

**BABEȘ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER
SCIENCE
SPECIALIZATION COMPUTER SCIENCE ENGLISH**

DIPLOMA THESIS

Fake News Detection using Machine Learning Algorithms

Supervisor

**PhD Student Coste Claudia-Ioana,
Prof. PhD Andreica Anca-Mirela**

Author
Suceava-Codreanu Octavia-Maria

2023

**UNIVERSITATEA BABEȘ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ ENGLEZĂ**

LUCRARE DE LICENȚĂ

**Identificarea știrilor false prin
utilizarea algoritmilor de învățare
automată**

Conducător științific

**Drd. Coste Claudia-Ioana,
Prof. univ. Dr. Andreica Anca-Mirela**

*Absolvent
Suceava-Codreanu Octavia-Maria*

2023

ABSTRACT

Considering the development of technology, as well as the number of people that are using the internet nowadays, detection of fake news has become a very important task to combat misinformation. Although not harmful at first, misinformation can lead to significant negative aftermaths. Therefore, this paper aims to find the most effective way to detect false information, task that is heavily reliant on text processing methods and the selection of appropriate classification algorithms. The results obtained from the experiments demonstrate the effectiveness of the Logistic Regression, Passive Aggressive Classifier, and Decision Tree in fake news detection. Each algorithm exhibits different strengths and weaknesses in terms of accuracy. Furthermore, the study explores the importance of feature selection techniques on the performance of the classifiers. For this task, three different approaches were chosen, where each time different combinations of features are taken into account. The final application is a user-friendly web page where individuals can input the necessary content and receive an evaluation of whether the news is genuine or fake. Overall, the findings of this research provide valuable guidance when seeking to develop effective strategies for combating the spread of fake news.

This paper consists of eight well-structured chapters. It begins with an introduction to fake news and its historical impact, after which it explores existing research in the field and provides a detailed overview of Natural Language Processing (NLP) techniques, including text preprocessing and feature extraction. The paper also discusses different machine learning algorithms used for fake news detection. Furthermore, it presents the project's architecture, testing methods, and results analysis, comparing them with relevant findings from the literature. The paper concludes by discussing potential future directions in the field of fake news detection.

Contents

1	Introduction	1
2	Fake News	4
2.1	Defining Fake News	4
2.2	The Impact of Fake News: A Historical Perspective	4
3	Previous work	6
4	Natural Language Processing	10
4.1	History of NLP	10
4.2	Text preprocessing	11
4.3	Text to features	12
4.4	TF-IDF	12
5	Machine Learning	14
5.1	Types of Machine Learning algorithms	14
5.2	Logistic Regression	15
5.3	Decision Trees	16
5.4	Passive-Aggressive Classifier	16
6	Application development	18
6.1	Problem Definition	18
6.2	Analysis and application architecture	19
6.2.1	Functional and non-functional requirements	19
6.2.2	Application architecture	20
6.3	Technologies	22
6.4	Testing	24
6.5	User Manual	25
7	Classification methodology	28
7.1	Dataset	28
7.2	Text preprocessing and feature extraction	30

7.3 Models	31
7.4 Results and comparisons	33
8 Conclusions and Future Work	37
Bibliography	39

Chapter 1

Introduction

Motivation

The rise of social media has made it much easier for people to get news, as people often turn to social platforms, such as Facebook or Twitter, for finding news. According to a survey conducted in 2021, 48% of U.S. adults rely on social media for their news [Wal21]. Even individuals who do not actively follow news pages on social media are susceptible to exposure, given the nature of these platforms where anyone can share and post content without any restrictions.

The degree to which misinformation can affect one's life goes beyond trivial deceptions, such as taking vitamin C to cure coronavirus [Bro20], and it can significantly impact one's mental and physical health.

To illustrate this, we can take a look at how the COVID-19 pandemic led to a "fake news pandemic". The circumstances surrounding the pandemic, characterized by lack of control, anxiety and fear regarding the future, created the perfect environment for spreading misinformation. The rapid spread of information through social media and other digital platforms allowed false or misleading information to circulate quickly. Notably, negative news often captures people's attention more easily than positive news [Sor15], and considering the fact that many news surrounding the virus were negative, they naturally gained more traction online.

Before talking about the impact of fake news on society, it is important to understand what circumstances intensify the spread of such news. A study conducted in India during the initial months of the COVID-19 outbreak in 2020 showed a drastic increase in debunked information, from just two instances in January to sixty instances by April [ASK⁺20]. This highlights the accelerated spread of misinformation surrounding important events.

Another study shows how frontline healthcare workers were affected by "infodemia" [Wor] during the COVID-19 outbreak. The study found that, out of a sample of 126 people, almost half of them claimed to be affected by fake news regarding

the pandemic. Moreover, the ones who claimed to be affected by false information exhibited higher levels of stress, anxiety and insomnia, compared to their counterparts [SVC⁺20].

Fake news are harmful not only to people's mental health, but physical health as well. A worrisome incident in 2020 exemplifies this, as Donald Trump's claim that chloroquine could treat COVID-19 led to reports from Nigeria of people being hospitalized due to overdoses of the substance [BA20].

Furthermore, in the political context, fake news can be used as a tool to damage reputations and manipulate people's beliefs, potentially endangering a country's future or individuals' well-being. One example of these news being used as a tool is the 2016 U.S. presidential election. During that time, a large amount of fake news was circulating on social media, with one study revealing that fake news articles favoring Donald Trump were widely shared on Facebook compared to those favoring Hillary Clinton [AG17]. Similarly, the conflict between Russia and Ukraine created a suitable environment for dissemination of fake news, with the intention of pushing people into taking a side [2423].

Combating misinformation online is a very important task for building and keeping a well-informed society. Taking into account all the negative effects misinformation can have, it is essential for people to have all the necessary tools to evaluate the veracity of the news they encounter on a daily basis.

Objectives

The challenge of fake news can be approached from multiple points of view, those being, for example, psychology, communication or computer science. From the perspective of computer science, this paper aims to contribute to the development of tools aimed at countering the "fake news" phenomenon. The research specifically focuses on investigating and analyzing several text processing techniques and classification algorithms for efficient false news identification.

Original contributions

This paper's original contributions include exploring the impact of different feature selection techniques on the overall performance of the classifiers. By examining various feature combination, we can observe how much of a difference do they make on the overall performance of the models, namely Logistic Regression, Decision Trees, and Passive-Aggressive Classifier. Furthermore, we explore the strengths and weaknesses of the TF-IDF vectorizer and the performance of the models on new data. Additionally, we will go beyond theoretical analysis by developing a practical solution in the form of a user-friendly web application.

Paper structure

The paper is structured in such a way that we can get a comprehensive insight into the topic. Chapter 2 begins with a detailed introduction that defines fake news and highlights their impact through historical events. Furthermore, in Chapter 3, we thoroughly explore other papers in order to gain a comprehensive understanding of the existing research in the field of fake news detection. Chapter 4 delves into the field of Natural Language Processing (NLP), starting with a history of it and further explaining the steps of text preprocessing, feature extraction and a more detailed explanation of TF-IDF, which we are using in our project. Moving forward, Chapter 5 focuses on Machine Learning (ML), presenting the types of ML algorithms and explaining the algorithms that we are using in our project. Chapter 6 offers a comprehensive overview of our project: problem definition, application architecture, technologies employed, testing methods and the user manual that facilitates ease of use. In Chapter 7 we are talking about the dataset, the NLP steps undertaken, the machine learning models, and an analysis of our results, including comparisons with relevant findings from existing literature. Lastly, Chapter 8 concludes this paper and discusses potential future directions in the field of fake news detection.

Chapter 2

Fake News

We've seen that the problem of fake news has become an important cause of concern. This chapter explores the phenomenon of fake news, including its definition and the historical effects it has had on public opinion and decision-making.

2.1 Defining Fake News

"Fake news" refers to intentionally false information that is presented as if it were real news. The main motivation behind the creation of fake news is typically to influence public opinion, advance a political or commercial agenda, or harm someone's reputation. This phenomenon has become more prevalent in recent years as social media gained more popularity, since it was easier to spread information widely and quickly.

It is important to keep in mind that the key characteristic of fake news is that its only purpose is to deceive. Therefore, it is different from other types of misinformation, such as satire or parody, as these are not intended to be taken seriously and they are intentionally exaggerated and humorous. Another example would be opinion articles, which also should not be included in the same category as fake news, since they reflect the author's views and perspective [McG22].

Because fake news is specifically designed to mislead readers, it can have serious consequences for public discourse and decision-making. Thus, by contrast, other forms of misinformation may be unintentional and less harmful.

2.2 The Impact of Fake News: A Historical Perspective

The concept of "fake news" has been around for centuries, although it was not called by that name until recently. For example, in the late 19th century journalists would practice what we call "yellow journalism", term that refers to newspapers

containing no well-researched news, all while using a catchy and sensational headline that will push people into buying the newspaper.

Even then, it has been proven how much these news can influence public opinion, when yellow journalism reached its peak, right before the Spanish-American War in 1898. After a U.S. battleship sunk in Havana harbor, two rival newspaper publishers, Joseph Pulitzer and William Hearst, published incredulous news about the incident, blaming Spain for it. Although not directly responsible for starting the war, the sensationalist headlines, such as "Destruction of the War Ship Maine Was the Work of an Enemy", helped to fuel anti-Spanish sentiment among the American public and create outrage, which made it hard for the government to not call for military action [The].

