Insper

Engenharia de Computação

Relatório Final de Iniciação Científica

Caio Emmanuel Soares Rocha

Orientador: Prof. Raul Ikeda Gomes da Silva

Estudo de *Generative Adversarial Networks* aplicadas ao Mercado Acionário Brasileiro

São Paulo

2022

Resumo

Este trabalho tem como fim estudar métodos de *Machine Learning*, mais em específico, a área de Redes Neurais, e reproduzir um trabalho na área, aplicando técnicas de *Generative Adversarial Network* (GAN) ao mercado financério. O relatório consiste de uma breve revisão bibliográfica sobre o histórico de técnicas para estudar e modelar o valor do preço de ações, a metodologia de coleta e análise de dados e o processo de construção de variáveis e implementação parcial de um modelo mais simples. Por fim, é apresentado o resultado da reprodução do trabalho base[1].

Palavras-chave: *Generative Adversarial Network*, Aprendizado Profundo, *Long Short-Term Memory*, Rede Neural Convolucional, Mercado Financeiro

Abstract

This work is an study on Machine Learning methods, mainly on Neural Network field and reproduce an work which applies Generative Adversarial Network (GAN) techniques in the financial market. This paper consists of a brief review over the literature on the history of techniques to study and model the value of the stock price, the methodology of data collection and analysis, the process of constructing variables and partial implementation of a model in Brazilian market context. Finally, the reproduction of the base work[1] is presented.

Palavras-chave: Generative Adversarial Network, Deep Learning, Long Short-Term Memory, Convolutional Neural Network, Financial Market

Sumário

1 Introdução **1**

2 Revisão Bibliográfica **1**

3 Metodologia **5**

3.1 Base de Dados 5

3.2 Variáveis construídas 6

3.3 Manipulação das variáveis 6

3.4 Construção e treinamento do modelo preliminar 7

3.5 Comparativo 9

4 Resultados Finais **9**

5 Conclusão **10**

A *Features* do modelo **12**

1 Introdução

Estudar e modelar o comportamento o valor do preço de uma ação na bolsa de valores tem se mostrado um desafio complexo devido a dinâmica caótica desse sistema e variáveis estocásticas não-estacionárias envolvidas no processo[2, 3].

Entretanto, os prêmios e incentivos para resolver este desafio são tão grandes quando o desafio, uma vez que a capacidade de estimar tais valores minimizaria os riscos de operação ao passo que poderia potencializar os retornos sobre investimento, motivando pesquisadores e organizações a investirem tempo e recursos nessa tarefa.

Uma técnica foi proposta por *Xingyu Zhou et al.*[1] e consiste em usar uma rede neural geradora que diminua o erro na estimativa do valor do preço (*forecast*) e na direção da previsão (*direction*) baseada na arquitetura de uma *Generative Adversarial Networks* (redes concorrentes geradoras, em tradução, ou a sigla GAN)[4, 5], chamada de *GAN-FD*.

O objetivo desse trabalho é reproduzir parcialmente a metodologia descrita no estudo original[1], apresentar os resultados preliminares obtidos e, sobre estes, fazer uma análise crítica de erros e acertos e detalhar os próximos passos.

2 Revisão Bibliográfica

Várias soluções foram desenvolvidas para lidar com séries temporais, no campo da econometria, alguns dos modelos mais populares são *autoregressive method (AR), moving average (MA)* e *autoregressive integrated moving average (ARIMA)*[6, 7] que, explicando brevemente, inferem o preço em um momento t utilizando uma combinação linear dos preços nos instantes $\{1, 2, ..., t-1\}$. Entretanto, um problema desta classe de modelos é que eles partem de premissas sobre a diferença entre o preço em dois instantes (como distribuição-t ou variáveis independentes e identicamente distribuídas), que não se confirmam com dados reais na maioria das vezes.

