
Name Of the Project

# EMAIL SPAM CLASSIFIER

Submitted By

Mrs. Swati Amit Motugade

FlipRobo SME

Gulshana Chaudhari

# ACKNOWLEDGMENT

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analysis skills. Also, I want to express my huge gratitude to Ms. Gulshana Chaudhari Mam (SME, Flip Robo), she is the person who has helped me to get out of all the difficulties I faced while doing the project.

A huge thanks to “Data trained” who are the reason behind Internship at Fliprobo. Last but not least my parents who are there to support me at every step of my life.

References used in this project:

1. SCIKIT Learn Library Documentation.
2. Blogs from towardsdatascience, Analytics Vidya, Medium.
3. Andrew Ng Notes on Machine Learning (GitHub).
4. Data Science Projects with Python Second Edition by Packt
5. Hands on Machine learning with scikit learn and tensor flow by Aurelien Geron.
6. Markines, B., Cattuto, C., & Menczer, F. (2009, April). “Social spam detection”. In Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the web.
7. Kushal Agarwalla, Shubham Nandan, Varun Anil Nair, D. Deva Hema, “Fake News Detection using Machine Learning and Natural Language Processing,” International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-7, Issue-6, March 2019.
8. https://www.javatpoint.com/nlp
9. https://www.educative.io/answers/preprocessing-steps-in-natural-language-processing-nlp
10. https://www.youtube.com/watch?v=5ctbvkAMQO4

11. https://www.youtube.com/watch?v=X2vAabgKiuM

Mrs. Swati Amit Motugade

# CHAPTER 1

# INTRODUCTION

## 1.1 INTRODUCTION


Menace of fake news is increasing day by day due to increasing uses of social media. With the advent of new age digital and social media like Facebook, Twitter, WhatsApp, Instagram etc. fake news has literally pervaded all spheres of life. Increasing uses of social media made it possible that the information reaches to masses in very short time. But many times, this becomes the causes of various fake news circulation which leads to the possibility of potential violence, hatred and impacts the social fabric in many

ways. Now it is time that appeals for some uniform guidelines and policy to tackle the menace of fake news.

As a larger group of people in India using social media platforms such as Facebook, Twitter, WhatsApp, Instagram etc. actively. Through these social media platforms, fabricated and manipulated content are increasingly gaining ground in India. Circulation of these fabricated and manipulated content certainly leading to the possibility of potential violence, hatred and also impacting the social fabric in many ways. The menace of fake news is not new, it is prevailed since the emergence of print media. However, its potential of reach has magnified with new online social media platforms and applications. The increasing use of digital and social media is amplifying the effect of the menace of fake news. In recent days, many cases of fake news have been registered for sharing false content through social media messaging and many people were arrested for sharing rumour-mongering content. Far-seeing the impact of fake news, on some occasions, Government has shut down Internet on the pretext of inciting violence and to stop the spread of misinformation. The main reason behind circulation of fake news through social media is lack of proper regulation and its implementation. The online platforms do not fall under comprehensive regulation like the mainstream media. A large number of online news portals are being set up due to the lack of proper entry barriers. The lack of adequate binding rules offers a larger scope for wrongdoing in case of online platforms.

Despite some researches and investigations on fake news, credible information on the creators and the intention behind it is still untraced. However, the intension behind sharing misinformation/ fake news may be communal polarisation, political and economic gains.

The growing communal polarisation of society on ideological lines has made the job of spreading fake news through social media easier. In the contrary the spread of hatred content among leaders or groups of the opposing ideologies further deepens and accelerate the prevailing communal hatred through social media.

## 1.2. BUSINESS PROBLEM FRAMING

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.

## Objective

1. Our sole objective is to classify the news from the dataset as fake or true news.
2. Extensive EDA of news
3. Selecting and building a powerful model for classification

# 1.3 CONCEPTUAL BACKGROUND OF DOMAIN PROBLEM:

Fake news has become one of the biggest problems of our age. It has serious impact on our online as well as offline discourse. One can even go as far as saying that, to date, fake news poses a clear and present danger to western democracy and stability of the society.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.

In this project, we are using some machine learning and Natural language processing libraries like NLTK, re (Regular Expression), Scikit Learn.

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

In-text pre-process 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.

There are two datasets one for fake news and one for true news. In true news, there is 21417 news, and in fake news, there is 23481 news. You have to insert one label column zero for fake news and one for true news. We are combined both datasets using panda's built-in function.

## 1.4 REVIEW OF LITERATURE

Mykhailo Granik et. al. in their paper [3] shows a simple approach for fake news detection using naive Bayes classifier. This approach was implemented as a software system and tested against a data set of Facebook news posts. They were collected from three large Facebook pages each from the right and from the left, as well as three large mainstream political news pages (Politico, CNN, ABC News). They achieved classification accuracy of approximately 74%. Classification accuracy for fake news is slightly worse. This may be caused by the skewness of the dataset: only 4.9% of it is fake news. Himank Gupta et. al. [10] gave a framework based on different machine learning approach that deals with various problems including accuracy shortage, time lag (BotMaker) and high processing time to handle thousands of tweets in 1 sec. Marco L. Della Vedova et. al. [11] first proposed a novel ML

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

Co dy Buntain et. al. [12] develops a method for automating fake news detection on Twitter by learning to predict accuracy assessments in two credibility-focused Twitter datasets: CREDBANK, a crowd sourced dataset of accuracy assessments for events in Twitter, and PHEME, a dataset of potential rumors in Twitter and journalistic assessments of their accuracies. They apply this method to Twitter content sourced from BuzzFeed's fake news dataset. A feature analysis identifies features that are most predictive for crowd sourced and journalistic accuracy assessments, results of which are consistent with prior work. They rely on identifying highly retweeted threads of conversation and use the features of these threads to classify stories, limiting this work's applicability only to the set of popular tweets. Since the majority of tweets are rarely retweeted, this method therefore is only usable on a minority of Twitter conversation threads. In his paper, Shivam B. Parikh et. al. [13] aims to present an insight of characterization of news story in the modern diaspora combined with the differential content types of news story and its impact on readers. Subsequently, we dive into existing fake news detection approaches that are

heavily based on text-based analysis, and also describe popular fake news datasets. We conclude the paper by identifying 4 key open research challenges that can guide future research. It is a theoretical Approach which gives Illustrations of fake news detection by analysing the psychological factors

# CHAPTER 2

# ANALYTICAL PROBLEM FRAMING

## 2.1 MATHEMATICAL MODELLING OF THE PROBLEM

Our objective is to classify the news as fake or real using ML algorithms. In this project we are going to use different types of algorithms which uses their own mathematical equation on background. This project comes with a csv data set. Initially data cleaning & pre-processing performed over data. Feature engineering is performed to remove unnecessary feature & for dimensionality reduction. In model building Final model is select based on evaluation benchmark among different models with different algorithms. Further Hyperparameter tuning performed to build more accurate model out of best model.

