Universidade Federal Rural do Semi-árido
Pró-reitoria de Graduação
Departamento de Ciências Exatas e Tecnologia da Informação
Curso de Bacharelado em Sistemas de Informação

Axel Vieira Gomes Costa

Classificador de *fake news* utilizando um modelo de aprendizado de máquina com técnicas de processamento de linguagem natural

Angicos
2020

Axel Vieira Gomes Costa

Classificador de *fake news* utilizando um modelo de aprendizado de máquina com técnicas de processamento de linguagem natural

Monografia apresentada a Universidade Federal Rural do Semi-árido como requisito de obtenção de título de bacharel em Sistemas de Informação.

Orientadora: Prof(a). Dr(a). Thatiana Cunha Navarro Diniz.

Angicos
2020

Axel Vieira Gomes Costa

Classificador de fake news utilizando um modelo de aprendizado de máquina com técnicas de processamento de linguagem natural

Monografia apresentada a Universidade Federal Rural do Semi-árido como requisito de obtenção de título de bacharel em Sistemas de Informação.

Orientadora: Prof(a). Dr(a). Thatiana Cunha Navarro Diniz.

Aprovado em: ___ /___ /___

Banca Examinadora:

_____

Prof(a). Dr(a). Thatiana Cunha Navarro Diniz
Orientador (a) - UFERSA/Campus Angicos

_____

Prof(a). Me(a). Andrezza Cristina da Silva Barros Souza
UFERSA/Campus Angicos

_____

Prof(a). Dr(a). Francisco de Assis Pereira Vasconcelos de Arruda
UFERSA/Campus Angicos

Agradecimentos

Agradeço a Deus por ter me ajudado a chegar onde cheguei, por suas bençãos durante este caminho percorrido.

Aos meus pais Janemeire Costa e Alex Costa, por terem me ajudado desde o começo e nunca terem deixado de acreditar em mim.

A minha namorada Tawanny Borges, por toda força e motivação que me deu para que eu pudesse alcançar meu sonho.

Agradeço aos amigos que fiz durante esse período de graduação, a meu amigo “Fofão” que sempre esteve junto comigo em tudo, ao pessoal do PET por terem feito parte do meu crescimento tanto pessoal como profissional.

Agradeço a todos os professores que me ajudaram no meu crescimento, no meu aprendizado, pelas caronas para chegar em Angicos, nunca esquecerei tudo que fizeram por mim, sou eternamente grato de coração.

A minha orientadora, Prof. Dra. Thatiana Navarro e coorientadora Prof. Ma. Juscimara Avelino que me ajudaram e contribuíram para que este trabalho pudesse ser feito.

A todos que contribuíram de alguma forma para que eu pudesse estar aqui hoje realizando este sonho. Meu eterno agradecimento.

Resumo

Fake news tem um importante impacto social atualmente. Detectar *fake news* tem sido o objetivo de diversas pesquisas e o interesse em categorizá-las tem crescido. Métodos de aprendizado de máquina tem ganhado destaque. Portanto, este estudo objetivou implementar a técnica de classificação do classificador P*assiveAgressiveClassifer* em conjunto com o processamento de linguagem natural para classificar notícias em falsas e verdadeiras. Para tal, quatro *datasets* foram utilizados, contendo obrigatoriamente *title*, *text* e *label*. O treinamento foi dividido em 70% treino e 30% teste, em seguida, foram tratados com *TfidfVectorizer* e submetidos ao classificador *PassiveAgressiveClassifier* para a obtenção dos resultados. As métricas utilizadas foram *accuracy, precision* e *recall*. Todas as bases de dados obtiveram taxa acima de 78%, sendo consideradas ótimas. Entre as bases escolhidas, o *dataset* FK3 obteve o melhor desempenho, apresentando taxa de 99%. Os resultados obtidos neste trabalho mostram o potencial do classificador e o processamento de linguagem natural como uma alternativa a detecção de notícias falsas.

Palavras-chave: aprendizado de máquina, classificadores, *fake news*, inteligência artificial.

Abstract

Fake news has an important social impact currently. Detect fake news has been an objective of several researches and interest in tools to categorize it has grown. Machine learning methods havs been gaining emphasis. Therefore, this study aimed to implemente the Passive Agressive Classifer classification technique together with natural language processing to classify news as false or true. For this purpose, four datasets were used, containing mandatorily title, text and label. The training was divided into 70% training and 30% test, then they were treated with TfidfVectorizer and submitted to the PassiveAgressiveClassifier to obtain the results. The metrics used were accuracy, precision and recall. All databases obtained rates above 78%, being considered excellent. Among the bases chosen, the FK3 dataset had the best performance, presenting a rate of 99%. The results obtained in this work show the potential of the classifier and the natural language processing as an alternative to the detection of fake news.

Keywords: artificial intelligence, classifier, fake news, machine learning.

Lista de Figuras

Lista de Tabelas

Sumário

1. INTRODUÇÃO

O crescimento tecnológico nos últimos anos, incluindo o desenvolvimento de redes sociais amplamente utilizadas ao redor do mundo propiciou um acesso quase que ilimitado a notícias de diferentes fontes. As notícias falsas não são novidade, mas o termo *fake news* popularizou-se em uso global recentemente, embora exista há mais de cem anos (CARDOSO, 2019). A disseminação de *fake news* preocupa pela possibilidade de afetar o bem-estar político, econômico e social (VOSOUGHI, ROY, ARAL, 2018).

O aumento da disseminação de *fake news* pode ser sustentado e justificado pelo uso da Internet, uma vez que todos podem publicar ou postar informações, havendo assim uma tendência de que fatos ou mentiras possam ser propagados com mais facilidade (SORAJUDEEN, AZMI, ABUBAKAR, 2017). Os veículos de *fake news* carecem de normas e processos editoriais que garantem a precisão e credibilidade da informação (LAZER et al., 2018). Vosougui, Roy e Aral (2017) concluíram em seu estudo que as notícias falsas se espalham mais rápido nas mídias sociais do que as verdadeiras.

A presença de programas de Inteligência Artificial tem se difundido cada vez mais na sociedade nas mais diversas áreas, dentre as metodologias de Inteligência Artificial, aprendizado de máquina vem ganhando destaque (CALDAS, 2017). Aprendizado de Máquina ou *Machine Learning*, é uma ramificação da Inteligência Artificial, com intuito de fazer que uma máquina possa adquirir conhecimento através de informações passadas, com o mínimo de intervenção de terceiros (SOUSA, 2020).

1.1 MOTIVAÇÃO

O interesse por ferramentas que categorizam notícias como falsas ou verdadeiras vem crescendo nos últimos anos, principalmente em razão do impacto social que as *fake news* podem causar. Segundo Ahmed, Thaore e Saad (2017), detectar *fake news* é considerada uma tarefa complexa e mais difícil do que detectar, por exemplo, análises falsas de produtos, visto que essas notícias se propagam facilmente principalmente nas mídias sociais.

Diante a problemática das *fake news* a comunidade cientifica tem trabalhado em formas de poder identificar essas notícias, desenvolvendo e testando diversas maneiras para que seja possível classificá-las. Dessa forma, este trabalho propõe uma solução utilizando um algoritmo de aprendizado de máquina e técnicas de processamento de linguagem natural (língua humana) para identificar e classificar notícias em verdadeiras ou falsas, utilizando quatro bases de dados para testar e medir a eficácia do algoritmo proposto.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O principal objetivo deste trabalho é utilizar a técnica de classificação *PassiveAggressiveClassifier* juntamente com processamento de linguagem natural para classificar notícias em falsas ou verdadeiras.

