Machine Learning Engineer Nanodegree

Capstone Project

Bogdan Ionescu

7th January 2018

I. Definition

Project Overview

Blackjack, also known as twenty-one, is a comparing card game between usually several players and a dealer, where each player in turn competes against a dealer, but players do not play against each other. It is played with one or more decks of 52 cards, and is the most widely played casino banking game in the world. The objective of the game is to beat the dealer in one of the following ways:

- Get 21 points on the player's first two cards (called a "blackjack" or "natural"), without a dealer blackjack
- Reach a final score higher than the dealer without exceeding 21
- Let the dealer draw additional cards until their hand exceeds 21

The purpose of this project is to teach a machine how to play Blackjack 1 vs 1 against the dealer.

Problem Statement

Train a model using Reinforcement Learning to play Blackjack from the perspective of a player. The model will build an optimal policy by playing repeatedly against a dealer, and recording the rewards received for actions taken in different states during the training. The model will use the learned policy during testing phase, the scope being to achieve the best possible win percentage against dealer.

Metrics

The metric used for evaluating the model is win percentage. The formula that will be used for calculating it is:

$$\text{Win rate} = \frac{Number\ of\ games\ won}{Total\ number\ of\ games\ played}$$

This metric is appropriate for evaluating this model as it best illustrates the main point of interest for any casino player - the chance of winning a game in order to win money.

II. Analysis

Data Exploration

The building blocks for this type of project are states, actions, rewards and transitions. Each of these blocks is explained below:

1. States

The setup (of the game) prior to taking any action defines a state.

In this project, states are uniquely identified and stored as a tuple of the following 4 parameters:

- The sum of the cards in hand
- A flag that indicates if the hand is hard or soft (if the hand is hard, the flag is set to 0; if the hand is soft, the flag is set to 1)
- The number of cards in hand
- The value of the dealer card that is visible to the player

2. Actions

While in a state, throughout the game, the agent needs to take decisions in order for the game to progress. The decisions are also known as actions.

In this project, there are 2 actions that are available to the agent:

- Hit: the agent draws a new card from the deck and adds it to the ones in hand

- Stand: the agent ends its turn, the sum of the cards in hand is the one used for computing the outcome of the game

3. Rewards

The reward is an interpreter that tells the agent if the action taken in a state was good or bad, in order to help the agent learn what it should or shouldn't do if it encounters the same state again in the future.

Rewards have numerical value and are either positive or negative:

- Positive rewards are given when taking an action that increases the chances of the agent to win the game
- Negative rewards are given when taking an action that either decreases the chances for the agent to win the game or for losing the game

Rewards are computed using a fixed value (+2 for positive rewards and -2 for negative rewards), to which a random value from -1 to 1 is added. Therefore, the range for the rewards is between -3 and 3.

4. Transitions

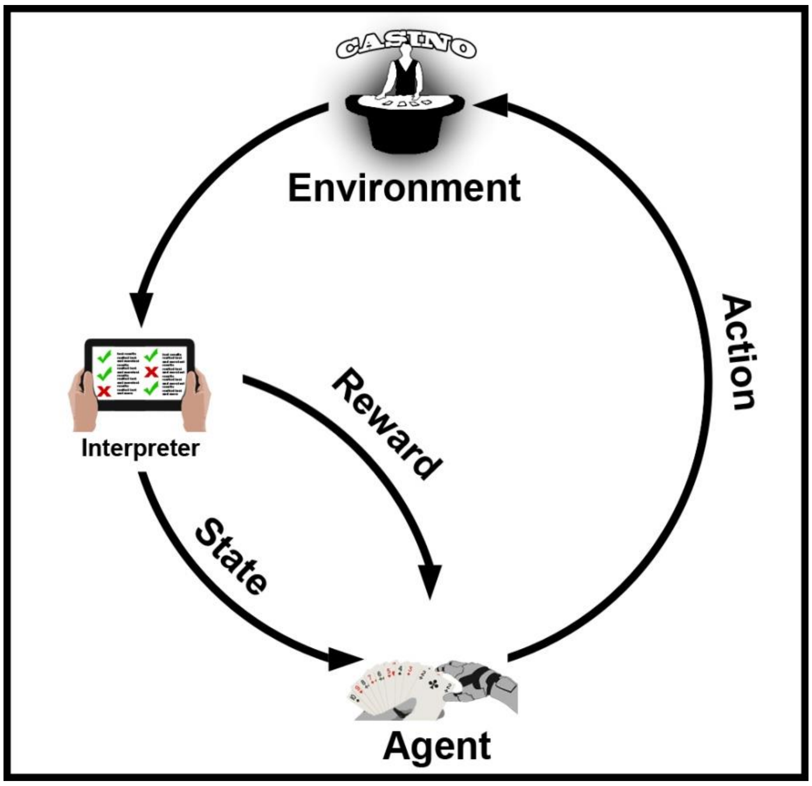
A transition is the process through which a state changes to another state.

