Universidade Federal do ABC

Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas

Curso de Engenharia de Informação

Trabalho de Graduação
Detecção de Notícias Falsas Utilizando Modelos de Aprendizado de Máquina

Matheus Ribeiro Barison Martins Silva

Santo André, São Paulo
2024

MATHEUS RIBEIRO BARISON MARTINS SILVA

DETECÇÃO DE NOTÍCIAS FALSAS UTILIZANDO MODELOS DE APRENDIZADO DE MÁQUINA

Trabalho de Conclusão de Curso apresentado à Universidade Federal do ABC como requisito parcial para a obtenção do título de Bacharel em Engenharia da Informação. Orientador: Prof. Dr. Cláudio José Bordin Júnior.

AGRADECIMENTOS

Agradeço a todos que emanaram energias construtivas na minha caminhada durante a graduação.

RESUMO

A disseminação descontrolada de dados e notícias falsas, especialmente pelas mídias sociais tem se tornado uma crescente preocupação na sociedade contemporânea. Este trabalho explora o potencial da inteligência artificial (IA) como uma ferramenta no combate a esse fenômeno. Foi realizada uma análise detalhada de técnicas de aprendizado de máquina supervisionado para a detecção de notícias falsas.

O estudo empregou duas bases de dados distintas com notícias em português, sendo a primeira composta por 3600 registros de notícias verdadeiras e 3600 de notícias falsas e a segunda, mais reduzida, com 221 notícias verdadeiras e 228 notícias falsas. Esses dados serviram de entrada para quatro modelos de aprendizado supervisionado - árvore de decisão, florestas aleatórias, regressão logística e *gradient boosting*.

Para a base de dados principal, os resultados foram melhores, destacando-se os *scores* de precisão: árvore de decisão (0,89), regressão logística (0,95), *gradient boosting* (0,96) e florestas aleatórias (0,93). Entretanto, na base de dados secundária, os *scores* foram mais baixos: árvore de decisão (0,57), regressão logística (0,58), gradient boosting (0,56) e florestas aleatórias (0,58).

Os resultados indicam que apesar de ter sido implementados algoritmos relativamente simples, com o uso de uma base de dados suficientemente grande e utilizando as etapas de pré-processamento pertinentes é possível alcançar bons resultado para classificação de notícias falsas. Este estudo contribui para a compreensão do papel da IA na luta contra a disseminação de notícias falsas, abordando os principais aspectos envolvidos nos processos de aprendizado de máquina supervisionado para problemas do tipo classificação.

Palavras chave: inteligência artificial; notícias falsas; disseminação de dados; aprendizado de máquina supervisionado; algoritmos de classificação.

ABSTRACT

The uncontrolled spread of data and false information, especially through social media, has emerged as a growing concern in contemporary society. This study explores the potential of artificial intelligence (AI) as a tool to combat this phenomenon. A detailed analysis of supervised machine learning techniques for detecting fake news was conducted.

Two distinct databases in Portuguese were employed, with the first comprising 3600 records of true news and 3600 of false news, and the second, smaller, consisting of 221 true news and 228 false news. These datasets served as input for four supervised learning algorithms - decision tree, random forests, logistic regression, and gradient boosting.

For the primary database, the results were better, with notable precision scores: decision tree (0.89), logistic regression (0.95), gradient boosting (0.96), and random forests (0.93). However, in the secondary database, the scores were lower: decision tree (0.57), logistic regression (0.58), gradient boosting (0.56), and random forests (0.58).

The findings suggest that despite implementing relatively simple algorithms, with the use of a sufficiently large dataset and relevant preprocessing steps, it is possible to achieve good results for the classification of fake news. This study contributes to understanding the role of AI in the fight against the spread of fake news, addressing key aspects involved in supervised machine learning processes for classification problems.

Keywords: artificial intelligence; fake news; dissemination of data; supervised machine learning; classification’s algorithms.

LISTA DE TABELAS

LISTAS DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

Ao passo em que as pessoas continuam desbravando os caminhos virtuais da sociedade atual, a criação de dados atinge uma escala impressionante e cresce a uma taxa elevada. A quantidade exorbitante de informações é resultado direto dos cliques dados, compartilhamentos feitos, transações realizadas e interações promovidas nas redes sociais. A presença cada vez maior de dispositivos conectados à internet, incluindo *smartphones*, computadores e dispositivos IoT (*Internet of Things*), impulsiona incessantemente a expansão do amplo ecossistema de dados. Segundo a *International Telecommunication Union (ITU)* [1] a quantidade de pessoas acessando a internet em 2006 era de aproximadamente 1 bilhão enquanto que em 2023 esse número cresceu para mais de 5 bilhões, como indica o gráfico da Figura 1.1:

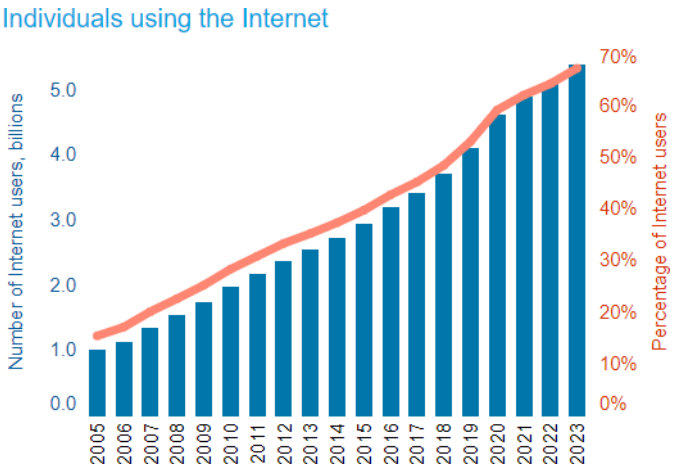


Figura 1.1: Número de Pessoas Utilizando Internet [1]

Esse crescimento não apenas reflete um avanço da tecnologia e da capacidade de globalização que ela proporcionou para a sociedade, mas também demarca uma nova era onde a informação se tornou um recurso extremamente abundante, entretanto, toda essa benesse traz consigo um cenário bastante desafiador de gerenciamento.

A percepção de impunidade digital pode estar relacionada à sensação de anonimato proporcionada pelo ambiente online. A internet oferece uma plataforma na qual, devido à relativa facilidade de ocultar identidades, alguns indivíduos podem se sentir protegidos ao praticar atividades ilegais ou antiéticas. Essa percepção pode contribuir para um aumento nos crimes virtuais, como fraudes, invasões de privacidade,

disseminação de *malware* e de informações falsas que geralmente é facilitada pela dificuldade em rastrear a verdadeira origem dos conteúdos na Internet e isso pode criar um ambiente propício para a propagação de desinformação.

A internet, como uma rede global de comunicação, desempenha um papel central na propagação dos dados. Informações podem atravessar fronteiras quase que instantaneamente, alcançando audiências globais em questão de segundos. Os benefícios da disseminação de todo esse acervo de dados se mostra bastante evidente. Todavia, esse cenário também corrobora com a potencialização da manipulação de informações, introduzindo o conceito de notícia falsa.

De acordo com CORNER et al [2] uma notícia falsa é um tipo de produção de mídia fraudulenta no qual o julgamento negativo e a intenção oculta é ainda mais forte que o viés da propaganda. Já para GELFERT [3], são apresentações deliberadamente de alegações falsas ou enganosas de notícias. Para simplificar o objeto de estudo deste trabalho, consideraremos notícia falsa, como informações falsas compartilhadas não apenas de maneira deliberada, mas também de maneira não intencional.

A disseminação desenfreada de notícias falsas tem se revelado como um fator de impacto significativo em debates importantes, como nas eleições brasileiras e norte-americanas. No contexto das eleições brasileiras, a influência de notícias falsas foi perceptível, afetando a percepção pública e, em certo grau, distorcendo o debate político. Foram feitos estudos pelos professores Pablo Ortellado e Fabrício Benvenuto, juntamente com Cristina Tardágia da agência de checagem de fatos Lupa, em uma pesquisa com 357 grupos públicos no *Whatsapp*. O estudo, intitulado "Eleição sem Fake" [4], revelou que apenas 8% das imagens mais compartilhadas nesses grupos poderiam ser consideradas verdadeiras. O objetivo da pesquisa era analisar a desinformação e a propagação de mensagens falsas durante o período eleitoral no Brasil. A amostra consistiu nos 357 grupos monitorados, reunindo cerca de 18 mil usuários que compartilharam 846 mil mensagens durante o primeiro turno das eleições. Embora os resultados não possam ser generalizados, forneceram indícios importantes sobre o fenômeno da disseminação de conteúdo falso no *Whatsapp.*

De maneira similar, as eleições norte-americanas também enfrentaram os desafios impostos pelas notícias falsas, com a manipulação de informações desempenhando um papel intrusivo. O *Facebook* admitiu que cerca de 126 milhões de seus usuários foram

expostos a postagens da Internet *Research Agency*, uma empresa ligada ao Kremlin, durante as eleições presidenciais. Essa quantia representa aproximadamente um terço da população dos Estados Unidos. Paralelamente, o X[1] identificou 3.814 contas envolvidas nessa atividade. As agências de espionagem dos Estados Unidos acusam diretamente Moscou de coordenar um plano abrangente, envolvendo a invasão de e- mails dos democratas[2], disseminação de notícias falsas e propaganda para favorecer a eleição de Donald Trump em detrimento de Hillary Clinton [5]. Os impactos das notícias falsas não se limitam apenas ao cenário político, estendendo-se a esferas sociais e econômicas. A propagação irresponsável de informações falsas pode alimentar teorias conspiratórias, prejudicar reputações e até mesmo incitar ações prejudiciais, seja no âmbito individual ou social.

