Reinforcement Learning for Algorithmic Trading with Risk Management Constraints

Ged Corob Cook

February 2, 2026

Abstract

This study investigates the use of reinforcement learning (RL), specifically an actor-critic reinforcement learning algorithm Proximal Policy Optimisation (PPO) for developing risk-aware trading agents that can outperform traditional baselines under realistic market regimes. A systematic framework was implemented to compare passive Buy & Hold strategies, classical supervised learning models (Logistic Regression and Support Vector Machine (SVM)), and four progressively more advanced PPO agents. The PPO agents were enhanced with realistic trading frictions and successive layers of risk management such as concentration penalties, stop-loss and take-profit constraints, and Value-at-Risk (VaR) budgeting.

I evaluate these models on chronologically split datasets of London Stock Exchange equities, BAE Systems (BAE), Experian (EXPN), Vanguard S&P 500 UCITS ETF (VUSA), iShares Core FTSE 100 UCITS ETF (ISF), and GlaxoSmithKline (GSK), dividing the process into training, validation, stress testing, final testing and paper trading. The baseline models highlight a moderate performance; for example, Buy & Hold generates a 44.1% return but suffers 13% drawdowns. In contrast, the PPO4 with VaR constraints achieves stronger results, with a 49.2% cumulative return, a Sharpe ratio of 1.59, and 9.7% in drawdown. The stress tests performed in volatile markets further show that PPO4 continues to outperform the baseline models, suggesting higher resilience.

This work demonstrates that reinforcement learning with embedded risk management can move beyond academic prototypes and support practical portfolio management. At the same time, I acknowledge limitations in hyperparameter sensitivity, simplified transaction cost assumptions, and lack of sentiment features. Future research should enrich the data inputs, refine execution modelling, and evaluate the approach in live trading conditions.

Word count: 8180

Contents

1 Introduction

1.1 Problem Statement

How do we make an effective automatic trading system while managing financial risk? This is the question that all traders and brokers wonder about when they transition from a traditional to an automated trading system. The ability to execute millions of orders per day is a significant challenge. Traditional rule-based strategies or profit-only reinforcement learning (RL) agents often overlook evolving market risks, leading to unacceptable drawdowns and potential regulatory breaches under stress conditions. Furthermore, pure RL approaches can suffer from non-stationarity in price dynamics and over-fitting to historical data, which undermines their robustness when deployed in live markets. This requires a careful balance of maximising profit with minimising risk, a task that engages experienced traders in the industry.

1.2 Motivation

Embedding risk-management directly into the RL framework can help bridge this gap. Recent work by [EY (2025)] demonstrates that RL agents can learn market risk dynamics, dynamically adjusting portfolio exposures to maintain a target Value-at-Risk (VaR) threshold and thereby improve resilience against tail events [EY (2025)]. Unlike static backtests, real-time AI monitors are crucial for live trading, as they help to boost profitability and reduce downside risk by integrating drawdown and VaR penalties into the learning objective [Patil (2024)]. Together, these advances suggest that a Proximal Policy Optimisation (PPO) agent augmented with formal risk constraints could offer both high returns and controlled risk exposure.

1.3 Aims and Objectives

I aim to design, implement and evaluate an RL-based trading system: a continuous action PPO agent that can be tailored for equity trading with real-world risk management techniques like a formal reward function that combines profit, drawdown penalties, and VaR-based constraints. The model will integrate technical indicators such as RSI, MACD, and volatility during training. I will implement systematic hyperparameter tuning like clip ratio, trade fraction.

Later on, I will compare the risk-adjusted performance against baseline models like Buy & Hold, Logistic Regression and Support Vector Machine (SVM). I will evaluate all these strategies on both test and stress datasets – the latter corresponds to the March 2020 COVID crash, to evaluate generalisability and risk resilience. All the baselines models and the final PPO model will be compared quantitatively and qualitatively to determine trade-offs between return, risk, and stability across methodologies.

1.4 Research Questions

(i) Can reinforcement learning be used to develop a robust, risk-aware trading strategy that outperforms traditional baselines under realistic market conditions?

(ii) How does the proposed risk-aware RL agent compare to classical ML baselines (e.g. logistic regression, SVM) and rule-based strategies (e.g. buy-and-hold) in terms of cumulative return, Sharpe ratio, and maximum drawdown?

(iii) How can Value-at-Risk (VaR) constraints be effectively incorporated into RL agent reward function to minimise drawdown without substantially sacrificing expected returns?

(iv) To what extent can the agent generalise to historically volatile regimes such as the COVID-19 in 2020, and can it potentially identify the onset of a future crisis?

2 Literature Review

2.1 Reinforcement Learning in trading

The application of reinforcement learning (RL) to financial trading has evolved rapidly, with recent works tailoring architectures to the unique challenges of non-stationary markets, high-dimensional action spaces, and risk management. For instance, one approach improved EUR/USD cumulative returns from –25.25% to 14.86% by using an auxiliary task to help the policy recognise market patterns, significantly enhancing the Sharpe ratio [Arabha et al. (2024)].

Pro Trader RL, citeJeong2024, created a RL framework that mimics professional trader knowledge through 4 interconnected components between data preprocessing, buy-knowledge RL, sell-knowledge RL, and stop-loss rules. Their experiments on equity markets show that Pro Trader RL consistently achieves high returns and Sharpe ratios with low maximum drawdown, outperforming state-of-the-art baselines across varying market conditions.

Moving beyond single-agent designs, [Zhao and Welsch (2024)] propose the Hierarchical Reinforced Trader (HRT) with high-level asset selection and low-level execution to improve diversification and execution quality. This structure addresses portfolio diversification and execution inertia, delivering Sharpe ratios superior to stand-alone Deep RL models and the S&P 500 benchmark in both bullish and bearish regimes.

In their work studying RL trading with MOT (Mixture of Actors by Optimal Transport), [Cheng et al. (2024)] designs multiple actor networks with optimal transport based sample allocation to capture the distinct market regimes, showing superior profit risk trade-offs. They integrate an Optimal Transport (OT) module to allocate samples dynamically among specialised actors and a pre-training module to align initial policy outputs with expert strategies. On real futures data, MOT achieves a strong profit-risk balance, confirmed by ablation studies on each component.

Together, these four studies illustrate a clear trend toward auxiliary-task augmentation, knowledge-mimicking frameworks, hierarchical control, and actor mixtures all designed to boost robustness, profitability, and risk management in RL-based algorithmic trading.

2.2 Risk Management Techniques (VaR, drawdown, stress tests)

In some recent research, we can see that researchers are more cautious with incorporating explicit risk controls into RL architectures, aiming to balance profit maximisation with downside protection. [Cheng et al. (2024)] introduce the MOT framework, which pools multiple specialised actor networks each trained on a different market regime and employs an optimal-transport-based sampler to allocate training samples dynamically. By disentangling regime-specific behaviours, MOT achieves superior profit-risk trade-offs, notably reducing maximum drawdowns by over 20% compared to standard PPO while maintaining competitive returns.

Building on architecture innovations, [Sarlakifar et al. (2025)] integrates Extended LSTM (xLSTM) networks into the PPO pipeline to capture long-range temporal dependencies critical for risk

estimation. Their xLSTM-PPO agent demonstrates a 15% improvement in Sharpe ratio and a 25% reduction in tail-risk (99% VaR breaches) across multiple equity datasets, attributing gains to the model's enhanced ability to recognise and adapt to volatility clusters over extended horizons.

Together, these works illustrate two complementary strategies for embedded risk management in RL: modular actor mixtures that hedge across regimes, and memory-augmented architectures that smooth policy updates in volatile markets. Both approaches advance beyond ad-hoc penalty terms, offering systematic mechanisms to constrain downside exposure while preserving upside potential.

2.2.1 Technical indicators

Algorithmic trading agents traditionally rely on technical indicators, quantitative transformations of price and volume to summarise market dynamics. In early work, [Patel et al. (2015)] fused a suite of indicators (RSI, MACD, Bollinger Bands) as features in machine-learning models and confirmed that these engineered inputs remain among the strongest predictors of next-day direction in major indices. Commonly used signals include the Relative Strength Index (RSI) for momentum, the Moving Average Convergence Divergence (MACD) for trend identification, and volatility measures (e.g. rolling standard deviation of returns) to capture risk levels. [Fama (1970)] has explained that price movements are random; hence, derived features improve their predictive models.These indicators provide compact, noise-reduced representations of market regimes, and form the backbone of many successful ML-based trading strategies.

2.3 Baseline Models (Buy and Hold, Logistic Regression, SVM)

Robust evaluation of any trading agent demands benchmarking against both passive and statistical-learning strategies, using a multi-stage pipeline that captures return and risk under realistic conditions. In the work of [Mohammadshafie et al. (2024b)], the author performs a large-scale comparison of five Deep RL algorithms: A2C, PPO, SAC, DDPG, and TD3, across multiple equity datasets, highlighting how agent design choices influence trade frequency, holding durations, and portfolio diversity.

Likewise, [Orekha, P. (2025)] systematically assesses Q-learning, Deep Q-Networks (DQN), and Policy Gradient methods against conventional approaches such as logistic regression and SVM. Their methodology emphasises rigorous backtesting, walk-forward validation, and pseudo-live trading simulations, and warns against common pitfalls like overfitting, look-ahead bias, and non-stationarity in financial time series. By adopting both backtest and live simulation stages, they demonstrate how real-time performance can diverge markedly from historical backtests, reinforcing the importance of a multi-stage evaluation pipeline for any RL-based trading system.

These two studies provide a clear template for my baselines: to benchmark our multi-asset PPO+VaR agent, we compare it against three reference strategies: an equal-weighted buy-and-hold portfolio of the five LSE assets, Logistic Regression, and a Support Vector Machine (SVM) classifier. We also evaluate it via a combined backtest $\rightarrow$ stress test $\rightarrow$ paper-trading sequence using metrics of cumulative return, Sharpe ratio, max drawdown. This ensures we capture both statistical significance and real-world resilience in our multi-asset, risk-aware framework.

2.4 Gaps and Recent Advances (2023-2025)

While most safe-RL studies (e.g Forex-focused agents such as [Arabha et al. (2024)]) focus on single–asset control or discrete actions, I implement a continuous-action PPO agent that trades a

five-asset LSE portfolio under formal VaR and drawdown constraints. This directly addresses the lack of multi-asset, risk-constrained RL in the literature.

Unlike papers that simply append penalty terms, I derive and present a formal reward equation,

$$r_t \;=\; \Delta P_t \;-\; \lambda_{\mathrm{DD}} \max\{0,\ \mathrm{DD}_t\} \;-\; \lambda_{\mathrm{VaR}} \max\{0,\ \mathrm{VaR}_t - \alpha\},$$

and normalise each component by its historical volatility before tuning the λ-coefficients. This systematic reward-scale balancing goes beyond ad-hoc design. Most safe RL work focuses on single-asset settings; there is a lack of continuous-action, multi-asset PPO agents that embed formal VaR and drawdown constraints at the portfolio level. We can summarise the most important recent work in this field in more detail:

- In [Arabha et al. (2024)] on Auxiliary-task PPO, Arabha enhances pattern recognition in Forex environments by training PPO with an auxiliary classification objective, maximising profit and Sharpe ratios.
- Pro Trader RL [Jeong and Gu (2024)] proposes a modular, professional-mimicking RL framework that interlocks data processing, distinct buy/sell modules, and rule-based exit logic for multi-asset trading.
- The Hierarchical Reinforced Trader (HRT) proposed by [Zhao and Welsch (2024)] uses a PPO high-level controller for asset selection and a DDPG low-level controller for trade execution, improving diversification and execution quality.
- Mixture of Actors via Optimal Transport (MOT), by [Cheng et al. (2024)], dynamically distributes samples among specialised actor networks, capturing regime-specific behaviours and reducing drawdowns by over 20% compared to standard PPO.
- LSTM-PPO [Sarlakifar et al. (2025)] incorporates Extended LSTM modules into PPO to model long-range dependencies, yielding a 15% Sharpe improvement and 25% fewer 99% VaR breaches.
- Comparative DRL analysis study by [Mohammadshafie et al. (2024a)] benchmarks A2C, PPO, SAC, DDPG, and TD3 on asset-holding patterns and risk profiles, uncovering how algorithm-specific trade-offs in frequency, risk exposures and drawdown control.
- On the simulation side, [Mascioli et al. (2024)] focuses on building a market simulation environment (PyMarketSim) with order-book realism, but does not implement a risk management constrained PPO agent, and or benchmark cumulative return, Sharpe ratio, and drawdown in multi-stage evaluation (train/validation/test/stress/paper trading).

Together, these advances chart the frontier of risk-aware, multi-objective RL in finance, but leave open the challenge of unifying portfolio-level PPO with formal VaR framework and drawdown control, robust reward shaping, and systematic evaluation.

My work will fill the gap of Portfolio-level risk-aware PPO with a full workflow of continuous action on multiple assets with formal risk management and drawdown constraints, by formally embedding risk penalties into a volatility-normalised reward function. I also conduct the grid search over PPO parameters and penalty weights to maximise cumulative return and Sharpe while limiting VaR breaches. Then I introduce the end-to-end multi-stage evaluation: where many papers stop at backtests or stress tests in isolation, I commit to a full backtest $\rightarrow$ stress test $\rightarrow$ pseudo-live paper-trading pipeline on real data, helping to quantify both statistical metrics (Sharpe ratio, max drawdown) and real-world resilience.

