**FAKE NEWS DETECTION USING NLP**

**PROBLEM STATEMENT :**

The proliferation of fake news in the digital age presents a critical challenge to the credibility of information and the functioning of democratic societies. This research proposes an innovative approach to combat this issue through the application of Artificial Intelligence (AI) and Natural Language Processing (NLP) techniques. The study leverages state-of-the-art NLP models, such as BERT and GPT, for fake news detection. A comprehensive dataset comprising a wide range of news articles, both real and fake, is used for training and evaluation.

The research focuses on feature extraction, sentiment analysis, and linguistic patterns to develop a robust fake news detection system. Machine learning algorithms, including supervised and unsupervised techniques, are employed to classify news articles into real or fake categories. Additionally, the research explores the use of deep learning methods to capture subtle linguistic cues that are indicative of deceptive content.

The performance of the AI-based detection system is rigorously evaluated using various metrics, including precision, recall, and F1-score, to ensure its reliability and effectiveness. The findings highlight the promising capabilities of AI and NLP in distinguishing between real and fake news with a high degree of accuracy.

The implications of this research extend to media organizations, social media platforms, and news consumers, offering a powerful tool to identify and combat the spread of disinformation. By addressing this critical issue, the study contributes to the broader goal of preserving the integrity of information in the digital age and upholding the principles of informed, responsible journalism.

**DESIGN THINKING PROCESS:**

****

Figure: Flow Diagram For Proposed System

**PROJECT PHASE DEVELOPMENT:**

What things you need to install the software and how to install them:

**Python 3.6**

This setup requires that your machine has python 3.6 installed on it. We can refer to this url https://www.python.org/downloads/ to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that checks this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/.

Setting up PATH variable is optional as we can also run program without it and more instruction are given below on this topic.

Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url https://www.anaconda.com/download/

We shall also need to download and install below 3 packages after you install either python or anaconda from the steps above

Sklearn (scikit-learn)

numpy

scipy

if you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages

pip install -U scikit-learn

pip install numpy

pip install scipy

if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages

conda install -c scikit-learn

conda install -c anaconda numpy

conda install -c anaconda scipy

**DATASET:**

**Dataset used**

The data source used for this project is LIAR dataset which contains 3 files with .tsv format for test, train and validation. Below is some description about the data files used for this project.

LIAR: A BENCHMARK DATASET FOR FAKE NEWS DETECTION

William Yang Wang, "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection, to appear in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017), short paper, Vancouver, BC, Canada, July 30-August 4, ACL.

The original dataset contained 13 variables/columns for train, test and validation sets as follows:

* Column 1: the ID of the statement ([ID].json).
* Column 2: the label. (Label class contains: True, Mostly-true, Half-true, Barely-true, FALSE, Pants-fire)
* Column 3: the statement.
* Column 4: the subject(s).
* Column 5: the speaker.
* Column 6: the speaker's job title.
* Column 7: the state info.
* Column 8: the party affiliation.
* Column 9-13: the total credit history count, including the current statement.
* 9: barely true counts.
* 10: false counts.
* 11: half true counts.
* 12: mostly true counts.
* 13: pants on fire counts.
* Column 14: the context (venue / location of the speech or statement).

To make things simple we have chosen only 2 variables from this original dataset for this classification. The other variables can be added later to add some more complexity and enhance the features.

Below are the columns used to create 3 datasets that have been in used in this project

* Column 1: Statement (News headline or text).
* Column 2: Label (Label class contains: True, False)

You will see that newly created dataset has only 2 classes as compared to 6 from original classes. Below is method used for reducing the number of classes:

* Original -- New
* True -- True
* Mostly-true -- True
* Half-true -- True
* Barely-true -- False
* False -- False
* Pants-fire -- False

The dataset used for this project were in csv format named train.csv, test.csv and valid.csv and can be found in repo. The original datasets are in "liar" folder in tsv format.