1.2.2 Objetivos específicos

- Identificar repositórios de notícias falsas e verdadeiras;
- Testar as bases de dados no modelo.
- Treiná-lo para classificar as notícias;
- Utilizar os repositórios obtidos para testar a eficiência do algoritmo;
- Identificar as principais métricas para medir o desempenho do algoritmo;

1.1 ORGANIZAÇÃO DO TRABALHO

O trabalho está organizado em cinco capítulos, no primeiro capitulo está a introdução, mostrando a motivação, definição do problema e os objetivos do trabalho. No segundo capitulo está o referencial teórico, o qual irá abordar alguns conceitos sobre as tecnologias utilizadas para o desenvolvimento do projeto. No terceiro capitulo está a metodologia que mostra todas as etapas para criação do código. No quarto capitulo temos os resultados, que irá mostrar a aplicação do modelo em cada base de dados escolhida. Finalmente, no quinto capítulo

apresentam-se as considerações finais do trabalho e apontamentos para trabalhos futuros.

2. REFERENCIAL TEÓRICO

2.1 APRENDIZADO DE MÁQUINA

O uso de aprendizado de máquina já vem sendo explorado através de diversas metodologias e estudos (APHIWONGSOPHON, CHONGSTITVATANA, 2018; GRUPPI, HORNE, ADALI, 2018). Os algoritmos de aprendizado de máquina estão aptos a adquirir e acumular conhecimento, objetivando melhorar a performance em tarefas específicas, sendo um dos tipos de algoritmos de Inteligência Artificial, que desenvolve programas que aprendem a fazer previsões com base em dados, sem a necessidade de auxílio de um programador (ALMEIDA, 2019). Alguns autores dividem o aprendizado de máquina classificando-o como não-supervisionado, supervisionado e semi-supervisionado.

O aprendizado não-supervisionado é feito com dados não rotulados, sendo uma técnica utilizada para encontrar subgrupos de objetos e tem como objetivo classificar as entidades em grupos mutuamente exclusivos baseadas na similaridade entre as instâncias (PORTO FILHO, 2017).

Por outro lado, o aprendizado supervisionado é descrito pelos valores de um conjunto de atributos, possuindo uma classe associada de maneira obrigatória, e essa classe descreve uma instância do fenômeno de interesse, o conceito no qual deseja-se induzir (SANCHES, 2003).

Contudo, para este trabalho foi utilizado o aprendizado semi-supervisionado. Este consiste em uma combinação das duas formas de aprendizado, supervisionado e não-supervisionado. Ao trabalhar com uma base dados com alguns milhares de informações o aprendizado supervisionado acabaria necessitando de muito mais recursos enquanto o aprendizado não-supervisionado teria menos gastos em recursos, porém haveria um comprometimento com a classificação dos dados. Dessa forma ao combinar as duas técnicas torna-se possível trabalhar com grandes *datasets* (GUERREIRO, 2017).

Logo, a escolha do aprendizado semi-supervisionado se encaixa para a realização do trabalho, visto que os *datasets* serão divididos em uma parte com rótulos para treinamento e uma outra parte sem rótulos, assim sendo rotulados após o treinamento. O objetivo principal desse tipo de aprendizado de máquina é rotular

diversos exemplos para um algoritmo de aprendizado de máquina supervisionado que irá rotular os dados através da melhor opção (MATSUBARA, 2004).

Para realizar a predição de rótulos são utilizados classificadores. A classificação tem como objetivo aprender uma função por meio dos dados de treinamento que construa a melhor predição dos rótulos das classes de dados nunca vistos (MACARIO FILHO, 2009). De acordo com Varella (2004), para que uma classificação seja efetiva é necessário que haja alguns pequenos erros.

Algumas ferramentas podem ser usadas para detecção de notícias falsas, como mostram estudos anteriores. Granik e Mesyura (2017) demonstraram a utilização do classificador *Bayes* para detecção de *fake news*. Existem diversos classificadores, o escolhido para ser utilizado neste trabalho foi o *PassiveAggressiveClassifier*. É um algoritmo que trabalha de forma online e que tem grande base teórica, além de ser eficiente e precisa. De acordo com Wang e Vucetic (2010), o algoritmo *PassiveAggressiveClassifier* é capaz de ao receber um novo exemplo, atualizar o modelo de forma que seja parecido ao atual e que tenha uma baixa perda no novo exemplo. O algoritmo é normalmente utilizado para classificar base de dados como *twitter*, *facebook*, entre outros. Não obstante, é capaz de medir e aplicar um peso para cada dado e assim classificá-lo de forma não tendenciosa.

Para a utilização do algoritmo *PassiveAggressiveClassifier* com linguagem natural é necessário fazer a vetorização desses dados. Nesse trabalho, o algoritmo escolhido para vetorizar os dados foi *TfidfVectorizer* que é capaz de realizar a transformação dos dados em um modelo de espaço vetorial. O algoritmo é bem relatado em algumas literaturas para realizar este tipo de procedimento (FELICIANO, 2018; KUMAR, 2020). O algoritmo irá vetorizar todo o texto do banco de dados e dessa forma o classificador *PassiveAggressiveClassifier* irá distingui-los.

2.2 TECNOLOGIAS UTILIZADAS

Além do aprendizado de máquina, diversas tecnologias foram utilizadas para realizar o trabalho, visando o objetivo final, sendo elas: *datasets, anaconda, jupyter notebook, python*, biblioteca *pandas,* biblioteca *numpy* e *scikit-learning*.

Os *datasets* são a base da análise de dados. Um *dataset* é um conjunto de dados tabulares em formato de planilha onde são divididas informação e características. Normalmente um *dataset* contém informações específicas sobre um determinado assunto. Os formatos mais comuns de um *dataset* são *xls*, *csv* e *tsv* (HOPPEN, 2018).

O pacote *Anaconda* é uma plataforma que visa facilitar o acesso a ferramentas de ciências de dados. Sua instalação é fácil e auxilia perfeitamente com a instalação de linguagens de programação como Python, *R* e outras linguagens utilizadas em ciência de dados (WANG, 2012). O *Anaconda* disponibiliza as tecnologias utilizadas neste trabalho, como o IDE e as bibliotecas de aprendizado de máquina.

O *Jupyter Notebook* é um derivado do *IPython* e é uma ferramenta gratuita para o desenvolvimento de códigos em forma de células, além da capacidade de ser utilizado em diferentes sistemas operacionais. Após ser instalado, ele é executado através de um servidor local http://localhost:8888/ (TOOMEY, 2017). Como a IDE pode ser utilizada através de execução de pequenas partes de códigos em células, torna-se mais fácil prever erros e corrigi-los, deixando também o código mais organizado e fácil de implementar outras bases de dados, além de ser possível desenhar gráficos. Todas essas características tornam o *jupyter notebook* uma IDE mais ágil e de melhor manuseio. (KLUYVER et al., 2016)

A linguagem *Python* é uma linguagem de programação de alto nível que é interpretada e orientada a objetos. Contém uma sintaxe clara e objetiva que beneficia a leitura do código, tornando a linguagem mais produtiva, além de ser multiplataforma sendo possível executá-la em sistemas operacionais *Unix*, *Mac* e *Windows* (BORGES, 2014). Dentro da ciência de dados, o *Python* se tornou uma linguagem habitual por suas características e que possui uma comunidade bem ativa.