In this project, a transition occurs any time the agent chooses the 'Hit' action as this determines the agent to arrive in a new state.

When choosing the 'Stand' action, the state remains the same for the agent and the outcome of the game is computed. Therefore, 'Stand' action does not trigger a transition.

Exploratory Visualization

The following graph depicts the flow of the game:



An initial state is generated when the game starts. The agent is then prompted to choose an action - which can either transition to a new state (in case the action is 'Hit') or it can trigger a function that computes the outcome of the game (in case the action is 'Stand').

Regardless of the action chosen, the interpreter computes a reward based on the state in which the agent was and the chosen action. The information is provided then to the agent.

The flow from above is repeated until the game ends; the flow is restarted whenever a new game starts.

Algorithms and Techniques

The core function behind this project is a Q-learning technique which is used to find an optimal action-selection policy for any MDP (Markov decision process).

It works by learning an action-value function, often denoted by Q(s, a), which ultimately gives the expected utility of taking a given action *a* in a given state *s*, and following an optimal policy thereafter.

A policy, often denoted by π, is a rule that the agent follows in selecting actions, given the state it is in. When such an action-value function is learned, the optimal policy can be constructed by simply selecting the action with the highest value in each state.

One of the strengths of Q-learning is that it is able to compare the expected utility of the available actions without requiring a model of the environment.

It has been proven that for any finite MDP, Q-learning eventually finds an optimal policy, in the sense that the expected value of the total reward return over all successive steps, starting from the current state, is the maximum achievable.

In this project, the learning process is accomplished by creating a Q-table which stores states as they are encountered while playing games and the pairs of action-rewards for each state. The Q-table is a dictionary that has the following format:

{(state_1) : {'Action_1' : Reward_value, 'Action_2' : Reward_value},

(state_2) : {'Action_1' : Reward_value, 'Action_2' : Reward_value},

...}

The format of a state is defined in the Data Exploration section above.

The mapping for actions in Q-table is the following:

- Action_1 is always 'Hit'
- Action_2 is always 'Stand'

The format of the rewards is described in the Data Exploration section above.

The logic behind the reward system is the following:

For hard hands, 'Hit' is rewarded positively as long as it brings the player closer to 21 points without bursting and negatively if player goes over 21 points. Similarly, 'Stand' is rewarded positively as long as it would prevent the player from bursting over 21 points, and negatively if standing prevents the player from getting closer to 21 points without bursting (in case a card would have been drawn).

For soft hands, the logic of the reward function is as follows:

- if the chosen action is 'Hit' and by drawing the card the hand is still soft, the reward is positive; if the chosen action is 'Stand' and by drawing a card the hand would still be soft, the reward is negative

- if drawing a card would make the hand go from soft to hard, the reward function launches a loop in which it makes the player draw cards until the value of the hand becomes greater than the initial value of the soft hand in order to determine if the reward should be positive or negative for the action chosen initially:

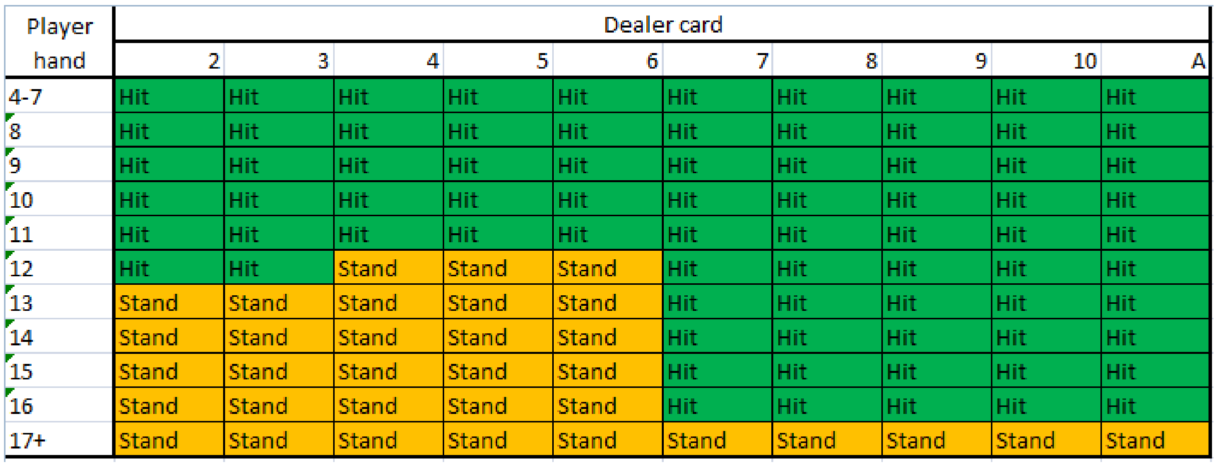
--after the value of the hand becomes greater than the initial value of the hand, if it is below 22 points, the reward is positive if the initial action was 'Hit' and negative if the initial action was 'Stand'

