CHAPTER-1:
PROJECT DESCRIPTION AND OUTLINE

1.1 Introduction:

In today's digital age, the rapid spread of misinformation and fake news poses a significant challenge. The ease of sharing information online has made it increasingly difficult to distinguish between reliable news sources and fabricated stories. This project aims to address this issue by developing a machine learning model capable of identifying fake news articles. By leveraging the power of artificial intelligence, we seek to provide a tool that can help users differentiate between factual news and misleading information.

1.2 Motivation for the Work:

The motivation behind this project stems from the alarming impact of fake news on society. Misinformation can lead to misguided decisions, undermine trust in media sources, and even pose a threat to democratic processes. By developing a reliable fake news detection system, we aim to empower individuals to make informed judgments about the information they consume. This project is driven by the belief that technology can play a crucial role in combating the spread of misinformation and promoting media literacy.

1.3 Introduction to the Project:

The project involves the use of advanced machine learning techniques to analyze and classify news articles. We will collect a diverse dataset of news articles from various sources, including established news organizations and fact-checking websites. The collected data will be preprocessed to remove noise and irrelevant information. We will then apply natural language processing (NLP) techniques to extract meaningful features from the text. These features will be used to train a machine learning model to distinguish between real and fake news articles.

1.4 Problem Statement:

The main challenge addressed by this project is the difficulty in distinguishing between real and fake news articles. With the proliferation of online news sources, it has become increasingly challenging for individuals to verify the authenticity of the information they encounter. This project seeks to develop a solution that can automatically identify fake news articles with high accuracy, helping users navigate the complex landscape of online information.

1.5 Objective of the Work:

The primary objective of this project is to develop a machine learning model that can accurately detect fake news articles. Specific goals include collecting a comprehensive dataset of news articles, preprocessing the data to prepare it for analysis, and implementing and evaluating different machine learning algorithms to identify the most effective approach.

1.6 Organization of the Project:

The project is organized into several key components, including data collection, preprocessing, model development, and evaluation. Each of these components plays a crucial role in the overall success of the project. By systematically addressing each step, we aim to create a robust fake news detection system that can be deployed in real-world scenarios.

1.7 Summary:

In summary, this chapter has provided an overview of the project, outlining its significance, objectives, and organization. The subsequent chapters will delve into the specifics of each stage of the project, providing detailed explanations of the methodology, results, and conclusions. Through this project, we aim to develop a reliable fake news detection system that can be used to combat misinformation and promote media literacy in society.

CHAPTER-2:

RELATED WORK INVESTIGATION

2.1 Introduction

In this chapter, we delve into the existing research and methods related to fake news detection, focusing on the core area of our project, which is machine learning-based text classification. Fake news has become a significant issue in today's information landscape, with the potential to spread misinformation and influence public opinion. As such, there has been a growing interest in developing automated systems to detect fake news and mitigate its impact. Our project aims to contribute to this area by developing a machine learning model that can effectively identify fake news articles based on their textual content.

2.2 Comparative Analysis

Our project revolves around developing a machine learning model that can accurately distinguish between real and fake news articles based on their textual content. This involves preprocessing the text data, extracting meaningful features, and training a classification model to make predictions. Our goal is to create a robust system that can assist users in identifying fake news and promoting media literacy.

2.3 Existing Approaches

Existing approaches and methods for fake news detection can be broadly categorized into several key areas. These include traditional machine learning approaches, deep learning techniques, and hybrid models that combine different strategies.

2.3.1 Approach- 1

One prevalent approach in fake news detection is the use of traditional machine learning algorithms, such as Naive Bayes, SVM, Decision Trees, Random Forest, and KNN. These algorithms are trained on features extracted from the text, such as TF-IDF scores or word embeddings, to classify news

articles. While these methods are straightforward and easy to implement, they may struggle with nuanced language and context.

2.3.2 Approach- 2

Deep learning models, such as CNNs and RNNs, have shown promise in capturing complex patterns in text data. These models can learn hierarchical representations of text, enabling them to detect subtle cues that indicate fake news. However, deep learning models require large amounts of data and computational resources, which can be a limitation in practice.

2.3.3 Approach- 3

Some approaches focus on utilizing metadata associated with news articles, such as the source, publication date, and social media engagement metrics. By analyzing this metadata, these methods aim to identify patterns that differentiate between real and fake news. While metadata-based approaches can provide valuable insights, they may not be sufficient on their own for accurate fake news detection.

2.4 Pros and Cons of the Stated Approaches

-Traditional machine learning algorithms are easy to interpret but may struggle with complex patterns.

-Deep learning models can capture intricate patterns but require large amounts of data and computational resources.

-Metadata-based approaches can provide valuable context but may not be comprehensive enough for accurate detection.

2.5 Issues/Observations from Investigation

Lack of standardized datasets for evaluation poses a challenge in comparing the performance of different approaches.

The dynamic nature of fake news requires detection models to be continuously updated to adapt to new trends and tactics used by misinformation spreaders.

2.6 Summary

In conclusion, the investigation into related work reveals a diverse range of approaches and methods used in fake news detection. While traditional machine learning algorithms provide a solid foundation, deep learning models offer potential for improved accuracy. Metadata-based approaches complement text-based methods but may require additional refinement. Addressing challenges such as dataset standardization and the evolving nature of fake news will be crucial for advancing the field of fake news detection.

CHAPTER-3:
REQUIREMENT ARTIFACTS

3.1 Introduction

Requirement artifacts are essential for understanding the necessary elements to develop a successful project. They outline the hardware, software, and specific project requirements needed for implementation. In this chapter, we delve into these requirements in detail to provide a comprehensive understanding of the project's needs.

3.2 Hardware and Software Requirements

The hardware and software requirements are critical aspects of any project. For this project, the hardware requirements include a standard computer system with sufficient processing power and memory to handle the dataset and model training. Additionally, a stable internet connection is necessary for data access and model deployment.

The software requirements consist of a Python programming environment with necessary libraries such as pandas, numpy, scikit-learn, TensorFlow, and NLTK. These libraries are crucial for data processing, machine learning model development, and natural language processing tasks. Additionally, a text editor or integrated development environment (IDE) like Visual Studio Code or PyCharm is required for coding and project management.