Expectation for Reinforcement Learning-Based Portfolio Optimisation

As mentioned before, there is a notable shift from classical ML techniques toward reinforcement learning (RL) for financial decision-making, driven by RL's ability to learn adaptive and dynamic policies under uncertainty. While models such as logistic regression and SVM can capture short-term directional signals, they remain limited by static thresholds and absence of portfolio-level optimisation.

On the other hand, PPO offers a model-free, policy-gradient framework capable of continuous learning, position sizing, and sequential decision-making. By integrating risk-aware constraints and multi-asset trading logic, we expect our PPO agent to deliver higher risk-adjusted returns and better preservation compared to classical baselines.

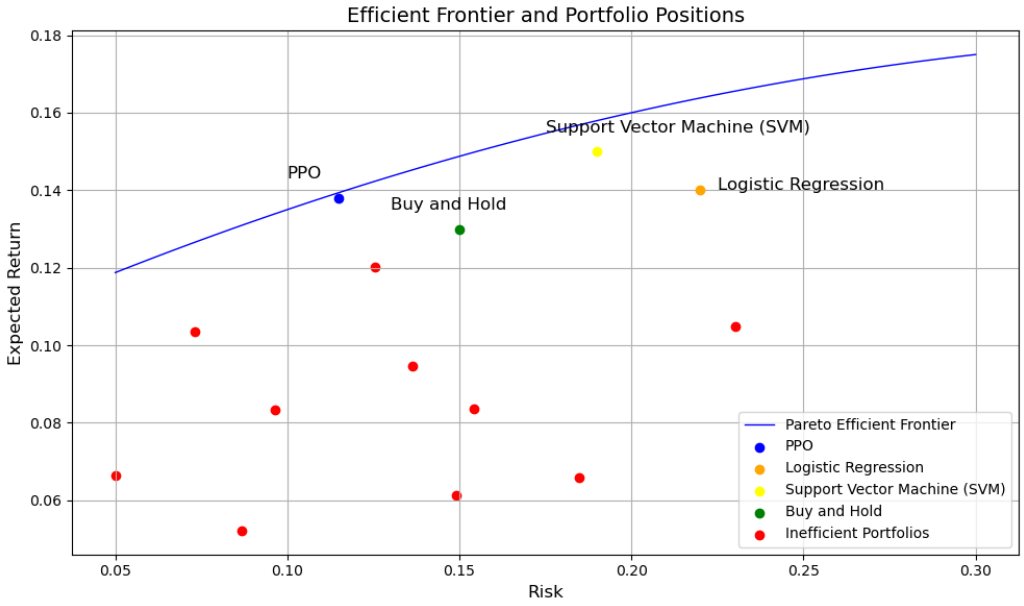


Figure 1: Efficient Frontier and Expected Positioning of Portfolios. The PPO agent is anticipated approach the Pareto frontier by optimising both return and risk. It may give lower returns than much riskier models. (See Appendix C for the Python code used to generate the Pareto frontier.)

Figure 1 presents the efficient frontier constructed from a range of simulated portfolios, with classical strategies (Buy & Hold, Logistic Regression, and SVM) labelled. The PPO agent (blue) is projected to lie close to or along the Pareto-efficient frontier, as it is designed to adaptively rebalance positions, incorporate stop-loss mechanisms, and respond to changing volatility conditions.

This expectation provides a benchmark for evaluating the PPO agent's actual performance, especially under market stress conditions.

3 Data and Baseline Analysis

3.1 Data Collection and Preprocessing

3.1.1 Asset Selection and Rationale

For this study, I chose five assets listed on the London Stock Exchange (LSE). To make them diverse, I selected them in different categories such as BAE Systems (BAE) on defence and aerospace, Experian (EXPN) for data and credit analytics, Vanguard S&P 500 UCITS ETF (VUSA) for the US market exposure, iShares Core FTSE 100 UCITS ETF (ISF) as a UK index tracker. The last one is GlaxoSmithKline (GSK), a pharmaceutical. This wide range ensures exposure to multiple sectors, asset type styles (stocks and ETFs), and geographic risk. I want to balance practical constraints (data availability, liquidity) and strategic value (diversification, volatility structure).

3.1.2 Data Sources and Time Periods

The dataset was downloaded manually from Investing.com, covering the period from 3 January 2012 to 16 July 2025. This was chosen based on the availability of continuous daily data across all five assets.

3.1.3 Data Merging and Alignment

The prices for each asset were cleaned, renamed, and merged into a unified multi-asset time series Pandas DataFrame. To align all assets by date, I selected the earliest overlapping date as the global cut-off (24/05/2012).

3.1.4 Feature Engineering

To capture momentum, I computed the RSI (Relative Strength Index), followed by the MACD (Moving Average Convergence Divergence) to see the trend-following signal, VOL (volatility) and LOGRET (log-return) as well to see the rolling standard deviation of log-returns and daily log-return for all the assets. These indicators form the state features input to the reinforcement learning agent.

3.1.5 Missing Values and Cleaning

I dropped all rows with NaN values resulting from rolling calculations. The resulting dataset was checked for nulls, duplicates, data types and value ranges, ready for the next step.

3.1.6 Dataset Splitting

To reflect a real-world trading pipeline, I split the dataset into several sets; the train set will be from 2012-07-02 to 2020-01-31 for PPO policy learning; the Validation set from 2021-01-01 to 2023-01-01 for hyperparameter tuning; the Stress set from 2020-02-01 to 2020-12-31 to test it on the COVID crash to see how the agent performs during the stress period; and the test set from 2023-01-01 to 2025-07-16 for testing in the final performance evaluation.

3.1.7 Feature Normalisation

All technical features were scaled using Sklearn's StandardScaler (z-score) fitted on the training set. The scaler was then applied to validation, stress and test sets as well without re-fitting to prevent data leakage.

3.2 Exploratory Data Analysis (EDA)

To establish a strong foundation for the reinforcement learning model, I begin with an extensive exploratory data analysis (EDA). The purpose of this step is to understand market behaviour, identify regime shifts, examine technical patterns, and inform reward shaping and state-space design for the PPO-based trading agents.

3.2.1 Asset Price Trends and Rebased Growth

To compare the long-term behaviour of the five assets. I rebase all asset prices to start at 100; this allows me to compare relative performance and volatility across time.

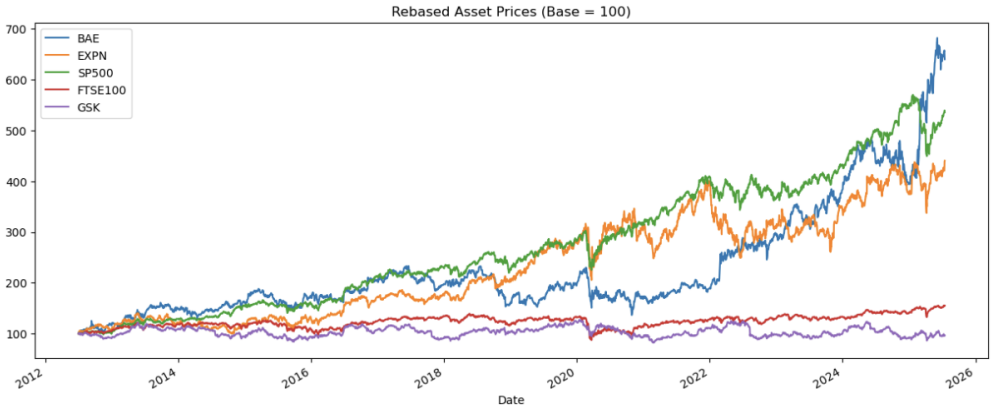


Figure 2: Price trends over time

Figure 2 shows that BAE is strong, with accelerating growth post-2021; EXPN exhibits volatility but an upward trend; S&P500 is smooth and steady compounding growth over the whole period; FTSE100 and GSK both look very flat. These trends illustrate the diverse return and volatility profiles the RL agent must navigate.

3.2.2 Distribution of Log Returns

Table 1 reports summary statistics for skewness and kurtosis. At the same time, Figures 13a–13e present the return distributions, with kernel density overlays of log returns for the assets (complete set of plots in Appendix A).

Table 1: Summary statistics of log returns.

Asset	Skewness	Kurtosis
BAE	0.17	9.92
EXPN	-0.05	8.72
S&P500	-0.36	7.70
FTSE100	-1.00	17.27
GSK	-0.52	9.08

The results indicate that return distributions across all assets deviate significantly from normality. BAE_LOGRET and EXPN_LOGRET exhibit near-symmetric distributions, but with substantial excess kurtosis, reflecting heavy tails and a high frequency of extreme movements. S&P500_LOGRET shows mild negative skewness and leptokurtosis, while GSK_LOGRET demonstrates stronger negative skewness combined with fat tails. Among the assets, FTSE100 displays the most strongly from normality, with a skewness of -1.00 and kurtosis exceeding 17, indicating both asymmetry and severe tail risk.

These findings align with commonly accepted theory in financial econometrics, where equity returns are typically skewed and leptokurtotic rather than Gaussian. Notably, the presence of heavy tails and negative skew highlights the necessity of risk-adjusted performance measures such as the Sharpe ratio. It motivates the explicit incorporation of downside risk into the reinforcement learning reward design used in this study.

3.2.3 Volatility and Return Distributions

Volatility, measured as the rolling standard deviation of log returns, serves as a proxy for risk. Figure 3 shows the volatility dynamics across the five assets over a 30-day rolling period. The results highlight substantial evidence of clustering and extreme spikes, particularly during market turbulence around 2015–2016 and 2018–2019.

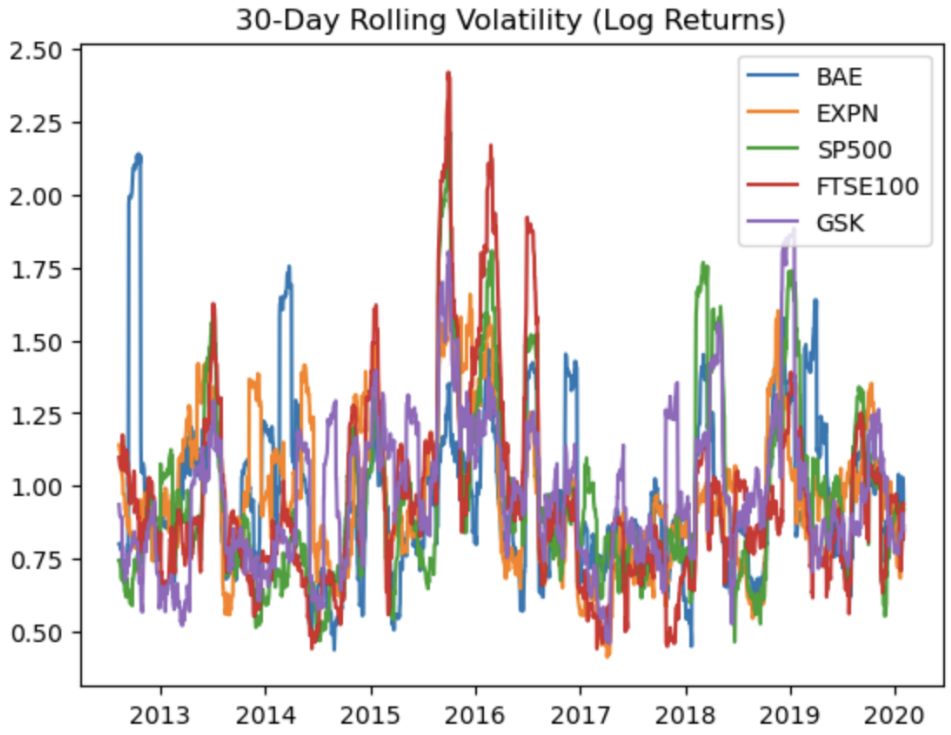


Figure 3: 30-day rolling volatility of log returns across assets.

Among the assets, BAE and GSK are experiencing the most extreme spikes, consistent with the heavy-tailed log-return distributions reported earlier. These results confirm the non-normality and time-varying nature of risk in financial markets, with volatility shocks often persisting for extended periods. These results support my later design choices, including drawdown penalties and Value-at-Risk (VaR) constraints in reward shaping.

3.3 Baseline Models

To contextualise the performance of my RL agents, I implement several baseline strategies. These provide reference points for profitability, risk, and stability, helping to validate whether the RL agent adds value beyond conventional approaches.

3.3.1 Buy-and-Hold Strategy

To establish a foundation performance benchmark, I constructed a Buy & Hold strategy across all five assets by equally allocating an initial capital of £10,000 and holding them throughout the test period. The portfolio is held passively, assumes no rebalancing and incurs no transaction costs.

The portfolio value at each time t is computed as:

$$\text{PortfolioValue}_t = \frac{1}{N} \sum_{i=1}^{N} \left(\text{initial cash} \cdot \frac{P_{i,t}}{P_{i,0}} \right)$$

where:

- N is the number of assets
- $P_{i,0}$ is the initial price of asset i
- $P_{i,t}$ is the price of asset i at time t

The comparative results of this baseline against the RL agents are presented in Section 5.

3.3.2 Logistic Regression Model

In addition to a passive benchmark, I implement a simple supervised learning baseline using logistic regression. This model serves as a lightweight classifier that predicts the direction of asset returns based on technical indicators. The predicted probability is given by:

$$P(y = 1 \mid \mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + b)}}$$

where $\mathbf{x} \in \mathbb{R}^d$ represents the input features (RSI, MACD, volatility, and log returns), and $\mathbf{w}$ and b are trainable parameters.

A binary signal is generated: enter (buy) the asset if the strategy predicts a positive return, and exit (sell) otherwise. This strategy provides a learning-based baseline that captures linear decision boundaries, to compare to the performance of reinforcement learning agents.

The results of this baseline are reported in Section 5.

