Merchine Learning

KOSPI와 외국 지수의 상관관계

Team 3

남정우, 전현준, 정제경, 전한성

Contents

데이터 전처리
(가져오기, 병합)

데이터 분석
및 시각화

데이터 선택
및 훈련

실제 예측

데이터 가져오기 및 전처리

데이터 가져오기 및 전처리

```
kospi = fdr.DataReader('KS11', '2022-01-01', '2022-12-31')
kospi.drop(['High', 'Low', 'Open', 'Volume', 'Adj Close'], axis = 1, inplace = True)
kospi.rename(columns = {'Close':'KOSPI'},inplace=True)
kospi
```

Date	KOSPI
2022-01-04	2989.239990
2022-01-05	2953.969971
2022-01-06	2920.530029
2022-01-07	2954.889893
2022-01-10	2926.719971
...	...
2022-12-05	2419.320068
2022-12-06	2393.159912
2022-12-07	2382.810059
2022-12-08	2371.080078
2022-12-09	2388.600098

230 rows × 1 columns

데이터 병합

```
df = pd.concat([kospi, sp500, nasdaq, japan, china, usd_krw], axis=1)
df = df.dropna()
df = round(df, 2)
df
```

Date	KOSPI	S&P	NASDAQ	Nikkei	SSEC	USD/KRW
2022-01-04	2989.24	4793.54	15622.72	29301.79	3632.33	1194.68
2022-01-05	2953.97	4700.58	15100.17	29332.16	3595.18	1196.50
2022-01-06	2920.53	4696.05	15080.86	28487.87	3586.08	1199.25
2022-01-07	2954.89	4677.03	14935.90	28478.56	3579.54	1205.78
2022-01-11	2927.38	4713.07	15153.45	28222.48	3567.44	1198.09
...	...	...	...	...	...	...
2022-12-02	2434.33	4071.70	11461.50	27777.90	3156.14	1303.25
2022-12-05	2419.32	3998.84	11239.94	27820.40	3211.81	1299.17
2022-12-06	2393.16	3941.26	11014.89	27885.87	3212.53	1304.37
2022-12-07	2382.81	3933.92	10958.55	27686.40	3199.62	1319.71
2022-12-08	2371.08	3963.51	11082.00	27574.43	3197.35	1314.10

198 rows × 6 columns

데이터 분석 및 시각화

데이터 분석 및 시각화

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 198 entries, 2022-01-04 to 2022-12-08
Data columns (total 6 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   KOSPI    198 non-null    float64
 1   S&P      198 non-null    float64
 2   NASDAQ   198 non-null    float64
 3   Nikkei   198 non-null    float64
 4   SSEC     198 non-null    float64
 5   USD/KRW  198 non-null    float64
dtypes: float64(6)
memory usage: 10.8 KB
```

```
df.describe()
```

	KOSPI	S&P	NASDAQ	Nikkei	SSEC	USD/KRW
count	198.000000	198.000000	198.000000	198.000000	198.000000	198.000000
mean	2528.177879	4102.650051	12273.970556	27280.171667	3231.781313	1292.443737
std	190.746290	289.616472	1287.963237	851.180460	160.514516	73.543591
min	2155.490000	3577.030000	10321.390000	24790.950000	2886.430000	1185.880000
25%	2393.145000	3900.280000	11204.387500	26645.765000	3098.352500	1225.307500
50%	2482.095000	4028.365000	11872.635000	27282.160000	3235.905000	1292.205000
75%	2691.300000	4354.820000	13379.032500	27910.937500	3305.237500	1338.380000
max	2989.240000	4793.540000	15622.720000	29332.160000	3632.330000	1443.960000

데이터 분석 및 시각화

상관관계

: **KOSPI** 와 **S&P**, 나스닥은
높은 상관관계를 보이는 반면,

환율과 **KOSPI**, **S&P**, **NASDAQ** 등은
매우 낮은('역의') 상관관계를 보임

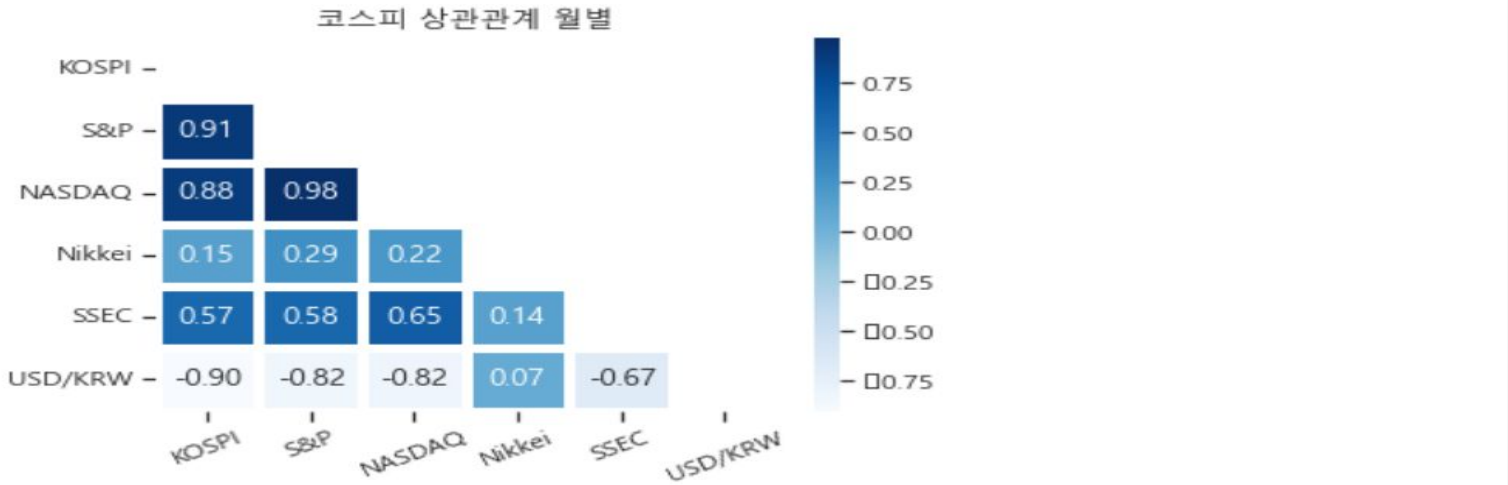
상관관계

