FLIP ROBO

FAKE NEWS DETECTION PROJECT

Submitted by

Raganee Verma

ACKNOWLEDGMENT

I would like to thank again Flip Robo Technologies as they gave me opportunity to do this work. I didn't know anything about Natural Language Processing before this project but now I have learned a lot. I am also thankful to Miss Khushboo Garg for her guidance and support.

Some of the reference sources are as follows:

- You Tube
- Stack Overflow
- Educative.io
- GitHub
- Coding Ninjas
- Google image

INTRODUCTION

BUSINESS PROBLEM FRAMING

Fake news's simple meaning is to incorporate information that leads people to the wrong path. Nowadays fake news spreading like water and people share this information without verifying it. This is often done to further or impose certain ideas and is often achieved with political agendas.

For media outlets, the ability to attract viewers to their websites is necessary to generate online advertising revenue. So, it is necessary to detect fake news.

REVIEW OF LITERATURE

Internet is one of the important inventions and a large number of persons are its users. These persons use this for different purposes. There are different social media platforms that are accessible to these users. Any user can make a post or spread the news through the online platforms. These platforms do not verify the

users or their posts. So, some of the users try to spread fake news through these platforms. This news can be propaganda against an individual, society, organization or political party. A human being is unable to detect all these fake news. So, there is a need for machine learning classifiers that can detect this fake news automatically. Use of machine learning classifiers for detecting fake news is described in this systematic literature review.

MOTIVATION FOR THE PROBLEM UNDERTAKEN

The fake news on social media and various other media is wide spreading and is a matter of serious concern due to its ability to cause a lot of social and national damage with destructive impacts. A lot of research is already focused on detecting it. This project makes an analysis of the research related to fake news detection and explores the traditional machine learning models to choose the best, in order to create a model of a product with supervised machine learning algorithm, that can classify fake news as true or false, by using tools like python scikit-learn, NLP for textual analysis. This process will result in feature extraction and vectorization; we propose using Python scikit-learn library to perform tokenization and feature extraction of text data, because this library contains useful tools like Count Vectorizer and Tiff Vectorizer. Then, we will perform feature selection methods, to experiment and choose the best fit features to obtain the highest precision, according to confusion matrix results.

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL/ ANALYTICAL MODELING OF THE PROBLEM

- The dataset provided here has a two file one is Fake news which contain 23481 rows and 4 columns and another is True news which contain 21417 rows and 4 columns
- Firstly, we merge both file and append label column for target variable

- 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 tiff 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 not contain 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
- After merging both true and fake dataset the sample data for reference is shown below.
- This is our final dataset which contain 44898 rows and 5 columns.

	title	text	subject	date	label
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017	0
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017	0
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017	0
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017	0
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017	0

Dataset description

There are 5 columns in the dataset provided:

The description of each of the column is given below:

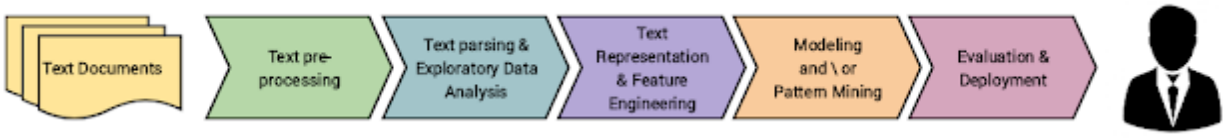
1. **title**: It is the title of the news.
2. **text**: It contains the full text of the news article.
3. **subject**: It represents the subject of the news article.

4. **date**: It represents the date of the news article published.
5. **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