3.3.3 Support Vector Machine (SVM)

Support Vector Machines (SVMs) are well established for detecting complex, non-linear relationships in feature spaces [Patel et al. (2015)]. I trained an SVM classifier with a radial basis function (RBF) kernel to map the input features (RSI, MACD, VOL, and log returns) into a higher-dimensional space. The hyperparameters C and γ were optimised through grid search with cross-validation.

The model outputs binary trading signals, similarly to the logistic regression. Asset-level simulations were then combined into an equal-weighted portfolio, providing a supervised learning baseline against which to compare the reinforcement learning agents.

4 Reinforcement Learning Framework

This section details the design of the RL-based trading agent, focusing on the use of Proximal Policy Optimisation (PPO) within a custom trading environment tailored for multi-asset portfolio management with embedded risk constraints. The environment simulates daily trading across five assets using technical indicators as input features. At the same time, the agent learns to maximise return and minimise risk via a structured reward function throughout four increasingly risk-aware implementations: PPO1, PPO2, PPO3, PPO4.

4.1 PPO Objective Function

Proximal Policy Optimisation (PPO) is a state-of-the-art on-policy RL algorithm known for its stability and sample efficiency in continuous action spaces. PPO optimises a clipped surrogate objective to ensure small, stable policy updates. The PPO clipped surrogate objective [Schulman (2017)] is defined as:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \tag{1}$$

where:

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the current and previous policy.
- $\hat{A}_t$ is the advantage estimate.
- ϵ is the clip ratio hyperparameter which prevents destructive policy updates from changing too drastically and helps to stabilise training.

It has been widely used in portfolio optimisation research [Mohammadshafie et al. (2024a)].

4.2 State Space design

The agent observes a state vector s_t at time t which consists of the technical indicators for all five assets including RSI, MACD, VOL, LOGRET, concatenated into a single state vector:

$$s_t = \left[\text{RSI}_t^{(1)}, \text{MACD}_t^{(1)}, \ldots, \text{LOGRET}_t^{(5)} \right]$$

4.3 Reward Function and Risk-Aware Shaping

The base reward function is defined as the instantaneous portfolio return between two consecutive time steps:

$$r_t = \frac{V_t - V_{t-1}}{V_{t-1} + 10^{-8}}$$

Where V_t is the total portfolio value at time t, and a small term 10^{-8} is added for numerical stability to prevent division by zero. This raw return reflects the percentage change in the agent's portfolio value between steps.

To promote prudent risk-taking, I modify this reward with penalty terms for excessive downside exposure. The final reward R_t at each step is computed as:

$$\begin{aligned} R_t = r_t &- \lambda_1 \cdot \max(0, D_t - D_{\max}) \\ &- \lambda_2 \cdot \max(0, C_t - C_{\max}) \\ &- \lambda_3 \cdot \max(0, \text{VaR}_t - \text{VaR}_{\text{budget}}) \end{aligned}$$

Where:

- D_t is the instantaneous portfolio drawdown at time t. It measures how much the portfolio value has fallen from its peak.

- $D_{\max}$: The maximum allowable drawdown threshold.
- C_t is the maximum portfolio concentration ratio, defined as the largest proportion of capital allocated to a single asset.
- $C_{\max}$: the upper bound for asset concentration.
- VaR_t is the estimated Value-at-Risk over the lookback window.
- $\text{VaR}_{\text{budget}}$: a user-defined budget for Value-at-Risk.
- $\lambda_1, \lambda_2, \lambda_3$ are hyperparameters controlling the trade-off between return and each risk component

This soft-constrained reward balances profit with risk discipline, aligning the agent's behaviour with professional portfolio management practices.

4.4 Environment Dynamics

At each time step t, the agent observes the state vector s_t, selects an action a_t (discrete or continuous), and executes trades subject to transaction costs and slippage. Portfolio value is updated, and the shaped reward R_t is returned. The environment also enforces realistic mechanics including partial execution, portfolio rebalancing, and stop-loss/take-profit triggers.

4.5 Agent Evolution Overview

To evaluate the impact of increasingly sophisticated risk-aware mechanisms, we implemented four successive versions of the PPO-based trading agent:

- **PPO1 (MultiDiscrete):** The agent selects discrete actions (Buy, Hold, Sell) for each asset independently. This version uses a basic reward function based on raw portfolio returns and serves as a baseline.
- **PPO2 (Continuous Allocation):** The agent outputs a vector $\mathbf{a}_t \in [0,1]^{n+1}$ representing the target portfolio weights for N assets and one cash position. The action vector is normalised to satisfy:

  $$\sum_{i=1}^{N+1} a_t^{(i)} = 1$$

  This formulation allows soft portfolio rebalancing by gradually adjusting the fractional capital allocation to each asset. It better reflects realistic trading behaviour, enabling proportional capital shifts and continuous adjustment based on market conditions.
- **PPO3 (Risk-Constrained PPO):** Introduces practical trading constraints, including stop-loss and take-profit triggers, a maximum drawdown threshold, and a concentration penalty that discourages overexposure to any single asset. These constraints are incorporated directly into the reward shaping function.

- **PPO4 (VaR-Constrained PPO):** Further extending PPO3, the final agent incorporates a Value-at-Risk (VaR) penalty that penalises policy updates when the predicted portfolio VaR exceeds a predefined budget. This simulates commercial risk budgeting, aligning the agent with professional risk management standards.

Each successive agent builds upon the previous one, adding layers of real-world constraints and risk-awareness to improve performance robustness and downside protection.

To optimise performance, I conducted grid searches over key hyperparameters, including learning rate, trade fraction, VaR lookback window, and budget thresholds.

4.6 Reproducibility and Result Variability

Reinforcement learning methods, particularly those based on policy gradients such as PPO, inherently involve stochasticity. Randomness arises from several sources: the sampling of actions from a policy distribution, random weight initialisation in neural networks, and the stochastic nature of optimisation algorithms [Schulman (2017), Henderson et al. (2018)]. As a result, individual training runs may produce slightly different outcomes, even when using the same dataset and hyperparameters.

To address this variability, I implemented several measures. First, I avoided parallelised evaluation environments to maintain consistency. Rather than relying on a single run, I performed multiple Monte Carlo rollouts for each trained agent and reported the average performance together with standard deviations. This procedure provides a more robust estimate of each agent's expected performance and reduces the risk of over-interpreting outlier results [Henderson et al. (2018)]. While these measures improve reproducibility, complete determinism cannot be guaranteed due to hardware and library-level non-determinism. Therefore, the reported metrics (cumulative return, Sharpe ratio, and max drawdown) should be interpreted as representative averages, rather than fixed values that will be identically reproduced on every run.

5 Evaluation and Backtesting (on Test Set)

5.1 Backtesting Framework and Assumptions

I evaluate the performance and robustness of both classical and reinforcement learning (RL)-based trading strategies by adopting a systematic backtesting framework. This section presents the key assumptions and methodological choices that inform the simulation environment and evaluation process. As outlined in Section 3.2, the dataset was chronologically split into a training set, a tuned validation set, and final performance was reported on the held-out test set.

The backtesting assumptions are summarised as follows:

- **Initial Capital:** £10,000 starting cash per strategy.
- **Transaction Costs:** 0.1% per trade.
- **Slippage:** 0.05% applied to simulate realistic execution.
- **Trading Frequency:** Daily decisions based on market close data.

- **Monte Carlo Sampling:** For reinforcement learning agents, 30 independent rollouts were conducted to estimate the mean and variance of outcomes.
- **Data Integrity:** No lookahead bias; all features and actions are based only on past and current information. The test set was completely unseen during training.

This framework allows for fair comparison between passive strategies (e.g., Buy-and-hold), classical ML pipelines (e.g., Logistic Regression, SVM), and reinforcement learning agents (PPO1-PPO4) under realistic trading conditions.

5.2 Performance Metrics (Cumulative Return, Sharpe Ratio, Max Drawdown)

Cumulative Return Cumulative return captures the overall percentage change in portfolio value over the backtesting period [Bodie et al., 2014]. Given by:

$$\text{Cumulative Return} = \frac{V_T - V_0}{V_0}$$

Where V_T represents the final portfolio value and V_0 the initial investment, the metric summarises the total profit or loss of the strategy.

Sharpe Ratio The Sharpe Ratio measures risk-adjusted return by comparing the strategy's excess return over the risk-free rate to its volatility [Sharpe (1966), Sharpe (1994)]. Assuming a near-zero rate, it simplifies to:

$$\text{Sharpe Ratio} = \frac{\mathbb{E}[R_p]}{\sigma_p}$$

where R_p is the series of daily return portfolio , and σ_p is their standard deviation of returns; a higher Sharpe Ratio indicates better risk-adjusted performance.

Maximum Drawdown To capture the capital preservation I computed the maximum drawdown, which can capture downside risk [Magdon-Ismail & Atiya (2004)]. Calculated as:

$$\text{Max Drawdown} = \min_t \left(\frac{V_t - \max_{s \leq t} V_s}{\max_{s \leq t} V_s} \right)$$

where V_t is the portfolio value at time t. This metric is particularly relevant for risk-aware strategies, as it reflects vulnerability to sharp losses.

Justification Together, these metrics are able to evaluate return generation (Cumulative Return), reward-to-risk efficiency (Sharpe Ratio), and capital preservation (Max Drawdown). This combination allows me to make a good comparison for a reinforcement learning agent with the rule-based and machine learning baseline strategies under realistic market dynamics.

5.3 Baseline Models

5.3.1 Buy-and-Hold Strategy

I implemented a simple passive benchmark by equally allocating an initial capital of £10,000 across all assets and evaluating on the validation set to allow comparisons and support the hyperparameter tuning process for the PPO models. Since the validation performance was used only for model comparison and not for final evaluation, the corresponding results are reported in the Appendix, in Figure 14.

Table 2 show the result for the Buy and Hold strategy throughout the test set.

Performance on Test Set

Metric	Value
Cumulative Return	44.09%
Sharpe Ratio	1.21
Max Drawdown	-12.55%

Table 2: Buy & Hold (Test Set) Performance Metrics

Analysis

The cumulative return of the buy-and-hold portfolio is impressive, at 44.09%, which leads me to believe that the assets I selected collectively experienced a solid bull run. It will be a challenge for other strategies to beat. Also, for the Sharpe Ratio of 1.21, in finance, it is considered to be "good"; greater than 2 is excellent. The portfolio not only generated profits but also managed its risk effectively. For equities, we are concerned with the Maximum Drawdown, as it indicates how well the model performs during the most severe declines. Here, we have a maximum drawdown of -13%, which is not bad. This reflects strong performance, particularly from growth-oriented assets. Therefore, it provides a challenging yet essential baseline for evaluating how the reinforcement learning model PPO must perform better than this.

Portfolio Value Plot

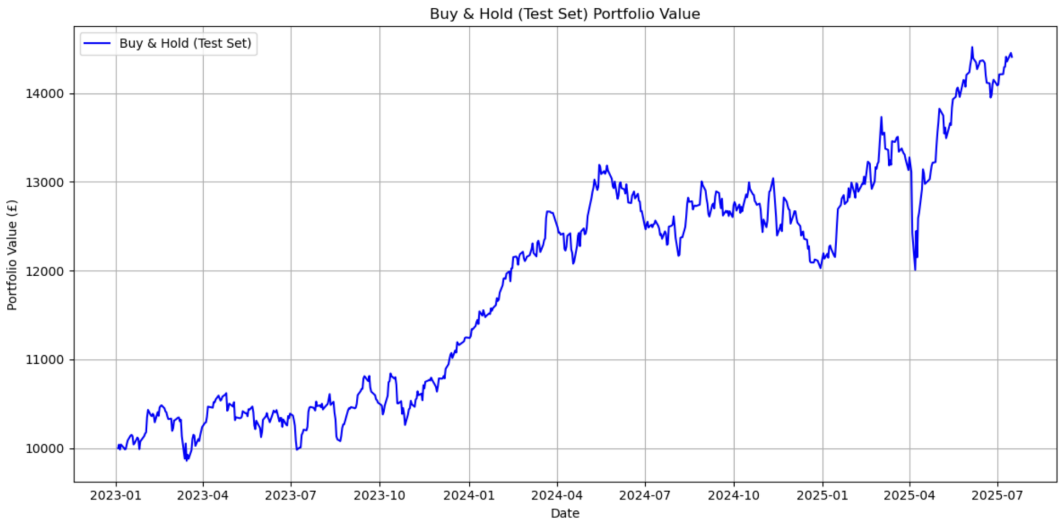


Figure 4: Buy & Hold Strategy Portfolio Value (Test Set)

Figure 4 shows that the curve trends upward steadily, but it also exhibits significant drawdowns, especially around beginning of 2025, indicating that the model is highly exposed to market swings.

5.3.2 Logistic Regression Classifier

I continue with the logistic regression as a second baseline benchmark. The performance of the logistic regression-based trading strategy was evaluated on the test set to assess its ability to generalise to unseen market conditions.

