FAKE NEWS ANALYSIS IN SOCIAL MEDIA

Using IBM Watson Machine Learning

Developed by : Karthika Ramachandran , Mariya Paul

SmartBridge – MiniProject Report

1. INTRODUCTION

1.1 Overview

We are in the period of data, each time we read a snippet of data or watch the news on TV, we search for a solid source. There is so much phony news spread all over the web and web-based entertainment. Counterfeit News is deception or controlled word that is gotten out across social media to harm an individual, office, or association. The spread of deception in basic circumstances can cause catastrophes. Because of the spread of phony news, there is a need for computational strategies to identify them. Thus, to forestall them is chief that should possibly use innovation, we have executed Machine Learning calculations and strategies like NLTK, and LSTM. Our commitment is bi fold. In the first place, we should presentthe datasets which contain both phony and genuine news and direct different analyses to coordinate phony news finder. We came by improved results contrasted with the existing frameworks.

1.2 Purpose

A great deal of fake news is roaring through the various social media platforms. During this case classification of any news, post, story, journal in to fake or real one has become a crucial them as fake and true and it's con jointly attracted a good interest from researchers round the world In line with several analysis studies that are administered to hunt out the impact of any false and fictional news on of us up on returning through such fakenews details. Falsified news or news is used in such how that individual begin basic mental process in one issue that may not true.

2. LITERATURE SURVEY

2.1Existing problem

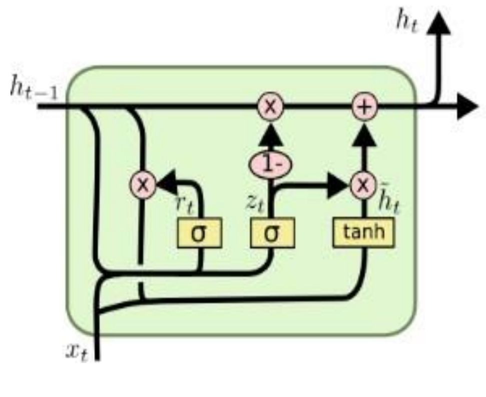
Distinguishing counterfeit news is accepted to be a mind-boggling task what's more, a lot harder than identifying counterfeit item surveys. With the open idea of the we band virtual entertainment, not with standing the new high-level pay advancements improve on the method involved with making and getting out the counterfeit words. While it's more obvious also , follow the aim and the effect of phony surveys, the goal and the effect of making promulgation by getting out counterfeit words cannot be estimated or seen without any problem.

For example, counterfeit survey influences the item proprietor, clients, and online stores; on the another hand, it isn't difficult to recognize the elements that impacted by the phony news.

This is because recognizing these substances requires estimating the news spread, which has demonstrated to be complex and assete scalated.

2.2Proposed Solution

Long short-term memory (LSTM) units are building blocks for the layers of a recurrent neural network (RNN). ALSTM unit is made out of a cell, an information door a result entry way, and a neglected door [12]. The phone is liable for "recollecting" values throughout a huge times pans of the connection of the word at the beginning of the message can impact the result of the word later in the sentence. Conventional brain networks can't recollect or keep the record of what all is passed before they are executed this stops the ideal impact of words that come in the sentence prior to having any effect on the closure words, and it appears to be a significant weakness**.**



