FAKE NEWS PROJECT

Submitted by:

SARANYA M

ACKNOWLEDGMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to Flip Robo Technologies, Bangalore for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my special gratitude and thanks to our institute DataTrained & others seen unseen hands which have given us direct & indirect help in completion of this project. With help of their brilliant guidance and encouragement, I was able to complete my tasks properly and were up to the mark in all the tasks assigned. During the process, I got a chance to see the stronger side of my technical and non-technical aspects and also strengthen my concepts.

INTRODUCTION

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.

Conceptual Background of the Domain Problem

Fake news is defined as a made-up story with an intention to deceive or to mislead. The rate of production of fake news has increased exponentially. In the past news obtained from newspaper, radio or TV were considered as the best and authentic source of information about the real world and ongoing situations but now everything has changed. In the run of popularity and ill mind set the media houses and social media are spreading fake news. It’s becoming harder and harder to say whether a piece of news is real or fabricated.

The effect of fake news can be seen everywhere. The fake news leads to communal disturbance, character assassination, mental trauma, sometimes it is used as a weapon to achieve some illicit plans etc. these are like wild fire which spread too quickly and difficult to control. Which creates difficulty in differentiating between fake news and authentic news.

Review of Literature

Technologies such as Artificial Intelligence (AI) and Natural Language Processing (NLP) tools offer great promise for researchers to build systems which could automatically detect fake news. However, detecting fake news is a challenging task to accomplish as it requires models to summarize the news and compare it to the actual news in order to classify it as fake. Moreover, the task of comparing proposed news with the original news itself is a daunting task as it's highly subjective and opinionated.

Motivation for the Problem Undertaken

The goal is to build a prototype to classify the news as fake or not fakes in order to bring awareness and reduce unwanted chaos.

➔ To apply data pre-processing and preparation techniques in order to obtain clean data.
➔ Explore the effectiveness of multiple machine learning approaches and select the best for this problem.

ANALYTICAL PROBLEM FRAMING

Model Building Phase

You need to build a machine learning model. Before model building do all data pre-processing steps involving NLP. Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science. Include all the steps like-

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best model

Data Sources and their formats

The dataset is spread across 20800 rows, each depicting the news and its related details. The data samples contain 6 fields which include 'Id', 'headline', 'written_by', 'news', and 'label'. The description of each of the column is given below:

1. **Id:** Unique id of each news article
2. **Headline:** It is the title of the news.
3. **News:** It contains the full text of the news article
4. **Unnamed:0:** It is a serial number
5. **Written_by:** It represents the author of the news article
6. **Label:** It tells whether the news is fake (1) or not fake (0).

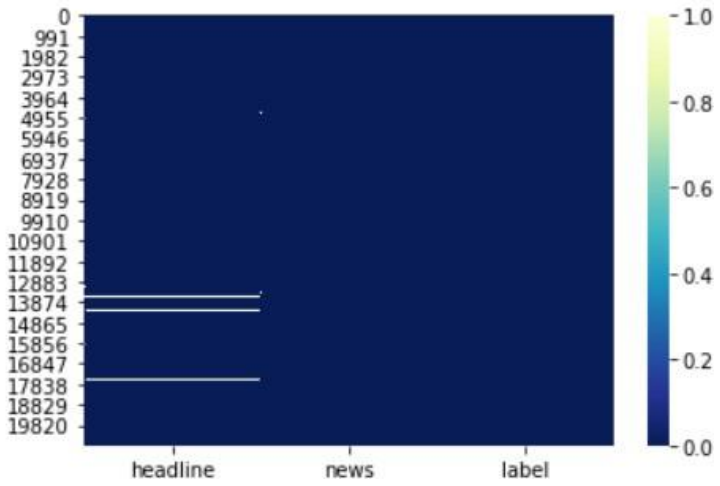
The sample data for the reference is as shown below:

Checking the missing values

Moving ahead, cleaning the dataset will remove errors which in turn will increase productivity and render highest quality information in decision making. Here, in this dataset, the null values are checked using isnull().sum(), which results in getting 2 features having null values in actual. The same has been visualized using heatmap.

```
headline    558
news         39
label         0
dtype: int64

<AxesSubplot:>
```



