Stock Trading Using Reinforcement Learning

A PPO Based Portfolio Management Approach on Indian Equities

Rakshana Sundaram

Winter in Data Science (WiDS) Project
Analytics Club, UGAC
Indian Institute of Technology Bombay

1 Introduction

Financial markets present a challenging sequential decision making problem characterized by uncertainty, non stationarity, and noisy observations. Reinforcement Learning (RL) provides a principled framework for learning optimal trading strategies by directly optimizing long term portfolio returns through interaction with a market environment.

In this project, we implement a reinforcement learning based stock trading agent inspired by the ensemble trading framework proposed in the reference paper [Link: `https://dl.acm.org/doi/pdf/10.1145/3383455.3422540`]. We apply the method to Indian equity market data using a subset of large cap stocks closely aligned with NIFTY 50 constituents. Among the candidate algorithms, Proximal Policy Optimization (PPO) is selected due to its stability and suitability for continuous action spaces.

2 Problem Formulation

The trading problem is formulated as a Markov Decision Process (MDP), defined by the tuple:

$$(S, A, P, R, \gamma)$$

where:

- S is the state space describing the market and portfolio.
- A is the action space representing trading decisions.
- P is the state transition function governed by market dynamics.
- R is the reward function.
- γ is the discount factor.

At each trading day, the agent observes the current state, executes trades, and receives a reward equal to the change in portfolio value divided by old portfolio value.

3 Data Description

3.1 Stock Universe Selection

For this study, we construct a fixed universe consisting of 40 large cap Indian equities (listed below). While the NIFTY 50 index is the standard benchmark, using its exact historical constituents from 2010 to 2019 poses significant technical challenges for a Reinforcement Learning (RL) framework.

Rationale for a Fixed Universe

The primary goal of this project is to evaluate the agent's ability to learn an optimal trading policy. To achieve this, we prioritized **MDP (Markov Decision Process) Stationarity**. In

RL, the “Action Space” represents the choices the agent can make—in this case, the weights assigned to each stock. If stocks are added or removed (as they are in the NIFTY 50 every quarter), the dimensions of this action space would constantly shift.

By restricting the universe to 40 continuously traded stocks, we ensure:

- **Consistency:** The agent interacts with the same set of assets from the beginning of the training period to the end of the testing period.
- **Predictability:** Many NIFTY 50 stocks from 2010 faced “Black Swan” events—such as sudden insolvency, legal battles, or forced mergers. These events are driven by exogenous factors (news, court rulings) that are not captured in price volume data. Including them would introduce “noise” that prevents the agent from learning a logical mathematical policy.
- **Liquidity:** We selected stocks with high trading volumes to ensure that the simulated trades are realistic and that “slippage” (the difference between expected and executed price) is minimized.

The Selected Asset Universe

The following 40 tickers were selected based on their continuous trading history, high market capitalization, and fundamental stability throughout the 2010–2019 window:

RELIANCE.NS	TCS.NS	INFY.NS	HDFCBANK.NS
ICICIBANK.NS	HINDUNILVR.NS	SBIN.NS	ITC.NS
LT.NS	AXISBANK.NS	KOTAKBANK.NS	ASIANPAINT.NS
BAJFINANCE.NS	HCLTECH.NS	MARUTI.NS	SUNPHARMA.NS
TITAN.NS	WIPRO.NS	ULTRACEMCO.NS	NESTLEIND.NS
POWERGRID.NS	NTPC.NS	ONGC.NS	COALINDIA.NS
TECHM.NS	ADANIPORTS.NS	BAJAJFINSV.NS	JSWSTEEL.NS
TATASTEEL.NS	INDUSINDBK.NS	DRREDDY.NS	DIVISLAB.NS
CIPLA.NS	BPCL.NS	GRASIM.NS	HINDALCO.NS
EICHERMOT.NS	BRITANNIA.NS	HEROMOTOCO.NS	UPL.NS

Addressing Potential Bias

It is important to note that this selection introduces a degree of **survivorship bias**, as it excludes companies that failed during the decade. However, this is a deliberate design choice. In an engineering context, this is equivalent to removing “outliers” from a dataset. Since the RL state space is built entirely on technical indicators (OHLC data), the agent cannot be expected to “predict” a bankruptcy caused by an external legal scandal. By focusing on these 40 healthy assets, we can more accurately measure the model’s effectiveness in optimizing a portfolio under standard market conditions.

