테슬라 주가 예측

[Tesla Stock Price Prediction]

20181082259 박명준

목 차

2018108259 박 명 준

01. 개요

2018108259 박명준

* 개요

: 지난 주가를 바탕으로 주식의

종가(close price)를 예측하는 것을 목표로 함

02. 필요성

20181082259 박명준

* 필요성

테슬라는 미국의 글로벌 IT 기업으로 전기차 부문이 주력 사업이지만 AI를 필두로한 소프트웨어 분야도 업계 최상위권으로 평가 받는 기업이다. 재작년과 작년을 통해 테슬라 주식은 폭등하고, 2021년에는 페이스북의 시가총액을 넘기까지 하며 대중의 관심을 끌었다.

주식에는 이렇게 변동적인 부분이 많아 주식의 등락을 예측하기 힘들다. 하지만 이런 예측을 통해 정확하진 않을지라도 근사값을 얻을 수 있다면, 주식 이용자들의 편의와 여러 방면에서 효과를 볼 수 있다.

03. 사용 라이브러리

2018102859 박명준

사용 라이브러리

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

1. Numpy : 행렬이나 일반적으로 대규모 다차원 배열을 쉽게 처리할 수 있도록 지원하는 파이썬 라이브러리

2. Pandas : 데이터 조작 및 분석을 위한 Pythin 프로그래밍 언어 용으로 작성된 라이브러리

3. matplotlib : 다양한 데이터를 많은 방법으로 도식화 할 수 있도록 하는 파이썬 라이브러리

사용 라이브러리

```
import matplotlib.pyplot as plt
import pandas_datareader as web
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import math
```

```
Using TensorFlow backend.
```

4. Scikit-Learn : Python 프로그래밍 언어용 기계 학습 파이썬 라이브러리

5. TensorFlow : 다양한 작업에 대해 데이터 흐름 프로그래밍을 위한 오픈소스 소프트웨어 라이브러리이며, 인공 신경망같은 기계 학습 응용프로그램 및 딥러닝(deep Learning)에도 사용된다.

TensorFlow

6. Keras : 케라스(Keras)는 파이썬으로 작성된 오픈 소스 신경망 라이브러리

Keras

04. 사용 데이터 셋 설명

2018108259 박명준

input data
파일 경로 찾기

import os

- os를 연결

os.walk

- 파일의 위치(경로)를 찾는 것
- kaggle/input/.../Tesla.csv

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```
/kaggle/input/tesla-stock-price/Tesla.csv - Tesla.csv.csv
```

사용
데이터 셋 설명

pd.read_csv

: 데이터 가져오기(데이터 import)

.info()

: 칼럼의 정보(이름, 행, 열, 타입 등) 확인

```
df = pd.read_csv('/kaggle/input/tesla-stock-price/Tesla.csv - Tesla.csv.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1692 entries, 0 to 1691
Data columns (total 7 columns):
Date         1692 non-null object
Open         1692 non-null float64
High         1692 non-null float64
Low          1692 non-null float64
Close        1692 non-null float64
Volume       1692 non-null int64
Adj Close    1692 non-null float64
dtypes: float64(5), int64(1), object(1)
memory usage: 92.7+ KB
```

사용
데이터 셋 설명

df.head()

: 불러온 데이터의 상위 5개 행을 출력

Date : 날짜

Open : 시작 주가

High : 고점

Low : 저점

Close : 종가(종료 주가)

Volume : 거래량

Adj Close : 장이 마감 된 후 조정 종가

```
df.head()
```

:

	Date	Open	High	Low	Close	Volume	Adj Close
0	6/29/2010	19.000000	25.00	17.540001	23.889999	18766300	23.889999
1	6/30/2010	25.790001	30.42	23.299999	23.830000	17187100	23.830000
2	7/1/2010	25.000000	25.92	20.270000	21.959999	8218800	21.959999
3	7/2/2010	23.000000	23.10	18.709999	19.200001	5139800	19.200001
4	7/6/2010	20.000000	20.00	15.830000	16.110001	6866900	16.110001

05. 데이터 전처리

2018108259 박명준

시계열
자료형 변환

판다스에서 시계열 자료형은

Timestamp가 있음

불러온 데이터가 날짜형 데이터인 것을 확인하였으며 Timestamp로 변환

pd.to_datetime() : 날짜형태의 자료형을 시계열 타입(Timestamp / datetime)으로 변환해준다.

