10만 전자 가즈아

6팀 PL 정기호
김명근
김혜진
신문혁

CONTENTS

SAMSUNG

SAMSUNG

CONTENTS_1 CONTENTS_2 CONTENTS_3 CONTENTS_4

1. 리서치 요약 : SAMSUNG Word_Cloud

1. 리서치 요약

삼성전자 사업보고서/2021.03.09

(단위 : 억원)

부 문		매출유형	품 목		제52기	제51기	제50기
CE 부문		제・상품, 용역 및 기타매출	TV, 모니터, 냉장고, 세탁기, 에어컨 등	계	481,733	453,228	426,498
IM 부문		제・상품, 용역 및 기타매출	HHP, 네트워크시스템, 컴퓨터 등	계	995,875	1,072,662	1,006,777
DS 부문	반도체 사업	제・상품, 용역 및 기타매출	DRAM, NAND Flash, 모바일AP 등	계	728,578	649,391	862,910
	DP 사업	제・상품, 용역 및 기타매출	스마트폰용 OLED, TV・모니터용 LCD 패널 등	계	305,857	310,539	324,650
	기타	-	-	계	△4,074	△4,750	△1,904
	부문 계				1,030,361	955,180	1,185,656
Harman 부문		제・상품, 용역 및 기타매출	디지털 콕핏, 텔레매틱스, 스피커 등	계	91,837	100,771	88,437
기 타		-	-	계	△231,736	△277,832	△269,654
합 계					2,368,070	2,304,009	2,437,714

2. 탐색적 데이터 분석

[Data]

```
df = yf.download('005930.KS', start='2016-01-01', end='2021-08-31')
df
```

	Open	High	Low	Close	Adj Close	Volume
Date						
2016-01-04	25200.0	25200.0	24100.0	24100.0	20770.701172	15346950
2016-01-05	24040.0	24360.0	23720.0	24160.0	20822.406250	10800100
2016-01-06	24160.0	24160.0	23360.0	23500.0	20253.585938	18337600
2016-01-07	23320.0	23660.0	23020.0	23260.0	20046.742188	14119400
2016-01-08	23260.0	23720.0	23260.0	23420.0	20184.638672	12888150
...	...	...	...	...	...	...
2021-08-25	76200.0	76600.0	74900.0	75700.0	75700.000000	22319664
2021-08-26	76100.0	76200.0	74600.0	74600.0	74600.000000	16671494
2021-08-27	74300.0	75000.0	73800.0	74300.0	74300.000000	15172748
2021-08-30	75400.0	75500.0	74200.0	74600.0	74600.000000	12686999
2021-08-31	74900.0	76700.0	74300.0	76700.0	76700.000000	24630370

1384 rows × 6 columns

2. 탐색적 데이터 분석

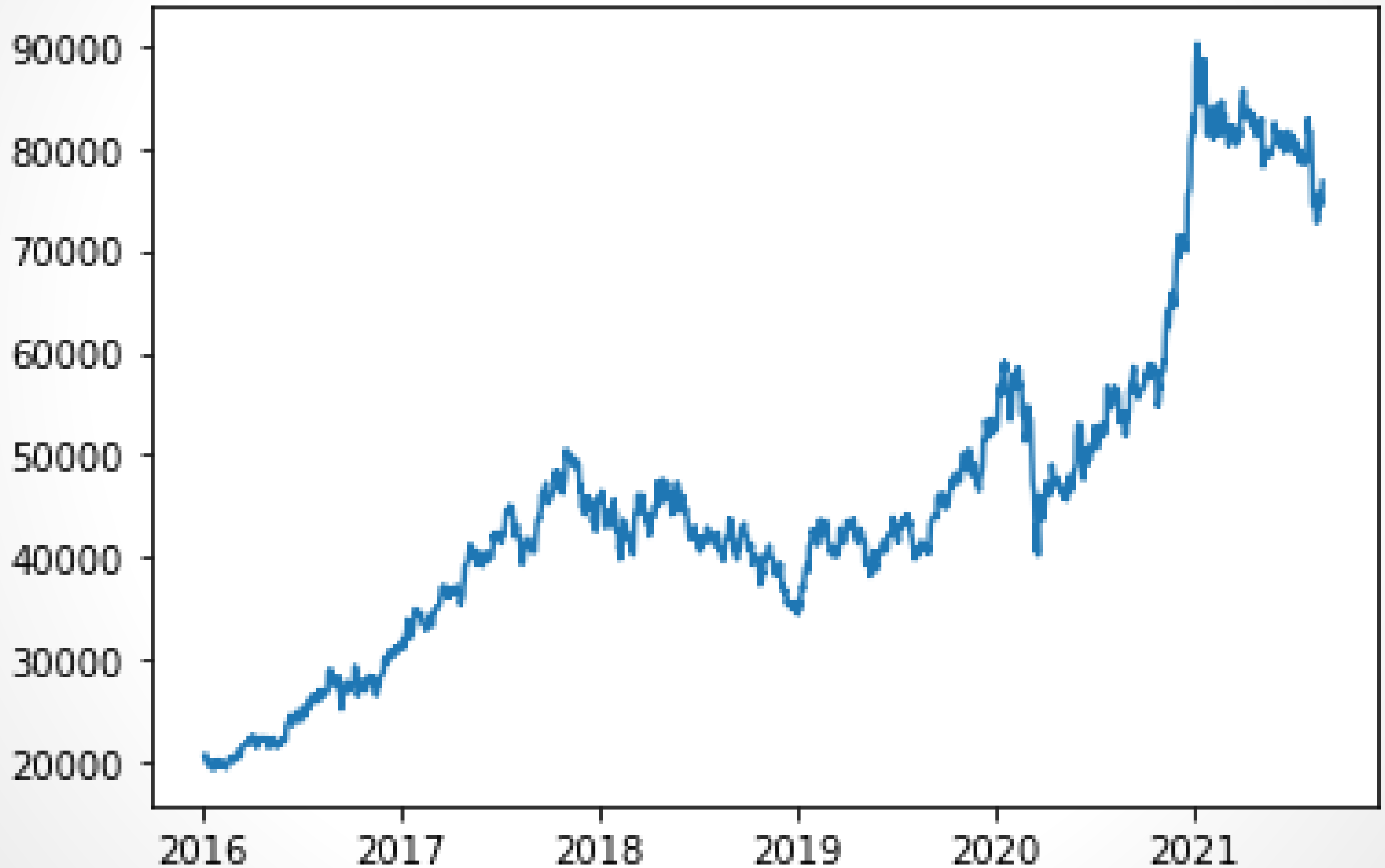
[Data]

```
df.describe()    # 삼성 주가
```

	Open	High	Low	Close	Adj Close	Volume
count	1384.000000	1384.000000	1384.000000	1384.000000	1384.000000	1.384000e+03
mean	49500.585260	49998.945087	49020.223988	49504.855491	46008.932914	1.463016e+07
std	15467.569951	15604.234335	15329.013050	15438.383127	16483.025364	8.280080e+06
min	21760.000000	22660.000000	21760.000000	22520.000000	19408.968750	0.000000e+00
25%	41770.000000	42100.000000	41487.500000	41840.000000	37275.034180	9.350674e+06
50%	47270.000000	47610.000000	46760.000000	47250.000000	43033.273438	1.247097e+07
75%	54855.000000	55500.000000	54300.000000	54925.000000	51502.114258	1.711926e+07
max	90300.000000	96800.000000	89500.000000	91000.000000	90198.078125	9.030618e+07

2. 탐색적 데이터 분석

[Data Plot]



3. Prophet

[열 변환]

	ds	y
0	2016-01-04	20770.701172
1	2016-01-05	20822.406250
2	2016-01-06	20253.585938
3	2016-01-07	20046.742188
4	2016-01-08	20184.638672
...	...	...
1379	2021-08-25	75700.000000
1380	2021-08-26	74600.000000
1381	2021-08-27	74300.000000
1382	2021-08-30	74600.000000
1383	2021-08-31	76700.000000

1384 rows × 2 columns

>>>

[ADF Test]

```
adf_test(df_samsung.y)
```

```
Test Statistic                  -0.765326
p-value                          0.829096
# of Lags Used                  18.000000
# of Observations Used        1365.000000
Critical Value (1%)             -3.435150
Critical Value (5%)             -2.863660
Critical Value (10%)            -2.567899
dtype: float64
```

[KPSS Test]

```
kpss_test(df_samsung.y)
```

```
Test Statistic            4.190858
p-value                   0.010000
# of Lags                24.000000
Critical Value (10%)      0.347000
Critical Value (5%)       0.463000
Critical Value (2.5%)     0.574000
Critical Value (1%)       0.739000
dtype: float64
```

3. Prophet

[Data Set 설정]

```
df_samsung_train = df_samsung[df_samsung['ds']<='2021-07-31']
df_samsung_test = df_samsung[df_samsung['ds']>'2021-07-31']
```

[Prophet Model 설정]

