UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Modelo de aprendizado de máquina para prever tendências no Índice Bovespa com a influência de notícias relacionadas

Gustavo Cousseau

Monografia - MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

Gustavo Cousseau

Modelo de aprendizado de máquina para prever tendências no Índice Bovespa com a influência de notícias relacionadas

Monografia apresentada ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Ivan José dos Reis Filho

Versão original

São Carlos

2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Cousseau, Gustavo
C868m Modelo de aprendizado de máquina para prever tendências no Índice Bovespa com a influência de notícias relacionadas / Gustavo Cousseau; orientador Ivan José dos Reis Filho. -- São Carlos, 2024.
49 p.

Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. . I. Reis Filho, Ivan José dos, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Gustavo Cousseau

Machine learning model to predict trends in the Bovespa Index with the influence of related news

Monograph presented to the Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, as part of the requirements for obtaining the title of Specialist in Artificial Intelligence and Big Data.

Concentration area: Artificial Intelligence

Advisor: Prof. Dr. Ivan José dos Reis Filho

Original version

São Carlos

2024

RESUMO

Cousseau, G. **Modelo de aprendizado de máquina para prever tendências no Índice Bovespa com a influência de notícias relacionadas**. 2024. 47p. Monografia (MBA em Inteligência Artificial e Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

Este trabalho tem como objetivo apresentar um modelo de previsão da variação percentual do índice Bovespa, um dos principais indicadores da Bolsa de Valores do Brasil, utilizando técnicas de aprendizado de máquina. A crescente busca por renda extra foi a motivação e justificativa para a realização deste trabalho, levando em consideração as análises preditivas do mercado financeiro, o qual pode fornecer informações importantes para investidores e analistas. Diante disso, foi implementado um modelo de Rede Neural Recorrente chama *Long Short Term Memory* (LSTM), conhecido pela sua eficácia em séries temporais. O modelo foi alimentado com as informações de mais importantes do índice Bovespa em relação a semana junto as informações de notícias relevantes para a variação do índice. O modelo tem como saída prever a variação como negativa, neutra ou positiva. Os resultados obtidos mostraram um valor de acurácia de 54% e outras métricas de avaliação destacaram a necessidade de melhorias para a identificação da classe negativa no modelo, o qual não classificou nenhum registro como negativo. Além disso, o estudo mostrou a relevância de utilizar redes neurais para previsões no mercado financeiro, abrindo possibilidades de mais estudos na área que explorem abordagens de outros modelos e outros parâmetros.

Palavras-chave: Bovespa, LSTM, aprendizado de máquina.

ABSTRACT

Cousseau, G. **Machine learning model to predict trends in the Bovespa Index with the influence of related news**. 2024. 47p. Monograph (MBA in Artificial Intelligence and Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

This work aims to present a model for predicting the percentage variation of the Bovespa index, one of the main indicators of the Brazilian Stock Exchange, using machine learning techniques. The growing search for extra income was the motivation and justification for carrying out this work, taking into account the predictive analyzes of the financial market, which can provide important information for investors and analysts. In view of this, a Recurrent Neural Network model called *Long Short Term Memory* (LSTM) was implemented, known for its effectiveness in time series. The model was fed with the most important information of the Bovespa index in relation to the week along with information from relevant news for the variation of the index. The model aims to predict the variation as negative, neutral or positive. The results obtained showed an accuracy value of 54% and other evaluation metrics highlighted the need for improvements to identify the negative class in the model, which did not classify any record as negative. Furthermore, the study showed the relevance of using neural networks for predictions in the financial market, opening up possibilities for further studies in the area that explore approaches of other models and other parameters.

Keywords: Bovespa, LSTM, machine learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MIMO	Multi-Input Multi-Output
LSTM	Long Short-Term Memory
RoBERTa	Robustly optimized
DIRMO	Direta Multi-Input Multi-Output
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
TF-IDF	Term Frequency - Inverse Document Frequency
IDF	Inverse Document Frequency
RMSE	Root Mean Square Error

SUMÁRIO

1 INTRODUÇÃO

Em muitas empresas há um forte interesse pelo total controle da qualidade. As políticas de gestão de qualidade na maioria das empresas evoluem continuamente, concentrando-se em questões de qualidade que podem ser críticas. Uma vez que a qualidade é amplamente reconhecida como um dos fatores-chave para o sucesso no mercado global (SAVINO; BRUN; XIANG, 2017). Com isso, alguns indicadores começaram a ser introduzidos para monitorar o resultado de ações de melhoria e estão sendo vistos como uma ferramenta para promover a melhoria contínua (FORTUIN, 1988).

Um indicador de desempenho pode ser definido como um item de informação coletado em intervalos regulares para monitorar o desempenho de um sistema. Esses indicadores são coletados em muitos sistemas complexos, como um sistema de educação. Porém, não são perfeitos, sem erros ou problemas de definição e interpretação, mas são indicadores importantes para o funcionamento do sistema e acompanhá-los é um aspecto do controle de qualidade (FITZ-GIBBON, 1990).

Para ter uma boa qualidade, se torna importante coletar qualquer tipo de dado ao longo de cada projeto para ajudar, no final, a entender o sucesso que foi o projeto. Com isso, os gerentes e líderes de equipe se obrigam a estarem cientes de tudo o que está acontecendo, seja bom ou ruim, e se for ruim encontrar uma solução para consertar e garantir que tudo funcione bem. É nesse momento que, acompanhar os indicadores de desempenho corretos ao longo do projeto ajudam a garantir que o projeto seja um sucesso (PHAKOE, 2023).

Ainda, Varisco *et al.* (2018) define que um indicador de desempenho é um conjunto de medidas com foco em atividades críticas. Com a ajuda desses indicadores, as empresas podem comprovar que existe uma lacuna entre o desempenho real e o desejado. Nesse sentido, os indicadores são usados para avaliar a eficiência e eficácia das ações no processo produtivo, parte dos processos ou também todo o sistema produtivo.

Além disso, alguns indicadores indicam o comportamento do mercado de ações, estando relacionados ao nível de riscos associados ao desempenho e geração de valores das empresas. Como exemplo, a IBOVESPA (Bolsa de Valores do Estado de São Paulo) tem a finalidade de instruir investidores e indivíduos em geral sobre o comportamento do mercado de ações. Não obstante, essa bolsa se tornou tão relevante que indica as expectativas de investidores em relação a economia (MONZONI; BIDERMAN; BRITO, 2006).

Sendo o indicador de desempenho médio mais importante do mercado acionista brasileiro (BRONDANI *et al.*, 2013), a IBOVESPA é composta por 87 ações sendo que a ação com maior participação no índice é a VALE de acordo com o site da B3.

Com base nessas informações, qualquer investidor é capaz de adquirir ações de empresas que compõem a IBOVESPA, tornando-se um acionista dessa empresa e tendo direitos em votos em reuniões. Essas ações são títulos representando uma fração do capital de uma empresa. Porém, o mercado de ações pode sofrer alterações de acordo com especialistas do mercado financeiro. Essas variações podem ser afetadas por crises político-econômicas e informações relatadas por mídias, essas informações podem ser bem recebidas ou não pelo mercado (JESUS, 2011).

1.1 Motivação e Lacunas

Devido a essas variações nos preços das ações, muitos estudos foram feitos para tentar prever o comportamento do índice BOVESPA, porém não é uma tarefa simples. O mercado financeiro é caracterizado por incertezas associadas à expectativa de curto, médio e longo prazo. A construção de modelos capazes de captar a dinâmica do mercado, visando reduzir as incertezas, vem sendo o objetivo de pesquisas e têm atraído o interesse de pesquisas acadêmicas e profissionais de mercado (OLIVEIRA; NOBRE; ZÁRATE, 2013).

No entanto, o maior desafio em prever os preços das ações é a natureza complexa do mercado. A maioria dos modelos de previsão são bastante limitados quando se trata de determinar a direção dos preços futuros em virtude da dificuldade inerente em fazer previsões precisas, especialmente quando ocorre grandes flutuações. Uma vez que não é suficiente ter baixas medidas quantitativas formais de precisão, o modelo precisa ter uma alta taxa de precisão para mudanças em direção, pois os movimentos ascendentes e descendentes são os que realmente importa (OLIVEIRA; NOBRE; ZÁRATE, 2013).

1.2 Objetivo Geral

Considerando as informações apresentadas, o presente trabalho tem como principal objetivo construir um modelo de aprendizado de máquina utilizando os conceitos de séries temporais baseando-se nas informações passadas do índice BOVESPA. Complementando o modelo, serão utilizadas as notícias que impactaram nas mudanças ascendentes e descendentes do índice como um atributo adicional. Essas notícias serão classificadas como positiva, negativa e neutra de acordo com a mudança no gráfico das ações.

1.2.1 Objetivos Específicos

Para alcançar o objetivo geral será necessário realizar os objetivos específicos a seguir:

- Coletar uma base de dados que representa os valores do índice BOVESPA desde o início de sua aplicação;

- Coletar notícias que podem influenciar na variação do índice BOVESPA;
- Classificar as notícias de acordo com a influência no gráfico do índice e também, sendo a influência positiva, negativa ou neutra;

Com o modelo e a aplicação das informações de notícias, espera-se que o resultado seja superior nos acertos de queda ou alta em relação aos erros. Possibilitando, assim, que com o modelo, seja possível um retorno lucrativo nos investimentos de ações.

1.3 Organização do texto

No próximo capítulo serão apresentados alguns conceitos em relação a teoria de aprendizado de máquina, e mostrar como aplicar séries temporais para predizer gráficos. Para o capítulo 3, serão abordados alguns trabalhos relacionados utilizando séries temporais e notícias que podem influenciar nos preços das ações do mercado. No capítulo 4, será feita uma explicação das bases de dados utilizadas para o modelo, e como foram obtidas. Para o capítulo 5 serão apresentados os resultados obtidos em relação ao modelo e compara-los com os trabalhos relacionados do capítulo 3. Por último, uma conclusão acerca dos resultados obtidos.

