FAKE NEWS DETECTION USING NLP

Introduction

In today's world, the credibility of news is crucial. With the rapid spread of information on social media, distinguishing fake news from real news is a pressing concern. Natural Language Processing tools offer a solution by using historical data to classify news articles accurately.

Problem Statement

False or misleading information can spread quickly and have real-world consequences. Sensational headlines are a long-standing issue, and on social media, the impact is amplified. We aim to classify news articles as fake or real, addressing this problem.

Objective

Our primary objective is to develop a powerful model that accurately categorizes news articles as fake or real. We will start by exploring the data and then build a robust classification model for this purpose.

Importing Libraries:

Let's begin by importing the essential libraries for our analysis and introducing our dataset.

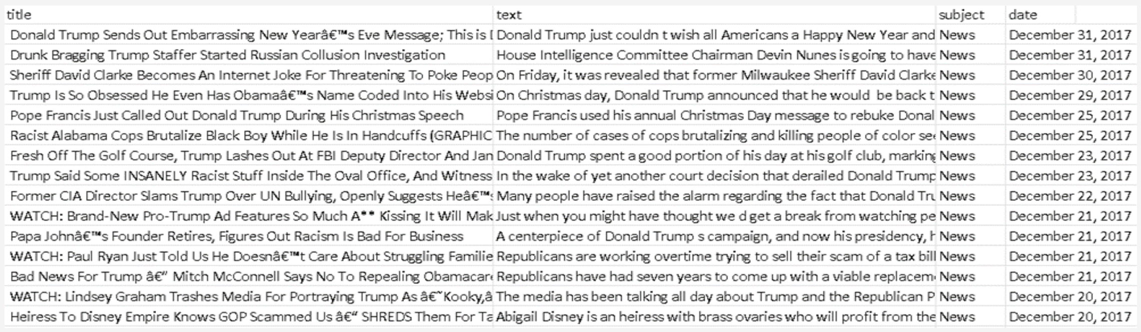
```python
#Basic libraries
import pandas as pd
import numpy as np

#Visualization libraries
import matplotlib.pyplot as plt
from matplotlib import rcParams
import seaborn as sns
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]
import cufflinks as cf
cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)

#NLTK libraries
import nltk
import re
import string
```

```python
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

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

#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
```

Importing the Dataset:

Let's introduce our dataset and explore its contents.

i) Data Source



title	text	subject	date
Donald Trump Sends Out Embarrassing New Yearâ€™s Eve Message; This is I	Donald Trump just couldn t wish all Americans a Happy New Year and	News	December 31, 2017
Drunk Bragging Trump Staffer Started Russian Collusion Investigation	House Intelligence Committee Chairman Devin Nunes is going to have	News	December 31, 2017
Sheriff David Clarke Becomes An Internet Joke For Threatening To Poke Peop	On Friday, it was revealed that former Milwaukee Sheriff David Clarke	News	December 30, 2017
Trump Is So Obsessed He Even Has Obamaâ€™s Name Coded Into His Websi	On Christmas day, Donald Trump announced that he would be back t	News	December 29, 2017
Pope Francis Just Called Out Donald Trump During His Christmas Speech	Pope Francis used his annual Christmas Day message to rebuke Donal	News	December 25, 2017
Racist Alabama Cops Brutalize Black Boy While He Is In Handcuffs (GRAPHIC	The number of cases of cops brutalizing and killing people of color se	News	December 25, 2017
Fresh Off The Golf Course, Trump Lashes Out At FBI Deputy Director And Jan	Donald Trump spent a good portion of his day at his golf club, marking	News	December 23, 2017
Trump Said Some INSANELY Racist Stuff Inside The Oval Office, And Witness	In the wake of yet another court decision that derailed Donald Trump	News	December 23, 2017
Former CIA Director Slams Trump Over UN Bullying, Openly Suggests Heâ€™s	Many people have raised the alarm regarding the fact that Donald Tru	News	December 22, 2017
WATCH: Brand-New Pro-Trump Ad Features So Much A** Kissing It Will Mak	Just when you might have thought we d get a break from watching pe	News	December 21, 2017
Papa Johnâ€™s Founder Retires, Figures Out Racism Is Bad For Business	A centerpiece of Donald Trump s campaign, and now his presidency, h	News	December 21, 2017
WATCH: Paul Ryan Just Told Us He Doesnâ€™t Care About Struggling Familie	Republicans are working overtime trying to sell their scam of a tax bill	News	December 21, 2017
Bad News For Trump â€” Mitch McConnell Says No To Repealing Obamacar	Republicans have had seven years to come up with a viable replacem	News	December 21, 2017
WATCH: Lindsey Graham Trashes Media For Portraying Trump As â€˜Kooky,â	The media has been talking all day about Trump and the Republican P	News	December 20, 2017
Heiress To Disney Empire Knows GOP Scammed Us â€“ SHREDS Them For Ta	Abigail Disney is an heiress with brass ovaries who will profit from the	News	December 20, 2017

