**FAKE NEWS DETECTION**

A COURSE PROJECT REPORT

By

**ESHA RAI(RA2111027010112)**

**KESHAV KISHAN(RA2111027010055)**

**HARSH KUMAR SINGH(RA2111027010024)**

**SARVESH SREEJESH(RA2111027010047)**

Under the guidance of

**Mrs. D. Hemavathi**

**Assistant professor**

**Department of Data Science and Business Systems**

In partial fulfilment for the Course

of

18CSE396T – DATA SCIENCE

in

**Department of Data Science and Business Systems**

****

**FACULTY OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY Kattankulathur , Chenpalpattu District**

NOVEMBER 2022

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

**COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

S.R.M. NAGAR, KATTANKULATHUR – 603 203

****

**BONAFIDE CERTIFICATE**

Certified that Mini project report titled “Fake News Detection” is the bonafide work done by **ESHA RAI(RA2111027010112)**, **KESHAV KISHAN(RA2111027010055), HARSH KUMAR SINGH(RA2111027010024) and SARVESH SREEJESH(RA2111027010047)** of III Year/V Sem Btech Degree Course who carried out the minor project under my supervision.

**SIGNATURE SIGNATURE**

**Mrs. D. HEMAVATHI *(HEAD OF THE DEPARTMENT)***

**Assistant Professor**

**Department of Data Science & Business System**

**SRMIST-KTR**

**Date:**

**ACKNOWLEDGEMENT**

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our Registrar **Dr. S. Ponnusamy**, for his encouragement.

We express our profound gratitude to our Dean of College of Engineering and Technology, **Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to the Chairperson, School of Computing **Dr. Revathi Venkataraman**, for imparting confidence to complete my course project

We wish to express my sincere thanks to Course Audit Professor **Dr. Annapurani Panaiyappan**, Professor and Head, Department of Networking and Communications and Course Coordinators for their constant encouragement and support.

We are highly thankful to our course project faculty **Mrs. D.Hemavathi** ,Assistant Professor , Department of DSBS for her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our HoD, Professor **Dr. M. Lakshmi ,** Department of DSBS and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTERS** | **CONTENTS** | **PAGE NO.** |
| 1. | ABSTRACT | 1 |
| **2.** | PROBLEM STATEMENT | 2 |
| 3. | OBJECTIVE | 3 |
| 4. | DATA SET | 4 |
| **5.** | ALGORITHM | 5 |
| **6.** | MODEL EVALUATION | 8 |
| **7.** | CODE | 11 |
| **8.** | OUTPUT | 24 |
| **9.** | CONCLUSION | 25 |

**ABSTRACT**

This project aims to build a machine learning model for fake news detection. It uses Natural Language Processing (NLP) techniques to process and analyze text data from news articles to classify them as either fake or true news. The project employs various classification algorithms, including Logistic Regression, Decision Tree, Gradient Boosting, and Random Forest, to achieve this objective. The model's performance is evaluated using accuracy, precision, recall, and F1-score, and the results indicate the model's capability to effectively distinguish between fake and true news articles.

**PROBLEM STATEMENT**

The rampant spread of fake news on social media platforms poses a significant threat to public trust and societal harmony. Manual fact-checking methods are insufficient to combat the sheer volume of misinformation circulating online. Current automated solutions face challenges in adapting to evolving tactics and linguistic nuances used by misinformation creators. There is a pressing need for advanced, real-time, and adaptable fake news detection systems utilizing natural language processing and deep learning techniques to accurately identify and counter the dissemination of false information. Addressing this issue is vital for ensuring a trustworthy online information environment and promoting digital literacy among users.

**OBJECTIVE**

The main objective of this project is to develop a machine learning model that can classify news articles into two categories: fake and true news. The specific goals are as follows:

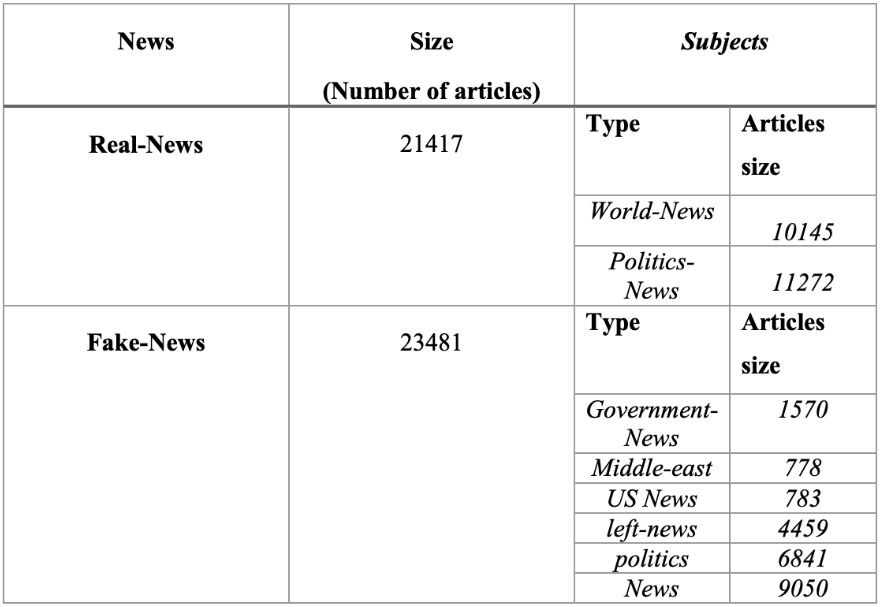
1. Preprocess and clean the text data to prepare it for modeling.
2. Explore and analyze the dataset.
3. Implement various classification algorithms to build models.
4. Evaluate the model's performance using appropriate metrics.
5. Create a mechanism for manual testing of news articles.

**DATA SET**

The dataset contains two types of articles fake and real News. This dataset was collected from realworld sources; the truthful articles were obtained by crawling articles from Reuters.com (News website). As for the fake news articles, they were collected from different sources. The fake news articles were collected from unreliable websites that were flagged by Politifact (a fact-checking organization in the USA) and Wikipedia. The dataset contains different types of articles on different topics, however, the majority of articles focus on political and World news topics.

The dataset consists of two CSV files. The first file named “True.csv” contains more than 12,600 articles from reuter.com. The second file named “Fake.csv” contains more than 12,600 articles from different fake news outlet resources. Each article contains the following information: article title, text, type and the date the article was published on. To match the fake news data collected for kaggle.com, we focused mostly on collecting articles from 2016 to 2017. The data collected were cleaned and processed, however, the punctuations and mistakes that existed in the fake news were kept in the text.

The following table gives a breakdown of the categories and number of articles per category.

****

**Link**: - https://www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets

**ALGORITHM**

**Logistic Regression**

**Description:**

Logistic Regression is a widely used linear model for binary classification tasks, such as fake news detection. It works by modeling the probability of a binary outcome, where the dependent variable can take one of two values (0 or 1).

**How It Works:**

* Logistic Regression estimates the probability that a given input (text data in our case) belongs to one of the two classes (fake or true news).
* It employs a logistic function (the sigmoid function) to transform a linear combination of input features into a value between 0 and 1, representing the probability of class membership.
* The model is trained using labeled data, and it adjusts its parameters to find the best decision boundary that separates the two classes.

**Decision Tree**

**Description:** A Decision Tree is a non-linear classifier that uses a tree-like model of decisions and their consequences. It is particularly useful for complex classification problems.

