Deep Detectives: Unmasking Fake News Through Deep Learning

A project report submitted to

MALLA REDDY UNIVERSITY

in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLGY

in

COMPUTER SCIENCE & ENGINEERING (AI & ML)

Submitted by

A. Manvitha	**:**	**2111CS020273**
B. Manya Vardhan	**:**	**2111CS020274**
Y. Yeshwanth Kumar	**:**	**2111CS020275**
P. Nikitha	**:**	**2111CS020319**
T. Nikhil	**:**	**2111CS020322**

Under the Guidance of

Prof T Ramya

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

2024

COLLEGE CERTIFICATE

This is to certify that this is the bonafide record of the Application Development entitled,**"Deep Detectives-Unmasking Fake News Through Deep Learning"** Submitted by A Manvitha(2111CS020273), BManya Vardhan(2111CS020274), Y Yeshwanth Kumar(2111CS020275), P Nikitha(2111CS020319), T Nikhil(2111CS020322)B. Tech IIIA year II semester, Department of CSE (AI&ML) during the year 2023-24. The results embodied in the report have not been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE
Prof T Ramya

HEAD OF THE DEPARTMENT
Dr.Thayyaba Khatoon
CSE(AI&ML)

EXTERNAL EXAMINER

ACKNOWLEDEMENT

We would like to express our gratitude to all those who extended their support and suggestions tocome up with this application. Special Thanks to our guide Prof T Ramya whose help and stimulating suggestions and encouragement helped us alltime in the due course of projectdevelopment.

We sincerely thank our HOD Dr.Thayyaba Khatoon for her constant support and motivation allthe time. A special acknowledgement goes to a friend who enthused us from the back stage. Last but not the least our sincere appreciation goes to our family who has been tolerant understanding our moods, and extending timely support.

ABSTRACT

In today's digital age, fake news is rampant and can spread rapidly, leading to misinformation and confusion. To address this issue, we propose a straightforward method utilizing deep learning techniques for detecting fake news. Our approach involves training advanced computer models to automatically analyze news articles and social media posts to determine whether they are reliable or misleading. We use deep learning algorithms, specifically designed to understand and process text, images, and videos. These algorithms learn from vast amounts of data to recognize patterns and signals that indicate whether a piece of content is genuine or fake. By incorporating techniques like sentiment analysis, which evaluates the emotions expressed in the text, and attention mechanisms, which focus on important parts of the content, our system becomes adept at discerning misinformation.

CONTENTS

CHAPTER NO	TITLE	PAGE NO
1	1 Introduction 1.1Project Definition 1.2Objective Of project 1.3Scope Of Project	1-3
2	2 Analysis 2.1 Project Planning and Research 2.2 Software requirement specification 2.2.1 Software Requirement 2.2.2 Hardware Requirement 2.3 Model selection and Architecture	4-7
3	3 Design 3.1 Introduction 3.2 DFD/ER/UML Diagram 3.3 Data set Description 3.4 Data preprocessing techniques 3.5 Methods and algorithms	8-10
4	4 Deployment and results 4.1 Introduction 4.2 Source code 4.3 Model implementation and training 4.4 Model evaluation metrics 4.5 Model deployment: testing and validation 4.6 Results	11-16
5	5 Conclusion 5.1 Project Conclusion 5.2 Future Scope	17

CHAPTER 1

1 INTRODUCTION

- In today's digital landscape, the spread of fake news poses a significant threat to the integrity of information and the stability of democratic processes.
- With the exponential growth of social media platforms and online news outlets, misinformation can proliferate rapidly, influencing public opinion, shaping societal beliefs, and even impacting political discourse.
- As a result, there is an urgent need for robust and efficient methods to detect and combat fake news effectively.
- Traditional approaches to fake news detection, such as manual fact-checking and rule-based algorithms, are increasingly inadequate in addressing the scale and complexity of the problem.
- These methods are time-consuming, labor-intensive, and often struggle to keep pace with the evolving tactics used by purveyors of misinformation.