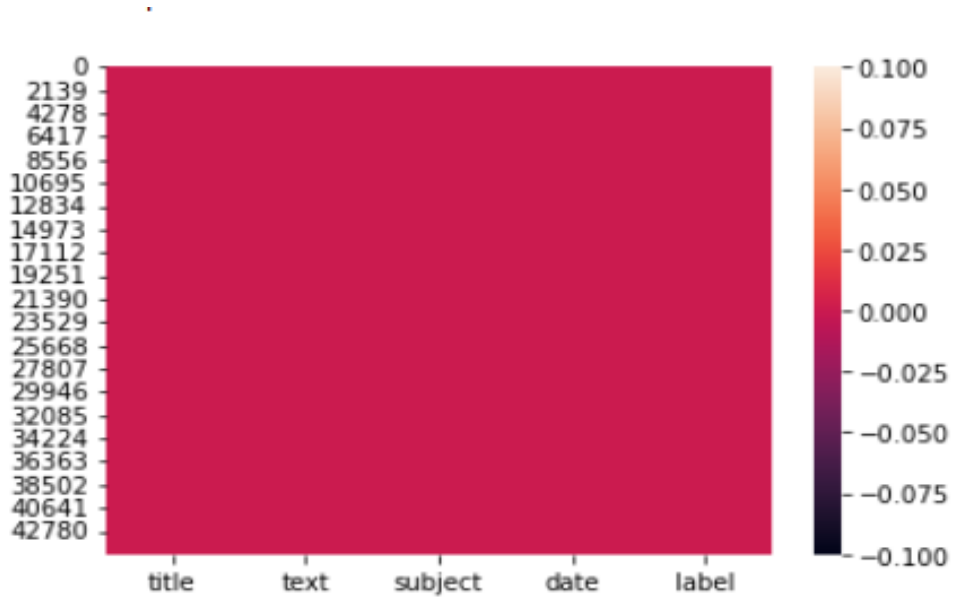


The above diagram shows how this pipeline generated numerical features and feed it into a machine learning algorithm. In this project, we are using some machine learning and Natural language processing libraries like NLTK, re (Regular Expression), Scikit Learn

DATA PREPROCESSING

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 null value by using isnull function and heatmap also but We found that no any missing value are present there.



- Then checked datatype of various features & found that all features are object datatype but here date is not object to so we can convert it into datetime.

```
1  # check datatypes and memory captured
2  df.info()
```

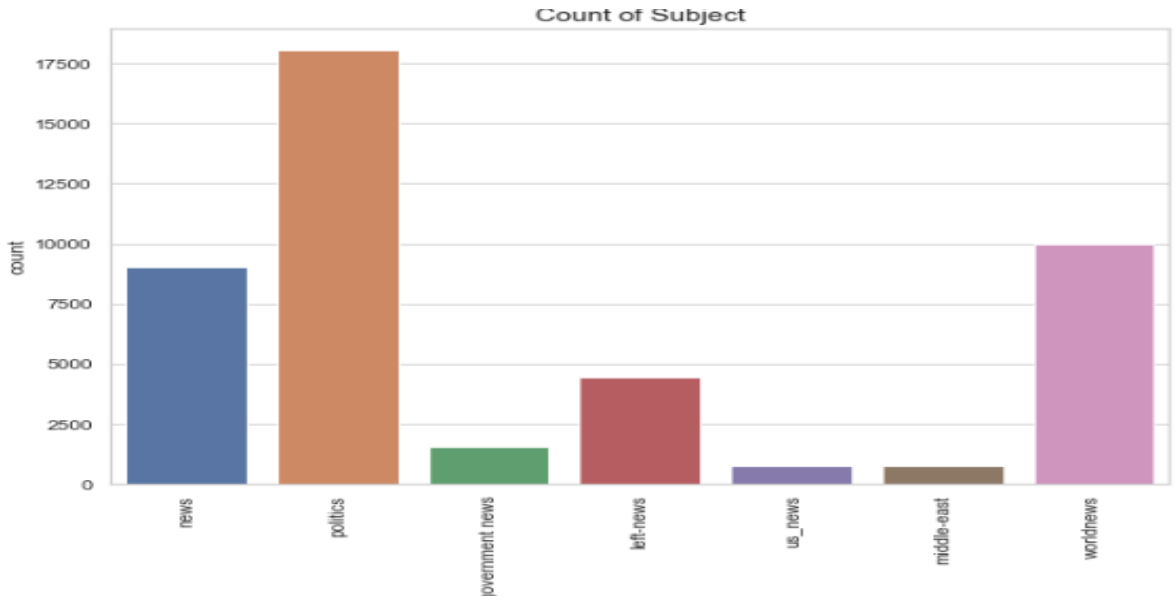
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44689 entries, 0 to 44897
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   title    44689 non-null  object
 1   text     44689 non-null  object
 2   subject  44689 non-null  object
 3   date     44689 non-null  object
 4   label    44689 non-null  int64 
dtypes: int64(1), object(4)
memory usage: 2.0+ MB
```

- Checking for unique values in each column and we observed that so many unique values present in title, text, and date column. 8 unique value presents in subject and 2 unique values in our target columns.

	Unique Values
title	38729
text	38646
subject	8
date	2397
label	2

- Checking distribution of fake and real news and observed that 47% of the news is fake.
- We see that both news is equally distributed. dataset is balanced which is good as it will help our model to classify more accurately, so we should expect a good accuracy score and no need to use oversampling or under sampling method.

- From count plot of subject, we observed that most of the news are politics background.



- Rename title as headline and text as news for better understanding and find the length of string

