ESTUDO COMPARATIVO ENTRE ALGORITMOS DE MACHINE LEARNING APLICADOS À PREVISÃO DE SÉRIES TEMPORAIS DO MERCADO FINANCEIRO.

Paulo Cesar Pereira Alves[1], Sandra Cristina Costa Prado[2]

[1] Discente do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas / paulo.alves39@fatec.sp.gov.br
[2] Docente do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas / sandra.prado01@fatec.sp.gov.br

RESUMO

No presente trabalho foi realizado um estudo comparativo da capacidade preditiva entre algoritmos de Aprendizado de Máquina aplicados a um problema de regressão chamado Previsão de Séries Temporais. Especificamente, foram utilizados dados que compõem uma Série Temporal de Mercado Financeiro, a partir dos quais tenta-se prever o volume de negociação do mercado que é um dos indicadores mais importantes em problemas financeiros. Foram utilizados três métodos de Aprendizado de Máquina: os métodos de Regressão Linear e Redes Neurais Convencionais adaptados ao contexto de Séries Temporais e o método das Redes Neurais Recorrentes. Todas as implementações foram realizadas usando pacotes da linguagem Python específicos para Aprendizado de Máquina, Aprendizagem Profunda e Análise e Tratamento de Séries Temporais. Utilizando a métrica estatística R2 foi possível realizar uma análise comparativa entre os três algoritmos sob algumas condições de tratamento dos dados, como a normalização da série temporal.

Palavras-chave: Aprendizado de máquinas; série temporal; mercado financeiro.

1 INTRODUÇÃO

A modernização e a crescente quantidade de dados produzidos diariamente têm trazido diversas oportunidades para profissionais e empresas que desejam utilizar essas informações de maneira estratégica. A aplicação da análise de dados, mais especificamente a análise preditiva, permite ao usuário identificar os principais comportamentos do mercado e suas possíveis tendências. A inserção da tecnologia neste contexto permite que uma grande quantidade de dados seja processada e analisada de maneira rápida e eficaz, contribuindo de forma significativa na tomada de decisões.

A inteligência Artificial, subárea da computação, possui diversas ferramentas e vertentes que ao serem implementadas, possibilitam a identificação e classificação desses dados, e principalmente a identificação de padrões através do aprendizado de máquina, utilizando métodos estatísticos e matemáticos aliados à programação de software (SOUZA, 2017).

Através dessas ferramentas que buscam extrair padrões de dados, muitos modelos de aprendizado de máquina têm sido elaborados com o intuito de capturar o comportamento das ações do mercado financeiro (MESQUITA, 2019).

Predizer preços de ações é sabidamente um problema muito difícil, mas predizer o volume de compra e vendas de ativos de um mercado financeiro em função do histórico recente, se torna mais fácil, além de oferecer muita utilidade para planejar estratégias de compra e venda de ativos (LEBARON E WEIGEND, 1998).

2 REFERENCIAL TEÓRICO

2.1 Inteligência Artificial

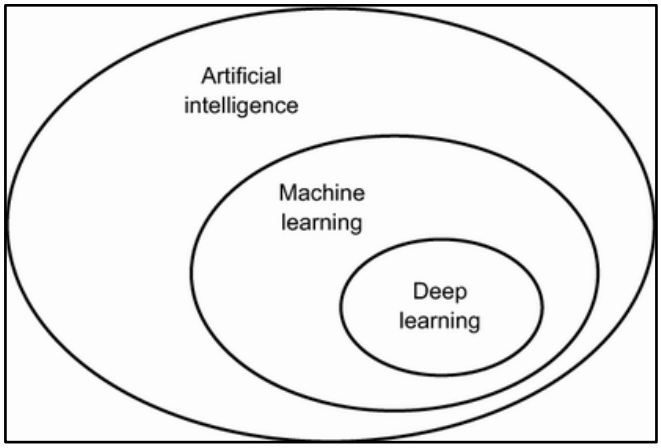
O avanço tecnológico, com a aceleração das máquinas em processamento e em quantidade de dados armazenados, é de prática que muitos países e empresas investem em grandes estudos voltados à inteligência artificial para melhor utilização desses dados, com isso criando sistemas cada vez mais automatizados e adaptativos.

Para os pesquisadores, a mente humana funciona como um computador. É possível construir programas que imitem nossa capacidade de raciocinar, identificar objetos, falar, compreender nossa linguagem, além de realizar outras tarefas que requerem inteligência, e de uma maneira similar e próxima do modo como os seres humanos fazem (TEIXEIRA, 1990).

A inteligência artificial surgiu oficialmente em 1956 quando um professor de matemática John McCarthy organizou um workshop de verão em Dartmouth College, NH, USA (CHOLLET, 2021). Desde então, este campo abrangente do conhecimento vem passando por um rápido desenvolvimento em suas técnicas e relevância, estando presente em diversos aspectos do nosso dia a dia (IGNACIO, 2021).

A inteligência artificial abrange o aprendizado de máquina (machine learning) e o aprendizado profundo (Deep Learning), mas também pode envolver atividades sem aprendizado.

Figura 1 - Inteligência Artificial



Fonte: CHOLLET, 2021, p 2

2.1.1 Machine Learning

São frequentes as aplicações em que as análises requerem metodologias mais elaboradas. Métodos tradicionais não são capazes de lidar de forma satisfatória com bancos de dados em que há mais covariáveis que observações, uma situação muito comum nos dias de hoje. Os avanços computacionais recentes permitiram que novas metodologias fossem exploradas (IZBICKI E SANTOS, 2020).

Um novo conceito foi iniciado quando começou a ter o paradigma de regras no lugar de respostas dos algoritmos, implantando características adaptativas, com isso definindo o que é machine learning: sistemas que melhoraram seu desempenho em uma determinada tarefa com mais e mais experiências ou dados (CHOLLET, 2021).

Um sistema de Machine Learning é treinado em vez de explicitamente programado. Ele é apresentado com muitos exemplos relevantes para uma tarefa, e encontra estrutura estatística nesses exemplos que eventualmente permite que o sistema venha com regras para automatizar a tarefa. (Chollet, 2021).

O estudo de aprendizado de máquina envolve diferentes áreas do conhecimento tais como Matemática, Estatística, Computação e Otimização, sendo assim um campo multidisciplinar (IGNACIO, 2021).

