MCST1043 RESEARCH DESIGN AND ANALYSIS IN DATA SCIENCE

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

REINFORCEMENT LEARNING FOR AUTOMATED TRADING IN STOCK MARKET

CANDIDATE : LEE HONG JIAN
LECTURER : ASSOC. PROF. DR MOHD SHAHIZAN BIN OTHMAN
VENUE :MP1,LEVEL 1,BLOCK N28A
DATE : 29/Jun 2025

UTM UNIVERSITI TEKNOLOGI MALAYSIA

Presentation Video :

https://www.youtube.com/watch?v=EbuiOBh_BTo

CHAPTER 1: RESEARCH INTRODUCTION

Research Introduction

- Definition of Machine Learning (ML) and Reinforcement Learning (RL):
  - ML focuses on creating algorithms to let agents learn from data without explicit programming (Vec et al., 2024).
  - DRL involves agents making decisions based on real-time interactions with the environment using a reward system (Barto et al., 2025).

- Deep Reinforcement Learning (DRL) in Trading:
  - DRL adapts to market changes, develops strategies based on trends, and enhances decision-making (Huang et al., 2024).

- Benefits of DRL in Trading:
  - Adaptability: Dynamic algorithms adjust based on market conditions.
  - Improved Decision-Making: Enhanced through market interaction and data-driven strategies.
  - Automation and Optimization: DRL reduces resource usage and increases efficiency (Kabbani & Duman, 2022).
  - Emotional Neutrality: DRL removes emotional biases in decision-making, unlike human traders who may be affected by emotions.

- Potential in Decision-Making and Forecasting:
  - DRL's ability to make effective decisions and forecasts makes it highly effective in the dynamic stock market (Sangve et al., 2025).

UNIVERSITI TEKNOLOGI MALAYSIA

Problem Background

- Emotional Biases in decision-making (Aziz et al. 2024).
- Lack of the automation in decision-making process (Kabbani and Duman, 2022).
- Multiple of the policy for optimize returns (Huang et al.2024)

The Goals of Study

- Improve efficiency of DRL model training in stock market
- Improvement in Trading Strategy Optimization
- Minimizing the emotional biases in decision-making (advancement of market prediction)

Research Mapping

Research Gap	Problem Statement	Research Question	Research Objective
Emotional biases affecting trading strategies during decision-making (Aziz et al. 2024).	Decision-making is influenced by the sentimental, will leading to suboptimal outcomes	How can emotional biases in decision-making be minimized?	To train an agent resistant to sentiment-driven biases using the DRL model.
Limitations of traditional trading strategies (Kabbani and Duman, 2022).	Lack of automation in decision-making processes.	What is the optimal model for automated decision-making?	To compare performance and analyze strengths/weaknesses of DQN, PPO, and SAC models
Policies that optimize return rates (Huang et al.2024)	Each model has different mechanisms/policy settings for training the agent	What is the best policy to optimizing returns?	To develop a dashboard visualizing returns from agents trained under different mechanisms.

CHAPTER 2:
LITERATURE REVIEW

Standard & Poor 500 (S&P 500)

- S&P 500 Introduction: Launched in 1957 to track 500 major U.S. companies on the New York Stock Exchange (NYSE).
- Index Updates: Regularly adjusted to reflect economic changes; over 900 companies added/removed.
- Outperformance: The S&P 500 consistently outperforms most active managers due to the inclusion of high-performing companies.
- Economic Representation: Represents diverse sectors of the U.S. economy, ensuring ongoing relevance

Date	Open	High	Low	Close	Adj Close	Volume
May 9, 2025	5,679.65	5,691.69	5,644.15	5,659.91	5,659.91	4,645,090,000
May 8, 2025	5,663.60	5,720.10	5,635.38	5,663.94	5,663.94	5,627,400,000
May 7, 2025	5,614.18	5,654.73	5,578.64	5,631.28	5,631.28	4,987,440,000
May 6, 2025	5,605.87	5,649.58	5,586.04	5,606.91	5,606.91	4,717,260,000
May 5, 2025	5,655.32	5,683.38	5,634.48	5,650.38	5,650.38	4,358,260,000
May 2, 2025	5,645.88	5,700.70	5,642.28	5,686.67	5,686.67	4,854,380,000
May 1, 2025	5,625.14	5,658.91	5,597.35	5,604.14	5,604.14	4,935,270,000
Apr 30, 2025	5,499.44	5,581.84	5,433.24	5,569.06	5,569.06	5,449,490,000
Apr 29, 2025	5,508.87	5,571.95	5,505.70	5,560.83	5,560.83	4,747,150,000
Apr 28, 2025	5,529.22	5,553.66	5,468.64	5,528.75	5,528.75	4,257,880,000
Apr 25, 2025	5,489.73	5,528.11	5,455.86	5,525.21	5,525.21	4,236,580,000
Apr 24, 2025	5,381.38	5,489.40	5,371.96	5,484.77	5,484.77	4,697,710,000
Apr 23, 2025	5,395.92	5,469.69	5,356.17	5,375.86	5,375.86	5,371,390,000
Apr 22, 2025	5,207.67	5,309.61	5,207.67	5,287.76	5,287.76	4,666,950,000
Apr 21, 2025	5,232.94	5,232.94	5,101.63	5,158.20	5,158.20	4,226,340,000
Apr 17, 2025	5,305.45	5,328.31	5,255.58	5,282.70	5,282.70	4,714,880,000
Apr 16, 2025	5,335.75	5,367.24	5,220.79	5,275.70	5,275.70	4,607,750,000
Apr 15, 2025	5,411.99	5,450.41	5,386.44	5,396.63	5,396.63	4,317,110,000
Apr 14, 2025	5,441.96	5,459.46	5,358.02	5,405.97	5,405.97	5,031,440,000
Apr 11, 2025	5,255.56	5,381.46	5,220.77	5,363.36	5,363.36	5,602,550,000
Apr 10, 2025	5,353.15	5,353.15	5,115.27	5,268.05	5,268.05	6,677,140,000
Apr 9, 2025	4,965.28	5,481.34	4,948.43	5,456.90	5,456.90	9,489,600,000
Apr 8, 2025	5,193.57	5,267.47	4,910.42	4,982.77	4,982.77	7,408,140,000
Apr 7, 2025	4,953.79	5,246.57	4,835.04	5,062.25	5,062.25	8,691,980,000

Existing Model Framework

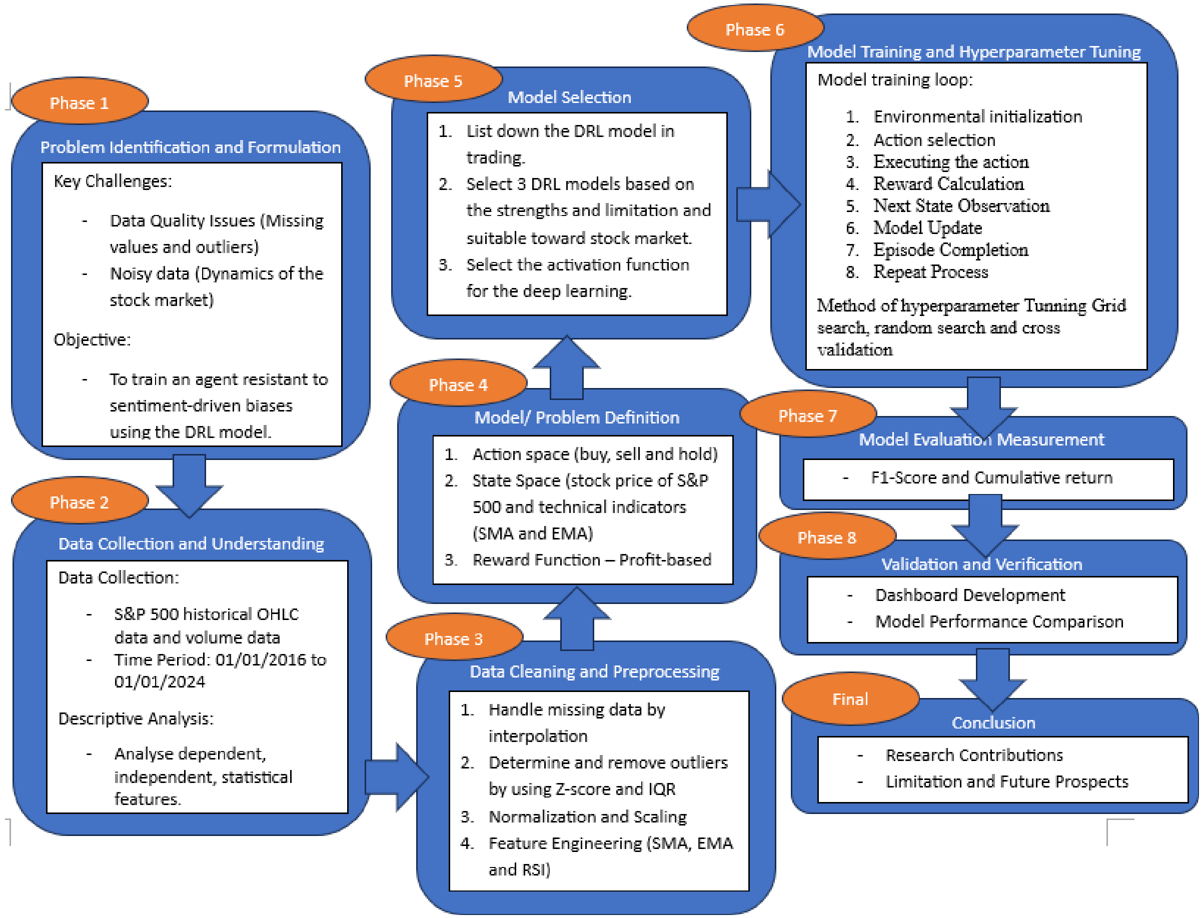
- Traditional Analysis in trading (Huang et al., 2024):
  - Fundamental Analysis
  - Technical Analysis
  - Statistical Models in Trading
- Best fit for stable stock market but not applicable for dynamics stock market (Huang et al., 2024).

