UNIVERSIDADE DO SUL DE SANTA CATARINA

ADRIANA MIYAZAKI DE MOURA

REDES NEURAIS RECORRENTES APLICADAS À PREVISÃO DE PREÇOS DE ATIVOS DA BOVESPA: COMPARATIVOS DE MODELOS PARA REGRESSÃO

Palhoça - SC

2019

ADRIANA MIYAZAKI DE MOURA

REDES NEURAIS RECORRENTES APLICADAS À PREVISÃO DE PREÇOS DE ATIVOS DA BOVESPA: COMPARATIVO DE MODELOS PARA REGRESSÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Matemática - Bacharelado da Universidade do Sul de Santa Catarina, como requisito parcial à obtenção do título de Bacharel(a) em Matemática.

Orientador: Prof(a). Osmar de Oliveira Braz Junior.

Palhoça - SC

2019

ADRIANA MIYAZAKI DE MOURA

REDES NEURAIS RECORRENTES APLICADAS À PREVISÃO DE PREÇOS DE ATIVOS DA BOVESPA: COMPARATIVOS DE MODELOS PARA REGRESSÃO

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do título de Bacharel(a) em Matemática e aprovado em sua forma final pelo Curso de Graduação em Matemática da Universidade do Sul de Santa Catarina.

Palhoça, 20 de dezembro de 2019.

_____

Professor e orientador Osmar de Oliveira Braz Junior, Msc.
Universidade do Sul de Santa Catarina

_____

Prof. Oscar Ciro Lopez Vaca, Doutor.
Universidade do Sul de Santa Catarina

_____

Profa. Diva Marília Flemming, Doutura.
Universidade do Sul de Santa Catarina

Este trabalho é dedicado à minha família, que sempre esteve presente nos momentos bons e ruins da minha vida, me apoiando e acreditando nos meus sonhos.

AGRADECIMENTOS

Agradeço à minha família por todo o apoio e suporte; aos professores e profissionais da Unisul que foram fundamentais no meu aprendizado e conquista do grau de bacharel de Matemática; à professora coordenadora do curso, Diva; ao meu orientador, Osmar; aos cientistas e pesquisadores que pavimentaram o caminho para a elaboração deste trabalho.

RESUMO

O comportamento das bolsas de valores pode ser melhor compreendido com o auxílio da modelagem matemática. Modelos matemáticos podem ser obtidos através de processamento computacional utilizando aprendizagem de máquina. Um dos ramos da aprendizagem de máquina que desenvolvem uma modelagem empregando algoritmos utilizando grafos com várias camadas é o *deep learning*, o qual possui arquiteturas específicas para lidar com séries temporais, as chamadas Redes Neurais Recorrentes. No presente trabalho, foram coletados dados dos preços de fechamentos diários de ações da bolsa de valores BOVESPA com o intuito de servir de variáveis independentes para a previsão do Índice BOVESPA. Essas informações foram utilizadas para realizar um comparativo entre as arquiteturas de *deep learning*: Redes Neurais Recorrentes de Jordan, Redes Neurais Recorrentes de Elman e Redes Neurais *Long Short Term Memory* (LSTM) em tarefa de regressão. No geral, todas as arquiteturas investigadas apresentaram desempenho satisfatório, medido pelo método dos mínimos quadrados. A arquitetura que apresentou melhor performance, foi a Rede Neural de Jordan. A fim de melhorar o entendimento do comportamento do Índice BOVESPA é sugerida a execução de novas pesquisas, utilizando outras configurações e variáveis para as arquiteturas aqui empregadas.

Palavras-chave: *Deep Learning*. Redes Neurais Recorrentes. Séries Temporais Financeiras. Bolsa de Valores. Redes Neurais Recorrentes de Jordan. Redes Neurais Recorrentes de Elman. *Long Short Term Memory*.

LISTA DE ILUSTRAÇÕES

LISTA DE GRÁFICOS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

As bolsas de valores são associações sem fins lucrativos, que asseguram o funcionamento e operação do mercado acionário. Através delas são executadas ordens de compra e venda de ações, o que possibilita o investimento em empresas por parte de investidores (LUQUET, 2007).

Podemos tentar entender melhor o comportamento das bolsas de valores através da utilização da modelagem (CAMPOS et al., 2011). O produto da modelagem matemática é um modelo matemático, o qual consiste de relações matemáticas e símbolos, cuja função é descrever ou traduzir um problema ou fenômeno do mundo real (MOREIRA, 2014). A modelagem matemática computacional pode ser empregada quando estamos diante de situações com elevado número de variáveis envolvidas. Assim, ferramentas computacionais e técnicas avançadas de programação são utilizadas na busca de uma solução ótima de problemas complexos, cujos cálculos são realizados de forma mais ágil com o auxílio do processamento computacional (GARCIA, 2010).

Deep learning (do inglês, aprendizagem profunda) é uma ferramenta computacional avançada utilizada para resolução de problemas, no qual os computadores aprendem através da experiência a entender determinadas situações. A estrutura de uma rede *deep learning* consiste de um grafo com muitas camadas, ou seja, um grafo profundo (GOODFELLOW et al, 2016). No presente trabalho, iremos realizar comparações entre modelagens matemáticas empregando técnicas de *deep learning*.

1.1 TEMA E DELIMITAÇÃO DO TEMA

O tema do trabalho é a análise de diferentes arquiteturas de *deep learning* na tarefa de regressão para modelagem matemática de preços do índice BOVESPA.

1.2 PROBLEMATIZAÇÃO

Diferentes tipos de problemas foram resolvidos com o emprego de métodos de *deep learning*, incluindo modelagem de preços de ações na bolsa de valores (CHEN; ZHOU; DAI, 2015; ROONDIWALA; PATEL; VARMA, 2017; NELSON; PEREIRA; OLIVEIRA, 2017; SELVIN et al., 2017; MUNTASER et al., 2019). Ao considerarmos problemas de regressão com séries temporais, temos que as arquiteturas de deep learning mais adequadas são as redes neurais recorrentes. Dentre elas, temos os métodos LSTM, Redes Neurais Recorrentes de Jordan e Redes Neurais Recorrentes de Elman (LEWIS, 2016). Neste trabalho, estamos interessados em descobrir qual dos métodos citados apresenta melhor desempenho na modelagem de preços do índice BOVESPA. Também é de interesse saber se os modelos apresentam desempenho suficiente para serem utilizados como ferramenta de auxílio na tomada de decisão de compra ou venda de ações na bolsa de valores brasileira.

1.3 JUSTIFICATIVAS

O investimento tem papel determinante no crescimento econômico de uma nação (Haberler, 1976). Os benefícios gerados no ato de investir são de fundamental importância: a produtividade de um país aumenta, eleva-se o PIB, empresas são estruturadas, empregos são gerados, a renda da sociedade em geral é aumentada (TOLEDO, 2006).

Para arrecadar capital, lançar ações na bolsa de valores é uma alternativa aos empréstimos de bancos e financeiras. Ações são títulos que representam uma parte mínima de uma empresa. Ao adquirir ações de uma empresa, o comprador investe nela seu próprio capital e torna-se seu sócio (PIAZZA, 2008, p. 18).

A fim de não por em risco seu patrimônio, um investidor deve utilizar ferramentas adequadas para avaliar a situação de uma empresa e seu potencial de crescimento (ANDRADE; TAVARES, 2010). Uma das ferramentas para auxiliar na decisão de compra ou venda de uma ação é a utilização de modelagem matemática para previsão do valor futuro de um ativo (MIRANDA; CORONEL; VIEIRA, 2013).

A previsão de preços de ações é de grande complexidade, já que sua modelagem envolve muitas variáveis e há numerosos dinamismos inerentes ao mercado. Tais como conflitos econômicos, fatores políticos, sociais ou macroeconômicos, os quais afetam diretamente a economia. Esses fatores externos podem influenciar na variação dos preços de um ativo de forma brusca (NELSON, 2017).

Diversas pesquisas foram desenvolvidas avaliando a performance das redes neurais para previsão de retornos, preços futuros, volatilidade e outras medições, empregando dados de bolsa de valores de diferentes países, incluindo a BOVESPA, a Tabela 1 mostra alguns desses estudos. Abreviações utilizadas na Tabela 1: LSTM; RNJ – Redes Neurais de Jordan; RNE – Redes Neurais de Elman; RNEJ – Redes Neurais Mistas de Elman e Jordan; RNR – Redes Neurais Recorrentes; RNC – Redes Neurais Convolucionais; MLP – *Multilayer Perceptron*; OM – outros métodos; País – bolsa de valores do país em questão. Como apresentado na Tabela 1 não há estudo que realize uma comparação entre os seguintes métodos, no mercado de ações do Brasil: LSTM, Redes Neurais Recorrentes de Jordan e Redes Neurais Recorrentes de Elman.

A hipótese que foi investigada no presente trabalho é a seguinte: se os métodos LSTM, Redes Neurais Recorrentes de Jordan e Redes Neurais Recorrentes de Elman apresentaram desempenhos satisfatórios em tarefas de regressão utilizando dados financeiros, então provavelmente irão apresentar um bom desempenho na previsão dos valores futuros do índice BOVESPA.

Este trabalho visou realizar a comparação entre três tipos de arquiteturas de redes neurais. A fim de realizar uma verificação justa, foi optado por manter os parâmetros dos modelos iguais entre as redes. Em busca de um modelo de maior acurácia, os parâmetros poderiam ser variados e mais camadas escondidas poderiam ser adicionadas na modelagem. Nesse ponto foram encontradas limitações, posto que o custo computacional de realizar o procedimento de *cross-validation* (descrito na seção de Metodologia) é alto e sua execução demorada, o que demandaria mais tempo do que o disponível para a realização do estudo.

Tabela 1 – Estudos empregando técnicas de *deep learning* para previsão, com dados de ativos de bolsas de valores.

Autor	**LSTM**	**RNJ**	**RNE**	**RNEJ**	**RNR**	**RNC**	**MLP**	**OM**	**País**
CHEN; ZHOU; DAI, 2015	X								China
ROONDIWALA; PATEL; VARMA, 2017	X								Índia
NELSON; PEREIRA; OLIVEIRA, 2017	X								Brasil
SELVIN et al., 2017	X				X	X			Índia
NAEINI; TAREMIAN; HASHEMI, 2010			X				X		Irã
FADDA; CAN, 2017		X	X	X					EUA
MESQUITA; OLIVEIRA; PEREIRA, 2019	X								Brasil
WERNER; BISOGNIN; ARAUJO, 2018	X							X	Brasil
MUNTASER et al., 2019							X		Brasil

Fonte: Elaboração da autora, 2019.

1.4 OBJETIVOS

Os objetivos do trabalho estão organizados em objetivo geral e específicos.

