L-Università ta' Malta
Faculty of Information & Communication Technology
Department of Computer Information Systems

Applying Reinforcement Learning to Blackjack

Liam Bugeja Douglas 17002L
Matthew Calafato 31102L
Aidan Seychell 165702L

Study-unit: **Reinforcement Learning**
Code: **ARI2204**
Lecturer: **Dr Josef Bajada**

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as “the unacknowledged use, as one's own, of work of another person, whether or not such work has been published, and as may be further elaborated in Faculty or University guidelines" (University Assessment Regulations, 2009, Regulation 39 (b)(i), University of Malta).

We, the undersigned, declare that the assignment submitted is our work, except where acknowledged and referenced.

We understand that the penalties for committing a breach of the regulations include loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected and will be given zero marks.

* Delete as appropriate.

(N. B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

Liam Bugeja Douglas	
Student Name	Signature
Matthew Calafato	
Student Name	Signature
Aidan Seychell	
Student Name	Signature

ARI2204	**Applying Reinforcement Learning to Blackjack**
Course Code	Title of work submitted

16/05/2022

Date

Contents

1. Overview

1.1 Rules

Blackjack is a game which is played with a standard 52-card deck and can contain one or multiple decks, especially in casinos. No matter the variation of the game, the aim is always the same: achieve a higher score than the dealer without exceeding a total sum of 21. If a player goes over the limit they automatically lose. Vice versa, if the dealer goes over 21, the player wins automatically.

1.2 Card Values

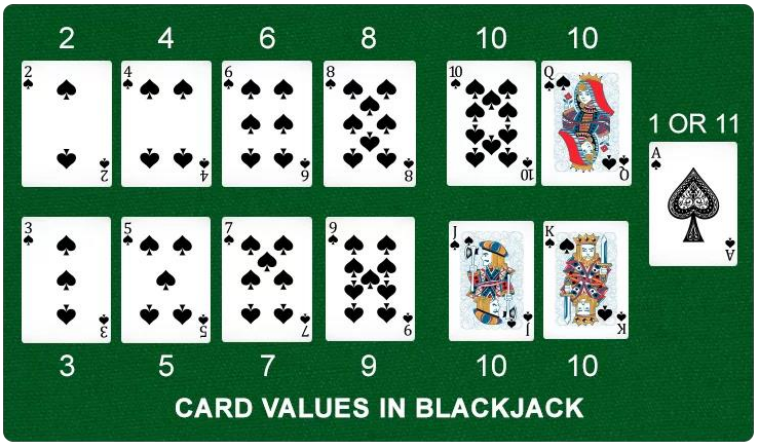


Figure 1: Card Values in Blackjack [1]

The image above shows the card values in blackjack, the numerical cards 2 to 10 all have their face value. Jacks, Queens and Kings have a value of 10. Aces have value of 11 but the player can choose to change its value as 1.

1.3 Play through

Each player is handed two cards face up whilst the dealer gets two cards with only one facing up. The dealer is always the last one to play, and after each player in the table is finished the dealer shows his second card. Once a player has two cards, they can either hit (draw more cards) or stand (draw no more cards) until they reach 21, however if this is exceeded the

player automatically loses. When it is the dealers turn, they play normally with the exception of hitting when under 16 and standing after 17.

At the end of the round, if the total sum of a player's cards is higher than that of the dealer the player wins, if it is lower the dealer wins. If it is the same, then it is a standoff (draw). In the event the first two cards of a player are an Ace and any 10-value card, it is an automatic win for the player (the hand is called a blackjack).[2]

2. Winning Strategy

2.1 Statistics

Ignoring ties, the percentage of times you are expected to win is 48% and lose 52%, this means that in the long run-on average most people lose money playing blackjack. When players use certain strategies such as card counting in casinos, they approximately have an edge of 1% and sometimes even lower compared to other players in the table.

The dealer has a higher win rate for one reason only, he or she always plays last, so if a player busts (goes over 21) the dealer automatically wins.[3]

1.2 Reinforcement Learning

Blackjack has been closely studied to improve the win rate for the player, when applying reinforcement learning in blackjack the agent has had a higher win rate than a random play. Also, when taking in consideration the dealer's card facing up, the agent had a slightly higher win rate.

In this project we will be using the following algorithms to try to improve the win rate of the agent: Monte Carlo On-Policy Control, SARSA On-Policy Control and Q-Learning Off-Policy Control. Each algorithm will have 4 different variations by using different ϵ-Greedy policies.

3. Environment Setup

For the reinforcement learning models to learn, a number of games need to be played. Each game is called an episode. Each episode is given a number *k* which denotes the number of episodes the script has run so far and increments with each episode.

Each episode consists of the following:

3.1 Deck of Cards

The card deck is created using a function createDeck() which returns a list of characters representing the different cards. This list is shuffled using the shuffle function found in the random library. This randomly shuffles the deck.

3.2 Hands

The agent and dealer's hands are made up of two lists (*playerHand, dealerHand*). At the start of the episode, two cards per hand are popped from the deck list. The first card of the dealer's hand is chosen as the visible card.

3.3 Agent Policy

The agent's policy is to always hit until reaching a total value of 12, this was done by hitting while the total sum is less than 12. After that between 12 and 20, 3 different policies can be used (Monte Carlo, SARSA, Q-Learning). These will be explained in further detail later on. Finally, if the agent has a sum of 21 it must always stand.

3.4 Dealer Policy

After the agent stops playing, the dealer invokes his policy. Hit until the total sum of the cards reaches at least 17, even if the agent has a higher sum. This was done by hitting until the sum is less than 17.

3.5 State Action Values

The point of the three Policies we used in this project (Monte Carlo, SARSA and Q-Learning) is to build a strategy sheet (shown in figure 2). This is like a table which, when taught, presents the best action to take in each possible state.

	2	3	4	5	6	7	8	9	T	A
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
17	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
12	H	H	S	S	S	H	H	H	H	H
11	D	D	D	D	D	D	D	D	D	D
10	D	D	D	D	D	D	D	D	H	H
9	H	D	D	D	D	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H
7	H	H	H	H	H	H	H	H	H	H
6	H	H	H	H	H	H	H	H	H	H
5	H	H	H	H	H	H	H	H	H	H

Figure 2: Strategy Sheet for Blackjack [4]

This table starts empty, and with each episode, depending on what actions were taken and the reward of the episode, each state starts modifying its value. After a substantial number of episodes, the values should converge and won't change a lot, which means an optimal table was built.

In our case, the table starts from a sum of 12 till 20. Each cell was represented as a dictionary with 4 attributes; hit value, hit count, stand value and stand count. The counts store the number of time that state action was performed, and the values symbolize which of the two actions to take.

4. Monte Carlo On-Policy Control

Unlike other algorithms, Monte Carlo methods do not require the full knowledge of the environment. They only need the history of the environment, which includes data such as the states, actions and rewards. Monte Carlo has two different policies on-policy and off-policy, in this project we mainly focused on the on-policy control. The main difference between the two is that on-policy methods improve the policy for decision making whilst off-policy methods improve the policy to generate better data.

Since the algorithm has a strict policy some state action values will never be explored, therefore not all possible actions will be given a chance. To counteract these one of the four different e-greedy policies starts by giving a random chance within the first action of the agent, thus giving the possibility to generate more episodes.[5][6]

The state action values are modified as follows:

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$$

Where:

$Q(S_t,\ A_t)$ is one of the cumulative states

$\frac{1}{N(S_t,\ A_t)}$ is the counter of that state action value

G_t is the reward

4.1 Implementation

For this project, we had 2 main implementations of the Monte Carlo approach. The exploring start, which was mentioned earlier and the non-exploring start. The only difference between the two is, when choosing the first action, the exploring start chooses randomly while the non-exploring start goes directly to the policy.

For each episode, the following process takes place after the environment is set up;

Firstly, if the agent's hand sums to 21, the agent automatically wins.

A list called *visitedSAV* is created to hold the actions done during an episode. This is than used at the end to modify the values of the state actions according to the reward of the game

A function was written to choose a random action from a particular state action. A random Boolean value is being created, which determines if the action will be a hit or a stand. The count of the appropriate state action is iterated, and the state action itself is stored inside the *visitedSAV* list. If the action was a stand, the function returns *true* otherwise, meaning hit, the hand is updated by popping another value from the deck and appended to the agent's hand and a *false* value is returned.

For the Monte Carlo itself, given it's an exploring start, the random function is called. If a stand action was done, the *visitedSAV* is returned from the Monte Carlo method. If the first action resulted in a hit, the policy is invoked.

For the policy, a loop was used until a stand action was taken or the sum of the cards exceeded 21. Firstly, a random number between 0 and 1 was created. If this number generated is smaller than the epsilon value, than a random action is taken. Otherwise, a condition finds out which of the two actions has a higher value in that particular state action, and executes that action. If both actions have the same value, a random action is taken. Given the action taken was not a stand, which means exiting the method and returning the *visitedSAV*, the hand is updated and the state action that was just done is updated in the *visitedSAV*.

After the Monte Carlo method exists, a reward is assigned based on the agent's and the dealer's cards. Based on that reward, the state action values that were taken in that episode (saved in the *visitedSAV* list) are modified with the formula mentioned above.

5. SARSA On-Policy Control

State Action Reward State Action (SARSA), is a Temporal-Difference Learning algorithm which means it is a model free method. Unlike Monte Carlo, SARSA updates the state action values upon transitioning from one state to another. [7]

To modify a state action value, the following function is used:

$$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A))$$

Where:

$Q(S, A)$: is the current state action value

α: is the step-size function

R: is the reward

γ: is the discount factor

$Q(S', A')$: is the next state action value

1.1 Implementation

Similarly, to the Monte Carlo implementation, a random method is used whenever a random action needs to be taken.

Since the state actions are updated during the process not at the end, 2 variables were created which hold the state action that was just performed, and another which holds the past state action. This is then used to modify the past state action with the above formula.

A loop takes place until the sum of the cards is over 21. Within the loop, the same epsilon greedy policy as the Monte Carlo takes place. The only difference is, that when an action takes place, given its not the first time going through the loop, the state action value is modified with a reward of 0. When a terminal state is reached (a stand or a hit which resulted in exceeding or equal to 21), a reward is calculated, and the state action value is modified with that reward, taking $Q(S, A)$ and $Q(S', A')$ as the same.

6. Q-Learning (SARSAMAX) Off-Policy Control

Q-Learning also known as SARSAMAX is a model free algorithm which seeks to find the best action given the current state action the agent is in. In other words, it is a method that tries to maximize the reward outcome. Q-Learning is very similar to SARSA, the only difference being that the $Q(S', A)$ is chosen as the best successor possible of the current state as shown below, denoted as *max*. [8]

$$Q(s,a) = r(s,a) + \gamma \max_{a} Q(s',a)$$

1.1 Implementation

A function was used to find the $maxQ\ (S', A)$. It rakes the current state we are in as parameter. According to the current state, it finds if the hit value or the stand value is larger and returns the state action value which is largest. If they are both equal, a random choice is made.

The process of choosing the actions and modifying the state action values is almost the same as the SARSA Policy. The only difference is when modifying the state action value.
When an action is chosen, given it is not a terminal state, the hand is updated, and the $maxQ\ (S', A)$ is found with the above function. The state action value is updated using the above formula.
When a terminal state is reached, the reward is calculated and the $Q(S, A)$ and $maxQ\ (S', A)$ are taken as the same.

7. Evaluation

7.1 Results of Different Algorithm Configurations

Below there are two different figures, the one-line graph represents the wins, draws and losses of the agent to every 1000-episode to a total of 500,000 episodes. Whilst the table represents the strategy the agent has developed after 500,000 for each possible state. This was done for each algorithm and the different configurations needed.

