Analysis of the Effects of Positional Features on NBA Shot Efficiency

Rahul Bhatt, Rohan Suresh, Shloak Jain, Aditya Bollam
Sports Analytics Group at Berkeley (SAGB)
University of California, Berkeley

ABSTRACT

We present a shot classifier for the NBA trained on positional player data and player ratings data that classifies a shot attempt into a make or a miss. This classifier has an accuracy of 56.23% on test data, and the precision, recall, and f1-score of our classifier was 0.54, 0.56, and 0.51 respectively. We also present a shot evaluator as an extension of this classifier that outputs a value, 0 to 1, corresponding to the efficiency of a shot. The evaluator was created using a combination of the probability values of our classifier and the type of field goal attempt.

The question this paper attempts to answer is what positional features correlate with made and missed shots. The relative strength of features, in terms of predicting made or missed shots, are summarized in figure 6. Our final model is a logistic regression model that uses positional features in combination with a heuristic based on the player's shooting ability. The coefficients of the model are listed in figure 4. The positional features were extracted from game positional data. We proceeded with a logistic regression model in particular as it works well in practice when there are multiple features being used to make a classification decision. Additionally, it is a lot easier to interpret the coefficients of a logistic regression model as opposed to a neural net. Interpretability is a key component to our research and the question we are attempting to answer. A novel aspect to our research is that it extends the classification model to create a shot evaluator that can be used to predict what features correlate with a quality shot. We believe this shot evaluator provides great value to teams trying to eliminate inefficiencies in the types of shots they take. After analyzing various features' effects on the shot evaluator's score, we concluded that distance from shooter to the basket had the highest correlation with a shot going in or not, and understandably so. One surprising result we also noticed is that distance to the nearest teammate had the second largest impact on the shot evaluator's score. This seems to imply that spacing is very important, especially in the modern NBA.

We believe that our research can have a significant impact on modern teams as it demonstrates a way to quantify the efficiency of a shot based solely on positional data, which all NBA teams now have access to. Our hope is that this analysis can be used by NBA coaching staffs to design efficient offenses.

BACKGROUND

In today's fast-paced shot-heavy NBA, designing plays to get shooters high-percentage looks at the basket becomes a crucial part of coaching. But before this problem can be tackled, it's first important to understand what makes a shot good. As there is a plethora of shot-time factors that affect whether the ball will go in or not, a robust model is needed to get to the heart of this problem. To address this issue, we have created a scoring efficiency classifier, which gives a binary reading on whether a shot will go in, as well as a shot evaluator, which rates the quality of a shot. With our models, coaches are more apt to design plays with higher likelihood of success and can quantitative view the impact of factors like if a shot if catch and shoot on its expected point value.

We explored a set of games in the 2015-16 NBA regular season involving the Portland Trail Blazers and used player tracking data to build these models. We chose the Portland Trail Blazers because we felt that their roster and play style in 2015-16 was an accurate representation of a modern NBA roster with high production from their guards and their perimeter style pick and roll offense. Rather than trying to expand the models to all NBA teams, we decided to focus on one so that we could use features that were personalized to the particular team and its style.

The game data was presented in 2 files: a game file and an events file. The game file contained a large table of the player and ball positions sampled every 0.04 seconds. The events file contained information about each play that occurred in the game, each characterized by an event type. We focused on event types 0 and 1, made and missed field goals, in order to build our models. After cleaning our data set and getting it into usable form, we then developed a set of utility functions such as finding shot times, player locations at the time of a shot, distances traveled by players, movement before a shot, and determining the type of shot. We could use these utilities to build out features and to build relevant visualizations.

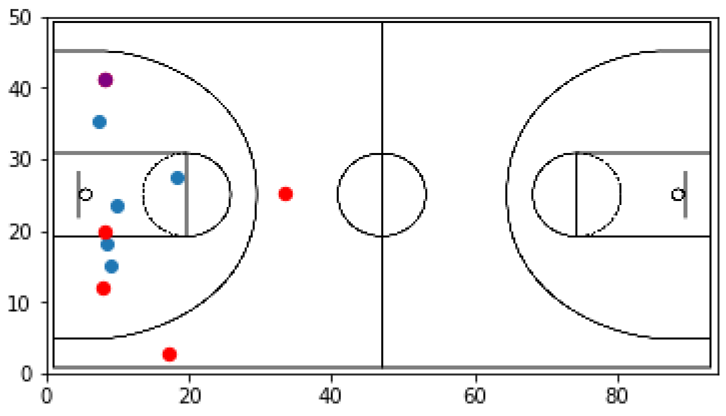

Figure 1

The above figure shows the locations of all 10 players at the time of a shot. The player in purple is the shooter, and his 4 red teammates are the Portland off-ball players. The 5 blue players are the Memphis players on defense.

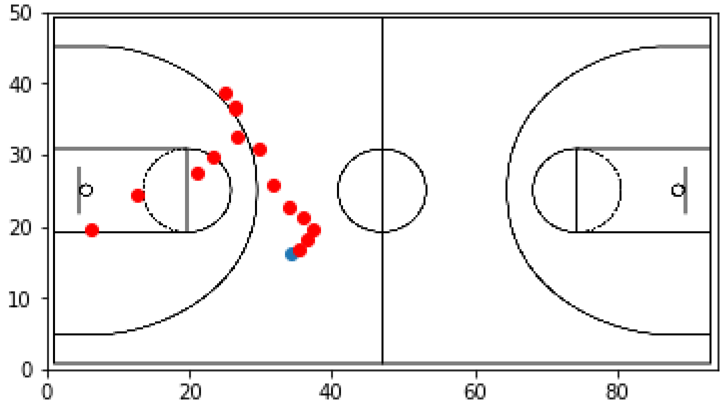

Figure 2a

The above figure shows the tracking of a single player over a possession. The player's starting point is the blue circle. This movement is broken down in Figure 2b and 2c.

