Sriraman Madhavan
Ezra Van Negri
Lisa Ann Yu

Nothing but Neural Nets: Predicting NBA Basketball Shot Success

Introduction

Late in the third quarter of the 2015-16 NBA Playoff Western Conference Semifinals, the Houston Rockets were staring down elimination. Down 3-2 in a best of 7 game series, they had a deficit of 19 points with under 15 minutes to play. In these key minutes, the Rockets found salvation in reserves Corey Brewer and Josh Smith who combined for 31 points over the final 12 minutes including 5 3-pointers. What made this comeback even more improbable was that Brewer and Smith were two of the NBA's 20 worst shooters that season--Brewer, especially, is nearly 30% less likely to make a given shot than an average player. As famed announcer Jeff Van Gundy often quips, "It is a make or miss league."[1]

To better understand what causes a "make" and what causes a "miss," our project examines basketball shot data and builds a predictive model that can evaluate the expected probability of a shot going in. Specifically, our model attempts to predict the shot made or missed binary response variable with two distinct feature sets. First, we attempt to model whether a shot will be made using the circumstances under which that shot was taken as features (our 'Shot Data Model'), such as shot type (e.g. dunk), shot distance (e.g. 5 ft.), and, in some models, the name of the shooter and his team. Second, we use coordinate data for the first second of the ball's trajectory to predict whether a shot goes in (our 'Coordinate Data Model'). We hope this study will provide insight into the ingredients of a successful shot. Our modeling will answer two questions. The first is a prediction question: from the circumstances of a shot, can we predict the probability it will go in? The second is an inference question: controlling for circumstances, which players are best at shotmaking and what factors have the biggest influence on shots being made?

These two models represent meaningful research in this space in two ways. First, the Shot Data Model represents a big step toward disentangling luck from skill in NBA shot results. For example, a team trying to maximize the expected value of its shots would have to differentiate situations when good shooters are taking bad shots from situations when bad shooters are taking good shots. These two instances might yield similar expected value but have drastically different consequences for how to improve. Second, little research has been done looking at shot trajectories, and our Coordinate Data Model represents early steps in what could be an interesting new field of research. Most basketball shot related research looks strictly at shot outcomes, but if we are able to predict shot outcomes from shot trajectories, we can examine factors that impact trajectories as a proxy for shot outcomes. This opens up research into what the ideal ball trajectory is, which has applications in shot training, and what factors most negatively impact shot trajectory, which has applications in coaching, specifically defense.

Related Work

Existing public research in this field is minimal, mostly due to the fact that the data needed to do the best analyses (SportVU data, the dataset we use) is no longer public and has generally been unavailable. NBA teams, of course, do extensive research on these questions but the results of their work are not made public. The research that is public exists in two forms. First, some have tried to create a primitive "expected effective field goal percent" (XeFG%) statistic which adjusts FG% for the league average field goal percentage in each area where a team's shots were taken.[2] In addition, there has been some older research on XeFG% using SportVU data before it became proprietary. That research was largely consistent with our findings: the Fansided XeFG% study on data from 2014-15 found that J.J. Redick and Stephen Curry as the 4th and 5th best shooters, relative to expectation; our logistic model placed them 5th and 6th.[3] More recently, research has been led by Second Spectrum, a company with access to the SportVU data that provides data and consulting services to NBA teams.[4] Unfortunately, their research is only made public through occasional articles from sportswriters that make use of their stats.[5] Not knowing what goes into their model makes it difficult to diagnose differences between their approach and ours.

The issues with proprietary data stifling research similar to our Shots Data Model is also present with our Coordinate Data Model research. Research in this field has been limited to just a few studies that were able to gain access to the SportVU data (we

[1] http://bleacherreport.com/articles/199800-the-nba-truly-a-make-or-miss-league
[2] See, e.g. http://hoopdata.com/teamxefg.aspx which has statistics by team for XeFG%. We call this metric "primitive" since it only adjusts percentages by the league average by zone, where more complicated models like ours control for more factors.
[3] https://fansided.com/stats/expected-effective-field-goal-percentage/
[4] https://www.secondspectrum.com/ourwork.html
[5] Some examples of sportwriters using these Second Sepcturm's shotmaking statistics include: Pelton, 2016 (http://www.espn.com/nba/insider/story/_/id/15456404/explaning-effects-shot-quality-shot-making-winning-games-nba) and Wagner, 2017 (https://fivethirtyeight.com/features/who-are-the-nbas-best-defenders-right-now/)

were only able to find three).[6] One such study, by Rajiv Shah, is the paper on which we based our general approach. Shah's paper used various machine learning techniques (Elastic Net, GBM, RNN, MDN) to model shot trajectories for three pointers. Our study is very similar, but attempts to generalize his approach to all jumpshots, not just three pointers. This makes our task slightly more difficult in that trajectories can vary significantly depending on shot distance.

