FLIP ROBO

Fake-News Detection Project

Submitted by:

Anchal Awasthi

ACKNOWLEDGMENT

I am happy to present this project after completing it successfully. I am thankful to Flip Robo Technology for providing me an opportunity to execute this project.

Following references and links helped me understand the concepts and helped me in

completion of the project.

1. https://stackoverflow.com
2. https://medium.com
3. https://towardsdatascience.com
4. https://www.analyticsvidhya.com

INTRODUCTION

Conceptual Background of the Domain Problem

Fake news has existed since the dawn of the printing press but in the age of internet and social media, it has found a tremendous application. Manipulation of algorithms of social media and search engines—to reach large audiences and mislead news consumers is a global trend now. Fake video clips, news stories with morphed media logos, bots, paid commentators for favourable online reputation (troll farm) have become very common. Governments are using the threat of fake news to clamp down on free speech.

Problem Statement

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.

So our work is to build a model to detect the Fake news. This project is highly associated with real word. Because in the world we stay, work & move depending on news. And circulation of Fake news can effect a lot to everyone.

Motivation for the Problem Undertaken

Click-baits lure users and entice curiosity with flashy headlines or designs to click links to increase advertisements revenues. This exposition analyzes the prevalence of fake news in light of the advances in communication made possible by the emergence of social networking sites. The purpose of the work is to come up with a solution that can be utilized by users to detect and filter out sites containing false and misleading information. We use simple and carefully selected features of the title and post to accurately identify fake posts.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

With continuous increase in available data, there is a pressing need to organize it and modern classification problems often involve the prediction of multiple labels

simultaneously associated with a single instance. Known as Multi-Label Classification, it is one such task which is omnipresent in many real world problems. In this project also, we have multi-label classification problem.

We have used Tf-Idf Vectorizer to vectorize the words in our dataset. TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction. It is very important for tuning performance on NLP projects.

The TF-IDF score for the word t in the document d from the document set D is calculated as follows:

$$tf\,idf\,(t, d, D) = tf\,(t, d)\ .\ idf\,(t, D)$$

Where:

$$tf\,(t, d) = log\ (1 + freq\ (t, d))$$

$$idf\,(t, D) = log\ \left(\ \frac{N}{count\ (d \in D{:}t \in d)}\ \right)$$

Data Sources and their formats

There are 6 columns in the dataset provided to you. The description of each of the column is given below:

“id”: Unique id of each news article

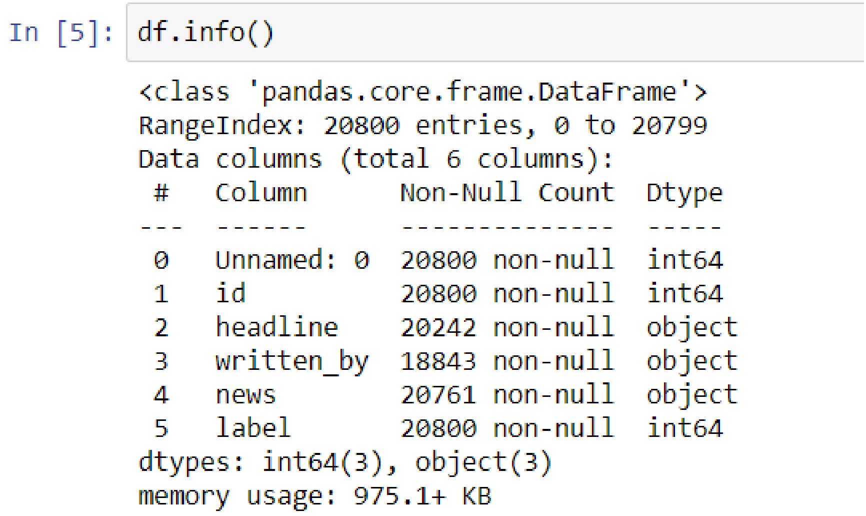
“headline”: It is the title of the news.

