

# UNIVERSITY OF LONDON

## INTERNATIONAL PROGRAMMES

### BSc Computer Science and Related Subjects



### CM3070 PROJECT

### FINAL PROJECT REPORT

**Multimodal VeriTruth Detection**

**Natural Language Processing**

**Author: Logesh Kiran**

**Student Number : 210201255**

**Date of Submission: 12/3/2024**

*Note: Report follows word count and not page count*

## **Final Year Project Report**

<b>Chapter 1: Introduction</b>	<b>4</b>
1.1 Project Details	4
1.2 Project Concept	4
1.3 Motivation	4
1.4 Research Questions	5
1.5 Deliverables and Justifications	5
<b>Chapter 2: Literature Review</b>	<b>6</b>
2.1 Rising crisis of Fake news, phishing and fraudulent activities	6
2.2 Increasing need for Natural Language Processing in Fake News Prevention	7
2.3 Harnessing the power of BERT model for text classification	8
2.4 Utilising Recurrent Neural Networks (RNNs) for optimised text classification	9
2.5 Summary of Literature Review	10
<b>Chapter 3: Project Design</b>	<b>11</b>
3.1 Domain and Users	11
3.2 Design Choices	12
3.3 Overall Structures	14
3.4 Technologies and Methodologies	14
3.5 Testing and Evaluation Techniques	15
3.5.1 Evaluation of Project	15
3.5.2 User Testing Survey	17
<b>Chapter 4: Implementation</b>	<b>18</b>
4.1 Importing datasets and preprocessing texts	18
4.1.1 Importing datasets	18
4.1.2 Text Processing	19
4.2 Setting up for Machine Learning	20
4.3 Logistic Regression Model	21
4.4 Decision Tree Classifier	21
4.5 Gradient Boosting Classifier	22
4.6 Random Forest Classifier	23
4.7 Recurrent Neural Network	24
4.7.1 Importing Necessary Modules	24
4.7.2 Tokenising training text data	24
4.7.3 Converting tokenised text sequences into padded sequences	24
4.7.4 Deep Learning Model Architecture	25
4.7.5 Training of deep learning model using TensorFlow's Keras API	25
4.7.6 Predict_rnn Function	26
4.8 BERT Model	27

4.9 Chat Box and Confidence Score Generator	29
<b>Chapter 5: Evaluation</b>	<b>30</b>
5.1 Evaluation of Ensemble Methods Models	30
5.1.1 Logistic Regression Model	30
5.1.2 Decision Tree Classifier	31
5.1.3 Gradient Boosting Classifier	31
5.1.4 Random Forest Classifier	32
5.2 Evaluation of RNN model	32
5.2.1 Plotting Training and Validation Loss	32
5.2.2 Plotting Training and Validation Accuracy	33
5.3 BERT Model Evaluation	35
5.4 User Testing Survey Evaluation	35
<b>Chapter 6: Conclusion</b>	<b>37</b>
6.1 Project Summary	37
6.2 Model Performance	37
6.3 Design Choices and Methodologies	37
6.4 Research Questions Summary	37
6.5 Future Developments	38
6.6 Limitations Faced	38
<b>Chapter 7: Appendices</b>	<b>39</b>
<b>Chapter 8: References</b>	<b>41</b>

## **Chapter 1: Introduction**

### **1.1 Project Details**

**Module Chosen:** CM3060 Natural Language Processing

**Project Idea Title 1:** Fake News detection

**Project Name:** Multimodal VeriTruth Detection

**Details:** The project will focus on creating a user-friendly and accessible platform for verifying news, primarily through the development of a fake news detector Chatbot.

**Github Repository Link:**

<https://github.com/kiranviews/Final-Year-Project/tree/44710a95a469b07b7ce99e85698d1aa0b00c157f/Multimodal%20Veritruth%20Detection>

### **1.2 Project Concept**

The project aims to create a user-engaging platform for verifying news content by developing a multimodal python project with a chatbot style input system. This project is motivated by the pressing need to combat the proliferation of fake news and misinformation. By leveraging natural language processing and machine learning, the project intends to empower users by providing a means to assess the credibility of news in real-time. The brief concept is that users should be able to send in a chunk of text into a chatbox. The bot should return a confidence score on how reliable the news is.

### **1.3 Motivation**

The motivation behind this project is to harness natural language processing and machine learning technologies to create a user-friendly, real-time verification system, aiding individuals in making informed decisions while engaging the community in reporting and addressing dubious content. The project aspires to contribute significantly to building a vigilant, informed, and resilient community in the face of rising spread of fake news, fraudulent activities and misinformation.

The motivation behind this project also stems from the escalating threat of fake news and scams, particularly prevalent in Singapore. Recent statistics released by the Singapore Police Force revealed a staggering loss of more than S\$330 million to scammers in the first half of 2023 alone. This drastic increase in financial exploitation through fraudulent activities underscores the urgent need for robust preventive measures. [1]

This alarming trend underscores the critical necessity for accessible and effective tools to combat the proliferation of fake news and scams. The project endeavours to offer an immediate solution,

empowering users with a reliable platform to discern the authenticity of news they encounter, thus mitigating the potential financial and societal impact of misinformation.

Hence, there is a strong need in creating an easy method for users to verify their news and use it as a guide to check on how much they can rely on a certain source material. By creating an easy interface, can allow for all demographics, respective of any levels of digital literacy to use the chat bot.

#### **1.4 Research Questions**

This project aims to answer the following research questions:

- How can a chatbot be effectively utilised to facilitate the submission and verification of news content?
- How can a fine-tuned model be integrated into the chatbot to classify submitted content as real or fake?
- What are the various models and machine learning methods can I use to accommodate all kinds of data and linguistics challenges for text classification?

#### **1.5 Deliverables and Justifications**

The project primarily aims to deliver the development of a multi-modal fake news classification system capable of processing textual information. The proposed system aims to employ natural language processing and machine learning techniques to discern the authenticity of news content by analysing text.

The core deliverables of this initiative include:

1. **Creation of a Multi-Modal Fake News Classification System:** The system will integrate sophisticated algorithms and models to process textual information by utilising multiple models focusing on different textual meanings and contexts. This multi-modal approach will enable a comprehensive analysis, considering various dimensions of news articles to determine their authenticity.
2. **Development of a confidence score generator:** The project involves crafting a confidence score generator which will give the confidence score across all models in the project. This will also be done in a chatbot style where users can paste in a chunk of news as an input and give the news prediction.

The justification for these deliverables is rooted in addressing the pressing need for a versatile and efficient tool to combat the proliferation of fake news and scams. The multi-modal classification system aims to enhance the accuracy and reliability of news verification, considering the various facets of textual information. The integration of the Chatbot serves to provide a convenient and user-friendly platform, ensuring easy accessibility for users to engage

in real-time news verification, thus contributing to the mitigation of the rising issue of misinformation and scams.

## **Chapter 2: Literature Review**

### **2.1 Rising crisis of Fake news, phishing and fraudulent activities**

This journal article addresses the pressing need for digital media literacy in the face of rising misinformation, cybercrimes, and online risks, particularly targeting adults, especially non-digital natives and older individuals [2]. It underscores how events like the 2016 U.S. presidential election and various online scams and phishing attacks highlight the crucial necessity for effective digital media literacy interventions beyond traditional classroom settings. Even though the context used in the article is US centric, it still applies to any digitally active and enhanced countries, like Singapore. Something of this sort occurred in Singapore when scammers used Prime Minister Lee's photo for crypto based scams and also created fake Channel News Asia articles using his face to lure in victims. [3]

The paper [2] emphasises the scarcity of research in educating adults, particularly non-digital natives, about safe social media usage, recognizing false information, and evading cybercrimes. This research paper [4] shows that the evolution of fake news extends beyond politics, potentially infiltrating the corporate world to harm competitors and influence financial markets using advanced strategies. This shift threatens corporate reputation and economic stability, emphasising the grave impact of sophisticated fake news scams on businesses and markets. This in turn directly affects the public at the end of the day. Numerous cybercrimes exploit individuals' lack of knowledge in verifying sources and identifying false information. Additionally, the proliferation of fake news during critical events, such as elections, significantly impacts public discourse and global relations.

The articles above emphasise the importance of evaluating information sources and understanding diverse forms of fake news—satire, hoaxes, and propaganda—and their potential consequences, including influencing elections, amplifying political divisions, and inciting unrest. Research [2] suggests that correcting misinformation once individuals believe it can be challenging, implying that teaching scepticism might prove more effective. It proposes evaluating existing educational materials and scam classifying methods and assessing their impact on adult populations, considering individual differences like political ideology and age in these interventions' effectiveness.

The articles underscore the escalating threat posed by fake news and its detrimental consequences for those affected. Upon reviewing the papers, there is a heightened sense of urgency and pressing need of this project to address this issue through the implementation of a

fake news detection project. The articles' emphasis on the complexities of rectifying misinformation reinforces the immediate need for action. This urgency compels me to focus on developing robust interventions, particularly geared toward enhancing digital media literacy among adults. The objective is to counter the adverse effects of fake news and bolster defences against online threats.

## **2.2 Increasing need for Natural Language Processing in Fake News Prevention**

Regarding the content of the scholarly article [5], it extensively highlighted the challenges posed by the rampant dissemination of fake news and the limitations of conventional fact-checking techniques. The central emphasis was on the inadequacy of traditional methods in keeping pace with the volume and speed of misinformation propagation.

The article strongly advocated for the integration of advanced technologies, specifically natural language processing (NLP) and machine learning, to effectively address the spread of fake news. It elucidated how these technologies could significantly enhance the verification processes by swiftly analysing and assessing the credibility of information.

Moreover, the article underscored the importance of adapting these technologies to local demographics and cultural nuances. It stressed that effective solutions for combating fake news should not only rely on global standards but also consider the linguistic and cultural intricacies of specific regions which is one of the primary focuses of this project.

This article [6], explains the use of NLP in detecting fake news on social media is a crucial area of research due to the detrimental impact of intentionally misleading information. Studies explore methods like the random forest algorithm and NLP techniques, such as counting vectors and RID matrices, to analyse word usage for identifying fake news. However, challenges persist, as word count similarities may not accurately reflect the true context or meaning, making it difficult to discern between genuine and fake news. The aim is to leverage NLP and algorithms like Random Forest to create models that classify articles, distinguishing between trustworthy information and hoaxes, thereby ensuring the credibility of news for readers. This is why this project aims to use multiple models which are trained for multiple linguistic needs to make the final confidence derivation much more accurate and precise.

