Quantifying NBA Shot Quality

Carly Hammond

March 19, 2019

Abstract

This paper examines the application of various machine learning algorithms to the problem of quantifying the quality of shots taken by basketball players in the NBA. Logistic regression, decision tree, random forest, and nearest neighbor models will be fit to an individual player's shot data and evaluated for accuracy. The goal of this work is to improve upon existing measures of shot quality by individualizing the models for a specific player. Only one player is examined in this work, though it will be seen that these methods can extend to all players.

1. Introduction

My motivation for this work is twofold. First, this work is founded in self-improvement. Prior to completion of this project, I was completely unfamiliar with machine learning algorithms, but found myself applying for jobs that expected this knowledge from an employee. Thus, I have used this opportunity to teach myself these algorithms and become better prepared to enter the job market. The second reason for this work is I am extremely interested in sports analytics, specifically basketball, and believe that this work has the potential to inform the decisions of NBA coaches and players.

The recent developments of player tracking technologies in professional sports have opened up vast opportunities for statisticians and data scientists to change the way players, coaches, and fans think about these games. Teams are collecting more data than ever before in an attempt to obtain new insights about the game, or to quantify what they already know intuitively. In basketball, most people that are familiar with the game have an idea of whether or not a shot has a high chance of going in as soon as they see it. Some of the things we naturally think about when making this decision are the distance of the shot, time left on the shot clock to potentially get a better shot, and how contested the shot is by a defender. This work is an attempt to see how these variables actually do impact the probability a shot goes in the basket, or shot make probability, for a given player. This is also what we will consider "shot quality". This information could be used by the player himself to improve upon weaknesses or make better in-game decisions, but it could also be used by opposing teams to force the player to take "bad shots" while defending them.

2. Related Work

At the MIT Sloan Sports Analytics conference in 2014, Chang et al. presented a new metric, now called quantified shot quality (qSQ), that used different machine learning techniques to estimate the make probability of an NBA shot. Their work was admittedly open-ended in that they anticipated models would improve as the data available evolves, but their error and predictability estimates will serve as a measuring stick to compare my own models. I hypothesized that individualizing the models for one player, rather than fitting models for all players combined, would reduced the error of the models and increase predictability because different players will inevitably tend to handle in-game situations differently.

3. Data and Variables

The larger data set consists of 500,000 observations of shots taken in the NBA during the 2014-2016 seasons. As mentioned previously, I chose to individualize modeling to a specific player, so the data used in this project is a subset of this larger data set. I chose my favorite player, LeBron James, to analyze. This left me with 2,708 shot observations with 15 variables including shot distance, the shooter average speed, whether or not the shooter dribbled before the shot, time left in the game, time left on the shot clock, time between receiving a pass and shooting, distance from nearest defender, height of the nearest defender, and whether or not the shot was made.

4. Methods

Since the variable of interest, whether or not a shot was made, is binary, this problem is one of classification. I wanted to have some idea of what made a "good" model that I could use to compare my results to. In the work by Chang et al., the measure of error used was the Brier score. Brier score measures the mean squared difference between the predicted probability, p_i, assigned to the possible outcomes for item i and the actual outcome y_i. Thus, lower Brier scores for a set of predictions indicate better calibration. Effectively, for this problem, the Brier score is the mean squared error (MSE) of the prediction.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(p_i - y_i)^2$$

An 80/20% training/test split was used to fit and test all of the models.