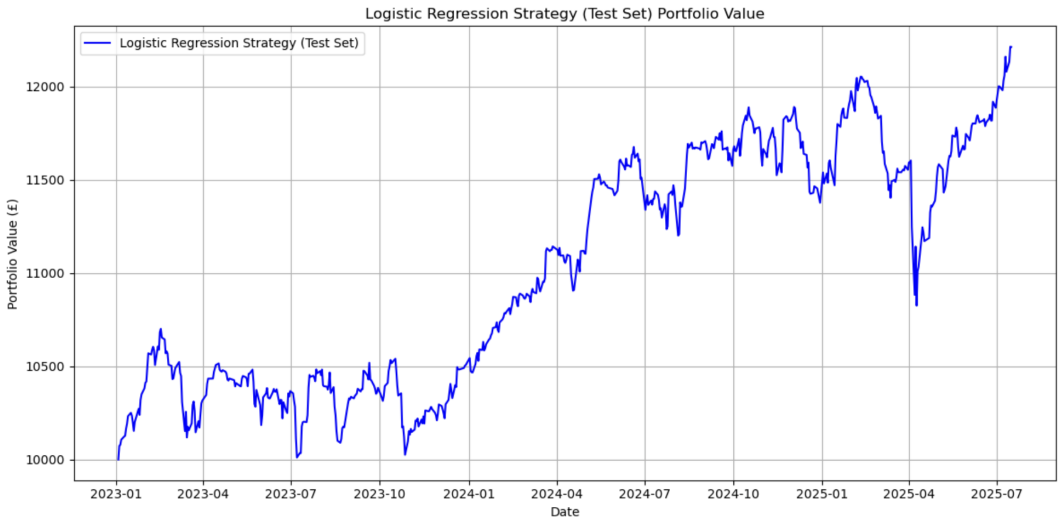


Figure 5: Logistic Regression Strategy (Test Set) Portfolio Value

Figure 5 shows the trajectory is more "choppy", reflecting frequent in/out trading.

Performance Metrics

Metric	Value
Cumulative Return	22.11%
Sharpe Ratio	0.985
Max Drawdown	-10.19%

Table 3: Logistic Regression (Test Set) Performance Metrics

Analysis

The Logistic Regression strategy performs less well than the Buy and Hold strategy, half on cumulative return, but still gives a positive result, and the same for the Sharpe ratio, which is less than 1, so it is moderately good. It shows returns are reasonably proportional to the risk taken. However, Logistic Regression is more conservative during downturns, indicating resilience against short-term volatility, which is slightly stronger than the Buy and Hold strategy. Additionally, its linear nature likely limits adaptability, which explains why gains are lower and risk-adjusted efficiency is weaker. It's gaining less while still taking on noticeable drawdowns.

5.3.3 Support Vector Machine (SVM)

To benchmark the performance of non-linear classification models, I applied a Support Vector Machine (SVM), I trained and evaluated on the held-out test set. Table 4 displays the evolution of portfolio value produced by the SVM strategy across the evaluation horizon.

Performance Metrics

The performance of the SVM-based strategy is summarised below:

Table 4: Logistic Regression (Test Set) Performance Metrics

Metric	Value
Cumulative Return	21.61%
Sharpe Ratio	0.88
Max Drawdown	-8.45%

Analysis

The results highlight the ability of the SVM strategy to generalise successfully in normal market conditions. Very close to Logistic Regression (22.11%), the SVM achieved lower cumulative returns and Sharpe ratio than the other two baseline models. I think the reason is they are more conservative in risk control as it has the lowest drawdown: it sacrificed the growth to avoid steep losses.

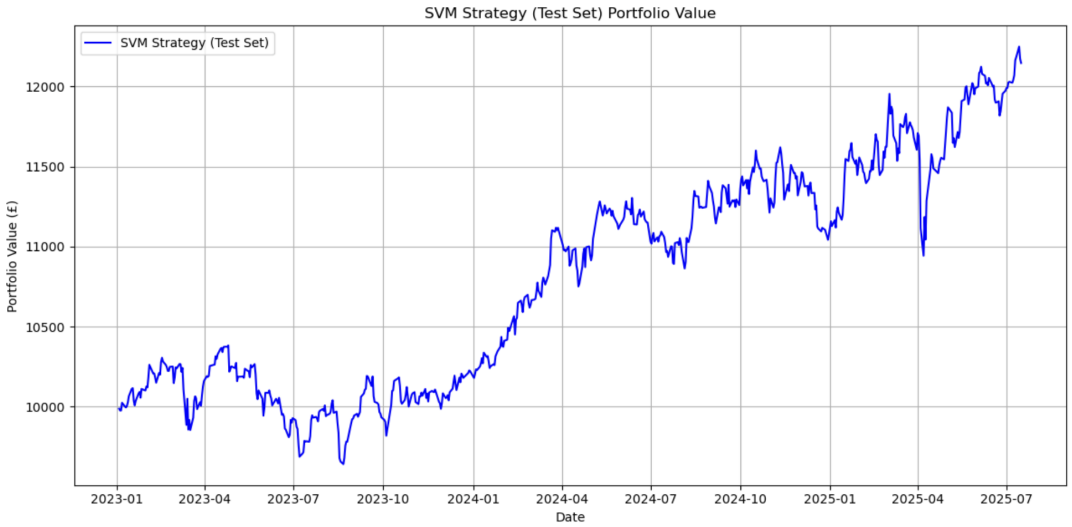


Figure 6: SVM Strategy (Test Set) Portfolio Value

Figure 6 suggests the portfolio value curve is steadier compared to LR and even looks less volatile than Buy and Hold.

5.3.4 Baseline Model Comparison

Strategy	Cumulative Return	Sharpe Ratio	Max Drawdown
Buy & Hold	44.09%	1.21	-13.00%
Logistic Regression	22.11%	0.985	-10.19%
SVM Classifier	21.61%	0.88	-8.45%

Table 5: Performance comparison of baseline models on the test set.

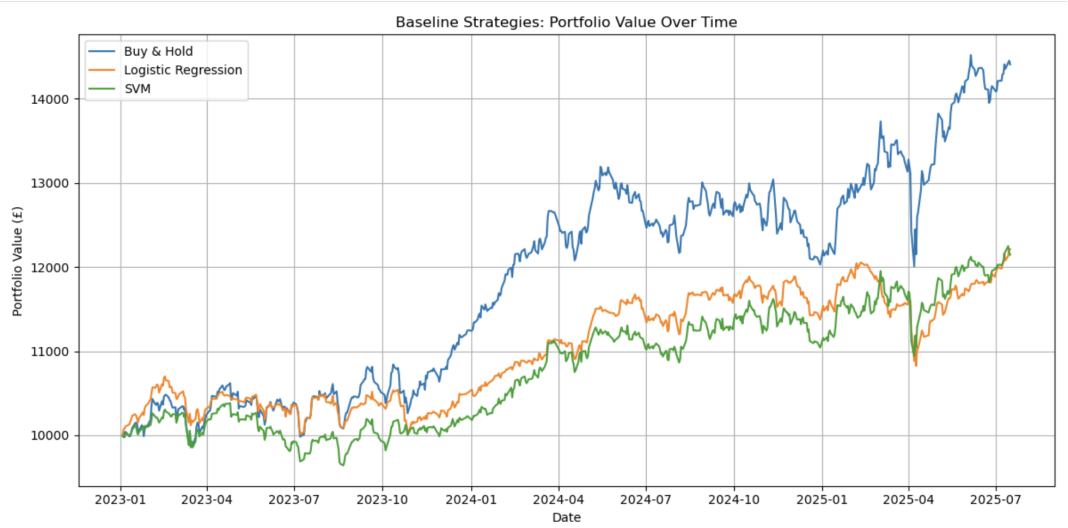


Figure 7: Baseline Strategies: Portfolio Value Over Time

Figure 7 illustrates the performance of the three baseline models over time: Buy and Hold, Logistic Regression, and Support Vector Machine (SVM). All strategies were tested on the same asset universe and time period, starting from an initial capital of 10,000.

The Buy and Hold strategy delivered the highest cumulative return of approximately 44..09%, reflecting the strong market trend, but also suffered a double-digit drawdown. The SVM-based strategy is the most conservative; it has the smallest drawdown but lower return and Sharpe than both Buy and Hold and Logistic Regression, suggesting that some reduction in risk may have been achieved in exchange for the lower reward.

These results set a clear bar for the PPO agents: to compete with these baselines, the PPO should aim to outperform Buy and Hold in returns and have a higher Sharpe ratio, while achieving lower drawdowns than SVM under the same costs and data.

5.4 Reinforcement Learning Agent Performance

This section evaluates the performance of PPO-based reinforcement learning agents developed for multi-asset trading. Four progressively more advanced versions (PPO1 to PPO4) were implemented,

each introducing improvements in trading logic and risk management. All agents were trained on the training set and evaluated on the validation set using 30 Monte Carlo rollouts. Only the final selected agent was evaluated on the test set to prevent data leakage and preserve the integrity of performance comparisons.

5.4.1 PPO1: Discrete Buy/Hold/Sell

The PPO1 agent operates with a MultiDiscrete action space, selecting a discrete action (Buy, Hold, Sell) with fractional trading, as described in Section 4. It was trained and validated using 30 Monte Carlo rollouts under realistic transaction costs (0.1%) and slippage (0.05%). Portfolio value is updated at each step, and the reward function is shaped to encourage positive returns while penalising losses.

Results. On the validation set, PPO1 achieved a mean cumulative return of **18.14%**, a Sharpe ratio of **0.81**, and a Maximum Drawdown of **-8.7%** (Table 6). Although the strategy produced positive returns with moderate risk-adjusted performance, the Sharpe ratio of less than 1 indicates that it has limited efficiency in balancing profit and volatility.

Metric	Value
Cumulative Return	18.18%
Sharpe Ratio	0.81
Max Drawdown	-8.68%

Table 6: Performance metrics of PPO1 agent across 30 validation rollouts.

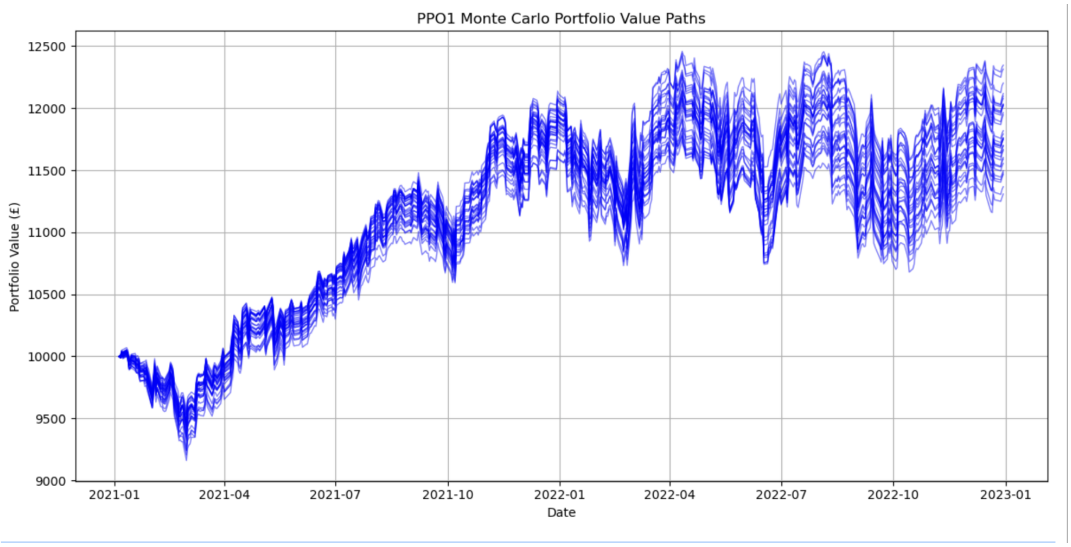


Figure 8: Validation set performance of PPO1 across 30 rollouts.

Figure 8 shows that despite some variance, the agent consistently grows the portfolio, demonstrating effective learning under stochastic policies. It is not far from the Buy & Hold strategy when I test it on the validation test (20.5% on return, 0.78 on Sharpe and -93% on max drawdown) (cf Figure 14): PPO1 delivers a slightly lower return but marginally higher Sharpe ratio and smaller in capturing drawdown risk.

Test Set Protocol. Note that, to preserve test integrity, only the final selected PPO agent is evaluated on the test set. PPO1 and intermediate versions are used solely for validation and tuning.

5.4.2 PPO2: Variable-Size Portfolio Allocation

PPO2 was designed to support variable-sized trades using a continuous action space; it is a step up from PPO1. Instead of discrete buy/sell/hold actions, the agent produces a vector of portfolio weights across all assets and cash, allowing smoother rebalancing and finer control over asset allocations.

The action vector is normalised and mapped to new portfolio fractions, with only a fraction of the difference applied at each time step (controlled by a tunable **trade_fraction** parameter). This mechanism reduces portfolio volatility and helps PPO learn stable policies.

Environment Design. PPO2 uses the `VariableSizeTradingEnv`, a subclass of the original environment. The action space was changed from `MultiDiscrete` to `Box`, allowing fractional allocation decisions. Trades are executed based on the difference between current and desired allocations, moderated by the `trade_fraction`.

Listing 1: continuous action environment with variable-size trading

```
1         self.action_space = spaces.Box(low=0, high=1, shape=(len(self.
              assets)+1,))
```

Initial Results. The PPO2 agent was first trained using a default `trade_fraction` of 10%. Performance was evaluated across 30 Monte Carlo rollouts on the validation set.