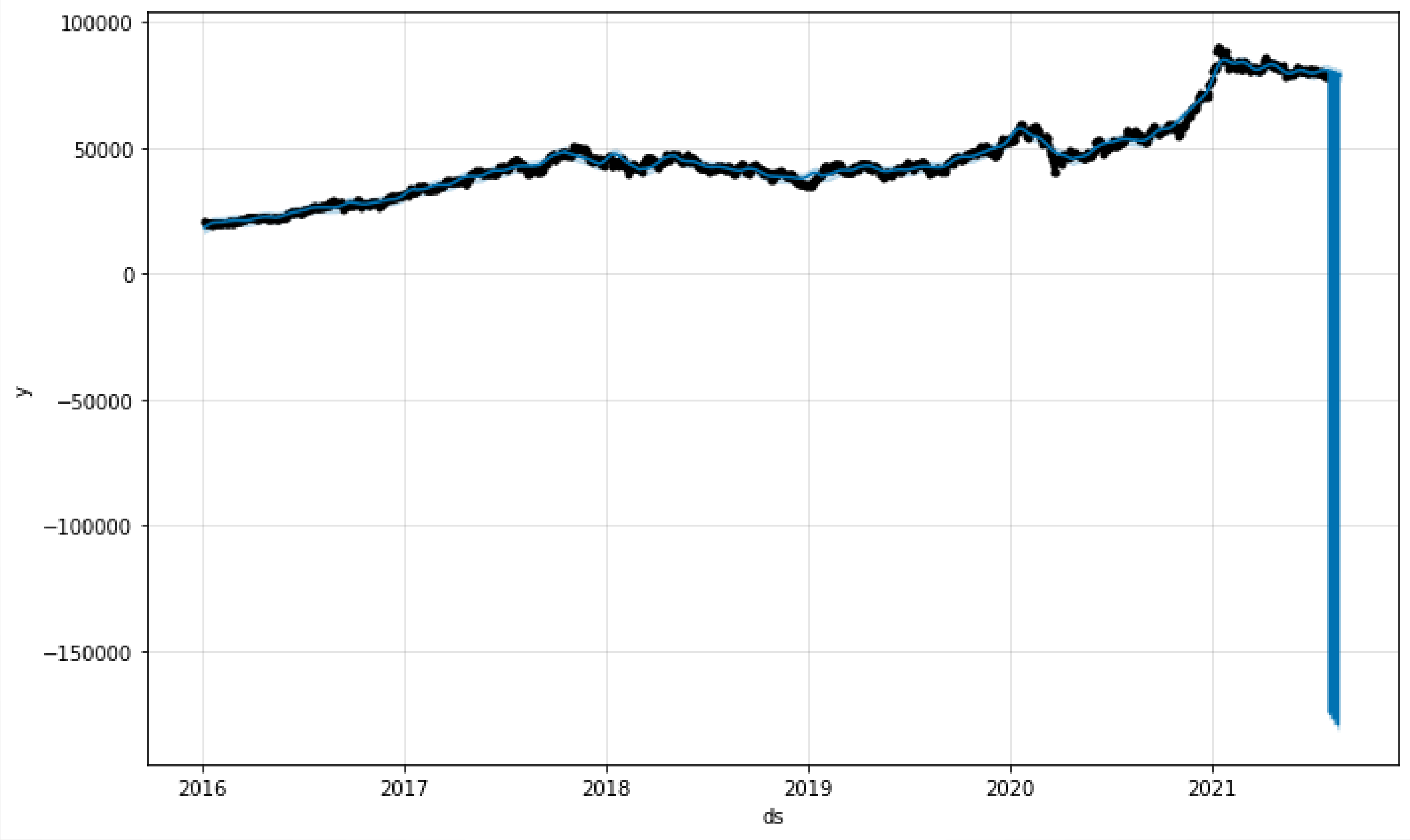
```
model = Prophet(seasonality_mode='multiplicative')
model.fit(df_samsung_train)
```

[1 Month 예측]

```
df_future1 = model.make_future_dataframe(periods=21, freq='d')
df_forecast1 = model.predict(df_future1)
```

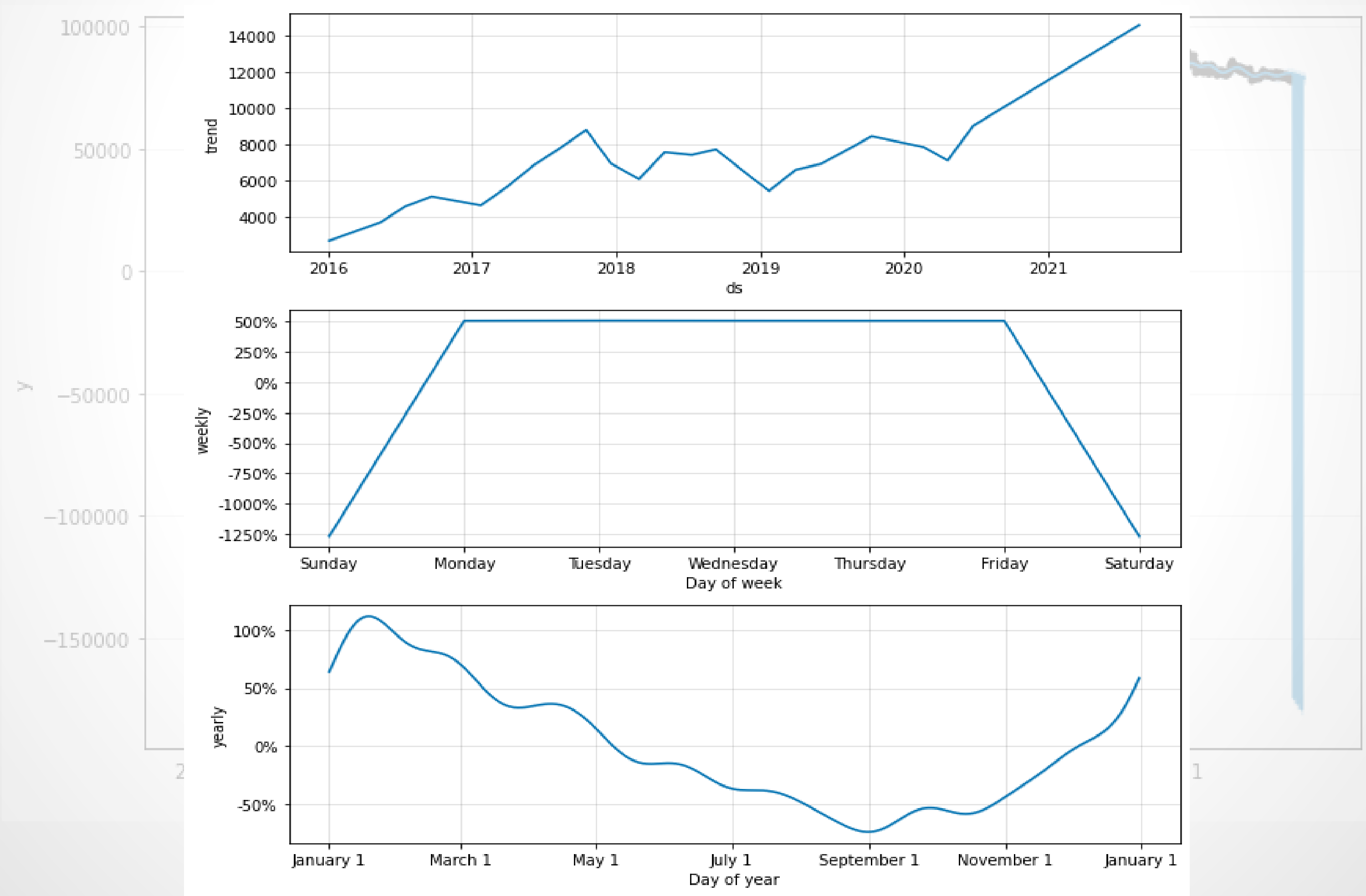
3. Prophet

[1 Month 예측 결과]



3. Prophet

[1 Month 예측 결과]



3. Prophet

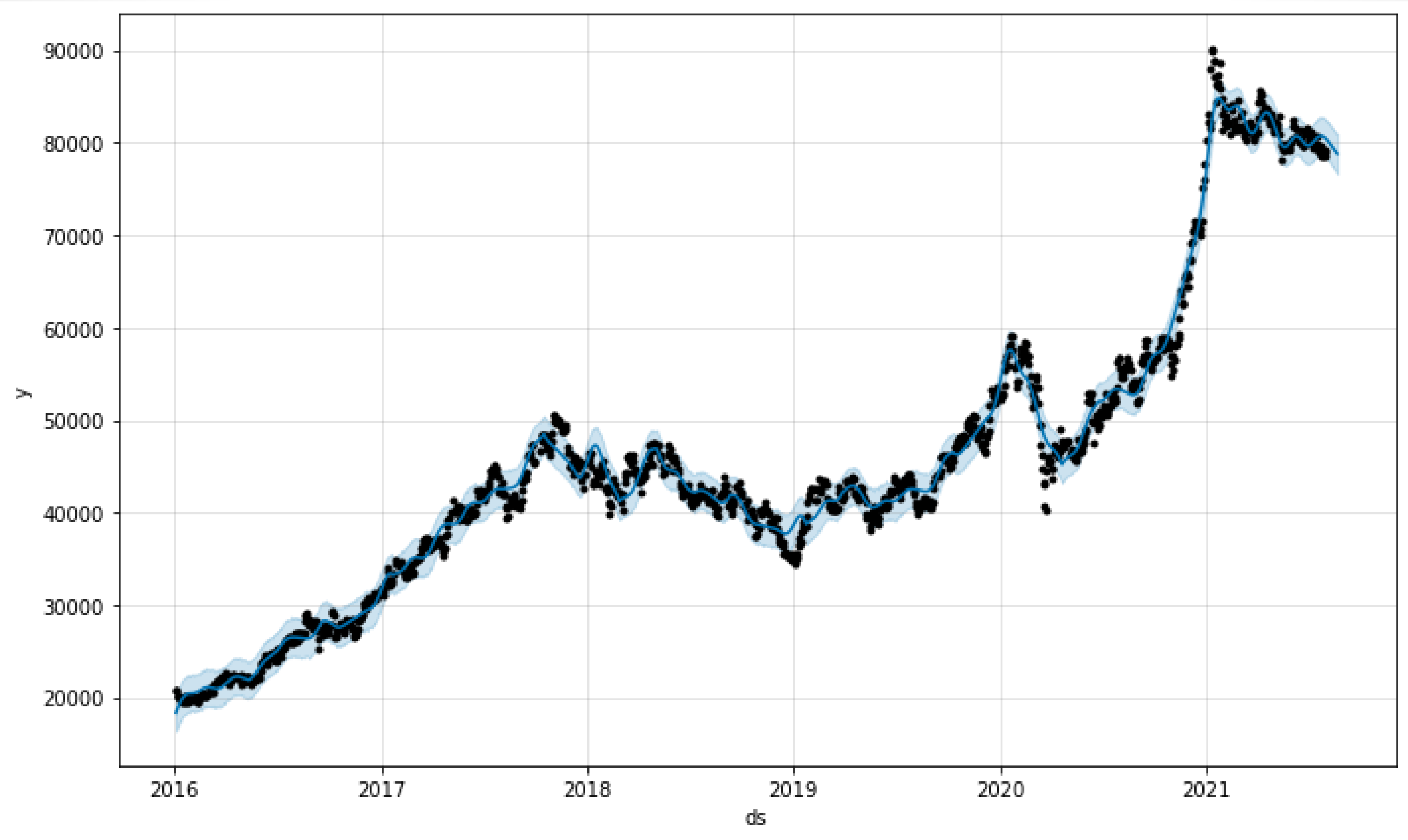
[weekly seasonality False 로 Prophet Model 설정]

```
prophet = Prophet(seasonality_mode = 'multiplicative',
                  yearly_seasonality=True,
                  weekly_seasonality=False,
                  daily_seasonality=False,
                  changepoint_prior_scale=0.5)
prophet.fit(df_samsung_train)
```

```
df_future = prophet.make_future_dataframe(periods=21, freq='d')
df_forecast = prophet.predict(df_future)
model.plot(df_forecast)
plt.tight_layout()
plt.show()
```

