Fake News Detection Project

Submitted by:

Shama Tanweer

Internship-12

ACKNOWLEDGMENT

I would like to express my sincere gratitude to FlipRobo Technologies for supporting me throughout the internship and giving me the opportunity to explore the depth of Data Science by providing multiple projects like this.

Separately, I would like to thank:

- SME khushboo Garg
- Data Trained Team

I have taken help from following sites whenever stuck:

- TF-IDF Vectorizer scikit-learn. Deep understanding TfidfVectorizer by… | by Mukesh Chaudhary | Medium
- https://stackoverflow.com/questions/66097701/how-can-i-fix-this-warning-in-xgboost

INTRODUCTION

Business Problem Framing

Fake news is a form of news consisting of deliberate disinformation or hoaxes spread via traditional news media or online social media. Such news items may contain false and/or exaggerated claims, and may end up being viralized by algorithms, and users may end up in a filter bubble. They may or may not contain spam words. As the scourge of "fake news" continues to plague our information enviromrnt,attention has turned towards devising automated solutions for detecting problemetic online content.

The goal of this project is to use natural language processing techniques to automate stance detection, since it is not practical for humans to fact check every piece of information produced by the media

Conceptual Background of the Domain Problem

The idea of fake news is often referred to as click-bait in social trends and is defined as a "made up story with an intention to deceive, geared towards getting clicks". Clickbait is an advertising tool used to get the attention of

users. Sensational headlines or news are often used as clickbait that navigate the user to advertisements.

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

➢ Review of Literature

Fake news is not a new concept. Before the era of digital technology, it was spread through mainly yellow journalism with focus on sensational news such as crime, gossip, disasters and satirical news.With the widespread dissemination of information via digital media platforms, it is of utmost importance for individuals and societies to be able to judge the credibility of it. Fake news is not a recent concept, but it is a commonly occurring phenomenon in current times. The consequence of fake news can range from being merely annoying to influencing and misleading societies or even nations. A variety of approaches exist to identify fake news

➢ Motivation for the Problem Undertaken

The exposure to real world data and the opportunity to deploy my skills in solving a real time problem has been the primary objective. This project was highly motivated project as it includes the real time problem of fake news which if we see are getting bigger as people are working hard to build good and real content and a single false news can ruin many things The main motivation was to classify the news in order to bring awareness and reduce unwanted chaos.

Analytical Problem Framing

Mathematical/Analytical modelling of problem

- The dataset provided here has a shape of (20800, 6). Which means it has 20800 rows and 6 columns.
- The target or the dependent variable named “Label” have two distinct values 0 and 1. Where 0 represents the news which are not fake or authentic while 1 represents 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 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 tfidf vectorizer then those processed data will be used to create best model using various classification based supervised machine learning algorithms like Logistic Regression, , Multinomial NB, Random Forest Classifier etc
- Dataset contains null value.

Data Sources and their formats

- The data is provide to us from our clint database.The sample data is in .csv format
- The sample data for reference is shown below.

Data Acquisition

```
df=pd.read_csv('train_news.csv')
df
```

	Unnamed: 0	id	headline	written_by	news	label
0	0	9653	Ethics Questions Dogged Agriculture Nominee as...	Eric Lipton and Steve Eder	WASHINGTON — In Sonny Perdue's telling, Geo...	0
1	1	10041	U.S. Must Dig Deep to Stop Argentina's Lionel ...	David Waldstein	HOUSTON — Venezuela had a plan. It was a ta...	0
2	2	19113	Cotton to House: 'Do Not Walk the Plank and Vo...	Pam Key	Sunday on ABC's "This Week," while discussing ...	0
3	3	6868	Paul LePage, Besieged Maine Governor, Sends Co...	Jess Bidgood	AUGUSTA, Me. — The beleaguered Republican g...	0
4	4	7596	A Digital 9/11 If Trump Wins	Finian Cunningham	Finian Cunningham has written extensively on...	1
...	...	...	...	...	...	...
20795	20795	5671	NaN	NeverSurrender	No, you'll be a dog licking of the vomit of yo...	1
20796	20796	14831	Albert Pike and the European Migrant Crisis	Rixon Stewart	By Rixon Stewart on November 5, 2016 Rixon Ste...	1
20797	20797	18142	Dakota Access Caught Infiltrating Protests to ...	Eddy Lavine	posted by Eddie You know the Dakota Access Pip...	1
20798	20798	12139	How to Stretch the Summer Solstice - The New Y...	Alison S. Cohn	It's officially summer, and the Society Boutiq...	0
20799	20799	15660	Emory University to Pay for '100 Percent' of U...	Tom Ciccotta	Emory University in Atlanta, Georgia, has anno...	0