--if the hand value ends up bursting over 21 points, the reward is negative if the initial action was 'Hit' and positive if the initial action was 'Stand'

Benchmark

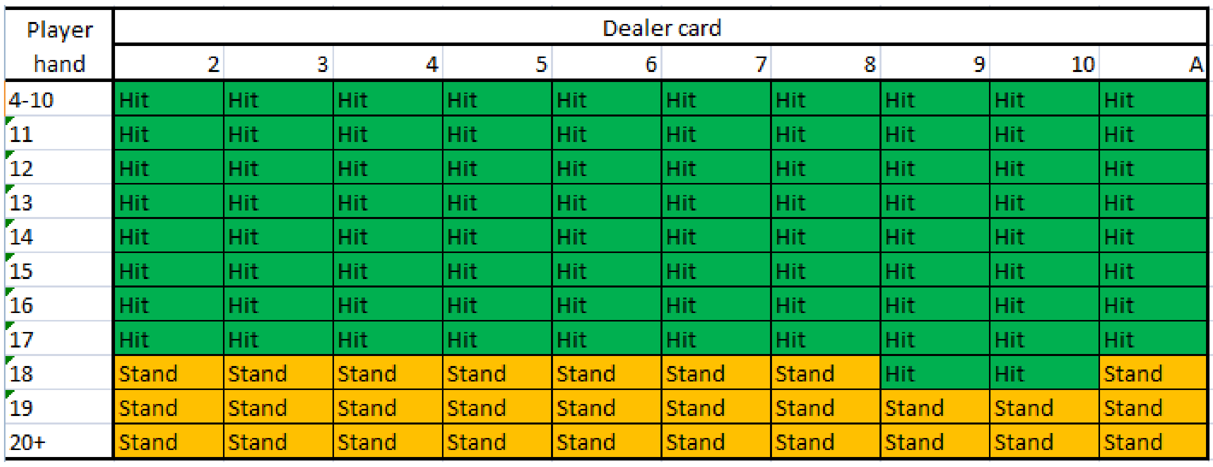
A benchmark depicting the basic strategies for single-deck blackjack where dealer stands on soft 17:

- When player hand is hard:



Player hand	Dealer card									
	2	3	4	5	6	7	8	9	10	A
4-7	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
8	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
9	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
10	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
11	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
12	Hit	Hit	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
13	Stand	Stand	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	Stand

- When player hand is soft:



Player hand	Dealer card									
	2	3	4	5	6	7	8	9	10	A
4-10	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
11	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
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	Hit	Hit	Hit	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

Note: In Blackjack, Aces can be count either as 1 or as 11 points - whichever helps the player most. A hand is considered soft whenever it contains an Ace that can be counted as 11 points without the player bursting 21 points.

By using the strategy illustrated in the tables above for Hard hands and Soft hands, the win rate for the player is 42.42%.

III. Methodology

Data Preprocessing

The model learns on its own how to play Blackjack by repeatedly playing the game and storing in a Q-table with all states encountered and rewards received for taking actions while in the states.

Since the model does not require any dataset, no data preprocessing is necessary.

Implementation

The implementation of this project can be split in 2 parts:

- A part referring to the development of Blackjack game mechanics
- Another part for implementing the mechanisms required for the player to learn through playing

For the first part the following components were developed:

- a Card class that is used for creating card instances. The functions implemented for this class include printing the suit and rank of the card, to interrogate the suit or the rank of a card, to validate if a combination of suit and rank exist

- a Deck class that is used for creating a deck of cards. The functions implemented for this class include creating a deck containing card instances of all the combinations of suits and ranks, shuffling the deck, dealing the top card from the deck, printing the deck

- a Hand class that is used for creating the player and dealer hands. The functions implemented for this class include printing the content of the hand, adding a card to the hand, computing the value of the hand, computing the number of aces in hand, computing the number of cards in hand

- create an event handler for dealing cards at the beginning of each game. This function creates a deck instance and shuffles it, creates 2 Hand instances - one for player and one for dealer, deals in turn 2 cards to both player and dealer which represent the starting hands

- create an event handler for hit action. This function adds a new card to the player hand

- create an event handler for stand action. This function is called when the player chooses to stand and is also used for triggering the game play for the dealer - as it only starts playing after player decides to stand. This function is also used for evaluating the outcome of the game (win, lose or tie) based on the final values of the player hand and dealer hand

For the second part the following components were developed:

- build_state function which is used to compute the current state of the player based on 4 parameters: the sum of the value of the cards in player hand, a flag which indicates if the hand is hard or soft, the number of cards in player hand, the dealer card that is visible to the player