2.1.2 Deep Learning

Com os avanços dos neurônios e arquiteturas de redes neurais artificiais, um novo conceito surgiu para entrar no subcampo da machine learning, o deep learning. O “profundo” em deep learning não é uma referência a tipos mais profundos alcançada pela abordagem, e vez disso, representa essa ideia de sucessivas camadas de representações (CHOLLET, 2021).

Deep learning usa camadas de neurônios matemáticos para processar dados utilizando as redes neurais (neural network). Com a implementação da função de ativação e a correção dos pesos e baías com o processo de backpropagation, o aprendizado profundo (deep learning) começou a ter mais representatividade no campo da IA.

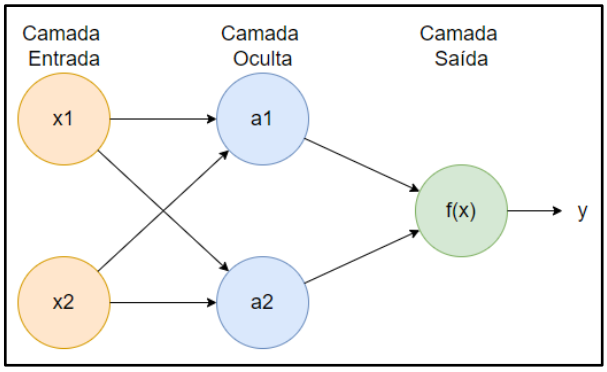
Uma rede neural é resumida em 3 camadas, a primeira é a de entrada, no qual entram os dados a serem analisados, a segunda é a camada oculta, essa responsável pela lógica do Deep Learning, nessa camada irá ter um tipo de função de ativação e a última é a camada de saída, que será a camada que resultará os dados depois de processado. As redes com múltiplas camadas, a saída da primeira camada se tornará a entrada da próxima, com isso tornando a rede um processo de três camadas, ficando mais simples o entendimento de cada processo.

2.1.2.1 Rede Neural de Camadas Única (Single Layer Neural Network)

Esse tipo de rede possui uma única camada oculta. Uma rede neural recebe uma entrada de vetor de p valores $x = (x_1, x_2, \ldots, x_n)$ e constrói uma função $f(x)$ para predizer a resposta de y. A figura da rede abaixo mostra uma simples rede feed-forward neural network.

As flechas indicam que cada camada alimenta a outra que está sendo direcionada, no caso, x1 alimenta a1 que alimenta f(x), seguindo o mesmo para todas as outras conexões.

Figura 2 - funcionamento de uma rede feed-forward neural network



Fonte: Adaptado de JAMES, 2021.

O modelo de rede neural tem a seguinte forma matemática:

$$f(x) = \beta_0 + \sum_{k=1}^{k} \beta_k \, h_k(x)$$

$$a_k = h_k(x) = \left(w_{k0} + \sum_{j=1}^{p} wkj \; x_j \right)$$

$$E(x) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Onde:

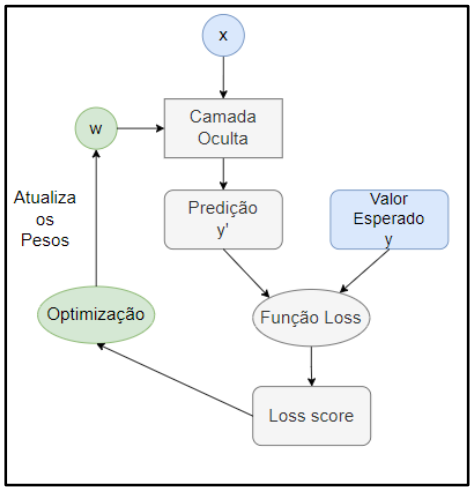
$h_k(x) = a_k \rightarrow$ função de ativação

A função de ativação é uma função não linear, permitindo análises de tarefas mais complexas, tornando-a capaz de aprender e executar essas tarefas complicadas, como tradução de idiomas e classificação de imagens, ela basicamente decide se o neurônio deve ser ativado ou não (CHOLLET, 2021).

$E(x) \rightarrow$ função loss

O trabalho da função loss (também chamada de função erro ou função de custo) é medir o quão longe a saída predita está da saída esperada (CHOLLET, 2021). O trabalho fundamental do aprendizado profundo é usar o valor da função loss e ir ajustando o valor dos pesos e baías utilizando o regime chamado backpropagation (CHOLLET, 2021), conforme a figura:

Figura 3 - Regime backpropagation



Fonte: Adaptado de CHOLLET, 2021.

O processo de treinamento de uma rede neural é minimizar o erro usando a descida do gradiente, esse processo reduz o valor da função de erro até convergir para um valor geralmente um mínimo local (brilliant.org), o treino mais otimizado é o mínimo global.

Figura mínimo local e global

A descida do gradiente atualiza os valores dos pesos e baías com a seguinte equação:

$$w_{i+1}^{\rightarrow} = w_i^{\rightarrow} - \alpha \frac{\partial E(x)}{\partial w_i^{\rightarrow}}$$

$$b_{i+1}^{\rightarrow} = b_i^{\rightarrow} - \alpha \frac{\partial E(x)}{\partial b_i^{\rightarrow}}$$

Onde:

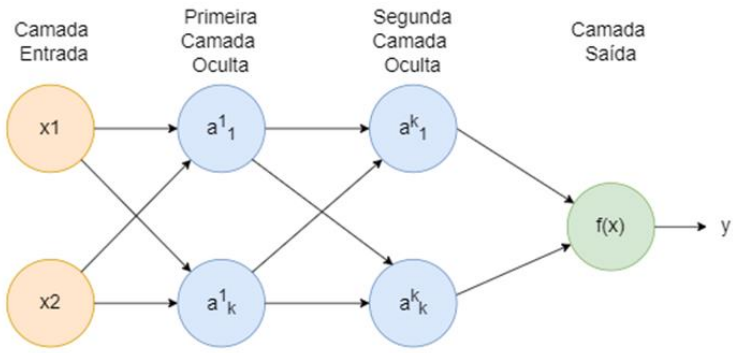
$\alpha \rightarrow$ é a taxa de aprendizado (learning rate)

Esta taxa é a taxa de velocidade de descida do gradiente, ou seja, é o tamanho de cada etapa na otimização do aprendizado, essa taxa é importante, por causa de suas consequências, podendo ficar preso em mínimos locais, não tendo uma boa performance, como podendo também não conseguir aprender nada.