- **Monte Carlo, Exploring Start, epsilon = 1/k**

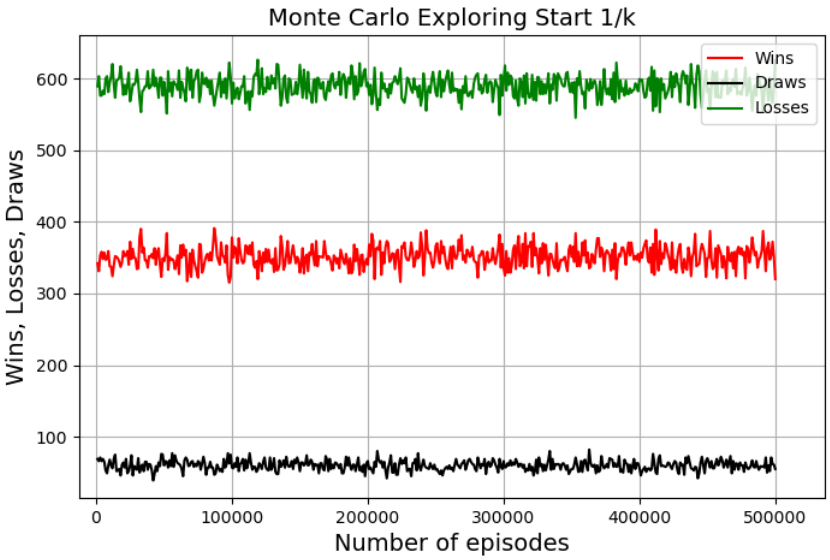


Table using Ace as 11:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT
14	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT
15	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT
16	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT
17	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

Table using Ace as 1:

	2	3	4	5	6	7	8	9	10	A
12	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	STAND
13	STAND	STAND	STAND	STAND	HIT	STAND	HIT	HIT	HIT	STAND
14	STAND	STAND	STAND	STAND	STAND	HIT	STAND	STAND	HIT	STAND
15	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT	HIT	STAND
16	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
17	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

- **Monte Carlo, Non-exploring start, epsilon = 1/k**

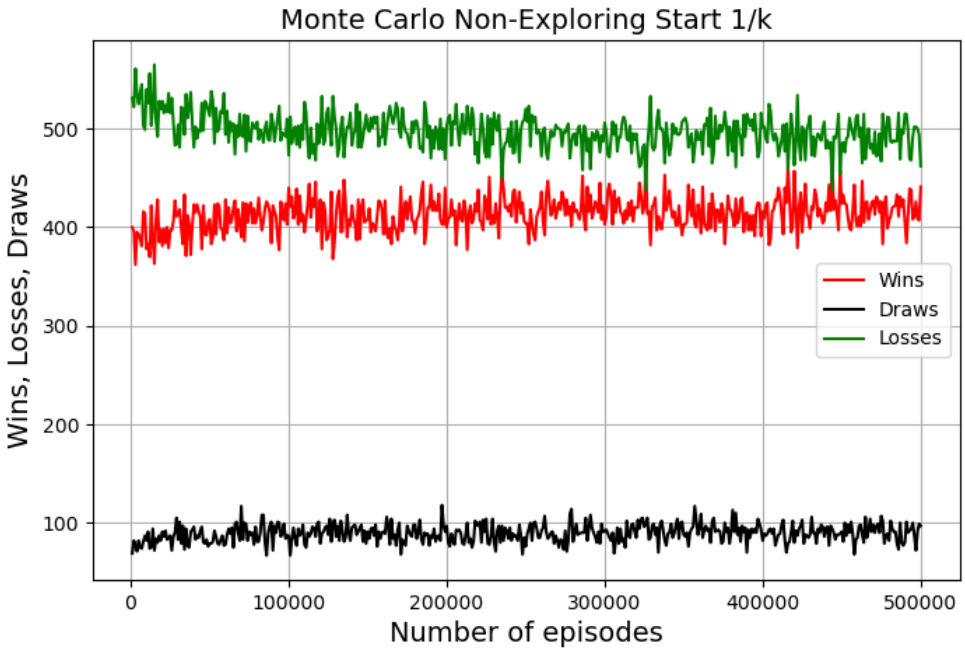


Table using Ace as 11:

	2	3	4	5	6	7	8	9	10	A
12	STAND	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT
14	STAND	STAND	STAND	HIT	HIT	HIT	STAND	HIT	HIT	HIT
15	HIT	HIT	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT
16	HIT	STAND	STAND	STAND	STAND	HIT	STAND	HIT	HIT	STAND
17	STAND	STAND	STAND	HIT	STAND	STAND	HIT	HIT	HIT	STAND
18	HIT	STAND	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT
19	STAND	STAND	STAND	HIT	STAND	STAND	STAND	STAND	STAND	HIT
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

Table using Ace as 1:

	2	3	4	5	6	7	8	9	10	A
12	STAND	STAND	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT
13	STAND	HIT	STAND	HIT	HIT	STAND	STAND	HIT	STAND	STAND
14	HIT	HIT	HIT	STAND	HIT	HIT	STAND	STAND	STAND	STAND
15	STAND	HIT	HIT	HIT	STAND	STAND	HIT	HIT	STAND	HIT
16	STAND	STAND	STAND	HIT	STAND	HIT	STAND	STAND	HIT	STAND
17	HIT	STAND	STAND	HIT	STAND	STAND	STAND	STAND	STAND	STAND
18	STAND	HIT	STAND	STAND	HIT	STAND	STAND	HIT	STAND	STAND
19	STAND	STAND	HIT	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT

- **Monte Carlo, Non-exploring start, epsilon = $\epsilon^{(-k/1000)}$**

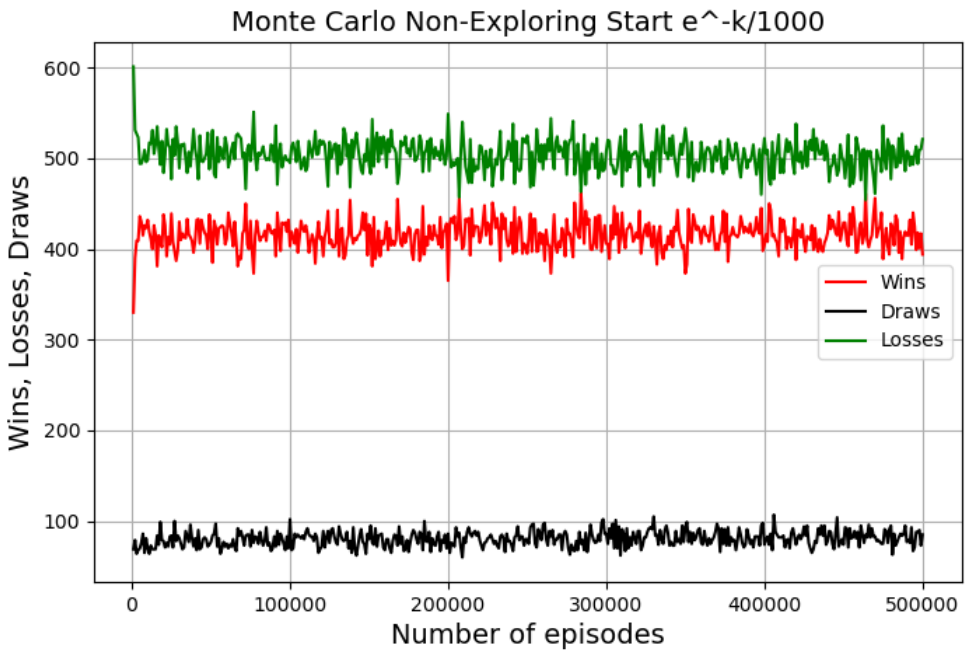


Table using Ace as 11:

	2	3	4	5	6	7	8	9	10	A
12	HIT	STAND	STAND	HIT	STAND	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	STAND	HIT	STAND	HIT	HIT	HIT	HIT	HIT
14	HIT	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT
15	STAND	STAND	STAND	STAND	STAND	HIT	STAND	HIT	STAND	HIT
16	HIT	HIT	STAND	STAND	STAND	HIT	HIT	STAND	HIT	HIT
17	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

Table using Ace as 1:

	2	3	4	5	6	7	8	9	10	A
12	STAND	STAND	STAND	HIT	STAND	HIT	HIT	HIT	STAND	HIT
13	HIT	HIT	HIT	STAND	STAND	HIT	HIT	STAND	STAND	STAND
14	HIT	STAND	HIT	STAND	HIT	HIT	HIT	STAND	STAND	STAND
15	STAND	STAND	STAND	HIT	STAND	HIT	HIT	HIT	STAND	STAND
16	STAND	HIT	STAND	STAND	HIT	STAND	STAND	STAND	STAND	STAND
17	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT
18	HIT	STAND	STAND	HIT	STAND	STAND	STAND	STAND	STAND	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	HIT	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

- **Monte Carlo, Non-exploring start, epsilon = $\epsilon^{(-k/10000)}$**

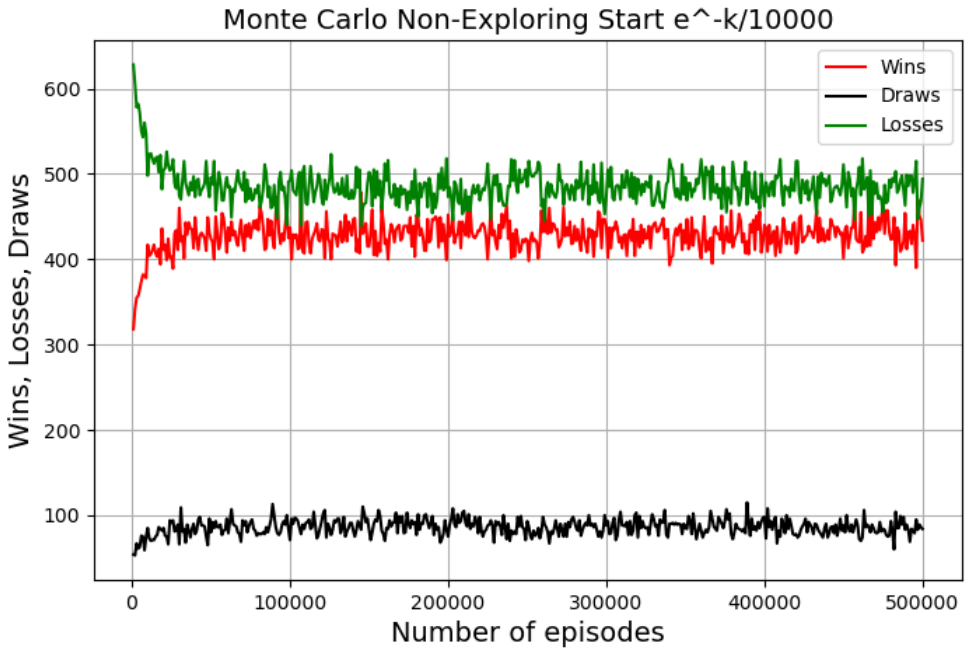


Table using Ace as 11:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	STAND	HIT	STAND	HIT	HIT	HIT	HIT	HIT
13	HIT	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT
14	HIT	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT
15	STAND	STAND	STAND	STAND	STAND	HIT	STAND	HIT	HIT	HIT
16	HIT	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT
17	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT	STAND	HIT
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

Table using Ace as 1:

	2	3	4	5	6	7	8	9	10	A
12	STAND	STAND	HIT	STAND	STAND	HIT	HIT	HIT	HIT	STAND
13	STAND	STAND	STAND	STAND	STAND	HIT	STAND	HIT	HIT	STAND
14	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	STAND
15	STAND	STAND	STAND	STAND	STAND	HIT	STAND	STAND	STAND	STAND
16	HIT	STAND	STAND	STAND	STAND	STAND	HIT	STAND	STAND	STAND
17	STAND	STAND	HIT	STAND	STAND	STAND	STAND	STAND	STAND	STAND
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

