**ARTIFICIAL INTELLIGENCE (AI) APPROACH FOR DETECTING FAKE AND COUNTERFEIT NEWS IN SOCIAL MEDIA**

**By**

**DAVIDSON OGARAKU ONYINYE**

**219074045**

**In partial fulfilment of the award OF MASTERS of Science in Computer Science,**

**Submitted to the Computer ScienceS Department, University of Lagos, Akoka, YABA, LAGOS, NIGERIA.**

**November, 2023**

# ABSTRACT

The pervasive spread of fake news in online social media has emerged as a critical threat to societal integrity and democratic processes. To address this pressing issue, this project harnesses the power of supervised AI algorithms aimed at detecting and mitigating the impact of fake news. Algorithms such as Passive Aggressive Classifier, Perceptron, and Decision Stump undergo meticulous refinement for text classification tasks, leveraging 29 models trained on diverse social media datasets. Data preprocessing involves rigorous cleansing and feature vector generation using TF-IDF and Count Vectorizers. The models' efficacy in discerning genuine news from falsified or exaggerated content is evaluated using metrics like Accuracy, Precision, Recall, and more. Results showcase diverse performance across datasets: Stacked Generalization achieved 0.9805 accuracy in Dataset 1, Bernoulli RBM scored 0.9991 in Dataset 2, Stacked Generalization reached 0.9755 in Dataset 3 and with a value of 0.6123, Stacked Generalization for dataset 4 was the algorithm that also achieved the maximum accuracy value. This research illuminates strategies for combating fake news, offering potential solutions to safeguard information integrity and democratic discourse, thus carrying profound implications for academia and real-world applications.

# CHAPTER ONE

# INTRODUCTION

## 1.0 Background to the study:

Online social media platforms have become increasingly influential in a time when digital communication and information exchange are advancing quickly. These platforms have emerged as the main avenues for the broadcast of news, presenting both previously unheard-of potential and difficulties. The frequency and effects of fake news, which is purposefully false or misleading material presented as news, are among the most urgent problems (Liu & Wu, 2018).

This study proposes a two-phased detection methodology designed to identify instances of fake news within the realm of social media. The recommended framework integrates supervised artificial intelligence algorithms with text analysis techniques, forming an innovative approach. In the initial stage of the project, text mining methodologies are employed to analyze a dataset comprising internet news. The primary objective of these text analysis techniques is to extract structured information from unstructured news stories.

In the subsequent phase, a total of twenty-nine algorithms for supervised artificial intelligence are enlisted to perform the task of classifying fake news from legitimate news. These algorithms encompass a diverse range, including “lbfgs” Logistic Regression, “liblinear’’ Logistic Regression, “newton-cg” Logistic Regression, ‘sag’ Logistic Regression, Random Forest, Perceptron, Ridge Classifier, CatBoost, Nearest Centroid, Stochastic Gradient Decent(SGD), SVC(Kernel=“linear”, C=0.025), SVC(gama=2, C=1), LinearSVC, ZeroR, Decision Tree, Passive Aggressive, Extra Tree, Random Patches, Voting, Stacked Generalization, Multi-layer perceptron(MLP), Bernoulli RBM, AdaBoost, Gradient Boosting, Ordinal Learning Model, XGBoost, Decision Stump, Complement Naïve Bayes, Multinomial Naïve Bayes.

To ensure robustness, these supervised algorithms are subjected to rigorous training and testing using five distinct datasets. Through this comprehensive approach, the study aims to effectively distinguish between bogus news and authentic news within the dynamic and complex landscape of social media.

The emergence of false news has wide-ranging effects. The public's confidence in media organizations and democratic institutions is also weakened, in addition to the credibility of information sources. Misinformation can spread like wildfire over social media networks, confusing the audience and possibly influencing their opinions and decisions (Monsees, 2021). Misinformation is frequently fueled by clickbait, political goals, or sensationalism. Researchers and professionals have turned to cutting-edge technology, such as supervised artificial intelligence algorithms, to assist in identifying false information within the dynamic and linked landscape of online social media in order to address this issue.

## 1.1 Statement of the Problem

A key societal concern is the spread of bogus news on social media websites. It gets harder and harder to tell whether news content is legitimate as consumers consume and distribute it without constantly checking its correctness. The issue is made worse by the inability of conventional fact-checking techniques to keep up with the rate at which information is shared on these platforms. Detecting fake news in online social media is a serious issue, and the goal of this study is to create a reliable solution by utilizing the power of supervised artificial intelligence algorithms.

## 1.2 Aim and Objectives

### 1.2.1 Aim

The project aims at adopting artificial intelligence (AI) approach for detecting fake and counterfeit news in social media.

### 1.2.2 Objectives

The pursuing of the following particular goals will help to accomplish the overall goal:

* To review and evaluate recent studies on supervised artificial intelligence algorithms and fake news detection techniques.
* To use text cleaning, tokenization, and feature extraction techniques to preprocess the gathered dataset.
* To use text categorization tasks necessary for false news identification using supervised artificial intelligence algorithms.
* To employ suitable assessment measures to methodically train and assess the resulting algorithms' performance.
* To evaluate the effectiveness of several supervised algorithms and determine the advantages and disadvantages of each for the detection of false news.
* To present findings and engage in a comprehensive discussion on the effectiveness of the proposed approach.

## 1.3 Methodology

The methodology initiates by defining the parameters for identifying fake and counterfeit news within social media, setting the scope and criteria for classification. Diverse datasets from social platforms undergo preprocessing, employing techniques like TF-IDF and Count Vectorization for feature extraction. Tailored AI algorithms, including Decision Trees and Random Patches and more, are selected and fine-tuned for text classification tasks, followed by validation on separate datasets to ensure robustness and prevent overfitting. Model evaluation using standard metrics such as Accuracy, Precision, and Recall allows comparison of performance, potentially leading to the implementation of ensemble techniques for improved detection capabilities.

## 1.4 Limitations of the Study

It's critical to understand key restrictions that could affect the scope and generalizability of the findings, even though this research intends to significantly advance the field of Identifying bogus news on social media sites. Some possible restrictions include:

1. Data Bias: The dataset used for training and evaluation may have biases that are typical of online social media platforms, which could have an impact on the effectiveness and generalizability of the model.
2. Dynamic Nature of social media: social media is dynamic by nature, and online social media platforms are dynamic spaces with frequently changing content. The efficiency of the created algorithms may therefore change over time.
3. Sophisticated Disinformation: Detection algorithms may struggle with highly sophisticated forms of disinformation that mimic legitimate content, potentially leading to false negatives.
4. Algorithm Dependency: A number of variables, including hyperparameter tuning, dataset size, and algorithmic choice, can affect how well supervised AI algorithms perform.
5. Evaluation Metrics: Although frequently employed, these measures might not accurately reflect the intricate nature of fake news identification, especially in real-world circumstances.

## 1.5 Conclusion

The research is organized as follows:

1. Chapter 2: gives a quick rundown of earlier attempts to identify fake news.
2. Chapter 3: gives a thorough explanation of the text mining procedure, the suggested model, the performance indicators, and the supervised artificial intelligence algorithms employed.
3. Chapter 4: outlines the datasets and experimental findings from four distinct datasets using twenty-nine supervised artificial intelligence algorithms.
4. Chapter 5: reflects on the results, provides concluding remarks, and suggests potential directions for future research.

# CHAPTER TWO

# LITERATURE REVIEW

## 2.0 Understanding Fake News

### 2.0.1 Fake News – Definition

According to dictionary.cambridge.org, Fake news is defined as information that is fraudulent or misleading but is presented as factual news (Cambridge Dictionary, 2023). It can come in a variety of shapes, like made-up tales, hoaxes, edited photos or videos, and false headlines. The dissemination of fake news frequently aims to confuse or mislead the public, and it can have serious social, political, and economic repercussions (UO Libraries, 2023). Fake news is a problem that has to be addressed since it can spread quickly and reach a large audience in the environment of online social media.

**Characteristics of Fake News**

According to (Lamprou et al. 2021), some of the characteristics of fake news may include:

**-** Information fabrication: False or entirely manufactured information is a common component of fake news articles and tales. This can contain made-up facts, figures, quotations, and even sources that don't exist.

**-** Deceptive Headlines**:** Sensational and false headlines are frequently used in fake news publications. They frequently exaggerate or mislead the substance in an effort to catch the reader's attention.

**-** Misleading photographs and movies: To further muddy the lines between fact and fiction, false news may also include modified photographs, photoshopped images, or deepfake movies.

**-** Impersonation and Satire: Some fake news is produced as satire or parody, while others involve fooling the public by using the persona of reputable news organizations or persons.

**Impact of Fake News**

- Social Division and Polarization: By reinforcing preexisting ideas and biases, encouraging distrust amongst various groups, and generating echo chambers, the propagation of fake news can lead to social division and polarization (liguides, 2022).

- Political Repercussions: False information has the power to sway public opinion, elections, and political judgment. It has been linked to the spread of erroneous information regarding candidates, laws, and current affairs (Lamprou et al. 2021).

- Economic Implications: The spread of false information can have an impact on the economy by undermining businesses, stock markets, and public confidence in financial institutions (Mwangi, 2023).

- Public Trust Erosion: Constant exposure to false information can cause people to lose faith in the media, authorities, and even the idea of truth itself (Mwangi, 2023).

### 2.0.2 History of Fake News

Spreading false or misleading information under the guise of authentic news with the intention of damaging someone's or something's reputation or profiting from advertising is known as fake news. Although the term "fake news" was first used in the 1890s, a time when sensationalized newspaper stories were common, the dissemination of false information has always been a problem. However, the term lacks a clear meaning and is frequently used widely to speak of any kind of false information. It's been utilized by well-known people to identify news that doesn't support them. On the other hand, disinformation refers to the malevolent deliberate dissemination of misleading information, frequently carried out by hostile foreign actors, especially during electoral processes (Safieddine & Hammad, 2020)

**Historical Examples**

According to (Posetti & Matthews, 2018), the examples below show a chosen chronology of a certain "information disorder" that spans time.

* 1835 – The Great Moon Hoax

The New York Sun published six pieces purporting to be based on astronomer Sir John Herschel's research regarding the discovery of (false) life on the moon.

* 1917 – The German corpse factory

During World War I, British propaganda emphasized demonizing the German allies. The Times and The Daily Mail published articles in 1917 alleging that the German troops were using their own soldiers' corpses to boil down for fats, bone meal, and pig food because of a lack of fat in Germany brought on by the British naval blockade. This had consequences when the first accounts of the Holocaust's atrocities surfaced during World War II.

The true accounts of Nazi crimes are claimed to have been initially rejected in 1917 because to the deception contained in news reporting.

* 1933 – Reich Ministry of Public Enlightenment and Propaganda established

Joseph Goebbels established the Reich Ministry of Public Enlightenment and Propaganda upon the rise of Nazism in order to spread Nazi ideologies of inciting hatred and murder of Jews through the press and theater, among other media. Nazi propaganda was a major source of inspiration for those who massacred thousands of European Jews and other victims of the Nazi regime. Furthermore, technology made it possible for millions of people to watch and approve of mass murder and racial profiling.

* 1939-1945 – World War II

The War that Hitler Won, by Edward Herzstein, was published in 1978. Herzstein called the Nazi propaganda effort "the most infamous propaganda campaign in history." Because of how well the Nazis harassed and demonized Jews, crimes were carried out with widespread backing, and Holocaust denial is still prevalent in the twenty-first century.

* 1972-1990s – South Africa’s propaganda war

To gain support and stifle criticism of its apartheid practices, the South African government developed a sophisticated, clandestine foreign lobbying and propaganda effort. The campaign was spearheaded by government minister Eschel Rhoodie and was aimed at powerful individuals in Western cities. The initiative was first noticed by local investigative journalists in the late 1970s, and it continued until the early 1990s.

* 1996 – The Daily Show begins

Satirical in nature, the news describes itself as "fake news." It all began with the introduction of an American television program that set the groundwork for the genre of satirical news to emerge as "some sort of corrective to, and replacement for, mainstream journalism."

* 2010 – Egypt's President is featured prominently in an Egyptian state-run newspaper that uses pictures of international leaders.

Al-Ahram newspaper 'photoshopped' a photograph of world leaders heading to the start of a session of Middle East peace talks to put then-President of Egypt, Hosni Mubarak, at the head of the group. In the unaltered image, he was, in fact, lagging.

Israel and Palestine are seen behind the US and its representative. The trickery was exposed by a blogger from Egypt

* 2016-2017 – Troll farms and ‘fake news’-for-profit

As the US election approached in 2016, allegations from the international media revealed that adolescents ran a profitable troll farm in the little village of Veles in the Former Yugoslav Republic of Macedonia. It was discovered that more than 100 websites supporting Trump and disseminating fraudulent material were registered in Veles. During the last three months of the campaign, one of their operators earned US $16,000. Included in the video are widely disseminated hoaxes concerning the Pope's endorsement of Donald Trump and the 'imminent indictment' of Hillary Clinton, the Democratic presidential nominee. The owners of the fake news websites made enormous sums of money when Google AdSense and other automated advertising engines tracked their wildly inaccurate content. In the last weeks of the campaign, President Obama talked a lot about the "digital gold rush" that Veles' fake news farm was having. Hyperpartisan "news" websites that profit from disseminating misleading information are also widespread in the United States. A BuzzFeed investigation in 2017 found that the fraudulent and misleading content that one Florida-based organization targeted on many websites, catering to both conservative and liberal audiences, was fueled by confounded outrage that generated a lot of Facebook engagement. One of their objectives was to "increase their advertising revenue or metrics."

* 2018 – Cambridge Analytica Scandal

The Observer, The New York Times, and Channel 4 News were informed in March 2018 by a whistleblower that a network of companies operating under the name "Cambridge Analytica"—a company that specializes in psychological profiling and micro-targeted political messaging—and a psychology professor at Cambridge University, who was working privately, had taken advantage of a sizable dataset that had been culled from millions of Facebook users. Using the information, the company targeted certain voter groups ahead of the 2016 US Presidential Election. According to undercover reporting by Chanel 4, company leaders boasted about using their data to target consumers with propaganda and fake information. Before leaving the company in 2016 to spearhead Donald Trump's presidential campaign, Steve Bannon was the vice president of the company. Executives from Cambridge Analytica were shown boasting on tape that they and its partners had worked in over 200 global elections, including those in the Czech Republic, Argentina, Nigeria, Kenya, and India. Whistleblower Christopher Wylie claims that Cambridge Analytica "cheated" the 2017 Brexit referendum. The disclosures led to the closure of the company.

**Evolution in the Digital Age**

Emergence of social media like the rise of platforms like Facebook, Twitter, and YouTube has given fake news a rapid and global distribution channel. With the internet, fake news can spread virally, reaching millions of people within hours. The lack of gatekeepers and fact-checkers on social media has exacerbated the problem (Adriani, 2019).

### 2.0.3 Origins of Fake News

False news can come from a variety of sources, including people looking to forward an agenda, bad actors, and even artificial bots. For the development of efficient detection and mitigation measures, it is imperative to comprehend the sources of fake news. Due to their large user bases and ease of information sharing, online social media platforms serve as a fertile environment for the dissemination of bogus news.

**Sources of Fake News**

* Individuals and Small organizations: Some false news is produced by people or small organizations who want to advance their own ideologies, objectives, or personal convictions.
* State-sponsored disinformation: To sway elections, public opinion, or geopolitical events, governments and state actors may launch fake news operations.
* Negative Bot Networks: On social media platforms, automated bot networks can amplify bogus news material, giving the impression that it is more common and reliable than it actually is.

**Motivations Behind Fake News**

* Political Manipulation: False information can be used to sway public opinion, undermine political rivals, or further a certain political objective.
* Financial Gain: Clickbait websites use advertising to make money, which encourages the creation of sensational and deceptive material.
* Ideological convictions: Regardless of its reality, some people and organizations broadcast fake news to support or advance their ideological convictions.