Handling the missing values

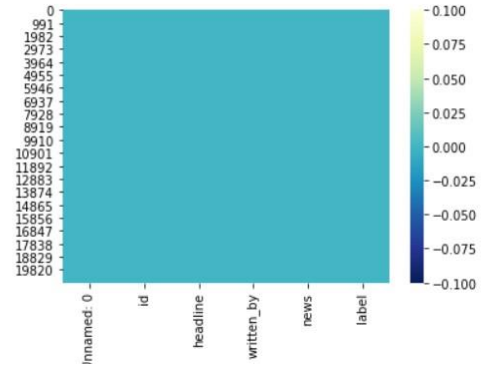
```
df['headline'].fillna('No headline', inplace=True)
df['written_by'].fillna('Not available', inplace=True)
df['news'].fillna("Not available", inplace=True)
```

```
#Checking the null values after handling them
df.isnull().sum()
```

```
Unnamed: 0    0
id            0
headline      0
written_by    0
news          0
label         0
dtype: int64
```

```
#Plotting heatmap for missing data
sns.heatmap(df.isnull(), cmap='YlGnBu_r')
```

```
<AxesSubplot:>
```



Analysing the Statistics

Statistical analysis summarizes and provides information about the sample data. It is calculated by reporting the mean along with a measure of variability (standard deviation(s) or standard error of the mean).

```
#Checking the statistical summary of the dataset
df.describe()
```

	Unnamed: 0	id	label
count	20800.000000	20800.000000	20800.000000
mean	10399.500000	10399.500000	0.500625
std	6004.587135	6004.587135	0.500012
min	0.000000	0.000000	0.000000
25%	5199.750000	5199.750000	0.000000
50%	10399.500000	10399.500000	1.000000
75%	15599.250000	15599.250000	1.000000
max	20799.000000	20799.000000	1.000000

-> Unnamed and id are the columns which are having the least importance data among all.

-> Label column has equal minimum and maximum value

-> Mean and standard deviation is same for them and the count is also equal.

Dropping unnecessary data

```
df.drop(['Unnamed: 0', 'id'], axis=1, inplace=True)
#Checking the dataset after dropping
df.head()
```

	headline	written_by	news	label
0	Ethics Questions Dogged Agriculture Nominee as...	Eric Lipton and Steve Eder	WASHINGTON — In Sonny Perdue's telling, Geo...	0
1	U.S. Must Dig Deep to Stop Argentina's Lionel ...	David Waldstein	HOUSTON — Venezuela had a plan. It was a ta...	0
2	Cotton to House: 'Do Not Walk the Plank and Vo...	Pam Key	Sunday on ABC's "This Week," while discussing ...	0
3	Paul LePage, Besieged Maine Governor, Sends Co...	Jess Bidgood	AUGUSTA, Me. — The beleaguered Republican g...	0
4	A Digital 9/11 If Trump Wins	Finian Cunningham	Finian Cunningham has written extensively on...	1

Pre-processing using NLP

Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted. This data is usually not necessary or helpful when it comes to analysing data because it may hinder the process or provide inaccurate results.

Before cleaning the data, a new column is created named 'length_before_cleaning' which shows the total length of the news respectively before cleaning the text.

The following steps were taken in order to clean the text:

- Transform the text into lower case.
- Replaced the email addresses with the text 'emailaddress'
- Replaced the URLs with the text 'webaddress'
- Removed the HTML tags
- Removed the numbers
- Removed extra newlines
- Removed the punctuations
- Removed the unwanted white spaces
- Removed the remaining tokens that are not alphabetic
- Removed the stop words

Tokenization

Word tokenization is the process of splitting a large sample of text into words. This is a requirement in natural language processing tasks where each word needs to be captured and subjected to further analysis.

After cleaning the text, each comment i.e., the corpus is split into words. Thus, the text is tokenized into words using word_tokenize().

Lemmatization

Lemmatization in NLTK refers to the morphological analysis of words, which aims to remove inflectional endings. It helps in returning the base or dictionary form of a word known as the lemma. The NLTK Lemmatization method is based on WorldNet's built-in morph function.Thus, the words are lemmatized using WordNetLemmatizer() after importing the necessary library to perform the same and then creating the instance for it.

