FAKE NEWS PREDICTION USING MACHINE LEARNING

Submitted in partial fulfillment of the

requirements for the award of

Bachelor of Engineering degree in Computer Science and Engineering

By

S ASHISH REDDY (Reg No – 39110937)
T SRIKAR (Reg No – 39110937)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHISALAI,

CHENNAI - 600119

april – 2023

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with ¯A‖ grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

<u>BONAFIDE CERTIFICATE</u>

This is to certify that this Project Report is the bonafide work of **S ASHISH REDDY (39110937) & T SRIKAR (39111039)** who carried out the Project Phase-1 entitled **“FAKE NEWS PREDICTION USING MACHINE LEARNING”** under my supervision from June 2022 to November 2022.

J.CruzAntony.

InternalGuide
DR. CRUZANTONY J

[illegible]

Head of the Department
Dr. L.LAKSHMANAN, M.E., Ph.D.

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF CSE

Submitted for Viva voce Examination held on 20.04.2023

[illegible] [illegible]

Internal Examiner **External Examiner**

DECLARATION

I, **S ASHISH REDDY (Reg No- 39111039),** hereby declare that the Project Phase-1 Report entitled **"FAKE NEWS PREDICTION USING MACHINE LEARNING"** done by me under the guidance of **Dr. Cruz Antony J** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 04-04-2023 S SAI ASHISH REDDY

CHENNAI **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completingit successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D**, **Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **DR. Cruz Antony J,** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-1 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Huge amounts of information are generated on social networking with various social media formats. There have provided very big volumes of posts that explosive increasing of the social media data on the web. When some event has occurred, many people discuss it on the web through social networking. They search or retrieve and discuss the news events as the routine of daily life. However, a very large volume of news or posts made users face the problem of information overloading during searching and retrieving. Unreliable sources of information expose people to a dose of fake news, hoaxes, rumors, conspiracy theories, and misleading news. Some types of news such as bad events from nature phenomenal or climate are unpredictable. W hen unexpected events happen there is also fake news that is broadcasted that creates confusion due to the nature of the events. Who knows the real fact from the event while most people believe the forward news from their credible friends or relatives. The fake news comes from the misinformation, misunderstanding, or the unbelievable contents which the creditability source. These are difficult to detect whether to believe or not when they receive the news information. Thailand is located in a tropical terrain. The rain is almost throughout the year, therefore, causing massive flooding in Thailand. The Thai Meteorological Department presents the information of weather forecast, hydrological information, and local climate. They have broadcasted the forecasting information to notify the public beforehand and protect their properties. However, the unpredictable natural phenomena" news such as rain, floods, forest fires, earthquakes, storms, cold and hot weather could be rapidly spread worldwide with misleading misunderstandings. Examples in flood conditions may be rumors such that the reservoir is broken. Flooding in places that have not actually occurred. It is rumored that the water does not flood, but actually the area is flooded. These rumors cause damage in the preparation of the actual disaster. In this paper, we propose methods to detect fake news.

CHAPTER 1

INTRODUCTION

1.1 General

Fake news denotes a type of yellow press which intentionally presents misinformation or hoaxes spreading through both traditional print news media and recent online social media. Fake news has been existing for a long time, since the “Great moon hoax” published in 1835. In recent years, due to the booming developments of online social networks, fake news for various commercial and political purposes has been appearing in large numbers and widespread in the online world. With deceptive words, online social network users can get infected by these online fake news easily, which has brought about tremendous effects on the offline society already. During the 2016 US presidential election, various kinds of fake news about the candidates were widely spread in the online social networks, which may have a significant effect on the election results. According to a post-election statistical report, online social networks account for more than 41.8% of the fake news data traffic in the election, which is much greater than the data traffic shares of both traditional TV/radio/print medium and online search engines respectively. An important goal in improving the trustworthiness of information in online social networks is to identify the fake news timely, which will be the main tasks studied in this paper.

Fake news has significant differences compared with traditional suspicious information, like, in various aspects: (1) *impact on society*: usually exist in personal emails or specific review websites and merely have a local impact on a small number of audiences, while the impact fake news in online social networks can be tremendous due to the massive user numbers globally, which is further boosted by the extensive information sharing and propagation among these users (2) *audiences’ initiative*: instead of receiving spam emails passively, users in online social networks may seek for, receive and share news information actively with no sense about its correctness; and (3) *identification difficulty*: via comparisons with abundant regular messages (in emails or review websites), are usually easier to be distinguished; meanwhile, identifying fake news with erroneous information is incredibly challenging, since it requires both tedious evidence-collecting and careful fact- checking due to the lack of other comparative news articles available.

These characteristics aforementioned of fake news pose new challenges on the detection task. Besides detecting fake news articles, identifying the fake news creators and subjects will actually be more important, which will help completely eradicate a large number of fake news from the origins in online social networks. Generally, for the news creators, besides the articles written by them, we are also able to retrieve his/her profile information from either the social network website or external knowledge libraries, e.g., Wikipedia or government-internal database, which will provide fundamental complementary information for his/her background check. Meanwhile, for the news subjects, we can also obtain its textual descriptions or other related information, which can be used as the foundations for news subject credibility inference. From a higher-level perspective, the tasks of fake news article, creator and subject detection are highly correlated, since the articles written from a trustworthy person should have a higher credibility, while the person who frequently posting unauthentic information will have a lower credibility on the other hand. Similar correlations can also be observed between news articles and news subjects. In the following part of this paper, without clear specifications, we will use the general fake news term to denote the fake news articles, creators and subjects by default.

1.2 OBJECTIVE

The main purpose of this project is to train the model in classifying datasets and predicting a piece of news whether is fake or not fake. To check the authenticity of the news with the most accuracy using machine learning algorithms.

1.3 LITERATURE SURVEY

Mykhailo Granik : In their paper shows a simple approach for fake news detection using naive Bayes classifier. This approach was implemented as a software system and tested against a data set of Facebook news posts. They were collected from three large Facebook pages each from the right and from the left, as well s three large mainstream political news pages (Politico, CNN, ABC News). They achieved classification accuracy of approximately 74%. Classification accuracy for fake news is slightly worse. This may be caused by the skewness of the dataset: only 4.9% of it is fake news.

Himank Gupta : Gave a framework based on different machine learning approach that deals with various problems including accuracy shortage, time lag (BotMaker) and high processing time to handle thousands of tweets in 1 sec. Firstly, they have collected 400,000 tweets from HSpam14 dataset. Then they further characterize the 150,000 spam tweets and 250,000 non- spam tweets. They also derived some lightweight features along with the Top-30 words that are providing highest information gain from Bag-of- Words model. 4. They were able to achieve an accuracy of 91.65% and surpassed the existing solution by approximately18%.

Marco L. Della Vedova : First proposed a novel ML fake news detection method which, by combining news content and social context features, outperforms existing methods in the literature, increasing its accuracy up to 78.8%. Second, they implemented their method within a Facebook Messenger Chabot and validate it with a real-world application, obtaining a fake news detection accuracy of 81.7%. Their goal was to classify a news item as reliable or fake; they first described the datasets they used for their test, then presented the content-based approach they implemented and the method they proposed to combine it. with a social-based approach available in the literature. The resulting dataset is composed of 15,500 posts, coming from 32 pages (14 conspiracy pages, 18 scientific pages), with more than 2, 300, 00 likes by 900,000+ users. 8,923 (57.6%) posts are hoaxes and 6,577 (42.4%) are non-hoaxes.

Cody Buntain : Develops a method for automating fake news detection on Twitter by learning to predict accuracy assessments in two credibility- focused Twitter datasets: CREDBANK, a crowd sourced dataset of accuracy assessments for events in Twitter, and PHEME, a dataset of potential rumours in Twitter and journalistic assessments of their accuracies. They apply this method to Twitter content sourced from BuzzFeeds fake news dataset. A feature analysis identifies features that are most predictive for crowd sourced and journalistic accuracy assessments, results of which are consistent with prior work. They rely on identifying highly retweeted threads of conversation and use the features of these threads to classify stories, limiting this works applicability only to the set of popular tweets. Since the majority of tweets are rarely retweeted, this method therefore is only usable on a minority of Twitter conversation threads.

Shivam B. Parikh : Aims to present an insight of characterization of news story in the modern diaspora combined with the differential content types of news story and its impact on readers. Subsequently, we dive into existing fake news detection approaches that are heavily based on text- based analysis, and also describe popular fake news datasets. We conclude the paper by identifying 4 key open research challenges that can guide future research. It is a theoretical Approach which gives Illustrations of fake news detection by analyzing the psychological factors.

1.4 PROPOSED SYSTEM

In this paper, there are some classification approaches for detecting fake online reviews, which are supervised techniques. Multi Layer Perception and Support Vector Machines (SVM) are used as classifiers in our research work to improve the performance of classification. This system focus on the content of the review-based approaches. A feature used is word frequency count, sentiment polarity, and length of review. We have used five classifiers in this project, they are Logistic Regression, Support Vector machine, K Nearest Neighbors, Multi Layer Perception, Random Forest. Out of which, the logistic regression showed the fastest result but SVM gave the accurate result most the time.

CHAPTER 2

2.1 GENERAL

Fake news is the content that claims people to believe with the falsification, sometime it is the sensitive messages. When the messages were received, they will rapidly disperse them to others. The dissemination of fake news in today‟s digital world has affected a specific group. Mixing both believable and unbelievable information on social media has made the confusion of truth. That is the truth will be hardly classified. However, the appearance of fake news causes a great threat to the safety of people‟s lives and property. There is misinformation (the distributor believes there are true) or disinformation (the distributor knows it is not a fact but he intentionally hoaxes) in fake news proliferation.

2.2 METHODOLOGIES

2.2.1 MODULE NAMES:

Logistic Regression:

As there are classifying text on the basis of a wide feature set, with a binary output (true/false or true article/fake article), a logistic regression (LR) model is used, since it provides the intuitive equation to classify problems into binary or multiple classes. They performed hyperparameter tuning to get the best result for all individual datasets, while multiple parameters are tested before acquiring the maximum accuracies from the LR model.

Support Vector Machine (SVM):

Support vector machine (SVM) is another model for binary classification problems and is available in various kernels functions. The objective of an SVM model is to estimate a hyperplane (or decision boundary) on the basis of a feature set to classify data points. The dimension of the hyperplane varies according to the number of features. As there could be multiple possibilities for a hyperplane to exist in an N -dimensional space, the task is to identify the plane that separates the data points of two classes with maximum margin 8.

Multilayer Perceptron (MLP):

A multilayer perceptron (MLP) is an artificial neural network, with an input layer, one or more hidden layers, and an output layer. MLP can be as simple as having each of the three layers; however, in our experiments, we have fine-tuned the model with various parameters and a number of layers to generatean optimum predicting model.

K-Nearest Neighbors (KNN):

KNN is an unsupervised machine learning model where a dependent variable is not required to predict the outcome on specific data. They provide enough training data to the model and let it decide to which particular neighborhood a data point belongs. KNN model estimates the distance of a new data point to its nearest neighbors, and the value of K estimates the majority of its neighbors‟ votes; if the value of K is 1, then the new data point is assigned to aclass that has the nearest distance.

Random Forest (RF):

Random forest (RF) is an advanced form of decision trees (DT) which is also a supervised learning model. RF consists of a large number of decision trees working individually to predict an outcome of a class where the final prediction is based on a class that received majority votes.

2.3 TECHNIQUE USED OR ALGORITHM USED

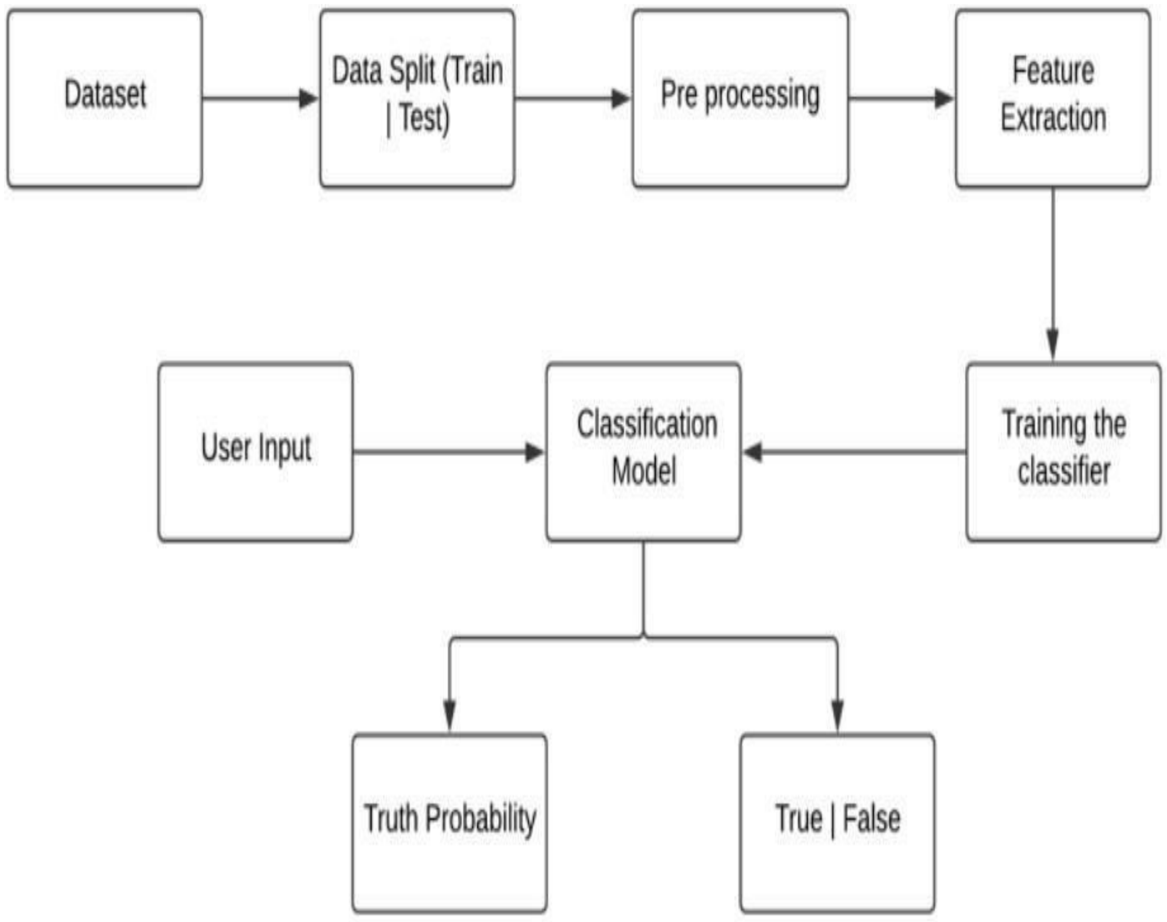
2.3.1 EXISTING TECHNIQUE

A Convolutional Neural Network (CNN) involves the multiplication of matrices that provide outputs to incorporate for the further training process. This method is known as convolution. That is why this type of neural network is called a convolutional neural network. In the case of NLP, words in a sentence or a news article are represented as word vectors. These word vectors are then used for training a CNN. The training is carried out by specifying a kernel size and a number of filters. A CNN can be multi-dimensional. In the case of text classification or NLP, a one-dimensional CNN (Conv1D) is used generally. Conv1D deals with one-dimensional arrays representing word vectors. In a CNN, a filter of fixed size window iterates through the training data, which at each step multiplies the input with the filter weights 9 and gives an output that is stored in an output array. This output array is a feature map or output filter of the data. In this way, a feature is detected from the input training data.

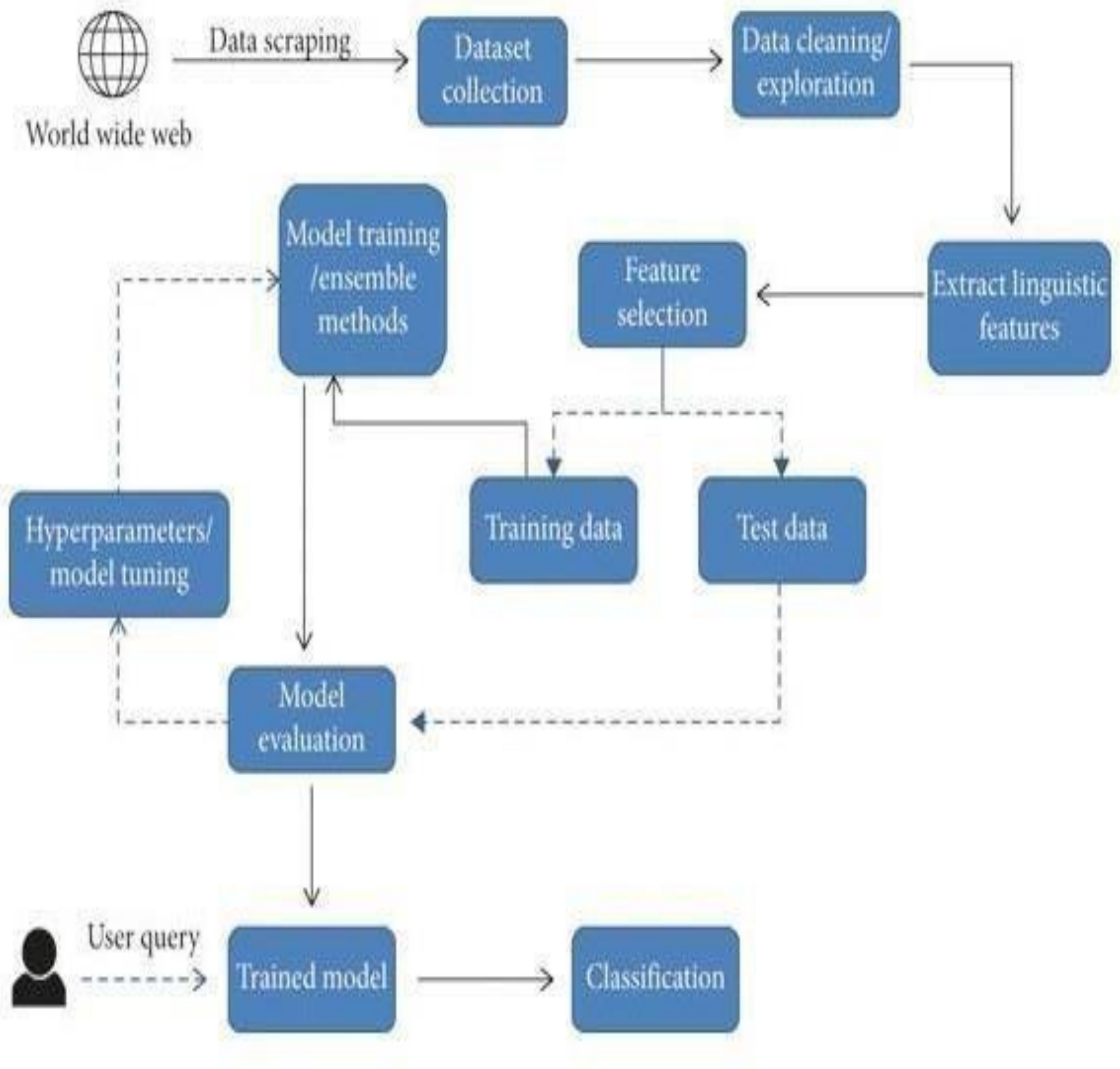
2.3.2 PROPOSED TECHNIQUE

This research uses five methods to classify the believable and unbelievable message from Twitter, there are Logistic Regression, Multi Layer Perceptron, K Nearest Neighbors, Random Forest, and Support Vector Machine (SVM). Logistic Regression is a well-known classification method. It is been defined that the collected tweet data T and class of data (Cx) which x is believable and unbelievable. The probability of tweet data T in class Cx can calculate. A neural network is a mathematical model for the information computation process with connectionism and parallel distributed processing. This concept comes from the bioelectric network simulation in the neural system. Support Vector Machine (SVM) is the classification method of supervised learning. There uses the hyperplane to splits two data class points with the maximum margin. There are four evaluation results are precision, recall, F-measure, and accuracy which computation from True Positive, True Negative, False Positive, False Negative. True Positive is the number of messages that are correctly classified by believable messages. True Negative is the number of messages that are correctly classified by unbelievable messages. False Positive is the number of messages that are incorrectly classified by believable messages. False Negative is the number of messages that are incorrectly classified by unbelievable messages.

2.4 SYSTEM ARCHITECTURE



2.5 IDEATION MAP



CHAPTER 3

REQUIREMENT ANALYSIS

3.1 GENERAL

Developers all around the world have been working on various technologies and are building various tools for software development. Using the best tools available helpsin building and understanding the project well.

3.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should be what the system does and not how it should be implemented.

●	Processor	-	i3/i5/i7
●	Speed	-	1.1 GHz
●	Ram	-	4 GB
●	Hard Disk	-	20 GB

3.3 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team‟s progress throughout the development activity.

●	Operating System	-	Windows 7
●	Coding Language	-	Python
●	IDE	-	Jupyter Notebook 11

3.4 FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of classifying the data items or the news as fake or not fake and predicting the result with most accuracy.

3.5 NON-FUNCTIONAL REQUIREMENTS EFFICIENCY

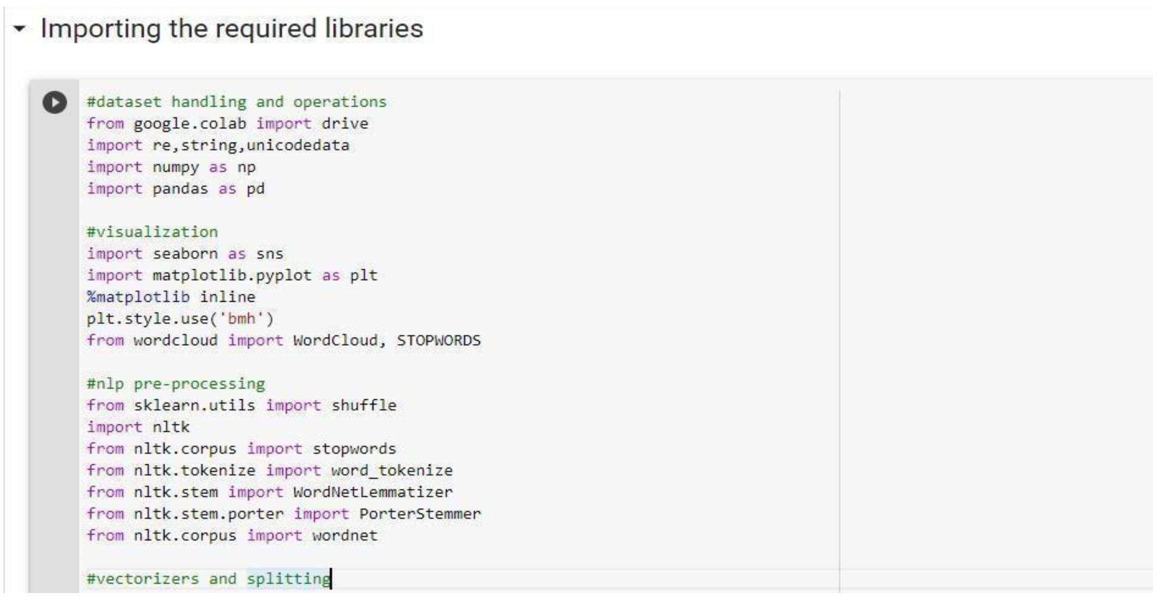
The code we written in the jupyter notebook must be processed with in few seconds. The Logistic Regression is compiled with in very less time where as the SVM and MLP took majority of the processing time. It is required to increase the efficiency of the system by training the data with less data items. With the increase in the number of data items the efficiency of the system is decreased. So to have accurate results with high efficiency it is required to follow the above precautions.

3.6 SIZE

The size of the jupyter notebook is small where- as the csv files which are used in thisproject are comparatively bigger.

