

XGBoost–Ledoit-Wolf Constrained MVP

This model constructs a risk-controlled portfolio via a three-step pipeline: generating non-linear return predictions using XGBoost, estimating a well-conditioned covariance matrix via Ledoit-Wolf shrinkage, and applying a constrained mean-variance optimization to determine optimal portfolio weights.

1. Data preparation

I utilize the dataset provided for the 'Common Task Framework' by Hellum et al. (2025), which has already been cleaned and restricted to US non-micro-cap stocks.

To prevent look-ahead bias and ensure information is not shared across time periods, I cross-sectionally standardize each feature on a monthly basis. Specifically, I rank each feature and scale it to a range between 0 and 1. I then subtract 0.5 so the final ranks are centered between -0.5 and 0.5. Missing values are imputed with the cross-sectional median of zero, yielding a complete dataset.

Additionally, I engineer a new target variable for the return prediction model, where I subtract the equally-weighted cross-sectional average excess return from each individual stock's excess return for that given month, i.e. $adjusted_ret_{i,t} = ret_exc_lead1m_{i,t} - \frac{1}{N_t} \sum_{j=1}^{N_t} ret_exc_lead1m_{j,t}$. This ensures that the mean of `adjusted_ret` for every month is exactly zero. I apply this transformation because the objective is to capture the relative performance between stocks in the cross-section, and this prevents the model from being unduly influenced by months with extreme average returns.

2. Return prediction

I use the XGBoost model from Chen and Guestrin (2016) to predict excess returns for the next month for each stock. I use the new variable `adjusted_ret` as the target variable. I select XGBoost, a scalable gradient boosting framework, over linear models for two reasons. First, it natively captures complex non-linear interactions without explicit specification. Second, its built-in L1/L2 regularization and sequential residual learning mitigate overfitting, which is critical for the low signal-to-noise ratio of stock returns.

2.1 Hyperparameter tuning

XGBoost requires the specification of a wide array of hyperparameters. To optimize these, I employ a two-stage hyperparameter tuning process. In the first stage, I define a search space across eight hyperparameters, resulting in 4,374 possible combinations. Because an exhaustive grid search would be computationally prohibitive, I randomly sample 150 combinations for evaluation. By analyzing the validation RMSE from the search, I identify the hyperparameters most critical to model performance. For the second stage, I perform a targeted grid search over the refined subset. The remaining, less impactful hyperparameters are fixed to the values of the best-performing model (lowest average RMSE) from the first stage. To mitigate overfitting, I apply a conservative selection criterion among the top 3 performing models in the second stage, explicitly choosing the least complex model.

Both tuning stages utilize the same cross-validation framework, restricted entirely to the pre-1990 training sample (`cttf_test == False`) to prevent look-ahead bias. Specifically, I implement an expanding-window cross-validation methodology. I initialize the training set with a minimum of 15 years and split the remaining in-sample period into five chronological validation folds. For each configuration, the model is initially trained on the 15-year window and evaluated on the subsequent validation fold. The training set then expands chronologically to incorporate each newly evaluated fold. This process repeats across all five folds, yielding the average validation RMSE as the final performance metric.

I do not include the number of boosting rounds as a parameter in the tuning grid. For the final model, I adopt the number of rounds utilized by the winning configuration during the validation phase, which results in approximately 125 rounds. The hyperparameter tuning is summarized in table 1 below.

Table 1 - Hyperparameter Tuning

Hyperparameter	Description	Stage 1 Search Space	Stage 2 Search Space	Final Selected
max_depth	Maximum tree depth (Complexity)	{3, 4, 5}	{3, 4, 5}	4
eta	Learning rate (Shrinkage)	{0.01, 0.05, 0.1}	{0.01, 0.05}	0.05
lambda	L2 regularization term on weights (Ridge)	{1.0, 5.0, 10.0}	{1.0, 5.0, 10.0}	10
alpha	L1 regularization term on weights (Lasso)	{0.0, 0.1, 1.0}	{0.0, 0.1, 1.0}	1
subsample	Fraction of observations used per tree	{0.3, 0.5, 0.7}	<i>Fixed at 0.7</i>	0.7
colsample_bytree	Fraction of features used per tree	{0.3, 0.5, 0.7}	<i>Fixed at 0.5</i>	0.5
gamma	Minimum loss reduction required to split	{0.0, 1.0, 5.0}	<i>Fixed at 0.0</i>	0
min_child_weight	Minimum number of observations per leaf	{10, 30}	<i>Fixed at 30</i>	30

2.2 Model training and prediction

I initialize the final XGBoost model using the optimal hyperparameters identified in the tuning phase, fix the boosting rounds at 125 and use adjusted_ret as the target variable. To generate out-of-sample predictions, I implement a strictly causal, expanding-window forecasting framework. Once a year, the model is retrained on all historically available data up to that point. This updated model is then used to predict the stock returns for the subsequent 12 months. This annual retraining procedure ensures the model continuously adapts to the most recent data while strictly precluding any look-ahead bias.

3. Portfolio construction

To translate the return predictions into tradable portfolio weights, I apply a constrained optimization framework. Each month, the out-of-sample XGBoost predictions serve as the expected return vector. These are combined with a dynamically estimated covariance matrix to construct a monthly-rebalanced portfolio that maximizes returns subject to certain constraints.

