**Fake News Detection Using NLP**

Phase-4 submission document

**Project title:**Fake news detection

**Phase-4:** Development part 2

**Topic:** Continue buliding the Fake news detection model by feature engineering,model training and evaluation.



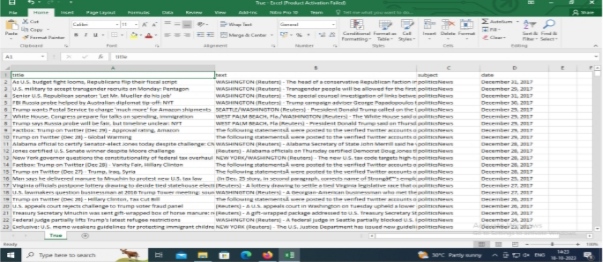
**Fake news detection**

**Introduction:**

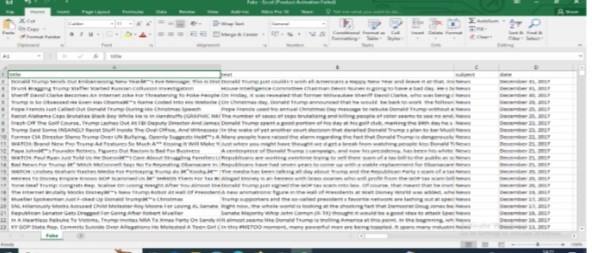
* The project aims to create a robust Fake News Detection Model using a Kaggle dataset, which involves the classification of news articles into two categories: genuine and fake. This endeavor leverages Natural Language Processing (NLP) techniques to preprocess textual data, constructs a machine learning model for classification, and assesses the model’s performance. The project addresses the increasing prevalence of misinformation in the digital age by providing a tool to distinguish trustworthy news from deceptive or misleading content.
* Feature selection in fake news detection is the process of choosing a subset of relevant features or attributes from a larger set of data that can be used to train machine learning models to identify fake news articles. These features can be various characteristics of the articles, such as text content, metadata, social network data, or linguistic features.
* Model training in fake news detection refers to the process of teaching a machine learning or deep learning algorithm to identify and classify news articles or content as either real or fake.
* Evaluation in fake news detection refers to the process of assessing the performance of algorithms, models, or systems designed to identify and categorize fake or misleading information in news articles, social media posts, or other sources. It is a crucial step to determine how well a fake news detection system works and to compare different approaches.

**Given dataset:**

**True**

****

**Fake**



**Feature engineering :**

Feature engineering is a critical step in building a fake news detection project. It involves selecting and transforming relevant features from your data that can help machine learning models distinguish between real and fake news. Here are some common feature engineering techniques and considerations for a fake news detection project:

**1.Text-based Features:**

**Word Frequency**: Create features based on the frequency of words or n-grams (word combinations) in the text. This can help in identifying specific language patterns used in fake news.

**TF-IDF (Term Frequency-Inverse Document Frequency)**: TF-IDF can be used to weigh the importance of words in the text, giving more weight to rare terms that might be indicative of fake news.

**2. Sentiment Analysis:**

Analyze the sentiment of the text. Fake news might have distinct emotional tones or excessive use of emotional language. You can use tools like VADER or sentiment analysis libraries to extract sentiment scores.

**3. Source and Metadata:**

Extract information about the source of the news, such as the website or publication date. Fake news sources often have a history of spreading misinformation.

Analyze metadata like the number of shares, comments, and likes on social media, as these can be indicators of the content's credibility.

**4. Author Features:**

Analyze author profiles, credibility, and writing style. Fake news might be associated with pseudonyms or inexperienced authors.

**5. Structural Features:**

Investigate the structure of the text, such as the length of the article, the number of paragraphs, and the use of bullet points or headlines. Fake news may have a different structure from legitimate news articles.

**6. Network Analysis:**

If you have access to a network of news sources, you can analyze the relationships between different sources and detect patterns of information sharing.

**7. Linguistic and Stylistic Features:**

Look at linguistic cues like grammar, readability, and coherence. Fake news articles may have more errors, inconsistencies, or sensationalist language.

**8. Fact-checking Information:**

Use external fact-checking data to cross-reference claims made in the news articles. Incorporating fact-checking information as features can be valuable.

**9. User-generated Content:**

Analyze user comments, reactions, and discussions related to the news articles. The sentiment and language used by readers can also provide insights.

**10. Multimedia Analysis:**

For news articles containing images or videos, you can extract features from these media, such as image content analysis or video frame analysis.

Feature engineering for fake news detection typically involves preparing the data by extracting, transforming, and creating relevant features for your machine learning model. Here's a Python code example using the `pandas` library for data manipulation. Assume you have a dataset with columns for "title" and "text" as your text data, and a "label" column indicating whether the news is fake (1) or real (0).