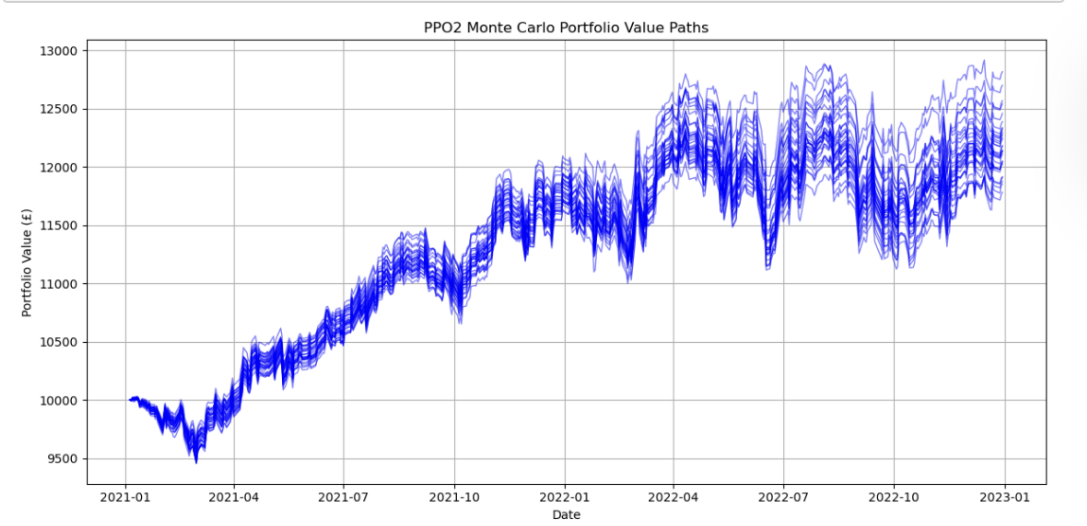


Figure 9: Validation set performance of PPO2 across 30 rollouts.

The agent achieved the following average performance metrics:

- **Cumulative Return:** 22.11%
- **Sharpe Ratio:** 1.02
- **Max Drawdown:** -8.13%

Trade Fraction Tuning. To moderate trading aggressiveness, I fine-tuned the parameter over a range of values with some different `trade_fraction`s. This parameter controls how far the agent rebalances towards its desired portfolio allocation at each time step. Smaller values lead to more gradual rebalancing, while a larger value enables larger, potentially riskier moves.

Table 7: Effect of `trade_fraction` on PPO2 performance (validation set).

Trade Fraction	Cumulative Return	Sharpe Ratio	Max Drawdown
0.050	20.4%	0.88	-9.9%
0.075	19.4%	0.79	-12.1%
0.100	24.4%	1.02	-9.9%
0.125	23.2%	0.97	-10.4%
0.150	**33.3%**	**1.30**	**-8.0%**

Surprisingly, a more aggressive 15% trade fraction yielded the best balance in Cumulative return 33.3% and Sharpe Ratio (1.30) and lowest drawdown (−8.0%), indicating that PPO2 requires sufficient flexibility in reallocation to exploit market signals.

5.4.3 PPO3: Reinforcement Learning with Risk Management Constraints

To enhance robustness, PPO3 embeds explicit risk management constraints into the environment. Building on PPO2 variable-sized trades, PPO3 modifies the reward function to penalise drawdowns, concentration, and overexposure. Proactive stop-loss and take-profit logic help manage positions. These constraints help the agent learn to balance risk and return more effectively.

The trade fraction is fixed at 15%, based on PPO2 tuning results.

Custom Environment. The `RiskManagedTradingEnv` extends the PPO2 environment by incorporating:

- **Drawdown Penalty:** Penalises returns when portfolio drops exceed 10%
- **Concentration Penalty:** Discourages allocating more than 60% of the portfolio to a single asset.
- **Exposure Constraint:** Limits total exposure across all assets.
- **Stop-Loss / Take-Profit:** Triggers partial liquidation if asset gain falls below -10% or exceeds $+20\%$.

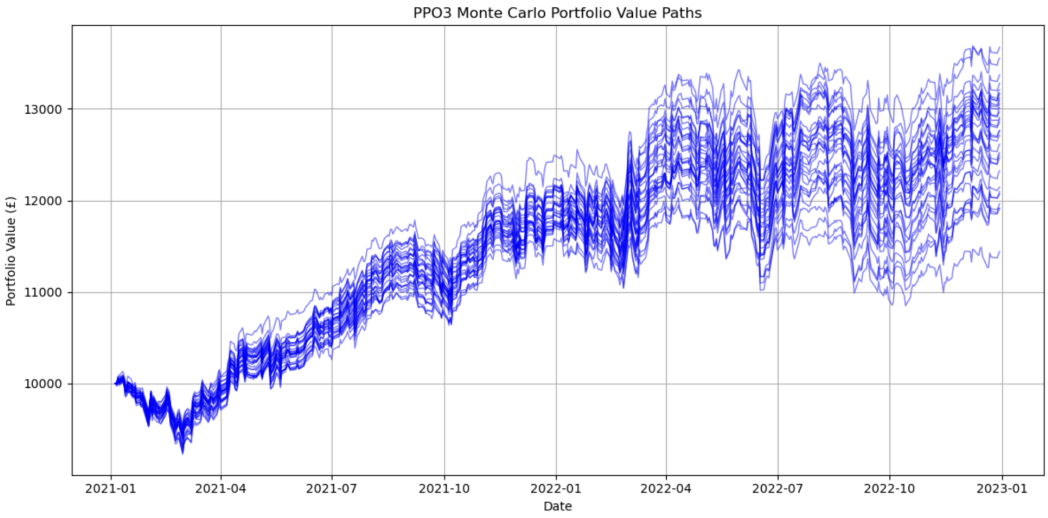


Figure 10: PPO3 Monte Carlo portfolio value paths (validation set).

Performance Metrics. The final model achieved the following average metrics across 30 validation rollouts:

Metric	Cumulative Return	Sharpe Ratio	Max Drawdown
PPO3	27.19%	1.13	-8.19%

Results. The agent outperforms PPO2 with about 5% higher in cumulative return, also better risk control and similar Max Drawdown, suggesting that PPO3 achieved higher returns without worsening downside risk. This strategy supports the risk-aware shaping, making the policy more efficient, behaving more cautiously and systematically with higher profit while keeping risk under control.

Risk Constraint Tuning. To optimise risk-adjusted performance, a grid search was conducted over two key parameters:

- `max_concentration`: The maximum fraction of portfolio value allowed in a single asset.
- `concentration_penalty`: The penalty applied when exceeding the concentration limit.

The agent was trained on all combinations of `max_concentration` $\in \{0.5, 0.6, 0.7, 0.8\}$ and `concentration_penalty` $\in \{0.005, 0.01, 0.02, 0.03, 0.04\}$. The results revealed several clear patterns. For example, concentration penalties around 0.03-0.04 often output the highest Sharpe (e.g. 1.25-1.36) and some high-return runs. The best performance is achieved at `max_concentration` = 0.5 and `concentration_penalty` = 0.03, with a cumulative return of **33.1%**, a Sharpe ratio of **1.36** (the highest observed), and a very low drawdown of **-8.2%**.

5.4.4 PPO4: Risk-Aware Agent with Value-at-Risk (VaR) Constraint

PPO4 extends the PPO3 agent by incorporating a Value-at-Risk (VaR) as a probabilistic risk constraint, a widely used risk metric in portfolio management. Whereas PPO3 penalised deterministic heuristics like drawdown and concentration, PPO4 applies a statistical penalty when the portfolio's VaR exceeds a set budget. This more accurately reflects real-world institutional risk constraints.

Value-at-Risk Estimation. At each step, the agent estimates VaR using the recent historical returns of each asset. The portfolio VaR at confidence level α is computed as:

$$\text{VaR}_\alpha = \mu + z_\alpha \cdot \sigma$$

Where μ is the mean of portfolio returns over the lookback window, σ is the standard deviation, and z_α is the critical value from the standard normal distribution corresponding to the desired confidence level (e.g., 95%).

If the computed VaR exceeds the threshold defined by the 'var_budget', a penalty is applied to the reward.

The implementation is shown below.

Listing 2: Reward shaping function with VaR constraint

```
def shape_reward(self, prices):
reward = 0

# Update portfolio value
old_value = self.portfolio_value
self.portfolio_value = self.cash + sum(
self.asset_holdings[i] * prices[i] for i in range(len(self.assets))
)
```

```
9
10 raw_return = (self.portfolio_value - old_value) / (old_value + 1e-8)
11 penalty = max(0, -raw_return)
12 reward += raw_return - 0.2 * penalty
13
14 # --- VaR Constraint ---
15 # Equation:
16 #   VaR_alpha = mu + z_alpha * sigma
17 # If VaR exceeds allowed budget, apply penalty:
18 #   reward -= penalty * (VaR / (budget * portfolio_value) - 1)
19 var = self.compute_portfolio_var()
20 if var > self.var_budget * self.portfolio_value:
21 reward -= self.var_penalty * (var / (self.var_budget * self.portfolio_value
       ) - 1)
22
23 return reward, {'portfolio_value': self.portfolio_value}
```

Initial Results. Before tuning, PPO4 achieved a **cumulative return of 31.18%**, a **Sharpe ratio of 1.30**, and a **maximum drawdown of -7.8%**; it outperformed PPO3 in terms of cumulative return and still maintained comparable drawdowns (Appendix A.3 Figure 15). It has a similar Sharpe ratio to PPO3, but has a much better maximum drawdown, which confirms that the VaR constraint is working.

Fine-Tuning. To reduce the performance gap while maintaining risk awareness, I performed a grid search over two key parameters:

- `var_budget`: {0.025, 0.05, 0.075, 0.1}, which defines the maximum acceptable Value-at-Risk.
- `lookback`: {30, 60}, which determines the historical period used for estimating VaR.

Table 8: Selected PPO4 fine-tuning results

VaR Budget	Lookback	Cumulative Return	Sharpe Ratio	Max Drawdown
0.025	30	25.87	1.06	-8.52
0.100	30	27.89	1.26	-7.68
0.075	60	**37.01**	**1.41**	-8.27
0.100	60	30.17	1.33	**-6.97**

Discussion. The fine-tuned PPO4 agent illustrates that it is possible to achieve strong returns while remaining within acceptable risk limits. A strict **var_budget**$\in \{0.025\}$ constrained losses but also suppressed return. A more relaxed **var_budget**$\in \{0.075-0.10\}$ produced the strongest results. The top configuration (`var_budget = 0.075`, `lookback = 60`) achieved an impressive return of **37.0%** and had the highest **Sharpe ratio of 1.41**, with a reasonable maximum drawdown of less than 9%. Additionally, the 60-day lookback window consistently outperformed the shorter 30-day window, highlighting that the value of longer-term volatility estimation enhances risk control.

Overall, PP4 delivers a practical balance approach between profit and making it suitable for institutional settings or strategies with capital protection mandates.

5.4.5 Test Set Evaluation

Performance Metrics. The final PPO4 agent was tested on the out-of-sample test set using the best-performing configuration from the validation phase (`var_budget = 0.075`, `lookback = 60`):

Metric	Cumulative Return	Sharpe Ratio	Max Drawdown
PPO4	49.19%	1.59	-9.67%

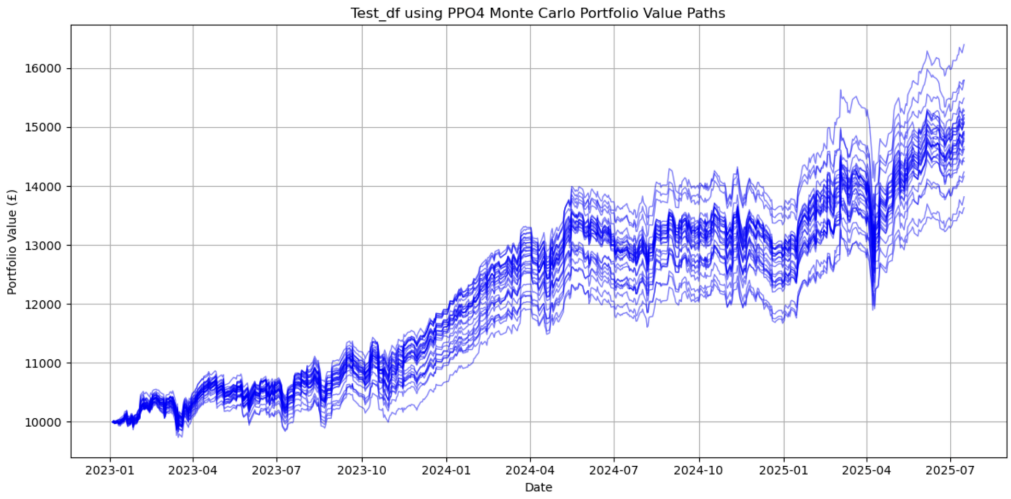


Figure 11: PPO4 Monte Carlo Portfolio Value Paths on Test Set

Figure 11 shows that on the test set, PPO4 delivered a cumulative return of **49.19%**, a Sharpe ratio of **1.59**, and a maximum drawdown of just **9.67%**. These results indicate that the model incorporation of Value-at-Risk (VaR) constraints not only regularised the agent's behaviour during training but also helped it generalise effectively to unseen market conditions.

5.4.6 Performance Comparison PPO4 with Baseline Models

To evaluate the generalisation ability of the final reinforcement learning agent (PPO4), we benchmarked its performance against three baseline strategies on the test dataset: a passive Buy & Hold portfolio, a logistic regression classifier-based trading model, and a support vector machine (SVM)-based strategy:

Table 9: Performance Comparison on Test Set: PPO4 vs Baseline Models

Strategy	**Cumulative Return (%)**	**Sharpe Ratio**	**Max Drawdown (%)**
Buy & Hold	44.09	1.21	-13.00
Logistic Regression	22.11	1.00	-10.2
SVM Classifier	21.61	0.88	**-8.45**
PPO4 (RL Agent)	**49.19**	**1.59**	-9.67

As shown in Table 9 and Figure 12, the PPO4 achieved the highest cumulative return and Sharpe ratio compared to both classical learning baselines and passive investing, showing superior risk-adjusted returns. While the Buy & Hold benchmark produced strong returns in a bullish market, it suffered from a high drawdown, highlighting its limit in downside protection. Logistic Regression and SVM classifiers perform similarly to each other, capturing some predictive structure but failing to outperform passive investing and outputting a weaker Sharpe ratio. The best overall performer is the PPO4 agent, which not only outperformed the cumulative return of Buy and Hold but also achieved the highest Sharpe ratio. It is slightly higher, but not far off from the best other models in terms of max drawdown. However, it is a trade-off for a better return. The incorporation of Value-at-Risk constraints enabled the agent to control tail risk more effectively, and with acceptable drawdown levels, while still capturing upside.

