**15CS376L – MINOR PROJECT II**

**SEMESTER – VI**

|  |  |
| --- | --- |
| **ACADEMIC YEAR** | **: 2019 - 2020** |
| **NAME & REG NO.** | **: Shivank Yadav RA1711003010488** |
|  |  |



**SCHOOL OF COMPUTING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,**

**S.R.M NAGAR, KATTANKULATHUR – 603203.**

**CHENGALPET DISTRICT**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,**

**KATTANKULATHUR – 603203.**

****

**BONAFIDE CERTIFICATE**

**Register No: RA1711003010488 Date: 08/05/2020**

Certified to be the Bonafide record of work done by SHIVANK YADAV

Of  B.Tech Ⅲ Year COMPUTER SCIENCE ENGINEERING course in SRM Institute of Science & Technology, Kattankulathur Campus for the subject MINOR PROJECT-Ⅱ .

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** |  | **Faculty In-charge** | **Head of the Department** |
|  |  |  |  |

Submitted for the University Examination held in ………………………………….

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** |  | **Internal Examiner** | **External Examiner** |

# **ACKNOWLEDGEMENT**

It is not possible to complete a project without the assistance and encouragement of other people. This one is certainly no exception.

On the very outset of this report, I would like to extend my sincere & heartfelt obligation towards all the personages who have helped me in this endeavor. Without their active guidance, help, cooperation & encouragement, I would not have made headway in the project.

##### I am extremely thankful and pay my gratitude to my faculty Dr. R. Kayalvizhi.

for her constant motivation, valuable guidance and support on completion of this project.

I extend my gratitude to SRM Institute of Science and technologyfor giving me this opportunity.

At last but not least gratitude goes to all of my friends who directly or indirectly helped me to complete this project report.

Any omission in this brief acknowledgement does not mean lack of gratitude.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **CONTENT** | **Page No.** |
|  | ABSTRACT | 4 |
|  | INTRODUCTION | 5-6 |
|  | ARCHITECTURAL DIAGRAM | 7-10 |
|  | IMPLEMENTATION | 11-14 |
|  | RESULT ANALYSIS | 15- 17 |
|  | CONCLUSION | 18 |
|  | REFERENCES | 18 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

# **ABSTRACT**

The goal of the project is to make an agent learn how to play the game of Blackjack. In the beginning we discuss the technological stack/requirements. Then we focus some time to understand the mechanics of the game and how this mechanics correlate to our OpenAI Gym Blackjack environment. Then we clear the basics of Reinforcement Learning and learn about some of the terminologies. We have a look at the very algorithm that we are going to use to solve this problem. That is the Monte Carlo Control Algorithm.

At this point, a basic understanding level is required to be developed in the mind of the readers to understand the algorithm. So we dive into a basic 4x4 Gridworld example. Through this example, I try to develop a visual understanding in the reader’s mind so that interpreting the algorithm becomes easy. The concept of Episodic tasks, Action space, State space, Reward, Policies, etc. is developed. Then we look at how Q-table is created and maintained that is, Prediction and Improvement processes of Monte Carlo Control Algorithm. We also talk about Greedy policies and e-greedy policies while choosing actions and thus, talk about exploitation vs exploration dilemma.

Then we go back to applying the knowledge we have learned above, on the Blackjack environment. First, we describe the environment through code. Then we see how the environment responds to out agent by making it play some rounds using equiprobable random policy.

Then we look at the algorithms that were used. And then we look at the code.

At the end, we compare out policy output with the optimal policy found in the research and found that we were able to reach very close to it.

# **INTRODUCTION**

**AIM:** Our Goal is to teach Machine to play Blackjack.

**TEXHNOLOGY STACK :**

* Python 3.6
* OpenAI gym module.
* Numpy, collections and sys for programming
* Matplotlib and basemap for plotting utilities.

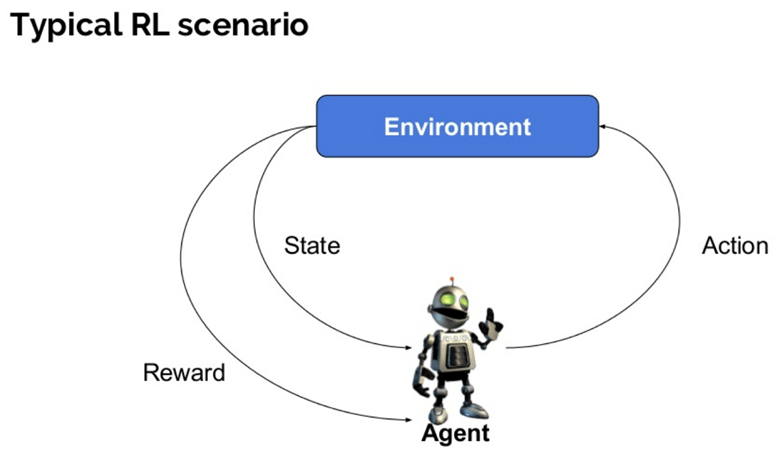
**BLACKJACK :**

* This is the version where the player plays with dealer.
* The game starts with dealer having one face up and one face down card, while player having two face up cards. They're playing against a fixed dealer.
* The goal is to obtain cards that **sum to as near as possible to 21 without going over**.
* Face cards (Jack, Queen, King) have point **value 10.**
* **Aces** can either count as **11 or 1**, If the player holds an ace that he could count as 11 without going bust, then the ace is said to be **usable.**
* This game is played with an infinite deck (or with replacement).
* The player can request additional cards (**hit=1**) until they decide to stop (**stick=0**) or exceed 21 (**bust**).
* After the player sticks, the dealer reveals their facedown card, and draws until his **sum is 17 or greater**. If the dealer goes bust the player wins.
* If neither player nor dealer busts, the outcome (win, lose, draw) is decided by whose sum is closer to 21. The reward for winning is **+1, drawing is 0, and losing is -1.**

**REINFORCEMENT LEARNING:**

* We have an **agent** and **environment**. Time is broken into **discrete time steps** and at every time step, the agent receives a **reward** and **state** from the environment, and chooses an **action** to perform in response.
* For any **episode**, the agent's goal is to find the **optimal policy** in order to maximize **expected cumulative reward**.