DRL Models in Trading

No	Model Name	Function of model in trading	Strength of model	Limitation of model	Results of paper	Citation
1	**Deep-Q-Network (DQN)**	By performing the deep neutral networks to calculate the Q-values, discrete trading actions (Sell, Hold, Buy) based on the mapping status (observations of market)	- Able to handles the high dimensional state spaces. - Without the handcrafted features, able to learn directly from the raw data (price of stock data)	- Trends to overestimate Q-values that may affect the grade of policy quality. - Only suitable for the discrete action spaces.	DQN-based trading outperformed rule-based strategies with higher cumulative returns and Sharpe ratios on historical stock data (Otabek et al., 2024).	Otabek, S., & Choi, J. (2024). Multi-level deep Q-networks for bitcoin trading strategies. Scientific Reports (Nature Publisher Group), 14(1), 771. doi: https://doi.org/10.1038/s41598-024-51408-w
2	**Proximal Policy Optimization (PPO)**	By the surrogate clipped objectives to balance the exploration and exploitation. It will help in stable policy update for discrete and continuous spaces.	- Efficient of sample and robust to hyperparameter choices. - Well handle of stochastics policy and helps in improvement of exploration.	- intensive computation - May achieve the local optima without the adequate exploration.	PPO agents adapt well toward the dynamics market and outperformed than DQN and A2C models in portfolio management (Sun., 2023)	Sun, Q. (2023). Reinforcement learning algorithms for stock trading (Order No. 31765482). Available from ProQuest Dissertations & Theses Global. (3186188497). Retrieved from https://vpn.utm.my/dissertations-theses/reinforcement-learning-algorithms-stock-trading/docview/3186188497/se-2
3	**Advantage Actor-Critic (A2C)**	Jointed policy (optimal actor) and value (critic) networks to reduce the variances of gradient and increase the speed of training convergence.	- Better trade-off for bias-variance. - Convergence faster than pure policy gradient methods.	- Less sample-efficient than PPO. - Sensitive to 'Noisy' data of financial market.	A2C able to improve the returns rate over the baseline DRL models (Goluža et al., 2024)	Goluža, S., Kovačević, T., Bauman, T., & Kostanjčar, Z. (2024). Deep reinforcement learning with positional context for intraday trading. Ithaca: doi: https://doi.org/10.1007/s12530-024-09593-6
4	**Twin Delayed DDPG (TD3)**	TD3 improved DDPG by using the double Q-learning clipped to reduce the overestimation bias and delayed of policy update for stability.	- More stable of training in continuous spaces - Better sample efficiency	- Sensitive to 'Noisy' data of financial market. - Additional computational overhead required.	TD3 achieved the higher cumulative returns and lower down the drawdowns than DDPG and PPO (Majidi et al., 2022)	Majidi, N., Shamsi, M., & Marvasti, F. (2022). Algorithmic trading using continuous action space deep reinforcement learning. Ithaca: Retrieved from https://vpn.utm.my/working-papers/algorithmic-trading-using-continuous-action-space/docview/2723274890/se-2
5	**Soft Actor-Critic (SAC)**	SAC is the off-policy actor-critic algorithm that can help in optimize the maximum entropy objective and will helps in exploration and robustness	- Robust toward hyperparameter variations - Have a strong exploration in continuous action spaces	- Complexity of computational - Tunning issues for financial data.	SAC agents overperforming than PPO and DDPG in the trading multiple assets with lower risk and high stability (Kong et al., 2023)	Kong, M., & So, J. (2023). Empirical analysis of automated stock trading using deep reinforcement learning. Applied Sciences, 13(1), 633. doi:https://doi.org/10.3390/app13010633

CHAPTER 3: RESEARCH METHODOLOGY

Research Framework Overview



Data Collection

Date	Open	High	Low	Close	Adj Close	Volume
May 9, 2025	5,679.65	5,691.69	5,644.15	5,659.91	5,659.91	4,645,090,000
May 8, 2025	5,663.60	5,720.10	5,635.38	5,663.94	5,663.94	5,627,400,000
May 7, 2025	5,614.18	5,654.73	5,578.64	5,631.28	5,631.28	4,987,440,000
May 6, 2025	5,605.87	5,649.58	5,586.04	5,606.91	5,606.91	4,717,260,000
May 5, 2025	5,655.32	5,683.38	5,634.48	5,650.38	5,650.38	4,358,260,000
May 2, 2025	5,645.88	5,700.70	5,642.28	5,686.67	5,686.67	4,854,380,000
May 1, 2025	5,625.14	5,658.91	5,597.35	5,604.14	5,604.14	4,935,270,000
Apr 30, 2025	5,499.44	5,581.84	5,433.24	5,569.06	5,569.06	5,449,490,000
Apr 29, 2025	5,508.87	5,571.95	5,505.70	5,560.83	5,560.83	4,747,150,000
Apr 28, 2025	5,529.22	5,553.66	5,468.64	5,528.75	5,528.75	4,257,880,000
Apr 25, 2025	5,489.73	5,528.11	5,455.86	5,525.21	5,525.21	4,236,580,000
Apr 24, 2025	5,381.38	5,489.40	5,371.96	5,484.77	5,484.77	4,697,710,000
Apr 23, 2025	5,395.92	5,469.69	5,356.17	5,375.86	5,375.86	5,371,390,000
Apr 22, 2025	5,207.67	5,309.61	5,207.67	5,287.76	5,287.76	4,666,950,000
Apr 21, 2025	5,232.94	5,232.94	5,101.63	5,158.20	5,158.20	4,226,340,000
Apr 17, 2025	5,305.45	5,328.31	5,255.58	5,282.70	5,282.70	4,714,880,000
Apr 16, 2025	5,335.75	5,367.24	5,220.79	5,275.70	5,275.70	4,607,750,000
Apr 15, 2025	5,411.99	5,450.41	5,386.44	5,396.63	5,396.63	4,317,110,000
Apr 14, 2025	5,441.96	5,459.46	5,358.02	5,405.97	5,405.97	5,031,440,000
Apr 11, 2025	5,255.56	5,381.46	5,220.77	5,363.36	5,363.36	5,602,550,000
Apr 10, 2025	5,353.15	5,353.15	5,115.27	5,268.05	5,268.05	6,677,140,000
Apr 9, 2025	4,965.28	5,481.34	4,948.43	5,456.90	5,456.90	9,489,600,000
Apr 8, 2025	5,193.57	5,267.47	4,910.42	4,982.77	4,982.77	7,408,140,000
Apr 7, 2025	4,953.79	5,246.57	4,835.04	5,062.25	5,062.25	8,691,980,000

- Stock Data:
  - S&P 500 historical OHLC data and volume data
  - OHLC data = Open, High, Low, Close prices
  - Volume data = Daily trading volume
- Source: Yahoo Finance
- Time Period : 01/01/2016 to 01/01/2024

Data Cleaning and Preprocessing

- Handling Missing Data
  - Interpolation (continuous time-series data)
- Removing Outliers
  - Use statistical methods (Z-score or IQR)
  - To ensure the model does not learn from irregular, unrepresentative data.