A utilização da inteligência artificial (IA) tem a capacidade de exercer um papel crucial na identificação de notícias falsas, representando uma abordagem inovadora para mitigar os desafios associados à disseminação de informações enganosas. Uma das ferramentas amplamente empregadas nesse contexto é o pacote *scikit-learn* em *Python*, que oferece funcionalidades essenciais para implementar modelos de aprendizado de máquina [6].

No processo de detecção de notícias falsas, a preparação do texto é fundamental. Isso envolve desde a limpeza do conteúdo textual até a transformação em representações numéricas, como o uso do TF-IDF (*Term Frequency - Inverse Document Frequency*) [7]. Em seguida, algoritmos de aprendizado supervisionado, como *Naive Bayes* [8], *Support Vector Machines* (SVM) [9] e modelos de *Decision Tree* [10] [28], são treinados com esses conjuntos de dados tratados, permitindo uma análise mais eficiente.

Para abordagens mais avançadas, redes neurais, especialmente modelos de processamento de linguagem natural como o BERT [11], são empregadas para capturar nuances semânticas complexas no texto, aprimorando a capacidade de discernir notícias falsas com base em contextos mais sofisticados. Além disso, técnicas de detecção de anomalias, como o uso de *Isolation Forests* [12], são aplicadas para identificar padrões incomuns no comportamento textual, proporcionando uma camada adicional de detecção. A avaliação do desempenho dos modelos é geralmente conduzida através de métricas como precisão, recall e F1-score, utilizando validação cruzada para garantir robustez em diferentes conjuntos de dados [13].

[1] Rede social, antigo *Twitter*.
[2] No contexto eleitoral dos EUA, "democrata" refere-se a membros ou apoiadores do Partido Democrata, que geralmente defendem políticas progressistas e maior intervenção do governo.

O propósito desta pesquisa é explorar como a inteligência artificial (IA) pode ser efetivamente empregada para combater os impactos da disseminação de notícias falsas. Para atingir esse objetivo, será conduzida uma análise de técnicas de aprendizado de máquina supervisionado, visando a detecção de notícias falsas. A pesquisa utilizará duas bases de dados distintas contendo notícias em português. Esses conjuntos de dados serão empregados como entrada para quatro algoritmos de aprendizado supervisionado: árvore de decisão, florestas aleatórias, regressão logística e *gradient boosting*.

O resultado esperado desta pesquisa é obter e comparar a eficácia dos algoritmos frente as duas bases de dados, além de promover uma compreensão do papel desempenhado pela IA na contenção da disseminação de notícias falsas. A abordagem adotada se concentrará nos aspectos fundamentais envolvidos nos processos de aprendizado de máquina supervisionado, como o pré-processamento e a vetorização do texto além da implementação dos algoritmos de classificação e a análise dos resultados.

O texto a seguir está organizado da seguinte forma: na Seção 2, faz-se um referencial teórico da área de aprendizado de máquina treinado. Na Seção 3, por sua vez, faz-se uma breve descrição de elementos de processamento em linguagens naturais. Em seguida, na Seção 4, descrevem-se a metodologia experimental e os resultados obtidos. Finalizando o texto, as conclusões e sugestões de aprimoramento estão relatadas na Seção 5.

2. REVISÃO BIBLIOGRÁFIC A

2.1. Aprendizado de Máquina

Uma das áreas mais notáveis dentro do mundo da Inteligência Artificial (IA) é o Aprendizado de Máquina (*Machine Learning*). Este campo envolve sistemas que são capazes de aprender de maneira autônoma a partir de dados fornecidos e com isso tomar suas próprias decisões com o mínimo de intervenção humana [14]. Este modelo é fundamentado no paradigma do aprendizado indutivo, no qual modelos matemáticos são desenvolvidos com base em inferências lógicas [15], permitindo assim a capacidade de identificar e aprender padrões bem como generalizar conceitos de interesse específico. O aprendizado indutivo adere a uma categorização hierárquica [16], esquematizada pela Figura 2.1:

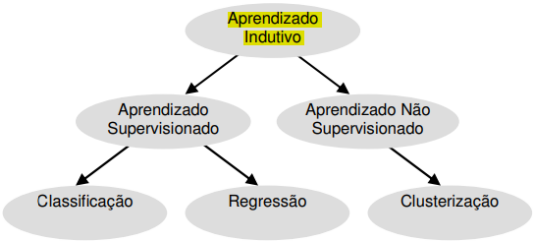


Figura 2.1: Categorias do Aprendizado de Máquina [16]

Nesse contexto, o aprendizado indutivo é conduzido mediante um conjunto de exemplos de treinamento cujos resultados são conhecidos, constituindo a chamada "base rotulada" [17]. Em outras palavras, o algoritmo é alimentado com uma base de treinamento contendo variáveis explicativas e a variável resposta, devidamente rotulada para cada cenário, com o propósito de permitir que o algoritmo treinado generalize e abstraia esse conhecimento para casos não rotulados [17]. Dentro do aprendizado supervisionado, há uma distinção entre problemas de classificação e problemas de regressão. A principal diferença entre esses dois tipos é que problemas do tipo classificação têm como objetivo classificar com variáveis discretas (ex.: dado e-mail é ou não *spam*), enquanto que problemas do tipo regressão estimam valores contínuos (ex.: preço de imóveis, cotação do Dólar, etc.) [18].

Por outro lado, o aprendizado de máquina não supervisionado usa algoritmos para analisar e agrupar dados não rotulados, ou seja, sem a necessidade de um conjunto de treinamento pré-definido. Esses algoritmos podem descobrir padrões ocultos ou

agrupamentos de dados sem a intervenção humana, sendo úteis para análise exploratória de dados [19].

2.2. Abordagens para Validação do Modelo de Aprendizado Supervisionado

Neste trabalho empregaram-se técnicas de abordagem supervisionada que, conforme explicado anteriormente, fundamentam-se na capacitação de um algoritmo com dados previamente rotulados. Essa metodologia pode ser aplicada a diversos problemas, especialmente aqueles que envolvem a previsão de uma variável categórica, neste trabalho a categorização se divide entre notícia falsa e verdadeira.

A validação efetiva de algoritmos de aprendizado supervisionado é essencial para garantir seu desempenho e generalização em situações do mundo real. Uma prática comum nesse contexto é a divisão do conjunto de dados rotulados em conjuntos de treinamento e validação, também conhecida como validação cruzada [20]. Nesse método, o algoritmo é treinado utilizando o conjunto de treinamento, enquanto o conjunto de validação é reservado para avaliar o desempenho por meio de métricas de erro.

Para conjuntos de dados de grande escala, a validação cruzada pode ser aprimorada pela introdução de um terceiro conjunto denominado conjunto de teste. Nesse cenário, o conjunto de treinamento mantém sua função original, enquanto o conjunto de validação é direcionado para avaliar o desempenho, orientando escolhas de algoritmos, seleção de atributos e otimização de hiperparâmetros. O conjunto de teste, por sua vez, é crucial para a avaliação do desempenho do algoritmo escolhido em um contexto mais próximo do real, sendo testado com exemplos inéditos [17].

Ao treinar o algoritmo com uma a base de dados, o resultado do ajuste pode ser dividido em três grupos principais, conforme ilustra a figura 2.2:

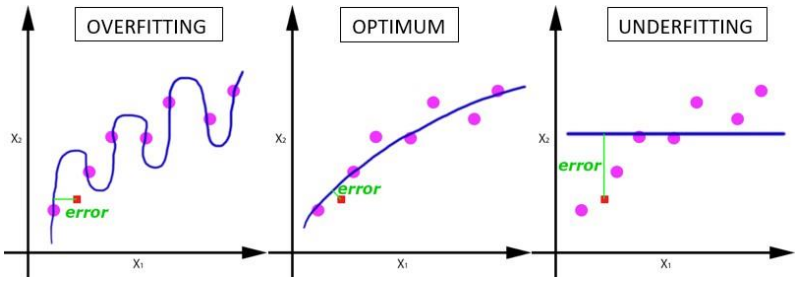


Figura 2.2: Possibilidades de Ajuste de Curva para Algoritmos Supervisionados [21]

O *underfitting* ocorre quando um modelo é muito simples para capturar a complexidade dos dados de treinamento. Nesse cenário, o modelo não tem êxito em se ajustar adequadamente aos padrões subjacentes dos dados, resultando em um desempenho insatisfatório. Evitar o *underfitting* é crucial para garantir que o modelo possa representar adequadamente a complexidade dos dados durante o treinamento [22].

Em contrapartida, no *overfitting* o modelo se ajusta excessivamente aos dados de treinamento, capturando até mesmo o ruído presente nos dados, seria como se o algoritmo decorasse os dados de treinamento. Isso pode levar a uma perda significativa de generalização, uma vez que o modelo se torna muito específico para o conjunto de treinamento, prejudicando seu desempenho em dados não vistos. Mitigar o *overfitting* é essencial para garantir que o modelo seja capaz de generalizar para novos dados [23].

