Revista Gestão & Tecnologia
e-ISSN: 2177-6652
revistagt@fpl.edu.br
http://revistagt.fpl.edu.br/

AVALIAÇÃO DE REDES NEURAIS PROFUNDAS PARA A PREVISÃO DE PREÇO DAS AÇÕES DA PETROBRÁS

EVALUATING DEEP NEURAL NETWORKS FOR PETROBRÁS' STOCK PRICE PREDICTION

EVALUACIÓN DE REDES NEURONALES PROFUNDAS PARA LA PREDICCIÓN DEL PRECIO DE LAS ACCIONES DE PETROBRÁS

Lidio Mauro Lima de Campos
Professor Adjunto III da Universidade Federal do Pará - UFPA na Faculdade de Computação / Instituto de Ciências Exatas e Naturais (ICEN). Doutor em Engenharia Elétrica pela UFPA
limadecampos@gmail.com
https://orcid.org/0000-0003-4315-829X

Yann Fabricio Cardoso De Figueiredo
Graduando em Bacharelado em Ciência da Computação pela Universidade Federal do Pará.
yann.fabricio@hotmail.com

Editor Científico: José Edson Lara
Organização Comitê Científico
Double Blind Review pelo SEER/OJS
Recebido em 12.07.2020
Aprovado em 24.08.2021

Este trabalho foi licenciado com uma Licença Creative Commons - Atribuição – Não Comercial 3.0 Brasil

RESUMO

Objetivo do Estudo: Na presente pesquisa investiga-se a acurácia das Redes Neurais Profundas (RNPs) com o intuito de encontrar o melhor modelo capaz de predizer os preços das ações da Petrobrás.

Metodologia/Abordagem: realizaram-se experimentos, com o uso de RNPs, variando-se uma série de parâmetros de treinamento das mesmas, ao final dos mesmos comparam-se as acurácias das RNPs e selecionam-se os melhores modelos capazes de predizer os preços das ações da Petrobrás.

Originalidade/Relevância: A originalidade da pesquisa diz respeito a utilização de RNPs para solução do problema, pois o modelo de predição com melhor acurácia é obtido por um processo de ajuste gradual dos parâmetros. A Proposta é relevante, pois permite predizer alterações dos preços das ações da Petrobrás, visando facilitar a tomada de decisão do investidor com previsões do preço das ações da empresa.

Principais Resultados: No geral, os melhores resultados quanto a predição de valores das ações ocorreu nas simulações com rede direta, enquanto a rede LSTM proporcionou um melhor cenário quanto as oscilações das ações na bolsa de valores. A rede direta mostrou-se eficiente por possui um erro menor nas métricas MAPE e RMSE, com 1.47% e 0.0147 respectivamente. No caso da rede LSTM foi possível observar um erro menor nas métricas MAE e MSE, com 0.0159 e 0.0005 respectivamente.

Contribuições teóricas/metodológicas: A pesquisa pode melhor evidenciar a capacidade preditiva modelos de RNPs para fins de estimação adequada de parâmetros, de modelos baseados em inteligência artificial, na tarefa de predição dos preços das ações da Petrobrás.

Palavras-chave: Redes Neurais Artificiais, Predição, Preço de ações, Resultados.

ABSTRACT

Objective of the Study: The present research investigates the accuracy of Deep Neural Networks (RNPs) in order to find the best model capable of predicting Petrobras' stock prices.

Methodology / Approach: Experiments were carried out, using RNPs, varying a series of training parameters, at the end of them they compare the accuracy of RNPs and select the best models capable of predicting prices Petrobras shares.

Originality / Relevance: The originality of the research concerns the use of RNPs to solve the problem, because the prediction model with better accuracy is obtained by a process of gradual adjustment of the parameters. The Proposal is relevant, as it allows predicting changes in the prices of Petrobras shares, in order to facilitate investor decision making with forecasts of the company's share price.

Main Results: In general, the best results regarding the prediction of stock values occurred in simulations with a direct network, while the LSTM network provided a better scenario regarding the oscillations of shares on the stock exchange. The direct network proved to be efficient because it had a smaller error in the metrics MAPE and RMSE, with 1.47% and 0.0147 respectively. In the case of the LSTM network, it was possible to observe a minor error in the MAE and MSE metrics, with 0.0159 and 0.0005 respectively.

Theoretical / methodological contributions: The research can better show the predictive capacity of RNP models for purposes of adequate parameter estimation, of models based on artificial intelligence, in the task of predicting Petrobras' stock prices.

Keywords: Artificial Neural Networks, Prediction, Share price, Results.

RESUMEN

Objetivo del estudio: La presente investigación investiga la precisión de las Redes neuronales profundas (RNP) para encontrar el mejor modelo capaz de predecir los precios de las acciones de Petrobras.

Metodología / Enfoque: se llevaron a cabo experimentos con el uso de RNP, variando una serie de parámetros de entrenamiento, al final de ellos comparan la precisión de los RNP y seleccionan los mejores modelos capaces de predecir precios Acciones de Petrobras.

Originalidad / Relevancia: La originalidad de la investigación se refiere al uso de RNP para resolver el problema, ya que el modelo de predicción con mayor precisión se obtiene mediante un proceso de ajuste gradual de los parámetros. La propuesta es relevante, ya que permite predecir cambios en los precios de las acciones de Petrobras, con el fin de facilitar la toma de decisiones de los inversores con pronósticos del precio de las acciones de la compañía.

Resultados principales: en general, los mejores resultados con respecto a la predicción de los valores de las acciones se produjeron en simulaciones con la red directa, mientras que la red LSTM proporcionó un mejor escenario con respecto a las oscilaciones de las acciones en la bolsa de valores. La red directa demostró ser eficiente porque tenía un error menor en las métricas MAPE y RMSE, con 1.47% y 0.0147 respectivamente. En el caso de la red LSTM, fue posible observar un error menor en las métricas MAE y MSE, con 0.0159 y 0.0005 respectivamente.

Contribuciones teóricas / metodológicas: la investigación puede mostrar mejor la capacidad predictiva de los modelos RNP para propósitos de estimación adecuada de parámetros, de modelos basados en inteligencia artificial, en la tarea de predecir los precios de las acciones de Petrobras.

Palabras clave: Redes Neuronales Artificiales, Predicción, Precio de la acción, Resultados.

1 INTRODUÇÃO

A Petróleo Brasileiro S.A. (Petrobras) é uma empresa estatal brasileira de capital aberto, sendo seu maior acionista a União Federal. Citam-se como principais ramos de negócio da empresa: petróleo, gás e energia, sendo incumbência da empresa, a atuação na exploração e/ou refino de petróleo, biocombustíveis e gás. A Petrobrás possui seu capital social composto por ações ordinárias e preferenciais, fazendo uso dos códigos PETR3 (ON) e PETR4 (PN) (Bovespa, 2020) na Bolsa de Valores.

Segundo Jesus e Sarti (2016) a política de dividendos da Petrobrás, prevista em seu estatuto, determina que o valor mínimo a ser distribuído é de 25% do lucro líquido ajustado, tanto para ações ordinárias (ON), quanto preferenciais (PN). Priorizando as PN, no caso de reembolso do capital e no recebimento dos dividendos de, no mínimo, 5% calculado sobre parcela do capital representada por essa categoria de ações, ou de 3% do valor do patrimônio líquido da ação, preponderando sempre o maior. Desde o ano 2000, embora o estatuto da companhia garanta (i) maiores dividendos para as ações PN, os pagamentos eram idênticos para ambas as categorias, e (ii) um valor mínimo a ser distribuído, a remuneração se dava de forma mais generosa do que a prevista.

Refletindo o corte de dividendos, tanto as ações ON (PETR3), quanto as ações PN (PETR4), despencaram. A trajetória declinante da ação da Petrobras fez com que houvesse uma redução de seu valor de mercado. No fechamento de 30 de dezembro de 2013, em valores de 2012, o valor de mercado da empresa era de R$ 202.708 milhões, correspondente a 61,46% do valor do patrimônio líquido de R$ 329.840 milhões (Jesus & Sarti, 2016). A Figura 1, mostra o gráfico de evolução da Ação da Petrobras (PETR4.SA), no período de Janeiro de 2013 a Janeiro de 2019.

Os sistemas baseados em inteligência artificial que fazem predições, podem ser enquadrados, como de apoio à decisão, que otimizam modelos que fornecem subsídios para operações de compra e venda de uma ação, visando dar suporte ao usuário durante o investimento.

