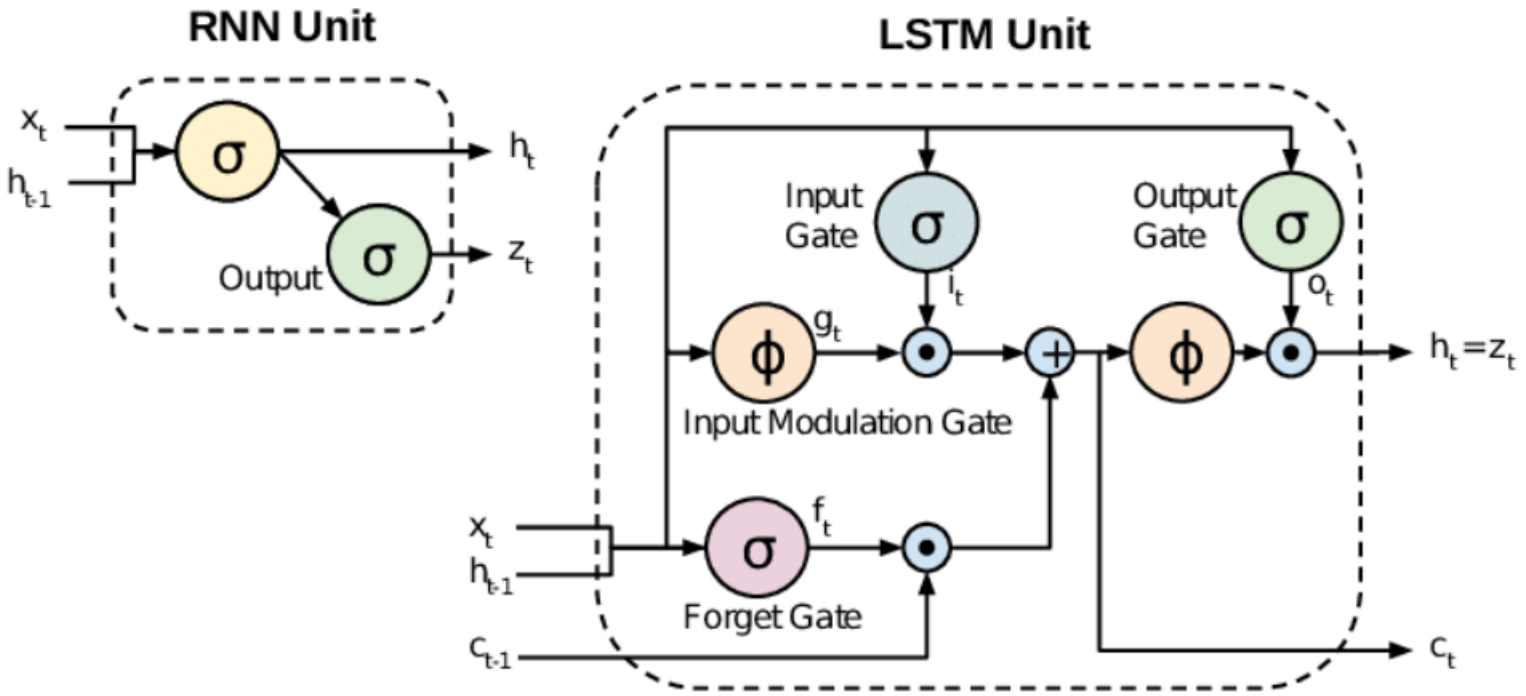
LSTM을 활용한 테슬라 주가 예측

2021 winter team
안혜림

LSTM을 활용한 테슬라 주가 예측

LSTM이란



LSTM은 RNN*의 단점을 보완하기 위해 변형된 알고리즘.
보통 신경망 대비 4배 이상 파라미터를 보유하여 많은 단계를 거치더라도 오랜 시간동안 데이터를 잘 기억함으로써 **과거 발생 데이터를 포함시켜 기억하여 다음 값을 예측**하는데 활용(시계열 데이터셋에 효과적인 모델).
ex. 날씨 예측, 주가 예측 등