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

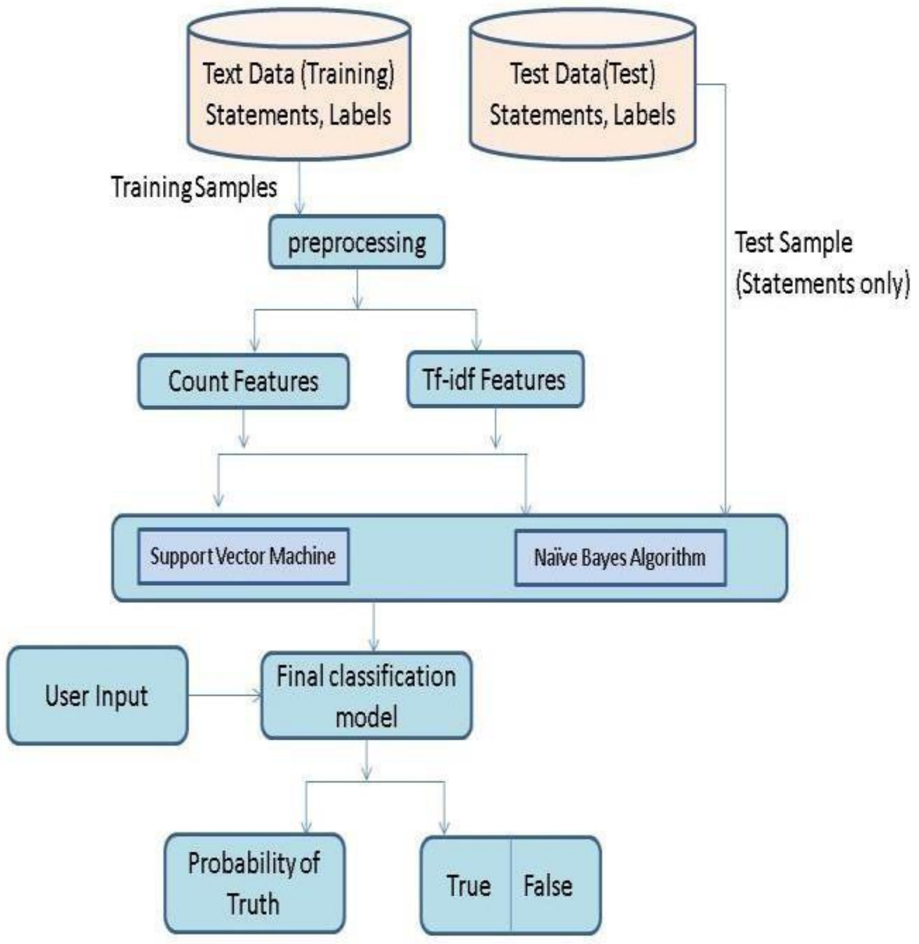
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

3. THEORITICAL ANALYSIS:

3.1 Block Diagram



Project Work Flow:

1. Install required packages and libraries
2. Understanding the data.
3. Model Building
4. Application Building
5. Final UI

3.2 HARDWARE AND SOFTWARE REQUIREMENTS IN THE PROJECT:

For running a machine learning model on the system you need a system with minimum of 16 GBRAM in it and you require a good processor for high performance of the model**.**

In the list of **Software requirements** you must have:

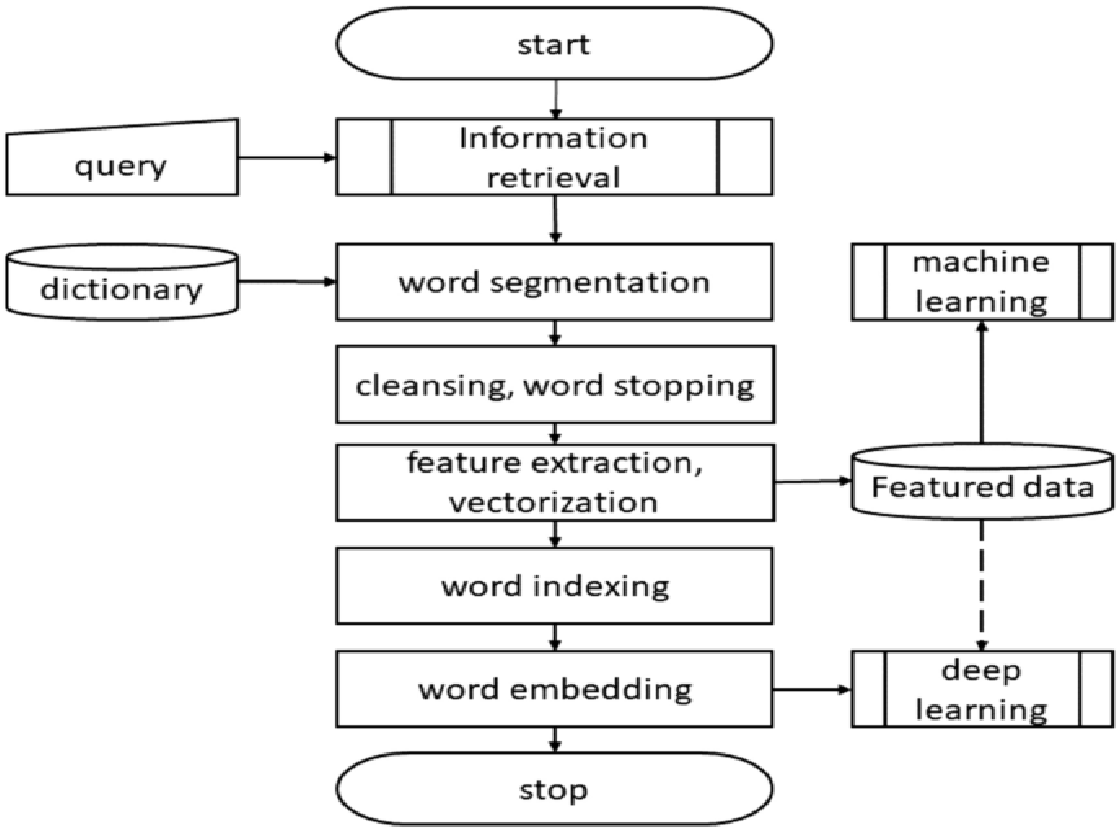
- Jupyter Notebook for programming, which can be installed by Anaconda IDE.
- Python packages.
- A better software for running the html and css files for application building phase e.g. spyder.

4.EXPERIMENTALI NVESTIGATIONS:

4.1 Data Preprocessing:-

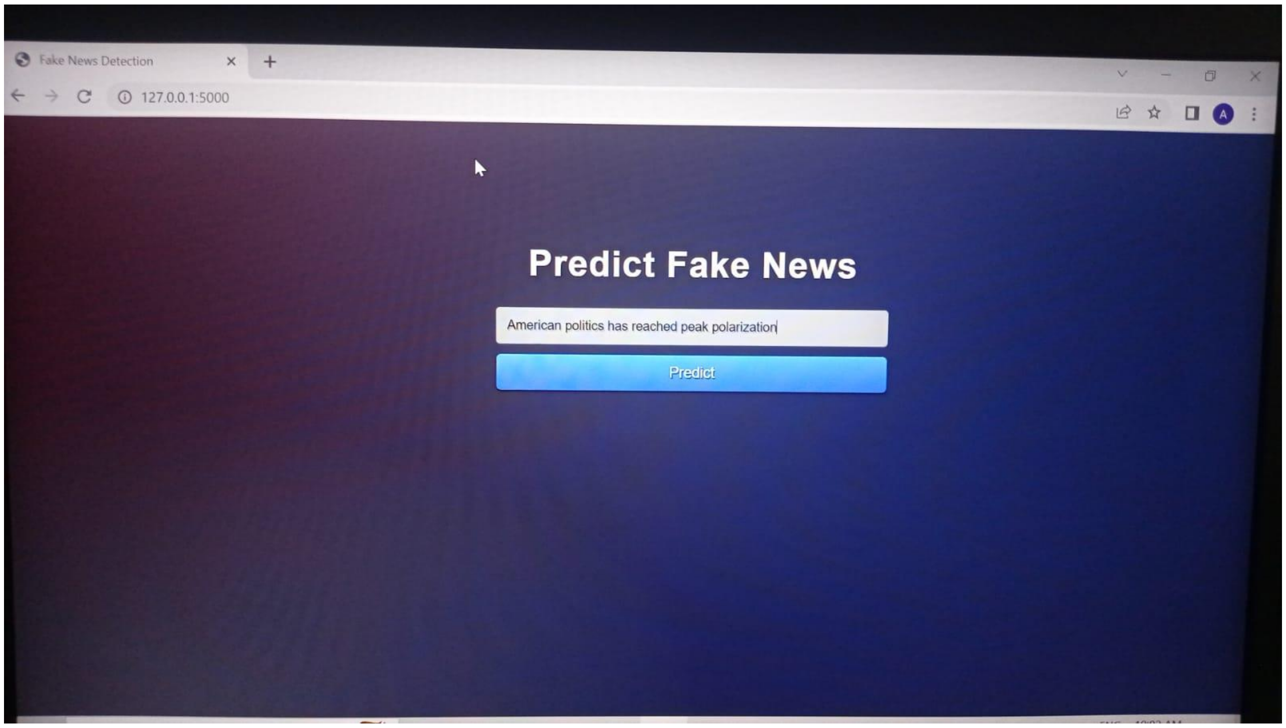
(i) Data Integration
(ii) Data Cleaning
(iii) Data Transformation
(iv) Data Reduction