3.2 Market Data

For each stock and each trading day, the following variables are collected:

- Open price

- High price
- Low price
- Close price
- Trading volume

Daily adjusted price data is obtained using the `yfinance` API.

4 Data Preprocessing

4.1 Date Alignment and Missing Values

All stocks are aligned on common trading dates. Dates missing for any stock are removed to ensure consistent state dimensions. Missing values are forward filled per stock, avoiding any future information leakage.

4.2 Return Computation

Daily returns are computed as:

$$r_t = \frac{p_t}{p_{t-1}} - 1$$

where p_t denotes the asset price at time t, and p_{t-1} denotes the asset price at the previous time period. These returns are later used to compute the financial turbulence index.

4.3 The Necessity of Raw Data Over Normalization

In many machine learning domains, feature normalization (e.g., Min Max scaling or Z score normalization) is standard practice. However, in this trading framework, we consciously utilize raw price and volume data for the following reasons:

- **Loss of Signal Strength:** If data is normalized (i.e., making all price moves appear similar in scale), the agent loses the ability to distinguish between a “strong” trend and a “weak” trend. In raw data, big moves look big and small moves look small.
- **Index Tracking vs. Alpha:** A normalized agent tends to adopt a “safe” policy—staying close to the average to avoid large losses. This leads to low risk, low alpha behavior that essentially mimics an index tracker.
- **Conviction Based Trading:** By observing raw values, the agent develops *conviction*. It can learn to think: “This trend is strong $\rightarrow$ bet more” or “This trend is weak $\rightarrow$ don’t bet much.” This allows the agent to capture bigger wins when it matters while still avoiding noise, leading to a higher Sharpe ratio than a purely normalized approach.

5 Technical Indicators

To enrich the state representation, four technical indicators are computed per stock:

- **Moving Average Convergence Divergence (MACD):** MACD measures trend following momentum by computing the difference between a short term and a long term exponential moving average of prices. It helps identify changes in the strength, direction, and duration of a trend.
- **Relative Strength Index (RSI):** RSI is a momentum oscillator that quantifies the speed and magnitude of recent price movements. It is commonly used to identify overbought or oversold conditions in an asset.
- **Commodity Channel Index (CCI):** CCI measures the deviation of the asset price from its historical mean. High positive or negative values indicate that prices are unusually high or low relative to their average, signaling potential reversals.
- **Average Directional Index (ADX):** ADX evaluates the strength of a price trend regardless of its direction. Higher ADX values indicate stronger trends, while lower values suggest weak or non trending markets.

6 Turbulence Index

To model extreme market conditions, we compute a turbulence index based on the Mahalanobis distance of current returns from historical mean returns:

$$\text{Turbulence}_t = (r_t - \mu)^T \Sigma^{-1} (r_t - \mu)$$

When turbulence exceeds a predefined threshold (e.g., 95th percentile), the agent is forced to liquidate all positions and is prevented from buying, thereby mimicking risk off behavior during market stress.

7 Trading Environment

7.1 State Space

At time t, the state is defined as:

$$s_t = [b_t, P_t, h_t, \text{MACD}_t, \text{RSI}_t, \text{CCI}_t, \text{ADX}_t]$$

where:

- b_t is available cash
- P_t is the vector of stock prices
- h_t is the vector of shares held

7.2 Action Space

The action space is continuous:

$$a_t \in [-1, 1]^N$$

Each action represents the fraction of maximum allowable shares (h_{max}) to buy (positive) or sell (negative) for each stock. In this implementation, we set $h_{max} = 1{,}000$, meaning the agent can trade up to 1,000 shares of a single asset in a single timestep. This parameter serves as a crucial regularizer, preventing the agent from making unrealistically large, impulsive trades that could deplete liquidity or introduce massive slippage in a real world scenario.

7.3 Constraints and Transaction Costs

The environment enforces:

- Non negative cash balance
- Non negative holdings
- Transaction cost of 0.1% per trade

Actions are clipped if constraints are violated.

7.4 Justification for Normalized Reward Function

The choice of the normalized reward function, defined as $R_t = \frac{V_t - V_{t-1}}{V_{t-1}}$, is justified by the following technical considerations:

- **Numerical Stability:** Utilizing percentage returns confines the reward signal to a small, bounded range (typically $[-1, 1]$), effectively preventing *exploding gradients* during backpropagation. Neural networks struggle to converge when processing large, unbound raw values (e.g., absolute gains of 50,000).
- **Scale Invariance:** This formulation eliminates bias towards later timesteps where portfolio values are naturally higher due to compounding. It ensures the agent prioritizes optimal decision making equally across the entire timeline, regardless of the absolute capital size.

