Avaliação de modelos preditivos para o mercado de ações utilizando *machine* e *deep learning*

Rafael Bourscheid da Silveira, Carine Geltrudes Webber[1]

[1]Centro de Ciências Exatas e Engenharias – Universidade de Caxias do Sul (UCS)
Caixa Postal 15.064 – 95.070-560 – Caxias do Sul – RS – Brasil

rbsilveira1@ucs.br, cgwebber@ucs.br

Resumo. *Investir no mercado de ações é bastante desafiador, e nos últimos anos houve um aumento no número de investidores no Brasil. O objetivo deste trabalho é realizar um estudo de caso, através da construção e avaliação de modelos preditivos, para a predição de preços de ações do mercado brasileiro. Para isso, foi realizada uma revisão sistemática da literatura a fim de compreender o estado da arte nesse tópico. Observou-se que os métodos com melhores resultados em bolsas de valores extrangeiras são LSTM e Random Forest, e há escassez de material envolvendo a bolsa brasileira. Obteve-se menor MSE e RMSE no experimento com Random Forest. Portanto, é viável o uso de métodos de inteligência aritificial no mercado brasileiro.*

1. Introdução

O número de brasileiros interessados em ampliar seu poder de compra aumentou consideravelmente. Estima-se que em 2021 houve um crescimento de 43% de investidores, sendo que a maior parte foram jovens [Quesada 2021]. Com uma oferta cada vez maior de material gratuito, produtos e serviços financeiros, e motivados por estímulos emocionais, desejo ou ambição, os brasileiros têm entrado cada vez mais no mercado de ações [Medeiros and Leite 2013]. Uma das aplicações dos sistemas preditivos está na previsão de valores de ativos financeiros. Recentemente registra-se um crescimento no interesse por conhecimento e sistemas que auxiliem na tomada de decisão financeira.

Quando uma empresa é registrada na Comissão de Valores Mobiliários (CVM)[1], passa-se a dizer que a mesma é de capital aberto, e ele é dividido em pequenas partes chamadas ações. Com isso, as ações dessa companhia podem ser negociadas publicamente, por intermédio de uma bolsa de valores. O Brasil possui uma única bolsa, a B3[2]. Investir no mercado de ações é bastante desafiador, já que prever o preço de uma ação envolve o trabalho com dados altamente voláteis, não-lineares e dinâmicos. Além disso, os preços sofrem influência de condições políticas e econômicas, tendências de mercado, sazonalidade, fatores psicológicos dos investidores, entre outros [Yadav et al. 2020].

O preço de um ativo, como uma ação ou um índice, flutua ao longo de um determinado período, sendo registrado através de cinco valores: abertura, fechamento, máximo, mínimo, e volume. Essas métricas são utilizadas por investidores para gerar as mais diversas análises, seja isoladamente, em conjunto, ou através da criação de indicadores

[1]Órgão responsável por desenvolver, fiscalizar e regular o mercado de valores mobiliários nacional, também chamado de mercado de capitais.

[2]https://www.b3.com.br/pt_br/

adicionais. De fato, não existe um único método eficaz para evitar prejuízos, apenas algumas técnicas destinadas a torná-los menores. Uma dessas práticas é minimizar a influência do fator emocional nas decisões de compra e venda através do uso de estratégias de negociação

Existem duas principais linhas de raciocínio na hora de se criar uma estratégia de negociação, as análises técnica e fundamentalista. A análise fundamentalista parte da premissa que o valor da ação de uma empresa é determinado por fatores econômicos e pelo desempenho financeiro da própria empresa. A análise técnica, linha seguida neste trabalho, crê que toda a informação utilizável já está embutida no preço da ação, portanto é possível utilizar dados prévios para estimar valores futuros [Elder 2006, Giacomel 2016].

Existem várias aplicações de computação no mercado financeiro, tais como predições no mercado de ações, avaliação de riscos de crédito, alocação de portfólio, precificação de ativos, predição de crises financeiras, entre outros [Mochón et al. 2007]. Ainda, é possível incorporar um modelo preditivo para apoio a tomadas de decisões na área financeira. Uma abordagem frequente nesse contexto é a dos robôs de investimento, também conhecida como *trading* algorítmico (*algorithmic trading*). Consiste em fazer uso de um *software* que consome informações em tempo real sobre o mercado de ações, realiza predições com base em critérios pré-estabelecidos, e executa ordens de compra/-venda em algum sistema de corretora, usualmente por meio de APIs[3].

A área de aprendizado de máquina, ou *Machine Learning (ML)*, é particularmente notável nesse tema, já com várias abordagens conhecidas. Recentemente, as técnicas de *Deep Learning (DL)*, ou aprendizagem profunda, estão em evidência. No contexto financeiro, vários trabalhos buscam construir modelos preditivos a partir de técnicas de *Deep Learning* [Ozbayoglu et al. 2020]. Entretanto, a literatura sobre o uso de tais métodos e a Bolsa de Valores de São Paulo (B3) ainda é escassa. O objetivo deste trabalho é realizar um estudo de caso, através da construção e avaliação de modelos preditivos, para a predição de preços de ações do mercado brasileiro.

A fim de abordar a temática proposta e os resultados obtidos, o presente artigo está organizado em 6 partes. Realizou-se uma revisão sistemática da literatura, descrita na Seção 2. A partir desse processo identificou-se o estado da arte na área, sendo possível planejar e implementar modelos preditivos para o mercado financeiro nacional. A Seção 3 apresenta conceitos e definições importantes ao entendimento do trabalho, e os principais métodos envolvidos. Na sequência, a Seção 4 apresenta o estudo de caso realizado a partir das descobertas, bem como a base de dados empregada e os experimentos realizados. Na sequência, a Seção 5 apresenta os resultados encontrados, bem como discussões acerca dos mesmos. Por fim, a Seção 6 encerra o artigo com as considerações finais, limitações, e contribuições do trabalho.

2. Trabalhos Relacionados

A fim de conhecer o estado-da-arte da área, foi feita uma revisão sistemática da literatura na base de dados *Science Direct*[4]. A chave de busca utilizada foi *("deep learning"AND*

[3]*Application Programming Interface*, ou Interface de Programação de Aplicações, conjunto de padrões estabelecidos para que uma aplicação (o robô) utilize as funcionalidades de outra (o sistema da corretora).

