenta convergir para um preço constante ao longo do tempo, tornando-o assim útil para previsão de curto prazo apenas. Já na rede neural é esperado que consiga trabalhar com previsões de longo prazo por estar utilizando camadas LSTM.

Figura 42 – Acoplagem preditiva



Fonte: Elaborado pelo autor

As previsões são próximos nos primeiros meses, porém as previsões da rede neural caem de modo geral ao longo do tempo, em referência à Figura 42.

Tabela 12 – Análise exploratória das previsões para 2023

Estatística	**SARIMA**	**LSTM**
Média Aritmética	32.51	29.34
Desvio Padrão	0.35	1.97
Valor Mínimo	32.03	27.10
Primeiro Quartil	32.19	27.85
Segundo Quartil	32.57	28.93
Terceiro Quartil	32.72	30.02
Valor Máximo	33.21	33.57

A média aritmética é menor na rede neural. Além disso, a volatilidade da rede neural é bem maior que o modelo estatístico. Essa característica é comum em redes neurais por apresentarem amplitudes geralmente maiores nas previsões. Note que a volatilidade dos modelos é menor que do ano 2022, sendo essa uma característica da sazonalidade na série temporal que era esperada conforme a Figura 21.

Métricas de Desempenho

As métricas de desempenho são calculadas com base no intervalo de previsão de 12 meses, isto é, previsão futura para o ano de 2023.

Tabela 13 – Métricas de desempenho

Métrica	**SARIMA**	**LSTM**
EAM	1.43	0.24
REQM	1.79	0.33
R^2	0.31	0.98
U de Theil	0.87	0.16

Ambos modelos apresentam métricas aceitáveis, porém a rede neural apresenta resultados muito melhores tornando-a mais adequada à previsão.

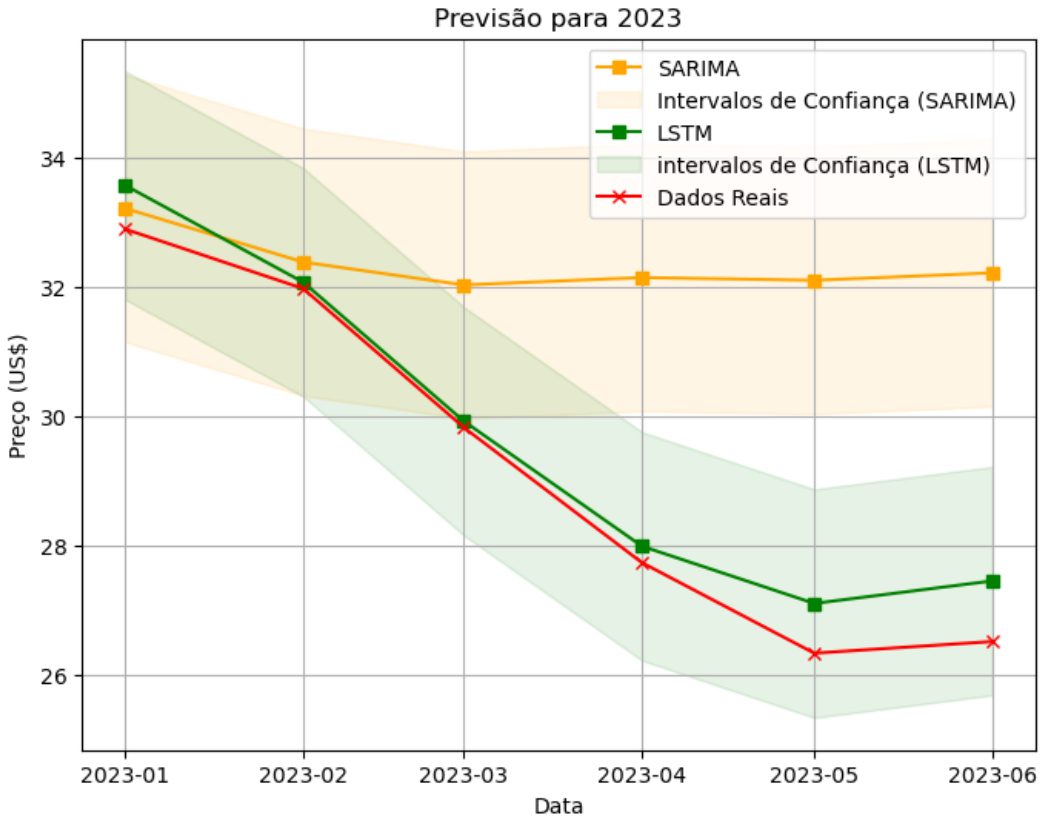
- O modelo estatístico apresenta mais erros do que a rede neural.
- A magnitude dos erros é maior no modelo estatístico do que na rede neural.
- A rede neural se ajusta melhor aos dados com 97.70% que é um ótimo resultado.
- A qualidade da previsão é melhor com a rede neural por 0.1581 estar próximo de 0.

Em questão de métricas a rede neural é mais indicada para fazer as previsões no intervalo proposto. O modelo estatístico também pode ser usado para previsões, porém é mais propício a erros.

5.5.3 Previsão Parcial

Agora é realizada uma análise preditiva no intervalo de 6 meses para averiguar o desempenho dos modelos em relação à dados reais. Não é realizada uma análise de previsão completa porque não existem todos os dados reais de 2023 durante o desenvolvimento deste trabalho. Para isso, os dados referentes ao primeiro dia dos meses de 2023 de janeiro à junho foram obtidos da mesma fonte para realizar a análise parcial.

Figura 43 – Previsão parcial



Fonte: Elaborado pelo autor

Tabela 14 – Comparação dos resultados da previsão parcial

Mês	**SARIMA**	**IC (95%)**	**LSTM**	**IC (95%)**
Janeiro	33.21	31.15 - 35.28	33.57	31.80 - 35.33
Fevereiro	32.38	30.32 - 34.45	32.07	30.30 - 33.83
Março	32.03	29.96 - 34.09	29.92	28.15 - 31.68
Abril	32.14	30.08 - 34.21	27.98	26.22 - 29.75
Maio	32.10	30.03 - 34.16	27.10	25.33 - 28.86
Junho	32.21	30.15 - 34.28	27.45	25.68 - 29.21

É feita então uma breve análise dos resultados:

- Os modelos conseguiram prever adequadamente janeiro e fevereiro.
- Os modelos conseguiram capturar a tendência de subida em junho.
- A tendência de queda nos primeiros meses se concretizou.
- O modelo estatístico não conseguiu previsões razoáveis de março e adiante.
- A rede neural conseguiu capturar as tendências de queda e subida no intervalo.
- A rede neural se saiu melhor que o modelo estatístico para previsão parcial.

Erro Absoluto Preditivo

O erro absoluto serve como uma medida direta da discrepância entre dados reais e previstos. Ao diminuir pela metade a amplitude do intervalo de confiança de cada modelo é possível verificar se os dados reais estão no intervalo.

