SOURCE CODE:

import matplotlib.pyplot as plt

import seaborn as sns

from pandas\_datareader.data import DataReader

import yfinance as yf

from flask import Flask,render\_template,request, redirect,url\_for

import mysql.connector

import requests

from bs4 import BeautifulSoup

from matplotlib import pyplot as plt

import numpy as np

import os

from math import ceil

import re

from datetime import datetime

import pandas as pd

import numpy as np

from textblob import TextBlob

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

sns.set\_style('whitegrid')

plt.style.use("fivethirtyeight")

UPLOAD\_FOLDER = 'static/file/'

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

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

mydb = mysql.connector.connect(host="localhost",user="root",password="",database="stock")

mycursor = mydb.cursor()

@app.route('/')

@app.route('/login')

def login():

return render\_template('login.html')

@app.route('/reg')

def reg():

return render\_template('reg.html')

@app.route('/uppage')

def uppage():

return render\_template('admin.html')

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

def register():

if request.method == 'POST':

name = request.form.get('name')

phone = request.form.get('phone')

password = request.form.get('password')

sql = "INSERT INTO users (`name`, `phone`, `password`) VALUES (%s, %s, %s)"

val = (name,phone,password)

mycursor.execute(sql, val)

mydb.commit()

return render\_template('login.html')

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

def validate():

if request.method == 'POST':

data1 = request.form.get('username')

data2 = request.form.get('password')

sql = "SELECT \* FROM `users` WHERE `name` = %s AND `password` = %s"

val = (data1, data2)

mycursor.execute(sql,val)

account = mycursor.fetchone()

if account:

return redirect(url\_for('index'))

elif data1 == 'Admin' and data2 == 'Admin':

return redirect(url\_for('uppage'))

else:

return render\_template('login.html',msg = 'Invalid')

@app.route('/index')

def index():

global company\_list,tech\_list,company\_name,start, end

tech\_list = ['AAPL', 'GOOG', 'MSFT', 'AMZN']

tech\_list = ['AAPL', 'GOOG', 'MSFT', 'AMZN']

end = datetime.now()

start = datetime(end.year - 1, end.month, end.day)

for stock in tech\_list:

globals()[stock] = yf.download(stock, start, end)

company\_list = [AAPL, GOOG, MSFT, AMZN]

company\_name = ["APPLE", "GOOGLE", "MICROSOFT", "AMAZON"]

for company, com\_name in zip(company\_list, company\_name):

company["company\_name"] = com\_name

df = pd.concat(company\_list, axis=0)

print(df.tail(10))

# print(AAPL.describe())

# print(AAPL.info())

plt.figure(figsize=(15, 6))

plt.subplots\_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(company\_list, 1):

plt.subplot(2, 2, i)

company['Adj Close'].plot()

plt.ylabel('Adj Close')

plt.xlabel(None)

plt.title(f"Closing Price of {tech\_list[i - 1]}")

plt.tight\_layout()

plt.savefig('static/img/plot1.png')

return render\_template('index.html',fpath = 'static/img/plot1.png')

@app.route('/daily')

def daily():

plt.figure(figsize=(15, 7))

plt.subplots\_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(company\_list, 1):

plt.subplot(2, 2, i)

company['Volume'].plot()

plt.ylabel('Volume')

plt.xlabel(None)

plt.title(f"Sales Volume for {tech\_list[i - 1]}")

plt.tight\_layout()

# plt.savefig('static/img/plot2.png')

ma\_day = [10, 20, 50]

for ma in ma\_day:

for company in company\_list:

column\_name = f"MA for {ma} days"

company[column\_name] = company['Adj Close'].rolling(ma).mean()

fig, axes = plt.subplots(nrows=2, ncols=2)

fig.set\_figheight(8)

fig.set\_figwidth(15)

AAPL[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,0])

axes[0,0].set\_title('APPLE')

GOOG[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,1])

axes[0,1].set\_title('GOOGLE')

MSFT[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,0])

axes[1,0].set\_title('MICROSOFT')

AMZN[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,1])

axes[1,1].set\_title('AMAZON')

fig.tight\_layout()

# plt.savefig('static/img/plot2.png')

for company in company\_list:

company['Daily Return'] = company['Adj Close'].pct\_change()

fig, axes = plt.subplots(nrows=2, ncols=2)

fig.set\_figheight(8)

fig.set\_figwidth(15)

AAPL['Daily Return'].plot(ax=axes[0,0], legend=True, linestyle='--', marker='o')

axes[0,0].set\_title('APPLE')

GOOG['Daily Return'].plot(ax=axes[0,1], legend=True, linestyle='--', marker='o')

axes[0,1].set\_title('GOOGLE')

MSFT['Daily Return'].plot(ax=axes[1,0], legend=True, linestyle='--', marker='o')

axes[1,0].set\_title('MICROSOFT')

AMZN['Daily Return'].plot(ax=axes[1,1], legend=True, linestyle='--', marker='o')

axes[1,1].set\_title('AMAZON')

fig.tight\_layout()