The point listed above are some of the motivations behind the creation and propagation of fake news according to (Townsend, 2017).

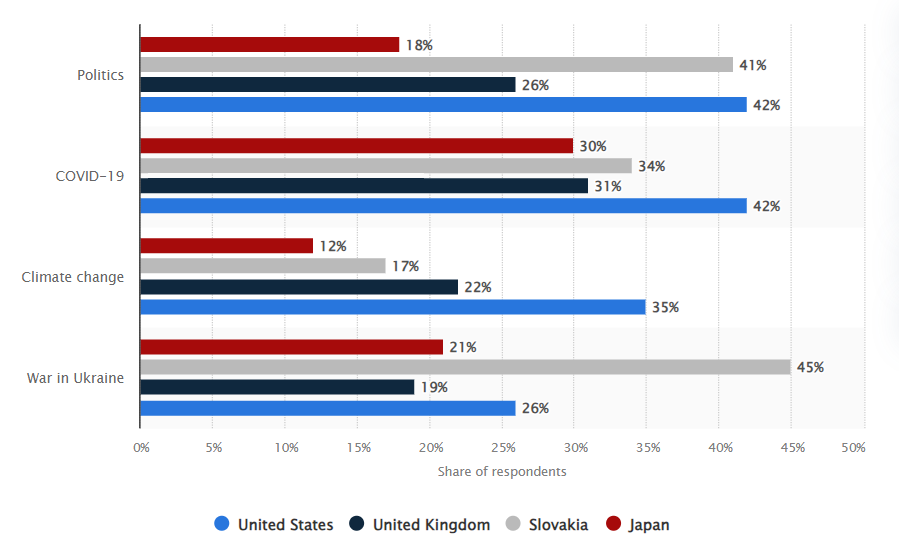


Figure 2.1 As of February 2023, select global news consumers encountered misleading or false information on critical subjects in the preceding week. (statista, 2023)

## 2.1 Evolving Trends

### 2.1.1 Traditional/Old-fashioned Fake News vs. Virtual Fake News

Traditional fake news was often spread by written materials, broadcasts on television or radio, or both. False information spread through digital channels, especially social media websites and platforms, is referred to as virtual fake news. Due to the usage of deepfakes, automated bots, and algorithmic amplification, the shift to virtual fake news has created additional difficulties, making detection and mitigation more difficult.

**Characteristics of Traditional Fake News**

* Predominance in Pre-Digital Media: Print media such as newspapers, magazines, television, and radio were the main sources of traditional fake news (Parliamentary question, 2018).
* Limited Dissemination Channels: Compared to virtual fake news, traditional fake news had a smaller impact on local or regional audiences.
* Absence of Multimedia Elements: Text and static photos were the mainstays of traditional false news.

**Characteristics of Virtual Fake News**

* Digital information Manipulation: Deep-fake movies, altered photographs, and computer-generated text are just a few examples of the digital tools used by virtual fake news to manipulate information.
* Social media platforms offer a global distribution network for fictitious fake news, enabling its quick spread to a sizable audience (Adriani, 2019).
* Rapid Dissemination: Fake news may spread like wildfire in a matter of hours, making it difficult to stop or refute.

### 2.1.2 A Technological Evolution

The spread of virtual fake news has been significantly influenced by technological advancement. Tools for image editing, natural language processing, and artificial intelligence advancements have made it simpler to produce and disseminate false information. For the purpose of creating AI-based detection algorithms and defenses against false news, it is essential to comprehend the technological components of virtual fake news (Hussein & Hejase, 2022). Technology development has aided in the production and detection of bogus news:

**Role of AI in Fake News Generation**

* Deepfake Technology: Deepfake technology powered by AI can produce extremely convincing fake films and audio recordings, making it challenging to distinguish modified information from real content (Gifu, 2023).
* Chatbots and Automation: Automated chatbots and scripts have the capacity to produce and spread false information at a large scale, overloading social media networks and their users.
* Content Generation Algorithms: Artificial intelligence (AI) algorithms are already capable of producing coherent and contextually relevant fake news pieces, further blurring the distinction between real and fake material.

**Role of AI in Fake News Detection**

* Natural terminology Processing (NLP): AI-powered NLP algorithms can examine text content to spot trends, contradictions, and terminology indicative of false information (Gifu, 2023).
* Image and video analysis: AI models can look for manipulation or deepfake characteristics in images and videos, which can help in the identification of visual misinformation.
* Network Analysis: AI-driven network analysis can find questionable trends in the distribution and sharing of content, potentially exposing fake news campaigns.

## Detecting and Combating Virtual Fake News

**Virtual Fake News - Challenges and Solutions**

Detecting virtual fake news within online social media presents a set of intricate challenges that require innovative solutions:

**Challenges in Detection**

* Volume and Velocity of Data: Manual fact-checking is all but impossible due to the sheer volume of content on social media and its quick diffusion (Endsley, 2018).
* Adapting Misinformation Tactics: To avoid discovery, malicious actors constantly modify their strategies and use new technology.
* False Positives and Negatives: Automated detection methods may produce false positives by wrongly detecting legitimate information or false negatives by overlooking sophisticated fake news.

**Solutions and Approaches:**

* Supervised Machine Learning Models: A large number of fake news detection systems use supervised machine learning models that have been trained on labelled datasets of false and real news articles.
* User Behavior Analysis: Some methodologies concentrate on examining user patterns, such as distributing dubious content or interacting with well-known fake news sites.
* Hybrid Approaches: Accuracy can be increased by combining AI-driven detection with human oversight via crowdsourcing fact-checking or AI-driven alerts for users.

## Related Works:

In 2022, Shaina and Chen suggested a novel architecture for detecting fake news that was intended to overcome issues like the early detection of bogus news and the lack of labelled data needed to train the detection model. This system uses data from the report articles and the social setting to identify fake news. It is based on a transformers design and was influenced by the BART architecture. The model's encoder blocks carry out the representation learning task. The issues of early fake news detection are further helped by the decoder blocks, which forecast future behavior based on historical observations. Compared to autoencoding models (exBake, BERT), autoregressive models (Grover, GPT-2) performed better for early detection. ExBAKE, FANG, 2-Stage Transf., Declare, TriFN, and VGCN-BERT all shown improved performance in later time steps, according to the authors. This behaviour was attributed to longer learning time. The performance of the LG, TextCNN, and XGBoost was noted to have been inferior to that of the other baselines.

An effective deep diffusive neural network model for fake news identification named FakeDetector is proposed in the publication "FakeDetector: Effective Fake News Detection with Deep Diffusive Neural Network" (Zhang et al., 2020). By extracting explicit and latent elements from textual data, the model simultaneously learns subjects, writers, and news article representations. This strategy is based on the finding that looking into correlations between news pieces, their subjects or topics, and their authors or distributors can help with false news identification (Qureshi et al., 2021).

In 2019, Benamira et al. examined the use of semi-supervised learning and graph neural networks for the identification of false news. The requirement for efficient detection techniques that can use both labeled and unlabeled data to increase accuracy is what spurred this strategy. In order to handle data represented in graph domains, neural network models known as "graph neural networks" (GNNs) were developed (Scarselli et al., 2009). They have been used for many different purposes, such as graph node categorization. Both labelled and unlabeled data are used in semi-supervised learning, a learning methodology, to train models. Graph neural networks and semi-supervised learning have the potential to enhance fake news identification. GNNs may capture interactions and dependencies between nodes by utilizing the graph structure of social networks or other data representations, which might be helpful for spotting patterns of fake news propagation. Additionally, semi-supervised learning with unlabeled data enables the model to learn from a bigger data set, potentially improving its capacity to reliably identify bogus news. The experimental results, in the authors' opinion, showed that the suggested methodology performed better than conventional classification algorithms, particularly when trained on a small sample of tagged articles.

A benchmark framework for examining and discussing machine/deep learning methods used in fake news detection was given by Galli et al. in 2022. framework intends to overcome the difficulties associated with fake news identification, such as the variety of subjects and language elaborations employed in its creation. It offered a consistent evaluation framework for contrasting the effectiveness of various detection models. The authors investigated how different elements, including textual, social, and network-based features, may be used to identify bogus news. They evaluated the effectiveness of several methods and offered details on the advantages and disadvantages of every strategy. Three datasets—FakeNews, a big unbalanced dataset; PHEME; and LIAR, two smaller datasets—were used for these analyses. First, different machine learning models (Logistic Regression, Decision Tree, SVC and Random Forest) were compared; the results showed that Logistic Regression was the most successful model in terms of efficacy and efficiency measures. In contrast to other models, Logistic Regression has several advantages, such as simplicity of interpretation, speed of implementation, and few tuning parameters. B.E.R.T achieved the greatest overall results because it conducted context-based word-level embedding, while being challenging to train. By combining multimedia and content analysis, a multimodal technique has been further developed to do a false image classification. This approach yielded the greatest results in terms of recall, accuracy, F1 and precision using multimedia data.

Using the swarming traits of fake news, the FakeSwarm fake news recognition system was introduced in 2023 (Wu & Ye, 2023). The authors used three distinct swarm feature types— metric representation, principal component analysis and location encoding—to demonstrate the importance of considering swarming characteristics in the identification of false news.. The ISOT FAKE NEWS dataset was used by the authors to carry out their analysis. There are 23,481 phony news pieces and 21,417 true news articles in the dataset, which contains news stories from 2015 to 2018. The fake news pieces were gathered from several sites that fact-checking agencies like Politifact and Wikipedia had identified as false. On the other hand, reliable content came through crawling Reuters.com. The authors' examination of the available data revealed that combining all three swarm feature categories produced an excellent accuracy of more than 97% and a f1-score.

A two-step method was proposed by (Özbay & Alataş, 2020) for identifying fake news on social media. Pre-processing the data was the initial step in the method's procedure to convert unstructured data sets into structured data sets. The news texts and other texts in the data set are vectorized using the Document-Term Matrix and the acquired TF weighting algorithm. In the second stage, 23 supervised AI algorithms were applied to the dataset, which had previously been text-mined and organized into a structured format. This study used publicly available datasets to empirically test twenty-three intelligent classification techniques. The four-evaluation metrics (i.e., recall, accuracy, F-measure and precision) were then used to compare these classification models. The authors claimed that the Decision Tree method has produced the best mean values in terms of accuracy, precision, and F-measure. In terms of recall metrics, the 1000 value algorithms ZeroR, CVPS, and WIHW appeared to be the best.

Wang et al. (2022) introduced a novel fine-grained multimodal fusion network (FMFN) to fully fuse textual and visual information for the purpose of identifying fake news. When a tweet has both text and an image, word embeddings from the text are extracted using a pretrained language model, and each word embedding can be considered a textual feature. Deep CNNs are utilized to extract various visual aspects from the image. Scaled dot-product attention was used to combine word embeddings from the text with multiple feature vectors representing different aspects of the image. This approach captures the dependencies between textual and visual features more accurately and accounts for correlations between different visual features. The FMFN, which fuses multiple visual features and multiple textual features, learns a joint representation that is superior to that learned by other methods for fusing visual representation and text representation obtained by combining a text representation with a visual representation when identifying fake news, according to the findings of an extensive experiment the authors carried out on a publicly available Weibo dataset.

The Mc-DNN multi-channel deep learning model was introduced by Tembhurne et al. in 2022. It employs and processes news headlines and articles from several channels to distinguish between real and fraudulent news. The performance of Mc-DNN was investigated using the combinations of RNN and CNN, GRU and CNN, LSTM and CNN, BiGRU and CNN, and BiLSTM and CNN. BiLSTM and CNN Mc-DNN were reported to achieve the maximum accuracy of 99.23% and 94.68%, respectively, in the task of false news identification on the ISOT fake news dataset and FND (false News Dataset, 2020).

The authors of a previous publication presented an entity debiasing framework (ENDEF), which generalizes fake news detection algorithms to future data by reducing entity bias from a cause-and-effect approach (Zhu, et al., 2022). Based on the causal connection connecting news entities, news contents, and news truthfulness, the authors separately modelled the contribution of each cause (entities and contents) during training. They reduced the direct influence of the entities during the inference stage in order to reduce entity bias. Extensive offline studies on the English and Chinese datasets demonstrate that the proposed method may greatly improve the performance of base false news detectors, and online tests validate its superiority in practice.

According to Murayama et al. (2021), the bulk of false news datasets are dependent on a specific time period. As a result, detection models developed using such a dataset struggle to identify unexpected fake news brought about by political and societal changes; they may also produce biased output from the input, such as names of particular people and organizations. Because it is a result of the origination date of news in each dataset, the authors called this diachronic bias. The authors developed masking techniques based on Wiki data to reduce the impact of human names and test whether they strengthen fake news detection algorithms through trials using in-domain and out-of-domain data. Based on their tests, the authors were able to confirm that these masking techniques increased model robustness and accuracy in out-of-domain datasets.

In a different approach, Min et al. (2022) formulated the problem of social context-based fake news identification as a diverse graph classification problem and introduced the Post-User Interaction Network (PSIN). This model effectively models the post-post, user-user, and post-user connections in social context while maintaining their inherent features through the use of a divide-and-conquer strategy. The authors employed an adversarial topic discriminator for topic-agnostic feature learning in order to broaden the method's applicability to recently emerging subjects. Extensive experiments demonstrate that the proposed method outperforms SOTA baselines in both on-topic and off-topic scenarios.

Sahoo & Gupta, 2021, proposed an autonomous detection of false news solution for the Chrome environment to detect fake news on Facebook. The authors used deep learning to analyze the behavior of the account by utilizing a range of Facebook account-related data in addition to other features connected to news articles. These authors claimed that multiple experimental analyses of actual material showed that their intended strategy to detecting false news had more accuracy than the current state-of-the-art methods.

A thorough analysis of the most advanced techniques for identifying malicious users and bots based on the various features suggested in the authors' unique taxonomy was published in a paper by Shahid et al. in 2022. In order to aid researchers who are new to this subject, the authors discussed numerous important problems and prospective future research areas in an effort to avoid the critical issue of false news detection.

Shu et al., 2019 looked at the problem of understanding and using social media user profiles for the identification of fake news in another paper. By tracking users' sharing habits, the authors were able to identify group representative users who are more likely to spread both false and accurate news. Subsequently, they carried out a comparative study of explicit and implicit profile attributes among various user groups to ascertain their capacity to facilitate the differentiation of false from authentic news. The authors demonstrated the usefulness of user profile features for exploitation with a bogus news categorization problem. The authors further confirmed the effectiveness of these characteristics using feature significance analysis.

In a study by Kaliyar et al., 2020, the authors analyzed the substance of the news piece and the echo chambers on social media—groups of people who have the same beliefs—to determine which news was fake. A tensor representing social context—that is, the association between user profiles on social media and news articles—was produced by combining news, user, and community data. The tensor and the news material were integrated to create a representation that included the social context as well as the news information. The suggested method had been tested on the real-world dataset BuzzFeed. The decomposition-derived parameters were used as features in the news classification process. An ensemble machine learning classifier (XGBoost) and a deep neural network model (DeepFakE) were employed for the classification task. According to the scientists, the proposed model (DeepFakE) beats the current techniques for identifying fake news since it uses deep learning on a combination of social context-based qualities and news content as an echo chamber.

