



Fake News Detection Using Machine Learning

Prepared By

Name: [Metab Hussin Jdeba]

IDNo.: [252001230]

Submitted in fulfillment of the requirements for the degree
Master of Science in Artificial Intelligence

At

College of Informatics

Midocean University

UAE

Supervisor Name: Dr. Ashraf Elsayed

College of Informatics, Midocean University, Comoros

[2026-semester-3]

Acknowledgements

I would like to express my sincere gratitude to Dr. Ashraf Elsayed for his invaluable guidance, continuous support, and encouragement throughout this research project. His expertise, insightful feedback, and unwavering dedication have significantly contributed to the quality and direction of this thesis.

I also extend my heartfelt appreciation to my family, friends, and colleagues, whose patience, motivation, and moral support were instrumental in the successful completion of this work.

A special thank you goes to my family for his generous help and continuous encouragement throughout this journey.

Abstract

The accelerated development of the online media and social networking platforms has transformed the information sharing process and, at the same time, established the environment which can be prone to the distribution of misinformation and fake news. Such a phenomenon is very dangerous to the opinion of people, the democratic process, and stability in society, and one of the urgent challenges of the world today is to detect fake news. The manual verification and fact-checking can never identify the vast amount and pace of the information spreading in the digital age.

The study will present a machine learning model of automatically identifying fake news through textual analysis. The model is based on preprocessing and converting raw text into structured format using natural language processing to ensure it is used in computational learning. Linguistic patterns will be represented in numbers with the help of such feature extraction methods as Bag-of-Words and Term Frequency-Inverse Document Frequency (TF-IDF). A range of supervised machine learning algorithms such as Logistic Regression, Support Vector Machine (SVM), Random Forest, and Naive Bayes will subsequently be designed and tested to establish their capability of classifying the content of news as real or fake.

Standard evaluation measures, including the accuracy, precision, recall, F1-score, and ROC-AUC are going to be used to evaluate the performance of the proposed models. It is predicted that a strong, interpretable scaled machine learning system can be developed to detect the fake news articles with high accuracy. By offering information on the weaknesses and strengths of classical methods of machine learning, this study will be aimed at contributing to the literature on automated misinformation detection.

Table of Contents

Acknowledgements	ii
Abstract.....	iii
List of Abbreviations	vii
Table of Figures	viii
Table of Tables	ix
Chapter 1 Introduction	1
1.1. Problem Definition	1
1.2. Thesis Objectives	3
1.3. Thesis Contribution	5
1.4. Thesis Organization	5
Chapter 2 Background and Related Work	7
2.1. Background.....	7
2.2. Conceptual Introduction and Fake News definition.....	9
2.3. Representation of Features and Data Modalities.....	10
2.4. Conventional machine learning practices.....	11
2.5. Deep Learning Approaches.....	12
2.6. Multi-View, Graph, and Multi-Modal Models.....	13
2.7. Domain and Language-Specific Studies.....	14
2.8. Surveys, Comparative Studies and Problems.	14
2.9. Summary and Research Gaps.....	16
2.10. Chapter Summary.....	19
Chapter 3: Proposed Approach.....	21
3.1. Introduction	21
3.2. Dataset Description	21
3.3. Data Source.....	23
3.4. System Overview	23

3.5.	Experimental Environment.....	26
3.6.	Data Preprocessing.....	26
3.7.	Model Development.....	29
3.8.	Evaluation Metrics	30
3.9.	Data Preparation and Preprocessing.	33
3.9.1.	Bias in Data and Characteristics of the source.	34
3.9.2.	Feature Representation Techniques.....	34
3.9.3.	Word-Level TF-IDF Features.....	36
3.9.4.	Word2Vec Document Embeddings	37
3.9.5.	Deep Learning Model sequence encoding.	38
3.9.6.	Character Level TF-IDF Features.	39
3.10.	Classification Models	40
3.10.1.	Traditional Machine Learning Models.....	42
3.10.2.	Deep Learning Models.....	43
3.10.3.	Transformer-Based Model.....	44
3.11.	Ensemble and Hybrid Architectures.....	45
3.11.1.	Word + Character TF-IDF with SVM.....	46
3.11.2.	Stacking Ensemble	47
3.11.3.	Deep + ML Voting Ensemble	48
3.12.	Model Evaluation Setup	50
3.13.	Summary	51
Chapter 4:	Experimental Results and Discussion	53
4.1.	Introduction	53
4.2.	Experimental Setup.....	54
4.3.	Overall Model Performance.....	57
4.4.	Traditional machine learning models Performance.	59
4.5.	Deep Learning Model Attainments.	61
4.6.	Transformer-Based Result.....	62
4.7.	Hybrid and Ensemble Models Performance.	63

4.8.	Discussion of Key Findings.....	64
4.8.1.	The Interpretation of Near-Perfect Results.....	65
4.8.2.	Limitations of the Experimental Results.	65
4.8.3.	Dataset Bias and Generalization Limitations.....	66
4.8.4.	Potential Extrapolation and Interpretation of Very High Accuracy.	67
4.8.5.	Limited Explainability Analysis	67
4.9.	Summary of the Study	67
4.10.	Chapter Summary	68
Chapter 5:	Conclusion and Future Work	70
5.1.	Conclusion.....	70
5.2.	Future Work	72
References		76
Appendix I :	Code	78

List of Abbreviations

Abbreviation Full Term

AI	Artificial Intelligence
AE	Autoencoder
BiLSTM	Bidirectional Long Short-Term Memory
BoW	Bag-of-Words
CNN	Convolutional Neural Network
CSV	Comma-Separated Values
DL	Deep Learning
F1-Score	Harmonic Mean of Precision and Recall
GCN	Graph Convolutional Network
GBCA	Graph-Based Co-Attention
ISOT	University of Victoria Information Security and Object Technology (Fake News Dataset)
KNN	K-Nearest Neighbors
LM	Language Model
LSTM	Long Short-Term Memory
ML	Machine Learning
MV-AE	Multi-View Autoencoder
NLP	Natural Language Processing
PUI-Net	Post-User Interaction Network
RNN	Recurrent Neural Network
ROC-AUC	Receiver Operating Characteristic – Area Under the Curve
SVM	Support Vector Machine
TF-IDF	Term Frequency–Inverse Document Frequency
XAI	Explainable Artificial Intelligence

Table of Figures

Figure 1.1 Experimental Workflow	4
Figure 1.2 visual representation of how the chapters, research files, and model iterations interact throughout the study	6
Figure 3.1 Proposed framework of the fake news detection system	25
Figure 3.2 Preprocessing stages	28
Figure 3.3 Model Architecture.....	29
Figure 3.4 Work Flow	30
Figure 4.1 Confusion matrix of TF-IDF + SVM	54
Figure 4.2 Confusion matrix of CNN + Trainable Embedding	55
Figure 4.3 Confusion matrix of Word + Char TF-IDF + SVM.....	55
Figure 4.4 Model Comparison - Fake News Detection	59

Table of Table

Table 2.1 Summary of Key Studies in Fake News Detection Using Machine Learning	18
Table 3.1 breakdown of the dataset categories	22
Table 4.1 Experimental Setup Summary	56
Table 4.2 Overall Performance Comparison of All Models	58
Table 4.3 Top-5 Model Ranking.....	60
Table 4.4 Class-wise Metrics for Selected Best Models	63

Chapter 1 Introduction

1.1. Problem Definition

The online news consumption has overtaken the traditional media in the digital age and this has resulted in the proliferation of information exponentially. Although this new digital world has made information more democratic to people, there is an increasing circulation of fake news and information that challenge public trust, stability in politics and social harmony. The fast spreading of fake news by means of social networks, news portals, and short video platforms has led to a critical necessity of the automated and intelligent detection systems that will be able to identify the fabricated or manipulated information in real-time.

Detection Fake news has become an important subdiscipline of Machine Learning (ML) studies. Different researchers have conducted researches to compare content-based and context-based methods of differentiating authentic and counterfeit news articles. Pereira et al. [1] continued this trend by introducing multi-view autoencoder networks, which can be used to train deep models to learn cross-domain representations to learn semantic subtleties between heterogeneous data views.

Some survey studies have summarized the developments and problems in the field. The methods of multi-view bootstrapping presented by Ying et al. [2] also improve the detection strength by using various news features, including author credibility, and propagation dynamics.

New attempts have been focused on the exploitation of user interaction networks and credibility modeling. Min et al. [3] suggested a divide and your conquer model that characterizes the user interaction on the social media to achieve higher accuracy in fake news detection whereas Bazmi et al. [4] devised a co-

attention network that encodes the topic specific user interaction and the credibility of the news source.

The growing sophistication of the misinformation patterns has prompted deployment of deep learning structures and representation learning models. Zhang et al. [5] integrated Graph Convolutional Networks (GCN) and the models of BERT-based co-attention, which are important in enhancing the semantic reasoning of detection models.

Multilingual misinformation has also become a challenge of interest. Alkudah et al. [6] conducted a comparative study of machine learning and deep learning frameworks on fake news detection in Arabic media by showing that hybrid solutions that integrate CNN and Bi-LSTM are better than the traditional classifiers, particularly when they are trained on language-specific data.

Although there have been major improvements, the area of fake news detection continues to have unresolved problems, which include dataset bias, model interpretability, domain adaptation, and cross-lingual scalability.

The key questions guiding this research are as follows:

- RQ1: What is the way to effectively integrate machine learning to identify fake news articles based on textual information?
- RQ2: Which preprocessing and feature extraction methods (e.g., Bag-of-Words, TF-IDF, Word2Vec) are the most effective in terms of their high accuracy and generalization in a wide range of news topics?
- RQ3: Which machine learning or deep learning algorithm, like Logistic Regression, Random Forest, Support Vector Machine (SVM), or Long Short-Term Memory (LSTM), is the most trustworthy in terms of classification in the detection of fake news?
- RQ4: How effective are state-of-the-art multi-view and explainable AI solutions in improving model interpretability and trust in fake news detection systems?

- RQ5: What are the ways that the results of this research can lead to the development of scalable, transparent and domain adaptive fake news detectors that can be deployed in the real world?

Such research questions provide both technical and ethical aspects of an issue-technical in how to create effective algorithms, and ethical in how to make automated misinformation filtering transparent and accountable [1], [6], [10].

1.2. Thesis Objectives

The growing complexity of the production and dissemination of fake news has posed pressing concerns about the manner in which computational systems can be able to differentiate authoritative and fake information correctly and ethically. In this regard, the proposed research will seek to understand how to incorporate Machine Learning (ML) tools to develop the powerful and explainable model that can identify fake news. The main research question is motivated by the fact that it is necessary to comprehend how text-based and semantic attributes can be identified and exploited to enhance the extent to which deceptive content can be classified. Therefore, some research questions have been developed to inform this study.

The research purposes are clearly associated with the questions provided above and are aimed at developing a multifaceted framework of fake news detection based on text data. Particularly, the research aims at preprocessing and cleaning the data set through the method of tokenization, removal of stop-words and lemmatization to warrant the quality and uniformity of textual input. Moreover, it will attempt to derive meaningful linguistic and semantic content in the news articles with the help of vectorization techniques like TF-IDF and word representations, make the text information to be effectively represented in terms of computational models. The study also aims to design and train several of them, such as Logistic Regression, Random Forest, Support Vector Machine,

and Long Short-Term Memory (LSTM) to assess and compare the results of their usage in detecting fake news. The standard performance measures including accuracy, precision, recall, F1-score, and ROC-AUC will be used to evaluate the models as they present a broad picture of the ability of any particular model to classify data.

These objectives are driven by two things, academic and societal. Academically, the study has been able to enhance the current advancement of the computational methods used in detecting misinformation by providing the benchmark between the traditional and deep methods on a large balanced dataset. Socially, the research contributes to the fight against the proliferation of misinformation in the entire world and rebuilding digital trust in news media.

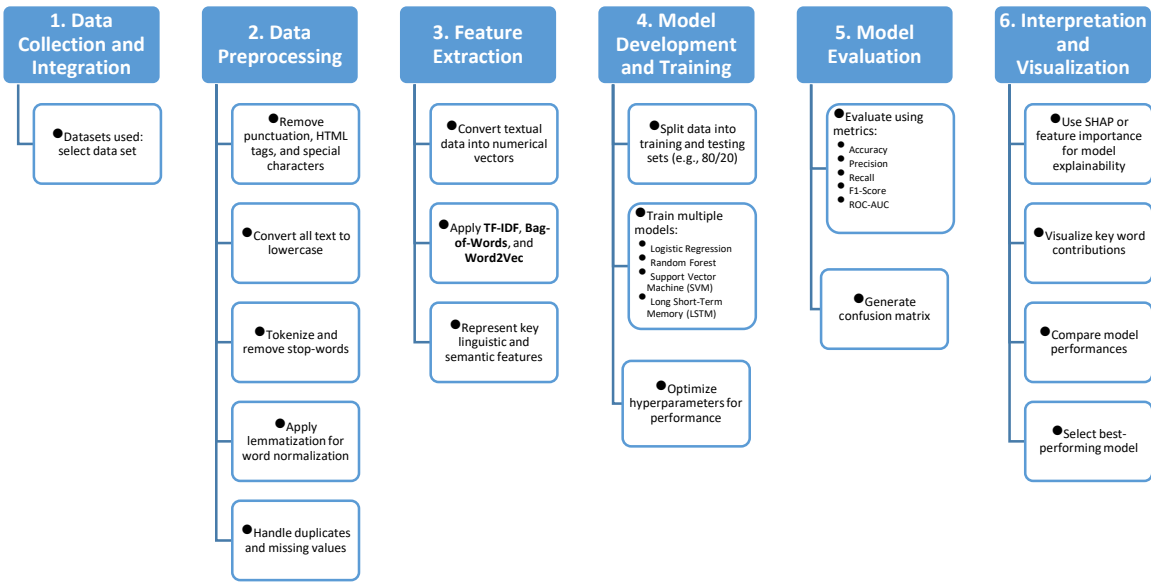


Figure 1.1 Experimental Workflow

1.3. Thesis Contribution

By answering the above objectives, the study will also be able to make a contribution to the growing field of work in computational misinformation detection through:

- Setting up a benchmark system to classify fake news on a structured English data set.
- Comparison and contrast of classical and neural methods of textual misinformation analysis.
- Emphasizing the need to make machines explain why something is true or false and the need to be ethical in content verification systems.

Overall, the proposed study aims to address issues related to the gap between theoretical research and real-life application to suggest a model that is structured, data-driven, and explainable and can help media organizations, policymakers, and technology platforms detect and counteract the spread of fake news in real-time.

1.4. Thesis Organization

The thesis is as follows:

- Chapter 1 presents history, statement of the problem, purpose and importance of the research.
- In chapter 2, the literature review on fake news detection from multiple perspectives, including linguistic-based, network-based, and multimodal approaches.
- Chapter 3 provides description of the proposed methods such as data pre processing, graph modeling, and machine learning and deep learning models employed.
- Chapter 4 writes about the experimental configuration, experimental results and analysis, presenting the performance outcomes of each model and comparing their effectiveness in detecting fake news.
- Chapter 5 is a conclusion of this work that throws emphasis on the important findings, limitations, and future directions of the study.

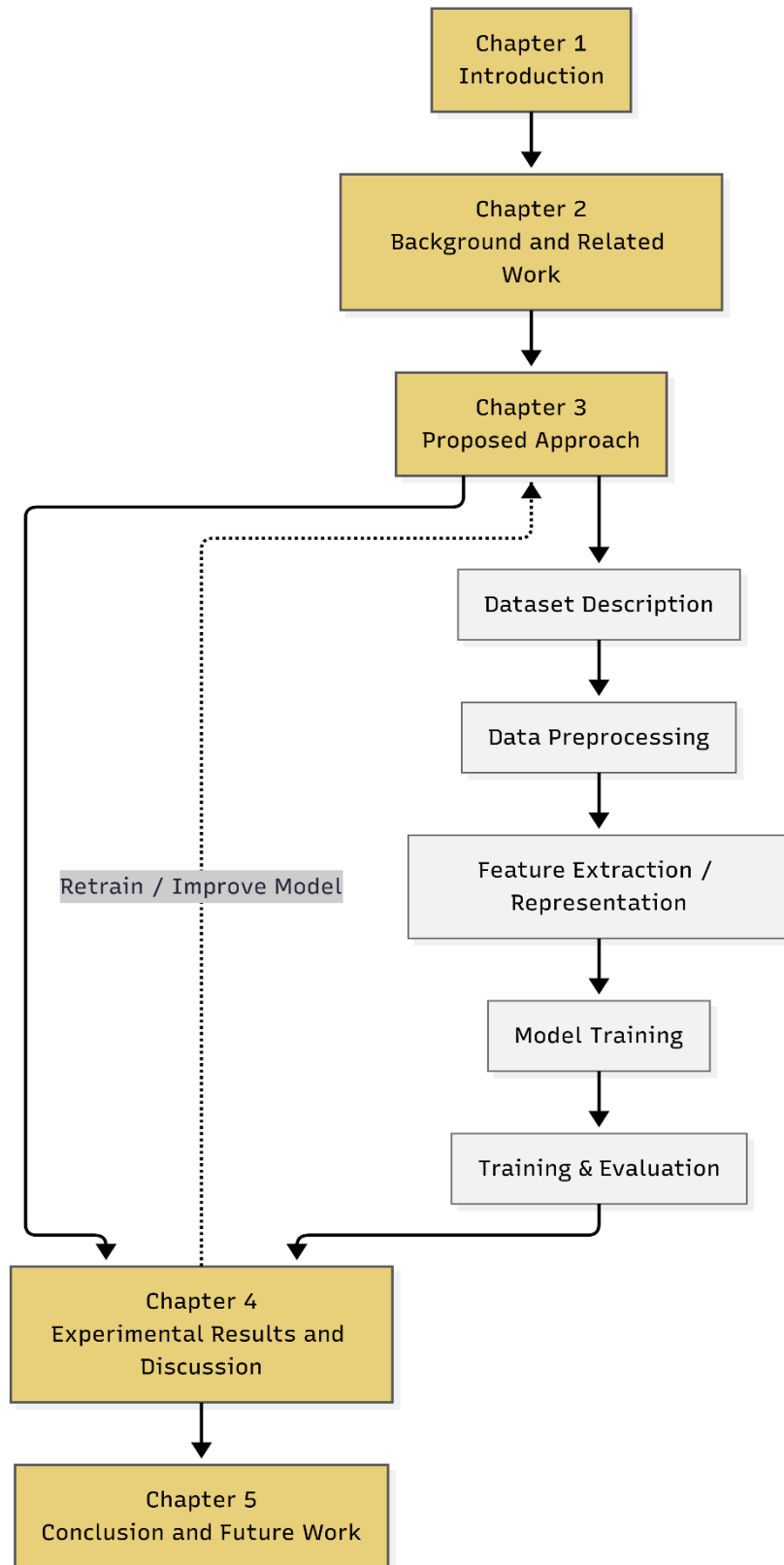


Figure 1.1 visual representation of how the chapters, research files, and model iterations interact throughout the study

Chapter 2 Background and Related Work

2.1. Background

With the introduction of the internet and the overwhelming growth of social media sites, the spread of information is immediate and international. This revolution has shifted the very nature of news consumption among the people in preference to speed over fact-checking. Although this connectivity can be used to share knowledge quickly, it can also be used to propagate fake news which can be defined as intentionally made or misleading information about a specific topic being presented as factual content [1]. This kind of misinformation has been identified to distort the perceptions of the people, affect campaigns, fuel social unrests, and this has also led to a lack of trust in professional journalism [5], [6].