Já no campo de *soft computing*, área que envolve inteligência artificial, algumas das principais soluções incluem a utilização de *Redes Neurais Artificiais*[8, 9] e *Support Vector Machine*[10, 11] e outras da área de *Deep Learning*[12], que têm atraído olhares devido

à capacidade dessa estrutura de extrair *features* abstratas dos dados, inclusive de dados não estruturais, como *tweets* ou *headlines* de jornais[13, 14]. Entretanto, essa classe de modelos têm uma série de problemas que dificultam a implementação como custo de treinamento das redes neurais, expertise necessária para lidar com as especificidades e restrições dos diferentes tipos de dados que podem ser passados por estas.

Nesse contexto surge o modelo de GAN-FD[1], abreviação para *Generative Adversarial Network - Forecast and Direction*, um novo modelo que combina as qualidades dos modelos de *Deep Learning* mas que usa apenas dados simples na entrada que podem ser extraídos de um *software* de *trading*, o que reduz a barreira de acesso para investidores e instituições menores.

A GAN-FD tem sua estrutura baseada nas *Generative Adversarial Network (GAN)*[4, 5], uma estrutura de rede neural desenvolvida para recriar imagens a partir de uma matriz ruidosa de dados fornecida. A estrutura principal de uma *GAN* consiste de duas redes neurais adversárias que competem entre si, a primeira é a rede **Geradora (G)** e a rede **Discriminante (D)**, que podem ser vistas na Figura 1. Enquanto G tem a função de gerar amostras de forma a enganar D, a rede D tem o papel de distinguir corretamente amostras reais de amostras geradas por G.

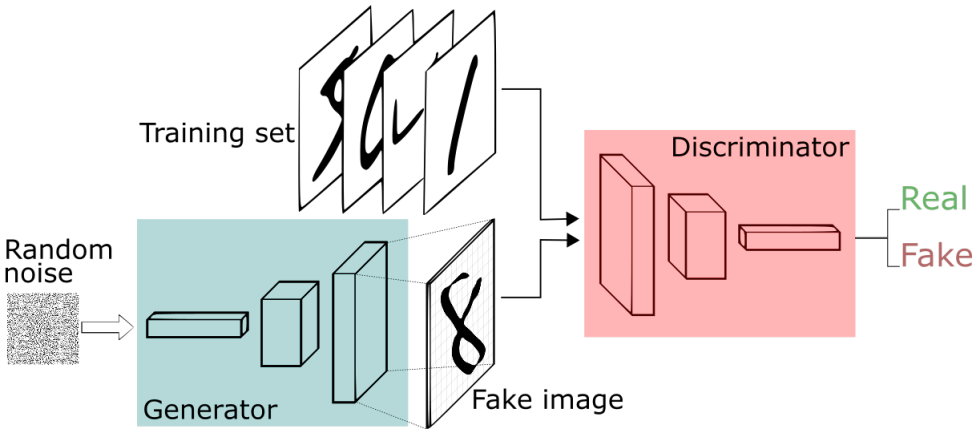

Figura 1: Arquitetura de uma *GAN* clássica[15]

Para implementar uma GAN-FD precisa-se adaptar as redes neurais escolhidas para compor G e D, de modo que seja possível treinar com o tipo de dado escolhido para realizar a estimativa e que diminua o erro na estimativa do preço (*forecast*) e na direção

da previsão (*direction*).

A solução encontrada por *Xingyu Zhou et al.*[1] é substituir as redes neurais que são tradicionalmente usadas como geradoras por uma *LSTM*, abreviatura de *Long Short-Term Memory*[16, 17], uma arquitetura baseada em *Recurrential Neural Networks* (RNN), uma classe de redes neurais desenvolvida para tratar com sequências de dados. Ao invés das redes neurais mais tradicionais, *Artificial Neural Networks* (ANN), que lidam com cada entrada da sequência de dados separadamente, as RNNs iteram seu *estado*[18] a cada nova entrada da sequência e esse estado é reiniciado entre o processamento de uma sequência e a próxima, em suma, é semelhante a uma rede neural com um loop, onde o output é usado como input em uma próxima iteração. Fazendo das RNNs ideais para processamento de séries temporais, como a série de preços ou indicadores de uma ação. A LSTM foi criado com o intuito de solucionar um problema que as primeiras RNNs possuiam: preservar informação passada entre unidades futuras da rede[18] e este é o principal motivo de usá-las.