[4]https://www.sciencedirect.com/

stocks AND finance) OR "algorithmic trading". Foram selecionados artigos do tipo *review articles* e *research articles* publicados na área *Computer Science* nos anos 2021, 2020 e 2019. A consulta foi realizada no início de Abril de 2021, e resultou em 88 artigos, dos quais 10 foram selecionados para leitura completa e tabulação. A Tabela 1 a seguir sumariza os trabalhos relacionados.

Tabela 1. Trabalhos relacionados, algoritmos utilizados, e bases de dados

Referência	Algoritmo(s)	Base(s) de dados
[Moghar and Hamiche 2020]	LSTM	*opening price* diário de duas ações listadas na NYSE
[Yadav et al. 2020]	LSTM	*Adjusted closing price* de ações de diferentes setores listadas no índice NIFTY 50
[Vijh et al. 2020]	Artificial Neural Network; Random Forest	Ações de diferentes setores (5 empresas, 10 anos)
[Chalvatzis and Hristu-Varsakelis 2020]	LSTM; Random Forest	Índices S&P500, DIJA, NASDAQ, R2000, de 10 anos.
[Eliasy and Przychodzen 2020]	LSTM	*Adjusted closing price* de 10 empresas de tecnologia listadas no S&P500 de seis anos; *Adjusted price* do S&P500
[Huck 2019]	Deep Belief Networks; Random Forest; Elastic Net Regression	Todas as ações que foram pelo menos uma vez parte do S&P900 de 1990 a 2015
[Khang et al. 2020]	Multilayer Perceptron; Mixed Deep Learning; Linear Regression	Dados diários de 220 empresas representando diferentes setores listados na HOSE
[Lanbouri and Achchab 2020]	LSTM	*S&P500 intraday trading data*: 484 observações entre 11/09/2017 a 16/02/2018
[Long et al. 2020]	Knowledge graph; Graph embeddings techniques; Clustering; Convolutional Neural Network; LSTM: attention mechanism and BiLSTM	Nove milhões de dados de registros de transações de clientes e informações de mercado sobre ações (fonte privada)
[Thakkar and Chaudhari 2020]	LSTM	Dados diários de 10 anos de uma empresa listada na NSE e BSE

A partir dessa análise, percebe-se uma forte prevalência do uso de LSTM e *Random Forest* em trabalhos recentes. A maioria dos *datasets* possuem várias semelhanças entre si: anotações de preços de ações ou índices ao longo de um certo intervalo de tempo. As principais diferenças entre as bases de dados estão na quantidade de ações, a localização das empresas, e a quantidade de registros. Os objetivos dos trabalhos variam bastante, porém todos, em algum momento, realizam alguma forma de predição de preço ou tendência de ações. Alguns trabalhos utilizam essa predição para inferir outras questões, enquanto outros focam em obter a melhor estimativa possível.

Ainda, foi possível perceber que a literatura é escassa no tocante a Bolsa de Va-

lores brasileira, a B3. Nos trabalhos que realizaram comparação entre modelos, LSTM e *Random Forest* obtiveram os melhores resultados. *Artificial Neural Network* também apresentou bons resultados em comparação com outras abordagens.

3. Métodos de *Machine* e *Deep Learning* aplicados aos Sistemas Preditivos

Esta seção trata de modelos de aprendizado de máquina, identificados na revisão sistemática, aplicados na área financeira. Por meio do estudo realizado, diversos métodos foram reconhecidos e dois deles apresentaram os melhores resultados: as redes LSTM e as *Random Forest*. As subseções seguintes descrevem brevemente o funcionamento destes métodos, os mecanismos de avaliação e comparação entre resultados. Para um estudo completo e detalhado sobre eles, indica-se [Haykin 2007] para redes neurais; [Goodfellow et al. 2016] para *deep learning*; [Hochreiter and Schmidhuber 1997] para LSTM; e [Breiman 2001] para as *Random Forest.*

3.1. *Random Forest*

Representar conhecimento extraído de dados através de estruturas simbólicas facilita a interpretação por humanos. Uma árvore de decisão organiza as informações em uma hierarquia de decisões, as quais são refinadas sucessivamente até a obtenção da classificação final. A estratégia utilizada por uma árvore de decisão é dividir e conquistar: um problema complexo é quebrado em questões mais simples, às quais aplica-se a mesma lógica. As várias soluções encontradas dos problemas menores são, então, agrupadas em formato de árvore a fim de obter uma resposta ao desafio inicial [Faceli et al. 2021].

Em decorrência disso, o processo de decisão é bastante compreensível. Em contrapartida, frequentemente a acurácia alcançada por modelos simbólicos seja inferior a modelos "caixa-preta". A fim de contornar isso, é comum agrupar modelos simbólicos para obter resultados melhores [Faceli et al. 2021].

A *Random Forest*, ou *Floresta Aleatória*, é um algoritmo simples e poderoso introduzido por [Breiman 2001], especialmente para problemas de classificação e regressão. É um conjunto de árvores independentes, treinadas separadamente em um subconjunto aleatório das variáveis. Trata-se de um modelo *ensemble* [5], no qual a saída é a combinação das decisões de um conjunto de classificadores, nesse caso as árvores, treinados individualmente [Huck 2019]. O número de árvores na floresta é um equilíbrio entre custo computacional e ganho de performance de aprendizado.

3.2. Redes *Long Short-Term Memory*

Redes neurais são uma família de algoritmos de aprendizagem nos quais redes de unidades funcionais simples e parametrizáveis, chamadas também de neurônios, são interconectadas para realizar um cálculo maior, e onde o processo de aprendizagem envolve treinar simultaneamente os parâmetros de todas as unidades da rede. Os neurônios formam a base para o projeto de redes neurais [Haykin 2007].