LSTM을 활용한 테슬라 주가 예측

데이터 셋 생성

In [2]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
stocks = pd.read_csv('TSLA.csv', header=0)
stocks
```

Out[3]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2010-06-29	3.800000	5.000000	3.508000	4.778000	4.778000	93831500
1	2010-06-30	5.158000	6.084000	4.660000	4.766000	4.766000	85935500
2	2010-07-01	5.000000	5.184000	4.054000	4.392000	4.392000	41094000
3	2010-07-02	4.600000	4.620000	3.742000	3.840000	3.840000	25699000
4	2010-07-06	4.000000	4.000000	3.166000	3.222000	3.222000	34334500
...	...	...	...	...	...	...	...
2536	2020-07-27	287.000000	309.588013	282.600006	307.920013	307.920013	80243500
2537	2020-07-28	300.799988	312.940002	294.884003	295.298004	295.298004	79043500
2538	2020-07-29	300.200012	306.962006	297.399994	299.821991	299.821991	47134500
2539	2020-07-30	297.600006	302.648010	294.200012	297.497986	297.497986	38105000
2540	2020-07-31	303.000000	303.410004	284.196014	286.152008	286.152008	61235000

2541 rows × 7 columns

데이터 : Yahoo finance - TSLA 검색 - Historical Data - csv 파일 다운로드. (TSLA 699.60 31.54 4.72% : Tesla, Inc. - Yahoo Finance)

총 2540 row로, 2010년 6월 29일부터 2020년 7월 31일까지의 데이터만 나와있음.
컬럼은 [일자(Date), Open(시가), High(고가), Low(저가), Close(종가), Adj Close(조정종가), Volume(거래량)]으로 구성.

LSTM을 활용한 테슬라 주가 예측

데이터 셋 생성

