**FAKE NEWS DETECTION USING NLP IN ARTIFICIAL INTELLIGENCE**

**TEAM MEMBER**

**au820421205045:Nilofernisha N**

**Phase-5 SUBMISSION DOCUMENT**

**Project: Fake News Detection**

**Phase 5: Project Documentation and Submission**

**PROBLEM STATEMENT:**

Detecting fake news using Natural Language Processing (NLP) involves developing a machine learning model or system that can differentiate between genuine news articles and fake or misleading information using a kaggle dataset.

**OBJECTIVES:**

* To minimize false positives and false negatives.
* Enabling timely responses to misleading information.
* Allows users to easily access and verify the authenticity of news articles.
* Focus on building public trust in the system's ability to combat fake news and provide accurate information.

**DESIGN THINKING:**

Design thinking can be a valuable approach to creating a fake news detection system using Natural Language Processing (NLP). Here's a design thinking process tailored to this specific context:

1. **Empathize:**

**Recognize the issue with fake news:** Start by learning about the subtleties and difficulties associated with fake news within the framework of NLP. This entails being aware of the characteristics, origins, and possible social repercussions of fake news.

**User research:** List the different parties—journalists, fact-checkers, social media companies, and the general public—that are involved in the identification of fake news. To learn about their particular demands and pain spots, conduct user studies, questionnaires, and interviews.

1. **Define:**

**Definition of the issue:** Specify the objectives and particular difficulties of utilizing natural language processing (NLP) to detect fake news based on your research. Establish your goals and the essential success factors, such as scalability and accuracy.

**Think with specialists:** Work together with data scientists, domain experts, and NLP experts to generate ideas for approaches and solutions for detecting false news using NLP techniques.

1. **Ideate:**

**conceive of NLP solutions:** Promote original thought and the generation of concepts for NLP-based false news detection techniques, such as topic modeling, sentiment analysis, named entity identification, and text analysis.

**Assess and rank the ideas:** Evaluate each idea's viability, possible impact, and suitability for the needs of the user. Give the most promising NLP methods a priority for future advancement.

1. **Test:**

**Carry out comprehensive testing:** Analyze the precision and effectiveness of the NLP-based system in spotting false information. Try it with several kinds of fake news, such as pictures, videos, and text.

**Obtain user opinions:** Involve prospective users to evaluate the system's usability and pinpoint areas for improvement, such as content moderators and fact-checkers.

Refine and iterate: Based on test results and user comments, make the necessary modifications to the NLP models and system.

1. **Implement:**

**Create the ultimate resolution:** Construct the NLP-based fake news detection system that is ready for production, making sure that the NLP models and algorithms are integrated.

**Connect to platforms:** Make sure the system can be incorporated into news websites, social media accounts, and other online spaces where fake news could proliferate.

1. **Iterate:**

**Frequently refresh:** Update the NLP models and system to reflect the most recent developments in NLP technology and the changing landscape of fake news.

**Adjust to fresh difficulties:** Keep an eye out for new issues and trends in fake news and adjust the system as necessary.

**PHASES OF DEVELOPMENT:**

1. Importing dataset
2. Importing libraries
3. Data preprocessing

* Concatenation of true and fake dataset
* Dropping unwanted columns
* Checking for null values
* Random shuffling of dataset

1. Model training

* Functions to process the text
* Splitting of data
* Splitting data into training and testing data

1. Feature extraction

* Text to vector

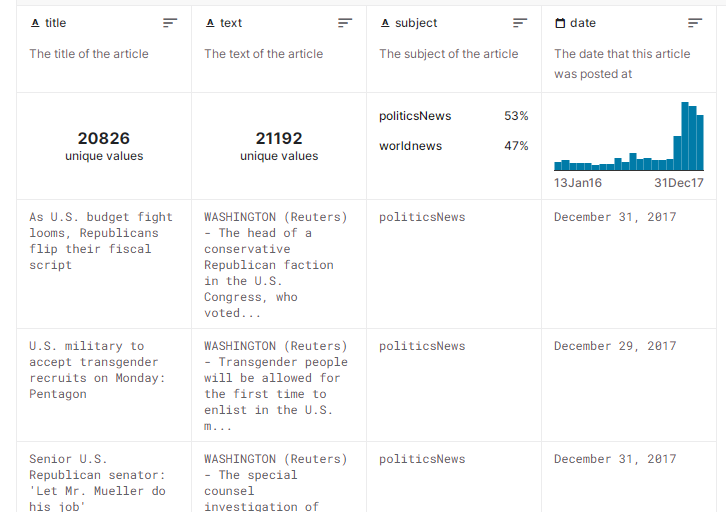
1. Classification technique

* Logistic regression
* Training the model
* Prediction
* Data visualization
* Evaluation
* Checking of our input
* Checking of our input using various classification models

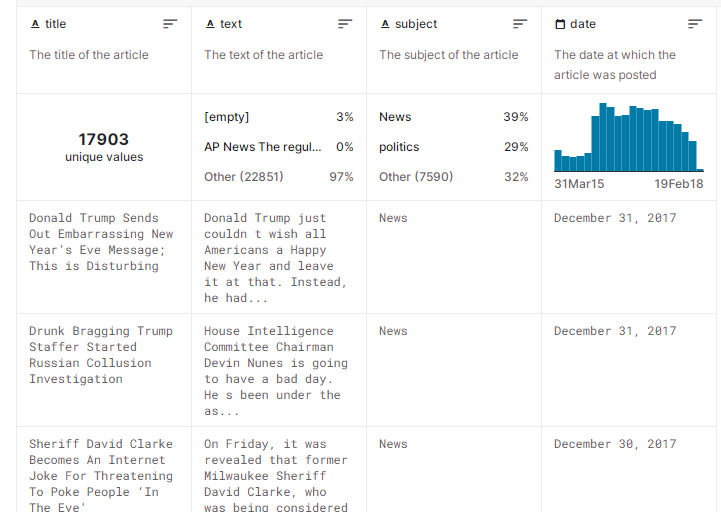
**GIVEN DATASET**

**Dataset Link:**[**https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset**](https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset)

**real.csv**

****

**fake.csv**

****

**DATA DESCRIPTION :**

**Title:** the title of a news article

**Text:** the text of the article; could be incomplete

**Subject:** display the field of the news

**Date:** publish date

**CHOICE OF CLASSIFICATION ALGORITHM:**

We have use the **Logistic Regression** algorithm for detecting whether news is real or fake.