- **Formula for Standardization:**

$$Z = \frac{X - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation.

$$Lower\ Bound = Q_1 - 1.5IQR$$

$$Upper\ Bound = Q_3 - 1.5IQR$$

Model/Problem Definition

- Research Aim: Use DRL to create an automated trading agent for stock market decisions with S&P 500 data.
- MDP Framework: Agent learns optimal trading (buy, hold, sell) to maximize cumulative return.
- State Space: Features from S&P 500 data: OHLC, closing price, moving average, and volume.
- Action Space: Discrete actions (buy, hold, sell) at each time step.
- Reward Function: $R_t = P_{t+1} - P_t$
- Objective: Maximize cumulative reward using discounted return with factor γ ($0 \leq \gamma \leq 1$).
- Evaluation Metrics: Compare DRL models against baseline strategies (buy/sell) based on profitability and risk-adjusted return.
- Goal: Identify the best model for long-term stock market strategy.

Model Selection

UTM UNIVERSITI TEKNOLOGI MALAYSIA

No	Model Name	Function of model in trading	Strength of model	Limitation of model	Results of paper	Citation
1	**Deep-Q-Network (DQN)**	By performing the deep neutral networks to calculate the Q-values, discrete trading actions (Sell, Hold, Buy) based on the mapping status (observations of market)	- Able to handles the high dimensional state spaces. - Without the handcrafted features, able to learn directly from the raw data (price of stock data)	- Trends to overestimate Q-values that may affect the grade of policy quality. - Only suitable for the discrete action spaces.	DQN-based trading outperformed rule-based strategies with higher cumulative returns and Sharpe ratios on historical stock data (Otabek et al., 2024).	Otabek, S., & Choi, J. (2024). Multi-level deep Q-networks for bitcoin trading strategies. Scientific Reports (Nature Publisher Group), 14(1), 771. doi: https://doi.org/10.1038/s41598-024-51408-w
2	**Proximal Policy Optimization (PPO)**	By the surrogate clipped objectives to balance the exploration and exploitation. It will help in stable policy update for discrete and continuous spaces.	- Efficient of sample and robust to hyperparameter choices. - Well handle of stochastics policy and helps in improvement of exploration.	- intensive computation - May achieve the local optima without the adequate exploration.	PPO agents adapt well toward the dynamics market and outperformed than DQN and A2C models in portfolio management (Sun., 2023)	Sun, Q. (2023). Reinforcement learning algorithms for stock trading (Order No. 31765482). Available from ProQuest Dissertations & Theses Global. (3186188497). Retrieved from https://vpn.utm.my/dissertations-theses/reinforcement-learning-algorithms-stock-trading/docview/3186188497/se-2
3	**Advantage Actor-Critic (A2C)**	Jointed policy (optimal actor) and value (critic) networks to reduce the variances of gradient and increase the speed of training convergence.	- Better trade-off for bias-variance. - Convergence faster than pure policy gradient methods.	- Less sample-efficient than PPO. - Sensitive to 'Noisy' data of financial market.	A2C able to improve the returns rate over the baseline DRL models (Goluža et al., 2024)	Goluža, S., Kovačević, T., Bauman, T., & Kostanjčar, Z. (2024). Deep reinforcement learning with positional context for intraday trading. Ithaca: doi: https://doi.org/10.1007/s12530-024-09593-6
4	**Twin Delayed DDPG (TD3)**	TD3 improved DDPG by using the double Q-learning clipped to reduce the overestimation bias and delayed of policy update for stability.	- More stable of training in continuous spaces - Better sample efficiency	- Sensitive to 'Noisy' data of financial market. - Additional computational overhead required.	TD3 achieved the higher cumulative returns and lower down the drawdowns than DDPG and PPO (Majidi et al., 2022)	Majidi, N., Shamsi, M., & Marvasti, F. (2022). Algorithmic trading using continuous action space deep reinforcement learning. Ithaca: Retrieved from https://vpn.utm.my/working-papers/algorithmic-trading-using-continuous-action-space/docview/2723274890/se-2
5	**Soft Actor-Critic (SAC)**	SAC is the off-policy actor-critic algorithm that can help in optimize the maximum entropy objective and will helps in exploration and robustness	- Robust toward hyperparameter variations - Have a strong exploration in continuous action spaces	- Complexity of computational - Tunning issues for financial data.	SAC agents overperforming than PPO and DDPG in the trading multiple assets with lower risk and high stability (Kong et al., 2023)	Kong, M., & So, J. (2023). Empirical analysis of automated stock trading using deep reinforcement learning. Applied Sciences, 13(1), 633. doi:https://doi.org/10.3390/app13010633

Model Selection (DQN, PPO, SAC)

No	Model	Reason
1	PPO	- Balance training stability and sample efficiency. - Able to handle discrete and continuous action spaces
2	DQN	- Able to handles the high dimensional state spaces - Basic DRL model
3	SAC	- Efficient Exploration - Sample efficient

Model Selection (DQN)

- **Deep Q-Network (DQN)**: A model-free, off-policy algorithm combining Q-learning and deep neural networks to approximate the Q-function, ideal for large state spaces like market data.
- **Objective**: Learn the optimal policy **π∗(s)** to maximize cumulative rewards by updating Q-values using deep neural networks.

Q-value Formula:

$$Q(s_t, a_t) = r_{t+k} + \gamma \cdot \max_a Q(s_{t+1}, a)$$

Q-value Update Formula:

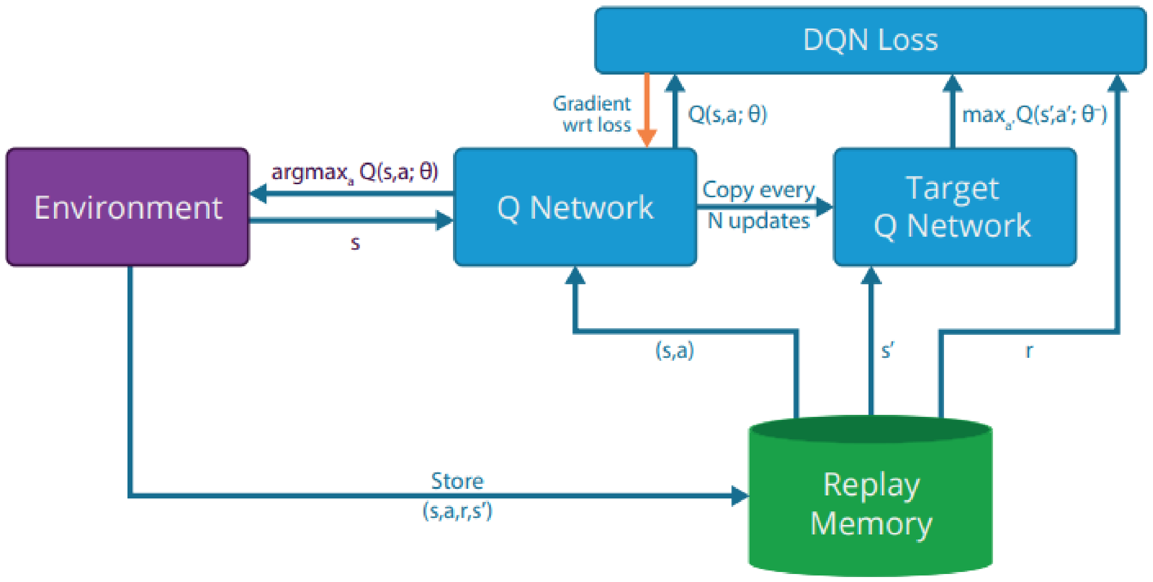
$$Q(s_t, a_t) = r_t + \gamma \cdot Q(s_{t+1}, a_{t+1}) \text{ (target Q-network)}$$

Loss Function: Mean squared error (MSE) between predicted and target Q-values:

$$\text{MSE} = \left(Q(s_t, a_t) - \hat{Q}(s_t, a_t)\right)^2$$

Q-network Update: Adjust weights θ using gradient descent:

$$\theta = \theta - \alpha \cdot \nabla_\theta \text{Loss}$$



Model Selection (SAC)

- **Soft Actor-Critic (SAC)**: A reinforcement learning (RL) algorithm used primarily in robotics to maximize expected long-term rewards and entropy.

