# Dynamic Portfolio Allocation Across Quantitative Strategies

## A Strategy-Level XGBoost Ensemble Approach

by

Jiawen Wei

An honors thesis submitted in partial fulfillment

of the requirements for the degree of

Bachelor of Science

Business and Economics Honors Program

NYU Shanghai

May 2026

| | |
|---|---|
| Professor Marti G. Subrahmanyam<br>Professor Christina Wang<br>Professor Wendy Jin | Professor Jingyuan Mo |
| Faculty Advisers | Thesis Adviser |

## Preface

This thesis examines dynamic portfolio allocation across quantitative strategies through a strategy-level XGBoost ensemble framework. The study is motivated by the observation that individual quantitative strategies often perform differently across market regimes, making static allocation difficult to sustain over time. Instead of predicting individual stock returns directly from raw market or accounting variables, this thesis focuses on the outputs of multiple strategy models and investigates whether machine learning can improve the allocation across them.

The research combines factor-based portfolio construction, backtesting, and supervised machine learning to evaluate the effectiveness of dynamic strategy allocation. It considers a set of heterogeneous quantitative strategies, including value, momentum, mean reversion, profitability, risk parity, and a three-down-low breakout strategy. These strategies are used as inputs to an XGBoost-based ensemble model, which generates portfolio weights and is evaluated against single-strategy results, an equal-weight strategy baseline, and the S&P 500 benchmark.

In addition to the main empirical framework, this thesis also examines several design choices that affect robustness and interpretability, including randomized year permutation, a simplified two-strategy allocation setting, and retraining versus no-retraining comparisons. Through these analyses, the thesis aims to assess whether a strategy-level ensemble approach can provide more stable and adaptive portfolio performance across different market conditions.

I hope this thesis contributes to the discussion of how machine learning can be applied not only to stock selection, but also to higher-level portfolio construction and strategy allocation.

**Acknowledgements**

As my college years come to an end, I would like to express my heartfelt gratitude to those who have accompanied me during this journey.

First, I would like to sincerely thank my advisor, Professor Jingyuan Mo, for his guidance and support over the past year and a half. His wisdom and humor not only helped me through the research process, but also sparked my interest in this field and encouraged me to pursue it with greater curiosity and commitment.

I am also deeply grateful to the Business and Economics Honors Program for giving me this valuable opportunity and for its continued support throughout the process. This experience has been meaningful to me both academically and personally, as it has shaped the way I envision my future.

Special thanks go to Jake Lin, my closest companion, for his patience, understanding, and steadfast presence in every aspect of life. I hope we will move forward together in the years to come, supporting and witnessing each other's growth.

I would also like to thank my best friends, Xiao Cheng and Amanda Ge, for their companionship over the past four years. Their friendship made difficult times easier to bear and joyful moments even more precious.

Finally, my deepest gratitude goes to my parents for providing me with a supportive educational environment and, more importantly, the freedom to make my own choices. Their trust and care have given me the courage to grow at my own pace.

To all of them, thank you for seeing me as I am, with all my imperfections, and for loving and believing in me nonetheless.

## Abstract

*This paper proposes a machine-learning-based framework for dynamically combining multiple classic quantitative trading strategies using XGBoost. Rather than applying ML at the individual stock level, we introduce a strategy-level ensemble approach: XGBoost learns time-varying weights across six heterogeneous strategies—value, momentum, mean reversion, profitability, risk parity, and a technical pattern (three-down-low breakout)—to construct a composite portfolio. Using daily data on the top 100 U.S. stocks by market capitalization (2013–2025), we show that (i) the XGBoost dynamic ensemble achieves more consistent risk-adjusted performance across market regimes than any single strategy; (ii) strategy-level combination substantially outperforms stock-level ML on the test set; (iii) randomized year-based train/validation splits, which feed a deduplicated pool of robust single-strategy variants into XGBoost, deliver the most consistent cross-period performance; and (iv) a deliberately simplified two-strategy version (Choose-2) reduces volatility and drawdown but does not match the full six-strategy ensemble. Overall, the results suggest that machine learning is more effective in this setting when used to allocate across structured strategy signals rather than to directly rank individual stocks. The study highlights the importance of strategy-level learning, robust validation design, and diversification across heterogeneous strategies in building adaptive quantitative portfolios.*



## Keywords

XGBoost; portfolio combination; strategy-level ensemble; quantitative trading; randomized year permutation; risk-adjusted performance

# Contents

# 1 Introduction

## 1.1 Background

Multi-factor and strategy-based investing are widely used in quantitative portfolio management because they provide systematic ways to capture return patterns associated with value, momentum, profitability, risk, and other market characteristics [1, 2, 3, 4]. However, individual quantitative strategies often perform differently across market environments. A strategy that works well during one period may lose effectiveness when market conditions change, investor behavior shifts, or macroeconomic uncertainty increases. As a result, no single strategy can be expected to consistently outperform across all regimes [5].

This regime dependence makes portfolio construction challenging. Traditional strategy combination methods, such as equal weighting or mean-variance optimization [6, 7], often rely on static or slowly changing assumptions about expected returns, risks, and correlations. These methods may provide a reasonable baseline, but they are less effective when the relationships among strategies change over time. In particular, they may fail to adjust quickly to structural market shifts or to capture nonlinear interactions among different strategy signals [8].

Combining multiple strategies can improve robustness, especially when the strategies have relatively low correlations with one another. Low-correlation strategies can provide diversification benefits because their returns do not move in exactly the same way across market conditions. This can help reduce portfolio volatility, smooth drawdowns, and improve stability across different regimes. Therefore, the key question is not only which individual strategies perform well, but also how to allocate dynamically across strategies as market conditions evolve.

Machine learning provides a potential solution to this problem [9]. XGBoost, in particular, offers a flexible ensemble learning framework that can capture nonlinear relationships and complex interactions among input features [10]. Instead of imposing fixed allocation rules, an XGBoost-based model can learn from historical strategy performance and adjust portfolio weights dynamically. This makes it suitable for examining whether strategy-level signals can

be combined more effectively than through traditional static allocation methods. In this thesis, XGBoost is applied at the strategy level to study whether dynamic allocation across multiple quantitative strategies can improve portfolio performance, robustness, and adaptability.

## 1.2 Research Questions

1. Can a machine learning model (XGBoost) learn time-varying optimal weights across heterogeneous strategies to improve risk-adjusted returns?
2. Is it more effective to apply ML at the strategy-weight level (combining strategy signals) or at the stock level (directly ranking individual stocks)?
3. How much do design choices—randomized year splits, a simplified low-correlation subset of strategies, extended binary switches (continuous vs. discrete labels, positive-only mapping, EMA smoothing, variant retention), and retrain vs. no-retrain inference—affect cross-period consistency?

## 1.3 Contributions

This thesis contributes to the literature on machine-learning-based portfolio construction by shifting the focus from stock-level prediction to strategy-level allocation. While many machine learning applications in quantitative finance attempt to predict individual stock returns directly, this thesis shows that the outputs of established quantitative strategies can themselves serve as meaningful inputs for an ensemble allocation model. By using the per-stock weights generated by six heterogeneous strategies as XGBoost features, the study provides a framework for learning across strategies rather than across individual securities.

