주가예측 보고서

신준재 김주원 안정모 정서현

CONTENTS .

01.
개념설명

DEEP LEARNING

TENSORFLOW .

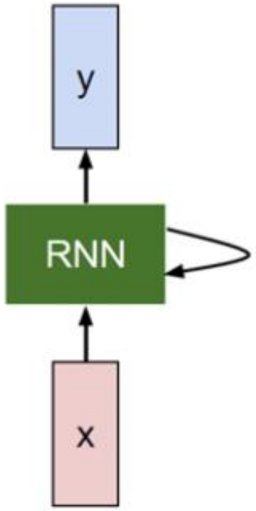
DEEP LEARNING

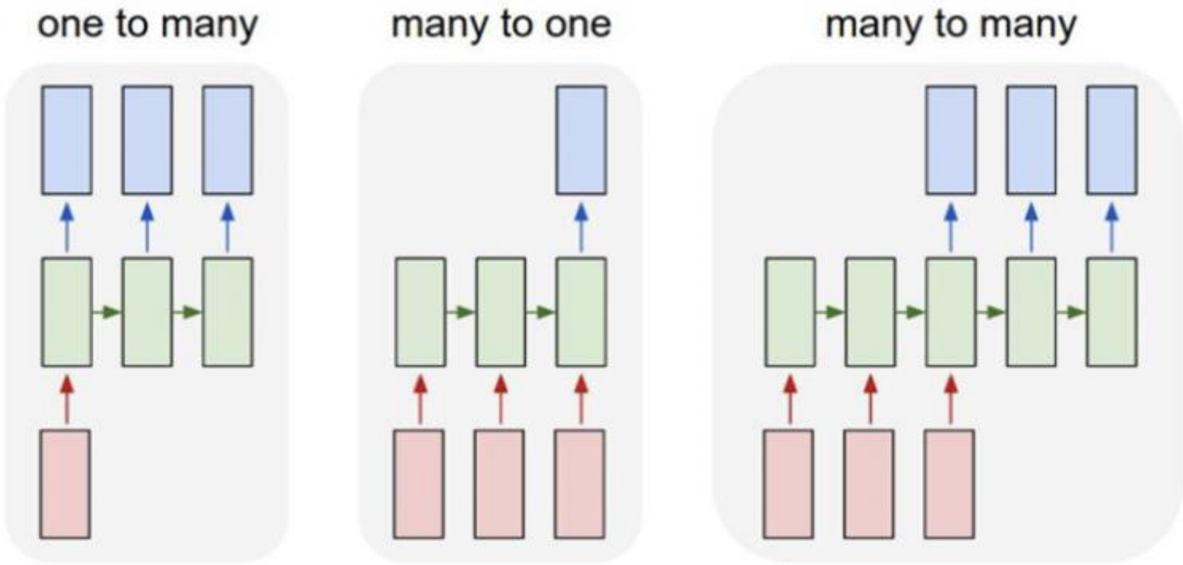
머신러닝의 한 종류로, 대규모의 데이터에서
중요한 패턴 이나 규칙을 학습하고,
이를 토대로 예측을 하는 기술

TENSORFLOW

구글이 개발한 파이썬 딥러닝 오픈소스 라이브러리

LSTM





RNN

-반복적이고 순차적인 데이터(Sequential data) 학습에 특화된 인공신경망의 한 종류

-데이터의 길이가 길어질 수록 앞쪽의 데이터 내용이 뒤쪽으로 전달되지 않는 '장기 의존성' 문제

LSTM

-'장기 의존성' 문제를 해결한 모델이 바로 LSTM (Long Short Term Memory network).

-LSTM은 Long-term Memory (장기기억)을 보존하는 능력이 있음.

MANY-TO-ONE

-여러 데이터를 학습하여 하나의 결과값을 만들어 냄.

▼

MANY-TO-ONE을 선택한 이유

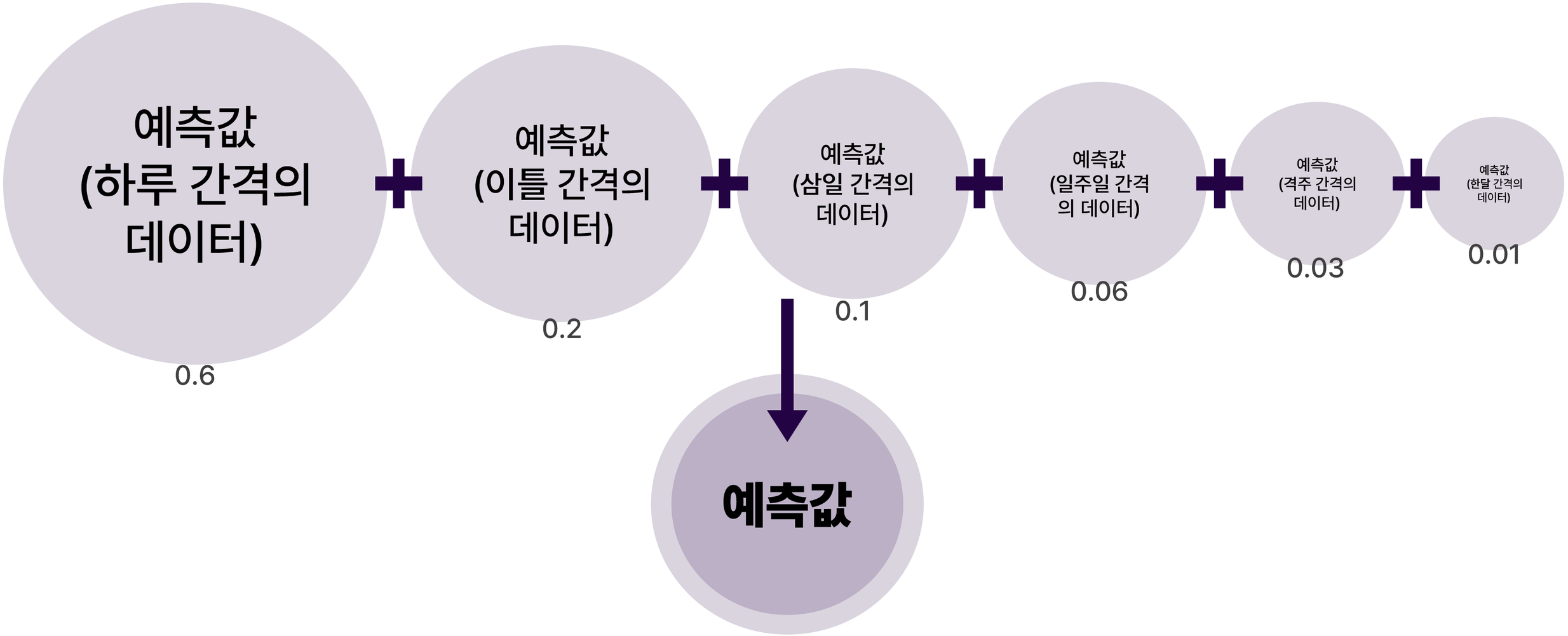
예상한 데이터가 오차가 있을 수 있는 문제 때문.
오차가 있는 데이터를 학습하여 결과값을 만들면 결과값은 더 큰 오차가 생김.

