Chapter 3

RESEARCH METHODOLOGY

3.1 Introduction

The research mainly focusses on application of deep reinforcement learning (DRL) techniques to develop an automated trading system for decision making in the stock market. Automated trading in decision making including the buy, sell or hold decisions to achieve the optimal cumulative return over time. The traditional trading strategy relies often on the fixed rules or can be called as supervised learning. However, DRL able to provide the powerful framework toward model specially handling the dynamics, uncertain of the financial stock market with learning the optimal trading policy through interaction with the real-time environment. The trading environment is modeled as the Markov Decision Process (MDP) which is the agent learning based on the observe the current market state and make the decision (takes an action which is buy, sell and hold) and receive the reward accordingly toward the portfolio performance.

The objective of the DRL agent is to learn a policy π(a|s) that optimize the cumulative discounted reward G_t defined as:

$$G_t = k = \sum_{\infty}^{0} \gamma^k R_{t+k+1} \tag{3.1}$$

Where:

R_t is the immediate reward at time t (the time when profit or loss based on trades)

$\gamma \in [0,1]$ is the discount factor which is determine the vital of future rewards relative to immediate gains. The S&P 500 data enable to provide the high volume of historical data including historical OHLC data and volume data. OHLC data is data for Open, High, Low, Close prices. Volume data is daily trading volume. The technical indicators where the transformed into state representations to guide the agent to make the decision.

As mentioned at Chapter 2 Literature Review, the input data will be Standard and Poor 500 (S&P 500). The range of data is from 01/01/2016 to 01/01/2024 in the Yahoo Finance. The adjusted closing price of the S&P 500 will be taken for analysis. The reason for using an adjusted closing price is that some of the stock may have dividends share splitting or consolidation that will impact the price of stocks which belong to the S&P 500. After the price is adjusted, the price before the dividend or conditions above will be amended to ensure the consistency of the data.

After the data is collected, some descriptive analysis and data pre-processing will be conducted to have a holistic overview of the data. Then, the dataset will be split into two which are training data and test data with the assistance of some library in Python. The model selection based on the ability, strength and weakness of the deep reinforcement learning (DRL) models. The model selected will be used to train on the training data. After the models are trained, they will try to run the test data and then the error between the decision making and actual data will be measured in the unit of F1-score and cumulative return over time. In the research that found in Chapter 2 (Literature Review), DRL models in trading field are evaluated into Deep Q-Network (DQN), Soft Actor-Critic (SAC), and Proximal Policy Optimization (PPO). Each algorithm represents the different approaches by using the basic as the MDP. DQN suited for discrete action spaces, SAC optimized for continuous control and robust exploration, and PPO balancing training stability and sample efficiency through policy gradient methods.

Some parts from Box-Jenkins method will be applied in order to find the correlation between the moving average and the contemporary index's price, and also those moving average data will be generated in order to feed into the algorithms. Then, a dashboard that shows the effect of the models will be developed in order to provide clearer insights for the investors and traders. The performance of the DRL models will be compared based on the ability to generate the profitable trading strategy based on the historical S&P 500 data with the

optimal cumulative returns. The methodology outlines the steps taken to prepare the data, define the problem, select and train models, tune hyperparameters, evaluate performance, and compare results, ultimately contributing to the development of an efficient RL-based trading system. Figure 3.1 shown the Research Framework for the research.