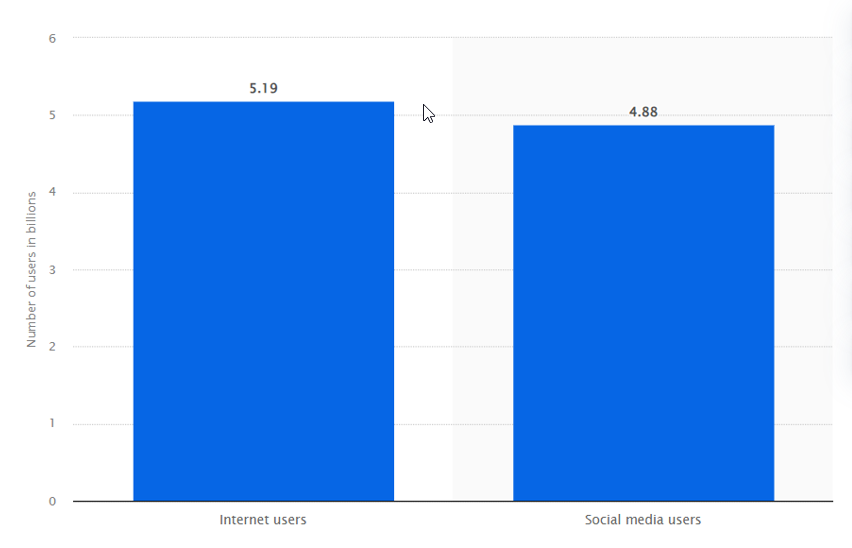


Figure 2.2 Number of internet and social media users worldwide as of July 2023(in billions) (statista, 2023)

# CHAPTER THREE

# METHODOLOGY

This chapter delves into the heart of the study, focusing on the implementation and methodology of utilizing Artificial Intelligence (AI) to combat the pervasive issue of fake and counterfeit news within the realm of social media. This section navigates through the intricacies of data collection, preprocessing, and the selection of supervised AI algorithms tailored for text classification tasks. By elucidating the process of model training, validation, and performance evaluation using standard metrics, this chapter offers a comprehensive insight into the strategies employed to discern between authentic and deceptive information circulating across various social media platforms.

## 3.0 Data Collection

### 3.0.1 Dataset 1

This dataset of political news was taken out of Kaggle, it was a dataset compiled and provided for a community prediction competition (Lifferth, 2018). It consists of 25116 rows of train data. Each containing id, title, author, text and label columns. And 5864 rows of test data. Each also containing id, title, author, and text columns. This dataset was one of four datasets used in train our different machine learning models.

### 3.0.2 Dataset 2

The second dataset came from the Universit of Victoria’s Online Academic Community (ISOT Fake News dataset, 2022). Thousands of false news and real articles are combined to create this fake news dataset. The dataset was assembled from a number of stories published on both reputable and shady news websites.

### 3.0.3 Dataset 3

A publicly accessible dataset for identifying false news is called LIAR (Wang W. Y., 2017). POLITIFACT.COM offers a thorough analytical report and links to the original documents for each case, where 12.8K manually labeled brief utterances were gathered over a ten-year period in a variety of circumstances. Research that involves fact-checking can also make use of this dataset. It's interesting to note that this new dataset is orders of magnitude larger than prior, comparable public fake news databases. The LIAR dataset contains 12.8K human labelled brief statements from POLITIFACT.COM's API; a POLITIFACT.COM editor verifies each statement.

### 3.0.4 Dataset 4

The fourth dataset was obtained via the https://zenodo.org/record/4561253 page on the Zenodo website. For the purpose of detecting fake news in text data, Verma et al. (2021) created the WELFake dataset, which has of 72,134 news stories with 35,028 true and 37,106 fraudulent news. These are arranged in columns for text, label, serial number, and title. The news heading is represented by the title, the news content is by the text, and the label indicates if the news is true or fake (0 being fake and 1 being real). The serial number begins at 0. Just 72,134 of the roughly 78,098 records are accessible according to the data set.

### table 3.1: Dataset source

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | URL | Source | Remark |
| Dataset 1 | https://www.kaggle.com/competitions/fake-news/data | Kaggle | Kaggle.com |
| Dataset 2 | https://onlineacademiccommunity.uvic.ca/isot/2022/11/27/fake-news-detection-datasets | onlineacademiccommunity.uvic.ca | onlineacademiccommunity.uvic.ca |
| Dataset 3 | https://paperswithcode.com/dataset/liar | paperswithcode | paperswithcode.com |
| Dataset 4 | https://zenodo.org/record/4561253 | Zenodo | zenodo.org |

## 3.1 Text Categorization

Text categorization also know has text classification is a task in natural language processing (NLP) and machine learning that entails classifying text documents into specified groups or labels according to their content (Khan & Yadav, 2019). Text classification aims to automatically classify text documents into one or more specified categories so that organizing, searching, and deriving insights from massive amounts of text data is made simpler.

CLASSIFICATION

Text Preprocessing

Features Extractions

UNSTRUCTURED

(Low Quality) TEXT

TEXT REPRESENTATION

Figure 3.1 Text Classification Pipeline (Naseem et al., 2020)

### 3.1.1 Text Preprocessing

A series of procedures known as text preprocessing are used to clean and get text data ready for analysis (Chai, 2022). The following actions are usually involved:

* **Lowercasing:** To guarantee uniformity in text data, convert every text to lowercase. This lessens the likelihood of case sensitivity problems, which facilitates word matching and processing.
* **Tokenization:** Divide the text into discrete words, or tokens, using tokenization. Tokenization plays a crucial role in segmenting sentences or paragraphs into manageable chunks for analysis.
* **Removal of Stop Words:** Termites such as "the," "and," "in," and "of" should be eliminated from the text. For many NLP tasks, these terms are typically not informative.
* **Removal of punctuations:** Symbols, special characters, and punctuation are frequently unnecessary for analysis.
* **Lemmatization**: Reducing words to their most basic or root form is known as lemmatization. Consolidating related words is aided by stemming and lemmatization (e.g., "running" and "ran" both become "run").
* **HTML Tag Removal:** You might need to remove HTML tags if the text in your data originates from web pages.
* **Text Normalization:** Handling email addresses, URLs, and other text-specific patterns may require additional normalization procedures.

### 3.1.2 Feature Extraction

Textual data must be transformed into vector or numerical representations before being fed into machine learning algorithms. This procedure is known as feature extraction. It involves converting unprocessed data into a more comprehensible and useful representation, and it is an essential stage in machine learning and data analysis (Srikumar, 2017). It seeks to locate and extract from the raw data the most pertinent and non-redundant characteristics, therefore improving the efficiency of machine learning algorithms. There are many techniques for feature extraction in text data. However, a combination of Bag-of-Words and TF-IDF (Term Frequency-Inverse Document Frequency) were the chosen methods.

**Bag-of-Words (BoW)**

The bag-of-words model is a method of representing text as an unordered collection of words. It is commonly used in natural language processing (NLP) and information retrieval (IR). The bag-of-words model disregards grammar and word order, but it keeps multiplicity. This means that if a word occurs multiple times in a document, it is counted multiple times in the bag-of-words representation of that document (Qader et al., 2019).

**Term Frequency (TF)**

A statistical metric called Term Frequency-Inverse Document Frequency (TF-IDF) assesses a word's significance in a document within a set of documents (Dodiya, 2021). It is a commonly used method in text mining and information retrieval.

The number of times a term (word) appears in a document is its term frequency (TF). It is a straightforward indicator of the frequency with which a term appears in a given document.

**Inverse Document Frequency (IDF)**

The measure of a term's frequency of occurrence in a set of documents is called Inverse Document Frequency (IDF). It is calculated by taking the logarithm of the total number of documents and dividing it by the number of documents that contain the term.

**TF-IDF**

The result of TF and IDF is TF-IDF. It evaluates a term's significance in a document by considering both its occurrence in the document and its rarity within the collection. A term's importance to the document and not merely its commonness is indicated by a term's high TF-IDF score (Dodiya, 2021).

## 3.2 Artificial Intelligence algorithms

This section a shade a light on the 30 different algorithms used in this paper. They include:

1. “lbfgs” Logistic Regression

refer to a specific type of logistic regression algorithm that uses the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimization method. Logistic regression is a statistical method used for binary classification problems. Based on one or more predictor factors, it estimates the probability of a binary result (1/0, Yes/No, True/False). L-BFGS is a popular optimization algorithm used for finding the minimum of a function, typically in the context of machine learning and numerical optimization. It is an iterative method that belongs to the family of quasi-Newton methods. L-BFGS is known for being memory-efficient and suitable for problems with a large number of parameters.

When combined, "lbfgs Logistic Regression" suggests that logistic regression is being used with the L-BFGS optimization method to train a binary classification model. This combination is often employed when dealing with machine learning tasks that require optimizing the logistic regression model's parameters to fit the data.

The sigmoid function, a logistic function, is used in logistic regression to map predictions and their probability. An S-shaped curve known as the sigmoid function transforms any real number into a range between 0 and 1.

For logistic regression, the sigmoid function is known as an activation function and is described as follows:

(1)

* *P*(*Y*=1∣*X*) is the probability of the target variable *Y* being 1 given the predictors *X*
* *β*0​,*β*1​,*β*2​,…,*βn*​ are the coefficients of the model
* *X*1​,*X*2​,…,*Xn*​ are the predictor variables.
* *e* is the base of the natural logarithm (Euler's number).

The logistic function (2)

transforms the output of a linear equation (*β*0​ + *β*1​ *X*1 + *β*2​ *X*2 +… + *β*n *X*n​) into a range between 0 and 1, representing probabilities. (Kanade, 2022)

The coefficients *β*0​,*β*1​,*β*2​,…,*βn* ​ are estimated using optimization algorithms (often maximum likelihood estimation) to minimize the error between predicted probabilities and the actual outcomes in the training data.

1. “liblinear” Logistic Regression

liblinear is a library for large linear classification, which includes logistic regression. It is a popular choice for training logistic regression models on large datasets, because it is fast and efficient. Coordinate descent is the method that liblinear utilizes to solve the logistic regression optimization issue.

Coordinate descent is an iterative algorithm that works by optimizing one coordinate at a time. It is a simple and efficient algorithm, and it is well-suited for large datasets. The "liblinear" solver is an optimization algorithm used to fit the logistic regression model. It's based on a linear support vector machine (SVM) algorithm and works well for small to medium-sized datasets. It optimizes the logistic regression cost function using techniques like coordinate descent. Refer to equations 1 and 2 for reference.

1. “newton-cg” Logistic Regression

The "newton-cg" solver stands for "Newton Conjugate-Gradient." It is a numerical optimization technique that combines the Newton-Raphson method with the conjugate gradient method to determine the best logistic regression model parameters. This solver is known for its efficiency and suitability for a wide range of logistic regression problems.

When combined, "newton-cg Logistic Regression" indicates that logistic regression is being used, and the "newton-cg" solver is chosen as the optimization method to train the logistic regression model. This combination is often applied when you have a logistic regression problem that benefits from the characteristics of the "newton-cg" optimization algorithm. Refer to equations 1 and 2 for reference.

1. “sag” Logistic Regression

Stochastic Average Gradient Descent (SAG) logistic regression is a solver that uses stochastic average gradient descent to train a logistic regression model. Stochastic average gradient descent (SAG) is an iterative algorithm that updates the model parameters by averaging the gradients of a small batch of samples at each iteration. SAG is a fast and efficient algorithm for training logistic regression models on large datasets. It is particularly well-suited for datasets with a large number of features. SAG is also less sensitive to the initial parameter values than other logistic regression solvers. Refer to equations 1 and 2 for reference.

1. Random Forest

is a supervised machine learning approach that can be used for regression and classification applications. In order to generate a final forecast, it builds a large number of decision trees during the training phase and averages them.

The foundation of Random Forests is the idea of ensemble learning, which combines the predictions of several different models to generate a forecast that is more accurate. Because of this, Random Forests are less likely than individual decision trees to overfit.

In contrast to decision tree classifiers, which receive the root node by using the gini index or information gain, random forests obtain the root node randomly, and the splitting of the attribute nodes also occurs randomly (Azeez et al., 2021).

2 (3)

= 1 – [(P+)2 + (P-)2]

P+ represents the likelihood of a positive class, whereas P\_ denotes the likelihood of a negative class (Saini, An Introduction to Random Forest Algorithm for beginners, 2022).

1. Perceptron

Its predictions are based on a linear predictor function, which combines the feature vector with a set of weights, because it is a linear classifier.

A collection of input values is fed into the perceptron, and it multiplies the results by a set of weights. After that, an activation function is applied to the total of these products to provide an output. Usually a step function, the activation function determines whether the input is higher or less than a threshold and outputs a binary value (0 or 1) accordingly. Many algorithms can be used to train a perceptron; however, the perceptron learning algorithm is the most widely used one. In order for this technique to correctly identify all of the training data, iteratively modifying the perceptron's weights is how it operates.

The following is the fundamental equation for a perceptron's operation:

Perceptron_6. (4)

w = vector of weights with actual values

b = bias, which is an independent component that modifies the boundary away from the origin.

x = is the input x values' vector.

(5)

m denotes the number of Perceptron inputs.

Either "1" or "0" can be used to represent the output. Depending on the activation function being utilised, it can also be represented as "1" or "-1." (Banoula, 2023)

1. Ridge Classifier

is a linear algorithm for classification that has the Ridge regression algorithm as its foundation. Based on features, it is used to categories data into two or more classes. Ridge regression is a regularized linear regression approach that prevents overfitting by including a penalty term in the loss function. Large model coefficients are penalized by this regularization term, which makes the model learn fewer complex correlations between the target variable and the data.

Ridge Classifier works by first converting the target variable into {-1, 1} and then treating the problem as a regression task. It then uses the Ridge regression algorithm to learn a linear model that can predict the target variable. The predicted class is then determined by the sign of the predicted target value.

Ridge Classifier's objective function aims to minimize the following cost function:

Cost =  +2) (6)

Here:

* Loss (​,​) represents the loss function used for classification (e.g., typically logistic loss or squared hinge loss).
* *α* is the regularization parameter that controls the strength of the penalty term.
* **w** denotes the coefficients (weights) of the linear function.

The Ridge Classifier seeks to find the optimal weights (**w**) that minimize this cost function, balancing between fitting the training data well and keeping the coefficients small to prevent overfitting.

1. Cat Boost Classifier (Cat Boost)

is a gradient boosting classifier made especially to deal with features that fall into one of the categories. It employs a range of methods to enhance gradient boosting's effectiveness on categorical data, such as:

- Ordered boosting: Cat Boost makes use of ordered boosting to discover connections among categorical variables with an inherent ordering.

- Feature hashing: To effectively handle a high number of categorical features, Cat Boost makes advantage of feature hashing.

- Oblivious trees: Cat Boost makes use of oblivious trees, which are decision trees that don't care which way the category features are arranged.

*F*(*x*) = *B*0​ + ​*fm*​(*x*) (7)

Where:

* *F*(*x*) is the final prediction for input *x*.
* *B*0​ is the initial prediction (often a global constant or the average of the target variable).
* *M* is the number of trees in the ensemble.
* *fm*​(*x*) represents the prediction of the *m*-th tree for input *x*.

To get at the end prediction, the starting prediction is added to the increments predicted by each individual tree ​*fm*​(*x*). These trees are built using the Cat Boost algorithm step-by-step, with each tree built trying to rectify the mistakes the ensemble has made thus far.

1. Nearest Centroid Classifier

is a simple and effective classification algorithm that assigns a new data point to the class whose centroid is closest to the new data point. The centroid of a class is the mean of all of the data points in that class.

To classify a new data point using Nearest Centroid Classifier, the following steps are typically taken:

* + Determine how far the new data point is from each class's centroid.
  + Assign the newly discovered data point to the class whose centroid is nearest to it.

Being a non-parametric technique, Nearest Centroid Classifier does not make any assumptions on the data's underlying distribution. Because of this, Nearest Centroid Classifier is an adaptable method that works well for a range of applications.

Let's define the key components:

* *xij*​ : Feature vector of the *i*-th sample in class *j*.
* *C j* ​: Centroid for class *j*.
* *N*: Number of features.
* *n j* ​: N umber of samples in class *j*.

The steps involved are:

**Centroid Calculation**

For each class *j*, calculate the centroid *cj*​ as the mean of the feature vectors belonging to that class:

*Cj* ​= ​ ​​*xij*​ (8)

This equation computes the centroid *cj*​ for class *j* by averaging the feature vectors xij within that class.

**Distance Calculation**

Given a new test instance xtest ​, compute the distance from xtest to each centroid *cj*​ using a distance metric like Euclidean distance:

Distance(xtest ​, *cj*​) = (xitest− *cji*​)2​ (9)

This equation calculates the Euclidean distance between the test instance xtest ​ and each centroid *cj*​ across all features.