# **ARCHITECTURAL DIAGRAMS AND EXPLAINATION**

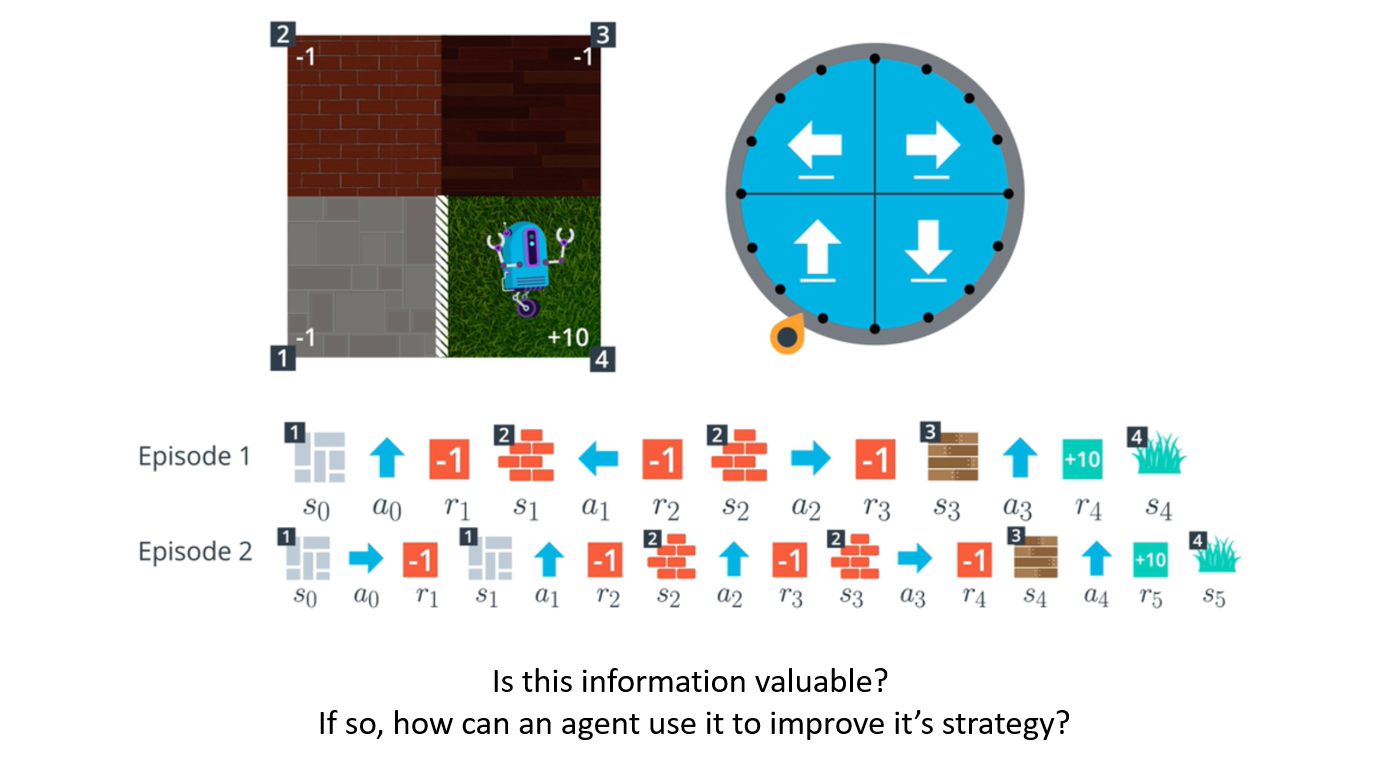


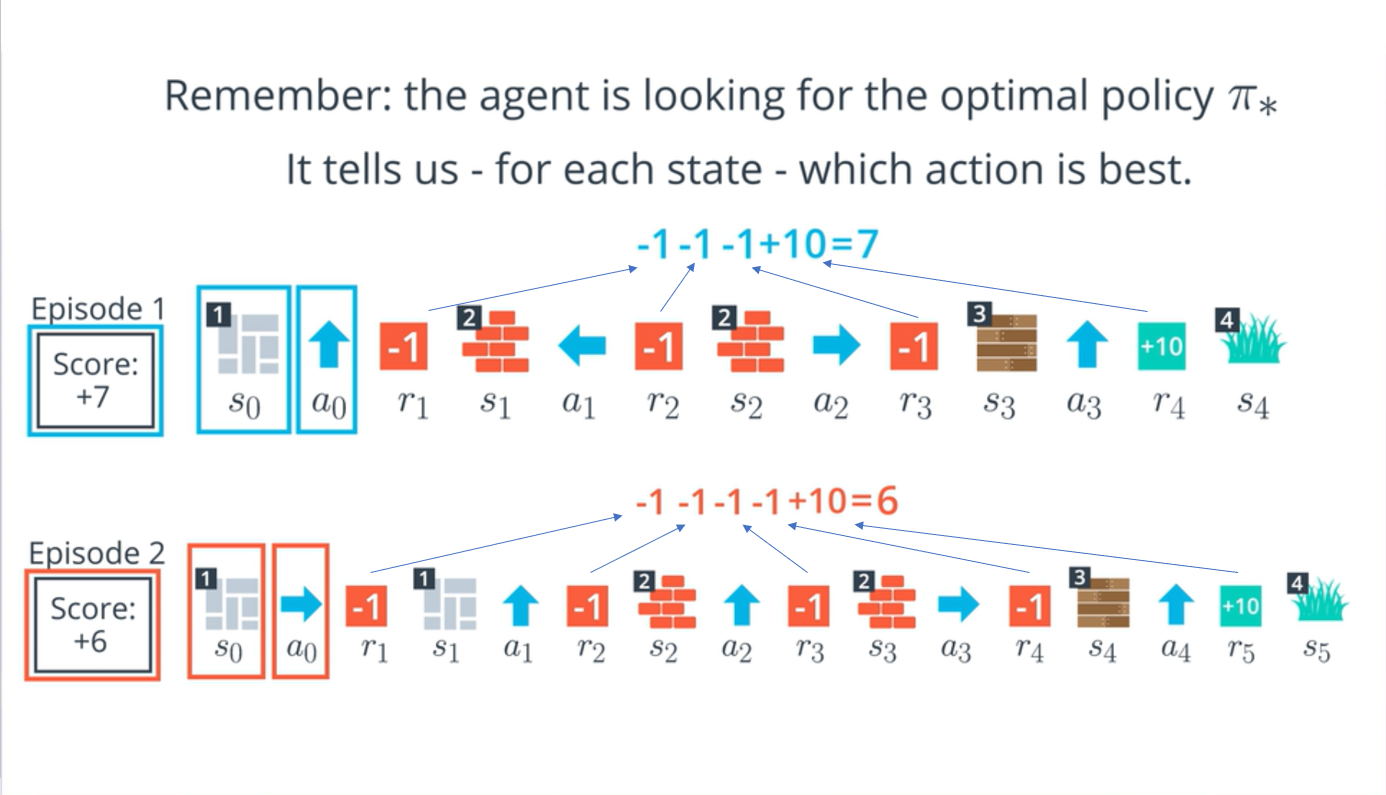
# 

# 

# M**ONTE CARLO CONTORL ALGLORITHM:**

# 





To Truly understand the environment, the agent needs more episodes:

