Analysis of Blackjack & Winning Strategies Using Simulations

Alexander Weber

December 3rd
2020

1 Abstract

Gambling is a popular recreational activity known worldwide. Many see it as a waste of time and money, while some believe that it can bring them great riches. Whatever the goal, instances of gambling can be seen everywhere, from lotteries to sports. Usually, gambling allows players to bet money on the predicted outcome of an event, winning or losing their bet if they are correct or incorrect.

Some of the most popular and well known forms of gambling are those played with decks of cards at casinos. One famous game is blackjack, a popular game played around the world. Originally known as Vingt-Un in the 17th century, it was popularized in the western world as blackjack during the turn of the 20th century. As with any game involving gambling, the ability to predict the outcome can result in a significant advantage when making a bet. In this paper we will examine some winning strategies by looking at the expected outcomes.

Within this paper, we will first give a background on the topic, followed by a discussion of the rules of blackjack. Next we will use an analytic approach to examine some simple situations and their expected outcomes. Continuing, we will extend these simple situations into more complex ones, using simulations to solve them. Finally, we will conclude with an discussion of our results.

2 Background

Predicting the outcome of card-based casino games has been a popular topic for decades. More commonly known as 'counting cards', gaining an upper hand over the house can lead to substantial wealth, if played correctly.

For this reason, significant work has been completed in regards to the analysis of blackjack and ways to play optimally. One of the most popular pieces of literature is Edward Thorp's "Beat the Dealer: A Winning Strategy for the

Game of Twenty-one" published in 1962. In his book Thorp discuses what is commonly known as the 'Basic Strategy', a series of rules for giving the player a slight edge over the house.

Since 1962, there has been extensive continuations of Thorp's work, mostly centering around a single player who is playing the game legally. We will build on this research and continue to examine ways of beating blackjack, both legally and illegally.

3 Rules

Before we begin our analysis of blackjack, we will first take a look at the game's rules. This is important as blackjack has many variations upon its rules and it is important for us to select the ones we'll be using. We will be working with the most common rule set for blackjack used by casino's and other gambling establishments.

For the rest of the paper we will use the terminology dealer or house to refer to the individual that the player is competing against. We will also use the term player to refer to us, as the individual trying to win.

3.1 Dealer Rules

The dealer represents the casino and is in charge of running the game. At the start of the game, the dealer will draw two cards from the deck for every player. One of these cards will be given face-up to the player, the other will be face-down. Following, the dealer will draw two card for himself, one face up. The dealer will then take bets from all players.

On the dealer's turn, the dealer must hit as long as the dealer's hand shows anything less than 17. We will investigate the reasoning behind this rule later into the paper.

3.2 Player Rules

Players can join a game before the dealer hands out cards, but may not leave a game until it is completed without forfeiting their bet. Note that this means that we do not consider surrendering an option. Players that join a game must bet anything between $10 and $500. Players are exclusively competing with the dealer and not with the other players.

On the player's first turn, they will be dealt two cards from the dealer. One of these cards will be face up, observable by all other players, while the second

one will be face down, only observable by the player.

On the player's turn, a player may stand or hit. Standing will skip the player's turn, keeping their hand. Once a player stands they are committed to their hand. On a hit, the dealer will draw an additional card, face up, and deal it to the player.

3.3 Deck Rules

Any number of decks may be used, but each deck of cards used must be comprised of 52 total cards, with an equal count of 4 cards for each numbered and face card. We will not be using jokers, and we will also be ignoring the suit of the card as it has no impact on the game.

Cards will not be observable by the player until they are drawn by the dealer from the deck. Cards drawn by the dealer will be observable by all players, except for the first initial card that is drawn face-down for each player. At any point following a game, our dealer will shuffle all of the cards back into a single deck.

3.4 Game Rules

Within our casino, there exists any number of tables. Each table is seated by a single dealer and up to 5 players.

At the beginning of a turn, the dealer will supply all participating players two cards, per the dealer rules. Following this, from left to right, each player will have their turn and decide to hit or stay. This will continue, starting from the left again, until all players either stand or bust, going over 21.