Another example of fake news that led to disastrous events is the rumor that Iraq has weapons of mass destruction. This information was a primary reason for the United States-led invasion of the Republic of Iraq in 2003, with President George W. Bush declaring in a conference that they aimed to *"disarm Iraq of weapons of mass destruction, to end Saddam Hussein's support for terrorism, and to free the Iraqi people"* [Nat]. Later on, these claims turned out to be false, after a United Nations inspection concluded that there was no evidence of Iraq possessing such weapons right before the invasion [Uni].

Chapter 3

Previous work

Due to the ease of access provided by modern technology, there is an increase in the amount of misinformation, hence many researchers have focused their efforts on developing a mechanism to distinguish between fake and authentic news. In this chapter, we examine several studies and strategies used to address the issue of fake news identification.

The automated detection of fake news is the main topic of the study done by Fayeze *et al.* [FAA]. They suggest a technique that distinguishes between fake and real news stories on Twitter using a variety of factors, for example retweet count or follower count. Due to the dataset's large number of features, they use techniques such as SelectKBest in order to select only the most important features and reduce the computational cost. For the machine learning models, they compare the results provided by Random Forest with the results provided by other popular models, such as Neural Networks, Gaussian Naïve Bayes and Decision Trees. Out of all the models, Random Forest gets the best prediction, with 99% accuracy. This achievement demonstrates the effectiveness of Random Forest in distinguishing between fake and real news based on the selected features. While this is an impressive result, it is important to mention that the dataset used in the study is relatively small and unbalanced, consisting of 3000 fake accounts and 1481 real accounts. Therefore, it is unclear how the models would perform on new and balanced data. The generalizability and reliability of the suggested technique would need to be confirmed by more study using larger and more varied datasets.

The research article titled "*Fake News Detection Using Machine Learning Ensemble Methods*" by Ahmad *et al.* [AYYA20] addresses the problem of detecting fake news using machine learning ensemble methods. For this study, they used three different datasets, denoted DS1, DS2 and DS2, and one that consists of all the other datasets combined, denoted DS4. To extract features from the textual data, Ahmad *et al.* employ Linguistic Inquiry and Word Count (LIWC), a linguistic analysis software. LIWC provides a comprehensive analysis of the emotional, cognitive, and structural

components of written text, allowing for a deeper understanding of the linguistic patterns associated with fake news articles. They compared individual classifiers, such as Logistic Regression or K-nearest neighbours, to ensemble methods, such as Bagging Classifier or AdaBoost, in order to emphasize the superiority of ensemble learners. Ensemble methods combine the predictions of different individual models to make more reliable predictions. However, individual models led to promising results as well. Logistic regression, although a simpler model, achieved an average accuracy of over 90% on three datasets. The authors mention that one explanation for this is fine-tuning the logistic regression model with different hyperparameters through a grid search. Additionally, the similar writing styles of authors in some datasets potentially led to higher accuracy for the logistic regression model. However, on DS4, which included various writing styles, the accuracy of logistic regression dropped to 87%. By employing ensemble methods and considering a various range of classifiers, Ahmad *et al.* contribute to the development of more accurate models for detecting fake news, while also showcasing the importance of combining diverse approaches and considering multiple datasets to improve the performance of machine learning algorithms.

Furthermore, we can observe the importance of feature engineering and dimensionality reduction approaches in the article titled *Fake News Stance Detection Using Deep Learning Architecture (CNN-LSTM)* by Umer *et al.* [UIU⁺20]. The study proposes an approach based on combining Convolutional Neural Networks (CNN) and Long Short Term Memory (LSTM) in order to determine the stance of a news article towards its headline. The dataset consists of 75385 labeled instances and 2587 article bodies. The labels for the articles include "agree" (relation between headline and article body), "disagree" (no relation between headline and article body), "discuss" (some match between headline and article body), and "unrelated" (completely different topics). Preprocessing techniques such as lowercase conversion, stopwords removal, stemming, and tokenization are applied to the dataset using NLP techniques. The study's methodology uses hybrid deep learning models and feature reduction techniques. Four data models are developed: one with all features without preprocessing, one with non-reduced preprocessed features, and two models with dimensionality reduction techniques (PCA and Chi-square). They first test the model on the dataset without preprocessing, which gives an accuracy of 78%. After preprocessing, the accuracy increases to 93%, which is a significant amount. However, reducing the dimensions of the feature vectors increases the accuracy to 95%, respectively 97%. Reducing the feature vector will result in a vector that contains only the most important information, which leads to reducing the complexity of the model and decreasing the chances of overfitting. This proves to be an advantage from a computational complexity point of view as well, as the models will need less

resources and less time to train on the given data.

The article "*Detection of Fake News using Machine Learning Models*" by Velivela Durga Lakshmi and Ch Sita Kumari [LK22] explores the use of machine learning algorithms to detect fake news. Their study tests machine learning models, such as Logistic Regression, Support Vector Classification, Random Forest Classifier and Naïve Bayes, in combination with different feature extraction techniques, like TF-IDF Vectorizer and Count Vectorizer. The preprocessing step involves removing stopwords and applying stemming to clean the text data, after which the preprocessed data is inputted into the Term Frequency-Inverse Document Frequency (TF-IDF) Vectorizer and Count Vectorizer to extract features. The extracted features are then passed through Principal Component Analysis (PCA) to reduce the dimensions of the data. While TF-IDF helps in capturing the unique characteristics of fake news articles that may contain distinctive terms, Count Vectorizer simply counts the occurrences of words in a document. Although one can argue that TF-IDF might be more suitable for the fake news detection task, this study demonstrates that both methods are effective, with the models producing the same accuracy in both cases.

The study "*A Benchmark Study of Machine Learning Models for Online Fake News Detection*" conducted by Khan *et al.* [KKA⁺21] aims to evaluate and compare the performance of various machine learning models in detecting online fake news. In this study, three datasets were used: Liar, Fake or Real News, and Combined Corpus. The "Liar" dataset consists of 12.8K human-labeled short statements, with six labels of truthfulness. Since the goal was to classify news as real or fake, the dataset was transformed into two labels: true and fake. The "Fake or Real News" dataset included news from the 2016 USA election cycle, with equal allocation of fake and real news. The Combined Corpus contained around 80k news articles, with a diverse range of topics. Data preprocessing was performed, step that included the removal of IP and URL addresses, stop words, and stemming. Different features were studied, such as lexical and sentiment features, n-gram features, Empath generated features, and pre-trained word embeddings. Various models were evaluated, including traditional machine learning models (SVM, LR, Decision Tree, AdaBoost, Multinomial Naive Bayes, k-NN), deep learning models (CNN, LSTM, Bi-LSTM, C-LSTM, HAN, Convolutional HAN), and advanced language models (BERT, RoBERTa, DistilBERT, ELECTRA, ELMo). The models were combined with several feature extraction techniques, including TF-IDF. When it comes to traditional machine learning models, Naïve Bayes with TF-IDF proves to perform the best, with an accuracy of 93%. It can also be noticed that the Naïve Bayes model has almost the same accuracy as LSTM models, both giving satisfactory results, but the latter is prone to overfitting and is greatly influenced by the size of the dataset. Out of all models, the best performing ones, particularly pre-trained language models like BERT, achieved

over 96% accuracy on two datasets (Combined Corpus and "Fake or Real News"). However, for the "Liar" dataset, which consisted of significantly shorter articles, the highest accuracy achieved was 75%. The study concluded that article length had an impact on model performance, longer articles providing more information for accurate classification. Furthermore, the Combined Corpus's misclassification analysis showed that the words "said," "study," and "research" were commonly used in fake news that were mistakenly categorized as real, which implies that false news providers modify quotes to support their own goals and increase the credibility of their publications.

Vijayaraghavan *et al.* [VWG⁺20] tested different models in their study, both traditional ones, such as Logistic Regression, and neural networks, such as LSTM. After preprocessing the data, different natural language processing models were applied, including CountVectorizer, TF-IDFVectorizer, and Word2Vec, to convert the text into numbers. When using TF-IDF for feature extraction, Logistic Regression had the highest accuracy, while Random Forest and neural networks did not provide the same satisfactory result. Conversely, neural networks performed best when combined with other feature extraction techniques, such as Count Vectorizer and Word2Vec. The combination of CountVectorizer and LSTMs is the best model, as it makes use of the architecture and word dependencies that LSTMs can capture.

Similarly, Samir Bajaj conducted a study [Baj17] that evaluates the performance of machine learning models and neural networks on a fake news dataset. Additionally, they tried a different design for CNN, adding an attention-like mechanism. The goal is to see if they can improve the features generated by the CNN by considering the influence of previous word combinations. This additional step is expected to improve the model's performance in understanding the relationships between words in a text. The study proves that this method was, indeed, effective, having CNN with max pooling and attention mechanism producing the highest precision of 0.97. Logistic Regression also produced satisfactory results, with a precision of 0.96. However, it achieved a lower F1 score compared to the other models, which further emphasizes the importance of nonlinear components in accurately capturing the relationships between features. Additionally, it is observed that, while the precision values were consistently high across all models, only the feedforward network and RNNs with complex activation units were able to attain a recall value higher than 0.7, demonstrating their capacity to correctly classify a significant number of positive samples.

Chapter 4

Natural Language Processing

Simply put, Natural Language Processing (NLP) is where computer science, linguistics and artificial intelligence meet. It is a subfield whose purpose is to study how a computer can interpret and "understand" human language, including the emotional tone conveyed through speech or text. Moreover, NLP aims to enable computers to not only understand, but also generate natural language, thus making the interaction between a computer and a human more natural [Gya22].

