Hierarchical Modeling to Predict Shot Outcome in NBA Games

Abstract

Knowing the probability of a shot being made is essential in basketball. There are many different factors that can affect the probability of a shot going in such as shot-distance, nearest-defender distance, shot type, time left in the quarter, home/away, etc. In this paper, we look at trying to predict the probability of a shot being successful for 133 players during the 2014-15 NBA regular season. After using classical machine learning models, we used three other methods to improve our accuracy. Other people that have worked on this problem, using random forest algorithms have been able to predict with approximately 61% test accuracy. Since a subset of our predictors affect each player differently, we used a mixed effects model. We used a generalized linear mixed effects model (Julia) as well as the mixed effects random forest model (MERF python package). Finally, we used Bayesian modeling (RStan). We found that we had the most success with the MERF package in python so far. This project can help distinguish "good" shots from "bad" shots, differentiate player strengths, and provide insight into the factors that most influence shooting percentage. The highest test accuracy obtained by our results was with MERF at 68.847%, without all predictors. This model provided a 0.449% improvement in test accuracy and significant improvement in true positive rate when compared to the standard random forest regression model (without incorporating random effects).

Introduction

Predicting the outcome of a shot in the NBA is an inherently difficult task, with five players moving at different speeds, different positions on the court, at different times of the game, along with the unique factor in basketball of two different clocks to consider. With the data available to us, not every player and distance combination occurs, making predicting outcome more difficult. Only shots during the run of play were used when trying to predict the outcome, also excluding blocked shots. These variables, along with countless others, make the task of predicting the shot outcome arduous, no matter how sophisticated the algorithm. Each player is influenced by each factor on the court differently, which needs to be accounted for by the model.

In this paper we compare the effectiveness of classical machine learning models and hierarchical modeling techniques when predicting the outcome of a shot in the NBA regular season. The hierarchical techniques model each player differently, which can allow coaches to better determine what is a “good” shot for his/her players. Coaches will have the ability to better identify what the best shots are based on the players on the court. With limited data, we recognize that the recommendations for some players will not be as effective than those with more data. However, despite the sparsity of data for some players, coaches can still determine what factors influenced them more than others.

Prior Research

There has been an intense amount of research into predicting the outcome of shots and the quality in NBA games. Chang et. al. introduced the idea of calculating the probability of a shot going in, comparing it to the rest of the league, then determining if it was a good shot. Wright et. al., using factorization machines, created a shot recommendation system for NBA

coaches *[6]*. Our approach differs from the approach in Wright et. al. in that we desire to provide coaches with more in depth recommendations. The system in *[6]* gives coaches the most and least desirable types of shots. We look to expand on their results to recommend shots for specific players. The recommendation system created by *[6]* showed that the most desirable types of shots are dunks and layups, which are easy to predict because they are the closest to the basket. The idea that each NBA player is affected differently by each variable was explored in *[3]*, in which the authors show that the "valuable space", where a player is most advantageous to their team, is different for each player. In our paper, we expand on that idea in that not just space is different for each player, but all (non player specific) player predictors are different.

Data

Our data came from combining two different public sources. The first is a Kaggle dataset that contains positional, closest defender, and timing data for every NBA shot in the 2014-2015 season. The second is from eightthirtyfour.com, and contains play-by-play data for all the shots described in the Kaggle. This data specifically added the ability to include the shot type (layup, fadeaway, etc). In order to reduce variability, we trimmed this dataset to only include shots made by the 133 players who had played in at least 58 games in 2014-2015 and at least 116 games combined in the two seasons prior. We chose the baseline number of 58 games per season because it is the official amount of NBA games required to be considered in statistical analysis.