General approaches include predicting whether a shot will be a make or miss by using as features the ball's trajectory alone, the trajectory of both the ball and the player, and static features such as shot type. Shah's study only looked at the trajectory of the ball, another study (a blog post) used kNN, logistic regression, 2D Gaussian, a mixture of 2D Gaussians, and NN with static features such as player, position, shot type, and initial location of player. A final study (Harmon et al. 2017) combined CNN and FFN to predict all shots from multiagent (i.e. all 10 players + ball) images of the ball's trajectory. We only looked at the ball's trajectory due to limited time dedicated to preprocessing the data. This problem is difficult, and Harmon et al.'s best model had an error rate of 39%. In our opinion, using a nonlinear NN which accounts for the sequence of the ball's trajectory is the best approach, and given more time, is the approach we would have taken.

Dataset and Features

Our data come from SportVU tracking cameras which are installed in the rafters of NBA arenas.[7] These cameras capture the (x,y) coordinates of every player and the (x,y,z) coordinates of the ball 25 times per second throughout the game. This data comes in two formats. The first is a dataset containing information on 205,020 shots taken during the 2015-16 NBA season. This is the 'shots dataset' which we split into 143,698 (70%) training, 40,678 (20%) dev, and 20,644 (10%) test examples and used to create our Shots Data Models. The shots data originally contained the result of each shot (make or miss) and 21 descriptive features about the circumstances of the shot, many of which we preprocessed. Of the 21 features, 11 were either redundant (i.e. Player Name and Player Id) or irrelevant (e.g. Game Id). Of the 11 remaining features, six were multi-level factor variables, and in particular Player Name had 437 levels and Action Type had 58. Any model that includes these variables, therefore, would be relatively sparse, even with over 140,000 training examples. To alleviate this issue, we split Action Type into two variables: Action Type Shot and Action Type Style, which are five- and ten-level factors respectively. This retained the two key pieces of information contained within the Action Type variable while reducing the feature space. We also used the coordinates of the shot location to create Shot Angle which measured the angle between the shooter and the basket. This variable ranged from -90° to 90°, except in rare circumstances. Other features used from the raw file include: Shot Distance, Shot Zone Basic (e.g. Mid-Range), Shot Zone Area (e.g. Right Side), Shot Zone Range (e.g. 8-16ft.), Player Name, and Team Name.

For our Coordinate Data model, we merged on (x, y, z) coordinate data associated with the ball's trajectory for each shot. This was a non-trivial task. We first used Event Id which links each shot to the coordinates for that shot. Unfortunately, each Event Id represented about 20 seconds of game time. From this, we wanted to extract only one second of coordinates associated with the ball's trajectory during the shot. To do this, we looked for the point at which the ball's coordinates were closest to that of either basket, then tracked that to the time at which the ball's coordinates matched the shot distance listed in the shots dataset. From that time point we took the next 20 (x, y, z) coordinates which should represent the first 0.8 seconds of the shot. These 60 coordinates were the features in our Coordinate Data model. After merging on the coordinate features, we limited our dataset to only jumpshots, since other kinds of shots (e.g. layups) have very different trajectories than jumpshots. Even though there remains significant variability within jump shot trajectories, this narrowed scope offered a better model for this subset.

Methods

Due to the particularities of the Shots and Coordinate datasets, we used a different set of models on each. For the Shots Data we used five supervised learning algorithms. Within the Shots Data we used two feature sets: the 'full' model, which contains dummy variables for each player (437 variables) and team (30 variables), and the 'partial' model, which excludes those two feature sets.