```
In [4]: stocks.set_index('Date', inplace = True)
        stocks
```

Out[4]:

Date	Open	High	Low	Close	Adj Close	Volume
2010-06-29	3.800000	5.000000	3.508000	4.778000	4.778000	93831500
2010-06-30	5.158000	6.084000	4.660000	4.766000	4.766000	85935500
2010-07-01	5.000000	5.184000	4.054000	4.392000	4.392000	41094000
2010-07-02	4.600000	4.620000	3.742000	3.840000	3.840000	25699000
2010-07-06	4.000000	4.000000	3.166000	3.222000	3.222000	34334500
...	...	...	...	...	...	...
2020-07-27	287.000000	309.588013	282.600006	307.920013	307.920013	80243500
2020-07-28	300.799988	312.940002	294.884003	295.298004	295.298004	79043500
2020-07-29	300.200012	306.962006	297.399994	299.821991	299.821991	47134500
2020-07-30	297.600006	302.648010	294.200012	297.497986	297.497986	38105000
2020-07-31	303.000000	303.410004	284.196014	286.152008	286.152008	61235000

날짜별 종가와 예측 종가를 알 수 있게 Date 컬럼을 인덱스로 지정.

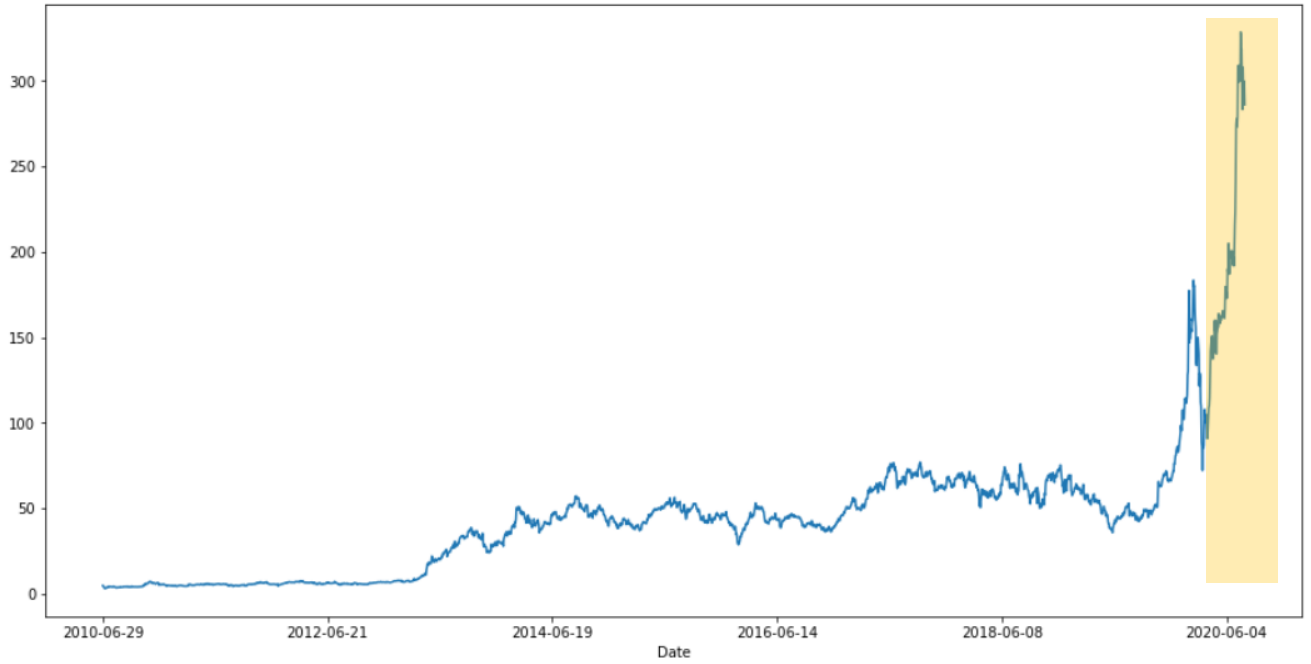
LSTM을 활용한 테슬라 주가 예측

데이터 도식화

```
In [4]: import matplotlib.pyplot as plt

        plt.figure(figsize = (16,8))
        stocks['Close'].plot()
```

```
Out[4]: <AxesSubplot:xlabel='Date'>
```



matplotlib 를 활용하여 종가(close)의 그래프를 그려봄.
테슬라의 주식가격은 2020년 6월 기준으로 급등하고 있음을 알 수 있음.

LSTM을 활용한 테슬라 주가 예측

정규화 작업

```
In [6]: data = stocks.filter(['Close']).values

In [7]: from sklearn.preprocessing import MinMaxScaler

In [8]: scalar = MinMaxScaler(feature_range=(0,1))
        scaled_data = scalar.fit_transform(data)
        scaled_data