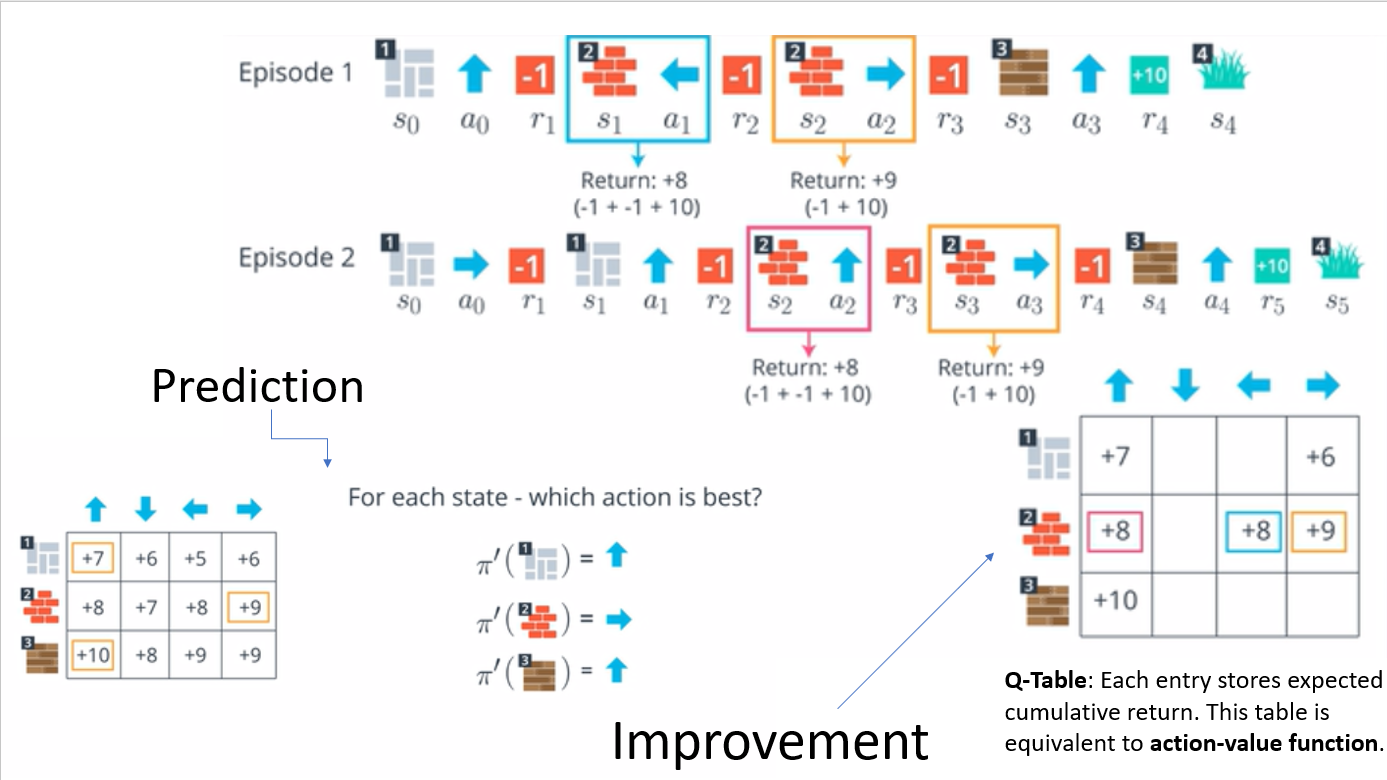
* **Reason 1**: The agent hasn’t tried out each action from each state.
* **Reason 2**: The environment’s dynamics are **stochastic.**

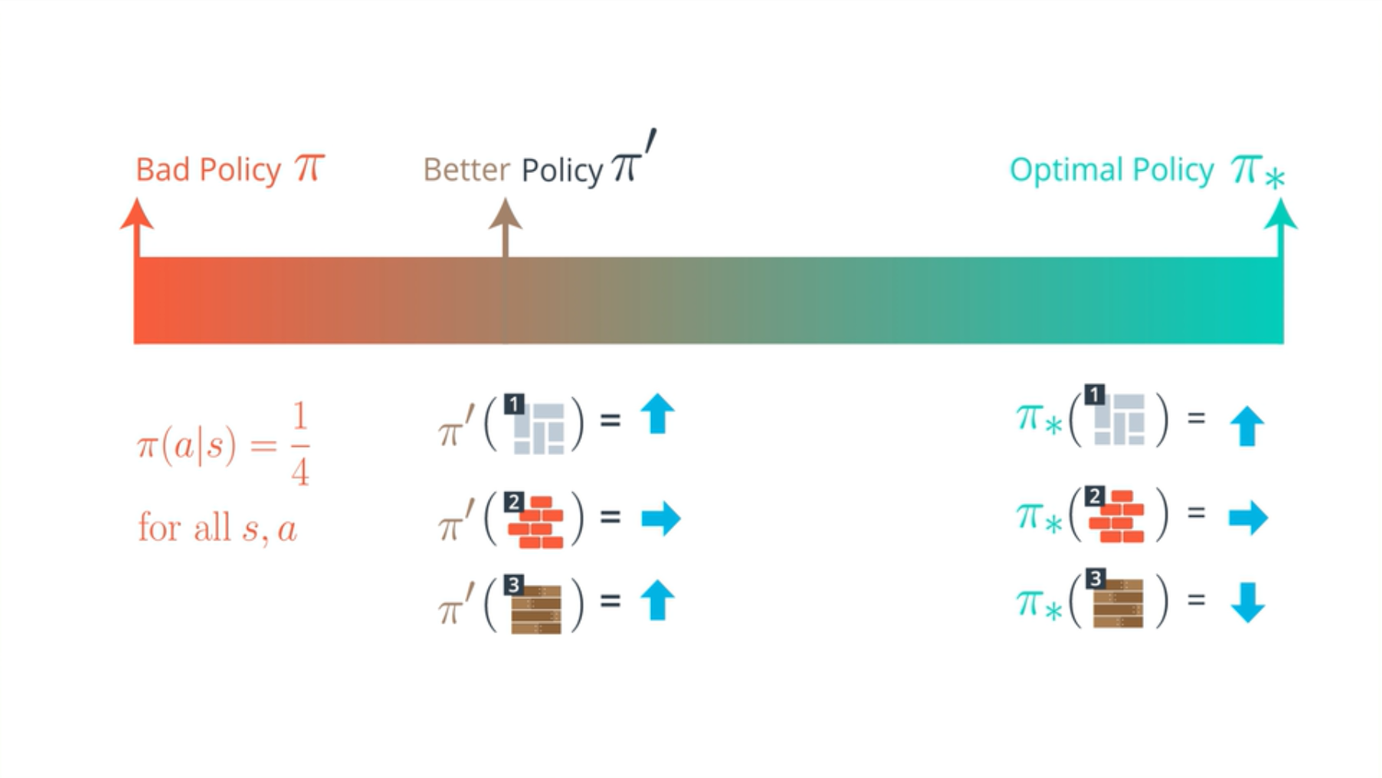
# This is the foundation principle of Monte Carlo Control methods.

