Benchmarking Machine Learning Methods for Portfolio Management: Challenges and Opportunities

Anonymous Authors[1]

Abstract

Machine Learning has become a powerful tool in portfolio management over the last decade. However, certain key practical challenges are often overlooked, including market diversity, realistic transaction costs for large trades, and robust testing constraints. This work evaluates the effectiveness and scalability of machine learning methods under more realistic conditions using the stocks that make up the S&P 500 and DJIA. We analyze Reinforcement Learning, Imitation Learning, DAgger and Model Based techniques. To the best of our knowledge, this is the first study to systematically compare all these approaches. Our findings demonstrate that the best methods outperform the annualized return and Sharpe ratio of the standard benchmarks.

1. Introduction

According to standard financial economic theory, investors aim to maximize their utility by achieving an optimal balance between the expected returns and the associated volatility of their investment portfolios (Fabozzi & Markowitz, 2011). To achieve this objective, diversification is essential, as it leverages the variability of asset returns (variances) and their relationships (covariances) to construct portfolios that reduce risk while maintaining expected returns, following modern portfolio theory (Fabozzi & Markowitz, 2011).

In passive investment, resources are allocated across a predefined set of assets, allowing returns to compound over time based on market performance, without frequent trading or active management. However, dynamic portfolio management involves actively adjusting allocations in response to evolving market conditions, leveraging new available data.

Processing available data has become increasingly relevant, as quantitative trading, where portfolio management is a primary focus, now accounts for over 70% of trading volume in developed markets such as the U.S. and Europe, and 40% in emerging markets like China and India (Sun et al., 2023). However, as emphasized by Liu et al. (2018) and Yu et al. (2019) effective strategies can be difficult to find due to the complexity and diversity of market dynamics. Therefore, the ability to process large data quickly and minimize human bias has driven the adoption of machine learning techniques (Chalvatzis & Hristu-Varsakelis, 2020; Chen, 2019).

Despite the growing interest in machine learning methods for portfolio management, various surveys (Dakalbab et al., 2024; Sun et al., 2023; Borkar & Jadhav, 2024) underscore the challenges in comparing different approaches effectively. This is primarily attributed to the lack of consistency across research efforts. While many studies demonstrate improvements in specific areas, they often overlook critical practical constraints that are addressed in other works.

The most frequently overlooked limitations are as follows. Although transaction costs are often incorporated, they ignore the price impact of large trades, which would significantly increase transaction expenses. Furthermore, constraints designed to limit large trades are rarely applied. As a result, agents are often allowed to buy and sell entire holdings overnight at fixed costs, leading to unrealistic returns.

Testing periods are often arbitrarily defined and frequently too short, making it challenging to assess their practical applicability (Sun et al., 2023). Furthermore, the full asset universe is rarely considered, as market representation is often limited to ETFs (e.g., Chalvatzis & Hristu-Varsakelis, 2020). This limits the ability to process stock-specific information effectively, diluting its impact across the index and potentially increasing costs. Additionally, most recent Reinforcement Learning (RL) and Imitation Learning (IL) studies simulate fewer than 30 stocks or ETFs, further reducing their relevance to real-world scenarios (e.g., Liu et al., 2018). This work aims to evaluate the effectiveness and scalability of various general machine learning approaches with realistic settings that could be implemented in practice.

Using the stocks that constitute the Standard & Poor's 500 Index (S&P 500) and the Dow Jones Industrial Average

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

(DJIA), rather than exchange-traded funds (ETFs), which are commonly used for computational simplicity (e.g., Chalvatzis & Hristu-Varsakelis, 2020), we analyze action and weights predictions based on RL and IL. We also employ Model-Based techniques, which, while less commonly applied in portfolio management, are widely utilized in algorithmic trading for single stocks and ETFs (e.g., Adegboye et al., 2023). To the best of our knowledge, this is the first study to compare all of them systematically using constituent stocks, though individual methods have been evaluated against one another.

Our results suggest that it is possible to outperform benchmark indices without using any input data at all. Another key finding is that price prediction approaches are more sensitive to strategy design than to the quality of the predictions themselves, likely due to the dynamic and complex nature of financial markets. Additionally, we find that less explored techniques such as Model-Based and Imitation Learning achieve the best results. Finally, we present alternatives that significantly outperform benchmark indices in terms of both annualized return and Sharpe Ratio.

2. Related Work

The application of machine learning (ML) techniques in portfolio management has become a central theme in recent financial research. Despite having similar objectives, all of these studies focus on quite different things. Wang et al. (2019) developed a risk-adjusted reinforcement learning (RL) method that uses the Sharpe Ratio as the reward function. Herbert (2024) integrated various ML methods for stock price prediction with post-prediction portfolio optimization strategies. In contrast, Xu et al. (2020) focus on directly predicting portfolio weights while introducing a penalty for large trades. Liu et al. (2018) Applied RL with segmented discrete actions within a classic Markov Decision Process (MDP) framework on the Dow Jones index during a bull market. Kong & So (2023) extend this approach by evaluating multiple RL methods with continuous actions across diverse environments, achieving strong results on the KOSPI30 and JPX30. Nevertheless, identifying the most promising methods and output types for further development remains challenging due to the absence of a standardized comparison framework.

Almahdi & Yang (2017) manages a small portfolio of ETFs using an RL agent with the Calmar Ratio as an alternative reward function. Yu et al. (2019) train a model-based agent using inputs from a price forecasting model and explores noise addition for state exploration. Benhamou et al. (2020) investigate the use of noise, macroeconomic features, and partially observable states in portfolio management. Wang et al. (2021) explore alternative reward functions, employing the negative maximum drawdown for portfolios with long and short positions. Huang & Tanaka (2022) propose a modular approach, separating stock prediction from strategy modules to improve portability. Dong & Zheng (2023) investigate imitation learning combined with RL, incorporating expert decisions into the loss function with decreasing influence over time. Caparrini et al. (2024) used ML methods to select an optimal subset of 15 stocks, constructing an equal-weight portfolio. Huotari et al. (2020) employed RL to predict portfolio weights, selecting the top 20 stocks during training and limiting trading to these stocks during testing. Unfortunately, alternative reward functions and state exploration techniques are evaluated in isolation, limiting meaningful comparisons of their impact and potential benefits. Moreover, scalability to portfolios with several hundred stocks remains uncertain.

