Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

Análise e Previsão de Séries Temporais via Facebook *Prophet*

Mateus Penteado Borges
Orientadora: Maria Sílvia de Assis Moura

Trabalho de Conclusão de Curso

Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

Análise e Previsão de Séries Temporais via Facebook *Prophet*

Mateus Penteado Borges
Orientadora: Maria Sílvia de Assis Moura

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do título de Bacharel em Estatística.

São Carlos
Fevereiro de 2024

Federal University of São Carlos
Exact and Technology Sciences Center
Department of Statistics

Time Series Analysis and Forecasting via Facebook's *Prophet*

Mateus Penteado Borges
Advisor: Maria Sílvia de Assis Moura

Bachelors dissertation submitted to the Department of Statistics, Federal University of São Carlos - DEs-UFSCar, in partial fulfillment of the requirements for the degree of Bachelor in Statistics.

São Carlos
January 2024

Mateus Penteado Borges

Análise e Previsão de Séries Temporais via Facebook *Prophet*

Este exemplar corresponde à redação final do trabalho de conclusão de curso devidamente corrigido e defendido por Mateus Penteado Borges e aprovado pela banca examinadora.

Aprovado em 29 de Janeiro de 2024

Banca Examinadora:

- Maria Sílvia de Assis Moura (Orientadora)
- Carlos Alberto Ribeiro Diniz
- Luis Ernesto Bueno Salasar

Dedico este trabalho ao meu avô, Orandi Alves Penteado, uma verdadeira fonte de inspiração em minha jornada. Suas histórias de vida, sabedoria e dedicação constante sempre me motivaram a perseguir meus objetivos com paixão e determinação.

À minha mãe, Maristela Carminato Penteado, expresso minha mais profunda gratidão. Seu apoio incondicional e exemplo de determinação me guiaram na escolha da graduação em Estatística, trilhando um caminho que reflete sua força e coragem.

Este trabalho é o resultado não apenas do meu esforço, mas também do amor, encorajamento e influência positiva que recebi de vocês. Que esta dedicação seja um testemunho do profundo respeito e apreço que sinto por ambos. Obrigado por moldarem quem sou e por me mostrarem a importância de seguir meus sonhos com amor e persistência.

Com carinho,

Mateus Penteado Borges

Resumo

A previsão é uma estimativa ou projeção do que pode acontecer no futuro, com base em informações e dados disponíveis do passado e presente. Em outras palavras, é uma tentativa de fazer uma suposição informada sobre o que provavelmente ocorrerá no futuro, utilizando informações anteriores e conhecimento do contexto atual. A previsão é aplicada em diversas áreas, como criptomoedas, finanças, clima, ciência, política e muito mais.

Na abordagem estatística, existem várias maneiras de realizar previsões. Um exemplo é o método de Alisamento Exponencial, conhecido por sua simplicidade e facilidade de compreensão, uma vez que suas fórmulas não apresentam grande complexidade. Além disso, temos as Redes Neurais, que são úteis para capturar relações não lineares presentes nos dados, melhorando a precisão das previsões.

A área de previsão é constantemente estudada e aprimorada pelos profissionais que a utilizam no dia a dia. Em 2017, dois cientistas de dados da empresa Facebook desenvolveram uma técnica inovadora denominada Facebook *Prophet*. Essa técnica foi criada para solucionar duas questões encontradas nos métodos tradicionais de previsão de séries temporais. A primeira diz respeito à inflexibilidade e fragilidade das técnicas de previsão completamente automáticas para incorporar suposições ou heurísticas úteis. A segunda questão está relacionada à dependência de um analista especializado e competente em ciência de dados para a utilização de ferramentas mais robustas.

Com a ascensão do tema das *criptomoedas* no mundo atual, há um interesse crescente por parte dos profissionais em obter modelos bem ajustados para as moedas digitais. Isso possibilita posicionar-se adequadamente em relação à questão da compra ou venda desses ativos.

No presente trabalho serão comparados três métodos úteis para previsões: Alisamento Exponencial (*Holt-Winters*), Redes Neurais e o Facebook *Prophet*. Serão utilizadas algumas técnicas de medidas de acurácia, tais como o Erro Médio Absoluto (EMA) e o Erro Quadrático Médio (EQM), para a seleção final do melhor modelo.

Palavras-chave: *Acurácia, Alisamento Exponencial, Criptomoedas, Facebook Prophet, Holt-Winters, Modelos, Previsão.*

Abstract

Forecasting is an estimate or projection of what may happen in the future based on information and data available from the past and present. In other words, it is an attempt to make an informed guess about what is likely to occur in the future, using previous information and knowledge of the current context. Forecasting is applied in various areas such as cryptocurrencies, finance, weather, science, politics, and more.

In the statistical approach, there are several ways to make forecasts. An example is the Exponential Smoothing method, known for its simplicity and ease of understanding, as its formulas are not overly complex. Additionally, we have Neural Networks, which are useful for capturing non-linear relationships present in the data, improving the accuracy of forecasts.

The field of forecasting is constantly studied and improved by professionals who use it in their daily work. In 2017, two data scientists from Facebook developed an innovative technique called Facebook *Prophet*. This technique was created to address two issues found in traditional time series forecasting methods. The first concerns the inflexibility and fragility of completely automatic forecasting techniques to incorporate useful assumptions or heuristics. The second issue is related to the dependence on a skilled and competent data science analyst to use more robust tools.

With the rise of cryptocurrencies in the current world, there is a growing interest among professionals in obtaining well-adjusted models for digital currencies. This enables them to position themselves appropriately regarding the issue of buying or selling these assets.

In this work, three useful forecasting methods will be compared: Exponential Smoothing (*Holt-Winters*), Neural Networks, and Facebook *Prophet*. Some accuracy measures such as Mean Absolute Error (MAE) and Mean Squared Error (MSE) will be used for the final selection of the best model.

Keywords: *Accuracy, Cryptocurrencies, Exponential Smoothing, Facebook Prophet, Forecasting, Holt-Winters, Models.*

Lista de Figuras

1.1 Tecnologia Blockchain (https://investidoringles.com/o-que-e-blockchain/) . 20

2.1 Representação de uma Rede Neural 27

4.1 Preço do Bitcoin (em USD) durante o período de 21/09/2023 até 21/12/2023 46
4.2 Previsão do preço do Bitcoin utilizando o método de *Holt-Winters*. 47
4.3 Previsão do preço do Bitcoin utilizando o método das Redes Neurais. . . . 47
4.4 Previsão do preço do Bitcoin utilizando o método Facebook *Prophet*. . . . 48
4.5 Preço do Ethereum (em USD) durante o período de 30/12/2018 até 30/12/2023. 49
4.6 Previsão do preço do Ethereum utilizando o método de *Holt-Winters* . . . 50
4.7 Previsão do preço do Ethereum utilizando o método de Redes Neurais . . . 50
4.8 Previsão do preço do Ethereum utilizando o *Prophet* 51

Lista de Tabelas

3.1 Comparação de Modelos para Diferentes Configurações (n de 60 a 300) . . 36

3.2 Comparação de Modelos para Diferentes Configurações (n = 1200 e n = 2100) 37

3.3 Comparação de Modelos para Diferentes Configurações (n = 1200 e n = 2100) 40

3.4 Comparação de Modelos para Diferentes Configurações (n = 1200 e n = 2100) 41

4.1 Medidas de Acurácia - Bitcoin 48

4.2 Medidas de Acurácia - Ethereum 51

Sumário

1 Introdução **19**

1.1 Criptomoedas 19

2 Materiais e Métodos **23**

2.1 Alisamento Exponencial 23

2.1.1 Descrição da Implementação Específica (`HoltWinters`) 25

2.1.2 Hiperparâmetros 25

2.2 Redes Neurais 26

2.2.1 Descrição da Implementação Específica (`nnetar`) 28

2.2.2 Fórmulas Matemáticas 28

2.2.3 Hiperparâmetros 28

2.3 Facebook *Prophet* 29

2.4 Escolha de Cada Método 31

2.5 Medidas de Acurácia 33

2.5.1 Erro Médio Absoluto 33

2.5.2 Porcentagem de Erro Médio Absoluto 33

2.5.3 Erro Quadrático Médio 34

2.5.4 Raiz do Erro Quadrático Médio 34

2.5.5 Porcentagem do Erro Quadrático Médio 34

3 Simulações **35**

3.1 Simulações de uma amostra 35

3.2 Simulações com adição de pontos discrepantes 39

3.3 Conclusão das Simulações 43

4 Aplicação em Dados Reais: Criptomoedas Bitcoin e Ethereum **45**

4.1 Aplicação - Bitcoin 46

4.2 Aplicação - Ethereum 48

5 Conclusão Final **53**

Referências Bibliográficas **55**

A Apêndice A **57**

Capítulo 1

Introdução

Previsão é uma estimativa ou projeção do que pode acontecer no futuro com base em informações e dados disponíveis do passado e presente, ou seja uma tentativa de fazer uma suposição informada sobre o que provavelmente ocorrerá no futuro, com base em informações anteriores e conhecimento do contexto atual. A previsão é usada em muitas áreas, incluindo criptomoedas, finanças, clima, ciência, política e muito mais.

Uma das maneiras de realizar uma previsão é por meio do uso da intuição e experiência, como, por exemplo, ao se perguntar: "Vai chover hoje à tarde?" Ao olhar para o céu e observar as nuvens, se você notar nuvens carregadas e escuras, isso pode indicar que há uma maior probabilidade de chuva. Além disso, é possível fazer previsões por meio de técnicas de análise de dados e métodos estatísticos, como as redes neurais, ou ainda modelos de alisamento exponencial.

Neste trabalho serão comparados alguns dos métodos de previsão já existentes na literatura, em específico o método de *Holt-Winters* e as Redes Neurais, com o método denominado *Facebook Prophet*, desenvolvido em 2017, a fim de aplicá-los em séries de dados financeiros reais que envolvem o preço diário das duas criptomoedas mais famosas até o momento: *Bitcoin* e *Ethereum*.

1.1 Criptomoedas

Criptomoeda é um tipo de moeda digital que utiliza técnicas de criptografia para garantir a segurança e a privacidade das transações, além de controlar a criação de novas unidades. Ao contrário das moedas tradicionais, que são emitidas e reguladas por governos ou instituições financeiras, as *"criptos"* são descentralizadas e operam em uma rede de

computadores chamada *blockchain*.

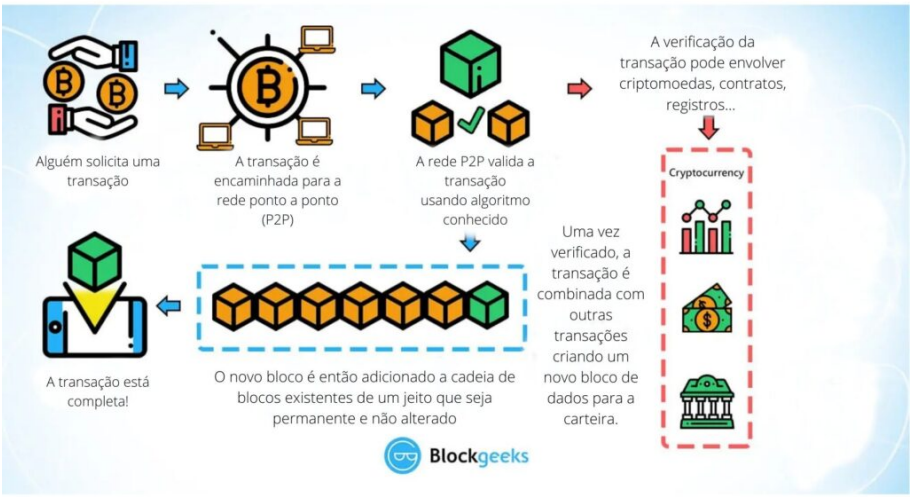

Figura 1.1: Tecnologia Blockchain (https://investidoringles.com/o-que-e-blockchain/)

Na Figura 1.1 temos o funcionamento de uma *blockchain*, um sistema de registro digital descentralizado que funciona como um livro de registros compartilhado. Nesse sistema, as transações são agrupadas em blocos, sendo que cada bloco contém um código único do bloco anterior. Isso forma uma cadeia contínua de registros, o que garante a segurança e a transparência das informações. Qualquer alteração em um bloco afetará todos os blocos seguintes, tornando a *blockchain* imutável e confiável.

Isso significa que as transações não dependem de intermediários, como bancos ou governos, e podem ser realizadas diretamente entre as partes envolvidas. Algumas das criptomoedas mais conhecidas incluem o famoso *Bitcoin*, criado pelo misterioso pseudônimo Satoshi Nakamoto (2008), e também o *Ethereum*, idealizado pelo programador russo-canadense Vitalik Buterin (2013).

As criptomoedas oferecem vantagens significativas, como descentralização, segurança e rapidez nas transações globais, além de promover a inclusão financeira. A utilização de criptografia avançada garante proteção e autenticidade nas transações, enquanto a eliminação de intermediários proporciona maior autonomia financeira aos usuários. Entretanto, alguns desafios precisam ser levados em conta, como a alta volatilidade dos preços, a limitada aceitação em estabelecimentos comerciais e a incerteza regulatória. Além disso, a falta de suporte governamental ainda é uma realidade presente nesse meio, o que pode gerar desconfiança em alguns usuários e impactar a adoção em larga escala.

Por se tratar de uma série financeira, um dos objetivos em finanças é a avaliação de riscos de uma carteira de ativos financeiros. Frequentemente, é preferível trabalhar com

log-retornos dos ativos para a modelagem da volatilidade (variância condicional de uma variável) em estudo. De acordo com Morettin (2006b), existem cinco fatos estilizados relativos a retornos financeiros :

- Os retornos são, em geral, não-auto-correlacionados;
- Os quadrados dos retornos são auto-correlacionados e apresentam uma correlação de *lag* 1 pequena e depois uma queda lenta das demais;
- Séries de retornos apresentam volatilidade ao longo do tempo;
- A distribuição dos retornos apresenta caudas mais pesadas do que uma distrinuição normal. Embora aproximadamente simétrica, é em geral leptocúrtica;
- Algumas séries de retornos são não-lineares.

