PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education And Scientific Research

University of Tlemcen – ABOU BAKR BELKAID - Algeria
FACULTY OF SCIENCE - DEPARTMENT of COMPUTER SCIENCE

Graduation Project
To obtain a Master's degree in Computer Science
Specialty: Information and Knowledge System
On the subject:

Application of Game-Based Metaheuristics for Fake News Detection.

Presented by :
M^{lle} BENSALAH Sarra
M^{lle} KACIMI Fatima Zahra
Presented : 25/06/2024

Before the jury composed of:

Chairman :	M^{me} Abdeldjelil Hanane	Assitant professor
Examiner:	M^{me} Marouf Radja	Assistant professor
Supervisor:	M^{r} MAHAMMED Nadir	Associate professor A
Co-supervisor:	M^{me} SEKKAL Nawel	Associate professor B

College year 2023/2024

Acknowledgements

I am deeply grateful to Almighty Allah for giving me the strength and ability to complete this work.

First,I would like to express my deep gratitude to our teacher and supervisor,Mr. MAHAMMED Nadir, for his availability, constant support, and invaluable advice throughout our project. His kind presence and insightful guidance have been of great help in our endeavors.

I also wish to extend my heartfelt thanks to my co-supervisor, Mrs. SEKKAL Nawel, for the time she devoted with great interest and understanding. Her assistance and guidance have been instrumental in the successful completion of this dissertation.

I am immensely thankful to the members of the jury, including Mr. Abdejlil and Mrs. Marouf, who agreed to dedicate their time and expertise to evaluate our work. Their constructive remarks and comments have allowed us to improve our dissertation and make it more robust.

I am deeply grateful to my parents, who have always been by my side, supporting, encouraging, and offering their invaluable advice. Their unconditional love and unwavering support have been a source of motivation and strength throughout my academic journey.

My gratitude also goes to my friends and colleagues who shared this journey with me, providing moral support, encouragement, and their precious camaraderie.

Thank you all immensely for your support, trust, and contribution to the realization of this project. Your encouragement has enabled us to overcome challenges and achieve our goals.

Dedication

I dedicate this work to those who have always been there for me, providing support and encouragement throughout this academic and personal journey.

To my mother, for her unconditional love, infinite patience, and constant encouragement that have enabled me to overcome every challenge. You have been my source of strength and inspiration.

To my father, for his wisdom, unwavering support, and wise counsel. You taught me the value of hard work and perseverance, and your example has guided me at every step of this journey.

To my sister, for her affection, understanding, and comforting words. Your presence has been a pillar of stability and joy in my life.

To all my friends, for their camaraderie, encouragement, and the unforgettable moments we have shared. Your support and words of encouragement have been essential sources of motivation.

To my mentors, for their expertise, wise advice, and academic support. Your guidance has been invaluable in completing this thesis, and your confidence in me has allowed me to grow as a researcher and as a person.

And not forgetting my partner and best friend Fatima Zahra for her moral support, patience, and understanding throughout this project.

Thank you to each of you for your support, love, encouragement, and trust, which have been invaluable throughout our journey. We are deeply grateful to have you by our side, and we dedicate this work to you with profound gratitude.

Sarra

Dedication

At the beginning, I praise and thank Allah, the Almighty, for His guidance in the completion of this research.

After the praises, I would like to express my gratitude to those who supported me with their prayers, to those who stayed up at night to illuminate my path, to the source of tenderness, to the most wonderful woman in existence, my dear mother, "Z. Khadidja".

And to the one whose name I carry with great pride, to the one who removed the thorns from my path to prepare the way for knowledge and taught me that the world is a struggle, to my dear father, "K. Mohammed".

To my brothers "Ahmed", "Youssef", "Abderrahman", and "Ibrahim", as well as my sister "Aicha", your support and advice played an essential role in the attainment of my degree. Thank you sincerely for all the moral and emotional support.

To my lifelong friends, "Riham", "Kawter", "Amel", "Amina", "Yousra","chaimaa" and "Ibtihal "who support me and rejoice in my success and achievements.

Not forgetting my partner and best friend "Sarra" for her moral support, patience, and understanding throughout this project.

To all those who are close to "Fatima" and who encouraged her to pursue learning and achieve success.

My professors and benefactors, I sincerely thank them for their appreciation and advice.

To all of these people, I humbly dedicate this research, praying to Allah, the Almighty, that He makes it beneficial for us.

Fatima Zahra

Résumé

La montée des réseaux sociaux en ligne a considérablement augmenté la propagation des fausses informations, posant des défis à la confiance et à la sécurité sur ces plateformes. Cette dissertation présente une étude complète comparant divers algorithmes de détection de fausses informations. Nous évaluons des algorithmes d'apprentissage automatique supervisés et non supervisés sur plusieurs ensembles de données réels provenant de réseaux sociaux tels que Facebook, Twitter et Instagram.

Nos résultats révèlent qu'aucun algorithme ne surpasse systématiquement les autres dans toutes les situations. Chaque méthode présente ses forces et ses faiblesses en fonction des caractéristiques de l'ensemble de données. Les modèles supervisés, y compris la Forêt Aléatoire, le KNN, le Boosting de Gradient, le SVM, le Naïve Bayes et l'Arbre de Décision, montrent généralement de bonnes performances. Pour les modèles non supervisés, le clustering hiérarchique dépasse souvent le k-means.

Pour améliorer la performance de détection, nous intégrons plusieurs stratégies et introduisons l'Algorithme d'Optimisation par le Golf (GOA) pour optimiser un modèle supervisé choisi pour la détection des fausses nouvelles. Nos résultats montrent que le GOA surpasse les algorithmes traditionnels en termes de précision, de rappel et de score F1.

Cette recherche contribue à une meilleure compréhension du comportement des algorithmes dans divers contextes et souligne l'importance de combiner plusieurs techniques pour obtenir des résultats optimaux. Les travaux futurs se concentreront sur l'élargissement du champ de comparaison pour inclure davantage de modèles bio-inspirés et explorer l'applicabilité pratique du GOA dans divers domaines.

Mots clé : Fausses informations, détection, réseaux sociaux, apprentissage automatique, apprentissage supervisé, apprentissage non supervisé, métaheuristiques, algorithmes basés sur les jeux.

Abstract

The rise of online social networks has significantly increased the spread of fake news , posing challenges to trust and security on these platforms. This dissertation presents a comprehensive study comparing various algorithms for detecting false information. We evaluate both supervised and unsupervised machine learning algorithms on multiple real-world datasets from social networks such as Facebook, Twitter, and Instagram.

Our findings reveal that no single algorithm consistently outperforms others in all scenarios. Instead, each method has its strengths and weaknesses depending on the dataset characteristics. Supervised models, including Random Forest, KNN, Gradient Boosting, SVM, Naive Bayes, and Decision Tree, generally perform well. For unsupervised models, hierarchical clustering often surpasses k-means.

To enhance detection performance, we integrate multiple strategies and introduce the Golf Optimization Algorithm (GOA) to optimize a chosen supervised model for fake news detection. Our results show that the GOA outperforms traditional algorithms in terms of accuracy, precision, recall, and F1-score.

This research contributes to a better understanding of algorithm behavior in various contexts and highlights the importance of combining multiple techniques to achieve optimal results. Future work will focus on expanding the scope of comparison to include more bio-inspired models and exploring the practical applicability of the GOA in diverse domains.

Keywords : Fake News, detection, social networks, machine learning,supervised learning, unsupervised learning, metaheuristics, games-based algorithms.

Table des matières

Table des figures

Liste des tableaux

List of Abbreviations

OSN: Online Social Networks.

ML: Machine Learning.

SVM: Support Vector Machines.

RF: Random Forest.

RNN: Recurrent Neural Networks.

CNN : Convolutional Neural Networks.

NB: Naive bayes.

LR: Logistic Regression.

GB: Gradien Boosting.

KNN: K-Nearest Neighbors.

DT: Decision tree.

SAF: Stacked Attention Network.

SAF /S: Stacked Attention Network with Self-attention.

SAF /A: Stacked Attention Network with Attention.

GA: Genetic Algorithm.

PSO : Particle Swarm Optimization.

GOA: Golf Optimization Algorithm.

Chapter 1

Introduction

The advent of social networking platforms such as Facebook, Twitter, and Instagram has revolutionized the landscape of global communication, transforming the way individuals connect, interact, and disseminate information worldwide. These platforms have facilitated the creation of online communities, enabling users from diverse locations to engage in communication and share content seamlessly. With the rise of online social networking, the dynamics of international communication have undergone a significant shift, allowing for unprecedented levels of interaction, meaningful discourse, and information dissemination to vast audiences. However, alongside the myriad benefits, these platforms have also witnessed a surge in the prevalence of false news and misinformation, posing serious threats to user safety, social order, and overall user experience. Establishing effective methods for identifying and mitigating the spread of false news on social media platforms has become imperative. This thesis endeavors to propose a comprehensive approach for automatically detecting and flagging instances of false news dissemination using state-of-the-art techniques from natural language processing, machine learning, and data analytics. Implementation of this method aims to facilitate timely intervention, efficient moderation, and enhance user safety and trust in social media environments.

1.1 Problem statement

Concerns regarding the proliferation of false news and misinformation on online social networking platforms have escalated significantly. The rampant spread of false information poses a substantial threat to user security, privacy, and the integrity of online discourse. Detecting false news presents formidable challenges, as malicious actors often craft deceptive narratives designed to evade detection and manipulate public opinion. Traditional methods of identifying false news based on content analysis or source credibility often prove inadequate in combating the dynamic and sophisticated tactics employed by purveyors of misinformation. Therefore, there is an urgent need for innovative approaches to detecting false news that leverage advanced analytics, machine learning, and artificial intelligence to effectively address this pressing issue.

1.2 Current literature and motivation

Protecting users from the harmful effects of false news and misinformation is a paramount concern in online social networks. Recent advancements in machine learning techniques have facilitated the automation and enhancement of false news detection processes.

Researchers have explored various methodologies, including Bayesian networks and clustering models such as K-Means, Naive Bayes, Support Vector Model, and K-Nearest Neighbor, to identify and mitigate the spread of false news. Furthermore, efforts have been made to critically evaluate and refine machine learning-based approaches through experiments, surveys, and literature reviews aimed at enhancing their effectiveness in combating false news dissemination.

1.3 Contribution and results

This study conducts a novel comparative analysis of different strategies for detecting false news in online social networks, aiming to contribute to the advancement of existing literature in the field. Platforms like Facebook, Twitter, and Instagram are among those considered in our analysis. Additionally, we explore the potential of bio-inspired algorithms in addressing the challenge of false news detection. Our objective is not to identify the ultimate false news detection method but rather to identify approaches within each category that align most effectively with the characteristics of target datasets.

The results of our study indicate that supervised machine learning models show promise in detecting false news in social media, with opportunities for further enhancement through parameter tuning. Similarly, unsupervised models like k-means exhibit potential in identifying patterns indicative of false news dissemination. Moreover, we propose a metaheuristic approach that integrates social media analytics with bio-inspired computing to locate instances of false news dissemination.

1.4 Dissertation structure

This thesis comprises four chapters. Chapter 2 offers background information and situates the dissertation within the contemporary state-of-the-art, leading to Chapter 3. The methods employed for conducting the comparative analysis are detailed in Chapter 4. Ultimately, Chapter 5 presents the results and provides a comprehensive discussion of the findings.

Chapter 2

Basic Concepts

2.1 Introduction

In today's interconnected world, social networks have become integral to our daily lives. They are essential tools for communication, fostering connections, and sharing ideas. However, despite these significant benefits, the misuse of social media is a growing concern, particularly through the spread of fake news. Misleading information, carefully crafted to deceive and manipulate unsuspecting users, is a prime example of this malicious activity. Such fabricated stories are used for a variety of harmful purposes, including spamming, phishing, spreading misinformation, inciting public panic, and even facilitating cyberbullying. It is crucial to remain vigilant, recognize the presence of fake news, and strive for a safer and more authentic digital environment.

This chapter will review various methods to detect and combat fake news. But first, let us clarify the fundamental concepts related to this issue.

2.2 Fake News detection

2.2.1 Web 2.0

is a term used to describe a new generation of web-based technologies and platforms that facilitate user-generated content, collaboration, and interaction. It represents a shift from static, one-way communication to dynamic, interactive communication. Web 2.0 applications include social networking sites, blogs, wikis, and other tools that allow users to create and share content, participate in online communities, and collaborate on projects

in real-time. This has transformed the internet into a more collaborative and participatory space [21].

2.2.2 Online Social Networks

are virtual platforms that allow people to create and maintain social connections, share content, and communicate with others over the internet. Social networks have become an integral part of modern life, with billions of users around the world connecting and sharing information through platforms like Facebook, Twitter, and LinkedIn. These platforms have transformed the way people interact and communicate, enabling individuals and groups to connect with each other regardless of geographic location.[2]

One of the key features of online social networks is the ability to create personal or professional profiles. Users can include information such as their name, location, interests, and profile picture. They can also connect with others by sending friend requests, following profiles, and joining groups or communities based on shared interests. Social networks enable users to share content like photos, videos, and messages with their connections, fostering new forms of digital media and user-generated content.

The rise of online social networks has profoundly impacted society, transforming communication and interaction. They facilitate the exchange of information and ideas, creating new digital media and online communities. Businesses and organizations have also benefited, using social networks for targeted and personalized marketing and advertising.

However, social networks raise concerns about privacy, security, and the potential for misinformation and online harassment. The ease with which fake news can spread on these platforms is particularly alarming. Many users are unaware of the security dangers associated with these kinds of communications, including the rapid dissemination of false information, which can lead to widespread misinformation.

2.2.3 Fake news

Fake news, also known as misinformation, is a form of deliberate misinformation or deceptive content created, disseminated, and shared with the intention of misleading readers into believing false or unverified information. Fake news can be used for various purposes such as manipulating public opinion, spreading harmful rumors, political misinformation, or generating clicks and advertising revenue. It is essential to fact-check information before sharing it to combat the spread of fake news and preserve the integrity of information.

The main characteristics of fake news are :

- Sensational Headlines: Overly dramatic or shocking titles designed to grab attention.
- Lack of Sources: Often lacks credible sources or references to back up claims.
- Poor Grammar and Spelling: Frequently contains errors that indicate a lack of professionalism.
- Emotional Manipulation: Aims to provoke strong emotional reactions rather than provide balanced information.
- No Author Information: Often does not provide information about the author or the author's credentials.
- Dubious Website: Published on websites that do not have a reputation for accurate or trustworthy news reporting.
- Inconsistencies: Contains contradictory statements or data within the same article.

These points can help identify fake news and encourage critical thinking when consuming information.

Fake news types

Several Fake news types can be identified including:

- News satire.
- News parody.
- Fabrication.
- Manipulation.
- Advertising.
- Propaganda.

[22]

2.3 Machine Learning

Machine learning is a branch of artificial intelligence that develops algorithms and techniques to enable computers to learn from data and make predictions or decisions without explicit programming. The core principle of machine learning is that algorithms can automatically identify patterns and relationships within data, resulting in models capable of making accurate predictions or informed actions. This data-driven approach has transformed numerous industries, including healthcare, finance, marketing, and transportation, allowing businesses to gain valuable insights and enhance their decision-making processes.

2.3.1 Machine learning models

There are multiple possible setups when following an ML-based AD. On one hand, the setup to be used may depend on the availability of labels indicating the actual nature of the initial training data.[23]

- **Supervised learning** :Supervised learning is a fundamental area of machine learning. In this approach, a model is trained on labeled data, where each data point is paired with a known target variable or outcome. The aim is to learn a relationship between the input features and the corresponding output labels. Common tasks in supervised learning include classification and regression. Classification predicts a discrete class or category, while regression predicts a continuous value. Algorithms frequently used in supervised learning include decision trees, support vector machines, and neural networks. Popular algorithms in this field also include KNN, decision trees, SVM, and Naive Bayes.

- **Unsupervised learning**:Unsupervised learning, unlike supervised learning, involves working with unlabeled data, where the model identifies patterns or structures without specific output labels. Common tasks in unsupervised learning include clustering and dimensionality reduction. Clustering algorithms group similar data points based on their inherent similarities or distances, uncovering hidden structures in the data. Dimensionality reduction techniques reduce the complexity of high-dimensional data while preserving essential information by projecting it into a lower-dimensional space. Principal Component Analysis (PCA) and k-means clustering are popular algorithms used in unsupervised learning.

Supervised models

Naive bayes classifier Naive bayes is a machine learning algorithm based on bayes theorem it is a supervised learning task and also assumed that the predictive attributes are independent. We define the Bayes theorem as :

$$P(class/features) = P(class) * P(features/class)/P(features) \tag{2.1}$$

- P(class/features) : Posterior Probability
- P(class) : Class Prior Probability
- P(features/class) : Likelihood
- P(features) : Predictor Prior Probability

Bayesian classifier is based on Bayes' theorem. Naive Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence. It is made to simplify the computation involved and, in this sense, is considered "naive".[1]

Remarks on the Naive Bayesian Classifier (figure 2.1) :

- Studies comparing classification algorithms have found that the naive Bayesian classifier to be comparable in performance with decision tree and selected neural network classifiers.
- Bayesian classifiers have also exhibited high accuracy and speed when applied to large databases [1].

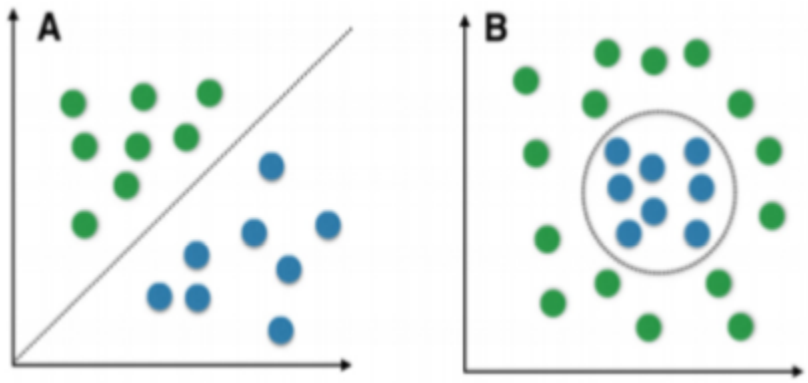

Figure 2.1: *Naive Bayes classifier [1]*

Decision tree Decision tree belongs to supervised learning algorithms. A decision tree is a popular classification method that generates tree structure where each node denotes a test on an attribute value and each branch represents an outcome of the test (see figure 2.2 [1])

The idea is to partition the data space into dense regions and sparse regions. It is a statistical-based algorithm where attributes are selected at the tree-of-nodes beginning at the root and ending at the leaves.

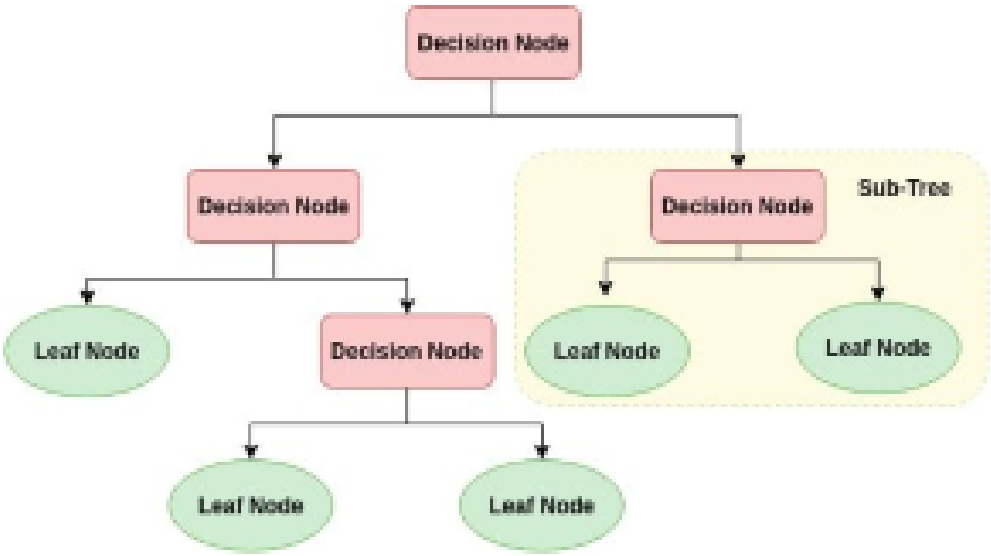

Figure 2.2: representation of decision tree [2]

[1] https://www.datacamp.com/

K-nearest neighbors The k-nearest neighbors (KNN) algorithm is a supervised machine learning algorithm used to solve both classification and regression problems,but especially in the classification.

The definition of nearest neighbors is based on the computation of the Euclidean distance from the new data point to each of the existing data points. The Euclidean distance is the most common distance measure [2].

The letter k is used to indicate the number of neighbors to use. To compute the k nearest neighbors, you simply compute the distance between your new data point and each of the data points in the training data. Depending on which number you have for k, you take the k data points that have the lowest distance.[2]

One of the drawbacks of K-Nearest Neighbors is that it is sensitive to inconsistent data (noisy) and missing value data. The figure below shows how the knn classifier works.

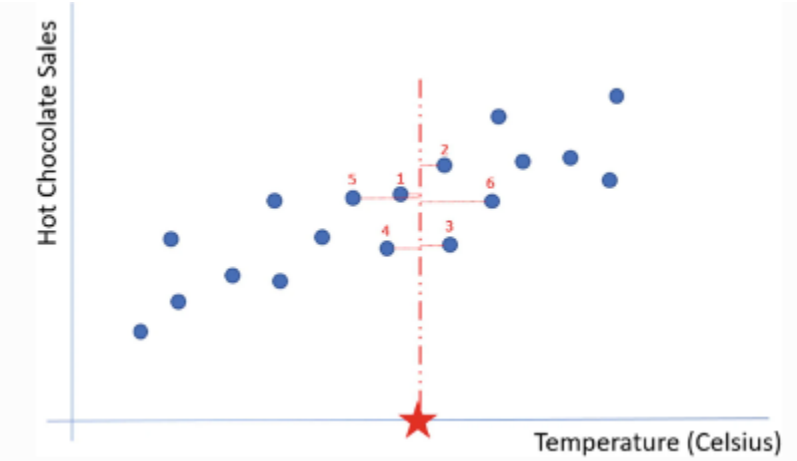

Figure 2.3: KNN classifier [2]

GRADIANT BOOSTING Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.[3] [24] When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest.[3] [24] [25] A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.

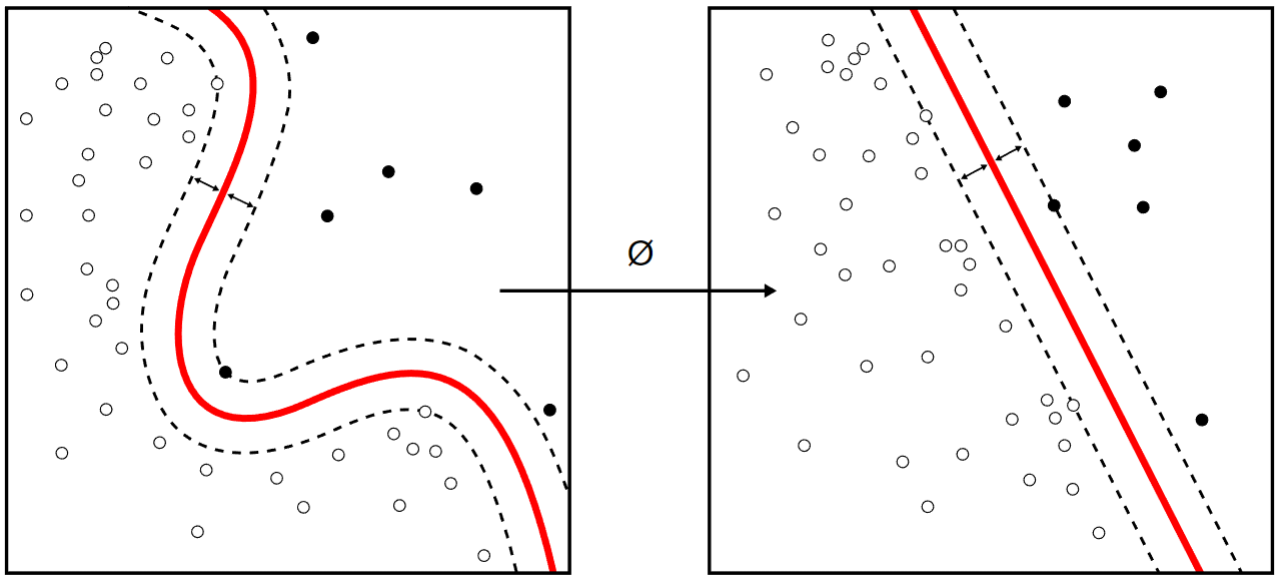

Figure 2.4: GRADIANT BOOSTING[3]

Random Forest Machine: Random Forest is a machine learning algorithm that considered as a supervised learning technique. It creates several Decision Trees on the subset of data.

Moreover, Random Forest is used in Regression and Classification of ML. It is proved the effectiveness of this algorithm on large datasets compared to other classifiers like: Neural Networks, Discriminant Analysis and Support Vector Machines (SVM)[26]

One of the most important benefits of Random Forest is that it can work with missing data, which is the relief of missing values by the variable that's common in a particular knot. The Random Forest can also handle big data snappily, give a advanced delicacy and help over-fitting problems. One the other hand, Random Forest requires numerous computational ressources and large memory for storehouse, due to the fact that it creates a lot of trees to save information piped generated from hundreds of individual trees.[4]

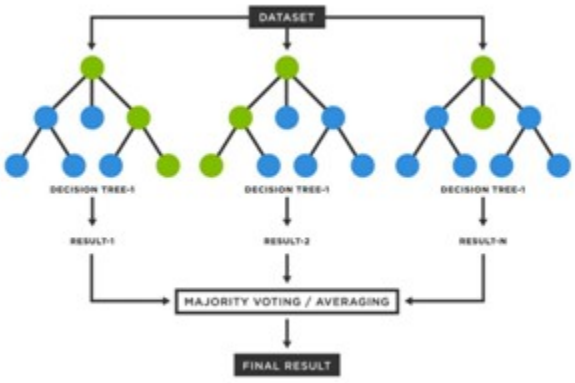

Figure 2.5: Random Forest Classifier [4]

Support Vectors Machine Support Vector Machines (SVMs) are supervised learning models for classification and regression problems. Support Vector Machines(SVMs) are supervised learning models for classification and regression problems. They can solve linear and nonlinear problems and use the concept of Margin to classify between classes.

SVMs give better accuracy than KNNs, Decision Trees, Naive Bayes Classifiers in most cases and have been known to outperform neural networks in a few instances.

The support vector machine algorithm's objective is to find a hyperplane in an N-dimensional space that distinctly classifies the data points and to find the optimal separating hyperplane or maximum-margin hyperplane, which separates the N different data points clusters [27].

- Support Vectors are the data points that are on or closest to the hyperplane and influence the hyperplane's position and orientation.
- Hyperplanes are decision boundaries that aid in classifying the data points.

  The figure 2.6 shows the representation pf hyper planes

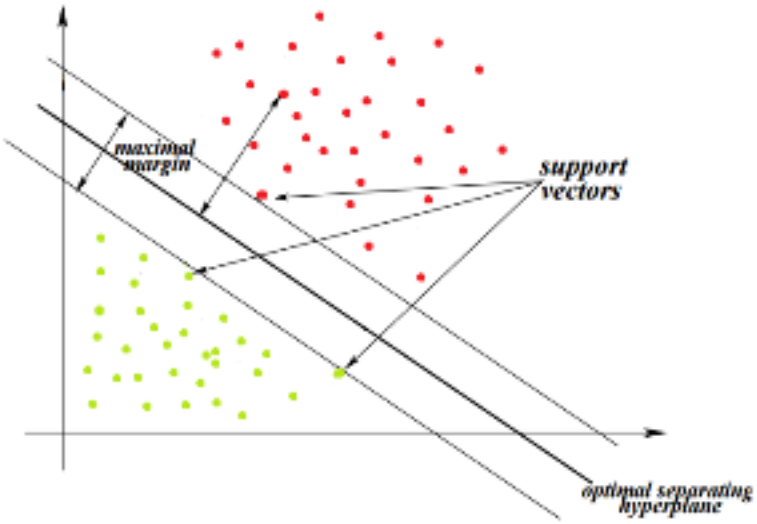

Figure 2.6: SVM Classifier [5]

Unsupervised model

Kmeans algorithm K-means clustering is one of the simplest and popular unsupervised machine learning algorithms .

For an attribute problem, each instance maps into a m dimensional space. The cluster centroid describes the cluster and is a point in m dimensional space around which instances belonging to the cluster occur.

The distance from an instance to a cluster center is typically the Euclidean distance though variations such as the Manhattan distance (step-wise distance) are common. As most implementations of K-Means clustering use Euclidean distance [28].

- A cluster refers to a collection of data points aggregated together .
- K is a target number, which refers to the number of centroids you need in the dataset.
- A centroid is the imaginary or real location representing the center of the cluster.

k-means clustering tries to find the similarity between the items and groups them into the clusters. K-means clustering algorithm works in three main steps.

- Select the k values.
- Initialize the centroids.

- Select the group and find the average.

The figure 2.7 shows the flowchart of kmeans clustering

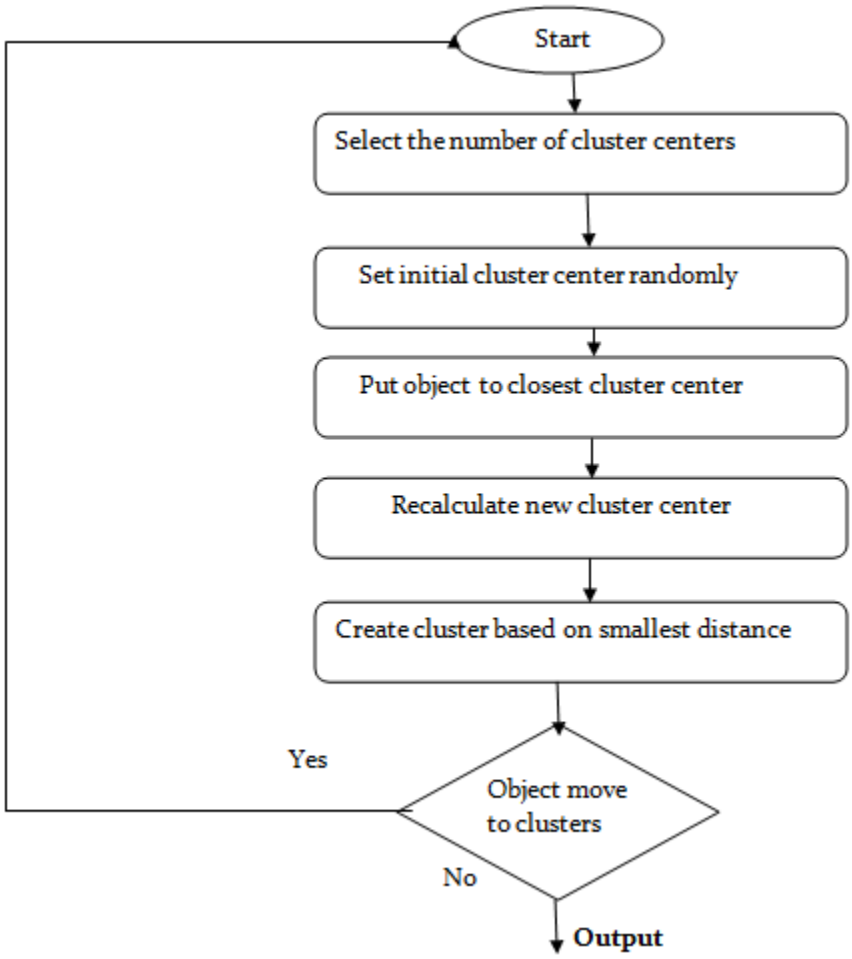

Figure 2.7: Flowchart of k-means clustering algorithm [6]

Hirarchical clustering Hierarchical clustering, or hierarchical cluster analysis, is a method that organizes similar objects into clusters. At the end of this process, we get a set of clusters where each cluster is distinct from the others, and the objects within each cluster share high similarity.

Hierarchical clustering starts by treating each individual object as a separate cluster. Then, it iteratively follows these steps:

- It identifies the two closest groups or clusters.

- It merges the two most similar clusters.

This process repeats until all clusters have been merged into larger clusters, forming a hierarchy of clusters.

2.4 Text Mining

Text Mining in Construction Text mining, also known as text analysis, is a process that involves extracting valuable information from unstructured text documents. In the construction industry, text mining plays a crucial role in analyzing various textual data sources such as contracts, project reports, emails, and more to enhance project management, identify risks, and improve stakeholder communication.

2.4.1 Applications of Text Mining

Contract Analysis

Text mining can be used to analyze construction contracts to identify key terms, obligations, and potential risks.

Project Reports

By applying text mining techniques to project reports, construction professionals can extract insights on project progress, issues, and performance indicators.

Email Communication

Analyzing email communications using text mining can help in understanding stakeholder interactions, project updates, and decision-making processes.

2.4.2 Benefits of Text Mining

Risk Identification

Text mining can assist in identifying potential risks and issues early in construction projects by analyzing textual data for warning signs and patterns.

Efficient Project Management

By automating the analysis of text documents, construction managers can streamline project management processes and make data-driven decisions.

Improved Communication

Text mining tools can enhance communication among project stakeholders by extracting and summarizing key information from textual sources.

2.5 Natural Language Processing

NLP stands for Natural Language Processing. It is a field of artificial intelligence that focuses on the interaction between computers and humans using natural language. NLP enables computers to understand, interpret, and generate human language, allowing for seamless communication between machines and humans.

