ISYE-6644 Final Project Report

Team 203 Member(s): Stone Hayden

Topic 12: Blackjack

Abstract

A blackjack player must make decisions to maximize his profit while being disadvantaged in 2 areas to the dealer, the dealer shows only one card and makes his decisions after the player. Due to these strategic disadvantages, the player must make his decisions based on highest return moves based on the current game state. This simulation aims to compare the consensus top blackjack strategy versus other strategies. Output analysis methods are used to determine if this top strategy is truly best, and if so, by how large of a margin.

Background

The game of blackjack begins with the player making a wager before any cards are dealt, then the player and dealer are dealt 2 cards each. Both player's cards are visible to all, but the first card dealt to the dealer is hidden and visible only to him. Additionally, the player makes his decisions before the dealer, putting him at a further advantage. However, the dealer has rules he must follow that can guide a player's actions. The dealer is required to keep adding cards until his total exceeds 16, if 17 and above the dealer stays. The player must compare his hand to the dealer's visible card and weigh which action he should take. Available actions to the player are hit, stand, split, or double-down. Taking advantage of splitting and doubling down are ways to offset the dealer's upper hand as these actions are unavailable to the dealer and can increase returns in well-leveraged situations.

This report will detail the methodologies compared in the simulation, the setup of the simulation program, analysis of the simulation's output and what information and can be gained from this, and finally, concluding thoughts on the methodologies tested. One blackjack strategy is widely accepted as providing the highest odds of winning. This simulation is focused on rigorously testing that strategy as the best and comparing it to 5 other strategies with popular (and not so popular) differing thoughts in certain situations.

Simulation Method

Wager Methodologies

This project aims to compare 6 different wagering methods. The first strategy is the consensus best wagering method. The following chart details each strategy:

Strategy	Description
Strategy 1	Consensus optimal blackjack strategy
Strategy 2	Always double down on 11, regardless of dealer's face card
Strategy 3	Hit on 17 if dealer is showing 8 or greater
Strategy 4	Always split a pair of 10's
Strategy 5	Always split any pair
Strategy 6	Always double-down on any pair

Table 1: Wager Method Descriptions

We start by creating an excel file of instructions for Strategy 1 and then modify the excel file to adapt it to each of the other strategies. Strategies two through four are small modifications of Strategy 1 that were found to be some of the more popular questions around what decision to make in each of those decisions. Strategies 5 and 6 are more egregious changes to the strategy where we either split or double-down ever pair, respectively. Since Strategies 2 through 4 are fairly minor changes to Strategy 1, and could be argued to be good decisions, their overall profit and win rate could be similar to Strategy 1, and even higher in some situations. To offset the probable variance in losses, many games will be played, consisting of thousands of hands each, to estimate the winningest overall strategy. Inversely, Strategies 5 and 6 are likely to be obviously bad methodologies and results should show this quickly. Still, these strategies will be simulated along the others to quantify how much worse they are and to remove any small variances. The following chart gives a visual view of Strategy 1 and the decisions to be made by the player compared to the dealer's face card.

