# FAKE NEWS CLASSIFIER USING ML AND NLP

BY:

MRS. SWATI AMIT MOTUGADE

# INTRODUCTION

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.

# PROBLEM STATEMENT

- 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.

# 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

# LITERATURE REVIEW

- 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.

# DATASET INFORMATION

- 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. We have to insert one label column zero for fake news and one for true news. We are combined both datasets using pandas built-in function.
- Dataset has 4 features like title, text, subject and date.

# DATASET LOADING

Before loading the dataset we need to import all the necessary libraries like 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')
```

# DATASET LOADING

1. Real news dataset

```
# #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
```

1. 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
```

# DATA PRE-PROCESSING

Since, there is no label column indicating the news real or fake but we are given two separate datasets for real and fake news we added one new column of label in both datasets i.e., label '0' for real news dataset and label '1' for fake news dataset.

```
# 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
```

# DATA PRE-PROCESSING

## 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']]
```

# DATA PRE-PROCESSING

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'])
```

# DATA PRE-PROCESSING

## 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
```

- 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.

# 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.

```
#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()
```

# DATA PRE-PROCESSING

## 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()
```

# DATA PRE-PROCESSING

- 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()))
```

# DATA PRE-PROCESSING

- 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

```
#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']
```

# DATA PRE-PROCESSING

- 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.

# DATA PRE-PROCESSING

- 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.

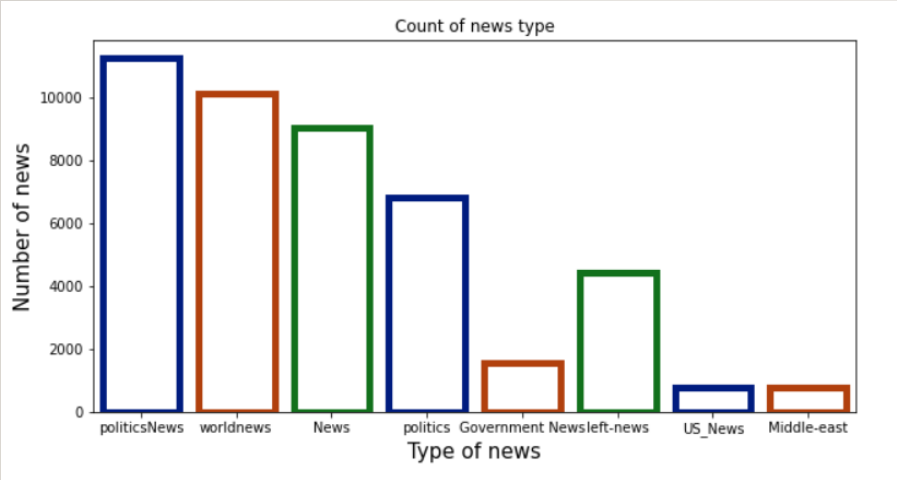
```
print(f'Original dataset shape : {Counter(y)}')
```

```
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.

