UNIVERSIDADE FEDERAL DO PAMPA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

GABRIEL RITTA CORREIA

USO DE ALGORITMOS PARA PREDIÇÃO DE SÉRIES TEMPORAIS APLICADO AO MERCADO DE CRIPTOMOEDAS

Bagé
2022

GABRIEL RITTA CORREIA

USO DE ALGORITMOS PARA PREDIÇÃO DE SÉRIES TEMPORAIS APLICADO AO MERCADO DE CRIPTOMOEDAS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Gerson Alberto Leira Nunes

Bagé
2022

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

C824u Correia, Gabriel Ritta
USO DE ALGORITMOS PARA PREDIÇÃO DE SÉRIES TEMPORAIS APLICADO AO MERCADO DE CRIPTOMOEDAS / Gabriel Ritta Correia.
77 p.

Trabalho de Conclusão de Curso(Graduação)-- Universidade Federal do Pampa, ENGENHARIA DE COMPUTAÇÃO, 2022.
"Orientação: Gerson Alberto Leira Nunes".

1. Séries Temporais. 2. Criptomoedas. 3. Aprendizado de Máquina. 4. Inteligência Artificial. I. Título.

SERVIÇO PÚBLICO FEDERAL

MINISTÉRIO DA EDUCAÇÃO

Universidade Federal do Pampa

GABRIEL RITTA CORREIA

USO DE ALGORITMOS PARA PREDIÇÃO DE SÉRIES TEMPORAIS APLICADO AO MERCADO DE CRIPTOMOEDAS

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 11 de Agosto de 2022.

Banca examinadora:

_____

Prof. Dr. Gerson Alberto Leira Nunes
Orientador
UNIPAMPA

_____

Prof. Dr. Carlos Michel Betemps
UNIPAMPA

_____

Prof. Dr. Leonardo Bidese de Pinho
UNIPAMPA

Assinado eletronicamente por **GERSON ALBERTO LEIRIA NUNES**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/08/2022, às 16:51, conforme horário oficial de Brasília, de acordo com as

normativas legais aplicáveis.

Assinado eletronicamente por **CARLOS MICHEL BETEMPS**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/08/2022, às 16:52, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **LEONARDO BIDESE DE PINHO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/08/2022, às 17:31, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0902060** e o código CRC **7E0C4593**.

Referência: Processo nº 23100.017415/2022-11 SEI nº 0902060

Dedico este trabalho aos meus pais, irmão, avós, tios, primos e namorada, por todo o apoio durante a jornada na faculdade.

AGRADECIMENTO

Agradeço a minha namorada Maria Elizabeth Bárcena Silva e ao meu orientador Gerson Alberto Leira Nunes por todo o apoio, formação e ajuda durante o desenvolvimento do trabalho de conclusão do curso. Além disso, não posso deixar de agradecer meus colegas e amigos Angelo Rodrigues, Marcelo Silva, Thiago Leal, Tiago Porto, Kelvin Clóvis, Eric Dias e Michel Almeida por compartilharem o conhecimento, pelo companheirismo, esforço e dedicação necessária para atingir os objetivos do curso.

O sucesso é ir de fracasso em fracasso sem perder o entusiasmo.

— Winston Churchill

RESUMO

O rápido crescimento da capacidade de processamento de dados decorrente da evolução do poder computacional, está diretamente relacionado com as técnicas de aprendizado de máquina e redes neurais, pois esta evolução permite a criação de sistemas capazes de realizar predições precisas de séries temporais. Este trabalho aborda conceitos de inteligência artificial, aprendizado de máquina, criptomoedas e principalmente modelos de predições de séries temporais, como: *Facebook 's Prophet*, ARIMA e LSTM. Este trabalho tem como objetivo verificar a capacidade desses modelos quanto a predição do valor de criptomoedas. Apresenta-se um referencial teórico que introduz conceitos importantes para o entendimento do assunto, bem como as principais tecnologias que são utilizadas na atualidade e os trabalhos correlatos. São apresentadas as métricas de desempenho adotadas (MAPE e RMSE) para avaliar os modelos e os testes realizados utilizando as ferramentas *Facebook's Prophet*, ARIMA e LSTM, bem como uma analise comparativa dos resultados obtidos. Os modelos desenvolvidos conseguiram acompanhar as tendências de alta e baixa da série temporal. O Facebook's Prophet obteve 10,93%, 16,32% e 40,14% de percentual de erro médio para as previsões de 1 dia, 1 semana e 1 mês respectivamente. Já o modelo ARIMA obteve 3,41%, 13,60% e 17,98% de percentual de erro médio para as previsões de 1 dia, 1 semana e 1 mês. O modelo LSTM obteve 8,00%, 7,46% e 23,37% de percentual de erro médio para as previsões de 1 dia, 1 semana e 1 mês. Por fim são apresentadas as considerações finais e os trabalhos futuros.

Palavras-chave: Inteligência Artificial; Séries Temporais; Facebook's Prophet; Amazon DeepAR; Criptomoedas; Aprendizado de Máquina .

ABSTRACT

The data processing capacity fast growth resulting from the evolution of computing power, attached with machine learning and artificial intelligence techniques are able to perform accurate time series predictions. This work addresses the concepts of Artificial Intelligence, Machine Learning, Cryptocurrencies and mainly Time Series calculation models, such as: *Facebook's Prophet*, ARIMA and LSTM. This work aims to verify the ability of these models to predict the value of cryptocurrencies. A theoretical framework is presented and its main references used, is also presented the main technologies that are used in the current days, as the correlated works. The performance metrics adopted to evaluate the models (MAPE and RMSE) and the tests performed using the tools *Facebook's Prophet*, ARIMA and LSTM are presented. The models developed were able to follow the up and down trends of the time series. Facebook's Prophet scored 10.93%, 16.32%, and 40.14% mean error percentage for 1-day, 1-week, and 1-month forecasts respectively. The ARIMA model, on the other hand, obtained 3.41%, 13.60% and 17.98% of average error percentage for 1-day, 1-week and 1-month forecasts. The LSTM model obtained 8.00%, 7.46% and 23.37% of average error percentage for 1-day, 1-week and 1-month forecasts. Finaly, this study present the final considerations and future works.

Keywords: Machine Learning; Cryptocurrencies; Time Series; Facebook's Prophet; Amazon DeepAR; Artificial Intelligence.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ARIMA	Autoregressive Integrated Moving Average
BOVESPA	Bolsa de Valores de São Paulo
BTC	Bitcoin
CO_2	Dióxido de Carbono
CPU	Central Processing Unit
ETH	Ethereum
GB	GigaBytes
GHZ	Gigahertz
GPU	Graphics Processing Units
IA	Inteligência Artificial
IEEE	Institute of Electrical and Electronics Engineers
LSTM	Long short-term memory
MAPE	Mean Absolute Percentage Error
MHZ	Megahertz
MLP	MultiLayer Perceptron
NOAA	National Oceanic and Atmospheric Administration
PETR4	Petróleo Brasileiro SA Petrobras Preference
RAM	Random Access Memory
RNA	Rede Neural Artificial
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RSI	Relative Strength Index
SAR	Stop And Reverse
SMA	Simple Moving Average

TCC	Trabalho de Conclusão de Curso
USD	United States Dollar
UNIPAMPA	Universidade Federal do Pampa
WEKA	Waikato Environment for Knowledge Analysis

SUMÁRIO

1 INTRODUÇÃO

As tentativas de prever o futuro estão presentes na humanidade há séculos, desde os primórdios meteorologistas tentaram prever o clima por conta da agricultura, para garantir a sobrevivência da colheita. Novas técnicas foram desenvolvidas para tentar realizar tais predições, algumas delas sendo: regressões lineares, regressões não lineares e *software* baseados em inteligência artificial. Historicamente, a sociedade vem demonstrando grande interesse neste ramo, um grande exemplo dos dias atuais são as previsões meteorológicas. Segundo Brochado (2017), a cerca de 25 anos atrás a precisão das previsões meteorológicas possuíam cerca de 50% de chance de acerto, evoluindo para 90% nos dias atuais, levando em consideração dois dias no futuro, pois quanto mais longe no futuro se deseja prever, menor será a precisão.

As máquinas com grande capacidade de processamento de dados se tornaram mais comuns e acessíveis no mercado, sendo este o principal recurso utilizado por ferramentas de *softwares* de inteligência artificial. O aprendizado de máquina está presente em uma vasta gama de ferramentas, dentre elas: caixa de *spam* do *e-mail*, carros autônomos, sugestões da *Netflix*, robótica, predições de séries temporais, entre outros.

A Inteligência Artificial, segundo Teixeira (2019), é o avanço tecnológico que possibilita a simulação da inteligência humana através de um sistema que permite a tomada de decisões de forma independente e sem interferência humana, utilizando como base, o reconhecimento de padrões em bancos de dados. Segundo Alpaydin (2021), aprendizado de máquina ou *Machine Learning* pode ser classificado como um subcampo da inteligência artificial. O aprendizado de máquina utiliza principalmente bancos de dados (lugares em que estão presentes as informações necessárias para alimentar o sistema) para realizar seus treinamentos. Essa técnica é considerada a forma mais promissora para se alcançar uma inteligência artificial próxima à humana. Existem três modelos de aprendizado de máquina, estes sendo: aprendizado supervisionado, aprendizado não supervisionado e aprendizado reforçado. Alguns exemplos de ferramentas que são utilizadas para desenvolver modelos de aprendizado de máquina relacionados à previsão de séries temporais são o *Facebook Prophet*[1] e o *Amazon DeepAR*[2], ambos funcionam através do aprendizado supervisionado.

As séries temporais são coleções de observações feitas ao longo do tempo (EHLERS, 2007). A principal característica deste tipo de dado é a dependência entre

[1] https://github.com/facebook/prophet

[2] https://docs.aws.amazon.com/sagemaker/latest/dg/deepar$_how - it - works.html$

as observações vizinhas, logo o interesse é analisar e modelar esta dependência. Outra característica importante é a ordem com que os dados são analisados, neste contexto a ordem com que os dados são analisados é crucial. De acordo com Korstanje (2021), existem diversas técnicas para analisar as séries temporais, algumas destas sendo: redes neurais recorrentes, o modelo ARIMA e suas variações, *Facebook's Prophet* e *Amazon DeepAR*, entre outros.

As criptomoedas são moedas virtuais que permitem que seus portadores possam realizar suas transações de maneira rápida e confiável (PIZZETTI, 2019), estão presentes em uma rede *blockchain*, estas utilizam serviços de criptografia que são responsáveis por proteger as transações e os dados de quem as efetua. A usabilidade de uma criptomoeda é semelhante a do dinheiro em espécie, serve como meio de troca, facilitando assim, transações comerciais. As criptomoedas são vulneráveis a diversos tipos de influências, desde comentários em redes sociais a crises globais. Estes acontecimentos, podem acarretar em um aumento ou decremento expressivo em seu valor. Desta maneira, busca-se estudar e compreender como a série temporal de uma criptomoeda funciona, através da identificação de padrões na variação do preço destas moedas.

1.1 Problema de Pesquisa

É possível identificar e prever um padrão de comportamento das criptomoedas através do uso de modelos de previsão de séries temporais?

1.2 Importância e Motivação da Pesquisa

Os países ao redor do mundo registram um rápido crescimento no número de investidores interessados pelo mercado de cripto ativos, todos procuram formas confiáveis de investir seus recursos. Sabendo-se da volatilidade deste mercado financeiro, torna-se necessário a utilização de métodos de previsão de mercado. A finalidade desta pesquisa é avaliar se é possível prever as variações de criptomoedas e ao mesmo tempo avaliar e comparar o desempenho das ferramentas *Facebook Prophet*, ARIMA e LSTM.

1.3 Objetivo Geral

O objetivo deste trabalho é determinar se é possível prever as tendências de alta e de baixa dos preços de criptomoedas através do uso de ferramentas de IA relativas a previsão de séries temporais, como *Facebook Prophet*, ARIMA e LSTM, usando como estudo de caso o *Ethereum*.

1.4 Objetivos Específicos

- Pesquisar referências bibliográficas e o estado-da-arte relacionado a séries temporais;
- Buscar trabalhos e artigos correlatos a essa pesquisa;
- Identificar as ferramentas utilizadas nos trabalhos correlatos;
- Implementar a solução usando modelos;
- Analisar os resultados obtidos;
- Validar os resultados obtidos através de estatísticas e métricas de erro;
- Escrever o trabalho de conclusão do curso.

1.5 Metodologia

Nesta seção é apresentada a classificação da pesquisa e as etapas definidas para a realização da mesma.

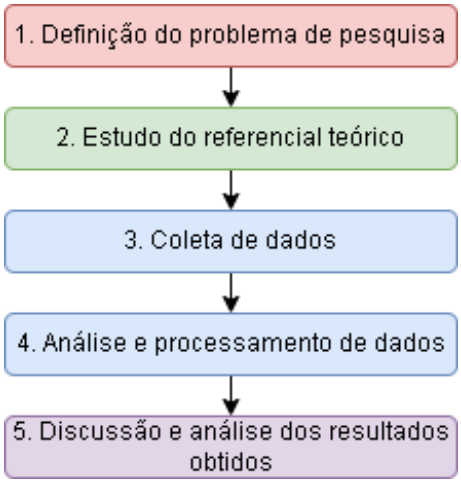
1.5.1 Classificação da Pesquisa

De acordo com Prodanov e Freitas (2013) "O método experimental consiste em submeter os objetos de estudo à influência de certas variáveis, em condições controladas e conhecidas pelo investigador, para observar os resultados que a variável produz no objeto". Logo, esta pesquisa é caracterizada pela manipulação das variáveis relacionadas com o objetivo de estudo, neste caso o objeto de estudo são as predições do preço do *Ethereum* e as variáveis os parâmetros dos modelos de previsão de séries temporais.

1.5.2 Etapas da Pesquisa

As etapas de pesquisa seguem conforme apresentado na figura 1. A definição do problema de pesquisa surgiu da necessidade do estudo de cripto ativos, com o objetivo de definir se é possível mapear o comportamento dos mesmos. Na fase de levantamento e estudo do referencial teórico, foi realizada uma pesquisa com o objetivo de identificar os trabalhos correlatos que utilizam analise de séries temporais. Na terceira etapa, realizou-se a coleta de dados de cripto ativos, através de bancos de dados confiáveis. Na quarta etapa, foi realizada uma analise e o processamento dos dados a fim de identificar padrões e tendências no comportamento de cripto ativos. Por fim, realizou-se uma análise e discussão dos resultados. Após a implementação dos modelos, foi avaliado o funcionamento das soluções desenvolvidas (através de métricas de desempenho). Por fim foi mostrado os resultados obtidos bem como os gráficos de desempenho.

Figura 1 – Etapas da pesquisa.



Fonte: Autor (2022)

1.6 Organização do Texto

No capítulo 1 é apresentada a introdução o problema de pesquisa e os objetivos desta pesquisa. Já no capítulo 2 são expostas as referências bibliográficas tais como: IA, criptomoedas, séries temporais e modelos de predição. Então, o capítulo 3 exibe os principais trabalhos correlatos ao tema de previsão de séries temporais. O capítulo 4 expõe os materiais e métodos utilizados. O capítulo 5 apresenta os resultados e discussões. Por fim, o capítulo 6 apresenta as conclusões finais e trabalhos futuros.

2 REFERENCIAL TEÓRICO

O presente capítulo aborda o levantamento de dados bibliográficos que alicerçaram a construção desta pesquisa. O foco da pesquisa está relacionado a inteligência artificial, aprendizado de máquina, criptomoedas, redes neurais recorrentes, séries temporais e seus modelos de predição. Portanto, o trabalho precisa abordar o conceito de criptomoedas a fim de entender o seu comportamento e possíveis fatores complicantes. Ademais, é necessário explicar o que são séries temporais e apresentar o estado da arte aplicável a esta pesquisa.

2.1 Inteligência Artificial

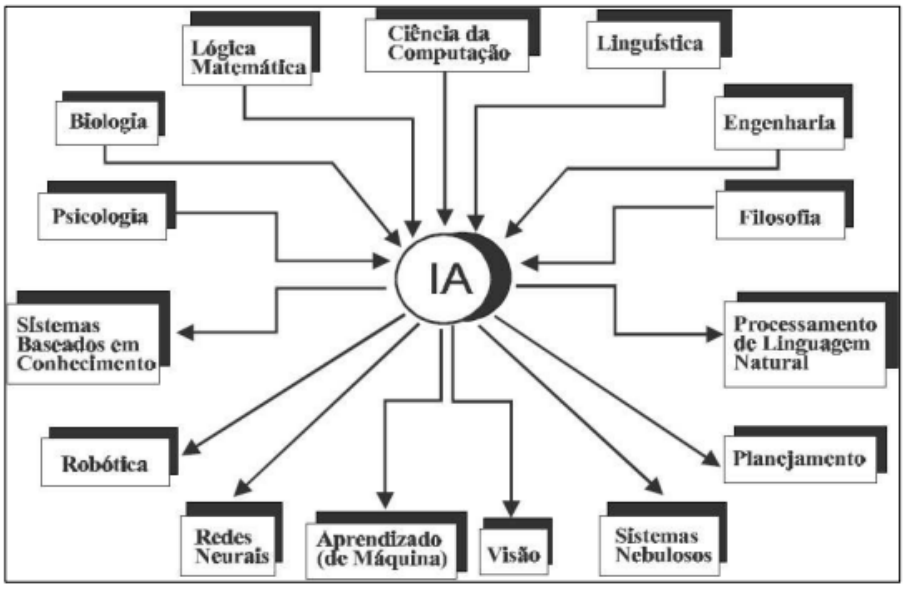
A inteligência artificial é um campo de pesquisa muito promissor, que busca simular uma inteligência humana, através de um sistema que toma decisões de forma independente (TEIXEIRA, 2019). A inteligência artificial esta presente em diversas ferramentas do nosso cotidiano, como por exemplo a caixa de *spam* do e-mail, a IA protege os usuários de e-mails que possuem conteúdo possivelmente malicioso. Além disso, esta presente em plataformas como *Spotify*[3] e *Netflix*[4], conforme o usuário utiliza os serviços da plataforma, a IA coleta os dados e aprende as preferências de cada um, estas informações serão utilizadas para recomendar filmes e séries que o usuário provavelmente assistirá. As lojas virtuais são um exemplo do nosso cotidiano. Elas utilizam técnicas de IA para recomendar produtos que possivelmente o usuário tem interesse baseado nas suas buscas realizadas na internet. A IA está presente em diversos campos de pesquisa como pode ser verificado na figura 2, através de um aglomerado de conhecimentos como: lógica matemática, ciência da computação e engenharia é possível obter um conjunto de saídas "resultados", como: redes neurais, robótica, aprendizado de máquina e entre outros.

De acordo com TOTVS (2019), é possível se obter uma série de benefícios através da utilização da IA. Isso ajuda a melhorar a tomada de decisões das empresas que dependem da análise de dados complexos. Outra grande vantagem da utilização da inteligência artificial, é a capacidade de analisar grandes volumes de dados, possuindo como objetivo o rastreio e detecção de tendências e sazonalidades, uma tarefa próxima do impossível para se analisar manualmente. Portanto, a IA possui um grande potencial

[3] https://www.spotify.com/br/
[4] https://www.netflix.com/br/

de escalabilidade, derivada da necessidade da análise de grandes bancos de dados que estão em constante crescimento. Sabendo que diariamente um grande volume de dados são despejados na rede, graças a grande capacidade de replicabilidade de processos, é possível realizar outra investigação minuciosa destes dados, com o objetivo de encontrar possíveis novas tendências e sazonalidades, assegurando que qualquer fluxo de trabalho se torne escalável.