```
df.rename(columns = {'title':'headline'}, inplace = True)
df.rename(columns = {'text':'news'}, inplace = True)
```

```
#New columns for checking length of headline and news feature
df['length_headline'] = df.headline.str.len()
df['length_news'] = df.news.str.len()
df.head()
```

	headline	news	subject	date	label	length_headline	length_news
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	12-31-2017	0	79	2893
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	12-31-2017	0	69	1898
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	12-30-2017	0	90	3597
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	12-29-2017	0	78	2774
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	12-25-2017	0	70	2346

Natural Language Processing-

Machine learning data only works with numerical features so we have to convert text data into numerical columns. So, we have to preprocess the text and that is called natural language processing.

In-text preprocess we are cleaning our text by steaming, lemmatization, remove stopwords, remove special symbols and numbers, etc. After cleaning the data, we have to feed this text data into a vectorizer which will convert this text data into numerical features.

- 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

We can't use text data directly because it has some unusable words and special symbols and many more things. If we used it directly without cleaning then it is very hard for the ML algorithm to detect patterns in that text and sometimes it will also generate an error. So that we have to always first clean text data. In this project, we are making one function 'cleaning data' which cleans the data

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

```
# 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 stopwords.words('english')]

   # 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]

   # 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
```

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

Adding additional attribute:

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

```
#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)
```

	headline	news	subject	date	label	length_headline	length_news	clean_news	clean_headline	clean_length_headline	clean_length_news
0	donald trump sends out embarrassing new year'...	donald trump just couldn t wish all americans ...	news	12-31-2017	0	79	2893	donald trump wish american happy year leave in...	donald trump sends embarrassing year message d...	55	1809
1	drunk bragging trump staffer started russian ...	house intelligence committee chairman devin nu...	news	12-31-2017	0	69	1898	house intelligence committee chairman devin nu...	drunk bragging trump staffer started russian c...	68	1277
2	sheriff david clarke becomes an internet joke...	on friday, it was revealed that former milwauk...	news	12-30-2017	0	90	3597	friday revealed former milwaukee sheriff david...	sheriff david clarke becomes internet joke thr...	66	2266
3	trump is so obsessed he even has obama's name...	on christmas day, donald trump announced that ...	news	12-29-2017	0	78	2774	christmas donald trump announced would back wo...	trump obsessed even obama name coded website i...	50	1790
	pope francis	pope francis						pope francis	pope francis		

```
# 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 3582955
Clean Length 2641127
Total Reduction =  941828
```

