

분류와 회귀를 이용한 투자모델 설정과 투자모델검증 시각화

월간 데이콘 KOSPI 기반 분석 시각화 경진대회

2022-09-01 : 2022-10-04

Index

1. 데이터개요
 2. 데이터 시각화
 3. 분류를 이용한 투자모델
 4. 회귀를 이용한 투자모델
 5. 대시보드
-

1. 데이터 개요

1. 데이터 개요

- Date 컬럼의 데이터타입을 변경 해주고 label 컬럼을 추가 했습니다.
- label 기준은 change ≥ 0 인 경우, 1로 레이블링 됩니다.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11024 entries, 0 to 11023
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   Date      11024 non-null   datetime64[ns]
 1   Close     11024 non-null   float64
 2   Open      11024 non-null   float64
 3   High      11024 non-null   float64
 4   Low       11024 non-null   float64
 5   Volume    11024 non-null   float64
 6   Change    11024 non-null   float64
 7   label     11024 non-null   int64  
dtypes: datetime64[ns](1), float64(6), int64(1)
memory usage: 689.1 KB
None
```

2. 데이터 시각화

2. 데이터 시각화

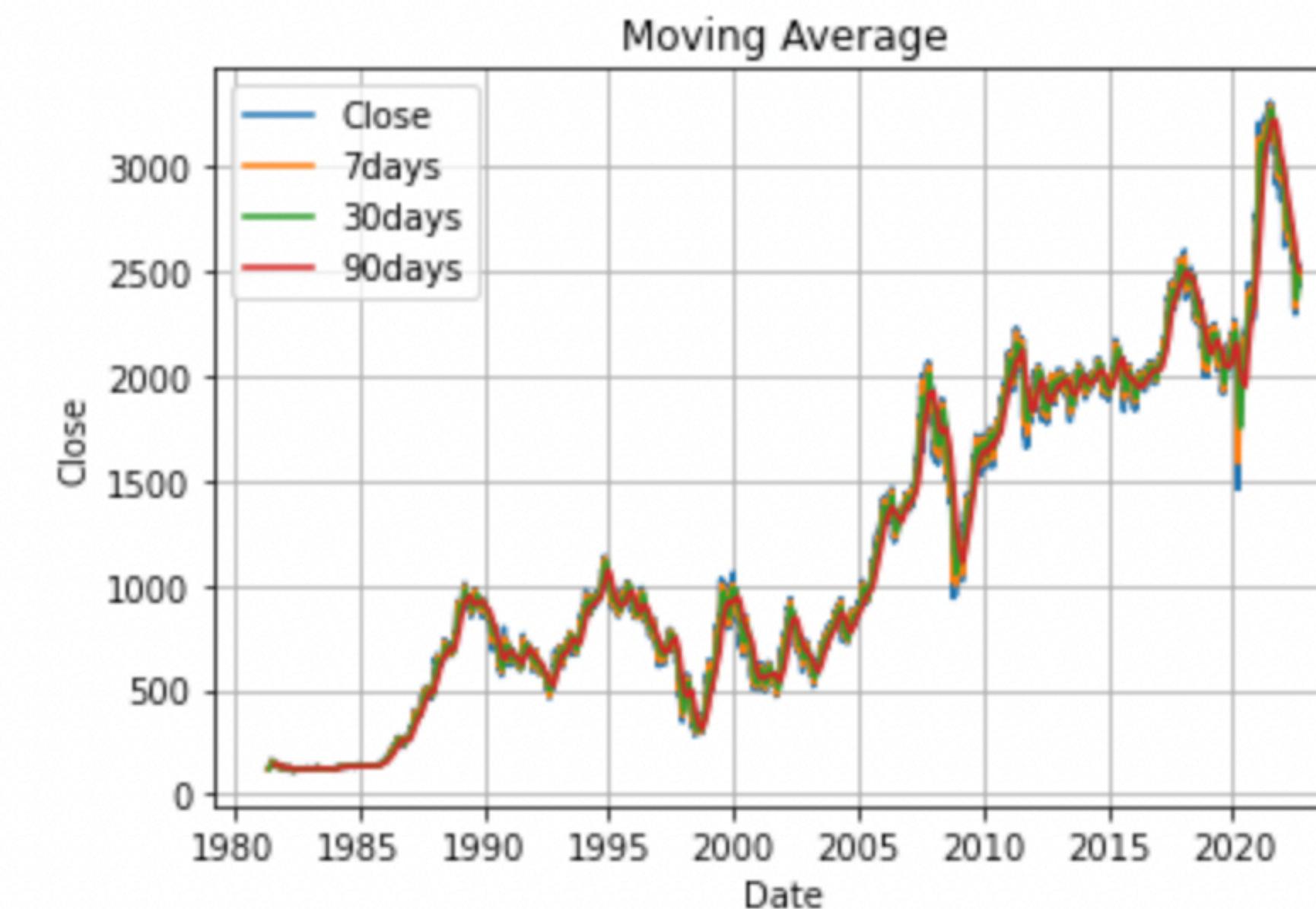
- pandas rolling 을 이용한 이동평균선입니다.
- 데이터 기간을 설정해서 기간을 조정 할 수 있습니다.
- grouper 를 이용하여 주간, 월간 .. 등 으로 변환한 이동평균선도 가능합니다.

이동평균선

```
plt.plot(df[ 'Date' ],df[ 'Close' ].rolling(window=1).mean())
plt.plot(df[ 'Date' ],df[ 'Close' ].rolling(window=7).mean())
plt.plot(df[ 'Date' ],df[ 'Close' ].rolling(window=30).mean())
plt.plot(df[ 'Date' ],df[ 'Close' ].rolling(window=90).mean())