Figura 2 – Áreas Relacionadas com a Inteligência Artificial



Fonte: GOMES (2010)

A Inteligência Artificial já está presente no dia a dia dos seres humanos, de forma direta através do reconhecimento facial, é possível desbloquear o telefone ao escanear o rosto do usuário ou até mesmo através de analises de comportamento do consumidor, a IA rastreia visitas a sites e produtos, com o objetivo de entender padrões de comportamento, desse modo, gerando relatórios mais detalhados para as empresas criarem ofertas e campanhas de *marketing* mais atrativas a estes consumidores. A IA esta indiretamente presente em diversas ocasiões distintas, como em câmeras de supermercados que são responsáveis por gerenciar o estoque de prateleiras, estas sendo capazes de identificar e notificar a falta de produtos em determinadas seções. Ela esta presente em aplicativos de rotas, estes aplicativos são responsáveis por interpretar em tempo real os dados de trânsito, conseguindo assim, sugerir rotas mais eficientes para os usuários, um grande exemplo para esse caso é o aplicativo *Waze*[5].

[5]https://www.waze.com/pt-BR/live-map/

2.2 Aprendizado de Máquina

O aprendizado de máquina é um campo da inteligência artificial cujo objetivo é desenvolver técnicas computacionais para aprender e construir sistemas capazes de adquirir conhecimento automaticamente (MONARD; BARANAUSKAS, 2003). O sistema de aprendizado é capaz de tomar decisões fundamentado em soluções que foram desenvolvidas no passado. Existem diversos sistemas de aprendizados diferentes, cada um com suas características particulares e comuns, sendo que cada sistema utiliza formas de aprendizado diferentes, como o aprendizado supervisionado, este estando diretamente relacionado com a presente pesquisa. De acordo com Monard e Baranauskas (2003), a indução é a forma de inferência lógica que permite obter conclusões genéricas sobre um conjunto particular de exemplos.

2.2.1 Aprendizado Supervisionado

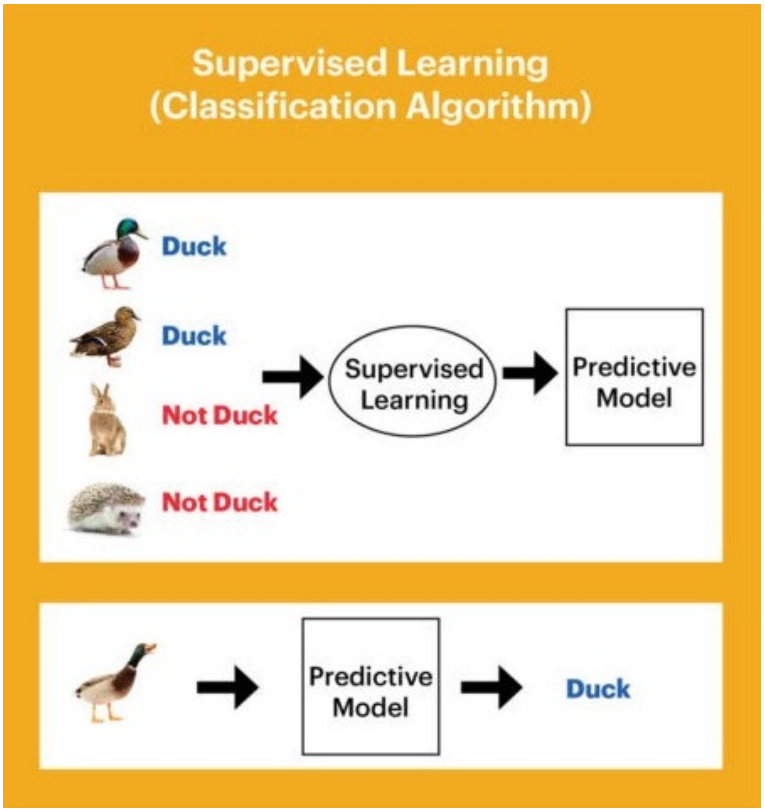
Algoritmos de aprendizado supervisionado utilizam exemplos rotulados como entrada, ao mesmo tempo a saída desejada é conhecida, esta é utilizada para realizar o treinamento dos modelos desenvolvidos (MONARD; BARANAUSKAS, 2003). Como exemplo, um *software* poderia rotular determinadas entradas como (pato) e (não é pato) como pode ser visto na figura 3, o algoritmo recebe uma coleção de entradas e saídas correspondentes a estas entradas, como exemplo: são enviadas muitas fotos de patos, e é dito que estes são patos, ao mesmo tempo, é enviada uma coleção de fotos de outros animais, e é dito que não são patos, após um treinamento exaustivo, o *software* aprenderá a associar entradas reais a saídas corretas, sem que seja necessário fornecer a resposta.

Os dados rotulados são particionados em duas categorias, o conjunto de treinamento é utilizado para treinar e construir o modelo, e o conjunto de testes tem como principal objetivo a validação da ferramenta, de forma que será verificado o comportamento e a precisão do modelo desenvolvido, de forma que esta possa ser ajustada a fim de se obter índices de precisão maiores.

Existem dois tipos principais de aprendizagem supervisionada, estes sendo classificação e regressão (NEVES; DIAS; CORDEIRO, 2018). A classificação é um método que treina o algoritmo para classificar os dados de entrada, estes recebem os dados rotulados e classificados, como o exemplo da figura 3, o algoritmo é treinado para classificar as imagens como (pato) e (não é pato), outro exemplo é a IA responsável por

identificar um e-mail como (spam) ou (não-spam), o algoritmo é classificador dos dados.

Figura 3 – Aprendizado Supervisionado.



Fonte: Kenji (2019)

O método de regressão é caracterizado por algoritmos que são treinados para prever saídas a partir de uma faixa previamente conhecida de valores. Por exemplo, tentar prever a demanda de um determinado produto em um período específico, ou até mesmo tentar prever o valor de um ativo (como exemplo o ouro) ou de uma criptomoeda (como por exemplo o *Ethereum*), através de uma análise de seu comportamento passado. Este comportamento pode ser visualizado a partir da apresentação da série histórica por meio de um gráfico, através de um estudo detalhado com a utilização de algoritmos de IA, onde serão detectadas tendências e sazonalidades no comportamento destes ativos, resultado em possíveis predições comportamentais.

2.3 Criptomoedas

As criptomoedas são ativos virtuais protegidos por criptografia (MILUTINOVIĆ et al., 2018), estando presentes exclusivamente em registros digitais, todas suas operações são realizados e armazenados em uma rede de computadores. Essas moedas surgiram

com a intenção de facilitar transações financeiras ou pagamentos entre indivíduos ou empresas, não sendo necessário a intermediação de uma instituição financeira. O funcionamento destes ativos apoia-se na tecnologia conhecida como *Blockchain*, este sendo uma espécie de livro contábil, responsável por concentrar todas as informações das transações realizadas. Todas as informações são armazenadas em blocos, que possuem um registro de hora e data. Sendo que periodicamente novos blocos são formados e acoplados a um bloco anterior, dessa forma é originada uma "cadeia de blocos". Antes de toda e qualquer informação ser inserida de fato na *Blockchain*, ela deve ser validada e aprovada, através de confirmações de outros nodos da rede, cada transação recebe um código alfa numérico único, que serve como identificador único de cada transação.

O *Bitcoin* foi a primeira criptomoeda criada, sua origem é datada em 2009, foi desenvolvida por um usuário que utilizou o pseudônimo Satoshi Nakamoto (COSTA, 2021). Em 22 de maio de 2010, a primeira compra oficial de um produto físico foi realizada, 10 mil *Bitcoins* foram utilizados para comprar duas pizzas no valor de 41 dólares (BAPTISTA, 2019), a partir deste ponto a moeda passou a ser entendida como dinheiro e meio de troca. Desde então, o *Bitcoin* só agregou valor e ganhou popularidade, sendo aceito como pagamento para todo tipo de produto ou serviço ao redor do mundo.

A maneira inicial de se obter esse ativo, foi a prova de trabalho, segundo Silva e Rodrigues (2016) "o processo matemático realizado pelo minerador ocorre por meio do algoritmo de criptografia *SHA-256*[6] que calcula um *hash*. Esse *software* faz com que o minerador precise descobrir um número inteiro de 4 bytes, denominado de *nonce*, capaz de satisfazer uma desigualdade (inequação) expressa em função desse algoritmo". Estas tarefas são transmitidas através de blocos, após resolvidos são adicionados à *Blockchain*, a fim de proporcionar um incentivo para a validação destes blocos, recompensas são distribuídas para quem contribuísse com a validação destes blocos. Segundo Silva e Rodrigues (2016) "Quando o *Bitcoin* foi lançado, o seu *reward* era de 50,00 BTC".

Estas criptomoedas são guardadas em carteiras, além de serem mantidas chaves públicas e privadas. A chave privada é uma espécie de senha, que permite o usuário assinar as transações e transferir criptomoedas para alguém, esta sendo a única informação necessária para acessar e fazer uso dos fundos. A chave pública funciona como o endereço do usuário na rede, em uma analogia simples, a chave pública pode ser considerada o endereço de *e-mail* e a chave privada a senha.

[6] https://docs.microsoft.com/pt-br/dotnet/api/system.security.cryptography.sha256?view=net-6.0

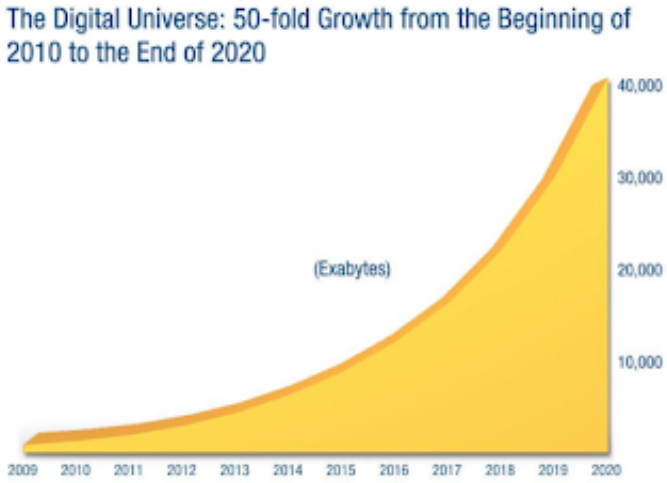
2.4 Séries Temporais

Uma série temporal é um conjunto de observações de uma variável ao longo do tempo, estas observações podem ser realizadas de forma contínua temporalmente, ou por um conjunto discreto de pontos (HANNAN, 2009). Cada observação é representada por um valor específico em determinados instantes, suas aplicações são amplas e essenciais em diversos campos de estudo, como por exemplo: finanças, marketing, economia, ciências sociais, meteorologia e etc. A análise de séries temporais tem como principal intuito a exploração do comportamento passado de variáveis, a fim de prever o comportamento futuro de um determinado problema, através da análise de padrões, tendências e sazonalidades. A principal desvantagem deste método de predições, é a dificuldade de realizar predições para períodos mais longos de tempo, pois são realizadas predições em cima de outras predições. Outro fator que dificulta a precisão destas predições, mais especificamente voltada ao mercado de criptomoedas, é a alta volatilidade que estas apresentam, decorrente de sua vulnerabilidade perante crises econômicas e até mesmo de postagens polêmicas de grandes influenciadores digitais do mercado. Em seu modelo clássico, as séries temporais são compostas de quatro padrões (SANTOS, 2016):

- Tendência (T): Essa componente consiste em um comportamento durante um período superior a um ano e indica qual a direção de deslocamento da série, um exemplo deste padrão é a tendência de crescimento da população mundial;
- Variações cíclicas (C): São variáveis que possuem um padrão de flutuações dos valores em períodos superiores a um ano, além de se repetir com certa periodicidade;
- Variações sazonais (S): Consiste em um comportamento típico em períodos de tempo inferiores a um ano. Como por exemplo o aumento da venda de sorvete durante o verão e o aumento de vendas durante feriados, tais como: natal, dia das mães, dia da criança e outros;
- Variações irregulares (I): São flutuações inexplicáveis, resultados de catástrofes globais "como a pandemia", atentados terroristas, decretos governamentais e até mesmo publicações em redes sociais feitas por influenciadores digitais.

As séries temporais podem ser representadas por meio de funções matemáticas, assume-se que o valor a ser obtido gira em torno da função de outra variável (ou de múltiplas variáveis) (FIGUEREDO, 2008). A função que determina uma série temporal,

Figura 4 – Estimativa de crescimento do volume de dados digitais de 2010 a 2020

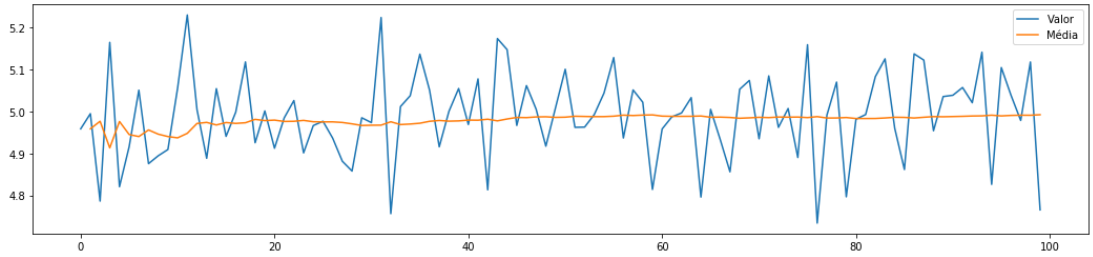


Fonte: Ávila (2017)

não necessariamente será uma função linear, esta pode ter qualquer formato (quadrática, exponencial, logarítmica...). Além disso, elas podem depender de mais de uma variável independente. Porém, grande parte do ferramental utilizado para entender as flutuações econômicas, utilizam apenas funções lineares, por serem mais fáceis de manipular. As técnicas de previsão podem ser divididas em duas classes (RAPOSO, 1992):

1. **Estacionária:** Segundo Raposo (1992), séries estacionárias se caracterizam por manterem um equilíbrio através de uma média constante, como pode ser visto na figura 5.

Figura 5 – Série estacionária

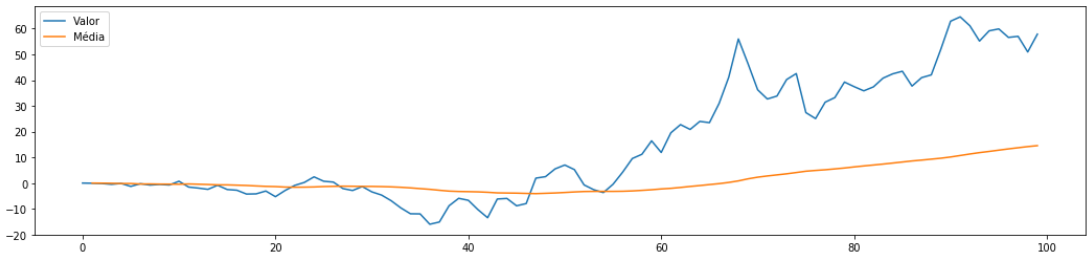


Fonte: Rabelo (2019)

2. **Não estacionária:** Series não estacionárias se caracterizam pela alteração da média ao longo do tempo, como pode ser verificado na figura 6.

Ao realizar uma análise de dados de séries temporais, é importante destacar que a ordem dos dados é crucial (EHLERS, 2007), não podendo ser possível aleatorizar a disposição destes dados. Ao embaralhar as entradas, perturba-se a tendência do conjunto, por consequência serão geradas hipóteses provavelmente incorretas.

Figura 6 – Série não estacionária



Fonte: Rabelo (2019)

2.5 Modelos de Predições de Séries Temporais

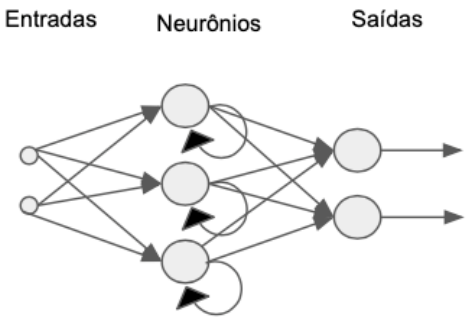
Modelos de séries temporais são muito utilizados para avaliar o comportamento de uma variável ao longo do tempo, possuem como principal propósito a detecção de tendências e padrões históricos contidos nas séries temporais analisadas, utilizando estas observações para realizar predições de valores futuros. O presente seção apresenta algumas das técnicas para realizar predição de séries temporais existentes, como: Redes neurais recorrentes, modelo ARIMA, variações do modelo ARIMA, *Amazon DeepAR* e *Facebook's Prophet*.

2.5.1 Redes Neurais Recorrentes

Redes recorrentes são redes que possuem ao menos uma célula recorrente (WEISS, 2019), ou seja, uma camada com memória, como pode ser visto na figura 7. Estas redes também podem ser parcialmente ou totalmente recorrentes, portanto todos os neurônios se encontram conectados entre si. Diferentes tipos de RNNs são usados para diferentes casos (IBM, 2020), tais como: redes um para um, um para muitos, muitos para um e muitos para muitos.

As redes neurais recorrentes são utilizadas principalmente para realizar a análise de uma sequência de dados, como um vídeo por exemplo, e não um dado estático, como uma imagem. Uma RNN é recorrente pois possui laços embutidos, este *loop* permite que a rede processe o resultado com base nos resultados anteriores, devido isto, as RNNs (Redes neurais recorrentes) são consideradas redes com memória (WEISS, 2021). Através disto, as RNNs podem armazenar informações ao processar novas entradas. Segundo Jones (2017), essa memória as torna ideais para tarefas de processamento em que as entradas

Figura 7 – Redes Neurais Recorrentes



Fonte: Weiss (2021)

anteriores devem ser consideradas (como dados da série temporal).

Figura 8 – Rede neural recorrente do Google

Fonte: Autor (2022)

Um grande exemplo de uma Rede Neural Recorrente, que esta presente no dia a dia de uma grande parcela da população mundial, é o campo de busca do *Google*, este sugere palavras para completar a sua pesquisa (DUARTE, 2019), baseado em uma coleção de grandes volumes de dados das palavras consecutivas mais frequentes utilizadas pelos usuários. Esses dados alimentam a rede neural, que é responsável por analisar e prever a próxima palavra ou palavras a serem escritas na sentença. Então, quando o usuário digita algo como "Previsão do...", o algoritmo sabe que é mais provável que este indivíduo digite "tempo"para completar a frase como pode ser verificado na figura 8, justamente por possuir uma maior recorrência nos dados analisados.