```
# 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 110252173
Clean Length 69790364
Total Reduction =  40461809
```

After doing all these steps, we found that all the words & special characters were removed from the dataset which is no use and consuming memory.

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

- Windows 10 (64-bit Operating System)

Device specifications

Device name	DESKTOP-SFEGRG9
Processor	Intel(R) Core(TM) i5-4300U CPU @ 1.90GHz 2.50 GHz
Installed RAM	4.00 GB (3.90 GB usable)
Device ID	93A5718E-1C43-4455-B1AF-FDB798AB4F0F
Product ID	00330-80000-00000-AA718
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

- Jupyter Notebook (Anaconda 3) – Python 3.7.6
- Microsoft Excel 201

LIBRARIES:

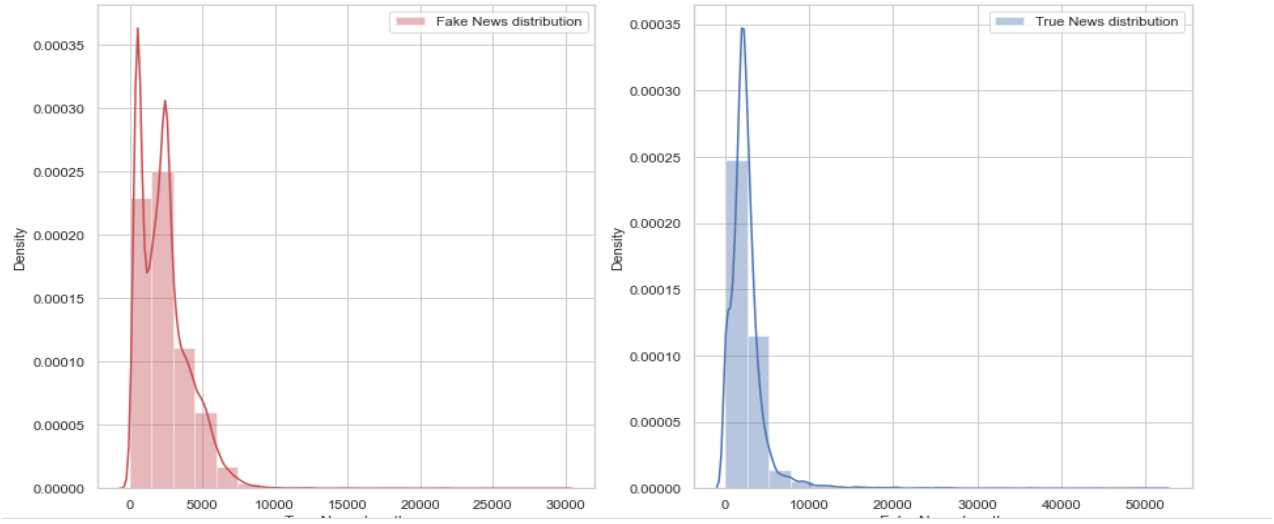
- ✓ Pandas: To read the Data file in form of data.
- ✓ Numpy: NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices
- ✓ Matplotlib: This library is typically used to plot the figures for better visualization 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.
- ✓ Wordcloud: Wordcloud package helps us to know the frequency of a word in textual content using visualization.

- ✓ NLTK: Natural language took kit is one of the most used libraries for building NLP projects.

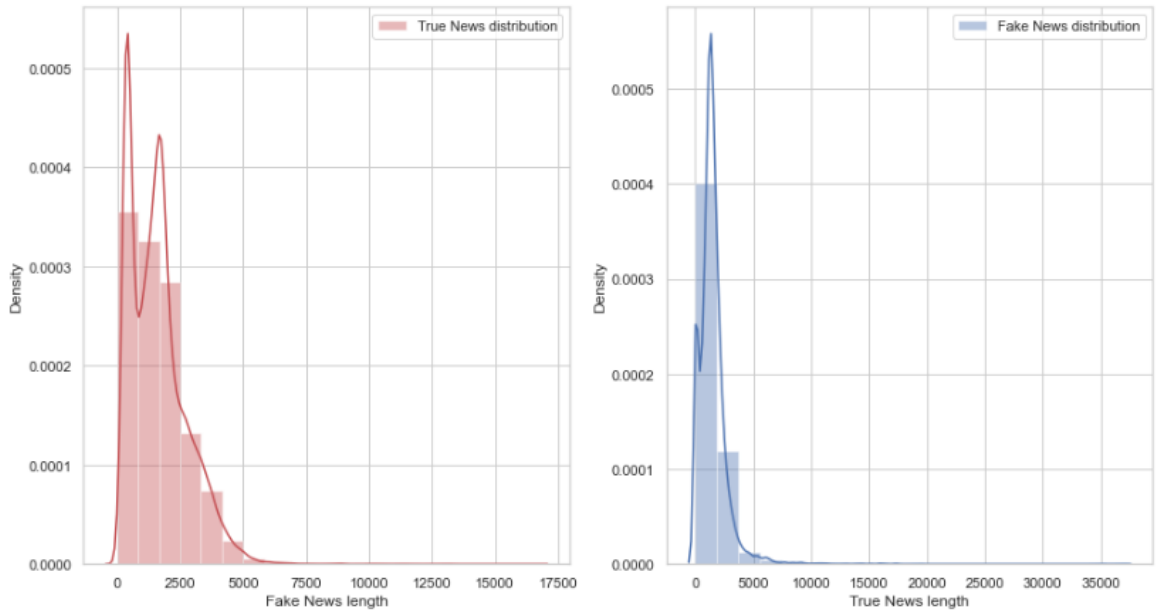
```
# Let's import the required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import re


