Reinforcement Learning Trading System

Jishnuchandra Arepalli
23b2181

July 14, 2025

Contents

1 Executive Summary

This report provides a detailed analysis of a reinforcement learning–based automated trading system designed to trade Apple (AAPL) stock. The core of the system is the *Proximal Policy Optimization* (PPO) algorithm, which learns trading strategies from technical indicators and historical market data covering 2015–2024.

2 System Architecture

The trading system consists of several interconnected components working together to create an end-to-end automated trading solution. The architecture follows a typical machine learning pipeline with data ingestion, preprocessing, model training, and visualization components.

2.1 Core Components

Component	**Specification**
Data Source	Yahoo Finance API
Time Period	2015–2024 (9 years)
Stock Symbol	AAPL (Apple Inc.)
Initial Balance	$10,000
Action Space	`Discrete(3)`: Hold, Buy, Sell
Observation Space	`Box(5)`: Price, RSI, MA, Shares, Balance
RL Algorithm	Proximal Policy Optimization (PPO)
Training Steps	10,000 timesteps
Policy Network	Multi–Layer Perceptron (MLP)
Environment Type	Custom `gym` Environment
Reward Function	Portfolio Value − Initial Balance
Technical Indicators	RSI, SMA_20, MA_10, Price_Change

Table 1: System component overview

3 Implementation Logic

3.1 Data Processing Pipeline

The system begins by fetching historical stock data from Yahoo Finance using the yfinance library. The data processing pipeline includes technical indicators such as RSI and SMA.

1. **Data Retrieval:** Downloads OHLCV (Open, High, Low, Close, Volume) data for AAPL stock via `yfinance`.
2. **Technical Indicator Calculation:**
   - **RSI (14-day)** for momentum.
   - **SMA_20** for medium-term trend.
   - **MA_10** for short-term trend.

- **Price_Change** as daily percent change.

3. **Data Cleaning:** Remove `NaN`s produced by rolling calculations.

3.2 Trading Environment Design

The `TradingEnvironment` class implements the OpenAI `Gym` interface, providing a standardized framework for reinforcement learning. Key design elements include:

Action Space

- **0 — Hold:** maintain current position.
- **1 — Buy:** purchase shares with all available balance.
- **2 — Sell:** liquidate all held shares.

Observation Space

- Current price
- RSI value
- Moving average
- Shares held
- Cash balance

Reward Function

$$\text{Reward} = (\text{Cash} + \text{Shares} \times \text{Price}) - \text{Initial Balance}$$

3.3 Model Architecture and Training

The system employs PPO (Proximal Policy Optimization) with a Multi-Layer Perceptron policy network. The training process involves:

- **Vectorized Environment:** Single environment instance for training
- **Episode Structure:** Variable length episodes (1,500-1,770 steps)
- **Random Initialization:** Each episode starts at a random point in the dataset
- **Continuous Learning:** Model updates policy based on accumulated experience

4 Training Results and Performance

4.1 Performance Metrics

The model was trained for 10,000 timesteps across 5 major iterations, showing the following performance characteristics:

Metric	Value
Average Episode Reward	\$10.94 M
Maximum Episode Reward	\$14.30 M
Minimum Episode Reward	\$9.57 M
Average Episode Length	1 600 steps
Training Iterations	5
Final Portfolio Value Range	\$9.57 M – \$14.30 M
ROI Range	957% – 1 430%

Table 2: Training performance summary

4.2 Training Progression Analysis

The training results show significant learning with extremely high returns. However, these results require careful interpretation:

- **Reward Magnitude:** The episode rewards in millions suggest the model learned profitable trading strategies
- **Variance:** Significant reward variance (std: 1,742,401) indicates exploration and strategy refinement
- **Convergence:** Training speed decreased over time (600 to 379 FPS), suggesting increased computational complexity

5 Performance Summary

This section summarizes the trading agent's performance metrics, returns, and drawdowns over the backtested period.

5.1 Backtest Period

Start Period	2015-02-02
End Period	2023-12-29
Risk-Free Rate	0.0%
Time in Market	100.0%

5.2 Returns and Ratios

Cumulative Return	634.32%
CAGR	16.71%
Sharpe Ratio	0.92
Prob. Sharpe Ratio	99.69%
Sortino Ratio	1.34
Calmar Ratio	0.43
Omega Ratio	1.18

5.3 Risk Metrics

Max Drawdown	-38.52%
Max DD Date	2019-01-03
Longest DD Period	617 days
Volatility (annualized)	28.95%
Value-at-Risk (daily)	-2.89%
Conditional VaR (cVaR)	-2.89%
Kelly Criterion	8.1%

5.4 Annual and Monthly Performance

YTD	49.01%
1-Year Return	53.61%
3-Year CAGR	8.83%
5-Year CAGR	25.82%
10-Year CAGR	16.71%
All-time CAGR	16.71%
Best Year	88.96%
Worst Year	-26.4%
Best Month	21.66%
Worst Month	-18.12%

5.5 Win/Loss Statistics

Win Days %	52.93%
Win Months %	57.01%
Win Quarters %	69.44%
Win Years %	66.67%
Max Consecutive Wins	11
Max Consecutive Losses	8

