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 investigates whether monthly return and volatility forecasts generated by machine-learning models can improve portfolio allocation across the ten Fama–French size portfolios (ME1–ME10) relative to simple benchmarks such as an equal-weight portfolio and a market proxy. Using data from the Kenneth R. French Data Library, I construct a monthly panel that combines the ten size deciles, the Fama–French three factors and the risk-free rate. For each portfolio, I engineer predictors based on 1–12 month return lags, rolling realised volatility over 3, 6 and 12 months, and lagged factor returns; these choices deliberately focus on a small, interpretable feature set that captures basic time-series dynamics without exploding the dimensionality.

Two model families are estimated in a strictly out-of-sample expanding-window scheme: Ridge regression with time-series cross-validation and histogram-based gradient boosting with early stopping. Ridge provides a regularised linear benchmark that handles correlated lags, while gradient boosting offers a flexible but still relatively robust non-linear alternative. Separate models for each decile produce one-month-ahead return and variance forecasts, which are aggregated into ME1–ME10 prediction panels. These panels feed a long-only mean–variance allocator with shrinkage covariance, turnover limits and transaction costs—design choices intended to stabilise the weights and approximate realistic trading frictions rather than maximise in-sample Sharpe ratios.

The results show that forecast-driven allocation offers some structure but only limited economic gains on size-sorted portfolios. Predictive R^2 values are small, and both model classes deliver net performance close to an equal-weight benchmark and a market proxy. The project demonstrates a clean, reproducible forecast-then-optimise pipeline and clarifies the practical limits of simple machine-learning signals in tactical allocation across the size dimension.

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

Contents

1 Introduction **3**

2 Literature Review / Related Work **4**
2.1 Fama–French Size Portfolios 4
2.2 Return Predictability 4
2.3 Forecast-then-Optimise Allocation 4

3 Methodology **5**
3.1 Data Description 5
3.2 Approach 5
3.3 Allocation 7
3.4 Implementation 7

4 Results **8**
4.1 Experimental Setup 8
4.2 Forecasting Accuracy 8
4.3 Portfolio Performance 9
4.4 Diebold–Mariano Tests 10
4.5 Visualisations 11

5 Discussion **11**

6 Conclusion and Future Work **12**
6.1 Summary 12
6.2 Future Directions 13

References **14**

A Additional Figures **15**

B Code Repository **15**

1 Introduction

The Fama–French size portfolios consist of ten value-weighted portfolios sorted by firms' market equity, ranging from ME1 (smallest stocks) to ME10 (largest stocks). These size-sorted portfolios are foundational empirical assets and are widely used in studies of asset pricing, market efficiency and systematic factor strategies. They serve as building blocks for factor models, tests of market efficiency and hedge-fund replication strategies.

In this project I focus on these ten textbook portfolios rather than on individual stocks because they offer a clean, well-documented dataset with a long history and relatively low microstructure noise. This allows me to concentrate on the design of the forecast-then-optimise pipeline and on overfitting control, instead of dealing with idiosyncratic data issues at the single-stock level.

While most academic work focuses on explaining the cross-sectional variation in the long-run average returns of these portfolios, investors may also be interested in whether short-horizon predictions of their returns and volatility can be used to improve tactical allocation. The central question of this project is therefore:

> *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?*

Answering this question requires a fully out-of-sample procedure that respects real-time information constraints. I therefore adopt a strictly expanding, walk-forward evaluation and avoid any cross-sectional look-ahead or re-use of future observations in training.

The project follows a workflow that mirrors how a simple quant investment process would actually operate:

- download and clean the Fama–French size deciles and FF3 factors from a standard academic source to ensure data quality;
- engineer features (lagged returns, realised volatility and factor lags) that are simple enough to interpret but rich enough to encode basic time-series structure;
- train two types of models per decile—Ridge regression as a linear, regularised benchmark and histogram-based gradient boosting as a non-linear alternative;
- produce return and volatility forecasts one month ahead for ME1–ME10;
- aggregate predictions into panels and feed them into a mean–variance allocator with shrinkage covariance, long-only weights and turnover control;
- evaluate both statistical predictive accuracy and economic performance relative to standard benchmarks (equal-weight and market).