2.6 Preprocessing Techniques

Tokenization

Tokenization, a fundamental preprocessing step, involves breaking down text into smaller units such as words or sentences. The process is influenced by the writing system and linguistic structure of languages, categorized as isolating, agglutinative, or inflectional.

Stop Word Removal

Stop words, commonly occurring but semantically insignificant words like 'and' or 'the,' are often removed during preprocessing. Their elimination reduces noise in the text data and improves system performance by focusing on more meaningful content.

Stemming Techniques

Stemming methods like affix removal stemmers or successor variety stemmers help identify the root forms of words by removing prefixes or suffixes. These techniques aid in reducing the size of the text data and improving the accuracy of information retrieval.

Lemmatization

Lemmatization involves reducing words to their base or root form, known as a lemma. This process helps in standardizing words to their dictionary form, making it easier to analyze and understand the text.

TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF is a statistical measure used to evaluate the importance of a term in a document relative to a collection of documents. It helps in identifying key terms that are significant in a specific document.

Part-of-Speech (POS) Tagging

POS tagging assigns grammatical categories (such as noun, verb, adjective) to words in a text. This information is valuable for understanding the syntactic structure of sentences.

2.7 Metaheuristic

Metaheuristics are techniques for solving problems that effectively search across a sizable solution space in search of ideal or nearly ideal answers. These tactics are used to direct the search process away from local optima and toward promising areas of the solution space. Many optimization issues, such as combinatorial, continuous, and multi-objective optimization, can be solved using metaheuristics. Genetic algorithms, simulated annealing, ant colony optimization, particle swarm optimization, and tabu search are a few examples of metaheuristic algorithms. When a problem needs a substantial amount of computer resources or is too difficult to be addressed using conventional optimization techniques, metaheuristics are frequently applied [29].

2.7.1 Metaheuristics and machine learning

In recent years, there has been growing interest in combining metaheuristics with machine learning techniques to further improve their efficiency and effectiveness in solving complex optimization problems. By integrating machine learning techniques into metaheuristics, it is possible to automate the parameter tuning process, improve the decision-making process, and learn heuristics from past solutions[7].

Machine learning-driven metaheuristics have proven to be successful across a range of optimization tasks, including hyperparameter tuning, feature selection, and portfolio optimization. Examples of such metaheuristics include deep reinforcement learning-based approaches, neural network-driven methods, and evolutionary algorithms integrated with machine learning components. These techniques have demonstrated promising results in tackling optimization problems.

Furthermore, machine learning methods find wide applications in various domains such as search engines, robotics, computer vision, finance, bioinformatics, and insurance, among others. With the growing availability of data and computing resources, machine learning-based metaheuristics are anticipated to become even more potent and efficient in addressing complex optimization challenges in the future.

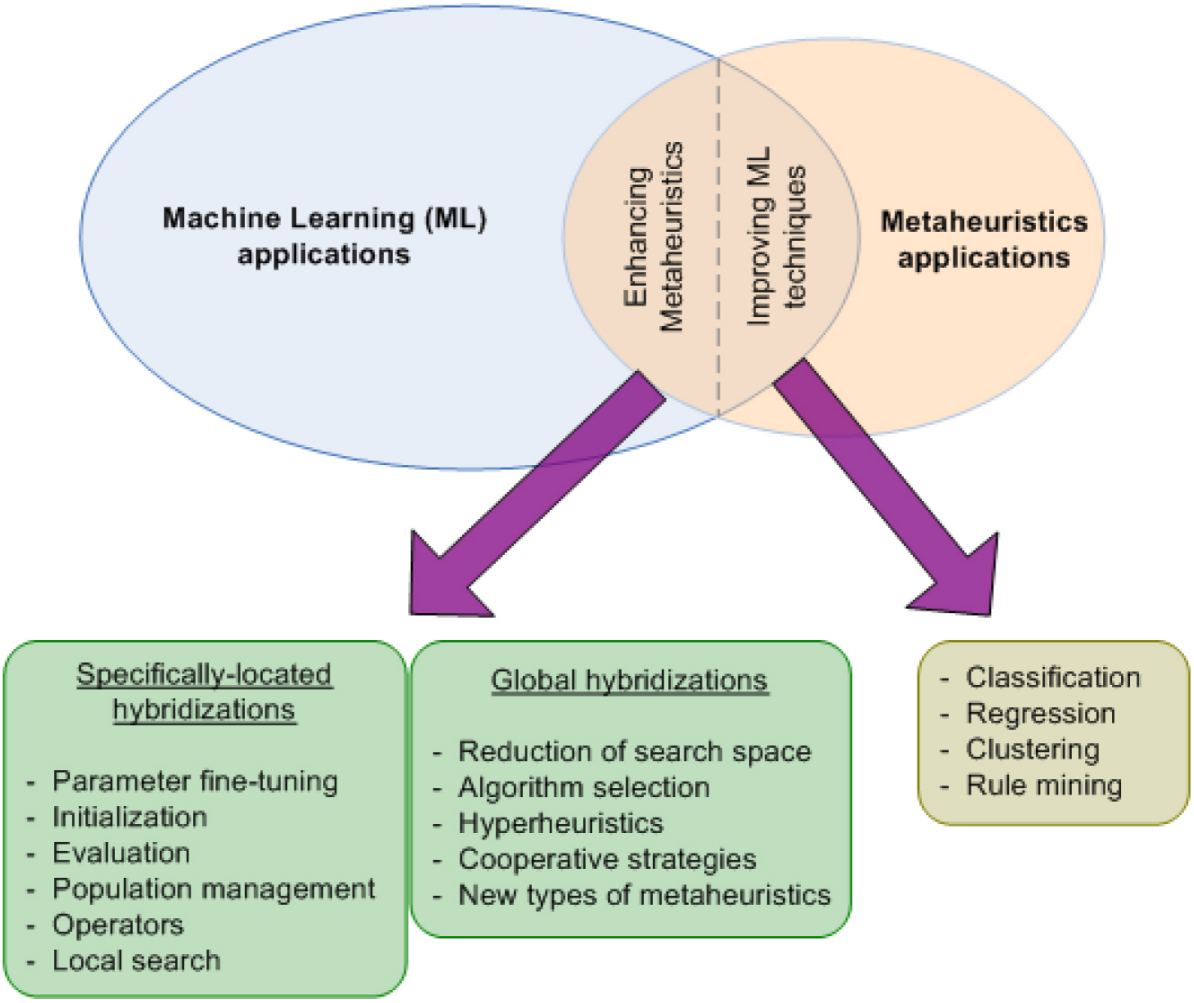

Figure 2.8: ML and Metaheuristics[7]

2.7.2 A Brief History

The history of metaheuristics as scientific methods for optimization can be traced back to the mid-20th century when heuristic approaches were used in various applications. The development of evolutionary algorithms marked a significant milestone, with Ingo Rechenberg and Hans-Paul Schwefel introducing evolutionary strategies in 1963 and L. J. Fogel et al. proposing evolutionary programming in 1966. J. Holland later pioneered genetic algorithms in the 1960s and 1970s, publishing a seminal book on the subject in 1975.[30]

The 1980s and 1990s witnessed notable advancements in metaheuristic algorithms.

Simulated annealing, inspired by the annealing process of metals, was introduced by S. Kirkpatrick et al. in 1983. The development of artificial immune systems by Farmer et al. in 1986 was another significant step. Glover pioneered the use of memory in metaheuristics with Tabu search in the 1980s, publishing a comprehensive book on the topic in 1997.

In 1992, Marco Dorigo presented his innovative work on ant colony optimization (ACO) in his PhD thesis. Genetic programming, introduced by John R. Koza in 1992, revolutionized computer programming. Particle swarm optimization was developed by James Kennedy and Russell C. Eberhart in 1995, followed by the vector-based differential evolution algorithm by R. Storn and K. Price in 1996-1997.[31]

Entering the 21st century, more exciting developments unfolded. The harmony search algorithm by Zong Woo Geem et al. in 2001 drew inspiration from music. Bacteria foraging and honey bee algorithms emerged in 2002 and 2004 respectively, with subsequent variations like the artificial bee colony algorithm. In 2008, the firefly algorithm was introduced, while 2009 saw the efficient cuckoo search algorithm by Xin-She Yang and Suash Deb, outperforming existing metaheuristics including particle swarm optimization [32] [33][34].

2.7.3 Metaheuristics operation

Metaheuristics operate according to a set of standard guidelines and procedures. An initial population or solution set is created first, frequently randomly or with the use of heuristics. This population represents potential answers to the current optimization issue. After that, an iterative process starts, with each iteration denoted by the terms generation or iteration. The metaheuristic algorithm assesses the fitness or quality of each population solution within each generation. The standard method for evaluating fitness is to use an objective function or a set of constraints that are unique to the optimization issue. Higher fitness values are regarded as superior or being nearer to the ideal solution. The method uses a variety of operators or transformations to provide new candidate solutions after evaluating fitness. These operators might use local search methods, mutation, or recombination. The goal is to explore the search space for new, possibly superior solutions.

The algorithm then chooses a subset of solutions to create the next set of solutions, frequently based on their fitness levels. Elitism, which preserves the best solutions, or stochastic selection techniques, like roulette wheel selection or tournament selection, may be used in this selection process.

Up until a termination condition is satisfied, the iteration process keeps going. The

number of generations, reaching a particular fitness threshold, or going over a computational time restriction can all be used as termination criteria.

Throughout the iterations, the metaheuristic algorithm maintains a memory or history of previously evaluated solutions. This memory can help guide the search process, avoiding revisiting already explored solutions or promoting diversification and intensification.[35][36]

Overall , Metaheuristics generally operate by creating an initial population, iteratively assessing fitness, applying operators to create new solutions, choosing the best solutions for the following generation, and continuing the process until a termination condition is satisfied. Memory enhancement and successful search space navigation are two benefits of memory use.[35]

2.7.4 Classification

Having explored the various classifications of metaheuristics, it's crucial to understand how these techniques are applied to tackle complex optimization problems. Metaheuristics offer flexible and effective approaches to navigate vast and often nonlinear solution spaces. By combining local and global strategies, these methods aim to find high-quality solutions while managing trade-offs between deep exploration and efficient exploitation. This balance is crucial for achieving optimal outcomes across diverse application domains, from engineering to operations research and artificial intelligence.

Local search and global search

Defining the sort of search strategy is one strategy. An improvement over straightforward local search algorithms is one kind of search method. The hill climbing algorithm, which is used to locate local optimums, is a well-known local search technique. Hill climbing does not, however, ensure that the world's best solutions will be discovered.

To enhance local search heuristic and locate better solutions, many metaheuristic theories were put forth. Simulated annealing, tabu search, iterated local search, variable neighborhood search, and GRASP are some examples of these metaheuristics. Both of these metaheuristics fall within the local search-based and global search categories.

Population-based metaheuristics are typically used for other global search metaheuristics that are not based on local search. Ant colony optimization, evolutionary computation, particle swarm optimization, genetic algorithm, rider optimization method, and others are examples of such metaheuristics.[37][38][39]

Single-solution vs. population-based

Metaheuristics can be categorized as single-solution or population-based. Single-solution approaches manipulate a single candidate solution iteratively, while population-based approaches maintain a population of solutions, allowing for exploration and exploitation. Population-based metaheuristics often exhibit better search capabilities but require additional computational resources.[37][40][41]

Hybridization and memetic algorithms

ybridization refers to the combination of different metaheuristic algorithms or techniques to create a more powerful and effective optimization approach. It leverages the strengths of multiple algorithms to enhance exploration and exploitation capabilities, ultimately improving the quality of solutions obtained. Memetic algorithms are a specific type of hybrid metaheuristics that combine global search strategies with local improvement heuristics to enhance solution quality and convergence speed.[42]

Parallel metaheuristics

Hybridization involves blending different metaheuristic algorithms or approaches to create a more powerful and effective optimization strategy. By combining various algorithms, hybridization aims to leverage the strengths of each to enhance both exploration and exploitation capabilities, ultimately improving the quality of solutions obtained.

Memetic algorithms represent a specific category of hybrid metaheuristics. They expedite convergence and enhance solution quality by integrating local improvement heuristics with global search techniques.

Nature-inspired and metaphor-based metaheuristics

Natural events or processes are used as inspiration for nature-inspired metaheuristics, which direct the search for the best answers. Examples include ant colony optimization, particle swarm optimization, and genetic algorithms. These algorithms use ideas like evolution, swarm behavior, and pheromone communication to simulate biological, physical, or ecological systems.

On the other hand, metaphor-based metaheuristics use metaphors from other areas to create optimization algorithms. They use concepts and tenets from domains unrelated to

their own, including social behavior, cultural development, or artistic processes. Metaphor-based metaheuristics provide novel approaches in optimization and the exploration of non-conventional solution spaces by providing distinctive viewpoints and creative problem-solving techniques.[8]

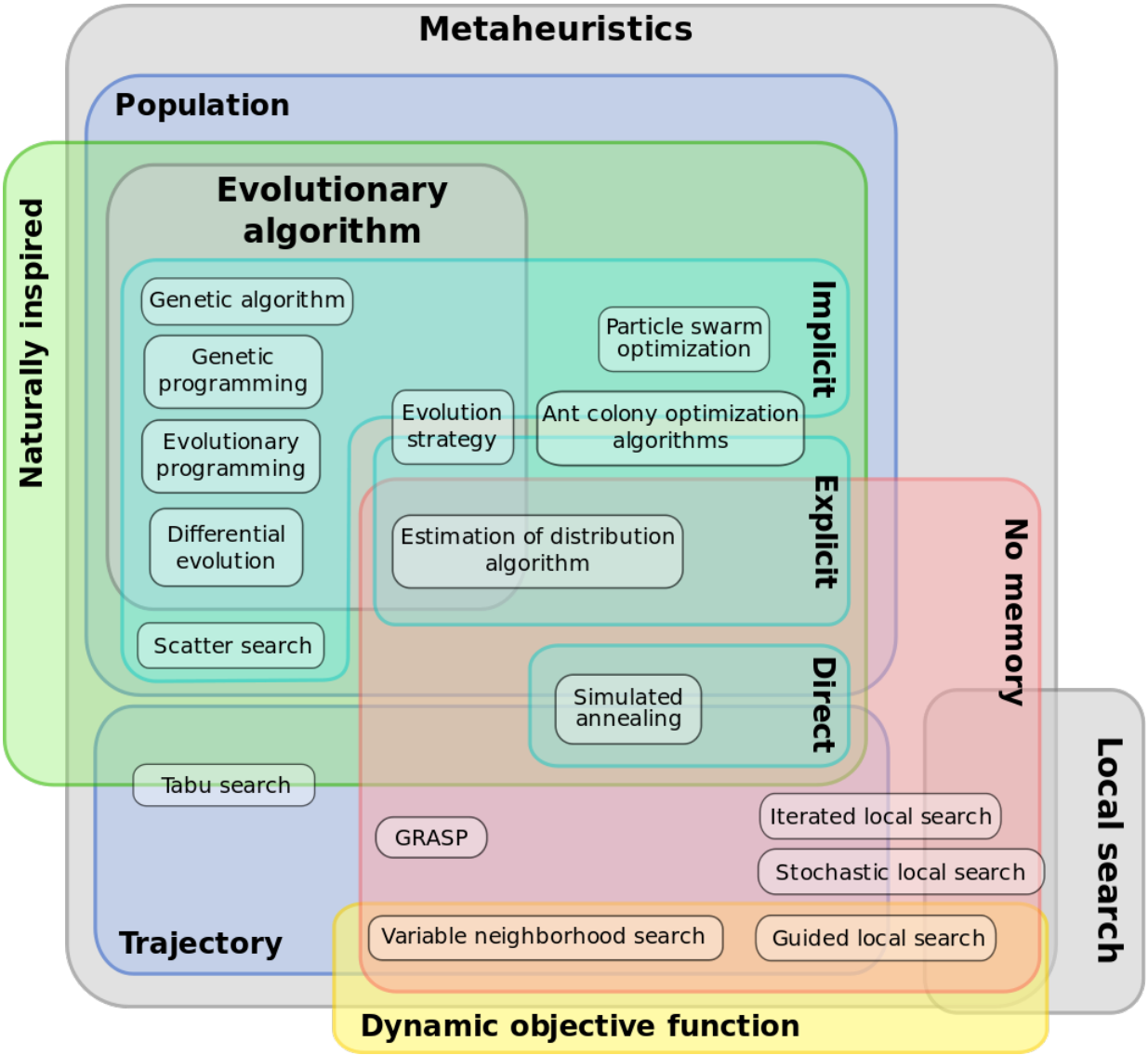

Figure 2.9: Euler diagram of the different classifications of metaheuristics[8]

2.7.5 Using machine learning for enhancing metaheuristics

The study of how machine learning methods have been applied to improve metaheuristics has been split into two sections for the sake of clarity. The first portion examines local-level hybridizations, while the second section examines global-level hybridizations. Each of them has been categorized by the appropriate topic [29].

2.7.6 Golf Optimization Algorithm (GOA)

Inspiration of GOA

Golf, an outdoor game or sport, unfolds on individual or team canvases, wielded by the adept manipulation of specialized clubs. The foundational tenets of this pastime dictate its essence—an artful journey of propelling a ball from its inaugural point towards a distant hole. This pursuit, executed through calculated swings and governed by a set of stipulations, encapsulates the essence of golf. Beneath this ostensibly straightforward surface, however, the game's regulations interpose complexities, engendering a heightened level of challenge. Central to this enterprise is the strategic finesse required to guide the golf ball into the awaiting hole. This strategic choreography, a manifestation of intellectual prowess, serves as a wellspring of inspiration for the conceptualization of a pioneering metaheuristic algorithm. The inception of the Golf Optimization Algorithm (GOA) derives its blueprint from this very strategy, seamlessly weaving its contours into a methodological framework. In the realm of the GOA, this strategic dance finds embodiment, its intricate steps delineated, and its conceptual underpinnings crystallized through rigorous mathematical modeling .[9] .

Strength of Golf Optimization Algorithm

The Golf Optimization Algorithm (GOA) demonstrates several strengths that contribute to its effectiveness in solving optimization problems:

- Innovative Approach: The GOA introduces a novel metaheuristic algorithm inspired by the game of golf, offering a unique perspective on optimization .
- Simulation of Golf Rules and Behavior: The GOA simulates the rules and behavior of players in the game of golf, providing a structured and intuitive framework for optimization .

- Efficiency: The algorithm has shown efficiency in solving optimization problems, as evidenced by its evaluation on fifty-two standard objective functions .
- Comparative Performance: The quality of results obtained from the GOA has been compared with ten well-known metaheuristic algorithms, showcasing its competitive performance .
- Real-World Applications: The GOA has been successfully applied to real-world engineering design problems and optimization challenges related to energy grid resilience, demonstrating its versatility and applicability .
- Mathematical Modeling: The GOA is mathematically modeled for exploration and exploitation phases, enhancing its precision and adaptability in optimization applications .

  These strengths collectively position the Golf Optimization Algorithm as a promising and effective tool for addressing a wide range of optimization challenges.

Initialization of GOA

The Golf Optimization Algorithm (GOA) is a population-based approach that randomly initializes the position of its members in the search space using Equation (2) . The population matrix X (Equation (1))represents individual GOA members, with each member evaluated for objective function values, compiled into a vector F (Equation (3)) . The best member is updated iteratively based on these evaluations . Initialization involves generating random values within specified bounds for problem variables, ensuring a diverse exploration of the search space .

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ x_{i,1} & \cdots & x_{i,d} & \cdots & x_{i,m} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,d} & \cdots & x_{N,m} \end{bmatrix}_{N\times m}, \tag{1}$$

$$X_i : x_{i,d} = lb_d + r \times (ub_d - lb_d), \tag{2}$$

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N\times 1}, \tag{3}$$

Mathematical Model of GOA

The algorithm's mathematical model intricately captures the strategic essence of golf gameplay through two distinct phases: exploration and exploitation.

Exploration Phase: Mimicking Player Shots Towards the Ball

The exploration phase, as depicted in Equations (4) and (5), mirrors a player's strong shot towards the ball, incorporating randomness and a parameter I to dictate movement dynamics towards or away from the hole. This phase emphasizes global search, where new positions for GOA members are calculated, with potential replacements based on improved objective function values.

$$X_i^{P1} : x_{i,d}^{P1} = x_{i,d} + r \times (B_d - I \times x_{i,d}) \tag{4}$$

$$X_i = \begin{cases} X_i^{P1}, F_i^{P1} < F_i \\ X_i, else, \end{cases} \tag{5}$$

Exploitation Phase: Precision Kicks for Local Search Optimization

In contrast, the exploitation phase, outlined in Equations (6) and (7), simulates precise kicks on the green to guide the ball into the hole. New positions are determined based on low-power shots, enhancing the algorithm's ability to navigate local search spaces with finesse. These equations elegantly blend the strategic finesse of golf with the optimization prowess of the GOA, striking a harmonious balance between exploration and exploitation strategies for effective problem-solving across diverse search landscapes.

$$X_i^{P2} : x_{i,d}^{P2} = x_{i,d} + (1 - 2r) \times \frac{lb_d + r \times (ub_d - lb_d)}{t} \tag{6}$$

$$X_i = \begin{cases} X_i^{P2}, & F_i^{P2} < F_i \\ X_i, & else, \end{cases} \tag{7}$$

In Figure 4.9 , the pseudo-code of the GOA algorithm is provided, and The figure 4.10 presents the flowchart of this algorithm .[9] .

Algorithm 1. Pseudocode of the GOA.

Start GOA.

1. Input the optimization problem information.
2. Set T (number of iterations) and N (number of GOA members).
3. For t = 1:T
4. Update best member of GOA as hole.
5. For i = 1:N
7. Phase 1:
8. Calculate new status of ith GOA member based on exploration phase of GOA using Equation (4).
9. Update ith GOA member using Equation (5).
10. Phase2: Exploitation
11. Calculate new status of ith GOA member based on exploitation phase of GOA using Equation (6).
12. Update ith GOA member using Equation (7).
13. end
14. Save best candidate solution so far.
15. end
16. Output best obtained solution.

End GOA.

Figure 2.10: Pseudo-code of GOA [9]

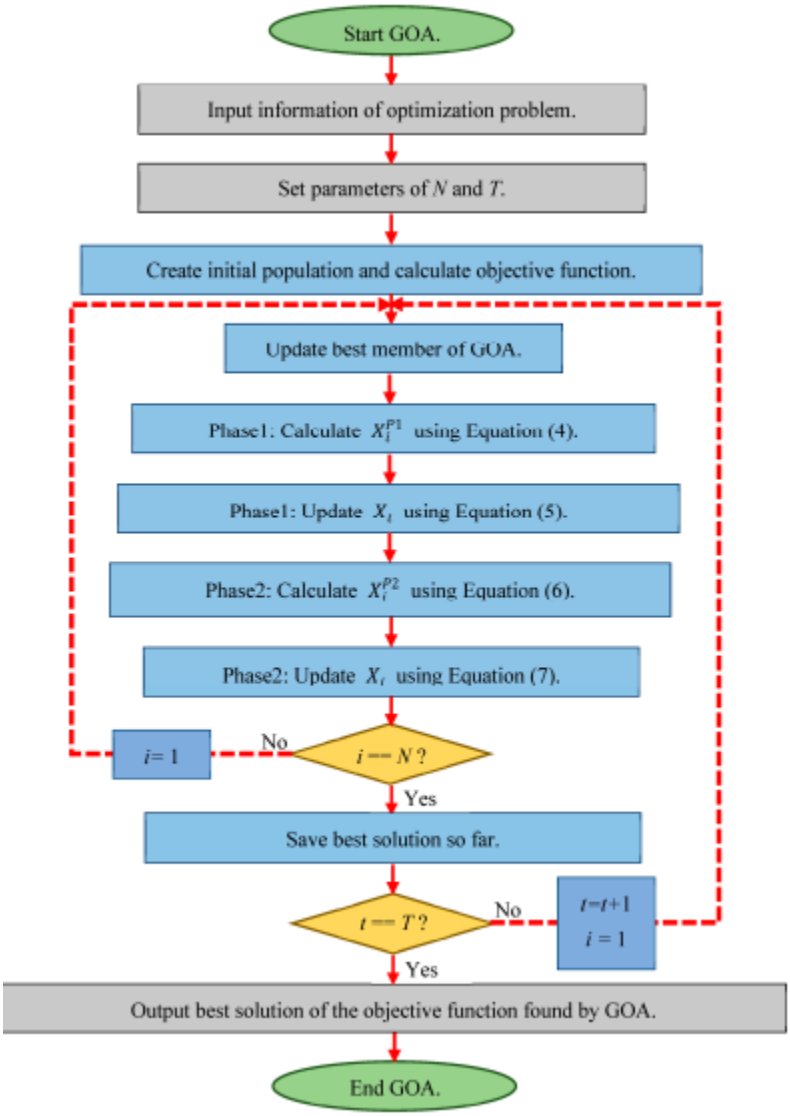

Figure 2.11: Flowchart of the GOA [9]

2.8 Conclusion

In summary, this section has presented an overview of the fundamental concepts that underpin our work. We covered online social networks, which are digital platforms facilitating social interactions, and false accounts, which refer to deceptive or fraudulent user profiles. Additionally, we discussed machine learning, which includes algorithms enabling computers to learn and make predictions, and metaheuristics, which are optimization algorithms aimed at finding optimal solutions to complex problems.

By grasping these concepts, we can delve deeper into research and applications at the intersection of online social networks, false information, machine learning, and metaheuristics.

Chapter 3

Fake News Detection on Social Networks : State of the art

3.1 Introduction

In this section, we review a literature study on methods for detecting fake news.

As social networks become increasingly popular, the number of individuals accessing various social media platforms is rising alarmingly. This surge has resulted in a significant amount of data being shared and stolen.

This chapter examines the literature on fake news detection using the analysis of behavioral features of those spreading false information. Various researchers have developed numerous approaches to identify false news. By comparing this new work with previous studies, this review also aims to evaluate its contribution.

Several techniques for identifying fake news focus on analyzing social networks and profiles to categorize characteristics or discrepancies that help differentiate between true and false information. Algorithms are employed to classify the retrieved data from profiles and posts, particularly to create a categorization for the detection of fake news.

1. **2023 Artificial intelligence applications in fake review detection**

   This article presents a comprehensive classification of research articles in the field of fake review detection using artificial intelligence. The study identified four main clusters of research articles: Cluster 1 focused on textual-based features, Cluster 2 on opinion mining, Cluster 3 on a supervised framework, and Cluster 4 on text classification in natural language processing. These clusters represent different approaches and methodologies employed in the detection of fake reviews. In terms of machine learning techniques, the research highlighted the use of semi-supervised ML models such as RCNN and RNN, along with other algorithms, for fake review detection. Textual-based features using ML classifiers emerged as a significant trend in applying AI to detect fake reviews. The study also emphasized the importance of addressing challenges such as evolving spammer tactics, real-time detection limitations, platform compatibility issues, and the need to differentiate between fake and honest reviews for more effective detection strategies[10] .

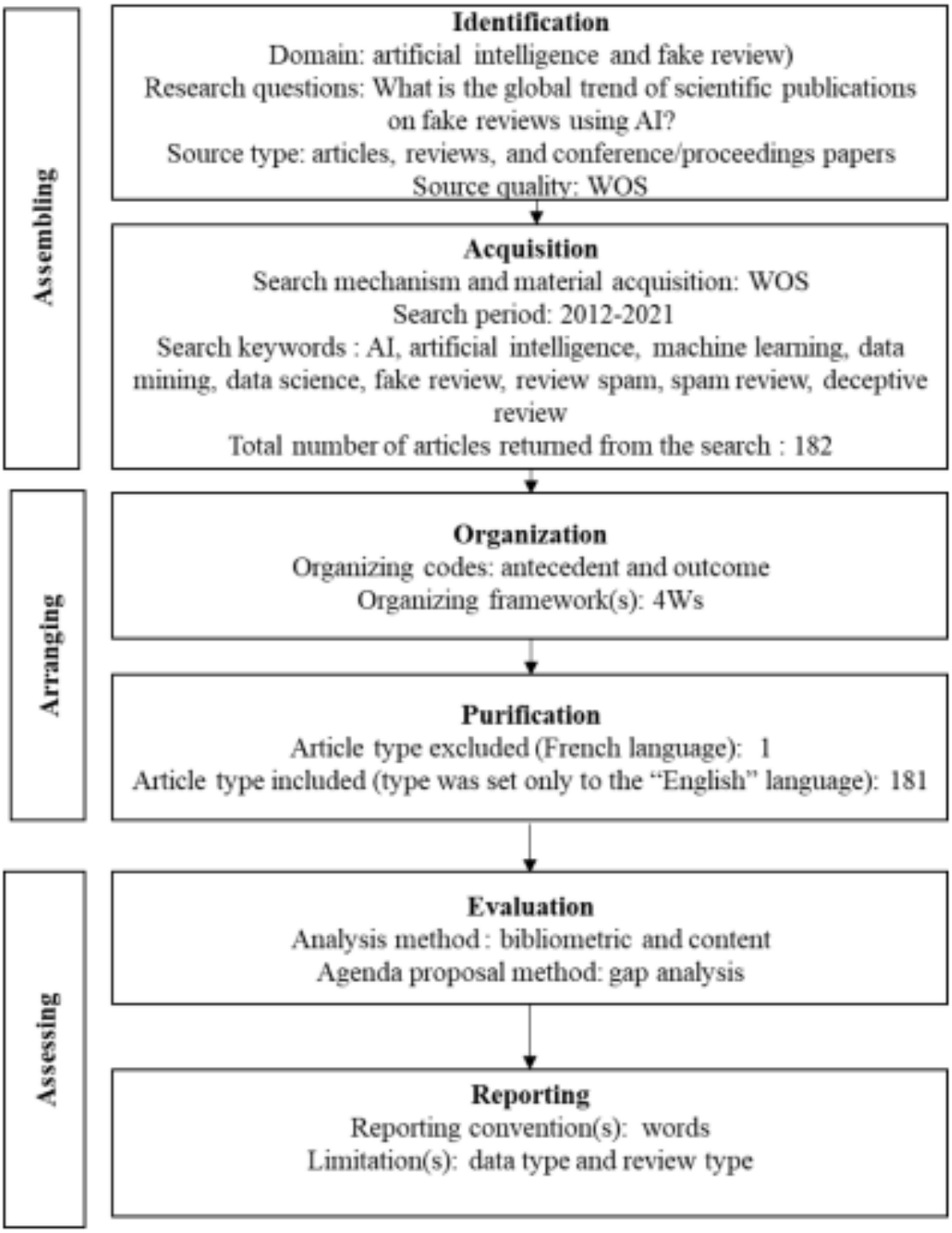

Figure 3.1: Application of the SPAR-4-SLR protocol. [10]

2. **2024 An overview of fake news detection: From a new perspective**

   In the realm of Online Social Networks (OSN), various Machine Learning (ML) techniques and algorithms are employed for fake news detection. One common ML algorithm used is Convolutional Neural Networks (CNN), which is effective in extracting features from textual data for classification tasks. Additionally, Support Vector Machines (SVM) are utilized for their ability to handle high-dimensional data and nonlinear relationships. Metaheuristic algorithms like Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) are applied to optimize model parameters and improve classification performance.

   When evaluating the results of fake news detection models, key metrics such as recall, accuracy, F1 score, and precision are commonly used to assess performance. Recall measures the proportion of actual positives that are correctly identified, accuracy determines the overall correctness of the model, F1 score balances precision and recall, and precision calculates the proportion of true positives among all positive predictions.

   In terms of datasets, popular benchmark datasets like Weibo1 and Weibo2 are frequently used for fake news detection tasks. These datasets contain a significant number of instances, events, and messages related to fake news propagation on social media platforms. Researchers leverage these datasets to train and evaluate their ML models for effective fake news detection. [11] .

Intention	Feature	Example
Mislead the Public	Special Symbol Features	The number of URLs
		The fraction of messages containing a URL
		Whether a message contains a personal pronoun in 1st, 2nd, or 3rd person
		Whether a message contains a question mark or exclamation mark
		The number of "@" tags in a message
Manipulate Opinions	Sentiment Features	The numbers of positive and negative emoticons used in a message
		Average sentiment score of a message
		Whether a message contains strong negative words
		Fraction of messages containing negative sentiment and positive sentiment
	Style Features	Style similarity of news messages
Attract User Attention	Topic Features	The fraction of hashtags (#)
		LDA-based topic distribution of a message
	Visual Features	Whether a message contains images and videos
		Whether a user has a profile image
		The time delay of an image
		Image distribution: clarity score, coherence score, etc.
		Whether an image matches the text
		The topic of an image
		The semantic of an image extracted by pretrained models
	Clickbait Features	Similarity between the headline and top sentences
		Informality and readability of a message
		Whether a message contains internet slang or swear words
		Whether a message uses repeated characters (e.g., ooh, aah, etc.)
GeneralFeatures	Temporal Features	The time difference between a repost and the original message
		Whether a message has periodic reposts/comments spikes
		The number of duplicated reposts/comments
	User Features	Number of friends, followers, and the ratio of followers and friends
		Whether a user is a VIP or a verified user
		Register time, client program type, location, organization, gender
		Whether social profiles in different social media are linked with each other
		Whether the profile of a user contains a description, URL, and location
		Ratio of messages containing event verbs Ratio of messages containing strong negative words
		The number of messages at posting time
	Other Linguistic Features	TF-IDF feature, part-of-speech tagging feature
		Bag-of-words, named entity recognizing feature

Figure 3.2: Features extracted based on intentions of fake news. [11]

3. **2023 A comprehensive survey of fake news in social networks: Attributes, features, and detection approaches**

   In the survey on fake news detection in online social networks (OSN), various machine learning (ML) algorithms and metaheuristics are employed to achieve accurate results. Some of the ML algorithms used include Support Vector Machine (SVM), Naïve Bayes, Logistic Regression, Random Forest, Long Short-Term Memory (LSTM), and Recurrent Neural Networks (RNN). Metaheuristic optimization techniques like Genetic Algorithm (GA) are also utilized to enhance the performance of the ML models. The results obtained from the application of these ML algorithms and metaheuristics in fake news detection include metrics such as recall, accuracy, F1 score, and precision. These metrics are crucial in evaluating the performance of the models in correctly identifying fake news instances. Additionally, the datasets used in these experiments contain a specific number of instances, which vary depending on the study and the scope of the research. Overall, the combination of ML algorithms, metaheuristics, and evaluation metrics provides a comprehensive framework for effectively detecting and classifying fake news in online social networks, contributing to the ongoing efforts to combat misinformation and promote online authenticity. [12] .

