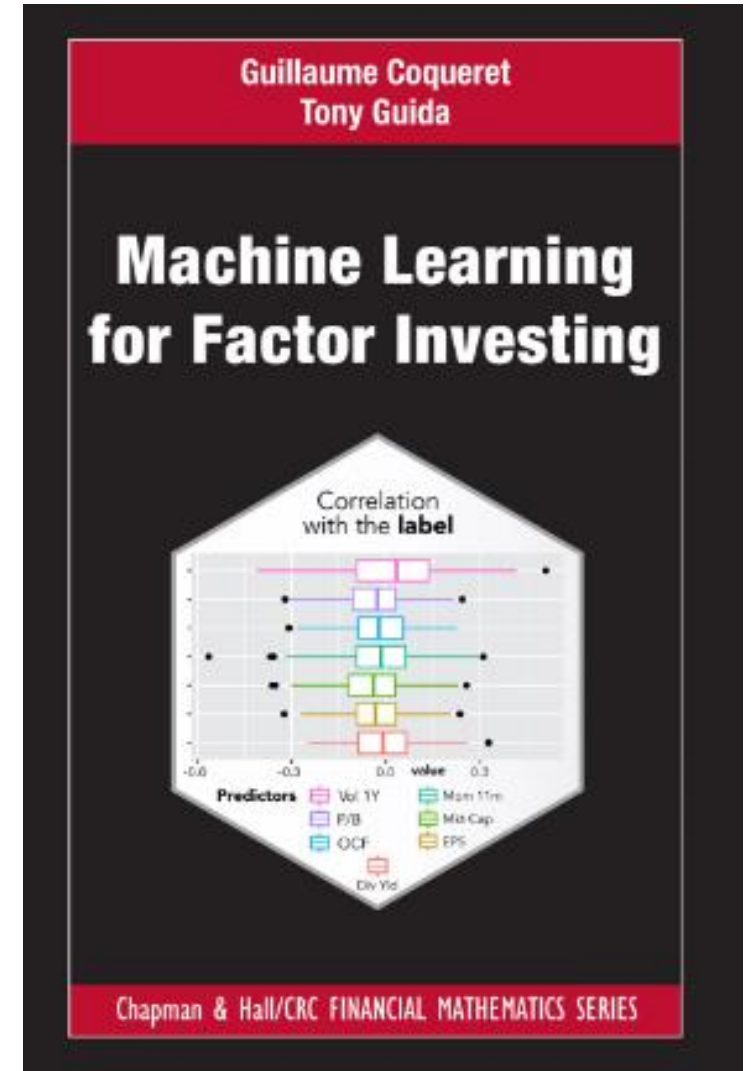


Machine Learning for Factor Investing

https://github.com/donghui-0126/ml_factor



목차

- 왜 Machine Learning을 주제로 했을까요?
 - 기존에 존재하는 Factor model
 - Factor modeling에서 어떻게 Machine Learning을 사용할까?
 - Data/Machine Learning에 대한 간단한 이해
 - 여러 Machine Learning기법에 대한 설명
 - 실제 Machine Learning을 사용한 결과
 - Model explanation
 - Back testing
-

왜 Machine Learning을 주제로 했을까요?


- 재무적인 책에서 배우는 이론 외에서도 머신러닝/딥러닝을 통해서 팩터를 어떻게 찾을 수 있을 지에 대한 통찰을 제공
 - 머신러닝을 통해서 기존에 발견하지 못한 Insight를 발견
 - 머신러닝에 대한 쉬운 가벼운 이해
 - 머신러닝을 사용한 팩터모델링 프레임워크 제공
-

기존에 존재하는 Factor model

Fama-French 5 Factor Model

	date	MKT_RF	SMB	HML	RMW	CMA	RF
716	2023-03-31	0.0251	-0.0694	-0.0885	0.0224	-0.0237	0.0036
717	2023-04-30	0.0061	-0.0256	-0.0004	0.0242	0.0286	0.0035
718	2023-05-31	0.0035	-0.0038	-0.0772	-0.0181	-0.0722	0.0036
719	2023-06-30	0.0646	0.0134	-0.0026	0.0218	-0.0162	0.0040
720	2023-07-31	0.0321	0.0286	0.0413	-0.0056	0.0062	0.0045

Fama-Mecbeth
회귀분석

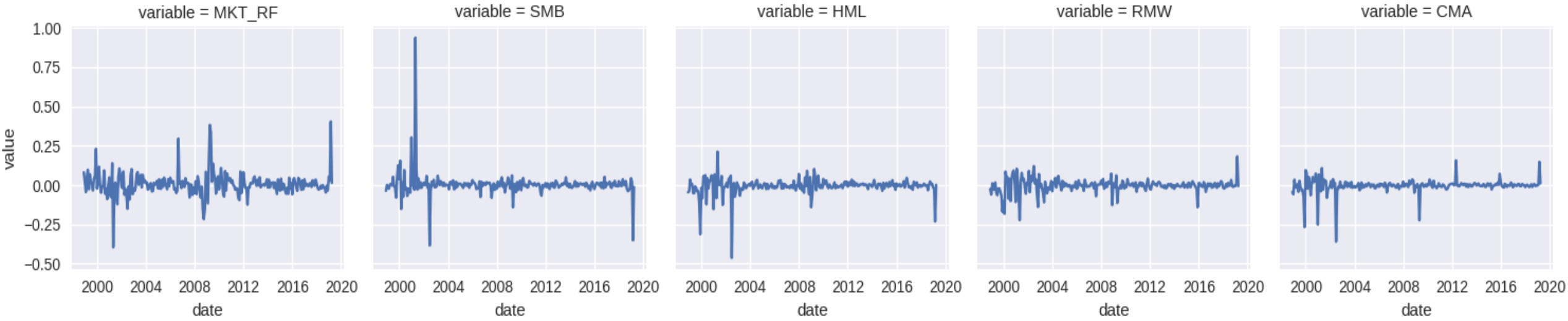


	Intercept	MKT_RF	SMB	HML	RMW	CMA
3	-0.002193	0.856555	0.864555	0.737517	0.334009	-0.282986
4	0.003300	0.296396	0.338474	-0.148201	0.655343	0.422797
7	0.006110	0.370519	0.649548	0.221608	0.260229	0.177670
9	0.003118	0.841591	0.611987	1.030835	0.144572	0.134616
16	0.000433	1.177064	-0.129665	1.347758	0.120328	-0.335390

