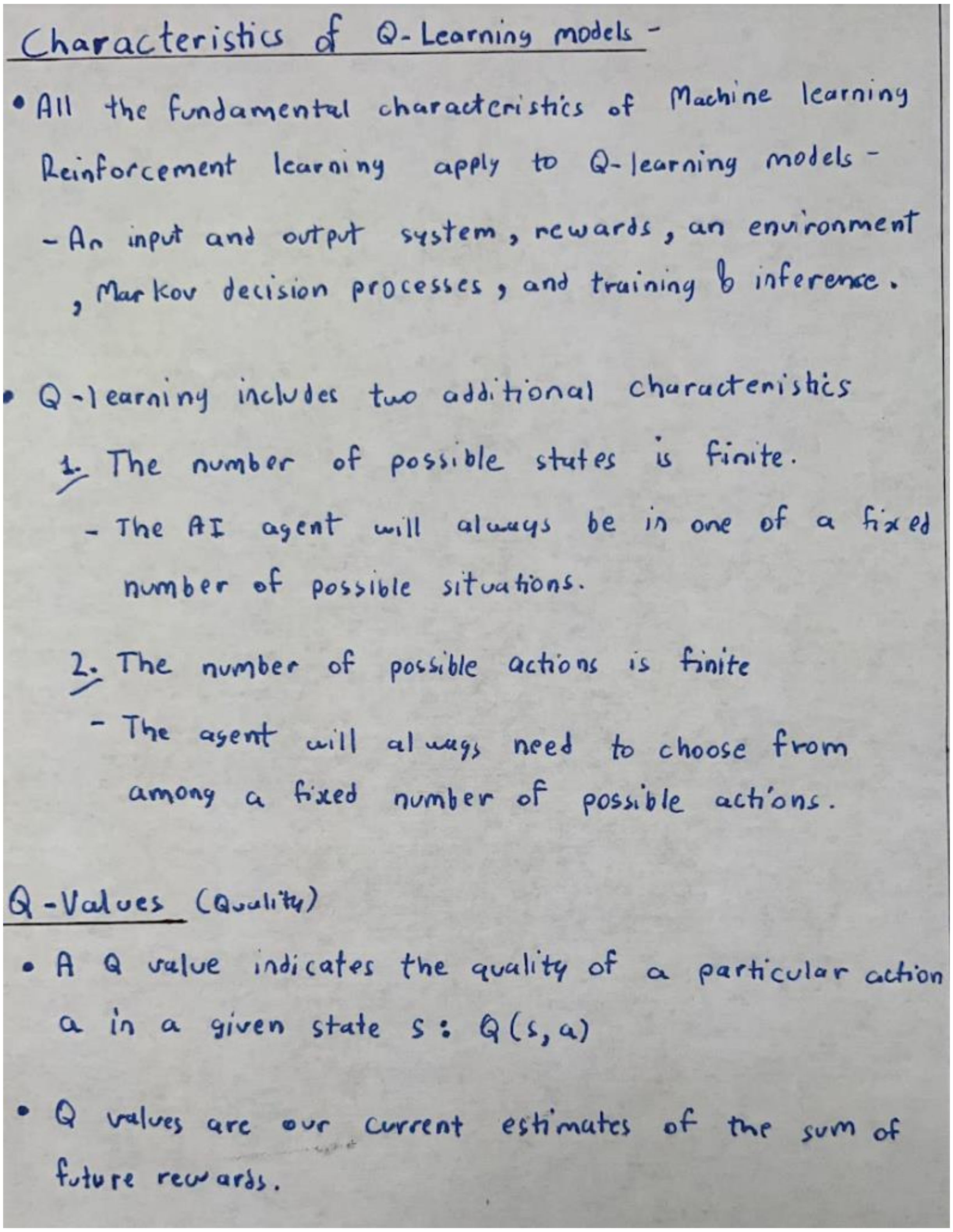
ML Project 3

Task

The goal of the assignment is to learn the trends in stock price and perform a series of trades over a period of time and end with a profit. In each trade you can either buy/sell/hold. You will start with an investment capital of $100,000 and your performance is measured as a percentage of the return on investment.

You will use the Q-Learning algorithm for reinforcement learning to train an agent to learn the trends in stock price and perform a series of trades. You will implement Q-learning algorithm from scratch. The purpose of this assignment is to understand the benefits of using reinforcement learning to solve the real world problem of stock trading.