Encontrar o ponto ótimo de desempenho é uma busca pela configuração ideal do modelo, onde ele será capaz de generalizar adequadamente, evitando tanto o *underfitting* quanto o *overfitting*. Este ponto ótimo é alcançado através da cuidadosa seleção de algoritmos, com um ajuste de hiperparâmetros e escolha de atributos relevantes. A Validação Cruzada desempenha um papel crucial nesse processo, ao dividir os conjuntos de dados de treinamento, validação e teste, permitindo uma boa avaliação do desempenho do modelo [24].

Portanto, observando os cenários que um modelo pode aderir e a maneira como o mesmo interage com o conjunto de dados, podemos encarar a sua otimização como um problema matemático no qual a busca é feita para encontrar o menor erro possível proveniente da interação do modelo com o conjunto de teste/validação [25].

Nos problemas de classificação é possível fazer uma avaliação da frequência com que cada classe obteve uma predição correta, introduzindo uma distinção binária, visto que o valor da predição só pode assumir dois valores, sucesso ou falha. Nesse contexto a matriz de confusão serve como complemento dessa métrica, pois a mesma descreve a distribuição das predições corretas e incorretas para cada classe, delineando elementos como verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos [14].

		Valor Predito	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Figura 2.3 Matriz de Confusão [26]

Embasado nesses valores é possível estabelecer algumas métricas importantes para o processo de avaliação, como a acurácia (*acc*), precisão (*prel*), recall (rec) e F1 *score* (*f_score*), definidas como [26]:

:

$$acc = \frac{TP+TN}{TP + TN + FP + FN}, \tag{3.1}$$

$$prel = \frac{TP}{TP + FP}, \tag{3.2}$$

$$rec = \frac{TP}{TP + FN}, \tag{3.3}$$

$$f\_score = \frac{2 * TP}{2 * TP + FP + FN}. \tag{3.4}$$

A acurácia informa a proporção de previsões corretas em relação ao total de previsões realizadas, é uma métrica comum em problemas do tipo classificação, porém ela não é muito precisa quando existe um desequilíbrio entres os dados das classes. Já a precisão indica a parcela de previsões positivas corretas sobre o total de previsões positivas feitas pelo modelo. Pode ser útil quando a intenção é diminuir a previsões falso positivas. Por outro lado, o *recall* representa a proporção entre as previsões positivas que foram corretas em relação ao total de instâncias positivas. Sua função é minimizar os falsos negativos. Finalmente, o F1 *Score,* é uma métrica que une a precisão e o recall em uma única medida, sendo uma média harmônica delas. Seu uso estimula o equilíbrio entre as duas medidas [27].

Já para problemas do tipo regressão, o dado que se quer prever é um dado contínuo, ou seja, um número qualquer, portanto utilizar a frequência de acerto não é a maneira mais eficiente de avaliar o modelo. Pode-se então estudar a distância média entre um conjunto de *n* objetos observados do valor original. Para isso utiliza-se algumas métricas como o erro quadrático absoluto (MAE), o erro médio quadrático (MSE) e o coeficiente de determinação (R^2) [29].

O erro médio absoluto (MAE) mede a média da diferença entre o valor observado e o valor original, essa definição não é afetada por valores que ficam muito distantes do original, conhecidos como *outliers* [30]. É definido pela seguinte equação:

$$MAE = \frac{1}{n} \sum_{i=0}^{n} |y_i - \hat{y}|. \quad (3.5)$$

Assim como o MAE, o erro quadrático médio (MSE) calcula a média entre o valor observado e o valor original [29], porém ao invés de utilizar o módulo da diferença usa-se o resultado da diferença elevado ao quadrado. Nessa métrica, dados fora da curva (*outliers*) exerce bastante influência no resultado, portanto valores altos de MSE indicam que o modelo não obteve uma boa performance [30]. Sua definição segue a seguinte equação:

$$MSE = \frac{1}{n} \sum_{i=0}^{n} (y_i - \hat{y})^2. \quad (3.6)$$

Já o R^2 é o percentual da variância dos dados que é explicado pelo modelo. Seu valor é dado por um percentual que varia de 0% a 100%, sendo 100% a representação de um modelo perfeito que consegue explicar todos os dados previstos [30]. É definido pela equação a seguir:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y - \hat{y})^2}{\sum_{i=1}^{n} (y - \bar{y})^2}. \quad (3.7)$$

Sendo y_i o i-ésimo valor do conjunto dos valores reais, $\hat{y}$ o i-ésimo valor do conjunto dos valores observados e $\bar{y}$ a média dos valores reais.

2.3. Algoritmos de Aprendizado de Máquina

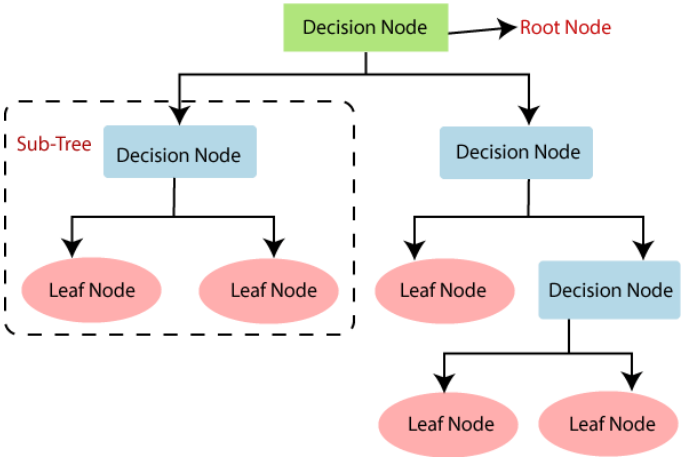
As técnicas de aprendizado de máquina não são parametrizadas, isso significa que elas não assumem uma forma específica para a função que mapeia os dados de entrada para as saídas desejadas, portanto isso implica que não é necessário que seja definida uma função matemática para representação dos dados. Em vez disso são usados algoritmos que utilizam os dados disponíveis para ajustar o modelo, permitindo que se identifique padrões complexos [31].

2.3.1. Árvores de Decisão

Uma árvore de decisão é um modelo de aprendizado de máquina que representa uma estrutura hierárquica de decisões lógicas para classificar ou regredir dados. No contexto de classificação, a árvore de decisão divide iterativamente o conjunto de dados com base em características específicas, criando ramos que representam diferentes caminhos de decisão [32].

Durante o processo de ajuste da árvore de decisão, são identificados os atributos mais eficazes para segmentar os dados, visando minimizar erros. A profundidade da árvore refere-se à quantidade de níveis nos quais os dados podem ser divididos na mesma, quanto maior a profundidade, maior a complexidade do modelo. Cada nó representa o resultado de uma separação determinada pela decisão tomada no nível imediatamente superior da árvore. As folhas indicam um nó terminal que não implica mais decisões, representando assim conjuntos de dados com características semelhantes. Por fim, um ramo corresponde ao conjunto de decisões subjacentes a um determinado nó [28].

Figura 2.4: Esquematização do Algoritmo de Árvore de Decisão [44]



Esse algoritmo oferece vantagens notáveis, incluindo facilidade de interpretação e baixa complexidade computacional durante a utilização. Além disso, ele é versátil, sendo aplicável tanto a dados qualitativos quanto quantitativos. No entanto, como desvantagem, há a possibilidade de as árvores se tornarem excessivamente grandes e complexas. Para mitigar esse problema, é necessário realizar podas utilizando critérios como o tamanho da árvore ou ganho de informação, entre outros, a fim de evitar o *overfitting* [17].

2.3.2. Florestas Aleatórias

Árvores de decisão tem como característica serem menos complexas, todavia, quando comparadas a outros modelos, geralmente possuem um poder preditivo menor [33]. Diante desse obstáculo, uma forma de melhorar sua capacidade de predição é utilizando este modelo de maneira conjunta e coordenada, lançando mão do uso de uma técnica conhecida como essemble. Esta técnica é uma combinação de vários modelos individuais que juntos formam um modelo mais robusto e assertivo.

No caso específico das florestas aleatórias, o essemble é composto por um conjunto de árvores de decisão. Cada árvore é treinada de forma independente, geralmente usando uma subamostra aleatória dos dados de treinamento e uma seleção aleatória de características em cada divisão da árvore. As previsões individuais das árvores são então combinadas para obter uma previsão final [28].

A principal ideia por trás do ensemble nesse algoritmo é reduzir o *overfitting*, explorando a diversidade das árvores. Cada árvore pode se especializar em diferentes aspectos dos dados, e a combinação delas ajuda a criar um modelo mais estável e preciso, melhorando assim a generalização para novos dados [34].

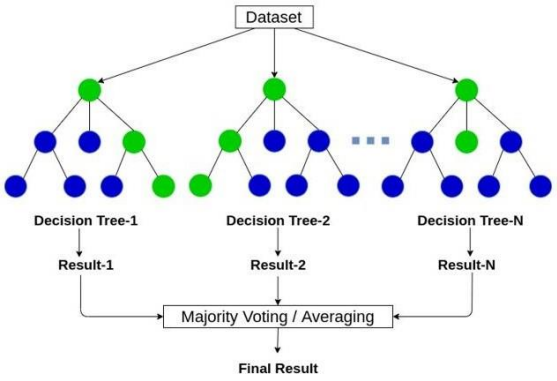


Figura 2.5: Esquematização do Modelo de Florestas Aleatórias [45]

2.3.3. Gradient Boosting

Assim como o algoritmo de florestas aleatórias, o G*radient Boosting* utiliza o método de essemble para combinar modelos de predição em um único modelo. Segundo HASTIE, T. et al. [36] *boosting* é uma das mais poderosas ideias dos últimos vinte anos. Essa técnica, como observado, também unifica outros modelos mais fracos, todavia essa unificação é feita de maneira sequencial, onde a ideia é treinar um primeiro modelo com os dados disponíveis e na sequência um outro modelo busca corrigir os erros observados na iteração anterior [28].

