FAKE NEWS DETECTION USING NLP

Introduction:

News media plays a crucial role in delivering global information to the public. People often trust news reports to be completely accurate. However, there have been cases where news outlets admitted to making mistakes. Some news stories can greatly impact not only individuals and governments but also economies. A single news report can influence economic trends based on public emotions and the political environment. Therefore, it's vital to distinguish between real news and misinformation. Natural Language Processing tools have been used to solve this problem by analyzing historical data and differentiating between real and fake news. In simple terms, we now have tools to detect fake news!

Problem Statement:

The issue of trustworthiness in information has been a long-standing problem affecting both businesses and society, including both traditional and online media. On social media, information can spread rapidly and widely. False or misleading information can have real-world consequences within minutes, affecting millions of users. Recent discussions have raised public concerns about this problem and suggested solutions to reduce it.

News outlets have a history of using sensational headlines to capture the audience's attention and sell their news. On social media platforms, the rapid spread of information makes it even easier for false or inaccurate news to have significant real-world impacts.

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.

Importing Libraries:

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

```
#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
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.

```
#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")

# print shape of true dataset with rows and columns and information
print ("The shape of the  data is (row, column):"+ str(true_news.shape))
print (true_news.info())
```

The fake news dataset comprises 23,481 entries, while the true news dataset contains 21,417 entries.

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.

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

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

Concatenating Title and Text of News

News must be classified based on both the title and the news content. Treating the title and content separately does not yield significant benefits. Therefore, we concatenate both columns in both datasets.

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

Converting the Date Columns to Datetime Format

We can utilize `pd.to_datetime` to convert our date columns into the desired date format. However, there is an issue with the "date" column in the fake news dataset, which contains

```
#Converting the date to datetime format
fake_news['date'] = pd.to_datetime(fake_news['date'])
true_news['date'] = pd.to_datetime(true_news['date'])
```

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
```

Text Processing

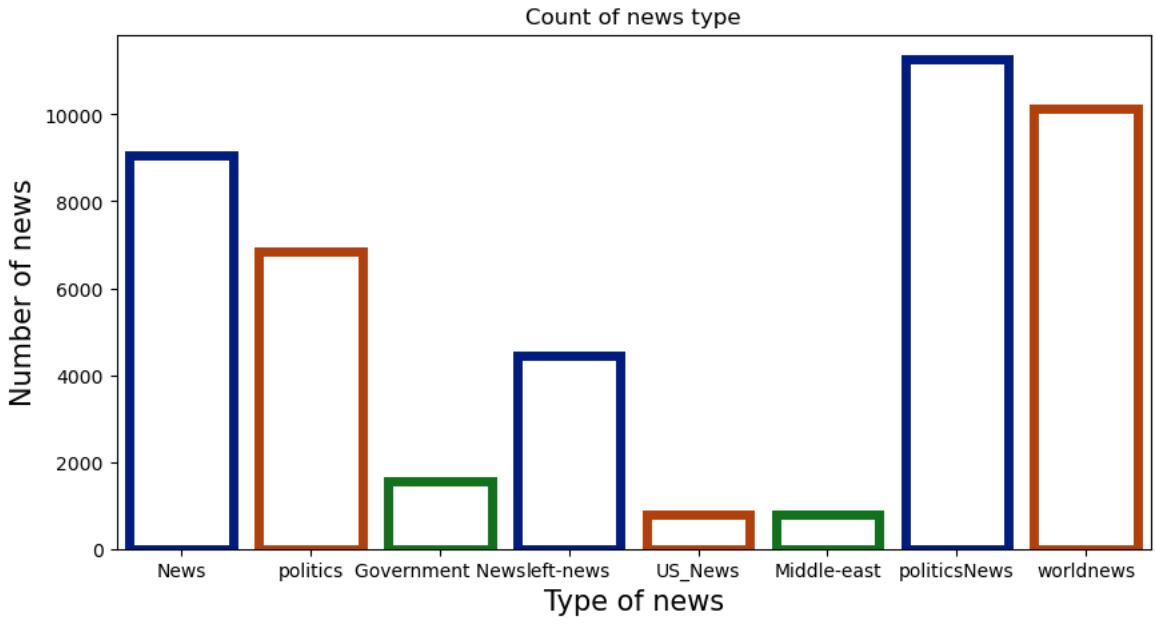
```
#Creating a copy
clean_news=news_dataset.copy()
```

```
def review_cleaning(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text

clean_news['news']=clean_news['news'].apply(lambda x:review_cleaning(x))
clean_news.head()
```

Data Visualization for News:

Count of News Subjects

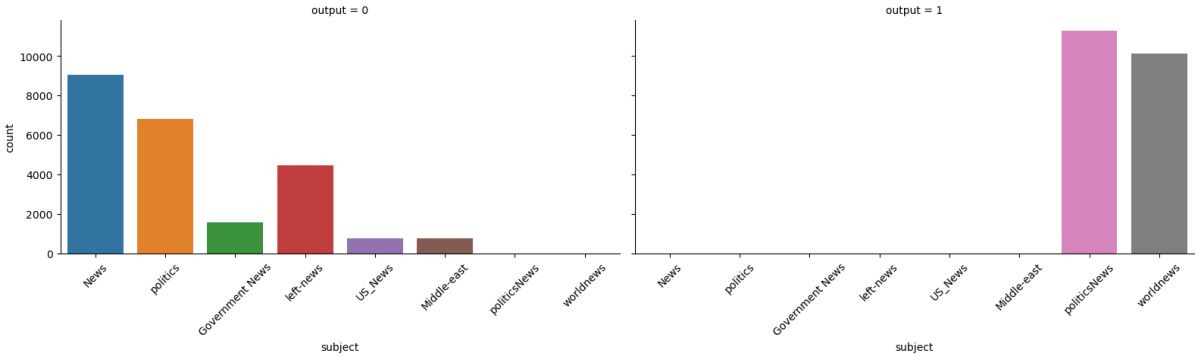
```
#Plotting the frequency plot
ax = sns.countplot(x="subject", data=clean_news,
                   facecolor=(0, 0, 0, 0),
                   linewidth=5,
                   edgecolor=sns.color_palette("dark", 3))

#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)
```



Count of News Subject based on true or fake

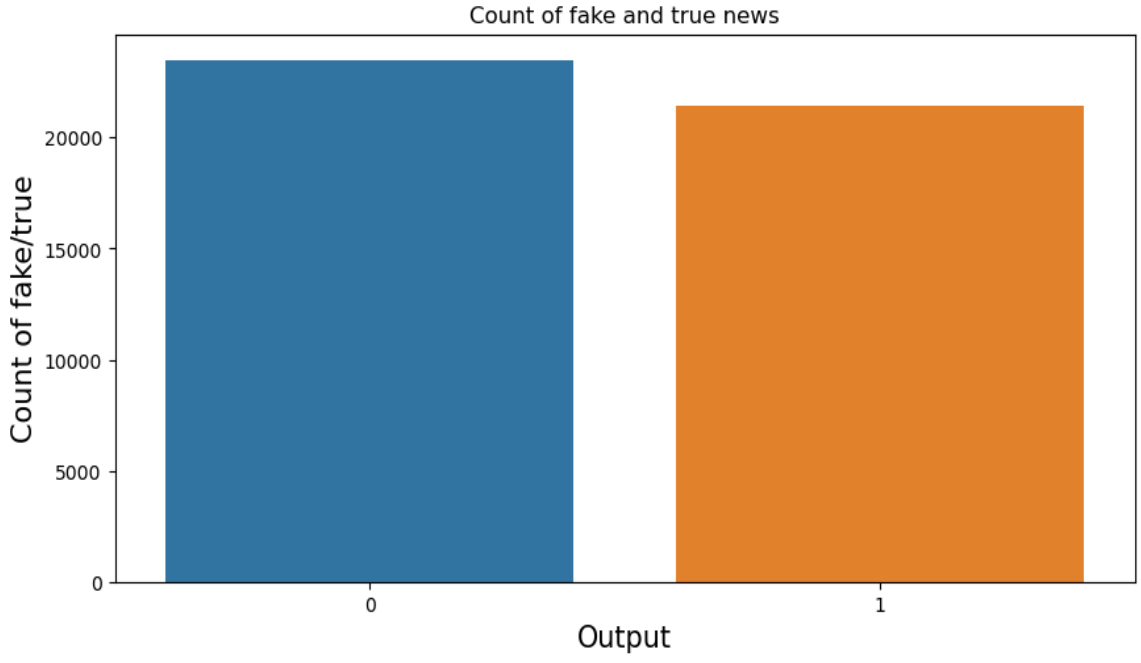
Let's examine the distribution of fake and true news to confirm whether our data is balanced.

```
g = sns.catplot(x="subject", col="output",
                data=clean_news, kind="count",
                height=4, aspect=2)

#Rotating the xlabels
g.set_xticklabels(rotation=45)
```



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)
```



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):
    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

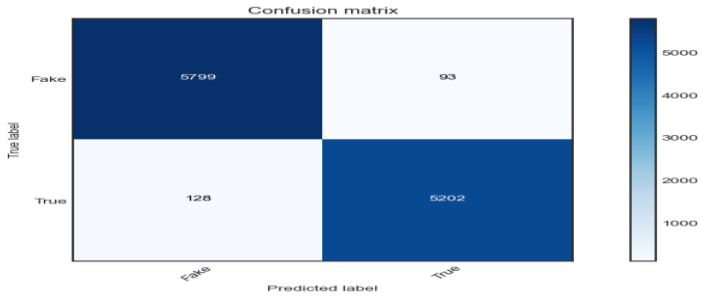
Logistic Regression with hyperparameter Tuning

```
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



LSTM (deep learning approach)

```
# 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)

# 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:

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

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.