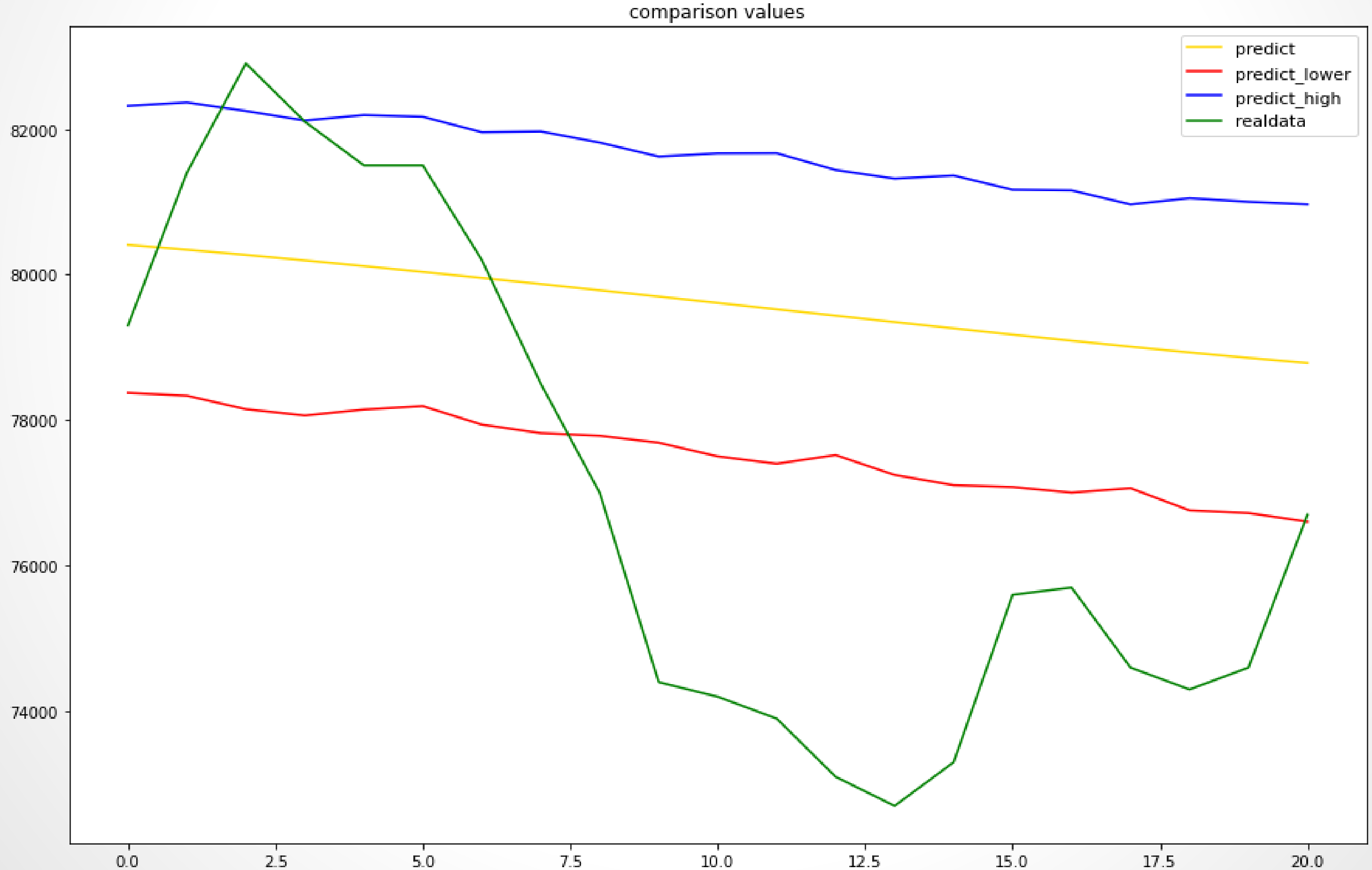
3. Prophet

[1 Month 예측 결과 : weekly seasonality False]



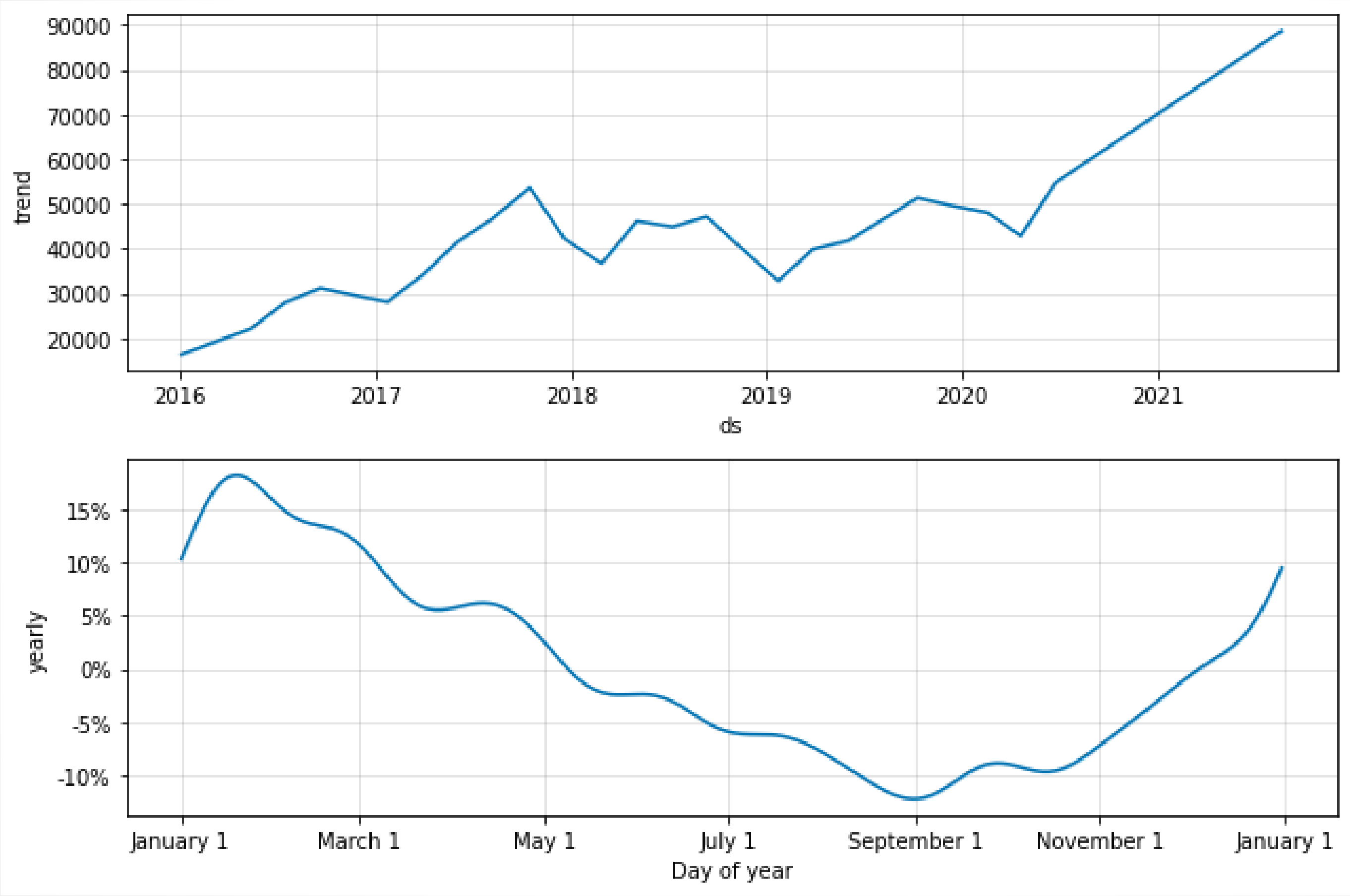
3. Prophet

[1 Month 예측 결과 : weekly seasonality False]



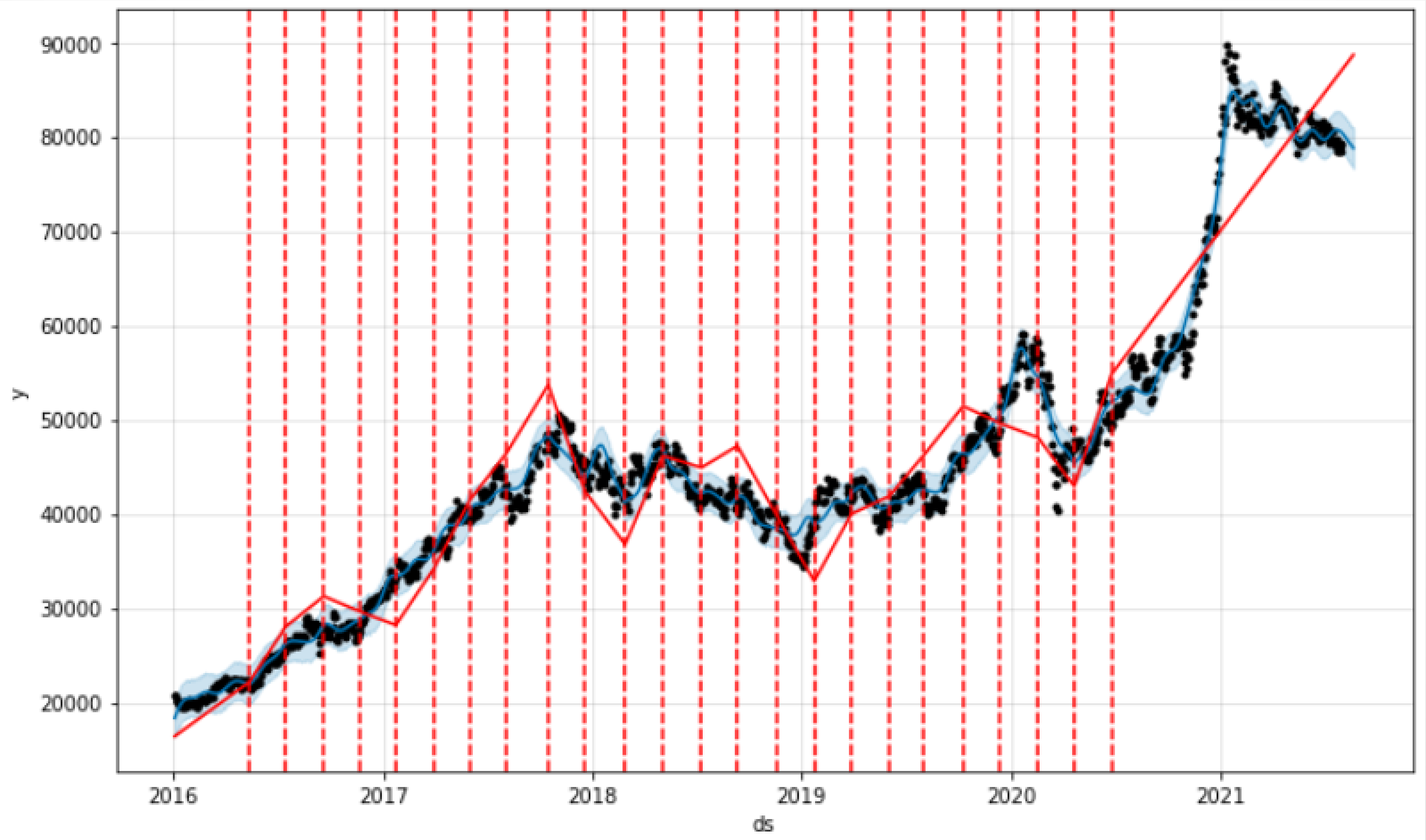
3. Prophet

[1 Month 예측 결과 : weekly seasonality False]



