pip install VaderSentiment

import pandas as pd

import numpy as np

from textblob import TextBlob

import re

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

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

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

df1 = pd.read\_csv('dow\_jones\_news.csv')

df2 = pd.read\_csv('dow\_jones\_stock.csv')

df1.head(5)

df1.shape

df2.head(5)

df2.shape

merge = df1.merge(df2, how='inner', left\_index=True, right\_index=True)

merge

merge.drop(columns=['Date\_y'], inplace=True)

merge

headlines = []

for row in range(0, len(merge.index)):

  headlines.append(' '.join(str(x) for x in merge.iloc[row, 2:27]))

clean\_headlines = []

for i in range(0, len(headlines)):

  clean\_headlines.append(re.sub("b[(')]",'', headlines[i]))

  clean\_headlines[i] = re.sub('b[(")]', '', clean\_headlines[i])

  clean\_headlines[i] = re.sub("\'", '', clean\_headlines[i])

clean\_headlines[20]

merge['combined\_news'] = clean\_headlines

merge['combined\_news'][0]

merge.head(5)

def getsubjectivity(text):

  return TextBlob(text).sentiment.subjectivity

def getPolarity(text):

  return TextBlob(text).sentiment.polarity

merge['subjective'] = merge['combined\_news'].apply(getsubjectivity)

merge['Polarity'] = merge['combined\_news'].apply(getPolarity)

merge.head(5)

def getSIA(text):

  sia = SentimentIntensityAnalyzer()

  sentiment = sia.polarity\_scores(text)

  return sentiment

compound = []

neg = []

pos = []

neu = []

SIA = 0

for i in range(0, len(merge['combined\_news'])):

  SIA = getSIA(merge['combined\_news'][i])

  compound.append(SIA['compound'])

  neg.append(SIA['neg'])

  neu.append(SIA['neu'])

  pos.append(SIA['pos'])

merge['compound'] = compound

merge['negative'] = neg

merge['neutral'] = neu

merge['positive'] = pos

merge.head(5)

keep\_columns = [ 'Open', 'High', 'Low', 'Volume', 'subjective', 'Polarity', 'compound', 'negative', 'neutral' ,'positive',  'Label' ]

df = merge[keep\_columns]

df

X = df

X = np.array(X.drop(['Label'], 1))

y = np.array(df['Label'])

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 32)

model = LinearDiscriminantAnalysis().fit(x\_train,y\_train)

pred = model.predict(x\_test)

pred

y\_test

print(classification\_report(y\_test, pred))

from flask import Flask, render\_template, request

from nltk.sentiment.vader import SentimentIntensityAnalyzer

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

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.metrics import classification\_report

import numpy as np

import pandas as pd

app = Flask(\_\_name\_\_)

def getSIA(text):

    sia = SentimentIntensityAnalyzer()

    sentiment = sia.polarity\_scores(text)

    return sentiment

@app.route('/')

def home():

    return render\_template('index.html')

@app.route('/analyze', methods=['POST'])

def analyze():

    if request.method == 'POST':

        merge = pd.read\_csv('path\_to\_your\_csv\_file.csv')  # Adjust the path to your CSV file

        compound = []

        neg = []

        pos = []

        neu = []

        SIA = 0

        for i in range(0, len(merge['combined\_news'])):

            SIA = getSIA(merge['combined\_news'][i])

            compound.append(SIA['compound'])

            neg.append(SIA['neg'])

            neu.append(SIA['neu'])

            pos.append(SIA['pos'])

        merge['compound'] = compounda

        merge['negative'] = neg

        merge['neutral'] = neu

        merge['positive'] = pos

        x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=32)

        model = LinearDiscriminantAnalysis().fit(x\_train, y\_train)

        pred = model.predict(x\_test)

        classification\_result = classification\_report(y\_test, pred)

        return render\_template('result.html', classification\_result=classification\_result)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)