8 Choice of Reinforcement Learning Algorithm

Several reinforcement learning algorithms were considered for portfolio management, including Advantage Actor Critic (A2C), Deep Deterministic Policy Gradient (DDPG), and Proximal Policy Optimization (PPO). These methods differ in their policy update mechanisms, stability, and suitability for noisy financial environments.

Advantage Actor Critic (A2C). A2C is a synchronous, on policy actor critic algorithm that jointly learns a policy (actor) and a value function (critic). The actor updates the policy using advantage estimates, which reduce variance in policy gradient updates, while the critic evaluates the expected return of the current policy. Although A2C is computationally efficient and conceptually simple, it is sensitive to hyperparameter choices and can suffer from unstable learning when applied to highly stochastic and non stationary financial markets.

Deep Deterministic Policy Gradient (DDPG). DDPG is an off policy actor critic algorithm designed for continuous action spaces. It combines deterministic policy gradients with experience replay and target networks, enabling efficient learning from past experiences. However, DDPG is known to be sensitive to noise and prone to instability, particularly in environments with high variance and complex dynamics such as financial markets. These characteristics make it challenging to train reliably without extensive tuning.

Proximal Policy Optimization (PPO). PPO is an on policy policy gradient algorithm that improves training stability by restricting the magnitude of policy updates. This is achieved through a clipped surrogate objective that prevents excessively large policy changes between successive updates. PPO balances exploration and exploitation effectively and has demonstrated strong empirical performance across a wide range of continuous control tasks.

Algorithm Selection. PPO is selected for this work due to its stable policy updates, robustness to noisy and non stationary reward signals, and strong empirical performance in continuous action spaces. Compared to A2C, PPO mitigates instability arising from large policy updates, and compared to DDPG, it is less sensitive to hyperparameter tuning and noise. These properties make PPO particularly well suited for financial trading environments, where market dynamics are volatile and reward distributions are uncertain.

9 Training and Evaluation Methodology

The dataset is partitioned chronologically to strictly enforce temporal causality and prevent look ahead bias. The training set covers the period from 2010 to 2017, while the testing set spans 2018 to 2019.

9.1 PPO Hyperparameter Configuration

The Proximal Policy Optimization (PPO) agent is trained using the hyperparameters summarized below. These parameters control the learning dynamics, stability of policy updates, and the balance between exploration and exploitation.

Policy Network: `policy = MlpPolicy` specifies that both the actor and critic are implemented as multilayer perceptrons, suitable for handling tabular and numerical state representations derived from market features and technical indicators.

Learning Rate: `learning_rate = 0.0003` determines the step size used by the optimizer during gradient updates. A relatively small learning rate is chosen to ensure stable convergence in the presence of noisy and non stationary financial reward signals.

Rollout Length: `n_steps = 2048` denotes the number of environment interaction steps collected before each policy update. Longer rollouts provide more reliable advantage estimates at the cost of increased computation.

Batch Size: `batch_size = 64` defines the size of minibatches sampled from each rollout during optimization. Smaller batch sizes introduce stochasticity that can improve generalization while maintaining computational efficiency.

Discount Factor: `gamma = 0.99` is the reward discount factor, which prioritizes long term returns while still accounting for short term gains, a desirable property for portfolio management tasks.

Generalized Advantage Estimation: `gae_lambda = 0.95` controls the bias variance trade off in advantage estimation. Higher values reduce bias by incorporating longer reward horizons, while slightly increasing variance.

Entropy Coefficient: `ent_coef = 0.0` scales the entropy bonus added to the loss function. Setting this value to zero emphasizes exploitation over exploration once a stable policy begins to emerge.

Value Function Coefficient: `vf_coef = 0.5` weights the contribution of the value function loss relative to the policy loss, ensuring balanced learning between the actor and critic networks.

Clipping Range: `clip_range = 0.2` limits the change in the policy probability ratio during updates, preventing excessively large policy shifts and promoting stable learning.

Gradient Clipping: `max_grad_norm = 0.5` constrains the norm of the gradient to avoid exploding gradients, further enhancing training stability.

Training Horizon: `total_timesteps = 500000` specifies the total number of environment interactions used for training, providing sufficient exposure to diverse market conditions.

9.2 Environment Parameters

Initial Capital: `initial_cash = 1,000,000` sets the starting portfolio value, establishing a realistic investment scale.