2.1. Other relevant research

In addition to the studies discussed above, we highlight relevant surveys and work in quantitative trading that broaden the understanding of industry approaches.

Dakalbab et al. (2024) review AI techniques in financial trading, categorizing them by output: price prediction, pattern recognition, portfolio weights optimization, and direct action prediction. Their work underscores the diversity of approaches and AI's adaptability across trading scenarios.

Sun et al. (2023) provide a structured analysis of quantitative trading tasks, categorizing them into algorithmic trading, portfolio management, order execution, and market making. They emphasize the limitations of rule-based methods, particularly their inability to generalize and their sensitivity to distributional shifts. Furthermore, they highlight the challenges inherent in financial markets, characterized by noise and unpredictability, which contribute to the gap between accurate predictions and profitable trading decisions.

Borkar & Jadhav (2024) extend these discussions by identifying key gaps, such as the absence of risk pattern identification in RL systems, the limited advancement of autogenerated strategies, and insufficient knowledge transfer across related domains. The authors highlight the need for more adaptive trading systems.

Several algorithmic trading studies illustrate the application of various action segmentation techniques, threshold-based methods, and other custom strategies (Adegboye et al., 2023; Eilers et al., 2014; Nakano et al., 2018; Dragan et al., 2019; Sermpinis et al., 2019). Other studies have focused on predicting these actions using reinforcement learning agents (Chen, 2019; Ye & Schuller, 2023; Bisi et al., 2020; Zhang et al., 2020). Unfortunately, none of this research offers a comprehensive comparison of different action mappings and post-prediction strategies.

Other relevant studies include Chalvatzis & Hristu-

Varsakelis (2020), which shows that better price forecasting does not always translate to higher profitability. Bao & Liu (2019) provide an in-depth analysis of price impact and transaction costs, examining multi-agent RL behavior in liquidation strategies. Frazzini & Pedersen (2018) conduct a comprehensive empirical study on real-world trading costs and their trends. Finally, Glosten & Milgrom (1985) introduce a theoretical framework for trading with asymmetric information, which remains relevant today.

3. Preliminaries

We model the trading environment as a *Markov Decision Process (MDP)*. An MDP is a tuple $\mathcal{M} = \langle S, A, r, p, \gamma \rangle$ where S is a finite set of *states*, A is a finite set of *actions*, $r : S \times A \times S \to \mathbb{R}$ is the *reward function*, $p(s_{t+1}|s_t, a_t)$ is the *transition probability distribution*, and $\gamma \in (0, 1]$ is the *discount factor*.

The goal is to find an *optimal policy* (Sutton & Barto, 2018). A *policy* $\pi(a|s)$ is a probability distribution over the actions $a \in A$ given a state $s \in S$. The *value* of a policy depends on how much reward it collects from the environment. That is, at each time step t, the environment is in a state $s_t \in S$. Then, an action $a_t \sim \pi(\cdot|s_t)$ is sampled from the policy and executed it. As a result, the state changes to $s_{t+1} \sim p(\cdot|s_t, a_t)$ and an immediate reward $r_{t+1} = r(s_t, a_t, s_{t+1})$ is generated. The process then repeats from s_{t+1}.

The *value function* $v_\pi(s)$ of a policy π in state $s \in S$ is defined as the expected discounted *return* (i.e., sum of rewards) when starting in state s and following π thereafter:

$$v_\pi(s) \doteq \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| S_t = s \right].$$

Then, an optimal policy π_* is defined as a policy that maximizes the value of all states. That is, $v_{\pi_*}(s) \geq v_\pi(s)$ for all state $s \in S$ and possible policies $\pi \in \Pi$.

In most practical applications, the state space is so vast that finding an optimal policy is infeasible. Instead, we settle for finding a policy that generalizes well (Cobbe et al., 2019). To do so, we model the policy $\pi_{\boldsymbol{\theta}}(a|s)$ using a function approximation method—such as a neural network—with parameters $\boldsymbol{\theta}$. We then learn $\pi_{\boldsymbol{\theta}}$ using a set of training states $S_{\text{train}} \subset S$ and evaluate its generalization on a set of testing states $S_{\text{test}} \subset S$, where $S_{\text{train}} \cap S_{\text{test}} = \emptyset$. The learned policy $\pi_{\boldsymbol{\theta}}$ generalized well if its performance is close to optimal when starting in the testing states $s \in S_{\text{test}}$.

To learn $\pi_{\boldsymbol{\theta}}$, we can use Reinforcement Learning (Sutton & Barto, 2018), Imitation Learning (Hussein et al., 2017), Dataset Aggregation (Ross et al., 2011), or a Model-Based Method (Moerland et al., 2023).

4. The Investor's Problem

This section formally defines the dynamic portfolio optimization problem, also known as the *investor's problem* (Cong & Oosterlee, 2017). At a high level, this problem involves an investor with an initial budget of B_0 who has to decide where to invest his wealth. The goal is to maximize his expected wealth within a finite time horizon. To do so, the investor can buy and sell stocks from the market.

We denote by $\mathcal{I}$ the set of *stocks* (i.e., companies available for trading). For each stock in $\mathcal{I}$, there exists a certain number of *shares*, which are the individual units of ownership available for purchase or sale. The (adjusted) close price of every share of i at time step t is denoted by the random variable P_{it}. The holdings of i at time step t, that is, the number of shares of i that we hold at time step t is denoted by h_{it}. The balance (i.e., cash) that is not invested in any stock at step t is denoted by b_t. The *portfolio value* V at time t is given by:

$$V_t = b_t + \sum_{i \in \mathcal{I}} P_{it} \cdot h_{it}, \tag{1}$$

where we assume that we hold no stocks at the initial time step; that is, $h_{i0} = 0$, for every $i \in \mathcal{I}$.