plt.grid()
plt.xlabel( 'Date' )
plt.ylabel( 'Close' )
plt.title( 'Moving Average' )
plt.legend([ 'Close' , '7days' , '30days' , '90days' ])
```

<matplotlib.legend.Legend at 0x12f39eeb0>



2. 데이터 시각화

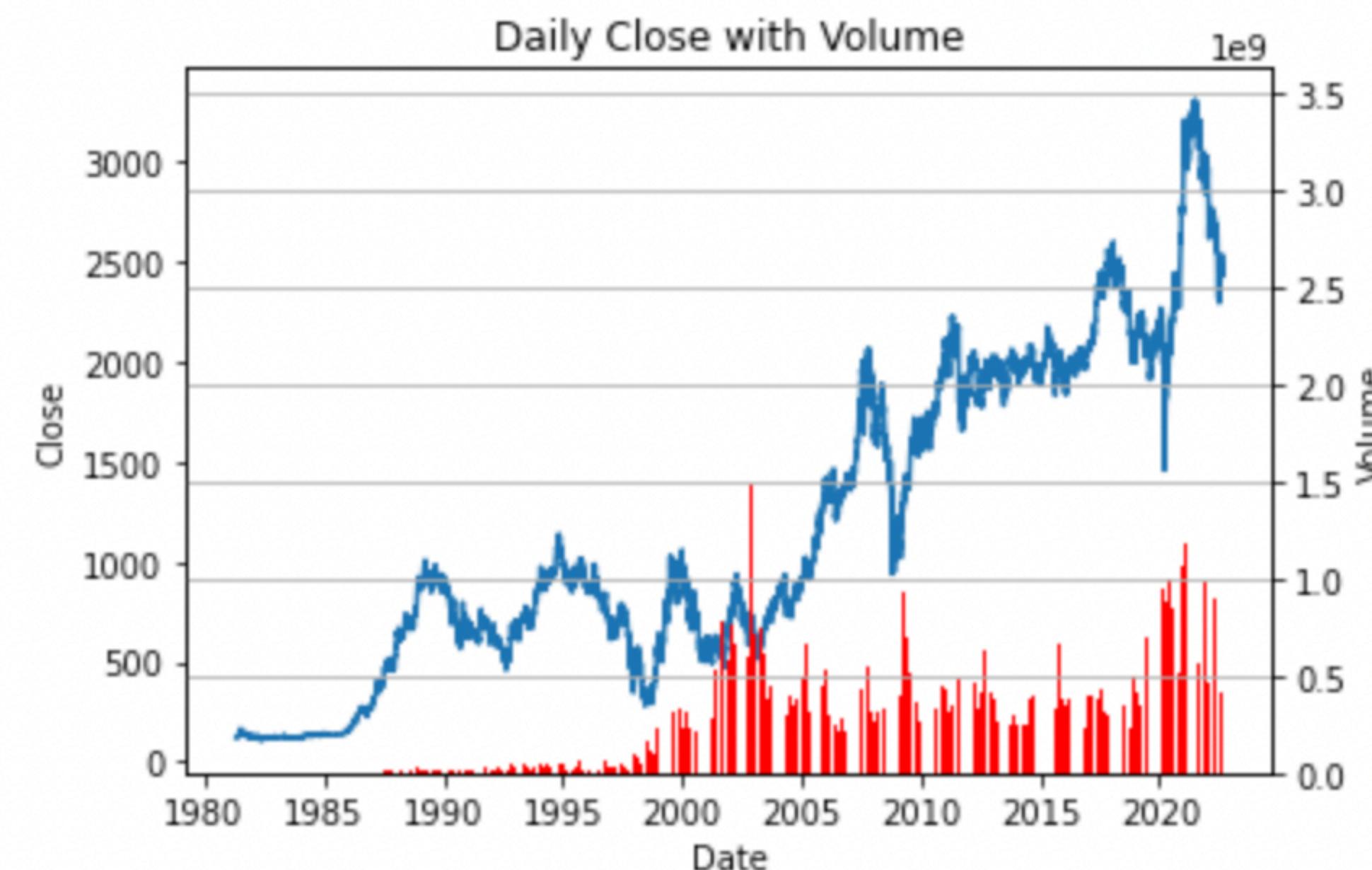
- matplotlib 의 이중축을 이용한 그래프입니다.
- 마찬가지로 기간조정, grouper 이용 가능합니다.

Close with Volume

```
fig, ax1 = plt.subplots()
ax1.plot(df[ 'Date' ], df[ 'Close' ])
ax1.set_xlabel('Date')
ax1.set_ylabel('Close')

ax2 = ax1.twinx()
ax2.set_ylabel('Volume')
ax2.bar(df[ 'Date' ], df[ 'Volume' ],color='red')

plt.title('Daily Close with Volume')
plt.grid()
```



2. 데이터 시각화

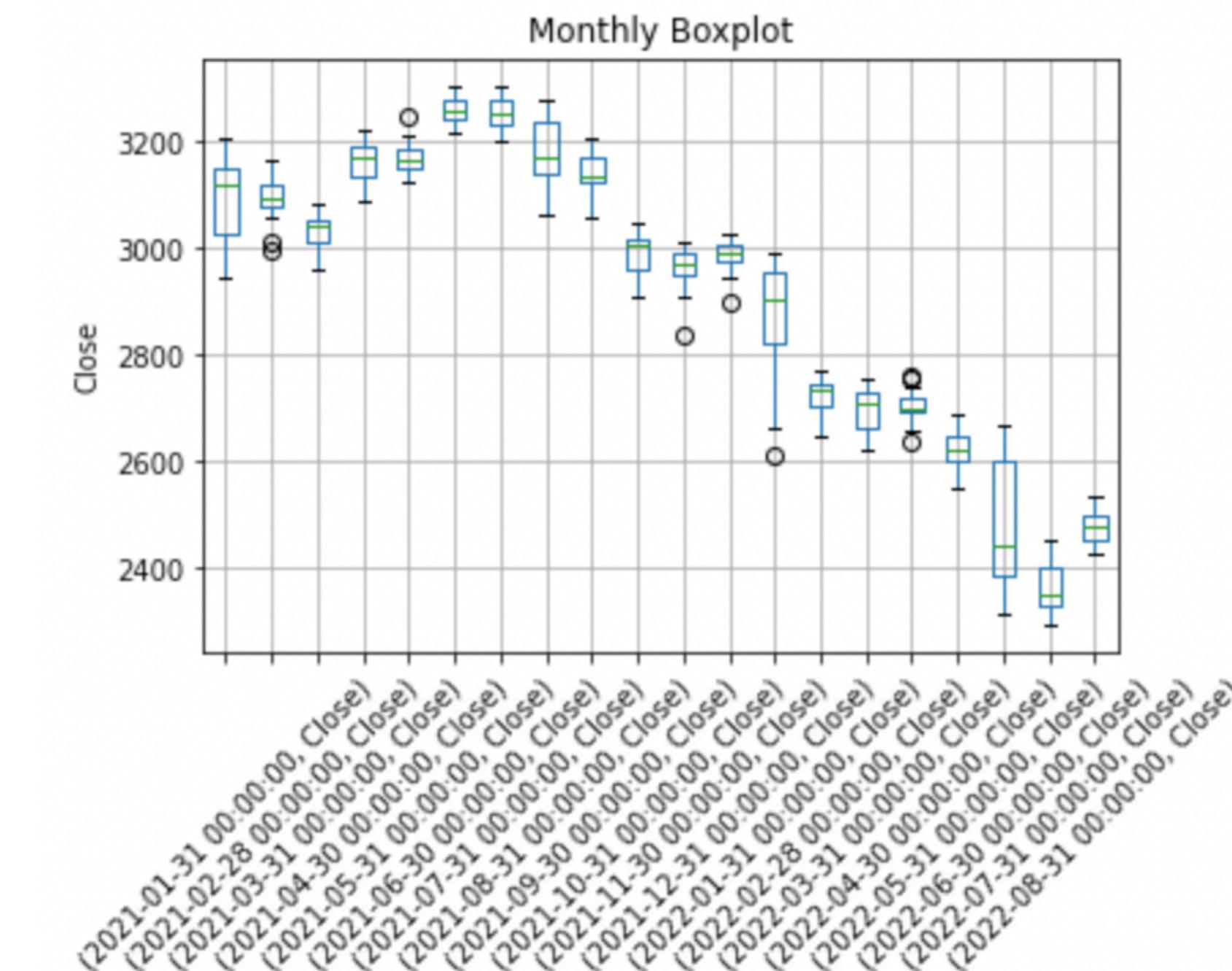
Monthly boxplot

- 일,주,월,연 단위 옵션 가능