Early detection models were very dependent on superficial linguistic features namely word frequency, sentiment polarity and grammatical structure, to differentiate between genuine and fake news [1], [6]. Nevertheless, the development of fake news material has rendered these strategies inadequate. The modern information distortions involve the use of sophisticated language structures, emotionally charged stories, and multi-modal features (images, videos, hashtags), which have to be comprehended by more sophisticated models that can discern a semantic, contextual, and relational pattern [2], [1].

In this regard, scholars have investigated numerous paradigms. Content models examine semantics of the text, whereas context models combine user interaction information and source reliability [2], [4]. Combining textual, visual, and social dimensions, hybrid and multi-view learning strategies, like the one suggested by Pereira et al. [1] can enhance the effectiveness of detection.

Though these have been advanced, there are still a number of challenges. First, there is a tendency to have dataset imbalance and domain dependency thus resulting in models, which do well in one application environment, and fail in a different application environment [6]. Second, semantic ambiguity and intentional sarcasm or satire make the detection through text more difficult [5]. Third, the emergence of AI-generated content and deep fake videos has brought about novel types of misinformation, which cannot be reduced to text manipulation [2].

Fake news has become one of the most topical phenomena of the digital information age that has significant impacts on the opinion of people, political rhetoric, and social harmony. Because of the popularity of social media sites and the ability to publish online, fake information can now reach out to millions of users in seconds, which can be verified by traditional means, notwithstanding. As a result, scientists and technologists have more and more concluded to the use of computational intelligence, or to be more precise, machine learning (ML) and deep learning (DL) to automate the process of misleading or deceptive news content. This chapter is the review of the literature on the topic of the fake news detection with the help of the ML and DL approaches, analyzing the conceptual background, methodological tendencies, and technological innovations that influenced the given sphere.

The research on fake news detection can be divided into some broad streams. The former stream consists of content-based techniques, which examine textual or linguistic features e.g. syntax, semantics and sentiment with the aim of revealing deceptive patterns. The second is context based and social based in approach with a strong interest in the role of user behavior, propagation networks, and source credibility in differentiating between authentic and

fabricated information. The third and the latest stream incorporates multi-view and multi-modal frameworks which are a combination of different perspectives textual, visual and social in order to understand the complex and dynamic nature of misinformation dissemination.

Machine learning techniques, such as Support Vector machines (SVM), random forest and Logistic regression, have long dominated initial research because they are interpretable and also efficient in structured feature spaces.

2.2. Conceptual Introduction and Fake News definition.

The fake news has become a widely spread notion of the digital age as the misinformation spreads faster on the online media platforms. In its most basic definition, fake news can be called the deliberate creation or deception of information which is offered under the guise of real news in order to elicit certain ideological, political, or financial implications on the audience. It is not just a question of textual falsity but it is a concerted mechanism of dissemination on both sides, on the one hand, content producers and on the other hand, social network players [4].

Analytically speaking, initial studies concentrated mainly on the detection of linguistic anomalies, i.e., vocabulary use, mood, composition, etc. as a form of falsification. The later research, however, identified that the intent of deception and virality could not be completely captured with the help of text because users and credibility of the source are also important factors [9]. It is this development that gave rise to hybrid analytical paradigms that combine natural language processing, network analysis and behavioral psychology.

The fake news conceptual definition also changed with the technological progress. With neural networks and language models reaching the point where they could produce human-like text [5], the distinction between genuine and synthetic text became even less distinct, and more advanced detection

mechanisms are required. Modern definitions, therefore, include intentional falsification as well as automated generation of misinformation and this time might not necessarily be human authored.

2.3.Representation of Features and Data Modalities.

The representation of features is a crucial aspect of fake news detection as the ability of a machine learning system to separate authentic and fake content depends on it. It is always stressed in the literature that a credible representation should reflect not only the language features of the news text but also the social and cultural aspects of its distribution. Content based representations dwell on the textual characteristics of the news articles that include lexical patterns, semantic coherence, and sentiment polarity [5].

Conversely, features based on the context are dependent on external features, including the level of user interaction, credibility of the source, and the pattern of propagation through social networks. The paper by Min et al. [3] coined the notion of post-user interaction networks, showing that the structural data regarding the user interaction, including comment threads or retweet graphs, can be much more predictive than textual information. In the same manner, Bazmi and Shakery [4] have modeled user and source credibility on topics by adopting multi-view co-attention models and highlighted that reliability of a claim is closely related to the trustworthiness of the claim publisher and the audience.

The representations that have become popular include hybrid and multi-view representations since they combine various modalities into a single latent space. Ying et al. [2] introduced a textual and social views bootstrapped multi-view strategy which concurrently undertakes textual and social views to enhance multi-platform generalization. Pereira et al. [1] developed this direction by introducing multi-view autoencoders which are trained on the heterogeneous

data sources that learn shared representations, effectively integration of linguistic information, user behavior and visual information[5].

Together, these works reflect the paradigm shift of a single-modality analysis of the text to a multi-modal feature integration, where the text, user behavior, visual or auditory content are in interaction with each other. This development is an indication of an increased realization of the fact that misinformation detection involves a comprehensive grasp of what is being expressed, how, and by whom it is expressed. These full-fledged representations are the foundations of the modern deep learning architectures and will guide the future generation of fake news detectors[6],[7].

2.4. Conventional machine learning practices.

Prior to the emergence of deep learning, fake news identification was entirely dependent on conventional machine learning (ML) algorithms, which implied the need to perform explicit feature engineering. These traditional methods had a basis in the belief that fake news could be detected by the help of the specific linguistic and statistical features. According to Kesarwani et al. [7], the algorithms initially used in the social media misinformation detection were the K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Naive Bayes, Decision Trees, and Logistic Regression. These models rely on hand designed features based on word frequencies, part-of-speech labels, sentiment polarity and stylistic differences.

The traditional ML methods have one benefit in that they are interpret-able; feature weights and decision boundaries give understandings into the linguistic and structural cues of deception. Along with these strengths, the conventional ML models have significant drawbacks [7]. They rely on handcrafted features, which are sensitive to variations in domains and languages besides being time consuming to extract. Such models find it hard to be generalized in data-sets due

to the fact that fake news writing styles vary in regions and subjects. Moreover, they are not easy to learn even simple contextual or temporal interactions, like how misinformation propagates over user interaction networks, without significant feature prediction [8].

2.5. Deep Learning Approaches

The development of deep learning has had a significant impact on the evolution of fake news detection since it substituted the manual feature engineering with data-driven representation learning. Contrary to the classical structure of machine learning models, deep architectures are capable of automatically learning high-dimensional correlations between the linguistic features, semantics and contextual information [10].

Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and their variants, like the Long Short-Term Memory (LSTM) and Bidirectional LSTM (BiLSTM) models are among the most popular architectures that are widely used. The RNN-based architectures were used to identify fake news by Cetiner [9], and it was established that sequential modeling of linguistic patterns can allow the systems to learn how to associate the time dependencies between sentences.

The GBCA model proposed by Zhang et al. [5] combines Graph Convolutional Networks (GCN) with BERT based on the co-attention line, and it has been shown to yield better generalization when combining language and relationship features. These architectures are effective to learn representations of the text content and the network relations between the users and publishers.

Multi-modal information integration has also been made easy through deep learning models. Even though deep networks have recorded state-of-the-art results, they exhibit a black-box behavior that presents challenges on interpretability and explainability [10].

2.6. Multi-View, Graph, and Multi-Modal Models.

The recent years have seen the growing interest in multi-view, graph-based and multi-modal models to detect fake news, as the idea that misinformation cannot be perceived through the text messages only has taken hold. Rather it is born out of the interaction of various views - textual, social, temporal and visual - which are used to give different signals. Pereira et al. [1] proposed multi-view auto-encoders (MV-AE), where the latent representations are trained with the help of encoding a variety of information streams such as the text of the article, social context, and user interaction data. This unsupervised method is superior to single-view baselines and finds association between different sources that are heterogeneous. On the same note, Ying et al. [2] suggested a bootstrapped multi-view representation system to match textual and propagation embedding and hence overcome the issue of partial or inaccurate data metrics by enabling one view to complement the other.

It has turned into a building block of future detection systems that is based on graphs. A multi-view co-attention network proposed by Bazmi and Shakery [4] also focuses on topic-specific relationships between users and news sources, and credibility as a relationship feature, but not as an intrinsic attribute. This can be best achieved with Graph Convolutional Networks (GCNs) by modeling the diffusion of fake news over social ties and diffusion of trust or skepticism over groups of users.

The multi-modal architectures further extend the paradigm by incorporating text with other modalities like images, audio and video. Zhang et al. [5] introduced a co-attention mechanism (GBCA) to unite GCN and BERT that allows learning semantics of texts and network structure simultaneously. These models also indicate a significant epistemic change: fake news is ceased to be viewed as the

document classification task but as a multi-relational phenomenon that is dynamic [6].

2.7.Domain and Language-Specific Studies.

Even though most of the research regarding the detection of fake news is conducted using English language datasets, recent research indicates the significance of the model adaptation to different linguistic and cultural contexts. The idiomatic expressions, the language peculiarities, and the dynamics of social media play a major role in spreading and perceiving misinformation. One of the most thorough studies on the detection of Arabic fake news was done by Alkudah et al. [6], who compared the applications of machine learning and deep learning to the Arabic Fake News Dataset. Their results showed that although deep learning models (e.g., CNNs and LSTMs) are more effective than traditional models, linguistic preprocessing is important because of the morphological complexity of Arabic and dialect differences. Equally, Al-Taie [9] conducted a comparative study in which word embeddings specific to the Arabic language (e.g., AraVec and multilingual BERT) have a significant positive effect on the performance of detection, as opposed to generic embeddings.

The problem of domain adaptation is not only limited to language but also to content type. As an example, author [3] concentrated on short-video media where misinformation is presented in visual and auditory, as opposed to textual, form.

2.8.Surveys, Comparative Studies and Problems.

The development of the fake news detector has been a well-documented phenomenon with a significant number of surveys and comparative studies, all of which provide a systematic perception of the methods, data, and issues. One of the most detailed taxonomies is introduced author [10], who divide ways of

detecting fake news into content-based, context-based, and hybrid frameworks. They compared the effectiveness of the linguistic and the network-driven ones and have found that the hybrid models, which incorporate the user interaction, publisher credibility, and semantic coherence, have always been more efficient than one view system only. Their discussion indicated the overlap of fact-checking and fake news detection especially in the field of natural language inference and knowledge-graph reasoning.

Domenico et al. [8] conducted a systematic review and associated fake news, social media dynamics, and marketing, showing how fake news campaigns take advantage of the emotion contagion and the algorithmic amplification processes. Like these insights help to connect the technical and behavioral approaches to the problem and support its interdisciplinary nature.

Relative comparative studies like those conducted by Alkudah et al. [6] tested how machine learning and deep learning models were relative to each other in the Arabic setting. Their findings affirm that linguistic diversity and an unbalanced dataset are important in accuracy and present a great limitation of cross-domain generalization.

There are a number of common challenges in all comparative works: labeled multilingual data is scarce, the models can be adversarial manipulated, there are no standard benchmarks, and the models are challenging to explain. Also, the majority of the detection systems are reactive, not proactive, as they are directed at detecting misinformation once it is published and not in anticipation of its appearance. All these factors highlight the importance of the need to conduct future research to incorporate the concepts of transparency, flexibility, and cross-linguistic soundness into the framework of fake news detection so that it is not only effective but also reliable in practical use.

2.9. Summary and Research Gaps

The review of fake news detection with machine learning shows impressive development in the last ten years, where the focus shifted on the basic classification of lexical entities to the advanced, multi-view, and multi-modal deep learning networks. Initial studies were devoted to extraction of textual features with classical machine learning algorithms SVM, Naive Bayes and Passive-Aggressive Classifiers [7]. These techniques were based on manual based linguistic indications and stylistic indicators and they though interpretable, were not flexible to invisible information. The next research trend adopted deep learning, which allowed automatic representation learning and semantic reasoning with models such as CNNs, LSTMs, and transformers [5], [9]. This change had a significant positive impact on the accuracy of detection and its robustness, but also created problems in the explainability and data dependency. Multi-view and graph-based studies, i.e., a study by Pereira et al. [1], a study by Bazmi and Shakery [4], have redefined the fake news detection as a relational issue. These models combine textual message, user action, and network, proving that it is impossible to comprehend misinformation out of its social and temporal context. In the meantime, the unsuitability of single-size-is-all models is emphasized by domain-specific studies (e.g., Alkudah et al. [6]). The language peculiarities, differences in culture, and inaccessibility of data in non-English language remain obstacles to generalization. Likewise, researchers of multimedia misinformation [3] indicate the increasing sophistication of fake news in question whereby manipulation is conducted in the forms of videos, audio, and images instead of text only.

In spite of these developments, there are still a number of research gaps. First, cross-lingual transferability is restricted by the absence of big, balanced and multi-lingual datasets. The majority of the currently available corpora are still

English-oriented, hindering further development of the areas, where misinformation factors are linguistically specific. Second, hybrid and deep models have high accuracy but lower interpretability, which makes automated systems less transparent and trustworthy [8]. Third, the detection of fake news is mainly reactive, and it is more oriented towards classification instead of early warning or prevention systems. Fourth, adversarial robustness has not been thoroughly studied so far fake content creators are becoming more and more able to take advantage of model vulnerabilities using generative AI methods [10].

Another multimodal fake news detector is proposed by Lu and Yao and combines multi-head attention mechanisms and residual convolutional networks. They take text, images, and video as their input and allow cross-modal alignment and further semantic fusion. The model shows good results in the datasets of LIAR (0.977 accuracy, 0.986 precision, 0.969 recall, F1 0.924), FakeNewsNet (0.986 accuracy, 0.979 precision, 0.965 recall, F1 0.900), and Weibo (accuracy 0.951, precision 0.961, recall 0.926, F1 0.843). It performs better than strong baselines like It has been shown that the study is exceptionally robust, resistant to perturbation of the data, and remarkably fast in inference (less than 0.02 s). The authors conclude that a combination of multimodal fusion, residual architectures, and attention can be used to produce a more reliable and deployable fake news detection system in a variety of media settings [9].

Al-Obaidi and Çağlıkantar compare a wide variety of machine learning and deep learning-based fake news detectors that are based on ISOT, FakeNewsNet, and FakeNewsNet + extensions (Half-True and Barely-True). Their work focuses on binary and multiclass detection, SVM, RF, XGBoost, LSTM, CNN-LSTM, and CNN-GRU are the testing algorithms. Findings indicate extremely high performance of LSTM-based models with an accuracy of 99.95% (ISOT) and 99.97% (FakeNewsNet). The authors point out that the performance differs

greatly in different datasets, and the incorporation of nuanced classes (Half-True, Barely-True) makes the models more complex and lowers their accuracy, especially in the case of LSTM AutoEncoder. Their research shows how the dataset structure, the ability to model text only, and alignment between models and datasets play a crucial role in the achievement of the best fake news detection accuracy [10].

To sum up, in all studies reviewed, the conclusion is that there is a necessity in explainable, adaptable and culturally inclusive detection frameworks. The research approach to be undertaken in future should revolve around cross-domain generalization, low-resource language modeling, and ethical AI incorporation as a way of improving the technical effectiveness and societal credibility of fake news detection models.

Table 2.1 Summary of Key Studies in Fake News Detection Using Machine Learning

Ref.	Author(s) & Year	Dataset Used	Algorithm / Technique	Evaluation Results	Main Conclusion
[1]	Pereira et al. (2025)	FakeNewsNet & social graphs	Multi-view autoencoder (MV-AE)	Accuracy 93%	Combining content & social signals in shared latent space improves detection.
[2]	Ying et al. (2023)	PolitiFact, GossipCop	Bootstrapped multi-view representation	90% accuracy	Aligning multiple feature views reduces noise and boosts generalization.
[3]	Min et al. (2022)	Twitter & Weibo data	Post-user interaction network (PUI-Net)	F1 $\approx$ 0.89	User interaction structure offers high predictive value beyond text content.
[4]	Bazmi & Shakery (2023)	Credibility-labeled corpus	Multi-view co-attention network	Accuracy 91%	Topic-specific modeling of users and sources enhances relational credibility scoring.
[5]	Zhang et al. (2024)	FakeNewsNet	GBCA: GCN + BERT + co-attention	Accuracy 95%	Integrating graph and textual features significantly improves detection accuracy.

[6]	Alkudah et al. (2025)	Arabic Fake News Dataset	CNN, LSTM, SVM comparison	DL: 92%, ML: 85%	Deep learning surpasses ML for morphologically rich Arabic text.
[7]	Kesarwani et al. (2020)	Twitter fake news	K-Nearest Neighbors (KNN)	84% accuracy	KNN effective for small datasets but limited scalability.
[8]	Domenico et al. (2021)	Marketing misinformation datasets	Systematic literature review	—	Fake news thrives via emotional contagion and algorithmic amplification.
[9]	Lu, Y., & Yao, N. (2025)	- LIAR dataset (political short text) - FakeNewsNet (English multimodal news) - Weibo dataset (Chinese social media multimodal data)	- Multimodal residual convolutional network - Multi-head attention mechanisms - Multimodal feature fusion (text + image + video) - Cross-modal alignment module	Best-case performance: - Accuracy: 0.977 - Precision: 0.986 - Recall: 0.969 - F1 score: 0.924 Also shows strong robustness, stability, and fast response (<0.02 s).	Integrating residual CNNs with multimodal attention significantly improves fake news detection across text, image, and video. Model is robust, fast, and performs better than BERT, RoBERTa, XLNet, ERNIE, and GPT-3.5 baselines.
[10]	Al-Obaidi, S. A., & Çağlıkantar, T. (2024)	ISOT, FakeNewsNet, Dataset1, Dataset2 (ISOT + LIAR-PLUS: Half-True, Barely True)	Multiple ML models (SVM, LR, DT, RF, XGBoost, Gradient Boosting) + Deep Learning models (CNN, CNN-LSTM, CNN-GRU, LSTM, LSTM AutoEncoder)	- ISOT: LSTM accuracy 99.95% - FakeNewsNet: CNN-GRU accuracy 99.97% - Dataset1: CNN-LSTM & LSTM nearly 100% - Dataset2: LSTM 98.64% - LSTM AutoEncoder lowest performance (49%–62%)	Model performance depends heavily on dataset structure. LSTM and CNN-based hybrids outperform others, while LSTM AutoEncoder is less suitable for fake news detection. Multiclass extensions (Half-True, Barely True) improve nuance in prediction.

2.10. Chapter Summary

The conceptual, methodological, and empirical development of detecting fake news based on machine learning provided in the literature review in this chapter demonstrated a shift in simple text-based detectors to more advanced multi-view and multi-modal detectors that are based on deep learning. In all the reviewed papers, it becomes clear that the issue of misinformation is not limited to the linguistic manipulation but goes far deeper, into the social, contextual, and

behavioral aspects. The chapter disclosed the way in which fake news has become a multidimensional research area involving the combination of natural language processing (NLP), network analysis, and modeling of cognition.

Initial works, e.g., by Kesarwani et al. [7], defined what feature-based learning would be based on, i.e., textual indicators of word frequency, sentiment, and syntax to learn on a conventional machine learning structure (SVM, KNN, Naive Bayes). These models proved to be reasonable but had restrictions in the form of manual feature engineering and in-scaling. Later studies took the direction of deep learning, such as Zhang et al. [6] who used CNNs, LSTMs and Transformer-based models (e.g. BERT, Falcon) that can learn complex semantic patterns independently. The excellent results of these models, which in many situations reached the 90 percent mark, justified the shift to data-driven representation learning.

