Detecting Fake News Using Natural Language Processing

Submitted by:

Aditya Paruleker

INTRODUCTION

- ## Business Problem Framing

  With lot of online content being created with each passing day, circulation of fake news has become one of the biggest problems in this era; it has a very serious impact not only on our online and offline discourse but also on the stability of this very society we live in. Data science can thus play a key role in this domain by protecting the online readers from the fake news using various natural language processing techniques. The primary objective is to use this data science knowledge to build a model to detect the fake news.

- ## Motivation for the Problem Undertaken

  Motivation behind this project is to help the people in telling apart the fake news from real. The information obtained from the model can help readers get the reliable news or day today updates that they can put their trust on.

Analytical Problem Framing

- Data Sources and their formats

  There are two dataset, one containing true news with 21417 rows and 4 columns and fake news with 23481 rows and 4 columns. Both the dataset are obtained from the kaggle website.

- Preprocessing Done

  We create and insert a label column named ‘report’ in both the dataset. In fake news dataset the report column will have all the input values as zeros whereas for true news dataset the report column will have all the input values as ones. Both the dataset are then combined using pandas built-in function before being analyzed to acquire basic insight. Once basic information is gained we proceed to the next phase,

  a) Null Value Analysis

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

  No null values are detected in the dataset. Since all the data is object type we skip the statistical analysis

  b) Data Cleaning

  - Strip all the html tags using beautiful soup html parser
  - Remove all the .com and https links.
  - Split the text and convert it to lower case.
  - Remove all the numbers from the string.
  - Using regex function, strip anything other than words and whitespaces like symbols.

- Eliminate the stopwords and punctuation from the string and then lemmatize the text.
- Finally remove any single character if remaining in the text.

c) Separating Target and Input features

Before proceeding with further preprocessing, the input features labelled x are separated from target features labelled y.

d) Vectorize

In NLP, word vectorization is a method i.e. used to map words or phrases from vocabulary to a corresponding vector of real numbers which used to find word predictions, word similarities/semantics. And this process of converting words into numbers is called Vectorization.

e) TF – IDF (Term frequency-inverse document frequency)

It evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document and the inverse document frequency of the word across a set of documents.

- Hardware and Software Requirements and Tools Used

Following libraries were used in this project,

Pandas

Created by Wes McKinney in 2008, this python library is widely used for data manipulation, analyzing and cleaning of data. Apart from this, it also helps in finding correlation between columns and in deleting rows.

Numpy
Created by Travis Oliphant in 2005, this python library provides an array object called ndarray i.e. up to 50x faster than traditional python lists. It was has functions can be used in linear algebra, matrices etc

Seaborn
Seaborn is a high level interface based data visualization library that uses matplotlib library underneath the working.

Matplotlib
Unlike saeborn, matplotlib is a low level data visualization python library. Majority of its function lies in the submodule named pyplot

Scikit-Learn
It provides tools for classification, regression, clustering and dimensionality reduction through its interface in python.

Pickle
Used for serializing (pickling) and de-serializing (unpickling) python object structure so that the data can be easily transferred from system to system and then stored in a file.

NLTK
NLTK is a standard python library that provides a set of diverse algorithms for NLP.

BeautifulSoup
Beautiful Soup is a Python library for pulling data out of HTML and XML files.

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

  In this study several machine learning models were tested and their results were compared before selecting the model that gave best result among all.

  Logistic Regression
  Using logistic approach for modelling, this supervised machine learning model aims to solve classification problems by predicting categorical outcomes, unlike linear regression that predicts a continuous outcome.

  KNeighbor Classifier
  The K-NN algorithm used in this classification stores all the available data and classifies a new data point based on the similarity. It is non parametric since it does not make any underlying assumption.

  Support Vector Classifier
  The SVC approach finds a hyperplane that creates a boundary between two classes of data to classify them.

  Ensemble Methods
  Ensemble technique uses several base estimators of machine learning model to predict the output. This results are then combined to improve the robustness of model over single estimator.

  Randomforest Classifier
  Using multiple decision trees as base, the model randomly performs the dataset sampling over the rows and features such that it form sample datasets for every model.

GradientBoost Classifier

The model after calculating the residual i.e. the difference between actual value and predicted target value; trains the weak model for mapping the features to that residual.