1.4.1 Objetivo Geral

Avaliar o desempenho dos métodos de *deep learning* específicos para séries temporais: LSTM, Redes Neurais Recorrentes de Jordan e Redes Neurais Recorrentes de Elman, para previsão de preços do índice BOVESPA.

1.4.2 Objetivos Específicos

O presente trabalho possui os seguintes objetivos específicos:

- Elaborar modelos de previsão (regressão) utilizando Redes Neurais Recorrentes para séries temporais.
- Comparar o desempenho dos modelos, utilizando três técnicas: LSTM, Redes Neurais Recorrentes de Jordan e Redes Neurais Recorrentes de Elman.

1.5 TIPO DA PESQUISA

O presente estudo se utiliza pesquisa quantitativa, ou seja, uma pesquisa cujos resultados podem ser quantificados e há utilização de linguagem matemática para descrição de um fenômeno, relações entre variáveis, etc (FONSECA, 2002). A pesquisa consiste em: coleta de dados, processamento e modelagem matemática computacional.

1.6 ESTRUTURA DO TRABALHO

Inicialmente, foi realizada pesquisa bibliográfica acerca da teoria das redes *deep learning*. Através da pesquisa, foi identificado que as redes neurais recorrentes são as mais adequadas para lidar com previsão de séries temporais.

Dentre as diversas arquiteturas de redes neurais recorrentes, foram selecionadas três para realizarmos um comparativo de desempenho, sendo elas: Redes Neurais Recorrentes de Jordan, Redes Neurais Recorrentes de Elman e LSTM.

Os métodos foram avaliados com o procedimento de *cross-validation* e com a utilização do Método dos Mínimos Quadrados (ambos os procedimentos estão descritos no capítulo de Fundamentação Teórica).

O Capítulo 2 apresenta o resultado da pesquisa bibliográfica, fornecendo ao leitor uma breve Fundamentação Teórica sobre os conceitos de *deep learning* e as Redes Neurais utilizadas neste trabalho.

Através da utilização linguagem *Python* e suas bibliotecas, foram realizados a aquisição dos dados dos preços de fechamento das ações analisadas e sua modelagem através das redes neurais citadas. Por fim, foi feita a investigação do poder preditivo das redes analisadas. O Capítulo 3 detalha como esse experimento foi realizado.

No Capítulo 4 são apresentados os resultados e a discussão do experimento.

Por último, no Capítulo 4 estão presentes as conclusões.

2 REFERENCIAL TEÓRICO

Para a compreensão dos assuntos abordados neste trabalho é necessário o entendimento do funcionamento dos métodos de *deep learning* e das arquiteturas de redes neurais recorrentes que serão aqui empregadas.

2.1 APRENDIZADO DE MÁQUINA E DEEP LEARNING

O aprendizado de máquina é um ramo da inteligência artificial, em que algoritmos "aprendem" a executar certas tarefas, de acordo com experiências ou informações passadas (MITCHELL, 1997, p. 1-3). A área, geralmente, é dividida em duas partes: aprendizado supervisionado e não-supervisionado (MURPHY, 2012).

No aprendizado supervisionado, a modelagem é realizada utilizando como base um conjunto de dados, chamado de conjunto de treinamento. O qual é composto de n tuplas ($\mathbf{x}_i$, y_i), onde $\mathbf{x}_i$ é um vetor de características e y_i o rótulo. No caso dos valores de y_i serem contínuos, temos um problema de regressão (LORENA; CARVALHO, 2007).

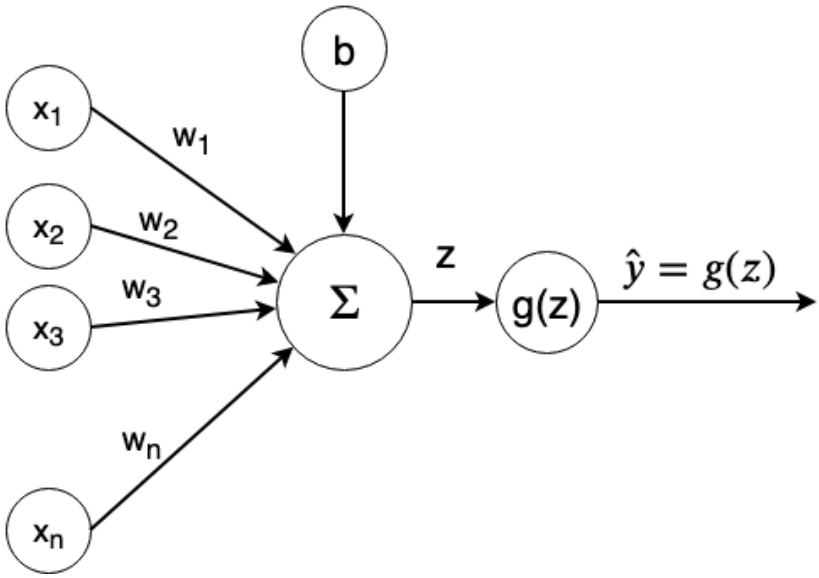
Nos problemas de regressão podem ser utilizados métodos *deep learning*. Os quais, a partir de dados não tratados, conseguem detectar as representações necessárias para realizar uma classificação ou regressão. Desse modo, para realizar a modelagem, não há necessidade de domínio profundo da área de expertise do modelo (LECUN; BENGIO; HINTON, 2015).

Historicamente, o *deep learning* surgiu na década de 1940 e ao longo dos anos possuiu diversos nomes, sendo um deles: redes neurais artificiais. Sua origem foi inspirada em modelos de neurônios cerebrais, porém sem compromisso com a simulação realística do funcionamento do cérebro (GOODFELLOW, 2016).

Um dos modelos iniciais de redes neurais foi o chamado *Perceptron* (ROSENBLATT, 1958), o qual é um modelo matemático que recebe n entradas (x_1, x_2, x_3, ..., x_n). Para cada entrada é atribuído um peso (w_1, w_2, w_3, ..., w_n). Na Figura 1 há um grafo que representa um *Perceptron* simples, no modelo, cada nó do grafo é denominado neurônio. O conjunto de nós que representa as entradas é chamado de camada de entrada, o último nó

representa a saída da rede corresponde à camada de saída. Na figura há apenas um neurônio escondido (representado pela letra grega sigma), cada neurônio escondido pode possuir um viés (b). O conjunto de todos os neurônios escondidos de uma camada compõe uma camada escondida (AMINI, 2019).

Figura 1 - Estrutura de um *Perceptron* simples



Fonte: Adaptado de AMINI, 2019.

No modelo do *Perceptron* simples acima, temos como saída do neurônio escondido (z) o produto linear do vetor de pesos com o vetor de entradas somado com o viés b (BISHOP, 2006):

$$z = b + \sum_{i=1}^{n} x_i w_i \qquad (1)$$

Na Equação 1, temos que o índice "i" está associado aos valores de x da camada de entrada e ao seus respectivos pesos w. O valor de z é entrada da função de ativação g(z), a qual introduz não-linearidade ao sistema. Frequentemente, são utilizadas funções sigmoides, de tangente hiperbólica ou de ativação linear retificada como funções de ativação (FACURE, 2017). O *Perceptron* simples tem então, como saída um valor estimado $\hat{y}$ para y (característica que se deseja prever):

$$\hat{y} = g(z) \qquad (2)$$

A Figura 2 apresenta um modelo de maior complexidade, uma rede neural com uma camada escondida. Assim como no *Perceptron*, cada camada escondida possui uma soma do produto linear dos valores cada neurônio da camada de entrada com o vetor de pesos. Dessa forma, para cada neurônio da camada escondida, temos (AMINI, 2019):

$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^{n} x_j w_{j,i}^{(1)} \qquad (3)$$

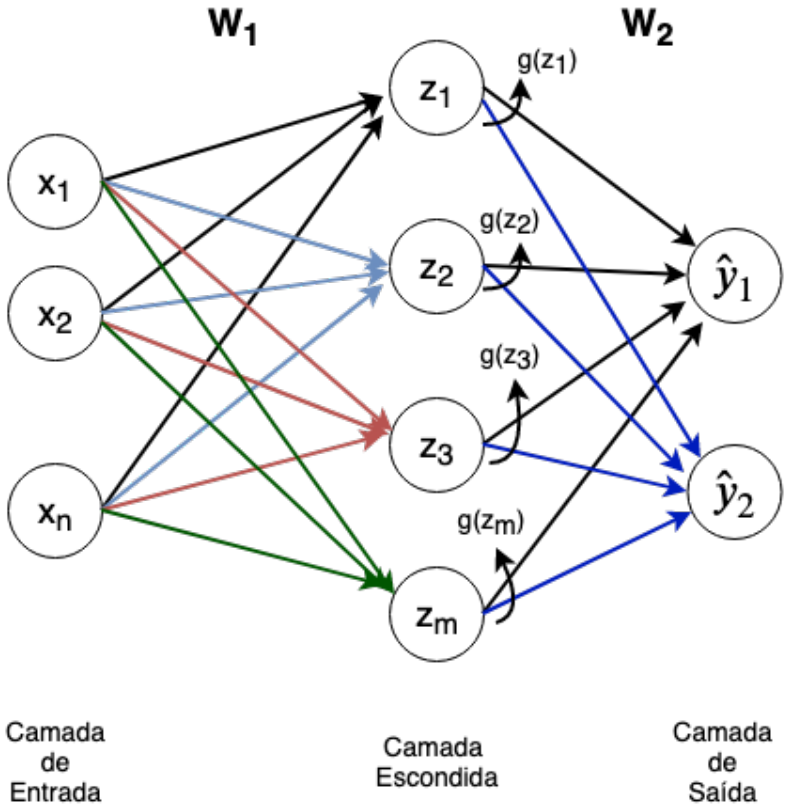
e

$$\hat{y}_i = g\left(w_{0,i}^{(2)} + \sum_{j=1}^{m} z_j w_{j,i}^{(2)}\right) \qquad (4)$$

Na Equação 3, o índice i representa um dos m neurônios da camada escondida, já o índice j está associado aos n valores de x da camada de entrada e seus respectivos pesos w. O valor (1) sobrescrito em w, na Equação 3, corresponde aos pesos das arestas com origem na camada de entrada e destino na camada escondida; $w_{0,i}^{(1)}$ são os valores dos vieses de cada neurônio i da camada escondida.

Analisando a Equação 4, o índice i representa um dos dois neurônios da camada de saída, o índice j corresponde aos m valores de z da camada escondida e seus respectivos pesos w; o valor (2) sobrescrito em w, corresponde aos pesos das arestas com origem na camada escondida e destino na camada de saída; assim $w_{0,i}^{(2)}$ são os vieses de cada neurônio i da camada de saída.