LSTM ou *Long short-term memory*, é uma das arquiteturas de RNNs mais populares (ARRUDA, 2021), estas adicionam conceitos de passagens a arquitetura, estes sendo responsáveis por controlar a passagem de informações entre suas etapas. Estas passagens são aplicadas na entrada e na saída da rede, e podem possuir os valores 0 e 1, que indicam quando uma informação deve ser propagada. Segundo Arruda (2021), "A LSTM utiliza três passagens diferentes: *forget gate*, *input gate* e *output gate*", estas são representados por *Ft*, *It* e *Ot*, respectivamente, todos possuindo como função o filtro de informações em suas respectivas etapas. Essas passagens ajudam a rede a mapear características temporais do problema, em que as informações serão utilizadas ou esquecidas quando a rede julgar pertinente. Este modelo ainda foi pouco explorado comparado a outros métodos de previsão que serão discutidos neste trabalho. Porém, demonstra-se promissor. Como qualquer problema de aprendizado profundo, as LSTMs requerem grandes quantidades de dados e poder computacional, e possuem como objetivo gerar dados esperados.

2.5.2 Modelo ARIMA

O modelo de previsão ARIMA é um algoritmo de estatísticas usado para predições de série temporais. Ele captura várias estruturas temporais (organizações com padrão de tempo) no conjunto de dados de entrada. Segundo Arce e Mahía (2003), em 1970, Box (2013) desenvolveu um corpo metodológico destinado a identificar, estimar e diagnosticar modelos dinâmicos de séries temporais em que a variável tempo desempenha um papel fundamental.

A modelagem ARIMA se tornou popular por seu alto índice de sucesso em fazer predições. Estima-se que em muitos casos, seus resultados são mais confiáveis do que as predições de outras modelagens, especialmente quando aplicado em predições de curto prazo (CAMPOS; CLEMENTE; CORDEIRO, 2006). Uma das desvantagens do modelo ARIMA, é a incompatibilidade com sazonalidades ou dados com ciclos repetidos.

ARIMA é a união de três conceitos (DUTRA, 2019):

- Auto regressivo (AR): Indica que a variável de interesse é regressada em seus próprios valores anteriores. Ou seja, o modelo prevê valores futuros com base em valores anteriores. Isso é semelhante a prever que amanhã estará frio, pois durante toda a semana a temperatura foi baixa.

- Integrado (I): A parte integrada indica que os valores foram substituídos com a diferença entre seus valores e os valores anteriores. Este processo diferenciador pode ter sido realizado mais de uma vez.
- Média Móvel (MA): Indica que o erro de regressão é na verdade uma combinação linear dos termos de erro, cujos valores ocorreram contemporaneamente e em vários momentos no passado.

Este modelo é representado por ARIMA(p,d,q), sendo (p,d,q) a representação da ordem do modelo (CAMPOS; CLEMENTE; CORDEIRO, 2006), este modelo possui quatro etapas, estas sendo:

- **Identificação:** Consiste em determinar os filtros (p, d, q), e a melhor ordem que representa a serie temporal. Inicialmente é determinado se a série é ou não estacionária, caso a série seja estacionária, o filtro d é considerado zero, logo, não é necessário realizar diferenciação para torná-la estacionária, neste caso, encontra-se ARIMA(p,0,q). Segundo CAMPOS, CLEMENTE e CORDEIRO (2006) "Se a série não for estacionária, faz-se necessário diferenciações para torná-la estacionária". A maioria das séries econômicas, tornam-se estacionárias com apenas uma ou duas diferenciações.
- **Estimação:** Após a determinação dos valores p, d e q, é iniciada a fase de estimação dos parâmetros do modelo de regressão. CAMPOS, CLEMENTE e CORDEIRO (2006) afirma que, qualquer que seja o método, o processo de estimação é extremamente trabalhoso e requer o uso de *software* específico. Experimentos realizados indicam que o método de máxima verossimilhança é superior ao de mínimos quadrados, quando o tamanho da série é pequena.
- **Verificação:** Esta etapa é responsável por verificar a consistência do modelo, portanto é verificado se existem parâmetros em excesso, se são relevantes e se os erros resultantes são auto correlacionados.
- **Previsão:** Após as etapas anteriores, o modelo é testado por meio de diversas simulações. O objetivo central é prever os valores futuros em uma série temporal.

2.5.3 Variações do modelo ARIMA

Algumas variantes do modelo ARIMA foram desenvolvidas com o objetivo de suprir as necessidades para casos específicos, alguns desses modelos são:

- SARIMA: Ou (Seasonal ARIMA), foi desenvolvido para casos em que há sazonalidades em series temporais.
- VARIMA: Ou (Fractional ARIMA) foi desenvolvido para casos em que há várias séries temporais como vetores.
- SARIMAX: É um modelo sazonal do ARIMA, X representa variáveis exógenas, essas são variáveis de séries temporais paralelas que não são modeladas diretamente por meio de processos AR, I ou MA, mas são disponibilizadas como uma entrada ponderada para o modelo.

2.5.4 *Facebook's Prophet*

O *Prophet*[7] surgiu da necessidade de análise de um grande volume de predições de negócios que estavam surgindo (RAFFERTY, 2021), os analistas através de ferramentas de previsão limitadas, não eram capazes de proporcionar a vazão de análises desejada, por conta disto, internamente na empresa *Facebook*, Sean J. Taylor e Ben Letham desenvolveram o *Facebook's Prophet*, e lançaram ao público em 2017. Esta é uma poderosa ferramenta utilizada para desenvolver e otimizar predições, esta sendo capaz de entender fatores relevantes que conduzirão seus resultados a maior precisão. Esta ferramenta foi projetada para ser de fácil utilização, tanto iniciantes quanto usuários avançados são capazes de realizar predições (KORSTANJE, 2021), não sendo necessário possuir conhecimento avançado em matemática e de estatísticas por traz das técnicas de predições de séries temporais.

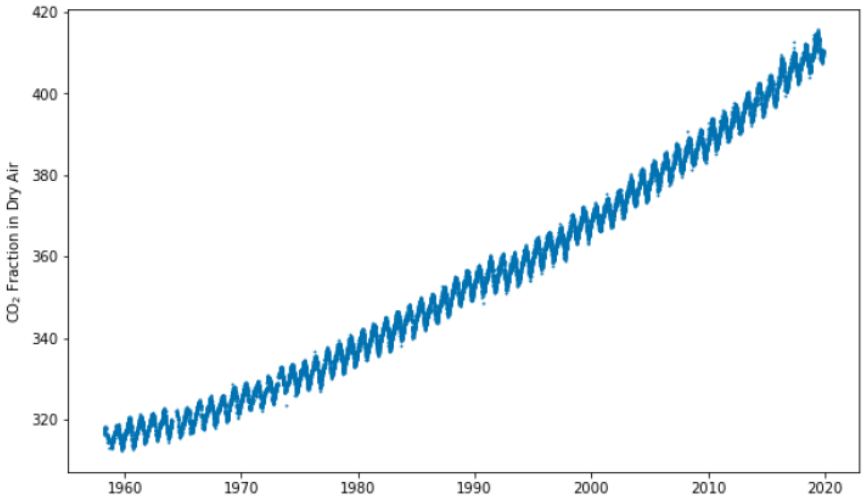
Esta ferramenta possui código aberto, significando que todo o código da aplicação esta disponível para ser modificado e inspecionado (RAFFERTY, 2021), isto trás um grande potencial a ferramenta, pois os usuários podem encontrar e arrumar *bugs*, bem como criar novas funcionalidades, sejam elas de propósito geral ou específico para as suas necessidades. Além disto, esta ferramenta pode ser utilizada através das linguagens de programação *Python*, C e R, todas contam com uma vasta gama de bibliotecas a disposição para serem utilizadas. A única desvantagem deste modelo, é a complexidade que estas bibliotecas podem adicionar a aplicação. O *Prophet* foi projetado para ser capaz de lidar com qualquer tarefa de previsão (RAFFERTY, 2021), que apresente as seguintes características:

[7] https://facebook.github.io/prophet/

- Dados que são capturados ao longo do tempo, sejam segundos, minutos ou horas, ao longo de dias ou semanas, com idealmente pelo menos um ano de histórico;
- Que possuam efeitos de sazonalidade que ocorram diariamente, semanalmente e/ou anualmente, como por exemplo o número de vendas de chocolate ao ano, que provavelmente se encontrará uma sazonalidade pela proximidade do feriado de páscoa;
- Outros eventos especiais que ocorrem de forma irregular;
- Mudanças de tendência, que podem ser originados por uma série de eventos, como catástrofes globais como a guerra e a pandemia.

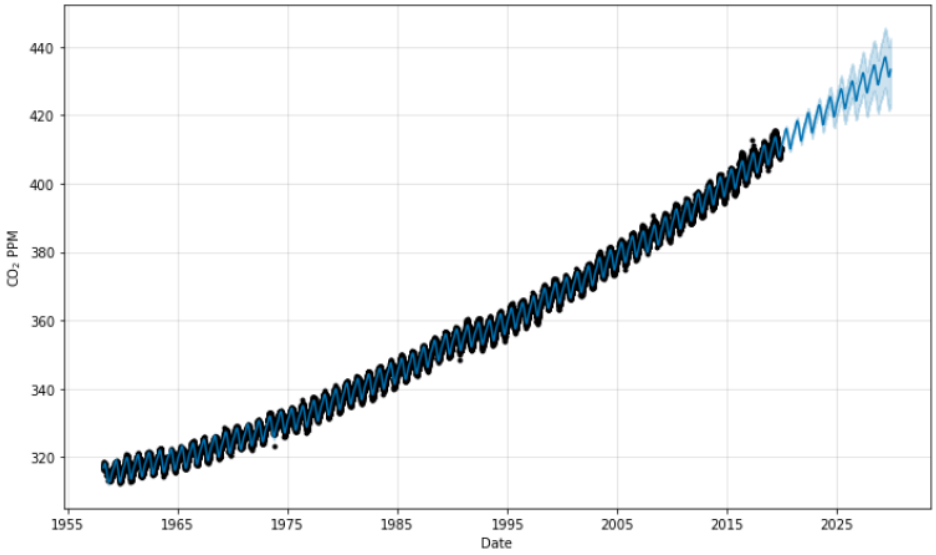
Um exemplo de sua aplicação, presente no livro *Forecasting Time Series Data with Facebook Prophet* escrito por Greg Rafferty, apresenta um estudo iniciado em março de 1958, realizado por Charles David Keeling. Charles coletou amostras de dióxido de carbono diariamente durante três anos, finalmente em 1961 foram publicados os dados coletados, foi possível verificar que os níveis de CO_2 possuem fortes variações sazonais, e que seus níveis estavam crescendo de forma constante, uma tendência que mais tarde foi denominada de *Keeling Curve*. *National Oceanic Atmospheric Administration* (NOAA), começou a realizar observações paralelas até os dias atuais, como pode ser verificado na figura 9. Este conjunto de dados possui por volta de dezenove mil observações diárias que foram realizadas ao longo de 53 anos, através de 12 linhas de código com a linguagem de programação *Python*, o autor realizou uma previsão quanto aos índices de CO_2 que estarão presentes na atmosfera em 10 anos no futuro, como pode ser verificado na figura 10.

Figura 9 – A Curva de Keeling



Fonte: Rafferty (2021)

Figura 10 – Previsão realizada pelo *Prophet*



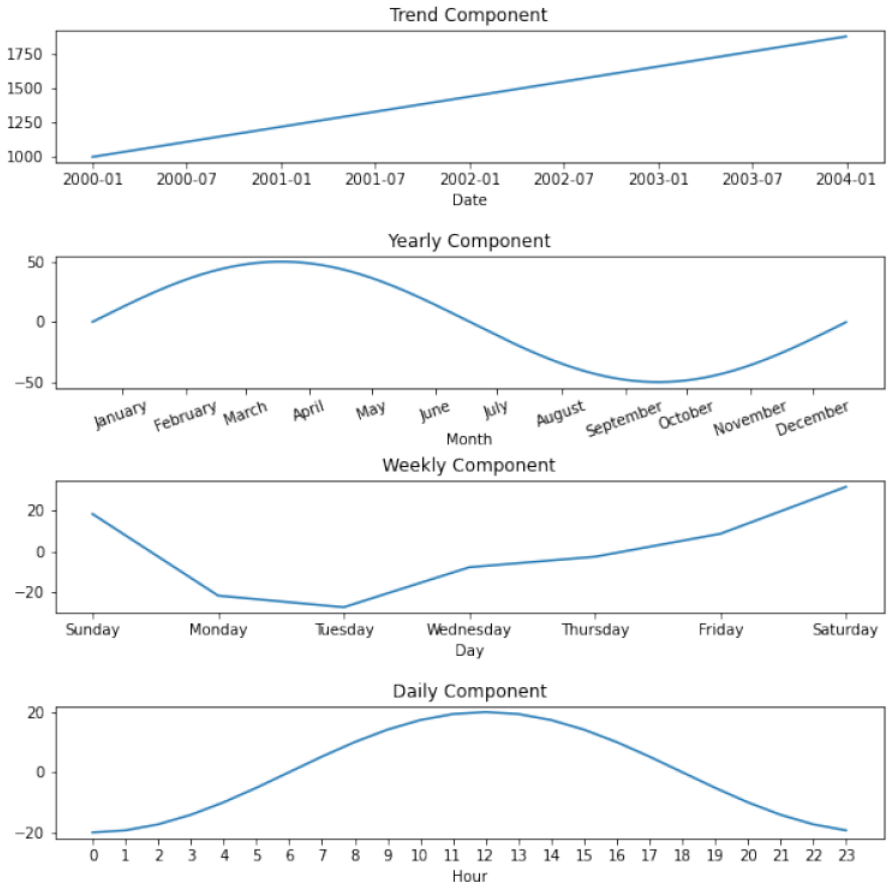
Fonte: Rafferty (2021)

Outro exemplo é um modelo de vendas de uma pequena loja de varejo, cuja analise pode ser visualizada na figura 11, (também presente no livro de Greg Rafferty) que analisa o histórico de vendas entre 1° de Janeiro de 2000 e o fim de 2003, esta analise é capaz de detectar que durante a semana, as vendas tendem a ser menores durante a terça-feira, porém durante o resto da semana elas recebem um constante aumento, até atingir o seu pico no sábado (*Weekly Component*). Ademais, é possível analisar que o pico das vendas diárias é atingido por volta do meio dia (*Daily Component*), bem como os meses que 50 vendas a cima e a baixo da média são realizadas (*Yearly Component*) e por fim a tendencia geral do negócio, que prevê um constante crescimento no negócio (*Trend Component*). Como visto anteriormente, o *Prophet* consegue obter resultados realizando um pequeno esforço (RAFFERTY, 2021), usuários iniciantes podem aplicar 12 linhas de código e obter resultados significativos, enquanto usuários mais experientes podem adicionar uma série de ajustes e recursos ao modelo, a fim de obter resultados mais precisos.

2.5.5 Amazon DeepAR

O DeepAR é um modelo de aprendizagem utilizado para prever séries temporais escalares, que utiliza redes neurais recorrentes (RNNs) para prever séries temporais univariadas ou multivariadas. O grande diferencial desta ferramenta, é a sua capacidade de compreender, em um único modelo, diversas séries temporais ao mesmo tempo, este é projetado para se beneficiar de correlações entre várias séries temporais, tornando-o

Figura 11 – Gráficos de comportamento e predições do *Prophet*



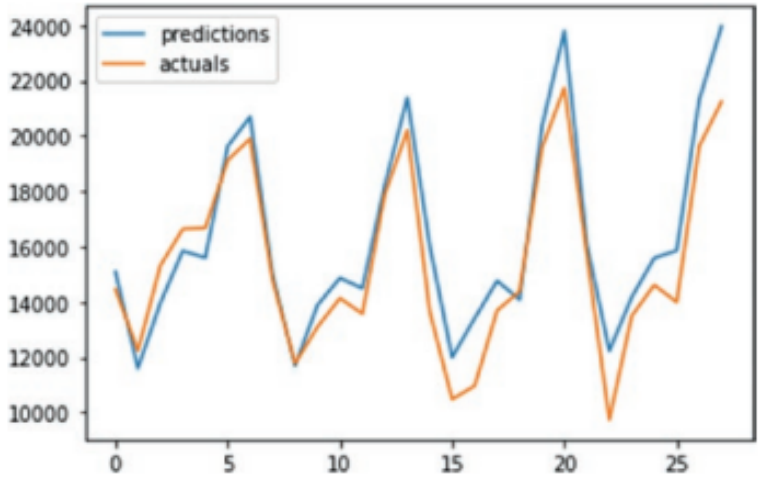
Fonte: Rafferty (2021)

um ótimo modelo para previsão multivariada, porém este modelo também pode ser utilizado em aplicações que possuem apenas uma série temporal. A interface é de fácil utilização para a construção de modelos, esta sendo uma das vantagens da utilização desta ferramenta. O *DeepAR* concorre diretamente com o *Facebook's Prophet*, ambos oferecem interfaces simples. Como usuários do modelo DeepAR, não há muito controle sobre as variáveis e o funcionamento interna do algoritmo, o modelo pode ser considerado uma caixa preta (KORSTANJE, 2021).

Ao utilizar esta ferramenta, são realizadas muitas predições por padrão, cada predição traça a rota de um futuro alternativo dada uma certa circunstancia. Geralmente busca-se apenas uma predição, porém é possível realizar uma média de todas as predições geradas, obtendo uma previsão geral. Durante a utilização da ferramenta, é possível definir alguns dados, como o número de predições que serão realizadas e também escolher a CPU(*Central Processing Unit*) ou a GPU(*Graphics Processing Units*), caso o usuário tenha este tipo de hardware para realizar predições.

O problema desta abordagem esta presente na aleatoriedade das predições

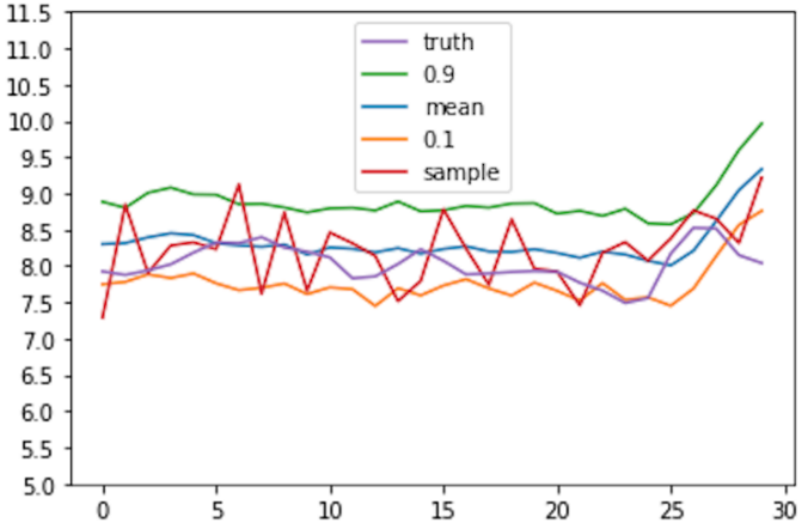
Figura 12 – Predições do DeepAR



Fonte: Korstanje (2021)

alternativas, pois ao executar o código múltiplas vezes, não se encontra-rá a mesma média final no resultado, (diferente dos modelos *Prophet* e ARIMA que sempre obtém os mesmos resultados, por conta de serem modelos que utilizam regressão linear) como pode ser observado na figura 12, a cor laranja representa a predição atual, e a cor azul representa uma predição anterior. Esta pode ser considerada a desvantagem do *DeepAR* (KORSTANJE, 2021). Um bom exemplo de sua aplicação esta no artigo produzido por Simon (2018), onde foram analisados dados de medições diárias da temperatura ambiente, coletados entre 1880 e 2014. Analisando estes dados Simon (2018) observou que a temperatura possui uma tendência ascendente.