For the full model we knew the large feature space (494 variables after expanding factors into dummies) required significant regularization to avoid overfitting.[8] Furthermore, we needed at least one model that would be useful for inference so we could determine which players or conditions had the largest impact on shotmaking. Therefore, we chose to run logistic regression with Lasso regularization and a neural net on the full model. Standard logistic regression applies the sigmoid function ($g(z) = 1 / (1 + \exp(-z))$) to a

[6] See, e.g., this study on free throw arcs, http://www.inpredictable.com/2015/05/introducing-sharc-shot-arc-analysis.html
[7] https://www.stats.com/sportvu-basketball/
[8] Alternatively we could have done some feature selection, but algorithms like forward stepwise selection would not be helpful with many level factor variables. Therefore, methods that employ heavy regularization and can do automatic feature selection are preferable for our purposes.

linear combination of the features to produce outputs as probabilities. These can then be discretized into binary outcomes by thresholding. Lasso changes the typical squared error cost function by adding the L1 norm of the feature coefficients times the hyperparameter lambda. This penalizes coefficients for being too large, thereby shrinking their values. Thus, logistic regression with lasso performs a sort of natural feature selection, zeroing out coefficients that have only a small impact on reducing squared error, which is useful when we have many potentially uninfluential features.

Next, we applied a neural net with RELU activation. A neural net can be thought of as an ensemble of many logistic regressions, or in this case logistic regressions with a "kink" due to the RELU activation. The neural net has neurons, which distill our X variables down into derived features. By choosing the number of neurons and layers, a neural net is able to capture more complex data structures than most machine learning algorithms. Because we have variables that are highly correlated and our response may be the result of complex interactions between variables, a neural net is an appropriate modeling option. We used limited-memory BFGS rather than stochastic gradient descent as an optimization algorithm. L-BFGS is a popular choice for running machine learning algorithms in one batch on a single machine, as we did.[9]

For the partial model, we still needed regularization but since the feature space was significantly smaller without the player and team variables it is less of an issue. Thus, for the partial model we tried Random Forest, Gradient boosting machine (GBM), and logistic regression with elastic net. Random forest is a tree-based algorithm that uses a random subset of the features to create decisions trees that split the data into two categories based on split points in the various variables. Since decision trees are too deterministic, random forest creates many such trees from different feature subsets then averages over those trees to create a composite tree model. This method tends to work fairly well, but despite its use of "bagging" (tree averaging) the method tends to overfit. GBM is another tree-based method but builds trees that help minimize the residuals of a function that approximates a function minimizing expected loss. GBMs use gradient descent and have slightly nicer regularization properties than random forest, in that they can regularize both with a shrinkage hyperparameter and by use of stochastic gradient descent. Logistic regression with elastic net is similar to lasso, but uses a combination of the L1 and L2 norms to regularize the logistic function. Elastic net has a nice property of offering slightly less regularization (especially for groups of correlated variables) while still setting coefficients to zero.

For the coordinate dataset we considered a number of approaches, but settled on a neural network to model this problem. Neural networks are advanced pattern recognition algorithms capable of extracting complex, nonlinear relationships. Since many trajectory features such as ball speed, angle, etc. involve complex interactions between the time series coordinates of the ball, a neural net seemed most appropriate. This model was also the highest performing technique in Shah's earlier trajectory modeling work. We use grid search to narrow down the hyperparameters, and report the results in the next section.

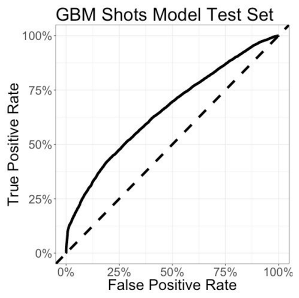
Results and Discussion

Our primary model evaluation metric was AUC (area under the curve). We had no reason to treat false positives as better or worse than false negatives, and are not interested in one particular point metric. Instead we wanted a summary metric, and chose AUC, since it is more interpretable than log loss. As described above we applied numerous machine learning techniques to our Shots Data. The results of these methods are listed in the table below along with the test set AUC curve for the best model, the GBM. For reference, about 45% of shots attempted are made, so if we predicted a miss for every shot, our AUC would be around 55%. For the logistic regression Lasso and Elastic Net, the hyperparameter lambda was selected through 10-fold cross validation. For the elastic net hyperparameter alpha, which denotes the split between L1 and L2 regularization (where 1 represents Lasso and 0 represents Ridge), we conducted a grid search on the dev set for alpha values 0 to 1 by 0.1 and found that alpha = 0.8 resulted in the highest AUC.

