Prophet 시계열 분석 예제 - 삼성전자 주가 예측

`yfinance` 패키지 설치가 필요하다

#01. 패키지 참조

```
import yfinance as yf
from pandas import DataFrame
from matplotlib import pyplot as plt
from prophet import Prophet
from prophet.plot import add_changepoints_to_plot
import seaborn as sb
import datetime as dt
# 평균 절대 오차 함수
from sklearn.metrics import mean_absolute_error
```

```
plt.rcParams['font.family'] = 'Malgun Gothic'
plt.rcParams['axes.unicode_minus'] = False
```

#02. 데이터 불러오기

```
#origin = yf.download('005930.KS', start='2020-01-01', end='2020-12-31')
origin = yf.download('005930.KS', start='2020-01-01')
origin
```

```
[**********100%%**********]  1 of 1 completed
```

	Open	High	Low	Close	Adj Close	Volume
Date						
2020-01-02	55500.0	56000.0	55000.0	55200.0	50037.410156	12993228
2020-01-03	56000.0	56600.0	54900.0	55500.0	50309.351562	15422255
2020-01-06	54900.0	55600.0	54600.0	55500.0	50309.351562	10278951
2020-01-07	55700.0	56400.0	55600.0	55800.0	50581.292969	10009778
2020-01-08	56200.0	57400.0	55900.0	56800.0	51487.769531	23501171
...	...	...	...	...	...	...
2023-10-13	68000.0	68500.0	67700.0	68000.0	68000.000000	9724086
2023-10-16	67900.0	68500.0	66800.0	67300.0	67300.000000	12599299
2023-10-17	67700.0	69900.0	67400.0	69400.0	69400.000000	17299253
2023-10-18	68900.0	70500.0	68800.0	70500.0	70500.000000	16493184

	Open	High	Low	Close	Adj Close	Volume
Date						
2023-10-19	69700.0	70000.0	69400.0	69600.0	69600.000000	9089427

936 rows × 6 columns

#03. 데이터 전처리

필요한 필드만 추출하기

```
target_df = origin[['Close']]
target_df.head()
```

	Close
Date	
2020-01-02	55200.0
2020-01-03	55500.0
2020-01-06	55500.0
2020-01-07	55800.0
2020-01-08	56800.0

Prophet 라이브러리 형식에 맞추기

날짜 인덱스를 일반 컬럼으로 변환

```
df = target_df.reset_index()
df.head()
```

	Date	Close
0	2020-01-02	55200.0
1	2020-01-03	55500.0
2	2020-01-06	55500.0
3	2020-01-07	55800.0
4	2020-01-08	56800.0

필드 이름 변경

```
df.rename(columns={'Date':'ds', 'Close':'y'}, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 936 entries, 0 to 935
Data columns (total 2 columns):
```

```
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ds      936 non-null    datetime64[ns]
 1   y       936 non-null    float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 14.8 KB
```