20800 rows × 6 columns

DataSet description

There are 6 columns in the dataset provided:
The description of each of the column is given below:

- "id": Unique id of each news article
- "headline": It is the title of the news.
- "news": It contains the full text of the news article
- "Unnamed:0": It is a serial number
- "written_by": It represents the author of the news article
- "label": It tells whether the news is fake (1) or not fake (0).

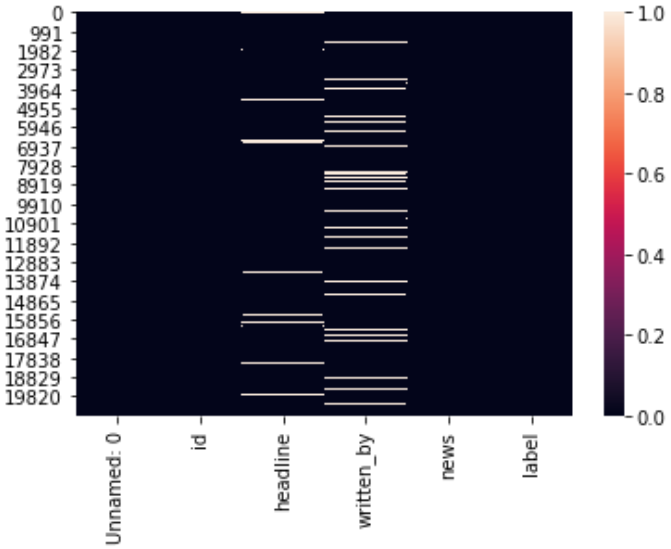
• Identification of possible problem-solving approaches (methods)

Used the following process for problem solving
1. Data Preprocessing
2. Building word dictionary
3. Feature extraction
4. Training classifiers
5. Testing
6. Performance evaluation using multiple metrics

Data Pre-processing

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 various ML model.We have applied various methods for data preprocessing methods in this project .

- First we check shape by using(df.shape)
- Then checked datatype of various features & found that all features are of int type except headline,written_by,news which are of object datatype
- checking for null values in each column:



It clearly shows that null values are present in the dataset,which needs to be removed.

- **Treating null values**

```
# filling 'Written_by' feature with unknown because sometimes there are anonymus authors,...
# filling up empty values in 'headline'  with 'No Headline'
# Dropping empty values in rows because we are detecting fake news here and for this news is needed..

df['written_by'].fillna('Unknown ',inplace=True)
df['headline'].fillna('no headlines ',inplace=True)
df.dropna(subset=['news'],inplace=True)
```

Dropping features

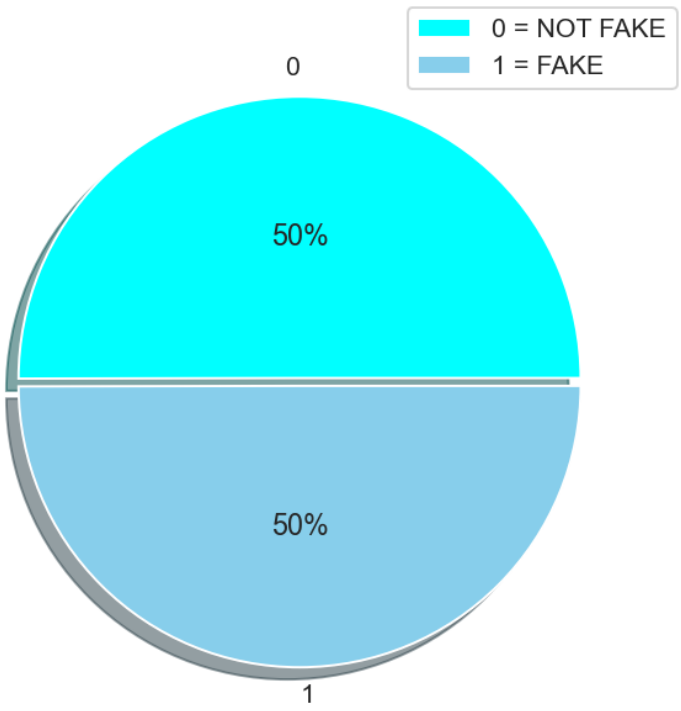
```
# Let's drop Unnamed: 0  & id from dataset as it does not seem importan
t
df.drop(['Unnamed: 0','id'],axis=1,inplace=True)
```