MANY-TO-MANY

-여러 데이터를 학습하여 여러 데이터를 예상하고, 예상한 데이터까지 학습하여 결과값을 만들어 냄.

02.
주가예측모델

주가예측모델


예측값
(하루 간격의
데이터)
0.6
예측값
(이틀 간격의
데이터)
0.2
예측값
(삼일 간격의
데이터)
0.1
예측값
(일주일 간격
의 데이터)
0.06
예측값
(격주 간격의
데이터)
0.03
예측값
(한달 간격의
데이터)
0.01
예측값

주가예측모델

마지막 데이터 이후에 올 값 예측

데이터 = [1, 2, 3, 4, 5, 6, 7, 8, 9]
1 간격의 데이터를 학습하여 예상한 결과 [1,2,3,4,5,6,7,8,9] >>> 9.8
2 간격의 데이터를 학습하여 예상한 결과 [2,4,6,8] >>> 10.6
3 간격의 데이터를 학습하여 예상한 결과 [3,6,9] >>> 9.1

9.8 x 0.7 + 10.6 x 0.2 + 9.1 x +0.1
= 9.89

주가예측모델


2021년 8월
오늘
일
월
화
수
목
금
토
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
광복절
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
하루 간격
이틀 간격
3일 간격
일주일 간격
2주 간격
한달 간격

03.
Python코드

자료출처 - 야후 주식API

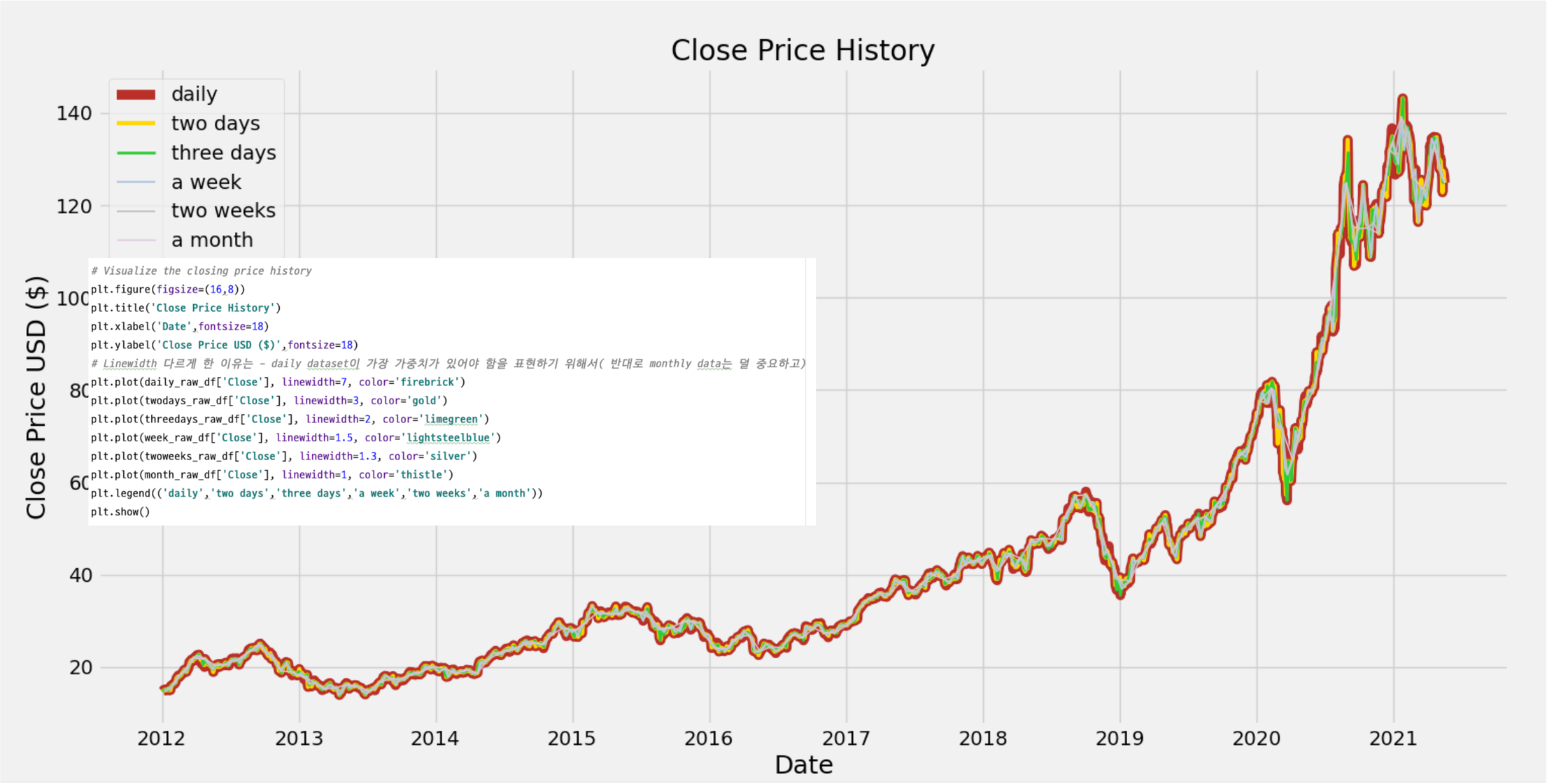
MANY-TO-ONE LSTM MODEL

CREATE THE RAW DATAFRAME.

```
# Get the stock quote
daily_raw_df = web.DataReader('AAPL', data_source='yahoo', start='2012-01-01', end='2021-08-18')
# 2일씩 가는 주가 예측 (dataset)
twodays_raw_df = daily_raw_df[len(daily_raw_df)::-2]
twodays_raw_df = twodays_raw_df[::-1]
# 3일씩 가는 주가 예측 (dataset)
threedays_raw_df = daily_raw_df[len(daily_raw_df)::-3]
threedays_raw_df = threedays_raw_df[::-1]
# 일주일씩 가는 주가 예측 (dataset)
# 장이 주말에는 운영이 안되므로 5일간격으로 dataset 만들어야 함
week_raw_df = daily_raw_df[len(daily_raw_df)::-5]
week_raw_df = week_raw_df[::-1]
# 이주일씩 가는 주가 예측 (dataset)
# 장이 주말에는 운영이 안되므로 10일간격으로 dataset 만들어야 함
twoweeks_raw_df = daily_raw_df[len(daily_raw_df)::-10]
twoweeks_raw_df = twoweeks_raw_df[::-1]
# 한달씩 가는 주가 예측 (dataset)
# 장이 주말에는 운영이 안되므로 20일간격으로 dataset 만들어야 함
month_raw_df = daily_raw_df[len(daily_raw_df)::-20]
month_raw_df = month_raw_df[::-1]
```