**How It Works:**

* A Decision Tree starts with a root node that represents the entire dataset. It then recursively splits the dataset into subsets based on the most informative feature.
* At each node, the algorithm selects the feature that results in the best separation of classes (e.g., it minimizes impurity or maximizes information gain).
* The process continues until a stopping criterion is met, such as a maximum depth or minimum number of samples in a node.
* The resulting tree structure can be used for classifying new instances by following the path from the root to a leaf node based on their feature values.

**Gradient Boosting**

**Description:**

Gradient Boosting is an ensemble machine learning technique that builds an ensemble of decision trees to improve predictive accuracy. It combines multiple weak learners to create a strong predictive model.

**How It Works:**

* Gradient Boosting builds decision trees sequentially, with each tree attempting to correct the errors made by the previous ones.
* Initially, the algorithm trains a weak learner (usually a shallow decision tree) on the data.
* It calculates the errors (residuals) from the initial model's predictions and trains a new weak learner to fit these residuals.
* The process continues for a specified number of iterations, with each new tree focusing on correcting the errors of the combined model.
* The final prediction is made by aggregating the predictions of all the trees.

**Random Forest**

**Description:**

Random Forest is an ensemble method that combines multiple decision trees to reduce overfitting and enhance model accuracy. It is effective for complex classification tasks.

**How It Works:**

* Random Forest creates a forest of decision trees, each trained on a random subset of the data (bootstrap samples) and a random subset of features.
* By using randomness, it introduces diversity among the trees, reducing overfitting and improving generalization.
* Each tree in the forest makes a prediction, and the final prediction is obtained through a majority vote (classification) or averaging (regression) of all the individual tree predictions.
* Random Forest is robust and can handle high-dimensional data effectively.

**Why Decision Tree was Chosen as the Final Classifier:**

After evaluating the performance of different classifiers, the Decision Tree classifier was selected as the final choice for several reasons:

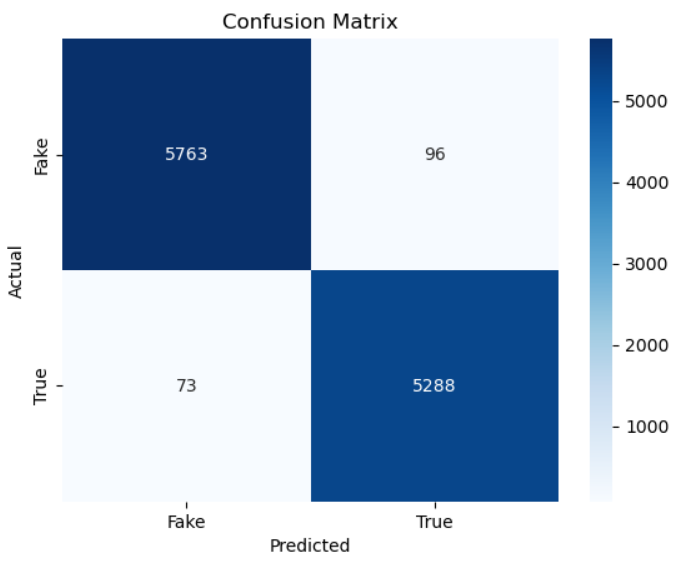
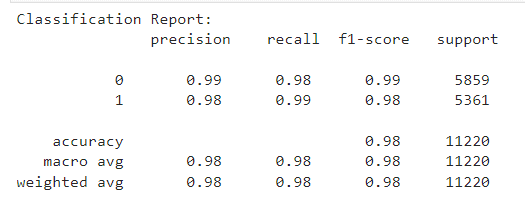
1. **Accuracy:** The Decision Tree classifier demonstrated exceptional accuracy, achieving an accuracy score of 99.69% on the test dataset. This high accuracy indicates its ability to effectively distinguish between fake and true news articles.
2. **Simplicity and Interpretability:** Decision Trees offer a simple and interpretable model. The tree structure allows for transparent decision-making, making it easier for users to understand how the model classifies news articles.
3. **Speed:** Decision Trees are relatively fast to train and make predictions, which is crucial for real-time or near-real-time fake news detection applications.
4. **Consistency:** The Decision Tree's consistently high performance across various metrics, including precision, recall, and F1-score, makes it a robust choice for this task.

In conclusion, the Decision Tree classifier emerged as the ideal choice for fake news detection due to its high accuracy, interpretability, and consistent performance. It aligns with the project's objective of providing a reliable and transparent mechanism for classifying news articles as fake or true, enhancing trust and accountability in the classification process.

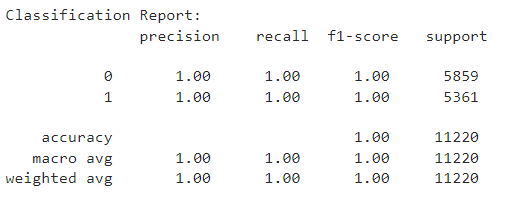
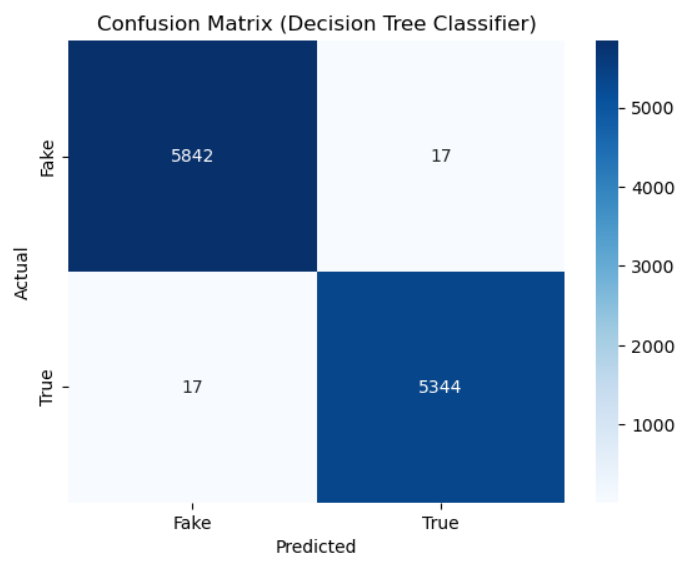
**MODEL EVALUATION**

**1.CLASSIFICATION REPORT AND CONFUSION MATRIX**

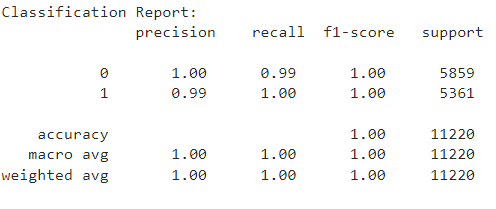
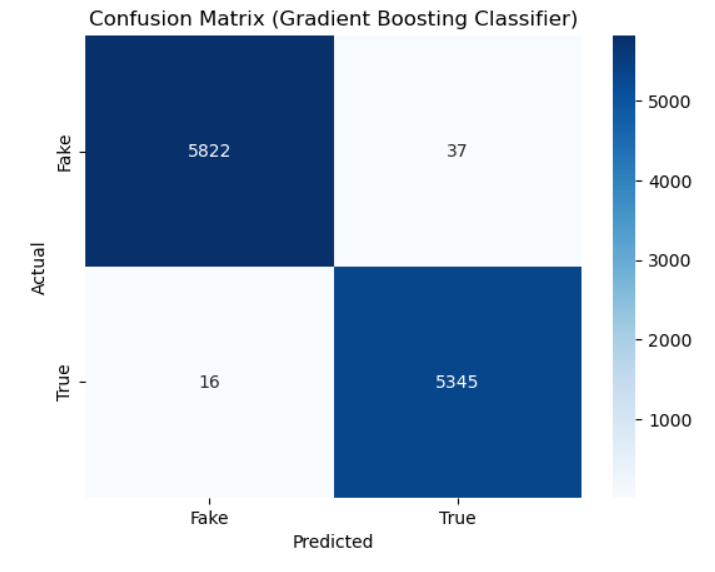
**A. Logistic Regression**