2 FUNDAMENTAÇÃO TEÓRICA

Esse capítulo busca estabelecer os fundamentos teóricos necessários para o entendimento dos métodos e técnicas nesta pesquisa. Inicialmente, serão abordados os conceitos e técnicas de Mineração de Texto, destacando-se as diferentes formas de representação e técnicas para otimizar essa representação. Posteriormente, serão discutidas as representação independentes e dependentes de contexto, com ênfase em algoritmos como *word2vec*, BERT, ROBERT e *transformers*. Em seguida, serão explirados os aspectos das Séries Temporais, contemplando análises determinísticas e estocásticas, sazonalidade, tendência, bem como estratégias de previsão, incluindo *cross-validation for time series*, *hold-out* e previsão de etapas à frente. Adiante, serão apresentadas as estratégias de fusão, como *early-fusion*, *late-fusion* e *joint-fusion*. Além disso, serão discutidos os principais modelos de previsão utilizados como LSTM, GRU, RNN e SVR. Em seguida, serão abordadas as métricas de avaliação, como Acurácia, F1 e Macro. Por fim, será feita uma breve descrição dos trabalhos relacionados, contextualizando-os no escopo desta pesquisa

2.1 Mineração de texto

Segundo Kao and Poteet (2007), a mineração de texto é a descoberta e extração de conhecimento interessante e não trivial de texto livre ou não estruturado. Isso abrange tudo, desde a recuperação de informações até a classificação e agrupamento de textos, até a extração de entidades, relações e eventos.

Ainda, a mineração de texto lida com a análise de texto com suporte para máquinas, utilizando técnicas de recuperação de informação, extração de informação, bem como Processamento de Linguagem Natural (PLN). Essa mineração se conecta com algoritmos de aprendizado de máquina, estatísticas e mineração de dados (HOTHO; NÜRNBERGER; PAASS, 2005).

O processo de mineração de textos é o mesmo que a mineração de dados, exceto pelo fato de que as ferramentas de mineração de dados lidam com dados estruturados. Os textos ou dados não estruturados geralmente se referem a informações que não estão em um banco de dados de linhas e colunas tradicional. Já os dados estruturados estão em campos fixos dentro de um registro ou arquivo, contidos em banco de dados relacionais e planilhas. Sendo assim, a mineração de textos tenta resolver problemas que ocorrem nas áreas de mineração de dados, aprendizado de máquinas, extração de informação, PLN dentre outras (VIJAYARANI *et al.*, 2015).

Esses dados não estruturados podem vir de *e-mails*, arquivos *HTML* e documentos de textos através da recuperação de informação. E para o processo de mineração de textos,

faz-se necessário a aplicação de um pré-processamento para a extração de informação. O pré-processamento é o primeiro passo da mineração de textos e pode ser dividido em quatro partes: extração, remoção de palavras comuns, derivação e representação (VIJAYARANI *et al.*, 2015).

O método de extração é utilizado para *tokenizar* cada palavra do texto. A remoção de palavras comuns, também conhecida como remoção de *stopwords*, é a remoção de algumas palavras que ocorrem com muita frequência e com a mesma frequência em textos relevantes e textos não relevantes para a pesquisa (WILBUR; SIROTKIN, 1992).

A terceira parte consiste em identificar a raiz de cada palavra, ou seja, encontrar uma palavra comum entre as diversas variantes da mesma palavra. Como exemplo, as palavras “apresentação“, “apresentando“ e “apresentado“ podem ser substituídas por uma só palavra, a palavra “apresentar“ (ORENGO; HUYCK, 2001).

Na última parte, o objetivo é representar o texto de alguma forma estruturada, possibilitando a extração de padrões através de algoritmos de aprendizado de máquina. Essa estrutura assume que os dados sejam representados na forma de uma matriz atributo-valor utilizando diferentes estratégias para a seleção dos atributos (termos) (). Essas estratégias podem ser divididas em representações vetoriais, independentes de contexto e dependentes de contexto e são apresentadas a seguir.

2.1.1 Representações vetoriais

Nesse tipo de representação, cada texto é representado como um vetor de termos. Esses termos podem ser uma palavra ou uma frase. Se o termo for palavra, cada palavra do texto representa uma dimensão independente em um espaço vetorial de dimensão muito alta. Qualquer texto pode ser representado por um vetor nesse espaço. Se o termo pertence ao texto, o vetor texto recebe um valor não nulo naquela dimensão do termo (SINGHAL *et al.*, 2001).

Nesse sentido, um vetor pode ser representado pela frequência de cada termo, do inglês *Term Frequency* (TF), consistindo do número de vezes que o termo aparece no conjunto de textos. A frequência é um valor entre 0 e N, onde N é o número de *tokens*. Em PLN, a frequência de termo é frequentemente convertida em probabilidade, usando alguns métodos como o Estimador de Máxima Verossimilhança (YAMAMOTO; CHURCH, 2001).

Outro método estatístico, segundo Al-Obaydy *et al.* (2022), é a Frequência de documento inversa de frequência de termo (TF-IDF) sendo a técnica estatística numérica e descritiva mais popular e amplamente utilizada como fator de ponderação. Esse mecanismo de ponderação descreve a importância das palavras com base na sua presença no conteúdo dos documentos. Com isso, o TF-IDF pode ser empregado para diversas tarefas,

como extração de *tokens* de palavras em artigos, cálculo de graus de similaridade entre documentos, determinação de classificação significativa, entre outras.

O termo de frequência pode ser calculado pela seguinte equação:

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d}} \tag{2.1}$$

onde $f_{t,d}$ é o número de termos iguais no mesmo documento, e o denominador na equação representa a quantidade total de todos os termos no documento.

E para calcular o TF-IDF do termo é necessário definir a equação IDF abaixo:

$$idf(t,D) = \log \frac{N}{|d : d \in D \text{ e } t \in d|} \tag{2.2}$$

onde N representa o número total de documentos e o denominador é o número de documentos onde o termo t aparece. Se o termo não aparecer há uma divisão por 0 e pode ser contornada adicionando 1 no numerador e no denominador.

Assim, a equação do TF-IDF pode ser definida pela multiplicação das equações (2.1) e (2.2):

$$TFIDF(t,d) = tf(t,d) * idf(t,D) \tag{2.3}$$

2.1.2 Representações independentes de contexto

As representações com *embeddings* de palavras são uma família de algoritmos que representam *tokens* com vetores densos de tamanho fixo ("*embeddings*"), de modo que palavras semelhantes obtenham representações vetoriais semelhantes. Essa representação é geralmente baseada em palavras vizinhas, ou seja, palavras diferentes com um uso relacionado obterão representações semelhantes, tais como "colher" e "garfo" (OFER; BRANDES; LINIAL, 2021).

O método mencionado foi popularizado com um algoritmo eficiente chamado *word2vec*, é um modelo de espaço vetorial semelhante à decomposição da matriz de coocorrência, capturando as probabilidades de *tokens* ocorrerem próximos uns dos outros no texto (GOLDBERG; LEVY, 2014).

O modelo *word2vec* possui duas arquiteturas: *bag-of-words* e *skip-grams*. Na arquitetura *bag-of-words*, o modelo prevê a palavra atual a partir de uma janela de palavras de contexto adjacentes (enquanto a ordem das palavras de contexto é ignorada). O oposto ocorre com a arquitetura *skip-grams*, o modelo utiliza a palavra atual para prever a janela adjacente contendo palavras de contexto (OFER; BRANDES; LINIAL, 2021).

As duas arquiteturas não levam em consideração a ordem das palavras e e também não consideram o contexto ao redor das palavras. Como exemplo, as frases “homem morde o cachorro“, “cachorro morde o homem“ e “amor morde“ tem a mesma representação vetorial em relação a palavra “morde“, independente da frase. Para contornar isso, o modelo de *embeddings* contextualizados incluem o contexto da palavra e a ordem em que está na frase (OFER; BRANDES; LINIAL, 2021).

Esses modelos são normalmente utilizados em redes neurais. As arquiteturas de aprendizagem profunda são Memória de Longo e Curto Prazo (LSTM), sequência a sequência (*seq2seq*) e atenção. Nos modelos de sequência a sequência, um texto é transformado usando um componente codificador e, em seguida, um decodificador separado usa a representação codificada para resolver alguma tarefa (um tarefa de tradução do inglês para o francês). Já os modelos de atenção usam camadas de atenção que permitem que a rede se concentre em *tokens* específicos no texto (OFER; BRANDES; LINIAL, 2021).

2.1.3 Representações dependentes de contexto

Uma representação dependente de contexto é a arquitetura *Transformer*, essa arquitetura de aprendizado profundo foi desenvolvida pelo Google e baseada no mecanismo de atenção multicabeças, proposto em um artigo de 2017 chamado “Atenção é Tudo o Que Você Precisa“ (*Attention is All You Need*). Essa arquitetura também converte um texto em um representação numérica chamada *tokens*, cada *token* é convertido em um vetor por meio de pesquisa em uma tabela de incorporação de palavras.

Para isso, essa estrutura utiliza um codificador que mapeia uma sequência de entrada de representações de símbolos $(x_1, ..., x_n)$ para uma sequência de representações contínuas $z = (z_1, ..., z_n)$. Dados z, o decodificador gera uma sequência de saída $(y_1, ..., y_n)$ de símbolos, um elemento por vez. A cada passo o modelo é auto-regressivo, consumindo os símbolos gerados anteriormente como entrada adicional ao gerar o próximo.

O *Transformer* segue essa arquitetura (Figura 1) geral usando autoatenção empilhada e camadas totalmente conectadas e pontuais para o codificador e o decodificador. O codificador é composto por uma pilha de $N = 6$ camadas idênticas. Cada camada possui duas subcamadas. A primeira camada é um mecanismo de autoatenção com várias multi cabeças e o segundo é uma rede *feed-forward* simples, totalmente conectada e posicionada. As saídas de cada camada produzem saídas de dimensão $d_{model} = 512$

O decodificador também é composto por $N = 6$ camadas idênticas. Além das duas sub-camadas em cada camada do codificador, o decodificador insere uma terceira sub-camada, que realiza atenção multi cabeças sobre a saída da pilha do codificador. Uma função de atenção pode ser descrita como o mapeamento de uma consulta e um conjunto de pares de valores-chave para uma saída, onde a consulta, as chaves, os valores e a saída são todos vetores. A saída é calculada como uma soma ponderada dos valores, onde o peso

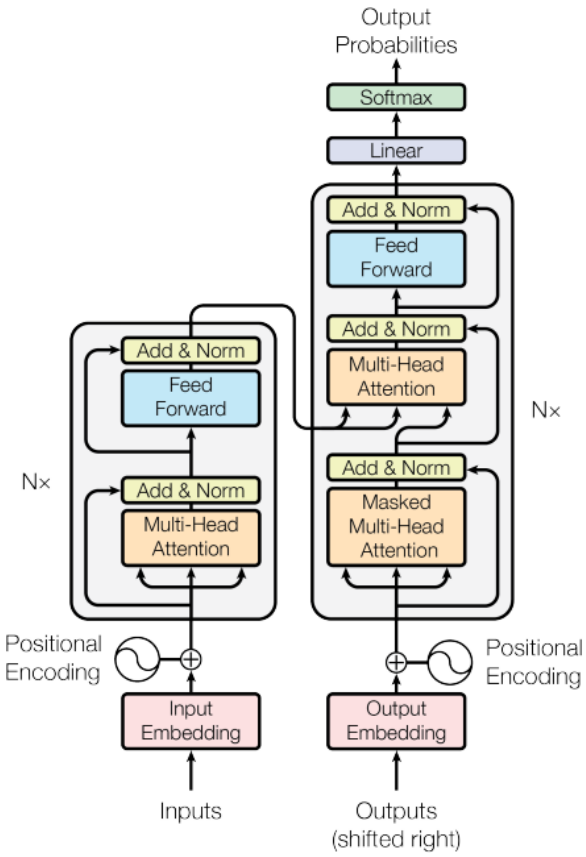

Figura 1 – Modelo de arquitetura do *Transformer*

atribuído a cada valor é calculado por uma função de compatibilidade da consulta com a chave correspondente.