The next significant concept that was covered is the emergence of multi-view and graph-based models where papers like Pereira et al. [1], and Bazmi and Shakery [4] used a combination of both textual and social cues to formulate relational dependencies among users, sources and content. These models were more generalizable and interpretable because they thought of misinformation as being a network phenomenon and not an unchanging document classification problem. Also, domain and multilingual studies, such as those, by Alkudah et al. [6] has highlighted the linguistic and cultural peculiarity of misinformation, especially in the Arabic and other non-English languages, and have shown that there is a severe deficit in cross-lingual generalization.

Chapter 3: Proposed Approach

3.1.Introduction

The chapter outline is the methodology of designing and testing fake news detection models that rely on machine learning (ML) and deep learning (DL) models. The aim of the study is to come up with an automated model that can identify both real and fake news articles using the ISOT Fake News Dataset. The approach is divided into several steps such as acquiring the dataset, data preprocessing, feature representation, model training and performance evaluation. All the stages are designed in order to be able to guarantee the reproducibility, scalability and reliability of the results.

The use of both ML and DL approaches is intended to deliver a comparative insight into the traditional models of features engineering and the recent neural model that is guided by learning the representations of the data. Machine learning methods like the Support Vector Machines (SVM), Logistic Regression, and Random Forest are chosen because they are interpretable and computationally efficient, whereas deep learning methods like the Long Short-Term Memory (LSTM) and Bidirectional LSTM (BiLSTM) are chosen due to their capability to learn semantic and contextual relationships in textual data. Through the investigation of the two paradigms, the study aims to establish the degree to which the automated extraction of features involving the use of deep networks can perform better than a manually designed method and still be explainable.

3.2. Dataset Description

Isot Fake News Database The main dataset that will be used in the current study and will be a strong benchmark in the evaluation of fake news detectors. This data set is comprised of the news items found in the credible and untrustworthy

internet sources over the period of 2016 and 2017. The real news subset has been created based on the Reuters.com, the internationally known and credible news agency whereas the fake news subset was found on the different web sites which are classified as misinformation sources by the Politifact and Wikipedia websites. The data is separated into two CSV files, True.csv which includes genuine news and Fake.csv which includes fake articles. Every file contains four important features namely, the title of the article, the main text, the category of the subject and the date when it was published and a combination of these features forms the complete text of the article, which can be used in the classification processes.

Overall, the collection that was used consists of about 44,898 news articles, which consist of 21,417 true and 23,481 fake entries. Under real-news category, there are 10,145 articles that fall under World News category and 11,272 articles that fall under Politics News category. The fake-news category, in its turn, contains 1,570 Government News, 778 Middle-East, 783 U.S. News, 4,459 Left-News, 6,841 Politics and 9,050 general News articles. This allocation proves that political and world news occupies the leading place in both categories indicating the thematic bias that is frequently inherent in the process of misinformation.

Each of the articles has four main features namely the title, text, subject, and date of publication. The information was pre-cleaned with the original style of writing and punctuation being retained to maintain authenticity. Table 2 presents the categories of elements of the dataset in a detailed breakdown.

Table 3.1 breakdown of the dataset categories

News Type	Total Articles	Category	Articles per Category
Real News	21,417	World News	10,145
		Politics News	11,272
Fake News	23,481	Government News	1,570

		Middle East	778
		U.S. News	783
		Left News	4,459
		Politics	6,841
		General News	9,050

The dataset provides a balanced and diverse set of topics, with politics and global news dominating both classes. The total sample size of 44,898 articles ensures sufficient variability for robust training and testing across algorithms.

3.3.Data Source

The data in this research is the ISOT Fake News Dataset which is also popular as a standard in the research of fake news detection [11]. The dataset is divided into two individual CSV files, one of which is True.csv which includes real news articles obtained through trusted sources, including Reuters, and another one is Fake.csv that collects fake news articles found on different unreliable and misleading websites.

Every record within the dataset comprises four primary attributes, namely the news title, the article text, and the subject category, as well as the publication date. The dataset has around 44,898 articles, which is large enough and varied enough to be used in the training and evaluation of machine learning and deep learning models.

This data set was chosen because it is balanced and is relevant to the textual fake news detector tasks.

3.4.System Overview

The suggested system was built as a full-fledged end-to-end fake news detector system. It is in its architecture based on successive phases of linked steps that lead to the ultimate classification decision. The former stage is concerned with the data acquisition and data construction, where data on fake and real news is loaded and assembled into one corpus. The second phase does the text cleaning and

normalization to transform raw news articles to a more homogenous form that is computational friendly. The third phase converts the text that has been purged into machine numbers that can be read by the machine and fit into various model groups. The fourth step involves training a number of classifiers based on these representations and the last step involves comparing the performance of these classifiers using the standard evaluation metrics. This stratified design renders the system to be modular and simple to comprehend.

One of the major design principles in the system is the adoption of various textual representations as opposed to using a single encoding strategy. Conventional machine learning models work with text best when it is transformed into sparse vectors e.g. TF-IDF, whereas neural networks require integer token sequences with zero length padding. In the same manner, word-level semantic representations can be represented by Word2Vec, whereas stylistic or subword hints can be represented by character n-grams. Combining these options within the same framework, the system helps in enriching analogy between lexical, semantic, sequential, and hybrid types of representations. This becomes particularly applicable to the area of fake news detection where malicious writing can manifest itself in the form of content schemes and stylistic anomalies.

The other feature of the suggested system is that it does not end at the classification of single models. Having trained the baseline and deep models, the study also presents combined models, such as a word-plus-character hybrid SVM model, an ensemble stacking model which uses a logistic regression meta-learner, and an ensemble based on voting, using a deep plus machine learning model. These additions are based on the concept that various models could represent various elements of the data and this integration can enhance strength and predictive validity. On the whole, it is not just a set of independent experiments but a single classification pipeline aimed to investigate the influence of alternative representation

and modeling strategies on the fake news detection performance in one controlled implementation environment.

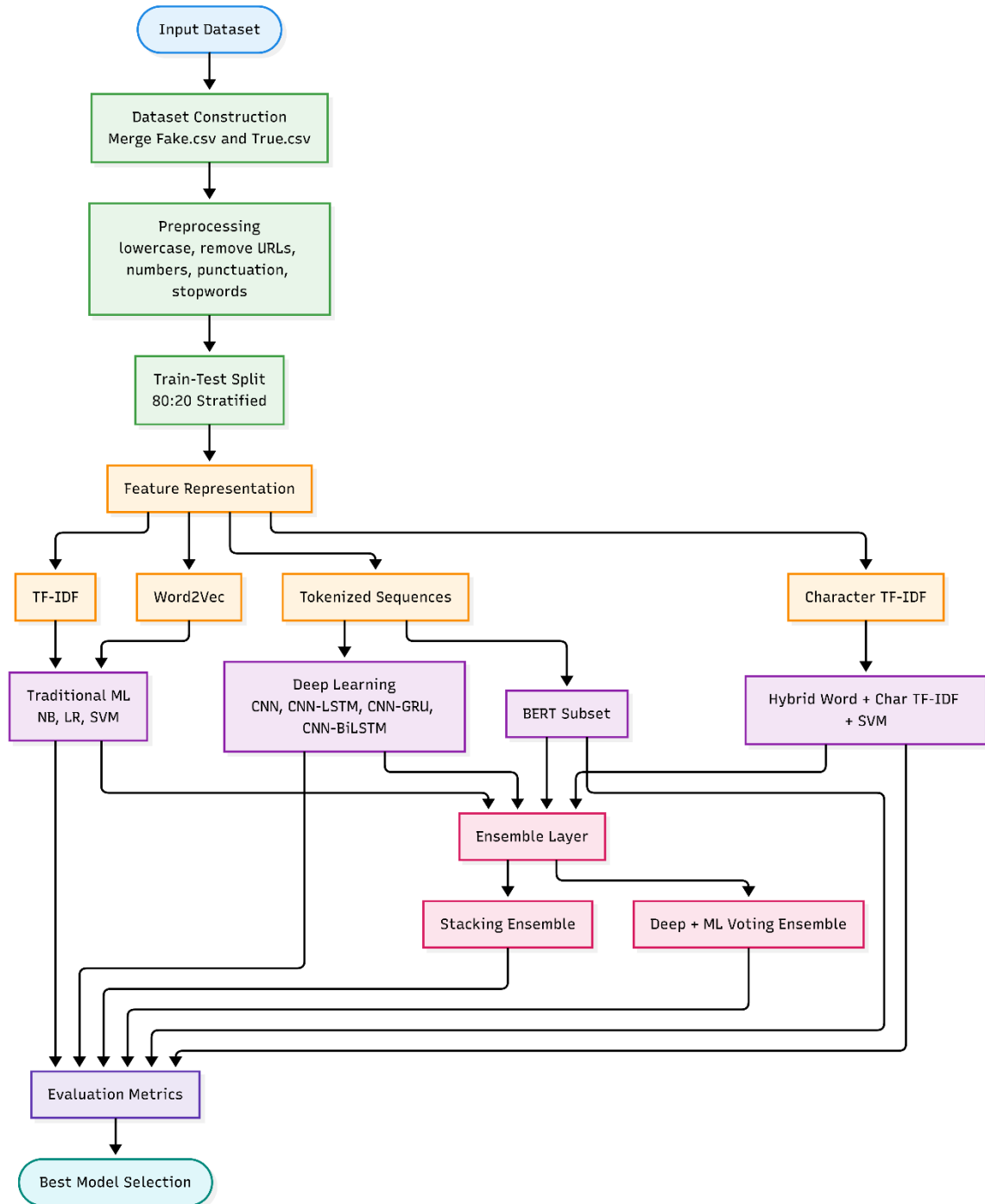


Figure 3.1 Proposed framework of the fake news detection system

3.5.Experimental Environment

In this study, all experiments were performed in the Google Colab environment, which is a cloud-based platform to execute Python code and has access to high-performance computational resources, such as the use of a graphics card (GPU acceleration). Google Colab eliminates the local hardware limitations and provides effective processing of massive datasets and models.

It was implemented in Python programming language with the help of a collection of popular data science and machine learning libraries. They are Pandas to manipulate and preprocess data, NumPy to perform numerical operations, and Scikit-learn to apply classical machine learning algorithms and metrics to evaluate them. Moreover, the creation and training of deep learning models were done with the help of TensorFlow and Keras.

This experimental design guarantees the reproducibility, scalability and computational efficiency of this experimental design and therefore it is applicable to both classical and deep learning methods used in detecting fake news.

3.6. Data Preprocessing

Data preprocessing is another essential part of the text classification process since it will convert raw textual information into an organized and machine understandable data that can be utilized to train the models. A systematic preprocessing pipeline was adopted in this study to provide quality and consistency in data. First, the two dataset files (True.csv and Fake.csv) were combined into one dataset and a new textual attribute was created by concatenating the title and the body of each article.

The dataset was then cleaned to eliminate noise and other irrelevant factors such as URLs, punctuations, numbers, and standard stopwords and all the text was

transformed to lowercase to ensure a consistent dataset. A labeling step followed after the cleaning step where real news articles received a label of (0) and fake news articles received a label of (1) was used to allow supervised learning.

Stratified sampling was then used to split the dataset into training and testing sets (80/20 respectively), making sure that both the classes are equally represented. Lastly, feature extraction methods like TF-IDF to be used with traditional machine learning models and word embeddings to be used with deep learning models were used to translate the processed text into numerical representations. This pipeline makes sure that the data in the input is well formatted and optimized in order to make the model perform well.

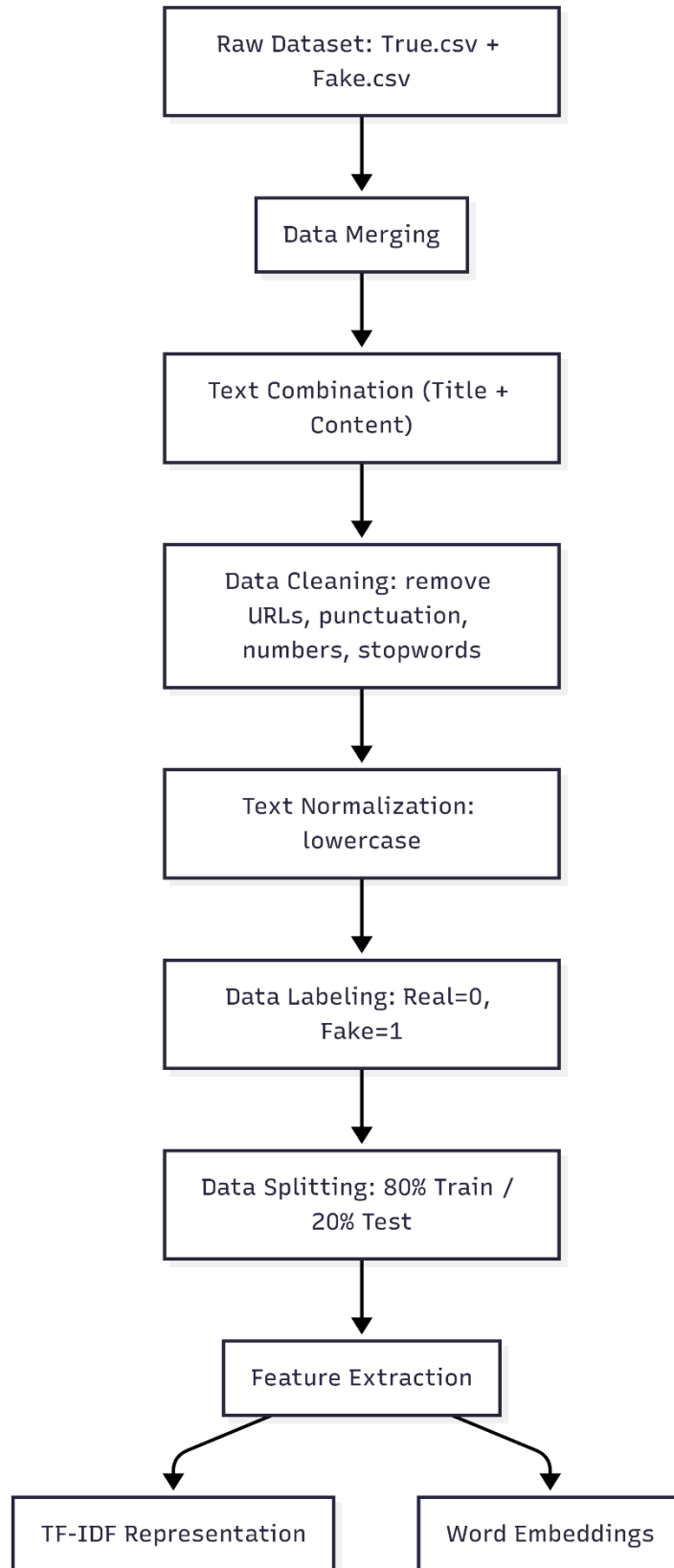


Figure 3.2 Preprocessing stages

3.7. Model Development

The phase of model development will be concerned with designing and training machine learning and deep learning-based architectures to classify fake news. The combination of these two paradigms makes it possible to perform a comparative analysis of the traditional models that are based on explicit feature engineering in contrast to neural networks that can learn to represent something automatically.

Three models are applied in the machine learning phase, including the Logistic Regression, Support Vector Machine (SVM), and Random Forest. Logistic Regression is a linear base on which probabilities are estimated of being a member of a certain class whereas SVM maximizes the distance between fake and real news by separating them with the help of a kernel. Ensemble method, namely, the Random Forest, is a non-linear correlating method that builds many decision trees together. The training of all ML models is done by using TF-IDF feature matrices and optimizing the grid search with 5-fold cross-validation to narrow down on hyper parameters like regularization strength, type of kernel and depth of the tree.

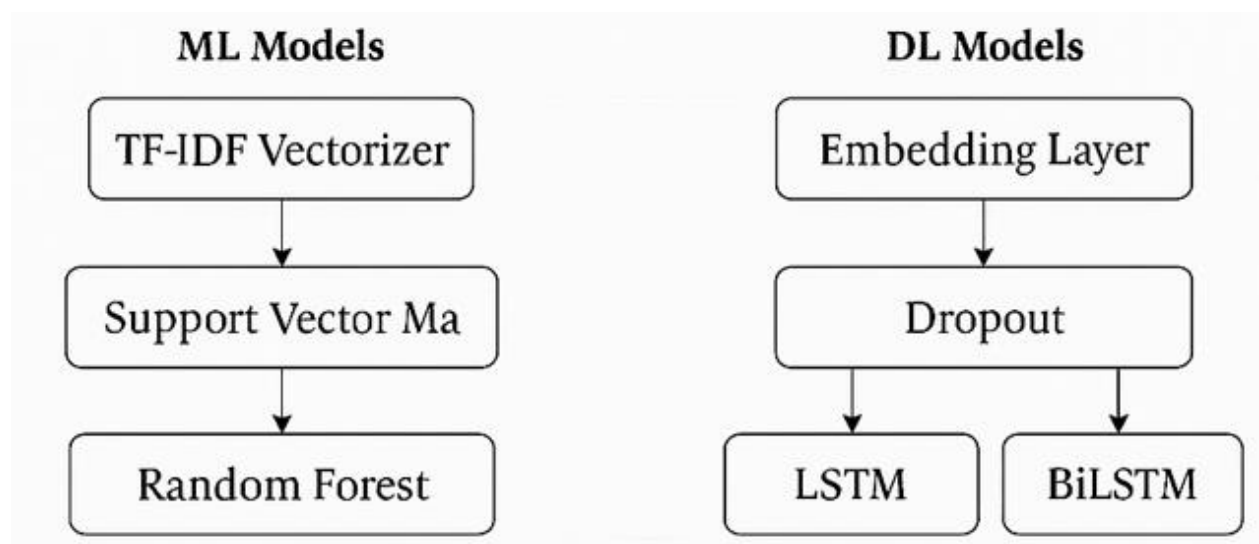


Figure 3.3 Model Architecture

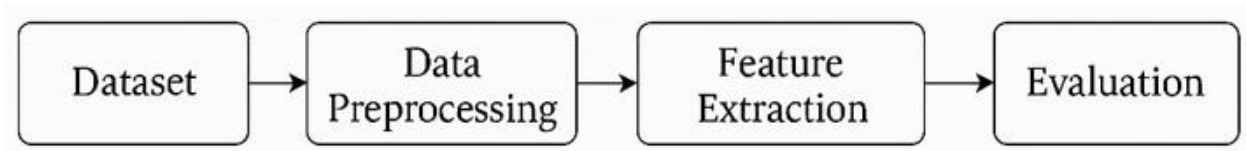


Figure 3.4 Work Flow

Deep learning stage uses successive neural structures that have the ability to learn the connection of context of the word embeddings. The LSTM model is chosen because of its potential to learn long term dependencies in text and Bidirectional LSTM (BiLSTM) improves the potential of LSTM because it allows input sequences to be processed in both forward and backward directions to gain deeper insight into their semantics. Also, a 1D Convolutional Neural Network (CNN) is employed to obtain local textual patterns like n-grams that can be helpful in detecting stylistic and syntactic clues. Both models have layers of embedding word vectors that are pretrained, dropout layers to regularize the model, and the Adam optimizer to optimize gradient descent.

The Multi-epoch training is done with mini-batches of 64 samples per epoch and epochs of 10-15. Models are all tested on the same test set. This two-framework design gives an idea of the comparison between the feature-based ML algorithms and the context-aware neural networks in identifying misleading content. Through a comparison of their merits and demerits, this study determines a holistic concept of the way computational models decipher linguistic deception in the digital media.

3.8. Evaluation Metrics

To determine the effectiveness and reliability of the developed models several evaluation metrics are used, which guarantees not only quantitative rigor but also clarity of interpretation. Accuracy is the main concept that will be evaluated, as it represents the general ratio of accurately defined cases. Nevertheless, due to the fact that accuracy might not be indicative of the performance in terms of

classes, other measures including precision, recall, and F1-score are added. Precision determines the number of predicted fake articles that are actually fake and recall is a metric used to determine how effective the model is at identifying all fake news in the data. The harmonic mean of precision and recall is also called F1-score which gives an unbiased perspective and is useful especially in datasets where there can be an imbalance in class.