1. Stochastic Gradient Descent (SGD Classifier)

is a training algorithm for machine learning models that uses optimization. It updates the model parameters iteratively in the direction of the loss function's negative gradient. The direction that the parameters should be adjusted in order to minimize the loss is indicated by the gradient of the loss function.

Because SGD is a stochastic algorithm, it only uses one training example at a time to update the model's parameters. For training huge datasets, where it would be impossible to change the parameters using the full dataset at once, this makes SGD particularly effective (Azeez et al., 2021).

*Y* = *mx* + *b*; the straight-line equation where m is the slope and b is its intercept (10)  
*m* = *m - әm*; *b* = *b - әb*; these are parameters with small change  
*Cost* = 2; this is the cost function for N samples

1. Support Vector Classifier - SVC (kernel=” linear”, C=0.025)

is a linear kernel support vector machine (SVM) classifier with a 0.025 regularization parameter. SVMs are a class of machine learning algorithms that are applicable to applications involving both regression and classification. In order to divide the data points into two classes, they locate a hyperplane in the feature space.

The linear kernel is a simple kernel that calculates the dot product between two data points. This makes it a good choice for datasets with a small number of features. The regularization parameter controls how much the model is penalized for misclassifying data points. A model with a larger regularization parameter will be less likely to overfit the training set and more complex (Saini, Guide on Support Vector Machine (SVM) Algorithm, 2023). The Support Vector Classifier (SVC) is a specific implementation of the Support Vector Machine (SVM) algorithm used for classification tasks. The mathematical representation of SVC closely aligns with the general formulation of SVMs for binary classification. For simplicity, let's consider a linear SVC for linearly separable classes. Given training data (xi, yi) where xi represents feature vectors and yi represents class labels (+1 or -1 for binary classification):

The decision function for SVC is similar to the SVM and is represented as:

(11)

Here:

* **x** represents the input feature vector.
* **w** represents the weight vector.
* *b* is the bias term.
* ⋅ denotes the dot product between vectors.

The optimization problem for SVC aims to find the optimal hyperplane that separates the classes while maximizing the margin and minimizing classification errors. It can be represented as:

minimize2) (12)

subject to*yi*​(**w**⋅**x***i*​+*b*) ≥ 1 for*i* = 1,2,...,*n*

Here:

* ||w|| represents the Euclidean norm or w
* The objective function minimizes 2to maximize the margin.
* The constraints ensure that data points are correctly classified and are sufficiently far from the decision boundary (at a distance of at least 1/||w||)

1. Support Vector Classifier - SVC (kernel="rbf", gama=2, C=1)

is a support vector machine (SVM) classifier with a regularization parameter of 1 and a radial basis function (RBF) kernel. One kind of machine learning method that may be applied to both classification and regression problems is RBF SVMs. In order to divide the data points into two classes, they locate a hyperplane in the feature space.

A non-linear kernel called the RBF kernel uses the distance between two data points to determine how similar they are. For datasets where the data is not linearly separable, this makes it a good option. The amount that the model is penalized for incorrectly identifying data points is determined by the regularization parameter. A model with a larger regularisation parameter will be more sophisticated and less prone to overfit the training set. Refer to equations 11 and 12 for reference.

1. LinearSVC

is a support vector machine (SVM) classifier with a linear kernel. SVMs are a class of machine learning algorithms that are applicable to applications involving both regression and classification. In order to divide the data points into two classes, they locate a hyperplane in the feature space. LinearSVC is similar to the linear SVM and SVC but is implemented with a different optimization algorithm, typically based on the LIBLINEAR library, making it more suitable for large-scale datasets. The optimization problem is expressed as:

LinearSVC is a faster and more efficient implementation of SVM classification for the case of a linear kernel. It also has fewer parameters to tune, making it easier to use.

(13)

Here:

* **w** represents the weight vector.
* *b* is the bias term.
* *C* is the regularization parameter, controlling the trade-off between maximizing the margin and minimizing the classification error.
* ⋅ denotes the dot product between vectors.

The objective function consists of two terms: the first term minimizes the norm of the weight vector to maximize the margin, while the second term represents the hinge loss, penalizing misclassifications. The parameter *C* balances the importance of these two terms, controlling the regularization strength.

LinearSVC aims to find the optimal **w** and *b* that define the hyperplane separating the classes by solving this optimization problem using efficient algorithms. It constructs a linear decision boundary in the input space to classify data points into different classes. LinearSVC is particularly efficient for large-scale datasets and linearly separable problems, offering a computationally tractable solution for linear classification tasks.

1. ZeroR Classifier

is among the most straightforward and fundamental machine learning algorithms. It is essentially a baseline or reference model that functions as a straightforward benchmark for assessing the effectiveness of more intricate machine learning models rather than a learning algorithm in the conventional sense.

The training dataset's most frequent class or value is the only factor used by the ZeroR algorithm to generate predictions. It allocates each instance in a classification task to the class that occurs most frequently in the training set. In regression tasks, each occurrence is given a constant value, usually the target variable's mean or median.

* The principle behind ZeroR is straightforward. For classification tasks: It predicts the most frequent class label from the training data for all instances in the test data, disregarding any features or input variables. Essentially, it always predicts the same class, the mode of the target variable in the training set.
* For regression tasks: It predicts the mean or median of the target variable in the training set for all instances in the test data, irrespective of the input features.

1. Decision Tree Classifier

is an algorithm for supervised machine learning that can be applied to applications involving regression and classification. It functions by building a model of the data that resembles a tree, with each node in the tree representing a decision. After that, the model makes predictions on fresh data points using this tree (Azeez & Fadhal, Classification of Virtual Harassment on Social Networks Using Ensemble Learning Techniques, 2023).

*E*(*S*) = p1 *log*2 *p*1 (14)  
  
where *p* is the *i*-th order probability,

*G*(*S*, *C*) = *E*(*S*) *-* ∑*w2values*(*C*)   *E*(*Sw*) (15)

1. Passive Aggressive Classifier

It is particularly useful when dealing with large-scale, streaming, or online learning scenarios where data arrives sequentially, and models need to adapt to changes. The "passive-aggressive" name is derived from the algorithm's behavior when making updates to its model. It adjusts its model parameters based on a trade-off between making a "passive" update (minimizing the loss function) and an "aggressive" update (correcting misclassifications).

The update rule for the Passive Aggressive Classifier is typically based on the loss function and the gradient of the loss:

Let's denote:

* *w* as the weight vector.
* *x* as the input vector.
* *y* as the true label.
* ŷ ​ as the predicted label.
* *η* as the learning rate.
* *C* as the regularization parameter.

The update rule for the weight vector *w* of the Passive Aggressive Classifier is often represented as:

*W* = *w* + *η* ⋅ (*y*− ŷ ​+ ​ ) *x* – *η* ⋅ *C* ⋅ *w*  (16)

Here:

* *y* − ŷ ​ represents the loss or error term.
* ​ represents the gradient of the loss with respect to the weight vector.
* The regularization term *C*⋅*w* penalizes large weights to control overfitting.

The Passive Aggressive algorithm adjusts the weights in such a way that it tries to minimize the loss while staying close to the previous weight vector. The aggressiveness of the update is controlled by the learning rate (*η*) and the magnitude of the error term.

1. Extra Tree Classifier

is an ensemble learning algorithm that predicts using a set of decision trees. While it is comparable to Random Forest Classifier, there are a few significant differences:

* + Extra Trees Classifier splits the features and samples randomly at each node of the tree, while Random Forest Classifier uses a more informed approach to selecting features and samples.
  + Extra Trees Classifier does not bootstrap the training data, while Random Forest Classifier does.

1. Random Patches

is an ensemble machine learning method that blends the ideas of random subspaces and bagging. Its main application is to enhance the functionality of machine learning models—particularly random forests and decision trees. Because Random Patches focuses on randomizing both the data and the features used for training, it is also known as "Feature Bagging".

**Sampling Data**

* For each model *i* in the ensemble *M*, a subset of the training data *Di*​ is randomly sampled with replacement from the original training set *D*. This sampling might include *N* samples from *D*, where *N* is less than the total number of samples in *D*.

**Selecting Features**

For each subset *Di*​, a random subset of features *Fi*​ is chosen. This might involve selecting a certain number *K* of features randomly from the total feature set *F*.

These randomly generated subsets are then used to train other models (such as decision trees or other classifiers) using the Random Patches technique. The predictions from each model may be combined by voting (for classification) or averaging (for regression) during inference (when producing predictions) to produce the final forecast.

1. Voting Classifier

is an ensemble learning algorithm that creates predictions by combining several classifiers. In order to arrive at a final prediction, a set of base classifiers is trained using the training data. The basis classifiers' predictions are then averaged.

Voting Classifier combines the predictions from multiple base models *M*1​, *M*2​,…,*Mn*​ as follows:

* For **Hard Voting**

Final Prediction=argmax((Mi(x) = c)) (17)

where 1 is the indicator function, *Mi*​(**x**) represents the prediction of the *i*th model for input **x**, and *c* is the class label.

* For **Soft Voting** (for classifiers that output probabilities)

Final Prediction= argmax((Mi(x) = c))

where *P*(*Mi*​(**x**)=*c*) represents the probability of class *c* predicted by the *i*th model for input **x**.

1. Stacked Generalization

Super learning, another name for stacked generalization, is an ensemble learning method that combines the predictions of several machine learning models to generate a final prediction that is more accurate. The way it operates is by using the underlying models' outputs to train a meta-model, also called a stacking model. In order to minimize the total error, the meta-model learns how to combine the predictions of the basic models.

stacked generalization (stacking) involves combining the predictions from multiple base models to train a meta-model that learns to make the final predictions. Let's break down the mathematical representation step by step:

Given:

* Training dataset (*X*, *y*)
* *M* diverse base models denoted as *f*1​, *f*2​,…,*fM*​

**Base Models Training**: Train the base models *fi*​ on the training dataset *X* to obtain predictions:

ŷ*i*​ = *fi*​(*X*), for *I* =1,2,…,*M* (18)

**Formation of the Meta-Features**: Use the predictions of these base models as new features (meta-features) for the meta-model. Form a new dataset consisting of the predictions:

*X*meta ​= [ ŷ1​, ŷ2​,…, ŷ*M*​] (19)

Here, *X*meta ​ is a matrix where each row represents an instance in the original dataset, and each column represents the predictions made by one of the base models.

**Training the Meta-Model**: Train a meta-model (meta-learner) *g* using the meta-features *X*meta ​ and the true target values *y*:

*g*(*X*meta ​) = *y* (20)

1. Multi-layer Perceptron Classifier

is a kind of artificial neural network (ANN) that is suitable for jobs requiring both regression and classification. Multiple layers of networked nodes, or neurons, comprise MLPs. Every neuron in one layer is linked to every other neuron in the layer above it.

Backpropagation is one type of supervised learning technique used to train MLP classifiers. The technique of backpropagation involves minimizing the loss function by modifying the weights of the connections between neurons. The MLP's prediction accuracy of the target values is indicated by the loss function.

Let's consider a simple MLP classifier with multiple hidden layers. Given:

* *X* represents the input features.
* *W*(*i*) represents the weights for the connections between the *ith* layer and (*i*+1)*th*layer.
* *b*(*i*) represents the bias terms for the *ith*layer.
* *A*(*i*) represents the activation of the *ith* layer.

The forward pass through an MLP with multiple hidden layers can be represented mathematically as follows:

**Input Layer to Hidden Layer (Layer 1 to Layer 2)**:

*Z*(1) = *X* ⋅ *W*(1)+*b*(1) (21)

*A*(1) = *σ*(*Z*(1))

where *Z*(1) is the weighted sum of the inputs, *A*(1) is the activation of the first hidden layer, and *σ* is the activation function (e.g., ReLU, sigmoid, tanh) applied element-wise to *Z*(1)

**Hidden Layers (Layer 2 to Layer N-1)**: For each subsequent hidden layer *i* (from 2 to N-1):

*Z*(*i*) = *A*(*i*−1) ⋅ *W*(*i*)+*b*(*i*) (22)

*A*(*i*) = *σ*(*Z*(*i*))

where *Z*(*i*) is the weighted sum of activations from the previous layer, and *A*(*i*) is the activation of the *ith* hidden layer.

**Last Hidden Layer to Output Layer (Layer N-1 to Output Layer)**

*Z*(*N*) = *A*(*N*−1) ⋅ *W*(*N*) + *b*(*N*)  (23)

ŷ ​= *σ*(*Z*(*N*))

where *Z*(*N*) is the weighted sum of activations from the last hidden layer, and ŷ ​ is the final output or prediction.

1. Bernoulli Restricted Boltzmann Machine (Bernoulli RBM)

is a kind of neural network model, more precisely an artificial neural network with generative stochastic properties. RBMs are employed in collaborative filtering, dimensionality reduction, and feature learning, among other unsupervised learning applications. The probability distribution type utilized for the binary units in the model is indicated by the term "Bernoulli" in the name.

Two levels of units make up a Bernoulli RBM: a visible layer and a concealed layer. The input data is represented by the visible layer, and the latent representation of the data is represented by the hidden layer. Because they are binary, the units in the visible layer can either be on or off. The concealed layer's units are binary as well.

It has a mathematical representation in terms of probability related to the visible and hidden units and the energy function.

Suppose we have a Bernoulli RBM with M hidden and N visible units.

**Energy Function**: The energy of a configuration of visible and hidden units in an RBM is given by:

*E*(**v**,**h**) Wijvihj - aivi - bjhj  (24)

Here:

* **V** = (*v*1​, *v*2​, …,*vN*​) represents the states of visible units.
* **h** = (*h*1​,*h*2​,…,*hM*​) represents the states of hidden units.
* *Wij*​ are the weights between visible unit *i* and hidden unit *j*.
* *ai*​ and *bj*​ are the biases for visible unit *i* and hidden unit *j*, respectively.

**Joint Probability**: The joint probability of a configuration of visible and hidden units in an RBM is defined using the energy function:

*P*(**v**,**h**) e -E(v,h)  (25)

Where *Z* is the normalization constant (partition function) calculated by summing over all possible configurations of visible and hidden units:

Z = e -E(v,h) (26)

**Conditional Probabilities**: The conditional probabilities for the states of the hidden units given the visible units and vice versa in a Bernoulli RBM are:

P(hj = 1|v) = *σ* ( Wijvi  + bj ) (27)

P(vi = 1|h) = *σ* ( Wijhj  + ai ) (28)

where *σ*(*x*) is the sigmoid function: *σ*(*x*)= (29)

In order to train a Bernoulli RBM, one usually maximizes the log-likelihood of the training data in order to determine the weights and biases. To update the parameters based on observed data samples, methods like stochastic gradient descent and Contrastive Divergence (CD) are frequently employed.

1. AdaBoost Classifier

is an ensemble learning approach that builds a strong classifier by combining several weak classifiers. It trains a set of weak classifiers iteratively using the training data, and then modifies the weights of the weak classifiers according to how well they perform. A weighted average of the predictions made by the weak classifiers makes up the AdaBoost Classifier's final prediction.

The ensemble prediction is computed by combining the weighted predictions of all the weak learners:

*H*(*x*)=sign( *αt ht* ​(*x*)) ( 30)

Where:

* *H*(*x*) is the final prediction for sample *x*.
* *αt*​ is the weight of weak learner *ht*​ .
* *T* is the total number of weak learners.

1. Gradient Boosting Classifier

is an ensemble machine learning algorithm used for classification tasks. It builds predictive models by combining multiple decision trees through gradient descent optimization. This method is known for its high predictive accuracy and robustness in handling complex relationships in data. It's widely used in applications like spam detection, fraud detection, and image classification. Key parameters to tune include the learning rate, number of trees, and tree depth.

*F*(*x*) = ​learning\_rate×weak\_learner\_predictiont (31)

Gradient Boosting adds weak learners one after the other that outperform the prior models in order to optimize the ensemble model. An effective prediction model is produced when each new learner fixes the mistakes committed by the previous ensemble.

