투자심리지수 제작

: 매수-매도 불균형과 여러 경제/금융 변수들에 대하여

TEAM 통테이토

김가영 남화승 이가영 이나현

1. 프로젝트 개요 및 목적

- **투자심리지수** : 투자심리의 변화를 일정기간 동안 파악하여 장세의 과열도를 측정하는 지표
- **투자자 심리** : 투자자의 공통된 판단오류로 인해 가격을 잘못 측정하는 현상을 의미한다.
- **개인투자자의 매수- 매도 거래량** : 투자심리지수의 대용변수로써 활용 가능

  종속변수(y)로써 일별 개인 매수-매도 거래량 불균형을 사용하여 투자심리지수를 구축하고자 한다.

'Investor Sentiment and the Cross-Section of Stock Returns-Baker and Wurgler(2006)'에 의해,
다변수 투자심리지수의 설명력이 높다는 점에 착안

→ 매수매도 불균형에 대해 여러가지 경제 및 금융 변수들로 회귀 분석을 진행함으로써 투자심리지수를 구축

2. 변수

1) 종속변수(y)

개인 매수-매도 거래량을 스케일링한 일별 개인매수매도 불균형

2) 독립변수(x)

1. CPI: 소비자 물가지수
2. CD: 주식투자 예치금, 추세보다 많은 금액이 예치되어 있을수록 투자자들은 낙관적인 심리를 보인다.
3. 기준금리: 거시 경제 지표로 시장의 유동성을 결정하는 기준. 여기서는 한국은행 기준금리만을 적용.
4. 소비자심리지수: 현재생활형편, 생활형편전망 등 6개 영역을 지수화한 지표
5. VKOSPI: 코스피 변동성 지수. 시장의 변동성 수준에 따라 시장 참여자들의 기대 심리가 달라진다.
6. 거래량: 코스피 시장 내 일별 전체 거래량 데이터. 투자자들의 과열 수준 판단 가능.
7. 상장주식 거래회전율: 시장 내 유동성을 판단 가능. (회전율↑ → 투기적 성향↑)
8. KOSPI MA120: 코스피 지수에 대한 120일 이동평균선. 시장의 거시적 추세 파악 가능
9. 환율: 원/달러 기준 환율을 채택

3. EDA

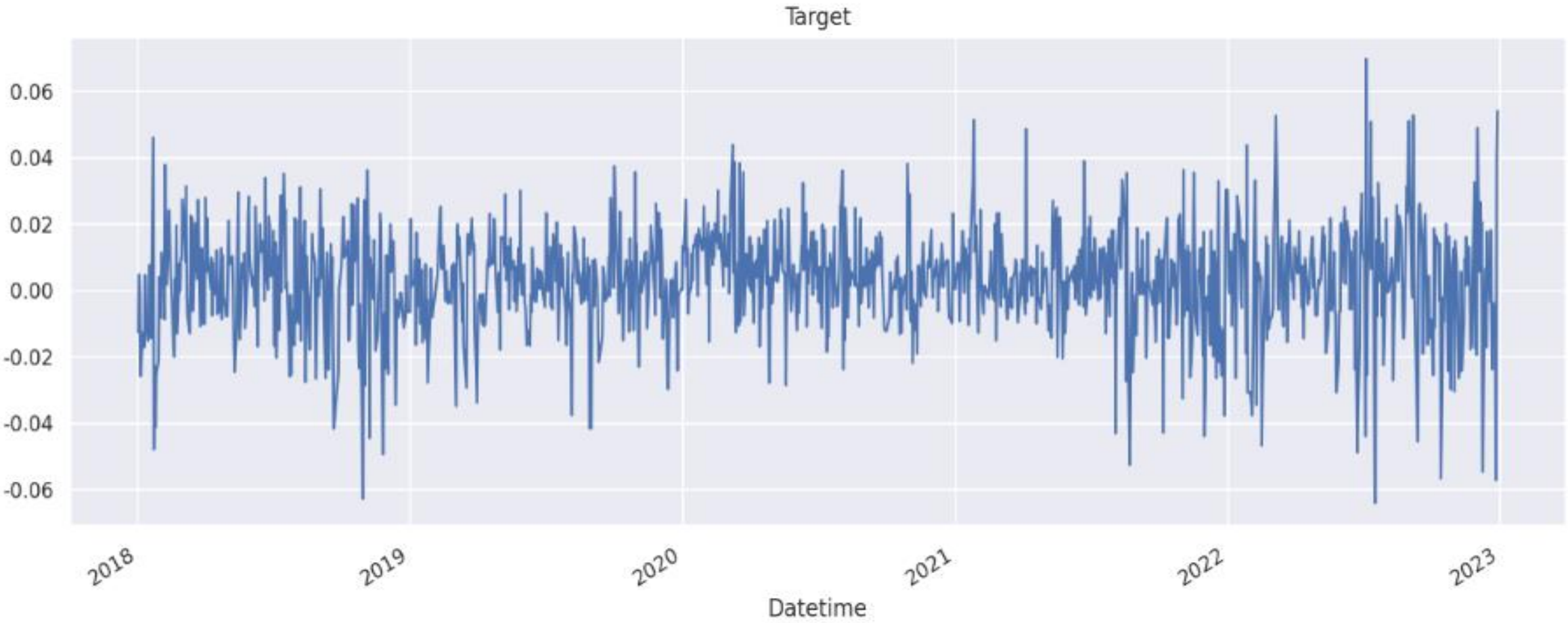
1) DATA SET

Datetime	cpi	cd	cons_sent	exchange	base	rt_rate	vkospi	volume	kospi_ma120	target
2018-01-02	98.106	27321579	111.0	1061.2	1.50	19.34	12.66	262205	2438.634831	-0.012518
2018-01-03	98.106	29106445	111.0	1064.5	1.50	19.34	12.64	331095	2439.400415	0.004646
2018-01-04	98.106	26859408	111.0	1062.2	1.50	19.34	12.22	333836	2440.116581	-0.000972
2018-01-05	98.106	27121167	111.0	1062.7	1.50	19.34	11.92	308770	2441.026331	-0.025804
2018-01-08	98.106	26730598	111.0	1066.0	1.50	19.34	12.31	311429	2442.071914	-0.013039
...	...	...	...	...	...	...	...	...	...	...
2022-12-23	109.280	43902495	90.2	1280.8	3.25	13.31	17.43	366989	2371.032672	-0.003814
2022-12-26	109.280	47542604	90.2	1274.8	3.25	13.31	17.46	427845	2371.172670	-0.005889
2022-12-27	109.280	45718241	90.2	1271.4	3.25	13.31	17.09	448499	2371.097754	-0.057144
2022-12-28	109.280	46984411	90.2	1267.0	3.25	13.31	17.53	405894	2371.001420	0.039391
2022-12-29	109.280	47046503	90.2	1264.5	3.25	13.31	18.40	361194	2370.185836	0.053954

