UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

UNIDADE ACADÊMICA DE SERRA TALHADA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

BRUNO LEONARDO DE OLIVEIRA SILVA

Construção de redes neurais para a predição do preço de ativos da Bolsa de Valores

Serra Talhada,
05/2023

Bruno Leonardo de Oliveira Silva

Construção de redes neurais para a predição do preço de ativos da Bolsa de Valores

Trabalho de Conclusão de Curso apresentada ao Curso de Bacharelado em Sistemas de Informação da Unidade Acadêmica de Serra Talhada da Universidade Federal Rural de Pernambuco como requisito parcial à obtenção do grau de Bacharel.

Orientador: Prof. Dr. Sérgio de Sá Leitão Paiva Júnior

Serra Talhada,
05/2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

S586c Silva, Bruno Leonardo de Oliveira
Construção de redes neurais para a predição do preço de ativos da Bolsa de Valores / Bruno Leonardo de Oliveira Silva. - 2023.
37 f. : il.

Orientador: Sergio de Sa Leitao Paiva Junior.
Inclui referências e apêndice(s).

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Sistemas da Informação, Serra Talhada, 2023.

1. Bolsa de valores. 2. Rede neural. 3. Long short-term memory. I. Junior, Sergio de Sa Leitao Paiva, orient. II. Título

CDD 004

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
UNIDADE ACADÊMICA DE SERRA TALHADA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

BRUNO LEONARDO DE OLIVEIRA SILVA

Construção de redes neurais para a predição do preço de ativos da Bolsa de Valores

Trabalho de Conclusão de Curso julgado adequado para obtenção do título de Bacharel em Sistemas de Informação, defendida e aprovada por unanimidade em 27/04/2023 pela banca examinadora.

Banca Examinadora:

Prof. Dr. Sérgio de Sá Leitão Paiva Júnior
Orientador
Universidade Federal Rural de Pernambuco

Prof. Dr. Paulo Mello da Silva
Universidade Federal Rural de Pernambuco

Prof. Dr. Ítalo Cesar de Souza Belo
Universidade Federal Rural de Pernambuco

AGRADECIMENTOS

Em primeiro lugar a minha mãe Maria do Socorro Lima Oliveira, que assim como meu irmão Ramon de Oliveira Silva e também o meu tio Antonio de Oliveira Neto, nunca deixaram de acreditar em mim.

A minha falecida vó Laura Lima de Medeiros, que me mostrou desde cedo o caminho dos estudos como a melhor forma de enriquecer o ser humano.

Ao meu orientador Sérgio de Sá Leitão Paiva Júnior, que foi o responsável por me apresentar o mundo das redes neurais, e também por me apoiar durante as dificuldades encontradas durante a realização desse trabalho.

“É impossível para um homem aprender aquilo
que ele acha que já sabe.”
(Epicteto)

RESUMO

A bolsa de valores se apresenta como uma alternativa de investimento para aqueles que buscam maior rentabilidade, mas em troca desse alta rentabilidade existe uma alta volatilidade, que para muitos investidores acaba resultando em operações negativas. Para contornar esses problemas, os investidores que se arriscam nesse mercado buscam métodos para analisar e até mesmo prever as movimentações de ativos. Um desses métodos é análise técnica que busca prever o movimento do ativo baseado em eventos anteriores. O objetivo desse trabalho é apresentar uma rede neural capaz de prever o preço de ativos da bolsa de valores, para aumentar a consistência em operações positivas. Para tanto, foi desenvolvida uma Rede Neural utilizando os dados de ativos da bolsa de valores, visando o sucesso da operações com a obtenção de resultados mais confiáveis. Os resultados do trabalho deixam claro que o modelo desenvolvido possui um ótimo desempenho, quando comparado com o método de análise técnica. Isso se comprova com os resultados das métricas de avaliação usadas para essa rede neural, como também as simulações de investimento com dados de diferentes ativos da bolsa de valores.

Palavras-chave: Bolsa de valores, Rede Neural, Long short-term memory.

ABSTRACT

The stock market presents itself as an investment alternative for those seeking greater profitability, but in exchange for this high profitability there is high volatility, which for many investors ends up resulting in negative operations. To get around these problems, investors who risk in this market seek methods to analyze and even predict the movements of assets. One of these methods is technical analysis, which seeks to predict the asset's movement based on previous events. The objective of this work is to present a neural network capable of predicting asset prices on the stock market, to increase consistency in positive operations. To this end, a Neural Network was developed using data from stock market assets, aiming at successful operations with more reliable results. The results of the work make it clear that the model developed has a great performance, when compared to the technical analysis method. This is proven with the results of the evaluation metrics used for this neural network, as well as the investment simulations with data from different stock market assets.

Keywords: Stock Market, Neural Network, Long short-term memory.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

RNA	Rede Neural Artificial
RNN	Recurrent Neural Network
RNR	Rede Neural Recorrente
LSTM	Long short-term memory
HME	Hipótese do Mercado Eficiente
CDB	Certificado de Depósito Bancário
MSE	Mean Squared Error
MSLE	Mean Squared Logarithmic Error
MAE	Mean Absolute Error

SUMÁRIO

Construção de redes neurais para a predição do preço de ativos da Bolsa de Valores

Bruno Leonardo de Oliveira Silva[1], **Sérgio de Sá Leitão Paiva Júnior**[1]

[1]Universidade Federal Rural de Pernambuco (UFRPE)
Caixa Postal 063 – 56.900-000 – Serra Talhada – PE – Brasil

brunoleonardo357@gmail.com, sslpaiva@gmail.com

Abstract. *The stock market presents itself as an investment alternative for those seeking greater profitability, but in exchange for this high profitability there is high volatility, which for many investors ends up resulting in negative operations. To get around these problems, investors who risk in this market seek methods to analyze and even predict the movements of assets. One of these methods is technical analysis, which seeks to predict the asset's movement based on previous events. The objective of this work is to present a neural network capable of predicting asset prices on the stock market, to increase consistency in positive operations. To this end, a Neural Network was developed using data from stock market assets, aiming at successful operations with more reliable results. The results of the work make it clear that the model developed has a great performance, when compared to the technical analysis method. This is proven with the results of the evaluation metrics used for this neural network, as well as the investment simulations with data from different stock market assets.*

Resumo. *A bolsa de valores se apresenta como uma alternativa de investimento para aqueles que buscam maior rentabilidade, mas em troca desse alta rentabilidade existe uma alta volatilidade, que para muitos investidores acaba resultando em operações negativas. Para contornar esses problemas, os investidores que se arriscam nesse mercado buscam métodos para analisar e até mesmo prever as movimentações de ativos. Um desses métodos é análise técnica que busca prever o movimento do ativo baseado em eventos anteriores. O objetivo desse trabalho é apresentar uma rede neural capaz de prever o preço de ativos da bolsa de valores, para aumentar a consistência em operações positivas. Para tanto, foi desenvolvida uma Rede Neural utilizando os dados de ativos da bolsa de valores, visando o sucesso da operações com a obtenção de resultados mais confiáveis. Os resultados do trabalho deixam claro que o modelo desenvolvido possui um ótimo desempenho, quando comparado com o método de análise técnica. Isso se comprova com os resultados das métricas de avaliação usadas*

para essa rede neural, como também as simulações de investimento com dados de diferentes ativos da bolsa de valores.

1. Introdução

A bolsa de valores é um mercado que se tornou muito popular no Brasil, como alternativa de investimento a poupança e outras formas de investimento como o Tesouro Direto e Certificados de Depósito Bancário (CDB). Segundo [Gomes 1997], mercado é o local onde são realizadas as transações, as pessoas que as realizam e o conjunto destas transações. Os investidores que buscam obter lucro nesse mercado operam de diferentes formas que podem ser definidas como modalidades, as mais comuns são a Day Trade, Swing Trade e Buy and Hold.

