

---

# WELCRON 주가예측

---

2020. 03. 19.

2조 | 김기목 김선규 박솔이 이상정 주원진

# Index

- 연구목적
- 연구방법
- 관련지표
- 주가예측
- 결과



# 연구목적

- 주식 시장에 나타난 과거의 데이터를 바탕으로 현재 시세를 예측하고자 함
- 기계학습을 활용함으로 보다 객관적인 분석을 통해 실제 주식투자에 활용하기 위함

# 연구방법

① 주식 데이터 수집

② 예측 모델 생성

③ 예측 모델 비교 및 평가

데이터 수집 기간

Training Data

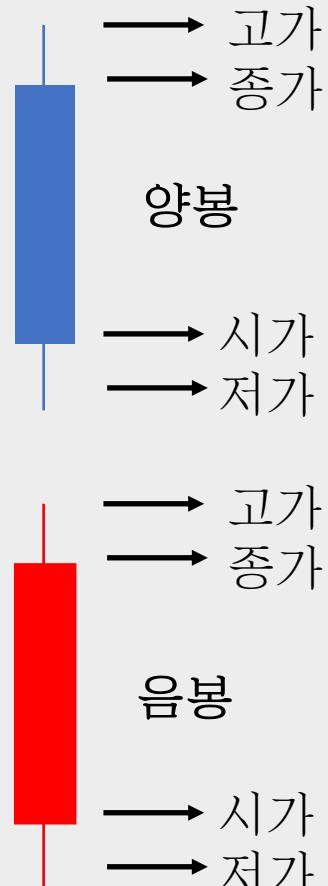
Test Data

2010.01.04~2018.12.28

2010.01.04~2017.03.06

2017.03.07~2018.12.28

# 관련지표



# 관련지표

- **볼린저 밴드(HBB(상단), MBB(중심), LBB(하단)):**  
20일 이동평균선을 기준으로 주가가 어느 정도 위치에 있는지 나타내는 지표
- **DMI:** 시장 방향성지표 (PDI(+DI: 주가상승), MDI(-DI: 주가하락), ADX)
- **지수 이동 평균**
- **스토캐스틱 인덱스(KDJ\_K, KDJ\_D, KDJ\_J)**
  - %K : 12일동안 고가 저가 범위 중 금일 종가가 어디에 위치해 있는지를 알려주는 지표
  - %D(Slow %K): %K의 5일 동안의 이동평균값
  - Slow %D: %D의 5일 동안 이동평균값
- **MACD :** 단기와 장기 이동평균선의 간격 차이를 통해 주가 흐름을 파악하는 보조지표
- **RSI :** 가격의 상승압력과 하락압력의 상대적인 강도를 나타내는 지표

# 주가예측

| 데이터 |



| 분류&예측 모델 |

SVM

Logistic Regression

Gradient Boosting

Naïve Bayes

Decision Tree

KNN

LSTM

ARIMA

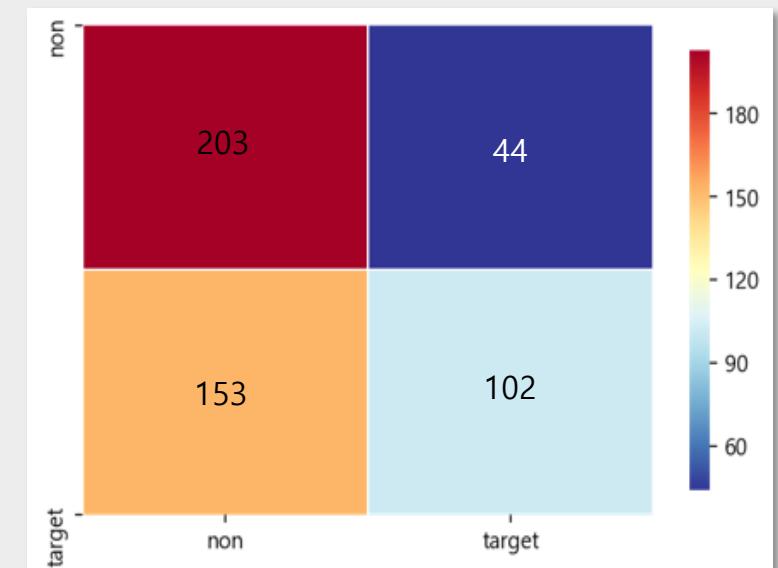
# 주가 등락 예측

# 주가 등락 예측\_Logistic

- ✓ 주가 하락[0]이라고 예측한 데이터의 57%만 실제 주가 하락[0], 주가 상승[1]이라고 예측한 데이터의 70%만 실제 주가 상승[1]
- ✓ 또한, 실제 주가 하락[0]인 데이터 중의 82%만 주가 하락[0]으로 판별, 실제 주가 상승[1] 인 데이터 중의 40%만 주가 상승[1]으로 판별

Logistic	precision	recall	f1-score	support
0[주가하락]	0.57	0.82	0.67	247
1[주가상승]	0.7	0.4	0.51	255
accuracy			0.61	502
macro avg	0.63	0.61	0.59	502
weighted avg	0.64	0.61	0.59	502

Confusion Matrix



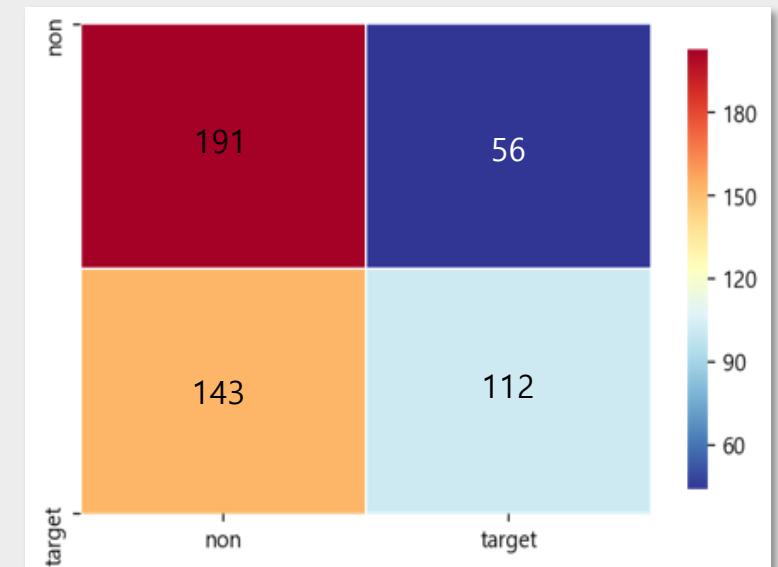
- macro : 단순평균
- weighted : 각 클래스에 속하는 표본의 갯수로 가중평균
- accuracy : 정확도, 전체 학습 데이터의 개수에서 각 클래스에서 자신의 클래스를 정확하게 맞춘 개수의 비율