In line with the article's insights, my project on multi-modal fake news classification aligns by integrating multiple models for text analysis, leveraging NLP and machine learning. Furthermore, my project aims to cater specifically to the Singaporean context, recognizing the necessity of a solution tailored to the linguistic and cultural characteristics of this region.

The scholarly articles recommendations serve as a guiding framework for my project, emphasising the significance of technological advancements, the incorporation of multi-modal

analysis, and the essential customization to address the pressing concerns of fake news in Singapore.

### **2.3 Harnessing the power of BERT model for text classification**

In recent years, detecting rumours on social media platforms, particularly Twitter, has become a critical endeavour due to the exponential spread of unverified information. The challenge in distinguishing between facts and rumours has led researchers to explore sophisticated models for rumour detection. The scholar article "CE-BERT-Based Model for Detecting Rumours on Twitter" offers insights into the effectiveness of BERT (Bidirectional Encoder Representations from Transformers) models for this purpose. [7]

The scholar article [7] exemplifies the growing significance of BERT-based models in rumour and fake news detection. BERT, known for its contextual understanding and word representations, has consistently exhibited state-of-the-art performance in natural language processing tasks. Specifically, when analysing Twitter data for rumour detection, the bidirectional nature of BERT proves beneficial in understanding the contextual nuances and the informal, noisy language commonly found in tweets. Researchers have compared various BERT variants like BERTBASE, RoBERTa, and others, demonstrating the superiority of BERT models over traditional machine learning models.

BERT's capacity to capture semantic and contextual information is particularly pertinent in the context of Twitter, where the brevity and dynamics of information dissemination present unique challenges. This will greatly benefit this project as a news classifier should take both semantic and contextual settings when classifying news. The scholar article underscores the ability of BERT models to process and analyse the contextual information embedded in short, real-time tweets, making them an ideal choice for detecting rumours on such platforms.

BERT's performance evaluation :

1. SQuAD v1.1 & v2.0: SQuAD, a dataset with 108k questions answerable from Wikipedia paragraphs, serves as a reading comprehension benchmark. [8] BERT's exceptional performance on this dataset marked a significant milestone, surpassing both previous top models and achieving results on par with human-level performance. ([Refer to image 1 in appendices](#))
2. SWAG, or Situations With Adversarial Generations, evaluates a model's commonsense inference ability via a dataset featuring 113k multiple-choice questions derived from video scenes. [8] The questions ask about the next plausible scenario, providing four options for the model to predict. BERT's performance surpassed that of previous top models, including human-level performance, marking a significant advancement in understanding commonsense reasoning. ([Refer to image 2 in appendices](#))



The emergence of BERT triggered widespread adoption in natural language processing tasks, showcasing a paradigm shift. [9] Its advancements led to improved results in subsequent applications, establishing BERT as a pivotal leap in intelligent text processing. Its bidirectional contextual understanding, rooted in transformer architecture with attention mechanisms, demonstrated superiority in comprehension. BERT's utilisation on vast datasets as a foundational model vastly enhanced problem-solving algorithms. Future prospects entail continual scientific breakthroughs through BERT's adaptation across diverse language processing challenges, promising substantial enhancements in various NLP algorithms, from translation to question-answering systems, via refined neural network architecture and training optimization.

The study [10] compares BERT, a state-of-the-art NLP model, with traditional methods like TF-IDF in four NLP scenarios, consistently demonstrating BERT's superior performance and ease of implementation. Transfer learning via pre-training significantly boosts BERT's effectiveness, especially evident in scenarios with smaller datasets. While acknowledging BERT's limitations, the approach suggests hyperparameter-tuned BERT models using Bayesian Optimization for improved results in various NLP tasks. For the consideration of a model for this fake news detection project, BERT demonstrated superiority in handling diverse NLP tasks, ease of implementation, and potential for enhanced performance through hyperparameter tuning make it a compelling choice. Its ability to capture complex contextual features could significantly enhance accuracy in classifying news authenticity.

## **2.4 Utilising Recurrent Neural Networks (RNNs) for optimised text classification**

Neural networks, akin to the human brain's structure and learning processes, excel at discerning patterns from vast data volumes. While they demonstrate prowess in tasks like identifying dog breeds from image pixels, Recurrent Neural Networks (RNNs) emerge as crucial for tasks involving sequences such as natural language processing. [11] RNNs possess a unique ability to retain memory of past inputs, enabling them to effectively handle sequential data. Unlike traditional networks limited to considering only current inputs, RNNs process both current and previous information, making them invaluable for tasks requiring contextual understanding, notably in distinguishing between fake and factual content in text classification for fake news detection. ([Refer to Image 3 in appendices](#))

In contrast to standard feed-forward networks, RNNs' internal memory enables them to comprehend and analyse sequences, making them pivotal for discerning context in tasks like fake news detection. Their proficiency in handling sequential data and retaining contextual information provides a significant advantage in identifying misleading information amidst a sea of textual data. Consequently, RNNs stand out as a critical tool in text classification, leveraging their capacity to process sequences effectively, thereby enhancing the accuracy and reliability of fake news detection systems.

The paper [12] addresses the limitations of traditional text classification methods in natural language processing, highlighting issues such as dimensionality, data scarcity, and limited adaptability. It focuses on the potential of Recurrent Neural Networks (RNN-based) within deep learning for text classification. RNNs, alongside Convolutional Neural Networks (CNNs) and Attention Mechanisms-Based models, emerge as pivotal in surpassing traditional methods when handling vast and complex datasets. [\(Refer to Image 4 in appendices\)](#)

Studies consistently showcase the superiority of RNN-based text classification in avoiding complex feature extraction processes and achieving heightened accuracy with unstructured data. The paper delves into the advantages of RNNs over traditional methods, elucidating the process within deep learning frameworks. This includes stages such as text preprocessing, developing distributed text representations, constructing RNN-based classification models, and conducting comprehensive performance evaluations.

## **2.5 Summary of Literature Review**

The literature review highlights the escalating challenges posed by fake news and cybercrimes, emphasising the crucial need for digital media literacy interventions. Studies underscore the impact of misinformation on elections, corporate stability, and public discourse, particularly stressing the scarcity of research focusing on educating adults, especially non-digital natives. This review stresses the importance of evaluating diverse forms of fake news and suggests teaching scepticism as a potential solution. Furthermore, it accentuates the growing necessity for leveraging advanced technologies, specifically Natural Language Processing (NLP) and models like BERT and Recurrent Neural Networks (RNNs), to combat the rapid dissemination of misinformation. While BERT demonstrates exceptional contextual understanding and performance across various NLP tasks, RNNs emerge as pivotal for effectively handling sequential data, particularly in text classification for fake news detection within deep learning frameworks. These insights collectively drive the impetus for this project, aiming to integrate multi-modal analysis while considering Singapore's linguistic and cultural nuances in addressing the challenges posed by fake news.

## **Chapter 3: Project Design**

### **3.1 Domain and Users**

The domain of my project revolves around combating the spread of misinformation in online media, specifically focusing on the detection of fake news. This is an increasingly critical area, considering the vast amount of information shared on social media and news platforms.

#### **Primary Users:**

1. **Vulnerable Individuals:** This includes individuals who may lack digital literacy skills or awareness of common online scams. These users are at higher risk of being targeted by scammers and may benefit from accessible tools and educational resources to identify and avoid fraudulent content.
2. **Community Organisations:** Non-profit organisations, community centres, and educational institutions that work directly with vulnerable populations can also benefit from this project. They can use the tools and insights provided to educate and support their members in recognizing and responding to online scams effectively.
3. **Consumer Protection Agencies:** Government agencies and consumer advocacy groups tasked with combating fraud and protecting consumers' rights can leverage the project's technology to enhance their efforts in identifying and addressing online scams.

Understanding the diverse needs of these users has been integral in shaping the design and selection of models used in this project. For instance, journalists and fact-checkers might require detailed insights into the linguistic nuances and context of news articles, hence the utilisation of models like BERT and RNN for their contextual understanding abilities. Conversely, social media platforms might favour ensemble methods like random forest and gradient boosting classifiers for their efficiency in handling large volumes of data.

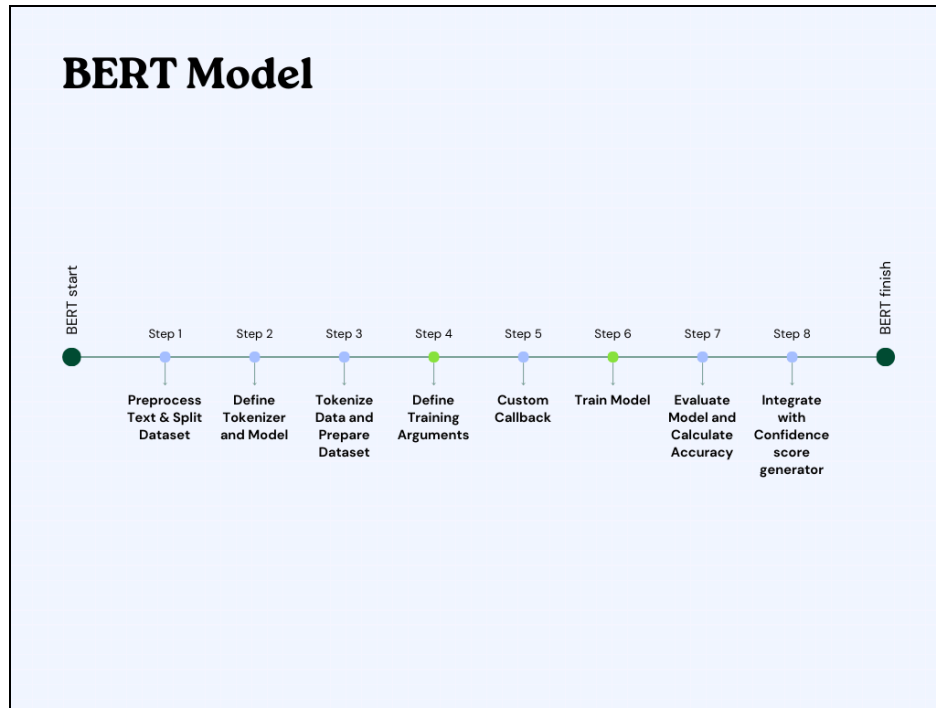
By catering to these user needs and the domain's demand for accurate news verification, the aim is to provide a robust and versatile system that addresses the challenges posed by the proliferation of fake news in the digital landscape.