MANY-TO-ONE LSTM MODEL

PLOT THE CLOSE PRICE HISTORY.




```
# Visualize the closing price history
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.xlabel('Date',fontsize=18)
plt.ylabel('Close Price USD ($)',fontsize=18)
# Linewidth 다르게 한 이유는 - daily dataset이 가장 가중치가 있어야 함을 표현하기 위해서( 반대로 monthly data는 덜 중요하고)
plt.plot(daily_raw_df['Close'], linewidth=7, color='firebrick')
plt.plot(twodays_raw_df['Close'], linewidth=3, color='gold')
plt.plot(threedays_raw_df['Close'], linewidth=2, color='limegreen')
plt.plot(week_raw_df['Close'], linewidth=1.5, color='lightsteelblue')
plt.plot(twoweeks_raw_df['Close'], linewidth=1.3, color='silver')
plt.plot(month_raw_df['Close'], linewidth=1, color='thistle')
plt.legend(('daily','two days','three days','a week','two weeks','a month'))
plt.show()
```

MANY-TO-ONE LSTM MODEL

CREATE THE TRAIN DATASET.

```
# Create a new dataframe with only the 'Close column
daily_data = daily_raw_df.filter(['Close'])
twodays_data = twodays_raw_df.filter(['Close'])
threedays_data = threedays_raw_df.filter(['Close'])
week_data = week_raw_df.filter(['Close'])
twoweeks_data = twoweeks_raw_df.filter(['Close'])
month_data = month_raw_df.filter(['Close'])
# Convert the dataframe to a numpy array
daily_dataset = daily_data.values
twodays_dataset = twodays_data.values
threedays_dataset = threedays_data.values
week_dataset = week_data.values
twoweeks_dataset = twoweeks_data.values
month_dataset = month_data.values
# Get the number of rows to train the model on
training_data1_len = math.ceil(len(daily_dataset) * .8)
training_data2_len = math.ceil(len(twodays_dataset) * .8)
training_data3_len = math.ceil(len(threedays_dataset) * .8)
training_data7_len = math.ceil(len(week_dataset) * .8)
training_data14_len = math.ceil(len(twoweeks_dataset) * .8)
training_data28_len = math.ceil(len(month_dataset) * .8)
```

TRAIN DATASET

모델을 학습시키기 위한 데이터셋

TEST DATASET

학습에 전혀 관여하지 않고
오직 '최종 성능'을 평가하기 위한 데이터셋

TRAIN DATASET을 RAW DATASET의 80%를 하는 이유

학습세트와 검증세트를 어떤 비율로 분할할지 판단하는 것 중요함.
학습 세트가 너무 작으면 알고리즘이 효과적으로
학습하기에 충분치 않을 수 있고,
검증 데이터가 너무 작으면 정확도, 정밀도, 재현율 등이
서로 차이가 많이 나서 신뢰하기 어려울 수 있기 때문.

MANY-TO-ONE LSTM MODEL

DATA SCALING.

```
# Scale the data
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data1 = scaler.fit_transform(daily_dataset)
scaled_data2 = scaler.fit_transform(twodays_dataset)
scaled_data3 = scaler.fit_transform(threedays_dataset)
scaled_data7 = scaler.fit_transform(week_dataset)
scaled_data14 = scaler.fit_transform(twoweeks_dataset)
scaled_data28 = scaler.fit_transform(month_dataset)


# Create the training data set
# Create the scaled training data set
train_data1 = scaled_data1[0:training_data1_len, :]
train_data2 = scaled_data2[0:training_data2_len, :]
train_data3 = scaled_data3[0:training_data3_len, :]
train_data7 = scaled_data7[0:training_data7_len, :]
train_data14 = scaled_data14[0:training_data14_len, :]
train_data28 = scaled_data28[0:training_data28_len, :]
```

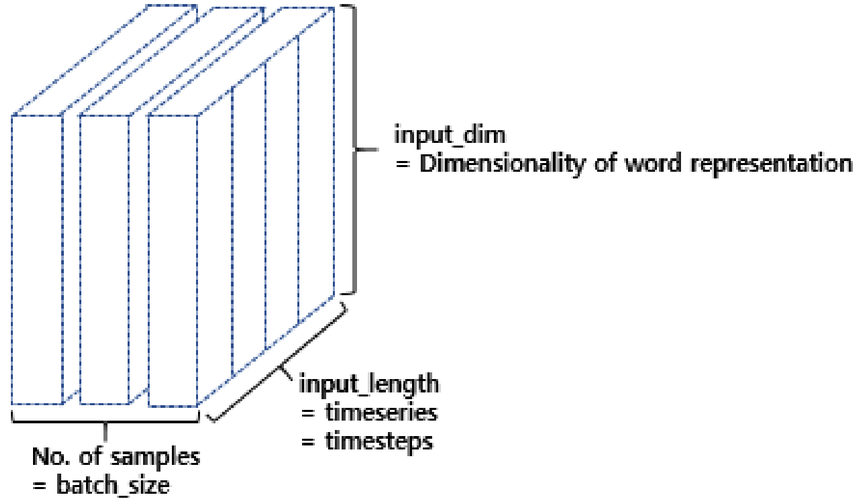
```
# Split the data into x_train and y_train data sets
x1_train = [] # 'x' will be the independent training variables or training features
x2_train = []
x3_train = []
x7_train = []
x14_train = []
x28_train = []
y1_train = [] # 'y' will be the dependent variables or target variables
y2_train = []
y3_train = []
y7_train = []
y14_train = []
y28_train = []


# 참고할(훈련시킬) 데이터 길이 ( 2000개 데이터일 때 60개로 잡은 기준으로 해서 비율 0.03으로 함- reference1기준)
# 데이터 사이즈가 너무 작을 수 있으므로 살짝씩 비율을 높여봄 -> 그냥 그대로 하는게 나은거 같기도 하고
reference1 = math.ceil(0.03*len(train_data1))
reference2 = math.ceil(0.05*len(train_data2))
reference3 = math.ceil(0.07*len(train_data3))
reference7 = math.ceil(0.08*len(train_data7))
reference14 = math.ceil(0.09*len(train_data14))
reference28 = math.ceil(0.1*len(train_data28))
```