State-Value Function: Learned by minimizing squared residual error:

$$V(s) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

Gradient:

$$\nabla_\theta V(s)$$

Q-function (Quality Function): Learned using the state-action value:

$$Q(s,a) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

Next step Q-function:

$$Q'(s,a) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

Gradient for Q-function:

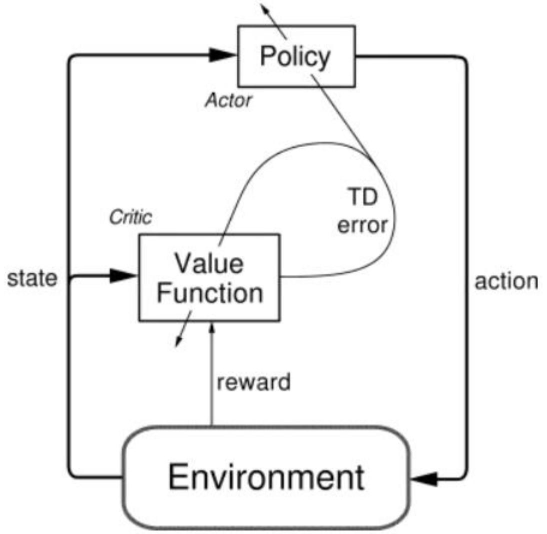
$$\nabla_\theta Q(s,a)$$

Policy Learning: The policy is learned by maximizing the expected reward:

$$\pi(a|s) = \arg\max_a Q(s,a)$$

Gradient:

$$\nabla_\theta \pi(a|s)$$



Model Selection (PPO)

- **PPO Overview**: A policy gradient-based DRL model developed by OpenAI to balance exploitation and exploration. It prevents large policy updates and maintains stability in learning.

Clipped Surrogate Objective:

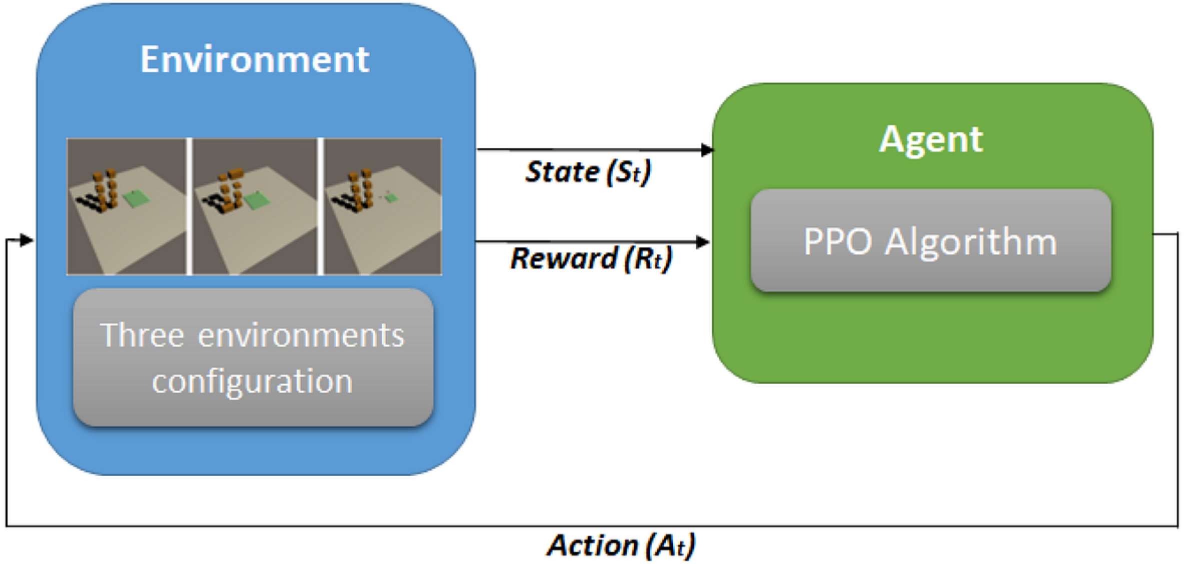
$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$

Total Loss Function:

$$L(\theta) = \mathbb{E}_t \left[L^{CLIP}(\theta) - c_1 L^V(\theta) + c_2 S[\pi](\theta)\right]$$

Advantage Estimation: Use **Generalized Advantage Estimation (GAE)** to calculate the advantage:

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \dots$$



Data Normalization & Splitting

- Normalization and Scaling
  - To ensure that the features are on the same scale and improve model training.
- Data Splitting (80% Training and 20% Test)

- **Formula for Min-Max Scaling:**

$$\text{Scaled Value} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

- **Formula for Standardization:**

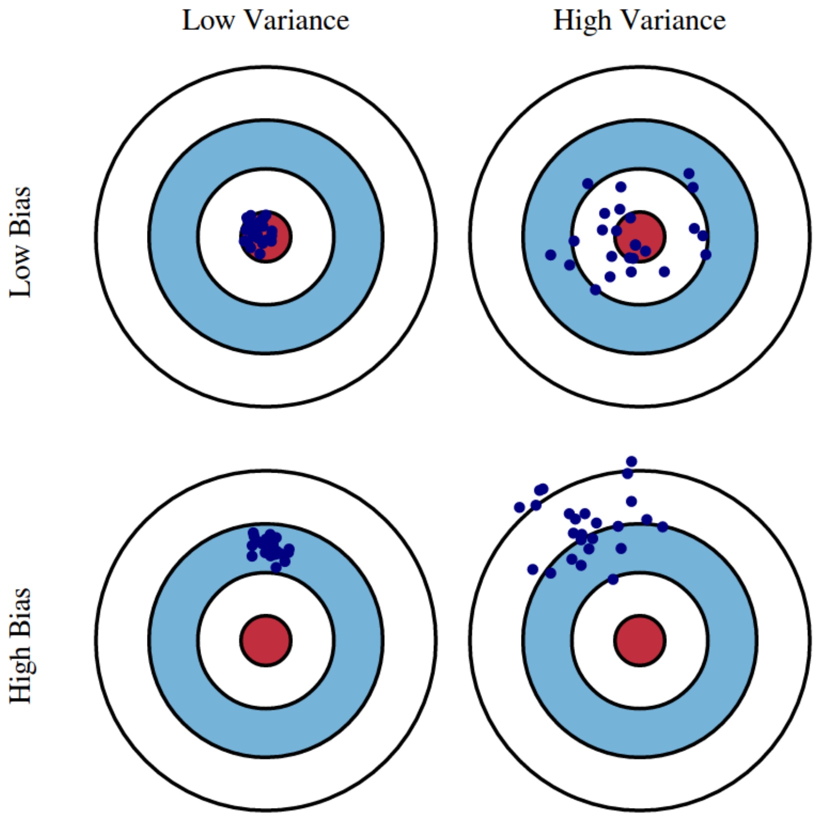
$$Z = \frac{X - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation.

Model Training

Training Loop:

- Environmental Initialization: Initialize the state space from input data.
- Action Selection: Choose an action using DQN, PPO, or SAC strategies.
- Executing Action: Perform the selected action (buy, sell, or hold).Reward Calculation: Calculate reward based on the profit or loss.
- Next State Observation: Observe the new market state after the action.
- Experience Replay (DQN only): Store experiences and sample from buffer for training.
- Model Update: Update the model using Bellman equation (DQN), clipped objective (PPO), or soft Bellman backup (SAC).
- Episode Completion: Terminate after a predefined number of time steps.
- Repeat Process: Continue the cycle to improve the model's policy and reward.