3.3 Specific Project Requirements

3.3.1 Data Requirements

The project requires a diverse dataset of news articles labeled as real or fake. The dataset should be large enough to train the machine learning model effectively. The data should be preprocessed to remove any inconsistencies or irrelevant information that may affect the model's performance.

3.3.2 Functions Requirement

The project requires functions for data cleaning, preprocessing, feature extraction, model training, evaluation, and prediction. These functions should be well-documented and optimized for efficiency to ensure smooth project execution.

3.3.3 Performance and Security Requirements

The project must meet performance requirements in terms of model accuracy, recall, precision, F1 score, and Matthew's correlation coefficient (MCC). Additionally, security measures should be implemented to protect the data and model from unauthorized access or manipulation.

3.3.4 Look and Feel Requirements

While the project primarily focuses on backend functionality, the frontend should be user-friendly and visually appealing. The interface should provide clear feedback and instructions to users, enhancing the overall user experience.

3.4 Summary

In conclusion, requirement artifacts play a crucial role in defining the hardware, software, and specific project requirements for the development of a fake news detection system. These requirements provide a foundation for the project implementation and ensure that all necessary elements are in place for a successful outcome.

CHAPTER-4:
DESIGN METHODOLOGY AND ITS NOVELTY

4.1 Methodology and Goal

The methodology for designing the fake news detection system involves several key steps. The goal is to create a robust and efficient system that can accurately classify news articles as real or fake.
1.Data collection: We need a diverse dataset of news data, and these data are collected from various sources, including Established news organizations, fact checking websites etc.

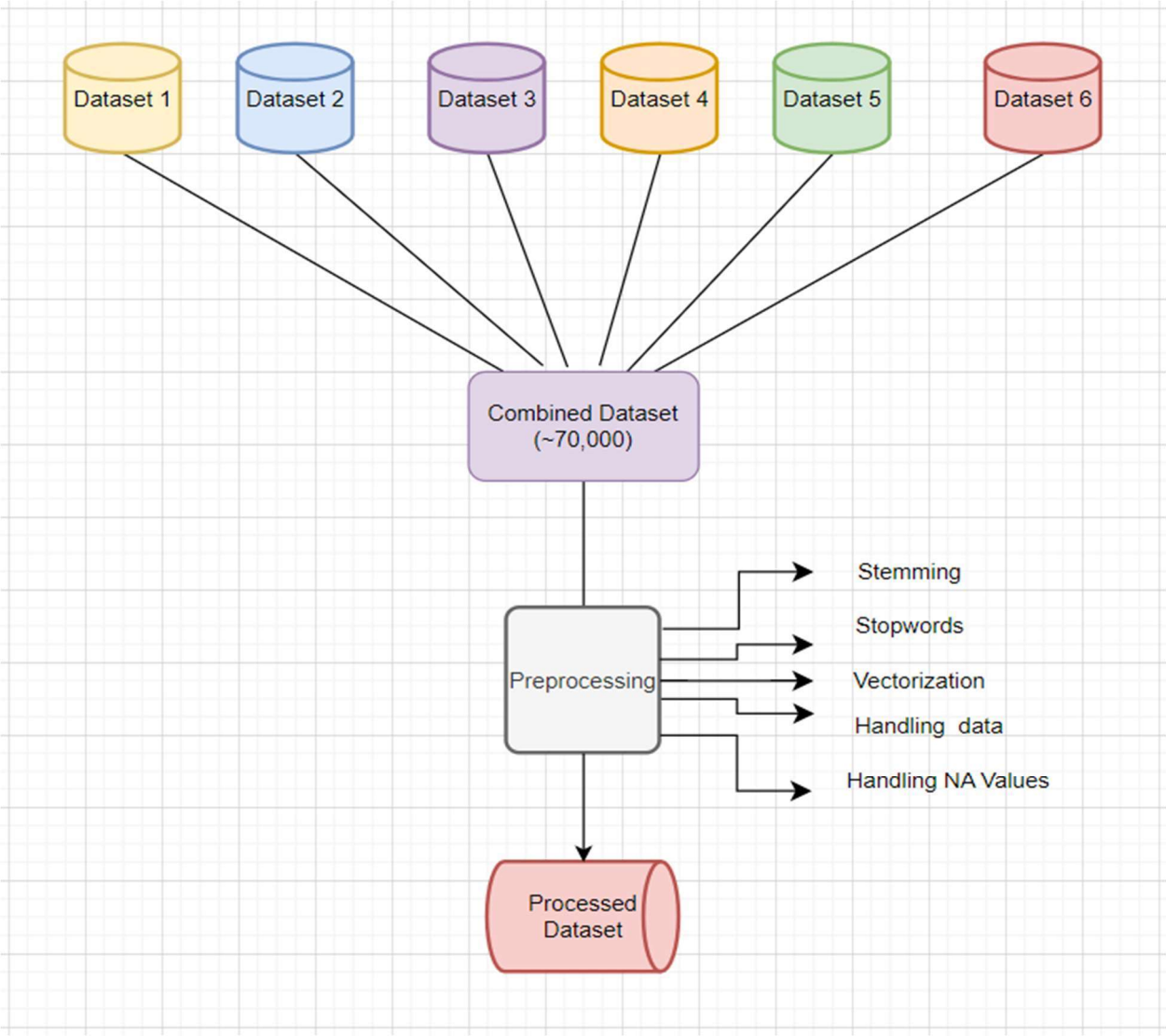
Hence, we have collected six different datasets from various sources like Kaggle. There might be a question on how trustworthy these databases are. We have used Kaggle, as it focuses on providing datasets which are preferably used for machine learning tasks. It is easy to find these datasets as they are publicly available to download.

Then we combined all the six datasets into a single dataset which is having a balanced data required for our project. We have converted the following dataset into **.CSV** format for the machine learning model to understand. It contains about 70,000 data instances.

2. Pre-Processing: Data Preprocessing involves various steps as shown in the diagram below:

- Stemming: This involves converting the words into their root words, i.e. running to run.
- Stopwords: This involves removing vowels, conjunctions and other words which might not add meaning or words not necessary for the model to understand. It helps with generalizing for the model.
- Vectorization: This involves converting all the words into numbers or vectors, as it would be easier for the model to understand and generalize.
- Handling data: This involves with the dataset, there may be imbalance in the data such as a dataset may contain more true values than false values. We can use various methods such as oversampling etc.