No Day Trade o investidor busca o lucro através de operações realizadas no mesmo dia [GOMES 2018], sendo assim, ele está interessado em fechar todas as suas posições que foram abertas durante o dia. O momento em que ele escolher fechar sua posição vai depender das suas estratégias e alvos daquele dia, muitos operam durante horas outros apenas por minutos. O Swing Trade implica que o investidor deve segurar as suas posições por um tempo maior que um dia [Canto 2020]. Buy and Hold é definido pelo fato de que o investidor busca a valorização desse ativo a longo prazo [GOMES 2018], nesse caso a posição do investidor fica garantida por meses ou até anos, podendo assim também, lucrar com dividendos pagos pela empresa daquele ativo.

Ainda que a bolsa de valores se apresente como forma de renda passiva e alternativa as formas mais conservadoras, temos fatores de risco que se tornam barreiras para pessoas que buscam entrar nesse mercado de investimentos. Quando se fala em dificuldades que oferecem riscos para esses mercados, não é muito difícil lembrar das crises de 2008 e 2020. Também conhecida como crise do Subprime, a crise de 2008 rapidamente se difundiu por todo o mundo, situando-se no âmbito da instabilidade global que adveio do processo de globalização financeira a partir dos anos 80 [Dantas 2020]. A crise de 2020 causou pânico diante das incertezas do Covid-19, o pânico tomou conta do mercado financeiro e da sociedade no geral, o medo da recessão global somado ao contexto da guerra de preços derreteu as bolsas no mundo todo [Dantas 2020].

Além das dificuldades relacionadas crises, ainda existe o fator inconsistência em operações lucrativas, que pode estar relacionado a operações baseadas em Análise técnica. Já que, segundo [Canto 2020], a validade da previsibilidade destas movimentações são questionadas por críticas com base na Hipótese do Mercado Eficiente (HME).

Diante de tudo isso, estudos como o de [Canto 2020] que apresenta avanço na área de Machine Learning com a previsão de preços através da análise de notícias do mercado

financeiro, mostram que existe a possibilidade de usar esse tipo de tecnologia para auxiliar o investidor a lidar com essas dificuldades.

As incertezas de um mercado em crise, com a volatilidade da bolsa que torna operações lucrativas uma tarefa difícil para investidores, levanta o seguinte questionamento: É possível uma rede neural prever os preços de ativos da bolsa de valores? Sendo assim, esse trabalho propõe a criação de uma rede neural capaz de predizer o preço do dia seguinte de ativos da bolsa de valores.

2. Objetivo

O objetivo geral deste trabalho é analisar dados de ativos da bolsa de valores do Brasil através de redes neurais. Para alcançar esse objetivo definiu-se os seguintes objetivos específicos:

- Analisar dados de diferentes ativos da bolsa de valores
- Desenvolver um modelo de rede neural para predição dos valores.
- Validar os resultados de predição de preço de ativos usando redes neurais

3. Referencial Teórico

3.1. Bolsa de Valores

A bolsa de valores é o mercado que empresas utilizam para a captação de recursos externos quando necessitam. Ela é, segundo [Dantas 2020], parte do sistema financeiro, que envolve um amplo conjunto de instituições e organizações que atuam na intermediação entre agentes tomadores de recursos e agentes poupadores. Além disso, as bolsas de valores consistem no ambiente físico e eletrônico de negociação de ações e demais valores mobiliários, constituído de forma organizada por seus integrantes, com o intuito de proporcionar um local adequado e seguro para os investidores e demais partes relacionadas [Silva 2017].

As empresas que buscam recursos nesse mercado, emitem novas ações no “mercado primário”, também conhecido como IPO, e as ofertam para investidores interessados em lucrar com a valorização de preço ou com dividendos, que é uma parte do lucro da empresa repartida entre os acionistas. Aos investidores interessados apenas na valorização da ação foi necessária a criação de um “mercado secundário”, onde os valores mobiliários já emitidos e listados nas bolsas de valores são comercializados livremente entre os investidores por intermédio das corretoras credenciadas e contratadas para tanto [Silva 2017].

Na bolsa de valores as negociações eram feitas em pregões “viva voz”, pelo qual compradores e vendedores, intermediados por corretoras associadas, realizavam as

negociações presencialmente [Silva 2017]. O avanço tecnológico permitiu aos investidores a possibilidade de participar desses pregões, que antes eram feitos apenas presencialmente e por voz, de forma remota e que as negociações sejam feitas com maior agilidade e em maior volume, pois é possível acompanhar diferentes ativos em uma mesma plataforma.

O preço de um ativo na Bolsa de Valores pode ser determinado por diversas razões que podem se relacionar entre si, entre essas, pode-se destacar a lei da oferta e demanda, perspectivas de crescimento da empresa associada ao papel e especulação [Canto 2020]. Diferentes métodos são utilizados por investidores para prever a movimentação desses ativos, e consequentemente lucrar com essa diferença. Um desses métodos é a análise técnica, que busca prever o movimento do ativo baseando-se nos preços anteriores, como também o volume de papeis negociados em determinado período. Esse tipo de técnica pode dar ao investidor falsos sinais, já que as criticas acerca dessa técnica baseiam-se na Hipótese do Mercado Eficiente (HME) [Canto 2020].

A HME se divide em três níveis, a HME fraca, HME semi-forte e HME forte. A mais fraca das hipóteses supõe que os preços do passado já refletem o preço presente. Já para a hipótese semi-forte, o preço atual teve como influencia não só os preços passados como também as informações publicas sobre o ativo. Por fim, a mais forte dessas hipóteses, supõe que o preço atual de um ativo é resultado de todos os conjuntos de informações acerca desse ativo, impossibilitando que até pessoas com informações privilegiadas sejam capazes de realizar lucro nesse mercado [Canto 2020].

Além da possível inconsistência nos lucros que o investidor pode ter, ao operar com análise técnica, existem problemas que afetam o mercado de forma geral, como por exemplo as crises globais. As duas mais conhecidas crises afetaram o mundo no início do século 21, sendo elas a crise de 2008 e 2020.

Na crise de 2008, segundo [Dantas 2020], com a expansão do crédito, e com um histórico de juros baixos no país as pessoas passaram a hipotecar suas casas para investir em mais imóveis, o que gerou uma valorização destes, alimentando ainda mais o mercado imobiliário. Muitas das pessoas que conseguiram os empréstimos para tais investimentos, não tinham formas de quitar essas dividas, levando essas pessoas a hipotecarem os imóveis mais de uma vez, já que a prática era permitida pelos bancos. Assim, a crise do subprime afetou o sistema financeiro mundial com uma reação em cadeia que afetou todas os principais mercados do globo.

A crise de 2020 que causou pânico diante das incertezas do Covid-19, gerou medo de uma recessão, aliado ao contexto de guerra de preços do petróleo entre Rússia e Arábia Saudita, levou a falência diversas empresas, que não tinham caixa suficiente para se sus-

tentar durante o período de paralisação [Dantas 2020].

Momentos críticos como esses aumentam ainda mais as chances de investimentos na bolsa de valores darem errado, pois o grande volume de informações que devem ser analisadas em pouco tempo dificulta a interpretação do mercado atual para o investidor. Em meio a situações como essa uma possível solução está no uso de Redes Neurais para realizar tarefas de análise de séries temporais.

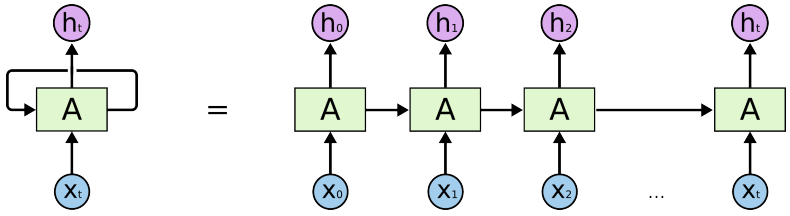
3.2. Redes Neurais

Quando se há uma grande quantidade de dados para serem analisados por seres humanos, o processo pode se torna lento, como também suscetível a falha humana, podendo haver uma má interpretação dos dados, que tornaria um investimento arriscado ao ponto de custar muito caro. Problemas dessa natureza, onde há uma série temporal de dados, podem ser amenizados ou até mesmo resolvidos com o uso de uma rede neural.