A rede discriminante é uma *Rede Neural Convolucional (CNN)* como na arquitetura original de uma *GAN*, que, nesse caso, recebe ambas a série de preços reais e a série de preços com o último elemento sendo o valor predito e através de operações de convolução retorna se a série falsa foi capaz de enganá-la ou não[19].

A estrutura completa da GAN-FD pode ser vista na Figura 2, onde X é a série de entrada, $\hat{Y}$ é a série de preços com o último valor sendo o preço estimado e Y é a série de preços real.

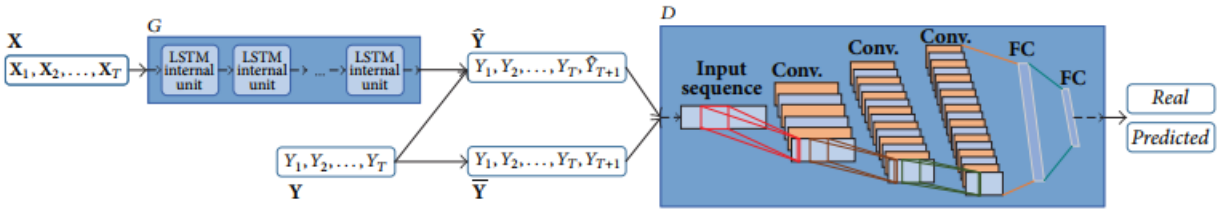

Figura 2: Arquitetura de uma GAN-FD[1]

É importante salientar que a arquitetura proposta na Figura 2 não corresponde a uma *GAN* tradicional, como na Figura 1. Uma vez que não possui como entrada um vetor ruidoso que é requisito para a geração de dados falseados. E também não corresponde a

estrutura de uma *Conditional-GAN*[4] que, apesar de receber na entrada uma porção de dados reais, não dispensa a necessidade do vetor ruidoso.

Para alimentar a rede, *Zhou et al.* utiliza uma base de dados composta por:

- Valores dos preços das ações que compõe o CSI 300 (índice composto por 300 ações do mercado acionário de Shangai e Shenzen) durante o ano de 2016, filtrando apenas aquelas cuja operação não foi cancelada durante esse período;
- Dados dos dez indicadores financeiros mais populares para análise técnica.

Para treinar a rede G, foi usado uma combinação de diferentes funções de *loss*[20]. Para explicar isto é importante definir X_t como sendo o conjunto de indicadores no instante t, Y_t é o preço de fechamento no mesmo instante.

A primeira função de *loss* é a *adversarial loss* de G ($L^G_{adv}(\hat{Y})$), vide equação (1).

$$L^G_{adv}(\hat{Y}) = L_{sce}(D(\hat{Y}), 1) \tag{1}$$

Onde $\hat{Y}$ é o preço de fechamento estimado pela rede G ($\hat{Y}_{T+1} = G(X)$), onde **X** é o *set* de todos os X_t (t = 1, 2, ... , T) e $L_{sce}(D(\hat{Y}), 1)$ é dado pela equação (2).

$$L_{sce}(D(\hat{Y}), 1) = -\sum_i log(\sigma(D(\hat{Y}_i))) \tag{2}$$

Entretanto, para assegurar que a rede G não somente confundiu D como estimou o preço com uma boa diferença, adiciona-se o *forecast error loss* (L_p), vide equação (3).

$$L_p(\bar{Y}, \hat{Y}) = ||\bar{Y} - \hat{Y}||_p \tag{3}$$

E por último, a direção da predição é importante elemento para realizar o trading, portanto, acrescenta-se a *direction prediction loss* (L_{dpl}) calculado conforme (4).

$$L_{dpl} = |sgn(\hat{Y}_{T+1} - Y_T) - sgn(Y_{T+1} - Y_T)| \tag{4}$$

Onde sgn é a função sinal que retorna -1 caso o valor seja negativo, 0 caso seja zero e 1 caso seja positivo.