Figura 13 – Predições realizadas pelo *DeepAR*



Fonte: Simon (2018)

Após isto, Simon (2018) utilizou a ferramenta *DeepAR* para prever a temperatura dos últimos 30 dias de 1984 e comparar com a realidade, como pode ser verificado na figura 13. Simon (2018) conclui que o *DeepAR* é uma excelente ferramenta para realizar predições, pois não é necessário escrever nenhum código de treinamento, logo o pesquisador pode focar em manipular os hiperparâmetros a fim de obter os melhores resultados possíveis.

3 TRABALHOS CORRELATOS

Nesta capítulo são apresentadas pesquisas que relatam aplicações e conceitos alinhados a presente proposta de estudo, com a finalidade de demostrar os principais tópicos levantados pelos autores, bem como as ferramentas e linguagens que foram utilizadas para realizar a pesquisa, qual o objeto alvo de estudo e os resultados obtidos por cada autor. Por fim será apresentada uma analise comparativa destes trabalhos com a presente pesquisa.

3.1 Predição de Séries Temporais aplicada ao mercado de Criptomoedas

As criptomoedas chamam a atenção de diversos investidores ao redor do mundo, graças a isso, diversos autores buscam desmistificar este mercado através da utilização de técnicas de análise de séries temporais. O trabalho de Almeida (2019), possui como principal objetivo o desenvolvimento de um modelo que utilize uma rede neural artificial que ajude a prever os melhores momentos para se comprar e vender o *Bitcoin*. Foi utilizada uma Rede Neural Recorrente (RNN), para realizar tais predições, através da linguagem de programação *Python*.

No artigo de Almeida (2019), o autor explica que foi utilizado um banco de dados que contém o histórico do valor do *Bitcoin* durante o período de 28/11/2014 até 29/11/2019. Esta base de dados foi extraída da plataforma CryptoDataDownload[8], que proporciona os dados de diversas criptomoedas, estes dados são disponibilizados no seguinte formato:

- **Data:** Dia analisado
- **Símbolo:** Combinação da moeda negociada em sua paridade, *Bitcoin* para dólar (BTC/USD)
- **Abertura:** Valor inicial
- **Máxima:** Valor máximo atingido
- **Mínima:** Valor mínimo atingido
- **Fechamento:** Último valor atingido
- **Volume BTC:** Quantidade total negociada em Bitcoin
- **Volume USD:** Quantidade total negociada em dólar

[8]https://www.cryptodatadownload.com/data/northamerican/

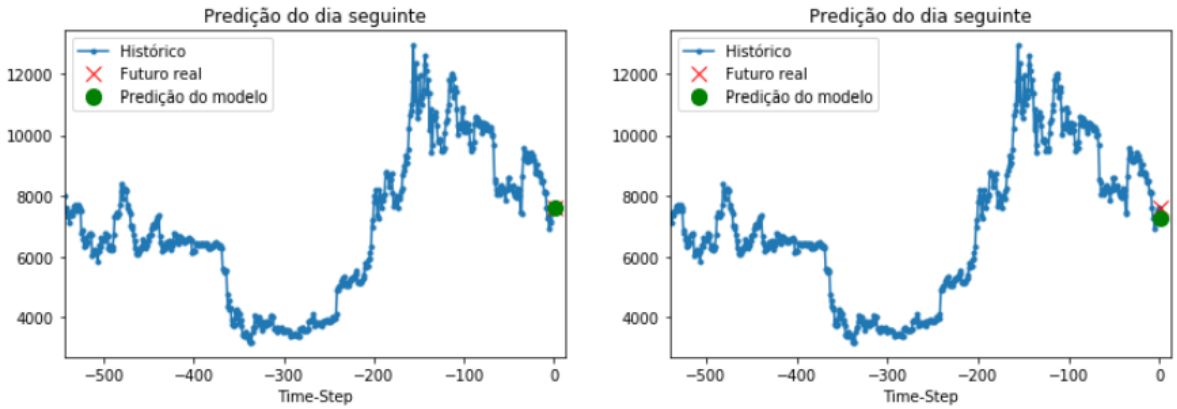
Durante a preparação do conjunto de dados, foi realizado o pré-processamento. Nesta etapa foram feitos os tratamentos necessários a serem aplicados no vetor de características que antecedem a entrada do treinamento, que é dividido nos seguintes passos:

1. Ordenação dos dados ascendentemente pelas datas
2. Somente a coluna de fechamento (dados da curva) são copiados para o novo vetor de características, denominado "values".
3. Vetores referentes a cada indicador são calculados com a biblioteca *Ta-Lib*[9]
4. Vetores calculados no passo anterior são concatenados ao novo vetor de características
5. Remoção de dados inválidos ou nulos do vetor de características gerado

Para a realização destes experimentos, Almeida (2019) utilizou máquinas disponibilizadas pelo ambiente do Google Colab[10], que possuem as seguintes especificações:

- **GPU:** Tesla K80 3.7, 2496 CUDA cores, 12GB GDDR5 VRAM
- **CPU:** Xeon Processor @2.3Ghz (1 core, 2 threads)
- **RAM:** Aproximadamente 12.6 GB
- **Disco:** Aproximadamente 33GB

Figura 14 – Experimentos do período I de 2014 à 2019

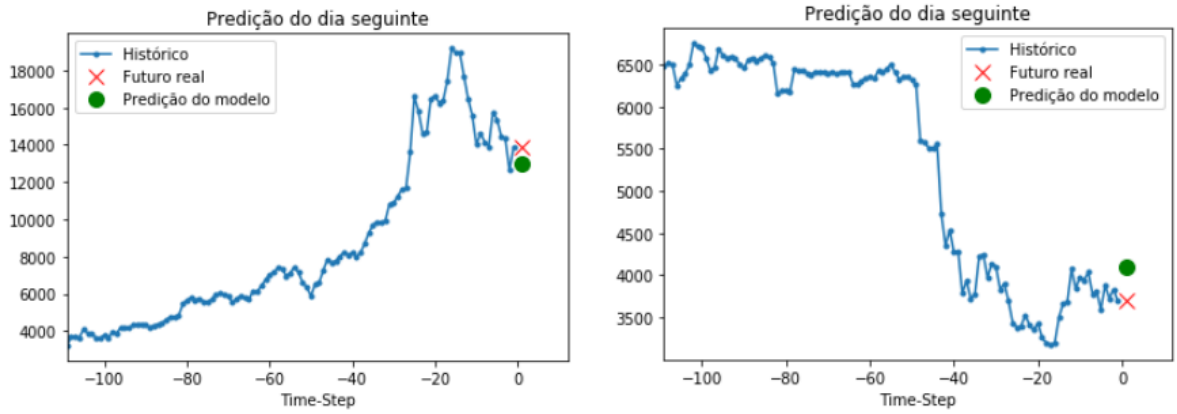


(a) Curva (b) Curva + RSI + SMA + SAR

Fonte: Almeida (2019)

[9]https://www.ta-lib.org/
[10]https://colab.research.google.com/

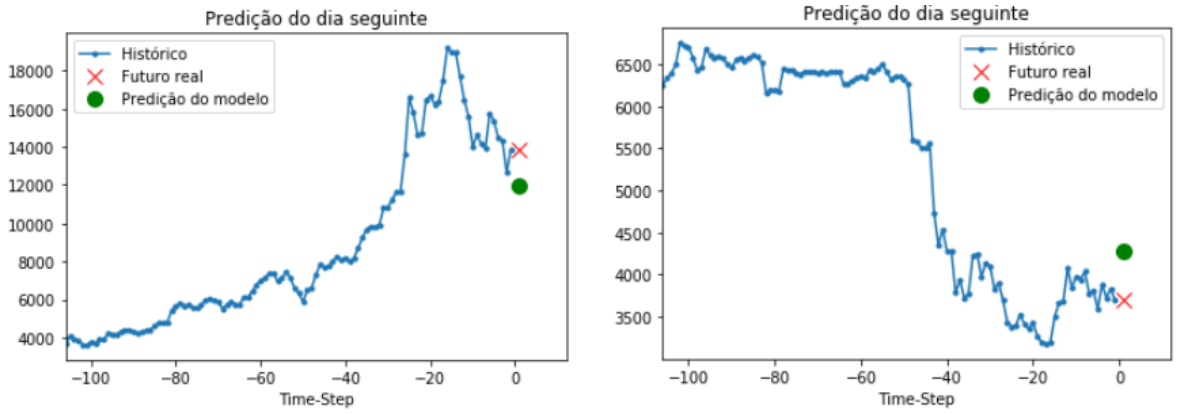
Figura 15 – Experimentos com dados da Curva dos teste



(a) 2017 (b) 2018

Fonte: Almeida (2019)

Figura 16 – Experimentos com dados da Curva + RSI + SMA + SAR



(a) 2017 (b) 2018

Fonte: Almeida (2019)

Nos experimentos de Almeida (2019), foram realizadas combinações dos indicadores (RSI, SMA e SAR) com a curva do gráfico, sendo obtidos os resultados que podem ser verificados nas figuras 14, 15 e 16. Almeida (2019) observou que os gráficos presentes nas figuras 15 e 16 se mostram ineficientes por conta do período em que foram realizadas as predições ser um momento de alta volatilidade da moeda. Almeida (2019) conclui que as redes neurais recorrentes se demonstram uma ferramenta promissora para a resolução do problema estudado, contudo, o *Bitcoin* demonstra ter uma volatilidade maior do que ativos convencionais, por conta de ser vulnerável a influências externas, como especulações em redes sociais e noticias de impacto mundial. Almeida (2019) acredita que para se obter predições mais precisas, tais métodos de predição necessitam de mais informações de entrada além dos dados observados nos gráficos.

3.2 Séries Temporais para Predição de Finanças no contexto de Criptomoedas

O trabalho de Garcia (2021) utilizou as técnicas dos modelos ARIMA, Redes Neurais do tipo *Long Short Term Memory* (LSTM) e o método de vetores de suporte (SVR), para estudar e tentar prever o comportamento do *Bitcoin*.

A linguagem *Python*[11] foi utilizada por Garcia (2021) para realizar a coleção de dados, modelagem da solução e testes. Foram utilizadas as bibliotecas *Mathplotlib*[12] para plotagem de gráficos e imagens, *scipy*[13] e *statsmodels*[14] para a criação do modelo ARIMA e realização de testes, a biblioteca *keras* para instâncias do modelo LSTM e por fim a biblioteca *scikit-learn* para instâncias do modelo SVR.

O conjunto de dados utilizado na pesquisa de Garcia (2021) foi extraído do site *CoinMarketCap*[15], sendo correspondentes ao período de 28 de abril de 2013 à 12 de março de 2021, e são compostos por: Data do registro, valor de abertura, valor de máxima, valor de mínima, valor de fechamento, quantidade total negociada e valor total em circulação, apenas serão utilizadas o valor de fechamento da moeda e suas respectivas datas, este conjunto conta com 2.876 amostras.

Após os testes Garcia (2021) constatou que o modelo LSTM apresentou um desempenho mais satisfatório comparado com os modelos ARIMA e SVR, pois possui um índice de erro de predição menor do que os demais modelos, como pode ser visualizado na figura 17, neste contexto, quanto menor o valor melhor é o desempenho do modelo.

Figura 17 – Comparação de desempenho para períodos de baixa volatilidade

	Índices de Avaliação		
Método	RMSE	MAPE	U
LSTM	429,883	3,66	1,824
ARIMA	2.470,216	34,849	1,031
SVR	1.555.376	12,22	1,16

Fonte: Garcia (2021)

Em contra partida, o modelo SVR obteve um desempenho superior ao ser aplicado em períodos mais voláteis da série, como pode ser observado na figura 18. Apesar de terem obtido percentuais de erro maiores, o ARIMA e o SVR mostraram-se superiores a rede LSTM, sugerindo um potencial superior de modelagem, quando são realizadas

[11] https://www.python.org/
[12] https://matplotlib.org/
[13] https://scipy.org/
[14] https://www.statsmodels.org/stable/index.html
[15] https://coinmarketcap.com/

predições com múltiplos passos a frente.

Figura 18 – Comparação de desempenho para períodos de alta volatilidade

	Índices de Avaliação		
Método	RMSE	MAPE	U
LSTM	7.892,22	18,29	4,01
SVR	5.369,25	9,47	2,53

Fonte: Garcia (2021)

Garcia (2021) conclui que os métodos de aprendizagem de máquina são eficazes para modelar e realizar predições de séries temporais de maneira geral, porém apresentam dificuldades quando aplicadas ao contexto das criptomoedas, justamente por se tratar de um mercado de alta volatilidade. De maneira geral, após avaliar os resultados obtidos, foi possível observar que os modelos aplicados são capazes de realizar predições de maneira eficiente.

3.3 Previsão de Séries Temporais no mercado financeiro de ações com o uso de Rede Neural Artificial

O mercado de ações é visado por muitos investidores ao redor do mundo, por conta do potencial de lucratividade que este mercado oferece. O trabalho de Carvalho (2018) possui como principal objetivo utilizar uma rede neural artificial com as arquiteturas *MultiLayer Perceptron* (MLP) e *Random Walk* (RW), sendo executada através da ferramenta *R-Studio*[16], que utiliza a linguagem de programação *R*, foram utilizadas as bibliotecas *neuralnet*[17] e *tseries*[18], visando realizar predições sobre o valor de ativos da Bolsa de Valores de São Paulo (BOVESPA). A rede MLP é estruturada em três camadas neurais, estas sendo a camada de entrada, intermediária e de saída. A camada de entrada recebe os estímulos do ambiente a ser classificado e é conectada à camada intermediária, esta é responsável por extrair a maioria das informações comportamentais da aplicação, por fim a classe intermediária é conectada a saída que informa a que classe pertence à amostra. O algoritmo de *Random Walk* (passeio aleatório) é um objeto matemático que descreve um percurso que é descrito por uma série de passes aleatórios. É utilizado em diversas áreas de conhecimento, dentre elas a área econômica, este sendo o objeto de estudo do trabalho de Carvalho (2018), pois o preço flutuante de uma ação pode ser

[16] https://www.rstudio.com/
[17] https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf
[18] https://cran.r-project.org/web/packages/tseries/index.html

aproximado por um modelo de *Random Walk*. O trabalho de Carvalho (2018) possui mais detalhes e explicações sobre o funcionamento do algoritmo *Random Walk*.

O conjunto de dados utilizados na pesquisa de Carvalho (2018) foi extraído da base de dados da BOVESPA, um total de 4.466 registros da ação *PETR4* da empresa Petrobras foram obtidos, estes sendo separados em dois lotes, um de 2.223 registros do período de 02/01/2000 à 30/12/2009 e outro lote de 1.973 registros do período de 04/01/2010 à 28/12/2017. Os dados selecionados para a analise foram: datas das operações e o preço efetivo de fechamento do pregão diário.

Após a realização de múltiplas predições e a obtenção dos seus resultados (presentes no quadro da figura 19), Carvalho (2018) observou que o modelo de *Random Walk* mostra-se mais eficiente do que a Rede Neural MLP, sua taxa de aprendizagem de 0,8 possui erro médio de 2,49% contra 4,99% da rede neural. O modelo de *Random Walk* obteve erro médio de 1,89% contra 7,58% da rede neural. Segundo Carvalho (2018) "Analisando-se os resultados obtidos depreende-se que ambos os processos são demasiadamente bons para a previsão de séries temporais, acompanhando a não linearidade do comportamento da série e trazendo resultados muito próximos do observado".

Carvalho (2018) conclui que as redes neurais podem ser usadas para a realização de predições de séries temporais, estas apresentam resultados muito favoráveis ao obterem predições próximas dos valores observados, indicando que estas ferramentas podem ser de grande auxilio para analistas financeiros.

3.4 Redes Neurais Artificiais para Previsão de Séries Temporais no mercado acionário

O estudo realizado por Marangoni et al. (2010) tem como objetivo utilizar redes neurais artificiais para verificar se é possível prever o preço futuro de fechamento de ações, através das habilidades de aprendizado das redes neurais, os dados escolhidos para serem analisados foram os preços de fechamento diário das ações da Petrobras (*PETR4*). O período compreendido de análise foi de janeiro de 1999 a maio de 2010. Marangoni et al. (2010) utilizou como ferramenta o *MATLAB r2009a*, foi desenvolvido um programa de mais de 300 linhas de código, utilizando a linguagem de programação C++ e também foi utilizada a ferramenta *Plot Tools* (que faz parte da biblioteca *matplotlib*) para a montagem dos gráficos de previsão.

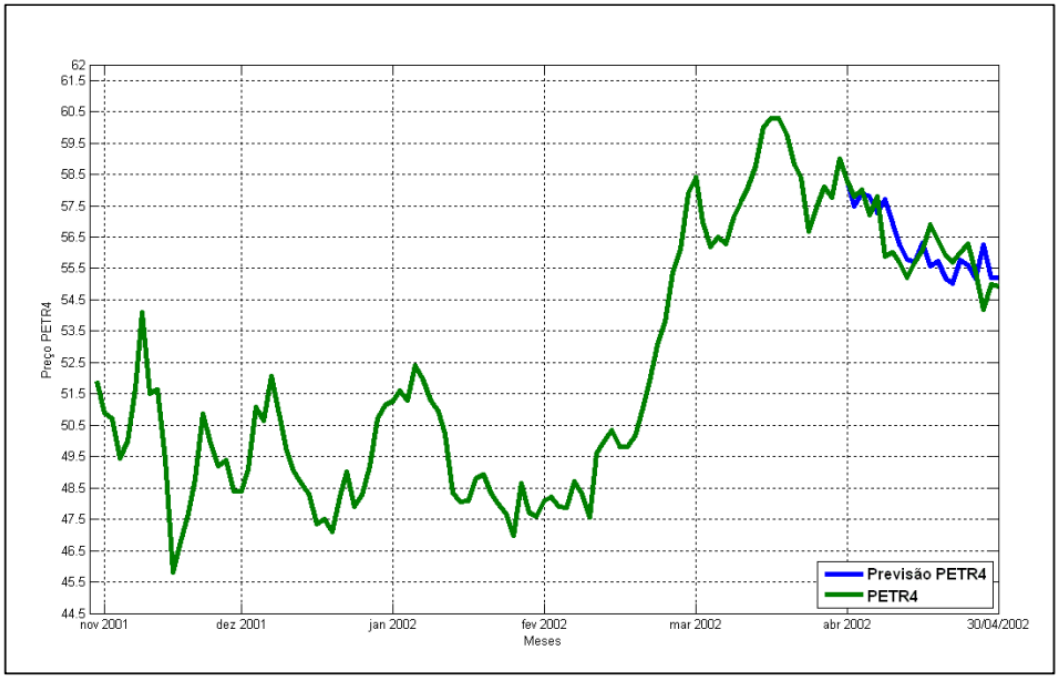
Figura 19 – Quadro comparativo dos resultados entre a rede neural MLP com a hipótese de Random Walk

Resultados						
	Cenário 1			Cenário 2		
Experimento	Interações	Erro RMSE - MLP	Erro RMSE - RW	Interações	Erro RMSE - MLP	Erro RMSE - RW
1	1230	0,057659278	0,025535941	660	0,114504915	0,019249829
2	1760	0,026703787	0,02424997	1360	0,133119741	0,019165655
3	1370	0,051365495	0,025742461	870	0,163654717	0,018089722
4	880	0,035786861	0,024461559	970	0,037707915	0,019378211
5	750	0,033425514	0,025251587	950	0,116923002	0,019327209
6	1150	0,057466288	0,024541882	650	0,039374456	0,018482957
7	910	0,069608328	0,024651951	200	0,091490599	0,018824853
8	730	0,043099599	0,024588257	1410	0,032347052	0,018327835
9	1020	0,091900937	0,02372751	1170	0,04095351	0,018655488
10	2110	0,027339725	0,024984203	820	0,075487526	0,018555133
11	1840	0,028145167	0,026149424	1420	0,100892976	0,019149326
12	1300	0,0423	0,025489412	260	0,106325683	0,018525667
13	890	0,078071939	0,024814822	490	0,027002221	0,019122994
14	1840	0,059189989	0,024289655	1020	0,05988059	0,019128395
15	1340	0,058555567	0,025196787	750	0,039719557	0,019082994
16	1180	0,03745168	0,025054243	540	0,032788732	0,019627004
Média		**0,049879385**	**0,024920604**		**0,075760825**	**0,018918329**
Desvio Padrão		**0,01910817**	**0,000627941**		**0,042947721**	**0,000432603**
Percentual		**4,99%**	**2,49%**		**7,58%**	**1,89%**

Fonte: Carvalho (2018)

É possível observar através da figura 20, uma das predições realizadas por Marangoni et al. (2010), é possível observar que o resultado das predições ficou próximo do comportamento real.