1.1 Project Definition

Fake news is everywhere these days, especially on social media and the internet. It spreads quickly and can really mess with people's opinions and beliefs, even affecting important stuff like politics. But stopping fake news isn't easy. Right now, most methods involve people fact-checking everything or using rules to spot fake stories. But that takes a lot of time and effort, and it's hard to keep up with all the fake stuff out there. Plus, as fake news gets more clever, these methods might not work as well. We need better ways to find and stop fake news, ones that can handle the huge amount of information online and keep up with new tricks. It's super important for keeping our information honest and our democracy strong**.**

1.2 Objective Of Project

The primary objective of employing deep learning techniques for fake news detection is to develop robust models capable of accurately discerning between genuine and deceptive news articles. This objective can be further broken down into specific goals:

High Accuracy: Build a model that achieves high accuracy in distinguishing between real and fake news articles. The model should minimize false positives (misclassifying real news as fake) and false negatives (misclassifying fake news as real).

Generalization: Ensure that the model generalizes well to unseen data from diverse sources and topics. It should not overfit to the training data but rather capture underlying patterns indicative of fake news across different contexts.

Real-time Detection: Develop models suitable for real-time or near-real-time detection of fake news. This involves optimizing the model for efficiency and scalability to handle large volumes of news articles with low latency.

Interpretability: Enhance the interpretability of the model's predictions to provide insights into the features and patterns contributing to classification decisions. This helps users understand why a particular news article was classified as fake or real.

Robustness: Create models that are robust to adversarial attacks and attempts to deceive the system. This involves incorporating techniques such as adversarial training and robust optimization to improve resilience against manipulation.

Ethical Considerations: Address ethical concerns related to the potential impact of fake news detection, including censorship, privacy, and bias. Ensure that the deployment of fake news detection systems aligns with ethical principles and respects user rights.

1.3 Scope Of The Project

Text-based Detection: The project focuses on detecting fake news primarily through analyzing the textual content of news articles. Other modalities such as images, videos, or social media posts are outside the scope.

Deep Learning Approaches: The project employs deep learning techniques such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), or transformer-based models for fake news detection.

Binary Classification: The project aims to classify news articles into two categories: real and fake. It does not address nuanced classifications such as satire or opinion pieces.

Model Deployment: The scope includes deploying the trained model into a production environment for real-time or batch processing of news articles.

Evaluation Metrics: The project evaluates the performance of the model using standard metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

CHAPTER 2

ANALYSIS

2.1 Project Planning And Research

- Data Loading and Exploration: You loaded fake news datasets, checked for missing values, and explored the data using visualizations.
- Data Preprocessing: You combined the title and text columns into a single column, applied stemming to preprocess the text data, and converted the text into numerical values using TF-IDF vectorization.
- Model Building: You split the data into training and testing sets, then built a logistic regression model using scikit-learn.
- Model Evaluation: You made predictions using the model and evaluated its accuracy.\
- Data Exploration: You could explore the distribution of fake and true news across different subjects or dates to identify any patterns or trends. Investigating the length of the news articles (number of words) and comparing it between fake and true news might provide insights into their writing style.
- Text Preprocessing: Besides stemming, you might want to explore other text preprocessing techniques such as lemmatization, which reduces words to their base or dictionary form. Handling of special characters, URLs, and other non-text elements could further improve the quality of your text data.
- Model Selection :While logistic regression is a good baseline model for binary classification tasks like this, you could also experiment with other algorithms such as Random Forest, Support Vector Machines, or even neural networks to see if they yield better performance. Hyperparameter tuning techniques like grid search or random search could be employed to optimize the performance of your chosen model.
- Model Evaluation: Besides accuracy, consider other metrics like precision, recall, and F1-score, especially in scenarios where class imbalance exists. Visualizing the confusion matrix or ROC curve can provide a more detailed understanding of your model's performance across different thresholds.

2.2 Software Requirement and Specification

2.2.1 Software Requirement

Python 3: The code is written in Python 3, so we need a Python interpreter installed on our system to run it.

Libraries:

numpy: For numerical computing.

pandas: For data manipulation and analysis.

seaborn, matplotlib.

pyplot: For data visualization.

nltk: Natural Language Toolkit for text processing.

scikit-learn: For machine learning tasks such as model building, preprocessing, and evaluation.

Datasets:

The code assumes that we have two CSV files named "Fake.csv" and "True.csv" containing fake and true news articles respectively. Ensure that these files are present in the specified directory (C:/Users/Hp/Desktop/AD-DL/News _dataset/) or modify the file paths accordingly.

Jupyter Notebook or Kaggle Environment:

The code appears to be written in a Jupyter Notebook or a similar environment like Kaggle, as indicated by the code comments and cell formatting

2.2.2 Hardware Requirements

Computer or Server: You need a computer or server with sufficient processing power to execute Python code. This could be a personal computer, a cloud-based virtual machine, or a server.