1265 rows × 10 columns

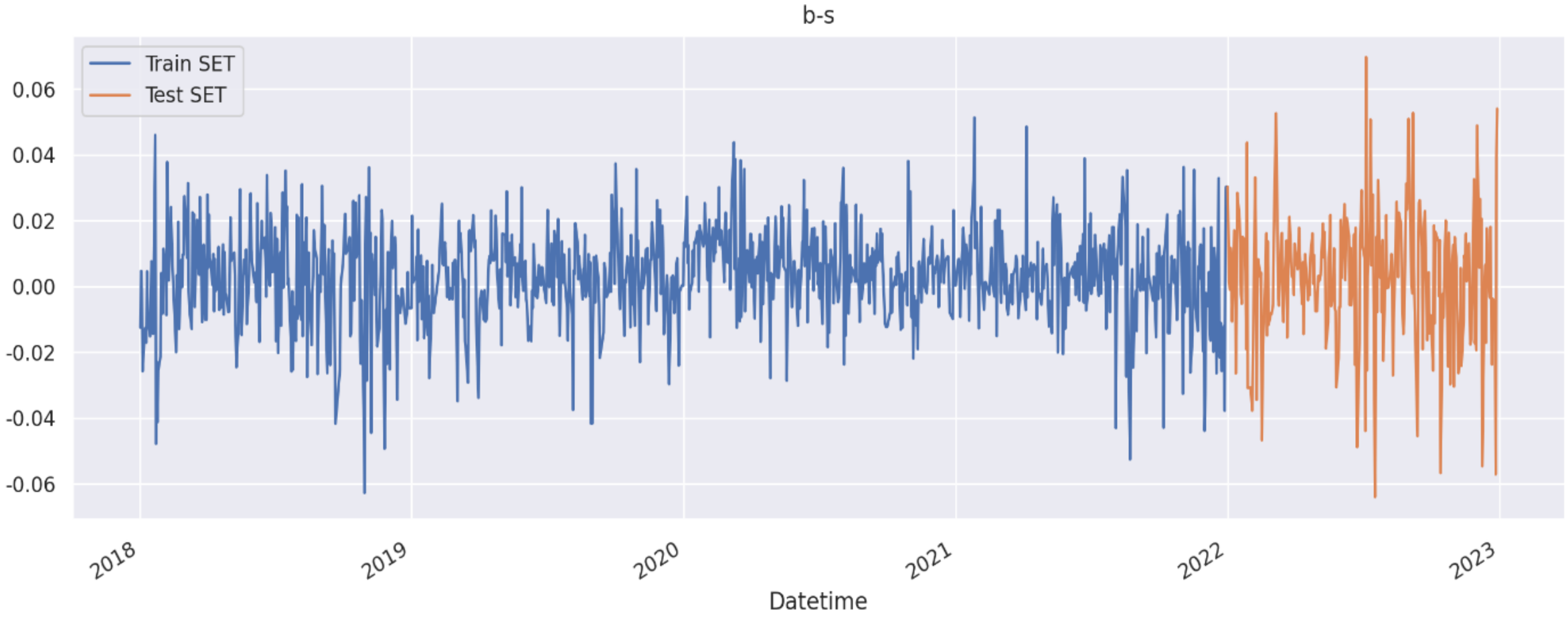
3. EDA

종속변수(Target)



4. 데이터 전처리

2022-01-01을 기준으로 train-test split을 진행 (80%:20%)



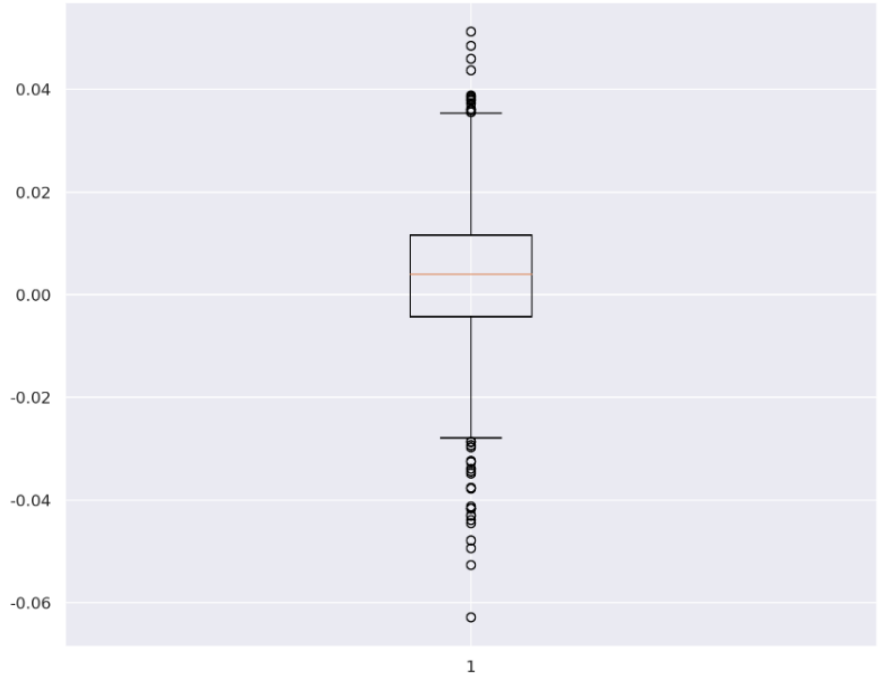
4. 데이터 전처리

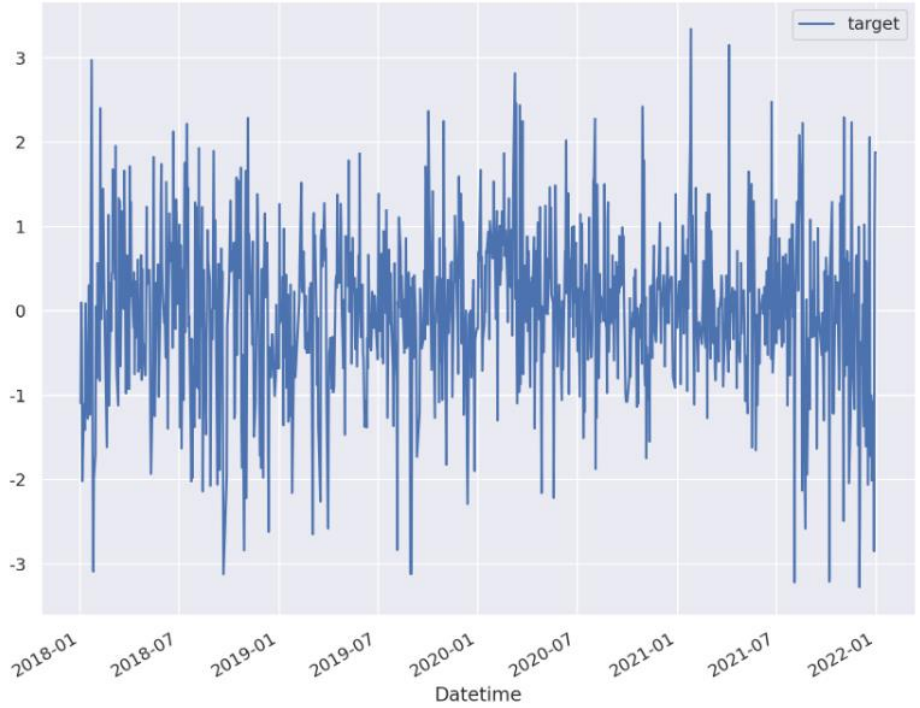
Scaling & Remove Outlier

: standard scaling을 채택해 스케일링 진행 & IQR의 2.5배를 이상치 제거 범위로 설정

```
ss= StandardScaler()

df_train_scaled = pd.DataFrame(ss.fit_transform(df_train), index = df_train.index, columns = df_train.columns)
df_test_scaled = pd.DataFrame(ss.transform(df_test), index = df_test.index, columns = df_test.columns)
```





