FLIP ROBO

FAKE NEWS DETECTION PROJECT

Submitted by:
Rahul Sharma

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. I am also grateful to my SME Miss Rashi Mathur for her constant guidance and support.

Some of the reference sources are as follows:

- Internet
- Coding Ninjas
- Medium.com
- Analytics Vidhya
- StackOverflow

TABLE OF CONTENTS

INTRODUCTION

BUSINESS PROBLEM FRAMING

News media has become a channel to pass on the information of what's happening in the world to the people living. Often people perceive whatever conveyed in the news to be true. There were circumstances where even the news channels acknowledged that their news is not true as they wrote. But some news has a significant impact not only on the people or government but also on the economy. One news can shift the curves up and down depending on the emotions of people and political situation.

It is important to identify the fake news from the real true news. The problem has been taken over and resolved with the help of Natural Language Processing tools which help us identify fake or true news based on historical data. The news is now in safe hands!

CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

The authenticity of Information has become a longstanding issue affecting businesses and society, both for printed and digital media. On social networks, the reach and effects of information spread occur at such a fast pace and so amplified that distorted, inaccurate, or false information acquires a tremendous potential to cause real-world impacts, within minutes, for millions of users. Recently, several public concerns about this problem and some approaches to mitigate the problem were expressed.

The sensationalism of not-so-accurate eye-catching and intriguing headlines aimed at retaining the attention of audiences to sell information has persisted all throughout the history of all kinds of information broadcast. On social networking websites, the reach and effects of information spread are however significantly amplified and occur at such a fast pace, that distorted, inaccurate, or false information acquires a tremendous potential to cause real impacts, within minutes, for millions of users.

REVIEW OF LITERATURE

Fake news is not a new concept. Before the era of digital technology, it was spread through mainly yellow journalism with a focus on sensational news such as crime, gossip, disasters and satirical news. With the widespread dissemination of information via digital media platforms, it is of utmost importance for individuals and societies to be able to judge the credibility of it. Fake news is not a recent concept, but it is a commonly occurring phenomenon in current times. The consequence of fake news can range from being merely annoying to influencing and misleading societies or even nations. A variety of approaches exist to identify fake news

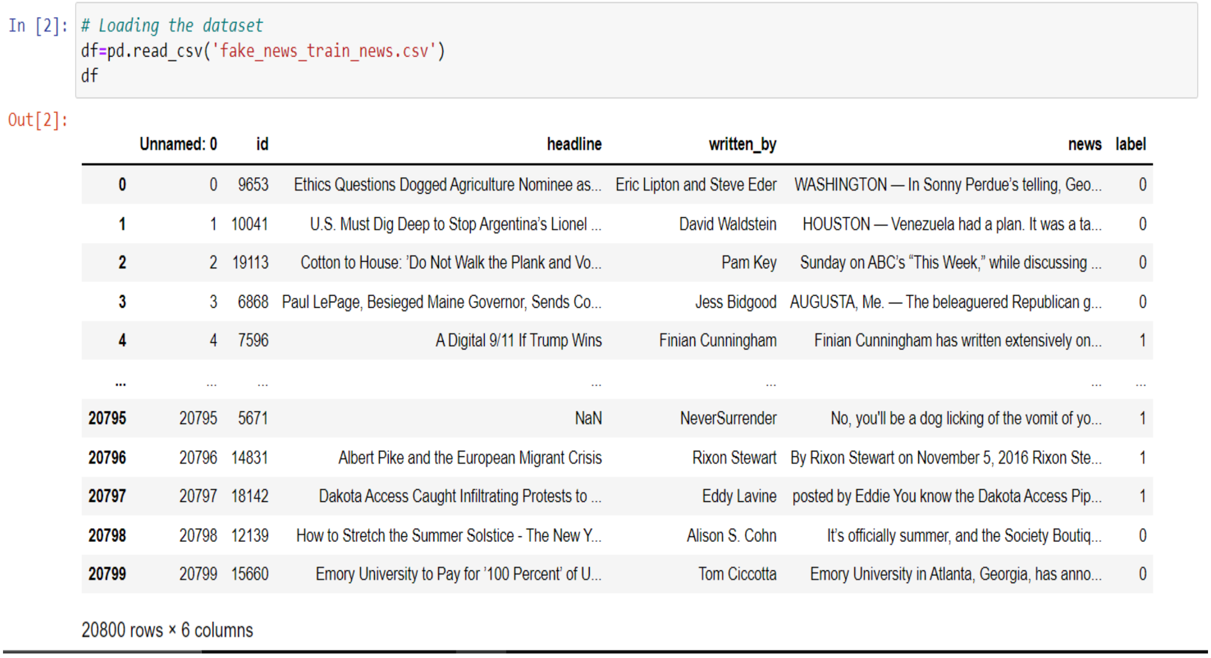
ANALYTICAL PROBLEM FRAMING

MATHEMATICAL/ ANALYTICAL MODELING OF THE PROBLEM

- The dataset provided here has a shape of (20800, 6). Which means it has 20800 rows and 6 columns?
- The target or the dependent variable named "Label" has two distinct values 0 and 1. Where 0 represents the news that is not fake or authentic while 1 represents the category of fake news. As the target column „Label" is giving binary outputs and all the independent variables has text so it is clear that it is a supervised machine learning problem where we can use, we can use the techniques of NLP and classification-based algorithms of Machine learning.
- Here we will use NLP techniques like word tokenization, lemmatization and tfidf vectorizer then those processed data will be used to create the best model using various classification based supervised machine learning algorithms like Logistic Regression, Multinomial NB, Random Forest Classifier etc
- The dataset contains null value.
- Train test is the best way to get the solution of these kinds of problems as that is the easiest and the efficient way to solve this problem.

DATA SOURCES AND THEIR FORMATS

- The data is provided to us from our client database. The sample data is in .csv format
- The sample data for reference is shown below.



```
In [2]: # Loading the dataset
        df=pd.read_csv('fake_news_train_news.csv')
        df
```

Out[2]:

	Unnamed: 0	id	headline	written_by	news	label
0	0	9653	Ethics Questions Dogged Agriculture Nominee as...	Eric Lipton and Steve Eder	WASHINGTON — In Sonny Perdue's telling, Geo...	0
1	1	10041	U.S. Must Dig Deep to Stop Argentina's Lionel ...	David Waldstein	HOUSTON — Venezuela had a plan. It was a ta...	0
2	2	19113	Cotton to House: 'Do Not Walk the Plank and Vo...	Pam Key	Sunday on ABC's "This Week," while discussing ...	0
3	3	6868	Paul LePage, Besieged Maine Governor, Sends Co...	Jess Bidgood	AUGUSTA, Me. — The beleaguered Republican g...	0
4	4	7596	A Digital 9/11 If Trump Wins	Finian Cunningham	Finian Cunningham has written extensively on...	1
...	...	...	...	...	...	...
20795	20795	5671	NaN	NeverSurrender	No, you'll be a dog licking of the vomit of yo...	1
20796	20796	14831	Albert Pike and the European Migrant Crisis	Rixon Stewart	By Rixon Stewart on November 5, 2016 Rixon Ste...	1
20797	20797	18142	Dakota Access Caught Infiltrating Protests to ...	Eddy Lavine	posted by Eddie You know the Dakota Access Pip...	1
20798	20798	12139	How to Stretch the Summer Solstice - The New Y...	Alison S. Cohn	It's officially summer, and the Society Boutiq...	0
20799	20799	15660	Emory University to Pay for '100 Percent' of U...	Tom Ciccotta	Emory University in Atlanta, Georgia, has anno...	0

20800 rows × 6 columns

Dataset description

There are 6 columns in the dataset provided:

The description of each of the column is given below:

- "id": Unique id of each news article
- "headline": It is the title of the news.
- "news": It contains the full text of the news article
- "Unnamed:0": It is a serial number
- "written_by": It represents the author of the news article
- "label": It tells whether the news is fake (1) or not fake (0).

Identification of possible problem-solving approaches (methods)

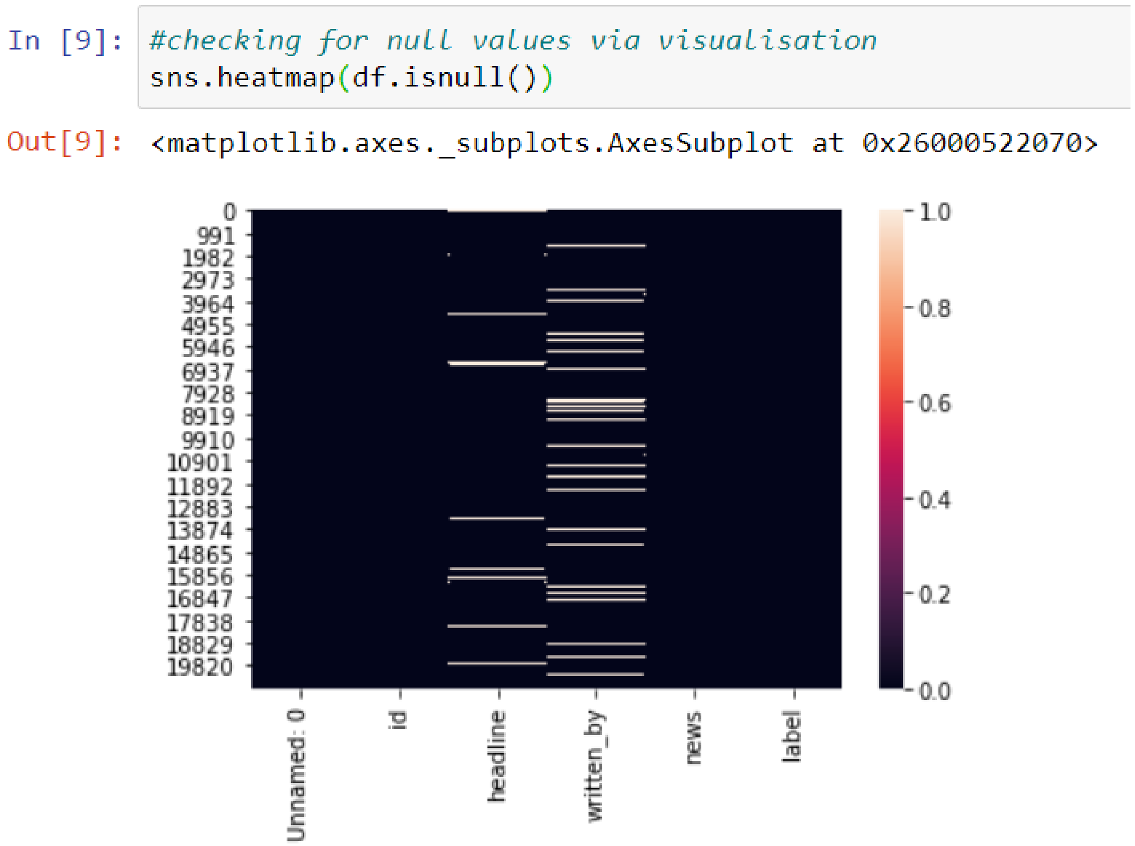
We have used the following process for problem-solving:

1. Data Preprocessing
2. Building a word dictionary
3. Feature extraction
4. Training classifiers
5. Testing
6. Performance evaluation using multiple metrics

DATA PREPROCESSING DONE

Data usually comes from a variety of source & is often inconsistent, inaccurate. Data preprocessing helps to enhance the quality of data and make it ready for the various ML model. We have applied various methods for data preprocessing methods in this project.

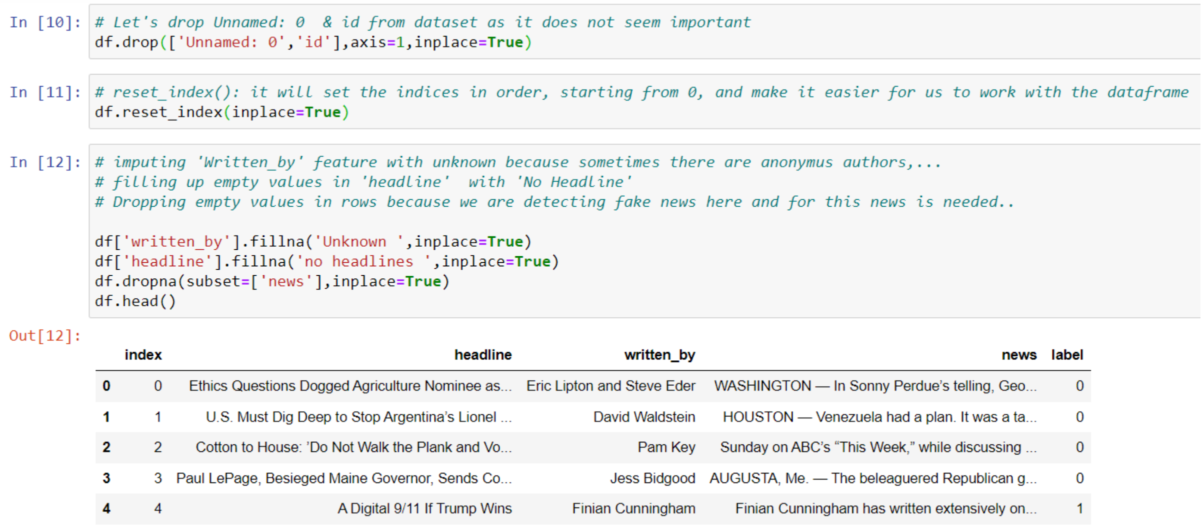
- First, we check shape by using (df. shape)
- Then checked datatype of various features & found that all features are of int type except headline, written_by, news which are of object datatype
- Checking for null values in each column



```
In [9]: #checking for null values via visualisation
        sns.heatmap(df.isnull())

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x26000522070>
```

It clearly shows that null values are present in the dataset, which needs to be removed.