Atualmente, existe uma ampla gama de arquiteturas de RNA (Rede Neural Artificial) que são capazes de desempenhar atividades de forma eficiente [Liberal 2021], como por exemplo as Redes Neurais Feedfoward, em que os neurônios são agrupados em camadas (uma camada de entrada, uma de saída e uma ou mais camadas escondidas), e existem apenas conexões para frente. Ela mapeia um conjunto de dados de entrada (instância) num valor de saída apropriado. Utiliza três ou mais camadas de neurônios (nós), com função de ativação não linear, portanto é capaz de distinguir dados separados não linearmente [Rodrigues and Mestria 2015].

As RNR (Rede Neural Recorrente), outro tipo de RNA, podem ser usadas para geração de textos e números, devido a sua capacidade de alimentar a próxima interação do nó com o próprio resultado de saída, permitindo que a rede neural consiga definir um contexto para aquele conjunto de dados.

Figura 1. Funcionamento da Rede Neural Recorrente



Fonte: [Olah 2015]

A Figura 1 mostra que o nó de uma RNR alimenta a sua entrada com o resultado da interação anterior, fazendo com que os dados processados ao longo do tempo seja influenciado pelas interações anteriores.

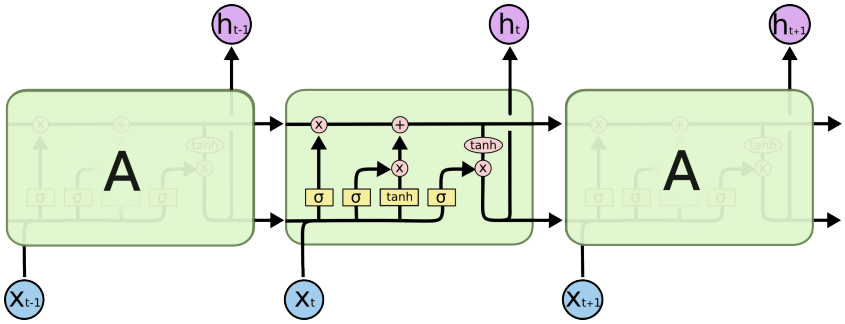
Para que redes neurais tenham um bom desempenho, deve-se configurar os hiperparâmetros. Existem hiperparâmetros adequados para cada tipo de rede, mas os citados a seguir normalmente são utilizados em quase todo tipo de rede. As funções de ativação, que segundo [Reis 2018] são usadas para introduzir a não-linearidade em modelos, o que permite que modelos de aprendizagem profunda aprendam limites de previsão não-lineares. Existem várias Funções de ativação, por exemplo, a Sigmoide é usada na camada de saída enquanto faz previsões binárias. Outros hiperparâmetros usados em redes neurais são o *batch size* e *epochs*, o primeiro é o número de vezes que todos os dados de treinamento são mostrados à rede durante o treinamento, já o segundo, é o número de subamostras dadas à rede após o qual atualização do parâmetro acontece [Reis 2018].

Quando se tem uma grande quantidade de dados a RNR perde a sua capacidade de aprendizado pelas camadas iniciais, devido o problema de Desaparecimento do gradiente [Leite et al. 2021]. Esse problema afeta a rede neural no momento da backpropagation, isto é, quando o valor de erro encontrado é usado para atualizar os pesos da rede neural [Pereira 2017]. A cada interação do backpropagation a célula tem o seu peso diminuído em relação ao anteriormente atualizado, fazendo com que as camadas iniciais da RNR possuam um peso muito baixo, tornando-as irrelevantes no aprendizado da rede neural. Como solução ao problema de desaparecimento de gradiente surge a Rede Neural LSTM (Long Short Term Memory).

3.3. Rede Neural LSTM

Em vez de neurônios, as redes LSTM têm blocos de memória que são conectados por meio de camadas. Um bloco contém portas que gerenciam o estado e a saída do mesmo. A operação se dá em uma sequência de entrada e cada porta dentro de um bloco usa as unidades de ativação sigmoide para controlar se elas são disparadas ou não, tornando a mudança de estado e adição de informações que fluem através do bloco condicional [Leite et al. 2021]. Na (Figura 2) podemos notar a influencia que as portas lógicas têm sobre o resultado de cada interação.

Figura 2. Funcionamento da Rede Neural LSTM



Fonte: [Olah 2015]

4. Método

As seguintes seções descrevem os passos que foram realizados para obtenção e tratamento dos dados antes de serem processados pela rede neural, como também, as ferramentas e processos que foram usados no desenvolvimento da rede neural LSTM.

4.1. Coleta de dados

Os dados necessários para o treino e teste da rede neural foram obtidos do yahoo finance através do uso da biblioteca yfinance. A obtenção foi feita definindo os tickers PETR4.SA, ITUB4.SA e FLRY3.SA que são os códigos referentes aos ativos da Petrobras, Banco Itaú e rede Fleury de laboratórios de análise médica respectivamente, na bolsa de valores. O intervalo de tempo definido foi entre 10/01/2010 e 31/12/2022 em formato "aaaa-mm-dd". Esses dados foram passados como parâmetros da função download da biblioteca yfinance, por fim, a função retornou os dados da bolsa na estrutura de um DataFrame.

4.2. Tratamento dos dados

A primeira etapa consiste em normalizar os dados para valores entre 0 e 1. Esse passo é necessário para melhorar o tempo de treino e adequar os dados ao padrão estabelecido pela RNA [Cunha et al. 2010]. Após a normalização os dados foram divididos em dois conjuntos, que inicialmente foram definidos com 80% dos dados para treinamento da rede e 20% para os testes e validações.

Para a criação dos conjuntos de dados de entrada, saída e validação da RNA uma função foi definida. Nessa função os parâmetros são um conjunto de dados e um valor inteiro chamado de "timeset". O primeiro parâmetro dessa função são os dados que já estão normalizados e divididos, já o segundo, é um valor "t"que serve para indicar que a cada "n" valores o valor de "Y" deverá ser igual a:

$$X_{t+n}$$

. No contexto dessa RNA o valor de "timeset"foi definido como 5, ou seja, a cada 5 preços de fechamento(X) o preço de previsão(Y) deverá ser igual ao valor do próximo dia. Assim temos a seguinte equação:

$$Y_n = X_{t+n}$$

Para finalizar o tratamento dos dados criou-se variáveis para conter os conjuntos de dados que retornam da função "create_dataset", variáveis essas que logo em seguida tiveram os seus formatos modificados pela função "reshape" da biblioteca Numpy para o

formato (5, 1), de valores de input e outuput, possibilitando o uso desse conjunto de dados para o treinamento da rede neural.

4.3. A Rede Neural LSTM

Os modelos de rede neural obtidos foram desenvolvidos utilizando a linguagem Python e a biblioteca Keras, que possibilita a criação de RNAs com um alto grau de customização. A construção desses modelos se deu a partir da definição de uma função que os compilou cada modelo, com base nos parâmetros Taxa de aprendizagem, Otimizador e Função de perda, descritos na tabela 1. A compilação se deu com base em um modelo sequencial com múltiplas camadas do tipo Long Short Term Memory, esse tipo de rede LSTM foi escolhida pois evita o problema de desaparecimento do gradiente que ocorre em redes neurais recorrentes convencionais [Leite et al. 2021].

O parâmetro metrics também foi passado na função compile em forma de lista, contendo o nome de três diferentes métricas de validação, sendo esses parâmetros mean_squared_error, mean_absolute_error, mean_squared_logarithmic_error, que representam o Erro Quadrático Médio, Erro Absoluto Médio e Erro Quadrático Logaritmo Médio, respectivamente. Essas métricas são importantes para a avaliação da rede posteriormente. Dessa maneira a função compile gerou 27 diferentes modelos que foram treinados e testados com os conjuntos de dados especificados anteriormente.