# 주가 등락 예측\_Gradient

- ✓ 주가 하락[0]이라고 예측한 데이터의 57%만 실제 주가 하락[0], 주가 상승[1]이라고 예측한 데이터의 67%만 실제 주가 상승[1]
- ✓ 또한, 실제 주가 하락[0]인 데이터 중의 77%만 주가 하락[0]으로 판별, 실제 주가 상승[1] 인 데이터 중의 44%만 주가 상승[1]으로 판별

Gradient	precision	recall	f1-score	support
0[주가하락]	0.57	0.77	0.66	247
1[주가상승]	0.67	0.44	0.53	255
accuracy			0.60	502
macro avg	0.62	0.61	0.59	502
weighted avg	0.62	0.60	0.59	502

Confusion Matrix



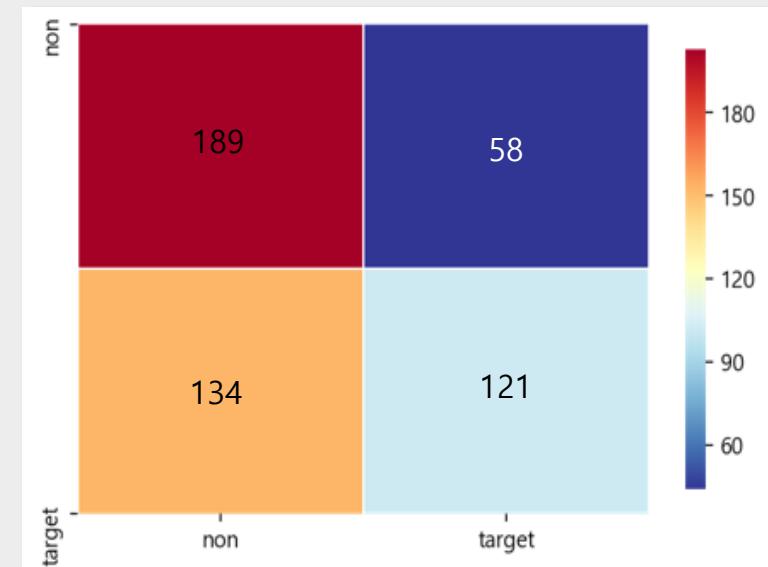
- macro : 단순평균
- weighted : 각 클래스에 속하는 표본의 갯수로 가중평균
- accuracy : 정확도, 전체 학습 데이터의 개수에서 각 클래스에서 자신의 클래스를 정확하게 맞춘 개수의 비율

# 주가 등락 예측\_SVM

- ✓ 주가 하락[0]이라고 예측한 데이터의 59%만 실제 주가 하락[0], 주가 상승[1]이라고 예측한 데이터의 68%만 실제 주가 상승[1]
- ✓ 또한, 실제 주가 하락[0]인 데이터 중의 77%만 주가 하락[0]으로 판별, 실제 주가 상승[1] 인 데이터 중의 47%만 주가 상승[1]으로 판별

SVC	precision	recall	f1-score	support
0[주가하락]	0.59	0.77	0.66	247
1[주가상승]	0.68	0.47	0.56	255
accuracy			0.62	502
macro avg	0.63	0.62	0.61	502
weighted avg	0.63	0.62	0.61	502

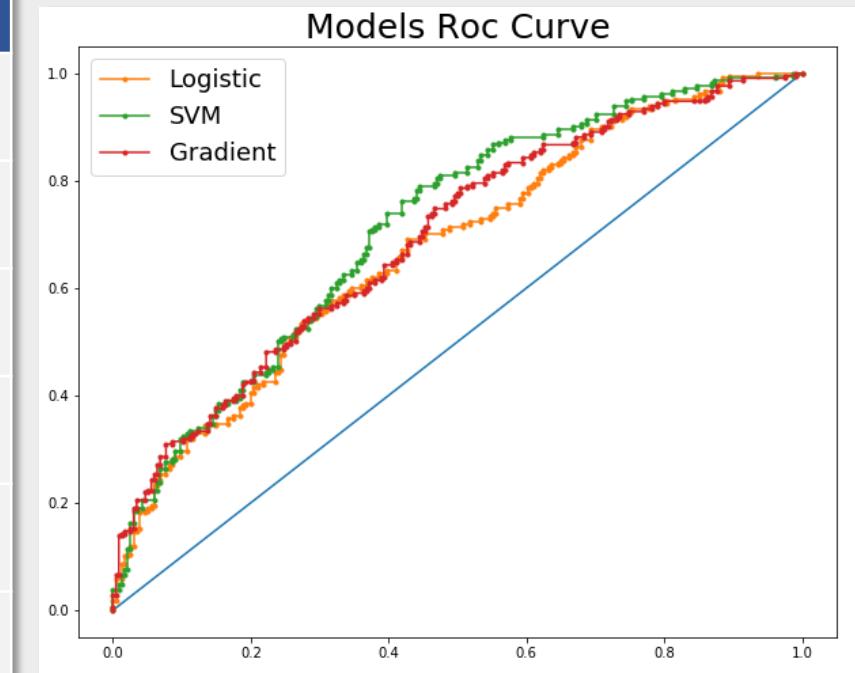
Confusion Matrix



- macro : 단순평균
- weighted : 각 클래스에 속하는 표본의 갯수로 가중평균
- accuracy : 정확도, 전체 학습 데이터의 개수에서 각 클래스에서 자신의 클래스를 정확하게 맞춘 개수의 비율

# 주가 등락 예측\_결과

	accuracy	precision	recall	f1-score	AUC score
SVM	0.62	0.68	0.47	0.56	0.62
Logistic	0.61	0.7	0.4	0.509	0.61
Gradient	0.6	0.67	0.44	0.53	0.61
NaiveBayes	0.59	0.7	0.36	0.48	0.6
DecisionTree	0.57	0.61	0.47	0.53	0.58
RandomForest	0.55	0.61	0.35	0.44	0.56
KNN	0.54	0.57	0.41	0.48	0.54