```
df[df['Date'] > '2021-01-01'].loc[:,['Date','Close']].groupby(  
    [pd.Grouper(key='Date', freq='1M')]).boxplot(  
    subplots=False,rot=45)
```

```
plt.title('Monthly Boxplot')  
plt.ylabel('Close')
```

```
Text(0, 0.5, 'Close')
```



- pandas grouper를 응용합니다.
- 월간 데이터들의 boxplot 을 시각화 하였습니다.
- 추세, 월간 변화폭 등을 체크할 수 있습니다.

2. 데이터 시각화

- scipy 인터폴레이팅을 이용합니다.
- 실제데이터를 보간하고 나머지 부분을 cubic spline 데이터로 채웁니다.
- 채워진 데이터로 부드러운 곡선을 시각화 했습니다.

Spline smoothing

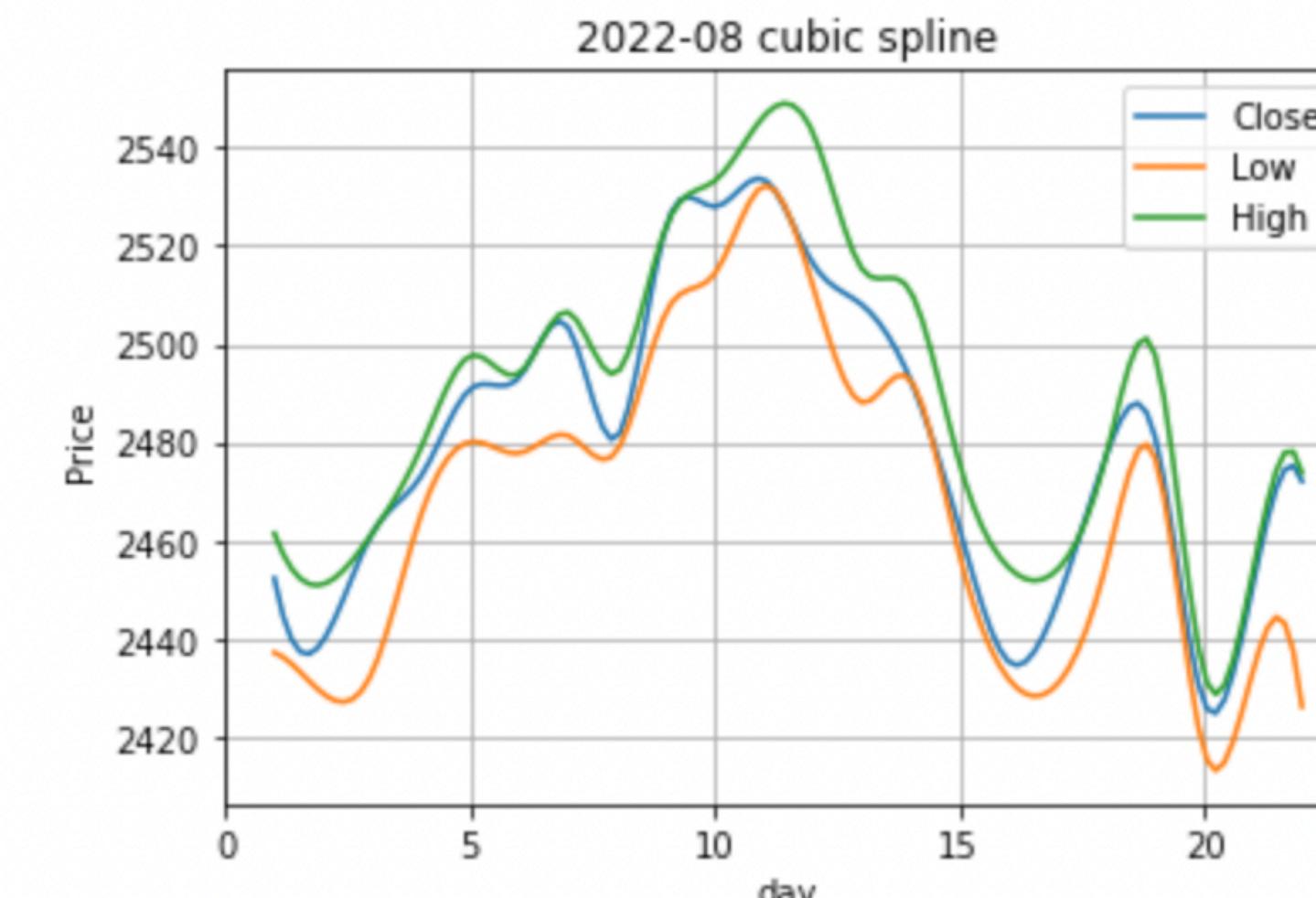
```
: def spline(feature):
    y = np.array(df[df['Date'] >= '2022-08-01'][feature])
    x = np.linspace(1., 22., 22)
    sp = interpolate.interp1d(x,y,kind='cubic')

    # sp = csaps.csaps(x, y, smooth=0.8)
    xs = np.linspace(x[0], x[-1], 120)
    ys = sp(xs)
    return xs,ys
```

```
: xc,yc = spline('Close')
xl,yl = spline('Low')
xh,yh = spline('High')

plt.plot(xc,yc)
plt.plot(xl,yl)
plt.plot(xh,yh)

plt.legend(['Close','Low','High'])
plt.title('2022-08 cubic spline')
plt.ylabel('Price')
plt.xlabel('day')
plt.grid()
```



3. 분류를 이용한 투자모델

3. 분류를 이용한 투자모델

- 사용 데이터는 2000 이후 데이터입니다.
- 전일 상승 했을 경우, 다음날도 오를것으로 기대하는 클래식한 투자방법이 있습니다.
- 위 투자방법의 승률을보고 분류를 적용한 투자모델을 정의했습니다.

	Date	Close	Open	High	Low	Volume	Change	label	df_split
0	2000-01-04	1059.04	1028.33	1066.18	1016.59	195900000.0	0.0301	1	1
1	2000-01-05	986.31	1006.87	1026.52	984.05	257700000.0	-0.0687	0	1
2	2000-01-06	960.79	1013.95	1014.90	953.50	203520000.0	-0.0259	0	0
3	2000-01-07	948.65	949.17	970.16	930.84	215660000.0	-0.0126	0	0
4	2000-01-10	987.24	979.67	994.94	965.02	240180000.0	0.0407	1	0
...
5592	2022-08-25	2477.26	2459.79	2477.26	2455.32	426230000.0	0.0122	1	1
5593	2022-08-26	2481.03	2489.14	2497.76	2476.75	520090000.0	0.0015	1	1
5594	2022-08-29	2426.89	2432.06	2432.89	2417.01	448750000.0	-0.0218	0	1
5595	2022-08-30	2450.93	2441.21	2453.91	2433.48	327210.0	0.0099	1	0
5596	2022-08-31	2472.05	2433.47	2473.75	2426.14	397290.0	0.0086	1	1

5597 rows × 9 columns

3. 분류를 이용한 투자모델

- 2차원 ndarray 형식의 arr 변수를 만들어 맵핑 해보았습니다.
- 실제로 전날 상승한 경우, 현재도 상승하는 경우의 수가 가장 많았습니다.
- 투자방법은 sp_label이 1로 판단되는 경우 투자를 해야하고 label이 1인 경우가 가장 많아야 승률이 높기 때문에 metric을 정밀도로 설정합니다.