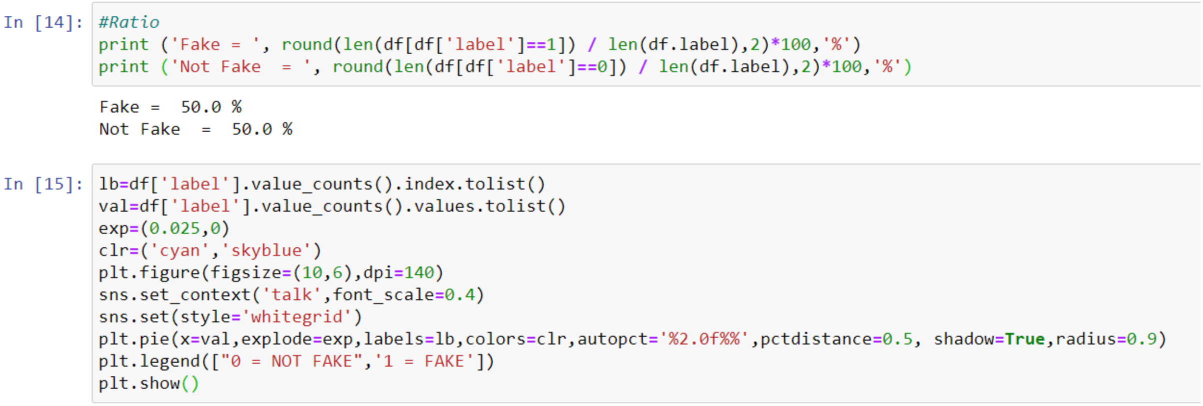
Treating null values



```
In [10]: # Let's drop Unnamed: 0  & id from dataset as it does not seem important
         df.drop(['Unnamed: 0','id'],axis=1,inplace=True)

In [11]: # reset_index(): it will set the indices in order, starting from 0, and make it easier for us to work with the dataframe
         df.reset_index(inplace=True)

In [12]: # imputing 'Written_by' feature with unknown because sometimes there are anonymus authors,...
         # filling up empty values in 'headline'  with 'No Headline'
         # Dropping empty values in rows because we are detecting fake news here and for this news is needed..

         df['written_by'].fillna('Unknown ',inplace=True)
         df['headline'].fillna('no headlines ',inplace=True)
         df.dropna(subset=['news'],inplace=True)
         df.head()

Out[12]:
```

	index	headline	written_by	news	label
0	0	Ethics Questions Dogged Agriculture Nominee as...	Eric Lipton and Steve Eder	WASHINGTON — In Sonny Perdue's telling, Geo...	0
1	1	U.S. Must Dig Deep to Stop Argentina's Lionel ...	David Waldstein	HOUSTON — Venezuela had a plan. It was a ta...	0
2	2	Cotton to House: 'Do Not Walk the Plank and Vo...	Pam Key	Sunday on ABC's "This Week," while discussing ...	0
3	3	Paul LePage, Besieged Maine Governor, Sends Co...	Jess Bidgood	AUGUSTA, Me. — The beleaguered Republican g...	0
4	4	A Digital 9/11 If Trump Wins	Finian Cunningham	Finian Cunningham has written extensively on...	1

Checking distribution of fake and real news



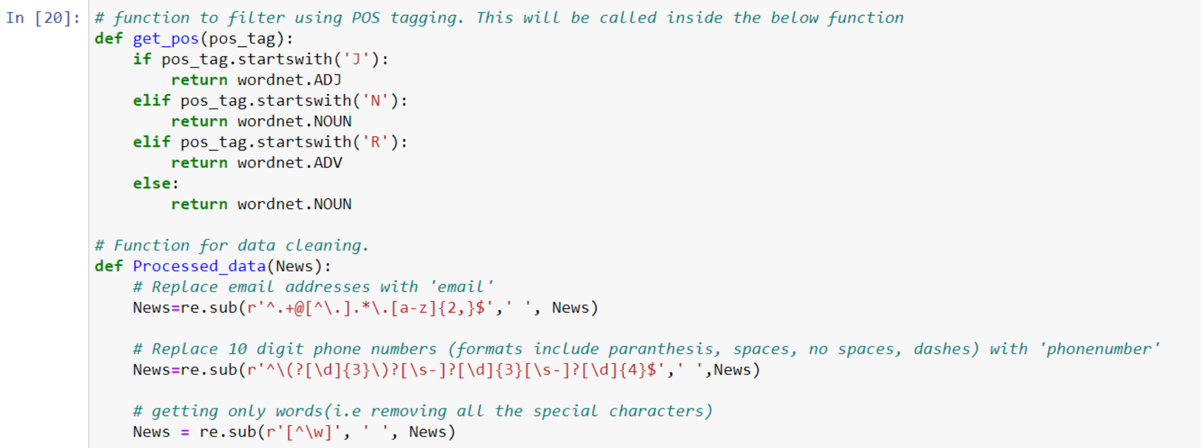
```
In [14]: #Ratio
         print ('Fake = ', round(len(df[df['label']==1]) / len(df.label),2)*100,'%')
         print ('Not Fake  = ', round(len(df[df['label']==0]) / len(df.label),2)*100,'%')

         Fake =  50.0 %
         Not Fake  =  50.0 %

In [15]: lb=df['label'].value_counts().index.tolist()
         val=df['label'].value_counts().values.tolist()
         exp=(0.025,0)
         clr=('cyan','skyblue')
         plt.figure(figsize=(10,6),dpi=140)
         sns.set_context('talk',font_scale=0.4)
         sns.set(style='whitegrid')
         plt.pie(x=val,explode=exp,labels=lb,colors=clr,autopct='%2.0f%%',pctdistance=0.5, shadow=True,radius=0.9)
         plt.legend(["0 = NOT FAKE",'1 = FAKE'])
         plt.show()
```

We see that both news is equally distributed .ie dataset is balanced which is good as it will help our model to classify more accurately, so we should expect a good accuracy score.

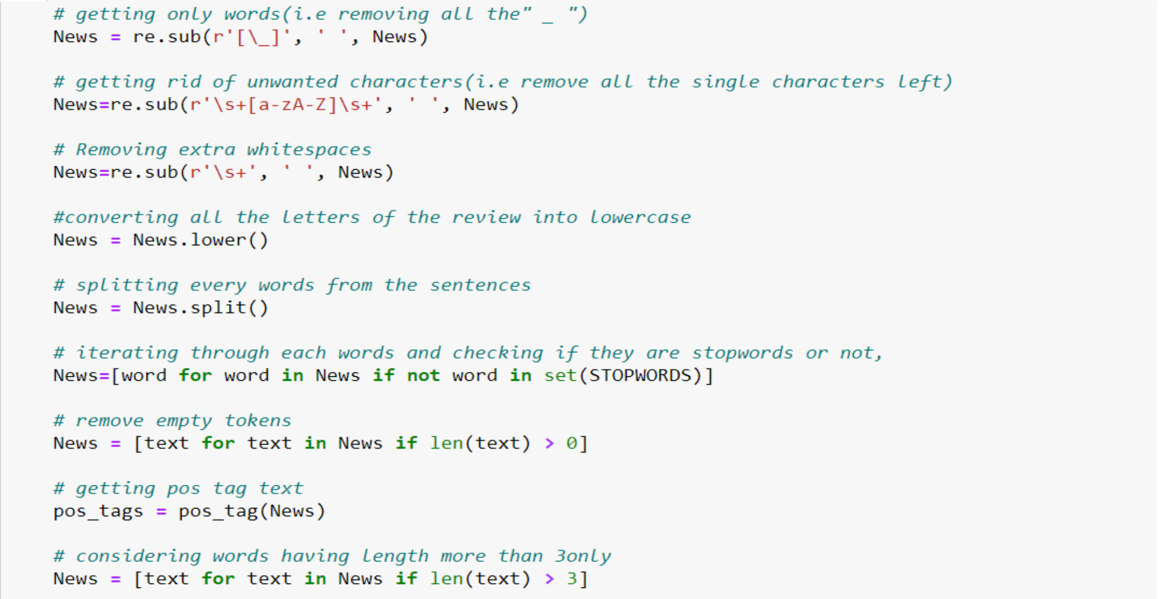
Cleaning the raw data-It involves the deletion of words or special characters that do not add meaning to the text. Important cleaning steps are as follows:

1. Lowering case
2. Handling of special characters
3. Removal of stopwords
4. Handling of hyperlinks

5. Removing leading and trailing white space
6. Replacing URLs with web address
7. Converted words to the most suitable base form by using lemmatization



```
In [20]: # function to filter using POS tagging. This will be called inside the below function
         def get_pos(pos_tag):
             if pos_tag.startswith('J'):
                 return wordnet.ADJ
             elif pos_tag.startswith('N'):
                 return wordnet.NOUN
             elif pos_tag.startswith('R'):
                 return wordnet.ADV
             else:
                 return wordnet.NOUN

         # Function for data cleaning.
         def Processed_data(News):
             # Replace email addresses with 'email'
             News=re.sub(r'^.+@[^\.].*\.[a-z]{2,}$',' ', News)

             # Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
             News=re.sub(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',' ',News)

             # getting only words(i.e removing all the special characters)
             News = re.sub(r'[^\w]', ' ', News)
```



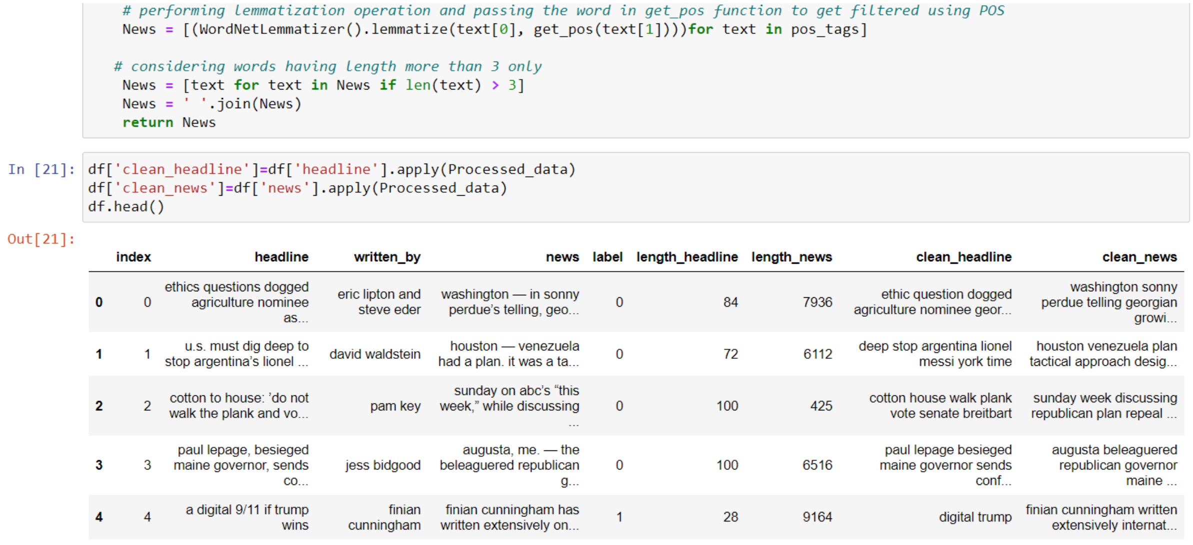
```
    # getting only words(i.e removing all the" _ ")
    News = re.sub(r'[\_]', ' ', News)

    # getting rid of unwanted characters(i.e remove all the single characters left)
    News=re.sub(r'\s+[a-zA-Z]\s+', ' ', News)

    # Removing extra whitespaces
    News=re.sub(r'\s+', ' ', News)

    #converting all the letters of the review into lowercase
    News = News.lower()

    # splitting every words from the sentences
    News = News.split()

    # iterating through each words and checking if they are stopwords or not,
    News=[word for word in News if not word in set(STOPWORDS)]

    # remove empty tokens
    News = [text for text in News if len(text) > 0]

    # getting pos tag text
    pos_tags = pos_tag(News)

    # considering words having length more than 3only
    News = [text for text in News if len(text) > 3]
```

For Data pre-processing we did some data cleaning, where we used WordNet lemmatizer to clean the words and removed special characters using Regexp Tokenizer and filter the words by removing stop words and then used lemmatizers and joined and return the filtered words.

Used TFIDF vectorizer to convert those text into vectors, and split the data and into test and train and trained various Machine learning algorithms.

Adding additional attribute:

To compare the length of headline & news before preprocessing and after preprocessing an addition column was added:

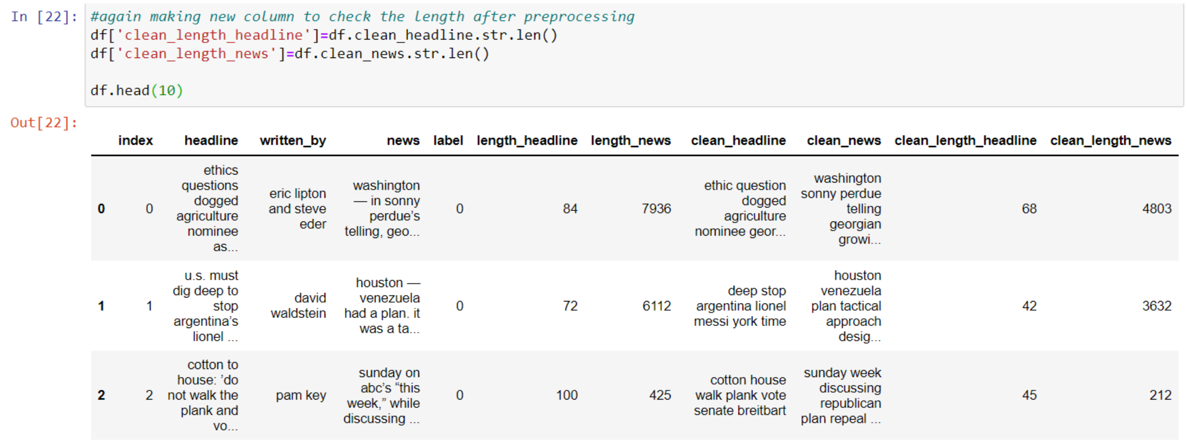


```
    # performing lemmatization operation and passing the word in get_pos function to get filtered using POS
    News = [(WordNetLemmatizer().lemmatize(text[0], get_pos(text[1])))for text in pos_tags]

   # considering words having length more than 3 only
    News = [text for text in News if len(text) > 3]
    News = ' '.join(News)
    return News
```

```
In [21]: df['clean_headline']=df['headline'].apply(Processed_data)
         df['clean_news']=df['news'].apply(Processed_data)
         df.head()
```

Out[21]:

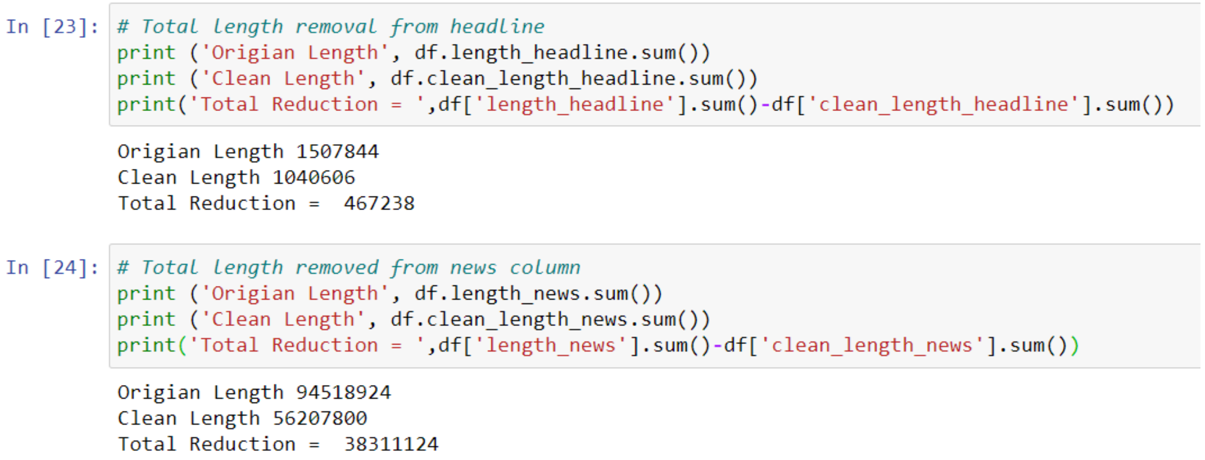
	index	headline	written_by	news	label	length_headline	length_news	clean_headline	clean_news
0	0	ethics questions dogged agriculture nominee as...	eric lipton and steve eder	washington — in sonny perdue's telling, geo...	0	84	7936	ethic question dogged agriculture nominee geor...	washington sonny perdue telling georgian growi...
1	1	u.s. must dig deep to stop argentina's lionel ...	david waldstein	houston — venezuela had a plan. it was a ta...	0	72	6112	deep stop argentina lionel messi york time	houston venezuela plan tactical approach desig...
2	2	cotton to house: 'do not walk the plank and vo...	pam key	sunday on abc's "this week," while discussing ...	0	100	425	cotton house walk plank vote senate breitbart	sunday week discussing republican plan repeal ...
3	3	paul lepage, besieged maine governor, sends co...	jess bidgood	augusta, me. — the beleaguered republican g...	0	100	6516	paul lepage besieged maine governor sends conf...	augusta beleaguered republican governor maine ...
4	4	a digital 9/11 if trump wins	finian cunningham	finian cunningham has written extensively on...	1	28	9164	digital trump	finian cunningham written extensively internat...



```
In [22]: #again making new column to check the length after preprocessing
         df['clean_length_headline']=df.clean_headline.str.len()
         df['clean_length_news']=df.clean_news.str.len()

         df.head(10)
```

Out[22]:

	index	headline	written_by	news	label	length_headline	length_news	clean_headline	clean_news	clean_length_headline	clean_length_news
0	0	ethics questions dogged agriculture nominee as...	eric lipton and steve eder	washington — in sonny perdue's telling, geo...	0	84	7936	ethic question dogged agriculture nominee geor...	washington sonny perdue telling georgian growi...	68	4803
1	1	u.s. must dig deep to stop argentina's lionel ...	david waldstein	houston — venezuela had a plan. it was a ta...	0	72	6112	deep stop argentina lionel messi york time	houston venezuela plan tactical approach desig...	42	3632
2	2	cotton to house: 'do not walk the plank and vo...	pam key	sunday on abc's "this week," while discussing ...	0	100	425	cotton house walk plank vote senate breitbart	sunday week discussing republican plan repeal ...	45	212



```
In [23]: # Total length removal from headline
         print ('Origian Length', df.length_headline.sum())
         print ('Clean Length', df.clean_length_headline.sum())
         print('Total Reduction = ',df['length_headline'].sum()-df['clean_length_headline'].sum())

         Origian Length 1507844
         Clean Length 1040606
         Total Reduction =  467238

In [24]: # Total length removed from news column
         print ('Origian Length', df.length_news.sum())
         print ('Clean Length', df.clean_length_news.sum())
         print('Total Reduction = ',df['length_news'].sum()-df['clean_length_news'].sum())

         Origian Length 94518924
         Clean Length 56207800
         Total Reduction =  38311124
```

After executing all these steps it was found that all the words & special characters were removed from the dataset which was of no use and consuming memory

DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS

For this data's input and output logic, we will analyse words frequency for each label, so that we can get the most frequent words that were used in different features.

