ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SISTEMA DE DETECÇÃO E CLASSIFICAÇÃO DE NOTÍCIAS FALSAS COM APRENDIZADO DE MÁQUINA

Gabriel dos Anjos Dantas Teixeira

São Paulo - SP

Dezembro de 2022

Catalogação-na-publicação

Teixeira, Gabriel
SISTEMA DE DETECÇÃO E CLASSIFICAÇÃO DE NOTÍCIAS FALSAS COM APRENDIZADO DE MÁQUINA / G. Teixeira -- São Paulo, 2022.
37 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Telecomunicações e Controle.

1.Processamento de linguagem natural 2.Aprendizado de máquina 3.Notícias falsas 4.Análise de dados I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Telecomunicações e Controle II.t.

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SISTEMA DE DETECÇÃO E CLASSIFICAÇÃO DE NOTÍCIAS FALSAS COM APRENDIZADO DE MÁQUINA

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Graduação em Engenharia.

Gabriel dos Anjos Dantas Teixeira

Orientador: Prof. Dr. Felipe Miguel Pait

Área de Concentração:
Engenharia Elétrica

São Paulo – SP

Dezembro de 2022

RESUMO

Este projeto busca apresentar uma ferramenta de combate a *fake News*, na forma de um website capaz de discriminar um texto jornalístico entre possivelmente falso ou verdadeiro. Através de ferramentas de processamento natural de linguagem e aprendizado de máquina, busca-se criar um modelo que contenha em seu núcleo uma estrutura padrão característica de notícias falsas, utilizando-se do vocabulário e frequência de uso de termos para tornar possível essa distinção.

São apresentadas todas as etapas seguidas, desde a obtenção da base, até o tratamento das informações, visualização dos dados, montagem do modelo e avaliação dos resultados, bem como conclusões acerca deste.

Ao fim do projeto, foi possível obter tal modelo, alcançando-se uma precisão de cerca de 95%.

ABSTRACT

This project seeks to present a tool to combat fake news, in the form of a website capable of discriminating a journalistic text between possibly false or true. Through natural language processing and machine learning tools, we seek to create a model that has at its kernel a standard structure usually characteristic of fake news, acquired through the vocabulary and frequency of use of certain terms, to make this distinction possible.
All the steps followed are presented, from obtaining the base, to the treatment of information, data visualization, assembly of the model and evaluation of the results, as well as conclusions about it.
At the end of the project, it was possible to obtain such a model, reaching an accuracy of about 95%.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1.INTRODUÇÃO

1.1. Motivação e Contextualização

O termo *fake news* ganhou popularidade nos últimos anos, em especial após as eleições para presidente nos Estados Unidos. O candidato Donald Trump lançava uso da palavra para desqualificar a candidatura de seus adversários, acusando-os de forjar mentiras que os favorecessem. Processo semelhante ocorreu durante o *Brexit*, no Reino Unido.

Essa expressão vem do inglês “notícia falsa”, e trata da produção e difusão de conteúdo mentiroso ou propositalmente distorcido, que busca utilizar de inverdades para manipular a opinião pública.

No Brasil, as *fake news* foram especialmente prejudiciais durante o período da pandemia do coronavírus: entre mentiras como implantação de *chips* através da vacina e métodos milagrosos de cura, a comunidade médica e científica lutou contra notícias sem qualquer embasamento, com o objetivo claro de desinformar e até mesmo favorecer candidatos (1).

Estudos indicam uma relação entre esse fenômeno e públicos simpáticos às correntes políticas de direita. Três principais motivos endossam essa afirmação: As organizações são mais hierarquizadas (têm linhas de comando verticais mais claras), lançam mão de mais recursos financeiros, e usam mensagens mais simplificadas, que atingem melhor o público. Dessa forma, um público que adota essa ideologia tem mais tendência a compartilhar esse tipo de conteúdo. Assim, é impossível separar o estudo de *Fake News* e as estruturas que as compõem de um estudo da forma como essa extrema direita se comunica (2).

Uma possível forma de detectar automaticamente uma notícia falsa é a utilização do aprendizado de máquina, ou *machine learning*. Esse tema é muito abordado na literatura científica e já possui aplicações reais. Sites como o Facebook e YouTube usam ferramentas de inteligência artificial para catalogar conteúdo potencialmente duvidoso ou mentiroso. Um processo capaz de identificar relações e padrões entre diferentes notícias, catalogadas entre verdadeiras e falsas, é capaz de detectar com alguma confidência à qual classe aquele artigo pertence (3).

A utilização computação para processamento e interpretação de textos é conhecida como Processamento Natural de Linguagem, ou *Natural Language*

Processing (NLP), e é um conjunto de técnicas que busca garantir a capacidade de processar linguagem de forma próxima à humana (4).

Busca-se nesse projeto desenvolver um algoritmo capaz de aprender com esses padrões, à partir de uma base previamente catalogada, e criar um website de uso fácil, que possa ser utilizado para rápida conferência de notícias encontradas pela internet.

1.2. Objetivos

O projeto consiste na criação de um modelo de *machine learning* capaz de classificar com alguma certeza se uma notícia em português é verdadeira ou potencialmente falsa.

Será utilizada uma base de notícias previamente curada e catalogada, extraídas de diferentes sites disponíveis na internet. A base será adequadamente processada e preparada para garantir uma boa análise dos dados e, posteriormente, aplicação do algoritmo de aprendizado.

Uma ferramenta estatística muito utilizada no campo do NLP é o TF-IDF. Esse recurso será utilizado para atribuir valores (*scores*) aos termos, possibilitando localizar padrões entre que associam o vocabulário de uma notícia à sua classificação.

À partir desses dados, serão criados e testados modelos que utilizam diferentes métodos de aprendizado, com base em estudos existentes na literatura sobre o tema. Através da comparação destes, utilizando métricas adequadas, o melhor método será utilizado como base para criação de um Website.

O website deve ser de fácil uso e possibilitar a rápida aplicação do modelo para classificar uma notícia.

1.3. Trabalhos anteriores

Ao longo desse documento são referenciados diversos trabalhos que serviram como base. Entretanto, destaca-se a pesquisa e projeto apresentados por Sekiguchi, Tsuji e Gonçalves (5), do qual fiz parte durante as etapas iniciais. Algumas das conclusões foram consideradas para reger escolhas apresentadas nesse desenvolvimento.

As principais são a escolha de um algoritmo TF-IDF para evitar gerar viés com o tamanho dos textos da base, uma vez que este é um parâmetro normalizado (ao contrário do bag of words, utilizado no projeto citado). Também foi desconsiderada a classificação com uso de parâmetros semânticos, por não ter apresentado bons resultados.

2.BASE TEÓRICA

2.1.Fake News

Apesar do termo Fake News ser normalmente utilizado para descrever conteúdo enganoso, não é sempre necessário haver intenção maliciosa para enganar o leitor. Diferentes interpretações de um mesmo conteúdo podem gerar interpretações equivocadas da realidade, associados à forma como este conteúdo é disseminado e apresentado.

Nesse contexto, torna-se difícil determinar com precisão o que é uma notícia falsa. Algumas fontes descrevem como sendo desinformação distribuída em larga escala, usualmente em redes sociais (6). Também podem ser considerados artigos com conteúdo fabricado (7).

Alguns autores separam *fake news* em sete classificações: sátira (enganoso sem intenção maliciosa), conexão falsa (primeira impressão que difere da realidade), conteúdo enganador (uso da informação para enganar ativamente e de forma prejudicial), contexto falso (informação válida em contexto irreal), conteúdo impostor (utilização e falsificação de fontes renomadas), conteúdo manipulador (montado para enganar o consumidor) e conteúdo fabricado (informação totalmente falsa, sem base, com intenção maliciosa e viés) (8).

2.2.1. Fact-checking

O termo *fact-checking* refere-se à ação de checar a veracidade de uma notícia. Esse ato é várias vezes relacionado a *fake news*, devido ao contexto. No Brasil, há várias e diferentes agências realizando esse trabalho manualmente, entre elas a Lupa e a Aos Fatos. A Agência Pública também era uma delas, mas seu projeto, Truco, foi encerrado em 2018. Todas utilizam uma metodologia parecida, em que trechos específicos da informação são verificados por especialistas. Essas frases são escolhidas pela chance de conterem informação falsa, dependendo de sua relevância ou impacto.

O método, no entanto, não é à prova de falhas. Estudos indicam que essa metodologia não pode ser considerada científica, e testa que cerca de 20% das análises de duas grandes agências estadunidenses tiveram classificações conflituosas (9).

Além das agências de *fact-checking*, há também um serviço criado por pesquisadores da USP e da UFSCar chamado *Fake Check*. O aplicativo classifica notícias utilizando aprendizagem de máquina com base apenas em seu texto. O projeto utilizou dois métodos analíticos: estudo da morfologia das frases e uso do *Bag of Words*, atingindo acurácia de cerca de 90% (10) .

2.2.Processamento de Linguagem Natural

O processamento de linguagem natural (Natural Language Processing - NLP) refere-se à área em comum entre computação e linguística que visa permitir a interpretação de textos e linguagens por computadores. Com suas técnicas, é possível montar sistemas que entendem e/ou geram a linguagem humana (teste de Turing, tradutores automáticos, bots de resposta) (11).