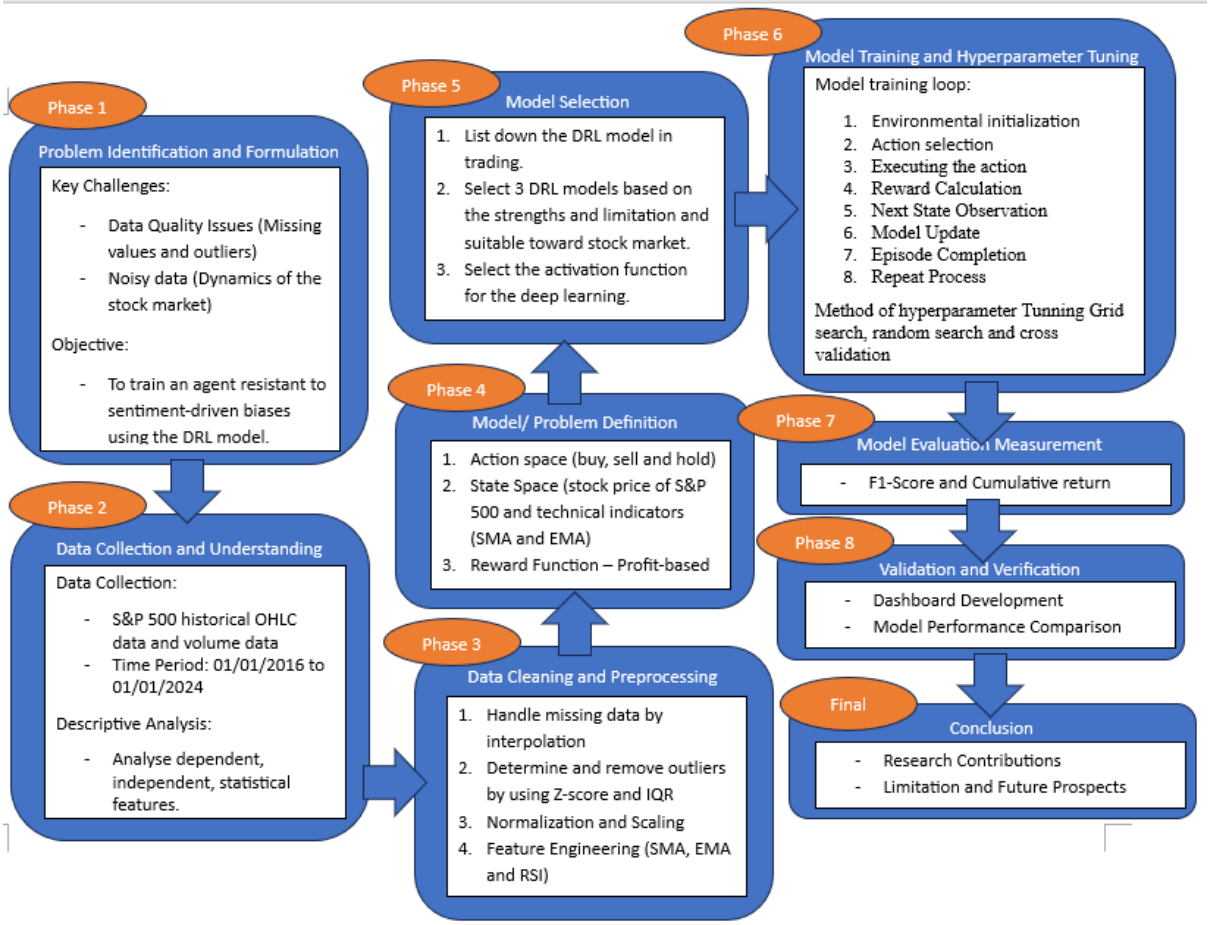

Figure 3.1 Research Framework

3.2 Data Cleaning and Pre-processing

Data cleaning and pre-processing is to handle the missing data, remove outliers, normalization and scaling of data to make the data more readable and enable to analyzed to gain more accurate insight.

First step for the data cleaning process is to handle the missing data which means the missing values can be interpolation to make the complexity of the dataset. Missing data is the common issues for the time series data. Those gaps might be will affect the model's training. Interpolation will be applied to handle the missing values in the dataset of the S&P 500. After that, the outliers need to eliminated by using the statistical methods to identify the outliers which is using the Z-score or IQR. Remove outliers is to ensure the model will not train based on the irregular/unrepresentative data. There are the Z-score and IQR formula.

$$Z = \frac{X - \mu}{\sigma} \tag{3.2}$$

Where:

Z = Z-score

X = Value of the data point

μ = mean of the dataset

σ = Standard deviation of the dataset

$$IQR = Q_3 - Q_1 \tag{3.3}$$

Where:

Q3 = 75^{th} percentile

Q1 = 25^{th} percentile

Formula of IQR for the outlier detection,

$$Lower\ Bound = Q_1 - 1.5IQR \tag{3.4}$$

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

If the data point is below lower bound or above the upper bound are the outliers. The outliers may affect the model as the misguided to learn of absurd patterns and lead to low accuracy.

Since the S&P 500 stock price is too large which makes the comparison meaningless. The normalization and scaling are importance to make DRL able to learn or calculate the results that approaches to intrinsic pattern of the data. Normalization and scaling are to ensure the features are on the same scale and able to improve the training of the model. Formula of the scaling is:

$$Scaled\ value = \frac{X - \min(X)}{\max(X) - \min(X)} \tag{3.5}$$

Where:

X = original data point

min(X) = minimum value of the data

max(X) = maximum value of the data

Scaled value is the scaled data point in the range [0,1].

3.3 Model /Problem Definition

The aims of the research is the application of deep reinforcement learning (DRL) techniques to develop an automated trading agent to make the decision toward the stock market. S&P 500 will be the input data for the research. With the Markov Decision Process (MDP) theory, the system designed to let the agent that can learns to make the optimal trading decisions

which is buy, hold and sell based on the historical stock data to optimize the cumulative return rate over time.

The problem including the state space, action space and reward function. For the state space S_t, where the S_t is defined from the S&P 500 data such as OHLC (Open, high, low and close), closing price, moving average, relative strength index (RSI) and volume. The state space is the various features from the S&P 500 as the input of the DRL models to let the agent can learn the policy that dictates the action to take at each of the time step to optimize the return rate over time.

The action space for the research is discrete state which is the buy, hold and sell. Those discrete actions are representing the action taken for the agent can make at each time step as the decision making of trading. The reward function for the model will be evaluated with the cumulative rewards over time which is the financial gain or loss from the specific actions. The reward function R_t at each of the time step as following:

$$R_t = P_{t+1} - P_t \tag{3.6}$$

Where:

R_t is the reward function at the time, t

P_t is the price of the stock at the time, t

P_{t+1} is the price of the stock at the next time step

The aims of the agent are to optimize the cumulative reward over time with using the discounted return G_t to sum up the future rewards, discounting with the factor γ (where $0 \leq \gamma \leq 1$) to prioritize the immediate rewards.

The performance of the DRL models will be evaluated with their profitability and risk adjusted return as the reward function. By comparing each of the models against the baseline

strategy such as buy and sell strategy. The research is to determine which of the model is most effective at the learning a trading policy that performs well toward the S&P 500 data for the long-term stock market strategy. The application of DRL for the automated trading ensure contribute toward the development of more efficient and profitable trading in decision making.

3.4 Model Selection (tanh function / select all activation function)

In the chapter 2, found some of the DRL models that are able to use in the trading system. Based on the chapter 2, the summary table of DRL model in the trading. Deep Reinforcement Learning (DRL) has revolution of the trading algorithms by allowing/train an agent to learn optimum trading policy from the dynamic and complex financial market through trial and error. The reward mechanism was the marking system that based on the policy set to give the reward or penalty toward an agent's actions. There are the summary table of DRL model including their function, strength, limitation and finding for selecting the best 3 model to further research.