from nltk.corpus import wordnet
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.stem import WordNetLemmatizer, SnowballStemmer
from nltk import pos_tag
from collections import Counter




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

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

Before cleaning:

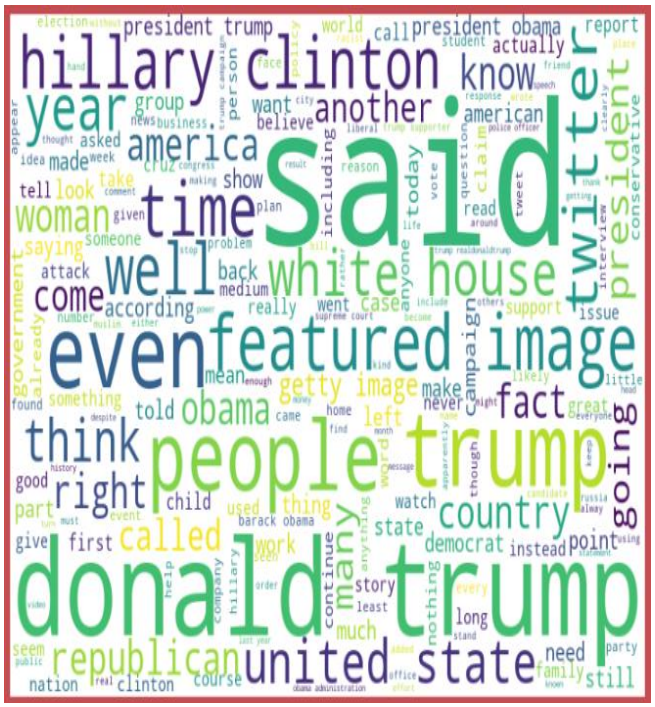


After cleaning:

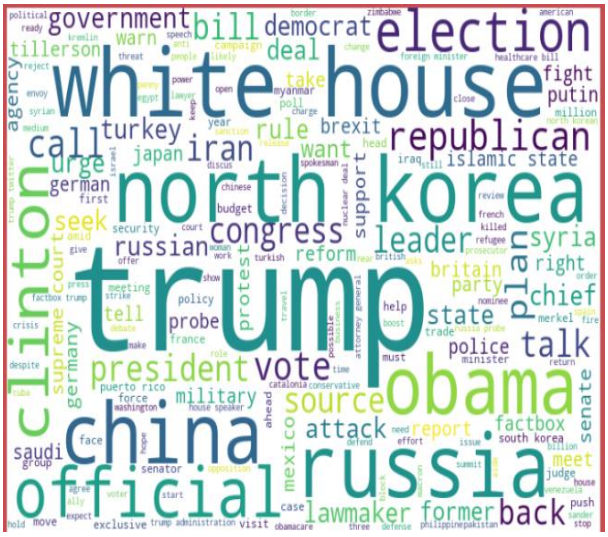


To get a better view of words contained in news. A word dictionary (word cloud) was made showing the words highly occurred in fake and real news for both headline and news column

<u>News columns</u>



Headlines columns



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.

MODEL/S DEVELOPMENT AND EVALUATION

IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES (METHODS)

After Understanding the dataset, we observed that the target variable label has two values so we can be concluded that it is a binary classification problem Therefore, I run my preprocessed data on 6 classification algorithms

6 different Classification Algorithms which are used to predict our model:

1. RandomForestClassifier
2. LogisticRegression
3. MultinomialNB
4. DecisionTreeClassifier
5. AdaBoostClassifier
6. GradientBoostingClassifier

Training Classifier:

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

TF (Term Frequency): The number of times a word appears in a document is its Term Frequency. A higher value means a term appears more often than others, and so, the document is a good match when the term is part of the search terms.