### **3.2 Design Choices**

Justification and explanation behind the design choices for each model incorporated into the fake news detection project:

#### **BERT (Bidirectional Encoder Representations from Transformers)**

BERT excels in understanding the contextual meaning of words within a sentence or paragraph. It captures bidirectional dependencies and context, crucial for deciphering the nuanced language often found in fake news articles. Its pre-trained nature on large corpora helps in grasping subtle linguistic cues, improving accuracy in identifying misinformation.

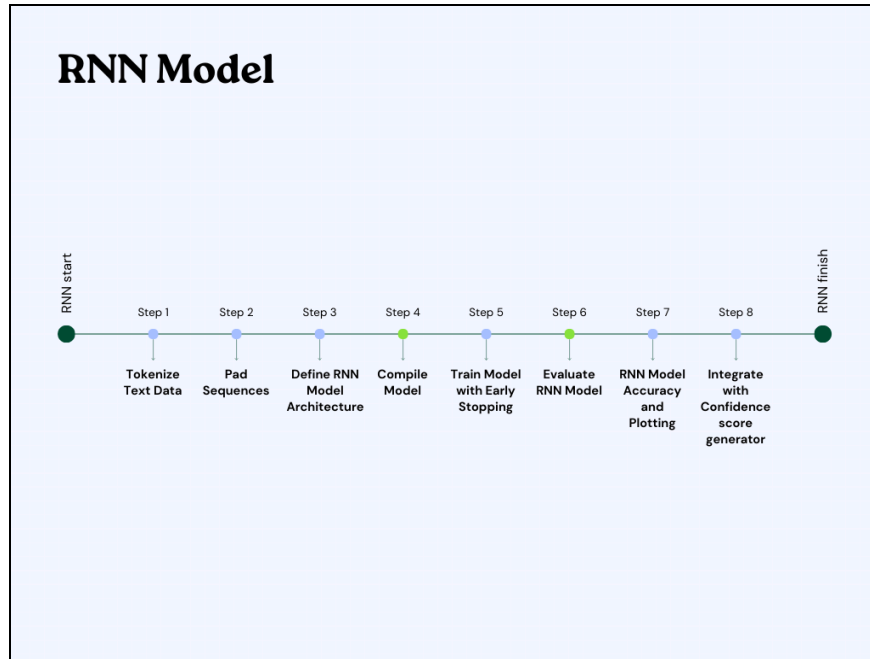


***BERT Model Execution flowchart***

The flowchart above demonstrates the flow of work on how the BERT model will be executed. The full implementation is covered in [Chapter 4: Implementation under BERT Model](#).

## **RNN (Recurrent Neural Network)**

RNNs are ideal for handling sequential data, which is common in textual information like news articles. Its ability to retain memory of past inputs makes it suitable for tasks where the order of words or phrases matters. In fake news detection, RNNs can capture the sequential structure of sentences and paragraphs, thus aiding in understanding the flow of information.



*RNN Model Execution flowchart*

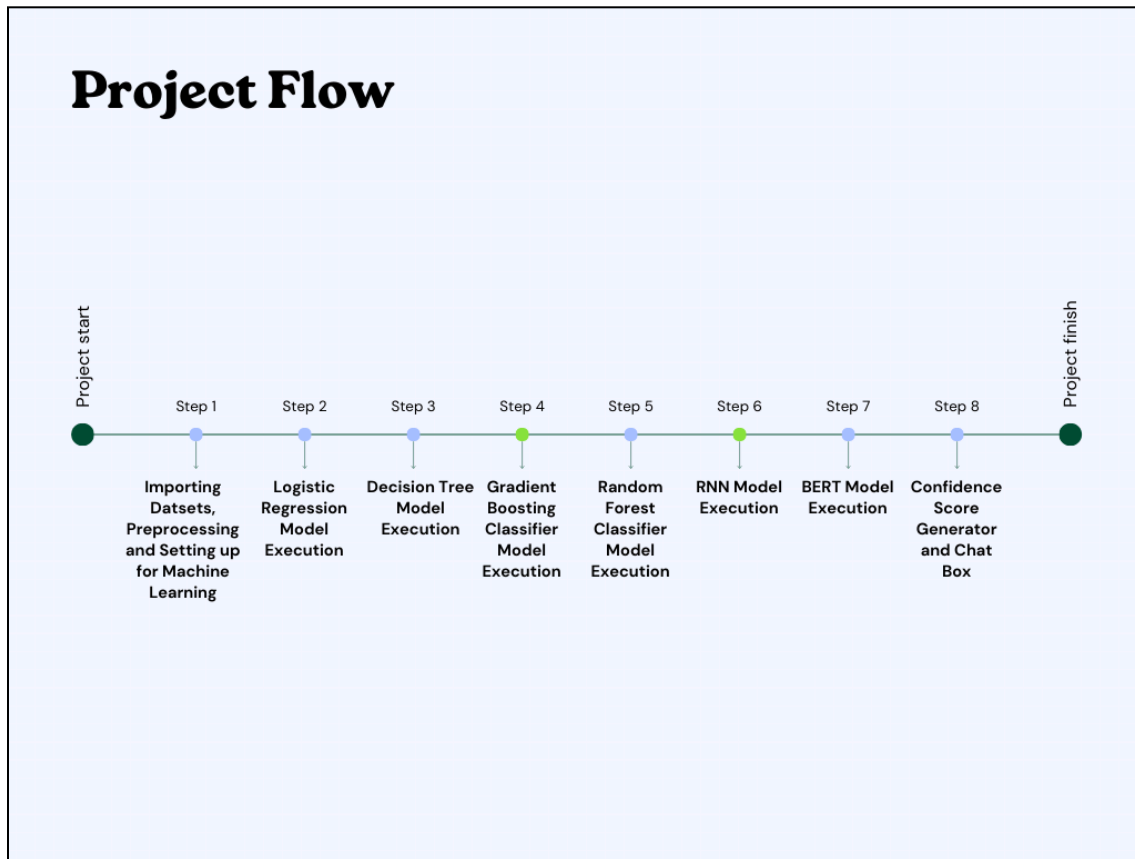
The flowchart above demonstrates the flow of work on how the RNN model will be executed. The full implementation is covered under [Chapter 4: Implementation under Recurrent Neural Network](#).

### **Logistic Regression, Decision Tree Classifier, Gradient Boosting Classifier, Random Forest Classifier (Ensemble Methods)**

Ensemble methods like decision trees, gradient boosting, and random forests are chosen for their ability to combine multiple models to produce more accurate predictions. They are robust against overfitting and handle both numerical and categorical data well. In the context of fake news detection, ensemble methods can leverage diverse decision-making processes to collectively make more informed predictions, potentially improving overall accuracy.

Each model offers unique strengths that contribute to the comprehensive detection of fake news. Their selection is based on the specific characteristics of fake news data, aiming to leverage their strengths in various aspects—context understanding, sequential analysis, and collective decision-making—to improve the overall accuracy and reliability of the detection system.

### 3.3 Overall Structures



*Project Execution Flowchart*

The flowchart above visualises how the project will be executed. The full breakdown of the execution will be explained in [Chapter 4](#) of this report.

### 3.4 Technologies and Methodologies

#### **BERT (Bidirectional Encoder Representations from Transformers)**

BERT excels in understanding contextual nuances within text, capturing bidirectional dependencies, and contextual information. This is crucial in fake news detection, as it helps in comprehending the subtle linguistic cues and context that might indicate misinformation. BERT is preferred for its ability to interpret complex language structures and its pre-trained model's adaptability to various language-related tasks without extensive fine-tuning. BERT is effective in capturing the semantic meaning and context of news articles, enhancing the accuracy of distinguishing between fake and genuine news based on language nuances.

## **RNN (Recurrent Neural Network)**

RNNs are adept at handling sequential data and retaining information from past inputs, making them ideal for tasks where the order of words matters, such as in news articles. Chosen due to its sequential analysis capabilities, RNN is good at enabling the understanding of the flow and structure of information in news articles. RNN helps in capturing dependencies among words and sentences, aiding in the identification of patterns specific to fake news propagation.

## **Ensemble Methods (Logistic Regression, Decision Tree Classifier, Gradient Boosting Classifier, Random Forest Classifier)**

Ensemble methods combine multiple models to make more accurate predictions than individual models. They are robust against overfitting and handle diverse data types effectively. They are Selected for their robustness and ability to handle both numerical and categorical data, which is common in diverse fake news datasets. By leveraging multiple decision-making processes, ensemble methods provide a collective insight into identifying fake news, improving the overall accuracy and reliability of the detection system.

To summarise, BERT and RNN were chosen for their abilities in understanding context and sequential data within news articles, capturing linguistic nuances and structural dependencies. Ensemble Methods are preferred for their robustness, collective decision-making abilities, and proficiency in handling diverse data types, enhancing the system's accuracy in detecting fake news across various contexts and formats.

These technologies and methods were selected based on their specific strengths, relevance to fake news detection tasks, and their collective ability to provide a comprehensive analysis of textual data, ultimately contributing to more accurate identification of fake news.

## **3.5 Testing and Evaluation Techniques**

### **3.5.1 Evaluation of Project**

In the evaluation of our fake news detection project, we've identified specific metrics crucial for assessing the performance of our models. We'll focus on key metrics such as **accuracy, precision, recall, and the F1-score**. These metrics are fundamental in gauging the effectiveness of our models in differentiating between genuine and fake news articles.

- Accuracy serves as a measure of overall correctness, showcasing the ratio of correctly predicted instances to the total instances.
- Precision delves into the accuracy of positive predictions, specifically the ratio of correctly predicted positive observations to the total predicted positive observations.
- Recall indicates the coverage of actual positives, revealing the ratio of correctly predicted positive observations to all actual positives.

- The F1-score, as a harmonic mean of precision and recall, provides a balanced measure, ensuring a comprehensive evaluation of our models' performance.

