####LSTM 구해보리기!####

from pykrx import stock

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import os

from matplotlib.pylab import rcParams

rcParams['figure.figsize']=20,10

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM,Dropout,Dense

from sklearn.preprocessing import MinMaxScaler

# Get Stock List

path = '/content/project3'

list\_name = 'Stock\_List.csv'

sample\_name = 'sample\_submission\_week4.csv'

# 종목 코드 로드

stock\_list =pd.read\_csv(os.path.join(path, list\_name))

stock\_list['종목코드'] = stock\_list['종목코드'].apply(lambda x: str(x).zfill(6))

stock\_list

# Get Data & Modeling

# 분석할 date 변수 지정

start\_date = '20100531'

end\_date = '20210917'

start\_weekday = pd.to\_datetime(start\_date).weekday()

max\_weeknum = pd.to\_datetime(end\_date).strftime('%V')

business\_days = pd.DataFrame(pd.date\_range(start\_date, end\_date, freq='B'), columns=['Date'])

print(f'WEEKDAY of "start\_date" : {start\_weekday}')

print(f'NUM of WEEKS to "end\_date" : {max\_weeknum}')

print(f'HOW MANY "Business\_days" : {business\_days.shape}', )

print(business\_days.head(20))

# ===== raw data loading =====

# 한 종목코드에 대한 주가 정보를 로드

stock\_code = stock\_list.loc[0, '종목코드']

stock\_df = stock.get\_market\_ohlcv\_by\_date(start\_date, end\_date, stock\_code).reset\_index()

investor\_df = stock.get\_market\_trading\_volume\_by\_date(start\_date, end\_date, stock\_code)[["기관합계", "외국인합계"]] .reset\_index()

stock\_df.columns = ["Date", "Open", "High", "Low", "Close", "Volume"]

investor\_df.columns = ["Date", "inst", "fore"]

# 영업일과 주가 정보를 outer 조인

train\_x = pd.merge(business\_days, stock\_df, how='left', on="Date")

train\_x = pd.merge(train\_x, investor\_df, how='left', on="Date")

# 종가데이터에 생긴 na 값을 선형보간 및 정수로 반올림

train\_x.iloc[:,1:] = train\_x.iloc[:,1:].interpolate(axis=0).round(0)

####train 데이터 전처리####

train\_x["Date"] = pd.to\_datetime(train\_x.Date, format="%Y-%m-%d")

train\_x

#df.index = df['Date']

#df

train\_x.index = train\_x['Date']

####종가 그래프 찍기####

plt.figure(figsize=(16,8))

plt.plot(train\_x["Close"], label = 'Close Price history')

####Date, 종가 데이터셋 만들기####

data = train\_x.sort\_index(ascending=True, axis=0)

new\_dataset=pd.DataFrame(index=range(0,len(train\_x)),columns = ['Date', 'Close'])

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

new\_dataset["Date"][i] = data['Date'][i]

new\_dataset["Close"][i] = data['Close'][i]

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

new\_dataset.index = new\_dataset.Date

new\_dataset.drop("Date", axis=1, inplace=True)

####데이터 분리####

final\_dataset = new\_dataset.values

train\_data = final\_dataset[0:2513, : ]

valid\_data = final\_dataset[2513: , : ]

####데이터 스케일해주기####

scaled\_data = scaler.fit\_transform(final\_dataset)

####Train 데이터 구성####

x\_train\_data = []

y\_train\_data = []

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

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

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

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

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

####LSTM 모델 만들기####

lstm\_model = Sequential()

lstm\_model.add(LSTM(units=50,return\_sequences=True,input\_shape=(x\_train\_data.shape[1],1)))

lstm\_model.add(LSTM(units=50))

lstm\_model.add(Dense(1))

lstm\_model.summary()

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

lstm\_model.fit(x\_train\_data, y\_train\_data, epochs=1, batch\_size=1, verbose=2)

####test\_data 구하기####

inputs\_data = new\_dataset[len(new\_dataset)-len(valid\_data)-60:].values

inputs\_data = inputs\_data.reshape(-1,1)

inputs\_data=scaler.transform(inputs\_data)

X\_test = []

for i in range(60, inputs\_data.shape[0]):

X\_test.append(inputs\_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))

####주가 예측####

predicted\_closing\_price = lstm\_model.predict(X\_test)

predicted\_closing\_price

####스케일한거 되돌려보리기!####

predicted\_closing\_price = scaler.inverse\_transform(predicted\_closing\_price)

predicted\_closing\_price

####시각화####

train\_data = new\_dataset[:2513]

valid\_data=new\_dataset[2513:]

valid\_data['Predictions']=predicted\_closing\_price

plt.plot(train\_data["Close"])

plt.plot(valid\_data[['Close', "Predictions"]])