# UNIVERSIDADE FEEVALE

# CIÊNCIA DA COMPUTAÇÃO

APRENDIZADO PROFUNDO E INTELIGÊNCIA ARTIFICIAL APLICADO NA ANÁLISE DO MERCADO FINANCEIRO

# Paulo Ricardo Prass

## Novo Hamburgo - RS

## 2022

**UNIVERSIDADE FEEVALE**

Aprendizado profundo e inteligência artificial aplicado na análise do mercado financeiro

**Trabalho de Conclusão de Curso apresentado na Universidade Feevale como requisito básico para a conclusão do Curso de Ciência da Computação.**

**Orientador (a): Gabriel da Silva Simões**

**Aluno (a): Paulo Ricardo Prass**

## Novo Hamburgo - RS

## 2022

# AGRADECIMENTOS

A minha mãe, Angela e ao meu padrasto Leonardo, como também ao meu pai João e aos meus irmãos Bruno e Juliana, pelo amor, conselhos, ajudas e incentivos que sempre me ofereceram.

Ao meu orientador Prof. Dr. Gabriel da Silva Simões, pelo apoio, paciência, pelo auxílio, confiança e ensinamentos.

A minha namorada Pâmela, pelo amor, companheirismo e incentivo.

A todos meus amigos, que sempre estiveram comigo.

A todos os professores e professoras que um dia me ensinaram e contribuíram para minha formação e conhecimento.

Ao Programa de Graduação da Universidade Feevale, por possibilitar o desenvolvimento e pesquisa deste trabalho.

# RESUMO

Este trabalho de TCC descreve algumas das técnicas e combinações de algoritmos de inteligência artificial e aprendizado profundo aplicadas em dados do mercado financeiro, apresentando que atualmente se percebe que há dificuldades de se prever para onde vão os preços de um determinado ativo financeiro. Através de estudos e pesquisas utilizando redes neurais, foram realizados treinamentos combinando algoritmos e análise de dados e com os resultados obtidos foi possível observar o valor preditivo graficamente. Até o final deste trabalho de TCC compreenderemos mais a respeito destes conceitos e modelos de redes neurais.

# ABSTRACT

This conclusion work describes some of the techniques and combinations of artificial intelligence and deep learning algorithms applied to financial market data, as it is currently perceived that there are difficulties in predicting where the prices of a given financial asset will go. Through studies and research using neural networks, trainings were carried out combining algorithms and data analysis and with the results obtained it was possible to observe the predictive value graphically. By the end of this conclusion work we will understand more about these concepts and models of neural networks.

# LISTA DE ABREVIATURAS E SIGLAS

ANN - Artificial neural network

API - *Application programming interface*

BSE - *Bombay Stock Exchange*

CNN - *Convolutional neural network*

CSV - *Comma-separated values*

DL - *Deep learning*

DRNN - *Deep recurrent neural network*

GRU *- Gated Recurrent Unit*

LOB - *Limit order books*

LSTM - *Long short-term memory*

MBO - *Market by order*

ML - *Machine learning*

MSE - *Mean Squared Error*

NIFTY - *National Stock Exchange Fifty*

NYSE - *New York Stock Exchange*

NSE - *National Stock Exchange*

RNN - *Recurrent neural network*

RMSE - *Root Mean Squared Error*

# LISTA DE FIGURAS

Figura 1 - Classificação da pesquisa (adaptado de BEZ, 2011).

Figura 2 - Arquitetura RNA.

Figura 3 - Diagrama de aprendizado supervisionado.

Figura 4 - Rede Neural para Rede Neural Recorrente.

Figura 5 - Arquitetura LSTM (traduzido de Deep Learning Book).

Figura 6 - Função Tahn versus Sigmoid (S. Sharma *et* al., 2014).

Figura 7 - Arquitetura GRU (traduzido de Deep Learning Book).

Figura 8 - Arquitetura GRU versus Arquitetura LSTM (traduzido de Deep Learning Book).

Figura 9 - Comparativo entre Rede Neural completa com a aplicação de Rede Neural com Dropout (Srivastava *et* al., 2014).

Figura 10 - Overfitting e Underfitting (Didática Tech *et* al., 2022).

Figura 11 - Regressão linear (A. Gupta *et* al., 2022).

Figura 12 - Histórico de preço PETR4.

Figura 13 - Histórico de preço MGLU3.

Figura 14 - Histórico de preço ABEV3.

Figura 15 - Resultado obtido modelo LSTM.

# LISTA DE TABELAS

Tabela 1 - CNN univariado com dados de uma semana anterior.

Tabela 2 - CNN univariado com dados de duas semanas anteriores.

Tabela 3 - LSTM univariado com dados de duas semanas anteriores.

Tabela 4 - LSTM com CNN univariado com dados de duas semanas anteriores.

Tabela 5 - LSTM convolucional univariado com dados de duas semanas anteriores.

Tabela 6 - Valores de RMSE resultantes.

Tabela 7 - Valores de RMSE resultantes do LSTM-1.

Tabela 8 - Valores de RMSE resultantes do LSTM-2.

Tabela 9 - Valores de RMSE resultantes do GRU-1.

Tabela 10 - Valores de RMSE resultantes do GRU-2.

Tabela 11 - Valores de RMSE resultantes do LSTM-1 sem dropout.

Tabela 12 - Valores de RMSE resultantes do LSTM-2 sem dropout.

Tabela 13 - Valores de RMSE resultantes do GRU-1 sem dropout.

Tabela 14 - Valores de RMSE resultantes do GRU-2 sem dropout.

**LISTA DE FÓRMULAS**

Fórmula 1 - Fórmula do erro quadrático médio.

Fórmula 2 - Fórmula da raiz quadrática do erro médio

## SUMÁRIO

**1. INTRODUÇÃO** ...........................................................................................11  
  
**2. PROPOSTA DO TRABALHO** ....................................................................14  
  
**3. BASE DE DADOS** ......................................................................................17

**4. METODOLOGIA DA PESQUISA** ...............................................................20

**5. REFERENCIAL TEÓRICO** .........................................................................22

5.1 REDES NEURAIS ARTIFICIAIS .......................................................22

5.2 PROCESSO DE APRENDIZAGEM ..................................................25  
 5.3 RECURRENT NEURAL NETWORK (RNN) ......................................27

5.4 LONG SHORT-TERM MEMORY (LSTM) ........................................29

5.5 GATED RECURRENT UNIT (GRU) ..................................................31

5.6 LSTM VERSUS GRU ........................................................................32

5.7 DROPOUT .........................................................................................35

5.8 MEAN SQUARED ERROR (MSE) ....................................................37

5.9 ROOT MEAN SQUARED ERROR (RMSE) ......................................39

**6. TRABALHOS CORRELATOS** …................................................................41

6.1 APRENDIZADO PROFUNDO PARA PREVISÃO DE AÇÕES...........41

6.2 REDE NEURAL PARA PREVISÃO DE SÉRIES TEMPORAIS..........41  
 6.3 PREVISÃO DE PREÇOS DE AÇÕES, LSTM E CNN........................42

6.4 APRENDIZADO PARA DADOS DO MERCADO POR PEDIDO........43

6.5 PREVISÃO DE PREÇO DE AÇÕES USANDO CNN E LSTM...........43

**7. AMBIENTE DE DESENVOLVIMENTO E EXPERIMENTAÇÃO** ................48

**8. EXPERIMENTOS** .......................................................................................53

# 9. CONCLUSÃO ............................................................................................66

9.1 TRABALHOS FUTUROS ..................................................................67

**10. REFERÊNCIAS** ........................................................................................68

# INTRODUÇÃO

Investir é o melhor caminho para as pessoas aumentarem as suas chances de serem bem-sucedidas em suas finanças. Por meio dos investimentos, é possível aumentar as oportunidades de crescimento de capital tanto da pessoa física como jurídica. Porém, para o cidadão comum interessado em investir no mercado financeiro, a falta de conhecimento na área pode ser uma barreira significativa frente à complexidade do ramo. Esta falta de conhecimento, muitas vezes está em não saber se o preço de um detrerminado ativo vai subir ou descer, isso pode ser uma barreira para grande parte das pessoas que iniciam neste tipo de investimento, mesmo com o estudo de muitos livros, cursos e práticas.

Mas o mercado financeiro é extremamente complexo, frequentemente é preciso reagir rápido ao preço. Devido a isso é necessário estar muito convicto ao realizar uma operação de compra ou venda de determinado ativo.

Visando estas dificuldades, este trabalho propõe-se a realizar um estudo com algoritmos de inteligência artificial e aprendizado profundo, como também testar diversos modelos de redes neurais, com o intuito de avaliar seus resultados preditivos no movimento do preço do mercado financeiro.

A inteligência artificial se refere a programação de software que possui a capacidade de aprender por conta própria, ou seja, é um algoritmo capaz de se aperfeiçoar por conta própria.

A inteligência artificial é uma área na Ciência da Computação responsável por simular a inteligência e o comportamento humano usando apenas máquinas. Indo um pouco na contramão do imaginário comum estimulado pelos filmes de ficção científica, o objetivo da inteligência artificial é de executar atividades humanas desde as mais simples até as mais complexas (Igor Pimenta *et* al., 2021).

O aprendizado profundo são algoritmos eficientes para aprender automaticamente, assim como a inteligência artificial, porém com um detalhe de que a aprendizagem é supervisionada ou semi supervisionada em camadas utilizando modelos de abstração de alto nível, com várias camadas de processamento, onde ocorrem diversas transformações durante o aprendizado.

Aprendizado profundo é um subconjunto do *machine learning* baseado em redes neurais artificiais. O processo de aprendizado é profundo porque a estrutura das redes neurais artificiais consiste em várias camadas: de entrada, saída e oculta. Cada camada contém unidades que transformam os dados de entrada em informações que a próxima camada pode usar para executar uma determinada tarefa preditiva. Graças a essa estrutura, um computador pode aprender por meio de seu próprio processamento de dados (Microsoft *et* al., 2022).

Estudos analisam que grande parte das redes neurais são organizadas em camadas de nós “*feed-forward*”, o que significa que os dados se movem por elas em apenas uma direção. Um nó individual pode estar conectado a vários nós na camada abaixo dele, dos quais recebe dados, e vários nós na camada acima dele, para os quais envia dados.

As redes neurais são um meio de fazer aprendizado de máquina, no qual um programa aprende a realizar alguma tarefa analisando exemplos de treinamento. As redes neurais são sistemas de computação com nós interconectados que funcionam com base nos neurônios do cérebro humano (SAS Institute *et* al., 2022).

Uma artificial neural network (ANN) é um sistema computacional eficiente cujo tema central é emprestado da analogia de redes neurais biológicas. O objetivo principal dela é desenvolver um sistema que possa realizar várias tarefas computacionais mais rapidamente do que os sistemas tradicionais. A ANN imita o processo do cérebro humano e resolve os problemas não lineares, é por isso que é amplamente utilizado para prever e calcular a tarefas complicadas (Bhanja S., Das A. *et* al., 2018).

A arquitetura de uma rede neural que é muito utilizada no meio acadêmico chamada de "feed-forward", ou totalmente conectada, possui uma camada de entrada chamada de “input layer”, nessa camada é feita a entrada de dados e valores, após a camada de entrada receber os valores de entrada os valores são passados a n camadas ocultas chamada de “hidden layers” onde os dados são memorizados e processados, após isso os valores são passados para a camada de saída também chamada de “output layer”, diferente das RNNs que são capazes de empregar a rede para sustentar a memória de eventos recentes e construir conexões entre cada unidade de uma rede, portanto, é totalmente adequada para as previsões econômicas.

Uma recurrent neural network (RNN) é uma classe de redes neurais que inclui conexões ponderadas dentro de uma camada (em comparação com as redes de feed-forward tradicionais, onde conecta alimentação apenas para camadas subsequentes). Como as RNNs incluem loops, elas podem armazenar informações ao processar novas entradas. Essa memória os torna ideais para tarefas de processamento onde as entradas anteriores devem ser consideradas (como dados da série temporal). Por esta razão, as redes de deep learning atuais são baseadas em RNNs (Jones T. *et* al., 2017).

Outra arquitetura de rede neural que é muito utilizada para realizar previsões econômicas é a Long short-term memory (LSTM), a LSTM tem três portas diferentes para remover os problemas em células RNN e também são capazes de processar pontos de dados únicos ou sequências inteiros, muitos estudos têm sido realizados sobre formas de previsão de mercado com esse tipo de arquitetura e a mesma será utilizada durante o trabalho.

A partir da problemática apresentada, este trabalho tem como objetivo realizar pesquisas de base de dados de determinados ativos do mercado financeiro, para utilizar como dados de entrada referentes ao deslocamento de preço do mesmo no mercado financeiro. Com estes dados de entrada um framework pode ser usado para treinar arquiteturas de redes neurais, o mesmo chama-se Tensorflow, e é capaz de lidar com grandes quantidades de dados e aplicar testes de modelos de redes neurais para tentar prever movimentos futuros.

É importante salientar que com a linguagem de programação python, juntamente com a biblioteca Tensorflow, é possível realizar diversos treinamentos de redes neurais com diversos tipos de refinamento de dados e algoritmos disponíveis para melhorar o valor preditivo dos modelos.

O mesmo trabalho irá contribuir em novas pesquisas relacionadas a área de predição de preços de ativos do mercado financeiro como também pode servir de utilidade para desenvolvimento de um futuro software para auxiliar iniciantes nos investimentos.

Para facilitar a visualização dos resultados, é possível passá-los para gráficos comparativos e assim comparar os resultados e avaliar os desempenhos para identificar com maior facilidade os dados relevantes.

**2. PROPOSTA DO TRABALHO**

A proposta deste trabalho é utilizar os modelos de aprendizado de máquina criados para realizar testes com arquiteturas de redes neurais recorrentes, através da plataforma de programação que possibilita a criação de modelos de aprendizado de máquina. Para treinamentos e testes de alguns papéis da bolsa de valores ibovespa (Índice da Bolsa de Valores de São Paulo), foram selecionados os pápeis com grandes volumes de negociação para um melhor balenceamento das redes durante os testes, são os seguintes: Petrobras com papel representado por “PETR4”, Magazineluiza com papel representado por “MGLU3” e Ambev com papel representado por “ABEV3”, espera-se mensurar o valor preditivo de cada modelo.

Realizar treinamentos de modelos de aprendizado de máquina, utilizando como base a criação de modelos, arquiteturas de redes neurais recorrentes através da plataforma de programação do Google, o Colab. Este é um serviço de nuvem gratuito hospedado pelo próprio Google para incentivar a pesquisa de Aprendizado de Máquina e Inteligência Artificial. É uma ferramenta que permite que você misture código fonte (geralmente em python) e texto rico (geralmente em markdown) com imagens e o resultado desse código, uma técnica conhecida como: notebook (“caderno”, em português) (Thiago G. S. *et* al., 2022). Em conjunto a plataforma de programação utiliza-se frameworks que permitem a criação, treinamento e validação de modelos de rede neural. Após a criação e organização do código para cada modelo, são realizados treinamentos dos modelos e validação dos resultados, com a finalização de todos os experimentos, os resultados de todos modelos serão comparados para verificar os seus respectivos valores preditivos.

