Rick Pfeffer

3 March 2021

ISYE 6644 Mini-Project #1

Let’s Gamble!

Abstract

This project built, simulated, and analyzed the game of blackjack, and specifically found the best strategy for a single player to beat the dealer. The simulation and output analysis showed that the player is seldom at an advantage when the dealer stands at 17 or more points.

The best strategy for the player is to stand at low point targets (12 to 15) if the dealer is showing cards likely to cause them to bust (six or less). Otherwise, if the dealer is showing cards that are likely to be advantageous (seven or greater), the player is more likely to win at higher point targets (17 or more). A detailed presentation of point targets per dealer showing points will be presented in the findings section.

Background

Blackjack, often called 21, is a casino card game wherein a player is pitted against the dealer. The object of the game is to win discrete rounds by having a higher score than the dealer. The order of play is as follows:

1. Each player (including the dealer) is dealt two initial cards. The sum of the points of each card constitutes the total points of the player’s hand.
2. The player can see only one of the dealer’s two initial cards.
3. The player may “hit,” or request an additional card from the dealer any number of times.
4. If the players point total goes above 21, the player “busts” and loses.
5. The player may “stand” at any time in their turn, meaning that they accept their hand and points for that turn and take no further action.
6. After all players have either stood or busted, the dealer will perform their turn. The dealer may stand or bust just like the players.
7. If the player’s points are higher than the dealer’s points at the end of the round (and they did not bust), or if the player did not bust and the dealer did bust, the player wins!
8. If the dealer’s points are higher than the player’s points, the player loses.
9. If the dealer and player have equal points, it is a tie.

Some important rules of play are as follows:

- The dealer must “hit” if their points are less than 17.
- The dealer must “stand” if their points are 17 or higher.
- If the player “busts,” they lose regardless of the dealer’s outcome.

Points values for cards are as follows (note that aces are discretionary):

Rank:	Ace	Two	Three	Four	Five	Six	Seven	Eight	Nine	Ten	Face card
Points:	1/11	2	3	4	5	6	7	8	9	10	11

Table 1: *Card points values by rank.*

Blackjack has already been extensively studied and statistically analyzed. It is accepted that the house has an edge over the player, and the margin of this edge depends upon certain rules that can change from casino to casino. For example, some casinos require the dealer to hit on 17 or less, instead of 16 or less. This puts the dealer at less of an advantage. Ultimately, the dealer is at an advantage for one primary reason: if a player busts, they lose their bet *regardless* of the dealer's outcome.

The remainder of the report will include: the methods used to complete the simulation (including assumptions and design decisions), a basic description of the code and logic, major findings and player strategy based on computed statistics, and concluding remarks.

Methods

To simulate the optimal blackjack strategy, I first started with some assumptions:

1. Players are not card counters. The only information they will have are the outputs of this analysis and what is currently visible on the table.
2. Simulation will not honor the "soft 17" rule, meaning the dealer *may* stand on 17 points.
3. The dealer *must hit* if their points are below 17.

Other decisions made for the scope of this simulation are as follows:

1. The simulation will not deal with betting or payouts.
2. The player may not surrender.
3. The player cannot double-down or split their hand.

With the above in mind, the simulation simply measures the best strategy to win the greatest number of hands. It does not look at making the most money over a certain number of games.

The simulation itself was built using object-oriented programming principles and was developed by hand using Python. Random numbers were generated using Python's "random" module. These were used for shuffling the simulated decks and shoes (multiple stacked decks) of cards and generating a random number of decks to be used per game.

The simulation ran 1,000,000 trials with 100,000 trials per point target for the player. Point targets were set at twelve to twenty-one, since it *never* makes sense to stand below twelve points (hitting once on eleven or less points cannot cause a bust). For each round, the player and dealer were dealt random cards and the player would begin hitting until reaching their minimum point target. The dealer would then hit until reaching at least 17 points. Rounds were resolved as per the steps previously mentioned in the "Background" section.

Findings

Unsurprisingly, the dealer and casino have a substantial advantage in blackjack over the average player who cannot count cards. **Figure 1** below shows the probability of winning any single round if a player chooses a specified point target.