De maneira resumida, o algoritmo inicia com um modelo que serve como base, geralmente uma árvore de decisão simples. Este modelo é ajustado aos dados de treinamento, mas, frequentemente, não é muito preciso em suas previsões. Os erros, ou resíduos, desse modelo são calculados. Em seguida, aplicamos o conceito de gradiente descendente, onde um novo modelo é ajustado para prever esses resíduos, tentando minimizar o erro. Este novo modelo é combinado com o modelo anterior, dando mais peso aos modelos que contribuem mais para a melhoria global do desempenho [35] [36].

Esse processo é repetido, a cada iteração um novo modelo é ajustado para prever os resíduos acumulados dos modelos anteriores, e a combinação de modelos é atualizada. Ao final de várias iterações, obtemos um modelo final que é uma combinação ponderada dos modelos mais simples. Essa abordagem de construir modelos sequencialmente, corrigindo os erros do modelo anterior, permite que o *Gradient Boosting* aprenda padrões complexos nos dados e crie modelos altamente precisos [35] [36].

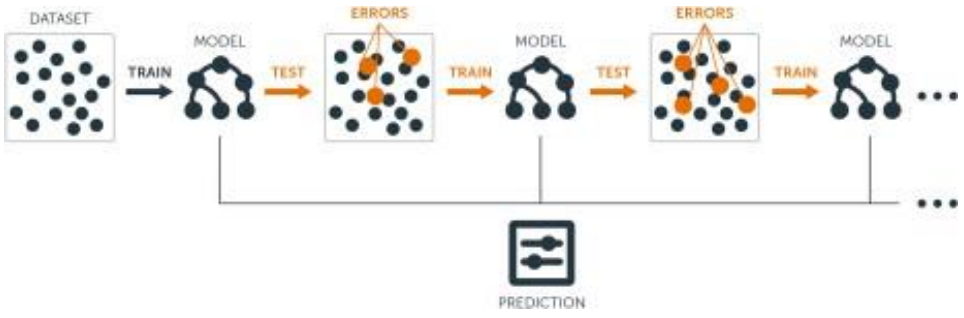


Figura 2.6: Esquematização do Modelo *Gradient Boosting* [46]

2.3.4. Regressão Logística

A regressão logística é um modelo estatístico usado em aprendizado supervisionado para prever a probabilidade de ocorrência de um evento binário. Em outras palavras, ela é comumente utilizada quando a variável de saída é de natureza categórica binária, onde os dois valores possíveis geralmente representam duas classes [17], no caso deste trabalho fake*/true.*

A principal diferença entre a regressão logística e a regressão linear é a função de ajuste utilizada. Na regressão logística, essa função é logística, que transforma a combinação linear das variáveis independentes em uma probabilidade entre 0 e 1. A função logística (também chamada de *Sigmoid*) é definida como [36]:

$$f(x) = \frac{1}{1+ e^{-x}}, \tag{4.1}$$

sendo x a combinação linear das variáveis independentes e seus pesos associados.

Para interpretar o resultado da regressão logística pode-se utilizar o conceito de fronteira de decisão, onde é escolhido um valor limite para decidir qual classe àquele resultado pertence. Por exemplo, se o valor de fronteira for 0.5, a classificação segue [36]:

$$g(x) = \begin{cases} 0, & x < 0.5 \\ 1, & x \geq 0.5 \end{cases}. \tag{4.2}$$

O treinamento da regressão logística envolve a maximização da verossimilhança (*likelihood*) dos dados observados, ajustando os coeficientes para maximizar a probabilidade de observar os rótulos reais dados os valores previstos, sendo a verossimilhança uma medida estatística que expressa quão provável é um conjunto específico de observações sob um modelo estatístico particular [37].

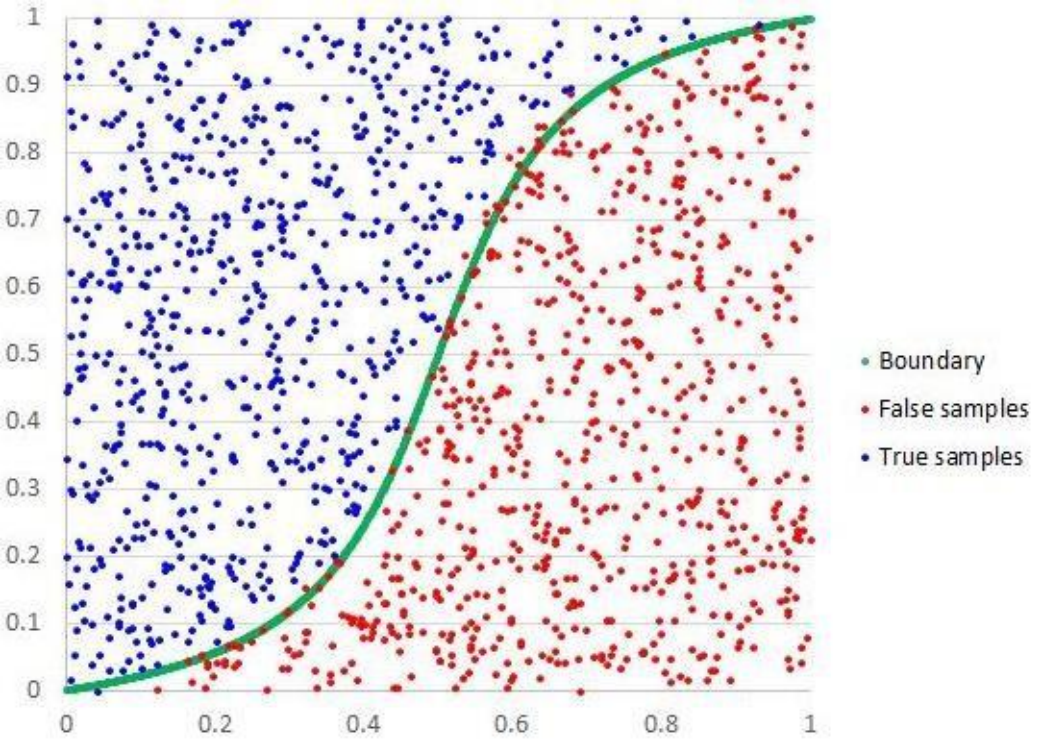


Figura 2.7 – Curva de Ajuste do Modelo de Regressão Logística [47]

3. ELEMENTOS DE PROCESSAMENTO EM LINGUAGEM NATURAL

O ingrediente indispensável para todas as abordagens que vimos até agora é a base de dados, já que ela que alimenta todos os modelos de aprendizado de máquina. Os algoritmos que tentam prever algum resultado se baseiam nos dados de entrada, promovem alguma manipulação nos mesmos e aplicam operações algébricas de modo a chegar no resultado mais provável.

Considere, por exemplo, a regressão logística para o caso binário. Assume-se que a variável de interesse y_i (previsão) apresenta-se em apenas dois valores {0,1} no ponto i. Portato, resumidamente, o método de regressão logística calcula a probabilidade da classe positiva ocorrer dado X_i , onde [38]:

$$P(y_i = 1 \,|X_i), \tag{5.1}$$

na qual,

$$p(X_i) = \frac{1}{1+ e^{-X_i * w - w0}} . \tag{5.2}$$

Em problemas de previsão nos quais os dados que alimentam o modelo são compostos em sua natureza por números, existe a facilidade de poder utilizá-los de diretamente, isto é, sem nenhum artifício intermediário para representá-los de modo que o algoritmo consiga interpretá-lo. Entretanto, quando a natureza dos dados que compõem a base não é numérica é necessário representar esse conjunto de forma numérica, para permitir a aplicação de expressões numéricas. Além disso, nem todos os dados contidos na base de treinamento são relevantes para extrair alguma informação, ou seja, existem partes do texto que não contribuem com a identificação de padrões que são necessários para treinar o modelo de aprendizado. Portanto, outro aspecto importante no pré-processamento é justamente promover uma seleção nos dados.

3.1. Tokenização

Na literatura utiliza-se o termo documento para se referir a subdivisões de texto. Essa divisão pode ser feita de diversas maneiras, em que uma unidade de documento pode representar um texto inteiro, um parágrafo, uma frase, um bloco de texto, dentre outras possibilidades dependendo do nível de granularidade desejado. A tokenização é a tarefa de dividir essa unidade de documento em pequenas partes chamadas de *token* [39], como pode ser observado na figura 5.1:

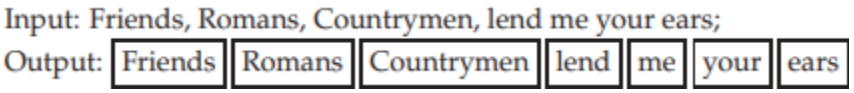


Figura 3.1: Processo de Tokenização [39]

Ao dividir um texto em *tokens* o trabalho feito em processos de análises sintáticas e semânticas se torna uma tarefa mais simples. Essas análises dependem da maneira como as palavras se relacionam entre si, o que é facilitado quando o texto está dividido em partes menores.