- createQ function that is used for creating default values for each action for all new states that are encountered

- choose_action function that is used to compute what action the player should take next. The action is chosen randomly during training phase; in testing phase the action is chosen based on which provides the biggest reward for the given state

- get_action function that is used to search for the action with a specific Q-value while in a state. If both actions have the same reward, the action is chosen randomly

- get_maxQ function is used to find the maximum Q-value of all actions in the state

- learn function is used to update the Q-value of actions in states

- reward function that is used for computing the reward when taking an action. This determines if the hand is hard or soft and based on this, it triggers separate functions for computing the reward

- hard_reward function is triggered for computing the value of the reward for hard hands

- soft_reward function is triggered for computing the value of the reward for soft hands

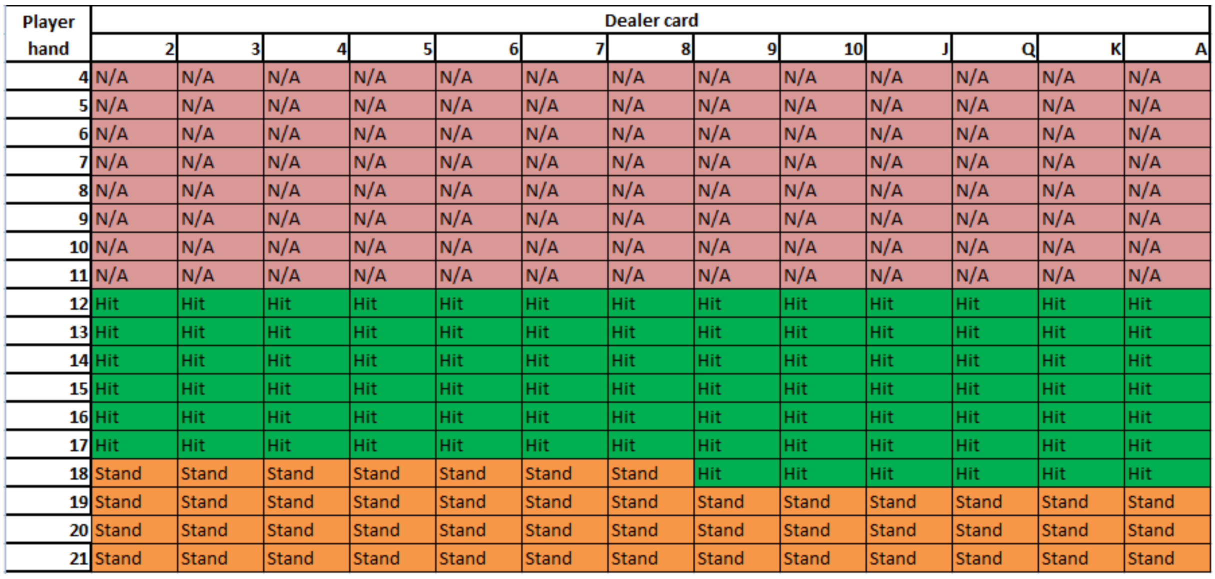
- run function is used for playing a game of Blackjack. It triggers various functions mentioned previously for starting the game, creating states, choosing actions, computing rewards and updating the values in the Q-table

There were no particular complications in developing any of the components from the 2 parts mentioned above, as each component was designed as simple as possible.

The most challenging part was to think in general how the reward function should work - which was detailed in the previous section - and what features to track in each state. However, once the logic of the reward system and the format of the states were defined, the implementation was straightforward.

Refinement

The initial implementation of the reward system was different for the soft hands - it was teaching the model the following logic:



Player hand	Dealer card												
	2	3	4	5	6	7	8	9	10	J	Q	K	A
4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
6	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
10	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
11	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
12	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
14	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
15	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
16	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
17	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit	Hit
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
21	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

With a large enough number of training trials, this approach was leading the model towards rigid results - similar to hard coding the actions from above directly in the player behaviour.

In order to overcome this situation, the reward system was updated for soft hands so that it gives the model full flexibility in learning what would be best to do in any soft hand state.

