UPIS – UNIÃO PIONEIRA DE INTEGRAÇÃO SOCIAL
FATEC – Faculdade de Tecnologia
Departamento de Informática
Bacharelado em Sistemas de Informação

Walesson Alves de Souza Marques
Walber Silva Martins

Inteligência Artificial, um estudo de caso com utilização de algoritmos de *machine learning* na detecção de *fake news*

Trabalho de Conclusão de Curso de Graduação

Brasília-DF
2021

Walesson Alves de Souza Marques
Walber Silva Martins

Inteligência Artificial, um estudo de caso com utilização de algoritmos de *machine learning* na detecção de *fake news*

Trabalho de Conclusão de Curso submetido ao Curso de Bacharelado em Sistemas de Informação da Faculdade de Tecnologia - FATEC, União Pioneira de Integração Social – UPIS como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação na área de concentração UPIS.

Orientador: Prof. Paulo Vitor Pereira Cotta

Brasília-DF
2021

Walesson Alves de Souza Marques
Walber Silva Martins

Sistemas de Informação

Este Trabalho de Conclusão de Curso foi julgado e aprovado para obtenção parcial do grau de Bacharel em Sistemas de Informação pelo Departamento de Informática da União Pioneira de Integração Social – UPIS.

Brasília-DF, 25 de Novembro de 2021.

Banca Examinadora:

Prof. Msc. Paulo Vitor Pereira Cotta.

Prof. Msc. Alexandre Mori.

AGRADECIMENTOS

Dedico este trabalho a todos que contribuíram diretamente ou indiretamente em minha formação acadêmica. Agradeço a todos aqueles que contribuíram no decorrer desta jornada, em especial a Deus. As nossas famílias que sempre esteve nos apoiando nos estudos. Aos professores por sempre nos guiar nos momentos difíceis. Ao Prof. orientador Paulo Vitor Pereira Cotta que teve papel fundamental para a elaboração deste trabalho.

"Se você tem um sonho, lute por ele, mesmo que todos te chamem de louco, sonhador e tentem te impedir de realizá-lo. Afinal, o sonho é teu, e o único que deve acreditar nele é você."

P.C Brunório, 2021

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O grande avanço da tecnologia permite o fácil acesso e uma grande quantidade de informações, por meio da internet, televisão, jornais e outros meios de comunicação, em alguns casos não é apresentado um conteúdo verídico, tornando-a uma notícia falsa ou *Fake news*. Esse termo *Fake News* vem ganhando muita repercussão, pois esse tipo de informação com rápida propagação torna-se algo ameaçador. A *Fake News* apresenta à sociedade um grande risco, com isso, esse trabalho tem como objetivo usar ferramentas de análise e processamento de texto como NLP (*Natural Language Processing*), em um conjunto de dados e utilizar algoritmos de *Machine Learning* com intuito de classificar notícias.

Palavras-chave: *Fake News*; *Machine Learning*; Dados;

ABSTRACT

The great advance of technology allows for the easy access and a large amount of information, through the internet, television, newspapers and other means of communication, in some cases true content is not presented, making it a false news or fake news. This term Fake News has been gaining a lot of repercussions, as this type of information with rapid dissemination becomes something threatening. Fake News presents society with a great risk, with this work aims to use analysis and text processing tools such as NLP, in a data set and use Machine Learning algorithms in order to classify news.

Keywords: *Fake News; Machine Learning, Data.*

1 INTRODUÇÃO

1.1 Considerações Gerais

Todo mundo depende de vários recursos *online* para ter acesso às notícias da era moderna, onde a internet é de maior alcance. Com o aumento do uso de plataformas de mídia social como Facebook, Twitter e outras, as notícias se espalham rapidamente entre milhões de usuários em um curto espaço de tempo. E as consequências das notícias falsas são de longo alcance, desde influenciar resultados eleitorais em favor de certos candidatos para criar opiniões tendenciosas. WhatsApp, Instagram, e muitas outras plataformas de mídia social são a principal fonte de divulgação de notícias falsas. Este trabalho fornece uma solução introduzindo um modelo de detecção de notícias falsas usando aprendizado de máquina. Este modelo requer como pré-requisito dados extraídos de vários sites de notícias. A técnica de *web scraping* é usada para extração de dados, que é mais usada para criar conjuntos de dados. Os dados são classificados em duas categorias principais, que são conjuntos de dados verdadeiros e conjuntos de dados falsos. O usuário pode descobrir se a notícia dada é falsa ou não no servidor web.

1.2 Descrição do Problema

O termo *Fake News* (notícias falsas) refere-se ao conteúdo de notícias que é falso, enganoso ou fabricado, em que os fatos, fontes ou declarações citadas do conteúdo de notícias não são verificados. Notícias falsas existiram na forma de fofoca, boato e desinformação ao longo da história humana. Para aumentar sua eficácia, esta notícia falsa é espalhada em toda a mídia social. Junto com os bilhões de pessoas usando a mídia social, também existem robôs, ou simplesmente bots, atuando nessas redes. Esses Bots ajudam a propagar notícias falsas mais rapidamente e aumentar sua popularidade nas redes sociais.

1.3 Justificativa e Importância do Trabalho de Conclusão de Curso

Atualmente, em 2021 os problemas com *Fake News* são bem recorrentes. Com isso, na maioria das circunstâncias o público-alvo são pessoas que são de pouca leitura, pouca instrução ou pessoas que possuem uma grande influência sobre outras pessoas. Portanto, o ideal é que o uso de algoritmos de *machine learning* facilite na classificação de notícias falsas e que seja possível atingir o objetivo.

1.4 Objetivos do Trabalho de Conclusão de Curso

1.4.1 Objetivo geral

Qualquer modelo de aprendizado de máquina requer principalmente um conjunto de dados para treinar ou testar o modelo. Para extrair grandes volumes de dados de sites e salvá-los em formato de tabela em um local arquivo ou um banco de dados, usamos a extração de dados da web, popularmente conhecida como técnica de raspagem da web. O objetivo é usar uma coleta de todos os dados recuperados de várias fontes usando as características vívidas do rastreador da web *Scripts 'Scrapy'* e *Python* e, em seguida, analisá-los de acordo com os requisitos. O rastreador da web baseado em python *'Scrapy'* também pode nos ajudar a recuperar o resultado desejado, à medida que se analisa o processo com código específico e se fornece o URL necessário para a iteração extrair os dados da URL de origem.

O conteúdo bruto precisou de determinado pré-processamento de dados antes de ser utilizado nas simulações. O pré-processamento de dados é uma técnica para exploração de dados que converte dados originais em uma forma adequada. Os dados reais são frequentemente imprecisos e, portanto, não poderiam ser utilizados na implementação dos modelos. Isso poderia causar alguns erros.

Dessa forma o objetivo a ser atingido é a identificação de fake News através do uso de técnicas de inteligência artificial.

