BEATING VEGAS:

Using Statistics to Make Better NBA Bets

Nathan Hemenway, Josh Rogers, and Michael Bauers

Colorado State University Statistics Department

Instructor: Aaron Nielsen

Introduction

Sports betting has seen an explosion of popularity in recent years, with it now legalized in around 30 states. It has grown to become a 3 billion dollar industry with millions of participants. Most people bet on sports for fun, but others use statistical methods in order to make it a profitable side hobby. Basketball is a popular sport to bet on, and the focus of our project. Betting on NBA games makes up a large part of the sports betting industry. The only sporting events that see more betting activity than the NBA Finals are March Madness and the Super Bowl. NBA fans collectively put up at least 5 million dollars per game during the NBA Finals (Legal Sports Betting).

One of the simpler types of betting is to bet which team will win according to a moneyline. Before a game, a sportsbook will publish what they believe the odds, also called the moneyline, are for a given team matchup. As more people bet leading up to the game, the moneyline will shift until finalized at the game's start. The sportsbook's goal is to distribute the odds in a way that makes them money, and they accomplish this by skewing the payout based on the odds. If one bets on a team that is favored, the payout will be significantly less than the payout of an "underdog" team less likely to win. This is because it is generally harder to pick a winning underdog than to pick a winning favorite. Finding and leveraging patterns in NBA data to make both educated and profitable betting decisions is what we aim to accomplish in this project. Our primary outcomes of interest will be the total profitability of our models as well as precision, which is how often our decision to bet is correct.

Our final strategy was to fit two separate models, one for picking underdog team bets and another for picking favorite team bets. We ended up switching to this strategy after our initial attempt of betting on our predicted winner for each game was not profitable. This strategy of separating underdog and favorite picks was more ideal because instead of just classifying each game as a home team win or loss, we could intelligently choose thresholds for what made each type of pick most profitable.

The data we used to fit our models came from Foxsheets. Foxsheets gives detailed statistics for both teams playing in a game, as well as betting information. Some of the types of team statistics include, but are not limited to: wins in season, wins against the spread, and current season statistics, as well as previous game performance metrics. This can show if a team has been playing well recently or are experiencing a cold streak. The betting information given includes money lines, point spreads, and total lines. This dataset tries to encompass all measurable aspects of the game, at a team level, that can lead to a team winning a basketball game. Our advisor for this project, Connor Gibbs, scraped this data from Foxsheets and allowed us to use it for our project.

Data Preparation

The original data set had two rows for each game, with each row giving statistics for each team that played in that game. It is hard to fit models on data with two rows corresponding to one outcome, and in this case the outcome was who won each game. We came up with two solutions on how to make the data easier to work with.

The first method we came up with was what we dubbed the "wide data". This is one way we turned data with two outcomes per game into one outcome. This was a binary outcome telling us whether or not the home team won each game. This was found by comparing the scores of the home and away teams. Obviously, if we only have one outcome per game, we

cannot have two rows for each game. The way we dealt with this issue was by widening the original data. If we originally had two rows for one game, each giving us statistics for one of the teams, our new data set had one row that had statistics for both teams, and told us whether or not the home team had won that game. There were two categories of data present, statistics representing the home team and the same statistics but for the away team. For example, for opponent statistics, we have information about the home team's past opponents as well as the away team's past opponents.

The second method we came up with was what we called the "difference data". This is another way we transformed the two rows per game into one row. This was done by subtracting the away team data from the home team data. This means that if we have a positive value for one of our transformed variables, then the original statistic for the home team was greater than that of the away team, and vice versa for negative values. This left one row with the same number of predictors as before, but now they account for both teams' statistics. This allowed us to input one row of data into our models to receive one output: whether the home or away team won the game. We found that the data values could end up very small so we standardized the differences and saved those values as an additional dataset.

The wide data was what we ended up using to fit our final set of models. We used this data for our multivariate regression model and wanted to be able to connect our variable importance information to our classification models. However, we made some changes prior to model fitting. Once we switched strategies from predicting whether the home or away team would win, to having two outcomes of interest - whether the underdog or favorite won, we modified our data accordingly. The new data told us who the underdog and favorite teams were, as well as giving us two binary outcomes for both if the underdog won and if the favorite won.

After we had transformed the data into a form that we could use to fit models, we proceeded to clean the data. The first step we took was removing obviously unimportant variables, which included the url, rotation, and place. Any variables giving information on dates or team names were also removed, so we were left with only numeric data. Next, since we pivoted the data to include both teams' statistics, we made sure there were no duplicate columns by removing any that existed.

The next problem that presented itself was collinearity, since we had over 400 variables in our dataset. Since the primary goal of our analysis was to maximize predictive power and profitability, we took a very conservative approach in what we removed. To deal with the collinearity, we removed variables that were at least 95% correlated, since these variables gave us essentially the same information. This left a bit of leftover collinearity, with about 2% of our data being at least 60% correlated. We saw this as acceptable since we were mostly interested in prediction. Usually you do not want low variance variables going into your models, but removing any significant amount of low variance data resulted in decreased model performance, so we opted to leave these variables in our dataset.