This was achieved by modifying the reward system so that:

- if action taken is 'Hit' and by drawing a card the hand is still soft, the reward is positive as it brings the player closer to 21 without bursting; 'Stand' action in this case would be rewarded negatively

- if action taken is 'Hit' or 'Stand' and by drawing a card the hand would become hard, the reward is computed after running a loop where player draws cards until the hand value becomes greater than the initial soft hand value. Then:

    -- if the value is less than 22, 'Hit' action is rewarded positively and 'Stand' action is rewarded negatively

    -- if the value is greater than 21, 'Hit' action is rewarded negatively and 'Stand' action is rewarded positively

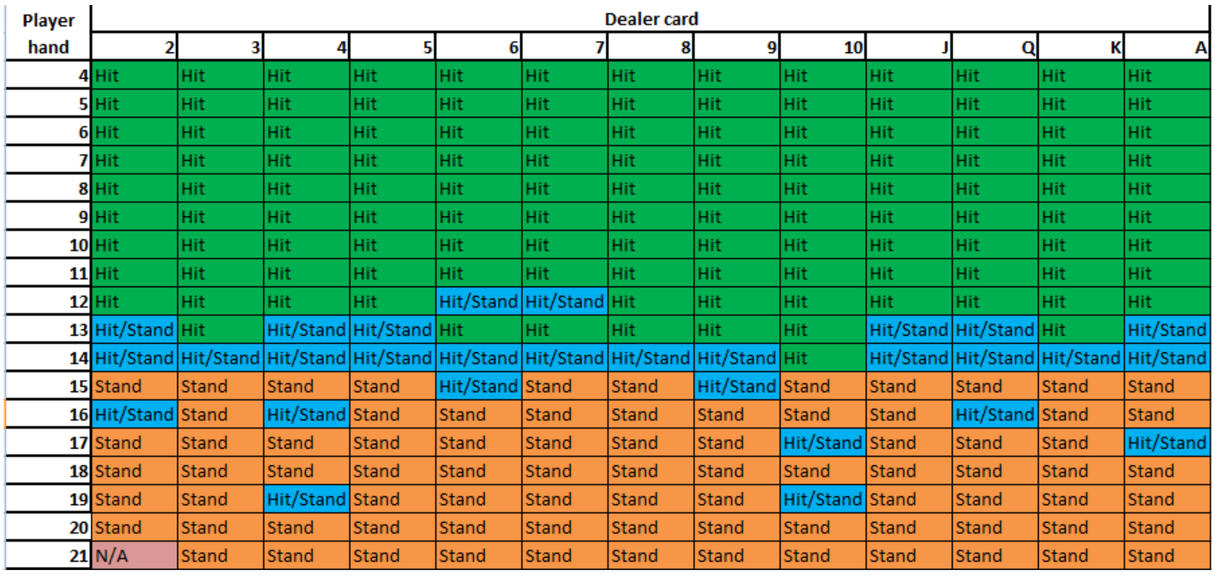
IV. Results

Model Evaluation and Validation

The win rate of the final model (with the modified soft reward function) for 100.000 testing trials after 1.000.000 training trials was 42.514%.

The Q-tables that the model learned can be summarized with the following 2 tables:

-- for hard hands



Player hand	Dealer card												
	2	3	4	5	6	7	8	9	10	J	Q	K	A
4	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
5	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
6	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
7	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
8	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
9	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
10	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
11	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
12	Hit	Hit	Hit	Hit	Hit/Stand	Hit/Stand	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit/Stand	Hit	Hit/Stand	Hit/Stand	Hit	Hit	Hit	Hit	Hit	Hit/Stand	Hit/Stand	Hit	Hit/Stand
14	Hit/Stand	Hit/Stand	Hit/Stand	Hit/Stand	Hit/Stand	Hit/Stand	Hit/Stand	Hit/Stand	Hit	Hit/Stand	Hit/Stand	Hit/Stand	Hit/Stand
15	Stand	Stand	Stand	Stand	Hit/Stand	Stand	Stand	Hit/Stand	Stand	Stand	Stand	Stand	Stand
16	Hit/Stand	Stand	Hit/Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Hit/Stand	Stand	Stand
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Hit/Stand	Stand	Stand	Stand	Hit/Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Hit/Stand	Stand	Stand	Stand	Stand	Stand	Hit/Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
21	N/A	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

The Hit/Stand situations in the table above are explained below - order is left to right, top to bottom:

- If the sum of the cards in hand is 12, number of cards in hand is 5 and the dealer card that is visible to the player is either '6' or '7' - model chooses to 'Stand'; Model chooses to 'Hit' in all other situations where sum of the cards is 12
- If the sum of the cards in hand is 13, number of cards in hand is 5 and the dealer card that is visible to the player is one of {'2', '4', '5', 'J', 'Q' or 'A'} - model chooses to 'Stand'; Model chooses to 'Hit' in all other situations where sum of the cards is 13
- If the sum of the cards in hand is 14, model chooses to 'Hit' in all cases except the ones mentioned below:

-- the number of cards in hand is 3 and the dealer card that is visible to the player is '3'

-- the number of cards in hand is 4 and the dealer card that is visible to the player is one of the following {'4', '5', '6', 'Q' or 'K'}

-- the number of cards in hand is 5 and the dealer card that is visible to the player is one of the following {'2', '3', '4', '5', 6', '7', '9', 'J', 'K' or 'A'}

-- the number of cards in hand is 6 and the dealer card that is visible to the player is '7'