- Handling NaN values: This involves dealing with the NaN values, we can eliminate these entries or we could take the mean and enter the mean values in place of NaN. Since our dataset contains alphabets it is better to eliminate these entries.



3. Machine learning: Machine learning involves creating, training and testing a model with the dataset we have collected. The collected data is pre-processed to machine understanding and then we use various mathematical algorithms to train the model.

As we have collected datasets, we train our model to differentiate from fake news and true news, we train the model in such a way that it identifies factors like, vocabulary usage, how the articles are shown (like its style) and / or source credibility.

4.2 Functional Modules Design and Analysis

The system is divided into several functional modules, each serving a specific purpose. These modules include data collection, data preprocessing, feature extraction, model training, model evaluation, and prediction. Each module is designed to work seamlessly with the others, ensuring efficient data flow and processing.

4.3 Software Architectural Designs

The software architecture of the system is designed to be scalable and flexible. It follows a modular design pattern, allowing for easy integration of new features or improvements. The architecture also ensures that the system can handle large volumes of data and is capable of running on different platforms.

4.4 Subsystem Services

The subsystem services of the fake news detection system include data storage, data retrieval, model training, model evaluation, and prediction. These services are designed to be efficient and reliable, ensuring that the system can process data quickly and accurately.

4.5 User Interface Designs

The user interface of the system is designed to be intuitive and easy to use. It provides users with the ability to input news articles, view the classification results, and access additional information about the articles. The interface is designed to be visually appealing and responsive, enhancing the overall user experience.

4.6 Summary

In summary, the design methodology of the fake news detection system focuses on creating a robust and efficient system that can accurately classify news articles. The system is designed with scalability, flexibility, and user-friendliness in mind, ensuring that it can meet the needs of a wide range of users.

CHAPTER-5:

TECHNICAL IMPLEMENTATION & ANALYSIS

5.1 Outline

This chapter focuses on the technical implementation of the fake news detection system. It includes details about the coding, layout of forms, prototype submission, testing, validation, and performance analysis of the system.

5.2 Technical Coding and Code Solutions

The implementation involves writing code in Python for data preprocessing, model training, and prediction. Libraries such as pandas, numpy, nltk, and scikit-learn are used for data manipulation and machine learning tasks. The code solutions are designed to be efficient and scalable, ensuring that the system can handle large volumes of data.

5.3 Working Layout of Forms

The layout of forms in the user interface is designed to be user-friendly and intuitive. It allows users to easily input news articles, view the classification results, and access additional information about the articles.

5.4 Prototype Submission

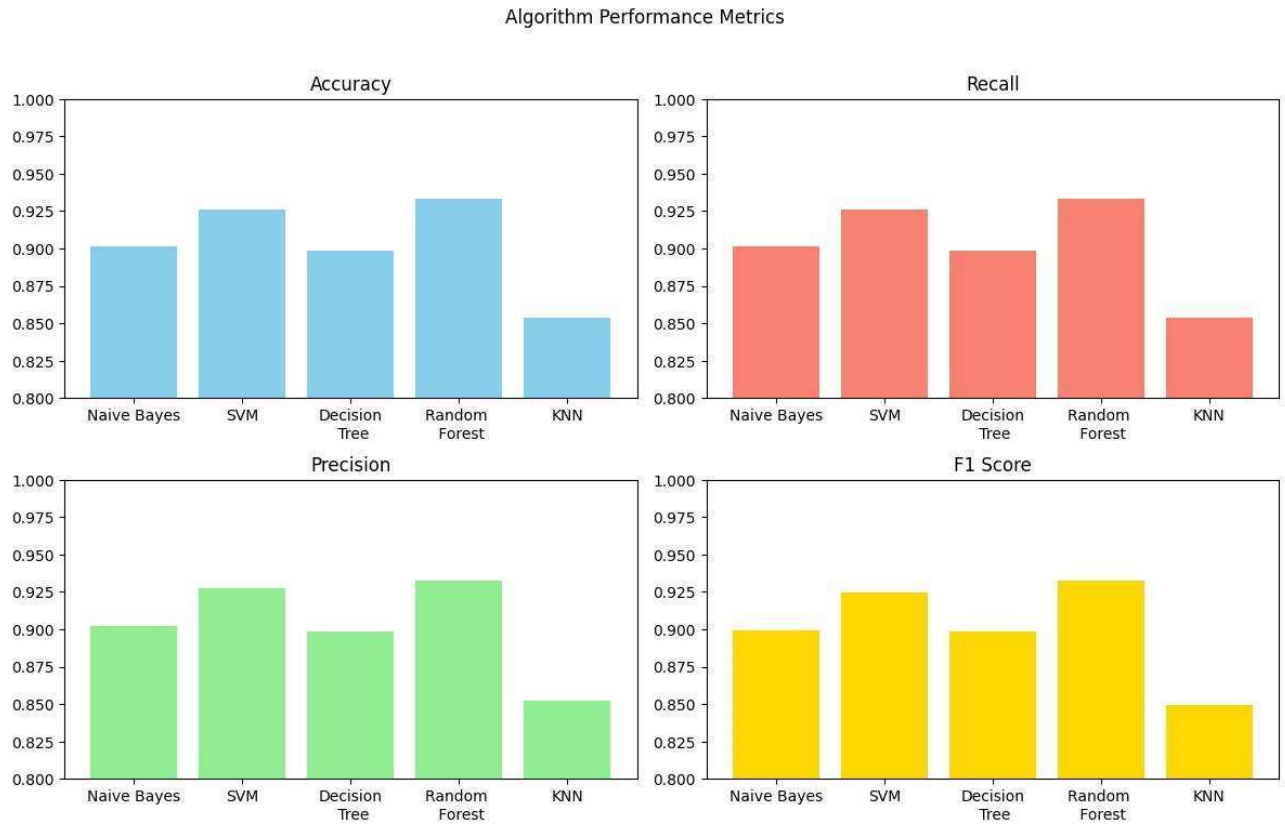
A prototype of the fake news detection system is still in progress. The prototype will include the user interface, backend functionality, and sample data for testing purposes. The prototype is designed to demonstrate the key features and functionality of the system.