기존에 존재하는 Factor model

Fama-French 5 Factor Model

	date	MKT_RF	SMB	HML	RMW	CMA	RF
716	2023-03-31	0.0251	-0.0694	-0.0885	0.0224	-0.0237	0.0036



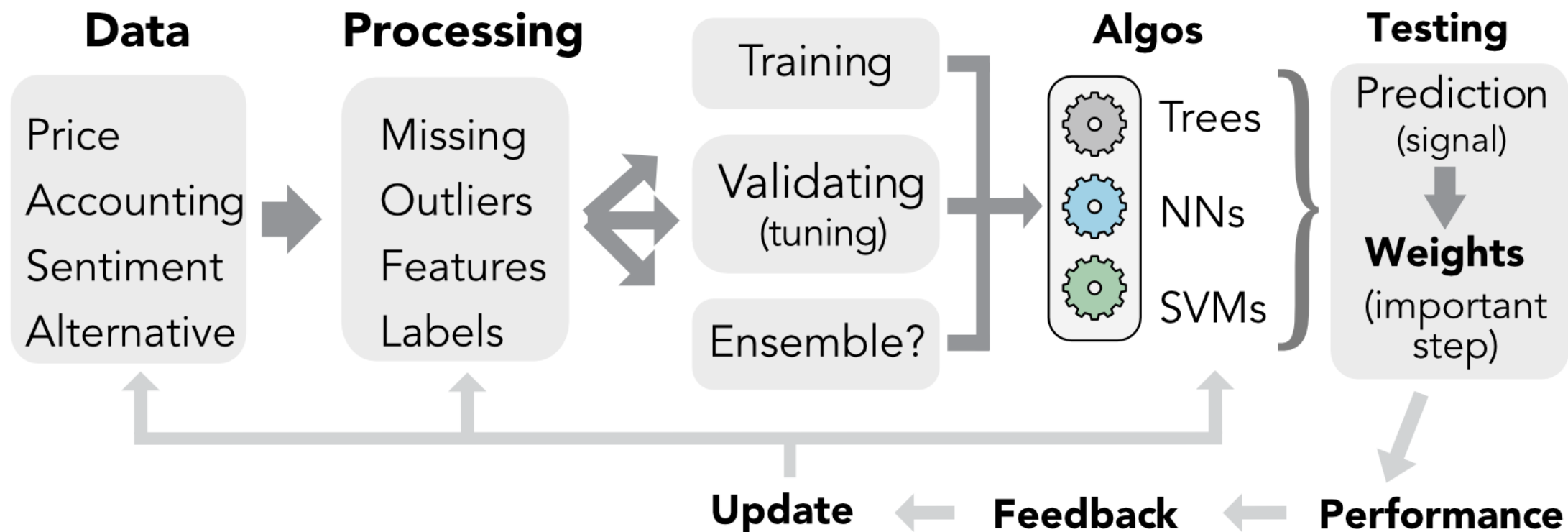
9	0.003118	0.841591	0.611987	1.030835	0.144572	0.134616
16	0.000433	1.177064	-0.129665	1.347758	0.120328	-0.335390

Factor modeling에서 어떻게 Machine Learning을 사용할까요?

- 기존의 모델들과 마찬가지로 데이터를 기반으로 수익률을 예측하는 방법입니다.
- 여러가지 factor를 기반으로 앞으로의 return을 예측합니다.
- 모델을 분석함으로써 새로운 Factor를 발굴할 수 있습니다.
- 모델의 성능이 유의미하다면, 모델을 이해하면서 새로운 통찰을 얻을 수 있습니다.

$$\mathbf{r}_{t+1,n} = f(\mathbf{x}_{t,n}) + \epsilon_{t+1,n},$$

Data/Machine Learning에 대한 간단한 이해



Data/Machine Learning에 대한 간단한 이해

- 93개의 Factor들을 사용해서 주식의 미래 1달/3달/6달/12달 수익률을 예측합니다.

Column Name	Short Description
stock_id	security id
date	date of the data
Advt_12M_Usd	average daily volume in amount in USD over 12 months
Advt_3M_Usd	average daily volume in amount in USD over 3 months
Advt_6M_Usd	average daily volume in amount in USD over 6 months
Asset_Turnover	total sales on average assets
Bb_Yld	buyback yield
Bv	book value
Capex_Ps_Cf	capital expenditure on price to sale cash flow
Capex_Sales	capital expenditure on sales
Cash_Div_Cf	cash dividends cash flow
Cash_Per_Share	cash per share
Cf_Sales	cash flow per share
Debtequity	debt to equity



R1M_Usd	return forward 1 month (LABEL)
R3M_Usd	return forward 3 months (LABEL)
R6M_Usd	return forward 6 months (LABEL)
R12M_Usd	return forward 12 months (LABEL)

Data/Machine Learning에 대한 간단한 이해

- 한달 간의 93개의 Factor들을 사용해서 주식의 미래 1달 수익률을 예측합니다.

Column Name	Short Description		
stock_id	security id		
date	date of the data		
Advt_12M_Usd	average daily volume in amount in USD over 12 months		
Advt_3M_Usd	average daily volume in amount in USD		
Advt_6M_Usd			
Asset_Turnover			
Bb_Yld			
Bv	book value		
Capex_Ps_Cf	capital expenditure on price to sale cash flow		
Capex_Sales	capital expenditure on sales		
Cash_Div_Cf	cash dividends cash flow		
Cash_Per_Share	cash per share		
Cf_Sales	cash flow per share		
DebtEquity	debt to equity		

한달 간의 데이터

R1M_Usd

R3M_Usd

R6M_Usd

R12M_Usd

return forward 1 month (**LABEL**)

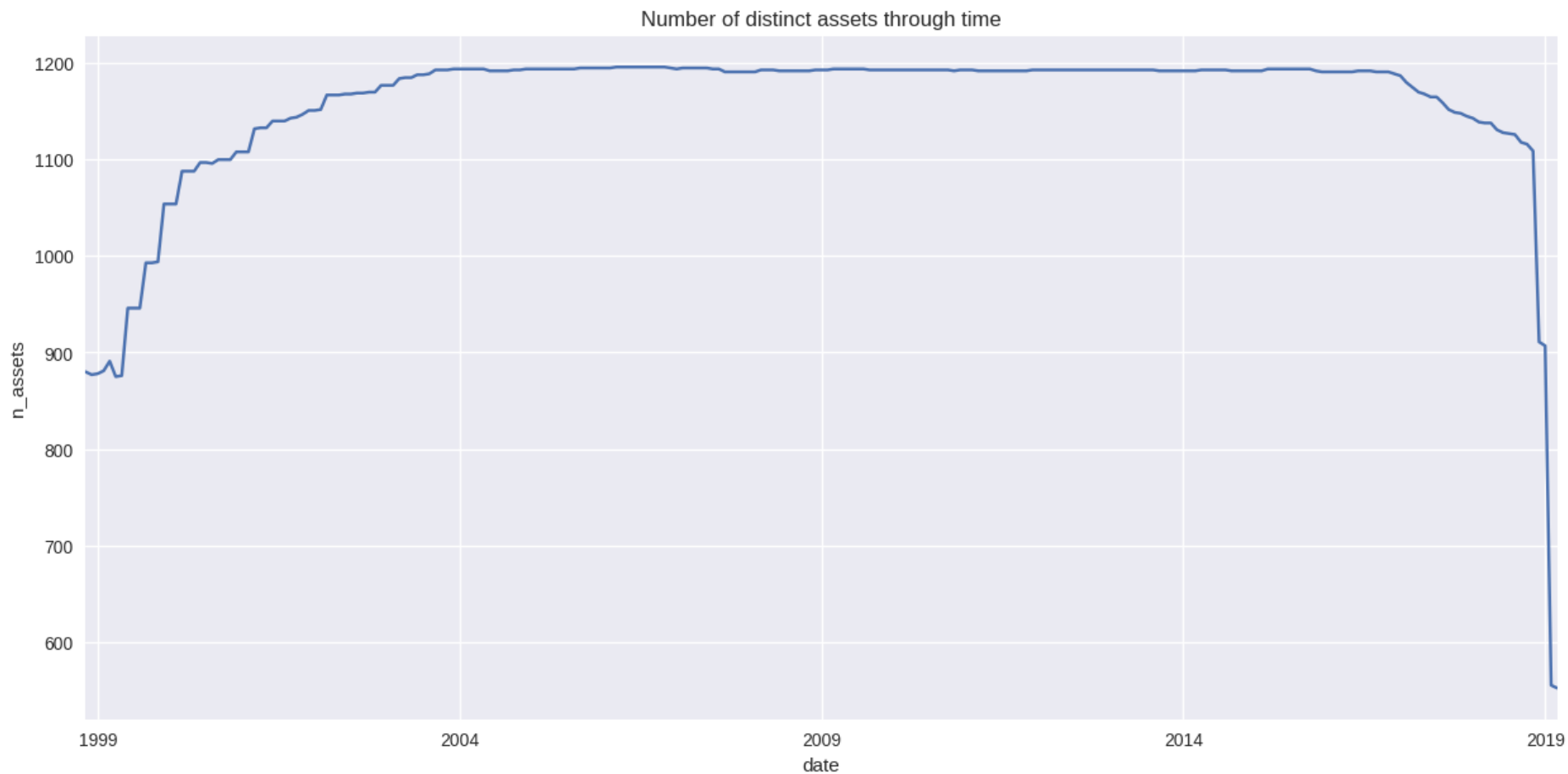
return forward 3 months (**LABEL**)