```
corr_matrix = df.corr()
corr_matrix
```

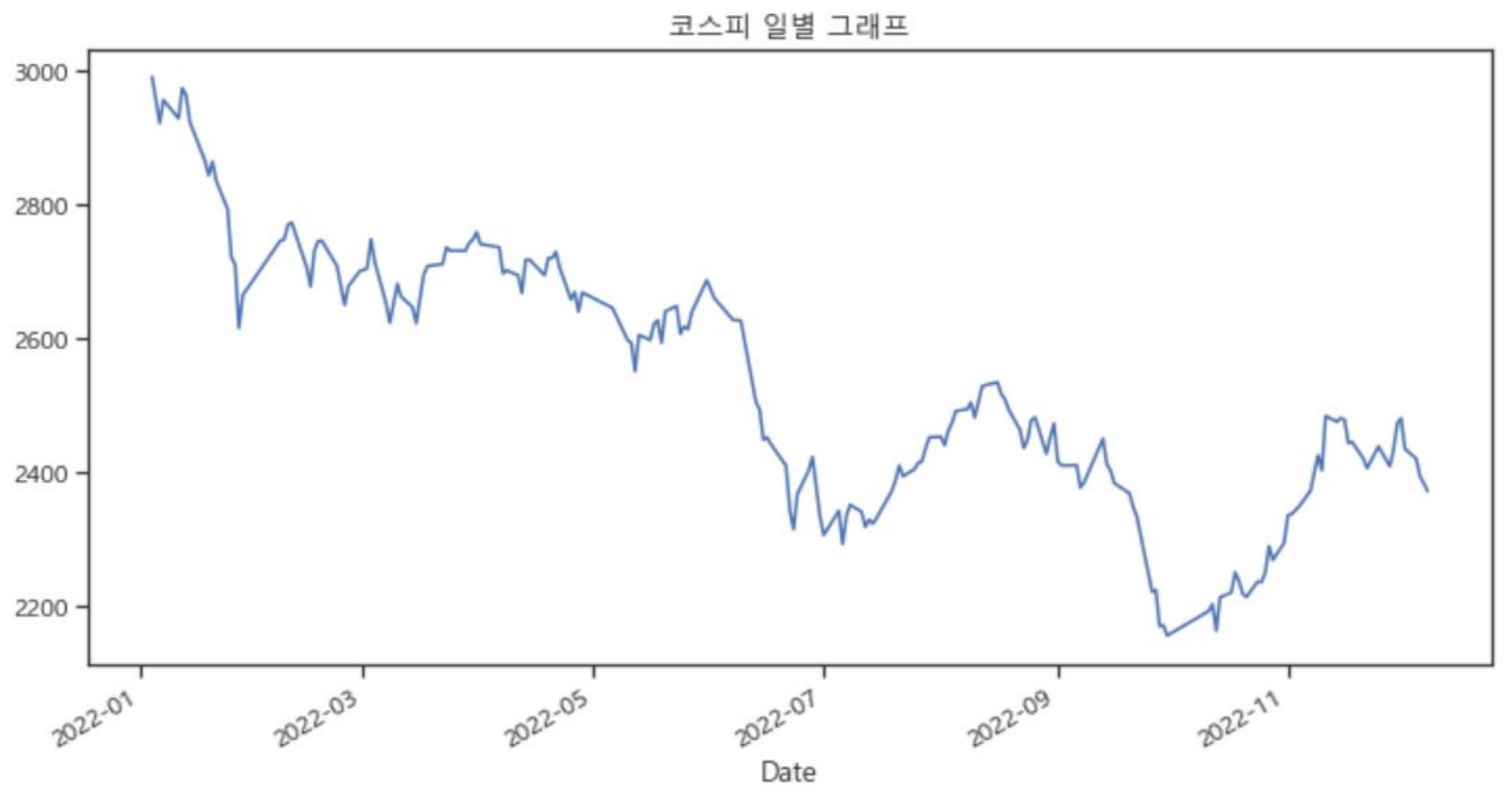
	KOSPI	S&P	NASDAQ	Nikkei	SSEC	USD/KRW
KOSPI	1.000000	0.909017	0.883460	0.154667	0.570777	-0.901626
S&P	0.909017	1.000000	0.977141	0.286885	0.576151	-0.822469
NASDAQ	0.883460	0.977141	1.000000	0.222767	0.654038	-0.817404
Nikkei	0.154667	0.286885	0.222767	1.000000	0.144749	0.065965
SSEC	0.570777	0.576151	0.654038	0.144749	1.000000	-0.672342
USD/KRW	-0.901626	-0.822469	-0.817404	0.065965	-0.672342	1.000000

상관관계 시각화자료