A biblioteca *Pandas* é uma biblioteca em *Python,* foi inicialmente criada por Wes McKinney no ano de 2008 e está em constante desenvolvimento até os dias de hoje. *Pandas* é um nome derivado de *panel data* que é um termo utilizado para grupos de dados com mais de uma dimensão (MCKINNEY, 2011). Sua principal característica em um modelo de aprendizado de máquina é o fato de transformar um *dataset* em um *dataframe*. De acordo com Chang (2012), o *dataframe* é uma lista de vetores e fatores, cada um representando uma coluna. Desse modo, é possível

fazer análises e manipulações dos dados de forma mais eficiente. A biblioteca pode ser baixada e utilizada de maneira gratuita, sua documentação está disponível no site pandas.pydata.org/docs/.

NumPy é uma biblioteca de *Python* e significa *Numerical Python*, a biblioteca foi criada em 2005 por Travis Oliphant, porém seu conceito surgiu desde os anos 1990. Contém os mais diferentes recursos para realizar qualquer tipo de cálculo numérico, mas tem como objetivo principal realizar cálculos numéricos em *arrays* multidimensionais (WALT, 2012). A biblioteca é gratuita e *open source*, ela está disponível para download junto de sua documentação no site numpy.org.

O *scikit-learn* é uma biblioteca de *Python* que possui várias implementações de algoritmos de aprendizado de máquina, tais como de regressão, classificação e agrupamento. Foi projetado para facilitar o uso do aprendizado de máquina através de uma linguagem simplificada, além de trabalhar junto com as bibliotecas *NumPy* e *SciPy* (AMARAL, 2018). O *scikit-learn* teve origem em um evento do Google como um projeto de David Cournapeau no ano de 2007. Em 2010, a INRA, instituição Francesa de pesquisa em ciência da computação e automação, patrocinou o projeto que teve seu lançamento em meados de janeiro de 2010 (SCIKIT-LEARN, 2020). A biblioteca é disponível gratuitamente e *open source* que pode ser acessada no site scikit-learn.org.

Com o *scikit-learn* foi possível realizar os procedimentos de aprendizado de máquina mais facilmente, o mesmo possui diversos algoritmos de aprendizado de máquina, dentre eles o PA e *TfidfVectorizer* que foi citado anteriormente. Possui também métricas que o tornam capaz de medir o grau de qualidade dos algoritmos, como o *confusion_matrix, accuracy_score, recall_score* e *precision_score*.

O *confusion_matrix* é responsável por analisar e mostrar as classificações reais e previstas feitas por um sistema de classificação. Normalmente é responsável pelos resultados do desempenho do sistema (VISA, 2011). Pode observar na Tabela 1, uma matriz 2x2 com dois tipos de classes:

Tabela 1 – Exemplo da *confusion_matrix*.

	Predição Negativa	**Predição Positiva**
Negativo Atual	Verdadeiro Negativo (VN)	Falso Positivo (FP)
Positivo Atual	Falso Negativo (FN)	Verdadeiro Positivo (VP)

Fonte: Autoria Própria.

- VP é o número de vezes que uma predição realmente é positiva;
- FP é o número de vezes que uma predição não é realmente positiva;
- VN é o número de vezes que uma predição é realmente negativa;
- FN é o número de vezes que uma predição não é realmente negativa;

Com essas métricas são realizados os cálculos para verificar o desempenho do classificador, calculando agora *accuracy*, *recall* e *precision*.

- *Accuracy* é a proporção do número total de previsões que realmente estavam corretas. Utiliza-se este cálculo para determiná-la:

$$Accuracy = \frac{VP + VN}{VP + FP + VN + FP}$$

- *Recall* é a proporção de casos positivos que foram identificados corretamente. Utiliza-se este cálculo para determiná-lo:

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

- *Precision* é a proporção dos casos positivos que estavam corretos. Utiliza-se este cálculo para determiná-la:

$$Precision = \frac{VP}{VP + FP}$$

Estas são as principais métricas utilizadas para analisar se o algoritmo possui ou não um bom desempenho. O critério para essa classificação se dá pela função sigmóide. A função sigmóide, conhecida como função logística, tem como finalidade transformar as grandezas de 0 até 1 (GOODFELLOW; BENGIO; COURVILLE, 2016). Pode notar a seguir o cálculo para determiná-la:

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

$$x = Entradas \times Pesos$$

Podemos ver na fórmula que x representa o valor de entrada que multiplica os pesos, ou seja, x representa as palavras dos textos que é multiplicada pelo seu grau de importância. A função sigmóide é muito utilizada para diversos algoritmos de aprendizado de máquina, de forma que o resultado sempre vai variar entre 0 e 1. Portanto, quando o resultado for mais próximo de 0 pior ele será e quanto mais próximo de 1 melhor será.

3. METODOLOGIA

3.1 VISÃO GERAL

Este trabalho propõe a criação de um classificador de notícias falsas utilizando aprendizado de máquina.

Para iniciar o projeto foi necessário fazer uma pesquisa com alguns *datasets* (conjuntos de dados) existentes, os quais estavam disponíveis no site do *Kaggle*. Dessa forma foram escolhidos quatro *datasets* com tabelas parecidas, contendo obrigatoriamente: *title, text* e *label*. O formato dos *datasets* são padrão *Character separated values* (csv). Os nomes dos *datasets* foram modificados para FK1, FK2, FK3 e FK4 com intenção de facilitar a manipulação durante os testes.

Para preparar o ambiente foi utilizado o *Anaconda*, um pacote que possui diversas bibliotecas, ferramentas e IDE's. Dentre todas as opções foi escolhido a linguagem de programação *Python* e o ambiente de desenvolvimento integrado (IDE) *Jupyter Notebook* para a criação do algoritmo. A linguagem *Python* foi escolhida por sua facilidade ao trabalhar com *datasets*, contendo bibliotecas específicas.

As bibliotecas utilizadas para o desenvolvimento do algoritmo foram:

- *scikit-learning*: Escolhida por sua facilidade e seu grande acervo voltado para área de aprendizado de máquina.
- *pandas*: Escolhida para trabalhar a mineração de dados dos *datasets*.
- *numpy*: Escolhida para realizar a verificação das informações do *dataset*.

Assim foi montado o ambiente de trabalho para o desenvolvimento do algoritmo de detecção de *fake news*.

3.2 *DATASET*

Os *datasets* escolhidos para o trabalho contém mais de 50 mil artigos, notícias e informações de vários tipos de fontes da internet. E todos estão disponíveis no site kaggle.com.

A plataforma *Kaggle* pertence ao Google e pode ser utilizada de forma gratuita. Nela a comunidade de cientistas de dados pode compartilhar experiências

e projetos junto com suas bases de dados (CARVALHO, 2018). Na Figura 1 pode observar uma amostra do *dataset*:

Figura 1 – Base de dados FK1.

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1
5	5	Jackie Mason: Hollywood Would Love Trump if He...	Daniel Nussbaum	In these trying times, Jackie Mason is the Voi...	0
6	6	Life: Life Of Luxury: Elton John's 6 Favorite ...	NaN	Ever wonder how Britain's most iconic pop pian...	1
7	7	Benoît Hamon Wins French Socialist Party's Pre...	Alissa J. Rubin	PARIS — France chose an idealistic, traditi...	0
8	8	Excerpts From a Draft Script for Donald Trump'...	NaN	Donald J. Trump is scheduled to make a highly ...	0
9	9	A Back-Channel Plan for Ukraine and Russia, Co...	Megan Twohey and Scott Shane	A week before Michael T. Flynn resigned as nat...	0

Fonte: Autoria Própria.