## 2.2 DATA SOURCES AND THEIR FORMATS

We are provided two datasets by Flip Robo one is for real news and other for fake news which are in the format of CSV (Comma Separated Values).

Importing the necessary libraries for loading datasets, visualization, data pre-processing, NLTK libraries, machine learning libraries, metrics libraries etc.

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import rcParams
import seaborn as sns

!pip install textblob

import textblob
from textblob import TextBlob



from plotly import tools
import plotly.graph_objs as go
from plotly.offline import iplot
%matplotlib inline
plt.rcParams['figure.figsize'] = [10, 5]

!pip install cufflinks
import sys
import cufflinks as cf
cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)
```

```
#NLTK libraries
import nltk
import re
import string
from nltk.corpus import stopwords
from wordcloud import WordCloud,STOPWORDS
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
# Machine Learning libraries
import sklearn
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
```

```
#Metrics libraries
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

#Miscellanous libraries
from collections import Counter

#Deep learning libraries
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Bidirectional
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout

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

# LOADING THE DATASETS

## 1. Real news dataset

We are given the two datasets one for real news and other for fake news. Both are in csv format and loaded these datasets using panda's read function.

```
# #reading the real news dataset
df_real=pd.read_csv(r"C:\Users\Swati\OneDrive\Desktop\DataScience\Projects\Fake News Classifier\True.csv")
df_real
```

| | title | text | subject | date |
|---|---|---|---|---|
| 0 | As U.S. budget fight looms, Republicans flip t... | WASHINGTON (Reuters) - The head of a conservat... | politicsNews | December 31, 2017 |
| 1 | U.S. military to accept transgender recruits o... | WASHINGTON (Reuters) - Transgender people will... | politicsNews | December 29, 2017 |
| 2 | Senior U.S. Republican senator: 'Let Mr. Muell... | WASHINGTON (Reuters) - The special counsel inv... | politicsNews | December 31, 2017 |
| 3 | FBI Russia probe helped by Australian diplomat... | WASHINGTON (Reuters) - Trump campaign adviser ... | politicsNews | December 30, 2017 |
| 4 | Trump wants Postal Service to charge 'much mor... | SEATTLE/WASHINGTON (Reuters) - President Donal... | politicsNews | December 29, 2017 |
| ... | ... | ... | ... | ... |
| 21412 | 'Fully committed' NATO backs new U.S. approach... | BRUSSELS (Reuters) - NATO allies on Tuesday we... | worldnews | August 22, 2017 |
| 21413 | LexisNexis withdrew two products from Chinese ... | LONDON (Reuters) - LexisNexis, a provider of l... | worldnews | August 22, 2017 |
| 21414 | Minsk cultural hub becomes haven from authorities | MINSK (Reuters) - In the shadow of disused Sov... | worldnews | August 22, 2017 |
| 21415 | Vatican upbeat on possibility of Pope Francis ... | MOSCOW (Reuters) - Vatican Secretary of State ... | worldnews | August 22, 2017 |
| 21416 | Indonesia to buy $1.14 billion worth of Russia... | JAKARTA (Reuters) - Indonesia will buy 11 Sukh... | worldnews | August 22, 2017 |

21417 rows × 4 columns

## 2. Fake news dataset

```
# #reading the fake news dataset
df_fake = pd.read_csv(r"C:\Users\Swati\OneDrive\Desktop\DataScience\Projects\Fake News Classifier\Fake.csv")
df_fake
```

| | title | text | subject | date |
|---|---|---|---|---|
| 0 | Donald Trump Sends Out Embarrassing New Year'... | Donald Trump just couldn t wish all Americans ... | News | December 31, 2017 |
| 1 | Drunk Bragging Trump Staffer Started Russian ... | House Intelligence Committee Chairman Devin Nu... | News | December 31, 2017 |
| 2 | Sheriff David Clarke Becomes An Internet Joke... | On Friday, it was revealed that former Milwauk... | News | December 30, 2017 |
| 3 | Trump Is So Obsessed He Even Has Obama's Name... | On Christmas day, Donald Trump announced that ... | News | December 29, 2017 |
| 4 | Pope Francis Just Called Out Donald Trump Dur... | Pope Francis used his annual Christmas Day mes... | News | December 25, 2017 |
| ... | ... | ... | ... | ... |
| 23476 | McPain: John McCain Furious That Iran Treated ... | 21st Century Wire says As 21WIRE reported earl... | Middle-east | January 16, 2016 |
| 23477 | JUSTICE? Yahoo Settles E-mail Privacy Class-ac... | 21st Century Wire says It s a familiar theme. ... | Middle-east | January 16, 2016 |
| 23478 | Sunnistan: US and Allied 'Safe Zone' Plan to T... | Patrick Henningsen 21st Century WireRemember ... | Middle-east | January 15, 2016 |
| 23479 | How to Blow $700 Million: Al Jazeera America F... | 21st Century Wire says Al Jazeera America will... | Middle-east | January 14, 2016 |
| 23480 | 10 U.S. Navy Sailors Held by Iranian Military ... | 21st Century Wire says As 21WIRE predicted in ... | Middle-east | January 12, 2016 |

23481 rows × 4 columns

Later checked for the shape of both datasets and also checked for the missing values.

```
# # print shape of fake dataset with rows and columns and information
df_real.shape
```

```
(21417, 4)
```

```
# # print shape of fake dataset with rows and columns and information
df_fake.shape
```

```
(23481, 4)
```

As a result, we got that the real news dataset has 21417 rows and 4 columns and fake news dataset has 23481 rows and 4 columns.