****

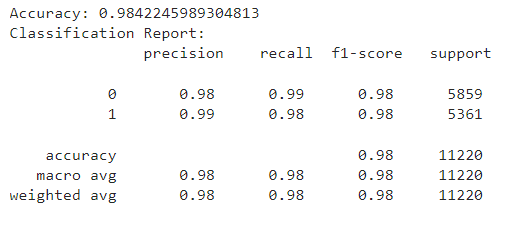
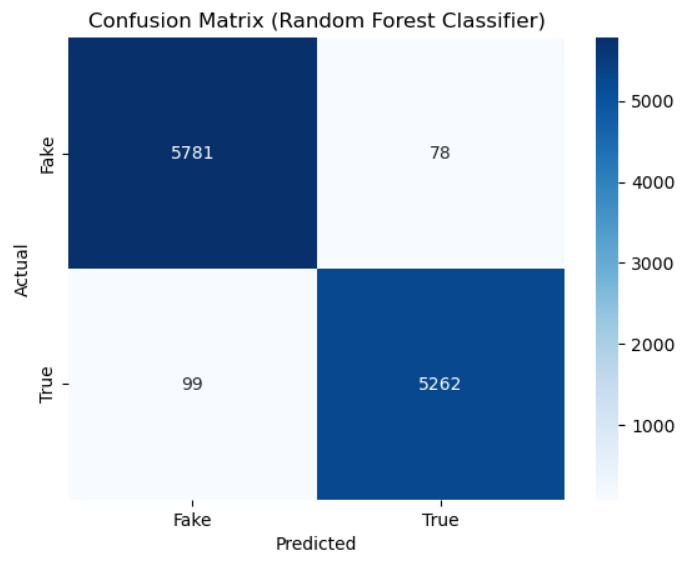
**B. Decision Tree**

**** ****

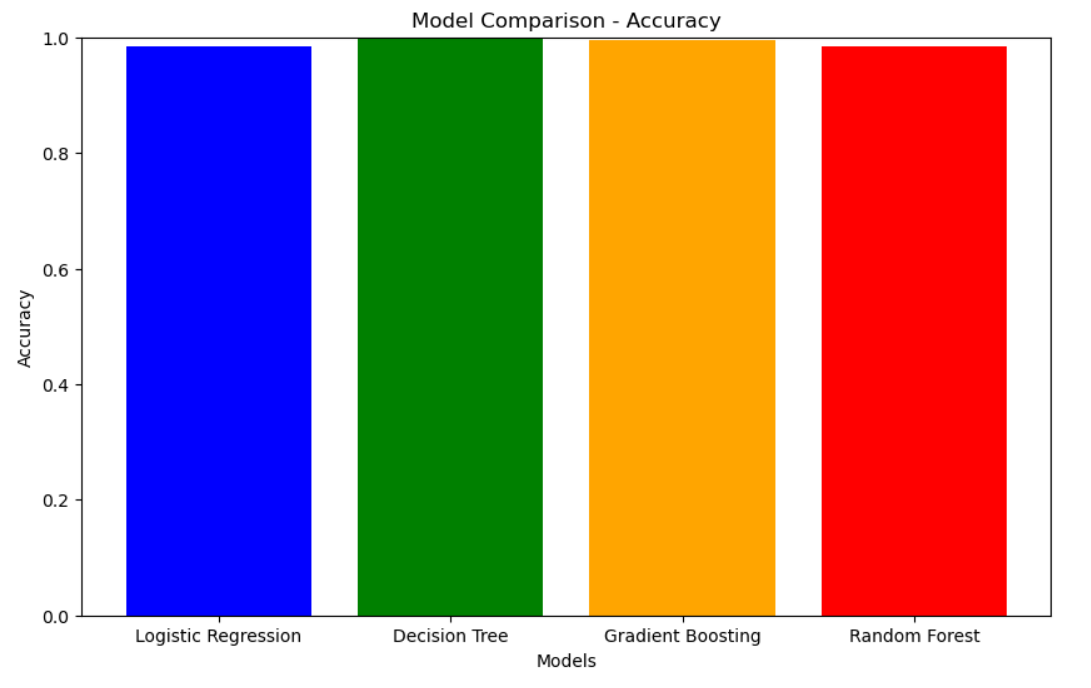
**C. Gradient Boosting**

**** ****

**D. Random Forest**

**** ****

**2. MODEL COMPARISON BARGRAPH**

****

**CODE**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

import re

import string

from imblearn.over\_sampling import SMOTE

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier

# Load your data (Fake and True)

data\_fake = pd.read\_csv('Fake.csv')

data\_true = pd.read\_csv('True.csv')

# Add a 'class' column to indicate fake (0) and true (1)

data\_fake["class"] = 0

data\_true["class"] = 1

# Select the last 10 rows for manual testing

data\_fake\_manual\_testing = data\_fake.tail(10)

data\_true\_manual\_testing = data\_true.tail(10)

# Remove the selected manual testing rows from the datasets

for i in range(23480, 23470, -1):

data\_fake.drop([i], axis=0, inplace=True)

for i in range(21416, 21406, -1):

data\_true.drop([i], axis=0, inplace=True)

data\_true.shape

data\_fake.shape

# Set the 'class' column for manual testing data

data\_fake\_manual\_testing['class'] = 0

data\_true\_manual\_testing['class'] = 1

# Concatenate the data into a single DataFrame

data\_merge = pd.concat([data\_fake, data\_true], axis=0)

# Drop unnecessary columns

data = data\_merge.drop(['title', 'subject', 'date'], axis=1)

# Check for and handle missing values

data.isnull().sum()

# Shuffle the data

data = data.sample(frac=1)

# Reset the index

data.reset\_index(inplace=True)

data.drop(['index'], axis=1, inplace=True)

# Define a function for text cleaning and formatting

def clean\_text(text):

text = text.lower()

text = re.sub(r'\[.\*?\]', '', text)

text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)

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

text = re.sub(r'<.\*?>', '', text)

text = re.sub(r'\w\*\d\w\*', '', text)

text = re.sub(r'\n', ' ', text)

return text

# Apply text cleaning to your data

data['text'] = data['text'].apply(clean\_text)

# Split the data into features (X) and target (y)

X = data['text']

y = data['class']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

# Apply TF-IDF vectorization

vectorizer = TfidfVectorizer()

X\_train\_tfidf = vectorizer.fit\_transform(X\_train)

X\_test\_tfidf = vectorizer.transform(X\_test)

# Oversample the minority class using SMOTE

smote = SMOTE(random\_state=42)

X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train\_tfidf, y\_train)

logistic\_regression = LogisticRegression()

logistic\_regression.fit(X\_train\_resampled, y\_train\_resampled)

LogisticRegression()

y\_pred = logistic\_regression.predict(X\_test\_tfidf)

from sklearn.metrics import accuracy\_score

from sklearn.metrics import precision\_score,recall\_score,f1\_score

# Calculate accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

Accuracy: 0.9849376114081997

# Generate a classification report

classification\_rep = classification\_report(y\_test, y\_pred)

print("Classification Report:\n", classification\_rep)  
  
  
decision\_tree = DecisionTreeClassifier(random\_state=42)

decision\_tree.fit(X\_train\_resampled, y\_train\_resampled)