Dataset Link: https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

```
#reading the fake and true datasets
fake_news = pd.read_csv('Fake.csv')
true_news = pd.read_csv('True.csv')

# print shape of fake dataset with rows and columns and
information
print ("The shape of the  data is (row, column):"+
str(fake_news.shape))
print (fake_news.info())
print("\n --------------------------------------- \n")
```

Preprocessing and Cleaning:

Before we delve into exploratory data analysis (EDA) and model building, we need to perform crucial preprocessing steps. Let's begin by creating the target column.

Creating the target column

Let's create the target column for both fake and true news. Here we are gonna denote the target value as '0' incase of fake news and '1' incase of true news

```
#Target variable for fake news
fake_news['output']=0
#Target variable for true news
true_news['output']=1
```

Concatenating Title and Text of News

```
#Concatenating and dropping for fake news
fake_news['news']=fake_news['title']+fake_news['text']
fake_news=fake_news.drop(['title', 'text'], axis=1)
```

```
#Concatenating and dropping for true news
true_news['news']=true_news['title']+true_news['text']
true_news=true_news.drop(['title', 'text'], axis=1)

#Rearranging the columns
fake_news = fake_news[['subject', 'date', 'news','output']]
true_news = true_news[['subject', 'date', 'news','output']]
```

Converting the Date Columns to Datetime Format

We can utilize `pd.to_datetime` to convert our date columns into the desired date format.

```
fake_news['date'].value_counts()
```

OUTPUT:

```
May 10, 2017        46
 May 26, 2016        44
May 6, 2016         44
May 5, 2016         44
May 11, 2016        43
                     ..
December 9, 2017      1
December 4, 2017      1
November 19, 2017     1
November 20, 2017     1
Jul 19, 2015          1
Name: date, Length: 1681, dtype: int64
```

Appending Two Datasets

To provide the model with a single dataset, we should append both the true and fake news data and preprocess it further for EDA.

```
frames = [fake_news, true_news]
news_dataset = pd.concat(frames)
news_dataset
```

Remove Stop words

Stop words are commonly used words (e.g., "the," "a," "an," "in") that search engines have been programmed to ignore. They are omitted during the indexing of entries for searching and when delivering search results in response to a query.

```
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()
```

OUTPUT

```
        subject  date        news        output
0       News     2017-12-31  donald trump sends embarrassing new year’s eve...     0
1       News     2017-12-31  drunk bragging trump staffer started russian c...      0
2       News     2017-12-30  sheriff david clarke becomes internet joke thr...      0
3       News     2017-12-29  trump obsessed even obama’s name coded website...    0
4       News     2017-12-25  pope francis called donald trump christmas spe...     0
```

Stemming & Vectorization:

```
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.

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

OUTPUT

```
        news
0       donald trump sends embarrassing new year’s eve...
1       drunk bragging trump staffer started russian c...
2       sheriff david clarke becomes internet joke thr...
3       trump obsessed even obama’s name coded website...
4       pope francis called donald trump christmas spe...
```