1.4.2 Objetivos específicos

Com objetivos específicos têm-se:

- Estudar técnicas utilizadas na inteligência artificial;
- Traçar uma pipeline usando:
    - *TF-IDF (Term Frequency - Inverse Document Frequency:* Onde cada documento recebe uma pontuação*. TF-IDF* usa frequência de palavras para identificar palavras que são mais importantes (ocorrem com mais frequência) em um documento. O *TF-IDF Vectorizer* converte documentos em *tokens*, aprende vocabulário, inverte as ponderações de frequência de documentos e permite que você criptografe novos documentos. Quando em comparação com uma implementação não vetorizada, a vetorização nesta técnica pode acelerar significativamente o processo de cálculo.
    - Diferentes classificadores para prever a falsa notícia. Um classificador nos ajudará a ordenar os dados automaticamente ou categorizar os dados em um ou mais de um conjunto de “Notícias *Fakes*”. Classificadores diferentes usarão os recursos extraídos. Todos os recursos extraídos serão usados pelos classificadores. Os classificadores empregados no Machine Learning e que podem dar aderência para o trabalho.

1.5 Hipóteses

As hipóteses que queremos trabalhar, são:

I. Fazer uma revisão sistêmica e pesquisa de padrões de como é empregado as notícias falsas;

II. Subentender pesquisas que possam responder qual o melhor modelo utilizar na classificação de notícias falsas;

III. Saber o quanto um modelo pode auxiliar na classificação de notícias falsas.

1.6 Estrutura e Organização do Trabalho de Conclusão de Curso

Esse trabalho de Conclusão de curso é composto pelos Capítulos:

- Capítulo 1: Introdução do Trabalho
- Capítulo 2: Revisão Bibliográfica
- Capítulo 3: Metodologia de Pesquisa
- Capítulo 4: Estudo de Caso
- Capítulo 5: Resultados
- Capítulo 6: Conclusões e Recomendações

2 REVISÃO BIBLIOGRÁFICA

2.1 Fake news

Fake news pode ser definida como um tipo de jornalismo sensacionalista ou propaganda que consiste deliberadamente desinformar ou espalhar boatos pela televisão, rádio ou pelas mídias sociais [5]. Outra definição um pouco mais ampla, seria uma coleção de informações falsas e imprecisas, fabricadas para imitar a forma de uma notícia convencional e pode ser classificada como desinformação criada propositadamente para enganar as pessoas [6].

2.2 Inteligência artificial

Neste trabalho, agregou-se um conjunto de informações sobre inteligência artificial, com o objetivo de descrever a respeito desta área em grande evolução, que tem desempenhado tanto no presente, quanto nas projeções do futuro.

A inteligência artificial é uma área na Ciência da Computação voltada para o desenvolvimento de sistemas inteligentes, ou seja, sistemas que conseguem pensar de forma semelhante a um ser humano, ou seja, consiga raciocinar, planejar, perceber e processar informações. O funcionamento vai além de certas ordens, podendo tomar decisões autônomas com base em padrões de um conjunto de dados.

Portanto, a inteligência artificial (IA), pode ser subdividida em partes que a compõem, e desta forma é introduzidos os conceitos de *Machine Learning*, *Deep Learning* e *Reinforcement Learning*.

Machine Learning, como o próprio nome já se propõe, é o processo de aprendizado contínuo de máquina. Consiste em explorar estudos e construção de algoritmos que possibilitam compreender de maneira autônoma, ou seja, consiste em preparar dados de entrada e deste modo a máquina pode aprender com estes dados e gerar saídas que atendem a situação.

O *Deep Learning* (aprendizado profundo), pode ser considerado um tipo ou um subcampo do *Machine Learning*, com diferencial de capacitar a máquina a realizar tarefas mais completas, como identificação de imagens, reconhecimento de fala, mineração com redes neurais convolucionais e classificação automática de

doenças oculares. O Deep Learning estabelece parâmetros básicos sobre esses dados e treina o computador para aprender sozinho ao usar várias camadas de processamento no reconhecimento de padrões [1]. Refere-se a imitar o aprendizado espontâneo humano onde, com a experiência, adquire-se a capacidade de executar uma série de atividades.

O *Reinforcement Learning* (aprendizado por reforço), é considerado um tipo de técnica de aprendizado de máquina que permite o agente aprender em um ambiente por meio de interações e feedbacks, ou seja, aprender a solucionar um problema por tentativa e erro. O *Reinforcement Learning* usa recompensas ou punições como sinais de comportamento positivo ou negativo.

2.3 Técnicas de IA

As áreas da Inteligência Artificial*, Machine Learning*, *Deep Learning* e *Reinforcement Learning* possuem técnicas a serem utilizadas, que são divididos em três técnicas: aprendizado supervisionado, aprendizado semi-supervisionado e aprendizado não supervisionado. A decisão para qual técnica utilizar, se faz através de uma análise dos dados, pois cada técnica tem um padrão e características pré estabelecidas.

2.3.1 Aprendizagem supervisionada

Na aprendizagem supervisionada, os algoritmos requerem um conjunto de dados rotulados para realizarem o treinamento. Os algoritmos de aprendizado supervisionado devem aprender somente características que são detalhadas dentro do conjunto de dados. Os modelos de reconhecimento de imagem são considerados aplicações comuns do aprendizado supervisionado, pois esses modelos aceitam um conjunto de imagens rotuladas e aprendem a evidenciar características comuns e um formato previsto.

2.3.2 Aprendizagem não supervisionada

No aprendizado não supervisionado, os desenvolvedores utilizam algoritmos em um conjunto de dados totalmente não rotulados. O algoritmo aprende tirar suas

próprias conclusões sobre os dados, ou seja, descobrir relações implícitas em um conjunto de dados.

2.3.3 Aprendizagem semi-supervisionada

A aprendizagem semi-supervisionada, é utilizada quando o desenvolvedor possui um conjunto pequeno de dados rotulados para o treinamento, assim como um conjunto maior de dados não rotulados. Entretanto, esse tipo de aprendizado é capaz de aprender diante de dados supervisionados e não supervisionados [2]. É viável utilizar esse tipo, quando o desenvolvedor almeja o aprendizado de informações dos dados para o algoritmo, porém também aprenda utilizando alguns dados supervisionados.

2.4 Modelos

2.4.1 *Logistic Regression*

Logistic regression é um algoritmo que lida com problemas de classificação e pode ser utilizado para prever a probabilidade de uma amostra pertencer a determinada classe. É possível utilizar na detecção de fake news para prever qual a probabilidade de uma notícia ser ou não *fake news* [11].

Este modelo é apresentado na equação da Figura 1, em que $x = (x_0, x_1, \ldots, x_n)$ é o vetor de variáveis, $\theta = (\theta_0, \theta_1, \ldots, \theta_n)$ é o vetor de parâmetros e σ, a função logística.