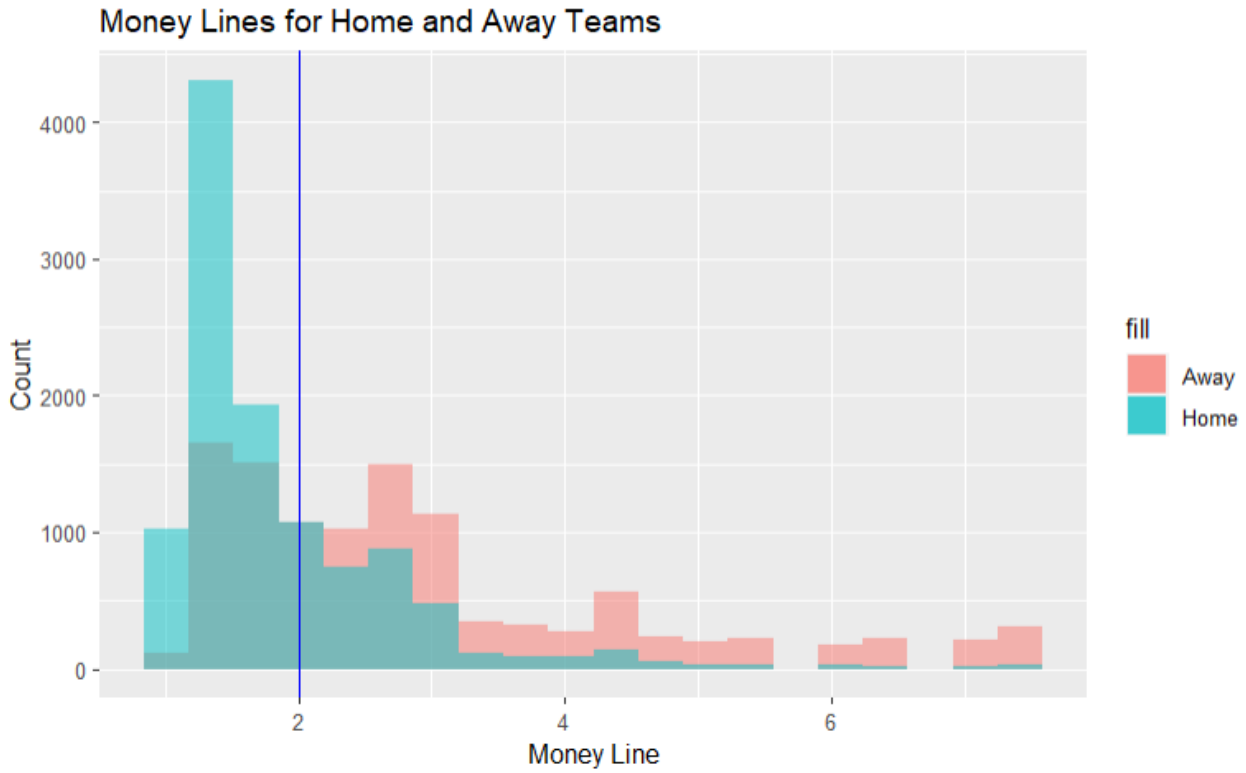
Missing data was present in our dataset, and we dealt with this in two ways. The first was removing columns that had at least 5% missing data, which was mostly columns related to playoff and division statistics. The second way was using the mice R package to perform data imputation on the remaining missing values. Mice creates multiple imputations by predicting the missing values based on the values in the other columns. This is preferable to imputing the same value, such as the mean, for all missing values in a column because it can more

accurately predict what the missing value might have been. When using the mice package on our dataset, we used one imputed dataset with 5 max iterations and predictive mean matching as the imputation method.

These missing values were mostly from early season games that did not have all the previous game statistics yet. By imputing these missing early season statistics, we are making the assumption that they are similar to the later games, and have not changed significantly over time. While we are adding bias to our data in favor of the games late in the season, we get the added benefit of more information going into our early season predictions. For example, a great team might have their star player get injured, which would typically negatively impact the team's future performance. This is something the data imputation does not account for.

To test our models, we randomly split up our data into an 80/20 training test split using the sample function in R. We set a seed beforehand for reproducibility and used the same data across our different models to ensure comparability. All code can be found on GitHub.

In the plot below, we compare the money lines for home and away teams. It appears that away teams tend to have higher moneylines than home teams do, which means away teams are more often the underdog. The vertical line at two gives the border that separates the favorites and underdogs, so if a team has a moneyline of at least two, then they are the underdog for that game. This is because the moneyline gives you the multiplier on your initial bet should you win. For underdog bets, you at least double your money if you win, since you are taking a more risky bet. Conversely, for favorites you can make up to just less than double your money.



Methods and Models:

1. **Multivariate Regression**

The multivariate regression model took the form:

$$\begin{bmatrix} y_{1,1} & y_{1,2} \\ \vdots & \vdots \\ y_{n,1} & y_{n,2} \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,q} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n,1} & \dots & x_{n,q} \end{bmatrix} \begin{bmatrix} \beta_{0,1} & \beta_{0,2} \\ \vdots & \vdots \\ \beta_{q,1} & \beta_{q,2} \end{bmatrix} + \begin{bmatrix} \epsilon_{1,1} & \epsilon_{1,2} \\ \vdots & \vdots \\ \epsilon_{n,1} & \epsilon_{n,2} \end{bmatrix}$$

Where the $y_{i,1}, y_{i,2}$ are the respective scores we have for the underdog and favorite teams, and are known outcomes when fitting the model. We have a matrix of predictor variables, which are fixed values we know prior to model fitting. We also have a matrix for the beta coefficients corresponding to each score, where the first column corresponds to the underdog score and the second column gives the coefficients for the favorite score. Our beta coefficients are fixed but unknown effects that our model is estimating. We also have a matrix of error terms, which are normally distributed random variables. For our subscripts, we have n =11078 and q = 427. The assumptions for a multivariate regression are similar to those of a typical linear regression model, except for two response variables. So we have that both $\epsilon_{1,1}, \dots, \epsilon_{n,1}$ and $\epsilon_{1,2}, \dots, \epsilon_{n,2}$ are both uncorrelated and normally distributed with mean 0 with constant variance. Also, the relationship between predictors and score is assumed to be linear. Put simply, the model performs a linear regression for both underdog and favorite score, with the addition of a covariance matrix for the beta coefficients. This covariance between coefficients allows us to find out which predictor variables are contributing to both response variables via multivariate analysis of variance. This is the main reason we fit this model, as it gives us an idea of what goes into a team performing well. The main focus of our analysis was on prediction accuracy, but we also wanted some idea of variable interpretation and importance, which this model helped provide via MANOVA.

