Advanced Programming 2025

Machine Learning–Based Stock Selection and Trading Strategy

Final Project Report

Marco D'Amico
marco.damico@unil.ch
Student ID: 25416579

December 13, 2025

Abstract

This project wants to understand whether a set of standard technical indicators can be used to predict short-horizon stock returns in the S&P 500 through a simple machine-learning framework. Using five years of daily data, I construct a cross-sectional dataset containing momentum measures, realized volatility, moving-average ratios, and the 14-day RSI, all evaluated monthly. I then train a range of classifiers—Logistic Regression, Random Forest, Gradient Boosting, and XGBoost—in an expanding-window walk-forward scheme to estimate the probability that each stock will have a positive next-month return.

The results indicate that predictive power is modest but non-zero, consistent with the high efficiency of the chosen equity market. Finally, a monthly balanced long top-decile strategy based on the best-performing model is constructed and compared to an equal-weight and a 6M momentum portfolios. Further analyses, such as subperiod comparisons, turnover estimation, and robustness checks across selected fractions and cost assumptions, confirm the limited economic value of purely technical, publicly available signals.

Keywords: data science, machine learning, probabilistic forecasting, stock selection, S&P 500, portfolio construction, momentum signals, backtesting

Contents

1 Introduction

Machine learning is becoming always more important in quantitative finance, especially for tasks such as cross-sectional return prediction and portfolio construction. Large equity markets such as the S&P 500 provide a good environment to study these methods because of their liquidity and efficiency. Indeed, according to the Efficient Market Hypothesis (Fama 1970), publicly available information should be reflected into prices almost immediately, making it difficult to exploit them in the short-term. Empirical studies show that technical and price-based predictive signals usually have low signal-to-noise ratios in large and liquid markets, and that even advanced ML models often produce only modest out-of-sample performance (Gu, Kelly, and Xiu 2020). This project wants to examine to what level simple ML techniques can generate economically meaningful information from publicly available signals in an efficient market.

The objective is to predict, for each stock and for each month-end, the probability that the next-month return will be positive. I set the problem as a binary classification exercise, where the target equals one when the next-month return is forecasted to be positive and zero otherwise. The dataset is composed by five years of daily prices and volumes for the current constituents of the S&P 500 index. This choice naturally introduces survivorship bias, as only firms that remained in the index as of today are analyzed, but it provides a clean and consistent panel for the modelling exercise. From the daily data, I construct a set of monthly technical features: short- and medium-term momentum (1M, 3M, 6M), realized volatility over the previous three months, price–moving-average ratios capturing short-term trend conditions, and the 14-day Relative Strength Index (RSI).

I included three families of machine learning classifiers in the modeling phase: logistic regression (with and without L2 regularization), tree-based ensemble models (Random Forest and Gradient Boosting), and XGBoost. Models are trained and evaluated using an expanding-window walk-forward procedure in which only information up to month $t-1$ is used to predict month t. This creates an out-of-sample sequence of predicted probabilities, which are evaluated using ROC–AUC, Brier score, and calibration plots. As expected, the predictive power is low but non-zero, which is consistent with prior findings that technical signals contain limited exploitable information in highly efficient markets (Gu, Kelly, and Xiu 2020).

To assess the usefulness of the predictions, I build a simple long strategy rebalanced monthly: each month stocks are ranked by their predicted probability of a positive return, and an equal-weighted portfolio is formed using the top decile. I then compare its performance against two common benchmarks: an equal-weight portfolio of the full S&P 500 and a 12–1 momentum strategy. Performances are evaluated using annualized return, volatility, Sharpe ratio, and maximum drawdown. I also study subperiod behaviours (pre-2022 and post-2022), turnover estimation, and how results change when varying both the selection fraction and trading costs.

Overall, the project highlights both the strengths and the limits of applying standard ML techniques to short-horizon stocks selection in liquid U.S. markets. The results show that while there is some predictive power, its economic value is constrained by market efficiency, model simplicity, and survivorship bias in the dataset.

The rest of the report is organized as follows. Section 2 reviews the main academic literature on return predictability, momentum, and ML in asset pricing. Section 3 describes the methodological setup, including the construction of the dataset, the feature engineering process, the modelling framework, and the walk-forward evaluation procedure. Section 4 presents the empirical results, covering both predictive performance and portfolio outcomes, together with robustness checks and subperiod analyses. Section 5 provides a discussion of the findings, their implications, and the main limitations of the approach. Finally, Section 6 concludes the report and outlines possible directions for future work.