Figura 1 - Equação do modelo logistic regression

$$h_\theta(\mathbf{x}) = \sigma(\mathbf{x}^T\theta)$$

Fonte : [11]

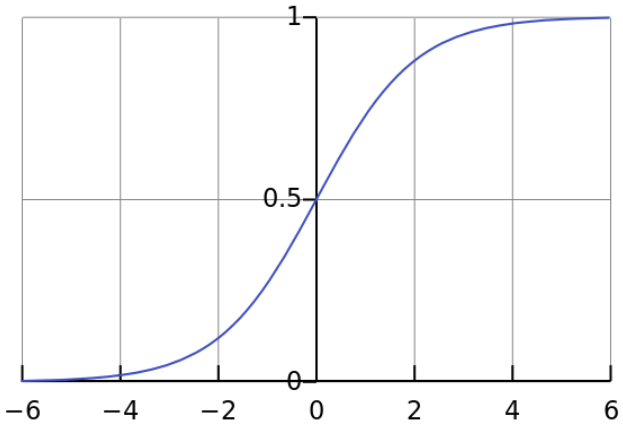
A função logística σ, conforme equação da Figura 2 e a Figura 3, tem por objetivo converter o valor produzido por $x^T\theta$ em uma probabilidade.

Figura 2 - Equação da Função Logística

$$\sigma(u) = \frac{1}{1 + e^{-u}}$$

Fonte : [11]

Figura 3 - Função Logística



Fonte : [17]

Logo, $h_\theta(x)$ pode ser interpretado como a probabilidade que a saída do modelo seja 1, dado o vetor de variáveis x, parametrizado por θ (Equação na figura 4).

Figura 4 - Probabilidade de saída do modelo Logistic Regression

$$h_\theta(\mathbf{x}) = P(y = 1|\mathbf{x}; \theta)$$

Fonte : [11]

Como a função logística varia entre 0 e 1, a predição pode ser feita definindo-se um limiar de 0,5, conforme a equação na Figura 5.

Figura 5 - Definindo limiar de 0,5 para predição do modelo Logistic Regression

$$\hat{y} = \begin{cases} 0, \ se \ h_\theta(\mathbf{x}) < 0,5 \\ 1, \ se \ h_\theta(\mathbf{x}) \geq 0,5 \end{cases}$$

Fonte : [11]

Para obter o modelo treinado, deve-se minimizar uma função custo por meio de algum algoritmo de otimização, resultando, portanto, no vetor de parâmetros θ desejado.

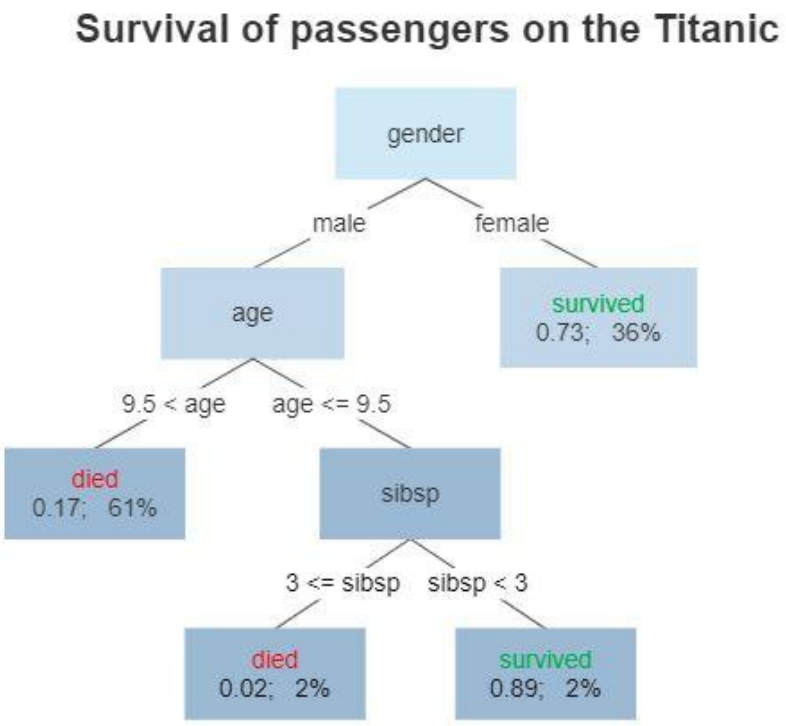
2.4.2 *Decision Tree Classifier*

Decision Tree é uma técnica de aprendizado que consiste na abordagem de dividir para conquistar para resolver um problema. Um problema complexo é dividido em subproblemas simples, os quais são divididos novamente usando a mesma estratégia. As soluções dos subproblemas podem ser combinadas, formando uma árvore, para termos a solução do problema [12].

Em tarefas preditivas, os nós da árvore de decisão envolvem o teste de um atributo em particular. Nós folhas são os responsáveis por dar a classificação que se aplica a todas as instâncias que levam a folha. Para classificar uma instância desconhecida, é percorrida a árvore de acordo com os valores dos atributos testados em seus nós sucessivos, e quando um nó folha é alcançado, a classificação é dada de acordo com a classe assinada no nó folha [12].

A Figura 6 mostra um exemplo de *decision tree*.

Figura 6 - *Decision Tree*



Fonte : [14]

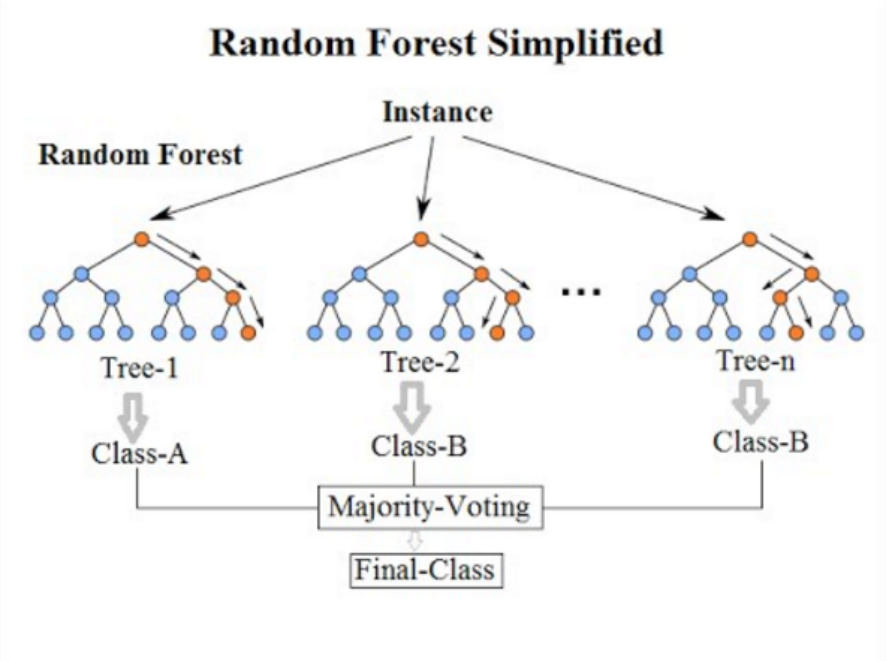
A construção de uma árvore de decisão é um problema recorrente. Primeiro escolhemos um atributo para ser a raiz da árvore e criamos um galho para cada possível valor. Para galho o processo é repetido recursivamente. Se em algum instante todas as instâncias do nó possuem a mesma classificação, o desenvolvimento da árvore naquele nó é parado [12].