“news”: It contains the full text of the news article

“Unnamed:0”: It is a serial number

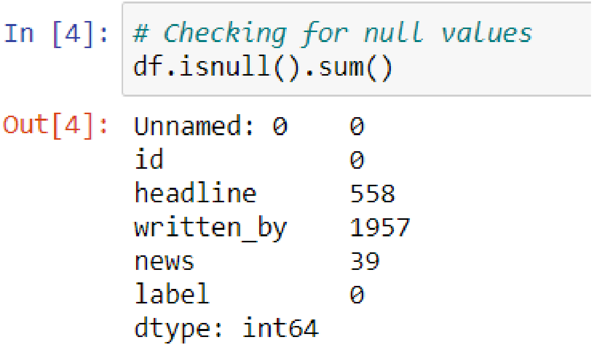
“written_by”: It represents the author of the news article

“label”: It tells whether the news is fake (1) or not fake (0).



```
In [5]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20800 entries, 0 to 20799
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  20800 non-null  int64 
 1   id          20800 non-null  int64 
 2   headline    20242 non-null  object
 3   written_by  18843 non-null  object
 4   news        20761 non-null  object
 5   label       20800 non-null  int64 
dtypes: int64(3), object(3)
memory usage: 975.1+ KB
```

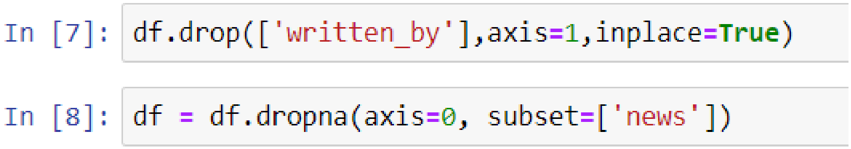
Checking if the data has null values:



```
In [4]: # Checking for null values
        df.isnull().sum()

Out[4]: Unnamed: 0       0
        id               0
        headline       558
        written_by    1957
        news            39
        label            0
        dtype: int64
```

There are 558 null values in headline, 1957 null values in written_by and 39 null values in news which needs to be treated for a more accurate result.

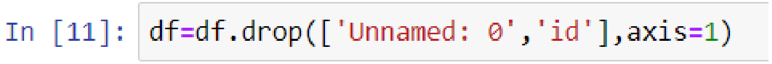
It is safe to drop *‘Written_by’* column as there are many null values and dropping all the rows with null values in the column news as we are working on analyzing the fake news.



```
In [7]: df.drop(['written_by'],axis=1,inplace=True)
```

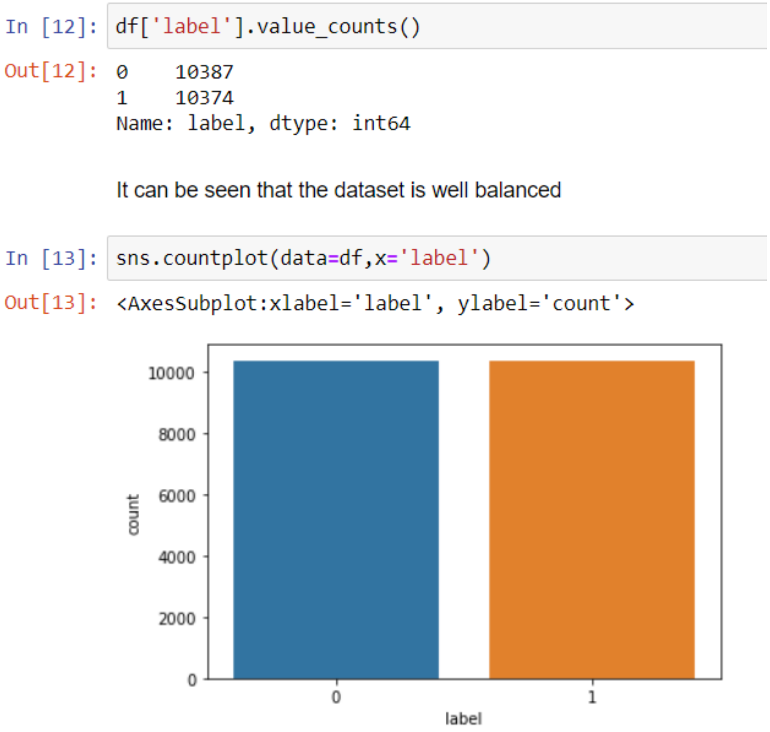
```
In [8]: df = df.dropna(axis=0, subset=['news'])
```

Columns like ‘Unnamed’ and 'id' columns can be dropped as they do not contribute to the target variable.



```
In [11]: df=df.drop(['Unnamed: 0','id'],axis=1)
```