Figura 1. Ação Petrobrás (PETR4.SA) - Jan 2013 a Jan 20119
Fonte: Petróleo Brasileiro S.A. - Petrobras (PETR4.SA). (2020). *Yahoo Finanças*. Recuperado em 19 maio, 2020, de https://br.financas.yahoo.com/quote/PETR4.SA/history?p=PETR4.SA

As redes neurais artificiais (RNAs) são ferramentas bastante eficientes para o reconhecimento de padrões. As mesmas têm sido utilizadas de forma crescente para em diversos ramos de negócios: predição de séries temporais (Almuammar & Fasli, 2019), reconhecimento humano usando a medida biométrica (Dasgaonkar & Chopade, 2018; Langner, Ahlström, & Kullberg, in press; Sharma, Agarwal, & Pandey, 2018), predição de radiação da energia solar (Shamshirband, Rabczuk, & Chau, 2019), visão por computador (Gustafsson, Danelljan, & Schön, in press; Poonia, Tiwari, & Mishra, 2018), o reconhecimento de voz (Ning, He, Wu, Xing, & Zhang, 2019), tradução automática (Sutskever, Vinyals, & Le, 2014), tomada de decisão no mercado financeiro (Gambogi, 2013), sistema de reconhecimento de vídeo (Kaushik, Gupta, & Bhatia, 2018), visão por computador (Poonia *et al.*, 2018), sistema de reconhecimento de vídeo (Kaushik *et al.*, 2018), análise de crédito bancário (Steiner *et al.*, 2004), tomada de decisão (Garcia & Campos, 2019).

As redes neurais recorrentes (RNNs) são adequadas para problemas de aprendizado supervisionado, nos quais o conjunto de dados tem uma natureza sequencial. A previsão de séries temporais não deve ser uma exceção. RNNs são redes essencialmente neurais com memória. Eles podem se lembrar de coisas do passado, o que é obviamente útil para prever alvos dependentes do tempo. No entanto, aplicá-los à previsão de séries temporais não é uma tarefa trivial.

Um tipo especial de modelo RNN é a Memória de Longo Prazo Redes (LSTM), através das quais as relações entre os dados de entrada e saída são modelados. Esses modelos baseados na RNN, são capazes de aprender com dados passados, nos quais as várias portas arquitetura de rede são empregadas para lembrar os dados passados e, assim, construir o modelo prospectivo em relação aos dados passados e atuais. Portanto, os dados de entrada são atravessados apenas uma vez (ou seja, da esquerda (entrada) para a direita (saída)).

O objetivo desse trabalho é apresentar os resultados da utilização de RNAs profundas (Deep Neural Networks) na tarefa de predição do preço de ações da Petrobrás. Na realização dos experimentos foram utilizadas diferentes arquiteturas de RNAs, dentre elas a *feedfoward*, a rede e a rede recorrente. Para a rede *feedfoward* utilizou-se o algoritmo *backpropagation* com 4 camadas e para a rede recorrente utilizou-se o algoritmo Long Short-Term Memory (LSTM). Assim sendo, nesta pesquisa é proposto um sistema de apoio à decisão que utiliza redes neurais profundas em séries de cotações históricas de ações da Petrobrás para prever valores futuros da mesma.

Para que os objetivos da pesquisa sejam alcançados, na seção 2 faz-se a revisão da literatura, na seção 3 apresentam-se conceitos sobre RNAs, na seção 4 detalham-se os algoritmos utilizados e as métricas utilizadas nas simulações. Na seção 5, apresentam-se a execução das etapas da metodologia e os resultados de simulação obtidos e finalmente nas seções 6, 7 e 8 as conclusões, agradecimentos e as referências bibliográficas consultadas.

2 REVISÃO DE LITERATURA

O objetivo do artigo de (Santos, Lucas, Silva, & Medeiro, 2015), foi aferir causalidade entre a ação preferencial da Petrobrás (PETR4) com o mercado futuro de commodities de petróleo (contratos com primeiro vencimento CL1) e o índice futuro do S&P500 (contratos com primeiro vencimento, SP1).

O objetivo do artigo de (Marques, 2012) foi testar três densidades de probabilidade para prever as probabilidades dos retornos das ações (PETR4) da Petrobras negociadas na BOVESPA a partir dos preços diários de fechamento no período de 26.06.2000 a 17.04.2012. As densidades candidatas foram a Normal, a Logística e a Cauchy. O teste de aderência de Kolmogorov-Smirnov sugere que a densidade Logística é a que melhor descreve a probabilidade dos retornos das ações da Petrobras. Os resultados indicam que o investidor

pode obter uma rentabilidade diária positiva com probabilidade de 0,5130271 e uma rentabilidade maior do que 5% ao mês com uma probabilidade de 0,4698542.

O objetivo do trabalho de (Jesus & Sarti, 2016) foi analisar por que a política de conteúdo local está na contramão da perspectiva de maximização de valor para os acionistas da Petrobras, mas, não do escopo de atuação de uma empresa com a sua natureza jurídica. A principal conclusão é que a ineficiência gerada pelo direcionamento das aquisições é inconsistente com a atual dinâmica de acumulação das grandes corporações, mas, não com o escopo de atuação de uma empresa com a natureza jurídica da Petrobras.

3 REDES NEURAIS ARTIFICIAIS

3.1 Redes Neurais Multi Layer Perceptron

RNAs são inspiradas biologicamente no funcionamento do cérebro, assim uma rede neural típica e constituída por um conjunto de neurônios interligados, influenciando uns aos outros formando um sistema maior capaz de reconhecer padrões por meio de treinamento. O algoritmo de aprendizado generaliza os dados e memoriza o conhecimento dentro dos parâmetros adaptáveis da rede, denominado de pesos.

Uma Rede Direta, como ilustrado na Figura 2, consiste em várias camadas compostas por nós (neurônios) onde cada neurônio de uma camada possui ligação com todos os neurônios da camada seguinte. Conforme visto na Figura 2, a rede direta, denominada *perceptron* multicamada, possui uma camada de entrada, que atua como um conjunto de sensores da rede captando os estímulos do ambiente e pode ter uma ou mais camadas intermediárias que é onde a maior parte do processamento é realizada através das conexões e seus pesos respectivos, podem ser considerados como extratoras de características e uma camada de saída onde o resultado final é concluído e apresentado. Ainda na Figura 2 não existem conexões entre a saída de um neurônio e algum outro neurônio localizado em uma camada anterior ao primeiro, ou seja, não possui ciclos, fato que caracteriza uma rede *feedforward*.

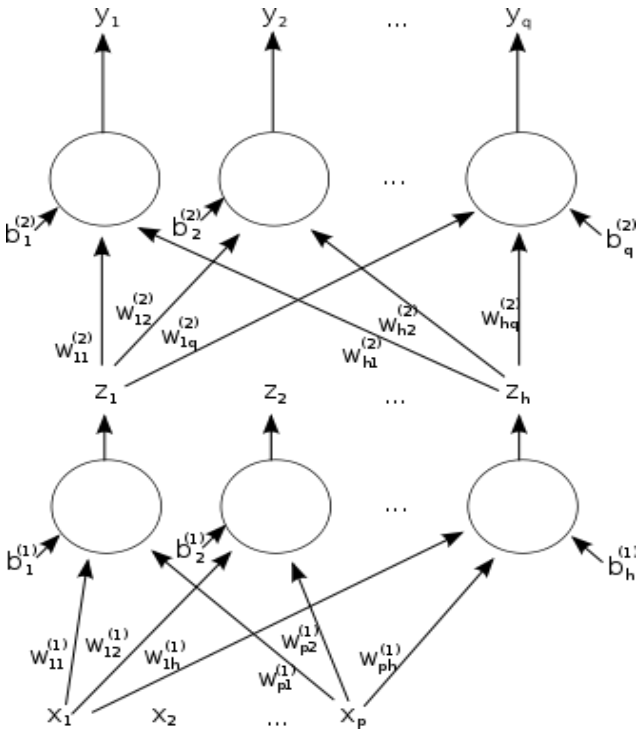

Figura 2. **Rede Perceptron Multicamada**
Fonte: Rede neural artificial, (2018). In *Wikipedia enciplopedia livre*. Recuperado em 01 junho, 2018, de https://pt.wikipedia.org/wiki/Rede_neural_artificial

A configuração pela qual os neurônios de uma RNA e estruturada depende do algoritmo de aprendizagem a ser utilizado para o treinamento. No presente trabalho é utilizada uma rede neural multicamadas compostas de camadas alinhadas e totalmente conectadas diretamente com uma camada de neurônios. Neste tipo de rede, as entradas são apresentadas na primeira camada, que é chamada camada de entrada. Esta distribui as informações de entrada para a camada oculta da rede, que por sua vez pode possuir uma ou mais camadas ocultas e uma camada de saída (Kovács, 2006). O valor de saída é obtido através da sequência de funções de ativação definido na camada oculta.

A aprendizagem é uma das características notáveis das RNAs. No aprendizado supervisionado, o treinamento possui um conhecimento sobre os dados, uma especie de instrutor que confere o quanto a rede está próxima de uma solução aceitável, este conhecimento, está representado sob forma de um conjunto de amostras de *entrada-saída*. O processo de treinamento modifica os pesos das RNAs com a finalidade de melhorar um

determinado critério de desempenho, de tal forma que, para o conjunto de entrada informado, a rede seja capaz de calcular uma saída o mais próximo possível da saída desejada.

