Predicting National Basketball Association Player's Positions Using Random Forest

Shadi Chamseddine – 100937807
Christopher Lee – 100937241

STAT 5703 W

Carleton University

Shirley Mills

Due Date: March 19, 2020

Statement of Problem

Given the increasing amount of data being collected in the world, companies are employing new ways to use this data to their benefit. One area in which data analytics is making big inroads is in professional sports. Teams are increasingly collecting and making use of sports statistics to help them analyze various tactics that can help them to gain an edge over their opponents. A primary example of this can be found in the National Basketball Association (NBA), where teams are increasingly collecting and analyzing advanced game level statistics which have provided insights which shifted the way teams assemble their rosters and play the game. For instance, teams have recognized the value of 3 point shots compared to 2 point shots from the data and have shifted to shooting more 3 point shots, putting emphasis on building rosters where all five players on the court have the ability to shoot the three point shot. Another drastic shift in the NBA has been towards what is called "positionless basketball", where teams have moved away from playing the traditional lineups based on player size and have prioritized versatile players who have the ability to contribute in all statistical categories and defend multiple positions. With the importance of data analytics to the sport continually growing, we are interested in taking a deeper look into how the progress of data analytics in sports has transformed the game of basketball in the NBA. Using a Random Forest model, we aim to predict an NBA player's position based solely on their in-game statistics.

Literature Review

The idea of random decision forests was introduced to the world by Tin Kam Ho (1995). In her paper, she recognized the attractiveness of decision trees but acknowledges the limitations with respect to increased complexity. Decision trees that are grown too complex to a fixed training set will run into a problem of overfitting on the training set. This will cause it to fit the training set extremely well but not on the testing set. To address these issues, she introduces the systematic creation of multiple decision trees independently. These decision trees are created using randomly selected variables in the training set. The classifications from each individual decision tree are then combined together with their accuracies preserved. It is shown the use of multiple classifiers can compensate for the bias of a single classifier. Accuracy is also shown to increase with the addition of new decision trees, albeit at a decreasing rate.

Breiman and Cutler (2001) extended on the method introduced by Tin Kam Ho. Using the Law of Large Numbers, the authors show random forest always converges to a limiting value of the generalization error and does not overfit with additional trees. Measuring the generalization error of the individual decision trees can be reduced down to the strength of the decision tree and the correlation between decision trees. Random selection of variables is one way to reduce the generalization error by maintaining the strength while minimizing the correlation. In addition to the random selection of variables in the training set for the creation of the individual decision trees, Leo Breiman and Adele Cutler also used bagging. Bagging creates a different bootstrap training set for each decision tree. These bootstrap training sets are created by sampling with replacement from the original training set. Each created bootstrap training set leaves out 33% of the data from the original training set. The leftover data is called the Out-Of-Bag (OBB) dataset. As a result, each decision tree run on the bootstrap training set can be tested on the OOB dataset. The error rate from each of these decision trees is called the OOB error. The OOB error is an estimation of the generalization error. As 33% of the data in the original training set is unused, the need for a test set from the original dataset is unnecessary.

Technical Analysis

Given we are working on the topic of random forests analysis, we will be using the 'randomForest' library in R. This library implements Leo Breiman's random forest algorithm. The usage of the random forest algorithm is as a function in which you declare your inputs. An example of a random forest function in R is as follows:

$$model = randomForest(y \sim x1 + x2,\ data = data\_set,\ ntree = 500,\ mtry = 10,\ replace = TRUE,\ importance = TRUE) \quad (1)$$

In the random forest example above, not all the possible inputs for the function are included, only the inputs relevant to our analysis are highlighted. In (1) above, "model" is the variable in which the random forest results are stored. "y" is the variable on which we would like our data to classify into, this variable must be declared as a factor variable, which means it contains categorical data. "$x1$" and "$x2$" are the independent variables. You can declare your independent variables of interest by listing them individually or include all your independent variables by using the "." symbol. "$data$" is used to declare what your dataset is. "$ntree$" declares the number of decision trees you wish to create in the random forest model. If this value is not declared by default the algorithm will 500 decision trees in the model. "$mtry$" declares the number of variables you wish to randomly split by at each node in the individual decision trees. If this value is not declared by default the algorithm will use the square root of the number of independent variables in the model. "$replace$" declares whether you wish to create the bootstrap training sets by resampling with or without replacement. "$importance$" declares whether the importance of the independent variables is to be stored.