Using previous definitions, we can define our objective: to maximize the portfolio value over a fixed period of time T. This model uses a fixed transaction cost of c, the annualized risk-free rate at step t is denoted by $R_{f,t}$, the number of trading days in a year is denoted by d, and the initial budget is B_0. Finally, we define $\mathcal{T}^+ = \{0..T\}$ and $\mathcal{T}^- = \{0..T-1\}$ to denote the time steps including (or not) the last time step T.

$$\max \mathbb{E}\left[b_T + \sum_{i \in \mathcal{I}} P_{iT} \cdot h_{iT} \right] \tag{IP}$$

$$s.t.\ h_{i,t+1} = h_{it} + a_{it} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}^- \tag{2}$$

$$b_{t+1} = b_t(1 + R_{f,t})^{\frac{1}{d}} - \delta_t \qquad \forall t \in \mathcal{T}^- \tag{3}$$

$$\delta_t = \sum_i P_{it} a_{it} (1 - c) \qquad \forall t \in \mathcal{T}^- \tag{4}$$

$$h_{it}, b_t \geq 0,\ h_{i0} = 0,\ \delta_t \in \mathbb{R} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}^+ \tag{5}$$

$$a_{it} \in \{-K, \cdots, K\},\ b_0 = B_0 \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}^+ \tag{6}$$

In addition to b_t and h_{it}, the model uses the decision variable $a_{it} \in \{-K, \cdots, K\}$ to denote the number of shares the investor buys or sells of stock $i \in \mathcal{I}$ at time step t, where K is the maximum number of full shares that can be bought or sold to keep transaction costs under a realistic fixed transaction cost (Frazzini & Pedersen, 2018).

The objective function is to maximize the expected portfolio value V_T at the last time step T. Constraint (2) updates the number of shares the investor has depending on how many he bought (or sold) in the last time step. Constraints (3) and (4) update the current balance of the investor. These constraints consider that the balance b_t will increase according

to the risk-free rate and could decrease (or increase) depending on whether the investor buys (or sells) shares. Finally, constraints (5) and (6) define the domain of the variables and the initial balance to B_0.

5. Simulation Environment

Modeling the investor's problem (IP) as an MDP is straightforward. The current state $s_t = (h_{it}, b_t)$ is represented by the value of the investor's holdings h_{it} and balance b_t. The actions $a_t = [a_{1t}, \cdots, a_{|\mathcal{I}|t}]$ include all the buying/selling decisions that are feasible in state s_t. The close prices P_{it}, transaction costs c, annualized risk-free rates r_f, and trading days d can be defined using historical data. Then, the reward function can be set as the change in portfolio value:

$$r(s_t, a_t, s_{t+1}) = V_{t+1} - V_t \tag{7}$$

As long as no discount is used (i.e., $\gamma = 1$), an optimal solution to this MDP will also be optimal to (IP).

Unfortunately, directly solving such an MDP is ineffective for finding a policy that generalizes well to unseen situations (i.e., a policy that makes sound financial decisions under market conditions not encountered during training). To achieve this objective, we integrated market features into the state, reduced the agent's action space, and normalized the rewards, as detailed below.

5.1. Generalization and Performance Metrics

In an ideal scenario, we would train a policy $\pi_{\boldsymbol{\theta}}$ utilizing all historical data available up to today and then report our earnings from buying and selling stocks based on $\pi_{\boldsymbol{\theta}}$ moving forward. Although this approach is not practically achievable, our simulator approximates it.

We defined three non-overlapping periods for training, validating, and testing our policies. As proposed by Kong & So (2023), we trained with data from January 2009 to December 2014. We validated using data from January 2015 to December 2015, and tested using data from January 2016 to July 2020. These time periods are sufficiently long to capture bull and bear markets (Pagan & Sossounov, 2003).

Then, regardless of how we use the training and validation periods to learn $\pi_{\boldsymbol{\theta}}$, we report the performance during the testing period using two widely recognized financial metrics. The first metric is the *Annualized Return (AR)*, defined as:

$$\text{AR} = \left(\prod_{t=1}^{T} \frac{V_t}{V_{t-1}} \right)^{\frac{d}{T}} - 1, \tag{8}$$

where d represents the number of trading days in a year (typically $d = 252$). Note that AR can be viewed as a normalized version of V_T (which is maximized by IP). In fact, when $d = T$, AR is equal to $(V_T - V_0)V_0^{-1}$.

The second metric we use is the *Sharpe Ratio (SR)*. This is a key financial metric that measures the excess return of a financial asset or portfolio above the risk-free rate per unit of risk, as quantified by return volatility. As a fundamental measure of risk-adjusted return, it is widely used to evaluate investment performance (Sun et al., 2023). When applied to a portfolio p, the Sharpe Ratio is computed using its annualized return and annualized volatility as follows:

$$\text{SR}_p = \frac{\text{AR}_p - R_f}{\sigma_p}, \tag{9}$$

where R_f is the annualized risk free rate and σ_p is the annualized volatility of the returns of portfolio p.

5.2. Historical Information: DJIA and S&P 500

We utilized two widely recognized benchmarks to simulate our trading environments: the Dow Jones Industrial Average (DJIA) and the Standard and Poor's 500 (S&P 500). These benchmarks are known as stock market indexes (Ryan & Villupuram, 2021; Frazzini & Pedersen, 2018). They track the performance of large companies in the U.S. and use a fixed strategy to define a portfolio across those companies. We note that the performance of these portfolios is quite strong.

Among these, the S&P 500 is widely regarded as the primary benchmark for the market portfolio in Modern Portfolio Theory (MPT). Due to its broad diversification and capitalization-weighted structure, it serves as a theoretical approximation of the market portfolio in Sharpe's Capital Market Theory, where all investors hold a combination of the risk-free asset and the market portfolio (Varian, 1993). Empirical studies consistently show that beating the S&P 500 in terms of both absolute return and Sharpe Ratio is extremely difficult, as active funds and alternative strategies often fail to consistently outperform it after adjusting for risk (French, 2008).

In more detail, the DJIA consists of 30 well-established firms while the S&P 500 tracks 500 large companies across various sectors. We used these benchmarks to create two trading environments. One uses the 30 stocks from the DJIA. The other uses 426 of the 500 stocks from the S&P 500 (we only included companies that were listed in the S&P 500 for the entire training period). Our objective is to apply ML techniques to learn dynamic portfolios that outperform both indices during the testing period.

We note that the DJIA benchmark has been used to evaluate RL methods (e.g., Yang et al., 2020; Kong & So, 2023), but this is the first study to assess RL methods in the S&P 500. Adapting RL techniques for the S&P 500 presents challenges due to its large action space. We observe that supervised learning methods have been evaluated on the S&P 500 benchmark under variants of portfolio manage-

ment (e.g., Jiao & Jakubowicz, 2017; Caparrini et al., 2024), also smaller subsets of the S&P 500 stocks have been used to evaluate performance (Huotari et al., 2020).