The evaluation aims are strategically aligned with the complexities of fake news detection. This project aims to minimise false positives, ensuring that genuine news is not incorrectly flagged as fake, maintaining the credibility of reliable information. Additionally, reducing false negatives is critical, minimising cases where fake news goes undetected and mitigating its potential impact on misinformation. Simultaneously, the goal is to improve the overall accuracy of predictions, enhancing the correctness of our detection system. To achieve optimal performance, it is important to prioritise maintaining a balance between precision and recall, striking an equilibrium that maximises the identification of fake news while minimising misclassifications.

In conducting a comprehensive evaluation, the project plans to perform in-depth experimentation and comparative analyses. This involves individually evaluating each model, such as BERT and RNN, to discern their strengths and weaknesses in discerning fake news. Furthermore, this project will assess the performance of ensemble methods against individual models to gauge the enhancement achieved through their combination. Hyperparameter tuning will be a pivotal aspect of the evaluation, allowing to optimise model performance by experimenting with various hyperparameter configurations.

In the visualisation and examination of our BERT and RNN implementations during training, there will plot loss curves and accuracy curves.

- For BERT, I will chart epochs against training and validation loss values, aiming to identify convergence and divergence between the two to detect signs of overfitting or underfitting.
- A ROC graph and confusion matrix will also be plotted for the BERT Model to evaluate it.
- Similarly, with RNNs, plotting epochs against training and validation losses, especially sequence-wise loss changes, will provide insights into their performance with sequential data.

These evaluation strategies and visualisations are deeply intertwined with the project's core goal: creating a robust fake news detection system. They provide a comprehensive framework for assessing the models' performance and guiding iterative improvements to ensure the system's effectiveness in combating misinformation in online media.



### **3.5.2 User Testing Survey**

With user testing surveys, Potential users are asked to interact with a system or product—in this case, the fake news classification model. The goal is to observe how users interact with and understand the system, as well as to gather feedback on its usability and effectiveness.

involve inviting individuals to use your classification model. Users are asked to read news articles or headlines and then use the model to determine if they think the content is fake or real. Users can then observe the process, note any difficulties they encounter, and gather their feedback on the accuracy and ease of use of your model on a google form that will be sent out.

By conducting user testing, it is possible to identify any issues or areas for improvement in the fake news classification model, ensuring that it meets the needs of its intended users and effectively distinguishes between fake and real news.

The questions and survey results are collated and explained in [Chapter 5: Evaluation under User Testing Survey Evaluation](#).

## Chapter 4: Implementation

### 4.1 Importing datasets and preprocessing texts

#### 4.1.1 Importing datasets

```
fake_df = pd.read_csv("Fake.csv")
true_df = pd.read_csv("True.csv")

fake_df.head()
```

```
fake_df["class"] = 0
true_df["class"] = 1
```

```
data = data_merge.drop(['title', 'subject', 'date'], axis=1)
```

This project utilises Pandas to read two CSV files, “Fake.csv” and “True.csv”, into separate dataframes. It adds a “class” column to each dataframe to label them as fake (0) and true (1). Then it merges the dataframes, drops unnecessary columns, and resets the index to prepare the data for analysis and the upcoming machine learning tasks.

#### 4.1.2 Text Processing

```
def preprocess_text(text):
    # Convert text to Lowercase
    text = text.lower()

    # Remove square brackets and their contents
    text = re.sub(r'\[.*?\]', '', text)

    # Remove non-alphanumeric characters
    text = ''.join(char for char in text if char.isalnum() or char.isspace())

    # Remove URLs
    text = re.sub(r'https?://\S+|www\.\S+', '', text)

    # Remove XML/HTML tags
    text = re.sub(r'<.*?>', '', text)

    # Remove punctuation
    text = ''.join(char for char in text if char not in string.punctuation)

    # Tokenize the text
    tokens = word_tokenize(text)

    # Remove stop words
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]

    # Remove words containing digits
    tokens = [word for word in tokens if not any(c.isdigit() for c in word)]

    # Lemmatize tokens
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    # Join tokens back into a single string
    processed_text = ' '.join(tokens)

    return text

data['text'] = data['text'].apply(preprocess_text)
data.head()
```

Text processing involves the following:

1. Converting text lowercase using the lower() method
2. Removing square brackets and their contents by using regular expression (re) to remove square brackets and any text within them.
3. Removing Non-Alphanumeric characters by utilising a list comprehension to keep only alphanumeric characters and whitespaces.
4. Removing URLs by using regular expression (re) to remove URLs starting with “http” or “https” as well as URLs starting with “www”
5. Remove XML/HTML tags by employing regular expression (re) to remove any XML/HTML tags (i.e., text within < and >).
6. Remove punctuation using the string.punctuation constant.
7. Tokenise the text by splitting the text into tokens using word\_tokenize from the nltk.tokenize module.

8. Remove stop words by removing common English stop words using the stopwords corpus from the nltk.corpus module.
9. Removing words containing digits by filtering out words containing digits using a list comprehension with isdigit().
10. Lemmatizing tokens using the WordNet lemmatizer from the nltk.stem module
11. Join tokens back into a single string by joining the processed tokens back onto a single string using ''.join(tokens)
12. Finally, it returns the preprocessed text

## 4.2 Setting up for Machine Learning

```
x=data['text']  
y=data['class']
```

x: Assigns the values from the 'text' column of the data to the variable x.

y: Assigns the values from the 'class' column of the data to the variable y.

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25)
```

The train\_test\_split function divides the dataset into two parts: training and testing sets. The training set (x\_train, y\_train) is used to train the model, while the testing set (x\_test, y\_test) is kept separate to evaluate the model's performance. By setting test\_size = 0.25, 25% of the data is allocated for testing, ensuring that the model learns from the majority of the dataset while having a distinct portion for evaluation.

```
from sklearn.feature_extraction.text import TfidfVectorizer  
vectorization = TfidfVectorizer()  
xv_train = vectorization.fit_transform(x_train)  
xv_test = vectorization.transform(x_test)
```

**Instantiating TF-IDF Vectorizer:**  
vectorization = TfidfVectorizer(): Creates a TF-IDF vectorizer object named vectorization.

**Fitting and Transforming Training Data:**  
Fits the vectorizer to training data (x\_train) and transforms it into TF-IDF matrices (xv\_train). Calculates TF-IDF values for each term in the training data.

**Transforming Testing Data:** Uses the fitted vectorizer to transform testing data (`x_test`) into TF-IDF matrices (`xv_test`). Applies the same transformation as learned from the training data.

### 4.3 Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression  
  
LR_model = LogisticRegression()  
LR_model.fit(xv_train, y_train)
```

**Instantiate Logistic Regression Model** instantiates a Logistic Regression model object named `LR_model`.

**Fit Model to Training Data** fits the Logistic Regression model (`LR_model`) to the TF-IDF transformed training data (`xv_train`) along with their corresponding labels (`y_train`). This step trains the model to learn the relationship between the features and the target classes.

```
pred_lr = LR_model.predict(xv_test)  
LR_model.score(xv_test, y_test)
```

Uses the trained Logistic Regression model (`LR_model`) to make predictions on the TF-IDF transformed testing data (`xv_test`). The predicted labels are stored in the variable `pred_lr`.

**Model Evaluation** calculates the accuracy score of the Logistic Regression model on the testing data (`xv_test`) and their corresponding true labels (`y_test`).

```
print(classification_report(y_test, pred_lr))
```

**Print Classification Report** prints a classification report which includes precision, recall, F1-score, and support for each class, based on the true labels (`y_test`) and the predicted labels (`pred_lr`).

#### 4.4 Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier  
  
DT_model = DecisionTreeClassifier()  
DT_model.fit(xv_train, y_train)
```

**Instantiate Decision Tree Classifier** instantiates a Decision Tree classifier object named DT\_model.

**Fit Model to Training Data** fits the Decision Tree classifier (DT\_model) to the TF-IDF transformed training data (xv\_train) along with their corresponding labels (y\_train). This step trains the model to learn the patterns in the feature-label relationship.

```
pred_dt = DT_model.predict(xv_test)  
DT_model.score(xv_test, y_test)
```

**Make Predictions** uses the trained Decision Tree classifier (DT\_model) to make predictions on the TF-IDF transformed testing data (xv\_test). The predicted labels are stored in the variable pred\_dt.

Model Evaluation calculates the accuracy score of the Decision Tree classifier on the testing data (xv\_test) and their corresponding true labels (y\_test).

```
print(classification_report(y_test, pred_dt))
```

**Print Classification Report** prints a classification report which includes precision, recall, F1-score, and support for each class, based on the true labels (y\_test) and the predicted labels (pred\_dt).

#### 4.5 Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier  
  
GB_model = GradientBoostingClassifier(random_state = 0)  
GB_model.fit(xv_train, y_train)
```

**Instantiate Gradient Boosting Classifier** instantiates a Gradient Boosting classifier object named GB\_model. The parameter random\_state=0 ensures reproducibility of results by fixing the random seed.

**Fit Model to Training Data** fits the Gradient Boosting classifier (GB\_model) to the TF-IDF transformed training data (xv\_train)

along with their corresponding labels (y\_train). This step trains the model to learn the patterns in the feature-label relationship.

```
pred_gb = GB_model.predict(xv_test)
GB_model.score(xv_test, y_test)
```

**Make Predictions** uses the trained Gradient Boosting classifier (GB\_model) to make predictions on the TF-IDF transformed testing data (xv\_test). The predicted labels are stored in the variable pred\_gb.

**Model Evaluation** calculates the accuracy score of the Gradient Boosting classifier on the testing data (xv\_test) and their corresponding true labels (y\_test).

```
print(classification_report(y_test, pred_gb))
```

**Print Classification Report:**  
print(classification\_report(y\_test, pred\_gb)): Prints a classification report which includes precision, recall, F1-score, and support for each class, based on the true labels (y\_test) and the predicted labels (pred\_gb).

## 4.6 Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
RF_model = RandomForestClassifier(random_state = 0)
RF_model.fit(xv_train, y_train)
```

**Instantiate Random Forest Classifier:**  
RF\_model = RandomForestClassifier(random\_state=0): Instantiates a Random Forest classifier object named RF\_model. The parameter random\_state=0 ensures reproducibility of results by fixing the random seed.

**Fit Model to Training Data:**  
RF\_model.fit(xv\_train, y\_train): Fits the Random Forest classifier (RF\_model) to the TF-IDF transformed training data (xv\_train) along with their corresponding labels (y\_train). This step trains the model to learn the patterns in the feature-label relationship.

