[모델 분석]

총 4가지 모델 및 백테스트를 진행하기에 앞서 모델 성능 및 영향력 있는 parameter를 확인해보기 위해 1년의 data를 기반으로 실험을 진행하였다.

최종적인 결과와 모델 평가는 5년의 data로 진행할 예정이며, 성능을 판단하기 위해 과거 데이터로 예측을 한 후, 실제 수익률과 비교를 통해 실질적인 성능을 판단 및 분석한다.

[데이터]

```
 0   day_of_week          17712 non-null  int64
 1   month                17712 non-null  int64
 2   ascent_1             17712 non-null  float64
 3   ascent_5             17712 non-null  float64
 4   ascent_10            17712 non-null  float64
 5   ascent_20            17712 non-null  float64
 6   3month_ascent_rate   17712 non-null  float64
 7   kospi_rate           17712 non-null  float64
 8   WTI_rate             17712 non-null  float64
 9   gold_rate            17712 non-null  float64
 10  cooper_rate          17712 non-null  float64
 11  dollar_ascent_1      17712 non-null  float64
 12  dollar_ascent_5      17712 non-null  float64
 13  dollar_ascent_10     17712 non-null  float64
 14  dollar_ascent_20     17712 non-null  float64
 15  dollar_ascent_65     17712 non-null  float64
 16  volume_5             17712 non-null  float64
 17  volume_20            17712 non-null  float64
 18  volume_65            17712 non-null  float64
```

Data frame은 위와 같은 feature 들로 구성되어 있다.

하지만, 전반적인 실험이 끝나고 나면 사용할 5년치 데이터에 있어서는 각 종목에 따른 가장 영향력을 많이 주는 시장 데이터 (환율, 유가, 금, 구리) 중 하나를 선택할 것이며, 최대한 feature 들간의 독립성을 유지할 정이다

➔ 현재 1,5,10,20일간의 수익률, 거래량이 모두 있으나, 일부 feature 삭제 예정

1) Linear regression

- Train data, Test data split result

```
for train_idx, test_idx in cv.split(X=data):
    train = data.iloc[train_idx]
    train_dates = train.index.get_level_values('date')
    test = data.iloc[test_idx]
    test_dates = test.index.get_level_values('date')
    df = train.reset_index().append(test.reset_index())
    n = len(df)
    assert n== len(df.drop_duplicates())
    print(train.groupby(level='code').size().value_counts().index[0],
          train_dates.min().date(), train_dates.max().date(),
          test.groupby(level='code').size().value_counts().index[0],
          test_dates.min().date(), test_dates.max().date())
```

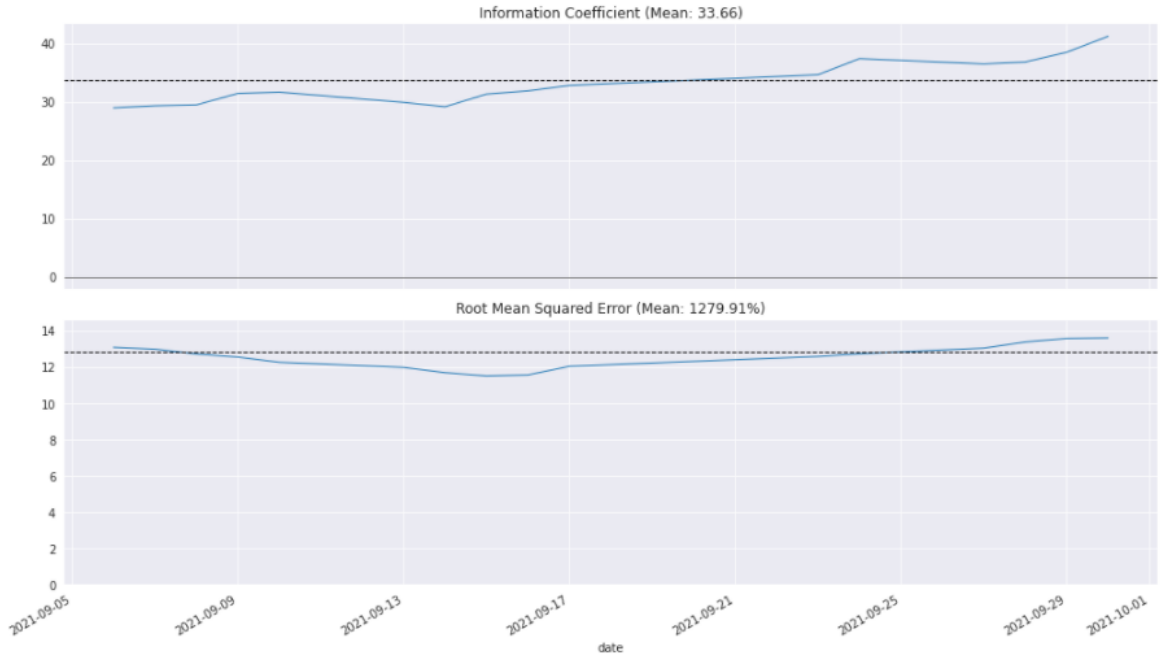
```
225 2020-10-21 2021-09-13 10 2021-09-14 2021-09-30
225 2020-10-06 2021-08-30 10 2021-08-31 2021-09-13
```