- Run and Evaluate models

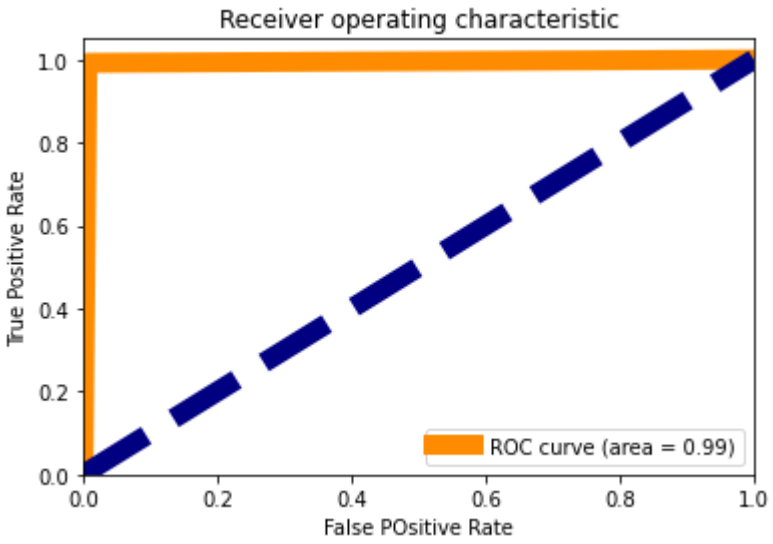
Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import r2_score,accuracy_score,classificatio
n_report,auc,roc_curve
from sklearn.model_selection import train_test_split,cross_val_sc
ore

lg = LogisticRegression()

x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,
random_state=42)
lg.fit(X,y)
pred_train=lg.predict(x_train)
pred_test=lg.predict(x_test)
print("Train Accuracy : ",round(accuracy_score(y_train,pred_train
)*100,2))
print("Test Accuracy : ",round(accuracy_score(y_test,pred_test)*1
00,2))
```

Output

```
Train Accuracy :  99.35
Test Accuracy :  99.29
cv score :  99.35
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      4733
           1       0.99      0.99      0.99      4247

    accuracy                           0.99      8980
   macro avg       0.99      0.99      0.99      8980
weighted avg       0.99      0.99      0.99      8980
```

```
Confusion Matrix
 [[4698   35]
 [  29 4218]]
```



Model Selection User Defined Function

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve,auc

def model_selection(algorithm_instance,x_train,y_train,x_test,y_t
est):
algorithm_instance.fit(x_train,y_train)
model_pred_train=algorithm_instance.predict(x_train)
 model_pred_test=algorithm_instance.predict(x_test)
 print("Accuracy of training model :",round(accuracy_score(y_train
 ,model_pred_train)*100,2))
 print("Accuracy of test data :",round(accuracy_score(y_test,model
 _pred_test)*100,2))

 cv_score=cross_val_score(algorithm_instance,X,y,cv=5)
 print("cv score : ", round(cv_score.mean()*100,2))
print("\nClassification report for test data\n",classification_repo
 rt(y_test,model_pred_test))
print("Classification report for training data\n",classification_re
 port(y_train,model_pred_train))
print("Confusion Matrix\n",confusion_matrix(y_test,model_pred_test))
print("\n")

fpr, tpr, thresholds = roc_curve(model_pred_test,y_test)
roc_auc= auc(fpr,tpr)
plt.figure()
```

```
    plt.plot(fpr,tpr,color='darkorange',lw=10,label='ROC curve (area =
%0.2f)' % roc_auc)
    plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0,1.05])
    plt.xlabel("False POsitive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("Receiver operating characteristic")
    plt.legend(loc='lower right')
    plt.show()
```

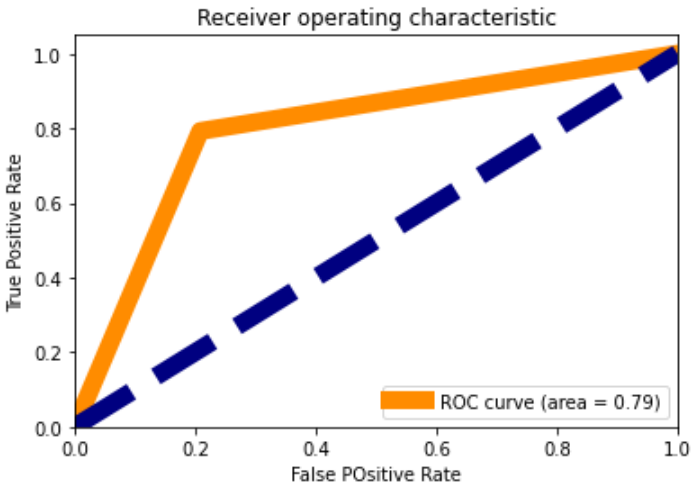
K Neighbour Classifier

```
from sklearn.neighbors import KNeighborsClassifier
k=KNeighborsClassifier()
model_selection(k,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 84.42
Accuracy of test data : 79.21
cv score :  74.24

Classification report for test data
              precision    recall  f1-score   support

           0       0.79      0.82      0.81      4733
           1       0.79      0.76      0.78      4247

    accuracy                           0.79      8980
   macro avg       0.79      0.79      0.79      8980
weighted avg       0.79      0.79      0.79      8980

Classification report for training data
              precision    recall  f1-score   support

           0       0.84      0.87      0.85     18748
           1       0.85      0.82      0.83     17170

    accuracy                           0.84     35918
   macro avg       0.84      0.84      0.84     35918
weighted avg       0.84      0.84      0.84     35918
```

```
Confusion Matrix
 [[3889  844]
 [1023 3224]]
```