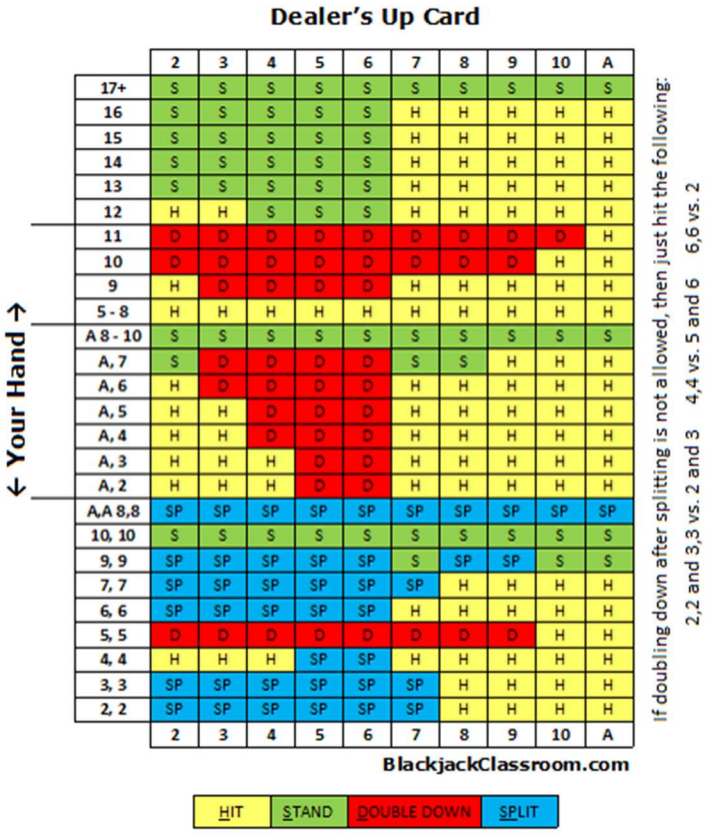


Table 2: Strategy 1 Actions

Comparing Other Strategies to Strategy 1 Chart

- Strategy 2: Continues Double-Down action across entire "11" row, as odds of drawing a 10 are statistically equal to the dealer having a hidden 10
- Strategy 3: Changes "17+" row to Hit if dealer showing 8/9/10/Ace, as player would likely lose
- Strategy 4: Changes "10, 10" row to all Double-Down's as 10 is a solid initial card for 2 hands
- Strategy 5: "A,A,8,8" row and below all become Splits as most pairs call for splits already
- Strategy 6: "A,A,8,8" row and below all become Double-Downs. Not wise but good to test

Python Code Initialization

In order to test the above methodologies, we must create a program in order to automatically run thousands of hands in a manner that is much more efficient than doing so physically. Each strategy is written into a CSV file to be read into a Python script that will tell the script which actions to perform in which situation. Included in this project are two python scripts, BlackJack.py and main.py. Blackjack.py exists only to define functions to be called in main.py; main.py serves as the simulation script.

```python
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from BlackJack import BlackJackTable

# Helper function for integrating CSV-file with BlackJackTable Card format
def replace_card_with_csv(card):
    return {"2": 2, "3": 3, "4": 4, "5": 5, "6": 6, "7": 7, "8": 8, "9": 9, "10": 10, "JACK": 10,
            "QUEEN": 10, "KING": 10, "ACE": "ACE"}[card]

def pandas_helper(x):
    if x == 0:
        return "Push"
    elif x > 0:
        return "Win"
    else:
        return "Lose"

loop = True #T/F
num_iterations = 60
it = 0

# Determine starting values
origin_money = 10000
stake = 100
simulation_rounds = 2000
```

Table 3: main.py Simulation Initialization

The above code is the initialization of the simulation script. This area defines the functions *replace_card_with_csv* and *pandas_helper* that assign number values to card names and define whether an individual hand is a win, loss, or tie. The loop variable is a Boolean variable to determine whether we will play one game per strategy consisting of up to 2,000 hands or if we will repeat that process *num_iterations* amount of times to compare final profits and win percentages across all strategies many times. The aforementioned “up to 2,000” hands comes into play as the script will determine whether the player does not have any more money left to wager and will end the game early. After the initialization code, the script will then move to either the single blackjack game or multiple blackjack games simulation based on the *loop* variable. The following code block simulates a single blackjack hand.

```python
        for game_round in range(simulation_rounds):
            while not bjt.is_finished():
                # Get the game state and determine dealers card
                game_state = bjt.get_game_state()
                column = strats[strat][str(replace_card_with_csv(game_state['dealer'][0]))]
                actions = []

                # Find the action for each hand being played
                for hand_index, hand in enumerate(game_state['player']):
                    # If the hand is already terminated, append Nothing and continue with next hand
                    if game_state['player'][hand_index][-1] == "TERMINATED":
                        actions.append("Nothing")
                        continue

                    # Get the value of the deck
                    value_to_take = bjt.get_final_value_of_deck(game_state['player'][hand_index])
                    for action_index, action in column.items():
                        # See if there is a combination in the strategy that fits the current hand
                        if "," in action_index and len(game_state['player'][hand_index]) == 2:
                            if (str(game_state['player'][hand_index][0]) == str(action_index.split(",")[0]) and str(
                                    game_state['player'][hand_index][1]) == str(action_index.split(",")[1])) or (
                                    str(game_state['player'][hand_index][0]) == str(
                                action_index.split(",")[1]) and str(game_state['player'][hand_index][1]) == str(
                                action_index.split(",")[0])):
                                actions.append(action)
                                break
                        else:
                            # Else just look at the value and determine action
                            if str(action_index) == str(value_to_take):
                                actions.append(action)
                                break

                bjt.do_action(actions)
```

