Markowitz-ML

1 Overview

Markowitz-ML constructs an ex-ante mean-variance optimal tangency portfolio by combining XGBoost-predicted expected returns with a Barra-style factor covariance matrix. Expected returns are estimated using the same XGBoost pipeline as Factor-ML, and the variance-covariance matrix is estimated using the same Barra USE4S factor model as Minimum Variance. The portfolio is scaled to a target annualized volatility of 10%.

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, following the methodology of Jensen et al. (2025). Security characteristics are percentile-ranked cross-sectionally (between 0 and 1) and then we subtract 0.5 to center it around zero. Missing values are imputed with zeros.

2.1 Training Setup

We use a rolling window of 10 years of training data. For each test month, the training data is split into 5 temporal folds (i.e., 5 consecutive blocks of months in chronological order) for cross-validation. The XGBoost base score (the initial prediction before any trees are added) is set to 0.

The frequency of hyperparameter tuning is controlled by the `test_period_length` parameter, which specifies the number of consecutive test months per chunk. With `test_period_length = 12` (the default), hyperparameters are re-tuned once per year: the model is trained on data up to the first month in the chunk and then used to predict all months in that chunk. Setting `test_period_length = 1` re-tunes every month.

2.2 Hyperparameter Tuning

Our approach to selecting hyperparameters follows a two-stage procedure.

Stage 1. We train 20 different models on the training data based on 20 sets of hyperparameters and a fixed learning rate of 0.15, with a maximum of 1,000 iterations and early stopping after 25 rounds without improvement. For each hyperparameter set, we evaluate the mean squared error across the 5 temporal folds and select the set that leads to the lowest average mean squared error.

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 and a maximum of 10,000 iterations (with the same early stopping rule of 25 rounds). 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), averaged across the 5 folds.

Finally, we train a model on the full training data (i.e., all data in the rolling window) 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, $\hat{\mu}_t$.

2.3 Hyperparameter Grid

The 20 hyperparameter sets are generated using the `grid_space_filling` function from the `dials` package (https://dials.tidymodels.org/) with the type set to "max_entropy" to cover the parameter space. The tuned hyperparameters and their ranges are:

- **Features** (`colsample_bytree`): fraction of features sampled for each tree, range $[1/K, 1]$ where K is the number of features
- **Tree depth** (`max_depth`): maximum depth of each tree, range $[1, 7]$
- **Sample size** (`subsample`): fraction of observations sampled for each tree, range $[0.2, 1]$
- **Penalty** (`reg_lambda`): L2 ridge penalty, range $[10^{-2}, 10^{2}]$ on a $\log_{10}$ scale

All parameters are drawn from their natural scales, except for penalty, which is drawn from a logarithmic (base 10) scale. Two additional parameters are held fixed: minimum child weight (`min_child_weight`) is set to 1, and minimum split loss (`gamma`) is set to 0.

3 Variance-Covariance Estimation

3.1 Factor Model

To estimate the variance-covariance matrix, we use an approach similar to the one used by MSCI Barra, which is based on the assumption that returns follow a linear factor model. The idea is to treat security characteristics as observable factor loadings and infer the latent factor returns from cross-sectional regressions of excess returns on security characteristics. Specifically, each day we estimate the cross-sectional ridge regression:

$$r_{i,t+1} = x'_{i,t}\hat{f}_{t+1} + \hat{\epsilon}_{i,t+1}, \tag{1}$$

where $r_{i,t+1}$ is the stock's realized excess return, x is a vector of the stock's characteristics, $\hat{\epsilon}_{i,t+1}$ is the regression residual, and $\hat{f}_{t+1}$ is the estimated ridge regression parameters, which we treat as estimated factor returns. The regression has no intercept; instead, all 12 industry dummies are included so that they absorb what would otherwise be the intercept. The ridge penalty is set to 10^{-4}. The stock characteristics are the 402 `features` and 12 industry dummies based on SIC codes and the industry definition from Kenneth French's webpage.[1]

The structure in Equation 1 implies that the variance-covariance matrix is:

$$\hat{\Sigma}_t = X_t \mathrm{Var}_t(\hat{f}_{t+1}) X'_t + \mathrm{diag}(\mathrm{Var}_t(\hat{\epsilon}_{t+1})),$$