Each model is put into a confusion matrix to visualize the true positives, false positives, true negatives as well as false negatives in order to give a detailed picture of the classification behavior. Cross-validation is also used to evaluate machine learning models in order to determine the performance of these models in unseen data. In the case of deep learning models, loss curves of training and validation are tracked on an epoch-by-epoch basis in order to identify overfitting or underfitting behaviors.

ML and DL models are then compared in terms of performance. Deep learning models are anticipated to have higher accuracy and F1-scores given that they have the ability to learn representations. Nevertheless, ML models, including SVM and Random Forest, can prove to be more interpretable and efficient in their computations. The assessment model thus puts emphasis on predictive capability as well as explainability and is in accordance with the aim of the study to determine the best but understandable fake news detection method.

The standard measures that are used to measure the performance of the models are as follows:

- Accuracy (ACC)- percentage of correctly predicted samples. The accuracy is the percentage of the correct classification cases out of all predictions. It is used to determine the effectiveness of the model on the whole. Nonetheless, accuracy can be misleading in situations of class imbalance since it does not differentiate between the types of errors (3.1).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.1)$$

- Precision (P) - percentage of the fake news that is predicted to be fake. Precision is a measure of how often fake news articles have been predicted correctly, compared to the number of articles that are predicted to be fake. High precision signifies that the model generates less false positives, i.e. it can hardly mistake real news as fake news (3.2).

$$Precision = \frac{TP}{TP + FP} \quad (3.2)$$

- Recall (R) - percentage of fake news that was accurately recognized. Recall (or sensitivity) is the fraction of the true fake news items that the model correctly predicted. The high recall means that the model is efficient in identifying the majority of fake news cases, reducing false negatives (3.3).

$$Recall = \frac{TP}{TP + FN} \quad (3.3)$$

- F1-Score (F1)- harmonic mean between precision and recall. The F1-score is the harmonic mean of precision and recall which gives a balanced figure of both measures. It is also applicable when the number of classes is not evenly distributed since it takes into consideration both false positives and false negatives (3.4).

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3.4)$$

- Confusion Matrix - to visualize the truth predictions and false predictions.

A confusion matrix is a table that is used to analyze the effectiveness of a classification model. It presents an overview of the prediction outcomes in terms of actual and predicted class labels.

It is comprised of four major elements:

- True Positives (TP): Fake news that is inaccurately labeled as fake.
- True Negatives (TN): Real news that is rightly deemed as real.
- False Positives (FP): Real news that is mistakenly recognized as fake.
- False Negatives (FN): Fake news that is mistaken as real.

The confusion matrix sheds more light into the nature of the classification errors and assists in comprehending the model performance beyond the aggregate measures. The comparison between ML and DL models is made in terms of these metrics to decide which method has a higher generalization and more robustness.

3.9.Data Preparation and Preprocessing.

Preprocessing and data preparation constitute a crucial component of the suggested method since the raw textual data may have noise that, in turn, may decrease the performance of the machine learning and deep learning algorithms. The preprocessing in this study was done after the title and the text fields were merged into a single content column. The cleaning process was carried out based on a text normalization function and a stopwords elimination phase that is customized. These steps were aimed at ensuring the textual input had more consistency, had less variation that was not useful and retained the most informative linguistic items to classify them. Because the models that are used in this study act directly on the text patterns, the preprocessing was to be done carefully to enhance the quality of features prior to the process of vectorization and training of the models.

The initial preprocessing step involved lowercasing of all the text. This minimizes the repetition of semantically the same words that are spaced out by capitalization. Then, URLs and links to the web were eliminated since they are not always relevant to the linguistic content of the article and can bring noise to the feature space. The numeric values were then deleted then punctuations filtered by regular expressions replacement. Lastly, one space was made out of many spaces and leading or trailing white spaces were divested. The cleaning took place after which the cleaned text was stored in a second column called cleantext. In another step, the NLTK stopwords list was used to remove stopwords in English. This winnowed out very frequent function

words that are not likely to play a strong part in the discrimination of fake and true articles.

After cleaning the text, the dataset was separated into input and target variables with cleantext as the feature input X , and the binary label as the target y . The data was further divided into `traintestsplit` subsets with `traintestsplit` test size 0.2, random state 42, and label variable stratification. This formed 35,918 training samples and 8,980 testing samples, according to the output of the notebook. Stratified splitting is of importance mainly in binary classification as it can maintain the distribution of classes in each subset and it is also fairer in assessing the generalization performance. In general, the preprocessing pipeline converted the raw data set into a standard and model friendly corpus that is capable of supporting traditional vector-based learning as well as deep sequence-based learning.

3.9.1. Bias in Data and Characteristics of the source.

Despite the fact that the dataset that was utilized in this study is reasonably large and fairly balanced at the level of classes, certain significant source and topic-related biases are present. The real-news articles are predominantly gathered as Reuters, whereas the fake-news articles are obtained via sites that can be recognized as misinformation sources. Besides this, both classes are dominated by political and world-news. Consequently, there are some models which can learn source-specific patterns of style, publishing conventions, or topic distributions rather than learn genuinely generalizable cues of misinformation. This problem is significant when looking at very high classification performance, as good performance on this dataset can be partly because of the particularities of the dataset, and not because of more general robustness in the real world.

3.9.2. Feature Representation Techniques.

One of the most significant design features of a fake news detector system is the feature representation since classification models cannot work with raw text. The

proposed framework thus uses a number of feature representation methods, all of which have been chosen to suit the purpose of a specific model. Instead of just using a single textual encoding strategy, the system incorporates sparse word-frequency weighting, dense semantic embeddings, padded token sequences and character level patterns. The selection is a product of the hypothesis that the fake news can be indicated by several layers of information, such as lexical significance, semantic resemblance, local stylistic indications, and sequential relationships among tokens. The framework expands its ability to study various classes of deceptive features by taking a variety of representations.

The family of representation is the first one and is based on TF-IDF that provides information on the relative significance of words and phrases in the corpus. It has proven to be ideally applicable to the traditional classifier since it produces high-dimensional sparse matrices that retain discriminative vocabulary. The second representation is a document embedding that is built on Word2Vec, in which every article is transformed into a dense representation by averaging the individual word representations. This method gives a more semantic perspective to the text as words that appear in the close context are likely to get similar vectors. The third one is to be used with neural network models and transforms text into sequences of integer tokens that are padded to some standard length. Both convolutional and recurrent networks can learn local and sequential features in this format and the token order is preserved unlike in sequence-based input formats.

Besides these general techniques, the framework also uses character-level TF-IDF. This representation comes in particularly handy when detecting fake news since deceptive articles can have stylistic signs, abnormal spelling, sensational formatting, or repeated character patterns, which cannot be fully represented at the word level. These subword messages are represented by character n-grams and supplement word-based features found by the model. The system implemented did not apply

character level TF-IDF in isolation but rather, it was used together with the word level TF-IDF to form a hybrid feature space to be used in SVM classification. Combined, these methods of feature representation offer the proposed system several alternatives of contextualizing textual information that can allow to compare the strategies of fake news detection in a wider and more informative way.

3.9.3. Word-Level TF-IDF Features

As the baseline machine learning models, word-level TF-IDF was employed. TF-IDF or Term Frequency-Inverse Document Frequency is a text vectorization method frequently used where the significance of a word within a document is calculated based on the frequency of that word in the entire corpus. When used in the fake news detection context, TF-IDF is useful as it focuses on meaningful words and short phrases at the expense of excessively frequent words. This would come in handy especially when differentiating between genuine and counterfeit news as fake articles tend to be based on unique lexical selections, emotive formulations or recurrent stylistic figures which can be identified by frequency-based weighting.

On the implementation, `TfidfVectorizer` was set to have the maximum number of features of 50,000 and n-gram of (1, 2). This implies that individual words as well as two-word combinations were represented. The use of bigrams is noteworthy since some of the misleading or persuasive patterns in fake news are not manifested in single words but in short phrases, as well. This could include such combinations as signal claims, urgency, blame, or sensationalism. When the training data was fitted with the vectorizer and both training and test sets transformed, the resulting matrices were of size (35918, 50000) and (8980, 50000) training and testing, respectively. These dimensions indicate that the representation will be large and sparse, as is the case of textual feature spaces in a document classification task.

Multinomial Naive Bayes, Logistic Regression and Support Vector machine models were fed with the TF-IDF representation. Of these, the TF-IDF and SVM combination gave extremely high scores in the notebook output which shows that weighting at lexical and phrase level is very effective on this data. This substantiates the argument that, though deep learning is popular, a well-designed sparse text representation is still a potent tool in fake news detection even in combination with linear classifiers that scale well in high-dimensional space. That is why, word-level TF-IDF is one of the fundamental representation strategies in the suggested scheme.

3.9.4. Word2Vec Document Embeddings

Word2Vec-based semantics representations were also adopted to supplement sparse TF-IDF features because of a study. Word2Vec is a neural embedding algorithm, which learns to map words into dense continuous vectors in a manner that words which are semantically similar are found close to each other in the vector space. Word2Vec concentrates on the contextual similarity and semantic relation as opposed to Term frequency and relevance of the given term in document as is the case with TF-IDF. This enables it to be useful in the fake news detection since certain misleading patterns can not rely solely on the frequency of exact vocabulary, but on the larger semantic sense presented by groups of related words throughout an article.

The training texts were initially tokenized in the implementation by dividing each of the cleaned documents into a list of words. A Word2Vec model was then trained on the tokenized training corpus with a 100-dimensional vector size, 5-window size, minimum frequency of 2 and 4 worker threads. The size of vocabulary was 68,508 words as reported in the notebook output. The research applied average document representations to convert full articles into fixed-length representations. Particularly, the 100-dimensional vectors of all in-vocabulary tokens were retrieved and averaged to create one 100-dimensional vector, in the case with each document. In case no

token of a document was in the vocabulary of Word2Vec, a zero vector of that size was placed instead. This gave training and testing feature matrices of shape (35918, 100) and (8980, 100) respectively.

The results of these Word2Vec document vectors were subsequently trained on Logistic Regression and SVM Classifiers. The findings revealed that the Word2Vec-based models performed poorly, but not as highly as the TF-IDF-based SVM models on this dataset. On a methodological level, though, they remain significant players in the suggested strategy since they are what gives lexical sparse features a semantic backbone and affords the comparison of the latter with dense embedding-based representations. The comparison is used to illustrate the impact that various types of language representations have on the model performance in the fake news detection.

3.9.5. Deep Learning Model sequence encoding.

Classical machine learning algorithms have a different input format than the models of deep learning. Rather than sparse vectors or averaged dense document embeddings, neural networks like CNNs, LSTMs and GRUs tend to take sequences of token indices. This is why the suggested framework comprises a sequence encoding phase, which transforms the text containing only clean words into fixed-length integer sequences to help them be used in training a neural network. The representation is necessary as it is able to maintain token ordering and is able to learn local context, phrase-level dependencies and sequence structure directly out of the data. Sequence modeling can also show patterns that are lost by frequency-based features, which are important in fake news detection when the word sequence matters and the interaction between neighboring phrases as well.

The Keras Tokenizer was employed in the implementation having a vocabulary size of 50,000 words. The tokenizer was applied to the training data and subsequently both the training and testing texts were converted into strings of integer word indices. As articles are of significantly different lengths, the pad-sequences were padded to

an equal length of 300 tokens with `pad_sequences`. This was to make all the input samples have the same dimensionality and to allow them to be run in mini-batches on the neural network architectures. The resulting padded arrays were of kind (35918, 300) (8980, 300) respectively on the training set and the testing set. These arrays were the direct input to CNN, CNN-LSTM, CNN-GRU and CNNBiLSTM models.

In the framework, this sequence representation is beneficial in a number of ways. To begin with, it enables the trainable embedding layers within the deep models to acquire task-specific word-representations. Second, it maintains the sequence of words, allowing convolutional layers to learn local n-gram style patterns and recurrent layers to learn more long-range dependencies between the terms in an article. Third, it offers a standardized input format to various deep architectures, and thus allows them to be compared easily. Consequently, sequence encoding is the interface between raw cleaned text to the neural network elements of the proposed fake news detection system.

3.9.6. Character Level TF-IDF Features.

The TF-IDF at the character level was added to the framework to focus on the sub word context and the stylistic cues of a word which can be missed by word level representations. Such signals in fake news detection may consist of unusual spelling, repetitive punctuation marks before cleaning, use of stylistic exaggeration, morphological fragments or typical phrase endings that occur throughout fake articles. Character n-grams can keep helpful data even when words are changed in slight modification. It is important to note that character-level modeling is particularly helpful when writing style is part of the reason behind the difference between fake and genuine news. Instead of just using full-word boundaries, the

character-level method considers smaller textual units and therefore gives one a finer picture of the linguistic structure.

Within its implementation, character-level representation was done with `TfidfVectorizer` with `analyzer=char`, n-gram range of (3,5), and a maximum of 50000 features. This implies that this system would extract character sequences of three, four and five lengths and weight these sequences based on their TF-IDF significance. The matrices that were obtained had a train size (35918, 50000) and a test size (8980, 50000). These results indicate that the representation space is expressive and large-scale with the model being able to reflect fine-grained textual patterns. The features of character level were not to substitute the word-level TF-IDF but to complement it with the information on the stylistic and subword level as well as lexical content. It is based on this reason that the study stacked up the word-level and character-level matrices horizontally and then trained a linear SVM in the hybrid feature space. This hybrid form was the most accurate in the test results reported in the summary of the experiment, which showed that the combination of semantic information of words and character-based stylistic features was extremely successful in the dataset. In terms of methodology, the given finding justifies the incorporation of character-level TF-IDF as a valuable component of the suggested framework, particularly in cases when it is not the goal of the analysis to comprehend what is being said, but how it is being written.

3.10. Classification Models

The feature extraction process is followed by model training and classification which is the next step of the proposed framework. The system was explicitly made to have various types of classifiers such that performance could be studied in the traditional, neural, and transformer-based methods. This multi-model design is significant as fake news detection is a complicated problem where no single model family can be perceived to be universal best. Linear models can be effective with sparse textual

representations, whereas the neural structure can be used to capture more sequential or semantic information. Transformer models, in their turn, provide contextual language comprehension at the cost of more computational resources. The study provides a firmer empirical foundation of comparing the strengths and limitations of their models because it trains multiple types of models under a single workflow.

Models used were chosen based on the feature representations applied in the system. Multinomial Naive bayes, Logistic regression and SVM were selected due to sparse or dense vector inputs given that they are the standard and reliable text classification baselines. CNN-based as well as hybrid recurrent neural architectures were added since they have the ability to both learn local phrase patterns and longer dependencies directly based on sequences. Moreover, BERT was trained on a smaller set to offer a current contextual base under the computation constraints of the experiment. Collectively, these models represent a very broad methodological spectrum, ranging between interpretable linear classification that is easy to comprehend at the other end to high-capacity neural language models on the other end.

The classification step is also a predictive element and a comparative structure. Both models are inputted in the representation format that is most appropriate to each design and both are tested on the same held-out test data and on the same standard classification measures. This provides uniformity in the comparison and the study to study the reaction of various learning paradigms to the same problem of fake news detection. In this regard, the offered framework is a practical detector and also an experimentation platform of knowing how model design impacts the classification accuracy, precision, recall, and robustness.

3.10.1. Traditional Machine Learning Models.

The first layer of classification of the proposed system was based on traditional machine learning models. The models are highly applicable to text classification as they can work effectively on high-dimensional vectors spaces and in many cases can perform well with no intensive computational requirements of deep learning. In this report, Multinomial Naive Bayes, Logistic Regression and Support Vector machine models were used in the conventional machine learning stage which was principally trained on the word-level TF-IDF representation. Besides that, other algorithms such as Logistic Regression and SVM were as well applied to the Word2Vec document embeddings, to investigate the performance difference between dense semantic vectors and sparse lexical weighting.

Classical probabilistic base Multinomial Naive Bayes is especially used when there is a count-based or weighted by TF-IDF text feature. The choice of Logistic Regression is due to the fact that it offers a powerful linear baseline and does not have significant problems with large sparse matrices, being easy to train. The Support Vector Machine, which is used in the notebook in LinearSVC to classify documents with representation in TF-IDF and in a linear-kernel SVC to classify documents with representation in vectors of Word2Vec was added as SVMs are well known to be able to perform well in document classification applications using high-dimensional data. The models have been especially helpful in detecting fake news since they can take advantage of variations in vocabulary, phrase usage and weighted linguistic patterns between fake and true articles.

The results provided indicate that the conventional models were very competitive particularly when combined with TF-IDF features. The TF-IDF plus the SVM model was able to provide the most accurate classification of the test set as compared to the traditional models based on the Word2Vec. This implies that sparse lexical and phrase based information is highly discriminatory on the existing data. Although

deep learning is becoming more popular, these findings show that the traditional machine learning may still be a highly effective and convenient option in fake news classification, especially when the text is already being carefully cleansed and represented using robust vectorization techniques.

3.10.2. Deep Learning Models

The proposed framework included deep learning models to be able to capture complex linguistic patterns that may not be completely described by linear decision space or human-engineered vector spaces. Neural networks are able to learn internal feature hierarchies on token sequences unlike traditional classifiers, which rely greatly on external extraction of features. This renders them especially well suited to fake news detection, in which meaning can not only be based on the presence of words, but also arrangement of phrases, interaction over context, and compositional design. The four neural models applied in this paper were: CNN with trainable embedding, CNN-LSTM, CNN-GRU and CNN-BiLSTM. The padded sequence representation mentioned above was used in all these models.

CNN model incorporated embedding layer and one dimensional convolutional layer with 128 filters and 5 kernel size, global max pooling, dropout layers and dense classification layers. Local textual patterns similar to informative n-grams had to be found using this architecture. This design was further extended by the hybrid recurrent models which used convolution layer and either LSTM, GRU or Bidirectional LSTM units. The LSTM and GRU versions were to learn sequential dependencies not only within the local neighborhoods but also the BiLSTM variant was to process sequences in both forward and backward directions to enhance contextual knowledge. Binary cross-entropy loss and the Adam optimizer were used to train all deep models. Validation accuracy was also used in early stopping to reduce overfitting and reconstruct the best performing weights.

According to the notebook output the deep learning models performed highly on the classification and the CNN and CNN-BiLSTM models were particularly good. They showed the capability to learn robust representations of the padded text sequences despite the fact that these models took more training time than the traditional baselines. The fact that they are considered in the framework is methodologically significant since they enable the study to compare the shallow and deep learning paradigms directly to each other on the same fake news task, which will add depth to the general assessment of the proposed approach.

3.10.3. Transformer-Based Model

As a supplement to the conventional and neural classifiers, an experiment based on transformers with the usage of BERT was also provided. Transformer models are contrasted with traditional recurrent and convolutional networks in that they are based on contextual self-attention that gives each token in a sequence the ability to attend to each other token. This allows a clearer representation of context and meaning of words especially when the same word can mean different things in the context of the surrounding text. Contextual modeling is useful in the detection of fake news due to the fact that often, language that is used to mislead, often relies on framing, implication and semantic nuance as opposed to specific lexical decisions. That is why, BERT was added to the proposed framework as a latest contextual benchmark.