1. Ordinal Learning Model

One kind of machine learning model that is used to predict ordered categorical variables is called an ordinal learning model. This indicates that rather than just predicting the class label, the model also predicts the rank or degree of a variable. When data is organically arranged, such in customer satisfaction surveys or medical diagnosis, ordinal learning models are frequently employed.

Because conventional classification and regression models do not account for the ordered character of the target variable, they are not immediately appropriate for ordinal data. To solve this problem and produce predictions that take into account the data's ordinal structure, ordinal learning models were created.

The Ordinal Logistic Regression is an illustration of an Ordinal Regression model. The cumulative probabilities connected to the ordered categories provide the formula for Ordinal Logistic Regression. Take variable Y, for example, which has K ordered categories (low, medium, and high).

The cumulative probabilities for a given category *k* are represented as:

(32)

Where:

* *P*(*Y* ≤ *k* ∣ *X*) is the probability that the outcome *Y* is less than or equal to category *k* given the input features *X*.
* *αk*​ is the intercept specific to category *k*.
* *β*1​,*β*2​,…,*βp*​ are the coefficients associated with each feature *X*1​,*X*2​,…,*Xp*​ respectively.

This formula extends logistic regression to accommodate ordered categorical outcomes. It is based on the logistic function, also known as the sigmoid function. Depending on the input features, the parameters αk and β's are determined during training to maximise the probability of detecting the provided ordinal outcomes.

1. Extreme Gradient Boosting (XGBoost)

is an ensemble learning algorithm, which means that it creates a stronger prediction by combining the predictions of several weak learners. Decision trees are the weak learners in XGBoost. The decision trees are constructed successively by XGBoost, and each tree is trained to fix the mistakes of the one before it. An objective function made up of a regularization term and a loss function is optimized by XGBoost. The loss function L and the regularization term Ω add up to the objective function that has to be minimized:

Objective = *L*(predictions, labels) + Ω(model) (33)

The particular equations utilised in the execution of XGBoost include the optimisation of the objective function, which combines regularisation and the loss function, as well as the gradient and Hessian computations for building and refining the ensemble of trees.

1. Decision Stump

is a straightforward machine learning model made up of a decision tree with only one level. The algorithm for binary classification relies on the value of a solitary input feature to generate predictions. Decision stumps are frequently employed as building blocks in more intricate machine learning methods, including gradient boosting machines and AdaBoost.

Finding the optimal split for the data based on a single feature is the formula for a decision stump. Assume that we have a dataset that contains the target variable y and one feature, x.

The decision stump's split condition can be represented as:

If *x* < *θ*: predict class *c*1​

Else: predict class *c*2​

Where:

* *x* is the feature value.
* *θ* is the threshold value used to split the data.
* *C*1​ and C2​ are the predicted classes on either side of the split.

The decision stump algorithm looks over the feature space to identify the optimal threshold value that reduces a given criterion, which is frequently information gain or purity. For example, it could maximise Gini impurity or minimise misclassification error in classification tasks.

1. Complement Naïve Bayes (CNB)

is a Naive Bayes classifier variation created to overcome some of the drawbacks of the original Naive Bayes algorithm.

The Naive Bayes classifier's primary drawback is its assumption that the features are unrelated to one another. In real-world data, this is frequently not the case because features might be connected. By identifying the relationships between characteristics and applying this knowledge to increase classification accuracy, Complement NB overcomes this drawback.

The Naive Bayes classifier's tendency to favor the majority class in unbalanced datasets is another drawback. Utilizing a weighted log-likelihood function that assigns greater weight to the minority class, Complement NB overcomes this constraint.

Bayes' theorem provides the following formula for the conditional probability in Complement Naive Bayes:

*P*(*c*∣**x**) = (34)

Where:

* *P*(*c*∣**x**) is the probability of class *c* given the features **x**.
* *P*(**x**∣*c*) is the likelihood of observing the features **x** given class *c*.
* *P*(*c*) is the prior probability of class *c*.
* *P*(**x**) is the evidence probability.

The class-conditional probability is computed differently in Complement Naive Bayes. Rather of modelling the chance of observing features given the class directly, CNB determines the chance of the features given the class's absence (i.e., the complement of the class):

*P*(**x** ∣ ¬ *c*) (35)

1. Multinomial Naïve Bayes (MNB)

Multinomial feature classification challenges are the specialty of MNB. Features that can have a set number of values, such the frequency with which a word appears in a document, are known as multinomial features.

The way MNB operates is by figuring out how likely each class is given the characteristics of the data item. Next, it is predicted that the class with the highest probability is the right class.

* *P*(**x**∣*c*) represents the likelihood of observing the features **x** given class *c*.
* Assuming feature independence, Multinomial Naive Bayes estimates this likelihood as the product of the probabilities of each feature (word or token) given the class.

*P*(**x**∣*c*) = ​*P*(*xi*​∣*c*) (37)

Where  *i*​ represents the *i*-th feature (word or token), and *n* is the total number of features.

* 1. Justification for the choice for Ensemble Models/Techniques chosen.

Twelve ensemble models—which include Random Forest, Cat Boost, Random Patches, Voting, Stacked Generalization, Multi-layer Perceptron (MLP), Bernoulli RBM, AdaBoost, Gradient Boosting, Ordinal Learning Model (OLM), XGBoost, and Extra Tree—were selected from a total of 29 methods. The fact that all 12 algorithms made their decisions by using one or more ensemble techniques (such as stacking, voting, bagging, and boosting) should be noted. Since Ada Boost, Bagging and Random Forest are existing meta estimators, the algorithms in the ensemble were selected to offer a fresh combination of techniques. A combination of Random Forest, SVC, and Decision Tree using the voting ensemble technique or Random Forest, SVC, and Gradient Boosting using stacking ensemble technique was seen to have produced superior outcomes, especially when compared to traditional algorithms.

## 3.4 Performance Metrics for Evaluation.

|  |  |  |
| --- | --- | --- |
|  | **PREDICTED NEGATIVE (0)** | **PREDICTED POSITIVE (1)** |
| **ACTUAL NEGATIVE (0)** | TRUE NEGATIVE (TN) | FALSE POSITIVE (FP) |
| **ACTUAL POSITIVE (1)** | FALSE NEGATIVE (FN) | TRUE POSITIVE (TP) |

### Table 3.40: THE CONFUSION MATRIX

The effectiveness of the supervised artificial intelligence algorithms on the test data was evaluated using a confusion matrix. It is particularly useful for evaluating a model's ability to predict different classes or categories within a dataset. A confusion matrix is used to show the outcomes of a model's attempt to classify a batch of data points into one of several predefined classes or labels. In a typical confusion matrix, the rows represent the actual classes, or ground truth, and the columns represent the classes that the model predicted. The cells of the matrix include counts of the number of data points that belong to each combination of actual and predicted classes. A confusion matrix can yield the following four crucial values:

* True Positives (TP): In these instances, the model successfully predicted the positive class and identified the positive class (the class it truly belongs to). This example shows bogus news that was labelled as such.
* True Negatives (TN): In these cases, the model properly identified the negative class as not falling under the positive category. This instance indicates real news that was classified as real news.
* False Positives (FP): happens when the model predicts the positive class when it should have predicted the negative class, but it does so wrongly. This example shows that genuine news was mislabeled as false news.
* False Negatives (FN): happens when a model predicts a negative class mistakenly when it should have predicted a positive class. In this case, bogus news was mistakenly identified as authentic news.

These values can be used to produce a number of performance measures, including:

* Accuracy: It assesses the overall accuracy of the model's predictions.

(38)

* Precision: It assesses how well the model predicted outcomes that materialized.

(39)

* Recall (Sensitivity or True Positive Rate): It evaluates how well the model detects every instance of positivity.

(40)

* Specificity (True Negative Rate): It evaluates the model's ability to identify every instance of negative.

(41)

* F1 Score: It provides an equitable evaluation of a model's efficacy and is the harmonic mean of recall and precision.

(42)

* Matthews Correlation Coefficient (MCC): is employed to assess the performance of a binary classification model. It offers a fair measure by taking into consideration false positives, false negatives, true positives, and true negatives—even in situations when the classes have different sizes. The MCC is ascertained using this formula.:

(43)

* KAPPA: also referred to as Cohen's Kappa, is a statistic used to assess how well a classification model is doing, particularly when the distribution of the classes is unbalanced. It assesses the degree of agreement between the actual and anticipated categories, accounting for the likelihood that agreement could also occur.

(44)

where:

* + Po is the observed agreement, the ratio of instances that were correctly predicted by the model.
  + Pe is the expected agreement, the probability that the model's predictions and the true labels would agree by chance.

The observed agreement (Po) is computed by dividing the total number of instances by the sum of the diagonal members of the confusion matrix.

The product of the marginal probabilities of the true and predicted labels, added together for each class, is the expected agreement (Pe).

* Area Under the Receiver Operating Characteristic curve (AUC-ROC or simply AUC):

is a performance metric that's commonly used to address binary classification problems. The ROC curve visually illustrates the trade-off between true positive rate (sensitivity) and false positive rate (specificity) for different categorization model thresholds.

The area under this ROC curve, or AUC, is a single scalar statistic that summarizes the model's performance over a variety of classification criteria. A higher AUC usually indicates better discrimination between positive and negative cases.

* False Discovery Rate (FDR): is a statistical metric applied to hypothesis testing and binary classification. It shows the percentage of false positives, or inaccurate positive predictions, among all of a model's positive predictions. The expected ratio of false positives to all positive test results is known as the false positive ratio, or FDR, in the context of hypothesis testing.

False Discovery Rate is calculated using the following formula:

(45)

* False Negative Rate (FNR): sometimes referred to as the Miss Rate, is a binary classification metric that quantifies the percentage of actual positive cases that a model mistakenly predicts as negative. It has the following definition:

(46)

* False Positive Rate (FPR): Often called the False Alarm Rate or Fall-Out, this binary classification indicator measures the proportion of real negative events that a model incorrectly interprets as positive. It is defined as follows:

(47)

* Negative Predictive Value (NPV): is a binary classification statistic that quantifies the percentage of actual negative instances among those that a model predicts to be negative. It has the following definition:

(48)

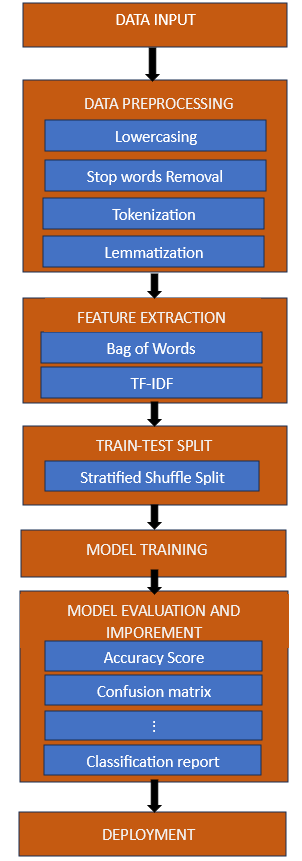


Figure 3.4 Architecture of fake news detection model.

This pipeline outlines the systematic approach in handling data for machine learning tasks. It starts with inputting raw data, followed by preprocessing steps such as lowering text, removing stop words, tokenizing, and lemmatizing the content. Feature extraction techniques like TF-IDF and Bag-of-Words transform the processed data into numerical representations. A Stratified Shuffle Split ensures balanced train-test splits, crucial for maintaining class proportions in classification tasks. The models are then trained on the training set. Evaluation metrics like accuracy score, confusion matrix, and classification reports assess model performance, allowing for iterative improvements. Finally, successful models are deployed for real-world applications, completing the cycle of transforming raw data into actionable insights.

# CHAPTER FOUR

# SYSTEM IMPLEMENTATION AND TESTING

## 4.0 Settings of Experiments

The Sci-kit learn module and the open-source python programming language was used to create the models used in the experiment. Jupyter notebook was used as the implementation and testing environment of choice. More information covering this is provided at a later section. Throughout this project, a variety of machine learning algorithms which can be categorized into traditional and ensemble classifiers were employed, including “lbfgs” Logistic Regression, “liblinear’’ Logistic Regression, “newton-cg” Logistic Regression, ‘sag’ Logistic Regression, Random Forest, Perceptron, Ridge Classifier, CatBoost, Nearest Centroid, Stochastic Gradient Decent(SGD), SVC(Kernel=“linear”, C=0.025), SVC(gama=2, C=1), LinearSVC, ZeroR, Decision Tree, Passive Aggressive, Extra Tree, Random Patches, Voting, Stacked Generalization, Multi-layer perceptron(MLP), Bernoulli RBM, AdaBoost, Gradient Boosting, Ordinal Learning Model(OLM), XGBoost, Decision Stump, Complement Naïve Bayes, Multinomial Naïve Bayes were thoroughly trained using our four datasets, and each algorithm produced outputs. Every algorithm that considered our performance metrics produced an output result that comprised the following, as was stated in chapter three: F1 Score, MCC, KAPPA, Accuracy, Specificity, Precision, Recall, FDR, FPR, FNR and NPV. We can determine which classifier performs best at recognizing fake news based on the accuracy levels.

## 4.1 Data Analysis for Dataset 1

Table 4.1 displays the outcome of all supervised machine learning algorithms used for Dataset 1 (Lifferth, 2018), Stacked Generalization which is an ensemble algorithm had the maximum accuracy score of 0.9805 overall. The highest accuracy score achieved by a traditional learning algorithm was attained by Ridge classifier with a value 0.9776. ZeroR performed the poorest overall, accuracy wise. ComplementNB achieved the highest overall precision score at 0.9959. With a precision value of 0.9863, Voting classifier was the highest ensemble model.