A biblioteca Spacy permite implemtação simples e quase automática de NLP em um projeto. Das componentes dessa biblioteca, destacam-se para o projeto:

- *Tokenizer*: divide o texto em partículas (tokens), como palavras, números, pontuações;
- *Part-of-speech (POS-tagging)*: faz a classificação das palavras levando em consideração a relação entre elas no texto e suas posições na frase;
- *Morphologizer*: faz a classificação morfológica das palavras;
  Lemmatizer: devolve a forma base da palavra (forma encontrada no dicionário);
- *Stemmer*: retira sufixos e/ou prefixos da palavra (aproximação da raiz da palavra);
- *Stop Words*: palavras que não agregam muito significado à frase, podendo ser retiradas sem muita perda de sentido;
- *Entity Recognizer* (ER): detector de entidades, como organizações, pessoas, países.

2.2.1. Token

Refere-se à conversão do conteúdo em palavras individuais (“Tokens”). Existem algumas formas de tokenização possíveis, como a quebra em unigramas (unidades de uma palavra) e digramas (unidades de duas palavras) (12). O objetivo é “quebrar” o texto em partes suficientes para serem analisadas sem perder informação, mas sem uma quantidade extremamente grande de dados, de forma que o espaço dimensional torne-se muito grande, dificultando o treino. A biblioteca de tokenização do *Natural Language Toolkit* costuma ser suficiente.

2.2.2. Vetorização

A vetorização consiste na transformação do texto em um vetor analisável, utilizando como elementos desse vetor os *tokens*, explicados anteriormente.

2.2.3. Lematização

A lematização é o processo, efetivamente, de deflexionar uma palavra para determinar o seu lema (as flexões chamam-se lexemas). Por exemplo, as palavras gato, gata, gatos, gatas são todas formas do mesmo lema: gato.1 Igualmente, as palavras tiver, tenho, tinha, tem são do mesmo lema ter. E bom, melhor e ótimo são lexemas do lema bom.

A lematização é útil quando queremos ver os usos de palavras em contextos sem importância das flexões. Por exemplo, para a criação e uso de índices ou na investigação linguística. A lematização também leva em consideração o contexto da palavra para resolver outros problemas, como a desambiguação, o que significa que ela pode discriminar entre palavras idênticas que têm significados diferentes, dependendo do contexto específico (13).

Dessa forma, é uma ferramenta muito útil para normalização de bases de texto, reduzindo o dicionário de palavras a serem analisadas. A biblioteca Spacy também conta com uma boa ferramenta de lematização em português.

2.2.4. Stop words

Stop words são palavras que podem ser consideradas irrelevantes para uma frase na análise do contexto de uma frase por NLP. São recorrentes no texto e não possuem grande valor quando tratando-se da análise de grandes quantidades de dados, uma vez que não trazem consigo muita informação. Exemplos são “de”,”que”,”para”,”não”. (14) Geralmente são elementos de ligação, negação ou pronomes. Existem diversas listas de *stop words* disponíveis na internet para o português, e não há um consenso ou definição sobre quais palavras se enquadram nessa categoria. Na tabela 1, são referenciadas algumas *stop words* da biblioteca NLTK:

'de', 'a', 'o', 'que', 'e', 'é', 'do', 'da', 'em', 'um', 'para', 'com', 'não', 'uma', 'os', 'no', 'se', 'na', 'por', 'mais', 'as', 'dos', 'como', 'mas', 'ao', 'ele', 'das', 'à', 'seu', 'sua', 'ou', 'quando', 'muito', 'nós', 'já', 'eu', 'também', 'só', 'pelo', 'pela', 'até', 'isso', 'ela', 'entre'

Tabela 1: Exemplo de stopwords

2.2.5 Bag of words

O Bag of Words é uma forma de representar o texto de acordo com a ocorrência das palavras nele. Traduzindo para o português, o “saco de palavras” recebe esse nome porque não leva em conta a ordem ou a estrutura das palavras no texto, apenas se ela aparece ou a frequência com que aparece nele. (15)

Nesse método estatístico, cada palavra tem um valor associado, que é o número de aparições no texto ou *corpus*. É uma ferramenta muito utilizada para classificação de textos com *Machine Learning*. (16) Um exemplo de aplicação de *bag of words* é demonstrado na tabela 2.

.

Texto inicial	Bag of words do texto
“A minha gata é a mais folgada”.	“A”: 2 “minha”: 1 “gata”: 1 “é”: 1 “a”: 1 “mais”: 1 “folgada”: 1

Tabela 2: Tabela 2: Exemplo de bag of words

2.2.6. TF-IDF

A técnica do TF-IDF informa sobre a frequência de aparição de um termo levando em consideração o balanço adequado entre a significância local e o conjunto de documentos analisados (17).

O *Term Frequency* (TF) é a frequência de aparição de um termo em um documento (texto) do *corpus*, conforme a equação 1 (18).

$$tf(t,d) = \frac{f_d(t)}{\max\limits_{w \in d} f_d(w)} \quad (1)$$

O *Inverse Document Frequency* (IDF) é razão entre o número de documentos processados Nd e o número de documentos contendo pelo menos uma ocorrência de um termo, expressa pela fórmula 2.

$$idf(t,D) = \log \frac{D}{\{d \in D : t \in d\}} \quad (2)$$

A multiplicação dos dois termos leva à função geral que descreve o TF-IDF, demonstrado na equação 3.

$$tfidf = tf * idf \quad (3)$$

A aplicação do TF-IDF em uma base resulta em um *score* por palavra, semelhante ao "peso" presente no Bag of Words. Esse valor é utilizado para comparações e encontrar padrões.

A utilização do TF-IDF é descrita como promissora para detecção de notícias falsas quando em conjunto com um classificador passivo-agressivo (19).

2.3. Aprendizado de Máquina

O aprendizado de máquina é um ramo da evolução de algoritmos computacionais projetados para emular a inteligência humana aprendendo com o ambiente e contexto. Eles são considerados a força motriz na nova era do chamado big data. Técnicas baseadas em aprendizado de máquina foram aplicadas com sucesso em diversos campos, desde reconhecimento de padrões, visão computacional, engenharia espacial, finanças, entretenimento e biologia computacional até aplicações biomédicas e médicas (20). No caso, este será utilizado para a classificação das notícias em

verdadeiras ou falsas. Essa classificação é feita por um modelo, que decide o resultado a partir de características (*features*) extraídas do texto. Para o projeto, as *features* em questão serão os *scores* TF-IDF de cada palavra.

2.3.1. Modelos

Foram utilizados modelos de classificadores que já foram utilizados em estudos de detecção de notícias falsas com NLP. Estes são o classificador passivo-agressivo (19), um classificador por *boosting* (21) e a *One-Class Suport Vector Machine* (OC-SVM) (22).

2.3.1.1. *One-class SVM*

Support Vector Machines são um dos vários algoritmos existentes para aprendizado de máquina. Essa ferramenta, utilizada tanto para problemas de classificação quanto regressão, data de 1995 e é de grande recorrência no campo da ciência de dados (23). É provado que, para um conjunto de dados de treino separável por um hiperplano, a complexidade deste plano pode ser definida como uma quantidade, um valor chamado "margem". Essa define a distância mínima entre um dado de exemplo e uma superfície de decisão. Um algoritmo de SVM busca minimizar o risco na decisão a partir da maximização da margem, permitindo que os pontos estudados sejam separados.

Uma categoria de SVM é o *One-Class SVM*. Esse tipo de implementação tem como objetivo detectar anomalias em dados com classificação binária. Dessa forma, as classificações são divididas apenas em "normal" e "anormal", com o *dataset* contendo apenas valores correspondendo ao primeiro tipo. Esse tipo de abordagem é muito útil em casos onde o evento anormal é raro (24).

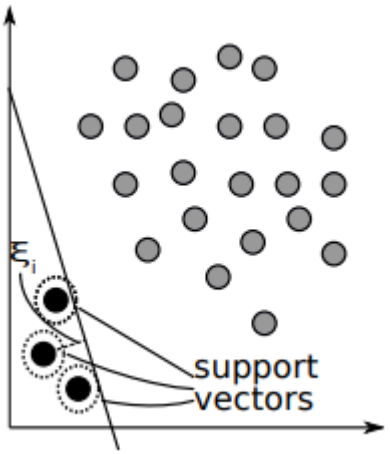

Figura 1: Exemplo bidimensional de espaço após aprendizado utilizando one-class SVM

A figura 1 demonstra um exemplo de conjunto dividido por um plano, onde os pontos cinzas representam outliers. Esses seriam os valores considerados “anormais”. A implementação desse método no problema proposto nesse projeto é válida, visto que é mais fácil encontrar textos jornalísticos verdadeiros (por exemplo, através de sites de notícias renomados) do que artigos com conteúdo falso.