3.3 PRÉ-PROCESSAMENTO DOS DADOS

Antes de utilizar os *datasets*, primeiro ele foi transformado em um *dataframe* e após isso feito o tratamento dos dados. Primeiramente foi visto se havia dados *NaN* (Nulos), caso houvesse os dados seriam substituídos por uma frase, como exemplo: “Sem Título”. A *label* de alguns *datasets* estavam como unidade binária (0 e 1), logo foi feita a substituição de 0 por falso e 1 por verdadeiro, para facilitar a visualização. Caso houvesse algum tipo de identificador, como *id*, seria excluído, pois não tem importância e poderia prejudicar aplicação do modelo. Dessa forma o *dataset* está pronto para ser utilizado. A seguir, na Figura 2, mostra um exemplo de como o *dataset* ficou após o tratamento:

Figura 2 - Dados após ser feito o tratamento.

	title	author	text	label
0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	TRUE
1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	FAKE
2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	TRUE
3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	TRUE
4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	TRUE
5	Jackie Mason: Hollywood Would Love Trump if He...	Daniel Nussbaum	In these trying times, Jackie Mason is the Voi...	FAKE
6	Life: Life Of Luxury: Elton John's 6 Favorite ...	Sem Autor	Ever wonder how Britain's most iconic pop pian...	TRUE
7	Benoît Hamon Wins French Socialist Party's Pre...	Alissa J. Rubin	PARIS — France chose an idealistic, traditi...	FAKE
8	Excerpts From a Draft Script for Donald Trump'...	Sem Autor	Donald J. Trump is scheduled to make a highly ...	FAKE
9	A Back-Channel Plan for Ukraine and Russia, Co...	Megan Twohey and Scott Shane	A week before Michael T. Flynn resigned as nat...	FAKE

Fonte: Autoria Própria.

3.4 *JUPYTER NOTEBOOK*

Para realizar a codificação do projeto foi escolhido o *Jupyter Notebook*. Primeiramente foi utilizado o *PyCharm*, porém houveram alguns problemas durante o desenvolvimento na IDE, a máquina demorava para abrir e realizar a execução do código. Dessa forma, a IDE escolhida e utilizada até o final do projeto foi o *Jupyter Notebook*.

3.5 BIBLIOTECAS

Para o desenvolvimento do projeto foram utilizadas as bibliotecas: *pandas*, *numpy* e *scikit-learning*. Ao utilizar o pacote *Anaconda* e a IDE *Jupyter Notebook* não foi necessário fazer as instalações das bibliotecas *pandas*, *numpy e scikit-learn*, pois já estão integradas no pacote, apenas foi necessário realizar as importações. Nesta subseção serão abordadas as bibliotecas.

3.5.1 Pandas

A biblioteca *pandas* foi utilizada para realizar o tratamento dos *datasets* instalados. Pode-se importá-la através do comando *import pandas*. Na Figura 3, pode ver sua importação além de um exemplo de sua utilização no projeto.

Figura 3 - Importando a biblioteca *Pandas*.

Fonte: Autoria Própria.

Nota-se na Figura 3 que após a importação da biblioteca existe o argumento *as pd* o qual é responsável por dar um “apelido” para biblioteca, então pode utilizar apenas *pd* no lugar de *pandas*. Sua principal funcionalidade no projeto se dá pela importação do *dataset* e por transformá-lo em um *dataframe*, tornando mais fácil a sua manipulação.

3.5.2 *NumPy*

Para fazer a análise das dimensões do *dataframe* foi utilizado a biblioteca *NumPy*. Foi utilizado o comando *shape*, o qual retornará uma *tupla* (lista imutável) com a quantidade de linhas e colunas, como mostra na Figura 4.

Figura 4 – Verificando dimensões do *dataframe*.

```
In [37]: df.shape
Out[37]: (6335, 4)
```

Fonte: Autoria Própria.

Como resultado da Figura 4 pode analisar que as dimensões do *dataframe* contém 6335 linhas e 4 colunas.

3.5.3 *Scikit-learn*

Para importar a biblioteca *scikit-learn* é utilizado o comando *import sklearn* e para escolher um algoritmo específico utiliza-se *from sklearn.”biblioteca” import* “algoritmo”, como pode observar na Figura 5.

Figura 5 - Importando bibliotecas

```
In [17]: #Importando as bibliotecas utilizadas no projeto
         import numpy as np
         import pandas as pd
         from sklearn.model_selection import train_test_split
         from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn.linear_model import PassiveAggressiveClassifier
         from sklearn.metrics import accuracy_score, confusion_matrix
```

Fonte: Autoria Própria.

Durante a realização do projeto foram utilizados cinco algoritmos da biblioteca *scikit-learn*, sendo eles: *train_test_split*, *TfidfVectorizer, PassiveAggressiveClassifier, accuracy_score*, *confusion_matrix*.

3.5.3.1 *train_test_split*

O algoritmo *train_test_split* permite que o usuário possa dividir o *dataframe* em uma porcentagem definida de treino e teste. É possível utilizar e ajustar os parâmetros, assim tornando o algoritmo mais eficiente e personalizado para o *dataframe* escolhido. Pode ver a sua utilização na Figura 6.

Figura 6 - Treinando o modelo.

```
In [31]: x_train,x_test,y_train,y_test=train_test_split(df['text'], labels, test_size=0.3, random_state=7)
```

Fonte: Autoria Própria.

Foram criadas quatro variáveis, sendo elas *x_train*, *x_test*, *y_train*, *y_test*. Nelas serão armazenados o treino e o teste de entrada (x) e saída (y). Precisou ser feito um ajuste dos parâmetros para que o algoritmo pudesse trabalhar da melhor forma com o *dataframe*. Primeiro foi necessário informar o que seria recebido: (*df['text']*) e sua saída (*label*), logo após foi escolhido que 70% do *dataframe* seria utilizado para os treinos e os outros 30% seriam utilizados para o teste. É necessário a utilização de um *random_state* para que os dados escolhidos não fossem aleatórios durante as execuções, podendo assim prejudicar no resultado final.

3.5.3.2 *TfidfVectorizer*

O *TfidVectorizer* foi utilizado para converter os textos em uma matriz de recursos TF (*Term Frequency*) e IDF (*Inverse Document Frequency*), pois só é possível processar com os modelos dados numéricos. Com esse algoritmo torna-se mais interessante realizar a análise dos textos, comparado a outros algoritmos como o *CountVectorizer*, por exemplo, que conta a quantidade de vezes que uma palavra aparece no documento assim surgindo uma tendência nos resultados, desconsiderando algumas palavras úteis que aparecem raramente durante o documento. O *TfidVectorizer* faz uma análise das palavras que são utilizadas mais frequentemente e adicione pesos a ela, para que as palavras importantes tenham

maior peso que outras que se repetem por várias vezes e não são tão importantes. Na Figura 7 mostra o ajuste de parâmetro.

Figura 7 - ajuste de parâmetros do *TfidVectorizer*.

```
In [32]: tf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
```

Fonte: Autoria Própria

Foi feito um ajuste de parâmetros para que o *TfidVectorizer* entendesse que os dados estariam em inglês e foi escolhida uma frequência máxima de 0.7 para cada palavra que aparecia no documento.

É necessário transformar esse vetor em dados para treino e teste, para isso foi preciso criar novas variáveis para recebê-los, conforme é mostrado na Figura 8:

Figura 8 - Transformando os dados em treino e teste.

```
In [33]: tf_train = tf_vectorizer.fit_transform(x_train)
         tf_test = tf_vectorizer.transform(x_test)
```

Fonte: Autoria Própria.

Os dados transformados são apenas as entradas, dessa forma apenas o *x_train* e *x_test* são transformados agora em *tf_train* e *tf_test*. Dessa forma, os dados foram vetorizados e divididos para serem aplicados no algoritmo de classificação *PassiveAggressiveClassifier*.