Once all player turns have completed, the dealer will continue to draw until at or above 17. At that point, players with a total score higher than the dealer will win, receiving double the initial bet. Those with a lower score will lose, forfeiting their wager. Upon a tie, the wager is returned to the player.

There are additional rules to blackjack such as doubling down, surrendering, and splitting. We will not be considering those as part of our analysis in order to keep the complexity simple, instead focusing on the core game strategy.

4 Analytic Approach

Before we use simulations to determine the best strategy for our dealer or player, we will first use an analytic approach to obtain some intuition about the problem.

First we will define our approach, looking at how we calculate our probabilities. Then we will look at two scenarios exploring ideal situations for both the dealer and the player. The first, an ideal situation for our dealer, is one where we have an infinite deck, where we draw cards with replacement. Secondly, we will look at an ideal situation for our player, in which we have a single deck that is shuffled only when every card has been drawn.

In regards to notation, we will use **C** to represent our sample space, P to denote the probability mass function for drawing a card, and c to represent a single card.

4.1 Card Drawing

The probability for our ideal card drawing is fairly simple. If we let the card we would like to draw be c, then the probability of drawing c is equal to the following.

$$P_1(c) = \frac{Number\ of\ cards\ c}{Size\ of\ the\ deck}$$

For example, if our goal was to draw a 5 of any kind from a deck of 52 cards, there would be four instances of a card with a value of 5 in our deck, giving us a probability of

$$P_1(c{=}5) = \frac{4}{52} = \frac{1}{13} = 0.0769$$

However, as we continue to draw cards from our deck, the total number of cards will shrink. Additionally, for every draw of a card c, any future draw of the same card value will have a reduced probability. Extending on the first equation, we can arrive to our second equation for probability assuming no replacement occurs.

$$P_2(c) = \frac{Number\ of\ cards\ c\ \ -\ \ Number\ of\ previous\ draws\ for\ c}{Size\ of\ the\ deck\ \ -\ \ Number\ of\ cards\ drawn}$$

As an example, if we were dealt a hand of two kings, the probability of being dealt an additional king would be equal to

$$P_2(c{=}K) = \frac{4\ Kings - 2\ Drawn}{52\ Cards - 2\ Drawn} = \frac{2}{50} = \frac{1}{25} = 0.04$$

4.2 Estimated Win Rate With Replacement

(Analysis of a Non Ideal Environment for the Player)

Using the equations above, we will now investigate what the win rate for the player using an infinite deck, or a deck where we replace cards with a new, identical card (replacement). In this situation, each draw of a card is independent of any previous draw, disallowing our player from gaining any advantage from counting cards. Additionally, this makes modeling this scenario fairly straightforward.

As a win is not determined by the player's score directly, but instead based off of whether or not our player has a higher score than the dealer, we will need to first calculate what the distribution of the dealers cards will be if they hit up until reaching 17.

To calculate the dealers distribution, we simply calculate the total possible ways to obtain a total score of 17 through 21, as well as all scores that bust. To do this we calculate a permutations set N such that for each hand H in N, the total sum of all cards in H is greater than or equal to 17 and the sum of all cards in H minus the last card in H is less than 17. As the probability for each card is identical as we will be using P_1 to calculate the probability with replacement, we do not need to calculate probabilities for each hand since they are all identical.

After calculating the total probability distribution for all hand scores per the above restrictions, we get the following results. Note that the expected outcome for a hand that does not bust is 19.

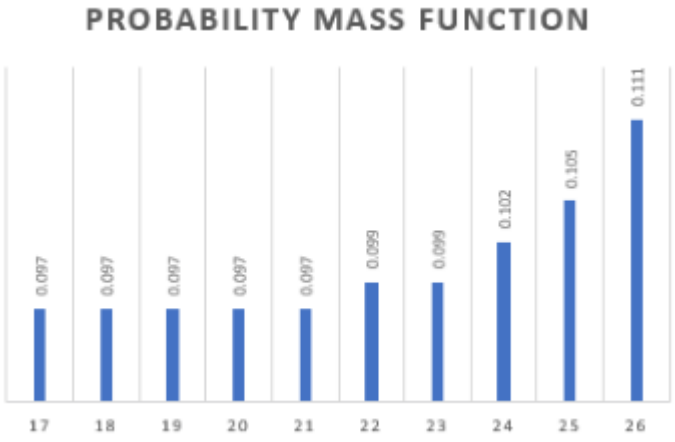

Fig 4.2.A Probability mass function for each hand's final score if stopping at 17 or higher. See Appendix for information on code.