```
pred_rf = RF_model.predict(xv_test)
RF_model.score(xv_test, y_test)
```

**Make Predictions:** `pred_rf = RF_model.predict(xv_test)`: Uses the trained Random Forest classifier (`RF_model`) to make predictions on the TF-IDF transformed testing data (`xv_test`). The predicted labels are stored in the variable `pred_rf`.

**Model Evaluation:** `RF_model.score(xv_test, y_test)`: Calculates the accuracy score of the Random Forest classifier on the testing data (`xv_test`) and their corresponding true labels (`y_test`).

```
print(classification_report(y_test, pred_rf))
```

**Print Classification Report** prints a classification report which includes precision, recall, F1-score, and support for each class, based on the true labels (`y_test`) and the predicted labels (`pred_rf`).

## 4.7 Recurrent Neural Network

### 4.7.1 Importing Necessary Modules

```
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Bidirectional, Dense, Dropout
```

The code imports TensorFlow for deep learning, including modules for sequence preprocessing and layers (like Embedding, LSTM, Bidirectional, Dense, and Dropout) essential for constructing a deep learning model for text classification using the Sequential model.

### 4.7.2 Tokenising training text data

```
max_vocab = 10000
tokenizer = Tokenizer(num_words=max_vocab)
tokenizer.fit_on_texts(x_train)
```

This code segment sets a maximum vocabulary size, instantiates a `Tokenizer` object with this size, and fits the `Tokenizer` to the training data, assigning unique integer indices to words based on their frequency. This prepares the text for further processing and model training.

By tokenizing the training text data, the Tokenizer object prepares the text for further processing and model training by converting each word into a unique integer index based on its frequency in the dataset.

#### 4.7.3 Converting tokenised text sequences into padded sequences

```
x_train_seq = tokenizer.texts_to_sequences(x_train)
x_test_seq = tokenizer.texts_to_sequences(x_test)
x_train_pad = pad_sequences(x_train_seq, padding='post', maxlen=256)
x_test_pad = pad_sequences(x_test_seq, padding='post', maxlen=256)
```

This code segment first converts tokenized text sequences into sequences of integer indices, representing each word with a unique index from the tokenizer's vocabulary. Then, it pads these sequences to ensure uniform length for efficient processing in a deep learning model. This preparation step allows the model to handle inputs of varying lengths consistently during training and evaluation.

#### 4.7.4 Deep Learning Model Architecture

```
model = Sequential([
    Embedding(max_vocab, 128),
    Bidirectional(LSTM(64, return_sequences=True)),
    Bidirectional(LSTM(16)),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
model.summary()
```

Embedding Layer: Maps each word index to a dense vector representation of size 128.

Bidirectional LSTM Layers: Process input sequences in both forward and backward directions, capturing contextual information effectively. The first LSTM layer has 64 units and returns sequences, while the second has 16 units.

Dense Layers: Two dense layers follow the LSTM layers. The first has 64 units with ReLU activation, introducing non-linearity. Dropout regularisation with a rate of 0.5 is applied to mitigate overfitting. The second dense layer has a single unit with a sigmoid activation function, producing a binary classification output.



#### 4.7.5 Training of deep learning model using TensorFlow's Keras API

```
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                              patience=1,
                                              restore_best_weights=True)

model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy'])

history = model.fit(x_train_pad, y_train,
                    epochs=30,
                    validation_split=0.1,
                    batch_size=100,
                    shuffle=True,
                    callbacks=[early_stop])

# Evaluate RNN model
rnn_loss, rnn_accuracy = model.evaluate(x_test_pad, y_test)

print(f"RNN Model Accuracy: {rnn_accuracy}")
```

**Early Stopping:** Defines an early stopping callback named `early_stop`, which monitors the validation loss and stops training if the validation loss does not improve after one epoch. It restores the best weights of the model.

**Model Compilation:** Compiles the model with binary cross-entropy loss function, Adam optimizer with a learning rate of `1e-4`, and accuracy as the evaluation metric.

**Model Training:** Trains the model on the padded training data (`x_train_pad`) and corresponding labels (`y_train`) for 30 epochs. It uses a validation split of 0.1, a batch size of 100, and shuffles the data. The early stopping callback is passed to the training process.

**Model Evaluation:** Evaluates the trained model on the padded test data (`x_test_pad`) and corresponding labels (`y_test`) to compute the loss and accuracy.

#### 4.7.6 Predict\_rnn Function

```
import pandas as pd

def predict_rnn(news):
    new_seq = tokenizer.texts_to_sequences([news])
    padded_seq = pad_sequences(new_seq, padding='post', maxlen=256)
    prediction = model.predict(padded_seq)
    return int(prediction[0][0])
```

##### Tokenization:

In this case, the input news article is tokenized using a tokenizer object, which assigns a unique integer index to each word in the article. This step essentially converts the textual content into a sequence of numerical indices that the RNN can understand and process.

##### Padding:

RNN requires inputs of fixed length. However, news articles can vary in length, so padding is necessary to ensure that all input sequences have the same length. The

tokenized sequence is padded with zeros to match a predefined length of 256 tokens. This ensures consistency in the input shape expected by the deep learning model during inference.

### Prediction:

Once the tokenized sequence is padded, it is ready to be fed into the trained deep learning model for prediction. The model processes the padded sequence and outputs a probability distribution over the possible sentiment classes (e.g., positive or negative sentiment).

### Returning Prediction:

Since the model outputs probabilities, the predicted sentiment needs to be converted into a discrete integer value representing the sentiment class. This is typically done by selecting the class with the highest probability. For example, if the probability of positive sentiment is higher than negative sentiment, the predicted sentiment is assigned a value of 1; otherwise, it is assigned a value of 0.

## 4.8 BERT Model

```
# Preprocess the text data
data['text'] = data['text'].apply(preprocess_text)

# Split the dataset into train and test sets with a smaller sample size
sample_size = 500
sampled_data = data.sample(n=sample_size, random_state=42)

x = sampled_data['text']
y = sampled_data['class']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)

# Define the BERT tokenizer and model
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained('bert-base-uncased')

# Tokenize the text data
train_encodings = tokenizer(x_train.tolist(), truncation=True, padding=True)
test_encodings = tokenizer(x_test.tolist(), truncation=True, padding=True)
```

**Data Preprocessing:** Preprocesses the text data in the 'text' column of a DataFrame called data using a function named preprocess\_text.

**Data Splitting:** Randomly samples a subset of the data with a specified sample size (sample\_size) and splits it into training and testing sets.

**BERT Tokenization:** Initialises the BERT tokenizer and model from the

```
# Prepare the datasets
class NewsDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

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

    def __len__(self):
        return len(self.labels)

# Create train and test datasets using encoded inputs and labels
train_dataset = NewsDataset(train_encodings, y_train.tolist())
test_dataset = NewsDataset(test_encodings, y_test.tolist())
```

```
# Training arguments with minimal settings
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=5,
    per_device_train_batch_size=5,
    per_device_eval_batch_size=5,
    warmup_steps=50,
    logging_dir='./logs',
    logging_steps=5
)

# Define a custom callback to Log training progress
# Define a custom callback to Log training progress
class TrainingProgressCallback(TrainerCallback):
    def __init__(self):
        super().__init__()
        self.train_loss_values = []

    def on_train_begin(self, args, state, control, **kwargs):
        print("Training started...")

    def on_train_batch_end(self, args, state, control, **kwargs):
        # Log training Loss
        if 'loss' in state.log_history[-1]:
            loss = state.log_history[-1]['loss']
            self.train_loss_values.append(loss)
            if state.global_step % args.logging_steps == 0:
                print(f"Step {state.global_step}/{args.num_train_steps}, Loss: {loss:.4f}")
```

```
# Trainer to train the model with the adjusted settings and data
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    callbacks=[TrainingProgressCallback()]
)

# Measure time taken for training
start_time = time.time()
trainer.train()
training_time = time.time() - start_time

# Measure time taken for evaluation
start_time = time.time()
predictions = trainer.predict(test_dataset)
eval_time = time.time() - start_time

pred_labels = predictions.predictions.argmax(-1)
accuracy = accuracy_score(y_test, pred_labels)
print(f"Accuracy: {accuracy}")
print(f"Evaluation Time: {eval_time} seconds")
```

'bert-base-uncased' pre-trained model.

**Tokenization:** Tokenizes the text data using the BERT tokenizer, ensuring truncation and padding for consistent input lengths.

**Dataset Preparation:** Defines a custom PyTorch dataset class (NewsDataset) to prepare the train and test datasets with encoded inputs and labels.

**Training Arguments:** Sets up training arguments, including the output directory, number of training epochs, batch sizes, warmup steps, and logging settings.

**Training Progress Callback:** Defines a custom callback (TrainingProgressCallback) to log training progress, specifically training loss at each step.

**Model Training:** Initialises a Trainer object with the BERT model, training arguments, and datasets, and trains the model while logging the training progress.

**Evaluation:** Measures the time taken for both training and evaluation. Uses the trained model to make predictions on the test dataset and calculates accuracy. Prints the accuracy and evaluation time.

## 4.9 Chat Box and Confidence Score Generator

```
def output_label(n):  
    if n == 0:  
        return "This is a fake news"  
    elif n == 1:  
        return "This is not a fake news"
```

```
# Define the fakenewsdetection function  
def fakenewsdetection(news):  
    # Preprocess the input news text  
    processed_news = preprocess_text(news)  
  
    # Tokenize the input news text using BERT tokenizer  
    inputs = tokenizer(processed_news, return_tensors="pt", max_length=256, truncation=True)  
  
    # Predictions using Logistic Regression  
    pred_lr = LR_model.predict(vectorization.transform([processed_news]))[0]  
  
    # Predictions using Decision Tree  
    pred_dt = DT_model.predict(vectorization.transform([processed_news]))[0]  
  
    # Predictions using Gradient Boosting  
    pred_gb = GB_model.predict(vectorization.transform([processed_news]))[0]  
  
    # Predictions using Random Forest  
    pred_rf = RF_model.predict(vectorization.transform([processed_news]))[0]  
  
    # Predictions using RNN model  
    rnn_pred = model(inputs.input_ids)[0][0].detach().cpu().numpy()  
    pred_rnn = 1 if any(rnn_pred >= 0.5) else 0  
  