Um neurônio é composto por um conjunto de sinapses, cada uma multiplicada por um peso; um somador para os sinais de entrada (combinador linear); e uma função de

[5] Técnica que consiste em agrupar diversos modelos a fim de obter uma performance preditiva maior que os modelos usados individualmente.

ativação para restringir a amplitude de saída, que tipicamente a limita ao intervalo unitário fechado. Também existe um bias, ou viés, com o efeito de modificar a entrada da função de ativação, e é um parâmetro externo do neurônio artificial. A forma que esses neurônios estão ligados depende do algoritmo de aprendizagem que será usado para treinar a rede [Haykin 2007]. Redes neurais no contexto do *Deep Learning* ganham o nome especial de redes neurais profundas, ou, simplesmente, redes profundas. Entre elas, destaca-se as redes neurais convolucionais e as recorrentes.

Uma rede *Long Short-Term Memory* (LSTM), ou Memória Longa de Curto-Prazo, é uma rede neural recorrente. É uma técnica de *deep learning* criada especificamente para análise de dados sequenciais, como uma série temporal. Caracteriza-se por ser capaz de lembrar tanto dados de curto prazo quanto de longo prazo. Destaca-se em aplicações como reconhecimento automático de fala, tradução de linguagens, reconhecimento de caracteres escritos e, claro, previsão de séries temporais [Ozbayoglu et al. 2020].

As redes neurais recorrentes podem usar as conexões de *feedback* para armazenar representações de eventos recentes, simulando assim uma memória de curto prazo. A fase de treinamento das redes costumam usar técnicas baseada em gradientes, as quais frequentemente causam um problema: sinais de erro retrocedendo no tempo tendem a explodir para valores muito altos ou sumir. A evolução temporal do erro retropropagado depende exponencialmente do tamanho dos pesos [Olah 2015, Goodfellow et al. 2016].

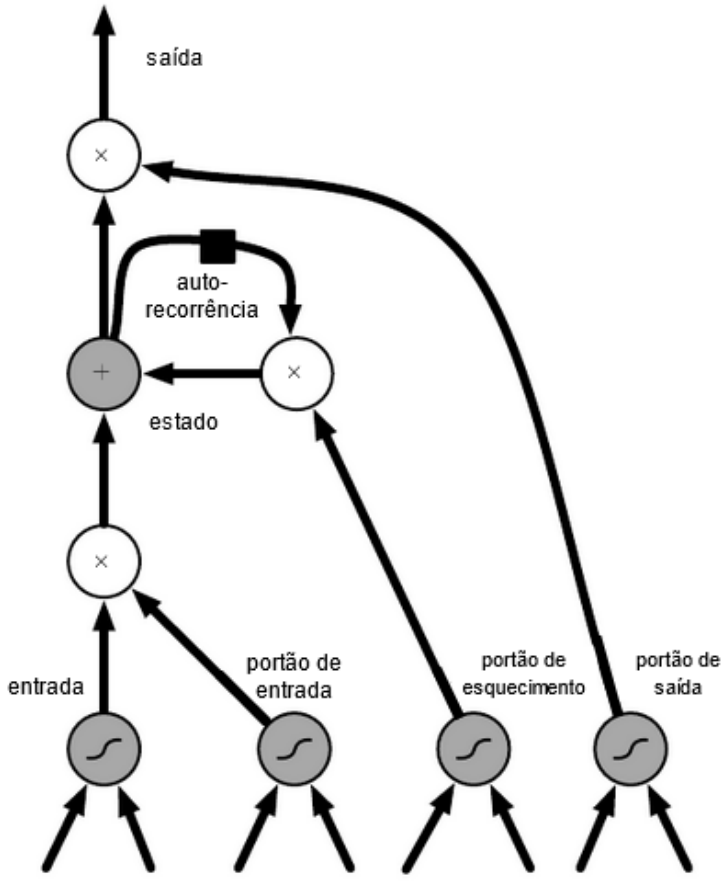

Figura 1. Arquitetura de uma célula de memória da LSTM [Hochreiter and Schmidhuber 1997].

A rede LSTM foi inicialmente proposta por [Hochreiter and Schmidhuber 1997] a fim de superar esses problemas ocasionados por erros retropropagados, sem perder as capacidades de curto prazo. O modelo, como originalmente descrito, tem como peça central uma célula de memória, composta por um estado interno e três portões: entrada (*input gate*), esquecimento (*forget gate*), e saída (*output gate*). Essa estrutura pode ser visualizada na Figura 1.

As células são conectadas uma a outra de forma recorrente, substituindo as unidades ocultas de redes recorrentes convencionais. Uma característica de entrada (*input feature*) é computada com um neurônio artificial clássico. Seu valor pode ser agregado ao estado caso o portão de entrada permita. A unidade de estado possui uma auto-recorrência cujo peso é controlado pelo portão de esquecimento. A saída da célula pode ser desabilitada pelo portão de saída. O estado interno possui uma auto-recorrência a fim de prover *feedback* com atraso de uma unidade de tempo. Uma descrição mais detalhada do funcionamento das redes LSTM pode ser encontrada em [Goodfellow et al. 2016].

Ainda, algumas implementações oferecem suporte a uma técnica de regularização para evitar *overfitting* chamada *Dropout* (abandono). Essa técnica consiste em ignorar aleatoriamente células (e suas conexões) durante a fase de treino de uma rede. Isso permite que as unidades se co-adaptem excessivamente [Srivastava et al. 2014].

Os modelos mais efetivos para o trabalho com dados sequenciais são as *Gated RNNs*, ou redes neurais recorrentes fechadas, em tradução livre. Essas redes são baseadas na ideia de criar caminhos através do tempo que possuem vetores gradientes que não desaparecem nem explodem. A cada ponto no tempo, as *Gated RNNs* podem variar os pesos atribuídos às suas conexões. As redes LSTM seguem essa lógica [Goodfellow et al. 2016]. Por fim, a LSTM mostra-se capaz de aprender dependências de longo prazo com mais facilidades que as arquiteturas mais simples [Goodfellow et al. 2016].

3.3. Critérios de Avaliação

Para avaliar-se um modelo preditivo deve-se utilizar um método de amostragem, tal como *cross-validation*, *hold-out* ou *leave-one-out* [Faceli et al. 2021, Mitchell 1997]. *Cross-validation* é uma família de técnicas voltadas a testar a capacidade de generalização do modelo. Dentre elas, foi utilizado o *K-fold Cross Validation*, o qual consiste em dividir o dataset em K partes e usar cada parte como teste, enquanto as $K - 1$ partes anteriores servem como a base de treino.