IDF (Inverse Document Frequency): Words that occur many times a document, but also occur many times in many others, may be irrelevant. IDF is a measure of how significant a term is in the entire corpus.

The TfidfVectorizer converts a collection of raw documents into a matrix of TF-IDF features.

```
# 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
def Tf_idf(text):
    tfid = TfidfVectorizer(min_df=2)
    return tfid.fit_transform(text)
```

```
# Inserting vectorized values in a variable x, which will be used in training the model
x=Tf_idf(df['subject'] + 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:  (44689, 72554)
```

TESTING OF IDENTIFIED APPROACHES (ALGORITHMS)

Import all the algorithms that is used for the training and testing are our model: -

```
# 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 sklearn.ensemble import GradientBoostingClassifier
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()
GB=GradientBoostingClassifier()
```

```
# 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(('GradientBoostingClassifier',GB))

```

RUN AND EVALUATE SELECTED MODELS

Firstly, we define methods which gives many information about all the models like:

Max_acuuracy_score,Best random state,classification metrics, log_loss,max_acc_score, Auc_roc_score,cross_val.

```
# 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
```

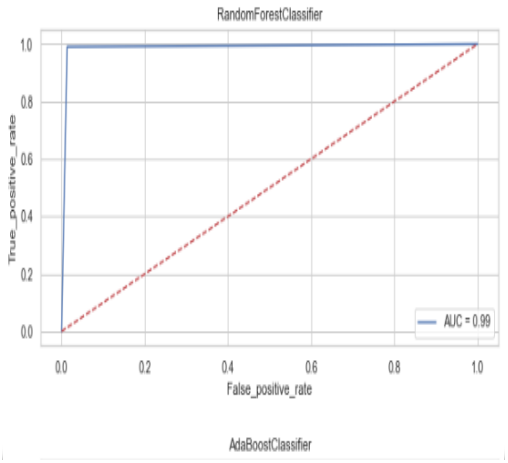
```
Model=[]
Score=[]
Acc_score=[]
cvs=[]
rocscore=[]
logloss=[]
#For Loop to Calculate Accuracy Score, Cross Val Score, Classification Report, Confusion Matrix,logloss

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')
```

Outcomes of the above models:

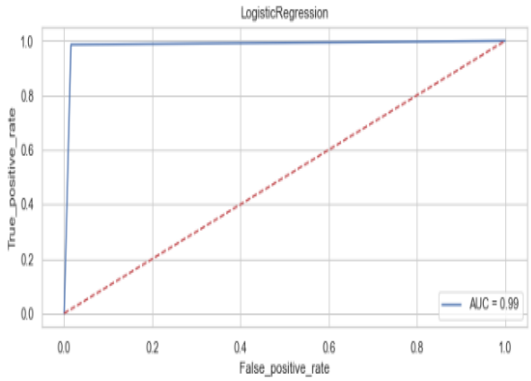
1. RandomForestClassifier:

```
Max Accuracy Score corresponding to Random State  47 is: 0.9896322816439174


Accuracy Score :  0.9882151115089133
Cross Val Score :  0.9988965023425835
roc auc score :  0.9883137741858878


Log loss :  0.40704135390088747
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      7044
           1       0.98      0.99      0.99      6363

    accuracy                           0.99     13407
   macro avg       0.99      0.99      0.99     13407
weighted avg       0.99      0.99      0.99     13407
```



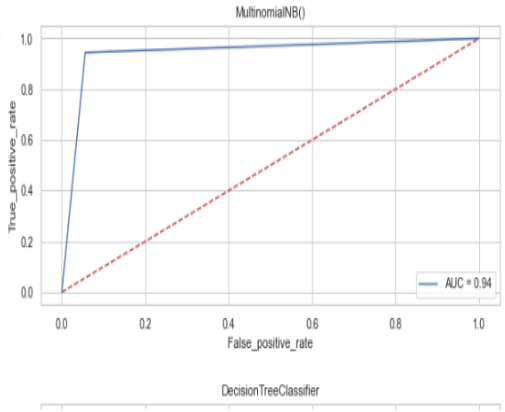
2. LogisticRegression:

```
Accuracy Score :  0.9851570075333781
Cross Val Score :  0.9971815932055517
roc auc score :  0.9852059767914002


Log loss :  0.5126654179087037
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      7044
           1       0.98      0.99      0.98      6363

    accuracy                           0.99     13407
   macro avg       0.99      0.99      0.99     13407
weighted avg       0.99      0.99      0.99     13407
```