2.4.3 *Random Forest Classifier*

Random forest são uma combinação de árvores de predição de modo que cada árvore depende dos valores de um vetor aleatório amostrado independentemente e com a mesma distribuição para todas as árvores da floresta [13].

A Figura 7 mostra um exemplo de random forest.

Figura 7 - *Random forest*



Fonte : [15]

2.5 TF-IDF

TF-IDF (*Term Frequency - Inverse Document Frequency*) é uma técnica que transforma palavras em números. É uma medida estatística que consegue avaliar a importância de uma palavra para o texto e a relevância em uma coleção de textos. Para encontrar o TF-IDF de uma palavra em um documento deve-se calcular multiplicando os termos TF e IDF.

Figura 8 - Função TF-IDF

$$TF - IDF(t,j) = TF_{i,j} * log\left(\frac{N}{df_i}\right)$$

Fonte : Autor

onde $tf_{i,j}$ é o número de ocorrências de i em j, o df_i é o número de documentos que contém i e N é o número total de documentos.

TF (Term Frequency): É a contagem de ocorrências de uma palavra em um documento. Segundo Bruno Stecanella existem várias maneiras de calcular TF, sendo a mais simples uma contagem bruta de ocorrências de uma palavra em um documento, em seguida existem maneiras de ajustar a frequência, pelo o comprimento do documento [8]. A função abaixo representa como é calculado o TF.

Figura 9 - Função TF

$$TF(t,j) = \frac{Ocorrências\ \ do\ termo\ t\ no\ documento\ j}{Total\ de\ termos\ no\ documento\ j}$$

Fonte : Autor

IDF (Inverse document frequency): O IDF é o inverso da frequência de documentos. Pois o IDF é utilizado para acrescentar um peso aos termos que aparecem raramente no documento[7]. Representado pela função abaixo:

Figura 10 - Função IDF

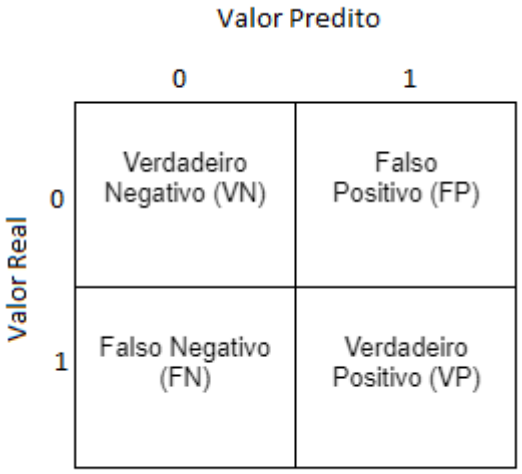
$$IDF(t) = Log\left(\frac{Total\ de\ documentos}{Documentos\ com\ termo\ t}\right)$$

Fonte : Autor

2.6 MÉTRICAS DE AVALIAÇÃO

Métricas de avaliação são utilizadas para analisar o desempenho dos modelos de *Machine Learning.* Neste estudo de caso foi utilizado aprendizado supervisionado, pois o conjunto de dados utilizado são rotulados. As métricas utilizadas, tais como: acurácia, precisão, recall e F1-score, foram calculadas através da matriz de confusão [10]. Apresentada pela Figura 11.

Figura 11 - Matriz de confusão



Fonte : Autor

- **Verdadeiro negativo (VN):** quando o modelo realiza a predição como classe negativa e ao verificar, era realmente negativa, ou seja, o modelo classificou corretamente.
- **Verdadeiro Positivo (VP):** quando o modelo realiza a predição como classe positiva e ao verificar, era realmente negativa, ou seja, o modelo classificou corretamente.
- **Falso positivo (FP):** quando o modelo realiza a predição como classe positiva e ao verificar, nota-se que a classe era negativa, ou seja, o modelo classificou errado.
- **Falso negativo (FN):** quando o modelo realiza a predição como classe negativa e ao verificar, nota-se que a classe era positiva, ou seja, o modelo classificou errado.

A acurácia é uma das métricas mais simples e importantes. Usada para avaliar o percentual de acertos. Representada pela Figura 12.

Figura 12 - Acurácia

$$Acurácia = \frac{VP + VN}{VP + FN + VN + FP}$$

Fonte : Autor

A precisão é calculada por uma equação representada na Figura 13, onde irá avaliar quantidade de verdadeiros positivos sobre a soma de todos que foram classificados como positivos.

Figura 13 - Precisão

$$Precisão = \frac{VP}{VP + FP}$$

Fonte : Autor

O *Recall* é calculado por uma equação representada pela Figura 14, onde avalia a capacidade do modelo de detectar com sucesso resultados classificados como verdadeiro positivo[9].

Figura 14 - *Recall*

$$Recall = \frac{VP}{VP + FN}$$

Fonte : Autor

O *F1-score* é calculado por uma equação representada pela Figura 15, ela é uma média harmônica calculada com base na precisão e o *recall* [9].

Figura 15 - F1-score

$$F1 - score = 2 * \frac{Precisão * Recall}{Precisão + Recall}$$

Fonte : Autor

3 METODOLOGIA

Este trabalho foi desenvolvido com base na pesquisa bibliográfica em artigos, buscando conhecer e analisar formas de se utilizar inteligência artificial para detectar fake news, o que compreende um levantamento de informações sobre inteligência artificial, como as áreas, técnicas e algoritmos. Com base nessas informações foi desenvolvido um modelo de inteligência artificial para testar e investigar a aplicabilidade de tal solução.

3.1 Obtenção do conjunto de dados

Para realizar o treinamento com algoritmos de machine learning é necessário um conjunto de dados com notícias falsas e notícias reais. Nesse estudo de caso foi utilizado o "*Fake.Br Corpus*", que é um conjunto de dados que possui notícias falsas e notícias verdadeiras [16]. Este conjunto de dados possui 3.600 notícias falsas e 3.600 notícias reais. As notícias estão divididas em categorias. A Tabela 1 mostra como está dividido o conjunto de dados.

Tabela 1 - Descrição do Fake.Br Corpus

Categoria	**Quantidade**	**Porcentagem**
Política	4.180	58,0
TV e celebridades	1.544	21,4
Sociedade e notícias diárias	1.276	17,7
Ciência e tecnologia	112	1,5
Economia	44	0,7
Religião	44	0,7
Total	7.200	100

Fonte: Autor

4 ESTUDO DE CASO

4.1 Tecnologias usadas

4.1.1 Python

Python é uma linguagem de programação popular de alto nível, interpretada, imperativa, orientada a objetos, de tipagem dinâmica e forte.