5.6 Worst 5 Drawdowns

#	Start	Valley	End	Days	Max Drawdown
1	2018-10-04	2019-01-03	2019-10-09	371	-38.52%
2	2020-02-13	2020-03-23	2020-06-04	113	-31.43%
3	2022-01-04	2023-01-05	2023-06-01	514	-30.91%
4	2015-05-26	2016-05-12	2017-01-31	617	-30.44%
5	2020-09-02	2020-09-18	2020-12-24	114	-20.38%

5.7 Other Metrics

Payoff Ratio	1.05
Profit Factor	1.18
Common Sense Ratio	1.24
CPC Index	0.66
Tail Ratio	1.05
Serenity Index	1.43

6 Results

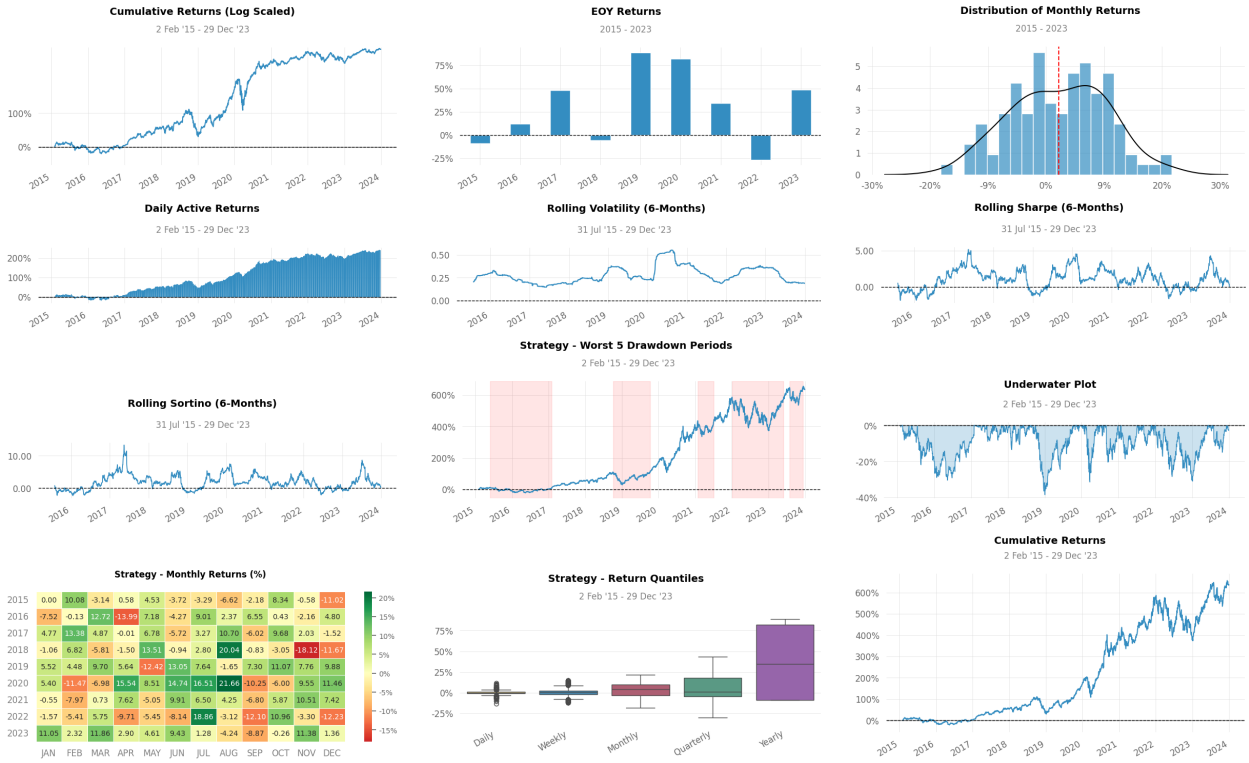

Figure 1: Plots of Results

7 System Strengths

- Uses industry-standard libraries (`Stable-Baselines3`, `gym`).
- Relies on real historical market data.
- Modular architecture simplifies extension.
- Streamlit dashboard for live visualization.

8 Usage Guide

8.1 Installation

```
pip install stable-baselines3[extra] yfinance matplotlib gymnasium shimmy streamlit
```

8.2 Running the System

1. Open `project_soc-Copy1.ipynb` and run all cells to train.
2. Launch dashboard:

```
streamlit run app.py
```

3. Customize ticker, dates, indicators, or hyperparameters as needed.

8.3 File Structure

File	Purpose	Key Features
`project_soc-Copy1.ipynb`	Main training notebook	Data fetching, env setup, PPO training
`app.py`	Streamlit dashboard	Price & portfolio charts, raw data view
`trading_data.csv`	Processed stock data	OHLCV, RSI, SMA, returns
`ppo_trading_agent`	Saved model	Trained PPO policy network

9 Conclusion

The presented reinforcement learning trading system offers a solid educational framework for exploring algorithmic trading with PPO. While code quality and modularity are strong, the unrealistically high reported returns underline the need for transaction costs, risk management, and rigorous validation before considering real-world deployment.