MANY-TO-ONE LSTM MODEL

RESHAPE THE DATA.

```
# Convert the x_train and y_train to numpy arrays for the LSTM model
x1_train, y1_train = np.array(x1_train), np.array(y1_train)
x2_train, y2_train = np.array(x2_train), np.array(y2_train)
x3_train, y3_train = np.array(x3_train), np.array(y3_train)
x7_train, y7_train = np.array(x7_train), np.array(y7_train)
x14_train, y14_train = np.array(x14_train), np.array(y14_train)
x28_train, y28_train = np.array(x28_train), np.array(y28_train)
```

```
# Reshape the data ( because LSTM model wants 3-dimensional data)
# print(x_train.shape) example-> result is (1543, 60), 2-dimension
x1_train = np.reshape(x1_train,(x1_train.shape[0],x1_train.shape[1],1))
x2_train = np.reshape(x2_train,(x2_train.shape[0],x2_train.shape[1],1))
x3_train = np.reshape(x3_train,(x3_train.shape[0],x3_train.shape[1],1))
x7_train = np.reshape(x7_train,(x7_train.shape[0],x7_train.shape[1],1))
x14_train = np.reshape(x14_train,(x14_train.shape[0],x14_train.shape[1],1))
x28_train = np.reshape(x28_train,(x28_train.shape[0],x28_train.shape[1],1))
# After reshaping, print(x_train.shape) example -> result is (1543, 60, 1), 3-dimension
# And to make the code robust, it is better to write the code instead of the number
# for example, x_train.shape[0] instead of 1543
```



DATA RESHAPING

LSTM 모델은 3차원의 텐서를 입력으로 받아야하므로 2차원의 데이터를 3차원의 데이터로 바꿈

MANY-TO-ONE LSTM MODEL

LSTM MODEL (ARCHITECTURE)

```
# Build the LSTM model (Architecture)
model1, model2, model3, model7, model14, model28 = \
    Sequential(), Sequential(), Sequential(), Sequential(), Sequential(), Sequential()
model1.add(LSTM(50, return_sequences=True, input_shape= (x1_train.shape[1],1)))
model2.add(LSTM(50, return_sequences=True, input_shape= (x2_train.shape[1],1)))
model3.add(LSTM(50, return_sequences=True, input_shape= (x3_train.shape[1],1)))
model7.add(LSTM(50, return_sequences=True, input_shape= (x7_train.shape[1],1)))
model14.add(LSTM(50, return_sequences=True, input_shape= (x14_train.shape[1],1)))
model28.add(LSTM(50, return_sequences=True, input_shape= (x28_train.shape[1],1)))
# Add second LSTM
model1.add(LSTM(50, return_sequences = False))
model2.add(LSTM(50, return_sequences = False))
model3.add(LSTM(50, return_sequences = False))
model7.add(LSTM(50, return_sequences = False))
model14.add(LSTM(50, return_sequences = False))
model28.add(LSTM(50, return_sequences = False))
```

```
# Add 25neurons Dense layer, and this is your regularly densely connected
# neural network layer with 25 neurons
model1.add(Dense(25))
model2.add(Dense(25))
model3.add(Dense(25))
model7.add(Dense(25))
model14.add(Dense(25))
model28.add(Dense(25))

model1.add(Dense(1))
model2.add(Dense(1))
model3.add(Dense(1))
model7.add(Dense(1))
model14.add(Dense(1))
model28.add(Dense(1))
```

LSTM 1st Layer : 50개의 뉴런
LSTM 2nd Layer : 50개의 뉴런
Dense 1st Layer : 25개의 뉴런
Dense 2nd Layer : 1개의 뉴런

MANY-TO-ONE LSTM MODEL

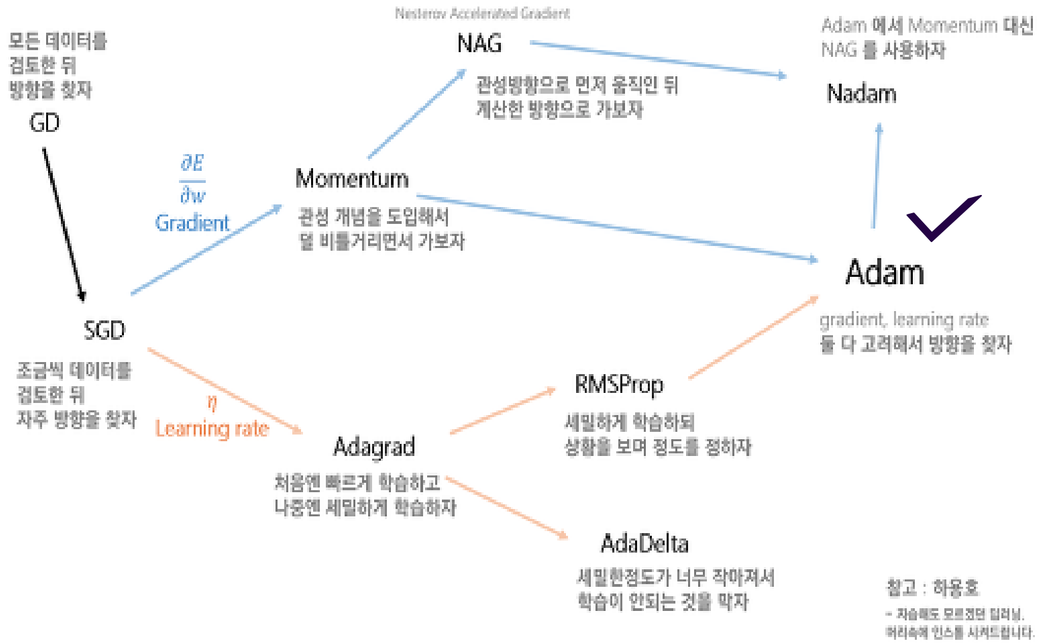
OPTIMIZER MODEL (ADAM - ADAPTIVE MOMENT ESTIMATION)

```
# Compile the model
model1.compile(optimizer='adam', loss='mean_squared_error')
model2.compile(optimizer='adam', loss='mean_squared_error')
model3.compile(optimizer='adam', loss='mean_squared_error')
model7.compile(optimizer='adam', loss='mean_squared_error')
model14.compile(optimizer='adam', loss='mean_squared_error')
model28.compile(optimizer='adam', loss='mean_squared_error')
# Optimization Algorithm
```