Tabela 15 – Erro absoluto da previsão parcial

Erro Absoluto	**SARIMA**	**LSTM**
Janeiro	0.32	0.68
Fevereiro	0.41	0.10
Março	2.21	0.10
Abril	4.41	0.25
Maio	5.77	0.77
Junho	5.70	0.94

Percebe-se que o modelo SARIMA conseguiu prever os dados melhor do que LSTM em janeiro. Entretanto, a rede neural se saiu melhor em fevereiro e continuou atingindo bons resultados durante todo o intervalo de previsão com erros absolutos abaixo de um dólar. Em relação aos intervalos de confiança de cada modelo, ao diminuir pela metade a amplitude são obtidos os valores 2.07 e 1.77 para o SARIMA e LSTM, respectivamente. Com base nos valores o SARIMA capturou o preço da soja apenas em janeiro e fevereiro enquanto LSTM conseguiu capturar em todos os meses.

Métricas da Previsão Parcial

As métricas de desempenho da previsão parcial são calculadas com base no intervalo de previsão referente ao primeiro semestre de 2023, isto é, 6 meses.

Tabela 16 – Métricas da previsão parcial

Métrica	**SARIMA**	**LSTM**
EAM	3.14	0.47
REQM	3.88	0.58
U de Theil	2.62	0.39

- O modelo estatístico apresenta muito mais erros do que a rede neural.
- A magnitude dos erros é muito maior no modelo estatístico do que na rede neural.
- O grau de ajustamento do modelo estatístico não é adequado para previsão parcial no intervalo proposto. Sendo assim, a métrica em questão não foi apresentada para comparar os modelos.
- A qualidade da previsão continua boa com a rede neural. Já o modelo estatístico apresenta resultado acima de 1, logo é inferior à previsão ingênua.

Em relação à previsão parcial é notório que a rede neural, de fato, consegue fazer boas previsões para o preço da soja. O modelo estatístico não conseguiu atingir resultados aceitáveis no intervalo previsto além dos dois primeiros meses.

6 Considerações Finais

A habilidade de antecipar com precisão os preços da soja é crucial para agricultores, traders e demais participantes do setor. A adoção desses modelos pode aprimorar a tomada de decisões estratégicas, como identificar o modelo ideal para aquisição ou venda de insumos agrícolas. Além dos métodos estatísticos convencionais, a utilização de inteligência artificial, exemplificada por meio de redes neurais, emerge como uma alternativa promissora. A incorporação de tecnologias de aprendizado de máquina pode proporcionais benefícios significativos ao setor agropecuário, notadamente no Paraná, ao lidar com dados que exibem comportamentos complexos, como os relacionados aos preço da soja.

Na análise da série temporal foram encontradas evidências de sazonalidade com periodicidade anual e estacionariedade na primeira diferença. O desvio padrão anual da série segue um comportamento de queda e subida por ano, o qual se espera uma queda da volatilidade em 2023 que se concretizou com base nas previsões dos modelos.

Para construir o modelo SARIMA foi necessário transformar a série temporal em log de modo a corrigir a normalidade dos resíduos. O modelo SARIMA com o melhor ajuste foi SARIMA(0,1,1)(0,0,1) com base em evidências de sazonalidade na série. Analisando o sumário do modelo foi constatada a presença de ruído branco que é fundamental para a aleatoriedade dos resíduos ao longo do tempo.

O desenvolvimento da rede neural LSTM foi baseado em vários testes utilizando dados de treinamento com de entradas de tamanho 12 para tentar capturar a sazonalidade da série. Foram projetadas duas camadas LSTM com 64 neurônios artificiais e uma camada densa para sua arquitetura totalizando próximo de 50 mil parâmetros. Foram escolhidas 1000 épocas para a passagem completa dos dados de treino de modo a diminuir a função de perda (erro quadrático médio) para um valor abaixo de 0.1. O modelo conseguiu se ajustar bem à série temporal com ótimas métricas para o intervalo de previsão proposto.

Os modelos tiveram algumas similaridades nas previsões para 2023 como queda do preço da soja nos primeiros meses e desvio padrão abaixo de 2022. Em relação às métricas, a rede neural se saiu melhor que o modelo estatístico para a previsão futura com ótimos resultados no geral. Ambos os modelos indicam queda no preço da soja em 2023 comparado ao ano anterior, com a rede neural indicando uma queda maior ao longo do tempo.

O modelo ARIMA era o mais indicado com base nos critérios de informação BIC e HQIC em relação aos modelos econométricos. Ao analisar o comportamento preditivo do modelo verificou-se que não é adequado para previsão por convergir rapidamente a uma reta horizontal. Isso indica que o modelo não conseguiu capturar os indícios de

sazonalidade da série temporal, implicando assim que uma extensão como SARIMA seria mais apropriado.

O modelo LSTM conseguiu capturar comportamentos complexos dos dados da série temporal impactando em métricas melhores mesmo com amostra pequena. É comum redes neurais serem projetadas para lidar com grande volume de dados, porém, com base nos resultados obtidos dessa monografia, redes neurais podem apresentar bons resultados mesmo com poucos dados ao usar parâmetros e número de épocas apropriados.

Em relação à previsão parcial, isto é, previsão dos 6 primeiros meses de 2023, o modelo SARIMA conseguiu ter previsão com margens de erro mais aceitável que LSTM em janeiro de 2023. Nos primeiros dois meses é visível, através de análise gráfica, uma competição acirrada entre os modelos. A rede neural se saiu muito melhor nos próximos meses enquanto mantém ótimas métricas, com erro absoluto preditivo abaixo de um dólar no intervalo de previsão de 6 meses. Além disso, a previsão da rede neural para fevereiro e março apresentou erros abaixo de 0.1 dólares, ou seja, ótimos resultados nesse período. Por fim, a rede neural conseguiu entender o comportamento de queda e subida do preço da soja em todos os meses da previsão parcial, contudo isso não significa que esses resultados continuarão até o intervalo de 12 meses proposto.

O modelo econométrico serve apenas para previsão no curto prazo até 2 meses com base no intervalo de confiança. Ele não serve para previsão no longo prazo porque o modelo converge assintoticamente para uma reta horizontal. Já a rede neural consegue previsões tanto de curto prazo quanto longo prazo justamente por ser projetada para um modelo de memória de curto-longo prazo.

Redes neurais requerem mais processamento computacional do que modelos econométricos. Dependendo da situação pode ser mais vantajoso desenvolver modelos de regressão de modo a minimizar tempo e custo computacional. Para um intervalo de previsão até dois meses poderia ser mais indicado o modelo SARIMA para tomada de decisão mesmo que a rede neural tenha se saído melhor em intervalos maiores de previsão.

Conclui-se que tanto o modelo econométrico quanto a rede neural são apropriados para a previsão do preço da soja para 2023. Embora apresentem abordagens distintas, os modelos conseguiram se ajustar adequadamente à série temporal e obtiveram resultados similares em alguns cenários, com a rede neural se mostrando mais adequada para previsões nesse cenário.

Referências

BABU, A; SUMA, B. Developing an ETL Pipeline for Data Analysis, International Journal of Computer Applications Technology, 2022.