```
mask = np.zeros_like(corr_matrix, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
sns.heatmap(corr_matrix, mask = mask, annot=True, linewidths=5, fmt='.2f', cmap='Blues')
plt.xticks(rotation=25)
plt.title('코스피 상관관계 월별')
plt.show()
```

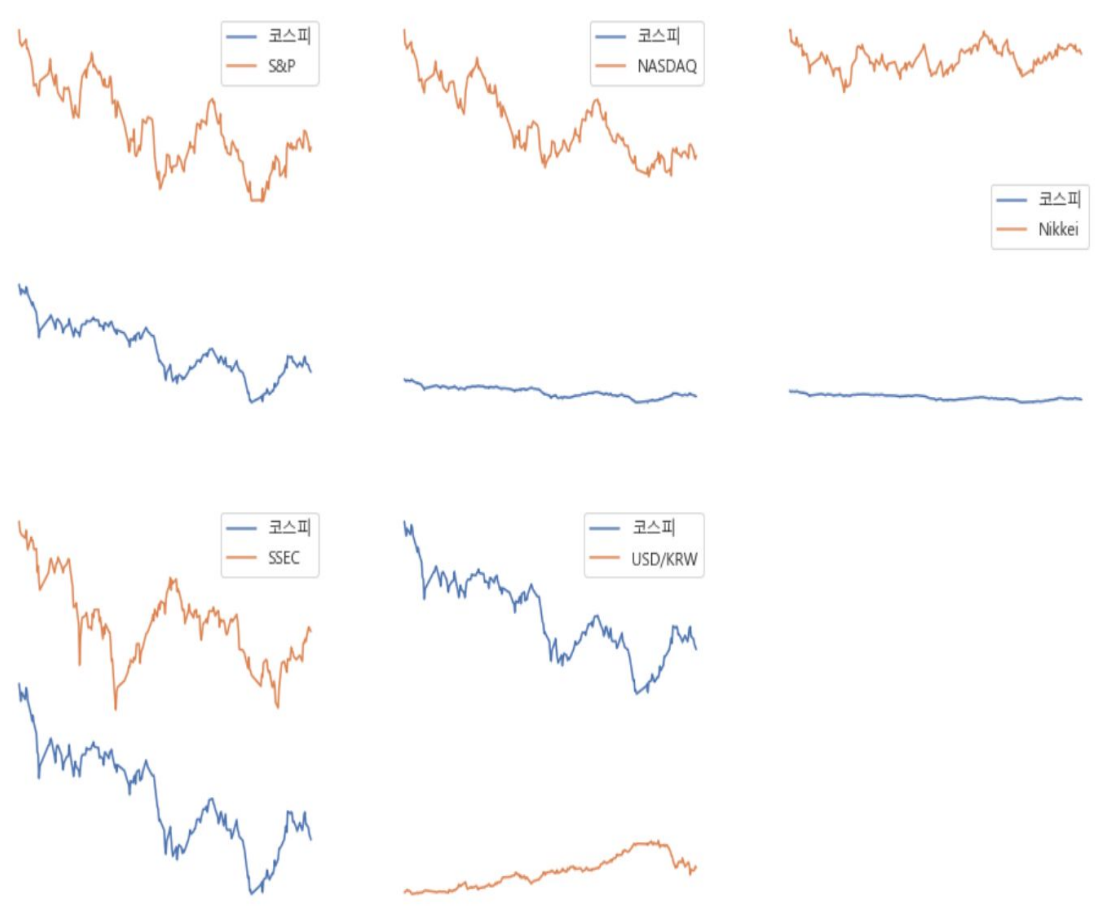


2022년 코스피 일별 코스피

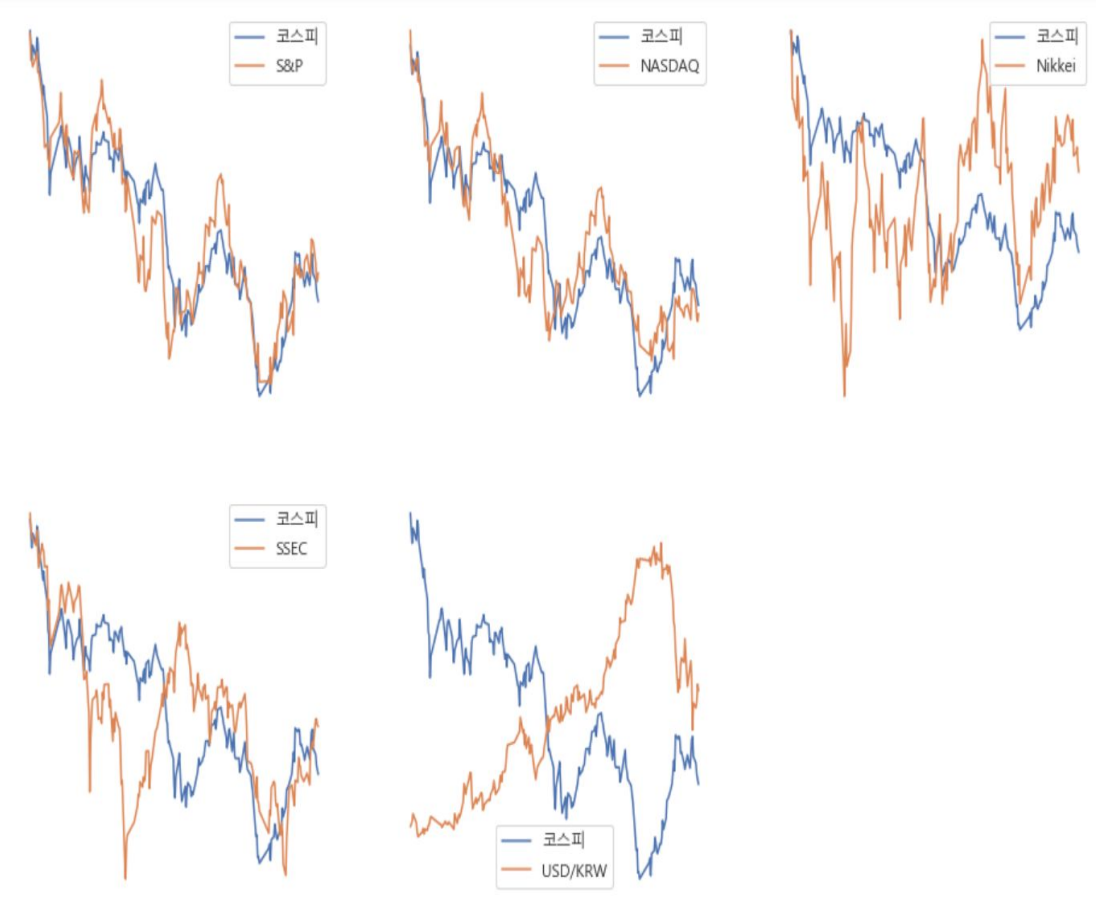


코스피와 외국 지수 1:1 매칭 그래프

Before



After



코스피와 외국 지수 1:1 매칭 그래프

Before

```
a = df.columns[1:]
plt.figure(figsize=(16,16))
for idx, col in enumerate(a) :
    plt.subplot(3,3,1+int(idx))
    sns.lineplot(data=df, x=df.index, y='KOSPI',label='코스피')
    sns.lineplot(data=df, x=df.index, y=col,label=col)
    plt.axis('off')
plt.show()
```

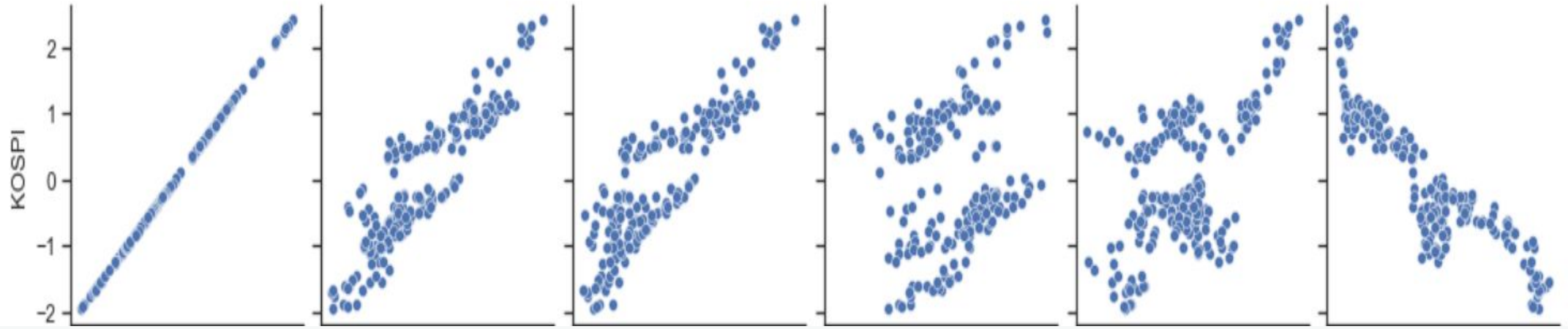
After