All the text cleaning or the above steps are performed by defining a function and applying the same using apply() to the ‘News’ column of the dataset. Below is the code shown:

```
#Defining the stop words
stop_words = stopwords.words('english')

#Defining the lemmatizer
lemmatizer = WordNetLemmatizer()
```

```
#Function Definition
def clean_text(text):

    # convert to lower case
    lowered_text = text.lower()

    # Replacing email addresses with 'emailaddress'
    text = re.sub(r'^.+@[^\.].*\.[a-z]{2,}$', 'emailaddress', lowered_text)

    # Replace URLs with 'webaddress'
    text = re.sub(r'http\S+', 'webaddress', text)

    # Removing the HTML tags
    text = re.sub(r"<.*?>", " ", text)

    # Removing numbers
    text = re.sub(r'[0-9]', " ", text)

    # Removing extra newline
    text = text.strip("\n")

    # Removing Punctuations
    text = re.sub(r'[^\w\s]', ' ', text)
    text = re.sub(r'\_',' ',text)

    # Removing the unwanted white spaces
    text = " ".join(text.split())

    # Splitting data into words
    tokenized_text = word_tokenize(text)

    # Removing remaining tokens that are not alphabetic, Removing stop words and Lemmatizing the text
    removed_stop_text = [lemmatizer.lemmatize(word) for word in tokenized_text if word not in stop_words if word.isalpha()]

    return " ".join(removed_stop_text)
```

```
# Applying the above custom function to the required features
df['headline'] = df['headline'].apply(lambda x: clean_text(x))
df['written_by'] = df['written_by'].apply(lambda x: clean_text(x))
df['news'] = df['news'].apply(lambda x: clean_text(x))
```

```
#Checking the feature after cleaning
df['news']
```

```
0        washington sonny perdue telling georgian growi...
1        houston venezuela plan tactical approach desig...
2        sunday abc week discussing republican plan rep...
3        augusta beleaguered republican governor maine ...
4        finian cunningham written extensively internat...
                               ...
20795                   dog licking vomit chinese overlord
20796    rixon stewart november rixon stewart nov migra...
20797    posted eddie know dakota access pipeline prote...
20798    officially summer society boutique society mem...
20799    emory university atlanta georgia announced fun...
Name: news, Length: 20800, dtype: object
```

```
df['headline']
```

```
0        ethic question dogged agriculture nominee geor...
1        u must dig deep stop argentina lionel messi ne...
2        cotton house walk plank vote bill pas senate b...
3        paul lepage besieged maine governor sends conf...
4                                      digital trump win
                               ...
20795                                          headline
20796                  albert pike european migrant crisis
20797    dakota access caught infiltrating protest inci...
20798                stretch summer solstice new york time
20799    emory university pay percent undocumented stud...
Name: headline, Length: 20800, dtype: object
```

```
#Creating new features for checking the length after cleaning of these 3 columns
df['Headlinelen_after_cleaning'] = df['headline'].map(lambda x: len(x))
df['WrittenBylen_after_cleaning'] = df['written_by'].map(lambda x: len(x))
df['Newslen_after_cleaning'] = df['news'].map(lambda x: len(x))
```

We also created new features for comparing the original length before cleaning and the new length after cleaning.

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

For doing this project, the hardware used is a laptop with high end specification and a stable internet connection. While coming to software part, I had used anaconda navigator and in that I have used **Jupyter notebook** to do my python programming and analysis.

For using an CSV file, Microsoft excel is needed. In Jupyter notebook, I had used lots of python libraries to carry out this project and I have mentioned below with proper justification:

```
#Basic libraries
import pandas as pd
import numpy as np

# Stats library
from scipy.stats import randint

#Visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from wordcloud import WordCloud

#NLTK libraries
import nltk
import re
import string
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer

# Machine Learning libraries
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
import xgboost as xg
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

#Metrics libraries
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import log_loss

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

MODEL/S DEVELOPMENT AND EVALUATION

Separating the input and output variables and Vectorizing X feature

```
#Before separating features, we need to convert the object data into number vectors and it can be done by using TF-IDF vectorizer
tf_vec = TfidfVectorizer(max_features=15000)
features = tf_vec.fit_transform(df['written_by'] + df['headline'] + df['news'])
#We are adding these features as we need to convert all of them together
```