2. **Logistic Regression**

Logistic Regression was the first classification method we employed when trying to pick bets for underdogs and favorites. Logistic regression was an obvious choice of model to fit since it is a simple to implement classification method. Logistic regression works by regressing the log odds of success, where the odds are defined by $\frac{p}{1-p}$ and p is the probability of success, so in this case p would be the probability of a team winning. We fit the model: $log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q$ where each $x_i : i = 1, \dots, q$ is one of the predictors in our data. The first assumption made for our logistic regression is that we have a binary response, which was the case for both the underdogs and favorites models. Another assumption was that each game was independent, which was again the case. One last assumption is that there is no collinearity between predictors.

We fit two logistic regression models, one for predicting the underdog winning and another for the favorite winning. To pick profitable bets, we needed to know the probability the underdog or favorite was to win each game. If you use the predict function in R, you get

predicted probabilities of success, which makes it simple to classify games as bettable or not according to different probability thresholds that we will later justify.

3. **Neural Network**

Neural networks are a generalizable machine learning framework useful for a wide array of problems. Depending on the complexity of data, neural networks can be useful as they allow for higher dimensionality in modeling. In addition, for patterns difficult to find in the data, adding non-linearity allows us to potentially model them more accurately. For our purposes, we can set up a neural network to perform binary classification and predict a win or loss for the underdogs and favorites. To accomplish this, we utilize python and the pytorch library (the code is adapted from Nikhil Krishnaswamy from the CS 445 course).

The neural network setup consists of three layers. The first is a simple linear input layer that takes in our input data and the output of subsequent layers. The next is the hidden layer, which adds nonlinearity to the weight adjustments. There are many options, but we chose to use tanh and relu as our functions. Finally, we added a dropout layer, which is a regularizing feature that aims to help generalize the model and prevent overfitting (Prasad, 2020; Verma, 2020).

Similar to logistic regression, we can configure the output of the neural network to give us the probabilities of success that we can apply a threshold to. To accomplish this, we run the final output through a sigmoid function, which transforms the output to between 0 and 1.

4. **XGBoost**

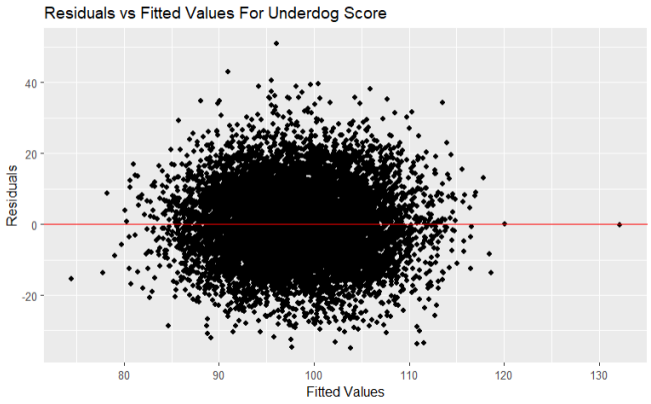
XGBoost stands for Extreme Gradient Boosting. This is a distributed algorithm that uses boosted decision trees. Boosting is used to make decision trees by relying on previous trees for information. The trees are built in a stepwise fashion using a loss function to assess error in each step. XGBoost can be used for a number of different applications, but we used it for classification purposes. Within the XGBoost library, there are a number of different hyperparameters that can be helpful with tuning the model. We chose to focus on max depth and nrounds. The max depth parameter refers to the maximum size of the decision trees and the nrounds parameter refers to the number of trees that are generated. We decided on the values of these hyperparameters based on research we performed as well as what combination maximized training precision. These turned out to be a max depth of four and nrounds of 25. There was an option to have the model give binary i.e., classification/logistic outputs. This means the output is the probability of success with a value between 0 and 1. We chose to do this so we can find the probability threshold as well as profitability the same way as the other methods. Another reason we decided to use XGBoost was that it is known for dealing with collinearity well, and our data contained a degree of collinearity.

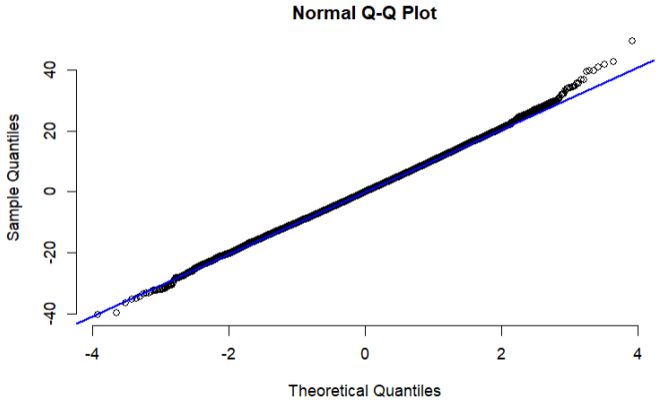
Model Assumptions and Picking Probability Thresholds:

1. **Multivariate Regression**

In the plot of residuals vs. fitted values for the underdogs outcome, the residuals appear to have mean zero with constant variance. While not pictured, this was also the case for favorites. The fit appears to be linear. The QQ plots of the residuals for both underdog and favorite scores gives enough evidence for normality (favorites pictured). While there is a bit of deviation from normality at the tails, this is not enough to be worrying. To test for correlation, we ran a Durbin-Watson test, which resulted in a p-value of 0.233 against the null hypothesis that

the correlation is zero, which is not enough evidence to say the residuals are correlated. Thus all the assumptions for the multivariate regression model were met.



