





# LSTM을 사용한 주식 예측

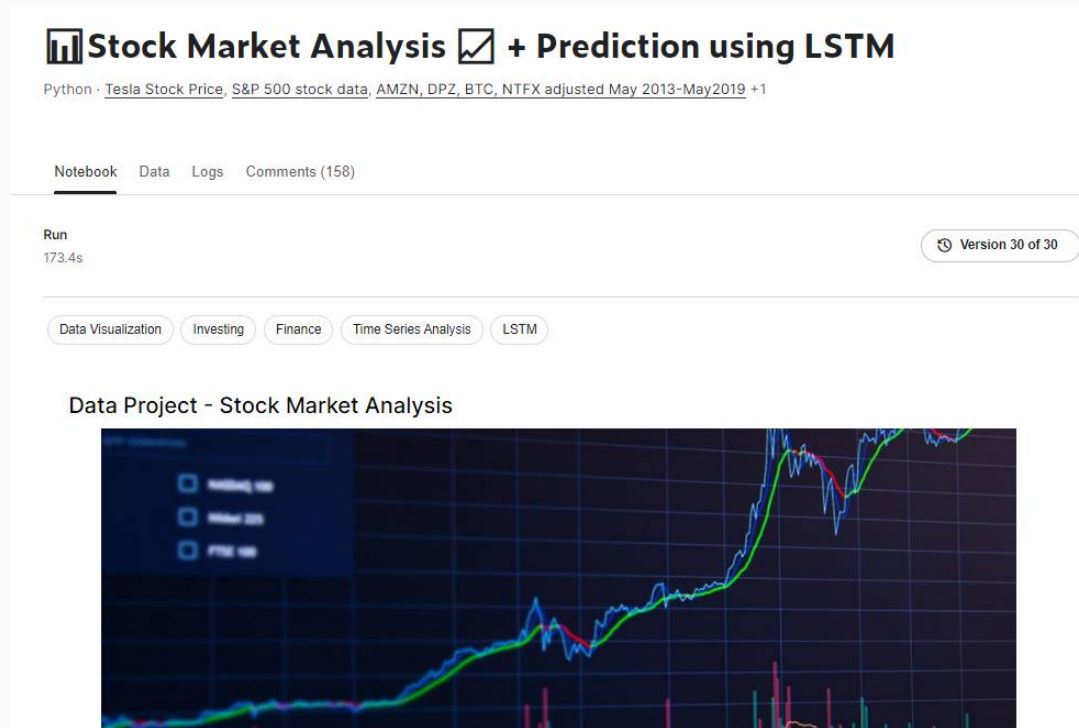
컴퓨터공학전공 권영기

---

## CONTENTS

-  01 주제 선정 및 데이터셋
-  02 데이터 전처리
-  03 LSTM을 이용한 주식 예측
-  04 결론 및 느낀 점

# 1. 주제선정 및 데이터셋



해당 오픈소스는 데이터셋을 이용하여  
LSTM(Long Short Term Memory) 방식을 통해 미래의 주가를 예측

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
plt.style.use("fivethirtyeight")
%matplotlib inline

# For reading stock data from yahoo
from pandas_datareader.data import DataReader

# For time stamps
from datetime import datetime
```

pandas\_datareader 패키지의 DataReader라는 함수는  
웹 상의 데이터를 DataFrame 객체로 만드는 기능을 제공합니다

DataReader라는 함수의 파라미터에는  
조회할 종목, 데이터를 가져올 소스,  
조회기간의 시작일, 종료일 순으로 입력을 할 수 있습니다

# 1. 데이터셋

```
# Get the stock quote
df = DataReader('AAPL', data_source='yahoo', start='2012-01-01', end=datetime.now())
# Show teh data
df
```



	High	Low	Open	Close	Volume	Adj Close
Date						
2012-01-03	14.732143	14.607143	14.621429	14.686786	302220800.0	12.650659
2012-01-04	14.810000	14.617143	14.642857	14.765714	260022000.0	12.718646
2012-01-05	14.948214	14.738214	14.819643	14.929643	271269600.0	12.859850
2012-01-06	15.098214	14.972143	14.991786	15.085714	318292800.0	12.994284
2012-01-09	15.276786	15.048214	15.196429	15.061786	394024400.0	12.973674
...	...	...	...	...	...	...
2021-05-03	134.070007	131.830002	132.039993	132.539993	75135100.0	132.539993
2021-05-04	131.490005	126.699997	131.190002	127.849998	137564700.0	127.849998
2021-05-05	130.449997	127.970001	129.199997	128.100006	84000900.0	128.100006
2021-05-06	129.750000	127.129997	127.889999	129.740005	77968100.0	129.740005
2021-05-07	131.258194	129.475006	130.850006	130.210007	74667644.0	130.210007