    # Predictions using BERT model  
    with torch.no_grad():  
        bert_outputs = model(**inputs)  
        bert_probabilities = torch.softmax(bert_outputs.logits, dim=-1)  
        bert_pred = torch.argmax(bert_probabilities, dim=-1).item()  
  
    # Aggregate predictions  
    predictions = [pred_lr, pred_dt, pred_gb, pred_rf, pred_rnn, bert_pred]  
  
    # Calculate confidence score  
    fake_count = predictions.count(0)  
    not_fake_count = predictions.count(1)  
    confidence_score = max(fake_count, not_fake_count) / len(predictions) * 100
```

```
# Output predictions  
print("\n\nLR Prediction: {}\nDT Prediction: {}\nGB Prediction: {}\nRF Prediction: {}\nRNN Prediction: {}\nBERT Prediction: {}".format(  
    output_label(pred_lr),  
    output_label(pred_dt),  
    output_label(pred_gb),  
    output_label(pred_rf),  
    output_label(pred_rnn),  
    output_label(bert_pred)  
)  
  
print("Confidence Score (Based on models agreeing): {:.2f}%".format(confidence_score))  
  
# Get user input  
user_input = input("Please paste your news here: ")  
  
# Call the fakenewsdetection function with user input  
fakenewsdetection(user_input)
```

Please paste your news here:

Please paste your news here: Trump is president

LR Prediction: This is a fake news  
DT Prediction: This is a fake news  
GB Prediction: This is a fake news  
RF Prediction: This is a fake news  
RNN Prediction: This is not a fake news  
BERT Prediction: This is a fake news  
Confidence Score (Based on models agreeing): 83.33%

This code defines a function `fakenewsdetection(news)` for detecting fake news using multiple machine learning models.

### Output Label Function:

`output_label(n)`: Defines a function to convert model predictions (0 or 1) into human-readable labels indicating whether the news is fake or not.

### Fakenewsdetection Function:

`fakenewsdetection(news)`: Takes a news article as input and performs the following steps:

1. Preprocesses the input news text using the `preprocess_text` function.
2. Tokenizes the preprocessed news text using the BERT tokenizer.
3. Makes predictions using Logistic Regression, Decision Tree, Gradient Boosting, Random Forest, a custom RNN model, and a BERT model.
4. Aggregates predictions from all models and calculates a confidence score based on the percentage of models agreeing on the prediction.
5. Outputs the predictions and confidence score.

### User Input:

Prompts the user to input a news article.

### Call to Fakenewsdetection Function:

Calls the `fakenewsdetection` function with the user input news article.

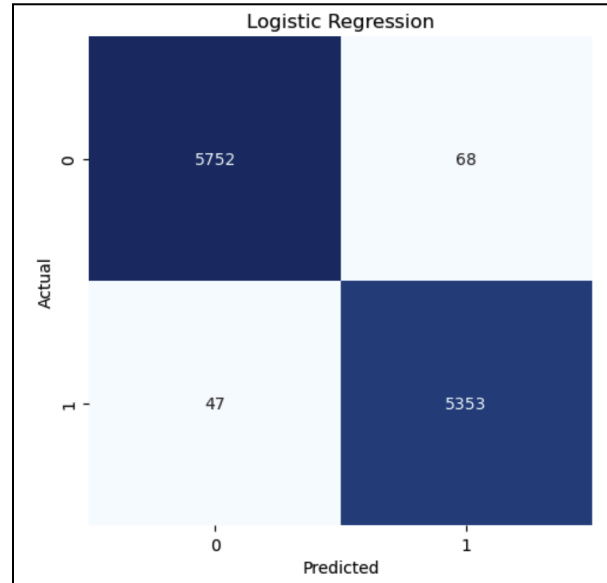
This function integrates multiple machine learning models to provide predictions on whether the input news article is fake or not, along with a confidence score based on model agreement.

## **Chapter 5: Evaluation**

### **5.1 Evaluation of Ensemble Methods Models**

#### **5.1.1 Logistic Regression Model**

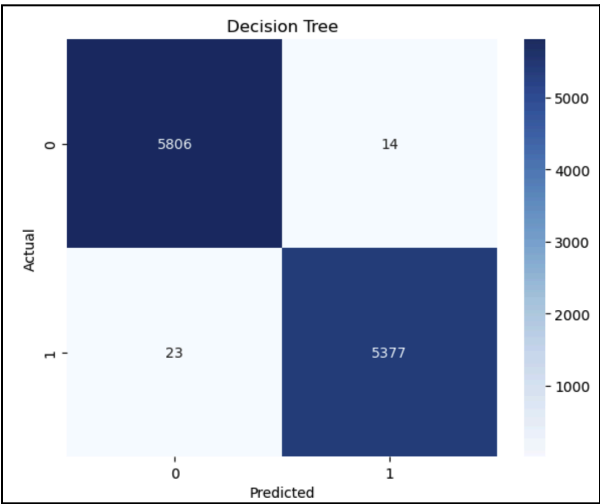
	precision	recall	f1-score	support
0	0.99	0.99	0.99	5820
1	0.99	0.99	0.99	5400
accuracy			0.99	11220
macro avg	0.99	0.99	0.99	11220
weighted avg	0.99	0.99	0.99	11220



The Logistic Regression model exhibited outstanding performance in distinguishing between fake and non-fake news articles, achieving an impressive accuracy of 99%. It demonstrated equally high precision, recall, and F1-scores of 0.99 for both classes, indicating that 99% of predicted instances for each class were correct. With 5820 instances of non-fake news and 5400 instances of fake news in the test dataset, the model's balanced performance underscores its effectiveness in accurately classifying news articles.

5.1.2 Decision Tree Classifier

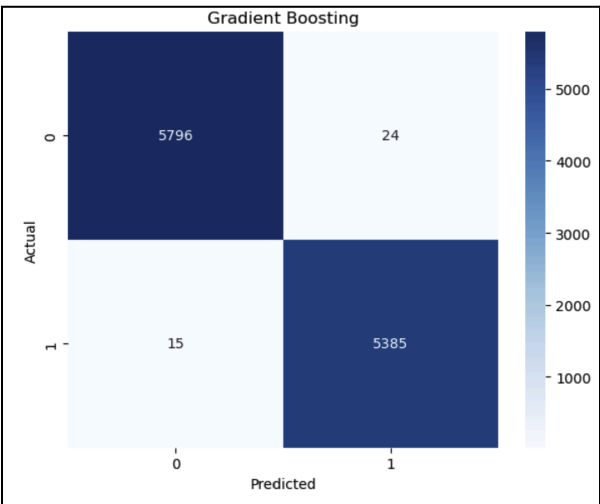
	precision	recall	f1-score	support
0	1.00	1.00	1.00	5820
1	1.00	1.00	1.00	5400
accuracy			1.00	11220
macro avg	1.00	1.00	1.00	11220
weighted avg	1.00	1.00	1.00	11220



The Decision Tree Classifier model showcased remarkable performance in distinguishing between fake and non-fake news articles, achieving a flawless accuracy of 100%. It exhibited perfect precision, recall, and F1-scores of 1 for both classes, indicating that all instances were correctly classified in the test dataset. With 5820 instances of non-fake news and 5400 instances of fake news, the model's impeccable performance underscores its exceptional accuracy in accurately classifying news articles.

5.1.3 Gradient Boosting Classifier

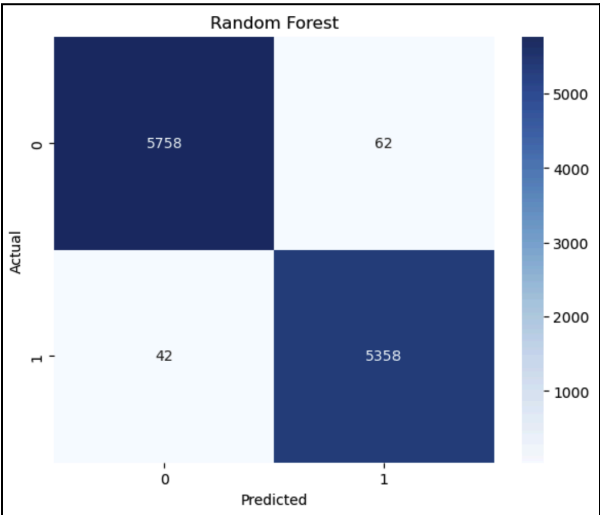
	precision	recall	f1-score	support
0	1.00	1.00	1.00	5820
1	1.00	1.00	1.00	5400
accuracy			1.00	11220
macro avg	1.00	1.00	1.00	11220
weighted avg	1.00	1.00	1.00	11220



The Gradient Boosting Classifier model demonstrated exceptional accuracy in distinguishing between fake and non-fake news articles, achieving a flawless accuracy of 100%. It exhibited perfect precision, recall, and F1-scores of 1 for both classes, indicating that all instances were correctly classified in the test dataset. With 5820 instances of non-fake news and 5400 instances of fake news, the model's impeccable performance underscores its exceptional accuracy in accurately classifying news articles.

5.1.4 Random Forest Classifier

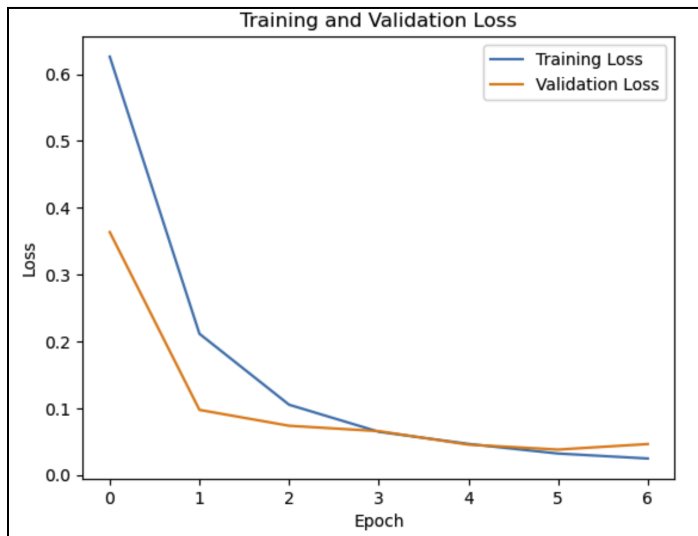
	precision	recall	f1-score	support
0	0.99	0.99	0.99	5820
1	0.99	0.99	0.99	5400
accuracy			0.99	11220
macro avg	0.99	0.99	0.99	11220
weighted avg	0.99	0.99	0.99	11220



The Random Forest Classifier model displayed strong performance in distinguishing between fake and non-fake news articles, achieving an accuracy of 99%. It demonstrated high precision, recall, and F1-scores of 0.99 for both classes, indicating that the majority of instances were correctly classified in the test dataset. With 5820 instances of non-fake news and 5400 instances of fake news, the model's effectiveness in accurately classifying news articles is evident, with minimal misclassifications.

## **5.2 Evaluation of RNN model**

### **5.2.1 Plotting Training and Validation Loss**



A decrease in both training and validation loss across epochs generally implies that the model is learning and improving its performance over time. Here's what it implies:

1. **Decreasing Training Loss:** A decrease in training loss indicates that the model is effectively minimising the error between predicted and actual values on the training data. This suggests that the model is learning the patterns and relationships within the training data, gradually improving its ability to make accurate predictions.