Checking distribution of fake and real news

```
#Ratio
print ('Fake = ', round(len(df[df['label']==1]) / len(df.label),2)*100,'%')
print ('Not Fake  = ', round(len(df[df['label']==0]) / len(df.label),2)*100,'%')
```

```
Fake =  50.0 %
Not Fake  =  50.0 %
```

```
lb=df['label'].value_counts().index.tolist()
val=df['label'].value_counts().values.tolist()
exp=(0.025,0)
clr=('cyan','skyblue')
plt.figure(figsize=(10,6),dpi=140)
sns.set_context('talk',font_scale=0.4)
sns.set(style='whitegrid')
plt.pie(x=val,explode=exp,labels=lb,colors=clr,autopct='%2.0f%%',pctdistance=0.5, shadow=True,radius=0.9)
plt.legend(["0 = NOT FAKE",'1 = FAKE'])
plt.show()
```



- We see that both news are equally distributed .ie dataset is balanced which is good as it will help our model to classify more accurately, so we should expect good accuracy score.

- Cleaning the raw data-It involves deletion of words or special characters that do not add meaning to the text.
  - Important cleaning steps are:
    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 most suitable base form by using lemmatization

```
# 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, spac
es, 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 char
acters left)
    News =re.sub(r'\s+[a-zA-Z]\s+', ' ', News)
    # Removing extra whitespaces
    News =re.sub(r'\s+', ' ', News, flags=re.I)

    #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 o
r not,
    News =[word for word in News if not word in set(STOPWORDS)]
```

```
    # 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_po
s function to get filtered using POS
    News = [(WordNetLemmatizer().lemmatize(text[0], get_pos(text[1])))f
or 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 lemmatizer to clean the words and removed special characters using Regexp Tokenizer and filter the words by removing stop words and then used lemmatizers and joined and return the filtered words.
Used TFIDF vectorizer to convert those text into vectors, and split the data and into test and train and trained various Machine learning algorithms.

Adding additional attribute :

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

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

```
#Adding New columns for processed headline and news
df['clean_headline']=df['headline'].apply(Processed_data)
df['clean_news']=df['news'].apply(Processed_data)
```

```
#New columns for length of headline and news after preprocessing

df['clean_length_headline']=df.clean_headline.str.len()
df['clean_length_news']=df.clean_news.str.len()
```

```
# Total length removal
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 1507844
Clean Length 1040606
Total Reduction =  467238
```