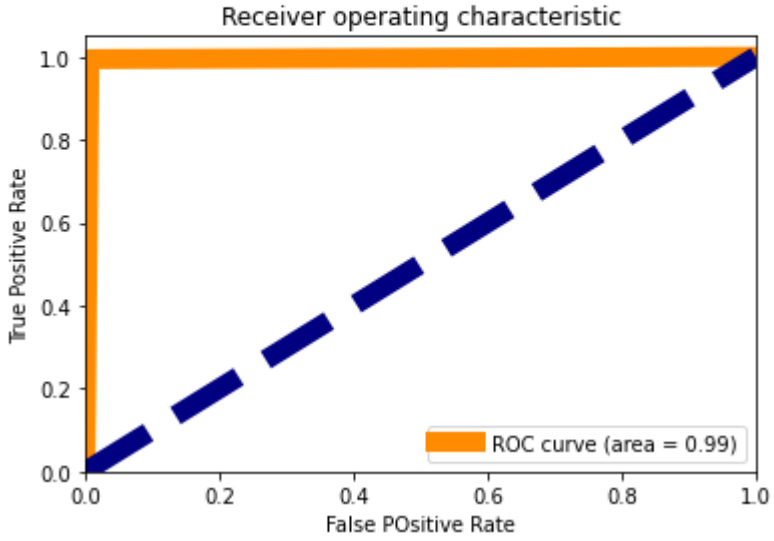
Support Vector Classifier

```
from sklearn import svm
s=svm.SVC()
model_selection(s,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 99.88
Accuracy of test data : 99.41
cv score :  99.04

Classification report for test data
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      4733
           1       0.99      0.99      0.99      4247

    accuracy                           0.99      8980
   macro avg       0.99      0.99      0.99      8980
weighted avg       0.99      0.99      0.99      8980

Classification report for training data
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     18748
           1       1.00      1.00      1.00     17170

    accuracy                           1.00     35918
   macro avg       1.00      1.00      1.00     35918
weighted avg       1.00      1.00      1.00     35918
```

```
Confusion Matrix
 [[4709   24]
 [  29 4218]]
```



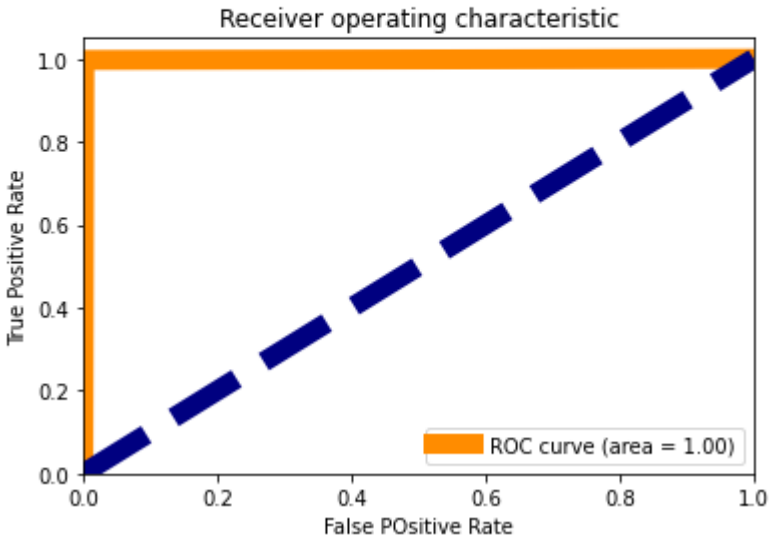
Decisiontree Classifier

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
model_selection(dtc,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 100.0
Accuracy of test data : 99.77
cv score :  99.48

Classification report for test data
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4733
           1       1.00      1.00      1.00      4247

    accuracy                           1.00      8980
   macro avg       1.00      1.00      1.00      8980
weighted avg       1.00      1.00      1.00      8980

Classification report for training data
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     18748
           1       1.00      1.00      1.00     17170

    accuracy                           1.00     35918
   macro avg       1.00      1.00      1.00     35918
weighted avg       1.00      1.00      1.00     35918
```