	Authors	Approaches used	Limitations	Future scope	Datasets
Support Vector Machine	(Alrubaian et al., 2016)	Short text entries on "growing" issues, and determine whether they are legitimate or not depending on information gained.	Linear and non-linear tweet features predict better accuracy.	Credibility analysis based on the time-sensitive and location-based approaches for better results and better reliability.	1,416,443 tweets by 489,330 unique users.
	(Prasetijo et al., 2017)	Character recognition uses intricate and multidimensional metrics.	Hoaxes from the websites on particular situations appear as linear and non-linear.	A voting process is based on the outcomes of queries that incorporate textual data and computational linguistics.	100 hoxe news from 200 websites.
	(Ahmed et al., 2018)	Characteristics based analysis of fake news.	Text classification along with review rating even predicts better accuracy.	Text features as slang words or filler words.	Reviews from Amazon.
	(Shu et al., 2020)	A wide range of geographical information and contextual issues are used for analysis.	Increase the No.of features for detecting fake news to predict more satisfactory accuracy.	Merging the Untrue repositories with front-end technologies including FakeNewsTracker.	FakeNewsNet Repository.
	(Meel et al., 2020)	To detect and evaluate, picture-graphic, textual, and emotional features are counted.	Already mentioned in the article.	There are now more labeled databases available. The participant's history of Tweets.	20015 News articles.
	(Yazdi et al., 2020)	Eliminating unnecessary attributes, reduce recognition time, and improves prediction efficiency.	Labelled and unlabelled characteristics are used to get the best results.	To increase accuracy in these groups, additional procedures are used.	BuzzfeedNews, BS Detector, & LIAR.
	(Probierz et al., 2021)	Balancing the quality of classification and data analysis in the detection of fake news.	Analyses of news titles with the entire news text.	Comparison of news title with entire news content.	ISOT fake news.
	(Choudhury and Acharjee, 2022)	Genetic algorithm through machine learning approaches.	Increase the textual features to obtain good accuracy.	To obtain an optimal solution on different datasets.	LIAR, Fake Job Posting.
Naïve Bayes	(Granik and Mesyura, 2017)	A simple artificial intelligence technique might yield effective results on a vital classification of erroneous information.	Increase the input data size to predict better results.	Since efficiency is low, apply additional approaches to this database to increase the ability to Repository bogus information.	BuzzFeed News.
	(Hiramath and GC, 2019)	The detection of false information is being explored on Durability, Recall, and Reliability.	Already there in the article.	They may employ additional methodologies to determine whether any more improvements.	Not Available
	(Abdulrahman and Awf, 2020)	It merely tracks the frequency of every term. Focused on headlines and content.	Already mentioned in the article, spreading fake news without text.	Pay attention to the headline, as well as the image, and the writing.	11000 news articles.

Figure 3.3: Machine learning approaches to fake news detection. [12]

4. **2020 A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities**

   The survey on fake news detection presents a comprehensive overview of various results and classifications in the field. It discusses the importance of interdisciplinary research in enhancing the accuracy and efficiency of fake news detection methods. The survey identifies traditional machine learning (ML) models as a key approach for classifying news cascades as true or fake. These models utilize supervised learning methods such as Support Vector Machines (SVMs), decision trees, naive Bayes, and random forests to analyze features extracted from news content at different language levels, including lexicon, syntax, semantic, and discourse. Additionally, the survey highlights the use of deep neural networks for news article classification based on cascades. By leveraging both traditional ML and deep learning techniques, researchers can develop robust models to combat the spread of fake news effectively. [13] .

Feature Group			PolitiFact data [Shu et al. 2018]				BuzzFeed data [Shu et al. 2018]			
			XGBoost		RF		XGBoost		RF	
			Acc.	F_1	Acc.	F_1	Acc.	F_1	Acc.	F_1
Non-latent Features	Lexicon	BOWs (f_s)	0.856	0.858	0.837	0.836	0.823	0.823	0.815	0.815
		Unigram+bigram (f_r)	0.755	0.756	0.754	0.755	0.721	0.711	0.735	0.723
	Syntax	POS tags (f_s)	0.755	0.755	0.776	0.776	0.745	0.745	0.732	0.732
		Rewrite rules ($r*$, f_s)	**0.877**	**0.877**	0.836	0.836	0.778	0.778	0.845	0.845
		Rewrite rules ($r*$, f_r)	0.749	0.753	0.743	0.748	0.735	0.738	0.732	0.735
	Semantic	LIWC	0.645	0.649	0.645	0.647	0.655	0.655	0.663	0.659
		Theory-driven [Zhou et al. 2019a]	0.745	0.748	0.737	0.737	0.722	0.750	0.789	0.789
	Discourse	Rhetorical relationships	0.621	0.621	0.633	0.633	0.658	0.658	0.665	0.665
	Combination	[Zhou et al. 2019a]	0.865	0.865	**0.845**	**0.845**	**0.855**	**0.856**	**0.854**	**0.854**
Latent Features		Word2Vec [Mikolov et al. 2013]	0.688	0.671	0.663	0.667	0.703	0.714	0.722	0.718
		Doc2Vec [Le and Mikolov 2014]	0.698	0.684	0.712	0.698	0.615	0.610	0.620	0.615

Figure 3.4: Multimodal Fake News Detection Models. [13]

5. **2019 A Survey on Fake News and Rumour Detection**

   The article provides a comprehensive survey on the detection of fake news and rumors in online social networks. It discusses various techniques and approaches proposed in the literature, highlighting the challenges and future directions in this evolving field. The lack of benchmark datasets is identified as a key issue, impacting the evaluation and comparison of detection methods. The study reviews the trends in research over the years, emphasizing the importance of data collection, feature extraction, and machine learning algorithms in detecting false information. The paper also addresses the interconnected nature of fake news and rumors, presenting insights on definitions, data sources, and detection techniques. Overall, the survey underscores the promising advancements in combating misinformation online while acknowledging the need for further research and improvements in this critical area. [14] .

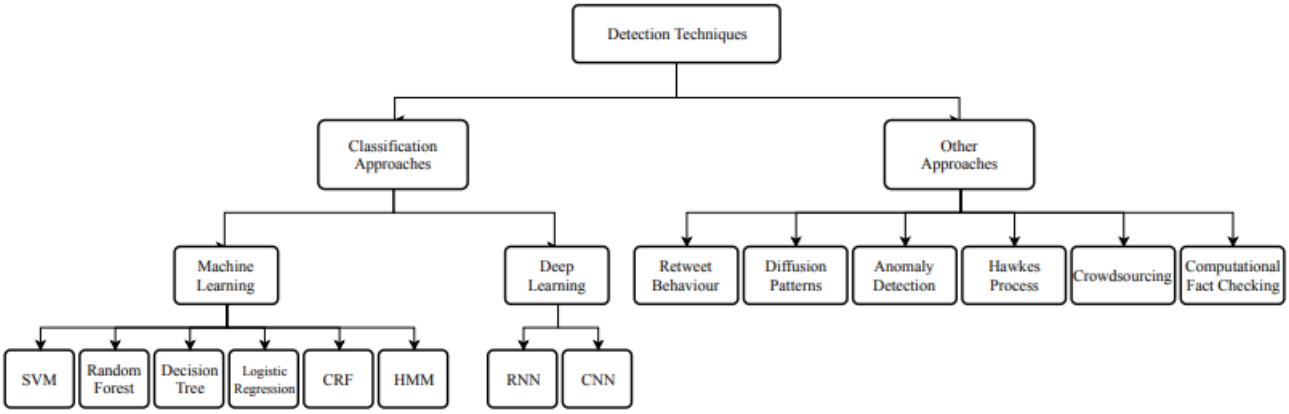

Figure 3.5: Different approaches to fake news and rumour detection proposed in the literature. [14]

6. **2018 A Survey on Fake News and Rumour Detection Techniques**

   In the FakeNewsNet dataset, researchers can explore various dimensions of fake news detection on Online Social Networks . They can leverage Machine Learning (ML) techniques, including algorithms such as Support Vector Machines (SVM), Logistic Regression (LR), Naive Bayes (NB), and Convolutional Neural Networks (CNN), to analyze news content and social context for fake news detection. By utilizing metaheuristic approaches, researchers can enhance the performance of ML models in classifying fake news. The evaluation metrics such as recall, accuracy, F1 score, and precision provide insights into the effectiveness of the detection models. The dataset contains instances from platforms like PolitiFact and GossipCop, enabling researchers to train and test their models on a diverse range of fake news samples. [15] .

Model	PolitiFact		GossipCop	
	Acc.	F1	Acc.	F1
SVM	0.580	0.659	0.497	0.595
LR	0.642	0.633	0.648	0.646
NB	0.617	0.651	0.624	0.649
CNN	0.629	0.583	0.723	0.725
SAF /S	0.654	0.681	0.689	0.703
SAF /A	0.667	0.619	0.635	0.706
SAF	0.691	0.706	0.689	0.717

Figure 3.6: Fake news detection performance on FakeNewsNet [15]

7. **Fake news detection in social media based on sentiment analysis using classifier techniques 2023**

   In the study on fake news detection in social media using sentiment analysis, various machine learning (ML) techniques were employed, including Naïve Bayes, passive-aggressive, and Deep Neural Network classifiers. These ML algorithms were utilized to classify missing data variables and extract useful features from the dataset. Additionally, metaheuristic approaches were explored to enhance classification accuracy. The results showed promising outcomes, with high recall, accuracy, F1 score, and precision values achieved. The dataset used in the study contained a substantial number of instances, providing a robust foundation for training and testing the ML models. [16] .

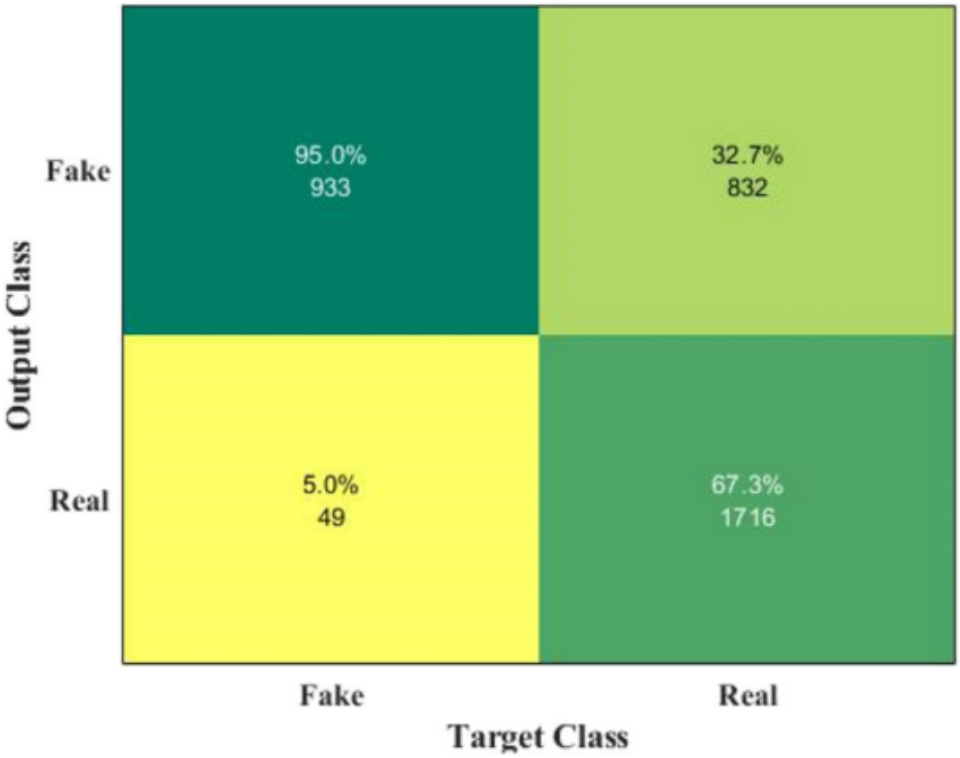

Figure 3.7: Confusion matrix[16]

8. **Automatic Detection of Fake News**

In this article, the authors address the task of automatic identification of fake news by introducing two new datasets: one obtained through crowdsourcing covering six news domains and another obtained from the web focusing on celebrities. They develop classification models utilizing lexical, syntactic, and semantic information, as well as text readability features. The machine learning classification is conducted using R with the caret and e1071 packages. The best performing models achieve accuracies comparable to human ability in spotting fake content. Various linguistic features such as Ngrams and Punctuation are extracted for building the detection models. The results show that different linguistic properties play a crucial role in explaining performance decreases across domains. The classification performance varies across news domains, with politics, education, and technology domains showing robustness against classifiers trained on other domains. Cross-domain analyses reveal a significant loss in accuracy when generalizing from crowdsourced data to celebrity news, indicating a bias towards truth in the predictions. The datasets contain a total of 70,975 words and 33,378 words, with varying numbers of instances per dataset. The experiments involve different machine learning algorithms and metaheuristics to achieve results in terms of recall, accuracy, F1 score, precision, and dataset instances. [17] .

Features (number of features)	Acc.	LEGITIMATE			FAKE		
		P	R	F1	P	R	F1
Punctuation (11)	0.70	0.67	0.77	0.72	0.73	0.63	0.68
LIWC - Summary (7)	0.65	0.66	0.61	0.63	0.64	0.68	0.66
LIWC - Linguistic processes (21)	0.64	0.64	0.63	0.63	0.63	0.64	0.63
LIWC - Psychological processes (40)	0.58	0.58	0.58	0.58	0.58	0.57	0.57
Complete LIWC (79)	0.67	0.68	0.66	0.67	0.67	0.68	0.67
Readability (26)	0.50	0.50	0.48	0.49	0.50	0.51	0.50
Ngrams (1378)	0.67	0.67	0.66	0.66	0.66	0.68	0.67
Syntax (1268)	0.67	0.67	0.68	0.67	0.68	0.66	0.67
All Features (2751)	0.73	0.73	0.72	0.72	0.73	0.74	0.73

Figure 3.8: Classification results for the Celebrity news data set.[17]

9. **Fake News Detection on Social Media: A Data Mining Perspective 2017**
   The article "Fake News Detection on Social Media" provides a comprehensive overview of fake news detection methods on online social networks (OSN). Various machine learning (ML) techniques such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and ensemble methods are employed to classify news articles as fake or real. The study evaluates the performance of these ML algorithms using metrics like recall, accuracy, F1 score, and precision. The dataset used for experimentation contains a significant number of instances to ensure robust evaluation. The results showcase the effectiveness of ML algorithms in accurately detecting fake news on social media platforms, with high precision and recall rates, ultimately contributing to the ongoing efforts to combat misinformation online. [18] .

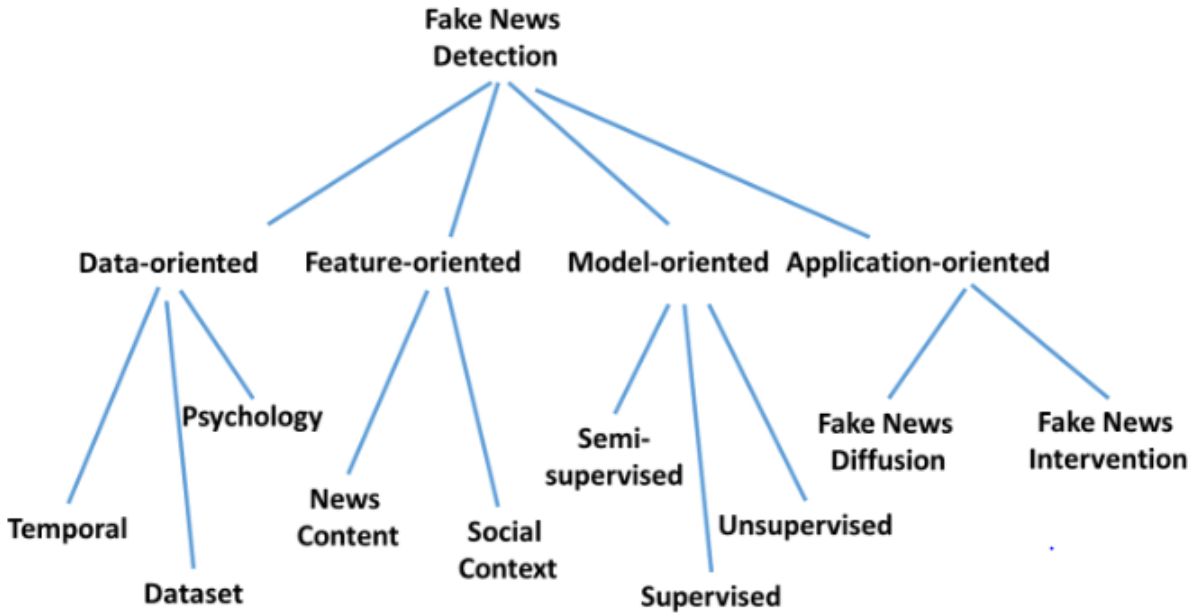

Figure 3.9: Future directions and open issues for fake news detection on social media.[18]

10. **Combating the infodemic: COVID-19 induced fake news recognition in social media networks**

The study focused on false news identification in Online Social Networks (OSN) using various types of Machine Learning (ML) models and algorithms. Traditional ML algorithms such as Logistic Regression, Random Forest, Naive Bayes, Support Vector Machine, and K Nearest Neighbors were initially employed, along with neural network models like LSTM, BiLSTM, GRU, and BiGRU with pre-trained BERT embeddings. The ensemble models and state-of-the-art transfer learning models were explored for classification. The results showcased an accuracy of 97% with a precision of 98% for both fake and real classes, along with an F1 score of 98% for each class. The dataset used for the study contained a specific number of instances, although the exact number is not specified in the provided excerpts. The research highlighted the superiority of ensemble and language models over other ML techniques in classifying fake news in the COVID-19 dataset, with BERT embeddings outperforming XLNet and ELMo. The study also emphasized the potential for future research to incorporate multimodal data and regional languages to enhance performance further. [19] .

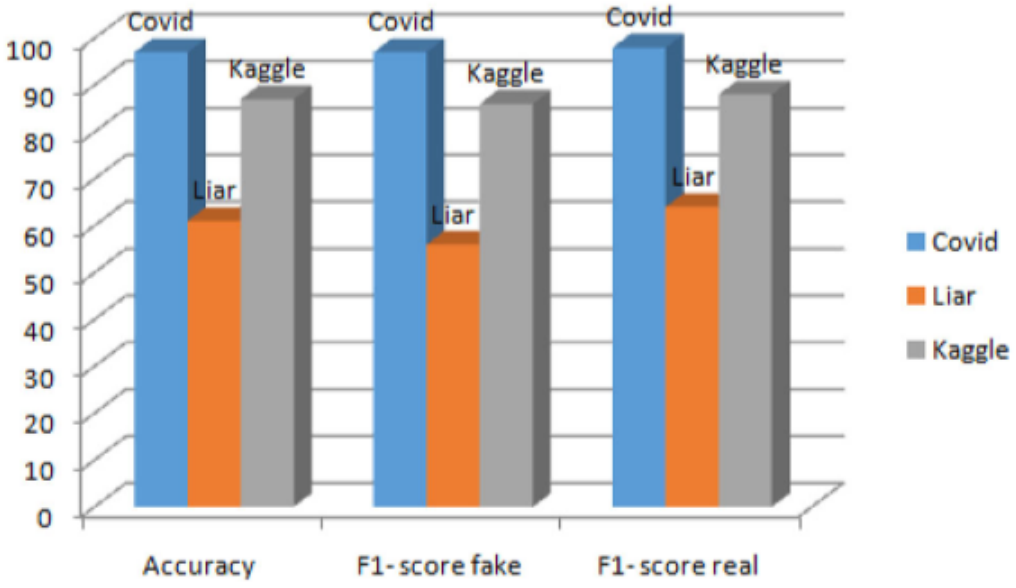

Figure 3.10: Performance analyses using different data set.[19]

11. **An Improved Classification Model for Fake News Detection in Social Media** The study "An Improved Classification Model for Fake News Detection in Social Media" focuses on detecting fake news on Online Social Networks (OSNs) using machine learning (ML) techniques. The proposed model employs a stack-ensemble approach with Support Vector Machine (SVM), Random Forest, and Recurrent Neural Network (RNN) as base learners, along with a Random Forest meta-classifier. The model was evaluated based on various metrics including recall, accuracy, F1 score, and precision. Results indicated a significant improvement in detection accuracy, with a 17.25% increase compared to the existing model. The sensitivity rate also improved by 15.78%, showcasing the model's enhanced capability in identifying fake news instances accurately. The dataset used in the study contained a specific number of instances, which were pre-processed and normalized for feature extraction and classification purposes. By leveraging ensemble ML algorithms and metaheuristics, the developed model demonstrated promising results in combating the spread of fake news on social media platforms.[20]

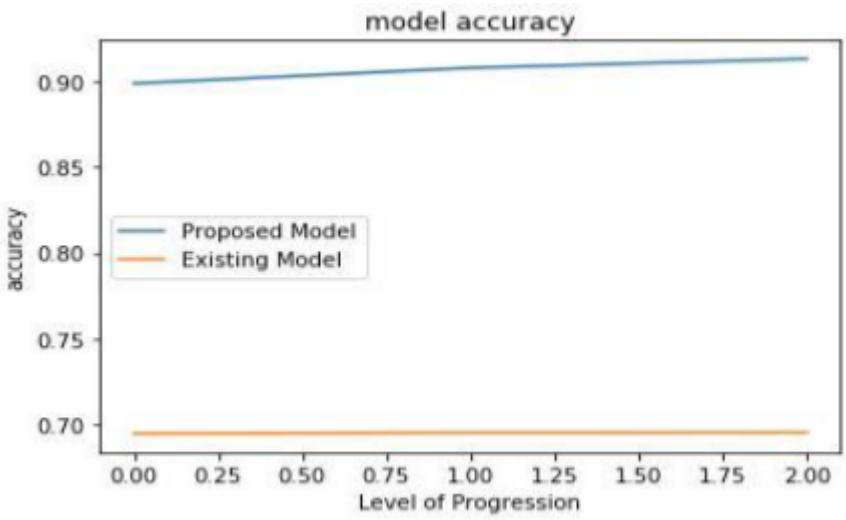

Figure 3.11: Detection Accuracy of the Existing and Proposed Model.[20]

3.2 Synthesize and discussion

We have summarized the literature presented above in a tabular format and provided a critique of the research conducted up to 2024. Our summary includes the key findings and limitations of the reviewed papers.

Reference	**OSN**	**ML Type**	**ML Algorithms**	**Meta-heuristic**	**Dataset-(Instances)**	**Results**
1		Supervised semi_supervised	CNN,RNN			
2		Supervised	CNN, SVM	GA,PSO		
3	Twitter Facebook	Supervised	SVM,NB, LR		10.22 m (COVID-19)	
4		Supervised	SVM,DT, NB,RF	GA,PSO		
5	Twitter	Supervised	SVM, DT,RF, RNN, CNN	Crowd sourcing, Anomaly Detection		F1-s(SVM)=0.84 preci(DC)=0.96, Preci(RN)=0.90, prec(RNN,CNN)=0.77
6		Supervised	SVM, NB, CNN, LR			PoliticalFact SVM(acc=0.58), GossipCop SVM(acc=0.49).

7		Supervised	NB,PA, DNN		44848	
8		supervised	svm		2131,2751	
9		Supervised	SVM ,KNN		51, 28001, 6194	
10		Supervised	LR,RF, NB,SVM, KNN		5600,5100	Recall=0.98, Accurcy=0.98, F1_score=0.98, precision=0.98
11		Supervised	SVM,RF, RNN		103212	Accurcy(RF)=0.819, precision(RF)=0.82

Table 3.1: Summarized of literature

The table 3.1 provides a detailed summary of various studies applying machine learning (ML) algorithms to data from online social networks (OSNs). Most studies focus on supervised learning, with some instances of semi-supervised learning. Supervised learning algorithms include convolutional neural networks (CNNs) and recurrent neural networks (RNNs), which are widely used for their ability to process sequential and image/text data. Support vector machines (SVMs) are frequently applied due to their effectiveness in high-dimensional spaces. Other commonly used algorithms are Naive Bayes (NB), logistic regression (LR), decision trees (DT), random forests (RF), deep neural networks (DNNs), and k-nearest neighbors (KNN), each chosen for their unique strengths in different scenarios.

Metaheuristic algorithms such as genetic algorithms (GA) and particle swarm optimization (PSO) are also utilized, particularly for tuning hyperparameters and enhancing model performance. These optimization techniques play a crucial role in improving the efficiency and accuracy of ML models.

The datasets used in these studies come from diverse sources, predominantly social media platforms like Twitter and Facebook. The size of these datasets varies significantly, from a few thousand instances to over ten million, as seen with a COVID-19 related dataset containing 10.22 million instances. This variety in dataset size and source reflects the wide applicability of ML algorithms to different scales and types of data.

Evaluation metrics such as accuracy, precision, recall, and F1 score are commonly employed to measure the performance of these models. For instance, one study achieved an F1 score of 0.84 using SVM and a precision of 0.96 using decision trees. Another study reported an accuracy of 0.58 for SVM on the PoliticalFact dataset and 0.49 on GossipCop, indicating that performance can vary based on the dataset and context. High-performance metrics, such as accuracy, recall, F1 score, and precision all at 0.98 in certain cases, underscore the potential of ML algorithms when properly tuned and applied.

3.3 Conclusion

In conclusion, this synthesis highlights the effectiveness of machine learning algorithms applied to online social networks like Twitter and Facebook. Supervised approaches are predominant, utilizing various algorithms such as CNN, RNN, and SVM. The results show impressive performance, with high precision and recall scores, despite some variations depending on the context and datasets. Overall, these studies demonstrate the significant potential of machine learning for analyzing and leveraging the massive data from social networks.

Chapter 4

Our contribution

4.1 Introduction

The initial phase of this study involved an in-depth analysis of various machine learning algorithms using a comprehensive collection of readily accessible datasets. The primary objective was to evaluate the performance of these algorithms based on a range of performance metrics, thus providing valuable insights.

The scope of the study was carefully defined, considering its objectives and available resources. The aim was to thoroughly understand the behavior of machine learning algorithms when applied to specific datasets, ultimately leading to meaningful conclusions. Subsequently, a detailed analysis and comparison of the results with relevant literature enabled the derivation of insightful findings.

4.2 Data processing

4.2.1 Dataset Collection :

Spanish Political Fake News :

The dataset titled ”Spanish Political Fake News” on Kaggle [1] consists of data related to fake news in the context of Spanish politics. It includes a comprehensive CSV file named ”D57000-complete.csv,” which contains various attributes and metadata regarding political fake news articles. The dataset contains 57231 records with the following columns: ID, Label, Titulo (Title), Descripcion (Description), and Fecha (Date).

The ’Label’ column has 33351 entries labeled as 1 (indicating real news) and 23880 entries labeled as 0 (indicating fake news). This dataset is intended for analysis and research into the characteristics and dissemination of fake news in Spanish political discourse.

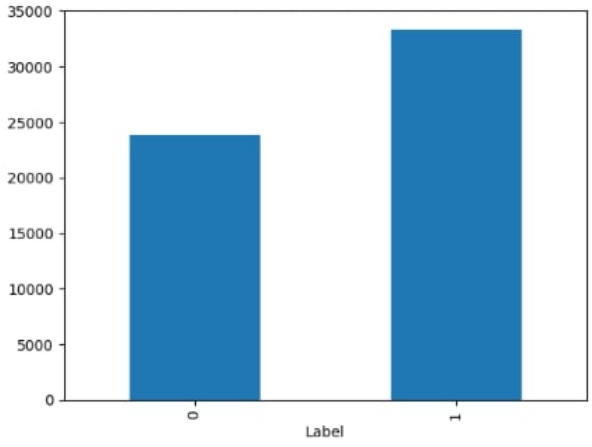

Figure 4.1: *Number of fake and real articles of Spanish Political Fake News dataset*

[1] https://www.kaggle.com/datasets/javieroterovizoso/spanish-political-fake-news?select=D57000_complete.csv

Attribut	Description
ID	Represents a unique identifier for each news item in the dataset.
Label	Indicates whether the news item is real (1) or fake (0). There are 33351 entries labeled as 1 (real news) and 23880 entries labeled as 0 (fake news).
Titulo	Contains the title of the news item. The title is usually a brief description that summarizes the main content of the news item.
Descripcion	Provides a more detailed description of the news item. Additional details about the topic or events mentioned in the news item are included here.
Fecha	Contains the date the news item was published. The date follows the format day/month/year.

Figure 4.2: *Attributs of Spanish Political Fake News dataset*

COVID-19 Fake News Dataset :

The COVID-19 Fake News Dataset, published by Abhishek Koirala on Mendeley Data, includes a collection of true and fake news related to COVID-19, gathered between December 2019 and July 2020. The dataset was sourced using Webhose.io and manually labeled. It contains three subcategories: false news, true news, and partially false news, with partially false and false news labeled as 0, and true news as 1. This dataset is intended for use in natural language processing, machine learning, and deep learning tasks.

The Dataset contains 3119 records with the following columns Unnamed: 0: An index column (integer). title: The title of the news article (mostly strings, one missing value). text: The content of the news article (strings). subcategory: The classification of the news article (strings indicating categories like ”false news,” ”true,” and ”partially false”). label: A binary label indicating the veracity of the news article (0 for false, 1 for true).

This dataset is valuable for researchers aiming to develop algorithms and models to combat the spread of fake news, particularly in the context of the COVID-19 pandemic

Figure 4.3: *Distribution of Article Categories in the COVID-19 Fake News Dataset*

Fake News Dataset Combined Different Sources :

The dataset titled "Fake News Dataset Combined Different Sources" on Kaggle[2] is a comprehensive collection designed to aid in the study and detection of fake news. This dataset aggregates news articles from multiple sources, providing a wide range of examples of fake news. It includes various features such as article titles, text, and Ground Label, which are crucial for building and training machine learning models to detect and classify fake news. The dataset also includes ground truth labels, with 42159 instances labeled as '0' (fake news) and 26886 instances labeled as '1' (real news). This dataset is particularly useful for natural language processing (NLP) tasks and can support the development of algorithms aimed at identifying unreliable news articles.

[2]https://www.kaggle.com/datasets/mohammadaflahkhan/fake-news-dataset-combined-different-sources

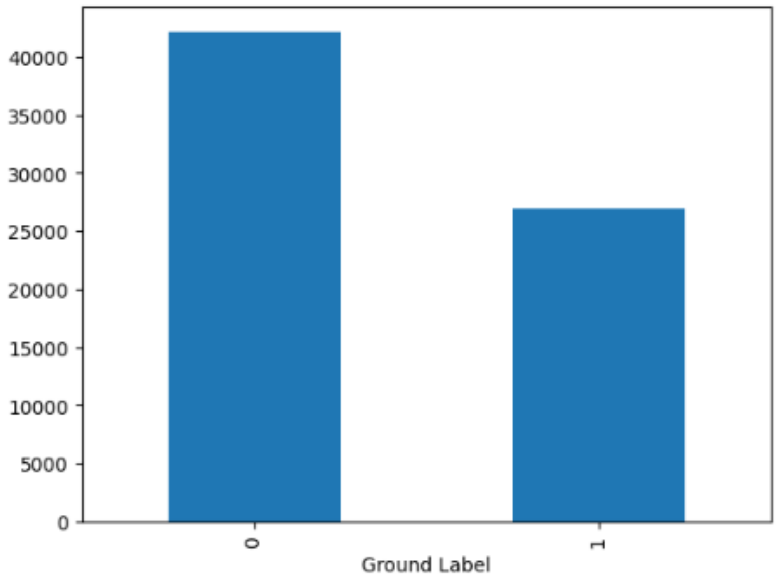

Figure 4.4: *Number of fake and real articles of Fake News Dataset Combined Different Sources*

Attribut	Description
ID	Represents a unique identifier for each news article in the dataset.
Title	Contains the title of the news article. The title is usually a brief description that summarizes the main content of the news article.
Text	Contains the full text of the news article. This feature includes the detailed content of the article.
Ground Label	Indicates whether the news article is fake (0) or real (1). There are 42159 instances labeled as 0 (fake news) and 26886 instances labeled as 1 (real news).
Source	The dataset aggregates news articles from multiple sources, providing a wide range of examples of fake news.

Figure 4.5: *Attributs of fake and real articles of Fake News Dataset Combined Different Sources*

BanFakeNews :

The "BanFakeNews" dataset on Kaggle[3] is designed for detecting fake news and includes 58478 news articles labeled as authentic or fake. It consists of four main files:

"Authentic-48K.csv" with 48678 authentic articles.

[3]https://www.kaggle.com/datasets/cryptexcode/banfakenews

"Fake-1K.csv" with 1299 fake articles.

"LabeledAuthentic-7K.csv" with 7202 labeled authentic articles.

"LabeledFake-1K.csv" with 1299 labeled fake articles.

Each article is described by several columns: articleID, domain, date, category, headline, content, and label (1 for authentic, 0 for fake). The label distribution shows 55880 authentic articles and 2598 fake articles, providing a solid foundation for machine learning projects aimed at identifying misinformation.