The project is designed to be fully reproducible: the entire pipeline can be run from a single entry point, `main.py`, which orchestrates all stages from data download to benchmark plots.

The remainder of the report is organised as follows. Section 2 reviews the relevant background and related work. Section 3 describes the data, methodology and implementation, explicitly motivating the main design choices. Section 4 presents empirical results on forecasting and portfolio performance. Section 5 discusses the findings and limitations, and Section 6 concludes with directions for future work.

2 Literature Review / Related Work

2.1 Fama–French Size Portfolios

The "Portfolios Formed on ME" dataset from the French Library provides ten monthly size-sorted portfolios dating back several decades. These portfolios are value-weighted within each size bucket and cover U.S. common stocks listed on the NYSE, AMEX and NASDAQ. ME1 contains the smallest firms; ME10 contains the largest. Their returns differ not only in level but also in volatility, cyclicality and exposure to macroeconomic conditions.

The accompanying Fama–French three factors consist of the market factor (Mkt–RF), the size factor (SMB) and the value factor (HML), alongside the risk-free rate RF. In this project, these factors function purely as predictors rather than as pricing factors. I use this specific dataset because it is widely used in the literature, free from survivorship bias and easy to obtain programmatically, which aligns with the goals of an applied programming project.

2.2 Return Predictability

Short-horizon predictability in equity returns is notoriously low. Linear time-series models using lagged returns often find only shallow autocorrelation, while predictors such as realised volatility and lagged factor movements may contain modest conditional information. Regularised models such as Ridge regression mitigate noise and collinearity among lags, whereas non-linear methods like gradient boosting can in principle capture structural breaks, interactions or asymmetric effects.

Given the course context and the modest cross-sectional dimension (ten portfolios), I deliberately restrict attention to a relatively small and transparent feature set and to two well-understood model families. This choice reflects a trade-off: rather than chasing marginal gains from very complex models, I focus on understanding how basic machine-learning tools behave in a noisy, low-predictability environment.

2.3 Forecast-then-Optimise Allocation

In a forecast-then-optimise design, expected returns and a covariance estimate are fed into an allocation engine. A standard 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 produce distinct optimal weights when combined with a carefully regularised covariance matrix.

I adopt this classical mean–variance framework because it connects directly to the theoretical literature, is easy to explain and implement, and allows me to isolate the effect of forecasting quality from more exotic allocation rules. More advanced approaches (e.g., robust optimisation) are intentionally left for the discussion and future work.

3 Methodology

3.1 Data Description

The empirical analysis is based on monthly data from the Kenneth R. French Data Library. Two datasets are used:

- **Size portfolios:** the "Portfolios Formed on ME" file provides the ten value-weighted size portfolios ME1–ME10. Returns are expressed in percent and cover a long sample of U.S. equity markets.
- **Risk factors:** the Fama–French three factors (Mkt–RF, SMB, HML) and the risk-free rate RF.

I work with *monthly* data rather than daily or weekly observations for two reasons. First, the French library provides long, clean monthly series, which simplifies data handling. Second, the pipeline trains and re-trains 40 separate predictive models (10 deciles $\times$ 2 targets $\times$ 2 model classes) in a walk-forward loop; at higher frequencies the computational cost and storage requirements would grow substantially.

The data layer transforms the raw files into a single, analysis-ready monthly panel. A dedicated script, `download_ff.py`, downloads the size portfolios and factors, parses the French-style headers, converts returns to decimals, standardises dates and merges everything into one canonical file:

`ff_size_deciles_with_ff3_monthly_decimals.csv`.

Using a single canonical file reduces the risk of misalignment between different intermediate datasets.

This panel contains, for each month and for each decile MEj:

- the portfolio's realised return;
- the three Fama–French factors;
- the risk-free rate.

Data quality issues mainly concern legacy encodings, non-standard headers and the need to align calendars across files. These are handled by a shared loader that normalises column names, strips byte-order marks (BOMs) and converts numerics safely. I chose to centralise all parsing logic in one place to avoid silent inconsistencies when new scripts are added.