Esse documento está estruturado da seguinte maneira: no Capítulo 2 apresentamos os métodos de previsão: Alisamento Exponencial, Redes Neurais e Facebook *Prophet* e também as medidas de acurácia que serão utilizadas posteriormente para a comparação do desempenho dos modelos. No Capítulo 3, conduzimos diversas simulações de dados e discutimos o desempenho de cada um dos modelos em um ambiente controlado. Em seguida, no Capítulo 4, realizamos uma aplicação em dados reais, de forma a analisar a capacidade preditiva dos três modelos em estudo. Finalmente, no Capítulo 5, discutimos todos os resultados obtidos ao longo do trabalho.

Capítulo 2

Materiais e Métodos

Neste capítulo, será definido os conceitos de Alisamento Exponencial, Redes Neurais e também a introdução do Facebook *Prophet*, assim como uma exemplificação de cada um dos métodos utilizados. Ademais, será introduzido os conceitos de Medidas de Acurácia, tais como o Erro Médio Absoluto (EMA) e o Erro Quadrático Médio (EQM), que serão úteis para a comparação entre os modelos.

2.1 Alisamento Exponencial

Segundo Morettin P.A e Toloi C.M.C (2006), o método de Alisamento Exponencial foi introduzido no final da década de 1950 por Charles C. Holt, um famoso estatístico e professor americano que contribuiu significativamente para a área de previsão em dados de séries de tempo. Posteriormente, o método foi aprimorado por Peter Winters, aluno de Holt que também ficou conhecido pelo desenvolvimento do método de Alisamento Exponencial.

Historicamente existem três técnicas relacionadas a esse método. A primeira, proposta por Holt, consistia na previsão de séries temporais em que os dados não apresentavam componentes de tendência e/ou sazonalidade. Essa técnica foi chamada de "Alisamento Exponencial Simples". Uma equação que representa esse método é:

$$\bar{Z}_t = \lambda \bar{Z}_{t-1} + (1 - \lambda) Z_t, \tag{2.1}$$

em que:

- $\bar{Z}_{t+1}$ é a previsão para o próximo período,

- Z_t é o valor observado na série temporal no período t,
- $\bar{Z}_t$ é a previsão anterior para o período t (ou o valor inicial para $t = 1$),
- λ é o fator de suavização exponencial, um valor entre 0 e 1 que controla a contribuição relativa do valor observado Y_t e da previsão anterior Z_t.

A segunda técnica, aprimorada também por Holt (1957) e denominada *Holt's linear trend method*, é uma extensão da técnica anterior, porém com a capacidade de ser útil para a previsão de dados que contenham o componente de tendência. Assim, temos que esse método é similar ao Alisamento Exponencial Simples, com a diferença de que agora é utilizado uma nova constante de suavização para modelar a tendência da série.

A fórmula para os valores do nível $\bar{Z}_t$ e de tendencia $\hat{T}_t$ é dada por:

$$\bar{Z}_t = \alpha Z_t + (1 - \alpha)(\bar{Z}_{t-1} + \hat{T}_{t-1}), \tag{2.2}$$

$$\hat{T}_t = \beta(\bar{Z}_t - \bar{Z}_{t-1}) + (1 - \beta)\hat{T}_{t-1}, \tag{2.3}$$

em que

- $\hat{T}_t$ é a taxa de crescimento estimada no tempo t,
- β é o fator de suavização para a tendência, em que $0 \leq \beta \leq 1$,
- Z_t é o nível estimado no tempo t,
- Z_{t-1} é o nível estimado no tempo $t - 1$,
- $\hat{T}_{t-1}$ é a taxa de crescimento estimada no tempo $t - 1$.

Note que ambas as técnicas supracitadas, ainda não suportavam dados que continham sazonalidade. Em 1960, com uma colaboração de Peter Winters, Holt resolveu esse problema com o desenvolvimento de uma a terceira técnica: *Holt-Winters' additive method*, esta que captura a sazonalidade e tendência para a previsão dos dados. Assim, mantendo as equações 2.2 e 2.3, adiciona-se uma nova equação para a sazonalidade ($\hat{S}_t$):

$$\hat{S}_t = \gamma \cdot (Z_t - \bar{Z}) + (1 - \gamma)\hat{S}_{t-s}, \tag{2.4}$$

em que

- $\hat{S}_t$ é o componente de sazonalidade no tempo t,
- γ é o fator de suavização para a sazonalidade, onde $0 \leq \gamma \leq 1$,
- s é o comprimento do ciclo sazonal.

Essa técnica é amplamente utilizada para dados que possuem componentes de tendência e/ou sazonalidade. Ela aplica pesos exponenciais decrescentes nos valores mais antigos, dando maior importância aos dados mais recentes. Isso significa que os valores mais atuais têm um impacto maior nas previsões.

Uma das vantagens principais do método, segundo Morettin P.A e Toloi C.M.C (2006), é a simplicidade e o fácil entendimento da mesma, haja vista que as fórmulas não apresentam tamanha complexidade. Ademais, pode-se citar a versatilidade que é permitida pela variação dos parâmetros e também a capacidade de produzir bons estimadores a curto prazo. No entanto, o método de *Holt-Winters* pode ser lento para o ajuste de novas tendências e também há dificuldades para determinar os valores desses parâmetros.

2.1.1 Descrição da Implementação Específica (`HoltWinters`)

A função `HoltWinters` do pacote `stats` em R é uma implementação do método de *Holt-Winters* para filtragem de séries temporais. Este método é projetado para prever séries temporais levando em consideração componentes de nível, tendência e sazonalidade. A implementação utiliza otimização para determinar os parâmetros desconhecidos, minimizando o erro quadrático da previsão.

2.1.2 Hiperparâmetros

A função `HoltWinters` permite a especificação de hiperparâmetros que afetam a modelagem da série temporal:

- **alpha:** Parâmetro de suavização para o componente de nível ($0 \leq \alpha \leq 1$).
- **beta:** Parâmetro de suavização para o componente de tendência ($0 \leq \beta \leq 1$).
- **gamma:** Parâmetro de suavização para o componente sazonal ($0 \leq \gamma \leq 1$).
- **seasonal:** Modelo sazonal, escolhendo entre "additive"ou "multiplicative".

Esses parâmetros controlam a contribuição relativa de cada componente na previsão.

Caso o leitor deseje aprofundar seus conhecimentos, recomenda-se consultar o livro de Holt (1957), referência fundamental para o método de Holt-Winters.

A implementação `HoltWinters` é escolhida por sua capacidade de modelar níveis, tendências e sazonalidades em séries temporais, tornando-a adequada para uma ampla variedade de aplicações. No entanto, é importante considerar a interpretação dos resultados, especialmente devido à dependência de otimização para determinar os parâmetros desconhecidos.

2.2 Redes Neurais

Outra técnica útil em previsões são as Redes Neurais, que conforme referenciado por Carvalho A. (2023), é um modelo de aprendizado de máquina composto por unidades chamadas de neurônios artificiais, que são organizados em camadas e conectados por pesos sinápticos. O nome do modelo vem da biologia, pois as redes neurais são inspiradas no funcionamento de um neurônio. Na biologia, o sistema nervoso é composto por células altamente especializadas chamadas neurônios, que desempenham um papel fundamental no funcionamento e no comportamento do corpo humano, bem como no processamento de informações.

Os neurônios são compostos por três partes principais: dendritos, corpo celular e axônios. Os dendritos são ramificações que atuam como terminais de entrada, recebendo sinais elétricos e químicos de outros neurônios ou de receptores sensoriais. O corpo celular contém o núcleo do neurônio e é onde ocorrem os processos metabólicos essenciais para o funcionamento da célula. Os axônios, por sua vez, são prolongamentos longos que transmitem os sinais elétricos gerados pelo neurônio para outros neurônios, músculos ou glândulas.

Estatisticamente, cada rede neural é formada por várias unidades de funcionamento (neurônios), conectadas por pesos (axônios) e organizados em camadas. A primeira camada, denominada de **Camada de Entrada** é quando os padrões são apresentados a rede. Em seguida, temos a **Camada Intermediária ou Escondida**, que é onde é feita a maior parte do processamento, através das conexões ponderadas. E por fim, a **Camada**

de Saída, que é onde o resultado final é concluído e apresentado.

Na Figura 2.1 está apresentado uma representação visual do funcionamento de uma rede neural.

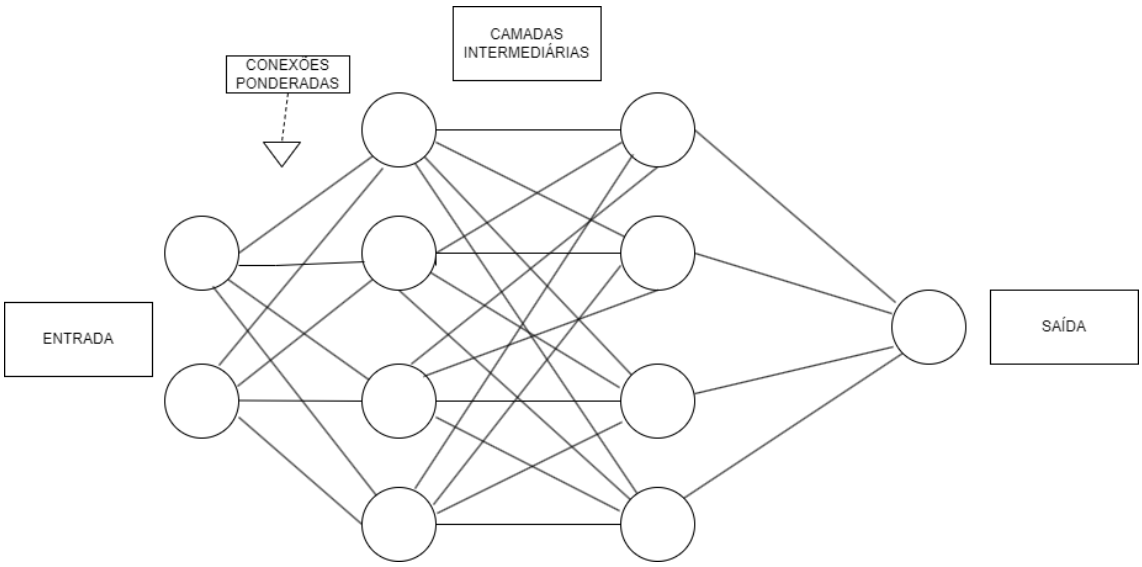

Figura 2.1: Representação de uma Rede Neural

As redes neurais têm sido amplamente empregadas como técnica de previsão para séries temporais devido às suas vantagens notáveis. Em primeiro lugar, a capacidade de aprendizado de padrões complexos permite que esses modelos capturem relações não lineares presentes nos dados, melhorando a precisão das previsões. Em segundo lugar, a flexibilidade das redes neurais as torna adequadas para diversos problemas de previsão de séries temporais, desde séries univariadas até multivariadas e com dados irregulares. Por fim, a capacidade de lidar com conjuntos de dados de alta dimensionalidade possibilita o processamento de múltiplas séries temporais relacionadas. No entanto, também existem desvantagens a serem consideradas. Em primeiro lugar, o treinamento de redes neurais requer uma grande quantidade de dados, o que pode ser uma limitação em cenários com disponibilidade limitada de informações. Além disso, a complexidade do modelo pode dificultar a interpretação dos resultados, prejudicando a explicabilidade do sistema. Por fim, a tendência das redes neurais de sofrer o super-ajuste, denominado também de *overfitting*, pode levar a uma pior generalização para novos dados, exigindo uma atenção cuidadosa na calibração dos hiperparâmetros durante o processo de treinamento.

2.2.1 Descrição da Implementação Específica (`nnetar`)

A função `nnetar` do pacote `forecast` em R é uma implementação de uma rede neural *feedforward* específica para previsão de séries temporais. A arquitetura da rede é composta por três camadas principais:

- **Camada de Entrada:** Recebe as entradas, que podem ser defasagens da série temporal (x_1, x_2, x_3, x_4) e outros regressores externos, se fornecidos.
- **Camada Oculta:** Realiza o processamento principal da rede. A combinação linear das entradas para o neurônio oculto j pode ser expressa como:

$$z_j = b_j + \sum_{i=1}^{4} w_{i,j} x_i$$

  Em seguida, a saída da camada oculta é modificada por uma função não linear, geralmente a função sigmoidal:

$$s(z) = \frac{1}{1 + e^{-z}}$$

- **Camada de Saída:** Produz as previsões finais.

2.2.2 Fórmulas Matemáticas

As fórmulas matemáticas envolvidas no processamento da camada oculta são fundamentais para entender o funcionamento da rede. A combinação linear (z_j) é a soma ponderada das entradas (x_i) com pesos $(w_{i,j})$ e um termo de viés (b_j). A função sigmoidal $(s(z))$ introduz não linearidades, o que é crucial para a capacidade da rede neural de modelar padrões complexos nos dados.

2.2.3 Hiperparâmetros

A função `nnetar` permite a especificação de alguns hiperparâmetros:

- **p:** Dimensão de incorporação para séries temporais não sazonais.
- **P:** Número de defasagens sazonais usadas como entradas.
- **k:** Número de neurônios na camada oculta.

Esses hiperparâmetros afetam a arquitetura da rede e podem ser ajustados manualmente ou selecionados automaticamente pela função, dependendo da configuração.

Caso o leitor deseje aprofundar seus conhecimentos, recomenda-se consultar o livro de Hyndman (2014). Este recurso abrange princípios fundamentais e práticas avançadas em previsão, fornecendo uma compreensão abrangente do campo.

A implementação `nnetar` é escolhida por suas vantagens notáveis, incluindo a capacidade de aprendizado de padrões complexos e flexibilidade para lidar com vários tipos de séries temporais. No entanto, é essencial considerar a necessidade de uma quantidade substancial de dados para o treinamento e as potenciais dificuldades na interpretação do modelo devido à sua complexidade.

2.3 Facebook *Prophet*

O Facebook, fundado por Mark Zuckerberg em 2004, é uma das redes sociais mais populares globalmente. Inicialmente concebida como uma plataforma de interação social para estudantes universitários, o Facebook expandiu rapidamente seu alcance, contando atualmente com bilhões de usuários ativos mensais, de acordo com o Canaltech (2023). Essa plataforma oferece diversos recursos, possibilitando aos usuários compartilhar *posts*, interagir com amigos e familiares, entre outras atividades. Além disso, o Facebook se tornou uma ferramenta essencial para muitas empresas e profissionais de marketing, oferecendo oportunidades de publicidade direcionada e ampliação do alcance de audiência.

