The Card Counter of Monte Carlo:
Testing blackjack strategies using Monte Carlo Simulations

Ty Martz

Abstract

In this experiment, I test 4 different blackjack strategies in a simplified game involving only the player and the dealer. Using python to create a simulated hand and monte carlo techniques to run the simulation a great number of times, I derive the winning percent and the profit or loss on any given hand as well as those of playing twenty hands in a visit to a casino. The four strategies involve one with a handful of rules, one that is more complex, one that is simple, and one that uses uniform PRNs to decide on the player's next move.

After running 1 million hands of each strategy, I have found that none perform well and each win percentage falls within the 30%-40% range. The best performing strategies are those which had some of the most logic built-in to their code, while the worst performing strategies were the most simple and the one using PRNs. Each strategy respectively, only has a 26%, 23%, 12%, or 6% probability of breaking even.

Overall, none of the strategies tested in this report perform well. They each seem to show some promise, but they have limitations as simpler techniques. Adding some complexity and furthering the possibilities in each strategy with more logic may prove to show stronger performance in blackjack.

Blackjack Background and Project Description

Blackjack has been around for centuries and is debated to have originated from the French in King Louis XV's court or possibly even way back with the Romans. These historic gamblers paved the way for the modern gambler to go to a casino floor or their neighbors dining room table to lose all their money to the house and hopefully avoid a visit to the atm.

Blackjack itself is a simple game in which each player is dealt 2 cards from 4-8 standard decks of 52. They can ask to add another card from the top of the deck to their hand to try and reach or get as close as possible to the sum of 21 without surpassing it. If your cards sum up to more than 21, then you go "bust" and lose your entire bet. Each card is worth their stated integer value except the face cards (Jack, Queen, King), which are all worth 10 and the Ace which can either be used as 1 or 11. The other key piece is that the dealer gets a hand as well, but one of her cards is dealt face up so all players can see and strategize based on their own hand as well as the one card shown by the dealer.

Now that we have gotten past this subpar description of the game itself, I can speak further about strategies. Since the goal is to get closer to 21 than the dealer without going bust, you need to make strategic decisions when it is your turn. Playing simply, you can either "hit" or "stay". A "hit" will get a random card from the deck added to your total, while a "stand" will end your turn as you wait for any other players and the dealer to complete their turns. *There are more strategies that involve "splits" and "doubling down", but for the sake of this project, we will stick with hitting and standing.*

At the end of every participant's turn (including the dealer), if you are closer than the dealer to 21, then you win and vice versa for the dealer. If you bust, then you lose regardless of the dealer's hand. If you draw (or push), then you and the dealer tie.

Here the 9-10 (19 total) hand beats the Q-7 (17 total) hand
Source: https://casinoadvizor.com/how-to-play-blackjack/

When betting in this simulation, the minimum is $100 and we will probably keep that as the only bet size. With this amount, if the player loses, then they lose their $100 bet. If the player wins, then they keep their bet and earn another $100.

Now that we have covered the game, we can look at some strategies. Take this chart:

Player	Dealer's card									
hard	2	3	4	5	6	7	8	9	10	A
4-8	H	H	H	H	H	H	H	H	H	H
9	H	Dh	Dh	Dh	Dh	H	H	H	H	H
10	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H
11	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	Rh	H
16	S	S	S	S	S	H	H	Rh	Rh	Rh
17+	S	S	S	S	S	S	S	S	S	S

Source: wizardofodds.com/

This explains when the best time is to hit, stay, split, or double based on the dealer's hand and the player's hand. There are numerous cards like this all based on the number of decks being used, house rules, and other variations of the game.

For my simulation I have come up with 4 strategies that will duke it out to make or lose the most money. They are summed up below along with the dealer's strategy:

Strategy A:

- Stand when your hand is 12-16 and when the dealer's up card is 2-6 or if your hand is 17 or higher.
- Hit if your hand is less than 12 or when your hand is 12-16 and when the dealer has 7-Ace.

Strategy B:

- If the dealer shows 2 or 3, hit until you have 13 or more.
- If the dealer shows 4, 5, or 6, hit until you have 12 or more.
- If the dealer shows 7, 8, 9, 10, or Ace, hit until you have 17 or more.
- If you have an Ace in your hand, take cards until you have 19 against the dealers 9, 10, or Ace, 18 for anything else.

