



People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
university of KASDI Merbah Ouargla



Faculty of New Information Technologies and Communication Department of Informatique

Thesis submitted in partial fulfillment of the requirements for the degree of

Master

In: Industrial Computing

By: **Hebbaz Yamina/ Bouhafis Rayan**

Subject

**Using machine learning techniques for
detecting Fake News**

Sustained before the jury composed of:

Mr. ABDERRAHIM .M.E.A	Prof	Ouargla university	President
Mrs. TOUMI .Ch	MCB	Ouargla university	Examiner
Mr. MERABTI .H	MCA	Ouargla university	Supervisor

Academic Year: 2022/2023

Dedication

Thank God

This enabled us to complete this step in our

academic journey The fruit of effort and

success, by His grace, I dedicate:

My great Mother, who never stop giving of herself in countless ways,

To my dear father, may God have mercy on him,

To my sisters, brothers, and all members of my

honorable family, My God protect them.

Yamina Hebbaz

Dedication

To my gift from God, and the great blessing in which I live, to those who are unmatched by anything in the universe, to whom God has commanded us to be righteous,
to those who have done so much, and who have given the irretrievable.

These words, my dear mother and father, I dedicate this effort to you, because you have been the best support for me throughout my academic career.

Loyal friends, who never ceased to help, assist and support me in the most difficult circumstances, wishing me success and prosperity.

To my loyal brothers and relatives my heart, respect and loyalty who stood by me, and their good wishes for success, support and encouragement, enabled me to pass stage of my life, I thank you very much, and a lot of respect.

Those who are happy with our success and saddened by our failure are dear professors.

I would like to dedicate these words to you as an expression of the great efforts you have made for me. You have the most beautiful rose-scented gift bouquets

Bouhafes rayan

Acknowledgments

In the name of Allah, the Most Gracious and the Most Merciful. All praises to Allah and His blessing for the completion of this thesis.

We thank God for all the opportunities, trials, and strength that have been seized We have to finish writing the note. We suffered a lot during this process, not only From an academic point of view, but also from a personal point of view.

First of all, we would like to extend our sincere thanks to our supervisor Prof. **Dr.Hocine Merabti** for his guidance, understanding and patience.

We would also like to thank all the people whose assistance marked a significant Milestone in the achievement

From the project wherever they are.

God bless everyone

Abstract

With the evolution of technology and the rise of social media, the proliferation of fake news has spread across a wide range of media platforms. This proliferation poses a significant challenge, exacerbated by the growing number of social media users and declining digital literacy. Existing solutions for detecting fake news have shortcomings, with the complexity of the task influenced by factors such as language type, news category and topic volatility. Machine Learning (ML) and Natural Language Processor (NLP) techniques offer viable means to address this issue by identifying patterns unique to fake news articles that are not present in authentic news content.

This study deals with a classification-based approach for the automation of fake news detection. Several methods were employed, including experimentation with different features (Count-Vectorizer, Tf-Idf Vectorizer, Bag-of-Words) and machine learning models (SVM, KNN, Random Forest, Naive Bayes, Decision Tree) to construct accurate detectors. Experiments were conducted on a real-world dataset, LIAR, to evaluate the performance of the models. The results showed that the SVM model using Tf-Idf Vectorizer features achieved the highest accuracy at 92%. These findings highlight the potential of Machine Learning models in the field of fake news detection, with a promising trajectory for further advancements in the future.

Keywords: Machine Learning Algorithms, Natural Language Processing, Fake News detection, Feature extraction, Classification.

ملخص

مع تطور التكنولوجيا وظهور وسائل التواصل الاجتماعي، تسارعت وتيرة انتشار الأخبار الكاذبة عبر مجموعة واسعة من المنصات الإعلامية. ويشكل هذا الانتشار تحديًا كبيرًا، ويزيد من حدته تزايد عدد مستخدمي وسائل التواصل الاجتماعي وتراجع المعرفة الرقمية. تنطوي الحلول الحالية للكشف عن الأخبار المزيفة على أوجه قصور، حيث يتأثر تعقيد المهمة بعوامل مثل نوع اللغة وفئة الأخبار وتنوع المواضيع. توفر تقنيات التعلم الآلي (ML) ومعالج اللغة الطبيعية (NLP) وسائل قابلة للتطبيق لمعالجة هذه المشكلة من خلال تحديد الأنماط الفريدة للمقالات الإخبارية المزيفة التي لا توجد في المحتوى الإخباري الحقيقي.

تتناول هذه الدراسة نهجًا قائمًا على التصنيف بهدف أتمتة الكشف عن الأخبار الكاذبة. تم استخدام العديد من الطرق، بما في ذلك تجربة ميزات مختلفة (Bag-of-words، Tf-Idf Vectorizer، Count-Vectorizer) ونماذج التعلم الآلي (SVM، KNN، Random Forest، Nive Bayes، و Decision Tree) لبناء كاشفات دقيقة. أُجريت التجارب على مجموعة بيانات من العالم الحقيقي، LIAR، لتقييم أداء النماذج. أظهرت النتائج أن نموذج SVM باستخدام ميزات Tf-Idf Vectorizer حقق أعلى دقة بنسبة 92%. تُبرز هذه النتائج إمكانات نماذج التعلم الآلي في مجال الكشف عن الأخبار الكاذبة، مع وجود مسار واعد لتحقيق المزيد من التقدم في المستقبل.

الكلمات المفتاحية: خوارزميات التعلم الآلي، معالجة اللغات الطبيعية، اكتشاف الأخبار المزيفة، استخراج الميزات، التصنيف.

Table of Contents

Abstract	I
ملخص	II
List of Figures	V
List of Tables	VI
Introduction	1
Chapter I Fake News	4
1. Introduction	5
2. Fake news	5
2.1. Definition.....	5
2.2. Fake news components	5
2.3. Fake news characteristics	6
2.4. How to detect a fake news	6
3. Analysis of News Content	7
3.1. Knowledge-based approaches	7
3.2. Machine Learning Approach.....	8
3.3. Hybrid approaches.....	9
4. Advantages and limitations of fake news detection	9
5. Fields of application for fake news	10
6. Conclusion	11
Chapter II MI and NLP for fake news detection	12
1. Introduction	13
2. Machine Learning:	13
2.1. Machine Learning Techniques	13
2.2. Natural Language Processing (NLP) :.....	15
2.3. Text classification process	15
3. Conclusion.....	26
Chapter III	27
Implementation results	27
and discussion	27
1. Introduction	28
2. Work environment and tools.....	28
2.1. Programming language.....	28
2.2. Code editor.....	28
2.3. Python libraries	29
3. System Architecture	30
3.1. Data preparation	31
3.2. Data splitting.....	37
3.3. Data representation.....	37
4. Results and discussion.....	39

4.1. Results of classifiers with and without hyper parameter tuning	40
4.2. Results using features combination	43
4.3. Comparison	45
5. Conclusion	47
General Conclusion	48
Bibliography	51

List of Figures

figure II 1:Text classification process	15
figure II 2:Graphical representation of the SVM algorithm.....	20
figure II 3:Graphical representation of the Random forests	21
figure II 4:Graphical representation of the Decision Tree	24
figure III- 1:The global Architecture of our system.....	30
figure III- 2:Example of importing a data package.....	31
figure III- 3:Accuracy of algorithms with properties for extraction using hyper parameter....	45
figure III- 4:accuracy of algorithms with properties for extraction using Default Initialization	46

List of Tables

Table III 1: Example of removing lower case	32
Table III 2: Example of lemmatization	33
Table III 3: Example of removing punctuation	33
Table III 4: Example of converting numbers to word.....	34
Table III 5: Example of expanding in tractions	34
Table III 6: Example of removing links	35
Table III 7: Example of removing multiple spaces.....	35
Table III 8: Example of removing stop words.....	36
Table III 9: Example of Replacing emojis with space	36
Table III 11: Performances of classifiers with and without Hyper parameter tuning.....	41
Table III 12: Hyper parameters of all algorithms	42
Table III 13: Accuracy of algorithms using features combination	43
Table III 14: Hyper-parameters of algorithms using features combination	44



Introduction

Introduction

The dissemination of false information, commonly known as fake news, has become a major problem in recent years. Fake news is defined as news that is intentionally false or misleading. It is often created to deceive people or to promote a particular agenda. The spread of fake news can have a negative impact on society and various domains by influencing individuals to make decisions based on inaccurate information.

There are a number of different techniques that can be used to detect fake news. One common approach is to use Machine Learning (ML) techniques to identify patterns in fake news articles [1]. Machine Learning techniques involve the extraction of relevant features from data, which are then used by classification models to make predictions or categorize the data. Machine learning algorithms can be trained on a dataset of known fake news articles and can then be used to classify new articles as either fake or real.

Another approach to fake news detection is to use Natural Language Processing (NLP) techniques. NLP techniques can be used to analyze the language used in an article to identify features that are associated with fake news. For example, NLP techniques can be used to identify articles that use sensationalized language, that make outlandish claims, or that are poorly sourced [2].

Problematic

The main problematic of this thesis is that there is no single, perfect method for detecting fake news. The problem is compounded by the fact that fake news is constantly evolving, as creators find new ways to deceive people. However, we believe that the machine learning approaches have the potential to be a valuable tool for detecting fake news. In the field of Machine Learning, features extraction plays a crucial role as it involves identifying and capturing relevant patterns and characteristics from data, which are subsequently utilized by classification models to make accurate predictions or categorize instances effectively. In this work, we aim to provide answers to the following questions: What is the difficulty of the detection task? Do we really need all the existing features, or should we focus on a smaller set of more representative features? Is there a trade-off between the discriminative power of the features and their robustness to model variations? Is there a clear-connection between the

General Introduction

features or models used and the type of fake news they can detect? We hope that our work will help to combat the spread of fake news and to make the internet a more reliable source of information.

Objectives

In this thesis, we will explore the use of ML and NLP techniques to detect fake news. We will first review the literature on fake news detection and discuss the different techniques that have been used. We will then present and implement some models to fake news detection. In particular, our purpose is to examine different methods in order to understand why certain techniques and models exhibit higher performance, emphasizing their advantages and limitations. This objective is accomplished mainly by means of experimentations on a known fake news articles dataset LIAR [3] involving various models, features, and pre- processing operations to improve accuracy.

We believe that the implemented models have the potential to be a valuable tool for detecting fake news. Our goal is therefore to offer an accurate and efficient model which can be easily applied to new articles. We hope that our work will help to combat the spread of fake news and to make the internet a more reliable source of information.

Thesis Organization The structure of this thesis consists of three chapters, which are organized as follows:

- In the initial chapter, the fundamental concepts and principles of fake news are addressed;
- The second chapter focuses on the application of Machine Learning models and Natural Language Processing techniques for the detection of fake news;
- The final chapter examines the results achieved by the implemented models



Chapter I

Fake News

1. Introduction

In recent years, several researchers have explored the field of fake news detection. Each of them has approached the problem from a unique perspective and utilized various methods. The objective of this chapter is to provide crucial and indispensable reminders that will help comprehend the fundamental terms used in this thesis. Initially, we will cover all aspects of fake news, including its characterization and components, followed by an examination of how to analyze it.

2. Fake news

2.1. Definition

Fake news refers to false or misleading information presented as news, often with the intention of deceiving the audience, manipulating public opinion, or generating revenue through clicks and shares. It can be spread through various channels, such as social media, websites, and even traditional media outlets. Although fake news has always been spread throughout history, the term “fake news” was first used in the 1890s when sensational reports in newspapers were common. Nevertheless, the term does not have a fixed definition and has been applied broadly to any type of false information. It’s also been used by high-profile people to apply to any news unfavorable to them [4].