CPU: The code primarily utilizes the CPU for tasks such as data loading, preprocessing, model training, and inference. A standard multi-core CPU should suffice for running the code efficiently.

Memory (RAM): While working with large datasets or performing intensive computations, having an adequate amount of RAM is essential to prevent memory-related issues. However, for the provided code, 4GB to 8GB of RAM should be sufficient for most cases.

Storage: You need enough storage space to store the code files, dataset files, and any additional files generated during the execution of the code. The storage requirements depend on the size of your datasets and any intermediate files generated during preprocessing or model training.

Internet Connection :An internet connection is required if you haven't already downloaded NLTK's stopwords corpus. The code attempts to download it using nlp.download('stopwords').

2.3 Model Selection and Architecture

Model Selection

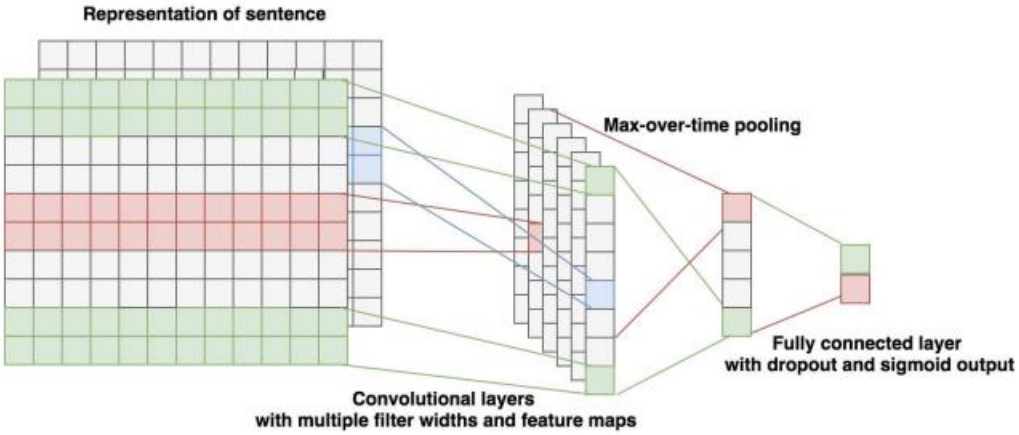
Random Forest: Random Forest is an ensemble learning method that fits multiple decision trees on various sub-samples of the dataset and combines their predictions. It often performs well with minimal hyperparameter tuning and is less prone to overfitting compared to individual decision trees.

Support Vector Machines (SVM): SVMs are powerful supervised learning models used for classification and regression tasks. They work well in high-dimensional spaces and are effective when there is a clear margin of separation between classes.

Gradient Boosting Machines (GBM): GBM is a machine learning technique that builds models in a stage-wise manner, where new models correct errors made by previous models. Popular implementations include XGBoost, LightGBM, and CatBoost, which are known for their high performance and scalability.

Neural Networks: Deep learning models, such as multi-layer perceptrons (MLPs) or convolutional neural networks (CNNs), can capture complex patterns in text data. However, they may require more computational resources and larger datasets for training.