Criando os parâmetros da combinação linear λ_{adv}, λ_p e λ_{dpl}, a função *loss* de G será aquela dada por (5).

$$L_G(X,Y) = \lambda_{adv} L^G_{adv}(\hat{Y}) + \lambda_p L_p(\bar{Y},\hat{Y}) + \lambda_{dpl} L_{dpl} \tag{5}$$

Já a função *loss* para treinar D será apenas a *adversarial loss* de D dado por (6).

$$L_D(X,Y) = L^D_{adv}(\bar{Y},\hat{Y}) = L_{sce}(D(\hat{Y}),0) + L_{sce}(D(\hat{Y}),1) \tag{6}$$

O algoritmo de treino usado será o *Stochastic Gradient Descent (SGD)*[21].

Já para validação e comparação, as métricas usadas serão a *Raiz do Erro Médio Quadrático Relativo*, ou *RMSRE* (*Root Mean Squared Relative Error*) definida pela equação (7).

$$RMSRE = \sqrt{\frac{1}{T_0}\sum_t \left(\frac{\hat{Y}_{t+1} - Y_{t+1}}{Y_{t+1}}\right)^2} \tag{7}$$

Utilizado para medir o erro relativo à distância entre a estimativa e o valor real.

E também a *Acurácia na Direção da Previsão*, *DPA* (*Direction Prediction Accuracy*) definido por (8).

$$DPA = \frac{100}{T_0}\sum_t I_t \tag{8}$$

Onde:

$$I_t = \begin{cases} 1, (\hat{Y}_{T+1} - Y_T))(Y_{T+1} - Y_T)) > 0 \\ 0 \end{cases} \tag{9}$$

3 Metodologia

3.1 Base de Dados

Para adaptar o trabalho ao mercado brasileiro, foi usada uma base de dados de ações da companhia Itaú, nos últimos 20 anos (maio de 2002 até maio de 2022). Não foi

tratado o *gap* que existe entre o preço de fechamento e abertura entre dois dias, pois o trabalho original não citava se resolveu esse gap.

3.2 Variáveis construídas

Para treinar o modelo capaz de estimar o valor do preço de fechamento são usadas um total de 15 *features* (variáveis) detalhadas no Apêndice A.

Sendo Open Price, High Price e Close Price retiradas diretamente da base de dados e as demais indicadores construídos a partir dos preços de fechamento. O valor para o parâmetro desses indicadores não foi especificado no trabalho original e, portanto, foi usado os valores mais comuns em análises técnicas de mercado. Os indicadores foram construídos usando uma biblioteca de *Python* chamada *Pandas Technical Analysis* (Pandas-TA)[22].

3.3 Manipulação das variáveis

As *features* criadas foram adicionadas ao conjunto de dados inicial e o novo conjunto foi separado em um conjunto de treinamento e um conjunto de testes, na proporção de 4:1, atentando em manter a disposição original do conjunto de dados, uma vez que a ordem dos preços importa.

Os dados de treinamento foram escalonados pelo método de mínimo e máximo da equação (10) em um intervalo de 0 a 1, pois esse é um pre-requisito para o formato da entrada de uma rede LSTM. Os dados de teste foram escalonados utilizando o mesmo mínimo e máximo dos dados de treinamento, para evitar um vazamento de dados, uma vez que não é conhecido durante o treinamento o mínimo e máximo dos dados de teste.

$$x_{escalonado} = \frac{x - Min(X)}{Max(X) - Min(X)}, x \in X \tag{10}$$

Após essa primeira manipulação, ambos dados de treinamento e dados de teste foram divididos na variável dependente que deseja-se estimar (preço de fechamento) e as *features* do modelo tal qual o trabalho original recomenda, isto é, utilizando os 30

primeiros minutos de um intervalo para prever o trigésimo primeiro minuto após estes. Dessa forma, obtem-se uma lista de matrizes (X) de dimensão 30x15 - 30 linhas, uma para cada minuto e 15 colunas uma para cada feature - e outra lista com os valores do preço de fechamento após cada intervalo de 30 minutos (y).