3.2. Remoção de palavras comuns: *Stop Words*

Outra ferramenta importante no pré-processamento de dados é a remoção de palavras comuns, conhecidas como *stop words.* Essas palavras podem ser artigos no texto, preposições e conjunções, embora sejam importantes para manter o texto fluído na linguagem humana, muitas vezes elas não representam relevância na identificação de padrões ou extração de informações em tarefas específicas no processamento de linguagem natural [42].

O método predominante para aplicar essa abordagem envolve a criação de uma lista de palavras ordenadas pela frequência com que aparecem em um determinado conjunto de dados. Essa lista é geralmente submetida a uma filtragem manual, considerando o valor semântico de cada termo. Em seguida, as *stop words* identificadas são descartadas durante o processo de indexação, contribuindo assim para um conjunto de dados mais refinado e concentrado nas palavras que verdadeiramente carregam significado e relevância para a tarefa em questão [39].

3.3. Lematização

A lematização é uma técnica que visa reduzir as palavras à sua forma base ou raiz, é fundamental nesse contexto. Ao realizar a lematização, é possível agrupar diferentes formas flexionadas de uma palavra em uma única representação, reduzindo assim a dimensionalidade do espaço de características e proporcionando uma visão mais abstrata do texto [39].

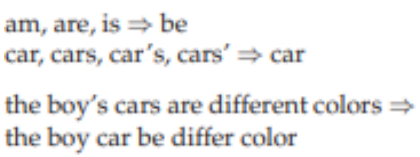


Figura 3.2: Exemplo de Lematização para o Idioma Inglês [39]

A lematização contribui para melhorar a generalização do modelo, permitindo que ele identifique padrões e relações semânticas de maneira mais robusta, mesmo diante de variações linguísticas.

Essa abordagem tem sido amplamente adotada em diversas aplicações de processamento de linguagem natural (PLN) e aprendizado de máquina. Estudos como o de Manning et al. [43] destacam a importância da lematização no pré-processamento de texto para tarefas como classificação de documentos. Ao fornecer uma representação mais coesa das palavras, a lematização auxilia na captura de informações semânticas e na melhoria do desempenho dos modelos de aprendizado de máquina em diversas áreas, como análise de sentimentos, categorização de notícias e identificação de tópicos, fortalecendo assim a capacidade dos modelos de generalizar conhecimento a partir dos dados de treinamento.

3.4. TF-IDF

Para se entender o conceito de TF-IDF (*Term Frequency – Inverse Document Frequency*) é necessário explorar alguns outros conceitos importantes nessa construção. Podemos analisar o termo como sendo uma palavra específica dentro do documento, revisitando a Seção 5.1, termo seria o *token* que se encontra dentro do texto que foi designado como documento. A frequência do termo, como o nome indica é uma pontuação dada ao termo baseado na quantidade de ocorrências do mesmo dentro de seu documento [39], sendo definida como [40]:

$$tf(t,d) = \frac{número\ de\ termos\ t\ no\ documento\ d}{número\ total\ de\ termos\ no\ documento\ d}. \tag{5.3}$$

Inverse Document Frequency é um mecanismo utilizado para atenuar o efeito de termos que são muito frequentes em uma certa coleção, reduzindo o peso da frequência do termo por um fator que é diretamente proporcional a essa frequência. A ideia aqui é atribuir um peso maior a termos que são considerados mais informativos, ou seja, termos que aparecem em poucos documentos do conjunto total dando mais importância a palavras que são mais exclusivas e menos comuns em todo o conjunto de documentos [39], sua definição segue a seguinte equação [40] [41]:

$$idf(t) = \log \frac{N}{df(t)+1}, \tag{5.4}$$

onde *N* é o número total de documentos e o *df(t*) representa o número de documento onde o termo *t* é presente.

Unindo os dois conceitos, frequência de termo e frequência inversa de documento podemos definir o TF-IDF como a multiplicação entre os dois, ou seja, o tf-idf de um termo *t* dentro de um documento *d* vai ser [39]:

- Maior quando *t* aparece muitas vezes dentro de um pequeno número e documentos;
- Menor quando o *t* ocorre poucas vezes no documento ou aparece em muitos documentos;
- Menor quando o *t* está presente em todos documentos.

Contudo podem-se interpretar os documentos como vetores com uma componente que corresponde a cada termo presente junto da pontuação que provém do TF-IDF. Portanto utilizando este artificio é possível transformar textos em matrizes de números, algo concreto que pode ser utilizado nas expressões algébricas que constituem os inúmeros algoritmos de aprendizado de máquina.

4. METODOLOGIA EXPERIMENTAL E RESULTADOS

Este capítulo descreve detalhadamente como os algoritmos foram implementados, explicando o processo de leitura e preparação dos conjuntos de dados, descrevendo as etapas de importação e organização dos dados em estruturas de dados tabulares, utilizando a biblioteca *Pandas*. No tocante ao pré-processamento dos dados textuais, foram aplicadas diversas técnicas, como a remoção de caracteres especiais, URLs e tags HTML, bem como a normalização de texto por meio da conversão para minúsculas e a eliminação de pontuações e números não relevantes para a análise. Ademais, o processo incluiu a lematização dos termos textuais, contribuindo para a redução da dimensionalidade e a melhoria da generalização dos modelos. No que tange aos algoritmos de aprendizado de máquina, a implementação foi conduzida por meio da biblioteca scikit-learn, que oferece uma ampla gama de algoritmos e ferramentas para modelagem e avaliação de desempenho. Entre os algoritmos explorados estiveram o *DecisionTreeClassifier*, *LogisticRegression*, *GradientBoostingClassifier* e *RandomForestClassifier*, cada um com ajustes específicos de hiperparâmetros visando otimizar o desempenho dos modelos. A avaliação dos modelos foi realizada utilizando métricas de desempenho, como acurácia, precisão, recall e pontuação F1, fornecendo uma visão abrangente do comportamento e da eficácia de cada algoritmo na classificação de notícias. Os resultados obtidos foram interpretados com base nessas métricas, permitindo uma análise aprofundada das vantagens e limitações de cada modelo em relação à detecção de notícias falsas. A figura 4.1 descreve as etapas da metodologia em um formato de diagrama de blocos.

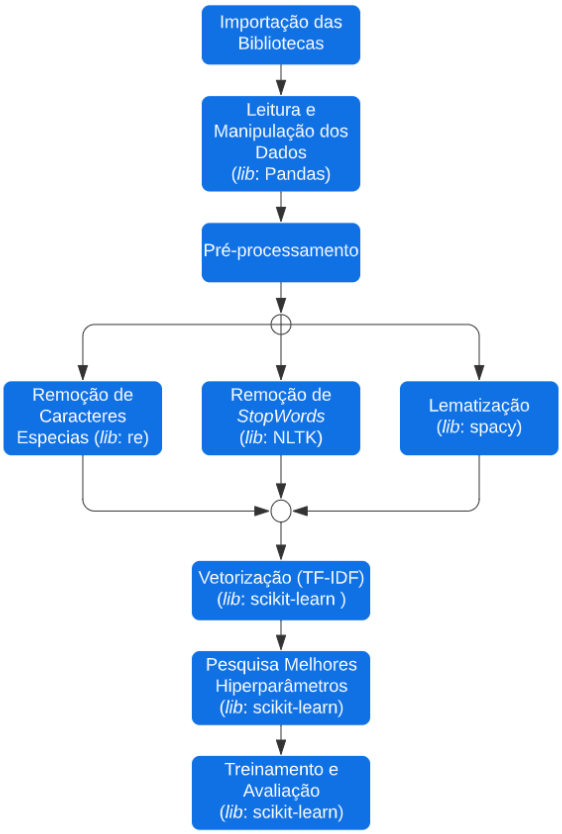


Figura 4.1: Fluxograma dos processos envolvidos no algoritmo.

4.1. Conjunto de Dados Utilizados

A proposição deste trabalho foi explicar alguns modelos de algoritmo de aprendizado de máquina e avaliar o desempenho dos mesmos para classificar uma determinada notícia como verdadeira ou falsa. Os dados utilizados estão disponíveis em [48] e [49]. Ambas as bases foram tratadas de modo a disponibilizar os mesmos parâmetros, sendo uma variável com o rótulo da classe, *true* para notícias verdadeiras e *fake* para notícias falsas.

A base disponibilizada em [48] está balanceada, sendo 3600 linhas contendo notícias verdadeiras e outras 3600 linhas com o teor de notícias falsas. Já a base encontrada em [49] possui uma quantidade menor de dados, também estando balanceada, com 221 registros de notícias verdadeiras e outros 238 registros com valores de notícias falsas.

4.2. Bibliotecas Utilizadas

4.2.1. Pandas

Essa é uma biblioteca de Python especializada em análise de dados. Neste trabalho, foi empregada para a leitura e manipulação de conjuntos de dados, utilizando estruturas de dados conhecidas como *DataFrames*. Um *DataFrame* é uma estrutura tabular bidimensional que organiza os dados em linhas e colunas, oferecendo uma forma intuitiva e eficiente de representar informações. As operações realizadas incluíram a concatenação dos dados, remoção de algumas colunas específicas e o embaralhamento dos dados para garantir uma análise imparcial. Essa funcionalidade foi fundamental para a manipulação eficaz dos conjuntos de dados utilizados.