Having obtained this data, we will now perform a similar procedure for the player. However, while the dealer will stop at 17 or higher, we will try a range of stopping values for the player. We will take this data and compare its distribution to the dealer. The following graph shows the average outcome of a hand that does not bust. We will consider a strategy better for the player if the expected outcome of the hand is higher than 19 and the bust ratio is at most equal to the dealers 50%. Below are our results for target values of 14 through 21.

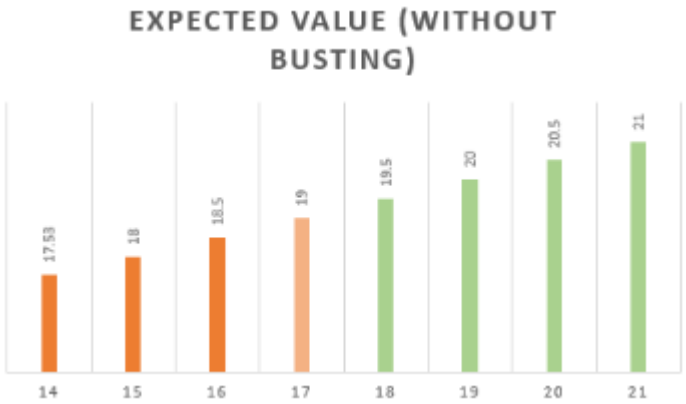

Fig 4.2.B Expected values for the following hands stopping at 14 through 21. Excludes the expected values of busts.

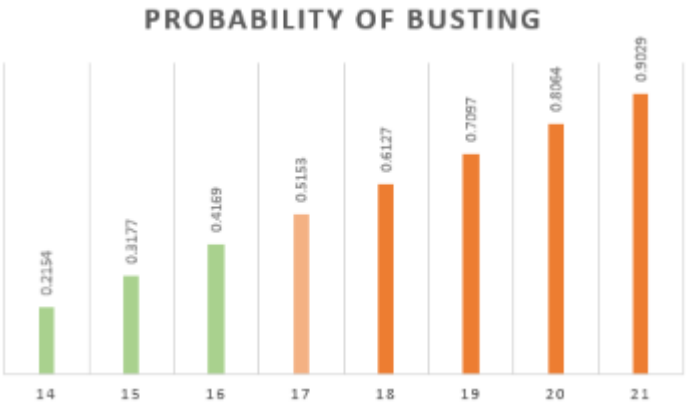

Fig 4.2.C Probability of busting with a given target. This is not the same as the probability of busting if hitting on a hand with the total value.

As we can see, there is no value that beats the dealers average expected value of 19 while maintaining a 50% or less bust ratio, confirming that 17 is the ideal target for the dealer. This means the player can only tie or lose over time, as the player hits first and would ultimately bust before the dealer.

4.3 Estimated Win Rate Without Replacement

(Analysis of an Ideal Environment for the Player)

We will now examine an ideal environment for the player, in which there is a single deck. This gives the player a significant advantage in predicting which card will come next, as each card draw is not independent. We will perform the same steps as discussed in 4.2, except we will use our equation P_2 before drawing another card, and hold if the expected card value will cause us to bust. We will also perform a similar calculation for the dealer, as our data obtained in 4.2 does not consider a single deck.

From our computation we obtain the following data.

