Personal Project

import os

import smtplib

import imghdr

from email.message import EmailMessage

import yfinance as yf

import datetime as dt

import pandas as pd

from pandas\_datareader import data as pdr

email\_address = 'georgefelty1997@gmail.com'

email\_password = 'topeqsvkvcvxhzhe'

msg=EmailMessage()

yf.pdr\_override()

start=dt.datetime(2018, 12,1)

now= dt.datetime.now()

stock="QQQ"

TargetPrice=180

msg["Subject"]= "Alert on "+stock

msg["From"]= email\_address

msg["To"]='kkfelty@uark.edu'

alerted=False

while 1:

df= pdr.get\_data\_yahoo(stock, start, now)

currentClose=df["Adj Close"][-1]

condition=currentClose>TargetPrice

if(condition and alerted==False):

alerted=True

message= stock +" Has activated the alert price of"+ str(TargetPrice) +\

"\nCurrent Price: "+str(currentClose)

msg.set\_content(message)

files=[r"C:\\Users\\kentf\\stocks\\QQQ\\FundamentalList.xlsx"]

for file in files:

with open(file,"rb") as f:

file\_data=f.read()

file\_name="FundamentalList.xlsx"

msg.add\_attachment(file\_data, maintype="application",

subtype='ocetet-stream', filenname=file\_name)

with smtplib.SMTP\_SSL('smtp.gmail.com',465) as smtp:

smtp.login(email\_address,email\_password)

smtp.send\_message(msg)

print("completed")

else:

print("no new alerts")

time.sleep(60)

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import yfinance as yf

df = yf.Ticker("QQQ-USD").history(period="max").reset\_index()[["Open"]]

df.shape

dataset\_train = np.array(df[:int(df.shape[0]\*0.8)])

dataset\_test = np.array(df[int(df.shape[0]\*0.8):])

print(dataset\_train.shape)

print(dataset\_test.shape)

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential, load\_model

from keras.layers import LSTM, Dense, Dropout

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

dataset\_train = scaler.fit\_transform(dataset\_train)

dataset\_train[:5]

dataset\_test = scaler.transform(dataset\_test)

dataset\_test[:5]

def create\_dataset(df):

x = []

y = []

for i in range(50, df.shape[0]):

x.append(df[i-50:i, 0])

y.append(df[i, 0])

x = np.array(x)

y = np.array(y)

return x,y

x\_train, y\_train = create\_dataset(dataset\_train)

x\_test, y\_test = create\_dataset(dataset\_test)

model = Sequential()

model.add(LSTM(units=96, return\_sequences=True, input\_shape=(x\_train.shape[1], 1)))

model.add(Dropout(0.2))

model.add(LSTM(units=96, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(units=96, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(units=96))

model.add(Dropout(0.2))

model.add(Dense(units=1))

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

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

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

model.fit(x\_train, y\_train, epochs=50, batch\_size=32)

model.save('stock\_prediction.h5')

predictions = model.predict(x\_test)

predictions = scaler.inverse\_transform(predictions)

y\_test\_scaled = scaler.inverse\_transform(y\_test.reshape(-1, 1))

fig, ax = plt.subplots(figsize=(16,8))

ax.set\_facecolor('white')

ax.plot(y\_test\_scaled, color='red', label='Original price')

plt.plot(predictions, color='cyan', label='Predicted price')

plt.legend()