SOFTWARE REQUIREMENTS AND TOOLS USED

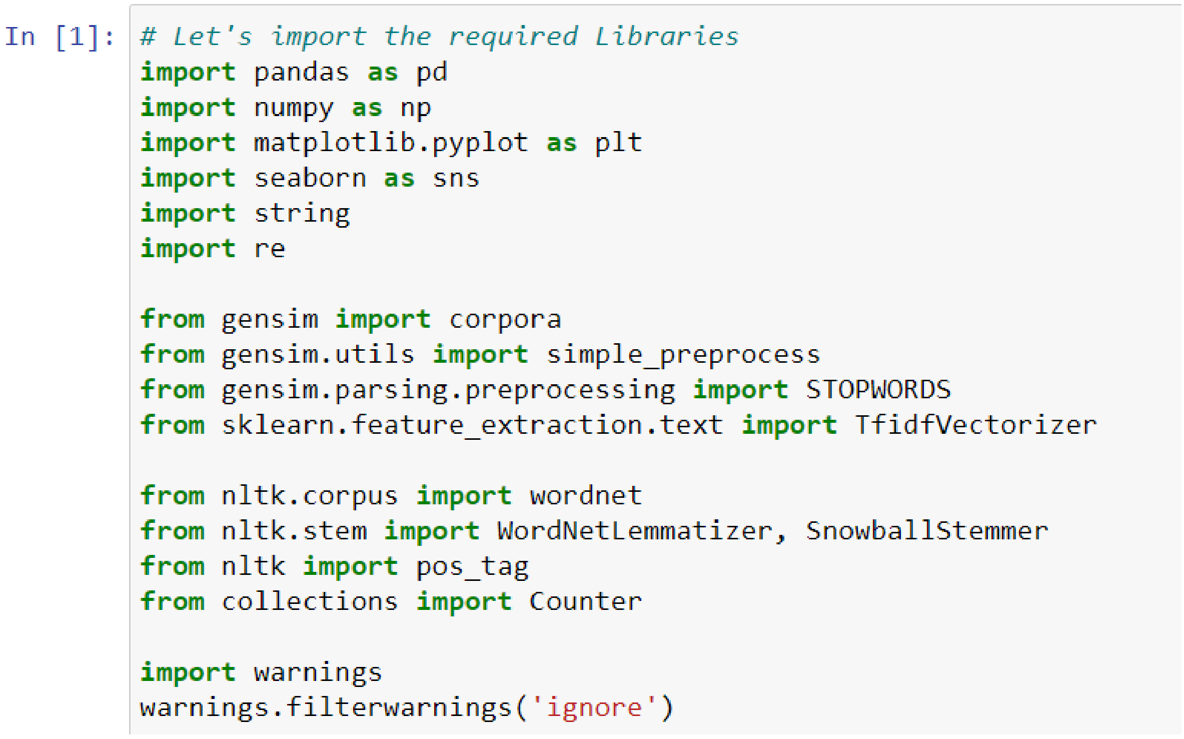
SOFTWARE:

Jupyter Notebook (Anaconda 3) – Python 3.8.11

Microsoft Excel 2019

LIBRARIES:

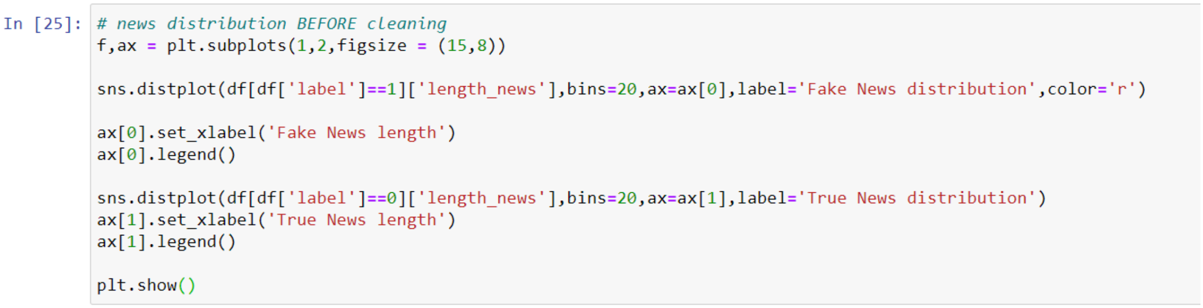
- Pandas: To read the Data file in form of data.
- Matplotlib: This library is typically used to plot the figures for better visualisation of data.
- Seaborn: A advanced version of Matplotlib
- Scikit Learn: This is the most important library for Machine Learning since it contains various Machine Learning Algorithms which are used in this project. Scikit Learn also contains Preprocessing library which is used in data preprocessing. Apart from this, it contains a very useful joblib library for serialization purpose using which the final model has been saved in this project.
- NLTK: Natural language took kit is one of the most used libraries for building NLP projects.



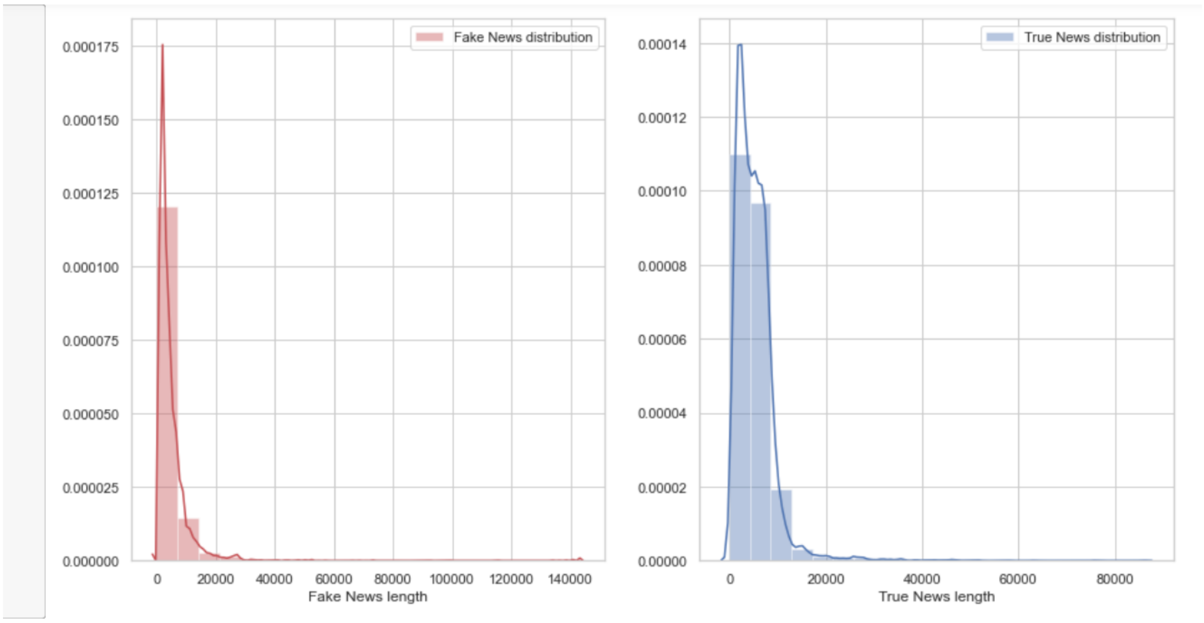
```
In [1]: # Let's import the required Libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import string
        import re

        from gensim import corpora
        from gensim.utils import simple_preprocess
        from gensim.parsing.preprocessing import STOPWORDS
        from sklearn.feature_extraction.text import TfidfVectorizer

        from nltk.corpus import wordnet
        from nltk.stem import WordNetLemmatizer, SnowballStemmer
        from nltk import pos_tag
        from collections import Counter

        import warnings
        warnings.filterwarnings('ignore')
```

Then we have plotted a graph to show the distribution of word count before cleaning and after cleaning

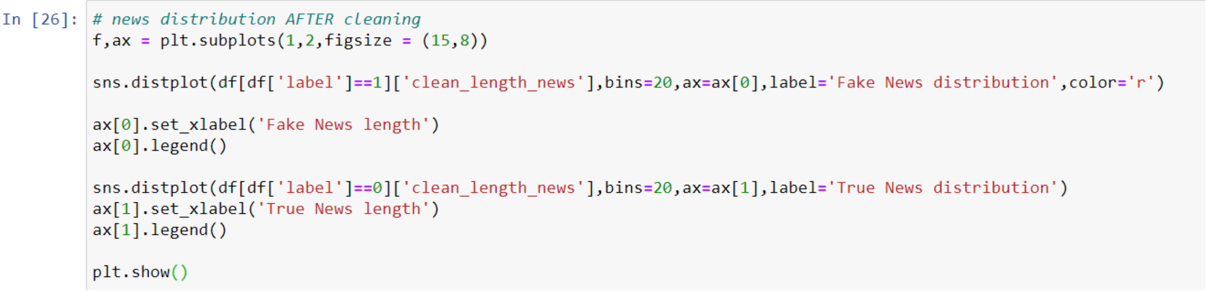
Before cleaning:



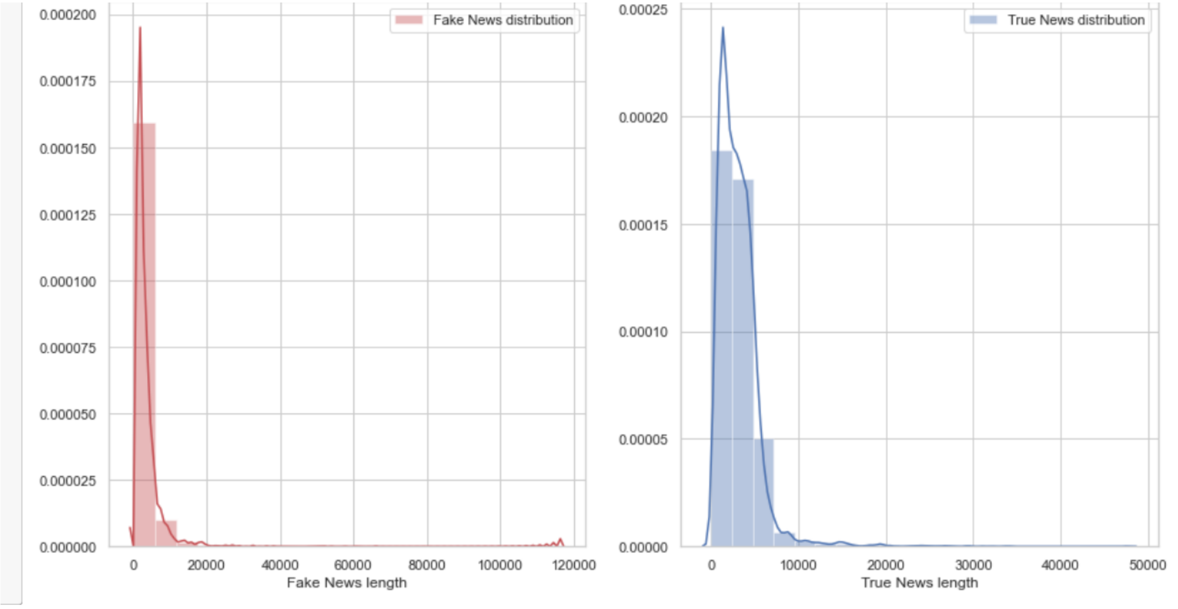
```
In [25]: # news distribution BEFORE cleaning
         f,ax = plt.subplots(1,2,figsize = (15,8))

         sns.distplot(df[df['label']==1]['length_news'],bins=20,ax=ax[0],label='Fake News distribution',color='r')

         ax[0].set_xlabel('Fake News length')
         ax[0].legend()

         sns.distplot(df[df['label']==0]['length_news'],bins=20,ax=ax[1],label='True News distribution')
         ax[1].set_xlabel('True News length')
         ax[1].legend()

         plt.show()
```