```
import seaborn as sns
sns.heatmap(arr,xticklabels=['sp_label:0','sp_label:1'],
            yticklabels=['label:0','label:1'],annot=True,fmt='d')
```

<AxesSubplot:>



3. 분류를 이용한 투자모델

- 각 method 별로 튜닝을 통해 모델을 찾습니다.
- dtc 를 예제형식으로Parms 구성품 맵핑을 합니다.
- 가장높은 정밀도를 보인 grad method 를 채택합니다.

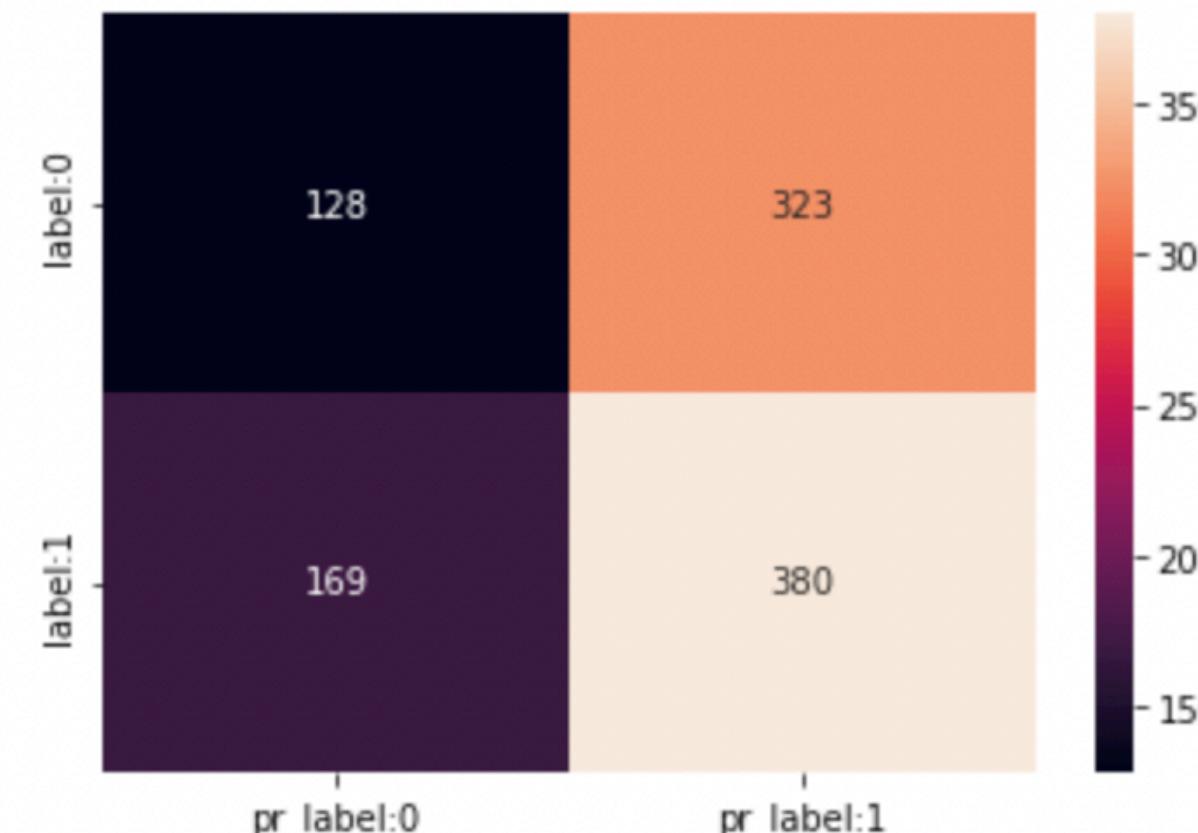
	Method	Parms	Precision
0	dtc	[8, 6, 2, [Volume, Open, High, Low]]	56.73
1	logi	[0, [Volume, Open, High, Low]]	53.37
2	knn	[8, [Volume, Open, High, Low]]	57.00
3	vote	[[Volume, Open, High, Low]]	52.42
4	ranf	[8, 1, 3, 30, [Volume, Open, High, Low]]	55.80
5	grad	[9, 4, 9, 0.1, [Volume, Open, High, Low]]	58.38
6	xgb	[[Volume, Open, High, Low]]	53.57

```
DecisionTreeClassifier(random_state=10,  
                      max_depth=max_depth,  
                      min_samples_leaf=min_samples_leaf,  
                      min_samples_split=min_samples_split)
```

3. 분류를 이용한 투자모델

- 1000개의 데이터로 모델검증을 진행했습니다. (약 18%의 데이터)
- 해당모델의 경우 승률은 54.1% 입니다.
- pr_label 이 1인 경우 투자를 해야하므로 해당경우 데이터를 모읍니다.

```
heat = pd.DataFrame([[i,j] for i,j in zip(list(df['label'])[-1000:],  
                                         list(pr.bundle(pr.GradB(9,4,9,0.1))  
                                         heat.columns = ['label','split_label']  
                                         arr = np.array([[0,0],[0,0]])  
  
                                         tar = pd.DataFrame(heat.value_counts()).reset_index()  
                                         for i in range(2):  
                                             tp = tar[tar['label'] == i]  
                                             for j in range(2):  
                                                 arr[i][j] = tp[tp['split_label'] == j][0]  
  
                                         arr  
  
                                         array([[128, 323],  
                                                [169, 380]])  
  
                                         sns.heatmap(arr,xticklabels=['pr_label:0','pr_label:1'],  
                                         yticklabels=['label:0','label:1'],annot=True,fmt='d')  
  
<AxesSubplot:>
```



3. 분류를 이용한 투자모델

- pr_label 이 1인 경우 투자를 해야하므로 해당경우 데이터를 모읍니다.
- 장이 시작할때 구매하고 장이 종료되는 시점에 판다고 가정합니다.
- 총 703회의 투자가 일어납니다.
- 투자케이스의 손실합은 -388이고 평균 구매가는 2594 입니다.
- metric 으로서 역할을할 수 있습니다.

	Date	Close	Open	High	Low	Volume	Change	Inv_case
0	2018-08-10	2282.79	2295.21	2295.62	2277.89	286160000.0	-0.0091	1
7	2018-08-22	2273.33	2273.68	2280.31	2268.91	269270000.0	0.0014	1
9	2018-08-24	2293.21	2276.16	2295.22	2271.35	347290000.0	0.0046	1
10	2018-08-27	2299.30	2297.32	2302.87	2289.41	307040000.0	0.0027	1
11	2018-08-28	2303.12	2312.14	2314.60	2299.45	276890000.0	0.0017	1
...
990	2022-08-18	2508.05	2499.30	2515.37	2488.09	381630000.0	-0.0033	1
991	2022-08-19	2492.69	2510.72	2510.72	2492.69	459830000.0	-0.0061	1
992	2022-08-22	2462.50	2467.38	2475.77	2457.08	422550000.0	-0.0121	1
995	2022-08-25	2477.26	2459.79	2477.26	2455.32	426230000.0	0.0122	1
997	2022-08-29	2426.89	2432.06	2432.89	2417.01	448750000.0	-0.0218	1

703 rows × 8 columns