4.2.2. Scikit-learn:

A biblioteca *scikit-learn* em Python é uma ferramenta essencial no campo de aprendizado de máquina e ciência de dados. Seu propósito principal é fornecer uma ampla gama de algoritmos de aprendizado de máquina, técnicas de pré-processamento de dados e ferramentas de avaliação de modelos, tudo integrado em uma interface consistente e fácil de usar. Os principais recursos utilizados desta biblioteca neste trabalho foram:

- train_test_split

É uma função da biblioteca *sklearn.model_selection* que é utilizada para dividir o conjunto de dados em conjuntos menores para serem utilizados no treinamento e teste.

- RandomizedSearchCV

Também faz parte do módulo *sklearn.model_selection,* é uma classe que realiza uma busca aleatoriamente pelos melhores hiperparâmetros sendo útil para otimizar os algoritmos na busca pelas melhores perdições.

- accuracy_score

Função da biblioteca *sklearn.metrics* que calcula a acurácia do modelo, sendo a acurácia a proporção de classificações corretas em relação ao total de classificações.

- classification_report

É também uma função da biblioteca *sklearn.metrics,* serve como um complemento de avaliação do modelo, essa função gera um relatório com métricas de classificação como precisão, *recall,* f1-score e *support* para cada classe.

- TfidfVectorizer

Da biblioteca *sklearn.feature_extraction.text,* é utilizado para converter coleções de documentos (subdivisões do texto) de um texto em representações numéricas usando o cálculo de TF-IDF.

- Pipeline

Pertence a biblioteca *sklearn.pipeline,* permite encadear várias etapas de processamento de dados e modelagem em um único estimador, foi utilizado para encontrar os melhores hiperparâmetros.

- ColumnTransformer

É uma função da biblioteca *sklearn.compose,* facilita a aplicação de transformações específicas em colunas específicas dos dados, foi utilizado no pipeline.

- OneHotEncoder

Transformador da biblioteca *sklearn.preprocessing,* que converte variáveis categóricas em representações *one-hot,* que a grosso modo utiliza números para representar uma divisão categórica.

- DecisionTreeClassifier

Implemetação de árvores de decisão para classificação, disponível no módulo *sklearn.tree.*

- LogisticRegression

Implementação do algoritmo de regressão logística, pertence ao módulo *sklearn.linear_model*.

- GradientBoostingClassifier

Implementação do modelo classificador de *boosting* baseado em árvores de decisão e pertence ao módulo *sklearn.ensemble.*

- RandomForestClassifier

Implementação do classificador de floresta aleatória, também pertence ao módulo *sklearn.ensemble*.

4.2.3. Re e String

O módulo re é utilizado para manipulação de expressões regulares, no qual, foi empregado para a remoção de algumas partes do texto que a princípio não seriam significativas para o processo de detecção de padrões que é abordado com a técnica de aprendizado de máquina. Por exemplo, com essa biblioteca foi possível retirar pontuações, trechos do texto que continham URL, quebras de linha dentre outras variações textuais. O módulo string também fornece funcionalidades para a manipulação de textos, sendo utilizado neste contexto para a remoção de pontuações.

4.2.4. Spacy:

O spaCy é uma biblioteca avançada de processamento de linguagem natural (PLN). Neste trabalho, o modelo pt_core_news_sm do spaCy foi utilizado para a lematização do texto, contribuindo para a normalização linguística.

4.3. Descrição do Código

4.3.1. Importação das Bibliotecas

Inicialmente foram importadas todas as bibliotecas que disponibilizaram os recursos citados na seção anterior e que foram imprescindíveis para o desenvolvimento deste projeto. O IDE (*Integrated Development Environment*) utilizado neste projeto foi o *Visual Studio Code* que se trata de um editor de código-fonte de código aberto amplamente adotado por sua leveza e extensibilidade. Para importar uma biblioteca qualquer, primeiramente é necessário fazer a sua instalação, com o uso do *Visual Studio Code* é possível utilizar seu terminal para isso, como demonstrado na figura 4.2.

```
PS C:\Users\Matheus> pip install pandas
Requirement already satisfied: pandas in c:\users\matheus\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (2.1.4)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\users\matheus\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from pandas) (1.26.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\matheus\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\matheus\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\matheus\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from pandas) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\matheus\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)

[notice] A new release of pip is available: 23.3.2 -> 24.0
[notice] To update, run: C:\Users\Matheus\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip
```

Figura 4.2: Exemplo de importação de uma biblioteca pelo terminal do Visual Studio Code

4.3.2. Leitura e Manipulação dos Dados

Antes de começar a análise, é crucial preparar os dados adequadamente. Inicialmente, dois conjuntos de dados foram carregados, um contendo notícias falsas e outro contendo notícias verdadeiras. Esses dados foram então combinados em um único *DataFrame*, atribuindo a classe 0 para as notícias falsas e a classe 1 para as notícias verdadeiras. Além disso, colunas desnecessárias, como índice e rótulo, foram removidas para simplificar a estrutura do *DataFrame*.

4.3.3. Pré-processamento de Texto

O texto que contém as notícias passou por um processo abrangente de pré-processamento para torná-lo adequado para análise por algoritmos de aprendizado de máquina. Isso incluiu algumas etapas, como:

- Conversão de todas as letras para minúsculas para garantir consistência.
- Remoção de caracteres especiais, URLs e tags HTML, que não contribuem para o significado semântico das notícias.
- Eliminação de pontuações e números, pois geralmente não são relevantes para a classificação de texto.
- Remoção de palavras que não agregam para a diferenciação do texto (*stopwords*).
- Lematização das palavras, reduzindo-as às suas formas básicas, o que ajuda a reduzir a dimensionalidade do espaço de recursos e melhora a generalização do modelo.

4.3.4. Vetorização de Texto

Como os algoritmos de aprendizado de máquina requerem entradas numéricas, as notícias foram convertidas em vetores numéricos usando a técnica TF-IDF (*Term Frequency-Inverse Document Frequency*). Essa abordagem atribui pesos a cada palavra com base em sua frequência no documento analisado e em todo o corpo do texto, destacando a importância relativa das palavras nas notícias.

4.3.5. Divisão do Conjunto de Dados

Para avaliar adequadamente o desempenho dos modelos, é essencial dividir o conjunto de dados em conjuntos de treinamento e teste. Foi utilizada a função

train_test_split para realizar essa divisão. Foi escolhido uma proporção de 80% para treinamento e 20% para teste, garantindo que houvesse dados suficientes para treinar os modelos e avaliar sua capacidade de generalização.

4.3.6. Treinamento e Avaliação de Modelos

Nesta etapa, foram explorados alguns algoritmos de classificação para avaliar o desempenho de cada um no problema de detecção de notícias falsas. Para cada algoritmo, seguiram-se as seguintes etapas:

- Árvores de Decisão

Foi utilizada a classe *DecisionTreeClassifier* para treinar um modelo de árvore de decisão. Foi realizada uma pesquisa aleatória de hiperparâmetros para otimizar a profundidade da árvore, o número mínimo de amostras em folhas, entre outros parâmetros que são utilizados neste modelo.

- Regressão Logística

Utilizou-se a classe *LogisticRegression* para implementar o modelo e conduziu-se uma pesquisa aleatória de hiperparâmetros para ajustar alguns parâmetros como o de regularização (C), que controla a complexidade do modelo.

- Gradient Boosting

Utilizou-se a classe *GradientBoostingClassifier* para colocar o algoritmo em prática e também foi conduzida uma pesquisa aleatória de hiperparâmetros para encontrar a combinação ideal de número de estimadores, taxa de aprendizado e profundidade máxima das árvores, dentre outros argumentos do algoritmo.

- Random Forest

Utilizando a classe *RandomForestClassifier* foi implementado o modelo com os hiperparâmetros otimizados através de uma pesquisa aleatória.

4.3.7. Avaliação dos Modelos

Após o treinamento, foi avaliado o desempenho de cada modelo utilizando métricas como precisão, recall e pontuação F1. Além disso, foi calculada a acurácia geral dos modelos utilizando o conjunto de teste. Essas métricas nos permitem entender o desempenho dos modelos e identificar suas vantagens e limitações.