We apply this aforementioned model to our NBA data. In the past, each player position had its own specialties. For example, point guards (PG) were typically known for their playmaking and high number of assists, whereas Centres (C) were known for blocking more shots and having higher field goal percentages since they shot most of their baskets close to the net. To some degree, these distinctions between player positions still exist, but to a much lower extent as compared to previous years. Positionless basketball has moved away from player specialization, and towards putting higher value on dynamic players who can contribute across a wider range of statistical categories.

Methodology

We begin by collecting all NBA player-level game statistics from the 1982 season to the 2019 season inclusive from Basketball-Reference.com. The 1982 season was chosen as our starting point because it was the earliest season in which all the relevant data variables for our analysis began being tracked and collected by the NBA. The dataset is then cleaned to prepare for the Random Forest model. Summary statistics of each variable by position is shown and plotted to examine the data further. The dataset is then fed into the Random Forest model with default parameters to identify the most important variables to the model. Variable importance is determined by the mean decrease Gini for each variable. Since the number of variables is very large, we run the Random Forest model again using only the most important variables from the first run, which we identify by boosting our variable set. We reduce the dimensions by keeping only the top 50% most important variables, which we later use as the parameters in our refined

models. Following this, we prepare our training sets and test sets by assigning a 70% and 30% split of the overall data respectively. With our training and test sets prepared, the next step is to tune the parameters of the Random Forest model in order to increase the accuracy of our results. Even though the Random Forest model creates bootstrap training sets to build the individual decision trees, 33% of the data is unused. Which is the OOB dataset, but a separate test set is still required since we are using the OOB errors to tune parameters of the Random Forest model. We begin by feeding our training set into another Random Forest model in order to find an optimal number of decision trees to be created. Once this parameter has been optimized, the training set is then fed into a separate Random Forest model in order to find an optimal number of variables to split by at each node in each created decision tree. With our optimized parameters, we run a final Random Forest model on the training set and then use the results to predict on the test set.

Results

A boxplot of the distribution between player position and their total rebound percentage, is shown as an example below in Figure 1. Total rebound percentage is an estimate of the percentage of the available rebounds a player grabbed while he was on the floor. This variable was the most important variable in the default Random Forest model when measured using Mean Decrease Gini.