2.2. Fake news components

Fake news incorporates various components, and here are some common typically found [5]:

- **Creator/Distributor:** those responsible for creating and disseminating fake news online can be individuals (human) or automated entities(non-human).
- **Targeted Audience:** fake news can target a wide range of individuals using various digital platforms, including social media users. This audience can encompass students, voters, parents, senior citizens, and others.
- **News Content:** The news content comprises both textual and multimedia elements, such as headlines, body text, and accompanying media. It also includes intangible aspects like the intended purpose, emotional tone, and underlying themes.
- **Social Context:** The social context refers to the environment in which news is shared

and circulated on the internet. Analyzing the social context involves studying user behavior and network dynamics, examining how online users interact with news content, and evaluating the ways in which news is shared online.

2.3. Fake news characteristics

Fake news often contains a number of common characteristics. These include:

- *Sensationalism*: Fake news stories are often designed to grab attention by being sensational or outrageous.
- *Click bait*: Fake news stories often use click bait headlines that are designed to make people click on them, even if the content of the story is not actually interesting or informative.
- *Misleading headlines*: Fake news stories often have misleading headlines that do not accurately reflect the content of the story.
- *Unsubstantiated claims*: Fake news stories often make unsubstantiated claims that are not supported by evidence.
- *Personal attacks*: Fake news stories often contain personal attacks on individuals or groups.
- *Extreme language*: Fake news stories often use extreme language that is designed to evoke strong emotions, such as anger, fear, or outrage.
- *Lack of author information*: Fake news stories often do not include author information, which makes it difficult to verify the accuracy of the information.
- *Poor grammar and spelling*: Fake news stories often have poor grammar and spelling, which can be a sign that the story is not credible.

2.4. How to detect a fake news

- **Number of comments**: The number of comments on a news article can be an indication of its engagement. However, it is important to note that fake news articles can also generate a lot of comments.
- **Sentiment of the comments**: The sentiment of the comments on a news article can be an indication of its veracity. For example, articles with a lot of negative comments are more likely to be fake.

3. Analysis of News Content

This section focuses on methods and techniques for analyzing the content of news articles to determine their authenticity and accuracy. It involves examining various aspects such as language, writing style, sources, and factual accuracy to determine the credibility of the news. Various approaches can be used for this purpose.

3.1. Knowledge-based approaches

Knowledge-based approaches utilize external sources of information and expert systems to verify the facts and the claims made in news articles. These approaches aim to cross-reference the information provided in the news with existing knowledge to identify inconsistencies or falsehoods [6].

Here are some of the external sources of information that can be used by knowledge-based approaches to fake news detection:

- **Databases:** Databases can store large amounts of information, such as historical facts, scientific data, and demographic statistics. This information can be used to verify the claims made in news articles.
- **Expert knowledge:** Experts in a particular field can provide valuable insights into the accuracy of news articles. For example, a medical expert can help to verify the claims made in a news article about a new medical study.
- **Ontologies:** Ontologies are formal representations of knowledge that can be used to reason about the relationships between different concepts. For example, an ontology can be used to reason about the relationship between a person, a place, and an event.

There are two main categories for knowledge-based methods which will be explained in the following sections:

A. Human Oriented Fact Checking

In this approach, human fact-checkers manually investigate the claims made in news articles by conducting research, consulting reliable sources, and verifying the information through various means. For example, organizations like Snopes (www.snopes.com), FactCheck.org (www.factcheck.org) and PolitiFact(<https://www.politifact.com/>) employ teams of fact-checkers who analyze news content and debunk false or misleading claims [7, 8].

The downside here is that these solutions can be time-consuming and expensive to develop

and maintain. They can also be limited by the quality and quantity of the external knowledge that is available [9].

B. Computational Oriented Fact Checking:

Computational-oriented fact-checking utilizes automated algorithms and computational methods to assess the credibility of news articles. These methods often involve natural language processing techniques, information retrieval, and statistical analysis to determine the likelihood of misinformation. Examples of such approaches include Claim Buster system (<https://idir.uta.edu/claimbuster/>) and Fact Mata system (<https://factmata.com>) [7, 10,11].

Computational oriented fact checking is a promising new approach to the problem of fake news. It has the potential to be more efficient and scalable than traditional fact checking methods, which are often limited by the time and resources available to journalists and fact-checkers.

However, computational oriented fact checking is still under development, and it has some limitations. For example, it can be difficult to develop models that are accurate and reliable, and it can be difficult to identify all of the sources of information that are relevant to a particular claim.

3.2. Machine Learning Approach

Based on patterns and features extracted from the news content, machine learning approaches employ algorithms that learn from data to detect patterns and make predictions. In the context of fake news detection, these models are trained on labeled datasets containing both fake and legitimate news articles [12, 13, 14, 15].

There are a number of different machine learning algorithms that can be used for fake news detection, including:

A. Classical Models

These are traditional machine learning algorithms that are trained on labelled datasets to classify news articles as real or fake based on features like language patterns, source credibility, and metadata. Example: Naive Bayes, Logistic Regression, Support Vector Machines (SVM) and Random Forests which are commonly used classical models for fake news detection.

B. Deep Learning:

Deep learning models, particularly neural networks, are used to develop more complex and sophisticated models that can learn from large amounts of data to detect fake news. These

models can learn complex representations from the text and capture intricate patterns. Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and transformer models like BERT and GPT have been employed for fake news detection tasks [16, 17,18].

The best machine learning algorithm for fake news detection depends on the specific dataset and the desired accuracy. In addition to machine learning algorithms, there are a number of other features that can be used to detect fake news, such as:

- **Stylistic features:** These features include the use of certain words or phrases, the length of sentences, and the overall tone of the article.

- **Content features:** These features include the factual accuracy of the article, the presence of bias, and the use of sensationalized language.

- **Social media features:** These features include the number of shares, likes, and comments on the article, as well as the sentiment of the comments.

By combining machine learning algorithms with other features or approaches, it is possible to create more accurate and reliable fake news detection models.

3.3. Hybrid approaches

Hybrid approaches combine multiple methods or techniques to enhance the accuracy and robustness of fake news detection systems. These approaches integrate knowledge-based methods with machine learning or combine different machine learning techniques to achieve better results. Some studies have proposed hybrid models that integrate both expert knowledge and machine learning algorithms to improve the overall performance of fake news detection systems [19, 20, 21]. As the problem of fake news continues to grow, it is likely that we will see more and more hybrid approaches being developed and used.

4. Advantages and limitations of fake news detection

Here are some advantages and limitations of fake news detection in general:

Advantages

- ✓ Protects public opinion: Detecting and preventing the spread of fake news helps maintain the integrity of public opinion and prevents the manipulation of people's beliefs and emotions.

- ✓ Preserves credibility: By identifying and removing fake news, media outlets and social

media platforms can maintain their credibility and trustworthiness, ensuring that users receive accurate and reliable information.

- ✓ Reduces harm: Fake news can have serious consequences, such as inciting violence, spreading fear, or damaging reputations. Detecting and stopping the spread of fake news can help minimize these negative effects.

- ✓ Encourages responsible journalism: The presence of effective fake news detection systems can encourage journalists and content creators to be more responsible and adhere to ethical standards, as they know that false information will likely be detected and removed.

Limitations

- ✓ False positives and negatives: No detection system is perfect, and there may be instances where legitimate news is flagged as fake or vice versa. This can lead to the suppression of valid information or the continued spread of misinformation.

- ✓ Difficulty in defining fake news: The line between fake news and biased or opinionated reporting can be blurry. It can be challenging to create a detection system that accurately distinguishes between the two without infringing on freedom of speech.

- ✓ Evolving tactics: As fake news detection systems improve, so do the tactics used by those who create and spread fake news. This can make it difficult for detection systems to keep up with the constantly changing landscape of misinformation.

- ✓ Potential for misuse: There is a risk that fake news detection systems could be misused by governments or other entities to suppress dissenting opinions or control the flow of information. This could lead to censorship and a lack of diverse perspectives in the public sphere.

In summary, while fake news detection systems offer several advantages in combating the spread of misinformation, they also face limitations and challenges that need to be addressed to ensure their effectiveness and prevent potential misuse.

5. Fields of application for fake news

Fake news can have a negative impact on individuals, society, and businesses. It is important to be aware of fake news and to be able to spot it. There are some of the fields where fake news is often applied:

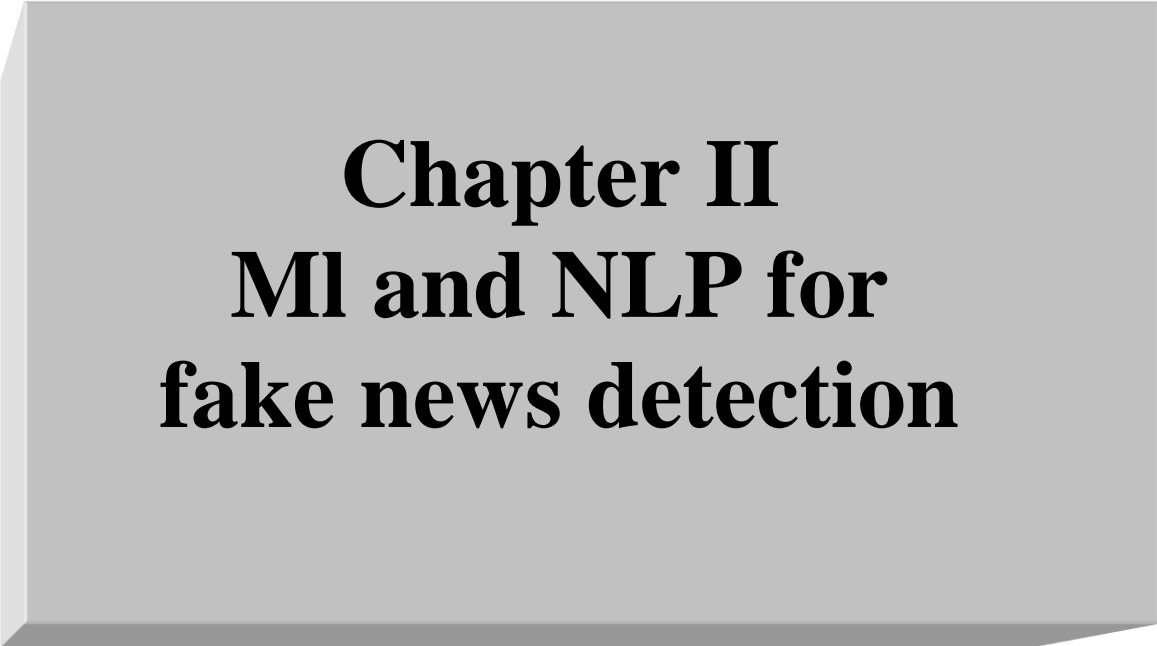
- **Politics:** Fake news is often used to influence elections and political campaigns. For example, fake news stories were spread during the 2016 US presidential election that claimed that Hillary Clinton was involved in a child sex ring. These stories were

widely shared on social media and may have influenced some voters to vote for Donald Trump.

- **Business:** Fake news is often used to damage the reputation of businesses or to promote the products or services of competitors. For example, fake news stories were spread about a company that claimed that its products were dangerous or that it was involved in illegal activities. These stories could lead to customers boycotting the company or to investors selling their shares.
- **Health:** Fake news is often used to spread misinformation about health and medical issues. For example, fake news stories have been spread that claim that vaccines cause autism or that certain foods can cure cancer. These stories can lead to people making harmful decisions about their health.
- **Crime:** Fake news is sometimes used to commit crimes, such as fraud or identity theft. For example, fake news stories have been spread that claim that people can get free money or that they can win a prize by providing their personal information. These stories can lead to people being scammed or having their identities stolen.
- **Society:** Fake news can also have a negative impact on society as a whole. It can lead to polarization, distrust, and violence. For example, fake news stories that were spread during the 2016 US presidential election may have contributed to the violence that erupted in some cities after the election.