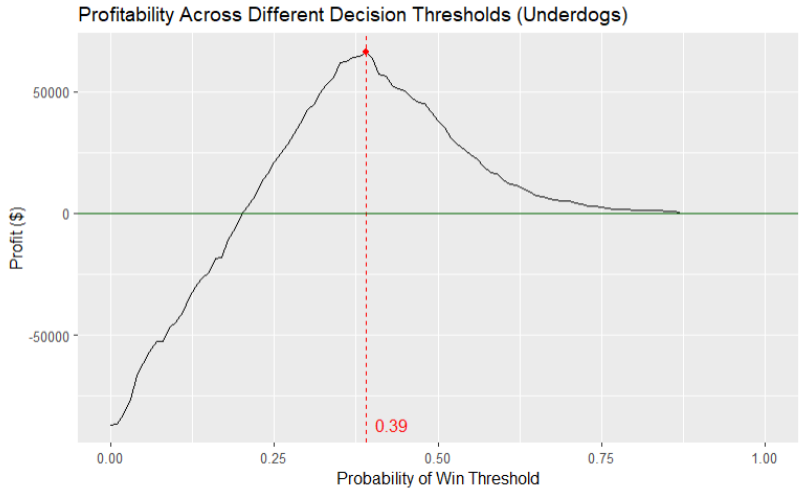


2. Logistic Regression

To deal with the logistic regression assumption of no collinearity between the predictor variables, we removed data that was 95% or more correlated. This left roughly 2% of the data that was at least 60% correlated. This means that we are bending our assumptions somewhat, but we are primarily focused on predictive power and not inference. So any overly large standard errors are not of huge concern.

3. Choosing a Probability of Success Threshold to Intelligently Pick Bets

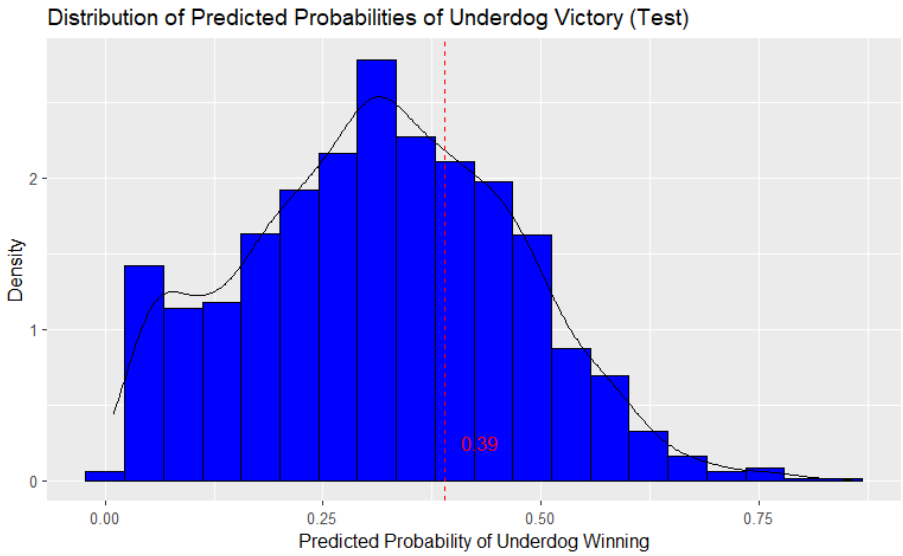
We used the predict function to find the probability that the underdog and favorite would win each game in our separate models. This gives us the opportunity to pick a probability of winning threshold that we consider good enough to bet on. This is going to be different for underdogs and favorites, because we have more upside on underdog bets and less on favorites. Thus our threshold for a favorite win must be higher than that for the underdog, because we will need our picks to be correct more often to compensate for the smaller amount of money made per bet. To choose these thresholds, we created a sequence of values ranging from 0 to 1 by 0.01. We fit the logistic regression model on our training data and calculated the profitability and precision across the sequence of threshold values. We stored the values of precision and profitability for each threshold in a matrix, which we used to find the threshold that yielded maximum profit.



In the adjacent plot of profitability by probability threshold, it can be seen that there is a real advantage to intelligently picking a threshold, rather than defaulting to the standard 0.5 usually used for classification. Further increases of the threshold after 0.39 result in lower profitability, so we would be profiting less if we chose 0.5. The next plot of the distribution of predicted probabilities for

underdogs winning in the test data gives the main reason why.

Because we are betting on games we predict as a win, our threshold determines how many games we can bet on. In this case, this corresponded to 32.99% of games. With a lower threshold, we predict more games as a win, causing us to bet on more games. The opposite occurs when we raise the threshold, where we bet on fewer games. The threshold tells us the minimum probability of winning required in order to consider a game bettable. So if we have a threshold of 0.5, we can only bet on games where the probability of the underdog winning is at least 0.5. If we lower our threshold to 0.39, we can still bet on the games we considered bettable before, but now we can also bet on games whose probability of the underdog winning is 0.39 to 0.49, which adds to the number of games we are betting on. If we instead raise our threshold from 0.39 to 0.5, now we can no longer bet on games with probabilities of the underdog winning between 0.39 and 0.49, so we are constricting the number of games we can bet on.