BEAR, Mark F.; CONNORS, Barry W.; PARADISO, Michael A. Neurociências: desvendando o sistema nervoso. Artmed Editora, 2008.

BLIEMEL, F.; MACKAY, D. B. Research Notes And Communications Theil's Forecast Accuracy Coefficient: A Clarification. Sage Publications, Inc., Vol. 10, No. 4, Nov. 1973.

BOX, G. E. P.; PIERCE, D. A. Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models. Taylor & Francis, Ltd., Vol. 65, No. 332, Dec. 1970.

BOX, George E. P.; JENKINS, Gwilym M. Análise de séries temporais. São Paulo: Cultrix, 1971.

BROWNLEE, J. Redes Neurais Recorrentes com Python: Guia Rápido de Início - Fundamentos de Tradução Sequencial de Máquinas para Iniciantes. São Paulo: Novatec, 2020.

BUSHAEV, V. Understanding RMSprop — faster neural network learning, 2018. Disponível em: <https://towardsdatascience.com> Acesso em: 12 de set. de 2023.

CENTRO DE ESTUDOS AVANÇADOS EM ECONOMIA APLICADA (CEPEA). Consultas ao Banco de Dados do Site. São Paulo, 2023. Disponível em: <https://www.cepea.esalq.usp.br/br/consultas-ao-banco-de-dados-do-site.aspx> Acesso em: 30 de abr. de 2023.

COMPANHIA NACIONAL DE ABASTECIMENTO. Acompanhamento da safra brasileira de grãos - Safra 2021/22. Brasília: Conab, 2022. Disponível em: <https://www.conab.gov.br/info-agro/safras/graos/boletim-da-safra-de-graos> Acesso em: 12 de maio de 2023.

STOA. aula7maio, 2015. Disponível em: <https://edisciplinas.usp.br> Acesso em: 5 de nov. de 2023.

GATTIS, N. É preciso agregar valor às commodities, diz ex-ministro da Agricultura. Exame, 2023. Disponível em: <https://exame.com/esferabrasil> Acesso em: 22 de set. de 2023.

GREFF, K. et al. Lstm: a Search space Odyssey, IEEE transactions on neural networks and learning systems, 2016.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning. Cambridge: MIT Press, 2016.

GRUS, Joel. Data Science do Zero: Primeiras Regras com o Python. São Paulo: Novatec Editora, 2018.

HAYKIN, Simon. Redes Neurais: princípios e práticas. 2 ed. Porto Alegre: Bookman, 2001.

HIRAKURI, H; LAZZAROTTO, J. O agronegócio da soja nos contextos mundial e brasileiro. Embrapa Soja, Londrina, PR, junho 2014.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. Painel de Indicadores. Rio de Janeiro, 2023. Disponível em: <https://www.ibge.gov.br/indicadores.html> Acesso em: 17 de set. de 2023.

Investing.com. IPCA Hoje - Acumulado 12 meses - Inflação do Brasil (Anual). Disponível em: <https://br.investing.com/economic-calendar/brazilian-cpi-410>. Acesso em: 12 de set. de 2023.

Investing.com. IPC EUA - Inflação nos Estados Unidos (Anual). Disponível em: <https://br.investing.com/economic-calendar/brazilian-cpi-733>. Acesso em: 12 de set. de 2023.

KILIÇ, I. Perceptron Model: The Foundation of Neural Networks, 2023. disponível em: <https://medium.com/@ilyurek/4db25b0148d> Acesso em: 10 de nov. de 2023.

KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization, 2014. Disponível em: <https://arxiv.org/abs/1412.6980> Acesso em: 12 de set. de 2023

MANKIW, N. Gregory. Macroeconomia. 9. ed. Rio de Janeiro: LTC, 2017.

McKINNEY, Wes. Python para Análise de Dados: Tratamento de Dados com Pandas, Numpy e Ipython. São Paulo: Novatec Editora, 2018.

MORETTIN, P. A.; TOLOI, C. M. C. Análise de Séries Temporais. Third. São Paulo: Blucher, 2018. V. 1.

MURPHY, K. Probabilistic Machine Learning: An Introduction. MIT Press. 2021.

OLAH, C. Understanding LSTM Networks, 2015. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> Acesso em: 10 de nov. de 2023.

PYTHON SOFTWARE FOUNDATION. Python Language Site: Documentation, 2023. Página de documentação. Disponível em: <https://www.python.org/doc>. Acesso em: 21 de jun. de 2023.

SANTOS, G. Uma aplicação de redes neurais recorrentes do tipo LSTM à previsão dos preços de curto prazo do mercado de energia elétrica. 2019.

SIN, C.; WHITE, H. Information criteria for selecting possibly misspecified parametric models. Journal of Econometrics, 1996.

SUTSKEVER, I. Training recurrent neural networks. University of Toronto, 2013.

WICKLIN, R. The essential guide to bootstrapping in SAS, 2018. Disponível em: <https://blogs.sas.com/content/iml/2018/12/12/essential-guide-bootstrapping-sas.html> Acesso em: 5 de nov. de 2023.

WIKIVOYAGE. Sul (Brasil). Disponível em: <https://pt.wikivoyage.org/wiki/Sul_(Brasil)> Acesso em: 30 de dez. de 2023.

WOOLDRIDGE, Jeffrey M. Introdução à econometria: uma abordagem moderna. 6. ed. São Paulo: Cengage Learning, 2016.

YILDIZ, S; DEĞİRMENCİ, M. Estimation of oxygen exchange during treatment sludge composting through multiple regression and artificial neural networks (estimation of oxygen exchange during composting), 2015. Disponível em: <https://www.researchgate.net/publication/285633971> Acesso em: 10 de nov. de 2023.

ZIEGLER, C. et. al. Previsão do preço de soja, trigo e milho por meio da metodologia Box & Jenkins, Caracas, vol. 41, n. 15, p. 20, 30 abr. 2020. Disponível em: <https://www.revistaespacios.com/a20v41n15/20411520.html> Acesso em: 30 de dez. de 2023.

Apêndice - Código das Análises

```
# O apéndice inclui todo código utilizado para a análise dos
# dados. O primeiro ambiente é para as análises e o segundo
# para o bootstrap. Algumas linhas devem ser ajustadas para
# evitar erro de sintaxe.

### Instalação ###

# A instalação é específica por causa de conflitos de versão
# gerados pelo Tensorflow. Note que Keras e Tensorboard já são
# integrados no Tensorflow.

# 1 - Instalar Anaconda Distribution 2022.10 pelos archives.
# 2 - Criar um environment para o Python 3.9.18.
# 3 - Instalar as bibliotecas fundamentais e o Jupyter Notebook.
# 4 - Instalar a biblioteca openpyxl para arquivos Excel 2010.
# 5 - Abrir JN no environment criado anteriormente.

###################
### Ambiente 1 ###
###################

### Bibliotecas ###

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import statsmodels.graphics.tsaplots as tsaplots
from statsmodels.tsa.stattools import adfuller, acf, pacf
from statsmodels.tsa.arima.model import ARIMA
from pmdarima import auto_arima
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import TensorBoard