Essa linguagem que foi desenvolvida para ser simples, fácil de aprender e muito versátil, ou seja, é possível utilizá-la para essa linguagem para diversas tarefas.

Pode construir aplicativos, criar sites, desenvolver programas, criar jogos, fazer análise de dados, inteligência artificial, entre outras atividades.

4.1.2 Pandas

Pandas é uma biblioteca escrita em Python para manipulação e análise de dados, particularmente ele oferece estrutura de dados e operações para manipular dados numéricos em formato tabular e séries de temporais. Pandas é uma biblioteca de software livre que é fornecida sob a licença BSD.

4.1.3 NumPy

O NumPy é uma biblioteca para a linguagem Python com funções para se trabalhar com computação numérica. Seu principal objeto é o vetor n-dimensional, ou ndarray.

4.1.4 NLTK

Uma das ferramentas mais utilizadas para processamento de linguagem natural é o *NLTK* (*Natural Language Toolkit*), que foi desenvolvido em Python e apresenta uma gama muito grande de recursos, como: classificação, tokenização, *stemming*, *tagging*, *parsing* e raciocínio semântico. Todas essas funções são utilizadas para análise dos textos, e abaixo é mostrado um exemplo de código que extrai uma árvore semântica de um conjunto de texto. Essa árvore apresenta as classes de cada palavra, ou seja, se uma palavra é uma preposição, artigo, verbo, passado, futuro e consegue identificar até nomes próprios.

Figura 16 - Identificação da classe de cada palavra

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

Fonte - [18]

Figura 17 - Identificação de nome próprio

```
>>> entities = nltk.chunk.ne_chunk(tagged)
>>> entities
Tree('S', [('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'),
           ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'),
       Tree('PERSON', [('Arthur', 'NNP')]),
           ('did', 'VBD'), ("n't", 'RB'), ('feel', 'VB'),
           ('very', 'RB'), ('good', 'JJ'), ('.', '.')])
```

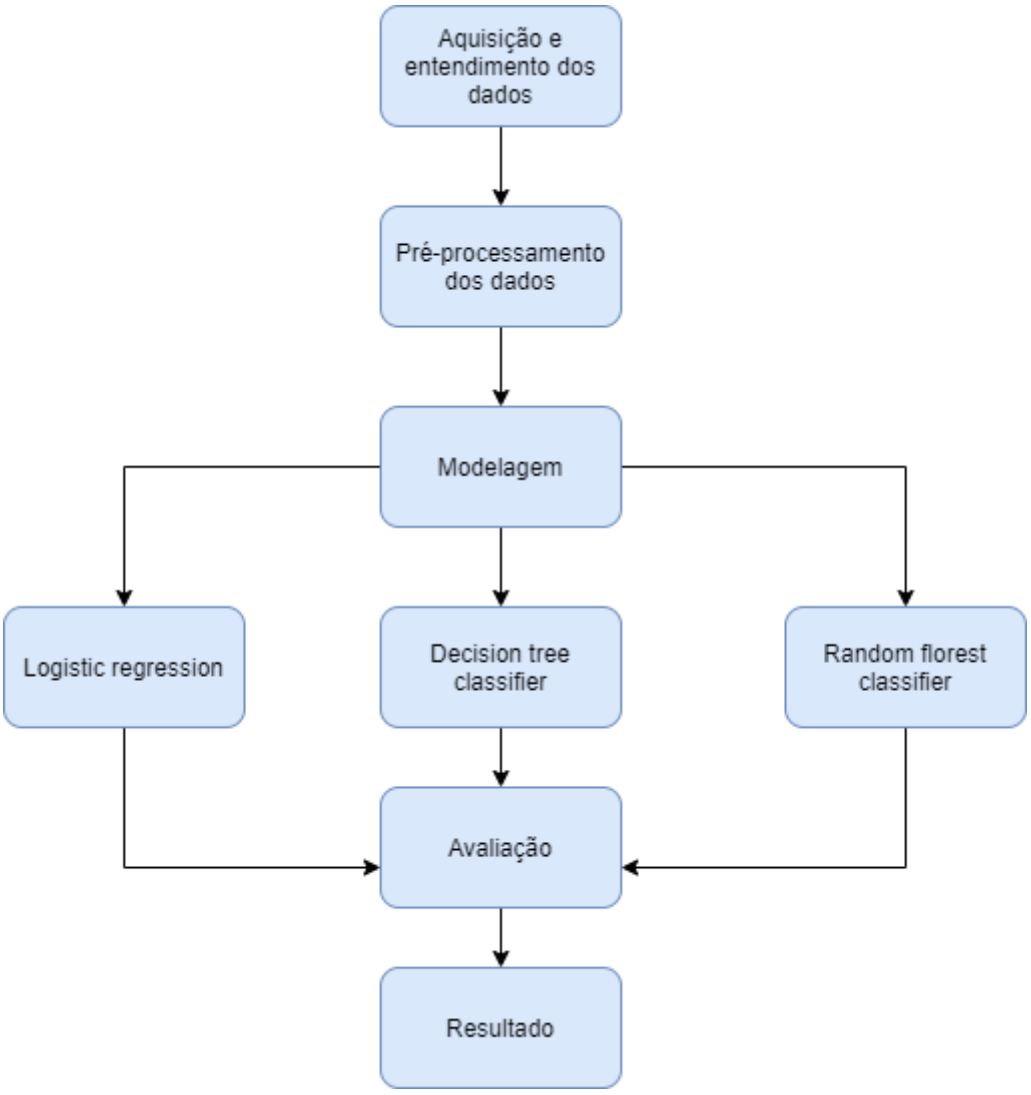
Fonte - [18]

4.1.5 scikit-learn

O scikit-learn é uma biblioteca da linguagem Python desenvolvida especificamente para aplicação prática de machine learning. Esta biblioteca dispõe de ferramentas simples e eficientes para análise preditiva de dados, é reutilizável em diferentes situações, possui código aberto, sendo acessível a todos e foi construída sobre os pacotes NumPy, SciPy e Matplotlib.

4.2 Implementação

Figura 18 - Descrição da implementação



Fonte : Autor

4.2.1 Aquisição e entendimento dos dados

Esta etapa foi mencionada na Seção 3.1, onde foi explicado a separação sobre o conjunto de dados. A parte de entendimento dos dados consiste em analisar os textos ou notícias, para identificar caracteres, símbolos e palavras indesejadas para a próxima etapa.

4.2.2 Pré Processamento

Está fase de pré-processamento que é realizado nas notícias, foi utilizado a linguagem de programação Python com a biblioteca NLTK *(Natural Language Toolkit)* , que possui diversos procedimentos de processamento de linguagem natural.

4.2.2.1 Regex

A figura 19 mostra duas variáveis responsáveis para remover símbolos e tudo que não for número ou letra dos textos.