```
Confusion Matrix
 [[4720   13]
 [   8 4239]]
```



Randomforest Classifier

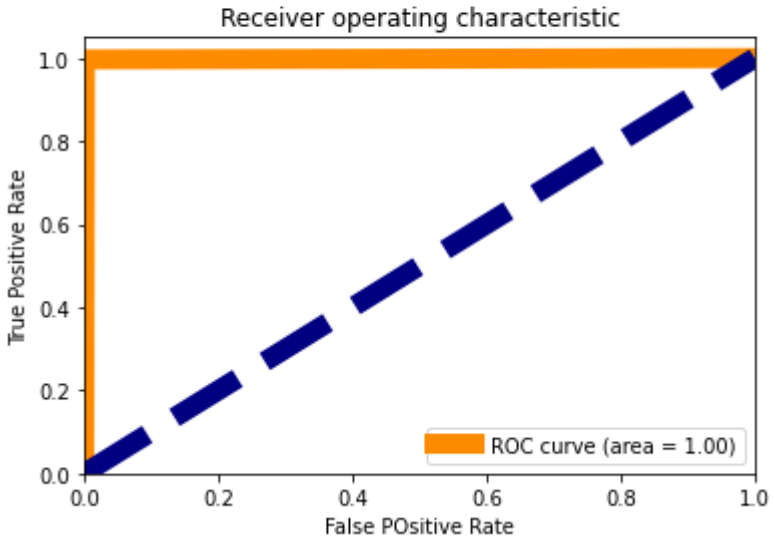
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

parameter={'criterion':['gini', 'entropy', 'log_loss'],
     'max_features' : [None,"sqrt","log2"],
      'class_weight':['balanced',' balanced_subsample'],
      'n_estimators':range(0,100,50)}
rf=RandomForestClassifier()
clf=GridSearchCV(rf,parameter)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

```
{'class_weight': 'balanced', 'criterion': 'entropy', 'max_features': None, 'n_estimators': 50}
```

```
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n_estimators=50, class_weight= 'balanced
',criterion='entropy', max_features= None)
model_selection(rf,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 99.99
Accuracy of test data : 99.79
cv score :  99.59

Classification report for test data
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4733
           1       1.00      1.00      1.00      4247

    accuracy                           1.00      8980
   macro avg       1.00      1.00      1.00      8980
weighted avg       1.00      1.00      1.00      8980

Classification report for training data
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     18748
           1       1.00      1.00      1.00     17170

    accuracy                           1.00     35918
   macro avg       1.00      1.00      1.00     35918
weighted avg       1.00      1.00      1.00     35918


Confusion Matrix
 [[4719   14]
 [   5 4242]]
```



GradientBoost Regressor

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier

parameter={'loss':['log_loss', 'deviance', 'exponential'],
     'criterion':['friedman_mse', 'squared_error', 'mse'],
     'max_features':['auto', 'sqrt', 'log2'],
     'n_estimators':range(0,100,50)}

rf3=GradientBoostingClassifier()
clf=GridSearchCV(rf3,parameter)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

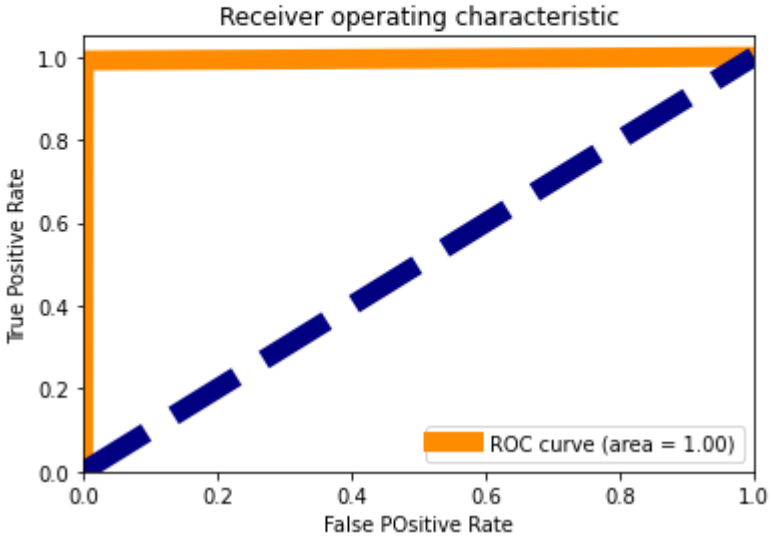
```
{'criterion': 'friedman_mse', 'loss': 'deviance', 'max_features': 'auto', 'n_estimators': 50}
```