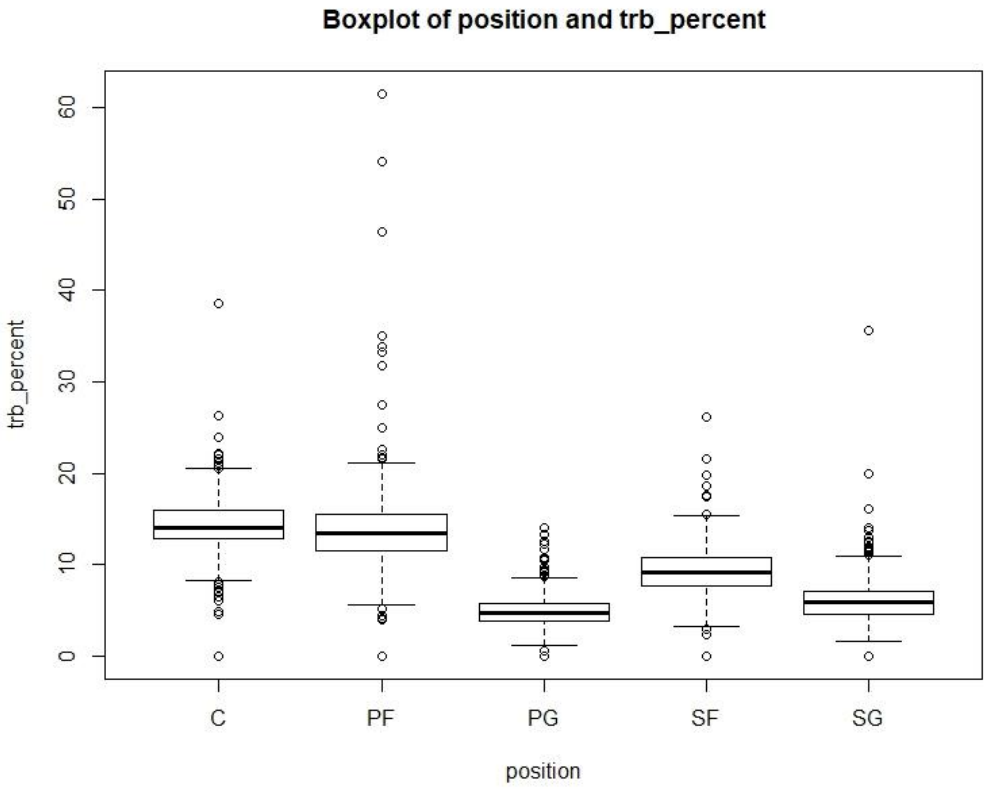

Figure 1. Boxplot of player position and total rebound percentage

From the above figure, we can see that the total rebound percentage is on average highest among C's and Power Forwards (PF), whereas it is the lowest for PG's as expected. We similarly map all our other game statistic variables by position via boxplot as above to determine the

distributions for those statistical categories. We will use these in our random forest to try and predict the position of each player in our dataset based solely on their statistical numbers.

One step in tuning the parameters of our Random Forest model is finding the most optimal number of variables to split by. Using the Random Forest model on the subset of important variables, the algorithm loops through each possible number of variable splits specified and plots the OOB errors at each given number of splits. From the model we determine the OOB errors are minimized when the Random Forest model is split by 1 variable at each node, this is illustrated below in Figure 2.