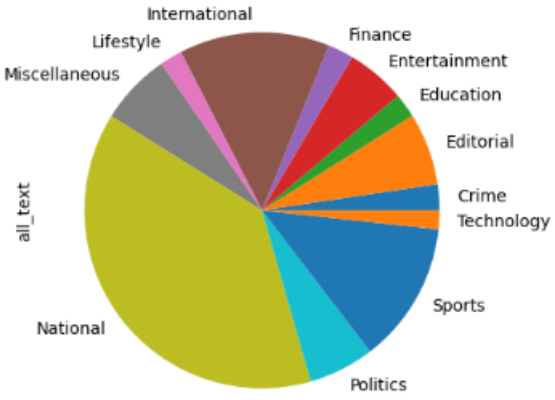

Figure 4.6: *Distribution of Article Category in the BanFakeNews*

Column Title	Description
articleID	ID of the news
domain	News publisher's site name
date	Category of the news
category	Category of the news
headline	Headline of the news
content	Article or body of the news
label	1 or 0 . '1' for authentic '0' for fake

Figure 4.7: *Attributs of Authentic-48K.csv and Fake-1K.csv*

Column Title	Description
articleID	ID of the news
domain	News publisher's site name
date	Published Date
category	Category of the news
source	Source of the news. (One who can verify the claim of the news)
relation	Related or Unrelated. Related if headline matches with content's claim otherwise it is labeled as Unrelated
headline	Headline of the news
content	Article or body of the news
label	1 or 0 . '1' for authentic '0' for fake
F-type	Type of fake news (Clickbait, Satire, Fake(Misleading or False Context))

Figure 4.8: *Attributs of LabeledAuthentic-7K.csv, LabeledFake-1K.csv*

Fake-and-Real-News-Dataset :

The "Fake and Real News Dataset" on Kaggle [4]contains two CSV files: "Fake.csv" and "True.csv". This dataset consists of a total of 44898 news articles. Each file includes the following columns:

title: The headline of the news article.

text: The main content of the article.

subject: The category or topic of the news article.

date: The publication date of the article.

label: A binary indicator where 1 denotes real news (23481 entries in "True.csv") and 0 denotes fake news (21417 entries in "Fake.csv").

[4]https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset/data?select=True.csv

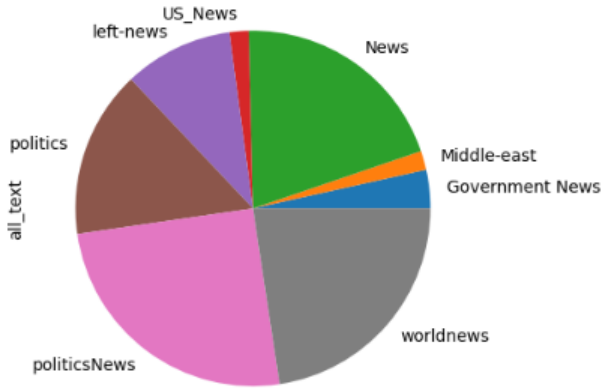

Figure 4.9: *Distribution of subject of fake-and-real-news-dataset*

Fake News Detection Datasets :

The "Fake News Detection Datasets" on Kaggle [5]contains two CSV files: fake.csv and true.csv. Each file has the following columns:

title: The headline of the news article.

text: The main content of the article.

subject: The category or topic of the news (e.g., politics, world news).

date: The publication date of the article.

The dataset is labeled with 0 for fake news and 1 for real news, containing 23481 fake and 21417 real news entries. The total size of the dataset is 44898 entries. This dataset is designed for training machine learning models to detect fake news.

[5]https://www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets

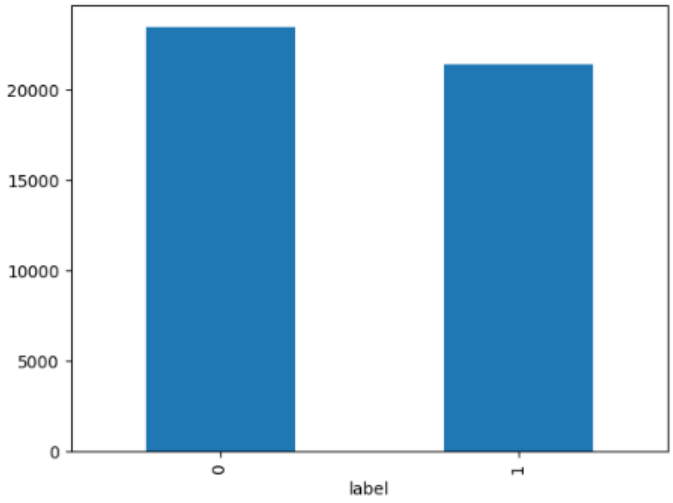

Figure 4.10: *Number of fake and real articles of Fake News Detection Datasets*

Fake news detection data :

The "Fake News Detection" dataset available on Kaggle [6] is a comprehensive resource designed to aid in the development and evaluation of algorithms for detecting fake news. The dataset, totaling 26000 news articles, is divided into training and test sets. Each article is uniquely identified and includes columns for the title, author, and full text. In the training set, a binary label indicates whether the article is fake (0) or real (1). The dataset is notably imbalanced, with 20800 articles labeled as fake and 5200 labeled as real. This dataset is ideal for training and testing machine learning models, natural language processing tasks, and research on misinformation.

	Real	**Fake**	**Total**
Records	20800	5200	26000
Percentage	80%	20%	100%

Table 4.1: Fake news detection data description

[6] https://www.kaggle.com/datasets/sathiyaak/fake-news-detection-data

Attribute	Description
id	Represents a unique identifier for each news article in the dataset.
title	Contains the title of the news article. Typically, it provides a brief description summarizing the main content of the article.
author	Indicates the author of the news article, if available.
text	Contains the full text of the news article. This field includes detailed content and information presented in the article.
label	Represents a label indicating the category of the news article, such as fake news or real news. This label is crucial for classifying articles in the dataset.

Figure 4.11: *Attributs of fake and real articles of Fake News Detection Datasets*

Fake news Detection data:

The dataset "Fake News Detection Data" on Kaggle [7]is designed for identifying fake news. It includes 20800 samples, with 10413 labeled as fake news (label = 1) and 10387 labeled as real news (label = 0). The dataset consists of four columns:

id: Unique identifier for each news article.

title: The title of the news article.

author: The author of the news article.

text: The full text of the news article.

label: The label indicating whether the news is fake (1) or real (0).

	Real	**Fake**	**Total**
Records	10413	10387	20800

Table 4.2: Fake news Detection data

Bangla And English Fake News Detection DataSet :

The "Bangla and English Fake News Dataset" on Kaggle[8] contains a total of 10000 articles, each labeled as either fake news or a genuine news story. Each article is represented by three columns:

[7]https://www.kaggle.com/datasets/athirakaladharan/fake-news-detection-data

[8]https://www.kaggle.com/datasets/sadikaljarif/bangla-and-english-fake-news-dataset

title:the article's title

text:the full content of the article

lebel: a label indicating whether the article is true or false

with 0 for false and 1 for true),The labels are balanced, with 5000 articles classified as false and 5000 as real.This dataset is particularly relevant for natural language processing (NLP) projects aimed at detecting fake news.

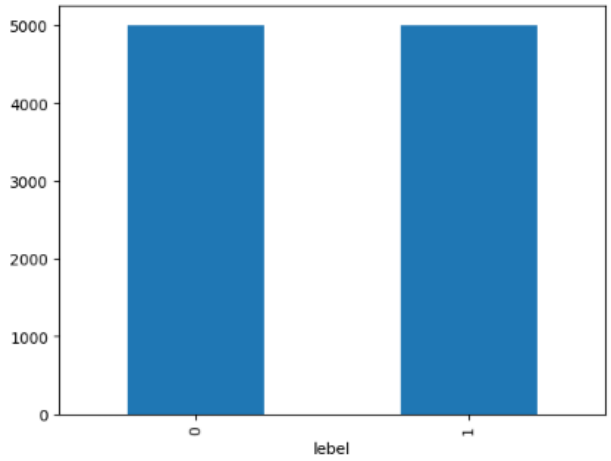

Figure 4.12: *Number of fake and real articles of Bangla And English Fake News Detection DataSet*

Detection of Fake News :

The "Detection of Fake News" dataset from Kaggle[9] contains a total of 6335 news articles, each labeled as either "FAKE" (false) or "REAL" (real). The data consists of four columns:

Unnamed: 0: A unique identifier for each article.

title: The title of the article.

text: The full text of the article.

label: The label indicating whether the article is false ("FAKE") or real ("REAL").

The labels are nearly balanced, with 3171 articles classified as false and 3164 as real. This dataset is essential for classification tasks and fake news detection, offering a good balance for training and testing machine learning models.

[9]https://www.kaggle.com/datasets/abhayku2002/detection-of-fake-news

	Real	Fake	Total
Records	3164	3171	6335
Percentage	49.94%	50.05%	100%

Table 4.3: Detection of Fake News dataset description

news.csv :

The "Fake News Detection" dataset on Kaggle [10]consists of 6335 news articles, each labeled to indicate whether it is fake or real. The dataset is structured with four columns:

Unnamed: 0: A unique identifier for each article.

title: The title of the article, providing a brief summary of the content.

text: The full text of the article, offering a comprehensive view for analysis.

label: A binary label where 0 indicates a fake article and 1 indicates a real article. The dataset is almost perfectly balanced, containing 3171 fake articles and 3164 real articles. This balance makes it particularly useful for training and testing machine learning models aimed at detecting fake news.

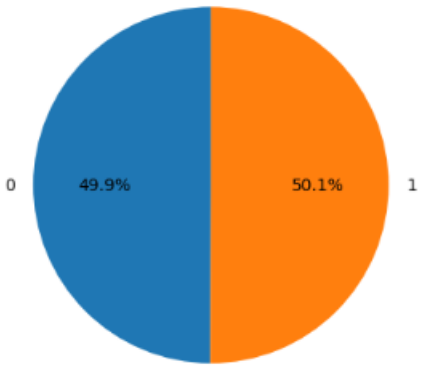

Figure 4.13: *Distribution of fake and real articles of news.csv DataSet*

4.2.2 Dataset preprocessing :

While a wealth of information is collected from various sources like the internet, surveys, and tests, this data is often imperfect. It can be corrupted with noise, contain missing values, or be in a format that's difficult to analyze directly. Data preprocessing addresses

[10]https://www.kaggle.com/datasets/phangud/newscsv

these issues. It's a crucial step in data analysis and machine learning, involving a series of techniques that transform raw data into a usable format.

We evaluated several classification algorithms before working with the dataset. We also carefully considered the types of numerical features. Additionally, the numerical aspects of other categorical features were modified.

Given the large number of attributes in the dataset, we prioritized identifying the most important features for our model. This involved removing attributes like "Categorie" that lacked statistical significance.

Missing Value Treatment :

Some of the provided datasets had no missing values, while others did. Missing data can result from various real-world issues and can be addressed through deletion or imputation. The presence of missing values limits the available data for analysis, reducing the study's statistical power and the reliability of its conclusions.

Cleaning :

Missing values can appear in a dataset due to various real-world issues and can be handled either through deletion or imputation. Missing values reduce the amount of data available for analysis, thereby decreasing the study's statistical power and ultimately affecting the validity of its findings. If a row in the dataset lacks values, it may be excluded from the table. Missing values for numerical attributes can be replaced with an empty string (' ') or with a fixed string, such as "Description not available", Here is a more detailed explanation of the cleaning steps:

Missing Value Detection:

missing_values = data.isnull().sum() - This function counts the number of missing values in each column of the DataFrame.

Displaying Missing Values:

Display the number of missing values for each column to identify where the missing values are located.

Missing Value Imputation:

The fillna method in pandas is commonly used to replace missing values with a specified value (e.g., data = data.fillna(")).

4.2.3 Max_feauture

We use the max_feature parameter when we want to limit the number of features used in the dataset, ensuring that only the most significant features are included in the analysis.

Correlation matrix

In this section, we used the correlation matrix after preprocessing the data and working with a maximum of 12 features. The correlation matrix was employed to detect the highest correlation between the features and the class. We use the correlation matrix for showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables, helping to summarize data, serve as input into more advanced analysis, and act as a diagnostic for advanced analyses. This process is done to understand the relationship between two variables and the strength of the association between them. By calculating the correlation matrix, we concluded the absence of high multicollinearity between the variables, indicating that the variables are not strongly correlated with each other. This is crucial because low multicollinearity facilitates the interpretation and robustness of statistical models.

4.2.4 Training fake news detection models :

Data Classification :

- We applied both classification types (supervised and unsupervised learning) to compare their results.
- Using pre-categorized training datasets, classification in machine learning programs leverages a wide range of algorithms to classify future accounts as either fake or real.
- We divided the data into two sets: a training set and a testing set.
- The training set is presented to our model, allowing it to learn from the data.
- The train-test split procedure is used to estimate the performance of machine learning algorithms when they make predictions on data not used to train the model.
- The data in the testing set is withheld from the model, and after training, the testing set is used to test its accuracy. The training set includes both features and

the corresponding labels, while the testing set only has the features, requiring the model to predict the corresponding labels.

- We used a test-train split of 30%-70% to compare the performance of machine learning algorithms for the predictive modeling of fake detection problems.
  - Train Dataset: Used to fit the machine learning model.
  - Test Dataset: Used to evaluate the fitted machine learning model.
- The objective is to estimate the performance of the machine learning model on new data, which was not used to train the model.

4.2.5 Parameters tuning

For every model, certain parameters were selected and provided with a range of possibilities. These parameters are the ones that have high impact towards detecting the illegitimate accounts and learning rate. This will then be implemented within bio-inspired algorithms.[21]

4.2.6 Testing fake news detection models

Machine learning models

In order to use the machine learning methods outlined above In order to assess different-machine learning algorithms, we created the test scenario using Python. The modelsare used to investigate the empirical link between the features and the probability of anillegitimate account. Since the goal of this study is to explore existing datasets of falseaccounts using a variety of statistical techniques and run machine learning algorithms onthem, it is a quantitative case study.

In our comparative analysis, we have employed several supervised models, including Naive Bayes, K-Nearest Neighbors, Decision Tree, Random Forest, and Gradient Boosting. Additionally, we have incorporated the K-Means model as an unsupervised technique.

To apply the machine learning methods discussed earlier and evaluate diverse machine learning algorithms, we set up a testing scenario using Python. The models were employed to examine the empirical association between the features and the likelihood of fake news.

Given that the aim of this study is to explore existing datasets of fabricated news utilizing a range of statistical approaches and apply machine learning algorithms to them, it represents a quantitative case study focusing on fake news analysis.

Implementation of Algorithm

- Step 1: Load and import the datasets
- Step 2: Clean the data by filling the missing values
- Step 3: Divide the all dataset in two parts: Test dataset and Train dataset
- Step 4 : Apply various machine learning techniques
- Step 5: Create the confusion matrix for each technique
- Step 6: Calculated the values of evaluation parameters of each techniques
- Step 7: Contrast the evaluation metrics across techniques and analyze the outcomes

Parameter Tunning

1. Supervised algorithms :

We have applied six supervised learning techniques: Naive Bayes, Decision Tree, Support Vector Machine, K-nearest neighbors algorithm, Gradient Boosting, and Random Forest Machine.

For K-nearest neighbors algorithm, hyperparameter tuning has been performed on the number of neighbors to consider in the model (k=3/k=5/k=7/k=9).

2. Unsupervised algorithms :

For unsupervised algorithms, we have implemented the K-Means model in its canonical form. In this approach, we provide the algorithm with the number of clusters, which is set to 2. The algorithm then automatically selects the initial centroids and iteratively refines them to partition the dataset into two clusters.

4.2.7 Chosen performance evaluation metrics

The detection of fake news can be evaluated using various performance measures, such as the F1 score, confusion matrix, and recall. In our study, we have used accuracy (ACC), F1

score, recall, precision, and entropy as performance metrics. Additionally, the confusion matrix is utilized to visualize the detection of fake news for the models.

- TP = True Positives, when our model correctly classifies the data point to the class it belongs to.
- FP = False Positives, when the model falsely classifies the data point.
- TN = These are the cases where the predicted "No" actually belonged to class "No".
- FN = These are the cases where the predicted "No" actually belonged to class "Yes".[43]
- Precision measures the model's ability to correctly classify values. It is calculated by dividing the number of correctly classified instances by the total number of instances classified as that specific class.

  Recall, on the other hand, measures the model's ability to predict positive values correctly. It answers the question, "How often does the model predict the correct positive values?" This is determined by the ratio of true positives to the total number of actual positive values.

-

- F1-score F1 score should be used when both precision and recall are important for the use case. F1 score is the harmonic mean of precision and recall. It lies between [0,1].[43]
- Entropy measures the randomness or disorder within a system.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$Recall = Sensitivity = \frac{TP}{TP + FN} \tag{4.3}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{4.4}$$

$$Entropy = \log 2(Precision) \times -Precision \tag{4.5}$$

4.3 Fake news detection system

Metaheuristic algorithms are a method used to solve various optimization problems across different fields, regardless of their multi-level complexity, which presents significant challenges. Therefore, the effectiveness of these algorithms should be assessed using difficult optimization problems.

4.3.1 Transition from natural to artificial

This part is dedicated to the transition from the natural life of the Golf to the life artificial as shown in the following table :

Natural	Artificial
Golfers playing on a natural course with varied terrain.	Solutions in a search space with varied constraints and objectives.
Golfers choosing clubs and strategies based on distance to the hole and course conditions.	Algorithm selecting operators and parameters based on distance to the optimal solution and problem conditions.
Golfers aiming to get the ball into the hole in the fewest strokes.	Algorithm aiming to find the optimal solution with the least number of iterations or computational resources.
Golf ball traveling towards the hole based on the golfer's skill and environmental factors (wind, slope).	Solution iterating towards the optimum based on algorithmic adjustments and problem landscape.
Group of golfers playing in a tournament to find the best player.	Population of candidate solutions in an iterative process to find the best solution.
The golfer with the lowest score at the end of the tournament.	The best solution with the highest fitness value or lowest objective function value at the end of the optimization process.
Distance between the golf ball and the hole after each stroke.	Distance between the current solution and the optimal solution in each iteration.

Figure 4.14: Transition from natural to artificial

The table 4.9 compares natural and artificial concepts, specifically focusing on the

Golf Optimization Algorithm and its objective of finding the best solution in a search space with varied constraints and objectives. It highlights the parallel between the actions and strategies of golfers in a natural course and the operations of an algorithm selecting parameters to achieve optimal solutions. The mention of distance calculations after each stroke suggests an iterative process to converge on the optimal solution, similar to the way a golfer iterates their strokes to get the ball into the hole.

In the context of the Golf Optimization Algorithm, this implies that the algorithm aims to minimize the number of iterations or computational resources needed to find the best solution, analogous to golfers aiming to get the ball into the hole in the fewest strokes. The comparison also emphasizes the influence of external factors and environmental conditions on the golf ball's path, akin to how algorithmic adjustments and problem landscape influence the solution's trajectory.

Overall, the table offers an intriguing perspective on how nature can inspire the development of algorithms and their application in solving real-world problems within artificial systems. It specifically explores the iterative and strategic aspects of problem-solving, drawing a parallel between a golfer's journey to the hole and an algorithm's search for optimal solutions. This highlights the potential of nature-inspired algorithms to address complex optimization problems in various fields.

4.3.2 How the chosen metaheuristic was used

The chosen metaheuristic algorithm, the Golf Optimization Algorithm (GOA), was selected for its unique approach inspired by the strategic choreography observed in the game of golf. The GOA is a population-based method that conducts a random search in the problem-solving space to find optimal solutions. The algorithm's members are positioned in the search space, determining the values of the problem variables. The population distribution is achieved through a uniform distribution, similar to other metaheuristic algorithms .

The GOA was designed as a novel approach in the field of metaheuristic algorithms, with its inspiration drawn from the strategic movements and intellectual prowess displayed in the game of golf. By simulating the rules and behaviors of golf players, the GOA aims to provide effective solutions to optimization problems by balancing exploration and exploitation during the search process .

In the study, the GOA was evaluated on fifty-two standard objective functions to

assess its efficiency in solving optimization problems. The algorithm's performance was compared with that of ten well-known metaheuristic algorithms to determine its effectiveness. Additionally, the GOA was tested on real-world applications, including engineering design problems and the optimization of energy carriers' operation with respect to energy network resilience, where it demonstrated superior performance and effective handling of practical scenarios.

4.3.3 Machine learning parametrs

Here are the machine learning parameters:

RandomForestClassifier:

- **n_estimators:** Number of trees in the RandomForestClassifier model.
- **max_depth:** Maximum depth of the decision trees. If set to None, nodes are expanded until all leaves are pure or until the minimum samples required to split a node are reached.
- **min_samples_split:** Minimum number of samples required to split an internal node.

4.3.4 Metaheuristics parameters

The parameters of a metaheuristic, such as the Golf optimization algorithm, are crucial for ensuring effective search and robust optimization. In our implementation, these parameters are defined as follows:

Objective_function: The objective function to optimize, here being the accuracy of the Random Forest model on a test set.

dimension: The number of parameters to optimize is 3, corresponding to the hyperparameters n_estimators, max_depth, and min_samples_split.

lower_bound and upper_bound: The search space bounds are [10, 1, 2] and [200, 50, 20], respectively, defining the limits of the hyperparameters. For instance, the number of trees (n_estimators) ranges from 10 to 200, the maximum depth (max_depth) from 1 to 50, and the minimum samples required to split a node (min_samples_split) from 2 to 20.

population_size: The population size is set to 10, meaning that 10 potential solutions are evaluated in each iteration.

max_iterations: The maximum number of iterations is 10, allowing the algorithm to gradually refine the solutions.

4.3.5 Used fitness function

The Golf Optimization Algorithm (GOA) uses a fitness function to evaluate the performance of solution candidates. This function specifically assesses the accuracy of a Random Forest classifier based on a set of hyperparameters. These parameters include the number of estimators (n_estimators), the maximum depth of the trees (max_depth), and the minimum number of samples required to split an internal node (min_samples_split). The fitness function first initializes a Random Forest model with these parameters and then trains the model using the training dataset (X_train, y_train). It makes predictions on the test dataset (X_test) and calculates the accuracy of these predictions by comparing them with the true labels (y_test). To facilitate the optimization, the accuracy score is returned as a negative value, since the GOA aims to minimize this function to find the optimal set of hyperparameters. as follows in the next algorithm :

Algorithm 1 Define the objective function

```
def objective_function(params):
                                                    ▷ Unpacking the input parameters
    n_estimators, max_depth, min_samples_split = params
                ▷ Creating a random forest classification model with the input parameters
    model = RandomForestClassifier(
        n_estimators=int(n_estimators),                    ▷ Number of trees in the forest
        max_depth=int(max_depth) if max_depth > 0 else None,   ▷ Maximum depth of the trees
        min_samples_split=int(min_samples_split),          ▷ Minimum number of samples required to split a node
        random_state=42                                ▷ Fixed random state for reproducibility
    )
                                                ▷ Training the model on the training data
    model.fit(X_train, y_train)
                                                          ▷ Predicting on the test data
    y_pred = model.predict(X_test)
                                              ▷ Evaluating the model with accuracy score
    score = accuracy_score(y_test, y_pred)

                                          ▷ Returning the negative score (for optimization)
    return -score
```

Define Fitness Function within the GOA

Within the Golf Optimization Algorithm (GOA), the fitness function is crucial for guiding the search for optimal hyperparameters of the Random Forest classifier. The algorithm operates by defining the search space with lower and upper bounds for each hyperparameter: n_estimators (10 to 200), max_depth (1 to 50), and min_samples_split (2 to 20). The GOA starts with a randomly initialized population of 10 potential solutions within these bounds. For each individual in the population, the fitness function is evaluated by training a Random Forest model and computing its accuracy on the test dataset. Over 10 iterations, the algorithm updates the population by exploring new potential solutions and refining the best individuals, using strategies to balance exploration and exploitation.

This process is repeated for a defined number of iterations (e.g., 10), ultimately identifying the set of hyperparameters that maximizes the accuracy of the Random Forest classifier as evaluated by the fitness function. The best individual found through this optimization process is then used to train a final model, whose performance is validated using accuracy, recall, precision, and F1 score metrics.

Algorithm 2 Define Objective_function within the Golf

```
fitness ← np.array([objective_function(individual) for individual in population])
best_individual ← population[np.argmin(fitness)]
best_fitness ← np.min(fitness)
for t in range(max_iterations) do
    best_individual ← population[np.argmin(fitness)]
    best_fitness ← np.min(fitness)
                                                          ▷ Phase d'exploration
    for i in range(population_size) do
        new_position ← population[i] + np.random.uniform(0, 1, dimension) * (upper_bound − i * population[i])
        new_position ← np.clip(new_position, lower_bound, upper_bound)
        new_fitness ← objective_function(new_position)
        if new_fitness < fitness[i] then
            population[i] ← new_position
            fitness[i] ← new_fitness
        end if
    end for
                                                          ▷ Phase d'exploitation
    for i in range(population_size) do
        new_position ← population[i] + (1 − 2 * np.random.uniform(0, 1, dimension)) * (lower_bound + np.random.uniform(0, 1, dimension) * (upper_bound − lower_bound)) / t
        new_position ← np.clip(new_position, lower_bound, upper_bound)
        new_fitness ← objective_function(new_position)
        if new_fitness < fitness[i] then
            population[i] ← new_position
            fitness[i] ← new_fitness
        end if
    end for
end for
```

```python
def objective_function(params):
    """
    Fonction objectif à optimiser. Elle évalue les performances d'un modèle RandomForestClassifier
    avec les hyperparamètres donnés.

    Args:
        params: Liste des hyperparamètres (nombre d'estimateurs, profondeur maximale, min_samples_split).

    Returns:
        Le score d'exactitude négatif pour l'optimisation.
    """
    n_estimators, max_depth, min_samples_split = params

    # Création d'un modèle RandomForestClassifier avec les hyperparamètres donnés
    model = RandomForestClassifier(n_estimators=int(n_estimators),
                                   max_depth=int(max_depth) if max_depth > 0 else None,
                                   min_samples_split=int(min_samples_split),
                                   random_state=42)

    # Entraînement du modèle sur les données d'entraînement
    model.fit(X_train, y_train)

    # Prédiction sur les données de test
    y_pred = model.predict(X_test)

    # Calcul du score d'exactitude
    score = accuracy_score(y_test, y_pred)

    # Nous voulons minimiser cette fonction, donc nous retournons le score d'exactitude négatif
    return -score
```

Figure 4.15: Screenshot 1 of the used code

```python
def golf_optimization_algorithm(objective_function, dimension, lower_bound, upper_bound, population_size, max_iterations):
    """
    Implémente l'algorithme d'optimisation Golf.

    Args:
        objective_function: La fonction objectif à optimiser.
        dimension: Le nombre de dimensions de l'espace de recherche.
        lower_bound: La borne inférieure de l'espace de recherche.
        upper_bound: La borne supérieure de l'espace de recherche.
        population_size: Le nombre d'individus dans la population.
        max_iterations: Le nombre maximal d'itérations.

    Returns:
        Le meilleur individu trouvé et son fitness.
    """
    lower_bound = np.array(lower_bound)
    upper_bound = np.array(upper_bound)

    # Initialisation de la population avec des valeurs aléatoires dans l'espace de recherche
    population = np.random.uniform(lower_bound, upper_bound, (population_size, dimension))

    # Évaluation de la fonction objectif pour chaque individu
    fitness = np.array([objective_function(individual) for individual in population])

    # Trouver le meilleur individu et son fitness
    best_individual = population[np.argmin(fitness)]
    best_fitness = np.min(fitness)

    # Boucle d'itération principale
    for t in range(max_iterations):
        # Mise à jour du meilleur individu et de son fitness
        best_individual = population[np.argmin(fitness)]
        best_fitness = np.min(fitness)
```

Figure 4.16: Screenshot 2 of the used code

```
        # Phase d'exploration
        for i in range(population_size):
            # Génération d'une nouvelle position pour l'individu i
            new_position = population[i] + np.random.uniform(0, 1, dimension) * (upper_bound - i * population[i])
            new_position = np.clip(new_position, lower_bound, upper_bound)

            # Évaluation de la fonction objectif pour la nouvelle position
            new_fitness = objective_function(new_position)

            # Mise à jour de l'individu si la nouvelle position est meilleure
            if new_fitness < fitness[i]:
                population[i] = new_position
                fitness[i] = new_fitness

        # Phase d'exploitation
        for i in range(population_size):
            # Génération d'une nouvelle position pour l'individu i
            new_position = population[i] + (1 - 2 * np.random.uniform(0, 1, dimension)) *
             (lower_bound + np.random.uniform(0, 1, dimension) * (upper_bound - lower_bound)) / t
            new_position = np.clip(new_position, lower_bound, upper_bound)

            # Évaluation de la fonction objectif pour la nouvelle position
            new_fitness = objective_function(new_position)

            # Mise à jour de l'individu si la nouvelle position est meilleure
            if new_fitness < fitness[i]:
                population[i] = new_position
                fitness[i] = new_fitness

    # Retourne le meilleur individu trouvé et son fitness
    return best_individual, best_fitness
```

Figure 4.17: Screenshot 3 of the used code

Experimental software environment

4.3.6 Testing software environment

Different tools are used for this study. All of them are free and open source.

- Python 3.10.12
- NumPy 1.25.2
- Matplotlib 3.7.1
- Pandas 2.0.3
- SciPy 1.11.4
- Scikit-learn 1.2.2

- Seaborn 0.13.1
- google colab

Python [11] is a high level general programming language and is very widely used in all types of disciplines such as general programming, web development, software development, data analysis, machine learning etc. Python is used for this project because it is very flexible and easy to use and also documentation and community support is very large.

NumPy [12]is very powerful package which enables us for scientific computing. It comes with sophisticated functions and is able to perform N-dimensional array, algebra, Fourier transform etc. NumPy is used very where in data analysis, image processing and also different other libraries are built above NumPy and NumPy acts as a base stack for those libraries

Matplotlib [13]is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

Pandas [14] is open source BSD licensed software specially written for python programming language. It provides complete set of data analysis tools for python and is best competitor for R programming language. Operations like reading data-frame, reading csv and excel files, slicing, indexing, merging, handling missing data etc., can be easily performed with Pandas. Most important feature of Pandas is, it can perform time series analysis

SciPy [15]is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data. With SciPy, an interactive Python session becomes a data-processing and system-prototyping environment rivaling systems, such as MATLAB, IDL, Octave, R-Lab, and SciLab.

[11]https://www.python.org/
[12]https://numpy.org
[13]https://matplotlib.org
[14]https://pandas.pydata.org
[15]https://docs.scipy.org/doc/scipy/tutorial/general.html

For this study, scikit-learn is used because it is based on python and can interoperate to NumPy library. It is also very easy to use.

Scikit-Learn (SKLearn) [16] is an environment that is integrated with Python programming language. The library offers a wide range of supervised algorithms . The library offers high-level implementation to train with the 'Fit' methods and 'predict' from an Classifier and also offers to perform the cross validation, feature selection and parameter tuning.

Seaborn [17]Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

4.4 Conclusion

This chapter has provided a detailed methodology for detecting fake news using machine learning techniques. Starting with data preparation, we handled missing values to ensure the integrity of the dataset. Subsequently, supervised and unsupervised learning models were trained and evaluated to detect fraudulent accounts. Hyperparameter optimization was performed using the Golf Optimization Algorithm (GOA), enhancing model performance. Performance metrics such as accuracy, F1 score, recall, and the confusion matrix were used to assess model effectiveness. By combining these approaches and utilizing Python tools like NumPy, Pandas, and Scikit-learn, we developed a robust system for analyzing and detecting fraudulent profiles, making significant contributions to online platform security.

[16]https://scikit-learn.org
[17]https://seaborn.pydata.org

Chapter 5

Results and discussion

5.1 Introduction

This section presents the results from our analysis of the datasets and a comparison of algorithms. Following preprocessing, descriptive analysis, and exploratory analysis, various machine learning techniques were used to process the datasets. The findings of the experiments are presented in the tables below, along with an explanation of the best performer as determined by several performance measures.

The results are grouped by datasets. Each subsection is organized according to the following taxonomy:

- Supervised techniques
- Unsupervised techniques
- Bio-inspired algorithms

5.2 Results of datasets

5.2.1 Spanish Political Fake News

- **Supervised Techniques :**

  Table 5.1 provides a summary of the calculated metrics for supervised training models on the Spanish Political Fake News dataset:

Spanish Political Fake News					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Random Forest	0.897	0.89	0.94	0.91	9.270
KNN (K=3)	0.64	0.69	0.71	0.70	9.241
KNN (K=5)	0.72	0.74	0.80	0.77	9.275
KNN (K=7)	0.76	0.77	0.84	0.80	9.298
KNN (K=9)	0.79	0.79	0.87	0.83	9.313
SVM	0.867	0.84	0.95	0.89	9.320
Naive Bayes	0.792	0.75	0.97	0.84	9.471
Decision Tree	0.894	0.91	0.91	0.91	9.202
GB	0.868	0.82	0.98	0.90	9.387

Table 5.1: Evaluation of supervised algorithms for Spanish Political Fake News dataset.

Table 5.1 summarizes the performance of various algorithms on the Spanish Political Fake News dataset with a 70% train and 30% test split. Random Forest achieved the highest accuracy at 0.897, followed by Decision Tree at 0.894. SVM and Gradient Boosting also showed strong performance with accuracies of 0.867 and 0.868, respectively. KNN's performance improved with higher K values, ranging from 0.64 to 0.79 in accuracy. Naive Bayes had an accuracy of 0.792, demonstrating good recall but lower precision. Overall, Random Forest and Decision Tree were the most effective in detecting fake news.