```

```
### Função Referente ao Teste ADF ###

def adftest(serie_temporal):
    ci, modelos = ['AIC','BIC'], ['n','c','ct']
    for i in ci:
        for j in modelos:
            result = adfuller(serie_temporal, autolag=i,
               regression=j)
            print('''### Teste Dickey-Fuller Aumentado (%s) ###

            Modelo selecionado: %s
            Estatística do teste: %.4f
            Valor-p: %.16f
            Valor Crítico:''' %(i,j,result[0],result[1]))
            for key, value in result[4].items():
                print('\t{}: {:.4f}'.format(key, value))
            print('''
            %s: %f\n''' %(i,result[5]))

### Função do Intervalo de Confiança para Previsões ###

# Algoritmo para intervalos de confiança com distribuição normal
# com nível de confiança de 95%.
# predict: Dados de previsão.
# err: Margem de erro.

def conf_int(predict, err):
    ic = []
    for value in predict:
        limite_inferior = value - 1.96 * err
        limite_superior = value + 1.96 * err
        ic.append([limite_inferior, limite_superior])
    return np.array(ic)

### Função das Métricas ###

# Algoritmo para obter métricas de desempenho.
# Foi desenvolvido porque a biblioteca scikit-learn não tinha a
# métrica U de Theil.
# series: Valores da série temporal.
# pred: Previsão dos valores observados.