4.1 Logistic Regression

Logistic Regression seemed like a logical first step since I was dealing with a binary outcome. I used the `regsubsets()` function in the leaps R package to select the variables in the model. AIC criterion was used to select from models considering all possible variables with no interaction terms except for the interaction between nearest defender distance (ndd) and defender height. This interaction makes sense to include intuitively, as the height of the defender and how close they are to the shooter will effect the shot angle that is necessary to

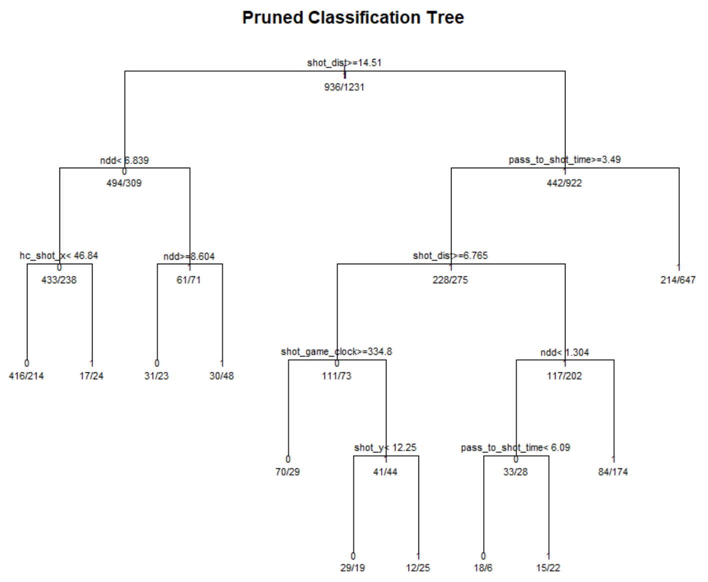


Figure 1: Simple decision tree

avoid getting blocked. The final model chosen was of the form:

$$
\begin{aligned}
log(\frac{p}{1-p}) = \beta_0 + \beta_1\text{dribbledTRUE} + \beta_2\text{shooter\_ave\_speed} + \beta_3\text{ndd} + \beta_4\text{defender\_height} \\
+\beta_5\text{pass\_to\_shot\_time} + \beta_6\text{shot\_dist} + \beta_7\text{dribbles} + \beta_8\text{ndd} \times \text{defender\_height}
\end{aligned}
$$

where p is the probability of making the shot, and ndd is the distance from the nearest defender at the time of the shot.