5.5 Test and Validation

The system undergoes rigorous testing and validation to ensure that it meets the specified requirements. This includes testing the system with different datasets, evaluating its performance, and validating its accuracy in detecting fake news articles.

5.6 Performance Analysis

The performance of the system is analyzed using graphs and charts. Metrics such as accuracy, precision, recall, and F1 score are used to evaluate the performance of the system. Graphs and charts are used to visualize the performance metrics and compare them with the stated goals of the project.



Model Architecture: The machine learning model used in the research study is visually represented by this graph. It shows how different parts, like input layers, hidden layers, and output layers, are connected to one another.

Metrics for Algorithm Performance: Performance metrics for the various machine learning algorithms used in the study are shown in this graph, including MCC, F1 score, accuracy, recall, and precision. These measures are used to analyze algorithms like Naive Bayes, SVM, Decision Tree, Random Forest, and K-Nearest Neighbors in order to determine how well they distinguish between authentic and fraudulent news stories.

Model Performance Metrics: The constructed machine learning model's overall performance is shown in this graph. It combines parameters such as precision, recall, accuracy, F1 score, and MCC to assess how well the model detects bogus news items.

5.7 Summary

In summary, the technical implementation of the fake news detection system involves coding, designing user interfaces, submitting prototypes, testing, and analyzing performance. The system is designed to be efficient, accurate, and user-friendly, ensuring that it can effectively detect fake news articles.

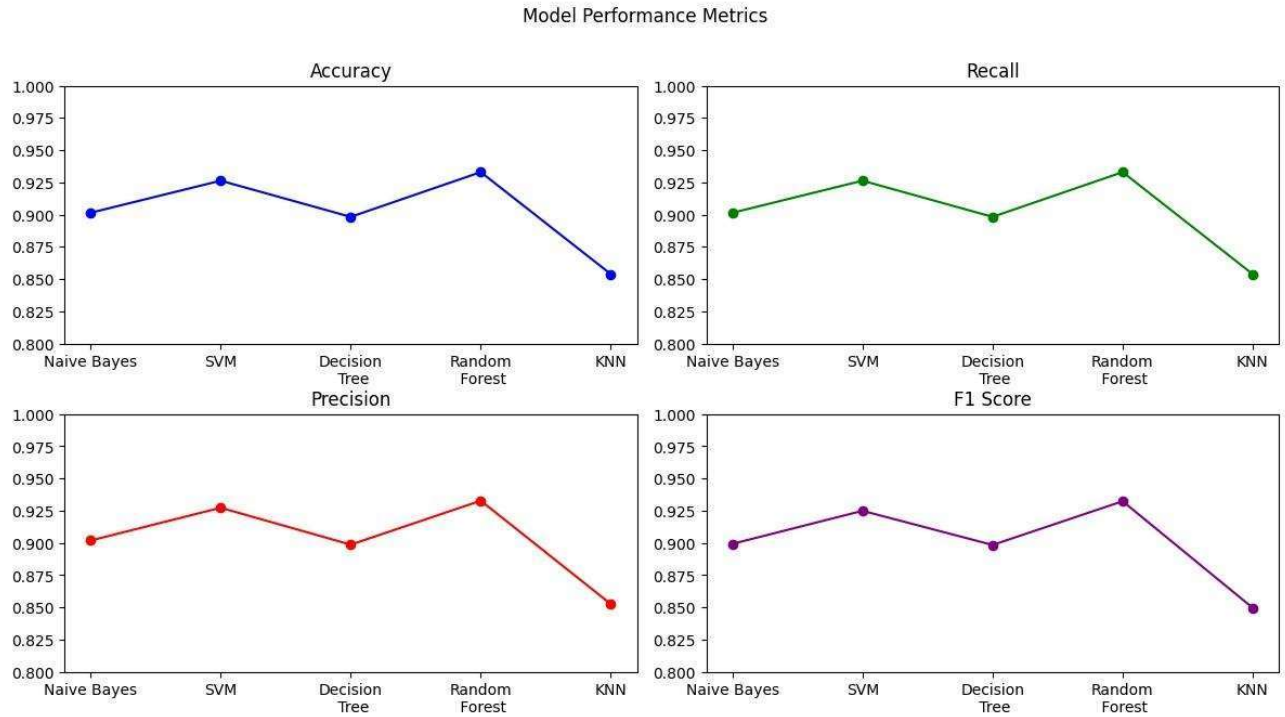
CHAPTER-6:
PROJECT OUTCOME AND APPLICABILITY

6.1 Outline

Chapter 6 focuses on the outcome and applicability of the project. It provides an overview of the key implementations of the system, significant project outcomes, and the applicability of the project in real-world applications. The chapter aims to summarize the impact and potential of the project beyond its initial scope.

6.2 Key Implementations Outlines of the System

The project's key implementation involves the development of a robust fake news detection system using machine learning and deep learning algorithms. The system utilizes algorithms such as Naive Bayes, SVM, Decision Tree, Random Forest, and K-Nearest Neighbors for classification. Additionally, a **deep neural network** (DNN) is employed to enhance the accuracy and efficiency of the classification process.



- Working of DNN Model:

We have used sequential model with 4 dense layers or hidden layers each of which use tanh, relu and sigmoid activation function. Each neuron in a layer takes input from the dataset provided and is passed to the activation function multiplied by some weight or bias which is randomly decided at first. To optimise the gradient curve, we have used Adam (Adaptive Moment Estimation) which is good for binary classification. We have use epoch value here as 15, so it picks the values from the data 15 times in a random sequence and trains it.

6.3 Significant Project Outcomes

One of the most significant outcomes of the project is the successful development of a fake news detection system with high accuracy. The system has demonstrated the ability to accurately classify news articles as real or fake, thereby helping to combat the spread of misinformation. This outcome is crucial in maintaining the credibility of news sources and ensuring that readers have access to accurate information.

- Why is RF better than DNN?