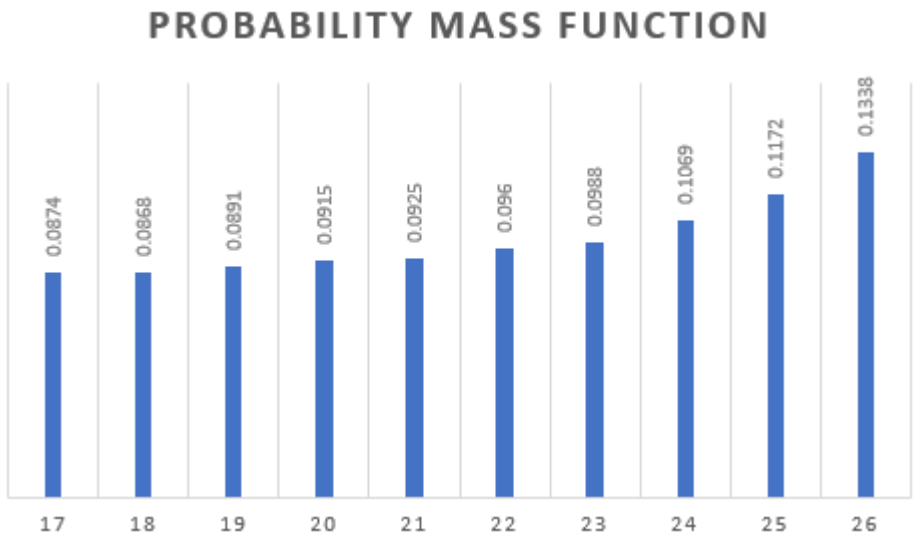

Fig 4.3.A Probability mass function for each hand's final score if stopping at 17 or higher. See Appendix for information on code.

Note that the probability mass function is roughly identical to the data obtained in 4.2. This is likely explained by the fact that the most important calculation for the final value is obtained from the final card, which is still fairly equally distributed, even if the hand that gets us there is different.

Now we will calculate all hand permutations for the player, but we will exclude permutations where the probability of a bust on the final card is more than 50

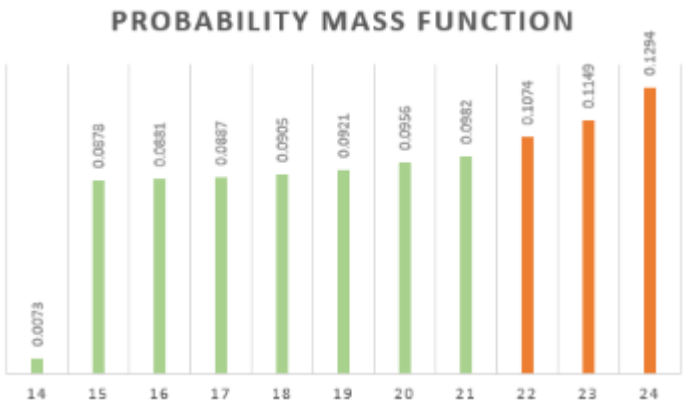

Fig 4.3.B Probability mass function of our hand's final value if we stop when the summation of P_2 for cards multiplied by its probability gives us a higher value than the remaining difference between our hand's score and 21.

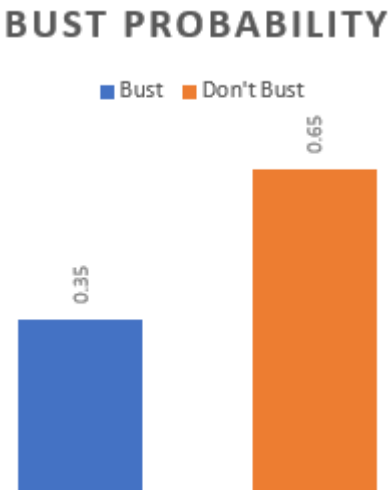

Fig 4.3.C Probability of busting using the above formula.

This data gives us a 65% chance of not busting, compared to the dealer's 45%. Additionally, this gives us an expected hand value of 19.8, higher than the dealer's expected hand value of 19. In theory, this strategy gives us an edge over the dealer. We will disprove this later in our simulation sections, and expose how the turn order has a major impact on the game.