Data Analysis



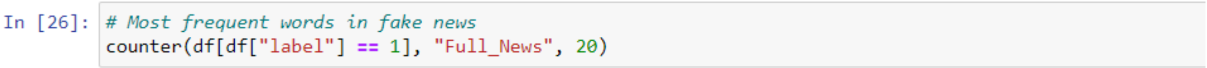
```
In [12]: df['label'].value_counts()

Out[12]: 0    10387
         1    10374
         Name: label, dtype: int64
```

It can be seen that the dataset is well balanced

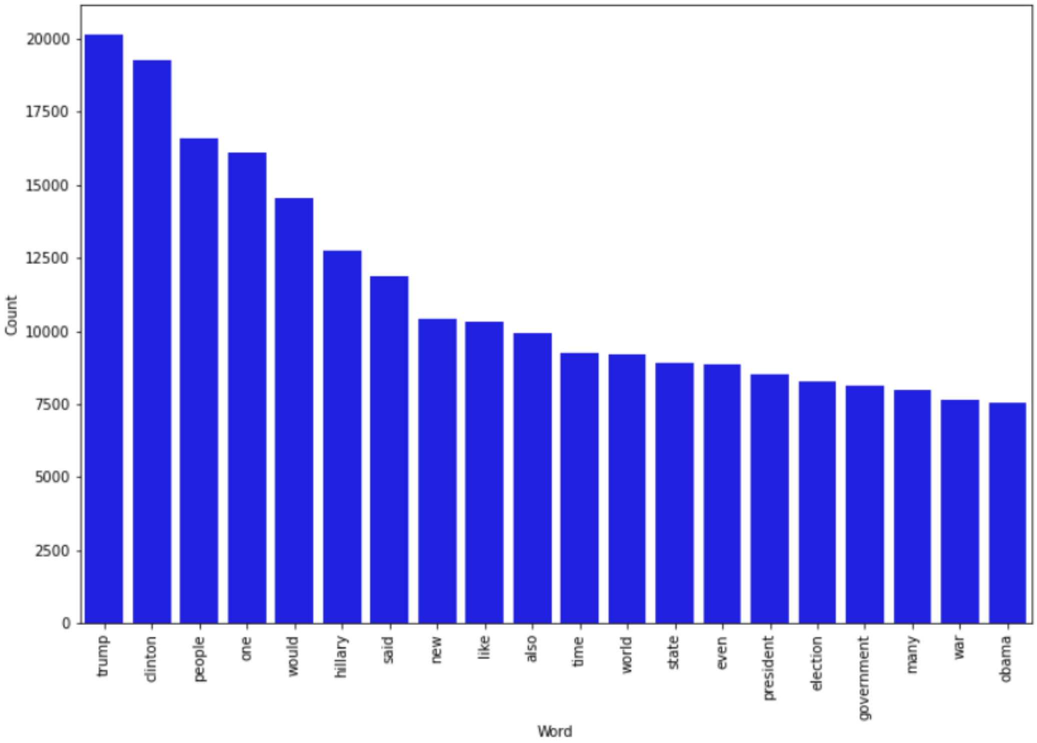
```
In [13]: sns.countplot(data=df,x='label')

Out[13]: <AxesSubplot:xlabel='label', ylabel='count'>
```

It can be seen that the dataset is well balanced and will produce good results, hence this column can be left un-altered.

Performing data analyses on the news column can reveal some very useful facts, let’s have a look at it.