```
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date',inplace=True)
```

튜플로 반환

df.shape : 행과 열의 개수를 튜플로 반환

1692행과 6열 튜플로 반환된 것을 확인 할 수 있다.

```
df.shape #1692 rows and 7 columns that the data frame have
```

```
(1692, 6)
```

06. 시각화

2018108259 박명준

시각화

plot(plt) : 일반적으로 둘 이상의 변수 간의 관계를 보여주는 그래프로 데이터를 나타내는 그래픽 기술

plt.figure(figsize)로 그래프 사이즈를 (16,8)로 설정

title : Close Price History(역대 종가)

```
#plotting the data
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['Close'], color='red')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD', fontsize = 18)
plt.show()
```

시각화

---

plt.plot(df['Close'], color = 'red')

- 데이터 프레임의 Close(종가) red 그래프로 변환

xlabel은 Date(날짜), ylavel은 Close Price USD(종가 USD)로 적어주며 fontsize는 18로 통일

plt.show : 그래프 보여주기

```
#plotting the data
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['Close'], color='red')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD', fontsize = 18)
plt.show()
```

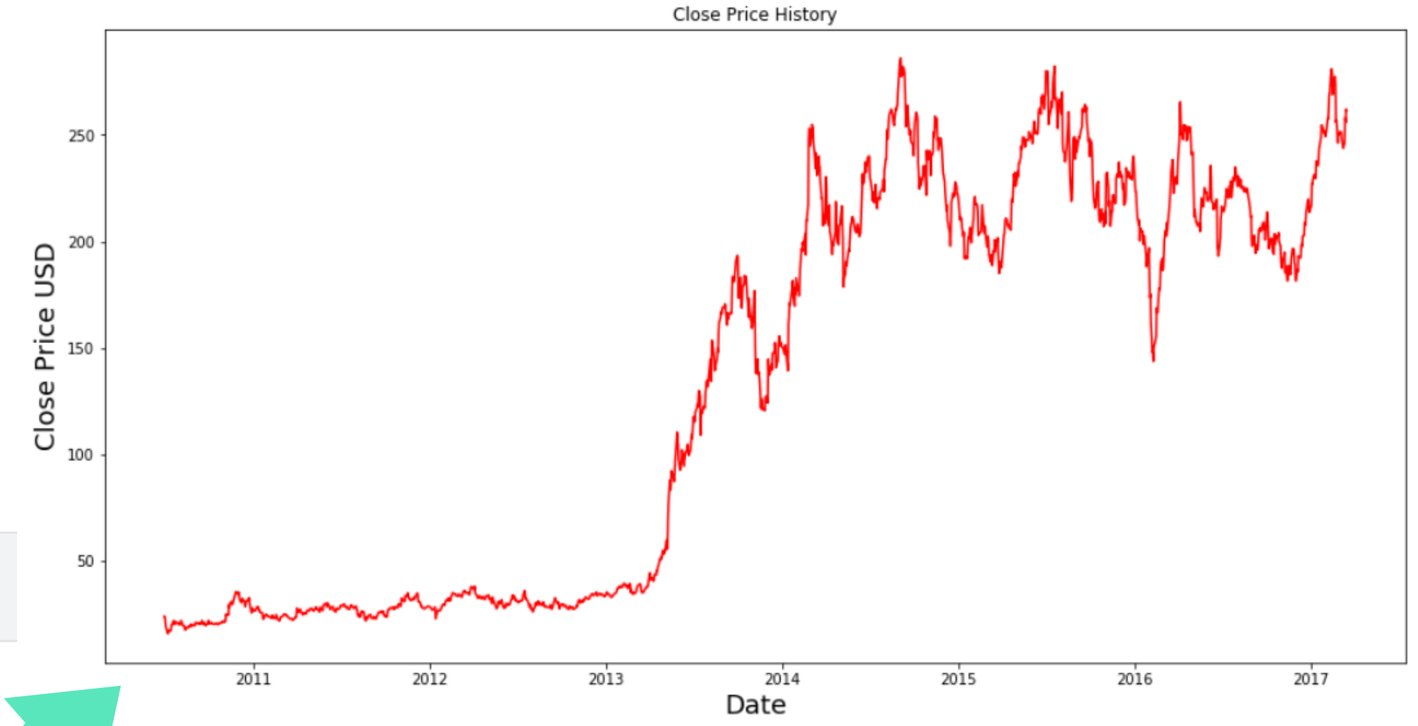
시각화

head로 본 상위 5개가 앞에 있는 것을 확인 가능

csv파일 값들의 그래프 시각화