- **Unsupervised Techniques :**

  Table 5.2 provides a summary of the calculated metrics for unuspervised training models on the Spanish Political Fake News dataset:

Spanish Political Fake News					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Kmeans	0.653	0.63	1.00	0.77	9.675

Table 5.2: Evaluation of unsupervised algorithms for Spanish Political Fake News dataset.

Table 5.2 shows the performance of the Kmeans algorithm on the Spanish Political Fake News dataset, using a 70% train and 30% test split. Kmeans achieved an accuracy of 0.653, precision of 0.63, recall of 1.00, F1 score of 0.77, and entropy of 9.675. This indicates that Kmeans has perfect recall but lower precision and moderate overall accuracy in detecting fake news.

- **Time :**

  Table 5.3 provides a summary of the testing time and training time in seconds for Spanish Political Fake News dataset:

Spanish Political Fake News							
Time	Train-Test size 70% - 30%						
	RF	**KNN**	**SVM**	**NB**	**DT**	**GB**	**KM**
Training time	108.344	0.009	1426.04	0.020	48.531	55.258	6.131
Testing time	1.987	678.072	161.939	0.004	0.027	0.028	0.003

Table 5.3: Training and Testing Times for Various Algorithms for Spanish Political Fake News dataset.

Table 5.3 compares the training and testing times in seconds for different machine learning algorithms. Random Forest (RF) has a training time of 108.344s and a testing time of 1.987s. K-Nearest Neighbors (KNN) trains in 0.009s but tests in 678.072s, indicating significant computational overhead during testing. Support Vector Machine (SVM) takes the longest to train (1426.04s) and test (161.939s). Naive Bayes

(NB) is the fastest to train (0.020s) and test (0.004s). Decision Tree (DT) and Gradient Boosting (GB) have moderate training times (48.531s and 55.258s, respectively) and low testing times (0.027s and 0.028s, respectively). K-means (KM) trains in 6.131s and has the shortest testing time at 0.003s.

- **Size :**

  Table 5.4 provides a summary of the size before and after preprocessing for Spanish Political Fake News dataset:

Spanish Political Fake News	
	(row,column)
Before preprocessing	(57231, 5)
After preprocessing	(57231, 2)

Table 5.4: The size before and after preprocessing for Various Algorithms for Spanish Political Fake News dataset.

- **Correlation matrix:**

  In the figure 5.1, the correlation matrix between the classification models of spanish Political Fake News dataset.

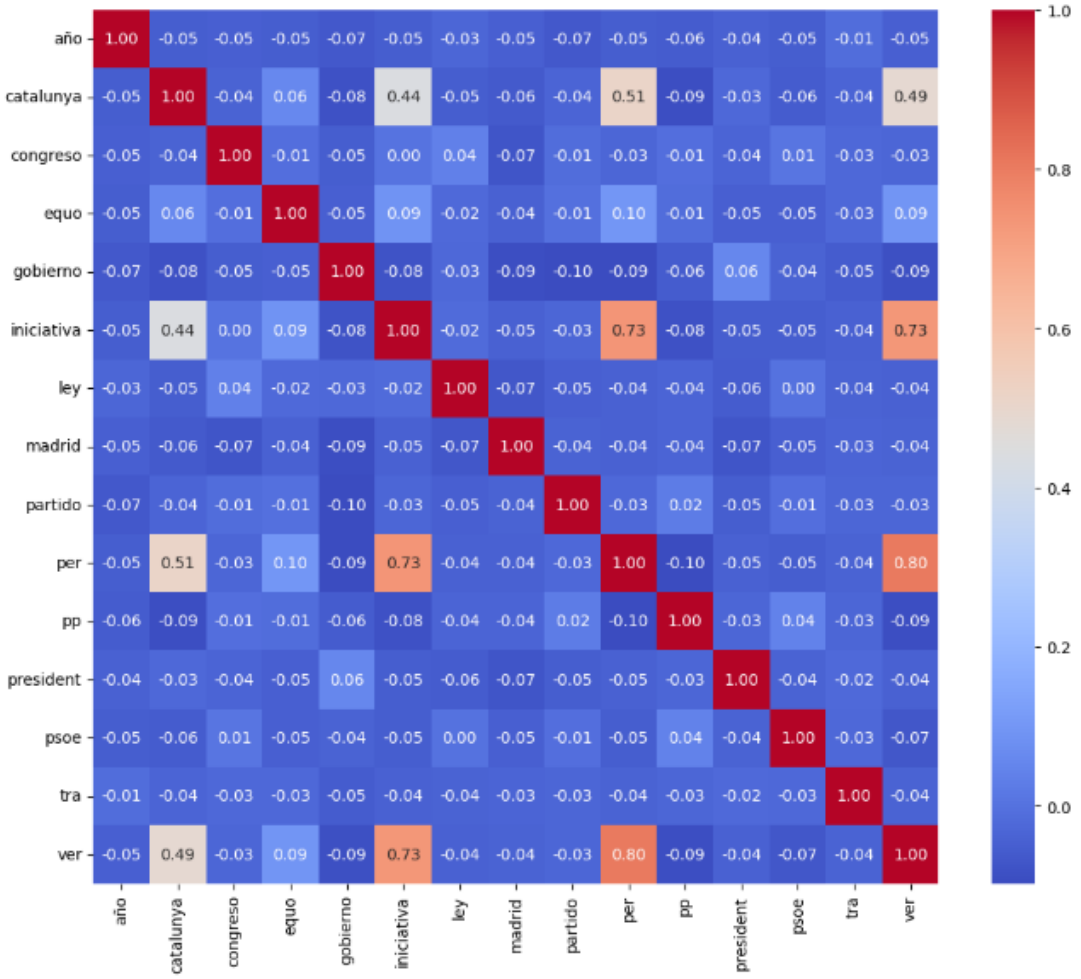

Figure 5.1: *Correlation matrix of spanish Political Fake News dataset*

Figure 5.1 displays a heatmap illustrating the correlation matrix between various terms, likely representing political entities or concepts. The correlation coefficients range from -1 to 1, with the color gradient indicating the strength and direction of the relationships. Red shades denote strong positive correlations, while blue shades indicate strong negative correlations or weak relationships. For example, terms like ”iniciativa” and ”per” show a high positive correlation (0.73), and ”per” and ”pp” have a very strong positive correlation (0.80), suggesting these terms frequently co-occur or share similar contexts. Conversely, terms like ”año” and ”madrid” exhibit a negative correlation (-0.06), implying they do not frequently appear together. This heatmap effectively highlights the interconnections between terms, providing insights into their associative patterns.

- **Confusion matrix:** In the figure 5.2, the confusion matrix for the classification random forest of spanish Political Fake News dataset.

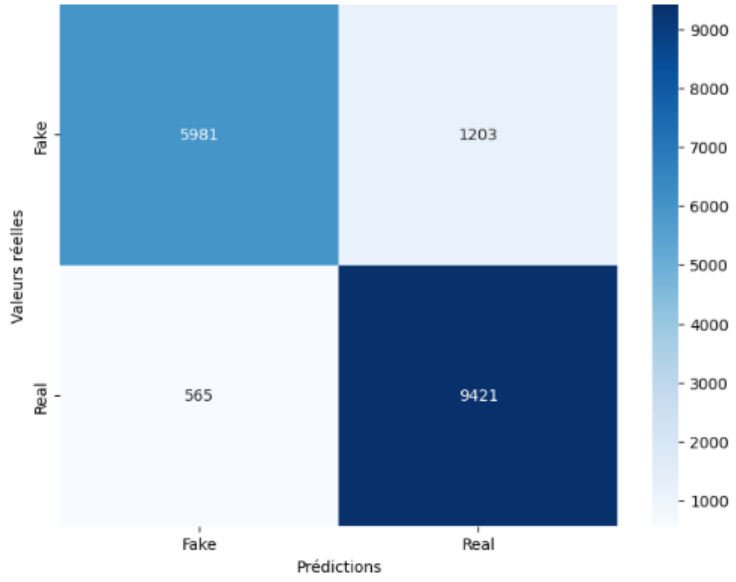

Figure 5.2: *Confusion matrix of spanish Political Fake News dataset*

Figure 5.2 it's the confusion matrix for the RandomForestClassifier model shows that it correctly classified 5981 "Fake" instances and 9421 "Real" instances. However, it made 565 false positives and 1203 true negatives.

5.2.2 COVID-19 Fake News Dataset

- **Supervised Techniques :**

  Table 5.5 provides a summary of the calculated metrics for supervised training models on the COVID-19 Fake News Dataset:

COVID-19 Fake News					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Random Forest	0.753	0.75	0.94	0.83	6.633
KNN (K=3)	0.72	0.81	0.75	0.78	6.32
KNN (K=5)	0.75	0.78	0.84	0.81	6.48
KNN (K=7)	0.73	0.75	0.87	0.81	6.56
KNN (K=9)	0.74	0.75	0.90	0.82	6.60
SVM	0.801	0.80	0.93	0.81	6.552
Naive Bayes	0.659	0.66	1.00	0.79	6.826
Decision Tree	0.715	0.77	0.79	0.78	6.424
GB	0.791	0.79	0.92	0.85	6.556

Table 5.5: Evaluation of supervised algorithms for COVID-19 Fake News dataset.

Table 5.5 summarizes the performance of various algorithms on the COVID-19 Fake News dataset, using a 70% train and 30% test split. The metrics include accuracy, precision, recall, F1 score, and entropy. SVM achieved the highest accuracy at 0.801, followed by Gradient Boosting at 0.791, and Random Forest at 0.753. KNN algorithms showed varying performance with accuracies between 0.72 and 0.75. Naive Bayes had the lowest accuracy at 0.659, despite a high recall. Overall, SVM and Gradient Boosting performed the best in detecting COVID-19 fake news.

- **Unsupervised Techniques :**

  Table 5.6 provides a summary of the calculated metrics for unspervised training models on the COVID-19 Fake News dataset:

COVID-19 Fake News					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Kmeans	0.566	0.82	0.42	0.56	5.746

Table 5.6: Evaluation of unsupervised algorithms for COVID-19 Fake News dataset.

Table 5.6 shows the performance of the Kmeans algorithm on the COVID-19 Fake News dataset, using a 70% train and 30% test split. Kmeans achieved an accuracy of 0.566, precision of 0.82, recall of 0.42, F1 score of 0.56, and entropy of 5.746. This indicates moderate accuracy and F1 score, with high precision but lower recall.

- **Time :**

Table 5.7 provides a summary of the testing time and training time in seconds for COVID-19 Fake News dataset:

COVID-19 Fake News							
Time	Train-Test size 70% - 30%						
	RF	**KNN**	**SVM**	**NB**	**DT**	**GB**	**KM**
Training time	5.262	0.005	14.839	0.015	2.438	20.083	3.864
Testing time	0.104	83.882	4.739	0.002	0.010	0.004	0.029

Table 5.7: Training and Testing Times in seconds for Various Algorithms for COVID-19 Fake News dataset.

Table 5.7 shows the training and testing times in seconds for various algorithms. KNN trains very quickly (0.005 s) but takes a long time to test (83.882 s). Gradient Boosting has the longest training time (20.083 s), while Naive Bayes is fast for both training (0.015 s) and testing (0.002 s).

- **Size :**

Table 5.8 provides a summary of the size before and after preprocessing for COVID-19 Fake News dataset:

COVID-19 Fake News	
	(row,column)
Before preprocessing	(3119, 5)
After preprocessing	(3119, 2)

Table 5.8: Training and Testing Times for Various Algorithms for COVID-19 Fake News dataset.

- **Correlation matrix:**

  In the figure 5.3, the correlation matrix between the classification models of COVID-19 Fake News dataset.

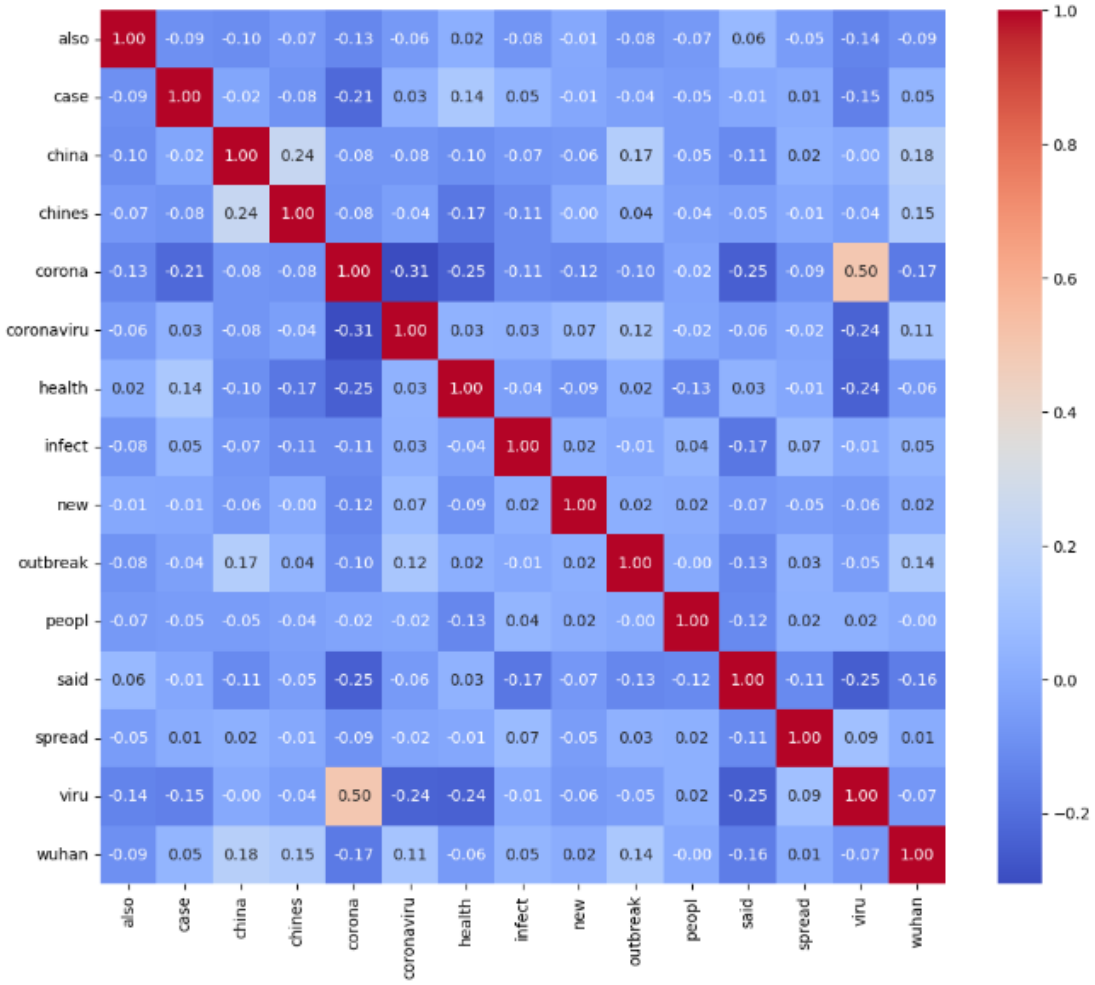

Figure 5.3: *Correlation matrix of COVID-19 Fake News dataset*

Figure 5.3 displays a heatmap illustrating the correlation matrix between various terms related to COVID-19 and its spread. The correlation coefficients range from -1 to 1, represented by a color gradient from blue (negative correlation) to red (positive correlation). Key observations include a strong positive correlation between ”virus” and ”coronavirus” (0.50), indicating these terms frequently co-occur. Another notable positive correlation is between ”corona” and ”virus” (0.50). Conversely, many terms show weak or negative correlations, such as ”china” and ”wuhan” (-0.09), suggesting limited co-occurrence. This heatmap effectively visualizes the associative patterns and relationships between key terms related to the pandemic, highlighting how certain terms are more closely linked in discussions.

- **Confusion matrix:** In the figure 5.4, the confusion matrix for the classification random forest of COVID-19 Fake News dataset.

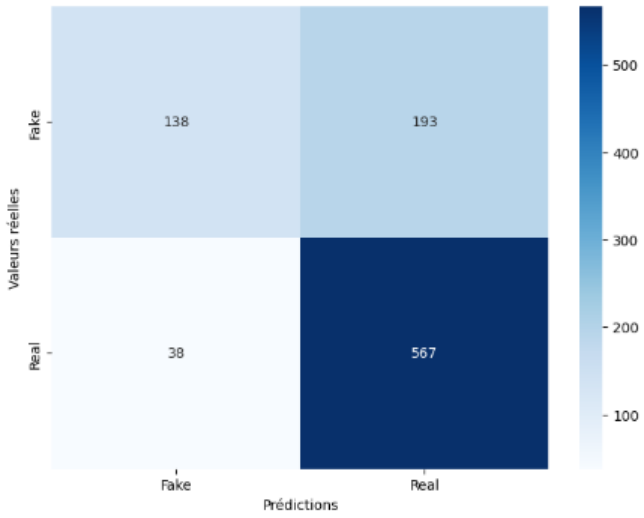

Figure 5.4: *Confusion matrix of COVID-19 Fake News dataset*

Figure 5.4 it's the confusion matrix for the RandomForestClassifier model shows that it correctly classified 138 "Fake" instances and 567 "Real" instances. However, it made 38 false positives and 193 true negatives.

5.2.3 Fake News Dataset Combined Different Sources

- **Supervised Techniques :**

  Table 5.9 provides a summary of the calculated metrics for supervised training models on the Fake News Dataset Combined Different Sources:

Fake News Dataset Combined Different Sources					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Random Forest	0.942	0.96	0.89	0.92	8.917
KNN (K=3)	0.78	0.89	0.49	0.63	8.390
KNN (K=5)	0.75	0.90	0.41	0.56	8.188
KNN (K=7)	0.73	0.90	0.35	0.50	8.046
KNN (K=9)	0.72	0.91	0.30	0.46	7.898
SVM	0.963	0.97	0.94	0.95	8.953
Naive Bayes	0.864	0.92	0.72	0.80	8.740
Decision Tree	0.940	0.93	0.91	0.92	8.969
GB	0.947	0.97	0.93	0.89	8.902

Table 5.9: Evaluation of supervised algorithms for Fake News Dataset Combined Different Sources.

Table 5.9 compares different algorithms on a combined fake news dataset using a 70%-30% train-test split, focusing on Accuracy, Precision, Recall, F1 score, and Entropy.

SVM had the best accuracy at 0.963, with high precision and recall. Random Forest and Gradient Boosting also performed well with accuracies of 0.942 and 0.947. Decision Tree was close with 0.940 accuracy. Naive Bayes had decent performance but lower recall. KNN had the lowest performance, with accuracy and F1 scores dropping as K increased. In summary, SVM, Random Forest, and Gradient Boosting were the top performers, while KNN was the weakest.

- **Unsupervised Techniques :**

  Table 5.10 provides a summary of the calculated metrics for unspervised training models on the Fake News Dataset Combined Different Sources:

Fake News Dataset Combined Different Sources					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Kmeans	0.525	0.35	0.27	0.31	8.722

Table 5.10: Evaluation of unsupervised algorithms for Fake News Dataset Combined Different Sources.

Table 5.10 shows the performance of the K-means algorithm on a combined fake news dataset using a 70%-30% train-test split. The metrics include Accuracy, Precision, Recall, F1 score, and Entropy.

K-means had a low accuracy of 0.525, with low precision (0.35), recall (0.27), and F1 score (0.31). This indicates that K-means is not very effective for this fake news detection task, showing weaker performance across all metrics.

- **Time :**

  Table 5.11 provides a summary of the testing time and training time in seconds for Fake News Dataset Combined Different Sources:

Fake News Dataset Combined Different Sources							
Time	Train-Test size 70% - 30%						
	RF	**KNN**	**SVM**	**NB**	**DT**	**GB**	**KM**
Training time	230.344	0.039	4299.58	0.081	180.041	360.683	16.466
Testing time	3.021	22493.7	632.963	0.021	0.038	0.066	0.018

Table 5.11: Training and Testing Times for Various Algorithms for Fake News Dataset Combined Different Sources.

Table 5.11 compares the training and testing times of various algorithms on the fake news dataset. KNN has the shortest training time (0.039 seconds) but the longest testing time (22493.7 seconds). SVM has a long training time (4299.58 seconds)

and a considerable testing time (632.963 seconds). Random Forest and Gradient Boosting have moderate training times (230.344 and 360.683 seconds) with shorter testing times (3.021 and 0.066 seconds). Naive Bayes and K-means are the fastest overall, with very short training (0.081 and 16.466 seconds) and testing times (0.021 and 0.018 seconds). Decision Tree has moderate training (180.041 seconds) and very short testing time (0.038 seconds).

- **Size :**

  Table 5.12 provides a summary of the size before and after preprocessing for Fake News Dataset Combined Different Sources:

Fake News Dataset Combined Different Sources	
	(row,column)
Before preprocessing	(69045, 4)
After preprocessing	(69045, 2)

Table 5.12: The size before and after preprocessing for Various Algorithms for Fake News Dataset Combined Different Sources.

- **Correlation matrix:**

  In the figure 5.5, the correlation matrix between the classification models of Fake News Dataset Combined Different Sources News dataset.

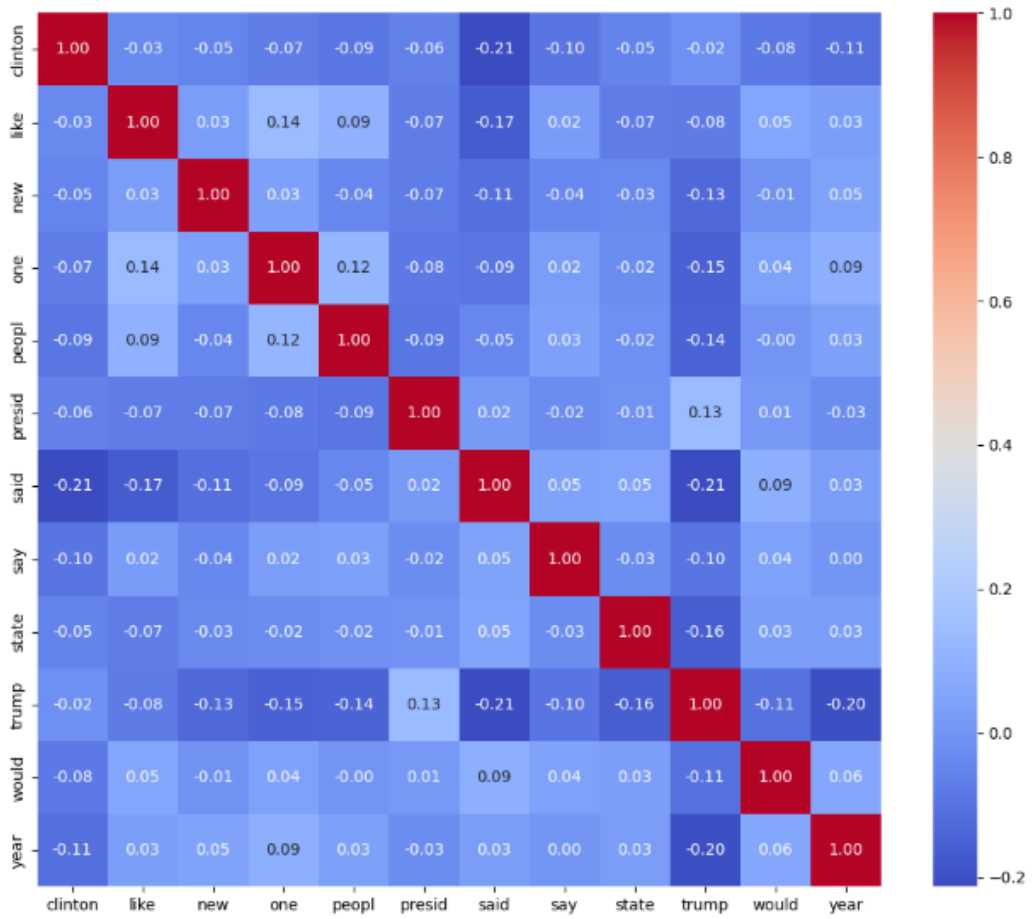

Figure 5.5: *Correlation matrix of Fake News Dataset Combined Different Sources*

Figure 5.5 displays a correlation matrix of words. The color red indicates positive correlation, while blue indicates negative correlation. The darker the color, the stronger the correlation. For instance, ”clinton” and ”year” have a strong negative correlation, while ”like” and ”new” have a positive correlation.

- **Confusion matrix:** In the figure 5.6, the confusion matrix for the classification random forest of Fake News Dataset Combined Different Sources.

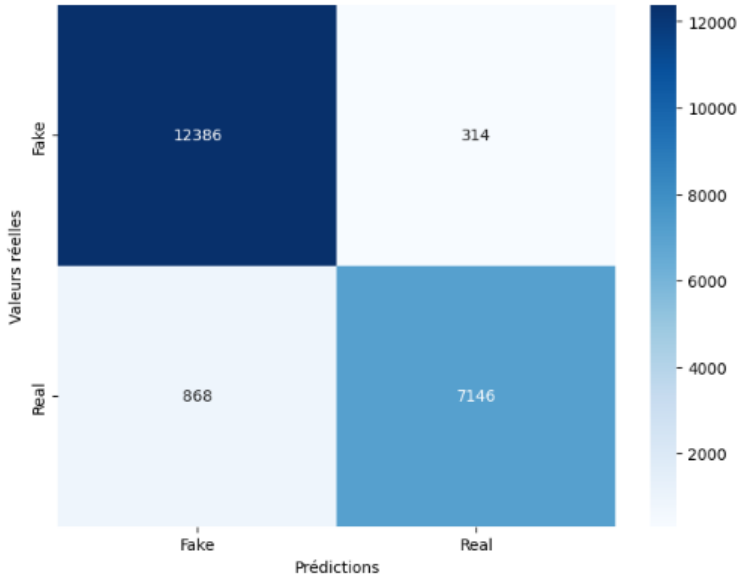

Figure 5.6: *Confusion matrix of Fake News Dataset Combined Different Sources*

Figure 5.6 it's the confusion matrix for the RandomForestClassifier model shows that it correctly classified 12386 "Fake" instances and 7146 "Real" instances. However, it made 868 false positives and 314 true negatives.

5.2.4 BanFakeNews dataset

- **Supervised Techniques :**

Table 5.13 provides a summary of the calculated metrics for supervised training models on the BanFakeNews dataset :

BanFakeNews dataset					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Random Forest	0.988	0.99	1.00	0.99	9.736
KNN (K=3)	0.96	0.97	0.99	0.98	9.75
KNN (K=5)	0.96	0.96	0.99	0.98	9.76
KNN (K=7)	0.96	0.96	0.99	0.98	9.76
KNN (K=9)	0.96	0.96	1.00	0.98	9.76
SVM	0.971	0.97	1.00	0.99	9.754
Naive Bayes	0.961	0.96	1.00	0.98	9.761
Decision Tree	0.979	0.99	0.99	0.99	9.723
GB	0.966	0.97	1.00	0.98	9.760

Table 5.13: Evaluation of supervised algorithms for BanFakeNews dataset.

Table 5.13 summarizes the performance of various algorithms on the BanFakeNews dataset, using a 70% train and 30% test split. It includes metrics like accuracy, precision, recall, F1 score, and entropy. Random Forest achieved the highest accuracy at 0.988, followed by Decision Tree at 0.979 and SVM at 0.971. KNN algorithms, with different K values, consistently scored 0.96 in accuracy. Naive Bayes and Gradient Boosting also performed well, showing strong precision and recall across the board.

- **Unsupervised Techniques :**

  Table 5.14 provides a summary of the calculated metrics for unuspervised training models on the BanFakeNews dataset :

BanFakeNews dataset					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Kmeans	0.278	0.97	0.25	0.40	8.392

Table 5.14: Evaluation of unsupervised algorithms for BanFakeNews dataset dataset.

Table 5.14 shows the performance of the Kmeans algorithm on the BanFakeNews dataset, using a 70% train and 30% test split. Kmeans achieved an accuracy of 0.278, with a precision of 0.97, recall of 0.25, F1 score of 0.40, and entropy of 8.392. This indicates that while Kmeans has high precision, it struggles with overall accuracy and recall in detecting fake news.

- **Time :**

  Table 5.15 provides a summary of the testing time and training time in seconds for BanFakeNews dataset :

BanFakeNews dataset							
Time	Train-Test size 70% - 30%						
	RF	**KNN**	**SVM**	**NB**	**DT**	**GB**	**KM**
Training time	23.585	0.006	203.842	0.012	13.212	24.131	1.044
Testing time	1.073	157.046	37.322	0.002	0.014	0.028	0.0017

Table 5.15: Training and Testing Times for Various Algorithms for BanFakeNews dataset
.

Table 5.15 presents the training and testing times in seconds for different algorithms applied to the BanFakeNews dataset. Random Forest (RF) has the longest training time at 23.585 seconds, followed by Support Vector Machine (SVM) with 203.842 seconds. On the other hand, K-nearest neighbors (KNN) with k=9 (KM) has the shortest training time of 1.044 seconds. For testing time, KNN with k=5 has the

highest time of 157.046 seconds, while Decision Tree (DT) and Gradient Boosting (GB) have the lowest testing times, both under 0.02 seconds. These figures indicate that KNN takes the longest time for testing, whereas DT and GB require the shortest time. The significant difference in training and testing times among algorithms highlights the importance of considering computational efficiency when selecting a model for the BanFakeNews dataset, with KNN demanding the most time and DT and GB being the most efficient.

- **Size :**

  Table 5.16 provides a summary of the size before and after preprocessing for BanFakeNews dataset dataset:

BanFakeNews dataset	
	(row,column)
Before preprocessing	(58478, 10)
After preprocessing	(44898, 2)

Table 5.16: The size before and after preprocessing for Various Algorithms for BanFakeNews dataset .

- **Correlation matrix:**

  In the figure 5.7, the correlation matrix between the classification models of BanFakeNews dataset

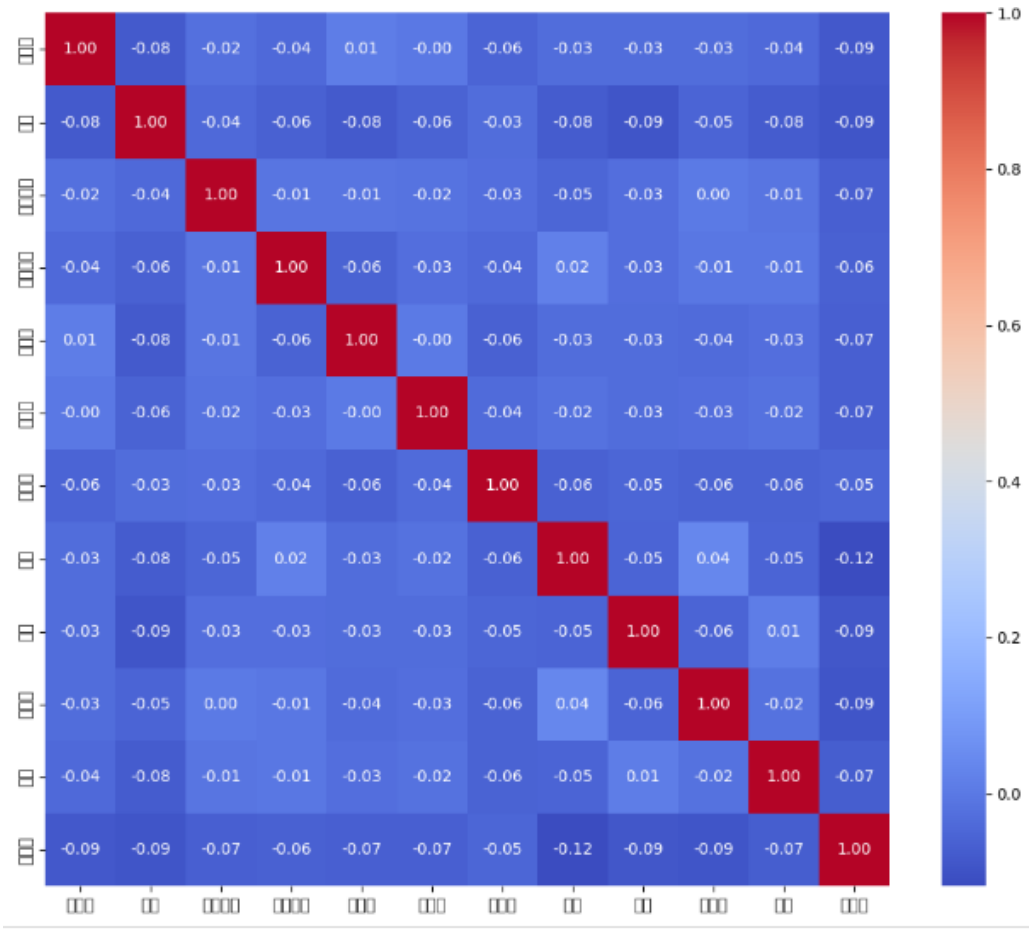

Figure 5.7: *Correlation matrix of BanFakeNews dataset*

Figure 5.7 displays a correlation matrix in this Figure shows the strength and direction of the linear relationships between various variables. The variables are represented by the rows and columns of the matrix, and the correlation coefficient between each pair of variables is represented by the color and number in the corresponding cell. Positive correlations are shown in red, with darker shades indicating stronger correlations, while negative correlations are shown in blue. The diagonal of the matrix consists of 1s, indicating perfect positive correlations between each variable and itself.