No presente trabalho, diversas pesquisas foram feitas envolvendo algoritmos de inteligência artificial e aprendizado profundo, que serão utilizados durante os testes. Além disso, pesquisas foram feitas para encontrar melhores arquiteturas de redes neurais para trabalhar com análises de predição de preços da bolsa de valores ibovespa, o Ibovespa foi criado em janeiro de 1968 e é um índice de retorno total, ou seja, além de considerar as variações nos preços dos ativos que fazem parte de sua carteira teórica, ele reflete o impacto do pagamento de todas as categorias de proventos das empresas emissoras dessas ações. O índice funciona como um termômetro do mercado acionário do Brasil e mede, através de um sistema de pontos baseado em reais, o desempenho médio de uma carteira teórica com as ações mais representativas e negociadas em Bolsa (Clear *et* al., 2022). Os valores financeiros da ação da Petrobras com o papel da bolsa de valores ibovespa representado por “PETR4”, Magazineluiza com papel representado por “MGLU3” e Ambev com papel representado por “ABEV3”, foram obtidos através da api Yahoo Finance, e um modelo foi obtido através de pesquisas, onde o modelo encontrado foi testado e resultados de predição foram encontrados.

Então, com este arquivo modelo, onde a predição e a arquitetura de rede neural utilizada é a LSTM, mais modelos foram criados, testes e treinamento foram feitos, como também modelos utilizando diferentes tipos de abordagem na formulação, adicionando mais camadas ao conjunto da arquitetura, e a utilização da arquitetura de rede neural recorrente, chamada Gated Recurrent Unit (GRU). Em português, Gated Recurrent Unit (GRU) é um mecanismo de bloqueio em recurrent neural network (RNN) semelhante a uma Long Short-Term Memory (LSTM), mas sem um output gate (DeepAI *et* al., 2022).

A fim de obter diferentes resultados, e de testar diferentes tipos de recursos que os frameworks disponibilizam para a realização da iteração dos testes das arquiteturas de redes neurais, serão realizados testes utilizando modelos com Dropout e sem Dropout. O Dropout funciona zerando aleatoriamente uma porcentagem de neurônios nas camadas da rede neural. Dropout é uma técnica utilizada na maioria das redes neurais modernas para prevenir sobre-ajustamento (Srivastava *et* al., ‎2014). Além disso, como métrica para a avaliação do modelo preditivo, será utilizada uma predição baseada no erro quadrático médio, MSE (da sigla em inglês Mean Squared Error), sendo definido como média do quadrado da diferença entre os valores reais e estimados. O MSE é usado para verificar quão próximas as estimativas ou previsões estão dos valores reais.

Ao final será realizado o cálculo da raiz quadrática do erro médio, RMSE (da sigla em inglês Root Mean Squared Error) é apenas a raiz quadrada do MSE, onde o erro retorna à unidade de medida do modelo, métrica para comparação que é utilizada em séries temporais por ser mais sensível a erros maiores devido ao processo de quadratura.

Ao final, os resultados obtidos dos cálculos realizados em cada um dos modelos serão comparados para então verificar qual deles teve um resultado mais próximo a zero, pois quanto mais próximo a zero o cálculo resultante do RMSE, menor é o erro do modelo.

Também é de interesse criar uma tabela para comparação dos resultados dos modelos em tabela do Microsoft Excel, um programa de planilha, que é usado para registrar e analisar dados numéricos, uma planilha como uma coleção de colunas e linhas, onde o ponto onde uma coluna e uma linha se encontram é chamada de célula, as letras alfabéticas geralmente são atribuídas a colunas e os números geralmente são atribuídos a linhas, embora você possa alternar isso, o endereço de uma célula é dado pela letra que representa a coluna e o número que representa a linha (Edivaldo *et* al., ‎2017), pois utilizando o Excel se torna mais fácil de realizar a criação de gráficos, pois os mesmos auxiliam na visualização e identificação de modelos com maior desempenho preditivo.

# 3. BASE DE DADOS

A fonte de entrada de dados ou dataset utilizado neste trabalho é amplamente utilizada por desenvolvedores de softwares de negociação que exigem a análise técnica de dados do mercado financeiro. O nome dela é Yahoo Finance, application programming interface (API) gratuita disponibilizada pelo Yahoo para consultar dados de várias ações brasileiras e do exterior, simples de ser utilizada, a mesma retorna os dados em tempo real com o histórico completo com os seguintes atributos: “data”, “abertura”, “máxima”, “mínima”, “fechamento”, “volume” e “dividendos”. Todos esses dados são obtidos em tempo real e são consultados diretamente pela API, e o resultado da consulta desta API retorna direto como um Pandas DataFrames, que é basicamente um objeto com um conjunto de funções para manipulação de dataset para realizar testes e treinamento de redes neurais, os dados colhidos em tempo real utilizando Yahoo Finance api foram utilizados nos experimentos desenvolvidos no presente trabalho.

No trabalho de Mehtab S. e San J. (2020) para treinar e testar os modelos preditivos propostos, os valores históricos do índice National Stock Exchange Fifty (NIFTY) para o período entre 29 de dezembro de 2014 a 31 de julho de 2020, os registros do índice NIFTY foram baixados na forma de um arquivo de variáveis separadas por vírgula (CSV) do arquivo site Yahoo Finance. Os seguintes atributos constituíram os registros diários de Valores do índice NIFTY 50: “data”, “aberto”, “alto”, “baixo”, “próximo” e “volume”. Os modelos foram testados nos valores do índice NIFTY 50 durante o período de 31 de dezembro de 2018 (que era uma segunda-feira) a 31 de julho de 2020 (que era uma sexta-feira). Os cinco modelos que foram propostos incluem dois modelos baseados em rede neural convolucional (CNN) e três modelos baseados em Long Short-Term Memory (LSTM).

Os modelos preditivos dos autores Mehtab J., San J. e Dasgupta S. (2021) foram construídos para prever com precisão o valor de abertura do preço das ações da empresa Bharat Forge listada no NSE, Índia. A proposta inclui três modelos de regressão que são construídos em redes neurais convolucionais (CNN’s) e quatro modelos preditivos baseados em Long Short-Term Memory (LSTM). Usando dados de preços de ações da empresa Bharat Forge para o período de 31 de dezembro de 2012 (que era uma segunda-feira) a 9 de janeiro de 2015 (que era uma sexta-feira). Durante este período de tempo, os dados do preço das ações foram capturados em um intervalo de 5 minutos pela ferramenta Metastock. O treino e o teste são realizados a partir de conjuntos de dados que consistem em 19.500 e 20.500 registros, respectivamente e cada registro consiste nos seguintes atributos: “data”, “intervalo de tempo”, “abertura”, “alta”, “baixa”, “fechamento” e “volume”.

No trabalho dos autores, foram utilizados dados de séries temporais da empresa Bombay Stock Exchange (BSE) de New York Stock Exchange (NYSE), os dados foram utilizados a fins de treinamento, validação e teste. Os índices básicos como preço de abertura, preço alto e preço baixo são tomados para o preço de entrada e de fechamento do resultado. Os dados da BSE de 1 de janeiro de 2016 a 31 de dezembro de 2017, foram coletados. O total dos dados representavam 493 registros, 70% dos registros foram utilizados como treinamento, representando 345 registros, enquanto 15% foram rotulados para validação, representando 74 registros, e mais 15% foram utilizados para a finalidade do teste, representando 74 registros.

Antes de aplicar esses dados no modelo de previsão, é feito o pré-processamento dos dados. Na fase de pré-processamento, funções de normalização, limpeza e algoritmos são aplicadas para dimensionar os dados em um determinado intervalo, checar valores nulos para remover estes valores, ou torna-los igual a zero, como também a remodelagem dos dados em matrizes aceitas pelas arquiteturas de redes neurais a serem utilizadas.

O pré-processamento de dados é um procedimento que serve para preparar os dados brutos e torná-los adequados para o modelo de aprendizado de máquina. É o primeiro e crucial passo ao criar um modelo de aprendizado de máquina. Na criação de um projeto de aprendizado de máquina, nem sempre os dados estão limpos, de forma adequada a serem utilizados nos modelos, muitas vezes os mesmos precisam ser formatados, para isso, usamos a tarefa de pré-processamento de dados.

Como primeiro passo para o pré-processamento dos dados é necessário utilizar a função “isna()” do Pandas DataFrame que checa a quantidade de valores nulos nos dados de entrada, em caso de conter algumas ações podem ser tomadas, como utilizar a função “dropna()”, que remove os valores nulos que contém nos dado de entrada, como também pode ser utilizada a função “fillna()”, onde um parâmetro pode ser passado para essa função em que os valores nulos serão substituídos pelo valor passado como parâmetro, como por exemplo “0”. Em seguida, esse conjunto de dados é filtrado para pegar apenas a coluna “Close” onde se encontram os valores de fechamento do DataFrame, este será o conjunto de dados a ser utilizado pelo modelo, mas ainda é necessário realizar a normalização para que seja preparado para a utilização como entrada de dados nos modelos de previsão de redes neurais para a finalidade de treinamento, validação e teste.

**4. METODOLOGIA DA PESQUISA**

A Figura 1 a seguir apresenta um resumo da metodologia que será utilizada neste trabalho, destacando-se em vermelho os atributos pelos quais as caracterizam.

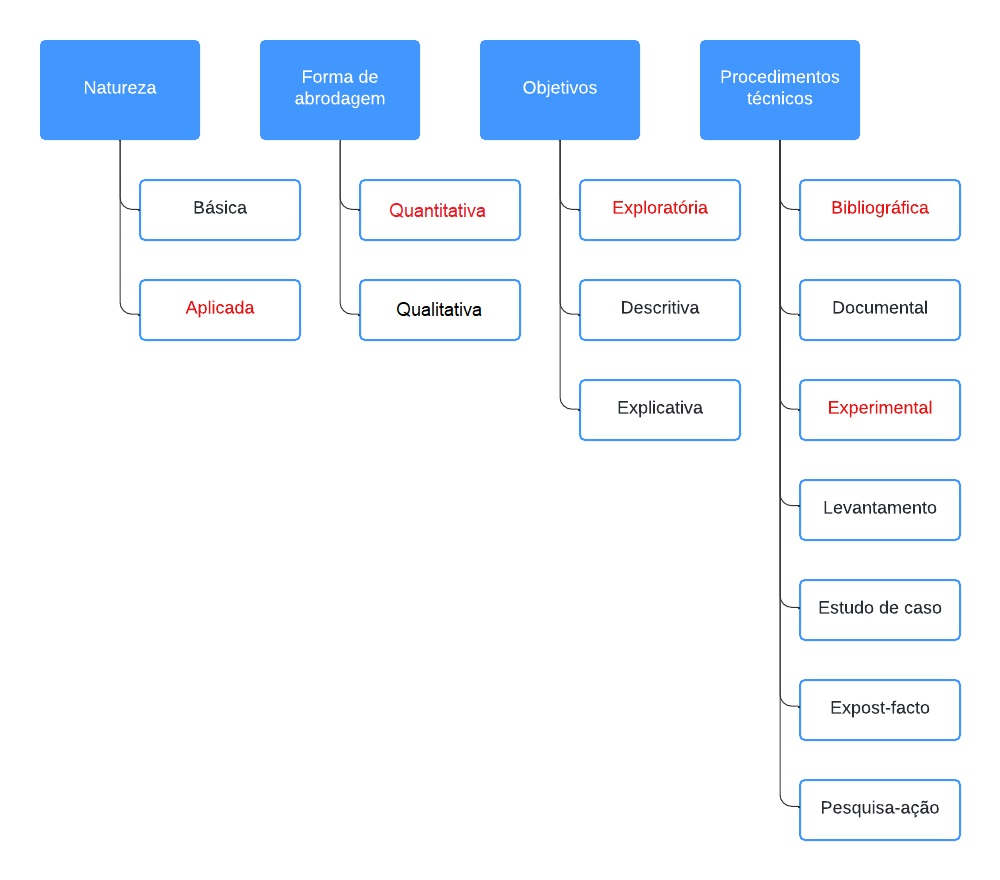


Figura 1: Classificação da pesquisa (adaptado de BEZ, 2011).

Este trabalho é caracterizado de natureza aplicada, pois o protótipo do mesmo resultará em oferecer um experimento de comparação entre análise de diversos modelos de redes neurais aplicados para prever valores de ativos selecionados do mercado financeiro.

O projeto será abordado de forma quantitativa, pois a avaliação do mesmo será conduzida com base nos valores observados dos modelos.

É possível enquadrar o presente trabalho como conceito de pesquisa exploratória, pois procura conhecer os valores preditivos de modelos de redes neurais.

Em relação a procedimentos técnicos, o presente trabalho pode ser considerado como bibliográfico e experimental, isto em função da necessidade de pesquisas de datasets, algoritmos e modelos de redes neurais. Com o intuito de conhecer o estado-da-arte de redes neurais, a fim de identificar as melhores práticas ao utilizá-las, como também suas limitações.

Como um primeiro protótipo já foi obtido e testado com sucesso, o próximo passo para continuar a pesquisa, serão feitas buscas de arquiteturas de redes neurais que se destaquem na identificação de padrões e refinamento dos dados e realizar a modificação do estilo de arquitetura de rede neural através da IDE de programação Colab, do Google, criando novos arquivos de código no intuito de testar outros tipos de redes neurais como por exemplo a GRU.

Para facilitar a visualização dos resultados, serão criados gráficos comparativos para observar o MSE e o RMSE resultante de cada um dos modelos, o erro quadrático médio (Mean Square Error - MSE) mede a quantidade de erro em modelos estatísticos. Ele avalia a diferença quadrática média entre os valores observados e previstos. Quando um modelo não tem erro, o MSE é igual a zero. À medida que o erro do modelo aumenta, seu valor aumenta (F. Jim *et* al., 2022).

Um conjunto de dados pode ter vários erros residuais, mas apenas um MSE. Como o erro quadrático médio é calculado usando números quadrados, é sempre um valor positivo (B. Alexa *et* al., 2022).

Segundo (G. Satish *et* al., 2020) o RMSE indica:

* O ajuste absoluto do modelo aos dados.
* Fornece o erro médio de previsão do modelo em unidades da variável de interesse.
* São pontuações orientadas negativamente, o que significa que valores mais baixos são melhores.