O método *hold-out* separa o conjunto de dados em duas amostras, sendo uma usada para treino e a segunda para teste. Recebe como parâmetro o percentual a ser usada em cada partição (exemplo 70%-30%).

O método *leave-one-out* separa uma única instância para teste e usa o restante dos dados para treino. Executa-se este processo para cada instância do conjunto.

Para modelos preditivos desenvolvidos a partir de algoritmos de regressão, onde é esperado que o modelo forneça um valor, as métricas se baseiam no cálculo do erro. A Equação 1 apresenta o cálculo do erro e, sendo ele a diferença entre o valor previsto pelo modelo e o valor real esperado.

$$e = previsto - real \tag{1}$$

O erro médio quadrático (MSE) avalia a média dos quadrados dos erros, e é frequentemente usado para avaliar quão perto os valores estimados estão dos valores reais. Para um problema preditivo, contendo um conjunto de instâncias n, deve-se calcular o somatório do quadrado dos erros e para cada instância, dividindo-se pelo número de instâncias do conjunto n (Equação 2). Quando deseja-se comparar modelos e verificar qual deles se comporta melhor perante os dados, pode-se utilizar o MSE:

$$MSE = \frac{\sum_{i=1}^{n}(y_i - x_i)^2}{n}, \tag{2}$$

onde y_i é o valor previsto e x_i o valor real.

4. Materiais e Métodos

O trabalho é uma pesquisa de natureza exploratória, a qual visa investigar, compreender e aplicar técnicas de *Machine* e *Deep Learning* no contexto de tratamento de dados financeiros, com o intuito de construir modelos preditivos para a compra e venda de ações. Para isso, o trabalho foi desenvolvido em quatro etapas:

- Etapa 1: identificação de bases de dados (construção do *dataset*)
- Etapa 2: pré-processamento dos dados
- Etapa 3: aplicação de algoritmos
- Etapa 4: comparação e avaliação dos modelos

Foi utilizado o processo de descoberta de conhecimento em bases de dados, ou *Knowledge Discovery in Databases* (KDD) [Fayyad et al. 1996], voltado a identificar padrões de dados válidos, novos, potencialmente úteis e, por fim, compreensíveis. Embora tenha origem na área de bancos de dados, ele vem sendo amplamente utilizado na construção de sistemas preditivos. A abordagem pode ser visualizada na Figura 2.

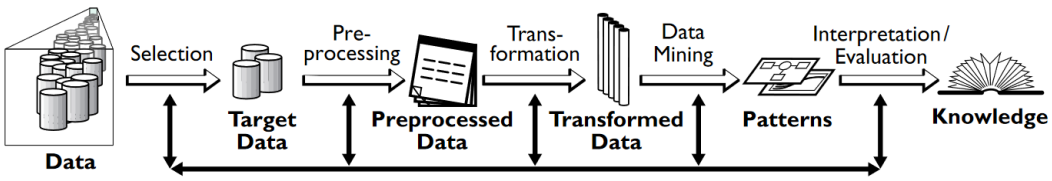

Figura 2. Passos que constituem o processo de descoberta de conhecimento em bases de dados [Fayyad et al. 1996]

A partir de um grande conjunto de dados disponíveis na internet, seleciona-se aqueles que de fato são relevantes. Após, é feito um pré-processamento, como limpeza e tratamento de ruído ou valores vazios. Na sequência, são feitas transformações a fim de encontrar características úteis para representar os dados, a depender da tarefa que se deseja realizar. Depois, escolhe-se e aplica-se uma ou mais técnicas de *data mining* para procurar por padrões nos dados. Por fim, o resultado encontrado é interpretado e avaliado, a fim de efetivamente convertê-lo em conhecimento. A qualquer momento é

possível voltar a passos anteriores. Por fim, é interessante usar o conhecimento descoberto, através, por exemplo, da incorporação a algum sistema ou em tomada de decisões [Fayyad et al. 1996].

4.1. Etapa 1: Identificação da Base de Dados

A Análise Técnica baseia-se em avaliar dados históricos de uma (ou mais) ações, realizar o cálculo de indicadores, e tomar decisões relacionadas a compra e venda das ações [Elder 2006]. Portanto, faz-se necessária uma base com as informações de negociação ao longo do tempo. Para aplicação empírica, identifica-se a base de dados pública *Ibovespa Stocks*[6]. O *dataset* contém três arquivos: ações listadas no Ibovespa; taxa Selic; e taxa de câmbio real/dólar.

A Bolsa de Valores brasileira disponibiliza em seu website[7] a possibilidade de se realizar consultas a dados históricos, os quais são disponibilizados diretamente em tela ou através de arquivos CSV. A base escolhida apresenta uma compilação dessas informações, contendo as ações que fazem parte do índice Ibovespa. O *dataset* contempla desde julho de 1994 a dezembro de 2020, período no qual é possível garantir cotações em Real (R$). No arquivo principal é possível perceber a presença de sete colunas, descritas a seguir:

- *datetime*: data na qual houve negociação de uma determinada ação.
- *ticker*: código da ação, geralmente uma abreviação a partir do nome
- *open*: preço de abertura da ação no dia
- *close*: preço de fechamento da ação no dia
- *high*: preço máximo pelo qual a ação foi negociada no dia
- *low*: preço mínimo pelo qual a ação foi negociada no dia
- *volume*: total em dinheiro movimentado em negociações da ação no dia

Além disso, o *dataset* apresenta um arquivo com a Taxa Selic do período, e outro com o câmbio Real/Dólar. Ambos os arquivos possuem formato muito semelhante: um campo com a data e outro com o valor da respectiva taxa.

4.2. Etapa 2: Pré-processamento

A etapa de pré-processamento é responsável por realizar a limpeza dos dados. Também podem ocorrer outras manipulações e cálculo de novas características (*features*) a partir da base existente.