```
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)
```

```
corpus[1]
```

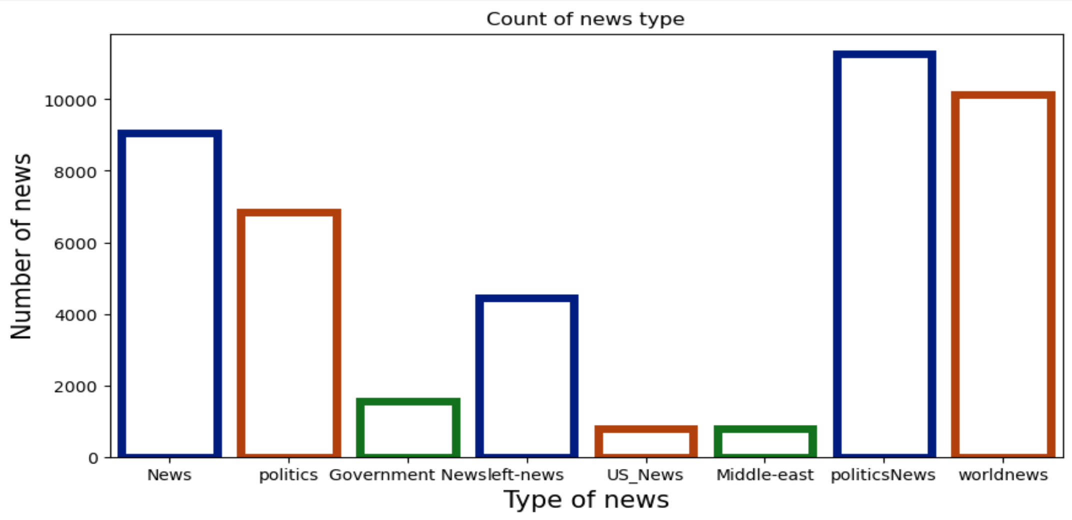
Feauture Extraction:

Data Visualization for News:

Count of News Subjects

```
#Plotting the frequency plot
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)
```

OUTPUT

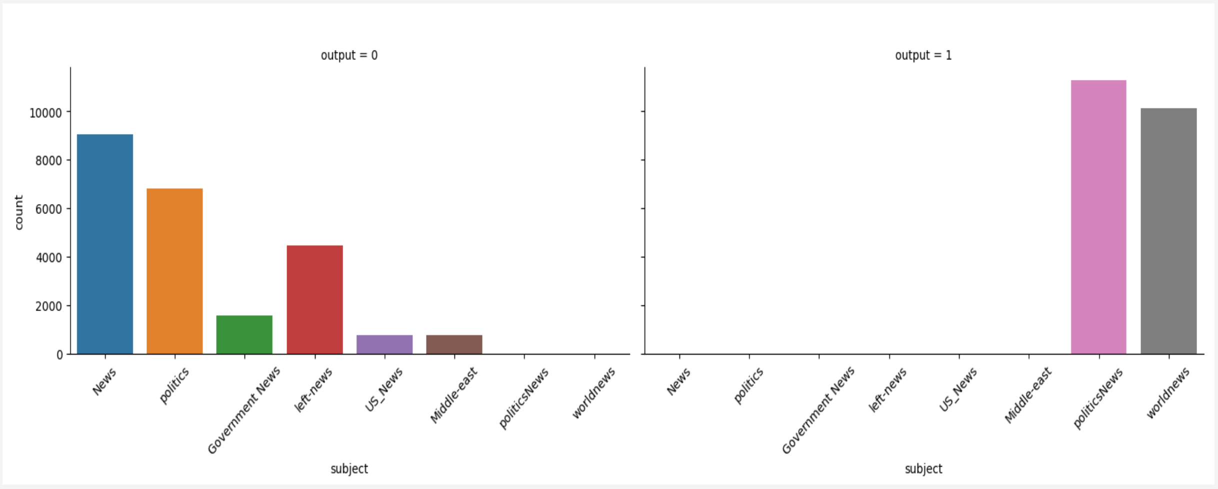


Count of News Subject based on true or fake

Let's examine the distribution of fake and true news to confirm whether our data is balanced.

```
g = sns.catplot(x="subject", col="output",
                data=clean_news, kind="count",
                height=4, aspect=2)
#Rotating the xlabels
g.set_xticklabels(rotation=45)
```

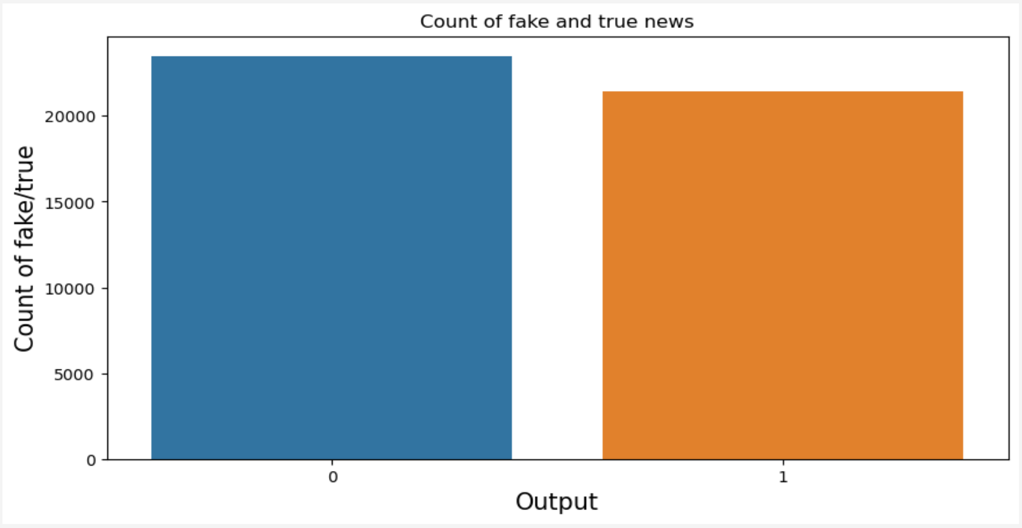
OUTPUT



Count of fake news and true news