```

```
80 def metrics(series, pred):
81     real, real_prev = series[-len(pred):],
          series[-len(pred)-1:-1]
82     mae = np.mean(np.abs(real - pred))
83     rmse = np.sqrt(np.mean((real - pred)**2))
84     r2 = 1 - (np.sum((real - pred)**2) / np.sum((real -
          np.mean(real))**2))
85     theil_u = np.sqrt(np.sum((real - pred)**2) / np.sum((real -
          real_prev)**2))
86
87     print('''### Métricas de Desempenho ###
88
89     EAM: %.4f
90     REQM: %.4f
91     R2: %.4f
92     U de Theil: %.4f''' %(mae, rmse, r2, theil_u))
93
94 ### Função de Previsão com LSTM ###
95
96 # begin: valor inicial da previsão
97 # end: valor final da previsão
98 # year: ano inicial da previsão
99
100 def predict_LSTM(begin, end, year):
101     x_input = st[begin:end].values
102     temp_input = list(x_input)
103     lst_output = []
104     i = 0
105     while(i < n_steps):
106         if (len(temp_input) > n_steps):
107             x_input = np.array(temp_input[1:])
108             x_input = x_input.reshape((1, n_steps, n_features))
109             yhat = model.predict(x_input, verbose=0)
110             temp_input.append(yhat[0][0])
111             temp_input = temp_input[1:]
112             lst_output.append(yhat[0][0])
113             i=i+1
114         else:
115             x_input = x_input.reshape((1, n_steps, n_features))
116             yhat = model.predict(x_input, verbose=0)
117             temp_input.append(yhat[0][0])
118             lst_output.append(yhat[0][0])
```

```
119             i=i+1
120     lst_index = pd.date_range(start='{}-01-01'.format(year),
           periods=12, freq='MS')
121     previsao_LSTM = pd.DataFrame({'Preço': lst_output},
           index=lst_index)
122     previsao_LSTM['Data'] = previsao_LSTM.index
123     previsao_LSTM['Preço'] =
           pd.to_numeric(previsao_LSTM['Preço'].astype(float))
124     previsao_LSTM =
           previsao_LSTM.set_index('Data')['Preço'].asfreq('MS')
125     return previsao_LSTM
126
127 ### Carregar o arquivo com os dados ###
128
129 df = pd.read_excel('dados\consulta.xlsx')
130 df.head()
131
132 # Dataframe: Tipos de dados
133
134 print(df.dtypes)
135
136 # Dataframe: Verificar valores ausentes
137
138 print(df.isna().sum())
139
140 ### Tratamento dos dados ###
141
142 st =
143 df.iloc[3:].drop(columns='Unnamed: 1').reset_index(drop=True)
144 st.columns = ['Data','Preço']
145 st['Data'] = pd.to_datetime(st['Data'], format='%m/%Y')
146 st['Preço'] =
       pd.to_numeric(st['Preço'].str.replace(',','.')).astype(float)
147 st = st.set_index('Data')['Preço'].asfreq('MS')
148 print(st)
149
150 ### Análise Exploratória de Dados ###
151
152 print(st.describe())
153
154 ### Gráfico da Série Temporal ###
155
```

```
156 plt.plot(st)
157 plt.title('Série Temporal')
158 plt.xlabel('Tempo (més)')
159 plt.ylabel('Preço (US$)')
160 plt.grid(True)
161
162 ### Decomposição Aditiva da Série Temporal ###
163
164 decomp = sm.tsa.seasonal_decompose(st)
165 fig, (ax1,ax2,ax3,ax4) = plt.subplots(4,1,figsize=(15,8))
166 decomp.observed.plot(ax=ax1, title='Observações')
167 decomp.trend.plot(ax=ax2, title='Tendéncia')
168 decomp.seasonal.plot(ax=ax3, title='Sazonalidade')
169 decomp.resid.plot(ax=ax4, title='Resíduo')
170 plt.tight_layout()
171
172 ### Dispersão dos Dados ###
173
174 # Aqui é feita regressão polinomial de grau 2 (tendéncia).
175
176 var_X = np.column_stack((list(range(132)),
      np.array(list(range(132))) ** 2))
177 var_X = sm.add_constant(var_X)
178 reg = sm.OLS(st.values, var_X).fit()
179
180 sns.scatterplot(st, marker='o', color='b')
181 sns.lineplot(x=st.index, y=reg.predict(var_X), color='r')
182 plt.title('Gráfico de Dispersão')
183 plt.xlabel('Tempo (més)')
184 plt.ylabel('Preço (US$)')
185 plt.show()
186
187 ### Série Temporal Dessazonalizada ###
188
189 # Pegar a série temporal e diminuir da média móvel de 12 termos.
190 # Após isso calcular a média móvel e desvio padráo móvel da série
191 # dessazonalidade e fazer gráfico acoplado.
192
193 st_des = (st - st.rolling(12).mean()).dropna()
194 st_des_ma = st_des.rolling(12).mean()
195 st_des_std = st_des.rolling(12).std()
196
```

```
fig, ax = plt.subplots()
st_des.plot(ax=ax)
st_des_ma.plot(ax=ax, color='r')
st_des_std.plot(ax=ax, color='g')
plt.title('Série Temporal Dessazonalizada')
plt.xlabel('Tempo (més)')
plt.ylabel('Preço (US$)')
plt.legend(['Dados Observados', 'Média Móvel',
'Desvio Padráo Móvel'])
plt.grid(True)
plt.tight_layout()

### Volatilidade Anual (desvio padráo anual) ###

ano = list(range(2012,2023))
dp_anual = [st[i*12:i*12+12].std() for i in
   range(int(len(st)/12))]

plt.figure(figsize=(8,6))
plt.plot(ano, dp_anual, marker='o', color='g')
plt.xlabel('Ano')
plt.ylabel('Desvio Padráo')
plt.title('Volatilidade Anual do Preço')
plt.grid(True)
plt.show()

### Histograma dos Resíduos ###

plt.figure(figsize=(10,6))
sns.displot(decomp.resid.dropna(), bins=30, kde=True)
plt.title('Histograma')
plt.xlabel('Frequéncia')
plt.ylabel('Densidade')
plt.grid(True)
plt.show()

### Teste de Jarque-Bera ###

jb_1, jb_2, jb_3, jb_4 = sm.stats.jarque_bera
(sm.tsa.seasonal_decompose(st).resid.dropna())
jbl_1, jbl_2, jbl_3, jbl_4 = sm.stats.jarque_bera
(sm.tsa.seasonal_decompose(np.log(st)).resid.dropna())
```

```
238
239 print(f"Teste JB: Estatística de teste =
240 {jb_1}, p-valor = {jb_2}, Skew = {jb_3}, Kurtosis = {jb_4}")
241 print(f"Teste JB (log): Estatística de teste =
242 {jbl_1}, p-valor = {jbl_2}, Skew = {jbl_3}, Kurtosis = {jbl_4}")
243
244 ######################
245 ### Modelo SARIMA ###
246 ######################
247
248 ### Transformação da Série Temporal ###
249
250 st_log = np.log(st)
251 ma_log = st_log.rolling(12).mean()
252
253 fig, ax = plt.subplots()
254 st_log.plot(ax=ax)
255 ma_log.plot(ax=ax, color='r')
256 plt.title('Série Temporal')
257 plt.xlabel('Tempo (més)')
258 plt.ylabel('Variação do Preço (US$)')
259 plt.legend(['Dados Observados', 'Média Móvel'])
260 plt.grid(True)
261 plt.tight_layout()
262
263 # Teste ADF em Nível
264
265 adftest(st_log)
266
267 ### Diferenciar a Série Temporal ###
268
269 std = st_log.diff(1).dropna()
270 std_ma = std.rolling(12).mean()
271
272 fig, ax = plt.subplots()
273 std.plot(ax=ax)
274 std_ma.plot(ax=ax, color='r')
275 plt.title('Série Temporal na Primeira Diferença')
276 plt.xlabel('Tempo (més)')
277 plt.ylabel('Variação do Preço (US$)')
278 plt.legend(['Dados Observados', 'Média Móvel'])
279 plt.grid(True)
```

```
plt.tight_layout()

# Teste ADF na Primeira Diferença

adftest(std)

### Componente Sazonal ###

st_cs = sm.tsa.seasonal_decompose(st_log).seasonal
st_csma = st_cs.rolling(12).mean()

fig, ax = plt.subplots()
st_cs.plot(ax=ax)
st_csma.plot(ax=ax, color='r')
plt.title('Componente Sazonal')
plt.xlabel('Tempo (més)')
plt.ylabel('Variação do Preço (US$)')
plt.legend(['Dados Observados', 'Média Móvel'])
plt.grid(True)
plt.tight_layout()

### Plotar ACF e PACF ###

lag_acf = acf(std, nlags=25)
lag_pacf = pacf(std, nlags=25)

plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
plt.plot(lag_acf, marker='s')
plt.xlabel('Lags')
plt.ylabel('Coeficiente de Correlação')
plt.axhline(y=-1.96 / (np.sqrt((len(std) - 1))), linestyle='--',
    color='r', linewidth='0.9')
plt.axhline(y=0, linestyle='--', color='k', linewidth=0.7)
plt.axhline(y=1.96 / (np.sqrt((len(std) - 1))), linestyle='--',
    color='r', linewidth='0.9')
plt.title("Função de Autocorrelação (MA)")

plt.subplot(1,2,2)
plt.plot(lag_pacf, marker='s')
plt.xlabel('Lags')
plt.ylabel('Coeficiente de Correlação')
```

```
320 plt.axhline(y=-1.96 / (np.sqrt((len(std) - 1))), linestyle='--',
    color='r', linewidth='0.9')
321 plt.axhline(y=0, linestyle='--', color='k', linewidth=0.7)
322 plt.axhline(y=1.96 / (np.sqrt((len(std) - 1))), linestyle='--',
    color='r', linewidth='0.9')
323 plt.title("Função de Autocorrelação Parcial (AR)")
324
325 plt.show()
326
327 # Estimar o melhor SARIMA (AIC)
328
329 auto_arima(st_log, start_p=0, d=1, start_q=0,
330                    max_p=2, max_q=2, D=0, m=12,
331                    with_intercept=False, trace=True)
332
333 # Estimar o melhor SARIMA (BIC)
334
335 auto_arima(st_log, start_p=0, d=1, start_q=0,
336                    max_p=2, max_q=2, D=0, m=12,
337                    information_criterion='bic',
                       with_intercept=False, trace=True)
338
339 # Estimar o melhor SARIMA (HQIC)
340
341 auto_arima(st_log, start_p=0, d=1, start_q=0,
342                    max_p=2, max_q=2, D=0, m=12,