Figura 20 – Comparação Previsão x Preço Real PETR4



Fonte: Carvalho (2018)

Marangoni et al. (2010) conclui que, as predições do mercado acionário estão longe de ser simples e de soluções perfeitas, porém é possível obter predições que atingem resultados satisfatórios ao se aplicar técnicas de redes neurais artificiais para analisar as séries temporais destes mercados, quando estes se encontram em um momento de "pouca agitação internacional".

3.5 Comparativo dos trabalhos

Os estudos apresentados individualmente são de grande auxílio para os pesquisadores da área, obtendo resultados majoritariamente positivos em momentos de normalidade do mercado, porém, quando são analisados dados que sofreram interferência de eventos externos, todos autores relatam dificuldades em realizar predições eficientes pelos algoritmos utilizados, gerando predições distantes do esperado. Através da tabela 1 é possível observar um compilado das ferramentas, linguagens e objetos de estudo almejados pelos autores. O presente trabalho utilizou modelos de predição modernos, tais como o *Facebook's Prophet*, o modelo ARIMA e o modelo LSTM, com o objetivo de obter predições mais precisas e comparar o desempenho destas ferramentas.

Tabela 1 – Comparativo dos trabalhos

Trabalho	**Linguagem**	**Ferramenta(s)**	**Objeto(s) de Estudo**
Almeida (2019)	Python	Rede Neural Recorrente	Bitcoin
Garcia (2021)	Python	ARIMA, LSTM e SVR	Bitcoin
Carvalho (2018)	R	MLP e RW	PETR4
Marangoni et al. (2010)	C++	Algoritmo de Backpropagation	PETR4
Presente Trabalho	Python	Prophet, ARIMA e LSTM	Ethereum

Fonte: Autor (2022).

4 MATERIAL E MÉTODOS

Durante a etapa de desenvolvimento foram utilizadas bibliotecas e repositórios de código aberto para realizar a modelagem e a análise dos dados. Todos experimentos utilizam máquinas gratuitas disponibilizadas pelo ambiente do *Google Colab*[19] com as seguintes especificações:

- GPU: Tesla P100-PCIE, que contém 16GB de memória de vídeo e possui uma frequência de núcleo de 1190 MHz.
- CPU: Intel(R) Xeon(R) CPU @ 2.00GHz, que contém 6 núcleos, 12 *threads* e 15 MB de memória *cache*.
- RAM: Aproximadamente 12GB disponíveis
- Disco: Aproximadamente 30 GB disponíveis

Para a realização das predições foram utilizados dados obtidos através da biblioteca *yfinance*[20], 1703 entradas datadas dos dias 09/11/2017 á 07/07/2022 foram extraídas (os dados são extraídos do *yahoo finance*[21]). Para extrair os dados, foi necessário utilizar o trecho de código presente na linha 10.

```
10 df = yf.download('ETH-USD', start="2017-11-09", end= "2022-07-07")
```

Neste caso, foi salvo na variável "df", uma tabela com diversos dados específicos do *Ethereum*, tais como: data, valor de abertura, valor mais alto do dia, valor mais baixo do dia, valor de fechamento e o volume de fechamento. Neste trecho é informado a sigla da moeda que se deseja obter os dados, neste caso, foi utilizado o *ETH* e a sigla da moeda, neste caso foi utilizado o Dólar. As variáveis *start* e *end* são responsáveis por informar para a biblioteca o intervalo de dias que serão extraídos os valores, *start* representa o dia inicial e *end* representa o dia final. Uma variável importante de ser mencionada é a *interval*, esta sendo responsável por portar o intervalo de tempo que se deseja realizar a extração dos dados, por padrão, esta variável recebe "d", representando o período de 24 horas, logo, caso a variável não seja inserida, será aplicado o seu valor padrão na função. Para a realização das predições foram utilizados os valores de fechamento e as datas respectivas de cada fechamento. Para este caso, também seria possível utilizar o valor de abertura, o importante aqui é definir apenas uma das colunas. O intervalo de extração dos dados foi definido como diário, pois os modelos realizam predições nos

[19] https://colab.research.google.com/
[20] https://pypi.org/project/yfinance/
[21] https://finance.yahoo.com/

mesmos intervalos em que leem os dados, logo, caso deseje realizar uma predição de uma hora no futuro, seria necessário extrair os dados com o intervalo de hora a hora. Tendo em vista isto, todos os modelos projetados, possuem como objetivo prever o valor do ETH em um dia, uma semana e um mês no futuro. Todos os códigos presentes nos apêndices representam versões dos *softwares* que realizam predições de uma semana no futuro, para realizar a predição de um dia e um mês são necessários alguns ajustes que serão especificados na seção de resultados.

4.1 Métrica de desempenho adotada

Para avaliar um sistema de modelagem e predição de dados, é necessário adotar um conjunto de métricas que permite avaliar e entender os resultados obtidos, por meio destas, é possível definir qual a eficiência dos modelos desenvolvidos. Com a finalidade de realizar uma comparação justa de todos os modelos, foram utilizadas as mesmas métricas de desempenho para compará-los, estas sendo: a média do Erro percentual absoluto médio (MAPE) e a raiz quadrada do erro médio (RMSE) de diversos períodos iniciais distintos. O MAPE é uma medida que utiliza os valores absolutos para impedir que os erros positivos e negativos cancelem uns aos outros. A equação 1 expõem as variáveis: A que representa o valor real, F que representa o valor da predição realizada e n que representa o número de observações.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A-F}{A}\right| \tag{1}$$

A equação 2 expõem as variáveis da métrica RMSE, esta calcula a raiz do erro médio quadrático da diferença entre o valor real e o valor predito pelo modelo, quanto menor o resultado for, melhor. Sendo Y o valor real, X o valor predito e n o número de predições realizadas.

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(Y-X)^2}{n}} \tag{2}$$

Para realizar estes cálculos, foi adotado um procedimento diferente para cada modelo, ao utilizar o modelo *Prophet*, este através da função *cross_validation* retorna todos os cálculos prontos, não exigindo muito esforço. Por sua vez, o modelo ARIMA e o modelo LSTM requerem a utilização das linhas de código presentes no apêndice D e

E, estas linhas fazem uso da biblioteca *sklearn.metrics*[22], esta sendo uma biblioteca que contém diversas funções para calcular métricas. Para o modelo ARIMA basta enviar o vetor de predições e o vetor de testes e a biblioteca calcula o MAPE e o RMSE de todo o período, pois estes já possuem o formato ideal, porém ao utilizar o modelo LSTM, foi necessário extrair os dados um a um, foi definido manualmente a posição nos vetores, ao gerar os valores, foram extraídos manualmente e inseridos no vetor que foi utilizado para alimentar a função que calcula as métricas.

4.2 *Facebook's Prophet*

Nesta seção, é descrito o passo a passo do funcionamento do algoritmo *Facebook's Prophet*. Os trechos a seguir fazem parte do código que pode ser visualizado no Apêndice A. Após a obtenção dos dados, é necessário manipulá-los, pois o *Prophet* apenas aceita como entrada uma tabela com duas colunas, uma possuindo o nome de "ds", que representa o dia e outra com o nome "y"que representa o valor de fechamento. A manipulação utilizada pode ser verificado no trecho de código abaixo:

```
11 df.reset_index(inplace = True)
12 df = df[['Date', 'Adj Close']]
13 df.columns = ['ds', 'y']
```

A linha 11 é responsável por mover todo conteúdo do *index* da tabela, para uma coluna. Isto é necessário pois ao puxar as informações através da biblioteca, o *index* da tabela possui os dias de cada operação. Ao utilizar o *Prophet* é necessário que os dados estejam presentes nas colunas, não sendo possível utilizar o *index*. Ao executar o comando da linha 11, é possível verificar na tabela 2, a disposição dos dados antes de serem manipulados, já na tabela 3 é possível verificar os dados após a manipulação, o conteúdo dos dias passou a estar presente em uma coluna.

Tabela 2 – Dados antes da manipulação

Date	Open	High	Low	Close	Adj Close	Volume
2017-11-09	308.64	329.45	307.05	320.88	320.88	893249984
2017-11-10	320.67	324.71	294.54	299.25	299.25	885985984
2017-11-11	298.58	319.45	298.19	314.68	314.68	842300992

Fonte: Autor (2022).

[22]https://scikit-learn.org/stable/modules/model_evaluation.html

Tabela 3 – Dados após a manipulação

Index	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-11-09	308.64	329.45	307.05	320.88	320.88	893249984
1	2017-11-10	320.67	324.71	294.54	299.25	299.25	885985984
2	2017-11-11	298.58	319.45	298.19	314.68	314.68	842300992

Fonte: Autor (2022).

Feito isto, é necessário extrair apenas as colunas que serão utilizadas pelo modelo, ou seja, as linhas das colunas data e preço de fechamento (*Date* e *Close* respectivamente), este procedimento é realizado nas linhas 12 e 13, o resultado desta operação pode ser visualizada na tabela 4. Feito isso, os dados estão prontos para serem inseridos no modelo, porém, ainda existe uma série de parâmetros a serem configurados antes da realização das predições.

Tabela 4 – Dados antes da manipulação

Index	Date	Adj Close
0	2017-11-09	320.88
1	2017-11-10	299.25
2	2017-11-11	314.68

Fonte: Autor (2022).

As linhas 15 e 16 definem uma série de parâmetros a serem estabelecidos, estes sendo:

Parâmetros da Linha 15:

- ***daily_seasonality*** : Determina se serão aplicadas variáveis diárias de sazonalidades nas predições;
- ***seasonality_mode*** : Esta opção determina que há alterações nas larguras ou alturas de períodos sazonais ao longo do tempo.

Parâmetros da Linha 16:

- ***name***: Determina qual componente de sazonalidade deve ser aplicado, neste caso foi utilizado semanalmente;
- ***period*** : Período da sazonalidade em dias que deve ser aplicado (neste caso, será aplicado durante toda a semana);
- ***fourier_order*** : Ordem de *fourier* que será aplicada na sazonalidade.

```
15 m = Prophet( daily_seasonality=True, seasonality_mode='multiplicative')
16 m.add_seasonality(name='weekly', period=7, fourier_order=7)
17 model = m.fit(df)
```

Após a configuração destes parâmetros, a variável *model*, que é um objeto do *Prophet*, recebe os dados (da tabela armazenada em df) e as configurações estabelecidas nas linhas 15 e 16 e realiza a execução do método *fit*, que passa estes parâmetros para o construtor deste objeto. Por fim, na linha 19 é realizada a predição do modelo utilizando o *cross_validation*.

```
19 df_cv= cross_validation(model,initial='1500days',period='50days',horizon ='7days')
20 df_performance = performance_metrics(df_cv)
21 df_performance.head(n=50)
22 fig = plot_cross_validation_metric(df_cv, metric='mape')
```

Como parâmetros para realizar a predição, foi necessário enviar para esta função: o modelo com suas devidas configurações e dados inseridos após a execução do método "*fit*", o valor *initial* que representa o dia inicial de predições (neste caso as predições começam a partir do dia 1500), o *period* que determina quantos dias serão adicionados ao *initial* para realizar a próxima predição e o *horizon* que determina o período em que serão realizadas as predições. Após realizar a predição, será salvo na variável "df_cv"os resultados da predição.

Tabela 5 – Resultados das predições

Index	ds	yhat	yhat_lower	yhat_upper	y	cutoff
0	2017-11-09	3650.38	3518.53	3787.81	3270.27	2021-09-10
1	2017-11-10	3674.23	3538.56	3803.55	3410.13	2021-09-10
2	2017-11-11	3660.87	3530.83	3794.50	3285.51	2021-09-10

Fonte: Autor (2022).

Ao analisar a tabela 5, é possível perceber que esta é composta pelas seguintes colunas:

- ***ds***: Determina a data final da predição;
- ***yhat***: É aproximadamente a média da parte inferior somado a parte superior do intervalo de predição (este é o valor da predição do modelo);
- ***yhat_lower***: Parte inferior do intervalo de predição;
- ***yhat_upper***: Parte superior do intervalo de predição;
- ***y***: Valor real do *Ethereum*;
- ***cutoff***: Determina a data inicial da predição;

Após possuir os resultados em mãos, ao executar a linha 20, o método *performance_metrics* calcula as métricas de desempenho para os resultados obtidos, salvando na variável *df_performance* estes dados, uma pequena parcela destes dados pode

ser verificado na tabela 6. A linha 21 determina quantas linhas da tabela serão exibidas na tela.

O *horizon* representa quantos dias no futuro serão realizadas as predições, (*mse, rmse, mae, mape e mdape*) representam parâmetros de desempenho, sendo a métrica *mape* (erro percentual absoluto médio) e a raiz quadrada do erro médio (RMSE) as utilizadas para fins de comparação neste trabalho, *coverage* é a cobertura das estimativas de *y_hat_lower* e *y_hat_upper*.

A linha 22 da figura é responsável por exibir os gráficos das métricas de desempenho previamente descritas. É importante destacar que, os resultados que podem ser visualizados na tabela 6, são a média de todas as múltiplas predições que foram realizadas. Neste caso, o modelo realizou predições durante os seguintes intervalos de sete dias: (1500 à 1506), (1550 à 1556), (1600 à 1606), (1650 à 1656) e (1700 à 1706), calculou as métricas para cada período e por fim realizou a média.

Tabela 6 – Resultados dos cálculos de desempenho

Index	horizon	mse	rmse	mae	mape	mdape	coverage
0	1 days	63524	252.04	219.57	0.0895	0.0961	0.25
1	2 days	49493	222.47	179.56	0.0682	0.0733	0.50
2	3 days	48479	220.18	162.70	0.0594	0.0475	0.50
3	4 days	118143	343.71	290.06	0.1132	0.1138	0.50
4	5 days	161128	401.40	322.96	0.1193	0.1297	0.50
5	6 days	200762	448.06	394.13	0.1517	0.1551	0.25
6	7 days	249713	499.71	425.09	0.1632	0.1889	0.25

Fonte: Autor (2022).

```
58 plt.figure(figsize=(15,9))
59 plt.grid(True)
60 date_range = df_weekly[to_row:].index
61 plt.plot(df_weekly[0: to_row+1]['Adj Close'], 'red')
62 plt.plot(date_range, testing_data, color = 'red', label = 'Preco original do ETH')
63 plt.plot(date_range, model_predictions, color = 'blue', marker = '',
64 linestyle = 'dashed', label = 'Previsao do preco do ETH')
65
66 plt.title('Predicao ETH utilizando o modelo ARIMA')
67 plt.xlabel('Data')
68 plt.ylabel('Preco US$')
69 plt.legend()
70 plt.show()
```

Por fim, o trecho de código acima, é responsável por mostrar a comparação de

dados de teste com os dados preditos pelo modelo, este gráfico pode ser encontrado na seção de resultados.

4.3 ARIMA

Nesta seção, é descrito o passo a passo do funcionamento do modelo ARIMA. Os trechos a seguir fazem parte do código que pode ser visualizado no Apêndice B. A linha 11 é responsável por obter os dados do *Ethereum*, seguindo o mesmo padrão adotado no modelo *Prophet*.

```
11 df = yf.download('ETH-USD', start="2017-11-09", end="2022-07-07")
12 df_weekly = df.resample('w').mean()
13
14 to_row = 214  # 214 x 7 = dia 1500
15 training_data = list(df_weekly[0: to_row]['Adj Close'])
16 testing_data = list(df_weekly[to_row:]['Adj Close'])
```

Na linha 12 é realizado um tratamento importante sobre os dados. O conjunto de dados é convertido de observações diárias para observações semanais, como pode ser verificado na figura 7, pois o modelo ARIMA trabalha de maneira diferente do que o modelo previamente visto. Este modelo sempre olha um número de dias no passado (determinado pelo usuário) e, com base nisso realiza a predição do próximo valor, ou seja, se os dados estivessem dispostos em intervalos de um dia, o modelo consultaria os últimos x dias (definido pelo usuário) e com base nisso, realizaria a predição do próximo dia apenas, diferente do *Prophet* que realiza a predição de *N* dias no futuro (N é um valor definido pelo usuário). Logo, para realizar uma comparação justa com outros modelos (que irão prever os valores de uma semana no futuro), é necessário adotar este procedimento. A linha 14 é responsável por determinar a quantidade de dados que serão utilizados para treino (o restante é utilizado como dados de teste).

Tabela 7 – Conjunto de dados após manipulação

Date	Open	High	Low	Close	Adj Close	Volume
2017-11-12	310.64	323.19	299.57	310.68	310.68	1.05e+09
2017-11-19	329.24	343.07	325.30	336.14	336.1	8.69e+08
2017-11-26	402.49	427.20	393.66	418.63	418.63	1.33e+09
2017-12-03	461.75	485.05	442.56	460.53	460.53	1.50e+09
2017-12-10	455.77	470.77	436.24	452.53	452.53	1.72e+09

Fonte: Autor (2022).

Armazenamos na variável *to_row* o dia inicial em que se deseja realizar as predições, é importante destacar que a base de dados inicialmente possuía 1703 entradas, ao transforma-lá para observações semanais, restaram apenas 243 entradas, pois 1703 dividido por 7 = 243, logo, a fim de definir o dia inicial das observações, é necessário dividi-lo por 7, no caso da linha 14, foi definido o dia 1500 como data inicial de predição, logo 1500 dividido por 7 = 214. Na linha 15 são atribuídos todos os dados da posição 0 até a posição calculada previamente na variável *training_data*. Esta variável será utilizada mais tarde para realizar o treinamento do modelo. Por fim a linha 16 atribui o restante dos valores na variável *testing_data*, que serão utilizado mais tarde, com a finalidade de validar os resultados obtidos. Através do trecho de código abaixo é possível verificar na linha 26, a declaração de um vetor, que mais tarde será utilizado para guardar os valores das predições que forem realizadas. A linha 27 atribui a uma variável o valor correspondente ao tamanho dos dados de treino. Isto serve para indicar quantas predições devem ser realizadas no futuro, a fim de realizar predições durante o mesmo período dos dados de teste.