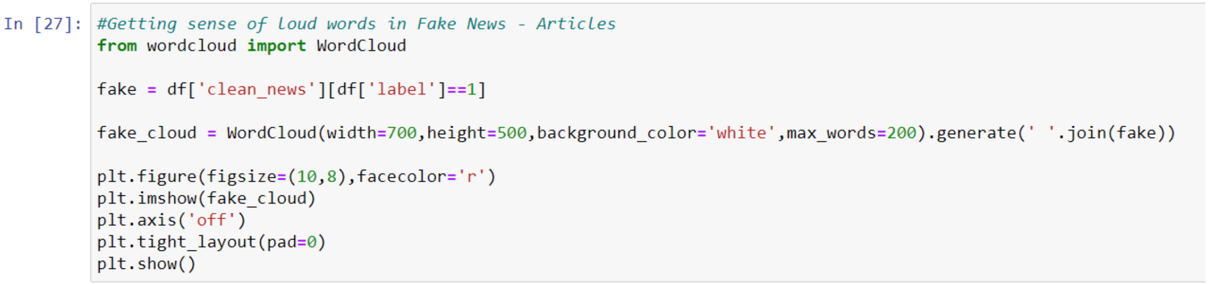
After cleaning



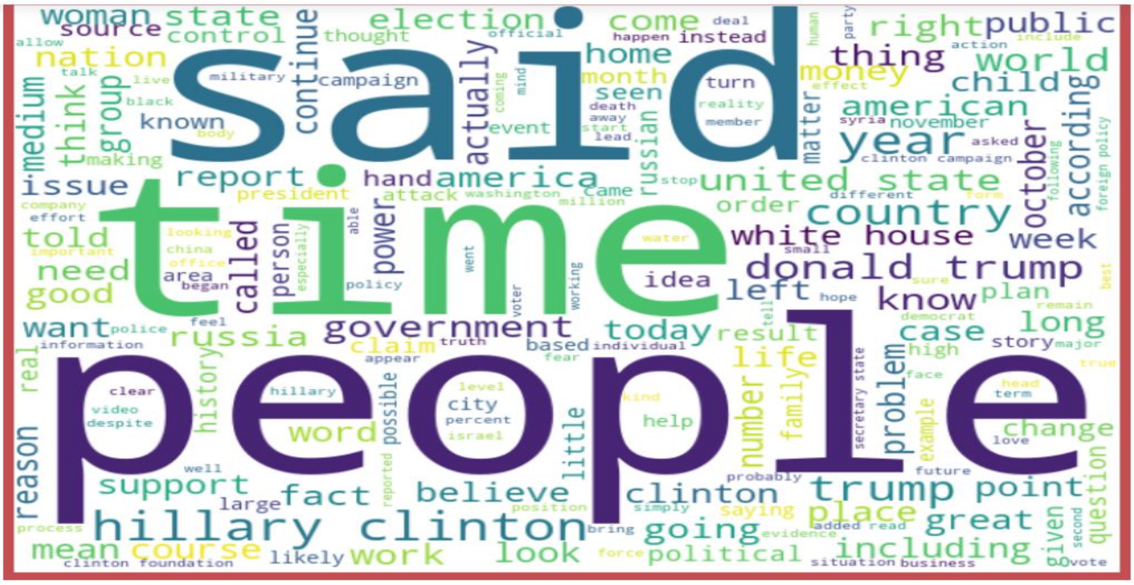
```
In [26]: # news distribution AFTER cleaning
         f,ax = plt.subplots(1,2,figsize = (15,8))

         sns.distplot(df[df['label']==1]['clean_length_news'],bins=20,ax=ax[0],label='Fake News distribution',color='r')

         ax[0].set_xlabel('Fake News length')
         ax[0].legend()

         sns.distplot(df[df['label']==0]['clean_length_news'],bins=20,ax=ax[1],label='True News distribution')
         ax[1].set_xlabel('True News length')
         ax[1].legend()

         plt.show()
```

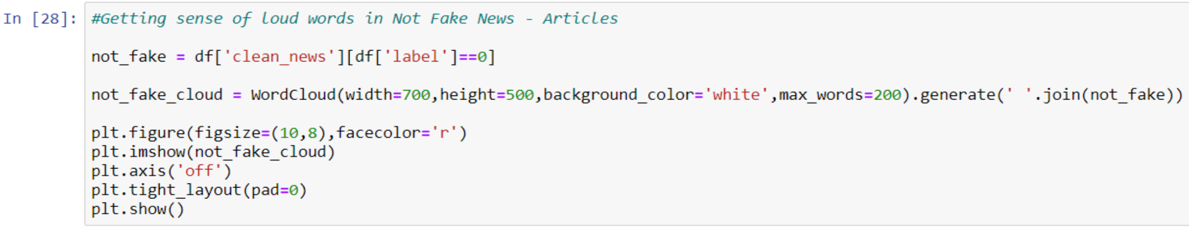


To get a better view of words contained in news. A word dictionary (word cloud) was made showing the words highly occurred in fake and real news for both headline and news column.

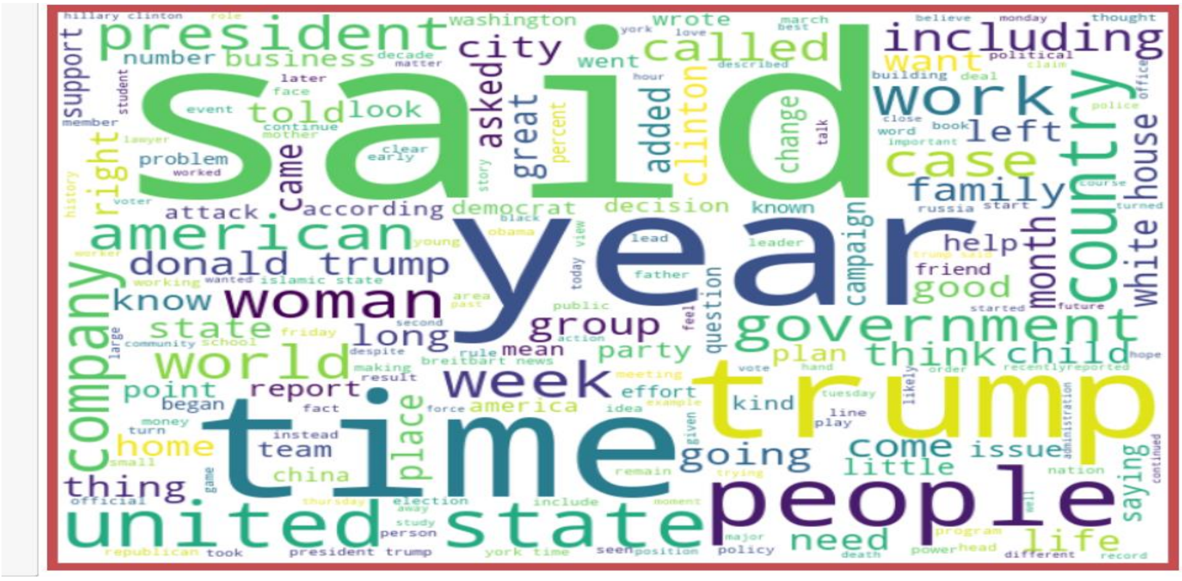


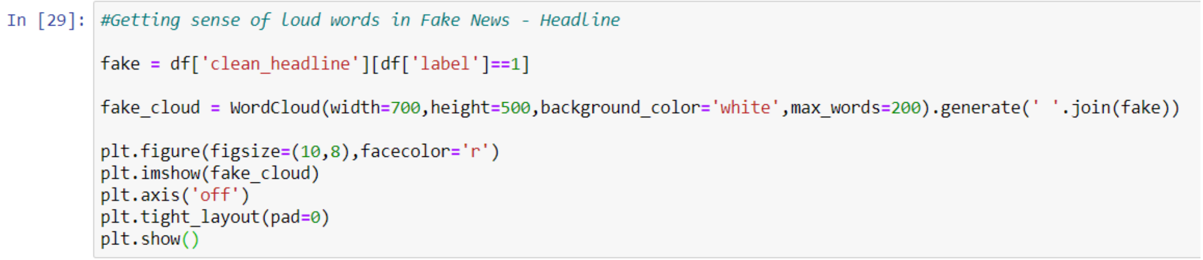
```
In [27]: #Getting sense of loud words in Fake News - Articles
         from wordcloud import WordCloud

         fake = df['clean_news'][df['label']==1]

         fake_cloud = WordCloud(width=700,height=500,background_color='white',max_words=200).generate(' '.join(fake))

         plt.figure(figsize=(10,8),facecolor='r')
         plt.imshow(fake_cloud)
         plt.axis('off')
         plt.tight_layout(pad=0)
         plt.show()
```

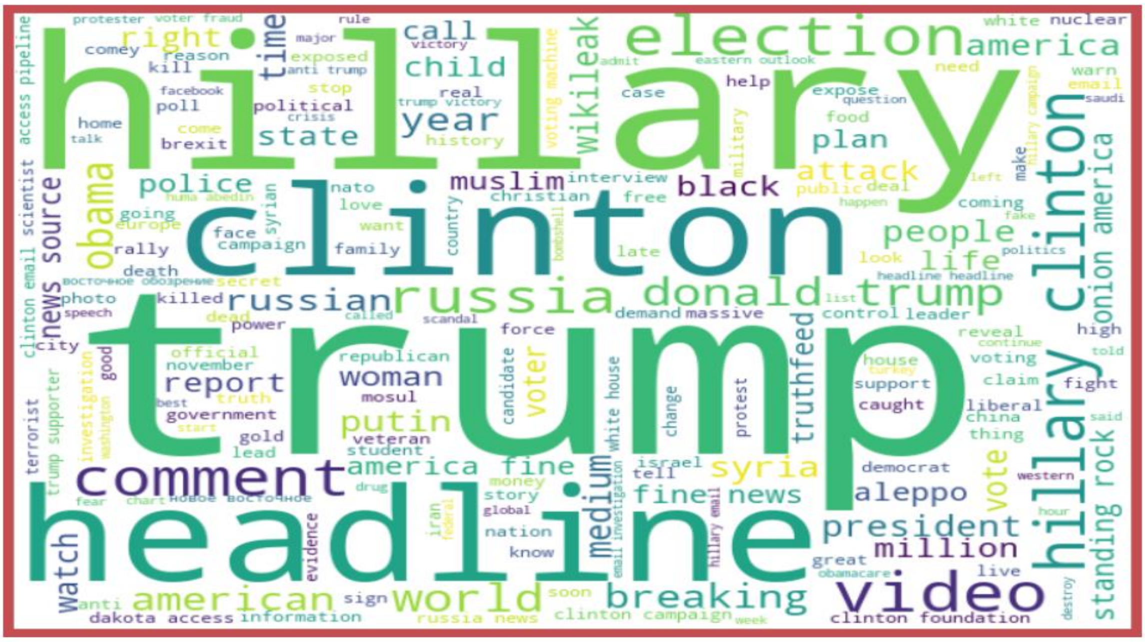


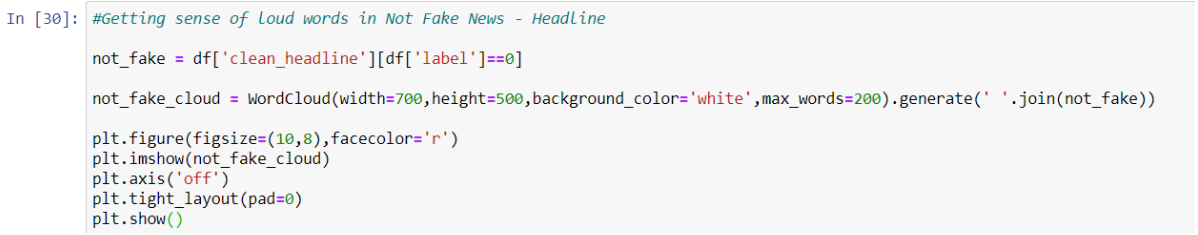


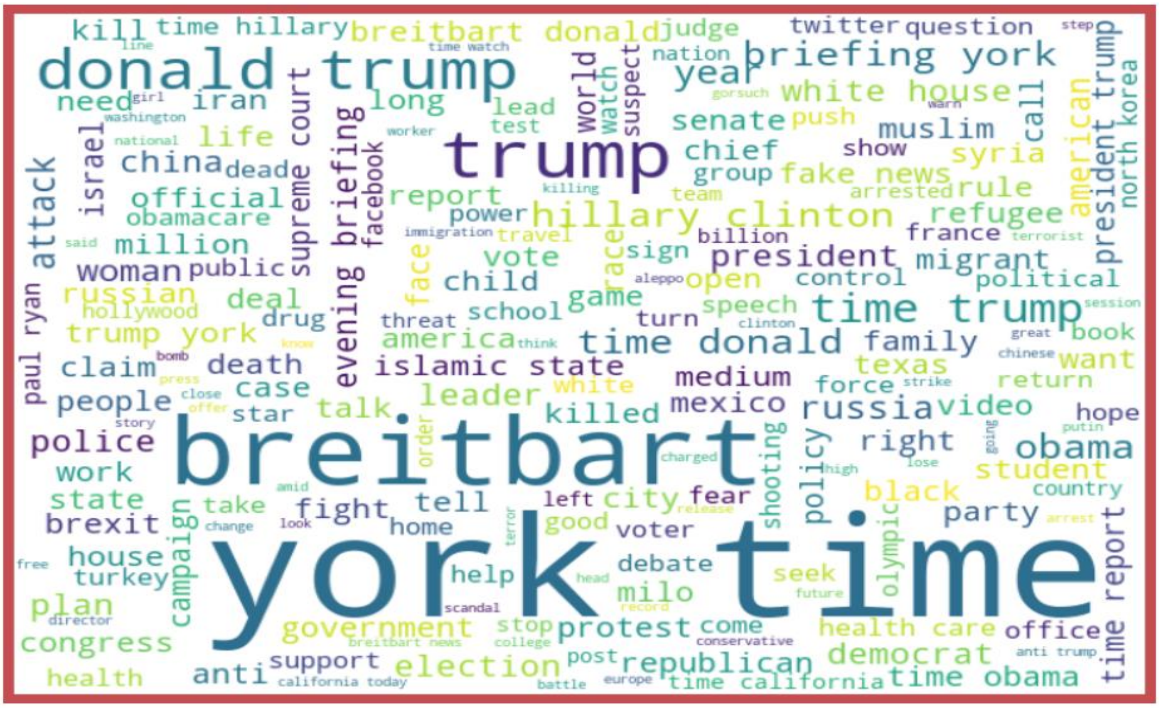
```
In [28]: #Getting sense of loud words in Not Fake News - Articles

         not_fake = df['clean_news'][df['label']==0]

         not_fake_cloud = WordCloud(width=700,height=500,background_color='white',max_words=200).generate(' '.join(not_fake))

         plt.figure(figsize=(10,8),facecolor='r')
         plt.imshow(not_fake_cloud)
         plt.axis('off')
         plt.tight_layout(pad=0)
         plt.show()
```




In [29]:
```
#Getting sense of loud words in Fake News - Headline

fake = df['clean_headline'][df['label']==1]

fake_cloud = WordCloud(width=700,height=500,background_color='white',max_words=200).generate(' '.join(fake))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(fake_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```




In [30]:
```
#Getting sense of loud words in Not Fake News - Headline

not_fake = df['clean_headline'][df['label']==0]

not_fake_cloud = WordCloud(width=700,height=500,background_color='white',max_words=200).generate(' '.join(not_fake))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(not_fake_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



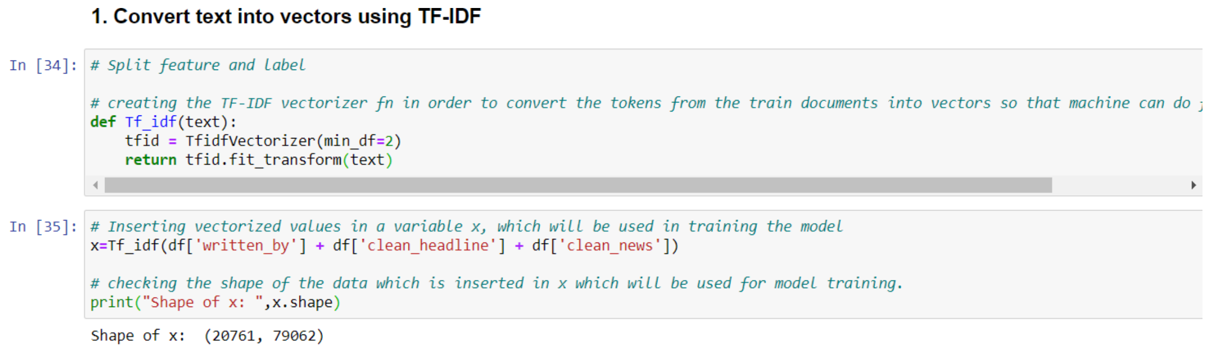
From the above we can see that the most frequent words on both labels and we can observe the words which are leading to fake news are trump, Clinton, prison, November, etc. and words which are leading to real news are said, agriculture, police, questions etc., so we can see that above dataset extensively deals with news around US presidential elections between Trump and Clinton.