5. 변수 선택 (variable selection)

Goodness of fit 과 principle of parsimony의 관점을 잘 조화시켜 적절한 변수선택

(1) Best Selection

변수 선택을 위해 사용되는 통계량 5가지 중 AIC와 F-statistic을 통해 모든 경우의 수를 판단하여
각각의 Best Model 설정

(2) Stepwise Selection

한 번 변수를 채택하면 해당 변수를 더 이상 검토하지 않는 전진선택법을 개선한 방법으로,
이미 선택된 변수가 새로운 변수에 의해 중요성을 상실하게 되는 지를 매단계에서 검토

5-(1) AIC

모든 경우의 수를 판단하기 위해 for문을 통해 변수들의 조합을 계산

함수 내 변수

- *combo* : 변수들의 가능한 조합
- *variables* : *combo* 값, 즉 변수 삽입
- *model* : ols 모델
- *aic* : *ols* 모델의 AIC 값

```
def best_lm_aic(df):
    result = {'variables':[], 'model':[], 'aic':[]}
    for k in range(1, len(df.columns)-1):

        for combo in itertools.combinations(df.columns[:-1], k):
            lm = ols('target~' + '+'.join(combo), data = df).fit()
            result['variables'].append(combo)
            result['model'].append(lm)
            result['aic'].append(lm.aic)

    min_arg = np.argmin(result['aic'])
    print(f"Best variable selection : {result['variables'][min_arg]}")
    return result['model'][min_arg]
```

5-(1) AIC

전처리한 데이터를 함수에 넣은 결과

```
Best variable selection : ('cons_sent', 'exchange', 'rt_rate', 'vkospi', 'kospi_ma120')
```

OLS Regression Results

Dep. Variable:	target	**R-squared:**	0.025
Model:	OLS	**Adj. R-squared:**	0.021
Method:	Least Squares	**F-statistic:**	5.257
Date:	Mon, 17 Jul 2023	**Prob (F-statistic):**	8.98e-05
Time:	21:05:02	**Log-Likelihood:**	-1424.3
No. Observations:	1013	**AIC:**	2861.
Df Residuals:	1007	**BIC:**	2890.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1.117e-18	0.031	-3.59e-17	1.000	-0.061	0.061
cons_sent	0.1484	0.064	2.315	0.021	0.023	0.274
exchange	0.0743	0.042	1.788	0.074	-0.007	0.156
rt_rate	0.1115	0.040	2.780	0.006	0.033	0.190
vkospi	0.1222	0.048	2.559	0.011	0.029	0.216
kospi_ma120	-0.1220	0.045	-2.741	0.006	-0.209	-0.035

Omnibus:	50.819	**Durbin-Watson:**	1.776
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	89.770
Skew:	-0.370	**Prob(JB):**	3.21e-20
Kurtosis:	4.257	**Cond. No.**	4.01

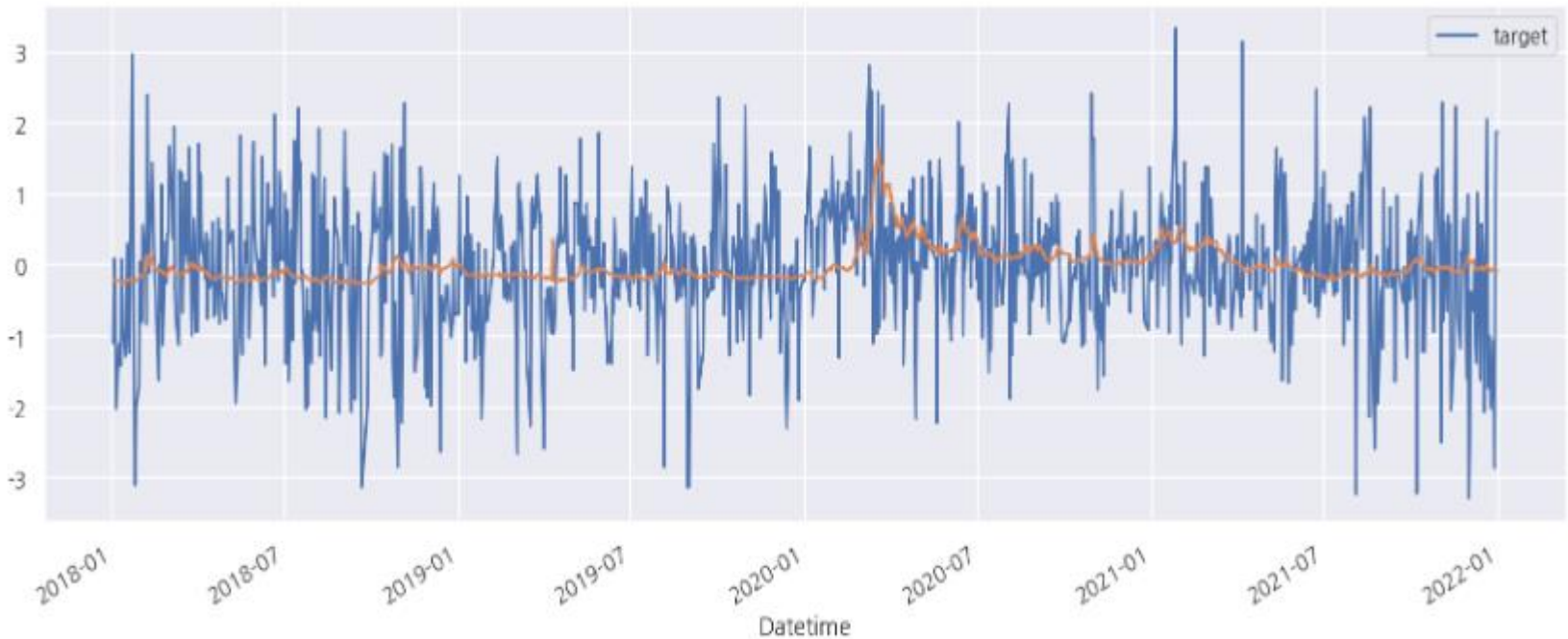
cons_sent, exchange, rt_rate, vkospi, kospi_ma120 총 5개 변수 선택