The thesis also provides empirical evidence that the level at which machine learning is applied matters for portfolio performance. The comparison between strategy-level and stock-level machine learning shows that the strategy-level approach achieves stronger out-of-sample performance, particularly in risk-adjusted returns and drawdown control. This finding suggests that strategy-level signals may preserve more useful allocation information than sparse stock-level inputs, and that machine learning can be more effective when used to combine existing investment

logic rather than replace it entirely.

In addition, this thesis introduces a robustness-oriented model selection design through randomized year permutation. By varying the train-validation year split and constructing a deduplicated pool of robust strategy variants, the study reduces dependence on a single validation period and provides a more stable basis for ensemble construction. The analysis of the Choose-2 simplification and retrain versus no-retrain settings further shows that greater model complexity does not necessarily lead to better out-of-sample performance. These results clarify the trade-offs among strategy diversity, simplicity, robustness, and adaptability, while highlighting the practical value of dynamic strategy combination in portfolio allocation.

# 2 Literature Review

## 2.1 Classic Quantitative Strategies

The foundation of quantitative investment strategies is built upon decades of empirical research identifying systematic sources of return, commonly referred to as factors or anomalies. The value premium was formalized in the three-factor model, which shows that stocks with high book-to-market ratios tend to earn higher average returns than stocks with low book-to-market ratios [1]. This premium is often interpreted as compensation for risk or as evidence of market mispricing. This framework was later extended by research on profitability, which shows that gross profitability has significant predictive power for the cross-section of average returns and can be viewed as the "other side of value" [3]. Profitable firms generate significantly higher returns despite having higher valuation ratios. The five-factor asset pricing model further incorporates profitability and investment factors, improving the explanation of average stock returns and weakening the role of the original value factor in some empirical settings [4].

Parallel to value and fundamental metrics, the momentum anomaly has been robustly documented in the literature. Strategies that buy past winners and sell past losers have been shown to generate significant positive returns over medium-term horizons, typically from three to twelve months [2]. This phenomenon is often attributed to delayed price reactions and investor behav-

ioral biases. Conversely, over longer horizons, evidence of mean reversion suggests that stock prices may overreact to unexpected news, causing past losers to outperform past winners in subsequent periods [11].

Beyond return-oriented factors, portfolio construction and risk management have evolved significantly with the advent of risk parity strategies. Risk parity focuses on allocating portfolio risk rather than capital, so that each asset or strategy contributes more evenly to total portfolio risk [12]. This concept was rigorously formalized by [13] through the mathematical properties of equally weighted risk contribution portfolios, which position themselves between minimum variance and equally weighted portfolios.

While these classic strategies—value, momentum, mean reversion, profitability, and risk parity—have individually demonstrated robust empirical efficacy across various market cycles, their dynamic combination remains a complex challenge. Traditional static allocation often fails to adapt to regime shifts. Recent advancements in machine learning, particularly ensemble methods such as XGBoost, provide flexible tools for capturing nonlinear relationships and feature interactions [10]. This flexibility motivates their use in dynamic strategy allocation, where portfolio weights may need to adjust as market conditions evolve. This study bridges the gap between traditional factor investing and modern machine learning by proposing a strategy-level ensemble approach using XGBoost to dynamically combine these well-established quantitative strategies, aiming to enhance risk-adjusted returns and portfolio robustness.

## 2.2 Portfolio Optimization and Combination Methods

The evolution of portfolio optimization and combination methods has been a central theme in quantitative finance, transitioning from classical mathematical frameworks to advanced machine learning ensembles. The mean-variance optimization framework formalized the trade-off between expected return and risk and became the foundation of modern portfolio theory [6]. However, MVO is notoriously sensitive to estimation errors in expected returns and covariance matrices, often leading to highly concentrated, unstable portfolios that perform poorly in practice. The Black-Litterman model addresses expected-return estimation by combining market-implied

equilibrium returns with subjective investor views in a Bayesian framework, thereby generating more stable and intuitive portfolio weights [14]. Covariance shrinkage methods reduce instability by blending the sample covariance matrix with a structured target estimator, thereby improving the reliability of mean-variance inputs [15]. This shrinkage technique significantly reduces estimation error and improves the out-of-sample performance of mean-variance portfolios.

Despite these theoretical advancements, the empirical efficacy of sophisticated optimization techniques has been heavily scrutinized. In a seminal and provocative study, [7] demonstrated that the naive equal-weighting (1/N) strategy frequently outperforms various optimized portfolios out-of-sample. They concluded that the theoretical gains from optimal diversification are often completely overshadowed by the estimation errors inherent in the parameters. This finding spurred the development of robust allocation techniques that do not rely on notoriously difficult expected return estimates, such as risk budgeting and risk parity. [16] formalized the risk budgeting approach, which allocates risk rather than capital across assets, ensuring that each component contributes a predefined amount to the total portfolio volatility, thus enhancing diversification and downside protection.

Recently, the integration of machine learning into asset allocation has expanded the set of tools available for modeling complex, nonlinear relationships and interactions in high-dimensional financial data. Machine learning models, particularly tree-based methods and neural networks, have shown strong predictive performance in empirical asset pricing and risk premium measurement [8]. These findings motivate the question of whether machine learning can be applied not only to individual securities, but also to the combination of pre-constructed strategy signals. In this context, dynamic multi-strategy combination uses algorithms such as XGBoost to adjust the relative weights of value, momentum, mean reversion, profitability, and risk-based strategies according to the information contained in their historical strategy signals. This ensemble methodology aims to combine the complementary information contained in different strategies while reducing reliance on any single strategy. It therefore provides a natural extension of traditional asset-level optimization to a higher-level strategy allocation problem.

## 2.3 Machine Learning in Quantitative Finance

The application of machine learning (ML) in quantitative finance has attracted growing interest over the past decade, with expanding applications in return prediction, portfolio construction, and risk modeling [9]. Traditional linear factor models, while foundational to modern finance, often fail to capture the complex, nonlinear dynamics and intricate interactions inherent in financial markets. Addressing this critical limitation, empirical work analyzes various ML methods in asset pricing and shows that nonlinear models, particularly tree-based algorithms and deep neural networks, significantly outperform traditional regression-based approaches in measuring asset risk premiums and forecasting cross-sectional returns. These findings underscore the critical role of predictor interactions and nonlinearities that advanced ML techniques can capture, thereby substantially improving predictive performance relative to leading regression-based approaches in some empirical settings [8].

Among ML methodologies, tree-based ensemble models, especially gradient boosting frameworks, are useful for financial modeling because they can capture nonlinear relationships, handle sparse inputs, and incorporate regularization. XGBoost is a highly scalable end-to-end tree boosting system that has become widely used in predictive modeling tasks, including financial applications [10]. Its usefulness is partly related to its efficiency in handling sparse data, its sparsity-aware algorithm, and its regularization techniques [10]. Building upon the gradient boosting paradigm, LightGBM further enhances computational efficiency and scalability for high-dimensional and large-scale datasets through Gradient-based One-Side Sampling and Exclusive Feature Bundling [17]. These tree-based models are suitable for financial prediction tasks because they can model nonlinear relationships, handle sparse inputs, and reduce overfitting through regularization. In empirical asset pricing, tree-based methods have also been shown to perform well relative to traditional linear models [8].