Lastly, we conservatively set the transaction cost at 0.2% for each trade. We also defined a limit of $K = 1000$ shares traded per step to ensure realistic transaction costs and to prevent large-volume trades (Frazzini & Pedersen, 2018). And set the initial initial balance to 1 million dollars.

5.3. State, Actions, and Rewards

To learn policies that generalize well to unseen market conditions, we included the current balance b_t, holdings h_{it}, and several stock-specific features along with general market features into the MDP state s_t. Specifically, as stock-specific features, we used factors derived from the Fama-French-Carhart model (Fama & French, 1993; Carhart, 1997), which include momentum, size, value, and profitability, the latter influenced by the gross profitability premium introduced by Novy-Marx (Novy-Marx, 2013). The general market features included the risk-free rates and their temporal variations. For risk-free rates, we utilized the daily returns of 13-week Treasury Bills. In principle, these features should provide sufficient information to learn a policy that can adapt to market changes not encountered during training.

The action $a_t = [a_{1t}, \cdots, a_{|\mathcal{I}|t}]$ encompass all potential buying and selling choices, where $a_{it} \in \{-K, \cdots, K\}$ at time step t. Note that the value of a_{it} indicates the number of shares to purchase (if $a_{it} > 0$) or sell (if $a_{it} < 0$) from stock $i \in \mathcal{I}$ at time step t.

Unfortunately, the complexity of learning an effective policy increases rapidly with larger values of K. To address this issue, previous works have explored four alternatives to constrain the action space:

1. **Direct**: $a'_{it} \in \{-1, 0, 1\}$ (e.g., Adegboye et al., 2023)
2. **Segmented**: $a'_{it} \in \{-5, \cdots, 5\}$ (e.g., Liu et al., 2018)
3. **Continuous**: $a'_{it} \in [-1, 1]$ (e.g., Kong & So, 2023)
4. **Weights**: $a'_{it} \in [0, 1]$ (e.g., Xu et al., 2020)

In all these cases, the agent is still permitted to buy (or sell) up to K shares. To accomplish this, for the first three cases, we linearly map a'_{it} to the $[-K, K]$ interval. Then, the mapped action $a_{it} \in \{-K, K\}$ is executed in the simulator.

In the last case, the action is viewed as a prediction of the optimal *portfolio weights*. The portfolio weights indicate the percentage of the value of the portfolio that is invested in each particular stock. Consequently, the sum of the a'_{it} across all the stocks adds up to 1. Based on these target weights, the simulator internally calculates how much to buy (or sell) of each stock to approach the proposed portfolio weights while complying with the limit of moving up to K shares per time step.

In this environment, agents can choose any actions $a'_{i,t}$, even if those actions are not truly feasible. To closely match the desired target, if a sale is attempted for a quantity that exceeds the available holdings, all current holdings are sold instead. Similarly, if a purchase is made with insufficient funds, the maximum possible amount of the predicted holdings is purchased within the available budget.

Finally, for the training phase, we used the following reward function:

$$r(s_t, a_t, s_{t+1}) = \frac{V_{t+1}}{V_t} - 1 \tag{10}$$

We observe that this reward differs from the ideal reward (7). However, in practice, we achieved better outcomes using (10). We believe this is because the range of this reward function is more constrained, yet it still encourage the agent to maximize its portfolio value at each step.

6. Machine Learning Methods

This section discusses various methods we use to learn policies $\pi_{\boldsymbol{\theta}}$ for solving the investor's problem. These methods are influenced by prior works, as detailed below.

Reinforcement Learning (RL). RL methods learn strong (and sometimes optimal) policies by directly interacting with the environment (Sutton & Barto, 2018). In this study we use Proximal Policy Optimization (PPO) (Schulman et al., 2017). PPO is a state-of-the-art RL method that models $\pi_{\boldsymbol{\theta}}$ using a neural network. Starting from a randomly initialized policy, PPO evaluates the current policy by running several agents in parallel. Then, it updates $\pi_{\boldsymbol{\theta}}$ to increase the probability of selecting actions that led to higher returns and decrease the probability of choosing actions that led to lower returns. This process repeats until PPO reaches a user-defined maximum number of iterations.

Similar to previous work, we will use an RL method to directly learn a policy by interacting with our simulator during the training period (e.g., Dong & Zheng, 2023; Liu et al., 2018; Huotari et al., 2020; Kong & So, 2023). The best model will be selected using the validation period, and we will report the performance of the learned policy during the test period. However, unlike prior methods, we will use PPO – which is known to find policies that generalize well (Cobbe et al., 2020) – and incorporate market features that previous works did not consider (see Section 5.3).

Imitation Learning (IL). IL proposes to learn $\pi_{\boldsymbol{\theta}}$ by mimicking the behavior of experts (Hussein et al., 2017). These methods assume the existence of a training set $\mathcal{T}$ with state-action pairs $(s, a^*) \in \mathcal{T}$, where a^* is an action that an expert would perform in state $s \in S_{\text{train}}$. This training set usually

comes from *expert traces*. That is, we let experts solve the problem and record the actions a^* they performed in every state s they encountered. Once we have the training set $\mathcal{T}$, we can learn $\pi_{\boldsymbol{\theta}}$ using standard supervised learning to maximize the probability of $\pi_{\boldsymbol{\theta}}(a^*|s)$ for all $(s, a^*) \in \mathcal{T}$.

To use IL, we define expert supervision by looking w days into the future. For stocks expected to increase in value beyond a threshold θ_b, the expert purchases them proportionately to the expected increase. For stocks expected to decline in value beyond another threshold θ_s, the expert sells as many as possible.

Formally, let $d_{it} = P_{i,t+w} \cdot P_{i,t}^{-1} - 1$ be the percentage price difference of stock i at step t compared to step $t + w$. Then,

$$a_{it}^* = \begin{cases} \min(K, \lfloor \frac{b_t}{P_{it}} \frac{d_{it}}{\sum_{i \in \mathcal{I}, d_{it} > 0} d_{it}} \rfloor), & \text{if } d_{it} > \theta_b, \\ \max(-K, -h_{it}), & \text{if } d_i < \theta_s, \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

With this technique, we aim for the agent to learn how to leverage state information to determine favorable actions, guided by supervision based on future events.