```
a = df.columns[1:]
plt.figure(figsize=(16,16))
for idx, col in enumerate(a) :
    plt.subplot(3,3,1+int(idx))
    sns.lineplot(data=df_s, x=df_s.index, y='KOSPI',label='코스피')
    sns.lineplot(data=df_s, x=df_s.index, y=col,label=col)
    plt.axis('off')
plt.show()
```

산점도

산점도

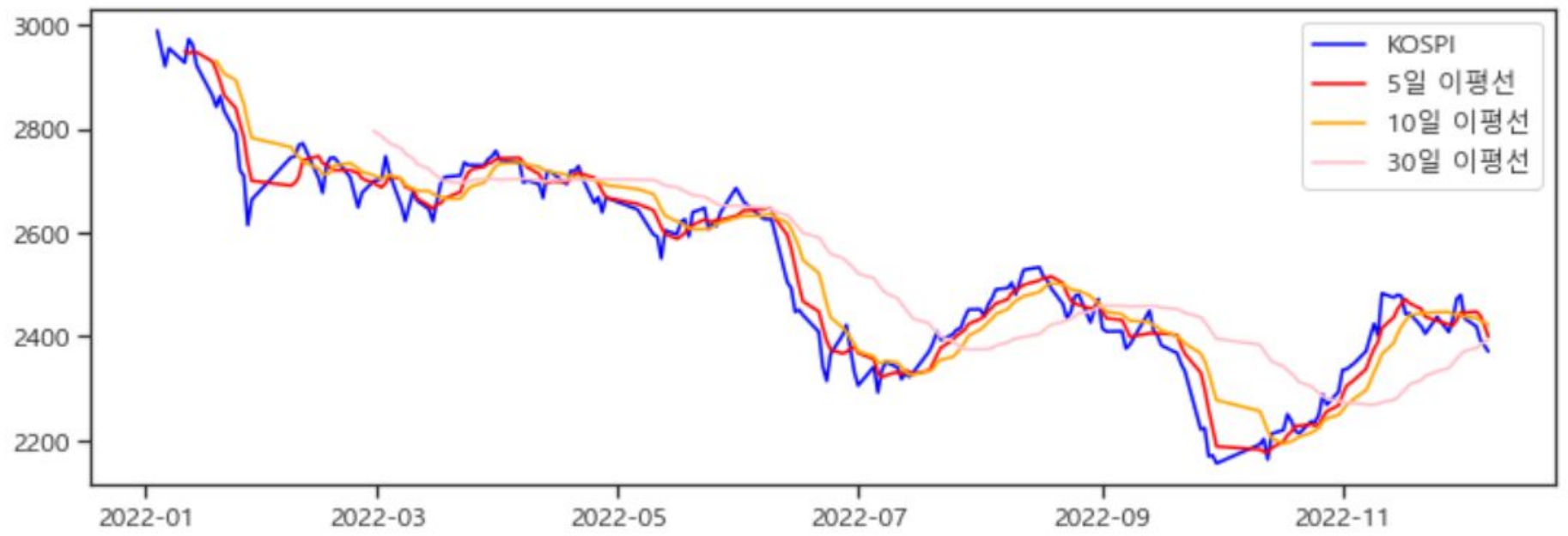
```
sns.set(font_scale=1.1) ## 폰트사이즈 조절
sns.set_style('ticks') ## 축 눈금 표시
# data = df_s[["KOSPI", "S&P","NASDAQ","Nikkei","SSEC","USD/KRW"]]
sns.pairplot(df_s[["KOSPI", "S&P","NASDAQ","Nikkei","SSEC","USD/KRW"]],diag_kind=None)
plt.show()
```



코스피 이동 평균선 그래프

```
fig = plt.figure(figsize = (12, 4))
chart = fig.add_subplot(1,1,1)

chart.plot(df['KOSPI'], color='blue' , label='KOSPI')
chart.plot(roll_mean5, color='red' , label='5일 이평선')
chart.plot(roll_mean10, color='orange' , label='10일 이평선')
chart.plot(roll_mean30, color='pink' , label='30일 이평선')
plt.legend(loc = 'best')
```

```
<matplotlib.legend.Legend at 0x16ee3dea940>
```



데이터 선택 및 훈련

데이터 분리

```
from sklearn.model_selection import train_test_split
X = df[['S&P', 'NASDAQ','Nikkei','SSEC','USD/KRW']]
y = df['KOSPI']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, shuffle=False)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(178, 5) (20, 5) (178,) (20,)
```

데이터 선택 및 훈련

결정트리 모델

결정트리 모델

```
tree_reg = DecisionTreeRegressor()
tree_reg.fit(X_train, y_train)
y_pred = tree_reg.predict(X_test)
```

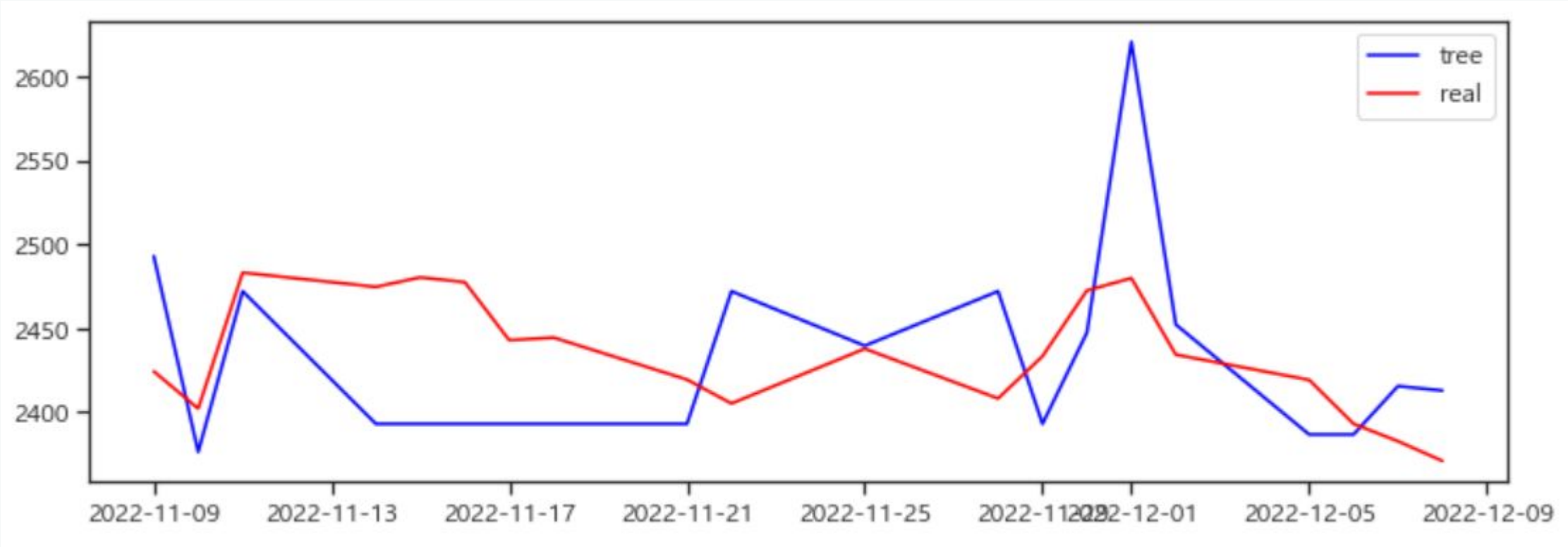
```
result = pd.DataFrame({'pred_tree' : y_pred, 'real' : y_test})
result['ratio(%)'] = abs((result['pred_tree'] - result['real']) / result['real']) * 100
result.head(-1)
```