return forward 6 months (**LABEL**)

return forward 12 months (**LABEL**)

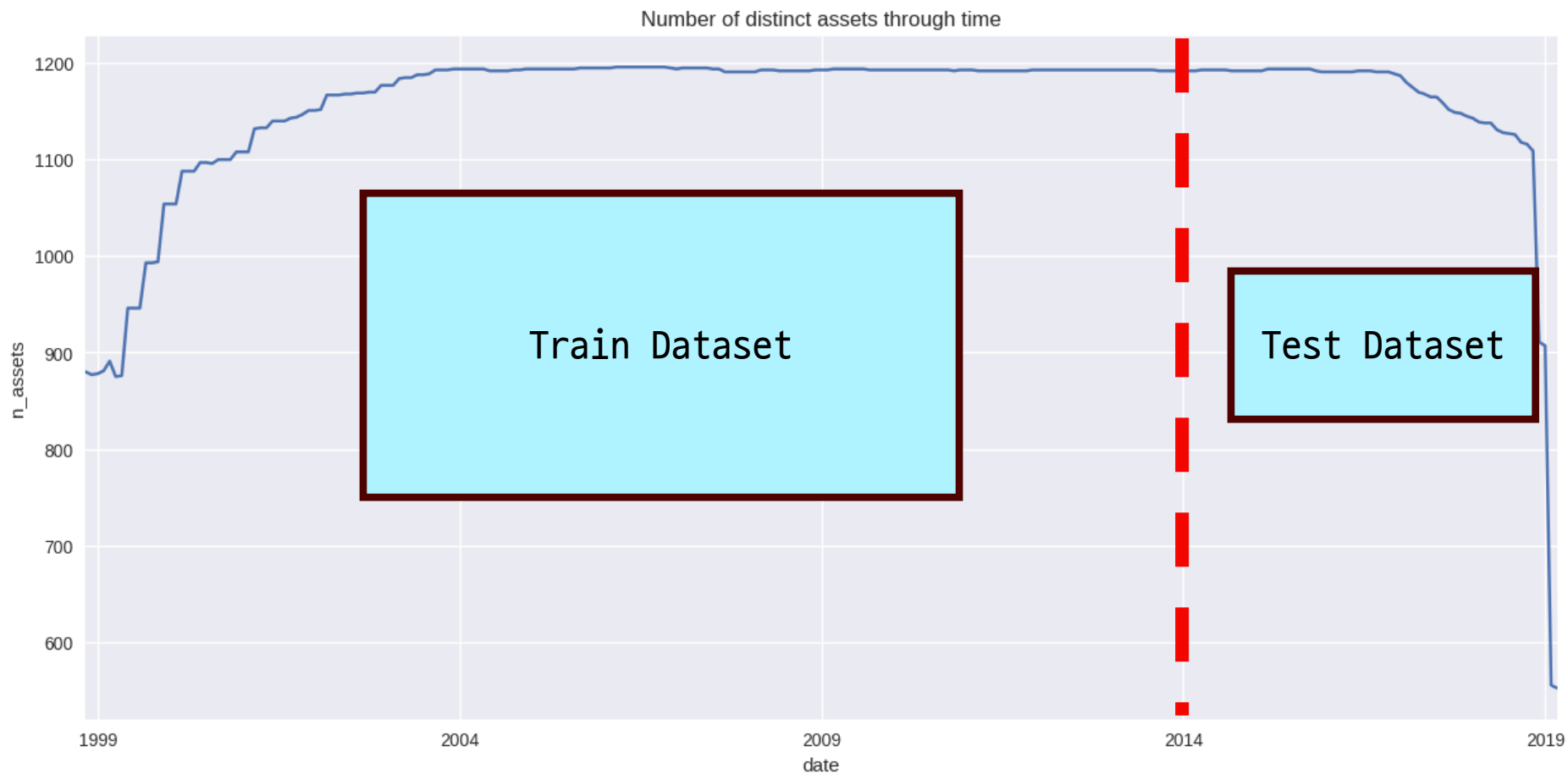
Data/Machine Learning에 대한 간단한 이해

- 1999년부터 2019년까지의 주식을 이용해서 학습합니다.