Table 3.1 Summary of DRL model 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 et al., (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	- 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

		continuous spaces.				
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 et al., (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 et al., (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	- Robust toward hyperparameter variations - Have a strong exploration in	- Complexity of computational - Tunning issues for financial data.	SAC agents overperforming than PPO and DDPG in the trading multiple assets	Kong et al., (2023). Empirical analysis of automated stock trading using deep reinforcement learning. *Applied Sciences, 13*(1), 633. doi: https://doi.org/10.3390/app13010633

		maximum entropy objective and will helps in exploration and robustness	continuous action spaces		with lower risk and high stability (Kong et al., 2023)	

Based on the Table 3.1, there are 5 DRL model that can used in trading but still not practical yet. Those 5 DRL models illustrate the rich diversity of approaches in automated trading. DQN model is values-based models which is excel in discrete decision but limitation in scalability. Actor-critic methods which are PPO, A2C and SAC are balance policy and value learning, handling continuous actions with the better flexibility. Twin delayed methods address the stability and market complexity issues. Despite advances, there are some of the common limitations including instability of training, sensitivity toward 'Noisy' data and computational overhead.

The strengths, limitations and functions for each of the models are main consideration of the model selection for further research. After reviewing Table 2.1, 3 best-suited models in automated stock market trading are selected as the models that will used in the research. The 3 DRL models are PPO, DQN and SAC.

Table 3.2 Summary of the models selected

No	Models	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

Based on Table 3.2, there are the 3 models that selected as the research model. PPO able to balance the training stability and sample efficiency which is adaptive toward ‘Noisy’ data and non-stationary nature of stock market data. PPO able to handle discrete and continuous action spaces which is versatile toward different trading decision such as portfolio adjustments, buy or sell signals. Clopped objective able to prevent the large destructive policy updates which is able to reduce the risk of performance collapse. DQN model created as the basic of the DRL model that able to handles the high dimensional state spaces for the discrete action spaces with high effectively. With the Actor-critic structure allows agents to learn the complex policy while evaluating for the next better updates. SAC able efficient exploration which is important to

avoid the premature convergence in the dynamics/volatile financial market. With the off-policy training, sample efficient where allowing agents learn faster from the historical data.

The activation functions used in the hidden and output layers lies for the core of any neutral network-based DRL model. Those functions introduce non-linearity which is enable the network to learn the complex/dynamic patterns in the dataset such as the market behaviour and price movements. For the research, the Tanh (hyperbolic tangent) and sigmoid activation function selected for the underlying for neural network architectures that used for DRL models selected which is Deep Q-Network (DQN), Soft Actor-Critic (SAC), and Proximal Policy Optimization (PPO).

The Tanh function is a sigmoidal function which is the outputs values will be the range [-1,1]. Tanh function commonly used for the DRL because of ability to produce zero-centered outputs. Tanh function helps to maintain the balance during the weight updates and prevent the skewed gradients issues.

Formula Tanh function:

$$Tanh\ (x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.7}$$

Where the x is the input to the function which is the weighted sum of the inputs in the neural network layer.

The advantages of Tanh are zero-centered output, smooth gradient and efficient weight updates. Zero-centered output is the range from -1 to 1 and it helps to centering the data and enable to improve the convergence speed during training. Smooth gradient in the Tanh function is to make the suitable for backpropagation. Efficient weight updates to let the training becomes more balance due to the values are centered toward zero and helps to prevent the large oscillations in the process of learning. The challenges of Tanh also quite significant because of the vanishing gradient problem which means if the values is too large or too small, the gradient

of Tanh will be very small and it will affect the network hard to updates efficiently during backpropagation. It will slow down the training process during the deep network.

The sigmoid activation function is one of the sigmoidal activation function which is the output values in the range of 0 to 1. It will help to map the values to probability space and used for the output layer of classification models. In the DRL models, sigmoid function able to applied where the output needs to represent probabilities or called as when the models required to make the binary decision (such as buy or sell).

The formula of sigmoid function:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{3.7}$$

Where:

S(x) is the output of the sigmoid function (range of values from 0 to 1)

e is the Euler's number (around 2.718)

x is the input of the function

The advantages of the sigmoid are probabilistic output and smooth gradient. Probabilistic output as the output values only will between 0 and 1 which is can be interpreted as probabilities. It can help to predict the binary outcomes to make the decision either sell or buy. Smooth gradient of the sigmoid function like Tanh function, which is works well with backpropagation. The challenges of sigmoid function are not zero-centered and vanishing gradient problem. Sigmoid function unlike Tanh function because of the not zero centered which is the output value only 0 to 1 instead of Tanh -1 to 1. It will lead to skewed gradient during backpropagation. The convergence slowdown may happen as the updates to the weights

can be biased toward the positive values. Another challenge of sigmoid function is similar with Tanh function which is the vanishing gradient problem where the input data is too large or too small may affect the deep network to learn effectively during training.

3.4.1 DQN

Deep Q-Network (DQN) is the model-free, off-policy algorithm that can be used in deep reinforcement learning (DRL) which is combination of the Q-learning with the deep neutral networks to approximate the Q-function. DQN suitable for the large state spaces such as the dynamics market data set. In DQN, the neural network that used to approximate the Q-values for state-action pair.

The aim of DQN is to learn the optimal policy π∗(s) which is maximize the cumulative reward over time with keep updating Q-values by using deep neural network. The Q-value for the state-action pair as following:

$$Q(s_t, a_t) = E[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t, a_t] \tag{3.8}$$

Where:

s_t is the state at time, t

a_t is the action taken at time, t

r_{t+k} is the reward at time, t + k

γ is the discount factor which is the range ($0 < \gamma \leq 1$)

The purpose of the formula is to find a policy that can optimize the Q-values over time.

DQN able to utilize the Q-learning with the deep neural network to approximate the Q-values. The methodology for DQN algorithm will be as following:

1. Initialize Parameters and network
    a. Initialize Q-network. The neural network will initialize with the random weights to estimate the Q-value function. As the state s_t as the input and the output of the Q-values for all possible actions.

$$Q(s, a; \theta) \tag{3.9}$$

Where θ is the weights of the neural network

    b. Initialize Target Q-network is the creation of the copy Q-network as the target Q-network which is initialized with the same weights.