**DEFENITION:**

**Logistic Regression:**

Logistic regression is a statistical method used for binary classification tasks. Despite its name containing the term "regression," it's actually a classification algorithm rather than a regression technique.

The primary objective of logistic regression is to predict the probability that a given input belongs to a particular category or class. It's especially useful when the dependent variable is binary, meaning it has two possible outcomes (e.g., 0 or 1, yes or no, true or false).

**NECESSARY STEPS TO FOLLOW:**

1. DATA PREPROCESSING

3. MODEL TRANNING

2. FEATURE EXTRACTION

**DATA PREPROCESSING:**

1. **Data collection:** Compile unprocessed data from a range of sources, including files, databases, web scraping, sensors, and surveys.
2. **Data Inspection:** Look over the information to gain a rudimentary comprehension of its composition and organization. This entails looking for outliers, duplicates, and missing values.
3. **Cleaning Data:**

**a. Managing Missing Data:** Choose a plan of action for handling missing data. This could involve eliminating rows that contain missing values, impute missing values using a mean or median, or apply more advanced imputation techniques.

**b. Managing Duplicates**: Find and eliminate any rows or records in the dataset that are duplicates.

1. **Data Transformation:**

**a. Feature Selection:** Choose which features (columns) are relevant to the analysis or modeling task. Eliminate irrelevant or redundant features.

**b. Feature Engineering**: Create new features or transform existing ones to better represent the underlying patterns in the data. This can involve scaling, encoding categorical variables, or creating interactions between features.

**c. Text Data Processing:** Tokenize and preprocess text data, which can involve removing punctuation, stop words, and stemming or lemmatization.

1. **Data Splitting:**

Split the data into training, validation, and test sets for machine learning tasks. The training set is used to train the model, the validation set is used to tune hyperparameters and evaluate performance during training, and the test set is used to assess the final model's performance.

1. **Data Scaling:**

If using algorithms sensitive to feature scales (e.g., k-NN, SVM), ensure that all features have been appropriately scaled or standardized.

1. **Data Visualization:**

Visualize the data to gain insights, detect patterns, and identify relationships between variables. Visualization aids in exploratory data analysis (EDA).

**FEATURE EXTRACTION:**

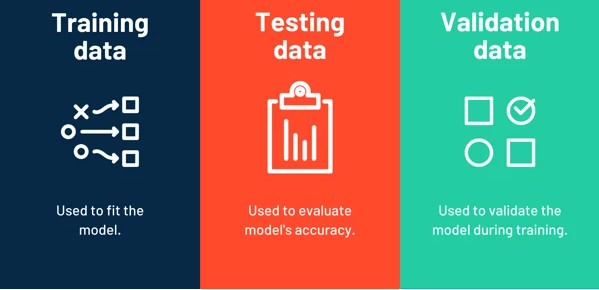
Feature extraction techniques encompass a variety of methods used to derive relevant, informative, and reduced representations from raw data. These techniques are employed across various domains such as natural language processing, image processing, signal processing, and more. Here we use word embeddings feature extraction technique.

**Word Embeddings:**

In natural language processing, word embeddings (e.g., Word2Vec, GloVe, FastText) convert words into dense, low-dimensional vectors that capture semantic relationships between words based on their usage in large text corpora.

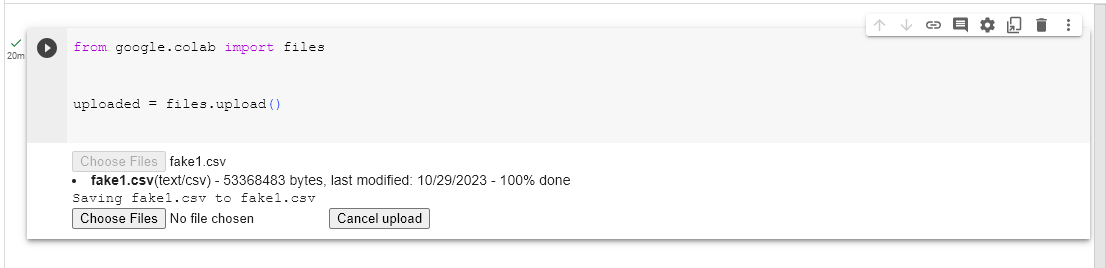
**MODEL TRANNING:**

Model training is a fundamental process in machine learning and deep learning where a machine learning model learns to make predictions or decisions based on input data. During the training process, a model is exposed to a dataset containing examples (input data) and their corresponding correct or target outputs (labels or responses). The model adjusts its internal parameters or weights to make its predictions as accurate as possible.



1. **Data Preparation:** The first step in model training involves preparing the training data. This includes collecting, cleaning, and preprocessing the data to ensure it's in a suitable format for the model. The dataset is typically split into two parts: a training set for training the model and a validation set to assess the model's performance during training and tune hyperparameters.\
2. **Training Loop:** The model is trained in a series of iterations or epochs. During each epoch, the model processes the training data, makes predictions, computes the loss, and adjusts the weights using the optimization algorithm.
3. **Validation:** At the end of each epoch or in regular intervals, the model is evaluated on the validation set to assess its generalization performance. This helps prevent overfitting, a common issue in machine learning where a model becomes too specialized on the training data and performs poorly on new, unseen data.