2.3.1.2. Métodos Ensemble de classificação

Métodos Ensemble são uma das ferramentas disponíveis no campo do aprendizado de máquina. Esse método consiste na utilização de diferentes classificadores para obtenção de um consenso, que é utilizado para definir a classe do item estudado. A combinação de vários modelos através do ensemble não garante um modelo com uma performance melhor do que suas partes separadas, mas reduz muito a chance de gerar um modelo com performance ruim. Dentro desses algoritmos, existem três categorias (25)

- *Bagging*: Divisão da base de treino em amostras treinadas com diferentes algoritmos e unidas posteriormente por votação (26).
- *Boosting*: Execução de uma mesma base de treinos para vários *weak learners* (algoritmo com desempenho pouco melhor que a escolha aleatória), seguido de uma atribuição de pesos valorizando os *weak learners* com melhor resultado. Esses pesos são utilizados na escolha da classificação final (27).

- *Stacking*: Método que consiste em aprender um conjunto de metadados ao invés de aprender um conjunto de dados original.

2.3.1.3. *Passive agressive classifier*

O classificador passivo agressivo é um algoritmo útil para lidar com grandes conjuntos de dados. O modelo é atualizado através de múltiplas iterações. Em cada uma delas, os pesos associados a um ponto são reavaliados caso a classificação esteja incorreta, ou mantidos caso esteja certa. Esse tipo de processamento é classificado como "*online*", pois os dados entram e são processados sequencialmente. Vários estudos citam este classificador como útil para analisar textos de redes sociais, por tratarem-se de bases grandes e com milhares de *features* (28) (29).

2.3.2.Avaliação dos modelos

- *Cross-validation*

Cross Validation é um método de treino e teste do modelo. O método mais comum é o *K-Fold*: A base é dividida em treino (usualmente 80%) e teste (20%), o algoritmo é executado e as métricas desejadas obtidas. Isso é repetido até todos os valores da base serem utilizados tanto para teste quanto para treino. Ao término, as métricas obtidas são agregadas por média e apresentadas. (30)

- *Overfitting e undefitting*

Overfitting é nome dado à situação onde o modelo descreve muito bem a base de treino, mas tem resultado ruim com a base de teste. Indica que o algoritmo é muito específico aos dados utilizados. *Underfitting* é o oposto: indica um modelo muito genérico, que não descreve bem os dados de treino. A figura 2 ilustra a ambas situações:

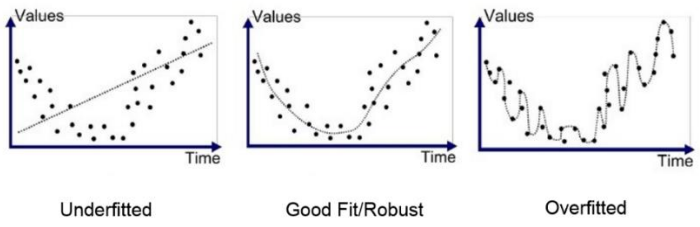

Figura 2: Representação de dados underffited, robustos e overfitted

- Maldição da dimensionalidade

Problema relacionado com a utilização de dados compostos por grande quantidade de *features* (dimensões). Isso pode trazer problemas tanto para visualização e análise das informações, como na etapa de treino, com o aumento da complexidade e poder computacional exigido para processamento dos cálculos do modelo. Costuma ser resolvido através de métodos de diminuição dimensional, como análise do discriminante linear. (31)

- Métricas

O estudo e aplicação de métricas de desempenho são parte crucial no estudo e implementação do aprendizado de máquina. A seguir, será apresentada uma breve descrição do conceito de matriz de confusão, um método muito comum para classificadores binários, mas que pode ser estendido para avaliações multi classe.

A matriz de confusão apresenta duas colunas e duas linhas, que distinguem os resultados da classificação. As linhas representam a característica real do dado, podendo ser esta verdadeira (caso o dado se encaixe na categoria) e falsa (caso não se encaixe). As colunas apresentam o resultado do modelo, sendo verdadeiro caso o algoritmo considere aquele ponto como pertencente à classe, e falso caso não considere. Dessa forma, chegamos na seguinte permutação entre colunas e linhas:

- Classe real e atribuída verdadeiras: verdadeiro positivo (*True Positive* - TP);
- Classe real verdadeira e atribuída falsa: falso negativo (*False Negative* - FN);
- Classe real falsa é atribuída verdadeira: falso positivo (*False Positive* - FP);
- Classe real e atribuída falsas: verdadeiro negativo (*True Negative* - TN).

	Verdadeiro (modelo)	Falso (modelo)
Verdadeiro (real)	TP	FN
Falso (real)	FP	TN

Tabela 3: Matriz de Confusão

A partir dessa tabela, surgem as principais métricas para classificadores (32).

A acurácia, definida como número de acerto sobre a amostra total, é a métrica mais utilizada por ser simples e de grande valor. Esta é descrita pela relação:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Existem, entretanto, alguns dos pontos negativos em utilizar apenas a acurácia como fonte de verdade, uma vez que uma das suas limitações é produzir valores menos distintivos e discrimináveis, além de pesar menos a favor de classes minoritárias. Dessa forma, em um caso com poucos exemplos de pontos pertencentes a determinado grupo, a acurácia pode não ser adequada (33)

A precisão mede os valores positivos previstos corretamente dentre toda a classe de resultados positivos. Uma precisão alta indica baixo número de falsos positivos, mas não necessariamente boa acurácia. É modelado como:

$$Pre = \frac{TP}{TP + FP}$$

O *recall* mede o número de positivos avaliados corretamente sobre o mesmo valor somado com os falsos negativos. Um bom recall indica baixa quantidade de falsos negativos. É descrito pela equação:

$$Rec\ = \frac{TP}{TP + FN}$$

A especificidade mede o número de negativos avaliados corretamente sobre o mesmo valor somado com os falsos negativos. Uma boa especificidade indica baixa quantidade de falsos negativos. É descrito pela equação:

$$Spe\ = \frac{TN}{TN + FP}$$

O valor F (*F-Measure*) é a média harmônica entre recall e precisão. A literatura atribui a essa métrica um melhor desempenho na avaliação de dados binários (33):

$$FM = \frac{2 * p * r}{p + r}$$

A média geométrica (GM) é usada para maximizar a taxa de verdadeiros positivos e verdadeiros negativos, mantendo ambos relativamente balanceados. Também existem casos reportados de desempenho elevado em problemas de classificação binários (33):

$$GM = \sqrt{Rec * Spe}$$

3. METODOLOGIA

A abordagem do projeto consistiu na divisão em algumas etapas, seguindo o pipeline de aquisição, análise e modelagem de dados, conforme a figura 3. Segue uma descrição resumida de cada bloco do processo:

- Etapa de aquisição de dados: Consiste tanto na utilização do *corpus* Fake.br, quanto na obtenção de novas notícias que serão agregadas ao *dataset*.
- Etapa de integração: Refere-se à implementação de utilização de um banco de dados para facilitar o acesso às informações.
- Etapa de limpeza e transformação: Tratamento da base de treino através das técnicas apresentadas, como pré-processamento, extração, remoção de *stopwords* e *stemming*, bem como da aplicação do algoritmo de TF-IDF.
- Modelagem: Implementação dos modelos de aprendizagem propostos e ajuste dos hiperparâmetros.
- Exploração e interpretação: Processo paralelo à modelagem, trata da avaliação dos dados e resultados do modelo para aprimorar o desempenho do algoritmo.
- Produto: Desenvolvimento da forma de apresentação ao usuário final

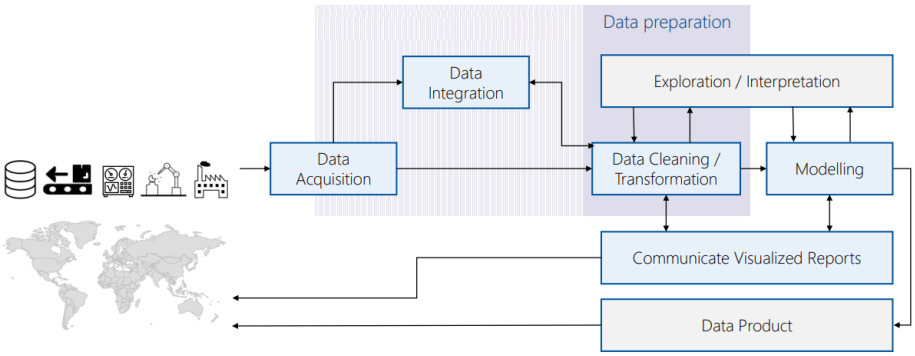

Figura 3: Pipeline de data analytics

4. DESENVOLVIMENTO

4.1. Base de dados

4.1.1. Fake.br Corpus

O Fake.br Corpus, chamado de Corpus no decorrer do projeto, é uma base de dados contendo notícias verdadeiras e falsas em português brasileiro. Essa base consiste de 7200 notícias, sendo 3600 de cada tipo, retiradas dos domínios (*sites*) listados nas tabelas 4 e 5.

Notícias falsas	Notícias verdadeiras
Diário do Brasil	G1
A Folha do Brasil	Folha de São Paulo
The Jornal Brasil	Estadão
Top Five TV	

Tabela 4: Domínios de origem das notícias do Fake.br Corpus

O corpus é resultado de um *paper* cujo objetivo foi criar uma base de notícias verdadeiras e falsas brasileiras, para estudo sobre *fake news*. Este foi obtido através de um processo de *crawling* (iteração em páginas na internet) seguido de uma curadoria intensa, e os artigos verdadeiros costumam tratar dos mesmos temas dos artigos falsos, garantindo balanceamento. (34)

4.1.2. Tratamento da base

Após a obtenção de ambas as bases, foi feito um tratamento que consiste em retirar notícias e partes do texto que não importam, como notícias repetidas ou em outra língua que não português brasileiro, textos em vazio ou anúncios. Foram utilizados SQL e Python para esta parte do projeto.