Beyond individual asset prediction, ML techniques are increasingly being utilized in the broader context of portfolio construction and dynamic asset allocation. Recent studies, such as [5], explore the integration of deep learning and reinforcement learning to develop dynamic trading strategies that adapt to shifting market regimes and macroeconomic conditions. However,

while much of the existing literature heavily focuses on applying ML at the individual stock or asset level to predict cross-sectional returns, there remains a notable gap in leveraging these advanced algorithms for strategy-level ensemble approaches. Specifically, dynamically combining multiple established quantitative trading strategies—such as value, momentum, mean reversion, profitability, risk parity, and three-down-low breakout—using robust ML models like XGBoost presents a highly promising yet underexplored avenue. This study aims to bridge this literature gap by proposing a dynamic multi-strategy portfolio combination framework using XGBoost, thereby extending the application of ML from traditional asset-level prediction to sophisticated strategy-level optimization and ensemble construction.

## 2.4 Ensemble Learning and Strategy Combination

Ensemble learning is an important machine learning framework that improves predictive performance by combining multiple base learners. The theoretical foundations of ensemble methods were largely established by seminal works such as Breiman's introduction of Bagging [18], which reduces variance through bootstrap aggregating, and Friedman's development of Gradient Boosting Machines [19], which sequentially minimizes loss functions to reduce bias. In recent years, these foundational techniques have been extensively adapted for quantitative finance, where market noise and non-stationarity pose significant challenges. While many applications of ensemble methods in finance focus on the asset level, such as predicting individual stock returns or selecting stocks, these methods can also be extended to strategy-level combination. Strategy-level ensembles aggregate diverse trading strategies rather than individual securities, with the goal of improving diversification and stabilizing portfolio performance. A recent study proposes dynamic weighting methods that allocate capital across multiple stock-selection models using performance metrics such as Information Coefficients [20]. Dynamic model selection and adaptive ensemble methods have been studied for time-series forecasting and financial applications, providing methodological motivation for selecting models under changing data conditions [21, 22]. Motivated by the need to reduce overfitting to a single validation window, this thesis adopts a randomized year permutation design. The approach re-runs strategy selection across multiple train-validation year splits, constructs a deduplicated pool of robust strategy variants,

and feeds this pool into the downstream XGBoost ensemble. Building on this motivation, this thesis proposes an XGBoost-based multi-strategy combination framework that operates at the strategy level and uses randomized year permutation to reduce sensitivity to a single validation regime.

## 2.5 Technical Analysis and Pattern Recognition

The efficacy of technical analysis and pattern recognition in financial markets has long been debated, with empirical studies providing mixed but informative evidence. Simple technical trading rules, such as moving averages and trading-range breaks, have been shown to contain predictive information in historical stock return data, challenging a strict random walk interpretation [23]. Building upon this, [24] introduced a systematic, nonparametric kernel regression approach to automate the recognition of visual chart patterns, confirming that certain technical indicators provide incremental information and practical value. Beyond equities, the widespread adoption of technical analysis is evident in other markets; for instance, [25] highlighted the complementarity of technical and fundamental analysis among foreign exchange dealers, suggesting that practitioners often integrate both paradigms to optimize trading decisions. In recent years, the intersection of technical analysis and advanced computational methods has catalyzed the development of sophisticated quantitative trading strategies. [26] illustrated the profitability of intraday trading rules based on optimized flag pattern recognition, emphasizing the necessity of algorithmic precision in modern high-frequency environments. More recently, machine learning methods have been used to make pattern recognition more systematic and data-driven. [27] proposed a novel candlestick pattern recognition model utilizing machine learning techniques, which effectively incorporates shape, location, and multiple technical indicators to improve trading decisions. In the context of ensemble learning, XGBoost has emerged as a powerful tool for processing complex technical features. A machine learning trading system using XGBoost with N-period Min-Max labeling has been shown to improve trading performance in its empirical setting [28]. Collectively, this body of literature underscores the transition of technical analysis from subjective charting to objective, algorithm-driven pattern recognition. This evolution supports the inclusion of technical pattern-based signals as one component in a broader quantitative

strategy ensemble.

## 2.6 Research Gap

Existing ML-in-finance literature often focuses on stock-level prediction, using firm characteristics or asset-level signals as model inputs [8]. In contrast, fewer studies examine whether machine learning can dynamically weight pre-constructed strategy signals. This operates at a higher abstraction level and may offer a better signal-to-noise ratio by using strategy outputs rather than raw stock-level variables. This thesis bridges the gap by proposing a dynamic multi-strategy portfolio combination approach using XGBoost, operating strictly at the strategy level, and further evaluates how randomized year permutation, a low-correlation subset simplification, and retrain versus no-retrain inference interact with that framework.

# 3 Data

## 3.1 Stock Universe

The stock universe consists of the top 100 U.S. stocks by market capitalization as of 2010, drawn from S&P 500 constituents. To ensure data consistency across strategies and avoid gaps in backtesting, the sample is restricted to stocks with complete trading records throughout the study period. Price data are obtained from Yahoo Finance through `yfinance`, covering January 2013 to December 2025.

## 3.2 Price Data

Daily OHLCV data, including Open, High, Low, Close, and Volume, are downloaded from Yahoo Finance. Both adjusted and unadjusted prices are collected, but adjusted prices are used in the subsequent strategy construction and backtesting process to account for stock splits, dividends, and other corporate actions. Unadjusted prices are retained as auxiliary raw price records.

## 3.3 Fundamental Data

The fundamental data are obtained from WRDS Compustat quarterly files. The main variables used in our model are Book Value Per Share (BVPS) and Return on Assets (ROA), which support the construction of the value and profitability strategies. Since these accounting variables are reported at a quarterly frequency, they are forward-filled to the daily trading calendar after being aligned by report date. This process allows the fundamental signals to be matched with daily stock price data while preserving the timing structure of the original accounting information.

## 3.4 Sample Split and Data Preprocessing

Table 1 summarizes the training, validation, and test periods used for our model.

Table 1: Train, validation, and test periods.

| Period | Date range | Role | Market characteristics |
|---|---|---|---|
| Train | 2014-01-01 – 2019-12-31 | Model training | Bull market, low volatility |
| Validation | 2020-01-01 – 2022-12-31 | Hyperparameter tuning | COVID crash, recovery, rate hikes |
| Test | 2023-01-01 – 2025-12-30 | Out-of-sample evaluation | AI rally, mixed conditions |

The training period covers 2014–2019 and is used for model estimation. The validation period covers 2020–2022 and is used for hyperparameter tuning. The test period covers 2023–2025 and serves as the out-of-sample evaluation window. Data from 2013 are used to generate initial rolling signals and lagged features at the beginning of the training sample. After the sample split is defined, all price and fundamental variables are aligned by date and ticker before strategy construction, model training, and backtesting.

# 4 Methodology

## 4.1 Individual Strategy Definitions

Throughout this subsection, all six standalone strategies use *daily rebalancing*: on every trading date in the common calendar, rules prescribe an updated portfolio target before trades are lined up at the next open according to the shared backtesting convention.

### 4.1.1 Value Strategy

For each date, we compute price-to-book as $P/B = \text{Close}/\text{BVPS}$. Among firms with non-negative ROA (ROA $\geq 0$), the strategy buys the $N$ stocks with the lowest $P/B$ ratios—that is, the cheapest names on a book-equity basis within the solvent sample—and allocates capital to them at equal weights.