```
plt.figure(figsize=(16,6))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```



## 2. 데이터 전처리

```
# Create a new dataframe with only the 'Close column
data = df.filter(['Close'])

# Convert the dataframe to a numpy array
dataset = data.values

# Get the number of rows to train the model on
training_data_len = int(np.ceil( len(dataset) * .95 ))

training_data_len
```



```
# Scale the data
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)

scaled_data
```

```
# Create the training data set
# Create the scaled training data set
train_data = scaled_data[0:int(training_data_len), :]
# Split the data into x_train and y_train data sets
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<= 61:
        print(x_train)
        print(y_train)
        print()

# Convert the x_train and y_train to numpy arrays
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
```

필터를 걸어 증가만 들어있는 데이터프레임을 만듭니다

```
[array([0.00572147, 0.00633231, 0.00760099, 0.00808085, 0.00862367,
0.00904103, 0.00885031, 0.00852969, 0.00809297, 0.00944457,
0.0106635 , 0.01028759, 0.00822841, 0.01019361, 0.00825082,
0.01551431, 0.01495322, 0.01568568, 0.01726944, 0.01822855,
0.01814839, 0.01785265, 0.01911303, 0.02029878, 0.02164209,
0.02381183, 0.02033666, 0.02043075, 0.03097611, 0.0328722 ,
0.02961346, 0.03086831, 0.03084343, 0.034362 , 0.03386171,
0.03478765, 0.03645157, 0.03737752, 0.04004478, 0.04198786,
0.04254895, 0.0427452 , 0.03942288, 0.03862131, 0.03874017,
0.04186349, 0.04274244, 0.04463025, 0.04908028, 0.0501734,
0.05390621, 0.05390897, 0.05820147, 0.05954478, 0.05858842,
0.05771501, 0.05608565, 0.05082669, 0.0618997 , 0.06276759]), array([0.00633231, 0.00760099, 0.00808085, 0.00862367,
0.00904103, 0.00885031, 0.00852969, 0.00809297, 0.00944457,
0.0106635 , 0.01028759, 0.00822841, 0.01019361, 0.00825082,
0.01551431, 0.01495322, 0.01568568, 0.01726944, 0.01822855,
0.01814839, 0.01785265, 0.01911303, 0.02029878, 0.02164209,
0.02381183, 0.02033666, 0.02043075, 0.03097611, 0.0328722 ,
0.02961346, 0.03086831, 0.03084343, 0.034362 , 0.03386171,
0.03478765, 0.03645157, 0.03737752, 0.04004478, 0.04198786,
0.04254895, 0.0427452 , 0.03942288, 0.03862131, 0.03874017,
0.04186349, 0.04274244, 0.04463025, 0.04908028, 0.0501734,
0.05390621, 0.05390897, 0.05820147, 0.05954478, 0.05858842,
0.05771501, 0.05608565, 0.05082669, 0.0618997 , 0.06276759]), array([0.00633231, 0.00760099, 0.00808085, 0.00862367, 0.00904103,
0.00885031, 0.00852969, 0.00809297, 0.00944457, 0.0106635 ,
0.01028759, 0.00822841, 0.01019361, 0.00825082, 0.01551431,
0.01495322, 0.01568568, 0.01726944, 0.01822855, 0.01814839,
0.01785265, 0.01911303, 0.02029878, 0.02164209, 0.02381183,
0.02033666, 0.02043075, 0.03097611, 0.0328722 , 0.02961346,
0.03086831, 0.03084343, 0.034362 , 0.03386171, 0.03478765,
0.03645157, 0.03737752, 0.04004478, 0.04198786, 0.04254895,
0.0427452 , 0.03942288, 0.03862131, 0.03874017, 0.04186349,
0.04274244, 0.04463025, 0.04908028, 0.0501734, 0.05390621,
0.05390897, 0.05820147, 0.05954478, 0.05858842, 0.05771501,
0.05608565, 0.05082669, 0.0618997 , 0.06276759, 0.06062272]))
```



데이터 스케일링 진행합니다

### 3. LSTM을 이용한 주식 예측

#### 시계열데이터

- 시계열 데이터란 시간의 흐름에 따라 관찰된 데이터를 의미합니다.