3.5.3.3 *PassiveAggressiveClassifier*

O modelo de aprendizado de máquina *PassiveAggressiveClassifier* é um algoritmo de classificação será utilizado para analisar a saída, verificando se a notícia será falsa ou verdadeira. É um algoritmo normalmente utilizado para classificação de dados online, por exemplo os do *Twitter*. Pode visualizar sua aplicação na Figura 9.

Figura 9 - Modelo *PassiveAggressiveClassifier*.

```
In [34]: pac = PassiveAggressiveClassifier(max_iter=50)
         pac.fit(tf_train, y_train)
```

Fonte: Autoria Própria.

No primeiro momento foi iniciado o classificador com um máximo de 50 iterações. Após isso é feito o treino utilizando o modelo com os dados *tf_train* os dados vetorizados e *y_train* a sua saída (falso ou verdadeiro).

3.5.3.4 *accuracy_score*

Para verificar a acurácia do modelo na base de dados é utilizado a biblioteca *accuracy_score*, onde o resultado será obtido entre 0 e 1, sendo 0 o pior e 1 o melhor resultado possível. Porém, foram feitos alguns ajustes para o resultado ser percentual, como é mostrado na Figura 10.

Figura 10 - Cálculo da Acurácia.

```
In [35]: y_pred = pac.predict(tf_test)
         score = accuracy_score(y_test, y_pred)

         print(f'Accuracy: {round(score*100, 2)}%')
```

Fonte: Autoria Própria.

Primeiro é feito um *predict* do valor dos dados de teste que é denominado por *y_pred* e logo após é realizado o cálculo da acurácia para saber quão bom o modelo de aprendizado de máquina é para essa base de dados.

3.5.3.5 *confusion_matrix*

O algoritmo *confusion_matrix* é utilizado para verificar se o modelo é tendencioso. Ele apresentará quatro resultados: positivo, negativo, falso positivo e falso negativo. Na Figura 11 mostra a sua aplicação.

Figura 11 - Aplicando *confusion_matrix*.

```
In [36]: print(confusion_matrix(y_test, y_pred, labels=['FAKE', 'TRUE']))
```

Fonte: Autoria Própria.

O algoritmo faz uma verificação nos dados que foram testados e analisa se os resultados falso e verdadeiro são tendenciosos ou não.

4. RESULTADOS

Este capítulo contém os dados resultantes ao fim do projeto. Todos os resultados aqui mostrados, são referentes aos passos descritos no capítulo anterior.

4.1 METODOLOGIA DOS EXPERIMENTOS

Primeiramente, os experimentos realizados neste projeto foram executados na IDE *PyCharm*, porém houve alguns problemas de execução, como a demora para carregar todas as informações. Para trabalhar de forma mais organizada e eficiente, foi utilizado o *jupyter notebook*, que trouxe como vantagem a utilização de células, dessa forma, o programa não precisava ser executado de forma completa durante os testes. Assim, foi possível dividir o projeto em cada parte específica para caso houvesse alguma alteração necessária ou algum erro, seria mais fácil de entender e corrigir.

Para realizar os experimentos foram utilizadas quatro bases de dados para haver uma diversificação dos resultados, assim e possível verificar o grau de eficiência do programa em outras bases e compará-las. Os conjuntos de dados foram divididos em duas partes, sendo uma para treino e outra para teste.

Utilizando o programa Anaconda, foi possível obter com facilidade as configurações necessárias para realizar o projeto. Com isso, foi possível obter a IDE do *jupyter notebook* além das bibliotecas já baixadas. Para realizar os procedimentos de aprendizado de máquina, foi utilizado a biblioteca *scikit-learn*. Para realização do pré-processamento foi utilizada a biblioteca *TfidfVectorizer*. Para trabalhar com a base de dados, foi utilizada as bibliotecas *pandas* e *numpy*.

Para medir o desempenho do algoritmo de aprendizado de máquina e obter os resultados finais foram utilizadas as bibliotecas *confusion*, *matrix* e *accuracy_score*.

4.1.1 Configuraçções do ambiente

4.2 RESULTADOS

Primeiramente, são apresentados os dados na Tabela 3, mostrando as principais métricas para avaliar o desempenho do algoritmo de aprendizado de máquina, apresentando o resultado do experimento após utilizar o classificador *PassiveAggressiveClassifier*. O resultado foi comparado com cada banco de dado escolhido para verificar a eficiência do modelo.

O primeiro *dataset* utilizado para testar o algoritmo foi chamado de FK1.

Tabela 3 - Informações *dataset* FK1.

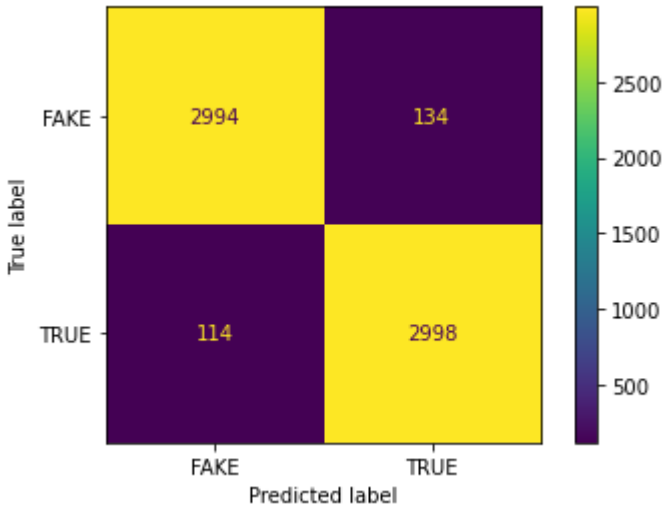
Shape	**Tamanho**	***NaN***	***Label***	**Tipo Arquivo**
(20.800, 5)	754 MB	2.554	0 ou 1	.csv

Fonte: Próprio Autor.

O primeiro *dataset* é um arquivo .csv, possui 20.800 linhas e 5 colunas (*id*, *title*, *author*, *text*, *label*). Para obter o melhor resultado, inicialmente foi feito o tratamento dos dados NaN, reduzindo-os a zero. Como saída, o *dataset* retorna duas formas de resultado, 0 (*falso*) ou 1 (*verdadeiro*).

O *dataset* foi dividido em 14.560 dados para treino e 6.240 para teste. A Figura 12 exibe os resultados do classificador *PassiveAggressiveClassifier* após utilização das principais métricas para medir seu desempenho.

Figura 12 - *Confusion Matrix* FK1.



Fonte: Autoria Própria.

É possível observar na Figura 12 que a matriz confusão do classificador *PassiveAggressiveClassifier* é representada em uma matriz 2x2. O classificador nesse *dataset* teve um total de 96,33% de acertos, onde o classificador previu que era falso e de fato era realmente falso, e 95,72% de acertos em que o classificador previu verdadeiro e realmente era verdadeiro.

Com isso é possível analisar que houve uma falha causando um total de 3,67% de falsos positivos e um total de 4,28% de falsos negativos. Como é possível perceber trata de um número baixo comparado à quantidade que o algoritmo foi capaz de acertar. Pode-se analisar na Tabela 4 a performance geral do modelo.

Tabela 4 - Performance geral.

Accuracy	***Precision***	***Recall***
0.9602564102564103	0.9602759763679303	0.9602564102564103

Fonte: Autoria Própria.

Pode analisar que nesse modelo o algoritmo *PassiveAggressiveClassfieir* foi capaz de ter uma taxa total de mais de 96% de *accuracy*, *precision* e *recall*. A *precision* indica onde o algoritmo marcou como verdadeiro, quando realmente era verdadeiro. O *recall* indica se for realmente dessa classe, o tão frequente classifica

como ela. E a *accuracy* indica a performance geral do modelo. O valor é medido entre 0 e 1, quanto mais próximo de 1 melhor é o modelo, ou seja, o resultado obtido está classificado como excelente para esse modelo nesse *dataset*.