### **Table 4.1** Result of Dataset 1 (Lifferth, 2018)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | **Acc** | **Prec** | **Rec** | **Spec** | **F1 Score** | **MCC** | **KAPPA** | **AUC** | **FPR** | **FNR** | **FDR** | **NVP** |
| “lbfgs” LR | 0.9627 | 0.9599 | 0.9699 | 0.9595 | 0.9629 | 0.9254 | 0.9254 | 0.9627 | 0.0404 | 0.034 | 0.04 | 0.9656 |
| “liblinear” LR | 0.9627 | 0.9599 | 0.9659 | 0.9595 | 0.9629 | 0.9254 | 0.9254 | 0.9627 | 0.0404 | 0.034 | 0.04 | 0.9656 |
| “newton-cg” LR | 0.9627 | 0.9599 | 0.9659 | 0.9595 | 0.9629 | 0.9254 | 0.9254 | 0.9627 | 0.0404 | 0.034 | 0.04 | 0.9656 |
| “Sag” LR | 0.9627 | 0.9599 | 0.9659 | 0.9595 | 0.9629 | 0.9254 | 0.9254 | 0.9627 | 0.0404 | 0.034 | 0.04 | 0.9656 |
| Decision Stump | 0.7745 | 0.7242 | 0.8876 | 0.661 | 0.7976 | 0.5634 | 0.5488 | 0.7743 | 0.3389 | 0.1123 | 0.2757 | 0.8543 |
| Perceptron | 0.9668 | 0.9642 | 0.9697 | 0.9638 | 0.9669 | 0.9336 | 0.9336 | 0.9668 | 0.0361 | 0.0302 | 0.0357 | 0.9694 |
| RidgeClassifier | **0.9776** | 0.9756 | 0.9798 | 0.9754 | 0.9777 | 0.9552 | 0.9552 | 0.9776 | 0.0245 | 0.0201 | 0.0243 | 0.9796 |
| MultinomialNB | 0.8543 | 0.9959 | 0.7119 | 0.9971 | 0.8303 | 0.7395 | 0.7087 | 0.8545 | 0.0028 | 0.288 | 0.004 | 0.7753 |
| NearestCentroid | 0.8463 | 0.7874 | 0.9495 | 0.7428 | 0.8609 | 0.7079 | 0.6926 | 0.8462 | 0.2571 | 0.0504 | 0.2125 | 0.9362 |
| SGD | 0.9752 | 0.9741 | 0.9764 | 0.974 | 0.9753 | 0.9504 | 0.9504 | 0.9752 | 0.0259 | 0.0235 | 0.0258 | 0.9763 |
| SVC (kernel=”linear”, C=0.025) | 0.9016 | 0.8552 | 0.9673 | 0.8358 | 0.9078 | 0.8103 | 0.8033 | 0.9015 | 0.1641 | 0.0326 | 0.1447 | 0.9623 |
| SVC (gama=2, C=1) | 0.961 | 0.9549 | 0.9678 | 0.9542 | 0.9613 | 0.9221 | 0.9221 | 0.961 | 0.0457 | 0.0321 | 0.045 | 0.9673 |
| LinearSVC | 0.9771 | 0.9769 | 0.9774 | 0.9768 | 0.9772 | 0.9543 | 0.9543 | 0.9771 | 0.0231 | 0.0225 | 0.023 | 0.9773 |
| ZeroR | **0.5007** | 0.5007 | **1.0000** | 0.000 | 0.6673 | 0.0000 | 0.0000 | 0.5000 | 1.0000 | 0.0000 | 0.4992 | 0.0000 |
| DecisionTree | 0.9596 | 0.9614 | 0.9577 | 0.9614 | 0.9595 | 0.9192 | 0.9192 | 0.9596 | 0.0385 | 0.0422 | 0.0385 | 0.9577 |
| PassiveAggressive | 0.9762 | 0.9778 | 0.9745 | 0.9778 | 0.9761 | 0.9524 | 0.9524 | 0.9762 | 0.0221 | 0.0254 | 0.0221 | 0.9745 |
| ComplementNB | 0.8536 | **0.9959** | 0.7105 | 0.9971 | 0.8293 | 0.7383 | 0.7073 | 0.8538 | 0.0028 | 0.2894 | 0.004 | 0.7744 |
| CatBoost | 0.9737 | 0.9717 | 0.9759 | 0.9715 | 0.9738 | 0.9476 | 0.9475 | 0.9737 | 0.0284 | 0.024 | 0.0282 | 0.9758 |
| Voting | 0.9802 | **0.9863** | 0.9740 | 0.9865 | 0.9801 | 0.9606 | 0.9605 | 0.9802 | 0.0134 | 0.0259 | 0.0136 | 0.9743 |
| SG | **0.9805** | 0.9826 | 0.9783 | 0.9826 | 0.9805 | 0.961 | 0.961 | 0.9805 | 0.0173 | 0.0216 | 0.0173 | 0.9784 |
| MLP | 0.9764 | 0.981 | 0.9716 | 0.9812 | 0.9763 | 0.9529 | 0.9528 | 0.9764 | 0.0187 | 0.0283 | 0.0189 | 0.9718 |
| BernoulliRBM | 0.9774 | 0.9769 | 0.9779 | 0.9768 | 0.9774 | 0.9548 | 0.9548 | 0.9774 | 0.0231 | 0.022 | 0.023 | 0.9778 |
| AdaBoost | 0.9634 | 0.9595 | 0.9678 | 0.959 | 0.9636 | 0.9269 | 0.9269 | 0.9634 | 0.0409 | 0.0321 | 0.0404 | 0.9674 |
| GradientBoosting | 0.9673 | 0.9655 | 0.9692 | 0.9653 | 0.9674 | 0.9346 | 0.9346 | 0.9673 | 0.0346 | 0.0307 | 0.0344 | 0.969 |
| OLM | 0.9627 | 0.9599 | 0.9659 | 0.9595 | 0.9629 | 0.9254 | 0.9254 | 0.9627 | 0.0404 | 0.034 | 0.04 | 0.9656 |
| Xgboost | 0.9670 | 0.9637 | 0.9707 | 0.9634 | 0.9672 | 0.9341 | 0.9341 | 0.967 | 0.0365 | 0.0292 | 0.0362 | 0.9704 |
| Random Forest | 0.9331 | 0.9678 | 0.8963 | 0.9701 | 0.9307 | 0.8687 | 0.8663 | 0.9332 | 0.0298 | 0.1036 | 0.0321 | 0.9031 |
| Random Patches | 0.9728 | 0.9809 | 0.9644 | 0.9812 | 0.9726 | 0.9458 | 0.9456 | 0.9728 | 0.0187 | 0.0355 | 0.019 | 0.9649 |
| ExtraTree | **0.9276** | 0.9816 | **0.9836** | 0.68 | 0.9234 | 0.8607 | 0.8553 | 0.9277 | 0.0163 | 0.1281 | 0.0183 | 0.8844 |

LR—Logistic Regression. SGD—Stochastic Gradient Decent. SVC—Support Vector Machine. SG—StackedGeneralization. MLP—Multilayer Perceptron. OLM —Ordinal Learning Model.

Traditional Model

Ensemble Model

### **Figure 4.1: Three Evaluation Metrics' graphical representations in comparison to every supervised machine learning model used with Dataset 1**

Figure 4.1 gives the graphical representation of all supervised machine learning algorithms used on Dataset 1 plotted against their corresponding accuracy, precision, recall and specificity performance metrics.

## 4.2 Data Analysis for Dataset 2

With an accuracy score of 0.9991, Bernoulli RBM had the best level of accuracy among all the algorithms, according to the data in Table 4.2. Decision Stump with an accuracy rating of 0.9962, was the model with the highest value. ZeroR had the greatest recall value 1.0000, but its accuracy score was a pitiful 0.5229. The Bernoulli RBM, with a value of 0.9960, comes next. At 0.9432, Complement NB has the lowest recall value.

### **Table 4.2** Result of Dataset 2 (ISOT Fake News dataset, 2022)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | **Acc** | **Prec** | **Rec** | **Spec** | **F1 Score** | **MCC** | **KAPPA** | **AUC** | **FPR** | **FNR** | **FDR** | **NVP** |
| “lbfgs” LR | 0.9853 | 0.9869 | 0.985 | 0.9857 | 0.9859 | 0.9706 | 0.9706 | 0.9853 | 0.0142 | 0.0149 | 0.013 | 0.9836 |
| “liblinear” LR | 0.9853 | 0.9869 | 0.985 | 0.9857 | 0.9859 | 0.9706 | 0.9706 | 0.9853 | 0.0142 | 0.0149 | 0.013 | 0.9836 |
| “newton-cg” LR | 0.9853 | 0.9869 | 0.985 | 0.9857 | 0.9859 | 0.9706 | 0.9706 | 0.9853 | 0.0142 | 0.0149 | 0.013 | 0.9836 |
| “Sag” LR | 0.9853 | 0.9869 | 0.985 | 0.9857 | 0.9859 | 0.9706 | 0.9706 | 0.9853 | 0.0142 | 0.0149 | 0.013 | 0.9836 |
| Decision Stump | **0.9962** | **0.9992** | 0.9936 | 0.9991 | 0.9964 | 0.9925 | 0.9925 | 0.9964 | 0.0008 | 0.0063 | 0.0007 | 0.9931 |
| Perceptron | 0.9905 | 0.9893 | 0.9925 | 0.9883 | 0.9909 | 0.9809 | 0.9809 | 0.9904 | 0.0116 | 0.0074 | 0.0106 | 0.9917 |
| RidgeClassifier | 0.9940 | 0.9952 | 0.9932 | 0.9948 | 0.9942 | 0.988 | 0.988 | 0.994 | 0.0051 | 0.0067 | 0.0047 | 0.9926 |
| MultinomialNB | 0.9428 | 0.9448 | 0.9459 | 0.9394 | 0.9454 | 0.8855 | 0.8855 | 0.9427 | 0.0605 | 0.054 | 0.0551 | 0.9406 |
| NearestCentroid | 0.9375 | 0.9369 | 0.944 | 0.9303 | 0.9404 | 0.8747 | 0.8747 | 0.9371 | 0.0696 | 0.0559 | 0.063 | 0.938 |
| SGD | 0.9927 | 0.994 | 0.9921 | 0.9935 | 0.993 | 0.9855 | 0.9855 | 0.9928 | 0.0064 | 0.0078 | 0.0059 | 0.9913 |
| SVC (kernel=”linear”, C=0.025) | 0.9703 | 0.9761 | 0.9668 | 0.974 | 0.9714 | 0.9405 | 0.9405 | 0.9704 | 0.0259 | 0.0331 | 0.0238 | 0.964 |
| SVC (gama=2, C=1) | 0.9874 | 0.9877 | 0.9881 | 0.9865 | 0.9879 | 0.9747 | 0.9747 | 0.9873 | 0.0134 | 0.0118 | 0.0122 | 0.987 |
| LinearSVC | 0.9944 | 0.9944 | 0.9948 | 0.9939 | 0.9946 | 0.9888 | 0.9888 | 0.9944 | 0.006 | 0.0051 | 0.0055 | 0.9943 |
| ZeroR | **0.5229** | 0.5229 | **1.0000** | 0.0000 | 0.6867 | 0.0000 | 0.0000 | 0.5000 | 1.0000 | 0.0000 | 0.4770 | 0.0000 |
| DecisionTree | 0.9964 | 0.9968 | 0.9964 | 0.9965 | 0.9966 | 0.9929 | 0.9929 | 0.9964 | 0.0034 | 0.0035 | 0.0031 | 0.9961 |
| PassiveAggressive | 0.9944 | 0.9952 | 0.994 | 0.9948 | 0.9946 | 0.9888 | 0.9888 | 0.9944 | 0.0051 | 0.0059 | 0.0047 | 0.9935 |
| ComplementNB | 0.9439 | 0.9492 | 0.9432 | 0.9446 | 0.9462 | 0.8876 | 0.8876 | 0.9439 | 0.0553 | 0.0567 | 0.0507 | 0.9381 |
| CatBoost | 0.9971 | **0.9996** | 0.9948 | 0.9995 | 0.9972 | 0.9942 | 0.9942 | 0.9972 | 0.0004 | 0.0051 | 0.0003 | 0.9944 |
| Voting | 0.9983 | 0.9984 | 0.9984 | 0.9982 | 0.9984 | 0.9966 | 0.9966 | 0.9983 | 0.0017 | 0.0015 | 0.0015 | 0.9982 |
| SG | 0.9975 | 0.9972 | 0.998 | 0.9969 | 0.9976 | 0.995 | 0.995 | 0.9975 | 0.003 | 0.0019 | 0.0027 | 0.9978 |
| MLP | 0.9919 | 0.9952 | 0.9893 | 0.9948 | 0.9922 | 0.9839 | 0.9838 | 0.992 | 0.0051 | 0.0106 | 0.0047 | 0.9884 |
| BernoulliRBM | **0.9991** | 0.9996 | 0.9988 | 0.9995 | 0.9976 | 0.9976 | 0.9983 | 0.9991 | 0.0004 | 0.0011 | 0.0003 | 0.9987 |
| AdaBoost | 0.9979 | 0.9992 | 0.9968 | 0.9991 | 0.9983 | 0.9983 | 0.9983 | 0.9979 | 0.0008 | 0.0031 | 0.0007 | 0.9965 |
| GradientBoosting | 0.9971 | 0.9992 | 0.9952 | 0.9991 | 0.9979 | 0.9942 | 0.9942 | 0.9972 | 0.0008 | 0.0047 | 0.0007 | 0.9948 |
| OLM | 0.9853 | 0.9869 | 0.985 | 0.9857 | 0.9859 | 0.9706 | 0.9706 | 0.9853 | 0.0142 | 0.0149 | 0.013 | 0.9836 |
| Xgboost | 0.9987 | 0.9996 | 0.998 | 0.9995 | 0.9988 | 0.9975 | 0.9975 | 0.9987 | 0.0004 | 0.0019 | 0.0003 | 0.9978 |
| Random Forest | 0.9896 | 0.992 | 0.9881 | 0.9913 | 0.9901 | 0.9793 | 0.9793 | 0.9897 | 0.0086 | 0.0118 | 0.0079 | 0.987 |
| Random Patches | 0.9975 | 0.9992 | 0.996 | 0.9991 | 0.9976 | 0.995 | 0.995 | 0.9975 | 0.0008 | 0.0039 | 0.0007 | 0.9956 |
| ExtraTree | **0.9797** | 0.9915 | 0.9696 | 0.9909 | 0.9804 | 0.9597 | 0.9595 | 0.9802 | 0.009 | 0.0303 | 0.0084 | 0.9674 |

LR—Logistic Regression. SGD—Stochastic Gradient Decent. SVC—Support Vector Machine. SG—StackedGeneralization. MLP—Multilayer Perceptron. OLM —Ordinal Learning Model.

### **Figure 4.2: Three Evaluation Metrics' graphical representations in comparison to every supervised machine learning model used with Dataset 2**

All of the supervised machine learning algorithms that were applied to Dataset 2 are graphically represented in Figure 4.2, where they are displayed against the appropriate performance measures for accuracy, precision, recall, and specificity.

## 4.3 Data Analysis for Dataset 3

Out of all the algorithms utilized, the Stacked generalization algorithm had the best accuracy and the fourth-highest precision scores. In terms of precision, Random Patches has the greatest value.

In terms of recall, the Stacked generalization classifier algorithm appears to be the best choice, with a value of 0.9707. With ZeroR, the lowest accuracy of 0.5143 is attained. With a precision value of 0.0000, ZeroR produced the lowest result. Additionally, ZeroR was also thought to be the lowest, having the lowest recall score 0.0000.