No momento em que o RMSE resultante de todos os modelos forem obtidos, os mesmos serão passados para gráficos e tabelas, e então analisados, comparados e avaliados afim de verificar qual deles terá um melhor desempenho preditivo.

**5. REFERENCIAL TEÓRICO**

Neste trabalho foram feitas diversas pesquisas de algoritmos de inteligência artificial e aprendizado profundo que serão utilizados durante os testes. Além disso, pesquisas foram feitas para encontrar melhores arquiteturas de redes neurais para trabalhar com análises de predição de preços da bolsa de valores ibovespa.

As pesquisas são fundamentais para o trabalho já que as mesmas servem como embasamento para fundamentar o conhecimento a respeito dos assuntos que norteiam este trabalho e que podem influenciar diretamente na performance de previsão dos modelos que serão testados.

**5.1 REDES NEURAIS ARTIFICIAIS**

As redes neurais artificiais (RNA) tentam imitar a rede de neurônios que compõem um cérebro humano para que os computadores tenham a opção de entender as coisas e tomar decisões de maneira humana. A RNA é projetada por computadores de programação para se comportar simplesmente como células cerebrais interconectadas.

Normalmente, os nós são organizados em uma camada ou vetor, com a saída de uma camada servindo como entrada para a próxima camada e possivelmente outras camadas. Essas camadas normalmente são organizadas de forma que a conexão inicial comece pelos neurônios na camada de entrada, onde a entrada de dados será passada para as camadas subsequentes através das conexões entre os neurônios para sua(as) camadas intermediárias, onde pesos são atribuídos para os neurônios para então resultar na camada de saída(A. Ponce *et* al., 2020).

Na Figura 2 é possível observar um exemplo de uma RNA juntamente com seus neurônios e ligações através de suas conexões para as camadas.

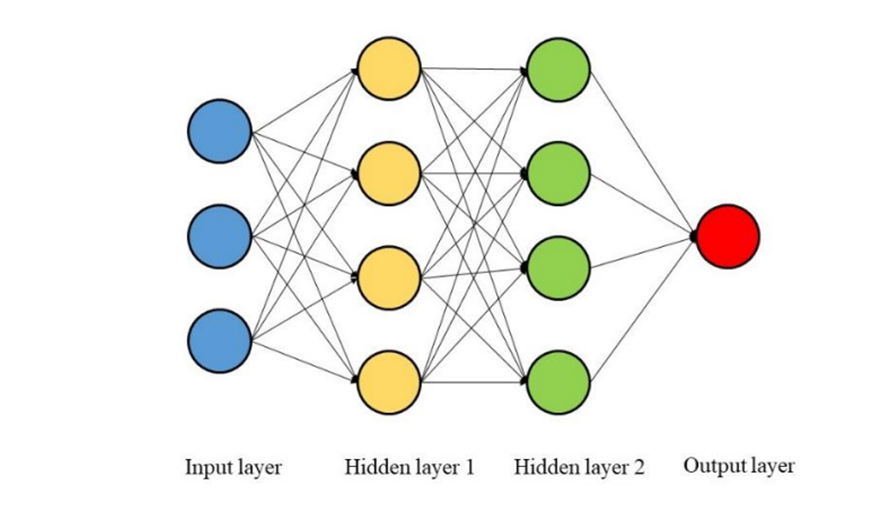


Figura 2 – Arquitetura RNA.

Um neurônio pode ser conectado a todos ou a um subconjunto dos neurônios na camada subsequente, com essas conexões simulando as conexões sinápticas do cérebro. Os sinais de dados ponderados que entram em um neurônio simulam a excitação elétrica de uma célula nervosa e, consequentemente, a transferência de informações dentro da rede ou do cérebro. Os valores de entrada para um elemento de processamento, são multiplicados por um peso de conexão que simula o fortalecimento das vias neurais no cérebro. É através do ajuste das forças ou pesos de conexão que o aprendizado é emulado nas redes neurais artificiais.

Estima-se que o cérebro humano tenha tantas células quanto a Via Láctea tem planetas. No caso humano, algo como 100 bilhões de neurônios. Cada um desses neurônios teria a capacidade para manter 1.000 conexões (sinapses) com outros neurônios, totalizando 100 trilhões (T. Jokura *et* al., 2019). No cérebro humano, os dados são armazenados de forma a serem distribuídos, e podemos extrair mais de uma parte desses dados quando necessário de nossa memória paralelamente, podemos dizer que o cérebro humano é composto de processadores paralelos.

A RNA é usada para identificar a inter-relação de variáveis ​​de entrada e variáveis ​​de saída. As variáveis ​​de saída podem ser previstas por uma estrutura de RNA adequada usando o banco de dados de entrada. No entanto, uma das desvantagens de uma RNA é que ela é um modelo de caixa preta, portanto, não poderia explicar o mecanismo que ocorre dentro do processo. Na modelagem caixa preta da RNA, as variáveis ​​de saída são previstas com base em suas relações individuais com as variáveis ​​de entrada. Portanto, na modelagem de RNA, o tamanho da amostra ou os dados devem ser grandes para evitar a previsão de overfitting. Um cenário de overfitting ocorre quando, nos dados de treino, o modelo tem um desempenho excelente, porém quando utilizamos os dados de teste o resultado é ruim (Didática Tech *et* al., 2020).

A estrutura da RNA contém vários neurônios nos quais os cálculos de interconexão são processados. Os pesos associados aos neurônios teriam um papel para fortalecer essas interconexões. Normalmente, a importância dos fatores de entrada seria testada usando uma análise de componentes principais. Por causa disso, apenas fatores de alta importância devem ser selecionados para modelagem de RNA. Conforme podemos imaginar, dados de treino são os dados que serão apresentados ao algoritmo de machine learning para criação do modelo. Estes dados costumam representar cerca de 70% da totalidade dos dados (Didática Tech o *et* al., 2022). Além disso, existem os dados de teste que são os dados que serão apresentados ao modelo após a sua criação, simulando previsões reais que o modelo realizará, permitindo assim que o desempenho real seja verificado. Estes dados costumam representar cerca de 30% da totalidade dos dados (Didática Tech *et* al., 2022).

O termo treinamento de uma rede neural consiste basicamente em fazê-la adotar valores de pesos e limiares baseados em suas amostras de entrada, de modo que qualquer outra amostra futura apresentada a ela seja corretamente classificada, isto é, apresentar as amostras/dados na entrada da rede e direcionar uma resposta desejada fazendo com que a rede seja obrigada a modificar seus pesos e limiares. Assim, qualquer outro valor apresentado a ela depois que a rede estiver corretamente treinada produzirá uma saída ou resultado igual ou próximo do desejado.

O Google usa uma rede neural de 30 camadas para alimentar o Google Fotos, bem como suas recomendações de “assistir a seguir” para vídeos do YouTube. O Facebook usa redes neurais artificiais para seu algoritmo DeepFace, que pode reconhecer rostos específicos com 97% de precisão. É também uma ANN que potencializa a capacidade do Skype de fazer traduções em tempo real (B. Marr *et* al., 2018).

**5.2 PROCESSO DE APRENDIZAGEM**

Uma rede neural artificial é geralmente uma rede computacional baseada em redes neurais biológicas que constroem a estrutura do cérebro humano. Assim como um cérebro humano possui neurônios interligados entre si, as redes neurais artificiais também possuem neurônios que estão ligados entre si em várias camadas das redes, esses neurônios são conhecidos como nós.

Os nós passam por um processo no momento em que é aplicada uma entrada de dados na RNA, é um processo no qual ela se familiariza com o problema que precisa resolver. Na prática, geralmente esses dados são coletados com base no que precisa-se criar as previsões, ou classificação, ou qualquer outro processamento. Esses dados são chamados de, conjunto de treinamento. Com base no comportamento durante o treinamento e na natureza do conjunto de treinamento, algumas classes de aprendizado são destacadas:

* Aprendizado não supervisionado – Deve-se dispor das amostras e das respectivas saídas desejadas para que os pesos e limiares sejam ajustados continuamente pelo algoritmo de aprendizagem (S. Eduardo et al., 2016).
* Aprendizado supervisionado – A saída deve se auto organizar em relação às particularidades do conjunto de amostras e assim identificar subconjuntos similares. Seus pesos e limiares são ajustados pelo algoritmo de aprendizagem de modo a refletir estas particularidades. (S. Eduardo et al., 2016).
* Aprendizado por reforço – Similar ao treinamento supervisionado, contudo seu algoritmo visa ajustar os pesos e limiares baseando-se em informações procedentes da interação com o sistema mapeado, visando reforçar as respostas satisfatórias (S. Eduardo et al., 2016).

O aprendizado supervisionado por ser o tipo de treinamento mais comumente utilizado, conforme pode ser observado na Figura 3, o conjunto de treinamento é representado na figura como “Padrões de entrada”, quando os dados de entrada são submetidos à rede neural, os seus pesos iniciais terão um resultado difente em sua saída, e ao se obter um valor de saída que será comparado a um valor desejado, e o resultado é o erro, com esse determinado erro que é aplicado na nova iteração com um novo valor de entrada na rede neural, gerando novos valores de peso dos neurônios, este processo é feito até que uma saída desejada seja obtida.

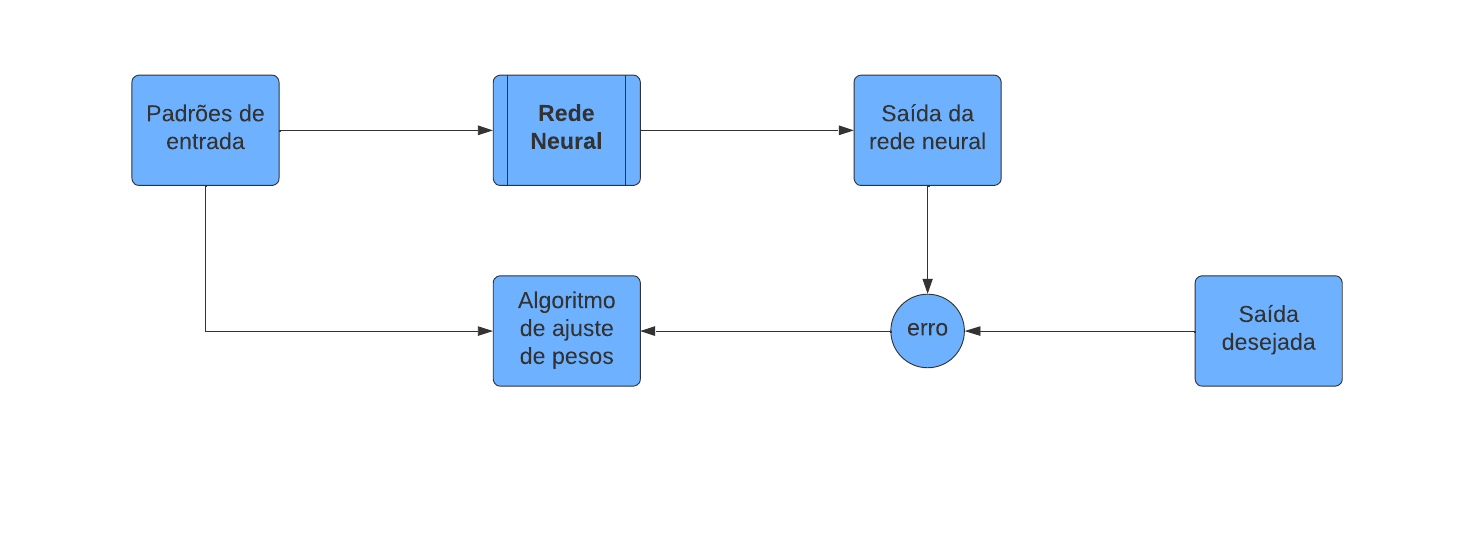


Figura 3 – Diagrama de aprendizado supervisionado.

O aprendizado supervisionado é mais comumente usado, ao obter um conjunto de treinamento que contém um vetor de valores de entrada e um vetor de valores de saída desejados, a rede calcula a saída para uma das entradas, a função de custo calcula o vetor de erro, este erro indica o quão próximo o palpite está da saída desejada, uma das funções de custo mais utilizadas é a do erro quadrático médio.

**5.3 RECURRENT NEURAL NETWORK (RNN)**

Recurrent neural network (RNN) é um tipo de rede neural onde a saída da etapa anterior é alimentada como entrada para a etapa atual. Nas redes neurais tradicionais, todas as entradas e saídas são independentes umas das outras, mas em casos como quando é necessário prever a próxima palavra de uma frase, as palavras anteriores são necessárias e, portanto, há a necessidade de lembrar as palavras anteriores. A principal e mais importante característica da RNN é o hidden state, que lembra algumas informações sobre uma sequência.

A RNN converte as ativações independentes em ativações dependentes, fornecendo os mesmos pesos e desvios para todas as camadas: “input layer”, “hidden state” e “output layer”, reduzindo assim a complexidade de aumentar os parâmetros e memorizando cada saída anterior, fornecendo cada saída como entrada para a próxima camada oculta. Portanto, essas três camadas podem ser unidas de modo que os pesos e o valor de todas as camadas ocultas sejam os mesmos, em uma única camada recorrente.

A RNN consiste em várias camadas recorrentes sucessivas, e essas camadas são modeladas sequencialmente para mapear sequências de dados, suas estruturas de rede são estáveis e são efetivamente utilizadas para alcançar o processo de classificação dos dados.

A RNN foi criada porque havia alguns problemas na rede neural:

* Difícil de lidar com dados sequenciais.
* Considera apenas a entrada atual.
* Não é possível memorizar entradas anteriores.
* Uma RNN pode manipular dados sequenciais, aceitando os dados de entrada atuais e as entradas recebidas anteriormente. RNNs podem memorizar entradas anteriores devido à sua memória interna.

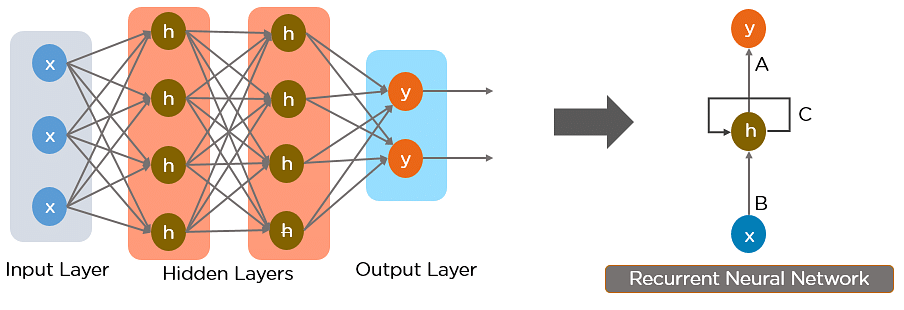


Figura 4 - Rede Neural para Rede Neural Recorrente.