2.1.2.2 Redes Neurais de Múltiplas Camadas (Multilayer Neural Networks)

Esse tipo de rede possui mais de uma camada oculta. Conforme pode-se notar na imagem abaixo:

Figura 4 - Backpropagation



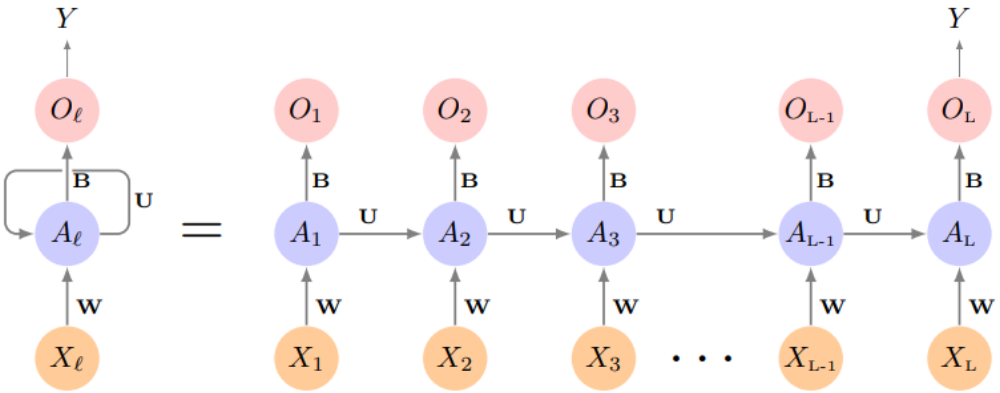
Fonte: Adaptado de JAMES, 2021.

As flechas indicam que cada camada alimenta a outra que está sendo direcionada, no caso, x1 alimenta a1, o resultado de a1 será a entrada de dados de ak1, e o resultado de ak1 alimentará f(x), seguindo o mesmo para todas as outras conexões. Observa que as flechas que determinam a direção da propagação dos dados, outro sentido de direção para a entrada de x1 é alimentar a12, o resultado se tornará a entrada para ak1 ou akk e o resultado alimentar a f(x), a múltipla camada se refere a ter mais de uma camada oculta, não a quantidade de neurônios, que no caso do exemplo abaixo, cada camada possui dois neurônios.

2.1.2.3 Redes Neurais Recorrentes (Recurrent Neural Networks)

Neste tipo de rede, os dados de entradas são recorrentes, ou seja, quando dados possuem sequências de informações, seja uma sequência de letras para formar uma frase, ou uma sequência de valores de uma ação no mercado financeiro conforme o tempo. A rede possui compartilhamentos de pesos e da camada anterior, produzindo um recurso de memorização de etapas.

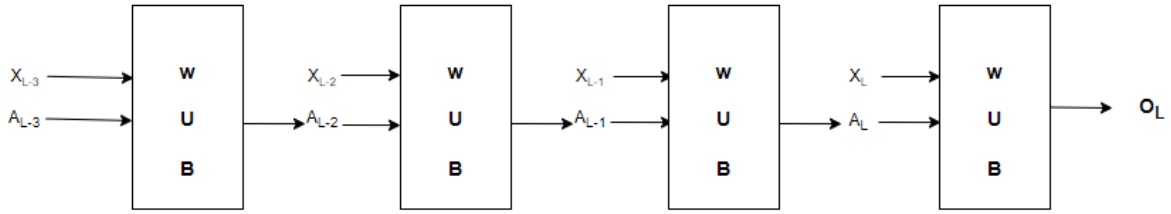
Figura 5 – Simples Recurrent Neural Network



Fonte: JAMES et. al., 2021.

A camada oculta A_L recebe o valor de entrada X_L e também recebe o compartilhamento da camada anterior que é chamado de vetor de ativação A_{L-1} dá sequência anterior para produzir o vetor de ativação atual, os pesos W, U e B, também são usados em cada processo sequencial, todas as etapas tem uma saída O_L, mas apenas a última que é relevante, consequentemente sendo a última a atualização dos pesos.

Figura 6 – Simples Recurrent Neural Network



Fonte: Adaptado de CHOLLET, 2021.

$$A_L = f(X_{L-1}, A_{L-1}, W, U, B) = f(X_{L-1}W + A_{L-1}U + B)$$

2.2 Previsão de Séries Temporais

Séries Temporais são um conjunto de observações sobre uma variável registrados em períodos regulares (REIS, 2007). Seu intuito é explorar o comportamento passado e também de prever o comportamento futuro em um determinado problema (SPADINI, 2021).

Identificar padrões não aleatórios na série temporal de uma variável de interesse, e a observação deste comportamento passado nos permite fazer previsões que colaborem com a tomada de decisões. Podemos enumerar os seguintes exemplos de séries temporais: temperaturas máximas e mínimas diárias em uma cidade, vendas mensais de uma empresa, valores de fechamento diários do Ibovespa, resultado de um eletroencefalograma, entre outros (REIS, 2007).

A teoria básica que norteia a análise de séries temporais é que há um sistema causal mais ou menos constante, relacionado com o tempo, que exerceu influência sobre os dados no passado e pode continuar a fazê-lo no futuro (SOUZA, 2020).

2.2.1 Autocorrelação de uma série temporal

Uma característica muito importante das séries temporais é o fato de os dados no tempo futuro apresentarem correlação com os dados do tempo passado. Ou seja, de alguma forma, o que acontece no futuro depende ou está correlacionado ao que

aconteceu no passado. Portanto, existe uma medida matematicamente bem definida para determinar se os dados apresentam este tipo de comportamento ao longo do tempo. Ela se chama medida de autocorrelação (SOUZA, 2020).

A autocorrelação de uma variável temporal, xt, onde t é o índice temporal dos dados, é dada pela seguinte fórmula:

$$\frac{1}{N}\sum_{t=0}^{N} x_t x_{t-1}$$

3 METODOLOGIA

Para o desenvolvimento deste trabalho foi feita inicialmente uma revisão de literatura pertinente sobre as tecnologias e ferramentas de desenvolvimento que poderiam ser empregados em sua execução. Também foi necessário buscar junto à Literatura os conhecimentos gerais e específicos da matéria a ser analisada para construir o conhecimento necessário.