Essa repartição foi feita seguindo o modelo do trabalho original, isto é, utilizando uma janela de 30 minutos para prever o trigésimo primeiro e movendo a janela de 1 em 1 minuto a frente até cobrir toda base de dados, como na Figura 3, fazendo o par (M, N) = (30, 1).

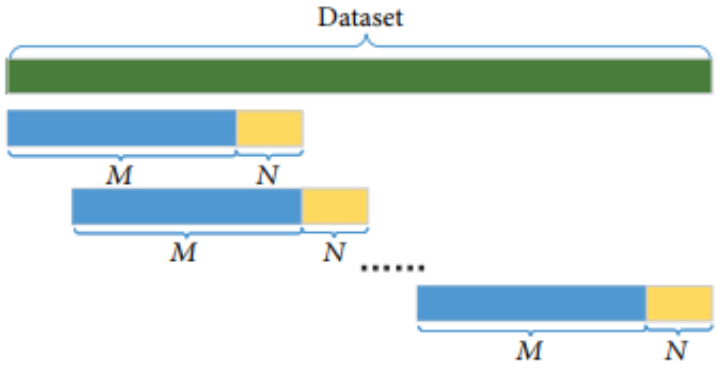

Figura 3: Esquema de segmentação do banco de dados[1]

3.4 Construção e treinamento do modelo preliminar

A primeira ideia para atacar o problema foi construir e treinar, separadamente, as redes G e D da GAN-FD, mas seguindo a arquitetura proposta no trabalho original.

Dessa forma, a rede Geradora consiste de uma LSTM de 121 unidades, como na Tabela 1, e esta foi construída utilizando o *framework* Keras apoiado na biblioteca TensorFlow do Python.

A rede foi treinada utilizando o algoritmo de SGD recomendado pelo trabalho original e foi utilizado o Erro Médio Quadrático como *loss function* e métrica para analisar a convergência da rede.

Modelo: Geradora		
Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 121)	68244
Total params: 68,244 Trainable params: 68,244 Non-trainable params: 0		

Tabela 1: Sumário da rede geradora

Como as etapas de geração e categorização dos dados foi feita separadamente, foi preciso treinar a rede G antes de prosseguir na construção da base de dados de treino que irá alimentar a rede geradora.

Com essa base, deseja-se testar a capacidade de uma rede Discriminante com a arquitetura proposta identificar a série como real ou falsa, portanto uma variável categórica é adicionada nessa nova base indicando se a série é real (1) ou falsa (0).

Por fim essa base final foi separada em dados de treino e teste para a rede discriminante usando a função *train test split* do módulo Scikit Learn[23] do Python.

A rede D é então construída tal qual a arquitetura do trabalho original, como mostra a Tabela 2.

Modelo: Discriminante		
Layer (type)	Output Shape	Param #
conv1d_26 (Conv1D)	(None, 14, 32)	160
conv1d_27 (Conv1D)	(None, 6, 64)	8256
batch_normalization_14 (BatchN	(None, 6, 64)	256
conv1d_28 (Conv1D)	(None, 2, 128)	32896
batch_normalization_15 (BatchN	(None, 2, 128)	512
dense_22 (Dense)	(None, 2, 128)	16512
flatten_11 (Flatten)	(None, 256)	0
dense_23 (Dense)	(None, 2)	514
Total params: 59,106 Trainable params: 58,722 Non-trainable params: 384		

Tabela 2: Sumário da rede discriminante

O treinamento foi realizado utilizando como parâmetros 10 epochs e um tamanho de batch de 32. Após o treinamento, foi gerada uma série de preços de fechamento falsos

($\hat{y}$) utilizando o conjunto de *features* X da base de testes (X_{test}). Essa série de preços foi separada em janelas de 30 minutos e concatenada em uma base com a série de preços reais também separada em janelas de 30 minutos, obtendo assim uma matriz, onde cada linha representa uma série de preços que pode ser real ou falsa e cada coluna um minuto no intervalo de 1 a 30.