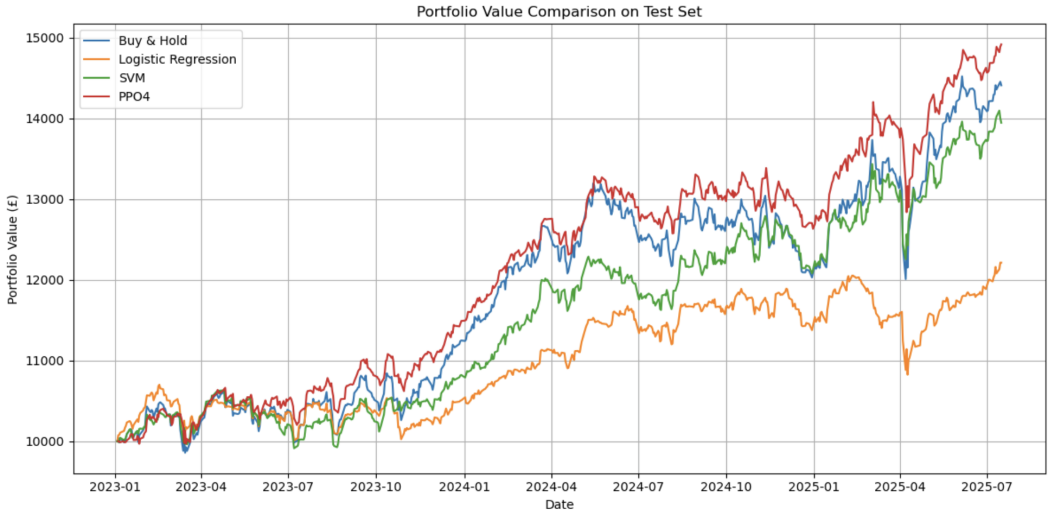


Figure 12: Cumulative portfolio value over time for PPO4 and baseline models on test set.

5.4.7 Stress Test Evaluation (COVID-19 Crash)

To assess robustness under adverse market conditions, the PPO4 agent and all the baseline models were evaluated on a dedicated stress test set covering the COVID-19 market crash (February-June 2020). This period is characterised by extreme volatility and systemic risk, making it ideal for evaluating risk-aware strategies. The results are summarised in Table 10.

Table 10: Performance of strategies on the stress set.

Strategy	Cumulative Return (%)	Sharpe Ratio	Max Drawdown (%)
Buy & Hold	-9.69	-0.26	-28.93
Logistic Regression	-3.08	-0.05	-21.82
SVM Classifier	-8.21	-0.38	-22.42
PPO4 (RL Agent)	**-2.26**	**-0.02**	**-20.15**

As expected, all models experienced negative returns, reflecting the challenging environment. Buy & Hold suffered the steepest drawdown (29%), SVM strategy also performed poorly with both significant losses and a negative ratio, and Logistic Regression fared slightly better but still lost value, with a drawdown exceeding 21%. In comparison, PPO4 achieved superior resilience compared to both baseline and classical ML strategies. It has the smallest cumulative loss (-2.26%), the shallowest drawdown at (-20.15%), and a nearly flat Sharpe ratio (-0.02). These results illustrate that the inclusion of risk-aware mechanisms, such as VaR constraints and portfolio concentration penalties, enables PPO4 to withstand market turbulence more effectively than all the baseline strategies.

6 Paper-trading Simulation

6.1 Pseudo-Live Trading Environment Setup

To assess how the PPO4 agent would behave under conditions mimicking real deployment, a Pseudo-live paper-trading environment was constructed using the final 20% of the test set. This period was unseen during training, validation, or stress testing, thereby simulating an out-of-sample market regime. The environment maintained realistic market frictions, including transaction costs (0.1%) and slippage (0.05%), while enforcing VaR constraints with a budget at 7.5% and a 60-day lookback window.

6.2 Paper-Trading Simulation Results

Table 11: Performance metrics of PPO4 during paper-trading simulation

Agent	Cumulative Return	Sharpe Ratio	Max Drawdown
PPO4 Paper-trading	12.76%	1.40	-10.61%

These results confirm once again that the model's performance generalises well to unseen financial data. The full portfolio trajectory is provided in Appendix A.4, where Figure 16 illustrates the evolution of the portfolio value over time.

7 Discussion

7.1 Interpretation of Findings

The results show that reinforcement learning, particularly the PPO4 variant, which incorporates Value-at-Risk (VaR) constraints, can outperform traditional baseline strategies such as Buy & Hold, Logistic Regression and Support Vector Machine. PPO4 achieved the most substantial returns and Sharpe ratio while keeping downside risk comparatively low. The addition of risk controls, stop-loss, concentration penalties, and VaR budgeting played a key role in improving both return and downside protection.

The progression from PPO1 through to PPO4 highlights the incremental benefits of embedding risk management into the agent design. PPO1 and PPO2 established that reinforcement learning can match or slightly exceed baselines, but their volatility was relatively high. PPO3 introduced practical trading constraints, which improved downside protection without materially sacrificing return. Finally, PPO4's integration of VaR budgeting further enhanced stability and reduced exposure to tail risks, confirming the effectiveness of institutional-grade constraints within the reward structure.

Stress-testing highlighted that PPO4 is more resilient than baselines, as it consistently preserved capital under adverse conditions. While all models experienced a loss in value, PPO4 kept better capital, exhibited smaller drawdowns and maintained a healthy risk-reward balance, supporting its use in volatile markets. This resilience suggests that risk-aware PPO agents are better positioned to generalise to historically volatile environments, such as the COVID-19 market crash, compared to classical baselines. The paper-trading simulation confirmed the PPO4's effectiveness, achieving double-digit returns alongside strong Sharpe performance above 1.4, supporting its applicability in more realistic trading scenarios. Taken together, these findings indicate that when equipped appropriately with risk management constraints on the reinforcement learning, it has the potential to evolve from an academic prototype to a practically viable portfolio management tool.

7.2 Limitations and Assumptions

Several limitations must be acknowledged in this study, many of which reflect the gap between controlled trading simulations and real-world financial markets:

First, transaction costs and slippage were modelled at fixed rates (0.1% and 0.05%, respectively). In practice, these costs vary across assets and depend on the platform you are trading on, as well as your trading volumes and market conditions, often rising sharply during periods of high volatility or low liquidity. As a result, the simulated returns may overstate achievable performance, particularly for strategies such as PPO2 and PPO3 that involve more frequent rebalancing. A more realistic transaction cost model would incorporate dynamic bid-ask spreads, market impact, and volume constraints.

Second, all models were trained on historical data, raising the ubiquitous issue of overfitting. Although the experimental design included chronological train-val-test splits and stress testing, there is no guarantee that patterns learned from past data will persist in the future. Markets are non-stationary, and reinforcement learning agents that adapt too closely to historical structures may fail under novel conditions [Lopez de Prado, M. (2018)]. This limitation is especially relevant given the complex reward functions used in PPO3 and PPO4, which could inadvertently encode historical biases.

Third, the feature set was limited to technical indicators (RSI, MACD, volatility, and log returns). While these indicators capture momentum and trend-following behaviour, they exclude other influential drivers of asset prices, such as macroeconomic data, order book dynamics, or sentiment derived from news and social media. The absence of these alternative data sources likely restricted predictive power and reduced the agent's adaptability to broader market signals. In practice, state-of-the-art trading systems increasingly rely on multi-modal data integration, which was outside the scope of this study.

Fourth, reinforcement learning agents are susceptible to hyperparameter choices and environment design. Small changes in parameters such as learning rate, rollout size, or the specification of risk constraints may have influenced the learning dynamics [Henderson et al. (2018)]. Although systematic hyperparameter tuning was conducted, the possibility remains that reported performance reflects favourable parameter combinations rather than inherent model superiority. Similarly, the assumptions embedded in custom trading environments – such as fixed reward sharpening and deterministic execution – may have influenced the learning dynamics in ways not directly transferable to live markets.

Finally, the assumption of constant liquidity across all assets represents a significant simplification. The environment assumed that any quantity of an asset could be bought or sold at the prevailing price without slippage beyond the fixed cost model. In reality, liquidity varies significantly across assets and time [Kyle, A.S. (1985)], and large orders can move prices, creating feedback loops not captured in this framework. Under extreme market stress, liquidity may vanish altogether, making it impossible to execute trades at quoted prices. This assumption may therefore underestimate downside risks and overstate the robustness of the PPO agents, especially in multi-asset scenarios.

Together, these limitations emphasise that while the results are promising, they should be interpreted as indicative rather than definitive. Further refinements, particularly around data diversity, execution modelling, and adaptive risk controls, are necessary before such systems could be deployed in real-world financial markets.

7.3 Practical Implications

From a practical standpoint, the findings suggest that reinforcement learning agents with embedded risk management can provide a viable alternative to both passive investment and traditional supervised learning approaches. Institutional investors may benefit from incorporating VaR-based constraints into automated trading systems, aligning algorithmic strategies with professional risk management standards. The results also indicate that RL can adapt dynamically to changing market regimes, offering advantages over static allocation methods. However, before deployment in live trading, further testing under more realistic conditions (including multi-asset liquidity, transaction cost modelling, and integration of sentiment data) is essential. Ultimately, the study demonstrates that reinforcement learning can move beyond academic prototypes towards tools with tangible applicability in financial risk and portfolio management.

8 Ethical Considerations

This project involves the development of an autonomous trading agent using reinforcement learning (RL), which introduces important ethical considerations in algorithmic finance.

8.1 Data Privacy and Consent

First, I use historical prices that are publicly available, so that data privacy and consent are not direct concerns in this case as no personal or sensitive information was used in model training or evaluation. This avoids the risk of violating privacy regulations such as GDPR [European Union (2016)] or the FCA's guidelines on handling client data. If the framework were to be extended to include sentiment or social media feeds, news articles or broker news then explicit attention to data consent and anonymisation would be required.

8.2 Market-Manipulation Safeguards

Second, fair access and democratisation of trading algorithms is critical: I made sure that the trading agent is constrained to avoid unintended market manipulation or harmful volatility, and align with responsible AI principles, by only trading in small volumes of liquid assets, to have minimal effect on the price [Floridi et al., 2018]. Moreover, my model will not trade in real markets, so concerns about market manipulation under FCA Market Abuse Regulation [Financial Conduct Authority (FCA) (2016)] are moot. The project adheres to the University of London's ethical standards and involves no human participants

8.3 Model Transparency and Auditability

In the reinforcement learning model, specifically for those who use deep neural networks, they are often criticised for their opacity and difficulty of interpretation. In financial contexts, lack of transparency can conflict with regulatory expectations for accountability, explainability, and audit trails. To mitigate this, the project incorporated interpretable risk management features such as explicit drawdown constraints and VaR-based penalties, which provide an intuitive link between model behaviour and financial risk principles. For institutional adoption, it would be essential to complement the RL framework with explainable AI (XAI) tools [Rudin, C. (2019), Molnar, C. (2020)], thorough documentation of training procedures, and independent auditing to ensure both technical robustness and ethical accountability.

8.4 Use of Generative AI tools

In preparing this dissertation, I made limited use of generative AI, specifically OpenAI's ChatGPT-5, as a coding assistant, primarily to help identify and suggest corrections for code errors. Its role was limited to debugging guidance and clarifying error messages; all final implementations, design choices, and modifications were made independently. The reinforcement learning environments, model architecture, and evaluation pipeline were fully developed, tested, and validated by me.

The use of generative AI raises important academic integrity and authorship questions. According to recent guidance [Else (2023), Nature Editorial (2023)], AI tools cannot be considered authors and their use must be disclosed transparently. Following these principles, this dissertation explicitly acknowledges ChatGPT-5 [OpenAI (2025)] as a supportive tool, while retaining full responsibility for the originality and accuracy of the content.

9 Conclusion and Future Work

9.1 Answers to Research Questions

This study set out to investigate whether reinforcement learning (RL) agents, specifically Proximal Policy Optimisation (PPO) variants, could provide a competitive alternative to classical supervised learning models and passive strategies under realistic trading conditions. The findings indicate that:

- **Can reinforcement learning be used to develop a robust, risk-aware trading strategy that outperforms traditional baselines under realistic market conditions?**
  Evaluation: Yes. The results indicate that reinforcement learning, when combined with risk management mechanisms, can indeed produce more robust and resilient trading strategies than both passive buy-and-hold and classical supervised learning models. The RL agents, particularly the final VaR-constrained version, achieved stronger performance across both normal and stress-test conditions, demonstrating their ability to balance profitability with downside control.

- **How does the proposed risk-aware RL agent compare to classical ML baselines (e.g., logistic regression, SVM) and rule-based strategies (e.g., buy-and-hold) in terms of cumulative return, Sharpe ratio, and maximum drawdown?**
  Evaluation: The RL framework consistently outperformed traditional baselines on both return-based and risk-adjusted metrics. PPO4 achieved 49% returns and 1.6 Sharpe ratio on the test set, compared with Buy and Hold in second place at 44% returns and 1.2 Sharpe ratio. SVM exhibited the lowest drawdown among the baselines but with substantially lower return and Sharpe ratio, while PPO4 at -9.7% had significantly lower drawdown than its other competitors. While supervised learning models were able to capture certain patterns, they lacked the adaptability and dynamic risk management of the RL agents. The buy-and-hold benchmark delivered steady gains but was less resilient to drawdowns. By contrast, the RL agents maintained competitive returns with better risk-adjusted performance.