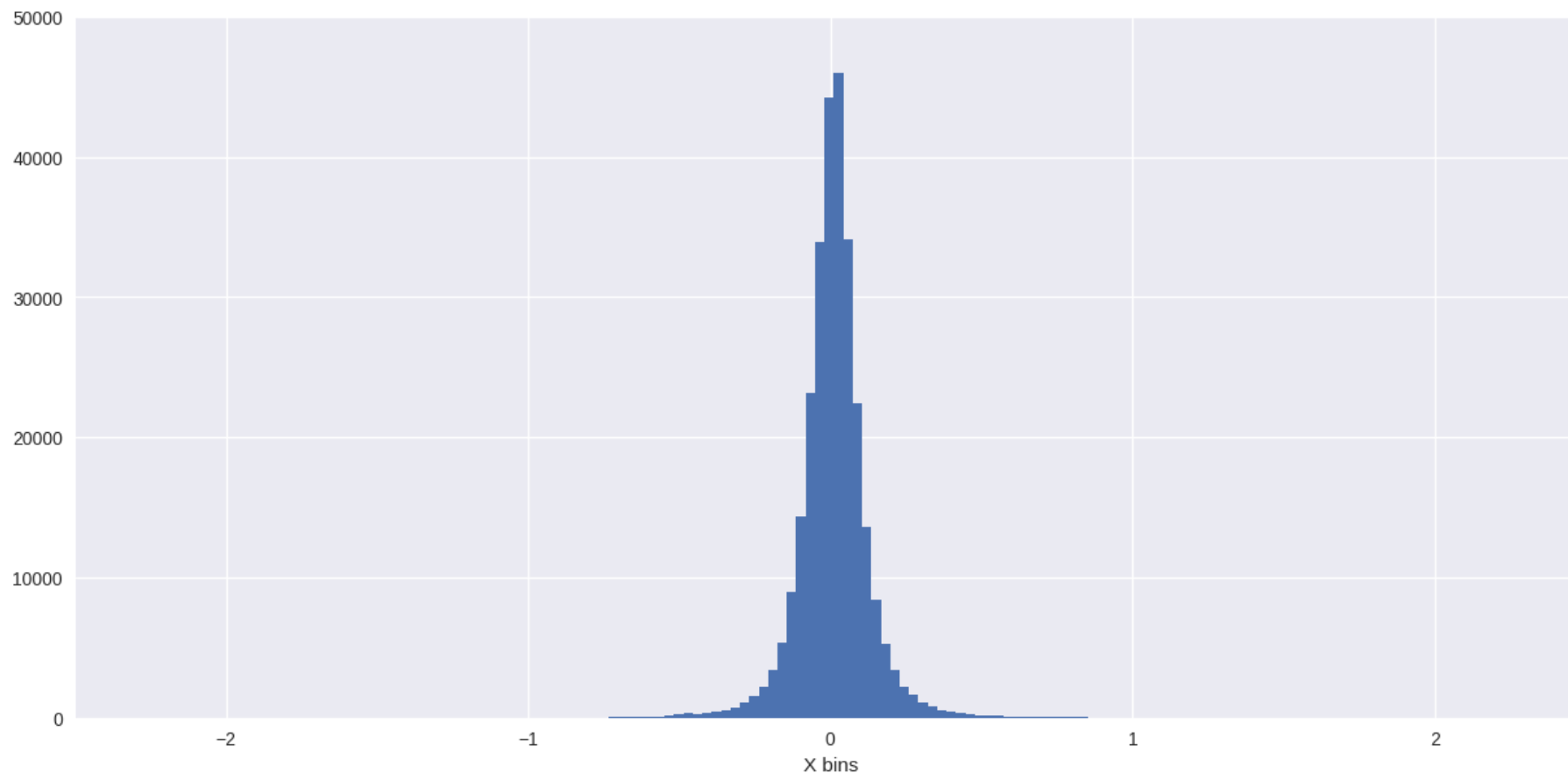
Data/Machine Learning에 대한 간단한 이해

데이터셋을 Train/Test Dataset으로 분할합니다.



Data/Machine Learning에 대한 간단한 이해

수익률 분포

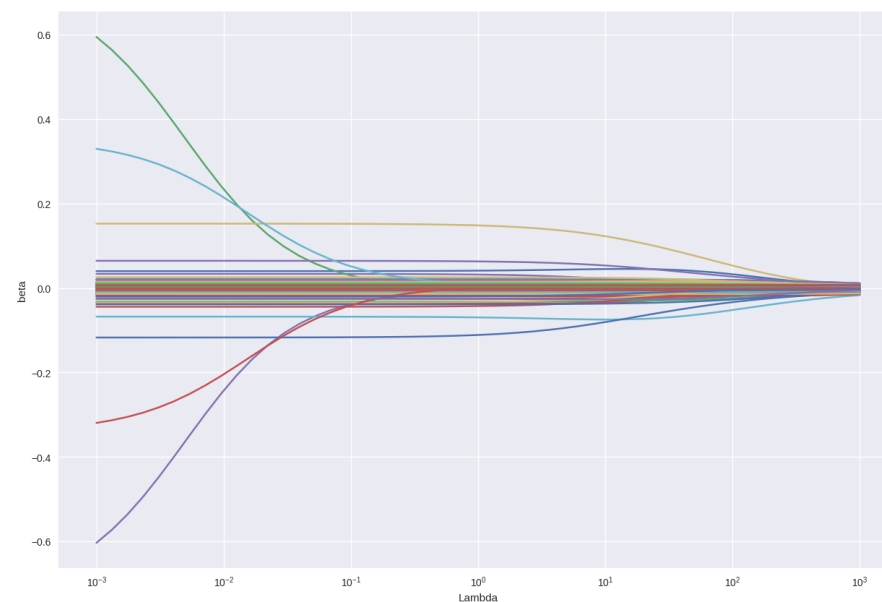
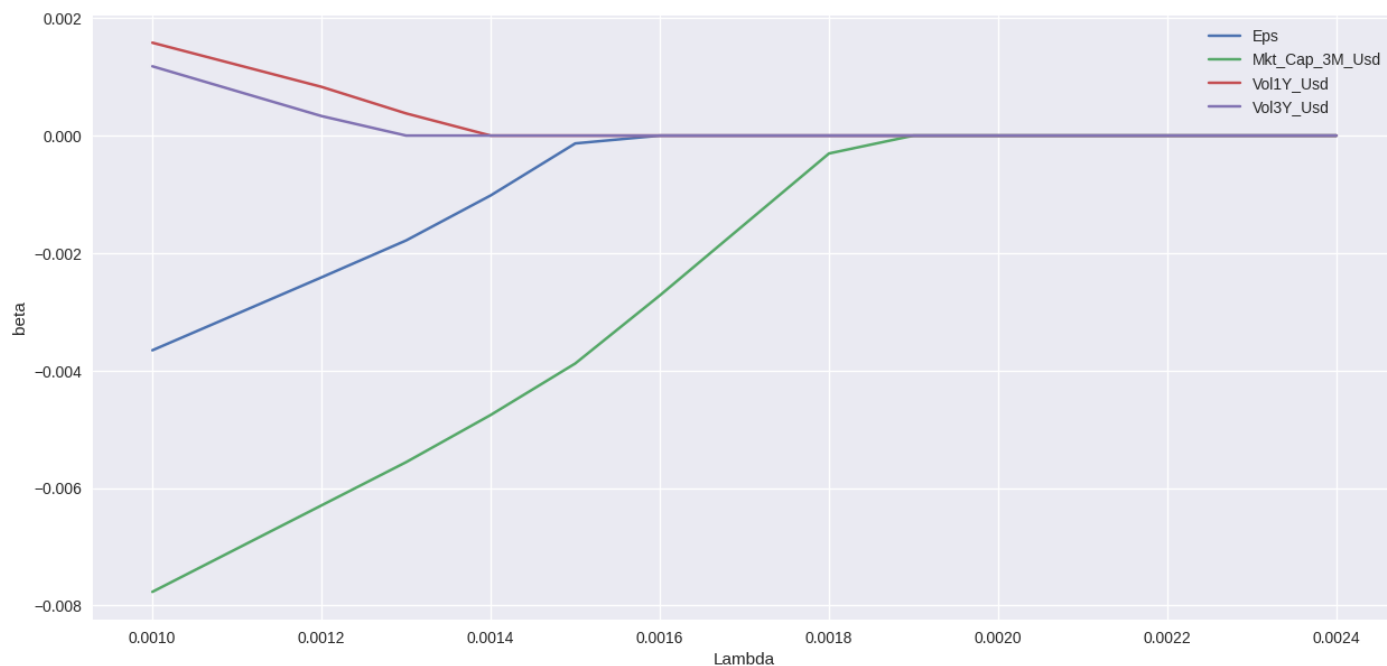