Hyperparameter Tuning

General Hyperparameters for DQN, PPO, and SAC:

- Learning rate, α
- Discount factor, γ
- Batch size
- Exploration rate
- Replay buffer size
- Number of timesteps per episode

- Model-Specific Hyperparameters:
  - DQN:
    - Target Network Update Frequency: Controls target Q-network updates.
    - Double Q-Learning: Reduces overestimation bias in Q-values.
  - PPO:
    - Clip Range: Prevents large policy changes for stability.
    - GAE Lambda: Balances bias and variance in advantage estimation.
  - SAC:
    - Entropy Coefficient: Controls exploration-exploitation trade-off.
    - Target Entropy: Determines the model's exploration during training.
- Hyperparameter Tuning Methods:
  - Grid Search: Exhaustive search of predefined hyperparameter combinations.
    - Steps: Define ranges → Create grid → Train & evaluate → Select best performance → Fine-tune.
  - Random Search: Randomly selects hyperparameter combinations.
    - Steps: Define ranges → Randomly sample → Train & evaluate → Select best performance.
  - Cross-Validation: Reduces overfitting, splits data into k-folds for training and validation.
    - Steps: Split data → Train on k-1 folds → Validate on remaining fold → Average performance.

Evaluation Metrics

- **F1-Score** - evaluation the accuracy of the model
  - Formula:

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Cumulative Return** - Measures the overall profitability of the model's trading decisions over time.
  - Formula:

$$\text{Cumulative Return} = \frac{\text{Ending Portfolio Value} - \text{Starting Portfolio Value}}{\text{Starting Portfolio Value}} \times 100$$

Dashboard Development

Visualization:

- Build an interactive dashboard to show:
  - Portfolio growth over time.
  - Comparison of total return for DQN, PPO, and SAC.
  - Real-time agent performance updates.

Model Comparison and Performance Analysis

Performance Comparison:

•Compare **DQN**, **PPO**, and **SAC** based on:

- Return optimization (total return).
- F1-Score
- Model robustness and stability over time.

Analysis:

•Identify the strengths and weaknesses of each model.

•Discuss trade-offs between risk and return in each DRL model.

CHAPTER 4:
INITIAL FINDINGS

Data Preprocessing/ Data cleaning

- Handles the missing data by interpolation

```
[ ] # Check if there are any missing values in the data
    if missing_data.any():
        print("Missing values found. Interpolation will be performed.")

        # Perform linear interpolation to fill missing values
        data_interpolated = data.interpolate(method='time', limit_direction='both')

        print("Interpolation completed.")

    else:
        # If no missing data, skip interpolation
        data_interpolated = data
        print("No missing values found. Skipping interpolation.")

    # Verify that there are no more missing values
    missing_data_after = data_interpolated.isnull().sum()
    print("Missing values after interpolation:\n", missing_data_after)
```

```
No missing values found. Skipping interpolation.
Missing values after interpolation:
 Price   Ticker
Close   ^GSPC     0
High    ^GSPC     0
Low     ^GSPC     0
Open    ^GSPC     0
Volume  ^GSPC     0
dtype: int64
```

Data Preprocessing/ Data cleaning

UTM UNIVERSITI TEKNOLOGI MALAYSIA

- Remove outliers

```
from scipy.stats import zscore

# Calculate z-scores for the close prices
data['zscore'] = zscore(data['Close'])

# Filter out data points where z-score is greater than 3 (outliers)
data_clean = data[data['zscore'].abs() <= 3]

# Drop the z-score column for the final cleaned data
data_clean = data_clean.drop(columns=['zscore'])

print(data_clean.head())
```

```
Price              Close         High          Low         Open      Volume
Ticker             ^GSPC        ^GSPC        ^GSPC        ^GSPC       ^GSPC
Date
2016-01-04   2012.660034  2038.199951  1989.680054  2038.199951  4304880000
2016-01-05   2016.709961  2021.939941  2004.170044  2013.780029  3706620000
2016-01-06   1990.260010  2011.709961  1979.050049  2011.709961  4336660000
2016-01-07   1943.089966  1985.319946  1938.829956  1985.319946  5076590000
2016-01-08   1922.030029  1960.400024  1918.459961  1945.969971  4664940000
```

Data Preprocessing/ Data cleaning

- Final checking for the data set.

```
# Check for remaining missing values
print(data.isnull().sum())

# Check for duplicated rows
print(data.duplicated().sum())

# Display the final cleaned data
print(data.head())
```

```
Price             Ticker
Close             ^GSPC     0
High              ^GSPC     0
Low               ^GSPC     0
Open              ^GSPC     0
Volume            ^GSPC     0
zscore                      0
Close Normalized            0
dtype: int64
0
Price             Close        High         Low          Open         Volume      \
Ticker            ^GSPC        ^GSPC        ^GSPC        ^GSPC        ^GSPC
Date
2016-01-04  2012.660034  2038.199951  1989.680054  2038.199951  4304880000
2016-01-05  2016.709961  2021.939941  2004.170044  2013.780029  3706620000
2016-01-06  1990.260010  2011.709961  1979.050049  2011.709961  4336660000
2016-01-07  1943.089966  1985.319946  1938.829956  1985.319946  5076590000
2016-01-08  1922.030029  1960.400024  1918.459961  1945.969971  4664940000

Price          zscore Close Normalized
Ticker
Date
2016-01-04 -1.495889          0.061864
2016-01-05 -1.491027          0.063229
2016-01-06 -1.522778          0.054315
2016-01-07 -1.579402          0.038420
2016-01-08 -1.604683          0.031323
```

Data Normalization and Spliting

UTM UNIVERSITI TEKNOLOGI MALAYSIA

- Normalized the split the data into train and test set

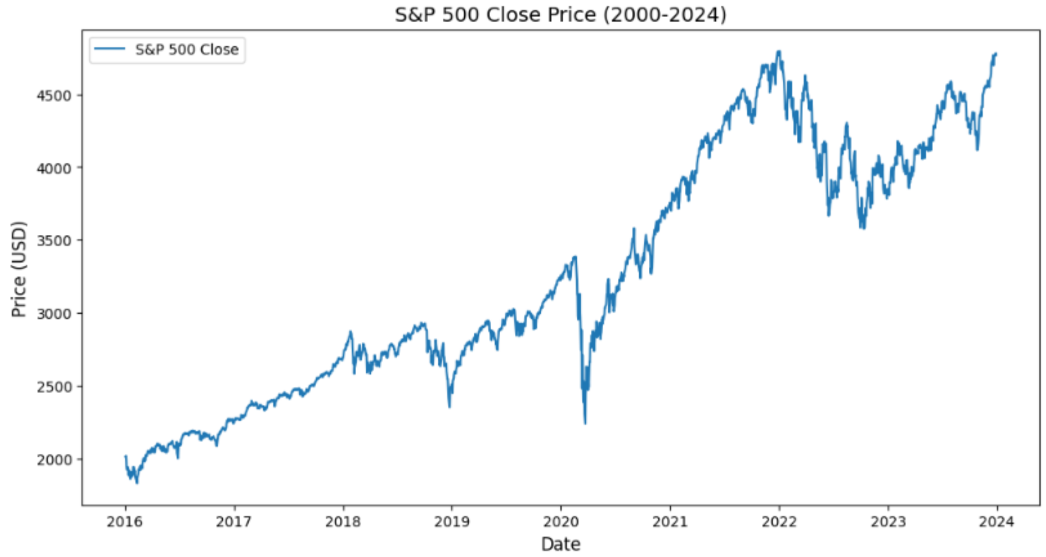
```python
import pandas as pd

# Load your data (replace with the actual file path or DataFrame)
# Assuming your data has a column 'Date' and 'Close' (or other stock-related data)

# Convert the Date column to datetime if not already done
data.index = pd.to_datetime(data.index)

# Calculate the split index
train_size = int(len(data) * 0.8)  # 80% for training

# Split the data into train and test sets
train_data = data.iloc[:train_size]
test_data = data.iloc[train_size:]

# Display the results
print("Train Data (80%):")
print(train_data.head())  # Show the first few rows of the train data
print("\nTest Data (20%):")
print(test_data.head())  # Show the first few rows of the test data
```

```
Train Data (80%):
Price             Close         High          Low         Open      Volume  \
Ticker            ^GSPC        ^GSPC        ^GSPC        ^GSPC       ^GSPC
Date
2016-01-04  2012.660034  2038.199951  1989.680054  2038.199951  4304880000
2016-01-05  2016.709961  2021.939941  2004.170044  2013.780029  3706620000
2016-01-06  1990.260010  2011.709961  1979.050049  2011.709961  4336660000
2016-01-07  1943.089966  1985.319946  1938.829956  1985.319946  5076590000
2016-01-08  1922.030029  1960.400024  1918.459961  1945.969971  4664940000

Price         zscore Close Normalized
Ticker
Date
2016-01-04 -1.495889         0.061864
2016-01-05 -1.491027         0.063229
2016-01-06 -1.522778         0.054315
2016-01-07 -1.579402         0.038420
2016-01-08 -1.604683         0.031323

Test Data (20%):
Price             Close         High          Low         Open      Volume  \
Ticker            ^GSPC        ^GSPC        ^GSPC        ^GSPC       ^GSPC
Date
2022-05-24  3941.479980  3955.679932  3875.129883  3942.939941  4923190000
2022-05-25  3978.729980  3999.330078  3925.030029  3929.590088  4802560000
2022-05-26  4057.840088  4075.139893  3984.600098  3984.600098  4709970000
2022-05-27  4158.240234  4158.490234  4077.429932  4077.429932  4375620000
2022-05-31  4132.149902  4168.339844  4104.879883  4151.089844  6822640000

Price         zscore Close Normalized
Ticker
Date
2022-05-24  0.819501         0.711850
2022-05-25  0.864216         0.724403
2022-05-26  0.959182         0.751062
2022-05-27  1.079704         0.784895
2022-05-31  1.048384         0.776103
```