```
ax=sns.countplot(x="output", 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)
```

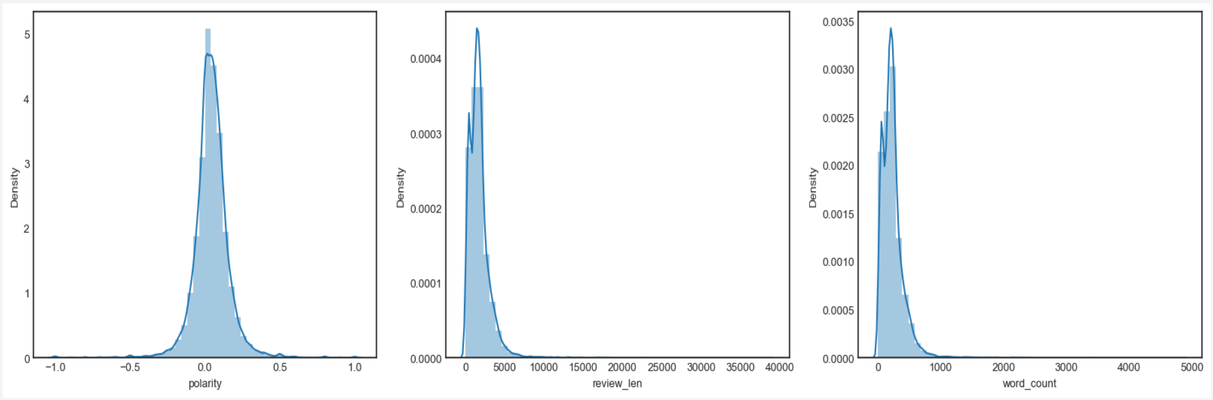
OUTPUT



Deriving new Feautures from the News

```
#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()
```

OUTPUT



N-gram Analysis

Top 20 words in News

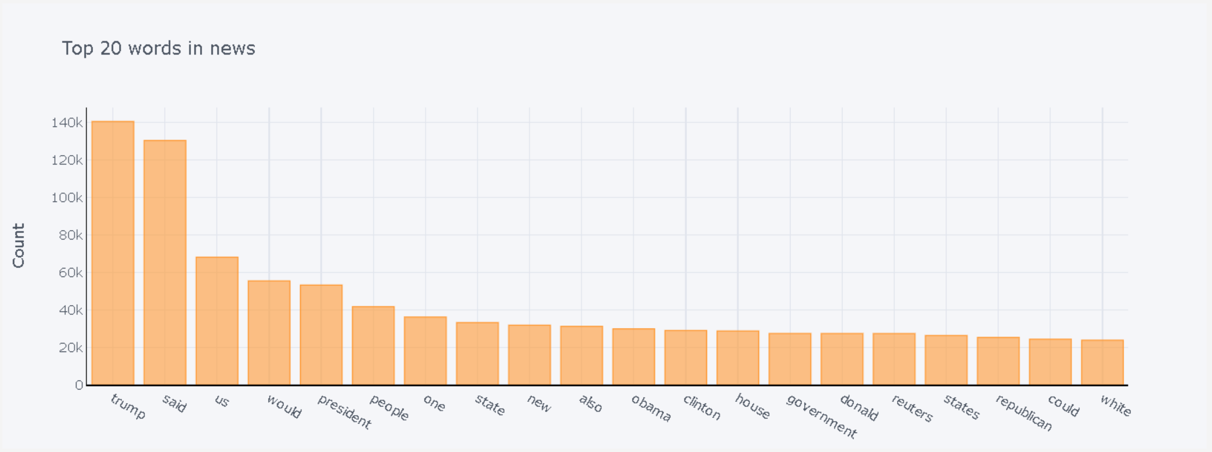
Let's look at the top 20 words from the news which could give us a brief idea on what news are popular in our dataset

