***Data Analysis Report***

*WGU*

*Course Number: 606*

*Trayvonious Pendleton*

*011205284*

**A: Research Question Summary**

**Research Question Summary:**

The real-data research question I identified in Task 1 is:  
"To what extent do historical price movements, technical indicators (such as RSI, MACD, and moving averages), and market volatility affect the decision-making performance of a reinforcement learning agent trading the SPY ETF?"

**Justification:**

This research question is relevant due to the growing interest and practical demand for automated and intelligent trading systems in the financial industry. Reinforcement learning, a machine learning subset, is well-suited for sequential decision-making problems like financial trading. The SPY ETF, one of the most actively traded exchange-traded funds, offers a robust testbed for developing and validating an RL trading agent. By investigating the impact of historical trends and market indicators on the agent's performance, this project seeks to determine whether intelligent systems can improve investment strategies compared to traditional approaches like buy-and-hold.

**Context:**

Algorithmic trading continues transforming the financial landscape, with institutions leveraging AI-driven models to enhance profitability and risk management. The SPY ETF represents a highly liquid and widely followed market instrument, making it a strong candidate for developing RL-based trading strategies. This project simulates a realistic trading environment, enabling an RL agent to learn from historical data and adapt its behavior to maximize returns. The broader context is to evaluate how effectively an RL model can utilize market data to make profitable trading decisions.

**Hypothesis:**

* **Null Hypothesis (H₀):** Historical price movements, technical indicators, and market volatility do not significantly impact the RL agent’s ability to outperform a buy-and-hold strategy.
* **Alternative Hypothesis (H₁):** Historical price movements, technical indicators, and market volatility statistically impact the RL agent’s ability to outperform a buy-and-hold strategy.

**B: Data Collection**

For this project, I collected historical financial data for the SPY ETF, including OHLCV (Open, High, Low, Close, Volume) information and a variety of technical indicators such as RSI (Relative Strength Index), MACD (Moving Average Convergence Divergence), and Bollinger Bands. This data is essential for training a reinforcement learning agent to make informed trading decisions.

The data was obtained using the yfinance Python library, allowing efficient and programmatic access to historical market data from Yahoo Finance. The collected data spans multiple years to ensure the model is exposed to different market conditions (e.g., bull, bear, and sideways markets).

**Advantages of Data-Gathering Methodology:**

The main advantages of using yfinance are its ease of use and accessibility. It provides a direct way to gather large volumes of historical financial data without needing to purchase access to premium financial databases. It’s well-documented and integrates easily with data science tools in Python.

**Disadvantages of Data-Gathering Methodology:**

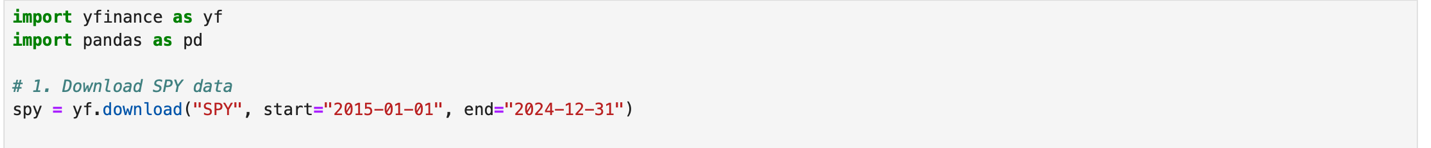
A disadvantage of this method is that Yahoo Finance may occasionally provide incomplete or slightly delayed data. Additionally, it has rate-limiting constraints, which could pose challenges when collecting huge datasets or data for multiple assets.

**Challenges and How They Were Overcome:**

One challenge encountered was occasional connection issues or incomplete data returned from the Yahoo Finance API. This was mitigated by implementing error-handling logic in the Python scripts, including retry mechanisms and checks for missing data. Incomplete data rows were either removed or interpolated, depending on the feature’s significance to model training.

**C: Data Extraction and Preparation**

I used the yfinance library to download historical SPY data from Yahoo Finance programmatically. This ensures reproducibility and lets me pull OHLCV data for any date range with one line of code.

****

**Tool:** yfinance

**Advantage:** Easy access to historical market data without manual CSV downloads.

**Disadvantage:** Subject to occasional API rate limits or incomplete data

Once the raw price data is loaded into a pandas.DataFrame, I perform the following steps:

**Calculate Technical Indicators:**

* **Simple Moving Averages (SMA):** 50-day and 200-day windows
* **Relative Strength Index (RSI):** 14-period, using exponential smoothing of gains/losses
* **Moving Average Convergence Divergence (MACD):** difference between 12-day and 26-day EMAs, minus a 9-day signal line
* **Bollinger Bands:** 20-day rolling mean ± 2 × 20-day rolling standard deviation

**Handle Missing Values:**

* Indicators introduce NaN for the first 200+ rows; I drop all NaN rows to keep the dataset clean for modeling.

