import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics import classification\_report

import re

import string

import matplotlib.pyplot as plt

true\_path = 'True.csv'

fake\_path = 'Fake.csv'

true\_news = pd.read\_csv (true\_path)

fake\_news = pd.read\_csv(fake\_path)

true\_news['class'] = 1

fake\_news['class'] = 0

true\_manual\_testing = true\_news.tail(10)

fake\_manual\_testing = fake\_news.tail (10)

true\_news = true\_news.iloc [:-10]

fake\_news = fake\_news.iloc [:-10]

data\_manual\_testing = pd.concat ([true\_manual\_testing, fake\_manual\_testing])

data \_merge = pd.concat ([true\_news, fake\_news])

subject\_count = data merge['subject'].value\_counts()

print(subject\_count)

subject\_count.plot(kind='bar')

plt.show()

class\_count = data \_merge ['class'].value\_counts()

print (class\_count)

class\_count.plot(kind='pie', autopct='%1.1f%%')

plt.show ()

data = data\_merge.drop(columns=['title', 'subject', 'date'])

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

print(data.isnull().sum())

def filtering(text):

text = re.sub(r'http\S+', '', text)

text = re.sub(r'<.\*?>', '', text)

text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)

text = re.sub(r'\d+', '', text)

text = text.lower()

return text

data['text'] = data['text'].apply(filtering)

x = data['text']

y = data['class']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=42)

vector = TfidfVectorizer ()

x\_train = vector.fit\_transform(x\_train)

x\_test = vector.transform(x\_test)

log\_reg = LogisticRegression()

log\_reg.fit(x\_train, y\_train)

y\_pred\_log\_reg = log\_reg.predict(x\_test)

print("Logistic Regression Classification Report:\n")

print(classification\_report(y\_test, y\_pred\_log\_reg))

dec\_tree = DecisionTreeClassifier()

dec\_tree.fit(x\_train, y\_train)

y\_pred\_dec\_tree = dec\_tree.predict(x\_test)

print ("Decision Tree Classifier Classification Report:\n")

print(classification\_report(y\_test, y\_pred\_dec\_tree))

def output\_label(n):

return "True News" if n == 1 else "Fake News"

def manual\_testing(news):

news = filtering(news)

news\_vector = vector.transform([news])

pred\_log\_reg = log\_reg.predict(news\_vector)

pred\_dec\_tree = dec\_tree.predict(news\_vector)

return f"Logistic Regression Prediction:

{output\_label(pred\_log\_reg[0])}\n

Decision Tree Prediction :{output\_label(pred\_dec\_tree[0])}"

input\_news = input("Enter the news text to test: ")

print(manual\_testing(input\_news))