2 Literature Review

For many years in the financial world it has been studied whether stock returns can be predicted from publicly available information. In a 1970 paper about the Efficient Market Hypothesis, Eugene Fama argues that asset prices should quickly reflect all available information, making it difficult for investors to earn abnormal returns (Fama 1970). Under this view, any predictable component in returns should be small and hard to exploit. Still, many other empirical studies analyzed patterns in returns that appear to contradict the strictest interpretation of market efficiency. One of the most known examples is momentum. In their influential study, Jegadeesh and Titman showed that stocks that performed well in the past often continue to overperform for several months, while past losers keep underperforming (Jegadeesh and Titman 1993). Yet, later research has shown that momentum results depend on details about the strategy implementation and can be significantly reduced once realistic transaction costs are considered, especially in large and highly liquid markets such as the S&P 500.

Beyond momentum, a wide range of return predictors has been proposed including firm size and valuation ratios such as book-to-market equity (Fama and French 1992). However, the number of "factors" has grown so much that avoiding statistical overfitting is nowadays a central topic. Harvey, Liu, and Zhu show that many factor models do not pass stricter testing standards, explaining how difficult it is to distinguish noise from real predictability (Harvey, Liu, and Zhu 2016).

In the last years, machine learning techniques became always more important in asset pricing, mainly because they can handle large sets of predictors and potential non-linear relationships. The work of Gu, Kelly, and Xiu shows that ML models can deliver small improvements in the cross-sectional forecasting of returns compared to traditional linear methods (Gu, Kelly, and Xiu 2020). They also emphasize that the economic gains are not impressive and sensitive to model design and data quality, and this is particularly evident in very liquid US stock markets.

This project fits into this context, but instead of relying on hundreds of firm characteristics, I focus on a small set of technical indicators computed from prices and volumes. The goal is not to identify a new market anomaly, but rather to study how a few standard machine-learning models behave in a realistic, low-signal environment, and to evaluate whether any weak predictive structure can translate into economic performance.

3 Methodology

3.1 Data Description

The dataset for this project is built from daily prices and volumes of the companies that are currently part of the S&P 500 index. I downloaded all data directly from Yahoo Finance, and for each stock I collected five years of daily observations from 01/11/2020 to 01/11/2025. Since I used today's S&P 500 list, the dataset naturally reflects survivorship bias: companies that were removed from the index during the sample period are not included. This is a limitation, but it keeps the cross-sectional universe clean and stable across all months, which is convenient for modeling.

From the raw Yahoo Finance output I only use two fields: the adjusted close price and the trading volume. Adjusted close already includes dividends and stock splits, so it is more reliable for computing returns and technical indicators. After downloading the data, I reshaped everything into a panel with one row per stock per date.

To move from daily to monthly frequency, I take the last trading day of each month and compute monthly returns as the percentage change from one month-end to the next. Some features used in the machine-learning models, such as realized volatility and RSI, require daily data, so the daily panel is kept to compute these indicators. All variables used to predict month

t returns are lagged properly, making sure that they only include information that would have been available before the end of month $t-1$, in order to avoid any form of look-ahead bias.

One small practical issue with Yahoo Finance is the format of ticker symbols. Certain stocks use a dot notation (e.g., "BRK.B"), but Yahoo replaces dots with hyphens (e.g., "BRK-B"). I standardize all tickers during preprocessing to keep them consistent throughout the entire pipeline. Aside from these minor details, Yahoo Finance offers data that are sufficiently clean and consistent for the purposes of this project.

3.2 Approach

In the modelling stage the objective is to estimate, for each stock at the end of the month, the probability that its return will be positive in the next month. This is set up as a cross-sectional binary classification problem. Each stock observation will be one if the next-month return is positive, and zero otherwise. These 0/1 labels are used only to train the models: once trained, the output will be a continuous probability between 0 and 1 that is used later to rank stocks and select the top 10%.

The features used for prediction come entirely from prices and volumes and are standard technical indicators among traders and quantitative analysts. Momentum (1M, 3M, 6M returns) measures whether a stock had an upward or downward trend in the past months. A three-month realized volatility is included as a simple measure of recent risk. The 50-day moving-average ratio compares the current price with a short-term trend, and the 14-day RSI tries to detect situations where a stock may be temporarily overbought or oversold. Most indicators are first computed using daily data (50-day moving average, RSI, and realized volatility). Then, it is kept only the last available daily value for each month. To keep the process realistic, all features used to predict month t rely only on information that was available before month-end t.