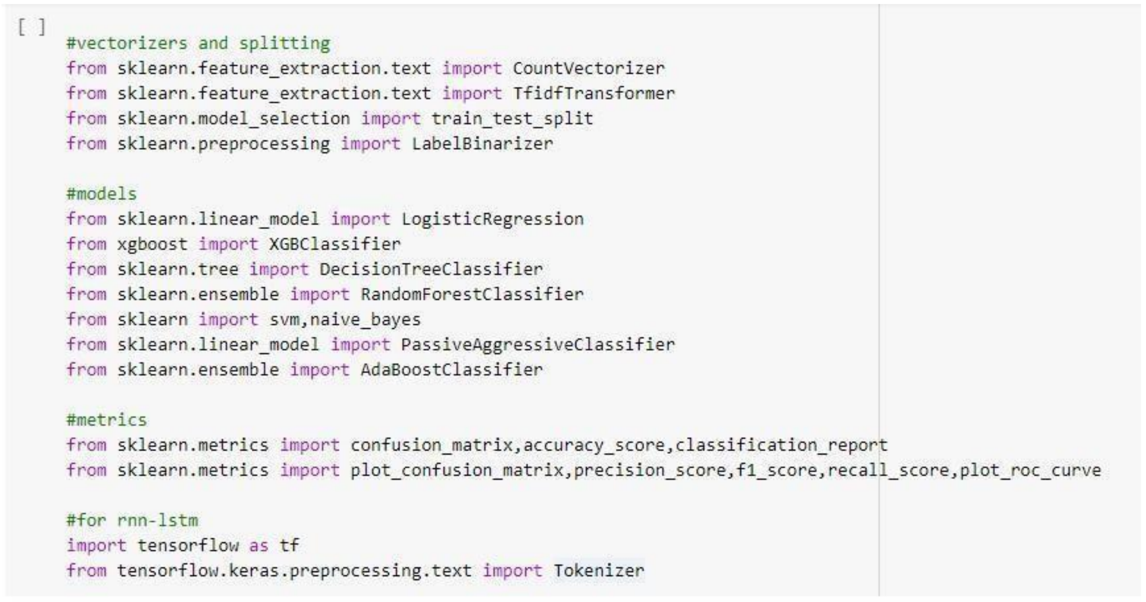
CHAPTER 4

4.1 IMPLEMENTATION

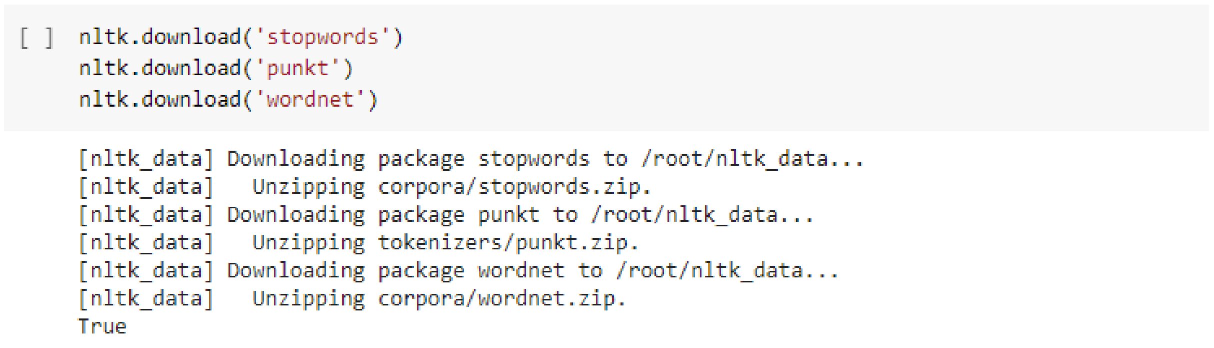


Importing the required libraries

```
#dataset handling and operations
from google.colab import drive
import re,string,unicodedata
import numpy as np
import pandas as pd

#visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('bmh')
from wordcloud import WordCloud, STOPWORDS

#nlp pre-processing
from sklearn.utils import shuffle
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
from nltk.corpus import wordnet

#vectorizers and splitting
```

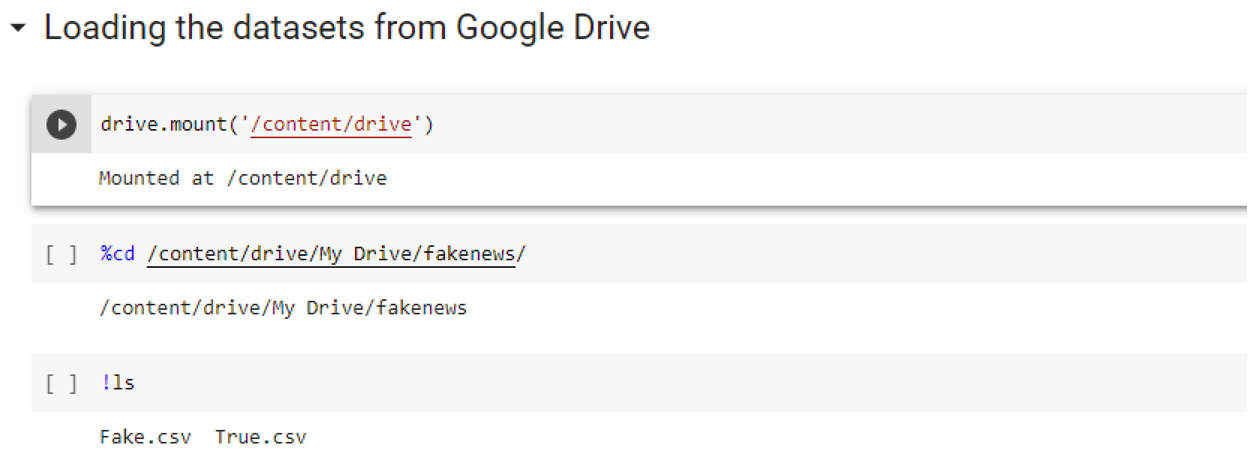


```
[ ]
#vectorizers and splitting
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer

#models
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm,naive_bayes
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.ensemble import AdaBoostClassifier

#metrics
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
from sklearn.metrics import plot_confusion_matrix,precision_score,f1_score,recall_score,plot_roc_curve

#for rnn-lstm
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
```



```
[ ] nltk.download('stopwords')
    nltk.download('punkt')
    nltk.download('wordnet')

    [nltk_data] Downloading package stopwords to /root/nltk_data...
    [nltk_data]   Unzipping corpora/stopwords.zip.
    [nltk_data] Downloading package punkt to /root/nltk_data...
    [nltk_data]   Unzipping tokenizers/punkt.zip.
    [nltk_data] Downloading package wordnet to /root/nltk_data...
    [nltk_data]   Unzipping corpora/wordnet.zip.
    True
```

Our dataset is uploaded on Google Drive. To read the dataset, we mount our drive by authorizing the google account and then movingto the folder where our dataset is present using „%cd‟. We check forthe folder content using „!ls.



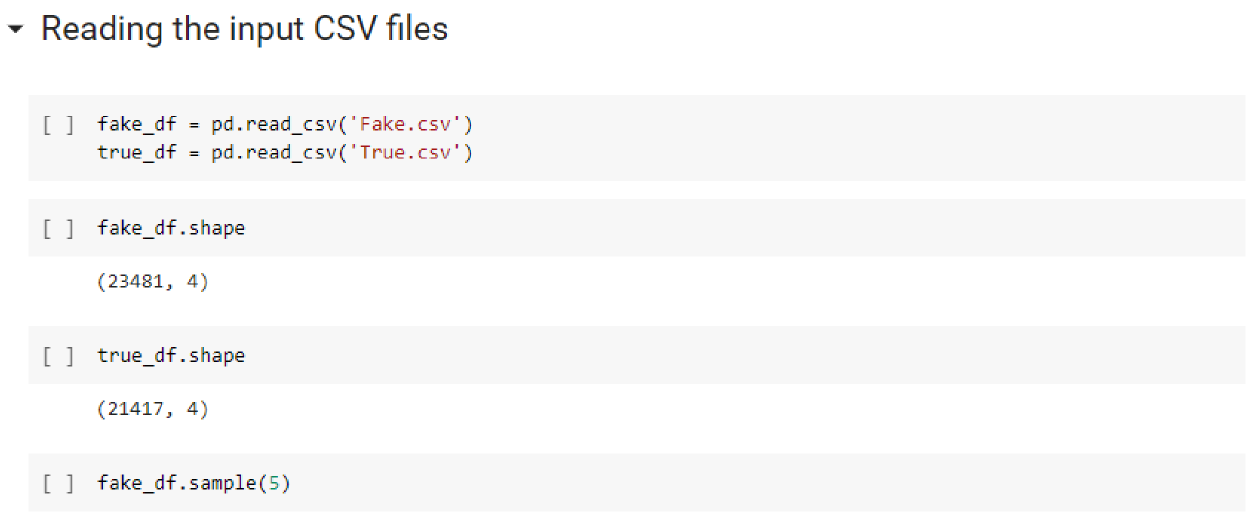
Loading the datasets from Google Drive

```
drive.mount('/content/drive')

Mounted at /content/drive
```

```
[ ] %cd /content/drive/My Drive/fakenews/

/content/drive/My Drive/fakenews
```

```
[ ] !ls

Fake.csv   True.csv
```

Now that our drive has been mounted, we read the dataset. We havetwo different datasets for Fake and True news. The fake news database consists of 23481 items and the true news dataset contains21417 items. We display a few records using the sample() function.



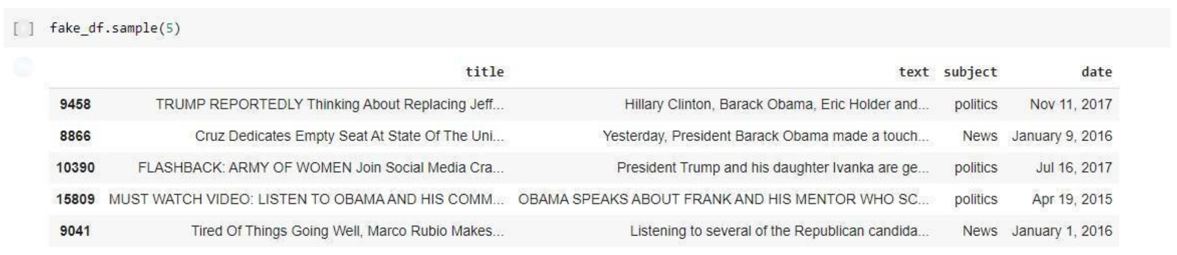
Reading the input CSV files

```
[ ] fake_df = pd.read_csv('Fake.csv')
    true_df = pd.read_csv('True.csv')
```

```
[ ] fake_df.shape

(23481, 4)
```

```
[ ] true_df.shape

(21417, 4)
```

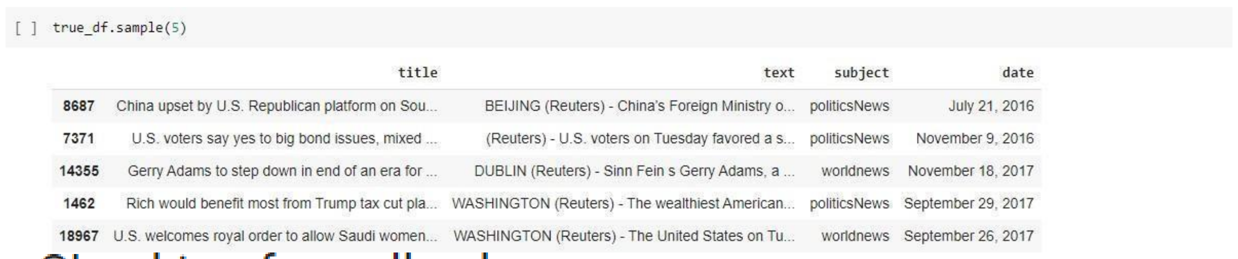
```
[ ] fake_df.sample(5)
```



```
[ ] fake_df.sample(5)
```

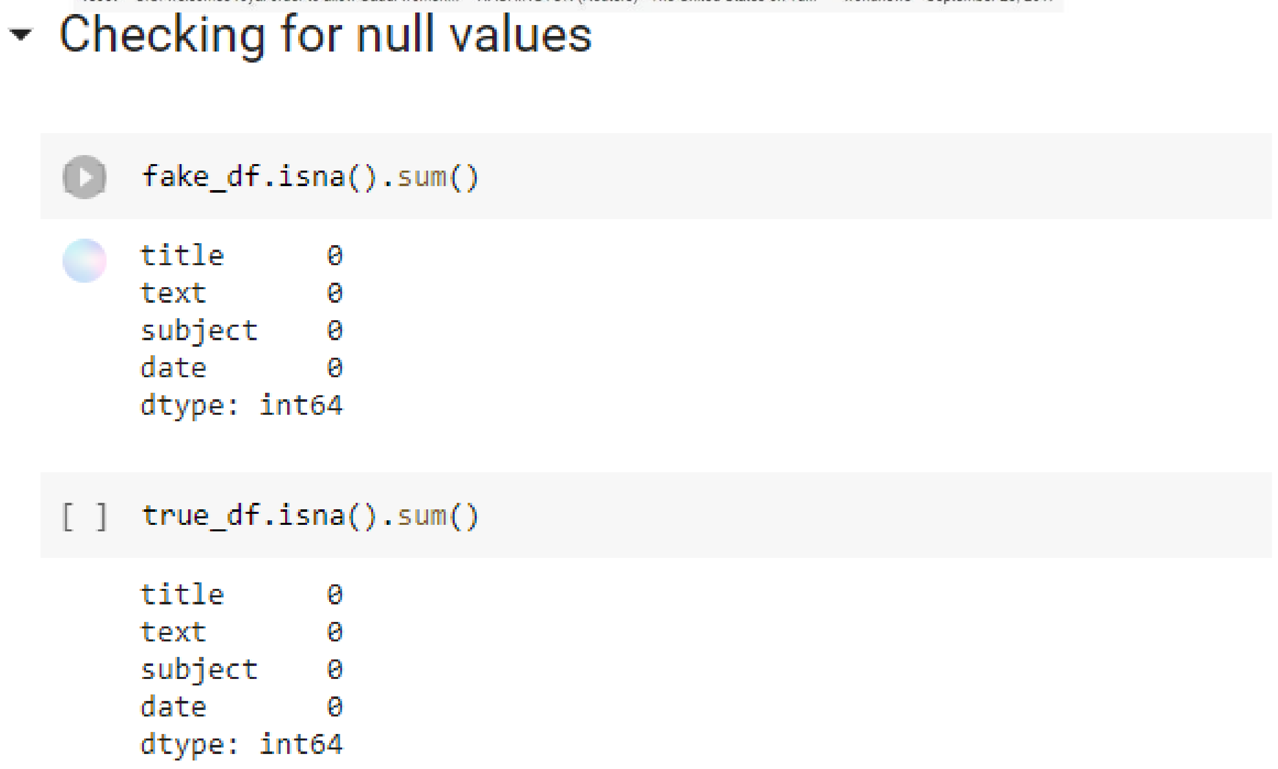
	title	text	subject	date
9458	TRUMP REPORTEDLY Thinking About Replacing Jeff...	Hillary Clinton, Barack Obama, Eric Holder and...	politics	Nov 11, 2017
8866	Cruz Dedicates Empty Seat At State Of The Uni...	Yesterday, President Barack Obama made a touch...	News	January 9, 2016
10390	FLASHBACK: ARMY OF WOMEN Join Social Media Cra...	President Trump and his daughter Ivanka are ge...	politics	Jul 16, 2017
15809	MUST WATCH VIDEO: LISTEN TO OBAMA AND HIS COMM...	OBAMA SPEAKS ABOUT FRANK AND HIS MENTOR WHO SC...	politics	Apr 19, 2015
9041	Tired Of Things Going Well, Marco Rubio Makes...	Listening to several of the Republican candida...	News	January 1, 2016

We check for null values in both datasets.



```
[ ] true_df.sample(5)
```

	title	text	subject	date
8687	China upset by U.S. Republican platform on Sou...	BEIJING (Reuters) - China's Foreign Ministry o...	politicsNews	July 21, 2016
7371	U.S. voters say yes to big bond issues, mixed ...	(Reuters) - U.S. voters on Tuesday favored a s...	politicsNews	November 9, 2016
14355	Gerry Adams to step down in end of an era for ...	DUBLIN (Reuters) - Sinn Fein s Gerry Adams, a ...	worldnews	November 18, 2017
1462	Rich would benefit most from Trump tax cut pla...	WASHINGTON (Reuters) - The wealthiest American...	politicsNews	September 29, 2017
18967	U.S. welcomes royal order to allow Saudi women...	WASHINGTON (Reuters) - The United States on Tu...	worldnews	September 26, 2017



Checking for null values

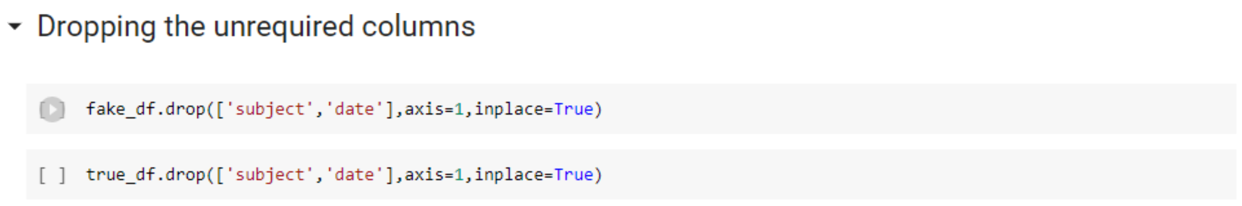
```
fake_df.isna().sum()
```

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

```
[ ] true_df.isna().sum()
```

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

We drop the columns that aren"t relevant for fake news detection. These columns have no effect on determining whether the news isfake or true.

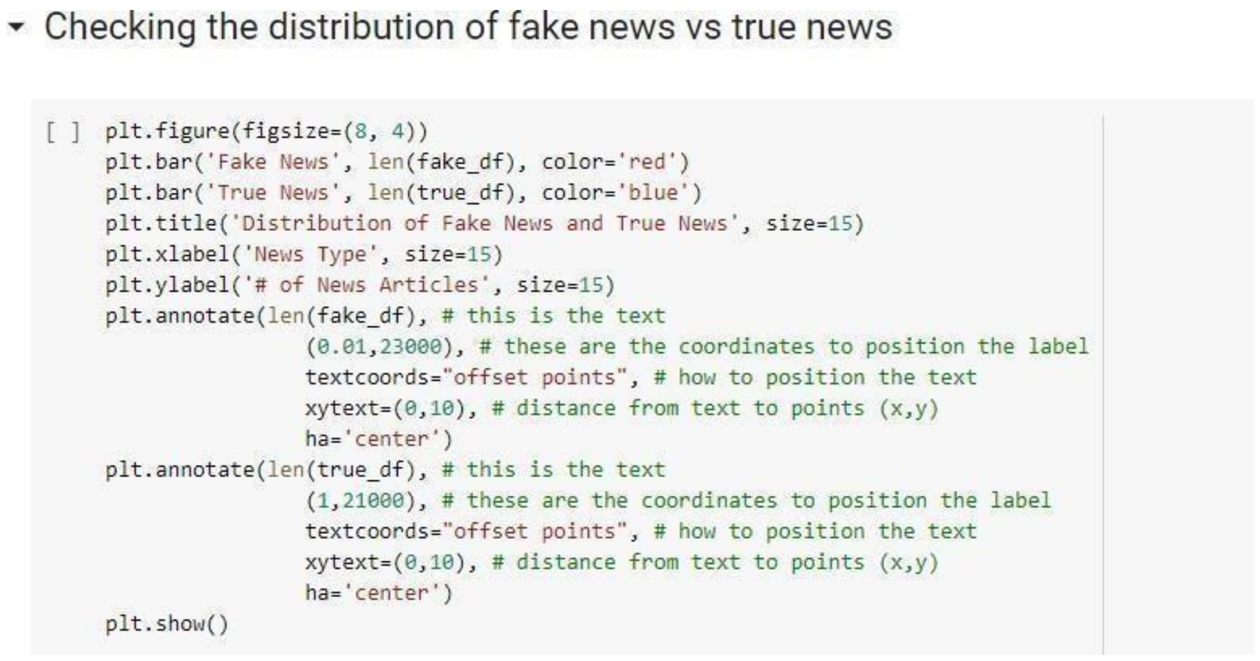


Dropping the unrequired columns

```
fake_df.drop(['subject','date'],axis=1,inplace=True)
```

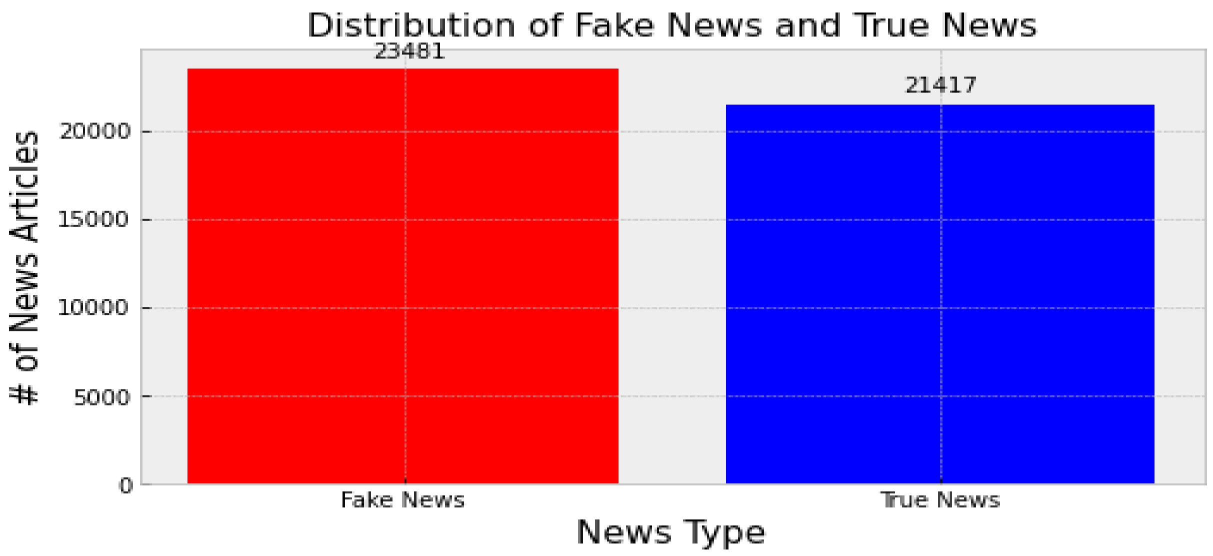
```
[ ] true_df.drop(['subject','date'],axis=1,inplace=True)
```

Then we check the distribution of fake news and true news by plottinga bar plot. We see that the bar for fake news is higher since it has more records than true news, but the distribution is perfect to train ourmodel as both the datasets have records in the range of 21-23k.

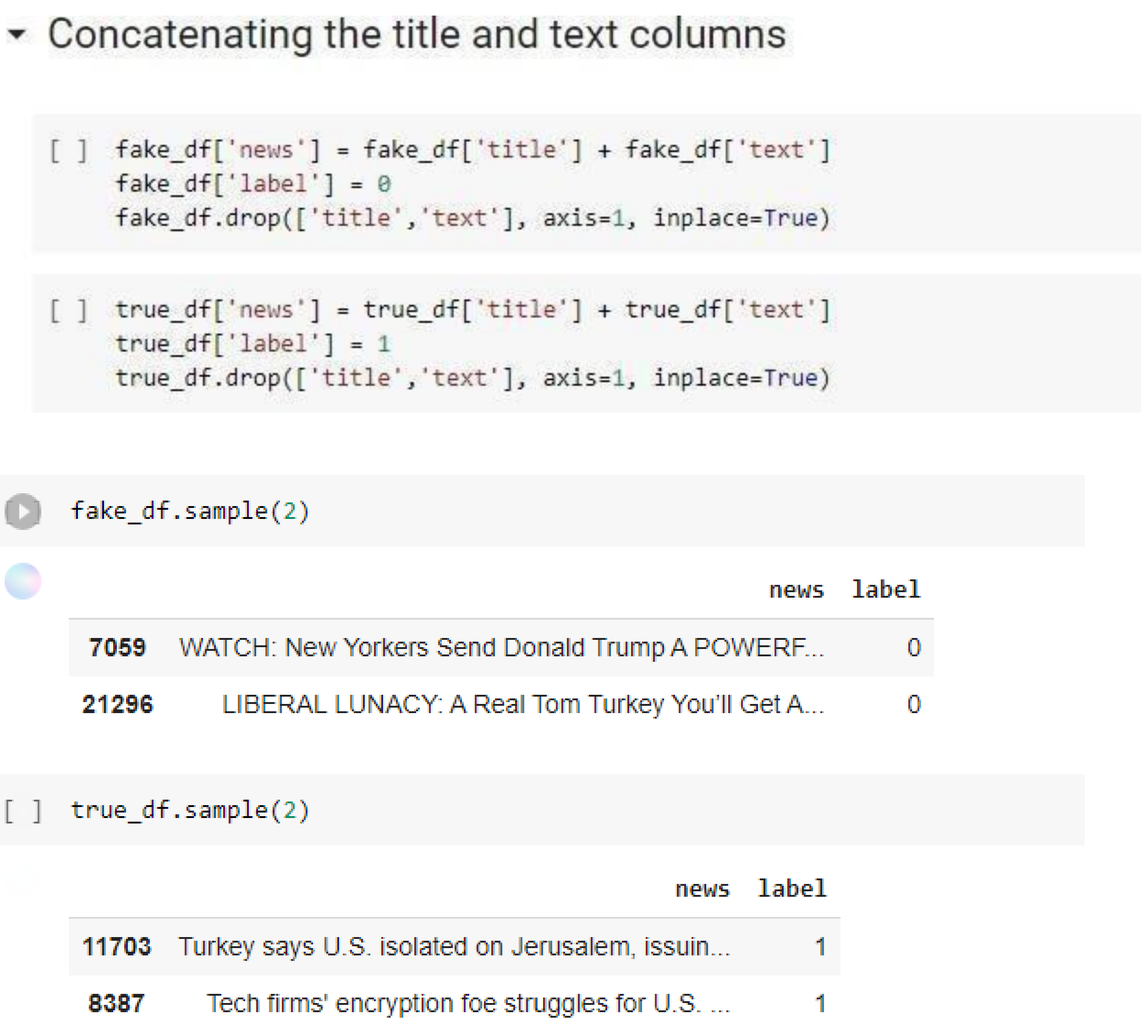


▾ Checking the distribution of fake news vs true news

```
[ ]  plt.figure(figsize=(8, 4))
     plt.bar('Fake News', len(fake_df), color='red')
     plt.bar('True News', len(true_df), color='blue')
     plt.title('Distribution of Fake News and True News', size=15)
     plt.xlabel('News Type', size=15)
     plt.ylabel('# of News Articles', size=15)
     plt.annotate(len(fake_df), # this is the text
                  (0.01,23000), # these are the coordinates to position the label
                  textcoords="offset points", # how to position the text
                  xytext=(0,10), # distance from text to points (x,y)
                  ha='center')
     plt.annotate(len(true_df), # this is the text
                  (1,21000), # these are the coordinates to position the label
                  textcoords="offset points", # how to position the text
                  xytext=(0,10), # distance from text to points (x,y)
                  ha='center')
     plt.show()
```



To simplify the data, we concatenate the title and text columns into news column in both the datasets and drop the former two. We createa new column „label‟ which has values of 0-fake and 1-true.