**Program Code for Loading and Preprocessing Dataset:**

import pandas as pd

import sklearn

import itertools

import numpy as np

import seaborn as sb

import re

import nltk

import pickle

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn import metrics

from sklearn.metrics import confusion\_matrix

from matplotlib import pyplot as plt

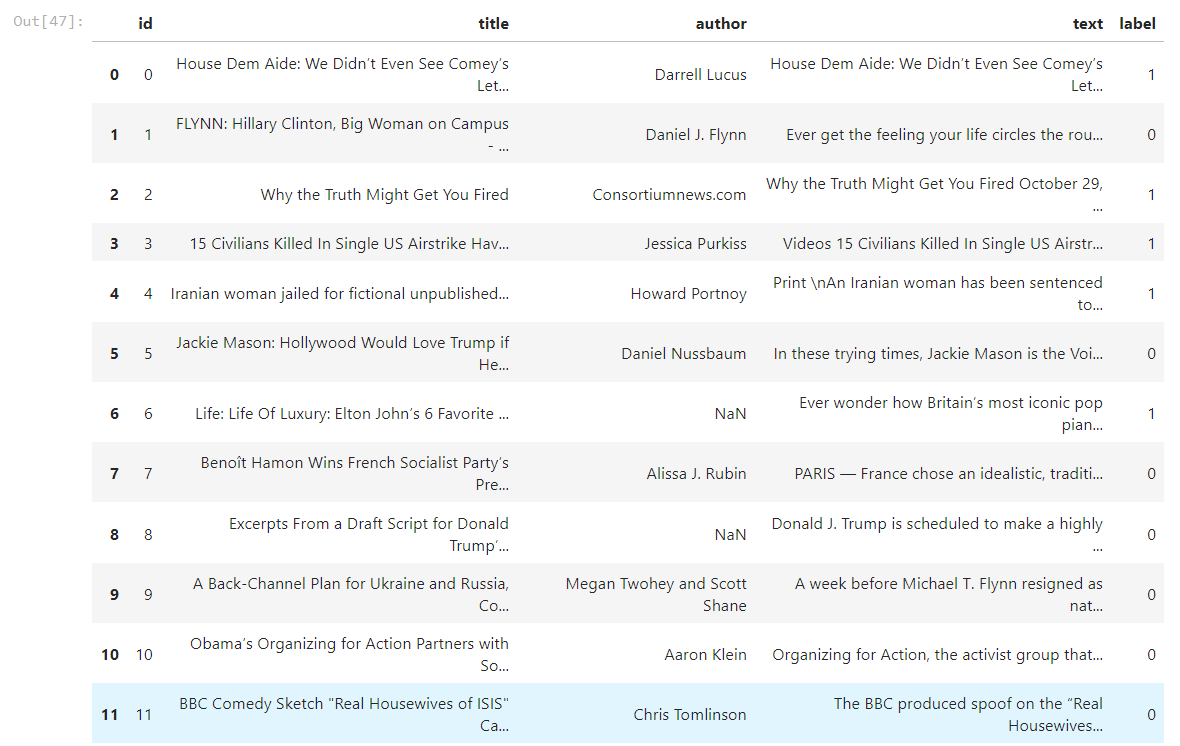
from sklearn.linear\_model import PassiveAggressiveClassifier

from nltk.stem import WordNetLemmatizer

from nltk.corpus import stopwords

train\_df = pd.read\_csv(r'C:\Users\Mayur\Downloads\train.csv')

train\_df.head(15)



train\_df = train\_df.drop("author", axis = 1)

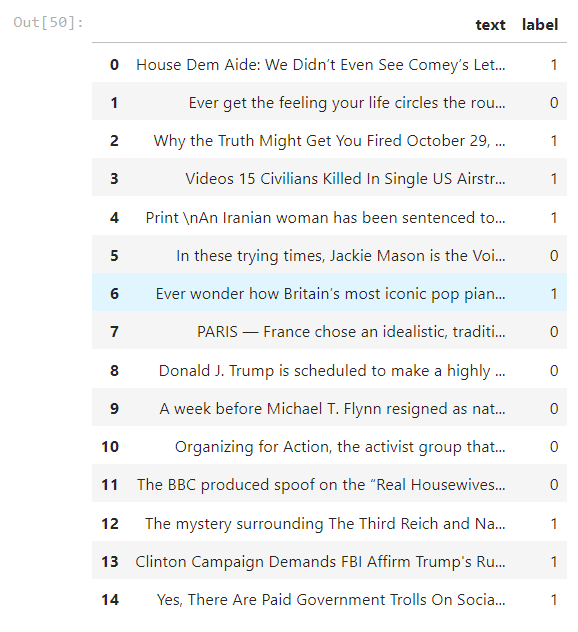
train\_df = train\_df.drop("title", axis = 1)

train\_df = train\_df.drop("id", axis = 1)

train\_df.shape



train\_df.head(15)



**Program Code for Feature-Extraction and Classification(Passive Aggressive):**

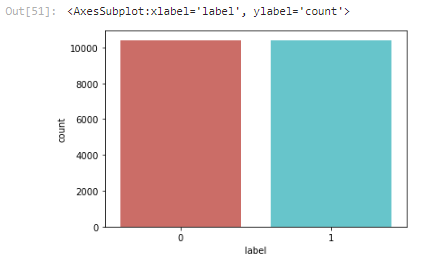
**(Choice of machine learning algorithm, model training, and evaluation metrics)**

def create\_distribution(dataFile):

return sb.countplot(x='label', data=dataFile, palette='hls')

# by calling below we can see that training, test and valid data seems to be failry evenly distributed between the classes

create\_distribution(train\_df)



def data\_qualityCheck():

print("Checking data qualitites...")

train\_df.isnull().sum()