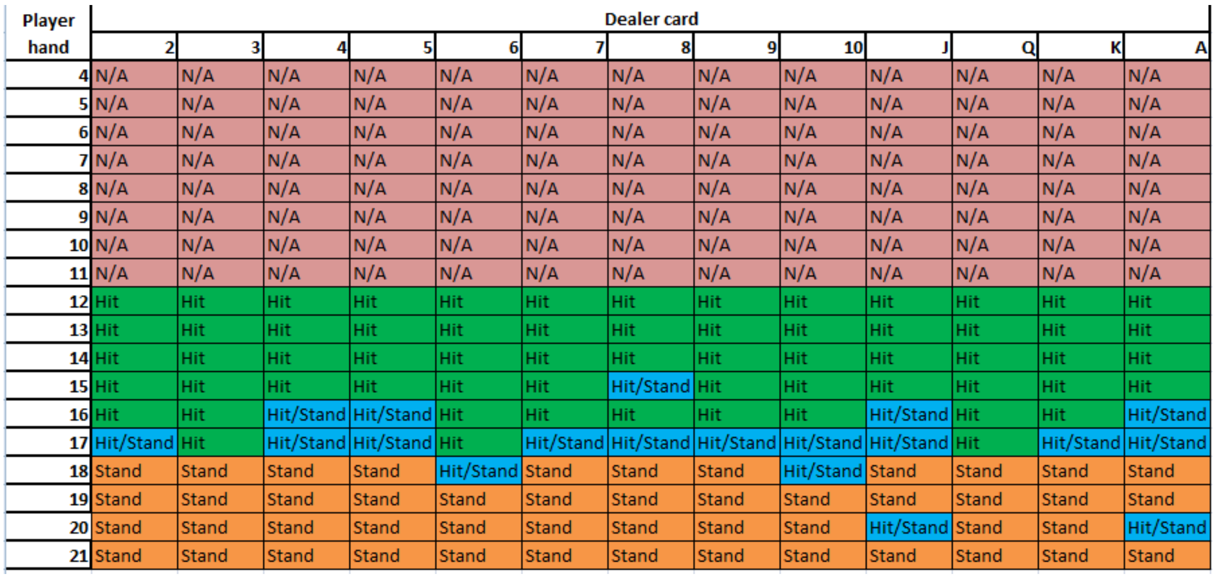
- If the sum of cards in hand is 15, number of cards in hand is 6 and the dealer card that is visible to the player is either '6' or '9' - model chooses to 'Hit'; Model chooses to 'Stand' in all other situations where sum of the cards is 15
- If the sum of cards in hand is 16, model chooses to 'Stand' in all cases except the ones mentioned below:

-- the number of cards in hand is 5 and the dealer card that is visible to the player is '2'

-- the number of cards in hand is 6 and the dealer card that is visible to the player is either '4' or 'Q'

- If the sum of cards in hand is 17, number of cards in hand is 6 and the dealer card that is visible to the player is either 'A' or '10' - model chooses to 'Hit'; Model chooses to 'Stand' in all other situations where sum of the cards is 17
- If the sum of cards in hand is 19, number of cards in hand is 7 and the dealer card that is visible to the player is either '4' or '10' - model chooses to 'Hit'; Model chooses to 'Stand' in all other situations where sum of the cards in 19

-- for soft hands:



Player hand	Dealer card												
	2	3	4	5	6	7	8	9	10	J	Q	K	A
4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
6	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
10	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
11	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
12	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
14	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
15	Hit	Hit	Hit	Hit	Hit	Hit	Hit/Stand	Hit	Hit	Hit	Hit	Hit	Hit
16	Hit	Hit	Hit/Stand	Hit/Stand	Hit	Hit	Hit	Hit	Hit	Hit/Stand	Hit	Hit	Hit/Stand
17	Hit/Stand	Hit	Hit/Stand	Hit/Stand	Hit	Hit/Stand	Hit/Stand	Hit/Stand	Hit/Stand	Hit/Stand	Hit	Hit/Stand	Hit/Stand
18	Stand	Stand	Stand	Stand	Hit/Stand	Stand	Stand	Stand	Hit/Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Hit/Stand	Stand	Stand	Hit/Stand
21	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

The Hit/Stand situations in the table above are explained below - order is left to right, top to bottom:

- If the sum of cards in hand is 15, number of cards in hand is 4 and the dealer card that is visible to the player is '8' - model chooses to 'Stand'; Model chooses to 'Hit' in all other situations where sum of the cards is 15
- If the sum of the cards in hand is 16, number of cards in hand is 4 and the dealer card that is visible to the player is one of the following {'4', '5', 'J' or 'A'} - model chooses to 'Stand'; Model chooses to 'Hit' in all other situations where sum of the cards is 16
- If the sum of the cards in hand is 17, model chooses to 'Hit' in all other cases than the ones mentioned below:

-- the number of cards in hand is 3 and the dealer card that is visible to the player is one of the following {'2', '4', 'A'}

-- the number of cards in hand is 4 and the dealer card that is visible to the player is one of the following {'4', '7', '8', '10', 'J', 'K' or 'A'}

-- the number of cards in hand is 5 and the dealer card that is visible to the player is one of the following {'2', '5', '8'}

- If the sum of the cards in hand is 18, the number of cards in hand is 5 and the dealer card that is visible to the player is either '6' or '10' - model chooses to 'Hit'; Model chooses to 'Stand' in all other situations where sum of the cards is 18
- If the sum of the cards in hand is 20, the number of cards in hand is 5 and the dealer card that is visible to the player is either 'J' or 'A' - model chooses to 'Hit'; Model chooses to 'Stand' in all other situations where sum of the cards is 20

Justification

The following 2 tables summarize where the 2 models (trained model and benchmark) are the same and where they differ in terms of game strategy for a given state:

-- for hard hands

Player hand	Dealer card												
	2	3	4	5	6	7	8	9	10	J	Q	K	A
4	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
5	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
6	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
7	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
8	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
9	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
10	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
11	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
12	Same	Same	Different	Different	Different	Different	Same	Same	Same	Same	Same	Same	Same
13	Different	Different	Different	Different	Different	Same	Same	Same	Same	Different	Different	Same	Different
14	Different	Different	Different	Different	Different	Different	Different	Different	Same	Different	Different	Different	Different
15	Same	Same	Same	Same	Different	Different	Different	Different	Different	Different	Different	Different	Different
16	Different	Same	Different	Same	Same	Different	Different	Different	Different	Different	Different	Different	Different
17	Same	Same	Same	Same	Same	Same	Same	Same	Different	Same	Same	Same	Different
18	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
19	Same	Same	Different	Same	Same	Same	Same	Same	Different	Same	Same	Same	Same
20	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
21	N/A	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same

-- for soft hands

Player hand	Dealer card												
	2	3	4	5	6	7	8	9	10	J	Q	K	A
4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
6	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
10	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
11	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
12	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
13	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
14	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
15	Same	Same	Same	Same	Same	Same	Different	Same	Same	Same	Same	Same	Same
16	Same	Same	Different	Different	Same	Same	Same	Same	Same	Different	Same	Same	Different
17	Different	Same	Different	Different	Same	Different	Different	Different	Different	Different	Same	Different	Different
18	Same	Same	Same	Same	Different	Same	Same	Different	Different	Different	Different	Different	Same
19	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
20	Same	Same	Same	Same	Same	Same	Same	Same	Same	Different	Same	Same	Different
21	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same

The final model has a win rate slightly higher than the benchmark (42.514% as opposed to 42.42%). Although this performance difference is not enough to make a clear statement about the better model, it shows that the solution is significant enough to have solved the problem.

V. Conclusion

Reflection

The development process behind this projects consisted of implementing the game mechanics for Blackjack in a first phase and to add functionalities that allow to model to learn how to play it in a second phase.

The game mechanics developed included creating a deck of cards that can be shuffled, creating the mechanism for dealing cards to dealer and player, keeping separate track of the player and dealer hands, implementing mechanisms for hitting and standing, implementing functionality for evaluating the outcome of a game and for reinitiating the game.

The additional functionalities developed for making the model learn how to play the game included creating the format of a state, creating a mechanism for choosing an action to take, creating a reward function to evaluate how good is the action taken for the current state, creating a map (Q-table) that stores states and the rewards received for the 2 possible actions.

After all components were developed, the model was trained an arbitrary number of times by repeatedly playing the game when taking random actions and storing the pairs of state-action-reward encountered in the Q-table. After completing the training, the model was tested on a number of games by selecting in each state the action that had the highest reward value based on the information stored in the Q-table.

The approach described above can be used as a general method of solving similar projects.

The most challenging aspects of the projects were to define the format of a state and implementing the reward system (the mechanism that evaluates how good is an action for the current state).