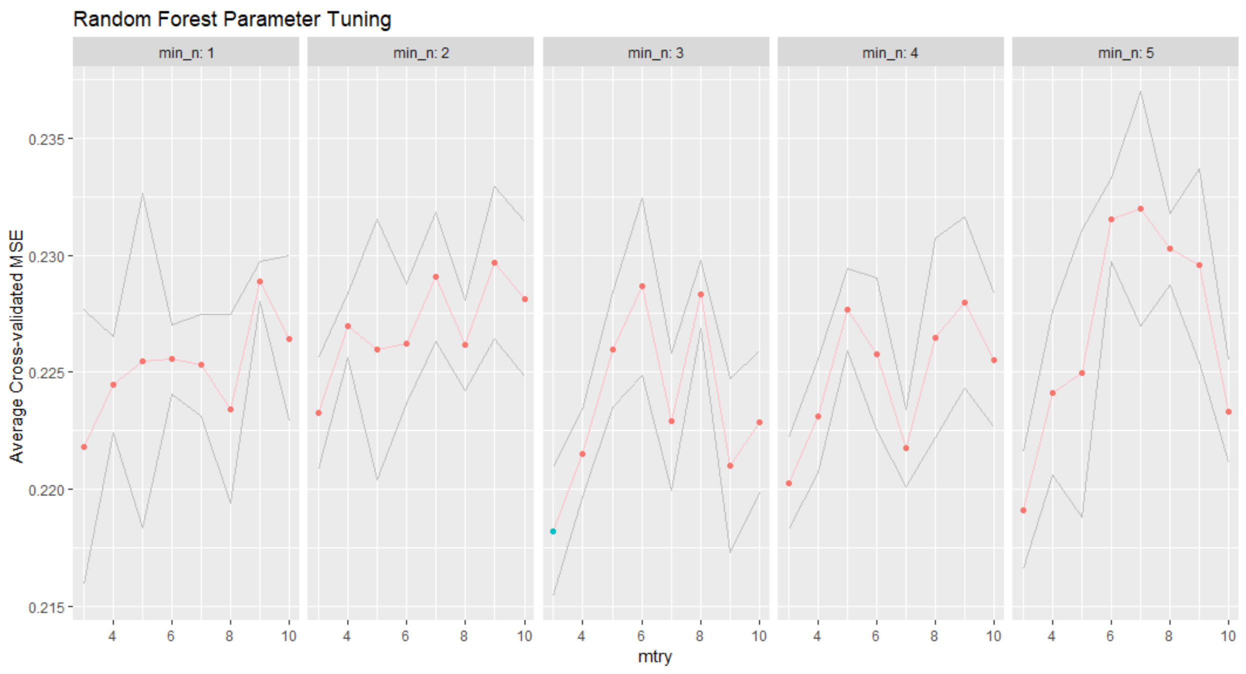
4.2 Decision Tree

Because of my unfamiliarity with machine learning algorithms I first wanted to fit a simple decision tree model so that when I fit more complex tree based models I would understand what was going on behind the code. This technique splits the sample into homogeneous sets based on the most significant splitter input variable. I used the rpart package in R to fit a simple, default model. The pruned decision tree can be seen in Figure 1. above to get an idea of variable importance.

4.3 Random Forest

Random Forest models use a “bagging” algorithm that repeatedly creates optimal decision trees on random subsets of variables. These decision trees together make up the “forest”. When the model sees an observation in the test set, each decision tree votes on the classification. The output of the model is usually the mode of these votes, but because I am concerned with the prediction probabilities rather than the classifications themselves, my output is the proportion of trees voting “make”. Because multiple decision trees are used in the ultimate classification process, these models are robust to overfitting.

The parsnip package from R’s “tidymodels” was used with five-fold cross-validation to tune the model parameters `mtry`, the number of variables randomly chosen at each split, and `min_n`, the minimum number of data points in a node that are required for the node to be split further. The forest in my model contained 1,000 trees. We can visualize the parameter tuning process in the figure below. The lowest average MSE across the five folds occurs at `min_n` = 3 and `mtry` = 3, seen in blue.



4.4 K-Nearest Neighbor

The K-Nearest Neighbor (KNN) algorithm is based on variable/feature similarity. It looks at how closely the variables of an observation in the test set resemble the training set. An observation is classified by a majority vote of its neighbors. It is assigned to the class most common among its k nearest neighbors. How close is close? R's parsnip package allows us to change the parameters weight_func, the type of kernel function that weights the distances between samples, and dist_power, the parameter, q, used when calculating the Minkowski distance

$$d(x_i, x_j) = (\sum_{s=1}^{p} |x_{is} - x_{js}|^q)^{\frac{1}{q}}$$

This corresponds to the Manhattan distance with $q = 1$ and the Euclidean distance with $q = 2$. In other words, we get to define how our algorithm sees how closely the variables of observations resemble each other.

Again, I used five-fold cross-validation to tune the model parameters. After tuning, the optimal number of nearest neighbors was 24. A rectangular kernel was found to be optimal, which is interesting in that it gives equal weights to all neighbors, and the Minkowski distance parameter of the final model was 5.

5. Results

As mentioned previously, the main metric that was used to evaluate the performance of these models was the Brier score, or mean square error of the prediction. This metric makes sense from a practical standpoint because we are not so much concerned with accuracy of classification as we are the classification probabilities themselves. It is the probability that ultimately gives us the shot quality. So, for example, if the actual data is:

$$\{make, miss, make, miss, make\}$$

and model A predicts with probabilities:

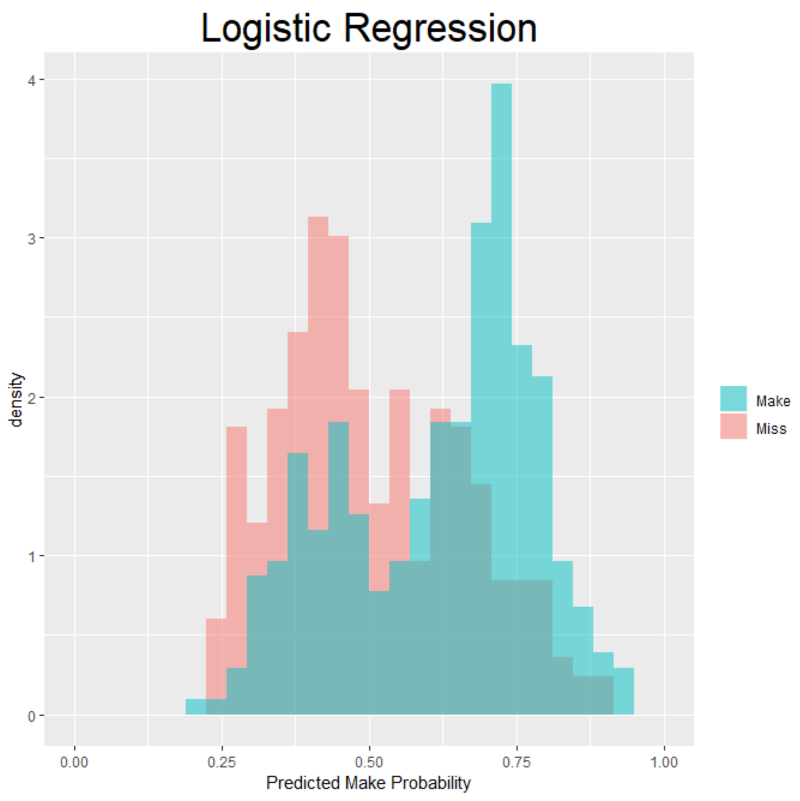
$$\{0.6, 0.4, 0.6, 0.4, 0.1\}$$

and model B with:

$$\{0.9, 0.1, 0.9, 0.6, 0.4\}$$

then model A has an MSE of 0.29 compared to model B's 0.15. Model A will correctly classify more of the data (misclassifications in red), but we would prefer model B because it was correct when the probabilities were at either extreme, and only incorrect at a probability close to 50%. This makes the probabilities themselves more informative to a coach or player.

Logistic regression had the lowest MSE of all the models, with 0.219. Figure 3 below shows a histogram of the test set and their predicted probabilities, with made shots appearing in blue and misses in pink. We can see see that higher probability shots are made more frequently than they are missed, and the reverse is true with low probability shots as we would expect. The bimodality of the shot predictions is an interesting feature of this plot that I have not yet been able to explain.