Na Figura 4 observa-se como seria uma rede feed-forward transformada para uma arquitetura de rede neural recorrente, onde a “Input Layer”, ou camada de entrada, os dados de entradas são correspondentes a “x”, as camadas ocultas da rede neural, ou “Hidden Layers”, momento que é dado novos pesos aos nós da rede neural, e a “Output Layer”, ou camada de saída, que se equipara ao resultado obtido da rede. No caso da conversão para uma RNN as iterações ocorrem repetidas vezes ao mesmo conjunto de dados de entrada, o que faz com que essa repetição utilizando o mesmo nó para repetidos testes com dados de entrada seja algo recorrente.

Redes neurais recorrentes sofrem de memória de curto prazo. Se uma sequência for longa o suficiente, elas terão dificuldade em transportar informações das etapas anteriores para as posteriores. Portanto, se você estiver tentando processar um parágrafo de texto para fazer previsões, as RNNs poderão deixar de fora informações importantes desde o início. Durante a etapa de backpropagation, as redes neurais recorrentes sofrem com o problema da dissipação do gradiente. Gradientes são valores usados para atualizar os pesos das redes neurais. O problema da dissipação, é quando este valor diminui à medida que se propaga novamente ao longo do tempo. Se um valor se torna extremamente pequeno, não contribui muito com o aprendizado (Data Science Academy *et* al., 2022).

**5.4 LONG SHORT-TERM MEMORY (LSTM)**

Na LSTM, enquanto o forget gate determina qual parte do estado anterior da célula deve ser retida, o input gate determina a quantidade de nova memória a ser adicionada, esses dois gates são independentes um do outro, o que significa que a quantidade de novas informações adicionadas pelo input gate é completamente independente das informações retidas pelo forget gate.



Figura 5 – Arquitetura LSTM (traduzido de Deep Learning Book).

Visualizando a Figura 6, percebe-se que na arquitetura LSTM o input gate e o output gate são acoplados por um forget gate. A RNN processa a sequência de vetores um por um, durante o processamento, ele passa o hidden state anterior para a próxima etapa da sequência. O hidden state atua como a memória das redes neurais, nele contém informações sobre dados anteriores que a rede já viu.

Uma função tanh garante que os valores fiquem entre -1 e 1, regulando assim a saída da rede neural. Já a função de ativação sigmóide, em vez de comprimir valores entre -1 e 1, comprime valores entre 0 e 1, isso é útil para atualizar ou esquecer os dados porque qualquer número multiplicado por 0 é 0, fazendo com que os valores desapareçam ou sejam “esquecidos”. Qualquer número multiplicado por 1 é o mesmo valor, portanto, esse valor permanece o mesmo ou é “mantido” e a rede pode aprender quais dados não são importantes e então, podem ser esquecidos ou quais dados são importantes e devem ser mantidos, conforme pode ser observado na Figura 6.

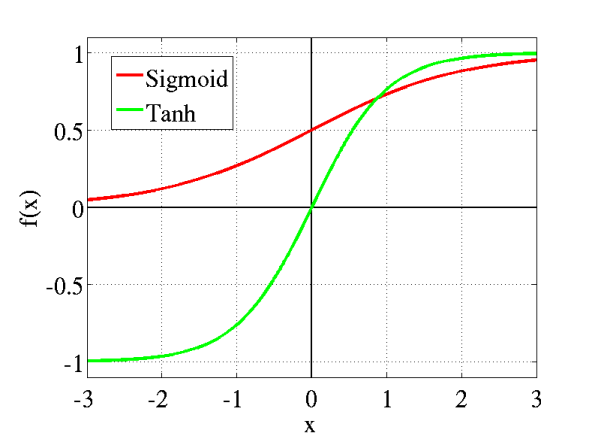


Figura 6 – Função Tahn versus Sigmoid (S. Sharma *et* al., 2014).

O forget gate decide quais informações devem ser descartadas ou mantidas, as informações do hidden state anterior e as informações da entrada atual são passadas pela função sigmóide, os valores saem entre 0 e 1, quanto mais próximo de 0 significa esquecer, e quanto mais próximo de 1 significa manter.

O input gate é responsável por atualizar a cell state, primeiro, o hidden state anterior é passado juntamente com a entrada atual para uma função sigmóide, isso decide quais valores serão atualizados transformando os valores entre 0 e 1. 0 significa não importante e 1 significa importante, também é passado o hidden state e a entrada atual para a função tanh para compactar valores entre -1 e 1 para ajudar a regular a rede, então é multiplicada a saída tanh pela saída sigmóide, a saída sigmóide decidirá qual informação é importante manter da saída tanh.

A cell state é calculada multiplicando o forget gate com a cell state anterior, isso dá a possibilidade de descartar valores na cell state se for multiplicado por valores próximos de 0, então, a saída do input gate faz uma adição pontual que atualiza a cell state para novos valores que a rede neural considera relevantes.

O output gate decide qual deve ser o próximo hidden state, lembrando que o hidden state contém informações sobre entradas anteriores, o hidden state também é usado para previsões, primeiro, é passado o hidden state anterior e a entrada atual para uma função sigmóide, em seguida, é passado a cell state recém-modificada para a função tanh, então é realizado uma multiplicação na saída tanh pela saída sigmóide para decidir quais informações o hidden state deve transportar, a saída é o hidden state, e a nova cell state e o novo hidden state são transportados para a próxima etapa de tempo.

A LSTM é uma arquitetura de recurrent neural network (RNN) que “lembra” valores em intervalos arbitrários. A LSTM é bem adequada para classificar, processar e prever séries temporais com intervalos de tempo de duração desconhecida. A insensibilidade relativa ao comprimento do gap dá uma vantagem à LSTM em relação a RNNs tradicionais (Deep Learning Book *et* al., 2022).

**5.5 GATED RECURRENT UNIT (GRU)**

A Gated Recurrent Unit (GRU), como o próprio nome sugere, é uma variante da arquitetura RNN e usa mecanismos de gating para controlar e gerenciar o fluxo de informações entre as células da rede neural. A estrutura da GRU permite capturar de forma adaptativa as dependências de grandes sequências de dados sem descartar informações de partes anteriores da sequência, esses gates são treinados para filtrar informações consideradas não necessárias, mantendo apenas o que é útil para se aperfeiçoar. Os gates são considerados como vetores que possuem valores entre 0 e 1, onde os valores são multiplicados por dados que são recebidos na entrada e do hidden state, caso o valor de algum dos vetores for 0, tanto na entrada quanto no hidden state, este valor será desconsiderado. Do contrário, se o valor for 1, significa que os dados correspondentes são importantes e serão usados, conforme pode ser observado na Figura 7.

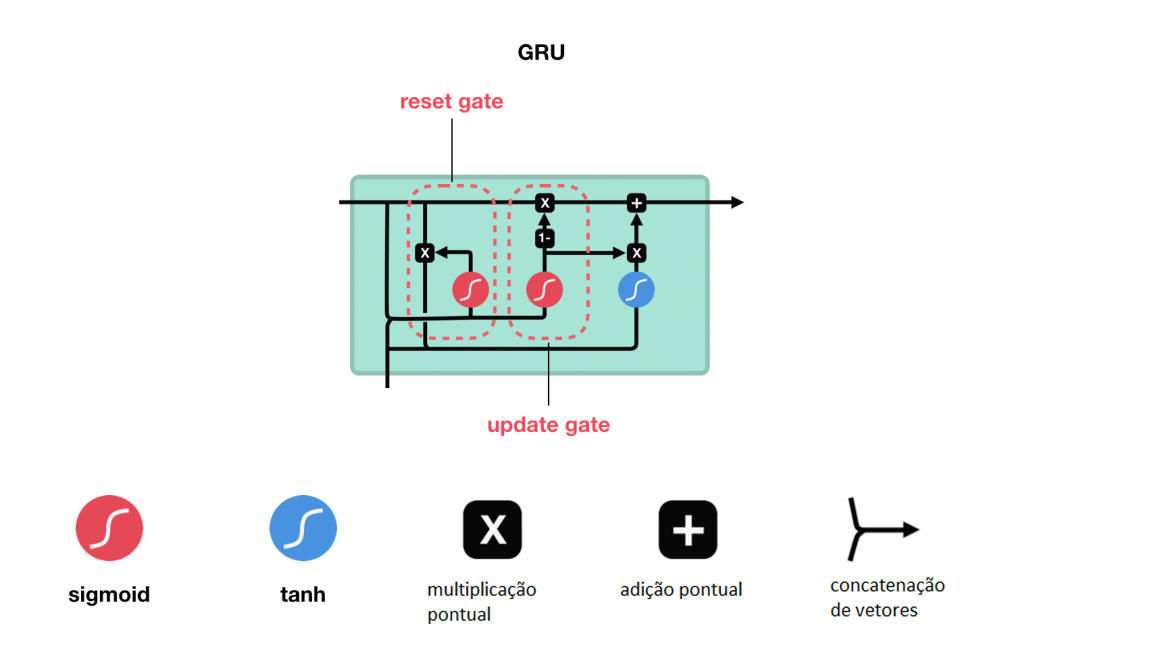


Figura 7 - Arquitetura GRU (traduzido de Deep Learning Book).

Visualizando a Figura 7, percebe-se que a arquitetura GRU têm mecanismos internos chamados de gate que podem regular o fluxo de informações. A RNN processa a sequência de vetores um por um, durante o processamento, ela passa o hidden state anterior para a próxima etapa da sequência, o hidden state atua como a memória das redes neurais, nele contém informações sobre dados anteriores que a rede salvou anteriormente.

Assim como na LSTM, a GRU possue a função tanh que garante que os valores fiquem entre -1 e 1, como também a função de ativação sigmóide que garante que os valores resultem entre 0 e 1, para atualizar ou esquecer os dados, ou manter quais dados são importantes.

O update gate decide quais informações descartar e quais novas informações adicionar. O reset gate é utilizado para decidir quanta informação passada deve ser esquecida.

**5.6 LSTM versus GRU**

GRUs e LSTMs foram criadas para resolver o problema do gradiente de fuga que o RNN padrão enfrenta, e ambas as variantes do RNN utilizam mecanismos de controle para controlar o fluxo de dependências de longo e curto prazo dentro da rede, embora GRUs e LSTMs contenham gates, a principal diferença entre essas duas estruturas está no número de gates e suas funções específicas. A função do update gate na GRU é muito semelhante aos gates de entrada e de esquecimento na LSTM, no entanto, o controle do novo conteúdo de memória adicionado à rede difere entre esses dois, conforme pode ser observado na Figura 8.

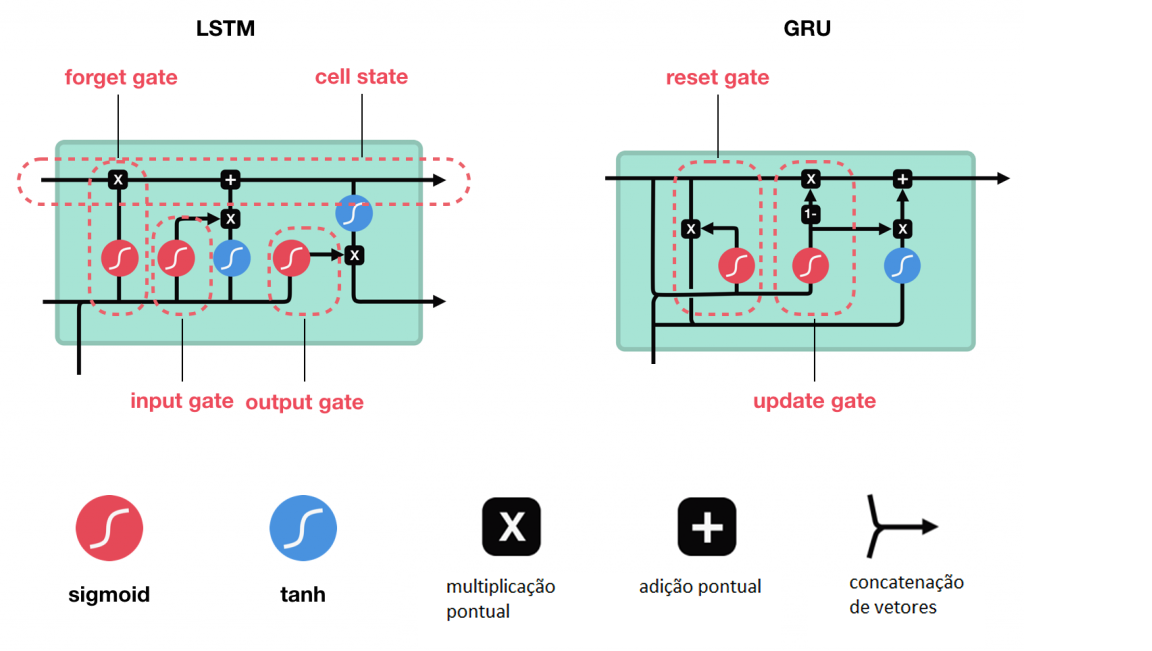


Figura 8 – Arquitetura GRU vs Arquitetura LSTM (traduzido de Deep Learning Book).

Outra diferença entre as estruturas é a falta do cell state na GRU, enquanto a LSTM armazena suas dependências de longo prazo na cell state e a memória de curto prazo no hidden state, a GRU armazena ambos em um único hidden state, no entanto, em termos de eficácia na retenção de informações de longo prazo, ambas as arquiteturas conseguem atingir seus objetivos de maneira eficaz.

A capacidade da GRU de manter dependências ou memória de longo prazo decorre dos cálculos dentro da célula GRU para produzir o hidden state, enquanto as LSTM’s têm dois estados diferentes passados entre as células, a cell state e o hidden state, que carregam a memória de longo e curto prazo, respectivamente, as GRU’s têm apenas um hidden state transferido entre as etapas de tempo, esse hidden state é capaz de manter as dependências de longo e curto prazo ao mesmo tempo devido aos mecanismos de controle e cálculos pelos quais o hidden state e os dados de entrada passam.

A GRU salva o estado da memória de longo prazo resultante dos cálculos dentro da célula para produzir o hidden state, enquanto as LSTMs possuem dois estados diferentes, a memória de longo e a de curto prazo, as mesmas são passadas entre as células a cell state e o hidden state, que carregam.

Existe um problema que pode ocorrer ao treinar uma rede neural recorrente, chamado de gradiente de fuga, o mesmo ocorre no momento de retropropagação, especialmente se a rede neural recorrente estiver processando sequências longas ou tiver várias camadas. O gradiente de erro é calculado no momento do treinamento e é usado para atualizar o peso da rede na direção certa e na intensidade correta, porém, este gradiente é calculado a partir do final da rede. Devido a isso, durante a retropropagação, os gradientes serão multiplicados em matrizes e encolherão ou explodirão exponencialmente para sequências longas, fazendo com que esse procedimento demore muito tempo. Caso o gradiente seja muito pequeno, isso significará que o modelo não realizará a atualização de seus pesos de forma eficaz, por outro lado, caso o gradiente for extremamente grande, fará com que o modelo fique instável.