The columns are same for both the datasets which are as below:

- title- contains news headlines
- text- contains news content/article
- subject- the type of news
- date- the date the news was published

Since the datasets contains the real news and fake news lets just add one more feature named as 'label' to both datasets and give the label '0' for real news and label '1' for fake news.

```
# add a label column with value 0 for real news data
df_real['label']=0
df_real
```

```
# add a label column with value 1 for real news data
df_fake['label']=1
df_fake
```

## Concatenating title and text of news

News has to be classified based on the title and text jointly. Treating the title and content of news separately doesn't reap us any benefit. So, lets concatenate both the columns in both datasets and remove the original features. Also rearrange the columns.

```
# merge the features 'title' and 'text' to create new feature 'news' for fake news data
df_fake['news']=df_fake['title']+df_fake['text']

#drop the original features 'title' and 'text' from df_fake
df_fake=df_fake.drop(['title', 'text'], axis=1)
```

```
#Rearranging the columns
df_real = df_real[['subject', 'date', 'news','label']]
df_fake = df_fake[['subject', 'date', 'news','label']]
```

Now the next step is to convert date column into datetime feature. Since, there are some links and news headlines in date column of fake news data, we have to remove these before converting the date column into datetime feature.

```
#Removing links and the headline from the date column
df_fake=df_fake[~df_fake.date.str.contains("http")]
df_fake=df_fake[~df_fake.date.str.contains("HOST")]
```

```
#Converting the date to datetime format for fake news data
df_fake['date'] = pd.to_datetime(df_fake['date'])
```

```
#Converting the date to datetime format for real news data
df_real['date'] = pd.to_datetime(df_real['date'])
```

## Combining two datasets

Since we are given two different datasets for real and fake news, we have to combine these datasets for further processing.

```
# now lets combine these two datasets to get a single dataframe using pandas concat function to build a model
frames = [df_real, df_fake]
df_news = pd.concat(frames)
df_news
```

# 2.3 DATA PRE-PROCESSING

## Text Processing

This is an important phase for any text analysis application. There will be many unusefull content in the news which can be an obstacle when feeding to a machine learning model. Unless we remove them, the machine learning model doesn't work efficiently.

## Text Cleaning

let's define a function which will convert all text into a lowercase and removes links, punctuations, text containing numbers and text in square brackets.

```
#Creating a copy
clean_news=df_news.copy()
def review_cleaning(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
clean_news['news']=clean_news['news'].apply(lambda x:review_cleaning(x))
clean_news.head()
```

## Stopwords

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We would not want these words to take up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words. NLTK (Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.

In our project, we are considering the English stop words and removing those words.

```
stop = stopwords.words('english')
clean_news['news'] = clean_news['news'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
clean_news.head()
```

## Feature Extraction

Let’s extract more features from the news feature such as

1. Polarity: The measure which signifies the sentiment of the news
2. Review length: Length of the news (number of letters and spaces)
3. Word Count: Number of words in the news

```
import textblob
from textblob import TextBlob
```

```
#Extracting the features from the news
clean_news['polarity'] = clean_news['news'].map(lambda text: TextBlob(text).sentiment.polarity)
clean_news['review_len'] = clean_news['news'].astype(str).apply(len)
clean_news['word_count'] = clean_news['news'].apply(lambda x: len(str(x).split()))
```

# Stimming and Vectorization

## Stemming

Stemming is a method of deriving root word from the inflected word. Here we extract the reviews and convert the words in reviews to its root word. for example,

writing->write

reading->read

There is another technique known as Lemmatization where it converts the words into root words which has a semantic meaning. Since it takes a lot of time, I'm using stemming.

```
#Extracting 'reviews' for processing
news_features=clean_news.copy()
news_features=news_features[['news']].reset_index(drop=True)
news_features.head()
```

```
stop_words = set(stopwords.words("english"))
#Performing stemming on the review dataframe
ps = PorterStemmer()

#splitting and adding the stemmed words except stopwords
corpus = []
for i in range(0, len(news_features)):
    news = re.sub('[^a-zA-Z]', ' ', news_features['news'][i])
    news= news.lower()
    news = news.split()
    news = [ps.stem(word) for word in news if not word in stop_words]
    news = ' '.join(news)
    corpus.append(news)#Getting the target variable
y=clean_news['label']
```

As computer cannot understand words and their sentiment, we need to convert these words into 1's and 0's. To encode it we use TFIDF.

## TFIDF (Term Frequency — Inverse Document Frequency

This is a technique to quantify a word in documents, we generally compute a weight to each word which signifies the importance of the word in the document and corpus. This method is a widely used technique in Information Retrieval and Text Mining.

Here we are splitting as bigram (two words) and consider their combined weight. Also, we are taking only the top 5000 words from the news.

```
tfidf_vectorizer = TfidfVectorizer(max_features=5000,ngram_range=(2,2))
# TF-IDF feature matrix
X= tfidf_vectorizer.fit_transform(news_features['news'])
X.shape
```

```
(44888, 5000)
```

Now let's set the feature 'label' as target variable y.

```
#Getting the target variable
y=clean_news['label']
```

## Checking for Balance of Data

Since the problem is of classification type, we have to check whether our target variable is balanced or not. We should be careful about when handling imbalance data. If it is imbalanced, the model will be biased towards the higher frequency class and returns max output.

```
print(f'Original dataset shape : {Counter(y)}')
```

```
Original dataset shape : Counter({1: 23471, 0: 21417})
```

The result shows that Our dataset is nearly a balanced one. So, let's leave balancing it.

# CHAPTER 3

# MODEL BUILDING

## TRAIN-TEST SPLIT

Using train test split function, we are splitting the dataset into 70:30 ratio for train and test set respectively.

```
# Divide the dataset into Train and Test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)
```

## Model Building

As we have successfully processed the text data, not it is just a normal machine learning problem. Where from the sparse matrix we predict the classes in target feature.