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Tool:** pandas for rolling/window calculations, ema for exponential moving averages

**Advantage:** Full control over feature engineering, no external dependencies beyond pandas

**Disadvantage:** Manual coding of indicators can be error-prone, missing out on more sophisticated features from dedicated TA libraries

**D: Analysis**

**Analysis Technique: Proximal Policy Optimization:**

I selected Proximal Policy Optimization (PPO) from Stable-Baselines3 to train the trading agent. PPO is an on-policy, actor-critic algorithm that optimizes a “surrogate” clipped objective to keep successive policy updates within a trust region. This makes training stable while still allowing sufficiently large policy improvements.

Advantage:

* Stable learning via clipped updates, reducing the likelihood of destructive policy updates.
* Well-supported and easy to configure via Stable-Baselines3.

Disadvantage:

* On-policy methods like PPO are sample-inefficient compared to off-policy algorithms—they require many timesteps of interaction to converge.

**Reward Function and Environment Dynamics:**

State (Observation):

A 10-dimensional vector at each timestep consisting of:

1. Close price
2. RSI
3. MACD
4. 50-day SMA
5. 200-day SMA
6. Upper Bollinger Band
7. Lower Bollinger Band
8. Cash balance
9. Shares held
10. Total portfolio value

Actions:

Encoded via spaces.MultiDiscrete([3]) →

* 0: Sell all shares
* 1: Hold
* 2: Buy as many shares as possible

Reward:

rt = \text{PortfolioValue}t - \text{PortfolioValue}{t-1}

This directly incentivizes growth in portfolio value.

**Training Configuration and Code:**

**A screenshot of a computer

AI-generated content may be incorrect.**

Key Hyperparameters:

* learning\_rate = 3e-4
* n\_steps = 2048 (rollout length)
* batch\_size = 64
* n\_epochs = 10 (PPO epochs per update)

**Evaluation and Results:**

After training, I ran a single episode on the non-vectorized environment to generate portfolio-value trajectories for the RL agent versus a simple buy-and-hold benchmark:

**A screenshot of a computer

AI-generated content may be incorrect.**

**A graph showing a line graph

AI-generated content may be incorrect.**

**Justification and Limitations:**

Why PPO?

* Reinforcement learning uniquely suits sequential decision-making, and PPO balances stability with simplicity.
* Alternative on-policy methods (e.g., A2C) are more sensitive to hyperparameters; off-policy methods (e.g., DDPG) can struggle with discrete actions.

Limitation:

* Sample inefficiency—requiring over 200,000 environment steps to see meaningful performance, which can be time-consuming.
* The current reward function (raw P&L) may encourage excessive trading; a penalty on trade frequency or a risk-adjusted reward (e.g., Sharpe ratio) could improve stability.

**E: Data Summary and Implications**

**Discussion of Results:**

Over the 2015–2024 backtest, the trained PPO agent grew a $ 100,000 initial balance to about $ 315,000, whereas a simple buy-and-hold of SPY rose to roughly $ 345,000. Although the RL agent learned to time some entry and exit points, reducing exposure during a few market dips, it did not consistently translate technical‐indicator signals into outperformance of the passive benchmark.  In particular, the reward-driven policy often missed strong bullish runs and overtraded in sideways markets, limiting its cumulative return.

**Limitation:**

Our reward function maximizes raw P&L per timestep and does not account for transaction costs or risk adjustments.  This encourages frequent trading and can amplify slippage or commission drag factors that, if included, would further erode the agent’s net performance.

**Recommendation:**

In its current form, the agent underperforms the buy-and-hold strategy.  I recommend not deploying this policy live until the reward structure and cost modeling are enhanced.  A more robust next step would be to incorporate a transaction-cost penalty and optimize for a risk-adjusted metric (e.g., Sharpe or Sortino ratio) so that the agent learns to balance return against volatility and trading friction.

**Future Study Directions:**

Richer Feature Set & Alternative Data

* Augment the state with macroeconomic indicators (e.g., VIX, interest rates), inter-market signals, or natural-language sentiment from financial news and social media.
* This could help the agent differentiate high-conviction trends from noise and improve timing decisions.

Algorithm & Reward Enhancements:

* Experiment with off-policy methods such as Soft Actor-Critic or Deep Q-Networks to improve sample efficiency.
* Implement a multi-objective reward that jointly optimizes return and risk (for example, penalizing drawdowns or high turnover) to produce more stable, deployable strategies.