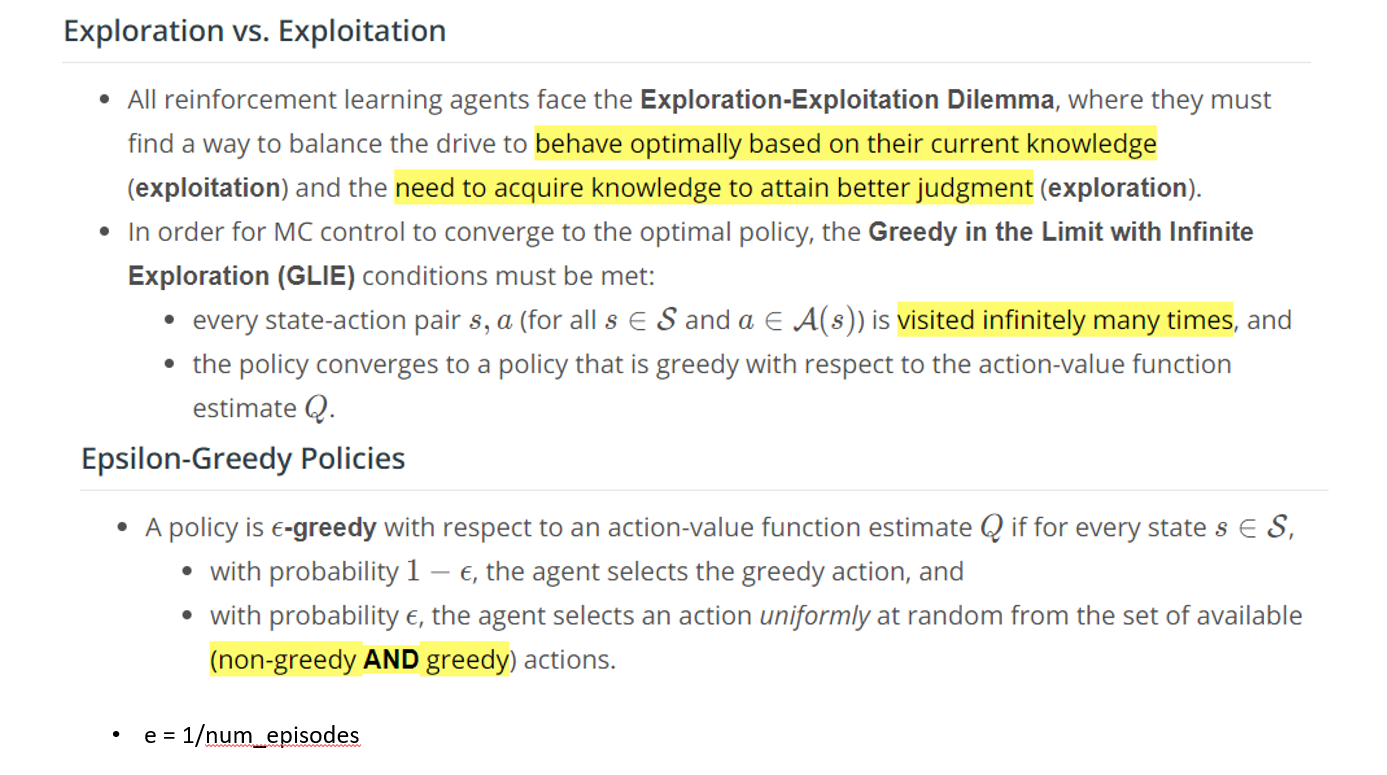
Monte Carlo methods - even though the underlying problem involves a great degree of randomness, we can infer useful information that we can trust just by collecting a lot of samples.

The approach is divided into 2 parts:

Prediction andImprovement

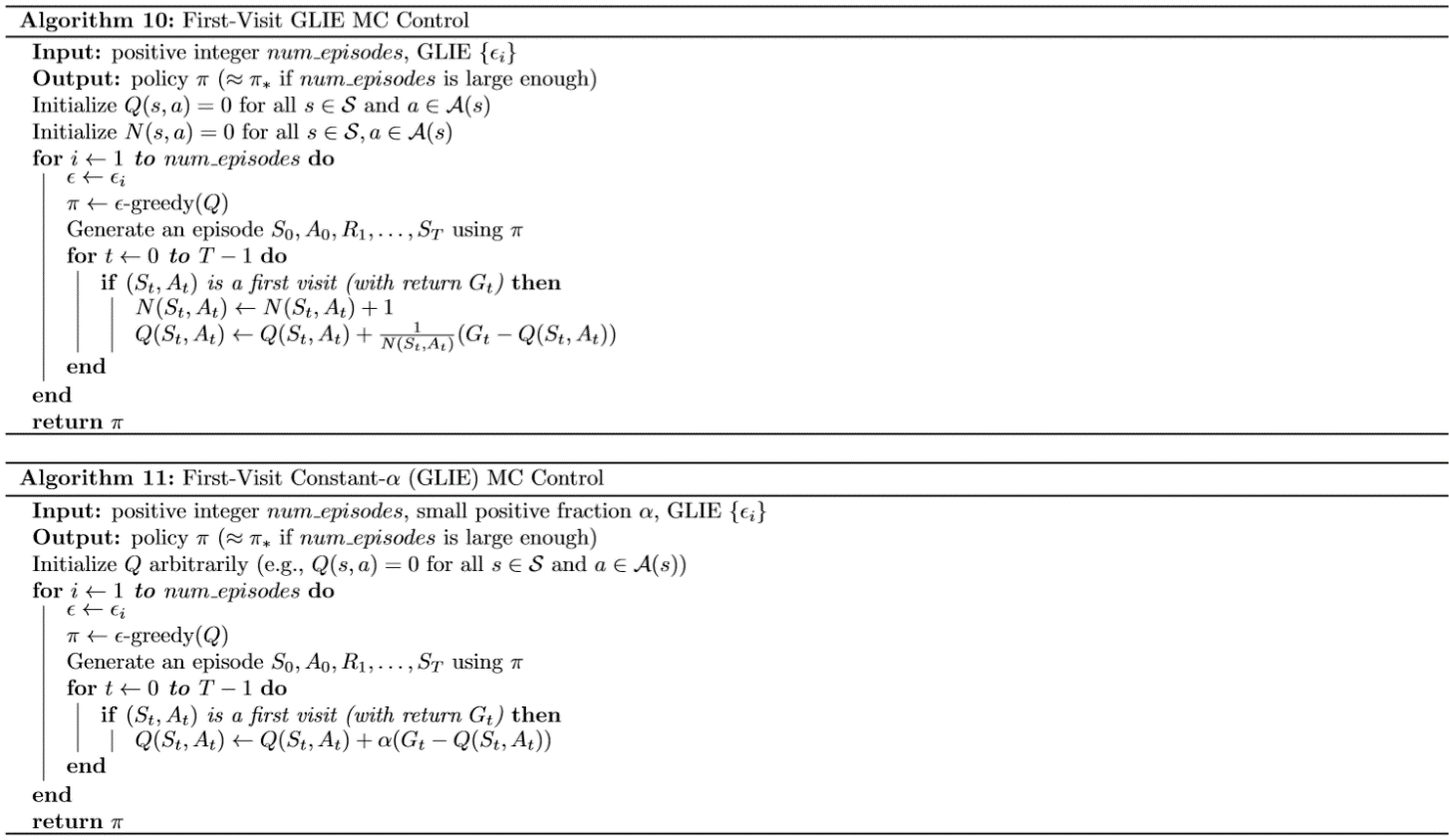


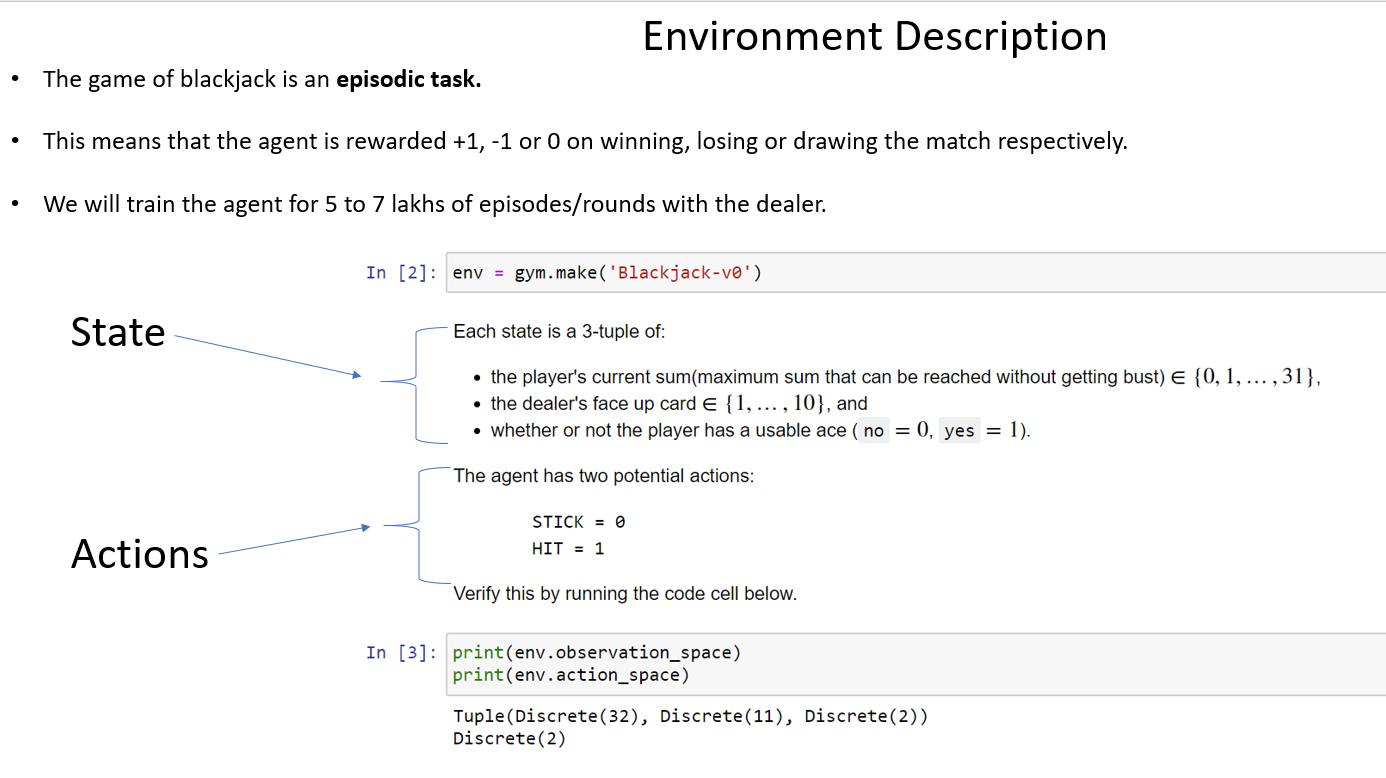




# **IMPLEMENTATION**

**APPLYING MONTE CARLO CONTROL ALGLORITHM ON BLACKJACK:**

****

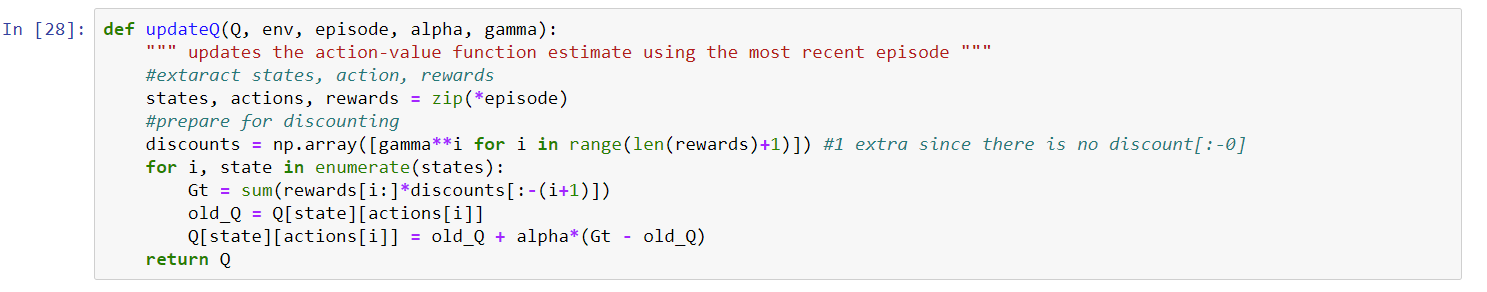




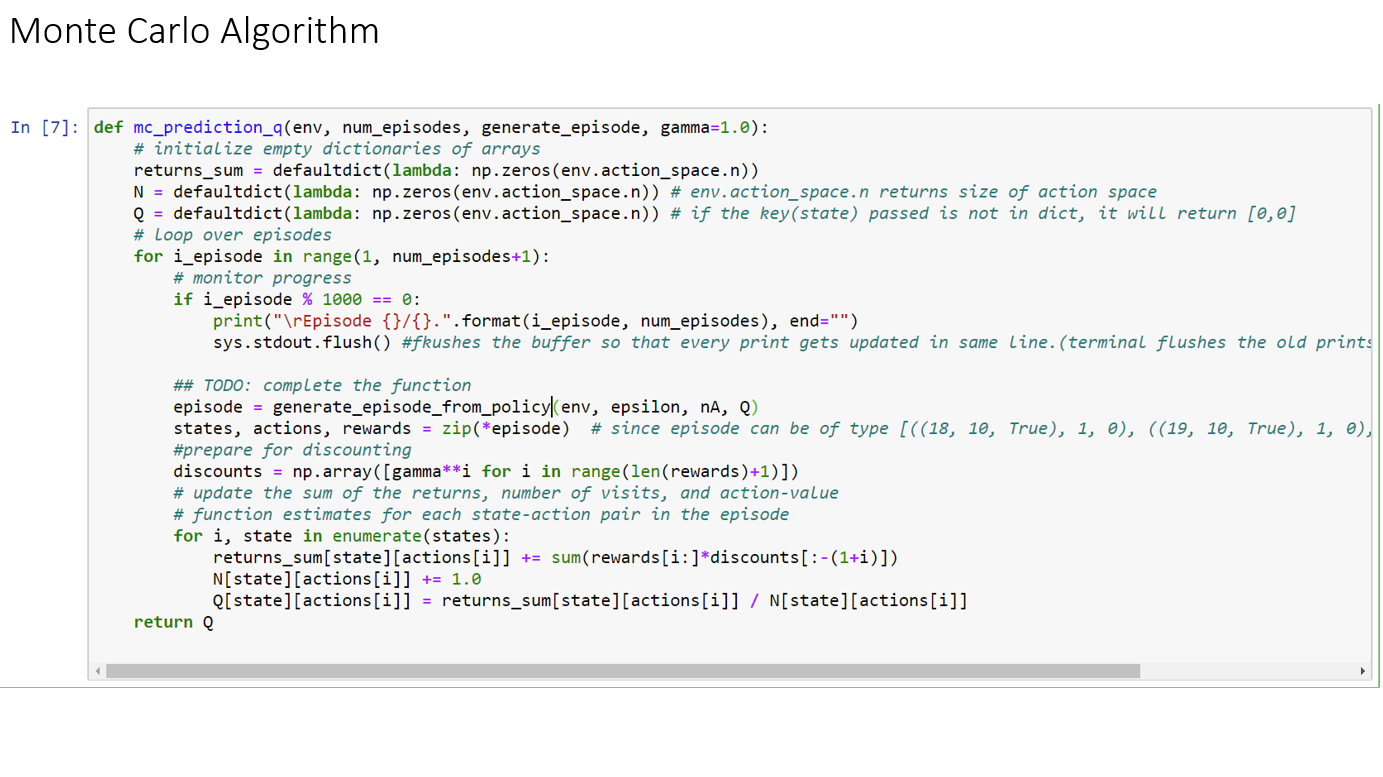


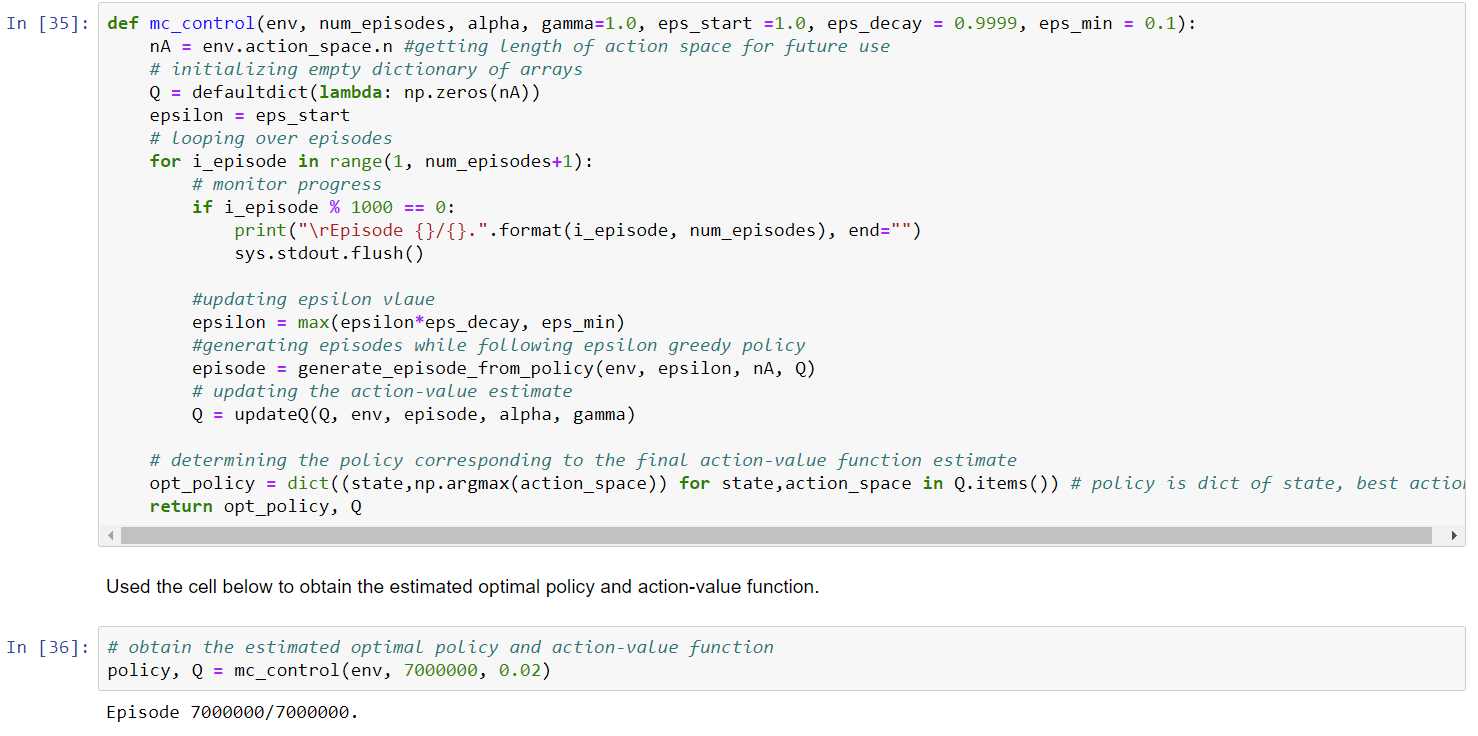
The **get\_probs()** function returns action according to epsilon greedy policy.

The **generate\_episode\_from\_policy()** function is responsible for generating episode using **env.step()** method the input to which, is the output of **get\_probs().**

****

The **updateQ()** updates the Q-table using the most recent episode returned by **generate\_episode\_from\_policy()** method.

****

****

Above is the function where everything fits into a full working agent. The **mc\_control()** function initializes a Q-table and then trains the agent for num\_episodes performing the steps described in Monte Carlo control algorithm using the helper functions described above.

