Fake News Detection

Introduction

Fake news is like a wildfire in the age of social media, especially during critical events like elections and pandemics such as Covid-19. It's spreading like a contagion, and it's becoming increasingly difficult to separate fact from fiction amidst the information overload. Taiwan, in the recent wave of Covid-19, has seen a deluge of fake news flooding platforms like LINE, Facebook, and PTT, a popular online forum. What's alarming is that many seniors are inadvertently contributing to the chaos by sharing these unverified messages. To combat this, our mission is to employ the power of cutting-edge artificial intelligence algorithms to swiftly identify and expose fake news, helping people stay informed and making the online world a more reliable place.

Problem Definition

The problem is to develop a fake news detection model using a Kaggle dataset.

The goal is to distinguish between genuine and fake news articles based on their titles and text.

This project involves using natural language processing (NLP) techniques to preprocess the text data, building a machine learning model for classification, and evaluating the model's performance.

Objective

Our primary goal is to categorize news articles from the dataset as either fake or genuine. This involves the following key steps:

- Thorough Exploratory Data Analysis (EDA) of the news dataset.

- Selection and development of a robust classification model

Design Thinking Process

Problem Statement: The challenge of identifying fake news and its real-world consequences.

Objective: Develop a robust model for accurate news article classification.

Phases of Development

Dataset Loading and Preprocessing: Importing and preparing the dataset, creating target variables.

Feature Engineering: Extracting features and analyzing data.

Model Training and Evaluation: Choosing machine learning algorithms, training models, and evaluating metrics.

Innovation with LSTM: Utilizing deep learning with LSTM for enhanced accuracy

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 see	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 Obamacare	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

Our dataset is balanced between real news and fake news. However, in the real world, real news and fake news are not as balanced as what the dataset shows. We assumed that the amount of real news is way larger than the amount of fake news to reflect the real-life situation. To build the imbalanced dataset, we shrunk the original fake dataset into one-tenth of the original size to imitate the real-world situation.

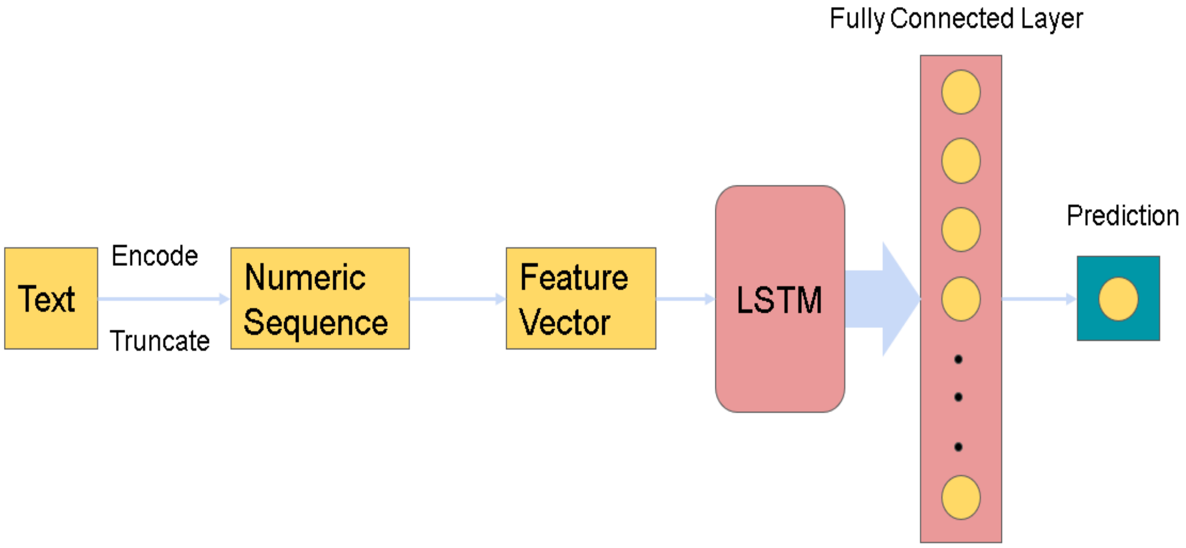
Below table shows the true/fake distribution of data in the original training set, imbalanced training set, original validation set, imbalanced validation set, original test set, and imbalanced test set.