BATCH SIZE AND EPOCHS

```
# Train the model
model1.fit(x1_train, y1_train, batch_size=1, epochs=3)
model2.fit(x2_train, y2_train, batch_size=1, epochs=3)
model3.fit(x3_train, y3_train, batch_size=1, epochs=3)
model7.fit(x7_train, y7_train, batch_size=1, epochs=3)
model14.fit(x14_train, y14_train, batch_size=1, epochs=3)
model28.fit(x28_train, y28_train, batch_size=1, epochs=3)
```

Optimizer 발전 과정



MODEL에 사용된 개념.

EPOCH

전체 데이터 셋에 대해 한번의 학습과정이 완료 된것
ex) epoch=40 -> 이건 전체 데이터를 40번 사용해서 학습하는 것

BATCH SIZE

한번의 epoch에서 모든 데이터를 한꺼번에 학습시킬 수 없음,
이때 몇번 나누어서 주는가 iteration, 각 interation 마다 주는 데이터 사이즈를 batch size 라고함

1 Epoch : 모든 데이터 셋을 한 번 학습
1 iteration : 1회 학습

minibatch : 데이터 셋을 batch size 크기로 쪼개서 학습

```
reference1 = math.ceil(0.03*len(train_data1))
reference2 = math.ceil(0.05*len(train_data2))
reference3 = math.ceil(0.07*len(train_data3))
reference7 = math.ceil(0.08*len(train_data7))
reference14 = math.ceil(0.09*len(train_data14))
reference28 = math.ceil(0.1*len(train_data28))
```

04.
결과

적용한 변수들과 최적의 모델

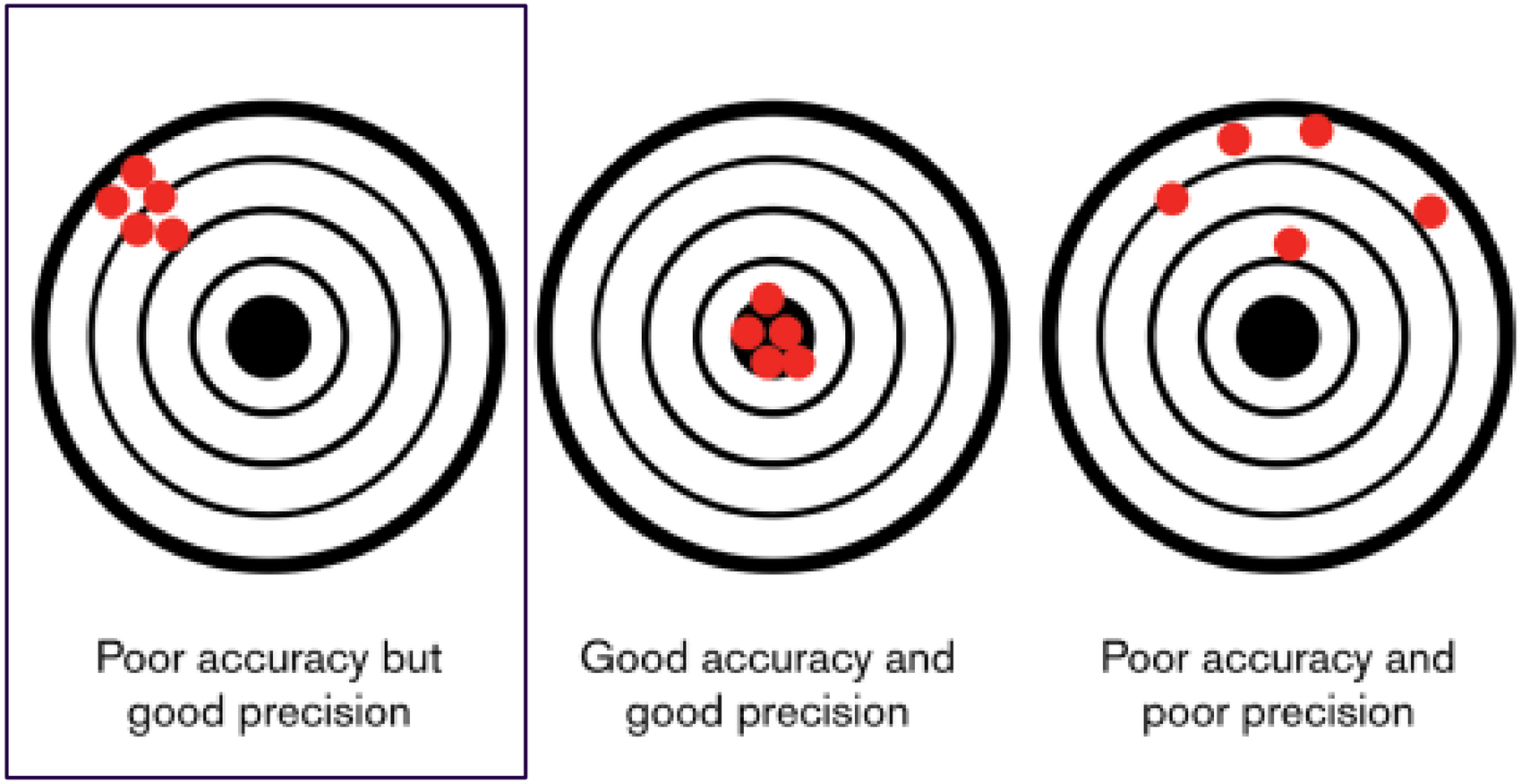
```
reference1 = math.ceil(0.03*len(train_data1))
reference2 = math.ceil(0.05*len(train_data2))
reference3 = math.ceil(0.07*len(train_data3))
reference7 = math.ceil(0.08*len(train_data7))
reference14 = math.ceil(0.09*len(train_data14))
reference28 = math.ceil(0.1*len(train_data28))
```

```
# Weight 지정
print("="*23 + " 2021.08.19 PRICE RESULT " + "="*23)
print(0.6*pred_price1 + \
      0.2*pred_price2 + \
      0.1*pred_price3 + \
      0.06*pred_price7 + \
      0.03*pred_price14 + \
      0.01*pred_price28)
```