A aprendizagem de Redes Multilayer Perceptron (MLP) e um processo iterativo, conhecido como aprendizagem por experiência, no qual padrões de treinamento (exemplos) são apresentados a rede e com base nos erros obtidos, são realizados ajustes nos pesos sinápticos, com o intuito de diminuir o erro nas próximas iterações. Sua estrutura é formada pela camada de entrada, que atua como os sensores da rede captando os estímulos do ambiente e pode ter uma ou mais camadas intermediárias que é onde a maior parte do processamento é realizada através das conexões e seus pesos respectivos, podem ser considerados como extratoras de características e uma camada de saída onde o resultado final é concluído e apresentado.

O treinamento de um Perceptron de Múltiplas Camadas (MLP) consiste em ajustar os pesos e os *thresholds* (bias) de suas unidades para que a classificação desejada seja obtida. Quando um padrão é inicialmente apresentado à rede, ela produz uma saída e, após medir a diferença entre a resposta atual e a desejada, são aplicados ajustes apropriados nos pesos de modo a reduzir esta distância. Este procedimento é conhecido como Regra Delta. O algoritmo mais utilizado para o treinamento destas redes MLP é uma generalização da Regra Delta denominada de Backpropagation.

O Backpropagation é baseado na regra de aprendizagem por correção de erro. Durante o treinamento com o algoritmo *Backpropagation*, a rede opera em uma sequência de dois passos. No primeiro, um padrão é apresentado à camada de entrada da rede. O sinal resultante se propaga através da rede, camada por camada, até que a resposta seja produzida pela camada de saída. No segundo passo, a saída obtida é comparada à saída desejada para esse padrão particular. Se esta não estiver correta, o erro é calculado. Este erro é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas vão sendo modificados à medida que o erro é retropropagado. Ao utilizar a retro propagação o algoritmo aprende de forma supervisionada, em tempo discreto e utiliza um método de gradiente descendente para correção de erro, assim a codificação executa um mapeamento entrada-saída, através da minimização de uma função custo. Está por sua vez, realiza iterativamente ajustes nos pesos sinápticos de acordo com o erro quadrático acumulado para o conjunto de treinamento. O processo de evolução da redução gradativa de

erro pode levar a convergência. A medida que a rede aprende, o valor do erro converge para um valor estável, assim o processo prossegue, até o critério de um valor mínimo de erro global seja atingido ou o número de épocas (interações) seja alcançado. Os passos do algoritmo de retro propagação são descritos abaixo.

Inicializar os pesos da rede. Cada peso deve ser ajustado aleatoriamente para um número entre –0.1 e 0.1.

$w1_{ij}$ =aleatório(-0.1;0.1), $w2_{ij}$ =aleatório(-0.1;0.1) para todo i=0,.....,A; j=1,.....,B; para todo i=0,.....,B; j=0,.....,C.

Inicializar as ativações das unidades limite. Seus valores nunca mudam.

x_o =1.0 e h_o =1.0

Escolher um par de padrão de entrada-saída. Supondo que o vetor de entrada seja x_i e o vetor de saída seja y_j . Atribuem-se níveis de ativação às unidades de entrada.

Propagar a ativação das unidades da camada de entrada para as unidades da camada oculta, usando a função de ativação.

$$h_j = \frac{1}{1+e^{-\sum_{i=0}^{A} w1ij.xi}} \text{ , para todo j=1,...,B} \quad (1)$$

Propagam-se as ativações das unidades da camada oculta para as unidades da camada de saída, usando a função de ativação.

$$o_j = \frac{1}{1+e^{-\sum_{i=0}^{B} w2ij.xi}} \text{ para todo j=1,.....,C} \quad (2)$$

Computar os erros das unidades da camada de saída, denotados por $\partial 2_j$. Os erros baseiam-se na saída real da rede (o_j) e na saída (y_j).

$$\delta 2_j = o_j(1-o_j)(y_j - o_j) \text{ para todo j=1,.....,C} \quad (3)$$

Computar os erros das unidades da camada oculta, denotados por δ1j.

$$\delta 1_j = h_j(1-h_j)\sum_{1=0}^{C}\delta 2_i.w2_{ij} \quad \text{para todo j=1,.....,B} \qquad (4)$$

Ajuste dos pesos entre a camada oculta e a camada de saída. O coeficiente de aprendizagem é denotado por η, sua função é a mesma de na aprendizagem por perceptrons.

$$\Delta w2_{ij}(t+1) = \Delta w2_{ij}(t) + \eta\delta 2_j.h_i \quad \text{para todo i=0,.....,B; j=1,.....,C} \qquad (5)$$

Ajuste os pesos entre a camada de entrada e a camada oculta

$$\Delta w1_{ij}(t+1) = \Delta w1_{ij}(t) + \eta\delta 1_j.x_i \quad \text{para todo i=0,.....,A; j=1,.....,B} \qquad (6)$$

Vá para a etapa 4 e repita. Quando todos os pares entrada-saída tiverem sido apresentados à rede, uma época terá sido completada. Repita as etapas de 4 a 10 para tantas épocas quantas forem desejadas.

As equações (7) e (8) mostram as fórmulas para atualização dos pesos, da camada de saída e da camada intermediária, respectivamente.

$$\Delta w2_{km}(t+1) = \Delta w2_{km}(t) + \eta\,[(y_{nm} - o_m)(1-o_m).o_m].h_k \qquad (7)$$

$$\Delta w1_{jk}(t+1) = \Delta w1_{jk}(t) + \eta\,[(1-h_k)h_k.\sum_{m=0}^{C}(y_{nm} - o_m)(1-o_m).o_m.w2_{km}]x_{nj} \qquad (8)$$

O interesse em redes de *feedforward* profundas (*deep feedforward networks*) foi revivido por volta de 2006 (Bengio, Lamblin, Popovici, & Larochelle, 2006) por um grupo de pesquisadores reunidos pelo Instituto Canadense de Pesquisa Avançada (CIFAR).

O algoritmo de retropropagação, descrito anteriormente, não é capaz de implementar mapeamentos dinâmicos. A questão, é como estender a estrutura das redes MLP – *Multi Layer Perceptron* – para que as mesmas possam mapear um comportamento variante no tempo, sendo assim capaz de tratar sinais temporais. O que significa oferecer características dinâmicas ao mapeamento realizado pela rede, tornando-a sensível a sinais que variem com o tempo. Para uma RNA ser considerada dinâmica, é preciso que possua memória. A forma

abordada nessa pesquisa, para incorporar memória a RNA, foi a utilização de RNNs, tais como a BPTT. A Figura 3 mostra uma arquitetura de RNA recorrente.

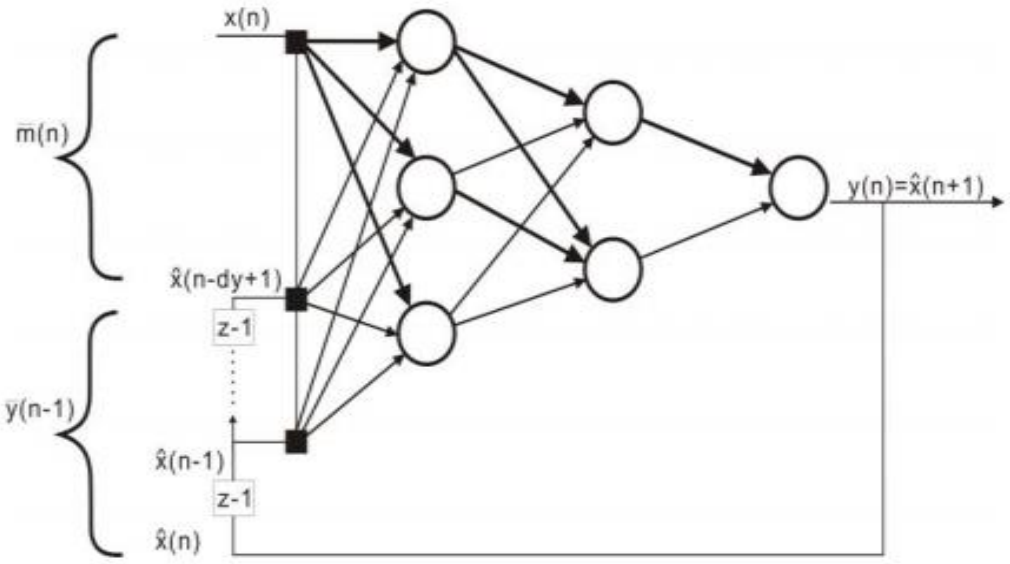

Figura 3. Estrutura da rede recorrente

3.2 Redes LSTM

As redes LSTM são um modelo de RNA recorrente e profundo, e elas têm a função de oferecer um melhor desempenho ao resolver o problema do desaparecimento de gradiente, algo comum quando se refere a uma rede recorrente lidando com grandes sequências de dados, e ainda é responsável por endereçar o problema da capacidade de lembrar relações por um longo período de tempo. O desaparecimento de gradiente é bem comum de ser encontrado no treinamento de RNNs, principalmente quando é muito longa a sequência de dados na entrada. Caso o sinal do erro seja pequeno, ao ser propagado durante o Backpropagation Through Time (BPTT), o gradiente tende a ir ficando menor e consequentemente prejudicando a atualização de pesos, deixando-a pequena, e fazendo com que o aprendizado da rede seja menos efetivo. Para tratar esse problema do desaparecimento do gradiente as redes LSTM fazem com que o fluxo de erro seja constante, isso é feito através de unidades especiais conhecidas como “portões”, responsáveis por ajustar os pesos da mesma maneira que o truncamento da sequência quando o dado não é necessário, caracterizado um esquecimento.