Figura 2 - Estrutura de uma rede neural com uma camada



Fonte: Adaptado de AMINI, 2019.

Uma Rede Deep Learning possui diversas camadas escondidas, assim, para um neurônio da camada escondida, o valor de z pode ser expresso por (AMINI, 2019):

$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{k-1} g\left(z_{k-1,j}\right) w_{j,i}^{(k)} \qquad (5)$$

Assim como nas equações anteriores, na Equação 5, temos a saída z de cada neurônio escondido com os seus índices, k é o índice com o número da camada escondida e i o índice de um neurônio dentro de uma camada k. Da mesma forma, o k sobrescrito nos pesos w também é correspondente à camada escondida analisada. Nessa equação, função de ativação é novamente representada por g, tendo como entrada o valor de saída do neurônio j da camada escondida anterior (k – 1). Os vieses estão representados por $w_{0,i}^{(k)}$.

O processo de encontrar uma rede *deep learning* ótima para a resolução de um problema específico envolve a definição de quais seriam os vetores de pesos $w^{(i)}$ e parâmetros adequados. Com esse intuito, procuramos minimizar uma função de custo J(**W**), a qual é uma medida da acurácia do modelo, ou seja, o quanto as estimativas calculadas pelo modelo diferem dos valores reais (MARUMO, 2018).

Uma das possíveis funções de custo utilizadas com modelos de regressão que possuem números reais contínuos é o Método dos Mínimos Quadrados (MMQ) (AMINI, 2019):

$$J(W) = \text{MMQ} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (6)$$

onde, n é o total de exemplos da amostra de teste, y é o valor observado real e ŷ o valor previsto pelo modelo (LEWIS, 2016). O MMQ também pode ser expresso em termos dos vetores de peso e entradas da rede:

$$J(W) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(x_i; W)\right)^2 \qquad (7)$$

onde $f(x_i; W)$é a função que resulta no valor previsto pelo modelo (ŷ) e W o vetor de pesos correspondente (AMINI, 2019).

O gradiente descendente é um dos algoritmos mais empregados para minimizar uma função de custo selecionada, ou seja, otimizar uma rede neural (RUDER, 2016). Ele se baseia no conceito de gradiente, do cálculo vetorial, o qual é um vetor, definido como:

$$\nabla J = \frac{\partial J}{\partial W1}\vec{\imath} + \frac{\partial J}{\partial W2}\vec{\jmath} + \frac{\partial J}{\partial W3}\vec{k} \qquad (8)$$

(GONÇALVES; FLEMMING, 2007).

O vetor gradiente tem direção e sentido na qual a função f tem maior crescimento. Dessa forma, o maior decrescimento da função é dado por$-\nabla f$(GONÇALVES; FLEMMING, 2007).

Uma das variações do algoritmo é o chamado gradiente descendente estocástico, ele utiliza o gradiente para calcular os parâmetros **W** (vetor de pesos) para cada exemplo contido na amostra de treinamento:

$$W = W - \eta \nabla J(W; x_i; y_i) \qquad (9)$$

onde, η é a taxa de aprendizado (RUDER, 2016).

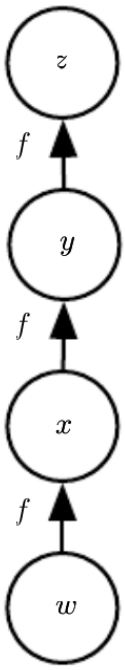
O algoritmo do gradiente descendente estocástico é dado por:

1. Os pesos **W** são inicializados de forma aleatória, seguindo uma distribuição normal de média zero e variância σ^2.
2. *Loop* até convergir:
3. Compute o gradiente $\nabla J_i(W; x_i; y_i)$do exemplo i da amostra de treino;
4. Atualize os pesos:$W = W - \eta \nabla J_i(W; x_i; y_i)$.
5. Retorna os pesos (AMINI, 2019).

As redes neurais *deep learning* aqui descritas são do tipo *feedforward*, o que significa que a informação flui adiante na rede. A propagação dos dados é realizada até um custo **J(W)** ser obtido. Porém, para o cálculo do gradiente, a informação flui em sentido oposto, através do algoritmo de *back-propagation.* O algoritmo utiliza a regra da cadeia do cálculo, assim, para um exemplo de rede Perceptron com múltiplas camadas como na Figura 3, teríamos (GOODFELLOW, 2016):

$$\nabla J = \frac{\partial z}{\partial w} = \frac{\partial z}{\partial y} . \frac{\partial y}{\partial x} . \frac{\partial x}{\partial w} \qquad (10)$$

Figura 3 - Estrutura de um *Perceptron* simples



Fonte: Adaptado de AMINI, 2019.

Na próxima seção iremos descrever um tipo específico de redes neurais *deep learning*: as redes neurais recorrentes.

2.2 REDES NEURAIS RECORRENTES

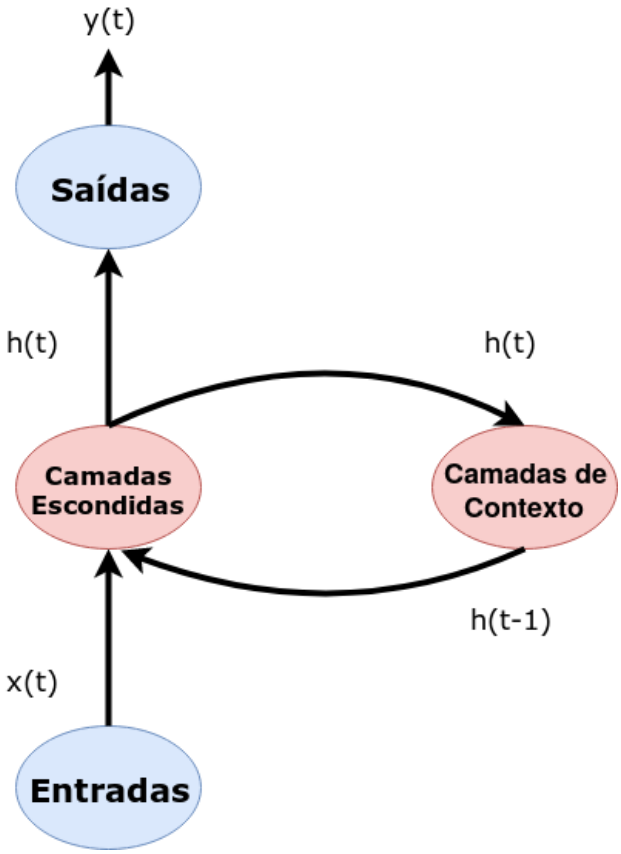
Dentre os métodos de *deep learning* que podem ser utilizados para regressão, se destacam as Redes Neurais Recorrentes, consideradas mais adequadas para processamento de dados de séries temporais. É possível encontrar diversas arquiteturas de Redes Neurais Recorrentes, dentre elas: Redes de Jordan, Redes de Elman e LSTM (LEWIS, 2016). A seguir, conceituaremos cada uma destas arquiteturas para melhor compreender suas estruturas e funcionamento.

2.2.1 Redes Neurais Recorrentes de Elman

As redes neurais recorrentes de Elman, representadas na Figura 4, tradicionalmente possuem: uma camada de entrada, uma camada escondida, uma camada de contexto e uma camada de saída (LEWIS, 2016), porém mais camadas escondidas podem ser acrescentadas ao modelo (WYSOCKI; LAWRYNCZUK, 2016). Na Figura 4, t representa o período de tempo, x(t) o vetor de entradas correspondente ao tempo t, h(t) o vetor com os resultados das camadas escondidas, h(t -1) o mesmo vetor para o período temporal imediatamente anterior (t-1) e, y(t) o vetor de saídas da rede neural (LEWIS, 2016).

Todos os neurônios da camada de contexto devem estar conectados à todos os da camada escondida, sendo que a quantidade de neurônios na camada escondida deve ser a mesma da camada de contexto. Ocorre uma memorização dos dados anteriores da série temporal (cópia dos valores anteriores) através dos neurônios da camada de contexto, que são alimentados pelos neurônios da camada escondida (LEWIS, 2016).

Figura 4 – Estrutura de rede neural recorrente de Elman

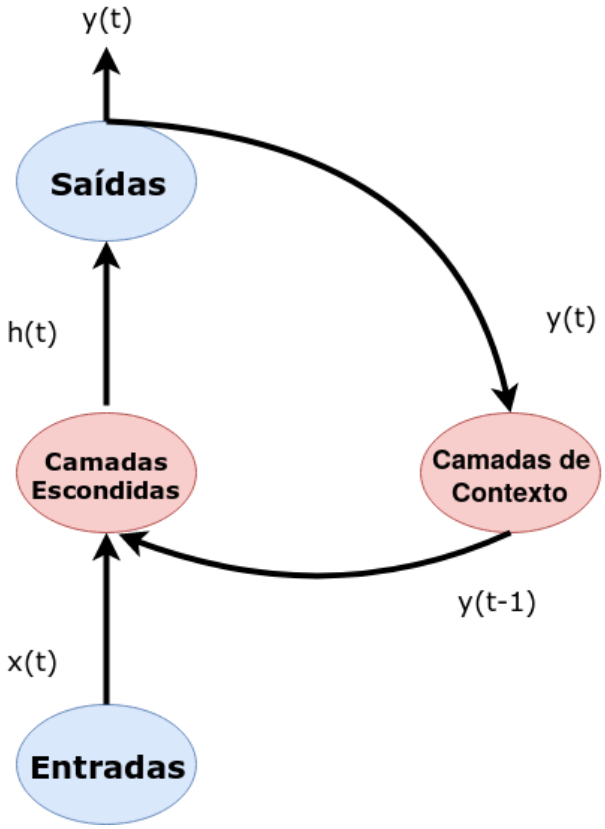


Fonte: Adaptado de LEWIS, 2016, p. 84.

2.2.2 Redes Neurais Recorrentes de Jordan

As redes neurais recorrentes de Jordan possuem estrutura semelhante às de Elman, na Figura 5 percebe-se que os neurônios da camada de contexto recebem os dados (y(t)) da camada de saída, ao invés da camada escondida (LEWIS, 2016).

Figura 5 – Estrutura de rede neural recorrente de Jordan



Fonte: Adaptado de LEWIS, 2016, p. 95.

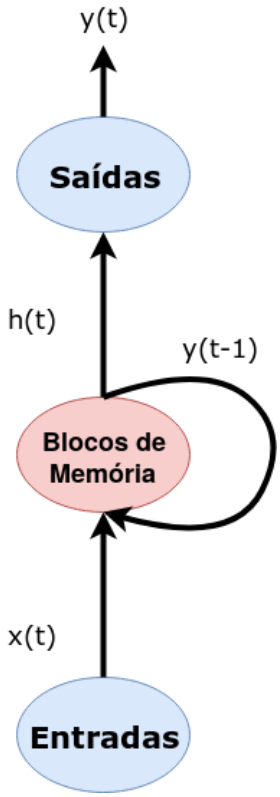
2.2.3 LSTM

Uma Rede LSTM é um tipo de rede neural recorrente, projetada para incorporar dependências temporais de longo termo em séries temporais, de forma efetiva e geral (GREFF et al., 2017).