```
df.head()
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	6/29/2010	19.000000	25.00	17.540001	23.889999	18766300	23.889999
1	6/30/2010	25.790001	30.42	23.299999	23.830000	17187100	23.83
2	7/1/2010	25.000000	25.92	20.270000	21.959999	8218800	21.959999
3	7/2/2010	23.000000	23.10	18.709999	19.200001	5139800	19.200001
4	7/6/2010	20.000000	20.00	15.830000	16.110001	6866900	16.110001



07. 학습 및 테스트

2018108259 박명준

모델 훈련(학습)

먼저 'Close'(종가) 열만 사용하여 새로운 데이터 프레임(df) 생성

데이터 프레임을 numpy array로 변환

모형을 훈련(학습)할 행의 수를 나타냄

```
# create a new data frame with only 'Close column'
data = df.filter(['Close'])
dataset = data.values #convert the data frame to a numpy array
training_data_len = math.ceil(len(dataset)*.8)  # number of rows to train the model on
training_data_len
```

```
1354
```

데이터 정규화

Scale은 데이터 전처리 과정 중 하나로, Scaling을 해주는 이유는 데이터 값이 너무 크거나 작은 경우 모델 학습 과정 중 0으로 수렴하거나 무한으로 발산해버릴 수 있음

그러므로 모델 훈련 데이터를 scale 시켜주고 있다.

```
#scale the data
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)
scaled_data
```

```
array([[0.02993635],
       [0.02971433],
       [0.02279455],
       ...,
       [0.88784039],
       [0.91122698],
       [0.9091918 ]])
```

데이터 정규화

2016년의 데이터까지 데이터를 훈련(학습) 시킨다.

```
#create the training dataset
#create the scaled training dataset

train_data = scaled_data[0:training_data_len, :]
#Split the data into x_train, y_train datasets
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<=60:
        print(x_train)
        print(y_train)
        print()
```

```
[array([0.02993635, 0.02971433, 0.02279455, 0.01258141, 0.00114713,
       0.        , 0.00614268, 0.00592066, 0.00462551, 0.00865897,
       0.01494967, 0.01513469, 0.01791   , 0.02260953, 0.01665186,
       0.01635583, 0.01924215, 0.02031528, 0.01905714, 0.01757696,
       0.01820603, 0.01683689, 0.01531972, 0.01894612, 0.02275755,
       0.02020426, 0.01720693, 0.01402457, 0.01406157, 0.01195234,
       0.00777087, 0.00666075, 0.00932504, 0.01102724, 0.01239639,
       0.01099023, 0.01106424, 0.01221137, 0.01602279, 0.01258141,
       0.0151717 , 0.01461664, 0.01443162, 0.01506069, 0.01361752,
       0.01720693, 0.01946418, 0.01942717, 0.01753997, 0.01887211,
       0.01816903, 0.01617081, 0.01820603, 0.01968621, 0.02286856,
       0.01902013, 0.01639284, 0.01946418, 0.01839106, 0.01506069])]
[0.01391355415474397]
```

reshape

x_train과 y_train을 numpy arr

ay로 변환

그리고 데이터를 바꾸지 않고

새롭게 shape을 형성

```
#convert the x_train and y_train  to numppy array
x_train,y_train = np.array(x_train), np.array(y_train)
```

```
#reshape the data
x_train = np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))
x_train.shape
```

```
(1294, 60, 1)
```

LSTM
Model

```
#Buil the LSTM model
model =Sequential()
model.add(LSTM(64,return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(64, return_sequences= False))
model.add(Dense(32))
model.add(Dense(1))
```

```
#Complie the model
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
#Train the model
model.fit(x_train,y_train, batch_size=1, epochs=10)
```

```
<keras.callbacks.callbacks.History at 0x7fcf282b5ba8>
```