2. Interact with the environment
    a. The agent observes the state, s_t of the environment at the each of time step, t
    b. The agent will select the action a_t based on the policy. The epsilon-greedy policy will select as initial policy where the agent will explore more on the random actions with the probability, ϵ and the exploits the optimal action based on the Q-values with the probability 1-ϵ.

$$a_t = \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg\max_a Q(s_t, a; \theta) & \text{with probability } 1 - \epsilon \end{cases}$$

Figure 3.2 Interact with the environment

3. Experience Replay
    a. Store the tuple which is (s_t, a_t, r_t, s_{t+1}) in the replay memory experience
    b. The replay memory experience helps to break the correlation between consecutive samples and it allowing the network to learn more effectively.
4. Sample Mini-Batch and Update Q-Network

a. The random mini-batch of experiences (s_t, a_t, r_t, s_{t+1}) at each time step is sampled from the replay memory experience
b. Targeted Q-network calculated as following:

$$y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-) \tag{3.10}$$

Where θ^- is the targeted weights of the neural network

c. Loss function. The Mean squared error (MSE) between the predicted Q-value and the target was calculated as following:

$$L(\theta) = E\left[\left(y_t - Q(s_t, a_t; \theta)\right)^2\right] \tag{3.11}$$

d. Update Q-network. By using the gradient descent to minimize the loss and update the Q-network weights as following:

$$\theta \leftarrow \theta - \alpha \nabla \theta L(\theta) \tag{3.12}$$

Where α is the learning rate.

5. Update Target Network. The target Q-network updated to match with the Q-network.
6. Repeat the Process for the interaction with environmental, replay memory experience and Q-network updates for each of the time step until the agent converges or a stopping criterion.

The key hyperparameters in DQN as following:

1. Discount Factor, γ. To determine/identify the importance of future rewards.
2. Learning Rate, α. To determine/identify the weights of the network are adjusted during training.

3. Epsilon, ϵ. The probability of select exploration (random action) rather than exploitation (current policy)
4. Batch Size. The numbers of samples from replay memory fir each update step.
5. Replay Memory Size. The maximum number for experiences stored in memory.
6. Target Network Update Frequency. The frequency for the targeted network is updated to match with Q-network.

The formula of the Q-values update at each time step:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-) - Q(s_t, a_t)] \quad (3.13)$$

Where:

$Q(s_t, a_t)$ is the current Q-value

r_t is the reward received after action taken, a_t

γ is discount factor,

$\max_{a'} Q(s_{t+1}, a'; \theta^-)$ is the target Q-value form the target network

θ^- is the targeted weights of the neural network

After the training, the evaluation of DQN model required to check the performance of the DQN model. The evaluation will be including test the policy and performance metrics. Test the policy is to check the performance of agent on the learned policy at the test environmental. With the testing, will identify the valines of the policy. The cumulative reward over time as the performance metrics to evaluate the DQN model after training.

The Figure 3.3 showed architecture of DQN.

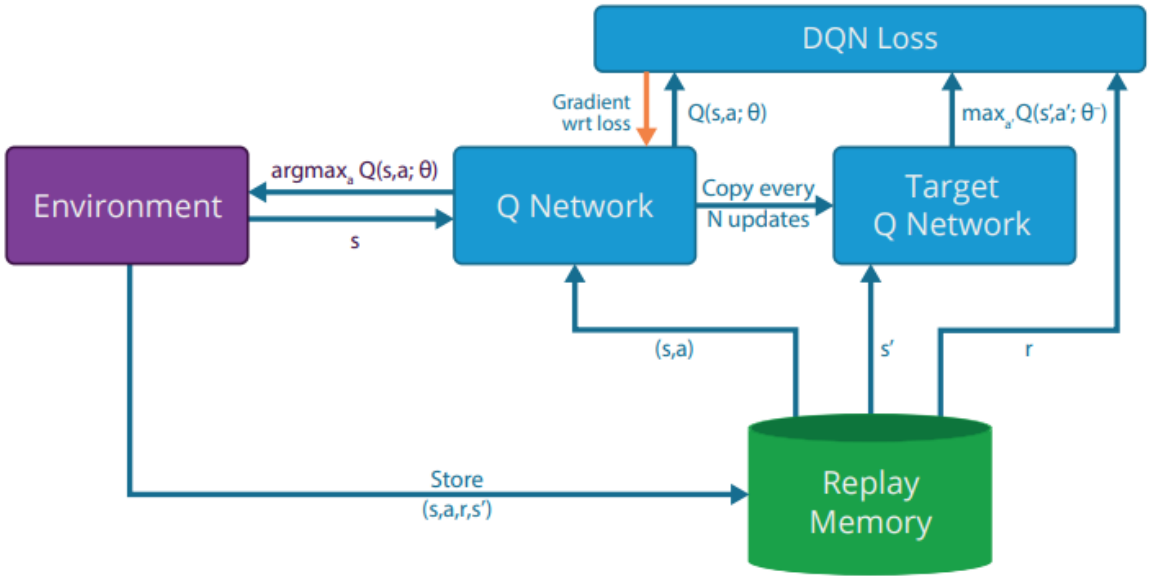

Figure 3.3 DQN architecture

3.4.2 Soft Actor-Critic (SAC)

SAC is a RL algorithm which primarily used in robotics field. The algorithm aims to find the optimal policy that can obtain that maximum expected long-term reward and long-term entropy. The Equation 3.1 stated below indicate the logic behind this algorithm.

$$J(\pi_\theta) = E_{\pi_\theta}[\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t) + \alpha H(\pi(.|s_t))] \tag{3.14}$$

The learning of the state-value function stated as Equation 3.15 is done through minimizing the squared residual error. Then it takes the first order of derivative of that state-value function to get the gradient shown as Equation 3.16.

$$J_V(\psi) = \mathbb{E}_{s_t \sim D}[\frac{1}{2}(V_\psi(S_t) - \mathbb{E}_{a_t \sim \pi_\phi}[Q_\theta(s_t, a_t) - \log \pi_\phi(a_t|s_t)])^2] \tag{3.15}$$

$$\hat{\nabla}_\psi J_V(\psi) = \hat{\nabla}_\psi V_\psi(s_t)(V_\psi(S_t) - Q_\theta(s_t, a_t) + \log \pi_\phi(a_t|s_t) \tag{3.16}$$

Then the learning of Q-function, which stands for quality function here is stated as Equation 3.17 below, and the next step Q-function is stated as Equation 3.18. The gradient for the learning can be obtained through Equation 3.19 below.