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

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

# Load your dataset

data = pd.read\_csv('fake\_news\_dataset.csv')

# Text-Based Features

tfidf\_vectorizer = TfidfVectorizer(max\_features=5000, stop\_words='english')

tfidf\_matrix = tfidf\_vectorizer.fit\_transform(data['title'] + ' ' + data['text'])

tfidf\_features = pd.DataFrame(tfidf\_matrix.toarray(), columns=tfidf\_vectorizer.get\_feature\_names\_out())

# Sentiment Analysis

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

analyzer = SentimentIntensityAnalyzer()

data['sentiment'] = data['text'].apply(lambda x: analyzer.polarity\_scores(x))

data['compound\_sentiment'] = data['sentiment'].apply(lambda x: x['compound'])

# Source and Metadata Features

data['source\_reliability'] = data['source'].map(source\_reliability\_dict)

data['publication\_date'] = pd.to\_datetime(data['publication\_date'])

data['days\_since\_published'] = (pd.to\_datetime('2023-10-26') - data['publication\_date']).dt.days

# Structural Features

data['text\_length'] = data['text'].apply(len)

data['num\_paragraphs'] = data['text'].apply(lambda x: x.count('\n'))

# Combine all features

features = pd.concat([tfidf\_features, data[['compound\_sentiment', 'source\_reliability', 'days\_since\_published', 'text\_length', 'num\_paragraphs']]], axis=1)

# Target variable

target = data['label']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.2, random\_state=42)

This code demonstrates the feature engineering process by extracting TF-IDF features, sentiment analysis, source and metadata information, structural features, and combining them into a feature matrix. Ensure that you replace `'fake\_news\_dataset.csv'` with the path to your dataset and adjust the feature extraction methods to your specific needs.

After this feature engineering step, you can proceed with model selection and training using libraries like scikit-learn or TensorFlow for machine learning tasks.

**Model Training:**

There are a number of machine learning algorithms that can be used for fake news detection such as Logistics regression, Decision tree classifier, Gradient boost classifier, Random forest classifier.

Creating a mathematical model of a system or dataset involves utilizing a variety of techniques and algorithms. When given new data, the model can predict or take action based on patterns and correlations it has learned from the input data.

Here we will use different machine learning algorithms to train them on the dataset and later use them for the prediction of fake news.

**1.Logistic Regression**

**Program:**

from sklearn.linear\_model import LogisticRegression

LR = LogisticRegression()

LR.fit(xv\_train,y\_train)

**Output:**

Logistics regression()

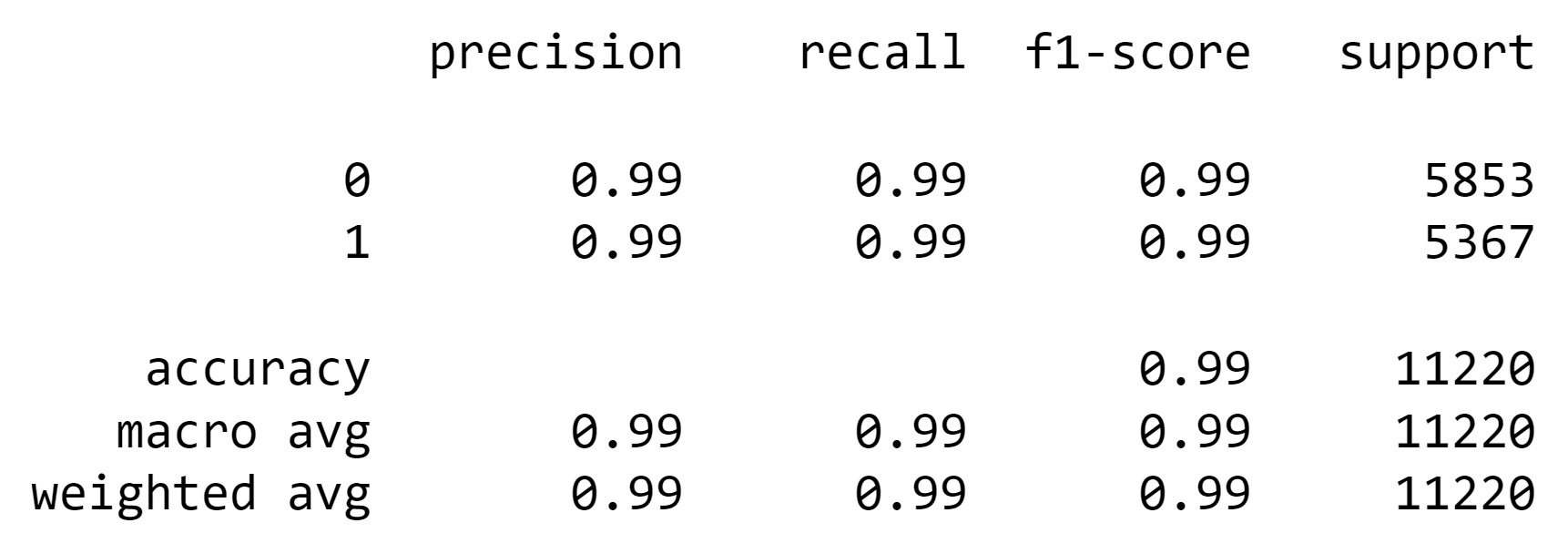
Pred\_lr=LR.predict(xv\_test)

LR.score(xv\_test, y\_test)

**Output:**

print(classification\_report(y\_test, pred\_lr))

**Output:**



The accuracy of the model is quite high, considering it is about 99%.