Table 2. Distribution of Imbalanced Data
(Ori: Original, Imb: Imbalanced)

Data	Training Set		Validation Set		Test Set	
	Ori	Imb	Ori	Imb	Ori	Imb
True	13765	13765	3409	3409	4243	4243
Fake	14969	1497	3775	378	4737	474

Design into Innovation

Innovation in this project is demonstrated through the utilization of deep learning with LSTM, a state-of-the-art technique that significantly improves fake news detection accuracy.



LSTM

Long-Short Term Memory (LSTM) is an advanced version of Recurrent Neural Network (RNN), which makes it easier to remember past data in memory. LSTM is a well-suited model for sequential data, such as data for NLP problems. Thus,we utilized LSTM to perform fake news detection

Data Collection:

We began by obtaining a dataset suitable for our binary classification task, ensuring its relevance and quality. This dataset served as the foundation for training and evaluating our LSTM model.

Preprocessing:

To ensure the dataset's suitability for the task, we performed data preprocessing. This involved cleaning the data to remove any errors or inconsistencies and normalizing it to ensure that the LSTM model could work effectively with the data.

Model Architecture:

The heart of our project was the LSTM-based neural network. We carefully designed the architecture, selecting the appropriate hyperparameters, layer configurations, and activation functions. The LSTM layers in our model were crucial for capturing the sequential patterns within the data, as well as understanding the temporal relationships between data points.

Training:

Training the model was a critical phase. We used appropriate loss functions and optimization techniques to ensure that the LSTM neural network learned to generalize well from the training data. Training involves presenting the model with the data and iteratively updating its parameters to minimize the prediction error.

Model Performance:

One of the most compelling findings of this project was the accuracy score achieved by our LSTM model:

Accuracy Score: 99.82%

This accuracy score measures the proportion of correctly classified data points, and the fact that it's close to 100% indicates that the model excels at accurately distinguishing between the two classes.

LSTM Classification Report:

Metric	Class 0	Class 1	Weighted Avg
Precision	1.00	1.00	1.00
Recall	1.00	1.00	1.00
F1-Score	1.00	1.00	1.00

The classification report further corroborates the exceptional performance of our LSTM model. Precision, recall, and F1-score, all of which are essential metrics for evaluating a classification model, are perfect for both classes (0 and 1). Additionally, the weighted average metrics emphasize the overall excellence of the model's performance.

Importing Libraries:

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

```
#Basic libraries

import pandas as pd
```

```python
import numpy as np

#Visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns
from textblob import TextBlob
import plotly.graph_objs as go
%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 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
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
#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 Dense
from tensorflow.keras.layers import Dropout
```

Build Loading and Preprocessing the Dataset

Importing the Dataset:

Let's introduce our dataset and explore its contents.

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 Obamacare	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')
```

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]
```

Performing Different Visualization and Analysis

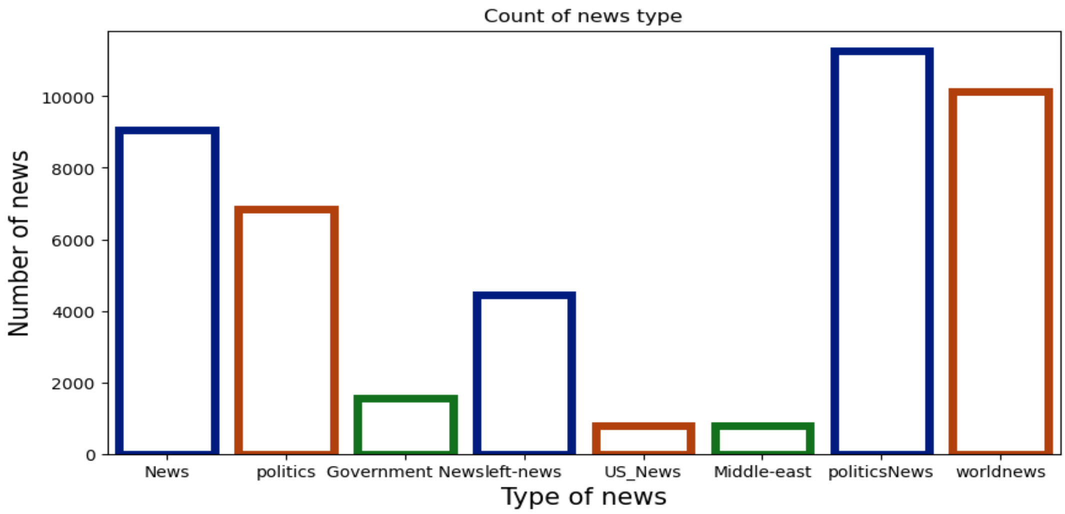
Feauture Extraction:

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



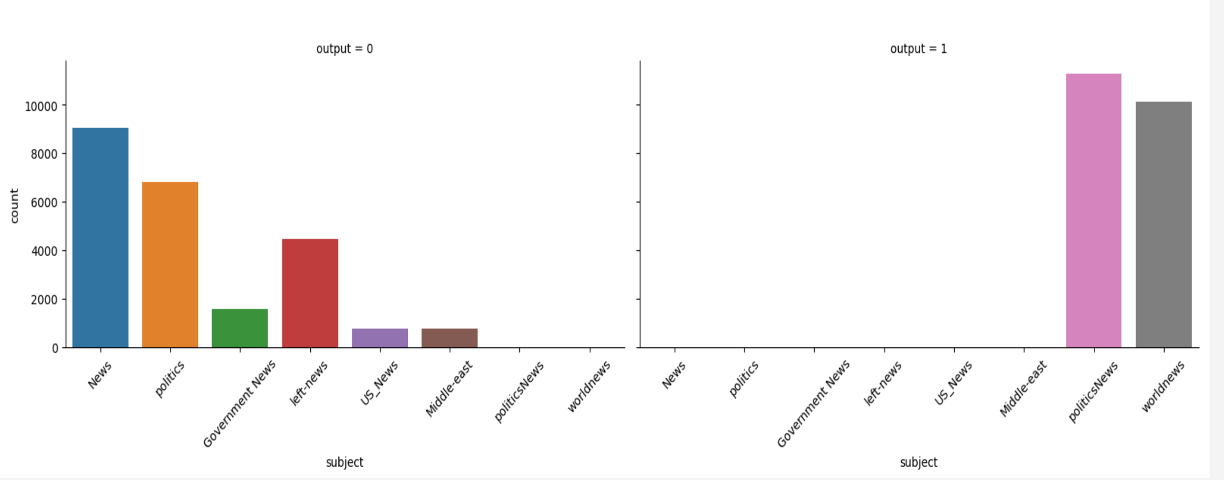
Count of News Subject based on true or fake