3. Prophet

[Change Point]



3. Prophet

[결과치 조정 – 상한가 및 하한가 설정]

```
# 상한가 설정
df_samsung_train['cap'] = 90000

#하한가 설정
df_samsung_train['floor'] = 10000

# 상한가 적용을 위한 파라미터를 다음과 같이 설정
re_prophet = Prophet(seasonality_mode = 'multiplicative',
                     growth = 'logistic',
                     yearly_seasonality = True,
                     weekly_seasonality = False,
                     daily_seasonality = False,
                     changepoint_prior_scale = 0.5)

re_prophet.fit(df_samsung_train)


# 21일 기간 예측
df_future2 = re_prophet.make_future_dataframe(periods = 21, freq = 'd')


# 상한가 하한가 설정

df_future2['cap'] = 90000
df_future2['floor'] = 10000
```

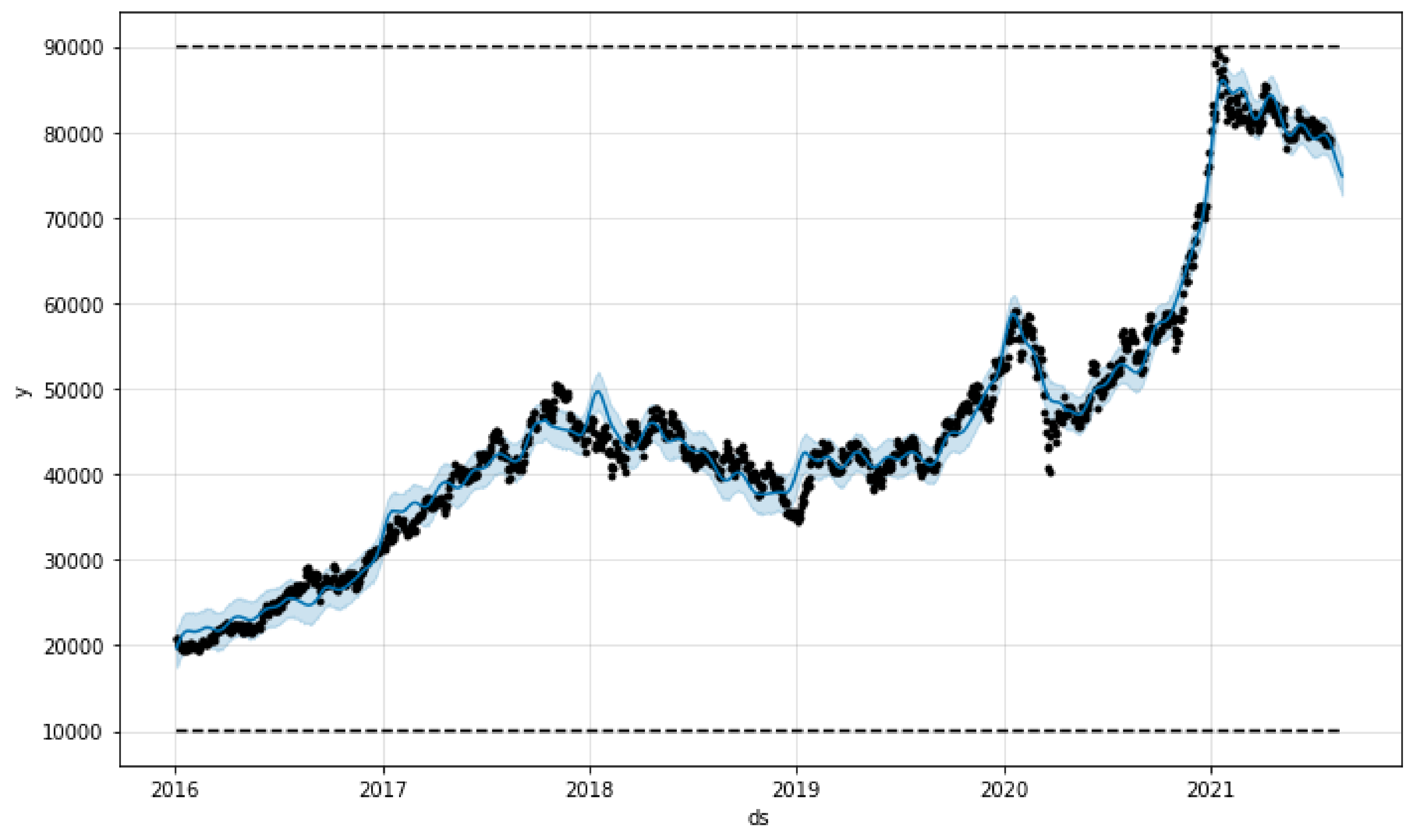
3. Prophet

[상한선 하한선 데이터 제거]

```
df_del_samsung = df_samsung_train
df_del_samsung.loc[df_samsung_train['y'] > 90000, 'y'] = None
df_del_samsung.loc[df_samsung_train['y'] < 10000, 'y'] = None

# prophet 모델 학습
rere_prophet = Prophet(seasonality_mode = 'multiplicative',
                       growth = 'logistic',
                 yearly_seasonality = True,
                 weekly_seasonality = False,
                 daily_seasonality = False,
                 changepoint_prior_scale = 0.5)

rere_prophet.fit(df_del_samsung)

# 21일 예측
df_future3 = rere_prophet.make_future_dataframe(periods = 21, freq = 'd')

# 상한가 하한가 설정
df_future3['cap'] = 90000
df_future3['floor'] = 10000
```

3. Prophet

[상한선 하한선 데이터 제거 plot]



3. Prophet

[Outlier 설정 후 Changepoint 설정]

```
fig = rere_prophet.plot(df_forecast3)
add_changepoints_to_plot(fig.gca(), rere_prophet, df_forecast3)
plt.show()
```

3. Prophet

[1 Month 예측 결과 : 상한선 하한선 데이터 제거]

```
df_forecast3[['ds','yhat', 'yhat_lower', 'yhat_upper']].tail(5)
```

	ds	yhat	yhat_lower	yhat_upper
1379	2021-08-16	75525.958015	73345.741464	77663.499368
1380	2021-08-17	75350.012784	73294.413443	77502.606011
1381	2021-08-18	75182.379422	73124.931138	77377.627010
1382	2021-08-19	75024.159061	72697.490991	77107.833837
1383	2021-08-20	74876.470098	72603.623330	77179.442031

```
pred_fbprophet_y_1 = df_forecast3.yhat.values[-21:]
```

3. Prophet

[결과 비교 분석]

```
df_test = pd.DataFrame({'FBprophet predict':  pred_fbprophet_y,
                        'FBprophet predict remove outlier':pred_fbprophet_y_1,
                        'real data':test_y})
```

```
df_test.tail(5)
```

	FBprophet predict	FBprophet predict remove outlier	real data
16	79069.492181	75525.958015	75700.0
17	78984.675650	75350.012784	74600.0
18	78902.163800	75182.379422	74300.0
19	78822.781641	75024.159061	74600.0
20	78747.471330	74876.470098	76700.0

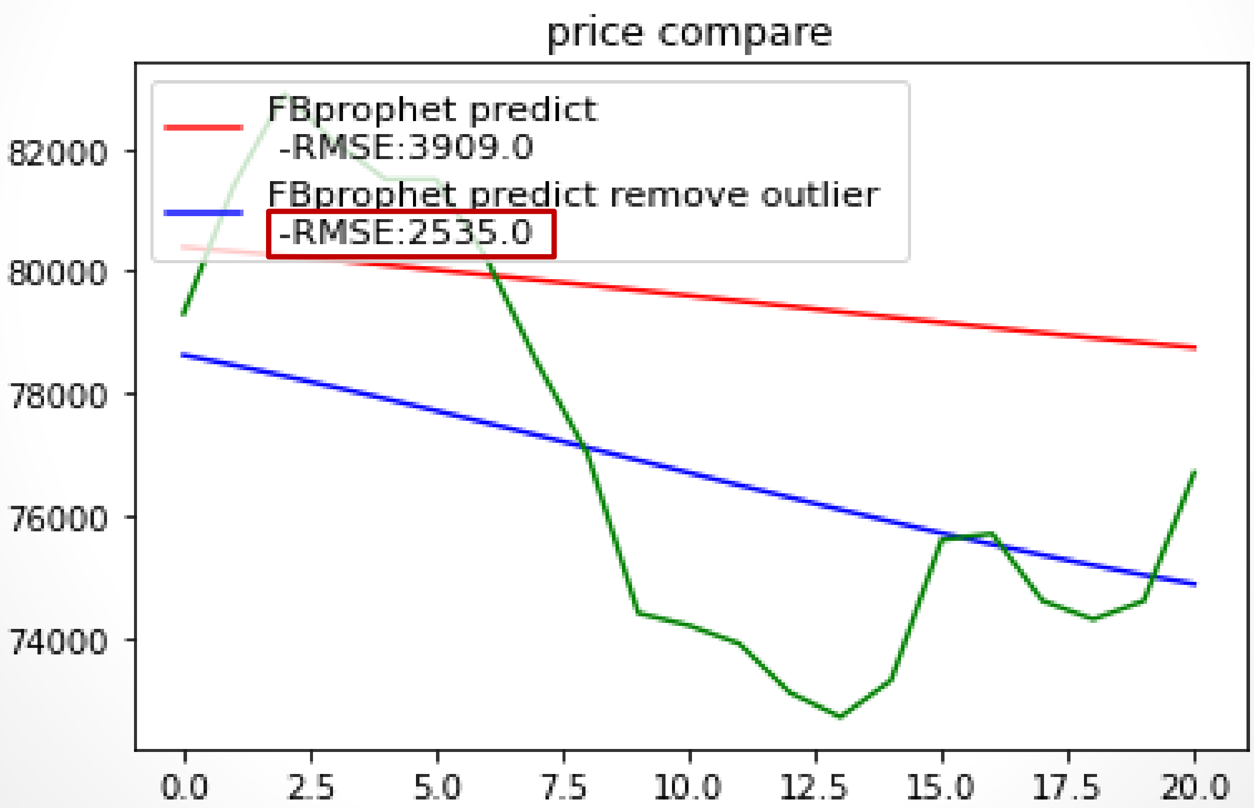
3. Prophet

[RMSE 검증]

```
from sklearn.metrics import mean_squared_error, r2_score
from math import sqrt

plt.figure(figsize=(20, 15))

# fbprophet 모델의  rmse
rmse_fbprophet = sqrt(mean_squared_error(pred_fbprophet_y, test_y))

# 전처리 진행한 fbprophet 모델의  rmse
rmse_fbprophet_1 = sqrt(mean_squared_error(pred_fbprophet_y_1, test_y))
```