3.2 Approach

For each decile MEj, a separate feature matrix is constructed using the script `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 clean, strictly increasing monthly index.

The choice of 12 lags is a compromise: it covers one full year of return history, which is economically meaningful, while keeping the number of predictors moderate and reducing the risk of overfitting in a relatively small sample. The 3-, 6- and 12-month volatility windows are intended to capture short-, medium- and longer-horizon risk dynamics without introducing a large family of overlapping horizons. Lagged factors are included because they provide simple, widely-used proxies for broad market conditions.

I deliberately avoid a very large feature zoo (e.g., many technical indicators or cross-sectional characteristics) because the main goal is to understand how a controlled, low-dimensional feature set interacts with different model classes and the allocator. Keeping the feature space small also makes it easier to compare Ridge and gradient boosting on equal footing.

The forecasting layer uses two families of models:

1. **Ridge regression** for returns and volatility, estimated via a standardised pipeline with time-series cross-validation to select the regularisation parameter. Ridge is used because it is simple, handles collinearity across lags and makes overfitting control explicit through the penalty parameter.

2. **Histogram-based gradient boosting** for returns and volatility, with early stopping, moderate depth and a low learning rate. Gradient boosting is chosen as a non-linear benchmark that can capture interactions and non-linearities, while still being relatively fast and stable on medium-sized tabular datasets.

I did not include more complex models such as deep neural networks or large random forests, because the dataset is not very large in time, the number of assets is small (ten deciles) and the main risk is overfitting to noise rather than underfitting. The focus is therefore on robust baselines rather than on maximising the complexity of the learner.

Forecasting is performed in a strictly out-of-sample expanding-window walk-forward scheme. After an initial warm-up period of 240 months (approximately 20 years) used only for estimation, forecasting proceeds in a walk-forward expanding-window scheme. For each month t in the evaluation period and for each decile MEj:

1. the model is trained on all available data up to $t-1$ (starting from the first month);

2. a one-step-ahead prediction of the return (or variance) at t is recorded;

3. the training window is expanded to include t before moving on.

I use an expanding window rather than a rolling window because this mimics a realistic learning process where an investor never discards old data, and because it maximises the effective sample size in later periods. This choice is particularly important given the weak predictability of monthly returns: discarding observations would further increase estimation noise.

Per-decile forecasts are then merged into month $\times$ decile panels using `build_lr_panel.py` for Ridge and `build_gb_panels.py` for gradient boosting. These scripts enforce ME1–ME10 ordering, pivot long-format predictions into wide matrices and keep only months where all ten forecasts exist. The decision to work with wide panels simplifies the allocation step, where vector and matrix operations are more natural.

Panels are produced for predicted returns, predicted variances and realised returns and stored under `results/oos_panel_*`. Keeping all intermediate panels on disk makes it easy to diagnose problems (e.g., misalignment) without re-running the full pipeline.

3.3 Allocation

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

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

The 120-month (ten-year) window for the covariance estimate balances two competing concerns: short windows react faster to regime changes but are extremely noisy; long windows are stable but may ignore structural breaks. Ten years is a pragmatic compromise that yields reasonably stable covariance matrices while still allowing for some time variation.

Shrinkage towards the diagonal and blending in the model-implied variances are used to stabilise $\hat{\Sigma}_t$ further and to reduce the impact of spurious correlations that often arise in small samples. Without these steps, mean–variance optimisation can produce extreme, unstable weights that are clearly unrealistic in practice.

Weights are constrained to be long-only, fully invested and capped at 40% per decile. These constraints are motivated by two considerations: they reflect typical restrictions faced by real-world long-only managers, and they prevent the optimiser from concentrating on a single decile based on small differences in noisy forecasts.

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

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

I explicitly model turnover and transaction costs because forecast-driven strategies can easily end up overtrading. The 10 basis point cost per unit of turnover is not meant to be a precise estimate of trading costs, but rather a simple, conservative penalty that prevents the optimiser from exploiting tiny, unstable signals.

