Machine Learning Model for the Casino Game Blackjack

Dylan Fair

dfair@uiowa.edu

Abstract: The objective of creating a blackjack model is to see how it too would compete against the dealer. The proposed solution uses similar inputs as a player would use but also adds the power of computing to increase accuracy on the count (A way to know how many high vs low cards are left in the deck). The goal is to see if a machine learning algorithm has a chance to beat the dealer or to be better than the basic strategy.

Introduction: The game of blackjack has been statistically solved using a method of basic strategy (the current best strategy for blackjack includes doubling and splitting cards), this method still gives the dealer a statistical edge over the player. With the computational power of a machine learning model, the goal is to see what new strategies the model can output, and how these models do vs each other and the known solutions of today. Our model will not be able to split (when given two of the same card values split them into their own hands) or double (double your bet to receive one more card, the player cannot hit after doubling).

Training Data Factors: For the two models presented today they vary from a more complicated strategy vs the dealer's strategy. The complicated strategy considers the current count and plays different strategies based on it. This is considered a deviation, where when the count is high (more higher value cards) a no bust strategy is played where the player doesn't hit over 11 ensuring they can't go over 21. When the count is low the player is more aggressive and hits more up until the hand value of 18. If the count is neutral the decision is made based off basic strategy, which includes not hitting when the dealer is showing a 5 which is statically the worse hand. This resulted in a win percent of 42 which did not include ties. The next decision process is the same as the dealer where the player hits until a value of 17 or higher is reached and never hits on a 17 or above.

Model Features: The model can only know what a player would know. Through running our simulations to generate training data, data was collected that the model could not know like the dealer's final hand value if they busted and other metrics. With this in mind, the model is given the dealer card value, the players hand value, whether the player hit or not, the current count the hand started at using a high low true count (where 2-6 count as +1, 7-9 count as 0, 10-Ace are -1, these are added together for the cards that have been dealt and then divided by the number of decks left in the shoe to give the true count), and the number of hits the player has taken. The dealer card value and player hand value are basic information that any player would know, while basic strategy uses the actual card types due to the ability to split and double. The hit metric is used in training the model so it knows if the player hit or not it is set to 1 for hit and 0 for stay but does not show if the player hit more than once, when using the model, the hit is set to 1 (the reason for this will be explained later). The true count is added as the decision process used in the simulations is used to count to know when the dealer has higher bust (go over 21) odds and when the player is safer to hit. Finally, the number of hits was added as the hit metric didn't handle cases where the player hit too many times (this will be discussed in further detail later on). The model is trained on an outcome that tries to predict what the player should have done. This is generated based on the following table.

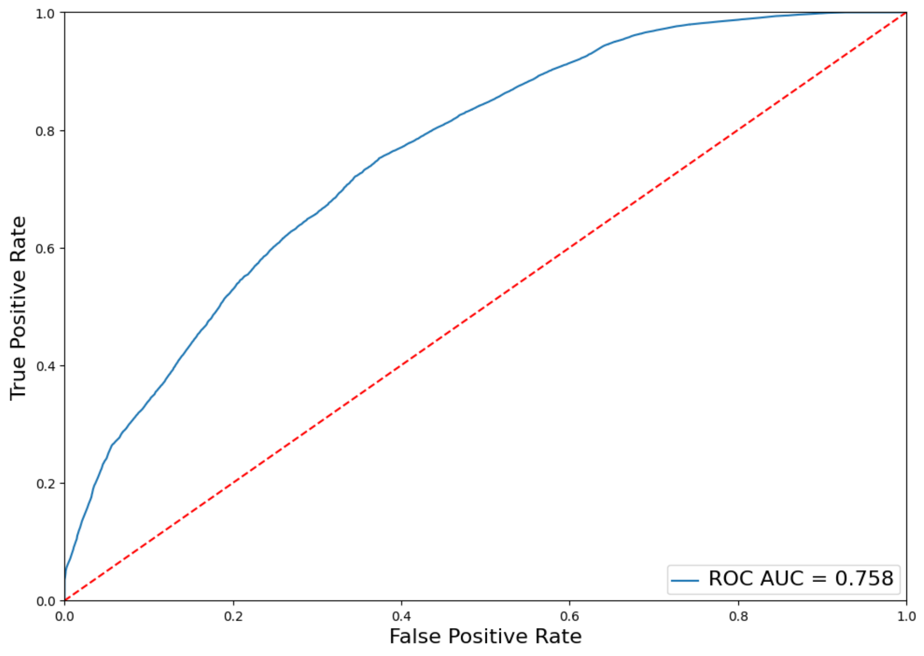
Outcome decision	**Hit**	**Stay**
Won	Hit (1)	Stay (0)
Lost	Stay (0)	Hit (1)