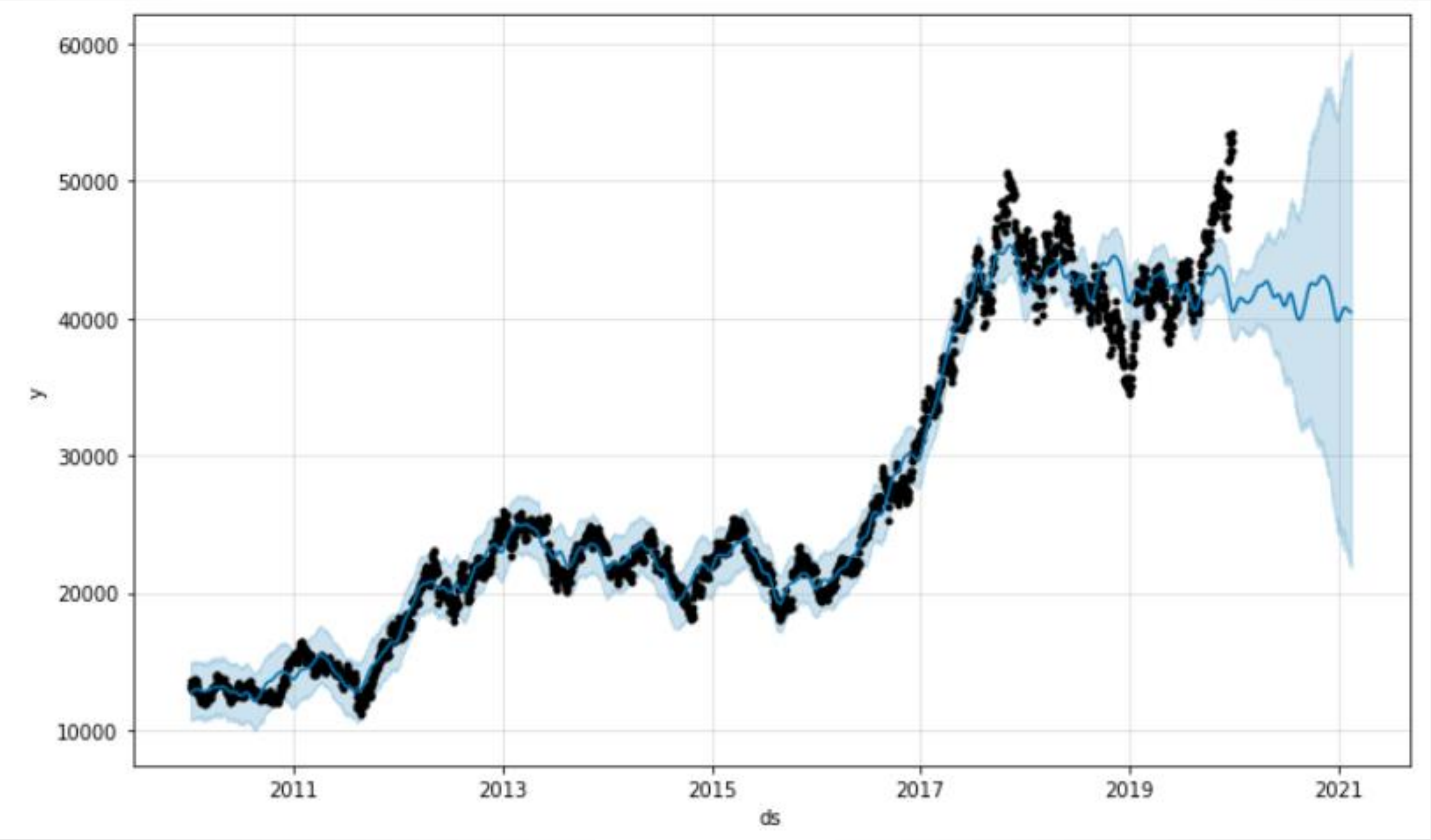
3. Prophet

[RMSE 검증 결과 Plot]



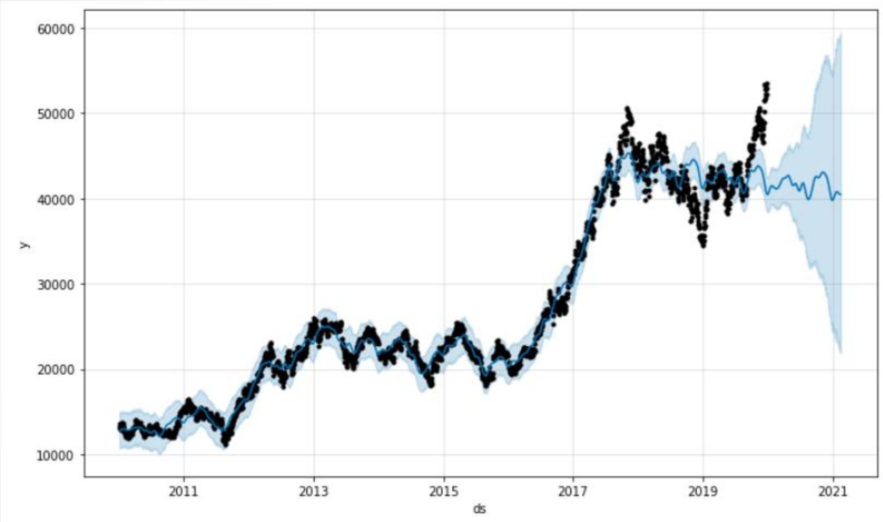
3. Prophet

[2010~2021 Data / 1Year 예측]



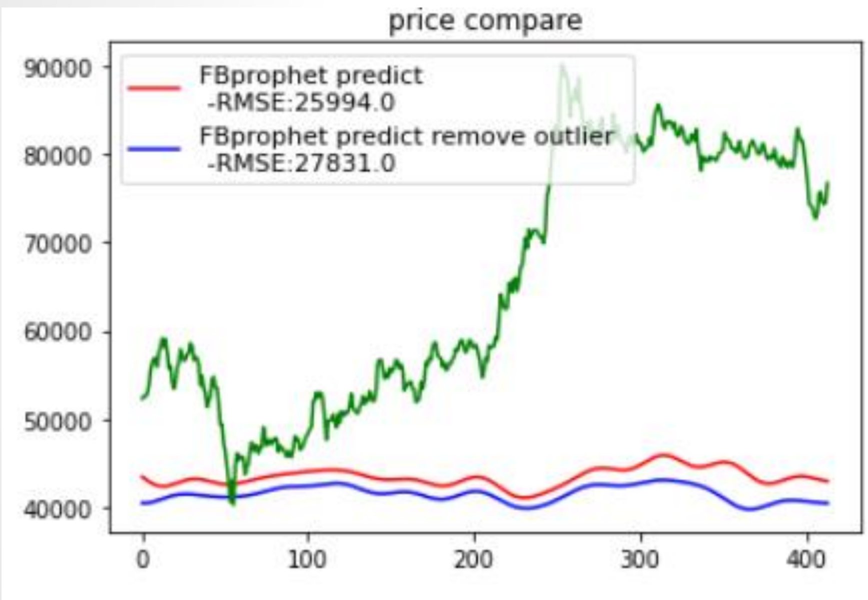
3. Prophet

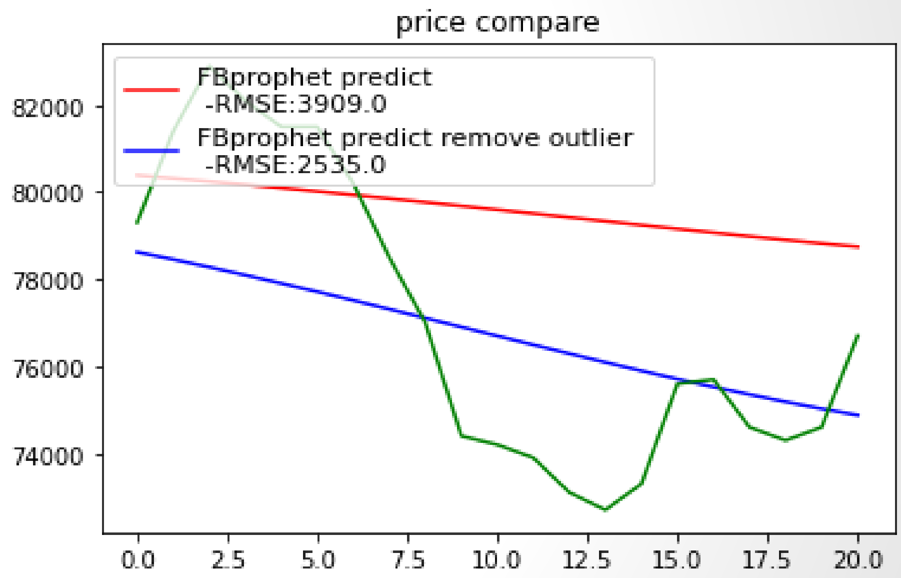
[Prophet Model 비교]



3. Prophet

[Prophet Model 비교]