As redes LSTM apresentaram os melhores desempenhos na modelagem matemática de diferentes tarefas e foram projetadas para apresentarem uma memória dos dados anteriores de uma série temporal mais longa do que as redes de Elman e Jordan. Como pode ser visto na Figura 6, sua estrutura é semelhante à das redes neurais recorrentes apresentadas anteriormente, porém as camadas de contexto e escondida são substituídas por blocos de memória. Os blocos de memória são compostos de três portões multiplicativos e

uma célula de memória. Os portões controlam o fluxo de informação na célula de memória (LEWIS, 2016).

Figura 6 – Estrutura de LSTM



Fonte: Adaptado de LEWIS, 2016, p. 122.

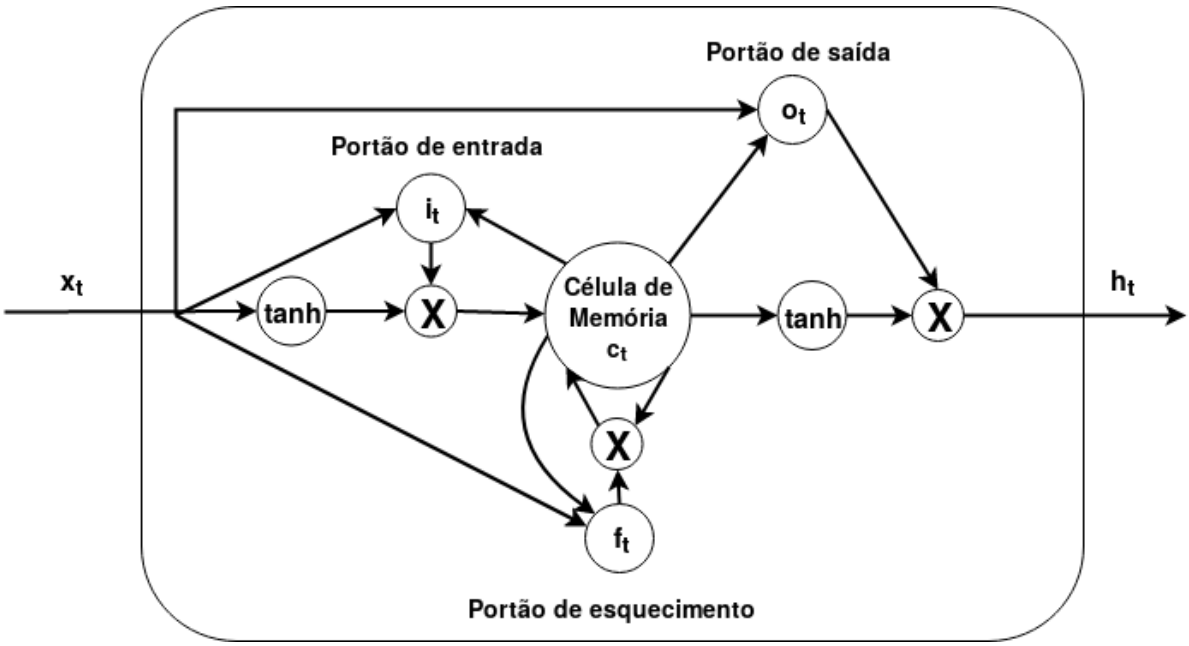
Um bloco de memória, representado pela Figura 7, possui os seguintes componentes:

- **Portão de entrada:** realiza a adição de informações ao bloco de memória. Funcionamento: tem duas entradas (x_t, o exemplo x no período t e, a saída da célula de memória); a saída da célula de memória contém o valor de x_t após passar por uma função tanh (o que faz seus valores estarem no intervalo entre -1 e 1) e multiplicado pelos respectivos pesos e adicionados de viés; ao passar por este portão ambas as entradas são multiplicadas pelos respectivos pesos e adicionadas de viés.
- **Portão de esquecimento:** utilizado para excluir os dados que não são mais úteis. Funcionamento: duas entradas alimentam este portão (x_t e a saída da célula de memória), elas são multiplicadas pelos respectivos pesos e adicionadas de um viés; o

valor serve de entrada para a função de ativação que possui saída binária, caso o resultado seja 1, a informação é retida, caso seja 0 é esquecida.

- **Portão de saída:** extrai as as informações úteis para a saída do bloco de memória. Funcionamento: possui duas entradas (x_t e a saída da célula de memória), os valores são multiplicados pelos respectivos pesos e enviados para a saída do bloco de memória (DATA, 2019).

Figura 7 – Bloco de memória de rede LSTM



Fonte: Adaptado de LEWIS, 2016, p. 128.

Na Tabela 2 podemos observar as diferenças e semelhanças entre as diferentes técnicas de *deep learning* apresentadas anteriormente.

Tabela 2 – Diferenças e semelhanças entre Redes Neurais de Elman, Redes Neurais de Jordan e *LSTM*

Técnica	**Camada de Entrada**	**Camadas Escondidas**	**Camadas de Contexto**	**Origem da Entrada da Camada de Contexto**	**Blocos de Memória**	**Camada de Saída**
Redes Neurais de Elman	Sim	Sim	Sim	Camada de Contexto	Não	Sim
Redes Neurais De Jordan	Sim	Sim	Sim	Camada de Saída	Não	Sim
LSTM	Sim	Não	Não	Não se Aplica	Sim	Sim

3 EXPERIMENTO

O experimento foi realizado utilizando linguagem de programação Python, versão 3.7.4. Python é uma linguagem interpretada, interativa e orientada a objetos, criada em 1991 (PYTHON, 2019). A escolha da linguagem foi devida à sua facilidade de uso e alta disponibilidade de bibliotecas para trabalhar com análise de dados e deep learning. O ambiente computacional do experimento possui sistema operacional Linux, distribuição Ubuntu, versão 18.04. Foi necessária a instalação de bibliotecas adicionais, que foi feita através do gerenciador de pacotes pip versão 9.0.1 (PIP, 2019), são elas:

- **pandas (versão 0.25.3)**: cuja função é prover estrutura de dados e ferramentas para análise de dados (PANDAS, 2019);
- **numpy (versão 1.17.4)**: pacote para computação científica, contém funções matemáticas e estruturas para utilização de matrizes multi-dimensionais (NUMPY, 2019);
- **pandas_datareader (versão 0.7.0)**: permite acesso a fontes de dados remotas (DATAREADER, 2019). Foi utilizado, no presente trabalho, para download dos dados da base do *Yahoo Finance* (YAHOO, 2019);
- **sklearn (versão 0.21.3)**: provê ferramentas para análise preditiva de dados (LEARN, 2019). Neste experimento, foi empregada para normalização dos dados no intervalo entre 0 e 1 e, para o cálculo do Método dos Mínimos Quadrados;
- **matplotlib (versão 3.1.1)**: biblioteca para criação e visualização de gráficos (MATPLOTLIB, 2019);
- **pyneurgen (versão 0.3.1)**: pacote que contém implementações de redes neurais (PYNEURGEN, 2019), foi utilizado nos experimentos com Redes Neurais de Elman e Redes Neurais de Jordan;
- **pickle (versão 4.0):** módulo para serialização e des-serialização de objetos (PICKLE, 2019), foi aqui utilizado para salvar uma cópia dos modelos de redes *deep learning* obtidos;
- **keras (versão 2.2.4)**: biblioteca com implementação de redes neurais (KERAS, 2019), foi empregada no experimento com a rede LSTM.

O processo de execução do experimento foi dividido em quatro etapas. Sendo a primeira a definição da população e do processo de amostragem; a segunda, a coleta de dados; a terceira foi a etapa de pré-processamento, onde os dados foram adequados para o experimento; a quarta o treinamento e execução do experimento.

3.1 POPULAÇÃO E PROCESSO DE AMOSTRAGEM

O Índice BOVESPA (IBOVESPA) é um índice do mercado de ações brasileiro, o qual é composto pelas ações mais negociadas e com maior volume de comercialização (B3, 2019), serão analisados dados históricos do IBOVESPA (preços de fechamentos diários) em período a ser definido. Os preços de fechamento diários das ações citadas no Apêndice 1, no mesmo período de tempo, foram empregados como variáveis do modelo para previsão dos preços do IBOVESPA.

3.2 COLETA DE DADOS

Os dados do estudo foram obtidos do banco de dados do *Yahoo Finance*[1], site pertencente ao grupo *Yahoo!*, que disponibiliza cotações de ativos e notícias do mercado financeiro (YAHOO, 2019). Disponível para utilização através de API (*Application Programming Interface*) na linguagem de programação *Python*, versão 3.7.4, a qual será empregada ao longo do trabalho.

Foram baixados todos os preços de fechamentos diários do IBOVESPA e das ações da BOVESPA disponíveis no banco de dados do *Yahoo Finance* (YAHOO, 2019), no período considerado. O código para o download dos dados se encontra no arquivo "1-generate_csv_from_tickers.py" do seguinte repositório github: <https://github.com/miya779/comparison_deeplearning_jordan_elman_lstm>. O código desse

[1] https://finance.yahoo.com/

arquivo gera um arquivo de extensão .csv com todo os preços de fechamento das ações da BOVESPA no período considerado.

3.3 PRÉ-PROCESSAMENTO

Algumas das ações apresentavam diversos valores de fechamento diários não disponíveis (em branco). O critério de seleção dos ativos empregados como variáveis independentes do modelo foi a escolha apenas dos ativos que apresentaram menos de 100 preços de fechamentos diários em branco no período. Assim, 198 ativos, além do Índice BOVESPA, atenderam ao critério de seleção, eles estão listados no Apêndice 1.

Os valores dos preços dos dias que não estavam disponíveis no banco de dados foram preenchidos através de interpolação linear, conforme código no Quadro 1.

Quadro 1 – Código Python – Interpolação Linear

```
1.  interpolated_data =
filtered_data.interpolate(method='linear',limit_direction='b
oth')
2.  data = interpolated_data
```

Os preços de fechamento das ações selecionadas da BOVESPA foram utilizados para prever os valores do IBOVESPA. Os preços foram convertidos para logaritmo natural na base e (código no Quadro 2), essa é uma técnica de pré-processamento que apresenta bons resultados para a previsão de séries temporais (BANHATTI; DEKA, 2012), ajudando a normalizar os dados (LEWIS, 2016).