Dentre as áreas de estudo e exploração realizadas pelo Facebook, destaca-se a previsão de séries temporais e a ciência de dados. Em fevereiro de 2017, Sean J. Taylor e Ben Letham desenvolveram internamente uma biblioteca de código aberto denominada Facebook *Prophet (2017),* a qual está disponível nas linguagens de programação Python e R. Essa ferramenta foi criada para solucionar duas questões encontradas nos métodos tradicionais de previsão de séries temporais. A primeira delas diz respeito à inflexibilidade e fragilidade das técnicas de previsão completamente automáticas para incorporar suposições ou heurísticas úteis. A segunda questão está relacionada à dependência de um analista especializado e competente em ciência de dados para o uso de ferramentas mais robustas.

Justamente por ser criada por cientistas de dados do Facebook, a ferramenta é otimizada para as tarefas de previsão utilizadas no dia a dia da empresa, com características

como:

- Observações registradas em intervalos horários, diários ou semanais, com um histórico ideal de um ano;
- Presença de sazonalidades distintas que seguem padrões "humanos", como variações ao longo da semana e ao longo do ano;
- Ocorrência de feriados importantes, mesmo que em intervalos irregulares, desde que essas datas sejam conhecidas antecipadamente;
- Casos em que há um número significativo de observações faltantes ou valores discrepantes que fogem do padrão;
- Presença de mudanças na tendência histórica, como lançamentos de produtos ou alterações nos registros;
- Tendências que seguem curvas de crescimento não lineares, nas quais a evolução da série temporal atinge um limite natural ou se estabiliza.

Essas características tornam o método uma ferramenta atrativa para a previsão de séries temporais, pois simplifica o processo de criação de previsões precisas e oferece opções de personalização intuitivas mesmo para não especialistas. Dessa forma, o *Prophet* se destaca como uma alternativa eficiente e acessível no campo da previsão de séries temporais, com potencial aplicação em diversas áreas, como demanda de produtos, tráfego em sites e tendências de uso e engajamento nas redes sociais, incluindo o Facebook.

Essencialmente, o método é um modelo de regressão aditivo, ou seja, é uma soma de vários componentes, sendo eles o componente de tendência; as curvas de sazonalidade anual, mensal e diária; e os efeitos de feriados/datas especiais.

$$\bar{Z}_t = T(t) + S(t) + h(t) + \epsilon_t, \tag{2.5}$$

em que:

- $T(t)$ é o componente de tendência,
- $S(t)$ é o componente de sazonalidade,
- $h(t)$ é os efeitos de feriados,

- ϵ_t é o ruído branco.

Ainda segundo Sean J. Taylor, o método se generaliza bem para a maioria das séries temporais. Os choques e a não estacionariedade são capturados pelo componente de tendência, enquanto a sazonalidade diária e anual, pode ser capturada combinando várias séries de Fourier. Ainda, os efeitos de feriado são capturados como regressores binários, o que se torna útil pois os feriados muitas vezes têm um efeito distinto em muitos tipos de dados de séries temporais.

Na mesma vertente, segundo os desenvolvedores do método, o Prophet demonstra sua eficácia ao gerar previsões tão precisas quanto as produzidas por especialistas em previsão, com uma notável redução de esforço. Este método permite uma flexibilidade única, possibilitando que analistas sem treinamento específico em métodos de séries temporais ajustem ou melhorem as previsões por meio de parâmetros facilmente interpretáveis. O *Prophet*, conforme destacado por Taylor e Letham, simplifica significativamente a criação de previsões precisas e razoáveis, oferecendo uma abordagem acessível mesmo diante da complexidade de escolha entre diferentes técnicas de previsão disponíveis no pacote, como ARIMA e suavização exponencial.

Como já citado, o *Prophet* é um método recente para auxiliar em uma previsão de série temporal que está constantemente evoluindo e se adaptando. Segundo Cuong Duong (2023), contribuidor para novas atualizações do *Prophet*, houve muitas ideias para aprimoramentos adicionais no modelo e, embora úteis, não há planos para serem incrementadas.

Finalmente, o *Prophet* funciona melhor com séries temporais que apresentam fortes efeitos sazonais e que possuem vários anos de dados históricos. O método é robusto em relação a dados faltantes e deslocamentos na tendência, e geralmente lida bem com valores atípicos. Para informações sobre a biblioteca do *Prophet* disponível no R, recomenda-se acessar o vídeo disponibilizado por Taylor e Letham (2017) neste `link`.

2.4 Escolha de Cada Método

Ao selecionar um método para previsão de séries temporais, é crucial considerar as características específicas dos dados e os requisitos do problema. Abaixo, tem-se orientações sobre quando cada método pode ser mais apropriado:

- ***Holt-Winters*:** Utilize o método de *Holt-Winters* em situações em que os dados apresentam padrões sazonais bem definidos. Este método é eficaz quando há uma

tendência linear ou exponencial nos dados, sendo ideal para considerar efeitos de sazonalidade tanto a curto quanto a longo prazo.

- **Redes Neurais:** Considere o uso de redes neurais quando os dados exibem padrões complexos ou não lineares. As redes neurais, implementadas na função `nnetar` do pacote `forecast` em R, oferecem uma abordagem flexível para previsão de séries temporais. Essa implementação utiliza uma rede neural *feedforward* e é escolhida pela capacidade de aprendizado de padrões complexos.

- **Prophet:** Escolha o *Prophet* em situações de séries temporais com múltiplas sazonalidades, especialmente aquelas de "escala humana"(diárias, semanais, anuais). É uma opção eficaz quando há feriados ou eventos especiais que afetam os padrões dos dados. O Prophet é particularmente útil para previsões em ambientes de negócios, oferecendo resultados precisos com configurações intuitivas e facilmente interpretáveis.

Lembre-se de ajustar essas orientações de acordo com as características específicas do seu conjunto de dados e os objetivos da sua previsão de séries temporais.

2.5 Medidas de Acurácia

A medida de acurácia é uma ferramenta fundamental para determinar a eficácia de modelos e algoritmos, pois fornece uma avaliação objetiva do quão bem um modelo está realizando previsões corretas em relação ao total de amostras avaliadas. Neste trabalho de conclusão de curso serão exploradas diferentes medidas de acurácia com o objetivo de analisar e comparar o desempenho de modelos em uma tarefa de classificação.

As medidas de acurácia selecionadas são o Erro Médio Absoluto (EMA), a Porcentagem de Erro Médio Absoluto (PEMA), o Erro Quadrático Médio (EQM), a Raiz do Erro Quadrático Médio (REQM) e a Porcentagem do Erro Quadrático Médio (PEQM). Foi utilizado o R para o cálculo das medidas selecionadas, sem a utilização de uma biblioteca já pronta. Os códigos estão disponíveis no Apêndice A.

Ademais, todas as medidas calculadas se basearam em uma divisão do conjunto de dados em treinamento e teste, isto é, comparamos o valor previsto do conjunto de teste com o verdadeiro valor da observação também do conjunto de teste.

2.5.1 Erro Médio Absoluto

O Erro Médio Absoluto (EMA) é uma medida que representa a média das diferenças absolutas entre os valores previstos e os valores reais. A fórmula é dada por:

$$\text{EMA} = \frac{1}{n}\sum_{i=1}^{n}|y_t - \hat{y}_t|, \tag{2.6}$$

em que:

- EMA é o Erro Médio Absoluto,
- n representa o número total de observações ou amostras,
- y_t é o valor observado da série,
- $\hat{y}_t$ é o valor previsto.

2.5.2 Porcentagem de Erro Médio Absoluto

A Porcentagem de Erro Médio Absoluto (PEMA) expressa o EMA como uma porcentagem em relação ao valor real, possibilitando uma compreensão mais intuitiva do erro médio relativo ao contexto do problema. Basicamente, temos:

$$PEMA = \frac{EMA}{\text{valor real}} \times 100\% \tag{2.7}$$

2.5.3 Erro Quadrático Médio

O Erro Quadrático Médio (EQM) é uma medida que calcula a média dos erros quadráticos entre as previsões e os valores reais. Essa métrica enfatiza erros maiores, devido ao uso do quadrado das diferenças. A fórmula do EQM é dada por:

$$EQM = \frac{1}{n}\sum_{i=1}^{n}(valor_{real_i} - valor_{previsto_i})^2, \tag{2.8}$$

em que

- n representa o número total de amostras.

2.5.4 Raiz do Erro Quadrático Médio

A Raiz do Erro Quadrático Médio (REQM) é a raiz quadrada do EQM, proporcionando uma métrica na mesma escala dos valores reais e facilitando a interpretação. A fórmula é dada por:

$$REQM = \sqrt{EQM} \tag{2.9}$$

2.5.5 Porcentagem do Erro Quadrático Médio

A Porcentagem do Erro Quadrático Médio (PEQM) permite avaliar o erro quadrático médio relativo em relação ao valor real, fornecendo uma perspectiva percentual do desempenho do modelo na tarefa de previsão. A fórmula é dada por:

$$PEQM = \frac{EQM}{\text{valor real}} \times 100\% \tag{2.10}$$

Capítulo 3

Simulações

3.1 Simulações de uma amostra

A condução de simulações desempenha um papel crucial na avaliação e comparação de modelos de previsão em séries temporais. Ao gerar séries temporais simuladas com características conhecidas e específicas, é possível criar um ambiente controlado para testar o desempenho de diferentes modelos. Neste contexto, optamos por simular séries temporais que exibem uma sazonalidade fixa (S=12) e um componente autorregressivo, que pode ser $sar = \{0.7, 0.8, 0.9\}$, com o intuito de avaliar o desempenho dos modelos *Holt-Winters*, Redes Neurais e *Prophet*. Através da comparação das previsões obtidas por esses modelos com os dados simulados, almejamos avaliar a eficácia do *Prophet*, em particular, em séries temporais com padrões sazonais e autocorrelação - cenários frequentemente encontrados em aplicações práticas de previsão.

Primeiramente, simulamos uma amostra de tamanho (**n=60**), utilizando o comando ***sarima.sim(sar=0.7, S=12, n=60)***, com parâmetros já mencionados anteriormente.

Para continuar a avaliação do método do Facebook *Prophet*, foi simulado um conjunto de 1000 amostras, variando o tamanho da amostra (n) para explorar diferentes cenários. No total, foram criados 15 cenários, tais como:

- 1000 amostras com $n = 60$ e $sar = 0.7$,
- 1000 amostras com $n = 60$ e $sar = 0.8$,
- 1000 amostras com $n = 60$ e $sar = 0.9$,
- 1000 amostras com $n = 120$ e $sar = 0.7$,

- 1000 amostras com $n = 120$ e $sar = 0.8$,

e assim por diante. Essa abordagem abrange uma ampla gama de tamanhos de amostra, permitindo uma análise abrangente da capacidade de cada modelo em lidar com diferentes padrões e características das séries temporais. Com um grande número amostral, é possível obter uma estimativa mais precisa da capacidade de cada modelo em lidar com diferentes padrões e características das séries temporais, contribuindo para a validação da robustez e eficácia dos modelos de previsão diante de cenários diversos.

Os resultados obtidos são mostrados nas tabelas abaixo.

Tabela 3.1: Comparação de Modelos para Diferentes Configurações (n de 60 a 300)

n	***sar***	**Modelo**	**EMA**	**PEMA**	**EQM**	**PEQM**	**REQM**
60	**0.7**	**Holt-Winters**	1.60	337.61	4.03	848.53	1.97
	0.7	**Redes Neurais**	1.59	335.56	6.00	1263.50	1.98
	0.7	**Prophet**	9.85	2072.66	223.13	46971.09	11.96
60	**0.8**	**Holt-Winters**	1.81	7940.46	5.18	22727.29	2.23
	0.8	**Redes Neurais**	1.77	7785.63	5.46	23952.29	2.21
	0.8	**Prophet**	11.26	49410.49	282.95	1241932.95	13.59
60	**0.9**	**Holt-Winters**	2.45	244.53	9.20	917.43	2.98
	0.9	**Redes Neurais**	2.44	243.33	9.49	945.60	2.98
	0.9	**Prophet**	13.28	1323.98	383.10	38181.81	15.93
120	**0.7**	**Holt-Winters**	1.46	604.74	3.34	1385.42	1.80
	0.7	**Redes Neurais**	1.26	522.94	2.96	1225.38	1.58
	0.7	**Prophet**	15.16	6284.44	479.85	198910.63	17.65
120	**0.8**	**Holt-Winters**	1.69	1605.89	4.48	4258.57	2.08
	0.8	**Redes Neurais**	1.52	1445.43	4.01	3820.48	1.90
	0.8	**Prophet**	16.90	16083.38	598.80	569798.66	19.61
120	**0.9**	**Holt-Winters**	2.19	845.76	7.50	2901.51	2.69
	0.9	**Redes Neurais**	2.07	800.80	7.06	2732.76	2.57
	0.9	**Prophet**	21.07	8149.83	930.39	359914.33	24.30
300	**0.7**	**Holt-Winters**	1.67	1162.91	4.35	3021.27	2.06
	0.7	**Redes Neurais**	1.44	1002.90	3.26	2262.96	1.78
	0.7	**Prophet**	1.62	1121.60	4.04	2806.43	1.94
300	**0.8**	**Holt-Winters**	1.95	763.30	5.93	2325.45	2.40
	0.8	**Redes Neurais**	1.73	677.05	4.74	1859.88	2.15
	0.8	**Prophet**	1.86	729.07	5.38	2112.10	2.23
300	**0.9**	**Holt-Winters**	2.53	553.51	10.18	2223.18	3.14
	0.9	**Redes Neurais**	2.33	508.19	8.63	1884.69	2.90
	0.9	**Prophet**	2.40	523.89	9.06	1978.00	2.88

Tabela 3.2: Comparação de Modelos para Diferentes Configurações (n = 1200 e n = 2100)

n	*sar*	Modelo	EMA	PEMA	EQM	PEQM	REQM
1200	**0.7**	**Holt-Winters**	1.69	1241.27	4.61	3394.77	2.09
	0.7	**Redes Neurais**	1.41	1037.47	3.15	2317.33	1.77
	0.7	**Prophet**	1.07	791.72	1.87	1373.70	1.36
1200	**0.8**	**Holt-Winters**	2.00	1053.50	6.39	3371.75	2.47
	0.8	**Redes Neurais**	1.74	915.76	4.74	2498.91	2.17
	0.8	**Prophet**	1.28	674.07	2.62	1384.28	1.62
1200	**0.9**	**Holt-Winters**	2.66	829.70	11.23	3495.88	3.29
	0.9	**Redes Neurais**	2.45	763.11	9.36	2914.22	3.04
	0.9	**Prophet**	1.77	550.59	4.90	1524.80	2.20
2100	**0.7**	**Holt-Winters**	1.78	866.68	5.18	2513.58	2.20
	0.7	**Redes Neurais**	1.42	690.32	3.15	1531.11	1.77
	0.7	**Prophet**	1.09	531.49	1.88	910.60	1.37
2100	**0.8**	**Holt-Winters**	2.09	626.74	7.05	2111.74	2.58
	0.8	**Redes Neurais**	1.77	528.49	4.84	1448.12	2.19
	0.8	**Prophet**	1.30	388.86	2.65	792.51	1.62
2100	**0.9**	**Holt-Winters**	2.76	404.18	12.14	1778.50	3.40
	0.9	**Redes Neurais**	2.52	368.55	9.84	1441.20	3.12
	0.9	**Prophet**	1.77	259.93	5.01	734.29	2.21

De acordo com a Tabela 3.1, ao considerar $n = 60$, independentemente do valor de *sar*, observa-se que o *Prophet* exibe consistentemente um desempenho inferior em comparação com os modelos *Holt-Winters* e Redes Neurais. Tomando como exemplo *sar = 0.7*, notamos que o *Prophet* apresenta valores mais elevados em todas as métricas avaliadas, incluindo EMA, PEMA, EQM, PEQM e REQM. Essa tendência persiste para *sar = 0.8* e *sar = 0.9*, indicando que, para amostras menores, o *Prophet* pode não ser tão eficaz quanto os outros dois modelos.