O problema de gradiente de fuga é resolvido nas arquiteturas de rede neural LSTM e GRU, isso porque redes neurais recorrentes comuns sempre alteram todo o conteúdo do hidden state em cada etapa de tempo, já LSTMs e GRUs mantêm a maior parte do hidden state existente enquanto adicionam novo conteúdo sobre ele. Isso permite que os gradientes de erro sejam propagados de volta sem desaparecer ou explodir muito rapidamente devido às operações de adição. Embora LSTMs e GRUs sejam as correções mais utilizadas para resolver o problema de gradiente de fuga, outra solução para o problema de explodir gradientes é o recorte de gradiente. O recorte define um valor limite definido nos gradientes, isso significa que, mesmo que um gradiente aumente além do valor predefinido durante o treinamento, seu valor ainda será limitado ao valor predefinido anteriormente, devido a isso, a direção do gradiente permanece não irá ser alterado, apenas a magnitude do gradiente será alterada.

As GRU’s e LSTM’s tentam resolver o problema do gradiente de fuga que pode vir com recurrent neural network padrão, uma GRU pode ser considerada uma variação da Long Short-Term Memory (LSTM) porque ambas têm um design semelhante e produzem resultados iguais em alguns casos. As GRU’s são capazes de resolver o problema do gradiente de fuga usando o update gate e o reset gate. O update gate controla as informações que fluem para a memória e o reset gate controla as informações que fluem para fora da memória. O update gate e o reset gate são dois vetores que decidem quais informações serão passadas para a saída, eles podem ser treinados para manter informações do passado ou remover informações irrelevantes para a previsão (DeepAI *et* al., 2022).

**5.7 DROPOUT**

O dropout pode ser implementado em cada uma das camadas da neural, e pode ser usado com a maioria dos tipos de camadas, em qualquer uma ou em todas as camadas ocultas da rede, bem como na camada visível ou de entrada, mas não é usado na camada de saída.

No dropout é introduzido um novo hiperparâmetro que especifica a probabilidade de eliminação das saídas da camada ou, inversamente, a probabilidade de retenção das saídas da camada.

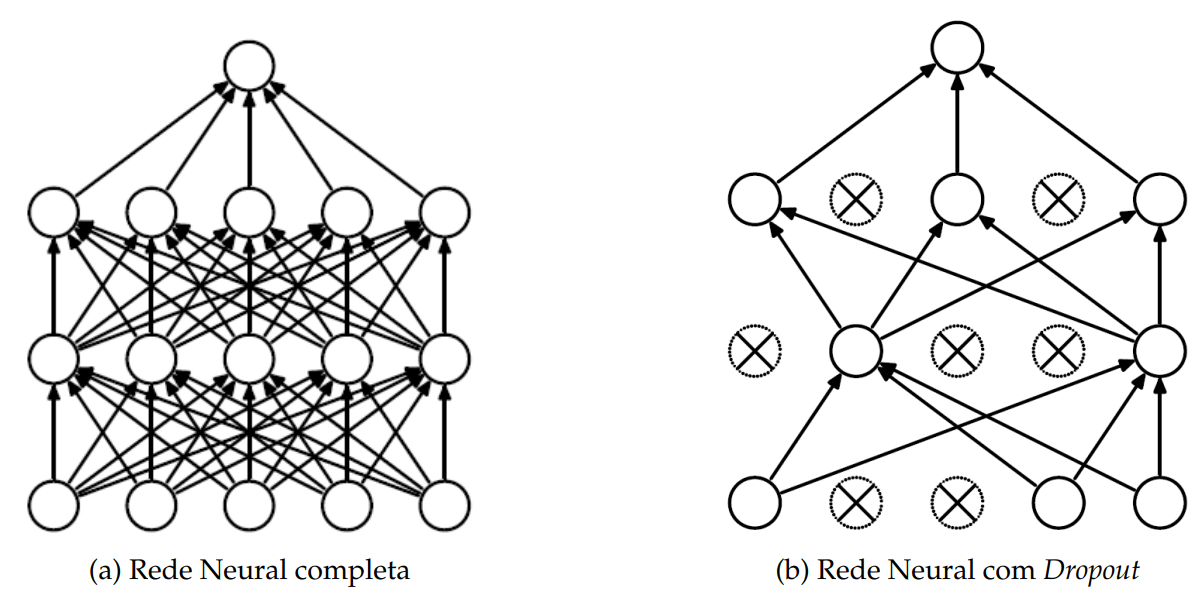


Figura 9 - Comparativo entre Rede Neural completa com a aplicação de Rede Neural com Dropout (Srivastava *et* al., 2014).

Dropout é um método de regularização que remove randomicamente neurônios da rede neural durante o treinamento para evitar co-adaptação entre neurônios vizinhos (Srivastava et al., 2014). É possível observar na Figura 9 como é feita a remoção randômica de neurônios durante o treinamento, comparado com a rede neural completa.

Um valor comum é uma probabilidade de 0,5 para reter a saída de cada nó em uma camada oculta e um valor próximo a 1,0, como 0,8, para reter as entradas da camada visível.

O objetivo das redes neurais é construir uma arquitetura de rede que consiga identificar elementos comuns em uma base de dados de modo a caracterizá-la. A rede com uma boa generalização possui uma boa performance tanto no conjunto de treino quanto no conjunto de testes. Um dos maiores desafios das redes neurais é encontrar o tamanho e complexidade da rede ideais para generalizar de forma satisfatória e o dropout auxilia nesse quesito.

Tamanho e complexidade exagerados, por exemplo, podem ter um efeito adverso, nesse caso, a rede tem uma performance excelente no conjunto de treinamento, mas apresenta dificuldade em identificar os padrões em novos conjuntos de dados, esse cenário é conhecido como sobreajuste, ou overfitting, e significa que a rede não aprendeu de fato a reconhecer os padrões de informação, apenas memorizou o comportamento da base de dados.

Por outro lado, sub-ajuste ou underfitting, ocorre quando a rede tem performance baixa tanto no conjunto de treinamento quanto no conjunto de teste, nesse caso, pode ser que tenha sido usada uma arquitetura inadequada ou com tamanho inferior ao necessário para a generalização.

Um exemplo de overfitting e de underfitting pode ser observado na Figura 10.

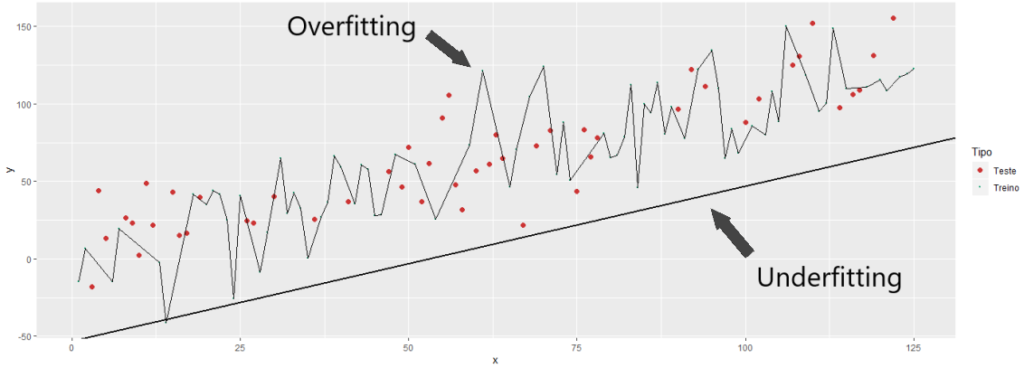


Figura 10 - Overfitting e Underfitting (Didática Tech *et* al., 2022).

Um cenário de overfitting ocorre quando, nos dados de treino, o modelo tem um desempenho excelente, porém quando utilizamos os dados de teste o resultado é ruim, neste caso, o modelo aprendeu tão bem as relações existentes no treino que acabou apenas decorando o que deveria ser feito e, ao receber as informações das variáveis preditoras nos dados de teste, o modelo tenta aplicar as mesmas regras decoradas, porém com dados diferentes esta regra não tem validade, e o desempenho é afetado (Didática Tech *et* al., 2022).

No cenário de underfitting, o desempenho do modelo já é ruim no próprio treinamento, o modelo não consegue encontrar relações entre as variáveis e o teste nem precisa acontecer, o modelo já pode ser descartado, pois não terá utilidade (Didática Tech *et* al., 2022).

**5.8 MEAN SQUARED ERROR (MSE)**

O erro quadrático médio, MSE (da sigla em inglês Mean Squared Error) é uma medida de quão perto uma linha ajustada está dos pontos de dados. Para cada ponto de dados, é calculada a distância verticalmente do ponto até o valor y correspondente no ajuste da curva (o erro) e eleva o valor ao quadrado. Em seguida, é calculada a soma de todos esses valores para todos os pontos de dados e, no caso de um ajuste com dois parâmetros, como um ajuste linear, divide pelo número de pontos menos dois. A quadratura é feita para que valores negativos não cancelem valores positivos, quanto menor o erro quadrático médio, mais próximo o ajuste está dos dados. O MSE tem as unidades ao quadrado de tudo o que é plotado no eixo vertical, conforme pode ser observado na Figura 11.

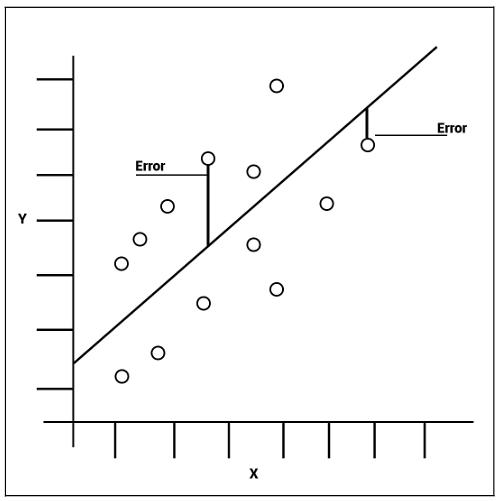
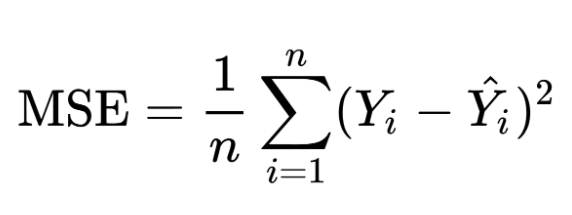


Figura 11 – Regressão linear (A. Gupta *et* al., 2022).

A fórmula do MSE é representada pela Fórmula 1 apresentada abaixo.



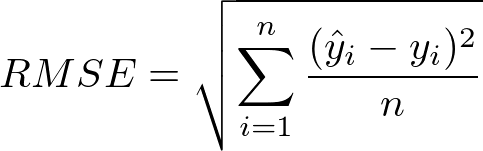
Fórmula 1 - Fórmula do erro quadrático médio.

Onde “n” representa o número de registros, “Σ” representa notação de soma, “Y”, representa o valor observado e “Ŷ” representa o valor da regressão.

É uma das funções mais comuns, usada inclusive nos problemas de regressão linear, ou seja, quando se sabe que a relação entre os dados a serem preditos e os dados de entrada é estritamente linear, essa função mede a diferença entre os resultados obtidos e o resultado real, eleva cada diferença ao quadrado, e depois calcula a média (D. Ceccon *et* al., 2019).

**5.9 ROOT MEAN SQUARED ERROR (RMSE)**

O RMSE será utilizado como métrica fundamental para a comparação de desempenho entre os modelos, onde comparado da seguinte forma, quanto menor o resultado obtido do RMSE, melhor é o desempenho apresentado pelo modelo. Com todas as etapas de treinamento, teste e validação, ou seja, com todo o procedimento de aprendizado supervisionado concluído, e com o MSE resultante de cada modelo, também calculamos o RMSE para cada um dos modelos, que é basicamente a raiz quadrada do MSE, onde o erro retorna à unidade de medida do modelo, a fórmula do RMSE é representada abaixo pela Fórmula 2.



Fórmula 2 - Fórmula da raiz do erro quadrático médio.

Onde “n” representa o número de registros, “Σ” representa notação de soma, “Y” representa o valor observado e “Ŷ” representa o valor da regressão.

Uma das características do RMSE é que os erros realizados pelas predições são elevados ao quadrado antes de ter a média calculada, por isso o mesmo se baseia do MSE. Portanto, pesos diferentes são atribuídos à soma de acordo com o “n”, conforme os valores de erros das instâncias de “Y” aumentam, o índice do RMSE consequentemente também aumentará consideravelmente. Ou seja, se houver um outlier, que são pontos fora de uma curva normal, no conjunto de dados utilizado, seu peso será maior para o cálculo do RMSE e, por consequência, a métrica será prejudicada, fazendo que se torne maior o valor do resultado.

A Raiz Quadrada do Erro Quadrático Médio, ou simplesmente RMSE em inglês, é uma excelente métrica para modelos de regressão, além de ser muito fácil de interpretar. Nada mais é que a diferença entre o valor que foi previsto pelo seu modelo e o valor real que foi observado, por exemplo, imagine que você está construindo um modelo para prever preço de casas. Esse modelo deveria ter previsto o valor de R$ 100 mil para um exemplo no seu conjunto, mas ele previu R$ 99 mil: esse -R$ 1 mil de diferença é o erro do seu modelo. Agora é só repetir esse processo para todo seu conjunto de treino ou teste, elevar o erro ao quadrado, tirar a média de todos os valores do conjunto e, por fim, calcular a raiz quadrada (P. Vasconcellos *et* al., 2018).

# 6. TRABALHOS CORRELATOS

Os trabalhos relacionados abaixo não empregam diversas arquiteturas de redes neurais, e demonstram a utilização de redes neurais na extração de características de séries econômico-financeiras para se obter predições no mercado de ações. Estes trabalhos motivaram a investigação da utilização de redes neurais para predição de retornos de ações.

**6.1 APRENDIZADO PROFUNDO PARA PREVISÃO DE AÇÕES**

No texto dos autores Nabipour, Nayyeri, Jabani, Mosavi (2020) foram confirmadas arquiteturas de redes neurais para superar cada um dos métodos ao prever séries temporais. De fato, uma tendência moderna em machine learning (ML), que é chamada de deep learning (DL), e pode considerar estruturas específicas não lineares profundas, outra ótima opção apontada são as recurrent neural network (RNN) que tem a habilidade do ponto de vista financeiro de séries temporais para extrair informações relevantes, outra ótima opção abordada no artigo “Deep learning for Stock Market Prediction” são as long short-term memory (LSTM) que é um subconjunto aprimorado de RNN.