4.2. Pré processamento dos textos

A etapa de pré-processamento dos textos é crucial para garantir a normalização e bom funcionamento do algoritmo de classificação. Para esse projeto, foi dividida em três etapas: pré-processamento, remoção de palavras comuns na língua portuguesa (*stop words*) e lematização (35).

O pré-processamento é utilizado para remover caracteres que não sejam úteis ao algoritmo de aprendizado e não agreguem valor à análise. Para garantir a normalização, as palavras também são todas convertidas para letras minúsculas. No bloco à seguir, é demonstrada a função responsável por esse comportamento: a primeira linha converte todos os caracteres para *lowercase* (minúsculo), a segunda elimina o caractere de quebra de linha e a terceira é uma expressão do tipo *RegEx* (*regular expression*), que permite remover todas as letras que não estejam especificadas.

```
      def pre_process_text(text):
          lowercase = text.lower()
          filter_line_breaks = lowercase.replace("\n", " ")
          filter_letters = "".join(re.findall("[a-zA-Z
áàâãéèêíïóôõöúçñ]+", filter_line_breaks))
          return filter_letters
```

À seguir, é aplicada a remoção das *stopwords*. Estas são descritas como palavras muito comuns na língua portuguesa, que não possuem grande valor para a análise de um texto, conforme apresentado na etapa de estudos. Como não há um consenso sobre quais são essas palavras, é necessário utilizar bibliotecas de NLP e artigos na Internet disponibilizam listas para serem utilizadas. No caso do projeto, duas seleções foram utilizadas: uma disponibilizada na biblioteca SpaCy, e outra encontrada no repositório *GitHub* (36).Segue o trecho responsável pela função de remoção dessas palavras.

```
nltk_stopwords = stopwords.words('portuguese')
word_dict = {}
for word in list_words + nltk_stopwords:
    if(not word  in word_dict):
        word_dict[word]=0
set_stopwords = word_dict.keys()

def stopword_removal(text):
    splitted_text = text.split(' ')
    stopword_list = set_stopwords
    clean_text = ""
    for word in splitted_text:
        if not word.strip() in stopword_list:
            clean_text = str(clean_text) + word + " "
    return clean_text
```

Por fim, a aplicação da lematização é de grande importância para reduzir o tamanho do vetor de *feature*. A técnica utiliza uma base pré-treinada para reduzir palavras à sua forma canônica, considerando seu sentido na frase. A biblioteca SpaCy possui uma ferramenta robusta de lematização em língua portuguesa, treinada com um *corpus* de artigos jornalísticos. Além disso, torna fácil a aplicação do processo em um texto. A seguir, é apresentada a aplicação no código:

```
import spacy
from spacy.lang.pt.examples import sentences
nlp = spacy.load("pt_core_news_md")

def lemmatizer(text):
    doc = nlp(text)
    lemmatized = " ".join([token.lemma_ for token in doc])
    return lemmatized
```

4.2.1. Term Frequency – Inverse document frequency

Conforme os estudos apresentados na seção “Base teórica”, o TF-IDF é uma técnica de análise estatística de corpus de texto, que pondera a frequência de ocorrência de uma certa palavra num documento e suas aparições em um conjunto. Os termos com maior frequência adquirem peso maior, mas esse fator é balanceado pelo inverso do número de ocorrências na seleção. Dessa forma, palavras que aparecem várias vezes em

muitos textos tem um peso menor, uma vez que provavelmente tratam-se de *stop words*. O valor resultante pode ser utilizado para estabelecer filtros do tipo passa-alta e passa-baixa, diminuindo a quantidade de *features* estudadas (37).

A vetorização consiste na transformação do texto em um vetor analisável. Tokenização é o processo de quebra do texto em elementos atômicos, que serão parte do vetor. No caso, faremos uma quebra simples por palavra. Esse tipo de vetorização é chamada unigrama. Alguns modelos utilizam n-gramas, conjuntos de n palavras que determinam uma *feature* (38).

A biblioteca sklearn apresenta uma função de vetorização TF-IDF pronta, que recebe um vetor de *strings* (textos) para determinação dos pesos por palavra, e retorna um objeto capaz de realizar a análise de um conjunto isolado (39). É possível determinar alguns parâmetros como frequência máxima e mínima desejada. No caso, realizaremos um filtro para retirar palavras com valor TF-IDF maior que 0.7.

O código a seguir apresenta a vetorização aplicada ao corpus estudado nesse projeto. Como o objetivo, no momento, é estudar os possíveis classificadores, o processo será aplicado a 80% das notícias, deixando o restante para teste do modelo.

```
      from sklearn.model_selection import train_test_split
      from sklearn.feature_extraction.text import TfidfVectorizer

      train_labels = df_mixed_simplified['Classification']

      x_train, x_test, y_train, y_test =
train_test_split(df_mixed_simplified['Text'], train_labels,
test_size=0.2, random_state=0)

      tfidf = TfidfVectorizer(max_df = 0.7) #max_df = 0.7

      tfidf_train = tfidf.fit_transform(x_train)
      tfidf_test = tfidf.transform(x_test)
```

A aplicação do processo na base gera um vetor de 64099 termos.

5. VISUALIZAÇÃO DOS DADOS

5.1. *Cloud of Words*

Existem algumas técnicas utilizadas para visualização dos dados no campo pro processamento de linguagem natural. Comumente esses estão associados

com a apresentação da frequência de aparição de termos por categoria a ser classificada. A seguir são apresentados alguns dos resultados obtidos com a análise da base (40).

A visualização *Cloud of Words* permite enxergar de forma intuitiva os termos mais presentes em cada uma das classificações. Isso é importante por se tratar de um dos fatores ponderados na aplicação da estatística TF-IDF. O tamanho da palavra indica quantas vezes ela aparece no conjunto de textos. Além disso, palavras normalmente próximas são representadas mantendo essa informação (41):

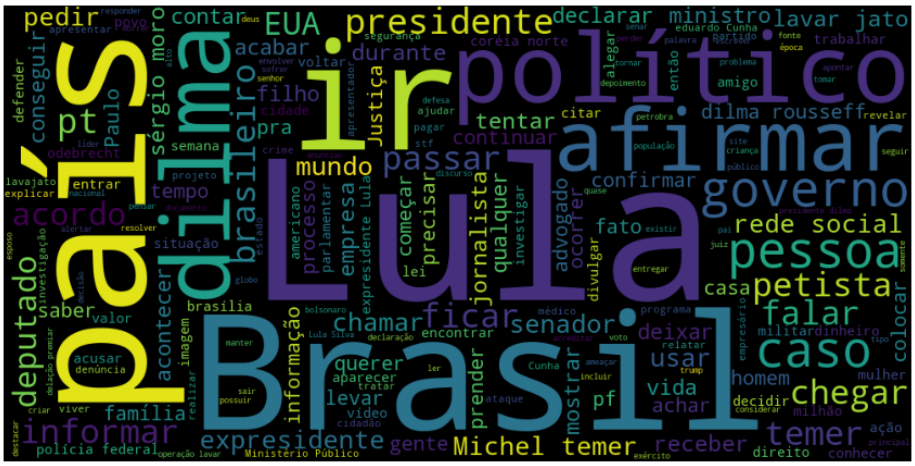

Figura 4: Cloud of words dos textos classificados como “falsos”

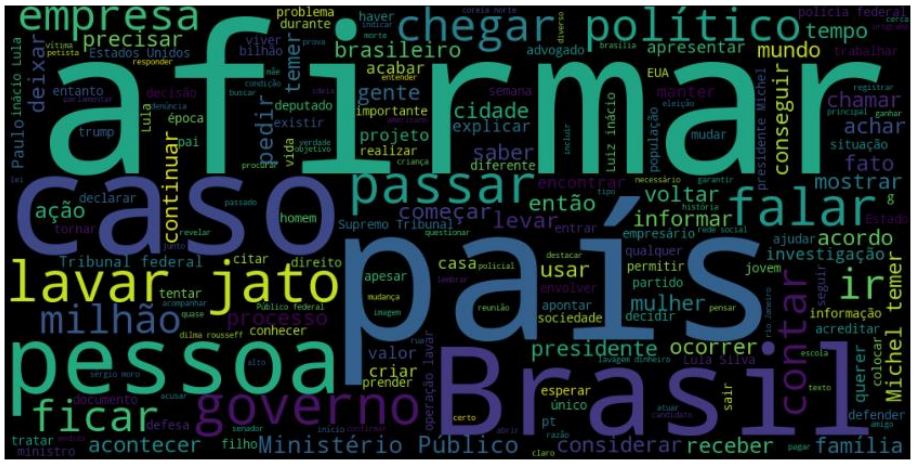

Figura 5: Cloud of words dos textos classificados como “verdadeiros”

Nota-se que alguns termos são muito comuns às duas classificações: “Brasil”, “país” e “político” são alguns exemplos. Esses termos, tendem a ter um *score* TF-IDF menor por estarem presentes em várias notícias.