3.1 Dados e ferramentas de predição

O dataset usado para o estudo de caso deste trabalho foi obtido na New York Stock Exchange, uma das principais bolsas de valores mundiais localizada em Mana, cidade de Nova York. Estes dados foram utilizados para comparar a performance preditiva de várias arquiteturas de aprendizagem profunda, bem como outros modelos de aprendizado de máquina (lineares e não-lineares). No caso da aprendizagem profunda, foi utilizado a API Keras que opera a customização e treinamento dos modelos com o pacote Tensorflow.

Os dados utilizados nesta monografia são compostos de 3 séries temporais, ao longo de um período que vai de 3 de dezembro de 1962 a 31 de dezembro de 1986. Estes dados foram colhidos diariamente, excetuando finais de semana e feriados (LEBARON E WEIGEND, 1998).

Cada série temporal é criada através do cálculo de uma variável que apresenta um significado bem definido dentro dos conceitos de econometria.

A primeira série temporal é referente ao volume de negócios na bolsa NYSE, essa é a fração de ações negociadas no dia, esse valor é trabalhado para deixar com aspecto de séries estacionária, o volume atual é dividido pela média móvel de 100 dias passados, e desse resultado, é retirado o logaritmo, resultando no logaritmo de volume. Esta é a variável a ser predita e foram criados modelos para a predição de vt baseados em dados de k dias anteriores (LEBARON E WEIGEND, 1998).

A segunda série temporal é o retorno, esse valor é definido a partir do valor real dos preços de um ativo Xt, nesse caso foi utilizado as primeiras diferenças do logaritmo do nível do índice Dow Jones Industrials como uma medida de retornos relativos das ações (LEBARON E WEIGEND, 1998).

$$r_t = \frac{X_t - X_{t-1}}{X_{t-1}}$$

A terceira série temporal é a log da volatilidade, segundo o artigo, é calculada como $log\ log\ (\sigma_t^2)$, onde: $\sigma_t^2 = \beta\sigma_{t-1}^2 + (1-\beta)r_t^2 \ com\ \beta = 0.9$. Essa expressão é um filtro exponencial do retorno r_t ao quadrado, a σ_t^2 é a variância incondicional da série temporal.

A máquina utilizada para a predição é a plataforma aberta Google Colab ou “Colaboratory”, essa plataforma é aberta e disponibiliza um campo de trabalho para qualquer pessoa escrever e executar código Python pelo navegador, essa escolha foi referente a disponibilidade de uma ferramenta on-line com acesso a GPU e sem custo financeiro, determinando uma comparação justa a todas as arquiteturas utilizadas.

A condução do trabalho seguiu parâmetros utilizados por Gareth James e outros autores citados, utilizando os dados colhidos para comparar a performance preditiva de várias arquiteturas de aprendizagem profunda, e também na utilização de tratamento de dados do dataset. As arquiteturas de rede neurais utilizadas foram: a Linear Regression modelo da Scikit-learn; o modelo de camadas Dense do

TensorFlow/Keras; e o modelo recorrente SimpleRNN, também do Tensor Flow/Keras.

4 RESULTADOS E DISCUSSÃO

Para obtenção dos resultados, os dados foram separados em dois grupos: grupo de treinamento (df_train) e de teste (df_test). O dataframe (df) possui uma coluna chamada "train", nessa coluna possui os dados *True* ou *False*, sendo *True* os dados para treinamento e *False* os dados para teste, conforme exemplificado com a figura 7.

Figura 7 - Dataframe New York Stock Change

	date	day_of_week	DJ_return	log_volume	log_volatility	train
1	1962-12-03	mon	-0.004461	0.032573	-13.127403	True
2	1962-12-04	tues	0.007813	0.346202	-11.749305	True
3	1962-12-05	wed	0.003845	0.525306	-11.665609	True
4	1962-12-06	thur	-0.003462	0.210182	-11.626772	True
5	1962-12-07	fri	0.000568	0.044187	-11.728130	True
...	...	...	...	...	...	...
6047	1986-12-24	wed	0.006514	-0.236104	-9.807366	False
6048	1986-12-26	fri	0.001825	-1.322425	-9.906025	False
6049	1986-12-29	mon	-0.009515	-0.371237	-9.827660	False
6050	1986-12-30	tues	-0.001837	-0.385638	-9.926091	False
6051	1986-12-31	wed	-0.006655	-0.264986	-9.935527	False

6051 rows × 6 columns

Fonte: Elaborado pelo autor.

Como o conjunto de dados utilizado para esse trabalho é histórico e são séries temporais, os dados de treinamento e teste terão uma quantidade de dados de entrada e essa quantidade será determinada com a variável Lag(atraso), como pode ser entendido com a figura 7. Os dados foram separados com três dias de atrasos (Lag), sendo Vt-3, Vt-2 e Vt-1 para os dados de 'log_volume', Rt-3, Rt-2 e Rt-1 para os dados

de ‘DJ-return’, Zt-3, Zt-2 e Zt-1 para os dados de ‘log_volatility’, e o ‘Target (Vt)’ ficou sendo a variável a ser predita. Essa figura foi apenas uma demonstração de como será a lógica da separação dos dados sequenciais, para a predição dessa série temporal.