O segundo *dataset* utilizado para testar o algoritmo foi chamado de FK2, conforme descrito na Tabela 5:

Tabela 5 - Informações *dataset* FK2.

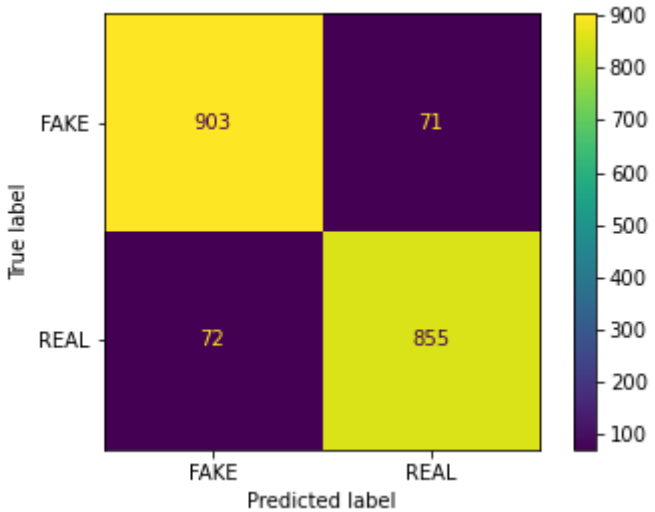
Shape	**Tamanho**	***NaN***	***Label***	**Tipo Arquivo**
(6.335, 4)	29.2 MB	0	FAKE ou REAL	.csv

Fonte: Próprio Autor.

O segundo *dataset* é um arquivo .csv, possui 6.335 linhas e 4 colunas (*id*, *title*, *text*, *label*). Como não havia dados nulos não foi necessário fazer o tratamento. Como saída, o *dataset* retorna duas formas de resultado: falso ou real.

O *dataset* foi dividido em 4.434 dados para treino e 1.901 para teste. A Figura 13 a seguir mostra os resultados do classificador *PassiveAggressiveClassifier* após a utilização das principais métricas para medir seu desempenho:

Figura 13 - *Confusion Matrix* FK2.



Fonte: Autoria Própria.

É possível ver na Figura 13 a matriz confusão do classificador *PassiveAggressiveClassifier*, representada em uma matriz 2x2. O classificador nesse *dataset* teve um total de 92,62% de acertos, onde o classificador previu que era falso e de fato era realmente falso e 92,33% de acertos em que o classificador previu verdadeiro e realmente era verdadeiro.

Com isso é possível analisar que houve uma falha causando um total de 7,38% de falsos positivos e um total de 7,67% de falsos negativos. De fato, o valor aumentou comparado ao *dataset* FK1, mas não de maneira significativa. A Tabela 6 mostra a performance geral do modelo:

Tabela 6 - Performance geral.

Accuracy	***Precision***	***Recall***
0.9274066280904787	0.927412335034098	0.9274066280904787

Fonte: Autoria Própria.

Nesse modelo, o algoritmo *PassiveAggressiveClassfieir* foi capaz de ter uma taxa total de mais de 92% de *accuracy*, *precision* e *recall*. O valor é medido entre 0 e 1, quanto mais próximo de 1 melhor é o modelo, ou seja, o resultado obtido está classificado como excelente para esse modelo nesse *dataset*.

O terceiro *dataset* utilizado para testar o algoritmo foi chamado de FK3, conforme descrito na tabela 7.

Tabela 7 - Informações *dataset* FK3.

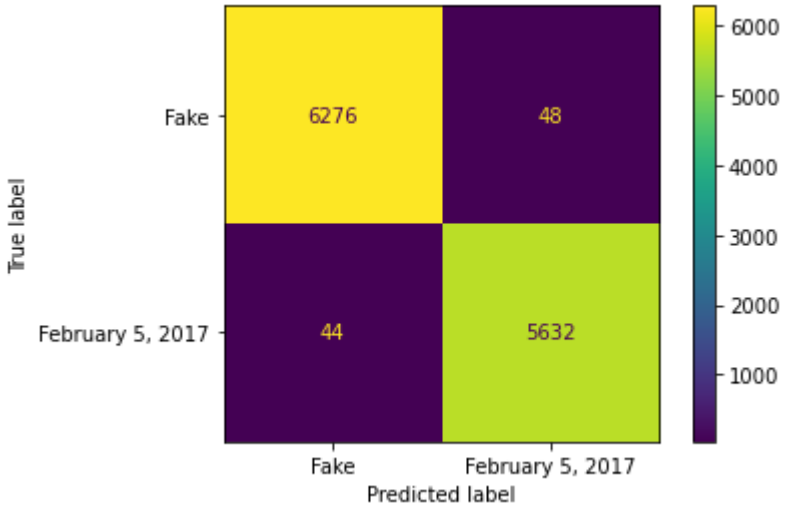
Shape	**Tamanho**	***NaN***	***Label***	**Tipo Arquivo**
(40.000, 6)	99.1 MB	0	FAKE ou REAL	.csv

Fonte: Próprio Autor.

O terceiro *dataset* é um arquivo .csv possui 40.000 linhas e 6 colunas (*id*, *title*, *author*, *text*, *date*, *label*). Como não havia dados nulos não foi necessário fazer o tratamento. Como saída, o *dataset* retorna duas formas de resultado: falso ou real.

O *dataset* foi dividido em 28.000 dados para treino e 12.000 para teste. Pode observar a seguir que a Figura 14 mostra os resultados do classificador *PassiveAggressiveClassifier* após utilização das principais métricas para medir seu desempenho:

Figura 14 - *Confusion Matrix* FK3.



Fonte: Autoria Própria.

É possível ver na Figura 14 a matriz confusão do classificador *PassiveAggressiveClassifier*, representada em uma matriz 2x2. O classificador neste *dataset* teve um total de 99,3% de acertos, onde o classificador previu que era falso e de fato era realmente falso e 99,2% de acertos em que o classificador previu verdadeiro e realmente era verdadeiro.

Desse modo, houve uma falha causando um total de 0,7% de falsos positivos e um total de 0,8% de falsos negativos. De fato, o valor de falhas é quase nulo. A seguir, a Tabela X mostra a performance geral do modelo:

Tabela 8 - Performance geral.

Accuracy	***Precision***	***Recall***
0.9923333333333333	0.9923333333333333	0.9923338295596362

Fonte: Autoria Própria.

Nesse modelo, o algoritmo *PassiveAggressiveClassfieir* foi capaz de ter uma taxa total de mais de 99% de *accuracy*, *precision* e *recall*. Como o *dataset* FK3 possuía uma quantidade maior de dados, o algoritmo ficou mais bem treinado. Dessa forma, é possível ver que sua performance é excelente, alcançando quase 100% de respostas certas.

O quarto *dataset* utilizado para testar o algoritmo foi chamado de FK4:

Tabela 9 - Informações *dataset* FK4.