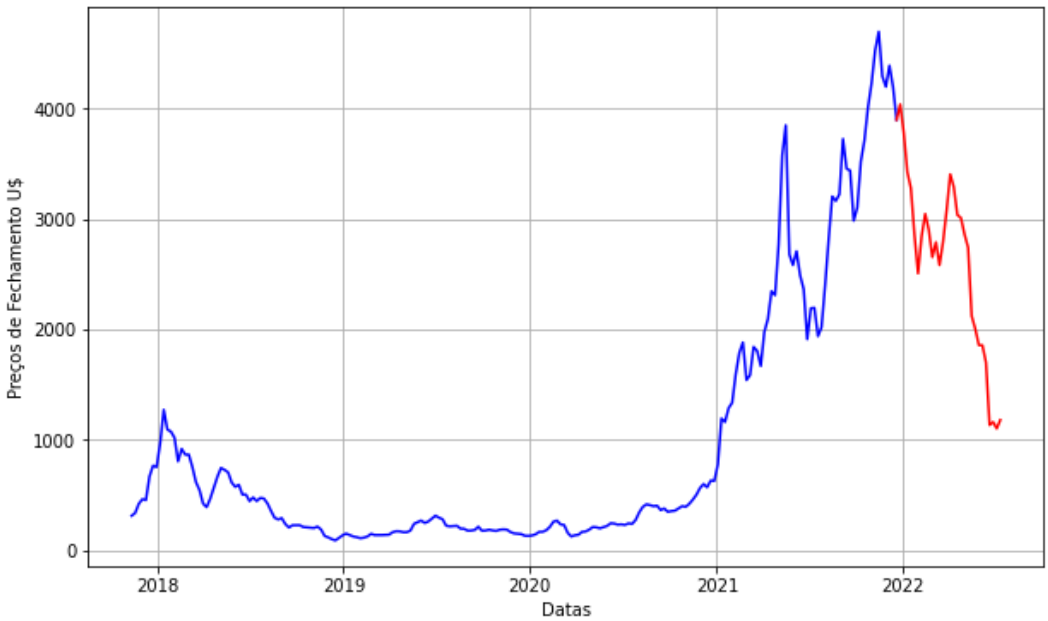
```
18 # Esta parte do codigo plota o grafico da distribuicao dos dados
19 plt.figure(figsize=(10,6))
20 plt.grid(True)
21 plt.xlabel('Datas')
22 plt.ylabel('Precos de Fechamento U$')
23 plt.plot(df_weekly[0: to_row+1]['Adj Close'], 'blue', label= 'Train data')
24 plt.plot(df_weekly[to_row:]['Adj Close'], 'red', label= 'Test data')
25
26 model_predictions = []
27 n_test_obser = len(testing_data)
```

Pode ser verificado na figura 21 a separação de dados realizada, sendo a linha azul os dados de treino, e a linha vermelha os dados de teste, que serão comparados com as predições realizadas. As linhas 19 à 24, são responsáveis por gerar este gráfico.

Através do trecho de código abaixo, é possível verificar o trecho de código responsável por realizar as predições do modelo ARIMA. A linha 29 representa um *loop for* que será executado um número de vezes equivalente ao valor obtido anteriormente pela variável *n_test_obser*, ou seja, durante todo o período em que se encontram os dados de teste. Na linha 30, são passados para o modelo os dados de treino e a variável *order* recebe os parâmetros (p, d, q). Cada parâmetro representa:

- **p**: O número de observações do passado que o modelo utiliza como referência para

Figura 21 – Separação de dados



Fonte: Autor (2022)

a próxima predição;

- **d**: Deve ser zero caso a série seja estacionária, caso contrário deve ser 1 ou 2 para aplicar diferenciações a série, a fim de transforma-lá em estacionária;
- **q**: Determina o valor da média móvel.

```
29 for i in range(n_test_obser):
30   model = ARIMA(training_data, order = (1,1,1))
31   model_fit = model.fit()
32   output = model_fit.forecast()
33   yhat = list(output[0])[0]
34   model_predictions.append(yhat)
35   actual_test_value = testing_data[i]
36   training_data.append(actual_test_value)
```

A linha 31 cria uma instância de um novo objeto ARIMA, em seguida todos os parâmetros previamente selecionados e o conjunto dos dados de treino são passados para o construtor. Em seguida, na linha 32, é realizado uma predição no futuro. É armazenado na variável *yhat* o valor da predição e, por fim, esta variável é adicionada ao vetor previamente declarado (*_model_predictions*). Finalmente, as linhas 35 e 36 são responsáveis por pegar o valor real do *Ethereum* e adicionar ao *training_data*, em outras palavras, nesta etapa são unificados os dados de teste com os dados de treino.

```
38 plt.figure(figsize=(15,9))
```

```
plt.grid(True)
date_range = df_weekly[to_row:].index
plt.plot(df_weekly[0: to_row+1]['Adj Close'], 'red')
plt.plot(date_range, testing_data, color = 'red', label = 'Preco original do ETH')
plt.plot(date_range, model_predictions, color = 'blue', marker = '',
linestyle = 'dashed', label = 'Previsão do preço do ETH')

plt.title('Predicao ETH utilizando o modelo ARIMA')
plt.xlabel('Data')
plt.ylabel('Preço US$')
plt.legend()
plt.show()
```

O trecho de código acima é responsável por imprimir o gráfico que compara os valores reais com os valores preditos pelo modelo, este gráfico pode ser visualizado na seção de resultados.

4.4 Modelo LSTM

Nesta seção, será descrito o passo a passo do funcionamento do modelo LSTM. Os trechos a seguir fazem parte do código que pode ser visualizado no Apêndice C. Na linha 14 do trecho de código abaixo, são obtidos os dados do *Ethereum*, da mesma forma como descrito anteriormente nos modelos *Prophet* e ARIMA.

```
df = yf.download('ETH-USD', start="2017-11-09", end="2022-07-07")

scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(df['Close'].values.reshape(-1,1))

time_intervals_to_train = 7
prediction_interval = 30

x_train = []
y_train = []
```

As linhas 16 e 17 são responsáveis por redimensionar a base de dados, com valores entre 0 e 1, sendo esta uma das exigências que o software LSTM possui para poder ser utilizado. Além disto, é necessário aplicar um reshape(-1,1) pois inicialmente o vetor é unidimensional, ao executar esta função, a entrada passa a ter duas dimensões, que é a entrada esperada pelo modelo LSTM. O resultado pode ser visualizado na figura 22. As

linhas 19 e 20 definem o valor 7 a duas variáveis.

Figura 22 – Redimensionamento de dados

```
array([[0.0500395 ],
       [0.0454642 ],
       [0.04872747],
       ...,
       [0.22214082],
       [0.23323122],
       [0.24393802]])
```

Fonte: Autor (2022)

O trecho de código a baixo mostra a parte responsável por manipular e inicializar as variáveis das linhas 19 e 20 previamente apresentadas. A variável da linha 19 carrega consigo o valor de quantos dias no passado o modelo irá consultar e analisar antes de realizar a predição de *N* dias no futuro. N é definido pelo valor que a variável da linha 20 carrega. Em resumo, este trecho de código configura os intervalos de predição.

```
25 for i in range(time_intervals_to_train, len(scaled_data) - prediction_interval):
26   x_train.append(scaled_data[i - time_intervals_to_train: i, 0])
27   y_train.append(scaled_data[i + prediction_interval, 0])
28
29 x_train = np.array(x_train)
30 y_train = np.array(y_train)
31
32 x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1],1))
```

Já o trecho de código abaixo mostra a definição de diversas camadas ao modelo (linhas 34 à 42), estas camadas são responsáveis por controlar a passagem de informações entre suas etapas. Estas passagens são aplicadas na entrada e na saída da rede, e podem possuir os valores 1 e 0, que indicam quando uma informação deve ser propagada ou não, respectivamente. A função de cada parâmetro esta presente na lista abaixo:

```
34 model = Sequential()
35 model.add(LSTM(128, return_sequences=True, input_shape = (x_train.shape[1],1),
36 activation = 'relu'))
37 model.add(Dropout(0.4))
38 model.add(LSTM(64, return_sequences=True, activation = 'relu'))
39 model.add(Dropout(0.3))
40 model.add(LSTM(32, activation = 'relu'))
41 model.add(Dropout(0.2))
42 model.add(Dense(1, activation = 'sigmoid'))
43
```

```
44 model.compile(loss = 'mean_squared_error', optimizer= 'adam', metrics=['accuracy'])
45
46 model.fit(x_train, y_train, epochs=1000, batch_size= 64)
```

- ***activation='relu'***: Permite que a rede neural aprenda dependências não lineares. Relu retornará a entrada diretamente se o valor for maior que 0, caso contrário retornará 0. A ideia é permitir que a rede aproxime uma função linear quando necessário, com a flexibilidade de também levar em conta a não linearidade;
- ***activation='sigmoid'***: Ao usar uma função de ativação linear em uma rede neural, este modelo poderá apenas aprender problemas linearmente separáveis, porém, com a adição de apenas uma camada oculta e uma função de ativação *sigmoid*, a rede neural poderá aprender uma função não linear;
- ***return_sequences=True***: Este parâmetro é responsável por retornar a saída do estado oculto para cada etapa de tempo de entrada;
- ***Dropout***: Quando este método é utilizado, as conexões de entrada e conexões recorrentes para unidades LSTM são excluídas probabilisticamente da ativação durante o treinamento de uma rede. Este método descarta uma parte das últimas informações utilizadas na camada anterior. É necessário utilizar este parâmetro para que o modelo seja eficiente tanto para teste quanto para treino. Em resumo, caso este método não seja utilizado, são obtidos resultados inferiores de desempenho.
- ***Dense***: Uma camada Densa alimenta todas as saídas da camada anterior para todos os seus neurônios, cada neurônio fornecendo uma saída para a próxima camada. Esta é a camada mais básica em redes neurais, um *Dense(1)* possui 1 neurônio. É necessário utilizar esta camada para guardar o resultado final da execução.

A linha 44 é responsável por determinar a métrica de desempenho a ser utilizada. A linha 46 é responsável por criar a instância de um objeto LSTM, que passa para o construtor todos os parâmetros previamente definidos (camadas, parâmetros, intervalos de predição e aprendizado), também são determinadas quantas *epochs* serão executadas (ciclos de treino da rede neural que serão executados) e o *batch_size* que é um hiper parâmetro de descida de gradiente que controla o número de amostras de treinamento a serem trabalhadas antes que os parâmetros internos do modelo sejam atualizados, o seu valor padrão é 64.

```
47 eth_prices = pd.to_numeric(df['Close'], errors = 'coerce').values
48
49 test_inputs = df['Close'].values
```

```
50 test_inputs = test_inputs.reshape(-1,1)
51 model_inputs = scaler.fit_transform(test_inputs)
52
53 x_test = []
54
55 for x in range(time_intervals_to_train, len(model_inputs)):
56   x_test.append(model_inputs[x - time_intervals_to_train:x, 0])
57
58 x_test = np.array(x_test)
59 x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
60
61 prediction_prices = model.predict(x_test)
62 prediction_prices = scaler.inverse_transform(prediction_prices)
```

Após definir os parâmetros do modelo, bem como os intervalos de predição, e criar a instância do modelo, o trecho de código acima demonstra o redimensionamento dos valores do *Ethereum*, estes são alocados na variável *x_test*. Na linha 61 é realizado a predição dos valores com base em todos os parâmetros que foram definidos anteriormente no modelo. A linha 62 é responsável por reverter o redimensionamento realizado anteriormente, a fim de obter os valores em dólares do *Ethereum*. Por fim o trecho de código abaixo é responsável por imprimir na tela o gráfico que compara os valores preditos com os valores reais do *Ethereum*. Estes gráficos estão presentes na seção de resultados.

```
64 plt.figure(figsize=(15,9))
65 plt.grid(True)
66 plt.plot(eth_prices, label = 'Pre os Ethereum')
67 plt.plot(prediction_prices, label = 'Predi  o realizada')
68 plt.title('Previs o dos pre os do ETH utilizando o modelo LSTM')
69 plt.xlabel('Intervalo de dia a dia')
70 plt.ylabel('Pre o U$S')
71 plt.legend()
```

5 RESULTADOS E DISCUSSÕES

Esta seção tem como objetivo mostrar os resultados das predições diárias e semanais obtidas pelos modelos *Prophet*, ARIMA e LSTM. Também possui como objetivo pontuar algumas alterações de código necessárias para mudar o escopo do modelo. É importante destacar que, os modelos *Prophet* e ARIMA necessitam de apenas uma execução a fim de encontrar o resultado final, pois ambos funcionam através da regressão linear. Já o modelo LSTM necessita de múltiplas execuções tendo em vista que cada simulação gera resultados distintos. Por conta deste modelo possuir um tempo de execução elevado, e a janela de tempo para a realização desta pesquisa ser curta, as métricas de desempenho representam a média de duas execuções. É importante destacar que os resultados obtidos pelos modelos desenvolvidos refletem execuções realizadas apenas com a série temporal do *Ethereum*, estes modelos obtém resultados distintos quando aplicados a diferentes ativos. Foi possível verificar isto através da utilização destes modelos com a finalidade de realizar predições da série temporal do *Bitcoin*.

5.1 Predições de um dia no futuro

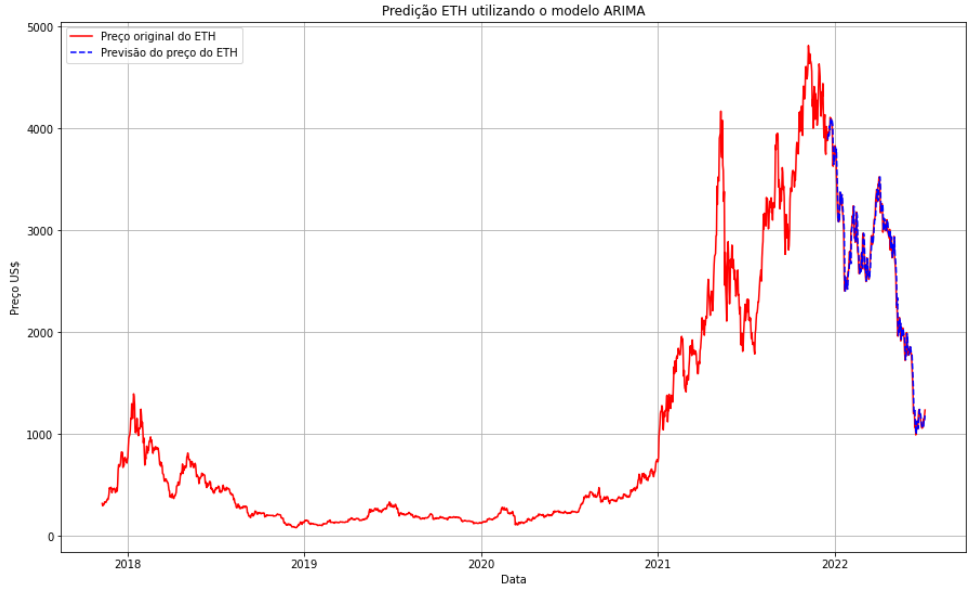
Para realizar a comparação de todos os dados de forma equivalente, foram analisados os mesmos períodos de tempo em todos os modelos. Todos os experimentos foram realizados entre os dias 1500 e 1650. Para a realização destas predições para diferentes períodos de tempo, foi necessário a realização de alguns ajustes em todos os modelos. Para o modelo ARIMA realizar predições de um dia no futuro, basta realizar as alterações que podem ser verificadas no trecho de código abaixo.

```
df = yf.download('ETH-USD', start="2017-11-09", end="2022-07-07")
df_weekly = df
# df_weekly = df.resample('w').mean()

to_row = 1500
training_data = list(df_weekly[0: to_row]['Adj Close'])
testing_data = list(df_weekly[to_row:]['Adj Close'])
```

Em resumo, é comentada a linha de código responsável por transformar o banco de dados de observações diárias para semanais. Após isto basta definir a variável *df_weekly* para receber a variável df, (desta maneira não é necessário alterar o nome de todas

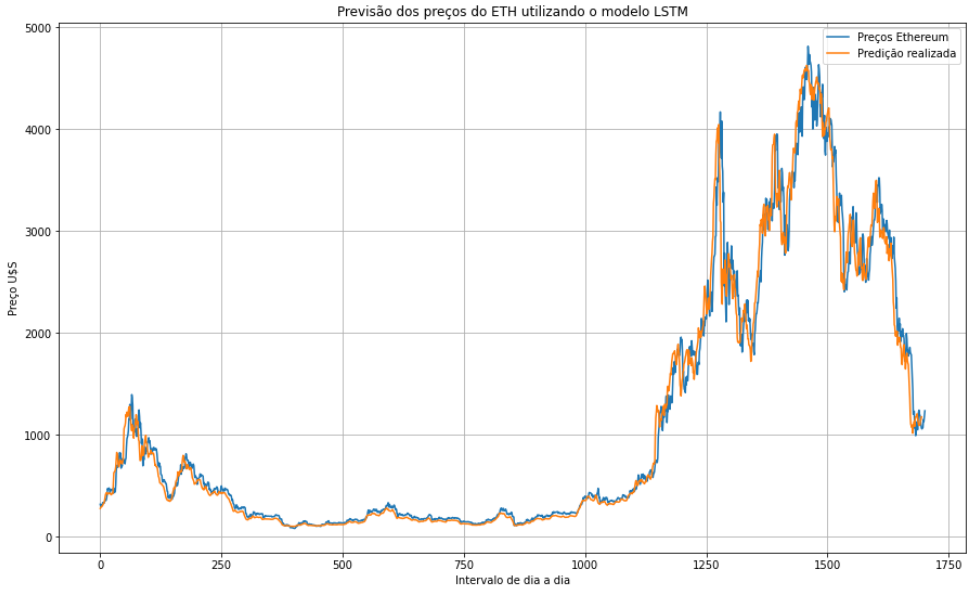
Figura 23 – Predições do modelo ARIMA para 1 dia no futuro



Fonte: Autor (2022)

variáveis no código), por fim basta selecionar o dia inicial das predições (neste caso foi selecionado o dia 1500). O modelo ARIMA obteve um **MAPE** de **3.41%** e um **RMSE** de **109,05**. O gráfico que compara os valores reais com os valores preditos pode ser verificado na figura 23.