3.4 Implementation

The pipeline is implemented as a collection of small Python scripts under `src/`, each with a clear single purpose. The repository is organised into five folders:

- `src/dataset`: download and cleaning of the Fama–French data;
- `src/utils`: feature construction, panel-building utilities and a summary script (`summarise_forecast_metrics.py`) that aggregates per-decile forecast reports into a single text file;
- `src/models`: per-decile forecasting (Ridge and Gradient Boosting);

- `src/alloc`: allocation and performance evaluation;
- `src/benchmark`: benchmark construction, summary tables and plots.

I chose this modular structure to mirror the logical stages of a typical data-science workflow (data, features, models, allocation, benchmarks). This makes it easier for a reader to inspect individual components in isolation and to re-run only a subset of the pipeline when debugging.

A single driver, `main.py`, sequentially runs all stages: downloading data, preparing features, training LR and GB models (for returns and volatility), summarising forecast metrics, building panels, running allocations, evaluating performance and generating benchmark outputs. Using one entry point instead of many shell commands reduces the scope for user error and makes the project easy to reproduce from scratch.

Practical implementation challenges include managing slow walk-forward training (mitigated by simplified feature selection and constrained hyperparameter grids), handling CSV parsing overhead (addressed by a dedicated loader) and enforcing strict time-series alignment across all panels.

4 Results

4.1 Experimental Setup

All experiments are conducted under a strictly out-of-sample expanding-window scheme: the forecast for month t uses data only up to $t-1$. The first 240 months (approximately 20 years) form an initial training window; out-of-sample evaluation starts after this warm-up period. The evaluation sample spans several decades and covers all ten size portfolios ME1–ME10.

The forecasting scripts generate out-of-sample predictions for returns and variances for each decile and for both model classes. 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).

The equal-weight portfolio is used as a benchmark because it is simple, requires no forecasting and is surprisingly hard to beat in practice. The market proxy provides a reference for a broad equity exposure. Comparing LR and GB against these two benchmarks helps to answer whether the extra modelling effort is justified once costs and constraints are taken into account.

4.2 Forecasting Accuracy

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

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

These metrics are chosen to capture complementary aspects of forecasting quality. R^2 measures performance relative to a simple constant-mean benchmark, while MAE and RMSE quantify the typical absolute and squared errors. Using all three reduces the risk of overinterpreting R^2 in a setting where the signal is weak and noise is large.

Because monthly returns are noisy, absolute R^2 values are small, but the metrics reveal consistent patterns: volatility forecasts track periods of market stress, small-cap deciles are harder to predict, and gradient boosting captures some non-linear structure without consistently outperforming Ridge at the monthly horizon. Per-decile metrics are saved in `results/reports_lr/`, `results/reports_gb/`, `results/reports_vol_lr/` and `results/reports_gb_vol/`. A separate utility, `src/utils/summarise_forecast_metrics.py`, merges these text reports into a compact summary file, `results/forecast_metrics/forecast_metrics_summary.txt`, which compares LR and GB side by side for both return and volatility forecasts.

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 a poorer R^2 but lower absolute error metrics; these differences, however, 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`.

From an overfitting perspective, the key observation is that the more flexible GB model does not collapse out-of-sample relative to Ridge: the differences in metrics are modest and consistent with both models trying to extract weak signals from very noisy series.

4.3 Portfolio Performance

Given predicted returns and variances, portfolio weights are computed each month via a long-only capped mean–variance rule:

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

The covariance estimate $\hat{\Sigma}_t$ combines a 120-month rolling sample covariance of realised returns, shrinkage toward the diagonal and a blended diagonal based on predicted variances with a small stability floor. The resulting weights are saved as `weights_baseline.csv` (LR) and `weights_gb_mv.csv` (GB).

Monthly gross returns are computed as

$$r_t^{\text{gross}} = \sum_{j=1}^{10} w_{j,t} \, r_{j,t},$$

and turnover is defined as

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

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

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

The explicit focus on *net* returns and on turnover is intentional: a forecasting model that looks good in terms of gross performance but requires extreme trading is unlikely to be attractive in practice. By reporting both return and turnover statistics, I can distinguish between models that add value and models that simply trade a lot.