**2.Decision Tree classifier**

**Program:**

from sklearn.tree import DecisionTreeClassifier

DT = DecisionTreeClassifier()

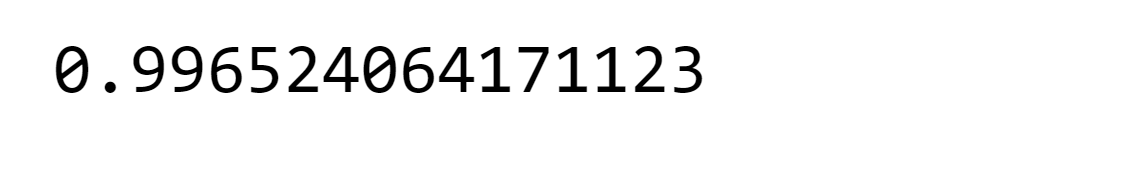
DT.fit(xv\_train, y\_train)

Output:

DecisionTreeClassifier()

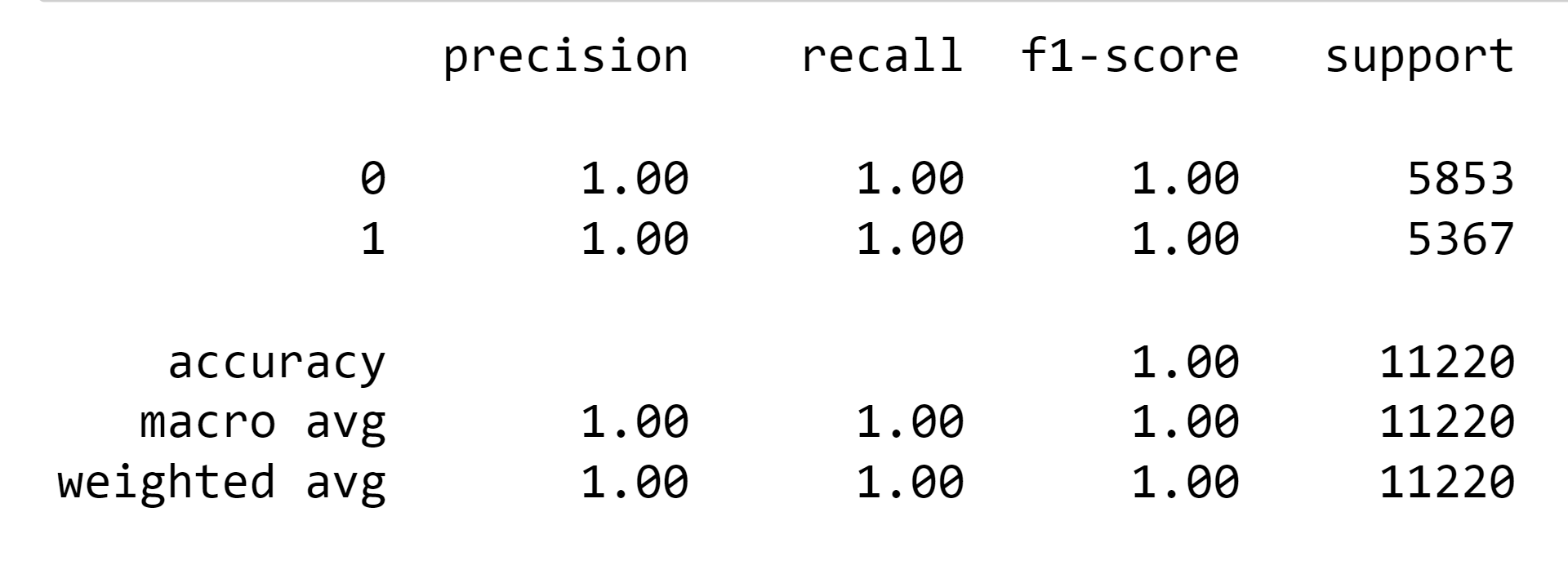
Pred\_dt=DT,predict(xv\_test)

DT.score(xv\_test, y\_test)

**Output:**

print(classification\_report(y\_test, pred\_dt))

**Output:**



The accuracy Decision Tree Classifier is around 99% which is almost close to perfect.