MODEL/S DEVELOPMENT AND EVALUATION

IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES (METHODS)

Understanding the problem is the first crucial steps in solving any problem. From the given dataset it can be concluded that it is a binary classification problem. Therefore I run my preprocessed data on 6 classification algorithm.

Training Classifier:
We converted all the text into vectors, using TF-IDF. Then we have split features and label.



1. Convert text into vectors using TF-IDF

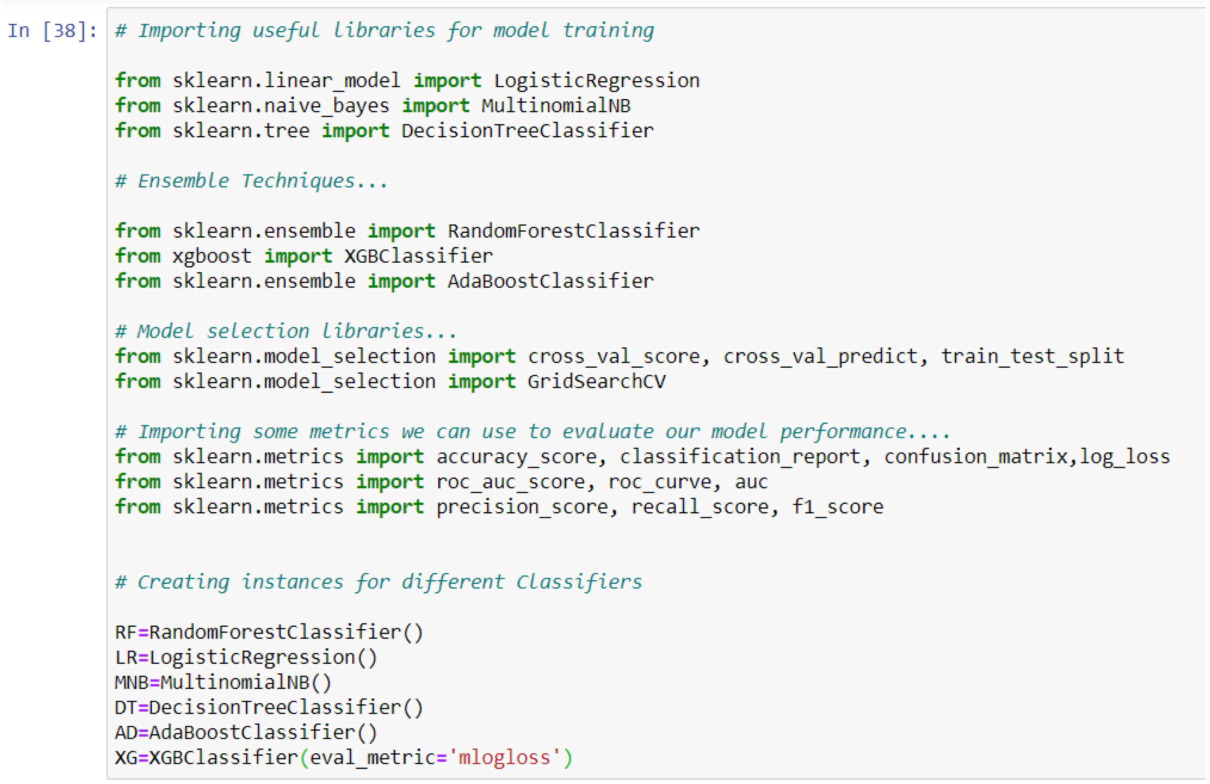
In [34]:
```
# Split feature and label

# creating the TF-IDF vectorizer fn in order to convert the tokens from the train documents into vectors so that machine can do
def Tf_idf(text):
    tfid = TfidfVectorizer(min_df=2)
    return tfid.fit_transform(text)
```

In [35]:
```
# Inserting vectorized values in a variable x, which will be used in training the model
x=Tf_idf(df['written_by'] + df['clean_headline'] + df['clean_news'])

# checking the shape of the data which is inserted in x which will be used for model training.
print("Shape of x: ",x.shape)
```

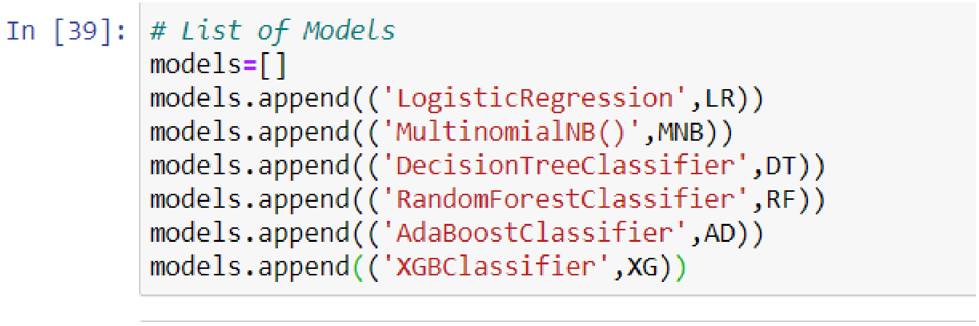
```
Shape of x:  (20761, 79062)
```

TESTING OF IDENTIFIED APPROACHES (ALGORITHMS)

The algorithms we used for the training and testing are as follows:-



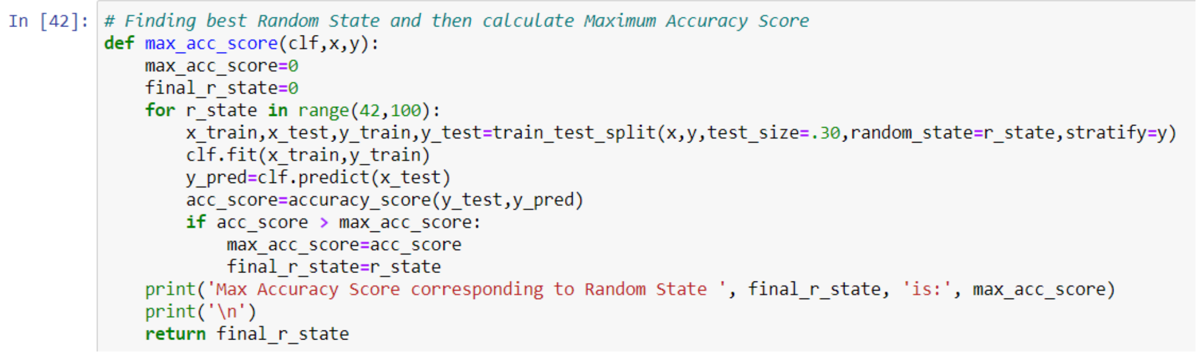
In [38]:
```
# Importing useful libraries for model training

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier

# Ensemble Techniques...

from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import AdaBoostClassifier

# Model selection libraries...
from sklearn.model_selection import cross_val_score, cross_val_predict, train_test_split
from sklearn.model_selection import GridSearchCV

# Importing some metrics we can use to evaluate our model performance....
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix,log_loss
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn.metrics import precision_score, recall_score, f1_score


# Creating instances for different Classifiers

RF=RandomForestClassifier()
LR=LogisticRegression()
MNB=MultinomialNB()
DT=DecisionTreeClassifier()
AD=AdaBoostClassifier()
XG=XGBClassifier(eval_metric='mlogloss')
```



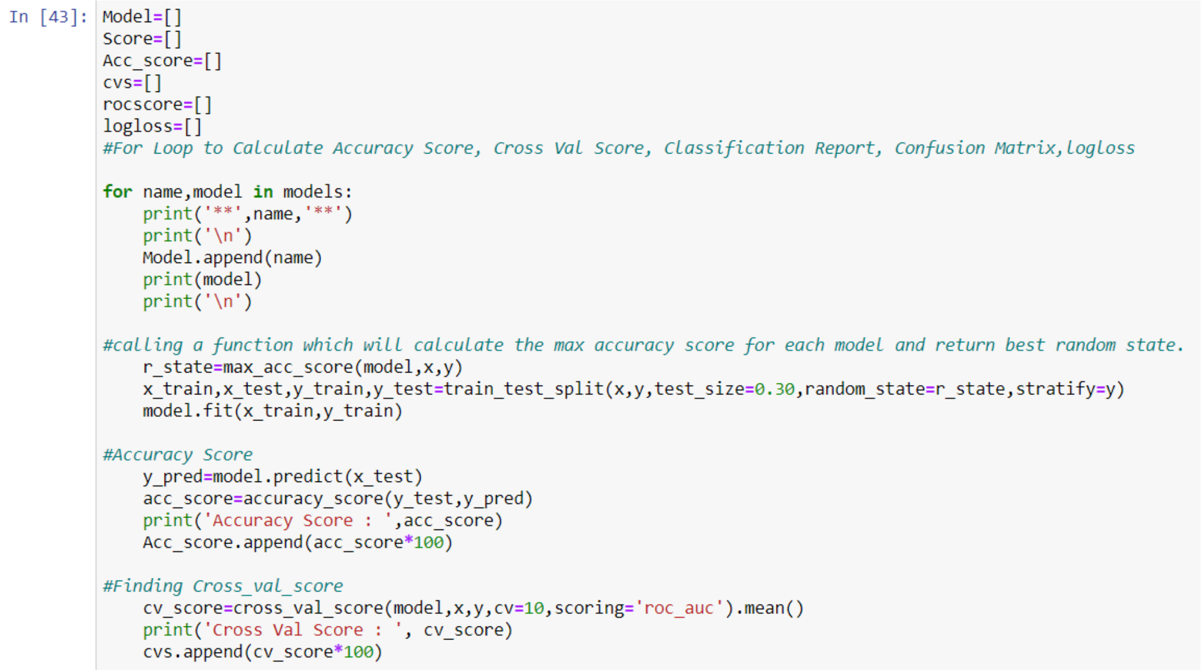
In [39]:
```
# List of Models
models=[]
models.append(('LogisticRegression',LR))
models.append(('MultinomialNB()',MNB))
models.append(('DecisionTreeClassifier',DT))
models.append(('RandomForestClassifier',RF))
models.append(('AdaBoostClassifier',AD))
models.append(('XGBClassifier',XG))
```

RUN AND EVALUATE SELECTED MODELS

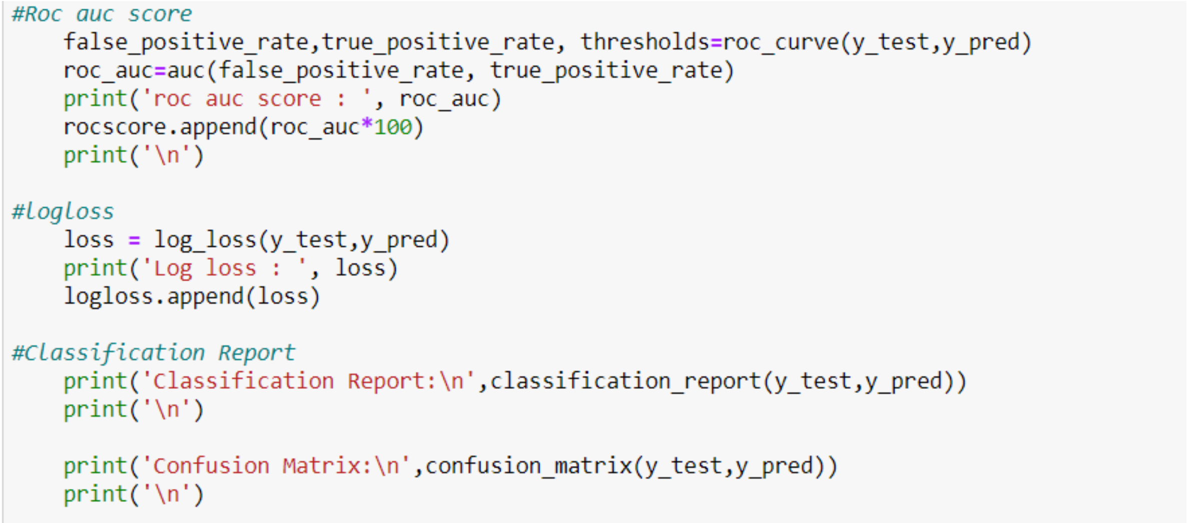
In my approach, I have first prepared a method that gives all necessary classification metrics of an algorithm like classification metrics, auc_roc score, confusion matrix, log_loss.



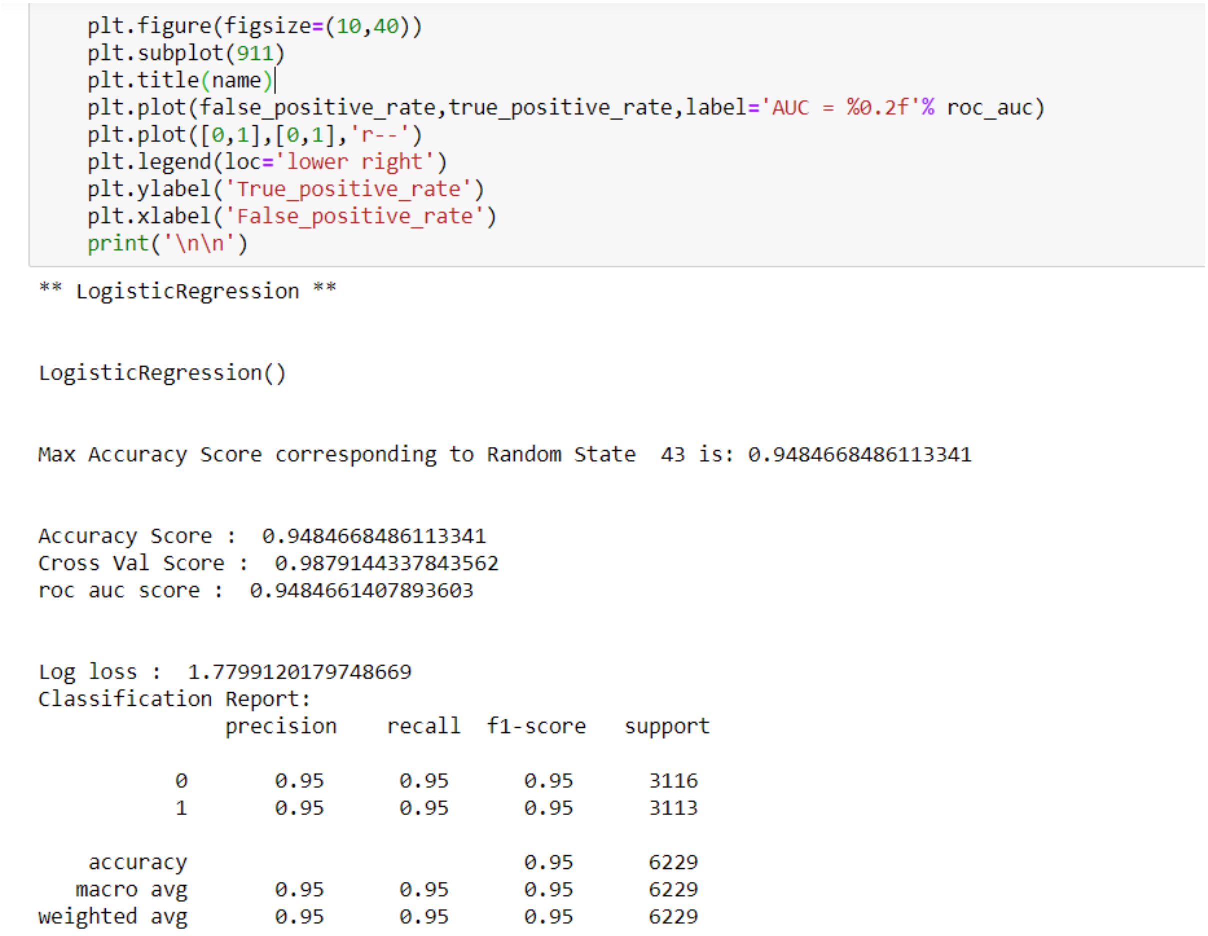
```
In [42]: # Finding best Random State and then calculate Maximum Accuracy Score
         def max_acc_score(clf,x,y):
             max_acc_score=0
             final_r_state=0
             for r_state in range(42,100):
                 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=r_state,stratify=y)
                 clf.fit(x_train,y_train)
                 y_pred=clf.predict(x_test)
                 acc_score=accuracy_score(y_test,y_pred)
                 if acc_score > max_acc_score:
                     max_acc_score=acc_score
                     final_r_state=r_state
             print('Max Accuracy Score corresponding to Random State ', final_r_state, 'is:', max_acc_score)
             print('\n')
             return final_r_state
```



```
In [43]: Model=[]
         Score=[]
         Acc_score=[]
         cvs=[]
         rocscore=[]
         logloss=[]
         #For Loop to Calculate Accuracy Score, Cross Val Score, Classification Report, Confusion Matrix,logloss

         for name,model in models:
             print('**',name,'**')
             print('\n')
             Model.append(name)
             print(model)
             print('\n')

         #calling a function which will calculate the max accuracy score for each model and return best random state.
             r_state=max_acc_score(model,x,y)
             x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=r_state,stratify=y)
             model.fit(x_train,y_train)

         #Accuracy Score
             y_pred=model.predict(x_test)
             acc_score=accuracy_score(y_test,y_pred)
             print('Accuracy Score : ',acc_score)
             Acc_score.append(acc_score*100)

         #Finding Cross_val_score
             cv_score=cross_val_score(model,x,y,cv=10,scoring='roc_auc').mean()
             print('Cross Val Score : ', cv_score)
             cvs.append(cv_score*100)
```