Out[8]: array([[0.00497173],
               [0.00493486],
               [0.00378564],
               ...,
               [0.91157198],
               [0.90443086],
               [0.86956736]])
```

정규화(normalization), 왜 하는가?

각 컬럼에 들어있는 데이터의 상대적 크기로 분석결과가 달라질 수 있다. 예를 들어 A변수는 0~1000까지의 값을, B변수는 0~10까지의 값을 갖는다고 하자. 이 경우 상대적으로 큰 숫자 값을 갖는 A변수의 영향이 더 커진다. 따라서 숫자 데이터의 상대적인 크기 차이를 제거할 필요가 있다. 즉, 데이터의 scale을 통일해야하는데, 이것을 rescaling이라고 한다. (normalization은 rescaling의 하나의 방법)

정규화(normalization), 어떻게 하는가?

모든 컬럼에 있는 데이터를 최소0, 최대1 사이의 비율 값으로 다 바꾼다. 데이터들을 각 컬럼의 최대값으로 나눴다고 생각하면 쉬운데, 데이터에 마이너스(-) 값들이 있을 수 있다. 그래서 최소값을 분모, 분자에 최소값을 빼주는 것이 필요하다.

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

시가 고가 저가 종가 중 종가(close) 데이터로 예측을 해보겠음.
딥러닝 모델이 학습을 잘하기 위해서는 정규화 작업이 필요함.(이 작업이 없으면 예측 효율이 떨어짐)
skleran 패키지에 있는 MinMaxScaler를 활용하여 전체 학습 데이터를 정규화해줌.
모든 종가 데이터가 0~1사이 값(feature_range)으로 비율이 저장될 수 있도록 scale 통일(rescaling이라고함)

LSTM을 활용한 테슬라 주가 예측

훈련 데이터 설정

In [9]:
```
import math
training_data_len = math.ceil(len(scaled_data)*0.8)
```

훈련 데이터(train data) 길이

In [10]:
```
train_data = scaled_data[0:training_data_len]
train_data
```

훈련 데이터(train data) 세팅

Out[10]:
```
array([[0.00497173],
       [0.00493486],
       [0.00378564],
       ...,
       [0.17662241],
       [0.17307645],
       [0.18002704]])
```

In [11]:
```
x_train = []
y_train = []

PAST_SET = 10

for i in range(PAST_SET, len(train_data)):
    x_train.append(train_data[i-PAST_SET:i, 0])
    y_train.append(train_data[i,0])
```

x_train에 PAST_SET을 추가

훈련 데이터는 전체 중 0.8 비중으로 설정.
PAST_SET = 과거 기간의 주가 데이터에 기반하여 다음날의 종가를 예측할 것인가를 정하는 parameter.
즉, 예측일로부터 10일 전까지의 데이터까지 모두 예측하는데 고려할 날들(과거 10일 기반 내일 데이터 예측)
* math.ceil() : 실수를 입력하면 올림하여 정수를 반환하는 함수 ex. math.ceil(-3.14)면 -3반환,
ex. math.ceil(5.5)면 6 반환

LSTM을 활용한 테슬라 주가 예측

포맷변환

```
In [12]: x_train, y_train = np.array(x_train), np.array(y_train)

In [13]: x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))

In [14]: x_train.shape
Out[14]: (2023, 10, 1)
```

lstm은 3d 모양이 필요

훈련 데이터셋과 테스트셋의 포맷팅.
훈련 데이터셋(x_train)의 차원 크기(shape)를 보면 총 2023개의 데이터가 각각 10일 전까지 포함됨.
(2023, 10, 1)의 1은 LSTM에서 요구하는 형태..

LSTM을 활용한 테슬라 주가 예측

케라스 특징 및 절차

케라스에서 제공하는 시퀀스 모델로 원하는 레이어를 쉽게 순차적으로 쌓을 수 있음.
다중 출력이 필요하는 등 좀 더 복잡한 모델을 구성하려면 케라스 함수 API를 사용하면 됨.
케라스로 딥러닝 모델을 만들 때는 다음과 같은 순서로 작성함.
다른 딥러닝 라이브러리와 비슷한 순서이지만 훨씬 직관적이고 간결함.

1. **데이터셋 생성하기**
   원본 데이터를 불러오거나 시뮬레이션을 통해 데이터를 생성합니다.
   데이터로부터 훈련셋, 검증셋, 시험셋을 생성합니다.
   이 때 딥러닝 모델의 학습 및 평가를 할 수 있도록 **포맷 변환**을 합니다.
2. **모델 구성하기**
   시퀀스 모델을 생성한 뒤 필요한 **레이어를 추가하여 구성**합니다.
   좀 더 복잡한 모델이 필요할 때는 케라스 함수 API를 사용합니다.
3. **모델 학습과정 설정하기**
   학습하기 전에 학습에 대한 설정을 수행합니다.
   손실 함수 및 최적화 방법을 정의합니다.
   케라스에서는 **compile() 함수**를 사용합니다.
4. **모델 학습시키기**
   훈련셋을 이용하여 구성한 모델로 학습시킵니다.
   케라스에서는 **fit() 함수**를 사용합니다.
5. **학습과정 살펴보기**
   모델 학습 시 훈련셋, 검증셋의 손실 및 정확도를 측정합니다.
   반복횟수에 따른 손실 및 정확도 추이를 보면서 학습 상황을 판단합니다.
6. **모델 평가하기**
   준비된 시험셋으로 학습한 모델을 평가합니다.
   케라스에서는 evaluate() 함수를 사용합니다.
7. **모델 사용하기**
   임의의 입력으로 모델의 출력을 얻습니다.
   케라스에서는 predict() 함수를 사용합니다.

LSTM을 활용한 테슬라 주가 예측

2. 모델 구성하기 - 레이어 추가