```
from sklearn.metrics import mean_absolute_error
score_mae_tree = mean_absolute_error(y_test, y_pred)

from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test, y_pred)
score_rmse_tree = np.sqrt(MSE)

print('tree MAE score:', score_mae_tree)
print('tree RMSE score:', score_rmse_tree)
```

```
tree MAE score: 47.777000000000065
tree RMSE score: 58.069723092158855
```

Date	pred_tree	real	ratio(%)
2022-11-09	2406.906848	2424.41	0.721955
2022-11-10	2388.315948	2402.23	0.579214
2022-11-11	2436.687856	2483.16	1.871492
2022-11-14	2416.743613	2474.65	2.339983
2022-11-15	2412.962271	2480.33	2.716079
2022-11-16	2422.229159	2477.45	2.228939
2022-11-17	2410.581758	2442.90	1.322946
2022-11-18	2413.062720	2444.48	1.285234
2022-11-21	2417.224792	2419.50	0.094036
2022-11-22	2433.554940	2405.27	1.175957
2022-11-25	2471.279505	2437.86	1.370854
2022-11-28	2433.440868	2408.27	1.045185
2022-11-29	2422.710337	2433.39	0.438880
2022-11-30	2453.320747	2472.53	0.776907
2022-12-01	2571.576558	2479.84	3.699293
2022-12-02	2465.422747	2434.33	1.277261
2022-12-05	2418.427162	2419.32	0.036905
2022-12-06	2405.043882	2393.16	0.496577
2022-12-07	2412.774986	2382.81	1.257548

결정트리 모델 예측 값 비교 그래프



리니어 모델

리니어 모델

```
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred = lin_reg.predict(X_test)
y_pred
```

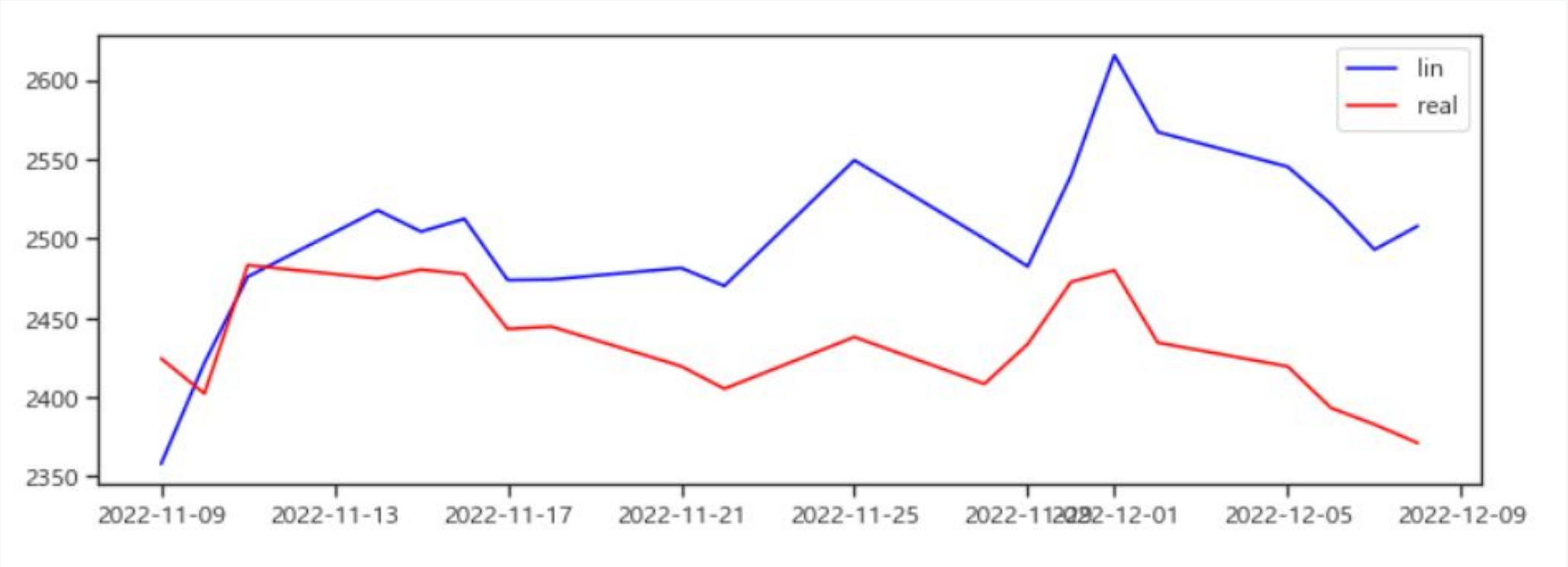
```
array([2357.83558715, 2421.51758542, 2475.83583948, 2517.7413115 ,
       2504.25143368, 2512.30288287, 2473.73442576, 2474.06447901,
       2481.38690363, 2470.0456278 , 2549.24771724, 2499.74218139,
       2482.2837619 , 2539.64888403, 2615.3840204 , 2567.04793705,
       2545.13375227, 2521.52482251, 2492.91328505, 2507.81605705])
```

```
result = pd.DataFrame({'pred' : y_pred, 'real' : y_test})
result['ratio(%)'] = abs((result['pred'] - result['real']) / result['real']) * 100
result.head(-1)
```

```
from sklearn.metrics import mean_absolute_error
score_mae_lin = mean_absolute_error(y_test, y_pred)

from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test, y_pred)
score_rmse_lin = np.sqrt(MSE)

print('linear MAE score:', score_mae_lin)
print('linear RMSE score:', score_rmse_lin)
```

```
linear MAE score: 73.51428209659082
linear RMSE score: 85.18678630977912
```

리니어 모델 예측 값 비교 그래프



랜덤포레스트 모델

랜덤포레스트 모델