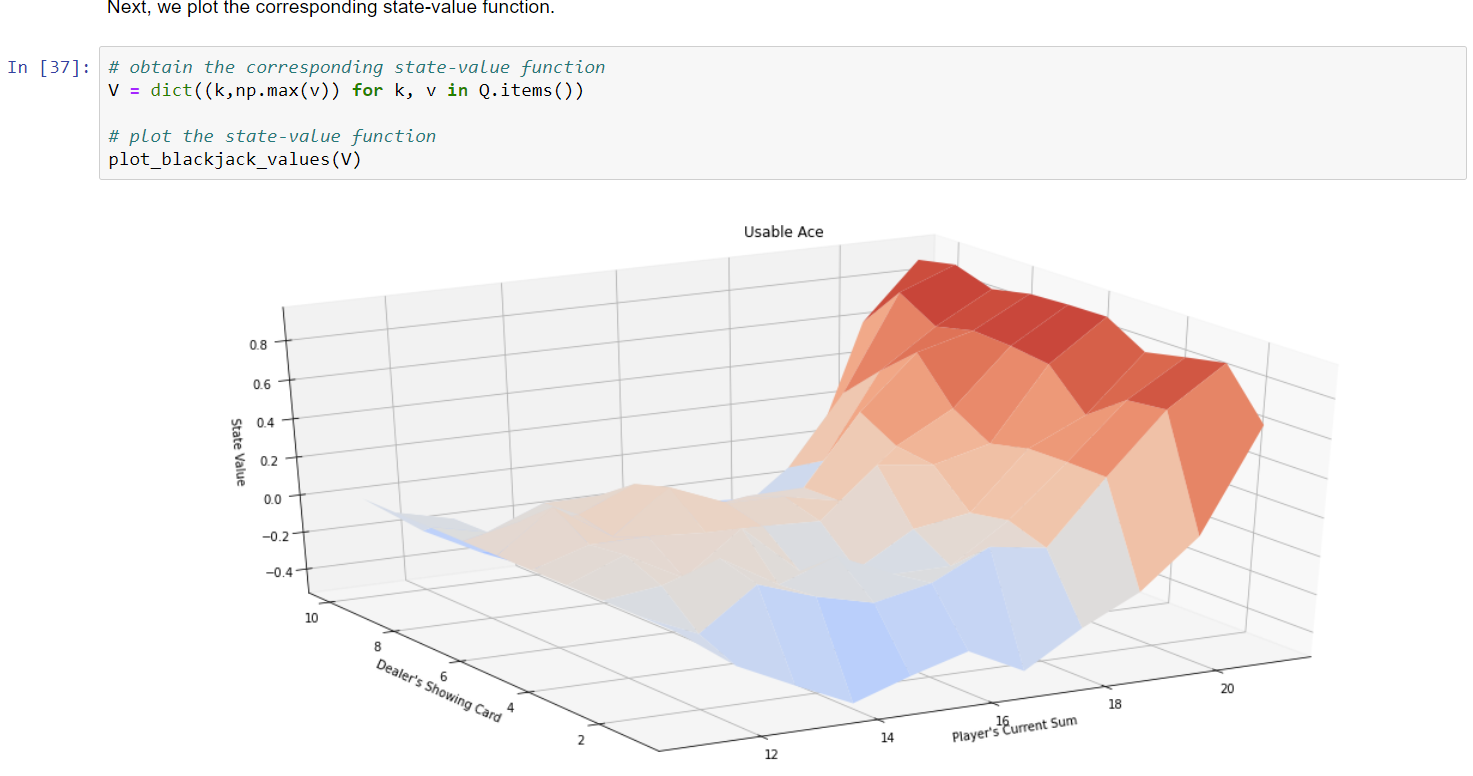
The code is filled with comments and function descriptions which will help in understanding the flow of the program. Reader is advised to start from mc\_control() method and then look into the helper functions whenever they are required.

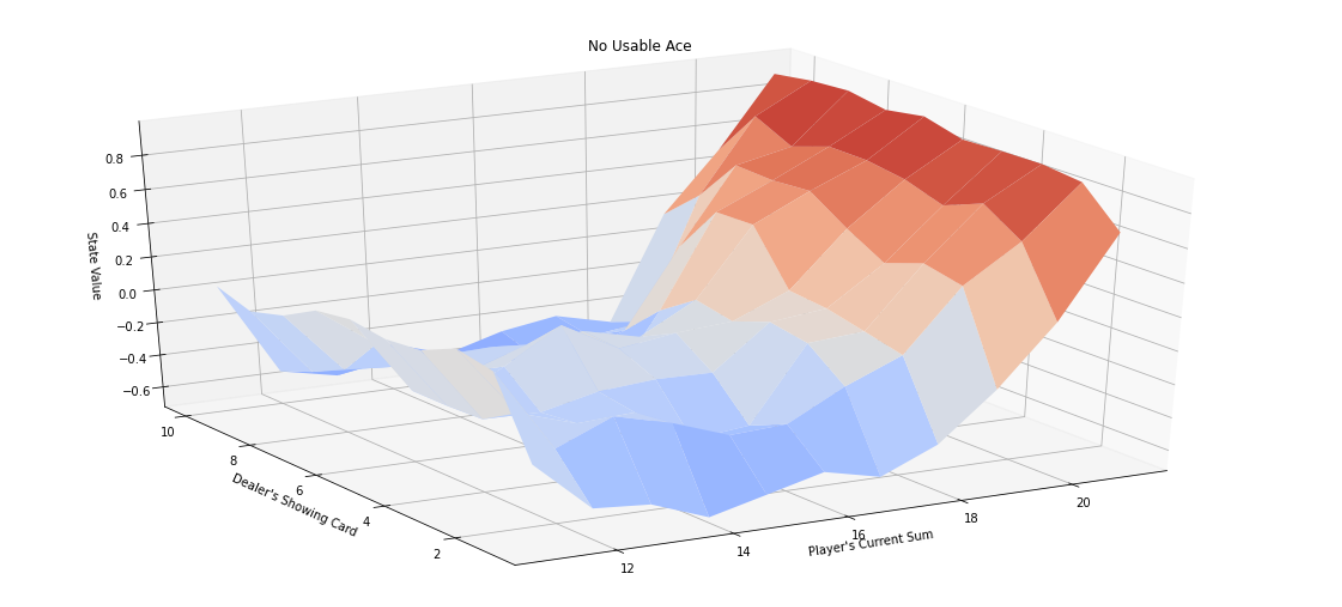
**HYPERPARAMETERS USED:**

* **num\_episodes** : 7000000 # no. of rounds of blackjack
* **alpha**: 0.02 # learning rate
* **gamma**: 1.0 # discount factor
* **eps\_start**: 1.0 # starting support exploration (equiprobable random policy)
* **eps\_min**: 0.1 # at every episode eps is multiplied with this to favor exploitation
* **eps\_decay**: 0.9999

