Factor-ML

1 Overview

Factor-ML uses XGBoost-predicted expected returns to form a long-short equal-weighted portfolio. Expected returns are estimated using the XGBoost model from Chen and Guestrin (2016), following the methodology of Jensen et al. (2024). Stocks are sorted into deciles based on predicted returns, and the portfolio goes long the top decile and short the bottom decile with equal weights within each side.

2 Expected Return Estimation

We estimate expected returns using the XGBoost model (Chen and Guestrin 2016) to predict realized excess returns over the next month as a function of security characteristics today. Security characteristics are percentile-ranked cross-sectionally and missing values are imputed with zeros (the cross-sectional median after ranking).

2.1 Hyperparameter Tuning

Our approach to selecting hyperparameters follows a two-stage procedure using 5-fold cross-validation within the training window.

Stage 1. We train 20 different models on the training data based on the 20 sets of hyperparameters from Table 1 and a fixed learning rate of 0.15. We select the set of hyperparameters that leads to the lowest mean squared error on the validation data.

Stage 2. We train a new model on the training data using the parameters found in the first stage, except that we now train the model with a learning rate of 0.01. We then record the optimal number of iterations for the model (i.e., the number of individual decision trees to include in the ensemble before the model starts to overfit), which is the number of model iterations that resulted in the lowest mean squared error on the validation data.

Finally, we train a model on the training and validation data using a learning rate of 0.01, the non-learning rate parameters from stage 1, and the optimal number of iterations from stage 2, and use this model to estimate expected returns, $\widehat{\mu}_t$.

2.2 Hyperparameter Grid

Table 1: XGBoost Hyperparameters

Set	Features	Tree depth	Sample size	Penalty	Learning rate
1	0.96	1	0.79	6.79	0.15/0.01
2	0.12	6	0.84	4.37	0.15/0.01
3	0.02	1	0.28	8.96	0.15/0.01
4	0.08	6	0.24	33.70	0.15/0.01
5	0.88	6	0.61	0.18	0.15/0.01
6	0.31	7	0.28	0.01	0.15/0.01
7	0.49	7	0.27	2.70	0.15/0.01
8	0.92	3	0.54	0.02	0.15/0.01
9	0.29	2	0.80	4.72	0.15/0.01
10	0.07	7	0.35	0.54	0.15/0.01
11	0.18	2	0.40	0.63	0.15/0.01
12	0.39	4	0.42	6.08	0.15/0.01
13	0.73	4	0.21	0.19	0.15/0.01
14	0.93	3	0.22	2.51	0.15/0.01
15	0.98	6	0.99	4.13	0.15/0.01
16	0.65	3	0.82	97.33	0.15/0.01
17	0.66	6	0.67	47.64	0.15/0.01
18	0.89	1	0.30	0.17	0.15/0.01
19	0.65	5	0.28	88.63	0.15/0.01
20	0.91	4	0.59	5.97	0.15/0.01

Note: The table shows 20 sets of hyperparameters that we search over when fitting the XGBoost model to predict next month's realized excess return. "Features" is the fraction of the features chosen randomly for each decision tree, "Tree depth" is the maximum depth of each decision tree, "Sample size" is the fraction of the observations chosen randomly for each decision tree, "Penalty" is an L2 (ridge) penalty, and "Learning rate" is the weight each new tree gets in the ensemble. We use a two-stage tuning strategy, where the learning rate is 0.15 in the first stage, and 0.01 in the second. We get the hyperparameter sets by specifying a tolerable range for each hyperparameter and then use the `parameters` function from the `dials` package (https://dials.tidymodels.org/) with the type set to "max_entropy" to get 20 sets that aim to cover the associated parameter space. The ranges are features $\in [1/\#\text{features}, 1]$, tree depth $\in [1, 7]$, sample size $\in [0.2, 1]$, and penalty $\in [10^{-2}, 10^{2}]$. All parameters are chosen directly from their natural scales, except for penalty, which is chosen from a logarithmic (base 10) scale.

3 Portfolio Construction

The Factor-ML portfolio uses the expected return estimates $\hat{\mu}_t$ from the XGBoost model and buys the 10% of stocks with the highest expected returns while shorting the 10% of stocks with the lowest expected returns. Stocks are equal-weighted within the long and short sides of the portfolio.

References

Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–94. ACM.

Jensen, Theis Ingerslev, Bryan T. Kelly, Semyon Malamud, and Lasse Heje Pedersen. 2024. "Machine Learning and the Implementable Efficient Frontier." *Journal of Financial Economics* 159: 103925.