Concatenating the title and text columns

```
[ ] fake_df['news'] = fake_df['title'] + fake_df['text']
    fake_df['label'] = 0
    fake_df.drop(['title','text'], axis=1, inplace=True)
```

```
[ ] true_df['news'] = true_df['title'] + true_df['text']
    true_df['label'] = 1
    true_df.drop(['title','text'], axis=1, inplace=True)
```

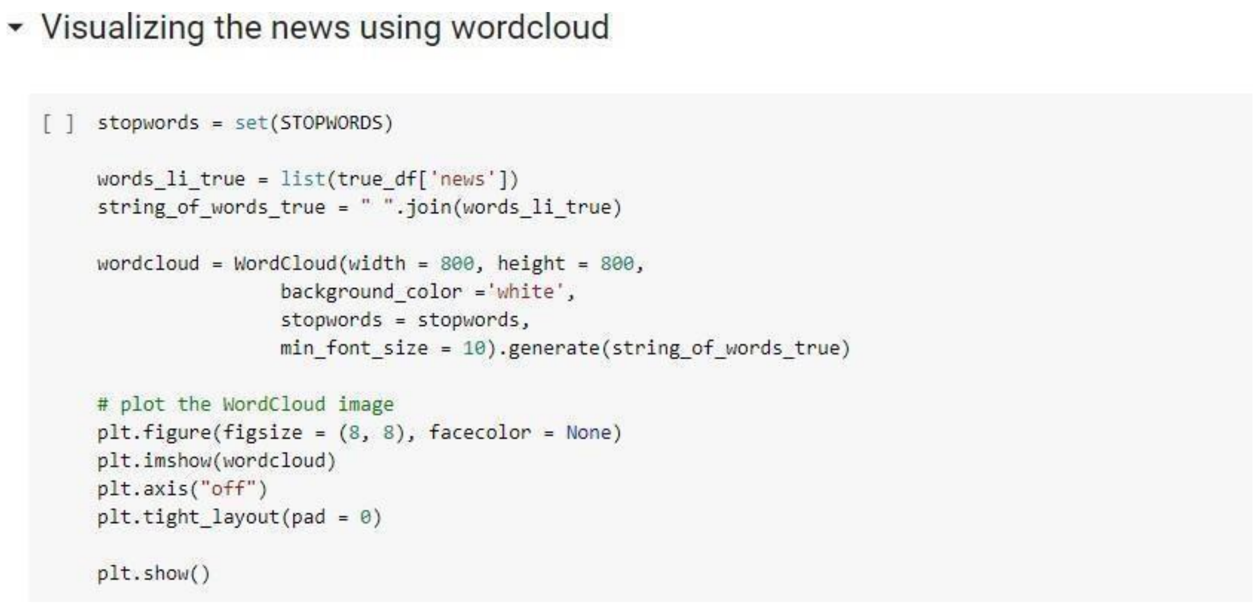
```
fake_df.sample(2)
```

	news	label
7059	WATCH: New Yorkers Send Donald Trump A POWERF...	0
21296	LIBERAL LUNACY: A Real Tom Turkey You'll Get A...	0

```
[ ] true_df.sample(2)
```

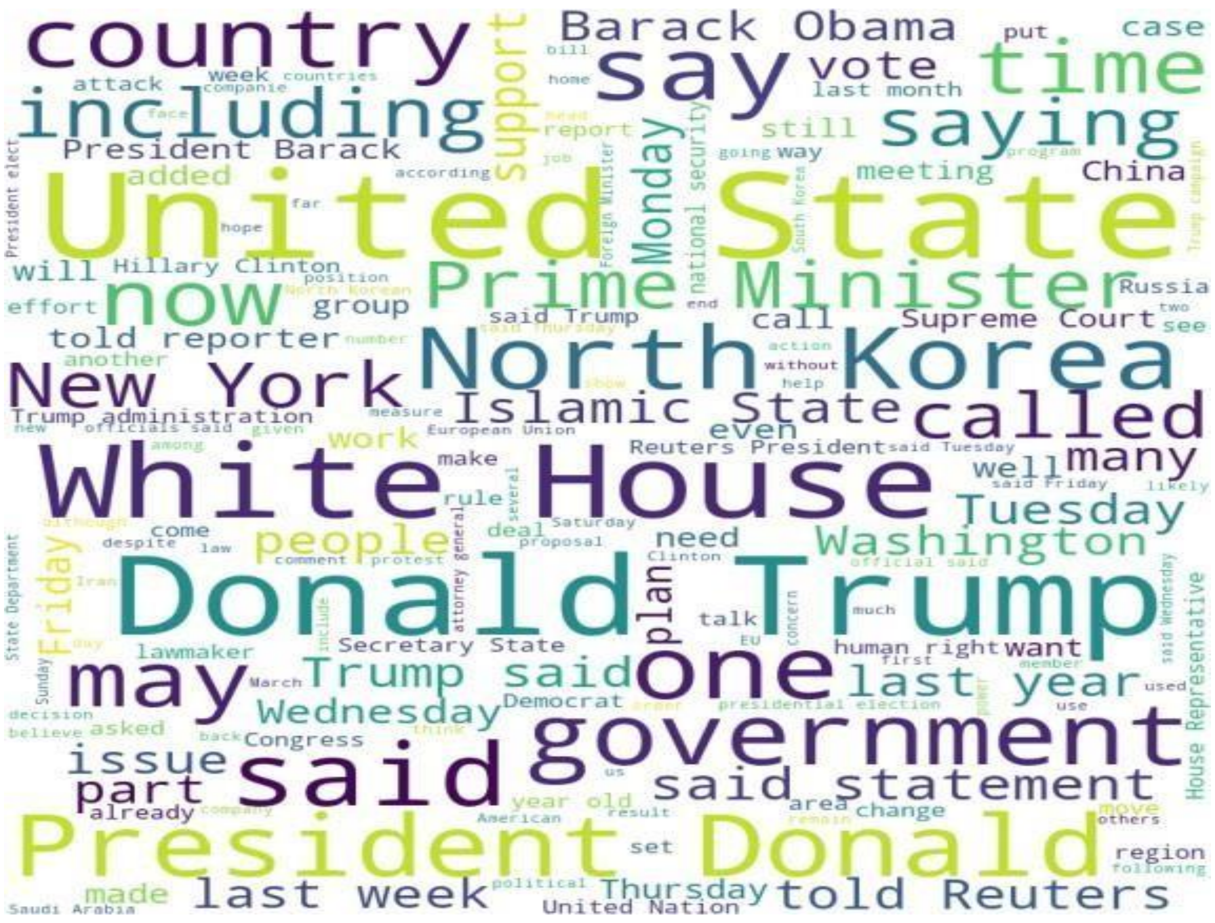
	news	label
11703	Turkey says U.S. isolated on Jerusalem, issuin...	1
8387	Tech firms' encryption foe struggles for U.S. ...	1

We visualize the news present in the True news dataset by creating a Wordcloud to highlight the popular words and phrases based on frequency and relevance.

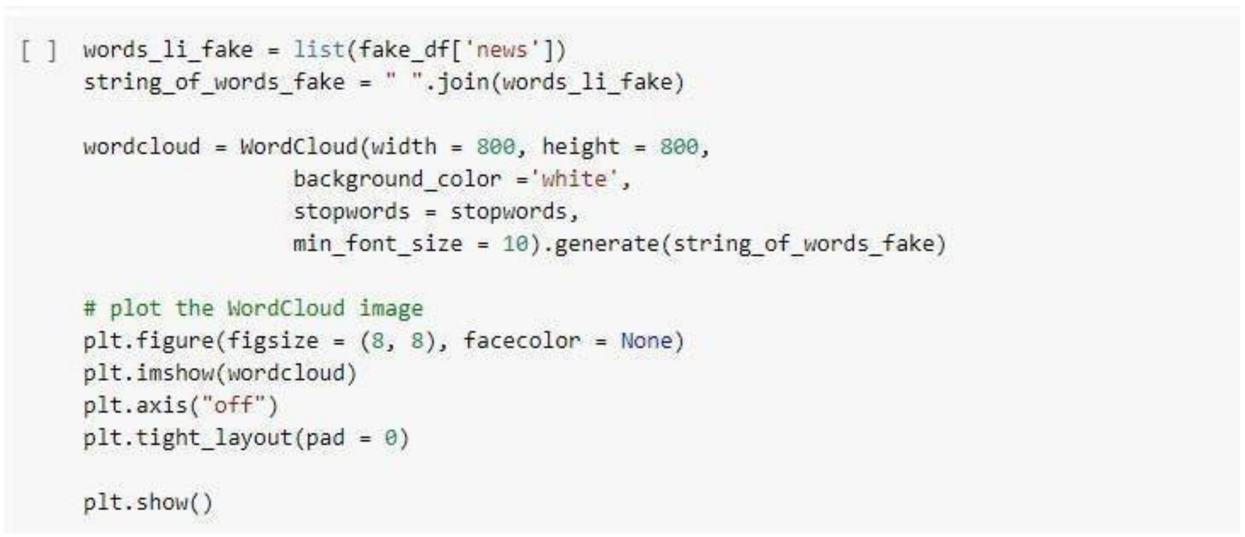


Visualizing the news using wordcloud

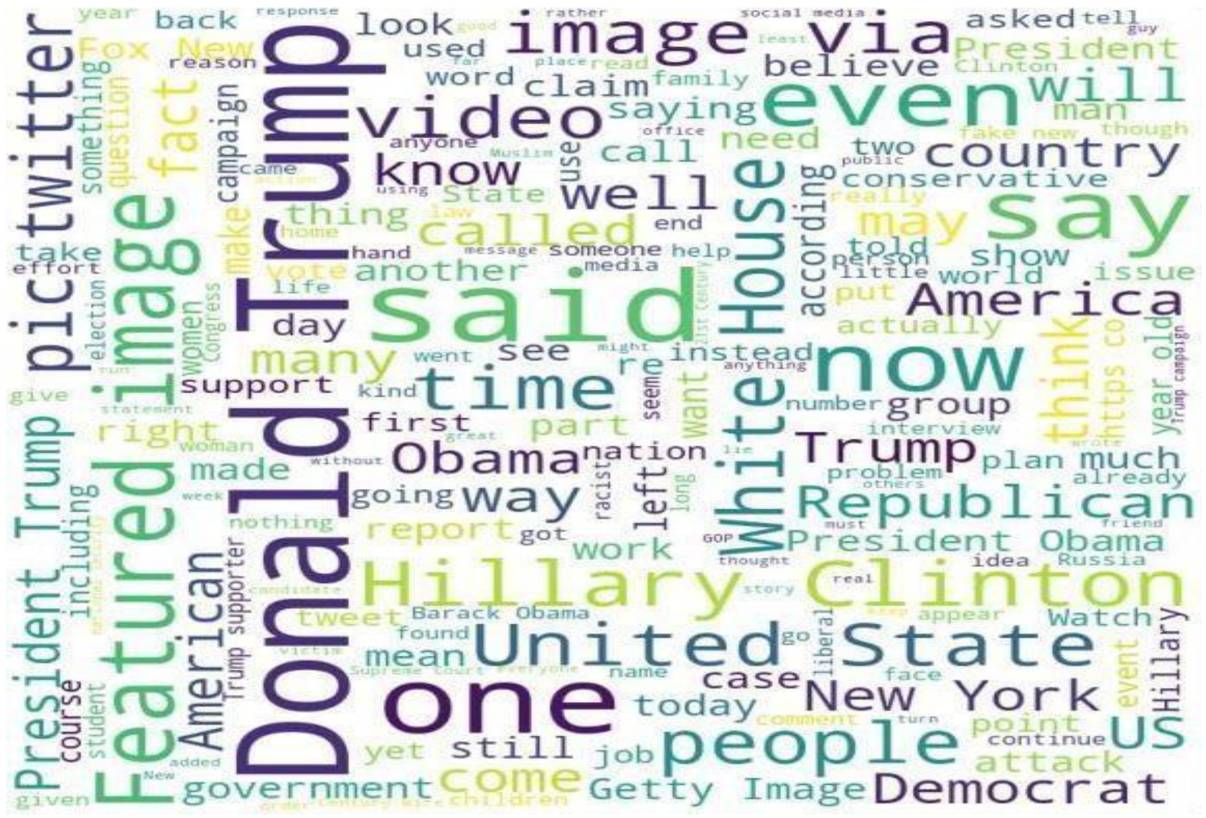
```
[ ]  stopwords = set(STOPWORDS)

     words_li_true = list(true_df['news'])
     string_of_words_true = " ".join(words_li_true)

     wordcloud = WordCloud(width = 800, height = 800,
                     background_color ='white',
                     stopwords = stopwords,
                     min_font_size = 10).generate(string_of_words_true)

     # plot the WordCloud image
     plt.figure(figsize = (8, 8), facecolor = None)
     plt.imshow(wordcloud)
     plt.axis("off")
     plt.tight_layout(pad = 0)

     plt.show()
```



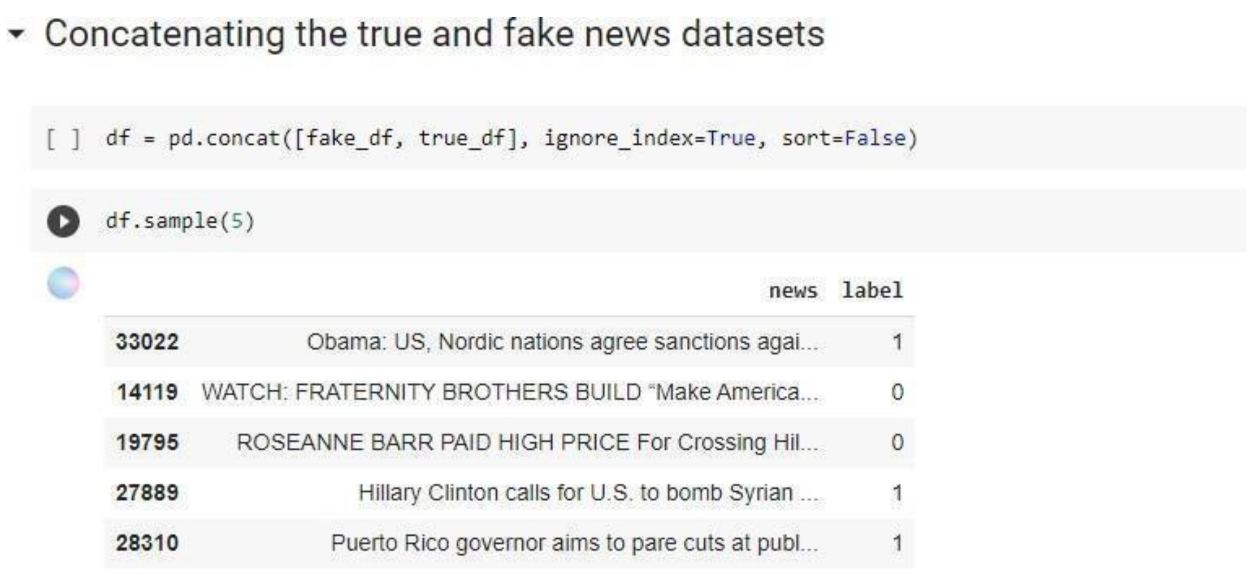
Similarly, to visualize the text or news present in fake news dataset,we create another word cloud.



```
[ ]  words_li_fake = list(fake_df['news'])
     string_of_words_fake = " ".join(words_li_fake)

     wordcloud = WordCloud(width = 800, height = 800,
                     background_color ='white',
                     stopwords = stopwords,
                     min_font_size = 10).generate(string_of_words_fake)

     # plot the WordCloud image
     plt.figure(figsize = (8, 8), facecolor = None)
     plt.imshow(wordcloud)
     plt.axis("off")
     plt.tight_layout(pad = 0)

     plt.show()
```



In order to build, train and test our models, we need to concatenate the datasets into a single dataset. We concatenate thedatasets in the new dataset.



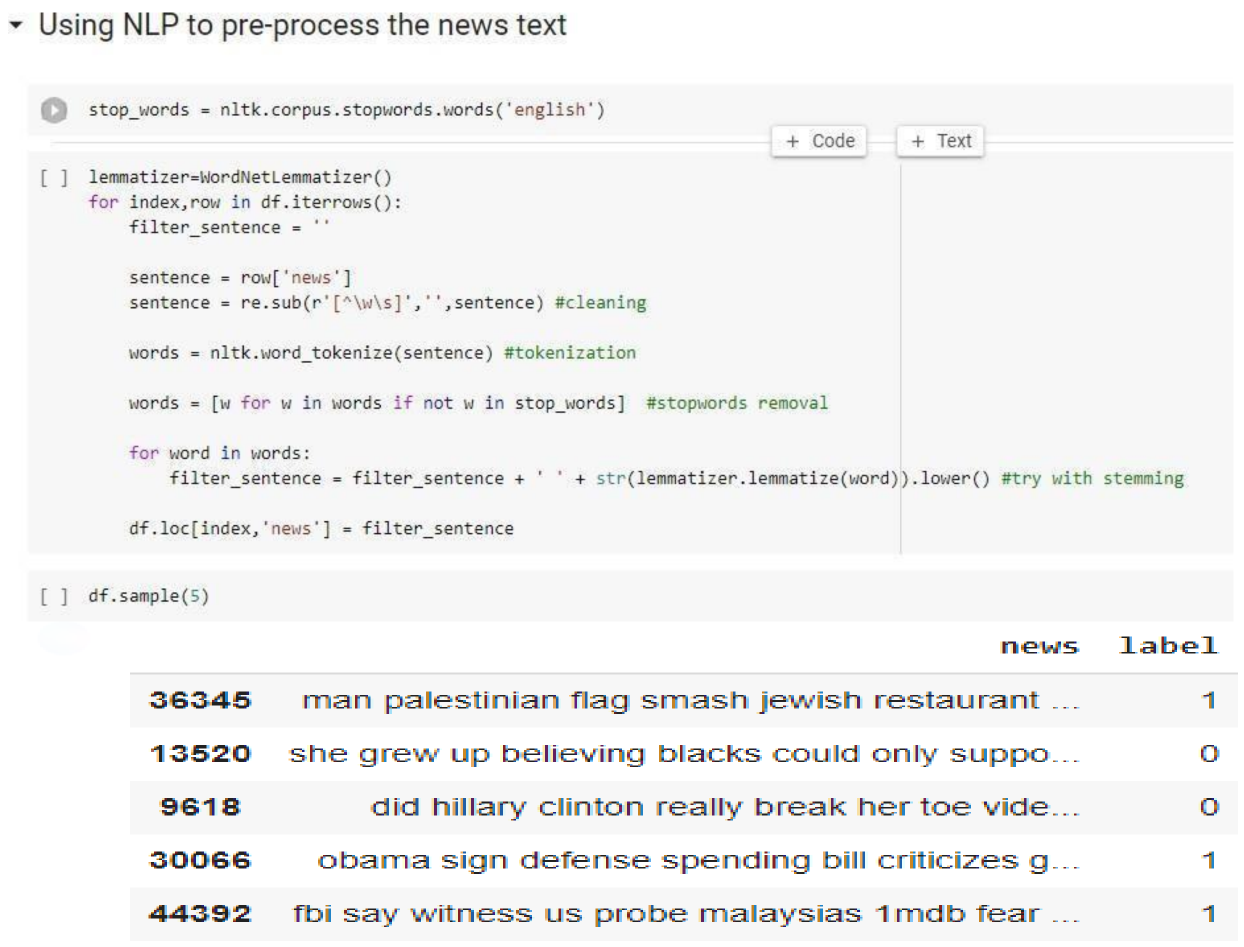
Concatenating the true and fake news datasets

```
[ ]  df = pd.concat([fake_df, true_df], ignore_index=True, sort=False)
```

```
df.sample(5)
```

	news	label
33022	Obama: US, Nordic nations agree sanctions agai...	1
14119	WATCH: FRATERNITY BROTHERS BUILD "Make America...	0
19795	ROSEANNE BARR PAID HIGH PRICE For Crossing Hil...	0
27889	Hillary Clinton calls for U.S. to bomb Syrian ...	1
28310	Puerto Rico governor aims to pare cuts at publ...	1

In order to preprocess the news text, we use NLP, where we first tokenize the text and then remove stop words, after which we lemmatize the text.



Using NLP to pre-process the news text

```
stop_words = nltk.corpus.stopwords.words('english')
```

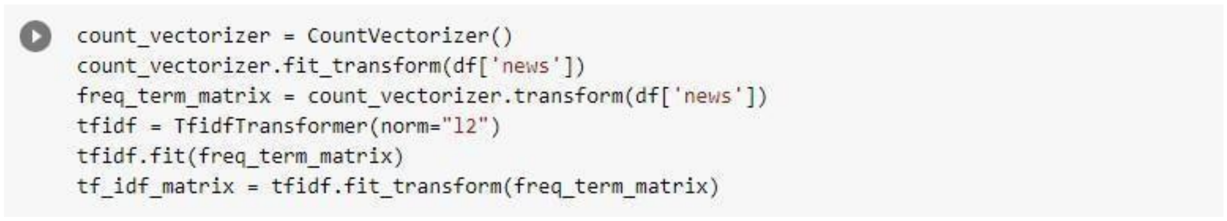
```
[ ] lemmatizer=WordNetLemmatizer()
    for index,row in df.iterrows():
        filter_sentence = ''

        sentence = row['news']
        sentence = re.sub(r'[^\w\s]','',sentence) #cleaning

        words = nltk.word_tokenize(sentence) #tokenization

        words = [w for w in words if not w in stop_words]  #stopwords removal

        for word in words:
            filter_sentence = filter_sentence + ' ' + str(lemmatizer.lemmatize(word)).lower() #try with stemming

        df.loc[index,'news'] = filter_sentence
```

```
[ ] df.sample(5)
```

	news	label
36345	man palestinian flag smash jewish restaurant ...	1
13520	she grew up believing blacks could only suppo...	0
9618	did hillary clinton really break her toe vide...	0
30066	obama sign defense spending bill criticizes g...	1
44392	fbi say witness us probe malaysias 1mdb fear ...	1

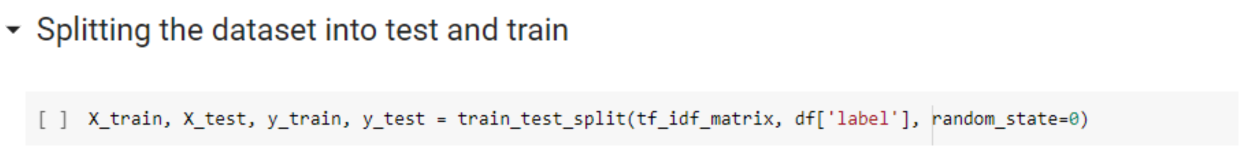
After that we use count vectorizer to transform our data from a collection of text documents to a matrix of token counts.

We then use tf-idf vectorizer to convert the collection of raw documents to a matrix of TF-IDF features.



```
count_vectorizer = CountVectorizer()
count_vectorizer.fit_transform(df['news'])
freq_term_matrix = count_vectorizer.transform(df['news'])
tfidf = TfidfTransformer(norm="l2")
tfidf.fit(freq_term_matrix)
tf_idf_matrix = tfidf.fit_transform(freq_term_matrix)
```

Now that we have pre-processed our data, we split the data intotesting and training data using train_test_split.

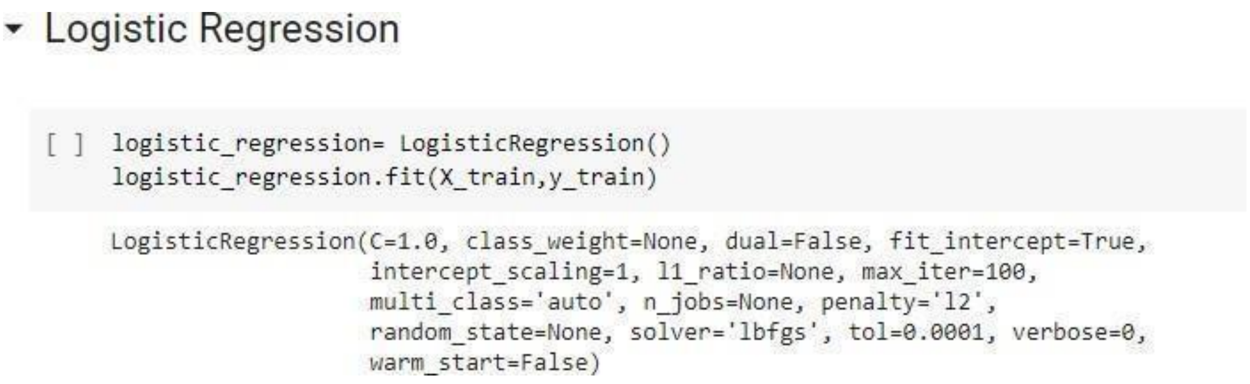


Splitting the dataset into test and train

```
[ ] X_train, X_test, y_train, y_test = train_test_split(tf_idf_matrix, df['label'], random_state=0)
```

Logistic Regression

We create our logistic regression model and fit our training data to it.

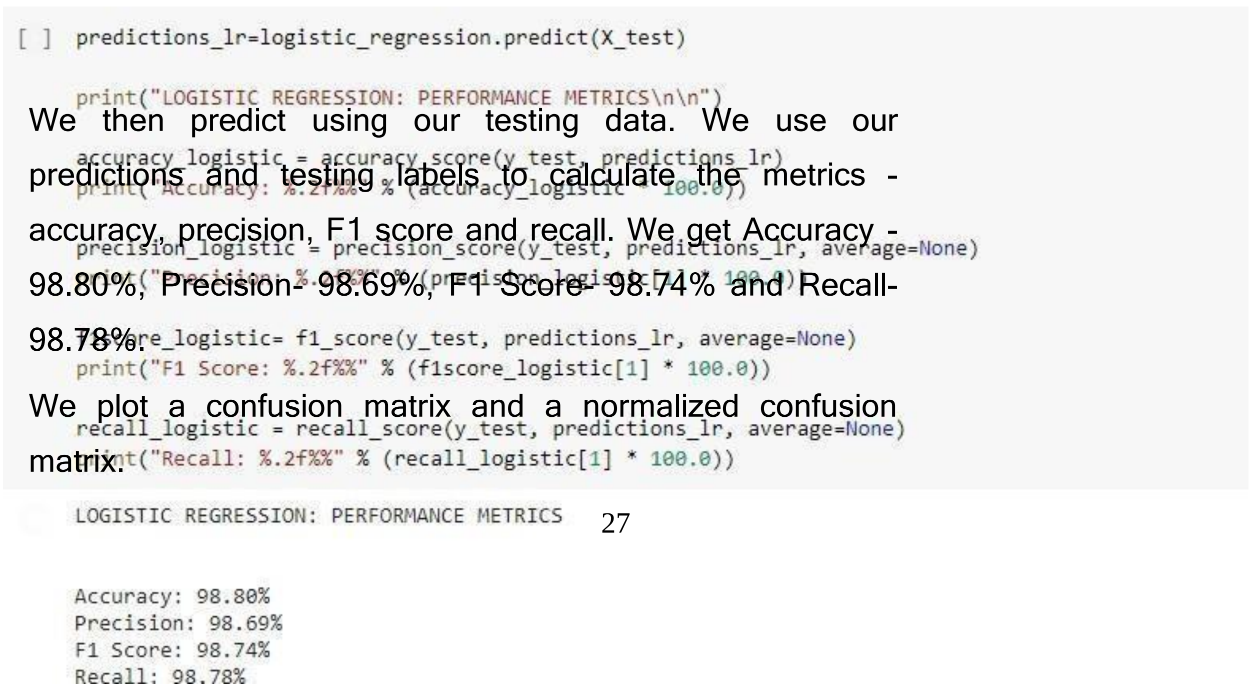


Logistic Regression

```
[ ] logistic_regression= LogisticRegression()
    logistic_regression.fit(X_train,y_train)

    LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                       intercept_scaling=1, l1_ratio=None, max_iter=100,
                       multi_class='auto', n_jobs=None, penalty='l2',
                       random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                       warm_start=False)
```

We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 score and recall. We get Accuracy - 98.80%, Precision- 98.69%, F1 Score- 98.74% and Recall- 98.78%.