LSTM

Model

LSTM은 Long Short-Term Me

mory로 장단기 메모리로 불림

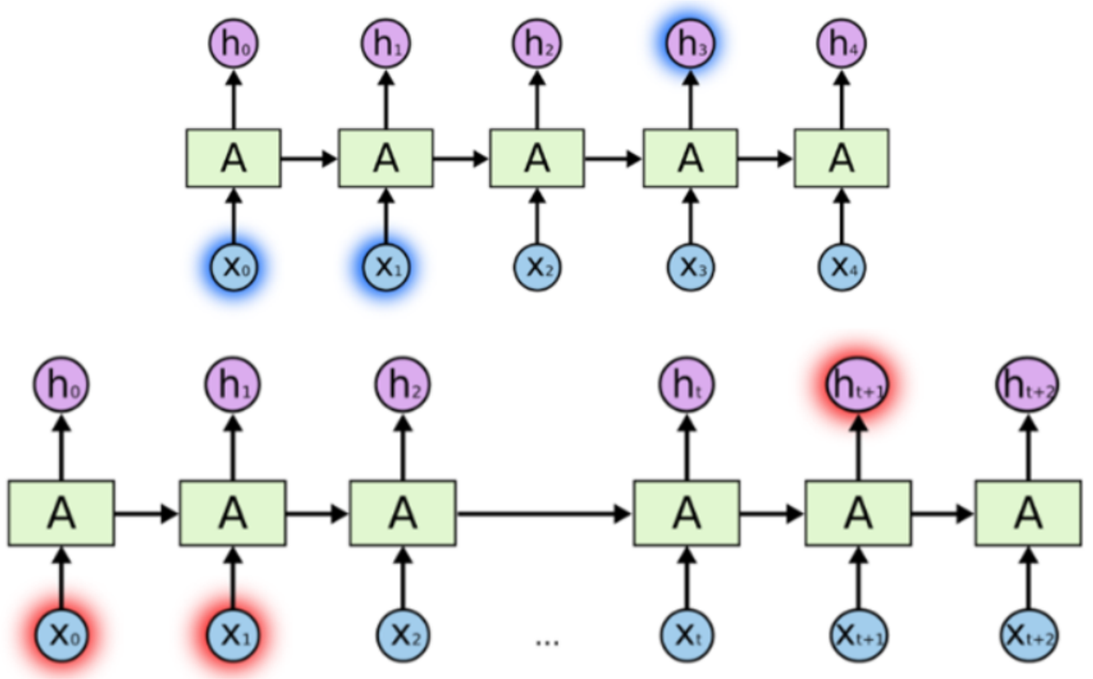
오랜 기간 동안 정보를 기억하는

일에서 특별한 작업 없이도 기본

적으로 취하게 되는 특성

케라스에서 LSTM의 라이브러리

를 가져다 씀



LSTM Model 구축 및 훈련

여러개의 LSTM 셀로 layer를 여러층 쌓아 모델 구현

.add()를 통해 쉽게 layer 추가 가능

.compile은 학습 전, 학습방식에 대한 환경설정 / 컴파일 아담 최적화는 1차 및 2차 모멘트의 적응적 추정을 기반으로 하는 확률적 경사 하강법

이것을 토대로 이제 학습을 시키는데

1개의 샘플로 10번 반복 학습

```
#Buil the LSTM model
model =Sequential()
model.add(LSTM(64,return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(64, return_sequences= False))
model.add(Dense(32))
model.add(Dense(1))
```

```
#Complie the model
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
#Train the model
model.fit(x_train,y_train, batch_size=1, epochs=10)
```

```
<keras.callbacks.callbacks.History at 0x7fcf282b5ba8>
```

예측 테스트 및 시각화

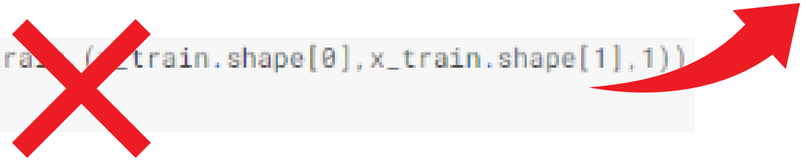
다시 데이터 셋을 생성하고 데이터를 n

umpy array로 반환 후

이번엔 train.shape이 아닌 test.shape

를 reshape해주며 테스트 준비

```
x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
x_train.shape
```



```
test_data= scaled_data[training_data_len-60:, :]
#create the data sets x_test and y_test
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])
```

```
#convert the data to a numpy array
x_test = np.array(x_test)
```

```
#reshape the data
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1],1))
x_test.shape
```

```
(338, 60, 1)
```

예측 테스트 및 시각화

예측 값 데이터 설정

.sqrt() : 제곱근 계산

rmse :root mean square error의 약어로, 평균 제곱 오차 함수에서 얻은 값의 제곱근

데이터를 표시해주고(train, valid) 예측한 것과 함께 시각화

```
#predicting the data
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
```

```
#get the root mean square error(RMSE)
rmse = np.sqrt(np.mean(predictions - y_test)**2)
rmse
```