Quadro 2 – Código Python – conversão dos dados para logaritmo natural

```
1.  data = data.apply(np.log)
```

Feito isso os preços foram normalizados de forma a estarem contidos no intervalo de 0 à 1, como mostra o trecho de código no Quadro 3.

Quadro 3 – Código Python – Normalização dos dados no intervalo de 0 à 1

```
1.  scaler_x = MinMaxScaler()
2.  x_train = scaler_x.fit_transform(x_train)
3.  x_test = scaler_x.transform(x_test)
4.  scaler_y = MinMaxScaler()
5.  y_train = scaler_y.fit_transform(y_train)
6.  y_test = scaler_y.transform(y_test)
7. x_input =
np.concatenate((x_train,x_test,np.zeros((1,np.shape(x_train)
[1]))))
8. y_input =
np.concatenate((y_train,y_test,np.zeros((1,1))))
```

3.4 TREINAMENTO E EXECUÇÃO

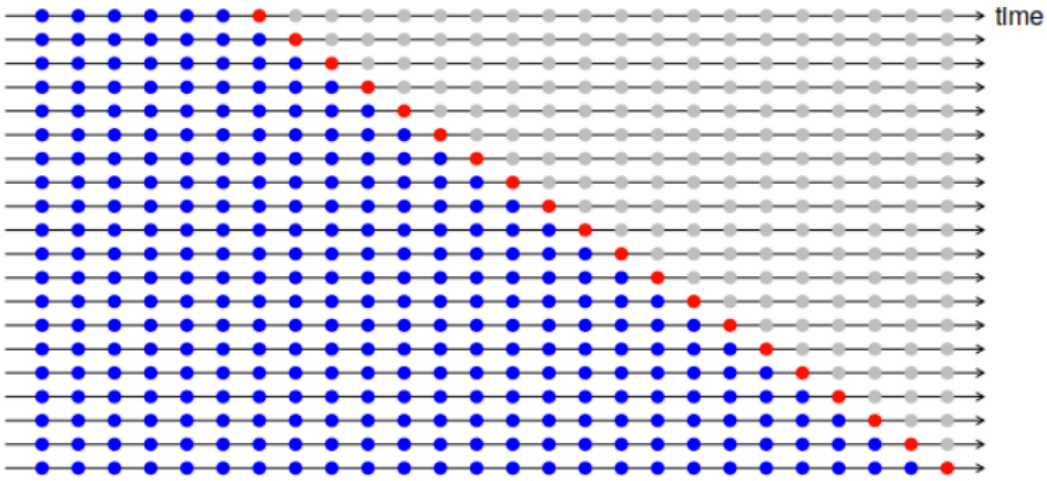
O código com o experimento pode ser consultado no arquivo "2-process.py" do seguinte repositório github: <https://github.com/miya779/comparison_deeplearning_jordan_elman_lstm>.

A precisão das previsões de um modelo é determinada de acordo com a avaliação da sua performance ao analisar novos dados que não utilizados anteriormente em seu cálculo. Com finalidade de calcular os desempenhos foi utilizado o método de *cross-validation*, nele os dados são divididos em dois conjuntos, um de treinamento que irá servir para a elaboração dos modelos e outro de teste. Para avaliar séries temporais, é gerada uma série de conjuntos de teste, sendo que cada um deles irá conter apenas uma observação. Conforme representado pela Figura 8, cada um desses conjuntos de teste será avaliado com um modelo construído com todos os dados ocorridos anteriormente à observação do conjunto de teste analisado

(HYNDMAN; ATHANASOPOULOS, 2018). No presente estudo, os conjuntos de testes foram gerados com os últimos 5% dos dados das séries temporais.

Os modelos matemáticos aqui desenvolvidos foram realizados utilizando Redes Neurais Recorrentes, por serem redes neurais mais adequadas ao tratamento de séries temporais, já que propiciam a existência de uma memória de curto prazo na rede, possibilitando a elaboração de modelos com maior complexidade (LEWIS, 2016).

Figura 8 – Método *cross-validation* para problemas de regressão



Fonte: HYNDMAN; ATHANASOPOULOS, 2018.

Foi realizado um comparativo entre Redes Neurais Recorrentes de Jordan, Redes Neurais Recorrentes de Elman e LSTM; na tarefa de regressão.

Através do Quadro 4, é possível visualizar trecho do código utilizado para o cálculo do modelo da Rede Neural de Jordan.

Quadro 4 – Código Python – Rede Neural de Jordan

```
1. fit1 = NeuralNet()
2.
fit1.init_layers(input_nodes,[hidden_nodes],output_nodes,Elm
anSimpleRecurrent())
3. fit1.randomize_network()
4. fit1.layers[1].set_activation_type('sigmoid')
5. fit1.set_learnrate(0.05)
6. fit1.set_all_inputs(x_input)
7. fit1.set_all_targets(y_input)
8. fit1.set_learn_range(0,i)
9. fit1.set_test_range(i, i+1)
10. fit1.learn(epochs=100, show_epoch_results = True,
random_testing = False)
11. mse = fit1.test()
12. all_mse.append(mse)
13. print("test set MSE = ", np.round(mse,6))
14. target = [item[0][0] for item in
fit1.test_targets_activations]
15. target =
scaler_y.inverse_transform(np.array(target).reshape((len(tar
get),1)))
16. pred = [item[1][0] for item in
fit1.test_targets_activations]
17. pred =
scaler_y.inverse_transform(np.array(pred).reshape((len(pred)
,1)))
18. real_y_test.append(target[0][0])
19. predicted_y_test.append(pred[0][0])
```

Os parâmetros utilizados para em todas as redes neurais foram: nós de entrada = 198 (correspondentes aos 198 ativos analisados); camada escondida = 1; nós na camada escondida = 7; nós de saída = 1; taxa de aprendizado = 0,05; número de épocas = 100.

O critério para definição do número de camadas escondidas foi o número mínimo de camadas possíveis, pois desejamos saber como as previsões se comportam com uma configuração mínima. O número de nós da camada escondida também atendeu ao mesmo critério. O número de nós de saída é apenas um, pois estamos prevendo apenas um resultado (previsão do índice BOVESPA no próximo dia). Para o número de épocas, foram testadas quantidades maiores e menores, para épocas acima de 100 a demora no processamento é maior e não há um aumento de performance considerável, portanto, foi definido a utilização de 100 épocas nos modelos.

Assim, um modelo da Rede Neural de Elman, teria a para cada neurônio da camada escondida um valor z, o qual é a entrada da função de ativação g(z) dado por:

$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^{n} x_j w_{j,i}^{(1)} \qquad (11)$$

Sendo que o índice i tem valor no intervalo 1 até 7 (incluindo 1 e 7), posto que o índice corresponde aos nós da camada escondida. Já o índice j, correspondente aos neurônios da camada de entrada, tem valor no intervalo fechado entre 1 e 396, pois estamos lidando com 198 neurônios na camada de entrada e 198 neurônios da camada de contexto (a qual contém os valores de saída dos neurônios da camada escondida no período de tempo anterior ao analisado t - 1). A camada de saída possui apenas um neurônio, dessa forma, o valor previsto $\hat{y}$ será dado por:

$$\hat{y} = g\left(w_0^{(2)} + \sum_{j=1}^{m} z_j w_j^{(2)}\right) \qquad (12)$$

Mais detalhes sobre os modelos e descrição dos índices e variáveis podem ser vistos na seção de Fundamentação Teórica.

A avaliação de performance dos modelos foi realizada através do Método dos Mínimos Quadrados, descrito anteriormente na seção de Fundamentação Teórica.

4 APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

Os resultados desta seção foram alcançados após a obtenção dos dados e a realização das modelagens utilizando as três redes neurais recorrentes: de Elman, de Jordan e LSTM.

4.1 RESULTADOS OBTIDOS

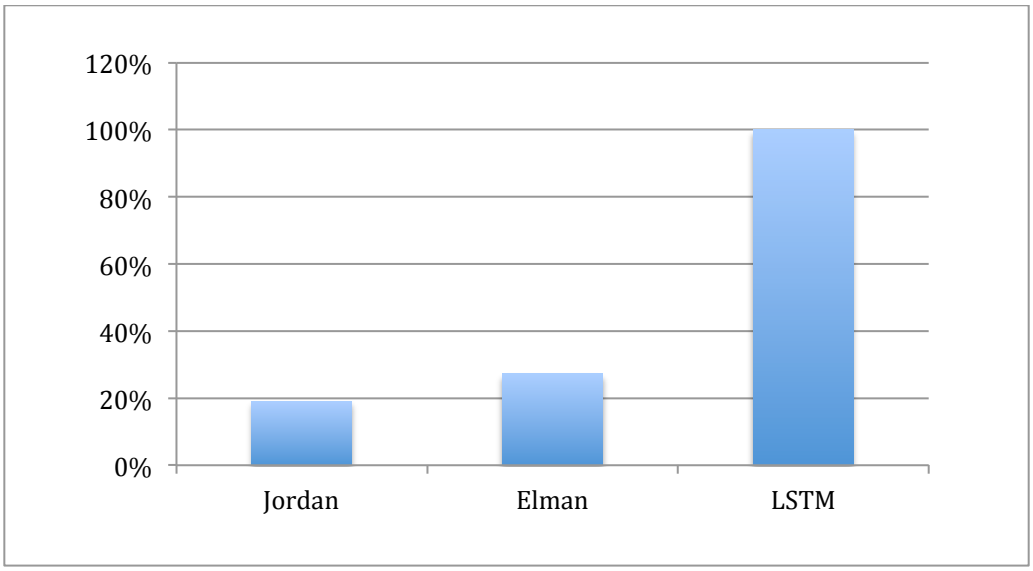
A Tabela 3 mostra o desempenho das Redes Neurais analisadas através do Método dos Mínimos Quadrados. Conforme podemos observar, todas as redes apresentaram um erro quadrático baixo, da ordem de 10^{-4}, com exceção da LSTM, que apesar de apresentar erro quadrático baixo, este foi da ordem de 10^{-3}. Este resultado indica que todos os métodos apresentaram desempenhos satisfatórios.

Tabela 3 – Desempenho das Redes Neurais analisadas utilizando o Método dos Mínimos Quadrados

Redes Neurais	**De Elman**	**De Jordan**	**LSTM**
MMQ	$3{,}2483*10^{-4}$	$2{,}2777*10^{-4}$	$1{,}1942*10^{-3}$

O Gráfico 1 facilita a visualização dos melhores desempenhos em termos de porcentagem, nele, a Rede LSTM, a qual obteve a pior performance, é considerada como 100%, as outras Redes Neurais obtiveram uma performance melhor. É importante ressaltar que foi atribuído o valor de 100% para a pior performance, assim, a Rede que apresentou o melhor desempenho é a que apresenta menor porcentagem no gráfico. É possível perceber que a Rede Neural de Jordan mostrou um desempenho superiormente considerável em comparação com a Rede LSTM. Já entre as Redes Neurais de Jordan e de Elman a diferença entre os desempenhos é pouca.