The model applied in the implementation was bert-base-uncased. Due to the computationally expensive nature of transformer fine-tuning, the experiment was carried out on a subset of the original dataset as opposed to the corpus. The notebook refers to 3,000 training instances and 1,000 testing instances as the reduced sample on which BERT was trained and tested respectively. The BERT tokenizer was used to tokenize the text and truncate or pad to the maximum sequence length of 256

tokens. The Hugging Face ecosystem was used to perform some fine-tuning and the outcome was stored to the experiment summary together with the training time. The given design choice enabled the study to incorporate a baseline based on transformers without going above the limits of the practical computational resources. Even though the BERT experiment had a high test accuracy, it is important to interpret it with caution since it was not trained and tested on the same size of the entire data as the other models. Accordingly, the transformer section shall be explained in the thesis as a contextual benchmark experiment, and not a comparison, directly equivalent with the full-dataset models. Despite this restriction, the BERT component is beneficial since it shows how the modern pretrained language model acts on the problem of classifying fake news and makes the methodological scope of the proposed framework more comprehensive.

3.11. Ensemble and Hybrid Architectures.

The last modeling phase of the proposed system is beyond the single classifiers, which also involves the use of hybrid and ensemble architectures. The principal reason of this design is that various models might reflect complementary characteristics of the same text. To take one example, a TF-IDF-based linear SVM can be much more effective in leveraging patterns of weighted vocabulary, whereas a deep neural network can be more successful in contextual and sequential patterns. The idea of ensemble learning is to bring together these advantages such that the end product or the resultant prediction is stronger than what any individual base model would give. This paper applied three of these strategies: hybrid word-plus-character TF-IDF SVM, a stacking ensemble and a voting-based deep plus machine learning ensemble.

The ensemble stage is significant to the thesis since it shows that the proposed system is not confined to isolated experiments. Rather, it develops into integrated

predictive strategies which are a continuation of the previous models. This also fits well into the larger research scope where the study is seeking to explore the possibility of using different perspectives of the same text to enhance the performance of fake news detection. Practically, the hybrid and ensemble approaches were introduced once the baseline and deep models had already trained, therefore, making use of their outputs or features instead of training the entire system again. This renders this approach computationally feasible, but still methodologically relevant.

According to the findings provided in the notebook, it reveals that the ensemble model and hybrid model were some of the strongest performing models in the whole series of the experiment. Specifically, the word-and-character TF-IDF SVM was the most accurate at the test stage and the stacking ensemble is also performing very well. The findings contribute to the notion that complementary textual indicators and classification views can be helpful in detecting fake news. In this regard, ensemble learning serves as an essential component of the suggested strategy and the ultimate phase of the model refinement in the present study.

3.11.1. Word + Character TF-IDF with SVM

The hybrid word-plus-character TF-IDF SVM classifier is one of the most effective models of the proposed framework. The model was constructed based on the supposition that it is necessary not only to detect fake news using semantic word content, but also to make reference to stylistic and subword-based regularities. Word-level TF-IDF identifies the significance of complete words and bigrams whereas character-level TF-IDF identifies orthographic and stylistic features including snippings, repeated endings, and local textual organization. Through the combination of these two features spaces, the classifier can get to learn what is being

said as well as the way it is being said. This applies particularly to fake news where stylistic sensationalism may be as informative as the content.

During the implementation, sparse horizontal stacking of the word-level TF-IDF matrix with the character-level TF-IDF matrix was used on the SciPy sparse horizontal stacking. The hybrid matrix that resulted thus represented both the lexical and subword information in one high-dimensional space. Then a linear SVM was trained on this combination feature space. This model was an obvious selection since SVMs tend to work especially well in large sparse spaces of vectors, and so can make use of the joint representation. The training and evaluation process was the same as with the other models and this allowed the same level of consistency in the reported results.

As presented in the summary of the experiment, this hybrid model had the best test accuracy compared to all the other approaches considered in the experiment. This finding contributes effectively to the conclusion that the combination of word level meaning and character level style is effective in this fake news dataset. This model is to be brought out in the thesis thus not only as another experiment but also as one of the main contributions of the implementation. It represents a viable and computationally-effective mechanism of boosting classification by not migrating completely to more costly end-to-end transformer designs.

3.11.2. Stacking Ensemble

The stacking ensemble was meant to utilize the prediction of various base models to create a meta-classifier at a higher level. Stacking The concept of stacking is that various models can fail in different ways, and a secondary learner can learn how to integrate their results better than each base model on its own. In this implementation, the stacking approach united three base predictors: TF-IDF + Logistic Regression, TF-IDF + SVM and Word2Vec + Logistic Regression. These models were chosen

due to the fact that they are powerful but contrasting views of learning two of sparse lexical space and one dense semantic embedding space.

The notebook used the trained base models to extract prediction outputs in order to build the meta-features. In the case of the logistic regression models, the probability of the predicted classes was used and in the case of the SVM model the probability of the predicted classes was used. These values were then summed up column wise to a new meta-feature matrix. The training meta-matrix was size (35918,3), and the testing meta-matrix size (8980,3) that each sample was modeled by three higher-level predictive signals. The meta-learner was then a logistic regression model that was trained on the training meta-features and then used to yield the final classification outputs on the test set.

Such an ensemble is methodologically handy in the sense that it demonstrates that the model outputs themselves can serve as features of a second-stage learner. In the thesis, it is to be outlined meticulously as a stacking methodology deployed, and not as an overtly streamlined meta-learning pattern. The notebook shows that the tool worked very well, although the description should not be exaggerated with the statements on the sophisticated optimization. To be more exact, we would say that the model is a stacking-based ensemble whereby the results of base-models are added together and trained through logistic regression as a meta-classifier.

3.11.3. Deep + ML Voting Ensemble

Together with the classic machine learning and deep learning models, an experiment with a transformer who employs BERT (bert-base-uncased) was run as a contextual benchmark. But owing to the computational constraints, the entire dataset was not trained with BERT. Rather it was tested on a smaller sample size of 4,000 records in total, including 3,000 training samples and 1,000 testing samples. Thus, it is important to view the BERT experiment as a limited-scope benchmark as opposed

to a directly equivalent comparison with the full-dataset models that are used in the rest of the study.

The third ensemble technique that the proposed framework presents is the combination of traditional machine learning classifier and a deep learning classifier. This model had been made so that the efficiency and high lexical discrimination of the TF-IDF SVM was combined with the capacity of the CNN-BiLSTM architecture to sequence context. The rationale for this method is that the two models can extract different yet complementary information in the text. Although the SVM is quite attentive to weighted lexical trends throughout the article, the CNN-BiLSTM is capable of acquiring local phrase characteristics and larger-scale sequence connections. When they are combined they can therefore be more robust particularly in situations where one model is unsure but the other one is sure.

During the implementation, TF-IDF SVM and CNN-BiLSTM predictions were generated on the training and testing set. In the case of the neural network, binary class outputs were transformed by using a threshold of 0.5 to transform the probabilities into binary class outputs. This was then concluded with a simple rule of voting whereby the results of the two models was added and a positive final class assigned when the aggregate score went to a minimum of one. This brings the ensemble nearer to soft-majority or inclusive voting system. It is worth noting that no retraining of a different meta-model was necessary hence the time spent by this ensemble in training was counted as none since it only comprised of inference and prediction combination.

The resultant model was very successful in the experiment summary, though not better than the hybrid word-plus-character SVM, and stacking ensemble. Nevertheless, it is still a significant aspect of the given suggestion since it shows how the models of various paradigms can be brought together in a straightforward yet efficient manner. This section can be introduced into the thesis as a proof that

the combination of deep and traditional approaches to learning can represent a realistic extension of the framework and can provide competitive performance at a comparatively low level of complexity in further implementation.

3.12. Model Evaluation Setup

A common set of metrics of classification and a common experimental protocol was used to evaluate the effectiveness of the proposed approach to all models. The assessments were developed on the 80:20 stratified train-test split as above where the models were trained on 35,918 samples and tested on 8,980 unseen samples. This out of sample testing process gives a direct estimate of the generalization of each model outside of the training data. The results were also stored in the experimental setup in a form of a structured list and later arranged them in a comparison table, meaning that all models could be ranked based on their performance in the test.

The evaluation measures of the notebook were training accuracy, test accuracy, precision, recall, F1-score and training time in seconds. Besides this, confusion matrices and comprehensive reports of the classification of each model were also made. Accuracy is the general rate of the correct predictions whereas precision signifies the reliability of the positive predictions. Recall indicates the model posing the positive class well, and F1-score is a harmonic mean of precision and recall. These measures are reasonable when it comes to fake news detection as they can provide a better picture of their quality of classification compared to only accuracy. The amount of time spent on training was also logged in so as to measure computational efficiency, which is of interest in the cases of comparing classical machine learning, deep learning, and transformer-based methods.

According to the summary table reported by the notebook, the performance of some models was very high. The hybrid word-plus-character TF-IDF SVM, TF-IDF SVM and stacking ensemble were among the best performers. Strong results were also

obtained with the deep learning models, and BERT demonstrated high accuracy on its small subset experiment. This evaluation section is to be highlighted in the thesis, not only in the predictive performance but also in the fairness of comparison because all models were tested according to the same basic split and reporting framework. It further notes that the special case of BERT also has its subset-based evaluation, where the result produced by it should not be compared with the models produced on a full dataset.

3.13. Summary

The chapter has introduced the proposed fake news detection framework that is developed in the present study. It starts with the building of a binary labeled dataset between Fake.csv and True.csv files and the development of a single content field comprising the title of articles and the text of their bodies. Upon data set preparation, lowercasing, URLs, digits, punctuations, stopwords are used to normalize the text. Stratified sampling is then used to divide the cleaned corpus into training and testing subsets to provide a fair and reproducible basis of the experiment. These phases provide the basis on which the rest of the system is developed.

The primary strategies of representation in the framework were also described in the chapter. Sparse lexical weighting was done using word-level TF-IDF, dense semantic representations were done using Word2Vec, sequence encoding was done using neural models and capturing subword and stylistic information was done using character-level TF-IDF. A general collection of models such as traditional machine learning classifiers, CNN-based neural architecture and a transformer-based BERT experiment was then matched with these representations. Moreover, hybrid and ensemble methods were presented by the framework including word-plus-character TF-IDF with SVM, a stacking ensemble, and a voting-based combination of deep

and machine-learning predictions. These elements combined make the suggested approach comprehensive and rich in its methodology.

In general, the presented plan proves that the detection of fake news can be enhanced with the help of the layered pipeline that is composed of cautious preprocessing, several types of textual representations, and various classification policies. The framework does not cover a single model family, but rather it is intended to draw parallels, integrate and analyze various approaches on a common platform. This chapter is thus the backbone of the implementation of the thesis. The next chapter is supposed to continue on this basis and present the experimental findings, comment on the performance of the various models in comparison, and explain why one representation and classifier had been successful on the chosen dataset and not the rest.

Chapter 4: Experimental Results and Discussion

4.1. Introduction

In this chapter, the experimental outcomes are displayed in regards to the fake news detection framework suggested in this paper and the relative effectiveness of the models implemented. The experiments were meant to measure the performance of various feature representation techniques and classification algorithms on the identical binary fake news detection problem. The reviewed models consisted of both time-honored machine learning methods, deep learning networks, benchmark transformer, and ensembles. This chapter has not just the goal to report model accuracy, but also to discuss the weaknesses and strengths of each strategy with respect to the data set, representation strategy, and computational cost.

The experimental environment was done in accordance with the methodology in the last chapter. The dataset was built based on Fake.csv and True.csv and cleaned by normalizing and removing stop words and divided into a training and testing dataset with an 80:20 stratified splitting. This gave 35,918 training samples and 8,980 testing samples. Several different forms of representation were then applied, such as word-level TF-IDF, Word2Vec document embeddings, neural model tokenized and padded sequences, and character-level TF-IDF which are used by the hybrid model. These spaces were combined with a set of different classifiers to give a wide comparison between the traditional, neural and ensemble paradigms.

The findings presented in this chapter reveal that there was a robust performance of all the major model families on the given dataset, yet definite performance gaps were still noted. Specifically, the full-dataset models that performed the best in terms of test accuracy were the hybrid Word + Character TF-IDF + SVM, TF-IDF + SVM and the Stacking Ensemble. Deep learning models also yielded very competitive results and the BERT experiment provided high quality results on a smaller subset.

These results indicate that the quality of representation was the key factor in the performance of the models and that highly developed hybrid sparse features were highly effective in this classification problem.

4.2. Experimental Setup

The experiments were done on the ISOT based fake and real news data that was used in the course of this paper. Once the fake and the true article files were merged and the cleaned text field created, the data was separated into a training set of 35,918 instances and testing set of 8,980 instances. This train-test split was stratified in order to maintain the balance of classes between the fake and true articles, thus justifying a more balanced assessment of the performance of generalization. Before feature extraction and model training, the preprocessing pipeline consisted of lowercasing, removal of URLs, removal of numbers, filtering punctuations, normalization of white spaces and removal of stopwords.

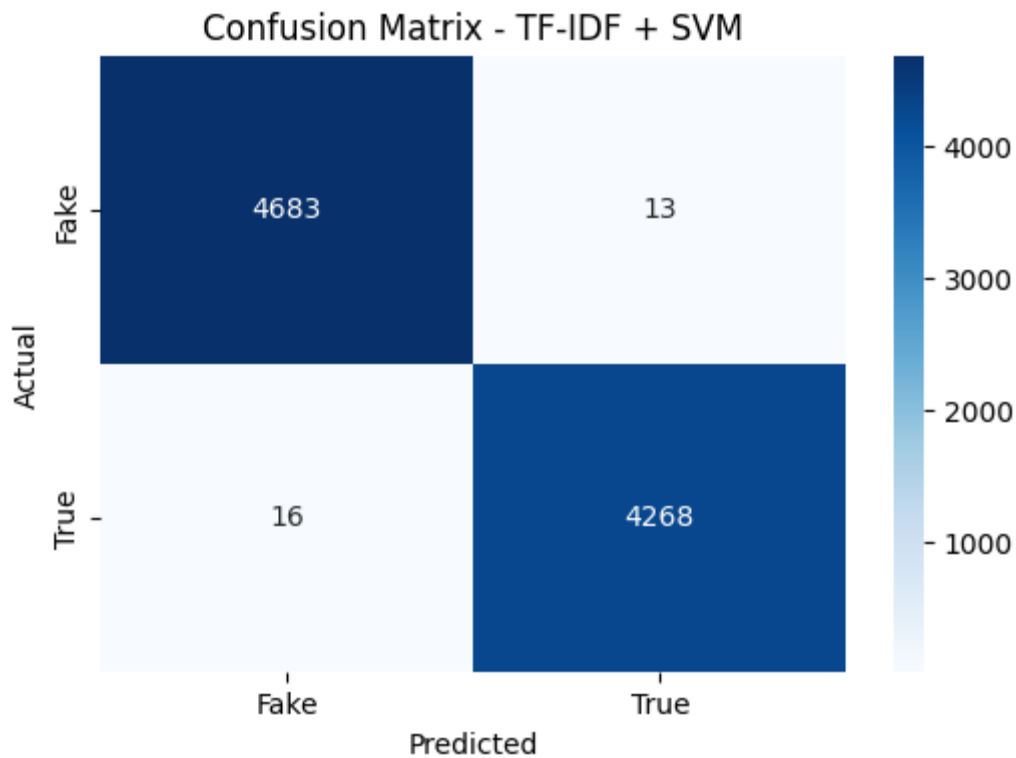


Figure 4.1 Confusion matrix of TF-IDF + SVM

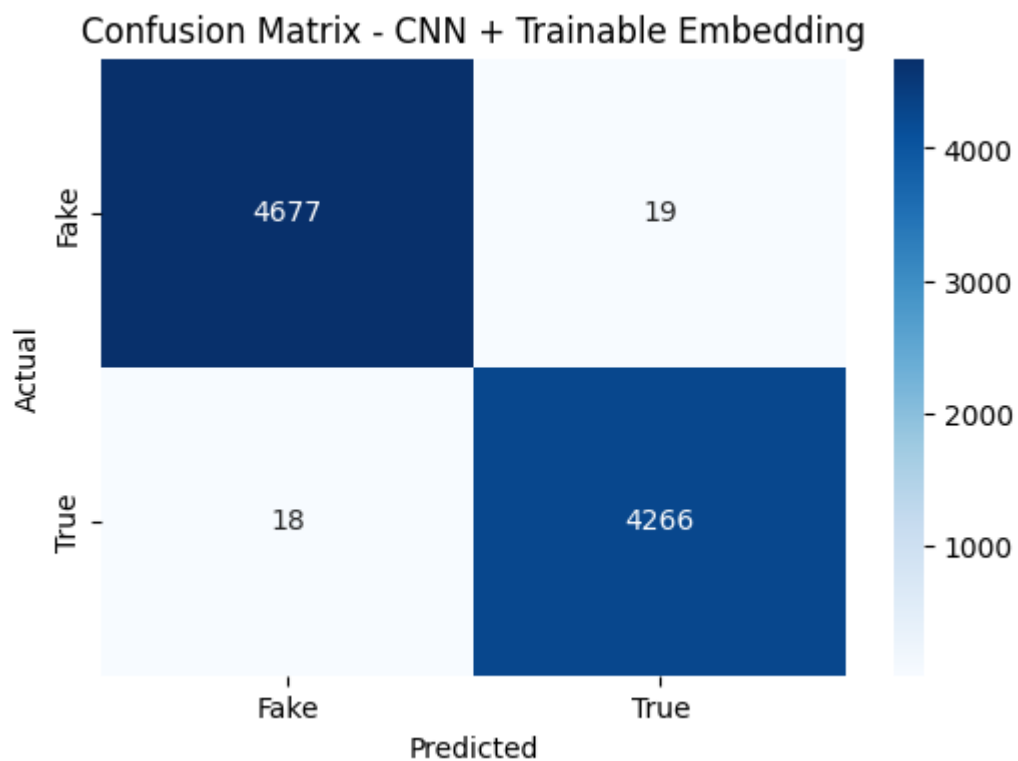


Figure 4.2 Confusion matrix of CNN + Trainable Embedding

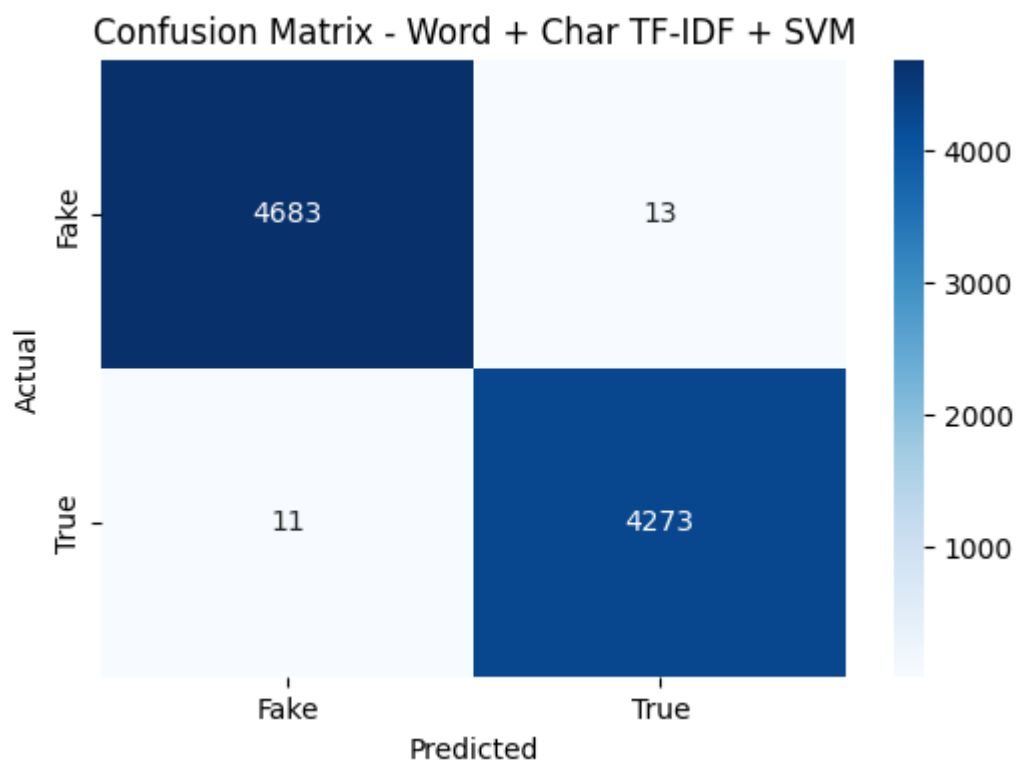


Figure 4.3 Confusion matrix of Word + Char TF-IDF + SVM

There were four major categories of models that were used in the experiments. The former category entailed conventional machine learning models that were prepared on word-level TF-IDF or Word2Vec document vectors. The second one was deep learning structures with padded text sequences, such as CNN, CNN-LSTM, CNN-GRU, and CNN-BiLSTM. The third type was a transformer-based BERT model that was fine-tuned on a smaller portion of the data because of computational limits. The fourth group was hybrid and ensemble methods which comprised Word + Character TF-IDF + SVM, stacking ensemble, and deep plus machine learning voting ensemble. The study was able to compare simple baselines, semantic embeddings, sequence models and mixed techniques within the same framework due to this design.