A função de atenção pode ser descrita como o mapeamento de uma consulta e um conjunto de pares de valores-chave para uma saída, onde a consulta, as chaves, os valores e a saída são todos vetores. A saída é calculada como uma soma ponderada dos valores, onde o peso atribuído a cada valor é calculado por uma função de compatibilidade da consulta com a chave correspondente.

A parte da multi cabeças aprende diversas representações de atenção simultaneamente. Na atenção multi cabeças, uma sequência de vetores é dividido em várias partes sendo que cada parte é utilizada como entrada para uma cabeça de atenção. Cada saída da cabeça representa uma matriz de pesos de atenção indicando a importância de cada *token* em relação aos outros *tokens* da sequência. Cada saída é combinada representando uma captura múltipla de atenção.

A saída do modelo é uma distribuição de probabilidade sobre o vocabulário de saída, sendo comparada com a saída esperada usando uma função de perda, como a entropia cruzada. E para maximizar essa perda, o modelo é atualizado com algoritmos de otimização como o gradiente descendente. Por fim, o modelo pode ser pré treinado com

grandes quantidades de dados não supervisionados e depois pode ser ajustado para tarefas de PLN.

Outro modelo dependente de contexto, introduzido por (DEVLIN *et al.*, 2018), é o modelo de linguagem chamado BERT, em português Representação do Codificador Bidirecional de Transformadores. BERT é projetado para pré-treinar representações bidirecionais profundas de texto não rotulado, condicionando conjuntamente o contexto esquerdo e direito em todas as camadas. Como resultado, o modelo BERT pré-treinado pode ser ajustado com apenas uma camada de saída adicional para criar modelos de última geração para uma ampla gama de tarefas, como resposta a perguntas e inferência de linguagem, sem modificações substanciais na arquitetura específica da tarefa.

A arquitetura do modelo BERT é baseada na arquitetura do *Transformer*, seu pré treinamento bidirecional permite que o modelo seja treinado da esquerda para a direita e da direita para a esquerda. Assim é possível capturar o contexto de uma palavra de acordo com o contexto de palavras anteriores como em modelos de linguagem tradicionais, mas também capturar o contexto capturar o contexto com palavras posteriores na sequência.

As camadas de atenção do modelo *Transformer* são empilhadas para formar o codificador do BERT. O pré-treinamento é dividido em duas partes: a primeira parte é o Modelo de Linguagem Mascarada onde algumas palavras são mascaradas aleatoriamente, e o modelo é treinado para prever as palavras mascaradas de acordo com o contexto das palavras não mascaradas; a segunda é a Predição da Próxima Sentença que ajuda o modelo a entender melhor as frases e suas relações nas tarefas que exigem compreensão de longo alcance (tradução automática ou resumo de textos) com a previsão de que se uma frase é a próxima frase em um conjunto de frases fornecidas ou se é uma frase aleatório do documento inteiro.

Devido aos ganhos de desempenho significativo dos modelos *Transformer* e BERT pode ser um desafio determinar quais aspectos contribuem mais. O treinamento é computacionalmente caro, limitando a quantidade de ajuste que pode ser feito, e muitas vezes é feito com dados de treinamento privados de tamanhos variados, limitando a capacidade de medir os efeitos dos avanços na modelagem. Para isso, uma replicação do pré-treinamento do BERT é apresentado por Liu *et al.* (2019) na qual inclui uma avaliação cuidadosa dos efeitos do ajuste do hiper-parâmetro e do tamanho do conjunto de treinamento. Os autores descobriram que o BERT foi significativamente sub-treinado e propuseram uma melhora para o treinamento chamado RoBERTa, que pode igualar ou exceder o desempenho de todos os métodos pós-BERT.

As modificações para melhoria incluem: treinar o modelo por mais tempo, com lotes maiores e mais dados; remover o objeto de previsão da próxima frase; treinamento em sequências mais longas; e alterar dinamicamente o padrão de mascaramento aplicado aos dados de treinamento. Diante dessas modificações, foi concluído que o desempenho

pode ser melhorado substancialmente quando o modelo for treinado por mais tempo, com lotes maiores e com mais dados; quando remover o objetivo de previsão da próxima frase; quando o treinamento em sequências mais longas; e quando alterar dinamicamente o padrão de mascaramento aplicado aos dados de treinamento.

2.2 Séries temporais

Esling and Agon (2012) definem uma série temporal como uma representação de uma coleção de valores obtidos de medições sequenciais ao longo do tempo. Ao modelar séries temporais, é possível estudar, analisar e prever o comportamento dos sistemas. As séries são compostas por dados sequenciais coletados de uma ou mais variáveis ao longo do tempo. Quando composta de uma variável X, representa as observações na forma de $(x_0, x_1, ..., x_{n-1}$ considerando um intervalo de tempo $t \in [0, n-1]$. E quando há mais de um variável, k variáveis são observadas em cada instante de tempo t, sendo descrita por $(x_{1t}, x_{2t}, ..., x_{kt})$ com $t \in [0, n-1]$ (ISHII; RIOS; MELLO, 2011).

Ainda, uma série temporal X_t pode ser formalmente definida pela soma de três componentes não observáveis: $X_t = \Gamma_t + S_t + \epsilon_t$, em que Γ_t representa a tendência, S_t é o sazonalidade e ϵ_t é um componente aleatório. A tendência representa as variações do comportamento da série e a sazonalidade indica se o comportamento da série tende a se repetir nos intervalos de tempo Δ_t.

2.2.1 Análises de séries temporais

As séries temporais podem ser determinísticas ou estocásticas. A determinística apresenta comportamento recorrente, ou seja, ela se repete ao passar do tempo com escalas iguais ou diferentes. Ademais, os sistemas que utilizam séries determinísticas podem ser modelados usando equações diferenciais determinísticas e se torna possível predizer as próximas observações de acordo com as informações anteriores. Por outro lado, as séries estocásticas, onde as observações futuras podem depender das já observadas, assim como nas séries determinísticas, podem também ser gerados por efeitos aleatórios (ISHII; RIOS; MELLO, 2011).

Os períodos das séries temporais que são mais curtos que um ano, como mês, podem variar de acordo com a sazonalidade. Ou seja, é um fenômeno que pode ocorrer com maior frequência em certos períodos do ano e com menos frequência em outros. Nem toda série com períodos mensais ou quadrimestrais são sazonais. Deve-se então, realizar uma análise da série se é ou não sazonal, essa análise serve para obter informações úteis nas tomadas de decisão. As descrições de padrões que mudam conforme o tempo em uma série fornecem uma base necessária para modelos explicativos, previsões e testes de hipóteses de intervenção. Sem uma descrição prévia, os modelos podem ser mal especificados, as previsões imprecisas e os testes de hipóteses errôneos (BLOCK, 1984).

Já a tendência representa uma direção ascendente, descendente ou horizontal ao longo do tempo. É possível capturar um padrão de crescimento e uma previsão do comportamento da série, e também, separar componentes de longo prazo em flutuações aleatórias e sazonais (HAMILTON, 2020).

2.2.2 Estratégias de previsão

Uma maneira de prever o comportamento de uma série temporal é utilizar algoritmos de classificação e uma forma de avaliar esses algoritmos é utilizar o método de validação cruzada *K-Fold*. Essa maneira pode ser mostrada por Bergmeir, Hyndman and Koo (2018), a configuração específica na qual a previsão de séries temporais é geralmente realizada usando métodos de aprendizado de máquina torna possível o uso do *K-Fold*. Ainda, o *K-Fold* é amplamente utilizado para avaliar a generalização de algoritmos em classificação, porém, quando se trata de séries temporais, profissionais muitas vezes não têm certeza sobre a melhor maneira de avaliar seus modelos.

Na avaliação de diferentes parâmetros para os estimadores dos modelos, há um risco de acontecer um *overfitting*, ou seja, todos os dados serem utilizados para o treinamento, fazendo com que a rede acerte todos os dados já vistos e nenhum que esteja fora do conjunto de dados. Uma forma de resolver esse problema é utilizar a validação cruzada com *k-fold*, onde o conjunto de dados é dividido em k conjuntos menores. O modelo é treinado usando $k - 1$ conjuntos menores como dados de treinamento é testado com o restante do conjunto (OJALA; GARRIGA, 2010).

A partir disso, outro conjunto menor é escolhido para o teste do modelo, até que cada conjunto menor seja utilizado. É feita uma média dos valores calculados depois de cada treinamento e teste para encontrar o parâmetro mais adequado. Ao aplicar essas avaliações em séries temporais, o conjunto de dados é dividido em pares de treino-teste sequenciais, a divisão consiste em um conjunto de treino que avança no tempo e conjunto de teste inclui os pontos seguintes.

Outro método utilizado é o método *hold-out* que divide o conjunto em duas partes, normalmente 70% ou 80% para treinamento e o restante para teste. Sendo uma abordagem simples e rápida é considerada menos confiável em conjuntos de dados pequenos, pois a escolha da divisão pode influenciar significativamente os resultados (KOHAVI *et al.*, 1995). Esse método, ao utilizar com séries temporais, é necessário preservar a ordem temporal.

Outra maneira de prever o comportamento da série temporal é utilizar a previsão de passos a frente. Esse tipo de previsão pode ser de apenas um passo a frente como de vários. Quando utilizado um passo a frente, o preditor usa todas ou algumas observações para estimar uma variável de interesse para o intervalo de tempo imediatamente após a última observação (XIONG; BAO; HU, 2013).