Constatou-se que o *dataset*, em sua forma original, possui dados irrelevantes à Análise Técnica, visto que, de acordo com essa abordagem, toda a informação necessária já está presente no preço das ações [Elder 2006]. A partir disso, foram descartados os arquivos relativos à Taxa Selic, bem como ao câmbio Real/Dólar.

Em sua versão original, o *dataset* conta com mais de um 1.8 milhão registros, ocupando cerca de 100.6MB, conforme a Figura 3 a seguir, obtida através da biblioteca Pandas [Wes McKinney 2010].

Em seguida, foi feita uma análise a fim de identificar as ações com maior volume de negociação no ano de 2019. Esse ano foi escolhido por anteceder os efeitos causados pela pandemia de COVID-19. Após, foi selecionada aquela com maior movimentação.

[6] https://www.kaggle.com/felsal/ibovespa-stocks

[7] http://www.b3.com.br/en_us/market-data-and-indices/data-services/market-data/historical-data/equities/historical-quotes/

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1883203 entries, 0 to 1883202
Data columns (total 7 columns):
 #   Column    Dtype
---  ------    -----
 0   datetime  object
 1   ticker    object
 2   open      float64
 3   close     float64
 4   high      float64
 5   low       float64
 6   volume    float64
dtypes: float64(5), object(2)
memory usage: 100.6+ MB
```

Figura 3. Informações sobre o *dataset* em seu estado inicial

Essa análise foi feita através de consulta ao arquivo CSV contendo as anotações de preços das ações. Primeiramente, o arquivo foi importado para a memória com o uso da biblioteca Pandas [Wes McKinney 2010], gerando um *DataFrame*[8]. Após, fez-se a conversão da coluna *datetime* (data), de texto simples, para um objeto tipo *datetime*, o qual permite manipulação.

Na sequência, em uma cópia do *DataFrame* gerado, foram removidas as anotações de preço, mantendo apenas *ticker*[9], data, e volume. A seguir, selecionou-se apenas os registros pertinentes ao ano 2019. Após, agrupou-se as ações com base no *ticker*, realizando-se a soma dos volumes. Por fim, ordenou-se o resultado para obter as cinco ações com maior volume de negociação.

Para a realização desse procedimento, foi utilizado o seguinte trecho de código Python, utilizando a plataforma *Google Colab*[10]. O resultado é discutido na Seção 5.

```
# Inicializacoes
import pandas as pd
import numpy as np
import plotly.express as px
url_dataset = # link para o dataset em csv

# carga de dados para uma acao
dataset = pd.read_csv(url_dataset)

# analise acoes mais populares em 2019
dataset['datetime'] = pd.to_datetime(dataset['datetime'])
clean_dataset = dataset.copy()
clean_dataset.drop(columns=['open', 'close', 'high', 'low'],
```

[8]Estrutura de dados manipulável com o conteúdo do arquivo. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

[9]Identificador da ação.

[10]https://colab.research.google.com/

```
    inplace=True)
clean_dataset = clean_dataset[clean_dataset['datetime'].dt.year
    == 2019]
volume = clean_dataset.groupby(['ticker']).sum()
volume.sort_values('volume', ascending=False, inplace=True)
volume.head(5)
```

A partir dessa análise, identificou-se quais as cinco ações com maior volume de negociação. Após, filtrou-se o *dataset* a partir do *ticker* correspondente à ação mais negociada (PETR4, conforme será apresentado na Seção 5), a fim de manter apenas as anotações desejadas.

Após esse processo, o *dataset* limpo conta com 5640 registros, ocupando cerca de 352.5KB, conforme a Figura 4 a seguir, obtida através da biblioteca Pandas [Wes McKinney 2010].

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5640 entries, 209606 to 1881973
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   datetime  5640 non-null   object
 1   ticker    5640 non-null   object
 2   open      5640 non-null   float64
 3   close     5640 non-null   float64
 4   high      5640 non-null   float64
 5   low       5640 non-null   float64
 6   volume    5640 non-null   float64
dtypes: float64(5), object(2)
memory usage: 352.5+ KB
```

Figura 4. Informações sobre o *dataset* em seu estado já limpo

Por fim, a base de dados foi dividida em 80% dos dados para treino e 20% para teste [Moghar and Hamiche 2020]. Nessa divisão foi utilizado o pacote *scikit-learn*[Pedregosa et al. 2011], o qual fornece uma implementação do método *hold-out* descrito na Seção 3.3. Com isso, 4512 instâncias foram destinadas para treino dos modelos, e 1128 para teste.

4.3. Etapa 3: Aplicação de Algoritmos

Após análise de trabalhos relacionados, conforme mencionado na Seção 2, foi possível perceber a prevalência dos métodos LSTM e *Random Forest* na análise de dados de bolsas de valores. Com base nisso, foram realizados testes utilizando essas duas abordagens, a fim de compará-las e verificar se é possível obter predições de preços de ações no mercado brasileiro.

Para realização dos testes utilizou-se o mecanismo de *Grid search*. *Grid Search* é uma técnica de otimização de hiper-parâmetros, bastante utilizada na construção de modelos preditivos [Chalvatzis and Hristu-Varsakelis 2020, Yadav et al. 2020]. Consiste em

elencar valores possíveis para alguns parâmetros e realizar testes combinados a fim de encontrar qual a configuração mais adequada. Neste trabalho foi utilizado o pacote *scikit-learn* [Pedregosa et al. 2011], o qual conta com uma implementação própria de *Grid Search*[11]. Para a comparação entre os dois modelos, foi utilizado o erro médio quadrático (MSE), conforme descrito na Seção 3.3. Nos próximos tópicos, detalha-se os experimentos realizados.

4.3.1. Experimento usando LSTM

A partir da leitura e análise de trabalhos relacionados, constatou-se a relevância de modelos baseados na rede *Long Short-Term Memory* (LSTM), a qual foi descita na Seção 3.2. Esse modelo de *deep learning*, proposto por [Hochreiter and Schmidhuber 1997], foi criado especialmente para o trabalho com séries temporais.

Para a geração do modelo LSTM, foram feitas etapas adicionais de pré-processamento. Realizou-se a normalização através de conversão para o intervalo [-1, 1], com o auxílio da biblioteca *scikit-learn*[Pedregosa et al. 2011]. Essa etapa é importante para atenuar os efeitos de valores extremos no processo de aprendizagem do modelo.