Figura 8 - Dataframe New York Stock Change Adaptado Para Predição

Vt-3	Vt-2	Vt-1	Rt-3	Rt-2	Rt-1	Zt-3	Zt-2	Zt-1	Target (Vt)
0.525306	0.346202	0.032573	0.003845	0.007813	-0.004461	-11.665609	-11.749305	-13.127403	0.210182
0.210182	0.525306	0.346202	-0.003462	0.003845	0.007813	-11.626772	-11.665609	-11.749305	0.044187
0.044187	0.210182	0.525306	0.000568	-0.003462	0.003845	-11.728130	-11.626772	-11.665609	0.133246
0.133246	0.044187	0.210182	-0.010824	0.000568	-0.003462	-10.872526	-11.728130	-11.626772	-0.011528
-0.011528	0.133246	0.044187	0.000124	-0.010824	0.000568	-10.977797	-10.872526	-11.728130	0.001607
...	...	...	...	...	...	...	...	...	...
-0.001507	0.108579	0.144112	-0.005267	0.005267	0.000310	-10.280234	-10.259191	-10.236318	-0.631830
-0.631830	-0.001507	0.108579	0.000298	-0.005267	0.005267	-10.385307	-10.280234	-10.259191	-0.365829
-0.365829	-0.631830	-0.001507	-0.001216	0.000298	-0.005267	-10.485360	-10.385307	-10.280234	-0.137014
-0.137014	-0.365829	-0.631830	0.002336	-0.001216	0.000298	-10.569258	-10.485360	-10.385307	-0.041932
-0.041932	-0.137014	-0.365829	-0.001418	0.002336	-0.001216	-10.665966	-10.569258	-10.485360	-0.125945

Fonte: Elaborado pelo autor.

As arquiteturas de rede neurais utilizadas foram a Linear Regression modelo da Scikit-learn, o modelo de camadas Dense do TensorFlow/Keras, e o modelo recorrente SimpleRNN também do Tensor Flow/Keras.

Foi utilizado todos os passos padrão para o treinamento do LinnearRegression do Scikit-learn. A figura 9 mostra o processo de treinamento e teste: primeiro é carregado o conjunto de dados chamando o método Update Beta; segundo é a separação dos dados de entrada e saída para predição, terceiro é carregado o modelo LinnearRegression; quarto os dados são enviados para treinamento; e por último é efetuado a predição com os valores de teste, printando o valor retornado.

Figura 9 – Treinamento LinearRegression

```
df = UpdateBeta(0.85)

x_train, y_train, x_test, y_test = nysedf(5)

modelo = LinearRegression()
modelo.fit(x_train, y_train)
print(modelo.score(x_test,y_test))
```

Fonte: Elaborado pelo autor.

A próxima rede neural é a *Sequencial* do *TensorFlow/Keras*, este modelo permite a configuração de camadas em série, como na explicação referente a múltiplas camadas *(Multilayer Neural Networks*), e outras arquiteturas, a camada *Dense é* uma camada que irá receber os dados de entrada, esses dados serão multiplicados por pesos e ativado por uma função de ativação, a saída dessa camada irá ser a entrada da próxima. No código em questão a primeira camada contém 12 neurônios e a função de ativação configurada é a tangente hiperbólica; a próxima é a camada de saída e tem apenas 1 neurônio com uma função de ativação padrão, isto é, a função tangente hiperbólica é a função padrão, no caso a demonstração da camada anterior é apenas pra mostrar como configurar uma função aleatória, conforme figura 10.

Figura 10 – Configuração da Arquitetura Dense Não Linear

```
def Sequential():
    modelo = tf.keras.Sequential([
    tf.keras.layers.Dense(12, activation='tanh'),
    tf.keras.layers.Dense(1)
    ])

    modelo.compile(loss='mean_squared_error', optimizer='adam',
                   metrics=[tf.keras.metrics.MeanSquaredError()])

    return modelo
```

Fonte: Elaborado pelo autor.

A última arquitetura é a Redes Neurais Recorrentes (Recurrent Neural Networks), no código a seguir, representado pela figura 11, a camada SimpleRNN é a camada recorrente com 12 neurônios, um *corpo* para os dados de entrada determinado pela variável shape, e a função de ativação tangente hiperbólica, a próxima camada é a de saída.

Figura 11 – Configuração da Arquitetura Recorrente RNN

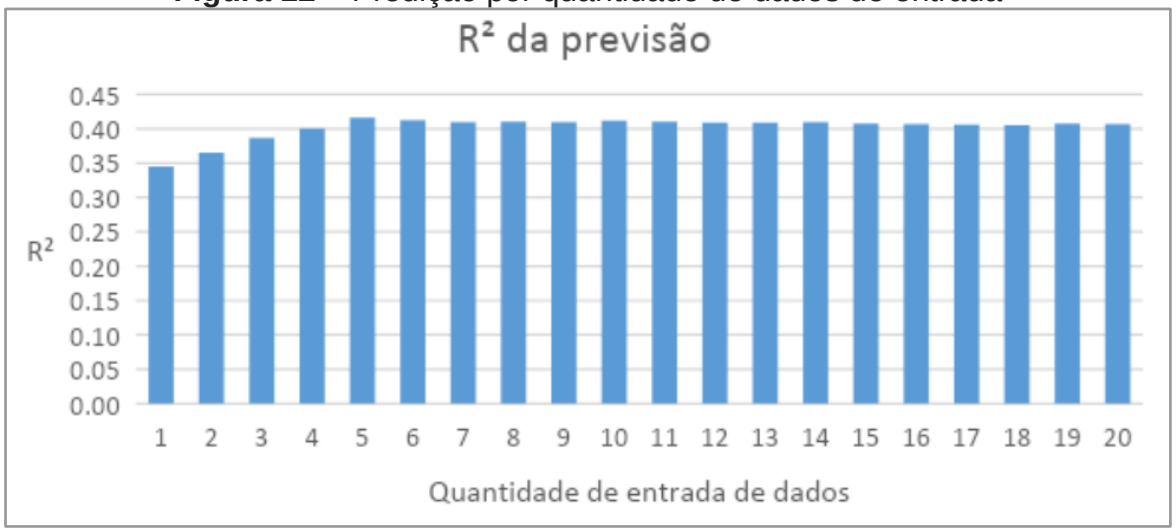
```
def simpleRNN(shape):
    modelo = tf.keras.Sequential([
        tf.keras.layers.SimpleRNN(12, input_shape=shape, activation='tanh'),
        tf.keras.layers.Dense(1)
    ])

    modelo.compile(loss='mean_squared_error', optimizer='adam',
                   metrics=[tf.keras.metrics.MeanSquaredError()])
    return modelo
```

Fonte: Elaborado pelo autor.

Com os dados e arquiteturas definidas para a predição, o primeiro processo desse trabalha foi determinar a quantidade necessária de itens de entrada para melhor performance de predição. No caso da figura 12, a melhor predição constatada foi de 5 dias, no qual teve o valor de R2 aproximadamente 0,415, sendo que, a quantidade de dados de entrada maiores do que 5 dias terão aproximação o mesmo valor, sendo muito pouco abaixo desse melhor valor constatado, com isso, a utilização padrão para o trabalho foi 5 dias para todos os testes a serem comparado.