5 Simulating Additional Strategies

Now that we have performed a quick analysis of two ideal scenarios, we will now simulate these scenarios using Python to build a casino. We will perform 50,000 samples and compare primarily how often our player beats the dealer. Starting off we will verify our data for the scenarios we studied analytically, then look at three more scenarios.

5.1 Verifying Analytic Data

First, we will perform simulations to determine the win frequency for our player using a deck without replacement as well as a single deck. We obtain the following data for both situations.

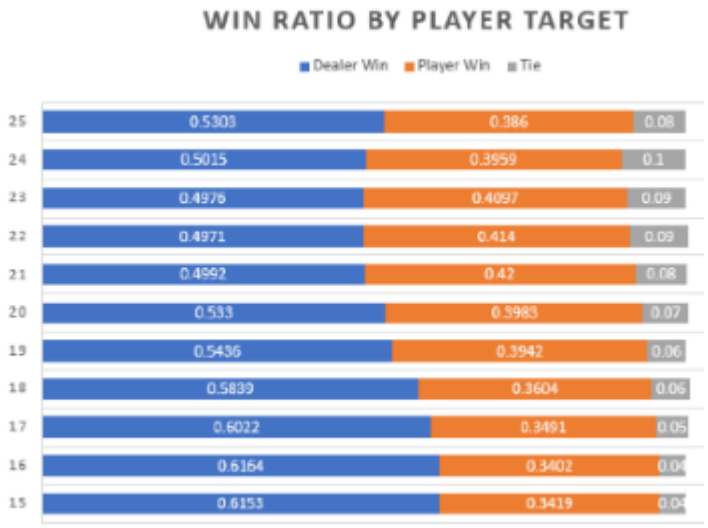

Fig 5.1.B Win probabilities based on player's target value.

This data seems to validate the data obtained in 4.2, such that the dealer will always maintain a slight edge over the player.

Moving on to comparing our analytic data in 4.3 against our simulation, we obtain the following data. Note that the target value is the score that the player will stop drawing at, if the estimated value of the next card plus the player's current hand exceeds it.

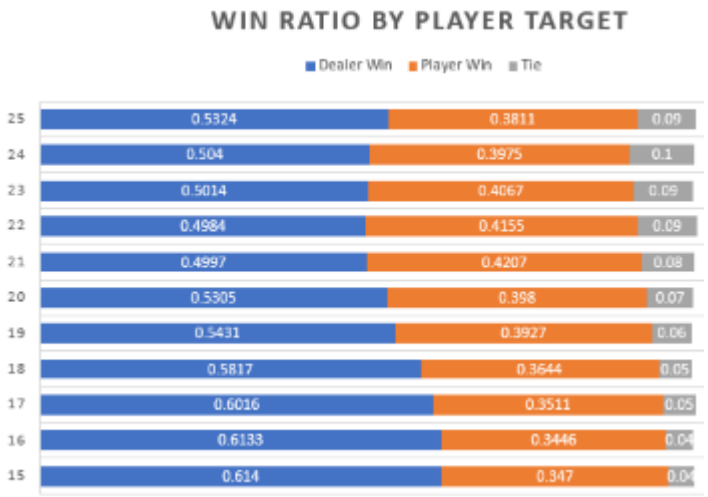

Fig 5.1.B Win probabilities based on player's target value.

This data seems to contradict our analysis of the player choosing when to hit based off of the estimated value of our single deck. However, considering that a bust for the player will always happen prior to the dealer, that gives the dealer an edge over the player. Additionally, this data seems to be identical to that obtained in 5.1, which may imply that the situations are less different than expected.

5.2 Simulating Five Decks

Continuing, we will increase our deck size to 5. This should demonstrate a decrease in the player's performance as it becomes harder to predict the overall outcomes. Below are our results.