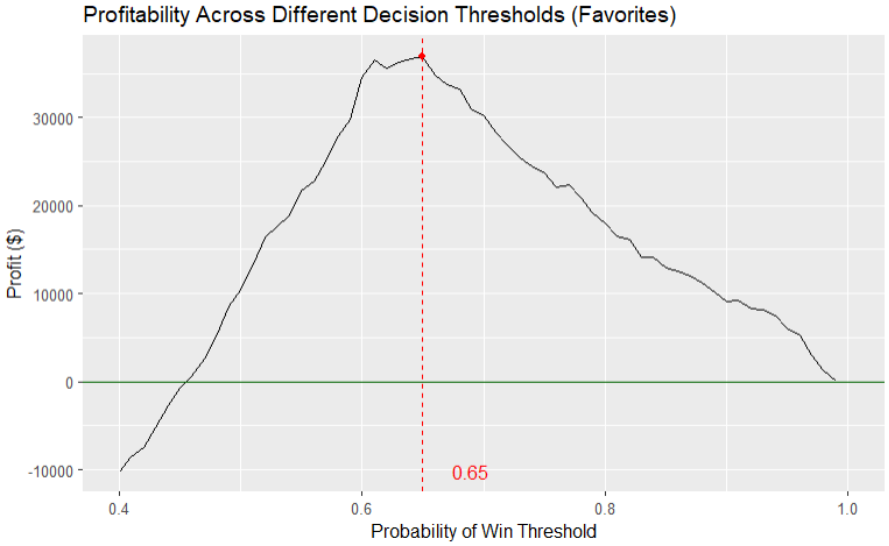


Most of the time we will predict the probability of an underdog winning is less than 50%, which can be seen in the above plot. This makes sense since an underdog is by definition the team we expect to lose. So while we obtain a higher training precision using a threshold of 0.5 (59.13%) vs 0.39 (49.91%), we make a smaller total profit since we are not betting on as many games. The reason we can still profit even if we expect we might be wrong more than half the time is because there is more upside on underdog bets. If we take E[Profit] = $0.39\times > \$200 - 0.61 \times \$100$, we can still be positive overall because the first term in our expected value is greater than the second.

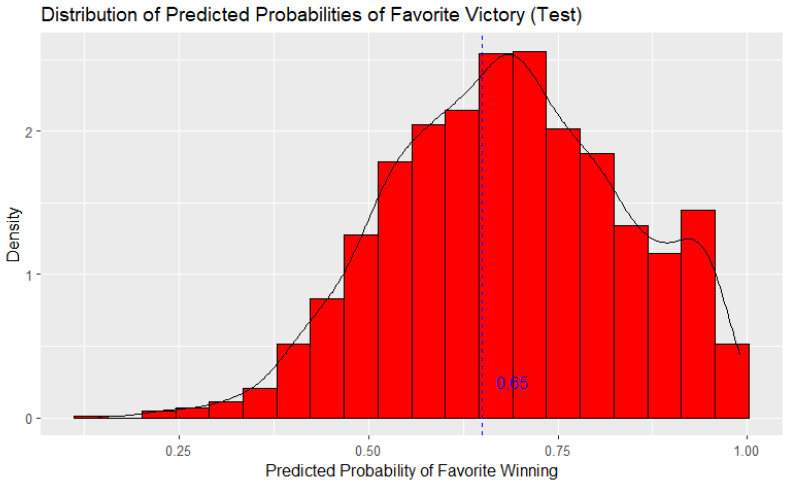
Similarly, when it came to picking a probability threshold for the favorite team model, we chose the threshold that gave us maximum profitability on our training data.

This resulted in a threshold of 0.65. This was consistent with our hypothesis that we would need to be more confident in our favorite picks since there is less upside on the bets. This threshold gave us a training precision of 79.49%.

We can see in the next plot of predicted favorite probabilities



that if we chose a threshold of 0.5, then we would be betting on the favorite for most of the games. This is not as profitable as having a higher threshold such as 0.65, because we have more potential downside than upside to our bets placed on favorite teams. When choosing a threshold of 0.65, we are considering 58.84% of games bettable, as that is the proportion of predicted probabilities greater than 0.65.



4. **Neural Network**

For our neural network, we tried a few different structures, which included a model with no hidden layers as our control, and one and two layer networks. Within these, we tried different dropout probabilities in the dropout layers. To our surprise, the best neural network was the control without any hidden node layers which is essentially a neural network version of logistic regression. Despite having the dropout layer, the more complicated model overfit on the data very fast and did not perform better than the simple neural net model, which is what we used in the end.

The threshold of success for underdogs and favorites was determined in the same method as above, by maximizing the training profitability. The thresholds for underdogs and favorites were 0.39 and 0.65, respectively. These happened to be the same values as what we obtained for our logistic regression models, so we were betting on roughly the same number of games.

5. **XGBoost**

We again found our probability of winning thresholds for our underdogs and favorites by maximizing the training profitability. This time around, we obtained lower values for our thresholds, with respective values of 0.36 and 0.62 for underdogs and favorites. Since we have lower thresholds for what constitutes a bettable game, we ended up betting on a greater percentage of games, with respective percentages of 39.98% and 65.97% for our underdogs and favorites. The effect of betting on teams with lower probabilities of winning will be seen in the results.

Results

1. **Multivariate Regression**

MANOVA results

	Pillai	Approximate F	p-value
Away Open Total Line	0.264	1913.909	<0.001
Away Money Line	0.064	366.395	<0.001
Away Line	0.019	101.189	<0.001
Home Money Line	0.013	71.011	<0.001
Away WIS	0.002	12.627	<0.001
Away All Games FG%	0.002	11.886	<0.001
Away Last Five Games FG%	0.002	8.861	<0.001
Home All Games FG%	0.001	7.745	<0.001

We ran a multivariate analysis of variance, aka MANOVA to get a sense for what variables were contributing the most to our model. This is given in the adjacent table. The nice thing about MANOVA is that it tells us about which variables are jointly contributing to both

underdog and favorite scores. The way we interpret variable importance is by looking at the Pillai statistic, which is a number between zero and one telling us how much a variable is contributing to our model. The p-values and F statistics might be a little imprecise, since we had a bit of lingering collinearity in our data.