Já na previsão de múltiplos passos, a tarefa de previsão consiste em predizer os próximos valores h em $[y_{n+1}, ..., y_{n+h}]$ de uma série temporal histórica $[y_1, ..., y_n]$ composta por n observações, onde $h > 1$ denotando o horizonte de previsão (TAIEB *et al.*, 2012).

Essas previsões podem ser divididas em diferentes estratégias:

2.2.2.1 Estratégia Recursiva

Essa estratégia envolve a utilização de um modelo único f de treinamento para executar a previsão de um passo:

$$y_{t+1} = f(y_t, ..., y_{t-d+1}) + w \tag{2.4}$$

com $t \in \{d, ..., n-1\}$.

Para h passos à frente, é necessário prever o primeiro passo e usa-lo como parte das variáveis de entrada para prever o próximo passo utilizando o mesmo modelo uma passo à frente. É repetido o processo até que todos os dados sejam previstos.

$$y_{n+h} = \begin{cases} f(y_n, ..., y_{n-d+1}) \text{ se } h = 1 \\ f(y_{n+h-1}, ..., y_{n+1}, y_n, ..., y_{n-d+h}) \text{ se } h \in \{2, ..., d\} \\ f(y_{n+h-1}, ..., y_{n+h-d}) \text{ se } h \in \{d+1, ..., h\} \end{cases} \tag{2.5}$$

Uma estratégia simples de implementação e que requer o treinamento de apenas um modelo. Porém, os erros podem ser acumulados ao longo das previsões, resultando em previsões menos precisas para h maiores.

2.2.2.2 Estratégia Direta

Diferente da recursiva, essa estratégia possui um modelo de treinamento para cada passo de previsão.

$$y_{t+h} = f_h(y_t, ..., y_{t-d+1}) + w \tag{2.6}$$

com $t \in \{d, ..., n-h\}$ e $h \in \{1, ..., h\}$.

As previsões são obtidas utilizando os modelos treinados H de acordo com a Equação 2.7:

$$y_{N+h} = f_h(y_N, ..., y_{N-d+1}) \tag{2.7}$$

A vantagem dessa estratégia é que cada modelo é otimizado para seu passo específico da previsão evitando acumular erros, diferente da estratégia recursiva. Mas, requer o treinamento de múltiplos modelos, podendo ser computacionalmente intensivo.

2.2.2.3 Estratégica Direta Recursiva

Combina as arquiteturas e os princípios das estratégias recursivas e diretas. A Direta Recursiva calcula as previsões com modelos diferentes para cada horizonte (estratégia Direta) e, a cada passo de tempo, amplia o conjunto de entradas adicionando variáveis correspondentes às previsões da etapa anterior (estratégia Recursiva). No entanto, ao contrário das duas estratégias anteriores, o tamanho d não é o mesmo para todos os horizontes.

$$y_{t+h} = f_h(y_{t+h-1}, ..., y_{t-d+1}) + w \tag{2.8}$$

onde $t \in \{d, ..., N-H\}$ e $h \in \{1, ..., H\}$.

Para obter as previsões, o modelo treinado H fica de acordo com a Equação 2.9:

$$y_{N+h} = \begin{cases} f_h(y_N, ..., y_{N-d+1}) \text{ se } h = 1 \\ f_h(y_{N+h-1}, ..., y_{N+1}, y_N, ..., y_{N-d+1}) \text{ se } h \in \{2, ..., H\} \end{cases} \tag{2.9}$$

Essa estratégia pode mitigar a amplificação de erros da estratégia recursiva e usa as informações adicionais em cada passo, podendo melhorar a precisão. No entanto, é mais complexa sua implementação e requer o treinamento de múltiplos modelos intermediários.

2.2.2.4 Estratégia MIMO

A estratégia MIMO (Múltiplas Entradas e Múltiplas Saídas), envolve treinar um modelo único para prever múltiplos passos à frente simultaneamente. Essa, aprende um modelo F de múltiplas saídas da série temporal.

$$[y_{t+H}, ..., y_{t+1}] = F(y_t, ..., y_{t-d+1}) + w \tag{2.10}$$

onde $t \in \{d, ..., N-H\}$, $F : \mathcal{R}^d \to \mathcal{R}^H$ é uma função com valor vetorial e $w \in \mathcal{R}^H$ é o ruído do vetor. As previsões são retornadas em uma única etapa por um modelo de múltiplos resultados onde:

$$[y_{t+H}, ..., y_{t+1}] = F(Y_N, ..., y_{N-d+1}) \tag{2.11}$$

Com isso, é capaz de capturar dependências entre diferentes passos de previsão. Treina um único modelo simplificando a implementação. Todavia, requer uma maior capacidade computacional e o ajuste do modelo pode ser mais complexo devido a simultaneidade das previsões.

2.2.2.5 Estratégia DIRMO

Combina a estratégia direta e a MIMO, adotando uma abordagem intermediária, o DIRMO prevê o horizonte H em blocos, onde cada bloco é previsto no estilo MIMO. Ou seja, é treinado um modelo separado para prever os primeiros k passos à frente diretamente e depois é utilizada a abordagem MIMO para treinar modelos subsequentes para prever múltiplos passos além dos primeiros k passos.

A vantagem de combinar os benefícios de otimização específica para certos passos com a capacidade de capturar dependências complexas pode oferecer um equilíbrio entre a complexidade e o desempenho. Pode ser mais complexo de implementar e requer um treinamento de múltiplos modelos, cada um com sua própria configuração.

2.3 Modelos de previsão

Os modelos de previsão dependem dos dados disponíveis. Se os dados são irrelevantes para a previsão, métodos qualitativos devem ser usados. Para o uso de métodos quantitativos, duas condições devem ser satisfeitas: informações numéricas sobre o passado e assumir que alguns aspectos dos padrões do passado podem continuar no futuro (HYNDMAN; ATHANASOPOULOS, 2018). Diante disso, alguns modelos são discutidos abaixo.

2.3.1 Redes Neurais Recorrentes

As RNN fazem parte das redes neurais artificiais que são feitas com camadas de unidades conectadas chamadas neurônios. Quanto maior o número de camadas, maior a complexidade da rede. Um maior número de camadas ou conexões recorrentes geralmente aumenta a profundidade da rede e permite que ela forneça vários níveis de representação de dados e extração de recursos, chamados de aprendizado profundo (SALEHINEJAD *et al.*, 2017).

Essas redes são uma classe de modelos de aprendizado de máquina feitas de neurônios artificiais com um mais ciclos de *feedback*, geralmente são ciclos recorrentes através do tempo. Um modelo simples de RNN contém três camadas, sendo a primeira de entrada, a segunda é a recorrente oculta e a última é a de saída. Assim, a camada de entrada possui N unidades. A entrada para essa camada de entrada é composta de uma sequência de vetores ao decorrer do tempo t como $..., x_{t-1}, x_t, x_{t+1}, ...$, onde $x_t = (x_1, x_2, ..., x_N)$.

Cada unidade de entrada em uma RNN totalmente conectada são conectadas às unidades ocultas na camada oculta, onde as conexões são definidas como uma matriz de pesos W_{IH}. Cada camada oculta tem M unidades escondidas $h_t = (h_1, h_2, ..., h_M)$ que são conectadas nelas mesmas ao decorrer do tempo com conexões recorrentes. Os estados ocultos da RNN são conjuntos de valores, que representam as informações dos estados passados através do tempo. Essa integração de informação pode definir o comportamento futuro da rede e fazer predições precisas na camada de saída.

Ademais, as RNN utilizam uma função de ativação que é normalmente utilizada nas camadas de saída. Podem ser usadas funções lineares e não lineares, as não lineares são mais poderosas pois podem traçar limites não lineares. Algumas das funções de ativação mais utilizadas são a *sigmoid*, *tanh* e *rectified linear unit* (ReLU) e têm recebido maior atenção do que as outras funções. A escolha das funções de ativação irão depender do problema a ser resolvido.

Após as funções de ativação, é aplicada a função de perda para avaliar o desempenho da rede comparando a saída da camada de saída com a saída desejada de acordo com o conjunto. E, assim como a escolha da função de ativação, a função de perda também depende do problema. Uma das mais conhecidas é a distância Euclidiana.

Com essas informações é possível treinar a RNN. Uma das dificuldades encontradas é a inicialização apropriada dos pesos na rede e no algoritmo de otimização para ajustes de forma a minimizar a perda de treinamento. A inicialização dos pesos na RNN é crucial, geralmente escolhidos os valores 0.01 ou 0.001.

O método de otimização é o gradiente descendente, um método simples e popular em aprendizado profundo. A ideia básica é ajustar os pesos do modelo encontrando as derivadas da função de erro em relação a cada membro das matrizes de pesos do modelo. Para minimizar a perda total, o gradiente descendente altera cada peso proporcionalmente à derivada do erro em relação a esse peso, desde que a ativação não linear funcione são diferenciáveis.

Como a RNN é uma estrutura ao longo do tempo, é necessário estender o gradiente descendente ao longo do tempo para treinar a rede, chamado de retropropagação ao longo do tempo. Porém, as RNN não podem aprender dependências temporais de longo alcance quando gradiente descendente é utilizado para treinamento. Isso é devido ao decaimento exponencial do gradiente, à medida que ele é retropropagado ao longo do tempo, o que é chamado de problema do gradiente descendente.

2.3.2 Long Short-Term Memory

O modelo LSTM ou Memória de longo prazo é um tipo de de rede neural recorrente projetada para resolver as limitações das redes neurais recorrentes tradicionais, particu-

larmente o problema de dependências de longo prazo. As RNN tradicionais sofrem com problema do gradiente de desaparecimento, o que torna difícil para as RNN aprenderem e lembrarem de dependências de longo prazo em dados sequenciais. As redes LSM superam esse problema por meio de uma arquitetura única que lhes permite reter informações por longos períodos (GRAVES; GRAVES, 2012).

A arquitetura LSTM se diferencia por sua célula de memória exclusiva, projetada para manter informações por longos períodos. Essa capacidade é alcançada por meio de uma série de portas que regulam o fluxo de informações para dentro e para fora da célula. Essas portas são cruciais para controlar o estado da célula e o estado oculto em cada etapa de tempo, garantindo que informações importantes sejam retidas e informações irrelevantes sejam descartadas.

Essa arquitetura permite que as redes LSTM mantenham um gradiente estável, possibilitando aprender dependências de longo prazo e ter um bom desempenho em tarefas que envolvem dados sequenciais, como previsão de séries temporais, modelagem de linguagem e reconhecimento de fala.