- **SARSA, epsilon = 0.1**

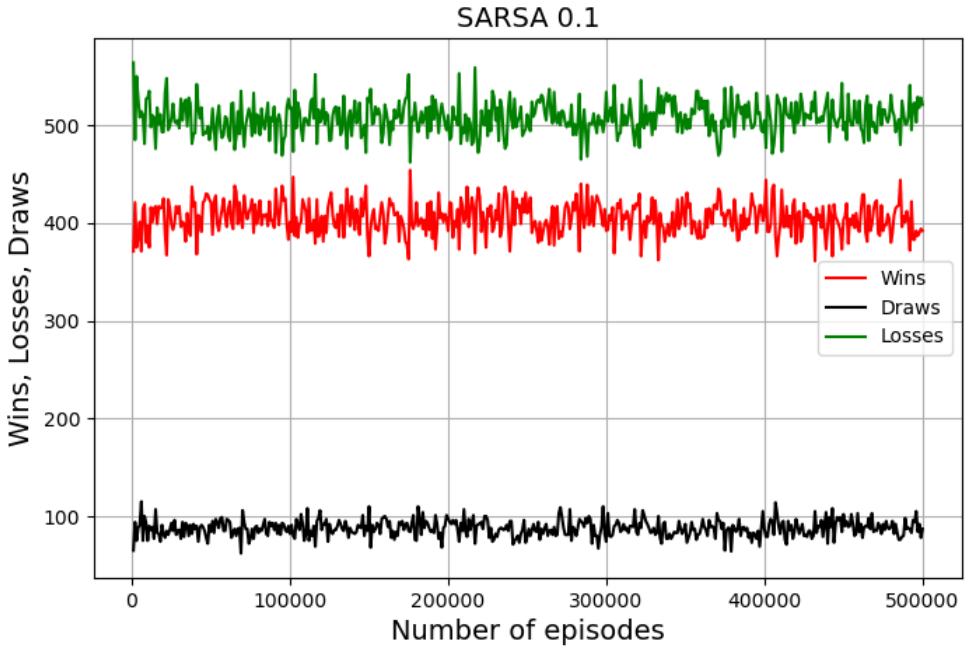


Table using Ace as 11:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
14	HIT	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT	HIT
15	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
16	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
17	STAND	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT	STAND
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

Table using Ace as 1:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT
14	HIT	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT	HIT
15	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
16	HIT	HIT	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT
17	STAND	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT	STAND
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

- **SARSA, epsilon = 1/k**

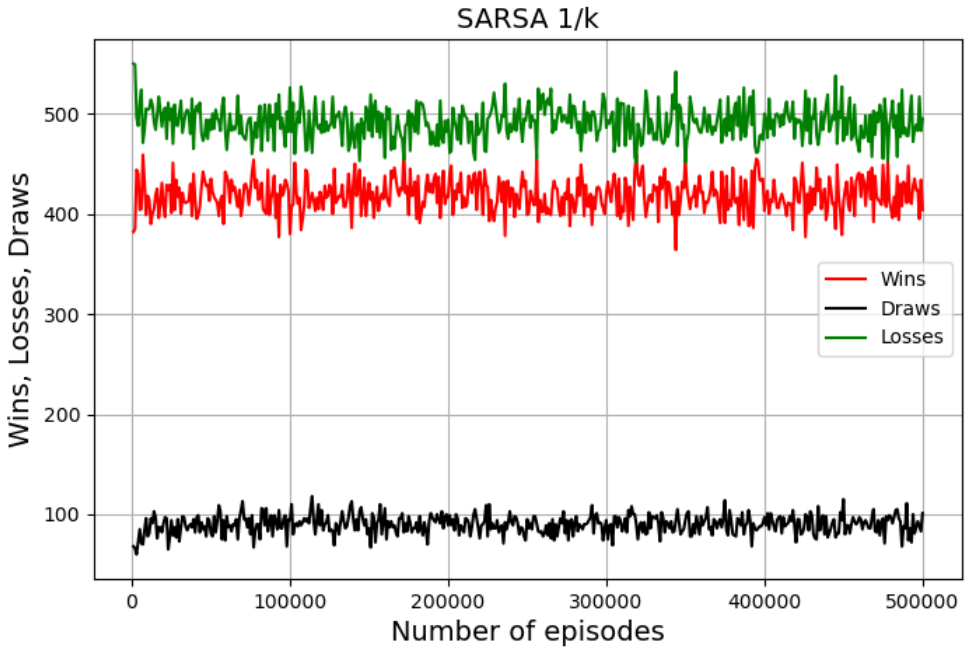


Table using Ace as 11:

	2	3	4	5	6	7	8	9	10	A
12	STAND	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
14	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
15	HIT	HIT	STAND	HIT	STAND	HIT	HIT	HIT	HIT	HIT
16	HIT	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT
17	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

Table using Ace as 1:

	2	3	4	5	6	7	8	9	10	A
12	STAND	HIT	STAND	HIT	STAND	HIT	HIT	HIT	HIT	STAND
13	HIT	HIT	HIT	STAND	STAND	HIT	HIT	HIT	HIT	STAND
14	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
15	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT	STAND
16	HIT	HIT	STAND	STAND	STAND	HIT	HIT	HIT	HIT	STAND
17	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT	STAND	STAND
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

- **SARSA, epsilon = $\epsilon^{(-k/1000)}$**

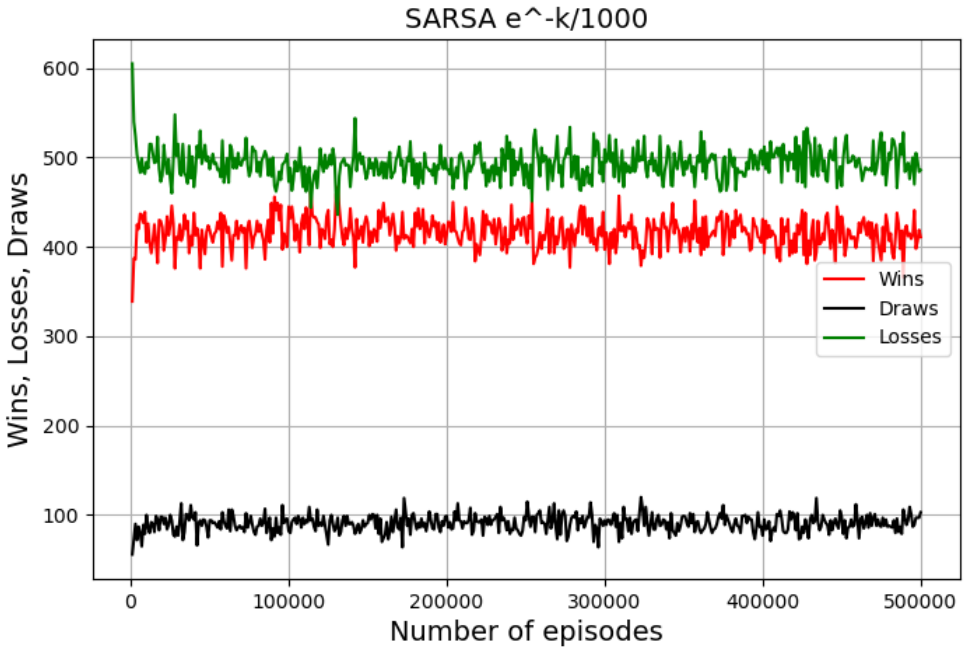


Table using Ace as 11:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
14	STAND	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT	STAND
15	HIT	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT
16	HIT	HIT	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT
17	HIT	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

Table using Ace as 1:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	STAND
14	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT	STAND
15	HIT	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	STAND
16	HIT	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT	STAND
17	HIT	STAND	STAND	STAND	STAND	STAND	HIT	STAND	HIT	STAND
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

- **SARSA, epsilon =** $e^{(-k/10000)}$

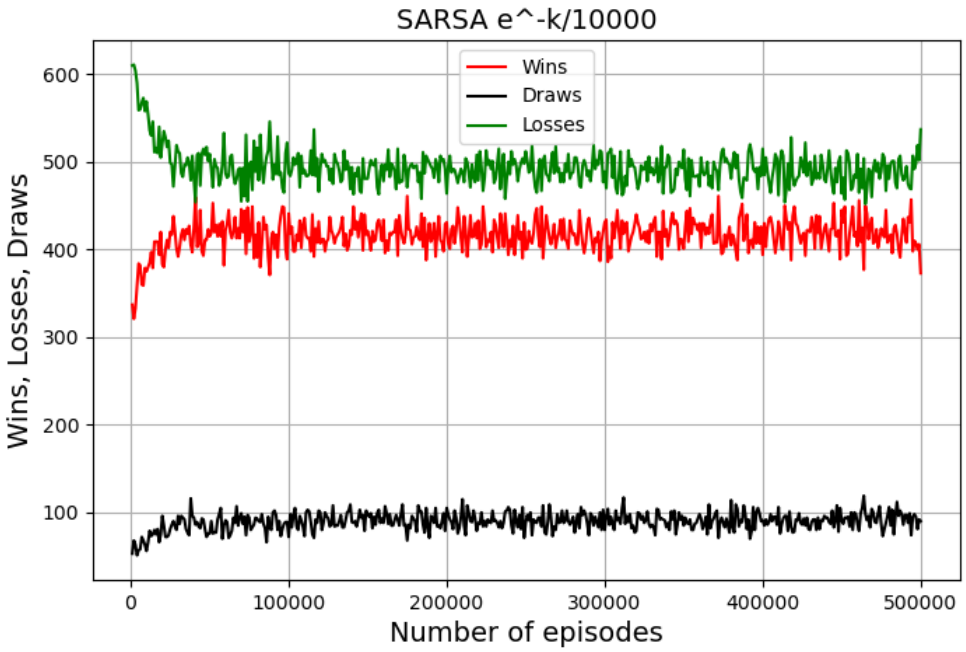


Table using Ace as 11:

	2	3	4	5	6	7	8	9	10	A
12	STAND	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
14	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT
15	HIT	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT
16	HIT	STAND	HIT	HIT	STAND	HIT	HIT	HIT	HIT	STAND
17	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT	HIT
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

Table using Ace as 1:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
13	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
14	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	STAND
15	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT
16	HIT	STAND	HIT	HIT	STAND	STAND	HIT	HIT	HIT	STAND
17	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT	STAND
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

- **Q-Learning, epsilon = 0.1**

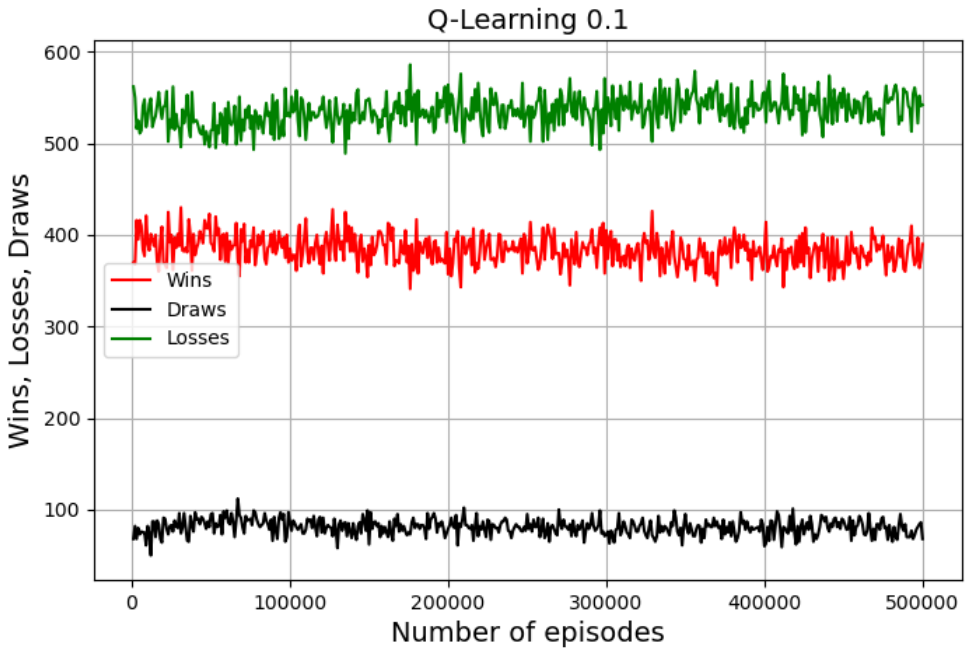


Table using Ace as 11:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
14	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
15	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
16	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
17	HIT	HIT	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT
18	STAND	STAND	STAND	STAND	HIT	HIT	STAND	HIT	HIT	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