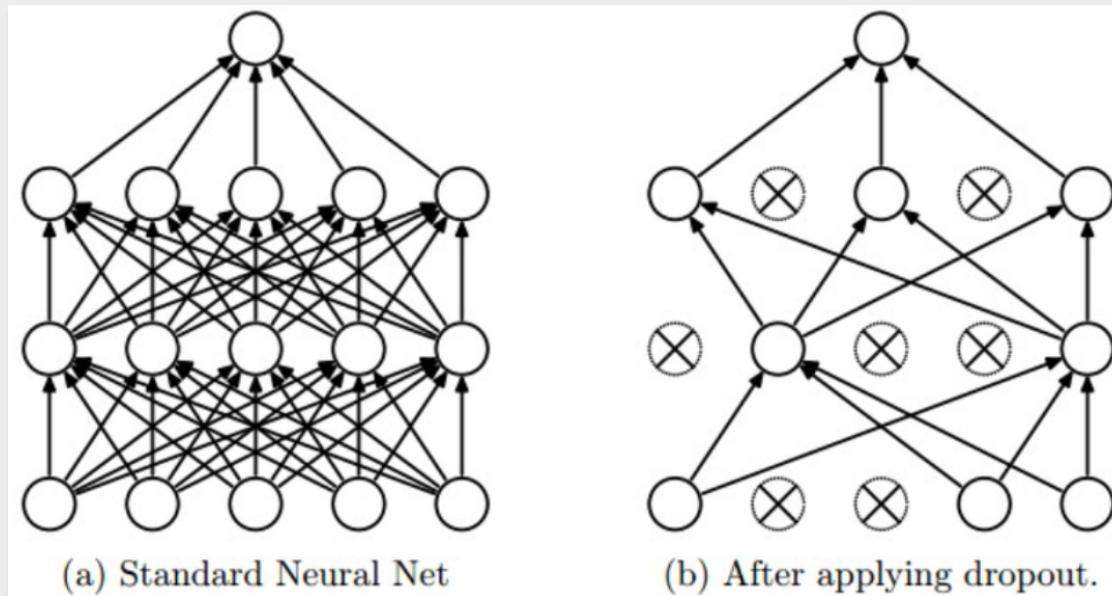
SVM > Logistic > Gradient

# LSTM

(Long Short Term Memory)

# LSTM

- ✓ 주가 데이터가 순서가 있는 시계열(Time-series) 데이터이기 때문에 LSTM 모델을 사용



dropout의 작동 원리

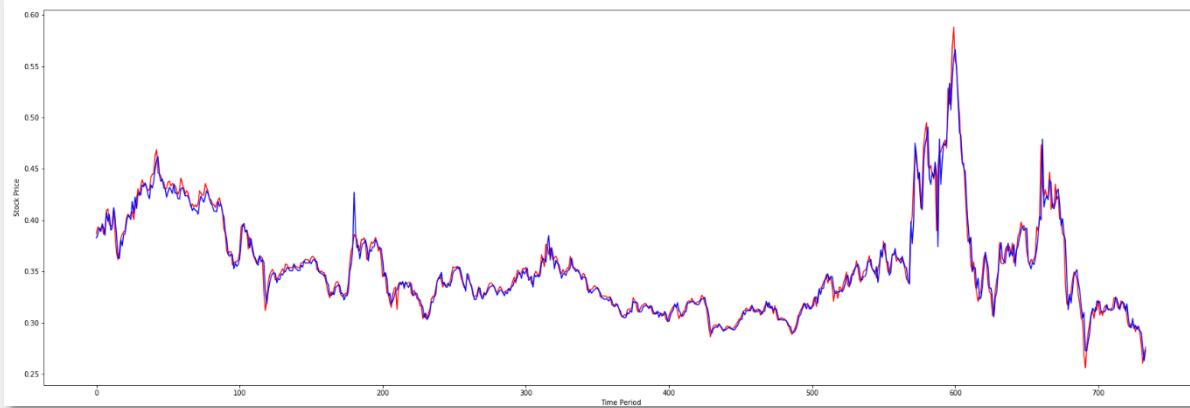
일반적으로 신경망에서 hidden layer의 개수가 많아지면 학습 능력이 좋아지지만, 망의 크기가 커지면 커질수록 overfitting에 빠질 가능성이 높음

Dropout : over-fitting을 줄이기 위한 regularization 기술

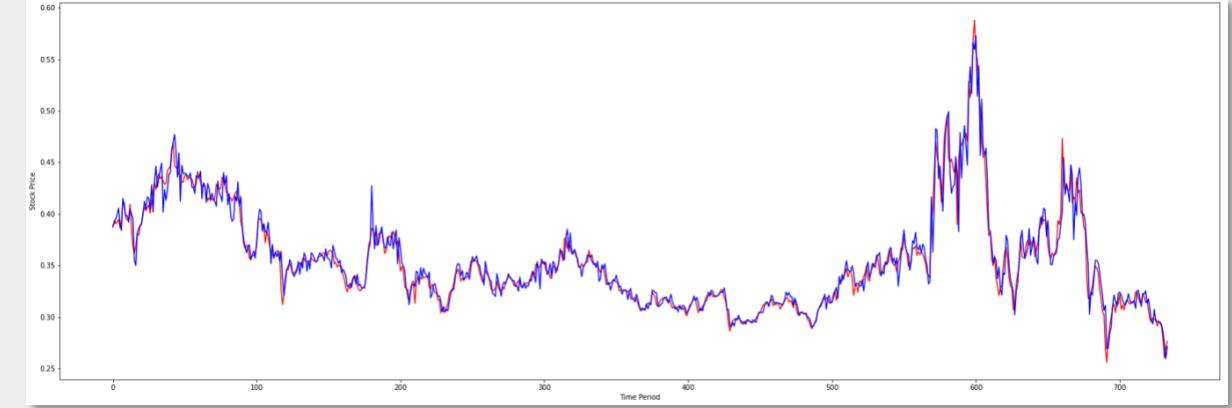
- 네트워크에서 일시적으로 유닛을 배제하고, 그 배제된 유닛의 연결을 모두 끊는다.
- keep\_prob : 주어진 유닛을 유지할 확률 즉, drop하지 않을 확률