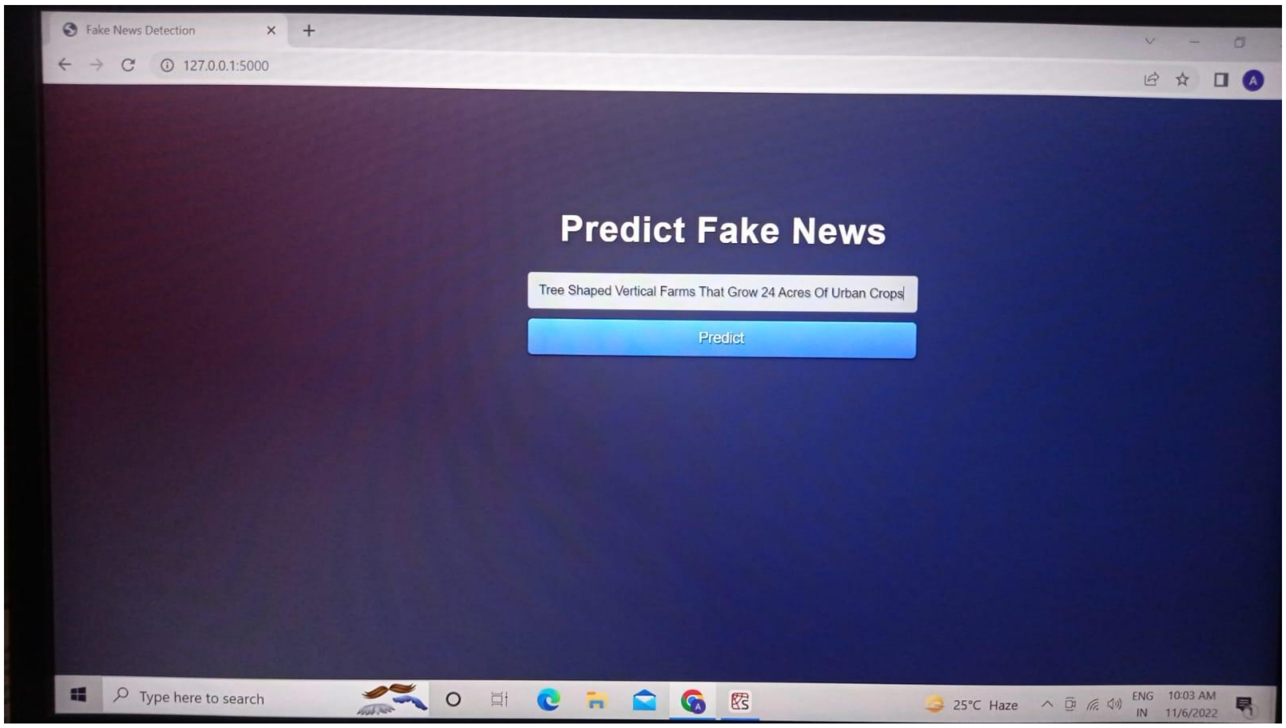
5.FLOWCHART:-


start
query
Information retrieval
dictionary
word segmentation
machine learning
cleansing, word stopping
feature extraction, vectorization
Featured data
word indexing
word embedding
deep learning
stop