Figura 12 – Predição por quantidade de dados de entrada

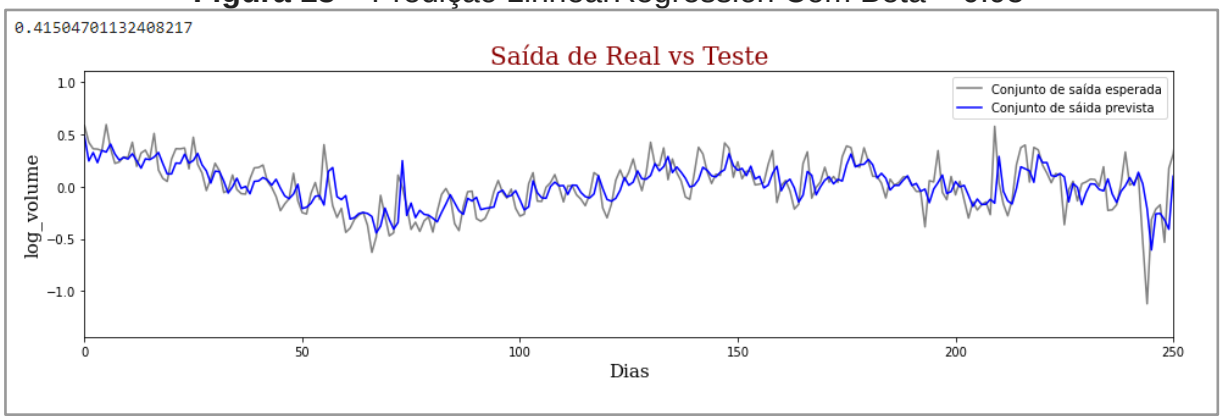


Fonte: Elaborado pelo autor.

Para os dados com a volatilidade padrão do próprio Data Frame, ou seja, com o valor padrão de Beta β 0.90, as arquiteturas LinnearRegression e a Dense ficaram praticamente empatados. Atualizando os dados de volatilidade com os valores de Beta em: 0.85, 0.90 ou 0.95, não foi obtido praticamente nenhuma vantagem, os valores de predição ficaram praticamente o mesmo. Foi feito testes com valor menor do que 0.85, e o resultado foi pior, por esse motivo esse teste não foi reportado pra comparação, a

figura 13 mostra o gráfico de comparação de valores esperado e predição com o $\beta = 0.95$.

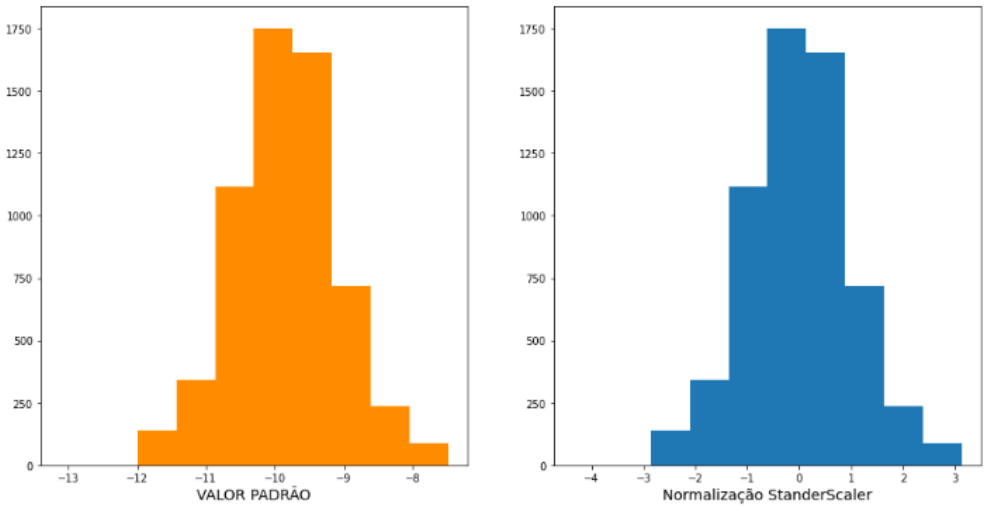
Figura 13 – Predição LinnearRegression Com Beta = 0.95



Fonte: Elaborado pelo autor.

Como não foi obtido diferença na predição com arquiteturas de rede neurais diferente, outro método que é muito utilizado na predição é a normalização dos dados, foi utilizado dois processos de normalização, a StanderScaler e MinMaxScaler, ambas utilizando a biblioteca Scikit-learn.

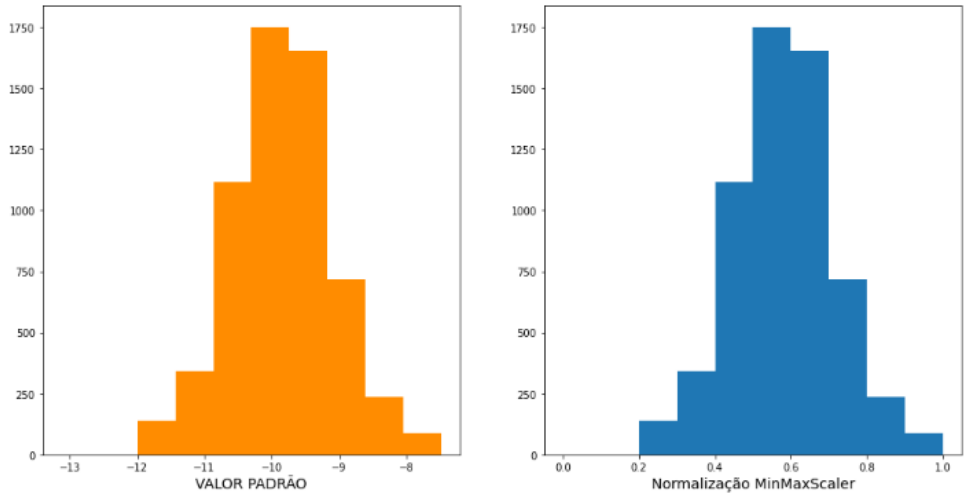
Figura 14 – Normalização StanderScaler



Fonte: Elaborado pelo autor.