# keep\_prob 향고

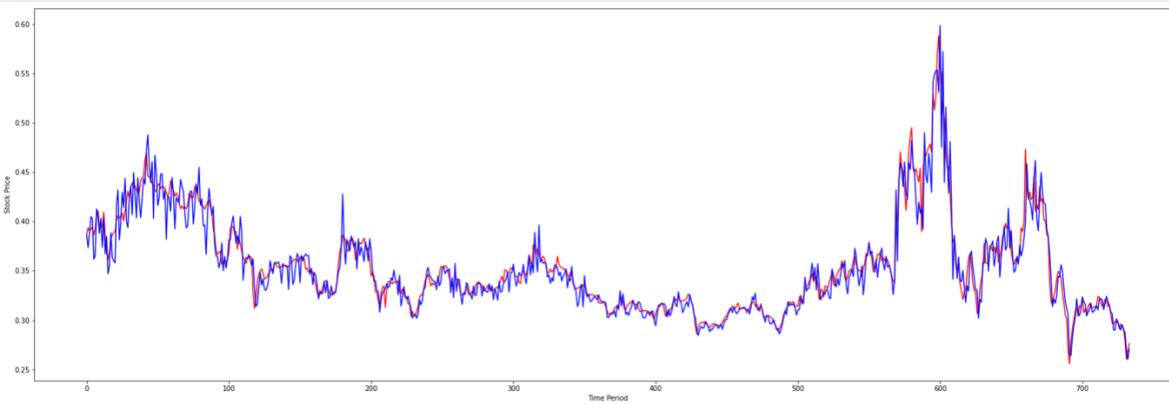
- keep\_prob = 1.0, softsign



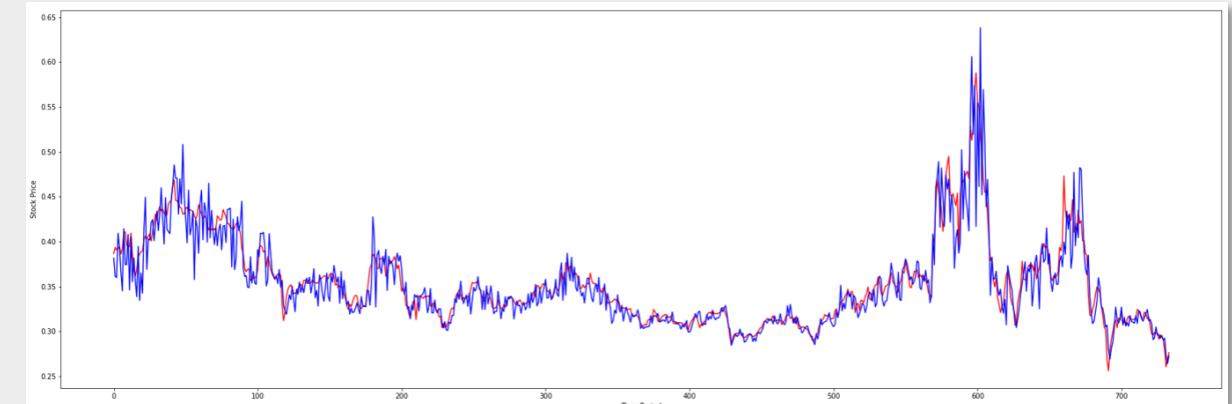
- keep\_prob = 0.9, softsign



- keep\_prob = 0.7, softsign

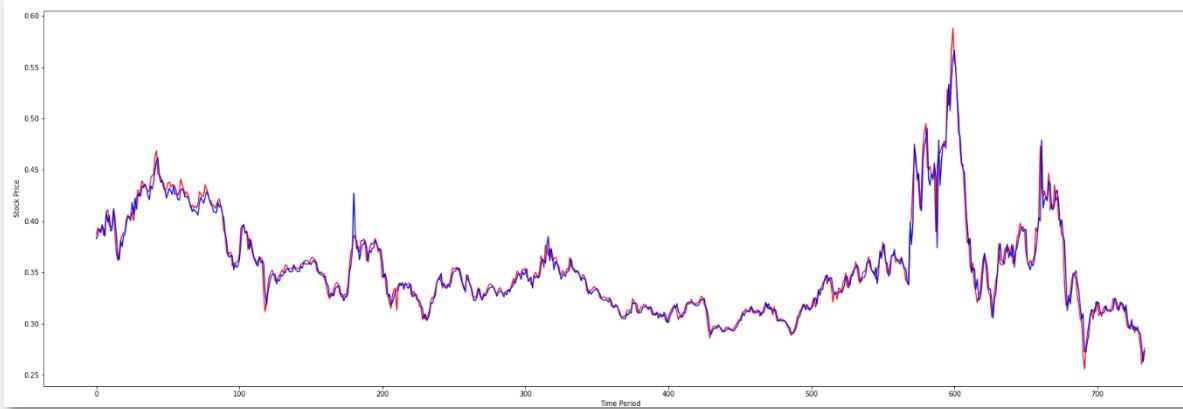


- keep\_prob = 0.5, softsign

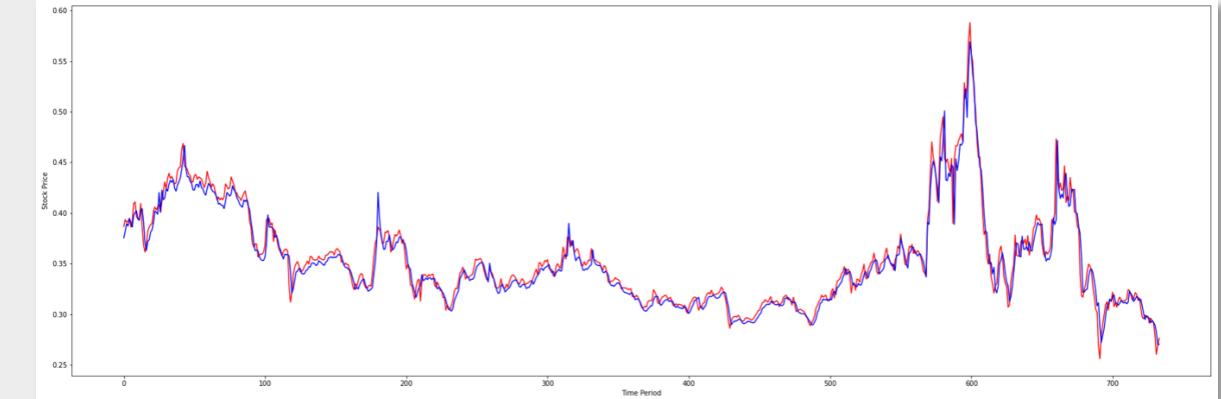


# 활성함수 비교

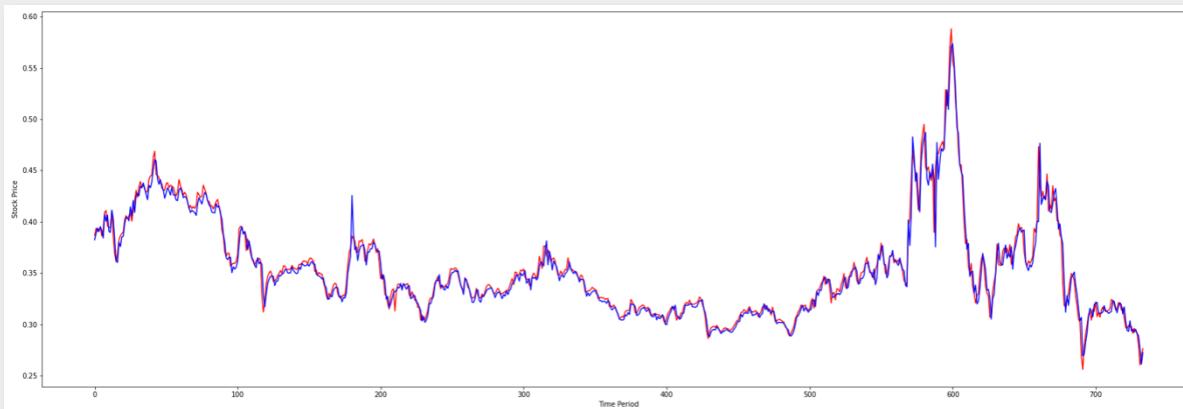
- keep\_prob = 1.0, softsign