Figura 19 - Regex

```
# Regex
REPLACE_BY_SPACE_RE = re.compile('[/(),.;!"?\':%&+¤]')
BAD_SYMBOLS_RE = re.compile('[^0-9a-z ]')
```

Fonte - Autor

4.2.2.2 Tratamento dos caracteres acentuados

A figura 20 mostra responsável pelo tratamento de caracteres acentuados, no qual será substituído por outro caractere sem acento.

Figura 20 - Função responsável pelo tratamento de caracteres acentuados

```
# Função para substituação de caracteres acentuados
def strip_accents(text):
  try:
    text = unicode(text, 'utf-8')
  except (TypeError, NameError):
    pass
  text = unicodedata.normalize('NFD', text)
  text = text.encode('ascii', 'ignore')
  text = text.decode("utf-8")
  return str(text)
```

Fonte - Autor

4.2.2.3 Pré-processamento completo

Apresenta-se na figura 21 a função responsável pela conversão dos textos para letras minúsculas, aplicar o Stemmer, remover os Stopwords e por aplicar o Regex e a função de Tratamento de caracteres acentuados.

Figura 21 - Função completa para realizar o pré-processamento

```
# Pré-processamento completo
def pre_process(text):
  text = text.lower()
  text = strip_accents(text)
  text = REPLACE_BY_SPACE_RE.sub(' ', text)
  text = BAD_SYMBOLS_RE.sub(' ', text)
  text = ' '.join(stemmer.stem(word) for word in text.split() if word not in STOPWORDS)
  return text
```

Fonte - Autor

Tabela 2 - Exemplo do pré-processamento completo

Antes	**Após**
'Dr. Ray peita Bolsonaro, chama-o de \x93conservador fake\x94 em entrevista a Danilo Gentili e divide a direita.\n\nEste site vem avisando Jair Bolsonaro que ele deveria abandonar a pauta estatista de vez e fazer um discurso mais convincente para aquela boa parte dos liberais e conservadores do Brasil que querem se ver livres das amarras estatais.\n\nTudo bem que as pesquisas ainda dizem que a maior parte do povo é contra as privatizações, mas o índice (pouco mais de 50% do povo) é fácil de ser revertido...'	'dr ray peit bolsonar cham conserv fak entrev danil gentil divid direit sit vem avis jair bolsonar dev abandon paut estat vez faz discurs convinc boa part liberal conserv brasil quer ver livr amarr estatal tud bem pesquis aind diz mai part pov contr privatizaco indic pouc 50 pov facil ser revert...'

Fonte - Autor

4.2.2.4 Extração das features e separação entre treino e teste

Apresenta-se na figura 22 a separação do conjunto de dados, será dividido em duas partes, “X” e “y”, onde o “X” receberá a feature “texto” e “y” irá receber a feature “Fake”. Em seguida na figura 21, será dividido os dados de treino e teste.

Figura 22 - Extração das features

```
X = df['texto']
y = df['Fake']
```

Fonte - Autor

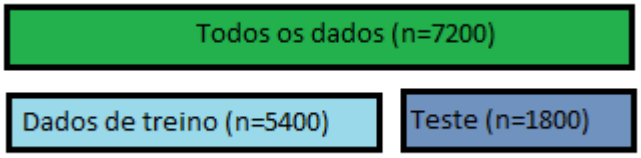
Apresenta-se na Figura 23, a quantidade de dados que foram utilizados para o treinamento e teste, onde pelo *Scikit Learn* estabelece um padrão de 30% para teste e 70% para treino, optamos por utilizar 25% para teste e 75% para treino pois é uma margem muito boa para realizar o treinamento e o teste.

Figura 23 - Código para separação dos dados

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Fonte - Autor

Figura 24 - Separação de dados



Fonte - Autor

4.2.2.5 TF- IDF

Este trecho de código é responsável pela construção da matriz TF-IDF, que está presente na figura 25.

Figura 25 - TF-IDF

```
# Instanciando TFIDF
tfidf = TfidfVectorizer()
X_train = tfidf.fit_transform(X_train)
X_test = tfidf.transform(X_test)
```

Fonte - Autor

4.2.3 Modelagem

Após o pré-processamento, o conjunto de dados estará prontos para a modelagem, nesta etapa, utilizamos três algoritmos, no qual, todos realizaram o treinamento e o teste, logo após foi implementado métricas de avaliação, com intuito de entender como os modelos se comportam com o conjunto de dados proposto.

4.2.3.1 *Logistic Regression*

Apresenta-se na figura 26, o código que, instância o modelo de *Logistic Regression*, o treinamento do modelo e a predição com os dados de teste.

Figura 26 - Código para instanciar o modelo Logistic Regression

```
LR = LogisticRegression()
LR.fit(X_train,y_train)
pred_lr=LR.predict(X_test)
```

Fonte: Autor

4.2.3.2 *Decision Tree Classifier*

Apresenta-se na figura 27, o código que, instância o modelo Decision Tree Classifier no qual foi utilizado hiperparâmetros com intuito de limitar a profundidade da árvore para que diminua a possibilidade de ocorrer overfitting, o treinamento do modelo e a predição com os dados de teste.

Figura 27 - Código para instanciar o modelo Decision Tree Classifier

```
dt = DecisionTreeClassifier(random_state=0, max_depth=8)
dt.fit(X_train, y_train)
pred_dt=dt.predict(X_test)
```

Fonte: Autor

4.2.3.3 *Random Forest Classifier*

Apresenta-se na figura 28, instância o modelo Random Forest Classifier, o treinamento do modelo e a predição com os dados de teste.

Figura 28 - Código para instanciar o modelo Random Forest Classifier

```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
pred_rf = rf.predict(X_test)
```

Fonte: Autor

5 RESULTADOS

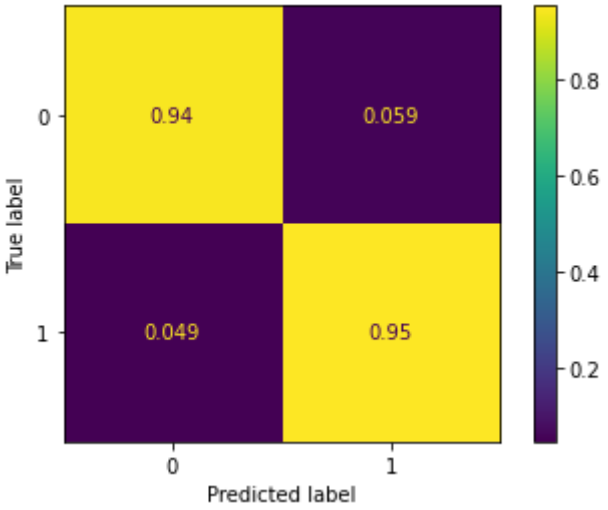
5.1 Análise dos resultados

Esta parte apresenta os resultados de todos algoritmos mencionados Seção.4.2.2, onde foram utilizados os algoritmos de classificação *Logistic Regression Classifier, Decision Tree Classifier e Random Forest*, da biblioteca *scikit-learn*. Serão apresentados os resultados para cada algoritmo utilizado.