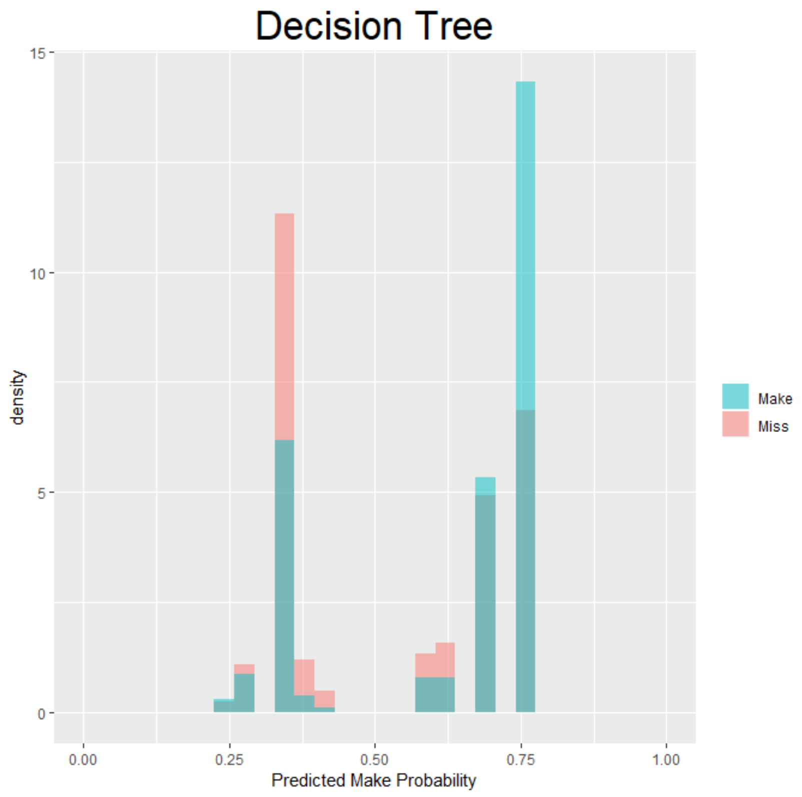


We can also look at these results in a confusion matrix, where the test set is tabulated so that we can see the classifications of all shots in the test set, and whether or not they were actually made. We see that the model accurately predicts made shots fairly well with a specificity of 71.7%, but does misclassify missed shots more often than we would like to see with sensitivity of 55.6%.

Logistic Regression Confusion Matrix

		Reference	
		0	1
Prediction	0	134	85
	1	107	215

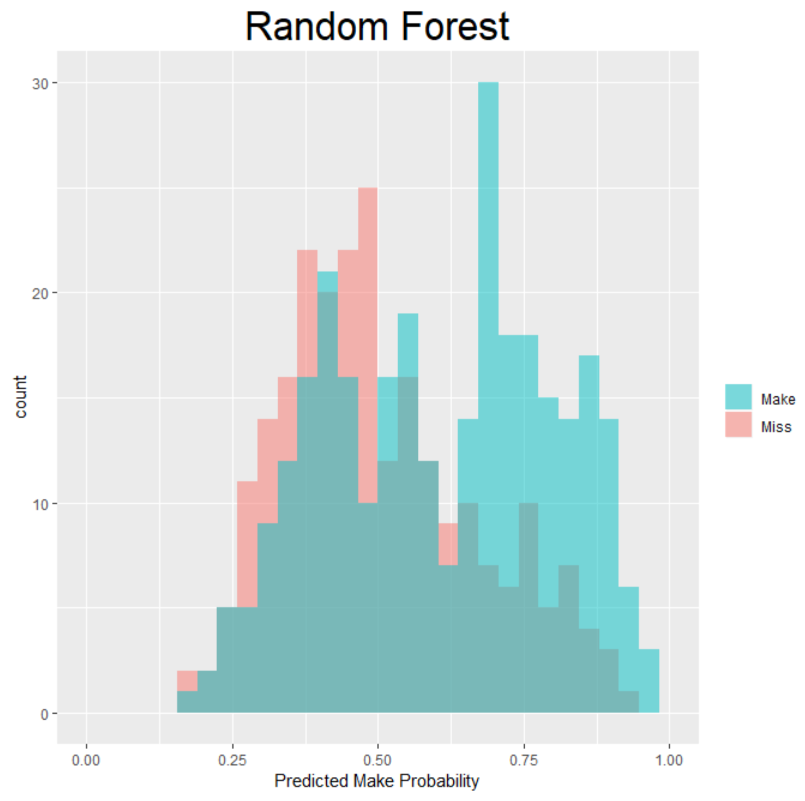
The simple decision tree model had an MSE of 0.235, which was the worst of my models and the only one that did not improve upon the MSE's of the models presented by Chang et al. in 2014. The same style of histogram can be seen in Figure 3 below, where we see that only a few probabilities are actually being predicted. This feature alone makes this particular model a poor choice for quantifying shot qualities, since it would only output these probabilities.



Decision Tree Confusion Matrix:

		Reference	
		0	1
Prediction	0	119	81
	1	122	219

The random forest model produced an MSE of 0.229, which improves upon previous work, but falls just short of the logistic regression model. The distribution of probabilities shows the number of falsely predicted made shots steadily decreases as the make probability increases, but the made shots are still bimodal, with a number of made shots having a predicted make probability below 0.5.