We also utilized data from basketball reference to incorporate individual player shooting percentages from the 2012-2013 season and 2013-2014 season. Our final dataset contains 35 variables for every unblocked field-goal attempt by one of the 133 players. We did not include blocked shots because the play-by-play description for these shots did not include any information other than the fact that the shot was blocked. This filtering resulted in a dataframe with 68,986 shots from the 2014-2015 season. The 34 different variables we included are:

Location (Home or Away), Game Clock, Shot Clock, Number of Dribbles, Touch Time before shooting, Shot Distance, Shot Type (2 or 3-pointer), Closest Defender Distance, Shot Distance divided by Closest Defender Distance, Past field-goal percentage (average from prior 2 seasons), past free-throw percentage (average from prior 2 seasons), Past 3-point percentage (average from prior 2 seasons), Shot Number for a given player in the game, End of Shot Clock Indicator (last 4 seconds), End of Game Clock Indicator (last 4 seconds), Quick Shot Indicator (first 5 seconds of the shot clock), Clutch Time Indicator*, Garbage Time Indicator*, Period, Point-differential, Jump-Shot Indicator, Putback Indicator, Layup Indicator, Bank Indicator, Dunk Indicator, Driving-shot Indicator, Tip-shot indicator, Pull-up Indicator, Fadeaway Indicator, Running-shot indicator, Hook-shot Indicator, Reverse Indicator, Turnaround Indicator, Finger-roll Indicator. *[[1]] [[2]] [[7]]*

*Clutch Time was calculated using the NBA definition (4Q or OT with less than 5 minutes and point differential is less than or equal to 5). Garbage time was calculated when the game met the lead in the fourth quarter was greater than the number of minutes left plus 9.

In order to measure out-of-sample accuracy when testing our machine learning algorithms, we split the data into training and test sets by player. For each player, the training set includes 70% of the data and the test set includes the remaining 30%.

Methodology

To start our exploratory data analysis, we used classic machine learning algorithms: logistic regression, K-nearest-neighbors, random forest regression, random forest classification, and XGBoost. These algorithms were used as a baseline measurement as they were performed on the entire dataset, with no use of subsets or clustering. To incorporate the individual qualities, tendencies, and capabilities of certain players, three approaches were used: a Bayesian hierarchical approach in RStan *[8]*, a mixed effects model run in Julia *[4],* and a mixed effects random forest in Python *[11]*. For each model, we measured its performance by computing test accuracy (number of correct predictions in the test set / total number of test observations), the sensitivity (true positive rate), and specificity (true negative rate).

Our first approach was Bayesian hierarchical modeling. The purpose of using the Bayesian approach in RStan is that we hypothesized that each player had a unique set of coefficients.

$$P_{ij} = \mathrm{B}_{j_0} + \mathrm{B}_{j_1}\mathrm{X}_{\mathrm{i}j_1} + \cdots + \mathrm{B}_{j_k}X_{ij_k}$$

$P_{ij} =$ Probability of success for i^{th} shot of player j

$$\gamma_{ij} = \log\left(\frac{P_{ij}}{1 - P_{ij}}\right)$$

$\gamma_{ij} = \sum_{k=1}^{K} B_{jk} * X_{ijk}$ (with k covariates)

$B_j = \left(B_{j_1} \ldots B_{j_k}\right)$ (with k covariates)

$N_j =$ the number of shots for player j

$B_j =$ vector of prediction coefficients for player j

$B_j \sim \gamma_k(0, \sigma^2)$ for each j

With uninformative prior distributions of each variable, we ran 5000 simulations (iterations) for each players shots. In order to have a large effective sample size along with convergence of the chains, the following parameters were set in the function: thin = 10, warmup = 1000, treedepth = 20, adapt_delta = .99. Having a larger than default treedepth (10), increases the computational efficiency. These parameters led to the convergence of the posterior distribution of each coefficient for each player. While the high adap_delta parameter and large sample size did not maximize computational efficiency, post warmup divergences may indicate that the posterior distributions are not accurate. Without these parameters, the effective sample size was low (>200) and divergences would occur. Past percentages were not used in this method because it increased computation time by days and the posterior distributions did not converge. Because the computational efficiency of this method was minimal, we explored other methods for predicting shot outcomes.