- keep\_prob = 1.0, relu

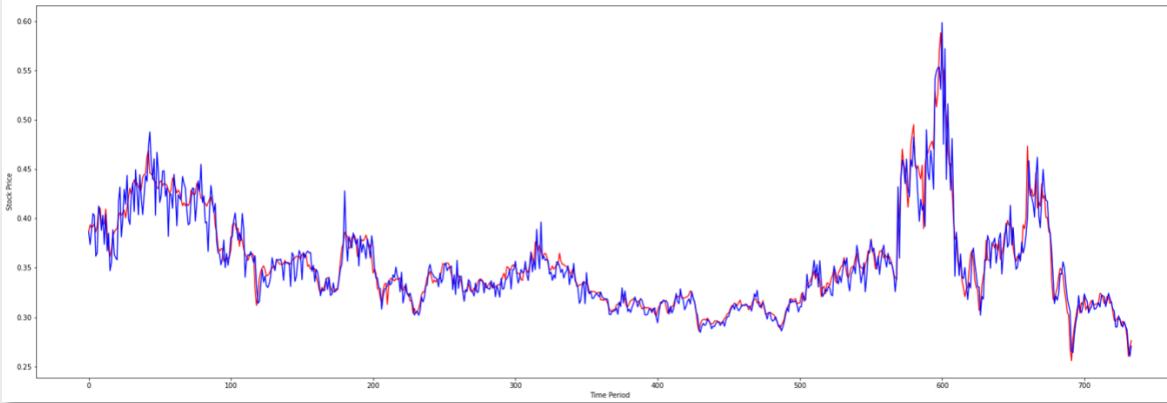


- keep\_prob = 1.0, tanh

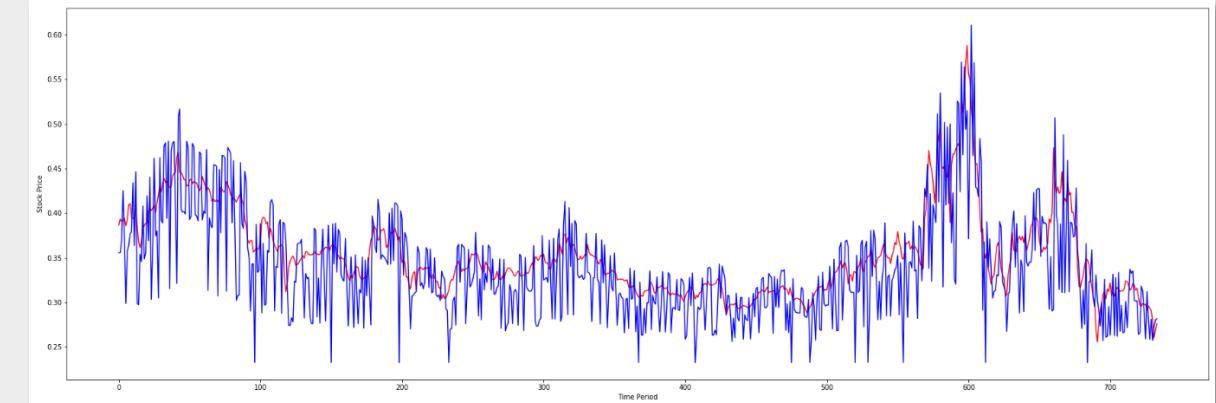


# 활성함수 비교

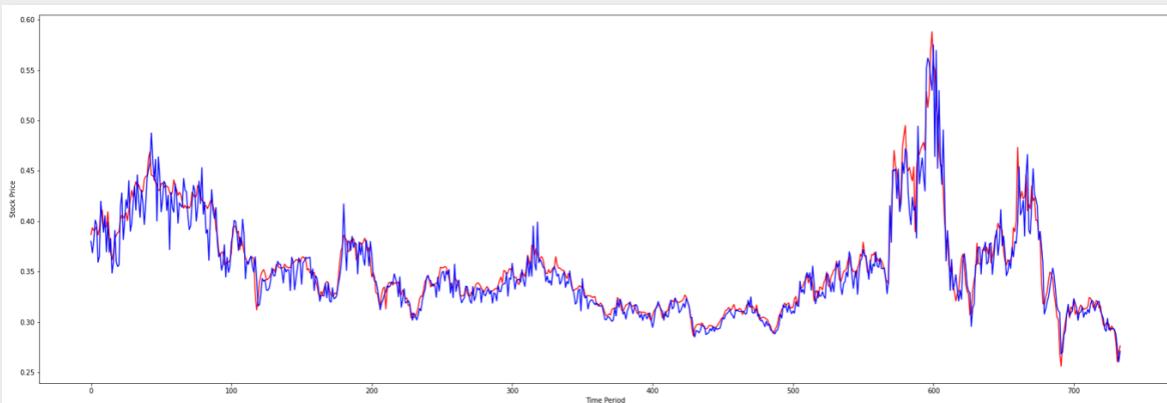
- keep\_prob = 0.7, softsign



- keep\_prob = 0.7, relu

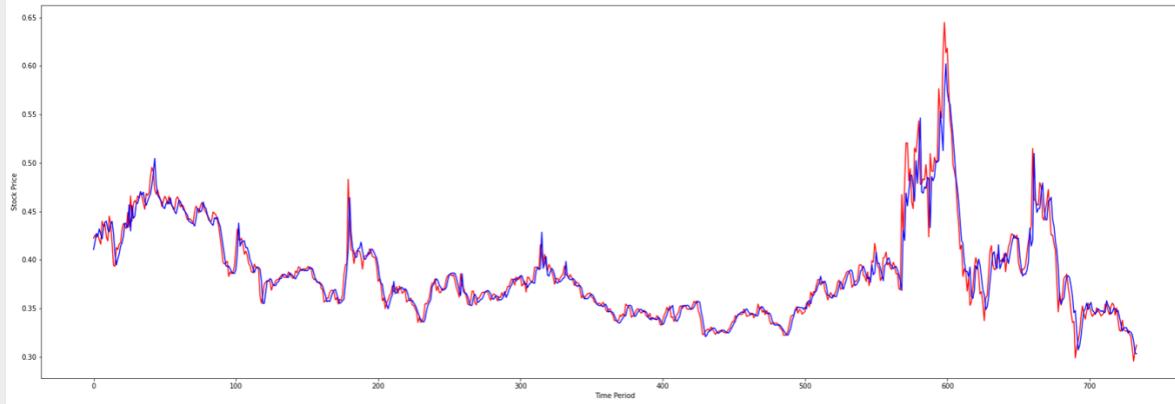


- keep\_prob = 0.7, tanh

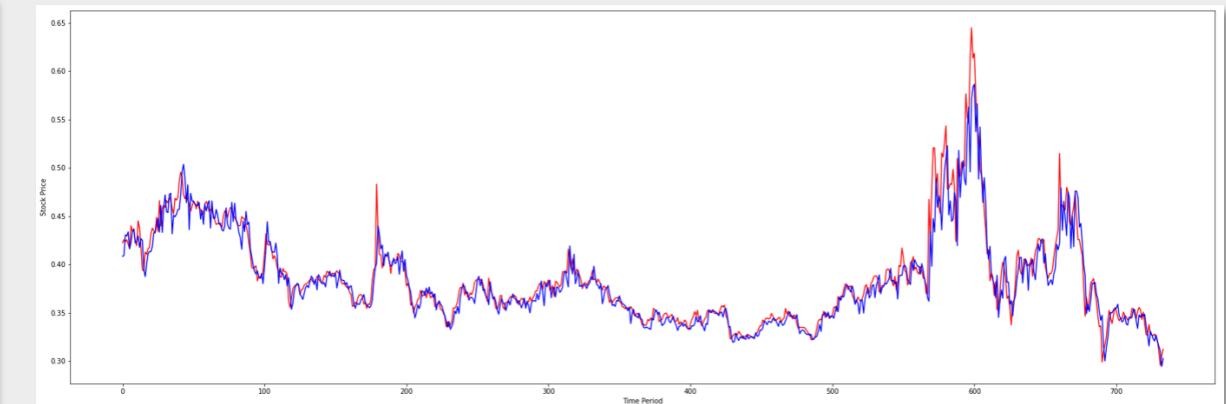


# keep\_prob 비교\_변수추가

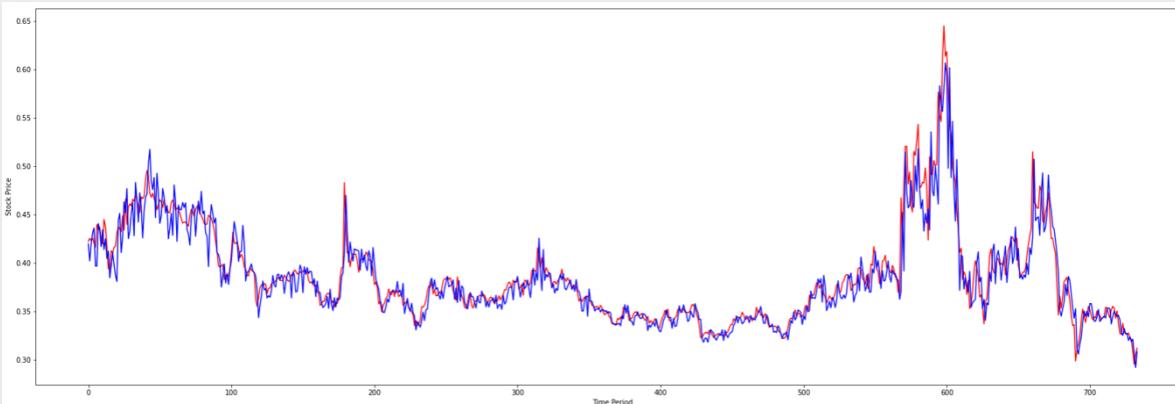
