Advanced Programming 2025

Predicting and Allocating Across Fama–French Size Portfolios with Machine Learning

Final Project Report

Tomas Papuga
tomas.papuga@unil.ch
Student ID: 25111589

December 2025

Abstract

This project asks whether monthly return and volatility forecasts from simple machine-learning models can improve allocation across the ten Fama–French size portfolios (ME1–ME10) relative to an equal-weight portfolio and a market proxy. Using data from the Kenneth R. French Data Library, I construct a monthly panel combining size deciles, the Fama–French three factors and the risk-free rate. For each portfolio, predictors are based on 1–12 month return lags, realised volatility over 3, 6 and 12 months, and lagged factor returns, keeping the feature set small and interpretable.

For each decile I estimate Ridge regression and histogram-based gradient boosting models in a strictly out-of-sample expanding-window scheme to forecast next-month return and variance. Forecasts are assembled into ME1–ME10 panels and passed to a long-only mean–variance allocator with shrinkage covariance, turnover limits and transaction costs.

Out-of-sample R^2 values are small, and forecast-driven allocations deliver net performance very close to an equal-weight benchmark and a market proxy. The project provides a reproducible forecast-then-optimise pipeline and illustrates the practical limits of simple machine-learning signals for tactical allocation along the size dimension.

Keywords: asset pricing, Fama–French size portfolios, portfolio allocation, machine learning, ridge regression, gradient boosting

Contents

1 Introduction

The Fama–French size portfolios are ten value-weighted portfolios sorted by firms' market equity, from ME1 (smallest) to ME10 (largest). They are standard empirical assets in asset pricing and are widely used in tests of factor models and market efficiency.

I focus on these ten portfolios rather than individual stocks because they form a clean, well-documented dataset with a long history and low microstructure noise. This allows me to concentrate on the forecast-then-optimise pipeline and on overfitting control.

The central question of the project is:

> *Do machine-learning forecasts of next-month return and volatility on Fama–French size portfolios lead to better risk-adjusted portfolio performance compared to simple benchmarks?*

To answer this, I use a fully out-of-sample expanding-window evaluation that respects real-time information constraints and avoids look-ahead bias.

The workflow mirrors a simple quantitative investment process:

- download and clean the Fama–French size deciles and FF3 factors from the French Library;
- engineer features (lagged returns, realised volatility, factor lags);
- train two model classes per decile (Ridge and histogram-based Gradient Boosting);
- produce one-step-ahead return and volatility forecasts for ME1–ME10;
- feed forecasts into a constrained mean–variance allocator with shrinkage covariance;
- evaluate predictive accuracy and economic performance against equal-weight and market benchmarks.

The entire pipeline can be reproduced from a single entry point, `main.py`, which runs all stages from data download to benchmark plots.

2 Literature Review / Related Work

2.1 Fama–French Size Portfolios

The "Portfolios Formed on ME" dataset provides ten monthly size-sorted portfolios for U.S. stocks listed on NYSE, AMEX and NASDAQ. Returns differ in level, volatility and cyclicality across deciles. The accompanying Fama–French factors (Mkt–RF, SMB, HML) and the risk-free rate RF are also available. In this project the factors are used purely as predictors. The dataset is standard in the literature, free from survivorship bias and easy to download programmatically.

2.2 Return Predictability

Short-horizon predictability in equity returns is known to be weak. Linear models using lagged returns typically find only shallow autocorrelation, while realised volatility and lagged factor movements contain limited conditional information. Regularised models such as Ridge regression help deal with noise and collinearity among lags, and non-linear methods such as gradient boosting can capture modest interactions or asymmetries.

Given the modest cross-sectional dimension (ten portfolios) and the course context, I restrict attention to a transparent feature set and two standard model families, focusing on how they behave in a noisy environment rather than on maximising complexity.

2.3 Forecast-then-Optimise Allocation

In a forecast-then-optimise setup, expected returns and a covariance estimate feed into an allocation engine. A classical choice is mean–variance optimisation:

$$\max_{w} \ w^{\top}\hat{\mu}_t - \frac{\lambda}{2} w^{\top}\hat{\Sigma}_t w \quad \text{s.t. } w_i \geq 0, \ \sum_i w_i = 1, \ w_i \leq \bar{w}.$$

Even weak predictability can lead to distinct optimal weights when paired with a regularised covariance matrix. I adopt this framework because it is theoretically grounded, easy to implement and allows a clear link between forecast quality and portfolio outcomes.