Table 4: main.py Code for Single Blackjack Game

The code in the above Table 4 begins with reading the maximum number of rounds that are to be played in the game. This snippet of code will pull functions from BlackJack.py to initialize the game state and define which cards are dealt to the player and which are dealt to the dealer. The aforementioned *replace_card_with_csv* functioned is called to assign numerical values to face cards. The script then compares the player's hand versus the dealer's and searches for any strategy actions to take. The script then performs the corresponding action and determines whether the hand is over and if the player or dealer won. The below image shows script actions at the end of each hand.

```
100                 # Apply the return to your current money pool
101                 money += stake * bjt.get_return()
102                 if money <= 0:
103                     money = 0
104                     df_returns = df_returns.append(
105                     {"Hands Played": game_round, "strat": f"Strategy {strat+1}", "dealer": bjt.get_game_state()['player'],
106                      'player': bjt.get_game_state()['player'], 'return': stake * bjt.get_return(), 'total': money},
107                     ignore_index=True)
108                     break
109                 # Write the data to the dataframe for later analysis
110                 df_returns = df_returns.append(
111                     {"Hands Played": game_round, "strat": f"Strategy {strat+1}", "dealer": bjt.get_game_state()['dealer'],
112                      'player': bjt.get_game_state()['player'], 'return': stake * bjt.get_return(), 'total': money},
113                     ignore_index=True)
114
115                 # Reset BlackJackTable
116                 bjt.reset_table()
```

Table 5: main.py End of Hand Recording

The above code finds the total money in or out by comparing the staked wager to the BlackJack.py function *get_return* which determines the wager if any splits or double-downs were made. That money is then added/subtracted from the variable *money* which is the player's betting pile. The script then looks to see if *money* is less than or equal to 0, which determines the player is out of cash and will end the game. At the end of each hand, that hand and current game state is recorded in the dataframe *df_returns*. The following chart lists each variable recorded and gives a description of the variable.

Variable	Description
Hands Played	Number of hands played at each game
Strat	Which strategy is being played
Dealer	Dealer cards per hand
Player	Player cards per hand
Return	Money in/out at each hand
Total	Total money available to be wagered
Profit	Total money currently available minus starting money

Table 6: main.py df_returns Columns

The above variables are appended to *df_returns* at the end of each hand except for *Profit*, which is calculated at the end of each tested strategy by subtracting the starting available amount of money. The code after this is mainly for creating plots to analyze final strategy statistics; these plots will be analyzed below. The code to loop through many iterations of games is similar to the discussed code above so it will not be displayed in this section. The main differences to the multiple games section are adding a loop to play a defined amount of games and an additional dataframe to record final results of each game for each strategy. All code will be provided for any desired additional review.

Output Analysis

Single Game Analysis – Analyze Several Thousand Hands per Strategy

We will test each strategy for several thousand hands to view the profit trend as the number of hands played increases. Additionally, we will view the win, push, and loss rate for each strategy and starting values of player and dealer's hands. The following chart shows the 6 strategies and the profit trend as the number of hands played increases. The parameters for this test are a maximum of 10,000 hands, and $10,000 as the starting available amount to wager. The left chart has a base wager of $50 and the right chart has a base wager of $100.