- keep\_prob = 1.0, softsign



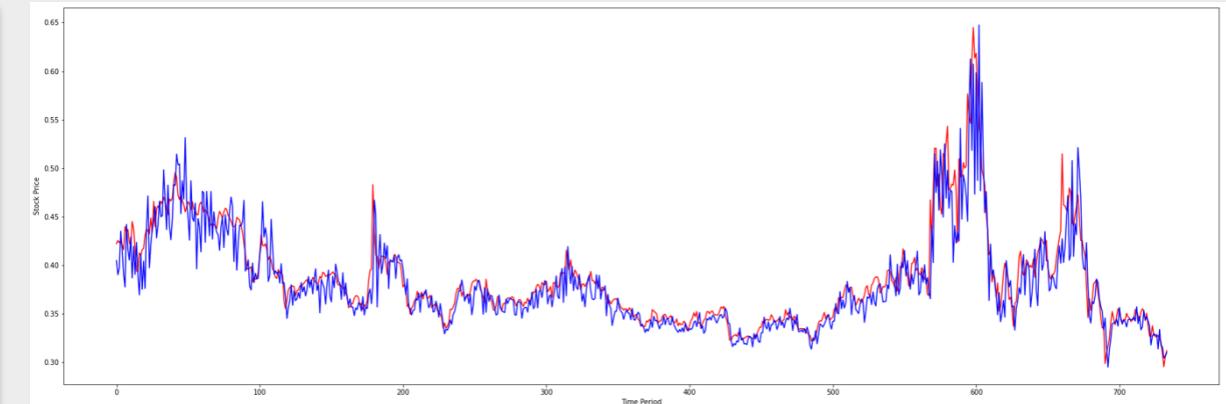
- keep\_prob = 0.9, softsign



- keep\_prob = 0.7, softsign



- keep\_prob = 0.5, softsign



# 예측결과

변수 추가 전

keep_prob	활성함수	rmse	예측값	실제값
1	softsign	0.009	2787	
1	relu	0.01	2760	
1	tanh	0.008	2799	
0.9	softsign	0.011	2785	2840
0.7	softsign	0.014	2850	
0.7	relu	0.044	2711	
0.7	tanh	0.0151	2769	
0.5	softsign	0.0199	2827	