A small set of machine-learning models is used to explore different levels of complexity. Logistic regression (with and without L2 regularization) serves as a simple and interpretable baseline. Tree-based models—Random Forest and Gradient Boosting—are added because they can capture nonlinear relationships between features. XGBoost is also included, since it is widely used in practice for tabular datasets.

The models are trained using an expanding-window, walk-forward procedure. The first twelve months of data create an initial training period. After that, for each month t, the model is trained on all observations from the beginning of the dataset up to month $t-1$. Then, it predicts the probability of a positive return for month t. The amount of training data increases every month, allowing the models to learn from more history. Each prediction is made strictly out-of-sample, using only past information.

The models are evaluated through three tools. The ROC–AUC score is used to measure how well the probabilities separate positive and negative returns. The Brier score checks how close the predicted probabilities are to the actual outcomes. Finally, calibration plots compare the predicted probabilities with the true frequencies, helping to understand whether the models tend to be overconfident or underconfident. These metrics provide a clear picture of how reliable the probability forecasts are before they are used in the portfolio construction step.

3.3 Implementation

This section shows only the most important parts of the code, just to give a clear idea of how the project works.

I developed the project in Python and I divided it into several modules so that each part of the pipeline (data loading, feature construction, modelling, and backtesting) is handled in a separate file. This keeps the workflow clear and makes it easy to reproduce the results.

The main script, `main.py`, coordinates all steps. It first downloads price data if they are not already available, then builds the monthly feature dataset, runs all machine-learning models in

a walk-forward scheme, and finally computes portfolio performance.

Before running the models, the script checks whether the raw price data and the pre-built feature dataset already exist. If not, they are created. This avoids downloading or rebuilding the data every time the script is executed.

```
# load or download raw price data
if not os.path.exists("data/sp500_prices.csv"):
    print("Downloading price data...")
    download_prices(period="5y")
else:
    print("Price data already available.")

# load or build ML dataset (features + next-month returns)
if not os.path.exists("data/ml_dataset.csv"):
    print("Building feature dataset...")
    df_ml = build_ml_dataset()
else:
    print("ML dataset already exists, loading it.")
    df_ml = pd.read_csv("data/ml_dataset.csv", parse_dates=["Date"])
```

Listing 1: Loading or generating main datasets

The list of S&P 500 companies is loaded through a helper function that reads the ticker symbols from a CSV file and converts them into the Yahoo Finance format:

```
def sp500_tickers():
    """
    It loads the S&P500 tickers from the CSV (data/sp500_tickers.csv).
    The file needs a column called 'Symbol'.
    """
    path = Path("data/sp500_tickers.csv")
    df = pd.read_csv(path)
    # Yahoo uses '-' instead of '.' in tickers
    tickers = (
        df["Symbol"]
        .astype(str)
        .str.replace(".", "-", regex=False)
        .tolist()
    )
    # remove duplicates and sort
    return sorted(set(tickers))
```

Listing 2: Loading S&P 500 tickers

Once the feature dataset is ready, the models are trained using an expanding-window walk-forward function. For each month in the sample (starting from month 12), the model is refit on all past observations and then used to predict the next month:

```
# train each model in walk-forward scheme and store predictions/metrics
for name, model in MODELS.items():
    print(f"Training {name} ...")
    pred, auc, brier = walk_forward_predict(
        df_ml, FEATURES_BASE, model, start_idx=12
    )
    results.append({"Model": name, "AUC": auc, "Brier": brier})
    pred_store[name] = pred
```

Listing 3: Walk-forward training and predictions

After the probabilities are attached to the dataset, the portfolio strategies are tested. The top-decile long-only strategy is computed using:

```
# run ML long-only strategy with a 10% selection and 5 bps trading cost
top_frac = 0.10
cost = 0.0005
```

```
bt = backtest_long_top(df_pred, top_frac=top_frac, cost=cost)
# compute standard performance metrics (CAGR, Vol, Sharpe, Max DD)
perf = perf_table(bt)
print_df_nice(
    perf,
    title="PERFORMANCE TABLE: ML Long-Only vs Benchmarks",
    show_index=True,
)
```

Listing 4: Backtesting the long-only strategy

This modular structure makes it easy to extend the project (for example by adding new models or indicators) while keeping the logic simple and clear. Each part of the pipeline can be inspected and updated independently, which also helps when sharing the code or running it on another machine.