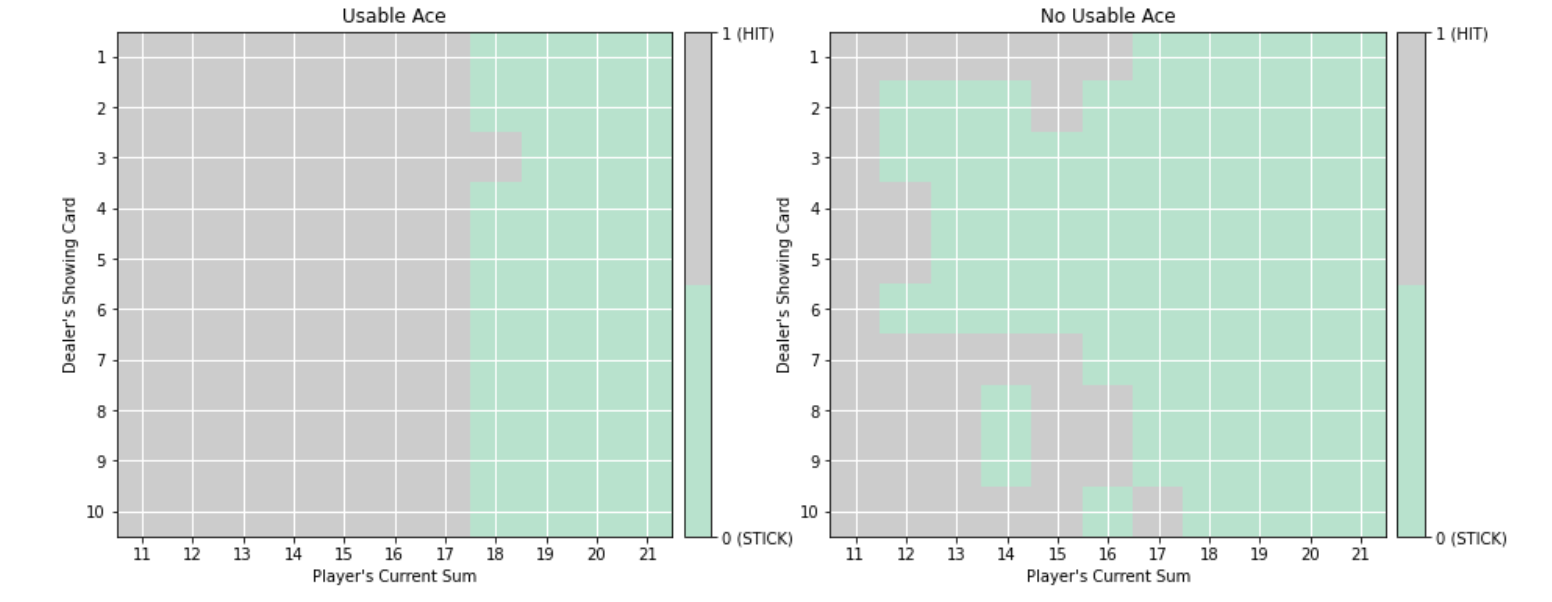
**RESULT AND ANALYSIS**

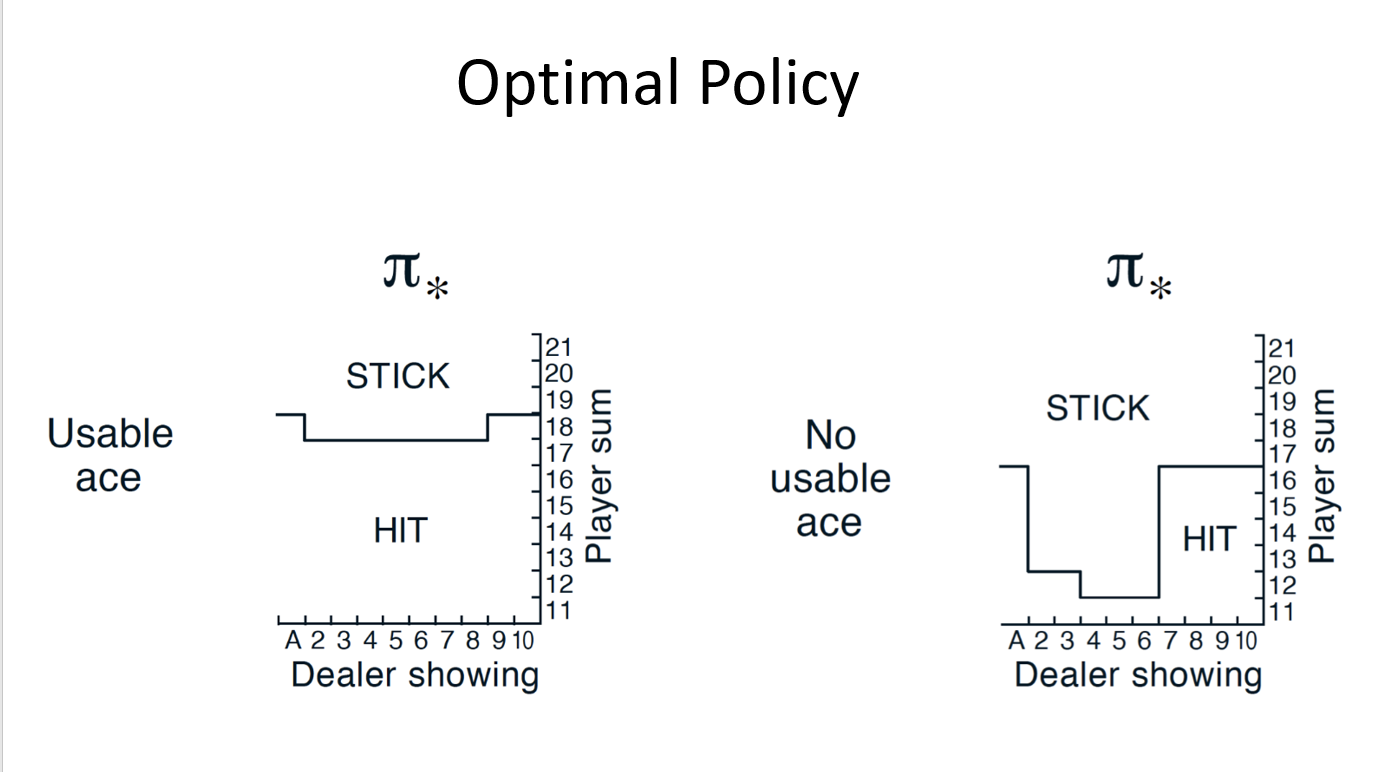
**PLOTTING STATE -VALUE FUNCTION.**

****

****

In the above figures, State-value is the average reward when the agent takes any action for that state. As you can see, the chances of getting positive average reward is lower when our total is between 12 and 17 but are very high for 18 above. When our sum is below 12, our win is totally dependent on the hit card and thus, is a chance, resulting in mean reward of 0.

**OUR OUTPUT POLICY:**

****

The **true** optimal policy 𝜋∗π∗ can be found in Figure 5.2 of the [textbook](http://go.udacity.com/rl-textbook) (and appears below).

As we can see, the agent has learned policies very close to the true optimal policies. For example, if we have usable ace, the agent choses to hit till we have sum 17 as we can use ace as 1 too in case, we get a high card. But when we don’t have usable ace, when the dealer’s face up card is from 2 to 7 we can stick if our cards sum crosses 12. This is because we have a higher probability of going bust on hitting than the dealer having a high face down card.

**CONCLUSION**

After playing 7 million blackjack games, without hardcoding, the agent was able to derive optimal policy. The Agent has learned to play Blackjack.

In future, I am planning to use artificial neural networks as function Q-value function approximator (DQN algorithm) to apply reinforcement learning to more complicated environments.

**REFERENCES**

1. Reinforcement Learning An Introduction (second edition) Richard S Sutton and Andrew G. Barto [textbook](http://go.udacity.com/rl-textbook)
2. Articles about reinforcement learning.