Our second approach was a generalized linear mixed effects model to predict shot outcomes with certain coefficients tailored to each individual player. This type of model allows predictor variables with different distributions (i.e. binomial) to be part of the prediction. In order to create a mixed effects model, we began by working with the “glmer” function from the “lme4”

package in R. This proved to be too inefficient to create the model on such a large and high-dimensional dataset, however, so we quickly transitioned to the "MixedModels" package for Julia *[4]*. Specifically, we created a model following the formula where $P(Y = 1)$ indicates a successful shot:

$$y = \frac{e^{\beta_0 + X\beta + Zu}}{1 + e^{\beta_0 + X\beta + Zu}}$$

where y is the outcome of the shot, β is a vector of fixed effects coefficients, X is a vector of fixed effects variables, u is a vector of random effects variables, and Z is the random effects design matrix. Initially, we chose past free throw percent, past field goal percent, and past three pointer percent to be fixed effects and made the rest random effects. The reasoning behind this was that every player is affected by his past performance the same, but they are not all affected the same by shot variables.

Finally, we utilized the mixed effects random forest model (MERF) in Python. Since we attained success using standard random forest algorithms, we wondered if we could incorporate random forest within a mixed effects model. We discovered a package called MERF in python *[10]*, which implements a mixed effects random forest model. This package was developed by Manifold Inc. to create an alternative for mixed effects regression models *[11]*. This model utilizes a random forest regression for the fixed effects, but models the random effects linearly. Note that the formula utilizes random forest regression and therefore, the predictions from this model are not probabilities. Additionally, one must note that the random effects are modeled independently. The equation for this model is given below:

$$y_j = f(X_j) + Z_j * b_j + e_j$$

j represents cluster id (Clusters signify the distinct values for the variable in which the random effects depend). We treat each player as its own cluster.

$y_j = (n_j * 1)$ vector of responses for shots taken by player j

$X_j = (n_j * p)$ matrix of fixed effects covariates for player j. Where n_j represents the number of shots taken by player j. And where p represents the number of fixed effect covariates

$f(.)$ = Random forest regression function for fixed effects

$Z_j = (n_j * q)$ matrix of random effects covariates for player j. Where q represents the number of random effect covariates.

$b_j = (q * 1)$ vector of random effect coefficients for player $j \sim N(0, \sigma_b^2)$

$e_j = (n_j * 1)$ vector of errors for player $j \sim N(0, \sigma_e^2)$

This algorithm requires y_j, X_j, Z_j to be known and uses an expectation-maximization (EM) algorithm to solve for unknowns σ_b, σ_e, b_j and $f(.)$. The EM algorithm is repeated iteratively until convergence. At every iteration, the package computes a measure of training error, generalized log loss (GLL). Although this training error typically increases with the number of iterations, overfitting is a legitimate deterrent for choosing a high number of iterations.

Predictions for MERF are fairly easy to generate given its predict() method. For new observations that include a familiar cluster id from the data used in training, the random effect is utilized in the prediction. However, for observations with new cluster ids, the predictions are generated solely using the fixed effects. Predictions are formed from the following equations:

For known clusters:

$$\hat{y} = f(X) + b_j Z$$

For unknown clusters:

$$\hat{y} = f(X)$$

For our problem, we chose the response to be a binary variable signifying if the shot was successful. We utilized player names to serve as the clusters because we wanted to model each player distinctly. We decided to utilize all the predictors in our dataset as fixed effects. For the random effects, we included all predictors besides past field-goal percentage, past 3-point percentage, and past free-throw percentage. The past shooting percentages remain constant for each player throughout the observations on the 2014-2015 dataset. Therefore, using these predictors as random effects would provide no value.