IL is not commonly used for portfolio management. However, some studies apply IL for pretraining a policy, which is then fine-tuned using RL (e.g., Dong & Zheng, 2023).

Dataset Aggregation (DAgger). Dagger is a variant of IL where the training set $\mathcal{T}$ is increased over time (Ross et al., 2011). A key issue with plain IL is that, at deployment time, the learned policy tends to reach states that are far from the distributions of states that the experts visit. When that occurs, $\pi_{\boldsymbol{\theta}}$ performs erratically. To tackle that problem, DAgger iteratively adds new states to the training data in the following manner. DAgger first learns a policy $\pi_{\boldsymbol{\theta}}$ using standard IL and tests that policy in the environment. Whenever the agent visits a state s_{new} that is not in $\mathcal{T}$, it asks an expert to provide an action a_{new} for the new state and adds the pair $(s_{\text{new}}, a_{\text{new}})$ to $\mathcal{T}$. Then, DAgger continues learning $\pi_{\boldsymbol{\theta}}$ over the increased training set. This process repeats until reaching a maximum number of iterations. As a result, DAgger tends to learn policies that are more robust than those learned by plain IL methods.

Surprisingly, DAgger hasn't been applied to address the investor's problem by previous works. To utilize DAgger, we employed the same supervision as in our IL method.

Model-Based Methods. Finally, we can learn a policy using a model-based approach (Moerland et al., 2023). These methods learn a model of the environment: $\hat{p}(s_{t+1}|s_t, a_t; \boldsymbol{\mu}) \approx p(s_{t+1}|s_t, a_t)$. That is, they learn to predict the next state s_{t+1} (i.e., the future) using the information from the current state s_t (i.e., the present). Then, they use $\hat{p}$ to infer a good policy to solve the problem.

Model-based approaches are commonly used to learn trading strategies (e.g., Dragan et al., 2019; Chalvatzis & Hristu-Varsakelis, 2020; Sermpinis et al., 2019). The key idea is to learn a model, such as a neural network, to predict the future price of each stock. Then, based on the model's predictions, a fixed strategy is used to determine what to buy and sell.

Inspired by those methods, we learn to predict the percentage price differences, defined as $d_{it} = P_{i,t+w} \cdot P_{i,t}^{-1} - 1$. After making predictions for each stock i, we determine an action a_{it} based on a fixed strategy. This strategy can be an optimization algorithm, a threshold-based approach, or a combination of previous methods (Yu et al., 2019).

We evaluate two well-known threshold strategies to determine the final action based on the predictions of $d_{i,t}$ (Adegboye et al., 2023; Dragan et al., 2019; Nakano et al., 2018; Eilers et al., 2014). The first strategy involves building $a_{i,t}$ with two fixed thresholds: one for buying, θ_b, and another for selling, θ_s (e.g., Dragan et al., 2019). The second strategy employs a variable threshold (Adegboye et al., 2023), where θ_b and θ_s increase (or decrease) depending on the proportion of the remaining balance: $b_t \cdot V_t^{-1}$.

7. Performance evaluation

The purpose of this evaluation is to provide actionable insights that guide future research by assessing the effectiveness of different ML methods for portfolio management. The primary metric used for this comparison is the annualized return (AR); however, we also consider the Sharpe Ratio (SR), as it is one of the most widely used measures for evaluating risk-adjusted returns (see Section 5.1).

To mitigate biases in the results, and given the variability and stochastic nature of these techniques, each result reported below is the average of five independent runs of training, validation, and testing (Kong & So, 2023). Due to constraints in computing power, we conducted our comparisons using the DJIA benchmark and assessed the scalability of the best-performing models on the S&P 500 dataset. For context, most models take over a day to process the DJIA dataset. In contrast, processing the S&P 500 dataset can require more than five days, even with substantial computing resources.

We compared Reinforcement Learning (RL), Imitation Learning (IL), and DAgger (DA) using the four action spaces discussed in Section 5.3. For the model-based methods, we trained a Neural Network (NN) and a Random Forest (RF) to predict prices and select actions based on the fixed and variable threshold strategies (see Section 6).

7.1. Results on the DJIA benchmark

Table 1 shows the overall performance of all the methods. First, we note that, generally, all the methods outperform the

Table 1. Machine Learning Techniques

ML TECHNIQUE	AR	SR
RL DIRECT	**13.76%**	**0.66**
RL SEGMENTED	11.95%	0.53
RL CONTINUOUS	10.85%	0.49
RL WEIGHTS	11.69%	0.54
IL DIRECT	13.73%	0.71
IL SEGMENTED	**17.21%**	**0.75**
IL CONTINUOUS	11.59%	0.56
IL WEIGHTS	15.21%	0.73
DA DIRECT	12.31%	0.62
DA SEGMENTED	**15.47%**	0.72
DA CONTINUOUS	11.50%	0.52
DA WEIGHTS	15.22%	**0.73**
RF FIXED THRESHOLD	0.98%	-0.02
RF VARIABLE THRESHOLD	**13.97%**	**0.68**
NN FIXED THRESHOLD	11.10%	0.52
NN VARIABLE THRESHOLD	**13.98%**	**0.64**
DJIA	9.957%	0.57

index in AR and SR. Next, we see that the best performing method is imitation learning using the segmented action space. Recall that this method learns $\pi_{\boldsymbol{\theta}}$ by mimicking the behavior of our expert policy while selecting a discrete action that ranges from -5 to $+5$. This is interesting because, overall, imitation learning has not typically been used to address dynamic portfolio management in the literature.

The second best performing approach was DAgger, also with segmented actions. This result was unexpected. Generally, DAgger tends to outperform imitation learning since it has access to more training data. Perhaps the distribution shift between the training and testing periods affected DAgger more than imitation learning.

Ultimately, the RL methods and model-based approaches (NN and RF) show comparable performance. They surpass the index, yet they are not as effective as IL or DA. However, it's important to mention that model-based methods benefit from significantly faster training times compared to RL.

7.2. Results on the S&P 500 benchmark

We now present our methods' performance in the S&P 500 benchmark. Due to time constraints, we tested only the top-performing methods from the DJIA benchmark. We arbitrarily excluded DA because it was too costly to run, and plain IL appeared to be sufficient for this task.