Architecture



2.1 Architecture

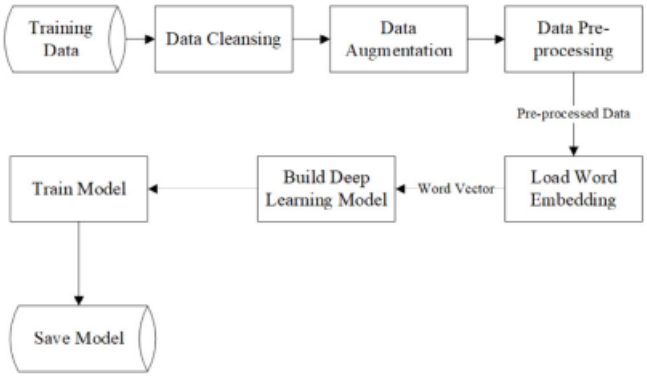
CHAPTER-3

3 DESIGN

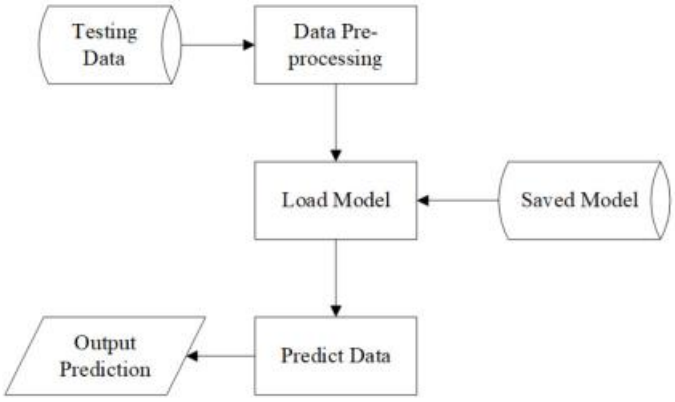
3.1 Introduction

- In today's digital landscape, the spread of fake news poses a significant threat to the integrity of information and the stability of democratic processes.
- With the exponential growth of social media platforms and online news outlets, misinformation can proliferate rapidly, influencing public opinion, shaping societal beliefs, and even impacting political discourse.
- As a result, there is an urgent need for robust and efficient methods to detect and combat fake news effectively.
- Traditional approaches to fake news detection, such as manual fact-checking and rule-based algorithms, are increasingly inadequate in addressing the scale and complexity of the problem.
- These methods are time-consuming, labor-intensive, and often struggle to keep pace with the evolving tactics used by purveyors of misinformation.

3.2 UML Diagrams



3.1 Training Phase



3.2 Testing Phase

3.3 Data Set Description

Data set used in this project is NEWS_DATASET

The datasets consisct of 2 csv files they are Fake and True

Fake News Dataset ("Fake.csv"):

This dataset contains fake news articles.

It likely includes features such as:

title: The title/headline of the news article.

text: The body/content of the news article.

subject: The subject/category of the news article (e.g., politics, entertainment).

date: The publication date of the news article.

Each row represents a single fake news article, with corresponding values for each feature.

True News Dataset ("True.csv"):

This dataset contains true news articles.

Similar to the fake news dataset, it likely includes features such as:

title: The title/headline of the news article.

text: The body/content of the news article.

subject: The subject/category of the news article.

date: The publication date of the news article.

3.4 Data Preprocessing Techniques

Combing text Coloumns: The title and text columns from the dataset are concatenated into a single column called "full_text". This step is performed to combine the title and content of the news articles into one text field for analysis.

Stemming: The text data undergoes stemming, a process of reducing words to their root or base form. The code uses the Porter Stemmer algorithm from NLTK (Natural Language Toolkit) to perform stemming. Stemming helps in reducing the dimensionality of the feature space by transforming words with similar meanings into the same root word.

Stopword Removal: Stopwords, which are common words that do not contribute much to the meaning of the text (e.g., "the", "is", "and"), are removed from the text data. This step helps in reducing noise in the data and improving the performance of the model. NLTK's stopwords corpus is used for this purpose.

Lowercasing: Convert all text to lowercase to ensure consistency and reduce the complexity of the vocabulary. This helps in treating words with different cases (e.g., "Word" and "word") as the same.

3.5 Methods And Algorithms

Data Loading and Exploration: Libraries such as pandas are used for loading CSV files and performing initial exploratory data analysis (EDA). Methods like pd.read_csv() and functions like .shape, .isnull(), and .value_counts() are employed for data inspection and summary statistics.

Data Preprocessing: Text preprocessing techniques include stemming, stopword removal, and concatenation of text columns. The Porter Stemmer algorithm from NLTK is utilized for stemming, and NLTK's stopwords corpus is employed for stopword removal.

Text Vectorization: The TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique is applied to convert the text data into numerical vectors. This is accomplished using scikit-learn's TfidfVectorizer().

Model Building: Logistic Regression is chosen as the classification algorithm for building the fake news detection model. The logistic regression model is instantiated using scikit-learn's LogisticRegression() class.

CHAPTER-4

4 DEPLOYMENT AND RESULTS

4. 1 Introduction

- In today's digital landscape, the spread of fake news poses a significant threat to the integrity of information and the stability of democratic processes.
- With the exponential growth of social media platforms and online news outlets, misinformation can proliferate rapidly, influencing public opinion, shaping societal beliefs, and even impacting political discourse.
- As a result, there is an urgent need for robust and efficient methods to detect and combat fake news effectively.
- Traditional approaches to fake news detection, such as manual fact-checking and rule-based algorithms, are increasingly inadequate in addressing the scale and complexity of the problem.
- These methods are time-consuming, labor-intensive, and often struggle to keep pace with the evolving tactics used by purveyors of misinformation.