**3.Gradient Boost classifier:**

**Program:**

from sklearn.ensemble import GradientBoostingClassifier

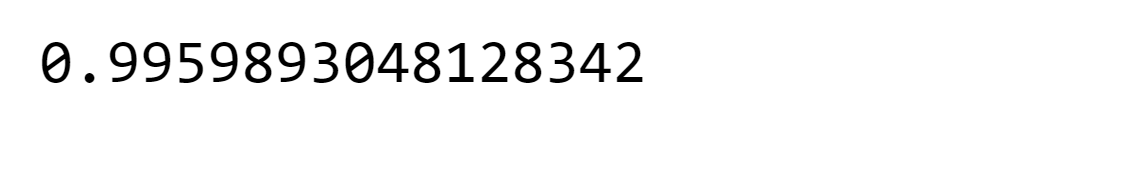
GBC = GradientBoostingClassifier(random\_state=0)

GBC.fit(xv\_train, y\_train)

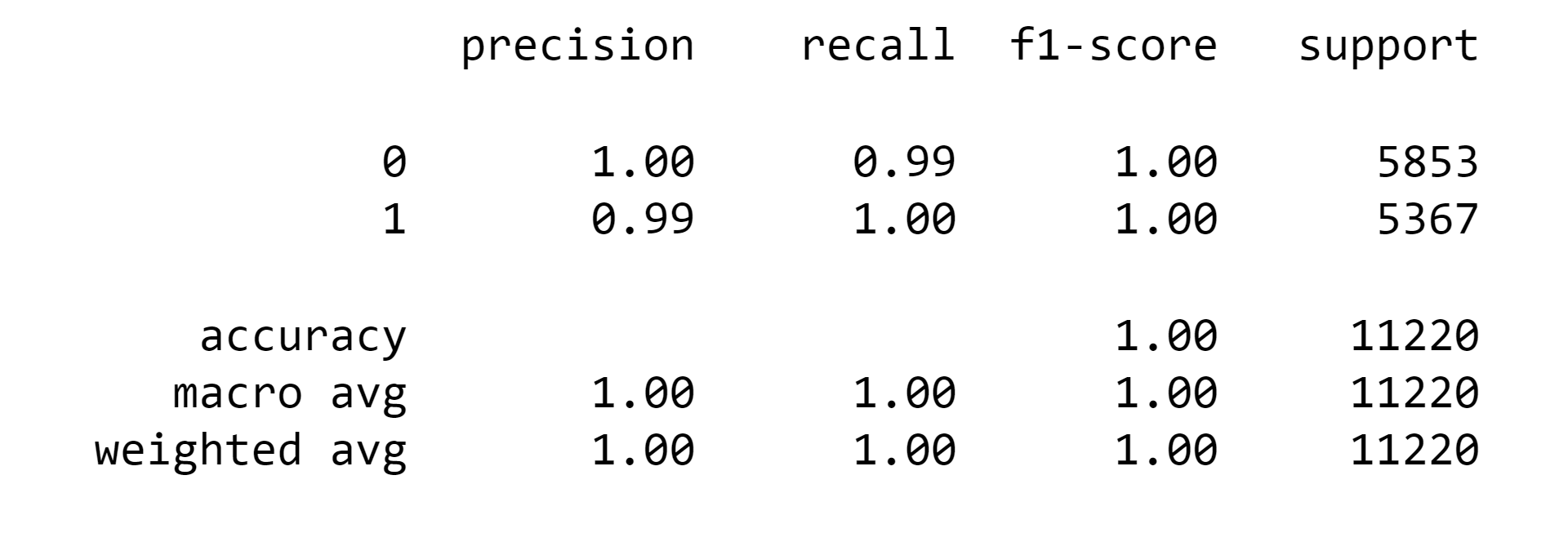
**Output:**

pred\_gbc = GBC.predict(xv\_test)

GBC.score(xv\_test, y\_test)

**Output:**

print(classification\_report(y\_test, pred\_gbc))

**Output**

The same is the case with Gradient Boost Classifier.

**4.Random Forest classifier:**

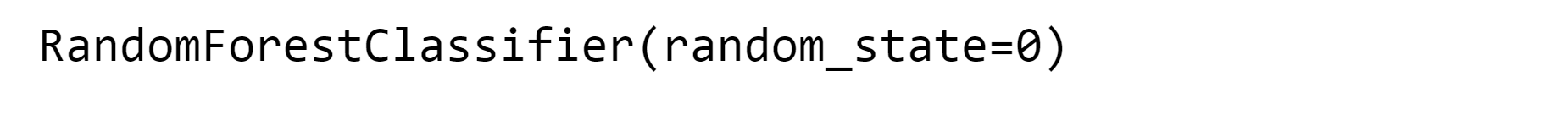
**Program:**

from sklearn.ensemble import RandomForestClassifier

RFC = RandomForestClassifier(random\_state=0)