```
rf3=GradientBoostingClassifier(criterion='friedman_mse', loss='deviance
', max_features= 'auto', n_estimators= 50)
model_selection(rf3,x_train,y_train,x_test,y_test)
```

Output

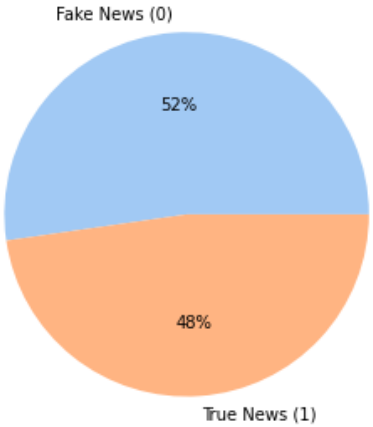
```
Accuracy of training model : 99.53
Accuracy of test data : 99.53
cv score :  99.35

Classification report for test data
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      4733
           1       0.99      1.00      1.00      4247

    accuracy                           1.00      8980
   macro avg       1.00      1.00      1.00      8980
weighted avg       1.00      1.00      1.00      8980

Classification report for training data
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     18748
           1       0.99      1.00      1.00     17170

    accuracy                           1.00     35918
   macro avg       1.00      1.00      1.00     35918
weighted avg       1.00      1.00      1.00     35918
```