Table 2 shows the overall results. We note the good performance of the model-based methods. NN with variable threshold outperforms the index by over 9% in AR and 0.3 in SR. Additionally, IL and RL also perform well. To the best of our knowledge, this is the first result demonstrating

Table 2. ML techniques scalability

APPROACH	AR	SR
RL DIRECT	14.57%	0.71
IL SEGMENTED	14.45%	0.71
IL WEIGHTS	15.56%	0.75
DA SEGMENTED	16.55%	**0.89**
DA WEIGHTS	0	0
NN VARIABLE	**19.22%**	0.87
RF VARIABLE	16.25%	0.73
S&P 500	10.28%	0.51

that RL and IL techniques can be scaled up to yield good results in the S&P 500 when using over 400 companies.

7.3. Ablation study

One aspect that caught our attention early in this research was the relatively low performance of the RL. We believed that including more informative features would enhance the performance of the RL method, but this did not have a significant impact either. As such, we considered the possibility that the RL agent might be disregarding the state information and forming a fixed portfolio based on the rewards it receives from the environment.

To test this hypothesis, we repeated the RL experiments using *dummy* observations. This means that, regardless of the state of the environment, the agent always receives the same observation – a constant value of one. Thus, this agent decides what to buy without any awareness of the current state of the environment. We note that this experiment is not entirely futile, because the agent still receives rewards and knows which actions it performs. Therefore, it can learn to consistently choose to buy the same stocks (in some proportions) until exhausting its balance. At that point, the agent earnings will depend on the value of its fixed portfolio.

Table 3 and Table 4 present results from the DJIA and S&P 500 benchmarks. The tables compare the performance of RL using complete information (top rows) versus using the dummy observation (middle rows). Each row indicates a different criterion used to select the best model. "AR" denotes selecting the model with the best AR in validation, "SR" indicates the model with the best SR in validation, and "last" refers to the model learned by the end of training.

From the tables, a key conclusion arises: RL, even without utilizing any information, can identify a portfolio that outperforms the indexes. Furthermore, its performance is competitive with using full information. This suggests a potential pathway for further improving RL. For some reason, RL agents struggle to fully leverage state information when addressing portfolio management tasks. We believe that understanding how to resolve this issue is an important

Table 3. DJIA Dummy state and method selection

METHOD	ANNUALIZED RETURN	SHARPE RATIO
AR RL	**13.76%**	**0.66**
SR RL	**13.76%**	**0.66**
LAST RL	13.21%	0.56
AR DUMMY	8.20%	0.33
SR DUMMY	11.64%	0.51
LAST DUMMY	**12.23%**	**0.58**
DJIA	9,957%	0.57

Table 4. S&P 500 Dummy state and method selection

METHOD	ANNUALIZED RETURN	SHARPE RATIO
AR RL	14.57%	0.71
SR RL	14.36%	0.70
LAST RL	12.32%	0.60
AR DUMMY	14.36%	0.71
SR DUMMY	**14.68%**	**0.72**
LAST DUMMY	13.36%	0.66
S&P 500	10.28%	0.51

direction for future work.

8. Discussion

Table 1 evidences that mapping to continuous actions increases learning complexity compared to discrete alternative. A closer analysis of this mapping reveals that it results in fewer trades, likely because the agent seeks to minimize transaction costs, given the difficulty of predicting exact zero positions (neutral actions). This behavior suggests the agent's preference for cost efficiency over exploration. These findings suggest that developing more effective mapping methods is a promising area for future research.

An interesting result evident in the last four rows of Table 1 is the greater importance of the trading strategy compared to the quality of price predictions or the specific technique used. This becomes significant when considering the inherent difficulty of making accurate price predictions consistently. While a highly accurate predictive model would invalidate this observation, such a model is unlikely given the dynamic nature of financial markets. Therefore, prioritizing the development of portfolio-level strategies is more effective than focusing on the accuracy of individual stock predictions.

Table 1 and Table 2 establish baselines for each technique and dataset, providing a foundation for future research. They also demonstrate that these methods outperform standard benchmarks by effectively leveraging available information. This evidence suggests that machine learning-driven portfolio strategies may identify market inefficiencies, challenging some aspects of the Efficient Market Hypothesis (EMH). This aligns with the increasing dominance of quantitative approaches in financial markets, as evidenced by their significant contribution to total trading volume (Sun et al., 2023).

Table 2 demonstrates that all the techniques discussed are scalable to the S&P 500.The increased asset variety and potential for minor optimizations can provide a greater advantage over the index. However, a significant drawback is the substantial increase in computational requirements. Given that the size of the first and last layers of the neural networks grows exponentially with the dimension, this presents a challenge for efficient implementation.

Tables 3 and 4 demonstrate that it is feasible to outperform market indexes, even without additional data. Notably, the agents in this experiment were optimized for annualized return, but they also outperform relative to the Sharpe Ratio. We strongly advocate for further exploration of this research direction, as it offers a promising way to reduce fund managers' reliance on ETFs or fund-of-funds strategies, which often result in higher costs for end investors.

Last, one of the key advantages of machine learning-driven portfolio management is its cost efficiency relative to traditional human-based trading strategies. Once deployed, algorithmic trading systems require no salaries, bonuses, or performance-based incentives, operating at a fraction of the cost of human traders. In contrast, active portfolio managers and discretionary traders demand high fixed compensation, often supplemented by profit-sharing agreements, significantly increasing the costs of active management.

9. Conclusions and Future Work

This study systematically compared Reinforcement Learning, Imitation Learning, DAgger, and Model-Based techniques to solve the Investor's Problem. We established a realistic framework and demonstrated that achieving better performance than standard indexes is possible with all these techniques. Furthermore, the scalability of these methods was illustrated using the high complexity space of S&P 500 constituent stocks. Finally, we showed that additional data was not necessary to outperform the standard index.

In future work, we propose to integrate variable transaction costs and price impact (Bao & Liu, 2019) into our decision prediction models, moving beyond the current framework that includes transaction limit constraints. This is particularly important for strategies that require large trading volumes, where transaction costs can significantly affect net returns. By incorporating these factors, the model will better reflect real-world market conditions, increasing its practical relevance for large-scale operations, such as those carried out by major hedge funds and asset management firms.