As redes LSTM possuem, assim como as redes recorrentes padrões (Figura 3), uma estrutura de cadeia com módulos de repetição, mas no caso da LSTM há a presença de 4 camadas de RNA para interação. A LSTM é capaz de adicionar ou remover dados ao estado de uma célula, e são regulados de forma cuidadosa por estruturas conhecidas como portas. Essas estruturas, as portas, são uma maneira opcional de deixar passar determinadas informações na rede, e elas são compostas por uma camada de rede neural sigmóide e uma operação de multiplicação pontual. A camada sigmóide faz a produção de números entre zero e um, descrevendo a quantidade de informação que deve ser liberada de cada componente. O zero indica que não deve passar nada, enquanto o um é o oposto, ou seja, deve passar tudo.

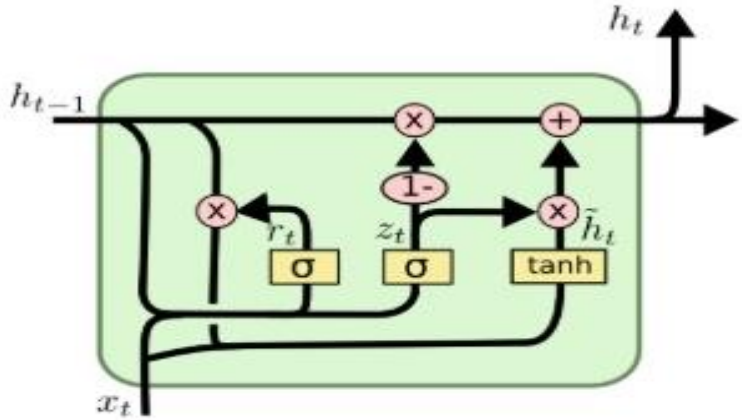

Figura 4. Estrutura da rede LSTM
Fonte: Junior, J. R. F. (2019, junho 11). *Redes neurais recorrentes – LSTM*. Recuperado em 20 maio, 2020, de https://medium.com/@web2ajax/redes-neurais-recorrentes-lstm-b90b720dc3f6

A estrutura da rede LSTM, como pode ser visto na Figura 4, é composta por quatro redes neurais e por várias células de memória. As células são responsáveis por reter a informação, enquanto os portões fazem o gerenciamento de memória. Em uma rede LSTM há a presença de três portões, sendo eles: Forget Gate, responsável pela tarefa de definir quais informações são utéis e quais não são para continuar no estado da célula; Input Gate, responsável por adicionar no estado da célula as informações uteis selecionadas; Output Gate, Responsável por extrair as informações uteis da célula para apresentar como uma saída. As setas em loop no estado da célula (memória de longo prazo) evidenciam a recursividade da mesma, permitindo assim que as informações dos intervalos anteriores sejam guardadas na célula LSTM. O processo dentro da célula ocorre com o estado dela sendo modificado pelo forget gate colocado abaixo do estado da célula, sendo que também ocorre um ajuste pela

porta de modulação presente na entrada da célula. Desta equação exposta, o estado da célula anterior sofre um esquecimento, é multiplicada com a forget gate e soma mais informações por meio da saída das portas de entrada.

Quanto a aspectos matemáticos tratando-se de redes recorrentes LSTM, é possível formular a dinâmica destes tipos de rede por transições determinísticas dos estados ocultos anteriores até os atuais. A dinâmica exposta é representada pela fórmula:

$$RNN: h_t^{l-1}, h_{t-1}^l \rightarrow h_t^l \quad (9)$$

O processo de uma rede LSTM começa com a determinação de quais informações não são importantes para permanecer nas células de memória e, portanto, serão ocultadas, e como já mencionado esse procedimento ocorre com uma função de ativação sigmoide associada ao forget gate. A fórmula para expressar essa primeira etapa é:

$$f_t = \sigma(W_f . [h_{t-1}, X_t] + b_f) \quad (10)$$

onde, f_t é o forget gate, σ é a função sigmoide, $W_f \, e \, b_f$ são as matrizes de peso e bias respectivamente, e ainda h_{t-1} é a saída da última unidade LSTM no tempo t-1 e X_t é a entrada atual.

O passo posterior é tratar de decidir quais informações armazenar no estado da célula. O procedimento começa pelo portão de entrada que decide os valores a serem atualizados. Posteriormente uma camada, contendo tangente hiperbólica, cria um vetor de novos valores candidatos (C_t) a serem adicionados ou não no estado da célula. As duas saídas devem ser combinadas para aplicar a atualização no estado da célula. Em seguida o processo recomeça com os mesmos passos até chegar novamente na camada da tangente hiperbólica, onde são atribuídos pesos aos valores que passam para decidir seu nível importância (-1 a 1). As saídas desse recomeço de passos são multiplicadas para atualizar o estado da célula. A nova memória gerada é adicionada à memória antiga C_{t-1}, resultando assim em C_t. As fórmulas que expressam este segundo passo são:

$$i_t = \sigma(W_i . [h_{t-1}, X_t] + b_i \quad (11)$$

$$M_t = \tanh(W_m . [h_{t-1}, X_t] + b_m \quad (12)$$

$$C_t = f_t . C_{t-1} + i_t . M_t \quad (13)$$

onde, C_t e C_{t-1} são estados da célula no tempo t e t-1, e ainda W e b são matrizes de pesos e bias da célula.

No último passo, a saída LSTM (h_t) é calculada baseada no estado da célula filtrada (o_t). Um sigmoide é calculado para ser decidido quais partes do estado da célula chegam na saída. A saída do estado da célula filtrada é multiplicada pelos novos valores gerados pela camada de tangente hiperbólica do estado da célula (C_t), com um valor variando entre -1 e 1. As fórmulas a seguir expressam esse último passo:

$$o_t = \sigma(W_o.[h_{t-1}, X_t] + b_o) \quad (14)$$

$$h_t = o_t.\tanh(C_t) \quad (15)$$

4 MATERIAIS E MÉTODOS

Para a realização das simulações foram usados algoritmos de redes neurais com arquitetura de rede direta e recorrente (LSTM), sendo ambos desenvolvidos com o auxílio de bibliotecas e frameworks populares em *machine learning* e *deep learning*. Foram utilizados, por exemplo: o keras, com as funções de treinamento e teste da base de dados; numpy, com as funções matemáticas para auxiliar no cálculo e armazenamento adequado dos dados no algoritmo; pandas, com as funções de tratamento de dados; e matplotlib, com a função de disponibilizar os resultados em forma de gráficos. Quanto aos métodos utilizados para avaliar os resultados foram utilizadas as seguintes métricas para medir erros: MAE, MSE, MAPE e RMSE.