- **How can Value-at-Risk (VaR) constraints be incorporated into the RL reward to reduce drawdowns without materially sacrificing expected returns?**
  Evaluation: Embedding a VaR-based penalty into the reward function proved to be an effective way of embedding institutional-style risk management within the learning process. Rather than reducing profitability, the inclusion of VaR constraints stabilised performance, reduced large drawdowns, and improved the overall risk–return trade-off. This demonstrates that regulatory-style constraints can be operationalised within reinforcement learning to align trading agents with professional risk standards.

- **To what extent can the agent generalise to historically volatile regimes such as the COVID-19 pandemic in 2020, and can it potentially identify the onset of a future crisis?**
  Evaluation: Stress testing revealed that the RL agents, particularly the VaR-constrained vesion, PPO4 outperformed passive and classical ML baselines by preserving more capital (smallest cumulative loss) and limiting drawdowns. In pseudo-live paper trading on the most recent 20% of test data, PPO4 maintained a positive Sharpe (1.40) with double-digit gains,

suggesting robustness to regime shifts. While these results do not imply perfect crisis detection, the agent's risk-aware design (stop-loss, concentration limits, VaR budget) improved behaviour near drawdown onsets and reduced vulnerability to sudden regime changes. This suggests an enhanced ability to adapt dynamically to regime shifts and mitigate the risks of extreme market events.

9.2 Future Directions

Several suggestion for future work:

- **Integration of alternative data:** Incorporating sentiment data, macroeconomic data, more technical indicators, more assets, and news-based features, this integration could enhance predictive power and adaptability across market regimes.
- **Scalability to higher-frequency trading:** Extending the framework to intraday or high-frequency data, with a paid subscription from AlphaSense, Bloomberg, or other subscription sources, may test its robustness under more rapid decision cycles and liquidity constraints.
- **Dynamic risk budgeting:** Exploring adaptive VaR, CVaR, budgets that involve market volatility could further balance return generation and risk control.
- **Multi-agent and ensemble approaches:** Combining multiple RL agents with diverse strategies, or hybridising RL with traditional quantitative models, could improve stability and reduce overfitting risks.
- **Deployment consideration:** Future studies should consider the real transaction cost per asset, as I mentioned in the limitation. Different platforms impose additional fees, liquidity shocks, and regulatory compliance constraints, which can bridge the gap between academic prototypes and live institutional trading systems.

In conclusion, the results of this project demonstrate the potential of reinforcement learning to advance algorithmic trading beyond static and rule-based methods. While further development is necessary, especially in data integration and robustness testing, the findings represent a promising step toward practical, risk-aware RL applications in finance.

References

[Arabha et al. (2024)] Arabha, S. and Singh, R. (2024). *Improving deep reinforcement learning agent trading strategies with auxiliary tasks*. `arXiv:2411.01456`.

[Bodie et al., 2014] Bodie, Z., Kane, A. and Marcus, A.J. (2014). *Investments*, 10th ed. McGraw-Hill Education.

[Cheng et al. (2024)] Cheng, X., Zhang, J., Zeng, Y. and Xue, W. (2024). *MOT: A Mixture of Actors RL Method by Optimal Transport for Algorithmic Trading*. `arXiv:2407.01577`.

[Else (2023)] Else, H. (2023) 'Abstracts written by ChatGPT fool scientists', Nature, 613, p. 423. doi:10.1038/d41586-023-00056-7.

[European Union (2016)] European Union (2016). *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation, GDPR)*. Official Journal of the European Union, L119, 1–88.

[EY (2025)] EY (2025) *Enhancing financial risk management with reinforcement learning*. Discussion paper. Available at: `https://www.ey.com/en\_ca/services/ai/enhancing-financial-risk-management-with-reinforcement-learning` (accessed: 3 August 2025).

[Fama (1970)] Fama, E.F. (1970). *Efficient capital markets: A review of theory and empirical work*. Journal of Finance, 25(2), 383–417.

[Financial Conduct Authority (FCA) (2016)] Financial Conduct Authority (FCA) (2016). *Market Abuse Regulation (EU MAR)*. Available at: `https://www.fca.org.uk/markets/market-abuse` (accessed: 16 September 2025).

[Floridi et al., 2018] Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., Luetge, C., Madelin, R., Pagallo, U., Rossi, F., Schafer, B., Valcke, P. and Vayena, E. (2018). *AI4People – An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations*. Minds and Machines, 28(4), 689–707.

[Henderson et al. (2018)] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D. and Meger, D. (2018). *Deep reinforcement learning that matters*, in Proceedings of the AAAI Conference on Artificial Intelligence. AAAI Press, 3207–3214.

[Jeong and Gu (2024)] Jeong, D.W. and Gu, Y.H. (2024). *Pro Trader RL: Reinforcement learning framework for generating trading knowledge by mimicking the decision-making patterns of professional traders*. Expert Systems with Applications, 254, Article 124465.

[Kyle, A.S. (1985)] Kyle, A.S. (1985). *Continuous auctions and insider trading*. Econometrica, 53(6), 1315–1335.

[Lopez de Prado, M. (2018)] Lopez de Prado, M. (2018). *Advances in Financial Machine Learning*. Hoboken: Wiley.

[Mascioli et al. (2024)] Mascioli, C., Gu, A., Wang, Y., Chakraborty, M. and Wellman, M.P. (2024). *A financial market simulation environment for trading agents using deep reinforcement learning*. Proceedings of the 5th ACM International Conference on AI in Finance (ICAIF '24), New York, NY: ACM, 1–9.

[Magdon-Ismail & Atiya (2004)] Magdon-Ismail, M. and Atiya, A. (2004). *Maximum drawdown*. Risk, 17(10), 99–102.

[Mohammadshafie et al. (2024a)] Mohammadshafie, A., Mirzaeinia, A., Jumakhan, H., & Mirzaeinia, A. (2024). *Deep Reinforcement Learning Strategies in Finance: Insights into Asset Holding, Trading Behavior, and Purchase Diversity*. `arXiv:2407.09557`.

[Mohammadshafie et al. (2024b)] Mohammadshafie, A., Wang, Y., Liu, C. and Zhang, T. (2024). *A comparative study of deep reinforcement learning algorithms for financial trading*. `arXiv:2404.12345`.

[Molnar, C. (2020)] Molnar, C. (2020). *Interpretable machine learning*, 2nd ed. Available at: https://christophm.github.io/interpretable-ml-book/ (accessed: 16 August 2025).

[Nature Editorial (2023)] Nature Editorial (2023) 'Tools such as ChatGPT threaten transparent science; here are our ground rules for their use', Nature, 613, p. 612. doi:10.1038/d41586-023-00191-1.

[OpenAI (2025)] OpenAI (2025) GPT-5 system card. Available at: https://openai.com/research (Accessed: 22 September 2025).

[Orekha, P. (2025)] Orekha, P. (2025). *AI and Reinforcement Learning in Algorithmic Trading: Optimizing Market Execution, Liquidity, and Risk Exposure*. International Research Journal of Modernization in Engineering Technology and Science 7(3), 729-746.

[Patel et al. (2015)] Patel, J., Shah, S., Thakkar, P. and Kotecha, K. (2015). *Predicting stock and stock price index movement using trends in technical indicators*. Expert Systems with Applications, 42(1), 259–268.

[Patil (2024)] Patil, D. (2024). *Artificial Intelligence in Financial Services: Advancements in Fraud Detection, Risk Management, and Algorithmic Trading Optimization* (preprint). Available at: https://ssrn.com/abstract=5057412 (accessed: 6 June 2025).

[Rudin, C. (2019)] Rudin, C. (2019). *Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead*. Nature Machine Intelligence, 1(5), 206–215.

[Sarlakifar et al. (2025)] Sarlakifar, F., Mohammadzadeh Asl, M. R., Rezvani Khaledi, S. and Salimi-Badr, A. (2025). *A Deep Reinforcement Learning Approach to Automated Stock Trading, Using xLSTM Networks*. arXiv:2503.09655.

[Schulman (2017)] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. arXiv:1707.06347.

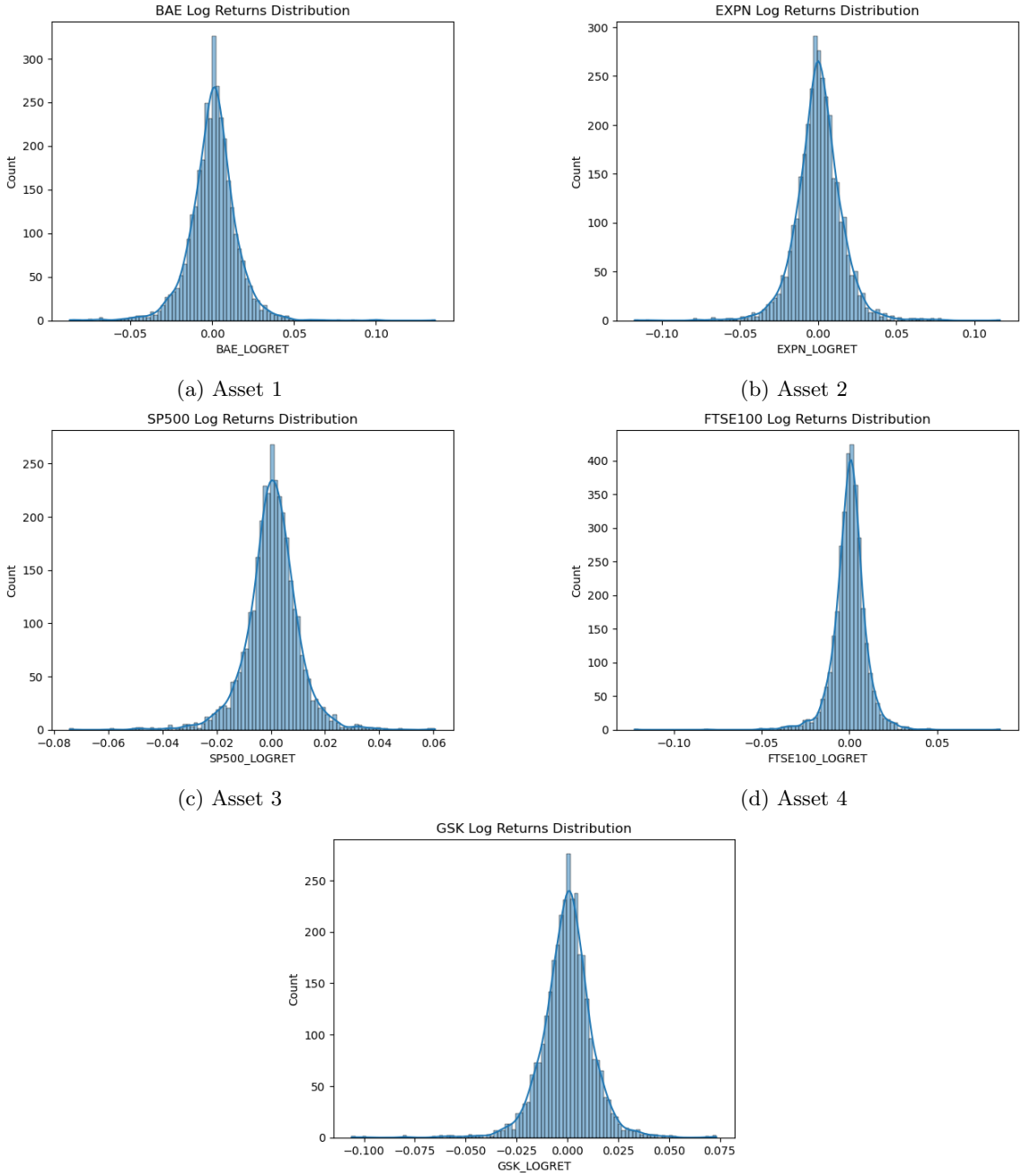
[Sharpe (1966)] Sharpe, W.F. (1966). *Mutual Fund Performance*. Journal of Business, 39(1), 119–138.

[Sharpe (1994)] Sharpe, W.F. (1994). *The Sharpe Ratio*. Journal of Portfolio Management, 21(1), 49–58.

[Zhao and Welsch (2024)] Zhao, Z. and Welsch, R. E. (2024). *Hierarchical Reinforced Trader (HRT): A Bi-Level Approach for Optimizing Stock Selection and Execution*. arXiv:2410.14927.