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t) \sim D}[\frac{1}{2}(Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t))^2] \tag{3.17}$$

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}[V_{\bar{\psi}}(s_{t+1})] \tag{3.18}$$

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(a_t, s_t)(Q_\theta(s_t, a_t) - r(s_t, a_t) + \gamma V_{\bar{\psi}}(s_{t+1})) \tag{3.19}$$

Then, the learning of the policy can be done through the Equation 3.20, and the gradient can be calculated through Equation 3.21.

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D, \epsilon_t \sim N}[\log \pi_\phi(f_\phi(\epsilon_t; s_t)|s_t) - Q_\theta(s_t, f_\phi(\epsilon_t; s_t))] \tag{3.20}$$

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \log \pi_\phi(a_t|s_t) + (\nabla_{a_t} \log \pi_\phi(a_t|s_t) - \nabla_{a_t} Q(s_t, a_t)) \nabla_\phi f_\phi(\epsilon_t; s_t) \tag{3.21}$$

The Figure 3.4 and Figure 3.5 showed the basic reinforcement learning for one time and architecture of actor critic.

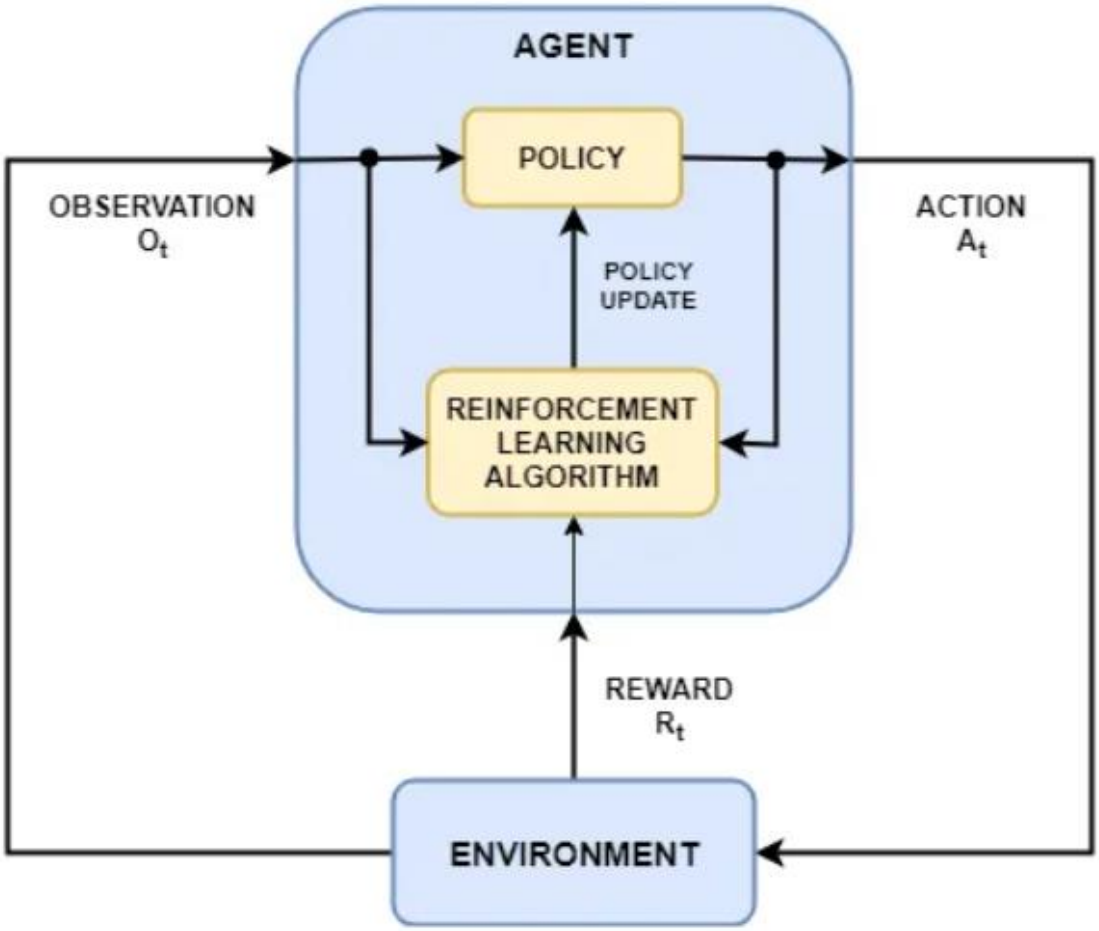

Figure 3.4: Basic reinforcement learning structure.