**6.2 REDE NEURAL PARA PREVISÃO DE SÉRIES TEMPORAIS**

Bhanja e Das (2019) propuseram a utilização de deep recurrent neural network (DRNN) aplicado para prever o índice de ações do mercado de ações BSE e NYSE para os diferentes métodos de normalização. No artigo “Impact of Data Normalization on Deep Neural Network for Time Series Forecasting” também foram propostas diversas técnicas de normalização de dados para aumentar a qualidade dos dados de entrada, alguns deles como o min-max normalization, técnica que consiste em normalizar os dados de entrada através do menor valor dado e do maior valor para normalizar a entrada em uma variação que vai de 0 a 1. Também foram apresentados outras técnicas de normalização de dados como decimal scaling normalization, z-score normalization, median normalization, sigmoid normalization e tahn estimators. Ao fazer os testes realizando as normalizações os autores obtiveram um maior desempenho ao utilizar a técnica de normalização tahn estimators mais eficaz para predição de séries temporais de deep recurrent neural network (DRNN).

**6.3 PREVISÃO DE PREÇOS DE AÇÕES, LSTM E CNN**

Mehtab, Sen e Dasgupta (2021) apresentaram um conjunto de modelos de regressão para previsão dos preços futuros das ações de uma empresa bem conhecida listados na National Stock Exchange (NSE) da Índia, e inclui três modelos de regressão que são construídos sobre convolutional neural networks (CNN’s) e quatro redes preditivas, modelos baseados em redes de long short-term memory (LSTM).

No trabalho foram construídos datasets usando preços de ações da Bharat Forge em intervalo de 5 minutos utilizando a ferramenta Metastock, os treinos e o teste de conjunto de dados consistem em 19500 e 20500 registros, e cada registro possuía os seguintes atributos.

* + Data
  + Intervalo de tempo
  + Aberto
  + Alto
  + Baixo
  + Fechado
  + Volume

Então eles construíram sete modelos de regressão baseados em deep learning para a previsão dos preços das ações, três modelos de arquitetura CNN, e quatro modelos de regressão de redes LSTM. Os modelos foram construídos utilizando parâmetros otimizados e testados em dados extremamente granulares de preços de ações coletados em um intervalo de cinco minutos. Os resultados experimentais mostraram que, embora os modelos exigissem grande divergência em sua precisão e velocidade de execução, todos eles produziram um nível muito alto de precisão em seus resultados. Em geral, os modelos baseados em CNN superaram seus equivalentes LSTM em termos de velocidade de execução e precisão na previsão.

**6.4 APRENDIZADO PARA DADOS DO MERCADO POR PEDIDO**

Os autores Zhang, Lim e Zohren (2021) desenvolveram suas pesquisas baseadas em dados de Market by Order (MBO) para modelagem preditiva com algoritmos de aprendizado profundo. Os dados MBO fornecem resolução total da microestrutura de mercado com dados de sequências de negociação sendo derivadas a partir dele. Além disso, utilizaram uma outra fonte preditiva de alta frequência que concentra-se predominantemente em limit order books (LOB).

Dados MBO são essencialmente um feed de dados baseado em mensagens que nos permite inferir a fila individual de posição para cada pedido individual, reconstruindo o livro de pedidos passo a passo.

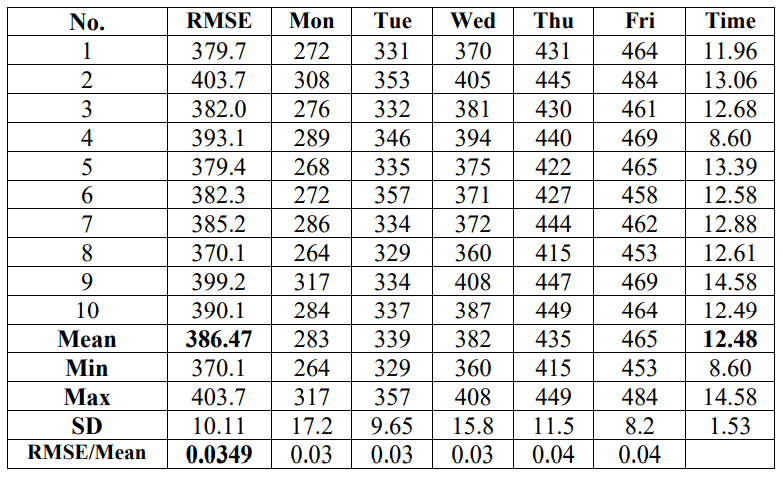
Um LOB é um registro de todas as ordens limitadas pendentes (ordens passivas) para um instrumento em um determinado momento e é classificado em diferentes níveis com base nos preços apresentados.

Comparando os modelos dos autores, utilizados dados MBO e LOB, respectivamente, verifica-se que é possível obter desempenho semelhante, mas ligeiramente inferior, modelando mensagens MBO brutas, quando comparado com a modelagem de dados LOB. Embora os dados MBO contenham mais informações, é mais difícil modelar as mensagens brutas em vez de LOBs, nota-se que os modelos de aprendizado profundo superam o modelo linear simples, levando a conclusão que a existência de características não lineares em séries temporais financeiras seja um ótimo recurso para aumentar a qualidade preditiva de modelos de redes neurais.

**6.5 PREVISÃO DE PREÇO DE AÇÕES USANDO CNN E LSTM**

Mehtab e Sen (2020) exploraram o poder de arquiteturas de redes neurais como a convolutional neural networks (CNN’s), como também foi utilizado outro tipo de modelo de aprendizado profundo de redes de long short-term memory (LSTM) na previsão de uma série temporal multivariada complexa como a série NIFTY 50. No artigo “Stock Price Prediction Using CNN and LSTM-Based Deep Learning Models”, foram apresentados cinco modelos preditivos diferentes. Os modelos são diferentes em suas arquiteturas e suas formas de dados de entrada também são diferentes. Os cinco modelos são: (1) modelo CNN com dados de entrada univariados da última semana, (2) modelo CNN com dados de entrada univariados das últimas duas semanas, (3) codificador-decodificador LSTM com dados univariados das últimas duas semanas, (4) Modelo LSTM codificador-decodificador CNN com dados de entrada univariados das últimas duas semanas, e (5) LSTM convolucional codificador-decodificador com dados de entrada univariados das últimas duas semanas.

Abaixo pode-se observar as tabelas com as médias dos resultados obtidos durante os testes.

  
Tabela 1 - CNN univariado com dados de uma semana anterior.

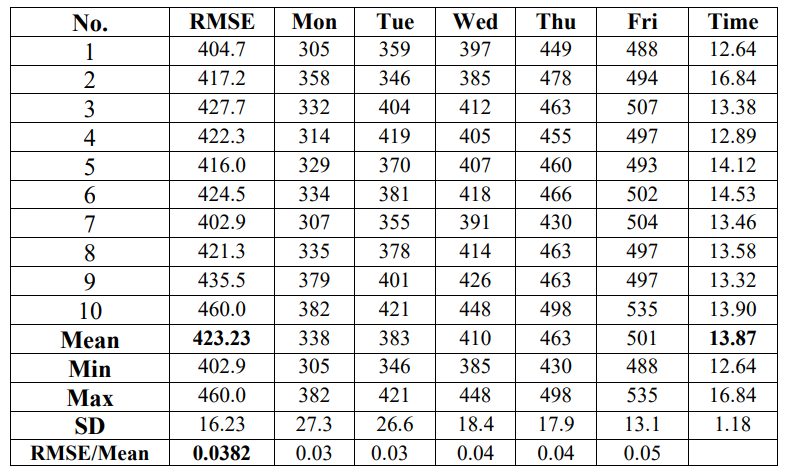


Tabela 2 - CNN univariado com dados de duas semanas anteriores.

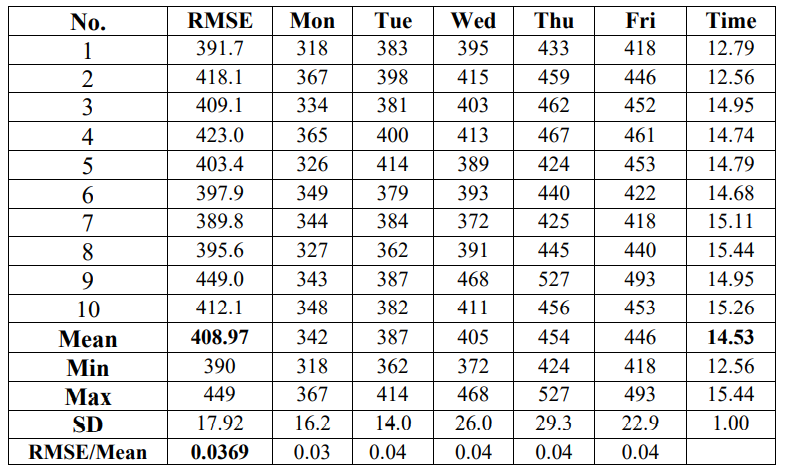


Tabela 3 - LSTM univariado com dados de duas semanas anteriores.

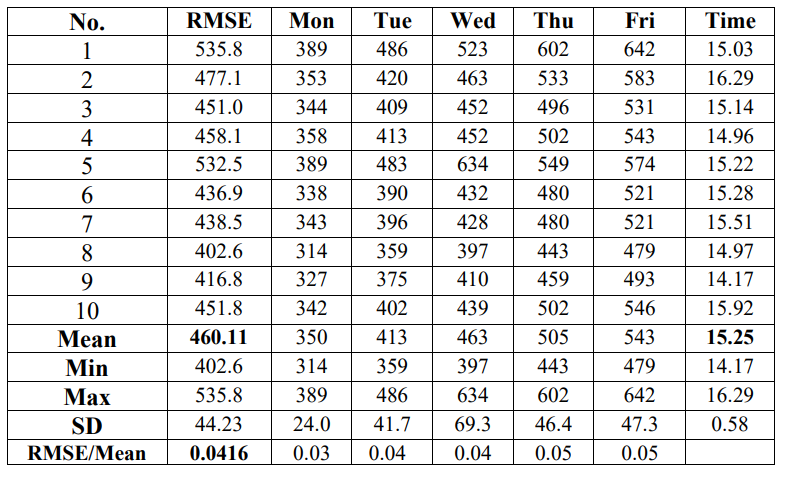


Tabela 4 - LSTM com CNN univariado com dados de duas semanas anteriores.

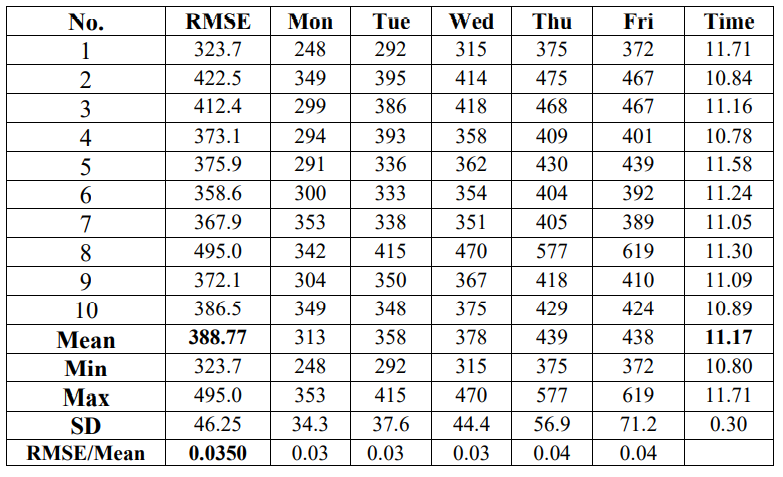


Tabela 5 – LSTM convolucional univariado com dados de duas semanas anteriores.

Os modelos foram construídos, otimizados e testados nos valores de índice diários do NIFTY 50. Enquanto todos os modelos exibiram altos níveis de precisão em suas previsões de desempenho, o LSTM convolucional codificador-decodificador univariado com dados de duas semanas anteriores como entrada, foi considerado o modelo mais preciso. No entanto, em termos de velocidade de execução, o modelo CNN univariado com dados de uma semana anterior como entrada foi considerado o mais rápido.

1. **AMBIENTE DE DESENVOLVIMENTO E EXPERIMENTAÇÃO**

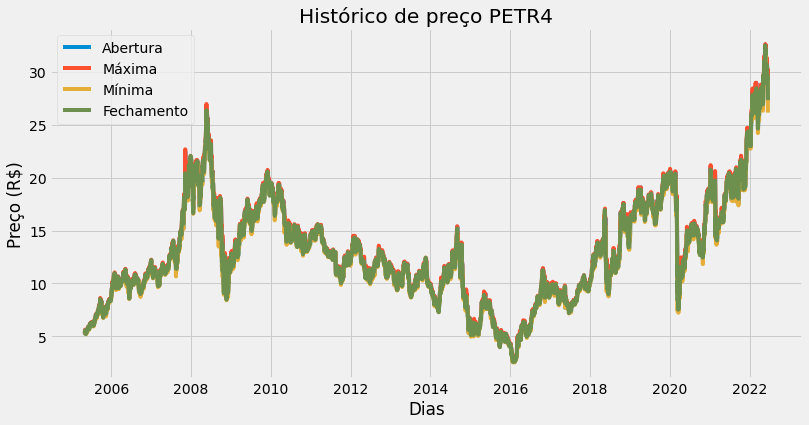
Foi adotada uma abordagem impirica para definição de hiperparâmetros como: dropout, quantidade de neurônios e épocas. Testes foram feitos, onde foram encontrados os respectivos valores para os parâmetros utilizados como por exemplo a quantidade de neurônios a utilizar em cada camada da rede neural de cada modelo, como também o respectivo valor de dropout de 20%.

O ambiente de programação escolhido para realizar os experimentos e testes obtidos até o momento foi o Colab do Google. Ele possibilita a programação com a linguagem python através de blocos de código, nele também existem algumas das bibliotecas que são utilizadas para testes e treinamentos com redes neurais, apenas com a necessidade de realizar a importação da biblioteca que será utilizada. Durante o treinamento será utilizado o método de aprendizado supervisionado.

Como ponto inicial para preparação do ambiente de testes foram realizadas as instalações e devidas importações da biblioteca utilizada como base de dados para trazer os valores financeiros da ação da Petrobras com a API da Yahoo Finance, também foram importadas as bibliotecas “math” para realização de funções matemáticas, a biblioteca “matplotlib” para criação e formulação de gráficos para exibição mais clara dos resultados, a biblioteca “pandas” para manipulação de DataFrames, também foi importada a biblioteca “keras”, que possui diversas funcionalidades e tipos de arquiteturas de redes neurais de forma simples de utilizar, e a “numpy”, que suporta o processamento de arranjos multi-dimensionais e matrizes, como também possui uma grande coleção de funções matemáticas de alto nível para operar sobre as matrizes.