```
#Function to get top n words
def get_top_n_words(corpus, n=None):
    vec = CountVectorizer().fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in
vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

#Calling function and return only top 20 words
common_words = get_top_n_words(clean_news['news'], 20)

#Printing the word and frequency
for word, freq in common_words:
    print(word, freq)
```

```
#Creating the dataframe of word and frequency
df1 = pd.DataFrame(common_words, columns = ['news' , 'count'])

#Group by words and plot the sum
df1.groupby('news').sum()['count'].sort_values(ascending=False).iplot(
   kind='bar', yTitle='Count', linecolor='black', title='Top 20 words in news')
```

OUTPUT

```
trump 140400
said 130258
us 68081
would 55422
president 53189
people 41718
one 36146
state 33190
new 31799
also 31209
obama 29881
clinton 29003
house 28716
government 27392
donald 27376
reuters 27348
states 26331
republican 25287
could 24356
white 23823
```



Top two words in News

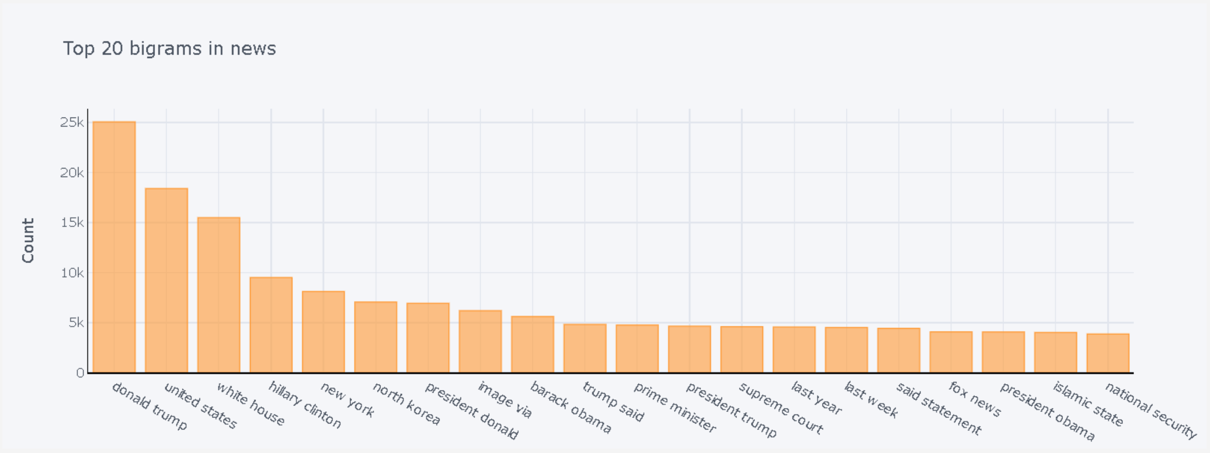
```
#Function to get top bigram words
def get_top_n_bigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(2, 2)).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in
vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1],
reverse=True)
    return words_freq[:n]

#Calling function and return only top 20 words
common_words = get_top_n_bigram(clean_news['news'], 20)

#Printing the word and frequency
for word, freq in common_words:
    print(word, freq)

#Creating the dataframe of word and frequency
df3 = pd.DataFrame(common_words, columns = ['news' , 'count'])

#Group by words and plot the sum
df3.groupby('news').sum()['count'].sort_values(ascending=False)
.iplot(
    kind='bar', yTitle='Count', linecolor='black', title='Top
20 bigrams in news')
```

OUTPUT