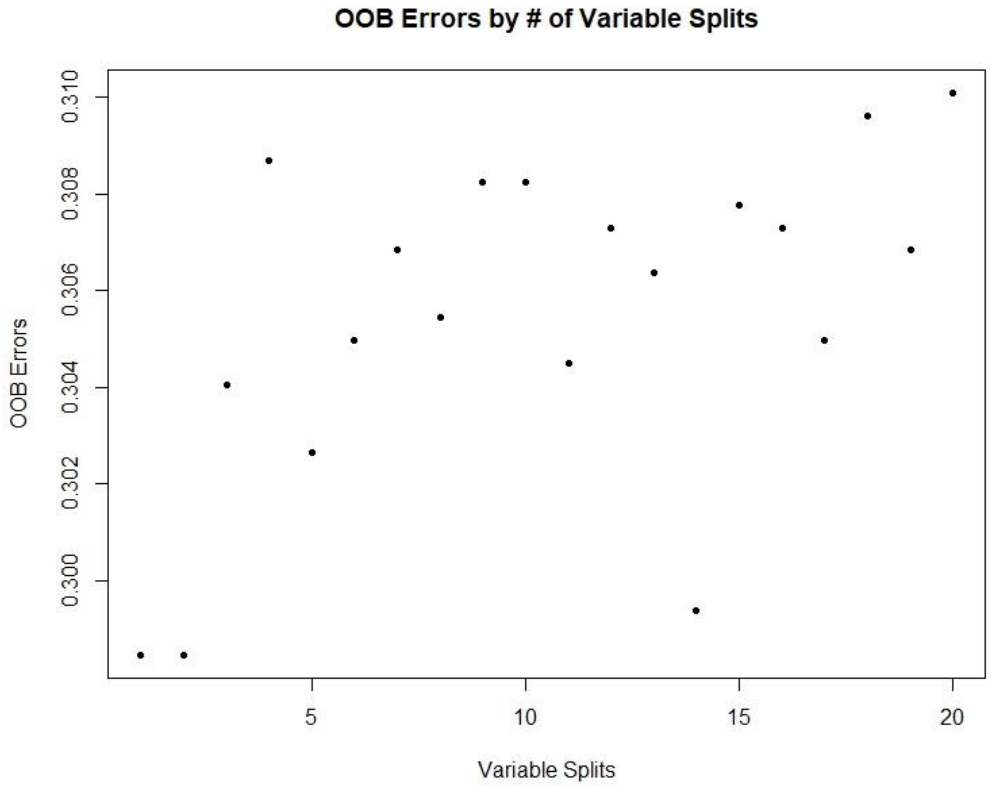

Figure 2. OOB errors by number of variable splits at each node

The second step in tuning the parameters of our Random Forest model is finding the most optimal number of decision trees to use, this is shown below in Figure 3. The green line represents the classification error rate for PF's. The purple line represents the classification error rate for Shooting Guards (SG). The turquoise line represents the classification error rate for Small Forwards (SF). The black line represents the OOB errors for the Random Forest model. The red line represents the classification error rate for C's. The blue line represents the classification error rate for PG's. The OOB errors are minimized when the Random Forest model uses 492 decision trees.