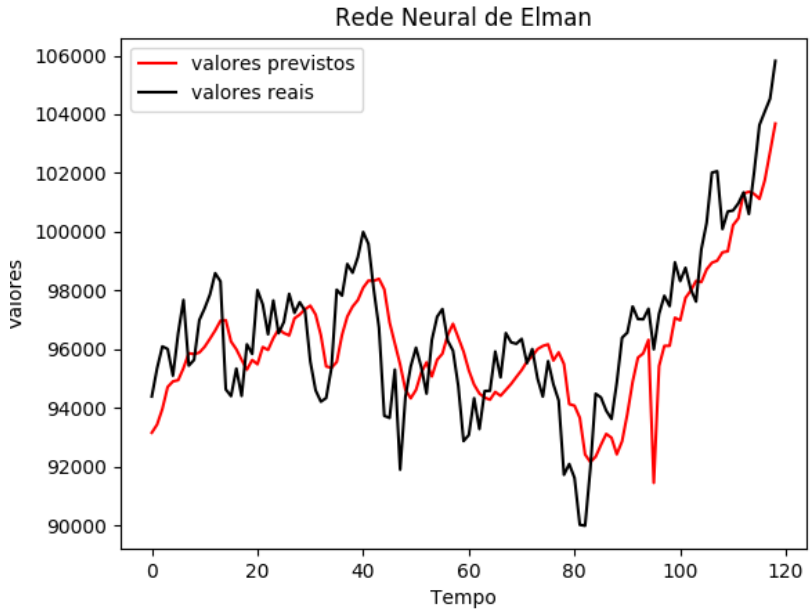
Gráfico 1 – Comparativo entre os desempenhos das Redes Neurais, tomando a Rede LSTM como 100%



Fonte: Elaboração da autora, 2019.

O Gráfico 2, o Gráfico 3 e o Gráfico 4 (elaborados utilizando a linguagem *Python* e a biblioteca *matplotlib*) exibem os valores de fechamento reais do Índice BOVESPA (na cor preta) e os valores previstos por cada arquitetura de rede neural (na cor vermelha).

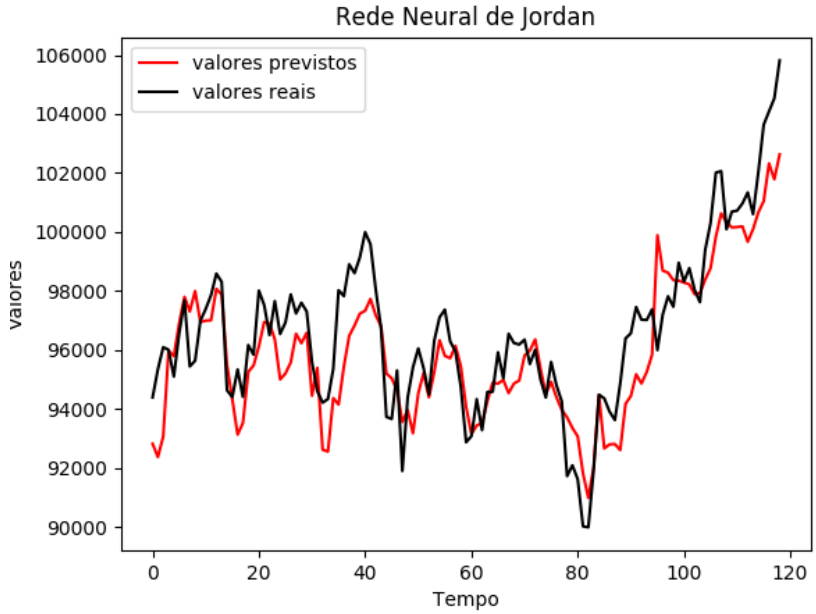
Gráfico 2 – Valores diários de fechamento reais do Índice BOVESPA e valores previstos pela Rede Neural de Elman, ao longo do período considerado.



Fonte: Elaboração da autora, 2019.

Conforme é possível ver no Gráfico 2, temos uma aproximação razoável entre os valores previstos pela Rede Neural de Elman e os valores reais.

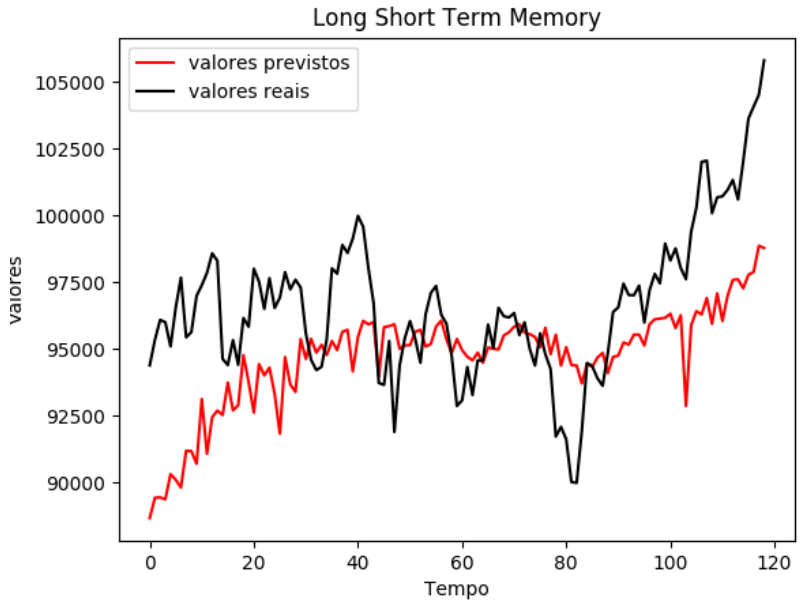
Gráfico 3 – Valores diários de fechamento reais do Índice BOVESPA e valores previstos pela Rede Neural de Jordan, ao longo do período considerado.



Fonte: Elaboração da autora, 2019.

O Gráfico 3 também apresenta uma boa aproximação entre os valores previstos pela Rede Neural de Jordan e os valores reais.

Gráfico 4 – Valores diários de fechamento reais do Índice BOVESPA e valores previstos pela Rede LSTM, ao longo do período considerado.



Fonte: Elaboração da autora, 2019.

Observando o Gráfico 4 podemos perceber que os valores de fechamento previstos pela Rede Neural LSTM seguiram um padrão semelhante de picos e vales dos valores reais, porém são percebidas diferenças consideráveis entre os dois valores.

O Gráfico 2 e o Gráfico 3 (resultados das previsões das Redes Neurais de Elman e de Jordan) mostram valores de fechamentos previstos com menor diferença do que o Gráfico 4 (LSTM). Também é perceptível nos gráficos que as previsões efetuadas pela Rede Neural de Jordan estão com melhor acurácia do que as efetuadas pela Rede Neural de Elman. Essas observações são consistentes com os desempenhos calculados pelo Método dos Mínimos Quadrados apresentados pelas redes neurais.

A Rede Neural de Jordan apresentou a melhor performance geral dentre as arquiteturas investigadas. Esse desempenho pode ser devido ao período considerado, às ações usadas como variáveis independentes do modelo, ao ativo previsto e outros fatores. Portanto, não é possível afirmar que a Rede Neural de Jordan irá apresentar melhor performance do que as outras arquiteturas em diferentes condições. Para tal confirmação, uma investigação mais ampla deveria ser conduzida.

5 CONCLUSÕES E CONSIDERAÇÕES FINAIS

Foi possível avaliar o desempenho dos métodos: LSTM, Redes Neurais Recorrentes de Jordan e Redes Neurais Recorrentes de Elman na previsão de preços do Índice BOVESPA. Dentre as arquiteturas empregadas na comparação, a que apresentou melhor desempenho foi a Rede Neural de Jordan. Contrariando uma ampla gama de estudos os quais apontaram que em diversos outros contextos, as redes LSTM apresentaram desempenhos superiores a outras arquiteturas (LEWIS, 2016). O presente trabalho, apenas investiga uma configuração específica para as redes neurais, mudanças na quantidade de camadas escondidas, número de neurônios em cada camada, etc. podem levar a outros resultados. Sendo possível que com variação nos parâmetros outra arquitetura demonstre resultado superior.

Para o melhor entendimento do comportamento do Índice BOVESPA recomenda-se a elaboração de novas pesquisas, utilizando outras arquiteturas, configurações e variáveis não empregadas. A inclusão de Processamento de Linguagem Natural no modelo *deep learning* seria de grande valia para uma melhor previsão, posto que, notícias e informações sobre economia têm um grande peso no preço das ações de bolsas de valores.

REFERÊNCIAS

AMINI, A. In: MIT 6.S191 – MIT's introductory course on deep learning methods and applications. **Introduction to Deep Learning**. 2019. Disponível em: < https://www.youtube.com/watch?v=5v1JnYv_yWs>. Acesso em 2 de setembro de 2019.

ANDRADE, V. A; TAVARES, C. B. **Avaliação de investimento sob a ótica das finanças empresariais**. Revista de administração em diálogo, 2010, v. 12, n. 2, p. 43-60.

B3. **Índice Bovespa (IBOVESPA).** Disponível em: <http://www.b3.com.br/pt_br/market-data-e-indices/indices/indices-amplos/ibovespa.htm>. Acesso em 5 de setembro de 2019.

BANHATTI, A. G.; DEKA, P. C. **Performance evaluation of artificial neural network model using data preprocessing in non-stationary hydrologic time series**. CiiT International Journal of Artificial Intelligent Systems and Machine Learning, Abril 2012, v. 4, n. 4, p. 223-229.

BISHOP, C. M. **Pattern recognition and machine learning**: 1 ed. Nova Yorque: Springer, 2006.

CAMPOS, C. R. et al. Educação estatística no contexto da educação crítica. Bolema de educação matemática, 2011, v. 24, n. 39, p. 473-494.

CHEN, Kai; ZHOU, Yi; DAI, Fangyan. **A LSTM-based method for stock returns prediction: A case study of China stock market.** 2015 IEEE International Conference on Big Data (Big Data). Washington: IEEE Computer Society, 2015, p. 2823-2824.

DATA Science Academy. **Deep Learning Book**, 2019. Disponível em: <http://www.deeplearningbook.com.br/>. Acesso em 29 de outubro de 2019.

DATAREADER, Pandas. **Pandas-datareader**. Disponível em: <https://pandas-datareader.readthedocs.io/en/latest/>. Acesso em 4 de dezembro de 2019.