```
#Let's Separate the input and output variables represented by X and y respectively
X=features
y=df['label']
```

```
print(X.shape,'\t\t', y.shape)    #Checking the shape of the data
```

```
(20800, 15000)          (20800,)
```

```
X  #Checking out X data
```

```
<20800x15000 sparse matrix of type '<class 'numpy.float64'>'
        with 4918746 stored elements in Compressed Sparse Row format>
```

```
y #Checking out y data
```

```
0        0
1        0
2        0
3        0
4        1
        ..
20795    1
20796    1
20797    1
20798    0
20799    0
Name: label, Length: 20800, dtype: int64
```

Splitting the train and test data

Training and testing the models minimize the effects of data discrepancies and better understand the characteristics of the model. The **training** data is used to make the machine recognize patterns in the data and the **test** data is used to see how well the machine can predict new answers based on its training. ‘X’ and ‘y’ were split for training and testing using train_test_split in a ratio of 70:30 respectively.

Building the model

After running various algorithms, the final results are as follows:

```
#Printing the results in a dataFrame
results = pd.DataFrame({"Model" : Model,
                        'accuracy_score' : score,
                        'cross_validation_score' : cvs,
                        'log_loss': l_loss,
                        'AUC_ROC Score' : rocscore,
                        'Precision' : precision,
                        'Recall' : recall,
                        'f1_score' : f1score
                        })
results
```

	Model	accuracy_score	cross_validation_score	log_loss	AUC_ROC Score	Precision	Recall	f1_score
0	Logistic Regression	94.871795	94.956731	1.771241	94.871458	0.946497	0.951344	0.948914
1	MultinomialNB	90.208333	90.557692	3.381942	90.214296	0.943523	0.855634	0.897432
2	DecisionTreeClassifier	91.506410	92.221154	2.933614	91.507013	0.919198	0.910371	0.914764
3	KNeighborsClassifier	58.605769	59.504808	14.297389	58.553536	0.547766	0.992958	0.706043
4	RandomForestClassifier	95.000000	95.254808	1.726953	95.001800	0.963109	0.935980	0.949351
5	AdaBoostClassifier	94.198718	94.033654	2.003717	94.198174	0.938413	0.946223	0.942302
6	GradientBoostingClassifier	94.246795	94.403846	1.987113	94.245738	0.935433	0.950704	0.943007

After running the algorithms and according to the scores of performance metrics and other scores, we can see that Logistic Regression and RandomForest Classifier algorithms are performing well. Now, we will perform Hyperparameter Tuning to find out the best parameters and try to increase the scores.

Hyperparameter Tuning

After tuning the model, we got logistic regression as the best performing model and the code is given below:

Logistic Regression

```
#Parameters list to pass in GridSearchCV
parameters={'penalty':['l1','l2','elasticnet'],'C':[10, 1, 0.1, 0.01,0.001,0.001]}
```

```
#Using GridSearchCV to run the parameters and checking final accuracy
LR=LogisticRegression()
grid=GridSearchCV(LR,parameters,cv=5,scoring='accuracy')
grid.fit(x_train,y_train)
print(grid.best_params_)  #Printing the best parameters obtained
print(grid.best_score_) #Mean cross-validated score of best_estimator
```

```
{'C': 10, 'penalty': 'l2'}
0.9563186813186813
```

```
#Using the best parameters obtained
LR=LogisticRegression(C=10,penalty='l2')
LR.fit(x_train,y_train)
pred=LR.predict(x_test)
print('Accuracy score: ',accuracy_score(y_test,pre)*100)
print('Cross validation score: ',cross_val_score(LR,X,y,cv=5,scoring='accuracy').mean()*100)
false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
roc_auc= auc(false_positive_rate,true_positive_rate)
print('roc_auc_score: ',roc_auc)
loss = log_loss(y_test, pre)
print("Log loss:", loss)
print('Classification report: \n')
print(classification_report(y_test,pre))
print('Confusion matrix: \n')
print(confusion_matrix(y_test,pre))
```

Finalizing the model

```
lr_prediction=LR.predict(X)
print('Predictions of Logistic Regression: ',lr_prediction)
```