4 Results

4.1 Experimental Setup

The whole project has been created and tested in Python 3 and run on a standard CPU-only environment. The main libraries are `pandas` and `numpy` for data manipulation, `scikit-learn` for the machine-learning models and metrics, `yfinance` for downloading market data from Yahoo Finance, and `matplotlib` for the plots. I also use `XGBoost` to include a gradient-boosted tree model in the comparison. I have also used some helper packages such as `os` and `pathlib` for file and path handling, and `tabulate` to print nicely formatted tables in the console. The code is organised into separate modules for data loading, feature construction, modelling, portfolio backtesting, and reporting, with the main script `main.py` running the whole pipeline. The constants `FEATURES_BASE`, `TOP_FRACS`, `COSTS` and `SUBPERIODS` define, respectively, the feature set, the portfolio selection fractions, the trading-cost grid, and the sample splits used in the analysis. The main set of models is stored in the `MODELS` dictionary, including two Logistic Regression specifications (with and without L2 regularization, both wrapped in a `StandardScaler`–`LogisticRegression` pipeline), a Random Forest classifier ($n_$estimators $= 200$, maximum depth 5, minimum 5 samples per leaf), a Gradient Boosting classifier ($n_$estimators $= 200$, maximum depth 3), a Ridge classifier (again inside a scaling pipeline), and an XGBoost classifier (with 150 trees, depth 3, learning rate 0.1, subsample and column subsample equal to 0.8, and ℓ_2 regularization $\lambda = 1.0$). All models use a fixed random seed to ensure reproducibility. The models are tested in a strictly out-of-sample walk-forward scheme. Daily prices and volumes over the last five market years are aggregated to monthly data: of which the first twelve months are used as an initial training window, then at each month t the models are re-estimated using all data up to $t-1$ to predict the probability of a positive return in month t. This produces a panel of monthly cross-sectional forecasts, from which I compute ROC–AUC and Brier scores. The model with the highest AUC over the full out-of-sample period is then used in the portfolio step. Portfolio experiments are run at monthly frequency. The main long-only strategy selects the top fraction of stocks by predicted probability p_{up}, with `TOP_FRACS` set to $\{0.10, 0.20\}$, and applies per-leg trading costs taken from `COSTS`, namely $\{0, 5, 10\}$ basis points (0.00, 0.0005, 0.001 in return space). The results are then summarised through annualised return (CAGR), volatility, Sharpe ratio, and maximum drawdown, computed both on the full sample and on two subperiods (2019–2021 and 2022–2024). I also track portfolio turnover for the ML strategy and generate calibration plots and feature-importance charts (based on `matplotlib`) to better understand how well the probabilities are calibrated and which predictors drive the model.

4.2 Performance Evaluation

In Table 1 the out-of-sample accuracy of all the models is presented using ROC–AUC and Brier Score. AUC measures the model's ability to rank stocks by relative attractiveness, whereas the Brier Score evaluates the calibration of predicted probabilities. Overall, the differences across

Table 1: Out-of-Sample Performance Across Models

Model	AUC	Brier
Random Forest	0.4942	0.2561
Gradient Boosting	0.4913	0.2621
XGBoost	0.4886	0.2623
Ridge Classifier	0.4870	0.2522
Logistic (L2)	0.4866	0.2574
Logistic (no Reg.)	0.4866	0.2575

specifications are small: all models achieve an AUC between 0.4866 and 0.4942, slightly below the 0.50 threshold associated with random guessing. The best-performing model is Random Forest (AUC = 0.4942, Brier = 0.2561), closely followed by the Gradient Boosting and XGBoost models. The linear classifiers models (Logistic Regression with and without regularization, and the Ridge classifier) perform slightly worse but have very close results.

Moving from simple linear models to more flexible non-linear methods provides only very small gains in terms of predictive power, suggesting that the relationship between the technical indicators and next-month returns is weak and that any nonlinearities add limited incremental information. Also interpreting the Brier Score suggests the same conclusion: Brier Scores around 0.25–0.26 show that probability estimates are noisy and poorly calibrated.

This result is consistent with finance literature: simple technical indicators at a monthly frequency should have low predictive power (Goyal and Welch 2008; Green, Hand, and Zhang 2017), and machine-learning models can extract only limited signal from them (Gu, Kelly, and Xiu 2020). The portfolio construction is then optimized using the Gradient Boosting model.