### **Table 4.3** Result of Dataset 3 (Verma, Agrawal, & Prodan, 2021)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | **Acc** | **Prec** | **Rec** | **Spec** | **F1 Score** | **MCC** | **KAPPA** | **AUC** | **FPR** | **FNR** | **FDR** | **NVP** |
| “lbfgs” LR | 0.9615 | 0.9657 | 0.9547 | 0.968 | 0.9602 | 0.9231 | 0.9231 | 0.9614 | 0.0319 | 0.0452 | 0.0342 | 0.9577 |
| “liblinear” LR | 0.9615 | 0.9615 | 0.9615 | 0.9679 | 0.9229 | 0.9229 | 0.9229 | 0.9613 | 0.032 | 0.0452 | 0.0343 | 0.9577 |
| “newton-cg” LR | 0.9615 | 0.9615 | 0.9547 | 0.9679 | 0.9613 | 0.923 | 0.9229 | 0.9613 | 0.032 | 0.0452 | 0.0343 | 0.9577 |
| “Sag” LR | 0.9615 | 0.9615 | 0.9547 | 0.968 | 0.9602 | 0.9231 | 0.9231 | 0.9614 | 0.0319 | 0.0452 | 0.0342 | 0.9577 |
| Decision Stump | 0.8064 | 0.9771 | 0.6157 | 0.9863 | 0.7554 | 0.653 | 0.6084 | 0.801 | 0.0136 | 0.3842 | 0.0228 | 0.7311 |
| Perceptron | 0.9682 | 0.9682 | 0.9682 | 0.9696 | 0.9672 | 0.9364 | 0.9364 | 0.9682 | 0.0303 | 0.0332 | 0.0321 | 0.9686 |
| RidgeClassifier | 0.9728 | 0.9792 | 0.9646 | 0.9807 | 0.9718 | 0.9458 | 0.9457 | 0.9726 | 0.0192 | 0.0353 | 0.0207 | 0.967 |
| MultinomialNB | 0.8823 | 0.8622 | 0.9017 | 0.864 | 0.8816 | 0.7656 | 0.7648 | 0.8829 | 0.1359 | 0.0982 | 0.1377 | 0.903 |
| NearestCentroid | 0.8662 | 0.8899 | 0.8267 | 0.9035 | 0.8267 | 0.7334 | 0.7316 | 0.8651 | 0.0964 | 0.1732 | 0.11 | 0.8466 |
| SGD | 0.9606 | 0.9680 | 0.9504 | 0.9703 | 0.9591 | 0.9214 | 0.9212 | 0.9604 | 0.0296 | 0.0495 | 0.0319 | 0.954 |
| SVC (kernel=”linear”, C=0.025) | 0.9209 | 0.9477 | 0.8860 | 0.9539 | 0.9159 | 0.8432 | 0.8415 | 0.92 | 0.046 | 0.0418 | 0.0522 | 0.8986 |
| SVC (gama=2, C=1) | 0.9683 | 0.9762 | 0.9581 | 0.978 | 0.9671 | 0.9368 | 0.9367 | 0.9681 | 0.0219 | 0.0418 | 0.0237 | 0.9611 |
| LinearSVC | **0.9748** | **0.9800** | 0.9678 | 0.9814 | 0.9739 | 0.9496 | 0.9496 | 0.9746 | 0.0185 | 0.0321 | 0.0199 | 0.97 |
| ZeroR | **0.5143** | 0.0000 | 0.0000 | 1.0000 | 0.0000 | 0.0000 | 0.0000 | 0.5000 | 0.0000 | 1.0000 | 0.0000 | 0.5143 |
| DecisionTree | 0.9456 | 0.9513 | 0.9359 | 0.9548 | 0.9435 | 0.8912 | 0.8911 | 0.9453 | 0.0451 | 0.064 | 0.0486 | 0.9404 |
| PassiveAggressive | 0.9726 | 0.9747 | **0.9688** | 0.9762 | 0.9717 | 0.9453 | 0.9453 | 0.9725 | 0.0237 | 0.0311 | 0.0252 | 0.9707 |
| ComplementNB | 0.8807 | 0.8563 | 0.9065 | 0.8564 | 0.8807 | 0.7629 | 0.7617 | 0.8814 | 0.1435 | 0.0934 | 0.1436 | 0.9065 |
| CatBoost | 0.9586 | 0.9722 | 0.9417 | 0.9746 | 0.9567 | 0.9176 | 0.9172 | 0.9582 | 0.0253 | 0.0582 | 0.0277 | 0.9466 |
| Voting | 0.9703 | 0.9755 | 0.963 | 0.9772 | 0.9692 | 0.9406 | 0.9405 | 0.9701 | 0.0227 | 0.0369 | 0.0244 | 0.9655 |
| SG | **0.9755** | 0.9787 | **0.9707** | 0.9800 | 0.9747 | 0.9510 | 0.9510 | 0.9753 | 0.0199 | 0.0292 | 0.0212 | 0.9725 |
| MLP | 0.9676 | 0.9714 | 0.9617 | 0.9733 | 0.9665 | 0.9353 | 0.9353 | 0.9675 | 0.0266 | 0.0382 | 0.0285 | 0.9642 |
| BernoulliRBM | 0.9583 | 0.97 | 0.9433 | 0.9725 | 0.9565 | 0.9168 | 0.9165 | 0.9579 | 0.0274 | 0.0566 | 0.0299 | 0.9478 |
| AdaBoost | 0.9460 | 0.953 | 0.935 | 0.9564 | 0.9439 | 0.8921 | 0.892 | 0.9457 | 0.0435 | 0.0649 | 0.0469 | 0.9397 |
| GradientBoosting | 0.9508 | 0.967 | 0.9304 | 0.97 | 0.9484 | 0.9021 | 0.9015 | 0.9502 | 0.0299 | 0.0695 | 0.0329 | 0.9366 |
| OLM | 0.9615 | 0.9656 | 0.9547 | 0.9679 | 0.9601 | 0.923 | 0.9229 | 0.9613 | 0.032 | 0.0452 | 0.0343 | 0.9577 |
| Xgboost | 0.9494 | 0.9679 | 0.9266 | 0.971 | 0.9468 | 0.8995 | 0.8987 | 0.9488 | 0.0289 | 0.0733 | 0.032 | 0.9334 |
| Random Forest | **0.9415** | 0.9499 | 0.9286 | 0.9537 | 0.9391 | 0.8831 | 0.8829 | 0.9412 | 0.0462 | 0.0713 | 0.05 | 0.934 |
| Random Patches | 0.9599 | **0.9804** | 0.9361 | 0.9823 | 0.9577 | 0.9205 | 0.9197 | 0.9592 | 0.0176 | 0.0638 | 0.0195 | 0.9422 |
| ExtraTree | 0.9418 | 0.9299 | 0.9518 | 0.9323 | 0.9408 | 0.8839 | 0.8836 | 0.9421 | 0.0676 | 0.0481 | 0.07 | 0.9535 |

LR—Logistic Regression. SGD—Stochastic Gradient Decent. SVC—Support Vector Machine. SG—StackedGeneralization. MLP—Multilayer Perceptron. OLM —Ordinal Learning Model.

### **Figure 4.3: Three Evaluation Metrics' graphical representations in comparison to every supervised machine learning model used with Dataset 3**

Figure 4.3 shows a graphic representation of all the supervised machine learning methods that were used on Dataset 3, arranged in relation to the relevant accuracy, precision, recall, and specificity performance metrics.

## 4.4 Data Analysis for Dataset 4

Stack generalization algorithm had the highest accuracy score of all the algorithms used. The next four machine learning algorithms, "lbfgs," "liblinear," "newton-cg," and "Sag," Logistic Regression all had the same accuracy rating of 0.6088. According to data in Table 4.4, the voting classifier's accuracy was the lowest. SVC (gama=2, C=1) had the maximum precision score of 0.6053. Gradient Boosting comes in second with a score of 0.5943 and Multinomial NB comes in third with a number of 0.5994. With a score of 0.0000, SVC (kernel="linear," C=0.025), ZeroR, and Decision Stump obtained the lowest results.

The three models with the lowest recall scores were MLP, ExtraTree, and Random Forest, at 0.3797, 0.3619, and 0.3530, respectively. With a score of 0.68, the Bernoulli RBM performed the best, followed by the Nearest Centroid algorithm (0.6269) and the Decision Tree (0.5178).

### **Table 4.4** Result of Dataset 4 (Wang W. Y., 2017)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | **Acc** | **Prec** | **Rec** | **Spec** | **F1 Score** | **MCC** | **KAPPA** | **AUC** | **FPR** | **FNR** | **FDR** | **NVP** |
| “lbfgs” LR | **0.6088** | 0.5763 | 0.4075 | 0.766 | 0.4774 | 0.1863 | 0.1794 | 0.5868 | 0.2339 | 0.5924 | 0.4236 | 0.6234 |
| “liblinear” LR | **0.6088** | 0.5763 | 0.4075 | 0.766 | 0.4774 | 0.1863 | 0.1794 | 0.5868 | 0.2339 | 0.5924 | 0.4236 | 0.6234 |
| “newton-cg” LR | **0.6088** | 0.5757 | 0.4064 | 0.766 | 0.4765 | 0.1851 | 0.1782 | 0.5862 | 0.2339 | 0.5935 | 0.4242 | 0.623 |
| “Sag” LR | **0.6083** | 0.5757 | 0.4064 | 0.766 | 0.4765 | 0.1851 | 0.1782 | 0.5862 | 0.2339 | 0.5935 | 0.4242 | 0.623 |
| Decision Stump | 0.5615 | 0.0000 | 0.0000 | 1.0000 | 0.0000 | 0.0000 | 0.0000 | 0.5000 | 0.0000 | 1.0000 | 0.0000 | 0.5615 |
| Perceptron | 0.5766 | 0.518 | 0.4944 | 0.6408 | 0.5059 | 0.136 | 0.1359 | 0.5676 | 0.3591 | 0.5055 | 0.4819 | 0.6188 |
| RidgeClassifier | 0.5961 | 0.5456 | 0.4721 | 0.693 | 0.5062 | 0.1689 | 0.1676 | 0.5826 | 0.3069 | 0.5278 | 0.4543 | 0.627 |
| MultinomialNB | 0.5947 | 0.5994 | 0.2282 | 0.8808 | 0.3306 | 0.1452 | 0.117 | 0.5545 | 0.1191 | 0.7717 | 0.4005 | 0.5937 |
| NearestCentroid | 0.5834 | 0.5208 | **0.6269** | 0.5495 | 0.5689 | 0.1754 | 0.1726 | 0.5882 | 0.4504 | 0.373 | 0.4791 | 0.6535 |
| SGD | 0.6049 | 0.5610 | 0.4554 | 0.7217 | 0.5027 | 0.1836 | 0.1809 | 0.5885 | 0.2782 | 0.5445 | 0.4389 | 0.6292 |
| SVC (kernel=”linear”, C=0.025) | 0.5615 | 0.0000 | 0.0000 | 1.0000 | 0.0000 | 0.0000 | 0.0000 | 0.5000 | 0.0000 | 1.0000 | 0.0000 | 0.5615 |
| SVC (gama=2, C=1) | 0.6000 | **0.6053** | 0.2527 | 0.8713 | 0.3566 | 0.1592 | 0.1325 | 0.562 | 0.1286 | 0.7472 | 0.3946 | 0.5989 |
| LinearSVC | 0.5859 | 0.5306 | 0.4821 | 0.6669 | 0.5052 | 0.1511 | 0.1506 | 0.5745 | 0.333 | 0.5178 | 0.4693 | 0.6225 |
| ZeroR | 0.5615 | 0.0000 | 0.0000 | 1.0000 | 0.0000 | 0.0000 | 0.0000 | 0.5000 | 0.0000 | 1.0000 | 0.0000 | 0.5615 |
| DecisionTree | **0.5556** | 0.4936 | 0.5178 | 0.5852 | 0.5054 | 0.1025 | 0.1024 | 0.5515 | 0.4147 | 0.4821 | 0.5063 | 0.6084 |
| PassiveAggressive | 0.5561 | 0.4939 | 0.4977 | 0.6017 | 0.4958 | 0.0994 | 0.0994 | 0.5497 | 0.3982 | 0.5022 | 0.506 | 0.6054 |
| ComplementNB | 0.6069 | 0.5686 | 0.4287 | 0.746 | 0.4888 | 0.1843 | 0.1796 | 0.5874 | 0.2539 | 0.5712 | 0.4313 | 0.6258 |
| CatBoost | 0.5937 | 0.5816 | 0.2616 | 0.853 | 0.3609 | 0.143 | 0.122 | 0.5573 | 0.1469 | 0.7383 | 0.4183 | 0.5967 |
| Voting | **0.5498** | 0.4866 | 0.4855 | 0.6 | 0.486 | 0.0855 | 0.0855 | 0.5427 | 0.4000 | 0.5144 | 0.5133 | 0.5989 |
| SG | **0.6123** | 0.5878 | 0.3875 | 0.7878 | 0.4671 | 0.1919 | 0.1821 | 0.5876 | 0.2121 | 0.6124 | 0.4121 | 0.6222 |
| MLP | 0.5571 | 0.4950 | 0.3797 | 0.6008 | 0.498 | 0.1018 | 0.1018 | 0.5509 | 0.3991 | 0.4988 | 0.5049 | 0.6066 |
| BernoulliRBM | 0.5947 | 0.5553 | **0.6800** | 0.7626 | 0.451 | 0.1541 | 0.1474 | 0.5711 | 0.2373 | 0.6202 | 0.4446 | 0.6115 |
| AdaBoost | 0.5908 | 0.5877 | 0.2238 | 0.8773 | 0.3241 | 0.1346 | 0.1085 | 0.5506 | 0.1226 | 0.7761 | 0.4122 | 0.5914 |
| GradientBoosting | 0.5942 | **0.5943** | 0.2349 | 0.8747 | 0.3367 | 0.1438 | 0.1175 | 0.5548 | 0.1252 | 0.765 | 0.4056 | 0.5942 |
| OLM | 0.6083 | 0.5757 | 0.4064 | 0.766 | 0.4765 | 0.1851 | 0.1782 | 0.5862 | 0.2339 | 0.5935 | 0.4242 | 0.623 |
| Xgboost | 0.5942 | **0.5943** | 0.2349 | 0.8747 | 0.3367 | 0.1438 | 0.1175 | 0.5548 | 0.1252 | 0.765 | 0.4056 | 0.5942 |
| Random Forest | 0.6044 | 0.5805 | 0.353 | 0.8008 | 0.439 | 0.1726 | 0.1607 | 0.5769 | 0.1991 | 0.6469 | 0.4194 | 0.6131 |
| Random Patches | 0.5698 | 0.5149 | 0.3251 | 0.7606 | 0.3986 | 0.0954 | 0.0896 | 0.543 | 0.2393 | 0.6748 | 0.485 | 0.5905 |
| ExtraTree | 0.5825 | 0.5354 | 0.3619 | 0.7547 | 0.4318 | 0.1268 | 0.1209 | 0.5583 | 0.2452 | 0.638 | 0.4645 | 0.6023 |

LR—Logistic Regression. SGD—Stochastic Gradient Decent. SVC—Support Vector Machine. SG—StackedGeneralization. MLP—Multilayer Perceptron. OLM —Ordinal Learning Model

### **Figure 4.4: Three Evaluation Metrics' graphical representations in comparison to every supervised machine learning model used with Dataset 4**

Figure 4.4 shows a graphic representation of all the supervised machine learning methods that were used on Dataset 4, arranged in relation to the relevant accuracy, precision, recall, and specificity performance metrics.

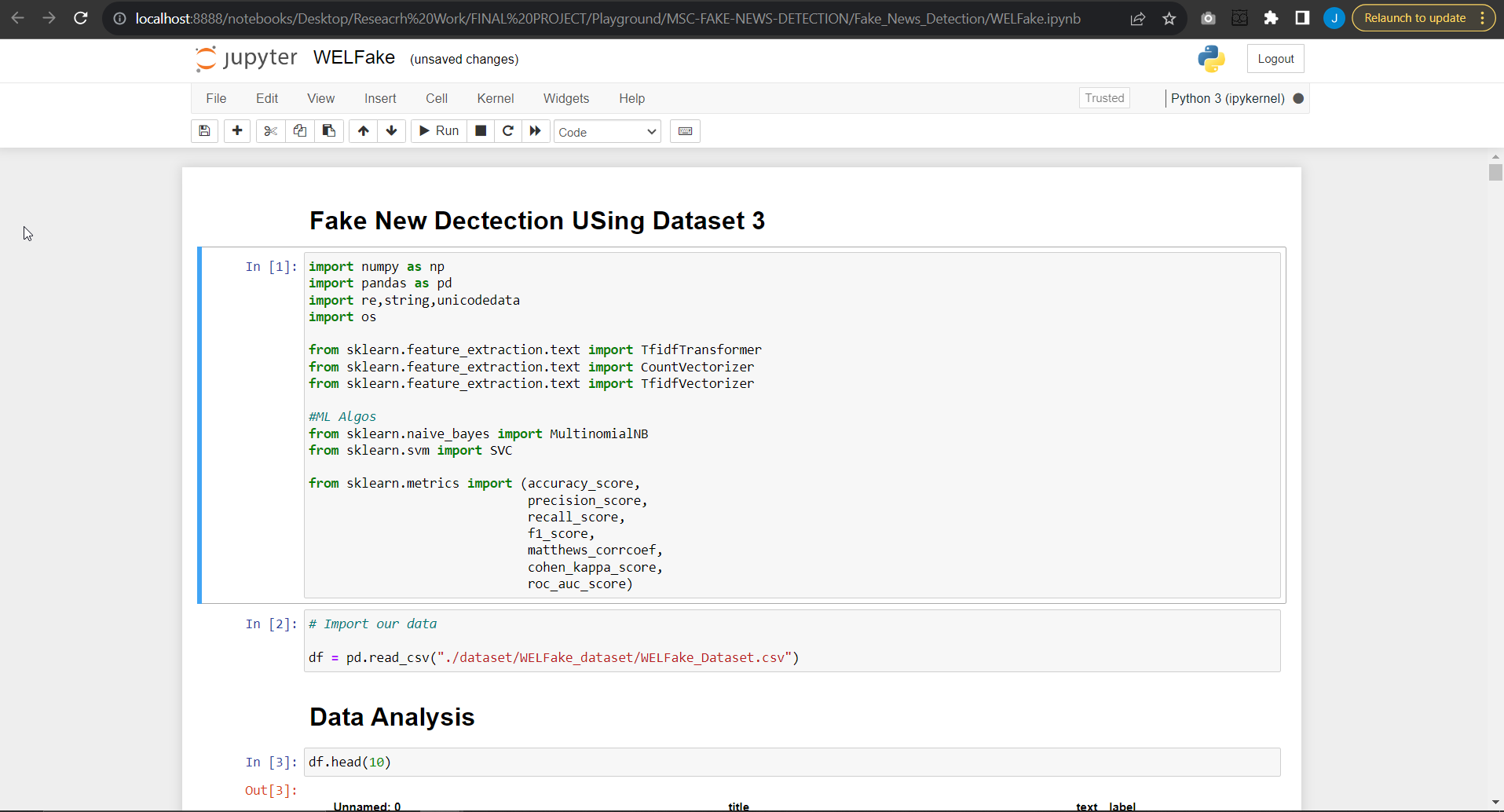
## 4.5 THE PROGRAMMING LANGUAGE SELECTION

Python was the programming language select for the execution of this project. Python is a robust and simple-to-learn programming language. Its object-oriented programming methodology is straightforward but efficient, and its high-level data structures are efficient. Python's interpreted nature, dynamic typing, and beautiful syntax make it a perfect language for scripting and quick application development across a wide range of platforms. Many machine learning applications are built on top of the language, which mostly makes use of widely used libraries and frameworks. High-level neural network building is made easier with Keras, TensorFlow and PyTorch are commonly used for deep learning applications, and Scikit-learn offers effective tools for modelling and data analysis.