This chapter begins by following the evolution of Natural Language Processing (NLP) throughout history. It then explores various aspects of NLP, such as text pre-processing methods, feature extraction methods, and a thorough analysis of TF-IDF.

4.1 History of NLP

The field of Natural Language Processing can be traced back to the 1950s, when computer scientists began taking into consideration the possibility of programming computers to understand and respond to human language. The most notable study in this direction is Alan Turing's seminal paper, *"Computing Machinery and Intelligence"*. Turing's paper starts off by posing the question *"Can machines think?"* [Tur50], after which he proceeds to address common objections against said question and introduces what he called the "imitation game". There are three participants in the game: a human judge and two other participants, one of which is a machine. The judge should not know which one is the human and which is the machine. The purpose of the game is to see if the judge can tell the machine from the human based solely on their responses to questions. If the machine can successfully mimic human conversation to the point that the judge is unable to distinguish it from the human participant, then it can be considered intelligent and has passed the Turing test.

One of the first chatbots ever created that attempted at passing the test is ELIZA, developed by Joseph Weizenbaum in 1966 [Jos66]. ELIZA was designed to mimic a

psychotherapist, and it responded to questions with pre-written responses based on keywords. The responses were meant to be supportive and empathetic, giving the impression of being able to hold a conversation. While ELIZA's capabilities were limited, it was still an important breakthrough in the field of NLP and it showed the possibility of machines to simulate human-like interaction.

Despite the significant advancements in Artificial Intelligence, no models have completely passed the Turing test. While for many years the Turing test has been used as a standard for AI models and considered a goal that needs to be reached in order to decide whether a certain model is successful or not, it is not the best way to measure a machine's ability to display intelligence. Nonetheless, achieving a passing score on the Turing test remains an important milestone, and several models have come very close to doing so [Thi22].

4.2 Text preprocessing

When working with raw text data, it is important to understand that not all of the text will be useful for feature extraction. Even more than that, unnecessary "noise" will make the model less effective. Therefore, text preprocessing is a crucial step in NLP, as it helps clean up the data in order to make it readable for machines. Normally, this process consists of three steps: [Agr22]

- Noise removal

This step involves the elimination of irrelevant content that does not contribute to the context of the data. Examples of such content include language stop-words (common words such as "the", "of", "in", etc.), URLs or hyperlinks, social media entities (hashtags, for example) and punctuation. By removing all such noisy entities, the text is effectively cleaned and made suitable for further analysis.

- Text normalization

Another type of irrelevant content is using multiple representations of a single word (for example, "reader", "reading", "reads" are all a variation of the same word, "read"). Two common text normalization techniques are stemming (a basic process that strips word suffixes) and lemmatization (an organized procedure that uses vocabulary and morphological analysis to obtain a word's root form).

- Word standardization

Text data may contain words or phrases not found in standard lexical dictionaries, such as acronyms, hashtags, and colloquial slangs, which can cause

problems for search engines and models. To address this issue, regular expressions and manually prepared data dictionaries can be used to remove this noise.

4.3 Text to features

In order to analyse the preprocessed text, it is necessary to transform it into features. Features are data that is ready to be given as input to machine learning algorithms. The primary objective at this stage is to represent text as vectors, which is why it is often referred to as "text vectorization". Here are some common techniques: [Duq20]

- Bag-of-words (BoW)

This is a simple technique for representing text data as a set of word counts. The idea is to create a "bag" of words for each document by counting the frequency of each word in the document. The downside of this method is that it doesn't capture the order of words in a document, which leads to ruining the semantic meaning of the initial text.

- Term frequency-inverse document frequency (TF-IDF)

This is a more complex and very common algorithm that is similar to Bow, but it also takes into consideration the order of the words. What it does is assigning a weight to each word based on its frequency in the document.

- Word embeddings

This is more recent and more advanced than the methods discussed previously, and it involves the use of Neural Networks. It creates vector representations of words, which are also designed to capture the semantic meaning of words. As a result, words with similar meanings are represented as similar vectors.

4.4 TF-IDF

This project will only use TF-IDF for text preprocessing, so let us dive deeper into the theoretical concepts of it.

TF-IDF (Term Frequency-Inverse Document Frequency) is a popular technique used in NLP to determine the importance of words in a document corpus. It combines two fundamental concepts: Term Frequency (TF) and Inverse Document Frequency (IDF).

Term Frequency (TF): TF calculates how many times a term appears in a document by using the formula:

$$TF = \frac{\text{Number of occurrences of a term in a document}}{\text{Total number of words in the document}}$$

A higher TF value for a certain word will indicate that the word appears more frequently in the document.

Inverse Document Frequency (IDF): IDF measures the rarity of a word across the entire document corpus by using the formula:

$$IDF = \ln \left(\frac{\text{Total number of documents in the corpus}}{\text{Number of documents containing the term}} \right)$$

Words' importance should not be treated equally, and IDF is used to take care of that aspect. Frequent words will have a lower weight, considering the fact that they are less informative, while unique words will have a greater weight.

TF-IDF: TF-IDF is calculated by multiplying the TF value of a word with its IDF value: `[scib]`

$$TF-IDF = TF \times IDF$$

Words that are common in a small number of documents will have a higher TF-IDF value, which means that they are more important than common words that exist in many documents, such as stop words. For example, if we find that a certain word appears in all documents, the IDF value will be $\ln(1)$, which is 0, meaning that the word does not contribute to the importance of a document. `[QA18]` `[R+03]`

Chapter 5

Machine Learning

Machine Learning (ML) is a subfield of artificial intelligence (AI) that is responsible of training algorithms in order to make predictions or decisions based on input data. These algorithms are made to identify patterns and they improve their performance over time as more data becomes available.

We've seen Machine Learning develop a lot through the years and has found applications in numerous domains. One of the major breakthroughs in the history of Machine Learning would be the creation of the perceptron algorithm, by Frank Rosenblatt in 1957. A perceptron is essentially a single-layer neural network, and it paved the way for the development of more advanced models [Abd20]. There have been achievements in game playing as well, such as Arthur Samuel's checkers-playing program in the 1950s. This was a landmark achievement in machine learning, as it was the first time a computer had beaten a human player at a game [Sam59].

This section will present the types of ML algorithms and it will explain in more detail the main algorithms that we used in our project.

5.1 Types of Machine Learning algorithms

Machine learning algorithms can be classified into four types, each designed for specific requirements and datasets. These algorithms differ in their approach, and by understanding them, researchers and practitioners can select the most appropriate model for their problem at hand [Sal19].

- Supervised learning

In this type of algorithms, the dataset is labeled and classified by users before being provided to the algorithm. The aim is for the algorithm to accurately predict the output for new input data based on the patterns it has learned from the training dataset.

- Unsupervised learning

Compared to supervised learning, unsupervised learning does not rely on labeled data for training. Instead, the algorithm is given an unlabeled dataset and is tasked with finding patterns within the data on its own. The goal is to discover hidden relationships or groupings in the dataset, without being provided an answer key.

- Semi-supervised learning

These algorithms are the middle ground between supervised and unsupervised ones. The algorithms are trained on a dataset that is partially labeled, using the labeled data to learn patterns that can be applied to the unlabeled data. Semi-supervised learning is often used when labeling each entry in the dataset is a time-consuming process, but precision is still required in the algorithm's output.

- Reinforcement learning

In this case, the algorithms learn to make decisions based on the feedback they receive from the environment. The algorithms learn through trial and error, with the goal of taking actions that lead to positive outcomes and avoiding actions that lead to negative outcomes. This approach has been proven effective in various fields, including game playing and robotics.

5.2 Logistic Regression

Logistic regression is a popular classification algorithm based on probability and used for classification problems. The outcome ranges from 0 to 1, representing the probability of success. Logistic regression applies a logit transformation to the odds, function defined as $\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$

Furthermore, this transformation is modeled using the following logistic function:

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

which will give us an S-shaped curve, as shown in [5.1](#)

The algorithm's goal is to estimate the coefficients that best fit the data, step for which we usually use maximum likelihood estimation (MLE), which involves finding the parameter values that maximize the likelihood of the observed data.

Once the optimal coefficients are found in logistic regression, the conditional probabilities are calculated for each observation. These probabilities are summed and used to predict the binary outcome: probabilities below 0.5 are predicted as 0, while those above 0.5 are predicted as 1. [\[IBM\]](#)

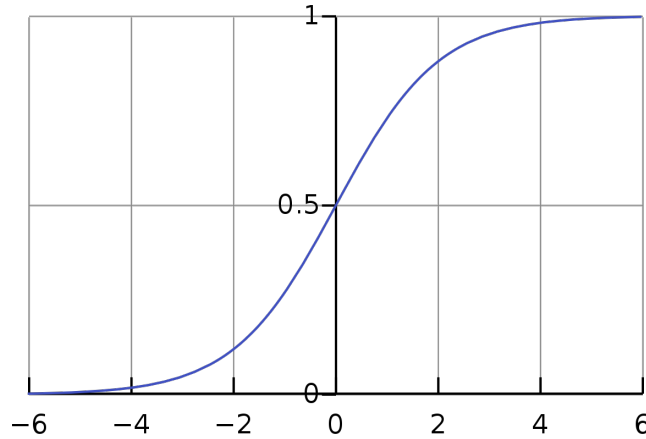


Figure 5.1: Sigmoid function [Qef]

5.3 Decision Trees

A decision tree is a classifier that partitions the instance space using nodes in a rooted tree structure. The tree has a "root" node without incoming edges, and other nodes have exactly one incoming edge. Internal nodes perform tests on attributes, splitting the space into sub-spaces based on attribute values. Leaves represent classes or probabilities. [RM70]

The algorithm aims to separate the dataset into pure leaf nodes, where all data points belong to the same class. However, it often results in mixed leaf nodes. In such cases, the algorithm assigns the most common class. Building the ideal tree is computationally infeasible, so a greedy approach is used to make locally optimal decisions. The algorithm selects the best split by evaluating the purity of resulting nodes using loss functions, such as Gini Impurity: [Ben21]

$$Gini = 1 - \sum_{i=1}^C p_i^2$$

where $p_i = \frac{\text{number of instances belonging to class } i}{\text{total number of instances}}$

5.4 Passive-Aggressive Classifier

Passive-Aggressive algorithms are online algorithms commonly used in big data applications. In machine learning, an "online algorithm" refers to an algorithm that learns from data instances sequentially, as they arrive one by one. Online algorithms update their model by making incremental adjustments to the parameters, without needing to use the entire dataset at once. This is different from batch learning, where all the data is processed together [Kar92].