Table 2 reports the main results of the long strategy with a portfolio created with top for 10% stocks compared with an equal-weight universe portfolio and a simple 6-month momentum strategy. All portfolio results incorporate a trading-cost assumption of 5 basis points per trade (0.0005 in return space), applied symmetrically on buy and sell transactions.

Table 2: Long-Only Performance: ML vs Benchmarks

Strategy	CAGR	Volatility	Sharpe	MaxDD
ML Top 10%	0.1177	0.1850	0.6365	-0.1274
Equal-Weight	0.1423	0.1714	0.8307	-0.1133
Momentum 6M Top 10%	0.2691	0.2328	1.1559	-0.1216

The ML Top 10% portfolio achieves an annualized return of 11.77% with a Sharpe ratio of 0.63. This is very close to the equal-weight benchmark (14.2% CAGR, Sharpe 0.83) and substantially below the 6-month momentum portfolio, which delivers 26.9% per year with a Sharpe of 1.15.

4.3 Visualizations

The calibration plot in Figure 1 compares predicted probabilities forecasts produced by the Gradient Boosting model with realised frequencies of positive next-month returns across probability buckets.

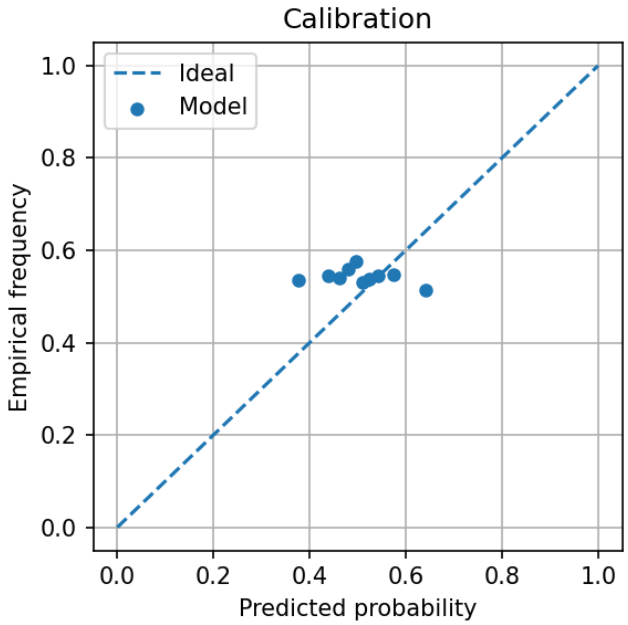

Figure 1: Calibration plot of the Gradient Boosting model.

If the model were well calibrated, the points would lie close to the 45-degree line: stocks assigned a probability of 0.6 of going up would in fact go up roughly 60% of the time, and so on. In practice, the points are almost flat: regardless of whether the model predicts 0.3, 0.5, or 0.6, the realized frequency remains around 0.50–0.55. This confirms that the model provides a weak ranking signal and that its probability scale is not meaningful.

Figure 2 plots cumulative returns for the ML Top 10% strategy, the equal-weight portfolio, and the 6-month momentum strategy.

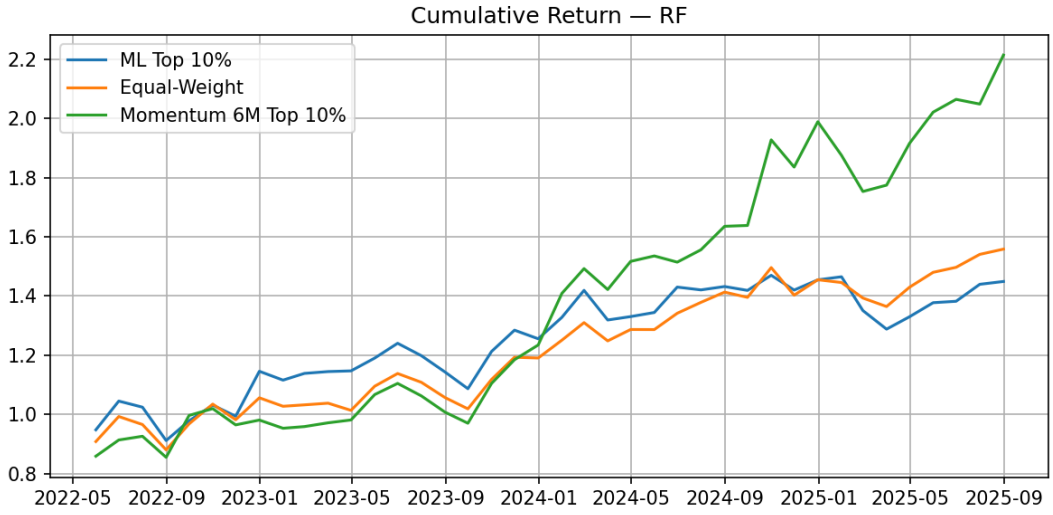

Figure 2: Cumulative returns for the ML Top 10% portfolio, the equal-weight benchmark, and the 6-month momentum strategy.

