pip install yfinance

import pandas as pd

import numpy as np

from pandas\_datareader import data as pdr

import matplotlib.pyplot as plt

import yfinance as yf

from keras.models import load\_model

start='2000-01-01'

end='2022-12-31'

df= yf.download('AMZN')

df

df= df.reset\_index()

df.head()

df = df[['Date', 'Close']]

df

df['Date']

import datetime

def str\_to\_datetime(s):

split = s.split('-')

year, month, day = int(split[0]), int(split[1]), int(split[2])

return datetime.datetime(year=year, month=month, day=day)

datetime\_object = str\_to\_datetime('1986-03-19')

datetime\_object

df

df.index = df.pop('Date')

df

import matplotlib.pyplot as plt

plt.plot(df.index, df['Close'])

import numpy as np

def df\_to\_windowed\_df(dataframe, first\_date\_str, last\_date\_str, n=3):

first\_date = str\_to\_datetime(first\_date\_str)

last\_date = str\_to\_datetime(last\_date\_str)

target\_date = first\_date

dates = []

X, Y = [], []

last\_time = False

while True:

df\_subset = dataframe.loc[:target\_date].tail(n+1)

if len(df\_subset) != n+1:

print(f'Error: Window of size {n} is too large for date {target\_date}')

return

values = df\_subset['Close'].to\_numpy()

x, y = values[:-1], values[-1]

dates.append(target\_date)

X.append(x)

Y.append(y)

next\_week = dataframe.loc[target\_date:target\_date+datetime.timedelta(days=7)]

next\_datetime\_str = str(next\_week.head(2).tail(1).index.values[0])

next\_date\_str = next\_datetime\_str.split('T')[0]

year\_month\_day = next\_date\_str.split('-')

year, month, day = year\_month\_day

next\_date = datetime.datetime(day=int(day), month=int(month), year=int(year))

if last\_time:

break

target\_date = next\_date

if target\_date == last\_date:

last\_time = True

ret\_df = pd.DataFrame({})

ret\_df['Target Date'] = dates

X = np.array(X)

for i in range(0, n):

X[:, i]

ret\_df[f'Target-{n-i}'] = X[:, i]

ret\_df['Target'] = Y

return ret\_df

# Start day second time around: '2021-03-25'

windowed\_df = df\_to\_windowed\_df(df,

'2021-03-25',

'2022-03-23',

n=3)

windowed\_df

def windowed\_df\_to\_date\_X\_y(windowed\_dataframe):

df\_as\_np = windowed\_dataframe.to\_numpy()

dates = df\_as\_np[:, 0]

middle\_matrix = df\_as\_np[:, 1:-1]

X = middle\_matrix.reshape((len(dates), middle\_matrix.shape[1], 1))

Y = df\_as\_np[:, -1]

return dates, X.astype(np.float32), Y.astype(np.float32)

dates, X, y = windowed\_df\_to\_date\_X\_y(windowed\_df)

dates.shape, X.shape, y.shape

q\_80 = int(len(dates) \* .8)

q\_90 = int(len(dates) \* .9)

dates\_train, X\_train, y\_train = dates[:q\_80], X[:q\_80], y[:q\_80]

dates\_val, X\_val, y\_val = dates[q\_80:q\_90], X[q\_80:q\_90], y[q\_80:q\_90]

dates\_test, X\_test, y\_test = dates[q\_90:], X[q\_90:], y[q\_90:]

plt.plot(dates\_train, y\_train)

plt.plot(dates\_val, y\_val)

plt.plot(dates\_test, y\_test)

plt.legend(['Train', 'Validation', 'Test'])

from tensorflow.keras.models import Sequential

from tensorflow.keras.optimizers import Adam

from tensorflow.keras import layers

model = Sequential([layers.Input((3, 1)),

layers.LSTM(64),

layers.Dense(32, activation='relu'),

layers.Dense(32, activation='relu'),

layers.Dense(1)])

model.compile(loss='mse',

optimizer=Adam(learning\_rate=0.001),

metrics=['mean\_absolute\_error'])

model.fit(X\_train, y\_train, validation\_data=(X\_val, y\_val), epochs=100)

model.save('keras\_main\_model.h5')

train\_predictions = model.predict(X\_train).flatten()

plt.plot(dates\_train, train\_predictions)

plt.plot(dates\_train, y\_train)

plt.legend(['Training Predictions', 'Training Observations'])

val\_predictions = model.predict(X\_val).flatten()

plt.plot(dates\_val, val\_predictions)

plt.plot(dates\_val, y\_val)

plt.legend(['Validation Predictions', 'Validation Observations'])

test\_predictions = model.predict(X\_test).flatten()

plt.plot(dates\_test, test\_predictions)

plt.plot(dates\_test, y\_test)

plt.legend(['Testing Predictions', 'Testing Observations'])

plt.plot(dates\_train, train\_predictions)

plt.plot(dates\_train, y\_train)

plt.plot(dates\_val, val\_predictions)

plt.plot(dates\_val, y\_val)

plt.plot(dates\_test, test\_predictions)

plt.plot(dates\_test, y\_test)

plt.legend(['Training Predictions',

'Training Observations',

'Validation Predictions',

'Validation Observations',

'Testing Predictions',

'Testing Observations'])

from copy import deepcopy

recursive\_predictions = []

recursive\_dates = np.concatenate([dates\_val, dates\_test])

for target\_date in recursive\_dates:

last\_window = deepcopy(X\_train[-1])

next\_prediction = model.predict(np.array([last\_window])).flatten()

recursive\_predictions.append(next\_prediction)

last\_window[-1] = next\_prediction

plt.plot(dates\_train, train\_predictions)

plt.plot(dates\_train, y\_train)

plt.plot(dates\_val, val\_predictions)

plt.plot(dates\_val, y\_val)

plt.plot(dates\_test, test\_predictions)

plt.plot(dates\_test, y\_test)

plt.plot(recursive\_dates, recursive\_predictions)

plt.legend(['Training Predictions',

'Training Observations',

'Validation Predictions',

'Validation Observations',

'Testing Predictions',

'Testing Observations',

'Recursive Predictions'])