O processo de normalização foi efetuado com a dados da variável log_volativity. A figura 14, mostra a normalização dos dados como padronização de dimensionamento de recurso, e a figura 15. mostra a padronização min-máx.

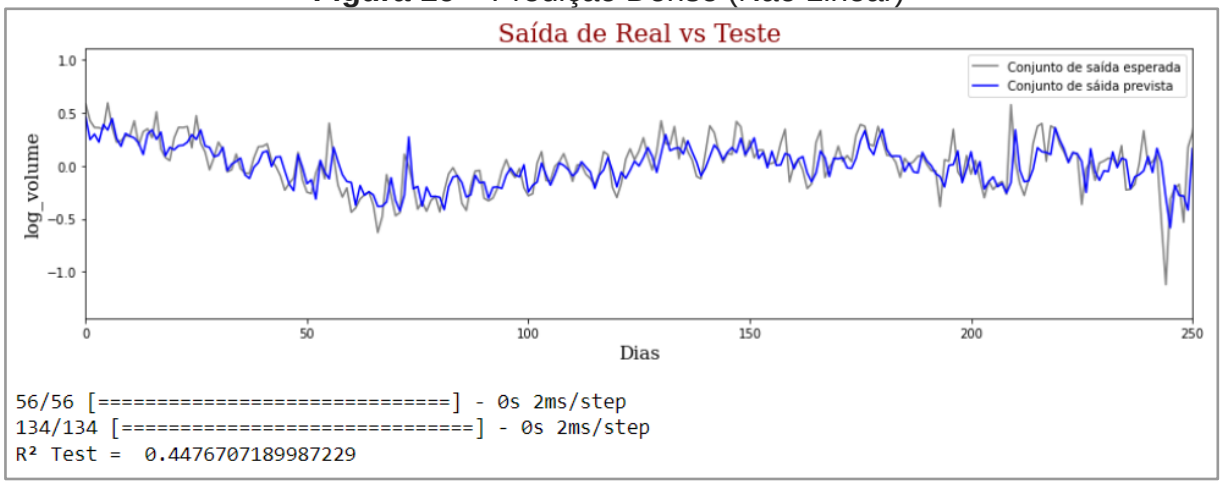
Figura 15 – Normalização MinMaxScaler



Fonte: Elaborado pelo autor.

Após efetuado esses ajustes nos dados, foi efetuado os treinamentos com todas as arquiteturas, a melhor predição foi com a Dense, já a LinnearRegression não melhorou a predição.

Figura 16 – Predição Dense (Não Linear)



Fonte: Elaborado pelo autor.

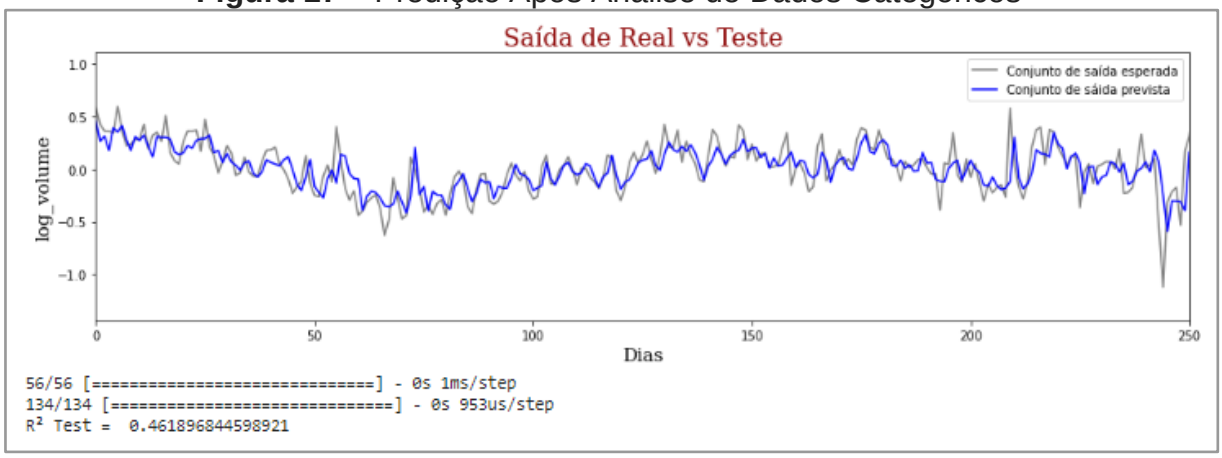
Um dos processos mais utilizados pelos estatísticos é a análise de dados categóricos, esse processo permite disponibilizar valores para balancear tipos de dados para predição, segundo o (JAMES, 2021 pg 431) “Todos os modelos podem ser melhorados com a inclusão da variável *day-of-week* (dias da semana) correspondente ao dia t do target V_t [...]”. Para realizar essa utilização de dados categóricos, foram efetuados três processos de tratamento de dados, sendo:

Substituição dos valores de texto para número, exemplo: 1 para segunda, 2 para terça, 3 para quarta, 4 para quinta e 5 para sexta;

Criação de colunas para cada dia da semana e representando o dia com o valor 1 para a coluna que representa o dia e zero para as demais;

Criação de duas colunas, seno e cosseno, e aplicando valor para cada dia, exemplo: 360º / 5 = 72, será 72 graus para cada dia, 72 para segunda, 144 para terça, ..., 360 para sexta.

Figura 17 – Predição Após Análise de Dados Categóricos



Fonte: Elaborado pelo autor.

Após os testes realizados, a predição teve uma correção significante referente a arquitetura LinnearRegression, que saltou de 0,41 para 0,46, sendo esse o mesmo resultado com a arquitetura Dense, a tabela 1 mostra os valores obtido por todos os testes, por todos os processos de modificação de dados efetuado nos dados.

Tabela 1 – Valores obtidos por todos os testes.