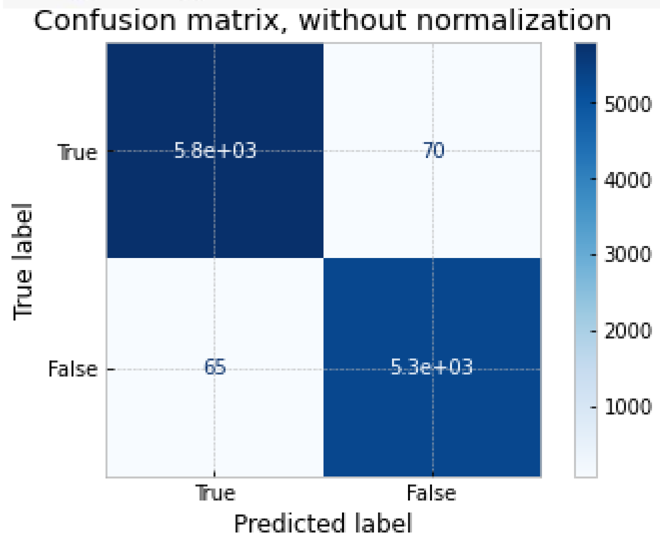
We plot a confusion matrix and a normalized confusion matrix.

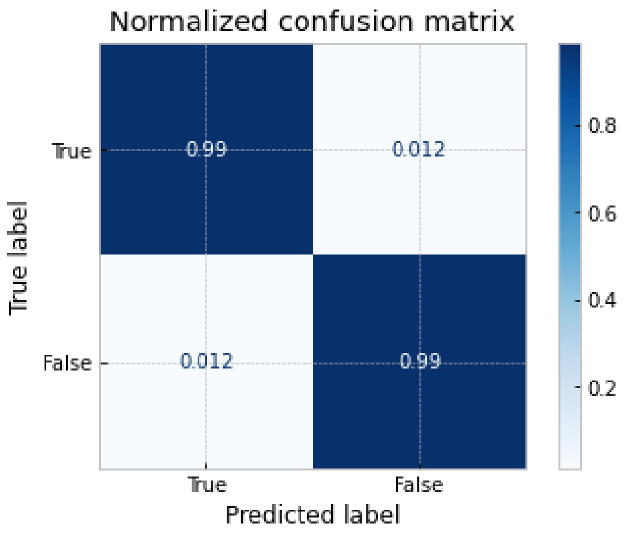


```
[ ] predictions_lr=logistic_regression.predict(X_test)

    print("LOGISTIC REGRESSION: PERFORMANCE METRICS\n\n")

    accuracy_logistic = accuracy_score(y_test, predictions_lr)
    print("Accuracy: %.2f%%" % (accuracy_logistic * 100.0))

    precision_logistic = precision_score(y_test, predictions_lr, average=None)
    print("Precision: %.2f%%" % (precision_logistic[1] * 100.0))

    f1score_logistic= f1_score(y_test, predictions_lr, average=None)
    print("F1 Score: %.2f%%" % (f1score_logistic[1] * 100.0))

    recall_logistic = recall_score(y_test, predictions_lr, average=None)
    print("Recall: %.2f%%" % (recall_logistic[1] * 100.0))

    LOGISTIC REGRESSION: PERFORMANCE METRICS


    Accuracy: 98.80%
    Precision: 98.69%
    F1 Score: 98.74%
    Recall: 98.78%
```

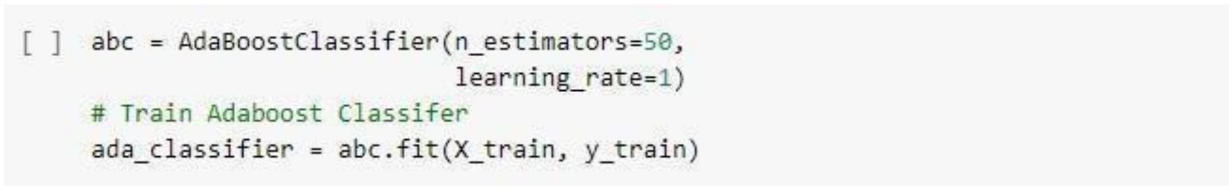
```
Confusion matrix, without normalization
[[5806   70]
 [  65 5284]]

Normalized confusion matrix
[[0.99 0.01]
 [0.01 0.99]]
```

```
class_names = ['True', 'False']

np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
titles_options = [("Confusion matrix, without normalization", None),
                  ("Normalized confusion matrix", 'true')]
for title, normalize in titles_options:
    disp = plot_confusion_matrix(logistic_regression, X_test, y_test,
                                 display_labels=class_names,
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
    disp.ax_.set_title(title)

    print()
    print(title)
    print(disp.confusion_matrix)

plt.show()
```





We create our AdaBoost Classifier model and fit our

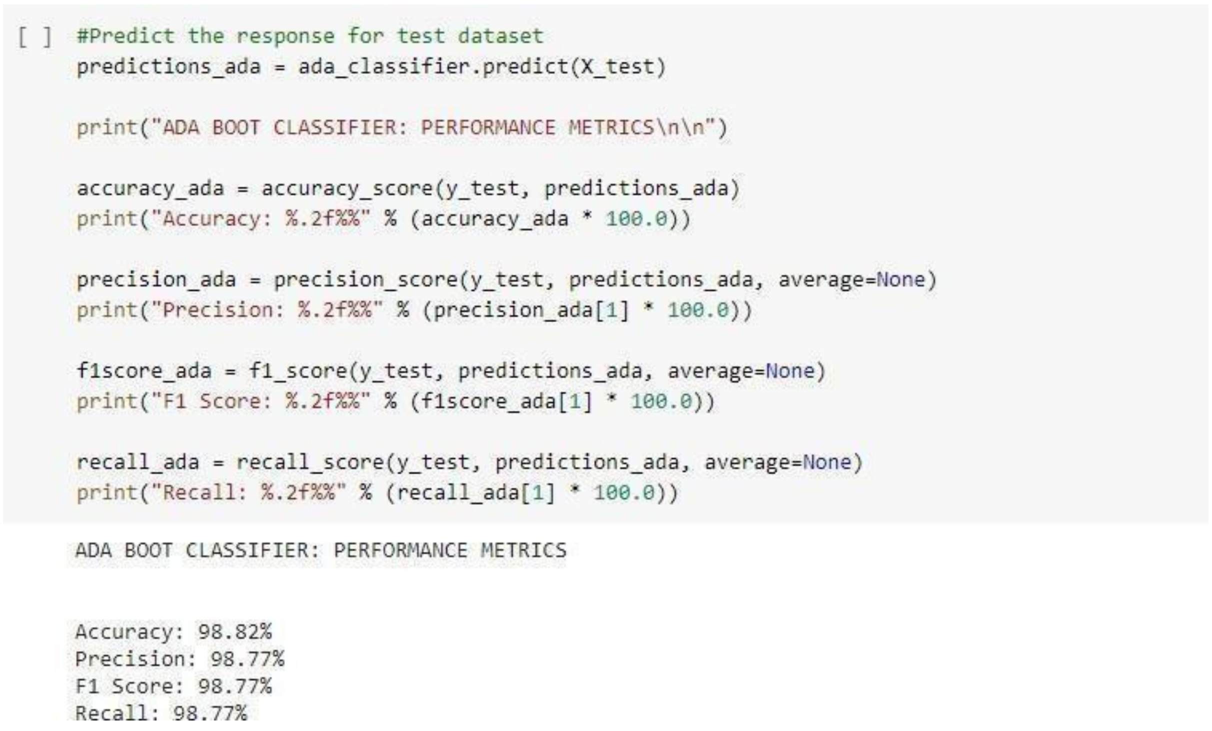


```
[ ]  abc = AdaBoostClassifier(n_estimators=50,
                              learning_rate=1)
     # Train Adaboost Classifer
     ada_classifier = abc.fit(X_train, y_train)
```

training data to it.

We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 scoreand recall. We get Accuracy - 98.82%, Precision- 98.77%, F1 Score-98.77% and Recall- 98.77%.

We plot a confusion matrix and a normalized confusion matrix.

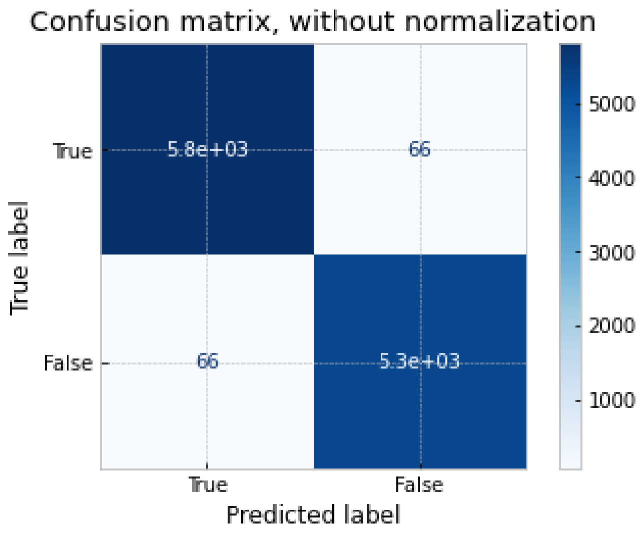


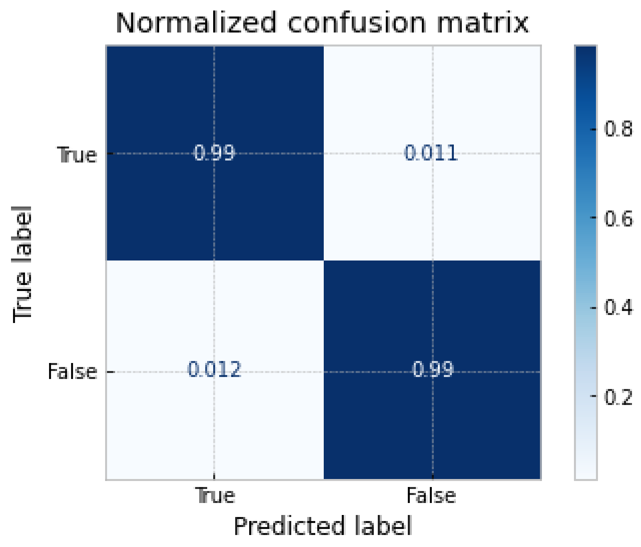
```
[ ]  #Predict the response for test dataset
     predictions_ada = ada_classifier.predict(X_test)

     print("ADA BOOT CLASSIFIER: PERFORMANCE METRICS\n\n")

     accuracy_ada = accuracy_score(y_test, predictions_ada)
     print("Accuracy: %.2f%%" % (accuracy_ada * 100.0))

     precision_ada = precision_score(y_test, predictions_ada, average=None)
     print("Precision: %.2f%%" % (precision_ada[1] * 100.0))

     f1score_ada = f1_score(y_test, predictions_ada, average=None)
     print("F1 Score: %.2f%%" % (f1score_ada[1] * 100.0))

     recall_ada = recall_score(y_test, predictions_ada, average=None)
     print("Recall: %.2f%%" % (recall_ada[1] * 100.0))
```

```
ADA BOOT CLASSIFIER: PERFORMANCE METRICS


Accuracy: 98.82%
Precision: 98.77%
F1 Score: 98.77%
Recall: 98.77%
```

```
class_names = ['True', 'False']

np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
titles_options = [("Confusion matrix, without normalization", None),
                  ("Normalized confusion matrix", 'true')]
for title, normalize in titles_options:
    disp = plot_confusion_matrix(ada_classifier, X_test, y_test,
                                 display_labels=class_names,
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
    disp.ax_.set_title(title)

    print()
    print(title)
    print(disp.confusion_matrix)

plt.show()
```

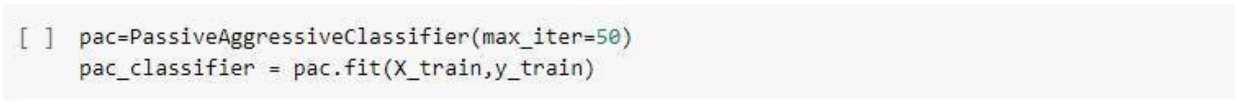
```
Confusion matrix, without normalization
[[5810   66]
 [  66 5283]]

Normalized confusion matrix
[[0.99 0.01]
 [0.01 0.99]]
```





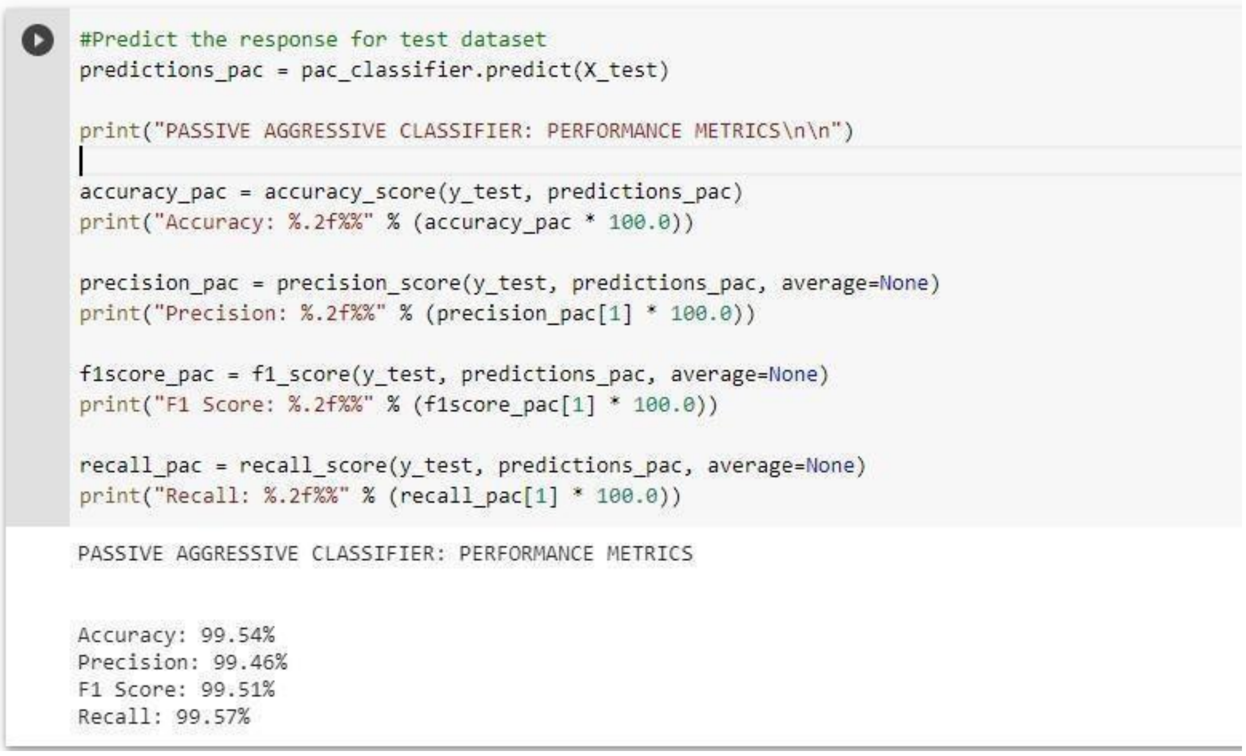
PAC

We create our Passive Aggressive Classifier model and fit ourtraining data to it.



```
[ ]  pac=PassiveAggressiveClassifier(max_iter=50)
     pac_classifier = pac.fit(X_train,y_train)
```

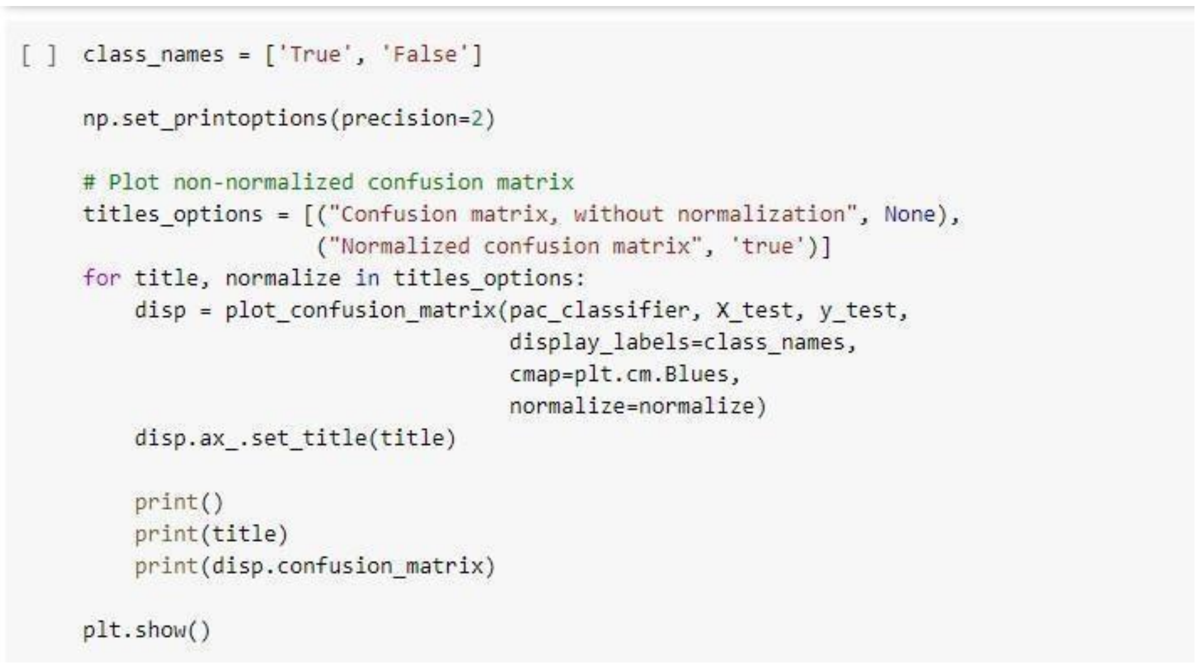
We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 scoreand recall. We get Accuracy - 99.54%, Precision- 99.46%, F1 Score-99.51% and Recall-99.57%.



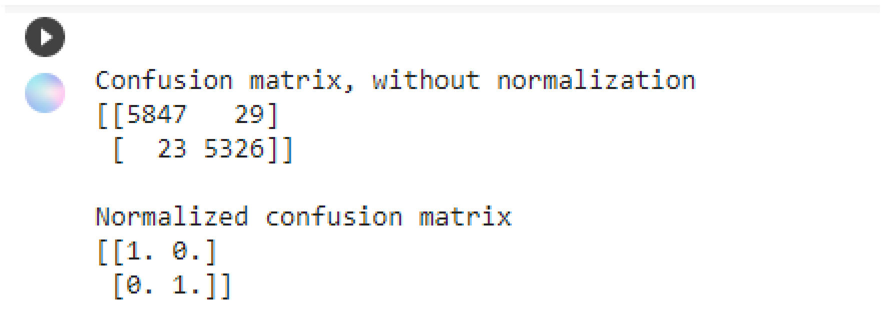
```
#Predict the response for test dataset
predictions_pac = pac_classifier.predict(X_test)

print("PASSIVE AGGRESSIVE CLASSIFIER: PERFORMANCE METRICS\n\n")

accuracy_pac = accuracy_score(y_test, predictions_pac)
print("Accuracy: %.2f%%" % (accuracy_pac * 100.0))

precision_pac = precision_score(y_test, predictions_pac, average=None)
print("Precision: %.2f%%" % (precision_pac[1] * 100.0))

f1score_pac = f1_score(y_test, predictions_pac, average=None)
print("F1 Score: %.2f%%" % (f1score_pac[1] * 100.0))

recall_pac = recall_score(y_test, predictions_pac, average=None)
print("Recall: %.2f%%" % (recall_pac[1] * 100.0))
```

```
PASSIVE AGGRESSIVE CLASSIFIER: PERFORMANCE METRICS


Accuracy: 99.54%
Precision: 99.46%
F1 Score: 99.51%
Recall: 99.57%
```

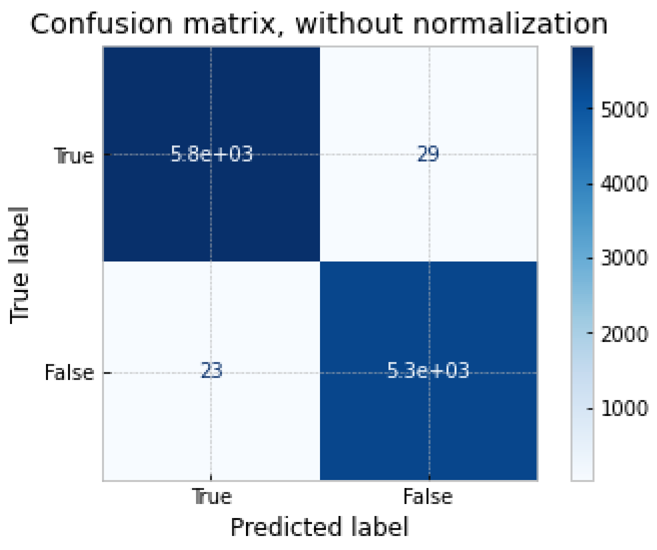
We plot a confusion matrix and a normalized confusion matrix.

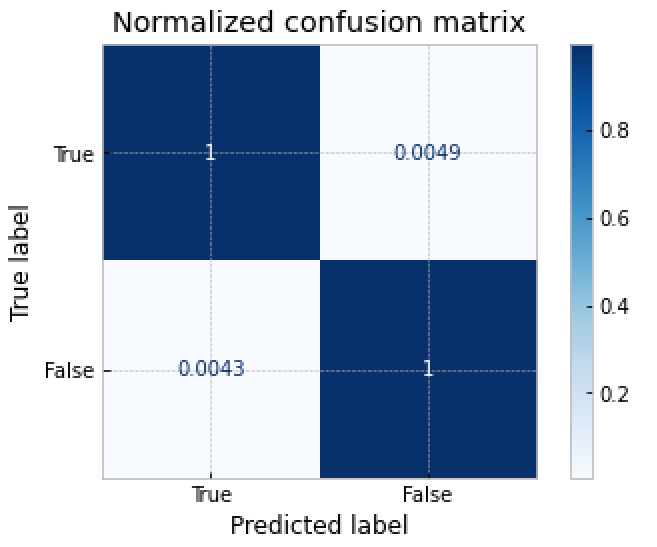


```
[ ]  class_names = ['True', 'False']

     np.set_printoptions(precision=2)

     # Plot non-normalized confusion matrix
     titles_options = [("Confusion matrix, without normalization", None),
                       ("Normalized confusion matrix", 'true')]
     for title, normalize in titles_options:
         disp = plot_confusion_matrix(pac_classifier, X_test, y_test,
                                      display_labels=class_names,
                                      cmap=plt.cm.Blues,
                                      normalize=normalize)
         disp.ax_.set_title(title)

         print()
         print(title)
         print(disp.confusion_matrix)

     plt.show()
```



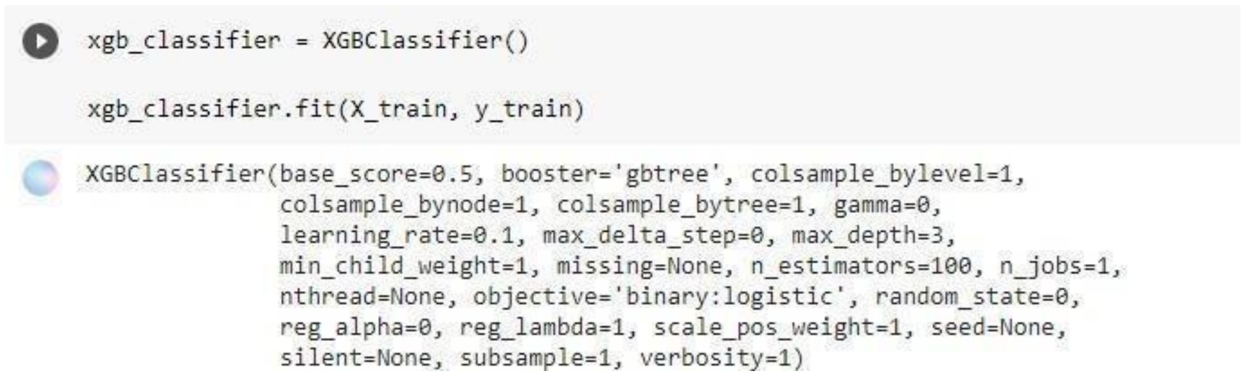
```
Confusion matrix, without normalization
[[5847   29]
 [  23 5326]]

Normalized confusion matrix
[[1. 0.]
 [0. 1.]]
```