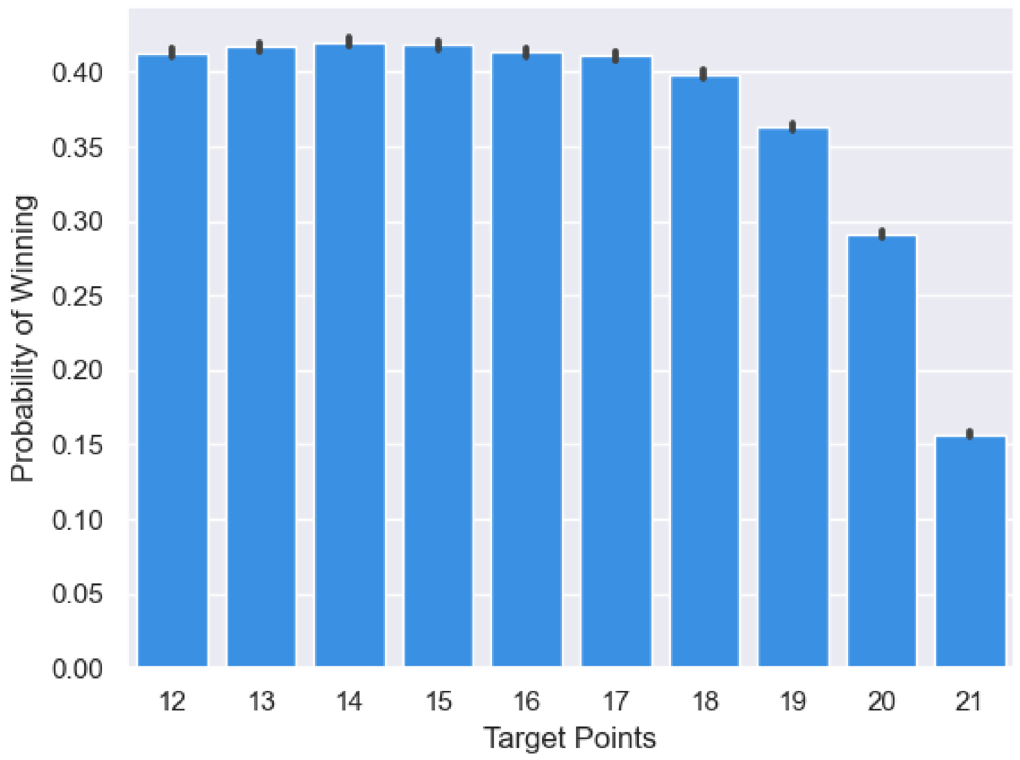

Figure 1: *Probability of winning with specified target points.*

We can see from the above figure that with this strategy, the player never gets above a fifty percent chance of winning. The best strategy is for the player to always shoot for a minimum score of 14, which will put their likelihood of winning any given hand at around 0.44. This is unsurprising, because if it were that easy, a lot more people would be playing blackjack for a living!

The player does however get some important information that they can consider. The dealer's likelihood of busting and staying is different depending on their initial two cards, of which the player can see one. We can determine the probability of a player winning a hand for a specific point target given that the dealer is showing a certain number of points to the player. **Figure 2** shows the relationship between a player's point target, the dealers "showing points" and the players probability of winning any given hand.