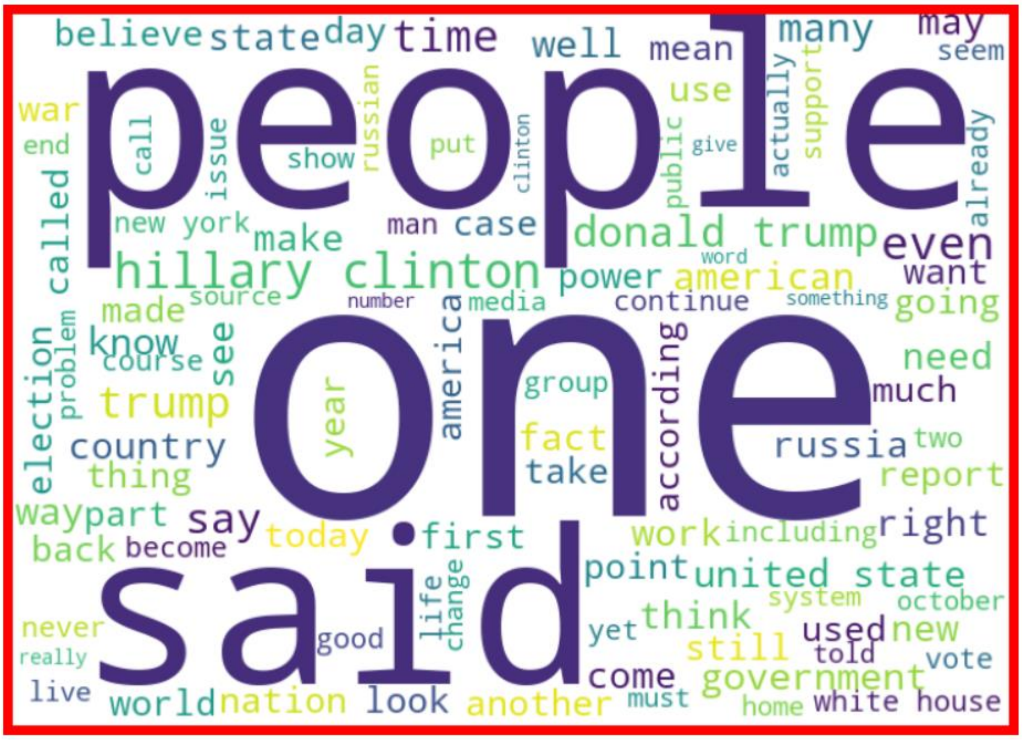


```
In [26]: # Most frequent words in fake news
         counter(df[df["label"] == 1], "Full_News", 20)
```

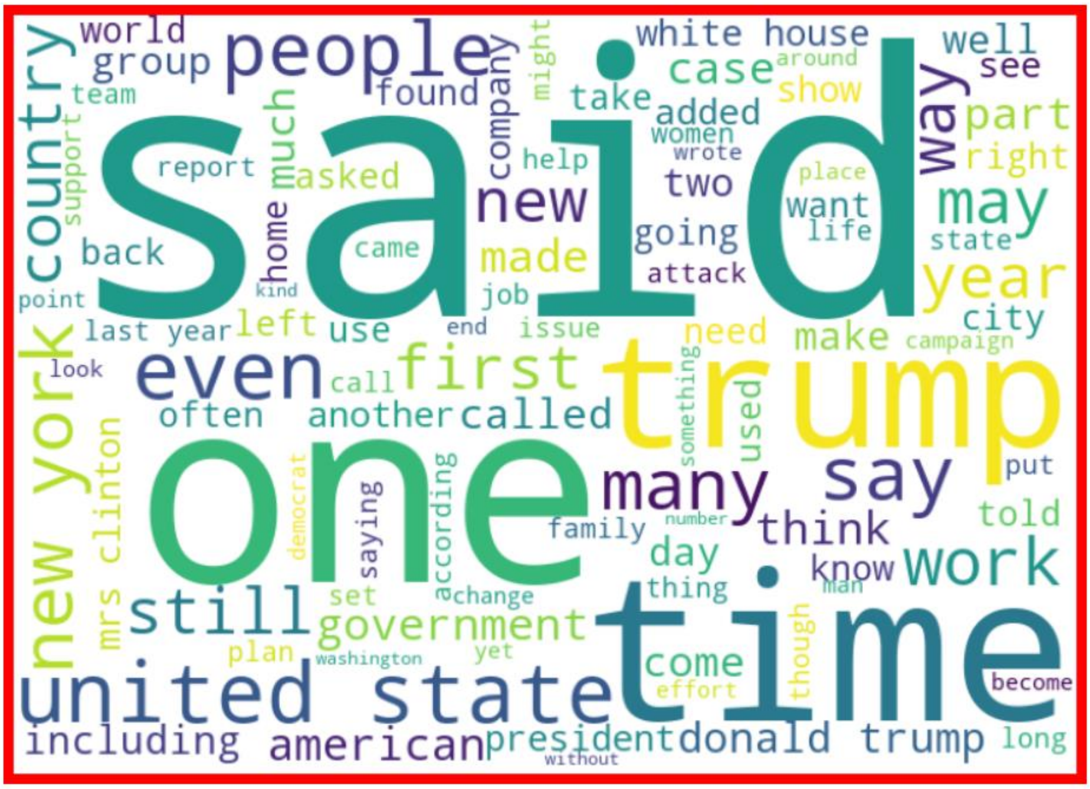


We can see that the top 10 words used in fake news are mainly political. Hence it is safe to conclude that maximum number of fake news revolves around politics.