#04. 데이터 시각화

최고가, 최저가 확인

```
df['y'].max(), df['y'].min()
```

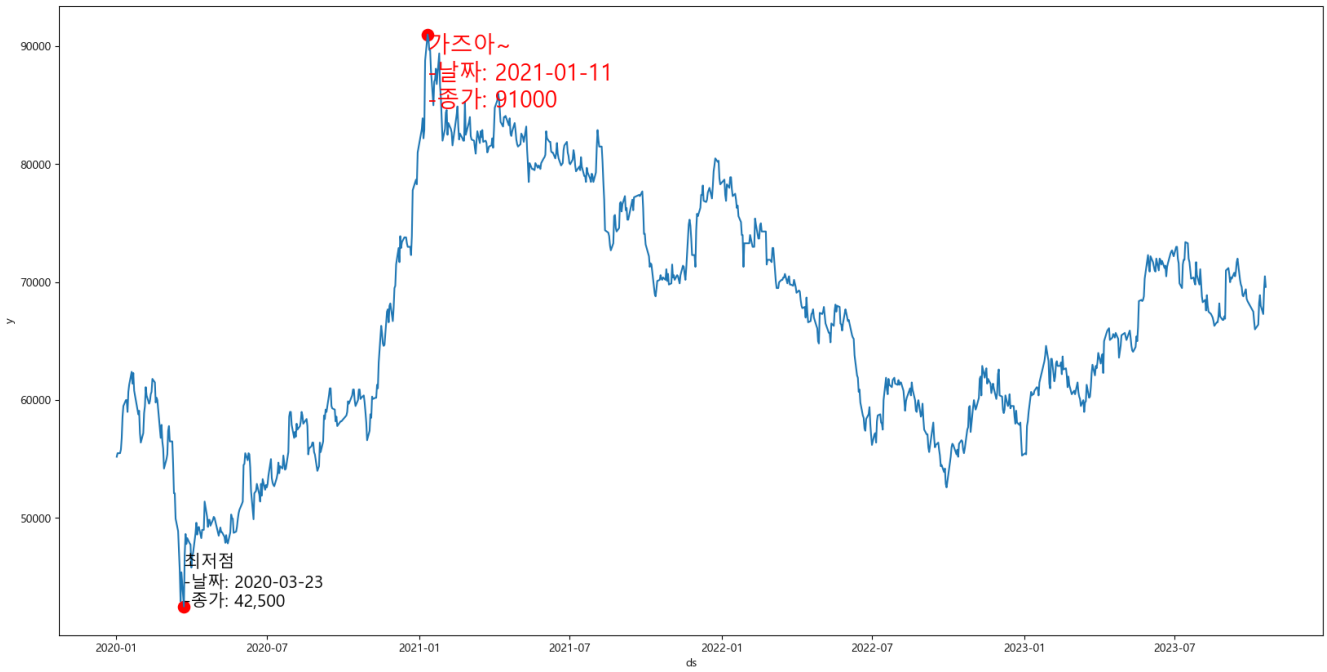
```
(91000.0, 42500.0)
```

```
minmax = df.query("y == @df['y'].max() | y == @df['y'].min()")
minmax
```

	ds	y
55	2020-03-23	42500.0
253	2021-01-11	91000.0

라인 그래프

```
min_date = dt.datetime(2020, 3, 23)
max_date = dt.datetime(2021, 1, 11)

plt.figure(figsize=(20, 10))
sb.lineplot(data=df, x='ds', y='y')
sb.scatterplot(data=minmax, x='ds', y='y', color='red', s=150, marker='o')

plt.text(min_date, 42500, '최저점 \n-날짜: 2020-03-23 \n-종가: 42,500',
         fontsize=15)

plt.text(max_date, 91000, '가즈아~ \n-날짜: 2021-01-11 \n-종가: 91000',
         fontsize=20, color='red', verticalalignment='top',
         horizontalalignment='left')

plt.show()
plt.close()
```