```
rf_reg = RandomForestRegressor()
rf_reg.fit(X_train, y_train)
y_pred = rf_reg.predict(X_test)
y_pred
```

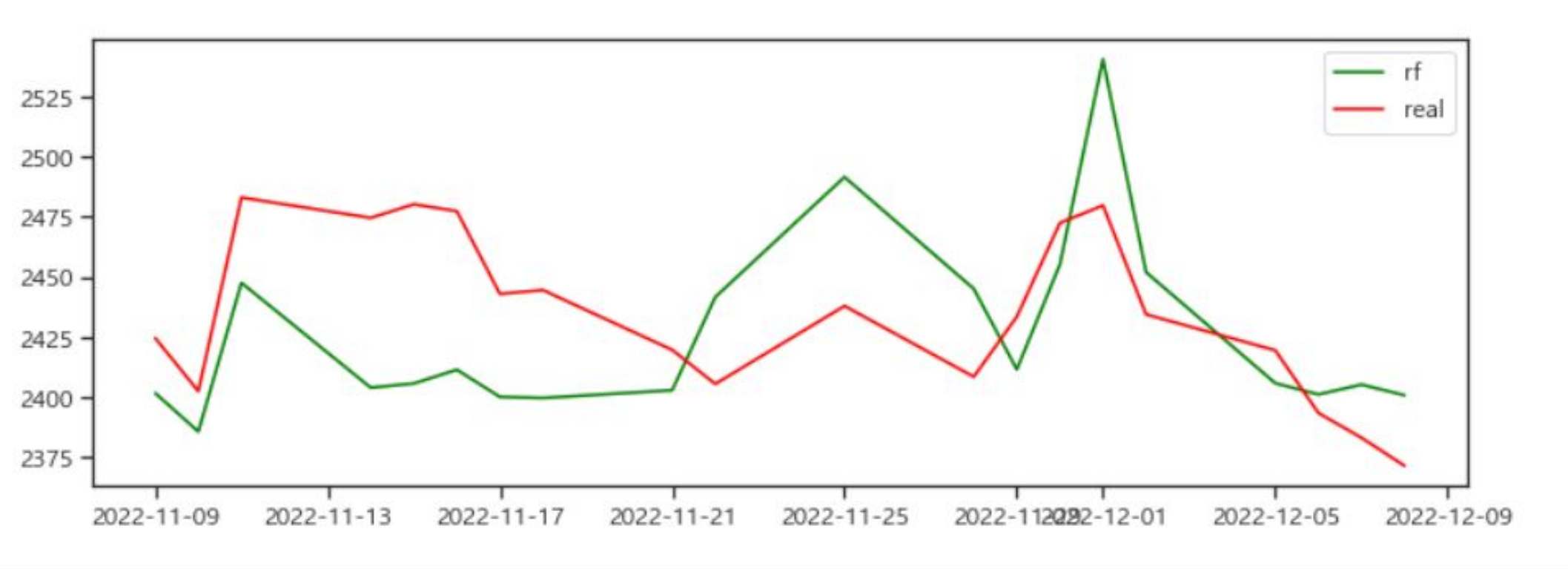
```
array([2401.4301, 2385.3466, 2447.5254, 2403.7335, 2405.4634, 2411.2162,
       2399.8631, 2399.409 , 2402.6561, 2441.5844, 2491.713 , 2445.027 ,
       2411.3509, 2455.2232, 2540.9559, 2451.9518, 2405.5826, 2400.8974,
       2404.9943, 2400.5629])
```

```
result = pd.DataFrame({'pred' : y_pred, 'real' : y_test})
result['ratio(%)'] = abs((result['pred'] - result['real']) / result['real']) * 100
result
```

```
from sklearn.metrics import mean_absolute_error
score_mae_rf = mean_absolute_error(y_test, y_pred)

from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test, y_pred)
score_rmse_rf = np.sqrt(MSE)

print('rf MAE score:', mean_absolute_error(y_test, y_pred))
print('rf RMSE score:', np.sqrt(MSE))
```

```
rf MAE score: 35.530829999999675
rf RMSE score: 40.793483071245284
```

랜덤포레스트 모델 예측 값 비교 그래프



그라디언트 부스트 모델

그라디언트 부스트 모델

```
from sklearn.metrics import make_scorer
# log 값 변환 시 언더플로우 영향으로 log() 가 아닌 log1p() 를 이용하여 RMSLE 계산
def rmsle(y, pred, convertExp=True):
    if convertExp:
        y = np.expm1(y)
        pred = np.expm1(pred)

    log_y = np.log1p(y)
    log_pred = np.log1p(pred)
    squared_error = (log_y - log_pred) ** 2
    rmsle = np.sqrt(np.mean(squared_error))
    return rmsle

rmsle_scorer = make_scorer(rmsle, greater_is_better=False)
```

```
gbrt_params = { 'learning_rate' : [0.01, 0.02, 0.03, 0.04], # 각 트리의 기여도
                'n_estimators' : [1000, 1500],
                'subsample' : [0.9, 0.5, 0.2],
                'max_depth' : [2, 4, 6, 8]

}

gbrt = GradientBoostingRegressor()

gridsearch_gbrt = GridSearchCV(gbrt, gbrt_params, scoring=rmsle_scorer, cv=5, n_jobs=-1)
```

```
%time gridsearch_gbrt.fit(X_train, y_train)
```

```
CPU times: total: 2.22 s
Wall time: 41.8 s

GridSearchCV(cv=5, estimator=GradientBoostingRegressor(), n_jobs=-1,
             param_grid={'learning_rate': [0.01, 0.02, 0.03, 0.04],
                         'max_depth': [2, 4, 6, 8],
                         'n_estimators': [1000, 1500],
                         'subsample': [0.9, 0.5, 0.2]},
             scoring=make_scorer(rmsle, greater_is_better=False))
```

그라디언트 부스트 모델

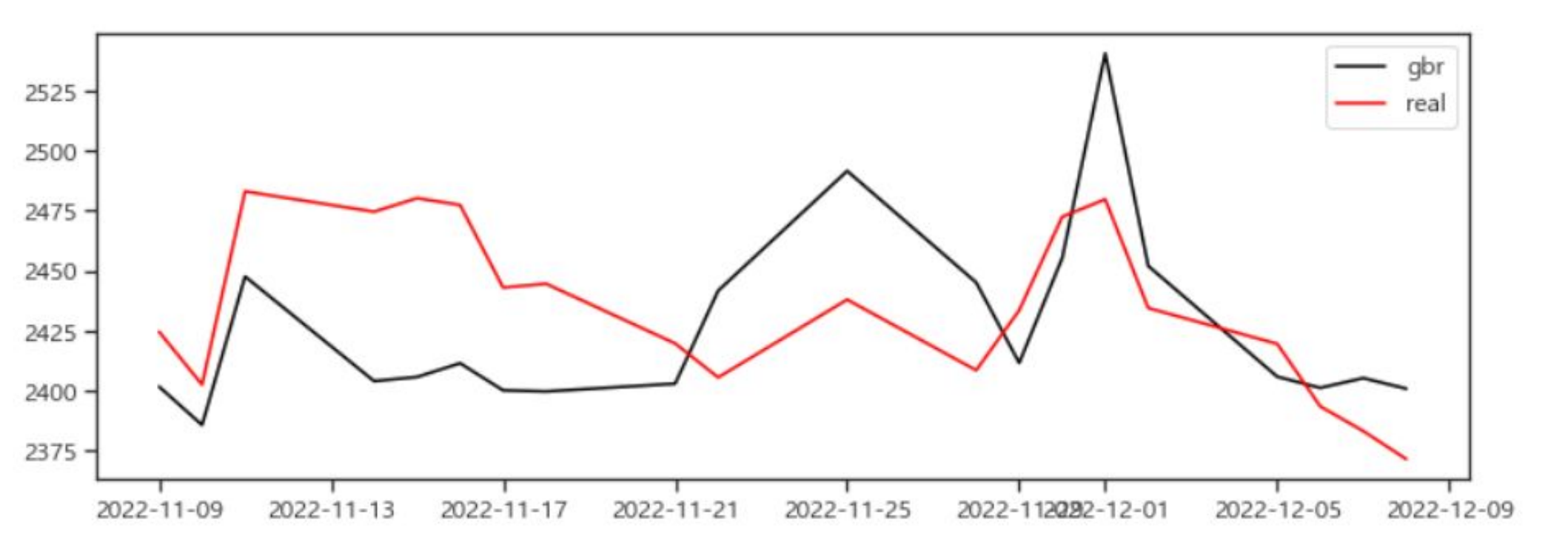
```
best_model = gridsearch_gbrt.best_estimator_