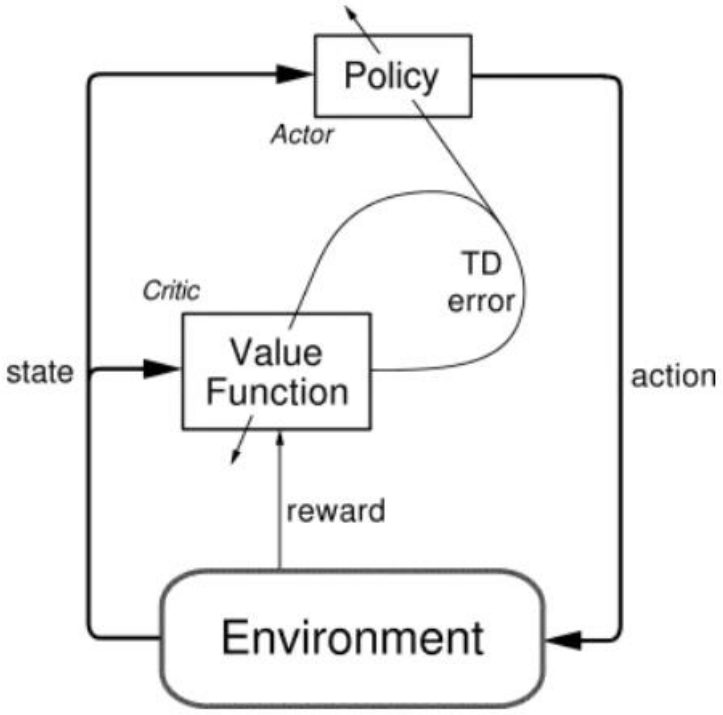

Figure 3.5: Soft-Critic architecture.

3.4.3 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is the DRL model with the policy gradient method that have developed by OpenAI. PPO able to prevent the large policy updates which is balance the exploitation and exploration. For the algorithm of PPO is actor-critic types which is maintaining actor(policy) and critic(value) in the same times. Actor is the policy network which is the action taken by given a state. Critic is value network which is estimate the value of the state. PPO uses a surrogate objective function that will avoid the large changes in policy to make the stability of learning process.

The Clipped surrogate objective as following:

$$L^{CLIP}(\theta) = E_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t] \quad (3.22)$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta-\text{old}}(a_t|s_t)} \tag{3.23}$$

Where:

$\pi_{\theta-\text{old}}$ is the old policy before update

π_θ is the policy parameterized by θ

ϵ is the small hyperparameter to limit updates

A_t is the advantage function at time step, t

$r_t(\theta)$ is the probability ratio

Total loss function as following:

$$L(\theta) = E_t[L^{CLIP}(\theta) - c_1 . (V_\theta(s_t) - R_t)^2 + c_2 . S[\pi_\theta](s_t)] \tag{3.24}$$

Where:

$V_\theta(s_t)$ is the estimated value function

R_t is the cumulative return

$S[\pi_\theta]$ is the entropy bonus for exploration

c_1 , c_2 is the coefficients to balance loss terms

The methodology for PPO algorithm will be as following:

1. Environmental setup
    a. Identify the state space and action space
2. Initialize Neural network
    a. Initialize the actor and critic networks with random weights
    b. Separate the network parameters depending on design
3. Data collection
    a. Collect N at the time steps for the data by run the policy in environment
    b. The transition store same as DQN (s_t, a_t, r_t, s_{t+1})
4. Advantage estimation
    a. By using generalized advantage estimation (GAE) to calculate the A_t

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots\ldots \quad (3.25)$$

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \quad (3.26)$$

5. Policy and value update
    a. Calculate the surrogate loss by using clipped objective
    b. Update the policy and value network by using the stochastic gradient descent
    c. Repeat the several epochs over mini-batches of collected data
6. Repeat process
    a. Continue repeat the step 1-5 to achieve the performance threshold.

The hyperparameters for the PPO including:

- Discount Factor, γ. To determine/identify the importance of future rewards.
- Learning Rate, α. To determine/identify the weights of the network are adjusted during training.

- Clipping parameter, ϵ. The probability of select exploration (random action) rather than exploitation (current policy)
- Batch Size. The numbers of samples from replay memory fir each update step.
- GAE parameter, λ. To control weighting of different multi-step estimates

Figure 3.6 shown the architecture of PPO.

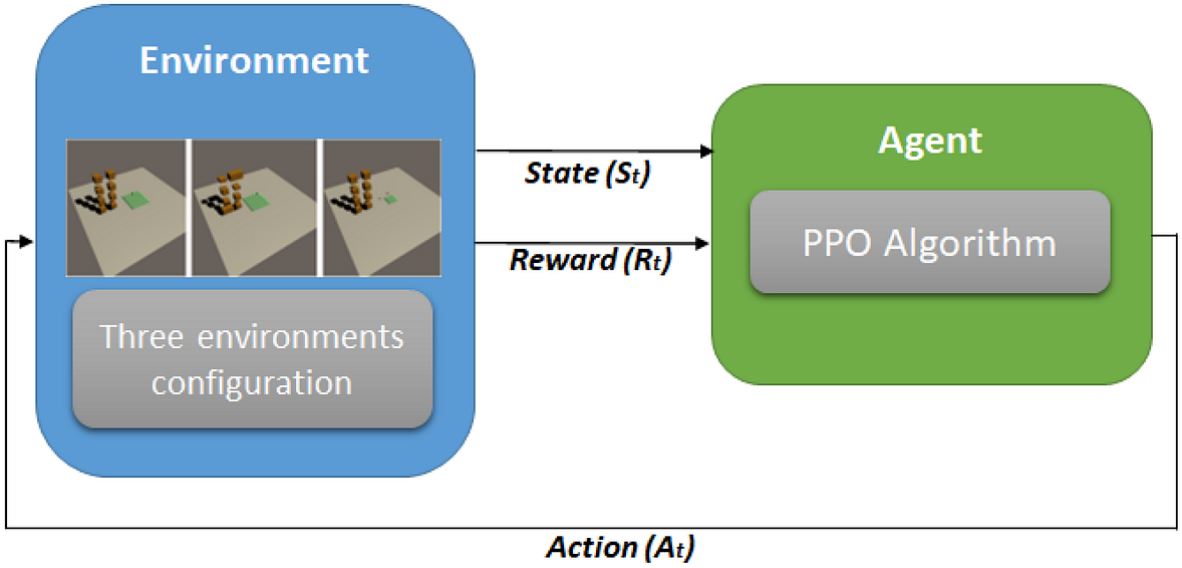

Figure 3.6: PPO architecture.