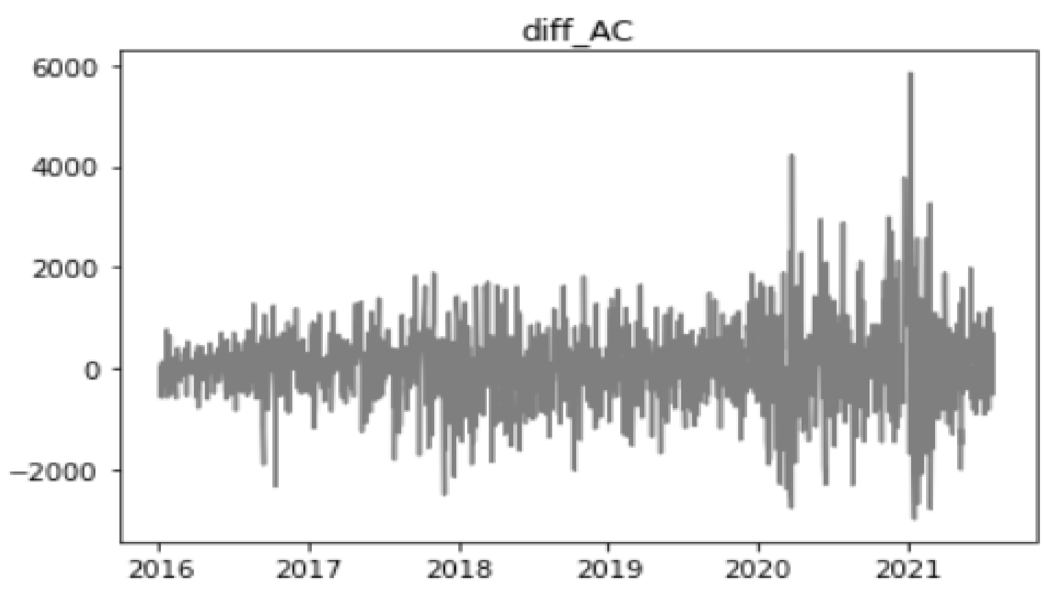




3. ARIMA

[Differencing]

```
diff_ACm = df_samsung['Adj Close'].diff(1)
diff_AC = diff_ACm.fillna(0)
plt.plot(diff_AC, color='black', alpha=0.5)
plt.title('diff_AC')
plt.show()
```



3. ARIMA

[Differencing한 결과로 adf_test / kpss_test]

```
adf_test(diff_AC)
```

```
Results of Dickey-Fuller Test:
Test Statistic                 -8.169455e+00
p-value                         8.701976e-13
#Lags Used                      1.700000e+01
Number of Observations Used     1.345000e+03
Critical Value (1%)            -3.435221e+00
Critical Value (5%)            -2.863691e+00
Critical Value (10%)           -2.567915e+00
dtype: float64
```

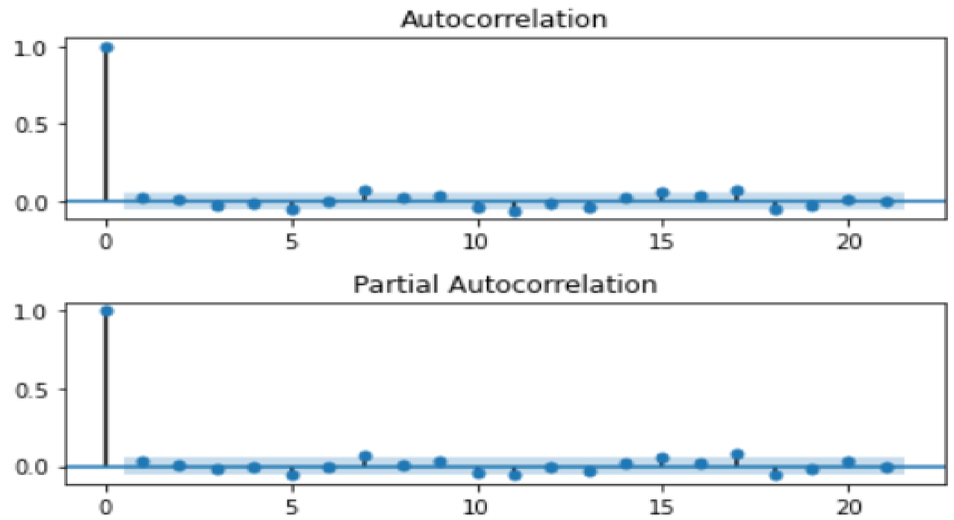
```
kpss_test(diff_AC)
```

```
Results of KPSS Test:
Test Statistic            0.096151
p-value                   0.100000
Lags Used                 3.000000
Critical Value (10%)      0.347000
Critical Value (5%)       0.463000
Critical Value (2.5%)     0.574000
Critical Value (1%)       0.739000
dtype: float64
```

3. ARIMA

[Data ACF / PACF]

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

N_LAGS = 21
SIGNIFICANCE_LEVEL = 0.05

fig, ax = plt.subplots(2, 1)
plot_acf(diff_AC, ax=ax[0], lags=N_LAGS, alpha=SIGNIFICANCE_LEVEL)
plot_pacf(diff_AC, ax=ax[1], lags=N_LAGS, alpha=SIGNIFICANCE_LEVEL)
plt.tight_layout()
```



3. ARIMA

[ACF, PACF TEST for diagnosis ARIMA(p,d,q)]

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.D.Adj Close	0.1931	0.480	0.402	0.688	-0.748	1.134
ma.L1.D.Adj Close	-0.1616	0.482	-0.335	0.738	-1.107	0.784

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.D.Adj Close	-0.5563	0.409	-1.359	0.174	-1.359	0.246
ma.L1.D.Adj Close	0.5876	0.409	1.435	0.152	-0.215	1.390
ma.L2.D.Adj Close	0.0371	0.028	1.307	0.191	-0.019	0.093

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.D.Adj Close	-0.5195	0.429	-1.210	0.227	-1.361	0.322
ar.L2.D.Adj Close	0.0357	0.028	1.276	0.202	-0.019	0.090
ma.L1.D.Adj Close	0.5501	0.429	1.283	0.200	-0.291	1.391

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.D.Adj Close	1.2928	0.057	22.592	0.000	1.181	1.405
ar.L2.D.Adj Close	-0.9307	0.035	-26.453	0.000	-1.000	-0.862
ma.L1.D.Adj Close	-1.2556	0.069	-18.103	0.000	-1.391	-1.120
ma.L2.D.Adj Close	0.8925	0.041	21.749	0.000	0.812	0.973

3. ARIMA

[ARIMA Model : Data Differencing]

```
model = ARIMA(df_samsung['Adj Close'], order=(2,1,2))
model_fit1 = model.fit(trend='nc',full_output=True, disp=True, start_ar_lags=4)
print(model_fit1.summary())
```

[ARIMA Model: Data 차분 _ Summary]

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:            D.Adj Close   No. Observations:                 1362
Model:                 ARIMA(2, 1, 2)   Log Likelihood              -11016.419
Method:                       css-mle   S.D. of innovations            787.948
Date:                Fri, 03 Sep 2021   AIC                          22042.839
Time:                        06:09:16   BIC                          22068.922
Sample:                             1   HQIC                         22052.603

=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
ar.L1.D.Adj Close     1.2928      0.057     22.592      0.000       1.181       1.405
ar.L2.D.Adj Close    -0.9307      0.035    -26.453      0.000      -1.000      -0.862
ma.L1.D.Adj Close    -1.2556      0.069    -18.103      0.000      -1.391      -1.120
ma.L2.D.Adj Close     0.8925      0.041     21.749      0.000       0.812       0.973
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            0.6945           -0.7695j            1.0365           -0.1331
AR.2            0.6945           +0.7695j            1.0365            0.1331
MA.1            0.7034           -0.7910j            1.0585           -0.1343
MA.2            0.7034           +0.7910j            1.0585            0.1343
-----------------------------------------------------------------------------
```

3. ARIMA

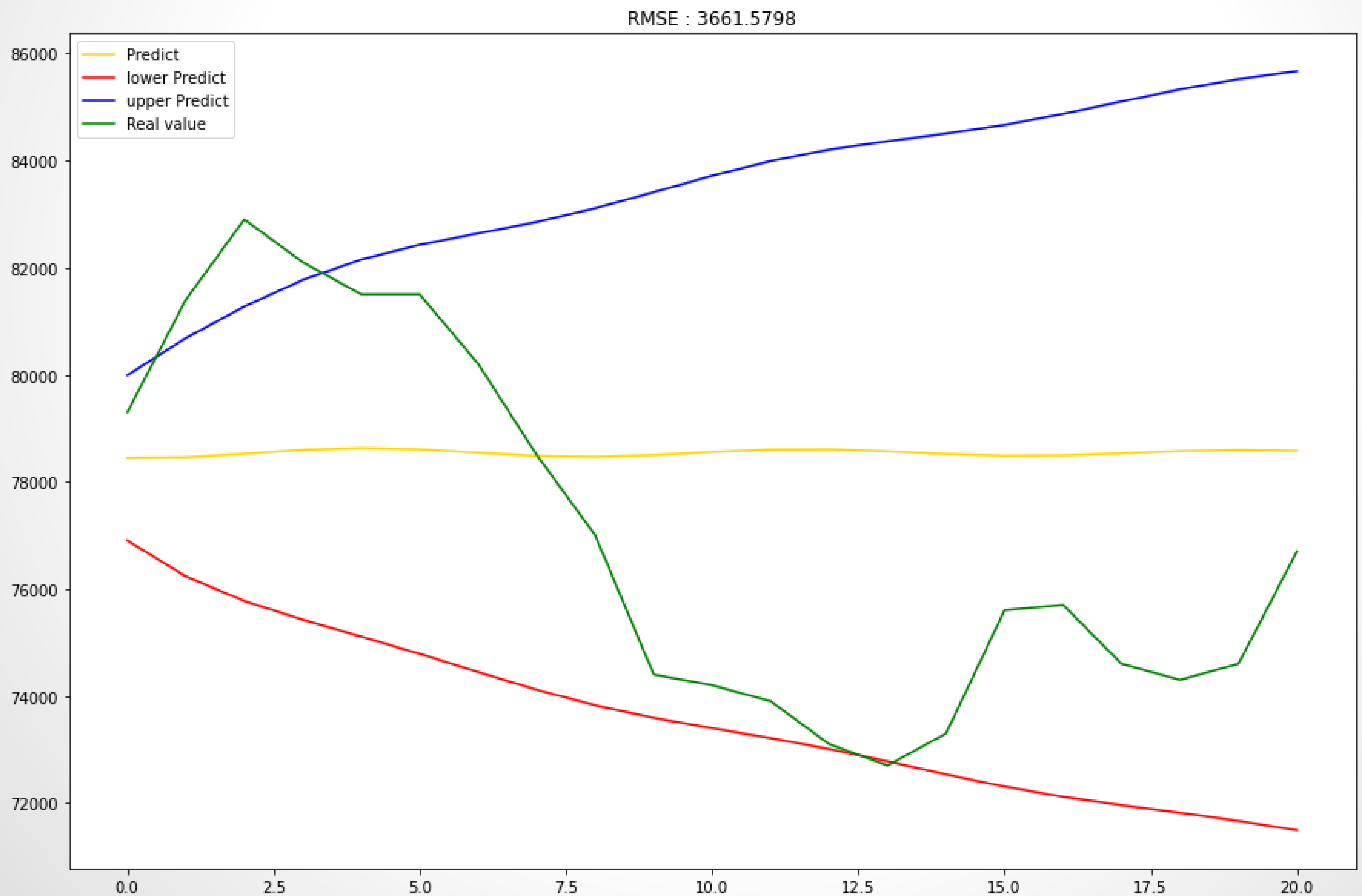
[ARIMA Model : Data Differencing]

```
# 마지막 21일의 예측 데이터 (2021-08-01 ~ 2021-08-31)
pred_arima_AC1 = forecast_data1[0].tolist()
# 해당 forecast의 [0]번째 array에 예측된 가격이 나옴.

# 실제 21일의 데이터 (2021-08-01 ~ 2021-08-31)
test_samsung = yf.download('005930.KS', '2021-08-01', '2021-08-31')['Adj Close']
test_AC1 = pd.DataFrame(test_samsung)

# 마지막 21일의 예측 데이터 최소값
pred_AC1_lower = []
# 마지막 21일의 예측 데이터 최대값
pred_AC1_upper = []

for lower_upper in forecast_data[2]:
    lower = lower_upper[0]
    upper = lower_upper[1]
    pred_AC1_lower.append(lower)
    pred_AC1_upper.append(upper)
```

```
pred_arima_AC1 = pd.DataFrame(np.array(pred_arima_AC1).astype(int))
pred_AC1_lower = np.array(pred_AC1_lower).astype(int)
pred_AC1_upper = np.array(pred_AC1_upper).astype(int)
test_AC1 = test_AC1.values
```