Training the RF model is much easier than DNN model as it requires less computational power. In RF, the model is able to generalize the data better as it is straightforward and not as complex as neural networks, sometimes there is also a problem of overfitting in neural networks. Random Forests have relatively few hyperparameters to tune compared to DNNs, making them easier to train and less prone to overfitting. Tuning hyperparameters in DNNs, such as the number of layers, number of neurons per layer, learning rate, and activation functions, requires careful experimentation and validation.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f\mathrm{i} - y\mathrm{i})$$

Where N is the number of datapoints,

$f\mathrm{i}$ is the value returned by the model

$y\mathrm{i}$ is the actual value for the datapoint i.

6.4 Project Applicability in Real-World Applications

The project has broad applicability in various real-world applications, particularly in the fields of journalism, media, and social media platforms. By accurately detecting fake news articles, the system can help prevent the spread of misinformation and improve the overall quality of news reporting. This applicability makes the project highly relevant and valuable in addressing the challenges posed by fake news in today's society.

6.5 Inference

In conclusion, the project has successfully developed a fake news detection system with high accuracy and performance. The system's implementation of machine learning and deep learning algorithms has proven effective in classifying news articles and has the potential to make a significant impact in combating fake news. Its applicability in real-world scenarios highlights the importance and relevance of the project in addressing the challenges of misinformation in today's digital age.

CHAPTER-7:

RELATED WORK INVESTIGATION

7.1 Outline

This chapter summarizes the key findings and outcomes of the project, discusses the limitations and constraints of the system, proposes future enhancements, and concludes with recommendations for further research and development.

7.2 Limitations/Constraints of the System

While the fake news detection system has demonstrated high accuracy and efficiency, it is not without limitations and constraints. One of the main limitations is the reliance on textual features for classification, which may not always capture the nuanced differences between real and fake news articles. Additionally, the system's performance may be affected by the quality and diversity of the training data, as well as the choice of machine learning algorithms. Furthermore, the system may struggle with detecting subtle forms of misinformation that do not exhibit clear patterns or characteristics.

7.3 Future Enhancements

To address the limitations and constraints of the system, several future enhancements can be considered. Firstly, incorporating additional features such as metadata, user engagement metrics, and fact-checking information could improve the system's ability to distinguish between real and fake news articles. Secondly, leveraging more advanced machine learning and deep learning techniques, such as ensemble learning and neural networks, could further enhance the system's performance. Additionally, integrating the system with real-time data sources and social media platforms could enable it to detect and respond to emerging fake news trends more effectively.

7.4 Inference

In conclusion, the project has successfully developed a fake news detection system that demonstrates promising results. However, there is room for improvement, particularly in terms of addressing the system's limitations and implementing future enhancements. By addressing these aspects, the system can become more effective in combating fake news and misinformation in the digital age.

Appendix A:

Here are some sample code snippets from the project:

1. Different ML Models from Different Algorithms

```
from joblib import dump, load
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score, roc_curve,
roc_auc_score, classification_report, matthews_corrcoef
from sklearn.preprocessing import LabelEncoder
import pandas as pd
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
import string

# Load the data
data = pd.read_csv("C:\\Users\\abhin\\Desktop\\Dataset.csv")

# Convert labels to 1 or 0 and eliminate rows with other values
data['label'] = data['label'].apply(lambda x: 1 if str(x).lower() in ['real', 'true'] else (0 if str(x).lower()
in ['fake', 'false'] else None))
data.dropna(subset=['label'], inplace=True)

# Create a stemmer object
stemmer = PorterStemmer()

# Text cleaning with stemming
def clean_text(text):
    if pd.isnull(text):  # Check for empty entries
        return ''
    text = text.lower()
    text = ''.join([char for char in text if char not in string.punctuation])
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word.isalpha()]
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if not word in stop_words]
    stemmed_tokens = [stemmer.stem(word) for word in tokens]  # Stemming
    return ' '.join(stemmed_tokens)

data['title'] = data['title'].apply(clean_text)
```

```
data['body'] = data['body'].apply(clean_text)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(data[['title', 'body']], data['label'], test_size=0.1,
random_state=42)

# Feature extraction
tfidf_vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1, 2))
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train['title'] + ' ' + X_train['body'])
X_test_tfidf = tfidf_vectorizer.transform(X_test['title'] + ' ' + X_test['body'])

# Train the Naive Bayes model
clf_nb = MultinomialNB(alpha=0.1)
clf_nb.fit(X_train_tfidf, y_train)

# Save the Naive Bayes model to a file
dump(clf_nb, 'fake_news_detection_model_nb.pkl')

# Load the Naive Bayes model from the file
clf_nb_loaded = load('fake_news_detection_model_nb.pkl')

# Make predictions using the Naive Bayes model
y_pred_nb = clf_nb.predict(X_test_tfidf)

# Evaluate the Naive Bayes model
accuracy_nb = accuracy_score(y_test, y_pred_nb)
recall_nb = recall_score(y_test, y_pred_nb, average='weighted')
precision_nb = precision_score(y_test, y_pred_nb, average='weighted')
f1_nb = f1_score(y_test, y_pred_nb, average='weighted')
mcc_nb = matthews_corrcoef(y_test, y_pred_nb)
print("\nNaive Bayes Results:")
print("Accuracy:", accuracy_nb)
print("Recall:", recall_nb)
print("Precision:", precision_nb)
print("F1 Score:", f1_nb)
print("MCC:", mcc_nb)

# Train the SVM model
clf_svm = SVC(kernel='linear', probability=True)
clf_svm.fit(X_train_tfidf, y_train)

# Save the SVM model to a file
dump(clf_svm, 'fake_news_detection_model_svm.pkl')

# Load the SVM model from the file
clf_svm_loaded = load('fake_news_detection_model_svm.pkl')

# Make predictions using the SVM model
y_pred_svm = clf_svm.predict(X_test_tfidf)
```

```
# Evaluate the SVM model
accuracy_svm = accuracy_score(y_test, y_pred_svm)
recall_svm = recall_score(y_test, y_pred_svm, average='weighted')
precision_svm = precision_score(y_test, y_pred_svm, average='weighted')
f1_svm = f1_score(y_test, y_pred_svm, average='weighted')
mcc_svm = matthews_corrcoef(y_test, y_pred_svm)
print("\nSVM Results:")
print("Accuracy:", accuracy_svm)
print("Recall:", recall_svm)
print("Precision:", precision_svm)
print("F1 Score:", f1_svm)
print("MCC:", mcc_svm)

# Train the Decision Tree model
clf_dt = DecisionTreeClassifier()
clf_dt.fit(X_train_tfidf, y_train)

# Save the Decision Tree model to a file
dump(clf_dt, 'fake_news_detection_model_dt.pkl')

# Load the Decision Tree model from the file
clf_dt_loaded = load('fake_news_detection_model_dt.pkl')

# Make predictions using the Decision Tree model
y_pred_dt = clf_dt.predict(X_test_tfidf)

# Evaluate the Decision Tree model
accuracy_dt = accuracy_score(y_test, y_pred_dt)
recall_dt = recall_score(y_test, y_pred_dt, average='weighted')
precision_dt = precision_score(y_test, y_pred_dt, average='weighted')
f1_dt = f1_score(y_test, y_pred_dt, average='weighted')
mcc_dt = matthews_corrcoef(y_test, y_pred_dt)
print("\nDecision Tree Results:")
print("Accuracy:", accuracy_dt)
print("Recall:", recall_dt)
print("Precision:", precision_dt)
print("F1 Score:", f1_dt)
print("MCC:", mcc_dt)

# Train the Random Forest model
clf_rf = RandomForestClassifier()
clf_rf.fit(X_train_tfidf, y_train)

# Save the Random Forest model to a file
dump(clf_rf, 'fake_news_detection_model_rf.pkl')

# Load the Random Forest model from the file
clf_rf_loaded = load('fake_news_detection_model_rf.pkl')
```