Passive-Aggressive algorithms are named based on their behavior:

- Passive

If the prediction is accurate, no modifications are made to the model, indicating the example does not significantly impact it.

- Aggressive

If the prediction is incorrect, adjustments are made to the model in order to rectify the error.

[alo23]

The goal of the Passive-Aggressive algorithm is to find a weight vector w that can accurately classify new, unseen data points. Initially, we start with an initial weight vector w_0 . Given training data (x, y) with binary labels, the Passive-Aggressive algorithm finds a weight vector w that accurately classifies new data.

For each example, the algorithm predicts a label \hat{y} . If \hat{y} matches the true label y , no update is needed. Otherwise, the weight vector is adjusted using a parameter C that controls the aggressiveness of the update in the algorithm, and a loss function L that measures the prediction error of the model [Lav15].

Chapter 6

Application development

The analysis of the application is the main topic of this section. We start off by providing clear explanation of the problem at hand. Afterwards, we provide an overview of the application's structure and functionality. The chapter also highlights the frontend and backend components, along with the technologies used. In addition, the testing procedure is described, and a user manual is offered to demonstrate how users can use the application.

6.1 Problem Definition

The problem addressed in this thesis is the detection of fake news using supervised machine learning techniques. The goal is to develop a model that can accurately classify news articles as either real or fake.

This problem can be formalized by particularizing a general mathematical definition of a supervised machine learning algorithm. We will have an input space X , which contains features derived from all the news articles in our dataset, and an output space Y , consisting of all possible labels. Considering the fact that we have a binary classification problem, Y will be $\{0, 1\}$, where 0 represents a real news article and 1 represents a fake news article.

Furthermore, from our dataset D consisting of n news articles, we will have to create our input space $X = (x_1, x_2, \dots, x_n)$. We denote each feature vector as $x_i = (x_i^1, x_i^2, \dots, x_i^d)$, where d represents the total number of unique terms across all the news articles, and x_i is the feature vector for the i^{th} news article in D [Bro19].

To achieve accurate predictions, we need to train a model that learns the underlying patterns in the dataset. This trained model will be capable of mapping the feature vectors of news articles to their corresponding labels. The objective is to achieve high precision and high recall simultaneously, in order to minimize both false positives (misclassifying real news as fake) and false negatives (misclassifying

fake news as real) [Pow20].

After extracting the features and training the model, we can proceed to the prediction phase, where our goal is to classify news articles as either real or fake. This process can be represented by a function $f : X \rightarrow Y$, denoted as $f(X) = Y$, where f takes as input a feature vector and it returns a label from Y , which can be either 0 (real news) or 1 (fake news).

6.2 Analysis and application architecture

6.2.1 Functional and non-functional requirements

Functional Requirements

The functional requirements (see Use Case Diagram 6.1) of the fake news detection system specify the capabilities of the application. For our application, we have the following functional requirements:

1. User Input: The front page has input fields where the user can enter the title, author, and text of a news article.
2. Result Display: After the user inputs the necessary information, the results will be displayed on the screen in a table format. The table will include the prediction generated by each machine learning model and feature extraction technique used in the project.

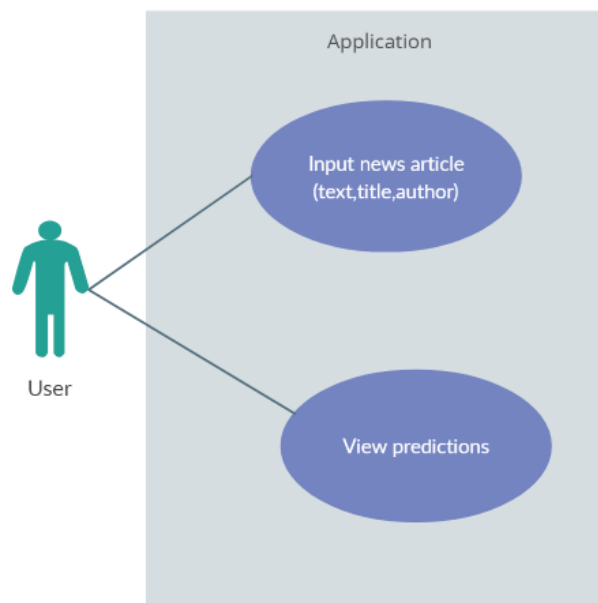


Figure 6.1: Use cases diagram

Non-functional Requirements

The non-functional requirements focus on the qualities of the application rather than specific functionalities. They generally describe aspects that contribute to user satisfaction. In our project, we have the following non-functional requirements:

1. **Real-time Results:** To provide immediate feedback, we use the `useCallback` function from React in order to retrieve and display the results in real-time as the user inputs information.
2. **Ease of Use:** The application aims to be simple and user-friendly. Additionally, it provides clear instructions for inputting the required information.

6.2.2 Application architecture

The application uses a client-server architecture, which enables effective communication between the server-side processing and the client-side application. Additionally, we present a diagram [6.2](#) to showcase the use of the client-server architecture and the way the main components interact with each other.

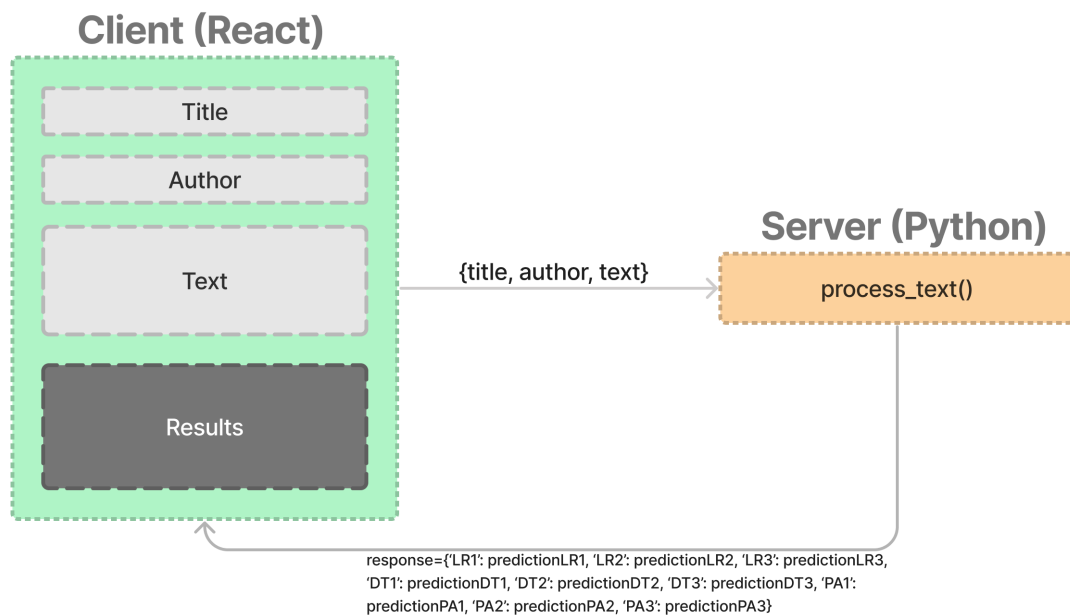


Figure 6.2: Client-server communication

Three input fields are present on the frontend interface: one for the title, one for the author, and one for the news article's text. As the user types in these details, the frontend sends this information to the server for real-time processing. The server then applies the trained machine learning models to categorize the news as true or fake using the provided title, author, and text. After obtaining the predictions, the

server will send them to the client side, which will display the results in a table on the screen.

Frontend

There are multiple files to be found, especially .css files, but let's take a look at the most important ones. The project structure, as depicted in Figure 6.3, follows the typical structure of a React project, with `App.js` rendering all the other components. As for the custom implemented components, `header.js` is responsible for the navigation bar at the top of the webpage. On the other hand, `page.js` is responsible for rendering the `TableML` component and managing the lower section of the webpage, that consists of an introduction and instructions regarding the application. The communication with the server happens in `tableML.js`. In this file, we can find the form where the user has to input the required information, and the table where we can see the predictions for each combination of model and feature extraction technique (see 6.4). By using the `React useCallback()` hook, the table is updated in real time as the user inputs the information.