3.5 Model Training

3.5.1 Feature Engineering

The feature engineering will apply after data preprocessing completed. The feature engineering including as following:

- Raw features – historical price data (OHLC-Open, High, Low, Close) and volume (daily trading record)
- Time-based Features – Time of the day

- Normalization – the features will normalize to 0 to 1 to avoid the hinder model of learning.
- Technical indicators are moving averages (simple Moving Average (SMA) and Exponential Moving Average (EMA) and RSI

3.5.2 Data Splitting

The dataset will split to 3 session which is training data, validation data and test data. The percentage distribution of the dataset will be 70% for training data set, 15% for validation data set, 15% for test data set. Training data is the 70% of available data for train model. Validation data is for hyperparameter tuning. Test data is to evaluate the model generalization after training and validation.

3.5.3 Deep Reinforcement Learning (DRL) Model Training Loop

The DRL model training loop can divide as following:

I. Environmental Initialization is the state space initialize as the environmental from the input data.
II. Action selection
    a. DQN – By using Q-network to approximate the Q-values, model will take the action based on the Q-values. The Q-values is the expected future reward for each time step in a given state. The model follows the epsilon-greedy strategy which is exploration in random action and exploitation is chose the best action.
    b. PPO – The model the select the action based on the policy network. The action taken is sampled from the probability distribution which is the updated policy based on the advantage function. PPO will using the clipped objective function to enable the stability of training process and avoid the large changes in policy.
    c. SAC – The model will select the action based on the sampling from the policy distribution and the impact of entropy to encourage exploration. The policy updated to optimize the trade-off between expected reward and entropy.

III. Executing the action is the model will perform the action based on the selection at II which is buy, sell or hold.

IV. Reward Calculation. The reward will be calculated based on the profit or loss of the action taken.

V. Next State Observation: The new market will be observed after the executing action (step III)

VI. Experience Replay (for DQN only): The experience replay buffer to store states, actions, reward and next state. The random sample of the experiences from the buffer is used to train the Q-network.

VII. Model Update

   a. DQN. By apply the Bellman equation to adjust the Q-values to minimize the difference between predicted and actual rewards.
   b. PPO. By optimizing the clipped objective function where to balance the current and new policy to avoid large updates.
   c. SAC. By using soft bellman backup equation with the entropy regularization to encourage exploration for the policy network and values network.

VIII. Episode Completion. After passed the predefined number of the time steps, will be terminated to allow the model proceeds to the next time steps.

IX. Repeat process. The process will be repeated in multiple time steps to improve the policy of model and reward received based on the action.

3.5.4 Bias-variance trade off

The bias-variance trade off will be consider during the model training. The high-bias may fail to capture the market trend (dynamic) which is the underfitting of the data and lead to poor decision making. The model with high bias because of the architecture too simple or the training data unable to provide sufficient features. A high variance model is overfitting of the data and leading to overperformance in training but poor in generalization of unseen data. The overfitting because of model is too complex. Figure 3.7 shown the bias-variance trade off.