XGBoost

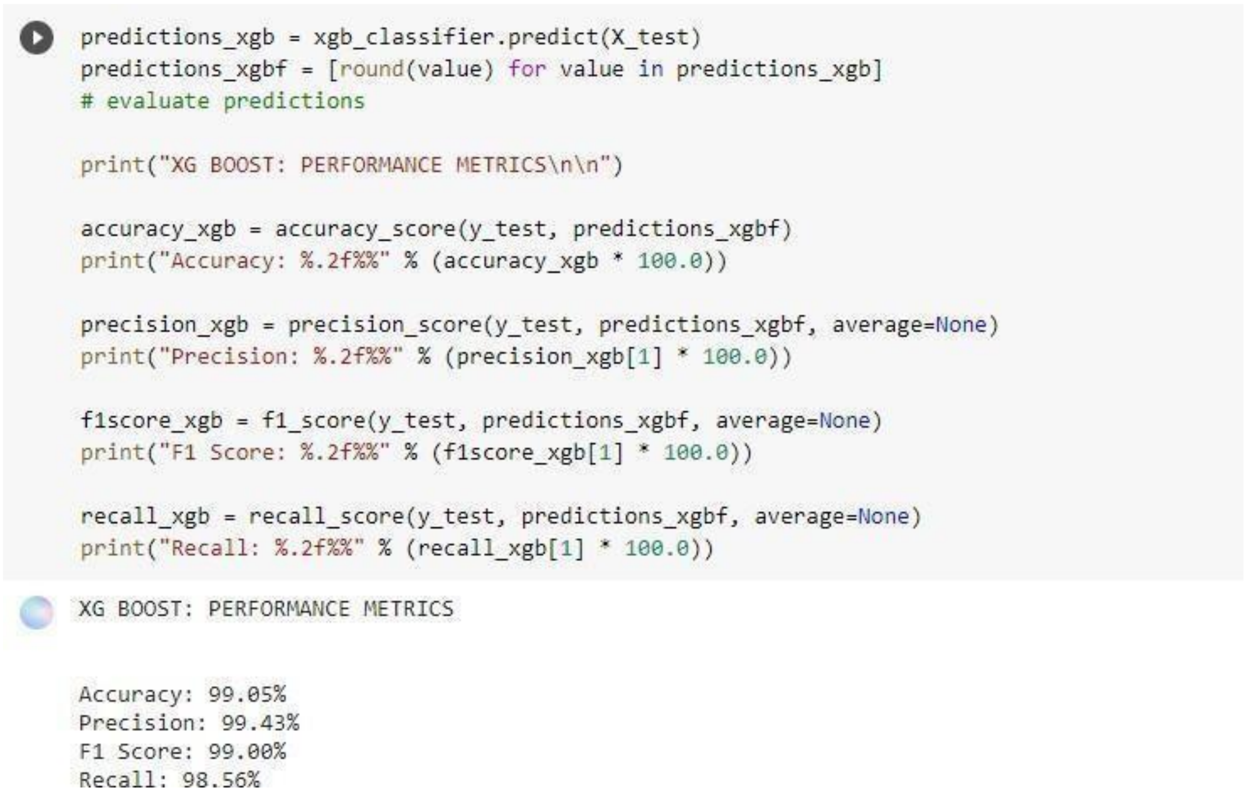
We create our XG Boost Classifier model and fit our training data to it.



```
xgb_classifier = XGBClassifier()

xgb_classifier.fit(X_train, y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

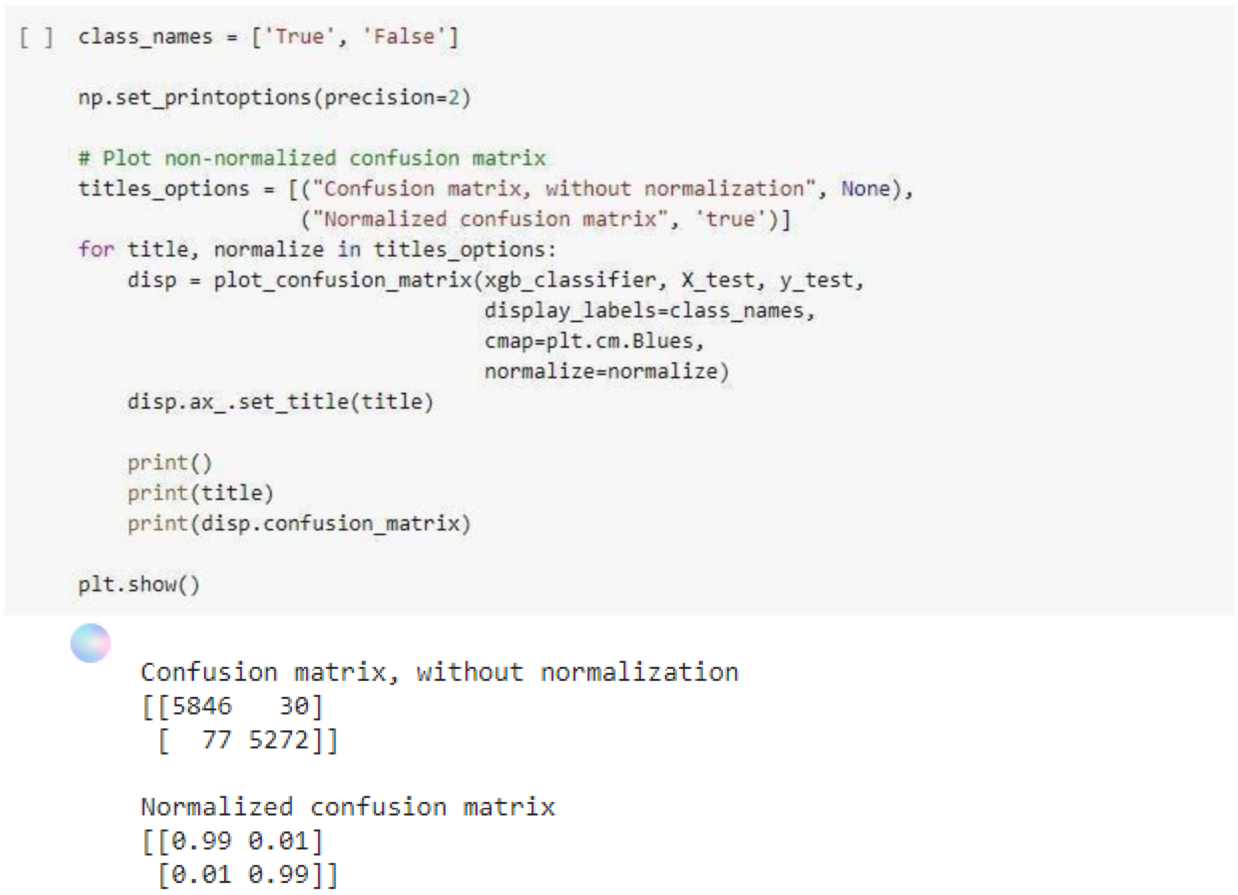
We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 scoreand recall. We get Accuracy - 99.05%, Precision- 99.43%, F1 Score-99.00% and Recall- 98.56%.



```
predictions_xgb = xgb_classifier.predict(X_test)
predictions_xgbf = [round(value) for value in predictions_xgb]
# evaluate predictions

print("XG BOOST: PERFORMANCE METRICS\n\n")

accuracy_xgb = accuracy_score(y_test, predictions_xgbf)
print("Accuracy: %.2f%%" % (accuracy_xgb * 100.0))

precision_xgb = precision_score(y_test, predictions_xgbf, average=None)
print("Precision: %.2f%%" % (precision_xgb[1] * 100.0))

f1score_xgb = f1_score(y_test, predictions_xgbf, average=None)
print("F1 Score: %.2f%%" % (f1score_xgb[1] * 100.0))

recall_xgb = recall_score(y_test, predictions_xgbf, average=None)
print("Recall: %.2f%%" % (recall_xgb[1] * 100.0))
```

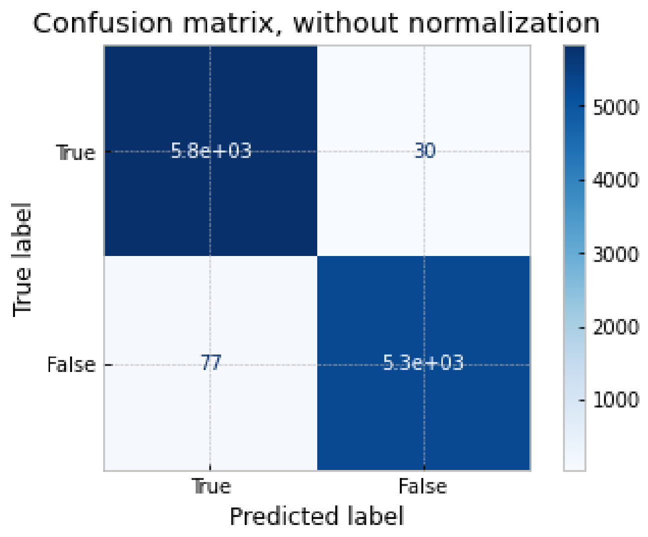
```
XG BOOST: PERFORMANCE METRICS


Accuracy: 99.05%
Precision: 99.43%
F1 Score: 99.00%
Recall: 98.56%
```

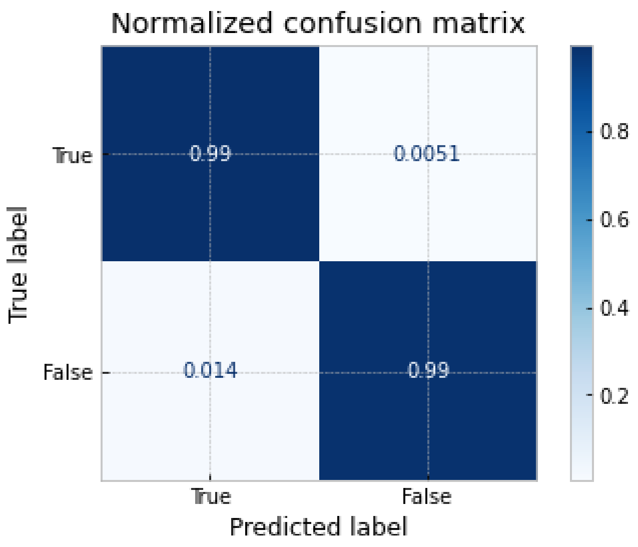
We plot a confusion matrix and a normalized confusion matrix.



```
[ ]  class_names = ['True', 'False']

     np.set_printoptions(precision=2)

     # Plot non-normalized confusion matrix
     titles_options = [("Confusion matrix, without normalization", None),
                       ("Normalized confusion matrix", 'true')]
     for title, normalize in titles_options:
         disp = plot_confusion_matrix(xgb_classifier, X_test, y_test,
                                      display_labels=class_names,
                                      cmap=plt.cm.Blues,
                                      normalize=normalize)
         disp.ax_.set_title(title)

         print()
         print(title)
         print(disp.confusion_matrix)

     plt.show()
```

```
Confusion matrix, without normalization
[[5846   30]
 [  77 5272]]

Normalized confusion matrix
[[0.99 0.01]
 [0.01 0.99]]
```

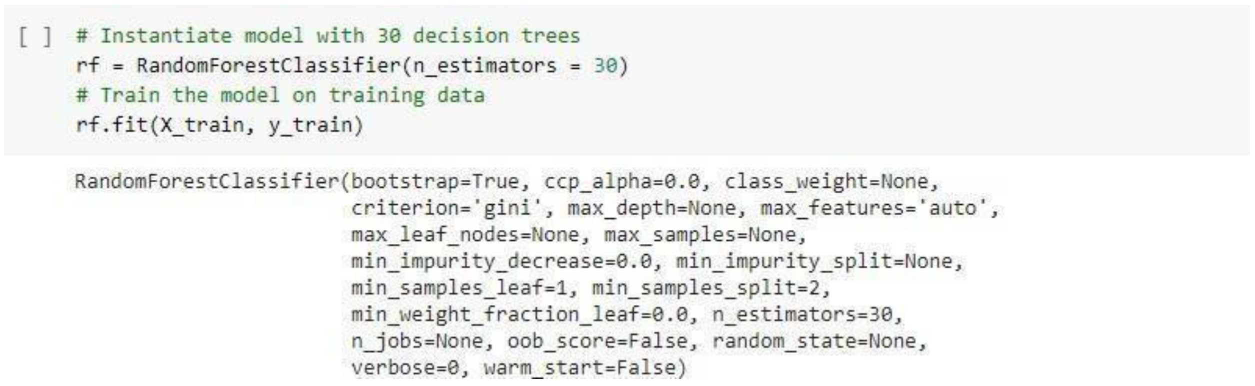




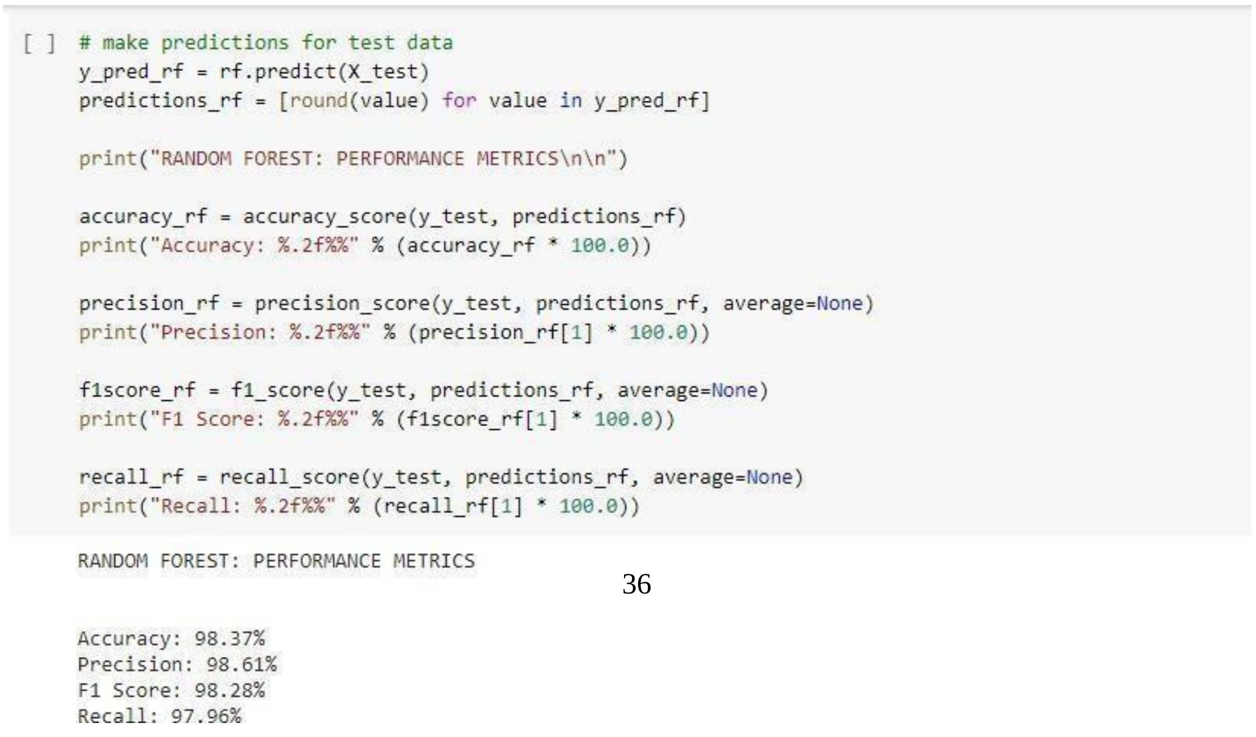
RANDOM FOREST

We create our Random Forest Classifier model and fit our trainingdata to it.

We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 scoreand recall. We get Accuracy



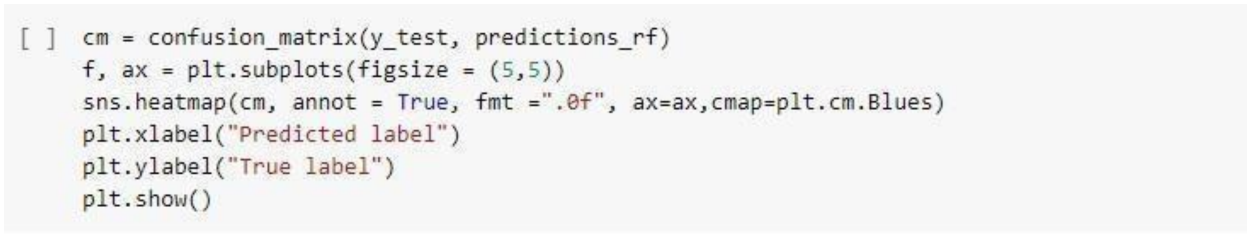
```
[ ]  # Instantiate model with 30 decision trees
     rf = RandomForestClassifier(n_estimators = 30)
     # Train the model on training data
     rf.fit(X_train, y_train)

     RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                            criterion='gini', max_depth=None, max_features='auto',
                            max_leaf_nodes=None, max_samples=None,
                            min_impurity_decrease=0.0, min_impurity_split=None,
                            min_samples_leaf=1, min_samples_split=2,
                            min_weight_fraction_leaf=0.0, n_estimators=30,
                            n_jobs=None, oob_score=False, random_state=None,
                            verbose=0, warm_start=False)
```



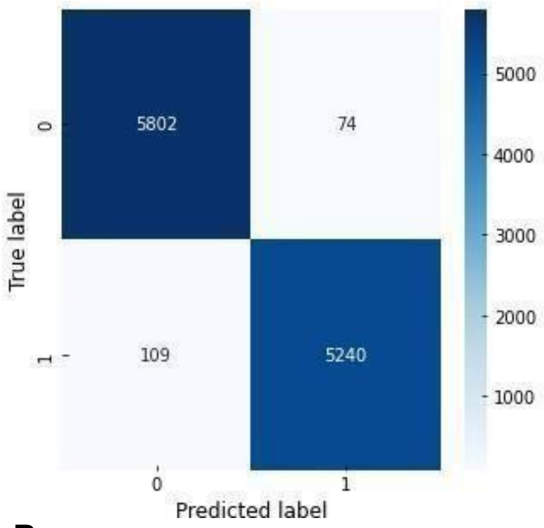
```
[ ]  # make predictions for test data
     y_pred_rf = rf.predict(X_test)
     predictions_rf = [round(value) for value in y_pred_rf]

     print("RANDOM FOREST: PERFORMANCE METRICS\n\n")

     accuracy_rf = accuracy_score(y_test, predictions_rf)
     print("Accuracy: %.2f%%" % (accuracy_rf * 100.0))

     precision_rf = precision_score(y_test, predictions_rf, average=None)
     print("Precision: %.2f%%" % (precision_rf[1] * 100.0))

     f1score_rf = f1_score(y_test, predictions_rf, average=None)
     print("F1 Score: %.2f%%" % (f1score_rf[1] * 100.0))

     recall_rf = recall_score(y_test, predictions_rf, average=None)
     print("Recall: %.2f%%" % (recall_rf[1] * 100.0))

     RANDOM FOREST: PERFORMANCE METRICS


     Accuracy: 98.37%
     Precision: 98.61%
     F1 Score: 98.28%
     Recall: 97.96%
```

- 98.37%, Precision- 98.61%, F1 Score-98.28% and Recall- 97.96%.

We plot a confusion matrix.

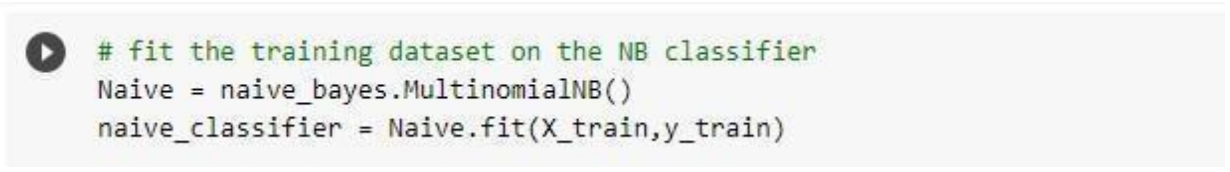


```
[ ] cm = confusion_matrix(y_test, predictions_rf)
    f, ax = plt.subplots(figsize = (5,5))
    sns.heatmap(cm, annot = True, fmt =".0f", ax=ax,cmap=plt.cm.Blues)
    plt.xlabel("Predicted label")
    plt.ylabel("True label")
    plt.show()
```



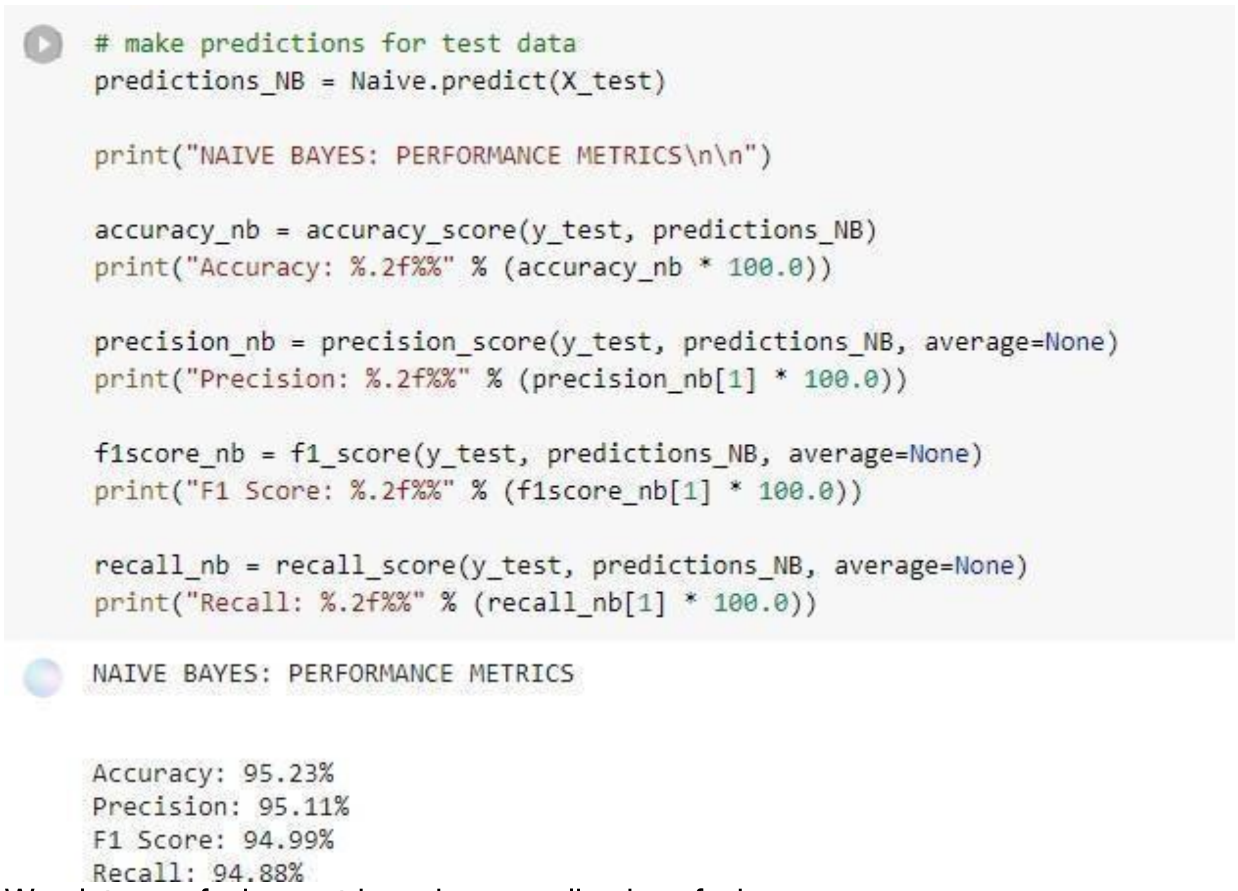
Naive Bayes

We create our Naive Bayes Classifier model and fit our training datato it.



```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
naive_classifier = Naive.fit(X_train,y_train)
```

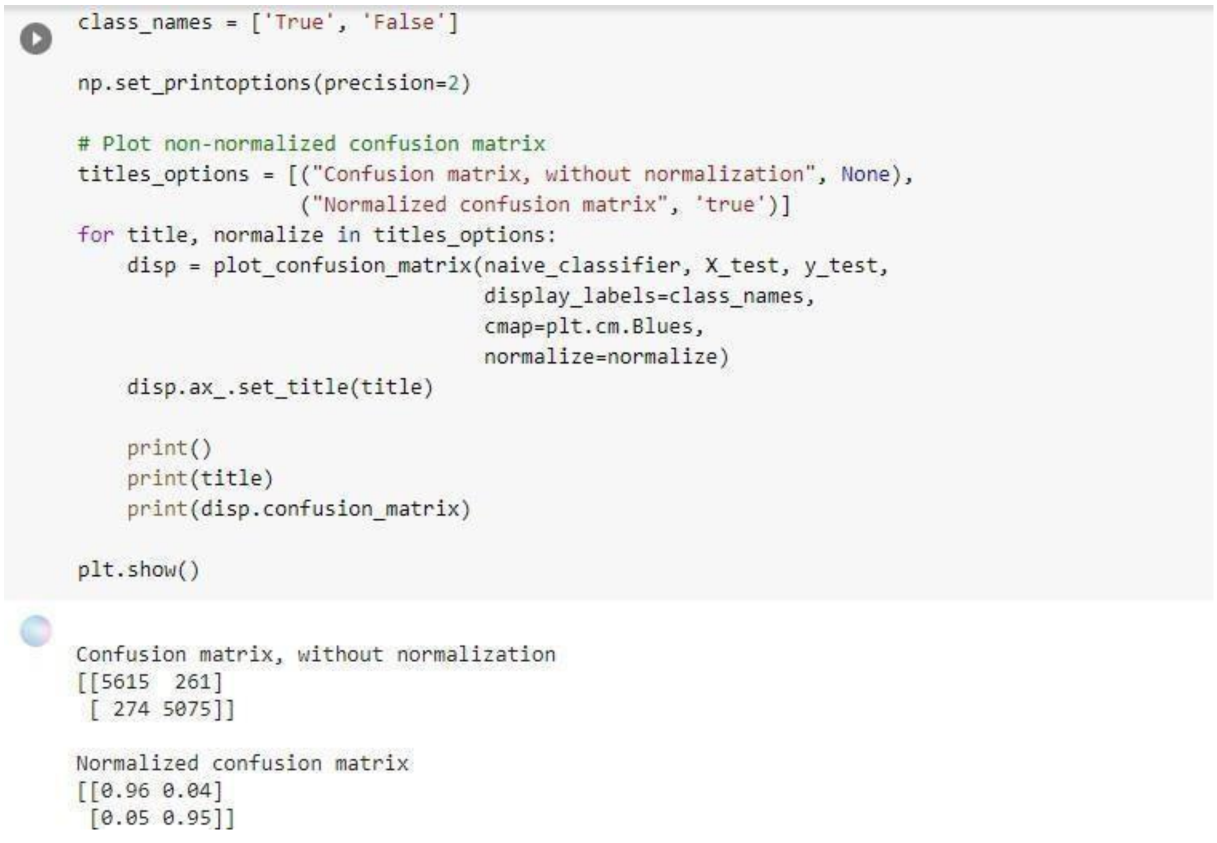
We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 scoreand recall. We get Accuracy - 95.23%, Precision- 95.11%, F1 Score-94.99% and Recall- 94.88%.