Although one may think that using variables as both fixed and random effects may introduce multicollinearity, the developers of MERF insist that this is not the case. The fixed effects are modeled using a random forest, while the random effects are modeled linearly. This modeling occurs separately and therefore, this utilization of variables in both portions of the model can be compared to using interactions of predictors in a model. For instance, it is perfectly acceptable to use predictors such as x1 and x1*x2 in the same regression model. Modeling the problem this way did not give any errors when implementing the MERF package in python.

Training with the MERF package allows the option to tune a few parameters. These two parameters are ‘n_estimators’, the number of trees to use in the random forest portion of the model, and ‘max_iterations’, which specifies the number of iterations to run the algorithm. MERF does not yet have the capability to stop on its own when it reaches a convergence. Instead, one must specify max_iterations to dictate the stopping point. We chose both of these thresholds in an effort to maximize test accuracy (defined below). We tested ‘n_estimators’ values from 200 to 1000, but did not go any higher than that due to complexity constraints. Similarly, we tested values for ‘max_iterations’ from 10 to 100 and chose the value that gave the best test accuracy. The values chosen for ‘n_estimators’ and ‘max_iterations’ were 750 and 30, respectively.

We trained the MERF model using the training set and attained predictions for the test data which ranged from -0.085 to 1.967. This regression is not particularly useful on its own because we want to obtain predictions as a classification (1 to predict a make and 0 to predict a miss). Therefore, we used the predictions from MERF to generate binary predictions by setting a threshold value. If the prediction was above a certain threshold, we would classify that prediction as a 1 and if the prediction was below that threshold, we would classify the prediction as a 0. We determined the threshold value (0.541) by testing values from 0.45 to 0.55 in increments of 0.001. We chose the threshold in this range that maximized test accuracy.

The MERF package is still currently being developed. The developers intend to add flexibility to use different algorithms to model the fixed effects (such as a boosting algorithms or neural networks) and would like to add optionality to use non-linear random effects.

Results

Method	Test Accuracy (%) Baseline=52.99%	Sensitivity	Specificity
Logistic Regression	64.562%	0.4551	0.8162
Random Forest Regression	68.398%	0.5075	0.8392
Random Forest Classification	68.644%	0.5637	0.7944
XGBoost	68.900%	0.6864	0.672
Bayesian Hierarchical Modeling	59.768%	.574678	.397971
Julia Generalized Linear Mixed Effects	63.451%	0.4210	0.8223
Mixed Effects Random Forest (MERF)	68.847%	0.5229	0.8341

Listed above are the test accuracies (computed as previously defined), sensitivities, and specificities generated from training traditional machine learning models and the three hierarchical models previously mentioned. As seen, the model with the greatest test accuracy and sensitivity was XGBoost. This suggests XGBoost predicts best on made shots. However, the MERF model yielded the highest specificity, which suggests that it is the best predictor of misses.

While the Bayesian RStan results were not great in comparison to the other algorithms tried, there were players that it did very well. The players that it did the best for were: Jeff Teague, Al Farouq Aminu, Lebron James, Trevor Booker, and Marc Gasol. For these players, the Bayesian approach did approximately 15% or better than guessing. While these players may not appear to have much in common they have one important characteristic, all of these players had the probabilities of guessing very close to .5, meaning if one guessed the same outcome for each shot, they would get ~50% correct.

With both the sensitivity and specificity close to .5, this indicates that the Bayesian model did not predict either makes or misses better. This model differed from others in that three of the players that it predicted worse on were big men, something that other algorithms did better on.

The original generalized linear mixed effects model (Julia implementation with only past performance as fixed effects) predicted shots with an overall accuracy of 0.63, a specificity of 0.76, and a sensitivity of 0.48. In an attempt to improve our results with these metrics, we also created a mixed effects model in Julia in which we included every variable as both a random effect and a fixed effect. This resulted in an accuracy of 0.635, a specificity of 0.822, and a