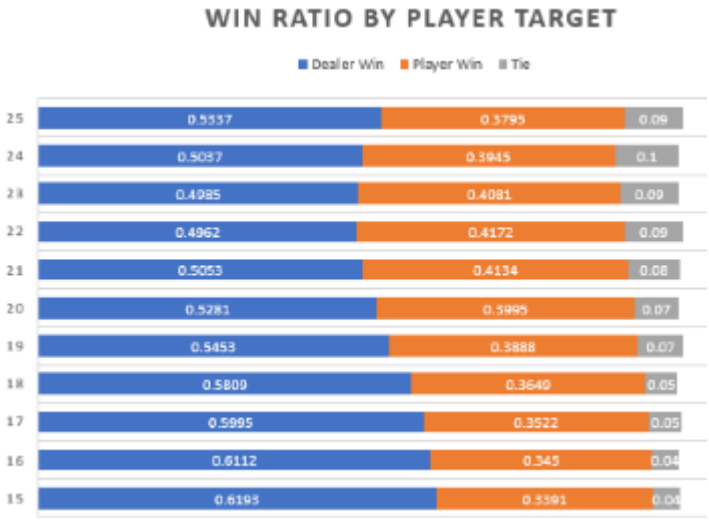

Fig 5.2.A Win probabilities based on player's target value.

As we can see, these data points are fairly similar to the ones in the above two scenarios, but are slightly closer to those calculated in the infinite deck example.

5.3 Simulating Additional Players

Next, we will perform the same scenario as we did in 5.1.B, however we will have 15 players draw two cards each before our player's turn. This will reduce the total deck size and make it easier for our player to calculate what the most likely card draw is, given that the player will be able to observe those 15 player's hands. In practice, this sort of situation would be unlikely and possibly illegal depending on the rules.

We obtain the following results using the above approach.

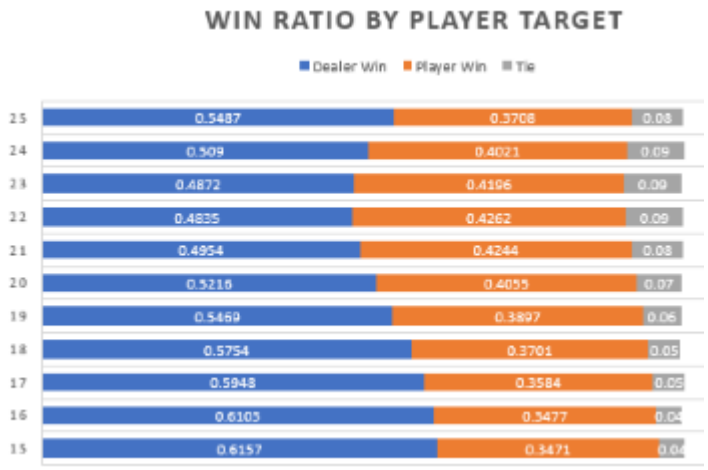

Fig 5.3.A Win probabilities based on player's target value.

As we can see, despite the increase in knowledge for the player, we are still not able to gain a significant advantage over the dealer. Ultimately, this has to do with the fact that our distribution, while smaller, is ultimately still represented

5.4 Inaccurate Dealer

Finally, we will explore the same situation in part 5.1.B, with a single deck, however in this case our dealer will play inaccurately, continuing to hit until reaching 16 instead of 17. We will also look at 18 instead of 17. Our player will use the same procedure as that in 5.1.B.

Our results are below.

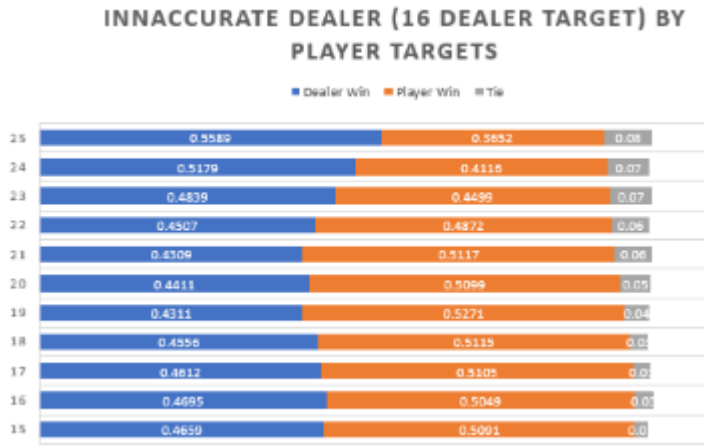

Fig 5.4.A Win probabilities based on player's target value.