Future work could explore RL methods with reward functions based on risk-adjusted metrics, such as Maximum Drawdown, Calmar Ratio, or Sharpe Ratio.

Impact Statement

This work advances the field of machine learning applied to portfolio management. By demonstrating that machine learning models can outperform traditional benchmarks, this research highlights the potential to reduce reliance on conventional investment vehicles like ETFs or fund-of-funds strategies. With further advancements, this could lead to significant reductions in management costs, improved transparency, and the preservation of voting rights, offering a practical and accessible alternative to conventional investment practices, particularly for smaller investors.

While we recognize the importance of responsible implementation to ensure fair access and address potential disparities, we believe this research is ultimately beneficial to the financial ecosystem. By focusing on developing efficient and accessible investment strategies, it seeks to create value for investors, fund managers, and the broader market as a whole.

References

Adegboye, A., Kampouridis, M., and Otero, F. Algorithmic trading with directional changes. *Artificial Intelligence Review*, 2023. doi: 10.1007/s10462-022-10307-0. Published online: 7 November 2022.

Almahdi, S. and Yang, S. Y. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, 2017. doi: 10.1016/j.eswa.2017.06.023.

Bao, W. and Liu, X.-Y. Multi-agent deep reinforcement learning for liquidation strategy analysis. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, Long Beach, California, 2019. URL https://arxiv.org/abs/1906.11046v1.

Benhamou, E., Saltiel, D., Ungari, S., and Mukhopadhyay, A. Bridging the gap between markowitz planning and deep reinforcement learning. In *Proceedings of the AAAI Conference*, AI Square Connect, France, 2020. Association for the Advancement of Artificial Intelligence.

Bisi, L., Sabbioni, L., Vittori, E., Papini, M., and Restelli, M. Risk-averse trust region optimization for reward-volatility reduction. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2020)*, 2020. Submitted to AAAI 2020.

Borkar, S. and Jadhav, A. Reinforcement learning techniques for stock trading: A survey of current research. *The Journal of Financial Data Science*, 2024. Summer 2024.

Caparrini, A., Arroyo, J., and Mansilla, J. E. S&p 500 stock selection using machine learning classifiers: A look into the changing role of factors. *Research in International Business and Finance*, 70:102336, 2024.

Carhart, M. M. On persistence in mutual fund performance. *The Journal of Finance*, 52(1):57–82, 1997. doi: https://doi.org/10.1111/j.1540-6261.1997.tb03808.x.

Chalvatzis, C. and Hristu-Varsakelis, D. High-performance stock index trading via neural networks and trees. *Applied Soft Computing*, 96:106567, 2020. URL https://doi.org/10.1016/j.asoc.2020.106567.

Chen, X. e. a. Deep robust reinforcement learning for practical algorithmic trading. *IEEE Transactions on Neural Networks and Learning Systems*, 30(3):740–753, 2019.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. In *International conference on machine learning*, pp. 1282–1289. PMLR, 2019.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

Cong, F. and Oosterlee, C. W. Accurate and robust numerical methods for the dynamic portfolio management problem. *Computational Economics*, 49:433–458, 2017.

Dakalbab, F., Talib, M. A., Nasir, Q., and Saroufil, T. Artificial intelligence techniques in financial trading: A systematic literature review. *Journal of King Saud University - Computer and Information Sciences*, 2024. doi: 10.1016/j.jksuci.2024.102015. Accepted: 21 March 2024.

Dong, L. and Zheng, H. Soft imitation reinforcement learning with value decomposition for portfolio management. *Applied Soft Computing*, 2023. doi: 10.1016/j.asoc.2023.111108. URL https://doi.org/10.1016/j.asoc.2023.111108.

Dragan, F., Stoean, C., and Stoean, R. Deep architectures for long-term stock price prediction with a heuristic-based strategy for trading simulations. *PloS One*, 14 (10), 2019. URL https://doi.org/10.1371/journal.pone.0223593.

Eilers, D., Dunis, C. L., von Mettenheim, H.-J., and Breitner, M. H. Intelligent trading of seasonal effects: A decision

support algorithm based on reinforcement learning. *Decision Support Systems*, 2014. doi: 10.1016/j.dss.2014.04.011. © 2014 Elsevier B.V. All rights reserved.

Fabozzi, F. J. and Markowitz, H. M. (eds.). *The Theory and Practice of Investment Management: Asset Allocation, Valuation, Portfolio Construction, and Strategies*, volume 198. John Wiley & Sons, Hoboken, New Jersey, second edition, 2011.

Fama, E. F. and French, K. R. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33:3–56, 1993. Received July 1992; final version received September 1992.

Frazzini, A. and Pedersen, L. H. Trading costs and asset prices: Evidence from the field. *Journal of Financial Economics*, 130(2):263–284, 2018. doi: 10.1016/j.jfineco.2018.06.011.

French, K. R. Presidential address: The cost of active investing. *Journal of Finance*, 63(4):1537–1573, 2008. doi: 10.1111/j.1540-6261.2008.01368.x.

Glosten, L. R. and Milgrom, P. R. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of Financial Economics*, 1985. doi: 10.1016/0304-405X(85)90044-3. URL https://doi.org/10.1016/0304-405X(85)90044-3. Received August 1983, final version received September 1984.

Herbert, G. Ml techniques in portfolio management. Master's thesis, University of Twente, P.O. Box 217, 7500AE Enschede, The Netherlands, 2024. Master Thesis, Business Administration - Financial Management.

Huang, Z. and Tanaka, F. Mspm: A modularized and scalable multi-agent reinforcement learning-based system for financial portfolio management, 2022. URL https://arxiv.org/abs/2102.03502v4.

Huotari, T., Savolainen, J., and Collan, M. Deep reinforcement learning agent for s&p 500 stock selection. *Axioms*, 9(4):130, 2020. doi: 10.3390/axioms9040130.

Hussein, A., Gaber, M. M., Elyan, E., and Jayne, C. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

Jiao, Y. and Jakubowicz, J. Predicting stock movement direction with machine learning: An extensive study on s&p 500 stocks. In *2017 IEEE International Conference on Big Data (Big Data)*, pp. 4705–4713, 2017. doi: 10.1109/BigData.2017.8258518.