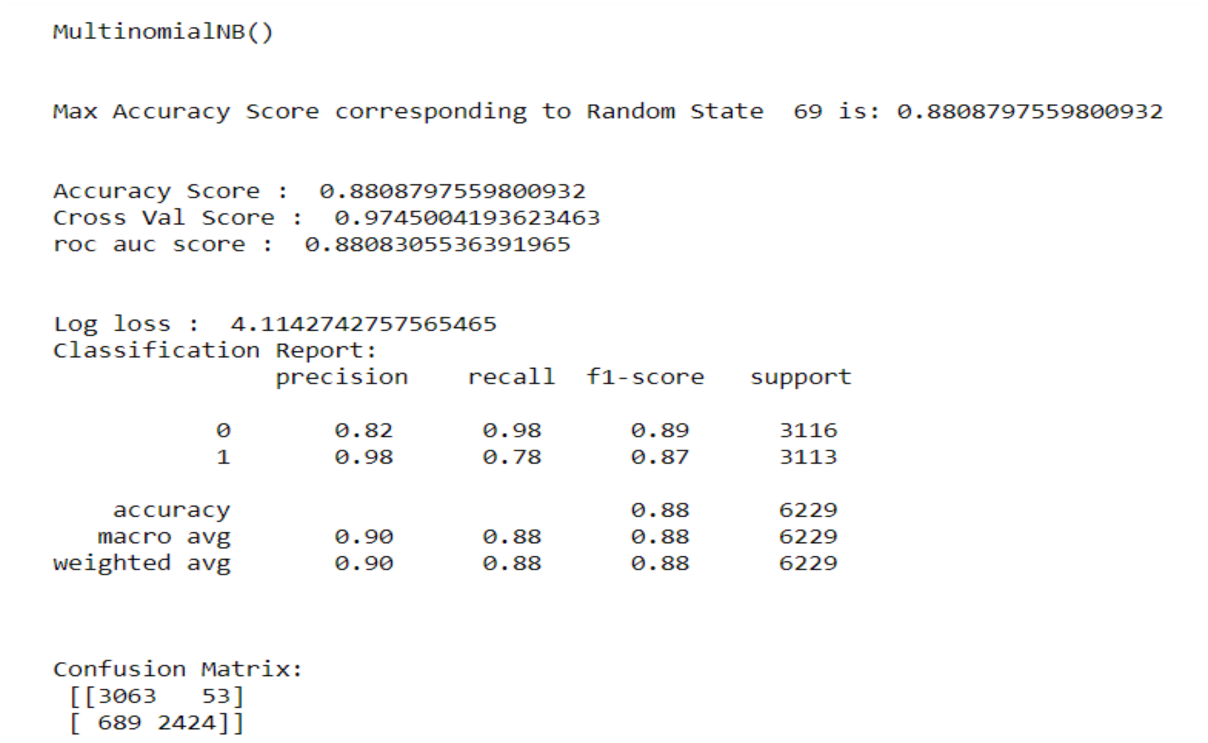


```
#Roc auc score
    false_positive_rate,true_positive_rate, thresholds=roc_curve(y_test,y_pred)
    roc_auc=auc(false_positive_rate, true_positive_rate)
    print('roc auc score : ', roc_auc)
    rocscore.append(roc_auc*100)
    print('\n')

#logloss
    loss = log_loss(y_test,y_pred)
    print('Log loss : ', loss)
    logloss.append(loss)

#Classification Report
    print('Classification Report:\n',classification_report(y_test,y_pred))
    print('\n')

    print('Confusion Matrix:\n',confusion_matrix(y_test,y_pred))
    print('\n')
```

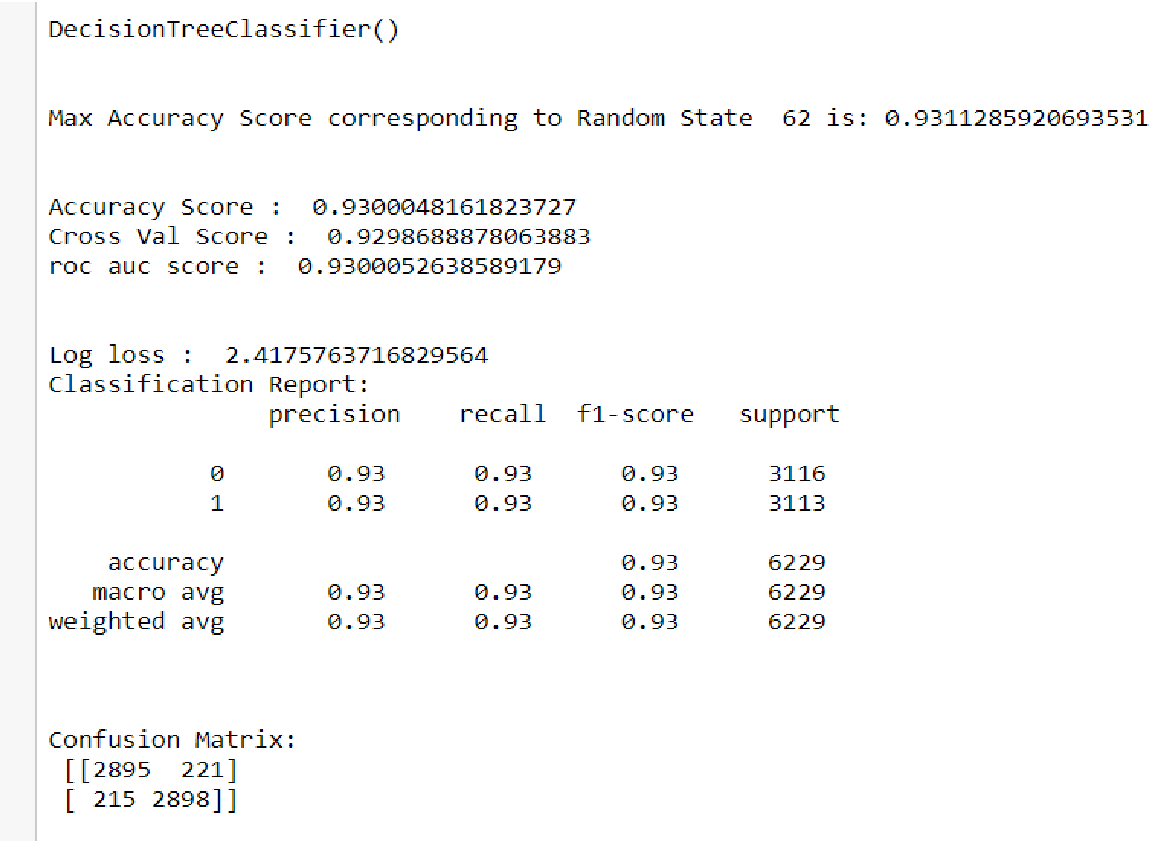


```
    plt.figure(figsize=(10,40))
    plt.subplot(911)
    plt.title(name)
    plt.plot(false_positive_rate,true_positive_rate,label='AUC = %0.2f'% roc_auc)
    plt.plot([0,1],[0,1],'r--')
    plt.legend(loc='lower right')
    plt.ylabel('True_positive_rate')
    plt.xlabel('False_positive_rate')
    print('\n\n')
```

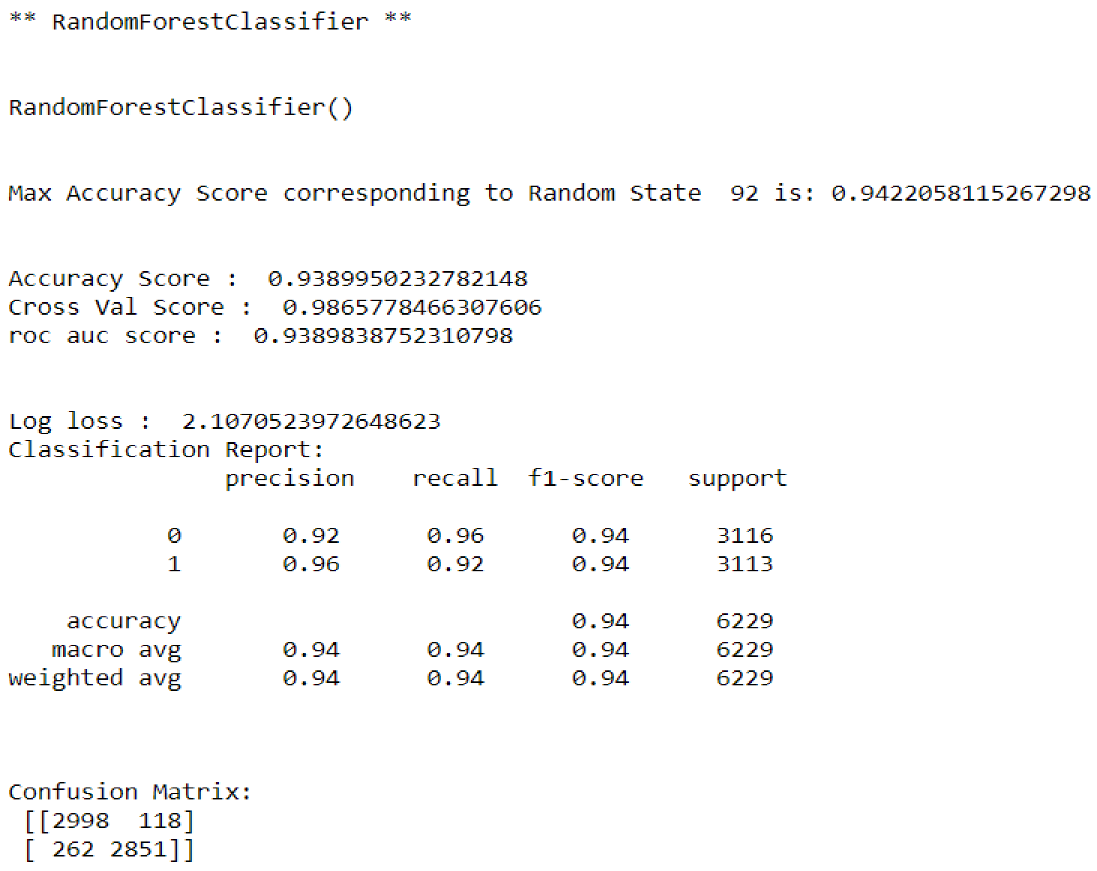
```
** LogisticRegression **


LogisticRegression()


Max Accuracy Score corresponding to Random State  43 is: 0.9484668486113341


Accuracy Score :  0.9484668486113341
Cross Val Score :  0.9879144337843562
roc auc score :  0.9484661407893603


Log loss :  1.7799120179748669
Classification Report:
               precision    recall  f1-score   support

           0       0.95      0.95      0.95      3116
           1       0.95      0.95      0.95      3113

    accuracy                           0.95      6229
   macro avg       0.95      0.95      0.95      6229
weighted avg       0.95      0.95      0.95      6229
```



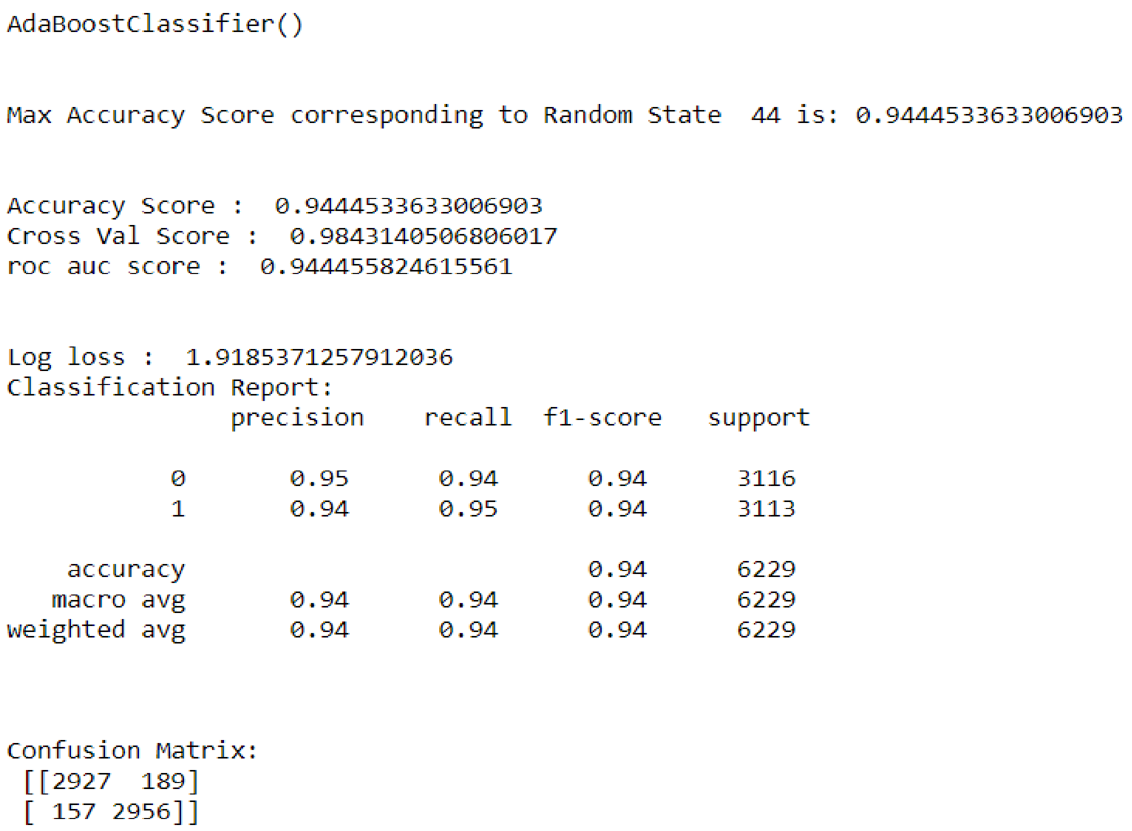
```
MultinomialNB()


Max Accuracy Score corresponding to Random State  69 is: 0.8808797559800932


Accuracy Score :  0.8808797559800932
Cross Val Score :  0.9745004193623463
roc auc score :  0.8808305536391965


Log loss :  4.1142742757565465
Classification Report:
               precision    recall  f1-score   support

           0       0.82      0.98      0.89      3116
           1       0.98      0.78      0.87      3113

    accuracy                           0.88      6229
   macro avg       0.90      0.88      0.88      6229
weighted avg       0.90      0.88      0.88      6229



Confusion Matrix:
 [[3063   53]
 [ 689 2424]]
```