For the GBM, there are four hyperparameters. We conducted a grid search over every combination of shrinkage (0.001, 0.01, 0.1), number of trees (200, 400, 600), interaction depth (2, 3, 4), and minimum observations in node (5, 10). Shrinkage had the largest impact on the AUC. All the models with a shrinkage of 0.1 outperformed those with a shrinkage of 0.01, which outperformed those with a shrinkage of 0.001. This aligns with intuition, since the data were highly collinear. Thus, we conducted a second grid search with shrinkage (0.8, 0.9, 0.11, 0.12), n.trees = 300, interaction.depth = 4, n.minobsnode = 10, and selected the combination shrinkage = 0.12, n.trees = 300, interaction depth = 4, minimum observations per node = 10 for our final model. For the neural net, the parameters to adjust were the number of hidden layers, number of units in each hidden layer, and the shrinkage parameter for L2 regularization. We tried various combinations of hidden layers and alphas to find the combination with the highest test AUC. Our final model was a 4-layer neural net of sizes (4, 3, 2, 1) and a shrinkage parameter of 0.1.

[9] This research indicates that L-BFGS is the preferred choice for "small" datasets that can be run in a single batch. Though our data has many rows, this applies. http://www.fuzihao.org/blog/2016/01/16/Comparison-of-Gradient-Descent-Stochastic-Gradient-Descent-and-L-BFGS/

Best Shots Data Models	**Train AUC**	**Test AUC**
GBM (Partial)	0.666	0.657
Random Forest (Partial)	0.673	0.657
Logistic Regression (Lasso, Full)	0.656	0.654
Logistic Regression (Elastic Net, Partial)	0.654	0.653
Neural Net (Full)	0.624	0.609



Given that our training AUC was similar to our test AUC, we likely did not overfit to the training set. Rather, this is a bias problem, and additional features would improve our AUC. Specifically, the features defender distance and shot contestedness, which are known from other research to have an impact on shot probabilities, would likely improve our model. It should also be noted that basketball shots are highly random and thus particularly high AUC values are not anticipated.

Since we had many shot models, we only examined the errors closely for the best model, GBM. Model recall (41.0%) is quite low. Given that a shot was made, we only classified a small portion of those correctly. Our precision is better (65.2%), although still quite low. Given that we predicted the shot was made, a majority of those actually did go in, although we had many false positives. The total percent of shots predicted as made is close to the truth; the GBM model predicted 45.3% of shots attempted went in, while in reality, 45.5% of the shots went in. The breakdown by Shot Type (91.7% of dunks, 56.1% of hook shots, 63.0% of jump shots, 59.5% of layups, and 63.9% of other shots were classified correctly) reveals that certain types of shots are easier to predict than others.

The Coordinate Data Model is strictly a prediction task, and for this we used a feed forward neural network. A neural net was ideal for this task due to the complex data structure and the potentially different impact trajectory might have at various distances (which were not in the model). To tune the neural net we did a grid search over the hyperparameters Regularization (0.0001, 0.001, 0.01), Number of layers (1 to 10), Activation function (tanh, identity, relu, logistic), Number of neurons, and Learning rate (0.001, 0.005, 0.01). The best combination was a network with 8 layers of 128 neurons each with RELU activation, a learning rate of 0.001, and a regularization parameter of 0.0001. This combination resulted in a train AUC of 0.852 and a test AUC of 0.712.

As we can see from the results, the coordinates model outperformed the shots data model. This could be because many of the features used in the shots model (such as shot distance and angle) could be implicitly derived by the neural network from the initial time series coordinates of the ball. Furthermore, for some close shots we likely included the actual coordinate of the ball passing through the basket, which was probably influential in helping the model correctly predict.

The logistic regression model also allowed us to perform inference. By including each player and team in the full Lasso model, players and teams with the largest coefficients are those with the largest impact on shooting percentages, controlling for shot circumstances. The table below (left) reports the highest and lowest player coefficients.

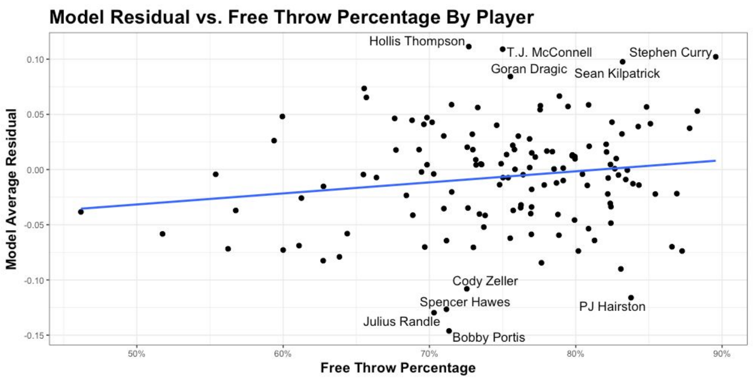
Regularized Logistic Regression Player Coefficients			
Highest Player Coefficients	**Log Odds Impact**	**Lowest Player Coefficients**	**Log Odds Impact**
Carl Landry	1.453	Tony Wroten	0.527
Karl-Anthony Towns	1.369	Joakim Noah	0.608
Jared Dudley	1.335	Terry Rozier	0.624
Andre Miller	1.319	Festus Ezeli	0.702
JJ Redick	1.308	Corey Brewer	0.704
Stephen Curry	1.272	Alex Len	0.712