Aumentando o tamanho da amostra para $n = 120$ e $n = 300$, observamos uma melhoria geral no desempenho de todos os modelos. No entanto, mesmo nessas configurações, o *Prophet* continua a exibir valores mais altos em comparação com os outros dois modelos. Destacamos, por exemplo, que para $n = 300$ e *sar = 0.7*, embora o *Prophet* tenha reduzido as discrepâncias em algumas métricas, ainda não alcançou a paridade com as Redes Neurais.

Já de acordo com a Tabela 3.2, ao realizar simulações com maiores tamanhos amostrais, o *Prophet* destacou-se como uma escolha eficaz para as previsões temporais nos cenários analisados. Para $n = 1200$, o método apresentou o menor *EMA* e *EQM* quando

comparado ao *Holt-Winters* e às Redes Neurais, independentemente do valor utilizado para o componente autoregressivo (sar). Essa diferença torna-se ainda mais evidente ao aumentarmos o tamanho amostral para $n = 2100$, em que observamos uma performance superior em relação aos outros dois modelos, com o *Prophet* obtendo os menores valores de *EMA* e *EQM* para todos os cenários.

Finalmente, é importante ressaltar que o tempo de processamento no R para as previsões das simulações de tamanho n = 1200 e n = 2100 foram superiores a seis horas. Para amostras menores, esse tempo variou de dez minutos até uma hora completa.

3.2 Simulações com adição de pontos discrepantes

Além da simulação de séries temporais com padrões conhecidos, incorporamos uma análise mais abrangente considerando a presença de pontos discrepantes em nossas amostras. A inclusão de pontos discrepantes é crucial, uma vez que esses valores atípicos podem ter um impacto significativo nas capacidades preditivas dos modelos. Esses pontos, representados por desvios substanciais dos padrões esperados, desafiam a capacidade dos modelos de previsão em lidar com situações inesperadas.

Ao introduzir pontos discrepantes em nossa metodologia, buscamos avaliar como o *Prophet* responde a eventos inesperados e se consegue captar mudanças abruptas nas séries temporais, quando comparado aos modelos *Holt-Winters* e Redes Neurais. A introdução desses pontos fora do padrão visa fornecer uma visão mais realista do desempenho dos modelos em ambientes dinâmicos e complexos.

Adicionamos uma camada adicional à nossa abordagem, incorporando pontos discrepantes nas séries simuladas. Para implementar essa variação, optamos por uma estratégia que substitui três valores atípicos, localizados nos quantis 25%, 50%, e 75%, respectivamente. Cada um desses pontos discrepantes foi definido como sendo igual ao dobro do valor máximo presente na série temporal. Essa escolha específica desafia a capacidade dos modelos de previsão em lidar com valores extremos, ampliando o escopo da análise para além de padrões regulares.

Os resultados obtidos são mostrados nas Tabelas 3.3 e 3.4 a seguir.

Tabela 3.3: Comparação de Modelos para Diferentes Configurações (n = 1200 e n = 2100)

n	***sar***	**Modelo**	**EMA**	**PEMA**	**EQM**	**PEQM**	**REQM**
60	**0.7**	**Holt-Winters**	2.07	270.33	7.18	937.76	2.60
	0.7	**Redes Neurais**	2.02	263.57	13.57	1773.09	2.72
	0.7	**Prophet**	18.53	2421.33	807.14	105491.58	23.68
60	**0.8**	**Holt-Winters**	2.40	252.56	9.35	985.28	2.97
	0.8	**Redes Neurais**	2.27	238.59	12.67	1334.41	2.98
	0.8	**Prophet**	20.54	2163.09	993.05	104601.10	26.24
60	**0.9**	**Holt-Winters**	2.97	617.52	13.46	2802.53	3.62
	0.9	**Redes Neurais**	3.28	682.09	21.92	4562.24	4.24
	0.9	**Prophet**	26.79	5576.62	1642.52	341897.57	33.65
120	**0.7**	**Holt-Winters**	1.56	8888.62	4.44	25247.06	2.09
	0.7	**Redes Neurais**	1.46	8326.71	4.40	25004.01	2.04
	0.7	**Prophet**	25.77	146520.53	1445.68	8218663.22	30.32
120	**0.8**	**Holt-Winters**	1.82	1363.84	5.98	4478.03	2.42
	0.8	**Redes Neurais**	1.73	1292.69	5.87	4401.13	2.37
	0.8	**Prophet**	29.92	22419.09	1939.74	1453583.54	35.13
120	**0.9**	**Holt-Winters**	2.29	454.45	9.12	1807.82	2.98
	0.9	**Redes Neurais**	2.23	442.99	10.66	2112.16	2.97
	0.9	**Prophet**	38.09	7549.95	3155.56	625486.99	44.58
300	**0.7**	**Holt-Winters**	1.67	2473.32	4.56	6761.75	2.12
	0.7	**Redes Neurais**	1.47	2173.70	3.60	5337.66	1.88
	0.7	**Prophet**	1.69	2511.16	4.69	6963.25	2.11
300	**0.8**	**Holt-Winters**	1.94	1208.88	6.21	3867.24	2.47
	0.8	**Redes Neurais**	1.73	1076.20	5.10	3178.06	2.24
	0.8	**Prophet**	1.95	1216.42	6.25	3895.53	2.43
300	**0.9**	**Holt-Winters**	2.52	747.26	10.39	3086.60	3.19
	0.9	**Redes Neurais**	2.36	699.29	9.28	2754.35	3.01
	0.9	**Prophet**	2.50	743.79	10.21	3030.73	3.08

Tabela 3.4: Comparação de Modelos para Diferentes Configurações (n = 1200 e n = 2100)

n	*sar*	Modelo	EMA	PEMA	EQM	PEQM	REQM
1200	**0.7**	**Holt-Winters**	1.66	1018.87	4.59	2808.99	2.09
	0.7	**Redes Neurais**	1.44	884.37	3.38	2072.18	1.83
	0.7	**Prophet**	1.10	674.74	2.08	1276.46	1.44
1200	**0.8**	**Holt-Winters**	1.97	897.62	6.31	2875.94	2.46
	0.8	**Redes Neurais**	1.77	804.62	4.99	2272.27	2.22
	0.8	**Prophet**	1.32	599.50	2.93	1332.78	1.70
1200	**0.9**	**Holt-Winters**	2.64	734.52	11.15	3103.04	3.29
	0.9	**Redes Neurais**	2.48	691.00	9.75	2713.68	3.10
	0.9	**Prophet**	1.82	507.69	5.44	1513.21	2.31
2100	**0.7**	**Holt-Winters**	1.76	920.47	5.08	2653.22	2.18
	0.7	**Redes Neurais**	1.45	759.31	3.34	1743.62	1.82
	0.7	**Prophet**	1.12	585.41	2.03	1062.57	1.42
2100	**0.8**	**Holt-Winters**	2.07	653.14	6.95	2192.58	2.56
	0.8	**Redes Neurais**	1.81	570.63	5.13	1618.08	2.26
	0.8	**Prophet**	1.34	421.11	2.88	908.20	1.69
2100	**0.9**	**Holt-Winters**	2.74	414.22	12.02	1819.90	3.39
	0.9	**Redes Neurais**	2.56	387.71	10.25	1551.59	3.19
	0.9	**Prophet**	1.84	277.91	5.50	831.97	2.31

De acordo com a Tabela 3.3, ao considerar $n = 60$, independentemente do valor de *sar*, observa-se que o *Prophet* exibe consistentemente um desempenho inferior em comparação com os modelos *Holt-Winters* e Redes Neurais. Tomando como exemplo *sar = 0.7*, notamos que o *Prophet* apresenta valores mais elevados em todas as métricas avaliadas, incluindo EMA, PEMA, EQM, PEQM e REQM. Essa tendência persiste para *sar = 0.8* e *sar = 0.9*, indicando que, para amostras menores, o *Prophet* pode não ser tão eficaz quanto os outros dois modelos.

Aumentando o tamanho da amostra para $n = 120$, observamos uma piora geral no desempenho do *Prophet*. Ao analisar $n = 120$, notamos que o *Prophet* continua a exibir valores mais altos em comparação com os outros dois modelos, incluindo um aumento significativo no EMA. Destacamos, por exemplo, que para $n = 120$ e *sar = 0.7*, os modelos de *Holt-Winters* e Redes Neurais obtiveram menores valores nas métricas comparadas.

Por outro lado, ao considerar $n = 300$, observa-se que o *Prophet* apresenta uma melhoria notável, com uma diminuição no EMA em comparação com configurações anteriores. No entanto, mesmo nessas configurações, o *Prophet* ainda não alcançou a paridade com os outros dois modelos em termos de precisão preditiva.

Já de acordo com a Tabela 3.4, ao realizar simulações com maiores tamanhos amostrais e com o acréscimo de pontos discrepantes, o *Prophet* destacou-se como uma escolha eficaz para as previsões temporais nos cenários analisados. Para $n = 1200$, o método apresentou o menor *EMA* e *EQM* quando comparado ao *Holt-Winters* e às Redes Neurais, independentemente do valor utilizado para o componente autoregressivo (*sar*). Essa diferença torna-se ainda mais evidente ao aumentarmos o tamanho amostral para $n = 2100$, em que observamos uma performance superior em relação aos outros dois modelos, com o *Prophet* obtendo os menores valores de *EMA* e *EQM* para todos os cenários.

Assim como no caso das simulações sem adição de pontos discrepantes, é importante ressaltar que o tempo de processamento no R para as previsões das simulações de tamanho n = 1200 e n = 2100 foram superiores a seis horas. Para amostras menores, esse tempo variou de dez minutos até uma hora completa.

3.3 Conclusão das Simulações

A condução de simulações desempenhou um papel crucial na avaliação e comparação de modelos de previsão em séries temporais. Ao gerar séries temporais simuladas com características conhecidas e específicas, foi possível criar um ambiente controlado para testar o desempenho de diferentes modelos. Neste contexto, optamos por simular séries temporais que exibem uma sazonalidade fixa (S=12) e um componente autorregressivo, que pode ser $sar = \{0.7, 0.8, 0.9\}$, com o intuito de avaliar o desempenho dos modelos *Holt-Winters*, Redes Neurais e *Prophet*.

Ao analisar os resultados, observamos padrões distintos em diferentes cenários. Em amostras menores ($n = 60$), o *Prophet* mostrou um desempenho inferior em comparação com os modelos *Holt-Winters* e Redes Neurais, indicando possíveis limitações do modelo em séries temporais mais curtas. A introdução de pontos discrepantes nas simulações revelou uma sensibilidade do *Prophet* a eventos inesperados, especialmente em cenários de menor volume de dados.

À medida que o tamanho da amostra aumentou, especialmente em configurações como $n = 1200$ e $n = 2100$, o *Prophet* apresentou melhorias notáveis, superando os outros modelos em termos de precisão preditiva. Esses resultados sugerem que o *Prophet* pode destacar-se em situações mais complexas, como aquelas com grandes conjuntos de dados e a presença de pontos discrepantes.

Em resumo, as simulações proporcionaram uma visão abrangente do desempenho dos modelos em diferentes contextos, fornecendo contribuições valiosas para a escolha adequada de modelos de previsão em diversas situações.

Capítulo 4

Aplicação em Dados Reais: Criptomoedas Bitcoin e Ethereum

A transição da teoria para a prática é essencial na validação e aplicação efetiva de modelos de previsão em séries temporais. Nesta seção, exploraremos a aplicação do modelo *Prophet* a conjuntos de dados reais, utilizando informações das criptomoedas Bitcoin (BTC) e Ethereum (ETH). A escolha de aplicar o modelo a dados do mundo real é motivada pela necessidade de avaliar a robustez e a adaptabilidade do *Prophet* em ambientes dinâmicos e complexos, como os mercados de criptomoedas.

Dados reais apresentam desafios únicos, incluindo volatilidade significativa, eventos imprevisíveis e influências externas que podem afetar drasticamente as séries temporais. Ao aplicar o *Prophet* a esses conjuntos de dados, buscamos entender como o modelo responde a flutuações de mercado, tendências de longo prazo e eventos extraordinários. Além disso, a aplicação em criptomoedas oferece uma oportunidade interessante de testar a capacidade do *Prophet* em lidar com ativos financeiros altamente dinâmicos.

Ao explorar dados reais de BTC e ETH, diretamente do site Yahoo Finance (2023), esperamos extrair informações sobre o desempenho do *Prophet* em cenários financeiros reais, contribuindo para uma compreensão mais abrangente de suas capacidades e limitações. Essa aplicação prática é essencial para validar a utilidade do modelo em situações do mundo real, onde a precisão preditiva é crucial para a tomada de decisões informadas.

4.1 Aplicação - Bitcoin

Extraído do site Yahoo Finance (2023), os dados referentes a criptomoeda Bitcoin são disponibilizados em diferentes tempos, sendo esses diários, mensais ou anuais. Para o caso dessa aplicação, foi utilizado os retornos diários do preço da criptomoeda (em USD) durante o período de 21/09/2023 até 21/12/2023.