- Model 구조

```
lr_predictions, lr_scores = [], []
lr = LinearRegression()
for i, (train_idx, test_idx) in enumerate(cv.split(X), 1):
    X_train, y_train, = X.iloc[train_idx], y.iloc[train_idx]
    X_test, y_test = X.iloc[test_idx], y.iloc[test_idx]
    lr.fit(X=X_train, y=y_train)
    y_pred = lr.predict(X_test)

    preds = y_test.to_frame('actuals').assign(predicted=y_pred)
    preds_by_day = preds.groupby(level='date')
    scores = pd.concat([preds_by_day.apply(lambda x: spearmanr(x.predicted,
                                                               x.actuals)[0] * 100)
                        .to_frame('ic'),
                        preds_by_day.apply(lambda x: np.sqrt(mean_squared_error(y_pred=x.predicted,
                                                                                y_true=x.actuals)))
                        .to_frame('rmse')], axis=1)

    lr_scores.append(scores)
    lr_predictions.append(preds)

lr_scores = pd.concat(lr_scores)
lr_predictions = pd.concat(lr_predictions)
```

: parameter 수정 없이 진행 (default값 사용)

: 성능 판단은 spearman 상관계수를 통해 학습 성능 판단 및 RMSE를 통해 예측의 정확도를 기준으로 한다

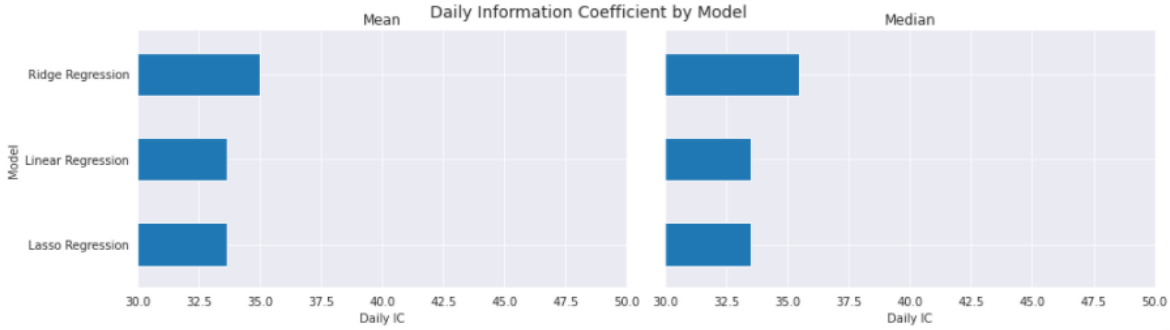


: Information Coefficient (IC) 의 값과 RMSE 모두 현재 학습이 잘 되고 있지 않음을 보임

: 현재 데이터의 양이 현저히 적으므로 보다 데이터의 양을 늘린 후, 다시 실험 예정

(train length를 60->225로 늘렸을 때, 약 정확도 20% 상승)

➔ 과대 적합 방지를 위해 Linear regression 뿐만 아니라, Ridge regression, Lasso regression 을 사용하였으나, 별다른 성능의 차이를 보이지 않음 (재실험 예정)



2) Logistic regression (Linear classification)

- Train data, Test data split result

```
for train_idx, test_idx in cv.split(X=data):
    train = data.iloc[train_idx]
    train_dates = train.index.get_level_values('date')
    test = data.iloc[test_idx]
    test_dates = test.index.get_level_values('date')
    df = train.reset_index().append(test.reset_index())
    n = len(df)
    assert n== len(df.drop_duplicates())
    print(train.groupby(level='code').size().value_counts().index[0],
          train_dates.min().date(), train_dates.max().date(),
          test.groupby(level='code').size().value_counts().index[0],
          test_dates.min().date(), test_dates.max().date())
```

```
225 2020-10-21 2021-09-13 10 2021-09-14 2021-09-30
225 2020-10-06 2021-08-30 10 2021-08-31 2021-09-13
```