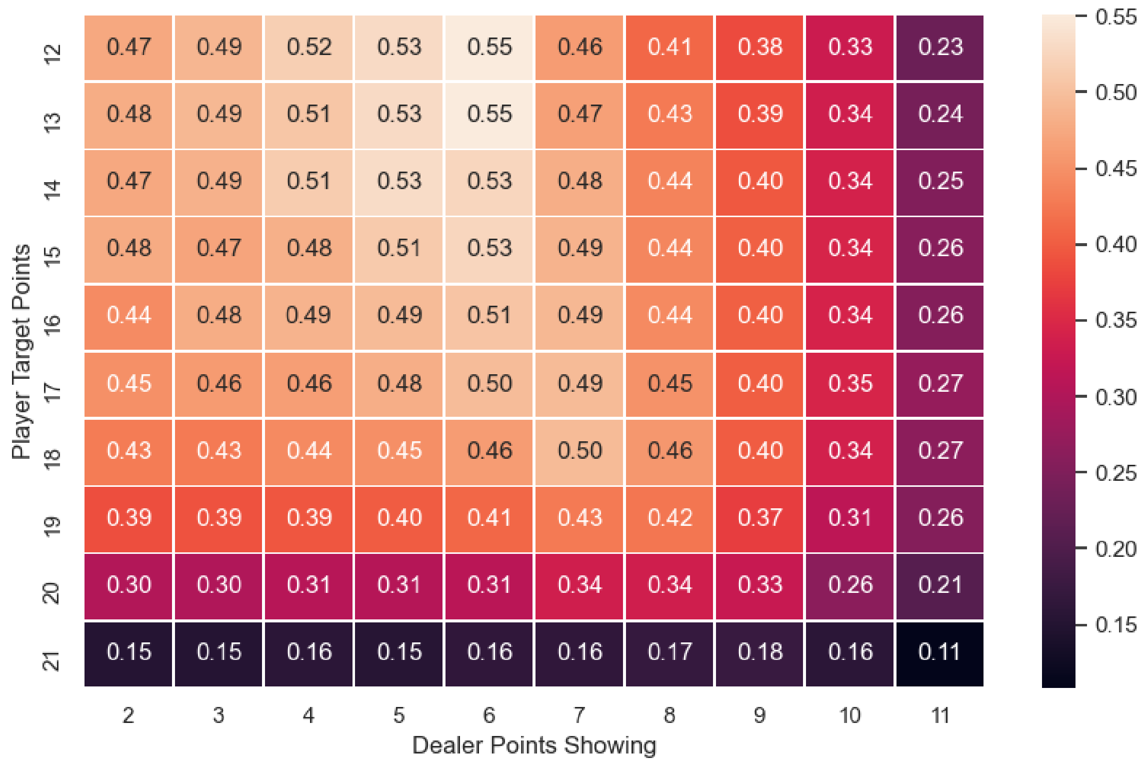

Figure 2: *Probability of winning – target points vs. dealer points showing.*

The table shows that likelihoods are mostly bleak for the player. With that said, there *are* some cases where probabilities finally jump above 0.50, especially when the dealer shows six points or less. Based on the above matrix, **Table 2** shows the recommended point target determined by the dealer's showing card.

Dealer Shows	2	3	4	5	6	7	8	9	Face Card	Ace
Target	13	12	12	12	12	18	18	14	17	17

Table 2: *Recommended point target depending on dealer's showing card.*

Interestingly, we can see the effect of the rule that the dealer must hit less than 17 points in the above table. Since the most likely card point value is ten (due to face cards and tens taking up over 30% of the deck), the dealer is likely to end with a point value of 17 if they start by showing seven. In this case, the player needs to shoot for a higher point value and not bet on the dealer busting.

Conclusions

While the player cannot have an upper hand over the dealer in blackjack (without some serious card counting skills, at least), this project shows the best chance the player can have. It is best for the player to focus on not busting when the dealer shows less than seven points, and to focus on attaining a higher value hand if the dealer shows seven or more points.

There is a lot of additional work that can be done to firm up this simulation and analysis. First off, the addition of betting and tracking overall winnings would be beneficial. It is not fully sufficient to track wins and losses because the true goal of the player is to maximize their earnings and not their total won rounds (although the latter may serve as a decent proxy).

Other potential next steps would be comparing the soft 17 rule to the hard 17 rule, adding in "intelligence" to the player and the ability to simulate card counting and attention to the number of decks in play.

Appendix

To run the simulation, simply run the “classes.py” file in your favorite python IDE or from the command prompt with python 3 installed. This will create the “output.csv” file with the 1,000,000 simulations and pertinent information and data. The “analysis.py” file will then reference the csv, and will create the figures and most of the tables shown in the report.

SOURCES

https://seaborn.pydata.org/generated/seaborn.barplot.html

https://stackoverflow.com/questions/62696868/highlighting-maximum-value-in-a-column-on-a-seaborn-heatmap