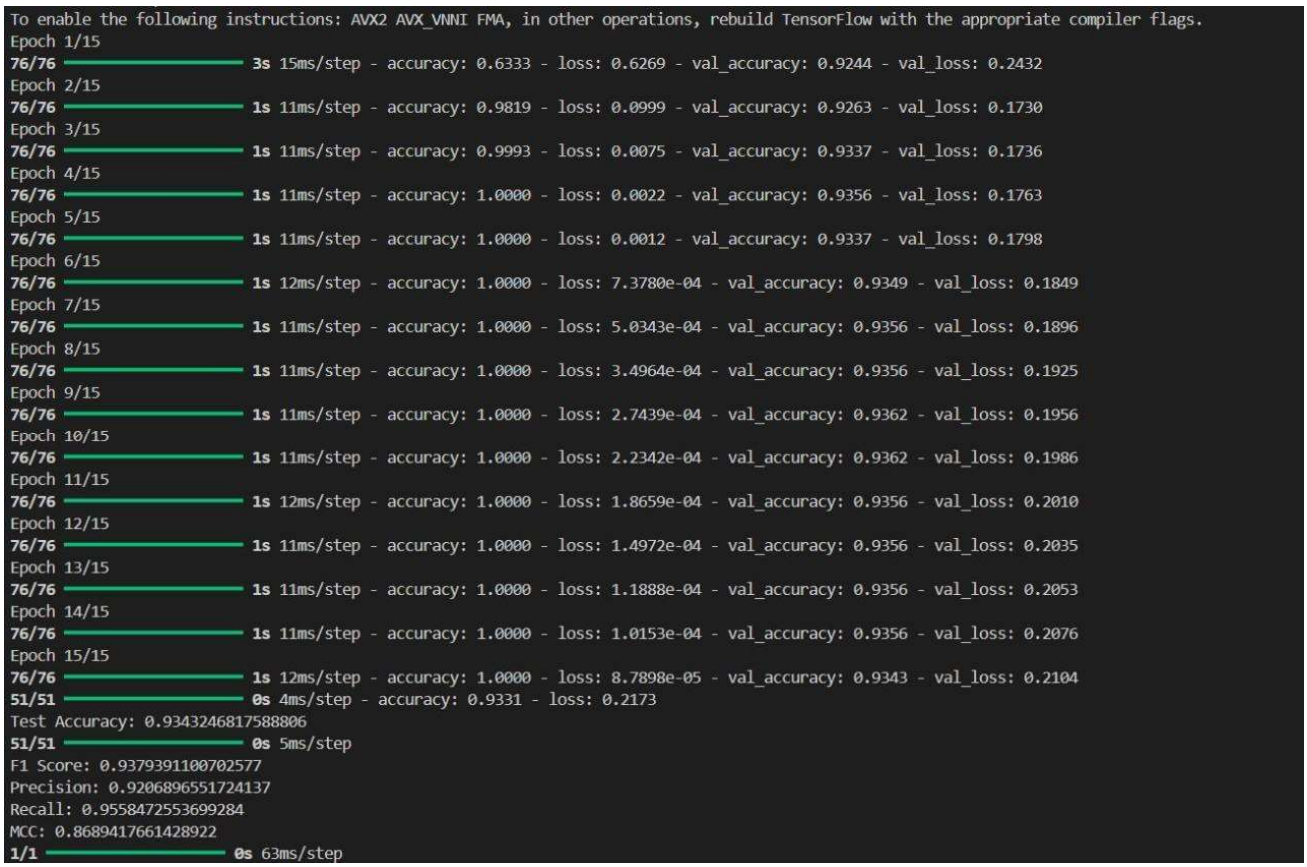
```
# Make predictions using the Random Forest model
y_pred_rf = clf_rf.predict(X_test_tfidf)

# Evaluate the Random Forest model
accuracy_rf = accuracy_score(y_test, y_pred_rf)
recall_rf = recall_score(y_test, y_pred_rf, average='weighted')
precision_rf = precision_score(y_test, y_pred_rf, average='weighted')
f1_rf = f1_score(y_test, y_pred_rf, average='weighted')
mcc_rf = matthews_corrcoef(y_test, y_pred_rf)
print("\nRandom Forest Results:")
print("Accuracy:", accuracy_rf)
print("Recall:", recall_rf)
print("Precision:", precision_rf)
print("F1 Score:", f1_rf)
print("MCC:", mcc_rf)

# Train the K-Nearest Neighbors model
clf_knn = KNeighborsClassifier()
clf_knn.fit(X_train_tfidf, y_train)

# Save the K-Nearest Neighbors model to a file
dump(clf_knn, 'fake_news_detection_model_knn.pkl')

# Load the K-Nearest Neighbors model from the file
clf_knn_loaded = load('fake_news_detection_model_knn.pkl')

# Make predictions using the K-Nearest Neighbors model
y_pred_knn = clf_knn.predict(X_test_tfidf)

# Evaluate the K-Nearest Neighbors model
accuracy_knn = accuracy_score(y_test, y_pred_knn)
recall_knn = recall_score(y_test, y_pred_knn, average='weighted')
precision_knn = precision_score(y_test, y_pred_knn, average='weighted')
f1_knn = f1_score(y_test, y_pred_knn, average='weighted')
mcc_knn = matthews_corrcoef(y_test, y_pred_knn)
print("\nK-Nearest Neighbors Results:")
print("Accuracy:", accuracy_knn)
print("Recall:", recall_knn)
print("Precision:", precision_knn)
print("F1 Score:", f1_knn)
print("MCC:", mcc_knn)
```