```
# Total length removal
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 94518924
Clean Length 56207800
Total Reduction =  38311124
```

After executing all these steps it was found that all the words & special characters were removed from the dataset which were of no use and consuming memory

➢ Data Inputs- Logic- Output Relationships

For this data's input and output logic we will analyse words frequency for each label, so that we can get the most frequent words that were used in different features.

Hardware and Software Requirements and Tools Used:

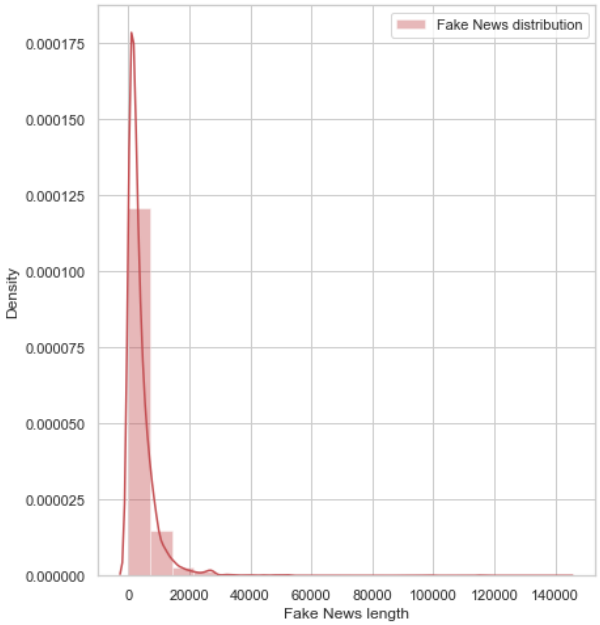
- Framework-Annconda
- IDE-Jupyter –Notebook
- Coding Language-Python
- Hardware used: system memory 8GB,
- Processor: core i3

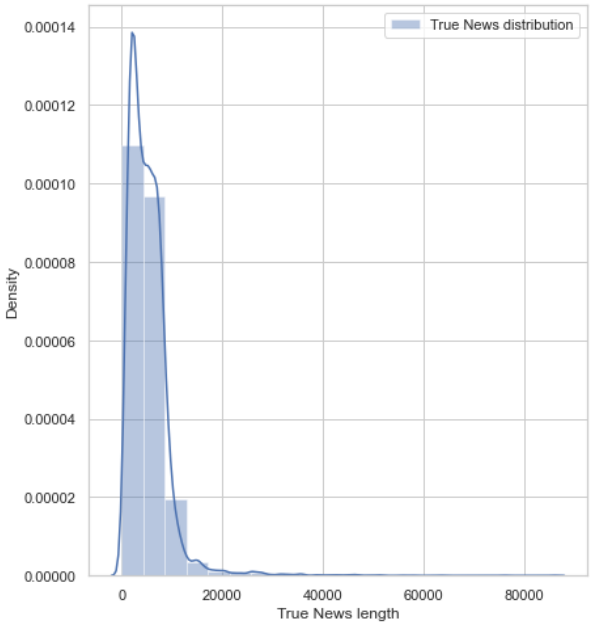
Libraries used: Below are the python library used in this project. •

- Pandas: To read the Data file in form of data.
- Matplotlib: This library is typically used to plot the figures for better visualisation 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 very useful joblib library for serialization purpose using which final model have been saved in this project.