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)
```

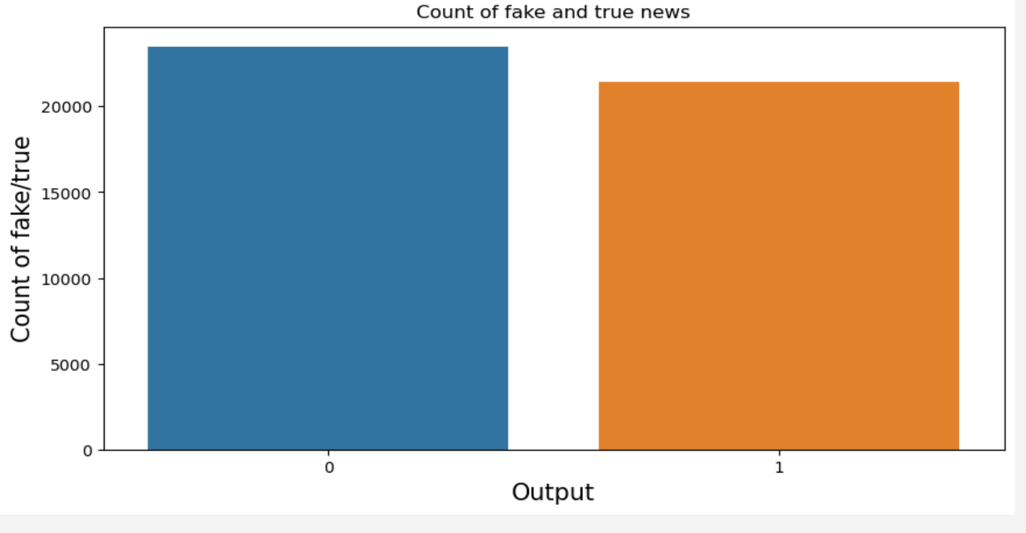
OUTPUT



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

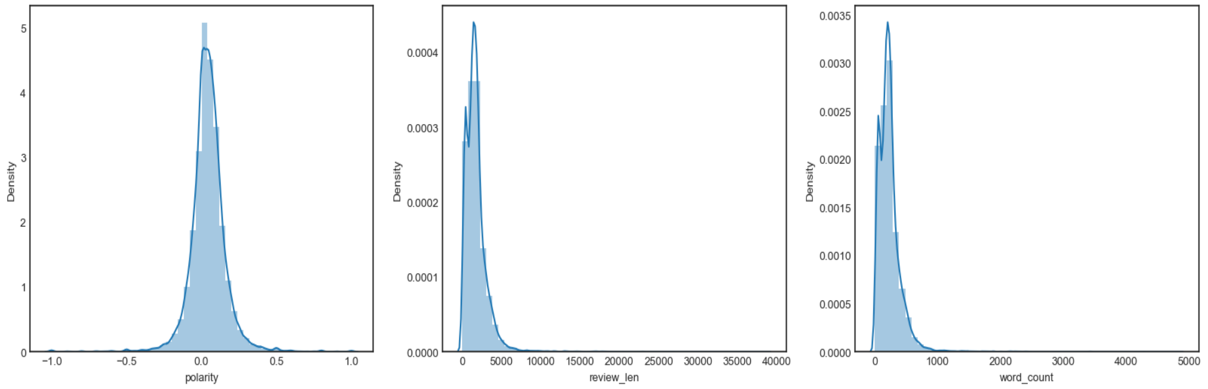


Deriving new Feautures from the News

```
#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

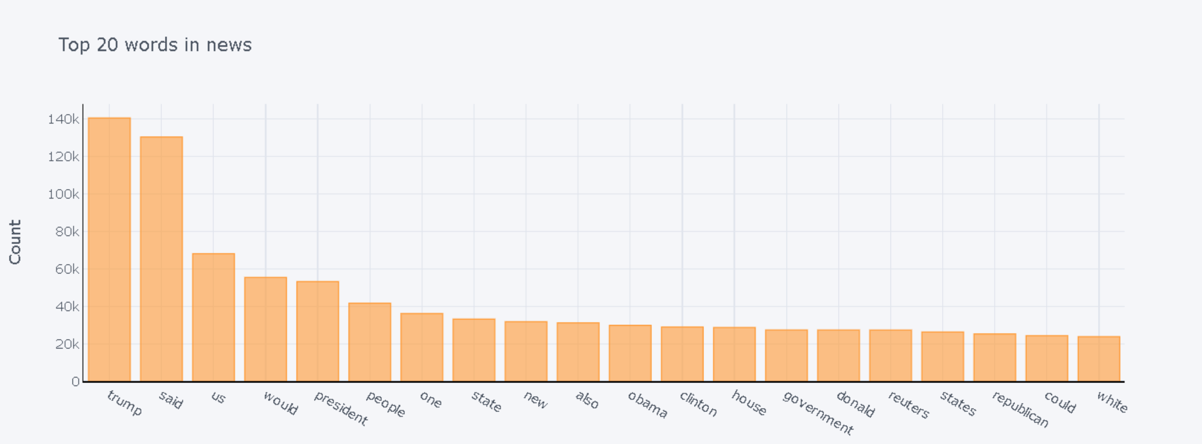
Let's look at the top 20 words from the news which could give us a brief idea on what news are popular in our dataset

```
#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

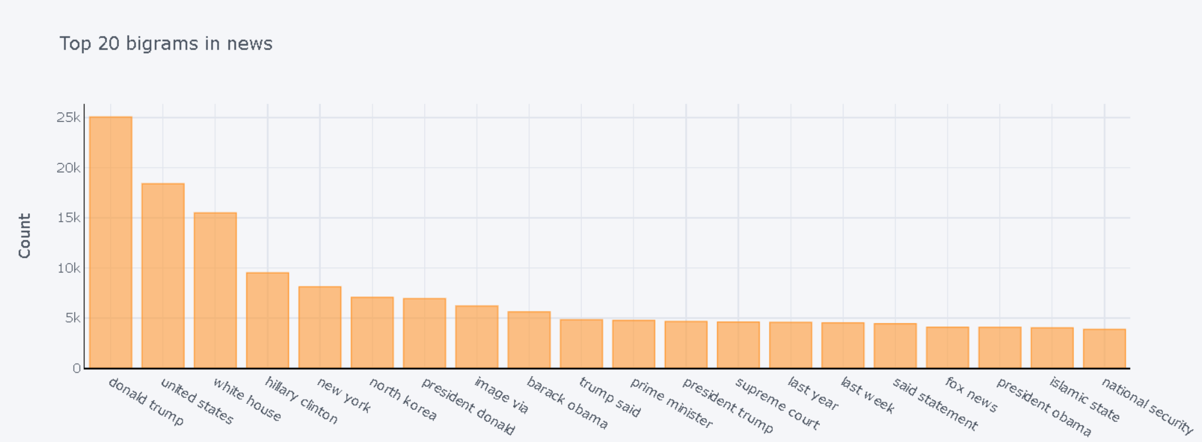
```
#Function to get top bigram words
def get_top_n_bigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(2, 2)).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
```