Na sequência, dividiu-se a base em *time steps* de 10 amostras, a fim de gerar as séries temporais de entrada para a LSTM. Cada registro na base de dados corresponde a um dia de negociação, a qual somente ocorre em dias úteis. Dessa forma, cada *time step* contém o equivalente a duas semanas de operações. Esse processo foi realizado conforme o trecho de código a seguir:

```
# Criacao do dataset por serie temporal
def create_dataset(X, y, time_steps=10):
   Xs, ys = [], []
   for i in range(len(X) - time_steps):
      v = X[i:(i + time_steps)]
      Xs.append(v)
      ys.append(y[i + time_steps])
   return np.array(Xs), np.array(ys)

time_steps = 10

X_train_f, y_train_f = create_dataset(x_train_transformed,
   y_train_transformed, time_steps)
X_test_f, y_test_f = create_dataset(x_test_transformed,
   y_test_transformed, time_steps)
```

onde X corresponde aos dados de entrada do modelo (as cinco colunas do dataset original), e y à saída esperada (coluna com o preço de fechamento do dia). Para cada registro i de entrada, é criado um vetor contendo os próximos 10 valores, e atribuído como valor de saída esperada o valor correspondente ao último valor i da sequência de entrada. A função foi executada tanto para a base de treino, quanto para a base de teste.

[11] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Após, foi criado o modelo preditivo a partir da implementação disponível no TensorFlow [Abadi et al. 2015]. Foi criado um modelo sequencial[12], contendo uma camada de entrada[13], uma camada LSTM[14], uma camada *Flatten*[15], e uma camada densa[16].

A camada de entrada é responsável por receber as sequências de dados criadas anteriormente. Foi criada passando como parâmetro o formato dos dados esperados. A camada LSTM, como o nome sugere, é onde está implementada a rede de interesse. Foram utilizadas 100 unidades recorrentes, a função de ativação tangente hiperbólico, e não foi utilizado *dropout*. A configuração baseou-se na literatura [Yadav et al. 2020]. A camada *Flatten* é responsável por "achatar"a saída da LSTM. A camada densa é utilizada para gerar a saída do modelo. Possui uma única unidade, a fim de gerar um único valor de saída.

Por fim, o modelo é compilado, utilizando o erro médio quadrado como função de perda (*loss*). O otimizador utilizado foi o Adam, popular em trabalhos de *deep learning* [Chalvatzis and Hristu-Varsakelis 2020, Eliasy and Przychodzen 2020, Thakkar and Chaudhari 2020]. O otimizador é um algoritmo responsável por ajustar os pesos das camadas a partir do resultado da *loss*, através de gradientes. O modelo descrito foi implementado usando o trecho de código a seguir:

```
model = keras.Sequential()
model.add(keras.layers.Input(shape=((X_train_f.shape[1],
   X_train_f.shape[2]))))
model.add(layers.LSTM(100, activation = 'tanh', dropout=0))
model.add(layers.Flatten())
model.add(keras.layers.Dense(units=1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.summary()
```

Para a fase de treinamento do modelo foram utilizadas 100 épocas, valor que forneceu o melhor resultado em [Moghar and Hamiche 2020]. Também foi utilizado um tamanho de *batch* de 10, e uma divisão de 10% para validação. Cada época é uma iteração em cima de todo o *dataset* de entrada. O tamanho do *batch* é o número de amostras que serão utilizadas em cada atualização do gradiente (etapa de otimização). A divisão de validação funciona de forma similar à divisão de treino e teste, retirando parte dos dados para usá-los no cálculo da *loss* e outras eventuais métricas. Também não foi utilizado *Shuffle*, ou randomização, dado que o *dataset* é uma série temporal. Após, foi feita a predição. Por fim, o valor previsto é re-escalado para um número real. A fase de treino foi implementada conforme o trecho de código a seguir:

```
model.fit(X_train_f, y_train_f, batch_size = 10, epochs = 100,
   shuffle=False, validation_split=0.1)
```

[12]https://keras.io/guides/sequential_model/
[13]https://keras.io/api/layers/core_layers/input/
[14]https://keras.io/api/layers/recurrent_layers/lstm/
[15]https://keras.io/api/layers/reshaping_layers/flatten/
[16]https://keras.io/api/layers/core_layers/dense/

4.3.2. Experimento usando *Random Forest*

O modelo de *Random Forest*, técnica apresentada na Seção 3.1, foi gerado utilizando a técnica de *Grid Search*, a partir da implementação do SciKit [Pedregosa et al. 2011][17]. Também foi utilizado o *KFold Cross-Validation* [Faceli et al. 2021, Pedregosa et al. 2011], apresentado na Seção 3.3.

Para a execução do *Grid Search*, considerou-se os parâmetros conforme a Tabela 2, baseada em [Chalvatzis and Hristu-Varsakelis 2020]. O número de árvores con-

Tabela 2. Parâmetros utilizados para *Grid Search* no modelo *Random Forest*

Parâmetro	Intervalo
Número de árvores ($n\_estimators$)	10, 20, 100, 1000
Profundidade máxima (max_depth)	1, 25, 50
Mínimo de amostras para divisão ($min\_samples\_split$)	2, 5
Máximo de nós folha (max_leaf_nodes)	None, 2, 5

trola quantos estimadores existirão no modelo. Valores mais altos aumentam a complexidade de tempo do modelo, não necessariamente levando a resultados melhores. O valor *default* é 100.

A profundidade máxima indica o maior tamanho de cada estimador. É um parâmetro importante, dado que caso o estimador cresce demais, pode levar ao problema de *overfitting*. O padrão da biblioteca é não limitar ($None$), ou seja, a árvore cresce até que todos os nodos tenham o mímimo de amostras por divisão.

O mínimo de amostras para divisão indica o mínimo de amostras que cada nodo interno deve ter para ser dividido. Aumentando o valor, haverão menos divisões, ajudando a evitar o *overfitting*. Porém, um valor muito alto pode levar a *underfit*. O valor *default* da biblioteca é 2.