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()


    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    thresh = cm.max() / 2.
    for i in range (cm.shape[0]):
        for j in range (cm.shape[1]):
            plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

## Model Selection

First select the best performing model by using cross validation. Let's consider all the classification algorithm and perform the model selection process

```
#creating the objects
logreg_cv = LogisticRegression(random_state=0)
dt_cv=DecisionTreeClassifier()
knn_cv=KNeighborsClassifier()
nb_cv=MultinomialNB(alpha=0.1)
cv_dict = {0: 'Logistic Regression', 1: 'Decision Tree',2:'KNN',3:'Naive Bayes'}
cv_models=[logreg_cv,dt_cv,knn_cv,nb_cv]

#Printing the accuracy
for i,model in enumerate(cv_models):
    print("{} Test Accuracy: {}".format(cv_dict[i],cross_val_score(model, X, y, cv=10, scoring ='accuracy').mean()))
```

```
Logistic Regression Test Accuracy: 0.9659594665748598
Decision Tree Test Accuracy: 0.935260370070789
KNN Test Accuracy: 0.6134179152128256
Naive Bayes Test Accuracy: 0.9373105638699313
```

From the results, we can see logistic regression outdone the rest of the algorithms followed by Naive Bayes and Decision Tree. That's great. So, let's go with logistic regression with hyperparameter tuning.

## Hyperparameter Tuning

We use regularization parameter and penalty for parameter tuning. let's see which one to plug.

```
param_grid = {'C': np.logspace(-4, 4, 50),
              'penalty':['l1', 'l2']}
clf = GridSearchCV(LogisticRegression(random_state=0), param_grid,cv=5, verbose=0,n_jobs=-1)
best_model = clf.fit(X_train,y_train)
print(best_model.best_estimator_)
print("The mean accuracy of the model is:",best_model.score(X_test,y_test))
```

```
LogisticRegression(C=24.420530945486497, random_state=0)
The mean accuracy of the model is: 0.9809163139526249
```

```
logreg = LogisticRegression(C=24.420530945486497, random_state=0)
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
```

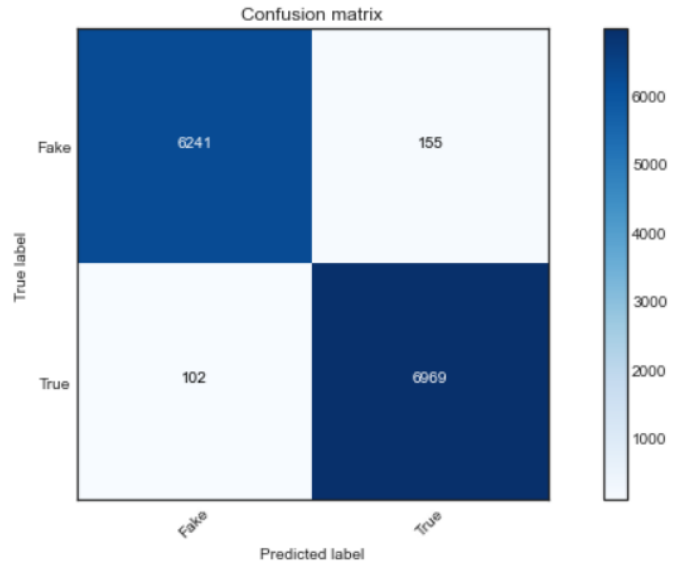
```
Accuracy of logistic regression classifier on test set: 0.98
```

We have got 98% accuracy. As already discussed before this is a biased dataset and we can easily get such higher accuracy without any effort in processing it. But for classification problems we need to get confusion matrix and check f1 score rather than accuracy.

## Confusion Matrix

```
cm = metrics.confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm, classes=['Fake','True'])
```

```
Confusion matrix, without normalization
```




Check out the diagonal elements (6241+6969), they are correctly predicted records and rest are incorrectly classified by the algorithm. Our model has done well (results are good by the data is biased :P)

## Classification Report

Considering Fake news, we should seriously consider precision score (False positive). We can't afford the mistakes when the model classifies fake news as true which will lead to chaos

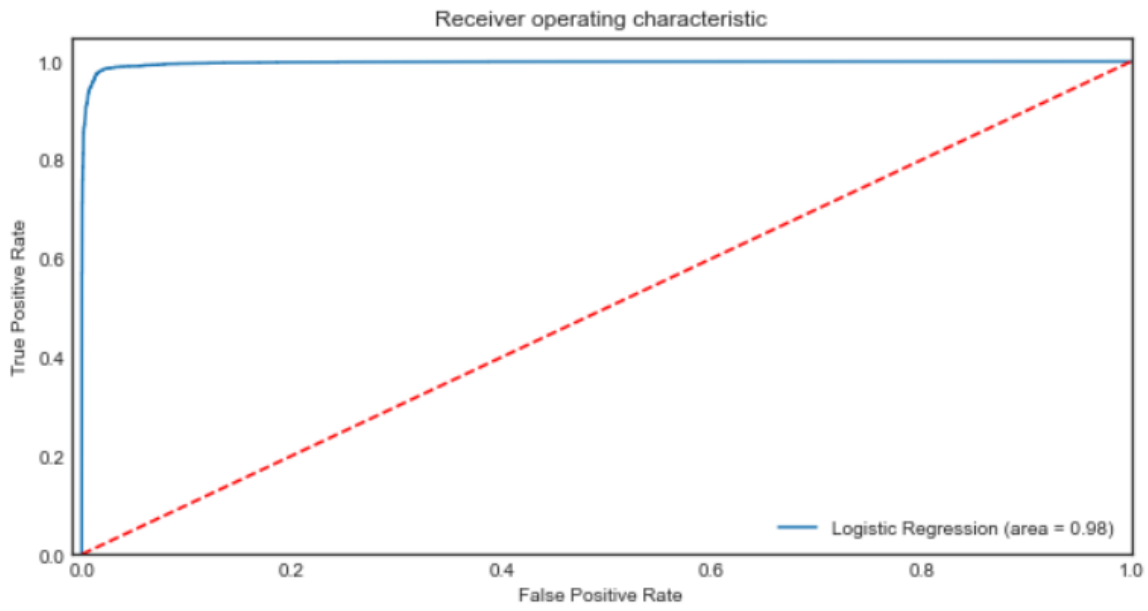
```
print("Classification Report:\n",classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      6396
           1       0.98      0.99      0.98      7071

    accuracy                           0.98     13467
   macro avg       0.98      0.98      0.98     13467
weighted avg       0.98      0.98      0.98     13467
```

All our scores are 98%. Certainly, unreal to get such values. There are only changes in the support.

## ROC-AUC Curve

```
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([-0.01, 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()
```




We should consider the AUC score here which is 98%. Very well. All metrics are performing good. The more far left the curve is better our model We can adjust our threshold based on our ROC curve to get results based on model requirements.