변수 추가 후

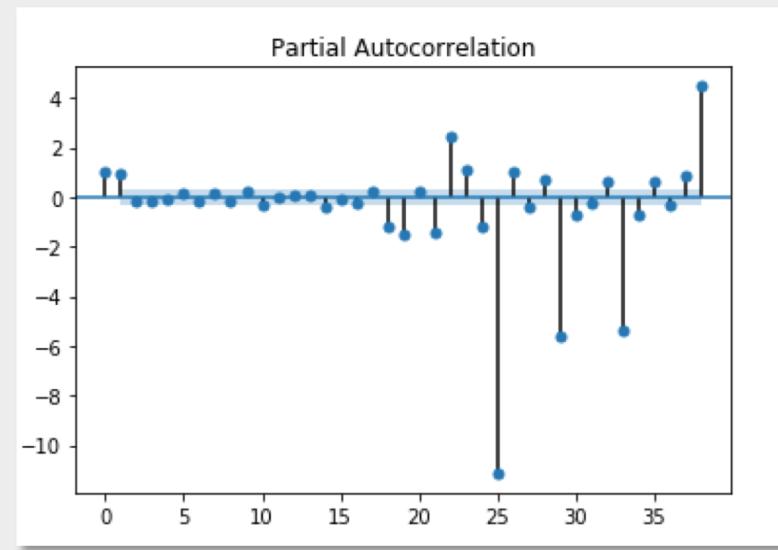
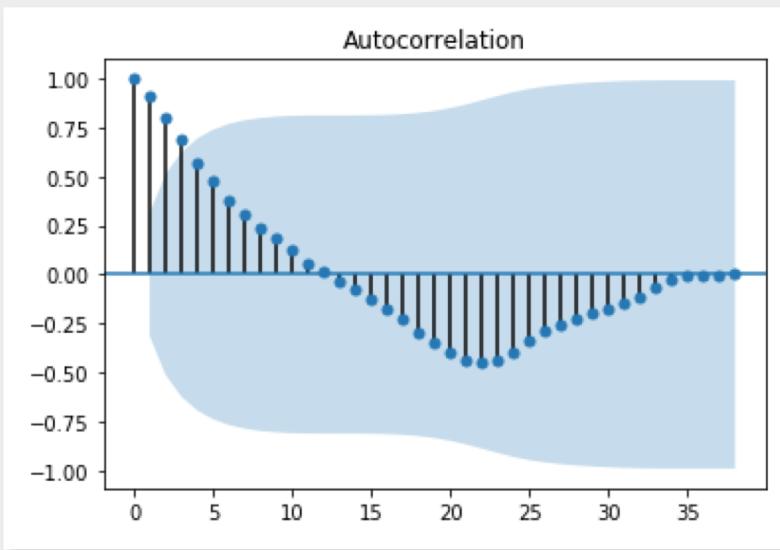
keep_prob	활성함수	rmse	예측값	실제값
1	softsign	0.014	2803	
1	relu	0.013	2821	
1	tanh	0.014	2817	
0.9	softsign	0.0163	2826	2840
0.7	softsign	0.0174	2825	
0.7	relu	0.04	2706	
0.7	tanh	0.027	2866	
0.5	softsign	0.022	2893	

# ARIMA

# ARIMA

특징: 종가(Close)만 가지고 예측함

- AR : 자기회귀, 이전 관측값의 오차항이 이후 관측값에 영향을 주는 모형
- I : Intgrated, 누적을 의미함. 차분을 이용하는 시계열모형들에 붙이는 표현
- MA : 이동평균. 관측값이 이전의 연속적인 오차항의 영향을 받는다는 모형



	Date	Adj Close
0	2017-12-01	3230.0
1	2017-12-04	3185.0
2	2017-12-05	3230.0
3	2017-12-06	3205.0
4	2017-12-07	3130.0
5	2017-12-08	3080.0
6	2017-12-11	3095.0
7	2017-12-12	3095.0
8	2017-12-13	3090.0
9	2017-12-14	3095.0
10	2017-12-15	3080.0
11	2017-12-18	3055.0
12	2017-12-19	3030.0
13	2017-12-21	2960.0
14	2017-12-22	2960.0