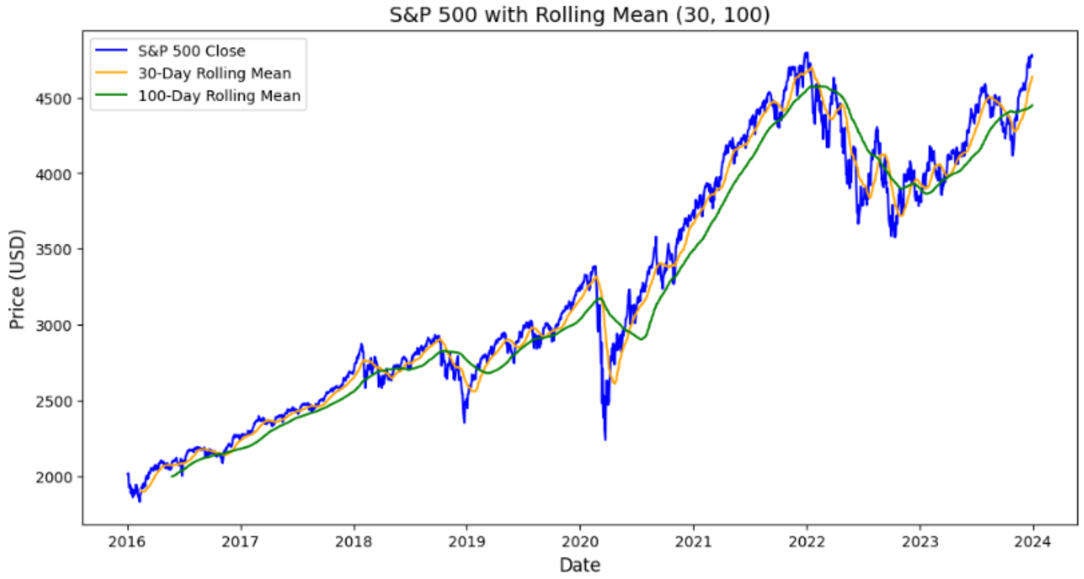
EDA

- Descriptive Statistics Analysis

```
# Plotting the Closing Price
plt.figure(figsize=(12,6))
plt.plot(data['Close'], label='S&P 500 Close')
plt.title('S&P 500 Close Price (2000-2024)', fontsize=14)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Price (USD)', fontsize=12)
plt.legend()
plt.show()
```



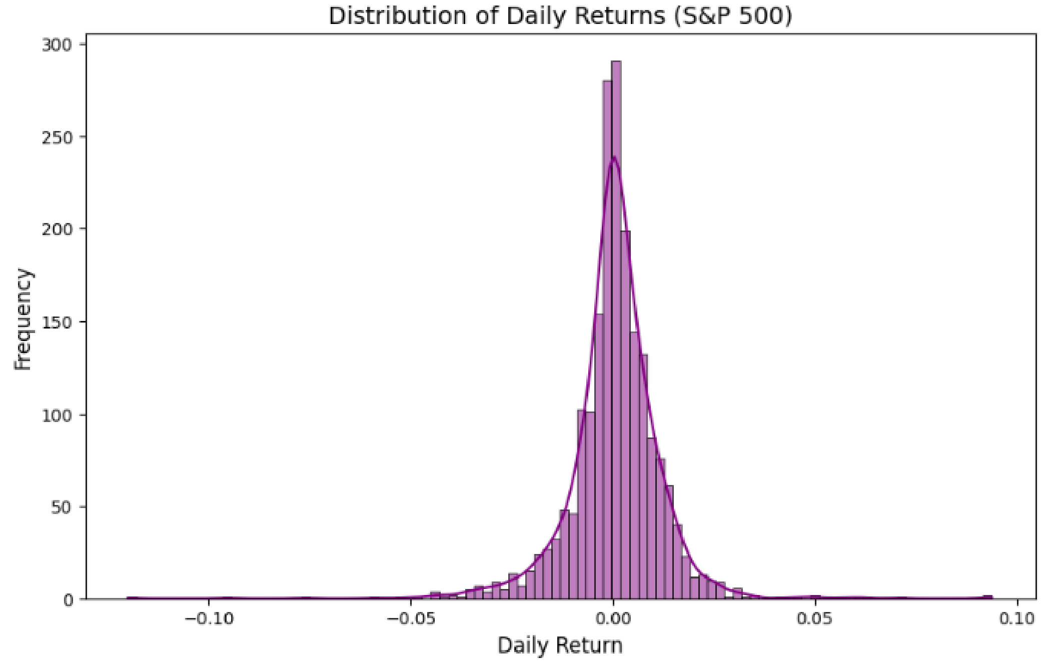

```
# Compute 30-day and 100-day rolling mean and standard deviation
data['30_day_MA'] = data['Close'].rolling(window=30).mean()
data['100_day_MA'] = data['Close'].rolling(window=100).mean()
data['30_day_STD'] = data['Close'].rolling(window=30).std()

# Plotting the closing price along with rolling mean and std
plt.figure(figsize=(12,6))
plt.plot(data['Close'], label='S&P 500 Close', color='blue')
plt.plot(data['30_day_MA'], label='30-Day Rolling Mean', color='orange')
plt.plot(data['100_day_MA'], label='100-Day Rolling Mean', color='green')
plt.title('S&P 500 with Rolling Mean (30, 100)', fontsize=14)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Price (USD)', fontsize=12)
plt.legend()
plt.show()
```