Tabela 1. Parâmetros de treino

Hiperparâmetros			
Taxa de aprendizagem	0,1	0,01	0,001
Otimizador	Adam	SGD	RMSprop
Função de perda	Mean Squared Error	Mean Absolute Error	Mean Squared Logarithmic Error
Batch size	16	32	64
Época	50	100	-

Fonte: Elaborada pelo autor

4.4. Treinando e testando a rede neural

Para o treino utilizou-se a função fit da biblioteca Keras, que recebeu como parâmetros os conjuntos de dados de treino definidos anteriormente, como também os valores de Época e Batch size, mostrados na tabela 1. Além disso, a função recebeu os conjuntos de dados de teste que foram passados como parâmetro de validação de cada modelo.

Os modelos foram testados após o treino com o uso da função predict e os mesmos dados de teste que foram utilizados para validação, para que fosse possível escolher o

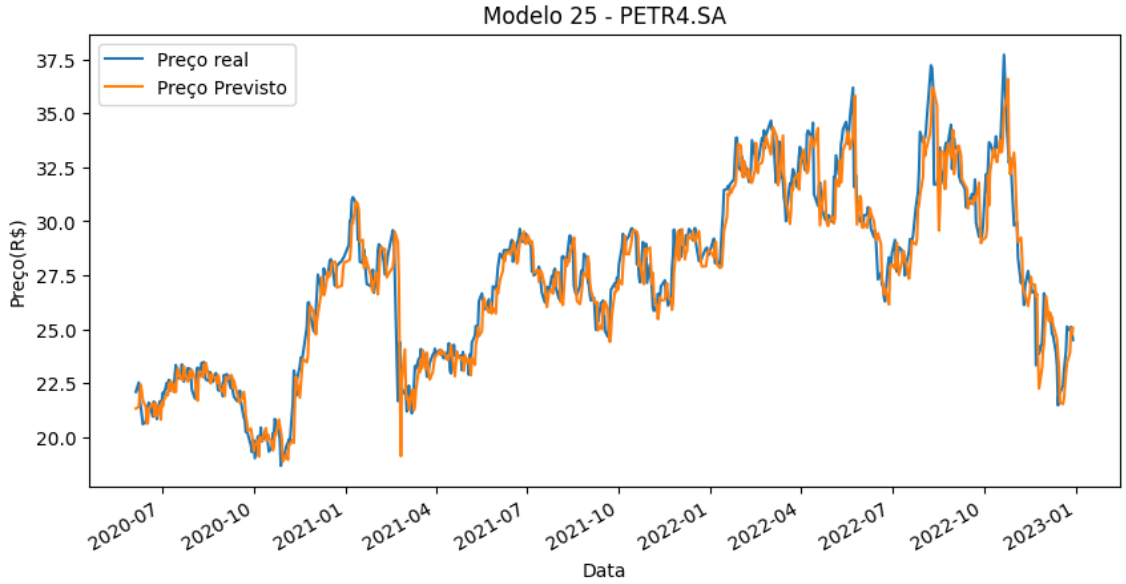
melhor modelo através das métricas definidas durante a fase anterior. O modelo que obteve o melhor desempenho dentre os 27 modelos treinados e testados foi o de número 25, com taxas MAE, MSLE e MSE iguais a 0,0006, 0,0174 e 0,0002 respectivamente, assim como a taxa de perda que foi igual a 0.0002. Quanto menor esses valores melhor para a rede neural, pois isso indica que o valor previsto pela rede está bem próximo ao valor real, na tabela 2 é possível observar as métricas de resultado dos 27 modelos. A figura 3 mostra o resultado dos teste com o modelo 25. Nesse gráfico é possível observar que o valor previsto foi próximo do valor real do ativo.

Tabela 2. Métricas de resultado dos 27 modelos treinados

Métricas dos 27 modelos				
Nº	Taxa de Perda	MAE	MSLE	MSE
1	0,0057	0,0057	0,0661	0,0021
2	0,0009	0,0009	0,0223	0,0003
3	0,2280	0,2280	0,4722	0,1133
4	0,0006	0,0006	0,0169	0,0002
5	0,0012	0,0012	0,0261	0,0004
6	0,0007	0,0007	0,0175	0,0003
7	0,0006	0,0006	0,0169	0,0002
8	0,0013	0,0013	0,0268	0,0004
9	0,0006	0,0006	0,0167	0,0002
10	0,2820	0,0834	0,2820	0,0251
11	0,0428	0,0023	0,0428	0,0008
12	0,0737	0,0077	0,0737	0,0029
13	0,0175	0,0006	0,0175	0,0002
14	0,0225	0,0009	0,0225	0,0003
15	0,0166	0,0006	0,0166	0,0002
16	0,0162	0,0005	0,0162	0,0002
17	0,0260	0,0012	0,0260	0,0004
18	0,0163	0,0006	0,0163	0,0002
19	0,2793	0,5048	0,6998	0,2793
20	0,0004	0,0011	0,0246	0,0004
21	0,2793	201,0992	14,1804	0,2793
22	0,0006	0,0017	0,0203	0,0006
23	0,2793	0,5244	0,7132	0,2793
24	0,2793	1,2337	1,0977	0,2793
25	0,0002	0,0006	0,0174	0,0002
26	0,0004	0,0012	0,0265	0,0004
27	0,0003	0,0008	0,0194	0,0003

Fonte: Elaborada pelo autor

Figura 3. Preço real e previsto do ativo PETR4.SA



Fonte: Elaborada pelo autor

A tabela 3 mostra os valores dos hiperparâmetros que resultaram nesse modelo.

Tabela 3. Parâmetros de treino do modelo 25

Hiperparâmetros - Modelo 25				
Taxa de aprendizagem	Otimizador	Função de perda	Batch size	Época
0,001	Adam	Mean Squared Logarithmic Error	64	100

Fonte: Elaborada pelo autor

5. Resultados

As seções a seguir descrevem os resultados obtidos pelo modelo 25, utilizando novos dados para demonstrar a capacidade de predição da rede neural. Além disso buscou-se também simular um cenário de investimento.

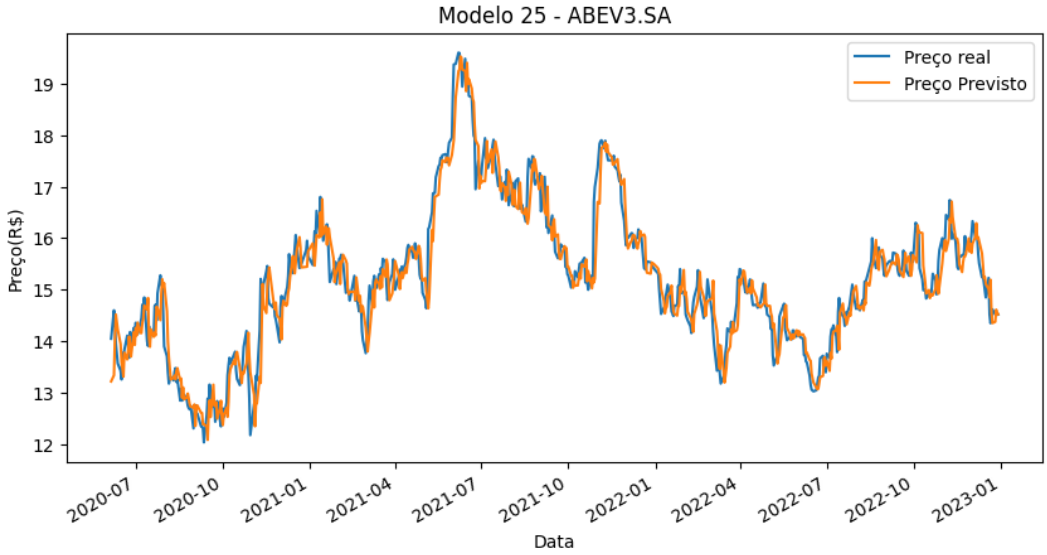
5.1. Predições de preço

Para essa etapa foi necessário a obtenção de novos dados da bolsa. Esse processo foi feito de forma parecida com o descrito anteriormente, para os dados de treino, a diferença aqui foi apenas no ticker, já que esses dados para predições são para os ativos ABEV3.SA e AZUL4.SA, tickers esses que representam a Ambev fabricante de bebidas e a empresa aérea Azul respectivamente. Com esses dados prontos para uso o próximo passo foi

apenas fornecer-los a função predict em forma de parâmetro, e por fim transformar e armazenar esses resultados no seu formato original, para facilitar a interpretação no gráfico.