2.3.2.1 *Forget Gate*

O *Forget Gate* decide quais informações do estado anterior da célula (C_{t-1}) devem ser descartados. Ele pega o estado oculto anterior (h_{t-1}) e a entrada atual (x_t) como entradas, processa-os por meio de uma função de ativação sigmóide e gera um valor entre 0 e 1 para cada componente da célula estado. Este valor determina até que ponto cada componente é retido ou esquecido.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.12}$$

2.3.2.2 *Inpput Gate* e Célula de Memória Candidata ($\tilde{C}_t$)

A porta de entrada (*Input Gate*) controla a atualização do estado da célula com novas informações. Ele também pega o estado oculto anterior e a entrada atual e os processa por meio de uma função de ativação sigmóide para decidir quais valores serão atualizados. Simultaneamente, uma célula de memória candidata ($\tilde{C}_t$) é criada usando uma função de ativação tangente hiperbólica.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.13}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.14}$$

2.3.2.3 Atualização do estado da célula (C_t)

O estado da célula é atualizado combinando o estado anterior da célula, escalonado pela *Forget Gate*, e a célula de memória candidata, escalonada pela porta de entrada.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{2.15}$$

2.3.2.4 Porta de saída (o_t) e estado oculto (h_t)

A porta de saída determina o próximo estado oculto, que é usado para o próximo intervalo de tempo e como parte da saída. Ele processa o estado oculto anterior e a entrada atual por meio de uma função de ativação sigmóide. O novo estado oculto é então calculado usando o estado atualizado da célula passado por uma função de ativação tangente hiperbólica, escalonada pela porta de saída.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.16}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{2.17}$$

2.3.3 Unidade Recorrente Fechada (GRU)

A GRU é um tipo de arquitetura de RNN introduzida por Cho *et al.* (2014) em 2014. Foi projetada para melhorar a RNN padrão e fornecer uma alternativa mais simples às redes LSTM, abordando alguns dos mesmos problemas, como o problema do gradiente de fuga.

Essa arquitetura combina o estado oculto e o estado da célula em um único vetor de estado, simplificando a arquitetura em comparação às LSTM. Usa duas portas, a porta de atualização e a porta de redefinição, para controlar o fluxo de informações.

2.3.3.1 Redefinir porta (r_t)

A porta de redefinição determina quanto do estado oculto anterior (h_{t-1}) deve ser redefinido ou esquecido. É calculado da seguinte forma:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{2.18}$$

2.3.3.2 Atualizar portão (z_t)

A porta de atualização controla quanto do estado oculto anterior deve ser passado para o próximo passo de tempo.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{2.19}$$

2.3.3.3 Estado oculto candidato ($\tilde{h}_t$)

O estado oculto candidato é uma atualização potencial do estado oculto. É influenciado pela porta de redefinição e pela entrada atual (x_t).

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \cdot h_{t-1}, x_t] + b_h) \tag{2.20}$$

2.3.3.4 Estado oculto final (h_t)

O estado oculto final é uma interpolação linear entre o estado oculto anterior e o estado oculto candidato, controla pela porta de atualização.

$$h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \tilde{h}_t \tag{2.21}$$

As vantagens da GRU incluem uma estrutura simples comparada com as LSTM, com menos portas e nenhum estado de célula separado, o que reduz a complexidade computacional. Além disso, geralmente são mais rápidas de treinar e requerem menos memória, tem desempenho comparável às LSTM, tornando-as uma alternativa viável em cenários onde os recursos computacionais são limitados.

2.4 Métricas de avaliação

As métricas de avaliação são cruciais para avaliar o desempenho de modelos de aprendizado de máquina, especialmente em tarefas de classificação. Três das métricas mais comumente usadas são Precisão, Pontuação F1 e Média Macro. Cada uma dessas métricas fornecem informações diferentes sobre o desempenho do modelo (FAWCETT, 2006).

2.4.1 Acurácia

A acurácia é calculada pela proporção de predições corretas em relação ao número de predições. É uma métrica simples e intuitiva frequentemente usada para avaliar a eficácia de um modelo de classificação.

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.22}$$

onde:

- TP (*True Positive*): instâncias que foram preditas como positivas corretamente;
- TN (*True Negative*): instâncias que foram preditas como negativas corretamente;
- FP (*False Positive*): instâncias que foram preditas como positivas incorretamente;
- FN (*False Negative*): instâncias que foram preditas como negativas incorretamente;

Essa métrica é muito útil quando o conjunto de dados possui classes igualmente distribuídas. No entanto, quando uma classe é mais dominante que as outras, a alta precisão pode não refletir o verdadeiro desempenho do modelo.

2.4.2 F1

A pontuação F1 é a média harmônica de precisão e recuperação. É particularmente útil ao lidar com conjuntos de dados desequilibrados, proporcionando um equilíbrio entre a precisão e a recuperação.

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{2.23}$$

$$\text{Recuperação} = \frac{TP}{TP + FN} \tag{2.24}$$

$$F1 = 2 * \frac{\text{Precisão} * \text{Recuperação}}{\text{Precisão} + \text{Recuperação}} \tag{2.25}$$

Com isso, equilibra precisão e recuperação, tornando-a uma métrica robusta para conjuntos de dados desequilibrados onde falsos positivos e falsos negativos são importantes. Também, fornece uma métrica única que captura a compensação entre a precisão e a recuperação do modelo. Porém, é menos intuitiva que a precisão.

2.4.3 Média Macro

A Média Macro calcula as métricas (Precisão, Recuperação e Pontuação F1) para cada classe individualmente e, em seguida, calcula a média dessas métricas. Esta abordagem atribui peso igual a cada classe, independente da sua frequência.

$$\text{Macro F1} = \frac{1}{N}\sum_{i=1}^{N}\text{F1}_i \tag{2.26}$$

onde N é o número de classes e $F1_i$ é a pontuação F1 para cada classe i.

Fornece uma visão equilibrada do desempenho do modelo em todas as classes, o que é particularmente útil em problemas de classificação multi classe com distribuições de classes desequilibradas. Garante que cada classe seja tratada igualmente, destacando o desempenho do modelo em todas as classes. No entanto, pode não refletir bem o desempenho geral se algumas classes forem significativamente mais importante que outras. Também, pode ser menos informativo se o conjunto de dados for altamente desequilibrado, pois não considera as diferentes frequências de classe.

3 TRABALHOS RELACIONADOS

Alguns trabalhos na literatura fornecem informações sobre o uso de série temporal para predizer o preço e as tendências do mercado de ações de acordo com o gráfico de preços de ações. Outros utilizam apenas informações da Internet, como notícias e sentimentos de usuários nas mídias sociais para análise dos eventos e impactos na movimentação dos preços. Abaixo são comentados alguns artigos que relacionam as informações de preços e/ou de notícias na *Web* para predizer preços de mercados.

De acordo com Zhang *et al.* (2018), uma nova abordagem para a previsão do mercado de ações é proposta utilizando informações heterogêneas. São extraídos eventos de notícias da *web* e os sentimentos dos usuários das redes sociais, investigam também, os impactos conjuntos nos movimentos dos preços das ações através de um *framework* de fatoração de matriz e tensor. Esse tensor é construído para unir os dados e capturar as relações entre os eventos e os sentimentos dos investidores.

Outro estudo feito por Li, Shang and Wang (2019), mostra um novo método de previsão de preços de petróleo bruto, utilizando mineração de textos de mídia online, com o objetivo de capturar os antecedentes de mercado mais imediatos das flutuações de preços. É utilizada um Rede Neural Convolucional (CNN) para a extração de padrões ocultos nas mídias de notícias. É proposto um método de agrupamento de recursos baseado no modelo de tópico *Latent Dirichlet Allocation* (LDA) para distinguir efeitos de vários tópicos de notícias online. A abordagem de síntese tópico-sentimento é baseada em texto para construção de série temporal com base no modelo CNN, análise de sentimento e identificação de tópico.

Diante disso, a abordagem a ser utilizada nesse trabalho envolve as tendências em relação aos preços do mercado de ações do índice Bovespa utilizando uma representação de textos vetorial. Para isso é necessário buscar um conjunto de informações em relação aos gráficos dos preços e informações de notícias. Será utilizado o método de validação cruzada *K-Fold* para fazer o treinamento e teste em diversas combinações nos conjuntos de representações vetoriais. E também, um método de combinação dos valores do gráfico e da classificação dos textos das notícias será utilizado como entrada para a uma rede neural convolucional predizendo as tendências no mercado de ações.

Outros trabalhos relacionados podem ser vistos na tabela **??** e as suas comparações entre os modelos utilizados e as representações de textos.

Tabela 1 – Referências e estratégias utilizadas para previsão de mercado financeiro.

Referência (Autores)	**Representação**	**Estratégia**	**Modelo**
Picasso *et al.* (2019)	Sentiment Embeddings, Análise Técnica	Previsão de Tendências	Redes Neurais
Li, Shang and Wang (2019)	Previsão de Preços	Redes Neurais Profundas	
Zhang *et al.* (2019)	Multisourced Data	Previsão de Tendências	Hidden Markov Model
Li, Wu and Wang (2020)	Sentimentos de Notícias, Preços de Ações	Previsão de Preços	Modelos de Regressão
Xu *et al.* (2020)	Tweets, Preços Históricos	Previsão de Movimentos	Redes de Atenção
Xu *et al.* (2020)	Múltiplas Fontes de Dados	Previsão de Movimentos	Modelos de Machine Learning
Zhang *et al.* (2018)	Fusão de Informações Heterogêneas	Previsão de Movimentos	Modelos de Machine Learning
Avramelou *et al.* (2024)	Aprendizado por Reforço	Deep Reinforcement Learning	
Tan *et al.* (2024)	Clues Visuais	Precificação de Ativos	Deep Learning
Zhang *et al.* (2024)	Preços de Ações, Texto	Previsão de Movimentos	Modelos de Fusão de Informação
Wang, Hsiao and Liou (2024)	Indicadores Técnicos, Fatores de Chip, Notícias de Ações	Previsão de Preços	Multi-Kernel Approach
Chang and Zhang (2023)	Comentários em Fóruns de Especialistas	Análise de Sentimentos	Modelos de Machine Learning
Zhao *et al.* (2023)	Dados Financeiros Heterogêneos	Previsão de Movimentos	Multi-Head Attention
Usmani and Shamsi (2023)	Previsão de Movimentos	LSTM	
Gu *et al.* (2023)	Sentiment Score, Efeito de Fim de Semana	Previsão de Preços	Deep Learning

Continua na próxima página...