FACURE, M. **Funções de ativação: entendendo a importância da ativação correta nas redes neurais**. 2017. Disponível em https://matheusfacure.github.io/2017/07/12/activ-func/. Acesso em 3 de setembro de 2019.

FADDA, S.; CAN, M. **Forecasting conditional variance of S&P100 returns using feedforward and recurrent neural networks**. Southeast Europe journal of soft computing, 2017, v. 6, n. 1, p. 58-69.

FONSECA, J. J. S. **Metodologia da pesquisa científica**. Fortaleza: UECE, 2002, p. 20.

GARCIA, D. **Análise do comportamento de marmorização de cartão revestido usando modelagem matemática em redes neurais**. Dissertação de mestrado. Universidade Federal do Paraná. Curitiba, 2010.

GONÇALVES, M. B.; FLEMMING, D. M. **Cálculo B: Funções de várias variáveis, integrais múltiplas, integrais curvilíneas e de supervfície**. 2 ed. São Paulo: Pearson Prentice Hall, 2007.

GOODFELLOW, I; et al. **Deep Learning**. MIT Press, 2016. Disponível em https://www.deeplearningbook.org/ . Acesso em 9 de julho de 2019.

GREFF, K. et al. **LSTM: A search space odyssey**. IEEE Transactions on neural networks and learning systems, 2017, v. 28, n. 10, p. 2222-2232.

HABERLER, Gottfried. **Crescimento Econômico e Estabilidade: uma análise da evolução e das políticas econômicas**. 1 ed. Rio de Janeiro: Zahar, 1976.

HYNDMAN, R. J.; ATHANASOPOULOS, G. **Forecasting: Principles and Pratice**. 2 ed. Melbourne, Australia: OTexts, 2018.

KERAS. **Keras: The Python Deep Learning Library**. Disponível em: <https://keras.io/>. Acesso em 4 de dezembro de 2019.

LEARN, Scikit. **Scikit Learn**. Disponível em: <https://scikit-learn.org/stable/>. Acesso em 4 de dezembro de 2019.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. **Deep Learning**. Nature, 2015, v. 521, p. 436-444.

LEWIS, N. D. **Deep Time Series Forecasting with Python**. CreateSpace Independent Publishing Platform, 2016.

LORENA, A. C.; CARVALHO, A. C. P. L. F. **Uma introdução às Support Vector Machines**. Revista de Informática Teórica e Aplicada, 2007, v. 14, n. 2, p. 43-67.

LUQUET, M. **Guia valor econômico de finanças pessoais**: 2 ed. São Paulo: Editora Globo, 2007.

MARUMO, F. S. **Deep learning para classificação de fake news por sumarização de texto**. Trabalho de conclusão de curso. Universidade Estatual de Londrinda. Londrina, 2018.

MATPLOTLIB. Disponível em: <https://matplotlib.org/>. Acesso em 4 de dezembro de 2019.

MESQUITA, Caio M.; OLIVEIRA, Renato; PEREIRA, Adriano C. M. **Utilização de uma rede neural LSTM e testes da razão da variância para previsões em séries de ativos da BOVESPA**. Workshop of Artificial Intelligence Applied to Finance, 2019. Disponível em http://www.comp.ita.br/labsca/waiaf/papers/CaioMesquita_paper_11.pdf . Acesso em 19 de julho de 2019.

MIRANDA, A. P.; CORONEL, D. A.; VIEIRA, K. M. **Previsão do mercado futuro do café arábica utilizando redes neurais e métodos econométricos**. Revista de estudos do CEPE, 2013, n. 38, p. 66-98.

MITCHELL, Tom M. **Machine Learning**. McGraw Hill, 1997.

MOREIRA, M. A. **Modelos científicos, modelos mentais, modelagem computacional e modelagem matemática: aspectos epistemológicos e implicações para o ensino**. Revista brasileira de ensino de ciência e tecnologia, 2014, v. 7, n. 2, p. 1-20.

MUNTASER, J. G. S., et al. Aplicação de Redes Neurais na Previsão das Ações do Setor de Petróleo e Gás da BM&FBovespa. Revista FSA – Periódico do Centro Universitário Santo Agostinho. v. 14, n.6, nov/dez. 2017. Disponível em http://www4.fsanet.com.br/revista/index.php/fsa/article/view/1456 . Acesso em 9 de julho de 2019.

MURPHY, Kevin P. **Machine Learning – A Probabilistic Perspective**. London: The MIT Press, 2012.

NAEINI, M. P.; TAREMIAN, H; HASHEMI, H. B. **Stock market value prediction using neural networks**. International Conference on Computer Information Systems and Industrial Management Applications, 2010.

NELSON, D. M. Q. **Uso de Redes Neurais Recorrentes para Previsão de Séries Temporais Financeiras**. Dissertação de mestrado. UFMG. Belo Horizonte, 2017.

NELSON, D. M. Q.; PEREIRA, A., C. M.; OLIVEIRA, R. A. **Stock market's price movement prediction with LSTM neural networks**. International Joint Conference on Neural Networks, 2017.

NUMPY. Disponível em: <https://numpy.org/>. Acesso em 4 de dezembro de 2019.

PANDAS. **Python Data Analysis Library**. Disponível em: <https://pandas.pydata.org/>. Acesso em 4 de dezembro de 2019.

PIAZZA, Marcelo C. **Bem-vindo à Bolsa de Valores**: 7 ed. São Paulo: Editora Novo Conceito, 2008.

PICKLE. **Python Object Serialization**. Disponível em: <https://docs.python.org/3/library/pickle.html>. Acesso em 4 de dezembro de 2019.

PIP. Disponível em: < https://pypi.org/project/pip/>. Acesso em 4 de dezembro de 2019.

PYNEURGEN. **Python Neural Genetic Algorithm Hybrids**. Disponível em: <http://pyneurgen.sourceforge.net />. Acesso em 4 de dezembro de 2019.

PYTHON. **General Python FAQ**. Disponível em: <https://docs.python.org/3.8/faq/general.html>. Acesso em 10 de dezembro de 2019.

ROONDIWALA, Murtaza; PATEL, Harshal; VARMA, Shraddha. **Predicting Stock Prices Using LSTM**. International Journal of Science and Research, 2017, v. 6, n. 4, p. 1754-1756.

ROSENBLATT, F. **The Perceptron: a probabilistic model for information storage and organization in the brain**. Psychological Review, 1958, v. 65, n. 6, p. 386-408.

RUDER, S. **An overview of gradient descent optimization algorithms**. 2016. Disponível em: <https://arxiv.org/pdf/1609.04747.pdf>. Acesso em 28 de ouburo de 2019.

SELVIN, Sreelekshmy, et al. S**tock price prediction using LSTM, RNN and CNN-sliding window model**. International Conference on Advances in Computing, Communications and Informatics, 2017.

TOLEDO, Cristiane S. **A importância do Mercado de Ações para o Crescimento Econômico do Brasil**. Trabalho de Conclusão de Curso. UFSC. Santa Catarina, 2006.

YAHOO Finance. Disponível em: <https://finance.yahoo.com/>. Acesso em 5 de setembro de 2019.

WERNER, Liane; BISOGNIN, Cleber; ARAUJO, Cristiano W. **Análise de técnicas de previsão: um estudo de caso para o volume de ações da Petrobras**. VIII Congresso Brasileiro de Engenharia de Produção. Ponta Grossa, 2018. Disponível em https://www.lume.ufrgs.br/bitstream/handle/10183/187956/001083943.pdf?sequence=1&isAllowed=y . Acesso em 19 de julho de 2019.

WYSOCKI, A.; LAWRYNCZUK, M. **Two- and Three-Layer Recurrent Elman Neural Networks as Models of Dynamic Processes**. In: SZEWCZYK, R.; ZIELINSKI, C.; KALICZYNSKA, M. Kaliczyńska M. Challenges in Automation, Robotics and Measurement Techniques. ICS. Advances in Intelligent Systems and Computing, vol. 440. Springer, 2016. p. 165-175.

APÊNDICES

APÊNDICE A – Código das ações utilizadas nas modelagens de redes neurais recorrentes

^BVSP; PETR4.SA; CIEL3.SA; ABEV3.SA; ITUB4.SA; ITSA4.SA; KROT3.SA; B3SA3.SA; VALE3.SA; OIBR3.SA; BBDC4.SA; VVAR3.SA; PETR3.SA; BBAS3.SA; RAIL3.SA;JBSS3.SA; CMIG4.SA; CCRO3.SA; GGBR4.SA; CSNA3.SA; UGPA3.SA; BRML3.SA; GOAU4.SA; GNDI3.SA; CRFB3.SA; TIMP3.SA; FNOR11.SA; LREN3.SA; MRVE3.SA; SUZB3.SA; USIM5.SA; GOLL4.SA; ECOR3.SA; BIDI4.SA; BBSE3.SA; LAME4.SA; BOVA11.SA; EMBR3.SA; BRFS3.SA; WEGE3.SA; SMLS3.SA; MULT3.SA; CYRE3.SA; MRFG3.SA; NATU3.SA; BTOW3.SA; BBDC3.SA; BEEF3.SA; RENT3.SA; ELET3.SA; ESTC3.SA; QUAL3.SA; LIQO3.SA; EGIE3.SA; SBSP3.SA; AZUL4.SA; TOTS3.SA; BRDT3.SA; ELET6.SA; VIVT4.SA; BRKM5.SA; DTEX3.SA; CPFE3.SA; ENBR3.SA; HYPE3.SA; HBOR3.SA; IGTA3.SA; ENGI11.SA; MYPK3.SA; RAPT4.SA; SLCE3.SA; CSAN3.SA; TRPL4.SA; CAML3.SA; POMO4.SA; BRAP4.SA; EVEN3.SA; AMAR3.SA; CESP6.SA; EQTL3.SA; PCAR4.SA; SANB11.SA; FLRY3.SA; JHSF3.SA; PRIO3.SA; SAPR4.SA; LINX3.SA; CMIG3.SA; CVCB3.SA; BPAN4.SA; ODPV3.SA; STBP3.SA; RADL3.SA; OIBR4.SA; IRBR3.SA; MGLU3.SA; HGTX3.SA; BRSR6.SA; DMMO3.SA; ALSC3.SA; SULA11.SA; GUAR3.SA; GRND3.SA; ITUB3.SA; TCSA3.SA; MOVI3.SA; ENEV3.SA; CNTO3.SA; VULC3.SA; CPLE6.SA; ABCB4.SA; TECN3.SA; SMTO3.SA; LUPA3.SA; HAPV3.SA; PTBL3.SA; LAME3.SA; VIVR3.SA; GFSA3.SA; ENAT3.SA; DIRR3.SA; JPSA3.SA; LIGT3.SA; TRPN3.SA; OMGE3.SA; PDGR3.SA; LCAM3.SA; EZTC3.SA; SEER3.SA; WIZS3.SA; BKBR3.SA; MILS3.SA; KLBN4.SA; PFRM3.SA; TPIS3.SA; ALPA4.SA; CSMG3.SA; RLOG3.SA; LOGN3.SA; PSSA3.SA; JSLG3.SA; MDIA3.SA; LPSB3.SA; VLID3.SA; BRPR3.SA; SHOW3.SA; MEAL3.SA; TEND3.SA; TUPY3.SA; LOGG3.SA; FJTA4.SA; FHER3.SA; LEVE3.SA; AZEV4.SA; GSHP3.SA; AALR3.SA; CARD3.SA; TRIS3.SA; TGMA3.SA; CPLE3.SA; UNIP6.SA; ANIM3.SA; ARZZ3.SA; FESA4.SA; FSRF11.SA; VIVT3.SA; SMAL11.SA; PINE4.SA; GBIO33.SA; PARD3.SA; KLBN3.SA; BOVV11.SA; GOAU3.SA; RCSL4.SA; BBRK3.SA; ITSA3.SA; EUCA4.SA; RSID3.SA; POSI3.SA; ROMI3.SA; MXRF11.SA; SGPS3.SA; FRAS3.SA; PIBB11.SA; TIET4.SA; KNRE11.SA; WSON33.SA; GPCP3.SA; ATOM3.SA; FLMA11.SA; FRTA3.SA; KEPL3.SA; SHUL4.SA; RBRF11.SA; GGBR3.SA; CELP3.SA; AGRO3.SA; PTNT4.SA; IVVB11.SA; SAPR3.SA; CRIV4.SA; DIVO11.SA; IDNT3.SA; CARE11.SA; BRCR11.SA; BMEB4.SA; BOBR4.SA; BEES3.SA; COCE5.SA; FJTA3.SA;