- NLTK: Natural language took kit is one of the most used library for building NLP projects.

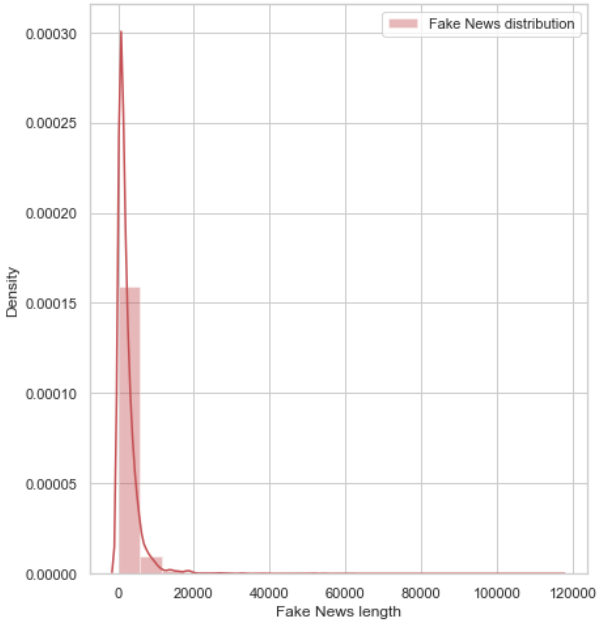
Then we have plotted graph to show distribution of word count before cleaning and after cleaning

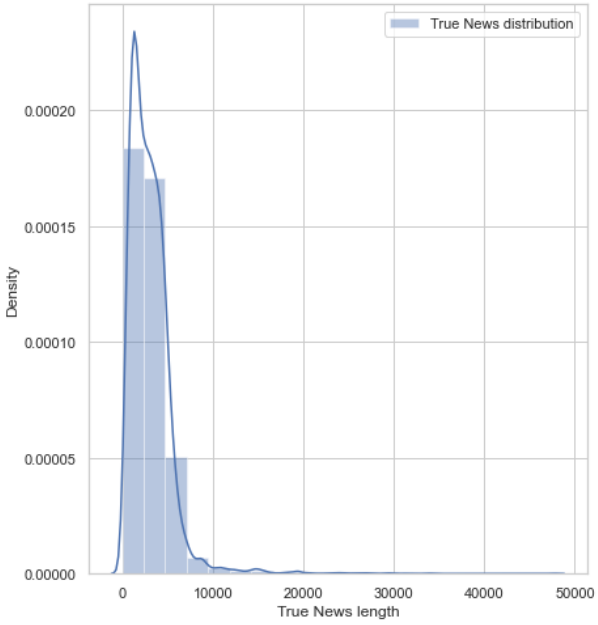
Before cleaning:





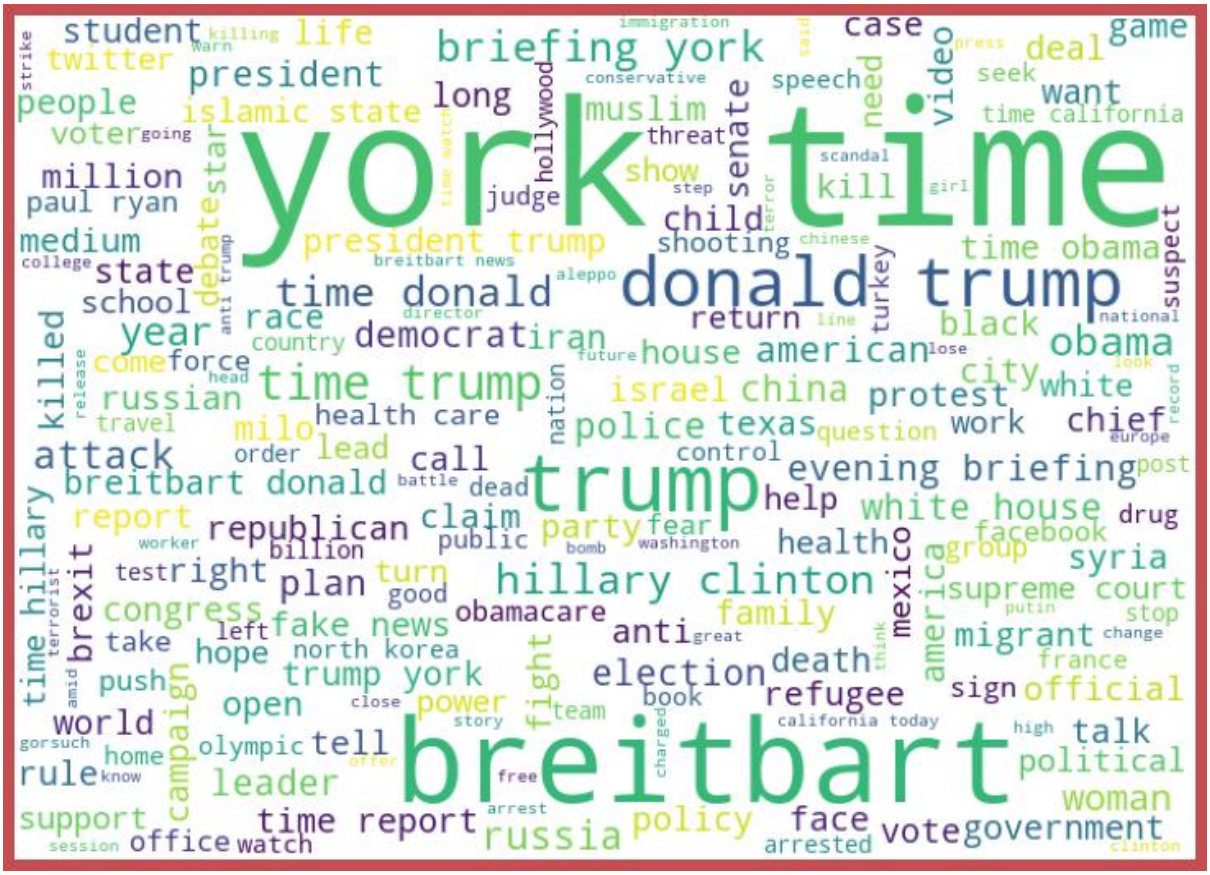
After cleaning





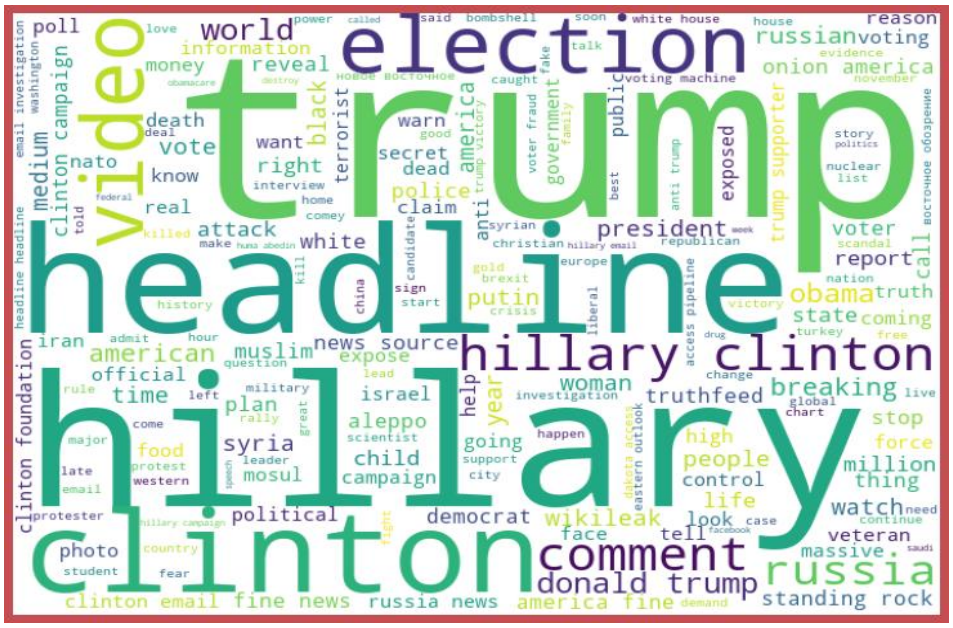
To get better view of words contained in news . A word dictionary (wordcloud) was made showing the words highly occurred in fake and real news for both headline and news column.