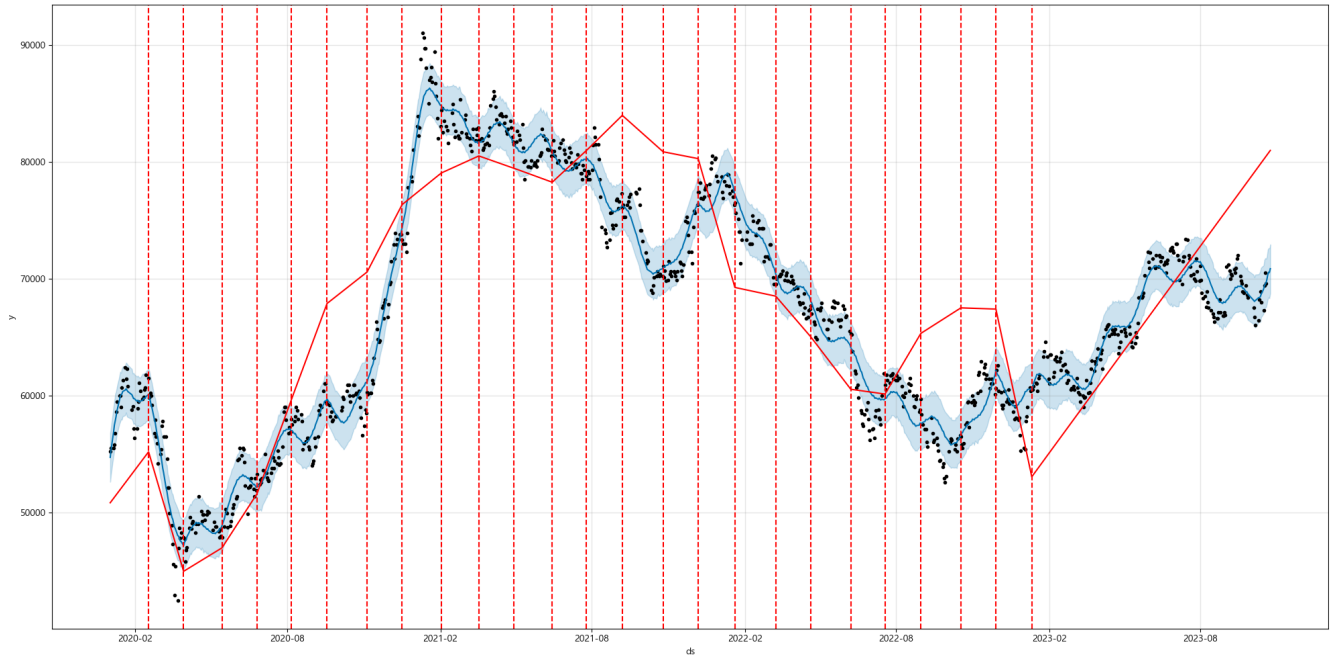


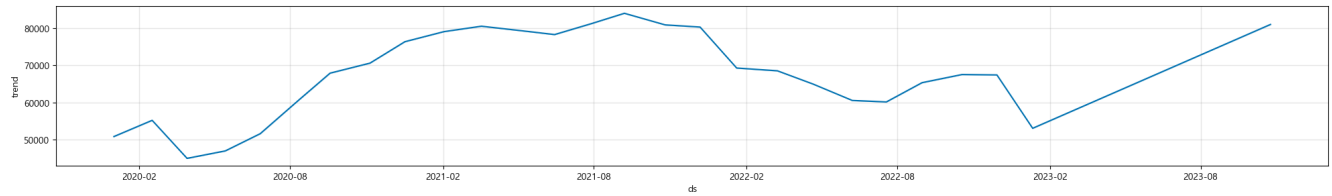
#05. 시계열 분석 수행

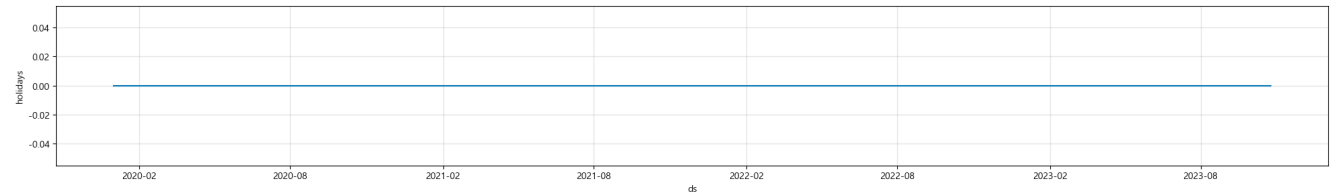
학습 모델 구축

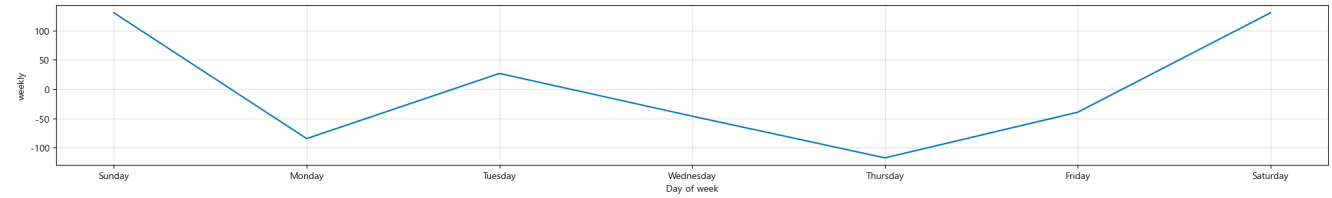
```
m = Prophet(
    # Trend
    changepoint_prior_scale=0.5,
    # Seasonality
    weekly_seasonality=True,
    yearly_seasonality=True,
    daily_seasonality=True
)

# 공휴일 데이터 추가
m.add_country_holidays(country_name='KR')

m.fit(df)