Kong, M. and So, J. Empirical analysis of automated stock trading using deep reinforcement learning. *Applied Sciences*, 13:633, 2023. doi: 10.3390/app13010633.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Liu, X.-Y., Xiong, Z., Zhong, S., Yang, H. B., and Walid, A. Practical deep reinforcement learning approach for stock trading. In *NIPS 2018 Workshop on Challenges and Opportunities for AI in Financial Services: the Impact of Fairness, Explainability, Accuracy, and Privacy*, Montréal, Canada, 2018.

Moerland, T. M., Broekens, J., Plaat, A., Jonker, C. M., et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

Nakano, M., Takahashi, A., and Takahashi, S. Bitcoin technical trading with artificial neural network. *Physica A: Statistical Mechanics and its Applications*, 2018. doi: 10.1016/j.physa.2018.07.017. Available online: 20 July 2018.

Novy-Marx, R. The other side of value: The gross profitability premium. *Journal of Financial Economics*, 108(1):1–28, 2013. doi: https://doi.org/10.1016/j.jfineco.2013.01.003.

Pagan, A. R. and Sossounov, K. A. A simple framework for analysing bull and bear markets. *Journal of Applied Econometrics*, 18(1):23–46, 2003. doi: 10.1002/jae.664.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Ryan, P. A. and Villupuram, S. V. Changes in the djia: Market reactions and impact of estimation window. *SSRN*, pp. 1–28, 2021. URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3823523. Date Written: April 9, 2021, Posted: April 12, 2021.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sermpinis, G., Karathanasopoulos, A., Rosillo, R., and de la Fuente, D. Neural networks in financial trading. *Annals of Operations Research*, 2019. doi: 10.1007/s10479-019-03144-y. Published online: 2019.

Sun, S., Wang, R., and An, B. Reinforcement learning for quantitative trading. *ACM Transactions on Intelligent Systems and Technology*, 2023. doi: 10.1145/3582560.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Varian, H. R. A portfolio of nobel laureates: Markowitz, miller, and sharpe. *Journal of Economic Perspectives*, 7 (1):159–169, 1993. doi: 10.1257/jep.7.1.159.

Wang, J., Zhang, Y., Tang, K., Wu, J., and Xiong, Z. Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*, Anchorage, AK, USA, 2019. doi: 10.1145/3292500.3330647. © 2019 Association for Computing Machinery.

Wang, Z., Huang, B., Tu, S., Zhang, K., and Xu, L. Deeptrader: A deep reinforcement learning approach for risk-return balanced portfolio management with market conditions embedding. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*. Association for the Advancement of Artificial Intelligence, 2021.

Xu, K., Zhang, Y., Ye, D., Zhao, P., and Tan, M. Relation-aware transformer for portfolio policy learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 4647–4653, 7 2020. Special Track on AI in FinTech.

Yang, H., Liu, X.-Y., Zhong, S., and Walid, A. Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the first ACM international conference on AI in finance*, pp. 1–8, 2020.

Ye, Z. J. and Schuller, B. W. Human-aligned trading by imitative multi-loss reinforcement learning. *Expert Systems with Applications*, 2023. doi: 10.1016/j.eswa.2023.120939. Published under the CC BY-NC-ND license.

Yu, P., Lee, J. S., Kulyatin, I., Shi, Z., and Dasgupta, S. Model-based deep reinforcement learning for dynamic portfolio optimization. *arXiv preprint arXiv:1901.08740*, 2019.

Zhang, Y., Zhao, P., Wu, Q., Li, B., Huang, J., and Tan, M. Cost-sensitive portfolio selection via deep reinforcement learning. *IEEE Transactions on Knowledge and Data Engineering*, 2020. URL https://arxiv.org/abs/2003.03051. Preprint available at arXiv:2003.03051 [cs.LG].

Table 5. Validation of post-prediction strategies

METHOD	ANNUALIZED RETURN
RF FIXED	**6.49%**
RF VARIABLE	3.96%
NN FIXED	3.81%
NN VARIABLE	2.33%
DJIA	-3,84%

Table 6. DJIA testing only best validation method for each technique

METHOD	ANNUALIZED RETURN	SHARPE RATIO
RL DIRECT	8.26%	0.32
RL DIRECT+	9.07%	0.38
RL SEGMENT	14.55%	0.73
RL SEGMENT+	13.43%	0.69
RL CONT	10.05%	0.41
RL CONT+	10.94%	0.49
RL WEIGHTS	11.05%	0.51
RL WEIGHTS+	11.62%	0.52
IL DIRECT	17.27%	0.86
IL DIRECT+	**19.33%**	**0.98**
IL SEGMENT	15.10%	0.59
IL SEGMENT+	16.03%	0.74
IL CONT	10.34%	0.53
IL CONT+	12.36%	0.60
IL WEIGHTS	7.59%	0.38
IL WEIGHTS+	9.99%	0.48
NN FIXED	8.64%	0.39
NN VARIABLE	14.21%	0.71
RF FIXED	3,09%	0.09
RF +	15.92%	0.75
DJIA	9.957%	0.57

A. Appendix

Additional results

Random Forest and Neural Network validation results can be found in Table 5:

Test results of best overall validation model for each technique instead of using the mean of 5 runs can be found in Table 6

ML hyper-parameters

The hyper-parameters for each ML technique are configured as follows:

Reinforcement Learning (RL): The value and policy functions employ three hidden layers with sizes (512, 512, 256) and (512, 256, 128), respectively. The learning rate is set to 0.003, with an n-step value of 128 and a batch size of 64.

Imitation Learning (IL): Both plain IL and DAgger implementations utilize three hidden layers of size (512, 256, 128), with a learning rate of 0.001.

Random Forest (RF): Randomized seeds are applied, with the number of estimators ranging between 50 and 200, tree depth between 5 and 30, minimum samples per leaf between 1 and 15, and minimum samples for splitting between 2 and 20.

Neural Network (NN): The architecture consists of three hidden layers with sizes (128, 64, 32), a maximum of 1000 iterations, and a randomly sampled learning rate between 0.0005 and 0.002.

Thresholds: For all methods, the buy and sell thresholds are set at approximately 1.5% and -2%, respectively.

Batch Size: A batch size of 32 is used when not specified.

Additional Parameters: All other hyper-parameters retain their default values.