6. Conclusion

The first chapter of our work serves as an introduction to the subject of our research, by introducing: the basic concepts of fake news, its components and characteristics. Furthermore, in the next chapter, we have explored various techniques and strategies employed for the detection of these fake news, focusing on the most significant ones that are relevant to our work.



Chapter II
MI and NLP for
fake news detection

1. Introduction

Machine Learning (ML) and Natural Language Processing (NLP) play pivotal roles in the detection of fake news. ML algorithms, such as classification, are employed to analyze patterns and distinguish between genuine and deceptive information. NLP techniques enable the extraction of meaningful insights from textual data, allowing algorithms to discern linguistic cues indicative of misinformation. By leveraging ML models trained on large datasets of both authentic and fabricated content, fake news detection systems can identify suspicious articles based on linguistic features, and contextual information. The synergy between ML and NLP empowers the development of robust and scalable solutions to combat the proliferation of fake news across various digital platforms.

Given this in consideration, the primary focus of this chapter will be to introduce the concept of Machine Learning in a broad sense, along with an exploration of their techniques employed for the categorization of fake news.

2. Machine Learning:

Machine learning ML is a branch of artificial intelligence that enables systems to learn and understand through algorithms. It revolves around the concept of training algorithms with Data, allowing computers to make predictions and solve specific tasks without explicit Programming.

ML plays a crucial role in the fight against fake news by offering an automated and scalable approach to its detection. ML algorithms analyze the text of the news article, looking for patterns in writing style, word choice, and usage of specific phrases commonly associated with fake news. Beyond the text itself, the algorithms can examine the social media context of the news, such as the source of the information, user engagement metrics (likes, shares), and the overall sentiment surrounding the article. In some cases, the analysis might even extend to the network of accounts sharing the news, identifying patterns of bots or coordinated campaigns spreading misinformation. There are several categories of ML that are used in the detection of fake news

2.1. Machine Learning Techniques

A. Supervised learning:

It entails providing computers with data and their corresponding desired outcomes, enabling them to make predictions of new input data [22].

✓ **Advantages:**

- Supervised learning utilizes labeled datasets, providing precise information about object classes.
- These algorithms excel at predicting outcomes based on previous experiences.

✓ **Limitations:**

- Complex tasks may prove challenging for these algorithms.
- Training the algorithm can be time-consuming due to the high computational requirements.

B. Unsupervised Learning:

It entails feeding computers with data alone, without external guidance, and tasting them with identifying meaningful patterns or structures, often through clustering techniques [23,24].

✓ **Advantages:**

- Unsupervised learning algorithms can handle complex tasks that are difficult for supervised Learning algorithms since they can work with unlabeled datasets.
- Unsupervised algorithms are advantageous for various tasks as obtaining unlabeled datasets is often easier compared to acquiring labeled datasets.

✓ **Limitations:**

- The output of unsupervised algorithms may be less accurate because the datasets are not Labeled, and the algorithms are not trained with exact output information.
- Working with unsupervised learning is more challenging as it involves working with unlabeled datasets that do not have direct mappings to specific outputs.

C. Reinforcement learning:

It involves emulating the behavior of an agent, wherein a machine interacts with its environment, learns from the consequences of its actions, and iteratively improves its behaviors to maximize rewards [25].

✓ **Advantages:**

- Reinforcement learning is effective in solving complex real-world problems that are challenging for conventional techniques.
- It enables the achievement of long-term results, as the agent learns to optimize its behavior overtime.

✓ **Limitations:**

- RL algorithms may not be the best choice for simple problems that can be solved using simpler methods.

- Implementing RL algorithms requires significant amounts of data and computational resources.

2.2. Natural Language Processing (NLP) :

Natural Language Processing (NLP) is a subfield of Artificial Intelligence that equips machines with the ability to understand and process human language. In the fight against fake news, NLP plays a crucial role by enabling the analysis and understanding of textual content. It also serves as a foundational layer for various techniques used in machine learning models for detection. Text Preprocessing, Feature Extraction, Named Entity Recognition (NER), and Semantic Analysis are some key NLP techniques used in fake news detection. In the next subsection, we will explain some of these techniques that help to achieve this detection purpose.

By leveraging NLP techniques for feature extraction and analysis, ML models can be trained to identify patterns and characteristics often associated with fake news. However, it is crucial to remember that NLP is just one piece of the puzzle. Combining NLP with human expertise, fact-checking, and critical thinking remains essential for a comprehensive approach to tackling the challenge of fake news.

2.3. Text classification process

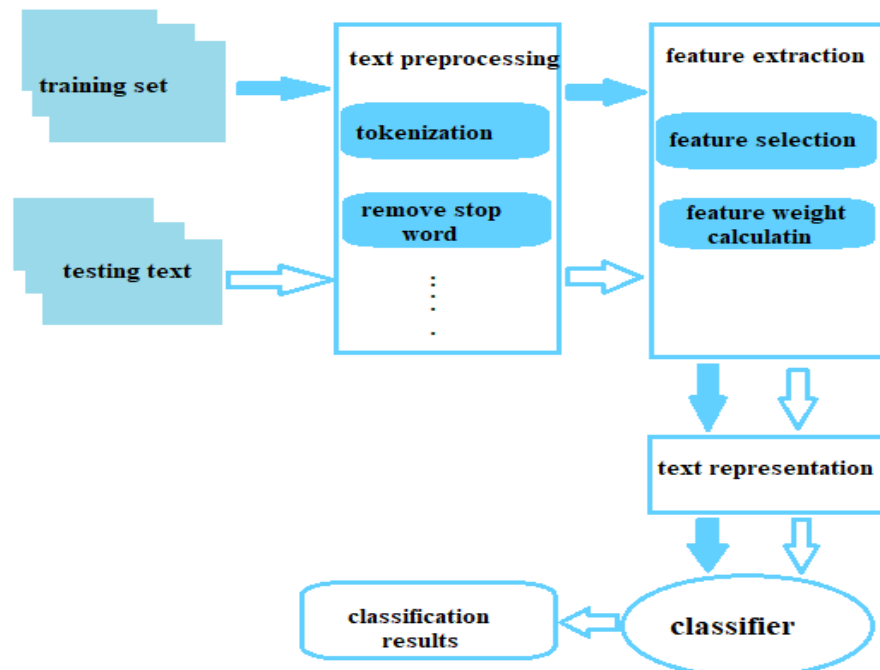


figure II 1:Text classification process

2.3.1. Pre-processing

Pre-processing is a crucial step in text classification that involves transforming raw text data into a format suitable for machine learning algorithms. The specific pre-processing steps applied may vary depending on the characteristics of the text data and the requirements of the classification task. Here are some common pre-processing steps in text classification:

- ✓ **Tokenization:** Splitting the text into individual words or tokens. This step essentially breaks down the text into its basic units, which can be analyzed further.
- ✓ **Lowercasing:** Converting all text to lowercase. This ensures that words like "Hello" and "hello" are treated the same way and don't create duplicate features.
- ✓ **Removing punctuation:** Stripping out any punctuation marks from the text. Punctuation often doesn't add significant meaning in text classification tasks and can be safely removed.
- ✓ **Removing stop words:** Stop words are common words that often occur frequently in text but carry little semantic meaning (e.g., "the", "and", "is"). Removing these words can reduce the dimensionality of the feature space and improve model efficiency.
- ✓ **Stemming and Lemmatization:** Stemming and lemmatization are techniques used to reduce words to their root or base form. For example, "running", "runs", and "ran" might all be reduced to the root word "run". Stemming is more heuristic-based and may result in non-words, while lemmatization typically uses a vocabulary and morphological analysis to return actual words.
- ✓ **Handling contractions and abbreviations:** Expanding contractions (e.g., "can't" to "cannot") and abbreviations (e.g., "USB" to "Universal Serial Bus") can help standardize the text and improve the model's ability to generalize.
- ✓ **Normalization:** Normalizing numbers, dates, URLs, and other special characters to a standard format. This ensures consistent treatment of these elements across the dataset.
- ✓ **Handling rare words and spelling corrections:** Dealing with rare or misspelled words can involve techniques such as replacing them with a special token, correcting spelling errors, or removing them altogether.

The specific pre-processing steps used in our case will be discussed in detail in Chapter 3.

2.3.2. Feature Extraction and vectorization:

Feature extraction and vectorization are essential pre-processing steps in text classification, as they transform raw text data into a format that machine learning algorithms can understand and learn from. The choice of feature extraction and vectorization techniques depends on the characteristics of the text data and the specific requirements of the classification task.

In feature extraction, we convert raw text data into numerical or categorical features that can be used as input for machine learning algorithms. This step involves capturing the important characteristics or patterns in the text data. Techniques for feature extraction include:

✓ **Bag-of-Words (BoW):** This technique represents a text document as a "bag" of its individual words, disregarding grammar and word order. Each word is assigned a unique index, and the vector representation of a document is created by counting the occurrences of each word in that document. The resulting vector is typically a high-dimensional sparse vector.

✓ **The Term Frequency-Inverse Document Frequency (TF-IDF):** is a technique for Evaluating the relevance of a document to a term, taking into account two factors: the Frequency of the word in the document (TF) and the number of documents containing This word (IDF) in the studied corpus.

The formula to calculate the TF-IDF of a term in a document is given:

$$TF-IDF(t, d) = TF(t, d) \times IDF(t) \quad (2.1)$$

Where:

TF (t, d): represents the frequency of term (t) in document (d)

IDF (t): is calculated as follows:

$$IDF(t) = \log (N / df (t, d)) \quad (2.2)$$

Where:

N: is the total number of documents in the corpus

DF (t, d): is the number of documents containing the term (t).

Using this formula, the TF-IDF assigns a weight to each term in the document, taking into account both its frequency in the document and its rarity in the entire corpus. Thus, terms that appear frequently in a specific document but rarely in the entire corpus will have a high TF-IDF Score, indicating their relative importance in that document.

TF-IDF is widely used for text vectorization, as it allows documents to be represented as digital Vectors, where each dimension corresponds to a term and the value of the dimension is the TF IDF score of the term in the document. This vector representation allows machine learning Models to process text data efficiently [26].

✓ **Count Vectorizer:** The Count Vectorizer works by tokenizing the text documents into individual words or tokens and then counting the occurrence of each word in each

document. It creates a matrix representation where each row represents a document, and each column represents a unique word in the corpus. The cell values indicate the frequency of the word in the respective document.

- ✓ **Word Embeddings:** Represent words as dense, lower-dimensional vectors in a continuous vector space. Popular techniques include Word2Vec, GloVe, and fast Text.
- ✓ **Topic Modeling:** Identifies latent topics within a collection of documents and represents documents in terms of their distribution over these topics.
- ✓ **Character-level Features:** Extracts features based on character n-grams, which can capture morphological or syntactical patterns.
- ✓ **Part-of-Speech (POS) Tagging:** Identifies the grammatical categories (e.g., noun, verb, adjective) of words in a sentence.
- ✓ **Named Entity Recognition (NER):** Identifies and classifies named entities (e.g., person names, organization names) in text data.

Once features are extracted, they need to be converted into a numerical vector format suitable for input to machine learning algorithms. This process is known as vectorization. Common techniques for vectorization include:

- ✓ **One-Hot Encoding:** Converts categorical features into binary vectors where each feature is represented by a binary indicator variable.
- ✓ **Count Vectorization:** Represents text data as a matrix where each row corresponds to a document and each column corresponds to a unique word, with cell values representing the frequency of each word in the corresponding document.
- ✓ **TF-IDF Vectorization:** Converts text data into numerical vectors using TF-IDF values instead of raw term frequencies.
- ✓ **Word Embedding Lookup:** Converts each word in a document to its corresponding word embedding vector representation.
- ✓ **Dimensionality Reduction:** Techniques such as Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) can be applied to reduce the dimensionality of feature vectors while preserving important information.