Utilizando o comando “Ticker” passando como parâmetro o código das ações da Petrobras, Magazine Luiza e Ambev, foi possível obter os valores financeiros de cada ação com um filtro do período de 1 de Maio de 2005 até 18 de Junho de 2022, como resultado foi se obtido o Pandas DataFrames com todos os dados necessários para a preparação dos dados e realização do pré-processamento.

Com os dados foi possível gerar um gráfico utilizando a biblioteca “matplotlib” pegando os atributos “Abertura”, “Máxima”, “Mínima” e “Fechamento”, conforme mostra as Figuras 11,12 e 13.

  
Figura 12 - Histórico de preço PETR4.

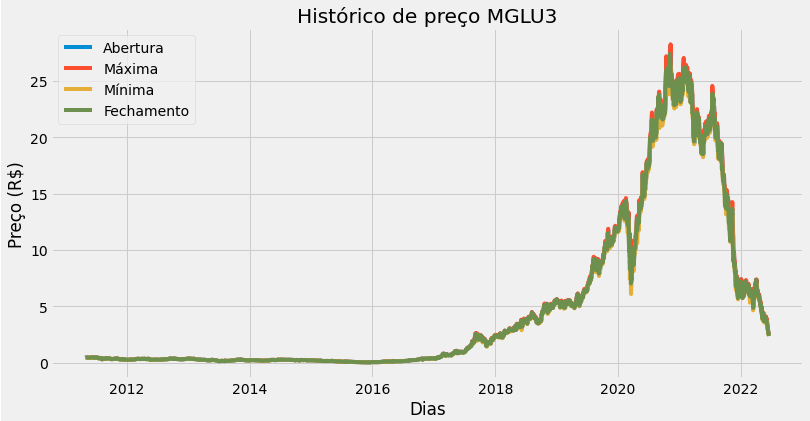


Figura 13 - Histórico de preço MGLU3.

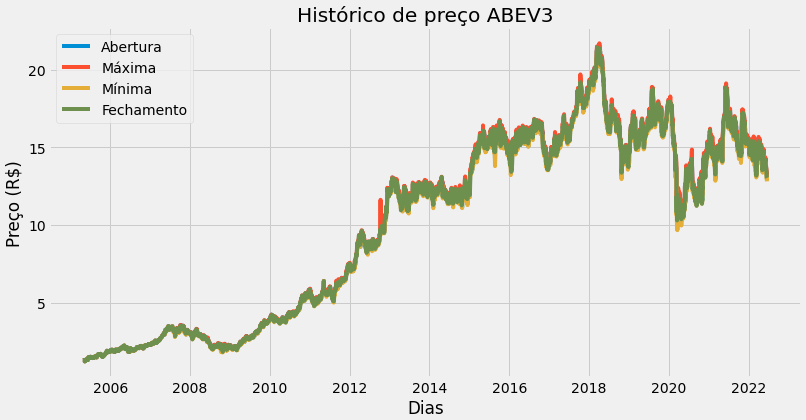


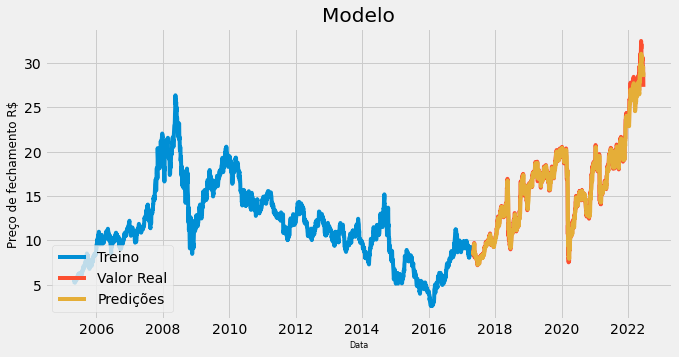
Figura 14 - Histórico de preço ABEV3.

Então foi realizado um novo filtro no dataset com os valores para pegar apenas a coluna “fechamento”. Após isso, o mesmo dataset foi convertido para uma lista numpy para treinar os modelos de LSTM e GRU, então, foi criada uma nova lista com 70% dos valores para a finalidade de realizar o treinamento. Está proporção foi definida de acordo com a literatura. Com esta lista de treinamento foi aplicada uma normalização de escala de mínima e máxima entre 0 e 1, o restante dos registros serão utilizados para validação.

Foram construídos quatro modelos de treinamentos, dois utilizando LSTM e dois utilizando GRU. Dois dos modelos contam com uma arquitetura de 50 neurônios e 4 camadas ocultas, também foi definido camadas de Dropout de 20% que funciona zerando aleatoriamente uma porcentagem de neurônios nas camadas da rede neural. Os dois outros modelos contam com uma arquitetura de 50 neurônios e 6 camadas ocultas, também foi definido camadas de Dropout de 20%. Dropout é uma técnica utilizada na maioria das redes neurais modernas para prevenir sobre-ajustamento (Srivastava *et* al., ‎2014). Para a compilação do modelo foi utilizado o optimizador adam, pois de acordo com (Kingma *et* al., 2014), o método é "computacionalmente eficiente, tem pouca necessidade de memória, invariante ao reescalonamento diagonal de gradientes e é adequado para problemas que são grandes em termos de dados/parâmetros".

O treinamento foi realizado passando a lista dos dos 70% dos valores reservados para o treinamento, lembrado que ela foi passada por completo para a entrada da rede neural utilizando a lista no formato numpy, no momento do treinamento foi definido um total de 100 épocas e com um tamanho de batch de 32, que é o número de amostras processadas antes de atualizar o modelo.

Com todo o processamento finalizado são realizadas as predições utilizando o dataset de teste, a lista com as predições obtidas são comparadas com o valor real através da geração de gráficos, o resultado obtido com os testes e o resultado é exibido na Figura 14.

  
Figura 15 - Resultado obtido modelo LSTM.

Onde “Treino”, em azul, representa a lista com os valores de treinamento, “Valor Real”, em laranja, representa a lista com os dados utilizados para validação e “Predições”, em amarelo, representa os valores obtidos através das predições realizadas pelo modelo.

Todo o procedimento realizado durante o aprendizado supervisionado foi repetido para os 4 modelos de arquitetura de rede neural, como também foram feitos testes com os 4 modelos sem a adição da técnica de Dropout para testar diferentes abordagens, com o intuito de verificar o quão efetivo pode ser utilizar o Dropout durante os testes.

Os valores respectivos resultantes de cada um dos modelos será levado para uma tabela comparativa onde será possível observar e comparar quais modelos terão um resultados mais satisfatórios, com o intuito de saber qual modelo teve um maior desempenho preditivo.

# EXPERIMENTOS

Os quatro modelos criados foram submetidos a testes e a todo o procedimento de aprendizado supervisionado. Os testes foram realizados com as três ações já mencionadas anteriormente, Petrobras, Magazine Luiza e Ambev. Além disso, os quatro modelos foram testados com e sem dropout para validar o quão efetivo pode ser utilizar o dropout, totalizando 24 resultados diferentes.

O resultado de cada treinamento realizado foi passado para uma tabela para que seus valores de RMSE sejam todos comparados, para verificar qual deles tiveram resultados mais baixos de erro.

Todos os valores resultantes de RMSE podem ser observados na Tabela 6.

| **Modelo** | **Ativo** | **RSME - Com dropout** | **RMSE - Sem dropout** |
| --- | --- | --- | --- |
| **LSTM-1** | **PETR4** | 0,05 | 0,1 |
| **LSTM-2** | **PETR4** | 0,36 | 0,36 |
| **GRU-1** | **PETR4** | 0,11 | **0,03** |
| **GRU-2** | **PETR4** | **0,03** | 0,5 |
| **LSTM-1** | **MGLU3** | 2,32 | 1,52 |
| **LSTM-2** | **MGLU3** | 3,4 | 4,05 |
| **GRU-1** | **MGLU3** | 0,55 | **0,27** |
| **GRU-2** | **MGLU3** | 1,75 | 1,82 |
| **LSTM-1** | **ABEV3** | **0,04** | 0,07 |
| **LSTM-2** | **ABEV3** | 0,92 | 0,36 |
| **GRU-1** | **ABEV3** | 0,2 | 0,08 |
| **GRU-2** | **ABEV3** | 0,64 | 0,06 |

Tabela 6 – Valores de RMSE resultantes.

Ao verificar os valores de RMSE obtidos no primeiro modelo LSTM e realizarmos um cálculo de média simples, somando todos os valores obtidos resultantes do RMSE e dividindo todos pelo total de resultados (3), observa-se na Tabela 7 que o resultado foi de 0,80.

| **Modelo** | **Ativo** | **RSME - Com dropout** |
| --- | --- | --- |
| **LSTM-1** | **PETR4** | **0,05** |
| **LSTM-1** | **MGLU3** | **2,32** |
| **LSTM-1** | **ABEV3** | **0,04** |
|  | **Média** | **0,80** |

Tabela 7 – Valores de RMSE resultantes do LSTM-1.

Verificando os valores de RMSE obtidos no segundo modelo LSTM ao realizar um cálculo de média simples, somando todos os valores obtidos resultantes do RMSE e dividindo todos pelo total de resultados (3), observa-se na Tabela 8 que o resultado foi de 1,56.

| **Modelo** | **Ativo** | **RSME - Com dropout** |
| --- | --- | --- |
| **LSTM-2** | **PETR4** | **0,36** |
| **LSTM-2** | **MGLU3** | **3,4** |
| **LSTM-2** | **ABEV3** | **0,92** |
|  | **Média** | **1,56** |

Tabela 8 – Valores de RMSE resultantes do LSTM-2.

Já ao analisar os valores de RMSE obtidos no primeiro modelo GRU e realizarmos um cálculo de média simples, somando todos os valores obtidos resultantes do RMSE e dividindo todos pelo total de resultados (3), observa-se na Tabela 9 que o resultado foi de 0,29.

| **Modelo** | **Ativo** | **RSME - Com dropout** |
| --- | --- | --- |
| **GRU-1** | **PETR4** | **0,11** |
| **GRU-1** | **MGLU3** | **0,55** |
| **GRU-1** | **ABEV3** | **0,2** |
|  | **Média** | **0,29** |

Tabela 9 – Valores de RMSE resultantes do GRU-1.

Ao verificar os valores de RMSE obtidos no segundo modelo GRU e realizarmos um cálculo de média simples, somando todos os valores obtidos resultantes do RMSE e dividindo todos pelo total de resultados (3), observa-se na Tabela 10 que o resultado foi de 0,81.

| **Modelo** | **Ativo** | **RSME - Com dropout** |
| --- | --- | --- |
| **GRU-2** | **PETR4** | **0,03** |
| **GRU-2** | **MGLU3** | **1,75** |
| **GRU-2** | **ABEV3** | **0,64** |
|  | **Média** | **0,81** |

Tabela 10 – Valores de RMSE resultantes do GRU-2.

Verificando os valores de RMSE obtidos no primeiro modelo LSTM sem a adição de dropout depois de cada camada ao realizarmos um cálculo de média simples, somando todos os valores obtidos resultantes do RMSE e dividindo todos pelo total de resultados (3), observa-se na Tabela 11 que o resultado foi de 0,56.

| **Modelo** | **Ativo** | **RMSE - Sem dropout** |
| --- | --- | --- |
| **LSTM-1** | **PETR4** | **0,1** |
| **LSTM-1** | **MGLU3** | **1,52** |
| **LSTM-1** | **ABEV3** | **0,07** |
|  | **Média** | **0,56** |

Tabela 11 – Valores de RMSE resultantes do LSTM-1 sem dropout.

Já ao analisar os valores de RMSE obtidos no segundo modelo LSTM sem a adição de dropout depois de cada camada e realizarmos um cálculo de média simples, somando todos os valores obtidos resultantes do RMSE e dividindo todos pelo total de resultados (3), observa-se na Tabela 12 que o resultado foi de 1,59.

| **Modelo** | **Ativo** | **RMSE - Sem dropout** |
| --- | --- | --- |
| **LSTM-2** | **PETR4** | **0,36** |
| **LSTM-2** | **MGLU3** | **4,05** |
| **LSTM-2** | **ABEV3** | **0,36** |
|  | **Média** | **1,59** |

Tabela 12 – Valores de RMSE resultantes do LSTM-2 sem dropout.

Ao verificar os valores de RMSE obtidos no primeiro modelo GRU sem a adição de dropout depois de cada camada e realizarmos um cálculo de média simples, somando todos os valores obtidos resultantes do RMSE e dividindo todos pelo total de resultados (3), observa-se na Tabela 13 que o resultado foi de 0,13.

| **Modelo** | **Ativo** | **RMSE - Sem dropout** |
| --- | --- | --- |
| **GRU-1** | **PETR4** | **0,03** |
| **GRU-1** | **MGLU3** | **0,27** |
| **GRU-1** | **ABEV3** | **0,08** |
|  | **Média** | **0,13** |

Tabela 13 – Valores de RMSE resultantes do GRU-1 sem dropout.

Verificando os valores de RMSE obtidos no segundo modelo GRU sem a adição de dropout depois de cada camada e realizarmos um cálculo de média simples, somando todos os valores obtidos resultantes do RMSE e dividindo todos pelo total de resultados (3), observa-se na Tabela 14 que o resultado foi de 0,79.

| **Modelo** | **Ativo** | **RMSE - Sem dropout** |
| --- | --- | --- |
| **GRU-2** | **PETR4** | **0,5** |
| **GRU-2** | **MGLU3** | **1,82** |
| **GRU-2** | **ABEV3** | **0,06** |
|  | **Média** | **0,79** |

Tabela 14 – Valores de RMSE resultantes do GRU-2 sem dropout.

Ao realizar uma avaliação geral entre todos os modelos, pode-se verificar que os modelos com menor média é o primeiro GRU, tanto para os modelos com dropout quanto para os modelos sem dropout a menor média foi a do primeiro modelo de GRU, tendo um desempenho superior aos demais.