# 5일간의 데이터를 예측
future = m.make_future_dataframe(periods=5, freq='D')
forecast = m.predict(future)

# 시각화
fig1 = m.plot(forecast, figsize=(20, 10))
add_changepoints_to_plot(fig1.gca(), m, forecast)

fig2 = m.plot_components(forecast, figsize=(20, 15))
```

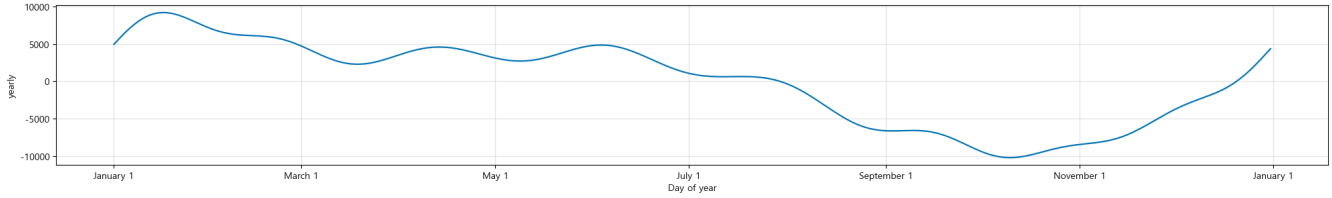
```
14:05:11 - cmdstanpy - INFO - Chain [1] start processing
14:05:11 - cmdstanpy - INFO - Chain [1] done processing
```