y_pred = best_model.predict(X_test)
y_pred
```

```
result = pd.DataFrame({'pred' : y_pred, 'real' : y_test})
result['ratio(%)'] = abs((result['pred'] - result['real']) / result['real']) * 100
result
```

```
from sklearn.metrics import mean_absolute_error
score_mae_gbr = mean_absolute_error(y_test, y_pred)

from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test, y_pred)
score_rmse_gbr = np.sqrt(MSE)

print('rf MAE score:', mean_absolute_error(y_test, y_pred))
print('rf RMSE score:', np.sqrt(MSE))
```

```
rf MAE score: 32.213211899832935
rf RMSE score: 39.05241085935489
```

그라디언트 부스트 모델 예측 값 비교 그래프



아리마 모델

ARIMA 모델

```
# 아리마 사용하여 코스피 예측
from statsmodels.tsa.arima_model import ARIMA
import statsmodels.api as sm
import statsmodels.tsa.arima_model as smt
import statsmodels
from itertools import product
```

```
arima_train = df[:'2022-12-02']
arima_test = df['2022-12-03':]
arima_train, arima_test
```

```
p = range(0,3)
d = range(1,2)
q = range(0,3)
pdq = list(product(p,d,q))  #조합 만들기

#최적의 파라미터 찾기
#(2,1,2)
aic =[]
for i in pdq:
  model_fit = sm.tsa.arima.ARIMA(df.KOSPI.values, order=(i)).fit()
  #model_fit = model.fit()
  print(f'ARIMA:{i} >> AIC : {round(model_fit.aic,2)}')
  aic.append(round(model_fit.aic,2))
```

```
ARIMA:(0, 1, 0) >> AIC : 1929.21
ARIMA:(0, 1, 1) >> AIC : 1930.84
ARIMA:(0, 1, 2) >> AIC : 1932.77
ARIMA:(1, 1, 0) >> AIC : 1930.86
ARIMA:(1, 1, 1) >> AIC : 1932.36
ARIMA:(1, 1, 2) >> AIC : 1934.77
ARIMA:(2, 1, 0) >> AIC : 1932.78
ARIMA:(2, 1, 1) >> AIC : 1934.77
ARIMA:(2, 1, 2) >> AIC : 1936.12
```

데이터 선택 및 훈련

아리마 모델

```
forecast_data = model_fit.forecast(steps=3)
forecast_data, arima_test
```

```
(194    2428.728103
 195    2428.232196
 196    2423.440444
 Name: predicted_mean, dtype: float64,
               KOSPI      S&P    NASDAQ    Nikkei      SSEC  USD/KRW
 Date
 2022-12-05  2419.32  3998.84  11239.94  27820.40  3211.81  1299.17
 2022-12-06  2393.16  3941.26  11014.89  27885.87  3212.53  1304.37
 2022-12-07  2382.81  3933.92  10958.55  27686.40  3199.62  1319.71
 2022-12-08  2371.08  3963.51  11082.00  27574.43  3197.35  1314.10)
```

```
pred_y = forecast_data.tolist()
pred_y
```

```
[2428.728102894387, 2428.232195925503, 2423.440443743366]
```

```
pred_arima_y = forecast_data.tolist()
test_y = y_test
```

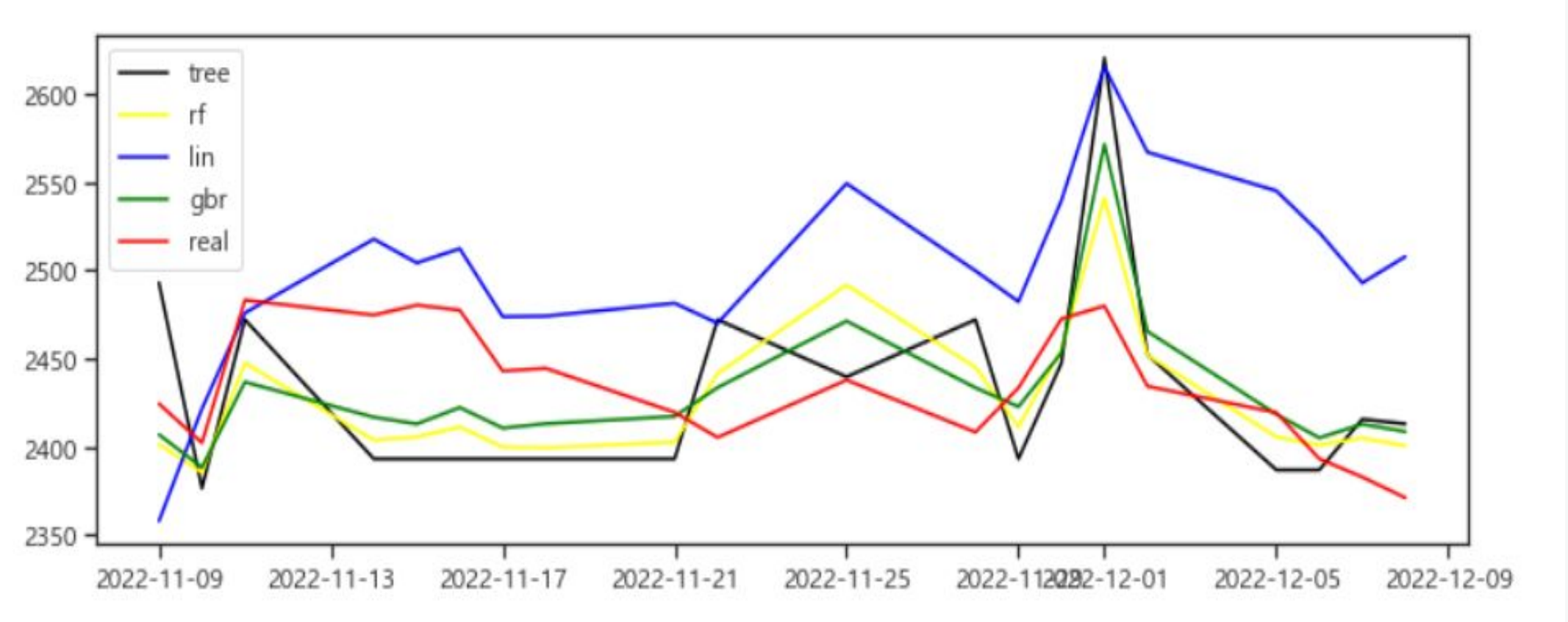
```
y_test
```