```
donald trump 25059
united states 18394
white house 15485
hillary clinton 9502
new york 8110
north korea 7053
president donald 6928
image via 6188
barack obama 5603
trump said 4816
prime minister 4753
president trump 4646
supreme court 4595
last year 4560
last week 4512
said statement 4425
fox news 4074
president obama 4065
islamic state 4014
national security 3858
```

donald trumpunited stateswhite househillary clintonnew yorknorth koreapresident donaldimage viabarack obamatrump saidprime ministerpresident trumpsupreme courtlast yearlast weeksaid statementfox newspresident obamaislamic statenational security05k10k15k20k25kExport to plot.ly »

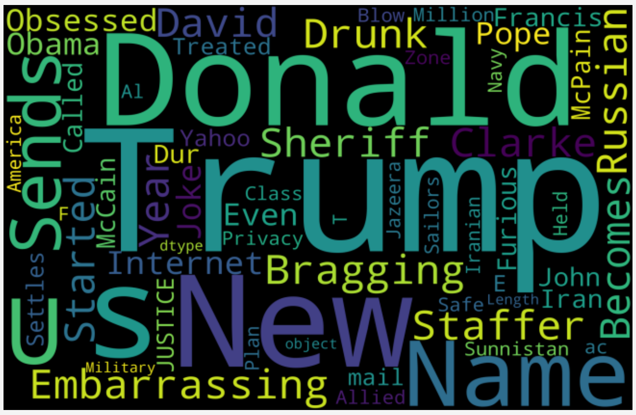
Top 20 bigrams in newsCount



Word Cloud of Fake and True News

```
text = fake_news["news"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

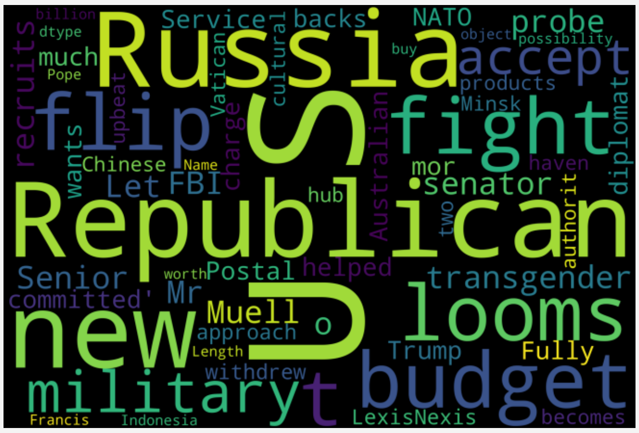
OUTPUT



```
text = true_news["news"]
```

```
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

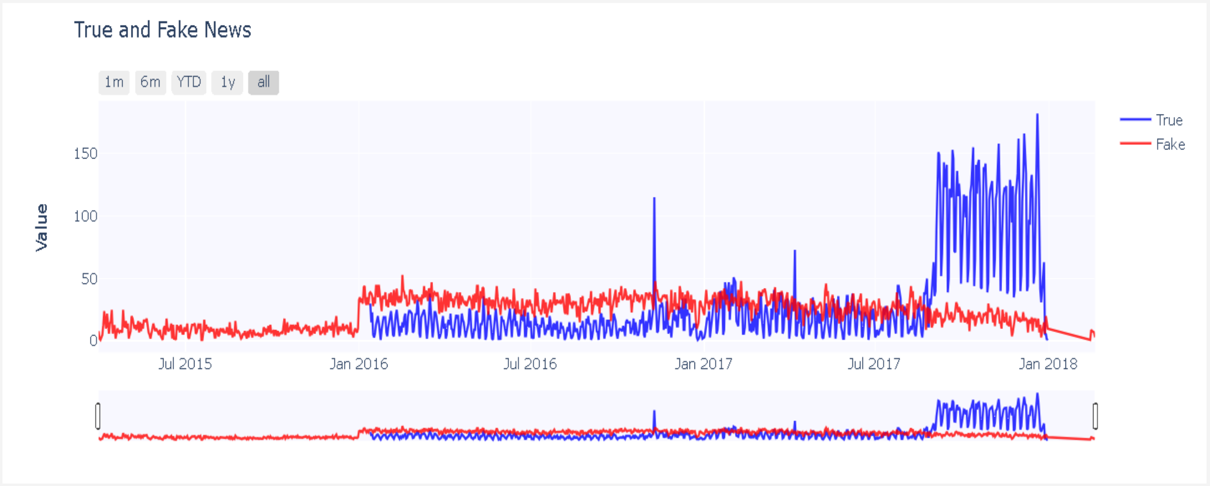
OUTPUT



Time series analaysis Fake and True News