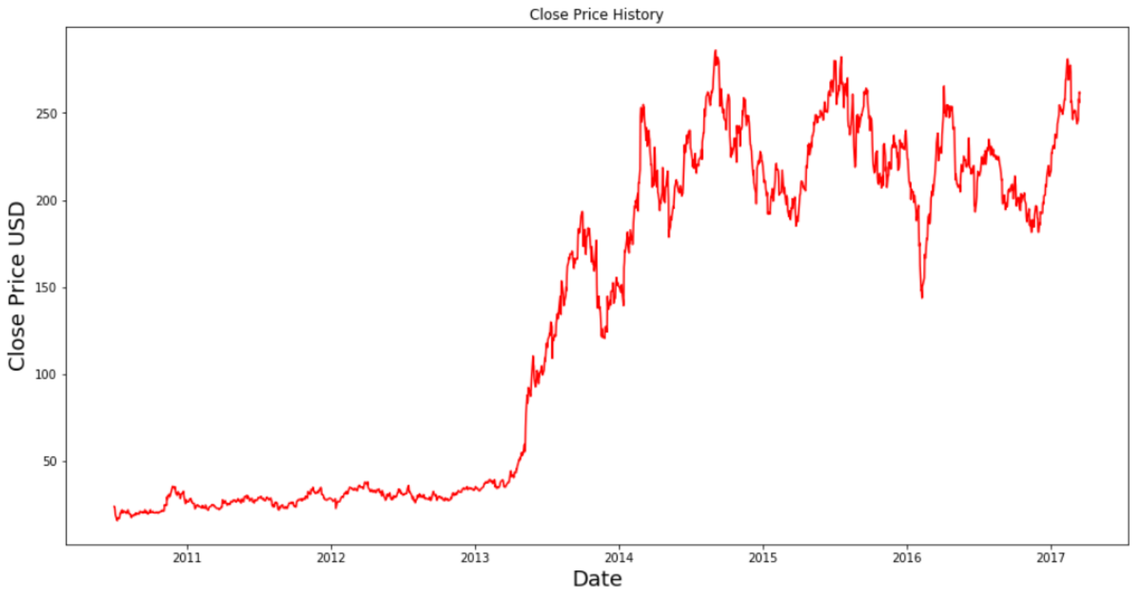
```
1.8063661957112147
```

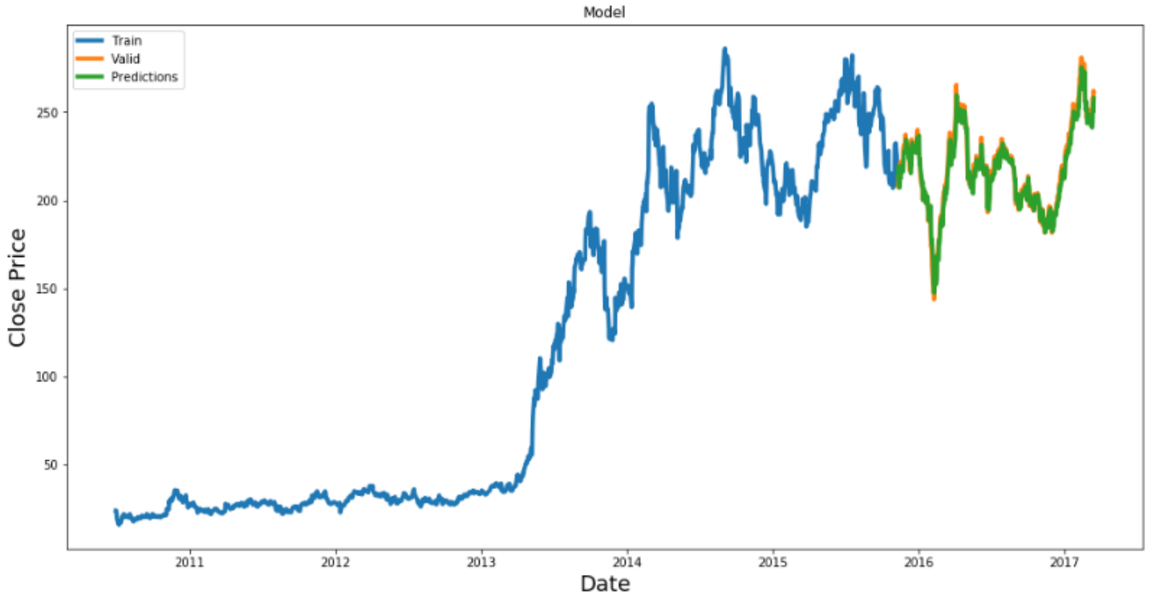
* rmse를 사용하면 모델 매개변수의 예상 값과 실제 값 사이의 차이를 쉽게 그릴 수 있고, 이를 통해 모델의 효율성을 명확하게 판단할 수 있음