Tipos de Redes Neurais	Scikit-Learn LinnearRegression	TensorFlow Dense Não Linear	TensorFlow SimpleRNN Recorrente
Beta 0.85	$R^2 = 0,4120$	$R^2 = 0,4106$	$R^2 = 0,3700$
Beta 0.90	$R^2 = 0,4135$	$R^2 = 0,4149$	$R^2 = 0,3585$
Beta 0.95	$R^2 = 0,4150$	$R^2 = 0,3994$	$R^2 = 0,3753$
Standardization	$R^2 = 0,4142$	$R^2 = 0,4452$	$R^2 = 0,4238$
MinMaxScaler	$R^2 = 0,4142$	$R^2 = 0,4424$	$R^2 = 0,4395$
Dias da Semana	$R^2 = 0,4142$	$R^2 = 0,4420$	$R^2 = 0,3817$
Dias get-dummies	$R^2 = 0,4590$	$R^2 = 0,4567$	$R^2 = 0,4390$
Dias Sem-Cos	$R^2 = 0,4513$	$R^2 = 0,4618$	$R^2 = 0,4233$

Fonte: Elaborado pelo autor.

5 CONSIDERAÇÕES FINAIS

A proposta desse trabalho foi aplicar três tipos de arquiteturas neurais para predição de dados futuros, sendo a rede linear, LinnearRegression, da biblioteca do Scikit-Learn, a rede de camadas Dense, construída através de camadas sequenciais, utilizando a biblioteca Keras do TensorFlow, e rede de camada recorrente obtida também em camadas sequencias com a camada recorrente SimpleRNN também da biblioteca Keras do TensorFlow, o valor a ser predito foi do próximo dia do volume do mercado financeiro obtido pela bolsa de valores da cidade de New York Stock Exchange.

Aplicou-se então os treinamentos com as arquiteturas descritas, e o resultado obtido com os dados padrão, ou seja, os dados obtidos sem ajustes, não resultaram em diferenças significantes de predição, apesar que o resultado da rede recorrente foi inferior, mas as duas outras redes obtiveram resultados praticamente iguais, constatando que para esse tipo de predição, uma simples rede linear disponibiliza um bom resultado, sem a necessidade de aplicar redes complicadas de serem explicadas,

sendo que o treinamento é praticamente instantâneo, enquanto a rede Dense teve que ter alguns minutos para obter um bom resultado.

Outra técnica realizada neste trabalho foi a normalização e padronização dos dados, no caso, transformar os dados entre a média no ponto zero, ou os dados ficarem entre zero e um, conforme figura 14 e 15, nesse caso foi possível ver uma melhora nos resultados obtidos pela rede recorrente e da rede de camada Dense, um dos pontos dessa valorização é a diminuição do valor do dado gravado na memória, consequentemente, “facilitando” o cálculo a ser realizado pelo processamento de dados, mostrando um possível técnica para melhorar a predição.

Verificando os efeitos sazonais, uma outra técnica utilizada foi a técnica de valorizar dados categóricos, com essa adição de dados categóricos foram obtidos melhores resultados, saltando cerca de cinco porcento o coeficiente de determinação obtido pela rede Linear, nesse caso, esses dados categóricos disponibilizam a possibilidade de aplicar um certo valor de peso para cada dia, referenciando a valorização desse período, consequentemente, atingindo pontos de sazonalidade que ocorre entres os períodos semanais.

Com esses dados obtidos, que podem ser analisados conforme descrito na tabela da tabela 1, sugere-se considerar a manutenibilidade dos dados, tratando com técnicas de normalização, padronização e categorização de dados, o resultado mostrou que uma simples regressão linear prediz melhor resultado do que uma rede recorrente, pelo menos pelos testes que foram feitos por esse trabalho e por ser na predição de dados seriais de mercado financeiro.

REFERÊNCIAS

CHOLLET, François. **Deep Learning with Python**. 2. ed. Shelter Island, NY 11964: Manning Publications Co., 2021. 814 p.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. **An Introduction to Statistical Learning: With Application in R.** 2. ed., 2021. Disponível em: https://hastie.su.domains/ISLR2/ISLRv2_website.pdf. Acesso em: 12 out. 2022.

IGNACIO, L. F. F. **Aprendizado de máquina: da teoria à aplicação.** Universidade Federal Fluminense, Rio de Janeiro, 2021. Disponível em: https://app.uff.br/riuff/bitstream/handle/1/22872/Lucas_Fran%C3%A7a.pdf?sequence=1. Acesso em 20 set. 2022.

IZBICKI, R.; SANTOS, T. M. **Aprendizado de máquina: uma abordagem estatística.** UFSCAR, São Carlos, 2020. Disponível em: http://www.rizbicki.ufscar.br/AME.pdf. Acesso em: 20 set. 2022.

MESQUITA, C. M. H. S. R. **Ciência de dados e aprendizado de máquina para predição em séries temporais financeiras**. Universidade Federal de Minas Gerais, Belo Horizonte, 2019. Disponível em: https://repositorio.ufmg.br/bitstream/1843/30444/1/CaioMarioHenriquesSilvaRochaMesquita.pdf. Acesso em: 19 set. 2022.

SPADINI, A. S. **Séries temporais e suas aplicações**. Disponível em: https://www.alura.com.br/artigos/series-temporais-e-suas-aplicacoes. Acesso em: 21 out. 2022.

SOUZA, C. E. **Séries temporais com Machine Learning**. Disponível em: https://medium.com/data-hackers/s%C3%A9ries-temporais-com-machine-learning-parte-1-e8fa62b82d48. Acesso em: 20 out. 2022.

SOUZA, L. A. M. **Aplicação de Aprendizado de Máquina para Predição de Prioridade em Gestão de Incidentes.** Rio de Janeiro, 2017. Disponível em: http://professorluizalberto.com.br/site/images/2020-1/TCC%20Aplica%C3%A7%C3%A3o%20de%20Aprendizado%20de%20M%C3%A1quina%20para%20Predi%C3%A7%C3%A3o%20de%20Prioridade%20em%20Gest%C3%A3o%20de%20Incidentes.pdf . Acesso em: 19 set. 2022.

REIS, M. M. **Análise de Séries Temporais.** UFSC, 2007. Disponível em: https://www.inf.ufsc.br/~marcelo.menezes.reis/Cap4.pdf. Acesso em: 22 out. 2022.

TEIXEIRA, J. F. **O que é Inteligência Artificial,** 1990. Disponível em: https://repositorio.ufsc.br/bitstream/handle/praxis/395/o%252%200que%20e%20inteligencia%20artificial.pdf?sequence=1. Acesso em: 19 set. 2022.