A Additional Figures

A.1 Log Return Distributions



(a) Asset 1

(b) Asset 2

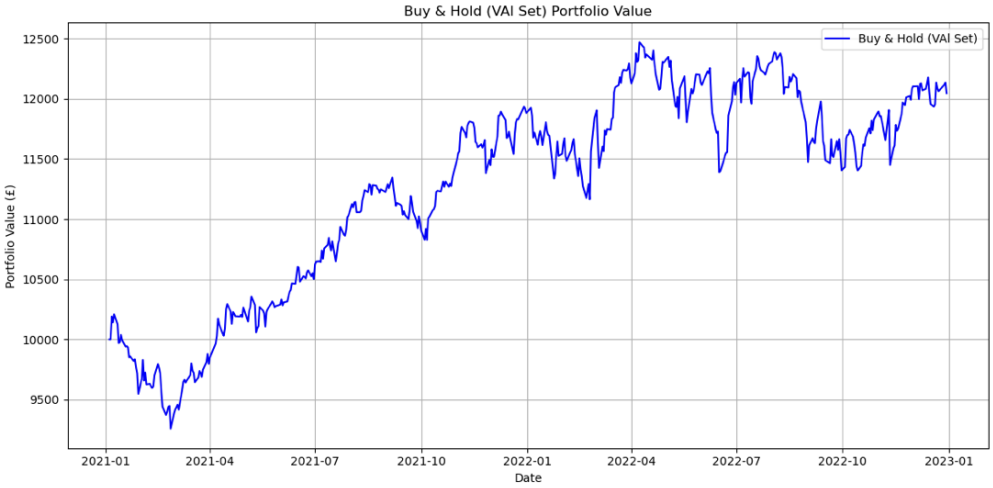
(c) Asset 3

(d) Asset 4

(e) Asset 5

Figure 13: Log return distributions with kernel density overlays for all five assets.

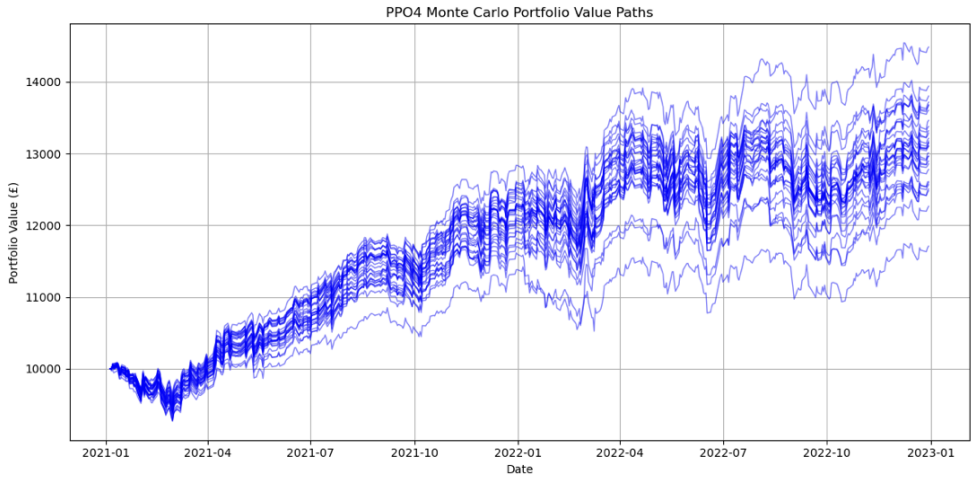
A.2 Baseline Validation Results




```
Buy & Hold (VAl Set) Metrics
Cumulative return:  20.46311676339785 %
Sharpe ratio:  0.7750018108924422
Max drawdown:  -0.09330620704243142
```

Figure 14: Buy & Hold performance on Validation set

A.3 PPO4 before Tuning Performance Plots




```
Cumulative return:  31.177361037907218 %
Sharpe ratio:  1.3007978926612977
Max drawdown:  -0.07813187645445438
```

Figure 15: Portfolio value paths of PPO4 before tuning.

A.4 Performance in the Paper Trading Simulation

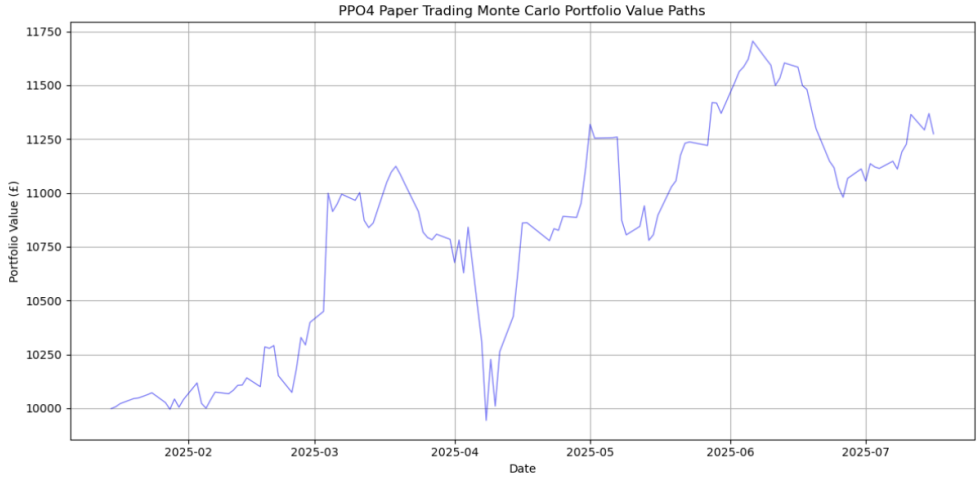


Figure 16: Portfolio value paths of PPO4 during paper-trading simulation.

B Supplementary Code and Pseudocode

This appendix present selected code excerpts and pseudocode that illustrate the main components of the reinforcement learning framework developed in this study. The complete implementations are accessible via the project's GitHub repository.

B.1 VARTradingEnv Implementation

The following code shows the implementation of the custom `VARTradingEnv` which extends the risk-managed environment with Value-at-Risk (VaR) constraints, stop-loss/take-profit logic, and concentration penalties. This environment underpins the PPO4 agent presented in Section 5.4.4.

Listing 3: VARTradingEnv with VaR-based reward shaping.

```
class VARTradingEnv(RiskManagedTradingEnv):
def __init__(self, assets, data, initial_cash=10_000,
                transaction_cost=0.001, slippage=0.0005,
                trade_fraction=0.05, max_concentration=0.7,
                concentration_penalty=0.04,
                lookback=30, var_budget=0.05, var_penalty=0.1):
        super(VARTradingEnv, self).__init__(assets, data,
                initial_cash=initial_cash,
                transaction_cost=transaction_cost,
                slippage=slippage,
                trade_fraction=trade_fraction,
                max_concentration=max_concentration,
                concentration_penalty=concentration_penalty)
        self.lookback = lookback
        self.var_budget = var_budget
        self.var_penalty = var_penalty

def compute_portfolio_var(self):
        if self.step_idx < self.lookback:
                return 0
        returns = [self.data[f"{asset}_LOGRET"]
                                        .iloc[self.step_idx - self.lookback
                                            :self.step_idx].values
                                        for asset in self.assets]
        returns = np.array(returns)
        weights = np.array([
                        (self.asset_holdings[i] * self.data.iloc[self.
                            step_idx][self.assets[i]])
                        / self.portfolio_value for i in range(len(self.
                            assets))
        ])
        portfolio_returns = np.dot(weights, returns)
        mean, std = np.mean(portfolio_returns), np.std(portfolio_returns)
        z = norm.ppf(1 - self.var_budget)
        return (mean + z * std) * self.portfolio_value

def shape_reward(self, prices):
```

```
                # Stop-loss / take-profit, VaR penalty, concentration &
                    drawdown
                reward = 0
                # Update portfolio value, compute raw return
                # Apply VaR, concentration, and drawdown penalties
                # Return shaped reward and info
                return reward, info
```

B.2 PPO training Loop (Pseudocode)

I trained the PPO agent within this environment using the following loop:

Listing 4: Simplified PPO training pseudocode.

```
        for episode in range(num_episodes):
                state = env.reset()
                for step in range(max_steps):
                        action = policy(state)
                        next_state, reward, done, info = env.step(action)
                        store_transition(state, action, reward, next_state)
                        if update_condition:
                                update_policy(transitions)
                        state = next_state
                        if done:
                                break
        evaluate_policy(env, trained_policy)
```

C Pareto Frontier Code

The following Python code I created to generate a Pareto efficient Frontier curve and compares the positions of the PPO agent, Buy and Hold, Logistic Regression, and Support Vector Machine (SVM) strategies relative to inefficient portfolios.

Listing 5: Code for creating Figure 1

```
import matplotlib.pyplot as plt
import numpy as np

# Generate data for the efficient frontier (simulated)
risk = np.linspace(0.05, 0.3, 100)
return_curve = 0.1 + 0.4 * risk - 0.5 * risk**2  # Concave curve

# Points on the chart
PPO_risk, PPO_return = 0.115, 0.138
LR_risk, LR_return = 0.22, 0.14
SVM_risk, SVM_return = 0.19, 0.15
BuynHold_risk, BuynHold_return = 0.15, 0.13

# Inefficient portfolios (random values below the frontier)
```

```
np.random.seed(1)
ineff_risks = np.random.uniform(0.05, 0.3, 10)
ineff_returns = np.random.uniform(0.05, 0.13, 10)

# Plotting
plt.figure(figsize=(10, 6))

# Efficient frontier curve
plt.plot(risk, return_curve, label='Pareto Efficient Frontier', color='blue', linewidth=1)

# Key portfolio points
plt.scatter(PPO_risk, PPO_return, color='blue', label='PPO', zorder=2)
plt.scatter(LR_risk, LR_return, color='orange', label='Logistic Regression', zorder=2)
plt.scatter(SVM_risk, SVM_return, color='yellow', label='Support Vector Machine (SVM)', zorder=2)
plt.scatter(BuynHold_risk, BuynHold_return, color='green', label='Buy and Hold', zorder=2)

# Inefficient portfolios
plt.scatter(ineff_risks, ineff_returns, color='red', marker='o', label='Inefficient Portfolios')

# Annotate points
plt.text(PPO_risk - 0.015, PPO_return + 0.005, 'PPO', fontsize=12)
plt.text(LR_risk + 0.005, LR_return, 'Logistic Regression', fontsize=12)
plt.text(SVM_risk - 0.015, SVM_return + 0.005, 'Support Vector Machine (SVM)', fontsize=12)
plt.text(BuynHold_risk - 0.02, BuynHold_return + 0.005, 'Buy and Hold', fontsize=12)

# Axis labels and title
plt.xlabel('Risk', fontsize=12)
plt.ylabel('Expected Return', fontsize=12)
plt.title('Efficient Frontier and Portfolio Positions', fontsize=14)
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()
```

D Hyperparameter Search Grids

To optimise performance, grid searches were conducted over key hyperparameters. Representative search ranges are summarised below:

Table 12: PPO Hyperparameter Search Grids

Parameter	Values Tested
Learning Rate	`[1e-4, 3e-4, 1e-3]`
Rollout Size (`n_steps`)	`[2048, 4096]`
Trade Fraction	`[0.025, 0.05, 0.075, 0.1, 0.125]`
Max Concentration	`[0.5, 0.6, 0.7, 0.8]`
Concentration Penalty	`[0.005, 0.01, 0.02, 0.03, 0.04]`
VaR Budget	`[0.025, 0.05, 0.075, 0.1]`
VaR Lookback Window	`[30, 60]`

These grids guided systematic tuning of PPO agents (PPO1-PPO4) and informed the final hyperparameter choices reported in the evaluation section.

E Additional Tables

This appendix presents detailed tabular results that complement the main evaluation section. These include Monte Carlo statistics across multiple runs, hyperparameter tuning sweeps, and per-asset performance breakdowns.

E.1 Monte Carlo Rollouts

Table 13 reports mean and standard deviation of performance metrics across 30 Monte Carlo rollouts for PPO agents.

Table 13: Monte Carlo evaluation results (30 runs).

Agent	Cumulative Return (Mean ± SD)	Sharpe Ratio (Mean ± SD)	Max Drawdown (Mean ± SD)
PPO1	18.14 ± 2.10	0.81 ± 0.15	-8.68% ± 1.2%
PPO2	22.11 ± 2.35	1.02 ± 0.18	-8.13% ± 1.0%
PPO3	27.19 ± 2.80	1.13 ± 0.21	-8.20% ± 1.1%
PPO4	31.18 ± 3.05	1.30 ± 0.19	-7.81% ± 1.0%

E.2 Hyperparameter Tuning Sweeps

Trade Fraction (PPO2). Performance sensitivity to `trade_fraction` values.

Table 14: Effect of `trade_fraction` on PPO2 validation performance.

Trade Fraction	Cumulative Return	Sharpe Ratio	Max Drawdown
0.050	20.44%	0.88	-9.91%
0.075	19.44%	0.79	-12.11%
0.100	24.39%	1.02	-9.99%
0.125	23.17%	0.97	-10.44%
0.150	**33.33%**	**1.30**	**-7.99%**

Concentration Penalty (PPO3)

Table 15: Effect of concentration penalty and maximum allocation on PPO3 validation performance.

Max Concentration	Penalty	Cumulative Return	Sharpe Ratio	Max Drawdown
0.5	0.005	30.81%	1.25	-7.99%
0.5	0.010	25.24%	1.10	-9.02%
0.5	0.020	26.33%	1.04	-8.70%
0.5	0.030	**33.14%**	**1.36**	-8.22%
0.5	0.040	24.67%	0.98	-9.51%
0.6	0.005	25.81%	1.09	-7.25%
0.6	0.010	30.61%	1.20	-9.22%
0.6	0.020	22.64%	0.95	-9.34%
0.6	0.030	27.67%	1.12	-8.90%
0.6	0.040	24.15%	0.97	-9.71%
0.7	0.005	31.49%	1.25	-7.98%
0.7	0.010	24.43%	0.95	-9.51%
0.7	0.020	23.15%	0.97	-9.16%
0.7	0.030	30.68%	1.20	-8.83%
0.7	0.040	29.84%	1.25	**-7.75%**
0.8	0.005	22.38%	0.91	-11.38%
0.8	0.010	30.83%	1.29	-8.03%
0.8	0.020	28.19%	1.16	-8.91%
0.8	0.030	21.28%	0.90	-11.17%
0.8	0.040	30.72%	1.22	-8.03%

Word count: 8180