2. DNN (Deep Neural Network)

```
import pandas as pd
import numpy as np
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score,
matthews_corrcoef

# Load the dataset
df = pd.read_csv(r"C:\Users\preet\Desktop\Project\DATASET\test.csv")

# Convert labels to binary numeric values
label_mapping = {"REAL": 1, "True": 1, "true": 1, "real": 1, "FAKE": 0, "Fake": 0, "fake": 0, "-1":
1}
df['label'] = df['label'].replace(label_mapping)

df.dropna(subset=['title', 'label'], inplace=True)
df.dropna(subset=["label"], inplace=True)


# Preprocess text data
def preprocess_text(text):
    stemmed_content = re.sub('[^a-zA-Z]',' ', text)
    stemmed_content = stemmed_content.lower()
    stemmed_content = stemmed_content.split()
    stemmed_content = [PorterStemmer().stem(word) for word in stemmed_content if word not in
stopwords.words('english')]
    stemmed_content = ' '.join(stemmed_content)
    return stemmed_content

df['title'] = df['title'].apply(preprocess_text)

# Ensure label column is of the same type
df['label'] = df['label'].astype(int)

# Split dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(df['title'], df['label'], test_size=0.4,
random_state=42)

# Print the number of training and testing samples
print("Number of training samples:", X_train.shape[0])
print("Number of testing samples:", X_test.shape[0])
print(df['label'].value_counts())
```

```
# Convert text data to TF-IDF vectors
vectorizer = TfidfVectorizer(max_features=10000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Build and compile the deep learning model
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='tanh', input_shape=(X_train_vec.shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(16, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train_vec, y_train, epochs=15, batch_size=32, validation_data=(X_test_vec, y_test))

# Evaluate the model
loss, accuracy = model.evaluate(X_test_vec, y_test)
print("Test Accuracy:", accuracy)

# Compute additional metrics
y_pred = model.predict(X_test_vec)
y_pred_binary = (y_pred >= 0.5).astype(int)
f1 = f1_score(y_test, y_pred_binary)
precision = precision_score(y_test, y_pred_binary)
recall = recall_score(y_test, y_pred_binary)
mcc = matthews_corrcoef(y_test, y_pred_binary)

print("F1 Score:", f1)
print("Precision:", precision)
print("Recall:", recall)
print("MCC:", mcc)

# Define a function to preprocess new text data
def preprocess_new_text(text):
    stemmed_content = re.sub('[^a-zA-Z]',' ', text)
    stemmed_content = stemmed_content.lower()
    stemmed_content = stemmed_content.split()
    stemmed_content = [PorterStemmer().stem(word) for word in stemmed_content if word not in
stopwords.words('english')]
    stemmed_content = ' '.join(stemmed_content)
    return stemmed_content

# Example input text
input_text = "Hillary Clinton in HUGE Trouble After America Noticed SICK Thing Hidden in this
Picture... * LIBERTY WRITERS NEWS"
```

```
# Preprocess the input text
preprocessed_input = preprocess_new_text(input_text)
# Convert the preprocessed text to TF-IDF vector
input_vec = vectorizer.transform([preprocessed_input])

# Make prediction using the trained model
prediction = model.predict(input_vec)

# Output the prediction
if prediction[0][0] >= 0.5:
    print("The news headline is likely to be real.")
else:
    print("The news headline is likely to be fake.")
```



Appendix B:

1. Comparison:

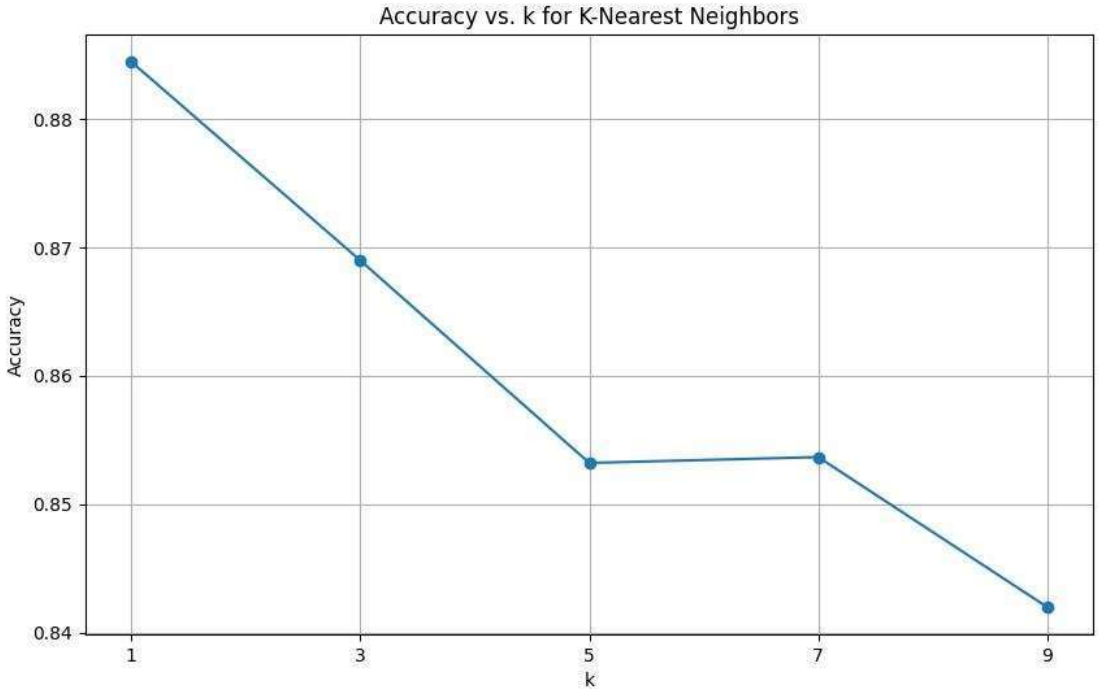
Now we are going to compare different machine learning algorithms that can be used for training the model, i.e. in this case fake news detection model.
We are going to compare this model's Accuracy, Recall, Precision, F1 Score and MCC using algorithms such as Naive Bayes SVM, Decision Tree, Random Forest, K-Nearest Neighbors, DNN.