4.1 Algoritmos RNA de rede direta e rede LSTM

Nessa seção apresenta-se o código desenvolvido no keras, que implementa uma rede neural direta com 4 camadas, utilizando os frameworks, citados na seção 4.

```
#RNA DE REDE DIRETA

import keras
import numpy as np
import pandas as pd
from keras import backend as K
import tensorflow as  tf
```

```
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, Flatten
from sklearn.preprocessing import MinMaxScaler

base = pd.read_csv("dataset/petr4-treinamento.csv")
base = base.dropna() #Remove as linhas onde há colunas com valores faltantes

#Normalização de dimensionamento de recursos
base_treinamento = base.iloc[:, 1:2].values
normalizador = MinMaxScaler(feature_range=(0, 1))
base_treinamento_normalizada = normalizador.fit_transform(base_treinamento)

#Criando uma estrutura de dados com 90 timesteps e 1 output
x_treino = []
y_treino = []

for i in range(90, 1242):
        x_treino.append(base_treinamento_normalizada[i-90:i, 0])
        y_treino.append(base_treinamento_normalizada[i, 0])

x_treino, y_treino = np.array(x_treino), np.array(y_treino)

#Remodelando
x_treino = np.reshape(x_treino, (x_treino.shape[0], x_treino.shape[1], 1))

### Predição na base de teste ###

base_teste = pd.read_csv('dataset/petr4-teste.csv')
preco_real_teste = base_teste.iloc[:, 1:2].values

base_completa = pd.concat((base['Open'], base_teste['Open']), axis=0)

entradas = base_completa[len(base_completa) - len(base_teste)-90:].values
entradas = entradas.reshape(-1, 1)
entradas = normalizador.transform(entradas)

x_teste = []
for i in range(90, 112):
   x_teste.append(entradas[i-90:i, 0])

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

#RMSE
def root_mean_squared_error(y_true, y_pred):
   return K.sqrt(K.mean(K.square(y_pred - y_true), axis=-1))

#MAPE
def mape(y_true, y_pred):
```

```
    if not K.is_tensor(y_pred):
        y_pred = K.constant(y_pred)
    y_true = K.cast(y_true, y_pred.dtype)
    diff = K.mean(K.abs((y_pred - y_true)) / K.mean(K.clip(K.abs(y_true),
                                            K.epsilon(),
                                            None)))
    return 100. * K.mean(diff)

#Criando modelo
model = Sequential()

#Adicionando camadas LSTM e alguma regularização de Dropout
model.add(Dense(50, input_shape=(x_treino.shape[1], 1), activation='relu'))
model.add(Dense(60, activation='relu'))
model.add(Flatten())
model.add(Dense(1, activation='relu'))

adam = keras.optimizers.Adam(learning_rate=0.01, beta_1=0.9, beta_2=0.999, amsgrad=False)
#sgd = keras.optimizers.SGD(momentum=0.1, lr=0.1, decay=0.1, nesterov=False)

model.compile(optimizer = adam, loss='mean_squared_error', metrics=['mean_absolute_error'])
#MAE
#model.compile(optimizer = adam, loss='mean_squared_error', metrics=['mean_squared_error'])
#MSE
#model.compile(optimizer = adam, loss='mean_squared_error',
metrics=['mean_absolute_percentage_error']) #MAPE
#model.compile(optimizer = adam, loss='mean_squared_error',
metrics=[root_mean_squared_error]) #RMSE

model.fit(x_treino, y_treino, nb_epoch=1000, batch_size=32)

preco_predito = model.predict(x_teste)
preco_predito = normalizador.inverse_transform(preco_predito)

#Evaluate the model
scores = model.evaluate(x_treino, y_treino)
print("%s: %.4f" % (model.metrics_names[1], scores[1]))

#Plotando resultados
plt.plot(preco_real_teste, color = 'red', label = 'Preco real')
plt.plot(preco_predito, color = 'green', label = 'Preco predito')
plt.title('Predicao do preco de acoes da Petrobras')
plt.legend()
plt.show()
```

Figura 5. Código Fonte da Rede Direta Multicamadas utilizando o Keras/Tensorflow

A seguir apresenta-se o código desenvolvido no keras, que implementa uma rede neural recorrente LSTM, utilizando-se os frameworks, citados na seção 4.

```
#RNA DE REDE RECORRENTE LSTM

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM
from sklearn.preprocessing import MinMaxScaler

#Leitura da base de dados
base = pd.read_csv("dataset/petr4-treinamento.csv")
base = base.dropna() #Remove as linhas onde ha colunas com valores faltantes

#Normalizacao de dimensionamento de recursos
base_treinamento = base.iloc[:, 1:2].values
normalizador = MinMaxScaler(feature_range=(0, 1))
base_treinamento_normalizada = normalizador.fit_transform(base_treinamento)

#Criando uma estrutura de dados com 90 timesteps e 1 output
x_treino = []
y_treino = []

for i in range(90, 1242):
        x_treino.append(base_treinamento_normalizada[i-90:i, 0])
        y_treino.append(base_treinamento_normalizada[i, 0])

x_treino, y_treino = np.array(x_treino), np.array(y_treino)

#Remodelando
x_treino = np.reshape(x_treino, (x_treino.shape[0], x_treino.shape[1], 1))

### Predicao na base de teste ###

base_teste = pd.read_csv('dataset/petr4-teste.csv')
preco_real_teste = base_teste.iloc[:, 1:2].values

base_completa = pd.concat((base['Open'], base_teste['Open']), axis=0)

entradas = base_completa[len(base_completa) - len(base_teste)-90:].values
entradas = entradas.reshape(-1, 1)
entradas = normalizador.transform(entradas)

x_teste = []
for i in range(90, 112):
   x_teste.append(entradas[i-90:i, 0])

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

#RMSE
```

```
def root_mean_squared_error(y_true, y_pred):
    return K.sqrt(K.mean(K.square(y_pred - y_true), axis=-1))

#MAPE
def mape(y_true, y_pred):
    if not K.is_tensor(y_pred):
        y_pred = K.constant(y_pred)
    y_true = K.cast(y_true, y_pred.dtype)
    diff = K.mean(K.abs((y_pred - y_true)) / K.mean(K.clip(K.abs(y_true),
                                            K.epsilon(),
                                            None)))
    return 100. * K.mean(diff)

#Criando modelo
model = Sequential()

#Adicionando camadas LSTM e alguma regularizacao de Dropout
model.add(LSTM(units=50,return_sequences=True,input_shape=(x_treino.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50))
model.add(Dropout(0.2))
model.add(Dense(units=1, activation='linear'))

adam = keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, amsgrad=False)
#sgd = keras.optimizers.SGD(momentum=0.1, lr=0.1, decay=0.1, nesterov=False)

model.compile(optimizer = adam, loss='mean_squared_error', metrics=['mean_squared_error'])
#MSE
#model.compile(optimizer = adam, loss='mean_squared_error', metrics=['mean_absolute_error'])
#MAE
#model.compile(optimizer = adam, loss='mean_squared_error',
metrics=['mean_absolute_percentage_error']) #MAPE
#model.compile(optimizer = adam, loss='mean_squared_error',
metrics=[root_mean_squared_error])

#RMSE

model.fit(x_treino, y_treino, epochs=100, batch_size=32)

preco_predito = model.predict(x_teste)
preco_predito = normalizador.inverse_transform(preco_predito)

#Evaluate the model
scores = model.evaluate(x_treino, y_treino)
print("%s: %.4f" % (model.metrics_names[1], scores[1]))

#Plotando resultados
plt.plot(preco_real_teste, color = 'black', label = 'Preco real')
plt.plot(preco_predito, color = 'green', label = 'Preco predito')
```

```
plt.title('Predicao do preco de acoes da Petrobras')
plt.legend()
plt.show()
```

Figura 6. Código Fonte da Rede LSTM utilizando o Keras/Tensorflow

4.2 Métricas utilizadas

As métricas utilizadas para avaliar os resultados obtidos na predição da base de dados são a Mean Absolute Error (MAE), Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE) e Root Mean Squared Error (RMSE). As métricas são formuladas como mostra abaixo:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |p_i true - p_i forecast| \quad (16)$$

$$MSE = \frac{1}{n}\sum(true - forecast)^2 \quad (17)$$

$$MAPE = \frac{100\%}{N}\sum_{I-1}^{N} \left|\frac{p_i true - p_i forecast}{p_i true}\right| \quad (18)$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_i true - p_i forecast)^2} \quad (19)$$

5 EXPERIMENTOS REALIZADOS

Os experimentos visaram variar, por conta de ter alcançado melhores resultados, a quantidade de neurônios da camada interna e o número de épocas com mais frequência. Os resultados com essa metodologia foram mais satisfatórios. Já quando foram variados a função de ativação e o otimizador, os resultados foram bem inferiores.

5.1 Simulações com a rede LSTM

As simulações com a rede LSTM se mostraram mais eficientes em predizer corretamente as variações dos valores das ações na bolsa de valores. Os parâmetros adotados e a avaliação de predição das métricas em cada uma das simulações estão especificadas abaixo, nas Tabelas 1 e 2.

Tabela 1
Parâmetros da rede LSTM

Simulação	Épocas	Batch	Otimizador	F.Ativação	T.Aprendizado	NINT1	NINT2
1	100	32	Adam	Linear	0.001	50	50
2	150	32	Adam	Linear	0.001	50	50
3	150	32	Adam	Linear	0.001	60	50
4	150	32	Adam	Linear	0.01	60	50
5	150	32	Adam	Linear	0.01	70	60
6	150	32	Adam	Relu	0.01	60	50
7	150	32	SGD	Linear	0.01	60	50
8	200	40	Adam	Linear	0.01	70	60

A simulação mostrada na Figura 7 proporcionou erro quadrático médio de 0.0006, erro absoluto médio de 0.0185, MAPE de 5.62% e RMSE de 0.0175. É possível observar que o preço predito não consegue acompanhar a oscilação do preço da ação (valor real). A simulação mostrada na Figura 8 mostra uma similaridade bem grande, quanto as oscilações, em comparação com as oscilações dos valores reais das ações, porém os valores em si novamente nunca são iguais. As métricas encontradas com a simulação foram: MSE de 0.0005, MAE de 0.0170, MAPE de 4.45% e RMSE de 0.0155.