```
(df['Close'] - df['Open']).sum()
```

```
-388.0799999999834
```

```
df['Open'].mean()
```

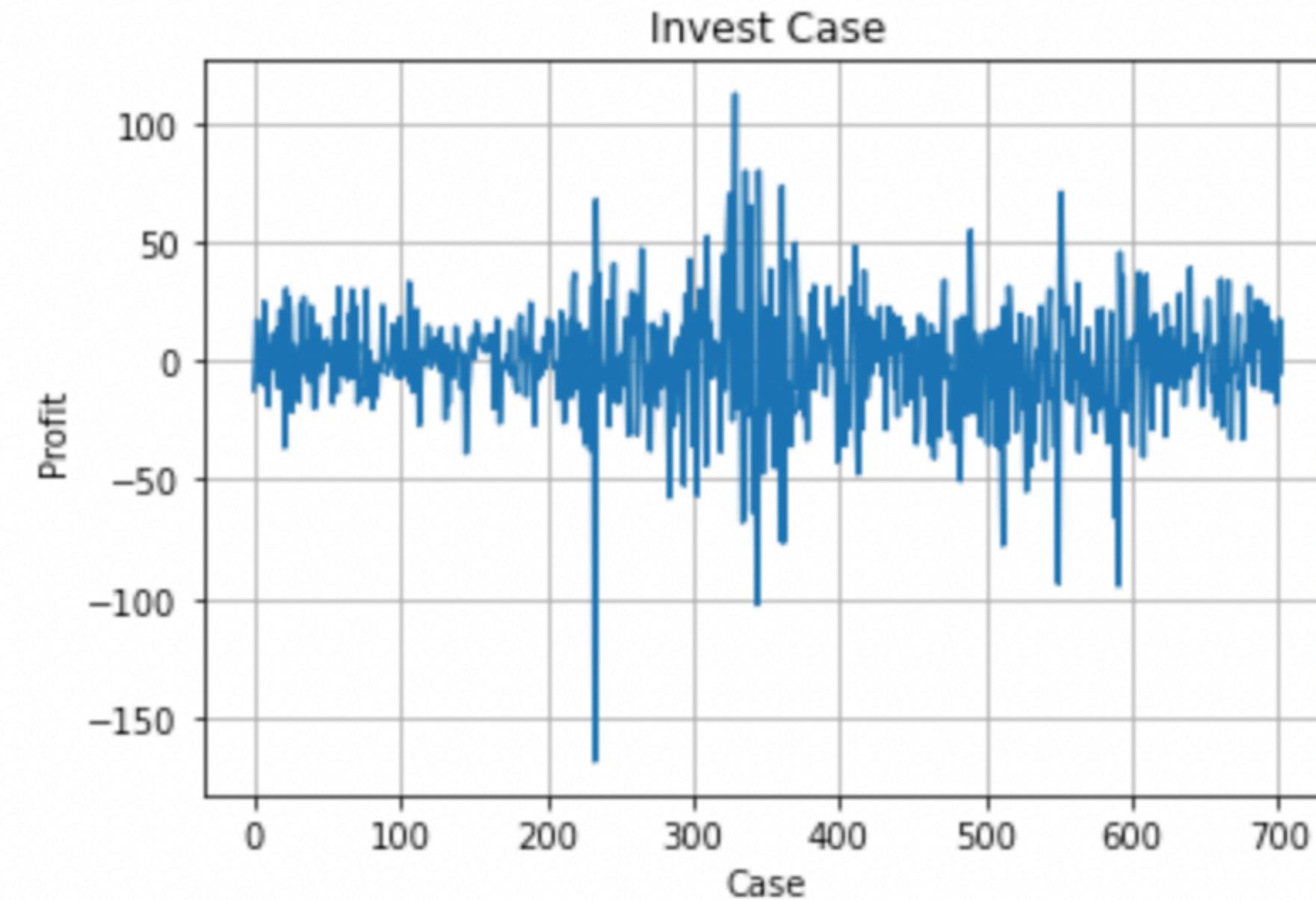
```
2593.932660028449
```

3. 분류를 이용한 투자모델

- 투자케이스에 대한 결과 시각화입니다.
- 결과분포가 0에 근접하여 형성됩니다.
- 좋은 투자모델은 그래프 분포가 0위로 형성되게 됩니다.

```
plt.plot(range(len(df)),df[ 'Close' ] - df[ 'Open' ])
plt.grid()
plt.title( 'Invest Case' )
plt.ylabel( 'Profit' )
plt.xlabel( 'Case' )

Text(0.5, 0, 'Case')
```



4. 회구를 이용한 투자모델

4. 회귀를 이용한 투자모델

- 마찬가지로 2000년 이후 데이터를 사용합니다.
- 회귀를 이용했을때, 너무 좋은 rmse가 나오는 경우는 지양해야 된다고 생각합니다. (이익 혹은 손실이 너무 작아지는 현상)
- 1000개의 데이터는 컷팅 해두고 test rmse를 확인했습니다.

```
x = df.loc[:len(df)-1001,features]
y = df.loc[1:len(df)-1000,'Close']
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5)
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)

(3677, 4) (920, 4) (3677,) (920,)

from sklearn.metrics import mean_squared_error,r2_score
lr.fit(x_train,y_train)
pred = lr.predict(x_test)
mse = mean_squared_error(pred,y_test)
rmse = np.sqrt(mse)

rmse

18.96481764512902
```

4. 회귀를 이용한 투자모델

- 당일 Open 가격보다 예측가격이 높게 측정되는 경우 투자를한다고 가정합니다.
- 장이 시작할때 구매하고 장이 종료되는 시점에 판다고 가정합니다.
- 총 452회의 투자가 발생합니다.
- 시뮬레이션 실제 손실합은 -294로 측정되고, 평균 구매가는 2471 입니다.
- 마찬가지로 위 정보를 metric으로 설정 할 수 있습니다.

	Date	Volume	Open	High	Low	Close	pred
0	2018-08-10	286160000.0	2295.21	2295.62	2277.89	2282.79	2295.369908
1	2018-08-13	335380000.0	2266.43	2271.82	2238.55	2248.45	2282.542599
2	2018-08-14	255220000.0	2249.86	2262.52	2247.37	2258.91	2250.020383
3	2018-08-16	312520000.0	2233.05	2244.08	2218.09	2240.80	2259.422372
6	2018-08-21	294340000.0	2248.04	2272.86	2244.59	2270.06	2254.431407
...							
990	2022-08-18	381630000.0	2499.30	2515.37	2488.09	2508.05	2523.779718
992	2022-08-22	422550000.0	2467.38	2475.77	2457.08	2462.50	2497.157119
993	2022-08-23	471170000.0	2449.31	2454.55	2431.83	2435.34	2467.171299
997	2022-08-29	448750000.0	2432.06	2432.89	2417.01	2426.89	2487.528186
999	2022-08-31	397290.0	2433.47	2473.75	2426.14	2472.05	2446.767072

452 rows × 7 columns