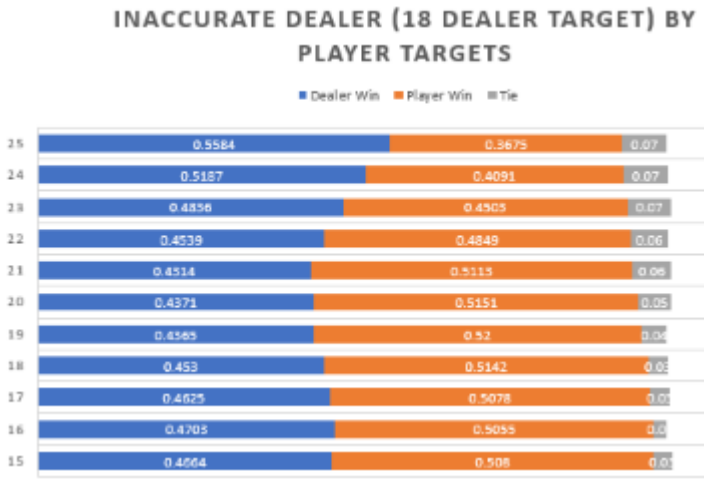

Fig 5.4.B Win probabilities based on player's target value.

What is interesting about this data is how the player will suddenly get a significant advantage on all target scores that do not bust. This reinforces not only the correctness of the dealer's target value of 17, but also shows how tight of a game blackjack is.

6 Conclusion

To review, we have gone through the background of blackjack, it's rules, took some analytic data for some simple situations, and then ran simulations for more complex ones. Reviewing the data, it is not easy to see that the player is not able to gain an advantage over the dealer using the rules that we've selected, as long as the dealer plays correctly. This is likely, as previously discussed, due to the nature of the way the turns in blackjack are ordered. When the player busts, the dealer is not given an opportunity to bust as well, thus favoring a player bust and a dealer win.

However, when the dealer does not play optimally, our player can gain an advantage by sticking to the defined rules, winning more than 50% of the time across our 50,000 independent trials.

Some future work could be to expand our rule set, allowing for more varied strategies by the players.

7 Appendix

A: Code used to generate data from figures in both 4.2 and 4.3.

```
cards = [2,3,4,5,6,7,8,9,10,'J','Q','K','A']

from collections import defaultdict
total_count = defaultdict(lambda: 0)

def permutate(current_score, score_list, cmap=None):
    if current_score >= target_score:
        if 'A' in score_list:
            a_index = score_list.index('A')
            score_list[a_index] = 1
            current_score -= 10

        if current_score >= target_score:
            total_count[current_score] += 1
            return

    else:
        # Performing permutate with all available cards
        for c in cards:
            if cmap and cmap[c] >= 4:
                continue

            new_map = cmap.copy()
            new_map[c] += 1

            new_score_list = score_list.copy()
            new_score_list.append(c)
            if type(c) == int:
                permutate(current_score + c, new_score_list, new_map)
            else:
                if c == 'J' or c == 'Q' or c == 'K':
                    permutate(current_score + 10, new_score_list, new_map)

                elif c == 'A':
                    permutate(current_score + 11, new_score_list, new_map)
```

B: Code used to generate data for section 4.3

```
def calc_prob(score_list):
    # Calculating the estimated next-hand probability
    total_score = 0
    total_count = 4 * len(scores)
    for score in scores:
        total_score += (score * 4)

    for score in score_list:
        index = cardsStr.index(str(score))
        total_score -= scoreMap[index]
        total_count -= 1

    return total_score / total_count

def permutate_prob(current_score, score_list, cmap=None):
    if calc_prob(score_list) + current_score > 21:
        if 'A' in score_list:
            a_index = score_list.index('A')
            score_list[a_index] = 1
            current_score -= 10

        if calc_prob(score_list) + current_score > 21:
            total_count[current_score] += 1
            return

    else:
        # Performing permutate with all available cards
        for c in cards:
            if cmap and cmap[c] >= 4:
                continue

            new_map = cmap.copy()
            new_map[c] += 1

            new_score_list = score_list.copy()
            new_score_list.append(c)
            if type(c) == int:
                permutate_prob(current_score + c, new_score_list, new_map)
            else:
                if c == 'J' or c == 'Q' or c == 'K':
                    permutate_prob(current_score + 10, new_score_list, new_map)

                elif c == 'A':
                    permutate_prob(current_score + 11, new_score_list, new_map)
```