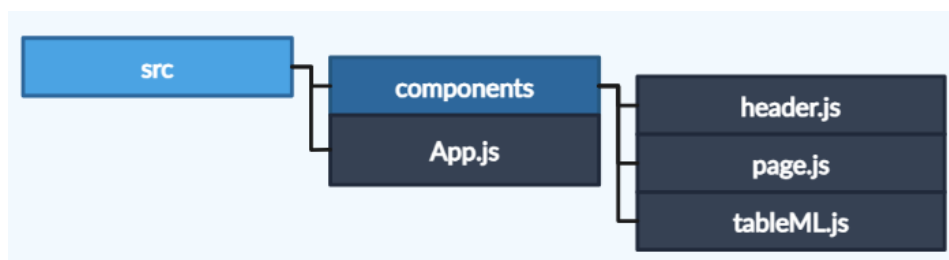


Figure 6.3: Project structure - React

Backend

Once again, we included a figure where can observe the most important files from the backend part (see 6.5). The most important file, `main.py`, is responsible for establishing the connection with the client. Using the Flask framework, we facilitate communication between the React client and the backend. Within the file we also implement the method `process_text()`, which receives a message from the client through the POST method, containing information regarding a news article. After generating the predictions, the same method will send back to the client a dictionary containing those predictions. Furthermore, we have separate files for each model, and the text preprocessing and feature extraction steps happen in the `text_preprocessing.py` file. Within this file, the method `preprocess_text(text)` is tested in the `preprocessing_method_test.py` file. These tests we check if the text is processed as expected, examining functionalities

	Text	Text + Author	Text + Author + Title
Logistic Regression	-	-	-
Decision Trees	-	-	-
Passive-Aggressive Classifier	-	-	-

Figure 6.4: Form and prediction table

such as converting the text to lowercase, deleting numbers, deleting punctuation or deleting stopwords. Lastly, in the `test_models.py` we test our trained models on new, unseen data.

6.3 Technologies

In this chapter, we will go over the various technologies used in our fake news detection project, as they are essential for putting the machine learning models into practice and creating the application. The main technologies utilized are Python and React. Additionally, we used multiple libraries from Python, such as the scikit-learn library, Flask, pandas, and NLTK. From data preprocessing to model training and deployment, each technology has its specific purpose at different project stages.

Python

Python is a flexible, extensively used programming language that is known for its simplicity and readability, as well as for its extensive set of libraries. It is especially used for AI and ML projects, where the algorithms' implementations can be difficult and time-consuming, therefore a well-structured, well-tested environment is essen-

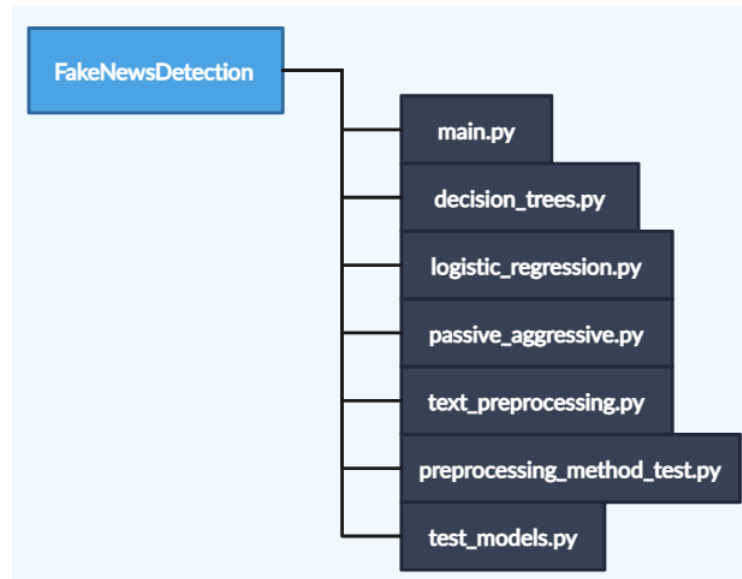


Figure 6.5: Project structure - Python

tial [Bek22]. Due to its wide library support and compatibility with machine learning and natural language processing applications, Python was our top option for the fake news detection project. We successfully handled tasks like data manipulation, data preprocessing and feature extraction using libraries like pandas and nltk. Moreover, its ease of integration with other technologies made it an ideal choice for creating our application and providing a visual representation of our results.

React

React is a popular JavaScript library used for building user interfaces easily. It is known for its declarative approach, which simplifies the debugging process. This approach proves to be a big advantage when managing the state of your application, simply because you do not get overwhelmed by the implementation details of representing that state. It is also known for having a component-based structure, which allows you to build encapsulated components and reuse them for more complex projects [Met]. In our fake news detection project, we chose React so that we can easily create a user-friendly and engaging web application. Most importantly, we could automatically update the user interface in response to predictions produced by our machine learning models, giving users feedback on the reliability of news stories in real-time.

scikit-learn

The scikit-learn library, also known as sklearn, is a powerful machine learning toolkit for Python. It aims to make machine learning accessible to individuals without specialized knowledge, using a high-level programming language. Since it is based on

the scientific Python ecosystem, it is simple to include into applications that are not often associated with statistical data processing [PVG⁺11]. It provides a large set of tools and algorithms for data preprocessing, feature extraction, and model training. To create our fake news detection models, we benefited from scikit-learn's implementation of logistic regression, decision trees, and the Passive Aggressive Classifier. We found it simple to initialize the models, train them using the fit function, and evaluate their effectiveness using scikit-learn's evaluation metrics.

pandas

pandas is a popular Python library that provides data manipulation and analysis functionalities [pan23]. It was a very useful tool during the data preprocessing phase of our project. It allowed us to load and transform the dataset, making sure that it was prepared for training our machine learning models. pandas' DataFrames made it easy to handle our data, and its built-in functions were exactly what we needed in order to easily perform tasks like aggregation and feature engineering.

NLTK (Natural Language Toolkit)

NLTK (Natural Language Toolkit) is an extensive library for natural language processing tasks in Python [Ste23]. We used NLTK to perform various text preprocessing tasks, such as tokenization or stop word removal. NLTK's resources, such as the stop words set, simplified the process of extracting meaningful features from the dataset in order to use it further for the machine learning models.

Flask

Flask is a framework in Python that allowed us to develop the backend of our fake news detection system. We used Flask to create an API that receives requests from the frontend, processes the input data, and returns the predictions made by our machine learning models. Flask's simplicity made it an ideal choice for building the backend infrastructure.

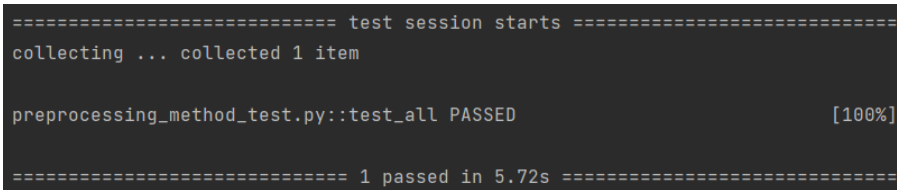
6.4 Testing

In software development, testing is essential for verifying the application's reliability and quality, which is what this chapter focuses on. For the testing part, we verified the text preprocessing method to ensure it gives the correct output before using the vectorizer.

The testing framework used for our project is `pytest`, which provides a simple and efficient way to write and execute tests. We created a separate file `preprocessing_method_test.py` that contains several test methods. They are designed to evaluate the functionality of the `preprocess_text()` function from the `text_preprocessing` file. We included the following methods:

- `test_lowercase()`: Tests the conversion of text to lowercase.
- `test_punctuation()`: Tests the removal of punctuation marks.
- `test_numbers()`: Tests the removal of numeric characters.
- `test_stopwords()`: Tests the removal of stopwords.
- `test_all()`: Executes all the test methods, with the execution result shown in [6.6](#).

These tests evaluate the expected output against the actual output of the `preprocess_text()` function using the `assert` statement.



```
==== test session starts ====
collecting ... collected 1 item

preprocessing_method_test.py::test_all PASSED [100%]

==== 1 passed in 5.72s =====
```

Figure 6.6: Testing result

6.5 User Manual

In this section, we will show how the user can interact with the application and how the predictions are shown in the table.

The upper part of the web page consists of a header and a short paragraph as depicted in [6.7](#).

Scrolling down the same page, we can find the input fields where the user can write the news article information, as well as the table where the predictions will be shown for each combination of models and features (see [6.8](#)).

Since the models will only predict on the combinations of features mentioned in the table, if the user only provides the title and the text, without the author, the models will predict using only the text of the article. The idea behind this choice is that we wanted to see if certain authors' names are related to news articles that are unreliable. In [6.9](#) we can see an example of an article and its' prediction. Each prediction resulted in 0, which means the news is real.