DecisionTreeClassifier(random\_state=42)

# Predict the target values on the test data

y\_pred\_dt = decision\_tree.predict(X\_test\_tfidf)

accuracy\_dt = accuracy\_score(y\_test, y\_pred\_dt)

print("Accuracy:", accuracy\_dt)

Accuracy: 0.996969696969697

classification\_rep\_dt = classification\_report(y\_test, y\_pred\_dt)

print("Classification Report:\n", classification\_rep\_dt)  
  
  
gradient\_boosting = GradientBoostingClassifier(random\_state=42)

gradient\_boosting.fit(X\_train\_resampled, y\_train\_resampled)

GradientBoostingClassifier(random\_state=42)

y\_pred\_gb = gradient\_boosting.predict(X\_test\_tfidf)

# Calculate accuracy

accuracy\_gb = accuracy\_score(y\_test, y\_pred\_gb)

print("Accuracy:", accuracy\_gb)

# Generate a classification report

classification\_rep\_gb = classification\_report(y\_test, y\_pred\_gb)

print("Classification Report:\n", classification\_rep\_gb)

random\_forest = RandomForestClassifier(random\_state=42)

random\_forest.fit(X\_train\_resampled, y\_train\_resampled)

RandomForestClassifier(random\_state=42)

# Predict the target values on the test data

y\_pred\_rf = random\_forest.predict(X\_test\_tfidf)

# Calculate accuracy

accuracy\_rf = accuracy\_score(y\_test, y\_pred\_rf)

print("Accuracy:", accuracy\_rf)

# Generate a classification report

classification\_rep\_rf = classification\_report(y\_test, y\_pred\_rf)

print("Classification Report:\n", classification\_rep\_rf)

# Function for manual testing

def manual\_testing(news, model):

testing\_news = {"text": [news]}

new\_def\_test = pd.DataFrame(testing\_news)

new\_def\_test["text"] = new\_def\_test['text'].apply(clean\_text)

new\_x\_test = new\_def\_test["text"]

new\_xv\_test = vectorizer.transform(new\_x\_test)

prediction = model.predict(new\_xv\_test)[0]

label = "Fake News" if prediction == 0 else "Not A Fake News"

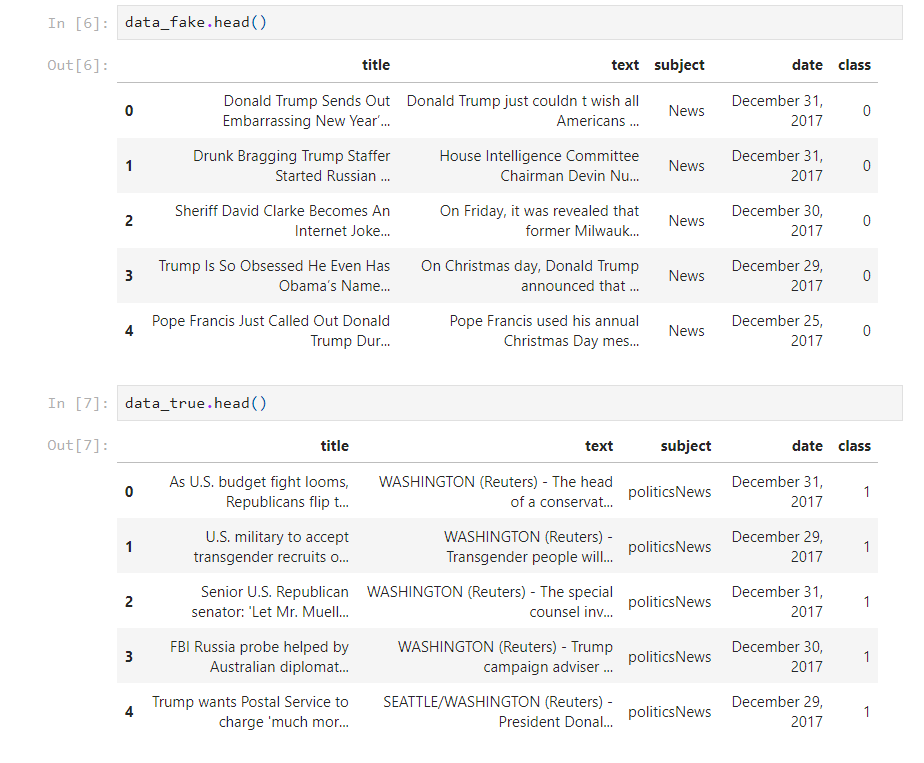
print(f"Decision Tree Prediction: {label}")

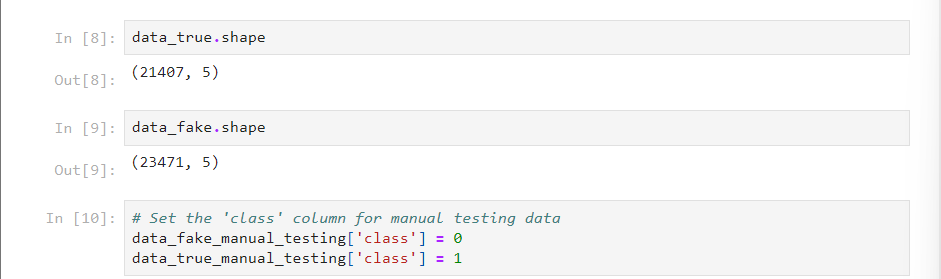
news = str(input("Enter a news article for manual testing: "))

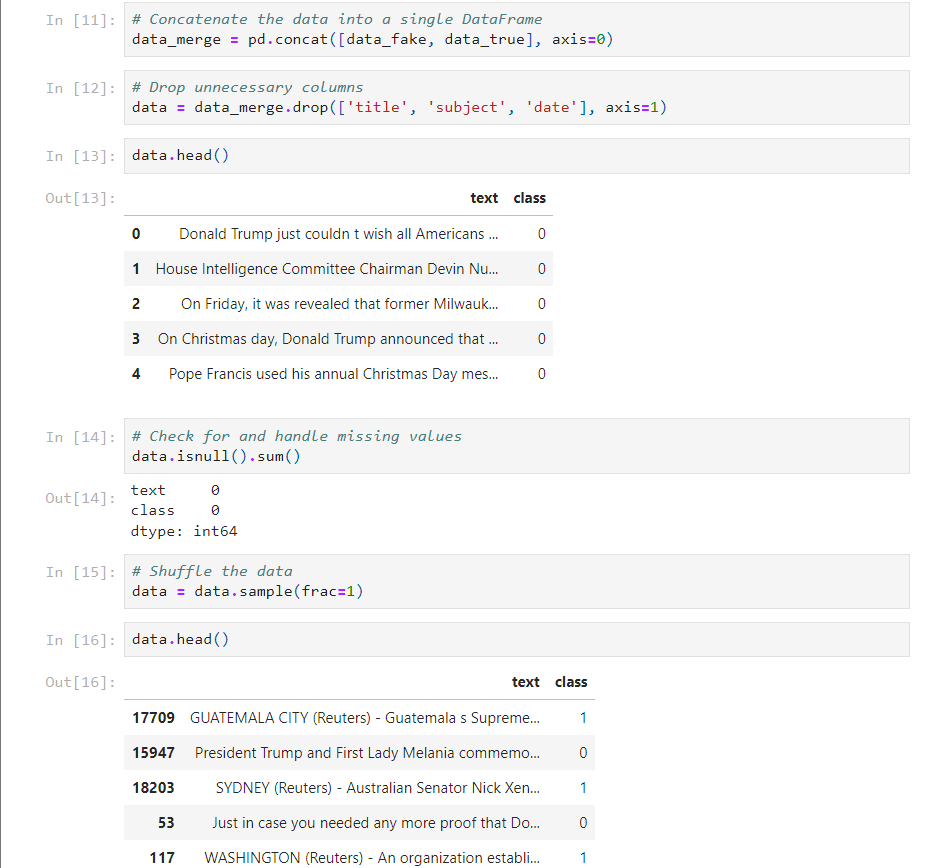
manual\_testing(news, decision\_tree)