6.RESULTS:-

Final output of the project:





7. ADVANTAGES

1. Advertisers take the Advantages of Fake News
2. Influencers also take benefits of Fake News
3. Political Warfare
4. Fun and Entertainment

DISADVANTAGES

1. Change in Public Opinion
2. Defamation is among the disadvantages of fake news
3. False Perception
4. Fake News may lead to Social Unrest

8.APPLICATIONS:-

Develop a machine learning program to identify when a news source may be producing fake news. We aim to use a corpus of labeled real and fake new articles to build a classifier that can make decisions about information based on the content from the corpus. The model will focus on identifying fake news sources, based on multiple articles originating from a source. Once a source is labeled as a producer of fake news, we can predict with high confidence that any future articles from that source will also be fake news. Focusing on sources widens our article misclassification tolerance, because we will have multiple data points coming from each source. An application will be built based on this machine learning program in order to increase accessibility to detection of fake news.

9.CONCLUSION:-

The majority of activities in the twenty first century are done online. Newspapers, which were once preferred in hard copies, are now being replaced by applications such as Facebook ,Twitter, and news articles. The growing issue of fake news further complicates matters by attempting to sway people's opinions and attitudes about the use of digital technology. When a person is misled by real news, one of two things can happen : people begin to believe that their assumptions about a specific topic are right. To fix this issue, we built our Fake News Detection system, which takes user feedback and classifies it as true or false. To do so, various NLP and Machine Learning algorithms must be used, as well as a suitable dataset for training the model. We also used the TF- IDF vectorizer to vectorize the text data. Various performance metrics are often used to evaluate performance.

In the aforementioned research summary and system analysis, we concluded that most of the researchpapers used naïve bays algorithm, and the prediction precision was between 70-76%, they mostly usequalitative analysis depending on sentiment analysis, titles, word frequency repetition [40][41][42]. Inour approach we propose to add to these methodologies, another aspect, which is POS textual analysis , it is a quantitative approach, it depends on adding numeric statistical values as features, we thought that increasing these features and using random forest will give further improvements to precession results.The features we propose to add in our dataset are total words (tokens), Total unique words (types), Type/Token Ratio(TTR), Number of sentences, Average sentence length(ASL), Number of characters, Average word length(AWL), nouns, prepositions, adjectives etc.

10.FUTURE SCOPE:-

Spreading of fake news always deliver a bad and negativeimpact to a society. Is still lots and lots of a confusion in asociety, when it comes to differentiating between fake and true news. Fake news really is a false alarm to any personasit always just misleads the readers, and the personal ways ends up being confused and not acting in the rightway. Their daily life with their naked eyes. So, this is whenour project can use certainly to predicts whether projectthe given news is fake or not? By considering our project'sideology people can at least be able to check whether thenews they have gotin the frontof their eyesare legitornot and the people will become more aware of the fakenews circulation. This system has been completed in this final year which certainly needs more improvements in the near future by using a flask.

11.BIBILOGRAPHY:-

[1]. Parikh , S. B., & Atrey, P. K. (2018, April). Media-RichFake News Detection: A Survey. In 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR) (pp.436-441).IEEE.

[2]. Conroy, N. J., Rubin, V. L., & Chen, Y. (2015, November).Automatic deception detection: Methods for finding fakenews. In Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for theCommunity (p. 82). American Society for InformationScience..