The notebook measured the performance of the models by training accuracy, test accuracy, precision, recall, F1-score, and training time using the number of seconds. Besides that, there were confusion matrices and complete classification reports. These actions gave a compromise analysis of the quality of prediction and computing speed. Accuracy was helpful in the general comparison whereas precision, recall, and F1-score were helpful in getting more insight regarding the class discrimination performance. Training time was also part of it since it enabled comparison of the cost of computation of the shallow, deep, and transformer-based approaches.

Table 1.1 Experimental Setup Summary

Item	Value
Dataset source	Kaggle fake and real news dataset
Input files	Fake.csv and True.csv
Fake news records	23,481
True news records	21,417
Total dataset size	44,898
Empty content rows	0
Text field used for modeling	Combined title + text as content
Cleaned text field	clean_text

Train-test split	80:20 stratified
Training samples	35,918
Testing samples	8,980
Word-level TF-IDF configuration	max_features=50000, ngram_range=(1,2)
TF-IDF train shape	(35,918, 50,000)
TF-IDF test shape	(8,980, 50,000)
Word2Vec configuration	vector_size=100, window=5, min_count=2
Word2Vec vocabulary size	68,508
Word2Vec train shape	(35,918, 100)
Word2Vec test shape	(8,980, 100)
Sequence tokenizer vocabulary	50,000 words
Maximum sequence length	300
Padded sequence train shape	(35,918, 300)
Padded sequence test shape	(8,980, 300)
Character TF-IDF configuration	analyzer = char, ngram_range=(3,5), max_features=50000
Character TF-IDF train shape	(35,918, 50,000)
Character TF-IDF test shape	(8,980, 50,000)
BERT subset train size	3,000
BERT subset test size	1,000
BERT max sequence length	256
BERT epochs	1

4.3. Overall Model Performance

The general analysis of the models reveals that the applied fake news detection framework was characterized by the high performance in most of the approaches. The most effective model during the experiment was the Word + Character TF-IDF + SVM classifier because it had a training accuracy of 0.999972 and a test accuracy of 0.997327, with a precision of 0.996967, recall of 0.997432 and F1-score of 0.997200. This finding suggests that word and character level information fusion resulted in a very discriminative feature space to detect fake news.

TF-IDF + SVM and the Stacking Ensemble, both of which have the same test-accuracy 0.996659, with precision 0.996730, recall 0.996265 and F1-score 0.996498 were the second-best performing models. The fact that these results resemble each other is indicative that the stacking ensemble did not significantly outperform its best linear TF-IDF baseline, despite the combination of several predictive signals. This finding is important as it demonstrates that even the linear based lexical representation using a linear margin-based classifier was extremely powerful on this dataset.

The neural models yielded the highest test accuracy of 0.996325 followed by CNN + Trainable Embedding, CNN + BiLSTM, CNN + GRU, and CNN + LSTM with test accuracy of 0.992316, 0.988085, and 0.990200 respectively. The TF-IDF + Logistic Regression baseline also held its own at 0.990089, the Word2Vec-based traditional models, Word2Vec + SVM and Word2Vec + Logistic Regression, were lower at 0.976837 and 0.975947 respectively. TF-IDF + MultinomialNB was the poorest model in the comparison table with a test accuracy of 0.959020, though even this base of performance was quite good as far as a binary text classification task was concerned.

Table 4.2 Overall Performance Comparison of All Models

Model	Train Accuracy	Test Accuracy	Precision	Recall	F1-Score	Training Time (s)
Word + Char TF-IDF + SVM	0.999972	0.997327	0.996967	0.997432	0.997200	11.299798
TF-IDF + SVM	0.999944	0.996659	0.996730	0.996265	0.996498	0.568365
Stacking Ensemble	0.999944	0.996659	0.996730	0.996265	0.996498	0.113705
CNN + Trainable Embedding	0.999861	0.996325	0.995108	0.997199	0.996153	28.256719
BERT (subset)	—	0.994000	0.991952	0.995960	0.993952	163.296224
Deep + ML Ensemble (SVM + CNN-BiLSTM)	0.999109	0.992539	0.984824	0.999767	0.992239	0.000000
CNN + BiLSTM	0.998998	0.992316	0.987283	0.996732	0.991985	79.804168

CNN + GRU	0.998775	0.990200	0.987907	0.991597	0.989748	43.227098
TF-IDF + Logistic Regression	0.994348	0.990089	0.987451	0.991830	0.989635	1.115155
CNN + LSTM	0.998497	0.988085	0.988310	0.986695	0.987501	52.212567
Word2Vec + SVM	0.976084	0.976837	0.972415	0.979225	0.975808	18.658500
Word2Vec + Logistic Regression	0.975695	0.975947	0.971051	0.978758	0.974890	0.363632
TF-IDF + MultinomialNB	0.960131	0.959020	0.959624	0.954248	0.956929	0.108068

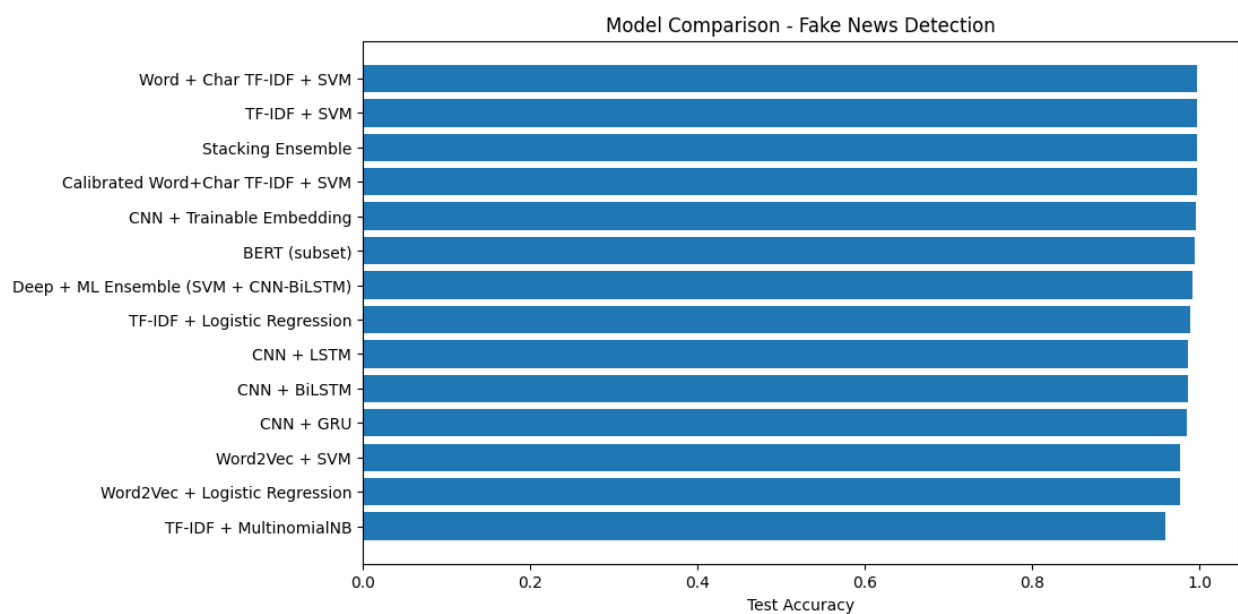


Figure 4.4 Model Comparison - Fake News Detection

4.4. Traditional machine learning models Performance.

The classical machine learning models provided a high level of baseline in fake news classification. The highest achievement was made by the TF-IDF + SVM as it achieved a test accuracy of 0.996659 with almost perfect precisions, recall, and F1-score measures. According to the classification report presented in the notebook, this model was almost as effective on fake classes as it was on true classes, which is evidence that the decision boundary of this model generalized to both labels. This

validates the appropriateness of linear SVM to high-dimensional text data that is sparse.

The TF-IDF + Logistic Regression model also had a good performance with the test accuracy of 0.990089, the precision of 0.987451, the recall of 0.991830, and the F1-score of 0.989635. Even though it showed slightly worse results than the SVM model, it nonetheless indicated that linear classifiers could obtain superior performance in association with a properly set TF-IDF representation. Comparatively, TF-IDF + MultinomialNB is associated with lesser and yet good performance with the test accuracy of 0.959020 and F1-score 0.956929. The difference states that the simple probabilistic assumptions in Naive Bayes did not work as well as the margin-based or discriminative linear approaches on this data.

The machine learning models on the basis of Word2Vec showed significantly worse performance than those on the basis of TF-IDF. Word2Vec + Logistic Regression was found to have a test accuracy of 0.975947 and Word2Vec + SVM had 0.976837. Though these are still respectable results, it demonstrates that averaged document embeddings could not maintain the same level of discriminative lexical detail as TF-IDF does the same with respect to this task. A possible cause is that averaging word vectors can diffuse certain phrase-level signals and local textual appearances that are very informative in detecting fake news. Consequently, the experiments indicate that the use of sparse lexical weighting was more successful in comparison to dense averaged embeddings in the chosen corpus.

Table 4.3 Top-5 Model Ranking

Rank	Model	Test Accuracy	Precision	Recall	F1- Score	Training Time (s)	Note
1	Word + Char TF-IDF + SVM	0.997327	0.996967	0.997432	0.997200	11.299798	Best full-dataset model
2	TF-IDF + SVM	0.996659	0.996730	0.996265	0.996498	0.568365	Best traditional baseline

3	Stacking Ensemble	0.996659	0.996730	0.996265	0.996498	0.113705	Matched TF-IDF + SVM
4	CNN + Trainable Embedding	0.996325	0.995108	0.997199	0.996153	28.256719	Best deep learning model
5	BERT (subset)	0.994000	0.991952	0.995960	0.993952	163.296224	Evaluated on subset only

4.5. Deep Learning Model Attainments.

The deep learning models also performed highly in classification, which reveals that neural sequence modeling is suitable in detecting fake news. The CNN + Trainable Embedding model yielded the best neural outcome with test accuracy of 0.996325, precision 0.995108, recall 0.997199 and F1-score 0.996153. It is an especially good finding in that it demonstrates that a rather simple convolutional network, with learned embeddings and appropriate regularization, can be almost as effective as the best traditional and ensemble models on this data.

The repeated hybrid models were also competitive. The test accuracy, precision and recall values of CNN + BiLSTM were 0.992316, 0.987283 and 0.996732 respectively with F1-score standing at 0.991985. CNN + GRU achieved the accuracy and F1-score of 0.990200 and 0.989748, respectively, whereas CNN + LSTM achieved 0.988085 and 0.987501. These findings suggest that by combining convolutional feature extraction with recurrent sequence modeling one can obtain meaningful contextual patterns in the text, but in this case they were not able to outperform the best sparse-feature models.

An interesting finding is that the most complex neural architecture did not give the best deep learning result. In its turn, the less complicated CNN model was more effective than CNN-LSTM, CNN-GRU, and CNN-BiLSTM with respect to test accuracy. This indicates that the local phrase-level data represented by convolutional filters was already very informative to the classification problem and that more

elaborate recurrent processing might not always be translated into a higher level of generalization on this dataset. It also shows that a higher architectural complexity will be able to increase the training cost without necessarily providing better performance.

4.6. Transformer-Based Result

The experiment with transformer based using BERT (bert-base-uncased) had a good test accuracy of 0.994000 in one of the reported runs and 0.993000 in another reported run. Nevertheless, BERT, in contrast to the other models, was trained on a smaller sample and tested on that sample as well. Particularly, the notebook demonstrates that the BERT experiment worked with 3,000 training instances and 1,000 test instances, the longest sequence length (256) and a single fine-tuning epoch. Due to such variations, its performance can only be seen as a contextual standard instead of a direct comparison to the entire dataset experiments.

BERT model time also took the longest time to train among the reported methods with an average of 163.30 seconds to train in a single time. This training is very expensive, even compared to the strongest traditional machine learning baselines and is as well expensive as some deep learning models. Practically, it implies that despite BERT providing powerful contextual modeling, it is computationally costlier and not as appealing in cases where lightweight implementation is needed or experimentation is necessary.

The BERT (bert-base-uncased)-based transformer-based experiment has shown high performance and proven the possibilities of contextual pretrained language models in detecting fake news. This outcome should however be approached with caution since BERT was trained and tested on a small sample of just 4,000 instances in total that includes 3,000 instances to train the model, and 1,000 instances to test it. In contrast to the other models, the evaluation of the BERT experiment was performed

under limited computational settings and on a single fine-tuning epoch instead of the entire dataset split. Thus its reported accuracy can be considered a subset-based benchmark result and not an entirely comparable result with the full-dataset models.

Table 4.5 Class-wise Metrics for Selected Best Models

Model	Class	Precision	Recall	F1-Score	Support
TF-IDF + SVM	Fake	1.00	1.00	1.00	4696
	True	1.00	1.00	1.00	4284
CNN + Trainable Embedding	Fake	1.00	1.00	1.00	4696
	True	1.00	1.00	1.00	4284
Word + Char TF-IDF + SVM	Fake	1.00	1.00	1.00	4696
	True	1.00	1.00	1.00	4284
Stacking Ensemble	Fake	1.00	1.00	1.00	4696
	True	1.00	1.00	1.00	4284
Deep + ML Ensemble (SVM + CNN-BiLSTM)	Fake	1.00	0.99	0.99	4696
	True	0.98	1.00	0.99	4284

** Note: Values are rounded off to two decimal points.*

4.7. Hybrid and Ensemble Models Performance.

The most sophisticated section of the experiment system was the hybrid and ensemble models that produced some of the best outcomes. The best all-purpose model was Word + Character TF-IDF + SVM model. It had the highest test accuracy of any full-dataset model at 0.997327 indicating the usefulness of lexical material combined with subword stylistic patterns. The value of the respective precision, recall, and F1-score was also highest in the comparison table, which proved that this is not a single gain in one indicator alone.

Stacking Ensemble also performed quite the same as the TF-IDF + SVM model with a test accuracy of 0.996659 and F1-score of 0.996498. In this ensemble, TF-IDF + Logistic Regression, TF-IDF + SVM and Word2Vec + Logistic Regression output were taken as meta-features to which a Logistic Regression meta-learner was trained. The stacking model also validated the hypothesis that a strong classifier can

be obtained by combining several prediction signals; although it did not surpass the hybrid word-plus-character model.

Deep + ML Ensemble (SVM + CNN-BiLSTM) provided a test accuracy of 0.992539, the precision of 0.984824, the recall of 0.999767, and F1-score of 0.992239. Although it is less than leading hybrid and stacking models, this is a good performance. Surprisingly, this ensemble yielded very high recall indicating that it worked especially well with positive cases. Because this procedure took a sparse-feature linear classifier with a deep sequence model, the results demonstrate that cross-paradigm integration would be useful, even though the improvement was not significant enough to outperform the highest sparse hybrid method in this study.

4.8. Discussion of Key Findings

Among the most significant results of the given work, one must mention that the most successful model was not the most computationally intensive one, but a hybrid sparse-feature model; the word-level and character-level TF-IDF with SVM. It means that, in the chosen dataset, lexical and stylistic indicators play a critical role in the fake news detection and can be effectively represented in high-dimensional sparsity. The outcome is significant as it refutes the belief that more advanced neural or transformer designs should necessarily be more successful than classical ones.

The second important conclusion is that TF-IDF-based models were always able to beat the Word2Vec-based baselines. This indicates that maintaining a precise lexical significance and brief expressions were more productive than averaged semantic imbeddings. The manifestations of sensationalism, and certainty, emotional framing, or textual formulae may be a powerful indicator in fake news detection. These cues usually get better saved in TF-IDF rather than in document-scale averaged embeddings, which is apt to blank out local discriminative characteristics.

Thirdly, another notable observation is that deep learning models were not dominant but very competitive. Specifically, the CNN model was extremely successful and almost as strong as the most powerful classical methods. Nevertheless, it did not lose to recurrent hybrids like CNN-LSTM or CNN-GRU. This suggests that the local patterns of phrases could be rather imperative than the visualization of the data at a long distance. It further suggests that the selection of the model must be based not only on the theoretical modeling ability, but also on the peculiarities of the language of the problem and on the trade-off between accuracy and computational cost.

4.8.1. The Interpretation of Near-Perfect Results.

The next significant fact is that certain class-wise metrics are showing 1.00 in the reported results. These values are to be taken with caution. The first one is that they are whole to two decimal places, which can visually indicate ideal classification during cases where the true values are slightly less than 1.00. Second, the test accuracies of the respective models are not really high as overall test accuracies of the respective models are below 1.00. Rather, such near-perfect values are probably indicative of the high separability of the chosen dataset and the existence of powerful dataset-specific cues. As a result, one should not consider such results as proof of perfect generalization in the real world.

4.8.2. Limitations of the Experimental Results.

In spite of the fact that the reported results are very strong, they may be regarded with certain caution. First, the experiments were done on a single main dataset, which is based on Fake.csv and True.csv. Although this is a large and useful dataset to benchmark with, it might not be representative of the variety of fake news that people are exposed to across domains, platforms, languages, or time. Thus, performance on this dataset is not necessarily the same because of greater cross-domain generalization.

Second, the result of BERT was delivered on a smaller subset and, thus, it is not comparatively stable with the models of the entire dataset. This discrepancy in the size of training and evaluation scale must be highlighted in the discussion of the thesis. Third, the stacking model was applied based on the outputs of previously trained models and a cross-validated stacked generalization model based on a logistic regression meta-learner, but without being described as a highly tuned or cross-validated stacking model. To be more precise academically, one can refer to it as a stacking-based ensemble, as opposed to a completely optimized meta-learning system.

Lastly, the vastly high accuracies which have been reported in a number of models could be because the chosen dataset is relatively learnable. This does not make the experiments any less valuable, but it implies that in the future, cross-dataset validation, robustness testing, and more difficult real-world scenarios should be incorporated into the work. These extensions would be used to establish whether the same ranking of the methods are also the same in more varied and noisier misinformation setting.

4.8.3. Dataset Bias and Generalization Limitations

The dataset bias is one of the primary limitation of the current study. Though the dataset is big and the proportion of classes is balanced, it might not be very representative of the entire range of misinformation one would be exposed to in a real-life setting. The real-news articles are mostly gathered using one professional source, and the fake-news articles are gathered using websites that are already related to misinformation. Such arrangement can enable models to acquire source-related writing style, pattern of formatting and subject inclination rather than identify misinformation in a domain-neutral way. Thus, the good results presented in this thesis can be, in part, due to the peculiarities of the dataset, and the findings should be cautiously applied.