### 4.1.2 Momentum Strategy

For each trading day, we compute simple momentum for every stock as cumulative return over fixed lookback horizons, $(\text{Close}_t/\text{Close}_{t-k}) - 1$. The implementation allows either a single horizon or two horizons; in the two-horizon case, the two return series are averaged into one score. On each rebalance date, we rank all stocks with valid scores and hold the top fraction $q$ of names by that score, with equal weights spread across the selected stocks. The lookback configuration and $q$ index a particular specification of the rule.

### 4.1.3 Mean Reversion Strategy

For each stock and date we form a $z$-score of the closing price relative to the mean and standard deviation of the prior trading window (length $L$ days, with mean and dispersion computed from past closes excluding the current observation, matching the feature construction in our pipeline). The strategy builds positions sequentially from day to day: existing holdings are closed when the $z$-score rises to at least a take-profit level $T$ or falls below a stop-loss threshold $S$; newly

eligible names are drawn from the oversold pool of stocks whose $z$-score is less than or equal to the (negative) entry threshold $B$. After applying these rules, every stock still in the book receives the same weight, $\min\{1/n,\, W\}$, where $n$ is the number of simultaneous positions and $W$ is an upper bound on per-name weight; all qualifying oversold names are held jointly rather than limiting the book to a fixed small number of names. The window length $L$ and the thresholds $B$, $S$, $T$, and $W$ are part of the strategy specification, constrained by $S \leq B < 0$.

#### 4.1.4 Profitability Strategy

On each date we take return on assets (ROA) from the forward-filled quarterly fundamentals, intersected with stocks that trade that day at a strictly positive price. Among valid observations we rank by ROA and hold the top $N$ names with the highest profitability, equal-weighted.

#### 4.1.5 Risk Parity Strategy

Using daily returns, we estimate each stock's risk over a rolling window of length $H$ trading days under one of two definitions: conventional volatility or downside-oriented volatility. We then rank stocks by this risk measure, keep the $M$ names with the lowest estimated risk, and assign portfolio weights in proportion to the inverse of each selected stock's risk, renormalized to sum to one. The choice of risk definition, $H$, and $M$ parametrizes this construction.

#### 4.1.6 Three-Down-Low Breakout Strategy

This pattern-driven long strategy proceeds in five-day bundles. Days 1–3 must each close below the open, with closes strictly declining and lows strictly falling from day to day. On day 4 we require a bullish rejection of the prevailing low: the open gaps below day 3's low while the close recovers above that low and finishes above day 3's close—a gap-down reversal. The setup is flagged after the day 4 close; execution is at the day 5 open using effective OHLC prices (with missing extremes filled from closes to stay consistent with the backtest harness). Risk is managed with a hard initial stop at a fraction $s$ of the entry price, a multiplicative trailing-stop rule keyed to new highs after entry (step multiplier $r$, offset $o$), and an exit when a symmetric "mirror"

reversal pattern fires for existing longs; any position is also limited to a maximum weight $W$ per name before rescaling. The tuple $(s, r, o, W)$ parametrizes risk controls around the pattern.

## 4.2 Individual Strategy Parameter Optimization

**Method**: Grid search over strategy-specific hyperparameters (e.g., top $N$, lookback window). For each candidate parameter vector, we backtest on the training window, validation window, a single contiguous training+validation window, and the test window. Sharpe ratio and Calmar ratio are computed as in Table 6. We aggregate them into a scalar objective on each window $w$:

$$\mathrm{Score}_w = \mathrm{Sharpe}_w + \mathrm{Calmar}_w \,. \tag{1}$$

**Selection rules** (exported tags match the codebase):

- `best_by_train`: maximize $\mathrm{Score}$ on the training period.
- `best_by_validation`: maximize $\mathrm{Score}$ on the validation period.
- `best_by_trainval`: maximize $\mathrm{Score}$ on the contiguous sample from the training start through the validation end.
- `best_by_robust`: maximize $\mathrm{robust\_score}$ defined below.

$$\mathrm{robust\_score} = \mathrm{Score}_{\mathrm{val}} - \lambda \left| \mathrm{Score}_{\mathrm{val}} - \mathrm{Score}_{\mathrm{train}} \right| , \qquad \lambda = 0.5 \,, \tag{2}$$

which rewards validation $\mathrm{Score}$ while penalizing large train–validation gaps (overfitting to a single regime). The main pipeline adopts `best_by_trainval` as the default single-strategy setting; `best_by_robust` is additionally used by the randomized year permutation pipeline (§4.6).

Table 2 reports, for each of the six rules, the discrete grids traversed during search alongside the optimum kept under `best_by_trainval`; symbols in the table match those introduced in §4.1 above.

Table 2: Individual strategy grids and `best_by_trainval` optima (fixed calendar split; symbols as in §4.1).

| Strategy | Symbol | Grid values | Best |
|---|---|---|---|
| Value | $N$ | {4, 6, 8, 10, 12, 15, 20, 25} | 10 |
| Mean reversion | $L$ | {126, 252} | 126 |
| Mean reversion | $B$ | $\{-2.5, -2, -3\}$ | $-2$ |
| Mean reversion | $S$ | $\{-3.5, -3, -2.5\}$ with $S \leq B$ | $-3.5$ |
| Mean reversion | $T$ | {0.0, 0.5, 1} | 0.0 |
| Mean reversion | $W$ | {0.20, 0.25} | 0.25 |
| Momentum | $k$ | nine horizons $\{5, \ldots 252\}$; single horizon or unordered pair averaged | `[189]` |
| Momentum | $q$ | {0.10, 0.15, 0.20, 0.25, 0.30} | 0.15 |
| Profitability | $N$ | {4, 6, 8, 10, 12, 15, 20, 25} | 8 |
| Risk parity | *type* | {`vol`, `downside_vol`} | `downside_vol` |
| Risk parity | $H$ | {21, 42, 60, 84, 126} | 126 |
| Risk parity | $M$ | {15, 20, 25, 30, 40} | 15 |
| Three-down low | $s$ | {0.95, 0.97, 0.98} | 0.95 |
| Three-down low | $r$ | {1.03, 1.05, 1.07} | 1.05 |
| Three-down low | $o$ | {0.01, 0.02, 0.03} | 0.03 |
| Three-down low | $W$ | {0.15, 0.20, 0.25} | 0.25 |

## 4.3 Static Equal-Weight Baseline

Before introducing any learned combination, we construct a static equal-weight baseline over the six optimized single strategies. At the start of each evaluation period, including the training, validation, and test periods, each strategy is assigned an equal allocation of $1/6$. The composite portfolio is then held without rebalancing within that period, so the strategy weights drift according to realized returns until the next period boundary. This baseline serves as a simple non-learning benchmark for evaluating whether the XGBoost ensemble adds value beyond a static combination of strategies.

## 4.4 XGBoost Dynamic Ensemble Model

### 4.4.1 Feature Construction

For each (date, stock) pair, the feature vector is six-dimensional: the target portfolio weight assigned by each of the six strategies (value, mean_reversion, momentum, profitability, risk_parity, three_down_low_breakout). A strategy that does not signal a given stock on a given date con-

tributes a zero weight. No other inputs (price, volume, raw fundamentals, macro data) are used.

#### 4.4.2 Label Definition

**Open-to-close return**: $y[t] = \text{Close}[t+1]/\text{Open}[t+1] - 1$. Signal generated at $T$ close $\rightarrow$ executed at $T{+}1$ open $\rightarrow$ evaluated at $T{+}1$ close. An optional $k$-day compound variant (`future_horizon_days`) is used in the extended grid (§4.8).