```
print((conf['pred'] - conf['Open']).sum()) # 예상  
print((conf['Close'] - conf['Open']).sum()) # 실제
```

```
6214.088573157267  
-294.9299999999914
```

```
conf['Open'].mean()
```

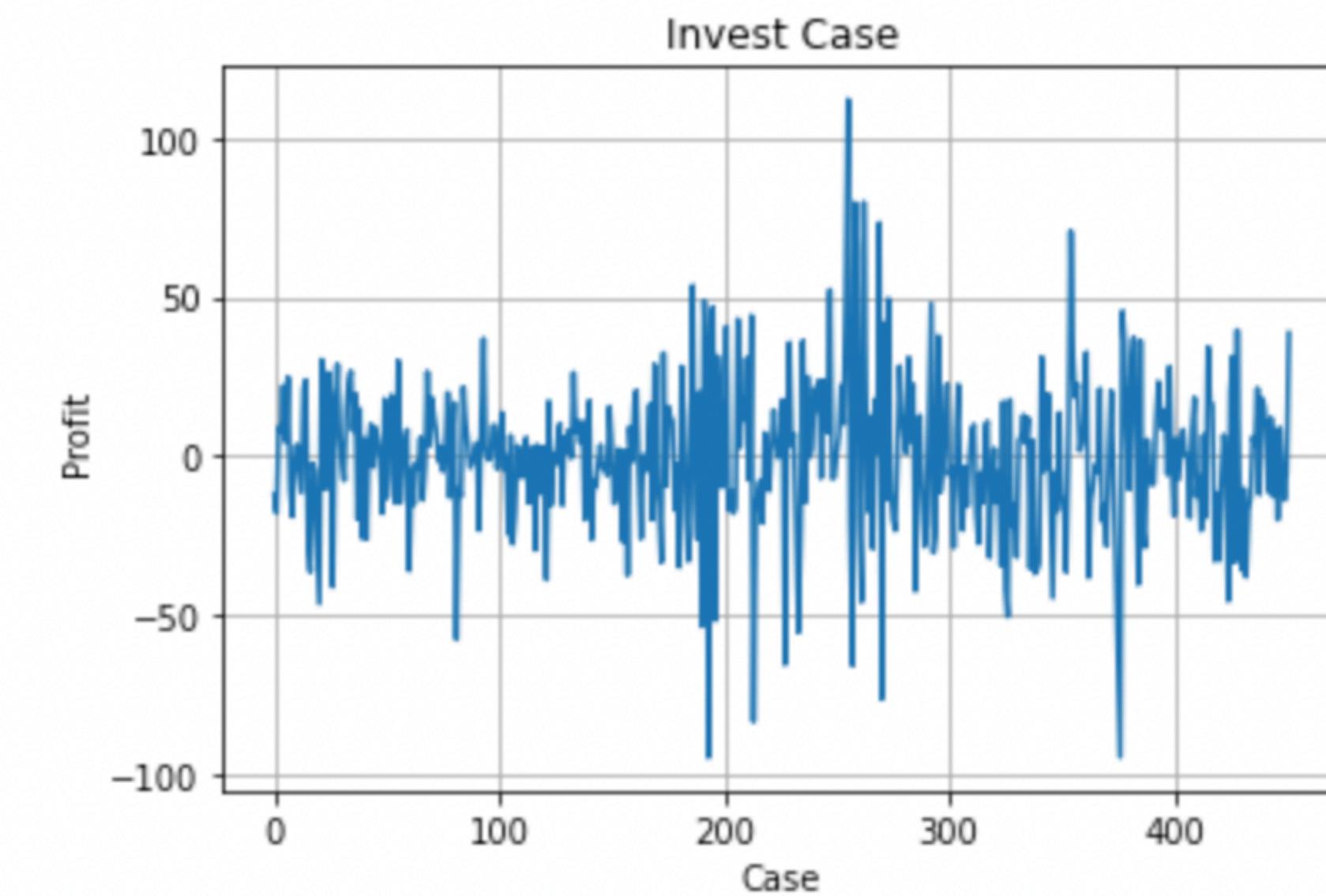
```
2470.9779867256634
```

4. 회귀를 이용한 투자모델

- 투자케이스에 대한 결과 시각화입니다.
- 결과분포가 0에 근접하여 형성됩니다.
- 좋은 투자모델은 그래프 분포가 0위로 형성되게 됩니다.
- lstm, gru 등 딥러닝 모델에도 같은 규칙을 이용할 수 있습니다.

```
plt.plot(range(len(conf)),conf[ 'Close' ] - conf[ 'Open' ])
plt.grid()
plt.title('Invest Case')
plt.ylabel('Profit')
plt.xlabel('Case')