BRIV4.SA; PMAM3.SA; USIM3.SA; KNRI11.SA; HGLG11.SA; KNCR11.SA; MNPR3.SA; APER3.SA; CESP3.SA; ETER3.SA; BBPO11.SA; SQIA3.SA; BIOM3.SA; TBOF11.SA; POMO3.SA; RDNI3.SA; PHMO34.SA; HAGA4.SA; IDVL4.SA; RNEW4.SA; HFOF11.SA; VISC11.SA; AZEV3.SA; ALUP3.SA; BCFF11.SA; BRKM3.SA; SSBR3.SA; NVDC34.SA; HGBS11.SA; KNIP11.SA; PYPL34.SA; KMIC34.SA; EBAY34.SA; UCAS3.SA; HGRE11.SA; RPMG3.SA; CRIV3.SA; AVGO34.SA; GGRC11.SA; MALL11.SA; SANB4.SA; NEXT34.SA; MOOO34.SA; WGBA34.SA; FIVN11.SA; BSEV3.SA; BRIV3.SA; VRTA11.SA; BRAP3.SA; CSXC34.SA; OFSA3.SA; TIET3.SA; SDIL11.SA; LOWC34.SA; SANB3.SA; CAMB4.SA; ALUP4.SA; OXYP34.SA; ABTT34.SA; CNIC34.SA; FIGS11.SA; CHME34.SA; CGAS5.SA; INEP3.SA; UBSR11.SA; ARMT34.SA; BAZA3.SA; TJXC34.SA; SIMN34.SA; JSRE11.SA; BPFF11.SA; BBVJ11.SA; HGCR11.SA; EALT4.SA; UNIP3.SA; XTED11.SA; BKNG34.SA; CHCM34.SA; RNGO11.SA; MUTC34.SA; MFII11.SA; TAEE4.SA; AAPL34.SA; RPAD6.SA; RAPT3.SA; CGRA4.SA; CSCO34.SA; INTU34.SA; ENGI4.SA; VRTX34.SA; SAAG11.SA; SPGI34.SA; CAON34.SA; AFLT3.SA; ENGI3.SA; COCA34.SA; PLAS3.SA; NOCG34.SA; INEP4.SA; BRAX11.SA; MMXM3.SA; FDMO34.SA; TAEE3.SA; SPTW11.SA; ADBE34.SA; TESA3.SA; ORCL34.SA; RPAD3.SA; OUJP11.SA; RNEW3.SA; MCOR34.SA; ADHM3.SA; CTNM4.SA; UPAC34.SA; STZB34.SA; EDGA11.SA; SPXI11.SA; WHRL4.SA; BCRI11.SA; MDLZ34.SA; DEEC34.SA; SCAR3.SA; MMMC34.SA; DHER34.SA; HOME34.SA; UPSS34.SA; TGAR11.SA; BRSR3.SA; CRPG5.SA; TGTB34.SA; FOFT11.SA; CGRA3.SA; LLIS3.SA; REGN34.SA; PGCO34.SA; TMOS34.SA; ELPL3.SA; HTMX11.SA; DGCO34.SA; IGBR3.SA; SLED4.SA; LILY34.SA; RCSL3.SA; CPTS11B.SA; MTSA4.SA; ELCI34.SA; CLSC4.SA; XPCM11.SA; EQIX34.SA; BBRC11.SA; CTKA4.SA; TRXL11.SA; ONEF11.SA; GRLV11.SA; VLOL11.SA; TXRX4.SA; FVPQ11.SA; FEXC11.SA; SCHW34.SA; TRPL3.SA; FMXB34.SA; CBOP11.SA; SBUB34.SA; NIKE34.SA; BOEI34.SA; ELEK4.SA; PSVM11.SA; QCOM34.SA; GOGL34.SA; SLBG34.SA; CCXC3.SA; BCIA11.SA; JBDU4.SA; FIIB11.SA; ABCP11.SA; FFCI11.SA; TELB4.SA; FVBI11.SA; JRDM11.SA; CEOC11.SA; CEBR6.SA; BLAK34.SA; BONY34.SA; METB34.SA; SCPF11.SA; FIND11.SA; BOAC34.SA; CTSA3.SA; WHRL3.SA; REDE3.SA; RBGS11.SA; THRA11.SA; GOGL35.SA; COLG34.SA; AXPB34.SA; EXXO34.SA; BEES4.SA; CEBR3.SA; IBMB34.SA; WALM34.SA; RPAD5.SA; HOOT4.SA; ENMT4.SA; PRSV11.SA; RBRD11.SA; HCRI11.SA; ISUS11.SA; BAUH4.SA; CRPG6.SA; UTEC34.SA; CLGN34.SA; CTSA4.SA; MATB11.SA; BTTL3.SA; TELB3.SA; BPAC3.SA;

EMAE4.SA; MMXM11.SA; HONB34.SA; DUKB34.SA; JPMC34.SA; CEPE5.SA; FIXX11.SA; SLED3.SA; RANI3.SA; MGEL4.SA; ACNB34.SA; FSPE11.SA; FSTU11.SA; WFCO34.SA; BMIN4.SA; COWC34.SA; GOVE11.SA; RIGG34.SA; CMCS34.SA; CCPR3.SA; CATP34.SA; MSFT34.SA; WUNI34.SA; RBBV11.SA; DEAI34.SA;
MELI34.SA; MCDC34.SA; BDLL4.SA; ATTB34.SA; SULA3.SA; GEOO34.SA; GEPA4.SA; ENMT3.SA; FBOK34.SA; SULA4.SA; GMCO34.SA; COTY34.SA; BGIP3.SA; NFLX34.SA; RBCB11.SA; BMYB34.SA; GILD34.SA; ABBV34.SA; UNIP5.SA; PEAB3.SA; BPAC5.SA; FRIO3.SA; ALPA3.SA; ITLC34.SA; BBFI11B.SA; SNSY5.SA; GDBR34.SA; MSCD34.SA; CEPE6.SA; LMTB34.SA; CESP5.SA; CELP6.SA; BRGE3.SA; BGIP4.SA; CPRE3.SA; BMEB3.SA; CTXT11.SA; RBVO11.SA; BNFS11.SA; NVHO11.SA; BMLC11B.SA; RBDS11.SA; RANI4.SA; AMZO34.SA; VERZ34.SA; CELP7.SA; DASA3.SA; CHVX34.SA; CELP5.SA; TEXA34.SA; ROST34.SA; COPH34.SA; USBC34.SA; SOND5.SA; NSLU11.SA; FAED11.SA; CNES11.SA; PORD11.SA; FIIP11B.SA; FAMB11B.SA; JNJB34.SA; VISA34.SA; CTGP34.SA; BRGE12.SA; MACY34.SA; MNDL3.SA; UBSG34.SA; CEEB3.SA; CGAS3.SA; TSLA34.SA; MRCK34.SA; OSXB3.SA; UNHH34.SA; KHCB34.SA; CSRN3.SA; MEND5.SA; PATI4.SA; LIPR3.SA; PNCS34.SA; SSFO34.SA; CRDE3.SA; BNBR3.SA; ECPR3.SA; PFIZ34.SA; JBDU3.SA; DOHL4.SA; EEEL3.SA; VLOE34.SA; BERK34.SA; DISB34.SA; PEPB34.SA; EALT3.SA; BAHI3.SA; TOYB4.SA; FDXB34.SA; MTIG4.SA; FPAB11.SA; MBRF11.SA; EURO11.SA; PABY11.SA; FESA3.SA; GPIV33.SA; EXGR34.SA; AVON34.SA; TOYB3.SA; AMGN34.SA; CSRN6.SA; GSGI34.SA; PNVL3.SA; ADPR34.SA; CEBR5.SA; ESTR4.SA; EKTR4.SA; HETA4.SA; JFEN3.SA; PEAB4.SA; TEKA4.SA; ENMA3B.SA; CBEE3.SA; HBTS5.SA; BRGE11.SA; NORD3.SA; GPAR3.SA; AIGB34.SA; AHEB3.SA; MOSC34.SA; CTNM3.SA; TWTR34.SA; PATI3.SA; TKNO4.SA; WLMM4.SA; ALMI11.SA; WPLZ11.SA; SHPH11.SA; PQDP11.SA; MAXR11.SA; CXRI11.SA; PLRI11.SA; XBOV11.SA; CXCE11B.SA; CXTL11.SA; RNDP11.SA; FCFL11.SA; EDFO11B.SA; RDES11.SA; ECOO11.SA; FMOF11.SA; TRNT11.SA; FLRP11.SA; DOMC11.SA; SAIC11B.SA; NPAR11.SA; DRIT11B.SA; BMKS3.SA; ATSA11.SA; BOVA.SA; PIBB.SA; DIVO.SA; ECOO.SA; DDNB34.SA; GOVE.SA; MATB.SA; BOVV.SA; KMBB34.SA; ISUS.SA; XBOV.SA; SMAL.SA; FIND.SA.