APPENDIX:-

app.py

```
#Importing the Libraries


#flask is use for run the web application.
import flask
#request is use for accessing file which was uploaded by the user on our application.
from flask import Flask, request,render_template
#from flask_cors import CORS

#Python pickle module is used for serializing
# and de-serializing a Python object structure.
import pickle

#OS module in python provides functions for interacting with the operating system
import os

#Newspaper is used for extracting and parsing newspaper articles.
#For extracting all the useful text from a website.
#from newspaper import Article

#URLlib is use for the urlopen function and is able to fetch URLs.
#This module helps to define functions and classes to open URLs
import urllib

#Loading Flask and assigning the model variable
app = Flask(_name_)
#CORS(app)
app=flask.Flask(_name_,template_folder='templates')

with open(r'C:/Users/HLC/Downloads/FAKE NEWS  ANALYSIS /Flask /fakenews.
pkl.pkl', 'rb')  as handle:
 model = pickle.load(handle)

@app.route('/') #default route
def main():
return render_template('main.html')

#Receiving the input url from the user and using Web Scrapping to extract the news
```

```
content

#Route for prediction
@app.route('/predict',methods=['GET','POST'])

def predict():
      #Contains the incoming request data as string in case.
  url =request.get_data(as_text=True)[5:]

       #The URL parsing functions focus on splitting a URL string into its components,
     #or on combining URL components into a URL string.
 url = urllib.parse.unquote(url)

    #A new article come from Url and convert onto string
  article = Article(str(url))

    #To download the article
 article.download()

    #To parse the article
  article.parse()

    #To perform natural language processing ie..nlp
article.nlp()
   #To extract summary
news = article.summary
print(type(news))

 #Passing the news article to the model and returing whether it is Fake or Real
 pred = model.predict([news])
print(pred)
  return render_template('main.html', prediction_text='The news is "{}"'.format(pred[0]))

if _name=="main_":
app.run(debug=False,threaded = False)
```

CSS

```
@importurl(https://fonts.googleapis.com/css?family=Open+Sans);
.btn { display: inline-block; *display: inline; *zoom: 1; padding: 4px 10px 4px; margin
bottom: 0; font-size: 13px; line-height: 18px; color: #333333; text-align: center;text-
shadow : 0 1px 1pxrgba(255, 255, 255, 0.75); vertical-align: middle; background-color:
#f5f5f5; background-image: -moz-linear-gradient(top, #ffffff, #e6e6e6); background
image: -ms-linear-gradient(top, #ffffff, #e6e6e6); background-image: -webkit-
gradient(linear, 0 0, 0 100%, from(#ffffff), to(#e6e6e6)); background-image: -webkit-
linear-gradient(top, #ffffff, #e6e6e6); background-image: -o-linear-gradient(top, #ffffff,
#e6e6e6);background-image: linear-gradient(top, #ffffff, #e6e6e6); background-repeat:
repeat-x; filter: p…
```

Html

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
<meta charset="UTF-8">
<title>Fake News Detection</title>
<link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>
<div class="login">
```

```
<h1>Predict Fake News</h1>

<!-- Main Input For Receiving Query to our ML -->
<form action="{{ url_for('predict')}}"method="post">
  <input type="text" name="news" placeholder="Enter the news url" required="required"
                                                                    />
<button type="submit" class="btnbtn-primary btn-block btn-large">Predict</button>

</form>

<br>
<br>
{{ prediction_text }}

</div>
</body>
</html>
```

.ipynb

```
#Importing libraries

import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix
import pickle
from sklearn.pipeline import Pipeline

df = pd.read_csv('C:/Users/HLC/Downloads/FAKE NEWS ANALYSIS/dataset/DATASET.csv')
df.head()

# Create a series to store the labels: y
# y = df.label

#cleaned file containing the text and label
X = df['text']  # independent variable
y = df['label'] #dependent variable
```

```
# Create training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.33, random_state=53)

# Initialize a CountVectorizer object: count_vectorizer
count_vectorizer = CountVectorizer(stop_words='english')

# Transform the training data using only the 'text' column values: count_train
count_train = count_vectorizer.fit_transform(X_train)

# Transform the test data using only the 'text' column values: count_test
count_test = count_vectorizer.transform(X_test)