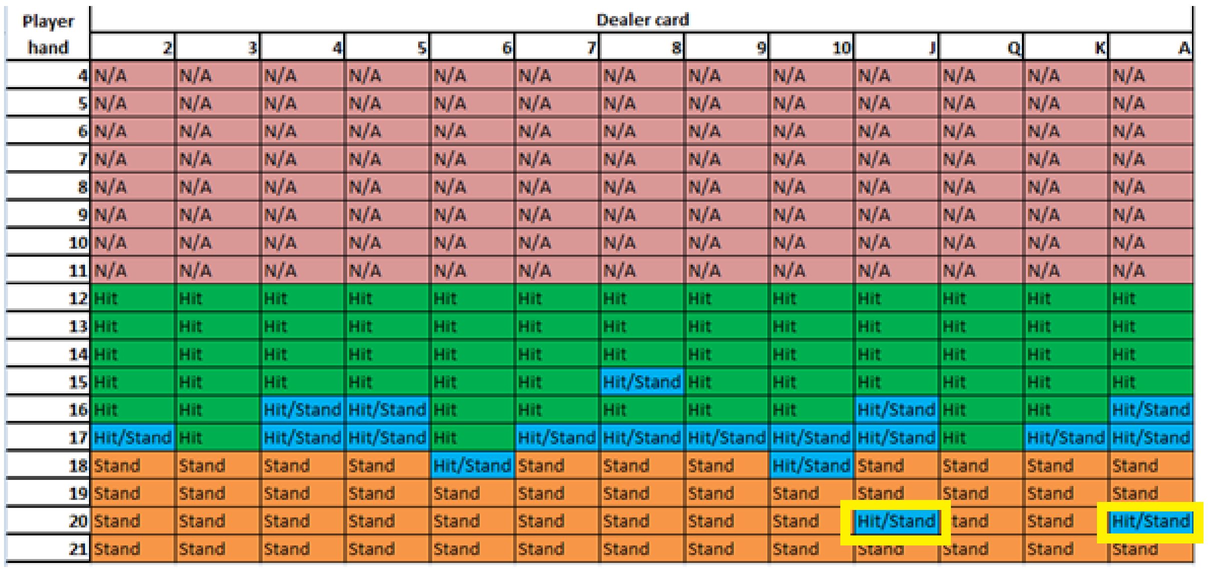
Defining the format of a state was challenging as it required to determine which parameters should be observed and stored while playing the game -- if the number of parameters is very large or if the parameters can take on many different values, this will give the model finer granularity when learning but the number of training trails grows exponentially. If the number of parameters is low or if they can take on only a few values, the model is trained faster but it might be impossible to learn correctly what to do - similar to underfitting.

The challenge to develop the reward system is thinking how the actions are evaluated in regard to the current state. If there is a behaviour hard coded in the reward system (e.g. if the sum of cards in hand is 16 you are always rewarded for hitting and always penalized for standing), with a number of training trials large enough, the behaviour will be transmitted to the model -- this is similar to hard coding the behaviour directly in the model. If the reward function is neutral in regard to the cards in hand, and the evaluation is done only based on whether taking the action in the current state prevents the player from losing, brings the player closer to 21 points or leads to losing by bursting over 21 points, this gives the model greater flexibility when learning.

Improvement

A possible improvement to the model would be to increase the number of training trials. This observation is derived from the entries stored in the Q-table after finishing the training of the model.

For example:



Player hand	Dealer card												
	2	3	4	5	6	7	8	9	10	J	Q	K	A
4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
6	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
10	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
11	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
12	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
14	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
15	Hit	Hit	Hit	Hit	Hit	Hit	Hit/Stand	Hit	Hit	Hit	Hit	Hit	Hit
16	Hit	Hit	Hit/Stand	Hit/Stand	Hit	Hit	Hit	Hit	Hit	Hit/Stand	Hit	Hit	Hit/Stand
17	Hit/Stand	Hit	Hit/Stand	Hit/Stand	Hit	Hit/Stand	Hit/Stand	Hit/Stand	Hit/Stand	Hit/Stand	Hit	Hit/Stand	Hit/Stand
18	Stand	Stand	Stand	Stand	Hit/Stand	Stand	Stand	Stand	Hit/Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Hit/Stand	tand	Stand	Hit/Stand
21	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

If the sum of the cards in hand is 20 for a soft hand, the reward is usually greater for 'Stand' action, than 'Hit'. However, the 2 boxes marked with yellow in the table above had the following corresponding entries in the Q-table for which 'Hit' reward is greater than 'Stand' reward:

((20, 1, 5, 'J'), {'Hit': 1.0813799078961435, 'Stand': 0.0})

((20, 1, 5, 'A'), {'Hit': 1.6970799542218795, 'Stand': 0.0})

This means that when the sum of the cards in hand is 20, the hand is soft, the number of cards in hand is 5 and the dealer card that is visible to the player is 'J', respectively 'A' - the reward is greater if the player chooses to 'Hit' instead of 'Stand'. This is because in both cases (which are very rarely encountered), by randomly choosing 'Hit' the sum of the cards in hand became 21 (which is highly improbable) and therefore 'Hit' action was rewarded positively.

By increasing the number of trials, the model might encounter these states multiple times and determine that it's always better to Stand in both situations, which might also increase the win rate of the model.

References

[1] https://en.wikipedia.org/wiki/Blackjack

[2] https://wizardofodds.com/games/blackjack/strategy/calculator/

[3] https://en.wikipedia.org/wiki/Q-learning