Table 2 summarises the main performance statistics for the LR and GB strategies and for two benchmarks: an equal-weight portfolio across ME1–ME10 and a market proxy (Mkt–RF + RF). The 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 focus directly on the relative behaviour of the two machine-learning allocations, Table 3 reports a side-by-side comparison of LR and GB using the dedicated allocation-comparison summary file, `results/alloc_comparison/summary_lr_vs_gb.csv`. The statistics confirm that 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.

From a design perspective, these results justify the choice of using GB mainly as a stability-improving alternative rather than as a way to chase much higher returns: GB delivers almost the same net performance as LR but trades considerably less.

4.4 Diebold–Mariano Tests

To assess whether these performance differences are statistically meaningful, Diebold–Mariano tests are run on the net-return series. Comparisons include LR versus EW_10 and Mar-

ket, GB versus EW_10 and Market, and GB versus LR. The results, taken from the file `benchmarks_tests.csv` in the `results/benchmark` folder, 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.

I use Diebold–Mariano tests because they are a standard way to compare forecast-based strategies over time; they explicitly account for the serial correlation of forecast errors. Across all comparisons, the signs and magnitudes of the DM statistics provide additional context. Positive values (e.g., LR_net vs. EW_10 or Market, and GB_net vs. EW_10 or Market) indicate that the forecast-driven strategies deliver slightly higher average net returns relative to the benchmarks, while the small negative value for GB_net vs. LR_net reflects a marginal advantage for LR. However, in all cases the statistics are small in absolute value and the associated p-values are comfortably above conventional significance thresholds. This implies that the observed differences in economic performance are driven by sampling noise rather than persistent forecasting advantages.

4.5 Visualisations

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

5 Discussion

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

A central design goal of the project was to avoid overfitting. This explains several modelling choices: the use of Ridge (a regularised linear model) instead of an unpenalised regression, the use of gradient boosting with conservative hyperparameters and early stopping, the strict expanding-window evaluation, and the inclusion of turnover limits and transaction costs in the allocator. The empirical evidence is consistent with these choices working as intended: the

flexible GB model does not blow up 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 broadly in line with theoretical expectations. If simple lag-based features and standard machine-learning models could systematically predict monthly equity returns, such opportunities would be quickly arbitraged away. In this sense, the negative or small R^2 values and the benchmark-like performance of the strategies are not signs of a flawed implementation, but rather reflections of the underlying difficulty of the problem.

A key limitation of the present setup is the simplicity of the feature set: return lags, realised volatility and factor lags. These variables capture basic dynamics but omit many potentially informative dimensions such as macroeconomic indicators, cross-sectional characteristics, sentiment measures or high-frequency volatility estimates. Likewise, the modelling choices—Ridge and gradient boosting—provide robustness and interpretability but do not exploit more advanced temporal architectures.

The allocation engine also relies on simplifying assumptions. Mean–variance optimisation with a blended diagonal covariance is pragmatic and stable, but it does not fully capture joint tail behaviour or model uncertainty. More flexible approaches—such as robust optimisation, Bayesian filtering or dynamic turnover constraints—could alter the resulting weight paths, especially in turbulent periods.

Overall, the project demonstrates that machine-learning predictions can be integrated into a clean and realistic investment pipeline, but that simple signals alone do not materially outperform standard benchmarks. The evidence supports a cautious interpretation of forecast-then-optimise methods when applied to size-sorted portfolios with basic features.

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 entirely out-of-sample expanding-window design, separate Ridge and gradient boosting models were trained for each decile to predict next-month returns and risk. These predictions were assembled into ME1–ME10 panels and fed into a long-only, capped mean–variance allocator with shrinkage covariance and realistic transaction-cost constraints.

The main methodological choices were guided by three principles: use well-understood, regularised models (Ridge and conservative GB) to control overfitting; keep the feature set small and interpretable; and evaluate all decisions in a realistic portfolio context that includes turnover and costs. Within this framework, the empirical results show that, although statistical predictability at the monthly horizon remains limited, the forecasted signals are sufficiently informative to influence portfolio weights. Both Ridge and gradient boosting deliver performance broadly comparable to equal weighting and the market proxy, with small differences in volatility, drawdowns and turnover. Neither model dominates standard benchmarks by a wide margin.