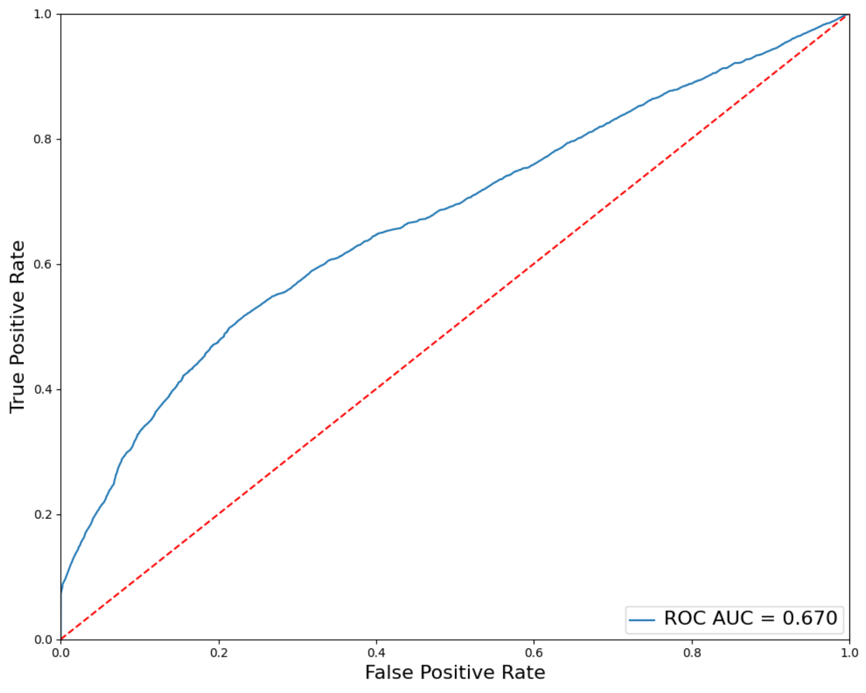
Where if the player made the correct decision that is the label whereas if they made the incorrect decision then the opposite is the label. This is because we are using binary cross entropy to train the model.

Training Method: The model was trained on a sequential model with 5 layers and three different functions inside to fit the data. The function at layer one was a Relu with 16 nodes, SoftMax at layer 3 with 32 nodes, and sigmoid at layer 5 with one node to make the data binary. These functions are activation functions that help to add non-linearity to the data. This was run with 30 epochs (each is a pass through the training data) and a batch size of 512 (the number of samples to work through before updating the weights in the model). The average time per epoch was 3 seconds. These parameters were chosen due to efficiency, not over-training, and loss rate. The training data was split 80 percent for training and 20 percent for testing.

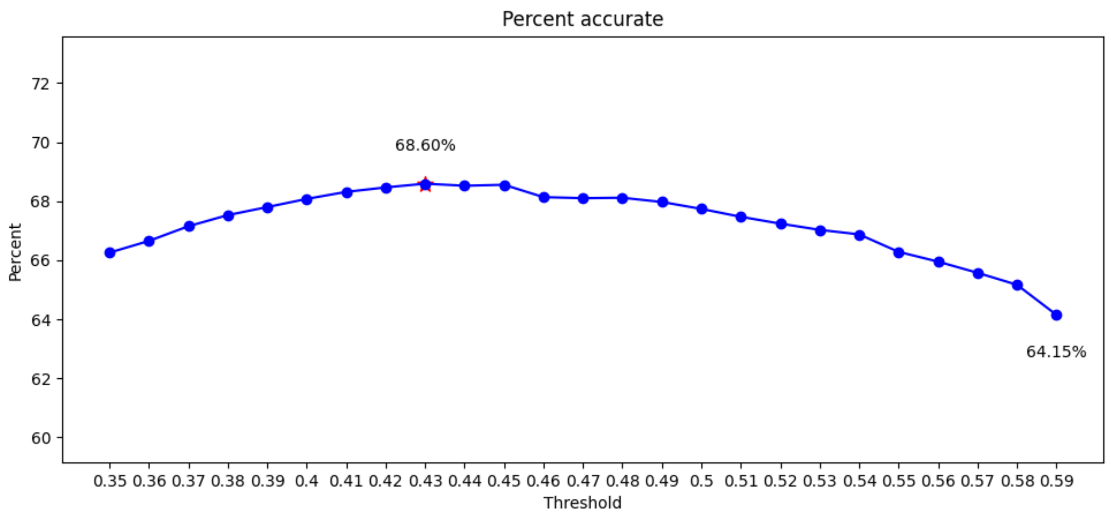
Model Output: The model outputs a value 0-1 which can be interpreted as probability. The input parameters are the dealer's current card value, the player's hand value, the hit which is set to 1 to say the player did hit, and the current count and number of hits. This output shows the probability the player wins given they hit with the situation being the other parameters in the model. This output is then translated to a 1 or 0 based on a threshold. This threshold is set to where any number higher is set to 1 (hit) and any number low is set to 0 (stay). We will explore how different thresholds change the performance of the model later.

Model Evaluation: The model evaluation is based on the remaining training data after the training has been completed. We will compare the results of two models. One uses a high low count to hit when the count is low (more low cards in the deck) and stay when the count is high (more high cards in the deck) especially when the dealer has a bust possibility (is showing a card 2-6) vs a model that was trained on data that used the same decision process as a dealer (hit until the hand value is greater than or equal to 17). We will name these models high-low and dealer from here on out. The first metric to look at is the True Positive (model predicted true and the correct outcome was true) vs the False Positive (model predicted true but it was actually False) and the area under the curve AUC, where a .5 AUC would be as good as a random model, and a number higher shows the ability to distinguish. Here is the high-low model graph followed by the dealer model graph.

