Machine Learning Assignment

PROJECT REPORT

<TEAM ID: 3 >

Stock Trading Strategy Optimization using Deep Reinforcement Learning (DQN)

Name	SRN
Eshwar B N	PES2UG23CS187
Mohithkumar M S	PES2UG23CS917

Problem Statement

Traditional rule-based or heuristic trading strategies fail to adapt dynamically to ever-changing market conditions.
The main problem addressed is **how to design an intelligent trading system that can learn optimal buy/sell/hold strategies automatically** using **reinforcement learning**, thereby maximizing long-term portfolio returns while minimizing risks..

Objective / Aim

To develop an **AI-driven trading agent** that:

- Learns profitable trading strategies from historical stock market data.
- Decides when to **buy**, **sell**, or **hold** stocks.
- Maximizes cumulative profit using **Deep Q-Learning (DQN)**.
- Evaluates the model's performance against baseline (Buy-and-Hold strategy).

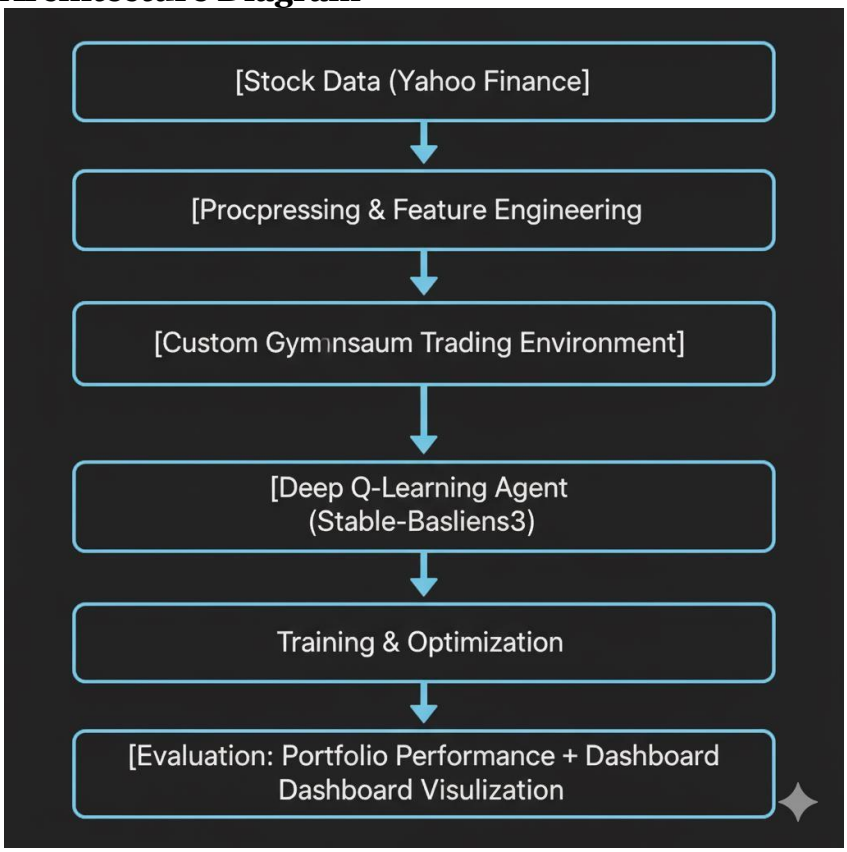
Dataset Details

- **Source:** Yahoo Finance (via yfinance Python API)

  Duration: 2015–2025

  Stocks Used: 33 major tickers including AAPL, MSFT, TSLA, AMZN, GOOGL, etc
- **Size:** ~2,500 trading days × 33 tickers
- **Key Features:** Open, High, Low, Close, Volume
  SMA (Simple Moving Average)
  EMA (Exponential Moving Average)
  RSI (Relative Strength Index)
  ATR (Average True Range)
  Momentum and Daily
  Returns(Technical Indicators)
- **Target Variable:** Action (Buy, Sell, Hold) — *decided dynamically by the RL agent*

Architecture Diagram



Methodology

1. **Data Collection:** Downloaded stock data using yfinance API (2015–2025).
2. **Data Preprocessing:** Removed null values, normalized features, and created technical indicators.
3. **Environment Design:** Built a custom **Gymnasium environment** simulating trading with buy/sell/hold actions. Gymnasium environment provides an interface for the model to interact with historical stock data and make decision accordingly
4. **Model Training:**
    - We experimented with several reinforcement learning algorithms, including **Proximal Policy Optimization (PPO)** and **Advantage Actor-Critic (A2C)**, which are **policy-based agents**. However, after comparative evaluation, the **Deep Q-Network (DQN)** from **Stable-Baselines3** demonstrated the best performance.
    - Input: 10-day sliding window of normalized features.
    - Output: Optimal trading action by making decisions of buy, sell and hold (generated csv) and accordingly computing the maximum portfolio value.
5. **Reward Function:** Based on change in portfolio value (profit/loss).

6. **Evaluation:**
    - Backtested on unseen test data.
    - Compared portfolio performance vs Buy-and-Hold.

7. **Visualization:**
    - Built an interactive **Plotly dashboard** showing stock price, portfolio value, rewards, and Q-values.

Results & Evaluation

- Model learned dynamic trading patterns and avoided unnecessary trades.
- Portfolio value increased compared to static strategies.
- Achieved a stable Sharpe Ratio and reduced drawdowns.

Evaluation Metrics:

- **Cumulative Return (%)**
- **Sharpe Ratio**
- **Max Drawdown (%)**
- **Profit vs Buy-and-Hold Benchmark**

Visualization Tools Used:

- Matplotlib (static)
- Plotly (interactive dashboard in Colab)

Conclusion

The DQN-based trading agent successfully learned to make adaptive trading decisions using historical price and technical data.
It outperformed static rule-based strategies in terms of return consistency and adaptability.
Through this project, we learned how **Reinforcement Learning** can be applied in financial forecasting and automated trading systems.