The ML strategy closely tracks the equal-weight benchmark over the whole period, while the momentum portfolio diverges upwards from mid-2023 onward and finishes with a much higher terminal wealth.

5 Discussion

The picture we can capture from the project is that across all machine-learning models, predictive accuracy remains low: with AUC values only slightly above 0.49, and weak probability calibration, with realised frequencies clustering around 50–55%. These patterns show that the models only extract a weak signal, but not one that is strong or stable enough to produce a meaningful separation between future winners and losers.

Accordingly, the ML Top 10% strategy performs very similarly to the equal-weight benchmark, and it is dominated by a simple 6-month momentum rule. This holds both in the full sample and in subperiods, and for different selection fractions and to the inclusion of different transaction costs.

These findings are fully consistent with the efficient markets theory: in large and liquid markets such as the S&P 500, publicly available information should be rapidly incorporated into prices.

The technical indicators used in this project are simple and widely monitored. Since they are easy to compute and trade on, quantitative investors and hedge funds already exploit these signals at scale.

The limited explanatory power observed in this project therefore reflects a market environment in which most easily accessible information has already been priced in.

6 Conclusion and Future Work

6.1 Summary

The project tries to evaluate whether machine-learning models trained on simple technical indicators can generate predictive power and economically meaningful portfolio performances in liquid U.S. equity markets.

It is clear that the models identify a very weak signal, but this signal does not translate into superior investment performance once realistic turnover and transaction costs are taken into account. These results highlight a central implication of market efficiency: when information is public, easy to access, and broadly used by investors, it is indeed very difficult to exploit the information to gain an extraordinary return.

Basic machine learning cannot create a real predictability where competitive markets have already accounted for most of the information; at best, it can stabilize weak signals and organize them more systematically. We may conclude noting that with simple public indicators at monthly frequency, the S&P 500 behaves in a manner that is largely consistent with an efficient and highly competitive market.

6.2 Future Directions

Future work could explore richer and more complex sources of information, such as firm fundamentals, analyst text, or alternative data. The analysis can be further extended to different horizons and international equity markets, but also to multi-asset contexts or less liquid markets. Such works may further clarify where machine learning provides the greatest value.

References

Fama, Eugene F. (1970). "Efficient Capital Markets: A Review of Theory and Empirical Work". In: *Journal of Finance* 25.2, pp. 383–417.

Fama, Eugene F. and Kenneth R. French (1992). "The Cross-Section of Expected Stock Returns". In: *Journal of Finance* 47.2, pp. 427–465.

Goyal, Amit and Ivo Welch (2008). "A Comprehensive Look at The Empirical Performance of Equity Premium Prediction". In: *Review of Financial Studies* 21.4, pp. 1455–1508.

Green, Jeremiah, John R. M. Hand, and Frank Zhang (2017). "The Characteristics that Provide Independent Information about Average U.S. Monthly Stock Returns". In: *Review of Financial Studies* 30.12, pp. 4389–4436.

Gu, Shihao, Bryan T. Kelly, and Dacheng Xiu (2020). "Empirical Asset Pricing via Machine Learning". In: *Review of Financial Studies* 33.5, pp. 2223–2273.

Harvey, Campbell R., Yan Liu, and Heqing Zhu (2016). "... and the Cross-Section of Expected Returns". In: *Review of Financial Studies* 29.1, pp. 5–68.

Jegadeesh, Narasimhan and Sheridan Titman (1993). "Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency". In: *Journal of Finance* 48.1, pp. 65–91.