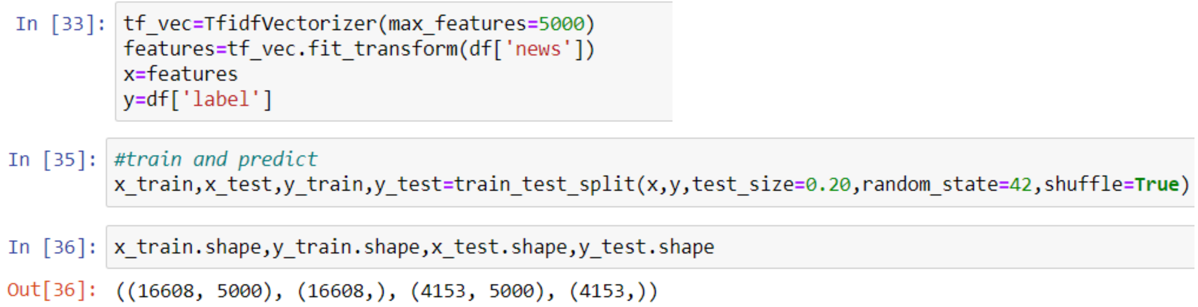
Let’s try to study the same with the help of a word cloud



We should also have a look at genuine news



Preparing data for modeling



```
In [33]: tf_vec=TfidfVectorizer(max_features=5000)
         features=tf_vec.fit_transform(df['news'])
         x=features
         y=df['label']

In [35]: #train and predict
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=42,shuffle=True)

In [36]: x_train.shape,y_train.shape,x_test.shape,y_test.shape

Out[36]: ((16608, 5000), (16608,), (4153, 5000), (4153,))
```

Model Building

Model/s Development and Evaluation

Testing of Identified Approaches (Algorithms)

i. LogisticRegression

ii. MultinomialNB

iii. GaussianNB

iv. DecisionTreeClassifier

v. KNeighborsClassifier

vi. RandomForestClassifier

vii. SGDClassifier

viii. PassiveAggressiveClassifier

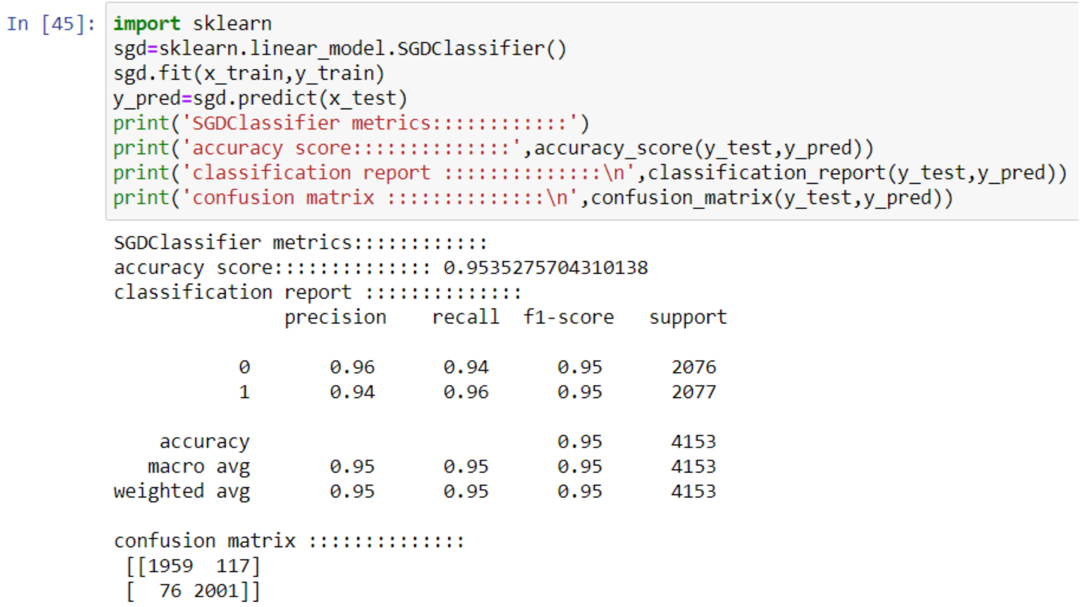
Key Metrics for Evaluation are

i. accuracy_score: As it is a classification problem accuracy score is required to check the accuracy score so it is used.

ii. confusion_matrix: To see the number tp,fp,tn,fn.

