id: unique id for news article

title: title of the news

subject: subject of the news

text: text of the article

date: date of the article

label: label that marks whether the news is fake or real

1- fake news

1. real news

import pandas as pd

import numpy as np

import re

from nltk.corpus import stopwords

import matplotlib.pyplot as plt

from nltk.stem.porter import PorterStemmer

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

fake\_d= pd.read\_csv(r"C:\Users\HP\Desktop\fake.csv")

true\_d= pd.read\_csv(r"C:\Users\HP\Desktop\true.csv")

fake\_d.head()

true\_d.head()

import nltk

nltk.download('stopwords')

#printing the stopwrods(word that add no value to the news)

print(stopwords.words('english'))

fake\_d.shape

fake\_d["label"]=1

true\_d["label"]=0

fake\_d.shape

train\_set=pd.concat([fake\_d,true\_d])

train\_set=train\_set.sample(frac=1).reset\_index(drop=True)

train\_set.head()

train\_set.dtypes

label\_distribution=train\_set['label'].value\_counts()

labels=[f"{label} ({count})" for label, count in zip(label\_distribution.index,label\_distribution.values)]

plt.figure(figsize=(4, 4))

plt.pie(label\_distribution,labels=labels,autopct='%1.1f%%',startangle=140)

plt.title('Label Distribution')

plt.show()

train\_set.isnull().sum()

# replacing empty string with null strings

#train\_set=train\_set.fillna()

# merging the subject and the title

train\_set['content']=train\_set['subject']+' '+train\_set['title']

print(train\_set['content'])

#separating the label and the data

X=train\_set.drop(columns='label',axis=1)

Y=train\_set['label']

print(X)

print(Y)

#stemming is a process of reducing a word to its root word

#eg: acting,actor,actress ----act is the root word

port\_stem= PorterStemmer()

def stemming(content):

stemmed\_content = re.sub('[^a-zA-Z]',' ',content)

stemmed\_content = stemmed\_content.lower()

stemmed\_content = stemmed\_content.split()

stemmed\_content = [port\_stem.stem(word) for word in stemmed\_content if not word in stopwords.words('english')]

stemmed\_content = ' '.join(stemmed\_content)

return stemmed\_content

train\_set['content']=train\_set['content'].apply(stemming)

print(train\_set['content'])

#separating the label and the data

X = train\_set['content'].values

Y = train\_set['label'].values

print(X)

Print(Y)

X.shape

Y.shape

#converting the textual data to numerical data

vectorizer= TfidfVectorizer()

vectorizer.fit(X)

X=vectorizer.transform(X)

print(X)

#splitting the dataset to training and test data

X\_train,X\_test,Y\_train,Y\_test= train\_test\_split(X,Y,test\_size=0.2,stratify=Y,random\_state=2)

model= LogisticRegression()

model.fit(X\_train, Y\_train)

#accuracy score on the training data

X\_train\_prediction = model.predict(X\_train)

training\_data\_accuracy = accuracy\_score(X\_train\_prediction,Y\_train)

print('accuracy score of the training data: ',training\_data\_accuracy)

#accuracy score on the test data

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction,Y\_test)

print('accuracy score of the test data: ',test\_data\_accuracy)

X\_new=X\_test[0]

prediction=model.predict(X\_new)

print(prediction)

if(prediction[0]==0):

print("The news is real")

else:

print("The news is fake")

print(Y\_test[0])