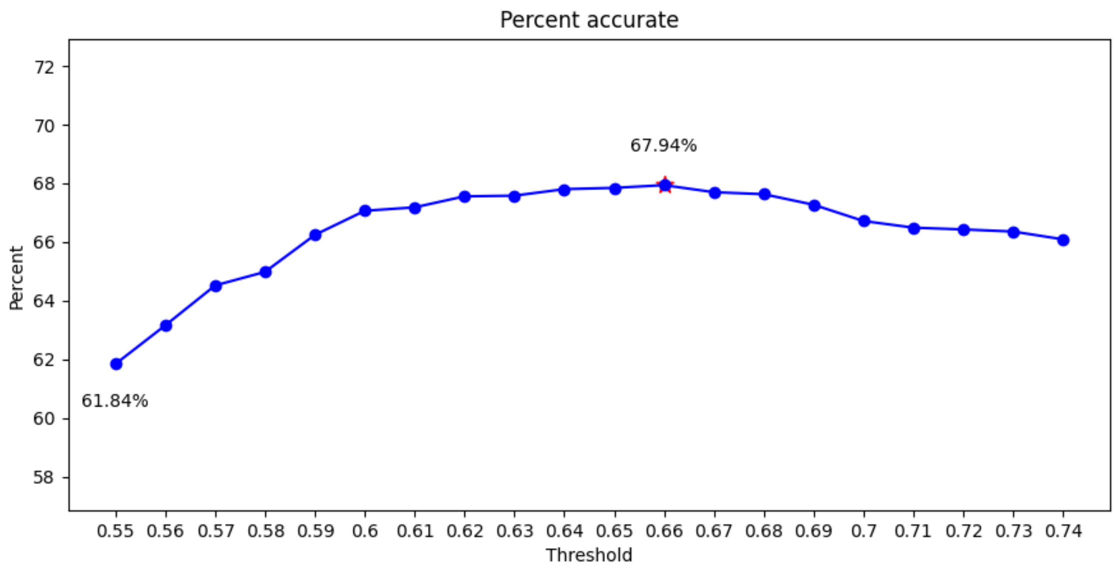




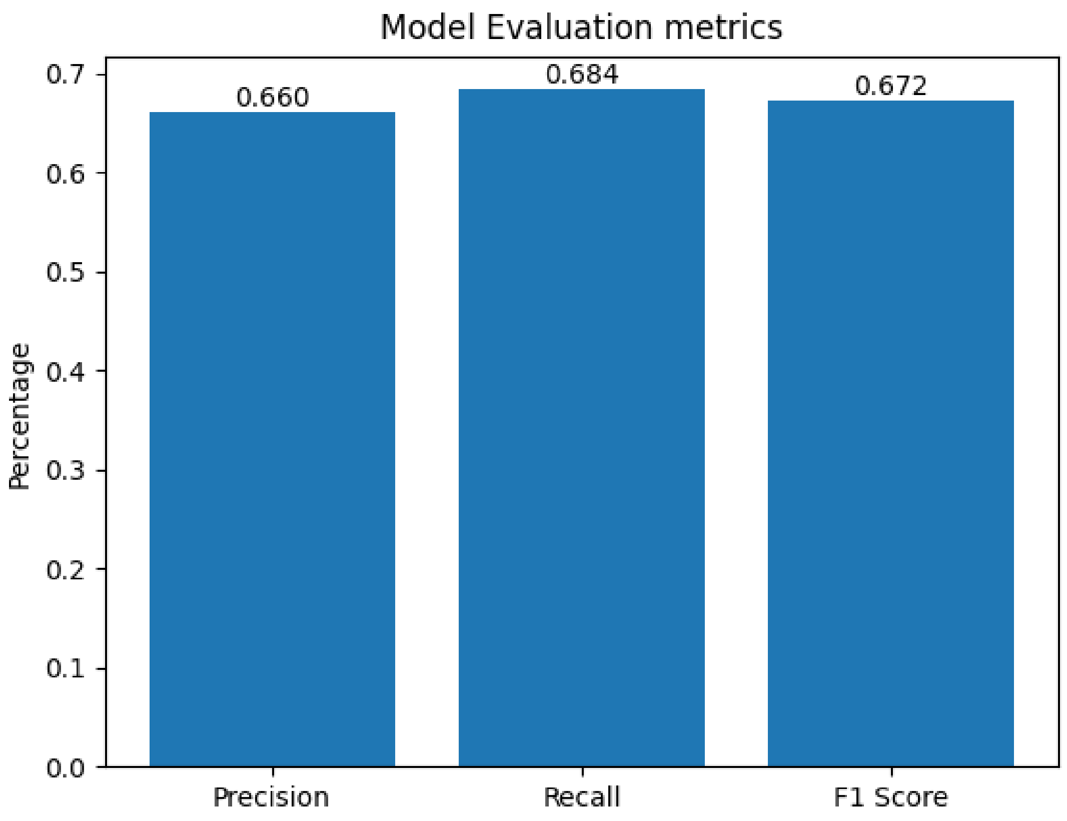
As you can see the high-low model graph shows that the model has a greater ability to classify data and predict better than the dealer's model. This shows that the high-low model has a better ability to classify data and make better predictions based on that data. The next metric to look at is their accuracy at different thresholds (this number is used to determine what is a hit and what is a stay). Here is the high low model accuracy at different thresholds.