Strategy C:

- Don't Bust (My Grandfather's Favorite)
- AKA Only hit when your sum is less than 12

Strategy R:

- Randomly generate a pseudo random unif(0,1)
- if player hand is less than 12 and random is greater/equal to 0.5, then hit
- if player hand between 12-18 and random is greater/equal to 0.7, then hit
- if player hand between 18-20 and random is greater/equal to 0.9, then hit
- otherwise stay

Dealer Strategy:

- Hit if dealer hand is less than 18 and less than one of the players
- Stand if dealer hand is greater than all of the players

I plan to simulate dealing the cards using random variables based on the decks and test these strategies flat out to see the general statistics and winning percentages. A Monte Carlo simulation with n number of runs for x number of hands per night at the casino will play all of the hands I never could and give us a good sense of the quality of each strategy.

Lastly, Let's go over some of the assumptions and limitations of this simulation:

- Using 4 standard 52 card decks shuffled together. The deck is reshuffled if it falls under 25% of total
- 1 Player and 1 dealer, player is always dealt the first card and player goes first
- Player can only "hit" or "stand"; no splits or doubles in my casino
- If player wins, they profit $100, if they lose, they lose $100

Now, let's jump into my virtual casino and see if we can win some money!

Main Findings

My simulation was created using python and the only libraries used were *random* for number generation, *pandas* for number manipulation, and *matplotlib* for visualization. The general code works in these steps:

1. Function to shuffle a new deck

2. Function to play a hand
    a. Function to implement the player's strategy
3. Run the dealer's strategy based on what the player did
4. Compare hands to get results
5. Simulate and display statistics

The problem I am looking into is how well my four strategies perform. Since it is impossible for me to play 1 million hands of blackjack in a reasonable amount of time (not for lack of trying), I can use a monte carlo simulation to simulate all these games instead. Monte Carlo simulations use random sampling to predict outcomes by simulating the random variables a large number of times. In this case, the random variable will be the cards dealt in the game of blackjack. There is a distribution of 13 unique cards in a deck and when shuffled properly, we can never know which ones we will draw. By simulating blackjack hands over and over again, we can better understand the winning percentage of each strategy and see which is best.

First, here are the results from playing one million hands for each strategy:

Strategy	Win %	Wins	Losses	Draws	Total Profit/Loss
A	38.66%	386,617	536,480	76,903	-$14,986,300
B	38.12%	381,229	542,638	76,133	-$16,140,900
C	34.81%	348,052	595,081	56,867	-$24,702,900
R	30.38%	303,825	629,284	66,891	-$32,545,900

Overall, the four strategies chosen are not recommended for use at your nearest casino. Playing 1 million hands and betting $100 on each hand will likely result in losses of tens of millions of dollars. In fact, it would be best to follow my personal strategy, which is don't gamble at Blackjack. Instead, host a game and play as the

dealer.

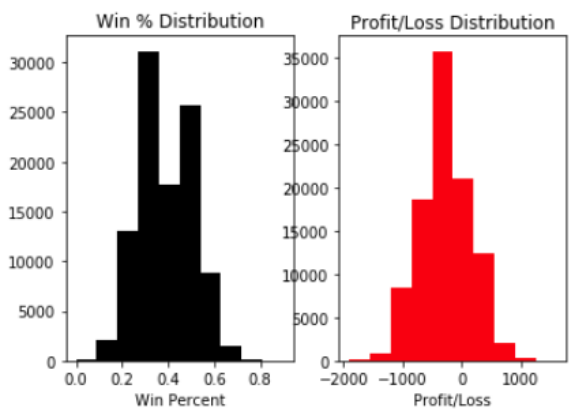
Strategies A and B had the best winning percent at ~38% and were nearly even across, wins, losses, and draws. These two strategies had more rules and logic implemented in them to make the best decisions given specific scenarios in the game. B is a bit more complicated than A covering more scenarios with greater specificity. Strategy C had a simpler method of playing with only one real rule to follow; only hit when your card sum is under 12. This strategy will only result in a win for you, about ⅓ of the time and is even less recommended. Strategy R, my personally crafted strategy based on Unif(0,1) PRNs performed worst of all with a win percent around 30% and losing ~$32.50 per hand. Sounds fun!