Tabela 2
Métricas da rede LSTM

Simulação	MAE	MSE	MAPE	RMSE
1	0.0185	0.0006	5.62%	0.0175
2	0.0170	0.0005	4.45%	0.0155
3	0.0159	0.0005	4.97%	0.0154
4	0.0206	0.0007	3.64%	0.0255
5	0.0171	0.0007	3.80%	0.0149
6	0.0205	0.0005	3.62%	0.0197
7	0.1008	0.0232	12.08%	0.1307
8	0.0185	0.0005	3.74%	0.0173

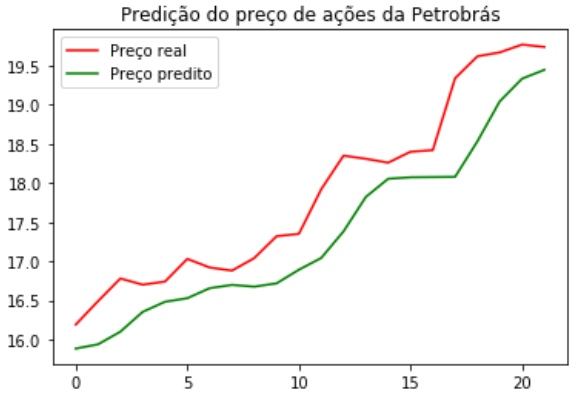

Figura 7. 1ª simulação da rede LSTM

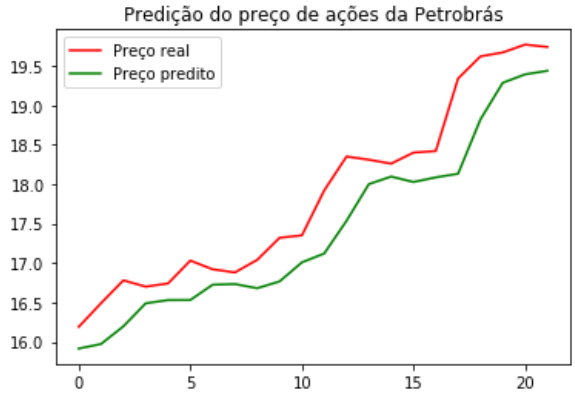

Figura 8. 2ª simulação da rede LSTM

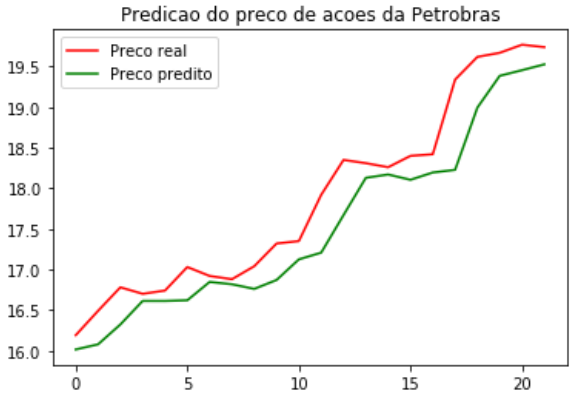

Figura 9. 3ª simulação da rede LSTM

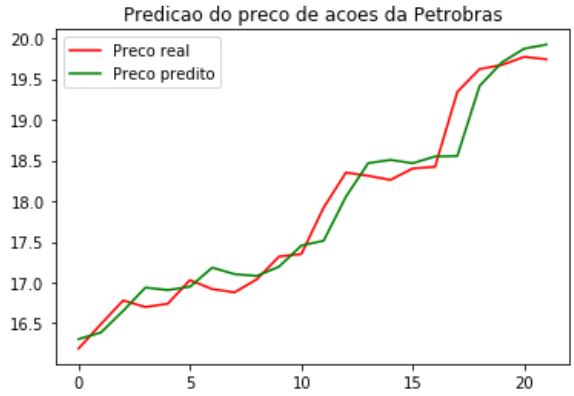

Figura 10. 4ª simulação da rede LSTM

A simulação mostrada na Figura 9 é bem similar a da Figura 8, porém percebe-se uma aproximação maior quanto as oscilações dos valores reais das ações na bolsa e até mesmo mais próximo do valor em si. As métricas encontradas com a simulação foram: MAE de 0.0159, MSE de 0.0005, MAPE de 4.97% e RMSE de 0.0154.

Na simulação da Figura 10 é possível observar a predição certa de alguns valores, porém as oscilações destes não se assemelham muito ao real. As métricas encontradas nessa simulação foram: MAE de 0.0206, MSE de 0.0007, MAPE de 3.64% e RMSE de 0.0255.

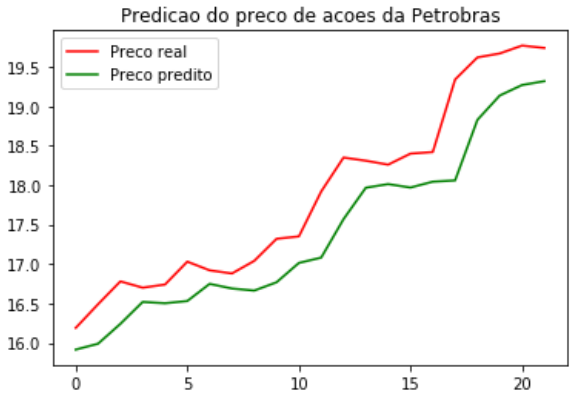

Figura 11. 5ª simulação da rede LSTM

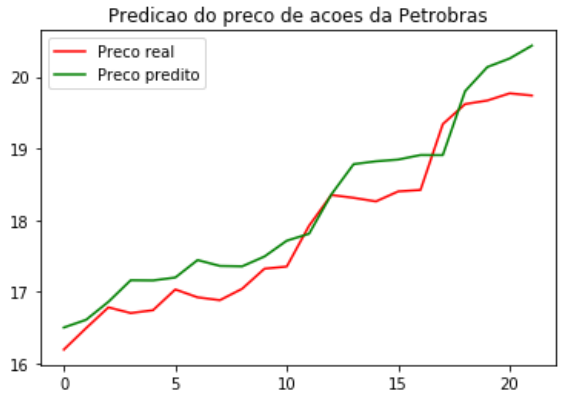

Figura 12. 6ª simulação da rede LSTM

Na simulação 5, Figura 11, tem-se um cenário parecido com o da 3ª simulação, porém não superando e nem se igualando. As métricas encontradas nessa simulação foram: MAE de 0.0171, MSE de 0.0007, MAPE de 3.80% e RMSE de 0.0149.

Na simulação 6, Figura 12, tem-se alguns valores iguais aos reais, porém as oscilações estão distantes da realidade. As métricas encontradas nessa simulação foram: MAE de 0.0205, MSE de 0.0005, MAPE de 3.62% e RMSE de 0.0197.

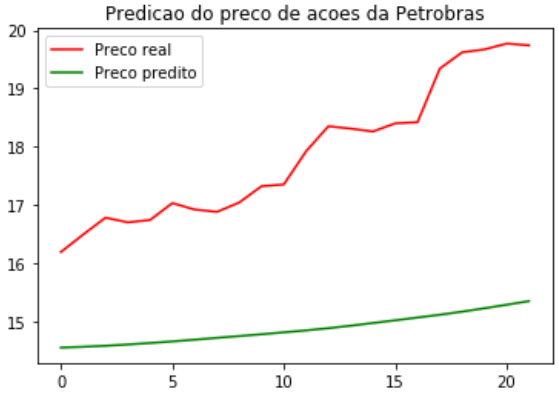

Figura 13. 7ª simulação da rede LSTM

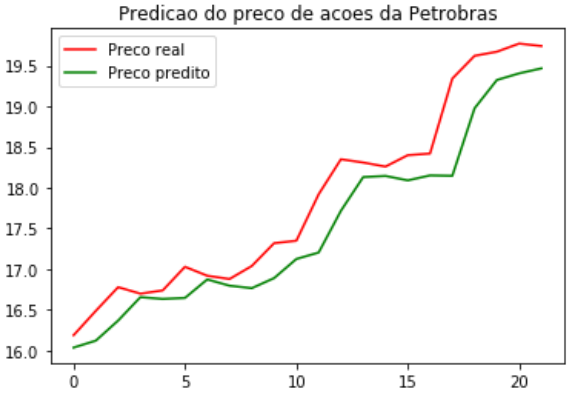

Figura 14. 8ª simulação da rede LSTM

A simulação 7, Figura 13, foi de longe a pior dentre as feitas, nem oscilações e nem os valores em si chegam perto do real. As métricas encontradas nessa simulação foram: MAE de 0.1008, MSE de 0.0232, MAPE de 12.08%, RMSE de 0.1307.