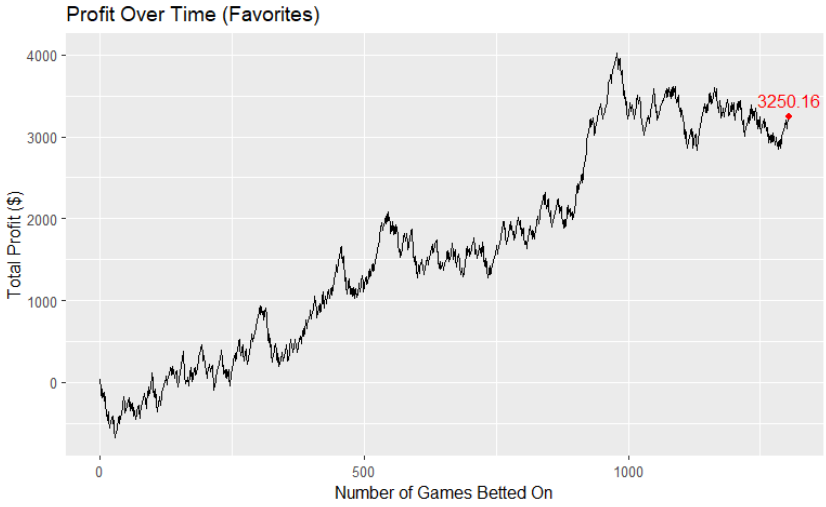
The most important variable in our score prediction model was the open total line. The given line is for away but this was the same for home as well. It makes sense that this helps us predict scores, since this line gives us the proposed total score on a game that people bet on. Home and away money lines were also important, which tells us how much of our initial bet we make if we win if we bet on the home or away team, respectively. The away line tells us the proposed score difference that we can bet on. The rest of the important variables tell us field goal percentages, which is what percent of non-free-throw baskets are made, as well as wins in season which tells us how many wins the team has. It makes sense that these variables are important since they give us an idea of how good a team is.

The adjacent table tells us which predictors were most significant for predicting the favorite team's score. The money lines and open total line made another appearance. An increase in the home and away money lines of one gives a prediction of 1.63 and 1.20 more points for the favorite team, respectively. We also have some statistics for if the home team won the previous games against the spread. If the home team won the last game against the spread, we predict the favorite team to score an additional 4.95 points. These values are difficult to practically interpret, since while the home team is more often the favorite, this is not always the case. Similarly, the predictors for underdog score were not super interpretable either since they tell us about the home and away teams. If we had more time or the foresight from the beginning, this model would have been more interpretable if we had instead coded up each statistic for the underdog or favorite team.

	Score (Favorite)		
Predictors	*Estimates*	*CI*	*p*
Away Money Line	1.20	0.87 – 1.54	**<0.001**
Open Total Line	0.42	0.35 – 0.49	**<0.001**
Home Money Line	1.63	1.09 – 2.17	**<0.001**
Home Previous Game 1 ATS Win	4.95	2.20 – 7.70	**<0.001**
Home Previous Game 2 ATS Win	5.35	2.61 – 8.08	**<0.001**
Home Previous Game 3 ATS Win	5.32	2.61 – 8.04	**<0.001**
Home Previous Game 5 ATS Win	4.65	1.97 – 7.33	**0.001**
Observations	11078		
R^2 / R^2 adjusted	0.263 / 0.233		

2. Logistic Regression

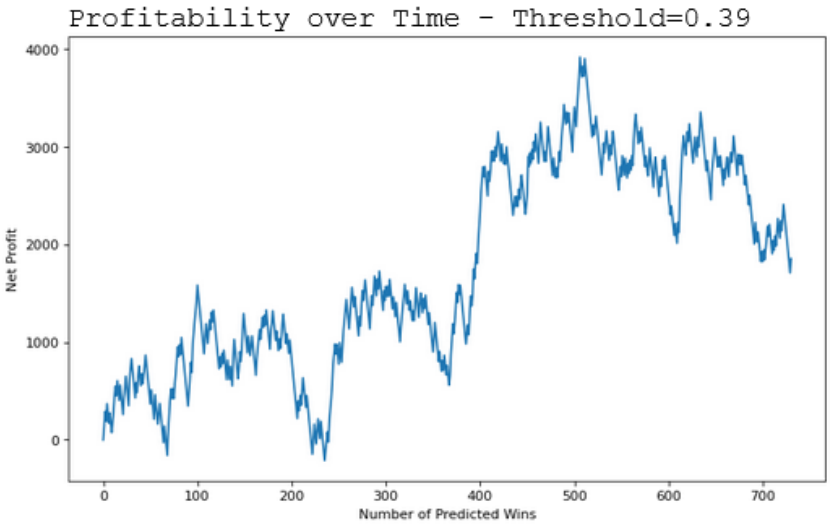
The logistic regression models performed fairly well for picking underdog and favorite bets. For the picking underdogs model, we got a test precision of 42.27%. For the picking favorites model, we got a test precision of 76.23%. We were interested in precision and not overall accuracy because precision tells us how often our bets hit. This is in contrast to overall accuracy which also tells us how well we choose when not to bet. We are not concerned with any opportunity cost associated with not betting



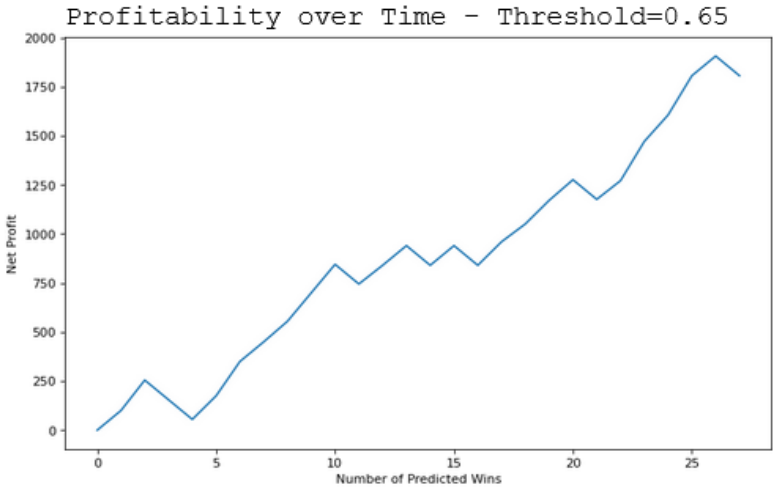
when we should have. We are concerned with getting the games we end up betting on correct. Both picking favorites and underdogs was profitable when using the logistic regression model. For the underdogs, we made a total profit of $1634.66 when betting $100 on each of the 731 games we classified as bettable. This averaged out to $2.24 per game, which is a 2.24% average return per game. For the favorites, we made a total profit of $3250.10, when betting $100 on each of the 1304 games we classified as bettable. This gave us a profit of $2.49 per game, which is a 2.49% average return per game.