C: Code used to generate data for section 5.1 through 5.3.

```
import random
from collections import defaultdict

cards = ['2','3','4','5','6','7','8','9','10','J','Q','K','A']

class Deck:

    def __init__(self, size: int):
        self.cards = ['2','3','4','5','6','7','8','9','10','J','Q','K','A']
        self.values = [2,3,4,5,6,7,8,9,10,10,10,10,11]
        self.size = size

        # Keeping track of the total number of drawn cards
        self.totalDrawn = 0
        self.drawnMap = defaultdict(lambda: 0)

        # Building the deck if our size is nonzero
        if self.size != 0:
            self.deck = list()
            for card in self.cards:
                for i in range(4 * size):
                    self.deck.append(card)

    def draw(self):
        if self.size == 0:
            return random.choice(self.cards)

        card_index = random.randint(0, len(self.deck) - 1)
        self.totalDrawn += 1
        card = self.deck[card_index]
        self.drawnMap[card] += 1
        return self.deck.pop(card_index)

    def count_prob(self, cardType):
        if self.size == 0:
            return 1/13

        # Calculating the probability of drawing this card type
        totalCards = (self.size * 4)
        totalCards -= self.drawnMap[cardType]

        return totalCards / (len(self.deck))
```

```
    def expectedDrawValue(self):
        if self.size == 0:
            o = 0
            for v in self.values:
                o += v
            return o / len(self.values)

        o = 0
        for i in range(len(self.cards)):
            o += (self.count_prob(self.cards[i]) * self.values[i])
        return o

target_player_score_set = [15,16,17,18,19,20,21,22,23,24,25]

for player_target_score in target_player_score_set:

    final_scores = {'dealer': 0, 'player': 0, 'tie': 0}

    for iteration in range(0,50000):

        # Creating the deck
        d = Deck(5)

        # Dealer drawing initial card
        dealer_card = [d.draw()]

        card_index = d.cards.index(dealer_card[0])
        dealer_score = d.values[card_index]

        # Drawing two card's for the player
        player_cards = [d.draw(), d.draw()]

        # Calculating the score
        player_score = 0
        for card in player_cards:
            card_index = d.cards.index(card)
            player_score += d.values[card_index]

        # Checking if we busted
        if player_score > player_target_score:
            # Do we have an ace?
            if 'A' in player_cards:
                a_index = player_cards.index('A')
                player_cards[a_index] = '1'
                player_score -= 10
```

```
while d.expectedDrawValue() + player_score <= player_target_score:
    new_card = d.draw()
    player_cards.append(new_card)
    card_index = d.cards.index(new_card)
    player_score += d.values[card_index]

    # Checking if we busted
    if player_score > player_target_score:
        # Do we have an ace?
        if 'A' in player_cards:
            a_index = player_cards.index('A')
            player_cards[a_index] = '1'
            player_score -= 10

# Drawing for the dealer
while dealer_score < 17:
    new_card = d.draw()
    dealer_card.append(new_card)
    card_index = d.cards.index(new_card)
    dealer_score += d.values[card_index]

    # Checking if we busted
    if dealer_score > 21:
        # Do we have an ace?
        if 'A' in dealer_card:
            a_index = dealer_card.index('A')
            dealer_card[a_index] = '1'
            dealer_score -= 10

if player_score > 21:
    final_scores['dealer'] += 1

elif dealer_score > 21:
    final_scores['player'] += 1

elif dealer_score > player_score:
    final_scores['dealer'] += 1

elif player_score > dealer_score:
    final_scores['player'] += 1

elif player_score == dealer_score:
    final_scores['tie'] += 1
```