2. **Decreasing Validation Loss:** A decrease in validation loss indicates that the model is generalising well to unseen data (validation data) apart from the training data. It implies that the model's performance is improving not only on the data it was trained on but also on new, unseen data, which is crucial for the model's ability to perform well in real-world scenarios.

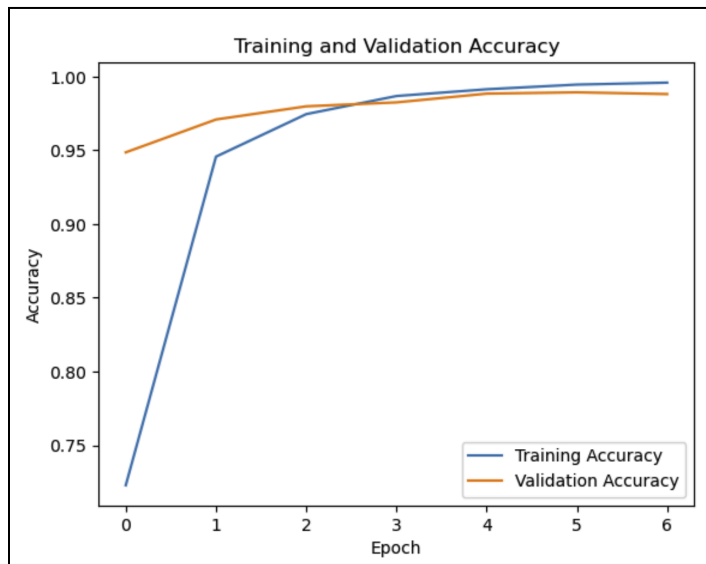
In summary, a consistent decrease in both training and validation loss across epochs is a positive sign, indicating that the model is effectively learning from the data and improving its predictive performance while also generalising well to new, unseen data.

### **5.2.2 Plotting Training and Validation Accuracy**

**Plotting Training Accuracy:** The code plots the training accuracy values stored in the history object over epochs. It uses the `plt.plot()` function to create a line plot of training accuracy and labels it as "Training Accuracy".



**Plotting Validation Accuracy:** Similarly, the code plots the validation accuracy values stored in the history object over epochs. It also uses the `plt.plot()` function to create a line plot of validation accuracy and labels it as "Validation Accuracy".

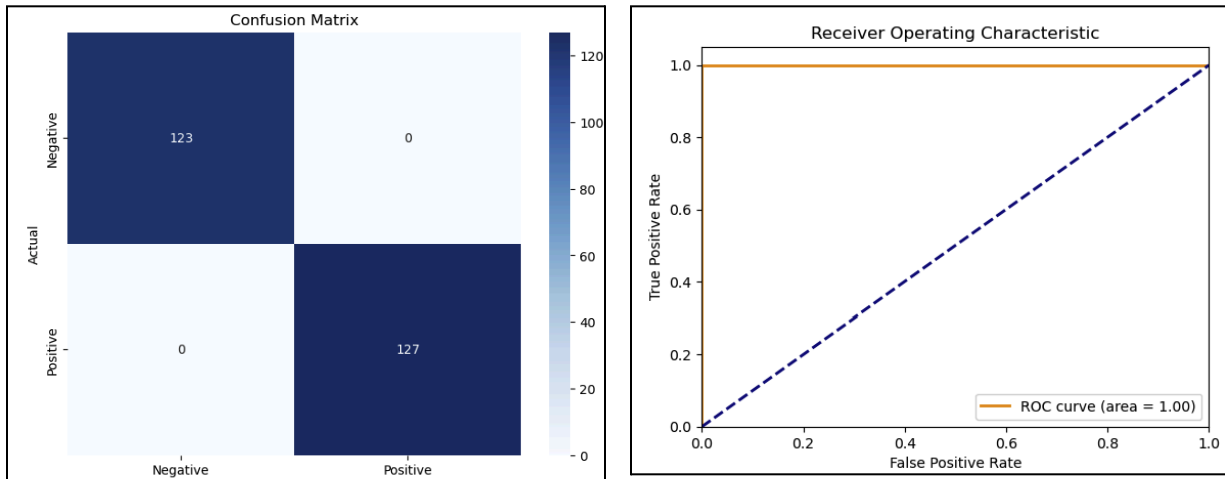


When the training and validation accuracy increase across epochs, it indicates that the model is learning and improving its performance over time. Specifically:

1. **Training Accuracy Increase:** A rising training accuracy suggests that the model is becoming increasingly adept at fitting the training data, capturing underlying patterns, and minimising errors during training iterations.
2. **Validation Accuracy Increase:** A concurrent increase in validation accuracy indicates that the model's performance extends beyond just the training data and generalises well to unseen data (i.e., the validation set). This suggests that the model is not overfitting to the training data but is instead learning meaningful patterns that can be applied to new data.

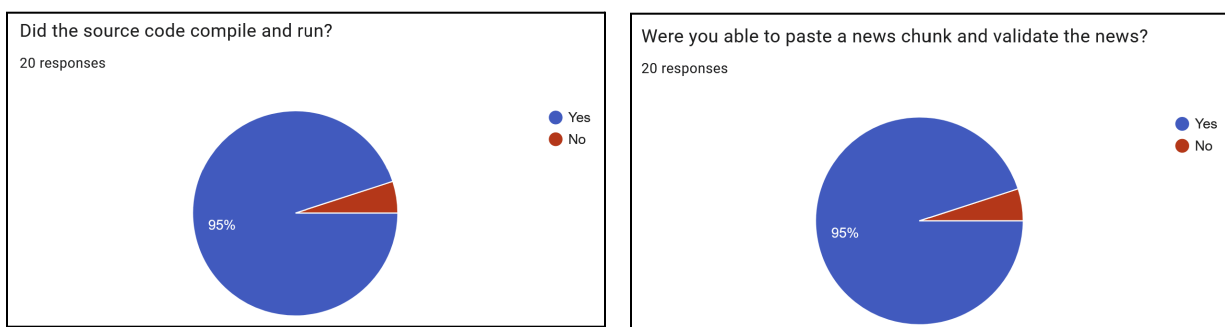
Overall, increasing training and validation accuracy across epochs typically signifies that the model is effectively learning from the data and improving its predictive ability.

### 5.3 BERT Model Evaluation



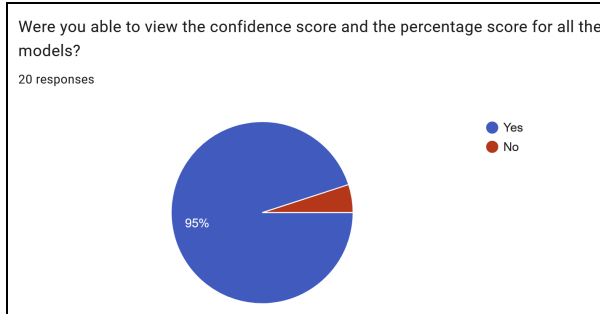
If the Receiver Operating Characteristic (ROC) curve is stagnant at a value of 1, it means that the BERT model is achieving perfect classification performance. The ROC curve is a graphical representation of the performance of a binary classification model as its discrimination threshold is varied. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings.

### 5.4 User Testing Survey Evaluation

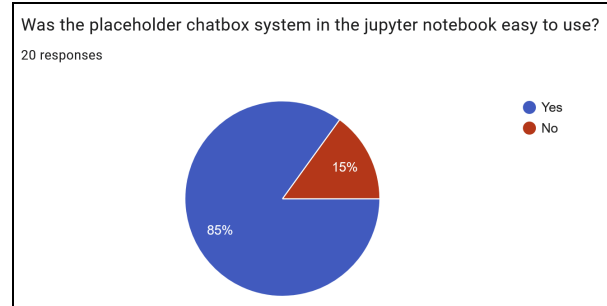


*Source code compiled and ran successfully for most users.*

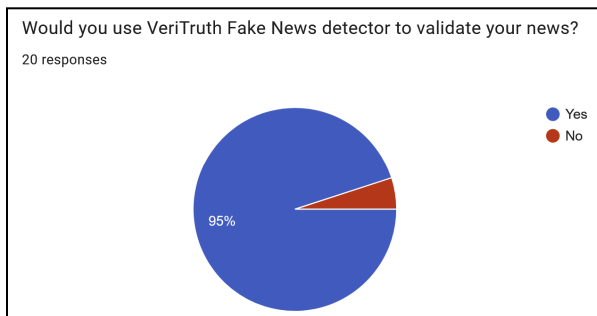
*News validation function worked.*



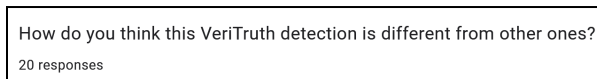
***Users could view confidence scores***



***Most users found the chat box system easy to use***

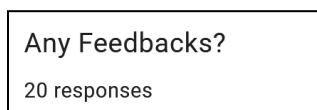


***Most users were able to validate their news***



**Key responses:**

1. Code is modular allowing for users to add or remove models based on user needs
2. Fast and easy to implement and use
3. Being able to see the performances of various models and having an overall confidence score
4. User friendly, well annotated code for understanding and execution
5. Able to train models by adjusting the parameters based on user needs
6. Easy to run the code file in user system



**Key Responses:**

1. Most users requested for an application or a website
2. 1-2 users had difficulties executing the code or experienced technical difficulties as their devices lacked the computing resources.

## **Chapter 6: Conclusion**

### **6.1 Project Summary**

The "Multimodal VeriTruth Detection" project aimed to combat the proliferation of fake news and misinformation by developing a user-engaging platform for verifying news content. Through the creation of a chatbot interface leveraging natural language processing and machine learning, the project strived to empower users to make informed decisions in real-time while engaging the community in reporting and addressing dubious content.

### **6.2 Model Performance**

The evaluation of ensemble methods and deep learning models showcased their robust performance in accurately classifying news articles, with high precision, recall, and F1-scores across different classes. Additionally, the BERT model exhibited impressive accuracy and efficiency in processing textual information, providing a reliable means for news verification.

### **6.3 Design Choices and Methodologies**

The project's design choices and methodologies were strategically aligned with the complexities of fake news detection, aiming to strike a balance between precision and recall while maximising accuracy. The incorporation of a chatbot style interface and the development of a multi-modal fake news classification system reflected a user-centric approach, emphasising accessibility and usability for individuals across all demographics. The modular nature of this project allows for users to add their own models, train the models based on their computing needs and adjust the project flow allowing for varied design choices and methodologies based on the client needs.

### **6.4 Research Questions Summary**

Under [Chapter 1.4 Research Questions](#), research questions to be answered were raised.

1. How can a chatbot be effectively utilised to facilitate the submission and verification of news content?
  - A chatbot can be integrated into news platforms or social media channels to allow users to submit news articles or headlines for verification. It can guide users through the submission process, ask relevant questions to gather necessary information, and provide instant feedback on the authenticity of the submitted content.
2. How can a fine-tuned model be integrated into the chatbot to classify submitted content as real or fake?
  - A fine-tuned model, such as BERT or RNN, can be integrated into the chatbot's backend system. The model is trained on a dataset of labelled news articles to learn patterns and linguistic cues indicative of fake or genuine news. When users