343                    information_criterion='hqic',
                       with_intercept=False, trace=True)
344
345 ### Modelo ARIMA(0,1,1) ###
346
347 modelo = ARIMA(st_log, order=(0,1,1)).fit()
348 modelo.summary()
349
350 ### Previsão com Intervalo de Confiança (ARIMA) ###
351
352 previsao_ARIMA_scale = modelo.predict(start='2023-01-01',
    end='2023-12-01')
353 previsao_ARIMA_normal = np.exp(previsao_ARIMA_scale)
354
355 arima_err = np.exp(np.sqrt(0.0023 * (1 + 0.3824**2)))
356
```

```
357 arima_ic = conf_int(previsao_ARIMA_normal, arima_err)
358
359 plt.plot(st.index, st.values, label='Dados Observados')
360 plt.plot(previsao_ARIMA_normal.index,
       previsao_ARIMA_normal.values, label='Previsões ARIMA')
361 plt.fill_between(x=previsao_ARIMA_normal.index,
       y1=arima_ic[:,0], y2=arima_ic[:,1], alpha=0.5,
362                  color='gray', label='Intervalos de Confiança')
363 plt.title('Série Temporal')
364 plt.xlabel('Tempo (més)')
365 plt.ylabel('Preço (US$)')
366 plt.legend()
367 plt.grid(True)
368
369 ### Dados de previsão ARIMA ###
370
371 print(previsao_ARIMA_normal)
372
373 ### Modelo SARIMA(0,1,1)(0,0,1,12) ###
374
375 modelo = ARIMA(st_log, order=(0,1,1),
       seasonal_order=(0,0,1,12)).fit()
376 modelo.summary()
377
378 ### Previsão com Intervalo de Confiança (SARIMA) ###
379
380 previsao_SARIMA_scale = modelo.predict(start='2023-01-01',
       end='2023-12-01')
381 previsao_SARIMA_normal = np.exp(previsao_SARIMA_scale)
382
383 sarima_err = np.exp(np.sqrt(0.0023 * (1 + 0.3918**2 +
       0.1821**2)))
384
385 sarima_ic = conf_int(previsao_SARIMA_normal, sarima_err)
386
387 plt.plot(previsao_SARIMA_normal.index,
       previsao_SARIMA_normal.values, marker='o', color='k')
388 plt.fill_between(x=previsao_SARIMA_normal.index,
       y1=sarima_ic[:,0], y2=sarima_ic[:,1], alpha=0.5)
389 plt.title('Previsão com SARIMA')
390 plt.xlabel('Data')
391 plt.ylabel('Preço (US$)')
```

```
392 plt.legend(['Dados Previstos', 'Intervalos de Confiança'])
393 plt.grid(True)
394 plt.tight_layout()
395
396 ### Dados de Previsão SARIMA ###
397
398 print(previsao_SARIMA_normal)
399
400 ### Intervalo de Confiança SARIMA ###
401
402 print(sarima_ic)
403
404 ### Métricas do SARIMA ###
405
406 previsao_SARIMA_scale_pred = modelo.predict(start='2022-01-01',
       end='2022-12-01')
407 previsao_SARIMA_normal_pred = np.exp(previsao_SARIMA_scale_pred)
408 metrics(st.values, previsao_SARIMA_normal_pred.values)
409
410 ####################
411 ### Modelo LSTM ###
412 ####################
413
414 ### Vetores de entrada e saída ###
415
416 def prepare_data(timeseries_data, n_features):
417     X, y =[],[]
418     for i in range(len(timeseries_data)):
419         end_ix = i + n_features
420         if end_ix > len(timeseries_data)-1:
421             break
422         seq_x, seq_y = timeseries_data[i:end_ix],
              timeseries_data[end_ix]
423         X.append(seq_x)
424         y.append(seq_y)
425     return np.array(X), np.array(y)
426
427 ### Sequéncia para o modelo LSTM ###
428
429 timeseries_data = st.values
430 n_steps = 12
431 X, y = prepare_data(timeseries_data, n_steps)
```

```
432
433 # Análise dos vetores
434
435 print(X), print(y), print(X.shape), print(y.shape)
436
437 ### Transformar X para [samples, timesteps, n_features] ###
438
439 n_features = 1
440 X = X.reshape((X.shape[0], X.shape[1], n_features))
441
442 print(X), print(y), print(X.shape), print(y.shape)
443
444 ### Construção do LSTM ###
445
446 # A ideia central é fazer testes para diferentes cenários.
447 # Quantidade de camadas, neurónios, funções de ativação, etc.
448 # Aqui só tem dados de treinamento.
449 # Não há dados de validação e de teste.
450 # Sendo assim, serão feitas análises estatísticas complementares.
451 # Você quer minimizar a função de perda o mais próximo de 0.
452 # Ao ter um resultado satisfatório, verifique o ajuste do modelo.
453 # As métricas de desempenho também são úteis na validação.
454 # Após isso, faça uma comparação entre os melhores modelos.
455 # Desta forma permite verificar se o comportamento é adequado.
456 # Se o comportamento do melhor modelo é parecido, então está ok.
457
458 model = Sequential()
459 model.add(LSTM(64, activation='relu', return_sequences=True,
      input_shape=(n_steps, n_features)))
460 model.add(LSTM(64, activation='relu'))
461 model.add(Dense(1))
462 model.compile(optimizer='adam', loss='mse')
463 model.summary()
464
465 ### Registro do modelo no Tensorboard ###
466
467 tensorboard_callback = TensorBoard(log_dir='logs/fit',
      histogram_freq=1)
468
469 ### Determinar o melhor modelo com redução de loss ###
470
```

```
model.fit(X, y, epochs=1000, verbose=1,
    callbacks=[tensorboard_callback])

# Para abrir o tensorboard você deve abrir o prompt de comando.
# Após isso selecione a pasta do JN do seu arquivo atual.
# Exemplo: cd C:\TCC (seleciona a pasta TCC do disco rígido)
# Então basta digitar:
# tensorboard --logdir=logs/fit
# Após isso basta acessá-lo geralmente em http://localhost:6006/
# Lembrar de limpar as informações ao refazer os testes.

### Previsão com LSTM ###

previsao_LSTM = predict_LSTM(len(st)-12, len(st), 2023)
print(previsao_LSTM)

### Métricas do LSTM ###

previsao_LSTM_pred = predict_LSTM(len(st)-24, len(st)-12, 2022)
metrics(st.values, previsao_LSTM_pred.values)

### Estimação do modelo LSTM ###

previsao_LSTM_all = []

for i in range(10):
    prev = predict_LSTM(i*12, i*12+12, 2013+i)
    previsao_LSTM_all.extend(prev.values)

lst_index = pd.date_range(start='2013-01-01',
    periods=len(previsao_LSTM_all), freq='MS')
modelo_LSTM = pd.DataFrame({'Preço': previsao_LSTM_all},
    index=lst_index)
modelo_LSTM['Data'] = modelo_LSTM.index
modelo_LSTM['Preço'] =
    pd.to_numeric(modelo_LSTM['Preço'].astype(float))
modelo_LSTM = modelo_LSTM.set_index('Data')['Preço'].asfreq('MS')

### Bootstrap por Reamostragem ###

def bootstrap(sample):
    resample = sample.copy()
```

```
509     for i in range(11):
510         np.random.shuffle(resample[i*12:i*12+12])
511     return resample
512 
513 ### Previsões bootstrap ###
514 
515 # Essas previsões foram feitas no ambiente 2.
516 
517 s1 = np.array([30.905914,28.853537,31.750362,34.674889,
518                32.721142,29.660192,29.077650,33.989597,
519                33.519291,32.280750,28.548050,29.339939])
520 
521 s2 = np.array([35.801109,34.964558,36.151859,34.378193,
522                32.045639,33.354939,33.441235,32.752998,
523                34.668011,31.825100,31.310980,34.823242])
524 
525 s3 = np.array([33.837746,35.596634,34.700790,31.648600,
526                36.785778,35.060898,38.916176,38.580769,
527                40.315804,43.303249,35.820595,33.408897])
528 
529 s4 = np.array([42.419243,34.719707,37.556728,38.295029,
530                37.948772,38.844646,38.807236,28.481318,
531                36.156094,35.610229,38.882404,36.135887])
532 
533 s5 = np.array([36.648148,33.724831,35.744335,32.129383,
534                31.771309,36.475113,39.357666,40.158710,
535                36.458225,36.517303,36.997467,33.729069])
536 
537 s6 = np.array([34.076870,35.338013,35.520142,35.205788,
538                34.759224,33.456642,34.713291,36.528690,
539                38.451008,37.161098,33.816299,32.160240])
540 
541 bootstrap_predict = np.concatenate((s1,s2,s3,s4,s5,s6))
542 
543 ### Gráfico das amostras por Bootstrap ###
544 
545 for i in range(6):
546     plt.plot(st.index, bootstrap(st.values), linewidth=0.35)
547 
548 plt.title('Bootstrap por Reamostragem')
549 plt.xlabel('Tempo (mês)')
550 plt.ylabel('Preço (US$)')
```

```
plt.legend(['B1', 'B2', 'B3', 'B4', 'B5', 'B6'])
plt.grid(True)

### Estimação bootstrap ###

for i in range(6):
    plt.plot(previsao_LSTM.index,
       bootstrap_predict[i*12:i*12+12], linestyle='dotted',
       marker='o')

plt.title('Estimação Bootstrap')
plt.xlabel('Data')
plt.ylabel('Preço (US$)')
plt.grid(True)
plt.tight_layout()

### Histograma bootstrap ###

plt.figure(figsize=(10,6))
sns.displot(bootstrap_predict, bins=30, kde=True)
plt.title('Histograma')
plt.xlabel('Frequência')
plt.ylabel('Densidade')
plt.grid(True)
plt.show()

### Teste Jarque-Bera bootstrap ###

jbb_1, jbb_2, jbb_3, jbb_4 =
   sm.stats.jarque_bera(bootstrap_predict)

print(f"Teste JB: Estatística de teste =
{jbb_1}, p-valor = {jbb_2}, Skew = {jbb_3}, Kurtosis = {jbb_4}")

### Previsão LSTM com Intervalo de Confiança ###

lstm_err = np.std(bootstrap_predict, ddof=1) / np.sqrt(12)