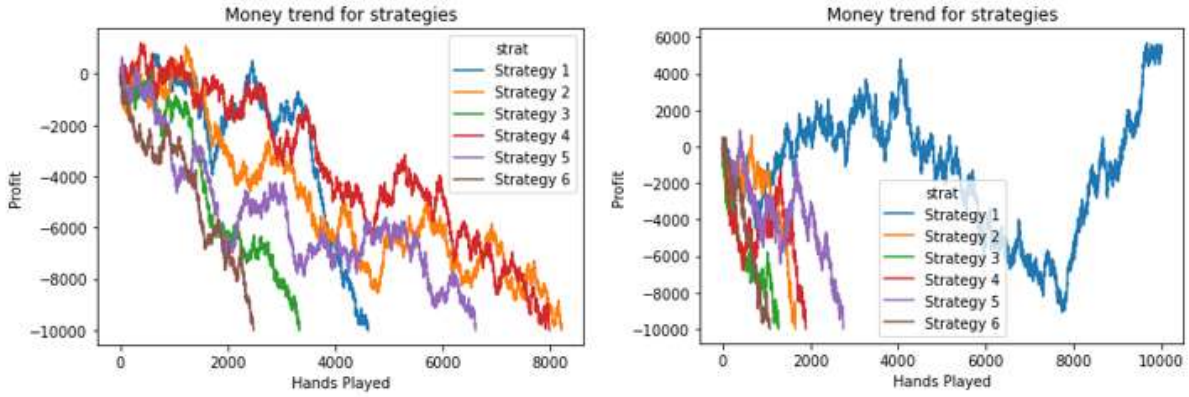


Table 7: Profit Trend per Strategy. Base Wager: Left: $50/Right: $100

Interestingly with the $50 strategy, Strategy 1, the top strategy, runs out of money in about 4,800 hands while Strategies 5, 4, and 2 all played more. However, with a $100 base wager, Strategy 1 not only plays all 10,000 possible hands, but also ends with approximately $6,000 dollars in profit, while all other strategies fail before 3,000 hands. However, in the chart on the right, Strategy 1 nearly runs out of money around 7,800 hands played but appears to go on an incredible run of luck and end in the profit over the next ~2,000 hands. Unsurprisingly, Strategy 6, which we expected to be the worst, is the first strategy to run out of money in both trials. Additionally, all strategies run out of money except for Strategy 1 in the $100 trial, which barely survived solely due to a string of luck. A lot of these trends are likely due to variance. Even though we played 10,000 hands, we only played 1 "game" and only had one starting account balance. This variance is why we will also examine statistics from hundreds of these simulated games.

While we may see variance in total profit among strategies in only one game, the win, push, and loss percentage should remain pretty consistent as that is measured against individual hands, which we have played 10,000 above per strategy. The following bar charts represent the win, push, and loss percentages for each strategy over 10,000 hands, these percentages are independent of base wager values. We will examine these win/push/loss rates and would expect them to be similar to the output statistics for multiple games.