A Additional Figures

A.1 Long–Short Strategy

Table 3: Long–Short ML Strategy (Top 10% minus Bottom 10%)

Strategy	CAGR	Volatility	Sharpe	MaxDD
ML (Long–Short)	-0.1371	0.1330	-1.0309	-0.4207

A.2 Subperiod Analysis (2022–2024)

Table 4: Subperiod Performance (2022–2024)

Strategy	CAGR	Volatility	Sharpe	MaxDD
ML Top 10%	0.1493	0.1945	0.7676	-0.1274
Equal-Weight	0.1510	0.1857	0.8130	-0.1133
Momentum 6M Top 10%	0.3031	0.1914	1.5835	-0.1216

A.3 Sensitivity Analysis

Table 5: Sensitivity Grid: Selection Fraction, Costs, and Period

Period	Top Fraction	Cost (bps)	CAGR (ML)	Sharpe (ML)
022–2024	0.10	0	0.1562	0.8028
2022–2024	0.10	0.0005	0.1493	0.7676
2022–2024	0.10	0.001	0.1425	0.7326
2022–2024	0.20	0	0.1556	0.8592
2022–2024	0.20	0.0005	0.1487	0.8214
2022–2024	0.20	0.001	0.1419	0.7838
Full	0.10	0	0.1231	0.6646
Full	0.10	0.0005	0.1164	0.6286
Full	0.10	0.001	0.1098	0.5928
Full	0.20	0	0.1201	0.6996
Full	0.20	0.0005	0.1135	0.6609
Full	0.20	0.001	0.1069	0.6224

A.4 Long–Short Cumulative Return

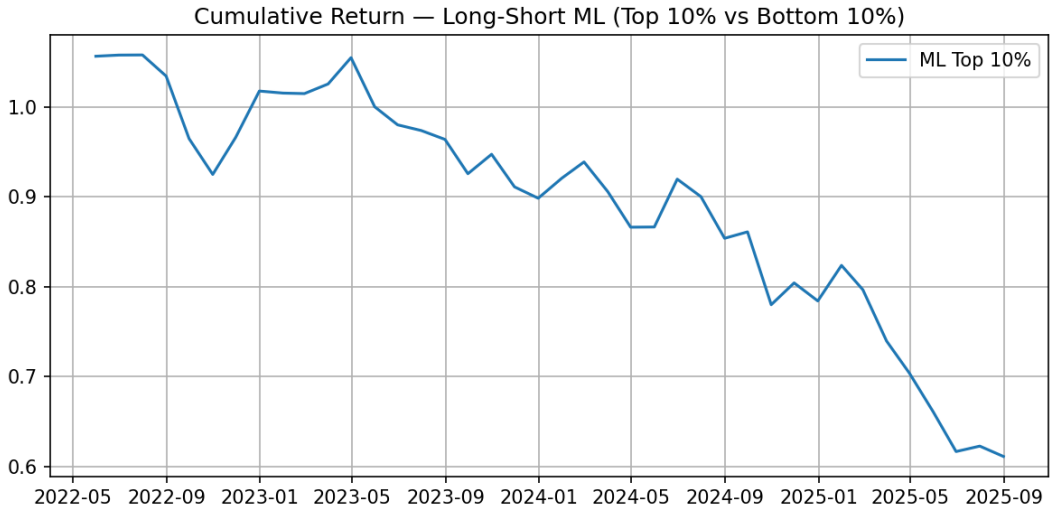

Figure 3: Cumulative returns of the long–short ML strategy. The performance is consistently negative, confirming the absence of a stable return spread.

A.5 Underwater Plot

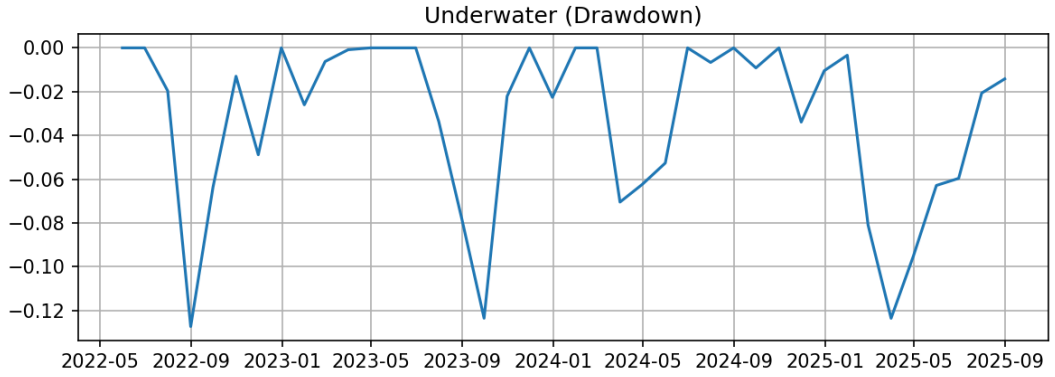

Figure 4: Underwater plot (drawdown) for the ML Top 10% strategy. Drawdowns reach approximately 14%, similar to the equal-weight benchmark.