Q learning Models:



Characteristics of Q-Learning models -

- All the fundamental characteristics of Machine learning Reinforcement learning apply to Q-learning models -
  - An input and output system, rewards, an environment, Markov decision processes, and training & inference.

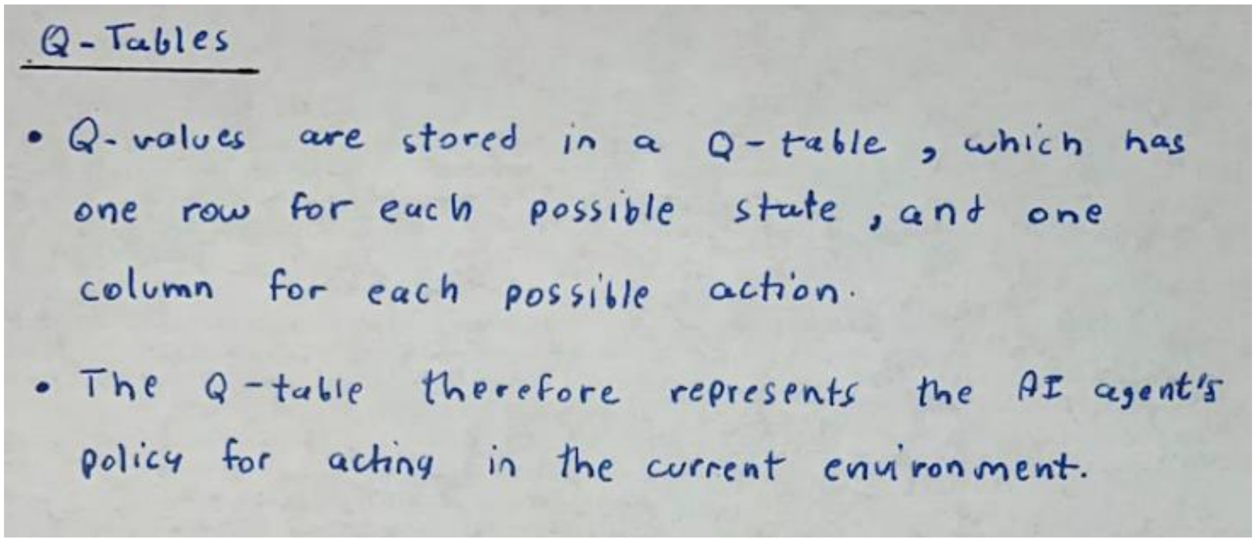
- Q-learning includes two additional characteristics
  1. The number of possible states is finite.
     - The AI agent will always be in one of a fixed number of possible situations.
  2. The number of possible actions is finite
     - The agent will always need to choose from among a fixed number of possible actions.

Q-Values (Quality)

- A Q value indicates the quality of a particular action a in a given state s: $Q(s, a)$

- Q values are our current estimates of the sum of future rewards.

Q Tables:



Q-Tables

- Q-values are stored in a Q-table, which has one row for each possible state, and one column for each possible action.
- The Q-table therefore represents the AI agent's policy for acting in the current environment.

Environment:

We have been given a dataset on the historical stock price for Nvidia for the last 5 years. The dataset has 1258 entries starting 10/27/2016 to 10/26/2021. The features include information such as the price at which the stock opened, the intraday high and low, the price at which the stock closed, the adjusted closing price and the volume of shares traded for the day.

We are also given an environment which gives us the observation, reward and Boolean describing whether the episode has ended.

There are 4 states 3 actions, so our Q table will be a 4*3 matrix.

The 3 actions are BUY, HOLD and SELL.

The four states/observations are integer in range 0 to 3 representing the four possible observations that the agent can receive. The observation depends upon whether the price increased or decreased on average in the number of days the agent considers, and whether the agent already has the stock or not.

The four observations represent the following scenarios:

0. Price of the stock has increased, and the agent doesn't hold any shares.

1. Price of the stock has increased, and the agent holds shares.

2. Price of the stock has decreased, and the agent doesn't hold any shares.

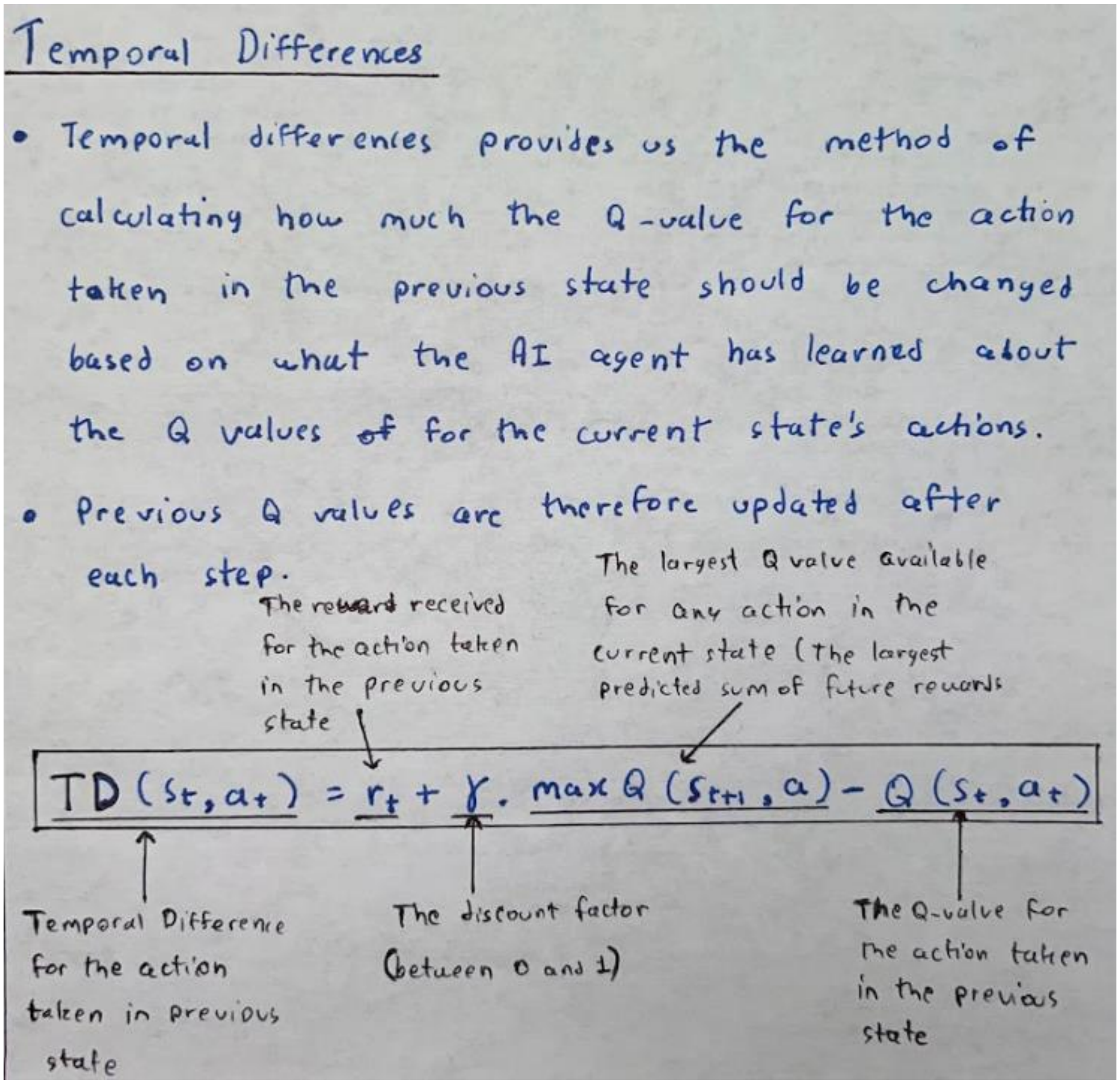
3. Price of the stock has decreased, and the agent holds shares.

The goal of our agent is to increase the total account value.

Implementing Q learning:

I have implemented the Q learning algorithm and running it for 320 episodes to train the model.

For this I have calculated the Temporal differences as explained below:



Temporal Differences

- Temporal differences provides us the method of calculating how much the Q-value for the action taken in the previous state should be changed based on what the AI agent has learned about the Q values ~~of~~ for the current state's actions.
- Previous Q values are therefore updated after each step.