```
Predictions of Logistic Regression:  [0 0 0 ... 1 0 0]
```

```
#Saving the model
import joblib
joblib.dump(LR,'Fake_News_Classifier.pkl')
```

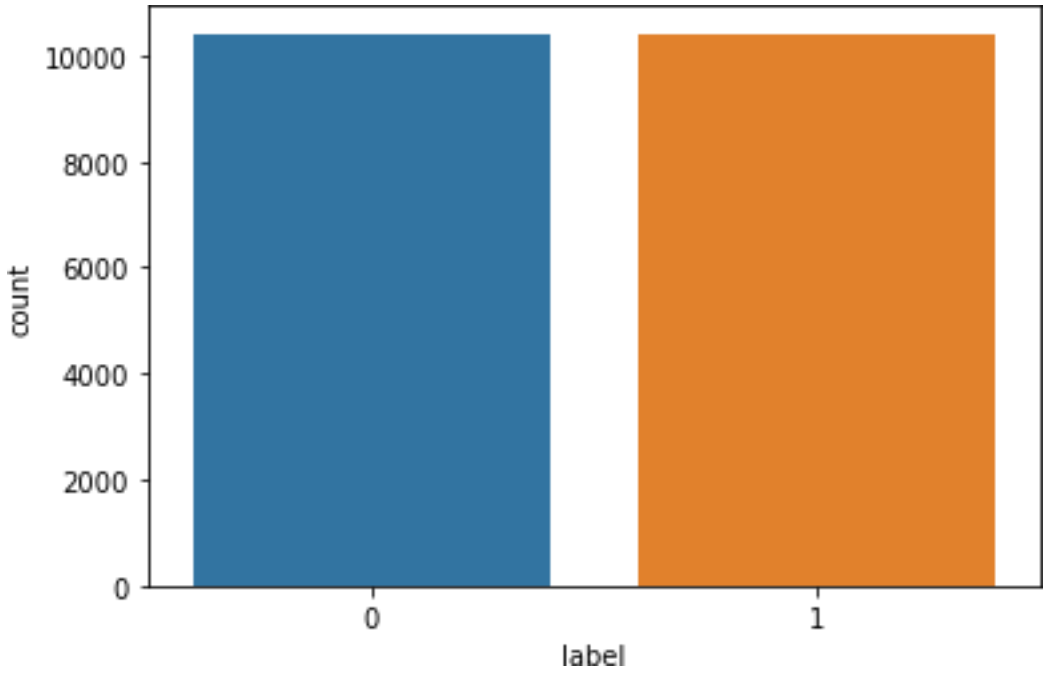
```
['Fake_News_Classifier.pkl']
```