: 금융 데이터의 시계열 특성을 고려하기 위해 Multiple TimeSeriesCV사용

- Model 구조

```
model = LogisticRegression(C=C,
                           fit_intercept=True,
                           random_state=42,
                           n_jobs=-1)

pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('model', model)])
```

: 분류를 위한 Logistic Regression 사용, C (regularization strength) 로

```
Cs = np.logspace(-5, 5, 11)
```

총 11개의 값으로 학습 실행 및 결과 확인

```
predict=                    day_of_week  month  ...  volume_20  volume_65
code   date                                    ...
80     2021-09-14             1      9  ...  12.289691   2.824913
       2021-09-16             3      9  ... -49.981243 -54.328184
       2021-09-17             4      9  ... -14.749955 -24.140101
       2021-09-27             0      9  ... -45.359936 -50.753797
       2021-09-28             1      9  ... -38.782208 -44.580708
...                         ...    ...  ...        ...        ...
330590 2021-09-17             4      9  ... -35.004139 -38.826570
       2021-09-27             0      9  ... -11.584788 -20.870437
       2021-09-28             1      9  ... -23.602542 -30.250591
       2021-09-29             2      9  ...  17.044070   8.028005
       2021-09-30             3      9  ... -27.988303 -32.622725

[492 rows x 18 columns]
predict=                   day_of_week  month  ...   volume_20   volume_65
code  date                                    ...
14820 2021-09-07             1      9  ...  183.842947  472.107678
      2021-09-08             2      9  ...   20.960429  155.691332
      2021-09-09             3      9  ...   35.497260  192.133180
      2021-09-10             4      9  ...  -31.796816   50.574564
      2021-09-13             0      9  ...   87.284081  317.432602

[5 rows x 18 columns]
```

: predict한 결과를 확인해봤을 때,

(C= 1e-05 / train set에서는 약 492일의 상승 예상, test set에서는 5일의 상승 예상 날짜)

➔ 해석: 종목 코드 80인 기업 같은 경우에는 결과값으로 보이는 9/14, 9/16, 9/17, 9/27, 9/28에 살 경우 3개월 이후 수익률이 상승하는 추세로 볼 수 있다.

```
predict=                   day_of_week  month  ...  volume_20  volume_65
code   date                                    ...
80     2021-09-28            1      9  ... -38.782208 -44.580708
       2021-09-29            2      9  ... -38.301201 -42.781212
670    2021-09-28            1      9  ... -63.323972 -56.366752
       2021-09-29            2      9  ... -42.851913 -31.294479
810    2021-09-28            1      9  ...  32.065325  24.878543
...                        ...    ...  ...        ...        ...
195990 2021-09-29            2      9  ... -69.552034 -80.369187
       2021-09-30            3      9  ... -68.658912 -79.421502
215200 2021-09-28            1      9  ... -28.670952 -34.540395
       2021-09-29            2      9  ... -12.435519 -22.946840
263720 2021-09-29            2      9  ...  24.766329 -43.456126

[129 rows x 18 columns]
predict=                   day_of_week  month  ...   volume_20   volume_65
code   date                                    ...
12510  2021-09-07            1      9  ...   68.413843   48.728234
       2021-09-08            2      9  ...  107.776528   88.147114
       2021-09-09            3      9  ...   43.114369   33.681407
       2021-09-13            0      9  ...  110.071579  108.613590
14820  2021-08-31            1      8  ...   28.693855  128.005243
       2021-09-01            2      9  ...   62.745089  197.329371
       2021-09-02            3      9  ...  -21.251336   50.013397
       2021-09-03            4      9  ...  -40.741891   15.251139
       2021-09-06            0      9  ...   12.874176  122.173548
       2021-09-07            1      9  ...  183.842947  472.107678
       2021-09-08            2      9  ...   20.960429  155.691332
       2021-09-09            3      9  ...   35.497260  192.133180
       2021-09-10            4      9  ...  -31.796816   50.574564
       2021-09-13            0      9  ...   87.284081  317.432602
41190  2021-09-07            1      9  ...   53.681858  158.130357
45660  2021-09-09            3      9  ...  104.585520   76.515871
272210 2021-08-31            1      8  ...   17.228526  119.190922

[17 rows x 18 columns]
```