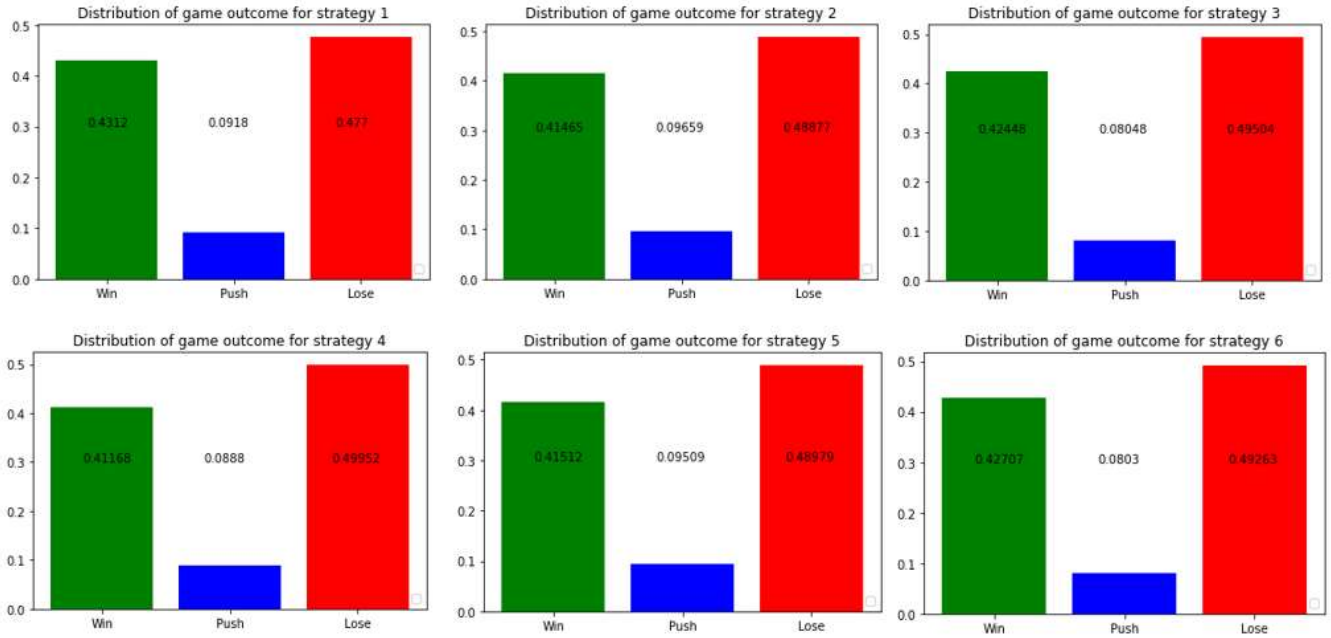


Table 8: Win/Push/Loss Percentages per Wager Methodology

As seen in the above chart, Strategy 1 has the lowest loss percentage and the highest win percentage. Surprisingly, Strategy 6 has the next highest win percentage, followed by strategies 3, 5, 2, and 4, respectively. Strategy 2 has the next lowest win percentage (unsurprisingly as it is closest in strategy to Strategy 1) followed by strategies 5, 6, 3, and 4 respectively. By percentages, Strategy 4 appears to be the worst strategy and Strategy 1 the best. However, Strategy 4 outlasted most strategies in hands played in both our previous trials, most likely due to better Split/Double-Down actions as losses/wins are recorded as a loss win whether it is a singular loss/win or doubled loss/win.

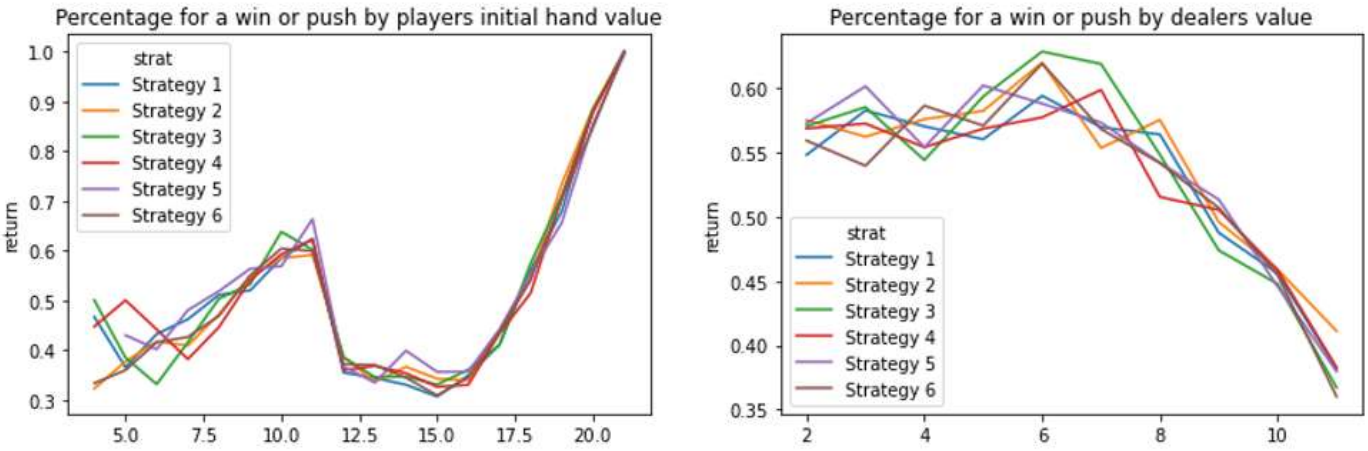


Table 9: Win/Push Percentage Based on Player/Dealer Initial Value

The above charts show the Win/Push percentage values based on the player and dealer's initial cards' value. Again, thee percentages are independent of base wager value and should remain consistent over multiple games as these percentages are determined per hand and were determined

over 10,000 hands in a single trial. For both graphs, all 6 strategies appear to follow the same trends, with a little more variation in Strategy 3 than the others.

For the player's initial hand value, win/push percentages are highest when the player has an initial value either around 8 to 11 or more than 18. This makes sense as if the player begins with 18+, he is already in a good position. And if the player has an 8 to 11, they are in a good position to add a card with a value of 10. Initial player value below 8 and between 11 to 18 have low probabilities to win. An initial value of 15 especially puts you in a hole as a hit is a likely bust, and a stay is an easy threshold for the dealer to surpass. For the dealer's initial hand value, the player's win/push percentage stays relatively steady around 55% if the dealer has below an 8. With a dealer's face card value being anywhere from 8 to 11, the player's win percentage value drops dramatically. This makes sense as the dealer is not only in a likely high hand value situation, but it forces the player to react to the high value card accordingly.