A figura 4.1 abaixo mostra o comportamento do preço diário do Bitcoin durante o período citado anteriormente.

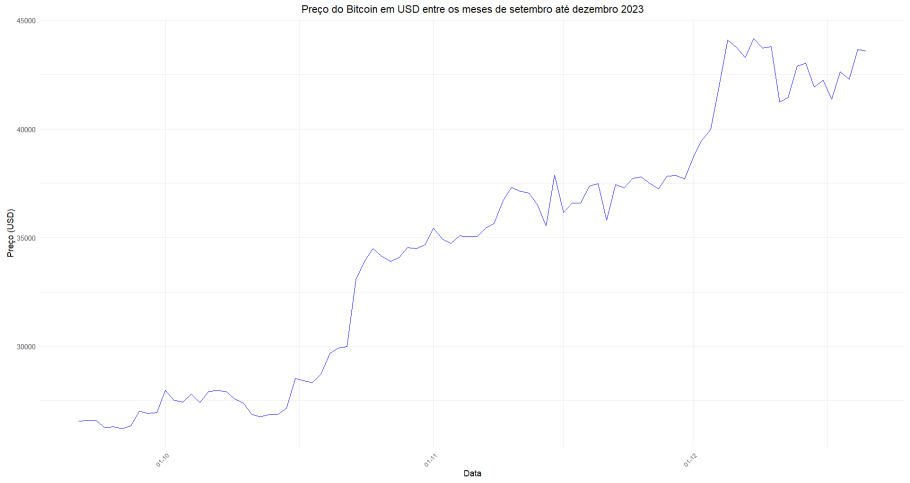

Figura 4.1: Preço do Bitcoin (em USD) durante o período de 21/09/2023 até 21/12/2023

Como já era de se esperar, por se tratar de uma série financeira e, especificamente, de uma criptomoeda, temos uma grande variação no preço, que é causada por fatores tais como a oferta e demanda e até mesmo a perspectiva dos investidores, seja ela otimista ou negativa.

Para o ajuste dos modelos, os dados foram divididos em 90% para treinamento e 10% para teste. A Figura 4.2 nos mostra a previsão do preço do Bitcoin durante o período de 23/11/2023 até 21/12/2023 pelo método de *Holt-Winters*.

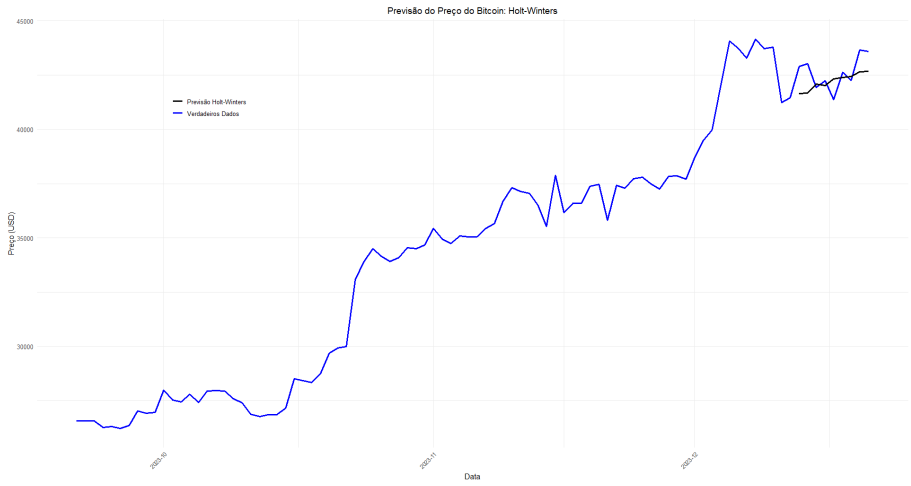

Figura 4.2: Previsão do preço do Bitcoin utilizando o método de *Holt-Winters*.

Conforme podemos observar na Figura 4.2, temos que o método foi capaz de capturar a tendência positiva do preço da criptomoeda durante o período e também de produzir uma boa previsão.

Da mesma maneira, foi ajustado o modelo por meio das Redes-Neurais.

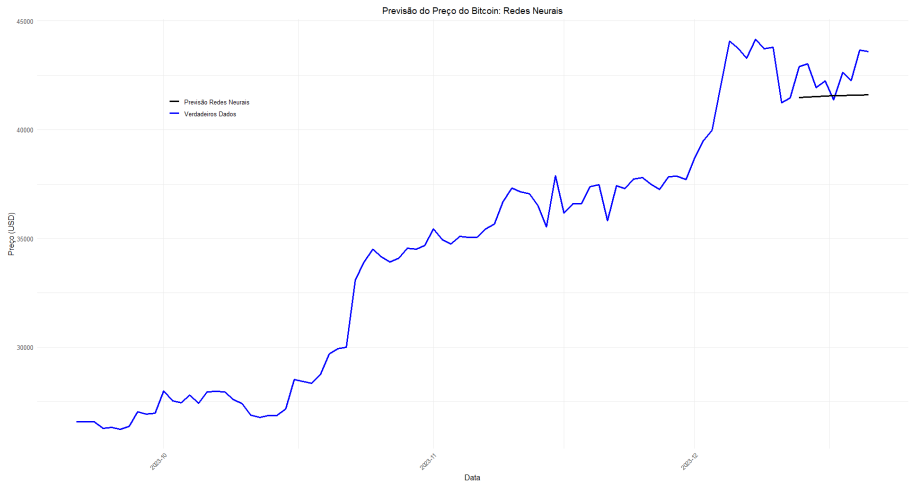

Figura 4.3: Previsão do preço do Bitcoin utilizando o método das Redes Neurais.

De acordo com a Figura 4.3, podemos perceber que o método de Redes Neurais ponderam maiores valores para as observações finais do conjunto de treinamento, isto é, o método não considerou a tendência de crescimento proveniente das observações iniciais. Assim, podemos perceber que a previsão ficou em torno de uma média dos valores finais do conjunto de treino.

Ademais, foi ajustado o Facebook *Prophet*.

Na Figura 4.4 pode-se perceber que o Facebook *Prophet* captou a tendência de crescimento do preço do Bitcoin, de forma semelhante ao método de Holt-Winters.

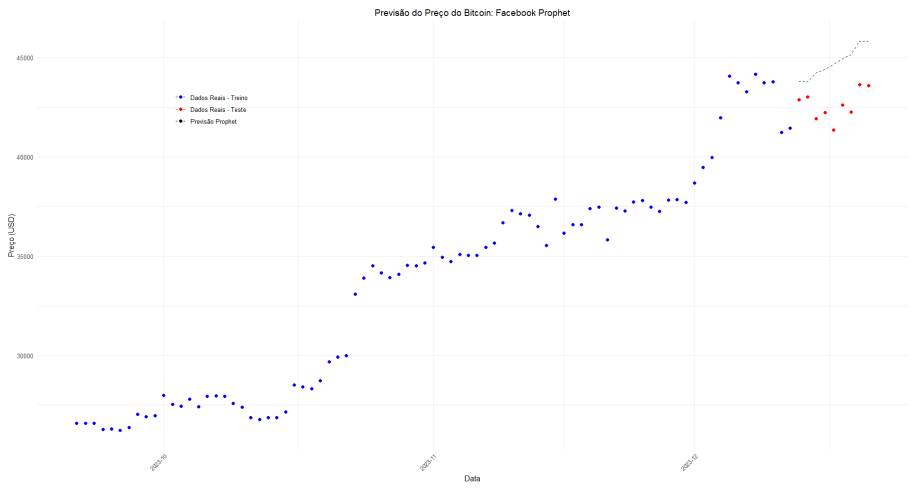

Figura 4.4: Previsão do preço do Bitcoin utilizando o método Facebook *Prophet*.

Por fim, a Tabela 4.1 mostra as medidas de acurácia para cada um dos métodos aplicados nesse conjunto de dados:

	EMA	**PEMA**	**EQM**	**REQM**	**PEQM**
Holt-Winters	696.67	1.63456	698596	1639.068	835.82
Redes Neurais	1113.65	2.612891	1642048	3852.62	1281.42
Facebook *Prophet*	2116.42	4.96562	5079010	11916.53	2253.66

Tabela 4.1: Medidas de Acurácia - Bitcoin

Conforme podemos observar pela Tabela 4.1, temos que o método de *Holt-Winters* obteve Erro Médio Absoluto e Erro Quadrático Médio para o conjunto de dados do Bitcoin, corroborando com a conclusão feita nos gráficos de previsão para cada modelo.

4.2 Aplicação - Ethereum

Com a mesma ideia já realizada no conjunto de dados do Bitcoin, foi realizada a análise para a criptomoeda Ethereum. Nesse caso, utilizamos os retornos diários do preço (em USD) dos últimos 5 anos, isto é, do dia 30/12/2018 até 30/12/2023.

A Figura 4.5 refere-se ao preço da criptomoeda em dólares americanos durante o período de 30/12/2018 até 30/12/2023.

No contexto da criptomoeda Ethereum (ETH), assim como observado no Bitcoin, evidenciamos uma notável volatilidade nos preços diários deste ativo. Essa instabilidade pode ser discernida ao longo de três distintos períodos temporais. Inicialmente, de 2019 a 2021, registramos um preço em torno de 100 USD, indicando uma fase de relativa estabilidade. No intervalo entre 2021 e 2022, observamos uma significativa ascensão nos valores,

Figura 4.5: Preço do Ethereum (em USD) durante o período de 30/12/2018 até 30/12/2023.

atingindo patamares elevados e alcançando quase 5.000 USD. Por fim, no período compreendido entre 2022 e a presente data, verificamos uma flutuação nos preços, oscilando entre 1.000 e 2.000 USD. Esses períodos distintos refletem as variações significativas no comportamento dos preços da Ethereum ao longo do tempo.

Para esse conjunto de dados, também foi utilizada a divisão entre 70% para treino e 30% para teste. A Figura 4.6 refere-se a previsão do preço do Ethereum atinginda pelo método de *Holt-Winters*.

Figura 4.6: Previsão do preço do Ethereum utilizando o método de *Holt-Winters*

Conforme ilustrado na Figura 4.6, observa-se que o método de *Holt-Winters* atribuiu considerável peso à queda do preço do ativo nas últimas observações, resultando em uma previsão com uma tendência negativa.

Em seguida, foi feita a previsão utilizando as Redes Neurais. A Figura 4.7 refere-se ao preço do ativo e a previsão atiginda pelas Redes Neurais.

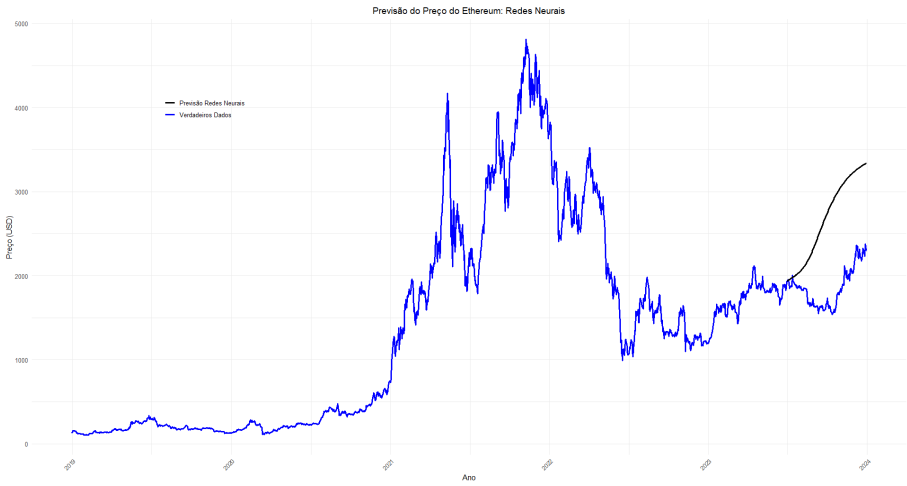

Figura 4.7: Previsão do preço do Ethereum utilizando o método de Redes Neurais

Conforme ilustrado na Figura 4.7, é possível observar que as previsões geradas pelas Redes Neurais foram capazes de identificar o início do crescimento no preço da criptomoeda. Entretanto, é importante ressaltar que durante o período mencionado, as previsões apresentaram valores substancialmente superiores aos dados reais.

Já para o *Prophet*, a Figura 4.8 mostra o comportamento da previsão realizada pelo método no mesmo período.

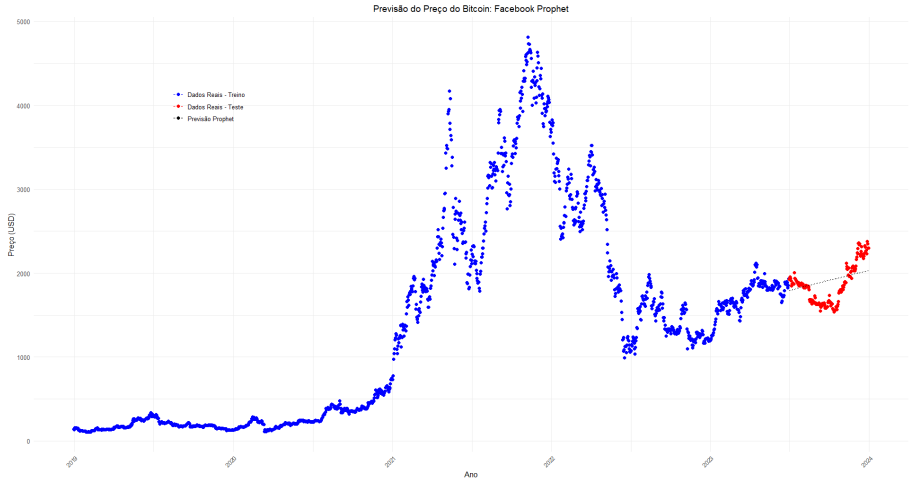

Figura 4.8: Previsão do preço do Ethereum utilizando o *Prophet*

O *Prophet*, por sua vez, conseguiu captar uma tendência positiva do preço da criptomoeda, embora não tenha previsto a queda inicial.

Por fim, a Tabela 4.2 refere-se as medidas de acurácia para os modelos utilizados na previsão do preço da criptomoeda Ethereum.