O máximo de nós folha define um limite na divisão das árvores, ajudando a reduzir o tamanho máximo do estimador. Com isso, reduz a probabilidade de *overfitting*. O valor *default* é não limitar ($None$).

Realizou-se o treinamento utilizando a técnica de *K-fold cross-validation* com 10 divisões (*splits*), apresentado na Seção 3.3. Por fim, foi utilizado o erro médio quadrático como critério de avaliação do modelo, similar à LSTM.

Após a execução do *Grid Search*, constatou-se que a melhor parametrização é utilizar profundidade máxima de 25 ($max\_depth : 25$), não limitar o número máximo de nós folha ($max\_leaf\_nodes = None$), manter 2 como o mínimo de amostras para divisão ($min\_samples\_split = 2$), e 10 árvores ($n\_estimators = 10$). A implementação do modelo pode ser vista no trecho de código a seguir, no qual a última linha é o resultado gerado.

```
crossvalidation = KFold(n_splits=10,shuffle=False)

estimator = RandomForestRegressor()
```

[17]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

```
hyperparams = {
   'n_estimators': [10, 20, 100],
   'max_depth': [None, 25, 50],
   'min_samples_split': [2, 5],
   'max_leaf_nodes': [None, 2, 5],
}

grid = GridSearchCV(estimator, param_grid=hyperparams,
   scoring='neg_mean_squared_error', verbose=1,
   return_train_score=False, cv=crossvalidation, n_jobs=-1)
grid.fit(X_train, np.ravel(Y_train))

% {'max_depth': 25, 'max_leaf_nodes': None,
   'min_samples_split': 2, 'n_estimators': 10}
```

4.4. Etapa 4: Comparação dos modelos

Para avaliar o desempenho dos modelos propostos, foram utilizados critérios frequentemente utilizados na literatura, conforme descoberto na Seção 2. Em trabalhos envolvendo Redes Neurais Deep Learning utiliza-se métricas quantitativas para avaliação dos modelos. Sendo assim, a avaliação segue os padrões estabelecidos na área, baseados em critérios numéricos de acurácia do modelo, em particular o erro médio quadrático (MSE), e a sua raiz quadrada, o RMSE. Após a comparação entre os modelos selecionados, selecionou-se aquele com menores métricas de erro como sendo o melhor.

5. Resultados e Discussões

A partir da etapa de pré-processamento, descrita na Seção 4.2, foi possível perceber quais as ações mais negociadas no ano de 2019. O papel com maior volume foi o da Petrobras (PETR4), conforme a Tabela 3. A partir dessa análise, escolheu-se essa ação para o treinamento dos modelos preditivos.

Tabela 3. Ações mais negociadas em 2019 na B3

Ticker	Volume
PETR4	3.557940e+11
VALE3	2.507101e+11
ITUB4	1.808763e+11
BBDC4	1.592653e+11
BBAS3	1.386589e+11

A execução dos experimentos com *Random Forest* e LSTM pode ser comparada tanto através de seu MSE quanto através do RMSE. Conforme evidenciado na Tabela 4, percebe-se que a variação usando *Random Forest* apresenta melhor resultado.

A execução do experimento de *Random Forest*, conforme detalhado na Seção 4.3.2, resultou em um modelo com alta precisão em relação aos dados reais. Obteve-se MSE de 0,00114 e RMSE de 0,02182, conforme exposto na Tabela 4. Percebe-se na Figura 5 que a linha vermelha, dos valores previstos, sobrepõe a linha azul, dos valores

Tabela 4. RMSE dos modelos avaliados

Modelo	MSE	RMSE
Random Forest	0.0011377295	0.0337302083
LSTM	2.9563245085	1.7193965536

reais, em quase a totalidade do gráfico. Com isso, pode-se dizer que o modelo conseguiu reagir às diferentes movimentações do preço da ação.

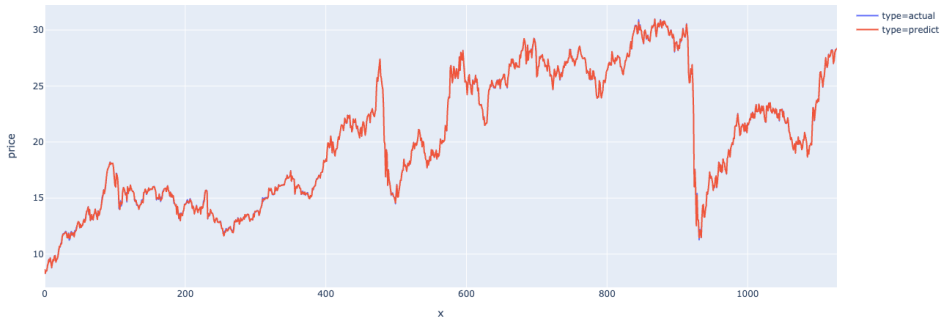

Figura 5. Predição de de PETR4 usando *Random Forest*

O experimento envolvendo LSTM, detalhado na Seção 4.3.1, gerou um modelo com menor precisão em relação aos dados reais. Obteve-se MSE de 2,95632 e RMSE de 1,71940, conforme exposto na Tabela 4. Percebe-se na Figura 6 que os valores previstos (em vermelho) acompanham a tendência dos valores reais (em azul) em quase toda a extensão do gráfico, mas há alguma diferença entre ambas as curvas.

Figura 6. Predição de PETR4 usando LSTM

A partir da execução dos modelos propostos, nota-se que é viável a predição de preços de ações da Bolsa de Valores brasileira, a B3. Esse resultado colabora com a hipótese de [Eliasy and Przychodzen 2020], que inteligência artificial pode ser utilizada para esse fim.

Ainda, percebe-se que a *Random Forest* (Figura 5) foi capaz de produzir predições mais assertivas, quando comparada com a LSTM (Figura 6). Essa percepção se sustenta ao analisar a métrica de erro dos modelos, conforme Tabela 4.

Era esperado que o experimento utilizando LSTM fornecesse melhores resultados, dado que é considerado o melhor modelo para o trabalho com séries temporais [Eliasy and Przychodzen 2020]. O resultado encontrado concorda com [Huck 2019], que também apresenta bom desempenho da *Random Forest*. Porém, contrasta com o encontrado por [Chalvatzis and Hristu-Varsakelis 2020], o qual indica que redes LSTM costumam ter melhor performance.