5-(1) AIC

Train 데이터로 값을 predict 하여 plot 그리기

```
pred = best_lm.predict(X_train_scaled)
```

```
f, ax = plt.subplots(1)
f.set_figheight(5)
f.set_figwidth(15)
_ = y_train_scaled.plot(ax = ax, style = '-')
_ = (2*pred).plot(ax = ax, style = '-')
```



5-(1) F-statistic

모든 경우의 수를 판단하기 위해 for문을 통해 변수들의 조합을 계산

AIC와 과정이 유사

함수 내 변수

- *combo* : 변수들의 가능한 조합
- *variables* : *combo* 값, 즉 변수 삽입
- *model* : ols 모델
- *F-value* : *ols* 모델의 F값

```
def best_lm_F(df):
    result = {'variables':[], 'model':[], 'F-value':[]}
    for k in range(1, len(df.columns)-1):

        for combo in itertools.combinations(df.columns[:-1], k):
            lm = ols('target~' + '+'.join(combo), data = df).fit()
            result['variables'].append(combo)
            result['model'].append(lm)
            result['F-value'].append(lm.fvalue)

    max_arg = np.argmax(result['F-value'])
    print(f"Best variable selection : {result['variables'][max_arg]}")
    return result['model'][max_arg]
```

5-(1) F-statistic

전처리한 데이터를 함수에 넣은 결과

```
Best variable selection : ('vkospi',)
```

OLS Regression Results

Dep. Variable:	target	**R-squared:**	0.014
Model:	OLS	**Adj. R-squared:**	0.013
Method:	Least Squares	**F-statistic:**	14.83
Date:	Mon, 17 Jul 2023	**Prob (F-statistic):**	0.000125
Time:	21:05:08	**Log-Likelihood:**	-1430.0
No. Observations:	1013	**AIC:**	2864.
Df Residuals:	1011	**BIC:**	2874.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1.117e-18	0.031	-3.58e-17	1.000	-0.061	0.061
vkospi	0.1202	0.031	3.851	0.000	0.059	0.182

Omnibus:	65.631	**Durbin-Watson:**	1.761
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	117.756
Skew:	-0.457	**Prob(JB):**	2.69e-26
Kurtosis:	4.398	**Cond. No.**	1.00

F-statistic 에 따르면 *vkospi* 변수만 채택하지만, p-value == 0 이라는 점에서 해당 모델 선택X

5-(2) Stepwise Selection

새로운 변수가 들어올 때마다 기존의 변수가 계속 유의하게 남아 있을 수 있는 지를
검토하기 위한 함수 정의 (코드 생략)

선택된 변수로 식을 만들어 ols 모델에 적합시키면 다음과 같은 결과가 나온다.

OLS Regression Results

Dep. Variable:	target	**R-squared:**	0.014
Model:	OLS	**Adj. R-squared:**	0.013
Method:	Least Squares	**F-statistic:**	14.83
Date:	Mon, 17 Jul 2023	**Prob (F-statistic):**	0.000125
Time:	21:09:48	**Log-Likelihood:**	-1430.0
No. Observations:	1013	**AIC:**	2864.
Df Residuals:	1011	**BIC:**	2874.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1.117e-18	0.031	-3.58e-17	1.000	-0.061	0.061
vkospi	0.1202	0.031	3.851	0.000	0.059	0.182

Omnibus:	65.631	**Durbin-Watson:**	1.761
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	117.756
Skew:	-0.457	**Prob(JB):**	2.69e-26
Kurtosis:	4.398	**Cond. No.**	1.00

F-statistic 에 따르면 *vkospi* 변수만 채택하지만, p-value == 0 이라는 점에서 해당 모델 선택X

5. 변수 선택 (variable selection)

(1) Best Selection

- AIC : cons_sent, exchange, rt_rate, vkospi, kospi_ma120 총 5개 변수 선택
- F-statistic : vkospi 만 선택

(2) Stepwise Selection : vkospi 만 선택

AIC 의 Best Model을 선택

변동성(vkospi), 상장주식 회전율(rt_rate), KOSPI MA120(kospi_ma120), 소비자 심리지수(cons_sent), 환율(exchange)