3 Methodology

3.1 Data Description

The empirical analysis uses monthly data from the Kenneth R. French Data Library:

- **Size portfolios:** "Portfolios Formed on ME" (ME1–ME10, value-weighted, monthly returns in percent).
- **Risk factors:** the Fama–French three factors (Mkt–RF, SMB, HML) and the risk-free rate RF.

I work with monthly data rather than higher-frequency observations because the monthly series are long, clean and sufficient to support a walk-forward scheme that retrains 40 predictive models (10 deciles $\times$ 2 targets $\times$ 2 model classes).

A dedicated script, `download_ff.py`, downloads the size portfolios and factors, parses headers, converts returns to decimals, standardises dates and merges everything into the canonical file

`ff_size_deciles_with_ff3_monthly_decimals.csv`,

which contains, for each month and decile MEj, the realised portfolio return, the three Fama–French factors and the risk-free rate.

3.2 Approach

For each decile MEj, a feature matrix is constructed using `prepare_features_full.py`. The feature set includes:

- 12 lags of the portfolio's own return;
- realised volatility over 3, 6 and 12 months;

- one-period lags of the Fama–French factors;
- a strictly increasing monthly index.

Twelve lags cover a full year of return history while keeping the predictor dimension moderate. The three volatility windows capture short-, medium- and longer-horizon risk dynamics, and lagged factors provide simple proxies for market conditions. I intentionally avoid a large feature zoo to keep comparisons between Ridge and Gradient Boosting clear and to limit overfitting risk.

The forecasting layer uses two model families:

1. **Ridge regression** for returns and volatility, with time-series cross-validation to select the regularisation parameter.
2. **Histogram-based Gradient Boosting** for returns and volatility, with early stopping, moderate depth and a low learning rate.

More complex models (deep networks, very large ensembles) are not considered because the sample is limited in time, the cross-section is small and the main risk is overfitting to noise.

Forecasting follows a strictly out-of-sample expanding-window scheme. After an initial warm-up period of 240 months (about 20 years), for each month t in the evaluation period and each decile MEj:

1. the model is trained on all data up to $t-1$;
2. a one-step-ahead prediction at t is recorded;
3. the training window is expanded to include t.

An expanding window mimics an investor who accumulates data over time and maximises the effective sample size in later periods.

Per-decile forecasts are merged into month $\times$ decile panels via `build_lr_panel.py` (Ridge) and `build_gb_panels.py` (Gradient Boosting). The scripts enforce ME1–ME10 ordering, pivot long-format predictions into wide matrices and keep only months where all ten forecasts exist. Panels for predicted returns, predicted variances and realised returns are stored under `results/oos_panel_*`.

3.3 Allocation

The allocation step uses these panels as inputs to a long-only mean–variance optimiser. For each month t:

- predicted returns form expected returns $\hat{\mu}_t$;
- a covariance estimate $\hat{\Sigma}_t$ is computed from a 120-month rolling window of realised returns, with diagonal shrinkage and blending of predicted variances;
- the mean–variance direction $\hat{\Sigma}_t^{-1}\hat{\mu}_t$ is projected onto a capped simplex with $w_i \geq 0$, $\sum_i w_i = 1$ and $w_i \leq 0.40$.

The 120-month window yields reasonably stable covariance matrices while allowing for time variation. Shrinkage towards the diagonal and the use of model-implied variances stabilise $\hat{\Sigma}_t$ further and limit extreme weights.

A turnover cap of 20% per month is applied, and net returns are computed as

$$r_t^{\text{net}} = r_t^{\text{gross}} - 0.001 \times \text{turnover}_t^{\text{eff}},$$

with turnover defined as

$$\text{turnover}_t = \frac{1}{2} \sum_{j=1}^{10} |w_{j,t} - w_{j,t-1}|.$$

The explicit modelling of turnover and a 10 basis point cost per unit of turnover prevent the strategy from relying on extreme trading to generate small apparent gains.

3.4 Implementation

The pipeline is implemented as a collection of small Python scripts under `src/`, with the following structure:

- `src/dataset`: download and cleaning of the Fama–French data;
- `src/utils`: feature construction, panel-building utilities and forecast-summary tools;
- `src/models`: per-decile forecasting (Ridge and Gradient Boosting);
- `src/alloc`: allocation and performance evaluation;
- `src/benchmark`: benchmark construction, summary tables and plots.

A single driver `main.py` runs all stages sequentially: downloading data, preparing features, training LR and GB models (returns and volatility), summarising forecast metrics, building panels, running allocations, evaluating performance and generating benchmark outputs.