The reward received for the action taken in the previous state

The largest Q value available for any action in the current state (The largest predicted sum of future rewards

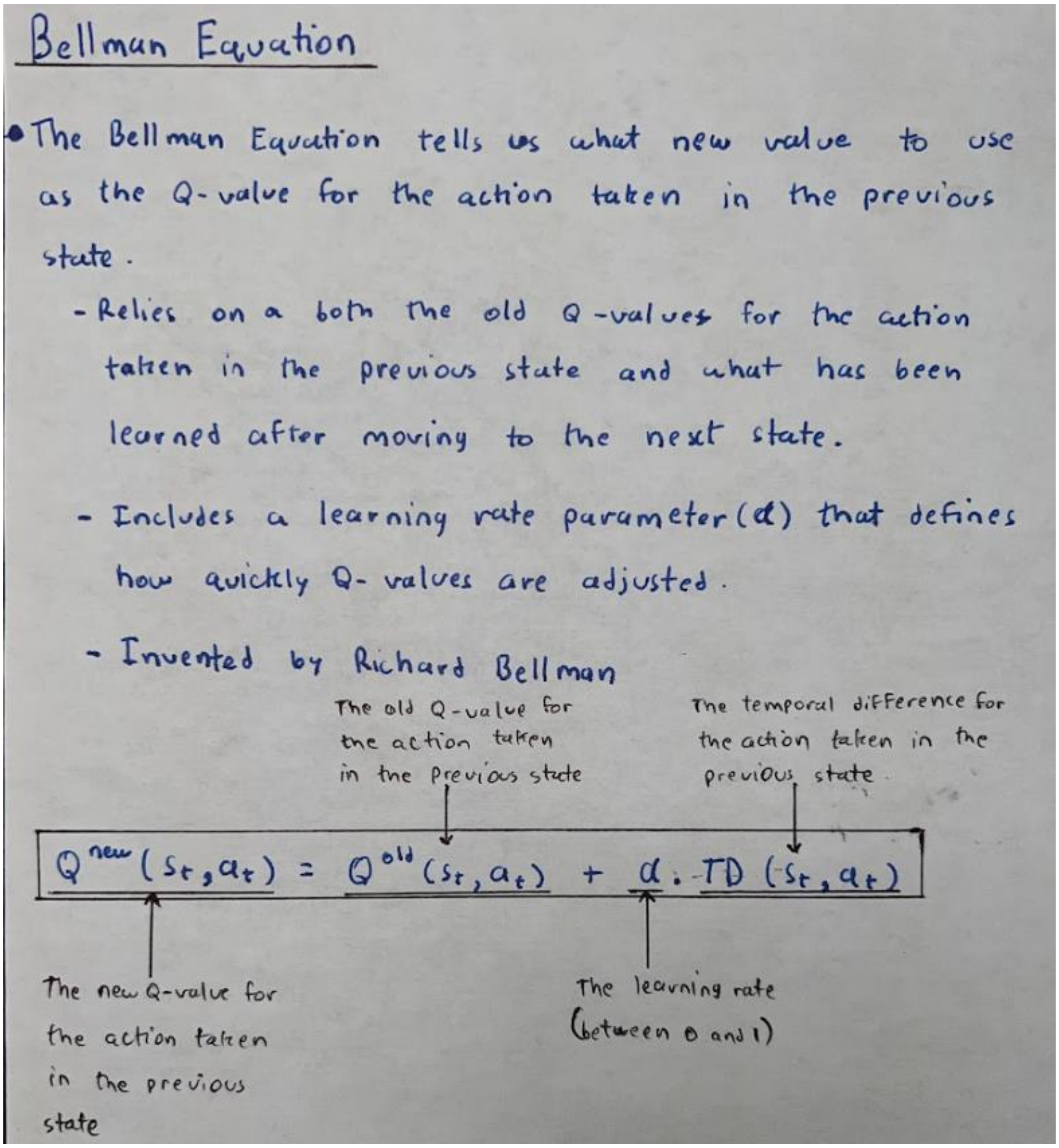
$$\boxed{TD(s_t, a_t) = r_t + \gamma \cdot \max Q(s_{t+1}, a) - Q(s_t, a_t)}$$

Temporal Difference for the action taken in previous state

The discount factor (between 0 and 1)

The Q-value for the action taken in the previous state

Then I am using the Bellman Equation to calculate the new Q values as explained below:

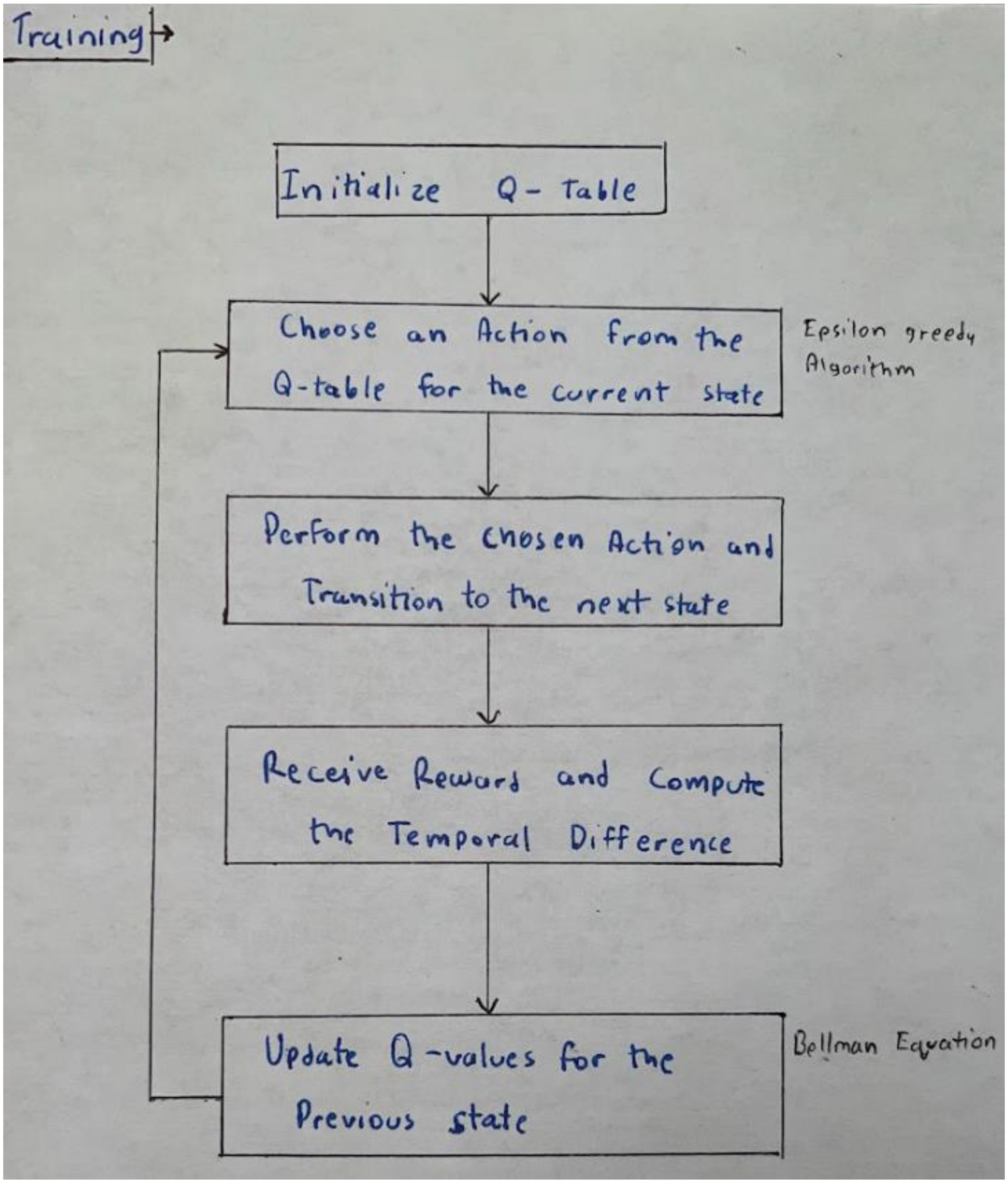


Bellman Equation

- The Bellman Equation tells us what new value to use as the Q-value for the action taken in the previous state.
  - Relies on a both the old Q-values for the action taken in the previous state and what has been learned after moving to the next state.
  - Includes a learning rate parameter (α) that defines how quickly Q-values are adjusted.
  - Invented by Richard Bellman

$$\boxed{\underbrace{Q^{new}(s_t, a_t)}_{\text{The new Q-value for the action taken in the previous state}} = \overbrace{Q^{old}(s_t, a_t)}^{\text{The old Q-value for the action taken in the previous state}} + \underbrace{\alpha}_{\text{The learning rate (between 0 and 1)}} \cdot \overbrace{TD(s_t, a_t)}^{\text{The temporal difference for the action taken in the previous state.}}}$$