```
Date
2022-12-01    2479.84
2022-12-02    2434.33
2022-12-05    2419.32
2022-12-06    2393.16
2022-12-07    2382.81
Name: KOSPI, dtype: float64
```

```
pred_arima_y
```

```
[2381.478082769998, 2376.643785285861, 2374.779967622734]
```

예측 값 비교 그래프(종합)


tree
rf
lin
gbr
real
2600
2550
2500
2450
2400
2350
2022-11-09
2022-11-13
2022-11-17
2022-11-21
2022-11-25
2022-12-01
2022-12-05
2022-12-09

실제 예측

실제 예측

```
# 나중에 실제 예측을 위한 가상 데이터
X_test_pred = pd.DataFrame({'S&P':[3967.92, 3999.36, 4030.01],
                            'NASDAQ':[11008.55, 11250.23, 11400.01],
                            'Nikkei':[27274.40, 27374.40, 27374.40],
                            'SSEC':[3127.62, 3187.62, 3207.62],
                            'USD/KRW':[1316.22, 1306.60, 1301.42]}, index = ['2022-12-09', '2022-12-12', '2022-12-13'])

X_test_pred
```

	S&P	NASDAQ	Nikkei	SSEC	USD/KRW
2022-12-09	3967.92	11008.55	27274.4	3127.62	1316.22
2022-12-12	3999.36	11250.23	27374.4	3187.62	1306.60
2022-12-13	4030.01	11400.01	27374.4	3207.62	1301.42

실제 예측

```
y_pred_rf = pd.DataFrame(rf_reg.predict(X_test_pred))
y_pred_rf.index = X_test_pred.index
y_pred_rf.columns = ['랜덤포레스트']

y_pred_tree = pd.DataFrame(tree_reg.predict(X_test_pred))
y_pred_tree.index = X_test_pred.index
y_pred_tree.columns = ['결정트리']

y_pred_lin = pd.DataFrame(lin_reg.predict(X_test_pred))
y_pred_lin.index = X_test_pred.index
y_pred_lin.columns = ['리니어']

y_pred_gbr = pd.DataFrame(best_model.predict(X_test_pred))
y_pred_gbr.index = X_test_pred.index
y_pred_gbr.columns = ['그라디언트']
```

```
y_pred_all = pd.concat([y_pred_rf, y_pred_tree, y_pred_lin, y_pred_gbr], axis=1)
```

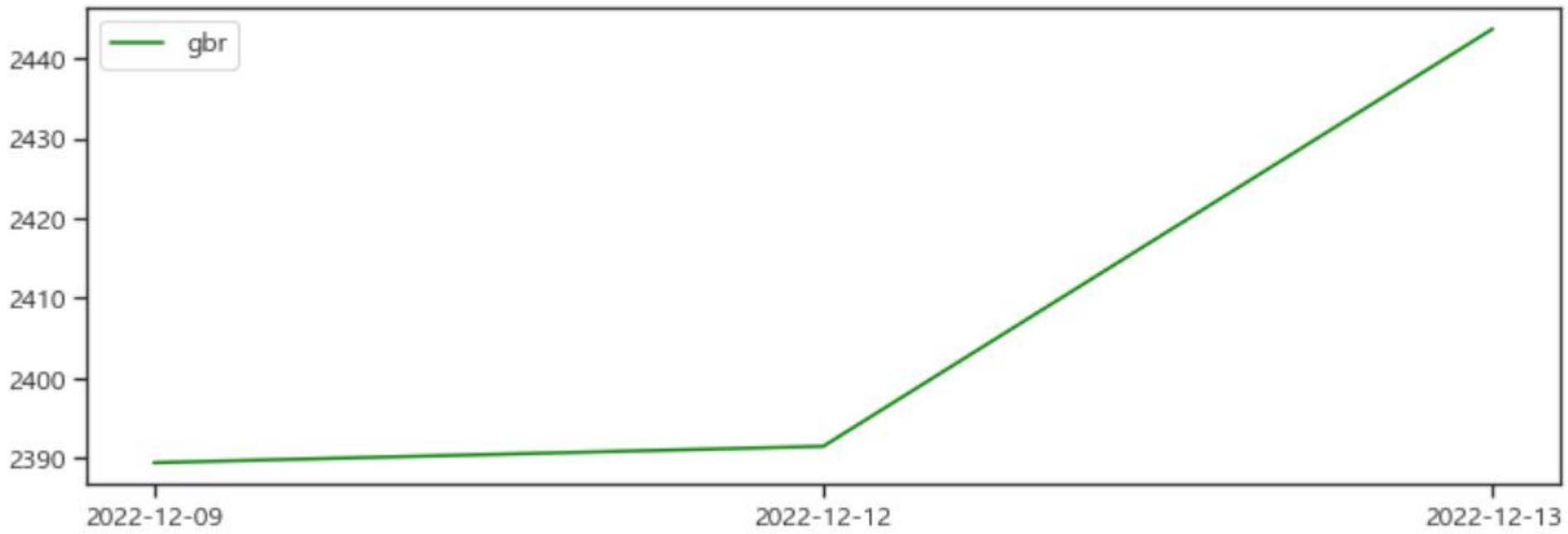
```
y_pred_all
```

	랜덤포레스트	결정트리	리니어	그라디언트
2022-12-09	2383.2452	2376.46	2510.533595	2389.473800
2022-12-12	2388.3552	2376.46	2525.101485	2391.517828
2022-12-13	2446.7029	2452.25	2538.800764	2443.632869

실제 예측 그래프

```
fig = plt.figure(figsize = (12, 4))
chart = fig.add_subplot(1,1,1)

chart.plot(y_pred_gbr, color='green' , label='gbr')
plt.legend(loc = 'best')
```

```
<matplotlib.legend.Legend at 0x16ee6525f70>
```



Merchine Learning

Thanks for
감사합니다

Team 3

남정우, 전현준, 정제경, 전한성

End