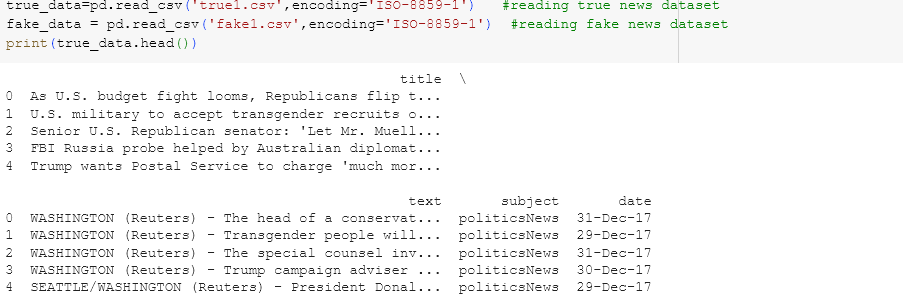
**UPLOADING DATASET**



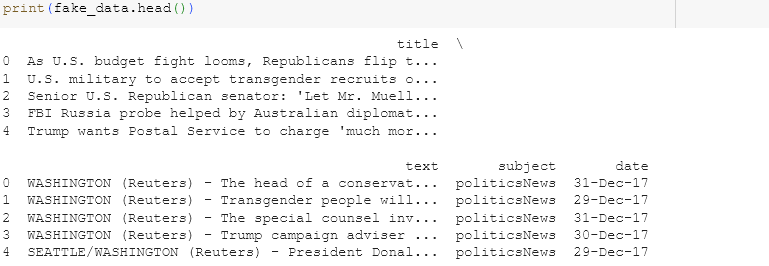
**IMPORTING LIBRARIES**

****

**TRUE DATASET:**

****

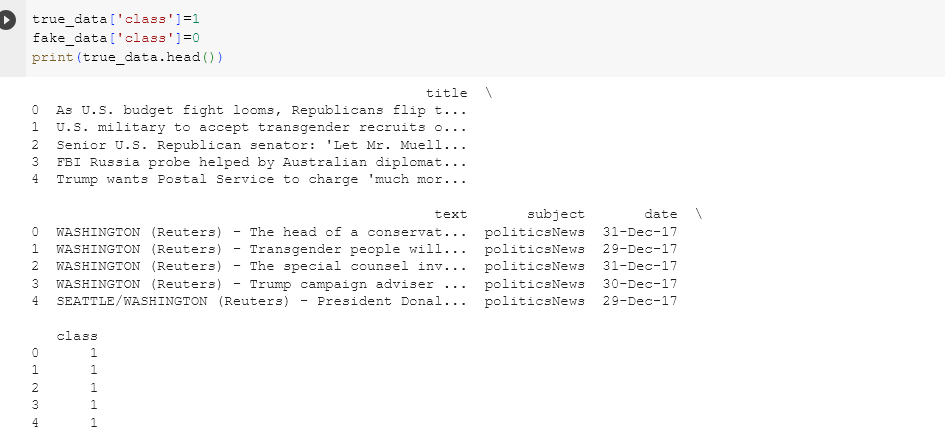
**FAKE DATASET:**

****

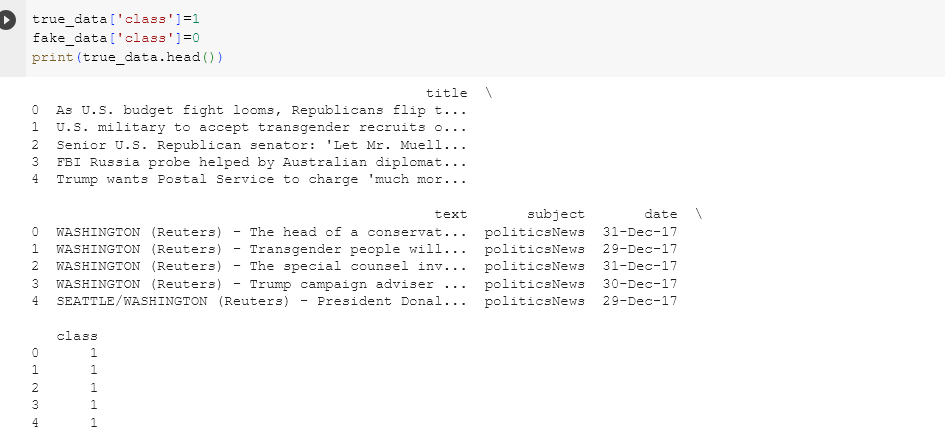
**DATA PREPROCESSING**

**Adding data attribute to dataset**

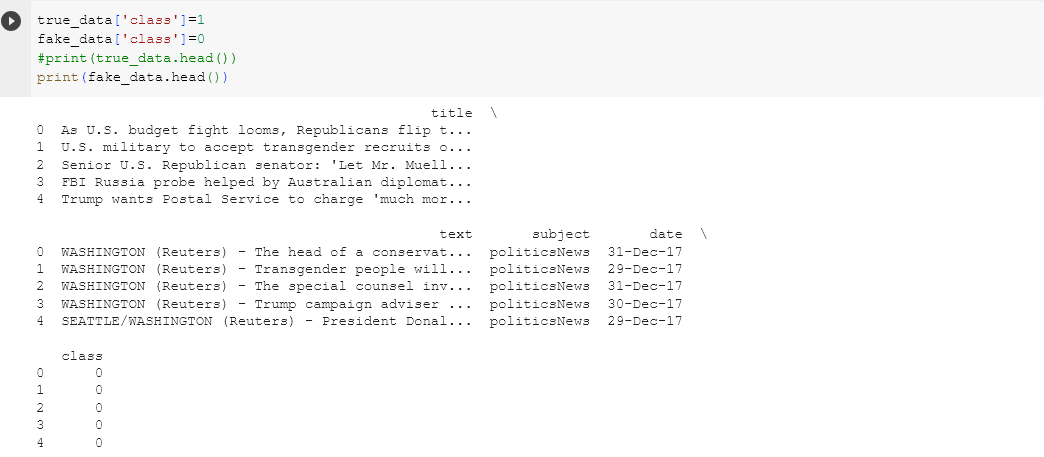
**True data**

****

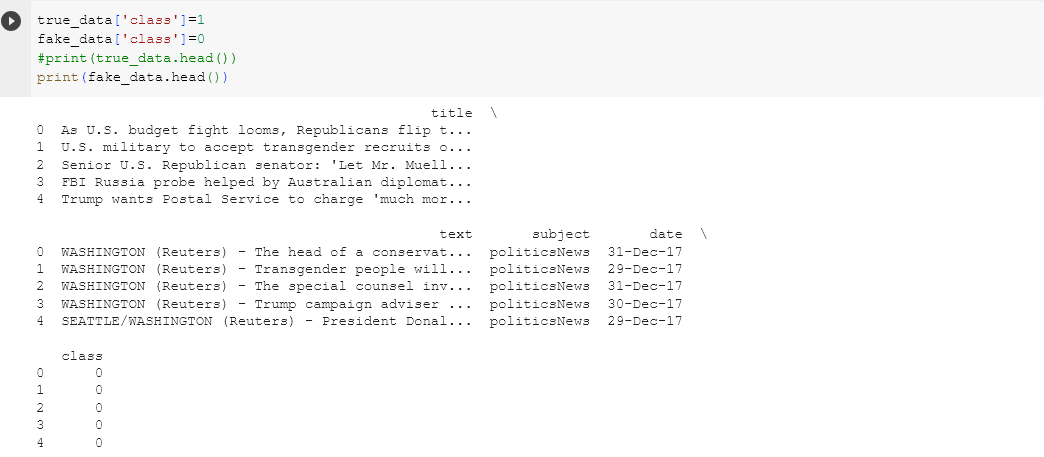
**Output**

****

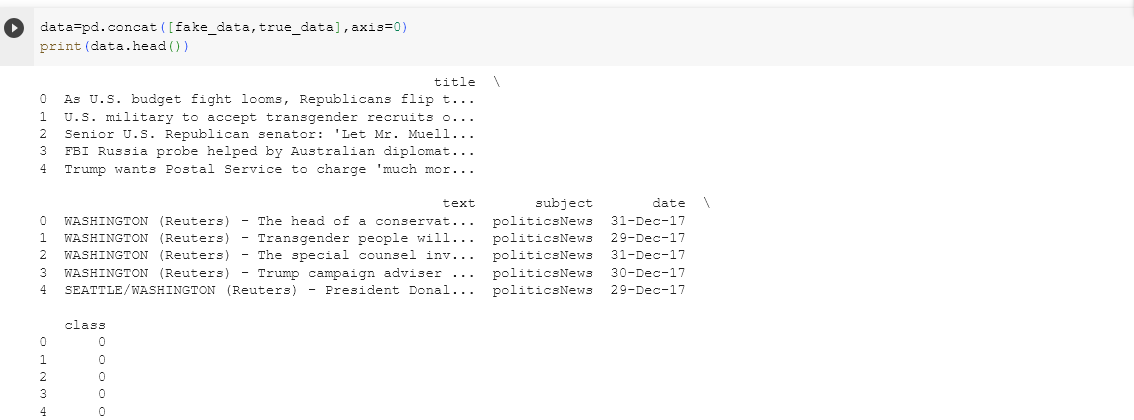
**Fake data**

****

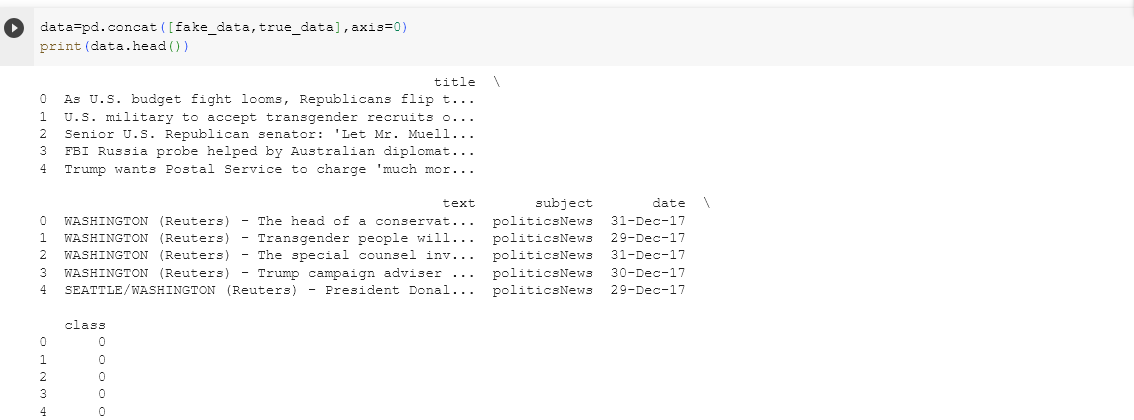
**Output**

****

**Concatenation of true and fake dataset**

****

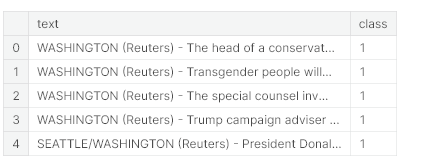
**Output**

****

**Dropping unwanted columns**

**UNWANTED.PNG**

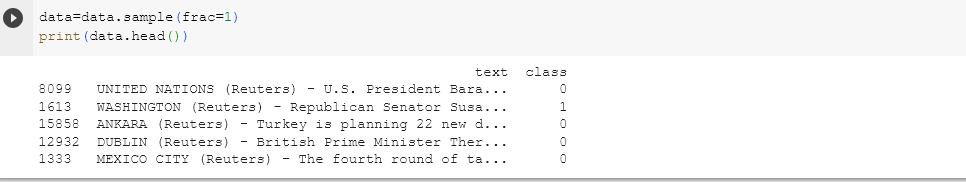
**Output**

****

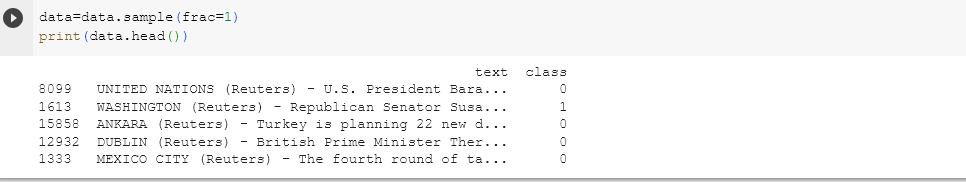
**Checking for null values**

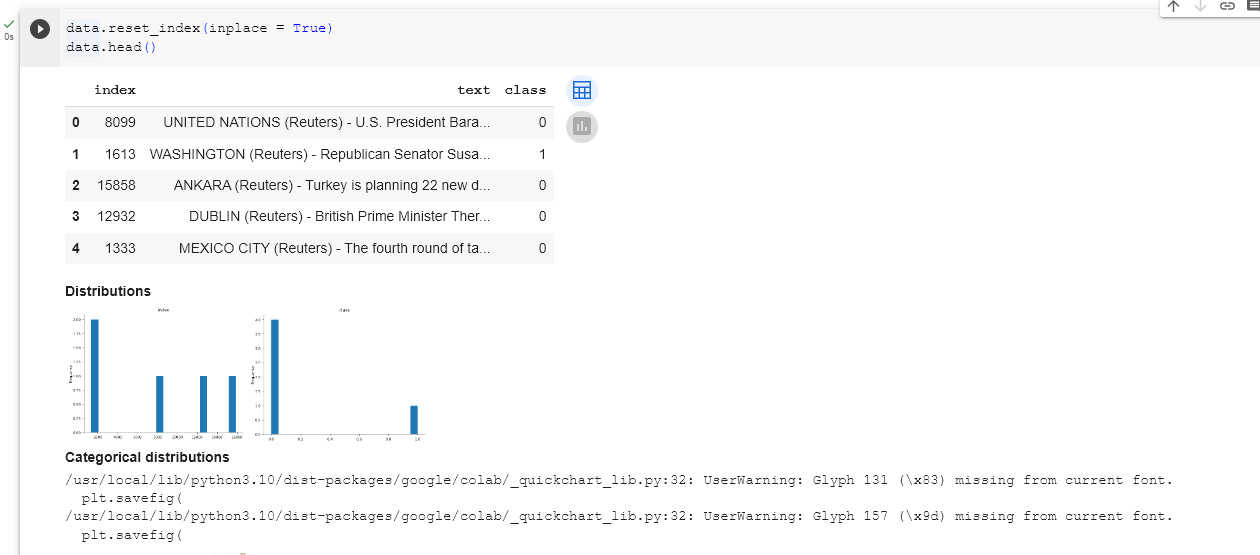
****

**Random shuffling of dataset**

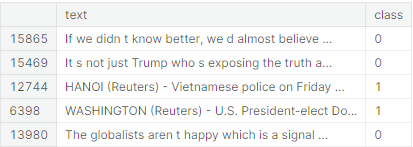
****

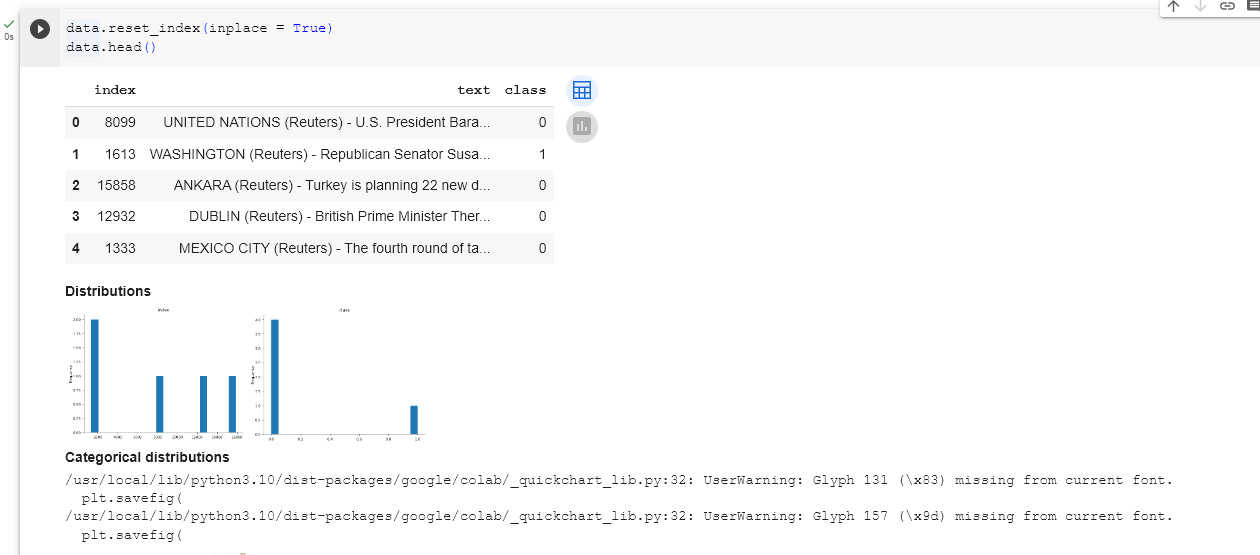
**Output**

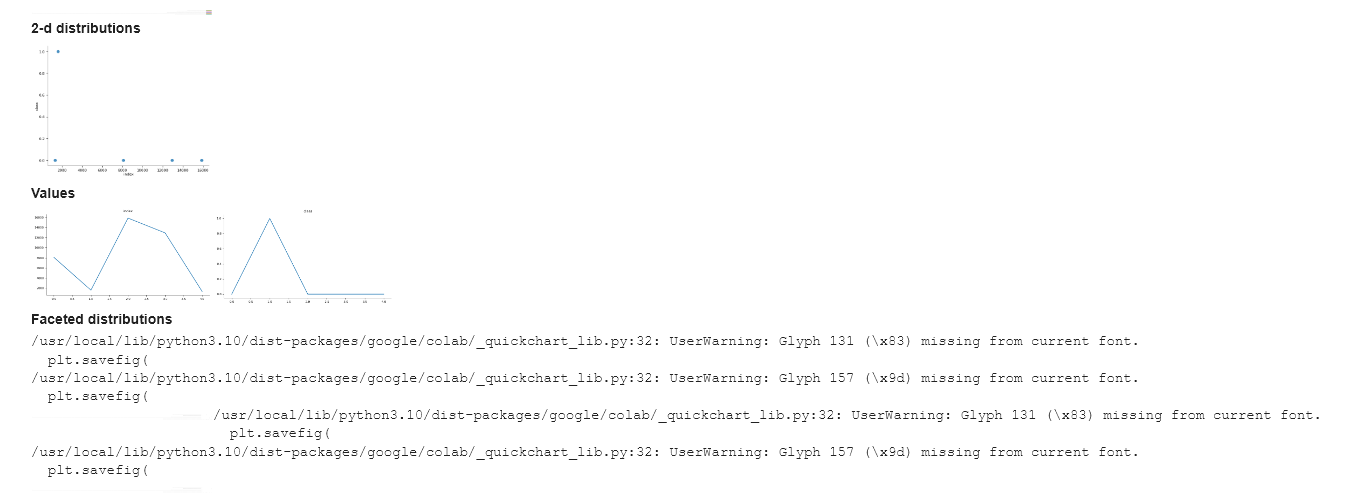