4 Results

4.1 Experimental Setup

All experiments follow the expanding-window design described in Section 3: the forecast for month t uses data only up to $t-1$, with the first 240 months used for initial training. The evaluation sample spans several decades and covers all ten portfolios ME1–ME10.

The forecasting scripts produce out-of-sample predictions for returns and variances for each decile and model class. The allocation scripts then compute portfolio weights, gross and net returns, turnover and summary statistics for:

- the Ridge-based strategy (LR);
- the Gradient Boosting-based strategy (GB);
- an equal-weight portfolio across ME1–ME10 (EW_10);
- a market proxy constructed from Mkt–RF plus RF (Market).

4.2 Forecasting Accuracy

For each decile ME1–ME10 and for both model classes, predictive accuracy is evaluated using

$$R^2 = 1 - \frac{\sum_t (y_t - \hat{y}_t)^2}{\sum_t (y_t - \bar{y})^2}, \quad \text{MAE} = \frac{1}{T}\sum_{t=1}^{T} |y_t - \hat{y}_t|, \quad \text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2}.$$

Per-decile metrics are saved in `results/reports_lr/`, `results/reports_gb/`, `results/reports_vol_lr/` and `results/reports_gb_vol/`, and summarised by `src/utils/summarise_forecast_metrics.py` into `results/forecast_metrics/forecast_metrics_summary.txt`.

Table 1 reports the average out-of-sample forecast metrics across ME1–ME10 for LR and GB. For returns, both models have small and slightly negative mean R^2, with GB achieving a less negative value and marginally lower MAE and RMSE. For volatility, GB attains a positive mean R^2 while LR delivers poorer R^2 but lower absolute error metrics. These differences are modest and do not translate into large performance gaps at the portfolio level.

Target	Model	Mean R^2	Mean MAE	Mean RMSE
Return	LR	-0.066	0.0408	0.0539
	GB	-0.025	0.0399	0.0529
Volatility	LR	-1.713	0.0065	0.0092
	GB	0.366	0.0133	0.0181

Table 1: Average out-of-sample forecast metrics across ME1–ME10 for LR and GB, based on `results/forecast_metrics/forecast_metrics_summary.txt`.

4.3 Portfolio Performance

Portfolio weights are computed each month via the long-only capped mean–variance rule

$$\max_{w} \; w^\top \hat{\mu}_t - \frac{\lambda}{2} w^\top \hat{\Sigma}_t w \quad \text{s.t.} \quad w_i \geq 0, \; \sum_i w_i = 1, \; w_i \leq 0.40,$$

using the covariance estimator described in Section 3. The resulting weights are saved as `weights_baseline.csv` (LR) and `weights_gb_mv.csv` (GB). Net returns incorporate turnover caps and transaction costs as specified earlier.

Table 2 summarises the main performance statistics for the LR and GB strategies and for the two benchmarks. Values are taken from `results/benchmark/benchmarks_summary.csv`.

Strategy	Mean (m)	Ann. Ret.	Ann. Vol.	Sharpe	Sharpe Low	Sharpe High	Sortino	Sortino Low	Sortino High	Max DD
LR (net)	0.011	0.142	0.175	0.764	0.534	1.003	1.100	0.736	1.562	-0.546
GB (net)	0.011	0.141	0.175	0.757	0.529	0.995	1.086	0.724	1.554	-0.535
EW_10	0.011	0.140	0.174	0.757	0.530	1.000	1.070	0.706	1.531	-0.527
Market	0.010	0.128	0.149	0.813	0.573	1.060	1.158	0.792	1.622	-0.503

Table 2: Performance summary from `results/benchmark/benchmarks_summary.csv`. Mean monthly return, annualised return and volatility, Sharpe and Sortino ratios with bootstrap confidence intervals, and maximum drawdown.

To compare the two machine-learning allocations more directly, Table 3 reports LR and GB side by side using `results/alloc_comparison/summary_lr_vs_gb.csv`. Returns and risk are extremely similar, while GB achieves substantially lower turnover.

Metric	LR baseline	GB MV	GB − LR
Mean monthly return	0.0112	0.0110	-0.0001
Annualised return	0.1424	0.1408	-0.0017
Annualised volatility	0.1753	0.1749	-0.0004
Sharpe ratio	0.7639	0.7573	-0.0066
Max drawdown	-0.5457	-0.5346	0.0111
Avg. turnover	0.5803	0.2308	-0.3494
Avg. turnover (after limit)	0.1992	0.1633	-0.0359

Table 3: LR vs GB allocation metrics from `results/alloc_comparison/summary_lr_vs_gb.csv`. "GB − LR" is the difference between the GB mean–variance allocator and the LR baseline.