Na base de notícias “falsas”, algumas palavras de interesse são “Lula”, “Político” e “Petista”. Além de denotar o forte caráter político desses textos, também indica a dominância da geração desse tipo de conteúdo pelos grupos ditos de direita, conforme já dito na primeira etapa do projeto.

Havendo a distinção entre ambas as bases, como apresentado nas imagens, o algoritmo de aprendizado tem chance maior de obter sucesso.

5.2. *Principal-component analysis*

A análise de componente principal é uma importante ferramenta para analisar conjuntos de dados com grandes dimensões, diminuindo a complexidade para bidimensional enquanto mantém os padrões e características do conjunto original (42). O processo funciona através da rotação dos eixos dos dados nas direções de maior variância (os componentes principais), seguido por um “achatamento” das variáveis, que levem à redução da dimensão (43) .

Os seguintes resultados foram obtidos com a aplicação do método:

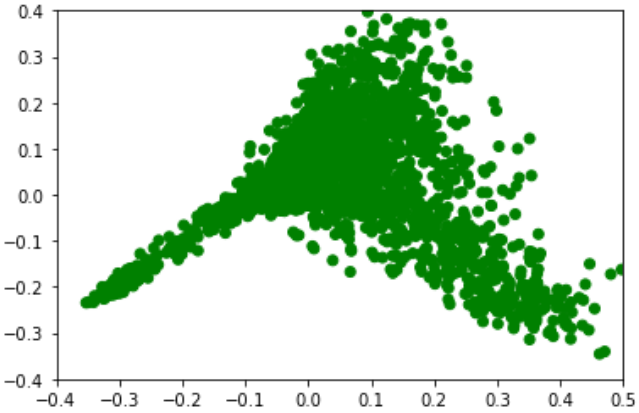

Figura 6: PCA para base verdadeira

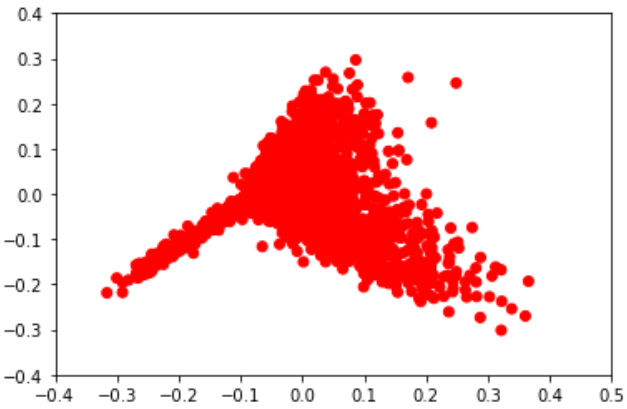

Figura 7: PCA para base falsa

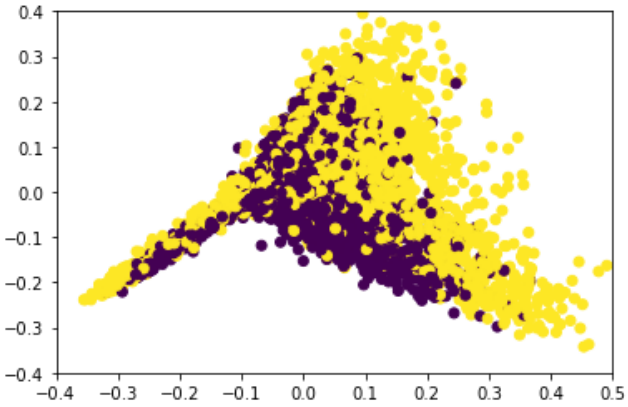

Figura 8: PCA para conjunto total

Os resultados indicam que o formato do espalhamento dos dados é muito semelhante entre ambas as bases. Entretanto, nota-se que a variância para os textos verdadeiros é maior, pois a amplitude da distribuição é maior. Isso indica uma variação maior de termos e frequências, que pode ser captada pelo algoritmo de aprendizado.

5.3. *Scatter text*

A biblioteca *Scatter Text* (44) utiliza a análise NLP providenciada pelo SpaCy para gerar uma página web interativa onde são apresentados os 2000 termos de maior relevância na análise TF-IDF de um corpus. Isso permite visualizar informações importantes como a distribuição dos termos entre cada classificação e o *score* associado a cada um deles. O seguinte gráfico foi gerado com utilização da ferramenta:

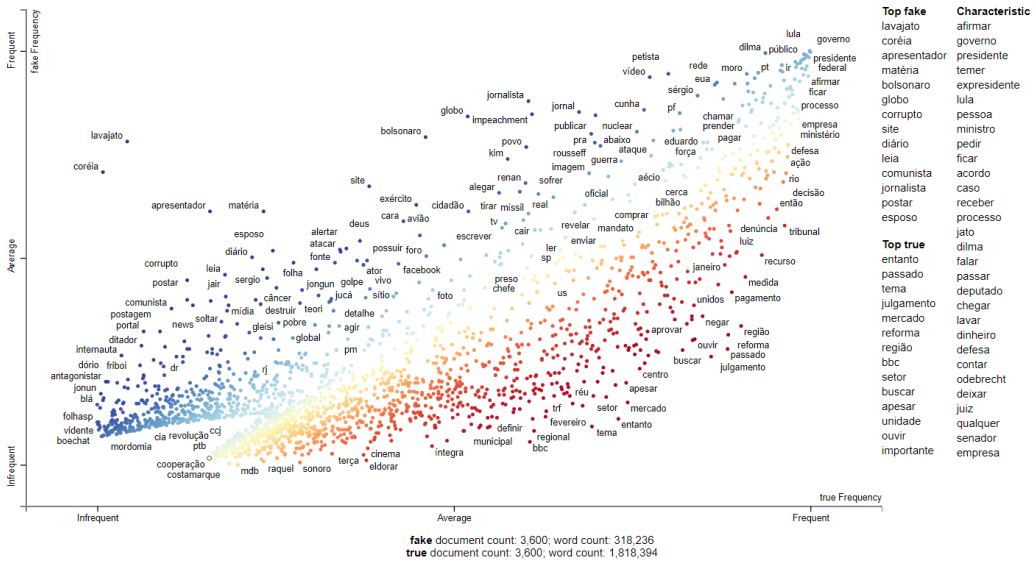

Figura 9: Scatter text da base de dados

As palavras estão localizadas de acordo com a sua frequência na base verdadeira (eixo horizontal) e falsa (eixo vertical). As palavras em amarelo têm frequências parecidas em ambas as classificações, enquanto as azuis são mais proeminentes em textos falsos e as vermelhas em verdadeiros. Nota-se que há espalhamento das distribuições, indicando novamente que há uma diferença notável entre as palavras utilizadas em cada uma das categorias. Além disso, há uma grande quantidade de palavras em azul acumuladas próximas da origem do texto, indicando uma boa diversidade de palavras (ainda que estas sejam utilizadas com menor frequência) dentre as notícias falsas, enquanto a distribuição das palavras da categoria de textos verdadeiros é menos densa e indica uma escolha de estilo de escrita mais “constante”.

6. MODELOS

Conforme apresentado anteriormente, três algoritmos já utilizados para o mesmo estudo desse projeto serão implementados, escolhendo-se aquele que obtiver melhor desempenho. À seguir, será descrito o desenvolvimento de cada um deles.

6.1. Classificador AdaBoost

O AdaBoost é um classificador *ensemble* baseado na tecnica de *boosting*. Este atua como um meta-estimador, que inicializa ajustando um classificador no conjunto de

dados original e, em seguida, ajusta cópias adicionais do classificador no mesmo conjunto de dados. Nestas, os pesos das instâncias classificadas incorretamente são ajustados de forma que os classificadores subsequentes se concentrem mais em casos difíceis (45). Os hiperparâmetros possíveis são:

- base_estimator: Escolha do estimador base sobre o qual é gerado o modelo. Por padrão, é utilizado um classificador do tipo árvore de decisão inicializado com profundidade 1.
- n_estimators: Valor máximo de estimadores até finalização do *boosting*. É um valor entre 1 e infinito, e caso ocorra um ajuste perfeito, o algoritmo é automaticamente finalizado. Por padrão, assume valor 50.
- learning_rate: Peso aplicado à cada estimador por iteração. Uma taxa maior aumenta a contribuição dos estimadores individuais. Por padrão, assume valor 1.0.
- algorithm: Algoritmo utilizado pelo modelo, conforme a biblioteca sklearn. Por padrão é utilizado o algoritmo “SAMME.R”, que costuma ter bons resultados em poucas iterações.

Serão testados modelos com os hiperparâmetros da tabela X.