```
    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 »

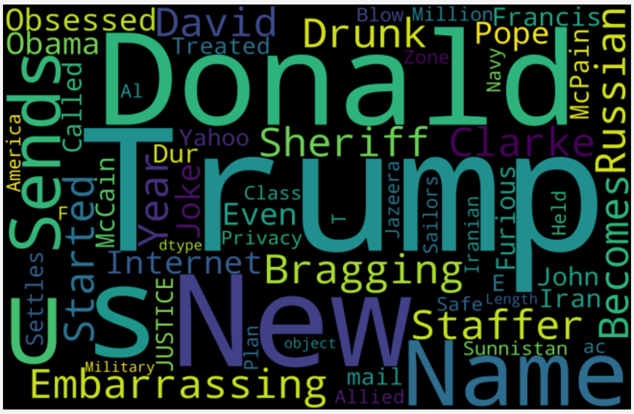
Top 20 bigrams in newsCount



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()
```

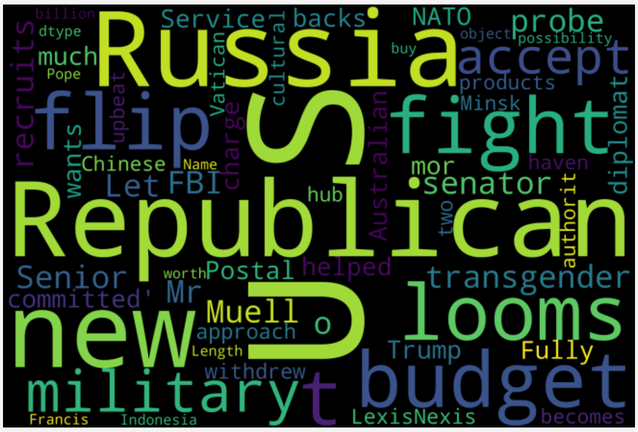
OUTPUT