```
Confusion Matrix
 [[4694   39]
 [   3 4244]]
```

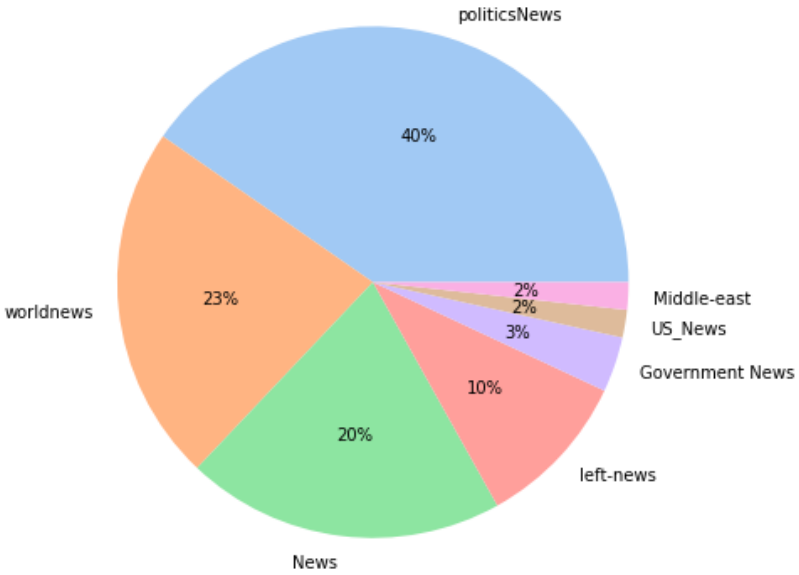


- Dataset Visualizations

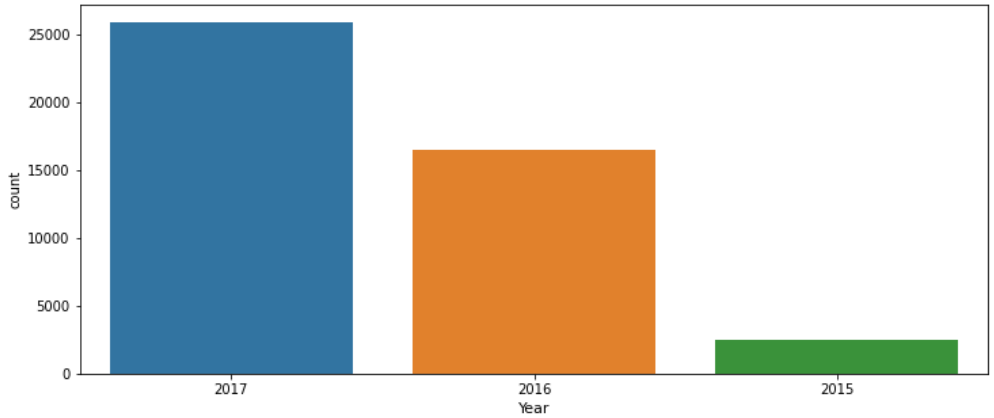
```
Number of fake and true news
0    23481
1    21417
Name: report, dtype: int64
```



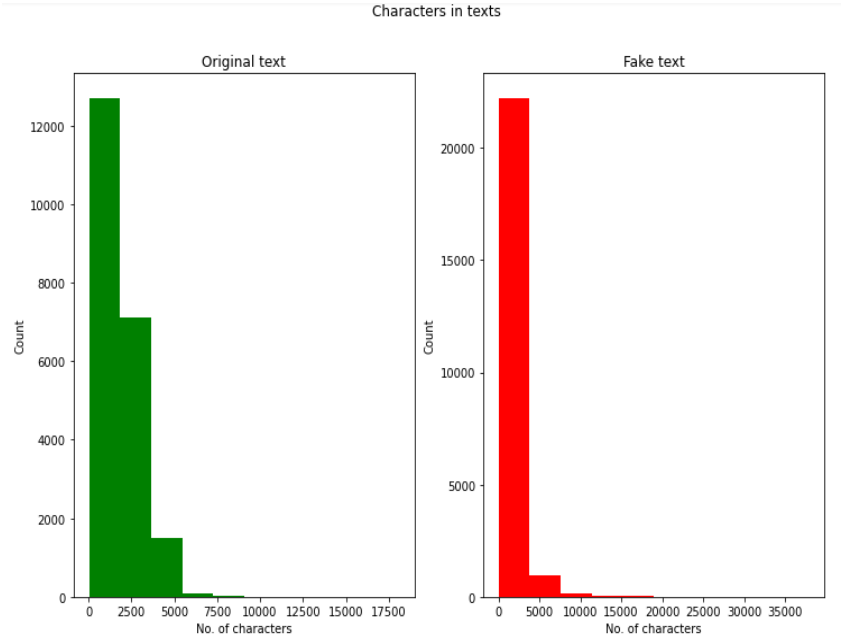
```