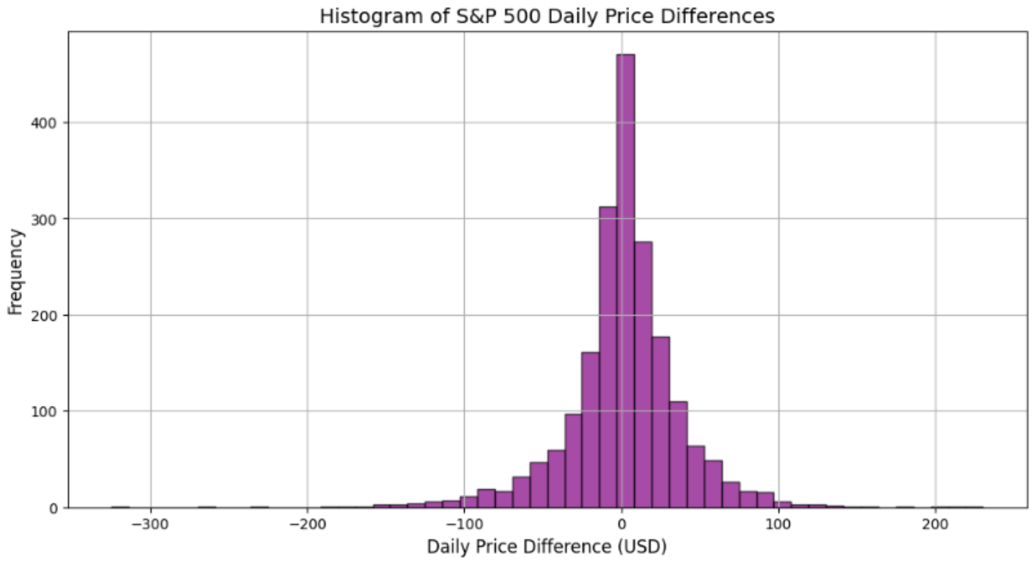
EDA

- Descriptive Statistics Analysis

```
[ ] # Compute daily returns
    data['Daily Return'] = data['Close'].pct_change()

    # Plot the distribution of daily returns
    plt.figure(figsize=(10,6))
    sns.histplot(data['Daily Return'], bins=100, kde=True, color='purple')
    plt.title('Distribution of Daily Returns (S&P 500)', fontsize=14)
    plt.xlabel('Daily Return', fontsize=12)
    plt.ylabel('Frequency', fontsize=12)
    plt.show()
```



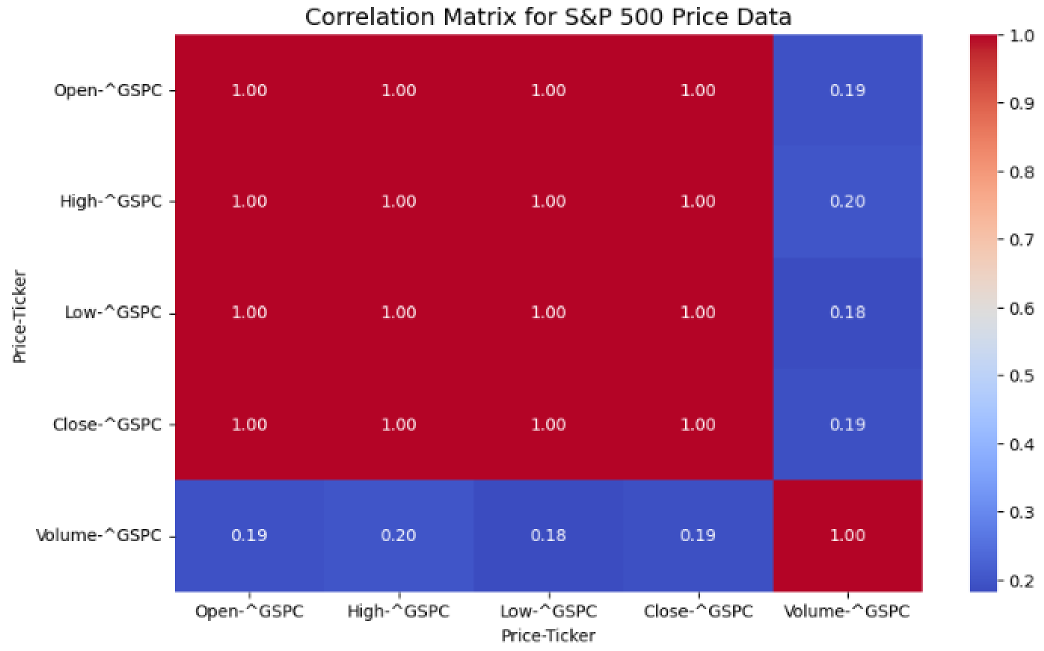

```
[ ] # Plotting the histogram for daily price differences
    plt.figure(figsize=(12, 6))
    plt.hist(data['Daily Difference'], bins=50, color='purple', edgecolor='black', alpha=0.7)
    plt.title('Histogram of S&P 500 Daily Price Differences', fontsize=14)
    plt.xlabel('Daily Price Difference (USD)', fontsize=12)
    plt.ylabel('Frequency', fontsize=12)
    plt.grid(True)
    plt.show()
```



EDA

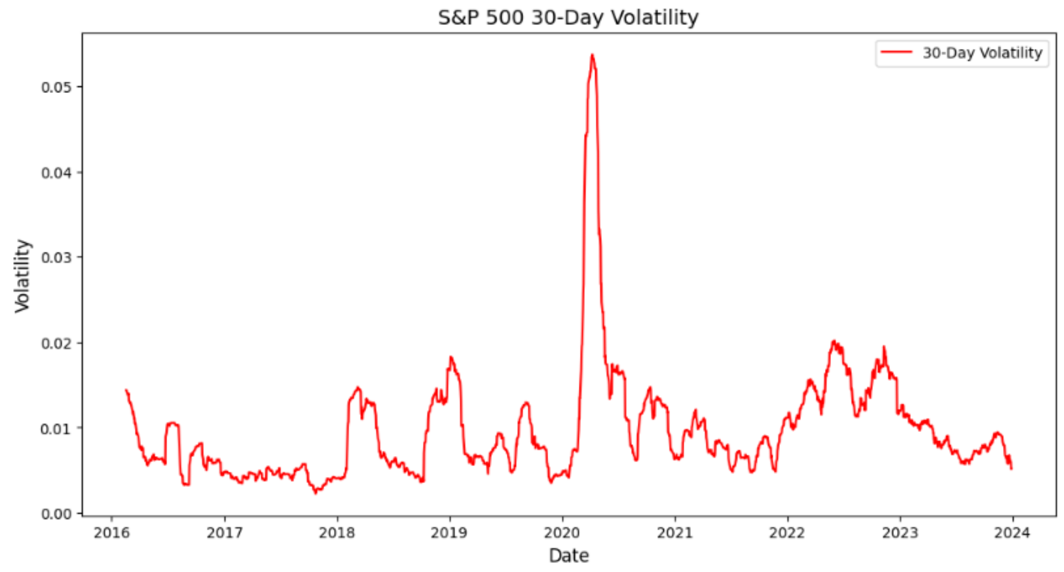
- Correlation Heat Map

```python
# Create a correlation heatmap
corr_matrix = data[['Open', 'High', 'Low', 'Close', 'Volume']].corr()

plt.figure(figsize=(10,6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix for S&P 500 Price Data', fontsize=14)
plt.show()
```



- Volatility plot of market

```python
# Compute 30-day volatility (standard deviation of daily returns)
data['30_day_volatility'] = data['Daily Return'].rolling(window=30).std()

# Plotting volatility
plt.figure(figsize=(12,6))
plt.plot(data['30_day_volatility'], label='30-Day Volatility', color='red')
plt.title('S&P 500 30-Day Volatility', fontsize=14)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Volatility', fontsize=12)
plt.legend()
plt.show()
```