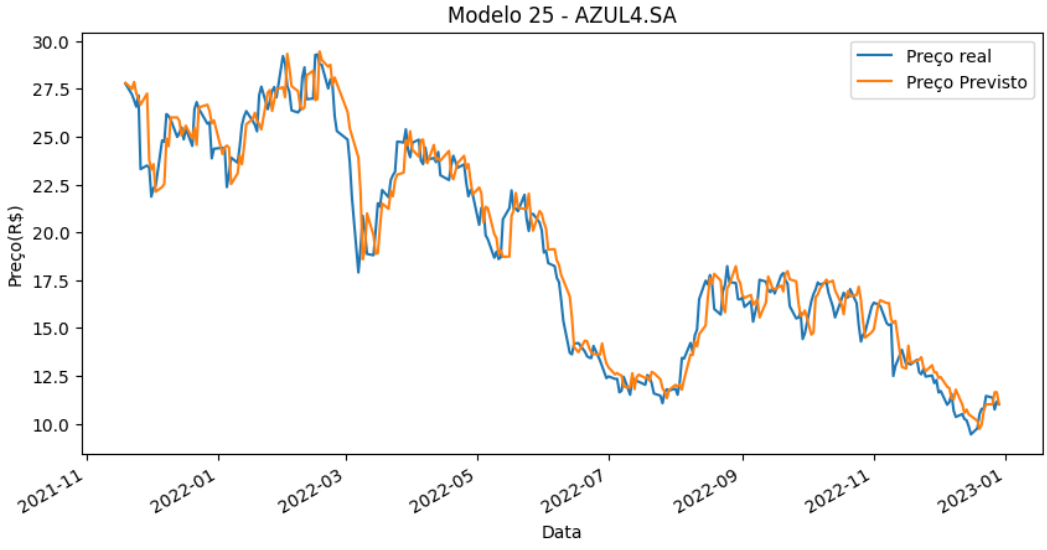
Nas figuras 4 e 5 é possível observar que os valores previstos pela rede continuam bem próximos do real, mesmo que esses dados não tenham sido usados para o treinamento do modelo 25, em outras palavras, pode-se considerar que o modelo é capaz de prever valores próximos ao real. Mesmo que os valores previsto não sejam exatos, é possível utilizar esse modelo para prever o movimento do mercado e consequentemente lucrar com essa variação.

Figura 4. Preço real e previsto do ativo ABEV3.SA



Fonte: Elaborada pelo autor

Figura 5. Preço real e previsto do ativo AZUL4.SA



Fonte: Elaborada pelo autor

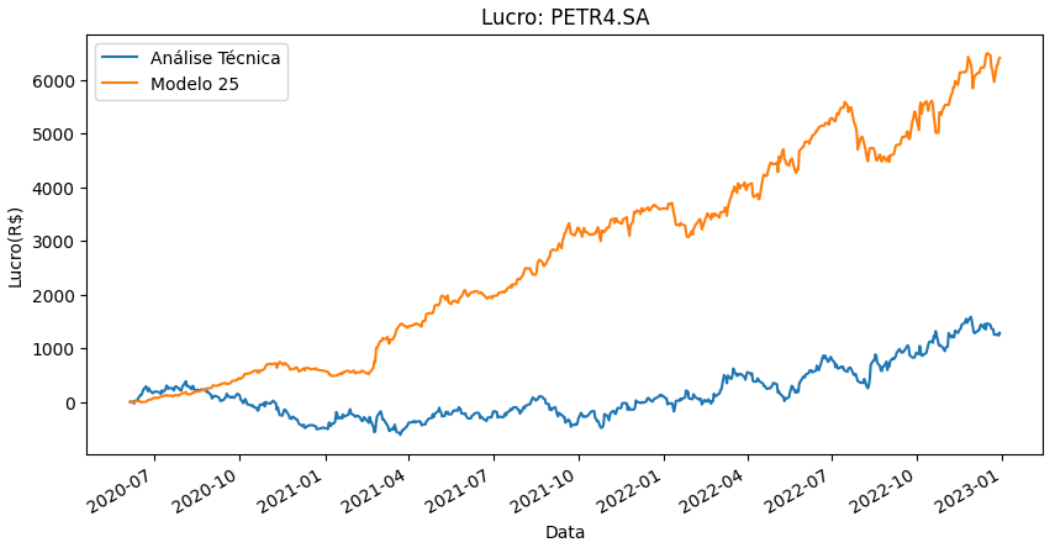
5.2. Simulação de investimento

Para melhor demonstrar a precisão desses modelos, foi definida uma função em python que calcula o lucro obtido nesse período de testes utilizando esses modelos e comparando esses resultados com um método de investimento tradicional, também conhecido como análise técnica, por fim, essa função organiza e apresenta esses resultados em forma de gráfico.

No método com a rede neural simulou-se que os aportes feitos pelo investidor naquele ativo tiveram como entrada o fechamento do dia anterior, ou seja, o investidor aporta o seu dinheiro após o fechamento do pregão anterior e antes do inicio do pregão atual. Nesse caso ele deverá encerrar essa posição quando o ativo atingir o valor previsto ou o valor de fechamento, caso esse alvo esteja mais próximo que o previsto.

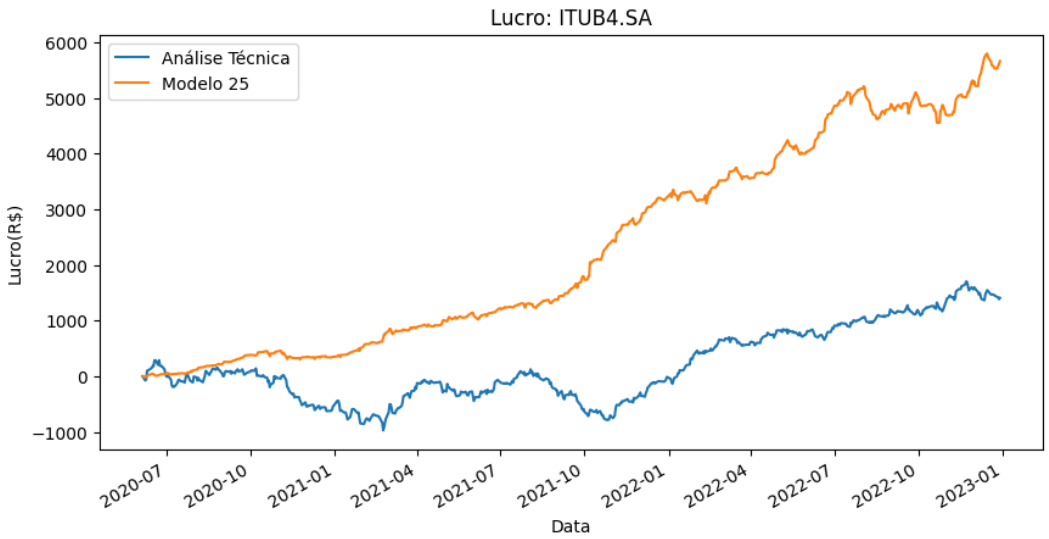
O segundo método simulado por essa função implica que o investidor avalia a direção do ativo no momento de abertura e faz a sua entrada baseando-se nessa informação. Em ambos os métodos considerou-se que o investidor alocou R$ 500,00 reais naquele ativo. Também foi definida para esses métodos uma variável que controla a quantidade de ações investidas a medida que o investidor tem retorno, limitando esse valor em 100 ações para simular um gerenciamento de risco. As figuras 6, 7, 8, 9 e 10 deixam claro que o método de investimento com a rede neural é superior ao método de análise técnica.