Referência (Autores)	**Representação do Texto**	**Estratégia**	**Modelo**
Yang *et al.* (2023)	Decisões Três Vias Sequenciais	Previsão de Movimentos	Modelos de Machine Learning
Wang *et al.* (2023)	Indicadores Técnicos, Sentimentos de Mídia Social	Previsão de Tendências	Modelos de Machine Learning
Tadphale *et al.* (2023)	Sentimentos de Notícias	Previsão de Taxas de Câmbio	Modelos de Regressão
Ma *et al.* (2023)	Classificação Agregada de Múltiplas Fontes	Previsão de Movimentos	Modelos de Classificação
Eslamieh, Shajari and Nickabadi (2023)	User2Vec	Previsão de Movimentos	Redes Neurais Convolucionais e Recorrentes
Wang *et al.* (2022)	Fusão de Informações Heterogêneas	Previsão de Movimentos	Abordagem Baseada em Grafos
Li *et al.* (2022)	Modelo de Pontuação e Triagem	Seleção de Ações	Modelos de Pontuação e Triagem
Lin *et al.* (2022)	Texto e Informação Numérica	Previsão de Preços	Redes Convolucionais com Atenção Espacial-Temporal
Filho, Marcacini and Rezende (2022)	Dados Textuais	Previsão de Preços de Commodities	Modelos de Machine Learning
Ye *et al.* (2022)	Comentários no Twitter	Previsão de Preços	Stacking Ensemble Deep Learning

Pode-se destacar que os trabalhos revisados apresentam diversas abordagens para prever os movimentos no mercado financeiro, com fontes de dados diversas e técnicas diferentes de representação de texto e modelos preditivos. Essas abordagens demonstram a importância de integrar dados de séries temporais com informações de notícias ou sentimentos, para obter previsões precisas e relevantes.

4 PROPOSTA DO MODELO PARA PREVER AS TENDÊNCIAS DO ÍNDICE BOVESPA

Para começar a construção do modelo foi feito um fluxograma demonstrado pela Figura 2 que acompanha os processos feitos até a entrada para o modelo LSTM.

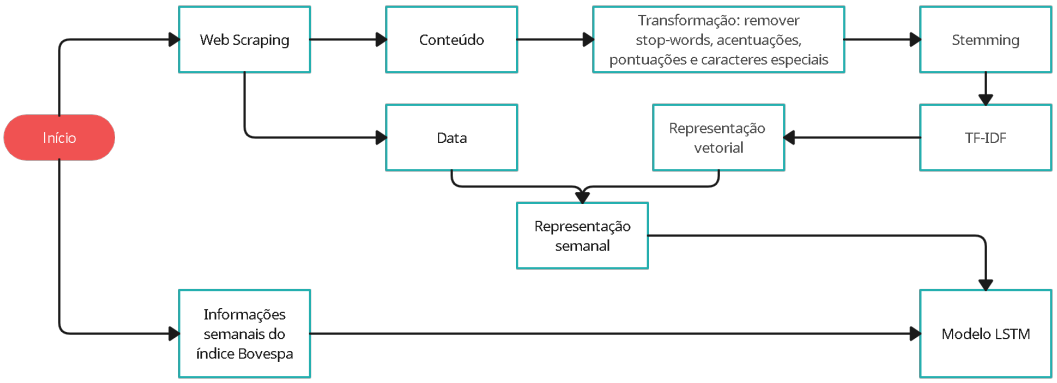

Figura 2 – Fluxograma dos processos que compõem o modelo

De início foi necessário coletar os dados do gráfico do índice Bovespa e as notícias que ocorreram no período. Foram coletadas notícias do site [1]www.exame.com sobre economia brasileira através de *web-scraping*. Com isso foram armazenadas as datas e os conteúdos de cada notícia. A partir do conteúdo foi necessário remover as *stop-words*, que são as palavras que não influenciam na análise do texto, como por exemplo “a“, “o“, “é“, “de“, “para“, “com“, “que“, remover acentuações, pontuações e caracteres especiais. Por último deixar todas as letras minúsculas.

Após essa transformação do texto, um processo de *Stemming* foi utilizado para reduzir as palavras para sua forma raiz, ou seja, deixar as palavras relacionadas no mesmo radical comum entre elas, como por exemplo “correndo“, “correr“ e “correria“, essas palavras se tornam “corr“. Esse processo ajuda a reduzir a quantidade de palavras do texto e para isso foi utilizada a a função *RSLPStemmer* da biblioteca NLTK do *Python*.

O último passo foi criar uma representação numérica (*bag of words*) das palavras mais importantes dos textos, essa representação mostra a quantidade de palavras que estão contidas no conjunto de textos de todas as notícias. Para isso foi utilizado o modelo estatístico TF-IDF, se alguma palavra aparece em muitas notícias, ela é considerada menos importante, se a sua frequência nos textos é baixa, ela é considerada mais importante.

Foram consideradas apenas as 30 palavras mais importantes para a junção dos dados com as informações do índice Bovespa. Os dados do índice são formados pelos

[1] Exame é uma revista brasileira mensal, de circulação nacional especializada em economia, negócios, política e tecnologia.

atributos:

- Data;
- Último: valor de fechamento do índice no final do dia de negociação, é o valor final do dia anterior de acordo com todas as transações feitas no dia anterior;
- Abertura: valor do índice no início do dia de negociação, é o ponto de partida daquele dia específico e pode ser comparado com o valor de fechamento do dia anterior para ver a mudança inicial;
- Máxima: valor mais alto atingido pelo índice durante o dia de negociação, mostrando os momentos de maior valorização;
- Mínima: valor mais baixo atingido pelo índice durante o dia de negociação, momento de maior depreciação;
- Volume: volume total das negociações (número de ações transacionadas) no dia;
- Variação percentual: variação percentual do índice em relação ao valor de fechamento do dia anterior.

Ainda, para uma análise a longo prazo, o período escolhido foi de uma semana entre as datas de cada registro do índice e pode ser visto um exemplo na Tabela 2. Diante disso, para juntar as informações das notícias com as informações do índice, foi necessário somar os valores de cada atributo da *bag of words* das notícias para formar uma *bag of words* da semana e normalizar os atributos do índice Bovespa para uma melhor convergência do modelo. Contudo, para o atributo de variação percentual, que é a classe utilizada para o modelo, foi considerado valores -1 para uma variação negativa, 0 para sem variação e 1 para variação positiva.

Data	**Último**	**Abertura**	**Mínimo**	**Máximo**	**Volume (M)**	**Variação (%)**
08/09/2024	134.882	134.574	135.879	133.591	38.66	0.23
01/09/2024	134.572	136.004	136.838	134.171	38.85	-1.05
25/08/2024	136.004	135.608	137.469	134.91	48.55	0.29
18/08/2024	135.608	133.953	137.04	133.953	45.66	1.24
11/08/2024	133.953	130.615	134.781	130.615	51.97	2.56

Tabela 2 – Números do índice Bovespa semanal

Com os dados das notícias e do índice Bovespa devidamente preparados e combinados, um modelo LSTM é implementado. O LSTM é um tipo especializado de Rede Neural

Recorrente para lidar com dados sequenciais e que aprende padrões em séries temporais. Esse modelo foi escolhido por sua habilidade em capturar dependências a longo prazo, ou seja, pode detectar como eventos e flutuações passadas influenciam tendências futuras.

Para aplicar esse modelo, a combinação dos dados foram transformados em uma estrutura adequada para o treinamento. Com os dados normalizados, janelas de tempo foram criadas. Cada janela contém os dados das últimas n semanas para prever o valor para a semana seguinte. Após, houve a divisão do conjunto de dados entre treinamento, validação e teste. O conjunto de treinamento foi usado para ajustar os pesos da rede. O conjunto de validação foi usado durante o treinamento para monitorar o desempenho do modelo e ajustar os hiperparâmetros, garantindo que ele não se ajuste demais aos dados de treinamento. Por fim, o conjunto de teste foi usado para avaliar o desempenho do modelo em dados não vistos.

A estrutura do modelo requer três dimensões de dados de entrada no seguinte formato: número de amostras, tamanho da janela e número de atributos. Para cada semana, o modelo recebe os atributos do índice e o conjunto de palavras mais importantes como entrada.

A arquitetura do modelo foi construída utilizando a biblioteca *Keras* disponível no *TensorFlow* do Python. Essa arquitetura consiste das seguintes camadas:

- Camada de entrada: o modelo começa com uma camada de entrada, que espera dados na forma de (tamanho da janela n, número de atributos p). Esta forma indica que o modelo usará dados das últimas n semanas para previsão;
- Camada LSTM: A primeira camada é uma camada LSTM com um 64 neurônios. Esta camada aprende padrões temporais dos dados sequenciais ao longo das janelas de n semanas;
- Camada densa: após a camada LSTM, uma camada densa com 8 neurônios e ativação ReLU é adicionada. Esta camada ajuda a extrair padrões não lineares adicionais da saída da cama LSTM;
- Camada de saída: a camada final é uma camada densa com um único neurônio e ativação linear. Esta camada prevê o valor da próxima semana do índice Bovespa.

O modelo foi treinado utilizando o otimizador Adam, o qual é altamente eficaz para redes neurais, junto com a função de perda chamada Erro Quadrático Médio (*Mean Squared Error*) para medir a diferença entre o valor predito e o valor real. Logo após o treinamento, o desempenho do modelo foi avaliado no conjunto de teste usando as seguintes métricas:

- Erro Absoluto Médio (MAE): a diferença absoluta média entre os valores previstos e reais;
- Erro Quadrático Médio (MSE): a média das diferenças quadradas, que penaliza erros maiores com mais pesos;
- Pontuação R^2 (Coeficiente de Determinação): indica o quão bem os valores previstos se ajustam aos dados reais, com valores variando de 0 a 1.

5 AVALIAÇÃO EXPERIMENTAL

Nesta seção é descrito os resultados e discussões encontrados após a aplicação do modelo de rede neural recorrente LSTM na combinação das notícias encontradas através de um *web-scraping* no site de notícias www.exame.com e das informações semanais do índice Bovespa de acordo com os dados coletados no site [1]https://br.investing.com/indices/bovespa-historical-data.

5.1 Conjuntos de Dados

As notícias foram armazenadas em arquivo *csv* para a aplicação da transformação do conteúdo em uma representação vetorial chamada *bag of words* que contém as 30 palavras mais importantes do documento (conjunto total dos conteúdos das notícias). Os radicais das palavras mais importantes foram: aind, alt, ano, aument, banc, bilho, brasil, cent, cresc, dev, diss, econom, empr, govern, indic, inflaca, invest, mai, med, merc, nest, nov, pal, pass, pib, pod, presid, segund, set e sobr. Cada radical é um atributo utilizado no conjunto representado por um valor numérico de 0 a 1.