****

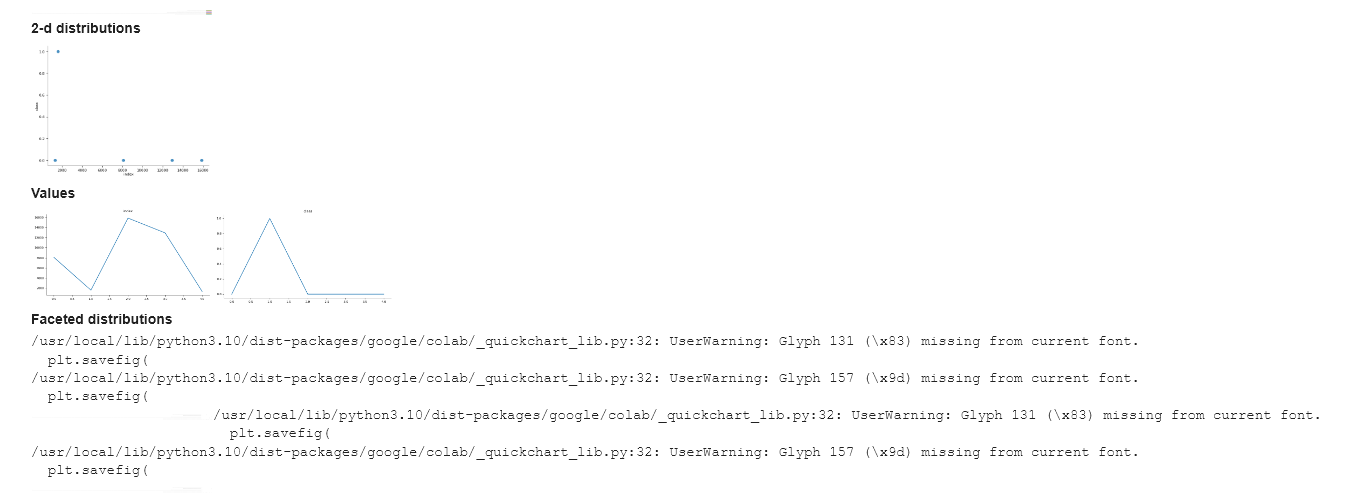
****

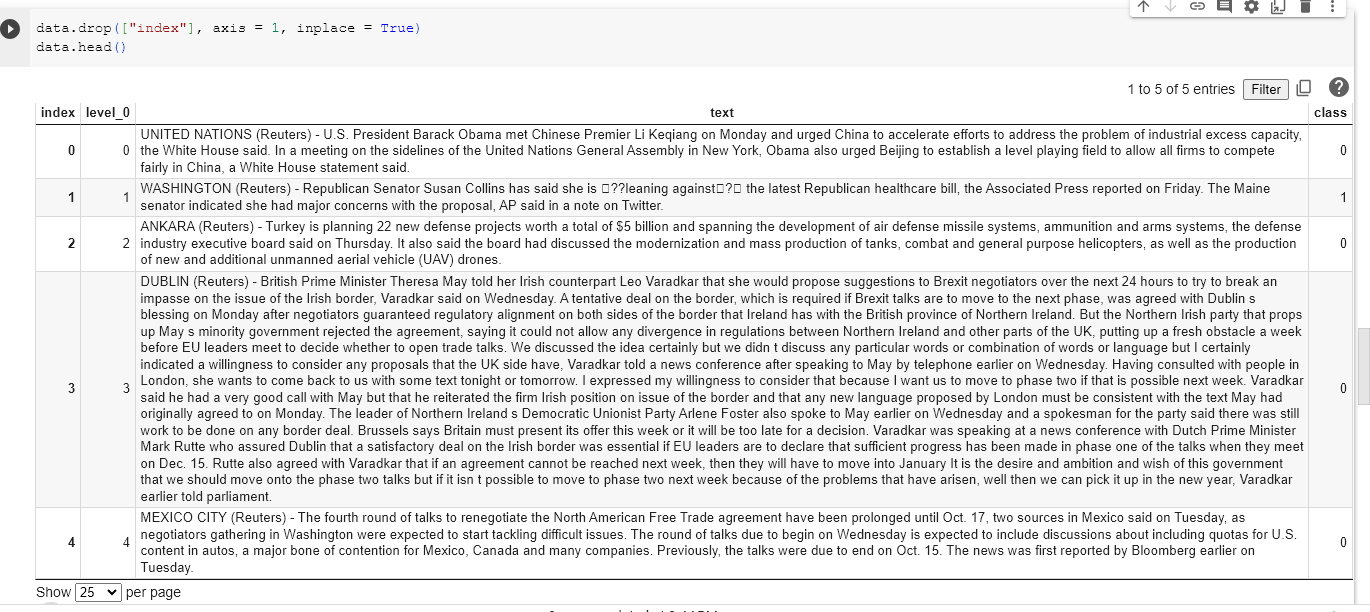
**Output**

****

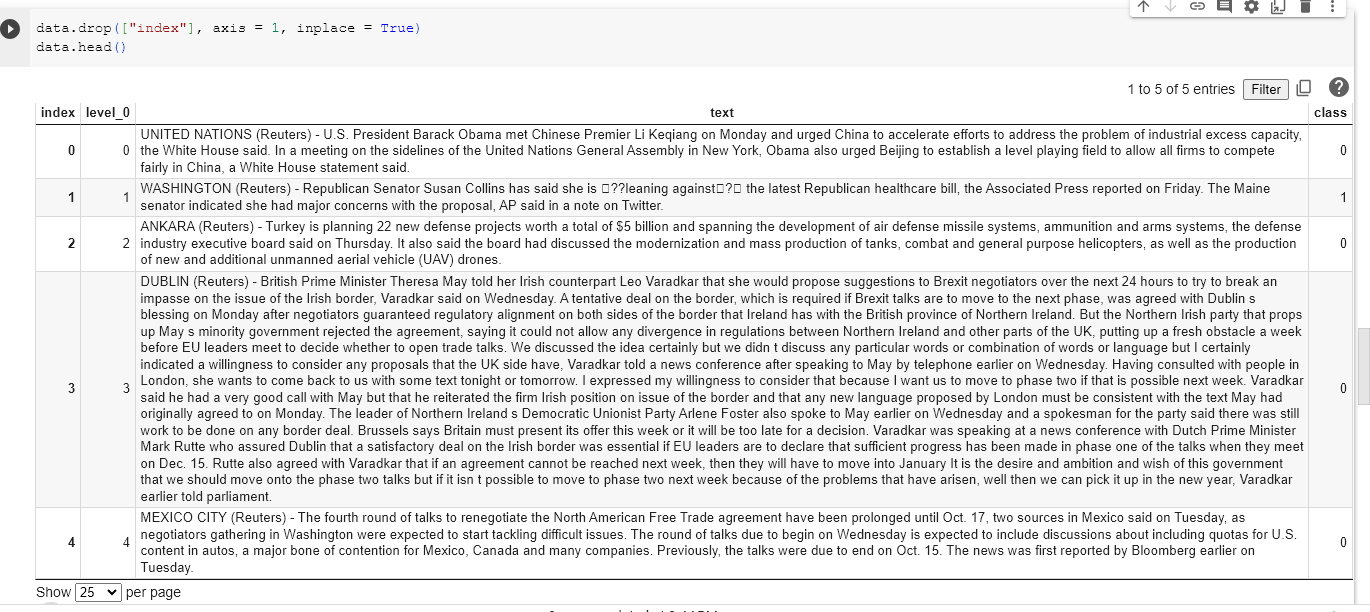
****

****



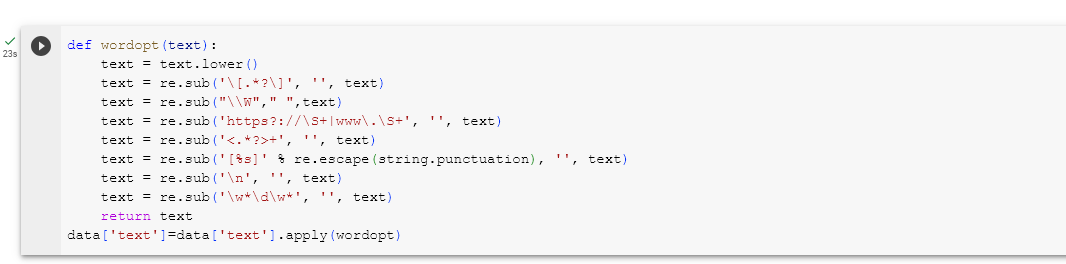
****

**Output**

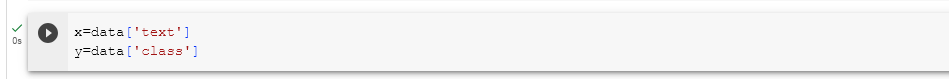
****

**MODEL TRAINING**

**Functions to process the text**

****

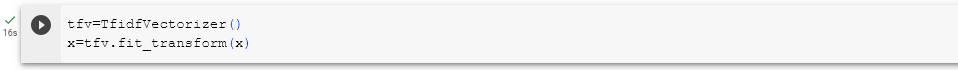
**Splitting of data**

****

**TEST AND TRAIN.PNG**

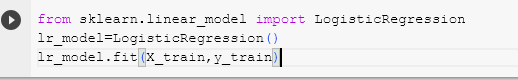
**FEATURE EXTRACTION**

**Text to vector**

****

**LOGISTIC REGERSSION**

**Training the model**

****

**Output**

**OUT.PNG**

**Prediction**

**PREDICTION.PNG**

**Output**

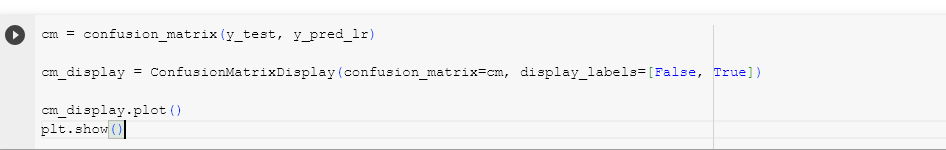
**PRE OUTPUT.PNG**

**ACC.PNG**

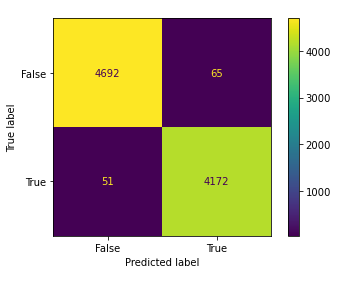
**Output**

**ACC OUT.PNG**

**Data visualization**

****

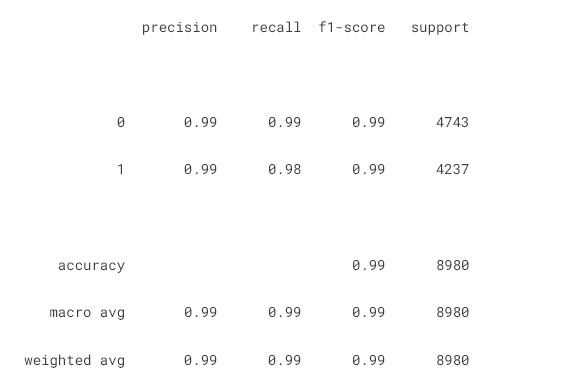
**Output**

****

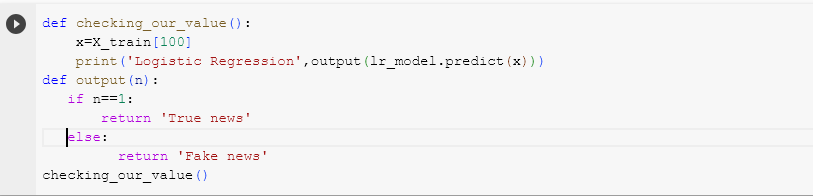
**Evaluation**

**EVAL.PNG**

**Output**

****

**CHECKING OF OUR INPUT**

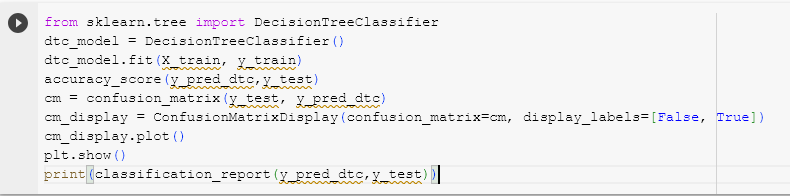