4.2 Source Code

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import re
import nltk as nlp
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

Fig: 4.1 Importing Necessary Libraries

```
[7]: fake_df = pd.read_csv("C:/Users/Hp/Desktop/AD-DL/News _dataset/Fake.csv")
     true_df = pd.read_csv("C:/Users/Hp/Desktop/AD-DL/News _dataset/True.csv")
```

Fig : 4.2 Loading Datasets

```
]: # Rows and columns of fake news dataset
print(f"Fake news dataset has: {fake_df.shape[0]} rows")
print(f"Fake news dataset has: {fake_df.shape[1]} columns")
```

```
Fake news dataset has: 23481 rows
Fake news dataset has: 4 columns
```

Fig:4.3 Checking the size of dataset

```
fake_df.isnull().sum()
```

```
title      0
text       0
subject    0
date       0
dtype: int64
```

```
true_df.isnull().sum()
```

```
title      0
text       0
subject    0
date       0
dtype: int64
```

Fig: 4.4 Preprocessing techniques

```
# Adding 'Fake' column to our datasets then join them together
fake_df['Fake'] = 1
true_df['Fake'] = 0
df = pd.concat([fake_df, true_df])
```

```
plt.style.use('ggplot')
plt.figure(figsize=(10,7))
sns.countplot(data=df, x='Fake')
```

```
<Axes: xlabel='Fake', ylabel='count'>
```

Fig: 4.5 Adding fake coloumns

```
# Correlation betwwen year and news
plt.style.use('seaborn-pastel')
plt.figure(figsize=(10, 7))
sns.countplot(data=df, x='title', hue='Fake'
```

```
C:\Users\Hp\AppData\Local\Temp\ipykernel_170
to the styles shipped by seaborn. However, t
  plt.style.use('seaborn-pastel')
<Axes: xlabel='title', ylabel='count'>
```

Fig 4.6: Correlaton between year and news

```
# Correlation between months and news
plt.style.use('bmh')
plt.figure(figsize=(12, 7))
sns.countplot(data=df, x='title', hue='Fake')
```

```
<Axes: xlabel='title', ylabel='count'>
```

Fig 4.7:Correlation between months and news

```
# Subjects count
df.title.value_counts()
```

```
title
coerce    44898
Name: count, dtype: int64
```

```
plt.style.use('seaborn-paper')
plt.figure(figsize=(12, 7))
```

```
C:\Users\Hp\AppData\Local\Temp\ipykernel_17020\2860176510.py:1: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplo
d to the styles shipped by seaborn. However, they will remain available as 'seaborn-v0_8-<style>'. Alternatively, directly use the
  plt.style.use('seaborn-paper')
<Figure size 1200x700 with 0 Axes>

<Figure size 1200x700 with 0 Axes>
```

```
nlp.download('stopwords')
```

```
[nltk_data] Error loading stopwords: <urlopen error [Errno 11001]
[nltk_data]     getaddrinfo failed>
False
```

```
# Joining the title and the content columns
df['full_text'] = df['title'] + ' ' + df['text']
```

```
df.head()
```

	title	text	subject	date	Fake	full_text
0	coerce	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017	1	coerce Donald Trump just couldn t wish all Ame...
1	coerce	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017	1	coerce House Intelligence Committee Chairman D...
2	coerce	On Friday, it was revealed that former Milwauk...	News	December 30, 2017	1	coerce On Friday, it was revealed that former ...
3	coerce	On Christmas day, Donald Trump announced that ...	News	December 29, 2017	1	coerce On Christmas day, Donald Trump announce...
4	coerce	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017	1	coerce Pope Francis used his annual Christmas ...