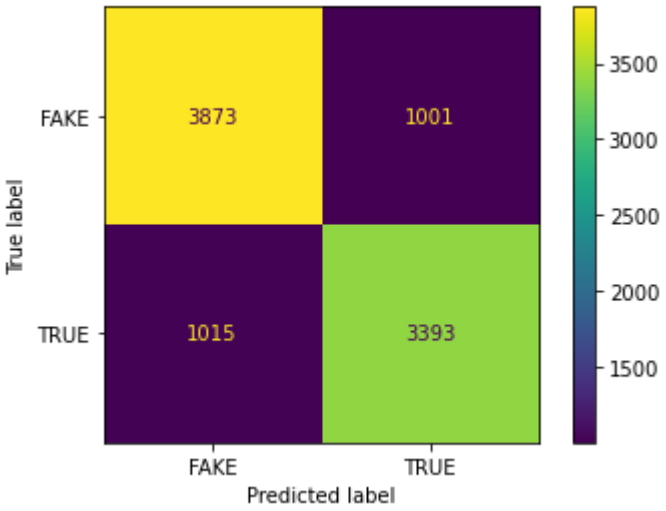
Shape	**Tamanho**	***NaN***	***Label***	**Tipo Arquivo**
(30.938, 4)	95,4 MB	0	FAKE ou TRUE	.csv

Fonte: Próprio Autor.

O segundo *dataset* é um arquivo *.csv*, possui 30.938 linhas e 4 colunas (*title*, *author*, *text*, *label*). Como não havia dados nulos não foi necessário fazer o tratamento. Como saída, o *dataset* retorna duas formas de resultado: falso ou verdade.

O *dataset* foi dividido em 21.656 dados para treino e 9.282 para teste. Abaixo, a Figura 15 com os resultados do classificador *PassiveAggressiveClassifier*, após a utilização das principais métricas para medir seu desempenho:

Figura 15 - *Confusion Matrix* FK4.



Fonte: Autoria Própria.

A Figura 15 mostra a matriz confusão do classificador *PassiveAggressiveClassifier*, representada em uma matriz 2x2. O classificador neste *dataset* teve um total de 79,23% de acertos, onde o classificador previu que era falso e de fato era realmente falso e 77,22% de acertos, em que o classificador previu verdadeiro e realmente era verdadeiro.

É possível analisar que houve uma falha causando um total de 20,77% de falsos positivos e um total de 22,78% de falsos negativos. A Tabela 10 mostra a performance geral do modelo:

Tabela 10 - Performance geral.

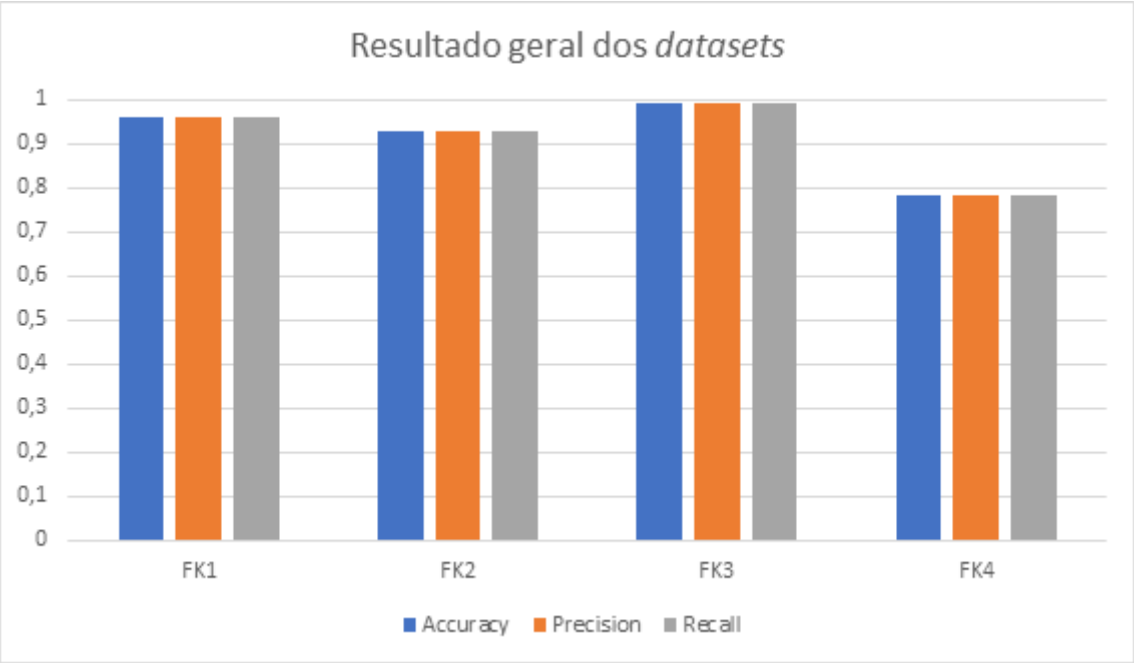
Accuracy	***Precision***	***Recall***
0.7828054298642534	0.7827750237409872	0.7828054298642534

Fonte: Autoria Própria.

Nesse modelo o algoritmo *PassiveAggressiveClassfieir* foi capaz de ter uma taxa total de menos de 90% de *accuracy*, *precision* e *recall*. O valor é medido entre 0 e 1, quanto mais próximo de 1 melhor é o modelo, ou seja, o resultado obtido com este *dataset* foi bom.

Para analisar o algoritmo de forma geral foi feito um gráfico comparando as principais métricas para análise de desempenho.

Figura 16 - Gráfico dos resultados gerais.



Fonte:Autoria Própria.

Na Figura 16 nota-se que o resultado de *Accuracy*, *Precision* e *Recall* são praticamente iguais, com pouca ou quase nenhuma diferença. Isso mostra que as métricas estão bem calibradas e não está havendo nenhum problema no processo.

A diferença entre os *datasets* FK1, FK2 e FK3 nota que são menores que 0.1, isso indica que o modelo tem se comportado bem com os diferentes tipos de *datasets* utilizados. É nítido que no *dataset* FK4 teve o pior desempenho dentre os demais, contudo, alcançou uma média de 0.78, isso indica que o modelo ainda é ideal.

5. CONSIDERAÇÕES FINAIS

Com as graves consequências que as *fake news* tem causado a nossa sociedade, esta pesquisa propôs um modo de classifica-las utilizando um modelo de aprendizado de máquina em conjunto com uma técnica de processamento de linguagem natural. Para classificar as notícias foi utilizado o algoritmo *PassiveAggressiveClassifier* e no processamento de linguagem natural foi utilizado o algoritmo *TfidfVectorizer*.

A abordagem utilizada para o desenvolvimento do projeto foi analisar as palavras que se repetiam com uma frequência máxima de sete vezes e utilizando o *TfdfVectorizer* para verificar o grau de importância de uma palavra contida no texto. Dessa forma o classificador seria capaz de analisar e verificar corretamente as palavras que teriam potencial para induzir uma informação a ser falsa ou verdadeira.

O treinamento foi feito utilizando quatro *datasets*, contendo todos obrigatoriamente *title*, *text* e *label*. Para realizar o treino os *datasets* foram divididos em 70% para treino e 30% para teste. Após isso os dados foram tratados com o *TfidfVectorizer* e passados pelo classificador *PassiveAggressiveClassifier* para obter os resultados.

Para os resultados foram utilizadas as principais métricas de medidas, sendo elas a *accuracy, prevision* e *recall*. Com os resultados obtidos foi possível analisar o quão eficiente o classificador era em relação a cada *dataset*. De acordo com Borges (2016), podemos classificar a precisão do algoritmo da seguinte forma:

- 0.9 - 1.0 > É considerado Excelente;
- 0.8 - 0.9 > É considerado Muito Bom;
- 0.7 - 0.8 > É considerado Bom;
- 0.6 - 0.7 > É considerado Suficiente;
- 0.6 - 0.5 > É considerado Ruim;
- Menor que 0.5 > Não utilizar o teste;

Na primeira base de dados, chamada de FK1, foi possível obter um resultado positivo pois de forma geral as métricas mediram uma taxa maior que 96% que indica que o modelo para esse caso é ótimo. No *dataset* FK2, a taxa medida foi um

pouco menor sendo ela maior que 92% e também indicando para esse caso o modelo ainda é ótimo. No *dataset* FK3 foi obtido o melhor resultado dentre as bases de dados escolhidas, sua taxa geral foi maior que 99%, o que para esse caso o modelo foi ótimo e mais preciso que nas outras bases de dados. E por fim o *dataset* FK4 apresentou uma taxa maior que 78%, a menor dentre todas as bases de dados escolhidas, porém como o valor passa de 70% ainda é considerado como um ótimo resultado para o modelo.