A simulação 8, Figura 14, se mostrou uma das melhores, com as oscilações das ações na bolsa de valores aproximando-se das reais. As métricas encontradas nessa simulação foram: MAE de 0.0185, MSE de 0.0005, MAPE de 3.74% e RMSE de 0.0173.

5.2 Rede direta

As simulações com a rede direta proporcionaram melhores resultados quanto a predição dos valores em si das ações na bolsa de valores. Os parâmetros adotados e as avaliações de predição das métricas alcançados estão representados abaixo, nas Tabelas 3 e 4.

Tabela 3
Parâmetros da rede direta

Simulação	Épocas	Batch	Otimizador	F.Ativação	T.Aprendizado	NINT1	NINT2
1	1000	32	Adam	Relu	0.01	50	60
2	1000	32	Adam	Relu	0.001	70	60
3	1000	32	Adam	Linear	0.01	70	60
4	1000	40	SGD	Linear	0.01	70	60
5	1000	32	Adam	Linear	0.01	70	70
6	2000	32	Adam	Linear	0.01	70	70
7	2500	32	Adam	Linear	0.01	70	70
8	3000	32	Adam	Linear	0.01	80	70

Tabela 4
Métricas da rede direta

Simulação	MAE	MSE	MAPE	RMSE
1	0.0161	0.2499	1.70%	0.0150
2	0.0273	0.0014	2.73%	0.4592
3	0.0157	0.0005	1.53%	0.0168
4	0.0561	0.0050	5.97%	0.0588
5	0.0155	0.0005	1.61%	0.0162
6	0.0153	0.0005	1.51%	0.0148
7	0.0146	0.0004	1.57%	0.0153
8	0.0146	0.0004	1.47%	0.0147

Na 1ª simulação, Figura 15, com a rede direta é possível ver poucos pontos onde o valor do preço das ações é semelhante ao real e as oscilações ficaram também longe de uma predição eficiente e correta. As métricas encontradas foram: MAE de 0.0161, MSE de 0.2499, MAPE de 1.70% e RMSE de 0.0150.

Na 2ª simulação, Figura 16, tem-se um resultado muito ruim, com oscilações e valores preditos bem distantes do real. As métricas encontradas foram MAE de 0.0273, MSE de 0.0014, MAPE de 2.73% e RMSE de 0.4592.

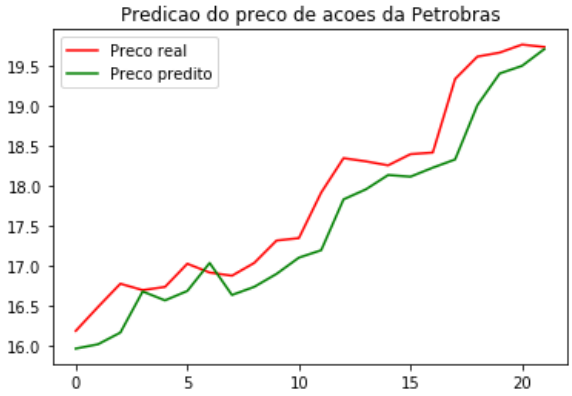

Figura 15. 1ª simulação da rede direta

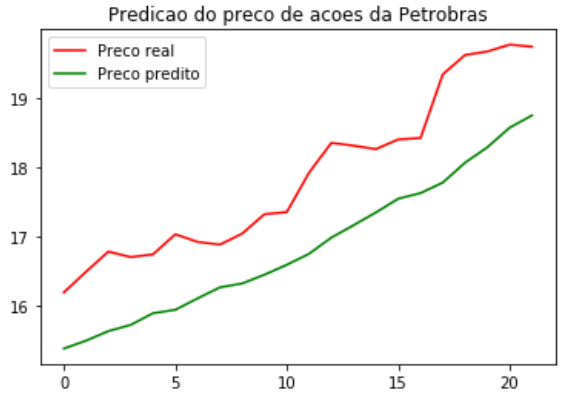

Figura 16. 2ª simulação da rede direta

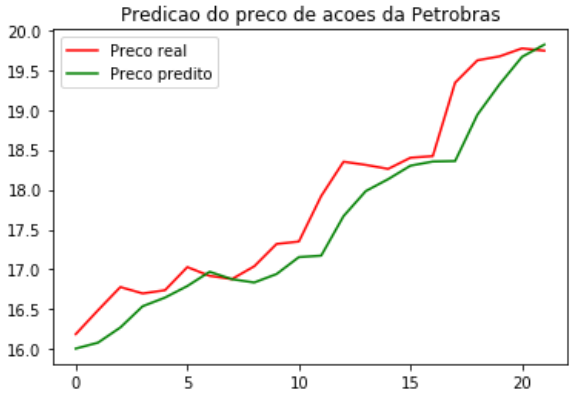

Figura 17. 3ª simulação da rede direta

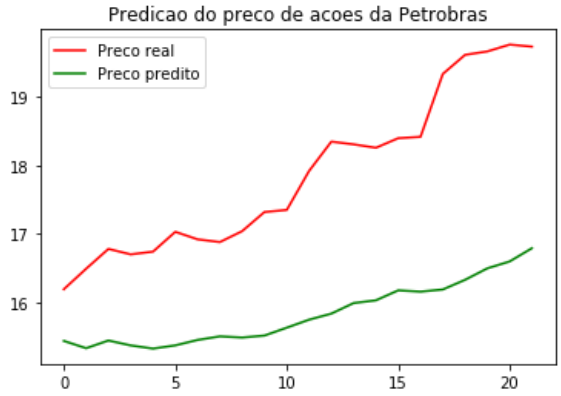

Figura 18. 4ª simulação da rede direta

A simulação 3, Figura 17, teve um resultado mais satisfatório que as duas anteriores, proporcionando oscilações mais próximas do real e alguns valores de ações igualando-se. As métricas encontradas foram: MAE de 0.0157, MSE de 0.0005, MAPE de 1.53% e RMSE de 0.0168.

Na simulação 4, Figura 18, na parte das métricas, tivemos o pior resultado em comparação com as anteriores, proporcionando tanto oscilação quanto valores bem ineficientes quanto a predição. As métricas nesta simulação foram: MAE de 0.0561, MSE de 0.0050, MAPE de 5.97% e RMSE de 0.0588.

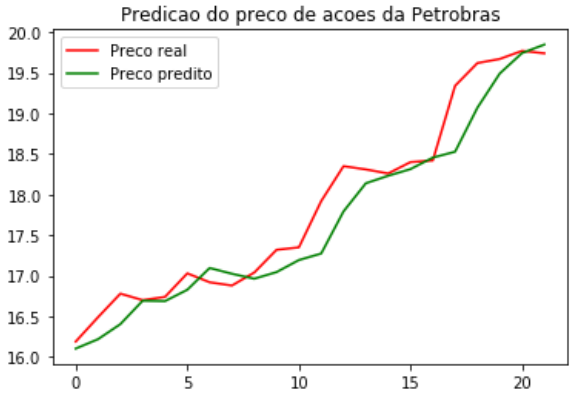

Figura 19. 5ª simulação da rede direta

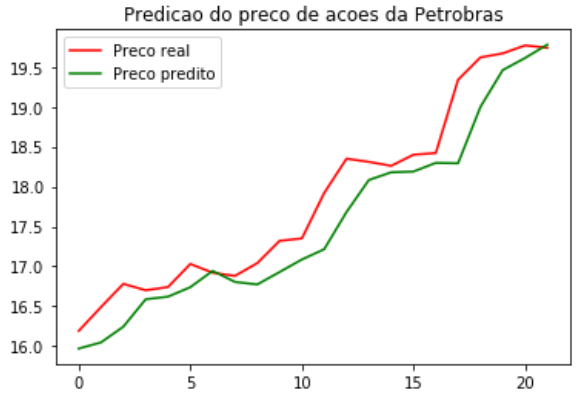

Figura 20. 6ª simulação da rede direta

Na simulação 5, Figura 19, na parte das métricas, foi obtido um resultado bem melhor do que nas anteriores, onde a oscilação ficou mais próxima do real e a predição de valores foi exata em alguns pontos do gráfico exposto na Figura 15. As métricas desta simulação foram: MAE de 0.0155, MSE de 0.0005, MAPE de 1.61% e RMSE de 0.0162.

Na simulação 6, Figura 20, tem-se, o melhor resultando considerando todas as métricas, é possível ver uma aproximação bem maior da realidade quanto a oscilação e ainda alguns valores foram preditos certos. As métricas nesta simulação foram: MAE de 0.0153, MSE de 0.0005, MAPE de 1.51% e RMSE de 0.0148.