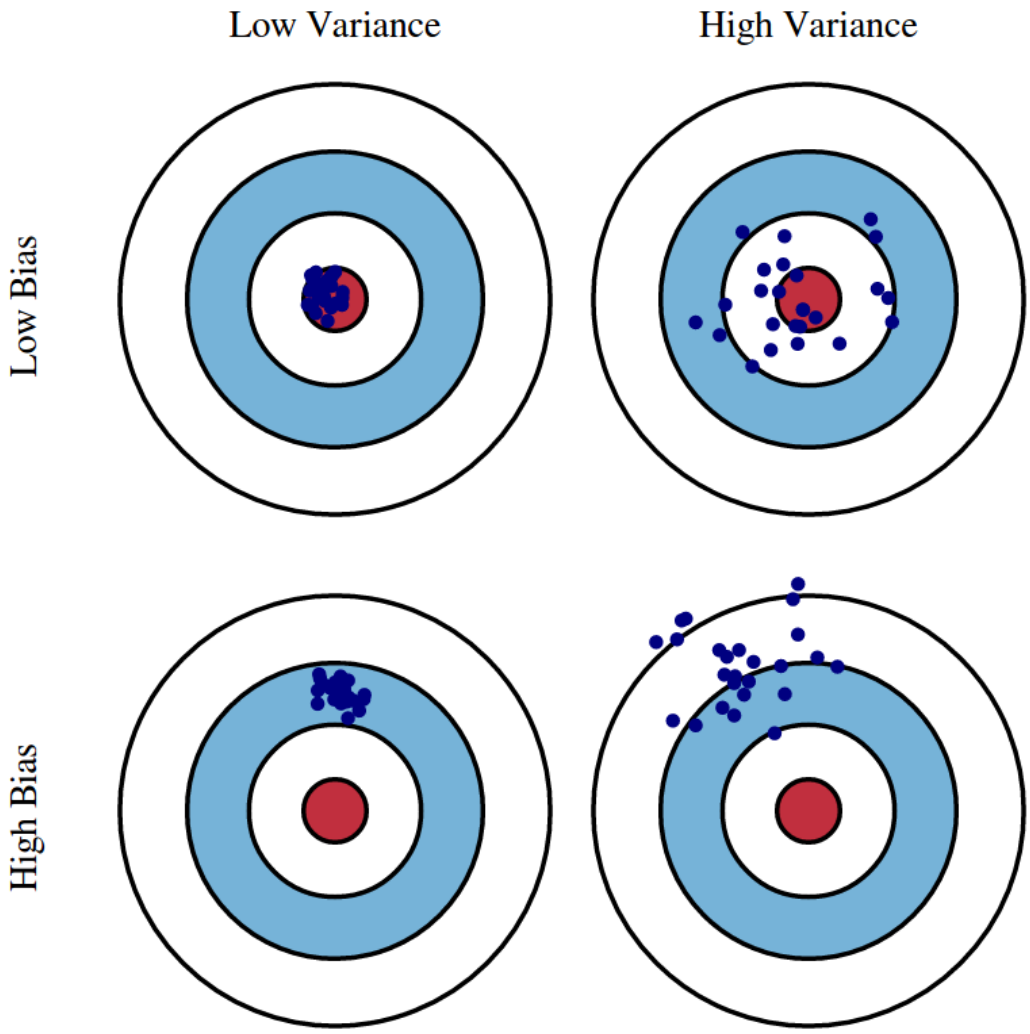

Figure 3.7 Summary of the bias-variance trade off

The methods of handling bias-variance trade off is regularization techniques which is the weight decay and using to mitigate high variance issues. Early stopping based on the performance in the validation set will help to avoid overfitting issues. Exploration strategy will ensure the model not too greedy and exploitative too early, it will helps to maintain to balance between bias and variance.

3.6 Hyperparameter Tuning

The general hyperparameter for DQN, PPO and SAC as following:

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

The model-specific hyperparameters of each of the DRL model as following:

1. DQN.
    a. Target Network Update Frequency – Control the target Q-network is updated with the parameter of main Q-network
    b. Double Q-Learning – help to mitigate the overestimation bias in Q-values
2. PPO.
    a. Clip Range – Prevent large changes in policy updates. It will help to increase the stability of the training process
    b. GAE lambda – generalized advantage estimation (GAE) to balance the bias and variance in estimation.
3. SAC
    a. Entropy Coefficient – Control the trade-off of exploration and exploitation to adding entropy regularization to objective function. It will help to increase the exploration.
    b. Target Entropy – To determine the exploration of the model during training.

The hyperparameter tuning methods can be divided into Grid search, random search, Bayesian optimization and cross-validation. The grid search is the systematically tries all the possible combination of the hyperparameter values within the predefined range. The grid search method

is simple to implementation but computationally expensive due to exhaustive nature of the search. The step of the grid search as following:

1. Define the range for each hyperparameter (learning rate, discount factor and batch size)
2. Create the grid of all possible combination of hyperparameter values.
3. Train the model for all the possible combination of hyperparameter and evaluate the performance on the validation data by using the metrics such as total reward, Sharpe ratio and maximum drawdown.
4. Select the best performance of the combination for the hyperparameter on the validation set.
5. Select the combination with fine-tune.

Random search is select the combination of hyperparameter randomly from the search state. The cost of the random search is lower than grid search but not guarantee can get the optimal hyperparameter data set. The step for random search as following:

1. Define the range for each hyperparameter (learning rate, discount factor and batch size)
2. Search the sample hyperparameter values randomly.
3. Train the model based on the random hyperparameter combinations and evaluate the performance of the model on the validation data set.
4. Select the best performance of the combination hyperparameter based on the validation set.

Cross-validation is the generalization of model ability and avoid the overfitting during hyperparameter tuning. The data set will split into k-folds and the model can be train and evaluate on each of the fold. The methods able to provide the better estimate of the model's generalization ability but the higher cost of computational due to multiple training runs. The step of the cross-validation as following:

1. Split the training data into k folds
2. Train the model by using k-1 folds and validate for each of the fold.
3. Average the performance across all folds
4. Evaluation criterion for selecting hyperparameter by using the average performance

3.7 Evaluation Metrics/Performance Measurement

Evaluation metrics is to assess the performance of DRL models (DQN, SAC and PPO). For this research, focus on the two critical performance measurement which is F1 score and cumulative return. F1 score is to measure the balance between precision and recall which can adapted for evaluate the decision of trading. Cumulative return is the measure the total return rate of the trading strategy making.

3.7.1 Accuracy Benchmark (F1-score & Cumulative return)

F1 score is used to evaluate the classification model for precision and recall phase. It will evaluate the performance of model in predicting buy and sell decision where the target with the minimum of false positive and false negatives.

$$\text{F1 Score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.27}$$

Where:

Precision measures how many of the predicted buy or sell action

Recall measures how many of the actual profitable buy or sell opportunity of the model.

Higher F1 score means that the precision and recall with the higher numbers of true negative or positive. Low F1 score means either the model has many false positive or negative, lead to poor trading decision.

Cumulative return is the financial metric which measures the total return of the trading strategy over a given period. The formula of cumulative return as following:

$$\text{Cumulative Return} = \frac{Final\ Portfolio\ value - Initial\ Portfolio\ value}{Initial\ Portfolio\ value} \quad (3.28)$$

Where:

Initial portfolio value is the starting value of the trading portfolio

Final portfolio value is the ending value of the portfolio

3.8 Dashboard Development

After the completion of the evaluation metrics/performance measures, dashboard development will be required to have the better visualization. Dashboard is to monitor the performance of an automated trading system to enhance the output of the DRL models is more visualise. The purpose of the dashboard is to provide a user-friendly interface that display metrics such as cumulative return, F1 score, trade history and portfolio performance for the each of the DRL model which is DQN, SAC and PPO. It will help to get the real-time agent performance updates.

3.9 Model Comparison and Performance Analysis

After the dashboard developed, the performance comparison between DQN, SAC and PPO will be based on the cumulative return, F1-score and model stability and robustness over time. With the comparison between DQN, SCA and PPO, will help to identify the strengths and weaknesses of each model.