GBM Shots Data Model Feature Importance	
Partial Model Feature	**Importance**
Shot Type (e.g. Jump Shot)	50.68
Shot Distance (e.g. 5 ft.)	34.03
Shot Style (e.g. Running)	11.24
Shot Angle (e.g. -15°)	1.96
Shot Court Zone (e.g. Midrange)	1.26
Shot Range (e.g. 8-16ft.)	0.65
Shot Side (e.g. Left Side)	0.17
Field Goal Type (e.g. 3 pointer)	0

As with any model, we hope to see the results align mostly with intuition. In this case the results are encouraging. Stephen Curry is perhaps the greatest shooter of all time. J.J. Redick and Jared Dudley, beyond being talented shooters, had career years in 2015-16 producing the first and seventh highest 3-point percentages in the NBA. On the other side, Joakim Noah has one of the worst shooting forms in recent memory and struggled to score from 0-10 feet from the basket, shooting about 20 percentage points below league average in that zone. Two surprising results were the appearance of Karl-Anthony Towns and Carl Landry among the NBA's best shooters. The numbers indicate both shot exceptionally well on long 2-point shots (10+ feet), an area where most players shoot poorly. Because both players are tall and do not have a reputation for being great shooters, many of these shots were likely wide open, a key omitted variable in our model. It is also possible both had a lucky year. In particular, Landry, a 9-year veteran, shot 12 percentage points above his career average from 10-16 feet, a reminder of the randomness of basketball shots.

The table above (right) uses the partial GBM model to examine the factors besides players and teams that are most influential in shotmaking. The top category is not a surprise. The shot type includes “dunk” and “layup,” which are both extremely high probability shots, and “jump shot” which is relatively low probability. The logistic regression coefficients for these factors were very large in magnitude, corroborating this result. It is also interesting that the influence of shot angle and shot side is so small, indicating that a 5 foot field goal from the side of the basket is roughly equivalent to a 5 foot shot from directly in front of the basket.

For most models it is difficult to evaluate how well the model is doing, especially when modeling highly random activities. Is 0.7 AUC a good result for this task, or should we do better? For our Shots Data Model an additional analysis helps us evaluate our model. Normal shots vary significantly on many dimensions including distance, type, and defensive pressure. A free throw, an unguarded shot from 15 feet, represents a natural control condition. We hypothesize that if our model perfectly controlled for all aspects of a shot’s circumstances, except the shooter’s identity, then the model residuals aggregated by player would reveal the most skilled shooters. Furthermore, we would expect these residuals to be highly correlated with free throw percentage. Thus, the correlation between these metrics represents a measure of how well our model controls for all factors external to a shot. As you can see in the graph, there is a slight positive correlation. Some of the names up top, such as Stephen Curry, indicate that we are on the right track. However, there are clearly omitted variables in our model. Information about defense is likely a large portion of this, and we would expect this correlation to improve significantly were defense features added.



Limitations and Future Work

As referenced above, the main limitation of our Shots Data model is the low AUC which we believe to be due to omitted variables. In particular, we have no information on the location of nearby defenders. Defender distance is a key determinant of shot success; the closer the defensive player, the more difficult the shot and the lower probability of success.[10] For future work we want to incorporate defender location, which can be done by merging together the Shots and Coordinate data and calculating the distance of each opposing player at the time of the shot.

The main limitations of the Coordinate Data model are twofold. First, we believe there are likely some data processing issues that impact the performance of our models. For example, the model performs extremely poorly on shots 1 foot from the basket. This makes sense; shots that close are over in less than a second, so our one second of data captures all sorts of things that might happen to the ball after it goes through the basket. In the future we might want to process the data differently for different length shots, or perhaps control for shot distance. The second limitation and area for future work is that a recurrent neural network (RNN) would better capture the relational aspects of the coordinate data. Shah did this and got very good performance; we tried to do the same but were unable to properly implement the RNN such that it performed well.