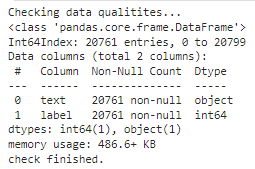
train\_df.info()

print("check finished.")

data\_qualityCheck()

train\_df = train\_df.dropna()

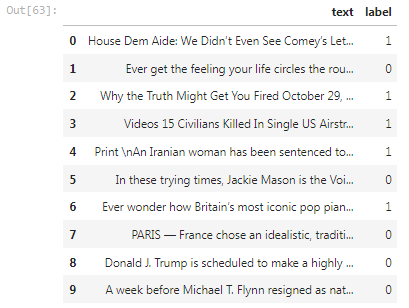
data\_qualityCheck()



train\_df.shape

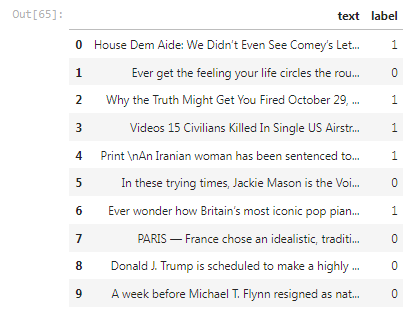


train\_df.head(10)



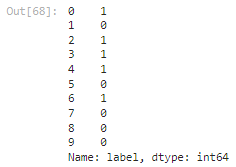
train\_df.reset\_index(drop= True,inplace=True)

train\_df.head(10)



label\_train = train\_df.label

label\_train.head(10)

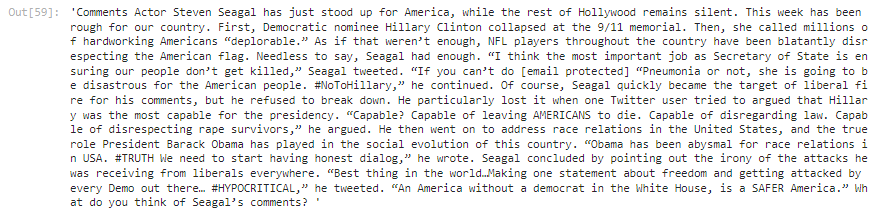


train\_df = train\_df.drop("label", axis = 1)

train\_df.head(10)



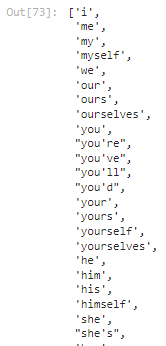
train\_df['text'][2188]



lemmatizer = WordNetLemmatizer()

stpwrds = list(stopwords.words('english'))

stpwrds



for x in range(len(train\_df)) :

corpus = []

review = train\_df['text'][x]

review = re.sub(r'[^a-zA-Z\s]', '', review)

review = review.lower()

review = nltk.word\_tokenize(review)

for y in review :

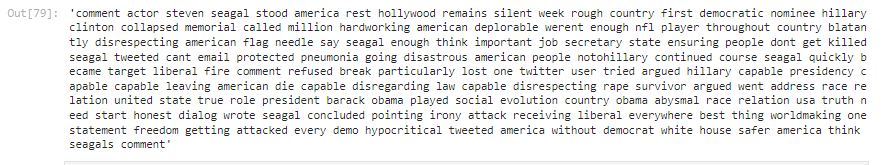
if y not in stpwrds :

corpus.append(lemmatizer.lemmatize(y))

review = ' '.join(corpus)

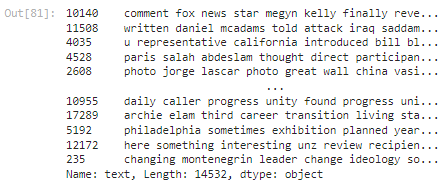
train\_df['text'][x] = review

train\_df['text'][2182]



X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(train\_df['text'], label\_train, test\_size=0.3, random\_state=1)

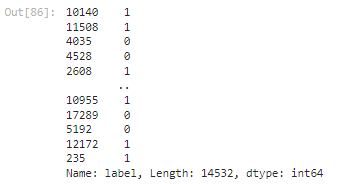
X\_train



X\_train.shape



Y\_train



tfidf\_v = TfidfVectorizer()

tfidf\_X\_train = tfidf\_v.fit\_transform(X\_train)

tfidf\_X\_test = tfidf\_v.transform(X\_test)

tfidf\_X\_train.shape



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, j in itertools.product(range(cm.shape[0]), 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')

classifier = PassiveAggressiveClassifier()

classifier.fit(tfidf\_X\_train,Y\_train)



Y\_pred = classifier.predict(tfidf\_X\_test)

score = metrics.accuracy\_score(Y\_test, Y\_pred)

print(f'Accuracy: {round(score\*100,2)}%')

cm = metrics.confusion\_matrix(Y\_test, Y\_pred)

plot\_confusion\_matrix(cm, classes=['FAKE Data', 'REAL Data'])