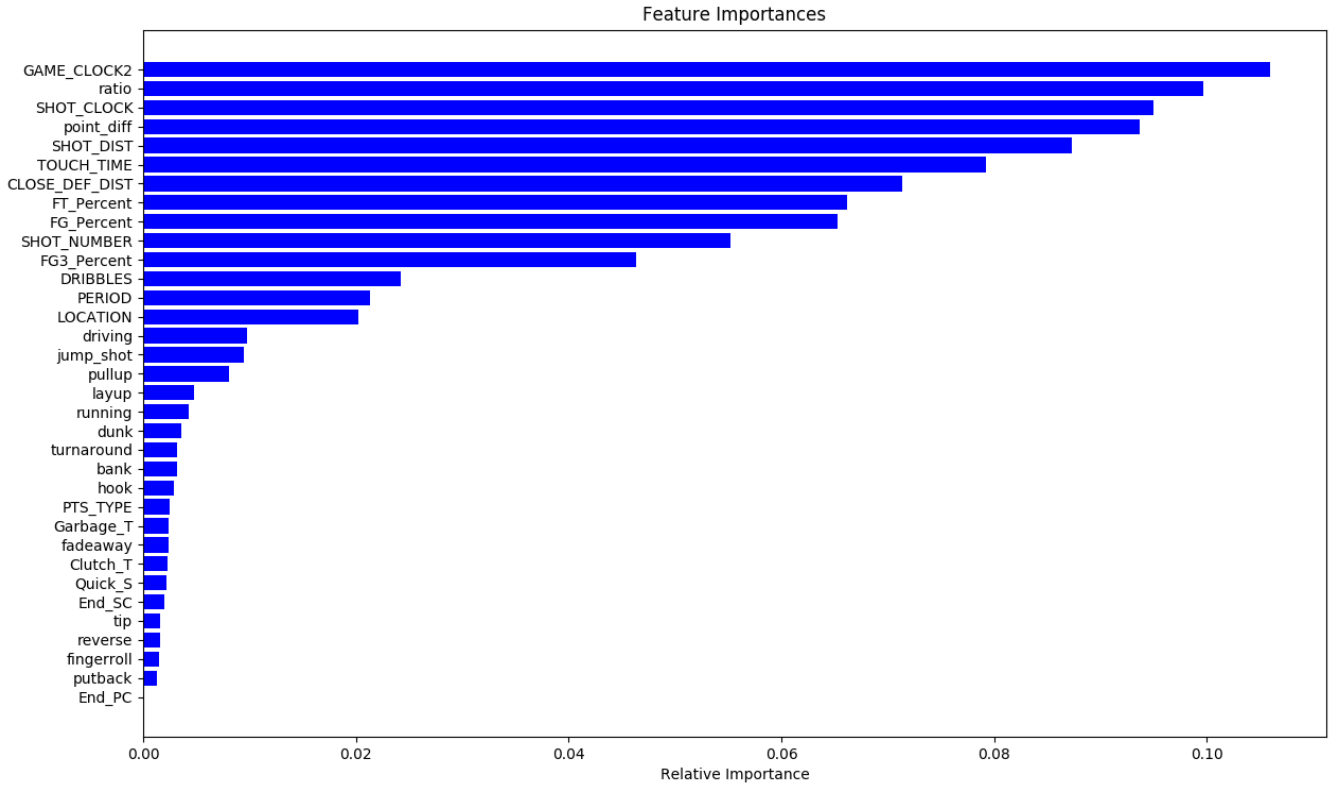
sensitivity of 0.42. While this model corrected predicted a slightly higher proportion of shots than the original, it produced a significantly worse sensitivity.

We also wanted to measure the results of both models in regard to how significantly they improved prediction accuracy from the baseline. We obtained baseline accuracy per player by predicting all of a player's shots as either makes or misses depending on which there were more instances of. Once we calculated the baseline accuracy for each player, we found the difference between the mixed effects model's accuracy and the baseline. For both generalized linear mixed effects models the average improvement in accuracy was 7%. Additionally, both models produced identical top five largest improvements by player. These were John Wall (+22%), Luc Mbah a Moute (+17.9%), Eric Bledsoe (+17.9%), Monta Ellis (+17.7%), and Marcin Gortat (+17.4%). These players are interesting because, aside from Luc Mbah a Moute, all of them frequently take shots from anywhere on the court. This shows that the model does not seem to favor shooters who only take one specific type of shot.