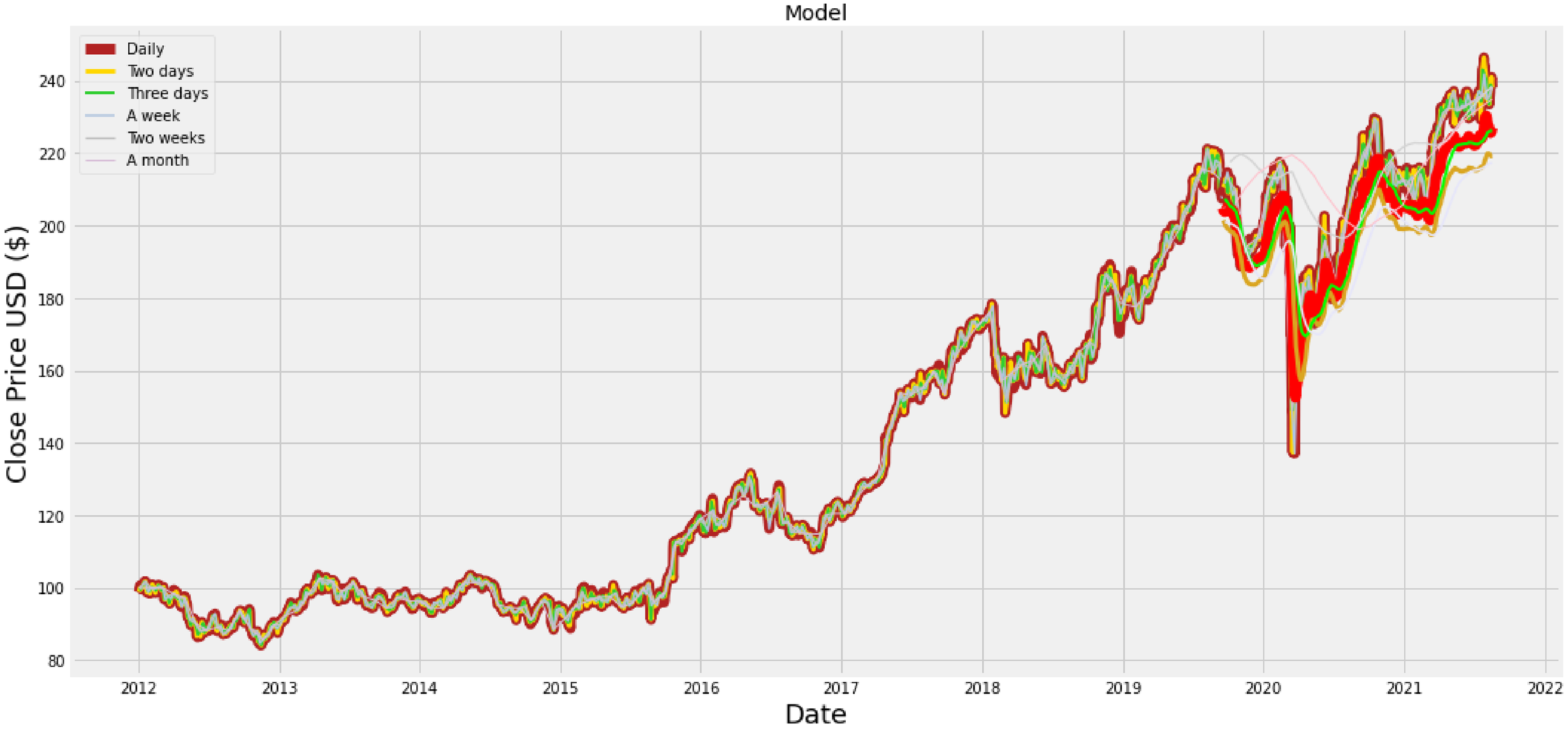
```
# Build the LSTM model (Architecture)
model1, model2, model3, model7, model14, model28 = \
    Sequential(), Sequential(), Sequential(), Sequential(), Sequential(), Sequential()
model1.add(LSTM(50, return_sequences=True, input_shape=(x1_train.shape[1],1)))
model2.add(LSTM(50, return_sequences=True, input_shape=(x2_train.shape[1],1)))
model3.add(LSTM(50, return_sequences=True, input_shape=(x3_train.shape[1],1)))
model7.add(LSTM(50, return_sequences=True, input_shape=(x7_train.shape[1],1)))
model14.add(LSTM(50, return_sequences=True, input_shape=(x14_train.shape[1],1)))
model28.add(LSTM(50, return_sequences=True, input_shape=(x28_train.shape[1],1)))
# Add second LSTM
model1.add(LSTM(50, return_sequences = False))
model2.add(LSTM(50, return_sequences = False))
model3.add(LSTM(50, return_sequences = False))
model7.add(LSTM(50, return_sequences = False))
model14.add(LSTM(50, return_sequences = False))
model28.add(LSTM(50, return_sequences = False))
```

```
# Train the model
model1.fit(x1_train, y1_train, batch_size=1, epochs=1)
model2.fit(x2_train, y2_train, batch_size=1, epochs=1)
model3.fit(x3_train, y3_train, batch_size=1, epochs=1)
model7.fit(x7_train, y7_train, batch_size=1, epochs=1)
model14.fit(x14_train, y14_train, batch_size=1, epochs=1)
model28.fit(x28_train, y28_train, batch_size=1, epochs=1)
```

적용한 변수들과 최적의 모델.



결과 MCDONALD INC.