```
#Getting sense of loud words in real News Headline

not_fake = df['clean_headline'][df['label']==0]

not_fake_cloud = WordCloud(width=700,height=500,background_color='white
',max_words=200).generate(' '.join(not_fake))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(not_fake_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



#Getting sense of loud words in Fake News Headline

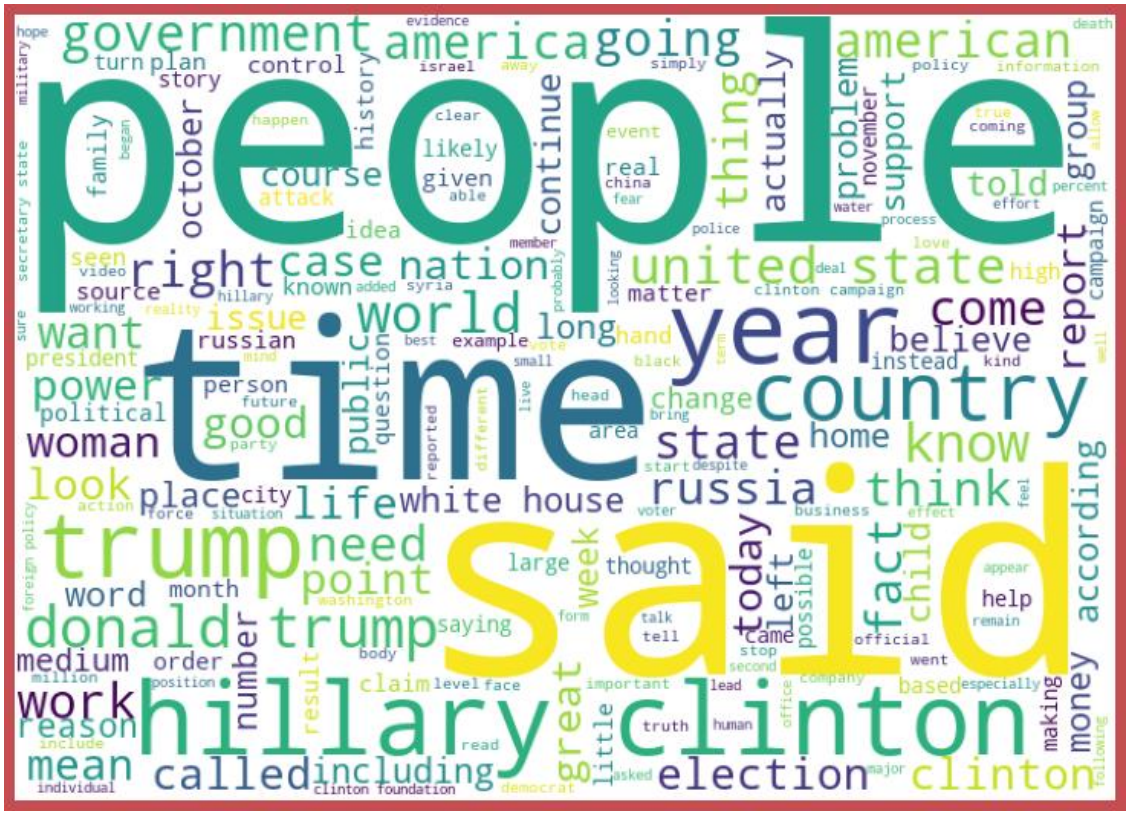
```
fake = df['clean_headline'][df['label']==1]

fake_cloud = WordCloud(width=700,height=500,background_color='white',ma
x_words=200).generate(' '.join(fake))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(fake_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



#Getting sense of loud words in Not Fake News - Articles

```
not_fake = df['clean_news'][df['label']==0]

not_fake_cloud = WordCloud(width=700,height=500,background_color='white
',max_words=200).generate(' '.join(not_fake))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(not_fake_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```


```
#Getting sense of loud words in Fake News Articles

fake = df['clean_news'][df['label']==1]

fake_cloud = WordCloud(width=700,height=500,background_color='white',ma
x_words=200).generate(' '.join(fake))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(fake_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



From the above we can see that most frequent words on both labels and we can observe the words which are leading to fake new are trump, Clinton, prison,november, etc and words which are leading to real news are said, agriculture,police ,questions etc, so we can clearly see that above dataset extensively deals with news around US presidential elections between Trump and Clinton.

➢ State the set of assumptions (if any) related to the problem under consideration

Only assumptions which were taken related to the problem was that we dropped the **unnamed:0 & id column** as it was not of much importance

Model/s Development and Evaluation

➢ problem-solving approaches:

Understanding the problem is the first crucial steps in solving any problem. From the given dataset it can be concluded that it is a binary Classification problem.Therefore I run my preprocessed data on 6 classification algorithm.

Training Classifier:

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

```
# creating the TF-IDF vectorizer fn in order to convert the tokens from the train documents into vectors so that machine can do
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['written_by'] + 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:  (20761, 79062)
```

```
y = df['label']
```

```
y.shape
```

```
(20761,)
```

Testing of Identified Approaches (Algorithms)

- **Below are the algorithms used for this problem:**

```
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 xgboost import XGBClassifier
from sklearn.ensemble import AdaBoostClassifier
```

Run and Evaluate selected models

In my approach I have first prepared a method which gives all necessary classification metrics of an algorithm like classification metrics , auc_roc score, confusion matrix,log_loss.

```
# 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=.3
0,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=[]
Acc_score=[]
cvs=[]
rocscore=[]
logloss=[]
#For Loop to Calculate Accuracy Score, Cross Val Score, Classification
Report, Confusion Matrix

for name,model in models:
    print('**************************',name,'**********************
****')
    print('\n')
    Model.append(name)
    print(model)
    print('\n')

#Now here I am 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,r
andom_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')
```

```
#....log _loss…….