```
In [15]: from keras.models import Sequential
         from keras.layers import Dense, LSTM

In [16]: x_train.shape[1]

Out[16]: 10

In [17]: model = Sequential()
         model.add(LSTM(50, return_sequences=True, input_shape = (x_train.shape[1], 1)))
         model.add(LSTM(50, return_sequences=False))
         model.add(Dense(25))
         model.add(Dense(1))
```

keras에서 딥러닝 모델 구성.
LSTM 레이어 2개와 Dense 2개로 구현.

케라스 Sequential 모델 시작하기

`Sequential` 모델은 레이어를 선형으로 연결하여 구성합니다. 레이어 인스턴스를 생성자에게 넘겨줌으로써 `Sequential` 모델을 구성할 수 있습니다.

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential([
    Dense(32, input_shape=(784,)),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
])
```

또한, `.add()` 메소드를 통해서 쉽게 레이어를 추가할 수 있습니다.

```
model = Sequential()
model.add(Dense(32, input_dim=784))
model.add(Activation('relu'))
```

LSTM을 활용한 테슬라 주가 예측

3. 모델 학습과정 설정하기 - compile() 함수

```
In [18]: model.compile(optimizer='adam', loss = 'mean_squared_error')
```

컴파일

모델을 학습시키기 이전에, `compile` 메소드를 통해서 학습 방식에 대한 환경설정을 해야 합니다. 다음의 세 개의 인자를 입력으로 받습니다.

- 정규화기 (optimizer). `rmsprp`나 `adagrad`와 같은 기존의 정규화기에 대한 문자열 식별자 또는 `Optimizer` 클래스의 인스턴스를 사용할 수 있습니다. 참고: 정규화기
- 손실 함수 (loss function). 모델이 최적화에 사용되는 목적 함수입니다. `categorical_crossentropy` 또는 `mse`와 같은 기존의 손실 함수의 문자열 식별자 또는 목적 함수를 사용할 수 있습니다. 참고: 손실
- 기준(metric) 리스트. 분류 문제에 대해서는 `metrics=['accuracy']`로 설정합니다. 기준은 문자열 식별자 또는 사용자 정의 기준 함수를 사용할 수 있습니다.

모델 구성 후 compile() 메서드를 호출해서 모델 학습 과정을 설정한다.
즉, 모델을 빌드하고 실행하기 전에 컴파일 하는 훈련 준비 단계로 생각하면 된다.

keras.Model.compile()에는 세 개의 파라미터가 중요하다.