lstm_ic = conf_int(previsao_LSTM, lstm_err)

plt.plot(previsao_LSTM.index, previsao_LSTM.values, marker='o',
   color='k')
```

```
plt.fill_between(x=previsao_LSTM.index, y1=lstm_ic[:,0],
    y2=lstm_ic[:,1], alpha=0.5)
plt.title('Previsão com LSTM')
plt.xlabel('Data')
plt.ylabel('Preço (US$)')
plt.legend(['Dados Previstos', 'Intervalos de Confiança'])
plt.grid(True)
plt.tight_layout()

### Intervalo de confiança LSTM ###

print(lstm_ic)

############################
### Análise Comparativa ###
############################

### Ajustes das previsões SARIMA e LSTM ###

modelo_SARIMA = modelo.predict(start='2012-01-01',
    end='2022-12-01')

plt.figure(figsize=(12,6))

# SARIMA

plt.subplot(1,2,1)
plt.plot(st_log[1:].index, st_log[1:].values,
label='Dados Observados')
plt.plot(modelo_SARIMA.index[1:], modelo_SARIMA.values[1:],
    label='Dados SARIMA', color='orange')
plt.title('Ajuste do Modelo SARIMA')
plt.xlabel('Tempo (més)')
plt.ylabel('Variaçáo do Preço (US$)')
plt.legend()
plt.grid(True)

# LSTM

plt.subplot(1,2,2)
plt.plot(st[12:].index, st[12:].values, label='Dados Observados')
plt.plot(modelo_LSTM.index, modelo_LSTM.values,
```

```
label='Dados LSTM', color='g')
plt.title('Ajuste do Modelo LSTM')
plt.xlabel('Tempo (més)')
plt.ylabel('Preço (US$)')
plt.legend()
plt.grid(True)

plt.show()

### Gráficos das previsões SARIMA E LSTM ###

plt.figure(figsize=(12,6))

# SARIMA

plt.subplot(1,2,1)
plt.plot(st.index, st.values, label='Dados Observados')
plt.plot(previsao_SARIMA_normal.index,
    previsao_SARIMA_normal.values, label='Previsões SARIMA')
plt.fill_between(x=previsao_SARIMA_normal.index,
    y1=sarima_ic[:,0], y2=sarima_ic[:,1], alpha=0.5,
                 color='gray', label='Intervalos de Confiança')
plt.title('Série Temporal')
plt.xlabel('Tempo (més)')
plt.ylabel('Preço (US$)')
plt.legend()
plt.grid(True)

# LSTM

plt.subplot(1,2,2)
plt.plot(st.index, st.values, label='Dados Observados')
plt.plot(previsao_LSTM.index, previsao_LSTM.values,
    label='Previsões LSTM')
plt.fill_between(x=previsao_LSTM.index, y1=lstm_ic[:,0],
    y2=lstm_ic[:,1], alpha=0.5,
                 color='gray', label='Intervalos de Confiança')
plt.title('Série Temporal')
plt.xlabel('Tempo (més)')
plt.ylabel('Preço (US$)')
plt.legend()
plt.grid(True)
```

```
666
667 plt.show()
668
669 ### Previsões SARIMA e LSTM ###
670
671 plt.plot(previsao_SARIMA_normal.index,
       previsao_SARIMA_normal.values, marker='o', color='orange')
672 plt.plot(previsao_LSTM.index, previsao_LSTM.values, marker='o',
       color='g')
673 plt.title('Previsão para 2023')
674 plt.xlabel('Data')
675 plt.ylabel('Preço (US$)')
676 plt.legend(['SARIMA', 'LSTM'])
677 plt.grid(True)
678 plt.tight_layout()
679
680 ### Estatística descritiva das previsões ###
681
682 print(previsao_SARIMA_normal.describe())
683 print(previsao_LSTM.describe())
684
685 ### Previsões parciais para 2023 ###
686
687 parcial_real = np.array([32.89,31.97,29.82,27.73,26.33,26.51])
688
689 st_extended = np.append(st.values, parcial_real)
690
691 ### Gráfico das previsões parciais ###
692
693 plt.figure(figsize=(8,6))
694 plt.plot(previsao_SARIMA_normal.index[:6],
       previsao_SARIMA_normal.values[:6], marker='s', color='orange')
695 plt.fill_between(x=previsao_SARIMA_normal.index[:6],
       y1=sarima_ic[:6,0], y2=sarima_ic[:6,1], alpha=0.1,
       color='orange', label='Intervalos de Confiança (SARIMA)')
696 plt.plot(previsao_SARIMA_normal.index[:6], previsao_LSTM[:6],
       marker='s', color='g')
697 plt.fill_between(x=previsao_SARIMA_normal.index[:6],
       y1=lstm_ic[:6,0], y2=lstm_ic[:6,1], alpha=0.1, color='g',
       label='Intervalos de Confiança (LSTM)')
698 plt.plot(previsao_SARIMA_normal.index[:6], parcial_real,
       marker='x', color='r')
```

```
plt.title('Previsão para 2023')
plt.xlabel('Data')
plt.ylabel('Preço (US$)')
plt.legend(['SARIMA', 'Intervalos de Confiança (SARIMA)',
'LSTM', 'intervalos de Confiança (LSTM)', 'Dados Reais'])
plt.grid(True)

### Erro Absoluto Preditivo ###

print(np.abs(previsao_SARIMA_normal.values[:6] - parcial_real))
print(np.abs(previsao_LSTM[:6].values - parcial_real))

### Métricas SARIMA ###

metrics(st_extended, previsao_SARIMA_normal.values[:6])

### Métricas LSTM ###

metrics(st_extended, previsao_LSTM[:6])

####################
### Ambiente 2 ###
####################

# Nesse ambiente é mostrado como usar o método de bootstrap
# por reamostragem para estimar os dados da rede neural LSTM
# para desenvolver os intervalos de confiança. A ideia é
# salvar os resultados reparadamente e depois agrupar as
# subamostras em um único conjunto bootstrap.

### Bibliotecas ###

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

### Série Temporal ###

df = pd.read_excel('dados\consulta.xlsx')
```

```
741 df.head()
742 st =
743 df.iloc[3:].drop(columns='Unnamed: 1').reset_index(drop=True)
744 st.columns = ['Data','Preço']
745 st['Data'] = pd.to_datetime(st['Data'], format='%m/%Y')
746 st['Preço'] =
       pd.to_numeric(st['Preço'].str.replace(',','.')).astype(float)
747 st = st.set_index('Data')['Preço'].asfreq('MS')
748
749 ### Função de Previsão com LSTM ###
750
751 def predict_LSTM(begin, end, year):
752     x_input = st[begin:end].values
753     temp_input = list(x_input)
754     lst_output = []
755     i = 0
756     while(i < n_steps):
757         if (len(temp_input) > n_steps):
758             x_input = np.array(temp_input[1:])
759             x_input = x_input.reshape((1, n_steps, n_features))
760             yhat = model.predict(x_input, verbose=0)
761             temp_input.append(yhat[0][0])
762             temp_input = temp_input[1:]
763             lst_output.append(yhat[0][0])
764             i=i+1
765         else:
766             x_input = x_input.reshape((1, n_steps, n_features))
767             yhat = model.predict(x_input, verbose=0)
768             temp_input.append(yhat[0][0])
769             lst_output.append(yhat[0][0])
770             i=i+1
771     lst_index = pd.date_range(start='{}-01-01'.format(year),
          periods=12, freq='MS')
772     previsao_LSTM = pd.DataFrame({'Preço': lst_output},
          index=lst_index)
773     previsao_LSTM['Data'] = previsao_LSTM.index
774     previsao_LSTM['Preço'] =
          pd.to_numeric(previsao_LSTM['Preço'].astype(float))
775     previsao_LSTM =
          previsao_LSTM.set_index('Data')['Preço'].asfreq('MS')
776     return previsao_LSTM
777
```

```
778 ### Vetores de entrada e saída ###
779
780 def prepare_data(timeseries_data, n_features):
781     X, y =[],[]
782     for i in range(len(timeseries_data)):
783         end_ix = i + n_features
784         if end_ix > len(timeseries_data)-1:
785             break
786         seq_x, seq_y = timeseries_data[i:end_ix],
              timeseries_data[end_ix]
787         X.append(seq_x)
788         y.append(seq_y)
789     return np.array(X), np.array(y)
790
791 ### Bootstrap por Reamostragem ###
792
793 def bootstrap(sample):
794     resample = sample.copy()
795     for i in range(11):
796         np.random.shuffle(resample[i*12:i*12+12])
797     return resample
798
799 ### Sequéncia para bootstrap por reamostragem ###
800
801 resample = bootstrap(st.values)
802 timeseries_data = resample
803 n_steps = 12
804 X, y = prepare_data(timeseries_data, n_steps)
805
806 ### Gráfico bootstrap ###
807
808 plt.plot(st.index, resample)
809 plt.title('Bootstrap por Reamostragem')
810 plt.xlabel('Tempo (més)')
811 plt.ylabel('Preço (US$)')
812 plt.grid(True)
813
814 ### Transformar X para [samples, timesteps, n_features] ###
815
816 n_features = 1
817 X = X.reshape((X.shape[0], X.shape[1], n_features))
818
```

```
819 ### Construção do LSTM ###
820
821 model = Sequential()
822 model.add(LSTM(64, activation='relu', return_sequences=True,
        input_shape=(n_steps, n_features)))
823 model.add(LSTM(64, activation='relu'))
824 model.add(Dense(1))
825 model.compile(optimizer='adam', loss='mse')
826 model.summary()
827
828 ### Determinar o melhor modelo com redução de loss ###
829
830 model.fit(X, y, epochs=1000, verbose=1)
831
832 ### Previsão com LSTM por reamostragem ###
833
834 previsao_LSTM = predict_LSTM(len(resample)-12, len(resample),
        2023)
835
836 print(previsao_LSTM)
837
838 ### Estimação bootstrap ###
839
840 plt.plot(previsao_LSTM.index, previsao_LSTM.values,
        linestyle='dotted', marker='o')
841 plt.title('Estimação Bootstrap')
842 plt.xlabel('Data')
843 plt.ylabel('Preço (US$)')
844 plt.grid(True)
845 plt.tight_layout()
```