# plt.savefig('static/img/plot3.png')

plt.figure(figsize=(12, 7))

for i, company in enumerate(company\_list, 1):

plt.subplot(2, 2, i)

company['Daily Return'].hist(bins=50)

plt.ylabel('Daily Return')

plt.title(f'{company\_name[i - 1]}')

plt.tight\_layout()

plt.savefig('static/img/plot4.png')

return render\_template('index.html',fpath = 'static/img/plot4.png')

@app.route('/predict')

def predict():

closing\_df = DataReader(tech\_list, 'yahoo', start, end)['Adj Close']

print(closing\_df.head())

tech\_rets = closing\_df.pct\_change()

print(tech\_rets.head())

sns.jointplot(x='GOOG', y='GOOG', data=tech\_rets, kind='scatter', color='seagreen')

# plt.savefig('static/img/plot5.png')

sns.jointplot(x='GOOG', y='MSFT', data=tech\_rets, kind='scatter')

# plt.savefig('static/img/plot6.png')

sns.pairplot(tech\_rets, kind='reg')

# plt.savefig('static/img/plot7.png')

return\_fig = sns.PairGrid(tech\_rets.dropna())

return\_fig.map\_upper(plt.scatter, color='purple')

return\_fig.map\_lower(sns.kdeplot, cmap='cool\_d')

return\_fig.map\_diag(plt.hist, bins=30)

returns\_fig = sns.PairGrid(closing\_df)

returns\_fig.map\_upper(plt.scatter,color='purple')

returns\_fig.map\_lower(sns.kdeplot,cmap='cool\_d')

returns\_fig.map\_diag(plt.hist,bins=30)

sns.heatmap(tech\_rets.corr(), annot=True, cmap='summer')

# plt.savefig('static/img/plot8.png')

sns.heatmap(closing\_df.corr(), annot=True, cmap='summer')

# plt.savefig('static/img/plot9.png')

rets = tech\_rets.dropna()

area = np.pi \* 20

plt.figure(figsize=(10, 7))

plt.scatter(rets.mean(), rets.std(), s=area)

plt.xlabel('Expected return')

plt.ylabel('Risk')

for label, x, y in zip(rets.columns, rets.mean(), rets.std()):

plt.annotate(label, xy=(x, y), xytext=(50, 50), textcoords='offset points', ha='right', va='bottom',

arrowprops=dict(arrowstyle='-', color='blue', connectionstyle='arc3,rad=-0.3'))

df = DataReader('AAPL', data\_source='yahoo', start='2012-01-01', end=datetime.now())

print(df)

plt.figure(figsize=(16,6))

plt.title('Close Price History')

plt.plot(df['Close'])

plt.xlabel('Date', fontsize=18)

plt.ylabel('Close Price USD ($)', fontsize=18)

# plt.savefig('static/img/plot10.png')

data = df.filter(['Close'])

dataset = data.values

training\_data\_len = int(np.ceil( len(dataset) \* .95 ))

print(training\_data\_len)

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature\_range=(0,1))

scaled\_data = scaler.fit\_transform(dataset)

print(scaled\_data)

train\_data = scaled\_data[0:int(training\_data\_len), :]

x\_train = []

y\_train = []

for i in range(60, len(train\_data)):

x\_train.append(train\_data[i-60:i, 0])

y\_train.append(train\_data[i, 0])

if i<= 61:

print(x\_train)

print(y\_train)

x\_train, y\_train = np.array(x\_train), np.array(y\_train)

x\_train = np.reshape(x\_train, (x\_train.shape[0], x\_train.shape[1], 1))

# from keras.models import Sequential

# from keras.layers import Dense, LSTM

# model = Sequential()

# model.add(LSTM(128, return\_sequences=True, input\_shape= (x\_train.shape[1], 1)))

# model.add(LSTM(64, return\_sequences=False))

# model.add(Dense(25))

# model.add(Dense(1))

# model.compile(optimizer='adam', loss='mean\_squared\_error')

# model.fit(x\_train, y\_train, batch\_size=1, epochs=1)

# test\_data = scaled\_data[training\_data\_len - 60: , :]

# x\_test = []

# y\_test = dataset[training\_data\_len:, :]

# for i in range(60, len(test\_data)):

# x\_test.append(test\_data[i-60:i, 0])

# x\_test = np.array(x\_test)

# x\_test = np.reshape(x\_test, (x\_test.shape[0], x\_test.shape[1], 1 ))

# predictions = model.predict(x\_test)

# predictions = scaler.inverse\_transform(predictions)

# rmse = np.sqrt(np.mean(((predictions - y\_test) \*\* 2)))

# print(rmse)

# train = data[:training\_data\_len]

# valid = data[training\_data\_len:]

# valid['Predictions'] = predictions

# plt.figure(figsize=(16,6))

# plt.title('Model')

# plt.xlabel('Date', fontsize=18)

# plt.ylabel('Close Price USD ($)', fontsize=18)

# plt.plot(train['Close'])

# plt.plot(valid[['Close', 'Predictions']])

# plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')

# plt.savefig('static/img/plot11.png')

return render\_template('index.html',fpath = 'static/img/plot11.png')

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

app.run(debug=True)