2.3.3. Algorithms

Different supervised learning classifiers are used for text classification. Here are some commonly used basic models:

a) Naïve Bayse

Naive Bayes is a classification method that uses Bayes' theorem to calculate conditional probabilities. In text classification, Naive Bayes is applied as follows: it aims to determine the classification that maximizes the probability of observing the words within a document. During The training phase, the classifier calculates the probabilities of a new document belonging to a Specific category based on the proportion of training documents within that category. Additionally, it calculates the probability of a given word appearing in a given text that the text Belongs to a certain category. When classifying a new document, the method calculates the Probabilities of it belonging to each category using Bayes' rule.

$$P(A | B) = \frac{P(B|A) \times P(A)}{P(B)} \quad (2.3)$$

Where:

- $P(A|B)$ is the posterior probability of hypothesis A given evidence B . This is the probability we are interested in determining.
- $P(B|A)$ is the likelihood of evidence B given that hypothesis A is true.
- $P(A)$ is the prior probability of hypothesis A , which is our belief about the probability of A before considering evidence B .
- $P(B)$ is the total probability of evidence B , often called the marginal likelihood or the normalizing constant. It represents the probability of observing evidence B irrespective of the truth of hypothesis A .

Bayes' rule, also known as Bayes' theorem, can be expressed using the following formula:

$$P(y | x_1, x_2, \dots, x_n) = \frac{P(y) \times P(x_1|y) \times P(x_2|y) \times \dots \times P(x_n|y)}{P(x_1) \times P(x_2) \times \dots \times P(x_n)} \quad (2.4)$$

Where:

- $P(y|x_1, x_2, \dots, x_n)$ is the posterior probability of class y given the features x_1, x_2, \dots, x_n .
- $P(y)$ is the prior probability of class y .
- $P(x_i|y)$ is the likelihood of feature x_i given class y .
- $P(x_i)$ is the probability of feature x_i .
- x_1, x_2, \dots, x_n are the features.

✓ **Advantage:**

- Classification using Naive Bayes Classifier (NBC) can be performed even with small datasets, making it suitable when limited data is available for training.
- NBC does not require a large volume of data during the learning phase, which can be beneficial in situations with limited data availability.

✓ **Limitations:**

- In cases where there is a high correlation between features, particularly in long documents with a rich vocabulary that promotes dependencies between descriptors, NBC may yield poor performance.
- NBC's performance may suffer when there are strong dependencies or correlations between features, leading to less accurate classifications.

b) Support Vector Machines

Support Vector Machines (SVM) are classification algorithms designed to find an optimal classifier that effectively separates data points and maximizes the margin between two classes. This classifier is represented by a linear hyper plane. The hyper plane divides the data points into two sets. The data points that are closest to the hyper plane, and crucial for determining its position, are known as support vectors.

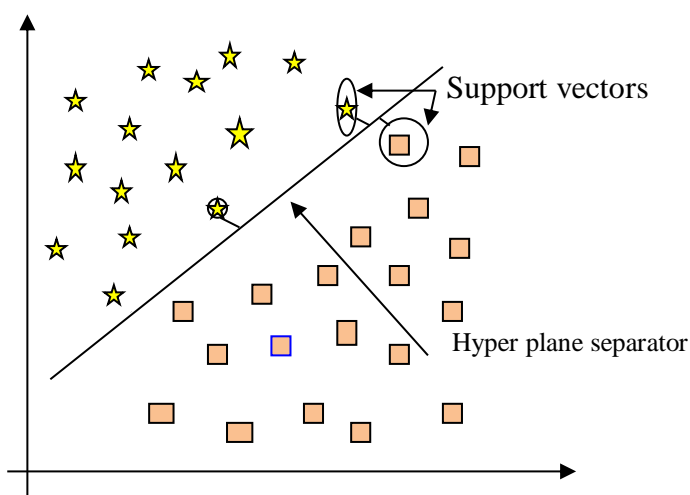


figure II 2: Graphical representation of the SVM algorithm

✓ **Advantage:**

- SVMs perform well even when limited prior knowledge about the data is available.
- SVMs scale relatively well to high-dimensional data, making them applicable to large datasets

✓ **Limitations:**

- SVMs can have long training times, especially for large datasets, which can be a drawback in time-sensitive applications.
- The final model, variable weights, and individual impact in SVMs can be challenging to Understand and interpret, limiting the transparency of the model

c) Random Forests

Random Forests is an implementation of decision tree-based algorithms that enables modelling of outcomes based on previous choices along different branches. By considering multiple decision trees, it aims to make the optimal decision based on the subsequent results. This Approach can be regarded as a form of anticipation, where the collective predictions of multiple trees contribute to a more comprehensive and reliable outcome [27].

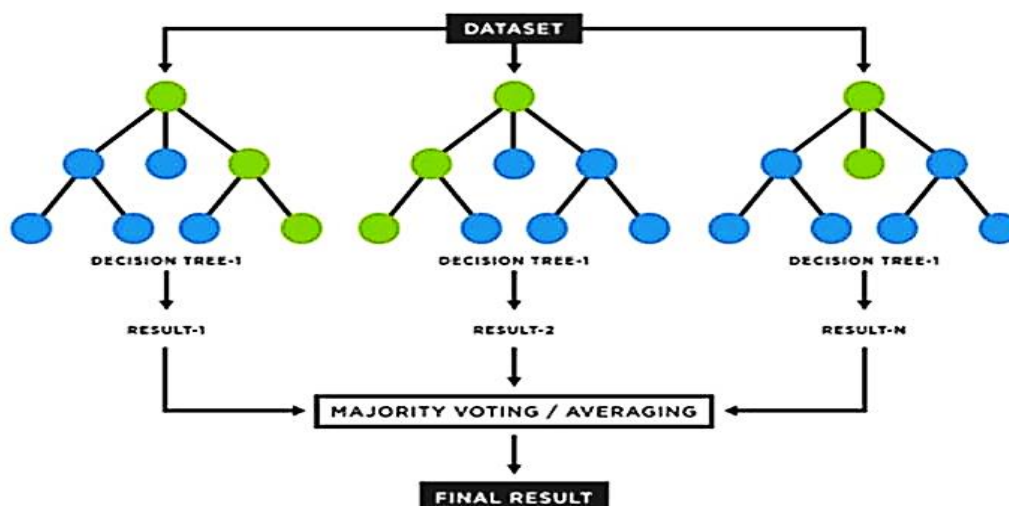


figure II 3:Graphical representation of the Random forests [28]

✓ **Advantage:**

- *High Accuracy:* Random Forest typically yields high prediction accuracy due to its ensemble nature, which combines multiple decision trees. This ensemble approach helps mitigate overfitting and improve generalization performance.
- *Robust to Overfitting:* Random Forest is less prone to overfitting compared to individual decision trees. By aggregating predictions from multiple trees and using techniques like bagging, it achieves better generalization on unseen data.
- *Feature Importance Estimation:* Random Forest provides a measure of feature importance, aiding in feature selection and understanding the underlying relationships in the data. This helps in identifying the most relevant features for prediction.

✓ **Limitations:**

- *Black Box Model:* Random Forest is often considered a black box model, making it less interpretable compared to simpler models like linear regression. Understanding the internal workings of individual trees within the ensemble can be challenging.
- *Memory and Computational Resources:* Training and tuning Random Forest models can be computationally expensive, especially as the number of trees and dataset size increase. This requires significant memory and computational resources.
- *Biased Towards Majority Classes:* Random Forest tends to be biased towards majority classes in imbalanced datasets, potentially leading to less accurate predictions for minority classes. Techniques like class weights or resampling may be needed to address this bias.

d) KNN (k-Nearest Neighbors)

KNN (k-Nearest Neighbors) is a classification algorithm that assigns new data points to classes based on the majority of their neighboring data points. In the k-Nearest Neighbors (KNN) algorithm, distances between data points are crucial for determining the "closeness" or similarity between instances. Several distance metrics can be used in KNN, but the most commonly used ones include:

- **Euclidean Distance:** This is the most common distance metric used in KNN. It calculates the straight-line distance between two points in Euclidean space. For two

points $P=(p_1,p_2,\dots,p_n)$ and $Q=(q_1,q_2,\dots,q_n)$ in an n -dimensional space, the Euclidean distance d is given by:

$$d(P,Q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2} \quad (2.5)$$

- **Manhattan Distance (Taxicab or City Block Distance):** This metric calculates the distance between two points by summing the absolute differences of their coordinates. For two points $P=(p_1,p_2,\dots,p_n)$ and $Q=(q_1,q_2,\dots,q_n)$, the Manhattan distance d is given by:

$$d(P,Q) = \sum_{i=1}^n |p_i - q_i| \quad (2.6)$$

- **Minkowski Distance:** This is a generalization of both Euclidean and Manhattan distances. It calculates the distance between two points using the p -th root of the sum of the absolute differences raised to the power of p . For two points $P=(p_1,p_2,\dots,p_n)$ and $Q=(q_1,q_2,\dots,q_n)$, the Minkowski distance d with parameter p is given by:

$$d(P,Q) = \left(\sum_{i=1}^n |p_i - q_i|^p \right)^{\frac{1}{p}} \quad (2.7)$$

When $p=2$, it reduces to the Euclidean distance, and when $p=1$, it reduces to the Manhattan distance.

✓ **Advantage:**

- *Simple Implementation:* KNN is straightforward to understand and implement, making it a good choice for beginners and for quick prototyping of classification tasks.
- *No Training Phase:* KNN is a lazy learner, meaning it doesn't explicitly train a model during the training phase. Instead, it stores the entire training dataset and makes predictions based on the closest instances during the testing phase. This makes the training process very fast.
- *Non-parametric Nature:* KNN is a non-parametric algorithm, meaning it makes no assumptions about the underlying distribution of the data. It can perform well on datasets where the decision boundary is highly irregular or nonlinear.

✓ **Limitations:**

- Computational Complexity: KNN can be computationally expensive, especially with large datasets, as it requires calculating distances between the query instance and all Training instances for each prediction.

- Choosing an Optimal K: Selecting the appropriate value of K (the number of neighbors to consider) can be challenging and requires domain knowledge or tuning.

✓ **Decision tree**

A decision tree is a valuable tool used for classification problems, employing a structure resembling a flow chart. Each internal node in the decision tree represents a condition or "test" Based on an attribute, and the tree branches out accordingly. The leaf nodes of the tree contain Class labels, which are determined after evaluating all the attributes. The path from the root to a Leaf node represents a classification rule.

One remarkable aspect of decision trees is their ability to handle both categorical and dependent Variables. They excel at identifying the most influential variables and effectively depicting their Relationships. Decision trees play a significant role in generating new variables and features, aiding data exploration, and making efficient predictions for the target variable.