4.8.4. Potential Extrapolation and Interpretation of Very High Accuracy.

The second limitation is the potential of overfitting or partial over-specialization to the data. Certain models scored very high in the training and testing models and some of the class-wise measures were found to be 1.00 upon rounding. Although these findings suggest that the dataset is very learnable, they can also imply that the models learned patterns that are particularly useful to this benchmark, but not necessarily also useful to other areas, time, or news sources. The reported performance can therefore be interpreted as good benchmark level performance and not as evidence of universal strength.

4.8.5. Limited Explainability Analysis

The other shortcoming of the study is that the explainability component was restricted. The model behavior interpretation was limited primarily to SHAP-based feature-importance analysis and no visual SHAP figures or local-example examples or the larger explainable AI methods using LIME, Integrated Gradients, or attention visualization were provided. This means that the research yields just the initial stage of explainability and not a fully explainable AI paradigm. One should admit this limitation, particularly since transparency and trust are valued in fake news detectors.

4.9. Summary of the Study

This study aimed at exploring the potential of using machine learning and deep learning algorithms to automatically identify fake news pieces using text-based information. The main issue that was discussed within the thesis was that it is very challenging to identify genuine and fake news in a scalable and computationally sound way. The previous sections of the thesis highlighted that the problem of fake news is not only a linguistic one, but also a wider concern of the lack of trust in the population, democracies, and integrity of digital information. Corresponding to the

abovementioned objectives, this study aimed to create a structured and data-driven framework of text classification that will be able to compare a number of detection strategies in the same consistent experimental environment.

The ISOT-style fake and real news dataset was used in the study to fulfill this objective; this dataset consisted of 23,481 fake news articles and 21,417 true news articles each containing 44,898 articles in total. To prepare the corpus to the model development, a preprocessing pipeline was used to normalize the text, delete URLs, digits, punctuation marks and stopwords. Stratified sampling was then used to divide the data into 35,918 training samples and 8,980 testing samples. Then, several types of textual representations were produced to ensure that one could compare classical machine learning, semantic embeddings, deep sequence models, and hybrid classifiers.

The paper then applied and compared various models, among which are TF-IDF with Multi-nomial Naive Bayes, Logistic regression and SVM; Word2vec with Logistic regression and SVM; CNN based deep learning models; a small subset of BERT benchmark and ensemble models such as stacking and deep plus machine learning voting. The experiments were measured based on conventional classification measures such as (accuracy, precision, recall, F1-score, confusion matrices, and training time). In this manner, the study achieved its comparative goal and offered a strong empirical foundation on the type of modeling decisions that were the most successful on the chosen data set.

4.10. Chapter Summary

This chapter introduced and reported the experiment findings of the proposed fake news detection framework. The results indicated that all the large model families performed well, with the best accuracy being recorded with the Word + Character TF-IDF + SVM. Standard TF-IDF-driven machine learning models, in particular, SVM, were highly effective, whereas deep learning models also did similarly well

with CNN providing the best neural outcome. The use of transformer-based contextual modeling was confirmed to be useful in the BERT subset experiment, which was more expensive to compute and tested on a smaller subset.

It was also shown in the chapter that hybrid and ensemble approaches can enhance strength and, occasionally, performance. Specifically, combining word-level and character-level features resulted in the best result using the entire dataset. Generally, these findings indicate that text representation is also a very effective feature in recognizing fake news and that even the classical linear models can be used to compete effectively with more sophisticated neural methods when the strong features are used. The findings in these results are found to support the effectiveness of the proposed framework and give a sound foundation of the ultimate conclusions of the thesis.

Chapter 5: Conclusion and Future Work

5.1. Conclusion

To sum up, this thesis has demonstrated machine learning as a highly viable tool of identifying fake news by analyzing texts. The study has shown that the classification of automated misinformation can be conducted with extremely high performance by building a single experimental framework and comparing several feature representations and families of models on the chosen dataset. The most effective model that was tested was the hybrid Word + Character TF-IDF + SVM model as it showed the best result on the full dataset and showed the significance of the combination of lexical and stylistic cues.

The general implication of the research is that the effectiveness of a fake news detector does not only rely on the selection of a contemporary model, but also on the selection of a suitable representation strategy and fitting the model to the dataset structure. Classical and hybrid approaches were at least equally significant in this thesis as deep and transformer-based ones. It is an essential finding since it supports the worth of principled experimental comparison instead of thinking that more elaborate architectures are inherently better.

On the whole, the research makes a significant contribution to the benchmark, has solid underpinnings of practical implementation, and has a clear direction of future improvement. Although critical issues in interpretability, robustness, multilingual generalization, and multimodal modeling still exist, the work on this topic has laid a substantial foundation in the future research work on fake news detection and the subsequent creation of reliable artificial intelligence systems that can facilitate information integrity in online space.

The most significant finding of the current work is that the most popular model was not the most sophisticated model, but a well-planned hybrid sparse-feature classifier.

The Word + Character TF-IDF + SVM model had highest full-dataset accuracy of 0.997327, 0.996967 precision, 0.997432 recall and 0.997200 F1-score. This finding suggests that the detection of fake news in the chosen corpus is strongly reliant on both the lexical patterns and the stylistic or subword indicators, and that a combination of these two sets of features forms an exceedingly productive representation space. In terms of methodology, it is one of the most powerful findings of the thesis since it shows that hybrid sparse-feature engineering is still very powerful in classifying fake news.

The second significant finding is that the traditional machine learning models based on TF-IDF always showed better results compared to Word2Vec-based baselines. The TF-IDF + SVM model was the most accurate test with a score of 0.996659 and Word2Vec + SVM and Word2Vec + Logistic Regression model had a weight of 0.976837 and 0.975947 respectively. This indicates that informative word and short phrase weighting were more helpful on this dataset as compared to averaged semantic embeddings. It also suggests that numerous classification indicators in fabricated news are reproduced by means of discriminative vocabulary and phraseology alternatives instead of semantic smoothing of documents.

Third, it was concluded that deep learning models were quite competitive but did not always exceed the strongest classical methods. The strongest deep model CNN + Trainable Embedding achieved 0.996325 test accuracy which is exceptionally strong and comparable to the best overall baselines. In the meantime, more advanced recurrent hybrids including CNN + LSTM, CNN + GRU, and CNN + BiLSTM were also effective, although they could not be better than the leading sparse-feature models. This implies that local phrase patterns on the chosen dataset were particularly informative, and they can be learnt with a high efficiency by either convolutional neural networks or linear classifiers using TF-IDF features. It also

demonstrates that the increased complexity of architectural design can not always ensure a high level of performance.

The BERT experiment using the transformer showed once again that contextual pretrained language models can perform well, as the test accuracy was 0.994000 in the given subset experiment. The fact that it was trained on a smaller sample of 3,000 training items and 1,000 testing items, however, can be understood as the indication that it is more of a benchmark than a true equivalent comparison with the full-dataset models. Nevertheless, its inclusion made the thesis stronger as it demonstrated that the modern approaches toward the transformer may be incorporated into the framework and also demonstrated the trade-off between the quality of the contextual modeling and the cost of the computation.

Lastly, the results of the ensemble demonstrated that the synthesis of model outputs can be useful as well. Stacking Ensemble was equal to the TF-IDF + SVM baseline at 0.996659 test accuracy, and the Deep + ML Ensemble (SVM + CNN-BiLSTM) at 0.992539. The implication of these findings is that ensemble learning is capable of giving strong performance, and in this study instance it was failing to be more robust compared to the strongest hybrid sparse-feature model. In general, the overall conclusion of the thesis is that fake news detection can be conducted to a great extent by utilizing a properly prepared text classification pipeline, and hybrid sparse representations with powerful linear classifiers can be viewed as some of the most feasible and effective solutions to this dataset.

5.2. Future Work

Some significant suggestions may be offered in terms of work in the future. The most direct and immediate extension is that of cross-dataset and cross-domain evaluation. Future research ought to apply the identical framework to extra data set including but not restricted to LIAR, FakeNewsNet, or domain-specific

misinformation set. This would assist in establishing whether the ranking in this thesis is not dependent on the ISOT-style dataset and whether the hybrid sparse-feature model can be added to harder and more varied misinformation environments. This would directly deal with the limitations of domain adaptation and benchmark of the literature review.

The second is a second significant direction, which is multilingual and low-resource fake news detection. In your thesis, the literature review has already highlighted that there is a low level of transferability between English-focused models and that there is the necessity of linguistically inclusive systems. Future research would thus be able to consider multilingual versions of BERT, Arabic-specific embedding, cross-lingual transfer models on non-English fake news datasets. This would not only project the existing text-based benchmark into a more universal relevant systematic but also address the one of the main research gaps that was previously identified in the thesis.

Explainable fake news detection is explainable in a third direction. Trust and accountability are the key issues under consideration in automated misinformation systems, which is why the future models are bound to include interpretability tools like SHAP, LIME, integrated gradients, or attention-based explanation techniques. These tools would allow determining the words, phrases, or patterns that make the largest contribution to a fake or a true classification. It would be better to add an explainability layer to make the system more useful to the journalists, moderators, and researchers who require a label, but the evidence that supported the prediction. This would be more consistent with the contribution to be expected as was stated in the thesis on the matter of transparent and explainable systems.

The fourth direction of the future is robustness and adversarial testing. According to the literature review, the number of the fake content creators who take advantage of the model flaws is growing, and AI-generated misinformation becomes more

advanced. Future research must thus compare the existing framework to paraphrased or adversarially perturbed or synthetically generated samples of fake news. Techniques that could be explored to enhance resilience could be adversarial training, data augmentation, and strong representations learning. This orientation is necessary when the system is to be transported out of the artificial benchmark environment into reality.

There is a fifth direction that is multimodal and social-context modeling. Fake news detectors in the future should utilize text alongside metadata including credibility of publisher, history of sources, patterns of user interaction, or embedded images and videos. In this aspect, graph neural networks, co-attention models, and multimodal transformers may be considered. As the existing thesis also took into consideration only textual characteristics, by making it multimodal and social-context information a framework, one could potentially have a more realistic picture of how misinformation is prone to dispersal on the internet. This would bring the work closer to the current research trends covered in the literature review also.

Lastly, future research may centre on the practical applications and lightweight inferences. As the given research revealed that hybrid sparse-feature models were very effective, it might be feasible to create a prototype based on the web or API and utilize the most effective classifier to screen news material in real-time. This type of system may be incorporated into editorial processes, learning media literacy tools, or pipelines of platform moderation. This would push the work off the comparative experimentation to the practical impact which is already one of the given motivations of the thesis.

The research on future work should consider the shortcomings of the current study by testing the proposed models on a variety of data sets across domains, languages, and time. The cross-dataset validation would assist in finding out whether the good performance achieved in this thesis is because of the true generalization capacity or

because of biases in the dataset. Furthermore, visual SHAP summaries and local explanation techniques like LIME, as well as attention-based interpretation of transformer-based models, should be considered as a more powerful way of explaining the future research. Such enhancements would not only enhance the accuracy of the system, but it would also be more transparent and acceptable to implement in practice.

References

1. Ingryd V. S. T. Pereira, George D. C. Cavalcanti, Rafael M. O. Cruz, Multi-view autoencoders for Fake News Detection, Preprint · April 2025, research gate DOI: 10.48550/arXiv.2504.08102
2. Q. Ying, X. Hu, Y. Zhou, Z. Qian, D. Zeng, and S. Ge, “Bootstrapping multi-view representations for fake news detection,” in AAAI Conference on Artificial Intelligence, vol. 37, no. 4, 2023, pp. 5384–5392.
3. Min E, Rong Y, Bian Y, Xu T, Zhao P, Huang J, Ananiadou S (2022) Divide-and-conquer: post-user interaction network for fake news detection on social media. In: Proceedings of the ACM Web Conference 2022, pp. 1148–1158
4. Bazmi, M. Asadpour, and A. Shakery, “Multi-view co-attention network for fake news detection by modeling topic-specific user and news source credibility,” Information Processing & Management, vol. 60, no. 1, p. 103146, 2023.
5. Zhang Z, Lv Q, Jia X, Yun W, Miao G, Mao Z, Wu G (2024) Gbca: graph convolution network and bert combined with co-attention for fake news detection. Pattern Recogn Lett 180:26–32
6. Alkudah NM, Idris NB, Abushariah MAM. 2025. Fake news detection in Arabic media: comparative analysis of machine learning and deep learning algorithms using the Arabic Fake News Dataset. PeerJ Comput. Sci. 11:e3272 DOI 10.7717/peerjcs.3272.
7. Kesarwani, S. S. Chauhan and A. R. Nair, "Fake News Detection on Social Media using K- Nearest Neighbor Classifier," 2020 International Conference on Advances in Computing and Communication Engineering (ICACCE), 2020, pp. 1-4,10.1109/ICACCE49060.2020.9154997
8. Domenico GD, Sit J, Ishizaka A, Nunan D. 2021. Fake news, social media and marketing: a systematic review. Journal of Business Research 124(2):329–341 DOI 10.1016/j.jbusres.2020.11.037.
9. Lu, Y., & Yao, N. (2025). *A fake news detection model using the integration of multimodal attention mechanism and residual convolutional network*. *Scientific Reports*, 15, 20544. <https://doi.org/10.1038/s41598-025-05702-w>

10. Al-Obaidi, S. A., & Çağlıkantar, T. (2024). *Automated fake news detection system*. *Iraqi Journal for Computer Science and Mathematics*, 5(4), 12–26.
<https://doi.org/10.52866/2788-7421.1200>
11. Bisailon, C. (2019). *Fake and real news dataset*. Kaggle.
<https://www.kaggle.com/datasets/clmentbisailon/fake-and-real-news-dataset>

Appendix I : Code

```
from google.colab import files

files.upload()

!mkdir -p ~/.kaggle

!mv kaggle.json ~/.kaggle/

!chmod 600 ~/.kaggle/kaggle.json

# Kaggle

!pip install -q kaggle


# Data processing

!pip install -q pandas numpy nltk tqdm


# Machine Learning

!pip install -q scikit-learn scipy


# Word Embeddings

!pip install -q gensim


# Deep Learning

!pip install -q tensorflow keras
```

```
# Transformers / BERT
```

```
!pip install -q transformers datasets evaluate accelerate
```

```
# Visualization
```

```
!pip install -q matplotlib seaborn plotly
```

```
# Extra utilities
```

```
!pip install -q xgboost lightgbm
```

```
import tensorflow as tf
```

```
print("TensorFlow version:", tf.__version__)
```

```
print("GPU available:", tf.config.list_physical_devices('GPU'))
```

```
!kaggle datasets download -d clmentbisailon/fake-and-real-news-dataset
```

```
!unzip -o fake-and-real-news-dataset.zip
```

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import accuracy_score, classification_report,  
confusion_matrix
```

```
import nltk

nltk.download('stopwords')


print("Libraries loaded successfully")

fake = pd.read_csv("Fake.csv")
true = pd.read_csv("True.csv")


print("Fake shape:", fake.shape)
print("True shape:", true.shape)


print("\nFake columns:", fake.columns.tolist())
print("True columns:", true.columns.tolist())

fake["label"] = 0
true["label"] = 1


data = pd.concat([fake, true], axis=0)


data = data.sample(frac=1, random_state=42).reset_index(drop=True)


print("Dataset shape:", data.shape)
```

```

data.head()

data["content"] = (data["title"].fillna("") + " " + data["text"].fillna("")).str.strip()


print("Rows:", len(data))

print("Empty content:", (data["content"].str.len() == 0).sum())


data[["content", "label"]].head()

import os


folders = ["data", "models", "results", "plots"]


for folder in folders:

    os.makedirs(folder, exist_ok=True)


print("Folders created successfully")

data.to_csv("data/fake_news_dataset.csv", index=False)


print("Dataset saved successfully")

import re


def clean_text(text):

```

```

text = str(text).lower()

text = re.sub(r"http\S+|www\S+", "", text)

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

text = re.sub(r"^\w\s", " ", text)

text = re.sub(r"\s+", " ", text).strip()

return text

```

```

data["clean_text"] = data["content"].apply(clean_text)

```

```

print("Cleaning done")

data[["clean_text", "label"]].head()

from nltk.corpus import stopwords

```

```

stop_words = set(stopwords.words("english"))

```

```

def remove_stopwords(text):

    words = text.split()

    words = [w for w in words if w not in stop_words]

    return " ".join(words)

```

```

data["clean_text"] = data["clean_text"].apply(remove_stopwords)

```

```

print("Stopwords removed")

data[["clean_text", "label"]].head()

X = data["clean_text"]
y = data["label"]


X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

print("Train size:", X_train.shape)
print("Test size:", X_test.shape)

from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer = TfidfVectorizer(
    max_features=50000,
    ngram_range=(1,2)
)

```

```
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
```

```
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

```
print("TF-IDF Train Shape:", X_train_tfidf.shape)
```

```
print("TF-IDF Test Shape:", X_test_tfidf.shape)
```

```
import os
```

```
import time
```

```
import json
```

```
import numpy as np
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
from sklearn.metrics import (
```

```
    accuracy_score,
```

```
    precision_score,
```

```
    recall_score,
```

```
    f1_score,
```

```
    confusion_matrix,
```

```
    classification_report
```

)

```
results = []
```

```
def evaluate_and_store(model_name, y_train, train_pred, y_test, test_pred,  
train_time):
```

```
    train_acc = accuracy_score(y_train, train_pred)
```

```
    test_acc = accuracy_score(y_test, test_pred)
```

```
    precision = precision_score(y_test, test_pred)
```

```
    recall = recall_score(y_test, test_pred)
```

```
    f1 = f1_score(y_test, test_pred)
```

```
    results.append({
```

```
        "Model": model_name,
```

```
        "Train Accuracy": train_acc,
```

```
        "Test Accuracy": test_acc,
```

```
        "Precision": precision,
```

```
        "Recall": recall,
```

```
        "F1-Score": f1,
```

```
        "Training Time (s)": train_time
```

```
    })
```

```

# Confusion Matrix

cm = confusion_matrix(y_test, test_pred)

plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["Fake", "True"],
            yticklabels=["Fake", "True"])

plt.title(f"Confusion Matrix - {model_name}")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()


# Classification report

print(f"\nClassification Report - {model_name}\n")

print(classification_report(y_test, test_pred, target_names=["Fake", "True"]))


print(f"{model_name} saved to results list")


print("Evaluation tools ready")

from sklearn.naive_bayes import MultinomialNB

start = time.time()

```

```

model_nb = MultinomialNB()
model_nb.fit(X_train_tfidf, y_train)

train_pred = model_nb.predict(X_train_tfidf)
test_pred = model_nb.predict(X_test_tfidf)

end = time.time()
train_time = end - start

evaluate_and_store(
    model_name="TF-IDF + MultinomialNB",
    y_train=y_train,
    train_pred=train_pred,
    y_test=y_test,
    test_pred=test_pred,
    train_time=train_time
)

from sklearn.linear_model import LogisticRegression

start = time.time()

```

```
model_lr = LogisticRegression(max_iter=1000)
```

```
model_lr.fit(X_train_tfidf, y_train)
```

```
train_pred = model_lr.predict(X_train_tfidf)
```

```
test_pred = model_lr.predict(X_test_tfidf)
```

```
end = time.time()
```

```
train_time = end - start
```

```
evaluate_and_store(
```

```
    model_name="TF-IDF + Logistic Regression",
```

```
    y_train=y_train,
```

```
    train_pred=train_pred,
```

```
    y_test=y_test,
```

```
    test_pred=test_pred,
```

```
    train_time=train_time
```

```
)
```

```
from sklearn.svm import LinearSVC
```

```
start = time.time()
```

```

model_svm = LinearSVC()

model_svm.fit(X_train_tfidf, y_train)

train_pred = model_svm.predict(X_train_tfidf)
test_pred = model_svm.predict(X_test_tfidf)

end = time.time()

train_time = end - start

evaluate_and_store(
    model_name="TF-IDF + SVM",
    y_train=y_train,
    train_pred=train_pred,
    y_test=y_test,
    test_pred=test_pred,
    train_time=train_time
)

print("Train Accuracy:", (train_pred == y_train).mean())
print("Test Accuracy:", (test_pred == y_test).mean())