* F-statistic과 단계적 선택법(SS)은 vkospi만을 변수로 채택

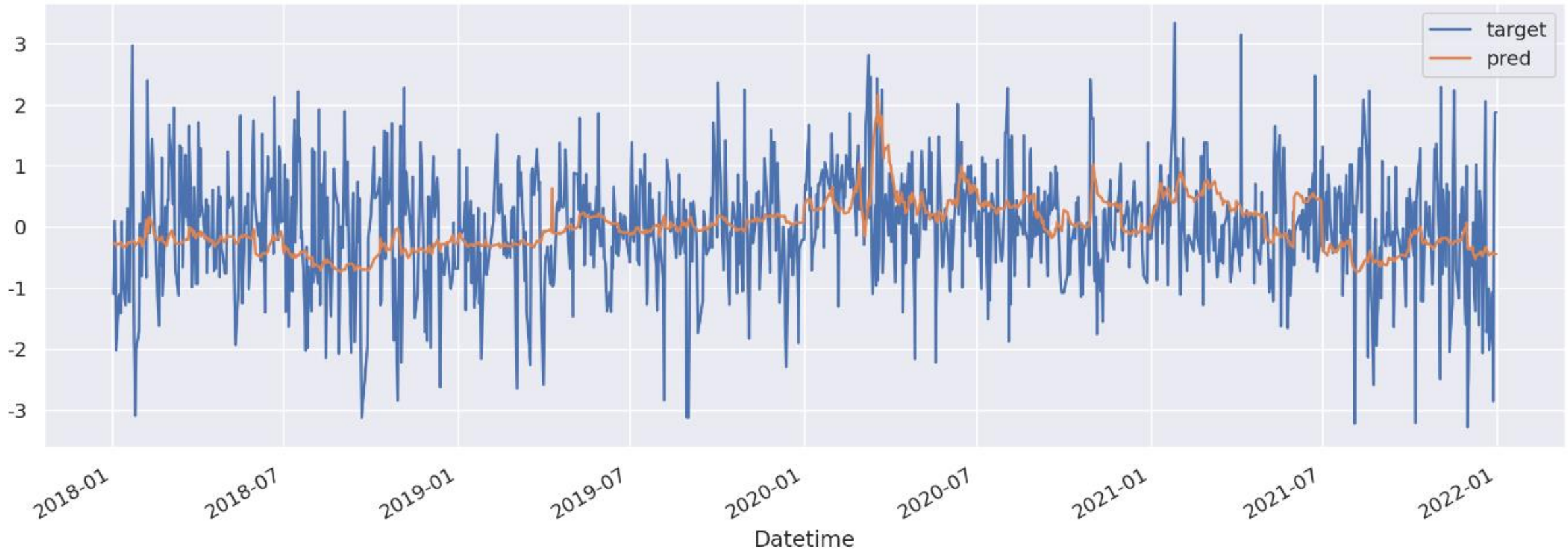
→ 우리가 지향하는 다변수 심리지수 구축과 방향성이 맞지 않음

→ p-value == 0 문제점

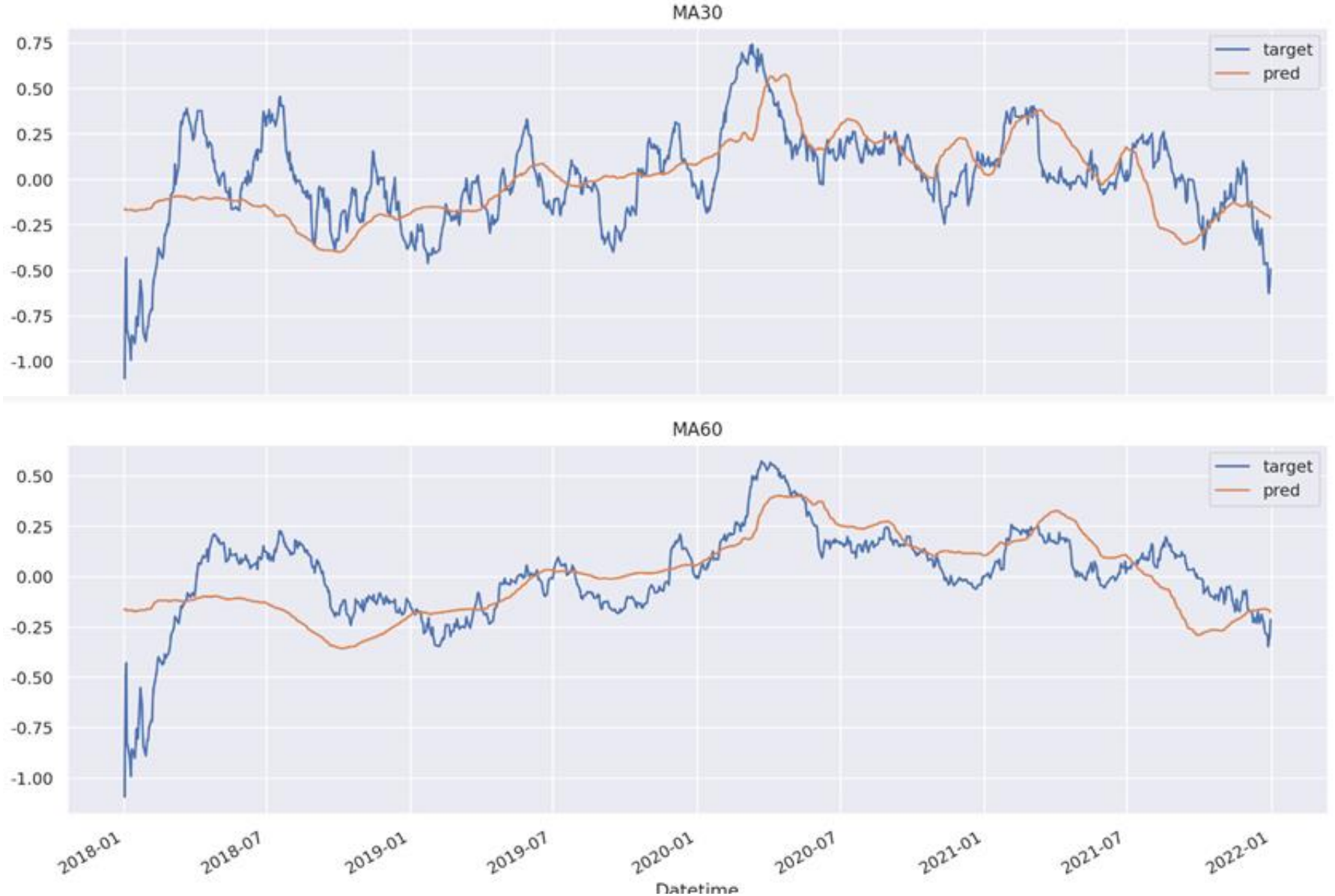
6-1. OLS 모델

선택된 5개의 변수에 대해 ols 메서드로 train set을 학습

```
ols_model = ols('target~vkospi+rt_rate+kospi_ma120+cons_sent+exchange', data = df_train_scaled).fit()
```



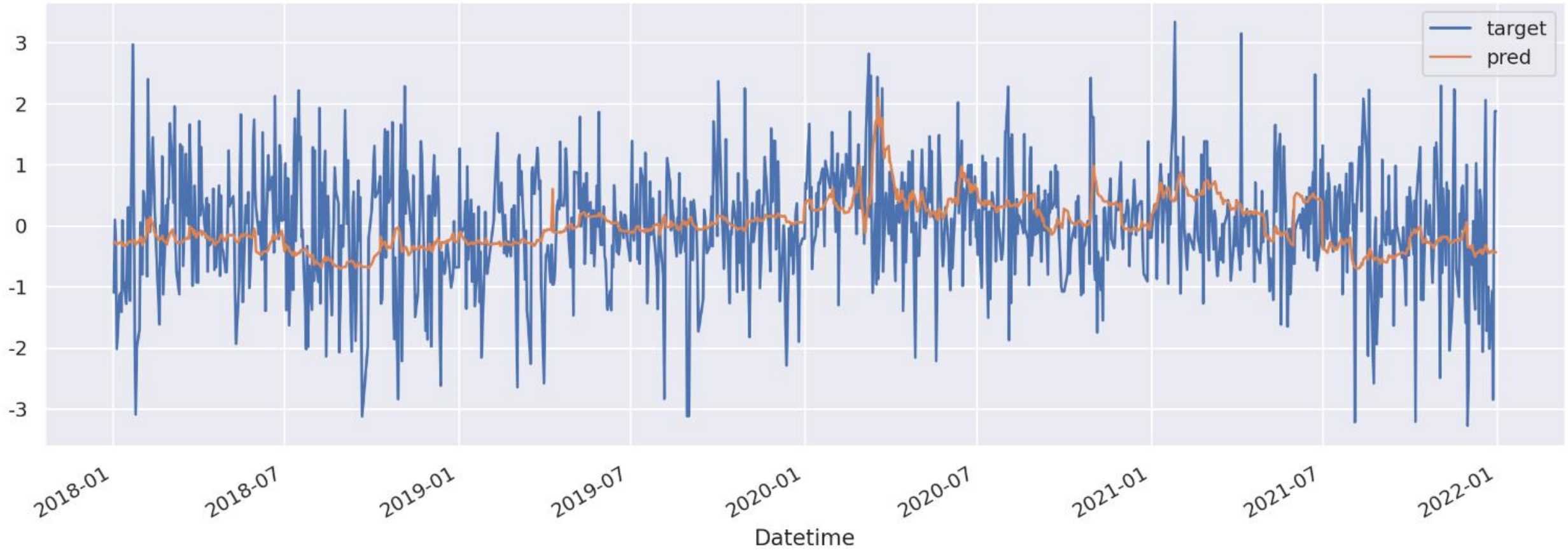
6-1. OLS 모델


MA30
target
pred
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
MA60
target
pred
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07
2021-01
2021-07
2022-01
Datetime