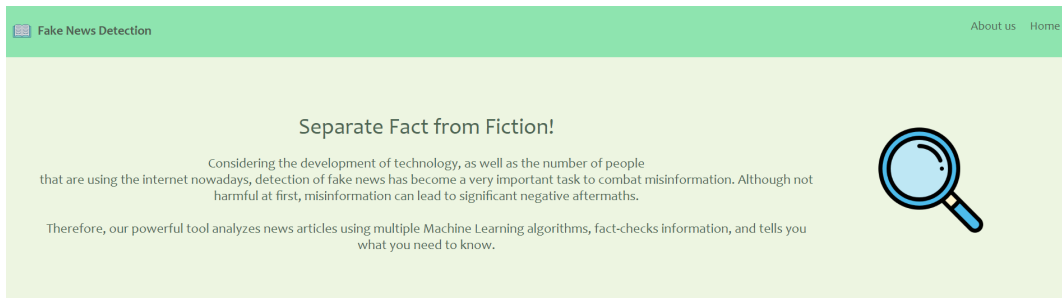


Figure 6.7: Upper part of the page

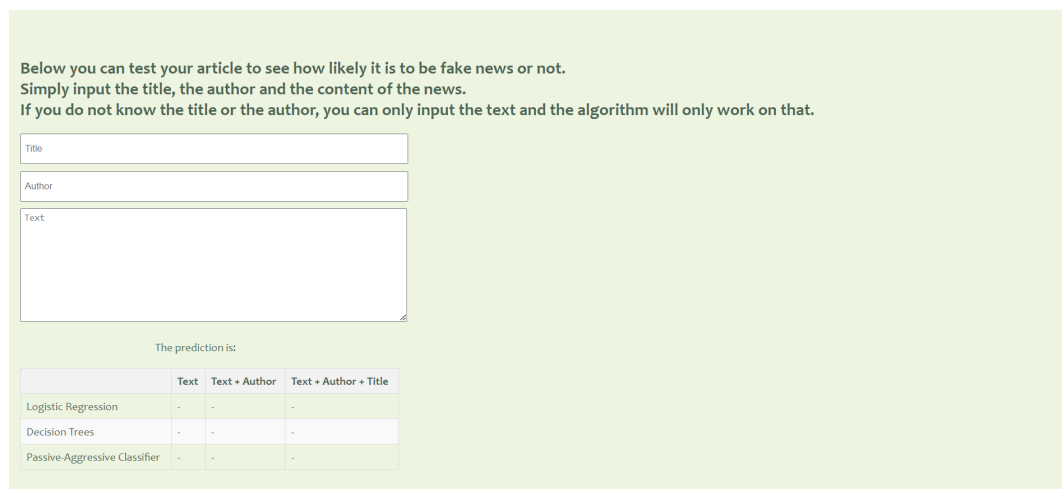


Figure 6.8: Lower part of the page

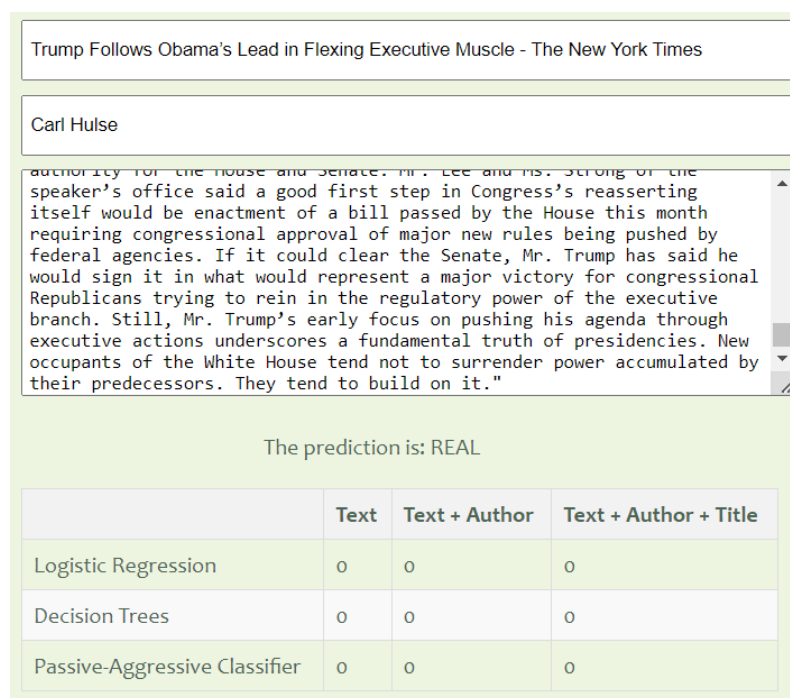


Figure 6.9: Example of a reliable article

Although the user can get a prediction if they only know the text of the news article, it is important to notice that, in some cases, complete information will provide different results. For example, in 6.10a only the text is provided, with the result being inaccurate, since the article is from *test.csv* and we know the prediction should be 1. After writing complete information, as shown in 6.10b, we notice that the result for one model has changed, showing the correct prediction. The text above the table that mentions the result ("FAKE"/"REAL") has not changed because the number of 0's is bigger than the number of 1's present in the table. However, one can notice the change in the table and double-check the information before reaching a conclusion.

"83-Year-Old Farmer Who Shot Burglar in the Foot Forced to Pay £30,000 Legal Fees"

Author

Mr. Hugill, he said, "we considered all the evidence in this case extremely carefully, and took full account of the situation Mr Hugill found himself in that evening. "We are satisfied that there was sufficient evidence to put the matter before a court and that it was in the public interest to do so." Back to work and ploughing his fields ready to sow wheat on Tuesday, Mr. Hugill thanked supporters and said he hopes to get his guns and ammunition back after they were seized by police so he can get back to shooting rabbits on the farm. "I have been shooting since I was less than 12 years old, and it has been part of my life on the farm," he told MailOnline. "I'm glad to get back to work and put the last 16 months behind me. Farming has been my life. "

The prediction is: REAL

	Text	Text + Author	Text + Author + Title
Logistic Regression	0		
Decision Trees	0		
Passive-Aggressive Classifier	0		

(a) Example of an unreliable article - text only

"83-Year-Old Farmer Who Shot Burglar in the Foot Forced to Pay £30,000 Legal Fees"

Virginia Hale

Mr. Hugill, he said, "we considered all the evidence in this case extremely carefully, and took full account of the situation Mr Hugill found himself in that evening. "We are satisfied that there was sufficient evidence to put the matter before a court and that it was in the public interest to do so." Back to work and ploughing his fields ready to sow wheat on Tuesday, Mr. Hugill thanked supporters and said he hopes to get his guns and ammunition back after they were seized by police so he can get back to shooting rabbits on the farm. "I have been shooting since I was less than 12 years old, and it has been part of my life on the farm," he told MailOnline. "I'm glad to get back to work and put the last 16 months behind me. Farming has been my life. "

The prediction is: REAL

	Text	Text + Author	Text + Author + Title
Logistic Regression	0	0	0
Decision Trees	0	0	1
Passive-Aggressive Classifier	0	0	0

(b) Example of an unreliable article - complete information

Chapter 7

Classification methodology

In this section, we will delve deeper into the specifics of the dataset that we used, the text preprocessing and feature extraction stage and the machine learning models implemented. Lastly, we will have a comprehensive analysis of our results, accompanied by visual representations, which we will compare with relevant findings of other researchers’.

7.1 Dataset

The dataset used in this project can be found on Kaggle [\[Kag\]](#). It consists of a multitude of news articles, as well as other attributes such as title or author. The fields for each entry in the dataset is as followed:

- **id**: a unique identifier for each news article.
- **title**: the title of a news article.
- **author**: the author of the news article.
- **text**: the content of the news article.
- **label**: a binary label indicating the reliability of the article:
 - **1**: fake news
 - **0**: real news

In the dataset archive, there are multiple files available. However, for training the dataset, we only used the *train.csv* file. It contains 20387 unique values, distributed almost equally, with 10387 values having the label 0, and 10413 having the label 1. For better visualization of how the data is divided, we can take a look in the figure [7.1](#).

used to convey correct information in trustworthy news pieces, these terms might be used to manipulate or mislead readers in fake news articles.

As mentioned above, there are other files present in the dataset, specifically *test.csv* and *submit.csv*. In *test.csv* we can find the same columns as in *train.csv*, except the **label** column. In *submit.csv* we can find the columns **id** and **label**, where, for each id in *test.csv*, we can see its label. There is a total of 5200 entries in this file, split into 2339 entries with label 0, and 2861 entries with label 1 7.3. This file will be used later so that we can test the already trained models on new, unseen data and check their performance in that scenario.

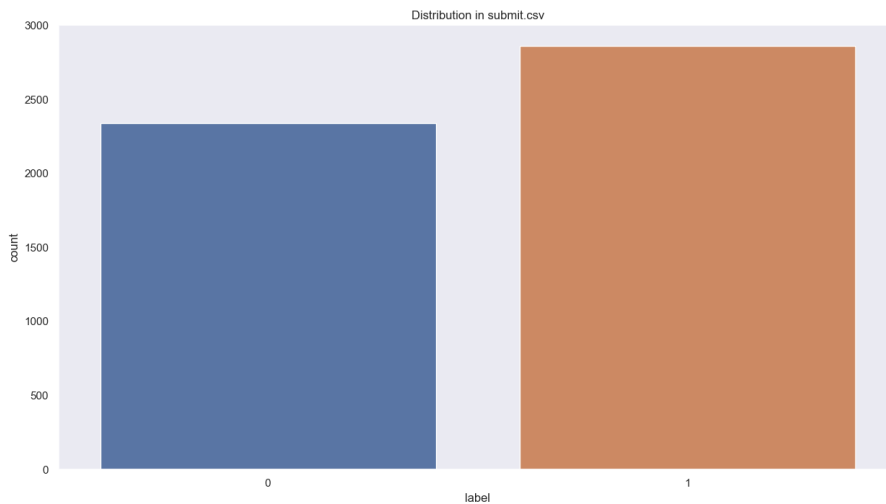


Figure 7.3: Distribution of fake and real news in submit.csv

7.2 Text preprocessing and feature extraction

The goal of text preprocessing is to clean and transform the raw text data into a format that is appropriate for analysis. For this part, we used the following steps:

- **Lowercasing:** the text is changed to lowercase in order to avoid variations brought on by different letter casing

```
text=text.lower()
```

- **Punctuation Removal:** symbols do not contribute to the meaning of the text, therefore they need to be removed

```
text=text.translate(str.maketrans(' ','',
string.punctuation))
```

- **Number Removal:** we remove numeric characters using regular expressions, as they also do not add much to the text's meaning

```
text=re.sub(r'\d+', '', text)
```

- Tokenization: the text is split into individual words using the `word_tokenize` function from the NLTK library. This way, we divide the text into smaller chunks, enabling further word-level analysis.

```
tokens=word_tokenize(text)
```

- Stopword Removal: common words ("the", "is", "and", etc.) have no significance in the text and are therefore removed using a predefined list of stopwords from the NLTK library.

```
tokens=[token for token in tokens if token not in  
stop_words]
```

The feature extraction procedure is then applied to the preprocessed text. The preprocessed text is turned into numerical features using the Term Frequency-Inverse Document Frequency (TF-IDF) method, which captures the significance of words in relation to the document's and the dataset's overall context. This is accomplished by using the TF-IDF vectorizer from the

`sklearn.feature_extraction.text` module. In this step, we demonstrate three scenarios of feature extraction:

- Text Only: only the preprocessed text is used as input for feature extraction.
- Text and Author: the preprocessed text is combined with the original author information and given as input for feature extraction.
- Title, Text, and Author: the preprocessed title, text, and author information are combined and given as input for feature extraction.