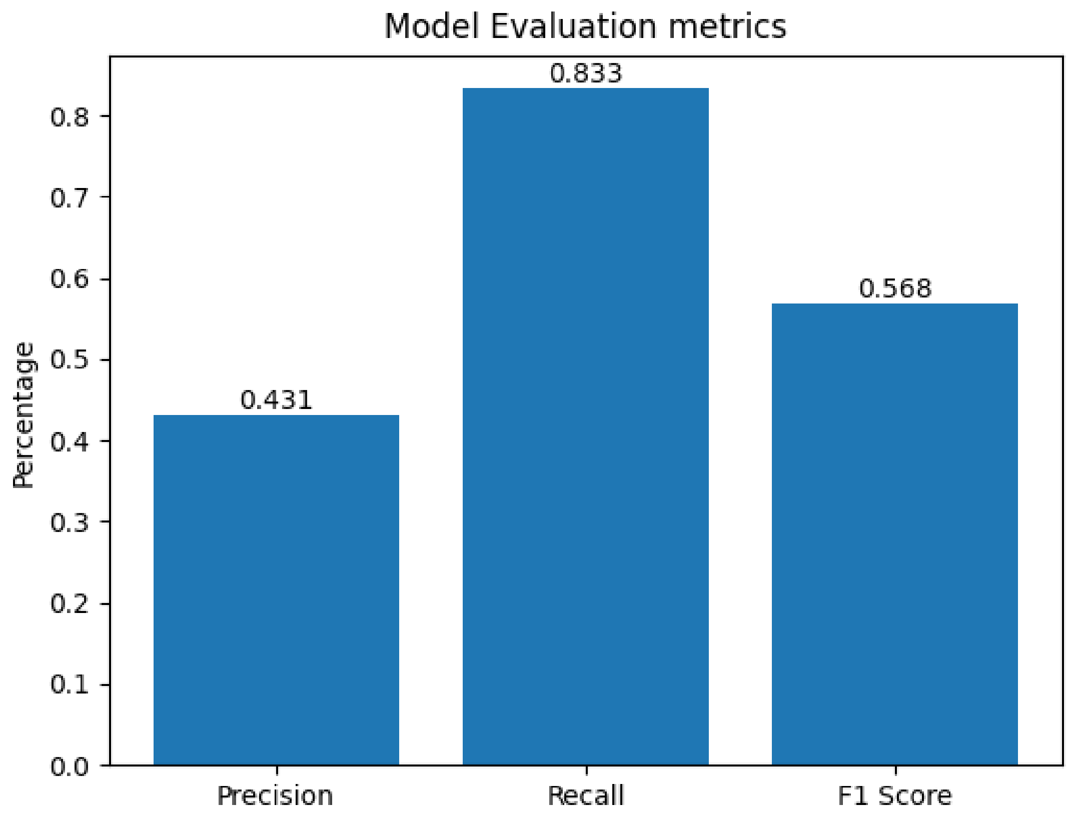
As you can see the max accuracy is at a threshold of .43 with a 68.6% accuracy rating. Now let's look at the dealer model to see the difference.



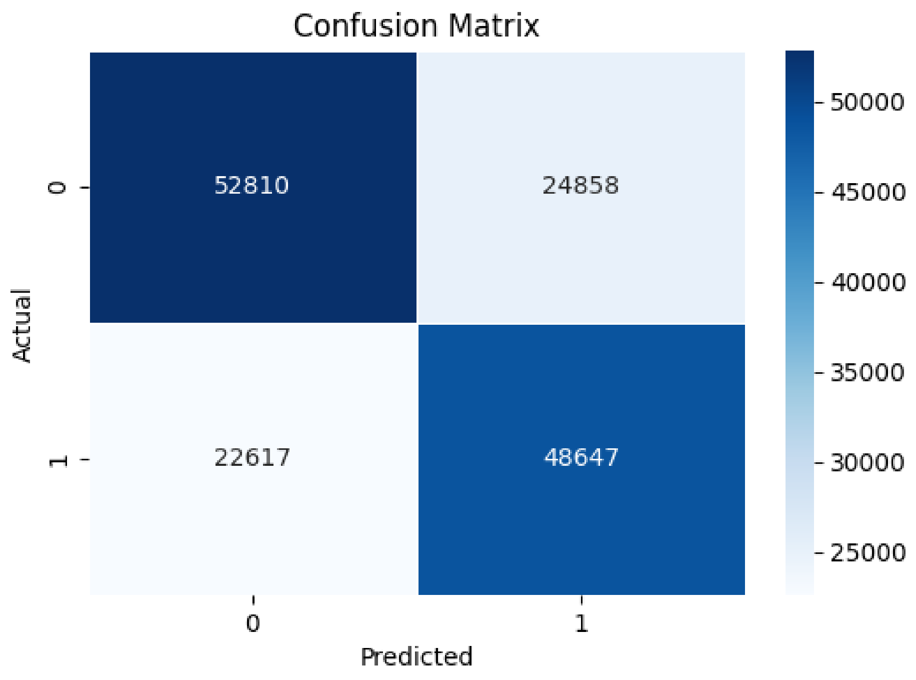
Here the best threshold is set to .66 with an accuracy of 67.94% which is less than one percent off the high-low model. The major difference here is the threshold. This can be explained as the dealer hits more often than the model which can look at the count and make decisions to stay more often so when it does hit it's more likely to win. This threshold also shows that the dealer's model on average outputs higher values than the high-low. Next, we can look at the Precision which looks at the number of true positives (the model predicted true, and the correct outcome was true) divided by the number of true positives plus the number of false negatives (outcomes guessed were false but the correct choice was true). The Recall shows all the positive instances and how many did the model correctly predict as positive. Finally, the F1 score which is the harmonic mean of the precision and recall, shows when there is a good balance between precision and recall. Here is the graph for these metrics for the high low model.