The results from the MERF model also seem to suggest that the model performs best on player's with a diverse shot selection. To accurately measure the marginal benefit of utilizing random effects in the MERF model, we compared the results of MERF to the results of random forest regression. This comparison isolates the random effects because the MERF model uses a random forest regression on the fixed effects. As seen in the first figure of this section, MERF provides a 0.449% test accuracy improvement over random forest regression. This is a significant improvement given the difficulty and randomness of predicting shots. The MERF model also has a higher sensitivity and specificity than the random forest regression model. The sensitivity is a much more significant improvement, suggesting that MERF is superior in its prediction of makes.

The top 5 players (that took at least 100 shots in the test set) in which saw the greatest improvement in test accuracy when utilizing the MERF model were: Kyle Lowry (+5.4%), Tim Duncan (+5.4%), Marreese Speights (+5.0%), Courtney Lee (+4.7%), and Mario Chalmers (+4.3%). As seen by these results, the players with the greatest improvement do not seem to have much in common. However, 6 of the top 12 players that saw the greatest improvement from MERF were point guards (Kyle Lowry, Mario Chalmers, Chris Paul, Tony Parker, Kemba Walker, and Mike Conley). Additionally, there were no dunkers or true three-point specialists that improved significantly with MERF, which suggests that MERF performs best with players who have a diverse shot selection.

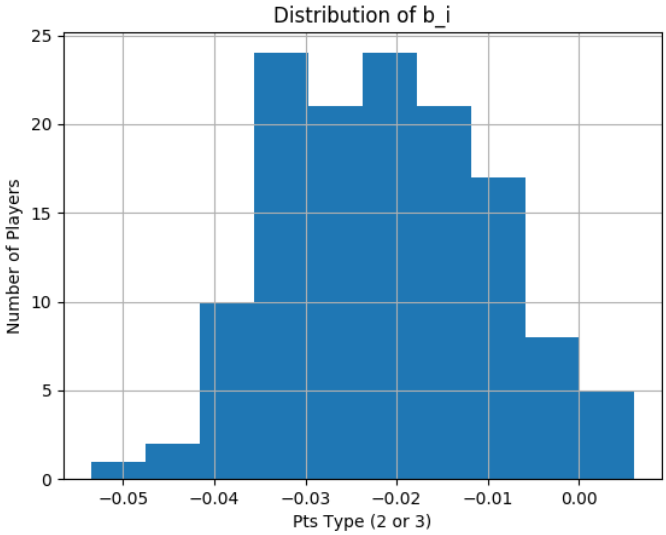
One of the benefits of using the MERF package, is that it allows one to analyze the fixed effects separately from the random effects. Below is the feature importance plot from the random forest regression (fixed effects):



*Note: $ratio = \frac{shot\ distance}{closest\ defender\ distance}$

The feature importance plot certainly shows some intuitive results such as the importance of shot distance, but some of the results are quite fascinating. The game clock, shot clock, touch time, and shot number are surprisingly significant covariates. Since these predictors are significant in the fixed effects portion of the model, it can be concluded that these features are generally significant across all players. Many shot type features such as fadeaway, reverse, finger roll, and putback were not important predictors in the fixed effects portion of the model.

One of the best features of the MERF model is the ability to look at the random effect coefficients for each individual player. This can give insight regarding which players are impacted most by certain features such as clutch time and points type (2 or 3-point shot). A histogram for the random effect coefficients for points type is shown below.