The resulting TF-IDF matrices are then used as input for the machine learning models to train and classify news articles as either real or fake.

7.3 Models

After preprocessing the text and extracting features, we proceed to train and test various models to determine if news articles are authentic or not. In this section, we will talk about model selection and training.

Logistic Regression (LR)

Given the nature of the problem as a binary classification task, logistic regression would be one of the most suitable classification algorithms. By analyzing the input

features, logistic regression evaluates the probability of a news article falling under the fake news category.

Decision Trees (DT)

Another type of algorithms frequently used for classification tasks is decision trees. They create a structure similar to a flowchart in order to make decisions based on features in the data. Each internal node of the tree represents a feature, and each leaf node represents a class label (in our case, real or fake news).

Passive Aggressive Classifier (PA)

The Passive Aggressive Classifier is another algorithm we use for fake news detection. Due to its online learning capabilities, which make it well-suited for effectively managing vast amounts of data, it is very helpful for our work. The algorithm makes updates to its model parameters based on individual training instances, allowing it to adapt to changing patterns in the data.

The TF-IDF matrices obtained from the feature extraction step are used as input to train the models. Thus, we use three variants of each of these models, denoted as LR1, LR2, LR3 for logistic regression, DT1, DT2, DT3 for decision trees, and PA1, PA2, PA3 for passive aggressive classifier. Each option corresponds to the three scenarios of feature extraction mentioned earlier:

- **LR1, DT1, PA1:** only the preprocessed text is used as input.
- **LR2, DT2, PA2:** the preprocessed text and author information are combined as input.
- **LR3, DT3, PA3:** the preprocessed title, text, and author information are combined as input.

All of these models can be found in the `scikit-learn` library. They are trained using the `fit` function, and the performance metrics such as accuracy, precision, recall, and F1-score are calculated using the test set. For each model variant, we iterate the training process 100 times and record the performance metrics, including average accuracy, precision, recall, and F1-score. The trained models are then saved as pickle files for future use.

7.4 Results and comparisons

This chapter’s goal is to present and evaluate the project’s findings while also conducting a comparison of our findings with those of other researchers. This chapter highlights the performance of the logistic regression, decision trees, and passive aggressive classifier models in detecting fake news, while showcasing the impact of different text preprocessing techniques, including text-only, text and author, and text, title, and author.

Therefore, we present our results of our evaluation on *train.csv* for each combination of feature extraction and machine learning models (see Table 7.4, Table 7.5, and Table 7.6). The objective was to emphasize the importance of additional information when determining whether a certain article contains false information or not. Unsurprisingly, the content of the article, containing the most comprehensive information, is the biggest factor when deciding the output. Additionally, we considered the author’s name alongside content, motivated by the observation that certain authors are known for writing false news. It was anticipated that the conclusion would change if specific authors were associated with specific labels. However, the difference between considering only the article content and including the author’s name was marginal, the accuracy increasing with approximately 0.01. Last but not least, we looked into the effects of including the article title along with the content and author. In the case of logistic regression and passive-aggressive classifier, the increase was even smaller than it was at the previous step. In contrast, we can observe a higher increase for the decision trees model, that went from 0.87, to 0.88, and eventually reaching 0.93, which is a significant difference.

	Accuracy	Precision	Recall	F1-score
Logistic Regression	0.943	0.936	0.951	0.944
Decision Trees	0.874	0.869	0.881	0.875
Passive-Aggressive Classifier	0.957	0.954	0.961	0.958

Figure 7.4: Results for text preprocessing only

All in all, the results are satisfactory, but we wanted to test the models on new data as well, for which we used the *test.csv* and *submit.csv* files. For this evaluation, we used two vectorizers from the training phase: one that considered only the **text** column, and another that combined the **text**, **author** and **title** columns. The difference between the two was minimal, so the results (see Figure 7.7a) we present here are the ones we achieved by using the second vectorizer, and the confusion matrix

	Accuracy	Precision	Recall	F1-score
Logistic Regression	0.947	0.941	0.954	0.948
Decision Trees	0.882	0.884	0.879	0.882
Passive-Aggressive Classifier	0.963	0.961	0.965	0.963

Figure 7.5: Results for text preprocessing and author

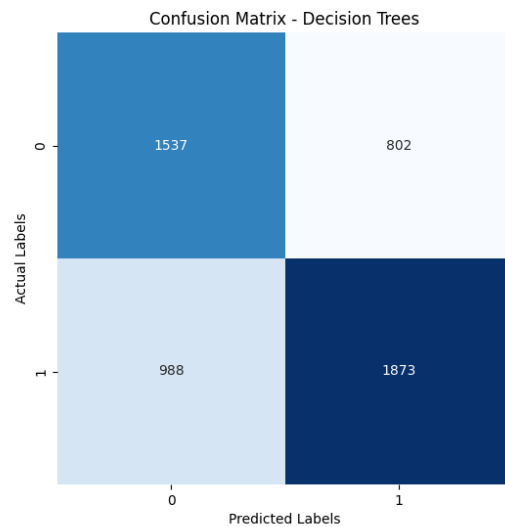
	Accuracy	Precision	Recall	F1
Logistic Regression	0.948	0.943	0.954	0.948
Decision Trees	0.931	0.926	0.936	0.931
Passive-Aggressive Classifier	0.964	0.962	0.967	0.965

Figure 7.6: Results for author, text and title preprocessing

corresponds to the model that had the highest scores, which is the decision trees model (see Figure 7.7b). In this case, the performance drops to around 64%, indicating the hardship of generalizing the models to unseen data. The presence of unseen patterns in the test dataset can be an important factor for this discrepancy, indicating the difference in content between the training set and this set. These findings highlight the importance of evaluating models on multiple and diverse datasets in order to determine their applicability and generalization potential in the real world.

	Accuracy	Precision	Recall	F1
Logistic Regression	0.641	0.687	0.638	0.661
Decision Trees	0.655	0.700	0.654	0.676
Passive-Aggressive Classifier	0.64	0.690	0.626	0.657

(a) Results from test.csv and submit.csv



(b) Confusion matrix for Decision Trees

As shown by Umer *et al.* [UIU+20], the length of the feature vector can have a significant impact on the results given by the models. By reducing the dimensionality, the model has to learn from a more concise set of features, which will lead to a computationally more efficient learning process. When dealing with a large number of features, the performance of machine learning algorithms may degrade, as it will be more difficult for it to understand the importance of feature. Therefore, we tried adding another step in our text preprocessing stage: stemming. In our `preprocess_text(text)` method, after tokenizing the text, we included the following lines:

```
stemmer = PorterStemmer()
tokens = [stemmer.stem(token) for token in tokens]
```

This step significantly reduced the number of features, however, after once again testing the models on the *train.csv* file, the results did not differ that much. The accuracy scores are shown in 7.8 for each model that received features extracted from the author and the preprocessed text and title. Testing these newly-trained models on the *test.csv* file, the results were not any better.

	Text	Text + Author	Text + Author + Title
Logistic Regression	0.916	0.917	0.918
Decision Trees	0.882	0.884	0.927
Passive-Aggressive Classifier	0.955	0.961	0.961

Figure 7.8: Results for author, text and title preprocessing

The observations above lead to the conclusion that the chosen feature extraction method does not perform very well on new, unseen data. TF-IDF relies on a predefined vocabulary derived from the training data, which makes it more difficult for it to provide good results on unseen data. Another drawback is that TF-IDF focuses on individual words and their frequencies within documents, it does not consider the broader context, which is important in fake news detection. Furthermore, we can pose the question "Why did we get over 90% accuracy When we tested the models on the test split from *train.csv*?". One possible explanation is that information from the test split unintentionally entered the training process, causing the vectorizer to learn patterns from the test split as well. This phenomenon is known as data leakage and can result in overly optimistic performance results.