# ARIMA\_RMSE

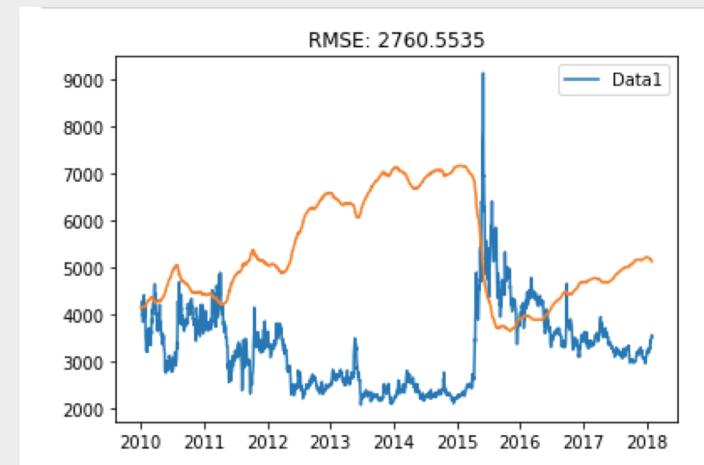
100101~180131은 약 2760

160101~180131은 약 423

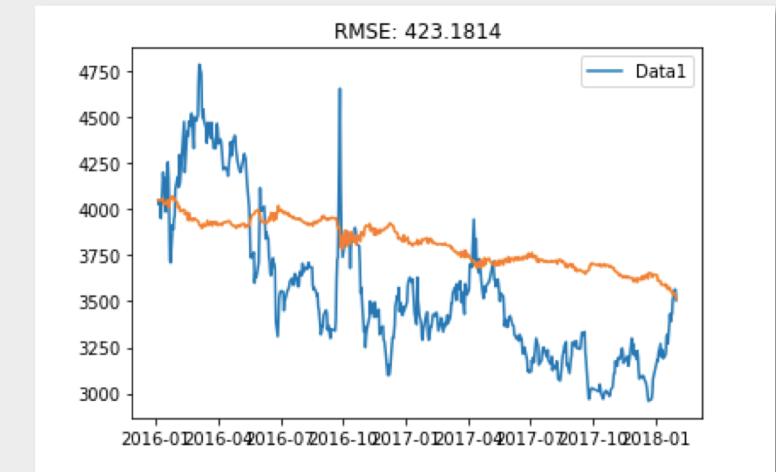
170101~180131은 약 340

171201~180131은 약 142

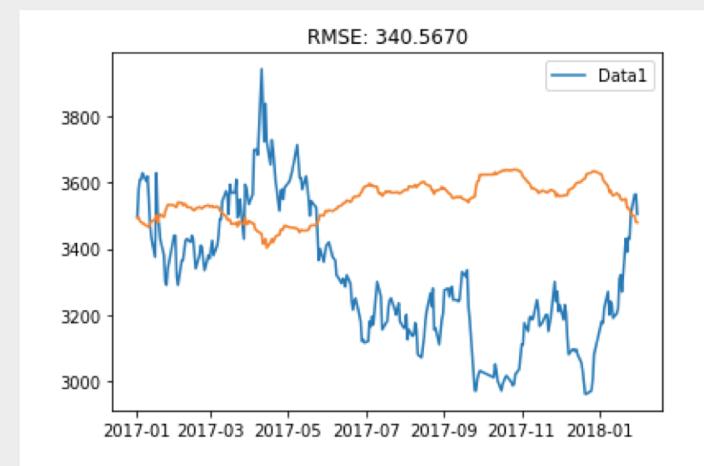
:기간이 줄 수록 RMSE가 작아진다



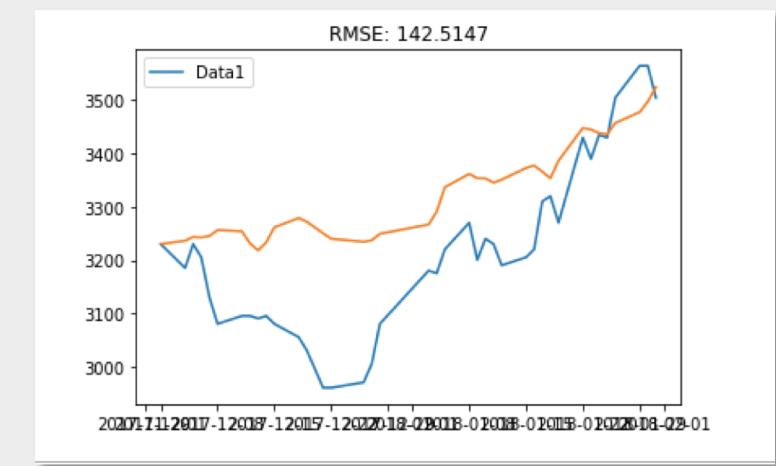
RMSE: 2760.5535



RMSE: 423.1814

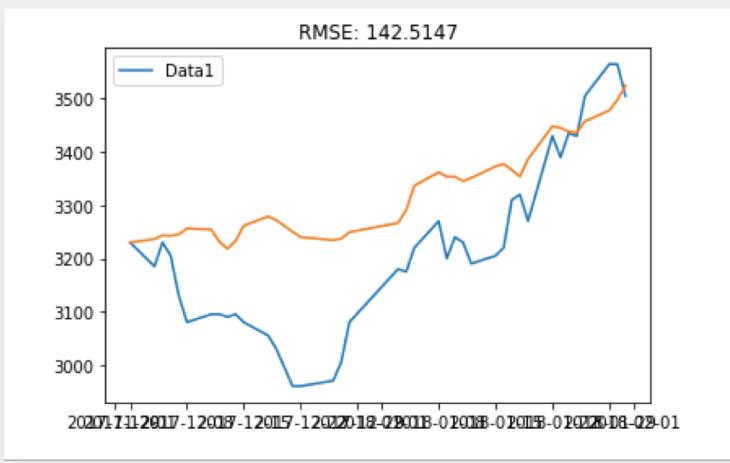


RMSE: 340.5670



RMSE: 142.5147

# ARIMA\_RMSE



RMSE: 142.5147  
→ 20171201 ~ 20180131 기

간의 종가(Close)를 학습시킴

```
In [39]: # ARIMA(3,1,2)
model = ARIMA(series, order = (3,1,2))
model_fit = model.fit(trend='nc', full_output = True, disp = 1)
print(model_fit.summary())
```

## ARIMA Model Results

Dep. Variable:	D.Adj Close	No. Observations:	38
Model:	ARIMA(3, 1, 2)	Log Likelihood	-201.288
Method:	css-mle	S.D. of innovations	45.934
Date:	Thu, 19 Mar 2020	AIC	414.577
Time:	02:17:26	BIC	424.402
Sample:	1	HQIC	418.072

	coef	std err	z	P> z	[0.025	0.975]
ar.L1.D.Adj Close	0.7457	0.279	2.675	0.012	0.199	1.292
ar.L2.D.Adj Close	-0.6473	0.243	-2.666	0.012	-1.123	-0.171
ar.L3.D.Adj Close	0.1136	0.231	0.491	0.627	-0.340	0.567
ma.L1.D.Adj Close	-0.8788	0.538	-1.635	0.112	-1.932	0.175
ma.L2.D.Adj Close	1.0000	1.055	0.948	0.350	-1.067	3.067
Roots						

	Real	Imaginary	Modulus	Frequency
AR.1	0.4991	-1.2743j	1.3686	-0.1906
AR.2	0.4991	+1.2743j	1.3686	0.1906
AR.3	4.7009	-0.0000j	4.7009	-0.0000
MA.1	0.4394	-0.8983j	1.0000	-0.1776
MA.2	0.4394	+0.8983j	1.0000	0.1776

## ARIMA\_결과

20180201의 종가를 예측

```
In [19]: fore = model_fit.forecast(steps=1)
print(fore)
# 예측값, stderr, upper bound, lower bound
(array([3528.58285722]), array([48.94399406]), array([[3432.6543916 , 3624.51132284]]))
```

날짜	종가
2018.02.01	3,540

20200319의 종가를 예측

```
In [99]: fore = model_fit.forecast(steps=1)
print(fore)
# 예측값, stderr, upper bound, lower bound
(array([5928.18713066]), array([733.87445142]), array([[4489.8196367 , 7366.55462462]]))
```

# Q & A