여러 Machine Learning 기법에 대한 설명

- Regularization이 적용된 Regression
- Decision Tree
- Neural Network

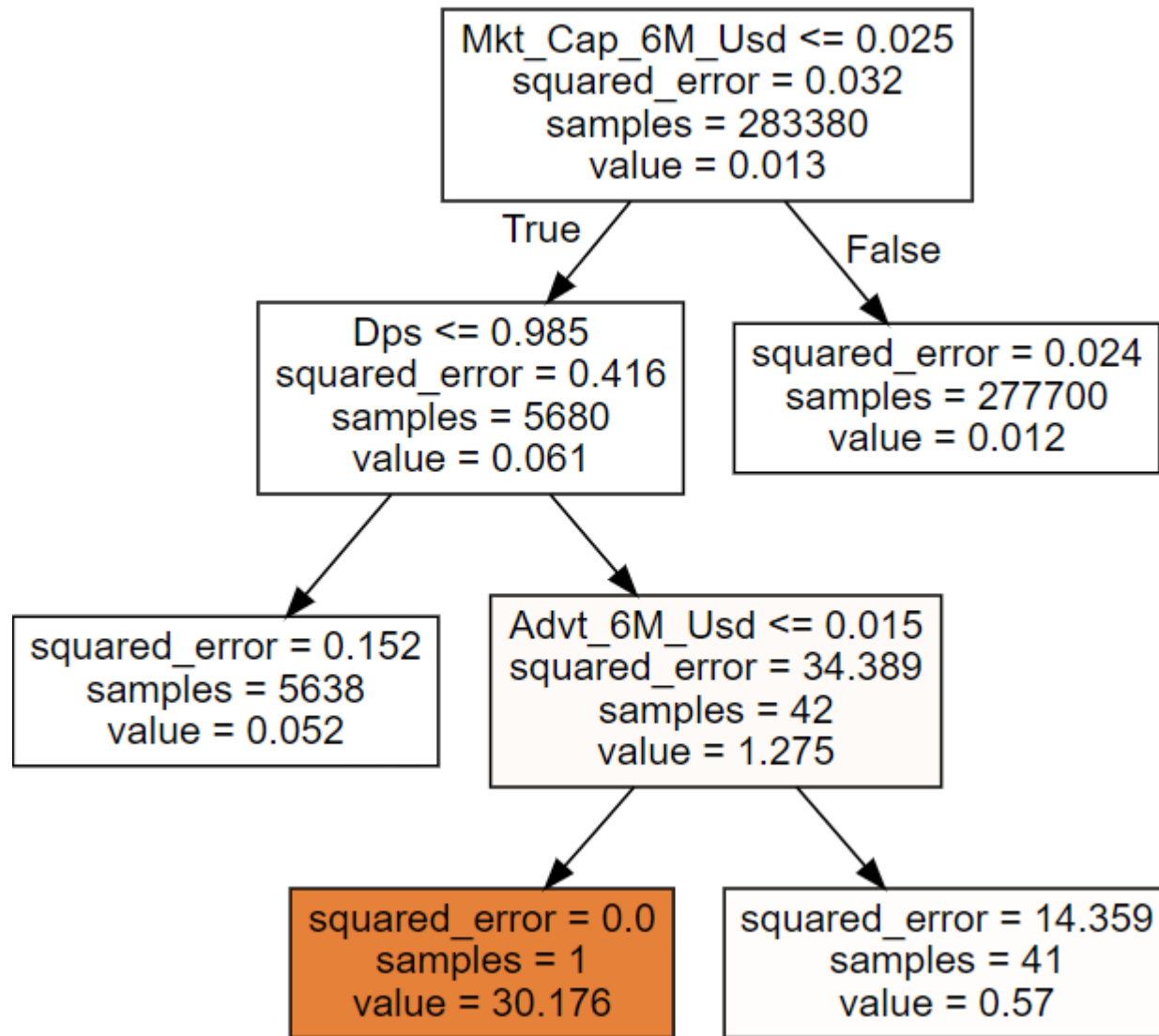
Regularization이 적용된 Regression

- 기존의 선형회귀와 다른 부분이 없지만,
규제 정도에 따라서 feature들이 0에 가까워지는 방법론
- 종속변수와 독립변수의 관계가 선형적일 것 같을 때 사용하면 효율적(선형회귀와 동일함)