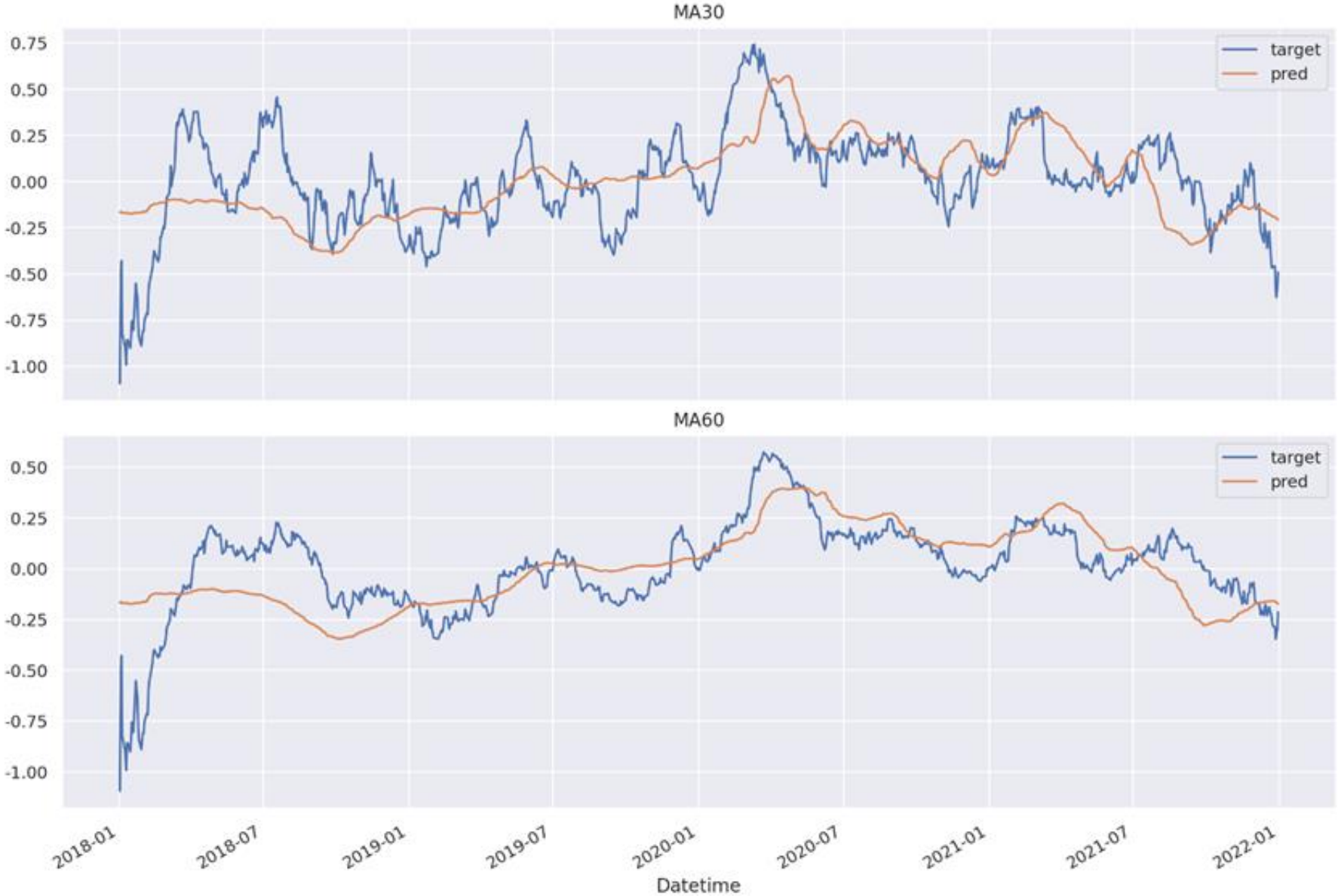
6-2. Ridge 모델

fit_regularized 메서드를 이용한 Ridge 정규화 진행 (잔차의 제곱합, 변수의 제곱합을 최소화)

```
ridge = ols('target~vkospi+rt_rate+kospi_ma120+cons_sent+exchange', data = df_train_scaled).fit_regularized(alpha = 0.01, L1_wt = 0)
```



6-2. Ridge 모델

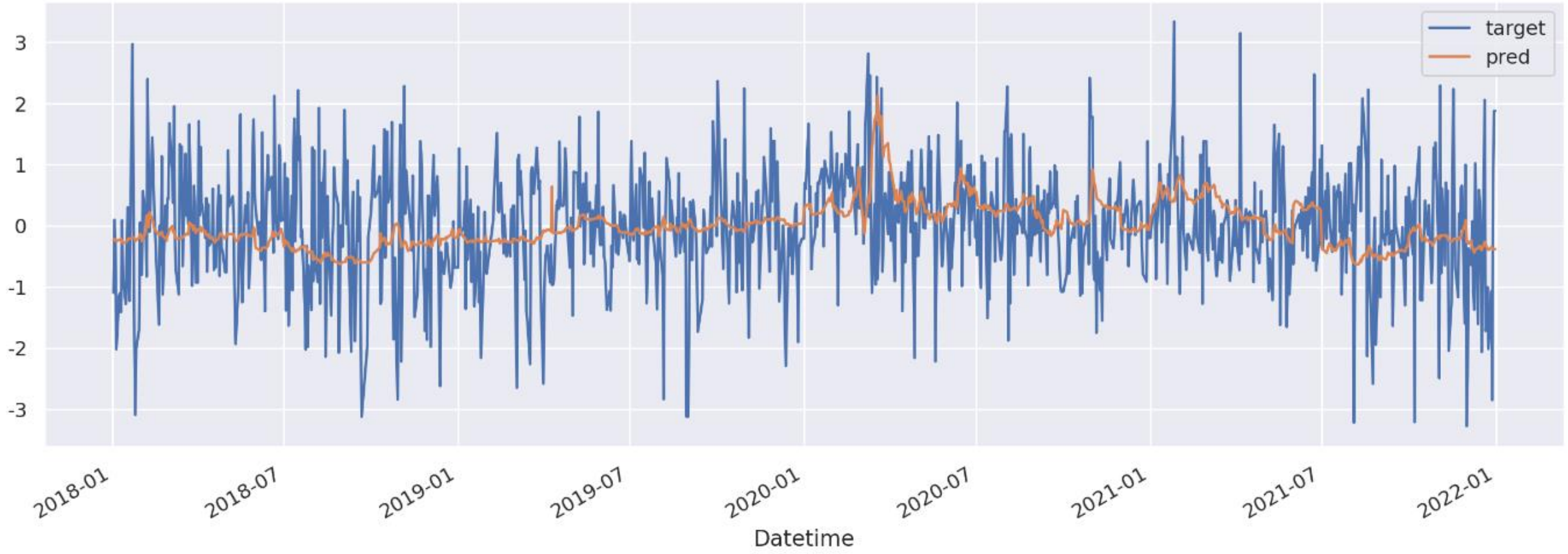

MA30
target
pred
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
MA60
target
pred
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07
2021-01
2021-07
2022-01
Datetime