Figura 6. Comparação do lucro entre os dois métodos para o ativo PETR4.SA



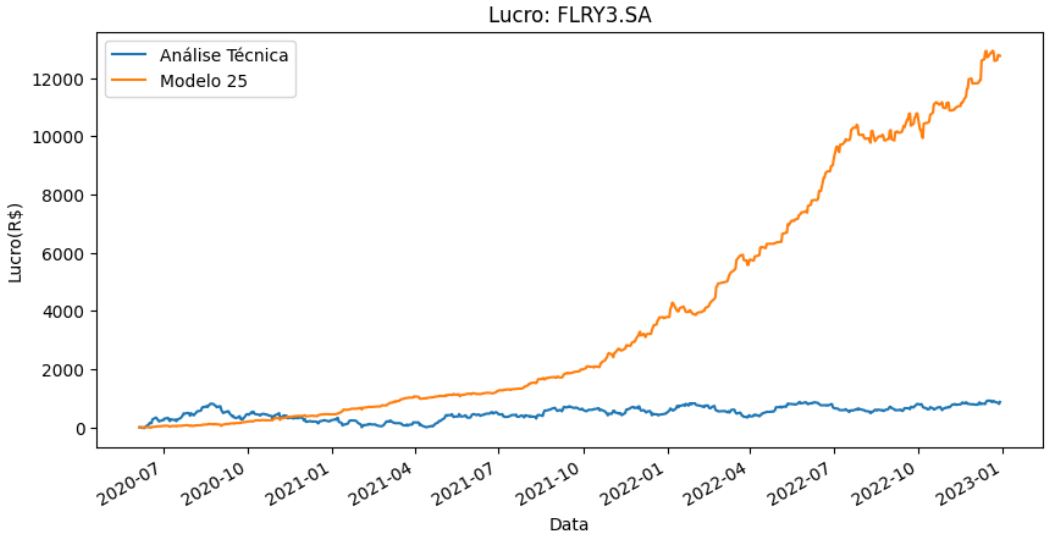
Fonte: Elaborada pelo autor

Figura 7. Comparação do lucro entre os dois métodos para o ativo ITUB4.SA



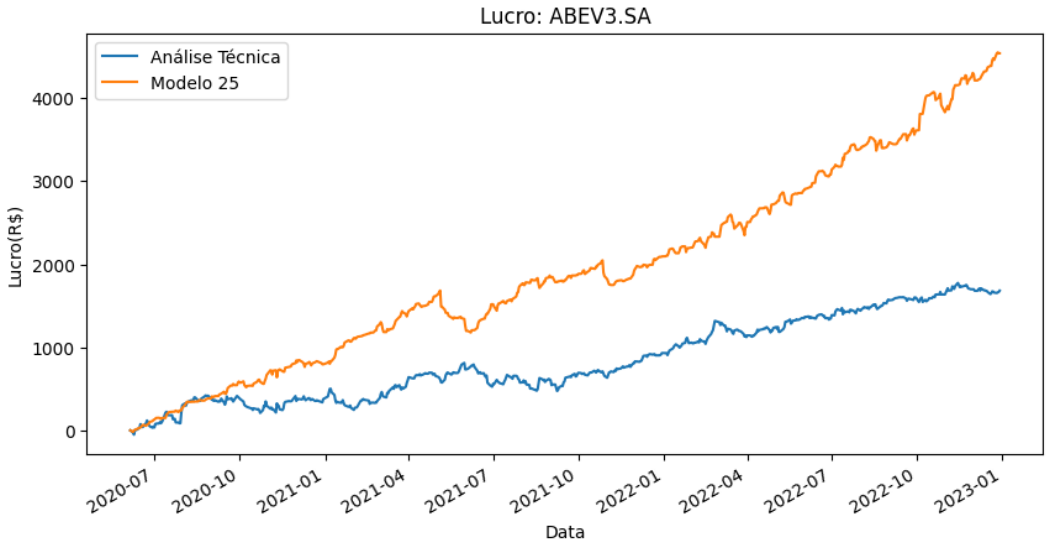
Fonte: Elaborada pelo autor

Figura 8. Comparação do lucro entre os dois métodos para o ativo FLRY3.SA



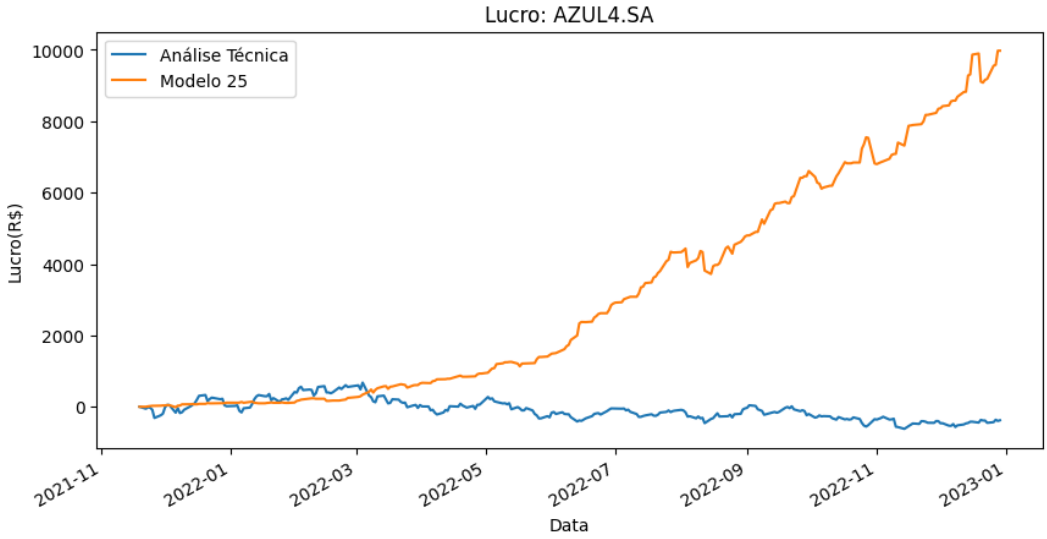
Fonte: Elaborada pelo autor

Figura 9. Comparação do lucro entre os dois métodos para o ativo ABEV3.SA



Fonte: Elaborada pelo autor

Figura 10. Comparação do lucro entre os dois métodos para o ativo AZUL4.SA



Fonte: Elaborada pelo autor

6. Conclusão

Com esse trabalho foi possível concluir que uma rede neural pode ser considerada uma vantagem para investimento na bolsa de valores, quando comparada com o método de análise técnica. A grande quantidade de informações que precisam ser analisadas pelos investidores antes de fazer qualquer aporte nesse mercado, torna o processo de investir na bolsa em uma tarefa complicada, que por muitas vezes leva o investidor a crer em falsos

sinais de entrada. Ao utilizar a rede neural para auxiliar nas análises de ativos, o investidor poderá aumentar a consistência de suas operações lucrativas e amenizar possíveis perdas nesse mercado.

Para trabalhos futuros podem ser sugeridas a possibilidade da criação de uma plataforma que auxilie o investidor utilizando-se das predições de redes neurais, unidos a outros modelos de predição como modelos de séries temporais.

Referências

Canto, L. G. (2020). Análise de notícias do mercado financeiro utilizando processamento de linguagem natural e aprendizado de máquina para decisões de swing trade.

Cunha, G. B. D., Luitgards-Moura, J. F., Lázaro, E., Naves, M., Andrade, A. O., Pereira, A. A., and Milagre, S. T. (2010). A utilização de uma rede neural artificial para previsão da incidência da malária no município de cantá, estado de roraima use of an artificial neural network to predict the incidence of malaria in the city of cantá, state of roraima.

Dantas, M. F. D. M. (2020). Comportamento da bolsa de valores no brasil diante das crises globais de 2008 e 2020.

Gomes, F. R. (1997). A bolsa de valores brasileira como fonte de informações financeiras.

GOMES, I. D. O. (2018). Estratégias para operações de day trade na b3.

Leite, S. J. O., de Oliveira, R. C. L., and de Campos, L. M. L. (2021). Predição de séries temporais da covid19: uma avaliação de redes neurais com células lstm.

Liberal, J. P. G. (2021). Comparação da construção de redes neurais nas linguagens r e python.

Olah, C. (2015). Understanding lstm networks.