- **Confusion matrix:** In the figure 5.8, the confusion matrix for the classification random forest of BanFakeNews dataset

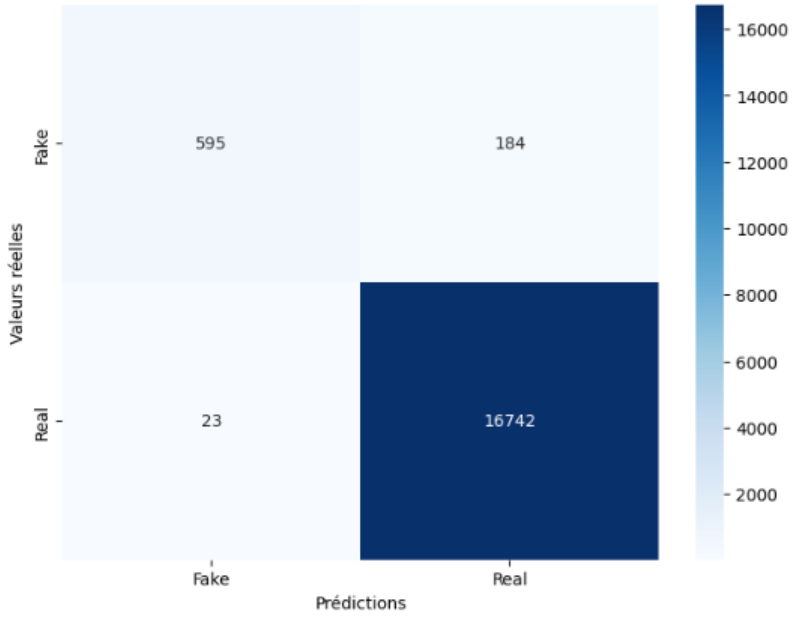

Figure 5.8: *Confusion matrix of BanFakeNews dataset*

Figure 5.8 displays a confusion matrix for the RandomForestClassifier model shows that it correctly classified 595 "Fake" instances and 16742 "Real" instances. However, it made 184 false positives and 23 true negatives.

5.2.5 fake-and-real-news-dataset

- **Supervised Techniques :**

Table 5.17 provides a summary of the calculated metrics for supervised training models on the fake-and-real-news-dataset :

fake-and-real-news-dataset					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Random Forest	0.989	0.99	0.99	0.99	8.867
KNN (K=3)	0.86	0.92	0.81	0.86	8.744
KNN (K=5)	0.87	0.93	0.81	0.86	8.732
KNN (K=7)	0.86	0.93	0.81	0.86	8.732
KNN (K=9)	0.87	0.93	0.80	0.86	8.718
SVM	0.991	0.99	0.99	0.99	8.864
Naive Bayes	0.924	0.92	0.94	0.93	8.897
Decision Tree	0.995	1.00	1.00	1.00	8.865
GB	0.995	1.00	0.99	1.00	8.862

Table 5.17: Evaluation of supervised algorithms for fake-and-real-news-dataset.

Table 5.17 presents the performance of different algorithms on the fake-and-real-news-dataset with a 70% train and 30% test split. It lists accuracy, precision, recall, F1 score, and entropy for each algorithm. Decision Tree and Gradient Boosting achieved the highest accuracy of 0.995, followed closely by SVM at 0.991 and Random Forest at 0.989. KNN algorithms showed slightly lower accuracy, ranging from 0.86 to 0.87. Naive Bayes performed well with an accuracy of 0.924. Overall, Decision Tree and Gradient Boosting were the best performers in detecting fake and real news.

- **Unsupervised Techniques :**

  Table 5.18 provides a summary of the calculated metrics for unspervised training models on the fake-and-real-news-dataset :

fake-and-real-news-dataset					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Kmeans	0.460	0.49	0.63	0.55	9.110

Table 5.18: Evaluation of unsupervised algorithms for fake-and-real-news-dataset dataset.

Table 5.18 shows the performance of the Kmeans algorithm on the fake-and-real-news dataset, using a 70% train and 30% test split. Kmeans achieved an accuracy of 0.460, precision of 0.49, recall of 0.63, F1 score of 0.55, and entropy of 9.110. This indicates that Kmeans has moderate recall but lower overall accuracy and precision in detecting fake and real news.

- **Time :**

  Table 5.19 provides a summary of the testing time and training time in seconds for fake-and-real-news-dataset :

fake-and-real-news-dataset							
Time	Train-Test size 70% - 30%						
	RF	**KNN**	**SVM**	**NB**	**DT**	**GB**	**KM**
Training time	97.079	0.022	1547.40	0.081	29.986	210.354	10.852
Testing time	1.4741	7904.44	220.479	0.022	0.022	0.049	0.009

Table 5.19: Training and Testing Times for Various Algorithms for fake-and-real-news-dataset .

Table 5.19 summarizes the training and testing times for various algorithms applied to the fake-and-real-news-dataset. Random Forest (RF) has a moderate training time of 97.079 seconds and a quick testing time of 1.4741 seconds. K-Nearest Neighbors (KNN) is the fastest to train at 0.022 seconds but has the longest testing time at 7904.44 seconds. Support Vector Machine (SVM) has a significantly high training

time of 1547.40 seconds and a testing time of 220.479 seconds. Naive Bayes (NB) is very efficient with minimal training (0.081 seconds) and testing (0.022 seconds) times. Decision Tree (DT) and Gradient Boosting (GB) show balanced training times of 29.986 and 210.354 seconds, respectively, with quick testing times. K-means (KM) has a relatively short training time of 10.852 seconds and the fastest testing time of 0.009 seconds.

- **Size :**

  Table 5.20 provides a summary of the size before and after preprocessing for fake-and-real-news-dataset dataset:

fake-and-real-news-dataset	
	(row,column)
Before preprocessing	(44898, 5)
After preprocessing	(44898, 2)

Table 5.20: The size before and after preprocessing for Various Algorithms for fake-and-real-news-dataset .

- **Correlation matrix:**

  In the figure 5.9, the correlation matrix between the classification models of fake-and-real-news-dataset

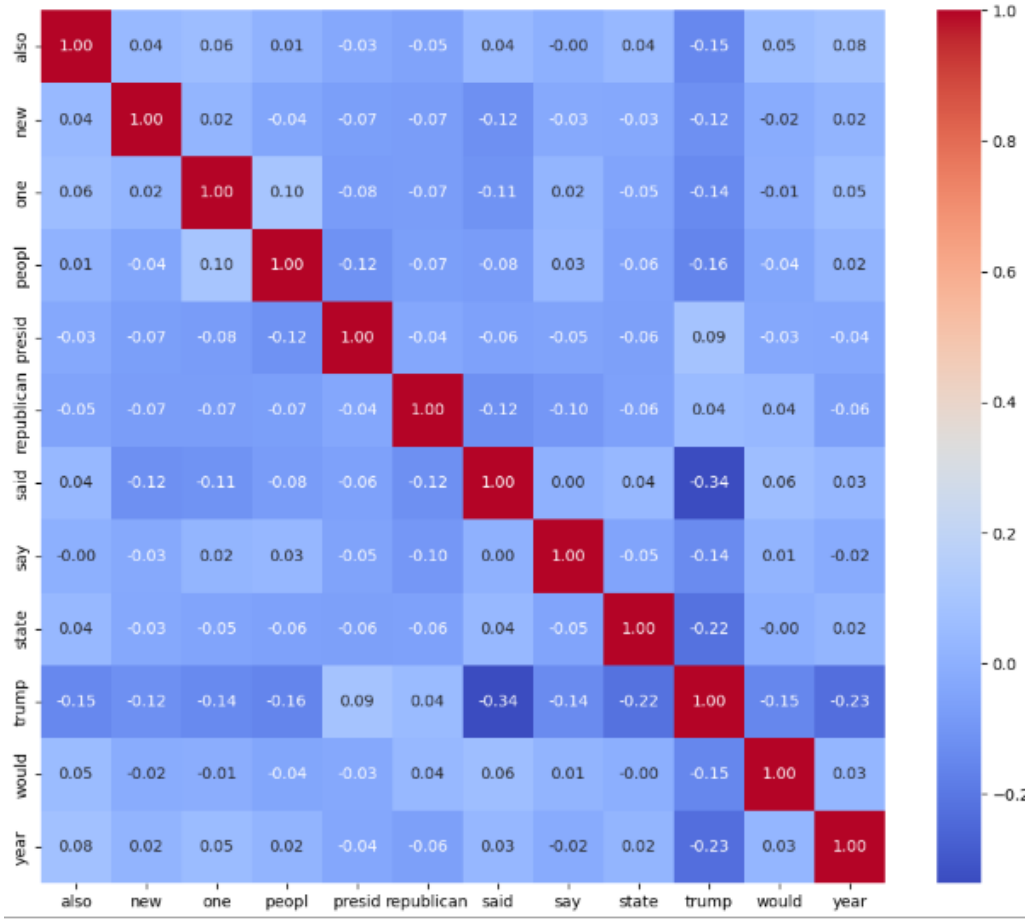

Figure 5.9: *Correlation matrix of fake-and-real-news-dataset*

Figure 5.9 shows a correlation matrix, which is a visual representation of the relationships between different variables. The numbers in the matrix represent the correlation coefficients between each pair of variables. A correlation coefficient of 1 indicates a perfect positive correlation, while a correlation coefficient of -1 indicates a perfect negative correlation. A correlation coefficient of 0 indicates no correlation. In this matrix, the correlation coefficients are generally small, suggesting that there is not a strong linear relationship between most of the variables. For example, the correlation coefficient between ”said” and ”trump” is -0.34, indicating a moderate negative correlation. This means that as the frequency of the word ”said” increases, the frequency of the word ”trump” tends to decrease. However, the correlation coefficients between ”said” and the other variables are much smaller, indicating that there is not a strong relationship between ”said” and those variables.

- **Confusion matrix:** In the figure 5.10, the confusion matrix for the classification random forest of fake-and-real-news-dataset

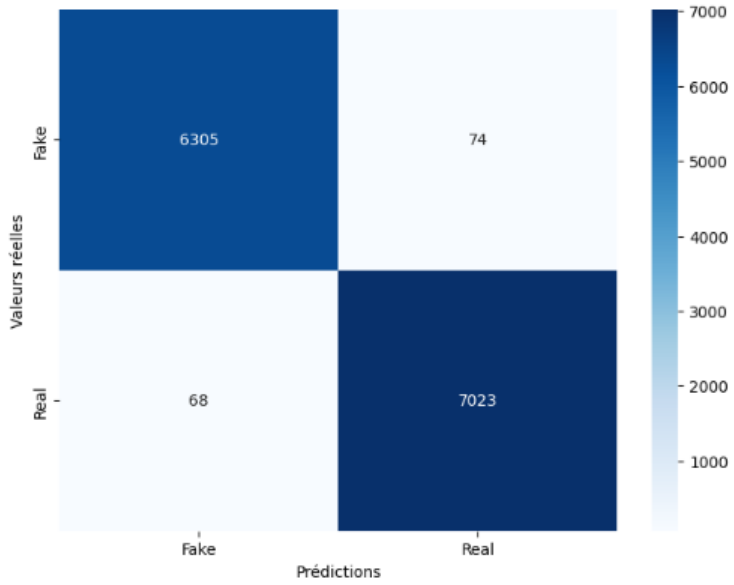

Figure 5.10: *Confusion matrix of fake-and-real-news-dataset*

Figure 5.10 it's a confusion matrix for the RandomForestClassifier model shows that it correctly classified 6305 "Fake" instances and 7023"Real" instances. However, it made 74 false positives and 68 true negatives.

5.2.6 Fake News Detection Datasets

- **Supervised Techniques :**

  Table 5.21 provides a summary of the calculated metrics for supervised training models on the Fake News Detection Datasets dataset:

Fake News Detection Datasets					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Random Forest	0.990	0.99	0.99	0.99	8.758
KNN (K=3)	0.73	0.92	0.47	0.62	8.075
KNN (K=5)	0.69	0.94	0.38	0.54	7.859
KNN (K=7)	0.67	0.93	0.32	0.48	7.700
KNN (K=9)	0.65	0.94	0.28	0.43	7.554
SVM	0.990	0.99	0.99	0.99	8.764
Naive Bayes	0.922	0.93	0.91	0.92	8.739
Decision Tree	0.994	0.99	0.99	0.99	8.759
GB	0.995	0.99	1.00	0.99	8.764

Table 5.21: Evaluation of supervised algorithms for Fake News Detection Datasets.

Table 5.21 summarizes the performance of various algorithms used for fake news detection with a 70% - 30% train-test split. Random Forest, SVM, and Decision Tree all show high accuracy, precision, recall, and F1 scores of around 0.99. Gradient Boosting performs slightly better with the highest accuracy of 0.995. Naive Bayes also performs well with an accuracy of 0.922. K-Nearest Neighbors (KNN) shows decreasing performance as the value of K increases, with the lowest accuracy at K=9. Entropy values are provided for each algorithm, with the highest around 8.764 for SVM and Gradient Boosting. Overall, Decision Tree and Gradient Boosting exhibit the best performance.

- **Unsupervised Techniques :**

  Table 5.22 provides a summary of the calculated metrics for unspervised training models on the Fake News Detection Datasets dataset:

Fake News Detection Datasets					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Kmeans	0.541	0.51	0.73	0.60	9.112

Table 5.22: Evaluation of unsupervised algorithms for Fake News Detection Datasets dataset.

Table 5.22 shows the performance of the K-means algorithm for fake news detection with a 70% - 30% train-test split. It has an accuracy of 0.541, precision of 0.51, recall of 0.73, F1 score of 0.60, and an entropy value of 9.112. Overall, K-means has moderate performance, with higher recall but lower accuracy and precision..

- **Time :**

Table 5.23 provides a summary of the testing time and training time in seconds for Fake News Detection Datasets dataset:

Fake News Detection Datasets							
Time	Train-Test size 70% - 30%						
	RF	**KNN**	**SVM**	**NB**	**DT**	**GB**	**KM**
Training time	77.722	0.026	1592.94	0.063	29.016	199.789	12.112
Testing time	1.228	7280.59	217.591	0.015	0.023	0.035	0.012

Table 5.23: Training and Testing Times for Various Algorithms for Fake News Detection Datasets dataset.

Table 5.23 presents the training and testing times for various algorithms applied to the Fake News Detection Datasets . The SVM algorithm has the longest training time at 1592.94 seconds and a substantial testing time of 217.591 seconds. In contrast, KNN shows the longest testing time at 7280.59 seconds but a very short training time of 0.026 seconds. Algorithms like Naive Bayes and Decision Tree have

both low training and testing times, making them efficient in terms of computational resources. Random Forest and Gradient Boosting have moderate training and testing times, balancing between efficiency and performance. K-means clustering shows a moderate training time (12.112 seconds) and the shortest testing time (0.012 seconds), indicating its efficiency in both stages.

- **Size :**

  Table 5.24 provides a summary of the size before and after preprocessing for Fake News Detection Datasets dataset:

Fake News Detection Datasets	
	(row,column)
Before preprocessing	(44898, 5)
After preprocessing	(44898, 2)

Table 5.24: The size before and after preprocessing for Various Algorithms for Fake News Detection Datasets dataset.

- **Correlation matrix:**

  In the figure 5.11, the correlation matrix between the classification models of Fake News Detection Datasets dataset

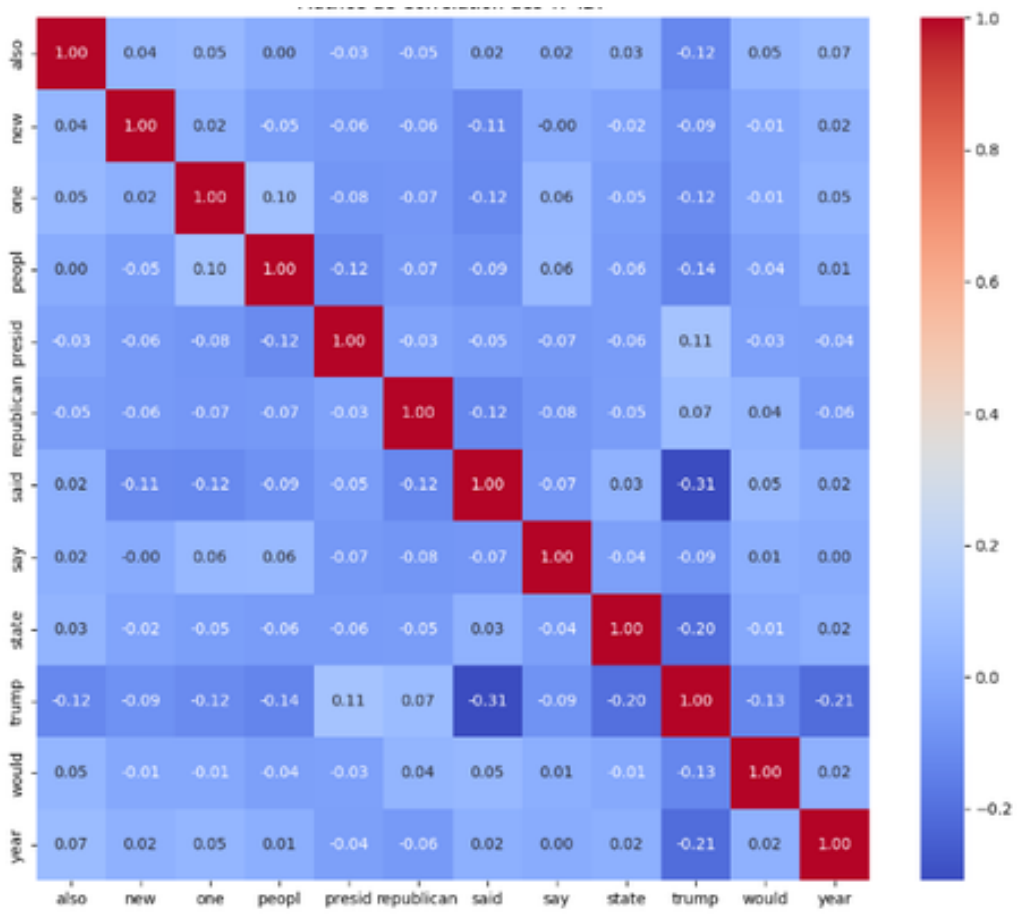

Figure 5.11: *Correlation matrix of Fake News Detection Datasets dataset*

Figure 5.11 presents heat map shows the correlation between words in a dataset. The most strongly correlated words are ”president” and ”republican”, with a correlation of 1. The words ”say” and ”said” are also strongly correlated, with a correlation of 0.99. The words ”say” and ”state” have a negative correlation of -0.31, meaning they rarely appear together. The words ”also” and ”year” have a correlation of 0.07, meaning they sometimes appear together. The heat map helps identify words that frequently appear together in the data and learn more about their relationships.

- **Confusion matrix:** In the figure 5.12, the confusion matrix for the classification random forest of Fake News Detection Datasets dataset

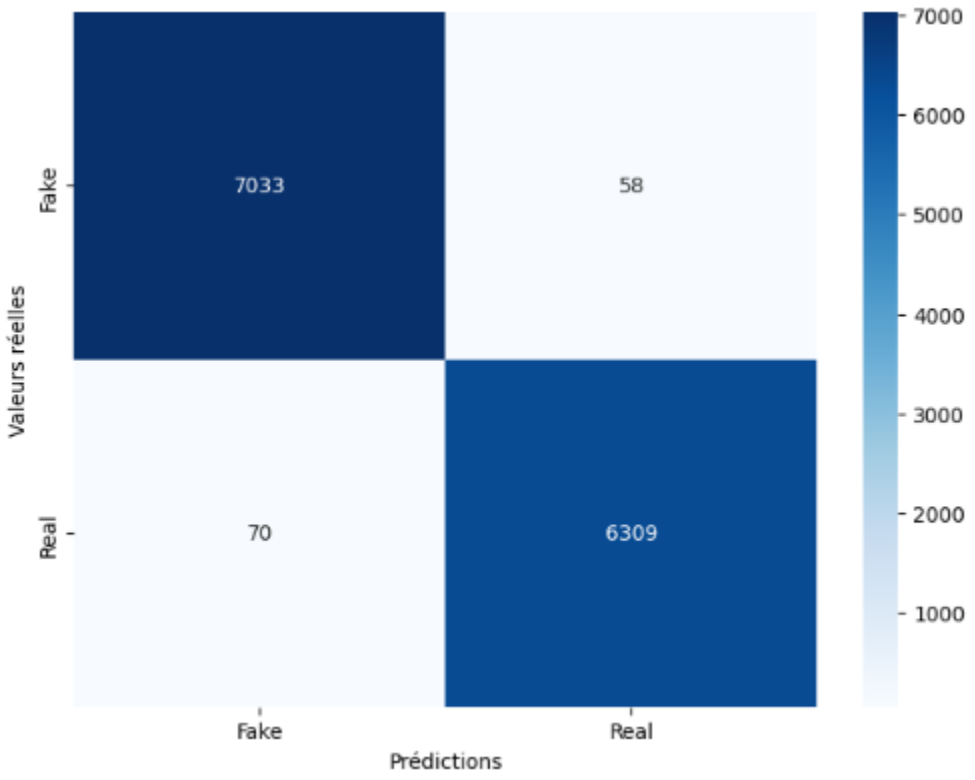

Figure 5.12: *Confusion matrix of Fake News Detection Datasets dataset*

Figure 5.12 presents the confusion matrix for the RandomForestClassifier model shows that it correctly classified 7033 "Fake" instances and 6309"Real" instances. However, it made 70 false positives and 58 true negatives.

5.2.7 Fake news detection data

- **Supervised Techniques :**

Table 5.25 provides a summary of the calculated metrics for supervised training models on the Fake news detection data dataset:

Fake news detection data					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Random Forest	0.794	0.80	0.99	0.89	8.951
KNN (K=3)	0.79	0.80	0.98	0.88	8.936
KNN (K=5)	0.79	0.80	0.99	0.89	8.952
KNN (K=7)	0.80	0.80	0.99	0.89	8.956
KNN (K=9)	0.80	0.80	1.00	0.89	8.960
SVM	0.799	0.80	1.00	0.89	8.961
Naive Bayes	0.797	0.80	0.99	0.89	8.955
Decision Tree	0.683	0.80	0.80	0.80	8.738
GB	0.799	0.80	1.00	0.89	8.961

Table 5.25: Evaluation of supervised algorithms for Fake news detection data.

Table5.25 summarizes the performance metrics of various supervised learning models applied to a fake news detection dataset. The models were evaluated using a train-test split of 70% training data and 30% testing data. The table includes the following metrics for each algorithm: Accuracy, Precision, Recall, F1 Score, and Entropy. The evaluated algorithms are Random Forest, K-Nearest Neighbors (KNN) with different values of K (3, 5, 7, and 9), Support Vector Machine (SVM), Naive Bayes, Decision Tree, and Gradient Boosting (GB). This summary provides a comprehensive comparison of how each algorithm performs in detecting fake news based on the listed metrics.

- **Unsupervised Techniques :**

  Table 5.26 provides a summary of the calculated metrics for unspervised training models on the Fake news detection data dataset:

Fake news detection data					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Kmeans	0.643	0.79	0.75	0.77	8.678

Table 5.26: Evaluation of unsupervised algorithms for Fake news detection data dataset.

Table 5.26 provides a summary of the performance metrics for an unsupervised learning model, specifically K-means, applied to a fake news detection dataset. The data was divided into a train-test split of 70% training data and 30% testing data. The metrics included in the table are Accuracy, Precision, Recall, F1 Score, and Entropy. This summary highlights the effectiveness of the K-means algorithm in identifying fake news based on these evaluation metrics.

- **Time :**

Table 5.27 provides a summary of the testing time and training time in seconds for Fake news detection data dataset:

Fake news detection data							
Time	Train-Test size 70% - 30%						
	RF	**KNN**	**SVM**	**NB**	**DT**	**GB**	**KM**
Training time	155.016	0.023	2021.9	0.051	239.21	211.41	16.021
Testing time	2.240	7065.12	351.35	0.014	0.027	0.037	0.011

Table 5.27: Training and Testing Times for Various Algorithms for Fake news detection data dataset.

Table 5.27 provides the training and testing times for various algorithms on the Fake News Detection dataset . Random Forest has a moderate training time of 155.016 seconds and a quick testing time of 2.240 seconds. KNN, despite its minimal training time of 0.023 seconds, has the longest testing time of 7065.12 seconds, indicating a

high computational cost during prediction. SVM stands out with the highest training time of 2021.9 seconds and a substantial testing time of 351.35 seconds. Naive Bayes is extremely efficient, with both training and testing times being the shortest at 0.051 and 0.014 seconds, respectively. Decision Tree and Gradient Boosting have relatively higher training times of 239.21 and 211.41 seconds but maintain low testing times. K-means shows a moderate training time of 16.021 seconds and the fastest testing time at 0.011 seconds. These results highlight significant variations in computational efficiency, with Naive Bayes and K-means being the most time-efficient algorithms overall.

- **Size :**

  Table 5.28 provides a summary of the size before and after preprocessing for Fake news detection data dataset:

Fake news detection data	
	(row,column)
Before preprocessing	(2600, 5)
After preprocessing	(2600, 2)

Table 5.28: The size before and after preprocessing for Various Algorithms for Fake news detection data dataset.

- **Correlation matrix:**

  In the figure 5.13, the correlation matrix between the classification models of Fake news detection data dataset

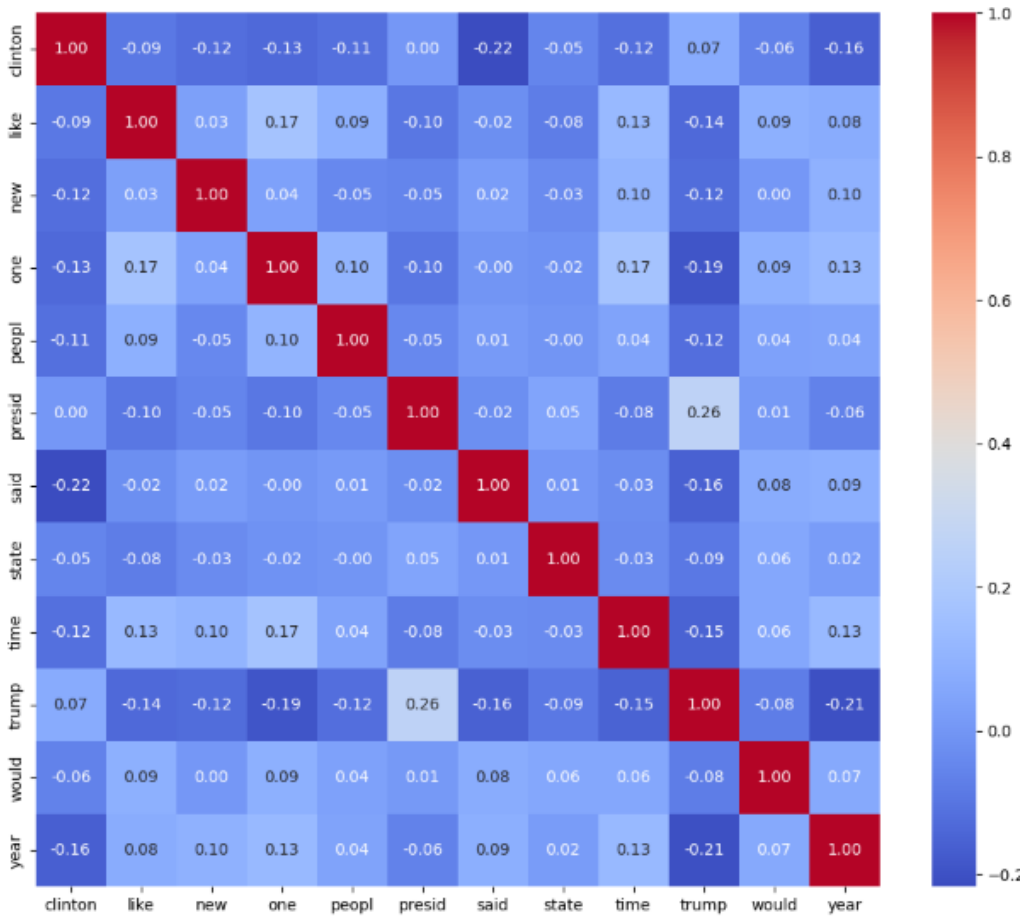

Figure 5.13: *Correlation matrix of Fake news detection data dataset*

Figure 5.13 shows a correlation matrix, which displays the correlations between different pairs of words. The correlation values range from -1 to 1, where 1 indicates a perfect positive correlation, -1 indicates a perfect negative correlation and 0 indicates no correlation. In this correlation matrix, words such as ”clinton”, ”trump” and ”said” show negative correlations with the word ”presid”. This could suggest that these words are often used in contexts opposed to the presidency. It is important to note that correlation does not necessarily imply causation. For example, the fact that ”clinton” and ”presid” have a negative correlation does not mean that ”clinton” causes a decrease in the use of ”presid”.

- **Confusion matrix:** In the figure 5.14, the confusion matrix for the classification random forest of Fake news detection data dataset

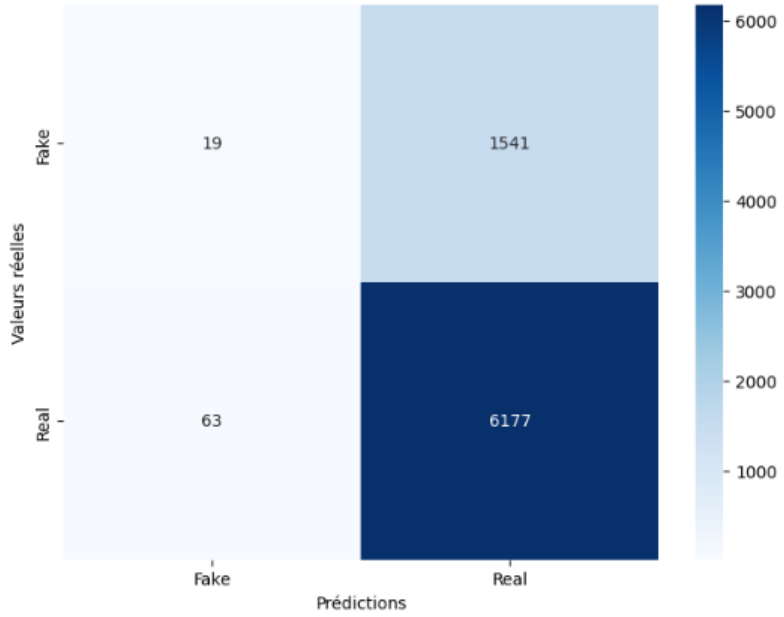

Figure 5.14: *Confusion matrix of Fake news detection data dataset*

Figure 5.14 presents the confusion matrix for the RandomForestClassifier model shows that it correctly classified 19 "Fake" instances and 6177 "Real" instances. However, it made 1541 false positives and 63 true negatives.

5.2.8 Fake news Detection data

- **Supervised Techniques :**

  Table 5.29 provides a summary of the calculated metrics for supervised training models on the Fake news Detection data dataset:

Fake news Detection data					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Random Forest	0.907	0.94	0.87	0.90	7.952
KNN (K=3)	0.83	0.87	0.78	0.82	7.92
KNN (K=5)	0.84	0.89	0.78	0.83	7.91
KNN (K=7)	0.84	0.89	0.78	0.83	7.90
KNN (K=9)	0.84	0.89	0.78	0.83	7.90
SVM	0.945	0.95	0.94	0.94	8.020
Naive Bayes	0.837	0.98	0.68	0.81	7.669
Decision Tree	0.893	0.89	0.89	0.89	8.034
GB	0.933	0.93	0.94	0.93	8.038

Table 5.29: Evaluation of supervised algorithms for Fake news Detection data.

Table 5.29 shows the performance of various algorithms on the Fake News Detection dataset, using a 70% train and 30% test split. SVM achieved the highest accuracy at 0.945, followed by Gradient Boosting at 0.933 and Random Forest at 0.907. KNN algorithms had consistent accuracy around 0.84. Naive Bayes had lower accuracy but high precision. Decision Tree also performed well with an accuracy of 0.893. Overall, SVM and Gradient Boosting were the top performers.

- **Unsupervised Techniques :**

  Table 5.30 provides a summary of the calculated metrics for unuspervised training models on the Fake news Detection data dataset:

Fake news Detection data					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Kmeans	0.533	1.00	0.06	0.1	5.135

Table 5.30: Evaluation of unsupervised algorithms for Fake news Detection data dataset.

Table 5.30 shows the performance of the Kmeans algorithm on the Fake News Detection dataset, using a 70% train and 30% test split. Kmeans achieved an accuracy of 0.533, with perfect precision at 1.00, but very low recall at 0.06, resulting in an F1 score of 0.1 and entropy of 5.135. This indicates that while Kmeans is highly precise.

- **Time :**

  Table 5.31 provides a summary of the testing time and training time in seconds for Fake news Detection data dataset:

Fake news Detection data							
Time	Train-Test size 70% - 30%						
	RF	**KNN**	**SVM**	**NB**	**DT**	**GB**	**KM**
Training time	53.470	0.015	815.21	0.045	36.967	168.789	6.132
Testing time	0.840	4543.15	137.70	0.027	0.017	0.028	0.012

Table 5.31: Training and Testing Times for Various Algorithms for Fake news Detection data dataset.

Table 5.31 provides an overview of the training and testing times for various algorithms on the Fake News Detection dataset. KNN has the shortest training time of 0.015 seconds but the longest testing time of 4543.15 seconds. SVM, on the other hand, has the longest training time at 815.21 seconds but a relatively shorter testing time of 137.70 seconds. Naive Bayes and K-means have the shortest combined times,

with training times of 0.045 and 6.132 seconds and testing times of 0.027 and 0.012 seconds, respectively. Random Forest, Decision Tree, and Gradient Boosting have moderate training and testing times, balancing both aspects. This highlights the trade-off between training and testing times among different algorithms.