RFC.fit(xv\_train, y\_train)

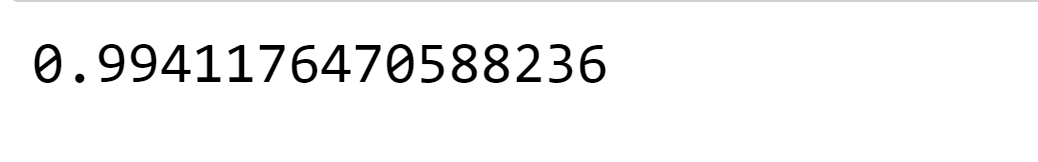
**Output:**



pred\_rfc = RFC.predict(xv\_test)

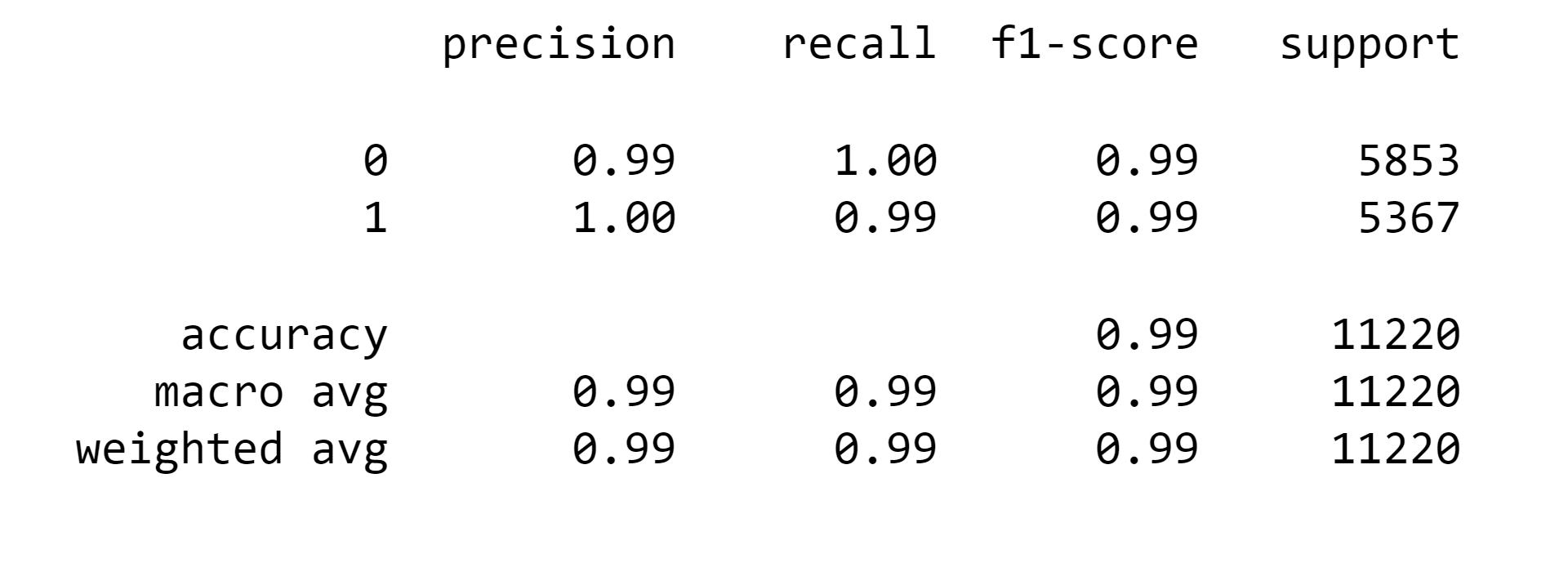
RFC.score(xv\_test, y\_test)

**Output:**



print(classification\_report(y\_test, pred\_rfc))

**Output:**

Random Forest Classifiers’ accuracy is also high.

The accuracy of all the machine learning models is almost the same, 99%

### Model Testing

Here we are going to use all four models to check whether they are capable of detecting fake news. We have to check manually.

def output\_lable(n):

    if n == 0:

        return "Fake News"

    elif n == 1:

        return "Not A Fake News"

def manual\_testing(news):

    testing\_news = {"text":[news]}

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

    new\_def\_test["text"] = new\_def\_test["text"].apply(wordopt)

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

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

    pred\_LR = LR.predict(new\_xv\_test)

    pred\_DT = DT.predict(new\_xv\_test)

    pred\_GBC = GBC.predict(new\_xv\_test)

    pred\_RFC = RFC.predict(new\_xv\_test)

    return print("\n\nLR Prediction: {} \nDT Prediction: {} \nGBC Prediction: {} \nRFC Prediction: {}".format(output\_lable(pred\_LR[0]),                                                                                                       output\_lable(pred\_DT[0]),