Table using Ace as 1:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
14	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
15	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
16	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
17	HIT	HIT	STAND	STAND	HIT	HIT	HIT	HIT	HIT	HIT
18	STAND	STAND	STAND	STAND	HIT	HIT	STAND	HIT	HIT	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

- **Q-Learning, epsilon = 1/k**

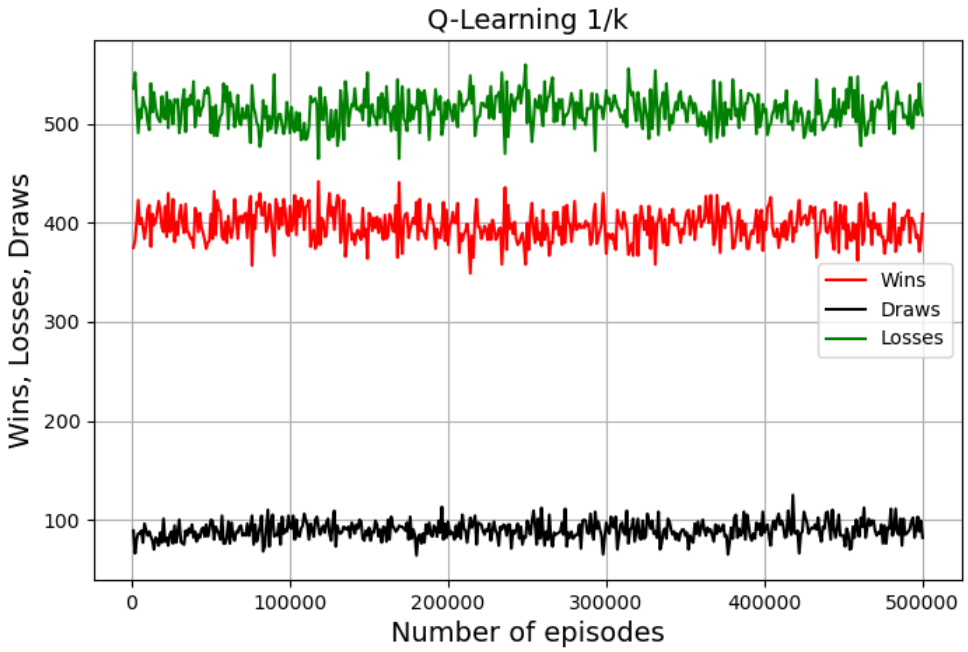


Table using Ace as 11:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
14	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
15	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
16	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	STAND
17	STAND	HIT	STAND	STAND	HIT	STAND	HIT	HIT	HIT	STAND
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT	HIT	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

Table using Ace as 1:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
14	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
15	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
16	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	STAND
17	STAND	HIT	STAND	STAND	HIT	STAND	HIT	HIT	HIT	STAND
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT	HIT	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

- **Q-Learning, epsilon =** $\epsilon^{(-k/1000)}$

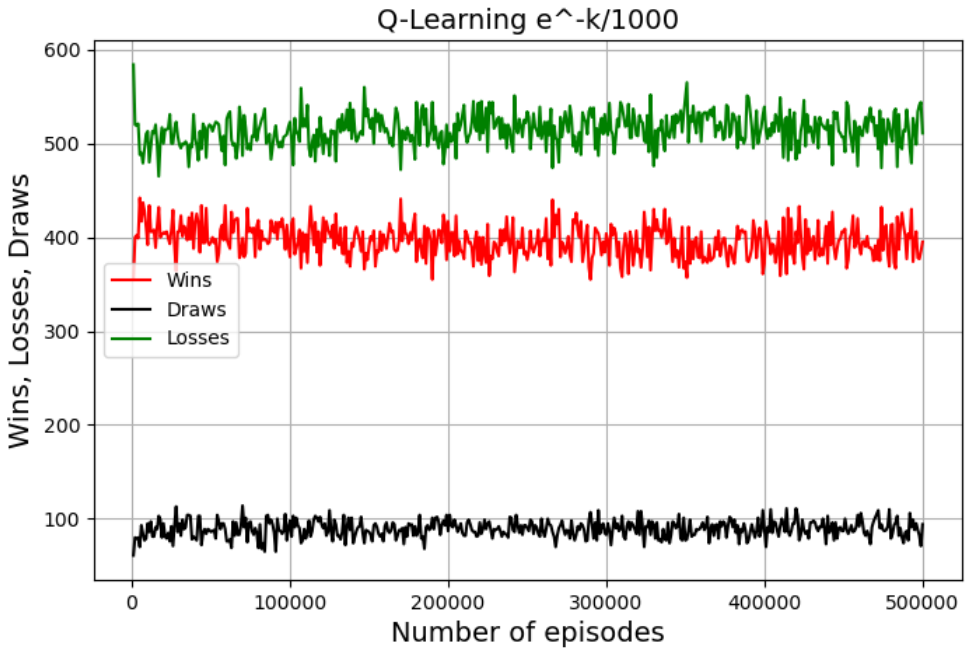


Table using Ace as 11:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
14	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
15	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
16	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
17	HIT	STAND	HIT	STAND	STAND	HIT	HIT	HIT	HIT	HIT
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT	HIT	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

Table using Ace as 1:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
14	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
15	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
16	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
17	HIT	STAND	HIT	STAND	STAND	HIT	HIT	HIT	HIT	HIT
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT	HIT	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

- **Q-Learning, epsilon =** $\epsilon^{(-k/10000)}$

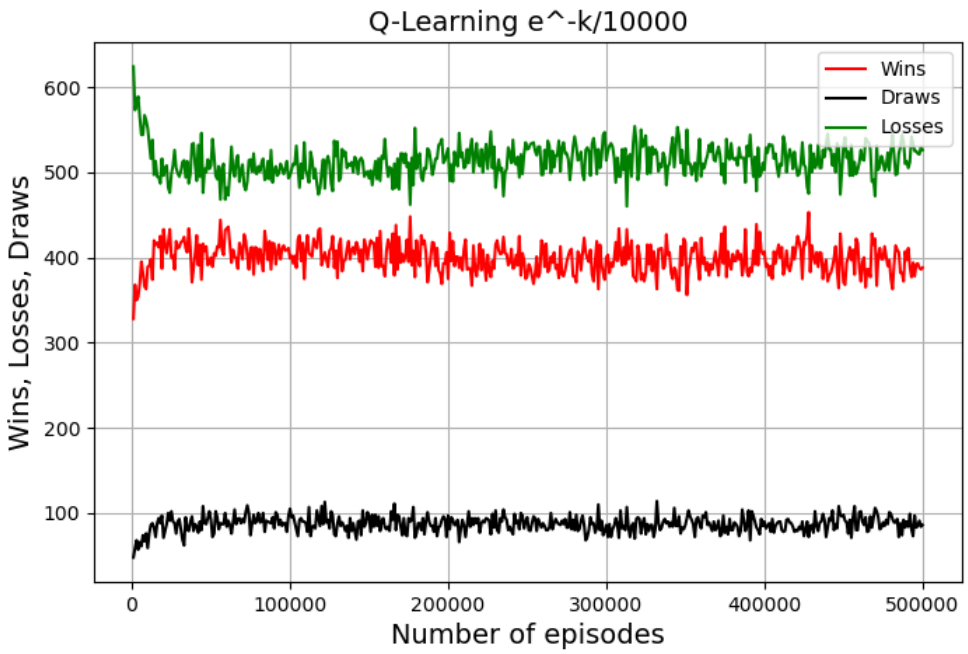


Table using Ace as 11:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
14	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
15	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
16	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
17	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	STAND
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

Table using Ace as 1:

	2	3	4	5	6	7	8	9	10	A
12	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT	HIT
13	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
14	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
15	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT	HIT	HIT
16	HIT	HIT	HIT	STAND	HIT	HIT	HIT	HIT	HIT	HIT
17	STAND	STAND	STAND	STAND	STAND	HIT	HIT	HIT	HIT	STAND
18	STAND	STAND	STAND	STAND	STAND	STAND	STAND	HIT	HIT	STAND
19	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND
20	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND	STAND

7.2 Analysis of Results

From the above results we can conclude the following. All the algorithms start with a win rate of around 300, but after each episode most algorithms move up in win-rate draw-rate and decrease in lose-rate. However, there are a select few of algorithms which do not improve but do not deteriorate with each episode as well.

The Monte Carlo algorithm had one configuration having an exploring start and three others without, one of the three also had the same epsilon policy as the exploring start. When comparing them we can conclude that the exploring start affects the agent on average by losing more games. We believe this happens due to the agent taking more risk than necessary, for example if the agent draws a 10-value card and a 9 the best decision would be to stand. However, with an exploring start we are giving the hit choice a 50% chance to occur. When using the e numerical constant, we can observe that the agent after a few thousand episodes drastically increases its win-rate and draw-rate and hence decreases it lose-rate. Comparing all the Monte Carlo configurations, the best results proved to be when epsilon was equal to $e^{(-k/10000)}$.

The SARSA algorithm had four different configurations with different epsilon values. The main difference between them was that the configuration having a non-e numerical constant in the epsilon had a constant average of wins to loss ratio. Whilst the e numerical constant configuration in the epsilon started with a lower win rate but climbed a bit further in win rate. Comparing all the SARSA configurations, the best results proved to be when epsilon was equal to $e^{(-k/10000)}$, since the win-rate did not fluctuate below a certain level compared to the others.

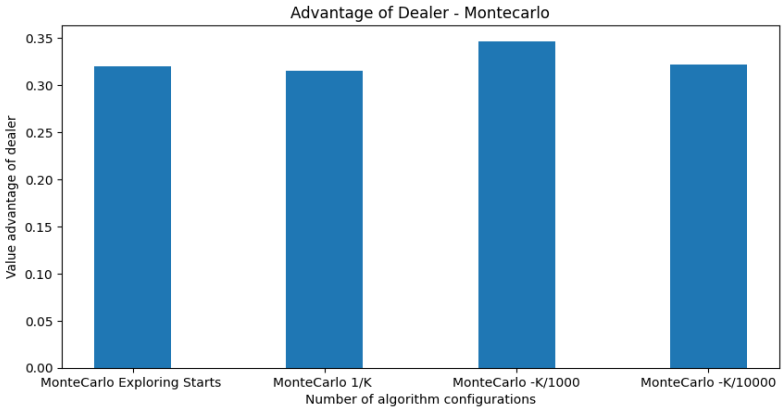
The Q-Learning algorithm also had four different configurations with different epsilon values and achieved similar results to that of the SARSA algorithm. Only difference being that after all the episodes they achieved very similar win to loss ratio at the end. Similarly, to SARSA, the best configuration seemed to be $e^{(-k/10000)}$.

After comparing the three algorithms, the Monte Carlo with epsilon $e^{(-k/10000)}$ proved to be the best trained model since after a certain number of episodes the win and loss rates seemed to almost converge on the graph. The win-rate seemed to be the highest overall.

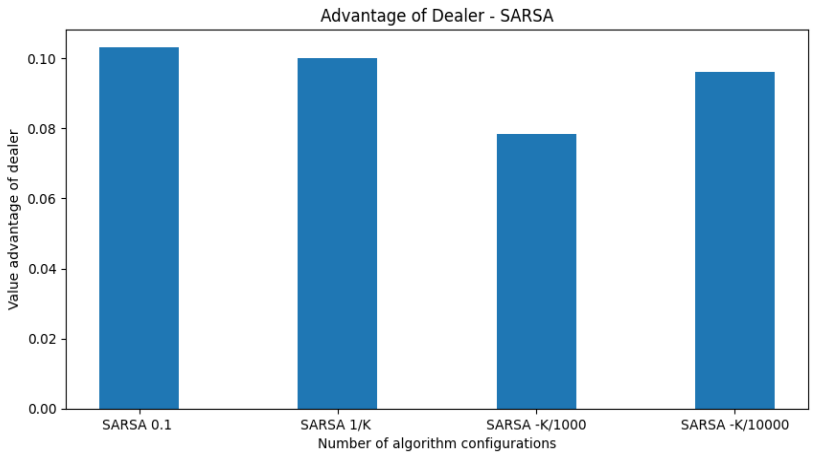
7.3 Dealer Advantage

The plot figures below represent the calculated dealer advantage for each algorithm with different epsilon values.