#### 4.4.3 Model Variants

**XGBRegressor**: objective `reg:squarederror`; directly predicts the next-day (or $k$-day) return.
**XGBRanker**: objective `rank:pairwise`; `eval_metric` = `ndcg`; cross-sectional ranking. The ensemble uses the ranker variant with `rank_decay` weighting.

#### 4.4.4 Training Protocol

**Single-pass grid search on validation**: For each hyperparameter combination (`n_estimators`, `max_depth`, `learning_rate`, `ema_decay`, `max_single_weight`, `future_horizon_days`, `upside_threshold`, `p_score_threshold`), the model is trained on the training window (2014–2019), scored by the validation-period backtest Sharpe + Calmar, and the argmax configuration is retained. We report results under two downstream inference protocols:

- **No-retrain**: the argmax model (trained only on train) is directly used to produce signals for validation and test.
- **Retrain**: the argmax hyperparameters are refit on train+val and then used to produce test signals only.

The retrain variant serves as our main out-of-sample reference on test; the no-retrain variant is retained to isolate how much the validation sample itself contributes when folded into training (see §5.6).

## 4.5 Strategy-Level vs. Stock-Level Combination

To isolate the benefit of operating at the strategy-weight abstraction, we build two parallel pipelines that consume the *same* six strategy signals:

- **Strategy-level (primary)**: features are the six strategy weights; the model learns the relative strength of the strategies over time and emits daily strategy weights, which are then combined into stock weights through the underlying per-stock strategy weights.
- **Stock-level (baseline for comparison)**: features are the same six strategy weights per (date, stock), but the model is trained as a cross-sectional ranker. Each day its output is a normalized vector of stock weights on the Top-$N$ names by predicted rank.

Table 3 contrasts the two approaches.

Table 3: Strategy-level vs. stock-level combination.

| Approach | Features | Prediction target | Portfolio construction |
|---|---|---|---|
| Strategy-level | Six strategy weights per stock | Next-day return (or rank) | Weighted combination of strategy signals |
| Stock-level | Six strategy weights per stock | Next-day rank | Daily normalized stock weights on Top-$N$ by score (zeros elsewhere) |

## 4.6 Randomized Year Permutation Pipeline

Because the fixed calendar split in Section 3 ties hyperparameter selection to one specific validation regime, namely the 2020–2022 COVID window, we also evaluate a randomized year permutation pipeline. Instead of preserving the chronological order within the 2014–2022 model-selection window, this pipeline randomly assigns years to the training and validation sets. The purpose is to reduce the dependence of strategy selection on a single validation period and to generate a more diverse set of robust strategy variants.

For each random split $s \in \{1, \ldots, S\}$, the procedure is as follows. First, the years from 2014 to 2022 are randomly partitioned into a training subset and a validation subset, while 2023–2025 is

always kept as the out-of-sample test period. Second, the full grid search is re-run for each of the six standalone strategies under this randomized train-validation split, and the `best_by_robust` variant is exported for each strategy. Third, the selected variants from all random splits are appended to a global pool. After all splits are completed, identical variants are deduplicated to form a pool of unique robust strategy variants.

The downstream XGBoost ensemble treats each unique strategy variant as a distinct feature column. As a result, a single strategy family can contribute multiple tuned versions to the model at the same time. The XGBoost training protocol, feature definition, and label definition are identical to those described in Section 4.4.

## 4.7 Simplified Two-Strategy Variant (Choose-2)

To test whether a low-correlation strategy subset can deliver better risk control and more stable performance, we build a simplified variant that uses only profitability and three-down-low breakout. Profitability is selected because it delivers the strongest standalone performance on the test set, while three-down-low breakout is included because it has relatively low correlations with the other strategies and therefore provides important diversification value. The remainder of the pipeline is identical to the randomized year permutation pipeline in Section 4.6, except that per-split grid search is performed only for these two strategies, and the XGBoost ensemble receives only the corresponding two-strategy pool of robust variants. This variant is designed to examine whether a compact strategy subset combining strong standalone performance with diversification value can reduce risk and drawdown while maintaining competitive portfolio performance.

## 4.8 Extended Switch Grid and `c1_p1_e1` Tuning

On top of the basic grid search, we introduce four orthogonal binary switches that affect how labels and weights are constructed:

- `c0/c1`: continuous label (raw forward return) vs. discrete label (binarized at `upside_threshold`).
- `p0/p1`: full sign-preserving mapping vs. positive-only mapping from model score to weight.

- `e0/e1`: no smoothing vs. exponential moving average smoothing (`ema_decay`) on the weight path.
- `r0/r1` (randomized pipeline only): full deduplicated variant pool vs. robust Top-2 sub-pool per strategy family.

Across the 16 combinations ("ab16"), the configuration `c1_p1_e1` (discrete labels, positive-only mapping, EMA smoothing) attains the highest validation score in both the fixed-split pipeline and the randomized pipeline; within the randomized pipeline, `r0` (full pool) slightly outperforms `r1` (robust Top-2 pool). For the `c1_p1_e1` configuration we then expand the parameter grid to enable finer tuning; the final best configurations are summarized in Tables 4 and 5.

Table 4: Extended XGBoost parameter grid and best configuration under the fixed split (`c1_p1_e1`, 14,580 candidates).

| Parameter | Grid values | Best |
|---|---|---|
| `n_estimators` | 100, 150, 200, 250 | 150 |
| `max_depth` | 2, 3, 4 | 4 |
| `learning_rate` | 0.03, 0.05, 0.08 | 0.03 |
| `ema_decay` | 0.7, 0.85, 0.95 | 0.7 |
| `max_single_weight` | 0.4, 0.5, 0.6 | 0.6 |
| `future_horizon_days` | 1, 3, 5, 10, 20 | 5 |
| `upside_threshold` | 0.0, 0.01, 0.02 | 0.01 |
| `p_score_threshold` | 0.0, 0.01, 0.02 | 0.02 |

Table 5: Extended XGBoost parameter grid and best configuration under the randomized year permutation pipeline (`c1_p1_e1+r0`, 29,160 candidates).

| Parameter | Grid values | Best |
|---|---|---|
| `n_estimators` | 100, 150, 200, 250 | 150 |
| `max_depth` | 2, 3, 4 | 4 |
| `learning_rate` | 0.03, 0.05, 0.08 | 0.03 |
| `ema_decay` | 0.7, 0.85, 0.95 | 0.7 |
| `max_single_weight` | 0.4, 0.5, 0.6 | 0.6 |
| `future_horizon_days` | 1, 3, 5, 10, 20 | 5 |
| `upside_threshold` | 0.0, 0.01, 0.02 | 0.01 |
| `p_score_threshold` | 0.0, 0.01, 0.02 | 0.02 |
| `objective_mode` | `pairwise`, `non_pairwise` | `pairwise` |

### 4.9 Backtesting Framework

The backtesting framework follows a next-day execution convention. Portfolio signals are generated at the close of day $T$, and the corresponding trades are executed at the open of day $T+1$. Transaction costs are incorporated as a $0.01\%$ commission plus $0.05\%$ slippage per unit of turnover. Portfolio performance is evaluated using the metrics summarized in Table 6.