    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_pre
d))
    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')
```

:

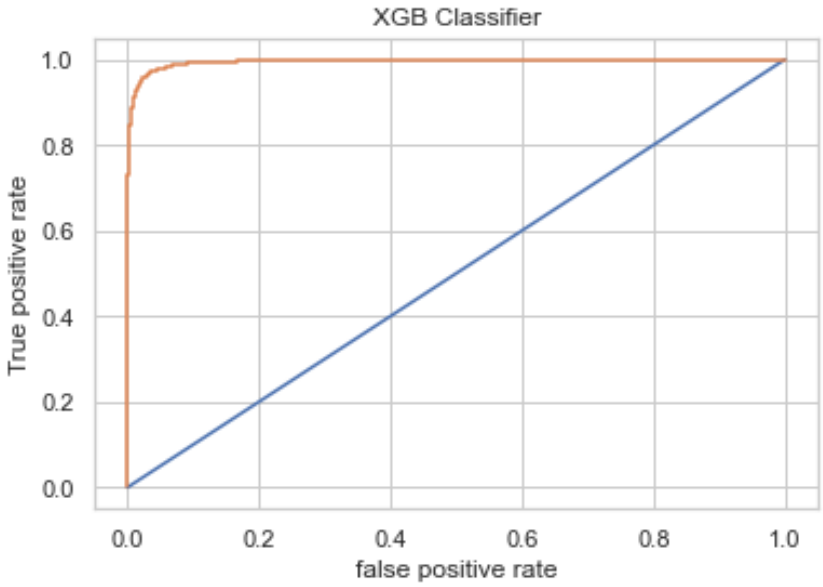
	Model	Accuracy Score	Cross Val Score	Log_Loss	Roc_Auc_curve
0	LogisticRegression	94.782469	98.746218	1.802091	94.782367
1	MultinomialNB()	84.026328	97.430162	5.517114	84.019065
2	DecisionTreeClassifier	93.482100	93.227825	2.251231	93.482392
3	RandomForestClassifier	92.117515	98.335620	2.722532	92.115835
4	AdaBoostClassifier	94.429282	98.466522	1.924080	94.429258
5	XGBClassifier	97.238722	99.543536	0.953722	97.238675

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

Final Model

```
# Using XGBClassifier for Final Model...
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=83,test_size=.30)
XG=XGBClassifier(eval_metric='mlogloss')
XG.fit(x_train,y_train)
XG.score(x_train,y_train)
XGpred=XG.predict(x_test)
print('Accuracy Score:','\n',accuracy_score(y_test,XGpred))
print('Log_Loss:','\n',log_loss(y_test,XGpred))
print('Confusion Matrix:','\n',confusion_matrix(y_test,XGpred))
print('Classification Report:','\n',classification_report(y_test,XGpred))
```

```
Accuracy Score:
 0.9680526569272756
Log_Loss:
 1.1034360024257013
Confusion Matrix:
 [[3011  108]
 [  91 3019]]
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      3119
           1       0.97      0.97      0.97      3110

    accuracy                           0.97      6229
   macro avg       0.97      0.97      0.97      6229
weighted avg       0.97      0.97      0.97      6229
```



Saving the best model

```
# Saving the best model.
import joblib
```

```
joblib.dump(XG,'Fake_news_Predict.pkl')
```

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

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

➢ Key Metrics for success in solving problem under consideration

- When it comes to evaluation of a data science model's performance, sometimes accuracy may not be the best indicator.
- Some problems that we are solving in real life might have a very imbalanced class and using accuracy might not give us enough confidence to understand the algorithm's performance.
- In the fake news problem that we are trying to solve,the data is totally balanced. so accuracy score nearly tells the right predictions.so the problem of overfitting in this problem is nearly not to occur. So here, we are using accuracy score to find better model.

CONCLUSION

➢ Key Findings and Conclusions of the Study

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 US presential 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 and news Facebook's CEO Mark Zuckerberg also got extensively question by congress.

➢ Learning Outcomes of the Study in respect of Data Science

It is possible to classify news content into the required categories of authentic and fake news however there will be always a bias to this kind of classification which depends on the behavioural pattern of the listener. However, using this kind of project an awareness can be created to know what is fake and authentic.

➢ Limitations of this work and Scope for Future Work

Machine Learning Algorithms like XGBoost,Adaboost and randomforest Classifier took enormous amount of time to build the model.

Using Hyper-parameter tuning for XGB would have resulted in some more accuracy.

Thankyou