Pereira, M. d. M. (2017). Aprendizado profundo: Redes lstm.

Reis, C. H. (2018). Otimização de hiperparâmetros em redes neurais profundas.

Rodrigues, L. M. and Mestria, M. (2015). Metodos de classificação baseado em redes bayesiana e neural para reconhecimento de atividade humana. In Bastos Filho, C. J. A., Pozo, A. R., and Lopes, H. S., editors, *Anais do 12 Congresso Brasileiro de Inteligência Computacional*, pages 1–6, Curitiba, PR. ABRICOM.

Silva, A. R. D. (2017). Aspectos regulatórios da bolsa de valores no brasil.

APÊNDICE A – Código da rede LSTM

rede_bolsa_de_valores

May 4, 2023

[ ]:
```
import yfinance as yf
import numpy as np
import pandas as pd
from keras.models import Sequential, load_model
from keras.layers import LSTM, Dense
from keras.optimizers import Adam, SGD, RMSprop
from sklearn.preprocessing import MinMaxScaler
```

[ ]:
```
# Define o tickers das ações usadas para treinar e testar os modelos
tickers = ["PETR4.SA", "ITUB4.SA", "FLRY3.SA"]
datas = []

# Baixa os preços históricos do Yahoo Finance
for tick in tickers:
    datas.append(yf.download(tick, start="2010-01-01", end="2022-12-31"))
```

[ ]:
```
# Preprocessa os dados
scaler = MinMaxScaler(feature_range=(0, 1))
data_scaled = []

for data in datas:
    data_scaled.append(scaler.fit_transform(data["Close"].values.reshape(-1, 1)))
```

[ ]:
```
# Divide os dados em grupos de treinamento e teste
train_size = []
test_size = []
train_data = []
test_data = []

for i, scaled in enumerate(data_scaled):
    train_size.append(int(len(scaled) * 0.8))
    test_size.append(len(scaled) - train_size[i])
    train_data.append(scaled[0:train_size[i], :])
    test_data.append(scaled[train_size[i]:len(scaled), :])
```

[ ]:
```
# Função para criar os dados de entrada e saída da rede LSTM
def create_dataset(dataset, time_steps=1):
    dataX, dataY = [], []
```

```
    for i in range(len(dataset) - time_steps - 1):
        a = dataset[i:(i + time_steps), 0]
        dataX.append(a)
        dataY.append(dataset[i + time_steps, 0])

    return np.array(dataX), np.array(dataY)
```

[ ]:
```
# Define o tamanho da janela de tempo para os dados de entrada
time_steps = 5
X_train = []
y_train = []
X_test = []
y_test = []

for i, data in enumerate(train_data):
    # Cria os dados de entrada e saída da LSTM com base na janela de tempo␣
 ↪definida
    temp_x, temp_y = create_dataset(data, time_steps)
    X_train.append(temp_x)
    y_train.append(temp_y)

    temp_x, temp_y = create_dataset(test_data[i], time_steps)
    X_test.append(temp_x)
    y_test.append(temp_y)

    # Reformatar os dados de entrada para que sejam compatíveis com a LSTM
    X_train[i] = np.reshape(X_train[i], (X_train[i].shape[0], X_train[i].
 ↪shape[1], 1))
    X_test[i] = np.reshape(X_test[i], (X_test[i].shape[0], X_test[i].shape[1],␣
 ↪1))
```

[ ]:
```
# Define os diferentes modelos a serem treinados

models = []
loss_functions = ["mean_squared_error", "mean_absolute_error",␣
 ↪"mean_squared_logarithmic_error"]
learning_rates = [0.1, 0.01, 0.001]
optimizers = ["adam", "sgd", "rmsprop"]

def get_optimizer(name: str, rate: float):
    match name:
        case 'adam':
            return Adam(learning_rate=rate)
        case 'sgd':
            return SGD(learning_rate=rate)
        case 'rmsprop':
            return RMSprop(learning_rate=rate)
```

```
for loss_func in loss_functions:
    for rate in learning_rates:
        for opt in optimizers:
            model = Sequential()
            model.add(LSTM(units=64, return_sequences=True,␣
 ↪input_shape=(X_train[0].shape[1], 1)))
            model.add(LSTM(units=128))
            model.add(Dense(1))

            model.compile(loss=loss_func, optimizer=get_optimizer(opt, rate),␣
 ↪metrics=['mean_squared_error', 'mean_absolute_error',␣
 ↪"mean_squared_logarithmic_error"])
            models.append(model)
```

[ ]:
```
# Definindo hiperparâmetros
histories = []
epochs = [50, 100]
batch_sizes = [16, 32, 64]

model_configs = []

for i, model in enumerate(models):
        temp_epoch = 0
        temp_batch_size = 0
        for epoch in epochs:
                temp_epoch = epoch
                for size in batch_sizes:
                        temp_batch_size = size
        model_configs.append({'id': i,'batch_size': temp_batch_size, 'epochs':␣
 ↪temp_epoch, 'loss': loss_functions[i//9]})
```

[ ]:
```
# Treinando os modelos
def train_models(xtrain_data, ytrain_data, xtest_data, ytest_data):
    for i, model in enumerate(models):
        for epoch in epochs:
            for size in batch_sizes:
                history = model.fit(xtrain_data, ytrain_data, epochs=epoch,␣
 ↪batch_size=size, validation_data=(xtest_data, ytest_data), verbose=0)
                histories.append(history.history)
        model_configs.append({'id': i,'batch_size': size, 'epochs': epoch,␣
 ↪'loss': loss_functions[i//9]})

for i in range(len(X_train)):
    train_models(X_train[i], y_train[i], X_test[i], y_test[i])
```

[ ]:
```
# Resultado das métricas de treino
for i, history in enumerate(histories):
    print(f"Train loss: {history['loss'][-1]:.4f}, Validation loss:␣
↪{history['val_loss'][-1]:.4f}, Train MAE: {history['mean_absolute_error'][-1]:.
↪4f}, Validation MAE: {history['val_mean_absolute_error'][-1]:.4f}, MSE:␣
↪{history['mean_squared_error'][-1]:.4f}, Validation MSE:␣
↪{history['val_mean_squared_error'][-1]:.4f}")
```

[ ]:
```
# Predições com dados de teste apenas da PETR4.SA
predictions = []

for model in models:
    y_pred = model.predict(X_test[0])
    y_pred = scaler.inverse_transform(y_pred)
    predictions.append(y_pred)
```

[ ]:
```
# Apenas predicoes para PETR4.SA
for i, predicted_price in enumerate(predictions):
    #Cria um dataframe com as datas e preços previstos e reais
    dates = datas[0].index[len(datas[0].index)-len(predicted_price):]

    predicted_df = pd.DataFrame(predicted_price, index=dates,␣
↪columns=["Predicted Price"])
    actual_df = pd.DataFrame(datas[0]["Close"].values[len(datas[0].
↪index)-len(predicted_price):], index=dates, columns=["Actual Price"])
    result_df = pd.concat([actual_df, predicted_df], axis=1)

    #Plota o gráfico com os preços previstos e reais
    result_df.plot(figsize=(10,5),title=f"Model {i+1}: loss␣
↪function={loss_functions[i//9]}, \nTrain loss: {histories[i]['loss'][-1]:.4f},␣
↪Validation loss: {histories[i]['val_loss'][-1]:.4f}, Train MAE:␣
↪{histories[i]['mean_absolute_error'][-1]:.4f}, Validation MAE:␣
↪{histories[i]['val_mean_absolute_error'][-1]:.4f}, MSE:␣
↪{history['mean_squared_error'][-1]:.4f}, Validation MSE:␣
↪{history['val_mean_squared_error'][-1]:.4f}")
```

[ ]:
```
# Remover comentarios para salvar os modelos treinados

# Salvando modelos
# for i, model in enumerate(models):
#     model.save(f"models/model_{i+1}_{loss_functions[i//9]}.h5")
```