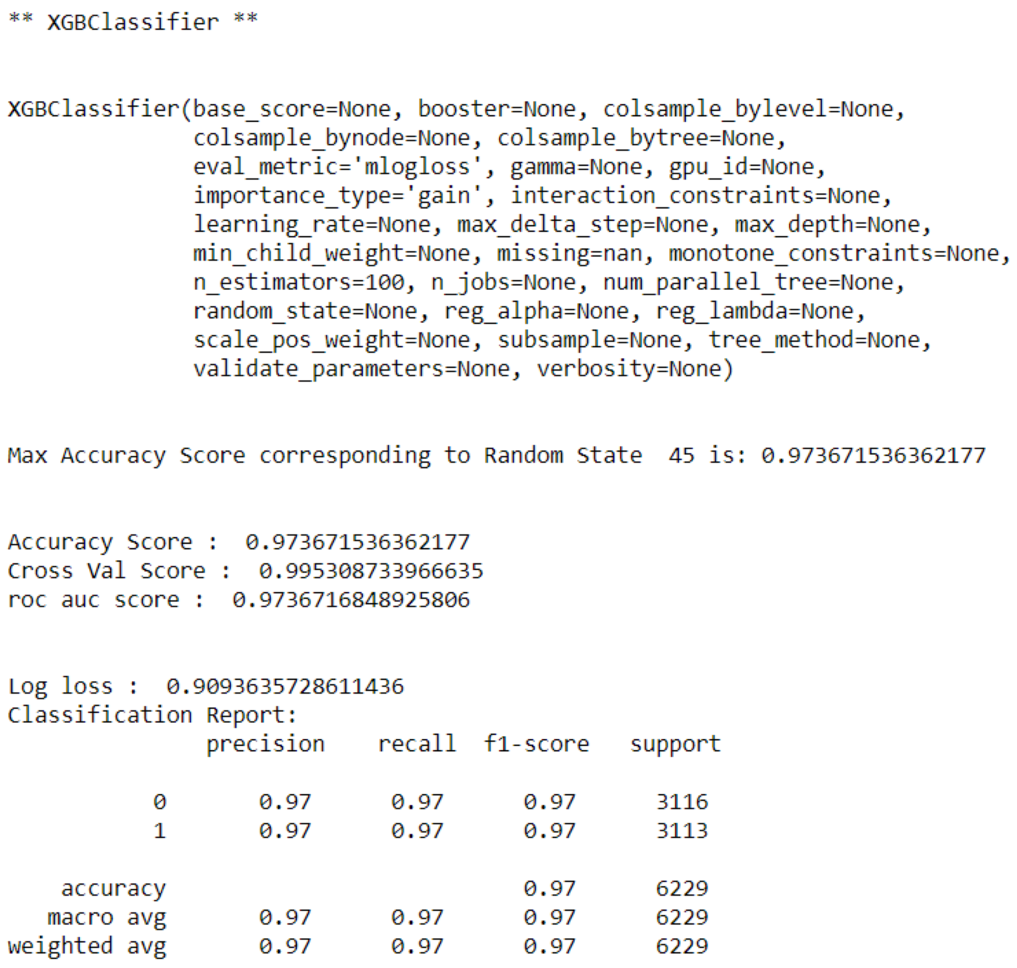
```
DecisionTreeClassifier()


Max Accuracy Score corresponding to Random State  62 is: 0.9311285920693531


Accuracy Score :  0.9300048161823727
Cross Val Score :  0.9298688878063883
roc auc score :  0.9300052638589179


Log loss :  2.4175763716829564
Classification Report:
               precision    recall  f1-score   support

           0       0.93      0.93      0.93      3116
           1       0.93      0.93      0.93      3113

    accuracy                           0.93      6229
   macro avg       0.93      0.93      0.93      6229
weighted avg       0.93      0.93      0.93      6229



Confusion Matrix:
 [[2895  221]
 [ 215 2898]]
```



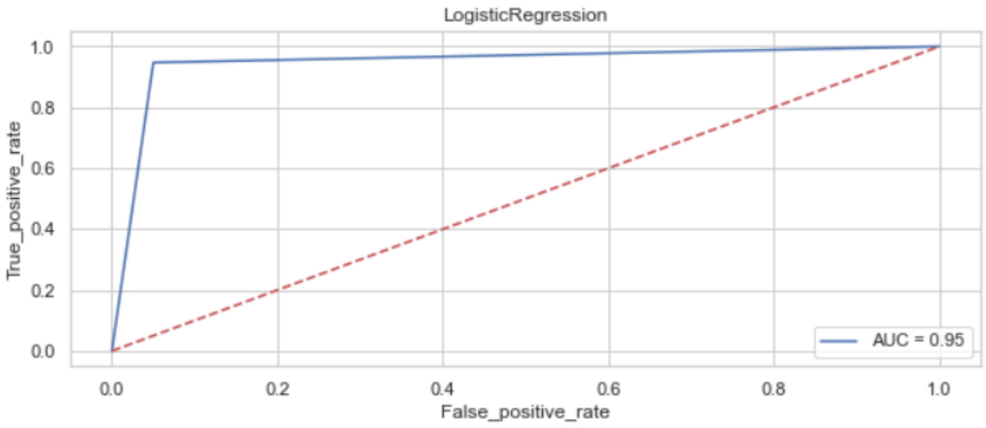
```
** RandomForestClassifier **


RandomForestClassifier()


Max Accuracy Score corresponding to Random State  92 is: 0.9422058115267298


Accuracy Score :  0.9389950232782148
Cross Val Score :  0.9865778466307606
roc auc score :  0.9389838752310798


Log loss :  2.1070523972648623
Classification Report:
               precision    recall  f1-score   support

           0       0.92      0.96      0.94      3116
           1       0.96      0.92      0.94      3113

    accuracy                           0.94      6229
   macro avg       0.94      0.94      0.94      6229
weighted avg       0.94      0.94      0.94      6229



Confusion Matrix:
 [[2998  118]
 [ 262 2851]]
```

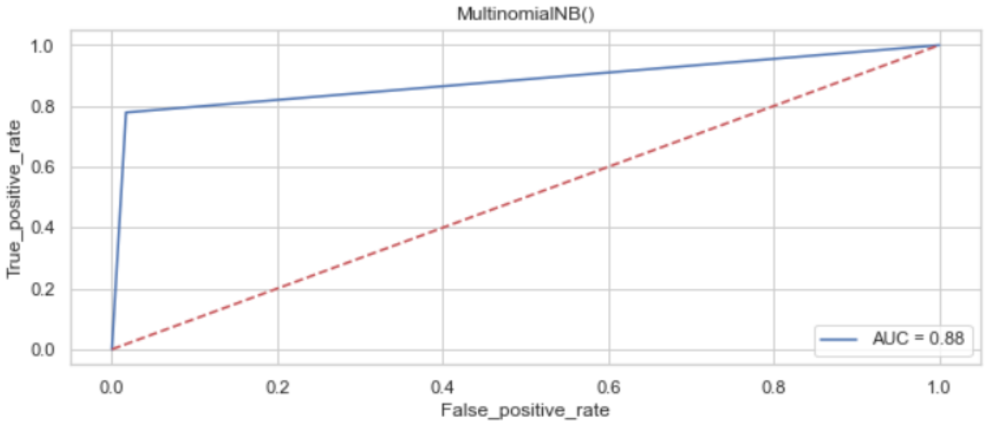


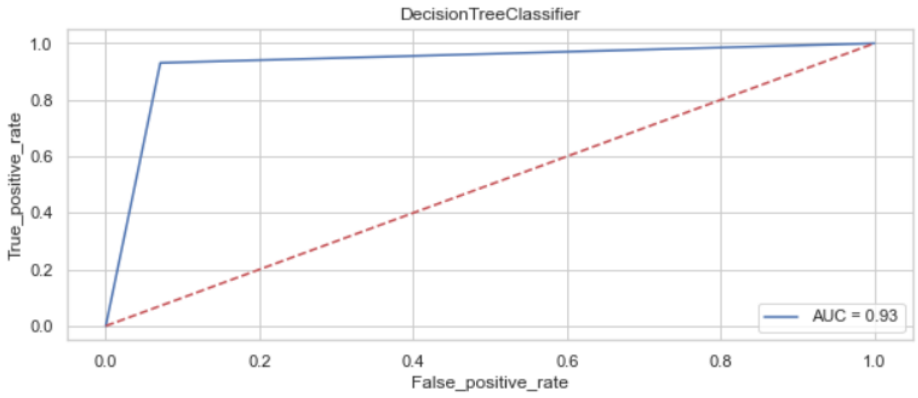
```
AdaBoostClassifier()


Max Accuracy Score corresponding to Random State  44 is: 0.9444533633006903


Accuracy Score :  0.9444533633006903
Cross Val Score :  0.9843140506806017
roc auc score :  0.944455824615561


Log loss :  1.9185371257912036
Classification Report:
               precision    recall  f1-score   support

           0       0.95      0.94      0.94      3116
           1       0.94      0.95      0.94      3113

    accuracy                           0.94      6229
   macro avg       0.94      0.94      0.94      6229
weighted avg       0.94      0.94      0.94      6229



Confusion Matrix:
 [[2927  189]
 [ 157 2956]]
```

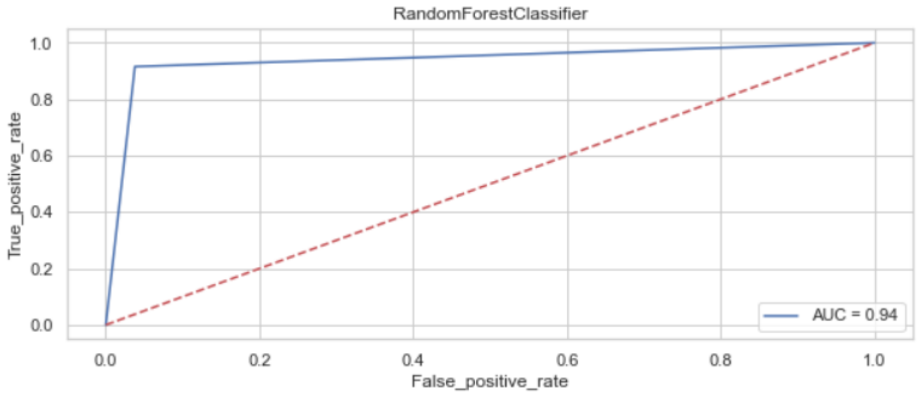


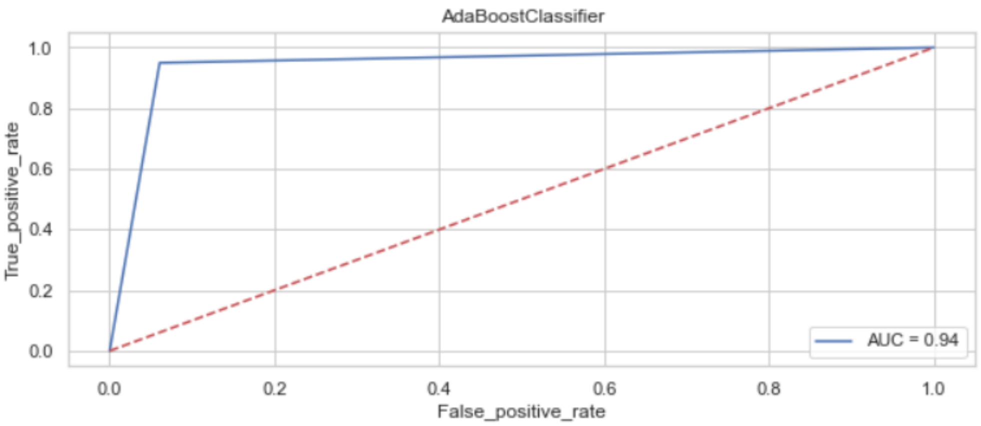
```
 ** XGBClassifier **


 XGBClassifier(base_score=None, booster=None, colsample_bylevel=None,
              colsample_bynode=None, colsample_bytree=None,
              eval_metric='mlogloss', gamma=None, gpu_id=None,
              importance_type='gain', interaction_constraints=None,
              learning_rate=None, max_delta_step=None, max_depth=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              random_state=None, reg_alpha=None, reg_lambda=None,
              scale_pos_weight=None, subsample=None, tree_method=None,
              validate_parameters=None, verbosity=None)


 Max Accuracy Score corresponding to Random State  45 is: 0.973671536362177


 Accuracy Score :  0.973671536362177
 Cross Val Score :  0.995308733966635
 roc auc score :  0.9736716848925806


 Log loss :  0.9093635728611436
 Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.97      0.97      3116
           1       0.97      0.97      0.97      3113

    accuracy                           0.97      6229
   macro avg       0.97      0.97      0.97      6229
weighted avg       0.97      0.97      0.97      6229
```

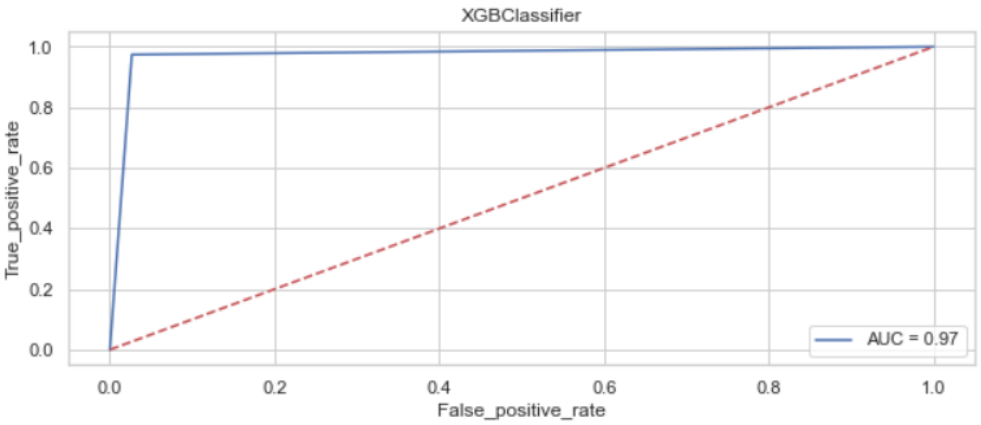

LogisticRegression
True_positive_rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False_positive_rate
AUC = 0.95