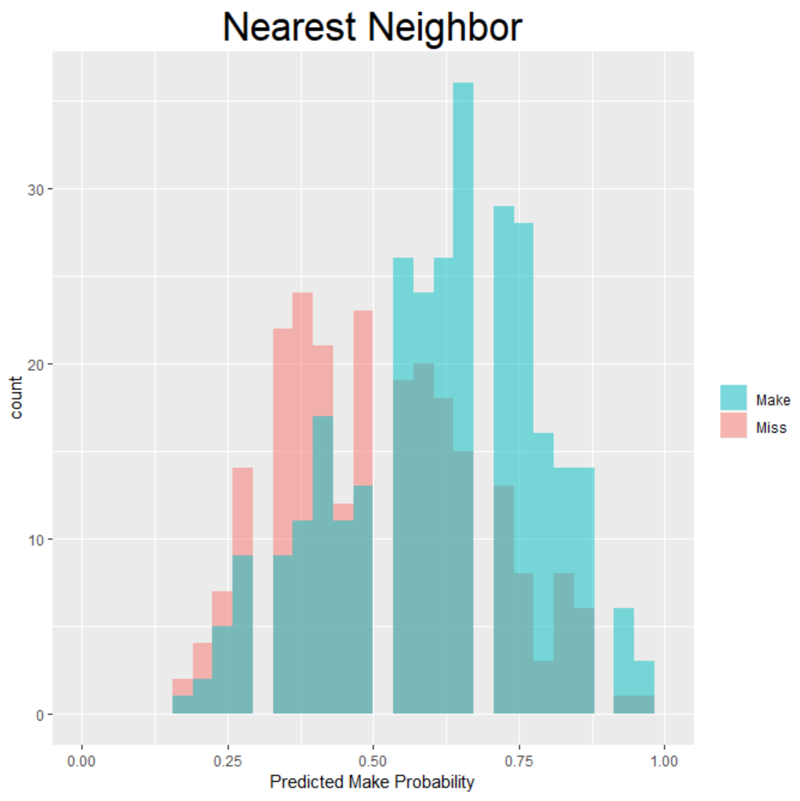


Like the logistic regression model, this model is very good at predicting made shots, correctly classifying at 67.7%, but it only correctly classifies 57.68% of missed shots. It may be important to note that this is the model with the highest sensitivity.

Random Forest Confusion Matrix:

		Reference	
		0	1
Prediction	0	139	97
	1	102	203

Lastly, the nearest neighbor model had an MSE of 0.228 when run on the test set. This is better than the random forest model, but still not reaching the performance of logistic regression. Also of concern with this model is the discretized probabilities that it outputs. There are some probabilities that appear to be impossible to achieve with this model, which could make it slightly less informative when judging shot quality.



This model has very high specificity, correctly predicting 74% of made shots, though it too does not perform well when predicting missed shots with a sensitivity at 53.53%.

Nearest Neighbor Confusion Matrix:

		Reference	
		0	1
Prediction	0	129	78
	1	112	222

6. Conclusion

After applying the four machine learning algorithms, it appears that for this specific player, LeBron James, logistic regression did the best job of minimizing the difference between shot quality and the true outcome. The lowest MSE achieved by Chang et al. when they looked at all players together was 0.2322, compared with our 0.2189. To put this into perspective, we would achieve an MSE of 0.25 if we simply predicted all shots with probability 0.5. So, our improvements are significant. As I mentioned before, the idea of this work is that this model selection process could be repeated for any player that has enough shot data to find improvements to a more general model.

Model	MSE
50% for all shots	0.2500
Best 2014 model (unspecified)	0.2322
Simple Decision Tree	0.2347
Random Forest	0.2299
Nearest Neighbor	0.2277
Logistic Regression	0.2189

Much like Chang et al. did in 2014, I too anticipate that with development of newer, more sophisticated player tracking technologies there is still room to improve this metric, but it appears that individualizing the models was indeed a step in the right direction.

References

[1] Y. Chang, R. Maheswaran, J. Su, S. Kwok, T. Levy, A. Wexler, and K. Squire. Quantifying shot quality in the nba. MIT Sloan Sports Analytics Conference, 2014.