```
# make predictions for test data
predictions_NB = Naive.predict(X_test)

print("NAIVE BAYES: PERFORMANCE METRICS\n\n")

accuracy_nb = accuracy_score(y_test, predictions_NB)
print("Accuracy: %.2f%%" % (accuracy_nb * 100.0))

precision_nb = precision_score(y_test, predictions_NB, average=None)
print("Precision: %.2f%%" % (precision_nb[1] * 100.0))

f1score_nb = f1_score(y_test, predictions_NB, average=None)
print("F1 Score: %.2f%%" % (f1score_nb[1] * 100.0))

recall_nb = recall_score(y_test, predictions_NB, average=None)
print("Recall: %.2f%%" % (recall_nb[1] * 100.0))
```

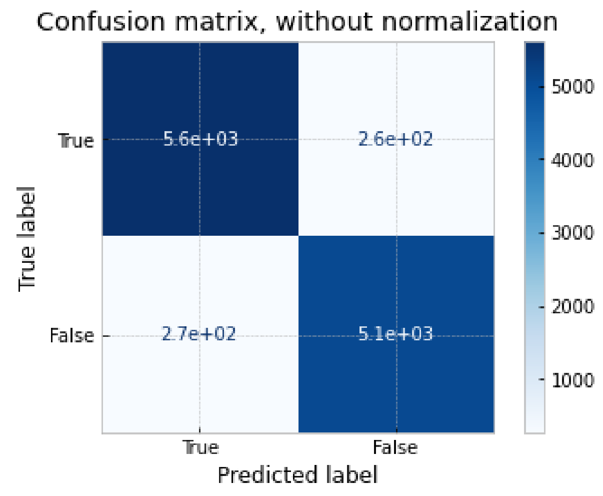
```
NAIVE BAYES: PERFORMANCE METRICS


Accuracy: 95.23%
Precision: 95.11%
F1 Score: 94.99%
Recall: 94.88%
```

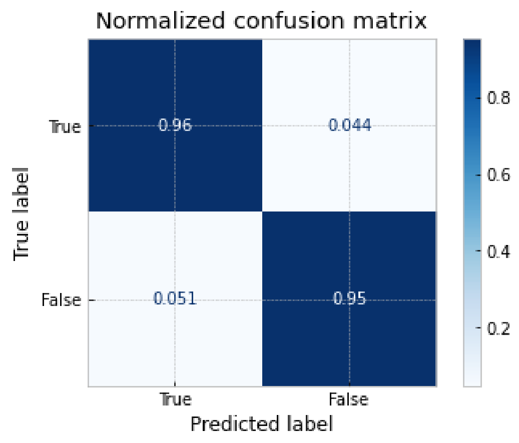
We plot a confusion matrix and a normalized confusion matrix.



```
class_names = ['True', 'False']

np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
titles_options = [("Confusion matrix, without normalization", None),
                  ("Normalized confusion matrix", 'true')]
for title, normalize in titles_options:
    disp = plot_confusion_matrix(naive_classifier, X_test, y_test,
                                 display_labels=class_names,
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
    disp.ax_.set_title(title)

    print()
    print(title)
    print(disp.confusion_matrix)

plt.show()
```

```
Confusion matrix, without normalization
[[5615  261]
 [ 274 5075]]

Normalized confusion matrix
[[0.96 0.04]
 [0.05 0.95]]
```





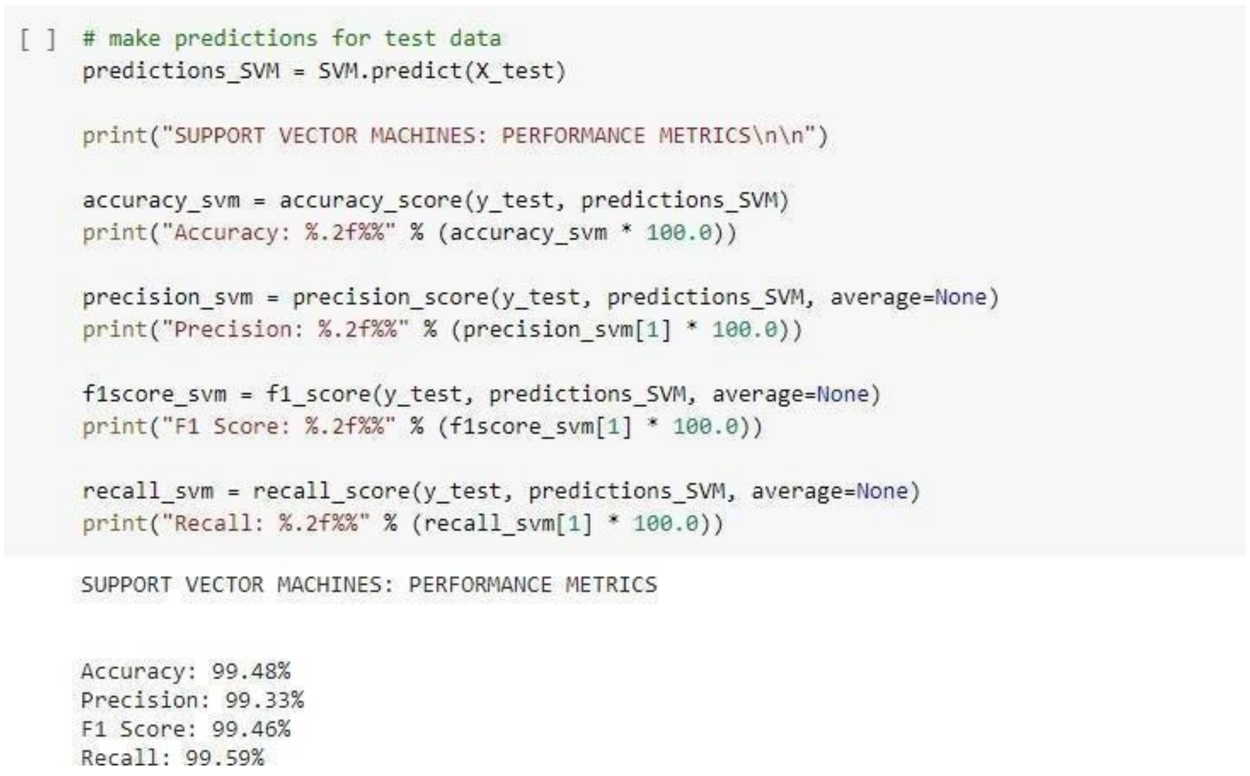
Support Vector Machine (SVM)

We create our SVM Classifier model and fit our training data to it.

```
[ ]  SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
     svm_classifier = SVM.fit(X_train,y_train)
```

We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 score and recall. We get Accuracy

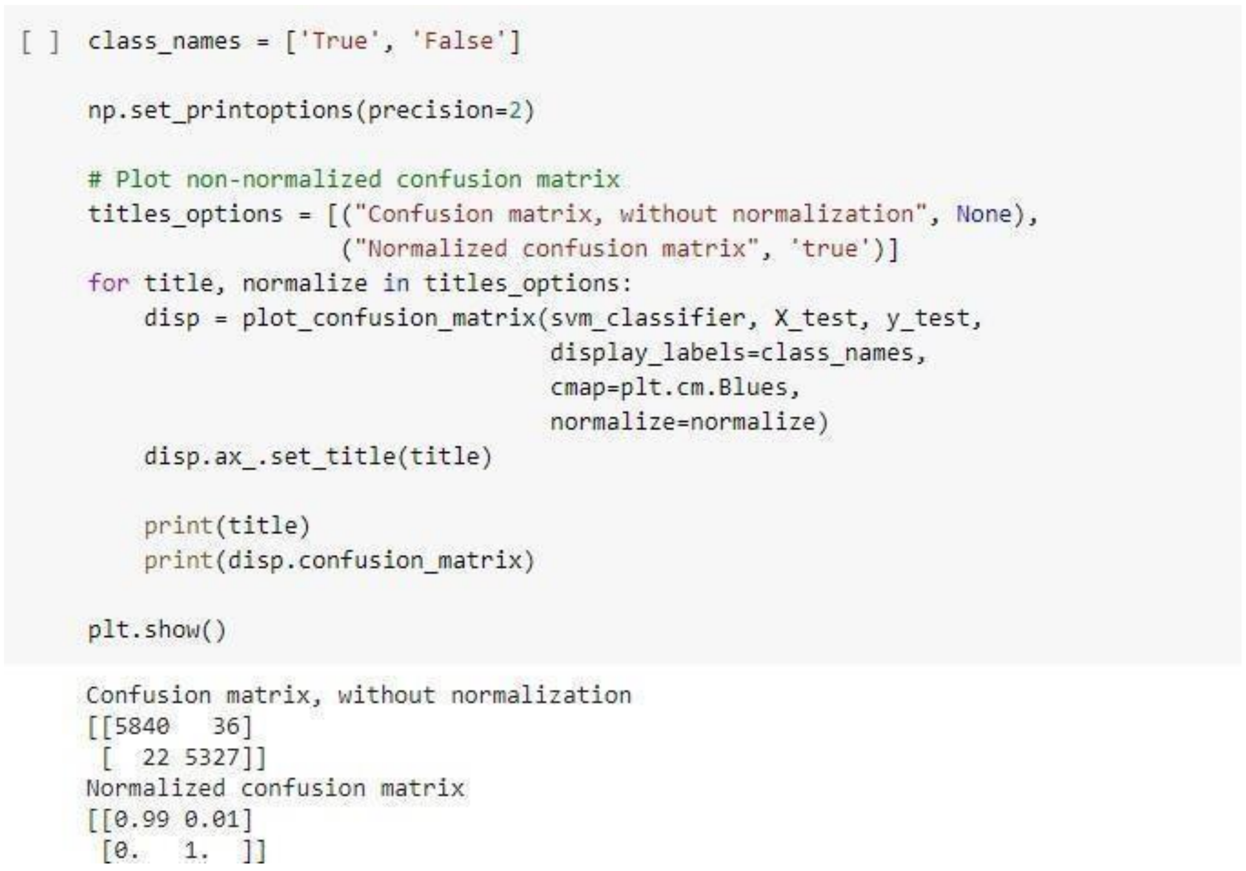
- 99.48%, Precision- 99.33%, F1 Score- 99.46% and Recall- 99.59%.



```
[ ]  # make predictions for test data
     predictions_SVM = SVM.predict(X_test)

     print("SUPPORT VECTOR MACHINES: PERFORMANCE METRICS\n\n")

     accuracy_svm = accuracy_score(y_test, predictions_SVM)
     print("Accuracy: %.2f%%" % (accuracy_svm * 100.0))

     precision_svm = precision_score(y_test, predictions_SVM, average=None)
     print("Precision: %.2f%%" % (precision_svm[1] * 100.0))

     f1score_svm = f1_score(y_test, predictions_SVM, average=None)
     print("F1 Score: %.2f%%" % (f1score_svm[1] * 100.0))

     recall_svm = recall_score(y_test, predictions_SVM, average=None)
     print("Recall: %.2f%%" % (recall_svm[1] * 100.0))
```

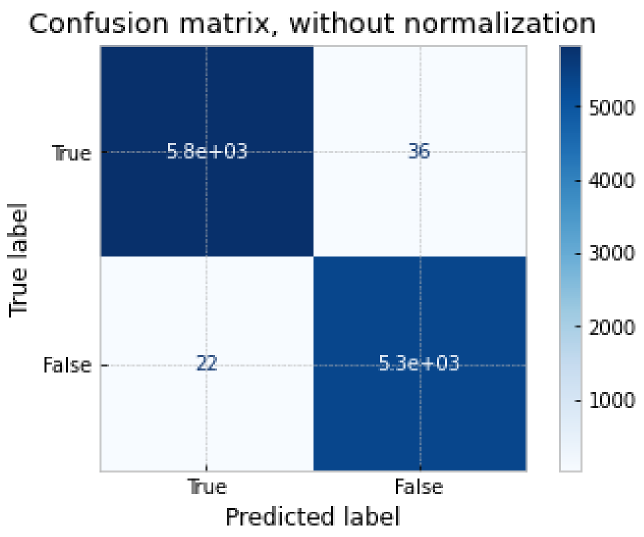
```
SUPPORT VECTOR MACHINES: PERFORMANCE METRICS


Accuracy: 99.48%
Precision: 99.33%
F1 Score: 99.46%
Recall: 99.59%
```

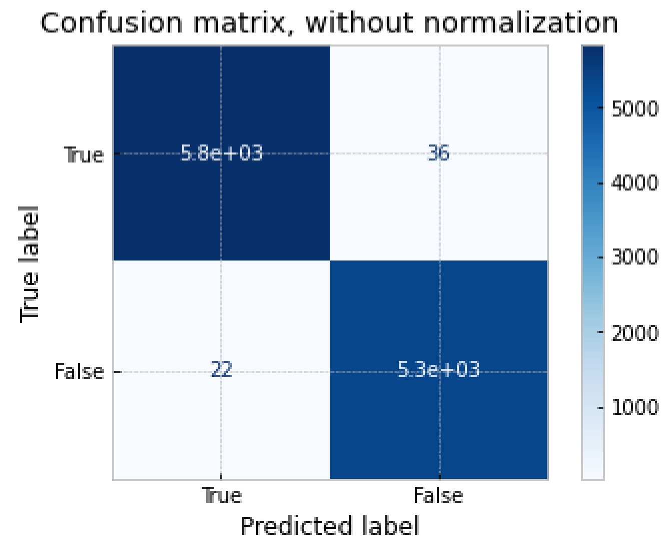
We plot a confusion matrix and a normalized confusion matrix.



```
[ ] class_names = ['True', 'False']

    np.set_printoptions(precision=2)

    # Plot non-normalized confusion matrix
    titles_options = [("Confusion matrix, without normalization", None),
                      ("Normalized confusion matrix", 'true')]
    for title, normalize in titles_options:
        disp = plot_confusion_matrix(svm_classifier, X_test, y_test,
                                     display_labels=class_names,
                                     cmap=plt.cm.Blues,
                                     normalize=normalize)
        disp.ax_.set_title(title)

        print(title)
        print(disp.confusion_matrix)

    plt.show()
```

```
Confusion matrix, without normalization
[[5840   36]
 [  22 5327]]
Normalized confusion matrix
[[0.99 0.01]
 [0.   1.  ]]
```



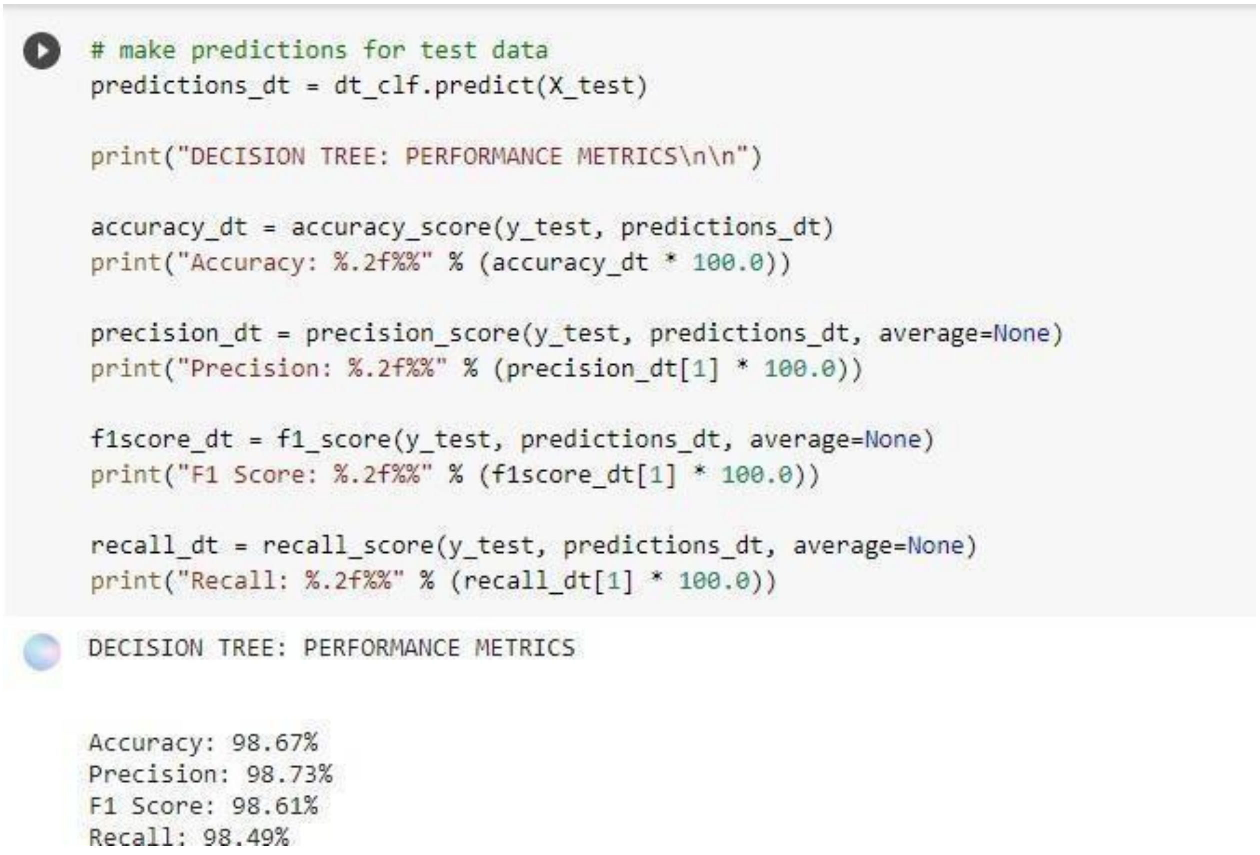


Decision Tree

We create our Decision Tree Classifier model and fit our training datato it.

```
[ ]  dt_clf = DecisionTreeClassifier()

     dt_clf = dt_clf.fit(X_train,y_train)
```

We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 scoreand recall. We get Accuracy - 98.67%, Precision- 98.73%, F1 Score-98.61% and Recall- 98.49%.

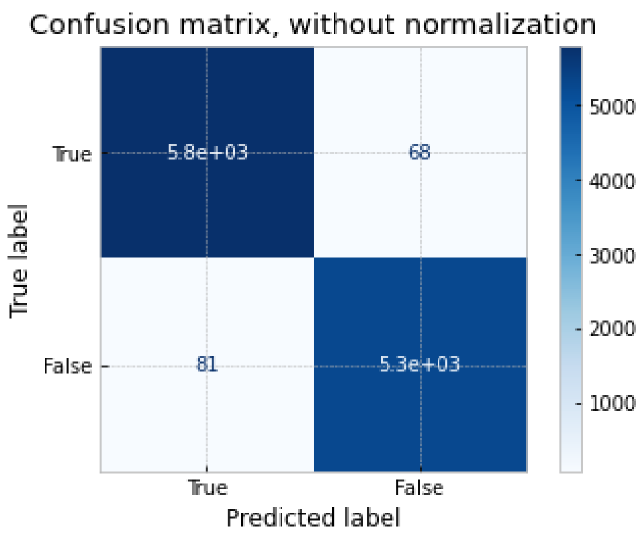

```
# make predictions for test data
predictions_dt = dt_clf.predict(X_test)

print("DECISION TREE: PERFORMANCE METRICS\n\n")

accuracy_dt = accuracy_score(y_test, predictions_dt)
print("Accuracy: %.2f%%" % (accuracy_dt * 100.0))

precision_dt = precision_score(y_test, predictions_dt, average=None)
print("Precision: %.2f%%" % (precision_dt[1] * 100.0))

f1score_dt = f1_score(y_test, predictions_dt, average=None)
print("F1 Score: %.2f%%" % (f1score_dt[1] * 100.0))

recall_dt = recall_score(y_test, predictions_dt, average=None)
print("Recall: %.2f%%" % (recall_dt[1] * 100.0))
```

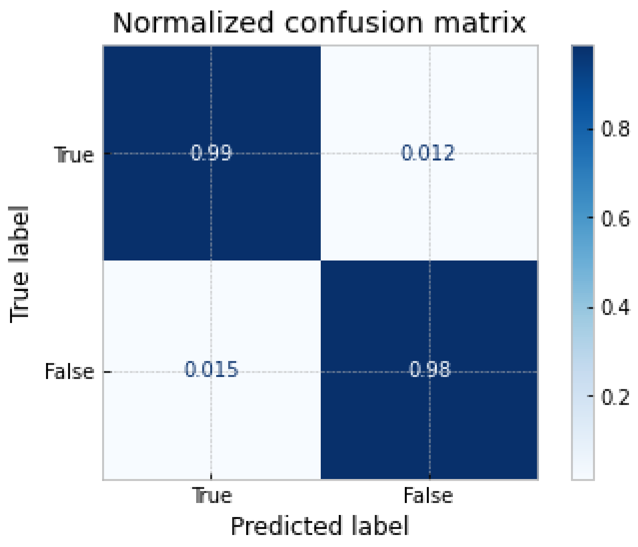
```
DECISION TREE: PERFORMANCE METRICS


Accuracy: 98.67%
Precision: 98.73%
F1 Score: 98.61%
Recall: 98.49%
```

We plot a confusion matrix and a normalized confusion matrix.

```
[ ]  class_names = ['True', 'False']

     np.set_printoptions(precision=2)

     # Plot non-normalized confusion matrix
     titles_options = [("Confusion matrix, without normalization", None),
                       ("Normalized confusion matrix", 'true')]
     for title, normalize in titles_options:
         disp = plot_confusion_matrix(dt_clf, X_test, y_test,
                                      display_labels=class_names,
                                      cmap=plt.cm.Blues,
                                      normalize=normalize)
         disp.ax_.set_title(title)
         print()
         print(title)
         print(disp.confusion_matrix)

     plt.show()
```

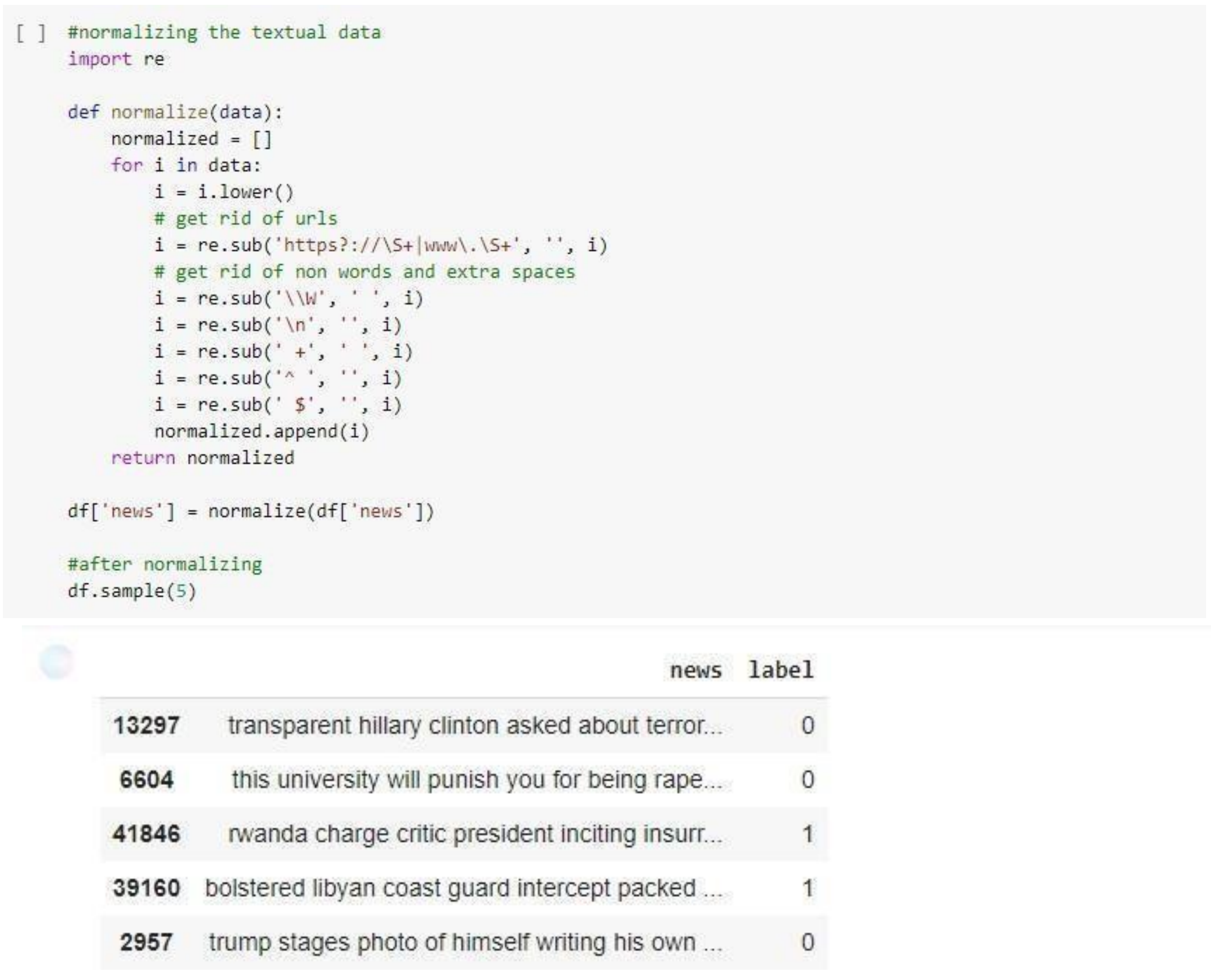
```
Confusion matrix, without normalization
[[5808   68]
 [  81 5268]]

Normalized confusion matrix
[[0.99 0.01]
 [0.02 0.98]]
```





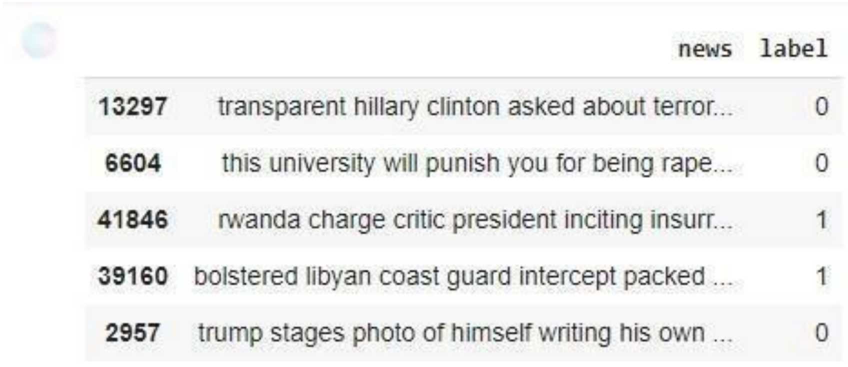
RNN

For our RNN model, we first preprocess the dataset news by normalizing it by removing non-words and extra space.