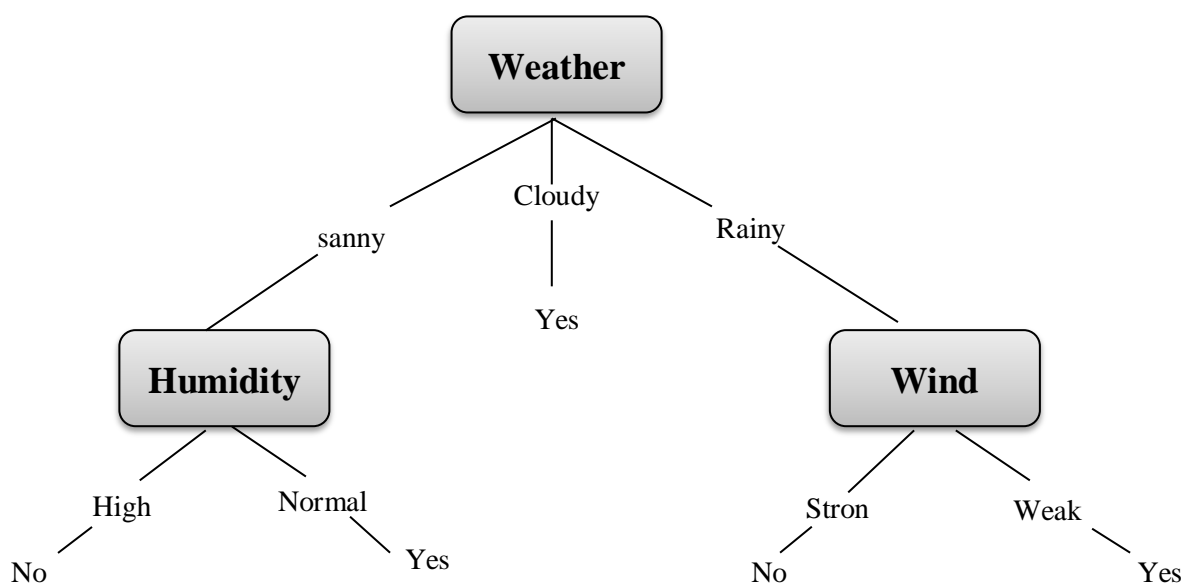


figure II 4:Graphical representation of the Decision Tree

✓ Advantage:

- *Interpretability:* Decision Trees are highly interpretable models, making them easy to understand and explain. Decision rules inferred from the tree structure can provide insights into the decision-making process.
- *No Assumptions about Data Distribution:* Decision Trees make no assumptions about the distribution of the data or the relationships between features, unlike parametric models such as linear regression. They can handle both numerical and categorical data without the need for extensive pre-processing.
- *Handles Non-linear Relationships:* Decision Trees can capture non-linear relationships between features and the target variable. They partition the feature space into rectangular regions, allowing them to model complex decision boundaries.

✓ Limitations:

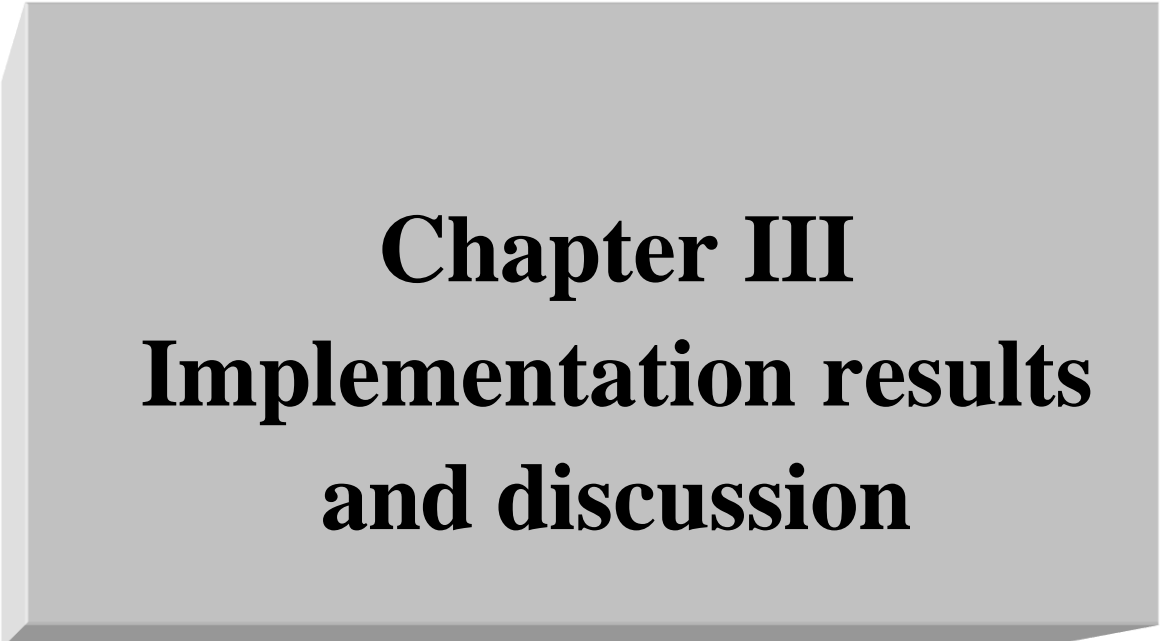
•-*Interpretability:* Decision Trees are highly interpretable models, making them easy to understand and explain. Decision rules inferred from the tree structure can provide insights into the decision-making process.

- *No Assumptions about Data Distribution:* Decision Trees make no assumptions about the distribution of the data or the relationships between features, unlike parametric models such as linear regression. They can handle both numerical and categorical data without the need for extensive pre-processing.
- *Handles Non-linear Relationships:* Decision Trees can capture non-linear relationships between features and the target variable. They partition the feature space into rectangular regions, allowing them to model complex decision boundaries.

3. Conclusion

In this chapter, we have introduced the fundamental principles, methodologies, and strategies for detecting and identifying fake news. Initially, we outlined the key phases required for constructing an automated system for detecting fake news. Furthermore, we thoroughly examined each phase, scrutinizing and delineating the various methods and techniques devised and employed.

In the next chapter, we delineate, in a sequential manner, the approaches we have implemented for this categorization task with the aim of enhancing and achieving higher precision.



Chapter III
Implementation results
and discussion

1. Introduction

The aim of our work is to explore the world of machine learning and natural language processing by building a system for detecting and classifying fake news. The various components of the architecture of our system are presented in this chapter.

2. Work environment and tools

2.1. Programming language

Python is a high-level, interpreted programming language with dynamic semantics. It is widely Popular among a large community of developers and programmers. Python is known for its simplicity and ease of learning, which contributes to reducing the cost of code maintenance. The extensive collection of libraries, or packages, in Python, promotes code modularity and reusability. Moreover, for most platforms, Python and its libraries are freely available, both in Source code and precompiled binaries, and can be redistributed without any charge.

2.2. Code editor

To build our system, we used the code editor PyCharm which is an Integrated Development Environment (IDE) specifically designed for Python development. It provides a comprehensive set of features and tools to enhance your coding experience, such as intelligent code completion, syntax highlighting, debugging capabilities, version control integration, and more. PyCharm Offers both free and paid versions, with the paid version offering additional advanced features and support.

2.3. Python libraries

Python has a rich ecosystem of libraries designed specifically for machine learning. Here are some popular Python libraries that we used in our work:

- **NumPy:** Provides the foundation for numerical computations essential for machine learning algorithms.
- **Pandas:** Offers data structures and tools for data manipulation and analysis, crucial for preparing and cleaning your dataset.
- **NLTK:** Natural Language Toolkit (NLTK) provides various functionalities for natural language processing tasks.
- **Re:** The re module is used for regular expression matching operations.
- **String:** This library provides various string manipulation functions.
- **Term color:** This library is used for printing colored text in the terminal.
- **Scikit-learn:** is a machine learning library that provides tools for data preprocessing, model selection, and evaluation.
- **Matplotlib :** It is a plotting library used for creating visualizations.
- **Seaborn:** Seaborn is a statistical data visualization library based on Matplotlib.

3. System Architecture

The different stages of our system can be summarized by the following diagram:

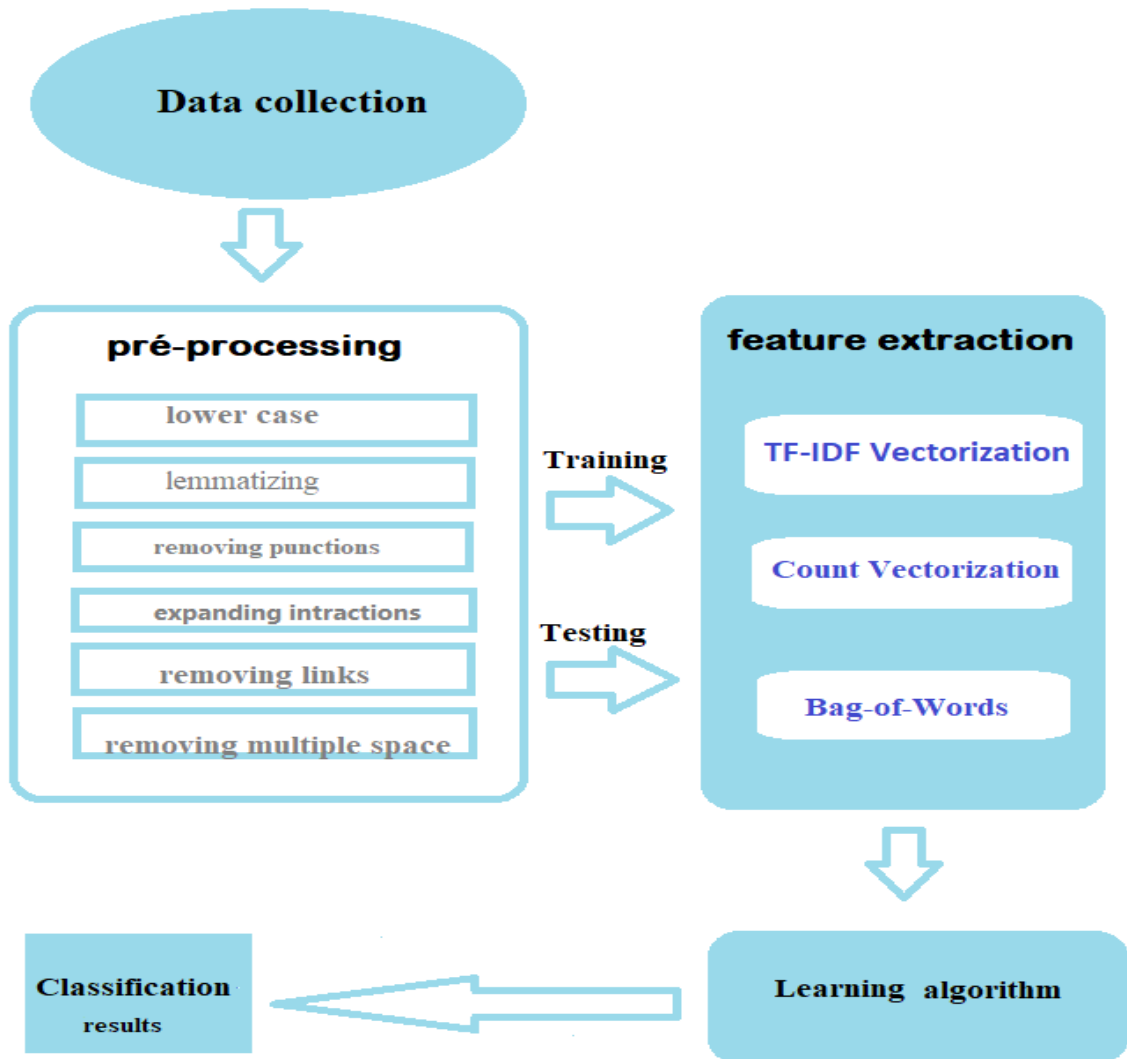


figure III- 1: The global Architecture of our system

3.1. Data preparation

A. Data collection

To assess the effectiveness of current evolving methods for categorizing fake news, we used a dataset named LIAR consisting of original and manufactured articles [3]. This dataset of 7,796 registered articles presents summary data pulled from a variety of contexts such as radio and television interviews, press releases, campaign speeches, and more.

Each statement in the dataset is accompanied by annotations indicating its correctness, title, and context [29].