Table 6: Performance and trading metrics.

| Metric | Definition |
|---|---|
| Total return | Cumulative portfolio return over the period |
| Annualized return | Geometric annualized return |
| Volatility | Annualized standard deviation of daily returns |
| Sharpe ratio | Annualized return / annualized volatility |
| Max drawdown | Largest peak-to-trough decline |
| Calmar ratio | Annualized return / \|max drawdown\| |
| Win rate | Fraction of positive-return trading days |
| Avg. turnover rate | Mean daily portfolio turnover |

## 5 Empirical Results

### 5.1 Individual Strategy Performance

Table 7 reports validation-set performance (2020–2022, includes COVID-19). Table 8 reports test-set performance (2023–2025). Table 9 summarizes S&P 500 buy-and-hold benchmarks.

Table 7: Individual strategy performance: validation set (2020–2022).

| Strategy | Ann. return | Sharpe | Calmar | Max DD |
|---|---|---|---|---|
| Three-Down-Low | 21.0% | 1.20 | 1.54 | $-13.6\%$ |
| Momentum | 18.8% | 0.75 | 0.66 | $-28.3\%$ |
| Risk Parity | 10.0% | 0.58 | 0.45 | $-22.3\%$ |
| Profitability | 12.7% | 0.57 | 0.50 | $-25.6\%$ |
| Mean Reversion | 9.8% | 0.45 | 0.21 | $-45.7\%$ |
| Value | 8.0% | 0.40 | 0.17 | $-46.6\%$ |

No single standalone strategy performs best across both periods. The Three-Down-Low strategy performs strongest during the validation period, which includes the COVID-19 shock and

Table 8: Individual strategy performance: test set (2023–2025).

| Strategy | Ann. return | Sharpe | Calmar | Max DD |
|---|---|---|---|---|
| Profitability | 24.8% | 1.53 | 1.54 | −16.0% |
| Value | 28.2% | 1.27 | 1.21 | −23.3% |
| Momentum | 16.3% | 0.98 | 1.14 | −14.3% |
| Three-Down-Low | 9.1% | 0.89 | 0.72 | −12.6% |
| Mean Reversion | 13.3% | 0.76 | 0.50 | −26.5% |
| Risk Parity | 6.6% | 0.63 | 0.58 | −11.5% |

Table 9: S&P 500 buy-and-hold benchmark.

| Period | Ann. return | Sharpe | Max DD |
|---|---|---|---|
| Train (2014–2019) | 9.9% | 0.79 | −19.8% |
| Validation (2020–2022) | 5.6% | 0.34 | −33.9% |
| Test (2023–2025) | 21.5% | 1.37 | −18.9% |

higher market volatility, while the Profitability strategy performs best during the calmer test period. This shift suggests that individual strategies are sensitive to market regimes, and it motivates the use of a dynamic multi-strategy combination rather than relying on one fixed strategy.

## 5.2 Strategy Correlation Analysis

Table 10 reports pairwise Pearson correlations of daily strategy returns. Fundamental strategies (Value, Profitability, Risk Parity) are highly correlated (0.73–0.94), limiting diversification within this group. Three-Down-Low Breakout has the lowest correlation with all other strategies (0.35–0.51), making it the most valuable diversifier. Profitability attains the highest test Sharpe among the six standalone rules (Table 8); together these motivate the profitability–three-down-low Choose-2 subset (§4.7).

## 5.3 Strategy-Level vs. Stock-Level Comparison

The results show that, during the test period, the strategy-level approach outperforms the stock-level approach across all major performance metrics, with the only exception being a higher turnover rate (Table 11). This suggests that although the strategy-level approach trades more frequently, it achieves stronger overall return performance and better risk-adjusted outcomes.

Table 10: Pairwise Pearson correlation of daily strategy returns (abbreviated labels).

| | EqW | MR | Mom | Prof | RP | Value | 3DLB |
|---|---|---|---|---|---|---|---|
| Equal Weight | 1.00 | 0.66 | 0.81 | 0.84 | 0.85 | 0.94 | 0.51 |
| Mean Reversion | | 1.00 | 0.53 | 0.58 | 0.58 | 0.63 | 0.35 |
| Momentum | | | 1.00 | 0.77 | 0.73 | 0.70 | 0.39 |
| Profitability | | | | 1.00 | 0.75 | 0.73 | 0.41 |
| Risk Parity | | | | | 1.00 | 0.74 | 0.42 |
| Value | | | | | | 1.00 | 0.50 |
| Three-Down-Low | | | | | | | 1.00 |

Therefore, the subsequent analysis focuses primarily on the strategy-level framework as the main modeling approach.

A possible explanation for this performance difference lies in feature density and information preservation. Although the stock-level approach also uses the weights generated by the six strategies as input features, these features are often highly sparse at the individual stock level. On many trading days, a large number of stocks receive zero weights from several strategies, so the model observes many zero-valued signals rather than informative variation. As a result, when strategy outputs are mapped into stock-level features, part of the original strategy information may be compressed or weakened. In contrast, the strategy-level approach preserves strategy signals at a more aggregated level, which may reduce information loss during feature construction. Compared with sparse stock-level inputs, these denser strategy-level signals may better reflect the changing usefulness of different strategies across market environments.

Table 11: Strategy-level vs. stock-level XGBoost on the test set (both rows use validation-tuned hyperparameters via grid search).

| Approach | Ann. return | Volatility | Sharpe | Calmar | Max DD | Turnover |
|---|---|---|---|---|---|---|
| Strategy-level (XGBoost_grid) | 17.8% | 13.2% | 1.35 | 1.26 | −14.1% | 22.6% |
| Stock-level (XGBoost_stock_grid) | 14.8% | 17.0% | 0.87 | 0.93 | −15.9% | 19.1% |

## 5.4 Ablation Study: ab16 Configuration Experiment

The ab16 results show that positive-only mapping and EMA smoothing are the most effective switch choices. Although the final configuration is selected based on validation-period performance, the test-period results in Tables 12 and 13 show that the `p1_e1` setting remains the

strongest out of sample. In the fixed-split pipeline, `p1_e1` produces the highest Sharpe and Calmar ratios, while in the randomized pipeline it remains highly competitive. These findings support the use of `c1_p1_e1`, with `r0` retained in the randomized pipeline, as the main specification.

Table 12: ab16 configuration experiment on the fixed split (test set). Rows are ordered by test Sharpe ratio. Because the `r0` and `r1` switches produce identical metrics under the fixed split, they are merged into a single row.

| Config | Sharpe | Calmar | Ann. return | Turnover |
|---|---|---|---|---|
| `c0_p1_e1` / `c1_p1_e1` | 1.31 | 1.26 | 17.8% | 22.6% |
| `c0_p0_e1` / `c1_p0_e1` | 1.23 | 1.18 | 15.8% | 8.0% |
| `c0_p0_e0` / `c1_p0_e0` | 1.22 | 1.18 | 15.6% | 8.6% |
| `c0_p1_e0` / `c1_p1_e0` | 0.58 | 0.45 | 6.6% | 69.4% |

Table 13: ab16 configuration experiment on the randomized year permutation pipeline (test set).