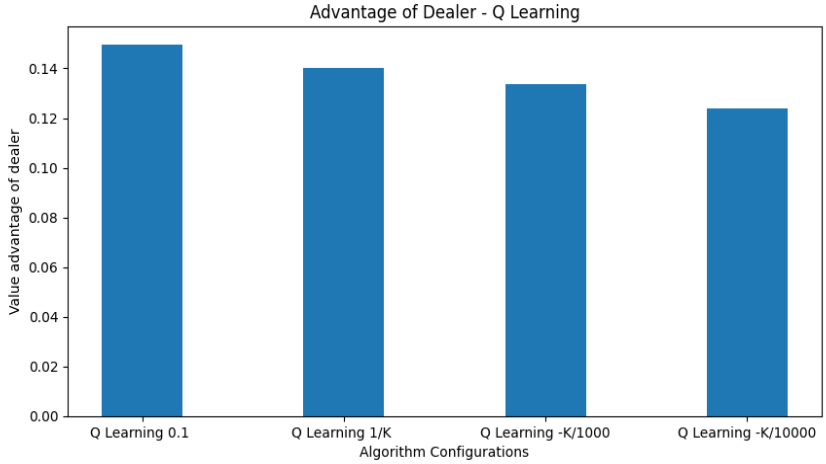
Monte Carlo



SARSA



Q-Learning



7.4 Analysis of Dealer Advantage

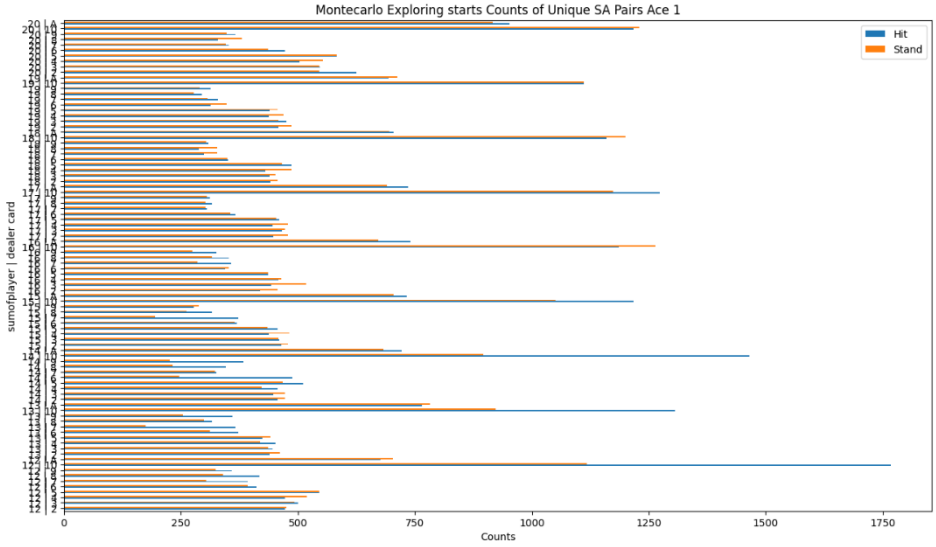
From the above figures we can conclude that the best algorithm and configuration to minimise dealer advantage was the SARSA On-Policy Control with an epsilon equal to $\in^{(-k/1000)}$ Since it has the lowest value of 0.08 advantage. On the other the worst policy was the Monte Carlo On-Policy Control with an epsilon equal to $\in^{(-k/1000)}$ giving the dealer a huge advantage of 0.30. These values were calculated using the following equation:

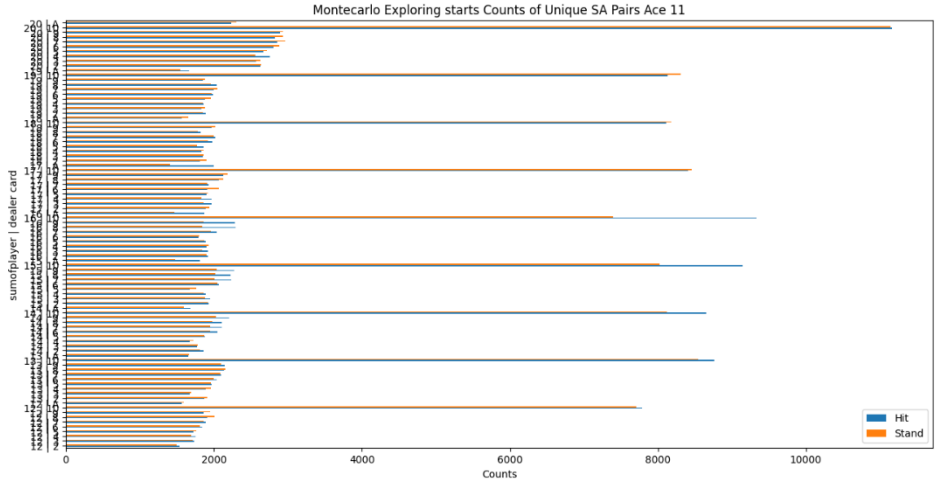
$$\frac{l-w}{l+w}$$

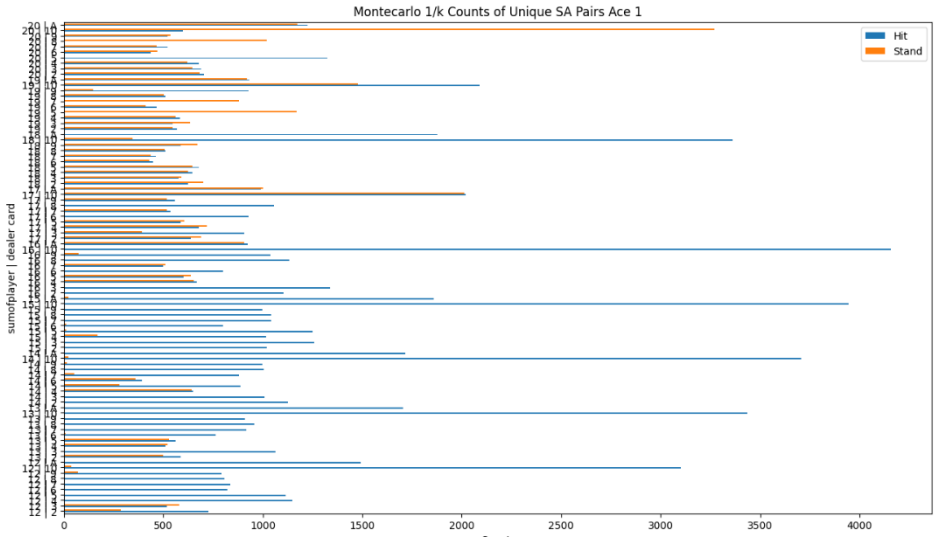
7.5 Hit and Stand Count

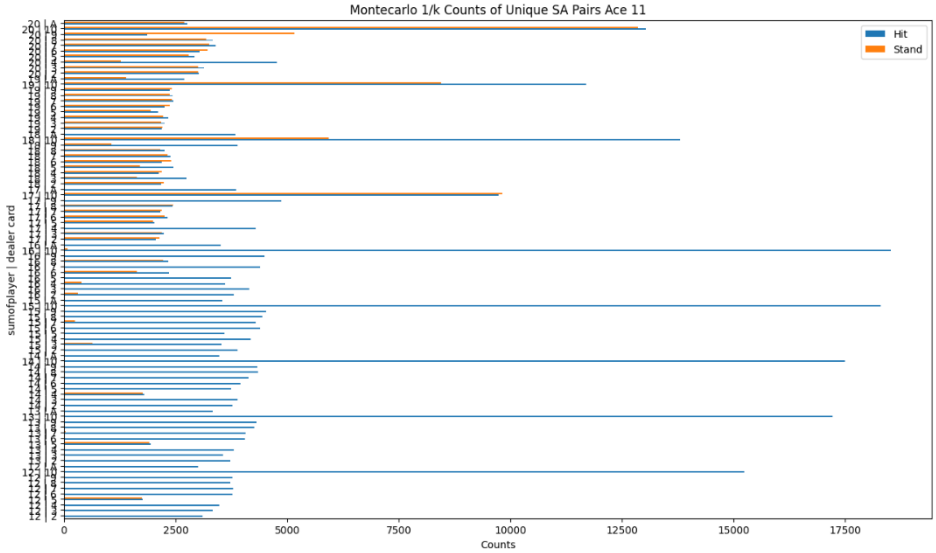
The bar charts below contain the hit and stand count for each algorithm and configuration.

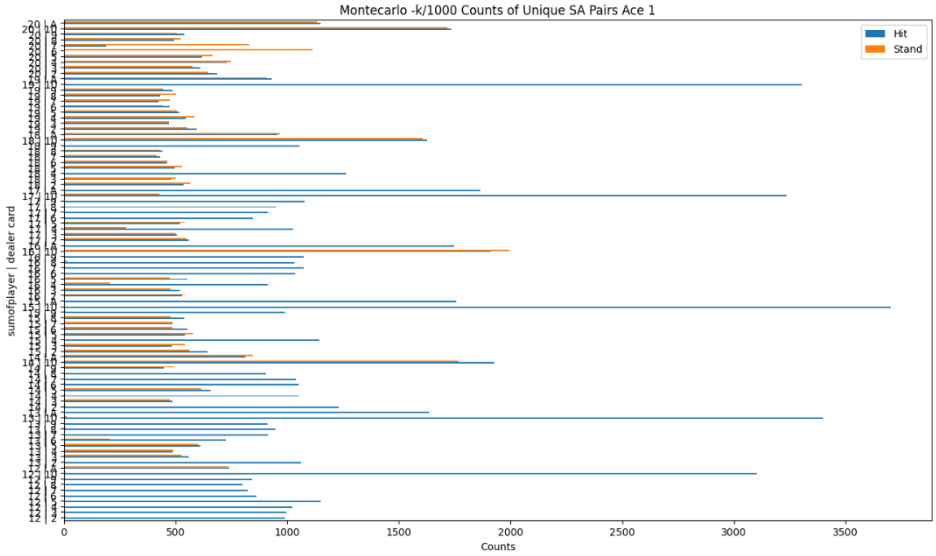
Monte Carlo

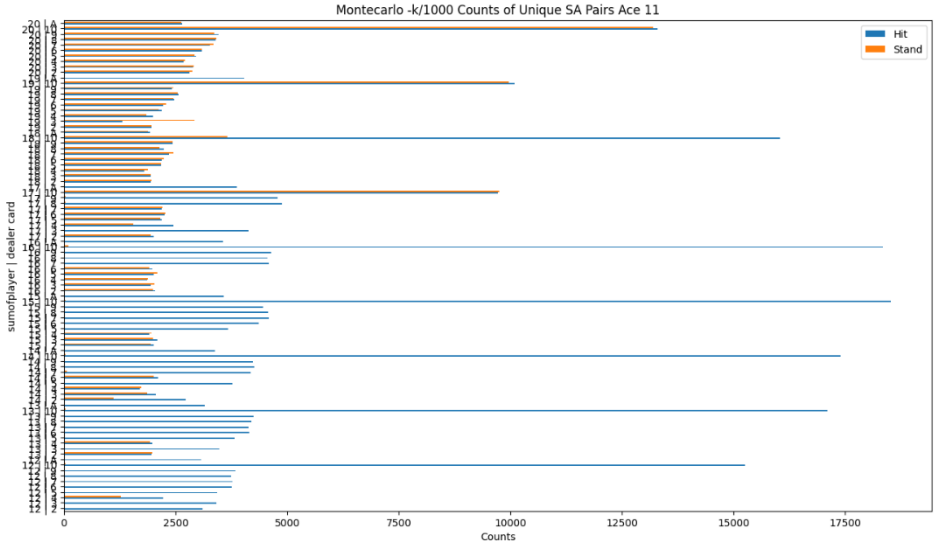

Montecarlo Exploring starts Counts of Unique SA Pairs Ace 1
Hit
Stand
sumofplayer | dealer card
Counts
0
250
500
750
1000
1250
1500
1750