The above plot shows us how much money we have made after betting on the favorite team for some number of games. We can see that we are generally making money over time. However, we lose money before we start profiting. The underdogs told a similar story, however, the amount of money made over time was a bit more variable. This is to be expected since we are a little less conservative in what we are betting on.

3. **Neural Network**



Our neural network gave similar results to that of our logistic regression model. For underdogs, we obtained a test precision of 42.2%. For favorites, we obtained a test precision of 76.3%. Our neural network picked underdog and favorite bets profitably, resulting in total profits of $1953.94 and $3078.72, respectively. In addition to the thresholds chosen by maximizing training profitability, we also tried using higher threshold values to obtain greater values of precision. This resulted in a notable change in our plots of profit over time. Switching the threshold to 0.65 for our underdog picks results in our profit becoming much more stable, since we're not betting on as many games and we have a greater value of precision for the games we end up betting on. The test precision for underdogs increased from 0.42 to 0.74, which is a substantial increase. Similarly for favorites, our test precision goes from 0.763 to 0.914 when we change our threshold from 0.65 to 0.9. We were less profitable with the higher thresholds, with respective total profits of $1806.29 and $2328.72 for the underdogs and favorites. Even though our total profitability was lower for these high threshold models, our average return per game improved greatly. For underdogs, our average profit per game increased from $2.68 to $66.90. For

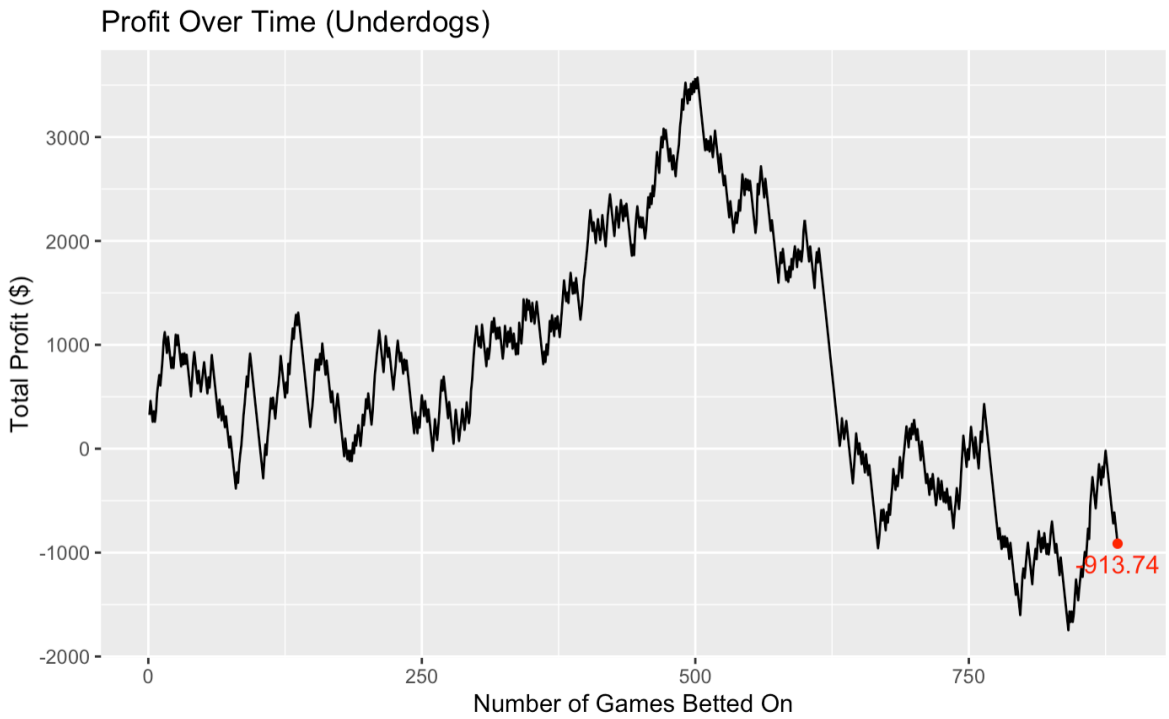


favorites, our average profit per game increased from $2.37 to $10.49.

4. **XGBoost**

Our XGBoost model performed fairly poorly compared to our logistic regression and neural network models. For the model that picked underdogs, we got a test precision of 41.99%. For the model that picked favorites, we got a test precision of 74.97%. While picking favorites was profitable, it was less profitable than our other models with a total profit of $404.54. On the other hand, picking underdogs was not profitable, with a total profit of -$913.74. We hypothesized that this was due to overfitting on the training data set, which resulted from choosing too large of a value for the maximum depth tuning parameter. The resultant overfit model had lower probability thresholds for picking underdog and favorites bets, with values of 0.36 and 0.62, respectively. This meant that we were betting on more games with a lower probability of success, which led to the poor test performance. In hindsight, tuning some of the other available hyperparameters could have improved the model and helped avoid overfitting. One hyperparameter in particular that could have helped is eta. This represents the step size shrinkage that can be used to prevent overfitting. The model gets the weights of the new features after each boosting step, and the eta shrinks the weights to make the model more conservative. By implementing this hyperparameter into the model, it could have ended up more profitable.

In the plot below of profit over time, we see that we are extremely profitable around game 500. However, our profitability falls off a cliff around game 600, where it proceeds to go negative. This illustrates the increased risk we take on when we pick a lower threshold for picking bets.



Method	Total Underdog Profit	Total Favorite Profit	Games Betted On (Underdog)	Games Betted On (Favorite)
Logistic Regression	$1634.66	$3250.16	731	1304
Neural Network	$1953.94	$3078.72	730	1298
XGBoost	-$913.74	$404.54	886	1462

Conclusion

Our primary goal of this project was to create models that picked profitable bets. Our logistic regression models were profitable for both picking favorites and underdogs. The neural nets we created were also profitable for both favorites and underdogs. While XGBoost wasn't profitable for picking underdogs, it did end up with a profit when it came to picking favorites. It is therefore safe to say that we accomplished our goal of picking profitable bets. We also got some good information about what was going into our models because of our multivariate regression analysis. It is important to note that a lot of what was going into our models seemed to be based on the betting information made available by Foxsheets. In combination with the team statistics, this resulted in us being able to outsmart the sportsbook over the long run.