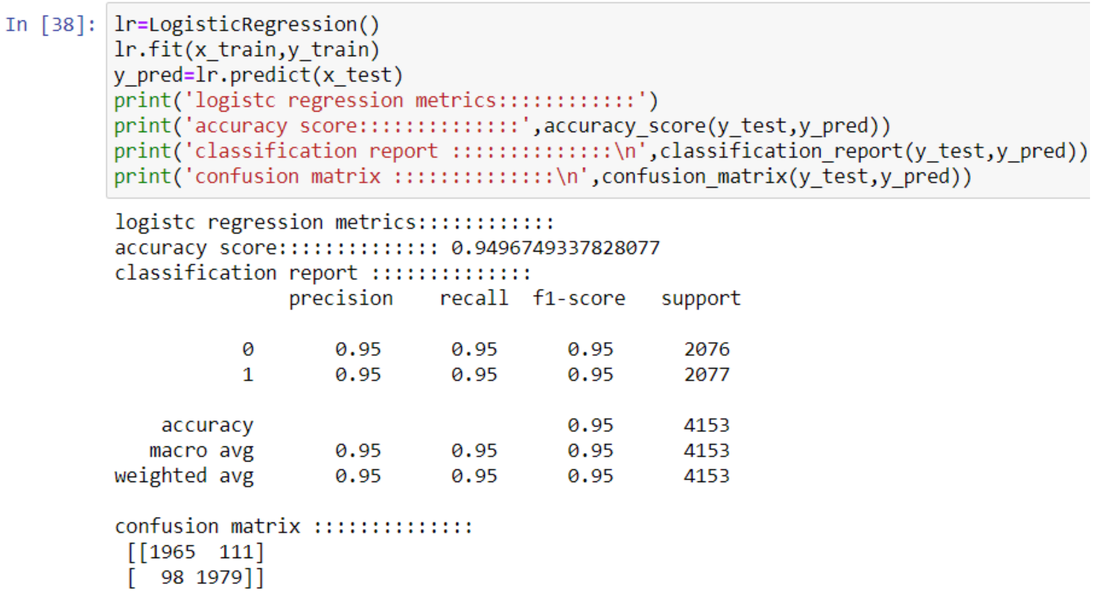
iii. Classification_report: To check all the metrics like precision,recall,f1-score it is used.

The best results are produced by Logistic regression and sgd classifier



```
In [45]: import sklearn
         sgd=sklearn.linear_model.SGDClassifier()
         sgd.fit(x_train,y_train)
         y_pred=sgd.predict(x_test)
         print('SGDClassifier metrics::::::::::::')
         print('accuracy score::::::::::::::',accuracy_score(y_test,y_pred))
         print('classification report ::::::::::::::\n',classification_report(y_test,y_pred))
         print('confusion matrix ::::::::::::::\n',confusion_matrix(y_test,y_pred))
```

```
SGDClassifier metrics::::::::::::
accuracy score:::::::::::::: 0.9535275704310138
classification report ::::::::::::::
               precision    recall  f1-score   support

           0       0.96      0.94      0.95      2076
           1       0.94      0.96      0.95      2077

    accuracy                           0.95      4153
   macro avg       0.95      0.95      0.95      4153
weighted avg       0.95      0.95      0.95      4153

confusion matrix ::::::::::::::
 [[1959  117]
 [  76 2001]]
```



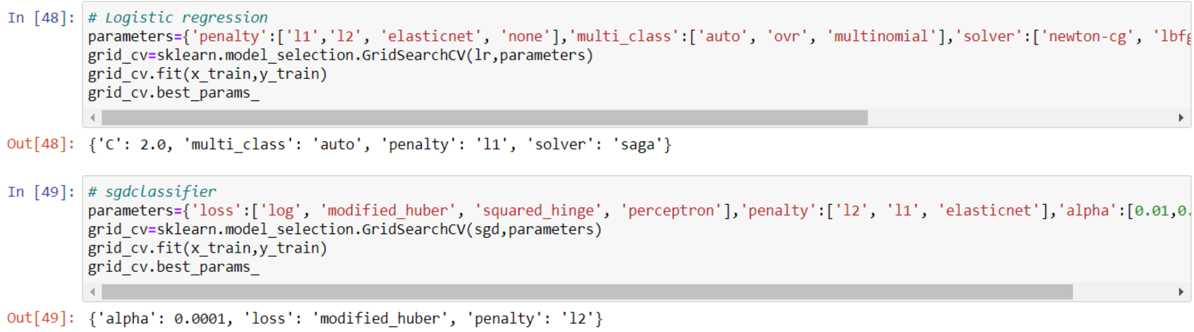
```
In [38]: lr=LogisticRegression()
         lr.fit(x_train,y_train)
         y_pred=lr.predict(x_test)
         print('logistc regression metrics::::::::::::')
         print('accuracy score::::::::::::::',accuracy_score(y_test,y_pred))
         print('classification report ::::::::::::::\n',classification_report(y_test,y_pred))
         print('confusion matrix ::::::::::::::\n',confusion_matrix(y_test,y_pred))
```

```
logistc regression metrics::::::::::::
accuracy score:::::::::::::: 0.9496749337828077
classification report ::::::::::::::
               precision    recall  f1-score   support

           0       0.95      0.95      0.95      2076
           1       0.95      0.95      0.95      2077

    accuracy                           0.95      4153
   macro avg       0.95      0.95      0.95      4153
weighted avg       0.95      0.95      0.95      4153

confusion matrix ::::::::::::::
 [[1965  111]
 [  98 1979]]
```