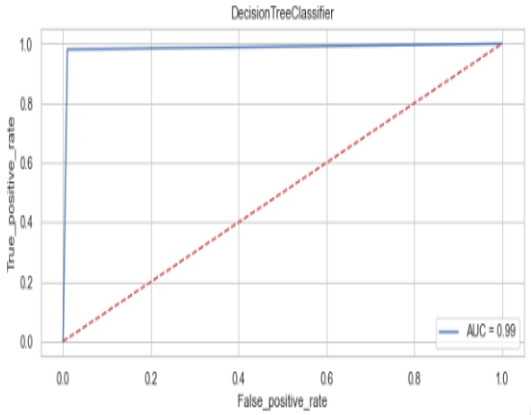
3. MultinomialNB:

```
Accuracy Score :  0.9439844857164168
Cross Val Score :  0.9780748774013299
roc auc score :  0.9439953243316542


Log loss :  1.9347309400432826
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.94      0.95      7044
           1       0.94      0.94      0.94      6363

    accuracy                           0.94     13407
   macro avg       0.94      0.94      0.94     13407
weighted avg       0.94      0.94      0.94     13407
```



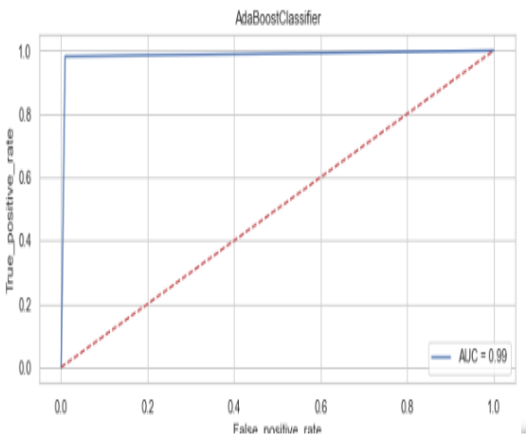
4. DecisionTreeClassifier:

```
Max Accuracy Score corresponding to Random State  70 is: 0.9856791228462743


Accuracy Score :  0.985604534944432
Cross Val Score :  0.9806364334718747
roc auc score :  0.9853659800148913


Log loss :  0.497205923475638 7
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      7044
           1       0.99      0.98      0.98      6363

    accuracy                           0.99     13407
   macro avg       0.99      0.99      0.99     13407
weighted avg       0.99      0.99      0.99     13407
```



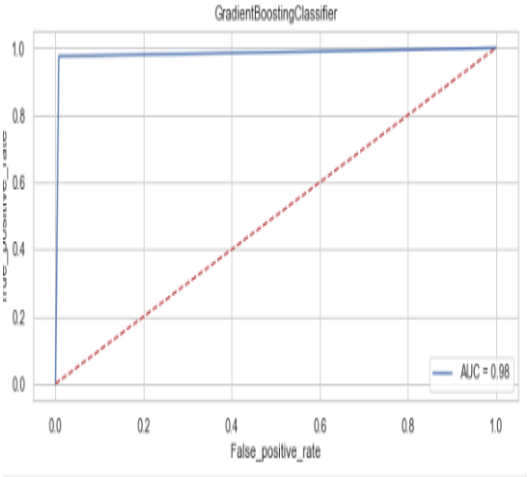
5. AdaBoostClassifier:

```
Max Accuracy Score corresponding to Random State  51 is: 0.9864250018646975


Accuracy Score :  0.9864250018646975
Cross Val Score :  0.9979299560586036
roc auc score :  0.9861999646058546


Log loss :  0.4688677614159879
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      7044
           1       0.99      0.98      0.99      6363

    accuracy                           0.99     13407
   macro avg       0.99      0.99      0.99     13407
weighted avg       0.99      0.99      0.99     13407
```