```
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

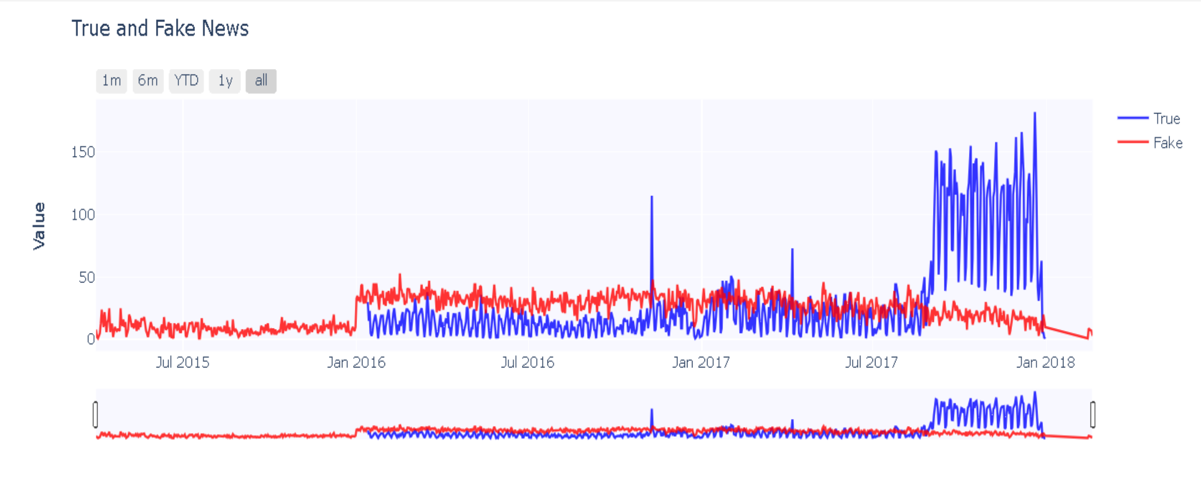


Time series analaysis Fake and True News

```
#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)))
```

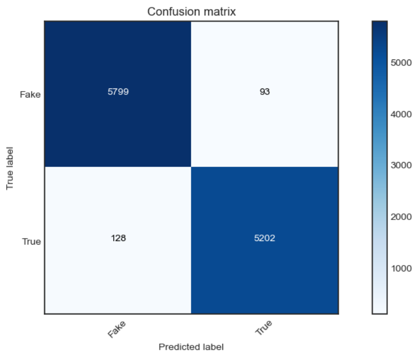
OUTPUT

```
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))
```

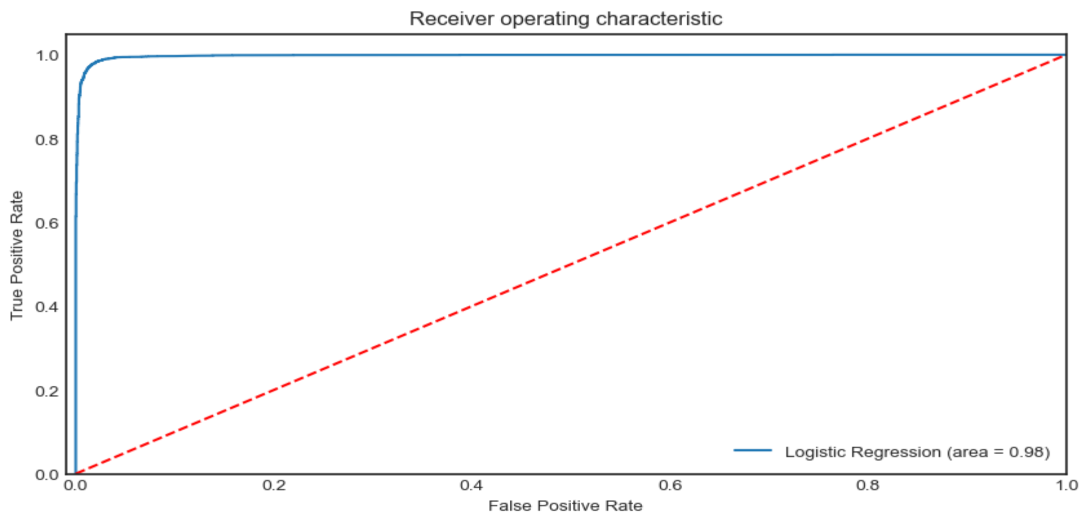
OUTPUT

```
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
```