Abaixo é possível observar os valores que foram previstos após a realização de todo o processo de aprendizado supervisionado, em todos os modelos de arquitetura de rede neural e, ao fazer a compilação dos modelos e predição dos mesmos, as predições resultantes foram obtidas e os resultados foram passados para gráficos no intuito de observar as previsões de maneira mais fácil do que analisar os valores previstos que são retornados como uma lista são apresentados na Tabela 15.

| Modelo | Resultado |
| --- | --- |
| LSTM-1 / PETR4 - Com Dropout | C:\Users\Paulo-PC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\LSTM_1_PETR.PNG |
| LSTM-1 / PETR4 - Sem Dropout | C:\Users\Paulo-PC\Desktop\Feevale\TCC\TC2\Resultado_modelos\Sem dropout\LSTM_1_PETR.png |
| LSTM-2 / PETR4 - Com Dropout | C:\Users\Paulo-PC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\LSTM_2_PETR.PNG |
| LSTM-2 / PETR4 - Sem Dropout | C:\Users\Paulo-PC\Desktop\Feevale\TCC\TC2\Resultado_modelos\Sem dropout\LSTM_2_PETR.png |
| GRU-1 / PETR4 - Com Dropout | C:\Users\Paulo-PC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\GRU_1_PETR.PNG |
| GRU-1 / PETR4 - Sem Dropout | C:\Users\Paulo-PC\Desktop\Feevale\TCC\TC2\Resultado_modelos\Sem dropout\GRU_1_PETR.png |
| GRU-2 / PETR4 - Com Dropout | C:\Users\Paulo-PC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\GRU_2_PETR.PNG |
| GRU-2 / PETR4 - Sem Dropout | C:\Users\Paulo-PC\Desktop\Feevale\TCC\TC2\Resultado_modelos\Sem dropout\GRU_2_PETR.png |
| LSTM-1 / MGLU3 - Com Dropout | C:\Users\Paulo-PC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\LSTM_1_MGLU.PNG |
| LSTM-1 / MGLU3 - Sem Dropout | C:\Users\Paulo-PC\Desktop\Feevale\TCC\TC2\Resultado_modelos\Sem dropout\LSTM_1_MGLU.png |
| LSTM-2 / MGLU3 - Com Dropout | C:\Users\Paulo-PC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\LSTM_2_MGLU.PNG |
| LSTM-2 / MGLU3 - Sem Dropout | C:\Users\Paulo-PC\Desktop\Feevale\TCC\TC2\Resultado_modelos\Sem dropout\LSTM_2_MGLU.png |
| GRU-1 / MGLU3 - Com Dropout | C:\Users\Paulo-PC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\GRU_1_MGLU.PNG |
| GRU-1 / MGLU3 - Sem Dropout | C:\Users\Paulo-PC\Desktop\Feevale\TCC\TC2\Resultado_modelos\Sem dropout\GRU_1_MGLU.png |
| GRU-2 / MGLU3 - Com Dropout | C:\Users\Paulo-PC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\GRU_2_MGLU.PNG |
| GRU-2 / MGLU3 - Sem Dropout | C:\Users\Paulo-PC\Desktop\Feevale\TCC\TC2\Resultado_modelos\Sem dropout\GRU_2_MGLU.png |
| LSTM-1 / ABEV3 - Com Dropout | C:\Users\Paulo-PC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\LSTM_1_ABEV.PNG |
| LSTM-1 / ABEV3 - Sem Dropout | C:\Users\Paulo-PC\Desktop\Feevale\TCC\TC2\Resultado_modelos\Sem dropout\LSTM_1_ABEV.png |
| LSTM-2 / ABEV3 - Com Dropout | C:\Users\Paulo-PC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\LSTM_2_ABEV.PNG |
| LSTM-2 / ABEV3 - Sem Dropout | C:\Users\Paulo-PC\Desktop\Feevale\TCC\TC2\Resultado_modelos\Sem dropout\LSTM_2_ABEV.png |
| GRU-1 / ABEV3 - Com Dropout | C:\Users\Paulo-PC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\GRU_1_ABEV.PNG |
| GRU-1 / ABEV3 - Sem Dropout | C:\Users\Paulo-PC\Desktop\Feevale\TCC\TC2\Resultado_modelos\Sem dropout\GRU_1_ABEV.png |
| GRU-2 / ABEV3 - Com Dropout | C:\Users\Paulo-PC\Desktop\Feevale\TCC\TC2\Resultado_modelos\Com dropout\GRU_2_ABEV.png |
| GRU-2 / ABEV3 - Sem Dropout | C:\Users\Paulo-PC\Desktop\Feevale\TCC\TC2\Resultado_modelos\Sem dropout\GRU_2_ABEV.png |

Tabela 15 – Valores obtidos das predições.

Ao fazer uma análise crítica dos modelos e seus resultados, temos como teste de melhor desempenho o primeiro modelo GRU testado como parâmetro a ação da Petrobras, representada pela sigla de ativo na bolsa de valores como PETR4, sem a utilização de dropout, como também o segundo modelo de GRU testado também como parâmetro a ação da Petrobras, com a utilização de dropout. Esse resultado foi possível de se obter graças a utilização do cálculo do RMSE, e como já citado, quanto mais próximo a zero é o resultado do cálculo, melhor é a predição do modelo, e conforme calculado em cada um dos testes de cada modelos os dois modelos tiveram os menores RMSE’s durante os testes.

# CONCLUSÃO

Estudos envolvendo redes neurais estão ganhando espaço na área de pesquisa da ciência, isso pelo fato de que as redes neurais funcionam semelhante ao cérebro humano, porém aprendem de maneira muito mais rápida, fazendo com que elas se tornem uma ótima opção para realizar previsões.

Conforme mencionado no presente trabalho, o objetivo do mesmo é criar um método auxiliar na tomada de decisão, ao realizar uma operação financeira de ativos da bolsa de valores, para assim, ajudar pessoas a investir melhor. Para que isso seja possível, testes, treinamentos, estudos e práticas foram realizadas no presente trabalho para desenvolver modelos de rede neural recorrente para tal análise e tentativa de previsão dos valores. Com os resultados dos estudos, percebe-se que diversas análises e previsões podem ser feitas utilizando redes neurais, como também percebe-se que redes neurais recorrentes se tornam uma melhor abordagem para previsões de sequências temporais, como ativos da bolsa de valores e seus respectivos preços de fechamento, por tempo.

O processo de aprendizagem foi simples graças a referências, onde dentre diversas classes de aprendizagem, foi utilizado o processo de aprendizagem supervisionado, isso pelo fato de que o procedimento de treinamento dos modelos se tornaram mais rápidos, isso porque o aprendizado supervisionado, o algoritmo aprende com o conjunto de dados de treinamento fazendo previsões interativas sobre os dados e ajustando-se para a resposta correta, e os modelos de aprendizado não supervisionados, por outro lado, trabalham por conta própria para descobrir a estrutura inerente de dados não rotulados, facilitando assim a organização dos passos que são tomados durante a etapa do processo de aprendizagem.

O estudo do trabalho de pesquisa foi bem sucedido, o mesmo envolveu a aprendizagem de um conjunto de métodos e algoritmos de aprendizado profundo e redes neurais para prever preço de ações. Como ativos de referência, este trabalho utilizou as ações da Petrobras, Magazine Luiza e Ambev. Conforme foi apresentado durante o trabalho, muito esforço foi necessário para a apresentação de quatro modelos diferentes que gerassem previsões utilizando-se as arquiteturas de Long Short-Term Memory (LSTM) e Gated Recurrent Unit (GRU), e ambas tiveram ótimos desempenhos preditivos, tanto com modelos utilizando dropout quanto dos que não utilizaram a técnica, além disso, realizando uma avaliação geral entre todos os modelos, foi possível identificar que os modelos com menor média de raiz do erro quadrático médio (RMSE), foi o primeiro modelo GRU, resultando em uma média geral de “0,29” com dropout, e “0,13” sem dropout, ao somar a média obtida nos testes com as 3 ações, como também o primeiro modelo LSTM, que teve uma média resultante de “0,80” com dropout, e “0,56” sem a utilização da técnica de dropout.

Foi possível observar que resultados significativos são possíveis de se obter através de redes neurais, e que através de pesquisas é possível prever a direção de preços de determinados ativos da bolsa de valor com baixo erro, com o estudo foi possível perceber que métodos alternativos podem auxiliar na tomada de decisão, depois de uma certa análise e processamento de dados, e tais previsões podem ser realizadas utilizando redes neurais recorrentes.

**9.1 TRABALHOS FUTUROS**

Para o futuro trabalho pretende-se fazer a componentização dos modelos, passando uma ação como parâmetro para prever os valores financeiros daquele determinado ativo, como também a criação de micro serviços para disponibilizar uma ferramenta de predição de fácil acesso e oferecer de forma gratuita as predições realizadas pelos modelos. Como também pretende-se criar uma plataforma que utilize os micro serviços criados a partir dos modelos para uma visualização simples e de fácil acesso para pessoas que não tenham conhecimento com o mercado financeiro tenham uma forma de analisar as ações e terem uma ideia de qual decisão tomar diante das dificuldades que é investir na bolsa de valores.

# 10. REFERÊNCIAS

White H. Economic prediction using neural networks: The case of IBM daily stock returns. Proceedings of the IEEE International Conference on Neural

Networks, 1988.

Elton E.; Gruber, M.; Brown, S.; Goetzmann, W. Modern Portfolio Theory and 55 Investment Analysis, 7 ed. John Wiley & Sons, Inc., 2007.

Fama E. F. Efficient capital markets II. Journal of Finance, v. 46, n. 5, dez. 1991.

Fama E. F. Efficient capital markets: A review of theory and empirical work. The Journal of Finance, American Finance Association, v. 25, n. 2, maio. 1970. ISSN 0022- 1082.

Cootner P. H. The random character of stock market prices. MIT Press. 1964. ISBN 9780262030090.

Neural Networks What they are & why they matter, SAS Institute Inc., disponível em https://www.sas.com/th\_th/insights/analytics/neural-networks.html, acessado 02 de Abril de 2022.

Guia Completo Inteligência Artificial: o que é, conceito e métodos de IA., Igor Pimenta, disponível em https://www.take.net/blog/tecnologia/inteligencia-artificial/, acessado 03 de Abril de 2022.

Aprendizado profundo x Aprendizado de máquina em Azure Machine Learning versus Machine Learning, Microsoft, disponível em https://docs.microsoft.com/pt-br/azure/machine-learning/concept-deep-learning-vs-machine-learning, acessado 03 de 2022.

Bhanja S., Das, A. Impact of Data Normalization on Deep Neural Network for Time Series Forecasting, IEEE, Dezembro. 2018.

Grawal S., Stock Prices Prediction using Keras LSTM Model. disponível em https://github.com/shimonyagrawal/Stock-Prices-Prediction-using-Keras-LSTM-Model/blob/main/Stock%20Price%20Prediction.ipynb, acessado 25 de Maio de 2022.

Jones T., Um mergulho profundo nas redes neurais recorrentes, disponível em https://imasters.com.br/data/um-mergulho-profundo-nas-redes-neurais-recorrentes, acessado 18 de Junho de 2022.

Penteado K., Métricas de avaliação para séries temporais, disponível em https://lamfo-unb.github.io/2017/10/22/Monte-Carlo-Dropout/, acessado em 19 de Junho de 2022.

Keras, Adam, disponível em https://keras.io/api/optimizers/adam/, acessado em 19 de Junho de 2022.

O que é ibovespa? Entenda como funciona o principal índice da bolsa, Clear, disponível em https://master.clear.com.br/o-que-e-ibovespa/, acessado em 29 de Agosto de 2022.

Santos T. G.,Google Colab: o que é, tutorial de como usar e criar códigos, disponível em https://www.alura.com.br/artigos/google-colab-o-que-e-e-como-usar, acessado em 30 de Agosto de 2022.

Gated Recurrent Unit, DeepAI, disponível em https://deepai.org/machine-learning-glossary-and-terms/gated-recurrent-unit, acessado em 30 de Agosto de 2022.

Root Mean Square Error (RMSE), c3.ai , disponível em https://c3.ai/glossary/data-science/root-mean-square-error-rmse/, acessado em 01 de Setembro de 2022.

F. Jim, Mean Squared Error (MSE), disponível em https://statisticsbyjim.com/regression/mean-squared-error-mse/, acessado em 03 de Setembro de 2022.

B. Alexa, What is Mean Squared Error? , disponível em https://study.com/learn/lesson/mean-squared-error-formula.html, acessado em 03 de Setembro de 2022.

G. Satish, What is Root Mean Square Error (RMSE), disponível em https://www.kaggle.com/general/215997, acessado em 03 de Setembro de 2022.

Edivaldo, Saiba o que é o Microsoft Excel, disponível em https://www.tudoexcel.com.br/planilhas/saiba-o-que-e-o-microsoft-excel-2918.html, acessado em 05 de Setembro de 2022.

T. Jokura, Qual é a capacidade do cérebro humano em bytes? , disponível em https://www.uol.com.br/tilt/colunas/pergunta-pro-jokura/2019/12/02/qual-e-a-capacidade-do-cerebro-humano-em-bytes.htm, acessado em 17 de Setembro de 2022.

A. Ponce, Redes Neurais Artificiais, disponível em https://sites.icmc.usp.br/andre/research/neural/, acessado em 25 de Setembro de 2022.

S. Eduardo, Arquiteturas e Topologias de Redes Neurais Artificiais, disponível em https://embarcados.com.br/redes-neurais-artificiais/, acesado em 25 de Setembro de 2022.

B. Marr, What Are Artificial Neural Networks - A Super-Simple Explanation For Anyone, disponível em https://finance.yahoo.com/news/artificial-neural-networks-super-simple-071406961.html, acessado em 26 de Setembro de 2022.  
  
W. Amaro, Utilização de Redes Neurais Recorrentes na Caracterização de Cargas Não Lineares em Sistemas Elétricos, disponível em https://www.feis.unesp.br/Home/departamentos/engenhariaeletrica/pos-graduacao/300-dissertacao-william\_mantovani.pdf, acessado em 26 de Setembro de 2022.

Underfitting e Overfitting, Didática Tech, disponível em https://didatica.tech/underfitting-e-overfitting/, acessado em 28 de Setembro de 2022.

P. Vasconcellos, Como saber se seu modelo de Machine Learning está funcionando mesmo, disponível em https://paulovasconcellos.com.br/como-saber-se-seu-modelo-de-machine-learning-est%C3%A1-funcionando-mesmo-a5892f6468b, acessado em 17 de Outubro de 2022.

Data Science Academy, Deep Learning Book, disponível em https://www.deeplearningbook.com.br/arquitetura-de-redes-neurais-gated-recurrent-unit-gru/, acessado em 18 de Outubro de 2022.

Didática Tech, Dados de Treino e Teste, disponível em https://didatica.tech/dados-de-treino-e-teste/, acessado em 23 de Outubro de 2022.

Srivastava, Dropout: a simple way to prevent neural networks from

overfitting. The journal of machine learning research, JMLR. org, 2014.

A. Gupta, Mean Squared Error : Overview, Examples, Concepts and More, disponível em https://www.simplilearn.com/tutorials/statistics-tutorial/mean-squared-error, acessado em 25 de Outubro de 2022.

D. Ceccon, Fundamentos de ML: funções de custo para problemas de regressão, disponível em https://iaexpert.academy/2019/08/19/fundamentos-de-ml-funcoes-de-custo-para-problemas-de-regressao/, acessado em 27 de Outubro de 2022.

S. Sharma, Activation Functions in Neural Networks, disponível em https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6, acessado em 16 de Novembro de 2022.