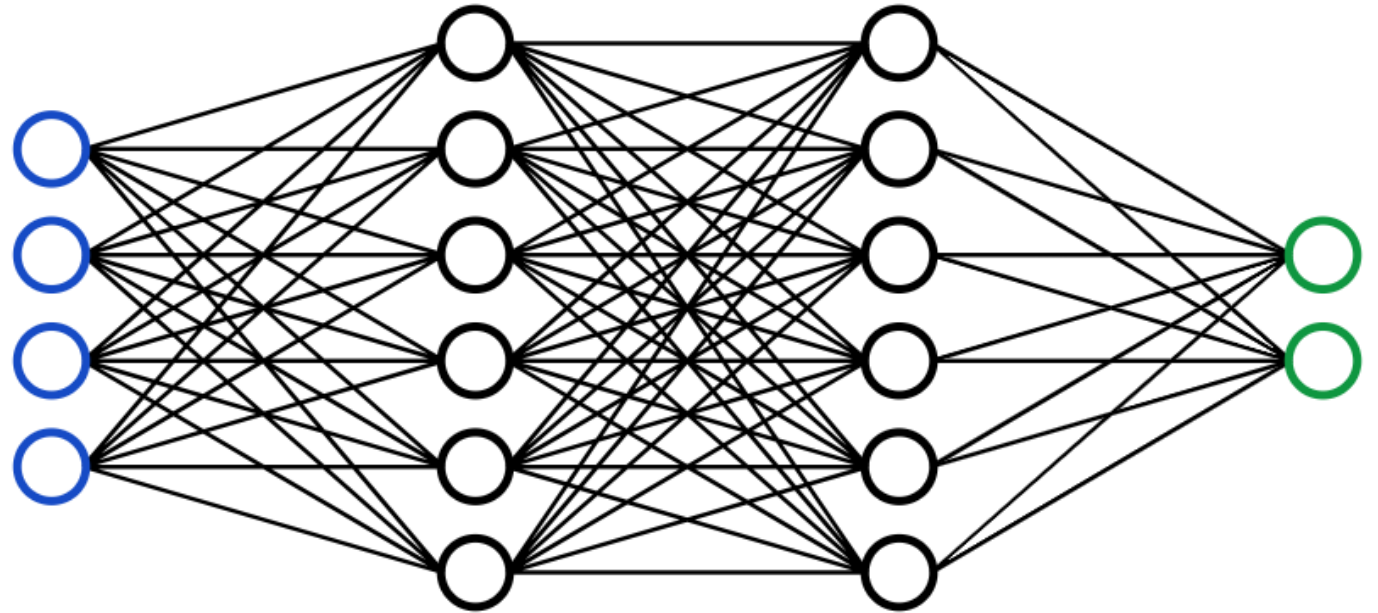
Decision Tree

- 데이터를 선형적으로 분석하지 않고
가지치기를 통해서 분류/회귀하는 방법

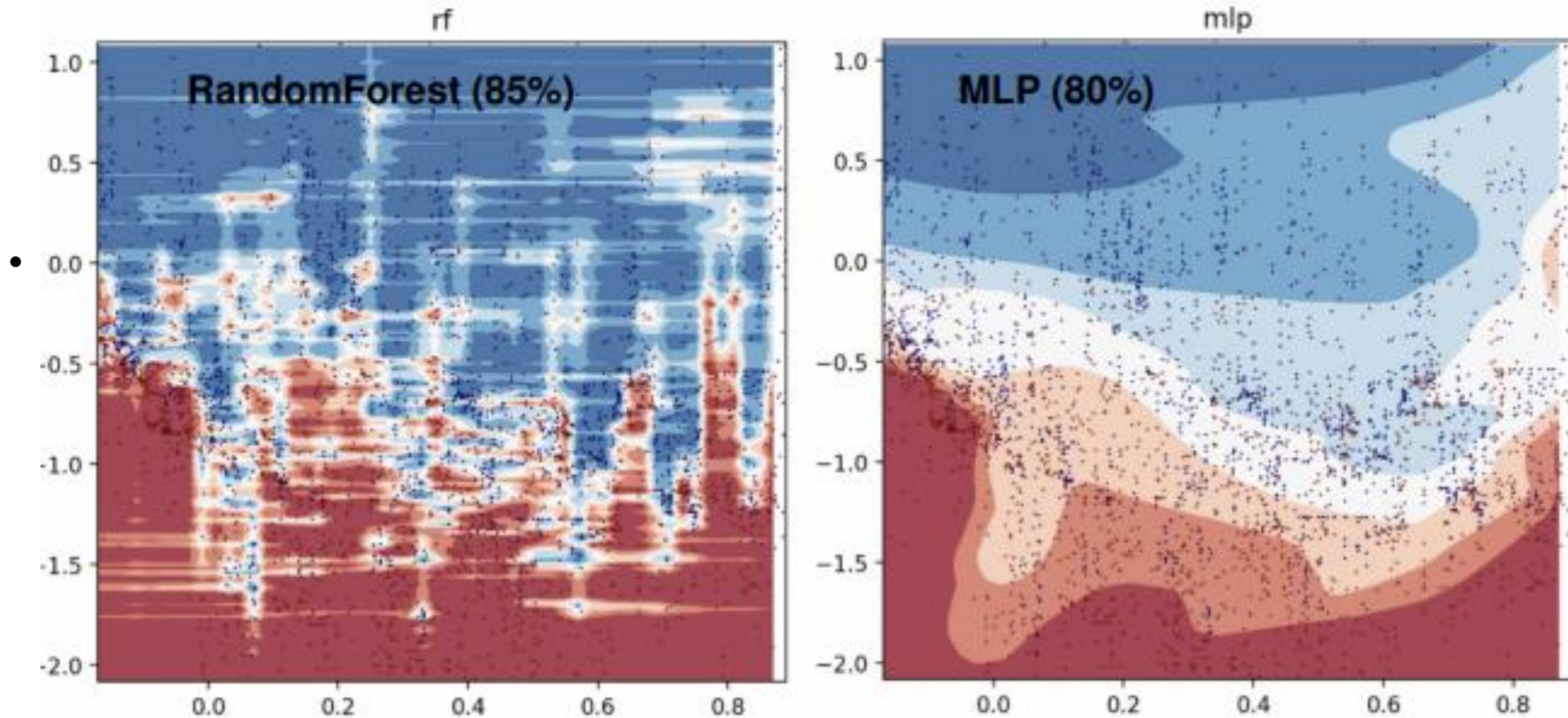


Neural Network

- 수많은 가중치들을 이용해서 학습하는 방법
- 비선형성이 예상되는 복잡한 데이터셋에 적용

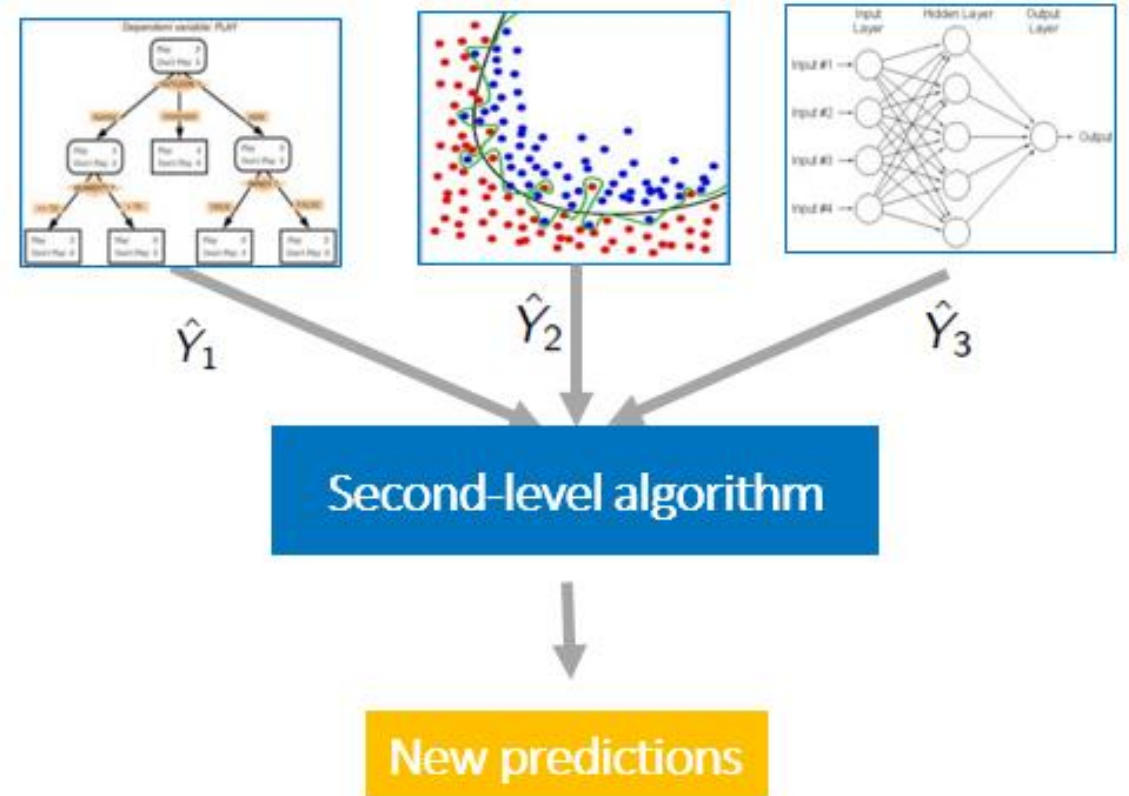


[TMI] Decision Tree vs Neural Network



Ensemble Model

- 여러 모델을 합쳐서 성능을 올리고 강건성을 높이는 방법



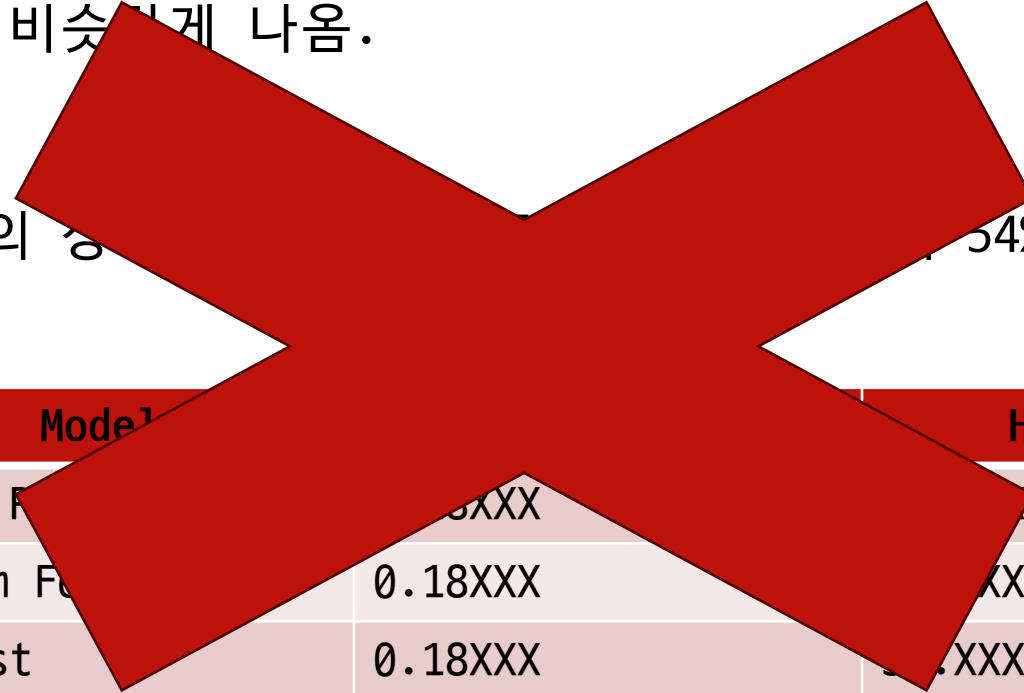
실제 Machine Learning을 사용한 결과

- 모델 성능이 거의 비슷하게 나옴.
- Hit Ratio(Return의 상승/하강을 올바르게 맞춘 비율)이 54%에 불과함.

Model	RMSE	Hit Ratio
Lasso Regression	0.18XXX	54.XXX%
Random Forest	0.18XXX	54.XXX%
XGBoost	0.18XXX	54.XXX%
Dense Layer Network	0.18XXX	54.XXX%
RNN Layer Network	0.18XXX	54.XXX%

실제 Machine Learning을 사용한 결과

- 모델 성능이 거의 비슷하게 나옴.
- Hit Ratio(Return의 정확도)가 54%에 불과함. ○



Model	Hit Ratio
Lasso Regression	0.18XXX%
Random Forest	0.18XXX%
XGBoost	0.18XXX%
Dense Layer Network	54.XXX%
RNN Layer Network	54.XXX%

Model explanation

- 모델 성능이 거의 비슷하게 나옴. 그러나 중요하다고 생각하는 Factor는 조금씩 다름

lasso_feature_importances	
Mkt_Cap_3M_Usd	0.009067
Vol3Y_Usd	0.004030
Eps	0.003746
Vol1Y_Usd	0.003230
Pb	0.002648

xgb_feature_importances	
Fa_Ci	0.097230
Share_Turn_12M	0.064882
Capex_Sales	0.060991
Mom_5M_Usd	0.048633
Noa	0.041626

rf_feature_importances	
Mkt_Cap_3M_Usd	0.151224