Caso	Algoritmo	Estimadores	Aprendizado
1	DecisionTreeClassifier	50	1
2	DecisionTreeClassifier	100	0.8
3	LogisticRegression	50	1
4	LogisticRegression	100	0.8

Tabela 5: Definição dos hiperparâmetros do classificador AdaBoost

O seguinte código descreve a implementação do modelo:

```
abc = AdaBoostClassifier(n_estimators=estimators,
                         learning_rate=rate,
                         base_estimator=classifier
                        )
scores = cross_val_score(abc, tfidf_train, y_train,
scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)

abc.fit(tfidf_train, y_train)
y_pred = abc.predict(tfidf_test)
```

6.2. Classificador *One-Class SVM*

O classificador *One-Class SVM* atua como um detector de outliers, muito útil para classificação binária em bases extremamente desbalanceadas. Por sorte não é o caso deste projeto, mas considerando-se que fazer a curadoria de notícias verdadeiras é uma tarefa muito mais fácil do que o mesmo para notícias falsas, uma avaliação do classificador apresenta-se válida. Os seguintes parâmetro tem maior importância:

- kernel: Especificação do *kernel* (algoritmo) a ser utilizado pelo classificador. Pode assumir valores como ‘linear’, ‘poly’, ‘rbf’ e ‘sigmoid’. O valor padrão é ‘rbf’.
- degree: Grau da função polinomial a ser utilizada para o *fitting* em caso de escolha do kernel ‘poly’.
- gamma: Coeficiente paramétrico do *kernel*. Geralmente assume o valor 1/(numero de features*variância da base).
- coef0: Termo independente do *kernel*. Por padrão é 0.
- tol: Tolerância de parada. Por padrão é 0.001.

Importante ressaltar que o classificador passivo agressivo se diferencia dos outros por trabalhar com apenas uma classe (no caso, a de notícias verdadeiras). Dessa forma, a tokenização TF-IDF e o treino foram gerados a partir de 90% da base com classificação verdadeira (cerca de 3300 textos). O teste foi feito contra os 10% restantes e toda a base falsa.

Os seguintes parâmetros serão utilizados:

Caso	Kernel	Degree	Gamma	Coef	Tol
1	rbf	X	scale	0	0.001
2	poly	6	scale	0	0.001
3	poly	3	auto	0	0.001
4	rbf	X	auto	0	0.001
5	poly	6	scale	0	0.00001
6	sigmoid	X	auto	0	0.001

Tabela 6: Definição dos hiperparâmetros do classificador One-Class SVM

6.3. Classificador Passivo agressivo

Os algoritmos passivo-agressivos são uma família de algoritmos para aprendizado em larga escala. Eles são semelhantes ao *perceptron*, pois não exigem uma taxa de aprendizado. No entanto, ao contrário do supracitado, eles incluem um parâmetro de regularização c. Conforme citado na etapa de pesquisa, são muito uteis em NLP por serem rápidos e trabalharem bem com a grande quantidade de *features* utilizadas.

Os parâmetros principais são:

- C: Tamanho máximo do *step*. Serve como regularização. Por padrão é 1.
- max_iter: Número máximo de iterações até utilizar um *fit* parcial. Por padrão é 1000.
- tol: Tolerância. Atua como critério de parada. Por padrão é 0.001.

Os parâmetros da tabela X serão utilizados:

Caso	C	Max_iter	Tol
1	1	1000	0.001
2	0.7	2000	0.001
3	0.5	4000	0.001
4	0.5	4000	0.0001

Tabela 7: Definição dos hiperparâmetros do classificador Passivo Agressivo

7. RESULTADOS E ANALISE

Nessa etapa, são demonstradas as métricas resultantes da aplicação dos classificadores previamente apresentados. Para todos os casos, o seguinte procedimento será aplicado:

- Divisão da base total em duas partes, uma de teste e uma de treino. As proporções seguirão, respectivamente, os valores de 80% e 20%. Os dados de cada conjuntos serão escolhidos aleatoriamente, e todos os modelos e parâmetros receberão os mesmos dados. (46)
- O tokenizador TF-IDF é aplicado na base de treino. Dessa forma, as palavras presentes nesse conjunto são utilizadas para montar o dicionário de valores associados a cada termo.
- A base de teste é tokenizada utilizando o modelo TF-IDF resultante do passo anterior. Palavras existentes no dicionário tem seu score associado. Palavras inexistentes são descartadas.
- Os algoritmos são aplicados no conjunto tokenizado de treino. É gerado um modelo.
- O modelo resultante é utilizado para classificar a base de teste.
- As métricas são obtidas à partir das classificações.

7.1. Classificador AdaBoost

O primeiro modelo a ser testado foi o classificador AdaBoost. Foram aplicados os parâmetros apresentados na tebela 6. Os resultados são apresentados na tabela 9.

	Caso 1	Caso 2	Caso 3	Caso 4
Accuracy	94.31%	94.79%	93.82%	92.29%
Precision	95.24%	95.42%	94.44%	88.72%
Recall	93.42%	94.24%	93.28%	97.12%
F-Score	94.32%	94.82%	93.86%	92.23%
GM-Score	94.31%	94.8%	93.82%	92.1%
T. treino	32.16s	54.97s	14.93s	28.12s
T. teste	7.97s	15.74s	14.20s	23.97s

Tabela 8: Resultados para classificação com AdaBoost

7.2. Classificador *One-class SVM*

Testou-se o modelo do classificador *One Class SVM*. Foram aplicados os parâmetros apresentados na tebela 7. Os resultados são apresentados nas tabelas 10 e 11.

	Caso 1	Caso 2	Caso 3
Accuracy	73.83%	73.57%	51.08%
Precision	97.88%	99.2%	98.5%
Recall	48,72%	47.14%	2.17%
F-Score	65.06%	64.07%	4.24%
GM-Score	69.43%	68.66%	14.72%
T. treino	22.13s	41.89s	19.54s
T. teste	27.60s	53.90s	25.84s

Tabela 9: Resultados para classificação com One-Class SVM (Casos 1 a 3)

	Caso 4	Caso 5	Caso 6
Accuracy	73.83%	72.58%	51.97%
Precision	97.88%	96.94%	99.5%
Recall	48,72%	46.64%	3.94%
F-Score	65.06%	62.98%	7.59%
GM-Score	69.43%	67.79%	19.86%
T. treino	22.13s	22.62s	20.04s
T. teste	27.60s	29.16s	19.86s

Tabela 10: Resultados para classificação com One-Class SVM (Casos 4 a 6)

7.3. Classificador Passivo Agressivo

O ultimo modelo a ser testado foi o classificador Passivo Agressivo. Foram aplicados os parâmetros apresentados na tebela 8. Os resultados estão contidos na tabela 12.

	Caso 1	Caso 2	Caso 3	Caso 4
Accuracy	95.35%	95.28%	95.1%	95.07%
Precision	94.37%	94.6%	94.47%	94.34%

Recall	96.57%	96.16%	96.02%	96.02%
F-Score	96.57%	95.37%	95.24%	95.17%
GM-Score	95.32%	95.26%	95.12%	95.05%
T. treino	0.07s	0.068s	0.061s	0.123s
T. teste	0.002s	0.002s	0.002s	0.001s

Tabela 11: Resultados para classificação com classificador passive agressivo

7.4. Interpretação dos resultados e escolha do modelo

O primeiro ponto importante a ser determinado para escolha do modelo que será utilizado é determinar quais são as métricas adequadas. Conforme descrito por (33), dois parâmetros que costumam ter bons resultados para analisar classificação de textos costumam ser o GM-Score e o F-Score. Estes são compostos pelas outras métricas obtidas.

Além destes, o *recall* é uma medida importante por relacionar-se com o tema do projeto. A métrica indica menor número de falso negativos, fator que deve ser considerado uma vez que uma notícia verdadeira ser classificada como falsa é menos prejudicial do que uma falsa ser classificada verdadeira. (47)

Por fim, o tempo de *fitting* (tempo para classificar um texto utilizando o modelo) deve ser baixo. Isso evita gargalos no servidor , uma vez que a predição ocorre em tempo real.

Dessa forma, é escolhido um caso de destaque para cada classificador, demonstrados na tabela 13.

	AdaBoost	OC-SVM	Pass. Agressivo
	Caso 2	Caso 1	Caso 1
Accuracy	94.79%	73.83%	95.35%
Precision	95.42%	97.88%	94.37%
Recall	94.24%	48,72%	96.57%
F-Score	94.82%	65.06%	96.57%
GM-Score	94.8%	69.43%	95.32%
T. treino	54.97s	22.13s	0.07s
T. teste	15.74s	27.60s	0.002s

Tabela 12: Compilação dos melhores resultados

Tanto o classificador One-*Class Support Vector Machine* quanto o passivo agressivo obtiveram melhor desempenho no caso padrão, utilizando os parâmetros convencionais da biblioteca SKLearn. O classificador AdaBoost, por outro lado, obteve melhor performance quando ajustado.

Salta aos olhos o desempenho ruim do OC-SVM comparado com os outros. Isso já era esperado: a base de treino foi muito reduzida, por contemplar apenas notícias verdadeiras (cerca de 45% menor). Além disso, o *recall* foi muito afetado pela característica principal do algoritmo de não precisar da segunda classe. Dessa forma, ele tornou-se muito bom em detectar e acertar quando a notícia é verdadeira (como indicado pela excelente precisão), mas ruim em classificar corretamente as notícias falsas (como indicado pelo *recall).*

Dessa forma, dentre os três, o **classificador passivo agressivo**, com os parâmetros do caso 1, é o que obteve melhores resultados, e foi utilizado na ferramenta de detecção desenvolvida.