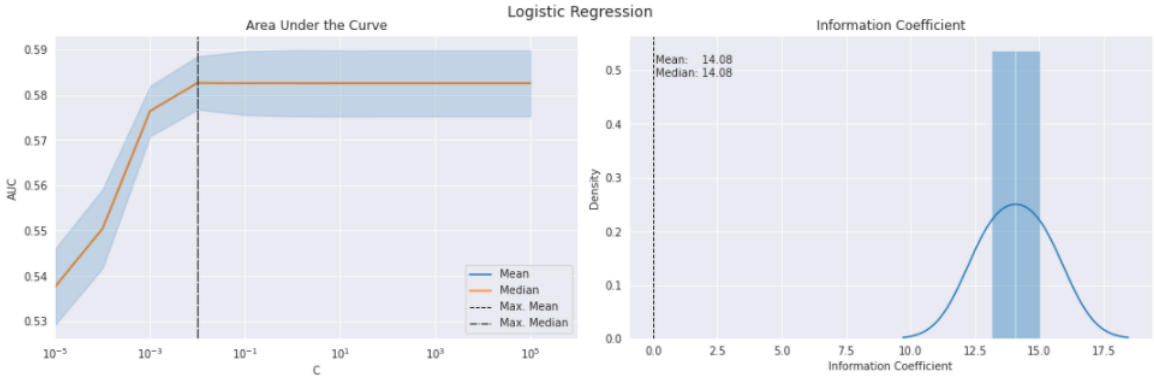
: predict한 결과를 확인해봤을 때,

(C=100000.0/train set에서는 약 129일의 상승 예상, test set에서는 17일의 상승 예상 날짜)

⇨ Regularization parameter값에 따라서 예측 결과 달라짐

⇨ 아직 1년치 데이터로 실험해 본 결과이므로 5년치 데이터로 추가적 실험 후 parameter값 수정 예정

- Parameter 값에 따른 Accuracy 및 Information Coefficient



: Regularization parameter 값이 1e-02부터는 별다른 차이가 없음을 확인

: 대략 최고 성능 Accuracy 57%

: information coefficient (IC)는 -1~1 사이의 값일 때, 유의미한 결과를 가진다. 보통 양수에 가까울수록 정확한 학습을 하고 있다는 판단의 기준이 된다. (Spearman 상관계수를 통해 알파 값이 제대로 학습되고 있는지에 따른 지표라고 생각될 수 있다.)

➔ 데이터 수를 늘리거나, 다른 parameter 조정을 통해 오차 값을 줄여

IC의 범위를 맞춰야지 판단 기준으로 사용할 수 있을 것을 판단된다.

3) Random forest regression

- Train data, Test data split result

```
60 2021-06-25 2021-09-17 6 2021-09-23 2021-09-30
60 2021-06-17 2021-09-09 6 2021-09-10 2021-09-17
60 2021-06-09 2021-09-01 6 2021-09-02 2021-09-09
60 2021-06-01 2021-08-24 6 2021-08-25 2021-09-01
60 2021-05-24 2021-08-13 6 2021-08-17 2021-08-24
60 2021-05-13 2021-08-05 6 2021-08-06 2021-08-13
60 2021-05-04 2021-07-28 6 2021-07-29 2021-08-05
60 2021-04-26 2021-07-20 6 2021-07-21 2021-07-28
60 2021-04-16 2021-07-12 6 2021-07-13 2021-07-20
60 2021-04-08 2021-07-02 6 2021-07-05 2021-07-12
```

: train length 60일, test length 6일

- Model 구조

```
rf_reg = RandomForestRegressor(n_estimators=100,
                               max_depth=None,
                               min_samples_split=2,
                               min_samples_leaf=1,
                               min_weight_fraction_leaf=0.0,
                               max_features='auto',
                               max_leaf_nodes=None,
                               min_impurity_decrease=0.0,
                               min_impurity_split=None,
                               bootstrap=True,
                               oob_score=False,
                               n_jobs=-1,
                               random_state=None,
                               verbose=0,
                               warm_start=False)
```

- Grid Search CV를 통한 최적의 parameter 찾기

```
gridsearch_reg = GridSearchCV(estimator=rf_reg,
                      param_grid=param_grid,
                      scoring=ic,
                      n_jobs=-1,
                      cv=cv,
                      refit=True,
                      return_train_score=True,
                      verbose=1)
```

```
gridsearch_reg.best_params_
```

```
{'max_depth': None, 'min_samples_leaf': 25, 'n_estimators': 250}
```

```
f'{gridsearch_reg.best_score_*100:.2f}'
```

```
'38.54'
```