5.1.1 *Logistic Regression*

O modelo foi treinado utilizando todos os dados de treinamento e avaliado com os dados de teste, alcançando a matriz de confusão nos dados de teste da Figura 29 e as métricas da Tabela 3.

Figura 29 - Matriz de confusão *Logistic Regression*



Fonte: Autor

Tabela 3 - Resultados Regressão Logística

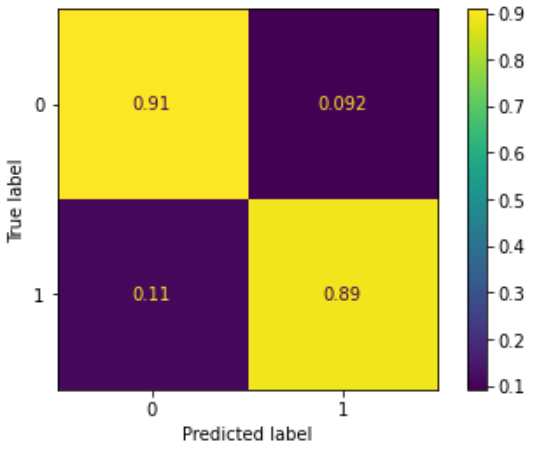
Métricas	**Valor**
Acurácia	95%
Precisão	0,945
Recall	0,945
F1-score	0,945

Fonte: Autor

5.1.2 *Decision Tree Classifier*

O modelo *Decision Tree Classifier* foi utilizado hiperparâmetro[1] para limitar a profundidade da árvore e diminuir a probabilidade de ocorrer *overfitting*[2], ele foi treinado utilizando todos os dados de treinamento e avaliado com os dados de teste, alcançando a matriz de confusão nos dados de teste da Figura 30 e as métricas da tabela 4.

Figura 30 - Matriz de confusão *Decision Tree Classifier*



Fonte: Autor

[1] Hiperparâmetro - são parâmetros ajustáveis que permitem controlar o processo de treinamento do modelo.

[2] *Overfitting* - É quando, nos dados de treino, o modelo tem um desempenho excelente, porém quando utilizamos os dados de teste o resultado é ruim. Pode-se dizer que o modelo "vicia" nos dados de treino.

Tabela 4 - Resultados Decision Tree Classifier

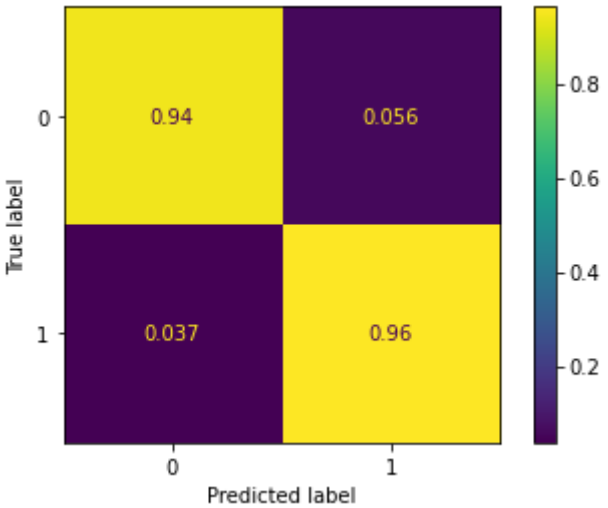
Métricas	**Valor**
Acurácia	90%
Precisão	0,90
Recall	0,90
F1-score	0,90

Fonte: Autor

5.1.3 *Random Forest Classifier*

O modelo *Random Forest Classifier* foi treinado utilizando todos os dados de treinamento e avaliado com os dados de teste, alcançando a matriz de confusão nos dados de teste da Figura 31 e as métricas da tabela 5.

Figura 31 - Matriz de confusão da *Random Forest Classifier*



Fonte: Autor

Tabela 5 - Resultados Random Forest Classifier

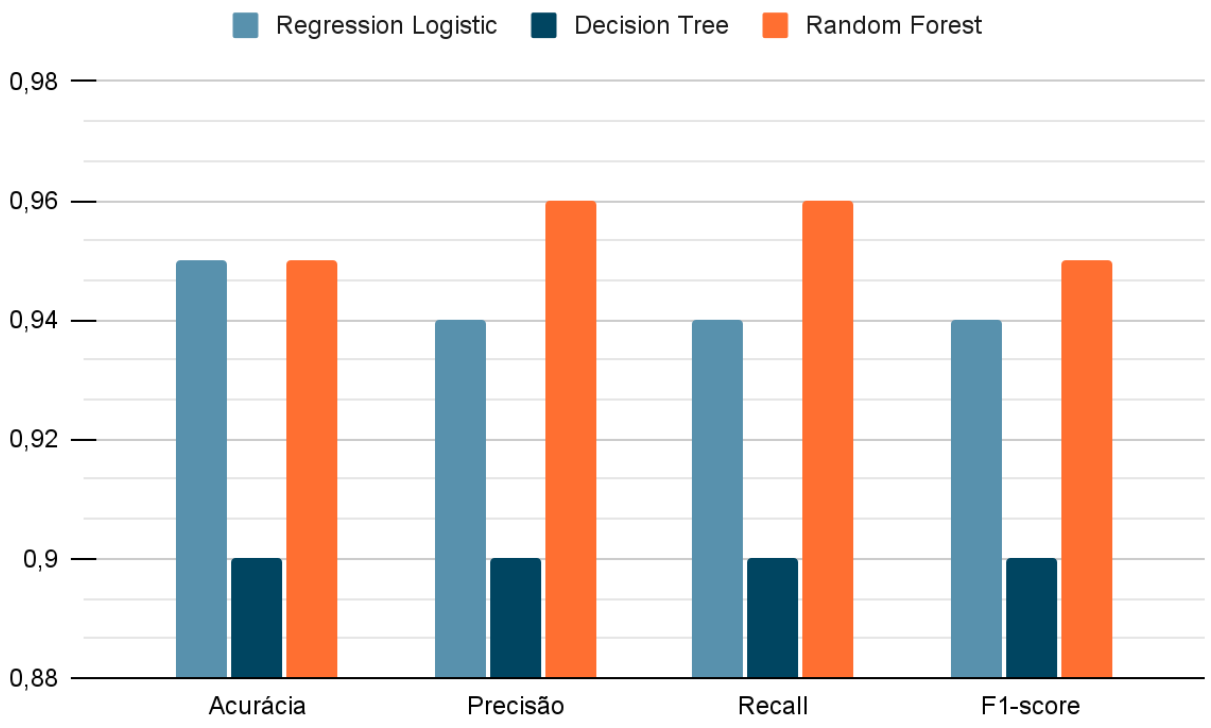
Métricas	**Valor**
Acurácia	95%
Precisão	0,955
Recall	0,955
F1-score	0,95

Fonte: Autor

5.2 Comparação entre os três modelos

A figura 32 apresenta visualmente o desempenho dos modelos avaliados nos dados de teste. Como observado, o modelo Random Forest obteve desempenho superior aos modelos de Regression Logistic e Decision Tree.