While our models for picking bets resulted in profitability, there exist some limitations. For one, the profits you would make in the real world would be strictly lower than what we have reported. This is because you do not get to keep all of your sports betting earnings. Both the sportsbook and Uncle Sam will take their cut of your winnings. This is through fees and income taxes. Another huge limitation is the amount of money you would need to go through this process. Our end profits were accumulated after betting $100 on between 700-1500 games. This would amount to putting $70,000-$150,000 total into sports betting, which is a lot of money. While you are not spending that much money outright, since if you keep winning you would keep getting your $100 back, you are still exposing yourself to a lot of financial risk in order to make $1600-$3250 in return (or even lose money). We had multiple models go into the negative for profits at certain points, which would require you to put additional money back into betting. A solution to this risk problem could be to pick a model that has a high threshold in order to get a higher level of precision and thus only bet on games you are very confident in. An example of such a model is our high threshold neural network. However, with the higher threshold comes a much lower number of games you can bet on (1% of games in this case) resulting in a slow profit over the long run.

In addition, there are some possible improvements that could help us overcome these limitations, and hopefully result in more profitability. The first is using better data. Our multivariate regression model would be more interpretable if we had coded up each statistic as belonging to the underdog or favorite instead of the home or away team. Also, the models' performance is limited by the degree to which a dataset can provide useful information. For this project, we would want more information that can better separate what constitutes a win or a loss. We could accomplish this by including more data, such as individual player statistics, or

through more data transformations of our dataset. The second possible improvement we could have made would be further tuning of the hyperparameters for our XGBoost model, as our model overfit the training data and was not very profitable on the test data. The third possible improvement would be using different and more complex models that better fit the data. Our results showed that our machine learning models did not perform better than a logistic regression, but machine learning should not be discounted entirely since better models might exist. One such example is bayesian neural networks, where prior distributions are leveraged to help prevent overfitting data.

References

Andri et mult. al., S. (2021). *DescTools: Tools for Descriptive Statistics*. https://cran.r-project.org/package=DescTools

Gibbs, Connor. (2018). Lecture on NBA Betting Data. Personal Collection of Connor Gibbs, Colorado State University, Fort Collins, CO.

Ford, Clay. "Getting started with Multivariate Multiple Regression." *University of Virginia Library,* url:(https://data.library.virginia.edu/getting-started-with-multivariate-multiple-regression/). Retrieved May 5, 2022.

Fox, J., & Weisberg, S. (2019). *An R Companion to Applied Regression* (Third). Sage. https://socialsciences.mcmaster.ca/jfox/Books/Companion/

Foxsheets, Statfox, http://foxsheets.statfoxsports.com/foxsheets.aspx?sportCode=nba. Feb. 2022.

"How Much Money Do Americans Bet On Sports?" (2022, April 7). *Legal Sports Betting.* url:(https://www.legalsportsbetting.com/how-much-money-do-americans-bet-on-sports/)

Kenton, Will. "Durbin Watson Statistic Definition." *Investopedia,* url:(https://www.investopedia.com/terms/d/durbin-watson-statistic.asp). Retrieved May 5, 2022.

Krishnaswamy, N. (n.d.). *NNet class in Pytorch* [Resource from CS 445, adapted to binary classification for use in this project.]. Colorado State University.

Leung, Kenneth. (2021, October 4). "Assumptions of Logistic Regression, Clearly Explained." *Towards Data Science,* url:(https://towardsdatascience.com/assumptions-of-logistic-regression-clearly-explained-44d85a22b290)

Lüdecke, D. (2021). *sjPlot: Data Visualization for Statistics in Social Science*. https://CRAN.R-project.org/package=sjPlot

Ortiz, A., & Smith, N. (2022, January 8). *Sports betting generating billions*. NewsNation Now. Retrieved February 28, 2022, from https://www.newsnationnow.com/morninginamerica/sports-betting-generating-billions-in-tax-revenue-across-us/#:~:text=(NewsNation%20Now)%20%E2%80%94%20Sports%20betting,over%20%243%20billion%20in%20revenue.

Prasad, A. (2020, September 13). *PyTorch For Deep Learning — Binary Classification (Logistic Regression)*. Medium. Retrieved May 6, 2022, from https://medium.com/analytics-vidhya/pytorch-for-deep-learning-binary-classification-logistic-regression-382abd97fb43

Rencher, Alvin. *Methods of Multivariate Analysis.* EBSCO Publishing, 2002.

van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, *45*(3), 1–67. https://doi.org/10.18637/jss.v045.i03

Verma, A. (2020, February 29). *PyTorch [Tabular] — Binary Classification | by Akshaj Verma*. Towards Data Science. Retrieved May 6, 2022, from https://towardsdatascience.com/pytorch-tabular-binary-classification-a0368da5bb89

What is XGBoost? NVIDIA Data Science Glossary. (n.d.). Retrieved May 6, 2022, from https://www.nvidia.com/en-us/glossary/data-science/xgboost/

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., … Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Yakowicz, W. (2022, January 10). *Where is sports betting legal? A guide to all 50 states*. Forbes. Retrieved February 28, 2022, from https://www.forbes.com/sites/willyakowicz/2022/01/07/where-is-sports-betting-legal-america-2022/?sh=42e176332342

Zach. (2020, December 4). *XGBoost in R: A step-by-step example*. Statology. Retrieved May 6, 2022, from https://www.statology.org/xgboost-in-r/

Zhu, H. (2021). *kableExtra: Construct Complex Table with "kable" and Pipe Syntax*. https://CRAN.R-project.org/package=kableExtra