Vol3Y_Usd	0.114708
Vol1Y_Usd	0.083249
Mkt_Cap_6M_Usd	0.072491
Pb	0.058959

ensemble_model_importances	
xgb	0.571890
lasso	0.222671
rf	0.205440

Model explanation

	lasso_feature_importances	xgb_feature_importances	rf_feature_importances	sum	weighted_sum
Mkt_Cap_3M_Usd	0.009067	0.005010	0.151224	0.165300	0.035951
Vol3Y_Usd	0.004030	0.004135	0.114708	0.122874	0.026828
Eps	0.003746	0.001997	0.012984	0.018728	0.004644
Vol1Y_Usd	0.003230	0.006672	0.083249	0.093151	0.021637
Pb	0.002648	0.005653	0.058959	0.067260	0.015935
Ni	0.000000	0.008060	0.006518	0.014578	0.005948
Ni_Avail_Margin	0.000000	0.000878	0.001858	0.002735	0.000884
Op_Prt_Margin	0.000000	0.001080	0.001021	0.002101	0.000827
Op_Margin	0.000000	0.005095	0.002770	0.007865	0.003483
Ocf_Toa	0.000000	0.000708	0.001493	0.002201	0.000712

Model explanation

- 모델의 성능이 좋았다면 더욱 유의미한 Factor겠지만 머신러닝을 통해서 Factor를 추정할 수 있다.

	weighted_sum
Fa_Ci	0.055792
Share_Turn_12M	0.037565
Mom_5M_Usd	0.036913
Mkt_Cap_3M_Usd	0.035951
Capex_Sales	0.035225
Vol3Y_Usd	0.026828
Noa	0.024248
Vol1Y_Usd	0.021637
Mom_11M_Usd	0.021022
Div_Yld	0.018760
Ebit_Noa	0.018389
Ni_Toa	0.018103
Mkt_Cap_6M_Usd	0.017146
Advt_12M_Usd	0.017005

Model explanation

- Neural NetWork에 대한 model 설명은 어디갔나요??