Figura 32 - Comparação de resultados dos modelos



Fonte - Autor

Tendo em visto os resultados apresentados pelos modelos, os testes foram feitos em um laboratório controlado baseado no conjunto de dados “*Fake.Br Corpus*” [16], mediante a isso não foi feito nenhum tipo de pesquisa avançada para o desbalanceamento do conjunto de dados, referente a classe de *fake news* ou notícias verdadeiras. Por esse motivo não há possibilidade de fazer nenhum tipo de metrificação ou entendimento, pois á necessidade de teste para verificar a convergência da I.A.

6 CONCLUSÕES E RECOMENDAÇÕES

6.1 Conclusões

Fake news são um problema que está presente na sociedade, buscando uma forma de diminuir ao máximo a propagação e disseminação de fake news, este trabalho teve como objetivo além de detectar notícias falsas em meio a notícias verdadeiras, também identificar qual é o melhor algoritmo para esse tipo de situação entre três algoritmos selecionados, o *Logistic Regression*, *Decision Tree Classifier* e *Random Forest Classifier*, que são os mais utilizados.

No decorrer deste trabalho foram utilizadas diferentes técnicas de aprendizado de máquina e programação, optando por utilizar tecnologias modernas e de código aberto, como a biblioteca Scikit-learn e a linguagem Python.

O treinamento foi realizado com algoritmos clássicos de aprendizagem de máquina, *Logistic Regression*, *Decision Tree Classifier* e *Random Forest Classifier*. O conjunto total de dados possui 7200 linhas, destes, 5400 foram utilizados para o treinamento e 1800 para teste. Todos os modelos foram treinados e testados com a mesma quantidade de dados, com intuito de identificar qual conseguiria obter melhor resultado.

Os resultado adquirido pelo modelo *Random Forest Classifier* se sobressaiu entre os demais, mesmo obtendo a acurácia igual do modelo *Logistic Regression* de 95%, ao se comparar outras métricas como *precisão*, *recall* e *F1-score*, o *Random Forest* obteve números melhores.

Ambos os modelos utilizados atenderam às expectativas gerando resultados satisfatórios, onde todos tiveram *acurácia*, *precisão*, *recall* e *F1-score* acima de 89,99%. Com isso, esse estudo de caso pode trazer grandes benefícios para sociedade, evitando-os de ler e disseminar notícias que apresentam conteúdo falso.

6.2 Recomendações para trabalhos futuros

Para trabalhos futuros o aumento do conjunto de dados para treinar os modelos podem melhorar ainda mais os resultados obtidos. A implementação de uma API (Application Program Interface) ou um portal que fosse possível passar

uma notícia para verificar se é ou não uma fake news, seria uma forma de testar ainda mais os modelos.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] SAS – Software & Soluções de Analytics. Deep Learning, o que é e sua importância.SAS Institute Inc. Disponível em: <https://www.sas.com/pt_br/insights/analytics/deep-learning.html>. Acesso em: 14 Out. 2021.

[2] Minerando dados, Tipos de Aprendizado de Máquina e a História dos criadores de vacas. Escrito por Rodrigo Santana, Março 2018 Disponível em: <https://minerandodados.com.br/tipos-de-aprendizado-de-maquina/>. Acesso em: 17 Out. 2021

[3] Search Enterprise AI, Reinforcement Learning, Escrito por Joseph M. Carew. Disponível em:

<https://searchenterpriseai.techtarget.com/definition/reinforcement-learning>. Acessado: 17 Out. 2021

[4] Monteiro R.A., Santos R.L.S., Pardo T.A.S., de Almeida T.A., Ruiz E.E.S., Vale O.A. (2018) Contributions to the Study of Fake News in Portuguese: New Corpus and Automatic Detection Results. In: Villavicencio A. et al. (eds) Computational Processing of the Portuguese Language. PROPOR 2018. Lecture Notes in Computer Science, vol 11122. Springer, Cham

[5] Junaed Younus Khan, Md. Tawkat Islam Khondaker, Sadia Afroz, Gias Uddin, Anindya Iqbal. A Benchmark Study of Machine Learning Models for Online Fake News Detection. arXiv:1905.04749v2, 2021

[6] Jeffrey Huang 2020 J. Phys.: Conf. Ser. 1693 012158. Disponível em <https://iopscience.iop.org/article/10.1088/1742-6596/1693/1/012158>

[7] TF-IDF from scratch in python on a real-world dataset. Escrito por William Scott Fevereiro 2019 Disponível em:

<https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089> Acessado Nov.2021

[8] MonkeyLearn, Understanding TF-ID: A Simple Introduction, Escrito por Bruno Stecanella. Disponível em:

<https://monkeylearn.com/blog/what-is-tf-idf/>. Acessado: 5 Nov. 2021

[9] Métricas de avaliação em machine learning: Acurácia, precisão, especificidade, F-score e curva ROC, Escrito por Diego Mariano. Disponível em:

<https://diegomariano.com/metricas-de-avaliacao-em-machine-learning/>. Acessado: 4 Nov. 2021

[10]Everything you Should Know about Confusion Matrix for Machine Learning, ,Aniruddha Bhandari - 17 de abril de 2020

<https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/>. Acessado: 5 Nov. 2021

[11] Detecção de notícias falsas utilizando aprendizado de máquina, Fernando Battisti - 20 de outubro de 2020

<https://repositorio.ufsc.br/handle/123456789/217232>

[12] Inteligência Artificial Aplicada a Detecção de Fake News, Leandro Massetti Ribeiro - 15 de julho de 2019

<https://monografias.ufma.br/jspui/handle/123456789/4251>

[13] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324

[14] Gilgoldm - 18 de Maio de 2020, Decision Tree, disponível em:

<https://upload.wikimedia.org/wikipedia/commons/e/eb/Decision_Tree.jpg>

[15] Venkata Jagannath, 24 Março de 2017, Random Forest, disponível em:

<https://upload.wikimedia.org/wikipedia/commons/7/76/Random_forest_diagram_complete.png>

[16] Monteiro R.A., Santos R.L.S., Pardo T.A.S., de Almeida T.A., Ruiz E.E.S., Vale O.A. (2018), Fake.br-Corpus, disponível em:

<https://github.com/roneysco/Fake.br-Corpus>

[17] Qef (talk) - 2 July 2008, The logistic sigmoid function, disponível em:

<https://upload.wikimedia.org/wikipedia/commons/8/88/Logistic-curve.svg>

[18] NLTK Project, Some simple things you can do with NLTK, disponivel em:

<https://www.nltk.org/>

APÊNDICE

Apêndice A - Código fonte

O código fonte do estudo de caso encontra-se disponível em: https://www.kaggle.com/walessonalves/fake-news-detection