4.4. Resultados e Análise

Depois de treinados, cada modelo de algoritmo utilizado obteve um resultado que foi avaliado com o auxílio de ferramentas disponíveis na biblioteca *sklearn.metrics.*

As tabelas 4.1 a 4.8 mostram os resultados referentes à base de dados número 1 [48]:

Tabela 4.1: Hiperparâmetros para o modelo Árvore de Decisão e seu score

Arvore de Decisão		
Hiperparâmetros	Valor	Score do modelo
Splitter	best	**0.89**
min_samples_split	5	
min_samples_leaf	2	
max_depth	20	
Criterion	Gini	
random_state	42	

Tabela 4.2: Principais resultados para o modelo Árvore de Decisão

	Precision	***recall***	***F1-score***	***Support***
0	0.88	0.90	0.89	726
1	0.89	0.88	0.89	710
Accuracy	-	-	0.89	1436
macro average	0.89	0.89	0.89	1436
weighted average	0.89	0.89	0.89	1436

Tabela 4.3: Hiperparâmetros para o modelo Regressão Logística e seu score

Regressão Logística		
Hiperparâmetros	Valor	Score do modelo
Penalty	l2	**0.96**
C	985.4669612827593	
max_iter	200	

Tabela 4.4: Principais resultados para o modelo Regressão Logística

	Precision	***recall***	***F1-score***	***Support***
0	0.94	0.97	0.96	890
1	0.97	0.94	0.96	905
accuracy	-	-	0.96	1795
macro average	0.96	0.96	0.96	1795
weighted average	0.96	0.96	0.96	1795

Tabela 4.5: Hiperparâmetros para o modelo Gradient Boosting e seu score

Gradient Boosting		
Hiperparâmetros	Valor	Score do modelo
n_estimators	100	**0.96**
min_samples_split	2	
min_samples_leaf	1	
max_depth	4	
learning_rate	0.1	

Tabela 4.6: Principais resultados para o modelo Gradient Boosting

	Precision	***recall***	***F1-score***	***Support***
0	0.95	0.97	0.96	890
1	0.97	0.95	0.96	905
accuracy	-	-	0.96	1795
macro average	0.96	0.96	0.96	1795
weighted average	0.96	0.96	0.96	1795

Tabela 4.7: Hiperparâmetros para o modelo Florestas Aleatórias e seu score

Florestas Aleatórias		
Hiperparâmetros	Valor	Score do modelo
n_estimators	100	**0.93**
min_samples_split	5	
min_samples_leaf	1	
max_depth	50	
criterion	Gini	

Tabela 4.8: Principais resultados para o modelo Florestas Aleatórias

	Precision	***recall***	***F1-score***	***support***
0	0.92	0.98	0.95	890
1	0.98	0.92	0.95	905
accuracy	-	-	0.95	1795
macro average	0.95	0.95	0.95	1795
weighted average	0.95	0.95	0.95	1795

As tabelas 4.9 a 4.16, por sua vez, mostram os resultados referentes à base de dados número 2 [49]:

Tabela 4.9: Hiperparâmetros para o modelo Árvore de Decisão e seu score

Arvore de Decisão		
Hiperparâmetros	Valor	Score do modelo
splitter	Random	**0.57**
min_samples_split	10	
min_samples_leaf	4	
max_depth	10	
criterion	Entropy	
random_state	42	

Tabela 4.10: Principais resultados para o modelo Árvore de Decisão

	precision	***recall***	***F1-score***	***support***
0	0.57	0.88	0.69	74
1	0.61	0.22	0.32	64
accuracy	-	-	0.57	138
macro average	0.59	0.55	0.50	138
weighted average	0.59	0.57	0.52	138

Tabela 4.11: Hiperparâmetros para o modelo Regressão Logística e seu score

Regressão Logística		
Hiperparâmetros	Valor	Score do modelo
penalty	l2	**0.58**
C	660.3868601918183	
max_iter	200	

Tabela 4.12: Principais resultados para o modelo Regressão Logística.

	precision	***recall***	***F1-score***	***support***
0	0.58	0.77	0.66	74
1	0.57	0.36	0.44	64
accuracy	-	-	0.58	138
macro average	0.58	0.56	0.55	138
weighted average	0.58	0.58	0.56	138

Tabela 4.13: Hiperparâmetros para o modelo Gradient Boosting e seu score

Gradient Boosting		
Hiperparâmetros	Valor	Score do modelo
n_estimators	300	**0.56**
min_samples_split	10	
min_samples_leaf	1	
max_depth	3	
learning_rate	0.01	

Tabela 4.14: Principais resultados para o modelo Gradient Boosting

	precision	***recall***	***F1-score***	***support***
0	0.57	0.76	0.65	74
1	0.55	0.34	0.42	64
accuracy	-	-	0.57	138
macro average	0.56	0.55	054	138
weighted average	0.56	0.57	0.55	138

Tabela 4.15: Hiperparâmetros para o modelo Florestas Aleatórias e seu score

Florestas Aleatórias		
Hiperparâmetros	Valor	Score do modelo
n_estimators	200	**0.58**
min_samples_split	10	
min_samples_leaf	1	
max_features	Sqrt	
max_depth	20	
criterion	Gini	

Tabela 4.16: Principais resultados para o modelo Florestas Aleatórias

	precision	***recall***	***F1-score***	***support***
0	0.59	0.76	0.66	74
1	0.58	0.39	0.47	64
accuracy	-	-	0.59	138
macro average	0.59	0.57	0.57	138

weighted average	0.59	0.59	0.57	138

Ao analisar o desempenho dos modelos de aprendizado de máquina em duas bases de dados distintas, observou-se resultados notáveis. Na base de dados maior, composta por 3600 linhas, a Regressão Logística e o Gradient Boosting destacaram-se, alcançando excelentes *scores* de 96% de assertividade. Já a Árvore de Decisão, embora sólida, apresentou um desempenho um pouco inferior, com um *score* de 89%, enquanto que o modelo Florestas Aleatórias obteve um resultado consistente de 93%.

No entanto, ao avaliar o desempenho desses modelos na base reduzida, com aproximadamente 250 linhas, observou-se uma queda significativa. Para Árvore de Decisão, por exemplo, registrou-se um *score* de 57% de assertividade, indicando possíveis desafios na generalização para conjuntos de dados menores. A Regressão Logística e o Gradient Boosting também mostraram redução no desempenho, com *scores* de 58% e 56%, respectivamente. Já o modelo de Árvores aleatórias teve um *score* de 59%, um pouco maior que os demais.

Essa diferença de desempenho entre as bases sugere que o tamanho do conjunto de dados desempenha um papel crucial na eficácia dos modelos. Enquanto a base maior proporcionou resultados melhores, a base menor reduziu a capacidade de generalização dos modelos.

5. CONCLUSÕES

Neste trabalho, aplicaram-se técnicas de aprendizado de máquina supervisionado para a detecção de notícias falsas. Observou-se que para a base de dados de tamanho maior (3600 linhas), os modelos de Regressão Logística e o Gradient Boosting atingiram taxas de assertividade de até 96%, evidenciando a efetividade do método. O pré- processamento é uma etapa importante em problemas com linguagem natural. A lematização em princípio, permitiria simplificar o texto que serve de entrada para os algoritmos, especialmente para base de dados contendo notícias em português, visto que a língua portuguesa possui muitas flexões. Todavia, ao aplicar o pré-processamento descrito na seção anterior, o resultado alcançado foi na média ligeiramente pior do que sem tratamento. Com isso foi levantado alguns pontos que auxiliam na hipótese para explicar esse resultado:

- Perda de Informação durante o Pré-processamento: A remoção de *stop words* e pontuações pode ter eliminado informações relevantes para a detecção de notícias falsas. A lematização também pode ter reduzido a variabilidade das palavras, prejudicando a capacidade do modelo de distinguir entre notícias verdadeiras e falsas.
- Características Específicas do Idioma Português: O pré-processamento pode ter afetado de maneira diferente as características específicas do idioma português em comparação com outros idiomas. Alguns idiomas têm estruturas gramaticais e padrões específicos que podem ser sensíveis a certas etapas de pré- processamento.

5.1. Sugestões de Aprimoramento

O presente estudo abordou a detecção de notícias falsas por meio de algoritmos tradicionais de aprendizado de máquina supervisionado. Embora esses métodos tenham demonstrado eficácia, a evolução contínua da área sugere a exploração de abordagens mais avançadas para aprimorar a capacidade de análise linguística. Nesse contexto, este artigo propõe a consideração de redes neurais recorrentes (RNNs), Long Short-Term Memory (LSTM) e arquiteturas avançadas, como os *Transformers*, para a detecção de notícias falsas, explorando sua capacidade de capturar relações complexas e nuances linguísticas.

Redes Neurais Recorrentes (RNN) e LSTM:

As RNNs, incluindo as LSTMs, oferecem uma perspectiva dinâmica ao processar sequências de dados, sendo especialmente aplicáveis em tarefas de processamento de linguagem natural (NLP). Sua habilidade em lidar com dependências temporais e capturar relações de longo prazo entre palavras torna-as promissoras para a detecção de notícias falsas, onde o contexto e a estrutura gramatical são importantes. Estudos indicam que modelos baseados em LSTM apresentam melhor desempenho em tarefas de NLP que envolvem compreensão contextual, contribuindo para uma análise mais profunda das sutilezas linguísticas [50] [51].

Transformers:

Por sua vez, os modelos baseados em *Transformers*, como o BERT [52], ganharam destaque recentemente ao superar barreiras em tarefas de NLP. Sua capacidade de compreender o contexto das palavras em uma sentença, explorando relações bidirecionais, é um avanço significativo. Ao incorporar a análise contextual em larga escala, os *Transformers* podem fornecer direcionamentos mais precisos sobre a veracidade do conteúdo, considerando não apenas o vocabulário, mas também as interações complexas entre palavras.

6. REFERÊNCIAS

[1] ITU. Percentage of individuals using the Internet, gráfico de barras, 2023. Disponível em: <https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx>. Acesso em: 27 nov. 2023.

[2] CORNER, J. et al. Fake news, post-truth and media–political change. Media, Culture & Society, 2017. Acesso em: 28 nov. 2023.