Em síntese, foi possível analisar que de forma geral o modelo de aprendizado de máquina *PassiveAggressiveClassifier* juntamente com o processamento de linguagem natural obtiveram resultados positivos para realizar a classificação de *fake news.* Dada a relevância do tema na atualidade e a importância da detecção de notícias como verdadeiras ou falsas para a diminuição de seu impacto na sociedade, com os resultados obtidos nesse trabalho é possível concluir que o modelo pode ser uma alternativa para solucionar esse problema.

REFERÊNCIAS

AHMED, Hadeer; TRAORE, Issa; SAAD, Sherif. Detection of online fake news using n-gram analysis and machine learning techniques. In: **International conference on intelligent, secure, and dependable systems in distributed and cloud environments**. Springer, Cham, 2017. p. 127-138.

ALMEIDA, Carlos Caetano de. **Identificação e Classificação de Imagens usando Rede Neural Convolucional e Machine Learning: Implementação em Sistema Embarcado**. 202 f. Tese (Doutorado) - Curso de Engenharia Mecânica, Universidade Estadual de Campinas, Campinas, 2019.

AMARAL, André Sampaio do. **Uma Metodologia Orientada a Dados para Precificação de Imóveis**. 2018. Trabalho de Conclusão de Curso. Universidade Federal do Rio Grande do Norte.

APHIWONGSOPHON, Supanya; CHONGSTITVATANA, Prabhas. Detecting fake news with machine learning method. In: **2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)**. IEEE, 2018. p. 528-531.

BORGES, L. S. Medidas de Acurácia diagnóstica na pesquisa cardiovascular. **Int J Cardiovasc Sci**, v. 29, n. 3, p. 218-22, 2016.

BORGES, Luiz Eduardo. **Python para desenvolvedores: aborda Python 3.3**. Novatec Editora, 2014.

CALDAS, Weslley Lioba. **Proposta de dois métodos semi-supervisionados baseados na máquina de aprendizagem mínima utilizando Co-Training.** 2017. 59 f. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal do Ceará, Fortaleza, 2017.

CARDOSO, Ivelise de Almeida. **Propagação e influência de pós-verdade e fake news na opinião pública**. 2019. Dissertação (Mestrado em Interfaces Sociais da Comunicação) - Escola de Comunicações e Artes, Universidade de São Paulo, São Paulo, 2019. Acesso em: 2020-09-21.

CARVALHO, M. H. D. Estudo comparativo dos métodos de word embedding na análise de sentimentos. 2018.

CHANG, Winston. **R graphics cookbook: practical recipes for visualizing data**. " O'Reilly Media, Inc.", 2012.

FELICIANO, Matheus Barbosa Domingues. **Classificação e Predição de Preços de Imóveis a Partir de Dados Estruturados e Não Estruturados**. 2018. Tese de Doutorado. Universidade Federal de Pernambuco.

GRANIK, Mykhailo; MESYURA, Volodymyr. Fake news detection using naive Bayes classifier. In: **2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)**. IEEE, 2017. p. 900-903.

GRUPPI, Mauricio; HORNE, Benjamin D.; ADALI, Sibel. An Exploration of Unreliable News Classification in Brazil and The US. **arXiv preprint arXiv:1806.02875**, 2018.
GUERREIRO, Lucas. **Aprendizado semi-supervisionado utilizando modelos de caminhada de partículas em grafos.** 2017.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep learning**. [*S.l.*]:MIT press, 2016

HOPPEN J., **Datasets, o que são e como utilizá-los**. 2018. Disponível em: https://www.aquare.la/datasets-o-que-sao-e-como-utiliza-los/. Acesso em: 15 nov. 2020.

KLUYVER, Thomas *et al*. Jupyter Notebooks-a publishing format for reproducible computational workflows. *In*: ELPUB. [*S.l.:s.n*.], 2016. p. 87–90

KUMAR, Vipin; SUBBA, Basant. A TfidfVectorizer and SVM based sentiment analysis framework for text data corpus. In: **2020 National Conference on Communications (NCC)**. IEEE, 2020. p. 1-6.

LAZER, David MJ et al. The science of fake news. **Science**, v. 359, n. 6380, p. 1094-1096, 2018.

MACARIO FILHO, Valmir; de Assis Tenório Carvalho, Francisco. **Um novo algoritmo de agrupamento semisupervisionado baseado no Fuzzy C-Means.** 2009. Dissertação (Mestrado). Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Pernambuco, Recife, 2009.

MATSUBARA, Edson Takashi. **O algoritmo de aprendizado semi-supervisionado co-training e sua aplicação na rotulação de documentos**. 2004. Tese de Doutorado. Universidade de São Paulo.

MCKINNEY, Wes et al. pandas: a foundational Python library for data analysis and statistics. **Python for High Performance and Scientific Computing**, v. 14, n. 9, 2011.

PORTO FILHO, Carlos Humberto. **Técnicas de aprendizado não supervisionado baseadas no algoritmo da caminhada do turista**. 2017. Dissertação (Mestrado em Bioengenharia) - Bioengenharia, Universidade de São Paulo, São Carlos, 2017.

SANCHES, Marcelo Kaminski. **Aprendizado de máquina semi-supervisionado**: proposta de um algoritmo para rotular exemplos a partir de poucos exemplos rotulados. 2003. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2003.

SCIKIT-LEARN. **About us**. Disponível em: https://scikit-learn.org/stable/about.html. Acesso em: 26 nov. 2020.

SIRAJUDEEN, Sakeena M.; AZMI, Nur Fatihah A.; ABUBAKAR, Adamu I. ONLINE FAKE NEWS DETECTION ALGORITHM. **Journal of Theoretical & Applied Information Technology**, v. 95, n. 17, 2017.

SOUSA, Maria Cristina Cordeiro Sousa. **Uma análise do algoritmo K-means como introdução ao aprendizado de máquinas**. 2020.

TOOMEY, Dan. **Jupyter for data science: Exploratory analysis, statistical modeling, machine learning, and data visualization with Jupyter**. Packt Publishing Ltd, 2017.

VARELLA, C. A. A. **ANÁLISE MULTIVARIADA APLICADA AS CIÊNCIAS AGRÁRIAS PÓS-GRADUAÇÃO EM AGRONOMIA CIÊNCIA DO SOLO**: CPGA-CS ANÁLISE DISCRIMINANTE. 2004.

VISA, Sofia et al. Confusion Matrix-based Feature Selection. **MAICS**, v. 710, p. 120-127, 2011.

VOSOUGHI, Soroush; ROY, Deb; ARAL, Sinan. The spread of true and false news online. **Science**, v. 359, n. 6380, p. 1146-1151, 2018.

WALT, Stéfan van der; COLBERT, S. Chris; VAROQUAUX, Gael. The NumPy array: a structure for efficient numerical computation. **Computing in science & engineering**, v. 13, n. 2, p. 22-30, 2011.

WANG, Peter.Popular Data Science Platform. [*S.l.:s.n*.], 2012. Disponível em:https://anaconda.com.

WANG, Zhuang; VUCETIC, Slobodan. Online passive-aggressive algorithms on a budget. In: **Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics**. 2010. p. 908-915.