```
#Creating the count of output based on date
fake=fake_news.groupby(['date'])['output'].count()
fake=pd.DataFrame(fake)
true=true_news.groupby(['date'])['output'].count()
true=pd.DataFrame(true)
#Plotting the time series graph
fig = go.Figure()
fig.add_trace(go.Scatter(
        x=true.index,
        y=true['output'],
        name='True',
    line=dict(color='blue'),
    opacity=0.8))
fig.add_trace(go.Scatter(
        x=fake.index,
        y=fake['output'],
        name='Fake',
    line=dict(color='red'),
    opacity=0.8))
fig.update_xaxes(
    rangeslider_visible=True,
      rangeselector=dict(
        buttons=list([
            dict(count=1, label="1m", step="month", stepmode="backward"),
            dict(count=6, label="6m", step="month", stepmode="backward"),
            dict(count=1, label="YTD", step="year", stepmode="todate"),
            dict(count=1, label="1y", step="year", stepmode="backward"),
            dict(step="all")
        ])
      )
)

fig.update_layout(title_text='True and Fake News',plot_bgcolor='rgb(248, 248,
255)',yaxis_title='Value')
fig.show()
```

OUTPUT



Model Training

Train test split (75:25)

Using train test split function we are splitting the dataset into 75:25 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=0.25,
random_state=0)
```

Model Building Fake News Classifier:

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """

    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:

```
#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()))
```

OUTPUT

Logistic Regression Test Accuracy: 0.9660040199274997

Decision Tree Test Accuracy: 0.9353049482414729

KNN Test Accuracy: 0.6119253084088696

Naive Bayes Test Accuracy: 0.9373328405462511

Logistic Regression with hyperparameter Tuning

```
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))
```

OUTPUT

```
LogisticRegression(C=24.420530945486497, random_state=0)

The mean accuracy of the model is: 0.9803065407235787
```

```
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)))
```

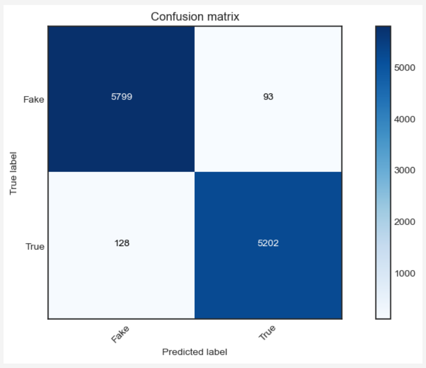
OUTPUT

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

Confusion Matrix

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

OUTPUT



Classification Report

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

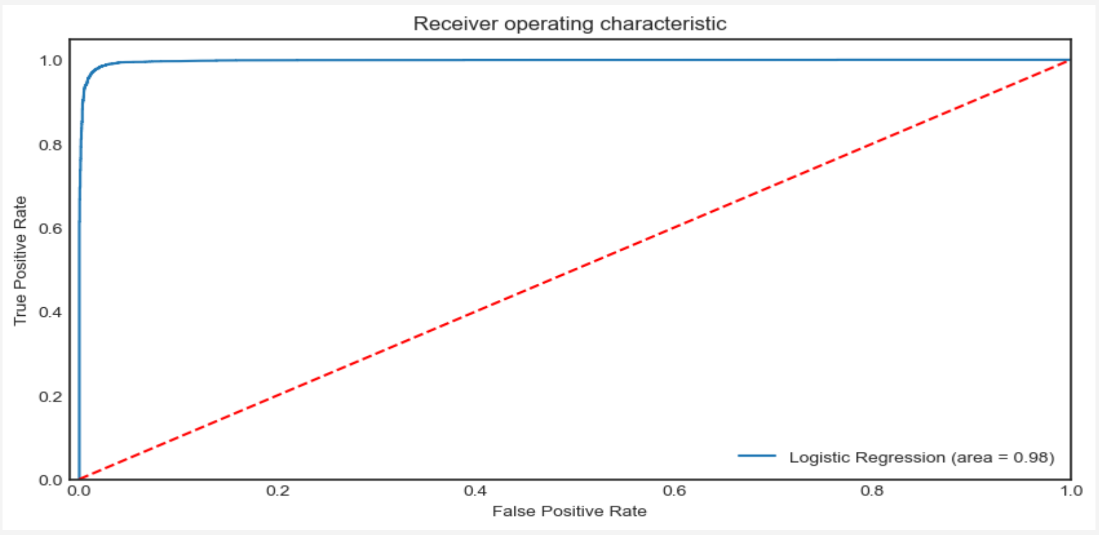
OUTPUT

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

           0       0.98      0.98      0.98      5892
           1       0.98      0.98      0.98      5330

    accuracy                           0.98     11222
   macro avg       0.98      0.98      0.98     11222
weighted avg       0.98      0.98      0.98     11222
```

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()
```