# EDA

## 1. Countplot of news subject



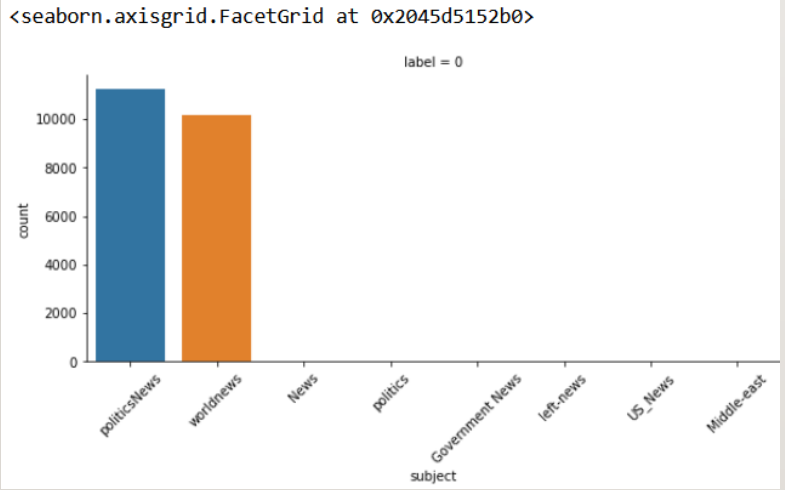

- **Observations**

- 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

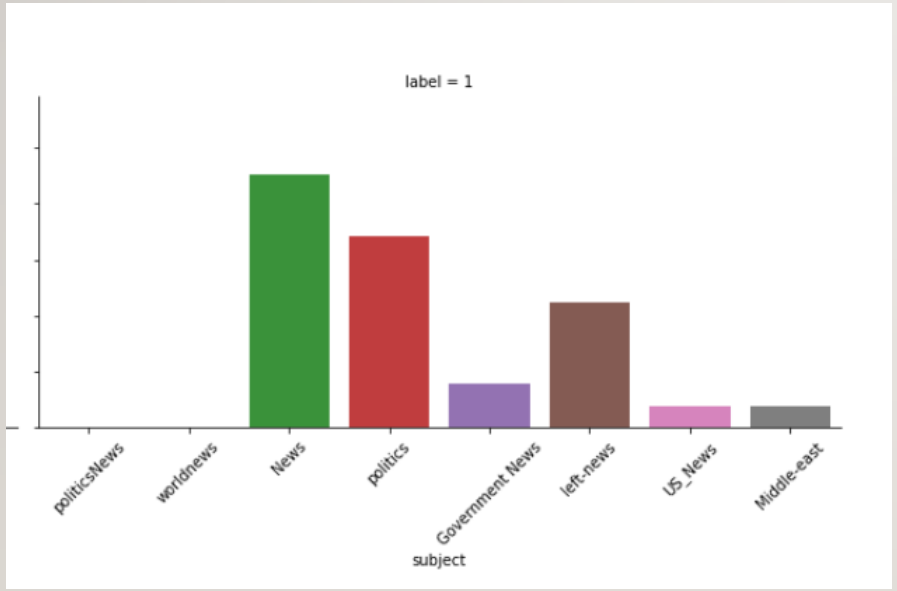
# EDA

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




- **Observations**
- Real news are present only in politics and world news and the count is high.

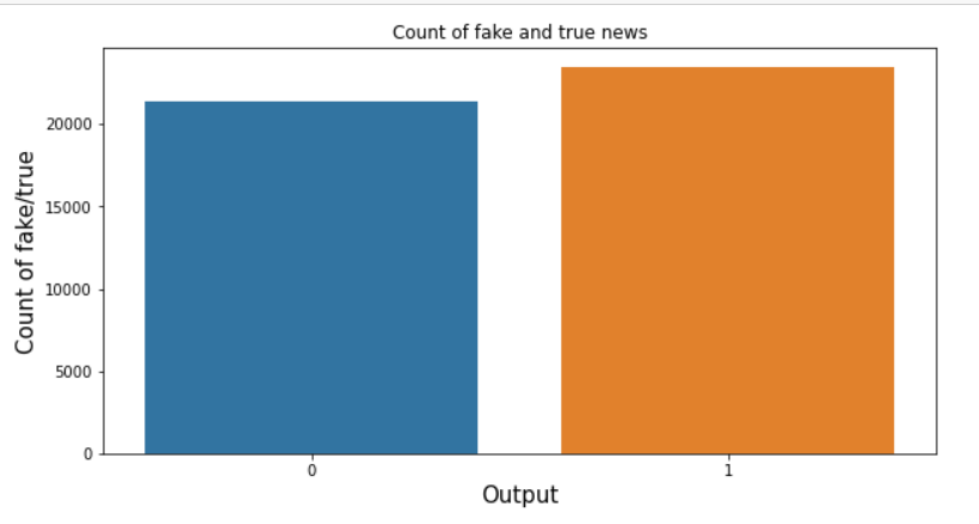
# EDA




- **Observations**
- Fake news are all over the category except politics and world news.

# EDA

## 3. Count of fake news and true news




- **Observations:**

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

# MODEL BUILDING

- The algorithms used for model building are given in the table below with their respective accuracy scores

| 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 |

The above table shows that logistic regression model gives us better accuracy score.

# 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.

# THANK YOU