Multiple Game Analysis – Analyze Hundreds of Games Consisting of Thousands of Hands

Now that we have analyzed values coming out of a few trials consisting of 10,000 hands, the simulation will run 100 games with a maximum hands played of 2,000 per strategy. In total, the simulation will have ran, and we will have analyzed 1.2 Million hands across the 6 strategies.

strat	total hands played	ending profit	win %	push %	loss %
Strategy 1	2000	3150.0	0.443000	0.093500	0.485000
Strategy 2	2000	3050.0	0.430688	0.091750	0.483745
Strategy 3	2000	1400.0	0.430833	0.091487	0.486000
Strategy 4	2000	3350.0	0.429450	0.091528	0.486750
Strategy 5	2000	700.0	0.430091	0.091843	0.483832
Strategy 6	2000	-900.0	0.430250	0.091443	0.485833

Table 10: Multiple Game Max Values by Strategy

The above image shows the maximum values of each strategy through 100 games, consisting of 2,000 hands. Strategy 1 has the highest win percentage and 2nd highest maximum profit over 100 games. Interestingly, Strategy 6 never gets into the profit through all its games. All 6 strategies have games where they make it through all 2,000 possible hands. Now, let's look at minimum values by strategy.

strat	total hands played	ending profit	win %	push %	loss %
Strategy 1	2000	-8250.0	0.424855	0.086643	0.463500
Strategy 2	1979	-10000.0	0.424927	0.087438	0.479107
Strategy 3	2000	-9750.0	0.425363	0.087722	0.479833
Strategy 4	2000	-9300.0	0.425000	0.088250	0.480545
Strategy 5	2000	-7650.0	0.425316	0.088100	0.479941
Strategy 6	1582	-10000.0	0.424959	0.087333	0.480417

Table 11: Multiple Game Minimum Values by Strategy

The minimum statistics show that only Strategies 2 and 6 ever hit a total loss in 2,000 hands, which makes sense in context of what we saw in the single game trials, that most strategies don't play

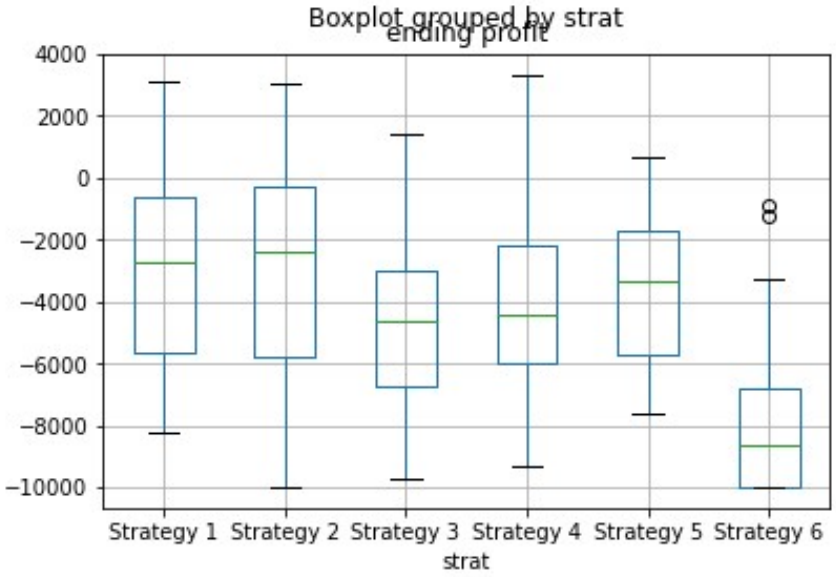
out before 2,000 hands. Strategy 5 has the lowest floor on ending profit but was 2nd worst in maximum profit. Strategy 1 is 2nd in lowest profit and 2nd in highest profit, again making a case that it is the best strategy; even with the worst win% of any strategy across any game.

```
In [198]: fifty_stake_df.groupby(by=["strat"]).median()
Out[198]:
            total hands played  ending profit     win %    push %    loss %
strat
Strategy 1                2000        -2700.0  0.426545  0.090883  0.482640
Strategy 2                2000        -2400.0  0.426686  0.090869  0.482541
Strategy 3                2000        -4625.0  0.426544  0.090793  0.482657
Strategy 4                2000        -4450.0  0.426505  0.090767  0.482605
Strategy 5                2000        -3350.0  0.426568  0.090858  0.482564
Strategy 6                2000        -8625.0  0.426484  0.090813  0.482693

In [199]: fifty_stake_df.groupby(by=["strat"]).mean()
Out[199]:
            total hands played  ending profit     win %    push %    loss %
strat
Strategy 1          2000.00000     -2956.2500  0.427219  0.090764  0.482017
Strategy 2          1999.34375     -2865.6250  0.426836  0.090765  0.482398
Strategy 3          2000.00000     -4606.2500  0.426716  0.090613  0.482671
Strategy 4          2000.00000     -4050.0000  0.426646  0.090642  0.482712
Strategy 5          2000.00000     -3629.6875  0.426753  0.090719  0.482529
Strategy 6          1926.18750     -7728.1250  0.426615  0.090643  0.482742
```