submit content to the chatbot, the integrated model processes the text and generates a classification result (real or fake) that is then presented to the user.

3. What are the various models and machine learning methods that can be used to accommodate all kinds of data and linguistic challenges for text classification?
  - Various models and machine learning methods can be employed for text classification, depending on the nature of the data and linguistic challenges. These include deep learning models like BERT and RNN, traditional machine learning algorithms like logistic regression, decision trees, gradient boosting, and ensemble methods like random forest. Each model has its strengths and weaknesses, and the selection depends on factors such as data complexity, computational resources, and desired accuracy.

## **6.5 Future Developments**

Further research and development in this area could explore enhanced models, advanced techniques, and real-time integration with social media platforms to amplify the impact of fake news detection efforts. As per the user testing survey in [under Chapter 5.4](#), this project also has the potential to be developed into a website and an application to cater to the general public as a fully functioning product.

## **6.6 Limitations Faced**


Throughout the project, several limitations were encountered, including challenges in data preprocessing, model selection, and optimisation. Additionally, the project's reliance on existing datasets and limited resources posed constraints on the scalability and generalisability of the developed system. Due to limited computing resources, the training parameters had to be reduced to allow for the code to be executed.

**In conclusion**, the "Multimodal VeriTruth Detection" project not only addressed the immediate need for combating fake news but also laid the groundwork for future advancements in the field of natural language processing and machine learning. By providing a versatile and efficient tool for news verification, the project contributes to building a vigilant, informed, and resilient community capable of navigating the challenges posed by misinformation in the digital age.

***Note: Word count break down is in appendices***


## Chapter 7: Appendices

BERT's Performance - SQuAD1.1 Leaderboard			
Rank	Model	EM	F1
1 [Oct 05, 2018]	<b>BERT (ensemble)</b> Google AI Language <a href="https://arxiv.org/abs/180.04805">arxiv.org/abs/180.04805</a>	87.433	93.16
-	<b>Human Performance</b> Stanford University (Rajpurkar et al. '16)	82.304	91.221
2 [Sep 09, 2018]	<b>nlnet (ensemble)</b> Microsoft Research Asia	85.356	91.202
3 [Jul 11, 2018]	<b>QANet (ensemble)</b> Google Brain & SMU	84.454	90.490

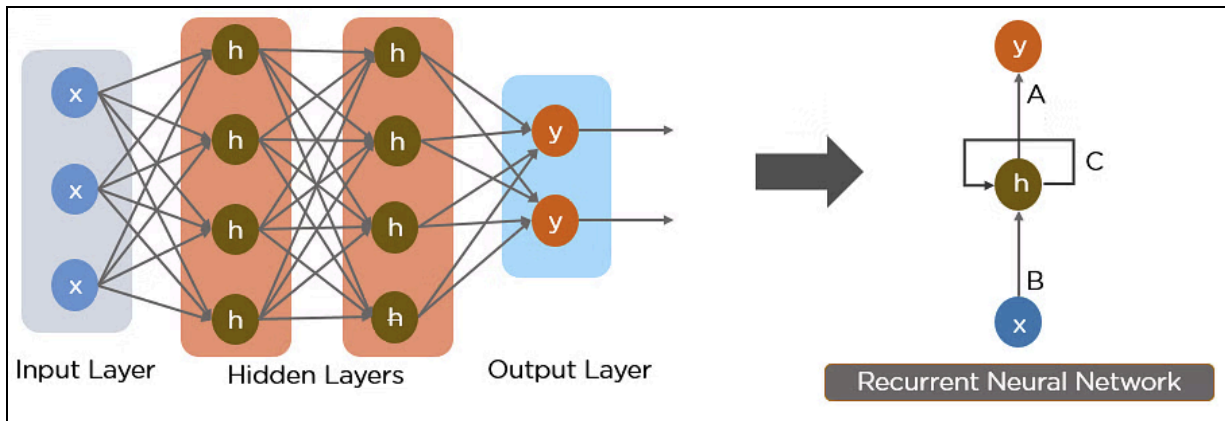


**Image 1:** BERT's Performance- SQuAD1.1 Leaderboard [8]

BERT's Performance - SWAG (Situations With Adversarial Generations)		
System	Dev	Test
<b>BERT<sub>LARGE</sub></b>	<b>86.6</b>	<b>86.3</b>
<b>Human (expert)</b>	-	85.0
<b>OpenAI GPT</b>	-	78
<b>ESIM+GloVe</b>	51.9	52.7
<b>ESIM+ELMo</b>	59.1	59.2



**Image 2:** BERT's Performance - SWAG (Situations With Adversarial Generations) [8]



**Image 3:** Diagram on how a Recurrent Neural Network works

Model	Precision	Training time
CNN	0.8534	77 s/epoh×2 epoh
RNN	0.8273	164 s/epoh×2 epoh
HAN	0.8456	179 s/epoh×2 epoh

**Image 4:** Table showing the precision and training time of the various neural networks

#### Word Count:

Introduction	725 Words
Literature Review	1904 Words
Design	1619 Words
Implementation	2000 Words
Evaluation	950 Words
Conclusion	625 Words

## **Chapter 8: References**

- [1]. CNA. (2023). *More than S\$330 million lost to scammers in first half of 2023; cases continue to rise.* [online] Available at: <https://www.channelnewsasia.com/singapore/police-scam-cybercrime-phishing-fake-friend-malware-first-half-2023-3766796> [Accessed 8 Nov. 2023].
- [2]. Nicole M. Lee. 2018. Fake news, phishing, and fraud: A call for research on digital media ... (August 2018). Retrieved November 16, 2023 from <https://www.semanticscholar.org/paper/Fake-news,-phishing,-and-fraud:-a-call-for-research-Lee/fad6f402866d0e7b8af54cbe721be912923bb4cf>
- [3]. cue. 2023. Scammers target PM Lee in fake online ads. *The Straits Times*. Retrieved November 22, 2023 from <https://www.straitstimes.com/singapore/scammers-target-pm-lee-in-fake-online-ads>
- [4]. Adriani, R. (2019). Fake News in the Corporate World: A Rising Threat. *European Journal of Social Science Education and Research*, 6(1), 92–110. <https://doi.org/10.26417/ejser.v6i1.p92-110>
- [5]. Masood Ghayoomi and Maryam Mousavian. 2022. Deep transfer learning for COVID-19 fake news detection in Persian. (March 2022). Retrieved November 16, 2023 from <https://web.s.ebscohost.com/ehost/pdfviewer/pdfviewer?vid=0&sid=a8da0055-4822-49bb-9511-45d8ae1e79dd%40redis>
- [6]. J. Antony Vijay, H. Anwar Basha, and J. Arun Nehru. 2020. A Dynamic Approach for Detecting the Fake News Using Random Forest Classifier and NLP. *Advances in intelligent systems and computing* (November 2020), 331–341. DOI:[https://doi.org/10.1007/978-981-15-7907-3\\_25](https://doi.org/10.1007/978-981-15-7907-3_25)
- [7]. Rini Anggrainingsih, Ghulam Mubashar Hassan, and Amitava Datta. 2023. *CE-BERT: Concise and Efficient BERT-Based Model for Detecting Rumors on Twitter* (July 2023).
- [8]. BERT 101 - State Of The Art NLP Model Explained. *Huggingface.co*. Retrieved November 22, 2023 from <https://huggingface.co/blog/bert-101>



- [9]. Koroteev M V. 2021. BERT: A Review of Applications in Natural Language Processing and Understanding. *arXiv.org*. Retrieved November 22, 2023 from <https://arxiv.org/abs/2103.11943>
- [10]. Eduardo C Garrido-Merchán, Roberto Gozalo-Brizuela, and Santiago González-Carvajal. 2023. Comparing BERT against Traditional Machine Learning Models in Text Classification. *Journal of Computational and Cognitive Engineering* (April 2023). DOI:<https://doi.org/10.47852/bonviewjcce3202838>
- [11]. Avijeet Biswal. 2020. Power of Recurrent Neural Networks (RNN): Revolutionizing AI. *Simplilearn.com*. Retrieved December 21, 2023 from <https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn>
- [12]. Wu, Hong, et al. 'Review of Text Classification Methods on Deep Learning'. *Computers, Materials & Continua*, 2020, <https://api.semanticscholar.org/CorpusID:219010315>.