****

**Output**

**checking out.PNG**

**CHECKING OF INPUT USING VARIOUS CLASSIFICATION TECHNIQUE**

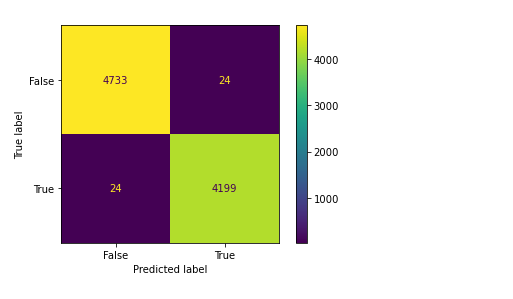
**DECISION TREE CLASSIFICATION**

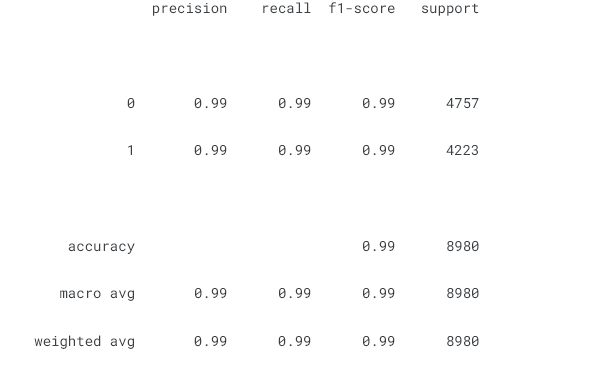
****

**Output**

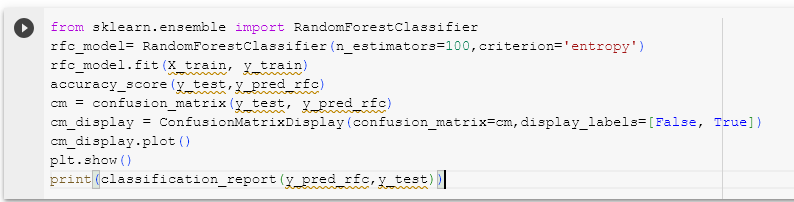
**DES1.PNG**

**DES2.PNG**

****

****

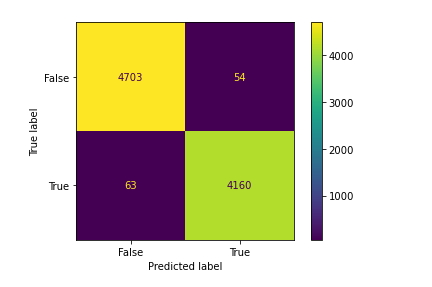
**RANDOM FOREST CLASSIFICATION**

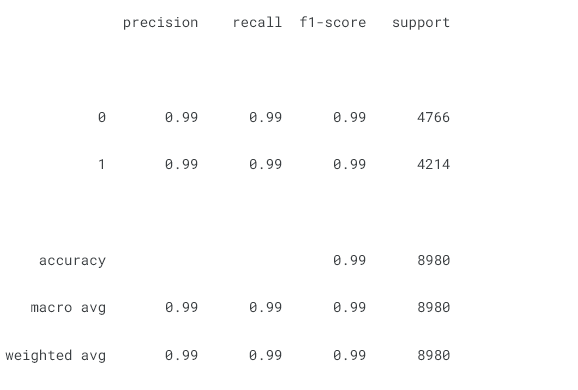
****

**Output**

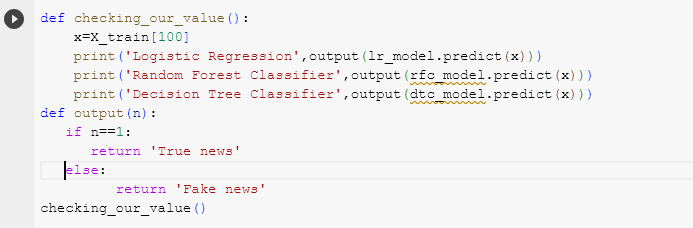
**RAN1.PNG**

**RAN2.PNG**

****

****

If you want a straightforward and interpretable model for binary classification, Logistic Regression is a good choice.

****

**Output**

****

**Full code:**

**Code:**

import numpy as np

import pandas as pd

**#for visualization of the data**

import matplotlib.pyplot as plt

import seaborn as sns

**#to split train and test data set**

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

from sklearn.feature\_extraction.text import TfidfVectorizer

**#for feature scaling**

**# for checking accuracy, precision, f1score, confusion matrix**

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix,ConfusionMatrixDisplay