	A	B	C	D	E
1		title	text	label	
2	8476	You Can Smell Hillary's	Daniel Greenfield, a	0	
3	10294	Watch The Exact Moment I	Google Pinterest Digg	0	
4	3608	Kerry to go to Paris in gest	U.S. Secretary of State	1	
5	10142	Bernie supporters on Twiti	" Kaydee King	0	
6	875	The Battle of New York: W	It's primary day in New	1	
7	6903	Tehran, USA		0	
8	7341	Girl Horrified At What She	Share This Baylee	0	
9	95	"Britain's Schindler"	A Czech stockbroker who	1	
10	4869	Fact check: Trump and Clin	Hillary Clinton and	1	
11	2909	Iran reportedly makes new	Iranian negotiators	1	
12	1357	With all three Clintons in I	CEDAR RAPIDS, Iowa "	1	
13	988	Donald Trump's Shocki	Donald Trump's	1	
14	7041	Strong Solar Storm, Tech R	Click Here To Learn	0	
15	7623	10 Ways America Is Prepar	October 31, 2016 at 4:52	0	
16	1571	Trump takes on Cruz, but I	Killing Obama administra	1	
17	4739	How women lead differen	As more women move	1	
18	7737	Shocking! Michele Obama	Shocking! Michele	0	
19	8716	Hillary Clinton in HUGE Tro	0	0	
20	3304	What's in that Iran bill that	Washington (CNN) For	1	
21	3078	The 1 chart that explains e	While paging through	1	
22	2517	The slippery slope to Trum	With little fanfare this	1	
23	10348	Episode #160 " SUNDAY	November 13, 2016 By	0	
24	778	Hillary Clinton Makes A Bi	Hillary Clinton told a	1	
25	3300	New Senate majority lead	Mitch McConnell has an	1	

figure III- 2:Example of importing a data package

B. Data pre-processing

Data pre-processing is a crucial step in fake news detection, as it helps to clean and prepare the data for analysis and model training. Here are some common pre-processing techniques used in our system:

- **Converting text to lowercase:** By converting text to lowercase, you ensure that words with different cases are treated as the same word, which helps in standardizing the text and reducing the vocabulary size. This preprocessing step is commonly performed before tokenization and other NLP tasks.

Table III 1: Example of removing lower case

Code	<pre>Text = "Hello, World!" lowercase_text = text.lower() print(lowercase_text)</pre>
Results	hello, world!

- **Tokenization:** is used to tokenize the input text into a list of words. Each word in the text becomes a separate token.

Original Sentence:

Tokenization is an important step in natural language processing.

Tokens:

['Tokenization', 'is', 'an', 'important', 'step', 'in', 'natural', 'language', 'processing', '.']

- **Lemmatization:** This technique helps in reducing word variations and standardizing the data. It converts words to their dictionary or canonical form.

Table III 2: Example of lemmatization

Code	<pre>import nltk from nltk.stem import WordNetLemmatizer from nltk.tokenize import word_tokenize nltk.download('wordnet') lemmatizer = WordNetLemmatizer() text = "The cats are running and jumping in the garden." tokens = word_tokenize(text) lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens] print(lemmatized_tokens)</pre>
Result	['The', 'cat', 'are', 'running', 'and', 'jumping', 'in', 'the', 'garden', '.']

- **Removing punctuation marks:** punctuation marks such as periods, commas, question marks, and exclamation marks often don't carry significant meaning and can add noise to the text data. Removing punctuation marks helps simplify the text and ensures that the subsequent NLP tasks focus on the essential content of the text.

Table III 3: Example of removing punctuation

Code	<pre>import string Text = "Hello, World!" punctuation_removed_text = text.translate (str.maketrans ("", "", string.punctuation)) print (punctuation_removed_text)</pre>
Result	Hello World

- **Converting numbers to words:** Converting numbers to words is a preprocessing step that involves replacing numerical digits with their corresponding word representations in text data. This transformation can be useful in natural language processing (NLP) tasks where the numerical values are not relevant or need to be treated as words rather than numerical values.

Table III 4: Example of converting numbers to word

Code	<pre>import inflect p = inflect.engine() numbers = [123, 4567, 89012, 345678] words = [p.number_to_words(num) for num in numbers] for num, word in zip(numbers, words): print(f"{num}: {word}")</pre>
Result	<pre>123: one hundred and twenty-three 4567: four thousand, five hundred and sixty-seven 89012: eighty-nine thousand and twelve 345678: three hundred and forty-five thousand, six hundred and seventy-eight</pre>

- **Expanding in tractions:** refers to providing the full form or meaning of an abbreviated word or phrase.

Table III 5: Example of expanding in tractions

Code	<pre>def expand_abbreviations(text): abbreviations = { "btw": "by the way", "lol": "laugh out loud", "omg": "oh my god", "idk": "I don't know", "jk": "just kidding" } words = text.split() expanded_words = [abbreviations.get(word.lower(),word) for word in words] expanded_text = " ".join(expanded_words) return expanded_text abbreviated_text = "btw, idk what jk means. lol!" expanded_text = expand_abbreviations(abbreviated_text) print(expanded_text)</pre>
Result	<pre>by the way, I don't know what just kidding means. laugh out loud!</pre>

- **Removing links:** Removing links from text data is a common preprocessing step in NLP tasks, especially when dealing with web content or social media data. Links often do not contribute to the semantics of the text and can introduce noise or distractions in the analysis.

Table III 6: Example of removing links

Code	<pre> import re def remove_links (text): url_pattern= re.compile (r'https?://\S+ www\.\S+') text_without_links = url_pattern.sub("", text) return text_without_links example_text = "Check out this website: https://example.com for more information." text_without_links = remove_links(example_text) print("Text without links:") print(text_without_links) </pre>
Result	<p>Text without links:</p> <p>Check out this website: for more information.</p>

- **Removing multiple spaces:** by removing multiple spaces, ensure that the text data is normalized and consistent, which can facilitate subsequent analysis or feature extraction steps.

Table III 7: Example of removing multiple spaces

Code	<pre> import re def remove_multiple_spaces(text): text = re.sub(r'\s+', ' ', text) return text.strip() text_with_multiple_spaces = "This is an example with multiple spaces." cleaned_text = remove_multiple_spaces(text_with_multiple_spaces) print("Cleaned text:", cleaned_text) </pre>
Result	<p>Cleaned text: This is an example with multiple spaces.</p>

- **Removing stop words:** it used to eliminate common words that do not carry significant meaning or contribute to the overall context. It can help reduce noise and improve the efficiency of text analysis.

Table III 8: Example of removing stop words

Code	<pre> stop_words = ["i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "he", "she", "her", "hers", "herself", "it", "they", "them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this", "that", "these", "now"] def remove_stop_words(sentence): words = sentence.split() filtered_words = [word for word in words if word.lower() not in stop_words] return ' '.join(filtered_words) sentence = "This is a simple example sentence with some stop words that need to be removed." filtered_sentence = remove_stop_words(sentence) print(filtered_sentence) </pre>
Result	simple example sentence stop words need removed.

- **Removing the emoji:** Emojis may not always convey meaningful information or may introduce noise in the text analysis, so we've replaced them with spaces.

Table III 9: Example of Replacing emojis with space

Code	<pre> import re emojis = [":)", ":(", ":D", "<3", ":P"] text = "I am feeling happy today! :) This is a great day. <3" for emoji in emojis: text = text.replace(emoji, " ") print (text) </pre>
Result	I am feeling happy today! This is a great day.

3.2. Data splitting

During this step, the dataset is divided into two subsets: one for training and the other for Testing. To achieve this, we suggest a split of 75% - 25%. This implies that 75% of the dataset will be utilized for training the model, while the remaining 25% will be set aside for testing and evaluation purposes.

3.3. Data representation

The data representation plays a crucial role in Capturing relevant information and enabling effective analysis. Several data representation techniques can be employed, including Bag-of-Words (BoW), Count Vectorization, and TF-IDF Vectorization, which We have explained in the following:

1. TF-IDF Vectorization

TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is a technique used to Represent text data by assigning weights to words based on their frequency in a document and Their occurrence across the corpus. It provides a more nuanced representation of the importance of words in a document.

Example:

Text 1: "I love to eat pizza." Text 2: "I love to eat burgers."

- ✓ **Step 1:** Create a Vocabulary

The only words are: ["I", "love", "to", "eat", "pizza", "burgers"]

- ✓ **Step 2:** Calculate Term Frequency (TF)

Text 1 TF: [1, 1, 1, 1, 1, 0]

Text 2 TF: [1, 1, 1, 1, 0, 1]

- ✓ **Step 3:** Calculate Inverse Document Frequency (IDF)

I DF: [0, 0, 0, 0, $\log(2/2)$, $\log(2/2)$] (assuming a corpus of two sentences)

- ✓ **Step 4:** Calculate TF-IDF

Calculate the TF-IDF by multiplying the term frequency with the inverse document frequency for each word.

Text 1: TF-IDF: [0, 0, 0, 0, $\log(2/2)$, 0]

Text 2 TF-IDF: [0, 0, 0, 0, 0, $\log(2/2)$]

The TF-IDF vector representations capture the importance of words in each sentence. Words that appear frequently in a specific sentence but rarely in other sentences will have higher TF-IDF Values, indicating their significance in distinguishing that particular sentence.

2. Count Vectorization:

Count vectorization represents text as a matrix of word counts. Each document is converted into a vector that captures the occurrences of each word. It considers word frequency, offering a more comprehensive understanding of word distribution compared to bow.

Example:

Text 1: "I love to eat pizza." **Text 2:** "I love to eat burgers."

- ✓ **Step 1:** Create a Vocabulary

The only words are: ["I", "love", "to", "eat", "pizza", "burgers"]

- ✓ **Step 2:** Vectorization

Text 1: [1, 1, 1, 1, 1, 0]

Explanation: The count of each word in the vocabulary is [1, 1, 1, 1, 1, 0] for the words ["I", "love", "to", "eat", "pizza", "burgers"].

The word "pizza" and "burgers" have a count of 0 since they do not appear in Sentence 1.

Text 2: [1, 1, 1, 1, 0, 1]

Explanation: The count of each word in the vocabulary is [1, 1, 1, 1, 0, 1] for the words ["I", "love", "to", "eat", "pizza", "burgers"].

The word "pizza" has a count of 0 since it does not appear in Sentence 2.

3. Bag-of-Words:

The bag-of-words (BOW) model is a representation that turns arbitrary text into fixed-length vectors by counting how many times each word appears.

Example:

Text 1: "I love to eat pizza." Text 2: "I love to eat burgers."

Text 3: "I love to eat pizza and burgers."

- ✓ **Step 1:** Determine the Vocabulary

The only words are: ["I", "love", "to", "eat", "pizza", "and", "burgers"]

Text 1: [1, 1, 1, 1, 0, 0,0]

Text 2: [1, 1, 1, 1, 0, 0,1]

Text 2: [1, 1, 1, 1, 1, 1,1]

4. Results and discussion

The suggested machine learning algorithms necessitate user-defined initialization of specific parameters (as outlined in Table 10). Within this framework, it becomes beneficial to devise an experimental design or propose methodologies for identifying the optimal parameter combinations automatically. Our approach involves exploring various techniques, including hyper parameter selection methods, to determine the most suitable parameters for each algorithm. Subsequently, we aim to compare the outcomes yielded by the default configuration of each algorithm against those achieved through our proposed parameter selection methodology.

Table III 10: Initialization parameters for Classifiers.

Classifiers	Parameters
Random forests	N_estimators == represented the number of forest trees Max_depth ==is the depth of the tree Min_sample_split == represents the class nodes upper Min_sample_leaf == are the leaves
KNN	n_neighbors == number of neighbors in the voting process p_values ==or probability value is, for a model given statistic
Naive Bayes	Alpha == probabilité a priori de différences classes
SVM	Gamma == Gamma is a hyper parameter that we must define before training the model. Gamma decides to curvature we want in a decision limit C == Cost of linear separability violation (it indicates to SVM optimization to what extent we wish to avoid to misclassify each training example).
Decision Tree	Criterion== The function used to measure the quality of a split (e.g., Gini impurity or information gain). Max Depth== The maximum depth of the tree. Min Samples Split==The minimum number of samples required to split an internal node.