- **Size :**

  Table 5.32 provides a summary of the size before and after preprocessing for Fake news Detection data dataset:

Fake news Detection data	
	(row,column)
Before preprocessing	(20800, 5)
After preprocessing	(20800, 2)

Table 5.32: The size before and after preprocessing for Various Algorithms for Fake news Detection data dataset.

- **Correlation matrix:**

  In the figure 5.15, the correlation matrix between the classification models of Fake news Detection data dataset

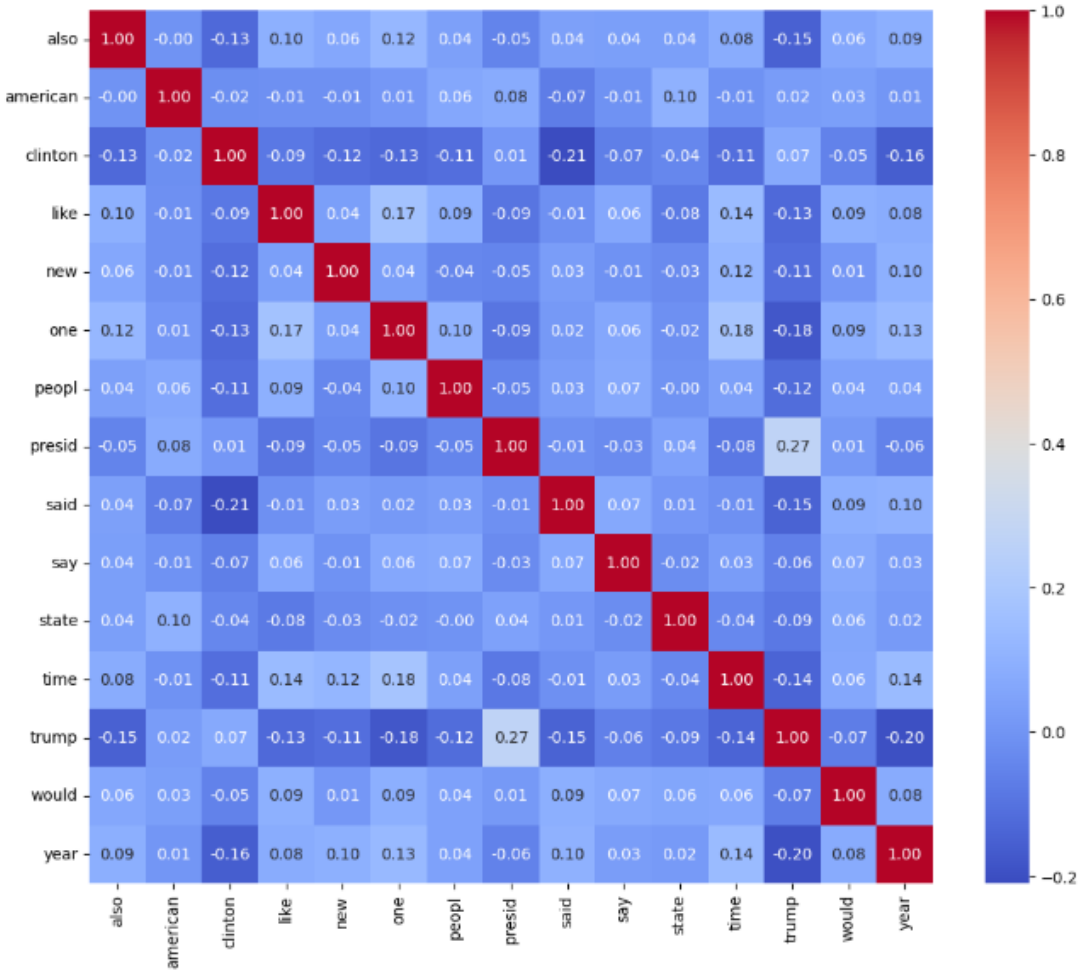

Figure 5.15: *Correlation matrix of Fake news Detection data dataset*

Figure 5.15 shows a correlation matrix. The correlation matrix shows the correlation between different words. The words are likely from a text corpus, and the correlation measures how often they appear together. For example, the word ”trump” is highly correlated with the word ”president,” which makes sense given the context of the text corpus. The correlation matrix can be used to understand the relationships between words and to identify patterns in the text corpus.

- **Confusion matrix:** In the figure 5.16, the confusion matrix for the classification random forest of Fake news Detection data dataset

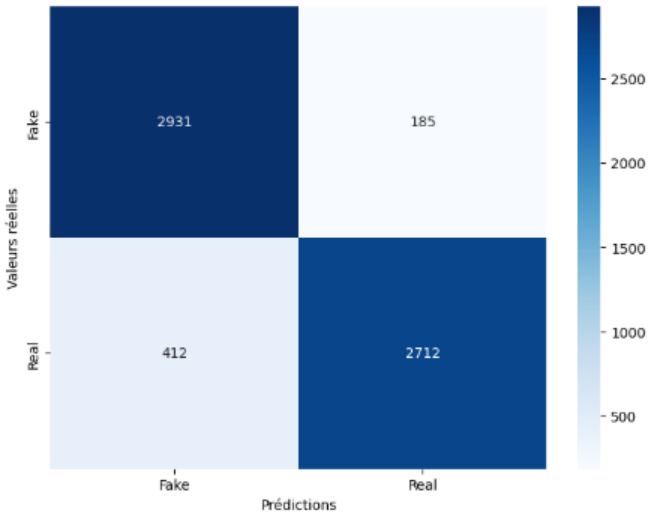

Figure 5.16: *Confusion matrix of Fake news Detection data dataset*

Figure 5.16 provides the confusion matrix for the RandomForestClassifier model shows that it correctly classified 2931 ”Fake” instances and 2712 ”Real” instances. However, it made 412 false positives and 185 true negatives.

5.2.9 Bangla And English Fake News Detection DataSet

- **Supervised Techniques :**

  Table 5.33 provides a summary of the calculated metrics for supervised training models on the Bangla And English Fake News Detection DataSet dataset:

Bangla And English Fake News Detection DataSet					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Random Forest	0.979	0.97	0.98	0.98	7.331
KNN (K=3)	0.83	0.78	0.92	0.85	7.481
KNN (K=5)	0.83	0.78	0.93	0.85	7.492
KNN (K=7)	0.84	0.79	0.93	0.86	7.486
KNN (K=9)	0.84	0.79	0.93	0.86	7.487
SVM	0.977	0.97	0.99	0.98	7.340
Naive Bayes	0.915	0.91	0.93	0.92	7.346
Decision Tree	0.99	0.99	0.99	0.99	7.323
GB	0.993	0.99	1.00	0.99	7.329

Table 5.33: Evaluation of supervised algorithms for Bangla And English Fake News Detection DataSet.

Table 5.33 compares various methods for detecting fake news in Bangla and English, using a 70% training and 30% testing split. It includes metrics like accuracy, precision, recall, F1 score, and entropy. Random Forest performed well with high accuracy, precision, recall, and F1 score. Decision Tree and Gradient Boosting also showed strong performance. K-Nearest Neighbors (KNN) had different results with different K values, while Support Vector Machine (SVM) achieved good accuracy. Overall, Decision Tree and Gradient Boosting stood out as top performers in detecting fake news.

- **Unsupervised Techniques :**

  Table 5.34 provides a summary of the calculated metrics for unspervised training models on the Bangla And English Fake News Detection DataSet dataset:

Bangla And English Fake News Detection DataSet					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Kmeans	0.539	0.53	0.70	0.61	7.601

Table 5.34: Evaluation of unsupervised algorithms for Bangla And English Fake News Detection DataSet dataset.

Table 5.34 presents the performance of algorithms on the Bangla and English Fake News Detection DataSet, using a 70% train-test split. Only one algorithm, Kmeans, is listed, showing an accuracy of 0.539 along with precision of 0.53, recall of 0.70, and F1 score of 0.61.

- **Time :**

  Table 5.35 provides a summary of the testing time and training time in seconds for Bangla And English Fake News Detection DataSet dataset:

Bangla And English Fake News Detection DataSet							
Time	Train-Test size 70% - 30%						
	RF	**KNN**	**SVM**	**NB**	**DT**	**GB**	**KM**
Training time	7.308	0.005	59.561	00.014	2.425	41.089	4.548
Testing time	0.294	392.50	20.251	0.003	0.007	0.014	0.002

Table 5.35: Training and Testing Times for Various Algorithms for Bangla And English Fake News Detection DataSet dataset.

Table 5.35 illustrates the training and testing times for different algorithms employed in detecting fake news within the Bangla And English Fake News Detection DataSet dataset. Among the algorithms, Decision Tree (DT) exhibits the shortest training time at 2.425 seconds, while Random Forest (RF) requires the most time for training, totaling 7.308 seconds. For testing, K-means (KM) proves to be the

fastest with a time of 0.002 seconds, whereas KNN (K-nearest neighbors) takes the longest at 392.50 seconds. These figures indicate significant variations in computational efficiency across algorithms, with DT and KM being the quickest to train and test, respectively, while RF and KNN demand more time.The table illustrates the training and testing times for different algorithms employed in detecting fake news within the Bangla And English Fake News Detection DataSet dataset. Among the algorithms, Decision Tree (DT) exhibits the shortest training time at 2.425 seconds, while Random Forest (RF) requires the most time for training, totaling 7.308 seconds. For testing, K-means (KM) proves to be the fastest with a time of 0.002 seconds, whereas KNN (K-nearest neighbors) takes the longest at 392.50 seconds. These figures indicate significant variations in computational efficiency across algorithms, with DT and KM being the quickest to train and test, respectively, while RF and KNN demand more time.

- **Size :**

  Table 5.36 provides a summary of the size before and after preprocessing for Bangla And English Fake News Detection DataSet dataset:

Bangla And English Fake News Detection DataSet	
	(row,column)
Before preprocessing	(10000, 3)
After preprocessing	(10000, 2)

Table 5.36: The size before and after preprocessing for Various Algorithms for Bangla And English Fake News Detection DataSet dataset.

- **Correlation matrix:**

  In the figure 5.17, the correlation matrix between the classification models of Bangla And English Fake News Detection DataSet.

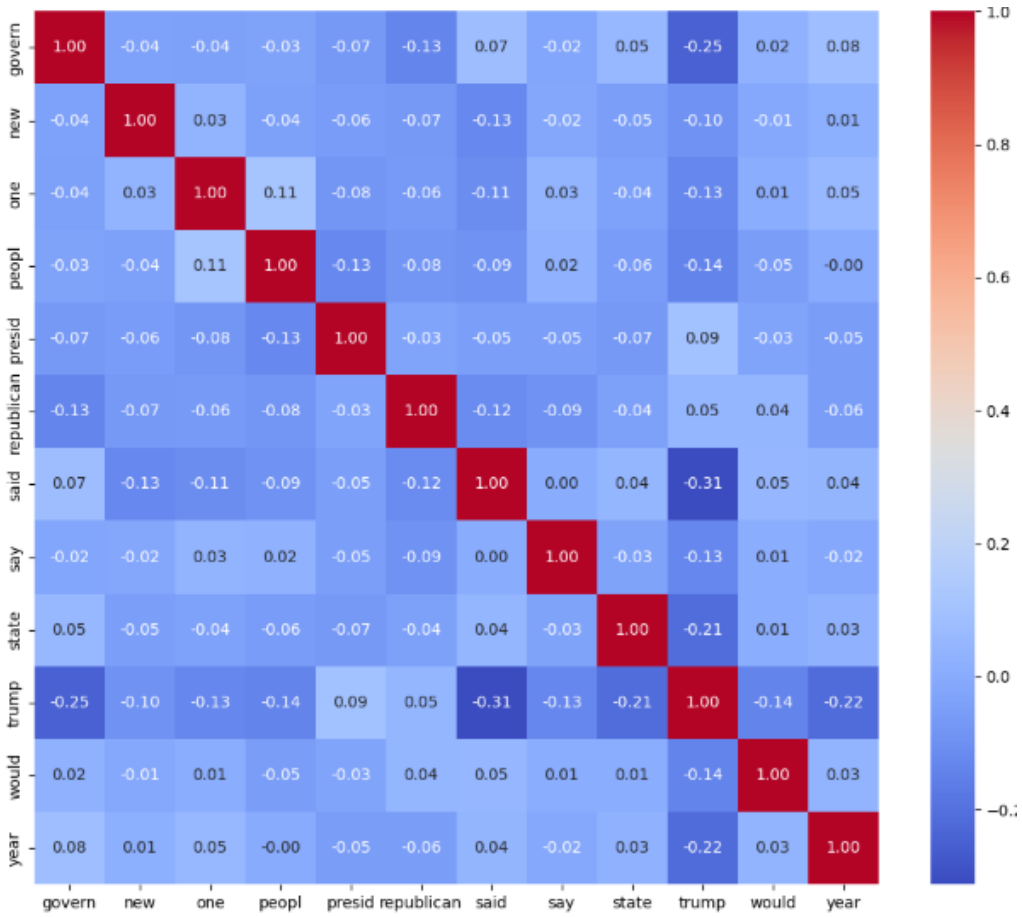

Figure 5.17: *Correlation matrix of Bangla And English Fake News Detection DataSet dataset*

Figure 5.17 present The correlation matrix shows that the words ”said” and ”republican” are highly correlated, with a correlation coefficient of 1.0. This suggests that these words often occur together in the text. The words ”trump” and ”state” are also highly correlated, with a correlation coefficient of -0.31. This suggests that these words often occur in opposition to each other in the text. Overall, the correlation matrix provides insights into the relationships between words in the text, which can be useful for understanding the overall themes and sentiment of the text.

- **Confusion matrix:** In the figure 5.18, the confusion matrix for the classification random forest of spanish Political Fake News dataset.

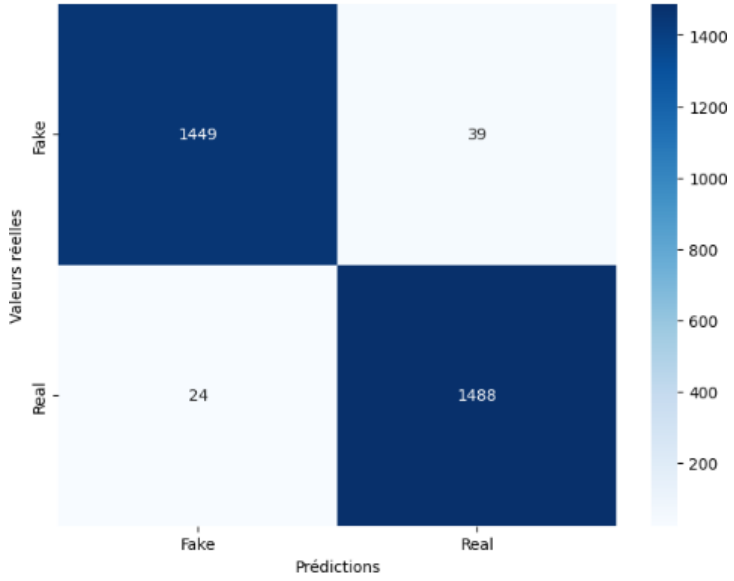

Figure 5.18: *Confusion matrix of Bangla And English Fake News Detection DataSet dataset*

Figure 5.18 presents the confusion matrix for the RandomForestClassifier model shows that it correctly classified 1449 "Fake" instances and 1488 "Real" instances. However, it made 39 false positives and 24 true negatives.

5.2.10 Detection of Fake News

- **Supervised Techniques :**

  Table 5.37 provides a summary of the calculated metrics for supervised training models on the Detection of Fake News dataset:

Detection of Fake News					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Random Forest	0.904	0.94	0.87	0.90	7.971
KNN (K=3)	0.84	0.78	0.94	0.85	7.025
KNN (K=5)	0.571	0.54	0.99	0.70	8.654
KNN (K=7)	0.83	0.77	0.94	0.85	7.041
KNN (K=9)	0.83	0.76	0.94	0.84	7.048
SVM	0.944	0.95	0.94	0.94	8.030
Naive Bayes	0.837	0.98	0.69	0.81	7.691
Decision Tree	0.896	0.89	0.90	0.90	8.058
GB	0.935	0.93	0.94	0.94	8.057

Table 5.37: Evaluation of supervised algorithms for Detection of Fake News.

Table 5.37 compares various algorithms used for detecting fake news with a 70%-30% train-test split. It includes metrics like Accuracy, Precision, Recall, F1 score, and Entropy. The Support Vector Machine (SVM) achieved the highest accuracy at 0.923, followed by Random Forest at 0.895. K-Nearest Neighbors (KNN) and Gradient Boosting also showed good performance. Naive Bayes and Decision Tree had lower accuracies, with Naive Bayes having the highest recall.

- **Unsupervised Techniques :**

  Table 5.38 provides a summary of the calculated metrics for unspervised training models on the Detection of Fake News dataset:

Detection of Fake News					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Kmeans	0.526	0.55	0.27	0.37	7.340

Table 5.38: Evaluation of unsupervised algorithms for Detection of Fake News dataset.

Table 5.38 shows the performance of the K-means algorithm in detecting fake news with a 70%-30% train-test split. The K-means algorithm achieved an accuracy of 0.363, precision of 0.40, recall of 0.61, F1 score of 0.48, and an entropy of 7.249, indicating overall modest performance

- **Time :**

Table 5.39 provides a summary of the testing time and training time in seconds for Detection of Fake News dataset:

Detection of Fake News							
Time	Train-Test size 70% - 30%						
	RF	**KNN**	**SVM**	**NB**	**DT**	**GB**	**KM**
Training time	7.365	0.005	637.439	00.011	4.505	43.754	2.982
Testing time	0.287	8432.43	516.726	0.003	0.004	0.013	0.026

Table 5.39: The size before and after preprocessing for Various Algorithms for Detection of Fake News dataset.

Table 5.39 shows the training and testing times for different algorithms used to detect fake news. KNN has the shortest training time (0.005) but the longest testing time (8432.43). SVM has the longest training time (637.439) and also a long testing time (516.726). Naive Bayes is the fastest, with a training time of (0.011) and a testing time of (0.003). Random Forest has a training time of (7.365) and a testing time of(0.287). Decision Tree trains in (4.505) and tests in (0.004). Gradient Boosting has

a training time of (43.754) and a testing time of (0.013). K-means trains in (2.982) and tests in (0.026).

- **Size :**

  Table 5.40 provides a summary of the size before and after preprocessing for Detection of Fake News dataset:

Detection of Fake News	
	(row,column)
Before preprocessing	(6335, 4)
After preprocessing	(6335, 2)

Table 5.40: Training and Testing Times for Various Algorithms for Detection of Fake News dataset.

- **Correlation matrix:**

  In the figure 5.19, the correlation matrix between the classification models of Detection of Fake News dataset.

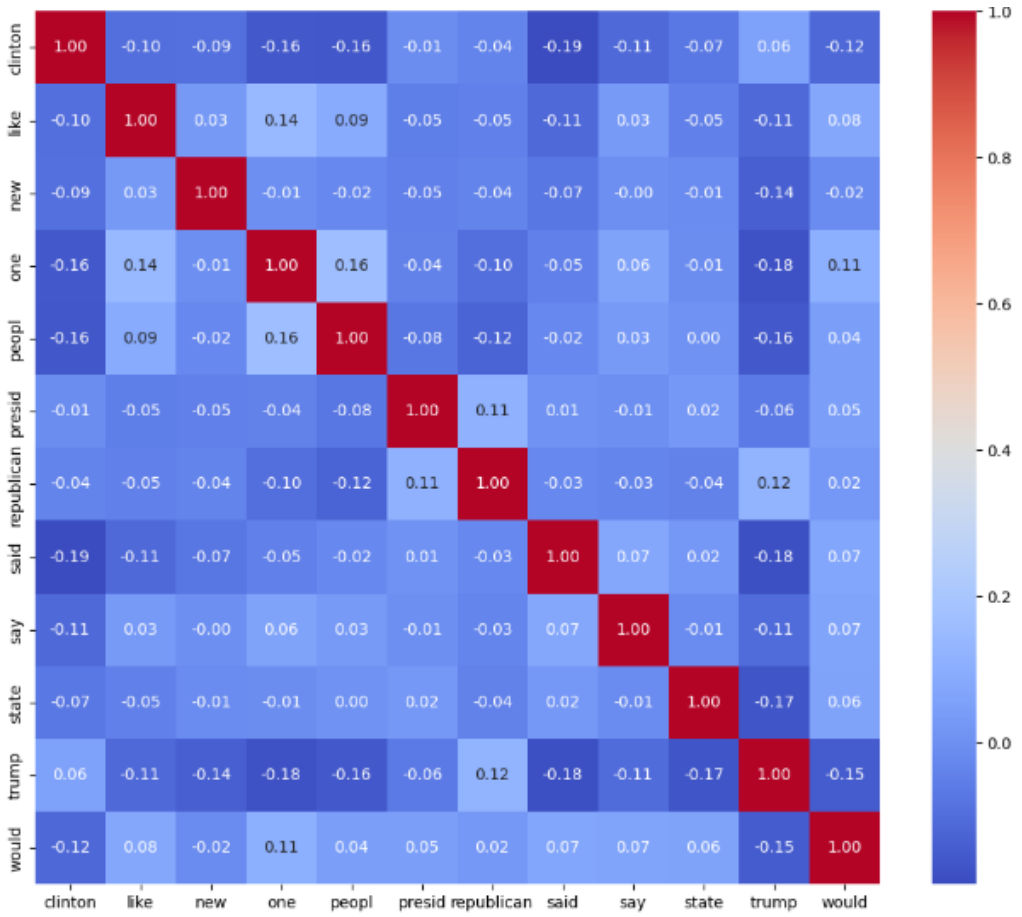

Figure 5.19: *Correlation matrix of Detection of Fake News dataset*

Figure 5.19 shows a correlation matrix. It's a visualization of how different words are related to each other, based on their co-occurrence in a set of text data. The correlation matrix suggests that the words "said," "republican," and "presid" are strongly correlated, which makes sense given that they are often used in the context of political discourse. The words "trump" and "would" are also highly correlated, likely due to their frequent use in discussing political policies and plans. On the other hand, words like "like" and "new" show weaker correlations with other words, suggesting that they are used more broadly in different contexts.

- **Confusion matrix:** In the figure 5.20, the confusion matrix for the classification random forest of Detection of Fake News dataset.

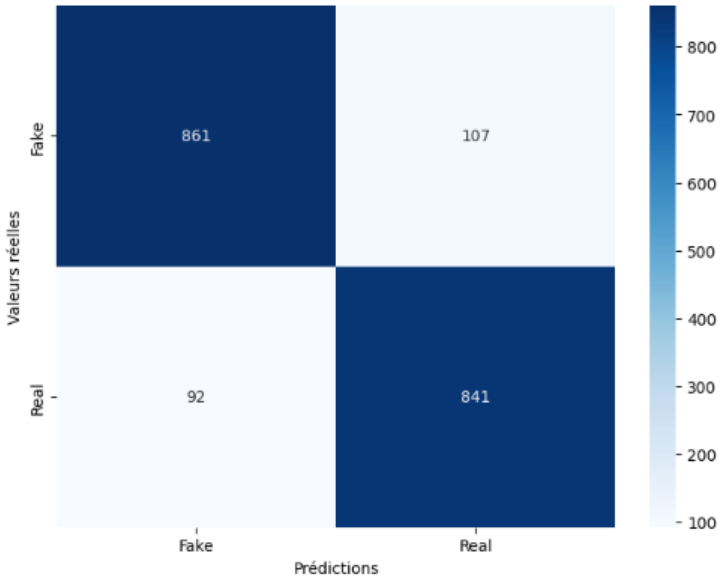

Figure 5.20: *Confusion matrix of Detection of Fake News dataset*

Figure 5.20 presents the confusion matrix for the RandomForestClassifier model shows that it correctly identified 861 ”Fake” and 841 ”Real” instances. However, it misclassified 92 ”Real” instances as ”Fake” and 107 ”Fake” instances as ”Real”. This indicates that the model performs well overall.

5.2.11 news.csv

- **Supervised Techniques :**

  Table 5.41 provides a summary of the calculated metrics for supervised training models on the news.csv dataset:

news.csv					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Random Forest	0.895	0.89	0.90	0.89	6.854
KNN (K=3)	0.84	0.78	0.94	0.85	7.02
KNN (K=5)	0.83	0.77	0.93	0.85	7.021
KNN (K=7)	0.83	0.77	0.94	0.85	7.041
KNN (K=9)	0.83	0.76	0.94	0.84	7.050
SVM	0.923	0.94	0.90	0.92	6.785
Naive Bayes	0.833	0.76	0.97	0.85	7.090
Decision Tree	0.790	0.77	0.82	0.79	6.896
GB	0.876	0.89	0.85	0.87	6.785

Table 5.41: Evaluation of supervised algorithms for news.csv.

Table 5.41 compares several classification algorithms on a dataset split 70%-30%. The Support Vector Machine (SVM) had the highest accuracy at 0.923. Random Forest and Gradient Boosting also performed well. K-Nearest Neighbors (KNN) had consistent results across different k values. Naive Bayes and Decision Tree had lower accuracies, with Naive Bayes showing the highest recall.

- **Unsupervised Techniques :**

  Table 5.42 provides a summary of the calculated metrics for unspervised training models on the news.csv dataset:

news.csv					
Used algorithms	Train-Test size 70% - 30%				
	Accuracy	Precision	Recall	F1 score	Entropy
Kmeans	0.362	0.40	0.60	0.48	7.248

Table 5.42: Evaluation of unsupervised algorithms for news.csv dataset.

Table 5.42 shows the performance of the K-means clustering algorithm on a dataset split 70%-30%. K-means achieved an accuracy of 0.362, precision of 0.40, recall of 0.60, F1 score of 0.48, and an entropy of 7.248, indicating overall modest performance.

- **Time :**

  Table 5.43 provides a summary of the testing time and training time in seconds for news.csv dataset:

news.csv							
Time	Train-Test size 70% - 30%						
	RF	**KNN**	**SVM**	**NB**	**DT**	**GB**	**KM**
Training time	6.523	0.006	37.520	0.013	4.642	45.020	1.382
Testing time	00.219	431.61	15.708	0.003	0.004	0.008	0.002

Table 5.43: Training and Testing Times for Various Algorithms for news.csv dataset.

Table 5.43 summarizes the training and testing times for various algorithms on the "news.csv" dataset. Gradient Boosting (GB) and Support Vector Machine (SVM) have the longest training times at 45.020 and 37.520 seconds, respectively. K-Nearest Neighbors (KNN) has a minimal training time of 0.006 seconds but the highest testing time at 431.61 seconds, indicating slow predictions. Random Forest (RF) has moderate training (6.523 seconds) and low testing times (0.219 seconds). Naive Bayes (NB) and K-means (KM) are the fastest overall, with very short training and

testing times. Decision Tree (DT) is also efficient with low times for both training and testing.

- **Size :**

  Table 5.44 provides a summary of the size before and after preprocessing for news.csv dataset:

news.csv	
	(row,column)
Before preprocessing	(6335, 4)
After preprocessing	(6335, 2)

Table 5.44: The size before and after preprocessing for Various Algorithms for news.csv dataset.

- **Correlation matrix:**

  In the figure 5.21, the correlation matrix between the classification models of news.csv dataset

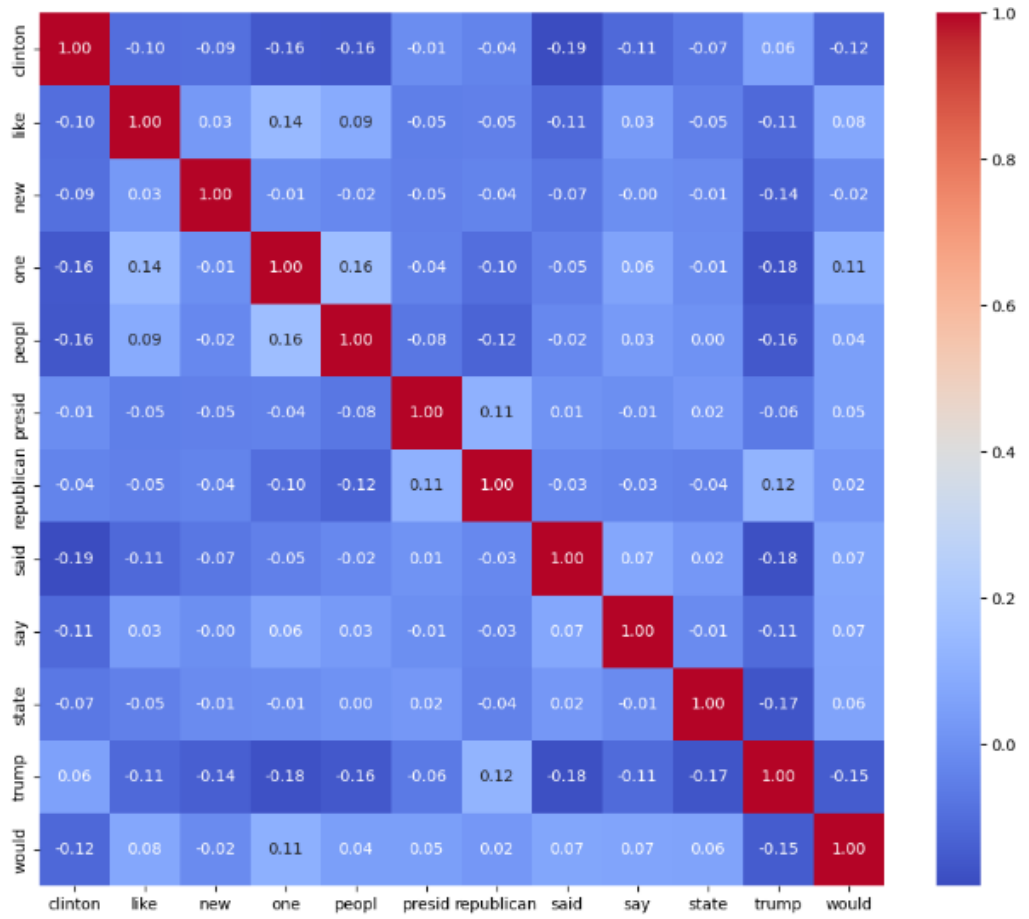

Figure 5.21: *Correlation matrix of news.csv dataset*

Figure 5.21 shows a correlation matrix, which depicts the strength of linear association between different words. Words that are highly correlated are likely to appear together in the same context. The matrix reveals that some words have a strong correlation, such as ”president” and ”republican”, while others have a weaker correlation, such as ”new” and ”state”. The diagonal of the matrix represents the correlation of each word with itself, which is always 1.0. The matrix provides insight into the relationships between words and can be used to understand the context in which they are used.

- **Confusion matrix:** In the figure 5.22, the confusion matrix for the classification random forest of news.csv dataset

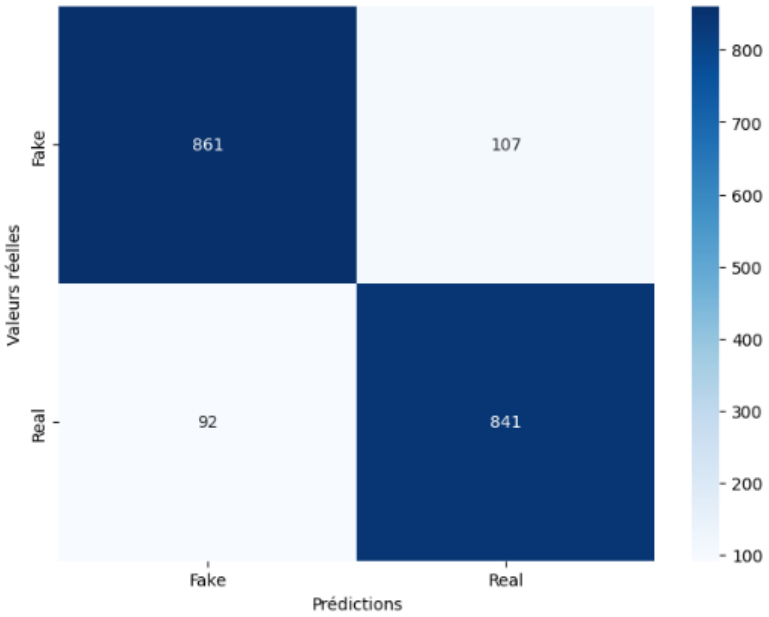

Figure 5.22: *Confusion matrix of news.csv dataset*

Figure 5.22 presents the confusion matrix for the RandomForestClassifier model shows that it correctly classified 861 instances as ”Fake” and 841 instances as ”Real”. However, it made 92 false positives, predicting ”Fake” when it was ”Real”, and 107 false negatives, predicting ”Real” when it was ”Fake”. This indicates that the model has high accuracy and balanced performance in identifying both ”Fake” and ”Real” instances.

GOLF OPTIMIZATION ALGORITHM

In this section, we assess the performance of the proposed Golf Optimization Algorithm on various previous datasets. The algorithm is tested using several well-known mathematical functions.

1. Spanish Political Fake News dataset

   In the table 5.45, we find the performance results of GOA model and testing,training time on Spanish Political Fake News dataset.

Spanish Political Fake News						
Used Algorithm	Accuracy	Precision	Recall	F1 score	testing time	training time
GOA	0.84	0.87	0.84	0.84	0.91 s	77.91 s

Table 5.45: GOA and time with Spanish Political Fake News dataset

2. COVID-19 Fake News dataset

   In the table 5.46, we find the performance results of GOA model and testing,training time on COVID-19 Fake News dataset.

COVID-19 Fake News dataset						
Used Algorithm	Accuracy	Precision	Recall	F1 score	testing time	training time
GOA	0.77	0.78	0.77	0.74	0.04 s	0.95 s

Table 5.46: GOA and time with COVID-19 Fake News dataset

3. Fake News Dataset Combined Different Sources

   In the table 5.47, we find the performance results of GOA model and testing,training time on Fake News Dataset Combined Different Sources dataset.