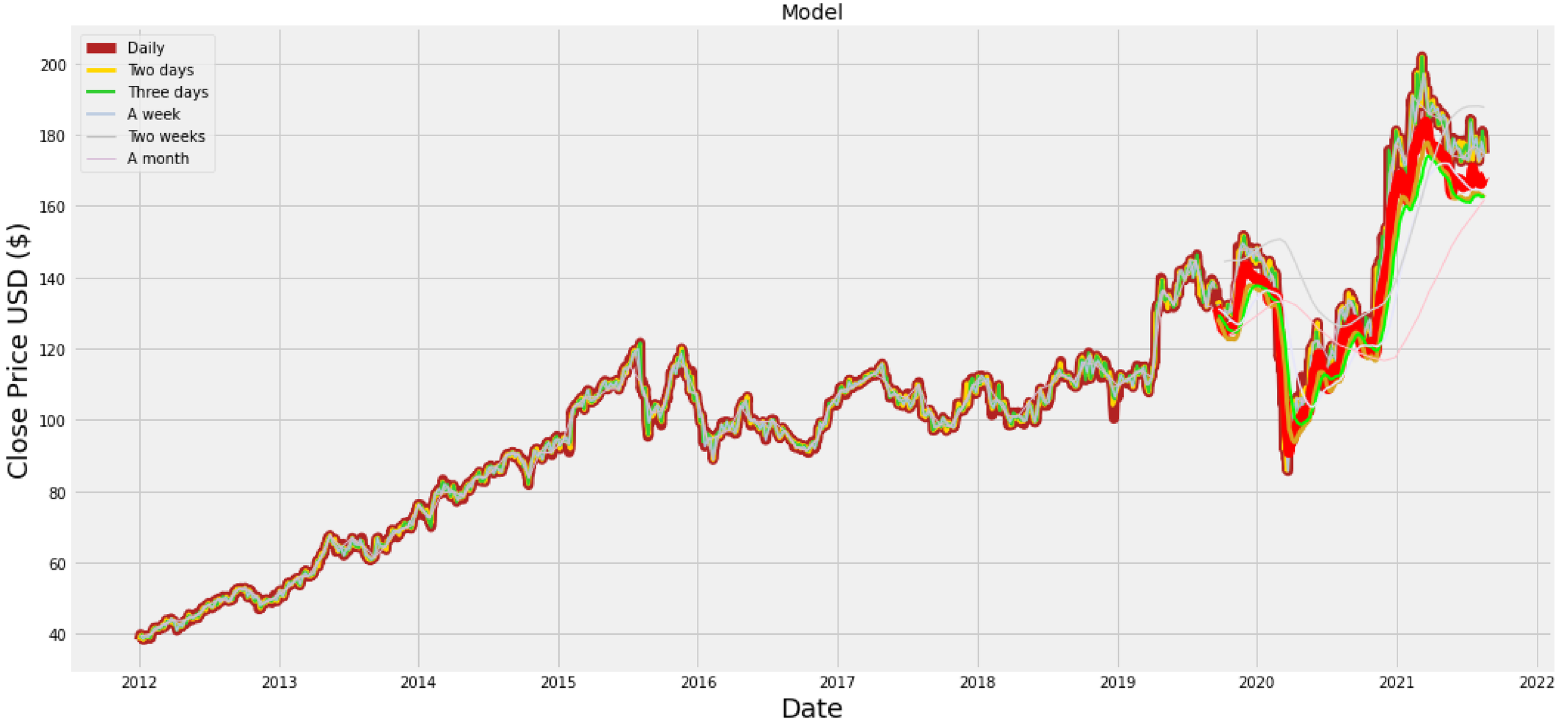


결과 MCDONALD INC.

Date	Open	High	Low	Close*	Adj Close**	Volume
Aug 20, 2021	236.96	239.39	236.32	238.49	238.49	1,989,700
Aug 19, 2021	236.68	238.12	236.07	237.23	237.23	1,455,400
Aug 18, 2021	239.44	239.76	237.78	238.08	238.08	2,215,500

```
======================== RMSE RESULTS (Daily TwoDays ThreeDays Weekly TwoWeeks Monthly) ========================
7.1656690234980305 13.436088057588894 7.9707571467997866 14.152908325195312 6.544777552286784 5.454089482625325
======================== PREDICTED PRICES RESULTS (Daily TwoDays ThreeDays Weekly TwoWeeks Monthly) ========================
[[234.1109]] [[225.30945]] [[230.39035]] [[220.43523]] [[239.74425]] [[238.26064]]
======================== 2021.08.19 PRICE RESULT ========================
[[231.36852]]
```

결과 THE WALT DISNEY COMPANY.

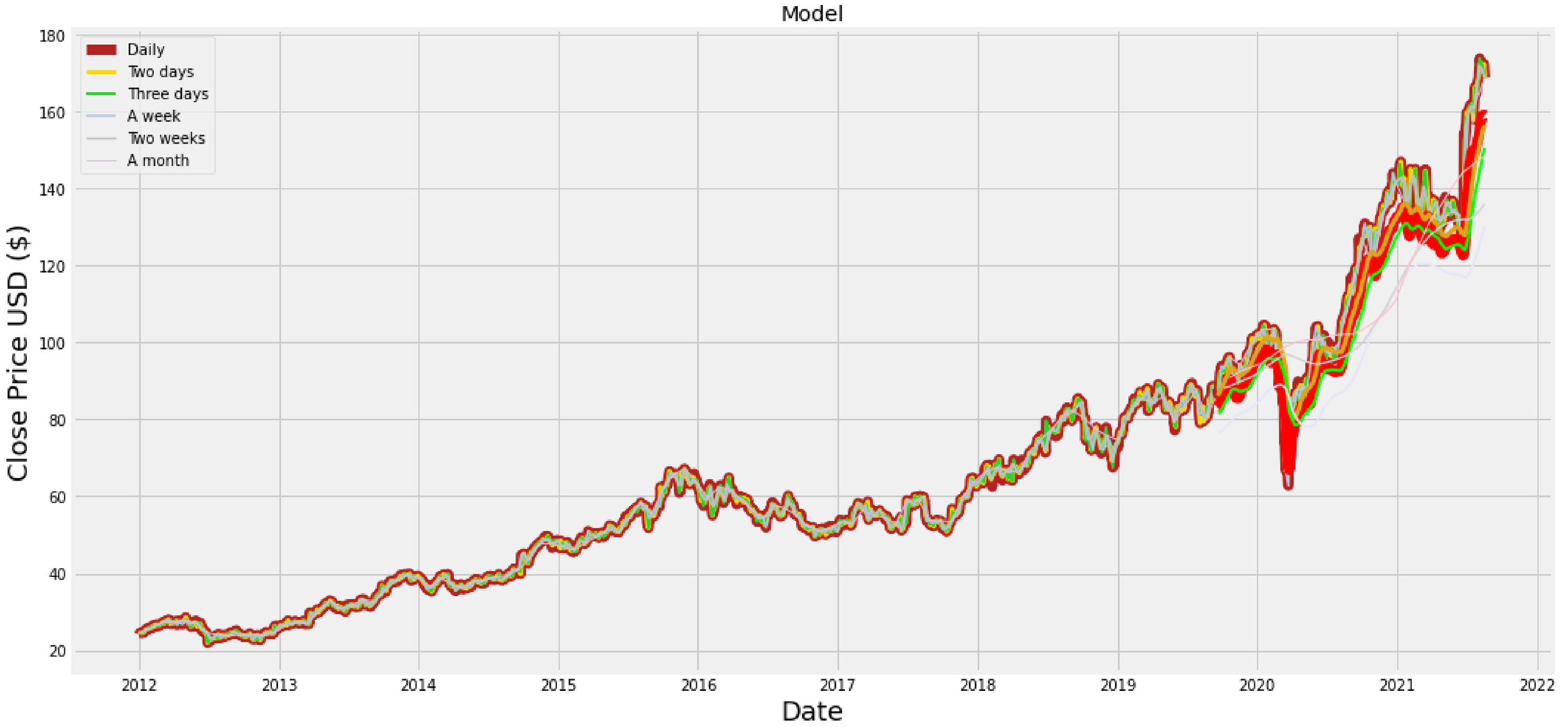


결과 THE WALT DISNEY COMPANY.