```

```
X_train_tokens = [text.split() for text in X_train]
```

```
X_test_tokens = [text.split() for text in X_test]
```

```
print("Example tokens:")
```

```
print(X_train_tokens[0][:20])
```

```
from gensim.models import Word2Vec
```

```
w2v_model = Word2Vec(
```

```
    sentences=X_train_tokens,
```

```
    vector_size=100,
```

```
    window=5,
```

```
    min_count=2,
```

```
    workers=4
```

```
)
```

```
print("Word2Vec trained successfully")
```

```
print("Vocabulary size:", len(w2v_model.wv))
```

```
import numpy as np
```

```
def document_vector(tokens, model):
```

```
    vectors = [model.wv[word] for word in tokens if word in model.wv]
```

```

if len(vectors) == 0:

    return np.zeros(model.vector_size)


return np.mean(vectors, axis=0)


X_train_w2v = np.array([document_vector(tokens, w2v_model) for tokens in
X_train_tokens])

X_test_w2v = np.array([document_vector(tokens, w2v_model) for tokens in
X_test_tokens])


print("Train shape:", X_train_w2v.shape)

print("Test shape:", X_test_w2v.shape)

from sklearn.linear_model import LogisticRegression


start = time.time()


model_lr_w2v = LogisticRegression(max_iter=1000)

model_lr_w2v.fit(X_train_w2v, y_train)


train_pred = model_lr_w2v.predict(X_train_w2v)

test_pred = model_lr_w2v.predict(X_test_w2v)

```

```

end = time.time()

train_time = end - start

evaluate_and_store(
    model_name="Word2Vec + Logistic Regression",
    y_train=y_train,
    train_pred=train_pred,
    y_test=y_test,
    test_pred=test_pred,
    train_time=train_time
)

from sklearn.svm import SVC

start = time.time()

model_svm_w2v = SVC(kernel="linear")

model_svm_w2v.fit(X_train_w2v, y_train)

train_pred = model_svm_w2v.predict(X_train_w2v)

```

```
test_pred = model_svm_w2v.predict(X_test_w2v)
```

```
end = time.time()
```

```
train_time = end - start
```

```
evaluate_and_store(
```

```
    model_name="Word2Vec + SVM",
```

```
    y_train=y_train,
```

```
    train_pred=train_pred,
```

```
    y_test=y_test,
```

```
    test_pred=test_pred,
```

```
    train_time=train_time
```

```
)
```

```
from tensorflow.keras.preprocessing.text import Tokenizer
```

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
max_words = 50000
```

```
max_length = 300
```

```
tokenizer = Tokenizer(num_words=max_words)
```

```
tokenizer.fit_on_texts(X_train)
```

```
X_train_seq = tokenizer.texts_to_sequences(X_train)
```

```
X_test_seq = tokenizer.texts_to_sequences(X_test)
```

```
X_train_pad = pad_sequences(X_train_seq, maxlen=max_length)
```

```
X_test_pad = pad_sequences(X_test_seq, maxlen=max_length)
```

```
print("Train shape:", X_train_pad.shape)
```

```
print("Test shape:", X_test_pad.shape)
```

```
import tensorflow as tf
```

```
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D,  
Dense, Dropout
```

```
from tensorflow.keras.callbacks import EarlyStopping
```

```
start = time.time()
```

```
model_cnn = Sequential([
```

```
    Embedding(input_dim=max_words, output_dim=100,  
    input_length=max_length),
```

```
    Conv1D(filters=128, kernel_size=5, activation="relu"),
```

```
GlobalMaxPooling1D(),  
Dropout(0.3),  
Dense(64, activation="relu"),  
Dropout(0.3),  
Dense(1, activation="sigmoid")  
)
```

```
model_cnn.compile(  
    loss="binary_crossentropy",  
    optimizer="adam",  
    metrics=["accuracy"]  
)
```

```
early_stop = EarlyStopping(  
    monitor="val_accuracy",  
    patience=2,  
    restore_best_weights=True  
)
```

```
history = model_cnn.fit(  
    X_train_pad, y_train,
```

```

validation_split=0.1,

epochs=5,

batch_size=128,

callbacks=[early_stop],

verbose=1

)

train_pred = (model_cnn.predict(X_train_pad) > 0.5).astype(int).flatten()
test_pred = (model_cnn.predict(X_test_pad) > 0.5).astype(int).flatten()

end = time.time()

train_time = end - start

evaluate_and_store(

    model_name="CNN + Trainable Embedding",

    y_train=y_train,

    train_pred=train_pred,

    y_test=y_test,

    test_pred=test_pred,

    train_time=train_time

)

```

```
from tensorflow.keras.layers import LSTM
```

```
start = time.time()
```

```
model_cnn_lstm = Sequential([  
    Embedding(input_dim=max_words, output_dim=100,  
input_length=max_length),  
    Conv1D(128, 5, activation="relu"),  
    LSTM(64),  
    Dropout(0.3),  
    Dense(64, activation="relu"),  
    Dropout(0.3),  
    Dense(1, activation="sigmoid")  
])
```

```
model_cnn_lstm.compile(  
    loss="binary_crossentropy",  
    optimizer="adam",  
    metrics=["accuracy"]  
)
```

```
history = model_cnn_lstm.fit(
```

```

X_train_pad, y_train,
validation_split=0.1,
epochs=5,
batch_size=128,
callbacks=[early_stop],
verbose=1
)

train_pred = (model_cnn_lstm.predict(X_train_pad) > 0.5).astype(int).flatten()
test_pred = (model_cnn_lstm.predict(X_test_pad) > 0.5).astype(int).flatten()

end = time.time()
train_time = end - start

evaluate_and_store(
    model_name="CNN + LSTM",
    y_train=y_train,
    train_pred=train_pred,
    y_test=y_test,
    test_pred=test_pred,
    train_time=train_time
)

```

```

)

from google.colab import drive

drive.mount('/content/drive')

from tensorflow.keras.layers import GRU

start = time.time()

model_cnn_gru = Sequential([

    Embedding(input_dim=max_words, output_dim=100,
input_length=max_length),

    Conv1D(128, 5, activation="relu"),

    GRU(64),

    Dropout(0.3),

    Dense(64, activation="relu"),

    Dropout(0.3),

    Dense(1, activation="sigmoid")

])

model_cnn_gru.compile(

    loss="binary_crossentropy",

    optimizer="adam",

    metrics=["accuracy"]

```

)

```
history = model_cnn_gru.fit(
```

```
    X_train_pad, y_train,
```

```
    validation_split=0.1,
```

```
    epochs=5,
```

```
    batch_size=128,
```

```
    callbacks=[early_stop],
```

```
    verbose=1
```

)

```
train_pred = (model_cnn_gru.predict(X_train_pad) > 0.5).astype(int).flatten()
```

```
test_pred = (model_cnn_gru.predict(X_test_pad) > 0.5).astype(int).flatten()
```

```
end = time.time()
```

```
train_time = end - start
```

```
evaluate_and_store(
```

```
    model_name="CNN + GRU",
```

```
    y_train=y_train,
```

```
    train_pred=train_pred,
```

```

y_test=y_test,

test_pred=test_pred,

train_time=train_time

)

from tensorflow.keras.layers import Bidirectional, LSTM

start = time.time()

model_cnn_bilstm = Sequential([

    Embedding(input_dim=max_words, output_dim=100,
input_length=max_length),

    Conv1D(128, 5, activation="relu"),

    Bidirectional(LSTM(64)),

    Dropout(0.3),

    Dense(64, activation="relu"),

    Dropout(0.3),

    Dense(1, activation="sigmoid")

])

model_cnn_bilstm.compile(

    loss="binary_crossentropy",

    optimizer="adam",

```

```

        metrics=["accuracy"]
    )

history = model_cnn_bilstm.fit(
    X_train_pad, y_train,
    validation_split=0.1,
    epochs=5,
    batch_size=128,
    callbacks=[early_stop],
    verbose=1
)

train_pred = (model_cnn_bilstm.predict(X_train_pad) > 0.5).astype(int).flatten()
test_pred = (model_cnn_bilstm.predict(X_test_pad) > 0.5).astype(int).flatten()

end = time.time()
train_time = end - start

evaluate_and_store(
    model_name="CNN + BiLSTM",
    y_train=y_train,

```

```

train_pred=train_pred,
y_test=y_test,
test_pred=test_pred,
train_time=train_time
)

!pip install transformers --upgrade

from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch

print("Torch available:", torch.cuda.is_available())
print("BERT PyTorch libraries loaded successfully")

train_sample_size = 3000
test_sample_size = 1000

X_train_bert = X_train.iloc[:train_sample_size].tolist()
y_train_bert = y_train.iloc[:train_sample_size].tolist()

X_test_bert = X_test.iloc[:test_sample_size].tolist()
y_test_bert = y_test.iloc[:test_sample_size].tolist()

print("Train sample:", len(X_train_bert))

```

```

print("Test sample:", len(X_test_bert))

tokenizer_bert = AutoTokenizer.from_pretrained("bert-base-uncased")

train_encodings = tokenizer_bert(
    X_train_bert,
    truncation=True,
    padding=True,
    max_length=256
)

test_encodings = tokenizer_bert(
    X_test_bert,
    truncation=True,
    padding=True,
    max_length=256
)

print("Tokenization completed")

print("Train samples:", len(train_encodings["input_ids"]))

print("Test samples:", len(test_encodings["input_ids"]))

import torch

```

```

class FakeNewsDataset(torch.utils.data.Dataset):

    def __init__(self, encodings, labels):

        self.encodings = encodings

        self.labels = labels


    def __getitem__(self, idx):

        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}

        item["labels"] = torch.tensor(self.labels[idx])

        return item


    def __len__(self):

        return len(self.labels)


train_dataset = FakeNewsDataset(train_encodings, y_train_bert)

test_dataset = FakeNewsDataset(test_encodings, y_test_bert)


print("Datasets ready")

print("Train size:", len(train_dataset))

print("Test size:", len(test_dataset))

import numpy as np

```

```

import time

from sklearn.metrics import accuracy_score

from transformers import AutoModelForSequenceClassification,
TrainingArguments, Trainer

model_bert = AutoModelForSequenceClassification.from_pretrained(
    "bert-base-uncased",
    num_labels=2
)

training_args = TrainingArguments(
    output_dir="./bert_results",
    num_train_epochs=1,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=16,
    eval_strategy="no",
    save_strategy="no",
    logging_steps=100,
    report_to="none"
)

trainer = Trainer(

```

```

    model=model_bert,

    args=training_args,

    train_dataset=train_dataset,

)

start = time.time()

trainer.train()

# prediction instead of evaluate
pred_output = trainer.predict(test_dataset)

bert_preds = pred_output.predictions.argmax(axis=1)
bert_acc = accuracy_score(y_test_bert, bert_preds)

end = time.time()

train_time = end - start

print("BERT Test Accuracy:", bert_acc)

print("Training Time:", train_time)

```

```

# save result

results.append({

    "Model": "BERT (subset)",

    "Train Accuracy": None,

    "Test Accuracy": bert_acc,

    "Precision": None,

    "Recall": None,

    "F1-Score": None,

    "Training Time (s)": train_time

})

print("BERT result saved successfully")

from sklearn.feature_extraction.text import TfidfVectorizer

char_vectorizer = TfidfVectorizer(

    analyzer="char",

    ngram_range=(3, 5),

    max_features=50000

)

X_train_char = char_vectorizer.fit_transform(X_train)

```

```

X_test_char = char_vectorizer.transform(X_test)

print("Char TF-IDF Train Shape:", X_train_char.shape)
print("Char TF-IDF Test Shape:", X_test_char.shape)

from scipy.sparse import hstack

from sklearn.svm import LinearSVC

X_train_hybrid = hstack([X_train_tfidf, X_train_char])
X_test_hybrid = hstack([X_test_tfidf, X_test_char])

start = time.time()

model_hybrid_svm = LinearSVC()
model_hybrid_svm.fit(X_train_hybrid, y_train)

train_pred = model_hybrid_svm.predict(X_train_hybrid)
test_pred = model_hybrid_svm.predict(X_test_hybrid)

end = time.time()

train_time = end - start

```

```

evaluate_and_store(
    model_name="Word + Char TF-IDF + SVM",
    y_train=y_train,
    train_pred=train_pred,
    y_test=y_test,
    test_pred=test_pred,
    train_time=train_time
)

import numpy as np

lr_probs = model_lr.predict_proba(X_test_tfidf)[:,1]
svm_preds = model_svm.predict(X_test_tfidf)

lr_w2v_probs = model_lr_w2v.predict_proba(X_test_w2v)[:,1]

X_stack = np.column_stack([
    lr_probs,
    svm_preds,
    lr_w2v_probs
])

```

```

print("Stack features shape:", X_stack.shape)

# Train stack features

lr_train_probs = model_lr.predict_proba(X_train_tfidf)[:, 1]

svm_train_preds = model_svm.predict(X_train_tfidf)

lr_w2v_train_probs = model_lr_w2v.predict_proba(X_train_w2v)[:, 1]


X_stack_train = np.column_stack([

    lr_train_probs,

    svm_train_preds,

    lr_w2v_train_probs

])


# Test stack features

lr_test_probs = model_lr.predict_proba(X_test_tfidf)[:, 1]

svm_test_preds = model_svm.predict(X_test_tfidf)

lr_w2v_test_probs = model_lr_w2v.predict_proba(X_test_w2v)[:, 1]


X_stack_test = np.column_stack([

    lr_test_probs,

    svm_test_preds,

    lr_w2v_test_probs

```

)

```
print("Train stack shape:", X_stack_train.shape)
```

```
print("Test stack shape:", X_stack_test.shape)
```

```
from sklearn.linear_model import LogisticRegression
```

```
start = time.time()
```

```
meta_model = LogisticRegression()
```

```
meta_model.fit(X_stack_train, y_train)
```

```
train_pred = meta_model.predict(X_stack_train)
```

```
test_pred = meta_model.predict(X_stack_test)
```

```
end = time.time()
```

```
train_time = end - start
```

```
evaluate_and_store(
```

```
    model_name="Stacking Ensemble",
```

```
    y_train=y_train,
```

```
    train_pred=train_pred,
```

```

    y_test=y_test,
    test_pred=test_pred,
    train_time=train_time
)

print("model_svm exists:", 'model_svm' in globals())
print("model_cnn_bilstm exists:", 'model_cnn_bilstm' in globals())

import numpy as np

# ML prediction
svm_preds = model_svm.predict(X_test_tfidf)

# Deep prediction
bilstm_probs = model_cnn_bilstm.predict(X_test_pad)
bilstm_preds = (bilstm_probs > 0.5).astype(int).flatten()

# Soft/majority voting
ensemble_preds = ((svm_preds + bilstm_preds) >= 1).astype(int)

# train predictions for saving
svm_train_preds = model_svm.predict(X_train_tfidf)
bilstm_train_probs = model_cnn_bilstm.predict(X_train_pad)

```

```

bilstm_train_preds = (bilstm_train_probs > 0.5).astype(int).flatten()
ensemble_train_preds = ((svm_train_preds + bilstm_train_preds) >= 1).astype(int)

train_time = 0.0 # no retraining, just ensemble inference

evaluate_and_store(
    model_name="Deep + ML Ensemble (SVM + CNN-BiLSTM)",
    y_train=y_train,
    train_pred=ensemble_train_preds,
    y_test=y_test,
    test_pred=ensemble_preds,
    train_time=train_time
)

from sklearn.calibration import CalibratedClassifierCV
from sklearn.svm import LinearSVC

start = time.time()

base_svm = LinearSVC()

calibrated_svm = CalibratedClassifierCV(

```

```

    base_svm,
    method="sigmoid",
    cv=3
)

calibrated_svm.fit(X_train_hybrid, y_train)

train_pred = calibrated_svm.predict(X_train_hybrid)
test_pred = calibrated_svm.predict(X_test_hybrid)

end = time.time()

train_time = end - start

evaluate_and_store(
    model_name="Calibrated Word+Char TF-IDF + SVM",
    y_train=y_train,
    train_pred=train_pred,
    y_test=y_test,
    test_pred=test_pred,
    train_time=train_time
)

```

```
results_df = pd.DataFrame(results)

results_df = results_df.sort_values(
    by="Test Accuracy",
    ascending=False
)

results_df

import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))

plt.barh(
    results_df["Model"],
    results_df["Test Accuracy"]
)

plt.xlabel("Test Accuracy")

plt.title("Model Comparison - Fake News Detection")

plt.gca().invert_yaxis()
```

```

plt.show()

results_df.to_csv("model_results.csv", index=False)

from sklearn.metrics import confusion_matrix

import matplotlib.pyplot as plt

import seaborn as sns

best_train_pred = calibrated_svm.predict(X_train_hybrid)
best_test_pred = calibrated_svm.predict(X_test_hybrid)

cm = confusion_matrix(y_test, best_test_pred)

plt.figure(figsize=(6,4))

sns.heatmap(

    cm,

    annot=True,

    fmt="d",

    cmap="Blues",

    xticklabels=["Fake", "True"],

    yticklabels=["Fake", "True"]

)

```

```

plt.title("Confusion Matrix - Calibrated Word+Char TF-IDF + SVM")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

from sklearn.metrics import roc_curve, auc

probs = calibrated_svm.predict_proba(X_test_hybrid)[: ,1]

fpr, tpr, _ = roc_curve(y_test, probs)

roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6,4))

plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.4f}")

plt.plot([0,1],[0,1], '--')

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title("ROC Curve - Calibrated Word+Char TF-IDF + SVM")

plt.legend()

plt.show()

```

```

feature_names_word = tfidf_vectorizer.get_feature_names_out()
feature_names_char = char_vectorizer.get_feature_names_out()

feature_names = np.concatenate([feature_names_word, feature_names_char])

svm_model = calibrated_svm.calibrated_classifiers_[0].estimator

svm_weights = svm_model.coef_[0]

top_indices = np.argsort(svm_weights)[-20:]

top_words = feature_names[top_indices]

print("Top words indicating TRUE news:\n")

for w in top_words[::-1]:
    print(w)

bottom_indices = np.argsort(svm_weights)[:20]

fake_words = feature_names[bottom_indices]

```

```

print("Top words indicating FAKE news:\n")

for w in fake_words:
    print(w)

from sklearn.model_selection import cross_val_score
from sklearn.svm import LinearSVC
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
import numpy as np

X_all = data["clean_text"]
y_all = data["label"]

# Word TF-IDF
word_vectorizer = TfidfVectorizer(
    analyzer="word",
    max_features=50000
)

```

```

# Char TF-IDF

char_vectorizer = TfidfVectorizer(
    analyzer="char",
    ngram_range=(3,5),
    max_features=50000
)

features = ColumnTransformer([
    ("word_tfidf", word_vectorizer, "clean_text"),
    ("char_tfidf", char_vectorizer, "clean_text")
])

model = Pipeline([
    ("features", features),
    ("svm", LinearSVC())
])

# 10-fold cross validation

scores = cross_val_score(
    model,
    data,

```

```
data["label"],  
cv=10,  
scoring="accuracy",  
n_jobs=-1  
)  
  
print("Fold Accuracies:", scores)  
print("Mean Accuracy:", np.mean(scores))  
print("Std Deviation:", np.std(scores))  
print(df.columns)
```