Python has libraries devoted to machine learning in addition to providing necessary tools for data analysis and manipulation. While NumPy provides numerical operations on big, multi-dimensional arrays, Pandas makes it easier to manipulate and analyze datasets efficiently. Matplotlib, Seaborn, and Plotly are three tools for data visualization that provide a variety of choices for producing static, animated, and interactive visualizations.

## 4.6 SOFTWARE AND HARDWARE REQUIREMENTS

To execute the source code for all implementations, it is necessary to install ANACONDA Navigator as it also installs JUPYTER Notebook, an important software for running python code through an interactive computing environment on your web browser. This notebook makes it possible to create and edit documents that explain the data analysis process in a way that is understandable to humans. A laptop with at least an Intel Core i3 processor or higher, 6 GB of RAM, should be suitable for carrying out needed task. Do note that your hardware require might grow based on certain computing tasks.



### Figure4.5 Jupiter Notebook Workspace

# CHAPTER FIVE

# CONCLUSION AND RECOMMENDATION

## 5.1 Conclusion

The issue of "fake news" is one of the primary issues with the development of technology (the Internet), social media, and other online activities. As was covered in chapter two, it can be used by evildoers for evil intentions. In order to identify fake news on social media, this study suggests a method that combines text mining with supervised AI algorithms. Text mining analysis and supervised artificial intelligence algorithms have been the subject of independent research. This combination model was tested using four different real-world data sets, and the outcomes were examined using the following metrics: MCC, KAPPA, F1 score, accuracy, specificity, recall, precision, FPR, FNR, FDR, and NVP.

A fake news detection model was constructed to identify fake news across four different datasets, which we then used to train our machine learning classifiers for determining whether a news item is authentic or fake. Extensive experiments were used to train all supervised machine learning techniques, such as “lbfgs” Logistic Regression, “liblinear’’ Logistic Regression, “newton-cg” Logistic Regression, ‘sag’ Logistic Regression, Random Forest, Perceptron, Ridge Classifier, CatBoost, Nearest Centroid, Stochastic Gradient Decent(SGD), SVC(Kernel=“linear”, C=0.025), SVC(gama=2, C=1), LinearSVC, ZeroR, Decision Tree, Passive Aggressive, Extra Tree, Random Patches, Voting, Stacked Generalization, Multi-layer perceptron(MLP), Bernoulli RBM, AdaBoost, Gradient Boosting, Ordinal Learning Model(OLM), XGBoost, Decision Stump, Complement Naïve Bayes, Multinomial Naïve Bayes, were trained through extensive experiments. Tables 4.1–4.4 provide detailed results about their performance on 4 different datasets.

As seen in the results presented in chapter 4, in all four different datasets, the ensemble learning techniques employed in this project, performed the best accuracy wise.

## 5.2 Contribution to Knowledge and Future Recommendations.

For this experiment, 29 machine learning algorithms were applied with four distinct social media datasets. This project also included mathematical and visual representations that illustrate the relationship between the dataset. Future research could enhance the current work by incorporating the detection of bogus news using both image and video sources.

# References

Adriani, R. (2019, 4 30). The Evolution of Fake News and the Abuse of Emerging Technologies. *European Journal of Social Sciences*, 18-24. doi:https://doi.org/10.26417/ejss-2019.v2i1-53

Azeez, N. A., & Fadhal, E. (2023). Classification of Virtual Harassment on Social Networks Using Ensemble Learning Techniques. *Applied sciences, 13*(7). doi:https://www.mdpi.com/2076-3417/13/7/4570

Azeez, N. A., Idiakose, S. O., Onyema, C. J., & Vyver, C. V. (2021, June 21). Cyberbullying Detection in Social Networks: Artificial Intelligence Approach. *Journal of Cyber Security and Mobility*. doi:https://doi.org/10.13052/jcsm2245-1439.1046

Banoula, M. (2023, May 10). *What is Perceptron: A Beginners Guide for Perceptron*. Retrieved from simplilearn: https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron#:~:text=Perceptron%20is%20a%20function%20that,dependence%20on%20the%20input%20value)

Behera, B., & Kumaravelan, G. (2020, May). Performance evaluation of Machine learning algorithms in Biomedical Document Classification. *International Journal of Advanced Science and Technology, 29*(5), 5704-5716.

Benamira, A., Devillers, B., Lesot, E., Ray, A. K., Saadi, M., & Malliaros, F. D. (2019). Semi-supervised learning and graph neural networks for fake news detection. *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (pp. 568–569). IEEE/ACM. doi:https://doi.org/10.1145/3341161.3342958

Chai, C. P. (2022). Comparison of text preprocessing methods. *Natural Language Engineering, 29*(3), 509-553. doi:https://doi.org/10.1017/s1351324922000213

Dodiya, T. (2021). Using term frequency - inverse document frequency to find the relevance of words in gujarati language. *International Journal for Research in Applied Science and Engineering Technology, 9*(4), 378-381. doi:https://doi.org/10.22214/ijraset.2021.33625

Endsley, M. R. (2018, October 30). Combating Information Attacks in the Age of the Internet: New Challenges for Cognitive Engineering. *Sage Journals*, 1081-1094. doi:https://doi.org/10.1177/0018720818807357

engati. (2021). *Ridge Regression*. Retrieved from engati: https://www.engati.com/glossary/ridge-regression

*Fake News Dataset*. (2020). Retrieved from https://www.kaggle.com/c/fake-news/data

Galli, A., Masciari, E., Moscato, V., & Sperlí, G. (2022). A comprehensive benchmark for fake news detection. *Journal of Intelligent Information Systems, 59*(1), 237-261. doi:https://doi.org/10.1007/s10844-021-00646-9

Gifu, D. (2023). An Intelligent System for Detecting Fake News. *Elsevier, 221*, 1058-1065. doi:https://doi.org/10.1016/j.procs.2023.08.088

Hussein, F., & Hejase, H. J. (2022). Artificial Intelligence and Fake News. *Vilnius University Press*, 39-71. doi:https://doi.org/10.15388/ZT/JR.2022.2

*ISOT Fake News dataset*. (2022, 11 27). Retrieved from onlineacademiccommunity: https://onlineacademiccommunity.uvic.ca/isot/2022/11/27/fake-news-detection-datasets/

kaggle. (2021). *kaggle\_parsed*.

Kaliyar, R. K., Goswami, A., & Narang, P. (2020). DeepFakE: improving fake news detection using tensor decomposition-based deep neural network. *The Journal of Supercomputing, 77*(2), 1015-1037. doi:https://doi.org/10.1007/s11227-020-03294-y

Kanade, V. (2022, April 18). *What Is Logistic Regression? Equation, Assumptions, Types, and Best Practices*. Retrieved from Spiceworks: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/

Khan, N. Z., & Yadav, S. R. (2019). Analysis of text classification algorithms. *International Journal of Trend in Scientific Research and Development: A Review, 3*(2), 579-581. doi:https://doi.org/10.31142/ijtsrd21448

Lamprou, E., Antonopoulos, N., Anomeritou, I., & Apostolou, C. (2021, July 13). Characteristics of Fake News and Misinformation in Greece: The Rise of New Crowdsourcing-Based Journalistic Fact-Checking Models. *MDPI Open Access Journals*, 417-439. doi:https://doi.org/10.3390/journalmedia2030025

Lifferth, W. (2018). *Fake News*. Retrieved from Kaggle: https://kaggle.com/competitions/fake-news

liguides. (2022, August 2). *Fake News*. Retrieved from TRU Libraries: https://libguides.tru.ca/fakenews/characteristics

Liu, Y., & Wu, Y.-F. (2018). Early Detection of Fake News on Social Media Through Propagation Path Classification with Recurrent and Convolutional Networks. *AAAI Conference*, *32.* New Orleans, Louisiana, USA. doi:https://doi.org/10.1609/aaai.v32i1.11268

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. *WWW '22: Proceedings of the ACM Web Conference 2022* (pp. 1148–1158). Association for Computing Machinery. doi:https://doi.org/10.1145/3485447.3512163

Monsees, L. (2021, May 26). Information disorder, fake news and the future of democracy. *Globalizations, 20*(1), 153-168. doi:https://doi.org/10.1080/14747731.2021.1927470

Murayama, T., Wakamiya, S., & Aramaki, E. (2021). Mitigation of Diachronic Bias in Fake News Detection Dataset. *Proceedings of the Seventh Workshop on Noisy User-Generated Text (W-Nut 2021)* (pp. 182–188). Association for Computational Linguistics. doi:https://doi.org/10.18653/v1/2021.wnut-1.21

Mwangi, E. (2023, May 29). Technology and Fake News: Shaping Social, Political, and Economic Perspectives. p. 37. doi:https://dx.doi.org/10.2139/ssrn.4462727

Naseem, U., Razzak, I., Khan, S. K., & Prasad, M. (2020). A Comprehensive Survey on Word Representation Models: From Classical to State-Of-The-Art Word Representation Language Models. *Association for Computing Machinery*. doi:https://doi.org/10.48550/arXiv.2010.15036

Özbay, F. A., & Alataş, B. (2020). Fake news detection within online social media using supervised artificial intelligence algorithms. *Physica A: Statistical Mechanics and Its Applications*. doi:https://doi.org/10.1016/j.physa.2019.123174

*Parliamentary question*. (2018, 8 23). Retrieved from European Parliament: https://www.europarl.europa.eu/doceo/document/E-8-2018-004331\_EN.html

Posetti, J., & Matthews, A. (2018). A short guide to the history of ’fake news’ and disinformation. *International Center for Journalist*, 2-15. Retrieved from https://www.icfj.org/sites/default/files/2018-07/A%20Short%20Guide%20to%20History%20of%20Fake%20News%20and%20Disinformation\_ICFJ%20Final.pdf

Qader, W. A., Ameen, M. M., & Ahmed, B. I. (2019). An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges. *International Engineering Conference (IEC)* (pp. 200-204). Erbil, Iraq: IEEE. doi:10.1109/IEC47844.2019.8950616

Qureshi, K. A., Malick, R. A., Sabih, M., & Cherifi, H. (2021). Complex Network and Source Inspired COVID-19 Fake News Classification on Twitter. *IEEE Access*, 139636-139656. doi:https://doi.org/10.1109/access.2021.3119404

Safieddine, F., & Hammad, R. (2020). Fake News: Origins and Political Impact. In *Handbook of Research on Recent Developments in Internet Activism and Political Participation* (pp. 103-212). IGI Global. doi:https://doi.org/10.4018/978-1-7998-4796-0.ch007

Sahoo, S. R., & Gupta, B. B. (2021). Multiple features based approach for automatic fake news detection on social networks using deep learning. *Applied Soft Computing, 100*. doi:https://doi.org/10.1016/j.asoc.2020.106983

Saini, A. (2022, August 26). *An Introduction to Random Forest Algorithm for beginners*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-random-forest-algorithm-for-beginners/

Saini, A. (2023, October 27). *Guide on Support Vector Machine (SVM) Algorithm*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks, 20*(1), 61-80. doi:https://doi.org/10.1109/tnn.2008.2005605

Shahid, W., Li, Y., Staples, D., Amin, G., Hakak, S., & Ghorbani, A. A. (2022). Are You a Cyborg, Bot or Human?—A Survey on Detecting Fake News Spreaders. *IEEE Access, 10*, 27069-27083. doi:https://doi.org/10.1109/access.2022.3157724

Shaina, R., & Chen, D. (2022). Fake news detection based on news content and social contexts: a transformer-based approach. *International Journal of Data Science and Analytics, 13*(4), 335-362. doi:https://doi.org/10.1007/s41060-021-00302-z

Shu, K., Zhou, X., Wang, S., Zafarani, R., & Liu, H. (2019). The role of user profiles for fake news detection. *ASONAM '19: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, (pp. 436–439). doi:https://doi.org/10.1145/3341161.3342927

Srikumar, V. (2017). An algebra for feature extraction. *Proceedings of the 55th Annual Meeting of the Association For* (pp. 1891–1900). Association for Computational Linguistics. doi:https://doi.org/10.18653/v1/p17-1173

statista. (2023). *false-information-topics-worldwide.*

Tembhurne, J. V., Almin, M. M., & Diwan, T. (2022). Mc-DNN. *International Journal on Semantic Web and Information Systems, 18*(1), 1-20. doi:https://doi.org/10.4018/ijswis.295553

Townsend, K. (2017, June 15). *Fake News: Methods, Motivations and Countermeasures*. Retrieved from SecurityWeek: https://www.securityweek.com/fake-news-methods-motivations-and-countermeasures/#:~:text=Trend%20Micro%20sees%20three%20current,beliefs%20or%20some%20other%20opinion.

*UO Libraries*. (2023, September 8). Retrieved from Fake News and Information Literacy: https://researchguides.uoregon.edu/fakenews/issues/defining#s-lg-box-22575759

Verma, P. K., Agrawal, P., & Prodan, R. (2021, February 25). *WELFake dataset for fake news detection in text data*. doi:10.5281/zenodo.4561253

Wang, J., Mao, H., & Li, H. (2022). FMFN: Fine-Grained Multimodal Fusion Networks for Fake News Detection. *Applied Sciences, 12*(3). doi:https://doi.org/10.3390/app12031093

Wang, W. Y. (2017). *LIAR*. Retrieved from Paperwithcode: https://paperswithcode.com/dataset/liar

wikipedia. (2023, September 8). *Fake news*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Fake\_news#:~:text=Fake%20news%20often%20has%20the,reports%20in%20newspapers%20were%20common.

Wu, J., & Ye, X. (2023). Fakeswarm: improving fake news detection with swarming characteristics. *Natural Language Processing and Machine Learning*. doi:https://doi.org/10.5121/csit.2023.130815

Zhang, J., Dong, B., & Yu, P. S. (2020). FakeDetector: Effective Fake News Detection with Deep Diffusive Neural Network. *2020 IEEE 36th International Conference on Data Engineering (ICDE)* (pp. 1826-1829). Dallas, TX, USA: IEEE. doi:10.1109/icde48307.2020.00180

Zhu, Y., Sheng, Q., Cao, J., Li, S., Wang, D., & Zhuang, F. (2022). Generalizing to the Future: Mitigating Entity Bias in Fake News Detection. *SIGIR '22: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information* (pp. 2120–2125). Association for Computing Machinery. doi:https://doi.org/10.1145/3477495.3531816

(2023). Retrieved from Cambridge Dictionary: https://dictionary.cambridge.org/dictionary/english/fake-news