Number of news in each subject
politicsNews       18113
worldnews          10145
News                9050
left-news           4459
Government News     1570
US_News              783
Middle-east          778
Name: subject, dtype: int64
```

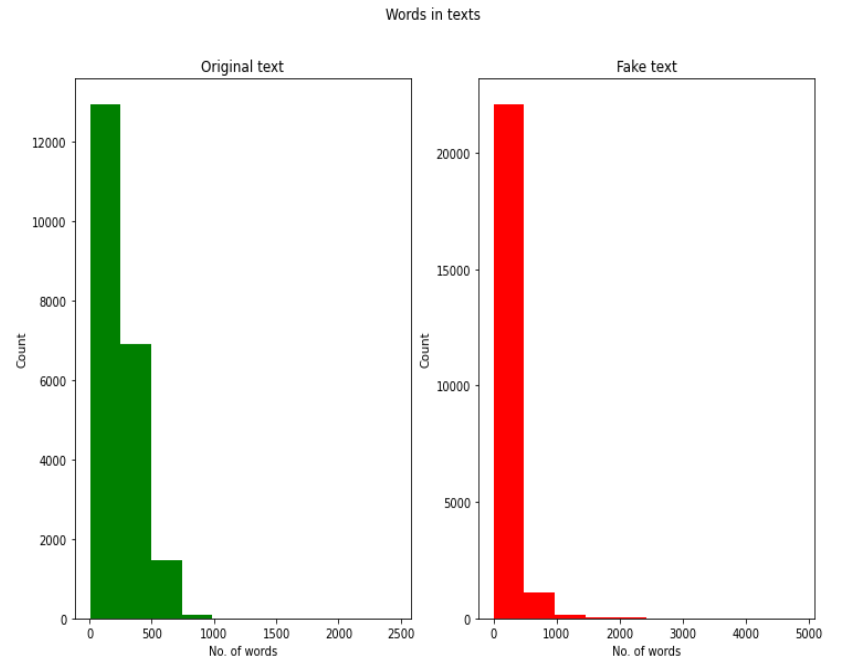


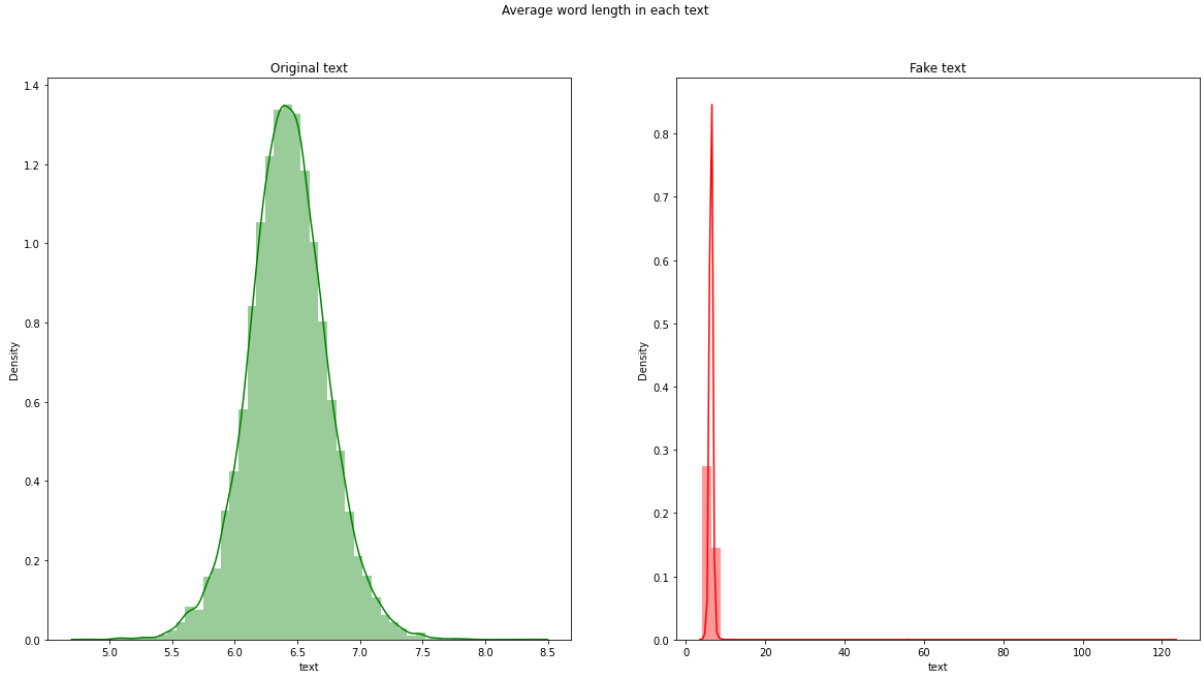