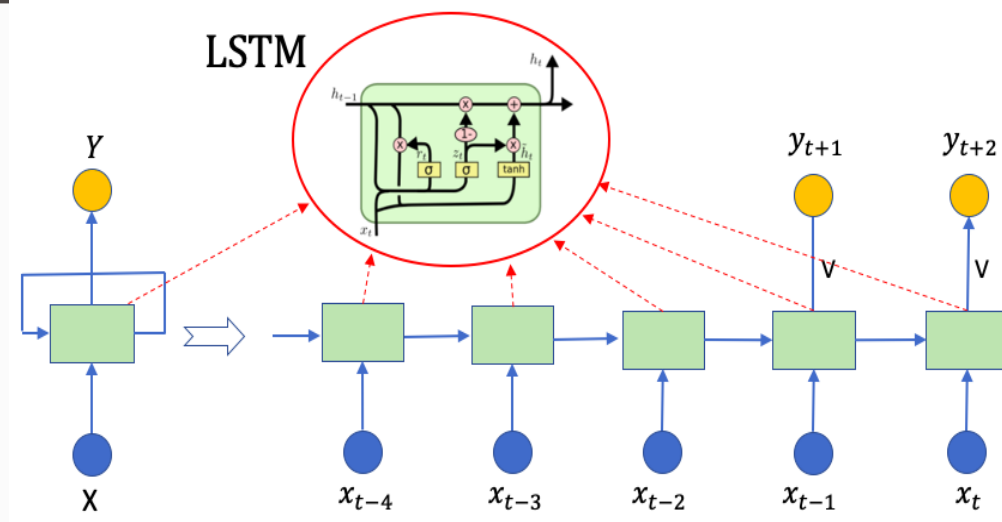
#### 정상성 시계열과 비정상성 시계열

- 정상성 시계열은 어떤 시계열자료의 변화 패턴이 평균값을 중심으로 일정한 변동폭을 갖는 시계열로 시간의 추이와 관계없이 평균과 분산이 일정합니다.
- 비정상성 시계열은 시간의 추이에 따라서 점진적으로 증가하는 추세를 보이거나 분산이 일정하지 않은 특징을 가진 시계열로 대부분의 시계열자료입니다.

### 3. LSTM을 이용한 주식 예측

#### LSTM(Long Short Term Memory)

- 자연어처리에 사용되는 RNN의 한 종류로써 긴 의존 기간을 필요로 하는 학습을 수행할 능력을 가지고 있습니다
- sequence를 저장하고 이를 학습에 활용함
- 데이터의 양과 기간이 길수록 보다 더 정확한 예측을 할 수 있음



### 3. LSTM을 이용한 주식 예측

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

# Build the LSTM model
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

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

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

```
# Create the testing data set
# Create a new array containing scaled values from index 1543 to 2002
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 ))
```

```
# Get the models predicted price values
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
```

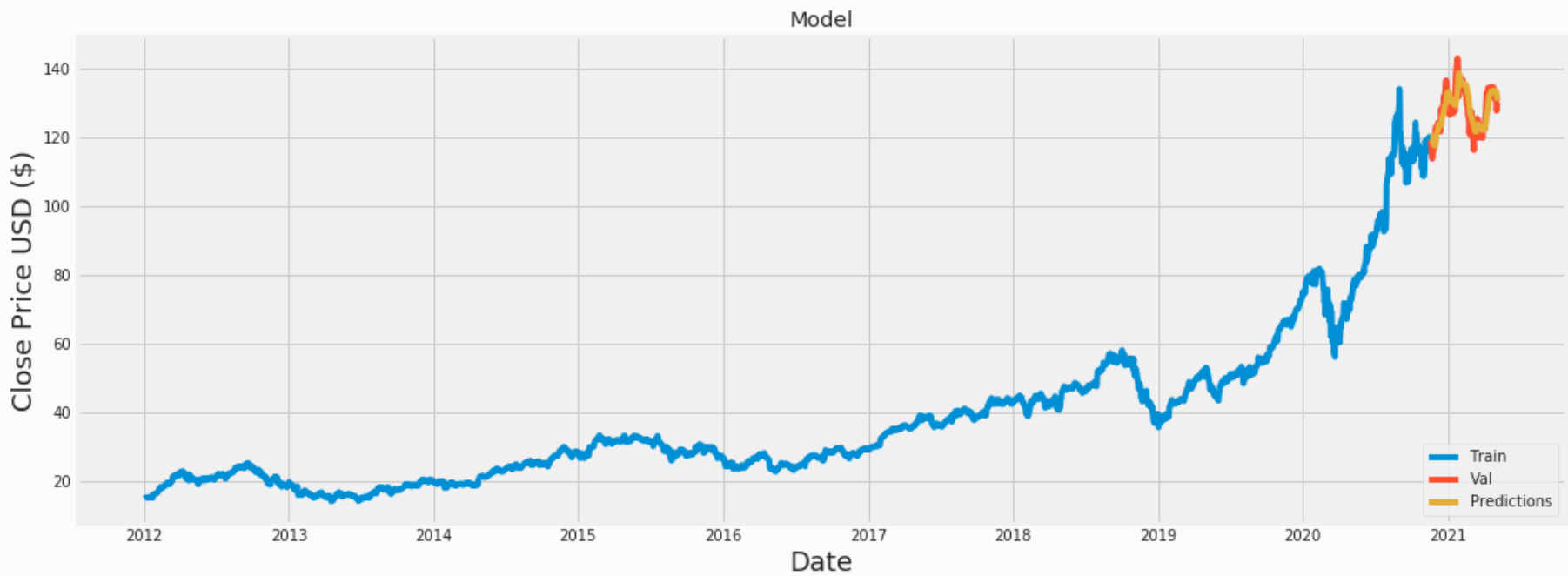
#### LSTM 모델을 컴파일 후 학습

```
# Plot the data
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions
# Visualize the data
plt.figure(figsize=(16,6))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
plt.show()
```