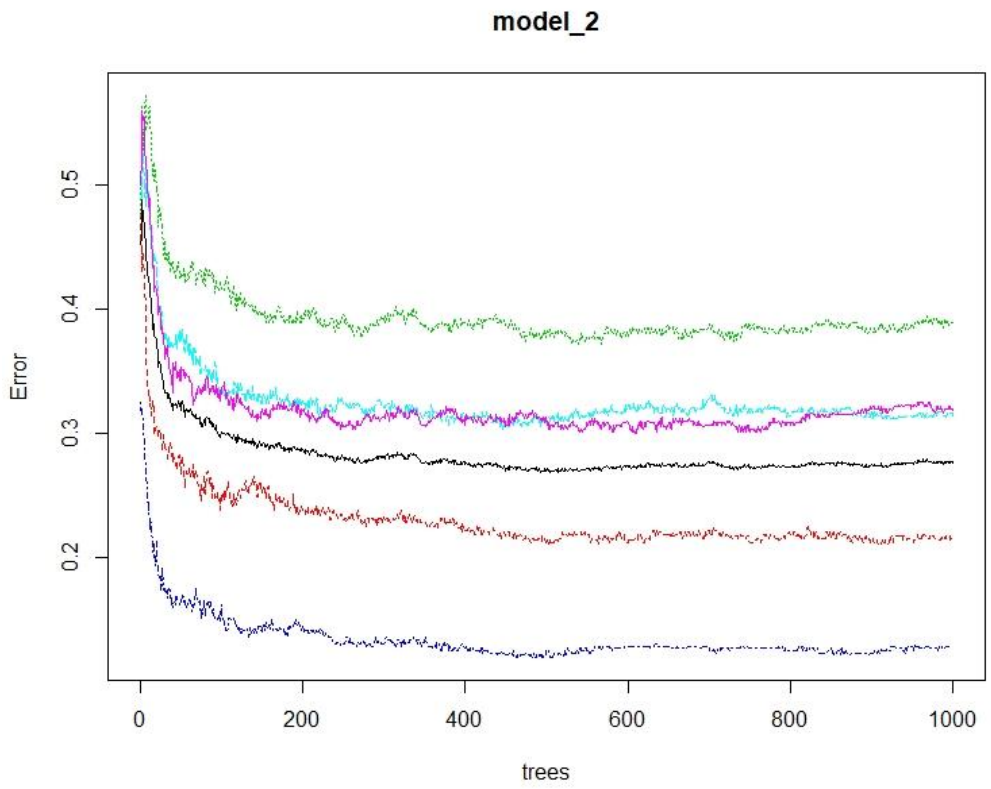

Figure 3. OOB errors by number of decision trees

Final results of the tuned Random Forest model using 492 decision trees and 1 variable at each split are shown below in Figure 4. The actual classifications from the dataset are along the row and the predicted classifications from the Random Forest model are along the columns. From the confusion matrix, we can see the model had the most difficulty classifying between the forward positions (C and PF) because they have very similar roles in the game and typically put up similar statistics. We also had some issues classifying SF's, as they sometimes fall statistically towards the PF end of the spectrum, while others play more like guards (PG or SG). Our model had the easiest time classifying PG's, as they have a pretty unique statistical combination as compared to all other positions, except for in the case of some SG's who play similarly to PG's which causes some misclassification. Overall, the OBB errors for our model sit at 30.2%

		Actual				
		C	PF	SF	SG	PG
Predicted	C	350	123	20	3	2
	PF	82	245	53	6	3
	SF	22	66	294	57	8
	SG	1	6	56	284	51
	PG	3	4	10	74	328
Classification Error		23.6%	44.8%	32.1%	33.0%	16.3%

Figure 4. Results of the tuned Random Forest model

The classification structure created from the tuned Random Forest model in Figure 4 is used to predict the position of players in the training set. The results of these predictions are shown below in Figure 5. The actual classifications from the dataset are along the columns and the predicted classifications from the Random Forest model are along the rows. The classification of player positions on the training set fits very well. From the confusion matrix, we can see that almost all players are correctly classified to their correct position, except for three data points.

		Actual				
		C	PF	SF	SG	PG
Predicted	C	457	0	0	0	0
	PF	0	442	0	0	0
	SF	1	1	433	0	0
	SG	0	0	0	424	0
	PG	0	1	0	0	392

Figure 5. Prediction results of the Random Forest model on the training set

The classification structure created from the tuned Random Forest model is Figure 4 is used to predict the position of players in the test set. The results of these predictions are shown below in Figure 6. The actual classifications from the dataset are along the columns and the predicted classifications from the Random Forest model are along the rows. We see C's are classified very well and most misclassified C's are classified as PF's, which are their closest positional comparison. PG's are classified very well, and most misclassified PG's are classified as SG's which are their closest positional comparison. The reason PG's are classified so well is because their assist statistic is much higher than for any other position.

		Actual				
		C	PF	SF	SG	PG
Predicted	C	350	123	22	1	3
	PF	82	245	66	6	4
	SF	22	66	294	56	10
	SG	1	6	57	284	74
	PG	3	4	8	51	328

Figure 6. Prediction results of the Random Forest model on the test set

Overall, our random forest model does a great job at being able to identify a player's position based solely on their in game statistics. As the accuracy rate of the model on the set aside test set is 71.7%, this means the model correctly classifies a player's position 71.7% of the time. By boosting and tuning our model, we are able to achieve greater classification accuracy.

Responsibility

Shadi Chamseddine
- Statement of problem
- Technical analysis
- Methodology
- Code to extract important variables (feature boosting)
- Slides

Christopher Lee
- Literature review
- Collecting NBA data
- Code for summary statistics and plot by position
- Random forest code and tuning parameters
- Slides

References

2018-19 NBA Player Stats: Advanced. (n.d.). Retrieved from https://www.basketball-reference.com/leagues/NBA_2019_advanced.html

Breiman, L. (2001, January). Random forests. Machine learning, 45(1), 5-32.

Breiman, L., & Cutler, A. (n.d.). Random Forests Leo Breiman and Adele Cutler. Retrieved from https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#varimp

Ho, T. K. (1995, August). Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 1, pp. 278-282). IEEE.

Jager, J. D. (2016, August 15). Predicting NBA Player Positions. Retrieved from https://nycdatascience.com/blog/student-works/predicting-nba-player-positions/

Liaw, A., & Wiener, M. (2018, March 25). Package 'randomForest'. Retrieved from https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

Practical Tutorial on Random Forest and Parameter Tuning in R Tutorials & Notes: Machine Learning. (n.d.). Retrieved from https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/tutorial-random-forest-parameter-tuning-r/tutorial/