In the study conducted by Vijayaraghavan *et al.* [VWG+20], similar to our ap-

proach, they used the same dataset and evaluated various machine learning models including neural networks, Support Vector Machine (SVM), and Logistic Regression. They mention that they only focused on the *train.csv* file, from which they only take into consideration the **text** column. Additionally, besides TF-IDF, they also used CountVectorizer and Word2Vec. To identify the optimal hyperparameters, they performed a grid search on each model, and for Logistic Regression in combination with TF-IDF, they found that the best option is initializing the model as `modelLR = LogisticRegression(solver='sag', C=7.74)`. The **solver** parameter mentions the algorithm used for optimizing the model (in this case, they found Stochastic Average Gradient to be the best option), and the **C** parameter specifies the regularization strength, smaller values meaning a stronger regularization, that will help avoiding overfitting [scia]. Inspired by their approach, we trained a Logistic Regression model the same way on *train.csv*, following similar preprocessing steps (excluding stemming, which was not mentioned in their study). Like them, we restricted our feature extraction to the **text** column only. Furthermore, we present the results we obtained after implementing the Logistic Regression model in the same manner, tested on both the *train.csv* and *test.csv* files [7.9]. In their study,

	Accuracy	Precision	Recall	F1-score
Logistic Regression – train.csv	0.954	0.948	0.960	0.954
Logistic Regression – test.csv	0.638	0.687	0.628	0.656

Figure 7.9: Results for Logistic Regression on train.csv and test.csv (only the text column)

Vijayaraghavan *et al.* obtained a result of 94%, and we obtained a slightly higher result of 95%. The difference is that they did not test the model on the unseen data from *test.csv*. When we evaluated the model on this file, the results were once again unsatisfactory, which only strengthens the idea stated above that TF-IDF might not be the best option for our task.

Chapter 8

Conclusions and Future Work

Conclusions

In this study, we examined the impact of several text preprocessing techniques in order to identify fake news using a variety of machine learning techniques. We obtained information regarding the effectiveness of these methods through our evaluation and comparison of Logistic Regression, Decision Trees, Passive Aggressive Classifier models, as well as different feature extraction combinations.

Our findings showed that a key factor in identifying fake news was the article's contents, which was captured by the text column. Additionally, there was a little increase in accuracy when the author's name was taken into account along with the article's content, indicating that some authors may be linked to fake news. With the exception of the Decision Trees model, which displayed a noticeable improvement, adding the article title in addition to the content and author did not result in any obvious performance gains.

The performance decreased to about 64% when we used the *test.csv* and *submit.csv* files to test the models on new data. This highlights the potential presence of unexpected patterns in the test dataset and shows the difficulty of generalizing the models to unknown data. Therefore, we can conclude that it is essential to evaluate the models on multiple datasets to determine their applicability and generalization potential in practical scenarios.

In the text preprocessing stage, we also examined the effects of stemming, which lowers feature dimensionality. However, the results did not differ significantly, so we can argue that our chosen feature extraction method, TF-IDF, did not perform well on new, unseen data. This limitation might be linked to the fact that TF-IDF relies on a predetermined vocabulary that was produced from the training data and concentrates on specific words and frequencies rather than taking into account broader contextual information necessary for fake news detection. One other potential problem would be the unintentional inclusion of information from the test

split during the training process, which led to good results when the models were trained and tested on *train.csv*, but unsatisfactory results when they were tested on *test.csv*.

Future Work

There are several areas for future exploration in fake news detection. One avenue is to investigate more sophisticated techniques for extracting features from the text, such as word embeddings, as they can extract deeper meaning and context from the text, thus potentially improving the accuracy of fake news detection.

Another important area that could be improved is the datasets used in this task. By gathering a greater variety of fake and real news articles, the dataset could be expanded in order to include more articles from various fields (e.g. sports, celebrities news, politics, events, etc). This would give the models a more varied set of instances to learn from and contribute to overcoming the difficulty of generalizing to new data.

In terms of model selection, pre-trained models like BERT can be very helpful, especially for multi-class classification problems. These models have the ability to take into account the context when embedding words, which makes them suitable candidates for improving fake news detection accuracy.

From a practical point of view, a real-time fake news detecting system may also be developed, which would be very helpful. By analyzing news stories as they are published, such a system could allow prompt detection and control of the spread of false information.

In conclusion, by addressing these potential paths for future research, we can progress the field of fake news identification and contribute to the creation of more reliable models that can help stop the spread of misleading information in the digital age.

Bibliography

- [2423] France 24. Truth or fake - ukraine: One year of misinformation and how it has shaped the war narrative, Feb 2023.
- [Abd20] Ahmad Abdullah. Machine learning evolution - the story of perceptron, Oct 2020.
- [AG17] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–236, 2017.
- [Agr22] Raghav Agrawal. Must known techniques for text preprocessing in nlp, Aug 2022.
- [alo23] alokesh985. Passive aggressive classifiers, Jan 2023.
- [ASK⁺20] Akbar, S., Kukreti, D., Sagarika, S., Pal, and J. Temporal patterns in covid-19 misinformation in india, Apr 2020.
- [AYYA20] Iftikhar Ahmad, Muhammad Yousaf, Suhail Yousaf, and Muhammad Ovais Ahmad. Fake news detection using machine learning ensemble methods. *Complexity*, 2020:1–11, 2020.
- [BA20] Stephanie Busari and Bukola Adebayo. Nigeria records chloroquine poisoning after trump endorses it for coronavirus treatment, Mar 2020.
- [Baj17] Samir Bajaj. The pope has a new baby! *Fake news detection using deep learning*, pages 1–8, 2017.
- [Bek22] Angela Beklemysheva. Why use python for ai and machine learning?, Dec 2022.
- [Ben21] Carolina Bento. Decision tree classifier explained in real-life: Picking a vacation destination, Jul 2021.
- [Bro19] Dorian Brown. What is supervised learning? a mathematical perspective, Sep 2019.

- [Bro20] Matt Brown. Fact check: Could taking vitamin c cure - or prevent -covid-19?, Mar 2020.
- [Duq20] Tiago Duque. How to turn text into features, Nov 2020.
- [FAA] Eslam Fayez, Amal Elsayed Aboutabl, and Sarah N. Abdulkader. Automated detection of fake news.
- [Gya22] Gyansetu. What is natural language processing? intro to nlp in machine learning, Sep 2022.
- [IBM] IBM. What is Logistic Regression?
- [Jos66] Joseph Weizenbaum. ELIZA. <https://www.masswerk.at/elizabot/>, 1966.
- [Kag] Kaggle. Fake news detection. <https://www.kaggle.com/c/fake-news/data>.
- [Kar92] Richard M Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future?, 1992.
- [KKA⁺21] Junaed Younus Khan, Md Tawkat Islam Khondaker, Sadia Afroz, Gias Uddin, and Anindya Iqbal. A benchmark study of machine learning models for online fake news detection. *Machine Learning with Applications*, 4:100032, 2021.
- [Lav15] Victor Lavrenko. Ir20.3 passive-aggressive algorithm (pa), Sep 2015.
- [LK22] Velivela Durga Lakshmi and Ch Sita Kumari. Detection of fake news using machine learning models. *International Journal of Computer Applications*, 183(47), January 2022.
- [McG22] Jane McGarrigle. Explained: What is fake news?: Social media and filter bubbles, Jul 2022.
- [Met] Meta. React – a javascript library for building user interfaces.
- [Nat] National Archives and Records Administration. President Discusses Beginning of Operation Iraqi Freedom.
- [pan23] pandas development team. About pandas. <https://pandas.pydata.org/about/>, 2023.
- [Pow20] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

- [PVG⁺11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
- [QA18] Shahzad Qaiser and Ramsha Ali. Text mining: use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, 181(1):25–29, 2018.
- [Qef] Qef. File:logistic-curve.svg.
- [R⁺03] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.
- [RM70] Lior Rokach and Oded Maimon. Decision trees, Jan 1970.
- [Sal19] Isha Salian. Supervised vs. unsupervised learning, Aug 2019.
- [Sam59] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.
- [scia] scikit-learn. LogisticRegression. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.
- [scib] scikit-learn. TfidfTransformer. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html.
- [Sor15] Stuart Soroka. Why do we pay more attention to negative news than to positive news?, May 2015.
- [Ste23] Steven Bird, Edward Loper, and NLTK contributors. Natural language toolkit. <https://www.nltk.org/>, 2023.
- [SVC⁺20] Ica Secosan, Delia Virga, Zorin Petrisor Crainiceanu, Lavinia Melania Bratu, and Tiberiu Bratu. Infodemia: another enemy for romanian front-line healthcare workers to fight during the covid-19 outbreak. *Medicina*, 56(12):679, 2020.
- [The] The Office of the Historian. U.S. Diplomacy and Yellow Journalism, 1895–1898.

- [Thi22] Big Think. The turing test: Ai still hasn't passed the "imitation game", Mar 2022.
- [Tur50] A. M. Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, October 1950.
- [UIU⁺20] Muhammad Umer, Zainab Imtiaz, Saleem Ullah, Arif Mehmood, Gyu Sang Choi, and Byung-Won On. Fake news stance detection using deep learning architecture (cnn-lstm). *IEEE Access*, 8:156695–156706, 2020.
- [Uni] United Nations. UN INSPECTORS FOUND NO EVIDENCE OF PROHIBITED WEAPONS PROGRAMMES AS OF 18 MARCH WITHDRAWAL, HANS BLIX TELLS SECURITY COUNCIL.
- [VWG⁺20] Sairamvinay Vijayaraghavan, Ye Wang, Zhiyuan Guo, John Voong, Wenda Xu, Armand Nasser, Jiaru Cai, Linda Li, Kevin Vuong, and Es-han Wadhwa. Fake news detection with different models. *arXiv preprint arXiv:2003.04978*, 2020.
- [Wal21] Mason Walker. News consumption across social media in 2021, Sep 2021.
- [Wor] World Health Organization. Infodemic.