3. ARIMA

[ARIMA Model 결과 Plot]



3. ARIMA

[Data log, Differencing 시행]

```
ln_AC = np.log(df_samsung['Adj Close'])
plt.plot(ln_AC, color='black', alpha=0.5)
plt.title('logscale_AC')
plt.show()
```

```
ln_diff_ACm = ln_AC.diff(1)
ln_diff_AC = ln_diff_ACm.fillna(0)
ln_diff_AC
```

3. ARIMA

[Data log, Differencing ADF / KPSS Test]

```
adf_test(ln_diff_AC)
```

```
Results of Dickey-Fuller Test:
Test Statistic                  -22.044447
p-value                           0.000000
#Lags Used                        2.000000
Number of Observations Used    1360.000000
Critical Value (1%)              -3.435167
Critical Value (5%)              -2.863668
Critical Value (10%)             -2.567903
dtype: float64
```

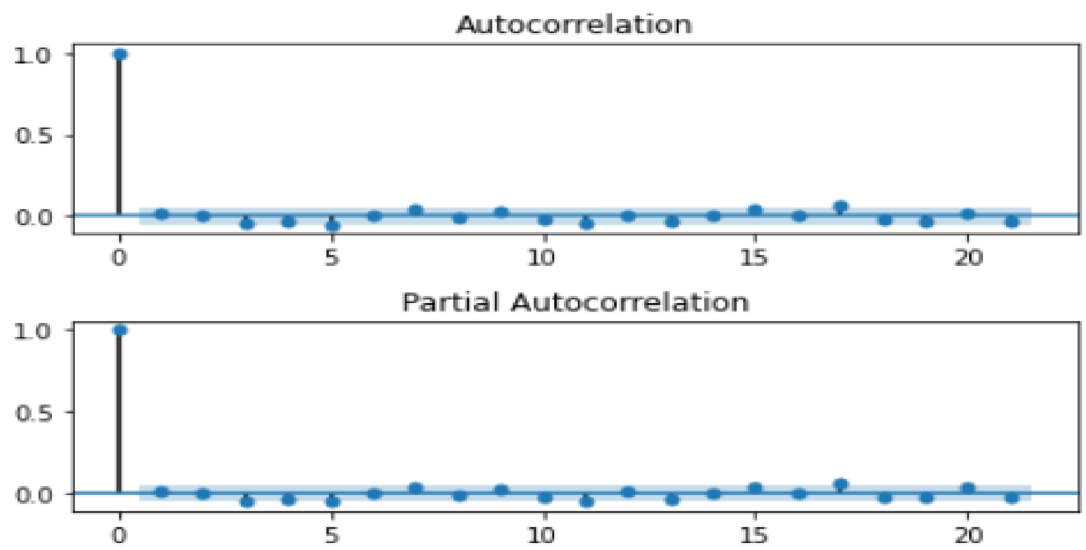
```
kpss_test(ln_diff_AC)
```

```
Results of KPSS Test:
Test Statistic             0.091651
p-value                    0.100000
Lags Used                  8.000000
Critical Value (10%)       0.347000
Critical Value (5%)        0.463000
Critical Value (2.5%)      0.574000
Critical Value (1%)        0.739000
dtype: float64
```

3. ARIMA

[Data log, Differencing ACF / PACF]

```
N_LAGS = 21
SIGNIFICANCE_LEVEL = 0.05 #음영값임....

fig, ax = plt.subplots(2, 1)
plot_acf(ln_diff_AC, ax=ax[0], lags=N_LAGS, alpha=SIGNIFICANCE_LEVEL)
plot_pacf(ln_diff_AC, ax=ax[1], lags=N_LAGS, alpha=SIGNIFICANCE_LEVEL)
plt.tight_layout()
```



3. ARIMA

[ACF, PACF TEST for diagnosis ARIMA(p,d,q)]

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0010	0.000	2.147	0.032	8.5e-05	0.002
ar.L1.D.Adj Close	-0.9921	0.012	-79.680	0.000	-1.017	-0.968
ma.L1.D.Adj Close	0.9950	0.010	102.521	0.000	0.976	1.014

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0010	0.000	2.125	0.034	7.61e-05	0.002
ar.L1.D.Adj Close	1.2023	0.026	45.933	0.000	1.151	1.254
ar.L2.D.Adj Close	-0.9665	0.024	-40.233	0.000	-1.014	-0.919
ma.L1.D.Adj Close	-1.1822	0.028	-42.497	0.000	-1.237	-1.128
ma.L2.D.Adj Close	0.9585	0.027	34.985	0.000	0.905	1.012

3. ARIMA

[ARIMA Model : Data log, Differencing]

```
model = ARIMA(ln_AC, order=(1,1,1))
model_fit2 = model.fit(trend='c',full_output=True, disp=True, start_ar_lags=2)
```

[ARIMA Model: Data 로그, 차분 _ Summary]

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:            D.Adj Close   No. Observations:                 1362
Model:                 ARIMA(1, 1, 1)   Log Likelihood                3635.981
Method:                       css-mle   S.D. of innovations              0.017
Date:                Fri, 03 Sep 2021   AIC                          -7263.962
Time:                        06:31:23   BIC                          -7243.095
Sample:                             1   HQIC                         -7256.150

=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 0.0010      0.000      2.147      0.032    8.5e-05       0.002
ar.L1.D.Adj Close    -0.9921      0.012    -79.680      0.000      -1.017      -0.968
ma.L1.D.Adj Close     0.9950      0.010    102.521      0.000       0.976       1.014
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -1.0080           +0.0000j            1.0080            0.5000
MA.1           -1.0051           +0.0000j            1.0051            0.5000
-----------------------------------------------------------------------------
```

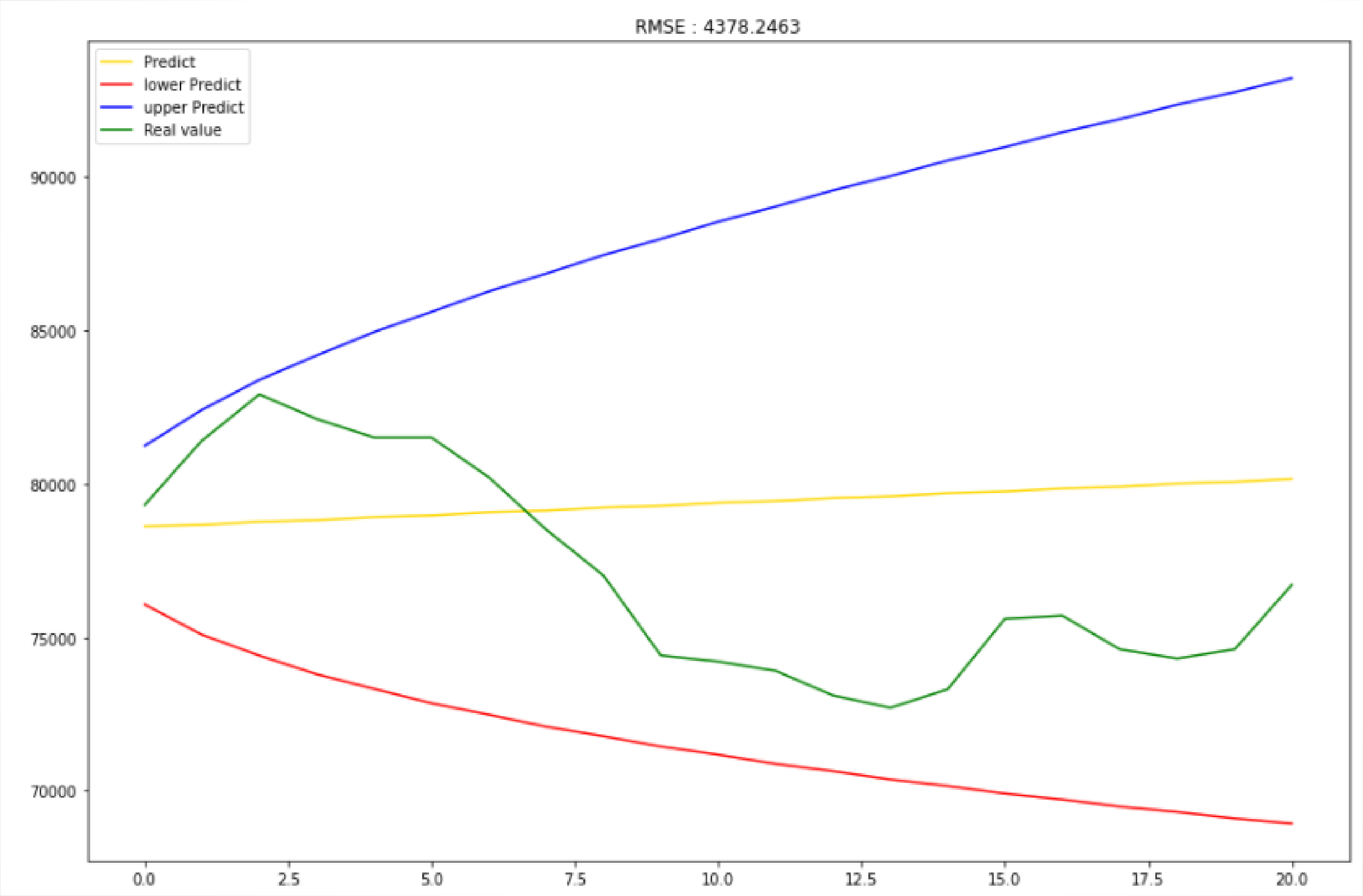
3. ARIMA

[1Month 예측: ARIMA Model (Data log, Differencing)]