[3] GELFERT, Axel. Fake news: A definition. Informal logic, v. 38, n. 1, p. 84-117, 2018. Acesso em: 28 nov. 2023.

[4] PORTELA, M. C. O uso de fake News e seu impacto nas eleições presidenciais de 2018, p. 18-19. Belo Horizonte, MG, 2019. Acesso em: 11 dez. 2023.

[5] MARS, A. Como a desinformação influenciou nas eleições presidenciais? El País, Nova York, 25 fev. 2018. Acesso em: 11 dez. 2023.

[6] PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, v. 12, p. 2825-2830, 2011. Acesso em: 15 jan. 2024.

[7] SALTON, G. & McGILL, M. J. Introduction to Modern Information Retrieval. Acesso em: 12 jan. 2024.

[8] MITCHELL, T. M. Machine Learning. McGraw-Hill Science/Engineering/Math, 1997. Acesso em: 16 jan. 2024.

[9] CORTES, Corinna; VAPNIK, Vladimir. Support-vector networks. Machine learning, v. 20, p. 273-297, 1995. Acesso em: 22 jan. 2024.

[10] REIMAN, L. et al. Classification and Regression Tress. Taylor e Francis, 1984. Acesso em: 22 jan. 2024.

[11] DEVLIN, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Acesso em: 25 jan. 2024.

[12] LIU, F. T.; TING, K. M.; ZHOU, Z. H. (2008). Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (pp. 413-422). IEEE. Acesso em: 25 jan. 2024.

[13] NOGARE, D. Performance de Machine Learning – Matriz de Confusão Disponível em: <https://diegonogare.net/2020/04/performance-de-machine-learning-matriz-de-confusao>. Acesso em: 25 jan. 2024.

[14] MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. Sistemas inteligentes-Fundamentos e aplicações, v. 1, n. 1, p. 32, 2003. Acesso em: 16 jan. 2024.

[15] BOCHIE, K. et al. Aprendizado Profundo em Redes Desafiadoras: Conceitos e Aplicações. Sociedade Brasileira de Computação, p. 3, 2020. Acesso em: 16 jan. 2024.

[16] AGUIAR, A. S. et al. Aprendizado de Máquina, Fundamentos Teóricos. p. 23. Disponível em: < https://www.maxwell.vrac.puc-rio.br/9637/9637_3.PDF >. Acesso em: 16 jan. 2024.

[17] SOUZA, D. H. M. Detecção de Fraudes em Operações com Cartões de Crédito: Uma Abordagem de Aprendizado de Máquina. p. 32, 2023. Acesso em: 16 jan. 2024.

[18] Aprendizado de máquina supervisionado: regressão vs. classificação. Disponível em: <https://ichi.pro/pt/aprendizado-de-maquina-supervisionado-regressao-vs-classificacao-27156968573150>. Acesso em: 17 jan. 2024.

[19] O que é Aprendizado não Supervisionado? | IBM. Disponível em: <https://www.ibm.com/br-pt/topics/unsupervised-learning>. Acesso em: 03 jan. 2024.

[20] COSTA, Paulo Jorge dos Santos. Sistema de detecção de fraude em pagamentos eletrônicos. Estudo e Implementação usando software de distribuição livre. 2019. Dissertação de Mestrado. Instituto Superior de Engenharia do Porto. Acesso em: 17 jan. 2024.

[21] AI WIKI. *Overfitting vs Underfitting*. 2019. Disponível em: <https://machine-learning.paperspace.com/wiki/overfitting-vs-underfitting>. Acesso em: 17 jan. 2024.

[22] Seleção do Modelo, Underfitting, e Overfitting — Dive into Deep Learning 0.17.1 documentation. Disponível em: <https://pt.d2l.ai/chapter_multilayer-perceptrons/underfit-overfit.html>. Acesso em: 17 jan. 2024.

[23] HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. The Elements of Statistical Learning, v. 2, p. 398, 2009. Acesso em: 17 jan. 2024.

[24] KOHAVI, Ron et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Ijcai. 1995. p. 1137-1145. Acesso em: 17 jan. 2024.

[25] CARAMANIS, Constantine; MANNOR, Shie; XU, Huan. 14 robust optimization in machine learning. Optimization for machine learning, p. 369, 2012. Acesso em: 17 jan. 2024.

[26] NOGARE, D. Performance de Machine Learning – Matriz de Confusão, 2020. Disponível em: <https://diegonogare.net/2020/04/performance-de-machine-learning-matriz-de-confusao/>. Acesso em: 18 jan. 2024.

[27] Aprendizado de máquina supervisionado: regressão vc. Classificação. Disponível em: <https://ichi.pro/pt/aprendizado-de-maquina-supervisionado-regressao-vs-classificacao-27156968573150 >. Acesso em: 18 jan. 2024.

[28] NEIVA, D. K. Interpretação de modelos complexos de aprendizado de máquina, p.26, 2023. Dissertação de Mestrado – Instituto de Ciências Matemáticas e de Computação, USP, São Carlos. Acesso em: 12 jan. 2024.

[29] JAMES, G. et al. An Introduction to Statistical Learning: with Applications in R. p.59-120. Acesso em: 18 jan. 2024.

[30] JÚNIOR, C. O. Métricas para Regressão: Entendendo as métricas R^2, MAE, MAPE, MSE e RMSE, 2021. Disponível em: <https://medium.com/data-hackers/prevendo-n%C3%BAmeros-entendendo-m%C3%A9tricas-de-regress%C3%A3o-35545e011e70> Acesso em: 05 jan. 2024.

[31] COGLIANESE, C. & LEHR, D. Regulating by robot: Administrative decision making in the machine-learning era. Geo. LJ, HeinOnline, v. 105, p. 1147, 2016. Acesso em: 18 jan. 2024.

[32] PODGORELEC, V. et al. Decision trees: na onverview and their use in medicine. Journal of medical systems, Springer, v. 26, p. 445-463, 2002. Acesso em: 18 jan. 2024.

[33] JOVIC, A.; BRKIC, K.; BOGUNOVIC, N. A review of feature selection methods with applications, 2015. p. 1200-1205. Acesso em: 22 jan. 2024.

[34] BREIMAN, Leo. Random forests. Machine learning, v. 45, p. 5-32, 2001. Acesso em: 22 jan. 2024.

[35] FRIEDMAN, Jerome H. Greedy function approximation: a gradient boosting machine. Annals of statistics, p. 1189-1232, 2001. Acesso em: 25 jan. 2024.

[36] HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. The elements of statistical learning (Vol. 2). Springer, 2009. Acesso em: 25 jan. 2024.

[37] DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society. Series B (Methodological), v. 39, n. 1, p. 1–38, 1977. Acesso em: 25 jan. 2024.

[38] 1.1. Linear Models - scikit-learn 0.22.2 documentation. Disponível em: <https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression>. Acesso em: 28 jan. 2024.

[39] MANNING, C., D.; RAGHAVAN, P. & SCHUTZE, H. An Introduction to Information Retrieval, 2009. Acesso em: 28 jan. 2024.

[40] PRADO, E. TF-IDF: entenda o que é Frequency Inverse Document Frequency! Disponível em:<https://semantix.ai/tf-idf-entenda-o-que-e-frequency-inverse-document-frequency/>. Acesso em: 30 jan. 2024.

[41] VAZ, A. L. TF-IDF — algoritmo de recomendação, 2022. Disponível em: <https://medium.com/data-hackers/tf-idf-algoritmo-de-recomenda%C3%A7%C3%A3o-6c3cbd55e439>. Acesso em: 30 jan. 2024.

[42] RAULJI, J. K. et al. Stop-Word Removal Algorithm and its Implementation for Sanskrit Language. International Journal of Computer Applications, vol. 150, 2016. Acesso em: 30 jan. 2024.

[43] MANNING, C. D. et al. The Stanford CoreNLP Natural Language Processing Toolkit. Association for Computational Linguistics, p. 55-60, 2014. Acesso em: 31 jan. 2024.

[44] JAVATPOINT. Machine Learning Decision Tree Classification Algorithm - Javatpoint. Disponível em: <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>. Acesso em: 05 jan. 2024.

[45] Anas Brital | Random Forest Algorithm Explained. Disponível em: <https://anasbrital98.github.io/blog/2021/Random-Forest/>. Acesso em: 05 jan. 2024.

[46] Gradient Boosting - AI Wiki. Disponível em: <https://machine-learning.paperspace.com/wiki/gradient-boosting>. Acesso em: 05 jan. 2024.

[47] Logistic Regression - AI Wiki. Disponível em: <https://machine-learning.paperspace.com/wiki/logistic-regression>. Acesso em: 05 jan. 2024.

[48] SANTOS, R. Fake.Br Corpus. Disponível em: <https://github.com/roneysco/Fake.br-Corpus> Acesso em: 25 ago. 2023.

[49] Brazilian Election Fake News 2018. Disponível em: < https://www.kaggle.com/datasets/caiovms/brazilian-election-fake-news-2018> Acesso em: 09 set. 2023.

[50] Vaswani, A., et al. (2017). “Attention Is All You Need.” Advances in Neural Information Processing Systems. Acesso em: 05 mar. 2024.

[51] Hochreiter, S., & Schmidhuber, J. (1997). “Long Short-Term Memory.” Neural Computation. Acesso em: 05 mar. 2024.

[52] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. Acesso em: 05 mar. 2024.