Text(0.5, 0, 'Case')
```

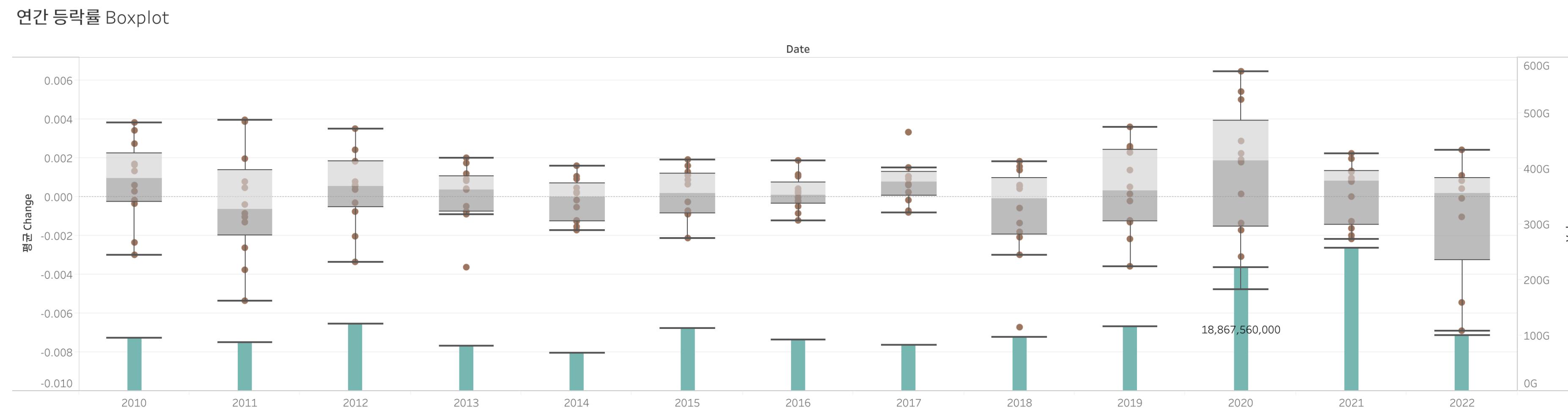


5. 대시보드

5. 대시보드

- 각 단계의 데이터프레임을 뽑아 대시보드를 구성했습니다.
- 대시보드는 태블로로 작성되었습니다.
- 참고용으로 png 파일만 슬라이드에 삽입하였습니다.

5. 대시보드



5. 대시보드

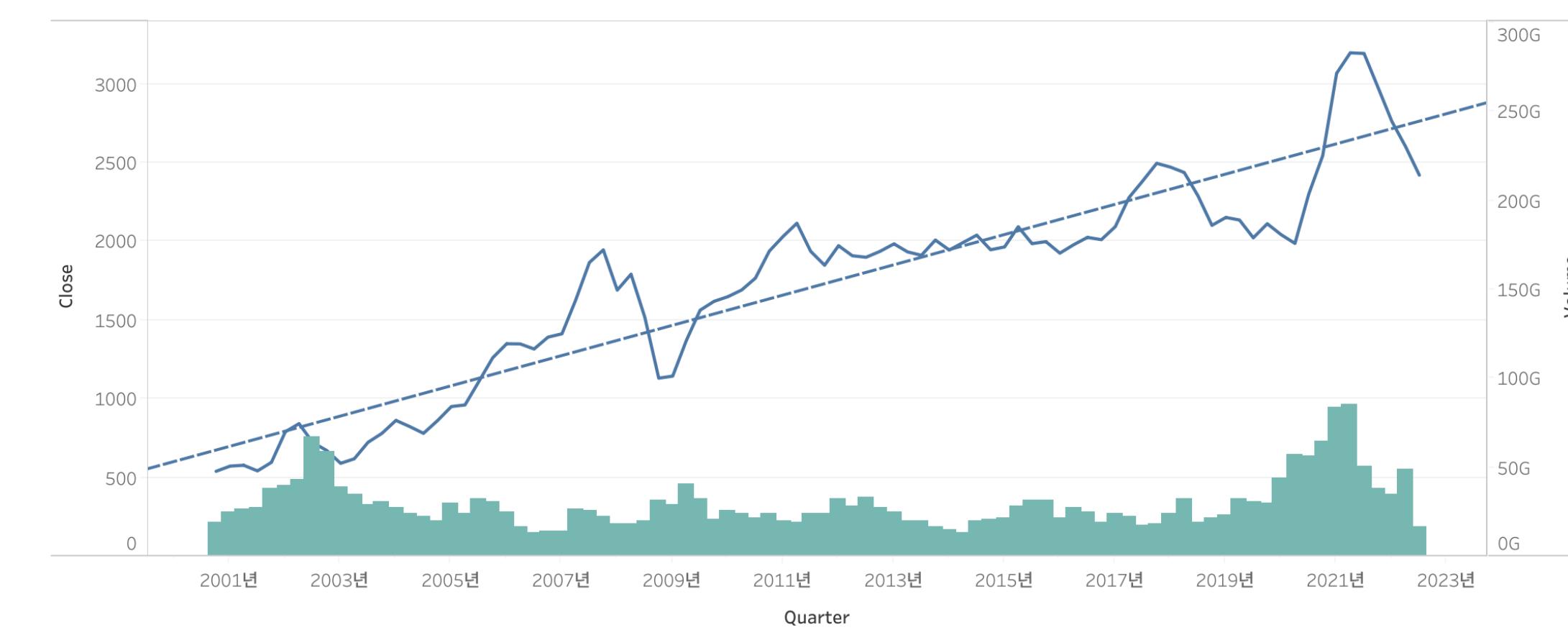
2022년 일간 그래프



2022 데이터표

Date일	Open	High	Low	Close	Volume
2022년 1월 3일	2,998	3,011	2,979	2,989	435,820,000
2022년 1월 4일	2,992	2,995	2,973	2,989	621,550,000
2022년 1월 5일	2,984	2,986	2,937	2,954	787,350,000
2022년 1월 6일	2,925	2,953	2,915	2,921	786,040,000
2022년 1월 7일	2,934	2,959	2,933	2,955	546,170,000
2022년 1월 10일	2,947	2,951	2,911	2,927	477,400,000
2022년 1월 11일	2,931	2,944	2,910	2,927	566,170,000
2022년 1월 12일	2,951	2,973	2,950	2,972	519,500,000
2022년 1월 13일	2,980	2,982	2,958	2,962	604,870,000
2022년 1월 14일	2,938	2,945	2,915	2,922	532,670,000
2022년 1월 17일	2,919	2,920	2,876	2,890	593,260,000
2022년 1월 18일	2,899	2,903	2,857	2,864	554,040,000
2022년 1월 19일	2,840	2,872	2,832	2,842	463,830,000
2022년 1월 20일	2,842	2,863	2,831	2,863	395,670,000
2022년 1월 21일	2,837	2,848	2,817	2,834	532,750,000