The accuracy is acceptable however they can be increased by performing hyper-parameter tuning using GridSearchCV.

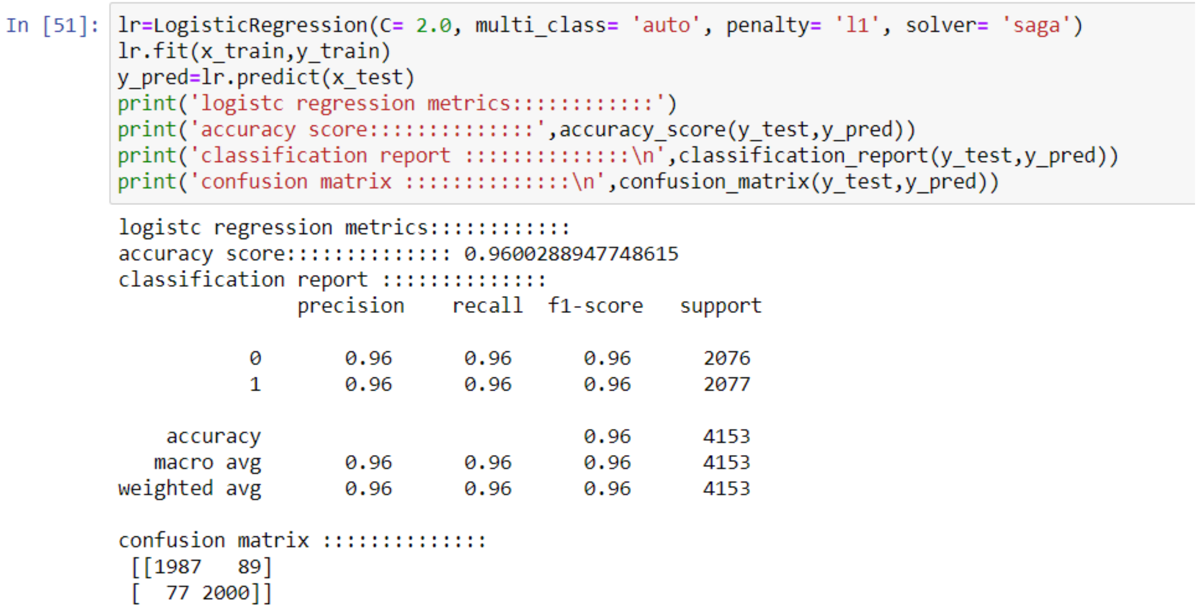
Hyper-parameter tuning is choosing a set of optimal hyper-parameters for a learning algorithm. A hyper-parameter is a model argument whose value is set before the learning process begins. The key to machine learning algorithms is hyper parameter tuning.

GridSearchCV is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyper parameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyper-parameters.