Fig 4.8: Intializing NLP

```
# Target variable and features
y = df['Fake']
X = df.drop('Fake', axis=1)
```

```
# Stemming process
def stemming_process(y):
    first_step = re.sub(r"^[A-Za-z]", ' ', y).lower()
    second_step = first_step.split()
    porter_stemmer = PorterStemmer()
    result = []
    for w in second_step:
        if w not in stopwords.words('english'):
            result.append(porter_stemmer.stem(w))
    return ' '.join(result)
```

```
# Converting X and y to numpy arrays
X = df['full_text'].to_numpy()
y = df['Fake'].to_numpy()
```

```
# Converting the text into numerical values
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
vectorizer.fit(X)
X = vectorizer.transform(X)
```

```
# Building the model
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Fig 4.9:Steaming process

```
# Fitting the model - Logistic Regression
lg = LogisticRegression()
lg.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

Fig 4.10: Logistic regression

```
# Predictiion
prediction = lg.predict(X_test)
```

```
# Score
accuracy = accuracy_score(prediction, y_test)
print(f"Model precision: {accuracy}")
```

```
Model precision: 0.9865478841870824
```

Fig 4.11: Model Acuracy

4.3 Model Implementation And Training

Data Loading and Preprocessing:The code starts by loading the fake and true news datasets from CSV files using pd.read_csv().Text preprocessing techniques such as stemming, stopword removal, and concatenation of text columns are applied to prepare the text data for modeling.

Feature Engineering :The text data is converted into numerical vectors using TF-IDF vectorization with TfidfVectorizer() from scikit-learn. This step transforms the text features into a format suitable for machine learning algorithms.

Model Building :The dataset is split into training and testing sets using train_test_split() from scikit-learn.

A logistic regression model is instantiated using LogisticRegression() from scikit-learn.

Model Training :The logistic regression model is trained on the training data using the fit() method. This step involves learning the parameters of the logistic regression model to make predictions on new data.

Model Evaluation :Predictions are made on the testing data using the trained logistic regression model with the predict() method. Model accuracy is evaluated using the accuracy_score() function from scikit-learn's metrics module.

4.4 Model Evaluation Metrics

Accuracy:Accuracy is calculated using the accuracy_score() function from scikit-learn's metrics module.

It is computed as the ratio of the number of correctly predicted instances to the total number of instances in the test set.Accuracy is suitable for balanced datasets where the classes are evenly distributed.

However, it may not be the best metric for imbalanced datasets, where one class dominates the other.

4.5 Model Deployment: Testing And Validating

Training-Testing Split: The dataset is split into training and testing sets using the train_test_split() function from scikit-learn. This split allows for evaluating the model's performance on unseen data.

Model Training: The logistic regression model (lg) is trained on the training set (X_train, y_train) using the fit() method. During training, the model learns patterns in the data to make predictions.

Model Prediction: After training, the model makes predictions on the testing set (X_test) using the predict() method. These predictions are compared against the actual labels (y_test) to assess the model's performance.

Model Evaluation: The accuracy score is computed using the accuracy_score() function from scikit-learn's metrics module. Accuracy measures the proportion of correctly classified instances out of the total number of instances in the testing set. This metric provides an indication of how well the model generalizes to unseen data.

4.6 Results

```
In [30]: # Predictiion
         prediction = lg.predict(X_test)

In [31]: # Score
         accuracy = accuracy_score(prediction, y_test)
         print(f"Model precision: {accuracy}")

         Model precision: 0.9865478841870824
```

Fig: 4.12 Final Results

CHAPTER 5

5 CONCLUSION

5.1 Project Conclusion

The existing methodology for fake news detection through deep learning, with a focus on Natural Language Processing (NLP), offers a structured approach to address the pervasive issue of misinformation in the digital era. By harnessing advanced NLP techniques within the Deep Detective framework, researchers and practitioners can develop sophisticated solutions to detect and combat fake news effectively.Through meticulous data collection and preprocessing, the methodology ensures the availability of high-quality datasets and standardized text representations essential for training deep learning models. By leveraging word embeddings or contextual embeddings, researchers can capture semantic nuances and contextual information critical for distinguishing between fake and real news articles.

5.2 Future Scope

Feature Engineering: Explore additional text preprocessing techniques such as lemmatization, n-grams, and feature selection to improve the model's predictive performance.

Model Selection: Experiment with different machine learning algorithms (e.g., Random Forest, Support Vector Machines, Neural Networks) to compare their performance and identify the most suitable model for the task.

Hyperparameter Tuning: Conduct hyperparameter tuning using techniques like grid search or random search to optimize the model's parameters and improve its performance.

Ensemble Methods: Explore ensemble methods such as bagging, boosting, or stacking to combine multiple models and potentially achieve better results.

Cross-Validation: Implement cross-validation techniques to assess the model's performance more robustly and mitigate overfitting.