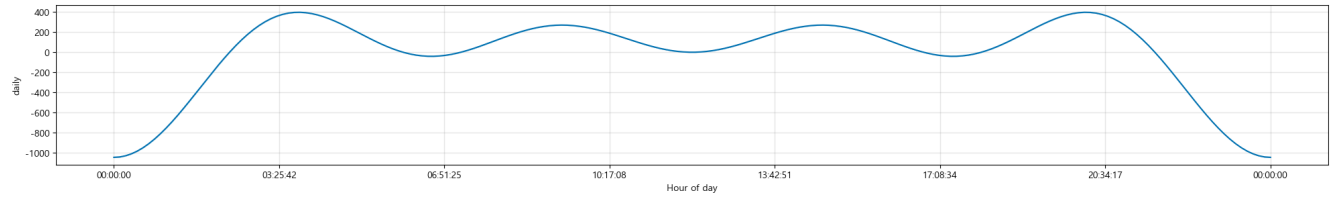












#06. 결과 평가

학습데이터에 대한 예측 결과 추출

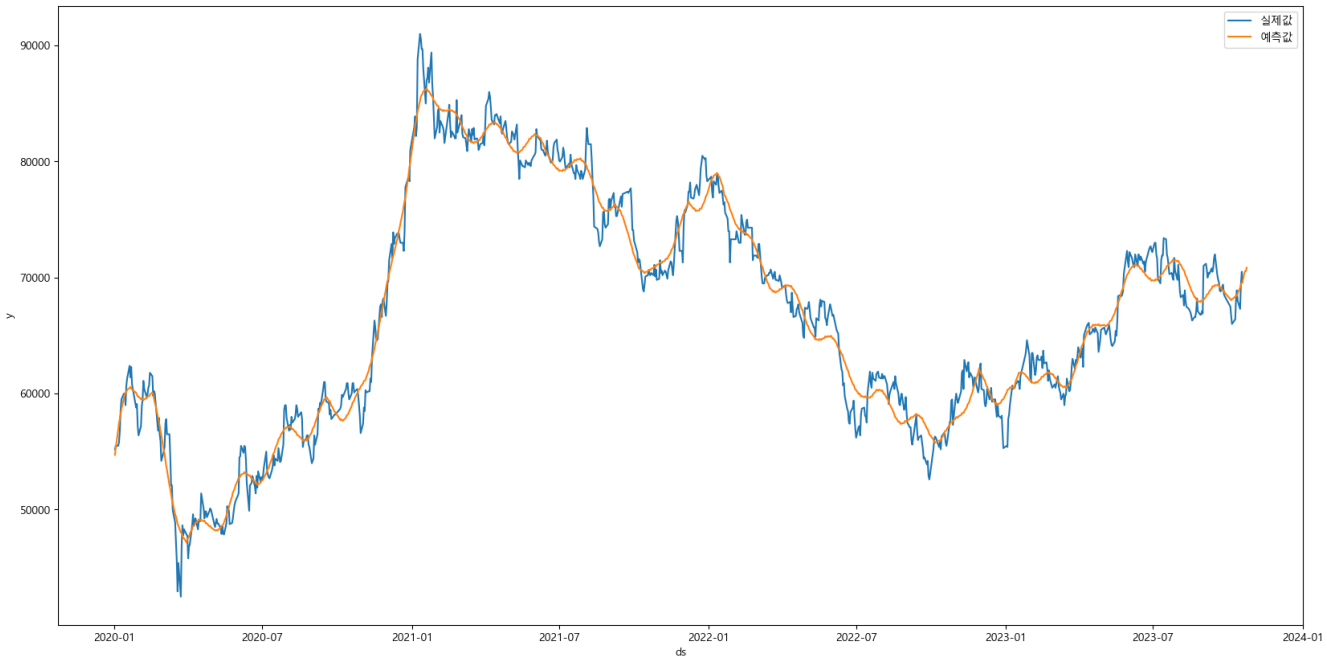
```
pred = forecast[['ds','yhat']]
pred
```

	ds	yhat
0	2020-01-02	54715.876695
1	2020-01-03	55347.480511
2	2020-01-06	56876.634125
3	2020-01-07	57470.603909
4	2020-01-08	57852.400796
...	...	...
936	2023-10-20	69955.103754
937	2023-10-21	70335.768803
938	2023-10-22	70545.896352
939	2023-10-23	70538.424341
940	2023-10-24	70853.956629

941 rows × 2 columns

실 데이터와 예측치 비교

```
plt.figure(figsize=(20, 10))
sb.lineplot(data=df, x='ds', y='y', label='실제값')
sb.lineplot(data=pred, x='ds', y='yhat', label='예측값')
plt.legend()
plt.show()
plt.close()
```



MAE (절대 평균 오차)

학습 모델의 하이퍼 파라미터를 조절하여 이 값이 가장 적게 나타나는 학습 모델을 찾는다.

최근 1년치 실제 데이터 추출

```
last_year = df.query('ds > "2022-10-19"')
last_year
```

	ds	y
689	2022-10-20	55500.0
690	2022-10-21	55900.0
691	2022-10-24	57500.0
692	2022-10-25	57700.0
693	2022-10-26	59400.0
...	...	...
931	2023-10-13	68000.0
932	2023-10-16	67300.0
933	2023-10-17	69400.0
934	2023-10-18	70500.0
935	2023-10-19	69600.0

247 rows × 2 columns

최근 1년치에 해당하는 예측값을 생성

```
y_pred = m.predict(last_year)
y_pred['yhat']
```

```
0      56826.614248
1      57015.317513
2      57289.445792
3      57498.145856
4      57515.590294
           ...
242    68654.219477
243    69107.025043
244    69408.908071
245    69534.545065
246    69668.103480
Name: yhat, Length: 247, dtype: float64
```

MAE값 계산

```
mae = mean_absolute_error(last_year['y'].values, y_pred['yhat'].values)
print('MAE: %.3f' % mae)
```

```
MAE: 1256.705
```