- loss : 최적화 과정에서 최소화될 손실함수를 설정하는 것으로, MSE(평균 제곱 오차)와 binary_crossentropy가 자주 사용된다
- optimizer : 훈련 과정을 설정하는 것으로, Adam, SGD 등이 있다
- metrics : 훈련을 모니터링하기 위해 사용한다

LSTM을 활용한 테슬라 주가 예측

4. 모델 학습시키기 - fit() 함수

In [19]:
```
model.fit(x_train,y_train, batch_size = 1, epochs = 3)
```

```
Epoch 1/3
2023/2023 [==============================] - 8s 4ms/step - loss: 1.6932e-04
Epoch 2/3
2023/2023 [==============================] - 8s 4ms/step - loss: 6.8230e-05
Epoch 3/3
```

Out[19]: epoch의 횟수가 많아질수록(훈련횟수가 많아질수록) loss는 줄어들고 accuracy는 증가하게 되고 어느 시점이 되면 일정 값에 수렴하다가 성능이 점점 떨어지기도 합니다.(오버피팅)

In [20]:
```
x_test = []
y_test = data[training_data_len:, :]
for i in range(PAST_SET, len(test_data)):
    x_test.append(test_data[i-PAST_SET:i, 0])
```

In [21]:
```
x_test = np.array(x_test)
```

In [22]:
```
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
```

fit() : 주어진 epoch 수 만큼 모델 학습시키는 메서드
batch_size : 한번에 사용해야 하는 트레이닝 데이터의 크기 지정
epoch : 3번 학습시킨다는 것(너무 많으면 오히려 효율이 떨어지므로 아주 간단한 구현을 위해 3번 학습)
loss : 결과 값과의 차이를 의미하므로 작을 수록/0.0000...에 수렴할수록 좋은 모델
epoch 횟수가 많아질수록(학습횟수가 많아질수록) loss는 줄어드나, 어느 시점이 되면 일정 값에 수렴하다가 성능이 점점 떨어짐(오버피팅)

LSTM을 활용한 테슬라 주가 예측

6. 학습과정 평가- 손실 및 정확도 측정(rmse) / 7. 모델 사용하기 - predict()

In [23]:
```
predictions = model.predict(x_test)

predictions = scalar.inverse_transform(predictions)
```

0~1로 정규화된 데이터값을 원래 값으로 되돌림

In [24]:
```
rmse = np.sqrt(np.mean(predictions - y_test)**2)
rmse
```

Out[24]:
```
1.1457317052059626
```

In [25]:
```
data = stocks.filter(['Close'])

train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions
```

```
<ipython-input-25-530ff23f5129>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  valid['Predictions'] = predictions
```

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

RMSE 공식

rmse(Root Mean Square Error) : 평균 제곱근 오차
오차의 제곱에 평균을 취하고 이를 제곱근 한 것. 추정 값 또는 모델이 예측한 값과 실제 환경에서 관찰되는 값의 차이를 다룰 때 흔히 사용하는 척도.(정밀도를 표현하는데 적합)
값이 작을수록 추측의 정확도가 높아지는 것을 뜻함

LSTM을 활용한 테슬라 주가 예측

예측

In [26]:
```
valid
```

Out[26]:

Date	Close	Predictions
2018-07-26	61.330002	66.093979
2018-07-27	59.436001	66.360451
2018-07-30	58.034000	65.352158
2018-07-31	59.627998	64.064110
2018-08-01	60.167999	64.367485
...	...	...
2020-07-27	307.920013	258.327881
2020-07-28	295.298004	271.136963
2020-07-29	299.821991	267.525513
2020-07-30	297.497986	269.153290
2020-07-31	286.152008	268.169769