```
pred_results=pd.DataFrame(lr_prediction)
pred_results.to_csv('FakeNews_Results.csv')
```

After predicting using test data, we will store the results in a csv file.

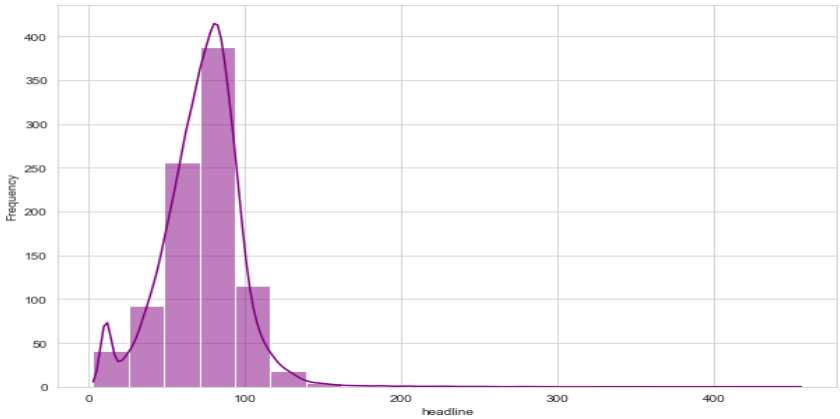
DATA VISUALIZATION

Plotting countplot for label



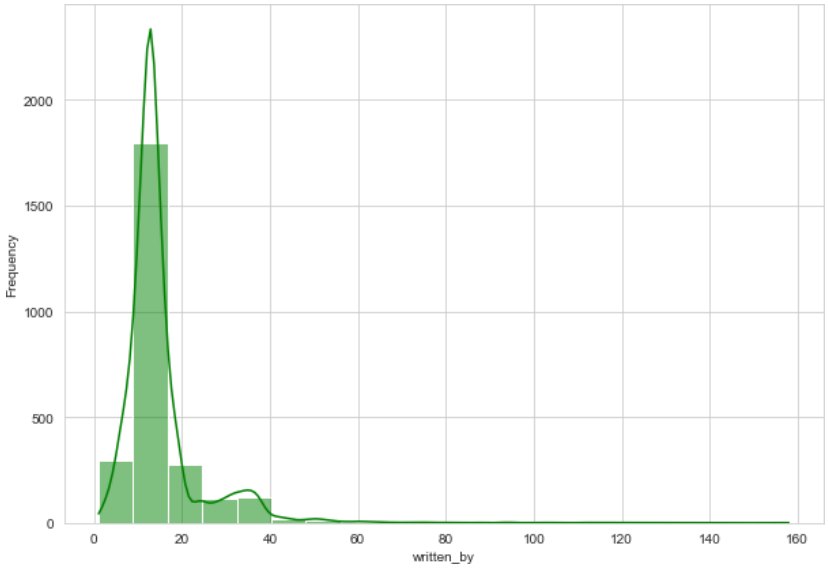
We can see that the data is equally distributed and balanced well. But fake data (1) has the maximum value than not fake (0) and the difference between them is 26

Histplots for checking the distribution

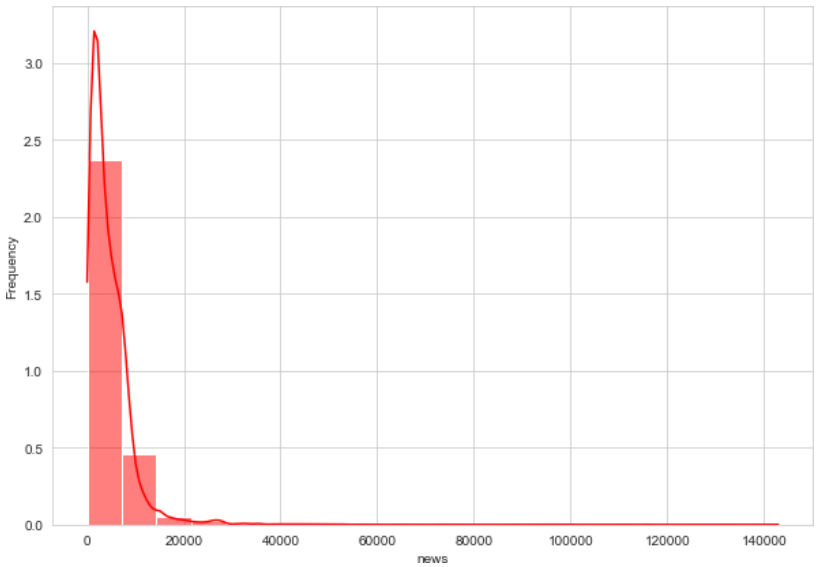


-> Majority of the comments are of length between 70 - 80, where the maximum length is above 400 and minimum length is 3.

-> The median value from the plot is approximately 75.



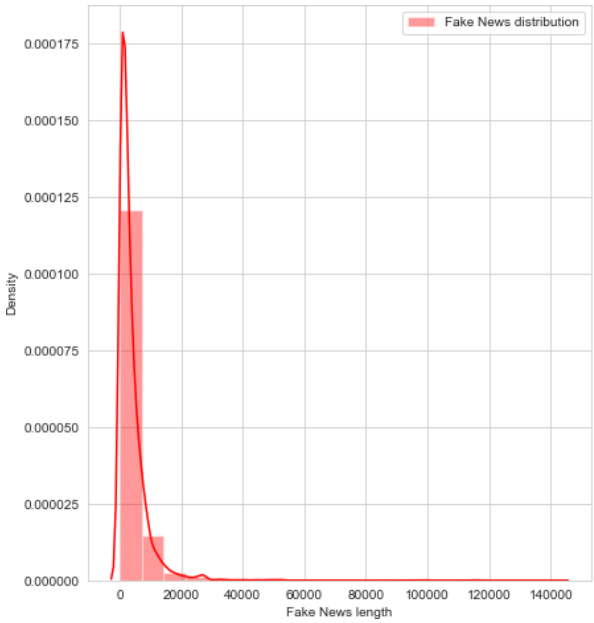
We can see that the majority length of the author name is of 10 words and the maximum length is nearly 160 words

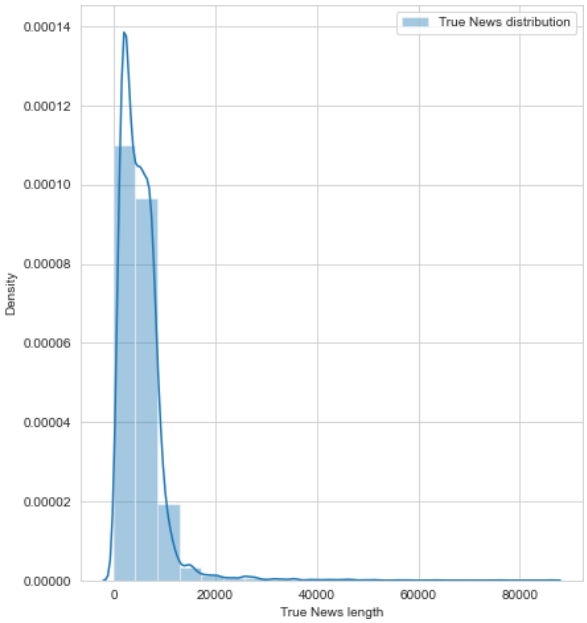


Majority of the news are of length below 1000, where maximum length is above 140000.

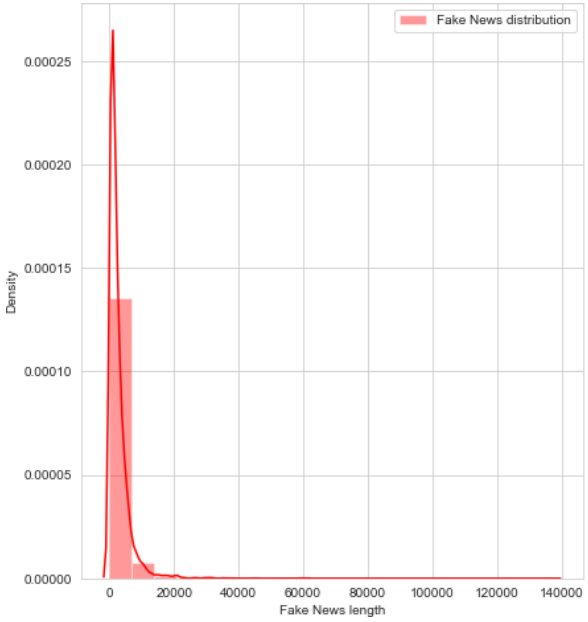
Plotting features before and after cleaning the data