| Config | Sharpe | Calmar | Ann. return | Turnover |
|---|---|---|---|---|
| `c0_p1_e1_r0` | 1.29 | 1.37 | 19.4% | 15.8% |
| `c0_p1_e1_r1` | 1.26 | 1.34 | 18.2% | 15.2% |
| `c1_p1_e1_r1` | 1.26 | 1.29 | 17.2% | 10.2% |
| `c1_p1_e1_r0` | 1.25 | 1.28 | 16.7% | 10.3% |
| `c0_p0_e1_r0` | 1.23 | 1.17 | 15.2% | 8.2% |
| `c1_p0_e1_r0` | 1.23 | 1.17 | 15.2% | 8.2% |
| `c0_p0_e0_r0` | 1.23 | 1.17 | 15.2% | 8.3% |
| `c1_p0_e0_r0` | 1.23 | 1.17 | 15.2% | 8.3% |
| `c0_p0_e1_r1` | 1.15 | 1.06 | 14.3% | 7.9% |
| `c1_p0_e1_r1` | 1.15 | 1.06 | 14.3% | 7.9% |
| `c0_p0_e0_r1` | 1.15 | 1.07 | 14.3% | 7.9% |
| `c1_p0_e0_r1` | 1.14 | 1.05 | 14.1% | 8.7% |
| `c0_p1_e0_r1` | 1.08 | 1.08 | 15.0% | 20.6% |
| `c1_p1_e0_r0` | 0.55 | 0.48 | 7.1% | 91.5% |
| `c0_p1_e0_r0` | 0.44 | 0.24 | 5.8% | 56.1% |
| `c1_p1_e0_r1` | 0.09 | 0.00 | −0.0% | 91.1% |

## 5.5 Choose-2 Variant Results

The Choose-2 variant (§4.7) achieves smaller volatility and shallower maximum drawdown than the full six-strategy randomized ensemble, consistent with the intuition that combining the highest test-period standalone Sharpe rule with a systematically low-correlation pattern strategy tightens the portfolio risk profile. However, its annualized return and Sharpe ratio fall short of the full variant on the test set, indicating that the four strategies dropped by Choose-2 still

carry complementary information that XGBoost exploits. We therefore retain the full six-strategy pool as the default and treat Choose-2 as a *risk-reduction* alternative rather than a performance-maximizing one.

## 5.6 Test Sample: No-Retrain versus Retrain

With hyperparameters frozen at the validation-argmax configurations from §4.4, we contrast two inference protocols on the test window (2023–2025): fit once on train only (*no-retrain*) versus refit on train+val before predicting test (*retrain*). Tables 14 and 15 report the resulting metrics.

Table 14: Test set (2023–2025), no-retrain (model fit on train only).

| Method | Ann. return | Sharpe | Calmar | Max DD | Turnover |
|---|---|---|---|---|---|
| S&P 500 | 21.5% | 1.37 | 1.14 | −18.9% | — |
| Baseline | 16.9% | 1.31 | 1.29 | −13.2% | — |
| XGBoost_Gridsearch | 16.5% | 1.19 | 1.21 | −13.7% | 23.2% |
| XGB_Grid_Extended | 9.1% | 0.82 | 0.67 | −13.6% | 15.3% |
| random_split_extended | 22.4% | 1.27 | 1.24 | −18.1% | 13.6% |

Table 15: Test set (2023–2025), retrain on train+val (hyperparameters frozen at the argmax of the validation grid search).

| Method | Ann. return | Sharpe | Calmar | Max DD | Turnover |
|---|---|---|---|---|---|
| S&P 500 | 21.5% | 1.37 | 1.14 | −18.9% | — |
| Baseline | 16.9% | 1.31 | 1.29 | −13.2% | — |
| XGBoost_Gridsearch | 17.4% | 1.29 | 1.24 | −14.1% | 22.7% |
| XGB_Grid_Extended | 8.1% | 0.78 | 0.64 | −12.7% | 15.1% |
| random_split_extended | 22.7% | 1.26 | 1.24 | −18.3% | 13.9% |

The direction of the retrain effect depends on the method. `XGBoost_Gridsearch` improves from Sharpe $1.19 \to 1.29$ and Calmar $1.21 \to 1.24$: the basic grid does not overfit the validation window, so folding validation into training adds effective sample size. `XGB_Grid_Extended` deteriorates from Sharpe $0.82 \to 0.78$ and Calmar $0.67 \to 0.64$: the extended grid's argmax already fits the 2020–2022 COVID regime tightly, and retraining on train+val reinforces that bias rather than aiding generalization. `random_split_extended` is nearly flat (Sharpe $1.27 \to 1.26$): hyperparameters chosen across many year-permutations weakly depend on any single validation slice, so incremental retraining changes little.

## 5.7 Validation versus Test under No-Retrain

Table 16 reports validation-period performance under the no-retrain setting, while Table 14 reports the corresponding test-period performance under the same inference protocol. This comparison examines whether methods that perform well during the validation period can maintain their performance in the out-of-sample test period. Validation-period retrain results are omitted because retrained models already include validation data during fitting.

Table 16: Validation set (2020–2022), no-retrain (model fit on train only).

| Method | Ann. return | Sharpe | Calmar | Max DD | Turnover |
|---|---|---|---|---|---|
| S&P 500 | 5.6% | 0.34 | 0.17 | −33.9% | — |
| Baseline | 13.6% | 0.69 | 0.45 | −29.9% | — |
| XGBoost_Gridsearch | 16.6% | 0.77 | 0.52 | −32.3% | 21.9% |
| XGB_Grid_Extended | 17.1% | 0.99 | 0.87 | −19.7% | 17.6% |
| random_split_extended | 21.8% | 0.85 | 0.60 | −36.2% | 13.2% |

In the validation period, all active strategy-combination methods outperform the S&P 500 in terms of annualized return and Sharpe ratio. Since the validation period is characterized by relatively high volatility and frequent shifts in strategy performance, these results suggest that the dynamic strategy-combination models are able to capture the validation-period environment during the model-selection stage. However, the rankings change substantially in the test period. The advantage of most active methods becomes less pronounced, while the S&P 500 itself performs strongly during 2023–2025. In particular, `XGB_Grid_Extended` shows a clear decline from validation to test, indicating that its strong validation performance does not fully carry over to the out-of-sample period and may depend more heavily on the specific validation regime. In contrast, `random_split_extended` remains highly competitive in the test period, suggesting better cross-period consistency.

Overall, the validation-to-test comparison shows that strong validation performance alone is not sufficient evidence of robustness. Configurations selected on a single validation window may not generalize equally well to later periods, while randomized year permutation helps reduce dependence on any single validation window and provides a more stable basis for selecting configurations.

# 6 Limitations

This study has several limitations. First, the stock universe is fixed based on the top 100 U.S. stocks by market capitalization as of 2010, which may introduce survivorship bias because delisted or bankrupt firms are excluded from the sample. Second, the analysis focuses on large-cap U.S. equities, so the results may not generalize to small-cap stocks, international markets, or markets with different trading rules and restrictions. Third, the study involves multiple rounds of hyperparameter search across both standalone strategies and ensemble configurations. Although validation and test splits are used to reduce overfitting, the broad search space still creates potential data-snooping risk, as some selected configurations may reflect sample-specific patterns rather than fully generalizable relationships. Fourth, transaction costs are modeled using a simplified fixed-rate structure, while real-world market impact, bid–ask spreads, liquidity constraints, and capacity limits are not fully captured. Finally, BVPS and ROA are aligned by public report date to reduce look-ahead bias. However, the backtest does not fully model real-time data availability frictions, such as database update delays or information-processing delays after financial statements are released.