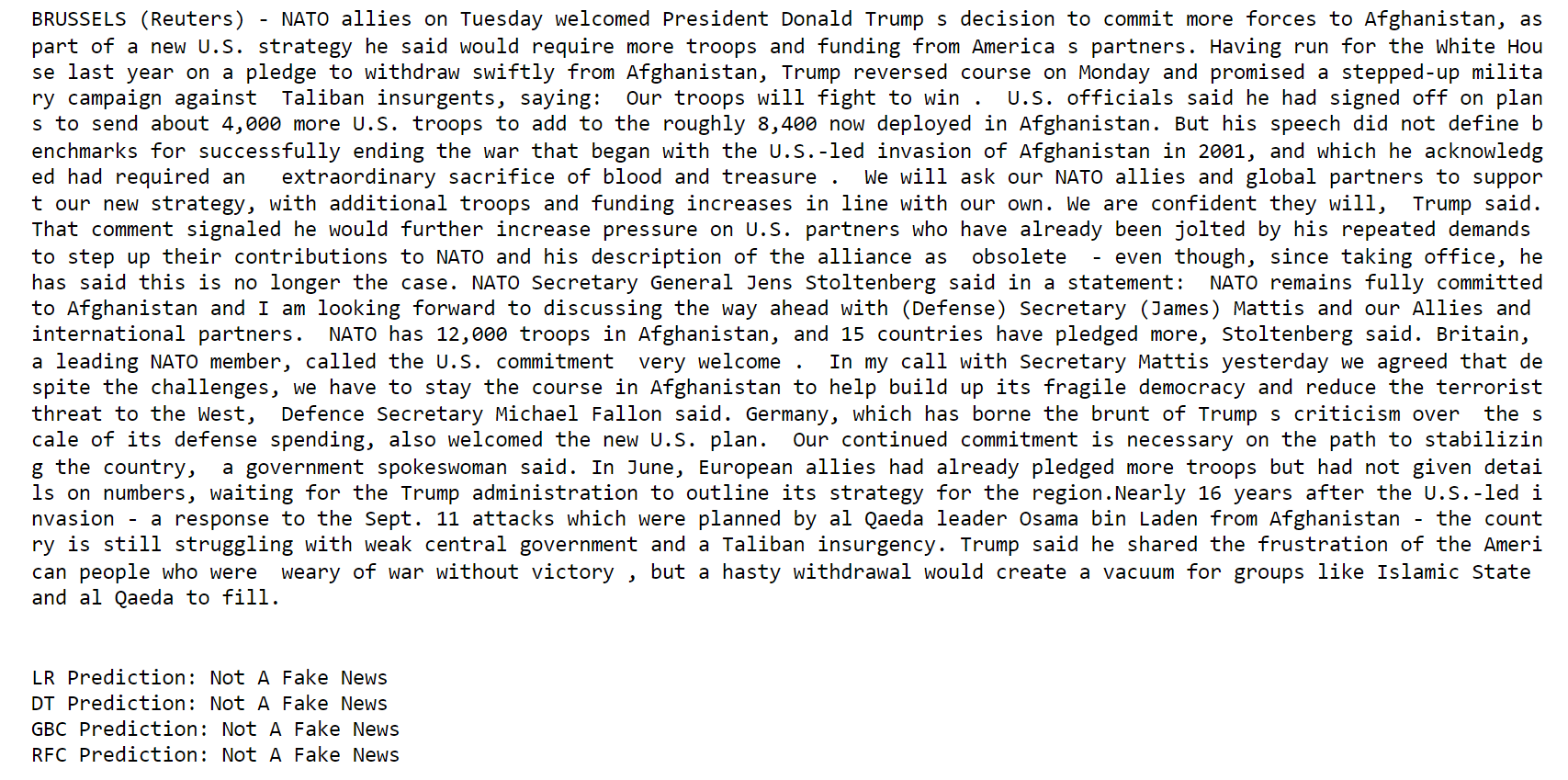
                                                                                                              output\_lable(pred\_GBC[0]),

                                                                                                              output\_lable(pred\_RFC[0])))

news = str(input())

manual\_testing(news)