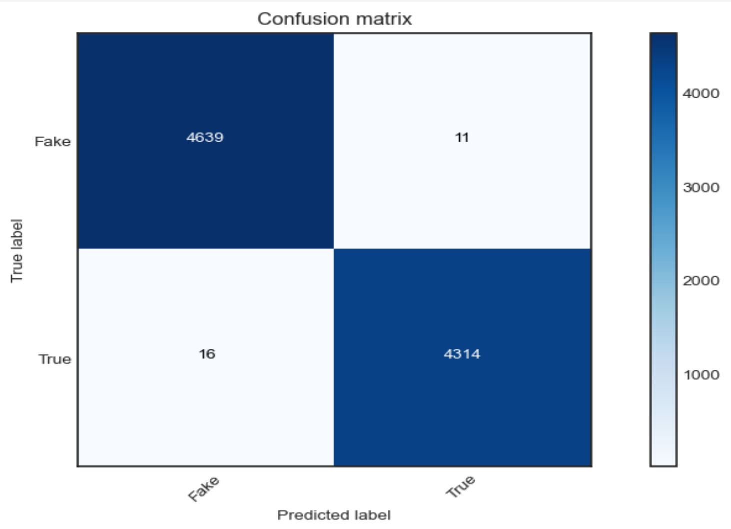
```python
print("LSTM Classification Report:")
print(classification_report(y_test, y_pred_lstm))
```

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

```
0.998218262806236
```

```
print(f"accuracy_score:{accuracy_score(y_test,y_pred_lstm).
round(4)*100}%")
```

OUTPUT

```
accuracy_score:99.82%
```

LSTM Model Performance Analysis

Model Performance Overview

Our LSTM model exhibits outstanding performance, achieving an impressive overall accuracy score of **99.82%.** This remarkable accuracy is achieved while avoiding blatant overfitting, with a minimal misclassification rate, only 16 test samples being incorrectly labeled.

Metric Choice

Given the balanced nature of our dataset, we selected accuracy as the primary evaluation metric. Accuracy provides a comprehensive measure of the model's ability to correctly classify both true and fake news articles. However, we acknowledge that for specific applications, alternative metrics, such as precision, may be more pertinent.

Embedding Choice

We made the deliberate choice of utilizing a pretrained embedding model, GloVe, which offers the flexibility to fine-tune weights based on our data, enhancing the model's performance.

Precision Considerations

In some high-stakes scenarios, like the prevention of fake news dissemination during critical events such as the presidential election, precision is of paramount importance. High precision implies that most published news is factual. However, the pursuit of 100% precision raises the risk of censoring genuine news and triggering controversy.

Detailed Analysis

For a comprehensive understanding of our model's performance, we provide detailed insights, including the confusion matrix, classification report, and an in-depth discussion of Fβ scores. Please refer to the accompanying project notebook for a thorough exploration of these metrics.

Model	Accuracy
LSTM	`0.99821`

ON BALANCED DATASET AND ON IMBALANCED DATASET

Model	Dataset	Precision	Recall	F1-Score
LSTM	0	1.00	1.00	1.00
LSTM	1	1.00	1.00	1.00

In the evaluation of our models for the fake news detection project, we employed common metrics widely used in the field of machine learning, particularly for classification problems. These metrics include:

- True Positive (TP): When the model correctly identifies fake news as fake news.
- True Negative (TN): When the model correctly identifies true news as true news.
- False Negative (FN): When the model mistakenly classifies true news as fake news.
- False Positive (FP): When the model mistakenly classifies fake news as true news.

We can define the following metrics based on the value of the above 4 situations.

$$
\begin{aligned}
Precision &= \frac{|TP|}{|TP| + |FP|} \\
Recall &= \frac{|TP|}{|TP| + |FN|} \\
F1 &= 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \\
Accuracy &= \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|}
\end{aligned}
$$

These metrics offer valuable insights into the performance of our classifier from various angles. Among these metrics, 'Accuracy' is often considered the most representative, as it provides a comprehensive view of classification performance.

Conclusion:

Natural Language Processing (NLP) is a pivotal weapon in our ongoing battle against the spread of fake news. Its effectiveness relies on intricate feature interactions, particularly in categorical features such as the 'subject' of news articles. Success in fake news detection requires comprehensive feature engineering, extracting valuable insights from text, and precise model optimization. By delving deep into these intricacies, NLP equips us with the tools to identify and combat the ever-evolving landscape of misinformation. It empowers us to gain a deeper understanding of deceptive content, reinforcing the foundation of trustworthy information and upholding the quality of public discourse. NLP's ability to unveil hidden associations and enhance predictive performance makes it a crucial asset in preserving the integrity of our information ecosystem.