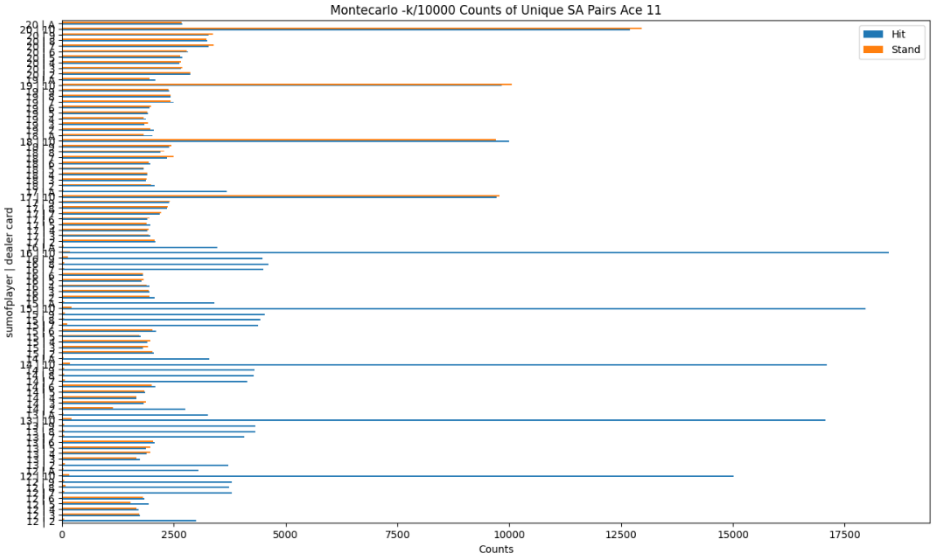

Montecarlo Exploring starts Counts of Unique SA Pairs Ace 11
Hit
Stand
sumofplayer | dealer card
Counts
0
2000
4000
6000
8000
10000

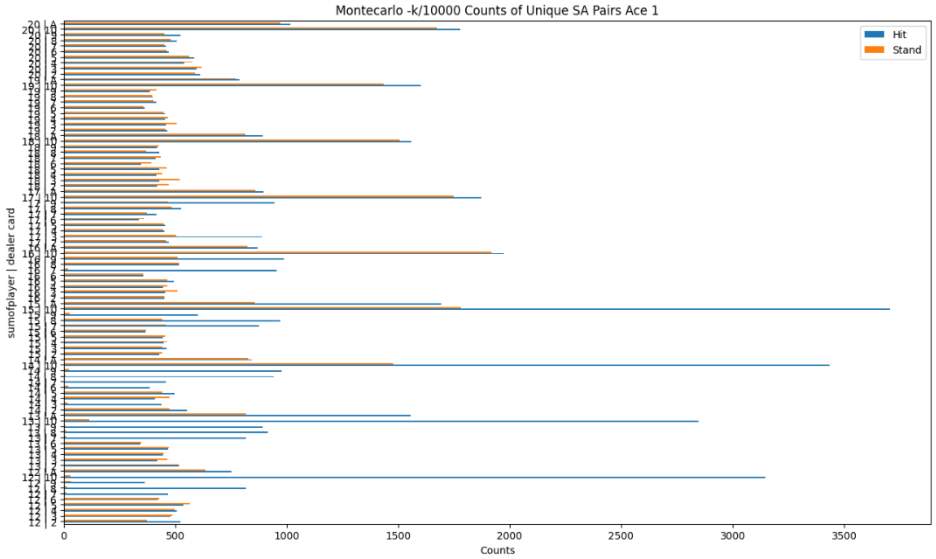

Montecarlo 1/k Counts of Unique SA Pairs Ace 1
Hit
Stand
sumofplayer | dealer card
0
500
1000
1500
2000
2500
3000
3500
4000


Montecarlo 1/k Counts of Unique SA Pairs Ace 11
Hit
Stand
sum of player | dealer card
Counts
0
2500
5000
7500
10000
12500
15000
17500

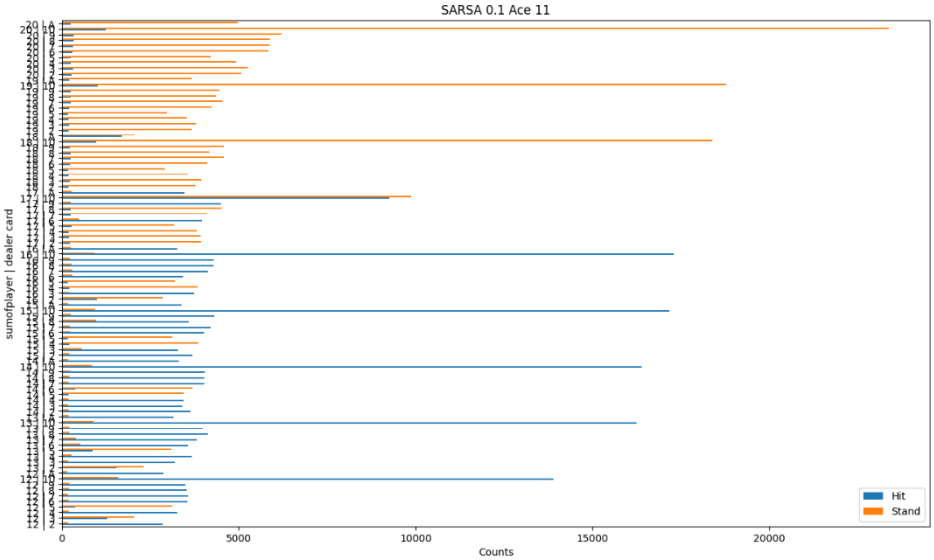

Montecarlo -k/1000 Counts of Unique SA Pairs Ace 1
Hit
Stand
sum of player | dealer card
Counts
0
500
1000
1500
2000
2500
3000
3500

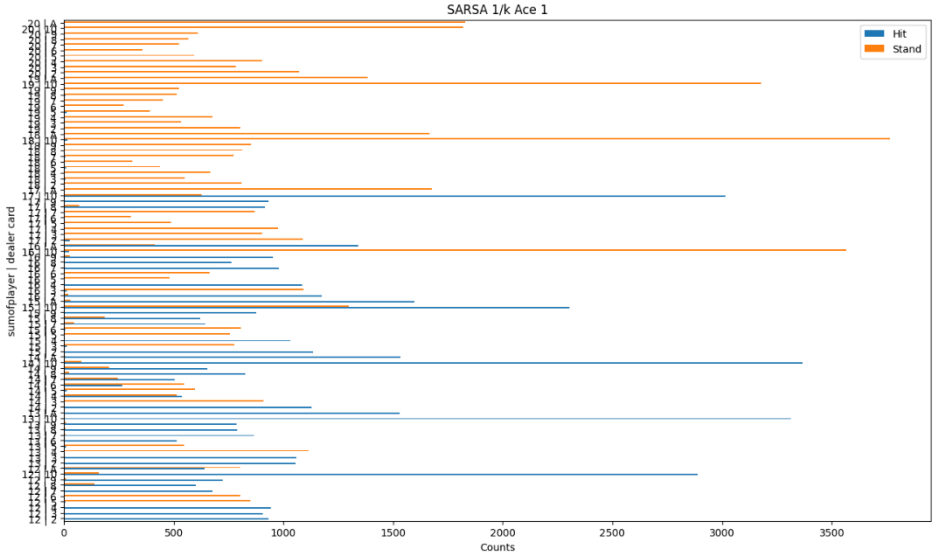

Montecarlo -k/1000 Counts of Unique SA Pairs Ace 11
Hit
Stand
sum of player | dealer card
Counts
0
2500
5000
7500
10000
12500
15000
17500

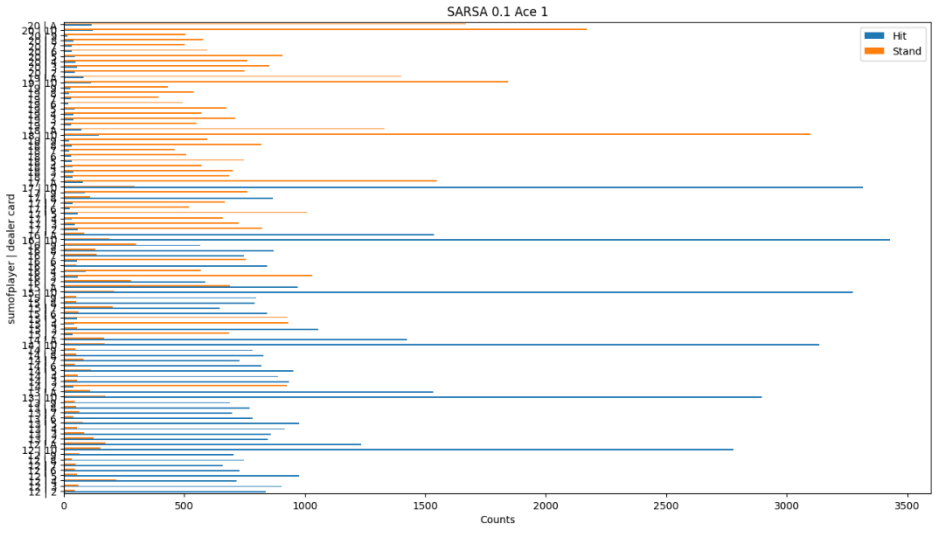

Montecarlo -k/10000 Counts of Unique SA Pairs Ace 11
Hit
Stand
sumofplayer | dealer card
Counts
0
2500
5000
7500
10000
12500
15000
17500

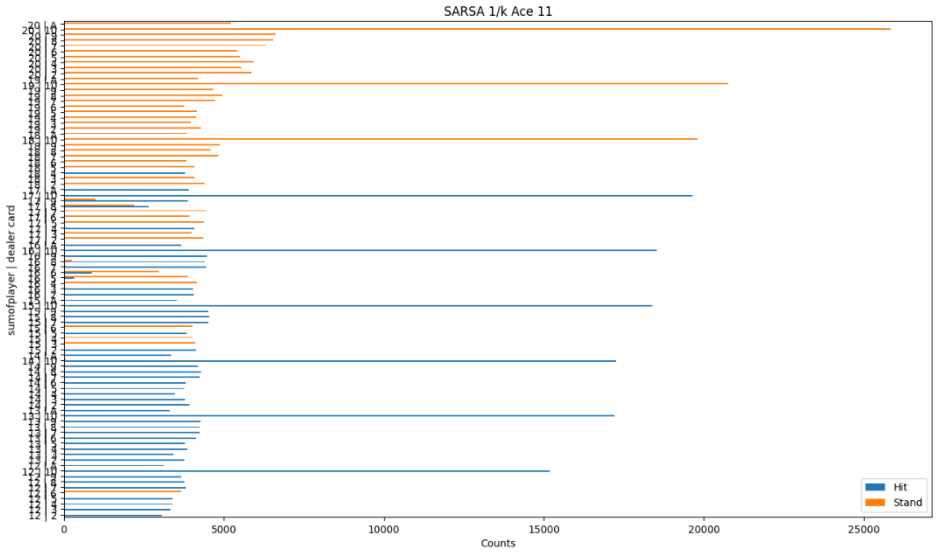

Montecarlo -k/10000 Counts of Unique SA Pairs Ace 1
Hit
Stand
sumofplayer | dealer card
Counts
0
500
1000
1500
2000
2500
3000
3500