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

```
#plot the data
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions
#Visialization the data
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price' ,fontsize=18)
plt.plot(train['Close'],linewidth=3.5)
plt.plot(valid[['Close','Predictions']],linewidth=3.5)
plt.legend(['Train','Valid','Predictions'], loc='upper_center')
```

예측 테스트 및 시각화


Close Price History
Close Price USD
250
200
150
100
50
2011
2012
2013
2014
2015
2016
2017
Date

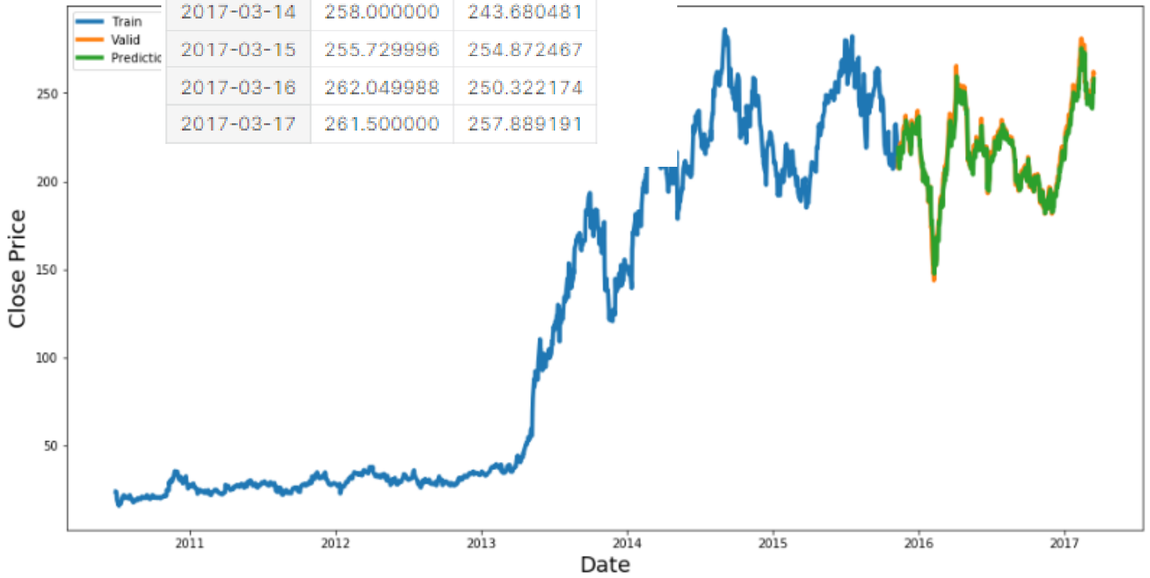

Model
Train
Valid
Predictions
Close Price
250
200
150
100
50
2011
2012
2013
2014
2015
2016
2017
Date

예측 테스트 및 시각화

이렇게 시각화 해보고 예측된 값을 원래 주식 종가와 비교해 봤을때 약간의 차이는 있지만, 거대한 차이는 없음

```
#show the valid and predicted price
valid
```

Date	Close	Predictions
2015-11-12	212.940002	218.829056
2015-11-13	207.190002	211.899506
2015-11-16	214.309998	207.144943
2015-11-17	214.000000	214.404114
2015-11-18	221.070007	212.449890
...	...	...
2017-03-13	246.169998	241.198273
2017-03-14	258.000000	243.680481
2017-03-15	255.729996	254.872467
2017-03-16	262.049988	250.322174
2017-03-17	261.500000	257.889191



08. 소감

2018108259 박명준

* 소감

: 파이썬과 캐글을 이번 수업으로 처음 접해보고 예측하는 모델 설계같은 과제도 처음 겪어봐서 많이 어려웠지만 이 과제를 하며 자료 조사하면서 딥러닝, 파이썬 등 에 대해 많이 알게 되었고, 주가 예측 프로그래밍이 어려운거구나를 새삼 느꼈습니다.

THANK YOU

2018108259 박명준