A.6 Feature Importance of the Random Forest

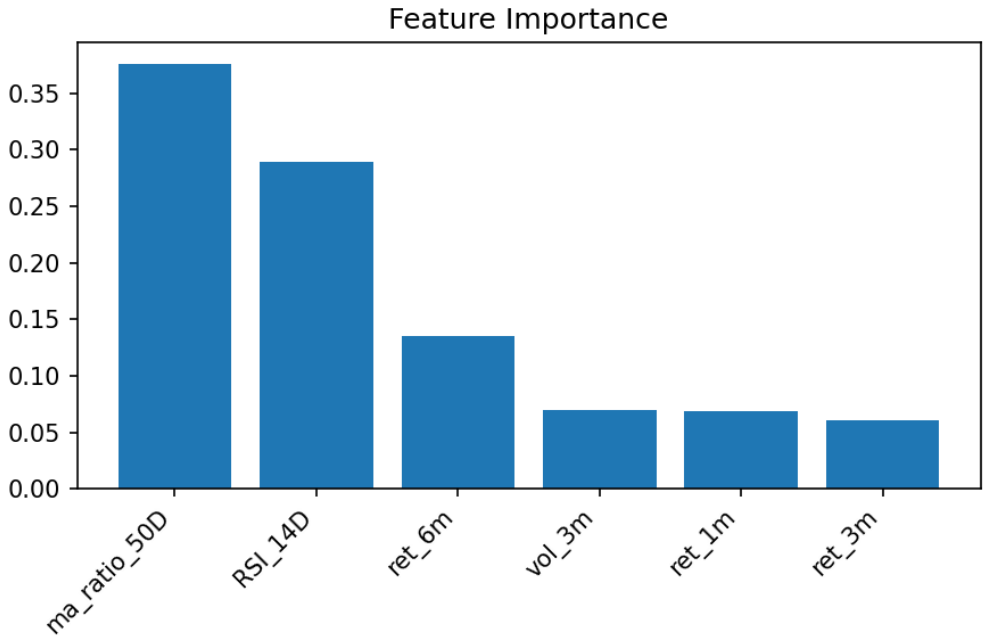

Figure 5: Feature importance obtained from the Random Forest model. The model assigns the highest relevance to medium-term technical indicators, in particular the 50-day moving-average ratio and the 14-day RSI, which together dominate the explanatory power.

B Code Repository

GitHub Repository: `https://github.com/marcodamico03/Marco_D-Amico_Project`

B.1 Repository structure

Marco_D-Amico_Project/

```
|-- data/              # Raw and cached S&P 500 datasets, ticker list
|-- features/          # Feature construction: momentum, volatility, MA ratios, RSI
|-- metrics/           # Evaluation metrics: AUC, Brier, calibration, portfolio stats
|-- models/            # ML models and walk-forward expanding-window logic
|-- notebooks/         # Exploratory analysis, debugging and ad-hoc testing
|-- portfolio/         # ML Top-decile, long-short, equal-weight, momentum code
|-- report/            # Plotting utilities and figure generation for the thesis
|
|-- main.py            # Full end-to-end pipeline (data → ML → portfolio → results)
|-- settings.py        # Global configuration variables and project-wide settings
|-- requirements.txt   # Python dependencies (pip)
|-- environment.yml    # Conda environment (fully reproducible setup)
|-- README.md          # Project documentation and usage instructions
|-- .gitignore         # Git ignore file
```

B.2 Installation and usage

On your terminal, clone the repository:

```
git clone https://github.com/marcodamico03/Marco_D-Amico_Project
cd Marco_D-Amico_Project
```

Create and activate the Conda environment:

```
conda env create -f environment.yml
conda activate ml-finance
```

Run the full pipeline:

```
python main.py
```

This command will:

- Load or download S&P 500 price data
- Construct monthly technical features
- Train ML models using a walk-forward expanding window
- Generate out-of-sample probability forecasts
- Build ML and benchmark portfolios
- Compute all performance statistics (CAGR, volatility, Sharpe ratio, drawdowns)
- Save all tables and figures in the `report/` directory