3.5 Comparativo

O comparativo de performance será feito contra uma rede geradora com a mesma topologia, porém treinada na ausência da rede discriminante. Dessa forma, visa-se verificar se a rede discriminante incrementa o desempenho ou não.

4 Resultados Finais

Durante o treinamento, foi verificado que ambas as redes convergiam rapidamente, conforme representado na Figura 4, de forma que a rede G perdia rápido por não conseguir se adaptar à dificuldade de treinar sobre a competição com a rede D.

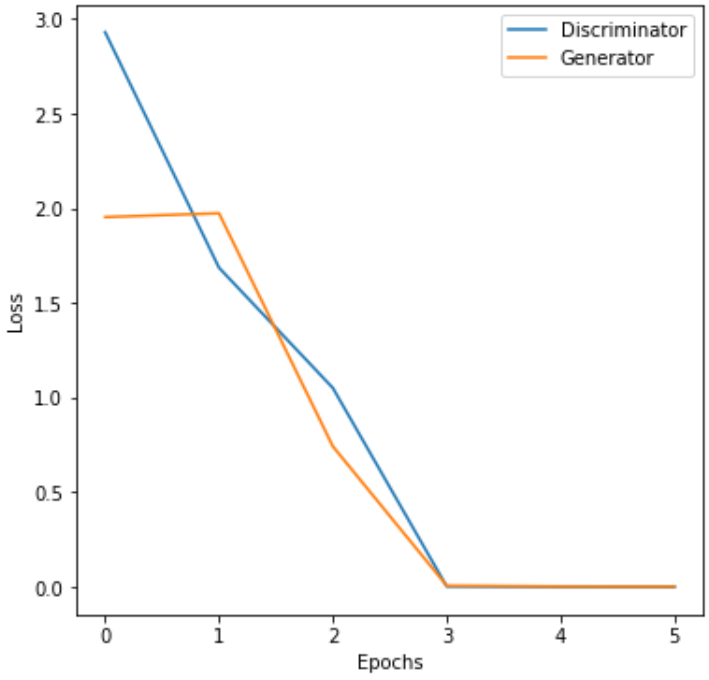

Figura 4: Convergência da *loss function* das redes

Após realizado treinamento, ao testar o modelo com a série temporal apresentada anteriormente, obteve-se o gráfico da Figura 5.