Conclusions

Our project looked at shotmaking from two angles. First, we explored a model that would account for the circumstances of a shot to see if it would go in. We tried a number of statistical modeling techniques and though our models did not perform as well as we would have liked--probably due to omitted variables--we were still able to perform well enough to get interesting inferences from our model. Second, we explored a model that would explicitly use the trajectory of the ball to predict shot success. Although this model was far more difficult from a data processing standpoint and we were not able to try as many modeling techniques, we were still able to outperform our simple shots model. We think this research represents meaningful analysis that would help NBA teams predict shot outcomes, one that could be expanded with better data and more time. Ultimately, the NBA is a make or miss league, but that process might not be as random as it seems.

[10] This has been seen in numerous studies. One such study looked at 3 point FG% vs. defender distance and highlighted the performance of some of the league’s best and worst shooters. Note some of the familiar names. https://imgur.com/a/sXiM3

References

Python
Scikit Learn
Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

Pandas
Wes McKinney. **Data Structures for Statistical Computing in Python**, Proceedings of the 9th Python in Science Conference, 51-56 (2010) (publisher link)

Numpy
Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. **The NumPy Array: A Structure for Efficient Numerical Computation**, Computing in Science & Engineering, **13**, 22-30 (2011), DOI:10.1109/MCSE.2011.37

R
Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22. URL http://www.jstatsoft.org/v33/i01/.

Tyler Hunt (2016). ModelMetrics: Rapid Calculation of Model Metrics. R package version 1.1.0. https://CRAN.R-project.org/package=ModelMetrics

Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan and Tyler Hunt. (2017). caret: Classification and Regression Training. R package version 6.0-76. https://CRAN.R-project.org/package=caret

Greg Ridgeway with contributions from others (2017). gbm: Generalized Boosted Regression Models. R package version 2.1.3. https://CRAN.R-project.org/package=gbm

Hadley Wickham (2017). tidyverse: Easily Install and Load 'Tidyverse' Packages. R package version 1.1.1. https://CRAN.R-project.org/package=tidyverse

Marvin N. Wright, Andreas Ziegler (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. Journal of Statistical Software, 77(1), 1-17. doi:10.18637/jss.v077.i01

Related Work
Beuoy, Michael, "Introducing ShArc: Shot Arc Analysis," *Inpredictable*, March 26, 2015. http://www.inpredictable.com/2015/05/introducing-sharc-shot-arc-analysis.html. Accessed 2017.

"Expected Effective Field Goal Percentage," *FanSided*. https://fansided.com/stats/expected-effective-field-goal-percentage/. Access 2017.

Harmon, Mark et al. "Predicting Shot Making in Basketball Learnt from Adversarial Multiagent Trajectories." (2016).

Pelton, Kevin. "Pat Riley is right. The NBA really is a 'make or miss league'", *ESPN*, May 4, 2016, http://www.espn.com/nba/insider/story/_/id/15456404/explaning-effects-shot-quality-shot-making-winning-games-nba. Accessed 2017.

"Predicting Basketball Shot Outcomes." https://enalisnick.wordpress.com/2014/11/24/predicting-basketball-shot-outcomes/

Second Spectrum, https://www.secondspectrum.com/ourwork.html

Shah, Rajiv C., and Rob Romijnders. “Applying Deep Learning to Basketball Trajectories.” *KDD, Large Scale Sports Analytic Workshop*, San Francisco, August 13-17, 2016. https://arxiv.org/abs/1608.03793. Accessed 2017.

Wagner, Kyle. “Who Are The NBA’s Best Defenders Right Now?”, *FiveThirtyEight*, November 28, 2017. https://fivethirtyeight.com/features/who-are-the-nbas-best-defenders-right-now/. Accessed 2017.

Data
Coordinate Data: https://github.com/rajshah4/BasketballData/tree/master/2016.NBA.Raw.SportVU.Game.Logs

Shots Data: https://github.com/sealneaward/nba-movement-data/tree/master/data/shots

Other data sources: https://www.basketball-reference.com; http://stats.nba.com/

Contributions

Sriraman preprocessed the coordinates dataset, created the feed forward neural network for the coordinate model, and edited the poster and report.
Ezra preprocessed the shots dataset, conducted the inference portion, and wrote a majority of the poster and report.
Lisa Ann created the shot models, conducted error analyses, and edited the poster and report.