## Deep learning-LSTM

in this part we use neural network to predict whether the given news is fake or not.

LSTM (long short term memory) helps in containing sequence information. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction

problems. This is a behaviour required in complex problem domains like machine translation, speech recognition, and more.

## One hot for Embedding layers

While one hot encoding the words in sentences will take the index from the vocabulary size. Let's fix the vocabulary size to 10000

```
#Setting up vocabulary size
voc_size=10000

#One hot encoding
onehot_repr=[one_hot(words,voc_size)for words in corpus]
```

We can see all the words in the sentences are transformed into their index from the vocabulary we created.

All the neural networks require to have inputs that have the same shape and size. However, when we pre-process and use the texts as inputs for our LSTM model, not all the sentences have the same length. In other words, naturally, some of the sentences are longer or shorter. We need to have the inputs with the same size, this is where the padding is necessary. Here we take the common length as 5000 and perform padding using pad sequence () function. Also, we are going to 'pre' pad so that zeros are added before the sentences to make the sentence of equal length.

```
#Setting sentence length
sent_length=5000

#Padding the sentences
embedded_docs=pad_sequences(onehot_repr,padding='pre',maxlen=sent_length)
print(embedded_docs)
```

```
[[   0    0    0 ... 3401 5764 3758]
 [   0    0    0 ... 5185 5415 1335]
 [   0    0    0 ... 8430 4296 7967]
 ...
 [   0    0    0 ... 3372 2931 7032]
 [   0    0    0 ... 3372 6942 8032]
 [   0    0    0 ... 3372 1050 7032]]
```

```
embedded_docs[1]
```

```
array([   0,    0,    0, ..., 5185, 5415, 1335])
```

## LSTM Model

At first, we are going to develop the base model and compile it. The first layer will be the embedding layer which has the input of vocabulary size, vector features and sentence length. Later we add 30% dropout layer to prevent overfitting and the LSTM layer which has 100 neurons in the layer.In final layer we use sigmoid activation function. Later we compile the model using adam optimizer and binary cross entropy as loss function since we have only two outputs.

```
#Creating the lstm model
embedding_vector_features=40
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(Dropout(0.3))
model.add(LSTM(100)) #Adding 100 lstm neurons in the layer
model.add(Dropout(0.3))
model.add(Dense(1,activation='sigmoid'))

#Compiling the model
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
print(model.summary())
```

## Fitting the LSTM Model

Before fitting to the model, let's consider the padded embedded object as X and y as y itself and convert them into an array.

```
# Converting the X and y as array
X_final=np.array(embedded_docs)
y_final=np.array(y)

#Check shape of X and y final
X_final.shape,y_final.shape
```

```
((44888, 5000), (44888,))
```

Let's split our new X and y variable into train and test and proceed with fitting the model to the data. We have considered 10 epochs and 64 as batch size. It can be varied to get better results.

```
# Train test split of the X and y final
X_train, X_test, y_train, y_test = train_test_split(X_final, y_final, test_size=0.33, random_state=42)

# Fitting with 10 epochs and 64 batch size
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=10,batch_size=64)
```

## Evaluation of model

```
# Predicting from test data
y_pred=model.predict_classes(X_test)

#Creating confusion matrix
#confusion_matrix(y_test,y_pred)
cm = metrics.confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm,classes=['Fake','True'])
```

```
#Checking for accuracy
accuracy_score(y_test,y_pred)
```

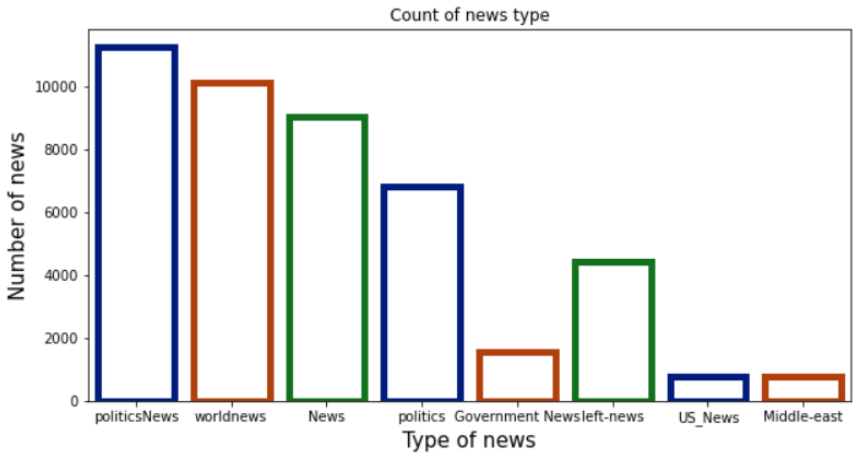
From the classification report we can see the accuracy value is nearly around 48%.

# CHAPTER 4

# EDA

## 1. Countplot of news subject

```
# Plotting the frequency plot for 'subject' feature
ax = sns.countplot(x="subject", data=clean_news,
                   facecolor=(0, 0, 0, 0),
                   linewidth=5,
                   edgecolor=sns.color_palette("dark", 3))

#Setting labels and font size
ax.set(xlabel='Type of news', ylabel='Number of news',title='Count of news type')
ax.xaxis.get_label().set_fontsize(15)
ax.yaxis.get_label().set_fontsize(15)
```




## Observations

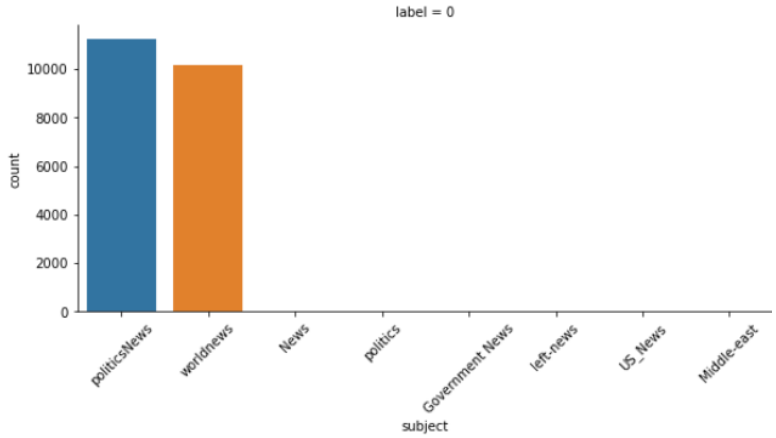
1. Our dataset has more political news than any other news followed by world news