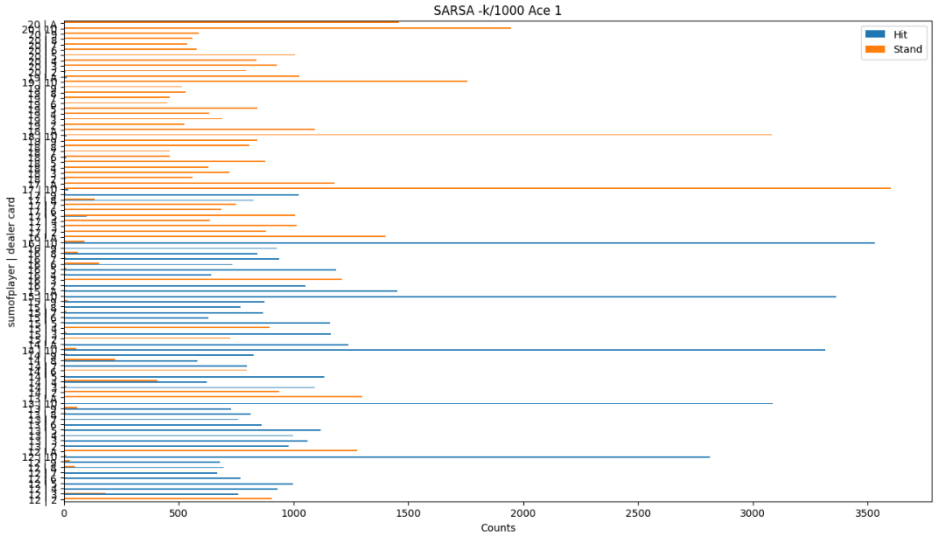
SARSA

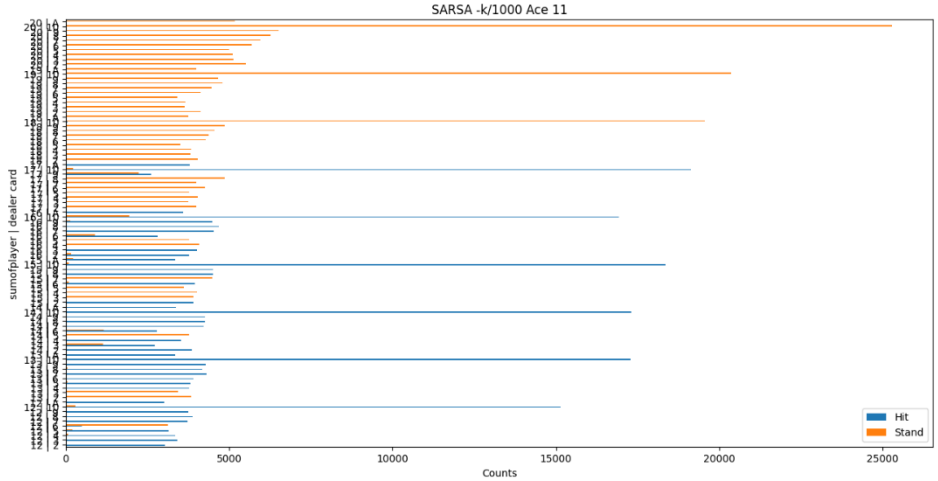

SARSA 0.1 Ace 11
sumofplayer | dealer card
Counts
Hit
Stand

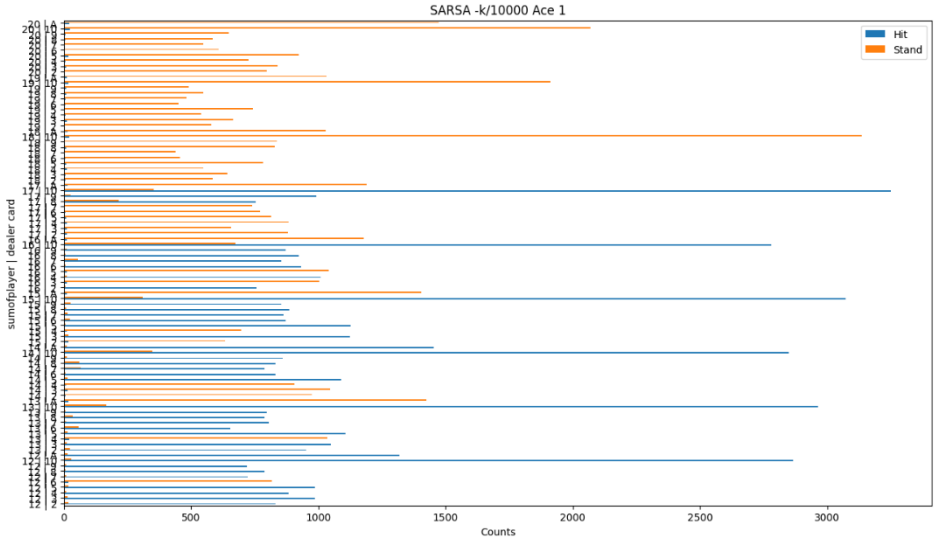

SARSA 1/k Ace 1
sumofplayer | dealer card
Counts
Hit
Stand

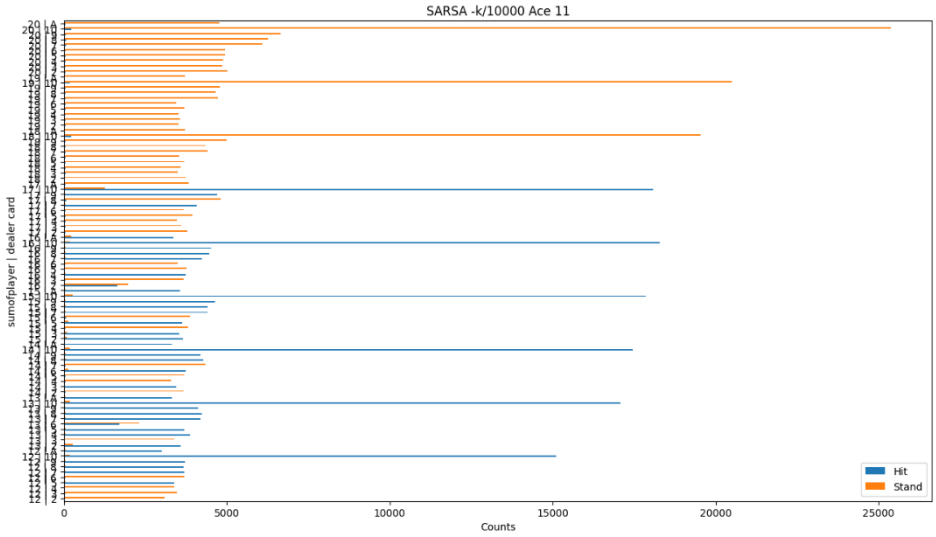

SARSA 0.1 Ace 1
sumofplayer | dealer card
Counts
Hit
Stand


SARSA 1/k Ace 11
sumofplayer | dealer card
Counts
Hit
Stand

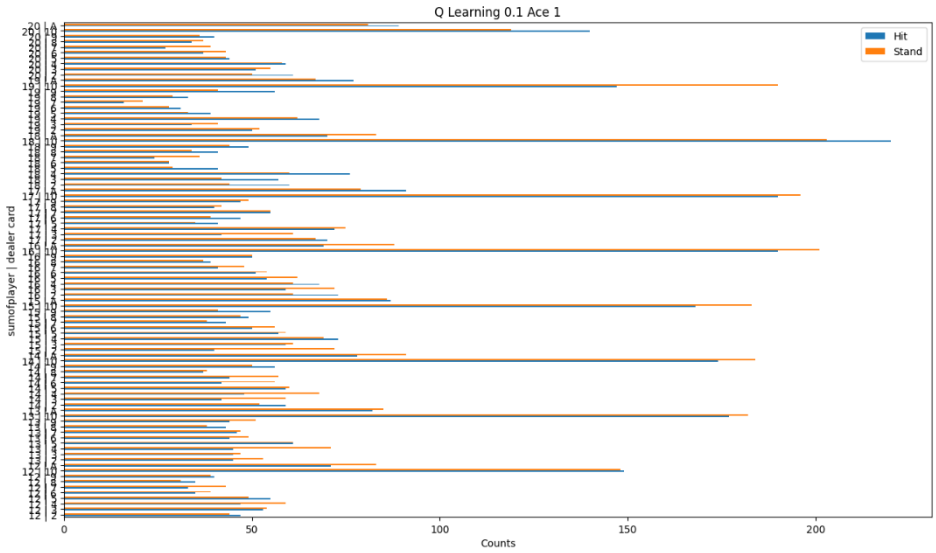

SARSA -k/1000 Ace 1
sumofplayer | dealer card
Counts
Hit
Stand

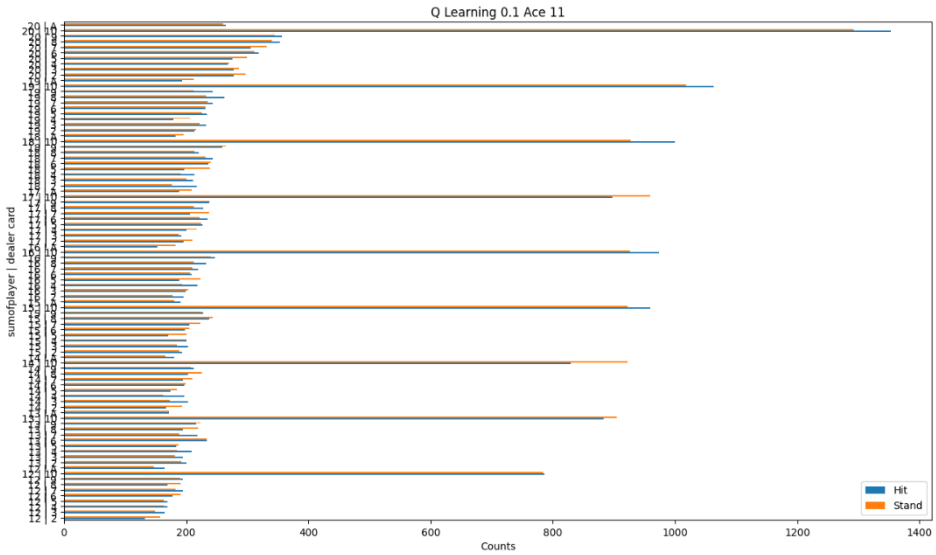

SARSA -k/1000 Ace 11
sumofplayer | dealer card
Counts
Hit
Stand

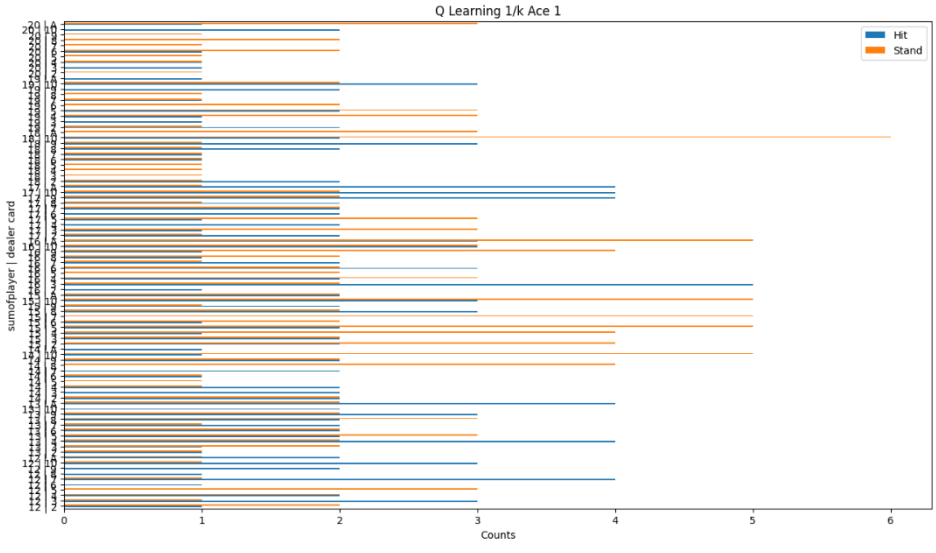

SARSA -k/10000 Ace 1
Hit
Stand
sum(player) | dealer card
Counts
0
500
1000
1500
2000
2500
3000