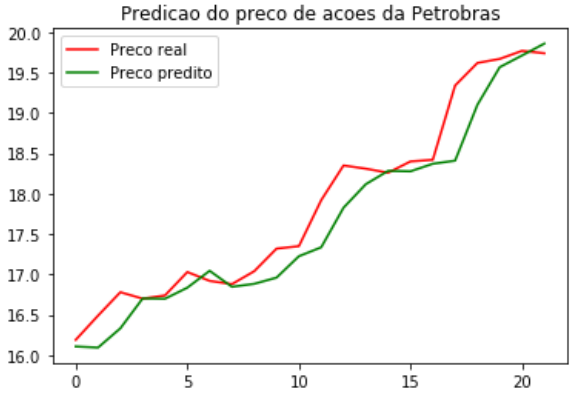

Figura 21. 7ª simulação da rede direta

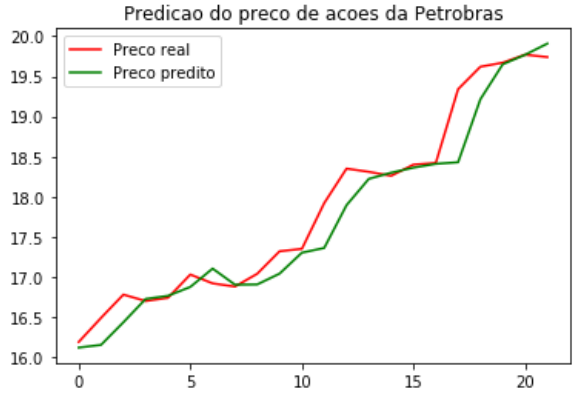

Figura 22. 8ª simulação da rede direta

Na simulação 7, Figura 21, tem-se o segundo melhor resultado dentre todas as simulações, proporcionando oscilação e valores bem satisfatórios e métricas com valores baixos. As métricas foram: MAE de 0.0146, MSE de 0.0004, MAPE de 1.57% e RMSE de 0.0153.

Na simulação 8, Figura 22, obteve-se o melhor resultado, tendo obtido uma melhora por conta do aumento do número de épocas nos parâmetros utilizados. As métricas foram: MAE de 0.0146, MSE de 0.0004, MAPE de 1.47% e RMSE de 0.0147.

6 CONCLUSÕES

Levando em consideração os valores de forma geral a rede direta proporcionou o melhor resultado nas simulações 7 e 8, muito parecidas quanto a métricas, onde a 8 é um pouco melhor por possuir um MAPE e um RMSE menores, 1.47% e 0.0147 respectivamente. As melhores simulações, feitas na rede direta, caracterizaram-se por predizer corretamente em diversas oportunidades o valor correto das ações na bolsa de valores, alcançando esse resultado principalmente por conta do aumento no número de épocas e uma variação pequena na quantidade de neurônios das camadas ocultas.

A rede LSTM mostrou-se mais eficiente no que se refere a oscilações das ações na bolsa de valores, mostrando altas e baixas semelhantes ao real. As melhores simulações nesse tipo de rede foram a 3 e 5, sendo que a 3 obteve melhor desempenho na métricas MAE e MSE, 0.0159 e 0.0005 respectivamente, enquanto a simulação 5 obteve melhores resultados nas métricas MAPE e RMSE, 3.80% e 0.0149 respectivamente. As 2 melhores simulações da rede LSTM tiveram configurações de parâmetros bem parecidas, tendo em comum: 150 épocas, batch de 32, otimizador adam e função de ativação linear. A diferença fica por conta da taxa de aprendizado, 0.001 para a simulação 3 e 0.01 para a 5, e na quantidade de neurônios das camadas internas, maior na simulação 5.Na rede LSTM os parâmetros que mais influenciaram nos resultados foram a taxa de aprendizado e a quantidade de neurônios das camadas ocultas.

Quanto aos piores resultados, em ambos tipos de rede, estão relacionados diretamente com a alteração de otimizador para SGD, resultando em um desempenho muito inferior com relação as simulações que utilizaram o otimizador adam. Na rede LSTM foi obtido o pior resultado, disparado, onde as métricas ficaram bastante altas e o gráfico gerado ficou totalmente inapropriado para fazer qualquer análise válida.

Agradecimentos

Agradecemos a Fundação Amazônia de Amparo a Estudos e Pesquisa - FAPESPA pela concessão da bolsa de iniciação científica.

REFERÊNCIAS

Almuammar, M., & Fasli, M. (2019, December). Deep learning for non-stationary multivariate time series forecasting. *IEEE International Conference on Big Data (BigData)*, Los Angeles, CA, USA, 2019.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2006). Greedy layer-wise training of deep networks. In B. Schölkopf, J. C. Platt, & T Hoffman (Eds.). *Advances in neural information processing systems 19* (pp. 153-160). San Diego: Neural Information Processing Systems Fundations.

Bovespa (Brasil). *Produtos*. Recuperado em 05 maio, 2020, de http://www.b3.com.br/pt_br/.

Dasgaonkar, K., & Chopade, S. (2018). Analysis of multi-layered perceptron, radial basis function and convolutional neural networks in recognizing handwritten digits. *International Journal of Advance Research, Ideas and Innovations in Technology*, *4*(3), 2429-2431.

Gambogi, J. A. (2013). *Aplicação de redes neurais na tomada de decisão no mercado de ações*. Dissertação de mestrado, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil.

Garcia, J. C. P., & Campos, L. M. L. (2019). Redes neurais apoiando a tomada de decisão na análise de crédito bancário e detecção do câncer de mama. *Revista Gestão e Tecnologia*, *19*, 90-112.

Gustafsson, F. K., Danelljan, M., & Schön, T. B. (in press). Evaluating scalable bayesian deep learning methods for robust computer vision. *Computer Science*. Retrieved May 16, 2020, from arXiv:1906.01620.

Jesus, L. B., Jr., & Sarti, F. (2016). Petrobras: política de conteúdo local, natureza jurídica, governança corporativa e performance econômica. *Economic Analysis of Law Review*, (2), 530-576.

Junior, J. R. F. (2019, junho 11). *Redes neurais recorrentes – LSTM*. Recuperado em 20 maio, 2020, de https://medium.com/@web2ajax/redes-neurais-recorrentes-lstm-b90b720dc3f6

Kaushik, A., Gupta, S., & Bhatia, M. (2018). A movie recommendation. System using Neural Networks. *International Journal of Advance Research, Ideas and Innovations in Technology*, *4*(2), 425-430.

Kovács, Z. L. (2006). *Redes neurais artificiais: fundamentos e aplicações*. (4a ed.). São Paulo: Livraria da Física.

Langner, T., Ahlström, H., & Kullberg, J. (in press). Large-scale biometry with interpretable neural network regression on UK Biobank body MRI. *Electrical Engineering and Systems Science*. Retrieved May 16, 2020, from arXiv:2002.06862

Marques, A. M. (2012). Avaliação de densidades para previsão dos retornos das ações da Petrobrás. *Revista Economia e Desenvolvimento*, *24*(2), 49-66.

Ning, Y., He, S., Wu, Z., Xing, C., & Zhang, L.-J. (2019). A review of deep learning based speech synthesis. *Applied Sciences*, *9*(19), 4050. doi: 10.3390/app9194050

Petróleo Brasileiro S.A. - Petrobras (PETR4.SA). (2020). *Yahoo Finanças*. Recuperado em 19 maio, 2020, de https://br.financas.yahoo.com/quote/PETR4.SA/history?p=PETR4.SA

Poonia, V., Tiwari, H. L., & Mishra, S. (2018). Hydrological analysis by Artificial Neural Network: a review. *International Journal of Advance Research, Ideas and Innovations in Technology, 4*(3), 265-670.

Rede neural artificial. (2018). In *Wikipedia enciplopedia livre*. Recuperado em 01 junho, 2018, de https://pt.wikipedia.org/wiki/Rede_neural_artificial

Santos, D. B., Lucas, E. C., Silva, V. A. B., & Medeiro, B. N. (2015, abril). Influência intradiária do preço internacional do petróleo nas ações da Petrobrás. *Journal of Financial Innovation, 1*(1), 4-17.

Shamshirband, S., Rabczuk, T., & Chau, K.-W. (2019). A survey of deep learning techniques: application in wind and solar energy resources. *IEEE Access*, *7*, 164650-164666. doi: 10.1109/ACCESS.2019.2951750.

Sharma, N., Agarwal, P., & Pandey, U. (2018). Offline handwriting recognition using neural networks. *International Journal of Advance Research, Ideas and Innovations in Technology, 4*(2), 1541-1545.

Steiner, M. T. A., Soma, N. Y., Shimizu, T., Nievola, J. C., Lopes, F. M., & Smiderle, A. (2004, novembro). *Redes neurais e arvores de decisão na análise do crédito bancário*. Trabalho apresentado em Simpósio Brasileiro de Pesquisa Operacional, São João del-Rei, MG, Brasil, 36. Recuperado em 01 junho, 2018, de http://www.din.uem.br/sbpo/sbpo2004/pdf/arq0035.pdf

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to sequence learning with neural networks*. Retrieved June 01, 2018, from http://arxiv.org/abs/1409.3215