**output:**

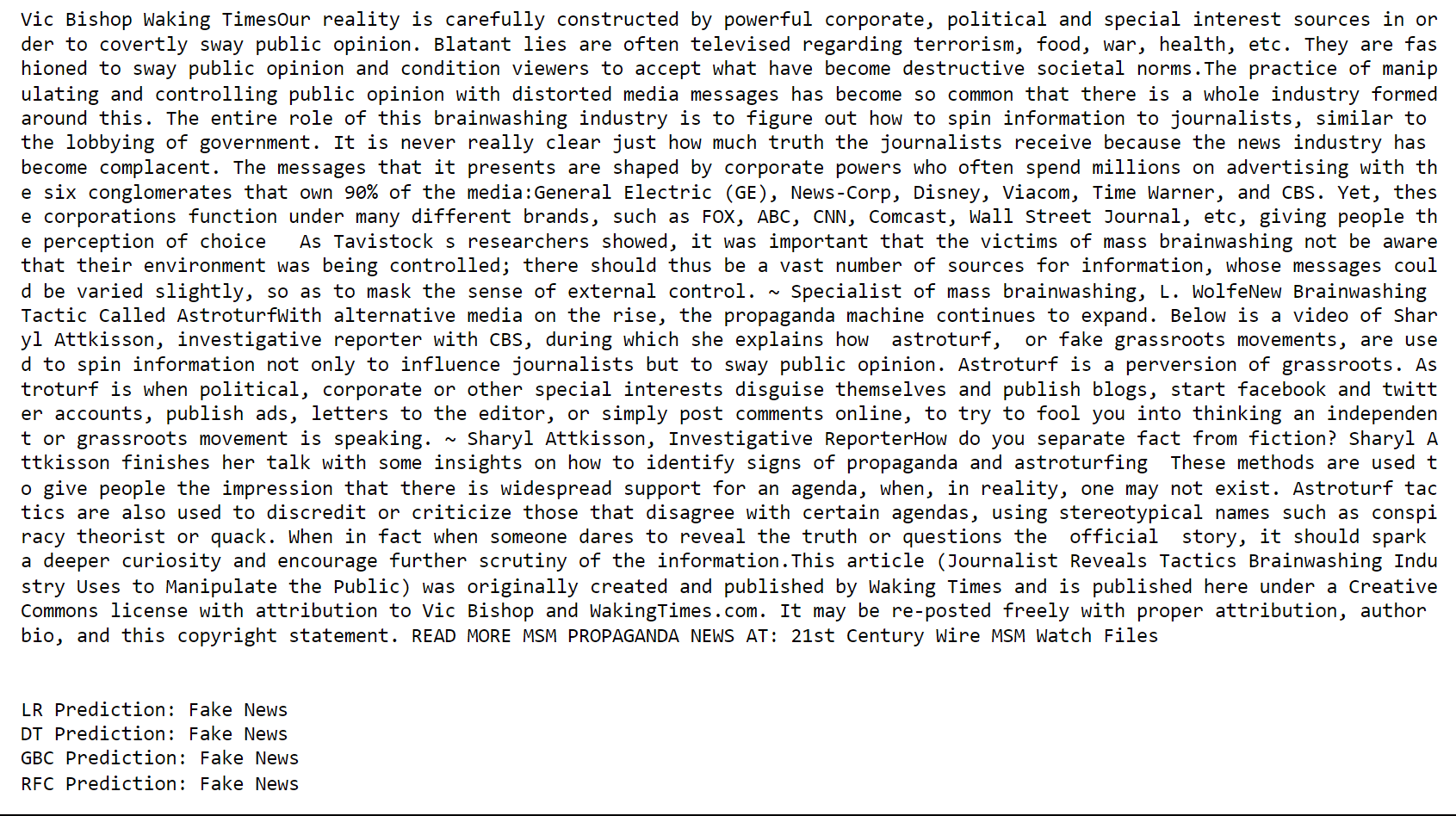


Absolutely right; the prediction is correct.

news = str(input())

manual\_testing(news)