Como o período do índice foi escolhido como semanal, foi considerado os valores dos atributos como semanais também. Com isso, no período de uma semana houve uma ou mais notícias ou nenhuma notícia. Para isso, se houve apenas uma notícia, os valores das *bag of words* foram mantidos os mesmos, para mais de uma notícia, os valores foram somados e normalizados para valores entre 0 e 1, e para nenhuma notícia, foram considerados valores nulos para cada atributo dos radicais.

Todos os dados do índice Bovespa foram normalizados levando em consideração cada atributo separadamente, com exceção da variação percentual, o qual levou-se em consideração a classe para a qual o modelo faria a previsão. Esse atributo teve como valores: valor -1 para variação percentual negativa, valor 0 para sem variação percentual e valor 1 para variação percentual positiva.

O conjunto de dados consistiu da concatenação dos dados do índice Bovespa e das *bag of words* correspondentes naquele período da semana. O período total foi escolhido de acordo com a disponibilidade das notícias, sendo a data mais antiga 24/06/2012 e a data mais recente 08/09/2024, totalizando 11007 notícias e 638 semanas. Portanto, o conjunto de dados tem 638 registros e 38 atributos, sendo um deles a classe Variação Percentual.

[1] O Investing.com ofecere notícias e informações sobre investimentos

5.2 Configuração Experimental

O conjunto foi configurado de forma a atender a entrada do modelo LSTM, alguns atributos estavam na forma de texto. Foram retiradas as vírgulas e colado pontos pra diferenciar os números decimais. Também, foram retirados os caracteres de porcentagem no atributo Variação Percentual. Come exceção da Variação Percentual e Data, os atributos foram normalizados conforme a equação (5.1)

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{5.1}$$

onde X é o valor original do atributo a ser normalizado, X_{min} é o menor valor do atributo em todo o conjunto, X_{max} é o maior valor do atributo em todo o conjunto e X' é o valor do atributo depois de normalizado. Todos os valores ficaram entre 0 e 1.

Os atributos de notícias, formado pela *bag of words* com as 30 palavras mais importantes também passaram pelo mesmo processo de normalização após a soma das notícias semanais formando uma *bag of words* semanal. Alguns dos registros semanais ficaram com os atributos da *bag of words* ficaram zerados, pois não tiveram notícias relacionadas na semana.

Essa combinação foi organizada na forma de uma janela de dados de 4 semanas anteriores para a entrada do modelo LSTM usada para prever o valor do índice da próxima semana. O tamanho da entrada consiste das 4 semanas e 37 atributos do conjunto. O modelo deve aprender com as sequências temporais, portanto cada entrada deve conter as 4 semanas anteriores e a próxima semana deve ser a saída do modelo e pode ser visto um exemplo na Figura 3.

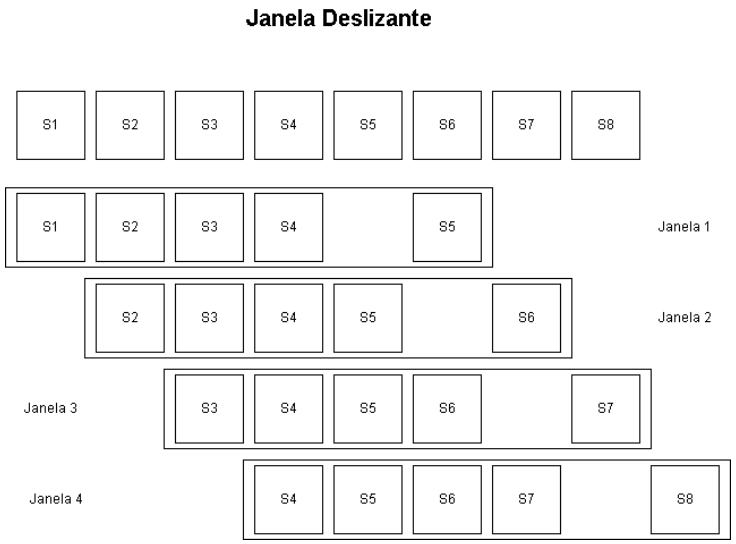

Figura 3 – Exemplo de janela deslizante

Com a estrutura feita, os dados foram divididos em 70% para treinamento, 15% para validação e 15% para teste. O treinamento foi utilizado para ajustar os pesos do modelo, aprendendo em cada época as relações entre os atributos. Durante o treinamento, o modelo foi avaliado periodicamente no conjunto de validação para medir a capacidade de generalização e ajuste de hiperparâmetros, interrompendo o treinamento se o desempenho do conjunto de validação piorar. Depois do termino de treinamento, o conjunto de teste foi usado para avaliar o desempenho final do modelo.

A arquitetura do modelo composta pela camada de entrada, camada LSTM, camada densa e com ativação ReLU e camada densa com ativação linear gerou o seguinte modelo:

Tipo de camada	**Forma de saída**	**Número de parâmetros**
LSTM	(None, 64)	26,112
Dense	(None, 8)	520
Dense	(None, 1)	9

Tabela 3 – Arquitetura do modelo LSTM utilizado na previsão do índice Bovespa.

5.3 Resultados e Discussões

Após treinar o modelo, os resultados foram gerados e analisados conforme a função de perda, a quantidade de época escolhida, a raiz quadrada do erro médio quadrático, a validação da perda e a validação do RMSE. As épocas geraram uma passagem completa por todo o conjunto de dados de treinamento e o modelo ajustou seus pesos com base no erro que cometeu nas previsões. Os valores da primeira e da última época são mostrados abaixo na tabela:

Época	**Loss**	**RMSE**	**Validation Loss**	**Validation RMSE**
1	0.6095	0.7799	0.4614	0.6793
100	0.2103	0.4584	0.2719	0.5214

Tabela 4 – Valores da primeira e última época.

O desempenho do modelo mostrou um bom progresso, pois tanto a perda quanto o RMSE, nos dados de treinamento e validação, estão diminuindo. Sugerindo que o modelo está aprendendo com os dados e não está acontecendo o *overfitting*. Nas primeiras 20 épocas, há uma redução significativa na perda e no RMSE, o que é comum em modelos que estão começando a aprender. Logo após a taxa de diminuição se estabiliza, apresentando normalidade.

Com essa estabilidade, a partir de uma certa época, as melhorias na perda de validação e RMSE se tornam menores. Podendo indicar que o modelo está se aproximando de um ponto de saturação em termos de aprendizado. Apesar da perda e o RMSE nos dados de validação foram diminuindo, a perda de validação começa a aumentar, podendo ser um sinal de *overfitting*. Neste caso, paradas antecipadas e técnicas de regularização ou até mesmo ajustes na arquitetura do modelo podem se tornar necessárias.

A partir dessa análise é possível ver as métricas de avaliação utilizadas, como a matriz de confusão, a acurácia, *recall*, *F1 score* e suporte.

Classe Real	**Negativa**	**Neutra**	**Positiva**
Negativa	0	0	0
Neutra	0	5	37
Positiva	0	7	46

Tabela 5 – Matriz de confusão para as previsões do modelo nas classes de tendências negativas, neutras e positivas.

Com isso, o modelo mostrou que teve um acerto maior para a classe positiva em relação as outras. Porém não conseguiu identificar nenhuma classe negativa no conjunto de teste. O conjunto de dados, conforme as configurações escolhidas, mostrou que dos 638 registros, 298 deles possuem variação negativa, 340 com variação positiva e nenhum registro sem variação.

Classe	**Precisão**	**Recall**	**F1-Score**	**Suporte**
0.0	0.42	0.12	0.19	42
1.0	0.55	0.87	0.68	53
Acurácia		0.54		
Média Macro	0.49	0.49	0.43	95
Média Ponderada	0.49	0.54	0.46	95

Tabela 6 – Relatório de classificação para o modelo com as classes 0 e 1.

A acurácia indica que mais da metade dos dados foram classificados corretamente. A precisão, sendo aproximadamente 49%, indica que, em média, o modelo não tem alta confiança nas previsões corretas para todas as classes. Com o *recall* em quase 50%, esse valor sugere que o modelo esta recuperando corretamente quase metade dos registros para cada classe, sendo que a classe positiva apresenta melhores resultados. Com o desbalanceamento dos acertos de cada classe, o resultado é refletido no valor da métrica *F1-Score* com 43.08%.

6 CONCLUSÕES

Este trabalho teve como objetivo a criação de um modelo de previsão das tendências do índice Bovespa utilizando um modelo de aprendiza de máquina chamado LSTM. A relevância deste estudo está na capacidade de fornecer informações que podem ser importantes para o comportamento do mercado financeiro, em específico o índice Bovespa, podendo ser crucial para analistas e investidores do mercado de ações.

Os resultados mostraram que o conjunto de informações dos gráficos semanais do índice Bovespa junto com as notícias que ocorreram na semana podem ser utilizados como entrada para os modelos de aprendizado de máquina. E nesse modelo LSTM, com as configurações apresentadas, chegou a uma precisão de 54%, indicando uma capacidade moderada de prever as variações do índice. A matriz de confusão apresentada ajudou a entender quais foram as classes que foram previstas pelo conjunto de testes, porém, não foi possível o modelo prever as classes de variação negativa, prejudicando as outras métricas de avaliação.

Apesar dos resultados apresentados, é importante mostrar quais são as possíveis limitações do modelo utilizado, que um conjunto com poucos registros pode influenciar no resultado, e também, utilizar algum método que possa balancear as classes e testar novamente os resultados. Em estudos futuros, podem ser modificados essas informações, coletar mais notícias de outras fontes de notícias e utilizar um conjunto maior do índice Bovespa, podendo ser um conjunto diário.

Essas técnicas de aprendizado e a constante evolução do mercado financeiro, podem ajudar analistas e investidores a terem mais confiança na hora de investir nesse mercado de ações. Os modelos podem ser utilizados para outros índices também, podendo ser necessário a modificação de parâmetros do modelo para melhor se ajustar as necessidades do estudo.

REFERÊNCIAS

AL-OBAYDY, W. I. *et al.* Document classification using term frequency-inverse document frequency and k-means clustering. **Indonesian Journal of Electrical Engineering and Computer Science**, v. 27, n. 3, p. 1517–1524, 2022.

AVRAMELOU, L. *et al.* Deep reinforcement learning for financial trading using multi-modal features. **Expert Systems with Applications**, Elsevier, v. 238, p. 121849, 2024.