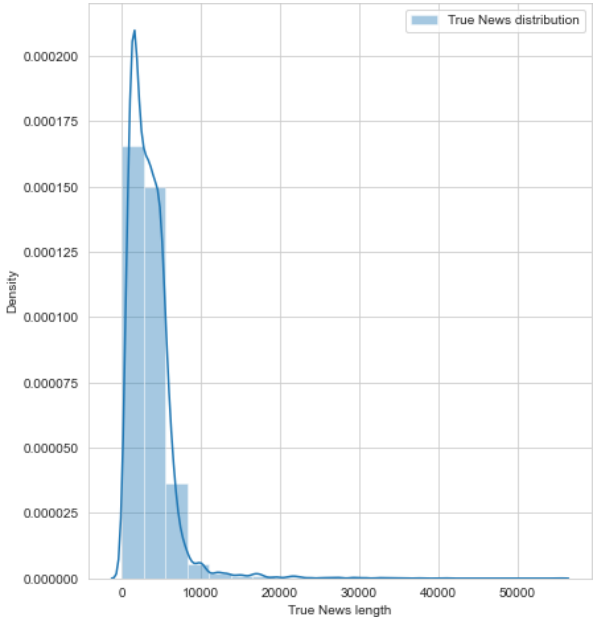
Before:





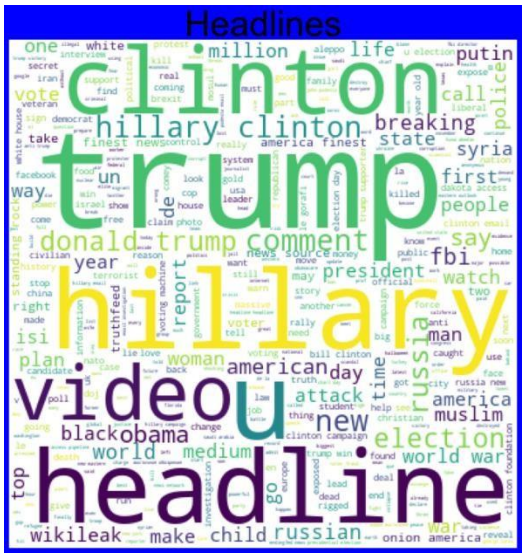
After:

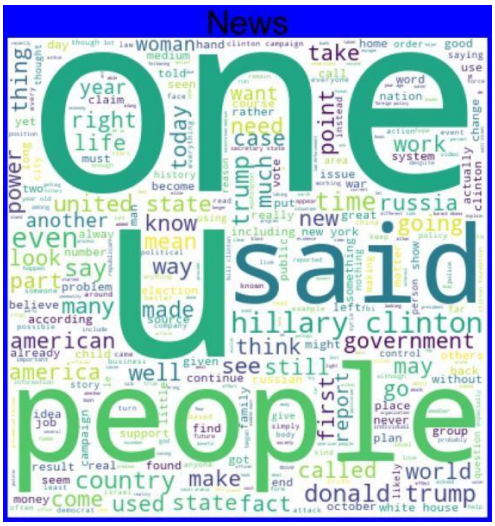




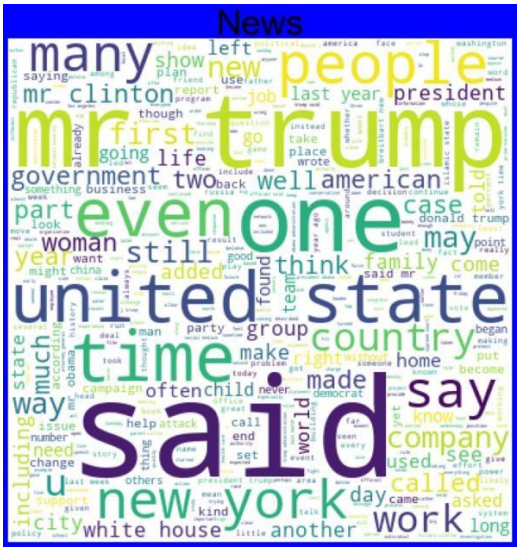
Plotting wordcloud for the features

For fake data:





For True data:



CONCLUSION

-> After the completion of this project, we got an insight of how to pre-process the data, analysing the data and building a model.

-> First, we imported the dataset which had nearly 20000 records.

-> We did all the required pre-processing steps like checking null values, datatypes check, dropping unnecessary columns, etc.

-> We did the Exploratory Data Analysis using various plots and recorded the observations.

-> Using NLP, we pre-processed the comment text and did other steps like:

- Removing Punctuations and other special characters
- Splitting the comments into individual words
- Removing Stop Words
- Stemming and Lemmatising
- Applying Count Vectorizer
- Plotting wordcloud for knowing the weightage of words used

-> We created many new features like length of words before pre-processing and after pre-processing in order to know the words cleaned after the necessary steps.

-> We applied Tf-idf Vectorizer for scaling the data into number vectors and for x feature we combined the written_by, news and headlines together.

-> Then, we split the data using train_test_split and then we started the model building process by running as many algorithms in a for loop, with difference metrics like cross_val_score, confusion matrix, auc_score, log loss, precision, recall, f1_score, etc.

-> We found that LogisticRegression and RandomForestClassifier were performing well. The next step was to perform hyperparameter tuning

technique to these models for finding out the best parameters and trying to improve our scores.

-> LogisticRegression Algorithm gave us good scores and metric values than RandomForestClassifier and therefore we finalized it as the best model.

-> We finalized the model by predicting the outputs, saving the model and storing the results in a csv file.

Problems faced while working in this project:

- More computational power was required.
- More missing data were present in the dataset.
- Loss was more for some algorithms.