Figura 24 – Predições do modelo LSTM para 1 dia no futuro



Fonte: Autor (2022)

Para utilizar o modelo LSTM a fim de realizar predições de 1 dia, basta modificar a variável *prediction_interval* para o número de dias que se deseja prever (neste caso 1)

como pode ser visto no trecho de código abaixo. O modelo LSTM obteve um **MAPE** de **8,00%** e um **RMSE** de **233,98**. O gráfico que compara os valores reais com os valores preditos pode ser verificado na figura 24.

```
time_intervals_to_train = 7
prediction_interval = 30
```

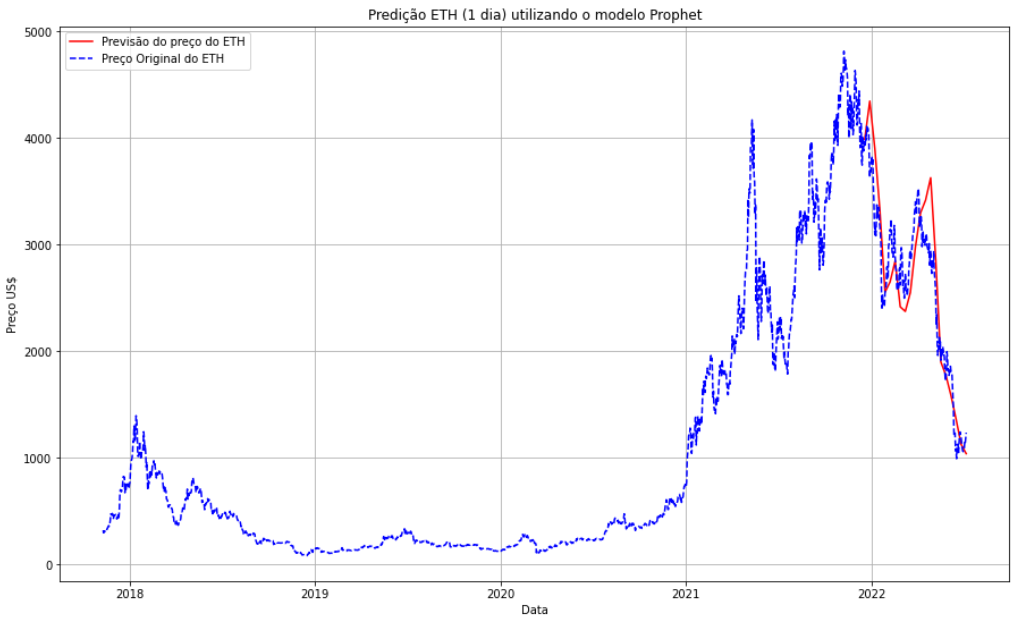
Por fim, para utilizar o modelo *Prophet*, foi necessário adotar as alterações presentes no trecho de código abaixo. Basicamente foi modificado o *period* para dez, isso significa que a cada predição realizada, a próxima começara com mais 10 dias no futuro antes de realizar a próxima predição (neste caso 1500+10), também foi modificado a variável *horizon* para 1, esta variável é responsável por definir em quantos dias no futuro será realizada a predição.

```
m = Prophet( daily_seasonality=True, seasonality_mode='multiplicative')
m.add_seasonality(name='weekly', period=7, fourier_order=7)
model = m.fit(df)

df_cv = cross_validation(model,initial='1500days',period='10days',horizon='1days')

df_performance = performance_metrics(df_cv)
df_performance.head(n=50)
```

Figura 25 – Predições do modelo *Prophet* para 1 dia no futuro



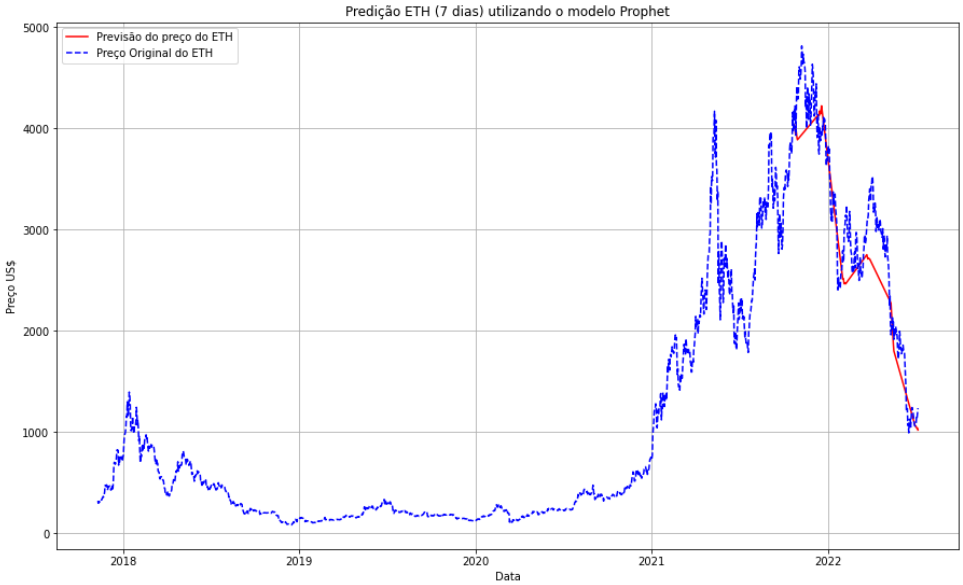
Fonte: Autor (2022)

O *Prophet* obteve um **MAPE** de **10,93%** e um **RMSE** de **283,18**. O gráfico que compara os valores reais com os valores preditos pode ser verificado na figura 25.

5.2 Predições de uma semana no futuro

Para a realização destas predições, novamente foi necessário realizar alguns ajustes em todos os modelos. Para o modelo *Prophet* realizar estas predições, o parâmetro *period* recebe o valor 10, isto determina que a próxima predição começara 10 dias a mais do que o dia inicial da última predição (neste caso 1500+10). Por fim também foi modificada a variável *horizon* para 7, esta variável é responsável por definir quantos dias no futuro será efetuada a predição. As predições realizadas pelo modelo *Prophet* obtiveram um **MAPE** de **16,32%** e um **RMSE** de **499,71**. O gráfico de sua predição pode ser visualizado na figura 26.

Figura 26 – Predições do modelo *Prophet* para 1 semana no futuro

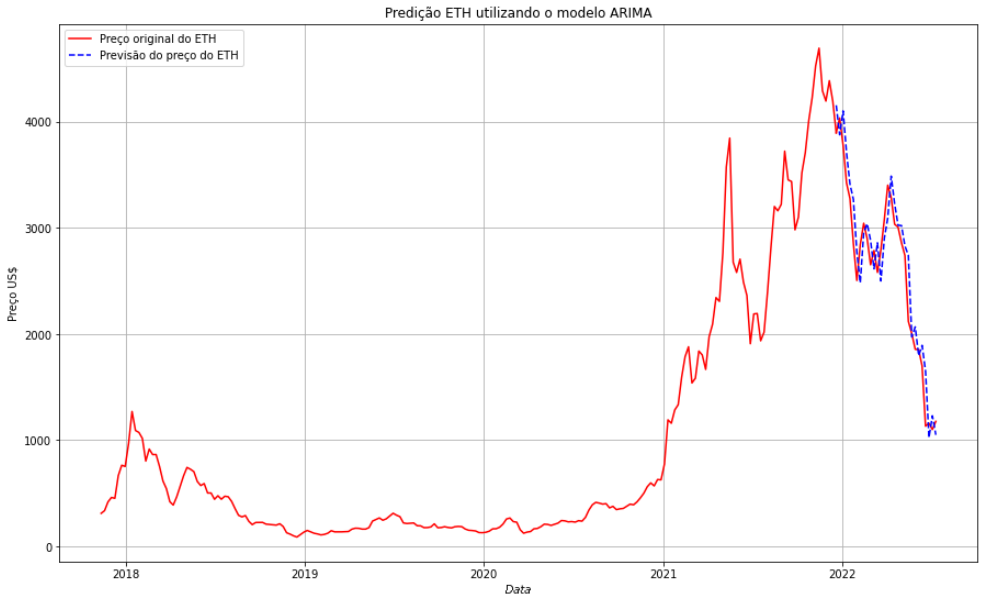


Fonte: Autor (2022)

Para o modelo ARIMA realizar predições de um mês no futuro, basta alterar a disposição dos dados para observações semanais. Em resumo, ao passar como parâmetro a letra 'w' para a função *resample*, a base de dados será modificada para observações semanais do valor do ETH. A variável *to_row* é definida para 50 pois (50x30=1500), este sendo o dia inicial que se deseja realizar as predições. Por sua vez, o modelo ARIMA obteve um desempenho um pouco melhor do que o *Prophet*, este atingiu um **MAPE** de **13,60%** e um **RMSE** de **392,15**. O gráfico das predições pode ser verificado na figura 27.

Por fim, para utilizar o modelo LSTM, basta modificar a variável *prediction_interval* para o número de dias que se deseja prever (neste caso 7). O modelo LSTM obteve um resultado superior em comparação aos outros modelos analisados. Este

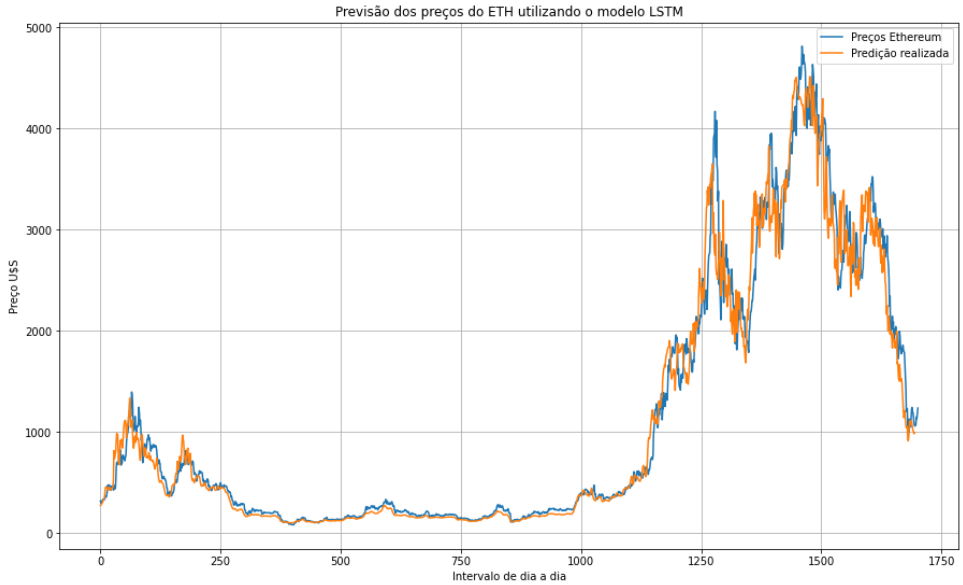
Figura 27 – Predições do modelo ARIMA para 1 semana no futuro



Fonte: Autor (2022)

obteve um **MAPE** de **7,46%** e um **RMSE** de **269,58**. O gráfico da predição pode ser visto na figura 28.

Figura 28 – Predições do modelo LSTM para 1 semana no futuro

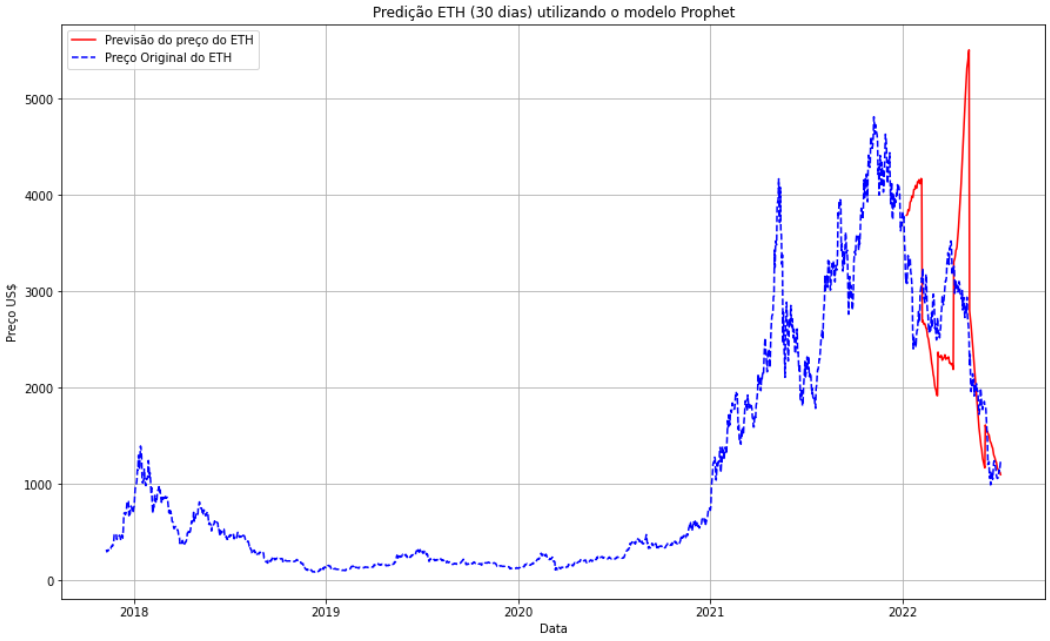


Fonte: Autor (2022)

5.3 Predições de um mês no futuro

Para a realização destas predições, novamente foi necessário realizar alguns ajustes em todos os modelos. Para o modelo *Prophet* realizar estas predições, o parâmetro *period* recebe o valor 30, isto determina que a próxima predição começara 30 dias a mais do que o dia inicial da última predição (neste caso 1500+30). Por fim também foi modificada a variável *horizon* para 30, esta variável é responsável por definir quantos dias no futuro será efetuada a predição. O *Prophet* obteve um **MAPE** de **40,14%** e um **RMSE** de **1359,34**. O gráfico que compara os valores reais com os valores preditos pode ser verificado na figura 29.

Figura 29 – Predições do modelo *Prophet* para 1 mês no futuro

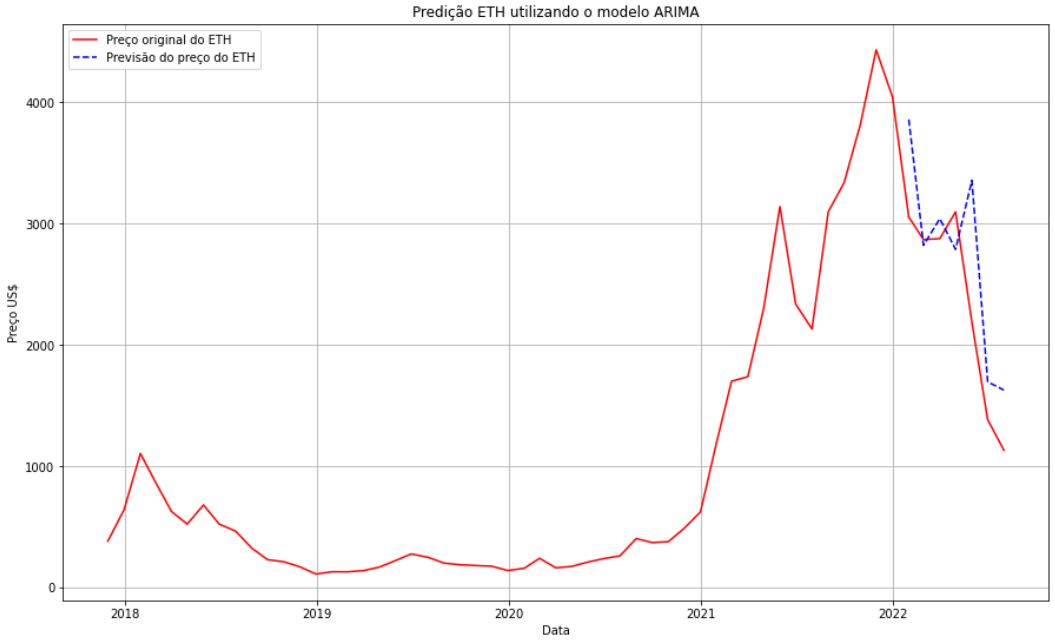


Fonte: Autor (2022)

Para o modelo ARIMA realizar predições de um mês no futuro, basta alterar a disposição dos dados para observações mensais. Em resumo, ao passar como parâmetro a letra 'm' para a função *resample*, a base de dados será modificada para observações mensais do valor do ETH. A variável *to_row* é definida para 50 pois (50x30=1500), este sendo o dia inicial que se deseja realizar as predições. O modelo ARIMA obteve um **MAPE** de **23,37%** e um **RMSE** de **595,77**. O gráfico que compara os valores reais com os valores preditos pode ser verificado na figura 30.

Por fim, para utilizar o modelo LSTM, basta modificar a variável *prediction_interval* para o número de dias que se deseja prever (neste caso 30). O modelo

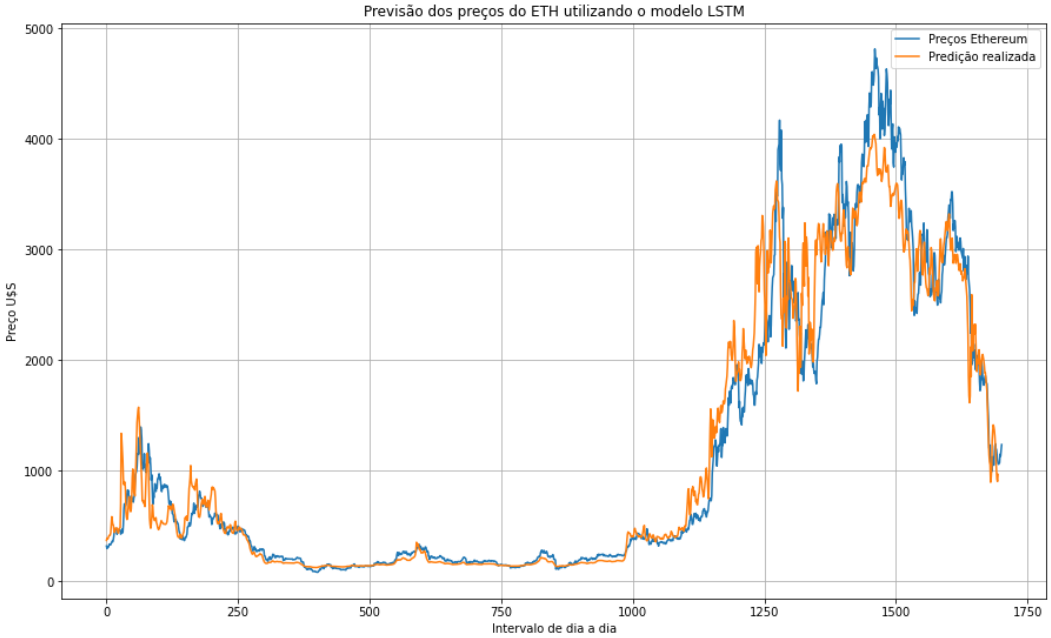
Figura 30 – Predições do modelo ARIMA para 1 mês no futuro



Fonte: Autor (2022)

LSTM obteve um **MAPE** de **17,98%** e um **RMSE** de **517,61**. O gráfico que compara os valores reais com os valores preditos pode ser verificado na figura 31.

Figura 31 – Predições do modelo LSTM para 1 mês no futuro



Fonte: Autor (2022)

5.4 Discussão dos Resultados

Após a realização de todos os experimentos, obteve-se a tabela 8, que possui como objetivo comparar o desempenho de todos os modelos utilizados, a fim de determinar qual ferramenta é mais eficiente em realizar predições sobre os valores futuros do *ETH*.