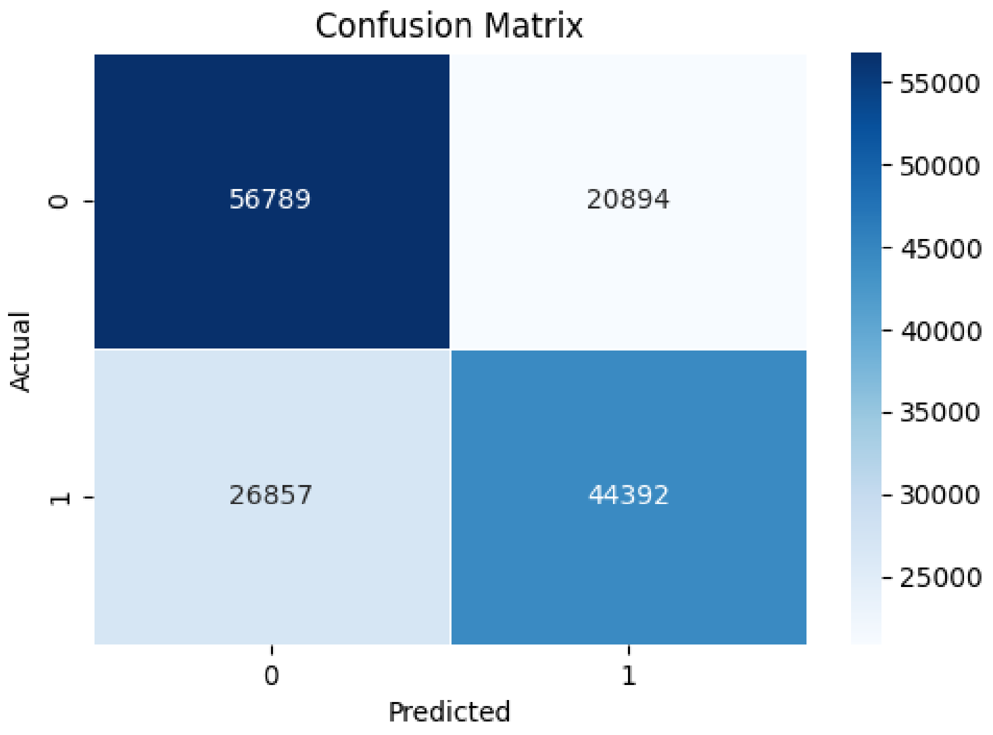
As you can see these numbers are not perfect (which would be a 1) but also not bad (near 0). As the training data is not perfect and the outcome is also not perfect (there are lose-lose situations where both hit and stay would lose) but their numbers being close is a good metric. Here is the graph for the dealer's model.



Here we see a huge difference in the precision and the recall. For precision this means that when the dealer model predicts hit it is most likely wrong, where the recall shows that the dealer model likes to predict one. Given this information we can see the dealer hits too often and this matches with the accuracy threshold as well. Whereas the high-low model when it predicts true is more likely than not correct. Seeing this data lets next look at the confusion matrixes for each model. These confusion matrixes were generated using the max accuracy threshold from the graphs above of .43 for the high-low model and .66 for the dealer model. Here is the high-low models confusion matrix.

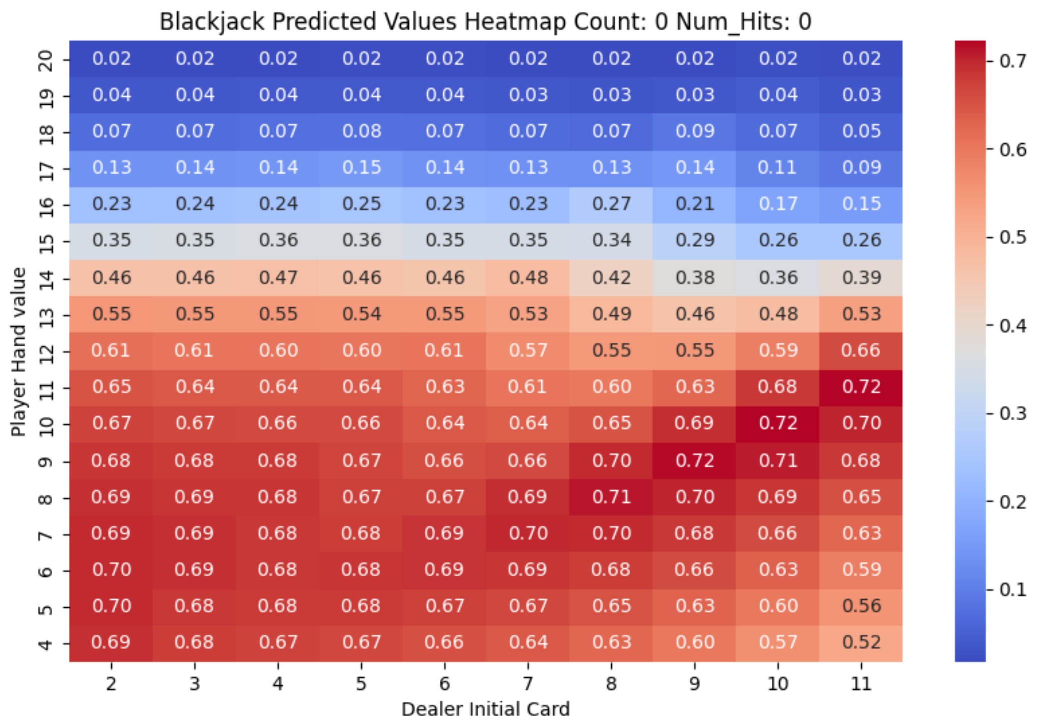


Here we can see the high low model is fairly accurate at predicting the values and has an even distribution between predicting hit and stay correctly vs predicting hit and stay incorrectly. Next let's look at the dealer model's confusion matrix with the .66 threshold.