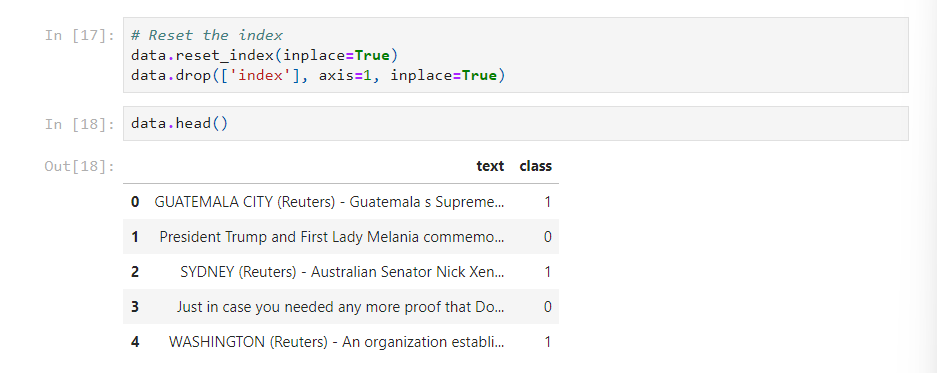
**SCREENSHOTS**

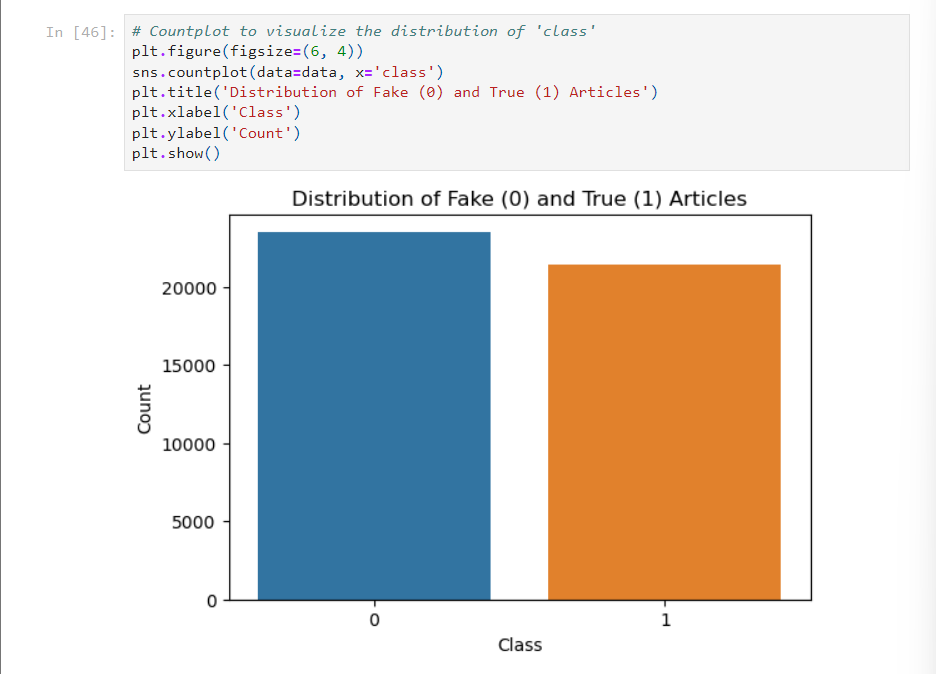


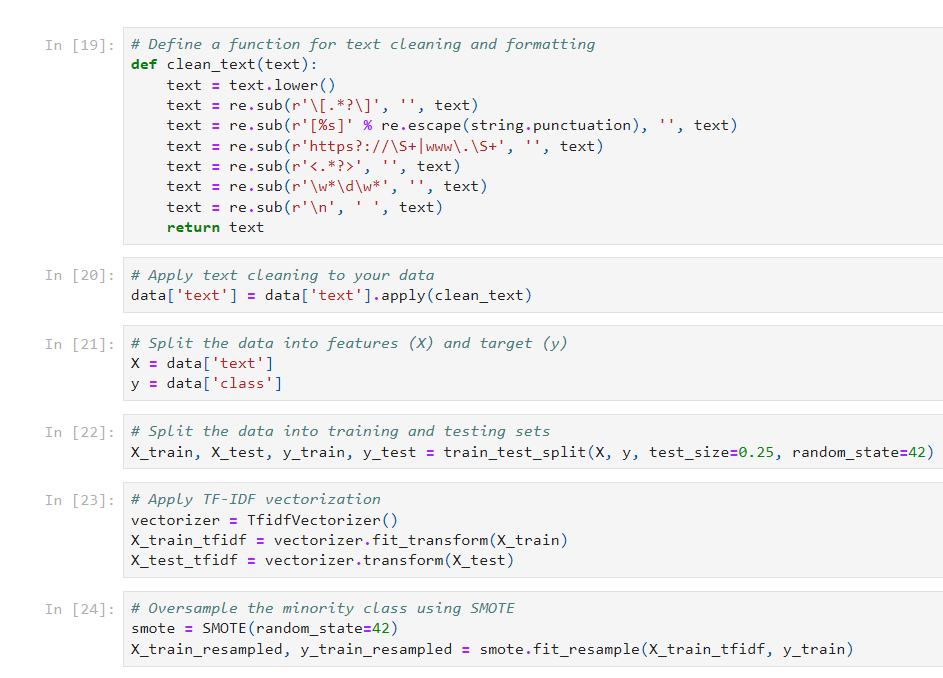


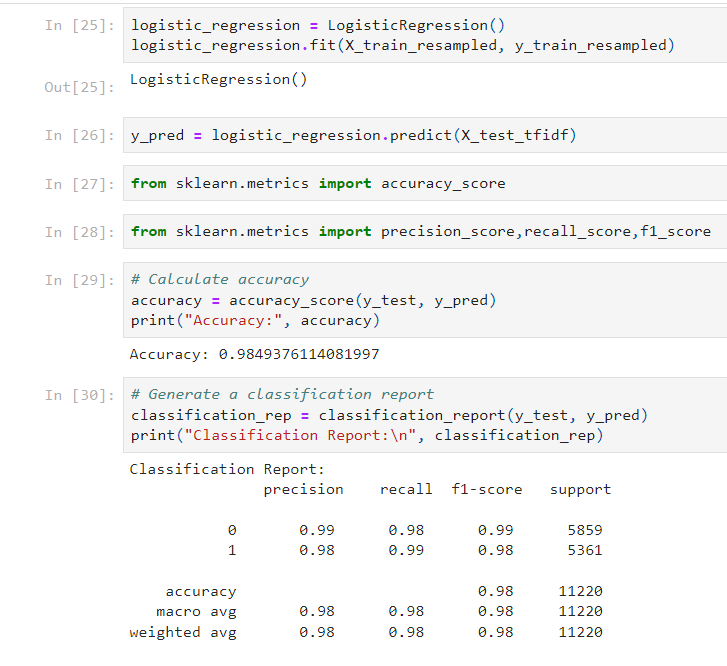




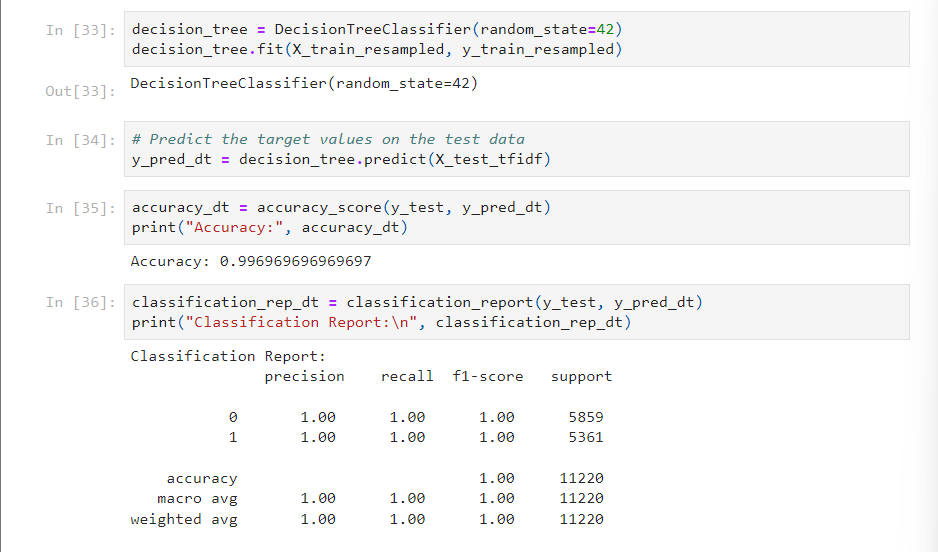


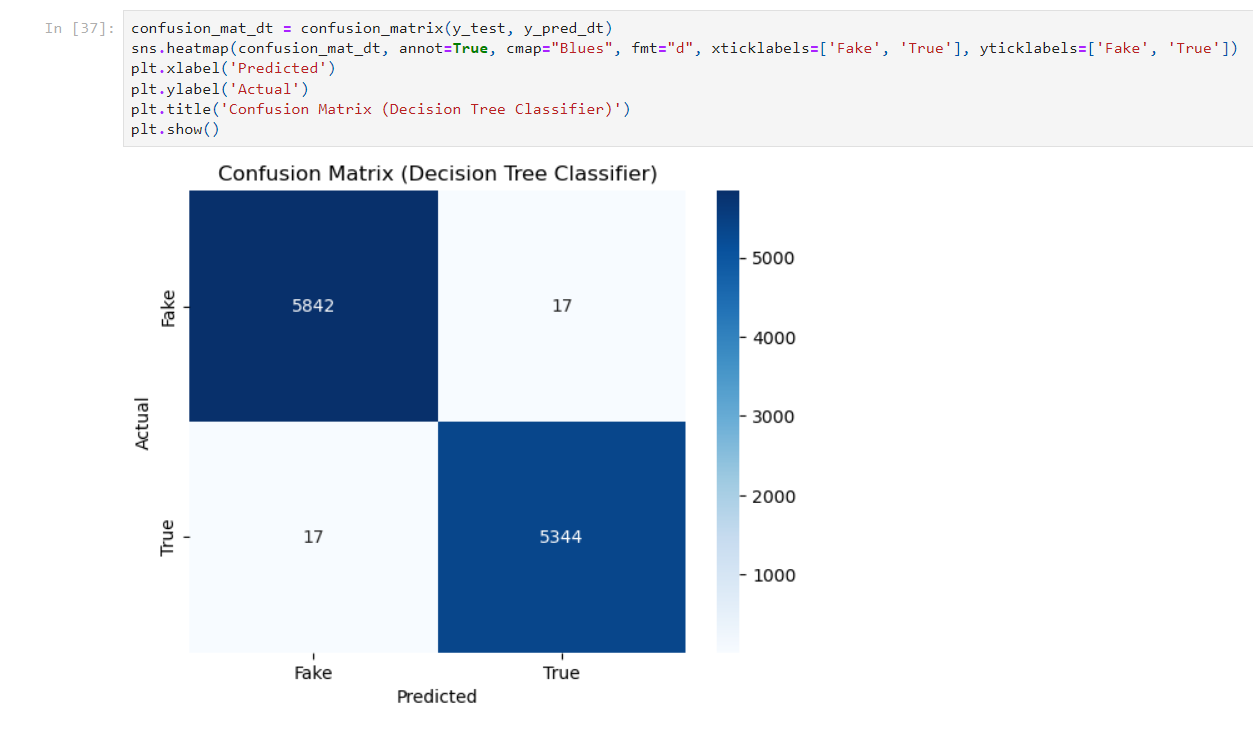


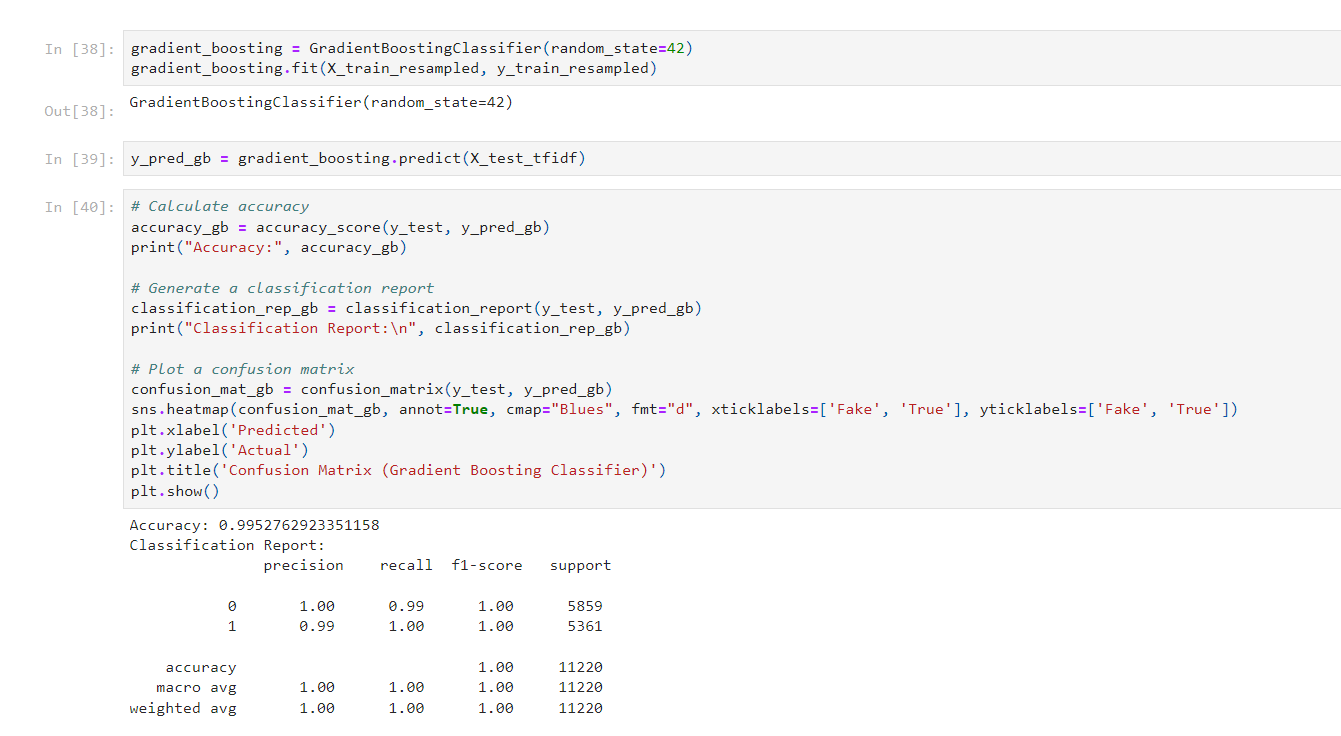


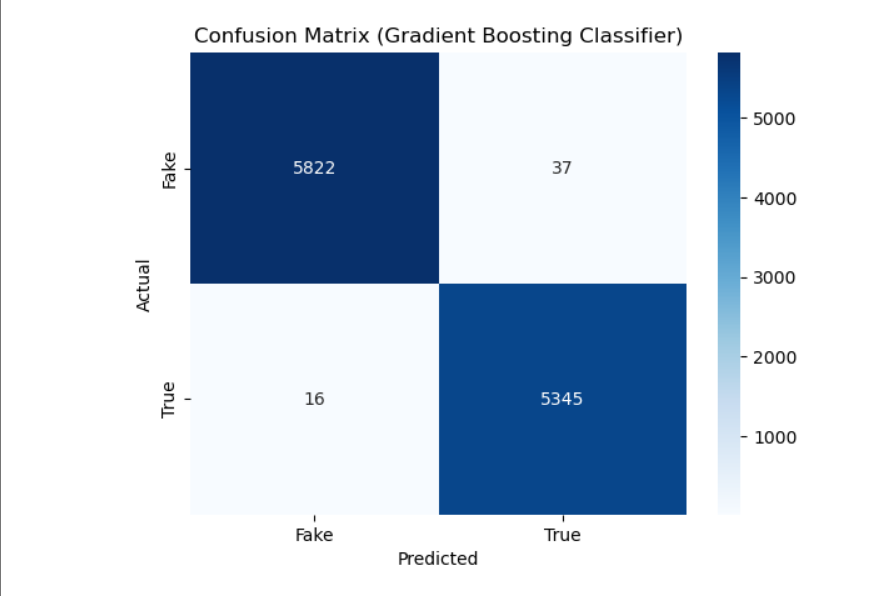


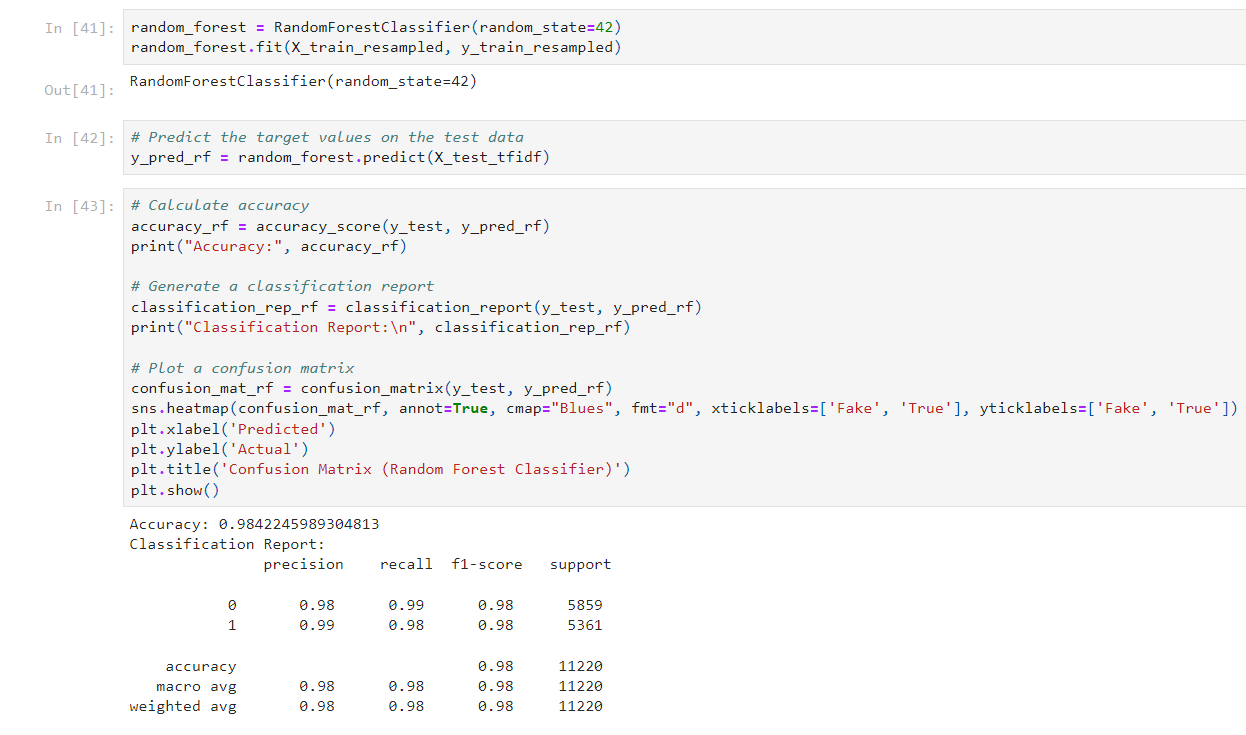


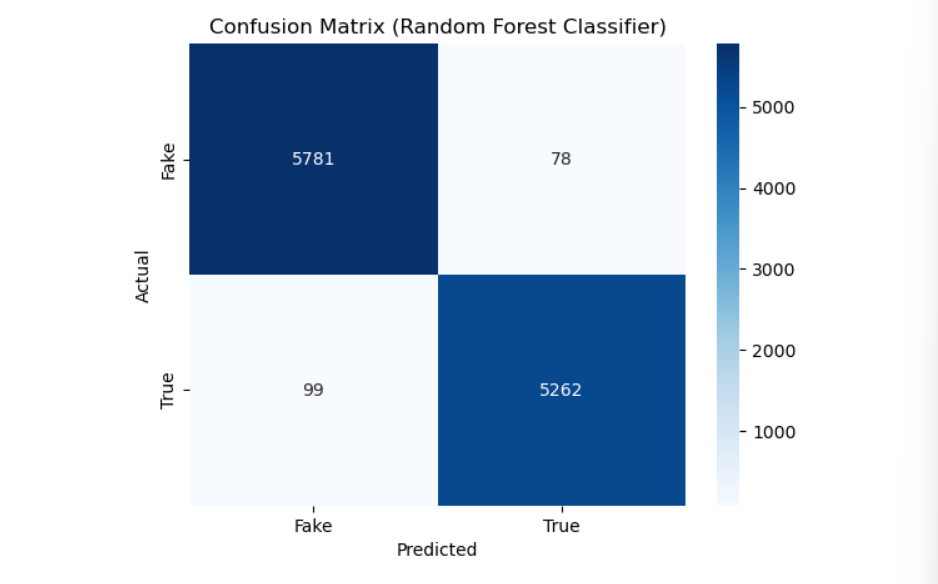