BERGMEIR, C.; HYNDMAN, R. J.; KOO, B. A note on the validity of cross-validation for evaluating autoregressive time series prediction. **Computational Statistics & Data Analysis**, Elsevier, v. 120, p. 70–83, 2018.

BLOCK, C. R. How to handle seasonality. **Statistical Analysis Center, Illinois Criminal Justice Information Authority**, 1984.

BRONDANI, E. L. A. *et al.* Influência do índice dow jones industrial average sobreo índice ibovespa. **Horizontes Empresariales**, v. 12, n. 2, p. 23–44, 2013.

CHANG, Z.; ZHANG, Z. Judging stock trends according to the sentiments of stock comments in expert forums. **Electronics**, MDPI, v. 12, n. 3, p. 722, 2023.

CHO, K. *et al.* Learning phrase representations using rnn encoder-decoder for statistical machine translation. **arXiv preprint arXiv:1406.1078**, 2014.

DEVLIN, J. *et al.* Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018.

ESLAMIEH, P.; SHAJARI, M.; NICKABADI, A. User2vec: A novel representation for the information of the social networks for stock market prediction using convolutional and recurrent neural networks. **Mathematics**, MDPI, v. 11, n. 13, p. 2950, 2023.

ESLING, P.; AGON, C. Time-series data mining. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 45, n. 1, p. 1–34, 2012.

FAWCETT, T. An introduction to roc analysis. **Pattern recognition letters**, Elsevier, v. 27, n. 8, p. 861–874, 2006.

FILHO, I. J. R.; MARCACINI, R. M.; REZENDE, S. O. On the enrichment of time series with textual data for forecasting agricultural commodity prices. **MethodsX**, Elsevier, v. 9, p. 101758, 2022.

FITZ-GIBBON, C. T. **Performance indicators**. [*S.l.: s.n.*]: Multilingual Matters, 1990. v. 2.

FORTUIN, L. Performance indicators—why, where and how? **European journal of operational research**, Elsevier, v. 34, n. 1, p. 1–9, 1988.

GOLDBERG, Y.; LEVY, O. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. **arXiv preprint arXiv:1402.3722**, 2014.

GRAVES, A.; GRAVES, A. Long short-term memory. **Supervised sequence labelling with recurrent neural networks**, Springer, p. 37–45, 2012.

GU, J. *et al.* Deep learning model with sentiment score and weekend effect in stock price prediction. **SN Business & Economics**, Springer, v. 3, n. 7, p. 119, 2023.

HAMILTON, J. D. **Time series analysis**. [*S.l.: s.n.*]: Princeton university press, 2020.

HOTHO, A.; NÜRNBERGER, A.; PAASS, G. A brief survey of text mining. **Journal for Language Technology and Computational Linguistics**, v. 20, n. 1, p. 19–62, 2005.

HYNDMAN, R. J.; ATHANASOPOULOS, G. **Forecasting: principles and practice**. [*S.l.: s.n.*]: OTexts, 2018.

ISHII, R. P.; RIOS, R. A.; MELLO, R. F. Classification of time series generation processes using experimental tools: a survey and proposal of an automatic and systematic approach. **International Journal of Computational Science and Engineering**, Inderscience Publishers, v. 6, n. 4, p. 217–237, 2011.

JESUS, D. C. d. Ibovespa x mídia: Análise de conteúdo da gazeta mercantil em busca da influência da imprensa na bolsa brasileira. Pontifícia Universidade Católica de São Paulo, 2011.

KAO, A.; POTEET, S. R. **Natural language processing and text mining**. [*S.l.: s.n.*]: Springer Science & Business Media, 2007.

KOHAVI, R. *et al.* A study of cross-validation and bootstrap for accuracy estimation and model selection. *In*: MONTREAL, CANADA. **Ijcai**. [*S.l.: s.n.*], 1995. v. 14, n. 2, p. 1137–1145.

LI, X.; SHANG, W.; WANG, S. Text-based crude oil price forecasting: A deep learning approach. **International Journal of Forecasting**, Elsevier, v. 35, n. 4, p. 1548–1560, 2019.

LI, X.; WU, P.; WANG, W. Incorporating stock prices and news sentiments for stock market prediction: A case of hong kong. **Information Processing & Management**, Elsevier, v. 57, n. 5, p. 102212, 2020.

LI, Y. *et al.* How to make machine select stocks like fund managers? use scoring and screening model. **Expert Systems with Applications**, Elsevier, v. 196, p. 116629, 2022.

LIN, C.-T. *et al.* Spatial-temporal attention-based convolutional network with text and numerical information for stock price prediction. **Neural Computing and Applications**, Springer, v. 34, n. 17, p. 14387–14395, 2022.

LIU, Y. *et al.* Roberta: A robustly optimized bert pretraining approach. **arXiv preprint arXiv:1907.11692**, 2019.

MA, Y. *et al.* Multi-source aggregated classification for stock price movement prediction. **Information Fusion**, Elsevier, v. 91, p. 515–528, 2023.

MONZONI, M.; BIDERMAN, R.; BRITO, R. P. d. Finanças sustentáveis e o caso do índice de sustentabilidade empresarial da bovespa. Centro de Estudos em Sustentabilidade (FGVces), 2006.

OFER, D.; BRANDES, N.; LINIAL, M. The language of proteins: Nlp, machine learning & protein sequences. **Computational and Structural Biotechnology Journal**, Elsevier, v. 19, p. 1750–1758, 2021.

OJALA, M.; GARRIGA, G. C. Permutation tests for studying classifier performance. **Journal of machine learning research**, v. 11, n. 6, 2010.

OLIVEIRA, F. A. D.; NOBRE, C. N.; ZÁRATE, L. E. Applying artificial neural networks to prediction of stock price and improvement of the directional prediction index–case study of petr4, petrobras, brazil. **Expert systems with applications**, Elsevier, v. 40, n. 18, p. 7596–7606, 2013.

ORENGO, V. M.; HUYCK, C. R. A stemming algorithmm for the portuguese language. *In*: **spire**. [*S.l.: s.n.*], 2001. v. 8, p. 186–193.

PHAKOE, T. Key performance indicators applied in construction. 2023.

PICASSO, A. *et al.* Technical analysis and sentiment embeddings for market trend prediction. **Expert Systems with Applications**, Elsevier, v. 135, p. 60–70, 2019.

SALEHINEJAD, H. *et al.* Recent advances in recurrent neural networks. **arXiv preprint arXiv:1801.01078**, 2017.

SAVINO, M. M.; BRUN, A.; XIANG, C. A fuzzy-based multi-stage quality control under the iso 9001: 2015 requirements. **European Journal of Industrial Engineering**, Inderscience Publishers (IEL), v. 11, n. 1, p. 78–100, 2017.

SINGHAL, A. *et al.* Modern information retrieval: A brief overview. **IEEE Data Eng. Bull.**, v. 24, n. 4, p. 35–43, 2001.

TADPHALE, A. *et al.* Impact of news sentiment on foreign exchange rate prediction. *In*: IEEE. **2023 3rd International Conference on Intelligent Technologies (CONIT)**. [*S.l.: s.n.*], 2023. p. 1–8.

TAIEB, S. B. *et al.* A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. **Expert systems with applications**, Elsevier, v. 39, n. 8, p. 7067–7083, 2012.

TAN, J. *et al.* Asset pricing via fused deep learning with visual clues. **Information Fusion**, Elsevier, v. 102, p. 102049, 2024.

USMANI, S.; SHAMSI, J. A. Lstm based stock prediction using weighted and categorized financial news. **PloS one**, Public Library of Science San Francisco, CA USA, v. 18, n. 3, p. e0282234, 2023.

VARISCO, M. *et al.* Kpis for manufacturing operations management: driving the iso22400 standard towards practical applicability. **IFAC-PapersOnLine**, Elsevier, v. 51, n. 11, p. 7–12, 2018.

VIJAYARANI, S. *et al.* Preprocessing techniques for text mining-an overview. **International Journal of Computer Science & Communication Networks**, v. 5, n. 1, p. 7–16, 2015.

WANG, H.-C.; HSIAO, W.-C.; LIOU, R.-S. Integrating technical indicators, chip factors and stock news for enhanced stock price predictions: A multi-kernel approach. **Asia Pacific Management Review**, Elsevier, v. 29, n. 3, p. 292–305, 2024.

WANG, J. *et al.* A graph-based approach to multi-source heterogeneous information fusion in stock market. **Plos one**, Public Library of Science San Francisco, CA USA, v. 17, n. 8, p. e0272083, 2022.

WANG, Z. *et al.* Learning-based stock trending prediction by incorporating technical indicators and social media sentiment. **Cognitive Computation**, Springer, v. 15, n. 3, p. 1092–1102, 2023.

WILBUR, W. J.; SIROTKIN, K. The automatic identification of stop words. **Journal of information science**, Sage Publications Sage CA: Thousand Oaks, CA, v. 18, n. 1, p. 45–55, 1992.

XIONG, T.; BAO, Y.; HU, Z. Beyond one-step-ahead forecasting: evaluation of alternative multi-step-ahead forecasting models for crude oil prices. **Energy Economics**, Elsevier, v. 40, p. 405–415, 2013.

XU, H. *et al.* Stock movement predictive network via incorporative attention mechanisms based on tweet and historical prices. **Neurocomputing**, Elsevier, v. 418, p. 326–339, 2020.

YAMAMOTO, M.; CHURCH, K. W. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. **Computational Linguistics**, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , v. 27, n. 1, p. 1–30, 2001.

YANG, X. *et al.* Multi-granularity stock prediction with sequential three-way decisions. **Information Sciences**, Elsevier, v. 621, p. 524–544, 2023.

YE, Z. *et al.* A stacking ensemble deep learning model for bitcoin price prediction using twitter comments on bitcoin. **Mathematics**, MDPI, v. 10, n. 8, p. 1307, 2022.

ZHANG, Q. *et al.* Incorporating stock prices and text for stock movement prediction based on information fusion. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 127, p. 107377, 2024.

ZHANG, X. *et al.* Enhancing stock market prediction with extended coupled hidden markov model over multi-sourced data. **Knowledge and Information Systems**, Springer, v. 61, p. 1071–1090, 2019.

ZHANG, X. *et al.* Improving stock market prediction via heterogeneous information fusion. **Knowledge-Based Systems**, Elsevier, v. 143, p. 236–247, 2018.

ZHAO, C. *et al.* A structured multi-head attention prediction method based on heterogeneous financial data. **PeerJ Computer Science**, PeerJ Inc., v. 9, p. e1653, 2023.