	EMA	**PEMA**	**EQM**	**REQM**	**PEQM**
Holt-Winters	203.99	10.95	63021.26	3384.33	251.04
Redes Neurais	808.30	43.41	836892.9	44942.33	914.82
Facebook *Prophet*	174.45	9.37	43060.88	2312.43	207.51

Tabela 4.2: Medidas de Acurácia - Ethereum

Como evidenciado na Tabela 4.2, é possível constatar que o *Prophet* apresentou os menores valores para EMA e EQM em comparação com as Redes Neurais e o *Holt-Winters*.

Capítulo 5

Conclusão Final

Em resumo, este trabalho teve como foco a previsão em séries temporais, promovendo uma análise comparativa entre métodos consagrados, como o *Holt-Winters*, a capacidade de identificação de padrões complexos proporcionada pelas Redes Neurais e a abordagem inovadora do Facebook *Prophet*.

O método de *Holt-Winters*, conforme discutido por Morettin (2006a), apresenta vantagens notáveis, tais como simplicidade, facilidade de compreensão das fórmulas, versatilidade proporcionada pela variação dos parâmetros e a habilidade de gerar estimativas precisas a curto prazo. Contudo, é importante destacar que esse método pode ser moroso para se adaptar a novas tendências, além de apresentar desafios na determinação dos valores ideais para seus parâmetros.

As Redes Neurais, por sua vez, demonstram uma notável capacidade de aprendizado de padrões complexos, permitindo que os modelos capturem relações não lineares presentes nos dados. A habilidade de lidar com conjuntos de dados de alta dimensionalidade também se destaca como uma vantagem do método, possibilitando o processamento eficiente de múltiplas séries temporais relacionadas. No entanto, é crucial considerar a exigência de grandes conjuntos de dados e a propensão ao overfitting, que resulta em uma generalização menos eficaz para novos dados.

O método do Facebook *Prophet*, desenvolvido em 2017 por Sean J. Taylor e Ben Letham, representa uma biblioteca de código aberto disponível nas principais linguagens de programação. Seu propósito é abordar duas questões inerentes aos métodos tradicionais de previsão de séries temporais: a inflexibilidade e fragilidade das técnicas completamente automáticas de previsão, e a dependência de um analista especializado em ciência de dados para a utilização de ferramentas mais robustas. Essa ferramenta destaca-se em tarefas

de previsão de séries temporais ao simplificar o processo de criação de previsões precisas, oferecendo opções de personalização intuitivas, mesmo para não especialistas.

Para analisar o desempenho do *Prophet* em comparação com *Holt-Winters* e Redes Neurais, realizamos simulações em conjuntos de 1000 amostras, variando o tamanho amostral. Incluímos sazonalidade fixa e componente auto-regressivo variável (*sar = 0.7, 0.8, 0.9*) para dois casos: sem pontos discrepantes e incluindo pontos discrepantes. Esses valores atípicos podem ter um impacto significativo nas capacidades preditivas dos modelos.

No cenário sem pontos discrepantes, observamos que o *Prophet* consistentemente apresentou desempenho inferior em comparação com os modelos de *Holt-Winters* e Redes Neurais em amostras pequenas. Entretanto, para amostras maiores, com tamanhos de n = 1200 e n = 2100, utilizando diversas medidas de acurácia, o método destacou-se como o mais eficaz entre os três modelos estudados. No caso com a adição de pontos discrepantes, chegamos a uma conclusão semelhante, em que, para amostras maiores, o *Prophet* também se destacou como uma escolha eficaz para as previsões temporais nos cenários analisados.

Ao aplicarmos os métodos nos conjuntos de dados reais, como os relacionados ao Bitcoin e Ethereum, nos deparamos com desafios inerentes à alta volatilidade dos preços, eventos imprevisíveis e influências externas que podem impactar significativamente as séries financeiras. No estudo da criptomoeda Bitcoin, o *Prophet* capturou a tendência de crescimento do preço do ativo, de maneira semelhante ao *Holt-Winters*, porém apresentou maior EMA e EQM quando comparado aos outros dois métodos.

No contexto do Ethereum, o *Prophet* resultou em uma previsão com uma tendência positiva com valores mais próximos as observações reais, quando comparado ao método Holt-Winters. Além disso, o Facebook Prophet registrou os menores valores de EMA e EQM, sendo considerado o modelo mais eficaz entre os métodos aplicados.

É crucial destacar que o *Prophet* é um modelo recentemente desenvolvido, continuamente sujeito a estudos evolutivos. Como sugestão, seria relevante explorar o método em séries temporais com características distintas das abordadas neste trabalho, considerando também outras métricas de acurácia.

Referências Bibliográficas

Buterin, V. (2013). Ethereum. https://ethereum.org/en/.

Canaltech (2023). Facebook. https://canaltech.com.br/empresa/facebook/.

Carvalho A., A. (2023). Redes Neurais. https://sites.icmc.usp.br/andre/research/neural/.

Cuong Duong (2023). Facebook Prophet in 2023 and Beyond. https://medium.com/@cuongduong_35162/facebook-prophet-in-2023-and-beyond-c5086151c138.

Holt (1957). Exponential smoothing: Holt's linear method. https://otexts.com/fpp3/holt.html.

Holt, C. C. (1957). Forecasting seasonals and trends by exponentially weighted moving averages. https://doi.org/10.1016/j.ijforecast.2003.09.015.

Hyndman, R. J. (2014). Forecasting: Principles and practice.

Morettin, M. T. (2006a). *Análise de Séries Temporais*. Edgard Blucher.

Morettin, P. A. (2006b). *Econometria Financeira: Um Curso em Séries Temporais Financeiras*. Editora Campus, São Paulo, Brazil.

Morettin P.A e Toloi C.M.C, T. (2006). *Análise de Séries Temporais*. Edgard Blucher.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.

Prophet (2017). Prophet: Forecasting at Scale. https://research.facebook.com/blog/2017/2/prophet-forecasting-at-scale/.

Taylor, S. J. e Letham, B. (2017). Prophet: Forecasting at scale. *Facebook Research*.

Yahoo Finance (2023). Yahoo Finance - Ethereum (ETH) Historical Data. https://finance.yahoo.com/quote/ETH-USD/history?p=ETH-USD.

Apêndice A

Apêndice A

```
#Aplicação Exemplo
library(forecast)
library(prophet)
library(MASS)
library(datasets)

dados <- USAccDeaths

plot(dados)

tamanho <- length(dados)*0.7

train <- window(dados, end = c(1977, 12))
test <- window(dados, start = c(1978, 1))



#Farei a previsão no dados_treino para os três métodos
#Começando por Holt-Winters :

hw_model <- HoltWinters(train)
hw_model

# Fazer previsões usando o modelo Holt-Winters
```

```
forecast_values_hw <- forecast(hw_model, h = length(test))



# Plotar os resultados com legenda
forecast:::plot.forecast(
  forecast_values_hw,
  xlab = "Anos",
  ylab = "Mortes",
  main = "Previsão de mortes nos EUA: Holt-Winters",
  cex = 0.4 # Diminuir o tamanho dos caracteres da legenda
)

# Adicionar legenda
legend("topright", # Posição da legenda no gráfico (canto superior direito)
       legend = c("Verdadeiros Dados", "Previsão Holt-Winters"),
       col = c("black", "blue"), # Cores para cada série
       lty = c(1, 1), # Tipo de linha para cada série
       lwd = c(1, 1), # Largura da linha para cada série
       cex = 0.8, # Tamanho da fonte da legenda
       inset = c(1978,9000)
)



#Agora por Redes Neurais
library(forecast)
library(ggplot2)

# Carregar a base de dados USAccDeaths
data("USAccDeaths")

# Transformar em série temporal
deaths <- ts(USAccDeaths, start = c(1973, 1), frequency = 12)
```

```
# Ajustar o modelo NNAR
nnar_fit <- nnetar(deaths)

# Fazer previsões com o modelo ajustado
forecast_result_nn <- forecast(nnar_fit, h = 12)

# Converter as previsões em um data frame
forecast_df_nn <- data.frame(
  Date = time(deaths),
  OriginalDeaths = deaths,
  ForecastedDeaths = c(rep(NA, length(deaths) -

  length(forecast_result_nn$mean)), forecast_result_nn$mean)
)

# Plotar a série original até 1978 em preto e as previsões de
1978 a 1979 em vermelho
ggplot(data = forecast_df_nn, aes(x = Date)) +
  geom_line(aes(y = OriginalDeaths), color = "black") +
  geom_line(aes(y = ForecastedDeaths), color = "red") +
  labs(title = "Previsão de Mortes nos EUA - Redes Neurais", x = "Ano",
  y = "Mortes") +
  theme_minimal()



#Facebook Prophet
# Carregar os dados da série temporal "USAccDeaths"
data <- as.numeric(USAccDeaths)

# Criar um dataframe com a coluna "y" (valor)
e criar a coluna "ds" (data) como uma sequência mensal
```

```
df <- data.frame(
  ds = seq(as.Date("1973-01-01"),
  by = "month", length.out = length(data)),
  y = data
)

# Visualizar os primeiros registros do novo dataframe
head(df)

# Filtrar o dataframe apenas até o ano de 1977
df_until_1977 <- subset(df, ds <= as.Date("1977-12-31"))

# Criar o modelo Prophet
model <- prophet(df_until_1977)

# Realizar a previsão para um ano depois
future <- make_future_dataframe(model, periods = 365)
forecast <- predict(model, future)

# Converter a coluna de data "ds" do objeto forecast para o formato Date
forecast$ds <- as.Date(forecast$ds)

# Criar o gráfico usando ggplot2
ggplot() +
  geom_point(data = df_until_1977, aes(x = ds, y = y, color = "Dados Reais"),

  shape = 16, size = 2) +
  geom_line(data = forecast, aes(x = ds, y = yhat, color = "Previsão Prophet")) +
  labs(x = "Anos", y = "Mortes") +
  ggtitle("Previsão de mortes nos EUA : Facebook Prophet") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

```
# Medidas de Acurácia

# Calcular métricas de acurácia Holt Winters
erro_medio_absoluto_hw <- mean(abs(test - forecast_values_hw$mean))
porcentagem_erro_medio_absoluto_hw <-
(erro_medio_absoluto_hw / mean(test)) * 100

erro_quadratico_medio_hw <- mean((test - forecast_values_hw$mean)^2)
porcentagem_erro_quadratico_medio_hw <-
(erro_quadratico_medio_hw / mean(test)) * 100

raiz_erro_quadratico_medio_hw <- sqrt(erro_quadratico_medio_hw)

# Redes Neurais
# Extrair apenas as previsões do objeto forecast_result_nn
forecasted_values_nn <- as.vector(forecast_result_nn$mean)

# Calcular métricas de acurácia
erro_medio_absoluto_nn <- mean(abs(forecasted_values_nn - test))
porcentagem_erro_medio_absoluto_nn <- (erro_medio_absoluto_nn
/ mean(test))* 100

erro_quadratico_medio_nn <- mean((forecasted_values_nn - test)^2)
porcentagem_erro_quadratico_medio_nn <- (erro_quadratico_medio_nn
/ mean(test)) * 100

raiz_erro_quadratico_medio_nn <- sqrt(erro_quadratico_medio_nn)

# Facebook Prophet
# Calcular métricas de acurácia
erro_medio_absoluto_prophet <- mean(abs(test - forecast$yhat[1:length(test)]))
porcentagem_erro_medio_absoluto_prophet <-
```

```
(erro_medio_absoluto_prophet / mean(test)) * 100

erro_quadratico_medio_prophet <- mean((test - forecast$yhat[1:length(test)])^2)
porcentagem_erro_quadratico_medio_prophet
<- (erro_quadratico_medio_prophet / mean(test)) * 100

raiz_erro_quadratico_medio_prophet <- sqrt(erro_quadratico_medio_prophet)




###Aplicação Ethereum

library(ggplot2)
library(forecast)

# Ajustes nos dados
dados <- ethdiario5y
dados$`Adj.Close` <- as.numeric(dados$`Adj.Close`)
dados <- na.omit(dados)

# Criar um dataframe com data e preço
eth_df <- data.frame(Date = as.Date(dados$Date), Price = dados$`Adj.Close`)

# Criar o gráfico
ggplot(eth_df, aes(x = Date, y = Price)) +
  geom_line(color = "blue") +
  labs(title = "Preço do Eth em USD",
       x = "Ano",
       y = "Preço (USD)") +
  theme_minimal() +
  scale_x_date(date_labels = "%Y", date_breaks = "1 year") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 8),
```

```
        plot.title = element_text(hjust = 0.5))

# Dividir os dados
indice_divisao <- round(nrow(eth_df) * 0.7)
train <- eth_df[1:indice_divisao, ]
test <- eth_df[(indice_divisao + 1):nrow(eth_df), ]

plot(train)

# Ajustando o modelo Holt-Winters com sazonalidade mensal
train_ts <- ts(train$Price, frequency = 7)
hw_model <- HoltWinters(train_ts)



forecast_values_hw <- forecast(hw_model, h = nrow(test))

# Converter as previsões para um dataframe
forecast_df <- as.data.frame(forecast_values_hw)

# Adicionar as datas correspondentes às previsões
forecast_df$Date <- test$Date

# Plotar os resultados com ggplot2
ggplot() +
  geom_line(data = eth_df, aes(x = Date, y = Price, color = "Verdadeiros Dados"),
  size = 1) +
  geom_line(data = forecast_df,
  aes(x = Date, y = `Point Forecast`, color = "Previsão Holt-Winters"),
  size = 1) +

  labs(title = "Previsão do Preço do Ethereum: Holt-Winters",
       y = "Preço (USD)", x = "Ano") +
  scale_x_date(date_labels = "%Y", date_breaks = "1 year") +
```

```
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 8),
        plot.title = element_text(hjust = 0.5)) +
  scale_color_manual(values = c("black", "blue"),
                     name = NULL) +
  theme(legend.position = c(0.2, 0.8))


# Medidas de Acurácia para Holt-Winters
# Calcular métricas de acurácia Holt Winters
erro_medio_absoluto_hw <- mean(abs(test$Price - forecast_values_hw$mean))
porcentagem_erro_medio_absoluto_hw <-
(erro_medio_absoluto_hw / mean(test$Price)) * 100

erro_quadratico_medio_hw <- mean((test$Price - forecast_values_hw$mean)^2)
porcentagem_erro_quadratico_medio_hw <- (erro_quadratico_medio_hw
/ mean(test$Price)) * 100

raiz_erro_quadratico_medio_hw <- sqrt(erro_quadratico_medio_hw)