Tabela 8 – Comparação de desempenho dos modelos

	Predição de 1 dia		Predição de 1 semana		Predição de 1 mês	
Modelo	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE
Facebook's Prophet	10,93%	283,18	16,32%	499,71	40,14%	1359,34
LSTM	8,00%	233,98	7,46%	269,58	23,37%	595,77
ARIMA	3,41%	109,05	13,60%	392,15	17,98%	517,61

Fonte: Autor (2022).

Analisando os dados da tabela 8, é possível concluir que o modelo ARIMA obteve os melhores resultados para predições de um dia no futuro, este obteve um MAPE de (3,41%) e um RMSE de (109,05). Já o modelo LSTM foi o modelo mais eficiente durante os testes realizados para predições de uma semana no futuro, possuindo a melhor média dos percentuais de erro médio (7,46%) e o RMSE mais baixo (269,58). O modelo ARIMA obteve os melhores resultados para predições de um mês no futuro, este obteve um MAPE de (17,98%) e um RMSE de (517,61). Por fim, é possível perceber que todas as ferramentas obtiveram resultados satisfatórios e também foram bem sucedidas ao identificar possíveis tendências de alta e baixa da moeda, como pode ser verificado em seus gráficos de valor real x valor predito, presentes nas figuras das seções acima. Entretanto, também foi possível identificar que em períodos específicos, os modelos tiveram dificuldades durante a realização das predições como é o caso do modelo *Prophet* nas figuras 26 e 29, do modelo LSTM na figura 28 e 31 e do modelo ARIMA na figura 30. É possível observar que todos os modelos perdem precisão conforme o alvo de predição é mais distante.

6 CONCLUSÕES FINAIS E TRABALHOS FUTUROS

Com base na revisão da literatura, é possível concluir que o *machine learning* pode ser uma ferramenta viável para o problema proposto por este trabalho. Isto é, definir o momento para realizar ações no cambio do *Ethereum*. É importante destacar, que a pesquisa aqui apresentada não deve ser tida como algo confiável para fins de investimentos pessoais. Pois, as criptomoedas, diferente de ativos convencionais, sofrem diversas interferências desconhecidas em seu valor, gerando alta volatilidade em seu comportamento. Alguns dos motivos são: especulações em redes sociais e notícias de cunho econômico mundial.

Este trabalho buscou investigar a eficiência dos modelos aplicados à serie temporal do *Ethereum*, possuindo como foco realizar predições de seu valor futuro a curto e médio prazo (1 semana, 1 dia e 1 mês no futuro). A partir deste trabalho, foi possível identificar prováveis tendências que a série demonstrou durante o tempo. Além disso, foi possível avaliar como modelos distintos preveem o seu comportamento e qual a sua precisão.

Como parâmetro de desempenho para este trabalho, quanto menor o erro percentual absoluto médio (MAPE) e quanto menor a raiz quadrada do erro médio (RMSE), maior é a eficiência do modelo utilizado. O modelo que obteve os melhores resultados foi o *LSTM* para predições de 1 semana no futuro. Por outro lado, para a predição de um dia e um mês no futuro, o modelo ARIMA se destacou.

De maneira geral, ao avaliar os resultados obtidos, é possível observar que os modelos propostos foram capazes de realizar predições das tendências de alta e queda do *Ethereum* de maneira eficiente para períodos mais curtos, como sugerem as figuras 23, 24 e 25, por outro lado, ao determinar períodos mais longos de previsão, houve uma redução considerável de precisão, como é possível observar nas figuras 29, 30 e 31. Também foi possível detectar que, durante períodos específicos de tempo houveram taxas de erro discrepantes em relação a média, possivelmente estes momentos específicos estão atrelados à interferências externas, ou a momentos de instabilidade do mercado, gerando volatilidade e perda de precisão nas predições. Da mesma maneira, foi possível observar momentos de alta precisão de predição, este evento possivelmente esta atrelado à momentos de estabilidade do mercado.

É importante destacar que os resultados obtidos para as predições de 1 dia do modelo LSTM possivelmente foram prejudicadas por conta de não ser possível testar todo o conjunto de resultados de forma automática, sendo necessário extrair apenas 10%

das amostras para a realização dos cálculos de desempenho de forma manual, a fim de concluir a pesquisa dentro do período de tempo estabelecido. Isto provavelmente está diretamente relacionado com o fato do modelo LSTM ter sido mais preciso para predições de 1 semana do que para 1 dia, sendo que os outros modelos utilizados sugerem o contrário, logo, é possível concluir que, caso 100% das amostras tivessem sido analisadas, provavelmente a predição de 1 dia no futuro seria mais precisa. Por fim é possível concluir que todos os objetivos estabelecidos por este trabalho foram cumpridos.

6.1 Trabalhos Futuros

No capítulo 6 foi possível concluir que estes modelos são capazes de prever as tendências de alta e baixa da série temporal estudada, alguns aspectos que podem contribuir para trabalhos futuros são:

- Explorar períodos de curto prazo, pois assim é possível evitar períodos de volatilidade do mercado;
- Realizar predições de horas no futuro, a fim de verificar a precisão das ferramentas;
- Utilizar 100% dos resultados do modelo LSTM para realizar os cálculos de desempenho;
- Realizar mais execuções do modelo LSTM.

REFERÊNCIAS

ALMEIDA, C. L. Predição de séries temporais aplicada ao mercado de criptomoedas. 2019.

ALPAYDIN, E. **Machine learning**. [S.l.]: MIT Press, 2021.

ARCE, R. D.; MAHÍA, R. Modelos arima. **Programa CITUS: Técnicas de Variables Financieras**, 2003.

ARRUDA, G. **LSTM para séries temporais**. 2021. Disponível em: <https://gdarruda.github.io/2019/02/03/LSTM-para-series-temporais.html>.

BAPTISTA, S. R. C. **Bitcoin e Blockchain: uma nova classe de ativos**. Tese (Doutorado), 2019.

BOX, G. Box and jenkins: time series analysis, forecasting and control. In: **A Very British Affair**. [S.l.]: Springer, 2013. p. 161–215.

BROCHADO, S. **Por que a previsão dos meteorologistas erra tantas vezes?** 2017. Disponível em: <https://super.abril.com.br/ciencia/por-que-os-meteorologistas-erram-tanto/>.

CAMPOS, P. A. C.; CLEMENTE, A.; CORDEIRO, A. A. L. D. Aplicação do modelo arima para previsão do preço do frango inteiro resfriado no grande atacado do estado de são paulo. In: **Anais do Congresso Brasileiro de Custos-ABC**. [S.l.: s.n.], 2006.

CARVALHO, V. P. Previsão de séries temporais no mercado financeiro de ações com o uso de rede neural artificial. Universidade Presbiteriana Mackenzie, 2018.

COSTA, L. W. M. da. Origem e formação da criptomoeda. **Brazilian Journal of Development**, v. 7, n. 8, p. 85936–85954, 2021.

DUARTE, G. **Redes Neurais | Redes Neurais Recorrentes**. 2019. Disponível em: <https://medium.com/turing-talks/turing-talks-26-modelos-de-predicao-redes-neurais-recorrentes-439198e9ecf3>.

DUTRA, J. K. d. S. Aplicação do modelo arima para previsão de vendas em uma indústria farmacêutica. Niterói, 2019.

EHLERS, R. S. Análise de séries temporais. **Laboratório de Estatística e Geoinformação. Universidade Federal do Paraná**, v. 1, p. 1–118, 2007.

FIGUEREDO, C. J. Previsão de séries temporais utilizando a metodologia box & jenkins e redes neurais para inicialização de planejamento e controle da produção. **Curitiba: UFPR**, 2008.

GARCIA, V. d. S. F. Séries temporais para predição de finanças no contexto de criptomoedas. 2021.

GOMES, D. d. S. Inteligência artificial: conceitos e aplicações. **Olhar Científico. v1**, n. 2, p. 234–246, 2010.

HANNAN, E. J. **Multiple time series**. [S.l.]: John Wiley & Sons, 2009.

IBM. **Recurrent Neural Networks**. 2020. Disponível em: <https://www.ibm.com/cloud/learn/recurrent-neural-networks>.

JONES, T. **Um mergulho profundo nas redes neurais recorrentes**. 2017. Disponível em: <https://imasters.com.br/data/um-mergulho-profundo-nas-redes-neurais-recorrentes>.

KENJI, B. **Machine Learning para Leigos**. 2019. Disponível em: <https://www.venturus.org.br/machine-learning-para-leigos/>.

KORSTANJE, J. **Advanced Forecasting with Python**. [S.l.]: Springer, 2021.

MARANGONI, P. H. et al. Redes neurais artificiais para previsão de séries temporais no mercado acionário. Florianópolis, 2010.

MILUTINOVIĆ, M. et al. Cryptocurrency. **-** , Društvo ekonomista"Ekonomika"Niš, n. 1, p. 105–122, 2018.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas inteligentes-Fundamentos e aplicações**, Manole, v. 1, n. 1, p. 32, 2003.

NEVES, D. V.; DIAS, F. C. A.; CORDEIRO, D. Uso de aprendizado supervisionado para análise de confiabilidade de dados de crowdsourcing sobre posicionamento de ônibus. In: SBC. **Anais do I Workshop Brasileiro de Cidades Inteligentes**. [S.l.], 2018.

PIZZETTI, F. V. A volatilidade das criptomoedas: um estudo com utilização de modelos garch. 2019.

PRODANOV, C. C.; FREITAS, E. C. D. **Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição**. [S.l.]: Editora Feevale, 2013.

RABELO, L. **Princípios básicos para criar previsões de Séries Temporais**. 2019. Disponível em: <https://medium.com/ensina-ai/principios-basicos-para-criar-previsoes-de-series-temporais-e58c451a25b#:~:text=SerieestacionA1ria&text=Umaserietemporalestacionaria,estatatisticasmudamcomotempo.>

RAFFERTY, G. **Forecasting Time Series Data with Facebook Prophet**. [S.l.]: Packt Publishing Ltd, 2021.

RAPOSO, C. M. Redes neuronais na previsão de séries temporais. **Rio de Janeiro: COPPE/UFRJ**, 1992.

SANTOS, D. M. D. Análise do fluxo de exames em um hospital por séries temporais. 2016.

SILVA, G.; RODRIGUES, C. K. d. S. Mineração individual de bitcoins e litecoins no mundo. **Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2016), Niterói, Rio de Janeiro, Brasil**, 2016.

SIMON, J. **Predicting world temperature with time series and DeepAR on Amazon SageMaker**. 2018. Disponível em: <https://julsimon.medium.com/predicting-world-temperature-with-time-series-and-deepar-on-amazon-sagemaker-e371cf94ddb5>.

TEIXEIRA, J. **O que é inteligência artificial**. [S.l.]: E-Galáxia, 2019.

TOTVS. **O que é Inteligência artificial: saiba como funciona e aplicações**. 2019. Disponível em: <https://www.totvs.com/blog/inovacoes/o-que-e-inteligencia-artificial/>.

WEISS, V. A. **Redes Neurais | Redes Neurais Recorrentes**. 2019. Disponível em: <https://medium.com/turing-talks/turing-talks-26-modelos-de-predicao-redes-neurais-recorrentes-439198e9ecf3>.

WEISS, V. A. **Arquitetura de Redes Neurais Artificiais**. 2021. Disponível em: <https://ateliware.com/blog/redes-neurais-artificiais>.

ÁVILA, T. **O que faremos com os 40 trilhões de gigabytes de dados disponíveis em 2020 ?** 2017. Disponível em: <http://governosabertos.com.br/sitev2/o-que-faremos-com-os-40-trilhoes-de-gigabytes-de-dados-disponiveis-em-2020/>.

APÊNDICE A – CÓDIGO DESENVOLVIDO DO FACEBOOK'S PROPHET

Figura 32 – Facebook's Prophet

```
[1] pip install fbprophet]
    pip install yfinance
```

```
import pandas as pd
from prophet import Prophet
import yfinance as yf
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
from prophet.plot import plot_cross_validation_metric
from prophet.plot import add_changepoints_to_plot
import matplotlib.pyplot as plt

df = yf.download('ETH-USD', start="2017-11-09", end="2022-07-07")
df.reset_index(inplace = True)
df = df[['Date', 'Adj Close']]
df.columns = ['ds', 'y']
```

```
[34] m = Prophet( daily_seasonality=True, seasonality_mode='multiplicative')
     m.add_seasonality(name='weekly', period=7, fourier_order=7)
     model = m.fit(df)

     df_cv = cross_validation(model, initial='1400 days', period='50 days', horizon = '7 days')
```

```
[ ] df_performance = performance_metrics(df_cv)
    df_performance.head(n=50)
    # fig = plot_cross_validation_metric(df_cv, metric='mape')
```

```
[36] plt.figure(figsize=(15,9))
     plt.grid(True)

     d_teste = df_cv[['ds', 'yhat']]

     plt.plot(d_teste['ds'], d_teste['yhat'], color = 'red', label = 'Previsão do preço do ETH');
     plt.plot(df['ds'], df['y'], color = 'blue', linestyle = 'dashed', label = 'Preço Original do ETH');
     plt.title('Predição ETH (7 dias) utilizando o modelo Prophet')
     plt.xlabel('Data')
     plt.ylabel('Preço US$')
     plt.legend()
     plt.show()
```

Fonte: Autor (2022)

APÊNDICE B – CÓDIGO DESENVOLVIDO DO MODELO ARIMA

Figura 33 – Parte 1 do Código ARIMA

```
[1] pip install yfinance  #Modelo ARIMA - by: Gabriel R. Correia
```

```
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error, mean_absolute_error
from math import sqrt
from pandas import DataFrame

df = yf.download('ETH-USD', start="2017-11-09", end="2022-07-07")
df_weekly = df.resample('w').mean() # trata o dataset para dados de semana em semana
```

```
[54] to_row = 214  # 214 x 7 = dia 1500
     training_data = list(df_weekly[0: to_row]['Adj Close'])
     testing_data = list(df_weekly[to_row:]['Adj Close'])
```

```
[55] # Esta parte do código plota o gráfico da distribuição dos dados
     plt.figure(figsize=(10,6))
     plt.grid(True)
     plt.xlabel('Datas')
     plt.ylabel('Preços de Fechamento U$')
     plt.plot(df_weekly[0: to_row+1]['Adj Close'], 'blue', label= 'Train data')
     plt.plot(df_weekly[to_row:]['Adj Close'], 'red', label= 'Test data')
```

```
[56] model_predictions = []
     n_test_obser = len(testing_data)
```

```
[57] for i in range(n_test_obser):
       model = ARIMA(training_data, order = (1,1,1))
       model_fit = model.fit()
       output = model_fit.forecast()
       yhat = list(output[0])[0]
       model_predictions.append(yhat)
       actual_test_value = testing_data[i]
       training_data.append(actual_test_value)
```

Fonte: Autor (2022)

Figura 34 – Parte 2 do Código ARIMA

```
[58] plt.figure(figsize=(15,9))
     plt.grid(True)

     date_range = df_weekly[to_row:].index

     plt.plot(df_weekly[0: to_row+1]['Adj Close'], 'red')
     plt.plot(date_range, testing_data, color = 'red', label = 'Preço original do ETH')
     plt.plot(date_range, model_predictions, color = 'blue', marker = '', linestyle = 'dashed', label = 'Previsão do preço do ETH')

     plt.title('Predição ETH utilizando o modelo ARIMA')
     plt.xlabel('Data')
     plt.ylabel('Preço US$')
     plt.legend()
     plt.show()
```

Fonte: Autor (2022)

APÊNDICE C – CÓDIGO DESENVOLVIDO DO MODELO LSTM

Figura 35 – Parte 1 do Código LSTM

```
[13] pip install yfinance  #Modelo LSTM - by: Gabriel R. Correia
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense, Dropout, LSTM
from tensorflow.keras.models import Sequential
from sklearn.preprocessing import MinMaxScaler
import yfinance as yf
from math import sqrt
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error, mean_absolute_error
from math import sqrt
from pandas import DataFrame

df = yf.download('ETH-USD', start="2017-11-09", end="2022-07-07")

scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(df['Close'].values.reshape(-1,1))

time_intervals_to_train = 7
prediction_interval = 7

x_train = []
y_train = []

for i in range(time_intervals_to_train, len(scaled_data) - prediction_interval):
  x_train.append(scaled_data[i - time_intervals_to_train: i, 0])
  y_train.append(scaled_data[i + prediction_interval, 0])

x_train = np.array(x_train)
y_train = np.array(y_train)

x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1],1))

model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape = (x_train.shape[1],1), activation = 'relu'))
model.add(Dropout(0.4))
model.add(LSTM(64, return_sequences=True, activation = 'relu'))
model.add(Dropout(0.3))
```

Fonte: Autor (2022)

Figura 36 – Parte 2 do Código LSTM

```
model.add(LSTM(32, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation = 'sigmoid'))

model.compile(loss = 'mean_squared_error', optimizer= 'adam', metrics=['accuracy'])

model.fit(x_train, y_train, epochs=10, batch_size= 64)

eth_prices = pd.to_numeric(df['Close'], errors = 'coerce').values

test_inputs = df['Close'].values
test_inputs = test_inputs.reshape(-1,1)
model_inputs = scaler.fit_transform(test_inputs)

x_test = []

for x in range(time_intervals_to_train, len(model_inputs)):
  x_test.append(model_inputs[x - time_intervals_to_train:x, 0])

x_test = np.array(x_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

prediction_prices = model.predict(x_test)
prediction_prices = scaler.inverse_transform(prediction_prices)
```

```
[223] plt.figure(figsize=(15,9))
      plt.plot(eth_prices, label = 'Preços Ethereum')
      plt.plot(prediction_prices, label = 'Predição realizada')
      plt.title('Previsão dos preços do ETH utilizando o modelo LSTM')
      plt.xlabel('Intervalo de dia a dia')
      plt.ylabel('Preço U$S')
      plt.legend()
```

Fonte: Autor (2022)

APÊNDICE D – CÓDIGO UTILIZADO PARA CALCULAR O MAPE E O RMSE DO MODELO ARIMA

Figura 37 – Calculo do MAPE e RMSE

```
[22] testing_data

[12] model_predictions

[23] mean_absolute_percentage_error(testing_data, model_predictions)

     0.034151957720952274

[24] mean_squared_error(testing_data, model_predictions, squared=False)

     109.05320854646705
```

Fonte: Autor (2022)

APÊNDICE E – CÓDIGO UTILIZADO PARA CALCULAR O MAPE E O RMSE DO MODELO LSTM

Figura 38 – Calculo do MAPE e RMSE

```
[61] eth_prices[1606]

     3522.83349609375

[62] prediction_prices[1606]

     array([3115.9543], dtype=float32)

     #MAPE

     y_true = [3336.63, 3401.98, 3385.15, 3281.64, 3449.55, 3445.05, 3522.83]
     y_pred = [2912.20, 2896.79, 3168.47, 2869.22, 2998.33, 3020.01, 3115.95]

     mean_absolute_percentage_error(y_true, y_pred)

     0.1192952138703994

     #RMSE

     y_true = [3336.63, 3401.98, 3385.15, 3281.64, 3449.55, 3445.05, 3522.83]
     y_pred = [2912.20, 2896.79, 3168.47, 2869.22, 2998.33, 3020.01, 3115.95]

     mean_squared_error(y_true, y_pred, squared=False)

     414.420087799118
```

Fonte: Autor (2022)