	<p>Min Samples Leaf== The minimum number of samples required to be in a leaf node.</p> <p>Max Features== The maximum number of features considered for splitting a node.</p>
--	--

4.1. Results of classifiers with and without hyper parameter tuning

Automatic hyper parameter optimization techniques, also known as hyper parameter optimization, automate the process of selecting the best hyper parameters for machine learning models. These techniques aim to find the optimal combination of hyper parameters that maximize the performance of the model on a given dataset, lead to improved model performance and reduced manual effort.

In this project, we perform grid search method to streamline the process of hyper parameter tuning. Grid search is a hyper parameter tuning technique used to find the optimal combination of hyper parameters for a machine learning model. It works by searching through a predefined grid of hyper parameter values and evaluating the model's performance using cross-validation. Here's how grid search works:

- ✓ **Define Hyper parameter Grid:** Specify a grid of hyper parameter values to search through. Each hyper parameter will have a list of possible values to try.
- ✓ **Define Evaluation Metric:** Choose an evaluation metric to assess the performance of the model for each combination of hyper parameters. Common evaluation metrics include accuracy, precision, recall, F1-score, or area under the ROC curve (AUC).
- ✓ **Cross-Validation:** Split the training data into multiple folds. For each combination of hyper parameters, train the model on $k-1$ folds and evaluate its performance on the remaining fold. Repeat this process k times (where k is the number of folds) to ensure that each fold is used as the validation set exactly once.
- ✓ **Select Best Model:** After evaluating the model's performance for all combinations of hyper parameters, select the combination that yields the best performance according to the chosen evaluation metric.
- ✓ **Train Final Model:** Once the best combination of hyper parameters is identified, train the final model using the entire training dataset with these optimal hyper parameters.

The table below (Table 10) seems to be showing the performance (in terms of accuracy) of different machine learning models on different vectorization techniques (Tfidf Vectorizer, CountVectorizer, and Bag of Words) with both default parameters and hyper parameter tuning.

Table III 10: Performances of classifiers with and without Hyper parameter tuning

	Tfidf_vect		Count_vect		bow_vect	
	Default parameter	Hyper parametre	Default Parameter	Hyper Parameter	Default Parameter	Hyper Parametre
Knn	0.8234	0.8828	0.8067	0.8075	0.7981	0.8028
Naïve bayes	0.8907	0.8901	0.8707	0.8714	0.8687	0.8681
Svm	0.8780	0.9221	0.8860	0.9094	0.8854	0.9107
Decision tree	0.7981	0.7841	0.7921	0.8061	0.7728	0.8048
Random forest	0.8967	0.8947	0.8114	0.8841	0.8054	0.8807

By looking at the table above, the SVM model achieved the highest accuracy (0.9221) with the Count_vect vectorizer and a specific hyper parameter setting. The Random Forest model performed consistently well across different vectorizer and hyper parameter settings. The Decision Tree generally had the lowest accuracy among the models compared.

The high accuracy of SVM can be attributed to its strong binary classification ability and its Compatibility with TF-IDF. TF-IDF reduces the importance of common words while Highlighting the importance of rare ones, enabling SVM to deal effectively with text-based problems in high-dimensional spaces. The combination of SVM and TF-IDF improves its data classification performance.

The choice of vectorizer and hyper parameter tuning can significantly impact the performance of machine learning models for text classification.

The table only shows accuracy scores, which is one metric for evaluating model performance. Other factors like *precision*, *recall*, and *F1* score might also be important depending on the application. So, the results might not generalize to other datasets or tasks.

It's important to compare different models and settings to find the best combination for fake news detection task.

The following table illustrates the different hyper parameter settings:

Table III 11: Hyper parameters of all algorithms

Model	Best parameter		
	Count	Tfidf	BOW
Knn	n_neighbors = 3 weights= uniform algorithm=auto		n_neighbors = 3 weights=distance algorithm=auto
Naïve bayes	Alpha = 0.01 Fit_prior=true		
Svm	Gamma= 'scale' C = 10 kernel=rbf		Gamma= 'auto' C = 10 kernel=rbf
decision tree	Max_depth = 10 Criterion=gini Min_sample_split =2 Min_sample_leaf =1	Max_depth = 10 Criterion= gini Min_sample_split = 5 Min_sample_leaf =2	Max_depth = 10 Criterion= gini Min_sample_split = 2 Min_sample_leaf = 1
Random forest	N_estimators ==300 Max_depth = 15	N_estimators ==100 Max_depth = 15	N_estimators == 300 Max_depth = 15

4.2. Results using features combination

It is important to highlight that comparable results were attained by employing the automatic hyper-parameter settings and Default Initialization. This symmetry can be attributed to the combination of each pair of features. The table below (Table 12) presents the obtained outcomes:

Table III 12: Accuracy of algorithms using features combination

	Tfidf+cont_vect		Tfidf+bow_vect	
	Default parameter	Hyper parametre	Default parameter	Hyper parametre
Knn	0.8161	0.8068	0.8187	0.8068
Naïve bayes	0.8194	0.8195	0.8194	0.8195
Svm	0.8474	0.8648	0.8647	0.8648
Decision tree	0.7748	0.7975	0.7748	0.7948
Random forest	0.8674	0.8734	0.8640	0.8754

The table above includes a comparison of the performance of several models using a range of different parameters, using Tfidf with Count_vect and bow_vect:

- SVM shows strong performance with a significant difference between the default settings and after hyper parameter adjustment.

Naïve Bayes shows stability in performance regardless of the hyper parameter.

Random Forest shows superior performance, especially after hyper parameter tuning.

- Decision Tree also benefits from hyper parameter tuning.

- Knn shows a slight improvement after adjusting the hyper parameter.

Therefore, the control results depend on the nature of the data and its ability to be improved, and the results may differ by choosing different hyper parameters.

The following table illustrates the used hyper-parameters:

Table III 13: Hyper-parameters of algorithms using features combination

Model	Best Par_tfidf+bow	Best par_tfidf+cont
Knn	n_neighbors = 9 algorithm=auto	n_neighbors = 9 algorithm=auto
Naïve bayes	Alpha = 0.5 Fit_prior=false	Alpha = 0.5 Fit_prior=false
Svm	Gamma= 'scale' C = 1 kernel=rbf	Gamma= 'scale' C = 1 kernel=rbf
Decision tree	Max_depth = 5 Criterion=gini Min_sample_split =2 Min_sample_leaf =1	Max_depth = 5 Criterion=gini Min_sample_split =5 Min_sample_leaf =2
Random forest	N_estimators =100 Max_depth = none Bootstrap=false Criterion=gini Min_sample_split =5 Min_sample_leaf =1	N_estimators =500 Max_depth = none Bootstrap=false Criterion=entropy Min_sample_split =5 Min_sample_leaf =1

4.3. Comparison

We will compare the accuracy of each applied algorithm with the properties using hyper-parameters, as shown below.

✓ Accuracy of algorithms with properties for extraction using Hyper parameter

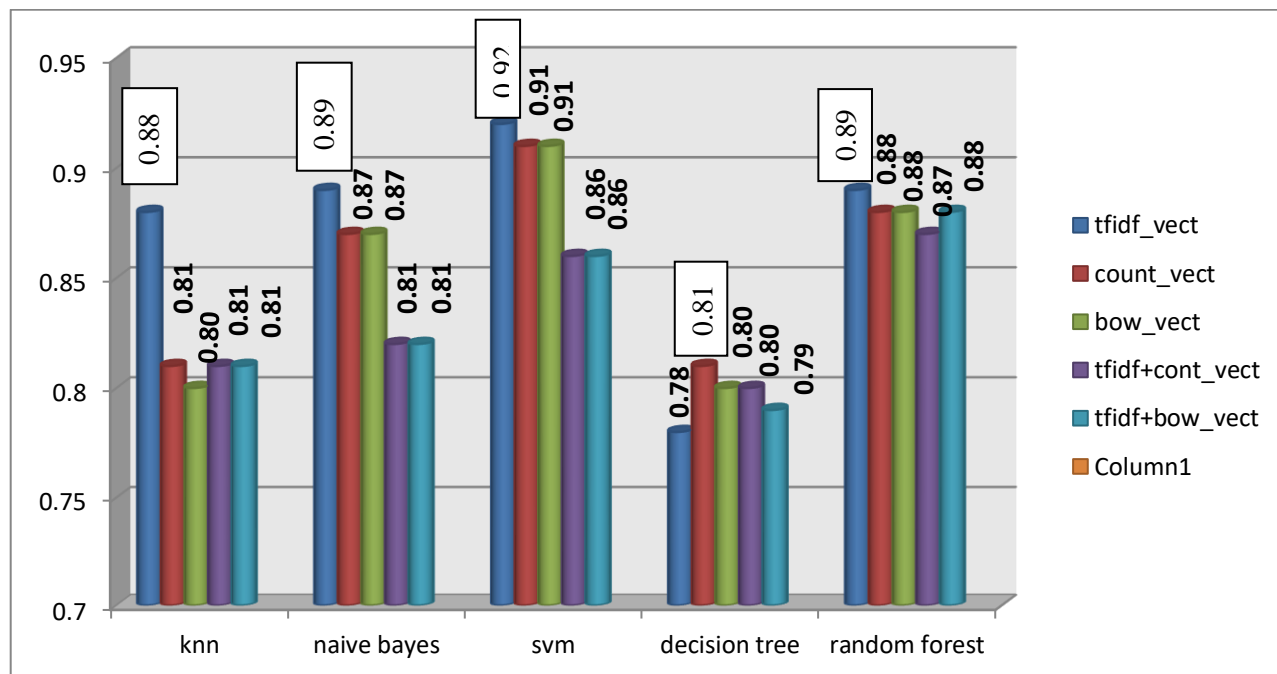


figure III- 3:Accuracy of algorithms with properties for extraction using hyper parameter

1- k-Nearest Neighbors (knn):

It is shown that knn performs well using the tf-idf transform (tfidf_vect) with a score of 0.88, but slightly underperforms using the count (count_vect) and bag-of-words (bow_vect) transforms. This indicates that the tf-idf conversion helped improve the performance of knn.

2-Naive Bayes: shows consistent performance across all types of transformations, and delivers the highest accuracy using a tf-idf transformation of 0.89. This shows the effectiveness of the Naive Bayes model in dealing with a variety of data transformations.

3- Support Vector Machine (SVM):

shows superior performance with all types of transformation models, achieving the highest accuracy scores among all models. This indicates the power of SVM in dealing with different data transformations and its effectiveness in separating classes.

4-Decision Tree:

The results may indicate that Decision Tree has slightly lower performance compared to some other models, especially using tf-idf and bag-of-words transformations. There may be a need to modify model parameters to improve its performance.

5-Random Forest:

shows strong performance with all types of transformations, especially with tf-idf and count transformations. Random Forest is a model that addresses the problems of random increase and performance improvement.

In conclusion, the choice of model depends on several factors such as the type of data, the size of the data, and the goal of the task. It is preferable to conduct a detailed analysis of the individual characteristics and needs of the problem you are facing to make the appropriate decision about the model to use.