# 7 Conclusion

This thesis examines whether machine learning can improve portfolio construction by learning how to allocate dynamically across quantitative strategies. Using six heterogeneous strategies as building blocks, the study constructs a strategy-level XGBoost ensemble and evaluates its performance against standalone strategies, an equal-weight baseline, stock-level XGBoost, and the S&P 500 benchmark.

The results suggest that strategy-level learning provides a more effective framework than directly learning stock rankings from sparse stock-level signals. By using strategy outputs as model inputs, the strategy-level approach can capture changes in strategy effectiveness over time and deliver more stable and competitive out-of-sample performance.

Robust model selection is another important finding of this study. The extended grid achieves strong validation performance but does not generalize as well to the test period. This suggests that greater model complexity and a larger search space may increase sensitivity to a specific validation window rather than improve out-of-sample performance. In contrast, the randomized year permutation pipeline reduces dependence on one fixed validation period and produces more consistent cross-period behavior. This indicates that the design of the validation procedure plays an important role alongside the choice of model itself.

The Choose-2 experiment further illustrates the trade-off between diversification and information loss. Although the compact two-strategy subset reduces volatility and drawdown, it does not fully match the performance of the complete six-strategy ensemble. This indicates that the full strategy pool still contains useful complementary information, even though some strategies are more highly correlated with one another.

Overall, this thesis shows that machine learning can be applied not only to stock selection, but also to higher-level strategy allocation. The proposed strategy-level XGBoost framework provides a practical way to combine traditional quantitative investment logic with modern machine learning methods.

## 7.1 Future Directions

Future research can extend this framework in several directions. First, the strategy-level XGBoost ensemble can be applied to other markets, such as the A-share market, to examine whether the framework remains effective under different trading rules, investor structures, and market frictions. In the A-share market, features such as daily price limits, T+1 settlement, and short-selling restrictions may create both additional implementation constraints and new sources of return predictability.

Future work can also expand the strategy pool beyond the six strategies used in this study and explore alternative machine learning models. Additional factors such as quality, low volatility, sentiment, liquidity, and macroeconomic regime indicators may provide richer information for

the ensemble model, while models such as Transformer-based architectures or temporal attention networks may be better suited for capturing time-dependent relationships in strategy performance.

Finally, the backtesting framework can be made more realistic by incorporating market impact, bid–ask spreads, liquidity constraints, and capacity limits. This would provide a more practical assessment of whether the proposed strategy-level framework can be implemented in real trading environments.

# References


[1] E. F. Fama and K. R. French, "Common risk factors in the returns on stocks and bonds," *Journal of Financial Economics*, vol. 33, no. 1, pp. 3–56, 1993.

[2] N. Jegadeesh and S. Titman, "Returns to buying winners and selling losers: Implications for stock market efficiency," *The Journal of Finance*, vol. 48, no. 1, pp. 65–91, 1993.

[3] R. Novy-Marx, "The other side of value: The gross profitability premium," *Journal of Financial Economics*, vol. 108, no. 1, pp. 1–28, 2013.

[4] E. F. Fama and K. R. French, "A five-factor asset pricing model," *Journal of Financial Economics*, vol. 116, no. 1, pp. 1–22, 2015.

[5] M.-Y. Day, C.-Y. Yang, and Y. Ni, "Portfolio dynamic trading strategies using deep reinforcement learning," *Soft Computing*, vol. 28, no. 15–16, pp. 8715–8730, 2024, online first 2023.

[6] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952.

[7] V. DeMiguel, L. Garlappi, and R. Uppal, "Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy?" *The Review of Financial Studies*, vol. 22, no. 5, pp. 1915–1953, 2009.

[8] S. Gu, B. Kelly, and D. Xiu, "Empirical asset pricing via machine learning," *The Review of Financial Studies*, vol. 33, no. 5, pp. 2223–2273, 2020.

[9] S. Emerson, R. Kennedy, L. O'Shea, and J. O'Brien, "Trends and applications of machine learning in quantitative finance," in *Proc. 8th International Conference on Economics and Finance Research (ICEFR)*, Lyon, France, 2019.

[10] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[11] W. F. De Bondt and R. Thaler, "Does the stock market overreact?" *The Journal of Finance*, vol. 40, no. 3, pp. 793–805, 1985.

[12] E. Qian, "Risk parity portfolios: Efficient portfolios through true diversification," PanAgora Asset Management, White paper, September 2005, available at: https://www.panagora.com/assets/PanAgora-Risk-Parity-Portfolios-Efficient-Portfolios-Through-True-Diversification.pdf.

[13] S. Maillard, T. Roncalli, and J. Teïletche, "The properties of equally weighted risk contribution portfolios," *The Journal of Portfolio Management*, vol. 36, no. 4, pp. 60–70, 2010.

[14] F. Black and R. Litterman, "Global portfolio optimization," *Financial Analysts Journal*, vol. 48, no. 5, pp. 28–43, 1992.

[15] O. Ledoit and M. Wolf, "Honey, I shrunk the sample covariance matrix," *The Journal of Portfolio Management*, vol. 30, no. 4, pp. 110–119, 2004.

[16] B. Bruder and T. Roncalli, "Managing risk exposures using the risk budgeting approach," Lyxor Asset Management, Working paper, 2012, available at SSRN: https://ssrn.com/abstract=2009778.

[17] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 3146–3154.

[18] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[19] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[20] L. Cai, Z. He, and C. Zhang, "Combined machine learning for stock selection strategy based on dynamic weighting methods," *arXiv preprint arXiv:2508.18592*, 2025.

[21] P. Yang and R. Lucas, "Dms, ae, daa: methods and applications of adaptive time series

model selection, ensemble, and financial evaluation," *arXiv preprint arXiv:2110.11156*, 2021.

[22] R. Yang, H. Liu, and Y. Li, "An ensemble self-learning framework combined with dynamic model selection and divide-conquer strategies for carbon emissions trading price forecasting," *Chaos, Solitons & Fractals*, vol. 173, p. 113692, 2023.

[23] W. Brock, J. Lakonishok, and B. LeBaron, "Simple technical trading rules and the stochastic properties of stock returns," *The Journal of Finance*, vol. 47, no. 5, pp. 1731–1764, 1992.

[24] A. W. Lo, H. Mamaysky, and J. Wang, "Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation," *The Journal of Finance*, vol. 55, no. 4, pp. 1705–1765, 2000.

[25] M. P. Taylor and H. Allen, "The use of technical analysis in the foreign exchange market," *Journal of International Money and Finance*, vol. 11, no. 3, pp. 304–314, 1992.

[26] R. Cervelló-Royo, F. Guijarro, and K. Michniuk, "Stock market trading rule based on pattern recognition and technical analysis: Forecasting the DJIA index with intraday data," *Expert Systems with Applications*, vol. 42, no. 14, pp. 5963–5975, 2015.

[27] Y. Lin, S. Liu, H. Yang, H. Wu, and B. Jiang, "Improving stock trading decisions based on pattern recognition using machine learning technology," *PLOS ONE*, vol. 16, no. 8, p. e0255558, 2021.

[28] Y. Han, J. Kim, and D. Enke, "A machine learning trading system for the stock market based on N-period Min-Max labeling using XGBoost," *Expert Systems with Applications*, vol. 211, p. 118581, 2023.