3.1 Missing return

To estimate the covariance matrix, I use a rolling one-year lookback window of daily returns. I first address missing daily returns. Since stocks with fewer than 200 valid return observations over the trailing 252 days have already been excluded from the dataset (Hellum et al., 2025), all remaining stocks possess sufficient data for robust imputation. I estimate missing returns using a beta-adjusted model with added noise, ensuring the stock's volatility is preserved. Missing returns for stock i on day t are estimated via a single-factor model: $r_{i,t} = \hat{\beta}_i r_{mkt,t} + \epsilon_{i,t}$. The market proxy, $r_{mkt,t}$, is the cross-sectional mean of all valid returns on day t , and $\hat{\beta}_i$ is estimated using the available overlapping observations. To preserve the stock's idiosyncratic volatility, the noise term is randomly drawn from a normal distribution, $\epsilon_{i,t} \sim N(0, \hat{\sigma}_{\epsilon,i}^2)$, where $\hat{\sigma}_{\epsilon,i}^2$ is the sample variance of the stock's valid residuals.

3.2 Covariance matrix estimation

Using the imputed daily returns, I estimate the covariance matrix for portfolio optimization. Since the standard sample covariance matrix (S) is susceptible to estimation error in high-dimensional settings, I employ the single-index shrinkage estimator introduced by Ledoit and Wolf (2003):

$$\hat{\Sigma}_{LW} = \alpha F + (1 - \alpha)S$$

This approach computes a weighted average between the noisy sample covariance matrix, S , and a highly structured single-index target matrix, F . The off-diagonal elements of the target assume all cross-sectional correlation is driven by the market proxy $r_{mkt,t}$, defined as $F_{ij} = \frac{s_{im}s_{jm}}{s_m^2}$, where s_{im} is the sample covariance between stock i and the market, and s_m^2 is the market variance. The diagonal elements retain the original sample variances $F_{ii} = S_{ii}$. The optimal shrinkage intensity, α , is derived to minimize the expected Frobenius norm between the estimator and the true covariance matrix. Following Ledoit and Wolf (2003), α is estimated from the data as $\max\left(0, \min\left(1, \frac{\hat{\kappa}}{T}\right)\right)$, where $\hat{\kappa}$ captures the ratio of the sample matrix's estimation variance to the target matrix's misspecification bias. This guarantees $\hat{\Sigma}_{LW}$ is well-conditioned while mitigating the impact of spurious correlations.

3.3 Optimization

The portfolio weights (w) are calculated via a constrained mean-variance optimization framework. To satisfy disciplined convex programming rules, the weight vector is decomposed into non-negative long (w^+) and short (w^-) parts, where that $w = w^+ - w^-$. The objective maximizes the expected portfolio return ($\mu^T w$). The problem is formulated using the CVXPY modeling language (Diamond and Boyd, 2016) and solved using the CLARABEL interior-point optimizer (Goulart and Chen, 2024):

$$\begin{aligned} & \max_{w^+, w^-} \mu^T (w^+ - w^-), \quad \text{s. t.} \\ (1) \quad & \sqrt{w^T \Sigma w} \leq 0.04, \quad (2) \quad 1^T (w^+ + w^-) \leq 3.0, \quad (3) \quad w^+, w^- \leq \frac{1}{N * 0.1}, \\ (4) \quad & 1^T w^+ \geq 1.0, \quad (5) \quad |1^T w| \leq \frac{1^T (w^+ + w^-)}{5} \end{aligned}$$

where μ is the expected return vector, Σ is the Ledoit-Wolf covariance matrix, N is the number of stocks, and 1 is a vector of ones. The constraints are set to ensure a balanced portfolio. Because a full grid search across all parameters was not computationally feasible, I focused the validation testing specifically on the target volatility as described below. The other constraints were fixed to reasonable values to ensure diversification and control leverage. These can most likely be improved.

1. Target Volatility Constraint (4%): Cap risk to maximize the Sharpe ratio. To determine this parameter, I ran a validation test using the same period employed in the hyperparameter tuning phase, evaluating annual volatility levels of [0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08]. The highest Sharpe ratio centered around 0.04, which was subsequently fixed for the final out-of-sample test.
2. Maximum Gross Exposure: Limited to 300% to prohibit the use of outsized leverage.
3. Position Concentration: Caps weights at $\frac{1}{N * 0.1}$, to enforce diversification, ensuring at least 10% of the stocks are held long when combined with (4).
4. Minimum Long Exposure: Ensures the sum of all long positions is strictly greater than or equal to 100%, guaranteeing the strategy is fully invested.
5. Net Exposure Boundary: This guarantees that neither the long nor the short leg can ever exceed 1.5 times the other, maintaining a balance between the long and short leg.

Finally, to ensure computational tractability and prevent matrix dimensional explosions, the investable universe (N) is dynamically capped at 2,000 equities each month. If the prediction universe exceeds this limit, the model selects the 1,000 stocks with the highest predicted returns and the 1,000 stocks with the lowest predicted returns, targeting the most likely candidates for construction.

References

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

Diamond, S., & Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(1), 2909–2913.

Goulart, P. J., & Chen, Y. (2024). Clarabel: An interior-point solver for conic programs with quadratic objectives. *arXiv preprint arXiv:2405.12762*.

Hellum, O., Jensen, T. I., Kelly, B. T., & Pedersen, L. H. (2025). The power of the common task framework. *SSRN Working Paper 5242901*.

Ledoit, O., & Wolf, M. (2003). Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance*, 10(5), 603–621.