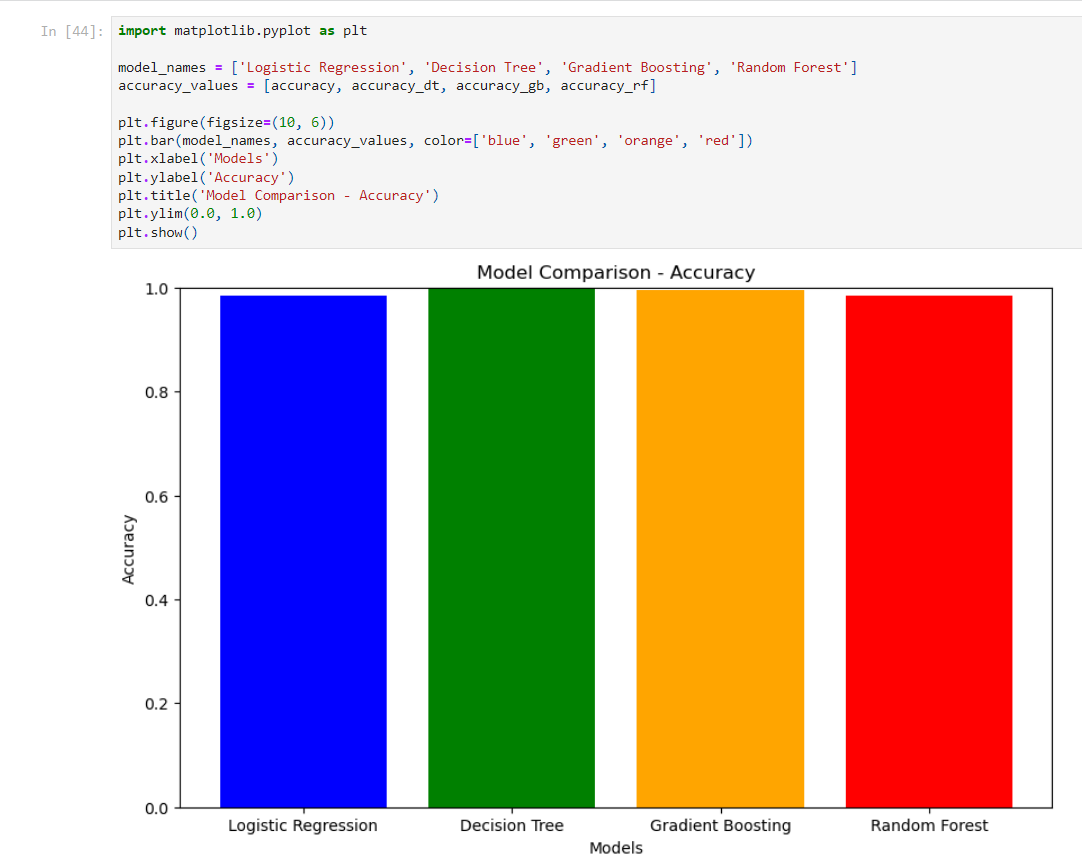


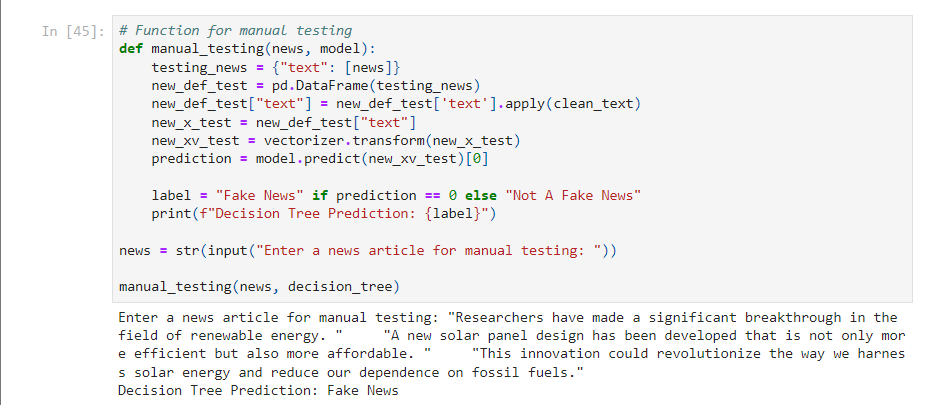




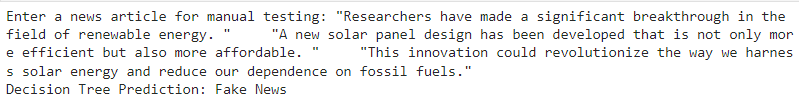


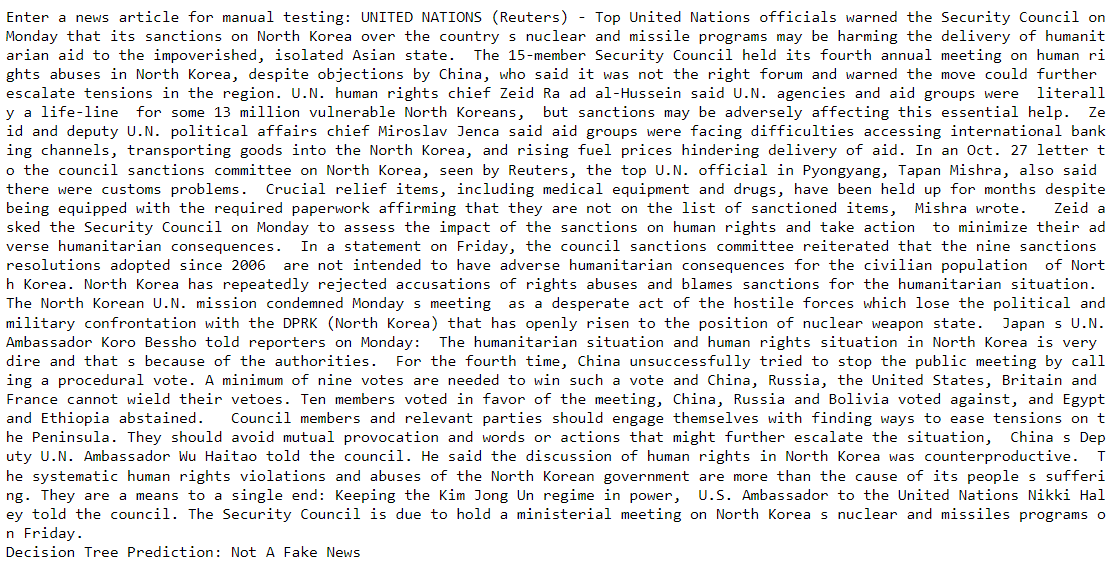






**OUTPUT**

****

****

The project produces the following key outputs:

* Model accuracy and classification reports for Logistic Regression, Decision Tree, Gradient Boosting, and Random Forest.
* A mechanism for manual testing of news articles, allowing users to input a news article for classification as fake or true.

**CONCLUSION**

In conclusion, this project successfully addresses the problem of fake news detection. It employs machine learning algorithms to classify news articles as either fake or true, achieving high accuracy and robust performance. The project provides a valuable tool for individuals, organizations, and fact-checking initiatives to combat the spread of false information in the digital age.

By implementing a range of algorithms and ensuring robust data preprocessing, the project demonstrates its ability to contribute to the ongoing efforts to promote information accuracy and credibility. Users can utilize the manual testing feature to quickly assess the authenticity of news articles, further enhancing their ability to make informed decisions.