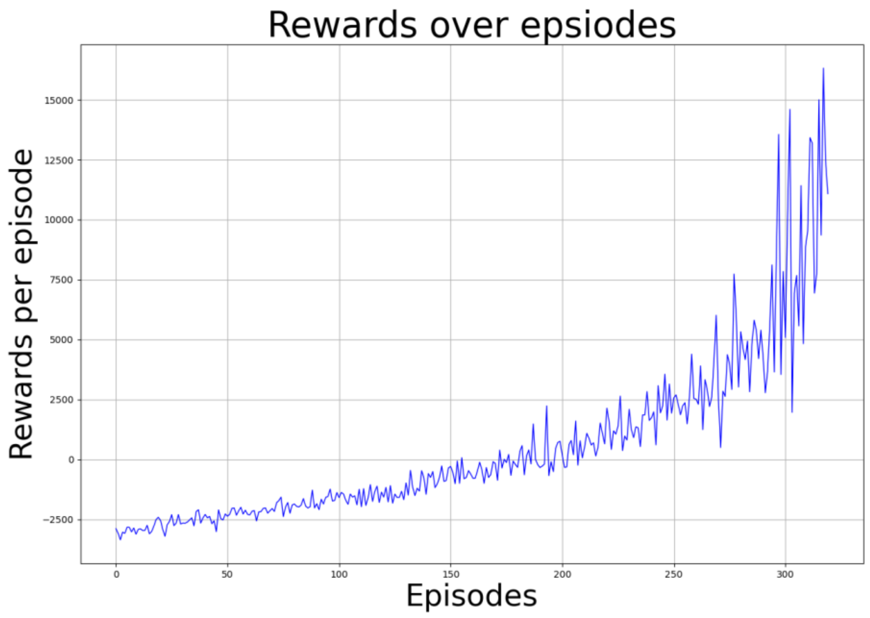
Training the model:

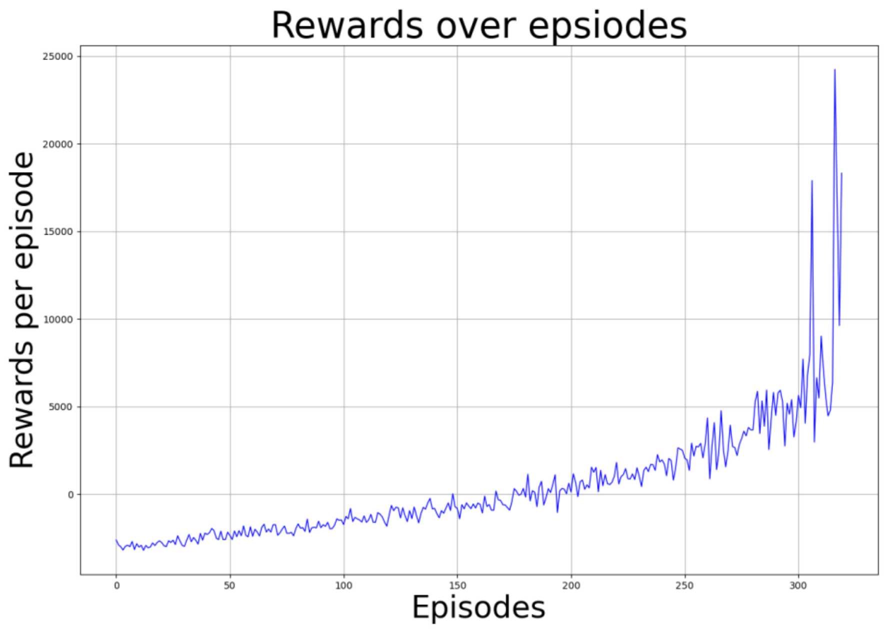

Training
Initialize Q-Table
Choose an Action from the Q-table for the current state
Epsilon greedy Algorithm
Perform the chosen Action and Transition to the next state
Receive Reward and Compute the Temporal Difference
Update Q-values for the Previous state
Bellman Equation

After training I am then evaluating the model and plotting the epsilon decay and total rewards per episode:

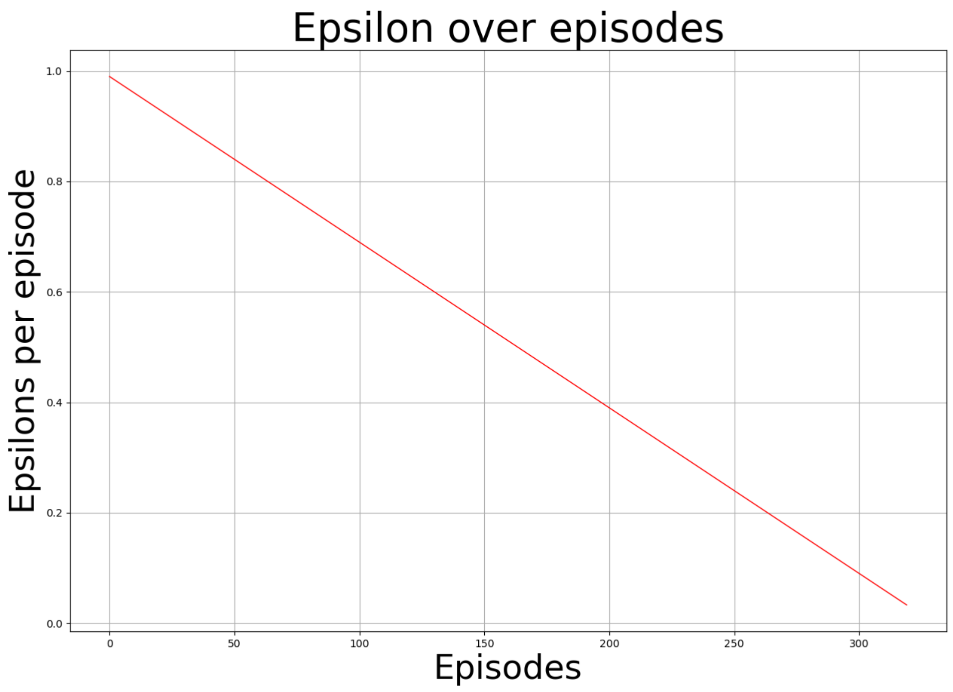
(Training and running the evaluation multiple times to observe the graphs)

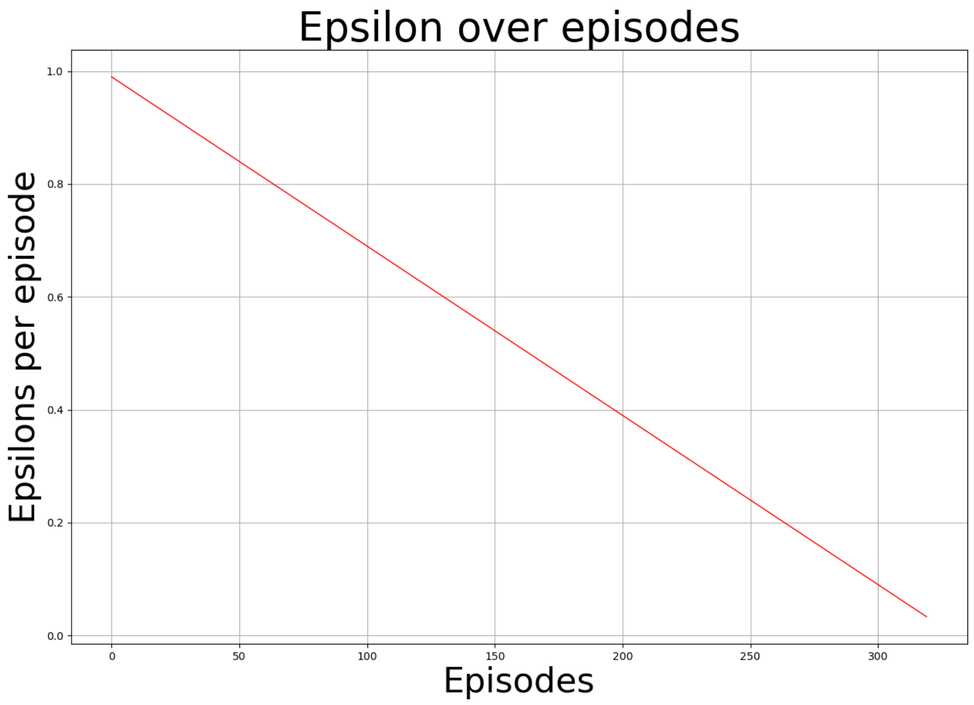
Rewards over Episodes:



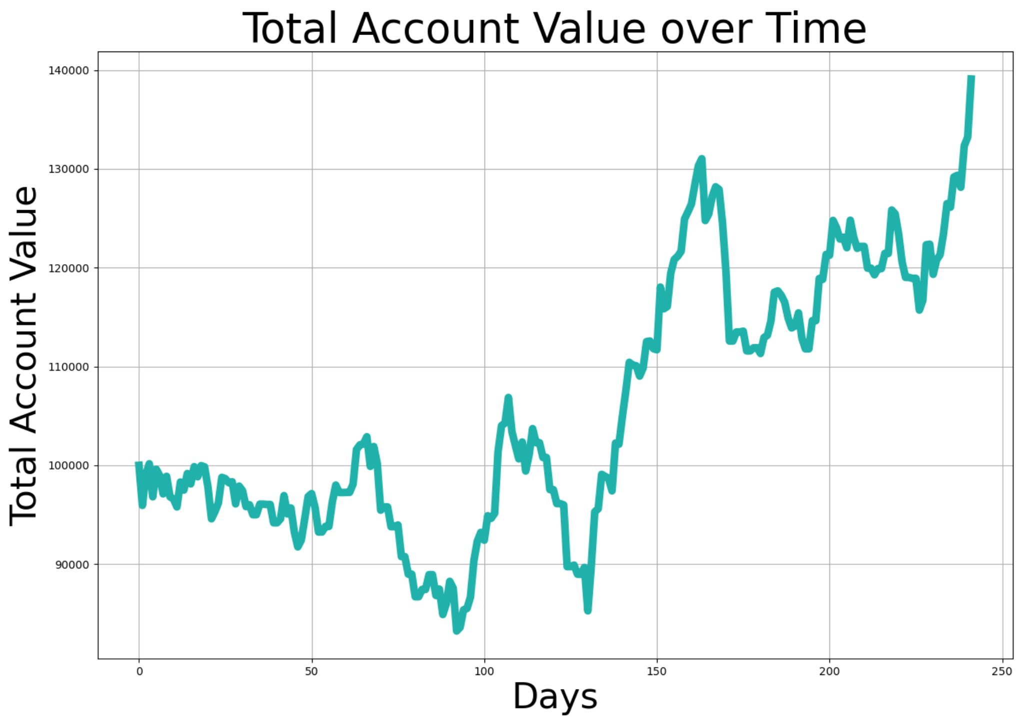


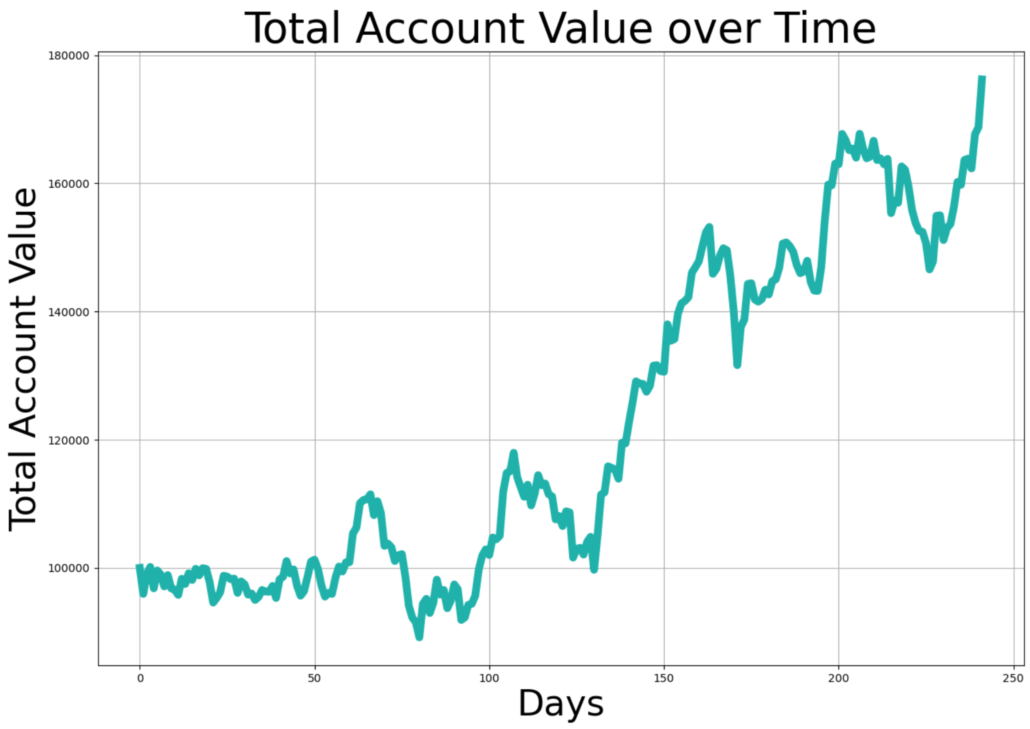
Epsilon over Episodes:





Total Account Value over Time:





As we can observe above the agent is giving variable account values over time as per the training variables between the range of 140000 to 180000.