```
[ ] #normalizing the textual data
    import re

    def normalize(data):
        normalized = []
        for i in data:
            i = i.lower()
            # get rid of urls
            i = re.sub('https?://\S+|www\.\S+', '', i)
            # get rid of non words and extra spaces
            i = re.sub('\\W', ' ', i)
            i = re.sub('\n', '', i)
            i = re.sub(' +', ' ', i)
            i = re.sub('^ ', '', i)
            i = re.sub(' $', '', i)
            normalized.append(i)
        return normalized

    df['news'] = normalize(df['news'])

    #after normalizing
    df.sample(5)
```

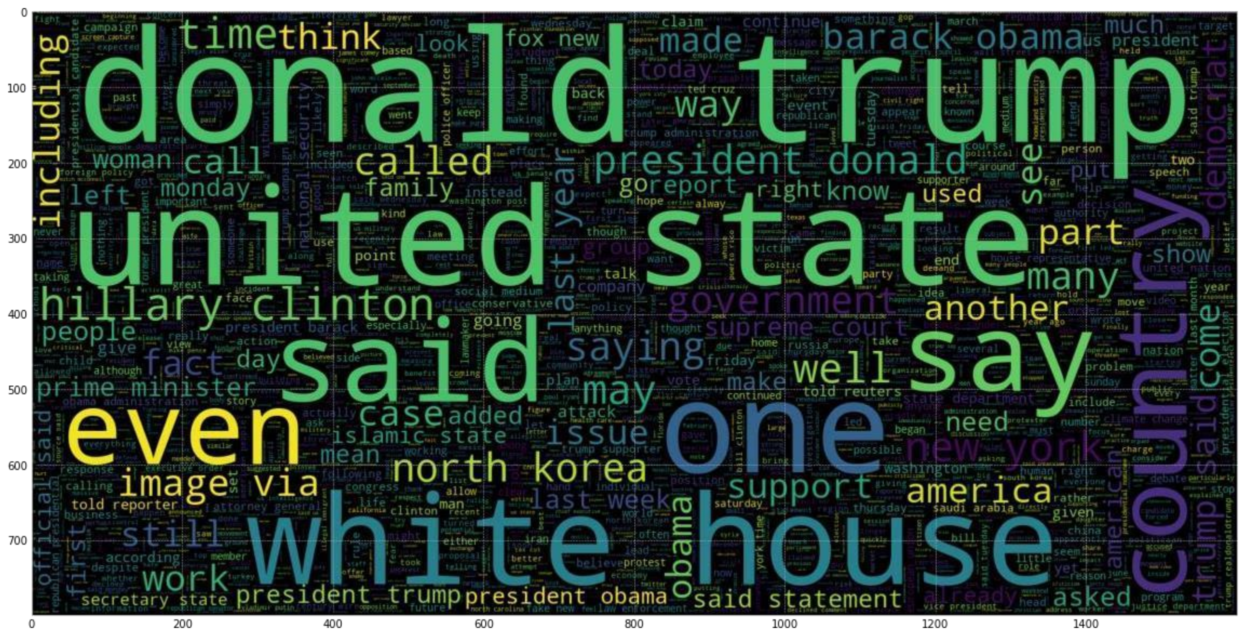
	news	label
13297	transparent hillary clinton asked about terror...	0
6604	this university will punish you for being rape...	0
41846	rwanda charge critic president inciting insurr...	1
39160	bolstered libyan coast guard intercept packed ...	1
2957	trump stages photo of himself writing his own ...	0



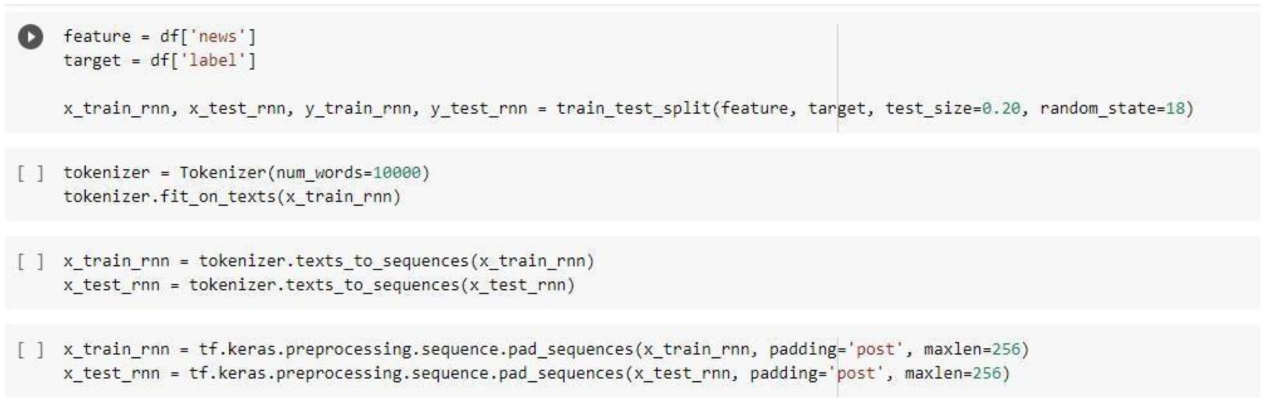
	news	label
13297	transparent hillary clinton asked about terror...	0
6604	this university will punish you for being rape...	0
41846	rwanda charge critic president inciting insurr...	1
39160	bolstered libyan coast guard intercept packed ...	1
2957	trump stages photo of himself writing his own ...	0

We then create a Wordcloud to highlight the popular words andphrases based on frequency and relevance.

```
[ ]  plt.figure(figsize = (20,20))
     wc = WordCloud(max_words = 3000 , width = 1600 , height = 800 , stopwords = STOPWORDS).generate(" ".join(df.news))
     plt.imshow(wc , interpolation = 'bilinear')

     <matplotlib.image.AxesImage at 0x7f8688b05610>
```



After that we divide the dataset features and target into testing andtraining data.



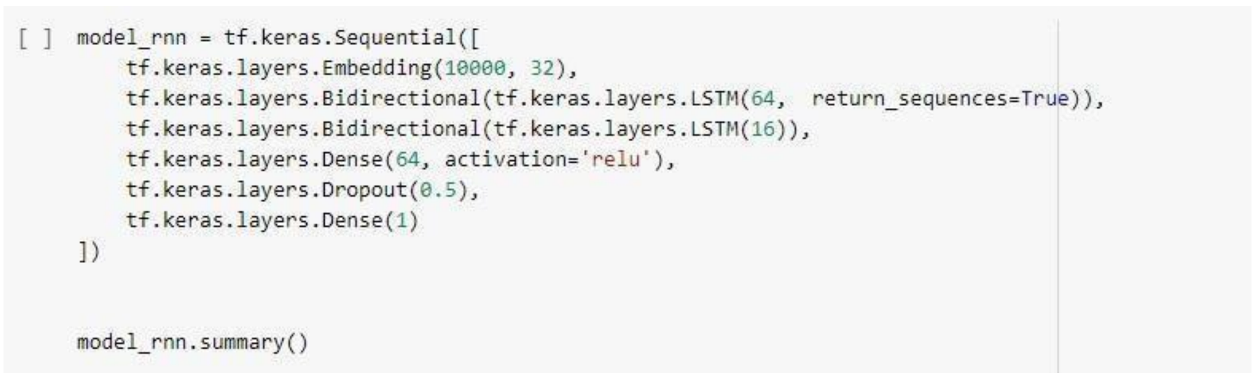
```
feature = df['news']
target = df['label']

x_train_rnn, x_test_rnn, y_train_rnn, y_test_rnn = train_test_split(feature, target, test_size=0.20, random_state=18)
```

```
tokenizer = Tokenizer(num_words=10000)
tokenizer.fit_on_texts(x_train_rnn)
```

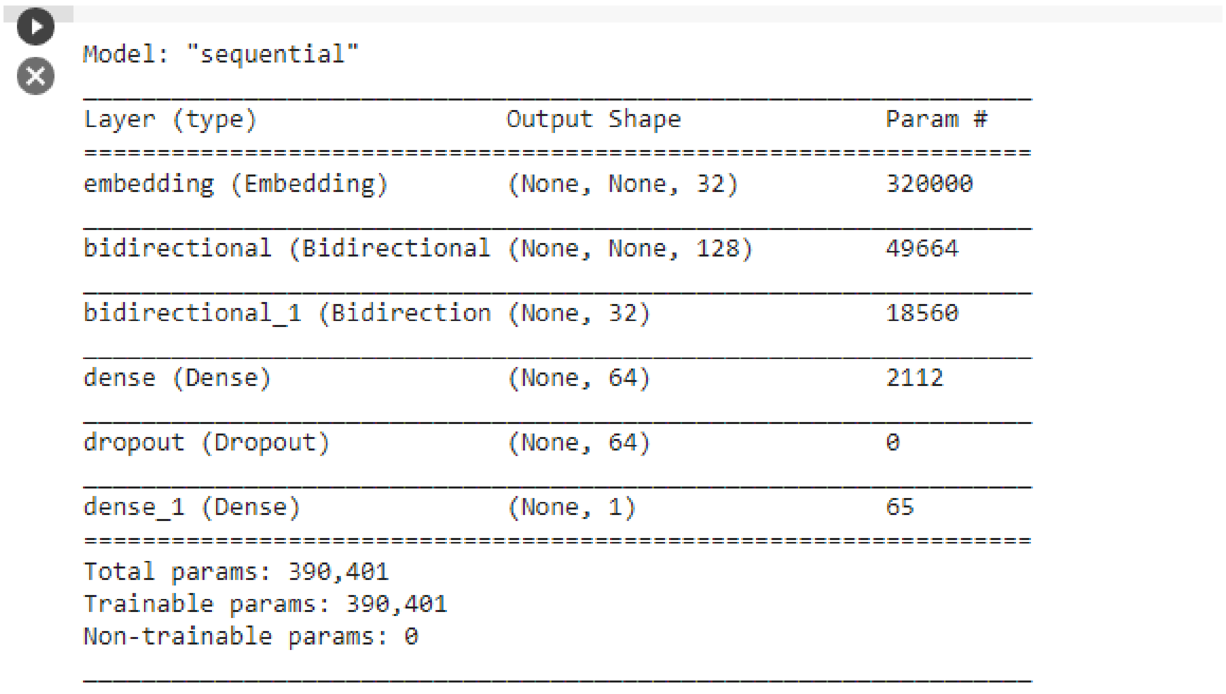
```
x_train_rnn = tokenizer.texts_to_sequences(x_train_rnn)
x_test_rnn = tokenizer.texts_to_sequences(x_test_rnn)
```

```
x_train_rnn = tf.keras.preprocessing.sequence.pad_sequences(x_train_rnn, padding='post', maxlen=256)
x_test_rnn = tf.keras.preprocessing.sequence.pad_sequences(x_test_rnn, padding='post', maxlen=256)
```

Then we build our RNN sequential model and add the layers to it.



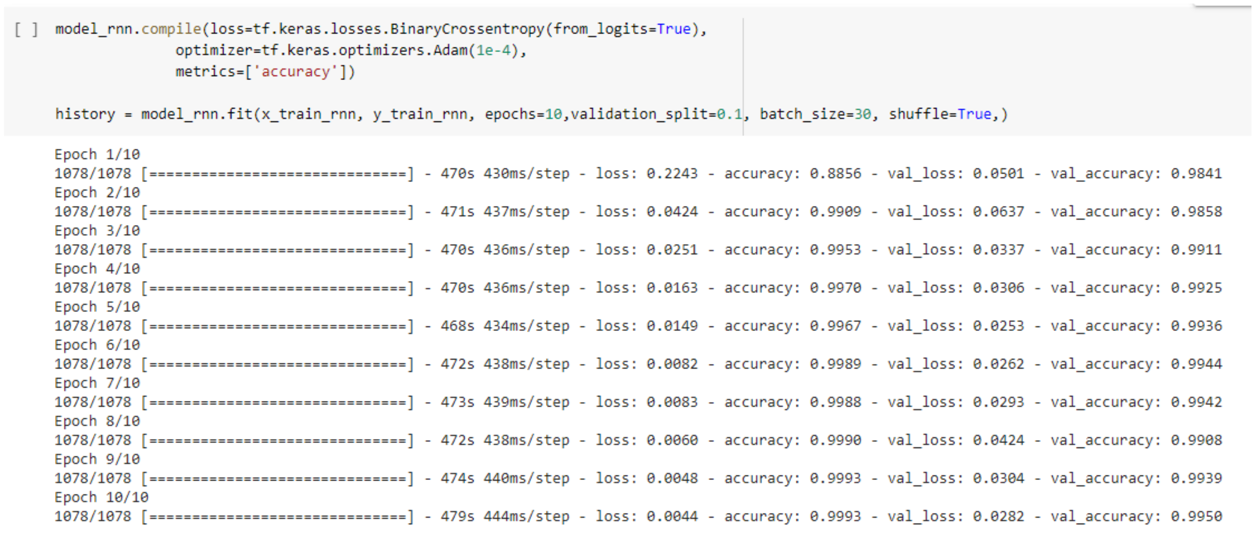
```
model_rnn = tf.keras.Sequential([
    tf.keras.layers.Embedding(10000, 32),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64,  return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(16)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1)
])


model_rnn.summary()
```



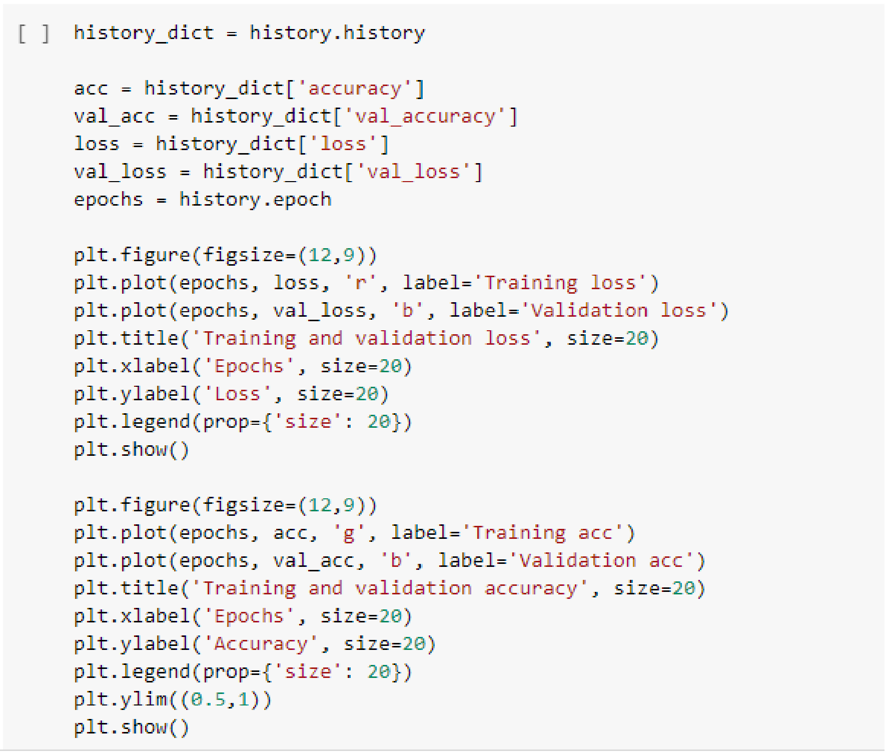
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 32)          320000
_________________________________________________________________
bidirectional (Bidirectional (None, None, 128)         49664
_________________________________________________________________
bidirectional_1 (Bidirection (None, 32)                18560
_________________________________________________________________
dense (Dense)                (None, 64)                2112
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 390,401
Trainable params: 390,401
Non-trainable params: 0
_________________________________________________________________
```

We compile our RNN model and fit the training data to it for 10 epochs. We see that with every epoch, the loss decreases and the accuracy increases. At the 10th epoch, we get a validation accuracyof 99.50% and the validation loss is 0.028, whereas the loss is of 0.0044 and accuracy is 99.93%.

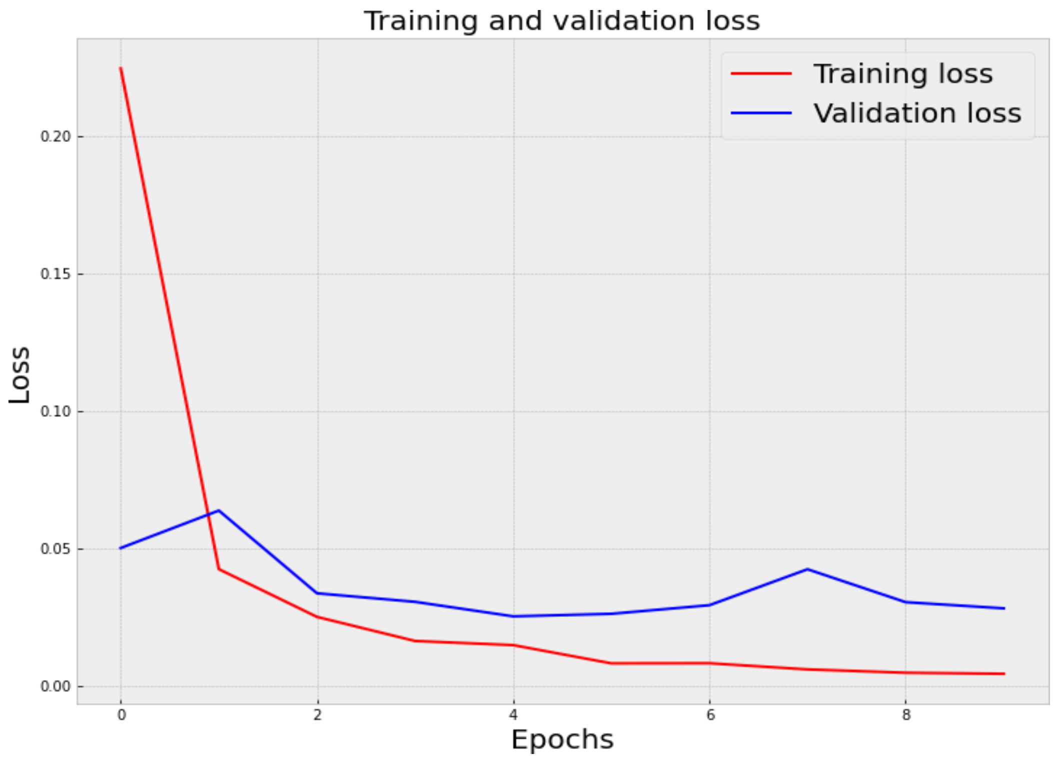
We then use the model history to plot the training loss and accuracy:

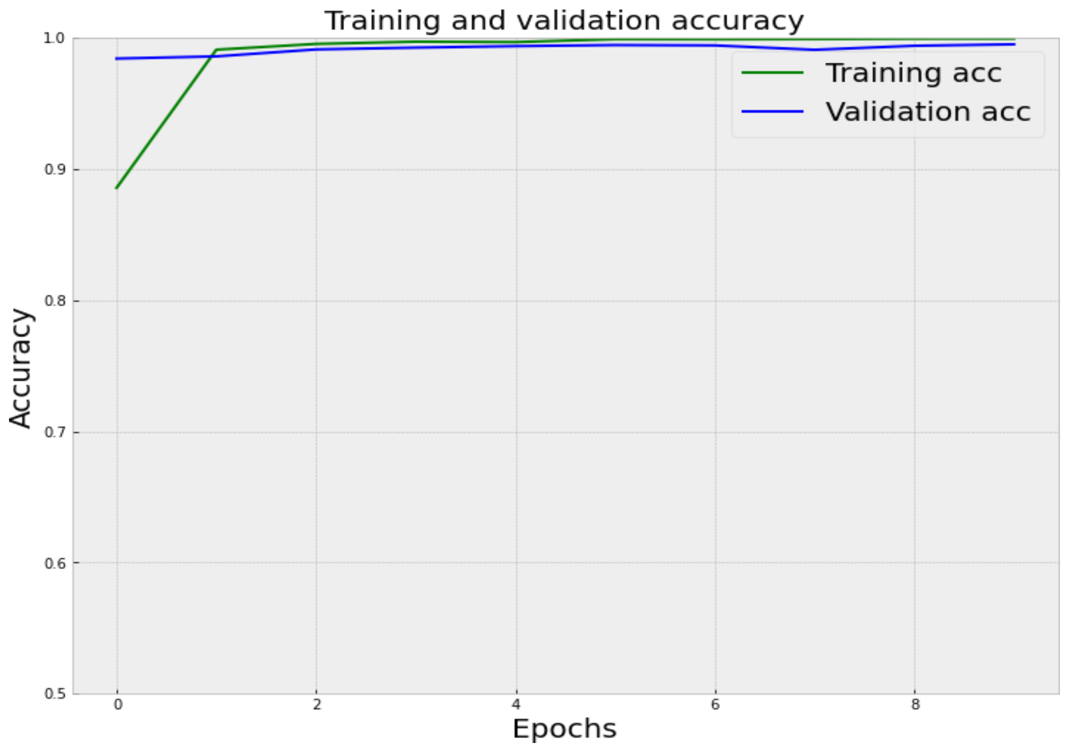


```
[ ]  model_rnn.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
                optimizer=tf.keras.optimizers.Adam(1e-4),
                metrics=['accuracy'])

     history = model_rnn.fit(x_train_rnn, y_train_rnn, epochs=10,validation_split=0.1, batch_size=30, shuffle=True,)

     Epoch 1/10
     1078/1078 [==============================] - 470s 430ms/step - loss: 0.2243 - accuracy: 0.8856 - val_loss: 0.0501 - val_accuracy: 0.9841
     Epoch 2/10
     1078/1078 [==============================] - 471s 437ms/step - loss: 0.0424 - accuracy: 0.9909 - val_loss: 0.0637 - val_accuracy: 0.9858
     Epoch 3/10
     1078/1078 [==============================] - 470s 436ms/step - loss: 0.0251 - accuracy: 0.9953 - val_loss: 0.0337 - val_accuracy: 0.9911
     Epoch 4/10
     1078/1078 [==============================] - 470s 436ms/step - loss: 0.0163 - accuracy: 0.9970 - val_loss: 0.0306 - val_accuracy: 0.9925
     Epoch 5/10
     1078/1078 [==============================] - 468s 434ms/step - loss: 0.0149 - accuracy: 0.9967 - val_loss: 0.0253 - val_accuracy: 0.9936
     Epoch 6/10
     1078/1078 [==============================] - 472s 438ms/step - loss: 0.0082 - accuracy: 0.9989 - val_loss: 0.0262 - val_accuracy: 0.9944
     Epoch 7/10
     1078/1078 [==============================] - 473s 439ms/step - loss: 0.0083 - accuracy: 0.9988 - val_loss: 0.0293 - val_accuracy: 0.9942
     Epoch 8/10
     1078/1078 [==============================] - 472s 438ms/step - loss: 0.0060 - accuracy: 0.9990 - val_loss: 0.0424 - val_accuracy: 0.9908
     Epoch 9/10
     1078/1078 [==============================] - 474s 440ms/step - loss: 0.0048 - accuracy: 0.9993 - val_loss: 0.0304 - val_accuracy: 0.9939
     Epoch 10/10
     1078/1078 [==============================] - 479s 444ms/step - loss: 0.0044 - accuracy: 0.9993 - val_loss: 0.0282 - val_accuracy: 0.9950
```

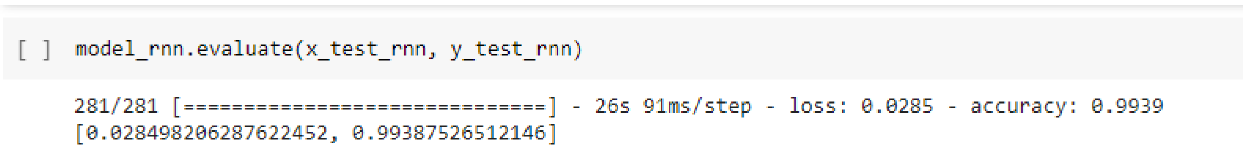


```
[ ]  history_dict = history.history

     acc = history_dict['accuracy']
     val_acc = history_dict['val_accuracy']
     loss = history_dict['loss']
     val_loss = history_dict['val_loss']
     epochs = history.epoch

     plt.figure(figsize=(12,9))
     plt.plot(epochs, loss, 'r', label='Training loss')
     plt.plot(epochs, val_loss, 'b', label='Validation loss')
     plt.title('Training and validation loss', size=20)
     plt.xlabel('Epochs', size=20)
     plt.ylabel('Loss', size=20)
     plt.legend(prop={'size': 20})
     plt.show()

     plt.figure(figsize=(12,9))
     plt.plot(epochs, acc, 'g', label='Training acc')
     plt.plot(epochs, val_acc, 'b', label='Validation acc')
     plt.title('Training and validation accuracy', size=20)
     plt.xlabel('Epochs', size=20)
     plt.ylabel('Accuracy', size=20)
     plt.legend(prop={'size': 20})
     plt.ylim((0.5,1))
     plt.show()
```



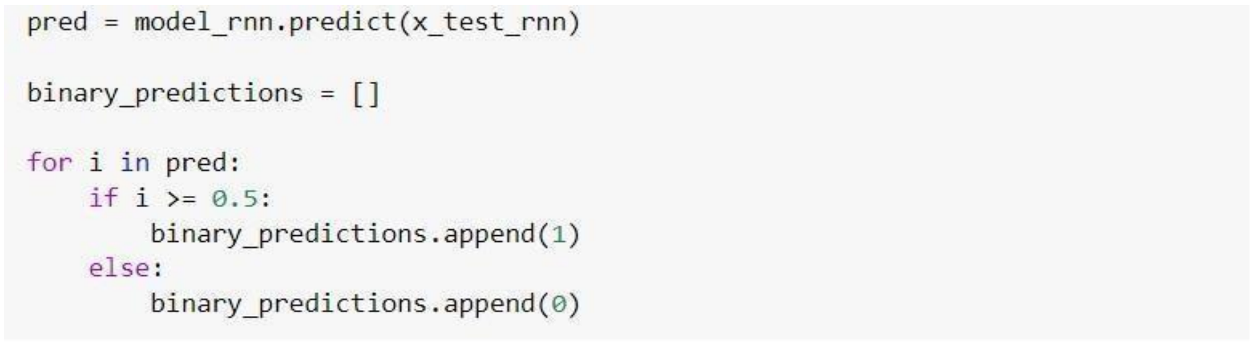


We evaluate the model with the testing data and the loss turns out tobe 2.84% and an accuracy of 99.387%.