6. Considerações Finais

Investir no mercado de ações é bastante desafiador. Sistemas preditivos para a área financeira podem auxiliar nesta tarefa. Conforme apresentado neste trabalho, o mercado de ações pode parecer um pouco aleatório, causando dúvidas e incertezas. Contudo, dentro dessa área, existem vários nichos já bastante conhecidos e explorados, como predições no mercado de ações, avaliação de riscos de crédito, alocação de portfólio, precificação de ativos, predição de crises financeiras, entre outros.

Com a ampliação do acesso à tecnologia, é natural que haja uma tendência de alta na popularidade da computação aplicada às finanças. Dentre as áreas da Computação, destaca-se a IA e seus métodos. A área de aprendizado de máquina, ou *Machine Learning*, é particularmente notável nesse tema, já com várias abordagens conhecidas. Recentemente, as técnicas de *Deep Learning*, ou aprendizagem profunda, têm ganhado bastante relevância.

Neste trabalho, de cunho exploratório, desenvolveu-se uma revisão sistemática da literatura referente a sistemas preditivos da área financeira. Por meio deste estudo, identificou-se que métodos como LSTM e *Random Forest* apresentaram bons resultados na previsão de preços de ações. Também foi realizada uma breve apresentação dos principais conceitos financeiros, de inteligência artificial, e dos algoritmos envolvidos.

Após, realizaram-se experimentos utilizando os algoritmos LSTM e *Random Forest* a fim de se obter um modelo preditivo para a compra e venda de ações da Bolsa de Valores brasileira, a B3. Apresentou-se o passo a passo realizado, desde a identificação da base de dados até a implementação dos modelos.

Por fim, os resultados foram apresentados. Com base na literatura, esperava-se que o algoritmo de melhor desempenho fosse o LSTM, porém o melhor modelo foi o construído a partir da *Random Forest*. Ambos os modelos, entretanto, exibiram resultados que comprovam a viabilidade de utilizar-se técnicas de inteligência artificial no mercado de ações brasileiro.

A quantidade e profundidade dos testes foram limitadas, em virtude do alto custo computacional em operações de otimização de modelos preditivos. Em decorrência disso, sugere-se realizar mais testes de configurações. É interessante aplicar técnicas de otimização de hiper-parâmetros como *Grid Search* em modelos baseados em LSTM, mas alerta-se para o alto custo computacional da tarefa. Sugere-se avaliar a capacidade de generalização dos modelos, bem como realizar subsequentes experimentos a fim de otimizá-la. Ainda, existe a oportunidade de verificar a influência de indicadores técnicos no desempenho de sistemas preditivos. Por fim, existe a possibilidade de se implementar um *software* a fim de integrar um modelo preditivo a uma corretora, a fim de realizar negociações automatizadas.

Referências

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. Disponível em https://www.tensorflow.org.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Chalvatzis, C. and Hristu-Varsakelis, D. (2020). High-performance stock index trading via neural networks and trees. *Applied Soft Computing*, 96:106567.

Elder, A. (2006). *Aprenda a operar no mercado de ações*. Elsevier, Rio de Janeiro, RJ, 11 edition. Tradução de Afonso Celso da Cunha Serra.

Eliasy, A. and Przychodzen, J. (2020). The role of ai in capital structure to enhance corporate funding strategies. *Array*, 6:100017.

Faceli, K., Lorena, A. C., Gama, J., Almeida, T. A. d., and Carvalho, A. C. P. d. L. F. d. (2021). *Inteligência Artificial*. LTC, Rio de Janeiro, RJ, 2 edition.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34.

Giacomel, F. d. S. (2016). Um método algorítmico para operações na bolsa de valores baseado em ensembles de redes neurais para modelar e prever os movimentos dos mercados de ações. Master's thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Haykin, S. (2007). *Redes Neurais*. Bookman, São Paulo. Tradução de Paulo Martins Engel.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Huck, N. (2019). Large data sets and machine learning: Applications to statistical arbitrage. *European Journal of Operational Research*, 278(1):330–342.

Long, J., Chen, Z., He, W., Wu, T., and Ren, J. (2020). An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in chinese stock exchange market. *Applied Soft Computing*, 91:106205.

Medeiros, Y. D. and Leite, E. d. S. (2013). Os investidores profissionais brasileiros e a cultura do lucro e do risco no mercado financeiro. In *XXII Congresso de Iniciação Científica da Universidade de Pelotas*, Pelotas, RS. Universidade Federal de Pelotas, Universidade Federal de Pelotas.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

Mochón, A., Quintana, D., Sáez, Y., and Isasi, P. (2007). Soft computing techniques applied to finance. *Applied Intelligence*, 29(2):111–115.

Moghar, A. and Hamiche, M. (2020). Stock market prediction using lstm recurrent neural network. *Procedia Computer Science*, 170:1168–1173. The 11th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops.

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

Ozbayoglu, A. M., Gudelek, M. U., and Sezer, O. B. (2020). Deep learning for financial applications : A survey. *Applied Soft Computing*, 93:106384.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Quesada, B. (2021). Número de investidores na bolsa cresce 43% e se aproxima dos 4 milhões. Publicado por Exame Invest. Disponível em: https://invest.exame.com/me/numero-de-investidores-na-bolsa-cresce-43-e-se-aproxima-dos-4-milhoes. Acesso em: 10 out. 2021.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Thakkar, A. and Chaudhari, K. (2020). Crest: Cross-reference to exchange-based stock trend prediction using long short-term memory. *Procedia Computer Science*, 167:616–625. International Conference on Computational Intelligence and Data Science.

Vijh, M., Chandola, D., Tikkiwal, V. A., and Kumar, A. (2020). Stock closing price prediction using machine learning techniques. *Procedia Computer Science*, 167:599–606. International Conference on Computational Intelligence and Data Science.

Wes McKinney (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

Yadav, A., Jha, C. K., and Sharan, A. (2020). Optimizing lstm for time series prediction in indian stock market. *Procedia Computer Science*, 167:2091–2100. International Conference on Computational Intelligence and Data Science.