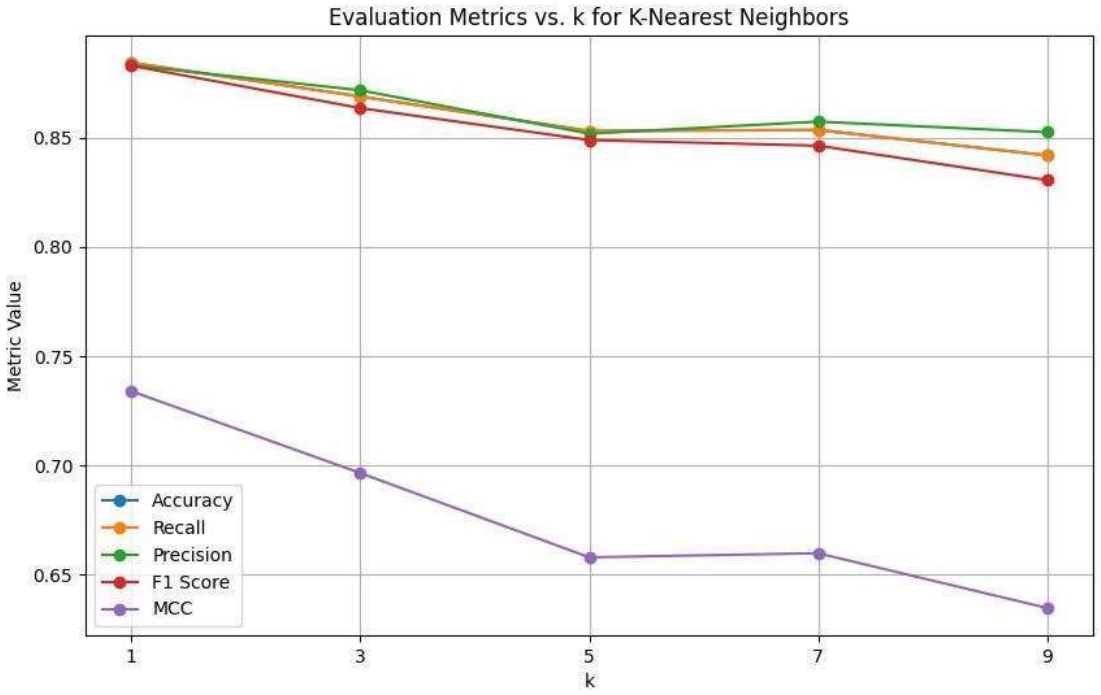
Model	Accuracy	Recall	Precision	F1 Score	MCC
Naive Bayes	0.901440	0.901440	0.901868	0.899327	0.772891
SVM	0.926308	0.926308	0.927305	0.924938	0.831341
Decision Tree	0.898256	0.898256	0.898694	0.898450	0.770352
Random Forest	0.932980	0.932980	0.932771	0.932364	0.846558
K-Nearest Neighbors	0.853829	0.853829	0.852612	0.849659	0.659288
Deep Neural Network	0.934324	0.955847	0.920689	0.937939	0.868942

2. Random Forest vs Sequential DNN:

	Random Forest	**Sequential DNN**
Structure	Ensemble of decision trees	Sequential layers of neurons
Interpretability	Moderate	Low
Training Speed	Fast	Slow
Scalability	Limited	High
Performance	Good on tabular data	Good on unstructured data
Overfitting	Tends to generalise well	Prone to overfitting without regularisation
Hyperparameters	Few, easy to tune	Many, requires careful tuning
Handling Features	Automatically handles feature importance	Requires feature engineering
Deployment	Easy to deploy, low computational cost	May require higher computational resources
Memory Usage	Low	High

3. K-Nearest Neighbors for different K Values: (k values = [1, 3, 5, 7, 9])





REFERENCES

[1] Dataset-1:https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset/data?select=True.csv [10-05-2024]

[2] Dataset-2: https://github.com/aryashah2k/FakeNewsDetectionSpacy[10-05-2024]

[3] Dataset-3: https://github.com/pratham3012/Fake-News-Detector/blob/main/news.zip[10-05-2024]

[4] Dataset-4: https://github.com/SaiSanjana2002/Fake_news_detection\[10-05-2024]

[5] Dataset-5: https://github.com/AadiYNWA/Fake-News-Detection[10-05-2024]

[6] Dataset-6: https://www.kaggle.com/datasets/saurabhshahane/fake-news-classification[10-05-2024]

[7] Research Paper-1: https://www.sciencedirect.com/science/article/pii/S2405844024007588[10-05-2024]

[8] Research Paper-2: https://www.sciencedirect.com/science/article/pii/S2352340924004098[10-05-2024]

[9] Research Paper-3: https://ieeexplore.ieee.org/abstract/document/8987258/[10-05-2024]

[10] Research Paper-4: https://www.sciencedirect.com/science/article/pii/S0169023X22001094[10-05-2024]

[11] Research Paper-5: https://ieeexplore.ieee.org/abstract/document/9532796[10-05-2024]