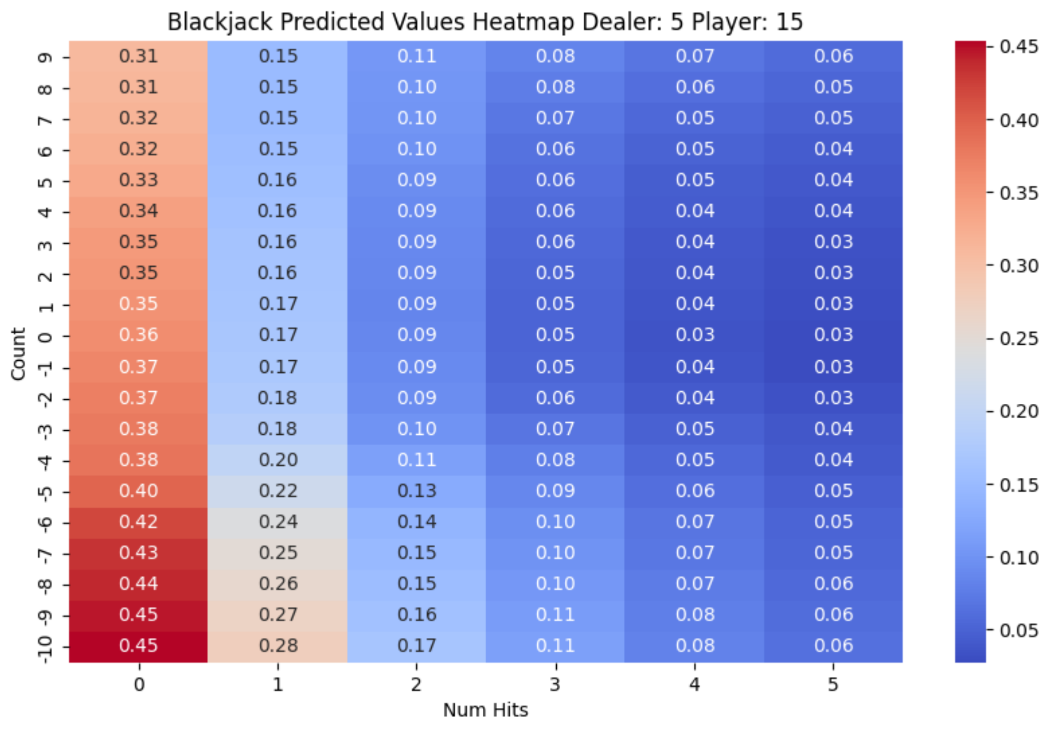
Here we can see the dealer predicts staying about 10,000 times more often than predicting the stay and is wrong roughly 69 percent of the time. While when the dealer predicts a hit, it is right double the amount then its wrong.

Output Value Evaluation: While all these metrics help us evaluate the model, the data collected can only be as accurate as the testing data we provided. Looking at the output values for different scenarios will help better understand the model's decision process given a situation imputed. First let's look at a table of all output values based on all possible player hand values vs each dealer card value. Here is the table for the High-Low Model.



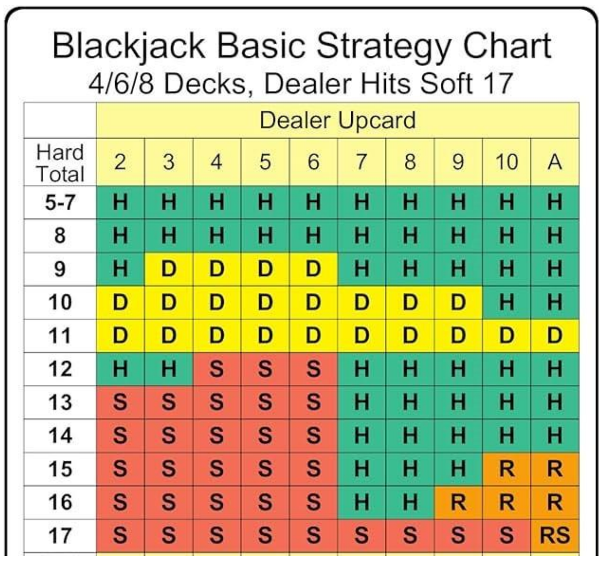
Player Hand value	2	3	4	5	6	7	8	9	10	11
20	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
19	0.04	0.04	0.04	0.04	0.04	0.03	0.03	0.03	0.04	0.03
18	0.07	0.07	0.07	0.08	0.07	0.07	0.07	0.09	0.07	0.05
17	0.13	0.14	0.14	0.15	0.14	0.13	0.13	0.14	0.11	0.09
16	0.23	0.24	0.24	0.25	0.23	0.23	0.27	0.21	0.17	0.15
15	0.35	0.35	0.36	0.36	0.35	0.35	0.34	0.29	0.26	0.26
14	0.46	0.46	0.47	0.46	0.46	0.48	0.42	0.38	0.36	0.39
13	0.55	0.55	0.55	0.54	0.55	0.53	0.49	0.46	0.48	0.53
12	0.61	0.61	0.60	0.60	0.61	0.57	0.55	0.55	0.59	0.66
11	0.65	0.64	0.64	0.64	0.63	0.61	0.60	0.63	0.68	0.72
10	0.67	0.67	0.66	0.66	0.64	0.64	0.65	0.69	0.72	0.70
9	0.68	0.68	0.68	0.67	0.66	0.66	0.70	0.72	0.71	0.68
8	0.69	0.69	0.68	0.67	0.67	0.69	0.71	0.70	0.69	0.65
7	0.69	0.69	0.68	0.68	0.69	0.70	0.70	0.68	0.66	0.63
6	0.70	0.69	0.68	0.68	0.69	0.69	0.68	0.66	0.63	0.59
5	0.70	0.68	0.68	0.68	0.67	0.67	0.65	0.63	0.60	0.56
4	0.69	0.68	0.67	0.67	0.66	0.64	0.63	0.60	0.57	0.52

This heat map gives an insight as to what the threshold should be to hit and stay and how the model reacts to different scenarios. The count and number of hits were set to 0 each for this table. This table also represents our training data which stayed on 16 and below when the count when 0, while the decision process for the training data didn't consider the number of hits this metric helped improve the performance of the model so let's see how that value changes the outputted value for the model.