6. GradientBoostingClassifier

```
Max Accuracy Score corresponding to Random State  63 is: 0.9841127769075856


Accuracy Score :  0.9841873648094279
Cross Val Score :  0.9974837675544981
roc auc score :  0.9837514233292398


Log loss :  0.5461522916373766
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      7044
           1       0.99      0.98      0.98      6363

    accuracy                           0.98     13407
   macro avg       0.98      0.98      0.98     13407
weighted avg       0.98      0.98      0.98     13407
```

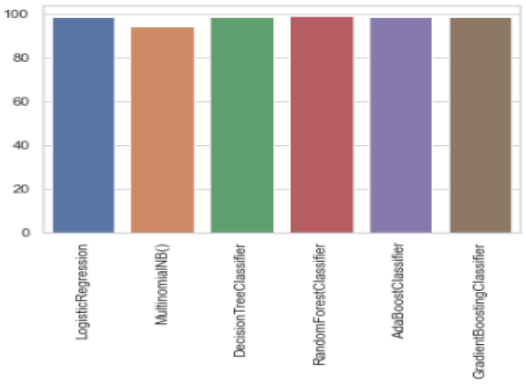


Comparison of all model performance

We compare performance of all model and find out the final model according to accuracy score and model performance.

We saw that the accuracy of Random Forest model is maximum and the difference between accuracy score and cross validation crore is minimum so we will choose the Rando Forest model is our best model and save it.

	Model	Accuracy Score	Cross Val Score	Log_Loss	Roc_Auc_curve
0	LogisticRegression	98.515701	99.718159	0.512665	98.520598
1	MultinomialNB()	94.398449	97.807488	1.934731	94.399532
2	DecisionTreeClassifier	98.560453	98.063643	0.497206	98.536598
3	RandomForestClassifier	98.821511	99.889650	0.407041	98.831377
4	AdaBoostClassifier	98.642500	99.792996	0.468868	98.619996
5	GradientBoostingClassifier	98.418736	99.748377	0.546152	98.375142



Final model selection

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

```
# Using Random ForestClassifier for final model...
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=47,test_size=.30)
RF=RandomForestClassifier()
RF.fit(x_train,y_train)
RF.score(x_train,y_train)
RFpred=RF.predict(x_test)
print('Accuracy Score:','\n',accuracy_score(y_test,RFpred))
print('Log_Loss:','\n',log_loss(y_test,RFpred))
print('Confusion Matrix:','\n',confusion_matrix(y_test,RFpred))
print('Classification Report:','\n',classification_report(y_test,RFpred))
```

```
Accuracy Score:
 0.9873200566868054
Log_Loss:
 0.4379558101046251
Confusion Matrix:
 [[6851  102]
 [  68 6386]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      6953
           1       0.98      0.99      0.99      6454

    accuracy                           0.99     13407
   macro avg       0.99      0.99      0.99     13407
weighted avg       0.99      0.99      0.99     13407
```

Save and prediction of the Model

After the good performance on data, we can save our model so that next time we can use it directly. ‘joblib’ and ‘pickle’ library used to save the machine learning model. From the Following step, you can save and load your model.

```
# Saving the best model.
import joblib
joblib.dump(RF,'Fake_news_Predict.pkl')
```

```
['Fake_news_Predict.pkl']
```

```
# Saving the Predicted values in csv file
pred_value.to_csv('Fake_news_Prediction.csv')
```

```
# Printing predicted values
pred_value=pd.DataFrame(data=y_test,)
pred_value['Predicted values']=RFpred
pred_value
```

	label	Predicted values
7445	0	0
12915	0	0
15057	0	0
6957	0	0
35344	1	1
...	...	...
21825	0	0
23238	0	0
38313	1	1
16661	0	0
5745	0	0

13407 rows × 2 columns

CONCLUSION

Today, we learned to detect fake news with Python. We took a Fake and True News dataset, implemented a Text cleaning function, TfidfVectorizer, initialized 6 different model from which Random Forest Classifier is our best model with the accuracy of 98.8%.