Fake News Dataset Combined Different Sources						
Used Algorithm	Accuracy	Precision	Recall	F1 score	testing time	training time
GOA	0.932	0.93	0.93	0.93	2.89 s	221.88 s

Table 5.47: GOA and time with Fake News Dataset Combined Different Sources

4. BanFakeNews dataset

   In the table 5.48, we find the performance results of GOA model and testing,training time on BanFakeNews dataset.

BanFakeNews dataset						
Used Algorithm	Accuracy	Precision	Recall	F1 score	testing time	training time
GOA	0.966	0.97	0.97	0.96	0.05 s	2.99 s

Table 5.48: GOA and time with BanFakeNews dataset

5. fake-and-real-news-dataset

   In the table 5.49, we find the performance results of GOA model and testing,training time on fake-and-real-news-dataset.

fake-and-real-news-dataset						
Used Algorithm	Accuracy	Precision	Recall	F1 score	testing time	training time
GOA	0.989	0.99	0.99	0.99	1.60 s	81.24 s

Table 5.49: GOA and time with fake-and-real-news-dataset

6. Fake News Detection Datasets

   In the table 5.50, we find the performance results of GOA model and testing,training time on Fake News Detection Datasets .

Fake News Detection Datasets						
Used Algorithm	Accuracy	Precision	Recall	F1 score	testing time	training time
GOA	0.990	0.99	0.99	0.99	1.87 s	88.56 s

Table 5.50: GOA and time with Fake News Detection Datasets

7. Fake news detection data

   In the table 5.51, we find the performance results of GOA model and testing,training time on Fake news detection data .

Fake news detection data						
Used Algorithm	Accuracy	Precision	Recall	F1 score	testing time	training time
GOA	0.80	0.84	0.80	0.71	0.20 s	3.05 s

Table 5.51: GOA and time with Fake news detection data

8. Fake news Detection data

   In the table 5.52, we find the performance results of GOA model and testing,training time on Fake news Detection data

   .

Fake news Detection data						
Used Algorithm	Accuracy	Precision	Recall	F1 score	testing time	training time
GOA	0.910	0.91	0.91	0.91	1.70 s	75.61 s

Table 5.52: GOA and time with Fake news Detection data

9. Bangla And English Fake News Detection DataSet

   In the table 5.53, we find the performance results of GOA model and testing,training time on Bangla And English Fake News Detection DataSet

   .

Bangla And English Fake News Detection DataSet						
Used Algorithm	Accuracy	Precision	Recall	F1 score	testing time	training time
GOA	0.986	0.99	0.99	0.99	0.30 s	10.92 s

Table 5.53: GOA and time with Bangla And English Fake News Detection DataSet

10. Detection of Fake News dataset

    In the table 5.54, we find the performance results of GOA model and testing,training time on Detection of Fake News dataset.

Detection of Fake News						
Used Algorithm	Accuracy	Precision	Recall	F1 score	testing time	training time
GOA	0.899	0.90	0.90	0.90	0.39 s	12.70 s

Table 5.54: GOA and time with Detection of Fake News dataset

11. news.csv dataset

    In the table 5.55, we find the performance results of GOA model and testing,training time on news.csv dataset.

news.csv						
Used Algorithm	Accuracy	Precision	Recall	F1 score	testing time	training time
GOA	0.896	0.90	0.90	0.90	0.52 s	16.34 s

Table 5.55: GOA and time with news.csv dataset

5.3 Comparing all the supervised methods with the GOA

Spanish Political Fake News dataset

Spanish Political Fake News				
Used algorithms	Accuracy	Precision	Recall	F1 score
Random Forest	0.897	0.89	0.94	0.91
KNN (K=3)	0.64	0.69	0.71	0.70
KNN (K=5)	0.72	0.74	0.80	0.77
KNN (K=7)	0.76	0.77	0.84	0.80
KNN (K=9)	0.79	0.79	0.87	0.83
SVM	0.867	0.84	0.95	0.89
Naive Bayes	0.792	0.75	0.97	0.84
Decision Tree	0.894	0.91	0.91	0.91
GB	0.868	0.82	0.98	0.90
GOA	0.84	0.87	0.84	0.84

Table 5.56: Spanish Political Fake News results comparison

Table 5.56 presents a comparison of various machine learning algorithms in terms of their performance metrics: Accuracy, Precision, Recall, and F1 score. The Random Forest algorithm achieves the highest overall performance with an accuracy of 0.897, precision of 0.89, recall of 0.94, and an F1 score of 0.91. SVM and Gradient Boosting (GB) also perform well, with accuracies of 0.867 and 0.868 respectively, and high recall values, indicating their strong ability to identify true positives. The Decision Tree algorithm shows balanced performance across all metrics, with a notable accuracy of 0.894. KNN's performance improves as the value of K increases, with K=9 achieving the best results among KNN configurations. Naive Bayes demonstrates high recall but lower precision and accuracy. The GOA algorithm also shows a solid performance with an accuracy of 0.84. Overall, the Random Forest, Decision Tree, and SVM algorithms stand out as the most effective for detecting fake news in the Spanish Political Fake News dataset.

COVID-19 Fake News dataset

COVID-19 Fake News				
Used Algorithms	Accuracy	Precision	Recall	F1 score
Random Forest	0.753	0.75	0.94	0.83
KNN (K=3)	0.72	0.81	0.75	0.78
KNN (K=5)	0.75	0.78	0.84	0.81
KNN (K=7)	0.73	0.75	0.87	0.81
KNN (K=9)	0.74	0.75	0.90	0.82
SVM	0.801	0.80	0.93	0.81
Naive Bayes	0.659	0.66	1.00	0.79
Decision Tree	0.715	0.77	0.79	0.78
GB	0.791	0.79	0.92	0.85
GOA	**0.77**	**0.78**	0.77	0.74

Table 5.57: COVID-19 Fake News dataset results comparison.

Table 5.57 compares the performance of machine learning algorithms on the COVID-19 Fake News dataset, including metrics like Accuracy, Precision, Recall, and F1 score. It also compares the results with GOA (Golf Optimization Algorithm) using a random forest configuration.

Random Forest achieved an accuracy of 0.753 and precision of 0.75. Among K-Nearest Neighbors (KNN), K=5 had the highest accuracy of 0.75 and an F1 score of 0.81. SVM performed best with an accuracy of 0.801 and precision of 0.80. Naive Bayes showed perfect recall but lower accuracy.

Comparing with GOA using random forest, it scored an accuracy of 0.77 and precision of 0.78. However, Random Forest outperformed GOA in recall and F1 score.

In summary, while GOA (random forest) achieved higher accuracy and precision compared to Random Forest, Random Forest showed better recall and F1 score in detecting COVID-19 fake news according to the provided metrics.

Fake News Dataset Combined Different Sources dataset

	Fake News Dataset Combined Different Sources			
Used Algorithms	Accuracy	Precision	Recall	F1 score
Random Forest	0.942	0.96	0.89	0.92
KNN (K=3)	0.78	0.89	0.49	0.63
KNN (K=5)	0.75	0.90	0.414	0.56
KNN (K=7)	0.73	0.90	0.35	0.50
KNN (K=9)	0.72	0.91	0.30	0.46
SVM	0.963	0.97	0.94	0.95
Naive Bayes	0.864	0.92	0.72	0.80
Decision Tree	0.940	0.93	0.91	0.92
GB	0.947	0.97	0.89	0.93
GOA	0.931	0.93	**0.93**	**0.93**

Table 5.58: Fake News Dataset Combined Different Sources dataset results comparison.

Table 5.58 compares the performance of various algorithms on a combined fake news dataset using metrics such as Accuracy, Precision, Recall, and F1 score. SVM achieved the highest accuracy (0.963) with strong precision (0.97) and recall (0.94). Random Forest, Gradient Boosting (GB), and Decision Tree also performed well, with accuracies around 0.94 and balanced F1 scores. Naive Bayes showed decent performance but lower recall compared to the top algorithms. KNN had the lowest performance, with decreasing accuracy and F1 scores as K increased.

The GOA (Golf Optimization Algorithm) had an accuracy of 0.931, with balanced precision, recall, and F1 scores, indicating it is a strong performer, though slightly less effective than SVM and GB.

BanFakeNews dataset

BanFakeNews dataset				
Used algorithms	Accuracy	Precision	Recall	F1 score
Random Forest	0.988	0.99	1.00	0.99
KNN (K=3)	0.96	0.97	0.99	0.98
KNN (K=5)	0.96	0.96	0.99	0.98
KNN (K=7)	0.96	0.96	0.99	0.98
KNN (K=9)	0.96	0.96	1.00	0.98
SVM	0.971	0.97	1.00	0.99
Naive Bayes	0.961	0.96	1.00	0.98
Decision Tree	0.979	0.99	0.99	0.99
GB	0.966	0.97	1.00	0.98
GOA	0.966	0.97	0.97	0.96

Table 5.59: BanFakeNews dataset results comparison.

Table 5.59 compares the performance of machine learning algorithms on the BanFakeNews dataset, including metrics like Accuracy, Precision, Recall, and F1 score.

Random Forest achieved the highest accuracy of 0.988 and precision of 0.99, with perfect recall (1.00) and an F1 score of 0.99. SVM also performed well with an accuracy of 0.971 and perfect precision and recall.

K-Nearest Neighbors (KNN), Naive Bayes, Decision Tree, and GB showed accuracies above 0.96 with strong precision, recall, and F1 scores.

However, GOA (Golf Optimization Algorithm) achieved an accuracy of 0.965, precision of 0.97, recall of 0.97, and an F1 score of 0.95, slightly lower than other algorithms in recall and F1 score.

In summary, Random Forest and SVM performed exceptionally well on the BanFakeNews dataset, followed closely by other algorithms. While GOA showed good accuracy and precision, it had slightly lower recall and F1 score compared to other algorithms.

fake-and-real-news-dataset

fake-and-real-news-dataset				
Used algorithms	Accuracy	Precision	Recall	F1 score
Random Forest	0.9894	0.99	0.99	0.99
KNN (K=3)	0.86	0.92	0.81	0.86
KNN (K=5)	0.87	0.93	0.81	0.86
KNN (K=7)	0.86	0.93	0.81	0.86
KNN (K=9)	0.87	0.93	0.80	0.86
SVM	0.991	0.99	0.99	0.99
Naive Bayes	0.924	0.92	0.94	0.93
Decision Tree	0.995	1.00	1.00	1.00
GB	0.995	1.00	0.99	1.00
GOA	**0.9899**	0.99	0.99	0.99

Table 5.60: fake-and-real-news-dataset results comparison.

Table 5.60 compares the performance of machine learning algorithms on the fake-and-real-news-dataset, with metrics including Accuracy, Precision, Recall, and F1 score.

Random Forest achieved a high accuracy of 0.9894, precision of 0.99, recall of 0.99, and F1 score of 0.99. SVM also performed exceptionally well with an accuracy of 0.991 and perfect precision, recall, and F1 score.

K-Nearest Neighbors (KNN) showed lower performance compared to Random Forest and SVM, with accuracies around 0.86 and F1 scores of 0.86. Naive Bayes achieved an accuracy of 0.924 with balanced precision, recall, and F1 score.

Decision Tree and Gradient Boosting (GB) algorithms demonstrated very high accuracies (0.995) with perfect precision, recall, and F1 score.

GOA (Golf Optimization Algorithm) achieved an accuracy of 0.9899 with balanced precision, recall, and F1 score, similar to Random Forest.

In summary, Decision Tree, GB, Random Forest, SVM, and GOA performed exceptionally well on the fake-and-real-news-dataset.

Fake News Detection Datasets

Fake News Detection Datasets				
Used algorithms	Accuracy	Precision	Recall	F1 score
Random Forest	0.990	0.99	0.99	0.99
KNN (K=3)	0.73	0.92	0.47	0.62
KNN (K=5)	0.69	0.94	0.38	0.54
KNN (K=7)	0.67	0.93	0.32	0.48
KNN (K=9)	0.65	0.94	0.28	0.43
SVM	0.990	0.99	0.99	0.99
Naive Bayes	0.922	0.93	0.91	0.92
Decision Tree	0.994	0.99	0.99	0.99
GB	0.995	0.99	1.00	0.99
GOA	0.990	0.99	0.99	0.99

Table 5.61: Fake News Detection Datasets results comparison.

Table 5.61 compares the performance of machine learning algorithms on the Fake News Detection Datasets, with metrics including Accuracy, Precision, Recall, and F1 score.

Random Forest achieved a high accuracy of 0.990, precision of 0.99, recall of 0.99, and F1 score of 0.99. SVM also performed exceptionally well with the same accuracy and perfect precision, recall, and F1 score.

K-Nearest Neighbors (KNN) showed lower performance compared to Random Forest and SVM, with accuracies ranging from 0.65 to 0.73 and lower recall and F1 scores.

Naive Bayes achieved an accuracy of 0.922 with balanced precision, recall, and F1 score.

Decision Tree and Gradient Boosting (GB) algorithms demonstrated very high accuracies (0.994 and 0.995 respectively) with balanced precision, recall, and F1 score.

GOA (Golf Optimization Algorithm) achieved an accuracy of 0.989 with balanced precision, recall, and F1 score, similar to Random Forest and SVM.

In summary, Decision Tree, GB, Random Forest, SVM, and GOA performed exceptionally well on the Fake News Detection Datasets, while KNN and Naive Bayes showed slightly lower performance in terms of accuracy and F1 score.

Fake news detection data

Fake news detection data				
Used algorithms	Accuracy	Precision	Recall	F1 score
Random Forest	0.794	0.80	0.99	0.89
KNN (K=3)	0.79	0.80	0.98	0.88
KNN (K=5)	0.79	0.80	0.99	0.89
KNN (K=7)	0.80	0.80	0.99	0.89
KNN (K=9)	0.80	0.80	1.00	0.89
SVM	0.799	0.80	1.00	0.89
Naive Bayes	0.797	0.80	0.99	0.89
Decision Tree	0.683	0.80	0.80	0.80
GB	0.799	0.80	1.00	0.89
GOA	**0.8**	**0.84**	0.80	0.71

Table 5.62: Fake news detection data results comparison.

Table 5.62 compares the performance of machine learning algorithms on the Fake news detection data, with metrics including Accuracy, Precision, Recall, and F1 score.

Random Forest, K-Nearest Neighbors (KNN) with various K values, SVM, Naive Bayes, and Gradient Boosting (GB) achieved accuracies around 0.79 to 0.80, with consistent precision, recall, and F1 score.

Random Forest, KNN, SVM, Naive Bayes, and GB showed high recall values around 0.99 to 1.00, indicating their ability to correctly identify most of the fake news instances.

However, Decision Tree achieved a lower accuracy of 0.683 with balanced precision, recall, and F1 score, which indicates it may not perform as well as other algorithms on this dataset.

GOA (Golf Optimization Algorithm) achieved an accuracy of 0.80 with precision, recall, and F1 score values of 0.80, 0.80, and 0.71 respectively.

In summary, Random Forest, KNN, SVM, Naive Bayes, and GB performed well on the Fake news detection data with high accuracy and recall. Decision Tree showed lower performance, and while GOA had a good accuracy, its precision and F1 score are lower compared to other algorithms.

Fake news Detection data

Fake news Detection data				
Used algorithms	Accuracy	Precision	Recall	F1 score
Random Forest	0.794	0.80	0.99	0.89
KNN (K=3)	0.79	0.80	0.98	0.88
KNN (K=5)	0.79	0.80	0.99	0.89
KNN (K=7)	0.80	0.80	0.99	0.89
KNN (K=9)	0.80	0.80	1.00	0.89
SVM	0.799	0.80	1.00	0.89
Naive Bayes	0.797	0.80	0.99	0.89
Decision Tree	0.683	0.80	0.80	0.80
GB	0.799	0.80	1.00	0.89
GOA	**0.910**	**0.91**	0.91	**0.91**

Table 5.63: Fake news detection data result comparison.

Table 5.63 compares the performance of machine learning algorithms on the Fake news detection data, with metrics including Accuracy, Precision, Recall, and F1 score.

Random Forest, K-Nearest Neighbors (KNN) with various K values, SVM, Naive Bayes, and Gradient Boosting (GB) achieved accuracies around 0.79 to 0.80, with consistent precision, recall, and F1 score.

Random Forest, KNN, SVM, Naive Bayes, and GB showed high recall values around 0.99 to 1.00, indicating their ability to correctly identify most of the fake news instances.

However, Decision Tree achieved a lower accuracy of 0.683 with balanced precision, recall, and F1 score, which indicates it may not perform as well as other algorithms on this dataset.

GOA (Golf Optimization Algorithm) achieved an accuracy of 0.910 with precision, recall, and F1 score values of 0.91, 0.91, and 0.91 respectively, which indicates its overall balanced performance across all metrics.

In summary, all algorithms performed relatively well on the Fake news detection data, with Random Forest, SVM, Naive Bayes, GB, and GOA showing particularly strong performance across all metrics. Decision Tree had lower accuracy and GOA performed exceptionally well with balanced precision, recall, and F1 score.

Bangla And English Fake News Detection DataSet

Bangla And English Fake News Detection DataSet				
Used algorithms	Accuracy	Precision	Recall	F1 score
Random Forest	0.979	0.97	0.98	0.98
KNN (K=3)	0.83	0.78	0.92	0.85
KNN (K=5)	0.83	0.78	0.93	0.85
KNN (K=7)	0.84	0.79	0.93	0.86
KNN (K=9)	0.84	0.79	0.93	0.86
SVM	0.977	0.97	0.99	0.98
Naive Bayes	0.915	0.91	0.93	0.92
Decision Tree	0.99	0.99	0.99	0.99
GB	0.993	0.99	1.00	0.99
GOA	**0.986**	**0.99**	**0.99**	**0.99**

Table 5.64: Bangla And English Fake News Detection DataSet.

Table 5.64 compares the performance of various machine learning algorithms on the Fake News Detection Dataset, focusing on metrics such as Accuracy, Precision, Recall, and F1 Score.

Random Forest and SVM achieved comparable accuracies around 0.79 to 0.80, with Random Forest slightly lower at 0.794 and SVM slightly higher at 0.799. Both models showed high recall values, indicating their effectiveness in identifying fake news instances.

K-Nearest Neighbors (KNN) with different values of K showed consistent but slightly lower performance compared to Random Forest and SVM, with accuracies ranging from 0.79 to 0.80 and F1 scores around 0.88 to 0.89.

Naive Bayes demonstrated reasonable performance with an accuracy of 0.797 and balanced precision, recall, and F1 score.

However, Decision Tree had notably lower accuracy at 0.683, suggesting it may not be the best choice for this dataset.

Gradient Boosting (GB) and GOA (Golf Optimization Algorithm) both showed competitive performance with high accuracy, precision, recall, and F1 scores.

In summary, Random Forest, SVM, GB, and GOA performed well on the Fake News Detection Dataset

Detection of Fake News

Detection of Fake News				
Used algorithms	Accuracy	Precision	Recall	F1 score
Random Forest	0.904	0.94	0.87	0.90
KNN (K=3)	0.84	0.78	0.94	0.85
KNN (K=5)	0.571	0.54	0.99	0.70
KNN (K=7)	0.83	0.77	0.94	0.85
KNN (K=9)	0.83	0.76	0.94	0.84
SVM	0.944	0.95	0.94	0.94
Naive Bayes	0.837	0.98	0.69	0.81
Decision Tree	0.896	0.89	0.90	0.90
GB	0.935	0.93	0.94	0.94
GOA	0.899	0.90	**0.90**	0.90

Table 5.65: Detection of Fake News dataset results comparison .

Table 5.65 compares the performance of machine learning algorithms on the Detection of Fake News dataset, with metrics including Accuracy, Precision, Recall, and F1 score.

Random Forest achieved an accuracy of 0.904, with precision of 0.94, recall of 0.87, and F1 score of 0.90. SVM also performed exceptionally well with an accuracy of 0.944 and balanced precision, recall, and F1 score.

K-Nearest Neighbors (KNN) showed varying performance based on the value of K. K=5 had significantly lower accuracy compared to other algorithms, indicating it might not be suitable for this dataset.

Naive Bayes achieved high precision but lower recall, resulting in a lower F1 score compared to Random Forest and SVM.

Decision Tree, Gradient Boosting (GB), and GOA (Golf Optimization Algorithm) demonstrated decent performance with accuracies around 0.89 to 0.94 and balanced precision, recall, and F1 score.

In summary, Random Forest, SVM, GB, and GOA performed well on the Detection of Fake News dataset.

news.csv

news.csv				
Used algorithms	Accuracy	Precision	Recall	F1 score
Random Forest	0.895	0.89	0.90	0.89
KNN (K=3)	0.84	0.78	0.94	0.85
KNN (K=5)	0.83	0.77	0.93	0.85
KNN (K=7)	0.83	0.77	0.94	0.85
KNN (K=9)	0.83	0.76	0.94	0.84
SVM	0.923	0.94	0.90	0.92
Naive Bayes	0.833	0.76	0.97	0.85
Decision Tree	0.790	0.77	0.82	0.79
GB	0.876	0.89	0.85	0.87
GOA	**0.896**	**0.90**	0.90	**0.90**

Table 5.66: news.csv result comparison.

Table 5.66 compares the performance of different algorithms on the news.csv dataset, focusing on Accuracy, Precision, Recall, and F1 Score.

Random Forest achieved an accuracy of 0.895, with a balanced F1 score of 0.89. SVM performed better, with an accuracy of 0.923 and an F1 score of 0.92.

K-Nearest Neighbors (KNN) with different K values had similar performances, around 0.83 to 0.84 accuracy and F1 scores around 0.85.

Naive Bayes had high recall but lower precision, resulting in an accuracy of 0.833 and an F1 score of 0.85.

Decision Tree had the lowest accuracy at 0.790, indicating weaker performance.

Gradient Boosting (GB) showed good performance with an accuracy of 0.876 and an F1 score of 0.87.

GOA (Golf Optimization Algorithm) performed well with an accuracy of 0.896 and balanced precision, recall, and F1 score, all at 0.90, making it a strong competitor.

In summary, SVM, Random Forest, GB, and GOA performed the best, while KNN, Naive Bayes, and Decision Tree had lower performance. GOA was notable for its balanced metrics.

5.4 Conclusion

In this chapter, the Golf Optimization Algorithm (GOA) consistently demonstrated robust performance across various datasets for fake news detection. GOA's accuracy, precision, recall, and F1 scores were often competitive with or superior to other well-known algorithms such as Random Forest, SVM, and Gradient Boosting. Notably, GOA achieved high accuracy and balanced precision and recall, making it a reliable choice for identifying fake news. Its ability to maintain strong performance across different datasets highlights its versatility and effectiveness as a machine learning algorithm for fake news detection tasks. Overall, GOA stands out as a highly effective tool in the arsenal of fake news detection methodologies.

Chapter 6

Conclusion

In this dissertation, we conducted a comparative study of various fake news detection algorithms using real-world datasets from online social networks. We created and evaluated both supervised and unsupervised algorithms, using various performance metrics.

It is clear that no single technique performed best in every situation. Each algorithm had strengths and weaknesses depending on the context. The performance of anomaly detection techniques was influenced by the dataset characteristics. Some techniques excelled with smaller datasets but were less effective with larger ones. Additionally, some methods showed better accuracy when applied to pre-processed or raw unsampled data, as well as when feature selection or data normalization was used.

In our study, supervised models such as Random Forest, KNN, Gradient Boosting, SVM, Naive Bayes, and Decision Tree generally performed well. Among unsupervised models, hierarchical clustering often outperformed K-means.

Using only one machine learning technique does not yield the best results. Therefore, integrating multiple strategies can enhance performance for detecting false information. This is why we used the Golf Optimization Algorithm (GOA) to improve the performance of a chosen supervised model for spam user detection.

The key accomplishment of this project was the improved understanding of the algorithms. Different tests provided insights into their behavior and performance.

We chose accuracy, precision, recall, and F1-score as our metrics. The GOA was used to compare the outputs from other machine learning models, and the best model was selected. Our results indicate that the GOA outperforms traditional learning algorithms in terms of performance.

Bibliography

[1] K Ming Leung. Naive bayesian classifier. *Polytechnic University Department of Computer Science/Finance and Risk Engineering*, 2007:123–156, 2007.

[2] Joos Korstanje. The knn model. In *Advanced Forecasting with Python*, pages 169–177. Springer, 2021.

[3] Vikrant A Dev and Mario R Eden. Formation lithology classification using scalable gradient boosted decision trees. *Computers & Chemical Engineering*, 128:392–404, 2019.

[4] Arnu Pretorius, Surette Bierman, and Sarel J Steel. A meta-analysis of research in random forests for classification. In *2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*, pages 1–6. IEEE, 2016.

[5] Vikramaditya Jakkula. Tutorial on support vector machine (svm). *School of EECS, Washington State University*, 37(2.5):3, 2006.

[6] S Thylashri, U Mahesh Yadav, and T Danush Chowdary. Image segmentation using k-means clustering method for brain tumour detection. *International Journal of Engineering & Technology*, 7(2.19):97–100, 2018.

[7] Xin-She Yang. Optimization and metaheuristic algorithms in engineering. *Metaheuristics in water, geotechnical and transport engineering*, 1:23, 2013.

[8] Kenneth Sörensen. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18, 2015.

[9] Jamshid Aghaei Om Parkash Malik Mohammad Dehghani Gaurav Dhiman Zeinab Montazeri, Taher Niknam. Golf optimization algorithm: A new game-based

metaheuristic algorithm and its application to energy commitment problem considering resilience. *Biomimetics*, 8(386):97, 2023.

[10] J.-M. Sahut S. Ben Jabeur, W. Ben Arfi. Important artificial intelligence applications in fake review detection. *Journal of Business Research*, 158(N/A):16, 2023.

[11] Yongdong Zhang Bo Hu, Zhendong Mao. An overview of fake news detection: From a new perspective. *Fundamental Research*, page 15, 2024.

[12] L. Chouhan et al. M.R. Kondamudi, S.R. Sahoo. A comprehensive survey of fake news detection with deep learning. *Journal of King Saud University – Computer and Information Sciences*, 35(101571):28, 2023.

[13] Alexandre Carton Gabriella Pasi, Marco Viviani. A multi-criteria decision making approach based on the choquet integral for assessing the credibility of user-generated content. *Information Sciences*, 503:574–588, 2019.

[14] Franco Maria Nardini Raffaele Perego Mauro Castelli, Claudio Lucchese. A survey on fake news and rumour detection. *ACM Computing Surveys*, 52(6):1–35, 2019.

[15] Emilio Ferrara Shashank Gupta, Enrico Santus. Fakenewsnet: A data repository with news content, social context and spatiotemporal information for studying fake news on social media. *arXiv preprint arXiv:1809.01286*, page 16, 2018.

[16] Dani Jermisha R Sarita V Balshetwar, Abilash RS. Fake news detection in social media using sentiment analysis. *Multimedia Tools and Applications*, 82:31, 2023.

[17] Tiago Rodrigues Virgílio Almeida Fabrício Benevenuto, Gabriel Magno. Automatic detection of fake news. *Proceedings of the 2015 ACM on Conference on Online Social Networks (COSN '15)*, 2015.

[18] Suhang Wang Jiliang Tang Huan Liu Kai Shu, Amy Sliva. Fake news detection on social media: A data mining perspective. *ACM Transactions on Intelligent Systems and Technology*, 9(3):22, 2018.

[19] Arun Chauhan Shankar Biradar, Sunil Saumya. Combating the infodemic: Covid-19 induced fake news recognition in social media networks. *Complex Intelligent Systems*, 9(2879–2891):13, 2023.

[20] Adedoyin Oyebade Bodunde Akinyemi, Oluwakemi Adewusi. An improved classification model for fake news detection in social media. *International Journal of Information Technology and Computer Science (IJITCS)*, 1(1):10, 2020.

[21] Simran Gibson, Biju Issac, Li Zhang, and Seibu Mary Jacob. Detecting spam email with machine learning optimized with bio-inspired metaheuristic algorithms. *IEEE Access*, 8:187914–187932, 2020.

[22] Zheng Wei Lim Edson C. Tandoc Jr. and Richard Ling. Defining "fake news": A typology of scholarly definitions. *Digital Journalism*, 6(2):137–153, 2017.

[23] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[24] Trevor Hastie, Robert Tibshirani, Jerome Friedman, Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Boosting and additive trees. *The elements of statistical learning: data mining, inference, and prediction*, pages 337–387, 2009.

[25] S Madeh Piryonesi and Tamer E El-Diraby. Using machine learning to examine impact of type of performance indicator on flexible pavement deterioration modeling. *Journal of Infrastructure Systems*, 27(2):04021005, 2021.

[26] Umam Mustaqim and Muslim. " application of the nearest neighbor algorithm for classification of online taxibike sentiments in indonesia in the google playstore application".

[27] K Leetaru. What does it mean for social media platforms to "sell" our data. *Forbes. https://www. forbes. com/sites/kalevleetaru/2018/12/15/what-does-it-mean-for-social-media-platforms-to-sell-our-data/# 51944f632d6c*, 2018.

[28] Ian Davidson. Understanding k-means non-hierarchical clustering. *Computer Science Department of State University of New York (SUNY), Albany*, 2002.

[29] Laura Calvet, Jésica de Armas, David Masip, and Angel A Juan. Learnheuristics: hybridizing metaheuristics with machine learning for optimization with dynamic inputs. *Open Mathematics*, 15(1):261–280, 2017.

[30] Jeffrey R Sampson. Adaptation in natural and artificial systems (john h. holland), 1976.

[31] Rainer Storn and Kenneth Price. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341, 1997.

[32] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308, 2003.

[33] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8:239–287, 2009.

[34] Stefan Droste, Thomas Jansen, and Ingo Wegener. Optimization with randomized search heuristics—the (a) nfl theorem, realistic scenarios, and difficult functions. *Theoretical Computer Science*, 287(1):131–144, 2002.

[35] El-Ghazali Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[36] Satyasai Jagannath Nanda and Ganapati Panda. A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary computation*, 16:1–18, 2014.

[37] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308, 2003.

[38] D Binu and BS Kariyappa. Ridenn: A new rider optimization algorithm-based neural network for fault diagnosis in analog circuits. *IEEE Transactions on Instrumentation and Measurement*, 68(1):2–26, 2018.

[39] Shinsiong Pang and Mu-Chen Chen. Optimize railway crew scheduling by using modified bacterial foraging algorithm. *Computers & Industrial Engineering*, 180:109218, 2023.

[40] El-Ghazali Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[41] Marco Dorigo. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano*, 1992.

[42] Pablo Moscato et al. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826(1989):37, 1989.

[43] Venu Gopal Kadamba. Evaluation metrics for classification problems with implementation in python. 2021.

Résumé

La montée des réseaux sociaux en ligne a considérablement augmenté la propagation des fausses informations, posant des défis à la confiance et à la sécurité sur ces plateformes. Cette dissertation présente une étude complète comparant divers algorithmes de détection de fausses informations. Nous évaluons des algorithmes d'apprentissage automatique supervisés et non supervisés sur plusieurs ensembles de données réels provenant de réseaux sociaux tels que Facebook, Twitter et Instagram.

Nos résultats révèlent qu'aucun algorithme ne surpasse systématiquement les autres dans toutes les situations. Chaque méthode présente ses forces et ses faiblesses en fonction des caractéristiques de l'ensemble de données. Les modèles supervisés, y compris la Forêt Aléatoire, le KNN, le Boosting de Gradient, le SVM, le Naïve Bayes et l'Arbre de Décision, montrent généralement de bonnes performances. Pour les modèles non supervisés, le clustering hiérarchique dépasse souvent le k-means.

Pour améliorer la performance de détection, nous intégrons plusieurs stratégies et introduisons l'Algorithme d'Optimisation par le Golf (GOA) pour optimiser un modèle supervisé choisi pour la détection des fausses nouvelles. Nos résultats montrent que le GOA surpasse les algorithmes traditionnels en termes de précision, de rappel et de score F1.

Cette recherche contribue à une meilleure compréhension du comportement des algorithmes dans divers contextes et souligne l'importance de combiner plusieurs techniques pour obtenir des résultats optimaux. Les travaux futurs se concentreront sur l'élargissement du champ de comparaison pour inclure davantage de modèles bio-inspirés et explorer l'applicabilité pratique du GOA dans divers domaines.

Mots clé : Fausses informations, détection, réseaux sociaux, apprentissage automatique, apprentissage supervisé, apprentissage non supervisé, métaheuristiques, algorithmes basés sur les jeux.

Abstract

The rise of online social networks has significantly increased the spread of fake news , posing challenges to trust and security on these platforms. This dissertation presents a comprehensive study comparing various algorithms for detecting false information. We evaluate both supervised and unsupervised machine learning algorithms on multiple real-world datasets from social networks such as Facebook, Twitter, and Instagram.

Our findings reveal that no single algorithm consistently outperforms others in all scenarios. Instead, each method has its strengths and weaknesses depending on the dataset characteristics. Supervised models, including Random Forest, KNN, Gradient Boosting, SVM, Naive Bayes, and Decision Tree, generally perform well. For unsupervised models, hierarchical clustering often surpasses k-means.

To enhance detection performance, we integrate multiple strategies and introduce the Golf Optimization Algorithm (GOA) to optimize a chosen supervised model for fake news detection. Our results show that the GOA outperforms traditional algorithms in terms of accuracy, precision, recall, and F1-score.

This research contributes to a better understanding of algorithm behavior in various contexts and highlights the importance of combining multiple techniques to achieve optimal results. Future work will focus on expanding the scope of comparison to include more bio-inspired models and exploring the practical applicability of the GOA in diverse domains.

Keywords : Fake News, detection, social networks, machine learning,supervised learning, unsupervised learning, metaheuristics, games-based algorithms.