#### LSTM 모델을 적용한 주식 예측한 값들을 시각화



# 4. 결론 및 느낀 점



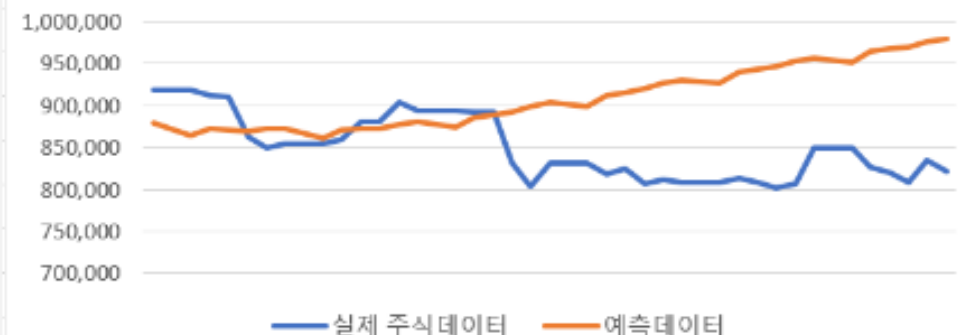
	Close	Predictions
Date		
2020-11-18	118.029999	120.124252
2020-11-19	118.639999	120.152763
2020-11-20	117.339996	120.112671
2020-11-23	113.849998	119.856140
2020-11-24	115.169998	119.065277
...	...	...
2021-05-03	132.539993	133.429199
2021-05-04	127.849998	133.178024
2021-05-05	128.100006	132.336975
2021-05-06	129.740005	131.428970
2021-05-07	130.210007	130.830887

실제 주식과 비교해보니 등락되는 타이밍과 종가의 값이 비슷한 양상을 보인다는 것을 알 수 있었습니다

## 4. 결론 및 느낀 점

```
1 rm(list=ls())
2
3 -----library-----#
4
5 library(lubridate)
6 library(httr)
7 library(rvest)
8 library(xts)
9 library(data.table)
10 library(dplyr)
11 library(prophet)
12
13 -----function-----#
14
15 t2n <- function(x) { as.numeric( as.POSIXct(strptime(x, "%Y-%m-%d %H:%M:%OS")) ) }
16 n2t <- function(x) { format(as.POSIXct(x, origin = "1970-01-01 09:00:00", tz="UTC"), "%Y-%m-%d %H:%M:%OS") }
17
18 -----data-----#
19
20 key = "005930"
21 url = paste0("https://fchart.stock.naver.com/sise.nhn?symbol=", key, "&timeframe=day&count=1000&requestType=0")
22 data = GET(url) %>% read_html() %>% html_nodes("item") %>% html_attr("data") %>% strsplit("\\|") %>% head(970)
23
24 -----data-preprocessing-----#
25
26 df = data.table( ds = sapply(data, function(x) { x[1] } ), y = sapply(data, function(x) { x[4] } ) )
27 df$y = df$y %>% as.double()
28
29 df$ds = paste(paste0(substr(df$ds, 1, 4), "-", substr(df$ds, 5, 6), "-", substr(df$ds, 7, 8)), "00:00:00")
30 df$ds_value = t2n(df$ds)
31
32 df = data.table( df %>% filter(y != 0) %>% filter( ds >= "2019-05-27" ) ); nrow(df)
33 n_pred = 90
34 df = rbind(df, data.table( ds = n2t( seq( t2n(tail(df, 1)$ds) + 60*60*24*1, t2n(tail(df, 1)$ds) + 60*60*24*(n_pred),
35 by = 60*60*24) ) ), y=NA, ds_value = NA ))
36 df$ds_value = t2n(df$ds)
37
38 -----modeling-----#
39
40 m = prophet( na.omit(df), yearly.seasonality=T, weekly.seasonality = TRUE, daily.seasonality = TRUE )
41 m_pred = predict(m, df )
42 df$yhat = m_pred$yhat
43 df$yhat_lower = m_pred$yhat_lower
44 df$yhat_upper = m_pred$yhat_upper
45 data12 <- df[483:572]
```

> LG화학



페이스북의 Prophet 모델

감사합니다