```
Number of news in each year
2017    25904
2016    16470
2015     2485
Name: 0, dtype: int64
```



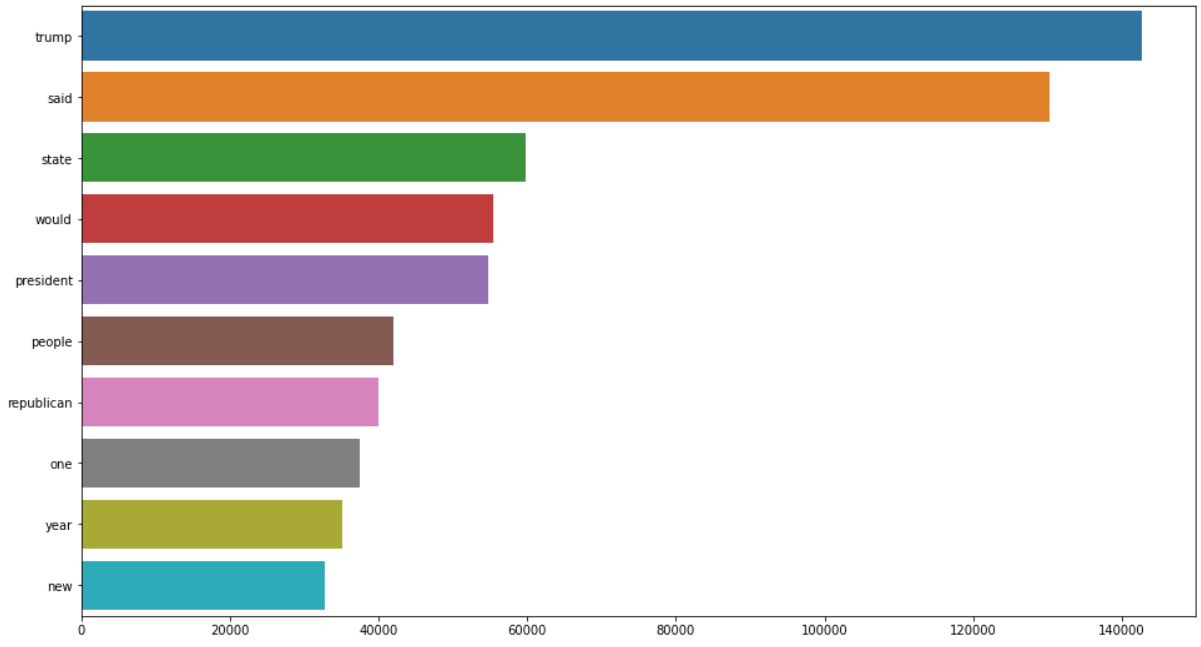
- Post Data cleaning Visualizations



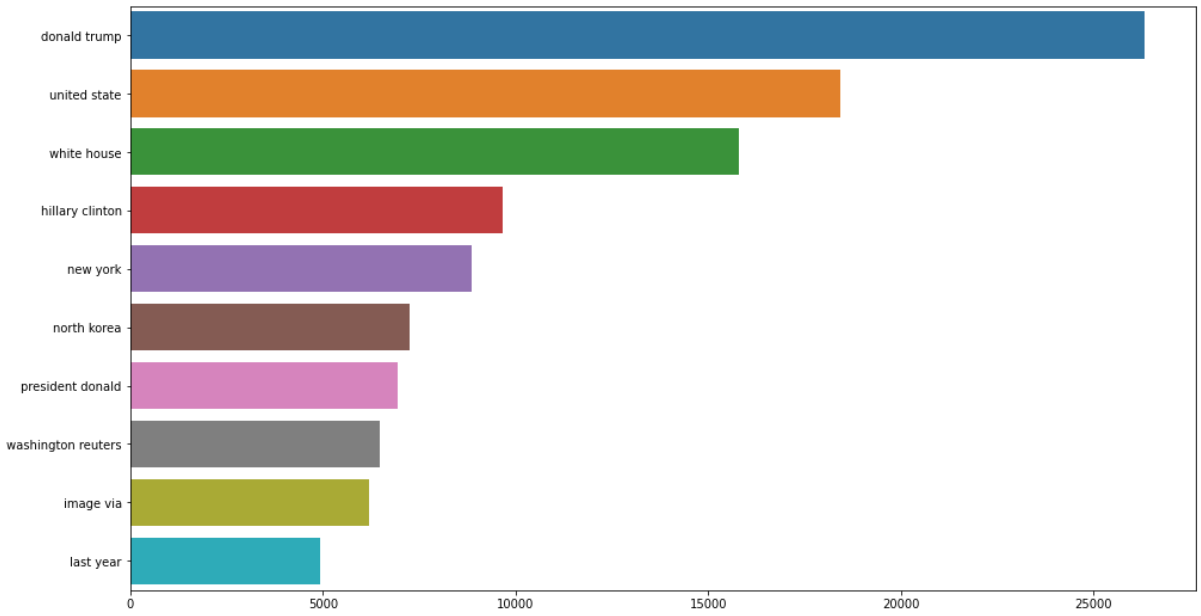




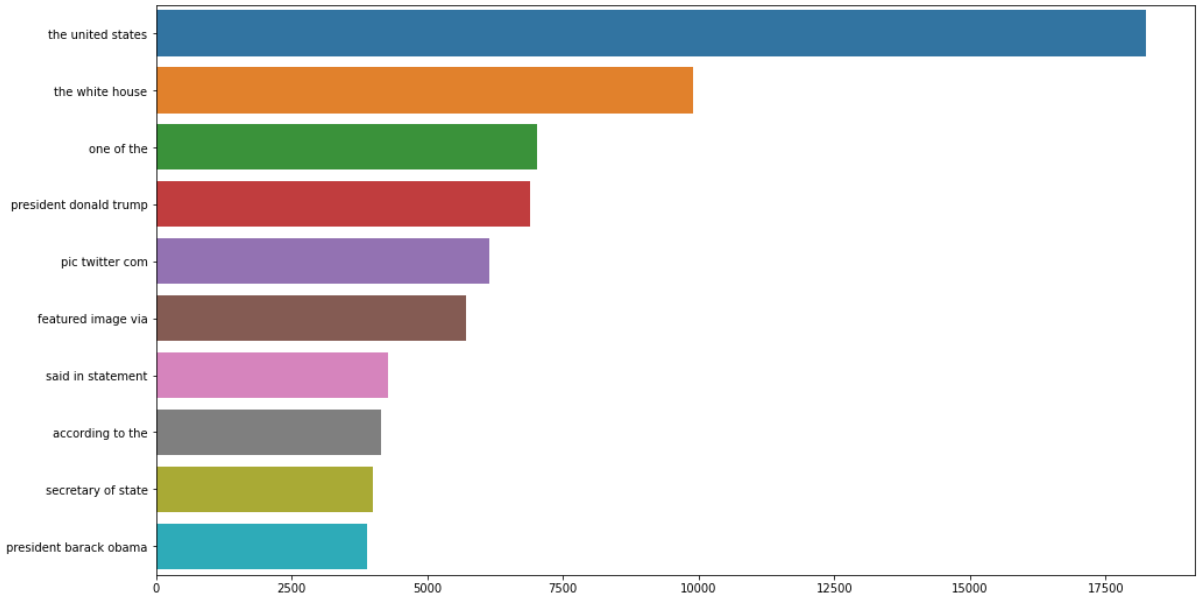
a) Unigram Analysis



b) Bigram Analysis



c) Trigram Analysis



- 2000 characters in text is most common in original text category while approx 4000 characters in text is most common in fake text category.
- 250 words in text is most common in original text category while approx 500 words in text is most common in fake text category.
- The average word length distribution in original text is greater than fake text.
- In unigram analysis, word 'trump' is found to have highest number of repetition.
- In bigram analysis, 'donald trump' is found to have highest number of occurrence.
- In Trigram analysis, 'the united state' is found to have highest number of occurrence.

CONCLUSION

- Key Findings and Conclusions of the Study

- In this study we found that randomforest classifier performs slightly better than logistic regression and rest of the algorithm tested.
- Trump is the most repeated word in entire dataset.
- Average length of text in original news is higher than the fake news text.

Thus we were successfully able to detect the fake news with minimum error.