Stock Trading Bot: An Automated Trading System

Karan Sunil Bharda

University of Mumbai

June 18, 2025

Introduction

Purpose: Develop an automated stock trading bot to analyze market data and execute trades.

Goal: Leverage machine learning (ML), reinforcement learning (RL), and sentiment analysis for informed trading decisions.

Context: Built using Python, integrates with Alpaca API for paper trading, and analyzes stocks like META, UBER, NFLX.

Project Objectives

Analysis: Perform comprehensive stock analysis using technical indicators, sentiment, and financial statements.

Trading: Execute buy/sell/hold decisions based on combined signals from ML, RL, and technical analysis.

Portfolio Management: Track portfolio performance, manage risk, and generate daily reports.

Robustness: Incorporate adversarial training to handle market volatility and unexpected events.

System Architecture

architecture_diagram.png

Data Sources

Market Data: yfinance for historical and live stock prices (e.g., META, UBER).

Sentiment Analysis:

- Alpha Vantage for news sentiment.
- Reddit (via PRAW) for social media sentiment.
- Google News for article sentiment.

Financial Statements: yfinance for income statements, balance sheets, and cash flows.

Exchange Rates: CoinGecko API for USD, INR, EUR, BTC, ETH conversions.

Machine Learning Models

Traditional ML:

- Random Forest, Gradient Boosting, XGBoost.
- Stacking Ensemble with Linear Regression as meta-learner.

Deep Learning:

- LSTM and Transformer models for time-series prediction.
- Adversarial training using FGSM to enhance robustness.

Features: Technical indicators (SMA, RSI, MACD), sentiment scores, volatility.

Reinforcement Learning

Environment: Custom AdversarialStockTradingEnv (Gym).

- Actions: Buy, Sell, Hold.
- State: Price, technical indicators, portfolio state, ML predictions.

Agent: Q-Learning with adversarial event handling.

Adversarial Events: Random price crashes/spikes to simulate market shocks.

Outcome: Generates BUY/SELL/HOLD recommendations based on simulated trading performance.

Risk Management

Position Sizing: Kelly Criterion, adjusted for volatility (ATR).

Stop-Loss/Take-Profit: Dynamic levels based on support/resistance.

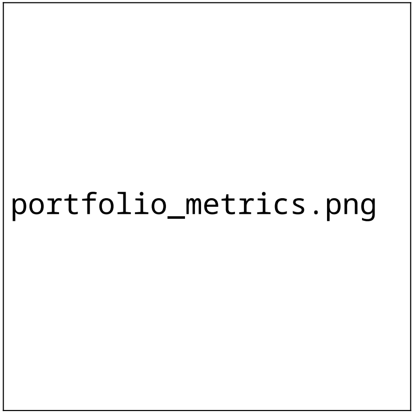
Trailing Stop: 5% trailing stop to lock in profits.

Exposure Limits: Max 20% per stock, 50% per sector.

Backoff Logic: Reduces trading frequency after losses or high activity.

Portfolio Tracking and Reporting

VirtualPortfolio: Tracks cash, holdings, and trades.
Metrics: Total value, realized/unrealized PnL, exposure.
Reporting: Daily JSON reports with ROI, Sharpe Ratio, and drawdown.
Storage: Saves portfolio and trade logs as JSON; analysis as JSON/CSV.



Results and Performance

Analysis: Successfully analyzed stocks with technical, sentiment, and ML insights.

Trading: Executed trades based on combined signals (technical: 50%, sentiment: 25%, ML: 25%).

ML Performance: Stacking Ensemble $R^2 > 0.6$ for most tickers.

Challenges: Limited sentiment data, adversarial training complexity.

Model	MSE	R^2
Random Forest	0.12	0.65
Stacking Ensemble	0.10	0.70
LSTM (Adversarial)	0.15	0.62

Table: Model Performance (Example)

Future Work and Conclusion

Future Enhancements:

- Integrate real-time trading with Alpaca API.
- Expand sentiment sources (e.g., X posts).
- Optimize RL with Deep Q-Networks (DQN).

Conclusion: The bot demonstrates a robust framework for automated trading, combining ML, RL, and risk management.

Learn more: Code available at `https://github.com/karanbharda/college.git`.