4) Random forest classification

- Model 구조

```
rf_clf = RandomForestClassifier(n_estimators=100,
                                criterion='gini',
                                max_depth=None,
                                min_samples_split=2,
                                min_samples_leaf=1,
                                min_weight_fraction_leaf=0.0,
                                max_features='auto',
                                max_leaf_nodes=None,
                                min_impurity_decrease=0.0,
                                min_impurity_split=None,
                                bootstrap=True,
                                oob_score=True,
                                n_jobs=-1,
                                random_state=42,
                                verbose=1)
```

- Grid Search CV를 통한 최적의 parameter 찾기

```
param_grid = {'n_estimators': [50, 100, 250],
              'max_depth': [5, 15, None],
              'min_samples_leaf': [5, 25, 100]}
```

```
gridsearch_clf = GridSearchCV(estimator=rf_clf,
                              param_grid=param_grid,
                              scoring='roc_auc',
                              n_jobs=-1,
                              cv=cv,
                              refit=True,
                              return_train_score=True,
                              verbose=1)
```

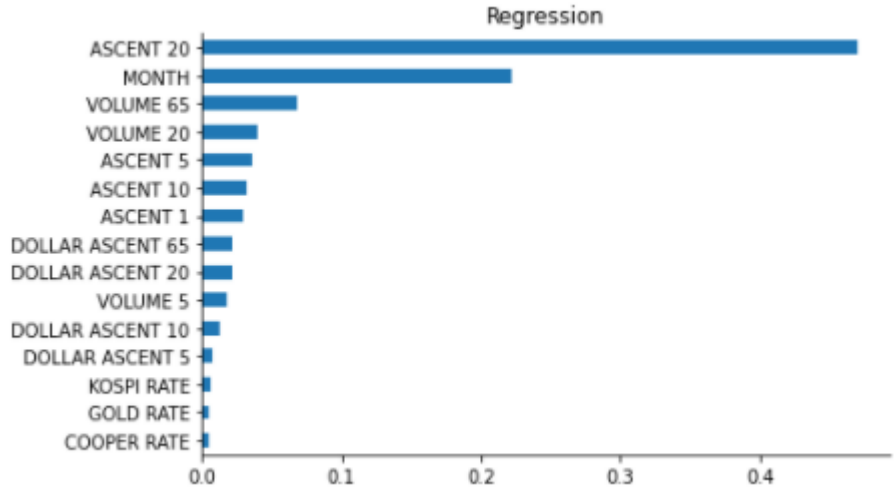
: 총 18가지의 경우에 대해 성능 확인

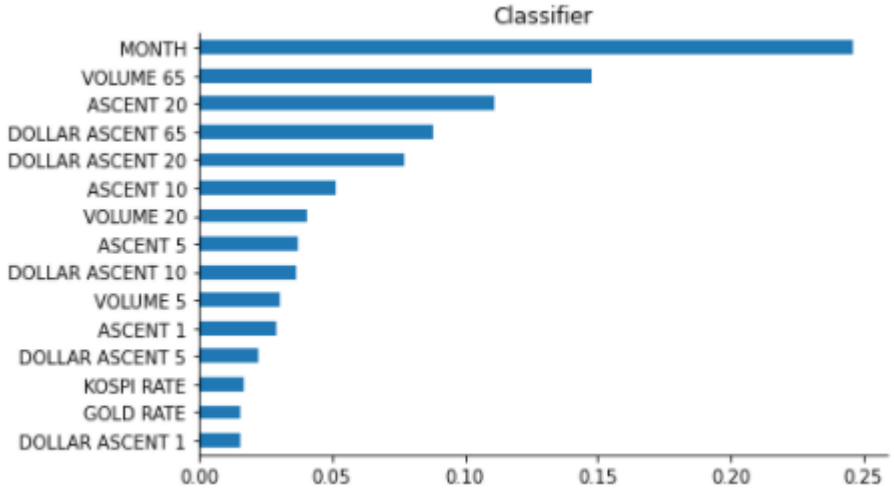
```
gridsearch_clf.best_params_
```

```
{'max_depth': 15, 'min_samples_leaf': 5, 'n_estimators': 250}
```

```
gridsearch_clf.best_score_
```

```
0.701936756751961
```





➔ Linear & Random Forest 모두 classification 이 성능이 좋음