4.4 Diebold–Mariano Tests

To assess whether performance differences are statistically meaningful, Diebold–Mariano tests are run on the net-return series. Comparisons include LR versus EW_10 and Market, GB versus EW_10 and Market, and GB versus LR. Results from `results/benchmark/benchmarks_tests.csv` appear in Table 4.

Comparison	DM Statistic	p-value
LR_net vs EW_10	0.85	0.39
LR_net vs Market	1.53	0.13
GB_net vs EW_10	0.75	0.45
GB_net vs Market	1.44	0.15
GB_net vs LR_net	-0.52	0.60

Table 4: Diebold–Mariano tests from `results/benchmark/benchmarks_tests.csv`. Positive values favour the first strategy. None of the p-values are below conventional significance thresholds.

Across all comparisons, the statistics are small and p-values well above conventional significance levels, indicating that observed differences are compatible with sampling noise.

4.5 Visualisations

The cumulative performance of the LR, GB and benchmark portfolios is summarised in Figure 1 in Appendix A. The paths are highly correlated, consistent with the summary statistics and Diebold–Mariano tests, and support the conclusion that forecast-driven allocation does not dramatically change long-run outcomes relative to simple benchmarks.

5 Discussion

The results show that forecast-driven allocation offers only limited economic gains when applied to size-sorted portfolios. Monthly ME1–ME10 returns contain little short-horizon predictability, so both linear and non-linear models face the same constraint that lagged monthly variables provide weak signals in a noisy environment. Out-of-sample R^2 values are small, and the LR and GB allocations behave similarly to a disciplined equal-weight strategy once turnover limits and costs are imposed.

Several modelling choices were aimed at avoiding overfitting: regularised Ridge instead of unpenalised regression, Gradient Boosting with conservative hyperparameters, strict expanding-window evaluation, and the inclusion of turnover limits and transaction costs. The empirical evidence is consistent with these choices working as intended: the flexible GB model does not collapse out-of-sample relative to Ridge, both strategies resemble the equal-weight benchmark, and Diebold–Mariano tests do not detect statistically significant performance differences.

The findings are in line with theoretical expectations. If simple lag-based features and standard machine-learning models could systematically predict monthly equity returns, such opportunities would likely be arbitraged away. The negative or small R^2 values and benchmark-like performance therefore reflect the underlying difficulty of the problem rather than a flawed implementation.

Limitations include the simplicity of the feature set (return lags, realised volatility and factor lags), which omits macroeconomic, cross-sectional, sentiment or high-frequency information, and the reliance on mean–variance allocation with a blended diagonal covariance, which does not fully capture joint tail behaviour or model uncertainty. More flexible allocation approaches could alter the weight paths, especially in turbulent periods.

Overall, the project shows that machine-learning predictions can be integrated into a realistic investment pipeline, but that basic signals alone do not materially outperform standard benchmarks on size-sorted portfolios.

6 Conclusion and Future Work

6.1 Summary

This project examined whether machine-learning forecasts of monthly returns and volatility for the Fama–French size portfolios can improve portfolio allocation relative to simple benchmarks. Using an out-of-sample expanding-window design, separate Ridge and Gradient Boosting models were trained for each decile to predict next-month returns and risk. The resulting forecasts were assembled into ME1–ME10 panels and passed to a long-only capped mean–variance allocator with shrinkage covariance and transaction-cost constraints.

The main design principles were to use well-understood, regularised models; keep the feature set small and interpretable; and evaluate all decisions in a portfolio context that includes turnover and costs. Within this framework, statistical predictability at the monthly horizon remains limited, but the signals are strong enough to influence portfolio weights. Both Ridge and Gradient Boosting deliver performance broadly comparable to equal weighting and the market proxy, with only small differences in volatility, drawdowns and turnover, and no statistically significant dominance over the benchmarks.

6.2 Future Directions

Several extensions could be explored:

- **Richer features:** add macroeconomic indicators, cross-sectional signals (value, momentum), sentiment measures or higher-frequency volatility estimates.
- **Alternative models:** test more expressive models (e.g. recurrent networks, larger tree ensembles, Bayesian approaches) under the same strict walk-forward design.
- **Multi-horizon forecasting:** jointly predict returns and risk over multiple horizons and link forecasts to dynamic risk-budgeting rules.