6-2. Ridge 모델

Ridge 모델의 하이퍼파라미터인 alpha와 fit_intercept을 튜닝하기 위해 그리드 서치(Grid Search) 수행

```
grid_search.fit(X_train_scaled, y_train_scaled)
grid_search.best_params_
```

하이퍼파라미터 튜닝 후 모델 최적화 (alpha=10, fit_intercept=False)

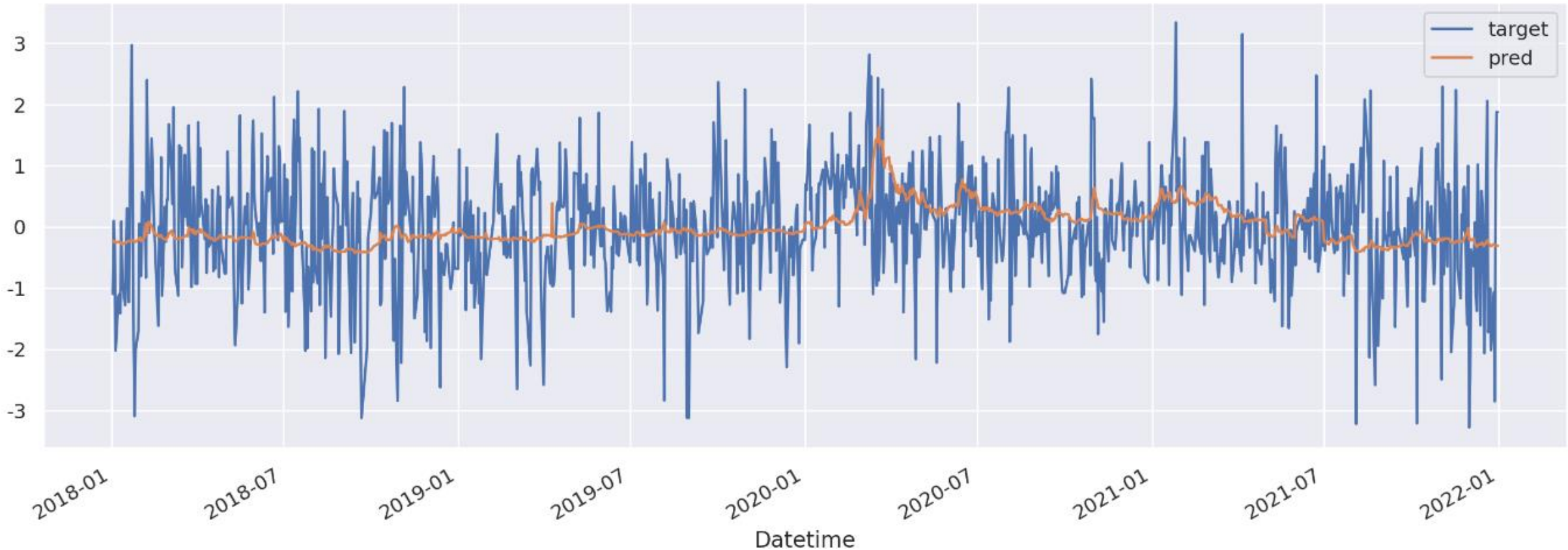


6-3. Lasso 모델

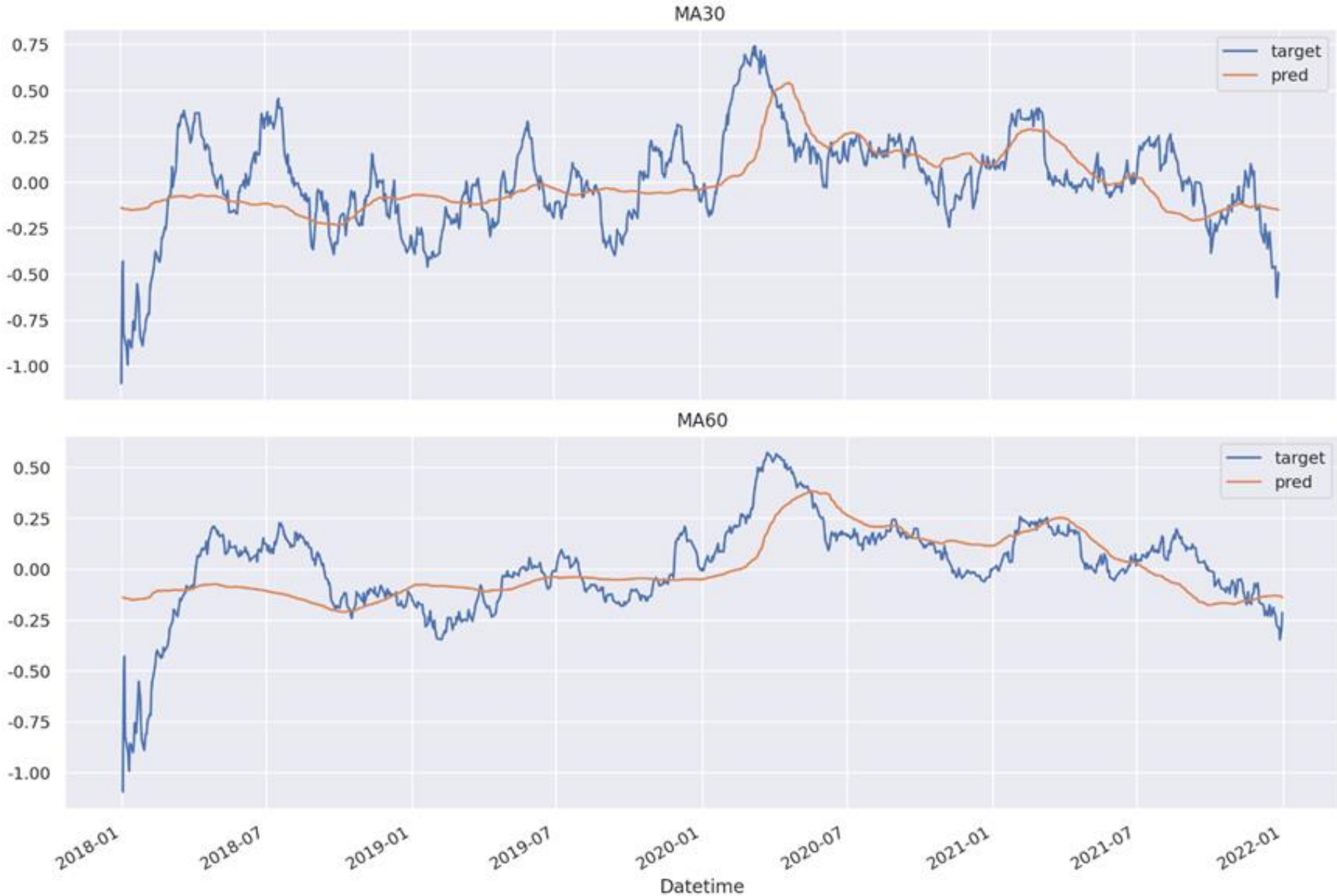
fit_regularized 메서드를 이용한 Lasso 정규화 진행 (잔차의 제곱합, 변수의 절댓값 합을 최소화)

```
lasso = ols('target~vkospi+rt_rate+kospi_ma120+cons_sent+exchange', data = df_train_scaled).fit_regularized(alpha = 0.01, L1_wt = 1)
```