1 Usando modelos salvos

[ ]:
```
# Carregando os 27 modelos
saved_models = []
for i in range(27):
    model = load_model(f"models/model_{i+1}_{loss_functions[i//9]}.h5")
    saved_models.append(model)

# Predição com os modelos carregados
predictions = []
for model in saved_models:
    y_pred = model.predict(X_test[0])
    y_pred = scaler.inverse_transform(y_pred)
    predictions.append(y_pred)
```

[ ]:
```
for i, predicted_price in enumerate(predictions):
    # Cria um dataframe com as datas e preços previstos e reais
    dates = datas[0].index[len(datas[0].index)-len(predicted_price):]

    predicted_df = pd.DataFrame(predicted_price, index=dates,␣
 ↪columns=["Predicted Price"])
    actual_df = pd.DataFrame(datas[0]["Close"].values[len(datas[0].
 ↪index)-len(predicted_price):], index=dates, columns=["Actual Price"])
    result_df = pd.concat([actual_df, predicted_df], axis=1)

    # Plota o gráfico com os preços previstos e reais
    result_df.plot(figsize=(10,5),title=f"Model {i+1}: loss␣
 ↪function={loss_functions[i//9]}")
```

[ ]:
```
# Obtendo as metricas de avaliação dos modelos
histories = []
for model in saved_models:
    histories.append(model.evaluate(X_test[0], y_test[0]))
```

[ ]:
```
for i, history in enumerate(histories):
    print(f"MODEL {i+1}: Validation Loss: {history[0]:.4f}, Validation MAE:␣
 ↪{history[1]:.4f}, Validation MSLE: {history[2]:.4f}, Validation MSE:␣
 ↪{history[3]:.4f}")
```

[ ]:
```
# Encontrando o melhor modelo dentre os 27 treinados
log_file = open("models/best_model.txt","w")

best_history = sorted(histories)[0]
best_model_index = 0
for i, h in enumerate(histories):
    if best_history == h:
        log_file.write(f"model_{i+1}_{loss_functions[i//9]}.h5")
```

```
        best_model_index = i

log_file.close()
print(f"MODEL {best_model_index + 1}: Validation Loss: {best_history[0]:.4f},␣
 ↪Validation MAE: {best_history[1]:.4f}, Validation MSLE: {best_history[2]:.4f},␣
 ↪Validation MSE: {best_history[3]:.4f}")
print(f"Best config: {model_configs[best_model_index]}")
print(f"Optimizer: {saved_models[best_model_index].
 ↪get_compile_config()['optimizer']['class_name']}")
print(f"Learning rate: {saved_models[best_model_index].
 ↪get_compile_config()['optimizer']['config']['learning_rate']:.3f}")
```

2 Usando as predições do melhor modelo

[ ]:
```
# Resultados Finais
def show_results(profit_history, dates, title, graph):
    final_profit = []
    for i, profit in enumerate(profit_history):
        if len(final_profit) > 0:
            final_profit.append(final_profit[i-1] + profit)
        else:
            final_profit.append(profit)

    # Plot final profit
    results = pd.DataFrame(final_profit, index=dates, columns=[title])
    if graph:
        results.plot(figsize=(10,5),title=f"Lucro: " + title)
    return results
```

[ ]:
```
# calculando lucro com a rede neural
def calc_model_profit(dataset, predictions, initial_money, graph):
    current_money = initial_money
    quantity = 0

    real_values = dataset["Close"].values[len(dataset.index)-len(predictions):]
    dates = dataset.index[len(dataset.index)-len(predictions):]

    # Ganhos Finais
    profit_history = []
    for i, value in enumerate(real_values):
        final_value = 0
        if i-1 < 0:
            profit_history.append(0)
        else:
            day_before_price = real_values[i-1] # Entrada
            predicted_price = predictions[i].item() # Previsto
```

```
            if day_before_price > predicted_price:
                # Verificando se o preco previsto e menor que o real
                if value > predicted_price and value < day_before_price:
                    final_value = day_before_price - predicted_price
                elif value > predicted_price and value > day_before_price:
                    final_value = day_before_price - value
            else:
                # Verificando se o preco previsto e maior que o real
                if value < predicted_price and value > day_before_price:
                    final_value = value - day_before_price
                else:
                    final_value = predicted_price - day_before_price

            # Define quantidade de ações compradas no dia com o valor de capital
            # Limitando a 100 ações para simular gerenciamento de risco
            if quantity > 100:
                quantity = 100
            else:
                quantity = current_money // day_before_price
            # Multiplica o lucro real pela quantidade de ações compradas no dia

            profit = final_value * quantity
            current_money += profit
            profit_history.append(profit)

    return show_results(profit_history, dates, 'Modelo 25', graph)
```

[ ]:
```
def calc_technical_profit(dataset, predictions, initial_money, graph):
    current_money = initial_money
    quantity = 0

    real_values = dataset[len(dataset.index)-len(predictions):]
    dates = dataset.index[len(dataset.index)-len(predictions):]

    technical_profit_history = []
    for i, open_value in enumerate(real_values["Open"]):
        if i-1 == -1:
            day_before_price = open_value # Entrada
        else:
            day_before_price = real_values["Close"][i-1] # Entrada
            today_close = real_values["Close"][i] # Fechamento hoje

        # Define quantidade de ações compradas no dia com o valor de capital
        # Limitando a 100 ações para simular gerenciamento de risco
        if quantity > 100:
                quantity = current_money // day_before_price
```

```
        else:
            quantity = 100

        final_value = 0
        if open_value < day_before_price:
            final_value = today_close - open_value
        elif open_value > day_before_price:
            final_value = open_value - today_close

        profit = final_value * quantity
        current_money += profit
        technical_profit_history.append(profit)

    return show_results(technical_profit_history, dates, 'Análise Técnica',␣
↪graph)
```

3 Testando e Simulando investimento para 2 novos ativos

[ ]:
```
# Define o ticker da ação usada para treinar e testar os modelos
tickers2 = ['PETR4.SA','ITUB4.SA', 'FLRY3.SA', 'ABEV3.SA', 'AZUL4.SA']
for ticker in tickers2:
    # Baixa os preços históricos do Yahoo Finance
    data = yf.download(ticker, start="2010-01-01", end="2022-12-31")

    # Preprocessa os dados
    scaler = MinMaxScaler(feature_range=(0, 1))
    data_scaled = scaler.fit_transform(data["Close"].values.reshape(-1, 1))

    train_size = int(len(data_scaled) * 0.8)
    test_size = len(data_scaled) - train_size
    train_data = data_scaled[0:train_size, :]
    test_data = data_scaled[train_size:len(data_scaled), :]

    # Define o tamanho da janela de tempo para os dados de entrada
    time_steps = 5

    # Cria os dados de entrada e saída da LSTM com base na janela de tempo␣
↪definida
    X_train, y_train = create_dataset(train_data, time_steps)
    X_test, y_test = create_dataset(test_data, time_steps)

    # Reformatar os dados de entrada para que sejam compatíveis com a LSTM
    X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
    X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

    # Predict
```

```
    y_pred = saved_models[best_model_index].predict(X_test)
    predictions = scaler.inverse_transform(y_pred)

    dates = data.index[len(data.index)-len(predictions):]

    predicted_df = pd.DataFrame(predictions, index=dates, columns=["Preço␣
↪Previsto"])
    actual_df = pd.DataFrame(data["Close"].values[len(data.
↪index)-len(predictions):], index=dates, columns=["Preço real"])
    result_df = pd.concat([actual_df, predicted_df], axis=1)

    # Plota o gráfico com os preços previstos e reais
    result_df.plot(figsize=(10,5),title=f"Modelo {25} - {ticker}",␣
↪xlabel="Data", ylabel="Preço(R$)")

    # Resultado com modelo 25
    model_25_result = calc_model_profit(dataset=data, predictions=predictions,␣
↪initial_money=500, graph=False)
    # Resultado com análise técnica
    technical_result = calc_technical_profit(dataset=data,␣
↪predictions=predictions, initial_money=500, graph=False)

    result = pd.concat([technical_result, model_25_result], axis=1)
    result.plot(figsize=(10,5),title=f"Lucro: {ticker}", xlabel="Data",␣
↪ylabel="Lucro(R$)")
```