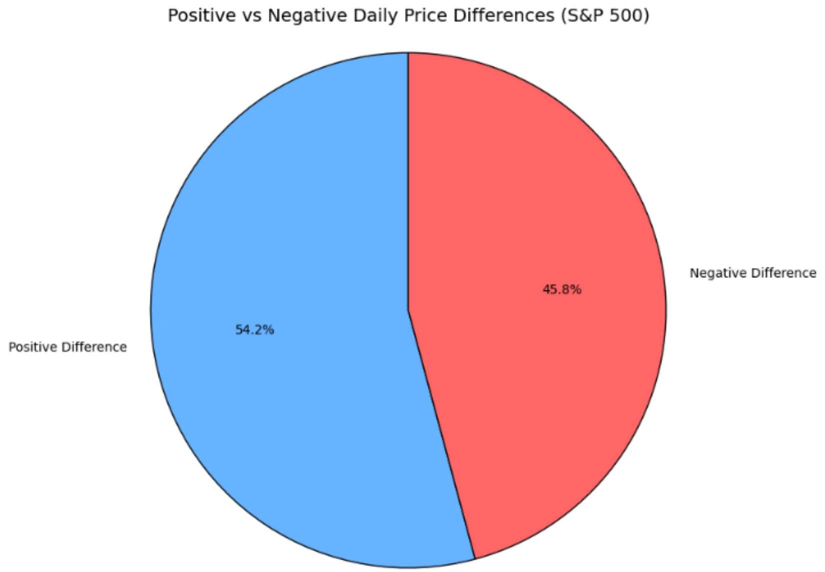


www.utm.my

EDA

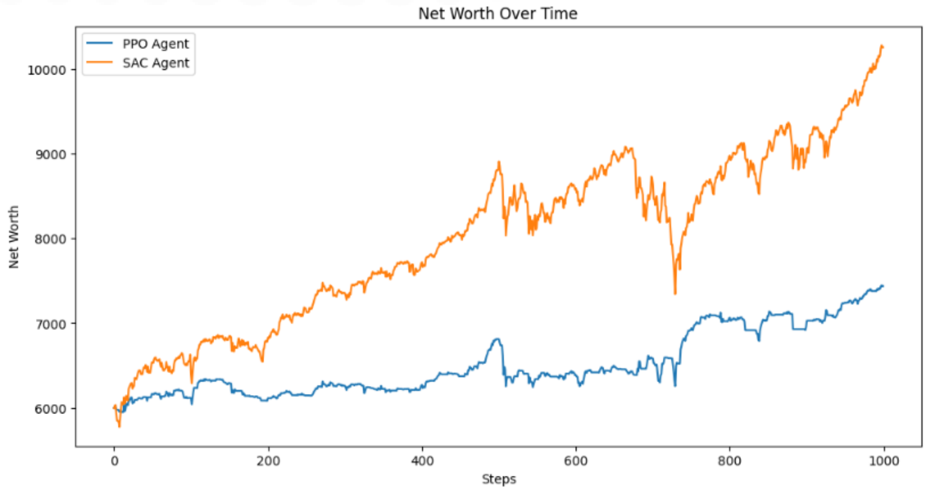
- Positive vs Negative Daily Price Differences (S&P 500)

```python
# Count the number of positive and negative differences
positive_diff = (data['Difference Type'] == 'Positive').sum()
negative_diff = (data['Difference Type'] == 'Negative').sum()

# Data for the pie chart
labels = ['Positive Difference', 'Negative Difference']
sizes = [positive_diff, negative_diff]
colors = ['#66b3ff', '#ff6666']  # Blue for positive, Red for negative

# Check if there are any positive or negative differences to plot
if sum(sizes) > 0:
    # Plot the pie chart
    plt.figure(figsize=(8, 8))
    plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90, wedgeprops={'edgecolor': 'black'})
    plt.title('Positive vs Negative Daily Price Differences (S&P 500)', fontsize=14)
    plt.axis('equal')  # Equal aspect ratio ensures the pie chart is drawn as a circle.
    plt.show()
else:
    print("No positive or negative daily differences to plot for the pie chart.")
```

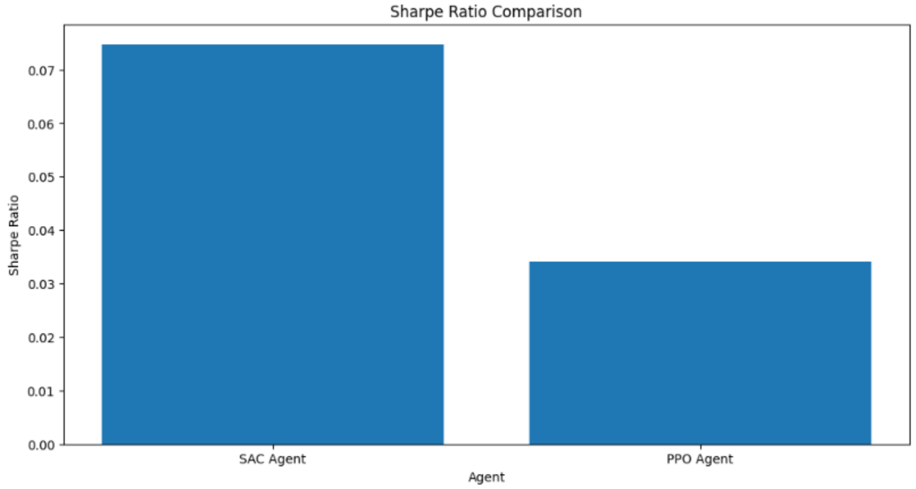


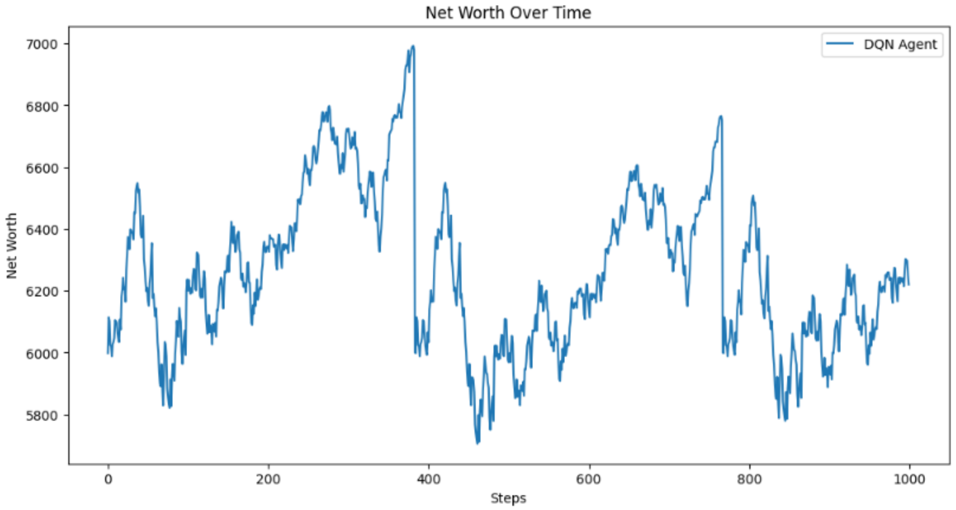
Expected Outcomes (Preliminary Results)

UTM UNIVERSITI TEKNOLOGI MALAYSIA

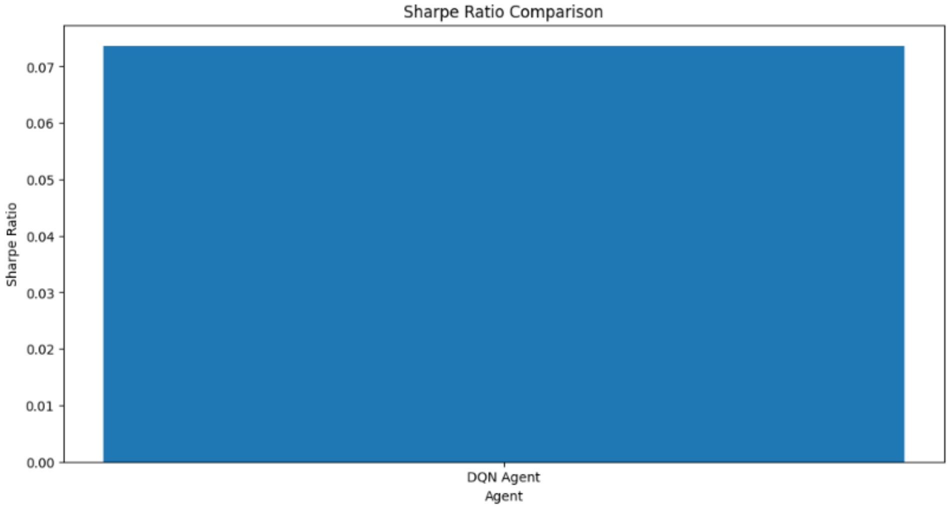


	Agent	Return	Standard Deviation	Sharpe Ratio
1	SAC Agent	0.000566	0.00757	0.074730
0	PPO Agent	0.000133	0.00388	0.034171





	Agent	Return	Standard Deviation	Sharpe Ratio
0	DQN Agent	0.000561	0.007621	0.073644



CHAPTER 5: CONCLUSION

Future Works

- The dashboard development for the performance analysis to visualize and analyze the performance of different DRL models (DQN, SAC and PPO).
  - Sharpe ratio, net worth over time and agent comparison.
  - F1 Score
  - Cumulative Return Rate
- The policy analysis of 3 DRL models:
  - Analyze performance metrics: return, Sharpe ratio, and standard deviation.
  - Behavioral insight: breakdown of each DRL model.
  - Correlate actions with changes in net worth.

THANK YOU

univteknologimalaysia utm.my utmofficial