Date	Open	High	Low	Close*	Adj Close**	Volume
Aug 20, 2021	173.00	175.21	172.65	175.12	175.12	6,230,800
Aug 19, 2021	174.00	174.68	172.56	173.25	173.25	8,478,600
Aug 18, 2021	175.90	176.79	174.50	174.74	174.74	7,063,500

```
====================== RMSE RESULTS (Daily TwoDays ThreeDays Weekly TwoWeeks Monthly) ======================
6.9516594469054676 10.630933651254196 10.592782393745754 10.580497033817252 3.3149129549662 27 15.47396977742513
====================== PREDICTED PRICES RESULTS (Daily TwoDays ThreeDays Weekly TwoWeeks Monthly) ======================
[[178.50356]] [[173.15596]] [[171.88]] [[169.98701]] [[194.13817]] [[165.17575]]
====================== 2021.08.19 PRICE RESULT ======================
[[176.59645]]
```

결과 NIKE INC.


Model
Daily
Two days
Three days
A week
Two weeks
A month
Close Price USD ($)
180
160
140
120
100
80
60
40
20
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
Date

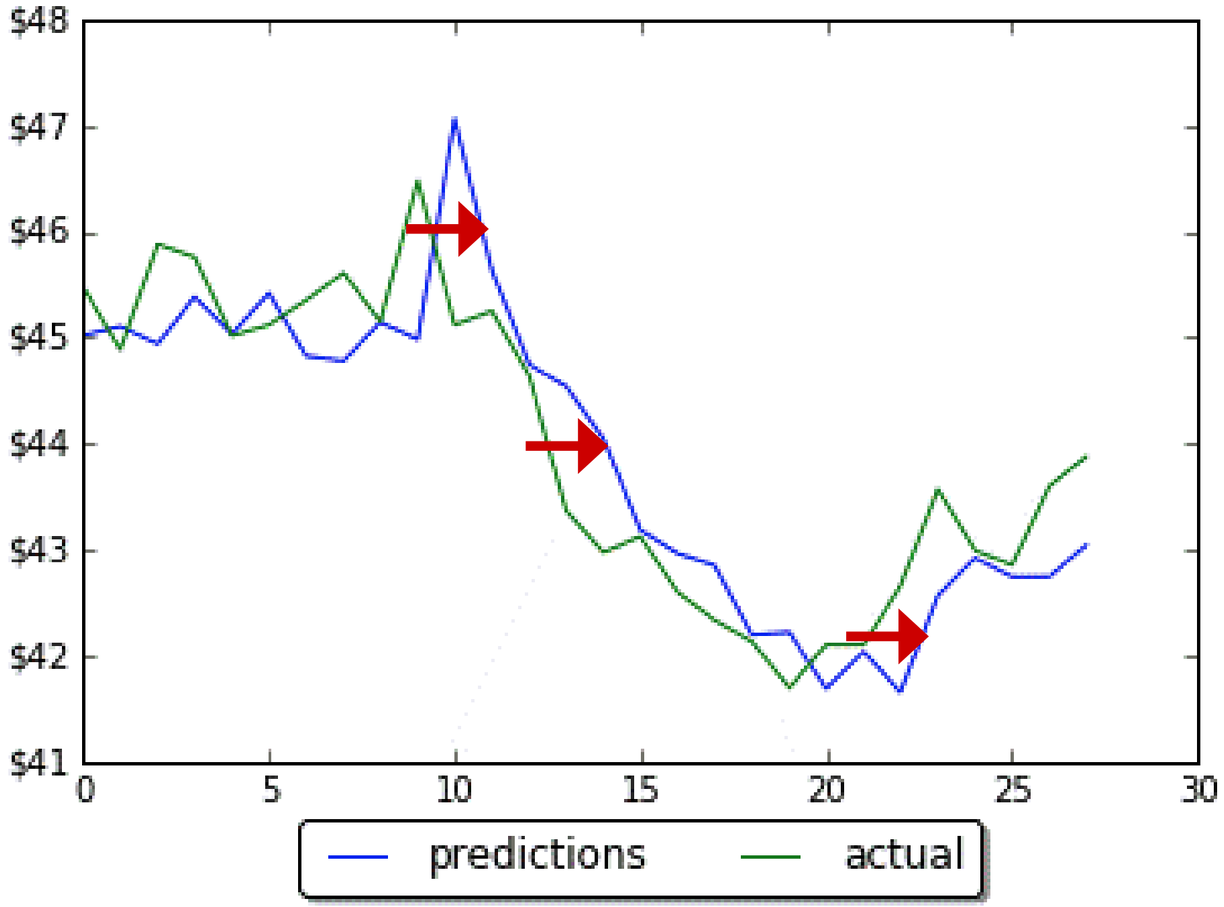
결과 NIKE INC.

Date	Open	High	Low	Close*	Adj Close**	Volume
Aug 20, 2021	166.27	168.01	165.42	167.79	167.79	4,039,100
Aug 19, 2021	166.96	167.54	164.37	165.59	165.59	7,336,300
Aug 18, 2021	169.75	171.80	168.66	168.81	168.81	5,487,400

```
====================== RMSE RESULTS (Daily TwoDays ThreeDays Weekly TwoWeeks Monthly) ======================
7.330566288025911 4.489997044082515 9.636790020865684 18.202012878103353 9.851477146148682 4.707642873128255
====================== PREDICTED PRICES RESULTS (Daily TwoDays ThreeDays Weekly TwoWeeks Monthly) ======================
[[163.16919]] [[160.13438]] [[155.57504]] [[134.19481]] [[140.0339]] [[152.43188]]
====================== 2021.08.19 PRICE RESULT ======================
[[159.26294]]
```

05.
한계점

모델의 한계.



LSTM을 이용한 예측이 Lagging 되는 현상

THANK YOU.

신준재 김주원 안정모 정서현