#######


# Ajustar o modelo nnetar

# Ajustar o modelo nnetar
nnetar_model <- nnetar(train$Price, repeats = 10)
```

```
# Fazer previsões usando o modelo nnetar
forecast_values_nnetar <- forecast(nnetar_model, h = nrow(test))

# Converter as previsões para um dataframe
forecast_df_nnetar <- as.data.frame(forecast_values_nnetar)

# Adicionar as datas correspondentes às previsões
forecast_df_nnetar$Date <- test$Date

# Plotar os resultados com ggplot2
# Plotar os resultados com ggplot2
gg_eth_plot <- ggplot() +
  geom_line(data = eth_df, aes(x = as.Date(Date), y = Price,
  color = "Verdadeiros Dados"),
  size = 1) +
  geom_line(data = forecast_df_nnetar, aes(x = as.Date(Date),
  y = `Point Forecast`, color = "Previsão Redes Neurais"),
  size = 1) +
  labs(title = "Previsão do Preço do Ethereum: Redes Neurais",

       y = "Preço (USD)", x = "Ano") +
  scale_x_date(date_labels = "%Y", date_breaks = "1 year") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 8),
        plot.title = element_text(hjust = 0.5)) +
  scale_color_manual(values = c("black", "blue", "red"),
                     name = NULL) +
  theme(legend.position = c(0.2, 0.8))  # Ajusta a posição da legenda

gg_eth_plot
```

```
# Medidas de Acurácia para Redes Neurais
# Extrair apenas as previsões do objeto forecast_values_nnetar
forecasted_values_nnetar <- as.vector(forecast_values_nnetar$mean)

# Calcular métricas de acurácia
erro_medio_absoluto_nnetar <- mean(abs(forecasted_values_nnetar - test$Price))
porcentagem_erro_medio_absoluto_nnetar
<- (erro_medio_absoluto_nnetar
/ mean(test$Price)) * 100

erro_quadratico_medio_nnetar <- mean((forecasted_values_nnetar - test$Price)^2)
porcentagem_erro_quadratico_medio_nnetar
<- (erro_quadratico_medio_nnetar / mean(test$Price)) * 100

raiz_erro_quadratico_medio_nnetar <- sqrt(erro_quadratico_medio_nnetar)

###############




library(prophet)

# Criar um dataframe com as colunas 'ds' (data) e 'y' (preço)
eth_df <- data.frame(ds = as.Date(dados$Date), y = as.numeric(dados$`Adj.Close`))

# Dividir os dados
indice_divisao <- round(nrow(eth_df) * 0.7)
train <- eth_df[1:indice_divisao, ]
test <- eth_df[(indice_divisao + 1):nrow(eth_df), ]
```

```
# Certifique-se de que a coluna ds é do tipo Date
train$ds <- as.Date(train$ds)
test$ds <- as.Date(test$ds)

# Certifique-se de que a coluna y é numérica
train$y <- as.numeric(train$y)
test$y <- as.numeric(test$y)

# Ajustar o modelo prophet com os dados de treinamento
prophet_model_fit <- prophet(yearly.seasonality = FALSE,
weekly.seasonality = TRUE, daily.seasonality = FALSE)



train_data <- data.frame(ds = train$ds, y = train$y)
prophet_model_fit <- fit.prophet(prophet_model_fit, train_data)

# Criar um dataframe com as datas do período de teste
future <- make_future_dataframe(prophet_model_fit, periods = nrow(test)
, include_history = FALSE)
forecast <- predict(prophet_model_fit, future)
forecast$ds <- as.Date(forecast$ds)

# Reexecute o código do ggplot
ggplot() +
  geom_point(data = train, aes(x = ds, y = y,
  color = "Dados Reais - Treino"), shape = 16, size = 2) +

  geom_point(data = test, aes(x = ds, y = y,
  color = "Dados Reais - Teste"), shape = 16, size = 2) +
  geom_line(data = forecast, aes(x = ds, y = yhat,
  color = "Previsão Prophet"), linetype = "dashed") +
```

```
  labs(title = "Previsão do Preço do Bitcoin:
  Facebook Prophet",
       y = "Preço (USD)", x = "Ano") +
  scale_x_date(date_labels = "%Y", date_breaks = "1 year")
  +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1,
  size = 8),
        plot.title = element_text(hjust = 0.5)) +
  scale_color_manual(values = c("Dados Reais - Treino" = "blue",
  "Dados Reais - Teste" = "red", "Previsão Prophet" = "black"),
                     name = NULL) +
  theme(legend.position = c(0.2, 0.8))


# Medidas de Acurácia para Prophet
# Calcular métricas de acurácia para Prophet
erro_medio_absoluto_prophet <- mean(abs(test$y - forecast$yhat))
porcentagem_erro_medio_absoluto_prophet <- (erro_medio_absoluto_prophet
/ mean(test$y)) * 100

erro_quadratico_medio_prophet <- mean((test$y - forecast$yhat)^2)
porcentagem_erro_quadratico_medio_prophet <- (erro_quadratico_medio_prophet
/ mean(test$y)) * 100

raiz_erro_quadratico_medio_prophet <- sqrt(erro_quadratico_medio_prophet)




##Aplicação Bitcoin
library(ggplot2)
library(forecast)
```

```
# Ajustes nos dados
dados <- btcdiario120
dados$`Adj.Close` <- as.numeric(dados$`Adj.Close`)
dados <- na.omit(dados)

# Criar um dataframe com data e preço
bitcoin_df <- data.frame(Date = as.Date(dados$Date), Price = dados$`Adj.Close`)

# Criar o gráfico
ggplot(bitcoin_df, aes(x = Date, y = Price)) +
  geom_line(color = "blue") +
  labs(title = "Preço do Bitcoin em USD entre
  os meses de setembro até dezembro 2023",
       x = "Data",
       y = "Preço (USD)") +
  theme_minimal() +
  scale_x_date(date_labels = "%d-%m", date_breaks = "1 month") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 8),
        plot.title = element_text(hjust = 0.5))


# Dividir os dados
indice_divisao <- round(nrow(bitcoin_df) * 0.7)
train <- bitcoin_df[1:indice_divisao, ]
test <- bitcoin_df[(indice_divisao + 1):nrow(bitcoin_df), ]

# Ajustando o modelo Holt-Winters com sazonalidade mensal
train_ts <- ts(train$Price, frequency = 7)
hw_model <- HoltWinters(train_ts)

# Fazer previsões usando o modelo Holt-Winters
para o mesmo comprimento que o conjunto de teste
```

```
forecast_values_hw <- forecast(hw_model, h = nrow(test))

# Converter as previsões para um dataframe
forecast_df <- as.data.frame(forecast_values_hw)

# Adicionar as datas correspondentes às previsões
forecast_df$Date <- test$Date

# Plotar os resultados com ggplot2
# Plotar os resultados com ggplot2
# Plotar os resultados com ggplot2
ggplot() +
  geom_line(data = bitcoin_df, aes(x = Date, y = Price,
  color = "Verdadeiros Dados"), size = 1) +
  geom_line(data = forecast_df, aes(x = Date, y = `Point Forecast`,
  color = "Previsão Holt-Winters"), size = 1) +
  labs(title = "Previsão do Preço do Bitcoin: Holt-Winters",
       y = "Preço (USD)", x = "Data") +
  scale_x_date(date_labels = "%Y-%m", date_breaks = "1 month") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1,
  size = 8),
        plot.title = element_text(hjust = 0.5)) +
  scale_color_manual(values = c("black", "blue"),
                     name = NULL) +
  theme(legend.position = c(0.2, 0.8))


# Medidas de Acurácia para Holt-Winters
# Calcular métricas de acurácia Holt Winters
erro_medio_absoluto_hw <- mean(abs(test$Price -
forecast_values_hw$mean))
porcentagem_erro_medio_absoluto_hw <- (erro_medio_absoluto_hw
```

```
/ mean(test$Price)) * 100

erro_quadratico_medio_hw <- mean((test$Price -
forecast_values_hw$mean)^2)
porcentagem_erro_quadratico_medio_hw <- (erro_quadratico_medio_hw
/ mean(test$Price)) * 100

raiz_erro_quadratico_medio_hw <- sqrt(erro_quadratico_medio_hw)




#######


# Ajustar o modelo nnetar

# Ajustar o modelo nnetar
nnetar_model <- nnetar(train$Price, repeats = 10)

# Fazer previsões usando o modelo nnetar
forecast_values_nnetar <- forecast(nnetar_model, h = nrow(test))

# Converter as previsões para um dataframe
forecast_df_nnetar <- as.data.frame(forecast_values_nnetar)

# Adicionar as datas correspondentes às previsões
forecast_df_nnetar$Date <- test$Date

# Plotar os resultados com ggplot2
ggplot() +
  geom_line(data = bitcoin_df, aes(x = Date, y = Price,
  color = "Verdadeiros Dados"), size = 1) +
```

```
  geom_line(data = forecast_df_nnetar, aes(x = Date,
  y = ‘Point Forecast‘, color = "Previsão Redes Neurais"),
  size = 1) +
  labs(title = "Previsão do Preço do Bitcoin: Redes Neurais",
       y = "Preço (USD)", x = "Data") +
  scale_x_date(date_labels = "%Y-%m",
  date_breaks = "1 month") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1,
  size = 8),
        plot.title = element_text(hjust = 0.5)) +
  scale_color_manual(values = c("black", "blue"),
                     name = NULL) +
  theme(legend.position = c(0.2, 0.8))




# Medidas de Acurácia para Redes Neurais
# Extrair apenas as previsões do objeto forecast_values_nnetar
forecasted_values_nnetar <- as.vector(forecast_values_nnetar$mean)

# Calcular métricas de acurácia
erro_medio_absoluto_nnetar <- mean(abs(forecasted_values_nnetar
- test$Price))
porcentagem_erro_medio_absoluto_nnetar <- (erro_medio_absoluto_nnetar
/ mean(test$Price)) * 100

erro_quadratico_medio_nnetar <- mean((forecasted_values_nnetar
- test$Price)^2)
porcentagem_erro_quadratico_medio_nnetar <- (erro_quadratico_medio_nnetar
/ mean(test$Price)) * 100
```

```
raiz_erro_quadratico_medio_nnetar <- sqrt(erro_quadratico_medio_nnetar)

###############




library(prophet)

# Criar um dataframe com as colunas 'ds' (data) e 'y' (preço)
bitcoin_df <- data.frame(ds = as.Date(dados$Date),
y = as.numeric(dados$`Adj.Close`))

# Dividir os dados
indice_divisao <- round(nrow(bitcoin_df) * 0.7)
train <- bitcoin_df[1:indice_divisao, ]
test <- bitcoin_df[(indice_divisao + 1):nrow(bitcoin_df), ]

# Certifique-se de que a coluna ds é do tipo Date
train$ds <- as.Date(train$ds)
test$ds <- as.Date(test$ds)

# Certifique-se de que a coluna y é numérica
train$y <- as.numeric(train$y)
test$y <- as.numeric(test$y)
# Ajustar o modelo prophet com os dados de treinamento
prophet_model_fit <- prophet(yearly.seasonality = FALSE,
weekly.seasonality = TRUE, daily.seasonality = FALSE)
train_data <- data.frame(ds = train$ds, y = train$y)
prophet_model_fit <- fit.prophet(prophet_model_fit, train_data)

# Criar um dataframe com as datas do período de teste
future <- make_future_dataframe(prophet_model_fit,
```

```
periods = nrow(test), include_history = FALSE)
forecast <- predict(prophet_model_fit, future)
forecast$ds <- as.Date(forecast$ds)

# Agora, reexecute o código do ggplot
ggplot() +
  geom_point(data = train, aes(x = ds, y = y,
  color = "Dados Reais - Treino"), shape = 16, size = 2) +
  geom_point(data = test, aes(x = ds, y = y,
  color = "Dados Reais - Teste"), shape = 16, size = 2) +
  geom_line(data = forecast, aes(x = ds, y =
  yhat, color = "Previsão Prophet"), linetype = "dashed") +
  labs(title = "Previsão do Preço do Bitcoin: Facebook Prophet",
       y = "Preço (USD)", x = "Data") +
  scale_x_date(date_labels = "%Y-%m",
  date_breaks = "1 month")
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45,
  hjust = 1, size = 8),
        plot.title = element_text(hjust = 0.5)) +
  scale_color_manual(values = c("Dados Reais - Treino" = "blue",
  "Dados Reais - Teste" = "red", "Previsão Prophet" = "black"),
                     name = NULL) +
  theme(legend.position = c(0.2, 0.8))



# Medidas de Acurácia para Prophet
# Calcular métricas de acurácia para Prophet
erro_medio_absoluto_prophet <- mean(abs(test$y - forecast$yhat))
porcentagem_erro_medio_absoluto_prophet
<- (erro_medio_absoluto_prophet
/ mean(test$y)) * 100
```

```
erro_quadratico_medio_prophet <- mean((test$y - forecast$yhat)^2)
porcentagem_erro_quadratico_medio_prophet
<- (erro_quadratico_medio_prophet
/ mean(test$y)) * 100

raiz_erro_quadratico_medio_prophet <- sqrt(erro_quadratico_medio_prophet)



##Simulações

library(forecast)
library(prophet)
library(MASS)
library(datasets)
library(astsa)

# Definir o número de simulações
num_simulacoes <- 1000

# Inicializar uma lista para armazenar as previsões de cada simulação
lista_previsoes_hw <- list()
lista_previsoes_nn <- list()
lista_previsoes_prophet <- list()

# Porcentagem de dados para treinamento
percent_treinamento <- 0.7  # ajuste conforme necessário

# Loop para realizar as simulações
for (i in 1:num_simulacoes) {
  set.seed(123 + i)  # Alterar a semente para cada simulação
  sAR <- sarima.sim(sar = 0.7, S = 12, n = 300)
  sAR_unrolled <- ts(sAR)
```

```
n <- length(sAR_unrolled)
indice_treinamento <- floor(n * percent_treinamento)

dados_treinamento <- sAR_unrolled[1:indice_treinamento]
dados_treinamento <- ts(dados_treinamento, frequency = 12)

dados_teste <- sAR_unrolled[(indice_treinamento + 1):n]
dados_teste <- ts(dados_teste, frequency = 12)

periodos_previsao <- length(dados_teste)

# Holt-Winters
fit <- HoltWinters(dados_treinamento)
previsoes_hw <- forecast(fit, h = periodos_previsao)
previsoes_hw <- previsoes_hw$mean[1:periodos_previsao]
lista_previsoes_hw[[i]] <- previsoes_hw