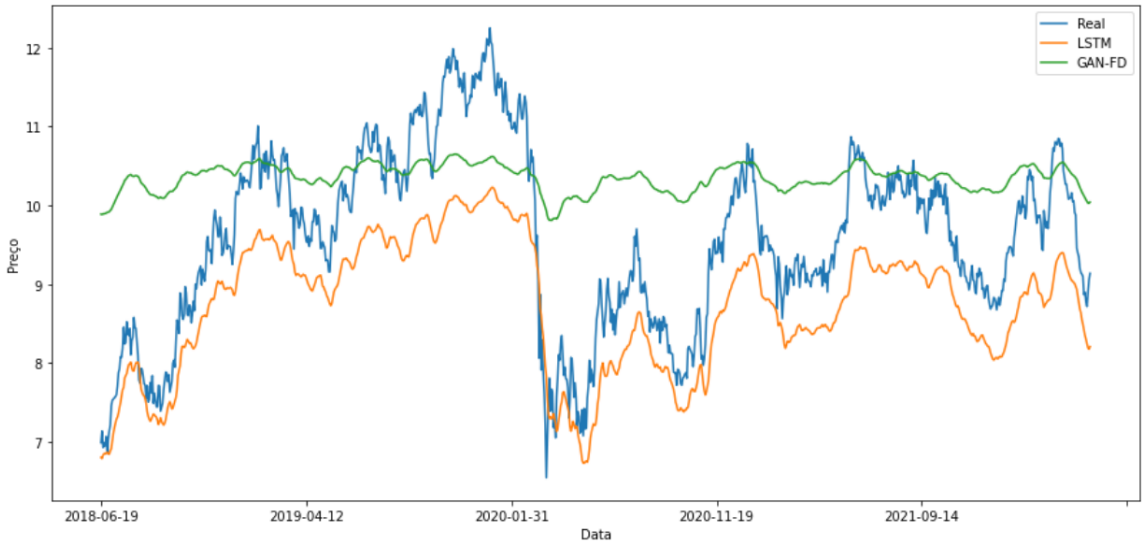

Figura 5: Comparativo de resultados

Da Figura 5, é possível verificar que a rede discriminante não melhora a performance da rede geradora. Pelo contrário, é possível ver que seu desempenho para servir como um modelo de previsão da direção ou de valor do preço é tão ruim quanto ou pior que a rede LSTM.

5 Conclusão

O desenvolvimento do estudo possibilita concluir que a abordagem de GANs da forma apresentada em por *Zhou et al*[1] adaptada ao mercado brasileiro não demonstrou eficácia em comparação com uma topologia mais simples.

Uma possível justificativa é que a rede discriminante dificulta o aprendizado da rede geradora, por isso a convergência foi tão rápida durante o treinamento. A rede discriminante cria uma janela onde a rede geradora é livre para estimar preços e, na topologia proposta pelo trabalho original, e essa janela acabou por ficar muito estreita.

Os possíveis próximos passos para o trabalho seriam aplicar técnicas de correção para o *gap* de fechamento, desenvolver e realizar o *backtesting* de uma estratégia utilizando a direção do preço apenas e não o valor previsto, realizar os mesmos testes mas

com diferentes topologias para as redes e verificar diferentes parâmetros dos indicadores e séries de preços.

Apêndice A *Features* do modelo

As *features* do modelo são:

- *Open Price (OP)*: preço de abertura extraído diretamente da base de dados com histórico de preços;
- *High Price (HP)*: preço máximo extraído diretamente da base de dados com histórico de preços;
- *Low Price (LP)*: preço mínimo extraído diretamente da base de dados com histórico de preços;
- *Bias*: obtido através da função ***.bias()*** do módulo Pandas-TA com os parâmetros padrões;
- *Bollinger Bands (BB)*: composto por três indicadores (Banda Alta, Baixa e Média), obtido através da função ***.bbands()*** do módulo Pandas-TA com os parâmetros padrões;
- *Directional Movement (DM)*: composto por dois indicadores (Positive e Negative), obtido através da função ***.dm()*** do módulo Pandas-TA com os parâmetros padrões;
- *Exponential Movement Average (EMA)*: obtido através da função ***.ema()*** do módulo Pandas-TA com os parâmetros padrões;
- *Simple Movement Average (SMA)*: obtido através da função ***.sma()*** do módulo Pandas-TA com os parâmetros padrões;
- *Stochastic Oscillator*: composto por dois indicadores (Rápido e Devagar), obtido através da função ***.stoch()*** do módulo Pandas-TA com os parâmetros padrões;
- *Moving Average Convergence and Divergence (MACD)*: composto por um indicador (Valor do MACD), obtido através da função ***.macd()*** do módulo Pandas-TA com os parâmetros padrões;

- *Relative Strength Index (RSI)*: obtido através da função ***.rsi()*** do módulo Pandas-TA com os parâmetros padrões;

Referências

[1] X. Zhou et al. “Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets”. Em: *Mathematical Problems in Engineering* 2018 (2018), pp. 1–11.

[2] Rami Al-Hmouz, Witold Pedrycz e Abdullah Balamash. “Description and prediction of time series: A general framework of Granular Computing”. Em: *Expert Systems with Applications* 42.10 (2015), pp. 4830–4839. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2015.01.060. URL: https://www.sciencedirect.com/science/article/pii/S0957417415000834.