Universidade Federal do Rio Grande – FURG
Instituto de Matemática, Estatística e Física
Curso de Bacharelado em Matemática Aplicada
Av. Itália km 8 Bairro Carreiros
Rio Grande-RS CEP: 96.203-900 Fone (53)3293.5411
e-mail: imef@furg.br Sítio: www.imef.furg.br

Ata de Defesa de Monografia

No dia 19 do mês de dezembro de 2023, às 15h30, foi realizada a apresentação pública da defesa do Trabalho de Conclusão de Curso do acadêmico **Aryel Soares Loureiro**, sob orientação da Profª. Doutora Raquel da Fontoura Nicolette, deste instituto, e intitulada **Previsão de Preços da Soja no Paraná: estudo comparativo entre modelos SARIMA e LSTM**. Para participar da banca avaliadora junto a orientadora foram convidados o Prof. Doutor Adilson da Silva Nunes IMEF/FURG, o Prof. Doutor Paul G. Kinas IMEF/FURG e o Prof. Dr. Rodrigo da Rocha Gonçalves ICEAC/FURG. Concluídos os trabalhos de apresentação e arguição, o candidato foi: (X) aprovado por unanimidade; ( ) aprovado somente após satisfazer as exigências que constam na folha de modificações, no prazo fixado pela banca; ( ) reprovada. Na forma regulamentar, foi lavrada a presente ata, que é abaixo assinada pelos membros da banca, na ordem acima relacionada.

gov.br Documento assinado digitalmente
RAQUEL DA FONTOURA NICOLETTE
Data: 16/01/2024 16:30:35-0300
Verifique em https://validar.iti.gov.br

Profª. Doutora Raquel da Fontoura Nicolette
(Orientadora-FURG)

gov.br Documento assinado digitalmente
ADILSON DA SILVA NUNES
Data: 19/01/2024 13:15:18-0300
Verifique em https://validar.iti.gov.br

Prof. Doutor Adilson da Silva Nunes
(Avaliador - IMEF - FURG)

gov.br Documento assinado digitalmente
PAUL GERHARD KINAS
Data: 16/01/2024 17:59:58-0300
Verifique em https://validar.iti.gov.br

Prof. Doutor Paul G. Kinas
(Avaliador – IMEF-FURG)

gov.br Documento assinado digitalmente
RODRIGO DA ROCHA GONCALVES
Data: 16/01/2024 18:17:43-0300
Verifique em https://validar.iti.gov.br

Prof. Doutor Rodrigo da Rocha Gonçalves
(Avaliador - ICEAC – FURG)