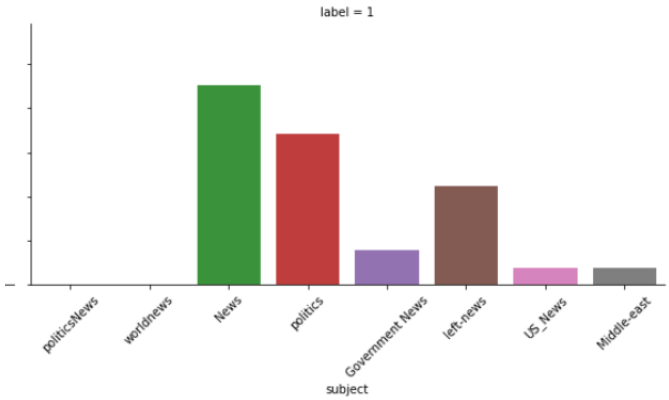
2. We have some repeated class names which expresses same meaning such as news, politics, government news etc which is similar to the alternative

# 2. Count of news subject based on real or fake

```
# labelwise countplot for subject
ax = sns.catplot(x="subject", col="label",
                 data=clean_news, kind="count",
                 height=4, aspect=2)

#Rotating the xlabels
ax.set_xticklabels(rotation=45)
```

```
<seaborn.axisgrid.FacetGrid at 0x2045d5152b0>
```






## Observations

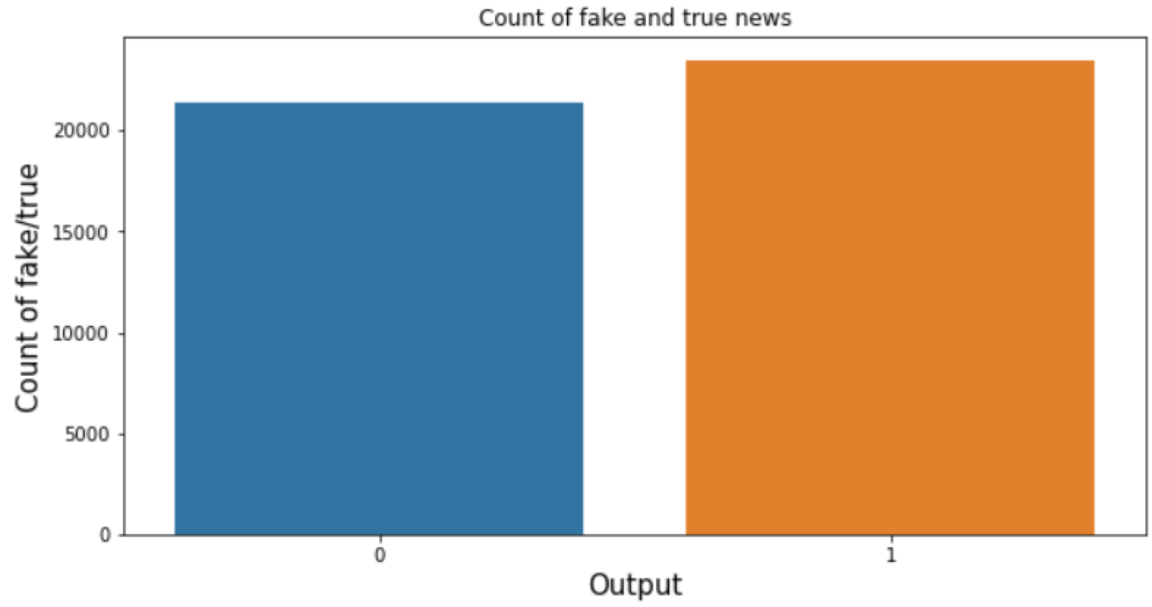
1. Fake news are all over the category except politics and world news.

2. True news are present only in politics and world news and the count is high.

3. THIS IS A HIGHLY BIASED DATASET and we can expect higher accuracy which doesn't signify it is a good model considering the poor quality of dataset.

# 3. Count of fake news and true news

```
# countplot for fake and real news
ax=sns.countplot(x="label", data=clean_news)

#Setting labels and font size
ax.set(xlabel='Output', ylabel='Count of fake/true',title='Count of fake and true news')
ax.xaxis.get_label().set_fontsize(15)
ax.yaxis.get_label().set_fontsize(15)
```

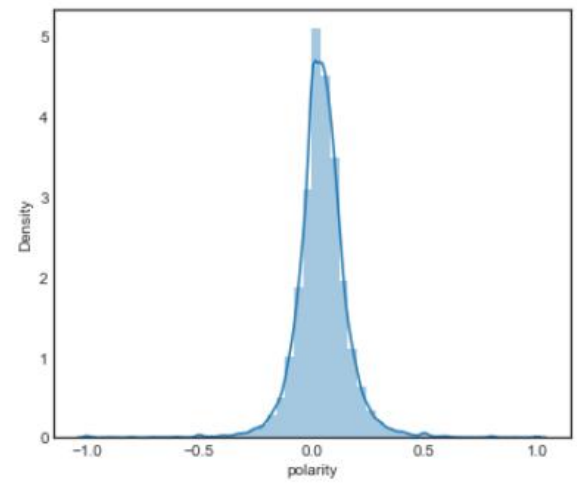


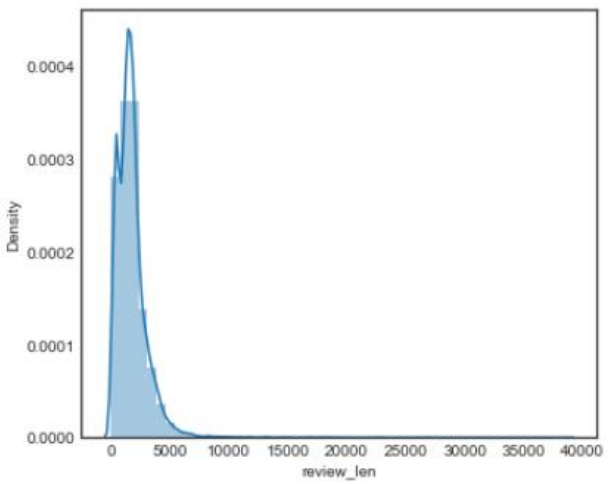
## Observations:

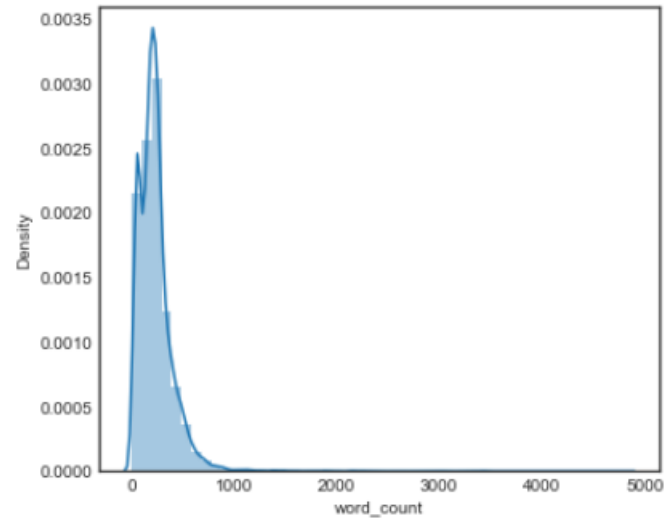
The fake news count is quite higher than real news count means our data is not much imbalanced.

## 4. Plotting the distribution of the extracted features

```
#Plotting the distribution of the extracted feature
plt.figure(figsize = (20, 5))
plt.style.use('seaborn-white')
plt.subplot(131)
sns.distplot(clean_news['polarity'])
fig = plt.gcf()
plt.subplot(132)
sns.distplot(clean_news['review_len'])
fig = plt.gcf()
plt.subplot(133)
sns.distplot(clean_news['word_count'])
fig = plt.gcf()
```



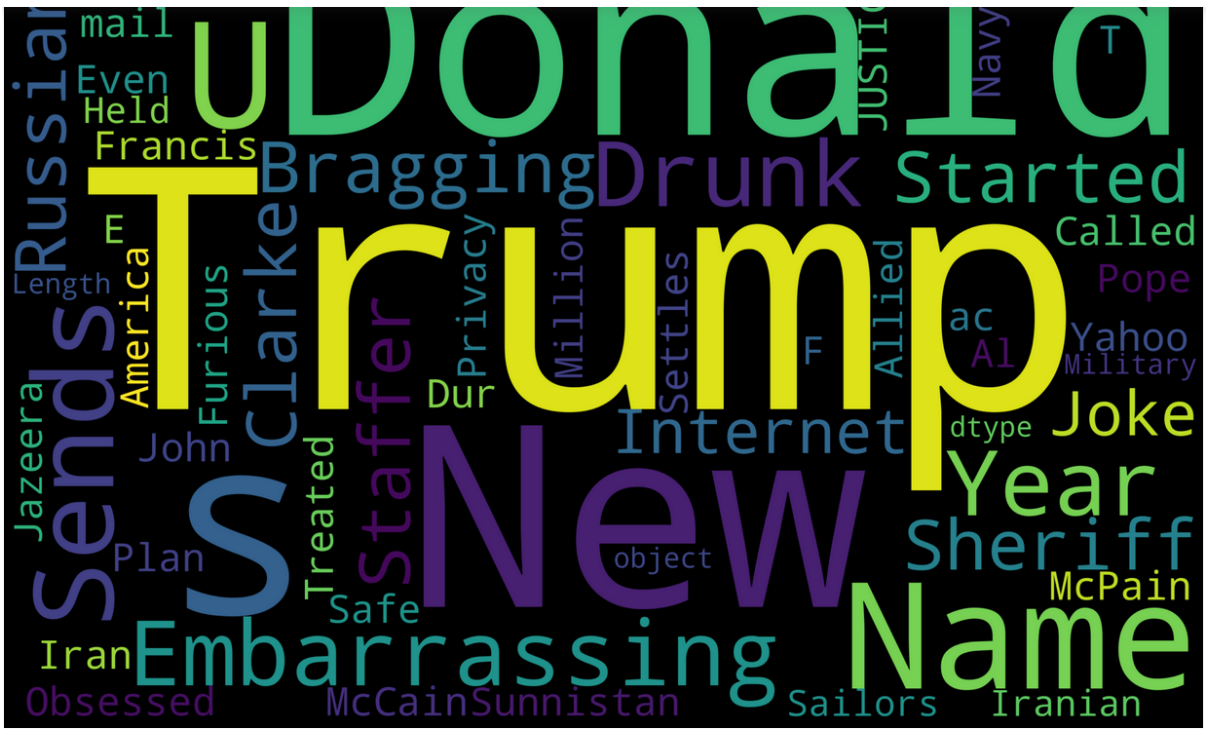
## Observations

1. Most of the polarity are neutral, neither it shows some bad news nor much happy news

2. The word count is between 0-1000 and the length of the news are between 0-5000 and few near 10000 words which could be an article.