Count	0	1	2	3	4	5
9	0.31	0.15	0.11	0.08	0.07	0.06
8	0.31	0.15	0.10	0.08	0.06	0.05
7	0.32	0.15	0.10	0.07	0.05	0.05
6	0.32	0.15	0.10	0.06	0.05	0.04
5	0.33	0.16	0.09	0.06	0.05	0.04
4	0.34	0.16	0.09	0.06	0.04	0.04
3	0.35	0.16	0.09	0.06	0.04	0.03
2	0.35	0.16	0.09	0.05	0.04	0.03
1	0.35	0.17	0.09	0.05	0.04	0.03
0	0.36	0.17	0.09	0.05	0.03	0.03
-1	0.37	0.17	0.09	0.05	0.04	0.03
-2	0.37	0.18	0.09	0.06	0.04	0.03
-3	0.38	0.18	0.10	0.07	0.05	0.04
-4	0.38	0.20	0.11	0.08	0.05	0.04
-5	0.40	0.22	0.13	0.09	0.06	0.05
-6	0.42	0.24	0.14	0.10	0.07	0.05
-7	0.43	0.25	0.15	0.10	0.07	0.05
-8	0.44	0.26	0.15	0.10	0.07	0.06
-9	0.45	0.27	0.16	0.11	0.08	0.06
-10	0.45	0.28	0.17	0.11	0.08	0.06

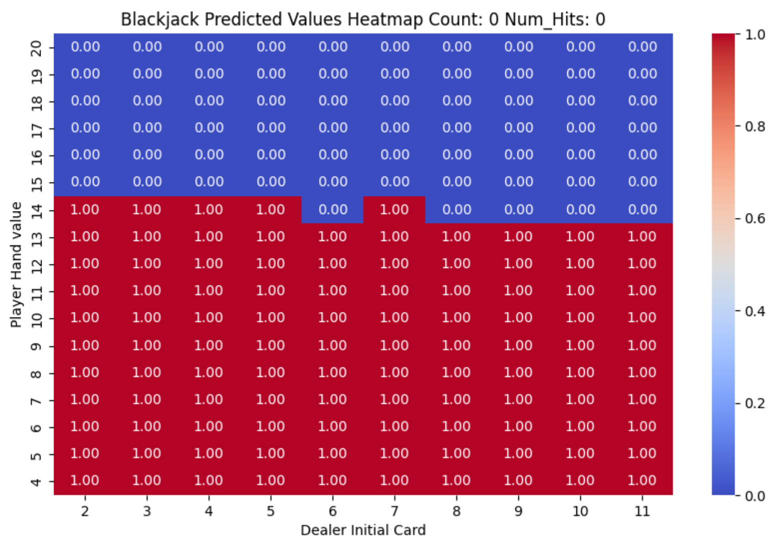
The dealer starts with a card value of 5 and the player has a hand value of 15. When the count is set to 0 the change based on the number of hits from 0 to 5 is .33. This change shows how much the model thinks the player will lose if the number of hits rises. While the count does have an effect on the output value the change when number of hits is 0 from a count of -10 to 10 is .14. This would make sense as when the count gets higher the player has a higher chance of busting, and both the dealer and player have a higher chance of receiving a higher initial hand value. Next, we can use the max accuracy threshold of .46 to build a hit and stay table and compare it to a basic strategy table. Here is the basic strategy table.



Blackjack Basic Strategy Chart
4/6/8 Decks, Dealer Hits Soft 17

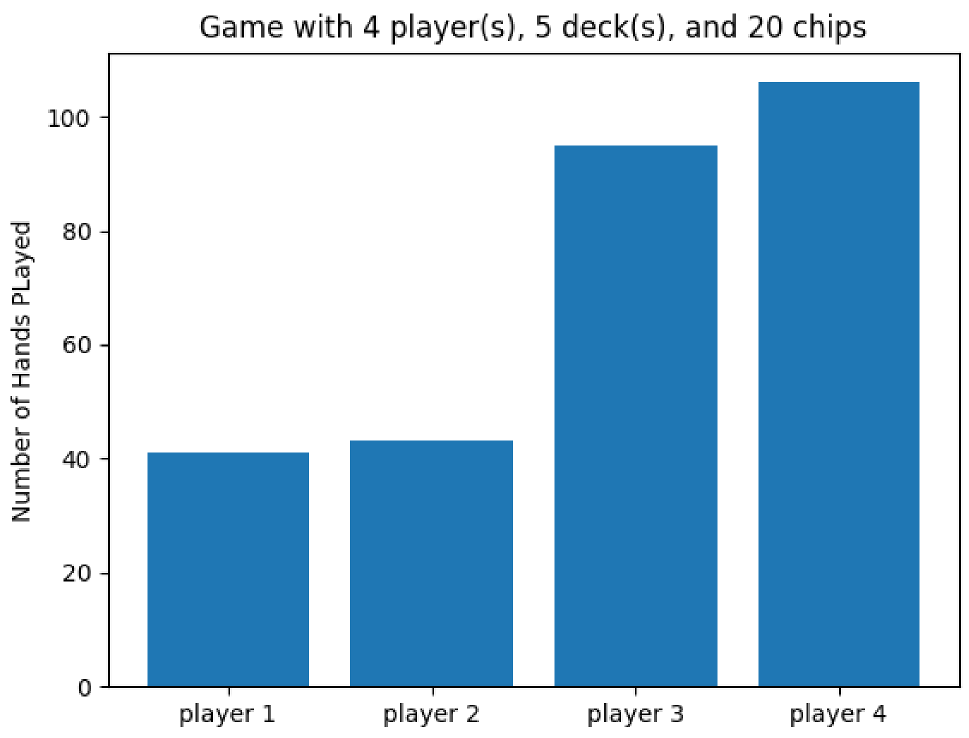
	Dealer Upcard									
Hard Total	2	3	4	5	6	7	8	9	10	A
5-7	H	H	H	H	H	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H
9	H	D	D	D	D	H	H	H	H	H
10	D	D	D	D	D	D	D	D	H	H
11	D	D	D	D	D	D	D	D	D	D
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	R	R
16	S	S	S	S	S	H	H	R	R	R
17	S	S	S	S	S	S	S	S	S	RS