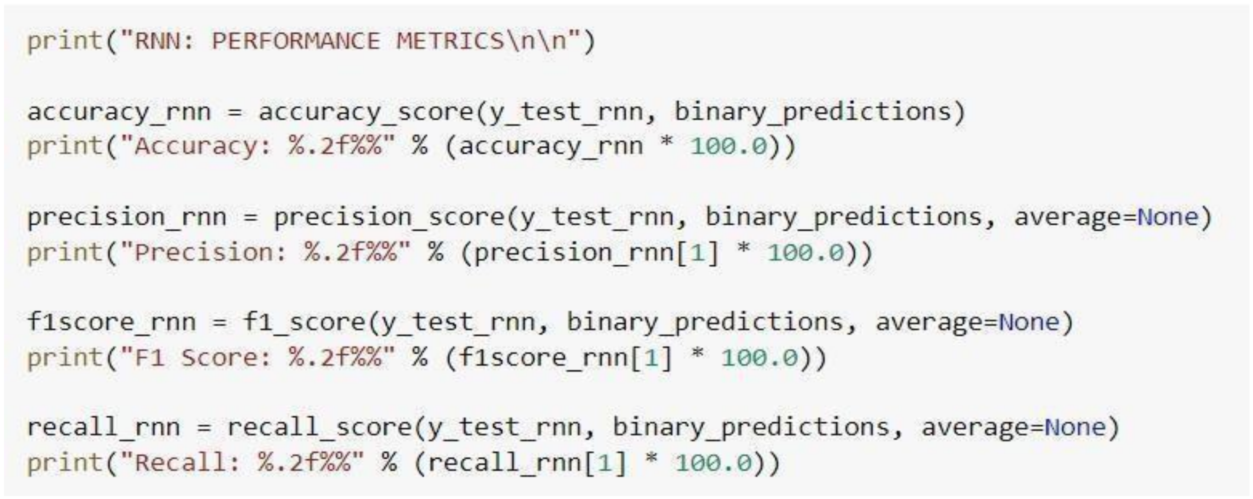


```
[ ]  model_rnn.evaluate(x_test_rnn, y_test_rnn)

     281/281 [==============================] - 26s 91ms/step - loss: 0.0285 - accuracy: 0.9939
     [0.028498206287622452, 0.99387526512146]
```

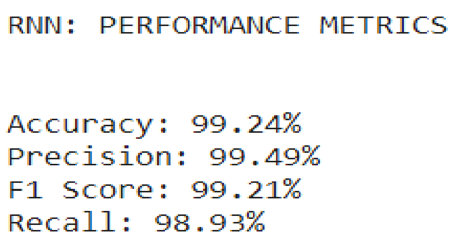
We then calculate other performance metrics. The accuracy as mentioned earlier is 99.39%, the precision is 99.35%, the f1 score is99.36% and the recall is 99.37%.



```
pred = model_rnn.predict(x_test_rnn)

binary_predictions = []

for i in pred:
    if i >= 0.5:
        binary_predictions.append(1)
    else:
        binary_predictions.append(0)
```

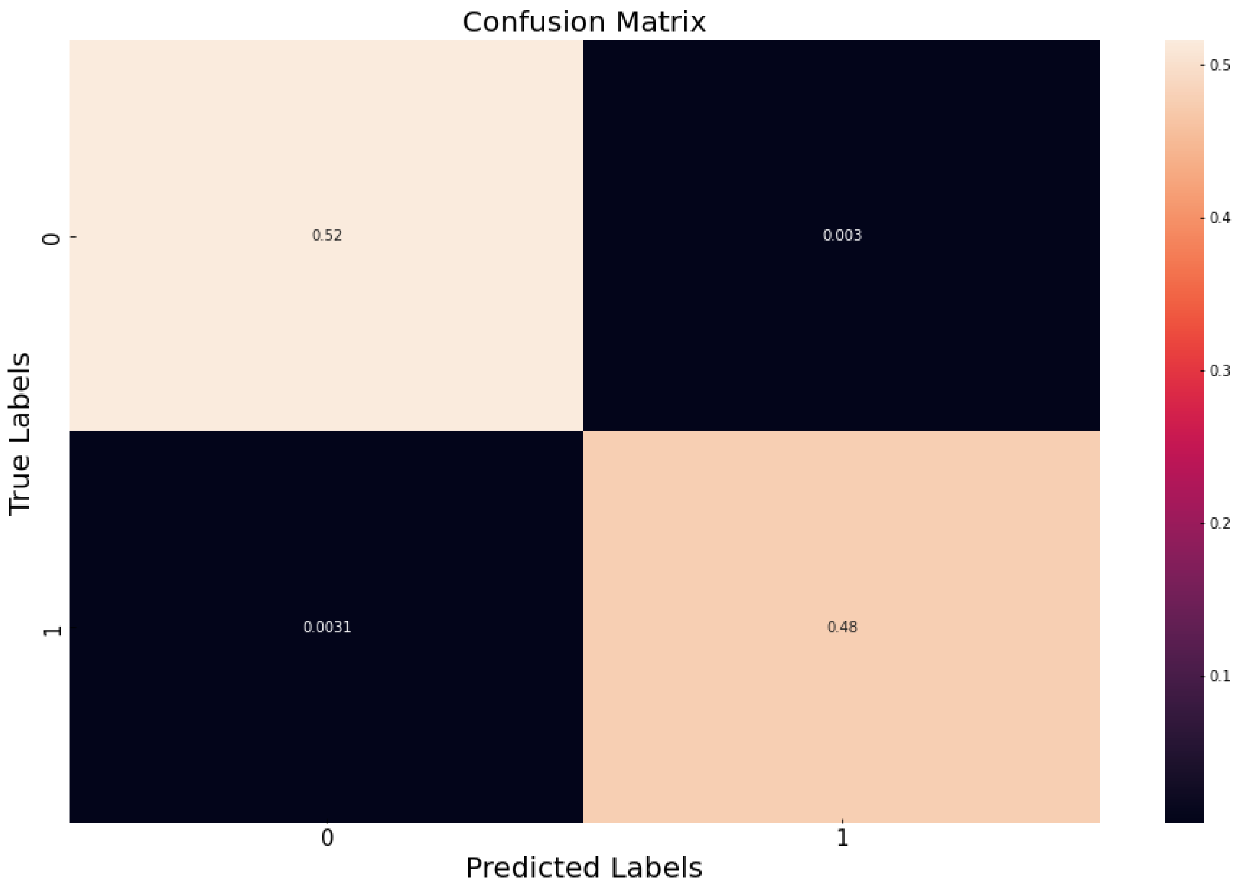


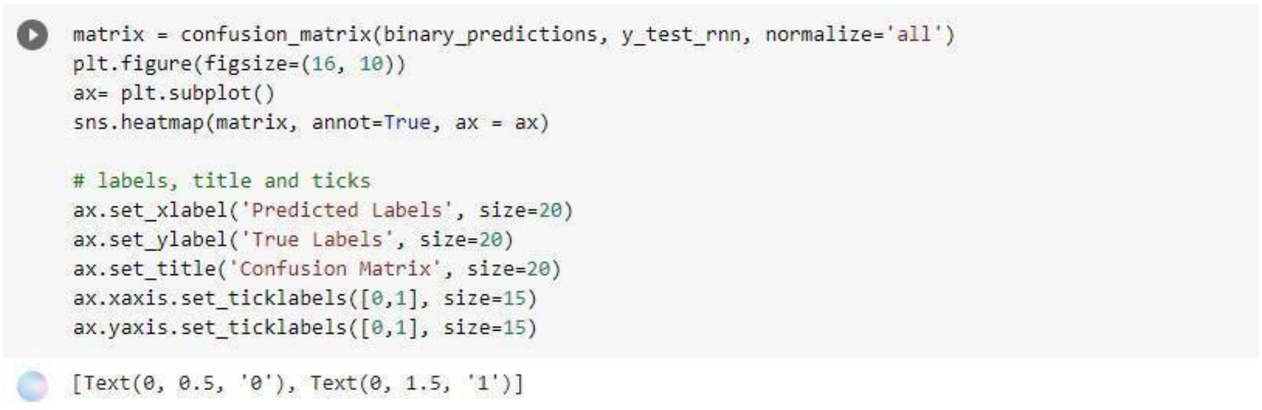
```
print("RNN: PERFORMANCE METRICS\n\n")

accuracy_rnn = accuracy_score(y_test_rnn, binary_predictions)
print("Accuracy: %.2f%%" % (accuracy_rnn * 100.0))

precision_rnn = precision_score(y_test_rnn, binary_predictions, average=None)
print("Precision: %.2f%%" % (precision_rnn[1] * 100.0))

f1score_rnn = f1_score(y_test_rnn, binary_predictions, average=None)
print("F1 Score: %.2f%%" % (f1score_rnn[1] * 100.0))

recall_rnn = recall_score(y_test_rnn, binary_predictions, average=None)
print("Recall: %.2f%%" % (recall_rnn[1] * 100.0))
```



```
RNN: PERFORMANCE METRICS


Accuracy: 99.24%
Precision: 99.49%
F1 Score: 99.21%
Recall: 98.93%
```

We also plot a confusion matrix to see how well has our modelclassified the records.



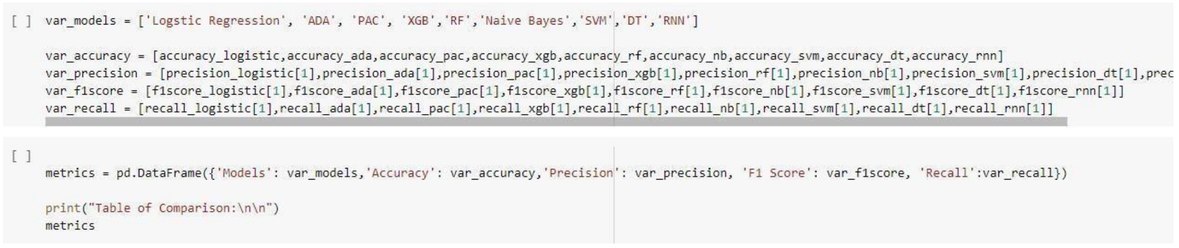


```
matrix = confusion_matrix(binary_predictions, y_test_rnn, normalize='all')
plt.figure(figsize=(16, 10))
ax= plt.subplot()
sns.heatmap(matrix, annot=True, ax = ax)

# labels, title and ticks
ax.set_xlabel('Predicted Labels', size=20)
ax.set_ylabel('True Labels', size=20)
ax.set_title('Confusion Matrix', size=20)
ax.xaxis.set_ticklabels([0,1], size=15)
ax.yaxis.set_ticklabels([0,1], size=15)
```

```
[Text(0, 0.5, '0'), Text(0, 1.5, '1')]
```

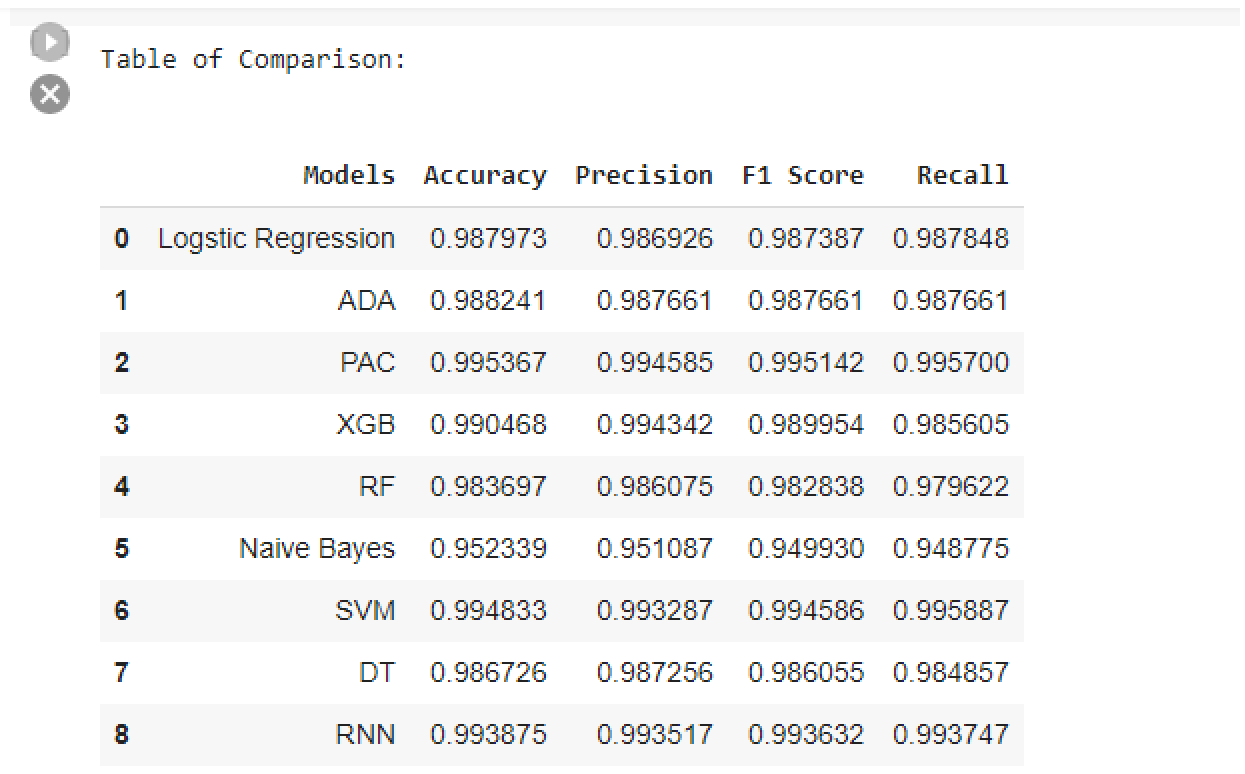
Comparing all the Models

We create separate lists to store the performance metrics of eachmodel and create a dataframe for it.



```
[ ] var_models = ['Logstic Regression', 'ADA', 'PAC', 'XGB','RF','Naive Bayes','SVM','DT','RNN']

    var_accuracy = [accuracy_logistic,accuracy_ada,accuracy_pac,accuracy_xgb,accuracy_rf,accuracy_nb,accuracy_svm,accuracy_dt,accuracy_rnn]
    var_precision = [precision_logistic[1],precision_ada[1],precision_pac[1],precision_xgb[1],precision_rf[1],precision_nb[1],precision_svm[1],precision_dt[1],prec
    var_f1score = [f1score_logistic[1],f1score_ada[1],f1score_pac[1],f1score_xgb[1],f1score_rf[1],f1score_nb[1],f1score_svm[1],f1score_dt[1],f1score_rnn[1]]
    var_recall = [recall_logistic[1],recall_ada[1],recall_pac[1],recall_xgb[1],recall_rf[1],recall_nb[1],recall_svm[1],recall_dt[1],recall_rnn[1]]
```

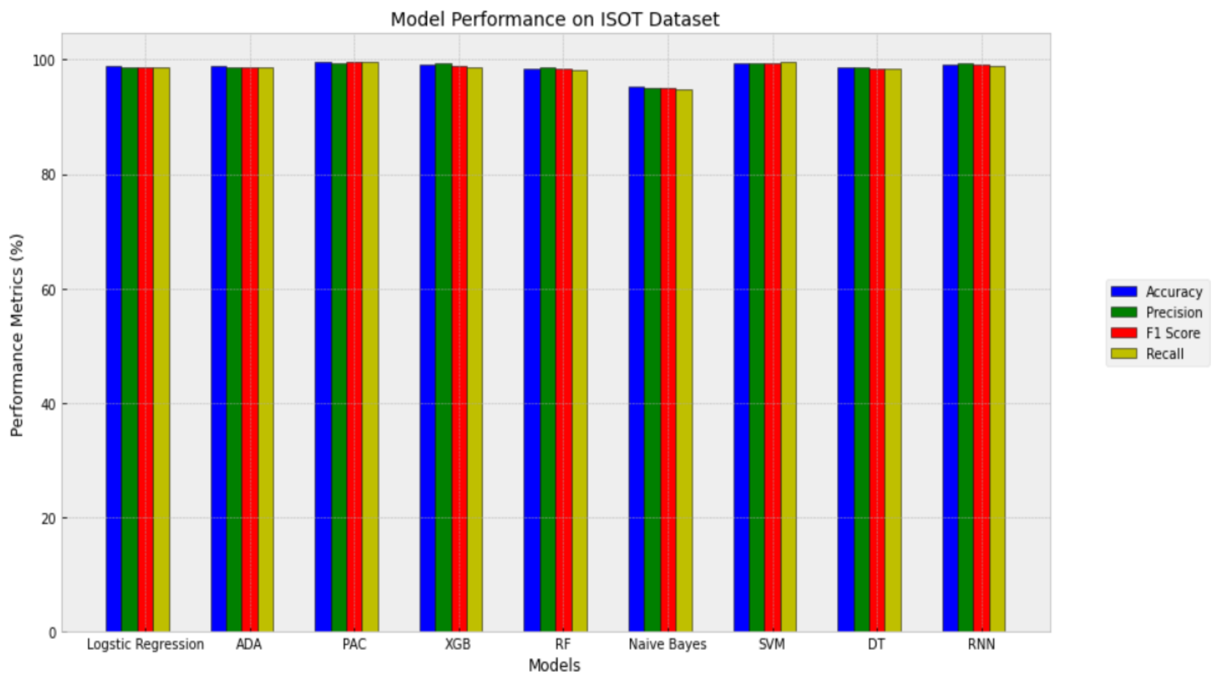
```
[ ]
    metrics = pd.DataFrame({'Models': var_models,'Accuracy': var_accuracy,'Precision': var_precision, 'F1 Score': var_f1score, 'Recall':var_recall})

    print("Table of Comparison:\n\n")
    metrics
```



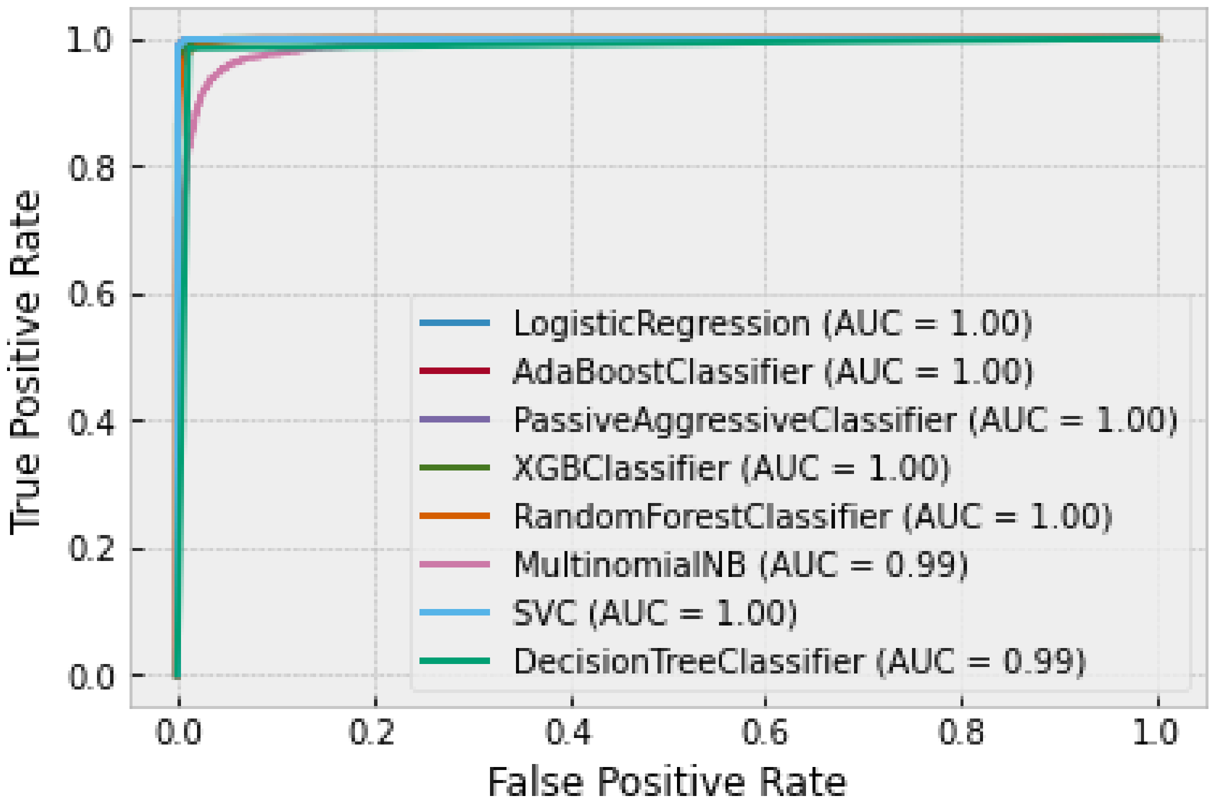
```
Table of Comparison:
```

	Models	Accuracy	Precision	F1 Score	Recall
0	Logstic Regression	0.987973	0.986926	0.987387	0.987848
1	ADA	0.988241	0.987661	0.987661	0.987661
2	PAC	0.995367	0.994585	0.995142	0.995700
3	XGB	0.990468	0.994342	0.989954	0.985605
4	RF	0.983697	0.986075	0.982838	0.979622
5	Naive Bayes	0.952339	0.951087	0.949930	0.948775
6	SVM	0.994833	0.993287	0.994586	0.995887
7	DT	0.986726	0.987256	0.986055	0.984857
8	RNN	0.993875	0.993517	0.993632	0.993747

We plot a bar graph to compare the performance metrics of all themodels on the ISOT dataset.



We also plot the ROC curve to better understand the performance ofeach model. We see that SVM is ideally the best classifier for our dataset.



CHAPTER 5

CONCLUSION AND REFERENCES

5.1 CONCLUSION

Fake news is the difficult problem because it is the rumors which it is too hard to identify the fact in contents. The lie‟s content was used into the corpus that has effected to the failure of IBM‟s Watson prototype testing in the late 2016. Motivation of the proliferation the truth and fake news requiring strenuous effort to detection. It will be used as fact checking to solve the fake news detection problem. Detection of fake news online is important in today's society as fresh news content is rapidly being produced as a result of the abundance of available technology In the discourse of not being able to detect fake news, the world would no longer hold value in truth. Fake news paves the way for deceiving others and promoting ideologies. These people who produce the wrong information benefit by earning money with the number of interactions on their publications. Spreading disinformation holds various intentions, in particular, to gain favor in political elections, for business and products, done out of spite or revenge. Humans can be gullible and fake news is challenging to differentiate from the normal news. Hence by this project, identifying fake news could help in guiding people through the right path.

We downloaded the ISOT dataset from https://www.uvic.ca/ecs/ece/isot/datasets/index.php and uploaded it to our drive, and then loaded it and preprocessed it using various NLP algorithms like tokenization, stop words removal, lemmatization and stemming. We vectorized the text documents using count vectorizer and tf-idf vectorizer. After preprocessing, we split the data into testing and training and we built nine modelsusing nine different classification algorithms and used the predictions to calculatethe performance metrics. The details of each are given below:

Sr	**Models**	**Accuracy**	**Precision**	**F1 Score**	**Recall**
1	Logistic Regression	0.987973	0.986926	0.987387	0.987848
	ADABoost Classifier	0.988241	0.987661	0.987661	0.987661
3	Passive Aggressive Classifier	0.995367	0.994585	0.995142	0.995700
	XG Boost	0.990468	0.994342	0.989954	0.985605
5	Random Forest	0.983697	0.986075	0.982838	0.979622
6	Naive Bayes	0.952339	0.951087	0.949930	0.948775
7	SVM	0.994833	0.993287	0.994586	0.995887
	Decision Tree	0.986726	0.987256	0.986055	0.984857
9	RNN	0.993875	0.993517	0.993632	0.993747

From the ROC curve and the bar plot which compares the performance of all the models, we conclude that the SVM (accuracy-99.48%, precision - 99.33%, f1 score-99.46%, recall-99.59%) is the best algorithm on our ISOT dataset for the task of fake news detection and classification.

5.2 REFERENCES

- Majed AlRubaian, Muhammad Al-Qurishi, Mabrook Al-Rakhami, Sk Md Mizanur Rahman, and Atif Alamri. 2015. A Multistage Credibility Analysis Model for Microblogs. In Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015 (ASONAM '15), Jian Pei, Fabrizio Silvestri, and Jie Tang (Eds.). ACM, New York, NY, USA, 1434-1440. DOI: http://dx.doi.org/10.1145/2808797.2810065
- Majed Alrubaian, Muhammad Al-Qurishi, A Credibility Analysis System for Assessing

  Information on Twitter, IEEE Transactions on Dependable and Secure Computing, 1-14.

  DOI : http://dx.doi.org/ 10.1109/TDSC.2016.2602338
- Manish Gupta, Peixiang Zhao, Jiawei Han, 2012. Evaluating Event Credibility on

  Twitter, Proceedings of the 2012 SIAM International Conference on Data Mining,

  153-164 , DOI: http://epubs.siam.org/doi/abs/10.1137/ 1.9781611972825.14
- Krzysztof Lorek, Jacek Suehiro-Wici´nski, Micha l Jankowski-Lorek, Amit Gupta,

Automated credibility assessment on twitter, Computer Science, 2015, Vol.16(2), 157-168, DOI: http://dx.doi.org/10.7494/csci.2015.16.2.157

- Ballouli, Rim El, Wassim El-Hajj, Ahmad Ghandour, Shady Elbassuoni, Hazem

  M. Hajj and Khaled Bashir Shaban. 2017. "CAT: Credibility Analysis of Arabic Content on Twitter." WANLP@EACL (2017).
- Alina Campan, Alfredo Cuzzocrea, Traian Marius Truta, 2017. Fighting Fake News Spread in Online Social Networks: Actual Trends and Future Research Directions, IEEE International Conference on Big Data (BIGDATA), 4453-4457
- Carlos Castillo, Marcelo Mendoza, and Barbara Poblete 2011. Information credibility on twitter. In Proceedings of the 20th international conference on World wide web (WWW '11). ACM, New York, NY, USA, 675-684. DOI: http://dx.doi.org/10.1145/1963405.1963500
- Muhammad Abdul-Mageed, Mona Diab, Sandra Kübler,2014. SAMAR: Subjectivity and sentiment analysis for Arabic social media, Computer Speech & Language, Elsevier BV, ISSN: 0885-2308, V.28, Issue: 20-37, DOI:

  http://dx.doi.org/10.1016/j.csl.2013.03.001
- Niall J. Conroy, Victoria L. Rubin, and Yimin Chen. 2015. Automatic deception detection: methods for finding fake news. In Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community (ASIST '15). American Society for Information Science, Silver Springs, MD, USA , Article 82 , 4 pages.
- Granik, M., & Mesyura, V 2017. Fake news detection using naive Bayes classifier. 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON), 900-903.
- Hertz, J., Palmer, R.G., Krogh. A.S. 1990 Introduction to the theory of neural computation, Perseus Books. ISBN 0-201-51560-1
- Thandar M., Usanavasin S. 2015 Measuring Opinion Credibility in

Twitter. In: Unger H., Meesad P., Boonkrong S. (eds) Recent Advances in Information and Communication Technology 2015. Advances in Intelligent Systems and Computing, vol 361. Springer, Cham.