- **Robust allocation:** replace or complement mean–variance optimisation with robust or Bayesian methods and more flexible turnover controls.
- **Broader universes:** extend beyond size-sorted portfolios to individual stocks, sectors or multi-asset universes to see whether richer cross-sectional variation yields stronger gains.

References

1. Banz, R. W. (1981). The relationship between return and market value of common stocks. *Journal of Financial Economics*, 9(1), 3–18.

2. Fama, E. F. and French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1), 3–56.

3. Gu, S., Kelly, B., and Xiu, D. (2020). Empirical asset pricing via machine learning. *Review of Financial Studies*, 33(5), 2223–2273.

4. Ledoit, O. and Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2), 365–411.

5. Markowitz, H. (1952). Portfolio selection. *Journal of Finance*, 7(1), 77–91.

6. Kenneth R. French Data Library. Fama/French factors and research portfolios ("Portfolios Formed on ME" and "Fama/French 3 Factors", monthly data), accessed 2025. Available at `https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html`.

A Additional Figures

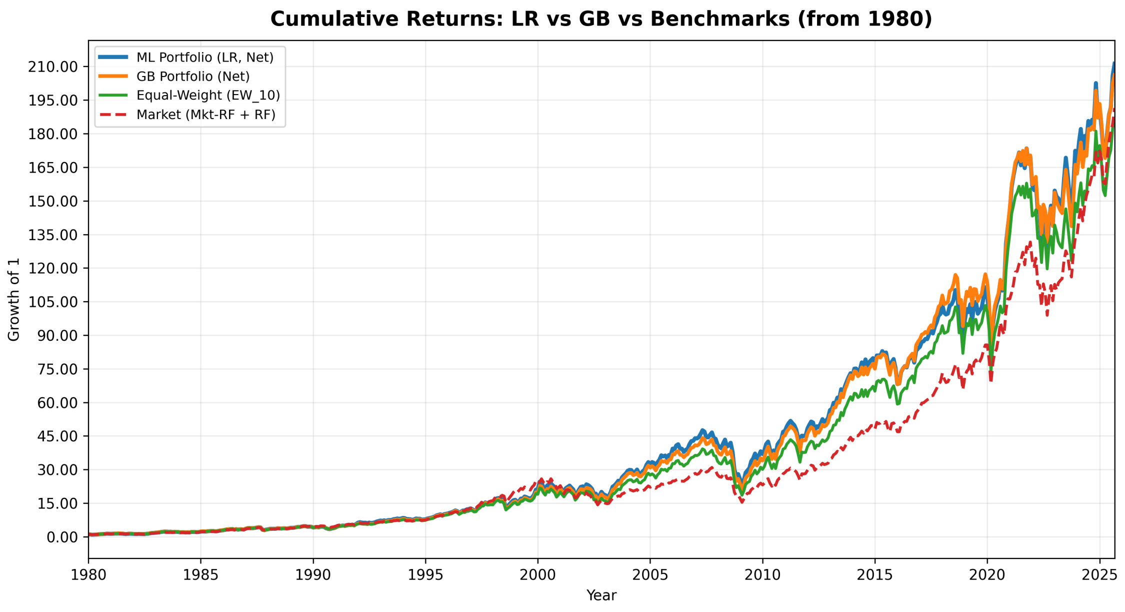

Figure 1: Cumulative returns of the LR, GB and benchmark portfolios (1980–2025).

B Code Repository

GitHub Repository: `https://github.com/Tommy7-8/Data-Science-Project`

The repository is organised as follows:

- `src/dataset`: download and cleaning of the Fama–French data.
- `src/utils`: feature construction, panel-building and forecast-summary utilities.
- `src/models`: training scripts for Ridge and Gradient Boosting models.
- `src/alloc`: allocation, performance evaluation and turnover handling.
- `src/benchmark`: benchmark construction, summary tables and plots.
- `results/`: all out-of-sample predictions, allocation weights, performance series and summaries (including `forecast_metrics_summary.txt` and benchmark files).

To reproduce the main results from scratch, from the project root:

1. Create a virtual environment, activate it and install dependencies:

   ```
   python -m venv .venv
   .\.venv\Scripts\activate
   pip install -r requirements.txt
   ```

2. Run the full pipeline:

   ```
   python main.py
   ```

   This single command executes the complete workflow: data download, feature engineering, model training, forecast-metric summarisation, portfolio allocation, evaluation and benchmark comparisons.