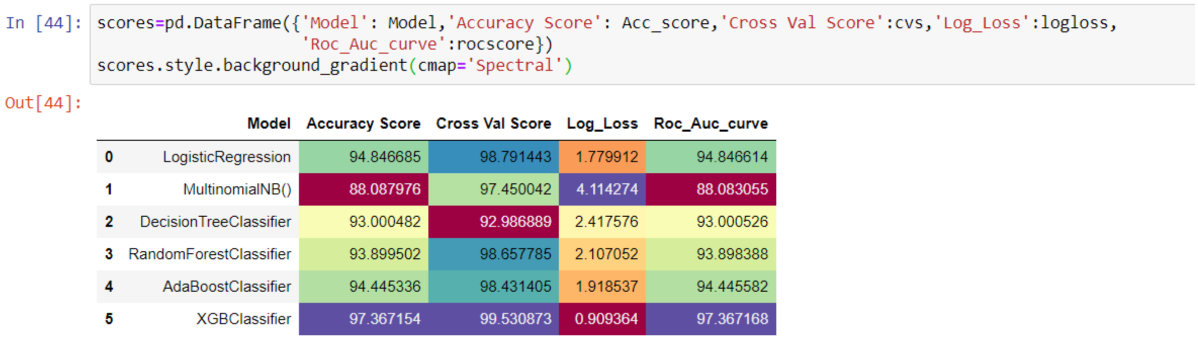

MultinomialNB()
True_positive_rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False_positive_rate
AUC = 0.88


DecisionTreeClassifier
True_positive_rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False_positive_rate
AUC = 0.93


RandomForestClassifier
True_positive_rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False_positive_rate
AUC = 0.94







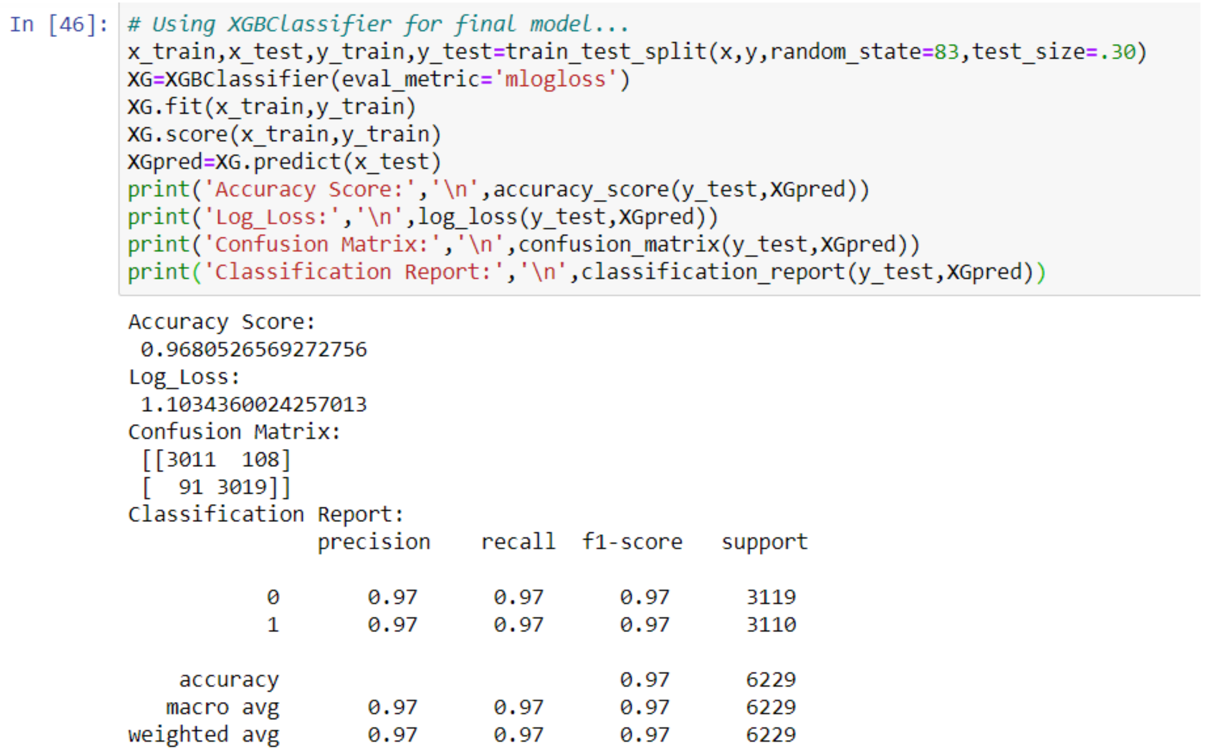
```
In [44]: scores=pd.DataFrame({'Model': Model,'Accuracy Score': Acc_score,'Cross Val Score':cvs,'Log_Loss':logloss,
                             'Roc_Auc_curve':rocscore})
         scores.style.background_gradient(cmap='Spectral')
```

Out[44]:

	Model	Accuracy Score	Cross Val Score	Log_Loss	Roc_Auc_curve
0	LogisticRegression	94.846685	98.791443	1.779912	94.846614
1	MultinomialNB()	88.087976	97.450042	4.114274	88.083055
2	DecisionTreeClassifier	93.000482	92.986889	2.417576	93.000526
3	RandomForestClassifier	93.899502	98.657785	2.107052	93.898388
4	AdaBoostClassifier	94.445336	98.431405	1.918537	94.445582
5	XGBClassifier	97.367154	99.530873	0.909364	97.367168

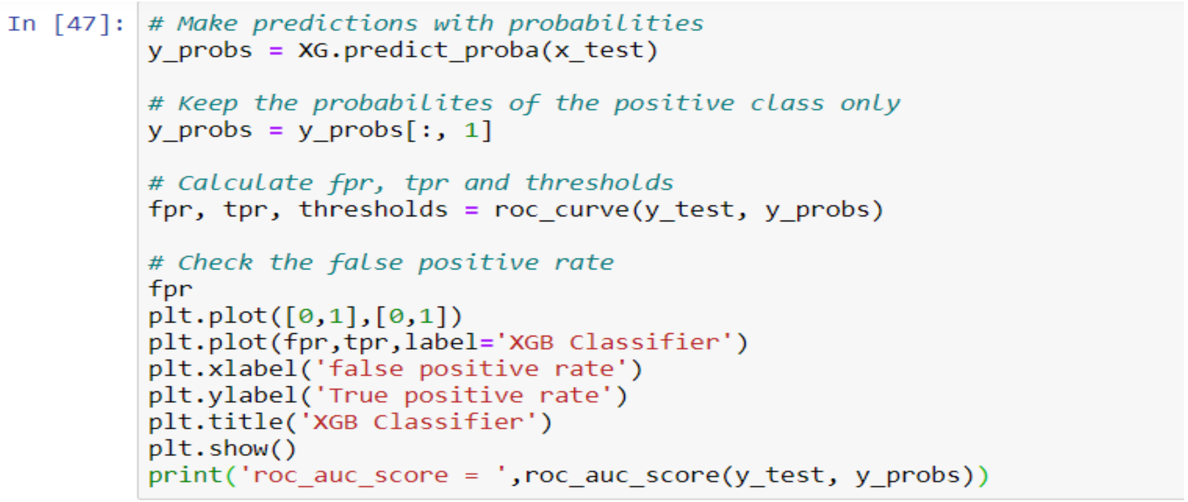
We choose the XGBoost Classifier model as the final one, as it gives the highest accuracy score & also log_loss value is minimum which indicates the better prediction

FINAL MODEL

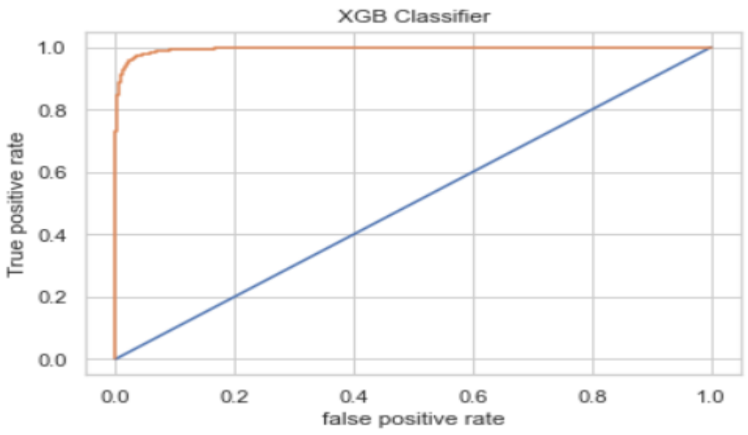


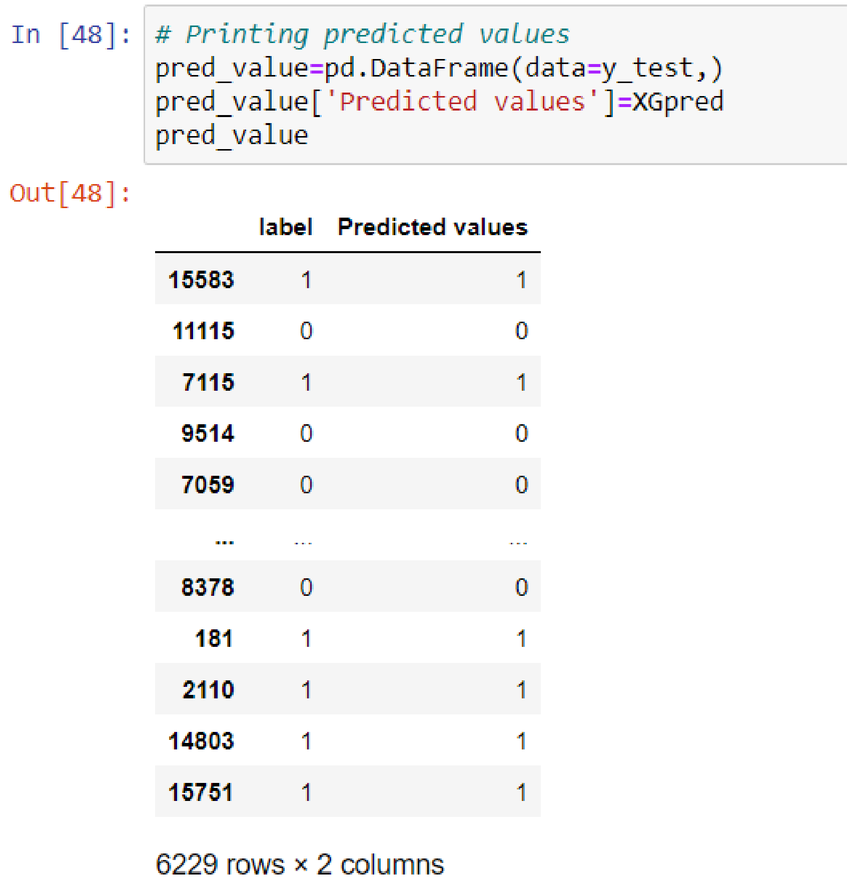
```
In [46]: # Using XGBClassifier for final model...
         x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=83,test_size=.30)
         XG=XGBClassifier(eval_metric='mlogloss')
         XG.fit(x_train,y_train)
         XG.score(x_train,y_train)
         XGpred=XG.predict(x_test)
         print('Accuracy Score:','\n',accuracy_score(y_test,XGpred))
         print('Log_Loss:','\n',log_loss(y_test,XGpred))
         print('Confusion Matrix:','\n',confusion_matrix(y_test,XGpred))
         print('Classification Report:','\n',classification_report(y_test,XGpred))
```

```
Accuracy Score:
 0.9680526569272756
Log_Loss:
 1.1034360024257013
Confusion Matrix:
 [[3011  108]
 [  91 3019]]
Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.97      0.97      3119
           1       0.97      0.97      0.97      3110

    accuracy                           0.97      6229
   macro avg       0.97      0.97      0.97      6229
weighted avg       0.97      0.97      0.97      6229
```

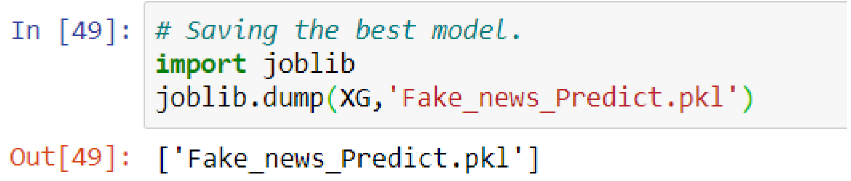


```
In [47]: # Make predictions with probabilities
         y_probs = XG.predict_proba(x_test)

         # Keep the probabilites of the positive class only
         y_probs = y_probs[:, 1]

         # Calculate fpr, tpr and thresholds
         fpr, tpr, thresholds = roc_curve(y_test, y_probs)

         # Check the false positive rate
         fpr
         plt.plot([0,1],[0,1])
         plt.plot(fpr,tpr,label='XGB Classifier')
         plt.xlabel('false positive rate')
         plt.ylabel('True positive rate')
         plt.title('XGB Classifier')
         plt.show()
         print('roc_auc_score = ',roc_auc_score(y_test, y_probs))
```

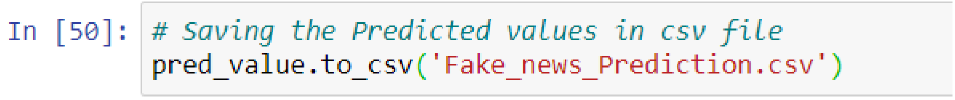




```
In [48]: # Printing predicted values
         pred_value=pd.DataFrame(data=y_test,)
         pred_value['Predicted values']=XGpred
         pred_value

Out[48]:
                label  Predicted values
         15583      1                 1
         11115      0                 0
          7115      1                 1
          9514      0                 0
          7059      0                 0
           ...    ...               ...
          8378      0                 0
           181      1                 1
          2110      1                 1
         14803      1                 1
         15751      1                 1

         6229 rows × 2 columns
```



```
In [49]: # Saving the best model.
         import joblib
         joblib.dump(XG,'Fake_news_Predict.pkl')

Out[49]: ['Fake_news_Predict.pkl']
```



```
In [50]: # Saving the Predicted values in csv file
         pred_value.to_csv('Fake_news_Prediction.csv')
```

KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION

- When it comes to the evaluation of a data science model’s performance, sometimes accuracy may not be the best indicator.
- Some problems that we are solving in real life might have a very imbalanced class and using accuracy might not give us enough confidence to understand the algorithm’s performance.

- In the fake news problem that we are trying to solve, the data is balanced. so accuracy score nearly tells the right predictions. So the problem of overfitting in this problem is nearly not to occur. So here, we are using an accuracy score to find a better model.

CONCLUSION

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

From the whole evaluation, we can see that the maximum number of words in fake news were regarding Trump, and Clinton and we can interpret that it was due to election campaign which was held during the US presidential election and we know these adverse effects of the voters which were influenced by the fake news and most of the real news had said, trump and president, and fake news which was cleared by trump's campaign, but can hardly see any clarity or real news from the side of Clinton, and due to which the impact we already saw on election results and regarding the election advertisement and news Facebook's CEO Mark Zuckerberg also got extensively question by congress.

LEARNING OUTCOMES OF THE STUDY IN RESPECT OF DATA SCIENCE

It is possible to classify news content into the required categories of authentic and fake news however there will be always a bias to this kind of classification which depends on the behavioural pattern of the listener. However, using this kind of project awareness can be created to know what is fake and authentic.

LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK

Machine Learning Algorithms like XGBoost, Adaboost and Randomforest Classifier took an enormous amount of time to build the model. Using Hyper-parameter tuning for XGB would have resulted in some more accuracy.