개선할 부분 3가지

- 1달 수익률 예측이 아닌, 3달/6달/12달 수익률 예측
- 1달 데이터만 사용하는 것이 아닌 n달 데이터를 사용해서 예측
- Feature selection을 진행 후 모델학습

1달 수익률 예측이 아닌, 3달/6달/12달 수익률 예측

3달 수익률 예측

```
[ ] 1 xgb_model = xgboost.XGBRegressor(n_estimators=40, learning_rate=0.01, gamma=0,
2                                     colsample_bytree=1, max_depth=7)
3 xgb_model.fit(X_train_3M, y_train_3M)
4
5 mse = np.mean((xgb_model.predict(X_test_3M) - y_test_3M)**2)
6 hitratio = np.mean(xgb_model.predict(X_test_3M) * y_test_3M > 0)
7
8 print(f'MSE: {mse} \nHit Ratio: {hitratio}')
```

MSE: 0.16062943896227205
Hit Ratio: 0.5466616820359447

6달 수익률 예측

```
▶ 1 xgb_model = xgboost.XGBRegressor(n_estimators=40, learning_rate=0.01, gamma=0,
2                                     colsample_bytree=1, max_depth=7)
3 xgb_model.fit(X_train_6M, y_train_6M)
4
5 mse = np.mean((xgb_model.predict(X_test_6M) - y_test_6M)**2)
6 hitratio = np.mean(xgb_model.predict(X_test_6M) * y_test_6M > 0)
7
8 print(f'MSE: {mse} \nHit Ratio: {hitratio}')
```

▶ MSE: 0.1573719557617103
Hit Ratio: 0.5467309130181939

12달 수익률 예측

```
▶ 1 xgb_model = xgboost.XGBRegressor(n_estimators=40, learning_rate=0.01, gamma=0,
2                                     colsample_bytree=1, max_depth=5)
3 xgb_model.fit(X_train_12M, y_train_12M)
4
5 mse = np.mean((xgb_model.predict(X_test_12M) - y_test_12M)**2)
6 hitratio = np.mean(xgb_model.predict(X_test_12M) * y_test_12M > 0)
7
8 print(f'MSE: {mse} \nHit Ratio: {hitratio}')
```

▶ MSE: 0.23059168900719546
Hit Ratio: 0.5205616017280054

1달 데이터만 사용하는 것이 아닌 n달 데이터를 사용해서 예측 + Feature selection을 진행 후 모델학습

XGB model

모델링

XGB모델과 NN모델을 사용

```
[86] 1 xgb_model = xgboost.XGBRegressor(n_estimators=100, learning_rate=0.01, gamma=0,
2                                     colsample_bytree=1, max_depth=7)
3 xgb_model.fit(top20_3M_training_X, top20_3M_training_y)
4
5 mse = np.mean((xgb_model.predict(top20_3M_testing_X) - top20_3M_testing_y)**2)
6 hitratio = np.mean(xgb_model.predict(top20_3M_testing_X) * top20_3M_testing_y > 0)
7 print(f'MSE: {mse} \nHit Ratio: {hitratio}')
```

```
MSE: 0.0322301872074604
Hit Ratio: 0.5063902418716956
```

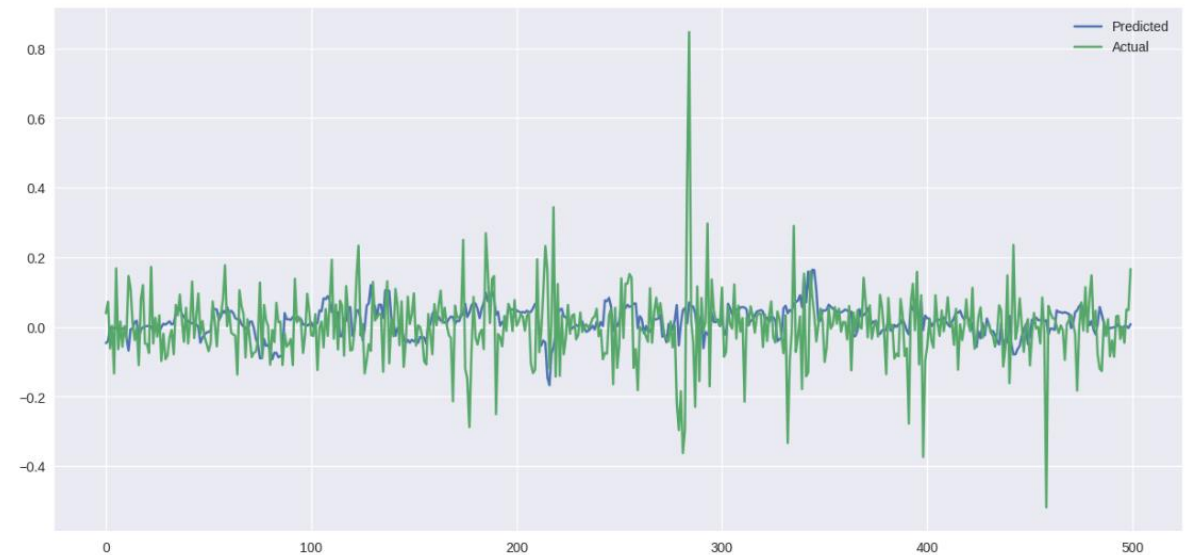
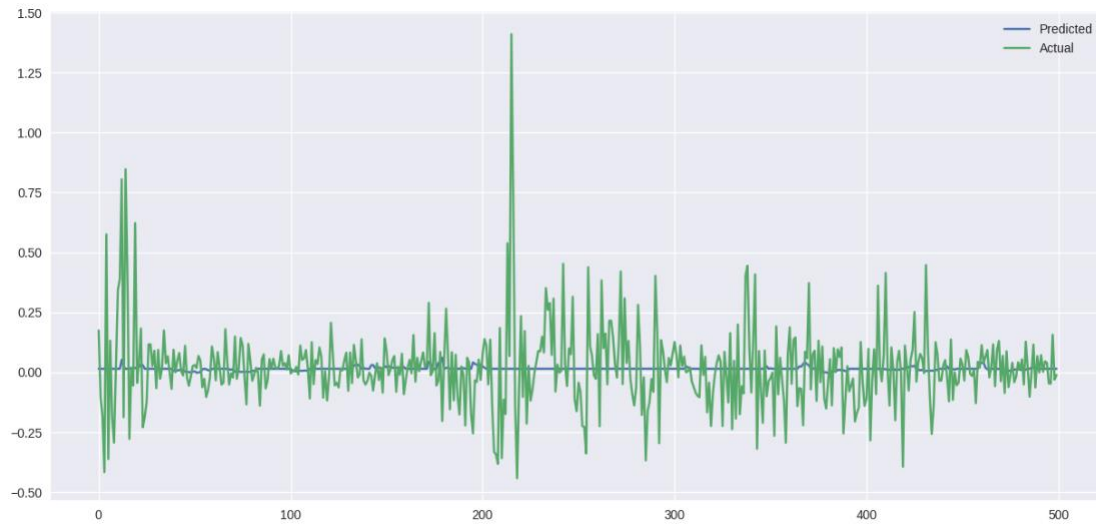
MSE는 줄었지만, Hit Ratio도 함께 줄었음.

왜 그럴까요?

Neural network

```
Epoch 36/100
272/272 [=====] - 1s 5ms/step - loss: 0.0314 - mean_absolute_error: 0.0887 - val_loss: 0.0238 - val_mean_absolute_error: 0.0719
Hit Ratio: 0.4961492027809
```

1달 데이터만 사용하는 것이 아닌 n달 데이터를 사용해서 예측 + Feature selection을 진행 후 모델학습



Backtesting

	avg_ret	vol	Sharpe_ratio	VaR_5	turn
EW	0.010654	0.056676	0.187987	-0.076794	0.078321
XGB_SR	0.013146	0.063434	0.207242	-0.083936	0.569412
LASSO_SR	0.014136	0.065450	0.215977	-0.084973	0.245735