2022년 1월 24일	2,824	2,828	2,781	2,792	473,990,000
2022년 1월 25일	2,796	2,790	2,704	2,720	639,960,000
2022년 1월 26일	2,730	2,744	2,708	2,709	472,650,000
2022년 1월 27일	2,709	2,723	2,614	2,614	487,730,000
2022년 1월 28일	2,618	2,669	2,592	2,663	434,190,000

분기간 종가와 거래량

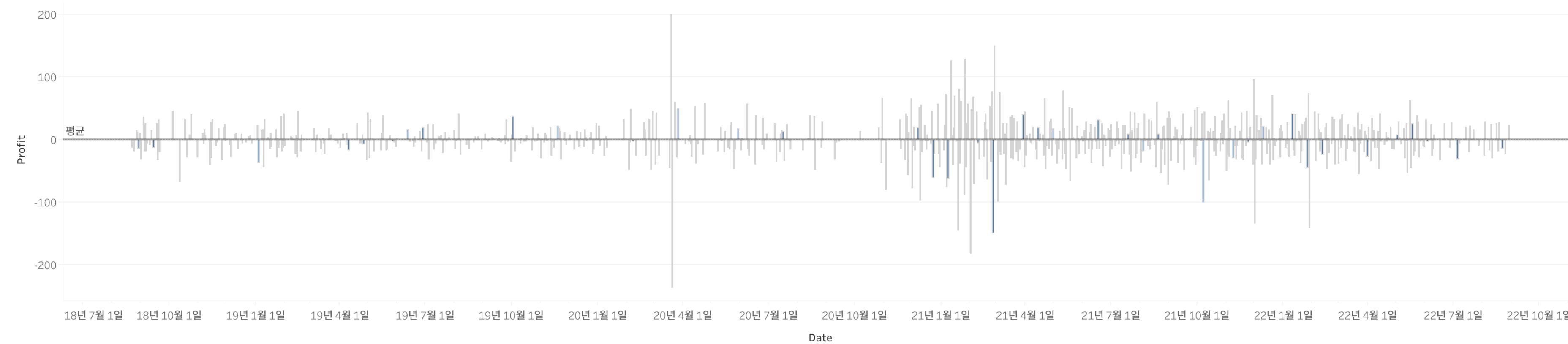


Date의 분기
2000년 3분기 - 2022년 3분기

측정값 이름
Close
High
Low
Open
Volume
Date일 하이라이트
하이라이트된 항목 없음

5. 대시보드

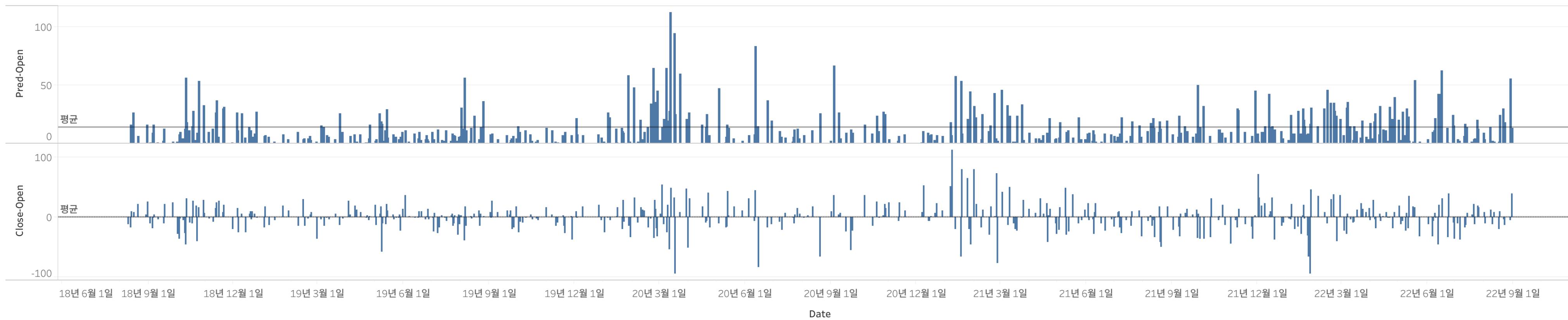
분류를 이용한 투자모델



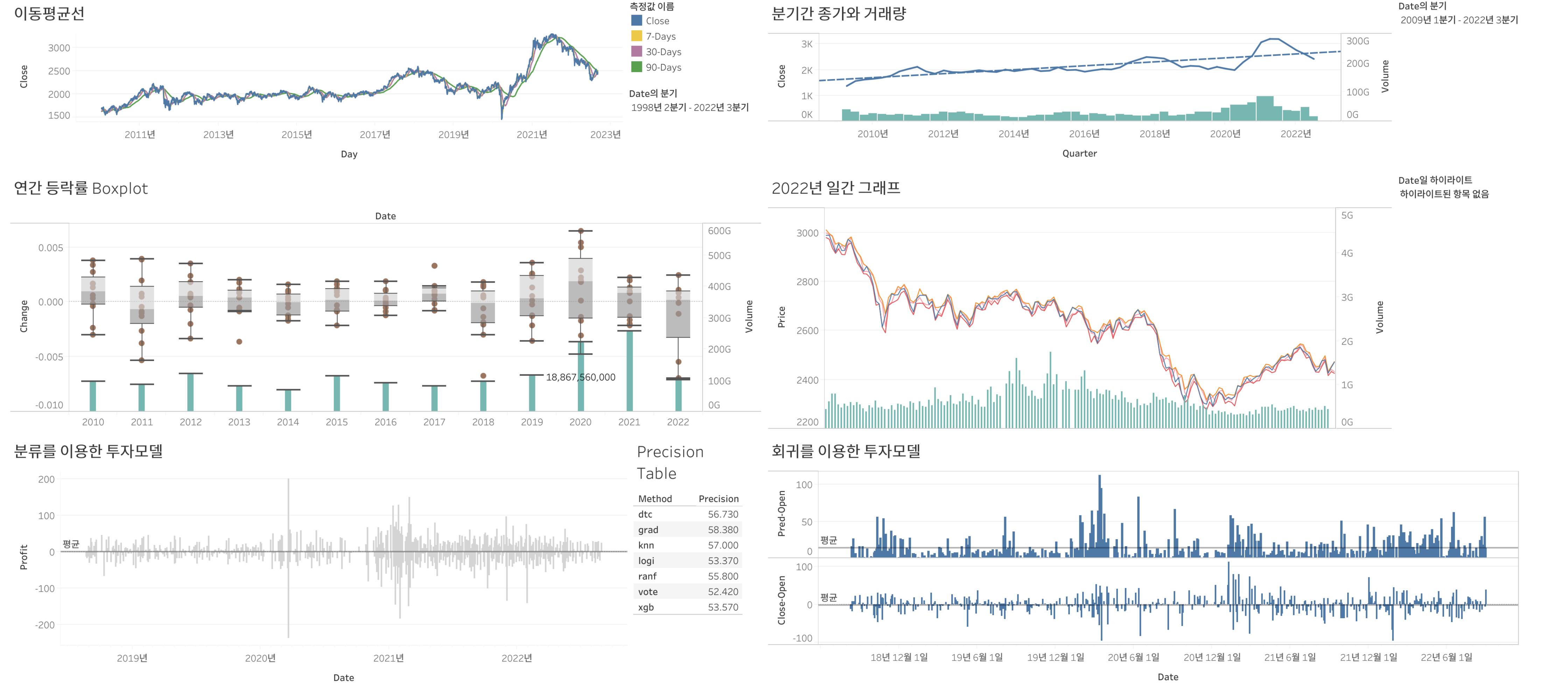
Precision Table

Method	Precision
dtc	56.730
grad	58.380
knn	57.000
logi	53.370
ranf	55.800
vote	52.420
xgb	53.570

회귀를 이용한 투자모델



5. 대시보드



마치며..

- 분류의 경우 투자 성공케이스가 많이 생성되게 만든 모델이여서 좋은모델을 찾기는 쉽지않다고 판단됩니다.
- 그러나 아직 시도해볼방법은 많습니다. 기간과 피쳐를 추가하고 방법을 추가할 수 있습니다.
- 회귀의 경우 노이즈삽입의 정도에따라 좋은모델을 찾을 수 있을것으로 판단됩니다. 피쳐를 추가하고 피쳐에대한 가중치를 설정하여 모델서치를하고 투자케이스 이익평균을 metric으로 베스트모델을 찾는방식을 생각할 수 있습니다.
- 회귀, lstm, gru 등의 딥러닝방식에서 사용할 수 있습니다.
- 투자규칙에 따라 승률이나 이익이 달라질 수 있습니다.
- AI 투자자동화 모델생성시 꼭 선행되어 검증되어야할 부분이라 생각합니다.