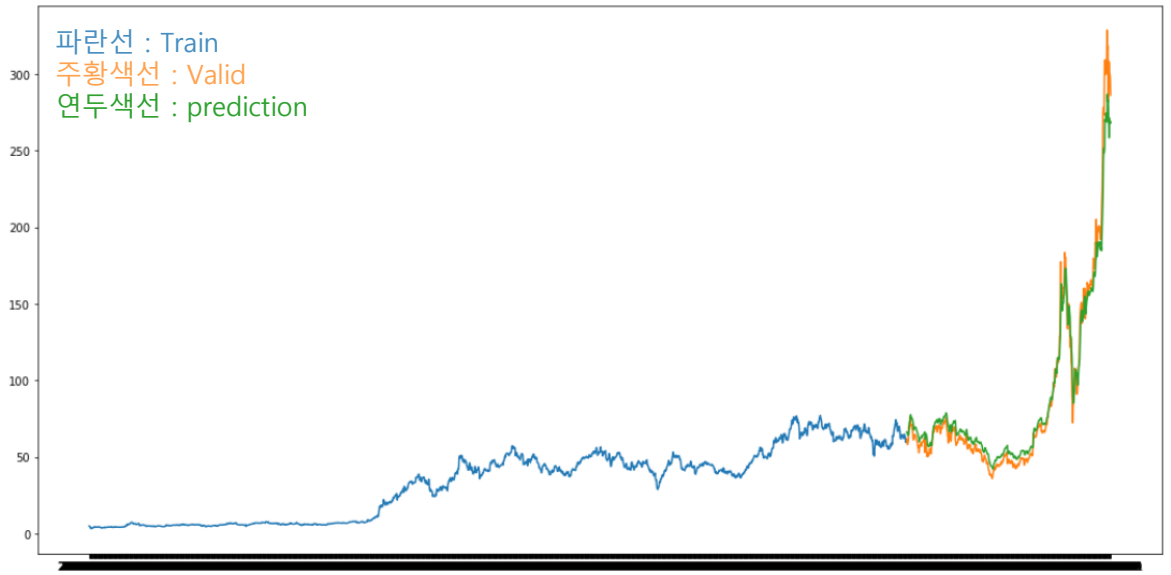
508 rows × 2 columns

왼쪽 Close : 실제값
오른쪽 Predictions : 예측값

LSTM을 활용한 테슬라 주가 예측

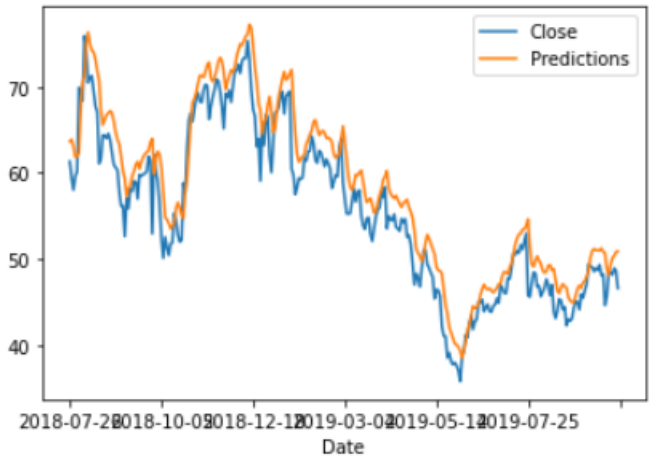
예측

```
In [27]: plt.figure(figsize=(18,9))
         plt.plot(train['Close'])
         plt.plot(valid[['Close', 'Predictions']])
Out[27]: [<matplotlib.lines.Line2D at 0x1941adcc9a0>,
          <matplotlib.lines.Line2D at 0x1941adcca00>]
```




```
In [39]: plt.figure(figsize=(28,14))
         valid[['Close', 'Predictions']][:300].plot()
Out[39]: <AxesSubplot:xlabel='Date'>

         <Figure size 2016x1008 with 0 Axes>
```



2018년 7월 26일부터 예측된 값을 알 수 있음.
종가(Close)와 예측(Predictions) 비슷한 추세를 보임

Train Data = 모델의 훈련을 위한 훈련용 데이터.
Test Data = 모델을 평가하기 위해 정답(결과)을 이미 알고있는 테스트용 데이터.
Validation Data = 여러 분석 모델 중 어떤 모델이 적합한지 선택하기 위한 검증용 데이터.

감사합니다.