8. COMPARAÇÃO COM OUTROS ESTUDOS

A seguir são apresentados resultados de outros estudos a fim de comparar a performance do modelo obtido com outros trabalhos na mesma área.

8.1. Kudari

Kudari et al. (19) realiza a classificação de notícias falsas em uma base em inglês utilizando dois métodos estatísticos diferentes: vetorização TF-IDF e *Bag of Words.* Também utiliza dois classificadores: Passivo agressivo e Naive-Bayes. Os resultados são apresentados na tabela 14.

Método	Bag of Words	Bag of Words	TF-IDF	TF-IDF
Classificador	Naive-Bayes	Pass. Agr.	Naive-Bayes	Pass. Agr.
Accuracy	87.01%	89.73%	80.06%	92.30%
Precision	92.96%	89.70%	96.26%	91.39%
Recall	80.00%	89.70%	62.42%	93.33%
F-Score	85.99%	89.70%	75.74%	92.35%
GM-Score	86.71%	89.73%	78.05%	92.29%

Tabela 13: Resultados obtidos por Kudari

8.2. Faustini e Covões

Faustini e Covões (22) estudaram a aplicação do algoritmo *One-Class SVM* e outros algoritmos que exigem apenas uma classe na base *FakeBr*, a mesma utilizada nesse artigo. A única métrica apresentada no artigo é o F-Score, e seus valores para diferentes classificadores estão na tabela 15.

Método	DCDistance	OC-SVM	EcoOCC	NBPC
F-Score	67%	64%	67%	67%

Tabela 14: Resultados obtidos por Faustani e Covões

8.3. Sekiguchi, Gonçalves e Tsuji

Nos trabalhos que precederam a elaboração desse projeto, Sekiguchi, Gonçalves e Tsuji (5) realizaram a análise de notícias na mesma base, utilizando dois métodos diferentes: Bag of Words e análise dos parâmetros sintáticos (características de cada texto, como número de substantivos, verbos e adjetivos). São utilizados diferentes classificadores, dentre os quais se destacaram o *perceptron* e *random forest*. No mesmo texto, entretanto, os autores citam um possível viés da classificação que atribui o *label* de verdadeiro à textos com mais palavras, uma possível consequência do uso de parâmetros não relativos. Seus resultados são demonstrados na tabela 16.

Método	Bag of Words	Bag of Words	Sintaxe	Sintaxe
Classificador	RF	Perceptron	RF	Perceptron
Accuracy	95.44%	95.14%	95.96%	96.34%
Precision	95.17%	94.78%	96.85%	96.81%

Tabela 15: Resultados obtidos por Sekiguchi, Gonçalves e Tsuji

8.4. Notas sobre os resultados

Comparando-se os desempenhos apresentados por estudos no mesmo tema com os obtidos no projeto, é possível confirmar os bons resultados do classificador proposto. As métricas, foram no geral, equivalentes ou melhores do que as obtidas com métodos diferentes ou semelhantes de classificação. Esse efeito é consequência de diferentes

fatores: diferentes bases, métodos de pré-processamento, escolha de hiper parâmetros dos modelos e dos métodos estatísticos.

9. Ferramenta web

Para disponibilizar a ferramenta ao usuário comum, optou-se por criar um sistema web composto por uma página e um *backend* simples. O fluxo de informação ocorre entre esses dois elementos: o usuário insere o texto na página, que é enviado para o *backend* através de um *request POST* . Após o pré-processamento, lematização e tokenização TF-IDF, o classificador passivo agressivo previamente treinado é aplicado. O resultado é enviado de volta à pagina, e exibido ao usuário.

Para armazenar a classe Python do modelo treinado foi utilizado o modulo Píckle. Isso permitiu separar a interface de treinamento do sistema web, uma vez que unir ambos seria uma tarefa trabalhosa.

A estrutura de computação da AWS (*Elastic Cloud Computing*, ou EC2) foi escolhida por ser uma opção gratuita e de rápida disponibilidade. A instância de computação do tipo *micro* conta com um sistema operacional Ubuntu e kernel Linux.

O framework web escolhido foi o Flask, que integrou-se muito bem com o projeto, uma vez que todas as etapas foram realizadas utilizando-se Python, mesma linguagem de desenvolvimento para esta estrutura. O Flask é classificado como um microframework, por ter um núcleo simples e poder ser utilizado para criar pequenos serviços web com poucas linhas, diferente de um framework como Django.

O *frontend* apresentado ao usuário foi desenvolvido à partir de um *template* gratuito disponível na internet. A comunicação e gerenciamento do envio de informações é efetuado com JavaScript e a biblioteca Ajax. O visual da página. A figura 10 demonstra o resultado final.

O servidor HTTP escolhido foi o NGinx, por ser open source, simples e ter uma comunidade empenhada que ajuda no suporte e dúvidas.

Para gerenciar o *hosting* do serviço flask, foi utilizado o WSGI Gunicorn, um gateway especificamente para python.

A figura 11 descreve o fluxo de informação entre os módulos do microserviço.

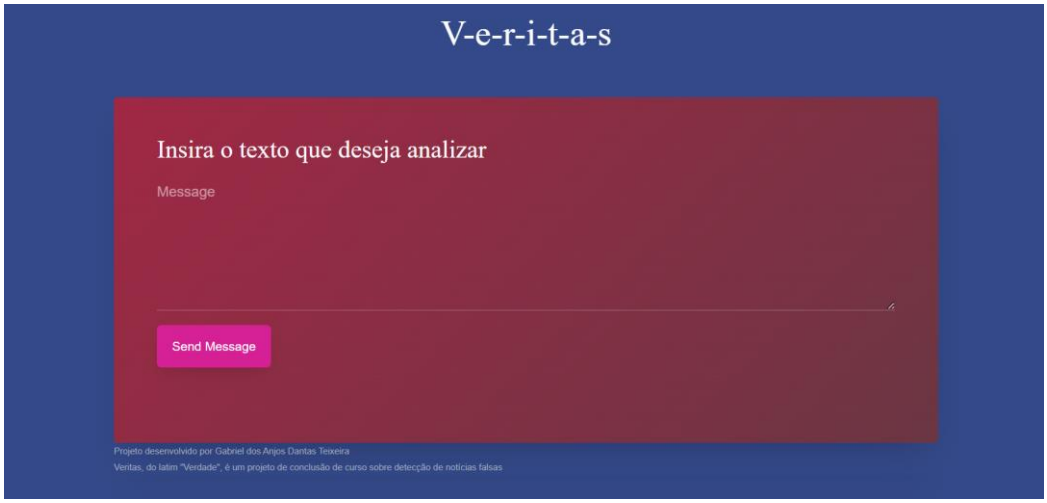

Figura 10: Página principal da ferramenta

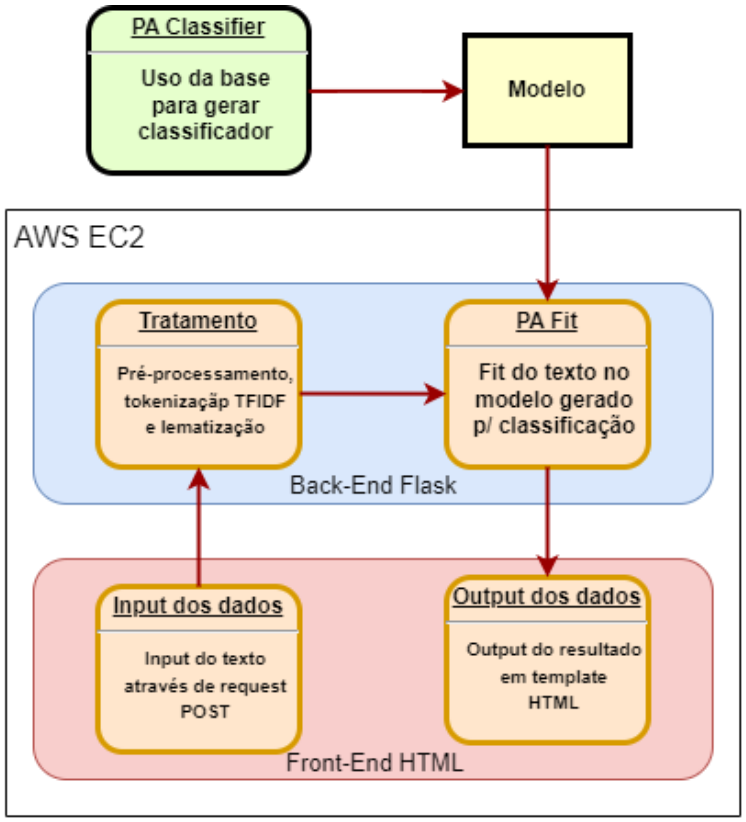

Figura 11: Fluxo de informação

10. CONCLUSÃO

Foi possível integrar o sistema web com o algoritmo classificador e, dessa forma, montar a ferramenta proposta desde o início do projeto: um site que pudesse ser utilizado para classificar notícias entre verdadeiras e falsas.