[1] Our approach differs from Jensen et al. (2025), who use a compressed version of 13 factor themes (made from the original 153 JKP characteristics) plus the 12 industry dummies. We use the raw characteristics to enable the variance-covariance estimate to change as we change the set of characteristics. The Sharpe ratio is higher if we use the 13 factor themes instead of the 402 raw characteristics. We also differ by using ridge instead of OLS regression. We use ridge because some of our characteristics are highly correlated, and using a small penalty makes the estimates robust to near multicollinearity.

where X_t is the matrix of stock characteristics, $\text{Var}(\hat{f}_{t+1})$ is the variance-covariance matrix of factor returns, and $\text{diag}(\text{Var}(\epsilon_{t+1}))$ is a matrix with the idiosyncratic variances in the diagonal and zero elsewhere.

3.2 Factor Return Covariance

We estimate $\text{Var}(\hat{f}_{t+1})$ as the exponentially weighted sample variance-covariance matrix of the last ten years of daily returns. The exponential weighting scheme gives observations j days from t a weight of $w_{t-j} = c \cdot 0.5^{j/\text{half-life}}$, where c is a constant ensuring that the weights sum to one, and it ensures that recent observations affect the estimate more than distant ones. Following the MSCI Barra USE4S model, we use a half-life of 504 days for correlations and 84 days for variances (Menchero, Orr, and Wang 2011, Table 4.1).

3.3 Idiosyncratic Variance

Similarly, we estimate each stock's idiosyncratic variance, $\text{Var}_t(\hat{\epsilon}_{i,t+1})$, as the exponentially weighted moving average of squared residuals, $\hat{\epsilon}_{i,t+1}$ from Equation 1, with a half-life of 84 days. The EWMA variance is initialized (seeded) from the first 63 observations (~3 months). The half-life is again chosen as the one from the MSCI Barra USE4S model (Menchero, Orr, and Wang 2011, Table 5.1). To estimate the idiosyncratic variance we require at least 200 non-missing observations within the last 252 trading days. For stocks without an idiosyncratic variance estimate, we estimate the function, $\ln\left(\sqrt{\widehat{\text{Var}}_t(\hat{\epsilon}_{i,t+1})}\right) = \hat{f}_t(x_{i,t})$, using a ridge regression model with a small ridge penalty of 10^{-4}, and use it to estimate the missing variances.[2]

4 Portfolio Construction

Given the estimated expected returns $\hat{\mu}_t$ and variance-covariance matrix $\hat{\Sigma}_t$, the Markowitz-ML portfolio solves for the tangency portfolio:

$$w_t^{\text{raw}} = \hat{\Sigma}_t^{-1}\hat{\mu}_t.$$

To avoid inverting the large $N \times N$ covariance matrix directly, we exploit the factor structure and use the Woodbury matrix identity, which reduces the problem to inverting a $K \times K$ matrix (where K is the number of factors).

The raw weights are then rescaled to target an annualized volatility of 10%:

$$w_t = w_t^{\text{raw}} \cdot \sqrt{\frac{\sigma^2_{\text{target}}/12}{w_t^{\text{raw}\prime}\hat{\Sigma}_t^{\text{monthly}} w_t^{\text{raw}}}},$$

where $\sigma_{\text{target}} = 0.10$ and $\hat{\Sigma}_t^{\text{monthly}} = 21 \cdot \hat{\Sigma}_t$ converts the daily covariance matrix to a monthly frequency. In practice, the realized volatility of the portfolio substantially exceeds the target because mean-variance optimization amplifies estimation error in the covariance matrix, concentrating weights on stocks whose risk is underestimated.

[2] Our approach follows MSCI Barra (Menchero, Orr, and Wang 2011, Eq. 5.3), except that they use an OLS regression. We use a ridge regression because some of our characteristics are highly correlated, and using a small penalty helps make the estimates robust to near multicollinearity.

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. 2025. "Machine Learning and the Implementable Efficient Frontier." *Review of Financial Studies (Forthcoming)*.

Menchero, Jose, D. J. Orr, and Jun Wang. 2011. "Barra US Equity Model (USE4)." Methodology Notes. MSCI Barra.