[3] A. Marszałek e T. Burczyński. “Modeling and forecasting financial time series with ordered fuzzy candlesticks”. Em: *Information Sciences* 273 (2014), pp. 144–155. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2014.03.026. URL: https://www.sciencedirect.com/science/article/pii/S0020025514003107.

[4] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[5] François Chollet. *Deep Learning with Python*. Manning, 2017. Cap. 8. ISBN: 9781617294433.

[6] J. D. Hamilton. *Time Series Analysis*. Vol. 2. Princeton University Press, 1994.

[7] Jeffrey Marc Wooldridge. *Introductory Econometrics: A Modern Approach*. ISE - International Student Edition. South-Western, 2009. ISBN: 9780324581621.

[8] Yakup Kara, Melek Acar Boyacioglu e Ömer Kaan Baykan. “Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange”. Em: *Expert Systems with Applications* 38.5 (2011), pp. 5311–5319. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2010.10.027. URL: https://www.sciencedirect.com/science/article/pii/S0957417410011711.

[9] A. Prastyo, D. Junaedi e M. D. Sulistiyo. “Stock price forecasting using artificial neural network: (Case Study: PT. Telkom Indonesia)”. Em: *2017 5th International Conference on Information and Communication Technology (ICoIC7)*. 2017, pp. 1–6. DOI: 10.1109/ICoICT.2017.8074673.

[10] Alaa Sheta, Sara Ahmed e Hossam Faris. “A Comparison between Regression, Artificial Neural Networks and Support Vector Machines for Predicting Stock Market Index”. Em: *International Journal of Advanced Research in Artificial Intelligence* 4 (jul. de 2015), pp. 55–63. DOI: 10.14569/IJARAI.2015.040710.

[11] Wei Huang, Yoshiteru Nakamori e Shou-Yang Wang. “Forecasting stock market movement direction with support vector machine”. Em: *Computers Operations Research* 32.10 (2005). Applications of Neural Networks, pp. 2513–2522. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2004.03.016. URL: https://www.sciencedirect.com/science/article/pii/S0305054804000681.

[12] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[13] Johan Bollen, Huina Mao e Xiaojun Zeng. “Twitter mood predicts the stock market”. Em: *Journal of Computational Science* 2.1 (2011), pp. 1–8. ISSN: 1877-7503. DOI: https://doi.org/10.1016/j.jocs.2010.12.007. URL: https://www.sciencedirect.com/science/article/pii/S187775031100007X.

[14] László Nemes e Attila Kiss. “Prediction of stock values changes using sentiment analysis of stock news headlines”. Em: *Journal of Information and Telecommunication* 1 (2021), pp. 1–20. DOI: 10.1080/24751839.2021.1874252. eprint: https://doi.org/10.1080/24751839.2021.1874252. URL: https://doi.org/10.1080/24751839.2021.1874252.

[15] *GAN image example*. URL: https://miro.medium.com/max/1400/1*JwoQerKlHBO9B2FI4BUVjQ.png. (accessed: 16.08.2022).

[16] Sepp Hochreiter e Jürgen Schmidhuber. “Long Short-term Memory”. Em: *Neural computation* 9 (dez. de 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[17] François Chollet. *Deep Learning with Python*. Manning, 2017. Cap. 6. ISBN: 9781617294433.

[18] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016. Cap. 10.

[19] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016. Cap. 9.

[20] Katarzyna Janocha e Wojciech Marian Czarnecki. "On Loss Functions for Deep Neural Networks in Classification". Em: *CoRR* abs/1702.05659 (2017). arXiv: `1702.05659`. URL: `http://arxiv.org/abs/1702.05659`.

[21] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc., 2019. Cap. 4.

[22] *Pandas-TA*. URL: `https://github.com/twopirllc/pandas-ta`. (accessed: 12.10.2021).

[23] *train*$_t est_s plit$. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`. (accessed: 12.10.2021).