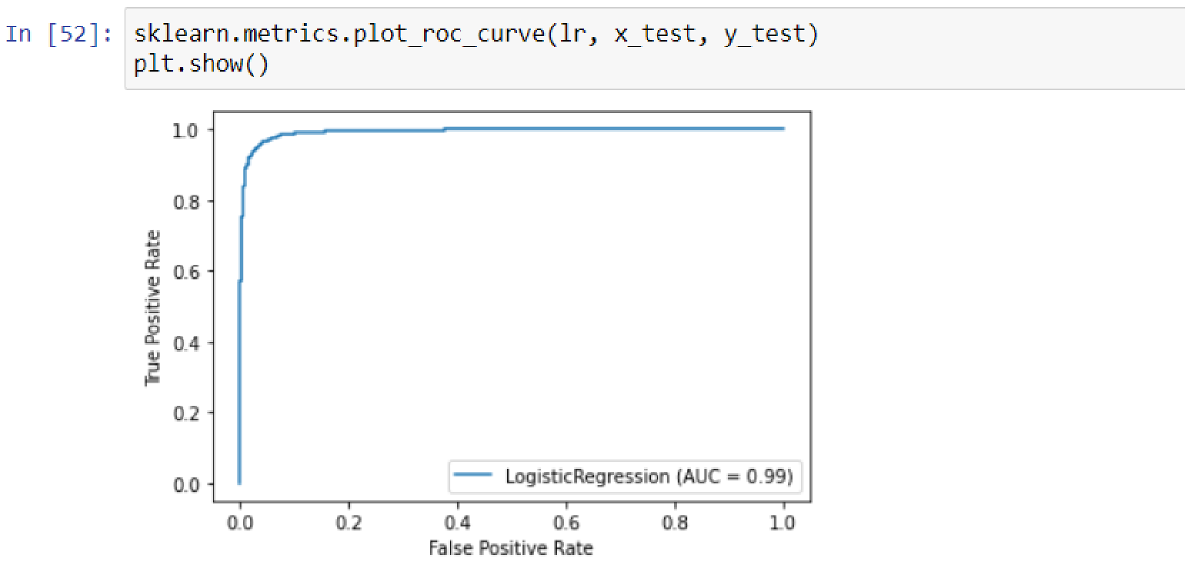


```
In [48]: # Logistic regression
         parameters={'penalty':['l1','l2', 'elasticnet', 'none'],'multi_class':['auto', 'ovr', 'multinomial'],'solver':['newton-cg', 'lbfg
         grid_cv=sklearn.model_selection.GridSearchCV(lr,parameters)
         grid_cv.fit(x_train,y_train)
         grid_cv.best_params_

Out[48]: {'C': 2.0, 'multi_class': 'auto', 'penalty': 'l1', 'solver': 'saga'}

In [49]: # sgdclassifier
         parameters={'loss':['log', 'modified_huber', 'squared_hinge', 'perceptron'],'penalty':['l2', 'l1', 'elasticnet'],'alpha':[0.01,0.
         grid_cv=sklearn.model_selection.GridSearchCV(sgd,parameters)
         grid_cv.fit(x_train,y_train)
         grid_cv.best_params_

Out[49]: {'alpha': 0.0001, 'loss': 'modified_huber', 'penalty': 'l2'}
```

After performing hyperparameter tuning and applying them, logistic regression produces the highest results which is 96%.



```
In [51]: lr=LogisticRegression(C= 2.0, multi_class= 'auto', penalty= 'l1', solver= 'saga')
         lr.fit(x_train,y_train)
         y_pred=lr.predict(x_test)
         print('logistc regression metrics:::::::::::')
         print('accuracy score:::::::::::::',accuracy_score(y_test,y_pred))
         print('classification report ::::::::::::::\n',classification_report(y_test,y_pred))
         print('confusion matrix ::::::::::::::\n',confusion_matrix(y_test,y_pred))

         logistc regression metrics:::::::::::
         accuracy score::::::::::::: 0.9600288947748615
         classification report ::::::::::::::
                       precision    recall  f1-score   support

                    0       0.96      0.96      0.96      2076
                    1       0.96      0.96      0.96      2077

             accuracy                           0.96      4153
            macro avg       0.96      0.96      0.96      4153
         weighted avg       0.96      0.96      0.96      4153

         confusion matrix ::::::::::::::
          [[1987   89]
          [  77 2000]]
```



```
In [52]: sklearn.metrics.plot_roc_curve(lr, x_test, y_test)
         plt.show()
```

Conclusion

- We have got Logistic Regression as best model since it’s giving us good result and other metrics are also satisfactory.
- Using Logistic Regression as our final algorithm we have predicted the values for test dataset and it’s also working well and is able to identify fake news.
- From displaying the data, it seems there is lot of special characters present in the data. So, it is better to proceed by filter it out.
- As the above data is in text, so presence of special characters and stopwords is always there.
- After proper cleaning and processing, decision tree classifier gives the highest accuracy as well as ROC Score.