하이퍼파라미터 튜닝 후 모델 최적화 (alpha=0.01, fit_intercept=False)



6-3. Lasso 모델

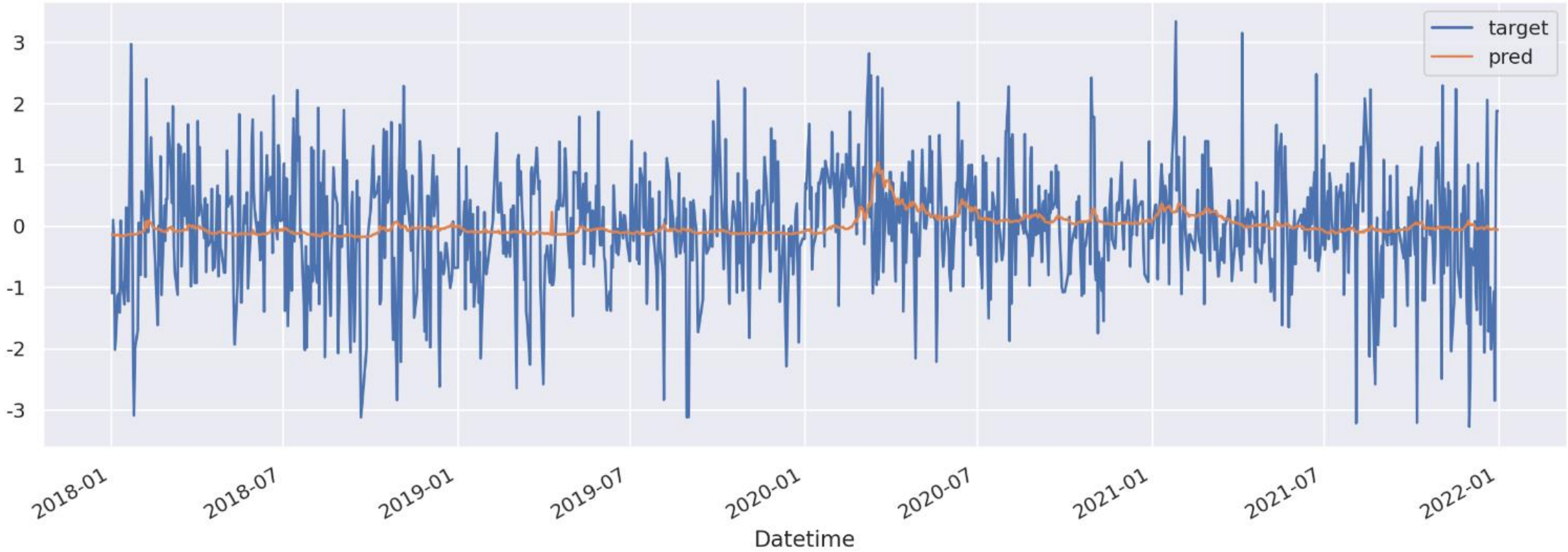

MA30
target
pred
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
MA60
target
pred
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07
2021-01
2021-07
2022-01
Datetime

6-4. Elastic Net 모델

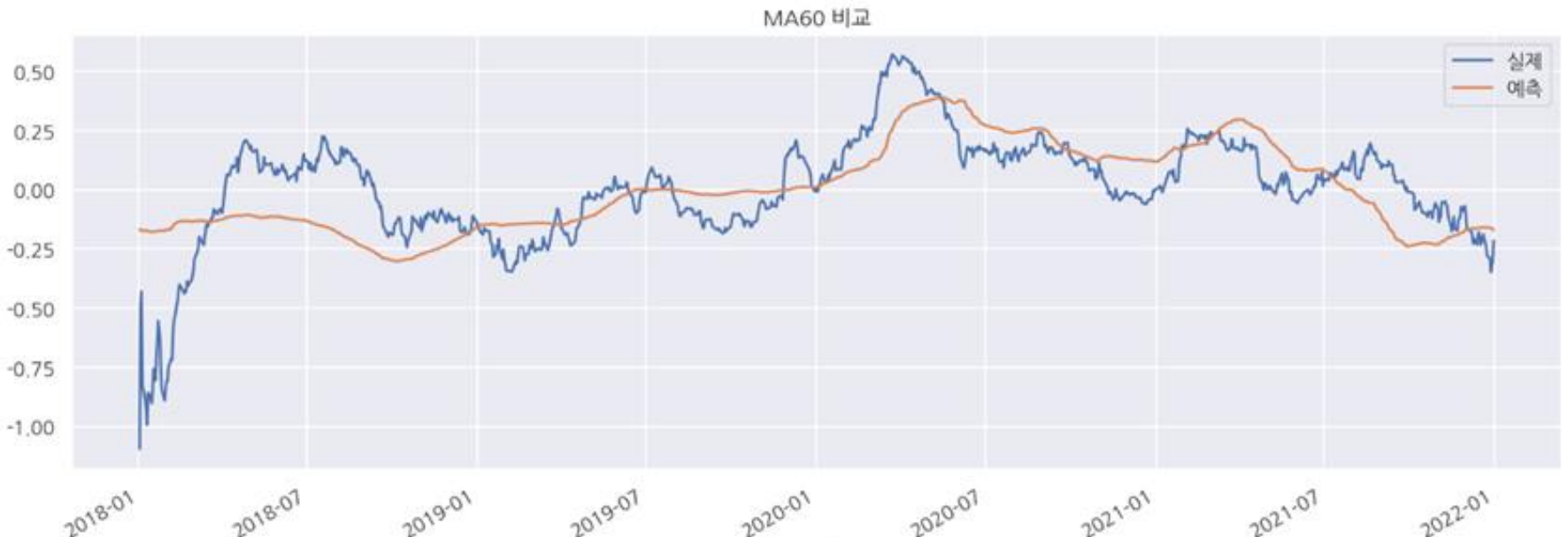
Elastic Net은 Ridge와 LASSO를 합쳐 놓은 형태

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \quad s.t. \quad \alpha|\boldsymbol{\beta}\| + (1-\alpha)\|\boldsymbol{\beta}\|^2 < t$$

하이퍼파라미터 튜닝 후 모델 최적화 (alpha=0.1, fit_intercept=False)



6-4. Elastic Net 모델



6-5. 네 가지 모델 결과 비교

〈OLS VS Ridge VS Lasso VS Elastic Net〉모델에 대해 결과 비교