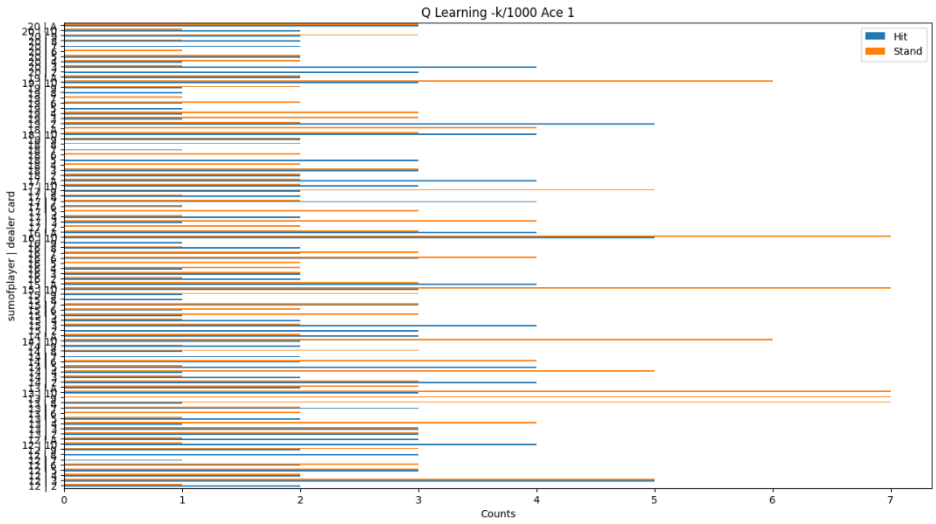

SARSA -k/10000 Ace 11
Hit
Stand
sum(player) | dealer card
Counts
0
5000
10000
15000
20000
25000

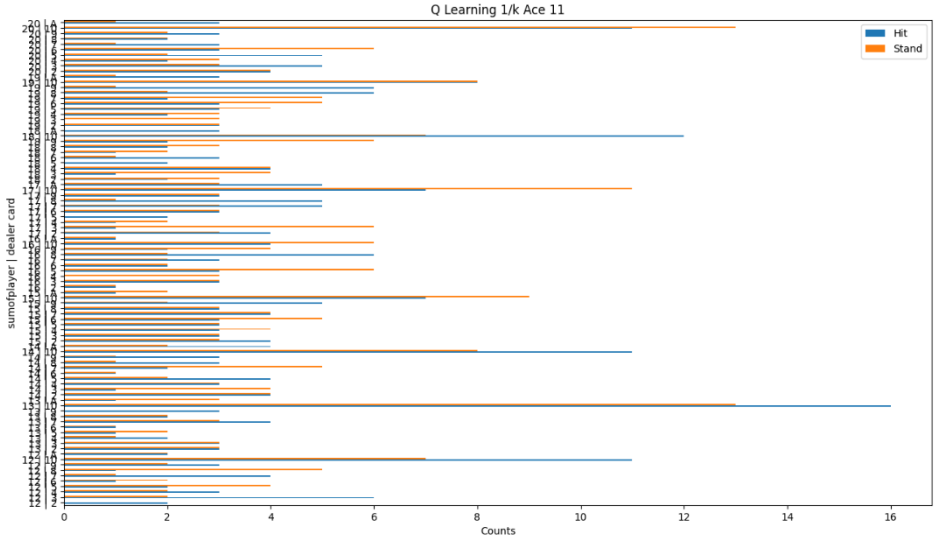
Q-Learning

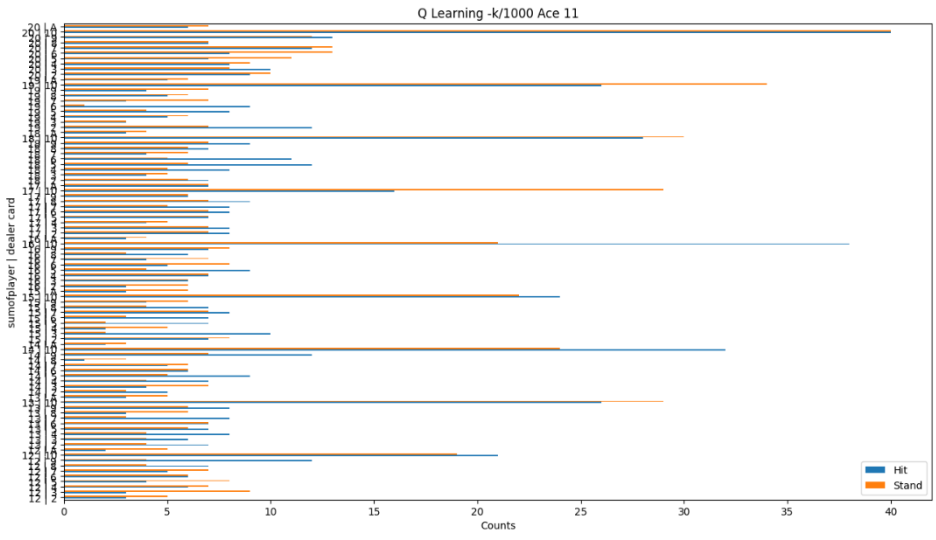

Q Learning 0.1 Ace 1
Hit
Stand
sum(player) | dealer card
Counts
0
50
100
150
200

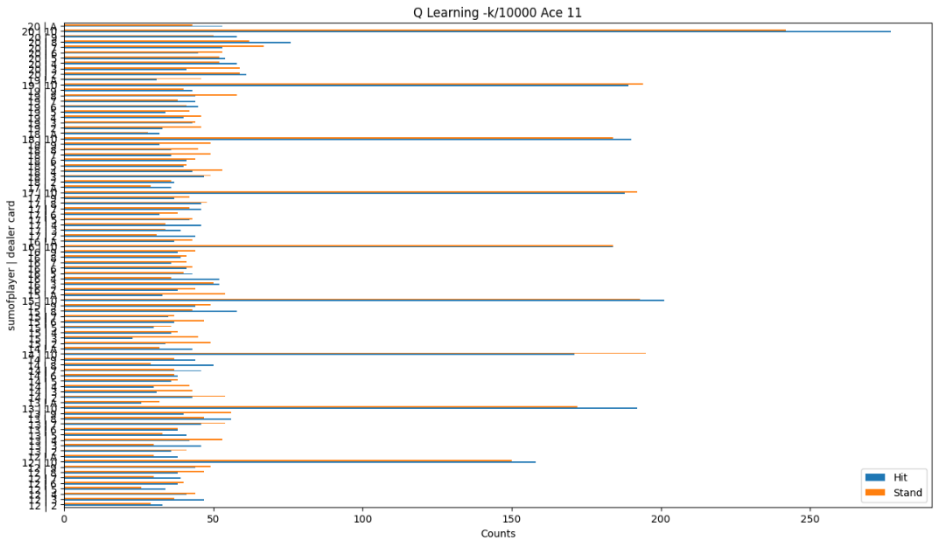

Q Learning 0.1 Ace 11
Hit
Stand
sum(player) | dealer card
Counts
0
200
400
600
800
1000
1200
1400

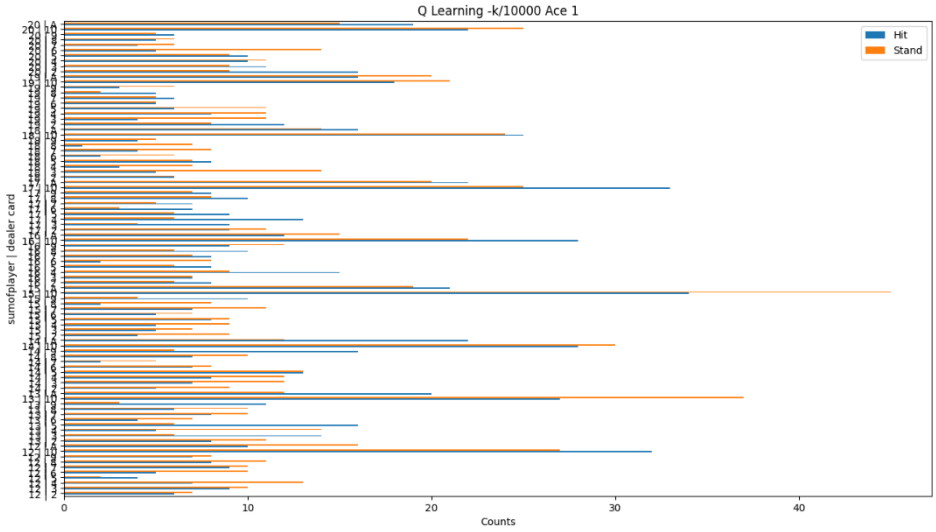

Q Learning 1/k Ace 1
Hit
Stand
sum(player) | dealer card
Counts
0
1
2
3
4
5
6


Q Learning -k/1000 Ace 1
Hit
Stand
sum(player) | dealer card
Counts
0
1
2
3
4
5
6
7


Q Learning 1/k Ace 11
Hit
Stand
sum(player) | dealer card
Counts
0
2
4
6
8
10
12
14
16


Q Learning -k/1000 Ace 11
Hit
Stand
sum(player) | dealer card
Counts
0
5
10
15
20
25
30
35
40





7.6 Analysis of Hit and Stand Count

Overall, all the algorithms showed the same pattern, the lower the hand value the agent had the higher probability it would choose to hit. In some instances the agent would not hit due to the dealer having weak cards such as 2 or 3 and would be safer to stand.

8. Distribution of Work

Work:	**Done By:**
Creating environment	Matthew Calafato /Liam Bugeja Douglas
Environment Utilities	Matthew Calafato
Monte Carlo implementation	Matthew Calafato/Liam Bugeja Douglas
SARSA implementation	Liam Bugeja Douglas/ Matthew Calafato
Q-Learning	Matthew Calafato
Documentation Overview	Liam Bugeja Douglas
Monte Carlo Documentation	Matthew Calafato /Liam Bugeja Douglas
SARSA Documentation	Liam Bugeja Douglas
Q-Learning Documentation	Matthew Calafato
Graphs	Aidan Seychell
Evaluation	Aidan Seychell / Liam Bugeja Douglas

Liam Bugeja Douglas LiamBD

_____ _____

Student Name Signature

Matthew Calafato

_____ _____

Student Name Signature

Aidan Seychell

_____ _____

Student Name Signature

9. References

[1] (). *Basic Rules in Playing Live Blackjack*.
Available: https://www.happistar.tips/tutorials/live-blackjack-rules/.

[2] (November 26th,). *Blackjack card values: Learn the hands of blackjack.*

Available: https://edge.twinspires.com/casino-news/blackjack-card-values-learn-the-hands-of-blackjack/.

[3] (). *Blackjack Statistics: Some Quick Stats To Keep In Mind Before And During Play.*
Available: https://www.goldentouchcraps.com/article.php?p=tamburin_005.html.

[4] (Feb 12,). *Winning Blackjack using Machine Learning*.
Available: https://towardsdatascience.com/winning-blackjack-using-machine-learning-681d924f197c.

[5] (Apr 29,). *Monte Carlo Methods in Reinforcement Learning — Part 2 off-policy Methods.* Available: https://medium.com/analytics-vidhya/monte-carlo-methods-in-reinforcement-learning-part-2-off-policy-methods-b9766d3e60d7.

[6] (Apr 29,). *Monte Carlo Methods in Reinforcement Learning — Part 1 on-policy Methods.* Available: https://medium.com/analytics-vidhya/monte-carlo-methods-in-reinforcement-learning-part-1-on-policy-methods-1f004d59686a.

[7] (Jul 23,). Introduction to Reinforcement Learning (Coding SARSA) — Part 4. Available: https://medium.com/swlh/introduction-to-reinforcement-learning-coding-sarsa-part-4-2d64d6e37617.

[8] (Mar 18,). *Simple Reinforcement Learning: Q-learning.*
Available: https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56.