# Print the first 10 features of the count_vectorizer
print(count_vectorizer.get_feature_names()[:10])

# Initialize a TfidfVectorizer object: tfidf_vectorizer
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)

# Transform the training data: tfidf_train
tfidf_train = tfidf_vectorizer.fit_transform(X_train)

# transform the test data: tfidf_test
tfidf_test = tfidf_vectorizer.transform(X_test)

# Print the first 10 features
print(tfidf_vectorizer.get_feature_names()[:10])

# Print the first 5 vectors of the tfidf training data
print(tfidf_train.A[:5])

count_df = pd.DataFrame(count_train.A, columns=count_vectorizer.get_feature_names())

# Create the TfidfVectorizer DataFrame: tfidf_df
tfidf_df = pd.DataFrame(tfidf_train.A, columns=tfidf_vectorizer.get_feature_names())

# Print the head of count_df
print(count_df.head())

# Print the head of tfidf_df
print(tfidf_df.head())
```

```
# Calculate the difference in columns: difference
difference = set(count_df.columns) - set(tfidf_df.columns)
print(difference)

# Check whether the DataFrame are equal
print(count_df.equals(tfidf_df))

# Instantiate a Multinomial Naive Bayes classifier: nb_classifier
nb_classifier = MultinomialNB()

# Fit the classifier to the training data
nb_classifier.fit(count_train, y_train)

# Create the predicted tags: pred
pred = nb_classifier.predict(count_test)

# Calculate the accuracy score: score
score = accuracy_score(y_test, pred)
print(score)

# Calculate the confusion matrix: cm
cm =confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
print(cm)

#Training and testing the "fake news" model with TfidfVectorizer
nb_classifier = MultinomialNB()

# Fit the classifier to the training data
nb_classifier.fit(tfidf_train, y_train)

# Create the predicted tags: pred
pred = nb_classifier.predict(tfidf_test)

# Calculate the accuracy score: score
score = accuracy_score(y_test, pred)
print(score)

# Calculate the confusion matrix: cm
cm = confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
print(cm)
```

```
#Improving the model to test a few different alpha levels using the Tfidf vectors,
# to determine.if there is a better performing combination

alphas = np.arange(0, 1, 0.1)

# Define train_and_predict()
def train_and_predict(alpha):
    # Instantiate the classifier: nb_classifier
    nb_classifier = MultinomialNB(alpha=alpha)

    # Fit to the training data
    nb_classifier.fit(tfidf_train, y_train)

    # Predict the labels: pred
    pred = nb_classifier.predict(tfidf_test)

    # Compute accuracy: score
    score = accuracy_score(y_test, pred)
    return score

# Iterate over the alphas and print the corresponding score
for alpha in alphas:
    print('Alpha: ', alpha)
    print('Score: ', train_and_predict(alpha))
    print()

class_labels = nb_classifier.classes_

# Extract the features: feature_names
feature_names = tfidf_vectorizer.get_feature_names()

# Zip the feature names together with the coefficient array
# and sort by weights: feat_with_weights
feat_with_weights = sorted(zip(nb_classifier.coef_[0], feature_names))

# Print the first class label and the top 20 feat_with_weights entries
print(class_labels[0], feat_with_weights[:20])

# Print the second class label and the bottom 20 feat_with_weights entries
print(class_labels[1], feat_with_weights[-20:])
```

```
#Creating a pipeline that first creates bag of words(after applying stopwords) & then applies
Multinomial Naive Bayes model
pipeline = Pipeline([('tfidf', TfidfVectorizer(stop_words='english')),
                    ('nbmodel', MultinomialNB())])

#Training our data
pipeline.fit(X_train, y_train)

#Predicting the label for the test data
pred = pipeline.predict(X_test)
print(confusion_matrix(y_test, pred))

#Serialising the file
#pickle.dump(nb_classifier,open('fake_news.pkl','wb'))
#Serialising the file
with open('model.pkl', 'wb') as handle:
    pickle.dump(pipeline, handle, protocol=pickle.HIGHEST_PROTOCOL)
```