# 5. Wordcloud of Fake and True News

```
# lets plot wordcloud for fake news data

text = df_fake["news"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (25, 18),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



## Observations

1.Most of the fake news revolves around Donald Trump and America

2.There are also fake news about privacy, internet etc.

# 6.wordcloud for real news data

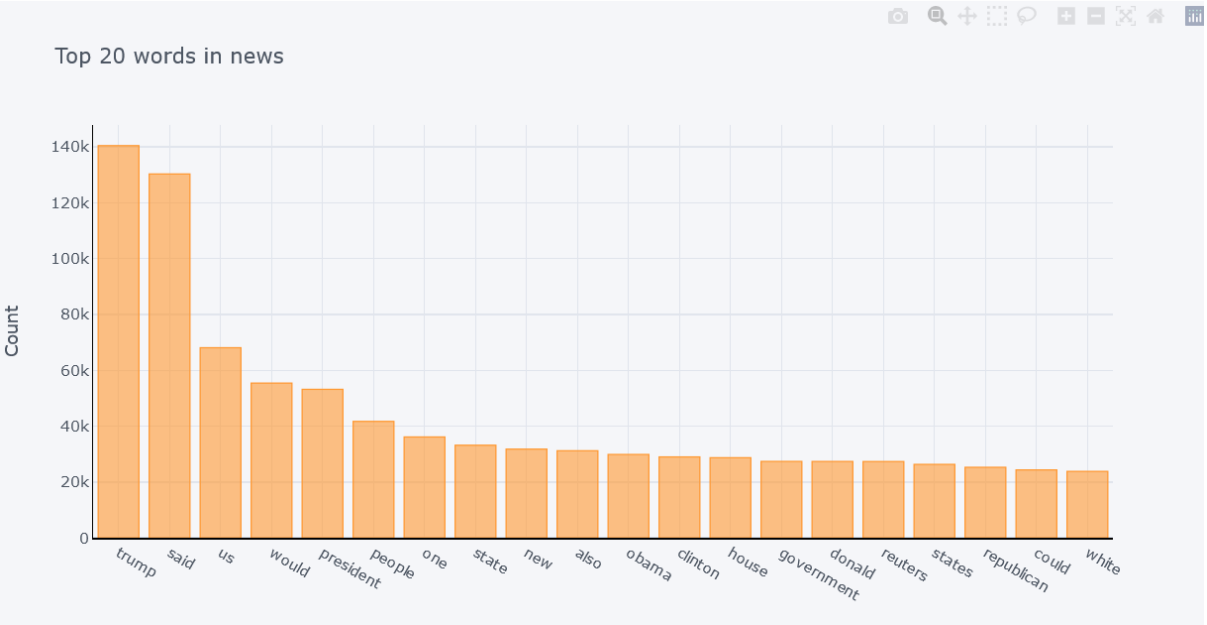
```
text = df_real["news"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (25, 18),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```


## Observations

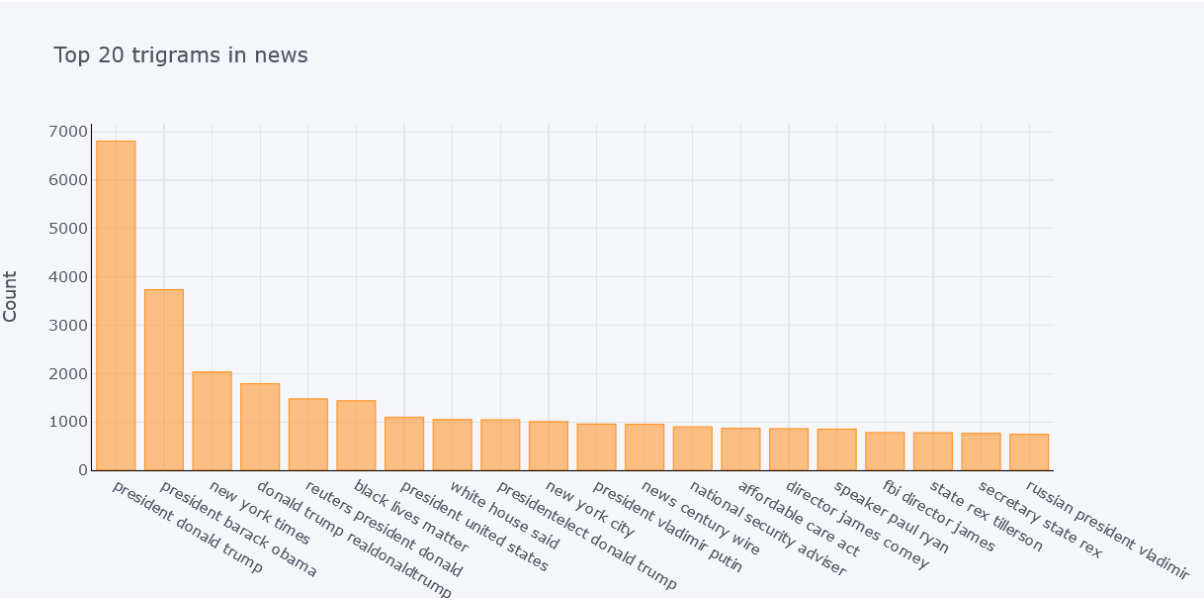
1. Real news doesn't involve much trump instead on Republican Party and Russia

2. There are news about Budget, military which comes under government news

## 7. Top 20 words in News




## 8. top trigram words



# CHAPTER 5

# CONCLUSION

## 5.1 RESULT

The accuracy scores of various models used are given in table below

| Sr. No. | Algorithm | Accuracy Score |
|---|---|---|
| **1** | Logistic Regression | 0.965959 |
| **2** | Decision Tree | 0.934993 |
| **3** | KNN | 0.613417 |
| **4** | Naïve-Bayes | 0.937310 |
| **5** | LSTM | 0.48 |

From the above table we can see that the logistic regression model gives us better accuracy score 0.965959.

## 5.2 CONCLUSION

The task of classifying news manually requires in-depth knowledge of the domain and expertise to identify anomalies in the text. In this research, we discussed the problem of classifying fake news articles using machine learning models and NLP. Fake news detection has many open issues that require attention of researchers. For instance, in order to reduce the spread of fake news, identifying key elements involved in the spread of news is an important step. Graph theory and machine learning techniques can be employed to identify the key sources involved in spread of fake news. Likewise, real time fake news identification in videos can be another possible future direction.

In the 21st century, the majority of the tasks are done online. Newspapers that were earlier preferred as hard-copies are now being substituted by applications like Facebook, Twitter, and news articles to be read online. WhatsApp's forwards are also a major source. The growing problem of fake news only makes things more complicated and tries to change or hamper the opinion and attitude of people towards use of digital technology. When a person is deceived by the real news two possible things happen- People start believing that their perceptions about a particular topic are true as assumed.

Thus, in order to curb the phenomenon, we have developed our Fake news Detection system that takes input from the user and classify it to be true or fake. To implement this, various NLP and Machine Learning Techniques have to be used. The model is trained using an appropriate dataset and performance evaluation is also done using various performance measures. The best model, i.e., the model with highest accuracy is used to classify the news headlines or articles. As evident above for static search, our best model came out to be Logistic Regression with an accuracy of 65%. Hence, we then used grid search parameter optimization to increase the performance of logistic regression which then gave us the accuracy of 75%. Hence, we can say that if a user feed a particular news article or its headline in our model, there are 75% chances that it will be classified to its true nature. The user can check the news article or keywords online; he can also check the authenticity of the website. The accuracy for dynamic system is 93% and it increases with every iteration.