Result of 100k Simulations runs of 20 hands played with each strategy:

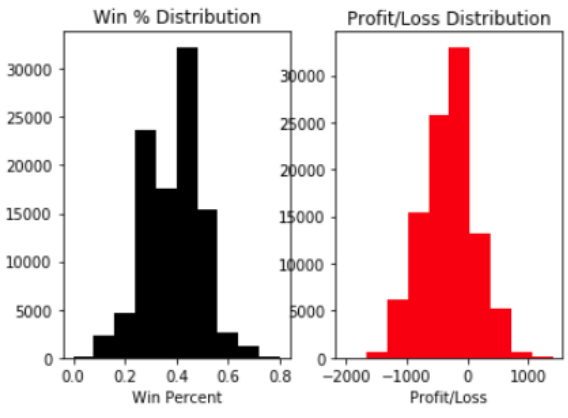
(Run twice, this simulation returns nearly the same exact results!)

Strategy	Mean Win %	Win % Std Dev	Mean Profit/Loss	Profit/Loss Std Deviation
A	39%	11%	-$295.95	$425.15
B	38%	11%	-$321.45	$424.11
C	35%	11%	-$497.64	$420.5
R	30%	11%	-$651.19	$405.82

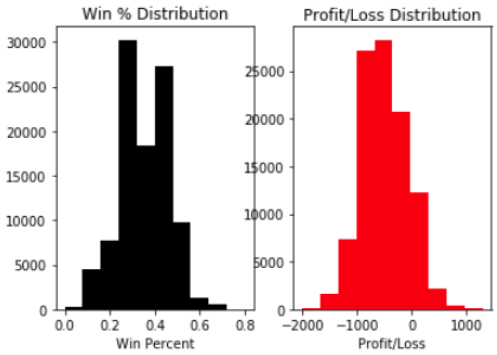
A Distribution



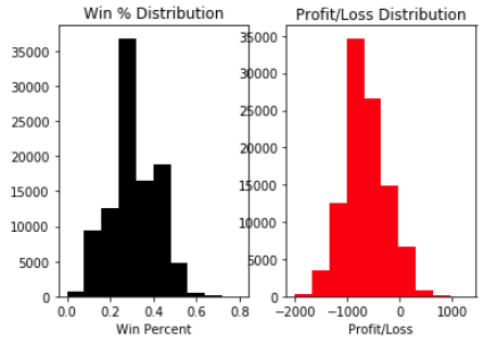
B Distribution



C Distribution



R Distribution



The above section runs through 100k simulations of 20 hands each. This simulates a night at the casino, playing 20 hands with the dealer and betting $100 per hand. Due to the central limit theorem, we see the distributions have all essentially become normalized. As expected the win percent results are the same as above. The distributions are where things get interesting. Looking mainly at standard deviation we can see that within one standard deviation of the mean, only strategies A and B have a chance to even turn a profit on any given night. Strategies B and C at mean losses of -$500 and up and standard deviations only in the $400s have no way to make a profit in the 68% area under the normal curve.

Assuming each distribution is relatively normal I can compute some simple probabilities that any one of these strategies at least breaks even using their z-scores. Using the formula $z = (x - \mu) / \sigma$, where x is set to zero, mu is the distribution mean, and sigma is the standard deviation, I can calculate z and get the approximate area under the curve to the right of zero.

- Strategy A:
  - z = (0 - (-295.95)) / 425.15 = .6961 → 1 - .74 or **26% to break even**
- Strategy B:
  - z = (0 - (-321.45)) / 424.11 = .7579 → 1 - .77 or **23% to break even**
- Strategy C:
  - z = (0 - (-497.64)) / 420.50 = 1.1834 → 1 - .88 or **12% to break even**
- Strategy R:
  - z = (0 - (-651.19)) / 405.82 = 1.6046 → 1 - .44 or **6% to break even**

Conclusions:

This project helped me to better understand using simulation principles with python. From initializing the correct parameters, to testing certain functions, and looping over multiple functions at once to get the results of millions of simulations. Monte Carlo simulations have also proven to be a powerful technique to show the central limit theorem in action and understand how normalizing random events can shine light on insightful statistics. It turns out, to be somewhat successful at Blackjack, you need both luck and more complex strategies on your side. I'm sure counting cards doesn't hurt either (if you don't get caught).

Appendix

Find code files attached in submission. There will be a Jupyter notebook, a python file, and a README.md file along with this report.