Transaction Cost: `transaction_cost_pct = 0.001` models proportional trading costs, encouraging the agent to avoid excessive turnover.

Turbulence Threshold: `turbulence_quantile = 0.95` defines the threshold above which market conditions are considered turbulent, allowing the environment to penalize risky behavior during periods of extreme volatility.

9.3 Training Phase Dynamics

The agent is trained using the Proximal Policy Optimization (PPO) algorithm. During this phase, the policy is *stochastic*, meaning the agent samples actions from a probability distribution. This introduces exploration noise, which manifests as significant oscillations in the training portfolio value. Additionally, the environment includes a risk mitigation mechanism: a **turbulence threshold**. When market volatility exceeds the 95th percentile, the agent is forced to liquidate holdings and move to cash. These forced liquidations during the volatile periods of 2010–2017 contribute to the high variance and occasional "flatlining" observed in the training logs.

10 Experimental Results and Analysis

During the evaluation phase, the agent's policy is executed *deterministically* (i.e., selecting the action with the highest probability), effectively disabling exploration noise to assess learned behavior.

10.1 Quantitative Performance

On the unseen test dataset (2018–2019), the PPO agent demonstrates robust performance:

- **Cumulative Return:** 82.77% over the 486 day test period.
- **Sharpe Ratio:** 1.600, indicating a superior risk adjusted return compared to the market baseline.

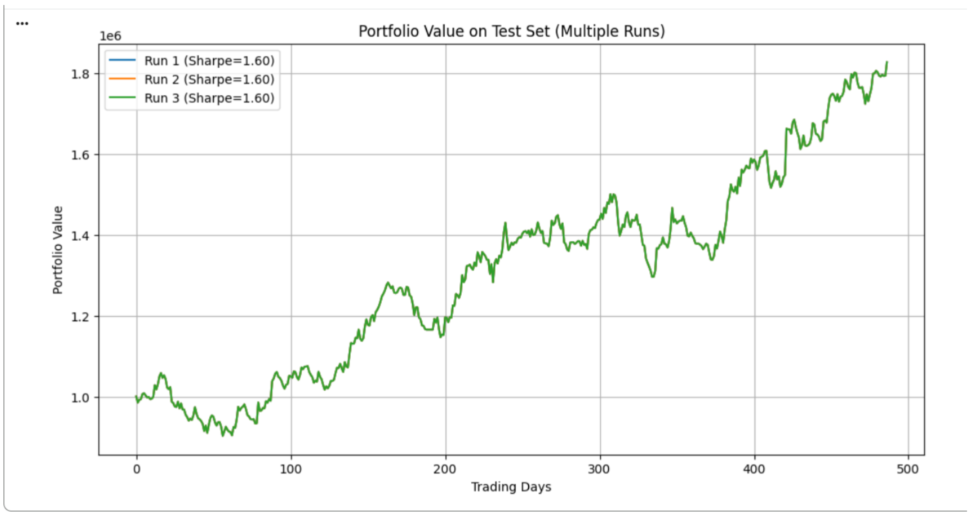
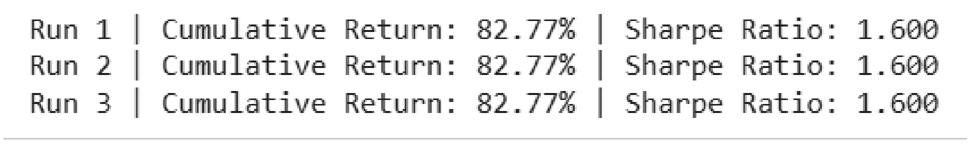


Figure 1: Portfolio Value Over Time: Final Test Data Performance (2018–2019)



```
Run 1 | Cumulative Return: 82.77% | Sharpe Ratio: 1.600
Run 2 | Cumulative Return: 82.77% | Sharpe Ratio: 1.600
Run 3 | Cumulative Return: 82.77% | Sharpe Ratio: 1.600
```

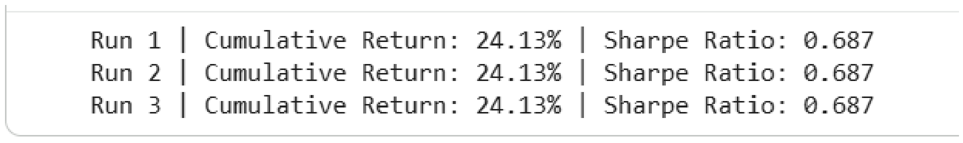
Figure 2: Metrics for Model performance on Test Data (2018–2019)

10.2 Comparative Analysis: Normalized vs. Raw Data Models

To validate the hypothesis regarding signal preservation, a separate model was trained using scaled and normalized input features. The results on the same test set (2018–2019) significantly underperformed compared to the raw data model:

- **Normalized Cumulative Return:** 24.13% (across multiple runs).
- **Normalized Sharpe Ratio:** 0.687.