OUTPUT



Deep Learning -LSTM

```
#Creating the lstm model
embedding_vector_features=40
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=se
nt_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=['accurac
y'])
print(model.summary())

from tensorflow.keras.preprocessing.text import Tokenizer

# Load Data
true_df = pd.read_csv('True.csv')
fake_df = pd.read_csv('Fake.csv')
# Data Preprocessing
# Combine and label the data
true_df['label'] = 1
fake_df['label'] = 0
data = pd.concat([true_df, fake_df])

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['label'],
test_size=0.2, random_state=42)
```

```python
# Tokenization and Padding
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(X_train)
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)
X_train_padded = pad_sequences(X_train_seq, maxlen=200, padding='post',
truncating='post')
X_test_padded = pad_sequences(X_test_seq, maxlen=200, padding='post',
truncating='post')

# LSTM Model
model_lstm = Sequential()
model_lstm.add(Embedding(input_dim=5000, output_dim=128,
input_length=200))
model_lstm.add(LSTM(128))
model_lstm.add(Dense(1, activation='sigmoid'))

model_lstm.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model_lstm.fit(X_train_padded, y_train, epochs=5,
validation_data=(X_test_padded, y_test))

# Evaluate LSTM Model
y_pred_lstm = model_lstm.predict(X_test_padded)
y_pred_lstm = [1 if val > 0.5 else 0 for val in y_pred_lstm]
```

OUTPUT

```
Epoch 1/5
1123/1123 [==============================] - 201s 176ms/step - loss: 0.3753 -
accuracy: 0.8394 - val_loss: 0.2382 - val_accuracy: 0.9302
Epoch 2/5
1123/1123 [==============================] - 188s 168ms/step - loss: 0.2894 -
accuracy: 0.8883 - val_loss: 0.1022 - val_accuracy: 0.9734
Epoch 3/5
1123/1123 [==============================] - 179s 160ms/step - loss: 0.0904 -
accuracy: 0.9726 - val_loss: 0.0611 - val_accuracy: 0.9635
Epoch 4/5
1123/1123 [==============================] - 194s 173ms/step - loss: 0.0358 -
accuracy: 0.9900 - val_loss: 0.0131 - val_accuracy: 0.9964
Epoch 5/5
1123/1123 [==============================] - 190s 169ms/step - loss: 0.0072 -
accuracy: 0.9983 - val_loss: 0.0123 - val_accuracy: 0.9970
281/281 [==============================] - 18s 61ms/step
```

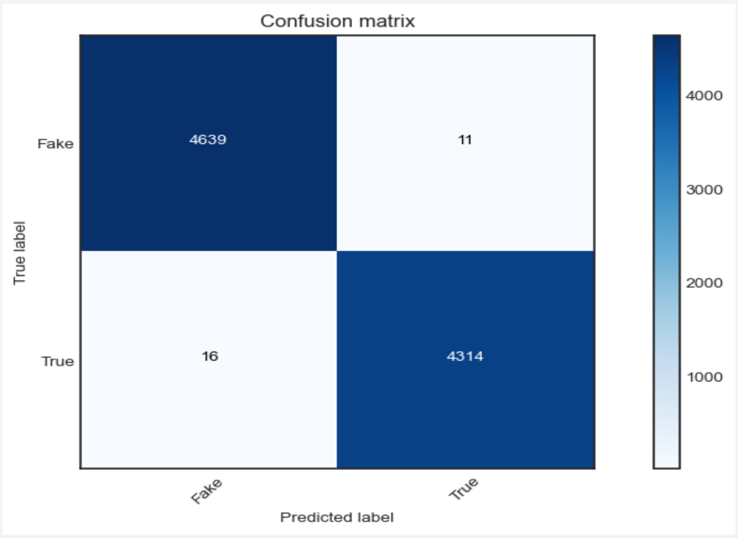
```python
print("LSTM Classification Report:")
print(classification_report(y_test, y_pred_lstm))
```

OUTPUT

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

           0       1.00      1.00      1.00      4650
           1       1.00      1.00      1.00      4330

    accuracy                           1.00      8980
   macro avg       1.00      1.00      1.00      8980
weighted avg       1.00      1.00      1.00      8980
```

Evaluation of Model

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



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

OUTPUT

```
0.9969933184855234
```

Conclusion:

Natural Language Processing (NLP) is a potent tool for combatting fake news by effectively identifying and categorizing it. The success of NLP models in this task may be influenced by factors such as feature correlation, particularly with categorical features like 'subject.' It underscores the importance of thorough feature consideration in fake news detection.