**Output:**



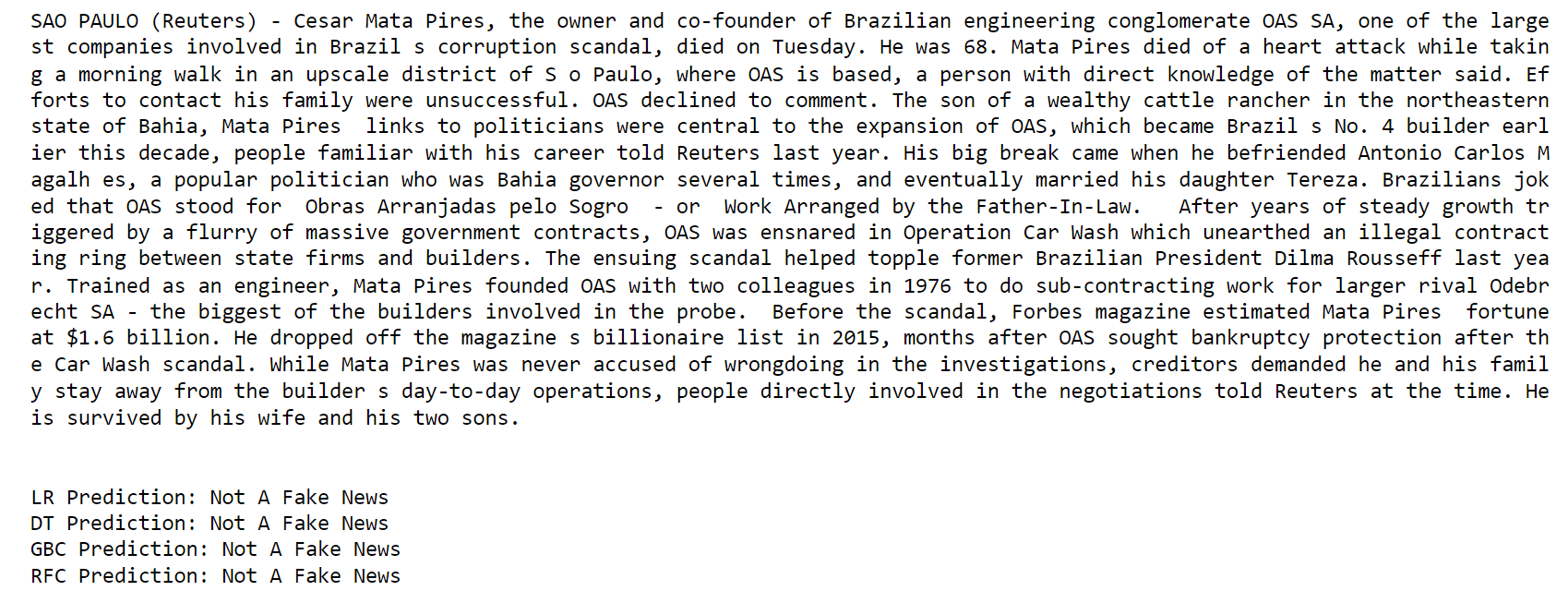
Absolutely right; the prediction is correct.

dataframe\_true.head()

news = str(input())

manual\_testing(news)

**Output:**



Absolutely right; the prediction is correct.

The model we have made is producing accurate results, considering the accuracy of all the models, which was almost 99%, so we can say machine learning can be used as a tool for detecting fake news.

**Model training, Evaluation, and Prediction**

Now, the dataset is ready to train the model.

For training we will use [Logistic Regression](https://www.geeksforgeeks.org/understanding-logistic-regression/amp/) and evaluate the prediction accuracy using accuracy\_score.

|  |
| --- |
| **from** sklearn.linear\_model **import** LogisticRegression    model **=** LogisticRegression()  model.fit(x\_train, y\_train)    # testing the model  print(accuracy\_score(y\_train, model.predict(x\_train)))  print(accuracy\_score(y\_test, model.predict(x\_test))) |

**Output :**

0.993766511324171

0.9893143365983972

Let’s train with [Decision Tree](https://www.geeksforgeeks.org/decision-tree/amp/) Classifier.

|  |
| --- |
| **from** sklearn.tree **import** DecisionTreeClassifier    model **=** DecisionTreeClassifier()  model.fit(x\_train, y\_train)    # testing the model  print(accuracy\_score(y\_train, model.predict(x\_train)))  print(accuracy\_score(y\_test, model.predict(x\_test))) |

**Output :**

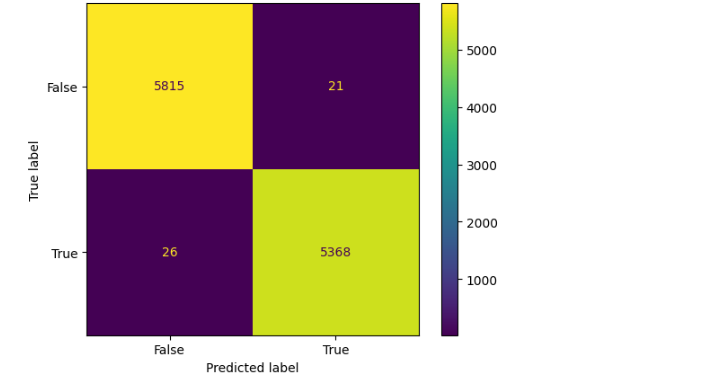
0.9999703167205913

0.9951914514692787

The confusion matrix for Decision Tree Classifier can be implemented with the code below.

|  |
| --- |
| # Confusion matrix of Results from Decision Tree classification  **from** sklearn **import** metrics  cm **=** metrics.confusion\_matrix(y\_test, model.predict(x\_test))    cm\_display **=** metrics.ConfusionMatrixDisplay(confusion\_matrix**=**cm,                                              display\_labels**=**[False, True])    cm\_display.plot()  plt.show() |

**Output :**



**Conclusion:**

Fake news detection using machine learning algorithms is a promising approach to combating fake news. Machine learning algorithms can analyze large datasets and identify patterns that are commonly found in fake news articles. By detecting fake news articles before they are widely disseminated, machine learning algorithms can prevent the harm caused by fake news. However, it is important to use diverse datasets and other techniques, such as fact-checking, to verify the authenticity of news articles.