**# regular expression**

import re

**# string manipulation**

import string

**#reading true news dataset**

true\_data=pd.read\_csv

**#reading fake news dataset**

fake\_data = pd.read\_csv

true\_data.head()

fake\_data.head()

**#labeling**

true\_data['class']=1

fake\_data['class']=0

true\_data.head()

fake\_data.head()

true\_data.shape

fake\_data.shape

**#Concatenation of true and fake dataset**

data=pd.concat([true\_data,fake\_data],axis=0)

data.head()

**#Dropping unwanted columns**

data.drop(['title','subject','date'],axis=1,inplace=True)

data.head()

**Checking for null values**

data.isnull().sum()

**Random shuffling of dataset**

data=data.sample(frac=1)

data.head()

data.reset\_index(inplace = True)

data.head()

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

data.head()

**Functions to process the text**

def wordopt(text):

text = text.lower()

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

text = re.sub("\\W"," ",text)

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

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

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

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

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

return text

data['text']=data['text'].apply(wordopt)

**Splitting of data**

x=data['text']

y=data['class']

**Text to vector**

tfv=TfidfVectorizer()

x=tfv.fit\_transform(x)

X\_train,X\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.20)

**Logistic regression**

from sklearn.linear\_model import LogisticRegression

lr\_model=LogisticRegression()

lr\_model.fit(X\_train,y\_train)

y\_pred\_lr=lr\_model.predict(X\_test)

y\_pred\_lr

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

cm = confusion\_matrix(y\_test, y\_pred\_lr)

cm\_display = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=[False, True])

cm\_display.plot()

plt.show()

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

**Decision Tree Classifier**

from sklearn.tree import DecisionTreeClassifier

dtc\_model = DecisionTreeClassifier()

dtc\_model.fit(X\_train, y\_train)

y\_pred\_dtc=dtc\_model.predict(X\_test)

y\_pred\_dtc

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

cm = confusion\_matrix(y\_test, y\_pred\_dtc)

cm\_display = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=[False, True])

cm\_display.plot()

plt.show()

print(classification\_report(y\_pred\_dtc,y\_test))

**Random Forest Classifier**

from sklearn.ensemble import RandomForestClassifier

rfc\_model=RandomForestClassifier(n\_estimators=100,criterion='entropy')

rfc\_model.fit(X\_train, y\_train)

y\_pred\_rfc=rfc\_model.predict(X\_test)

y\_pred\_rfc

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

cm = confusion\_matrix(y\_test, y\_pred\_rfc)

cm\_display=ConfusionMatrixDisplay(confusion\_matrix=cm,display\_labels=[False, True])

cm\_display.plot()

plt.show()

**CONCLUSION**

The use of Natural Language Processing (NLP) in detecting fake news is a promising method to combat misinformation. It uses text preprocessing techniques and feature extraction methods to capture linguistic nuances and contextual cues, requiring regular evaluation and refinement for accuracy.