Como prova do bom funcionamento do algoritmo, foram realizados alguns testes utilizando textos da própria base de treino. Os resultados apontados no site foram corretos, indicando que todos os módulos funcionaram corretamente.

Por fim, foram realizados alguns testes de uso, com páginas que não estivessem na base de treino. Nesse teste, foram utilizadas notícias de websites como G1, Estadão e Folha, sites da base falsa do *corpus* e alguns textos de correntes de Whatsapp.

O funcionamento do algoritmo mostrou-se adequado. Foi possível notar algum viés para apontar o texto como mentiroso, mesmo quando esse resultado fosse um falso positivo. Não é possível determinar as métricas com uma base pequena como esta, mas a priori repara-se que a acurácia de 95% não foi mantida. Uma primeira hipótese é que seja decorrente da diferença de datas entre as amostras do *corpus* (2017-2019) e os textos coletados para essa última validação. Ainda assim, a ferramenta apresentou comportamento consistente e consideravelmente confiável, e os objetivos propostos foram alcançados.

11. CONSIDERAÇÕES FINAIS

11.1. Conclusões do Projeto de Formatura

O resultado desse projeto foi a criação de uma ferramenta automatizada capaz de classificar um texto entre verdadeiro e falso. O classificador utiliza uma base pré treinada com os vocabulários e frequências de cada palavra, associando esses valores à classificação do texto, para fazer essa distinção. Não é feita nenhuma forma de *fact* checking ou pesquisa online. Apesar da base teórica e fundamento prático, ressalta-se que é um projeto acadêmico, que somente poderá ser disponibilizado ao público após testes mais amplos e rigorosos.

11.2. Trabalhos Futuros

Alguns temas e funções que foram pensados e testados, porém descartados, durante o desenvolvimento desse projeto são descritos à seguir

- Utilização do algoritmo de estado-da-arte Google BERT como classificador – a implementação em português deste ainda não é bem estabelecida
- Ampliação do *corpus* utilizando-se um *crawler*, a fim de atualizar as notícias da base

- Algoritmo *real-time* de aperfeiçoamento do modelo, utilizando os inputs do usuário para obter novos dados
- Utilização de outras informações além do corpo do texto: URL, metadados da página, buscas na internet

BIBLIOGRAFIA

1. **Galhardi, Cláudia Pereira.** Fato ou Fake? Uma análise da desinformação frente à pandemia da Covid-19 no Brasil. *Ciência & Saúde Coletiva 25.* 2020.

2. **Bucci, Eugenio.** Seriam as fake news mais eficazes para campanhas de direita?–uma hipótese a partir das eleições de 2018 no Brasil. *Novos Olhares 8.2.* 2019.

3. **Ahmed, Alim Al Ayub.** Detecting fake news using machine learning: A systematic literature review. *arXiv preprint arXiv:2102.0445.*

4. **Chowdhary.** Natural language processing. *Fundamentals of artificial intelligence.* 2020.

5. **SEKIGUCHI, Daniela Yumi.** *Sistema de detecção e classificação de notícias falsas.* São Paulo, 2021 : Trabalho de Conclusão de Curso (Graduação) – Escola Politécnica, Universidade de São Paulo, 2021.

6. **CARVALHO, M. F. C.** A. FAKE NEWS E DESINFORMAÇÃO NO MEIO DIGITAL: Análise da Produção Científica Sobre o Tema na Área de Ciência da Informação. Múltiplos Olhares em Ciência da Informação. *https://periodicos.ufmg.br/index.php/moci/article/view/16901*.

7. **PAPANASTASIOU, Y.** FAKE NEWS PROPAGATION AND DETECTION: A Sequential Mode. 2018.

8. **WARDLE, C.** FAKE NEWS. IT'S COMPLICATED. *First Draft.* [Online] 2017.

9. **Lim, C.** CHECKING HOW FACT-CHECKERS CHECK. *Research & Politics.* 2018.

10. **Lopes, Larissa.** USP e UFSCar criam ferramenta que detecta fake news. *Galileu.* [Online] https://revistagalileu.globo.com/Tecnologia/noticia/2018/10/usp-e-ufscar-criam-ferramenta-que-detecta-fake-news.html.

11. **Jones, Karen Sparck.** What is the role of NLP in text retrieval? *Natural language information retrieval. Springer, Dordrecht.* 1999.

12. **Menzli, A.** Tokenization in NLP: Types, Challenges, Examples, Tools. *Neptune AI.* [Online] https://neptune.ai/blog/tokenization-in-nlp.

13. **Balakrishnan, Vimala.** Stemming and lemmatization: A comparison of retrieval performances. 2014. 2014.

14. **Dias, Maria Abadia Lacerda.** *Extração automática de palavras-chave na língua portuguesa aplicada a dissertações e teses da área das engenharias.* s.l. : UNICAMP–Universidade Estadual de Campinas–Campinas.

15. **Matsubara, Edson Takashi.** Pretext: Uma ferramenta para pré-processamento de textos utilizando a abordagem bag-of-words. *Techinical Report 209.4.* 2003.

16. *A topical word embeddings for text classification.* **Lima, João Marcos Carvalho.** s.l. : Anais do XV Encontro Nacional de Inteligência Artificial e Computacional.

17. **Probierz, Barbara.** Rapid detection of fake news based on machine learning methods. *Procedia Computer Science.* 2021.

18. **Ramos, Juan.** Using tf-idf to determine word relevance in document queries. *Proceedings of the first instructional conference on machine learning.* 2003.

19. **Kudari, Jayashree M.** Fake News Detection using Passive Aggressive and TF-IDF Vectorizer. *International Research Journal of Engineering and Technology (IRJET).* 2020.

20. **El Naqa, Issam.** What is machine learning? *machine learning in radiation oncology.* 2015.

21. **Amjad, Maaz.** Bend the truth": Benchmark dataset for fake news detection in Urdu language and its evaluation. *Journal of Intelligent & Fuzzy Systems.* 2020.

22. **Faustini, Pedro, and Thiago Covões.** Fake news detection using one-class classification. *8th Brazilian Conference on Intelligent Systems.* 2019 .

23. **Shmilovici, A.** Support Vector Machines. *Data Mining and Knowledge Discovery Handbook.*

24. **Amer, Mennatallah.** Enhancing one-class support vector machines for unsupervised anomaly detection. *In Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description.* 2013.

25. **Pereira, Lucas.** Estudo de Técnicas de Ensemble para Classificação de Dados. 2021.

26. **Breiman, L.** Bagging predictors. Machine Learning. 1996.

27. **Freund, E. S.** A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences.* 1997.

28. **Mandical, Rahul R.** Identification of fake news using machine learning. *IEEE International Conference on Electronics, Computing and Communication Technologies.*

29. **Gupta, Saloni.** Fake news detection using passive-aggressive classifier. *Inventive Communication and Computational Technologies.*

30. **Brownlee, J.** A Gentle Introduction to k-fold Cross-Validation. Machine Learning Mastery. *machinelearningmastery.* [Online] https://machinelearningmastery.com/k-fold-cross-validation/.

31. **Arun, K.** Understanding Curse of Dimensionality. My Great Learning. *mygreatlearning.* [Online] 2020. https://www.mygreatlearning.com/blog/understanding-curse-of-dimensionality/.

32. **Susmaga, R.** Confusion Matrix Visualization. *Intelligent Information Processing and Web Mining. Advances in Soft Computing.* 2004.

33. **Hossin, Mohammad.** A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process.*

34. **L. S. Santos, Roney.** The Fake.Br corpus - a corpus of fake news for Brazilian Portuguese.

35. **Lambert, N.J.** Text Mining Tutorial. *Computational Social Sciences.*

36. **Lopes, António.** Github - Portuguese stop words. *Github* . [Online]

37. **Wendland, A, M.** Introduction to Text Classification: Impact of Stemming and Comparing TF-IDF and Count Vectorization as Feature Extraction Technique. *Communications in Computer and Information Science.*

38. **Singh, Anita Kumari.** Vectorization of text documents for identifying unifiable news articles. *International Journal of Advanced Computer Science and Applications.*

39. **scikit-learn developers.** sklearn.feature_extraction.text.TfidfVectorizer. [Online] 2022.

40. **Heimerl, Florian.** Word cloud explorer: Text analytics based on word clouds. *47th Hawaii international conference on system sciences.* 2014.

41. **Soni, Rounak.** News analysis using word cloud. *Advances in Signal Processing and Communication.*

42. **Lever, J, M Krzywinski.** Principal component analysis. *Nat Methods.* 2017.

43. **Jolliffe Ian, T.** Principal component analysis: a review and recent developments.

44. **Kessler, Jason.** GitHub - Beautiful visualizations of how language differs among document types. [Online]

45. **developer, scikit-learn.** sklearn.ensemble.AdaBoostClassifier. *scikit-learn.org.* [Online] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html.

46. **Monti, Federico.** Fake news detection on social media using geometric deep learning.

47. **Mankad, Sahil.** A Tour of Evaluation Metrics for Machine Learning. *analyticsvidhya.* [Online] https://www.analyticsvidhya.com/blog/2020/11/a-tour-of-evaluation-metrics-for-machine-learning/#:~:text=Precision%20is%20used%20as%20a,is%20to%20minimize%20false%20negatives..