# Redes Neurais
nnar_fit <- nnetar(dados_treinamento)
previsoes_teste <- forecast(nnar_fit, h = periodos_previsao)
previsoes_teste <- previsoes_teste$mean[1:periodos_previsao]
lista_previsoes_nn[[i]] <- previsoes_teste

# Prophet
dados_treinamento_prophet <- data.frame(
  ds = seq(from = 1, by = 1, length.out = length(dados_treinamento)),
  y = dados_treinamento
)

dados_treinamento_prophet$ds <- as.Date(dados_treinamento_prophet$ds,
origin = "2023-01-01")
```

```
  modelo_prophet <- prophet(dados_treinamento_prophet, yearly.seasonality = TRUE)
  future <- make_future_dataframe(modelo_prophet, periods = periodos_previsao)
  previsao_prophet <- predict(modelo_prophet, future)

  lista_previsoes_prophet[[i]]
  <- previsao_prophet$yhat[(indice_treinamento + 1):(n + periodos_previsao)]
}


#agora as medidas de acuracia

# Inicializar listas para armazenar as métricas de acurácia para Holt-Winters
lista_metricas_hw <- list()

# Calcular métricas de acurácia para cada simulação de Holt-Winters
for (i in 1:num_simulacoes) {
  erro_medio_absoluto_hw <- mean(abs(dados_teste - lista_previsoes_hw[[i]]))
  porcentagem_erro_medio_absoluto_hw
  <- (erro_medio_absoluto_hw / mean(dados_teste)) * 100
  erro_quadratico_medio_hw <- mean((dados_teste - lista_previsoes_hw[[i]])^2)
  porcentagem_erro_quadratico_medio_hw
  <- (erro_quadratico_medio_hw / mean(dados_teste)) * 100
  raiz_erro_quadratico_medio_hw <- sqrt(erro_quadratico_medio_hw)

  lista_metricas_hw[[i]]
  <- c(erro_medio_absoluto_hw, porcentagem_erro_medio_absoluto_hw,
  erro_quadratico_medio_hw,
  porcentagem_erro_quadratico_medio_hw, raiz_erro_quadratico_medio_hw)
}


# Converter listas em matrizes
matriz_metricas_hw <- do.call(rbind, lista_metricas_hw)
```

```
# Calcular médias das métricas para Holt-Winters
media_metricas_hw <- colMeans(matriz_metricas_hw)


##PARA REDES NEURAIS

lista_metricas_nn <- list()

# Calcular métricas de acurácia para cada simulação de Redes Neurais
for (i in 1:num_simulacoes) {
  erro_medio_absoluto_nn <- mean(abs(dados_teste - lista_previsoes_nn[[i]]))
  porcentagem_erro_medio_absoluto_nn <-
  (erro_medio_absoluto_nn / mean(dados_teste)) * 100
  erro_quadratico_medio_nn <- mean((dados_teste - lista_previsoes_nn[[i]])^2)
  porcentagem_erro_quadratico_medio_nn
  <- (erro_quadratico_medio_nn / mean(dados_teste)) * 100
  raiz_erro_quadratico_medio_nn <- sqrt(erro_quadratico_medio_nn)

  lista_metricas_nn[[i]] <-
  c(erro_medio_absoluto_nn, porcentagem_erro_medio_absoluto_nn,
  erro_quadratico_medio_nn,
  porcentagem_erro_quadratico_medio_nn, raiz_erro_quadratico_medio_nn)
}

# Converter listas em matrizes para Redes Neurais
matriz_metricas_nn <- do.call(rbind, lista_metricas_nn)

# Calcular médias das métricas para Redes Neurais
media_metricas_nn <- colMeans(matriz_metricas_nn)


#PROPHET
# Inicializar listas para armazenar as métricas de acurácia para o Prophet
```

```
lista_metricas_prophet <- list()

# Calcular métricas de acurácia para cada simulação do Prophet
for (i in 1:num_simulacoes) {
  previsoes_prophet_i <-
  lista_previsoes_prophet[[i]][1:length(dados_teste)
  ]  # Ajuste o comprimento das previsões

  erro_medio_absoluto_prophet <- mean(abs(dados_teste - previsoes_prophet_i))
  porcentagem_erro_medio_absoluto_prophet <-
  (erro_medio_absoluto_prophet / mean(dados_teste)) * 100
  erro_quadratico_medio_prophet <- mean((dados_teste - previsoes_prophet_i)^2)
  porcentagem_erro_quadratico_medio_prophet <-
  (erro_quadratico_medio_prophet / mean(dados_teste)) * 100
  raiz_erro_quadratico_medio_prophet <- sqrt(erro_quadratico_medio_prophet)

  lista_metricas_prophet[[i]] <-
  c(erro_medio_absoluto_prophet,
  porcentagem_erro_medio_absoluto_prophet,
  erro_quadratico_medio_prophet,
  porcentagem_erro_quadratico_medio_prophet, raiz_erro_quadratico_medio_prophet)
}

# Converter listas em matrizes para o Prophet
matriz_metricas_prophet <- do.call(rbind, lista_metricas_prophet)

# Calcular médias das métricas para o Prophet
media_metricas_prophet <- colMeans(matriz_metricas_prophet)



#COMPARAÇÃO
```

```
tabela_acuracia <- data.frame(
  Modelo = c("Holt-Winters", "Redes Neurais", "Prophet"),
  EMA = c(media_metricas_hw[1],
  media_metricas_nn[1], media_metricas_prophet[1]),
  PEMA = c(media_metricas_hw[2],
  media_metricas_nn[2], media_metricas_prophet[2]),
  EQM = c(media_metricas_hw[3],
  media_metricas_nn[3], media_metricas_prophet[3]),
  PEQM = c(media_metricas_hw[4],
  media_metricas_nn[4], media_metricas_prophet[4]),
  REQM = c(media_metricas_hw[5],
  media_metricas_nn[5], media_metricas_prophet[5])
)

# Exibir a tabela
print(tabela_acuracia)
library(xtable)
xtable(tabela_acuracia)



##Simulações com pontos discrepantes

library(forecast)
library(prophet)
library(MASS)
library(datasets)
library(astsa)

# Função para inserir pontos discrepantes em locais específicos
inserir_discrepantes_especificos <-
function(serie, proporcao_discrepantes) {
  n <- length(serie)
```

```
  num_discrepantes <- round(proporcao_discrepantes * n)

  # Definir valores para os pontos
  discrepantes (por exemplo, 2 vezes o máximo da série)
  valores_discrepantes <- 2 * max(serie)

  # Inserir pontos discrepantes em locais específicos
  serie_com_discrepantes <- serie
  indice_primeiro_quantil <- round(0.25 * n)
  indice_central <- round(0.5 * n)
  indice_ultimo_quantil <- round(0.75 * n)

  serie_com_discrepantes[c(indice_primeiro_quantil,
  indice_central, indice_ultimo_quantil)] <-
  valores_discrepantes

  return(serie_com_discrepantes)
}

# Definir o número de simulações
num_simulacoes <- 1000

# Inicializar uma lista para armazenar as previsões de cada simulação
lista_previsoes_hw <- list()
lista_previsoes_nn <- list()
lista_previsoes_prophet <- list()

# Porcentagem de dados para treinamento
percent_treinamento <- 0.7  # ajuste conforme necessário

# Loop para realizar as simulações
for (i in 1:num_simulacoes) {
  set.seed(123 + i)  # Alterar a semente para cada simulação
```

```
# Gerar série temporal com pontos discrepantes
sAR <- sarima.sim(sar = 0.9, S = 12, n = 60)
sAR_com_discrepantes <- inserir_discrepantes_especificos(sAR, 0.02)
# 2% dos pontos serão discrepantes

sAR_unrolled <- ts(sAR_com_discrepantes)

n <- length(sAR_unrolled)
indice_treinamento <- floor(n * percent_treinamento)

dados_treinamento <- sAR_unrolled[1:indice_treinamento]
dados_treinamento <- ts(dados_treinamento, frequency = 12)

dados_teste <- sAR_unrolled[(indice_treinamento + 1):n]
dados_teste <- ts(dados_teste, frequency = 12)

periodos_previsao <- length(dados_teste)

# Holt-Winters
fit <- HoltWinters(dados_treinamento)
previsoes_hw <- forecast(fit, h = periodos_previsao)
previsoes_hw <- previsoes_hw$mean[1:periodos_previsao]
lista_previsoes_hw[[i]] <- previsoes_hw

# Redes Neurais
nnar_fit <- nnetar(dados_treinamento)
previsoes_teste <- forecast(nnar_fit, h = periodos_previsao)
previsoes_teste <- previsoes_teste$mean[1:periodos_previsao]
lista_previsoes_nn[[i]] <- previsoes_teste

# Prophet
dados_treinamento_prophet <- data.frame(
```

```
    ds = seq(from = 1, by = 1, length.out = length(dados_treinamento)),
    y = dados_treinamento
  )

  dados_treinamento_prophet$ds <- as.Date(dados_treinamento_prophet$ds,
  origin = "2023-01-01")

  modelo_prophet <- prophet(dados_treinamento_prophet, yearly.seasonality = TRUE)
  future <- make_future_dataframe(modelo_prophet, periods = periodos_previsao)
  previsao_prophet <- predict(modelo_prophet, future)

  lista_previsoes_prophet[[i]] <-
  previsao_prophet$yhat[(indice_treinamento + 1):(n + periodos_previsao)]
}



#agora as medidas de acuracia

# Inicializar listas para armazenar as métricas de acurácia para Holt-Winters
lista_metricas_hw <- list()

# Calcular métricas de acurácia para cada simulação de Holt-Winters
for (i in 1:num_simulacoes) {
  erro_medio_absoluto_hw <- mean(abs(dados_teste - lista_previsoes_hw[[i]]))
  porcentagem_erro_medio_absoluto_hw <-
  (erro_medio_absoluto_hw / mean(dados_teste)) * 100
  erro_quadratico_medio_hw <- mean((dados_teste - lista_previsoes_hw[[i]])^2)
  porcentagem_erro_quadratico_medio_hw <-
  (erro_quadratico_medio_hw / mean(dados_teste)) * 100
  raiz_erro_quadratico_medio_hw <- sqrt(erro_quadratico_medio_hw)

  lista_metricas_hw[[i]] <-
```

```
  c(erro_medio_absoluto_hw,
  porcentagem_erro_medio_absoluto_hw,
  erro_quadratico_medio_hw,
  porcentagem_erro_quadratico_medio_hw,
  raiz_erro_quadratico_medio_hw)
}


# Converter listas em matrizes
matriz_metricas_hw <- do.call(rbind, lista_metricas_hw)
# Calcular médias das métricas para Holt-Winters
media_metricas_hw <- colMeans(matriz_metricas_hw)


##PARA REDES NEURAIS

lista_metricas_nn <- list()

# Calcular métricas de acurácia para cada simulação de Redes Neurais
for (i in 1:num_simulacoes) {
  erro_medio_absoluto_nn <- mean(abs(dados_teste - lista_previsoes_nn[[i]]))
  porcentagem_erro_medio_absoluto_nn <-
  (erro_medio_absoluto_nn / mean(dados_teste)) * 100
  erro_quadratico_medio_nn <- mean((dados_teste - lista_previsoes_nn[[i]])^2)
  porcentagem_erro_quadratico_medio_nn <-
  (erro_quadratico_medio_nn / mean(dados_teste)) * 100
  raiz_erro_quadratico_medio_nn <- sqrt(erro_quadratico_medio_nn)

  lista_metricas_nn[[i]] <-
  c(erro_medio_absoluto_nn,
  porcentagem_erro_medio_absoluto_nn,
  erro_quadratico_medio_nn,
  porcentagem_erro_quadratico_medio_nn,
```

```
  raiz_erro_quadratico_medio_nn)
}

# Converter listas em matrizes para Redes Neurais
matriz_metricas_nn <- do.call(rbind, lista_metricas_nn)

# Calcular médias das métricas para Redes Neurais
media_metricas_nn <- colMeans(matriz_metricas_nn)


#PROPHET
# Inicializar listas para armazenar as métricas de acurácia para o Prophet
lista_metricas_prophet <- list()

# Calcular métricas de acurácia para cada simulação do Prophet
for (i in 1:num_simulacoes) {
  previsoes_prophet_i <-
  lista_previsoes_prophet[[i]][1:length(dados_teste)]
  # Ajuste o comprimento das previsões

  erro_medio_absoluto_prophet <- mean(abs(dados_teste - previsoes_prophet_i))
  porcentagem_erro_medio_absoluto_prophet <-
  (erro_medio_absoluto_prophet / mean(dados_teste)) * 100
  erro_quadratico_medio_prophet <- mean((dados_teste - previsoes_prophet_i)^2)
  porcentagem_erro_quadratico_medio_prophet <-
  (erro_quadratico_medio_prophet / mean(dados_teste)) * 100
  raiz_erro_quadratico_medio_prophet <- sqrt(erro_quadratico_medio_prophet)

  lista_metricas_prophet[[i]] <-
  c(erro_medio_absoluto_prophet,
  porcentagem_erro_medio_absoluto_prophet,
  erro_quadratico_medio_prophet,
  porcentagem_erro_quadratico_medio_prophet, raiz_erro_quadratico_medio_prophet)
```

```
}

# Converter listas em matrizes para o Prophet
matriz_metricas_prophet <- do.call(rbind, lista_metricas_prophet)

# Calcular médias das métricas para o Prophet
media_metricas_prophet <- colMeans(matriz_metricas_prophet)



#COMPARAÇÃO

tabela_acuracia <- data.frame(
  Modelo = c("Holt-Winters", "Redes Neurais", "Prophet"),
  EMA = c(media_metricas_hw[1],
  media_metricas_nn[1], media_metricas_prophet[1]),
  PEMA = c(media_metricas_hw[2],
  media_metricas_nn[2], media_metricas_prophet[2]),
  EQM = c(media_metricas_hw[3],
  media_metricas_nn[3], media_metricas_prophet[3]),
  PEQM = c(media_metricas_hw[4],
  media_metricas_nn[4], media_metricas_prophet[4]),
  REQM = c(media_metricas_hw[5],
  media_metricas_nn[5], media_metricas_prophet[5])
)

# Exibir a tabela
print(tabela_acuracia)
library(xtable)
xtable(tabela_acuracia)
```