✓ Accuracy of algorithms with properties for extraction using Default Initialization

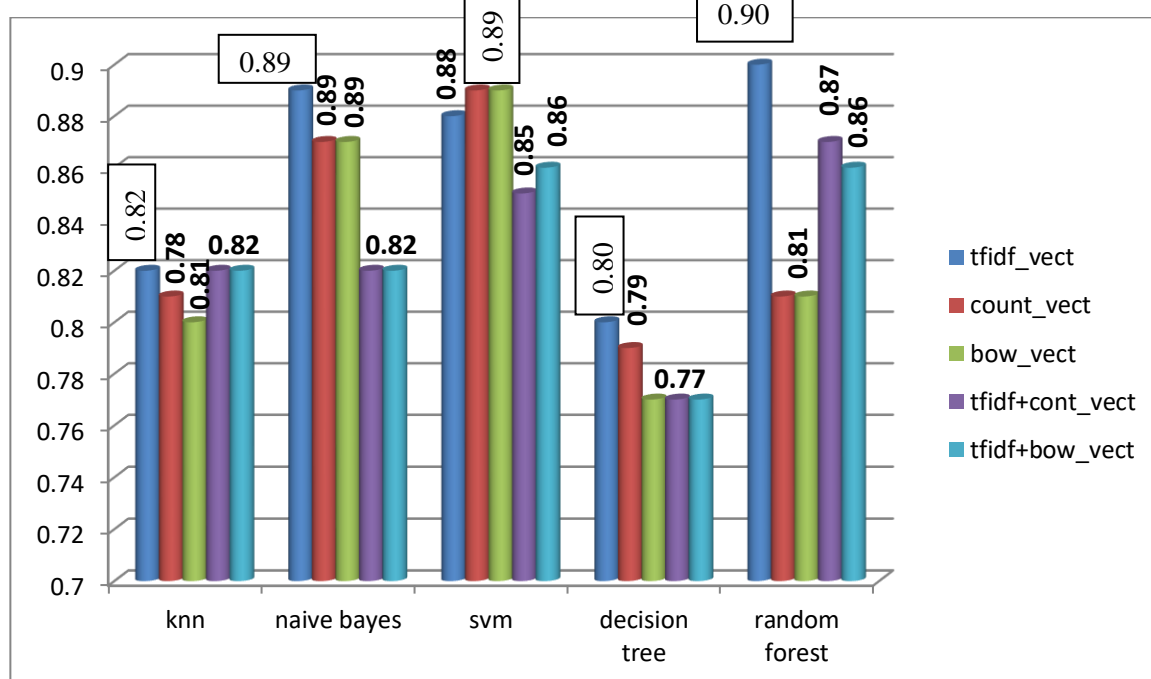


figure III- 4:accuracy of algorithms with properties for extraction using Default Initialization

After analyzing the results, we note that :

➤ **KNN (k-Nearest Neighbors):**

It is noted that the performance of KNN is similar using different representation techniques (Vectors), with accuracy ranging between 80% and 82%. This indicates that the KNN model is not significantly influenced by the method used to represent text.

➤ **Naive Bayes:**

Excellent performance is demonstrated using `tfidf_vect` with 89% accuracy. However, the performance drops dramatically with `tfidf+cont_vect` and `tfidf+bow_vect` to 82%. This indicates that using features such as `count_vect` and `bow_vect` does not effectively contribute to improving the performance of Naive Bayes.

➤ **SVM (Support Vector Machine):**

It is noted that SVM using ``count_vect`` and ``bow_vect`` has an accuracy of 89%, which is the highest value in this table. This indicates that the SVM model makes significant use of the representation of text using Bag of Words (``bow_vect``) and Frequency Count (``count_vect``) representation.

➤ **Decision Tree:**

The Decision Tree model shows modest performance regardless of the technique used. The accuracy level ranges between 77% and 80%. It seems that Decision Tree does not depend much on the type of representation used.

➤ **Random Forest:**

Random Forest shows excellent performance using ``tfidf_vect`` with 90% accuracy. This indicates that representing text by Term Frequency-Inverse Document Frequency (``tfidf_vect``) contributes significantly to improving the performance of this model. With other techniques, a decrease in performance is shown due to the representation effect.

In general, it turns out that ``tfidf_vect`` repeatedly emerges as the best technique in most models, especially in the case of Random Forest and Naive Bayes. However, performance can vary depending on the nature of the data and the type of task, so it is preferable to choose the optimal technique based on the project context and classification requirements.

5. Conclusion

In this chapter, we outlined the technical framework of our fake news detection system, exploring the tools and methodologies employed in its development. We highlighted the importance of Python and its libraries, such as NLTK and Scikit-learn, in facilitating machine learning tasks. Data preparation techniques, including preprocessing and representation methods like TF-IDF Vectorization, were elucidated. Additionally, we evaluated various machine learning algorithms, identifying optimal parameter settings and feature combinations for fake news detection. Overall, this chapter underscores our commitment to leveraging machine learning and natural language processing to combat fake news proliferation effectively.

General Conclusion

In conclusion, this research addressed the critical task of detecting fake news on social media, using basic concepts and principles. The exploration phase included a comprehensive review of existing methods for identifying fake news, with a particular focus on supervised learning algorithms.

Various machine learning algorithms, including Tf-idf, Kent, and Bow features, were considered for fake news detection. The research team opted for the Support Vector Machine (SVM) as the primary method for detecting fake news.

However, significant progress was demonstrated when using SVM with Tf-idf features, demonstrating its superiority over other combinations. This specific configuration, with finely tuned hyper parameters, resulted in a remarkable accuracy rate of 92.21%. Hence, we conclude that SVM emerges as the most effective algorithm for detecting fake news, due to its strong capabilities in dealing with the complex nature of misinformation on social media platforms. The success achieved with SVM underscores the importance of exploring and refining machine learning models to adapt to the evolving challenges posed by fake news.

Future works

- Develop advanced machine learning models: More complex and sophisticated models for detecting fake news should be explored and developed, including deep learning and end-to-end learning techniques.
- Improving the quality and quantity of data: It is important to collect larger and more diverse datasets to train models, while ensuring the quality and validity of the data.
- Integrating information from multiple sources: The accuracy of fake news detection can be improved by integrating information from multiple sources, such as social, biological, and historical data.
- Developing continuous updating mechanisms: Mechanisms must be established to update models and databases on a regular basis to keep pace with developments in the spread of fake news and its changes.
- Enhance transparency and accountability: Transparent criteria should be established to assess the performance of fake news detection models and ensure that stakeholders are held accountable when needed.

Bibliography

- [1] O'Brien, N. (2018). *Machine learning for detection of fake news* (Doctoral dissertation, Massachusetts Institute of Technology).
- [2] Oshikawa, R., Qian, J., & Wang, W. Y. (2018). A survey on natural language processing for fake news detection. arXiv preprint arXiv:1811.00770.
- [3] William Yang Wang, "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection, to appear in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017), short paper, Vancouver, BC, Canada, July 30-August 4, ACL. (<https://www.kaggle.com/datasets/csmalarkodi/liar-fake-news-dataset>)
- [4] Tandoc, E. C., Thomas, R. J., & Bishop, L. (2021). What is (fake) news? Analyzing news values (and more) in fake stories. *Media and Communication*, 9 (1), 110-119.
- [5] Baair, N. F., & Djefal, A. (2021, February). Fake news detection using machine learning. In 2020 2nd International workshop on human-centric smart environments for health and well-being (IHSH) (pp. 125-130). IEEE.
- [6] De Beer, D., & Mathee, M. (2021). Approaches to identify fake news: a systematic literature review. *Integrated Science in Digital Age 2020*, 13-22.
- [7] Guo, Z., Schlichtkrull, M., & Vlachos, A. (2022). A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, 10, 178-206.
- [8] Vlachos, A., & Riedel, S. (2014, June). Fact checking: Task definition and dataset construction. In Proceedings of the ACL 2014 workshop on language technologies and computational social science (pp. 18-22).
- [9] Ahmed, S., Hinkelmann, K., & Corradini, F. (2022). Combining machine learning with knowledge engineering to detect fake news in social networks-a survey. arXiv preprint arXiv:2201.08032.
- [10] Hassan, N., Zhang, G., Arslan, F., Caraballo, J., Jimenez, D., Gawsane, S., ... & Tremayne, M. (2017). Claimbuster: The first-ever end-to-end fact-checking system. *Proceedings of the VLDB Endowment*, 10(12), 1945-1948.
- [11] Cazalens, S., Leblay, J., Lamarre, P., Manolescu, I., & Tannier, X. (2018). Computational fact checking: a content management perspective. *Proceedings of the VLDB Endowment (PVLDB)*, 11(12), 2110-2113.
- [12] Hakak, S., Alazab, M., Khan, S., Gadekallu, T. R., Maddikunta, P. K. R., & Khan, W. Z. (2021). An ensemble machine learning approach through effective feature extraction to classify fake news. *Future Generation Computer Systems*, 117, 47-58.

- [13] Mishra, S., Shukla, P., & Agarwal, R. (2022). Analyzing machine learning enabled fake news detection techniques for diversified datasets. *Wireless Communications and Mobile Computing*, 2022, 1-18.
- [14] Sarnovský, M., Maslej-Krešňáková, V., & Ivancová, K. (2022). Fake news detection related to the covid-19 in slovak language using deep learning methods. *Acta Polytechnica Hungarica*, 19(2), 43-57.
- [15] Manzoor, S. I., & Singla, J. (2019, April). Fake news detection using machine learning approaches: A systematic review. In *2019 3rd international conference on trends in electronics and informatics (ICOEI)* (pp. 230-234). IEEE.
- [16] Zhang, Q., Guo, Z., Zhu, Y., Vijayakumar, P., Castiglione, A., & Gupta, B. B. (2023). A deep learning-based fast fake news detection model for cyber-physical social services. *Pattern Recognition Letters*, 168, 31-38.
- [17] Cherif, I. E. Deep Learning pour la prédiction des tremblements de terre.
- [18] Harrag, F., & Djahli, M. K. (2022). Arabic fake news detection: A fact checking based deep learning approach. *Transactions on Asian and Low-Resource Language Information Processing*, 21(4), 1-34.
- [19] Seddari, N., Derhab, A., Belaoued, M., Halboob, W., Al-Muhtadi, J., & Bouras, A. (2022). A hybrid linguistic and knowledge-based analysis approach for fake news detection on social media. *IEEE Access*, 10, 62097-62109.
- [20] Karwa, R. R., & Gupta, S. R. (2022). Automated hybrid Deep Neural Network model for fake news identification and classification in social networks. *Journal of Integrated Science and Technology*, 10(2), 110-119.
- [21] Okunoye, O. B., & Ibor, A. E. (2022). Hybrid fake news detection technique with genetic search and deep learning. *Computers and Electrical Engineering*, 103, 108344.
- [22] Tamene, A. Classification Automatique des documents textuels.
- [23] Refonaa, J., Reddy, G., Shabu, S. J., Dhamodaran, S., & Antony, J. C. (2022). Fake News Detection Using Machine Learning Approaches. *Mathematical Statistician and Engineering Applications*, 71(3s2), 485-494.
- [24] DJABALLAH, M. A. (2021). Système de prédiction de la consommation d'énergie basé Deep Learning.
- [25] Difani, F., & Merhabaoui, K. Machine Learning Based Model For Fake News Spreaders Detection (Doctoral dissertation, UNIVERSITY OF KASDI MERBAH OUARGLA).
- [26] Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. *Information*, 10(4), 150.

[27] Sirikulviriya, N., & Sinthupinyo, S. (2011, May). Integration of rules from a random forest. In International Conference on Information and Electronics Engineering (Vol. 6, pp. 194-198).

[28] Mohammed el mehdi Kadri-abdelmounaim SAGGAI, « breast detection using deep learning »breast detection using deep learning.

[29] Rohera, D., Shethna, H., Patel, K., Thakker, U., Tanwar, S., Gupta, R., ... & Sharma, R. (2022). A taxonomy of fake news classification techniques: Survey and implementation aspects. IEEE Access, 10, 30367-30394.