Table 12: Multiple Game Median & Mean Values by Strategy

Finally, we will look at mean and median statistics for all six strategies across 100 games. On average, all strategies can play 2,000+ hands in a game except for Strategy 6, which will average out 1,926 hands. In terms of profit, the true most important factor in gambling, Strategy 2 appears to usurp Strategy 1 as it's median profit is $300 more and it's average is roughly $100 more.



Looking at a box and whisker plot of each strategy's profits, it becomes clear that strategies 1 and 2 are the best, followed by 5, 4, then 3. Lastly, Strategy 6 is not a viable option, as we expected from the beginning when discussed in the introduction.

Conclusion

This project attempted to test multiple strategies against an optimum blackjack strategy found on Google. This report showed what the top strategy was and how each other strategy differed from the optimum strategy. The report would continue on to show how the simulation tool was setup in order to give the reader an understanding of what the scripts are working on behind the scenes. After initialization, the report analyzed individual games consisting of 10,000 hands each. Finally, the report would offer insight on each strategy by running each strategy through 100 games consisting of 2,000 hands played each.

After analysis, it became clear that the top online strategy was a strong contender for the best tested strategy. However, Strategy 2 would end up having the best statistics across 100 games. This isn't necessarily surprising as Strategy 2 was only a small modification to Strategy 1, changing just one action against the dealer's hand. There may still be noise in the output, but it's expected that strategies 1 and 2 performed so similarly and neither is probably "worse" than the other. Running 100 games consisting of 10,000 hands instead of 2,000 may have provided slightly different insight between strategies 1 and 2 but would have required too much computing power than available. Beyond strategies 1 and 2, Strategy 6 was the clear worst, which was expected. However, Strategy 5 surprised me with how well it performed against strategies 3 and 4 considering it was a much larger shift away from Strategy 1 than the others.

In conclusion, the best blackjack strategy you find online will perform reliably for you, but it appears safe to always double-down on an 11, if you're so inclined. Finally, our team cannot recommend blackjack for anything other than a recreation as it appears no matter which strategy you rely on, you'll be losing some money.

References

1. “Bar Charts in Matplotlib.” *Ben Alex Keen*, https://benalexkeen.com/bar-charts-in-matplotlib/.
2. “Matplotlib.pyplot.text¶.” *Matplotlib.pyplot.text - Matplotlib 3.5.0 Documentation*, https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.text.html.
3. “Matplotlib.pyplot.text¶.” *Matplotlib.pyplot.text - Matplotlib 3.5.0 Documentation*, https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.text.html.
4. “Matplotlib.pyplot.text¶.” *Matplotlib.pyplot.text - Matplotlib 3.5.0 Documentation*, https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.text.html.
5. “Pandas.dataframe.boxplot¶.” *Pandas.DataFrame.boxplot - Pandas 1.3.4 Documentation*, https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.boxplot.html.
6. Ph.D, Henry Tamburin. “15 Of the Best Blackjack Strategies.” *15 Best Blackjack Basic Strategies by Henry Tamburin Ph.D (Basic & Advanced)*, 888, 13 Oct. 2021, https://www.888casino.com/blog/blackjack-strategy/best-blackjack-strategies.
7. “Seaborn.lineplot¶.” *Seaborn.lineplot - Seaborn 0.11.2 Documentation*, https://seaborn.pydata.org/generated/seaborn.lineplot.html.