Overall, the evidence suggests that even weak but consistent predictive structure, when embedded within a carefully regularised allocation framework, can yield competitive but not dramatically superior portfolio outcomes. At the same time, the results reaffirm a central insight

from empirical finance: if simple lag-based models could reliably generate large improvements, such opportunities would be rapidly arbitraged away.

6.2 Future Directions

Several avenues for future work could extend and strengthen the analysis:

- **Richer features:** incorporate macroeconomic indicators, cross-sectional characteristics (such as value and momentum signals), sentiment measures or higher-frequency volatility estimates. This would test whether the limited gains observed here are due mainly to the feature set or to the inherent difficulty of predicting size-portfolio returns.
- **Alternative models:** explore more expressive models such as recurrent or convolutional neural networks, tree-based ensembles with richer interaction structures or Bayesian methods that explicitly model parameter uncertainty. Any such extension should be evaluated carefully for overfitting using the same strict walk-forward design.
- **Multi-horizon forecasting:** jointly predict returns and risk over multiple horizons and consider dynamic allocation rules that adjust risk exposure accordingly.
- **Robust allocation:** replace or complement mean–variance optimisation with robust or Bayesian portfolio construction, stress testing and more sophisticated turnover controls, to see whether allocation choices rather than forecast quality are the main bottleneck.
- **Broader universes:** extend the approach beyond size-sorted portfolios to cross-sections of individual stocks, sectors or multi-asset universes to test whether richer variation yields stronger economic 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

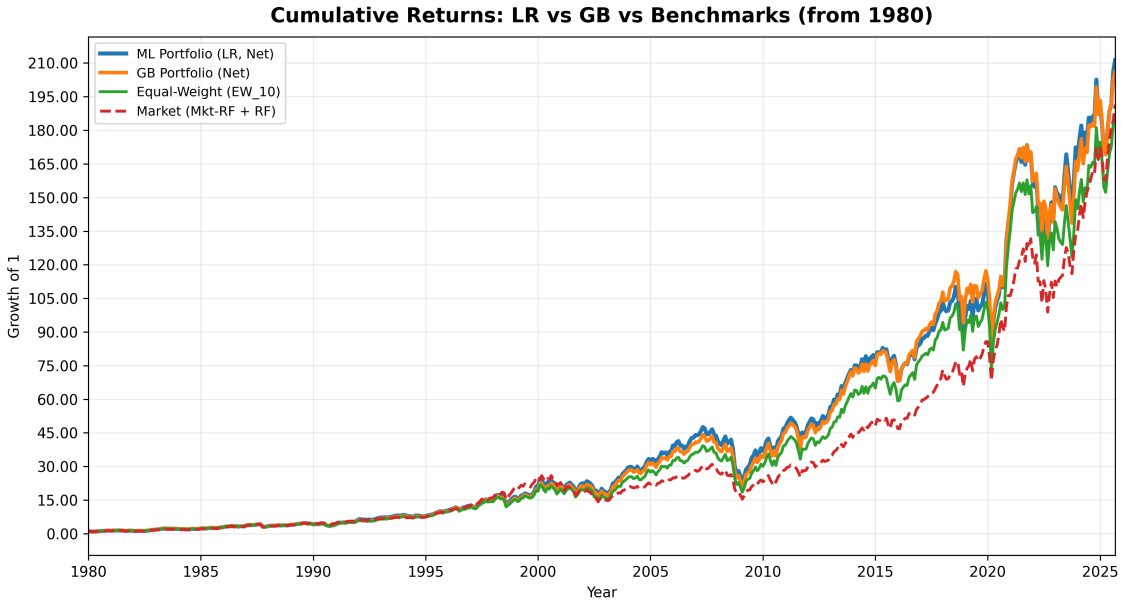

Figure 1: Cumulative returns of the LR, GB and benchmark portfolios (1980–2025).

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.