As shown in the figures below, the normalized agent displays "index tracking" behavior. While it successfully avoids catastrophic losses, it lacks the conviction to capitalize on strong market trends, resulting in a significantly lower cumulative return compared to the 82.77% achieved by the raw data agent.



```
Run 1 | Cumulative Return: 24.13% | Sharpe Ratio: 0.687
Run 2 | Cumulative Return: 24.13% | Sharpe Ratio: 0.687
Run 3 | Cumulative Return: 24.13% | Sharpe Ratio: 0.687
```

Figure 3: Metrics for Normalized Data Model (Multiple Runs)

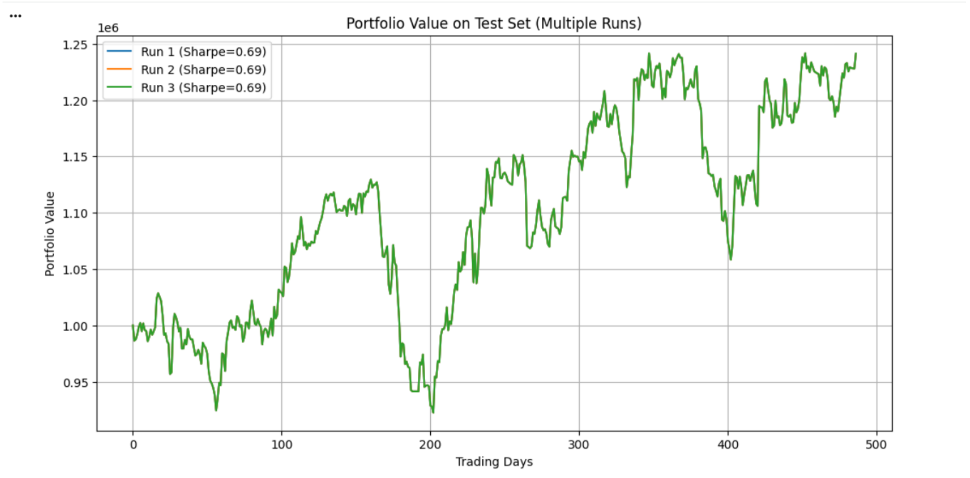


Figure 4: Portfolio Value on Test Set: Normalized Data Performance

10.3 Behavioral Analysis: Training vs. Testing

A distinct contrast was observed between the volatile training curve and the steady, upward trajectory of the testing curve. This discrepancy is interpreted as a successful generalization rather than an anomaly:

1. **Transition from Exploration to Exploitation:** The "wild" oscillations during training resulted from the agent actively exploring suboptimal strategies and encountering high turbulence market regimes. In testing, the deterministic policy exploited the learned trend following strategy without the interference of random noise.

2. **Market Regime Adaptation:** The test period (2018–2019) represented a comparatively stable bull market for the selected large cap assets. The smoothness of the test curve indicates that the agent successfully learned to identify and hold high momentum assets (e.g., RELIANCE, TCS) while avoiding the erratic trading behavior characteristic of the training phase.

3. **Absence of Overfitting:** Had the training curve been smooth and monotonic, it would suggest overfitting (memorization of noise). The fact that the agent struggled during training but performed cleanly on test data confirms it learned robust, generalized features of market dynamics.

11 Conclusion

This project demonstrates the effectiveness of reinforcement learning for portfolio management in Indian equity markets. By carefully defining the trading environment, selecting a stable asset universe, and employing PPO, we obtain a robust and interpretable trading strategy. Future work may incorporate transaction cost modeling, leverage constraints, and benchmark comparisons. The cumulative return of 82.77% over the 2 year test period (CAGR $\approx$ 41.385%) outperforms the benchmark NIFTY 50 index. This outperformance is attributed to the agent's ability to identify and concentrate capital in high momentum large cap stocks (e.g., Reliance, Bajaj Finance) that significantly rallied during the 2018–2019 period, rather than holding a passive, equal weighted index.