As seen by the histogram, basically all the coefficients for this variable are negative. This suggests that three-point shots reduce the likelihood of a made basket for most players, which is intuitive. However, the five players with positive coefficients are Joe Johnson, Brandon Knight, Steph Curry, Tim Duncan, and Al Jefferson. Duncan and Jefferson can be ignored because each took less than 10 threes on the entire season. However, Johnson, Knight, and Curry were all elite shooters from behind the arc in 2014-2015. These coefficients for certain covariates such as clutch time and end of shot-clock indicators can help coaches better understand their players' strengths and weaknesses. This can aid in decision-making and play-calling.

Discussion

The results of introducing random effects into models for shot classification definitely is thought-provoking. The results of all three hierarchical techniques yielded intriguing results, and even showed improvement over traditional models in the case of the MERF package. Not only are the results significant, but also the models are also valuable due to the potential to use the random effects coefficients to understand player strengths and weaknesses. This can help coaches, but it can also help broadcasters and fans become more intelligent and aware of the game. These models could provide an objective argument to the classic debate regarding whether a shot was a "good" shot. For instance, Damian Lillard's three-point shot to end the Thunder's 2018-2019 NBA season was highly debated. If our models suggest that a certain shot had a very low probability of success, it can be defined as a poor shot regardless of the outcome. Outcome bias often taints the perception of shot selection, which validates the usefulness of an objective measure for shot selection. Our research could be extended to provide a more thorough analysis by incorporating the opportunity cost of taking a certain shot. Perhaps one could study the expected value of the current shot compared to the expected value of the possible alternatives given the players on the court, shot clock, and game scenario. Our research solidifies the merit of using mixed effects modeling for this problem, but it also highlights areas for improvement. One of the main potential areas of improvement is the introduction of more predictors or more observations. Perhaps understanding the speed at

which a player was traveling or the height of the closest defender could give better prediction results. If we were given data that included the location of every player on the court, we could also see if a player was double-teamed for example.

For improving the Bayesian approach, one could group the shots both by player and by type (two or three point). With grouping by type, one could use past shooting percentages as prior probabilities in RStan, instead of uninformative priors. Lastly, to improve the generalized linear mixed effects model, one could experiment with different clustering strategies and group shots by something other than player. This may provide a better way to group shots and develop more significant coefficients. One other potential area to improve involves the MERF model. First of all, this model assumes that the random effect covariates are independent. The MERF package currently does not allow for the implementation of dependent random effects. The assumption that the random effect covariates are independent is likely not true and the ability to train the model without this assumption could improve the results. As previously mentioned, the developers of the MERF model, Manifold Inc., are still developing the package. The developers aim to add optionality for other models such as XGBoost for the fixed effects as well as the option to add non-linear random effects. These options could definitely help improve our model given the success of using XGBoost and other classical machine learning algorithms without the inclusion of random effects. All of these suggestions provide potential for future research.

Bibliography

[1] Kaggle nba shot logs. https://www.kaggle.com/ dansbecker/nba-shot-logs, 2016.

[2] NBA Play By Play Data By Season https://eightthirtyfour.com/data, 2019.

[3] D. Cervone, L. Bornn, K. Goldsberry. NBA Court Realty. MIT Sloan Sports Analytics Conference, 2016.

[4] A Julia package for fitting (statistical) mixed-effects models https://github.com/dmbates/MixedModels.jl

[5] 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.

[6] R. Wright, J. Silva, and I. Kaynar-Kabul. Shot Recommender System for NBA Coaches. KDD Workshop on Large-Scale Sports Analytics, 2016.

[7] CMU Reproducible Research Competition https://github.com/161siegels/Reproducible-Research-Competition

[8] Stan Development Team. 2018. Stan Modeling Language Users Guide and Reference Manual, Version 2.18.0. http://mc-stan.org

[9] R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/.

[10] S. Dey. Mixed Effects Random Forests in Python. Towards Data Science, 2017. https://towardsdatascience.com/mixed-effects-random-forests-6ecbb85cb177

[11] A Python package for fitting Mixed-Effects Random Forests https://github.com/manifoldai/merf/tree/master/merf