```
# 마지막 21일의 예측 데이터 (2021-08-01 ~ 2021-08-31)
pred_arima_AC2 = np.exp(forecast_data2[0].tolist())

# 실제 21일의 데이터 (2021-08-01 ~ 2021-08-31)
test_samsung = yf.download('005930.KS', '2021-08-01', '2021-08-31')['Adj Close']
test_AC2 = pd.DataFrame(test_samsung)

# 마지막 21일의 예측 데이터 최소값
pred_AC2_lower = []
# 마지막 21일의 예측 데이터 최대값
pred_AC2_upper = []

for lower_upper in np.exp(forecast_data2[2]):
    lower = lower_upper[0]
    upper = lower_upper[1]
    pred_AC2_lower.append(lower)
    pred_AC2_upper.append(upper)
```

3. ARIMA

[1Month 예측: ARIMA Model (Data 로그, 차분)]

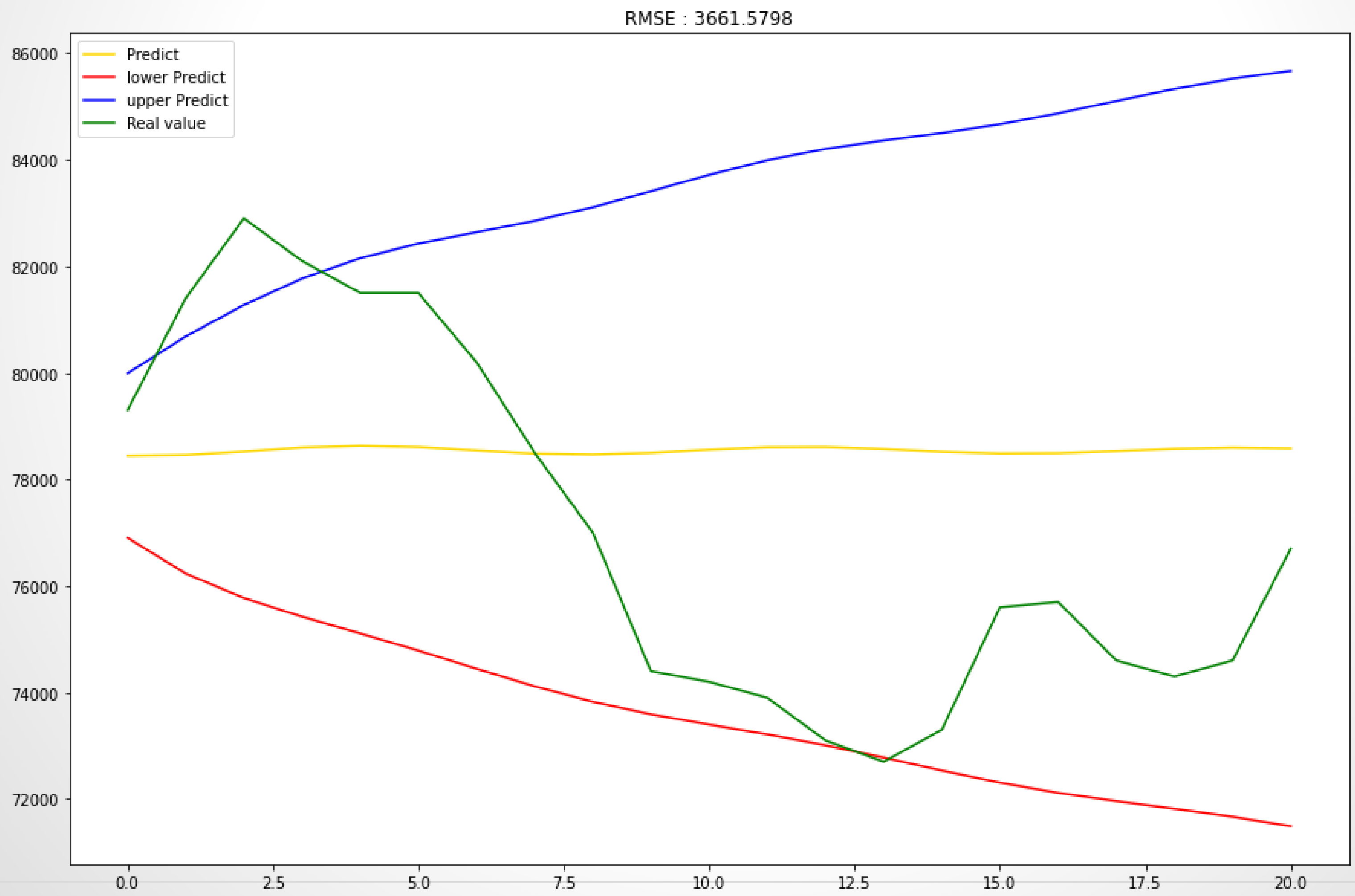


SAMSUNG

CONTENTS_1 CONTENTS_2 CONTENTS_3 CONTENTS_4

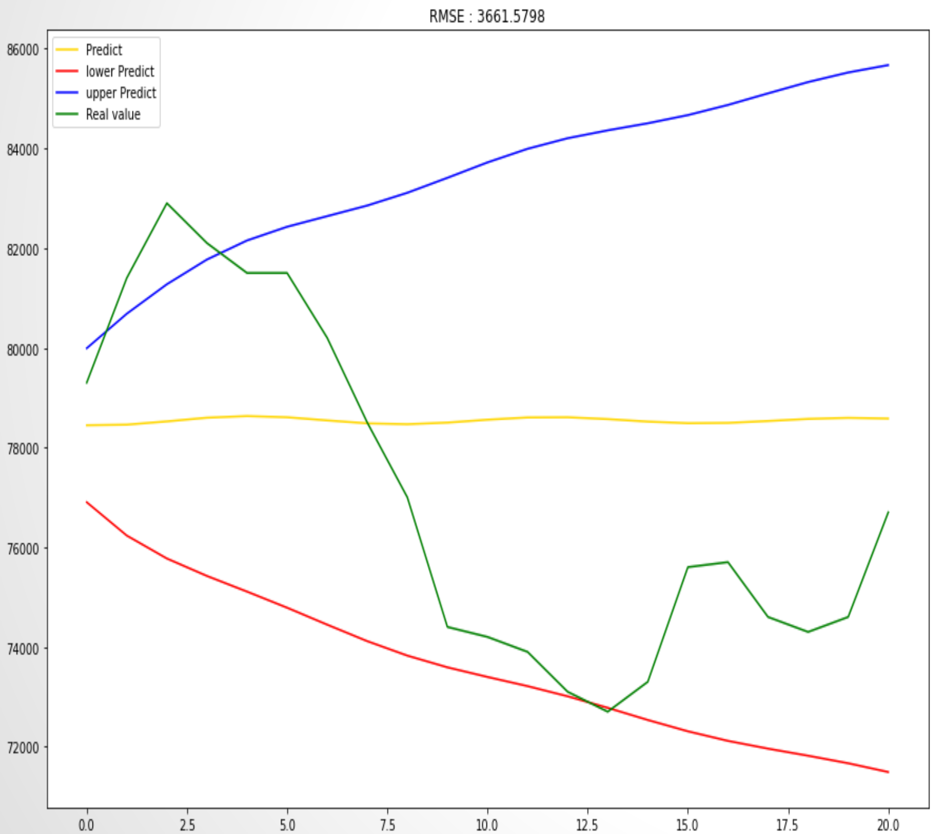
3. ARIMA

[ARIMA Model 결과 Plot]

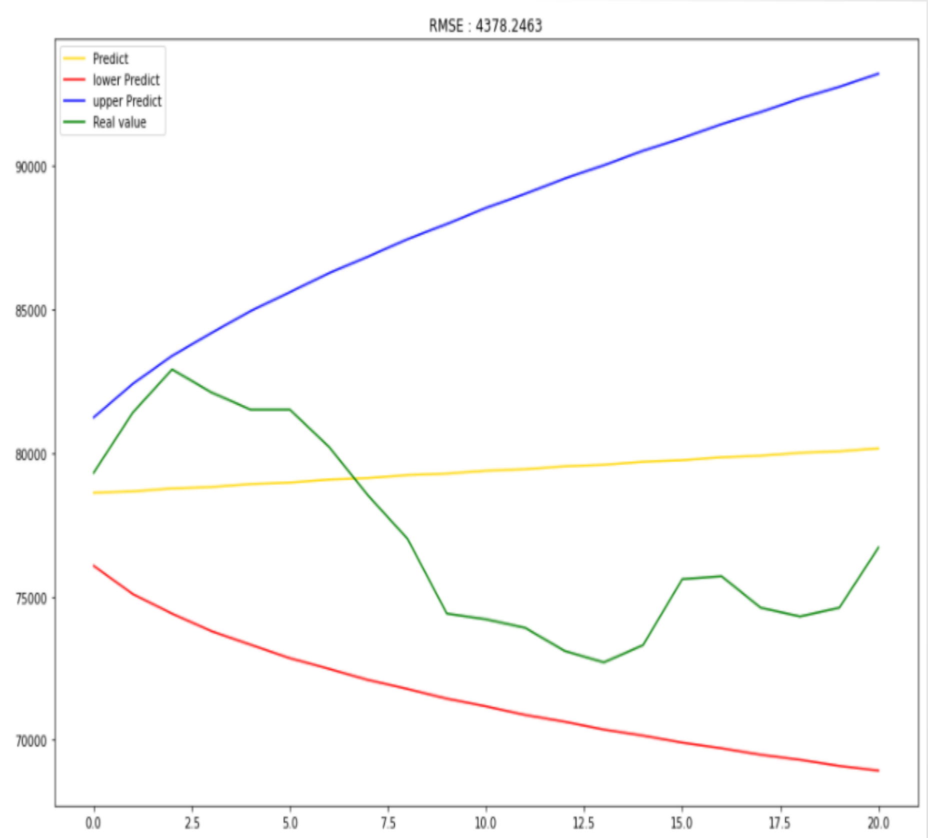


3. ARIMA

[ARIMA Model 결과 Plot]



[1Month 예측: ARIMA Model (Data 로그, 차분)]



4. 시사점 및 결론

SAMSUNG

PROPHET

ARIMA

Thank You