For the purpose of this discussion, we can assume that all doubles (Shown as a D) are just taken as a hit as are Rs in the basic strategy table. Now let’s look at our table.

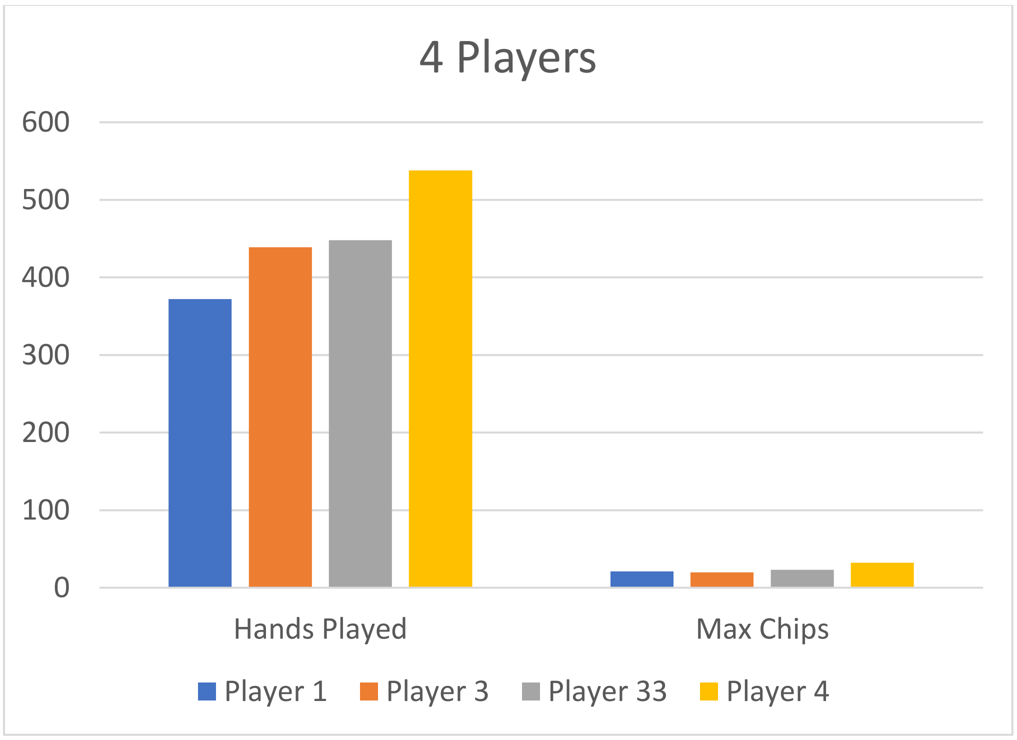


The main difference we can see is that the model’s strategy doesn’t hit on 14-16 when the dealer is showing a high card. The model also likes to hit on 12-14 when the dealer is showing a lower card.

Testing in a Real Game: While we can see our strategy differs from what’s proven as best, we can look at how our two models perform in a real game of blackjack. The dealer never hits on a 17 or above. Each simulation is run with 5 decks, 4 players each playing their respective models, and each player starts with 20 chips. A win adds a chip, a loss takes a chip, a tie does nothing, there is no bonus for a blackjack (21 from the initial hand value), and a player is removed when they run out of chips. Let’s see first how our dealer model did.



At the low end the worst player played 42 hands while the best player played 104 hands. The best player at one point had a max chips of 24. While these statistics are not bad let's see how the high low model does.



Here the worst player played 368 hands while the best player played 538 hands and had 32 chips at their max. This shows that the best player in the high-low model vs the best player in the dealer model played over 5 times as many hands. Even looking at the worse player vs the worse player the high-low model played almost 9 times as many hands. This is where the high low model stands out is in a real situation.

Conclusion: Based on comparing two different models we can see the performance of the high-low model far passes the performance of the dealer model. This is as expected as the dealer follows a single rule which doesn’t allow for variety of training data. We can also see that the model’s performance based on the remaining training data isn’t perfect, but we would not want it to be as we want the model to be better than the training data. With blackjack there is also a perfect decision or a perfect strategy so allowing the model to have room for interpretation can allow it to perform better as seen when it was able to player over 500 hands on just 20 chips. The evaluation showed that more variety could be better in this situation and possibly a better way to represent the data. Possible using each decision the player made instead of a singular hand which can contain multiple decisions.