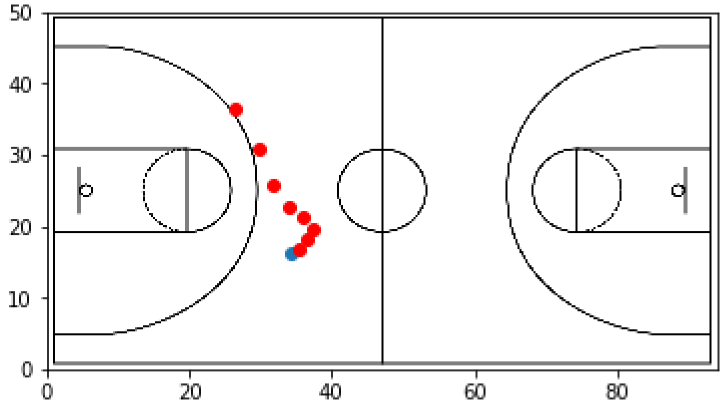

Figure 2b

The above figure shows the first part of the player's movement in the possession (starting at the blue circle) as he is moving along the perimeter to the right wing.

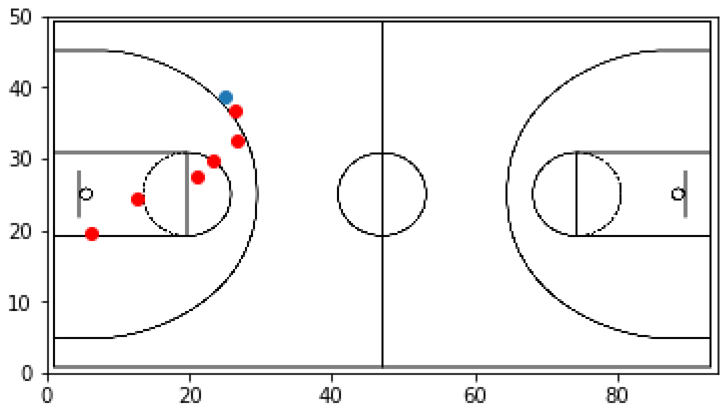

Figure 2c

The above figure shows the second part of the player's movement in Figure 2a. After the player moves to the right wing in figure 2b, he crosses back over to the left and cuts toward the basket.

Figures 2a-2c show how fine tuned the player tracking data is. This level of detail allowed us to generate very descriptive features.

We also took advantage of the NBA 2K16 players rating to factor in skill level of players. With this data, we created utilities that looked at factors like the shooter's and defender's ratings. The purpose of including this was to add a baseline heuristic into our model when predicting whether a shot would go in or not. After deciding on our features, we used our utilities to generate our training data and test data to analyze our models' performance.

Our goal with these models is to provide coaches with a way to estimate the expected value of their team's shots during the game as well as a way to help them provide quantitative feedback to their players about what factors they are doing well on and those they can improve on to increase offensive efficiency.

DATA

We used player tracking game data from a subset of games from the 2015-16 NBA regular season as the basis for our features and analysis. Each file represented the player positional data for a single game. The file {game_id}.csv corresponds to a unique game having that game ID number. For every 0.04 seconds, the game file contains an entry of the (x, y) position for every player on the court and the ball, along with a radius field that represents the height of the ball at that instant. Each game file has a corresponding events file {game_id}.csv; they are related by the same game id number. The events file contained information about each play that occurred in the game, each characterized by an event type. We focused on event types 0 and 1, made and missed field goals, in order to train our shot evaluation model. With respect to training data, we trained on all the Portland Trail Blazers games in the dataset. We chose the Portland Trail Blazers because we felt that their roster and play style in 2015-16 was an accurate representation of a modern NBA roster and to decrease the variability that would come from training on different teams. In practice, this training set can easily be modified to work on any given team's games. In the end our dataset contained over 6900 shot attempts and the various positional features associated with each shot. We found this dataset was large enough for the model we were developing as randomly changing the subset of the data we'd train the model on, did not significantly affect the model's performance.

Additionally, we also used player ratings data from NBA 2K for the 2015-16 season in order to develop features involving player skill. Using BeautifulSoup, we were able to scrape the data from 2Kmtc in order to integrate the data into our features. The main 2K data points we used were offensive ratings for inside and outside shooting to factor in the skill level of the shooter. If the player shooting the ball was not present in the 2K dataset, the shooter was given an average inside/outside rating. Thus, with a mix of positional features (team) and skill features (individual), we can effectively evaluate each shot.

TECHNOLOGIES

We built many tools to pull features that we wanted to test. To gather features of interest we used Python and various libraries like Pandas and a CSV library. Our data was kept in file format, so we did not need to query a database each time. We also used Matplotlib in Jupyter notebooks to create plots and visualizations. The main tool we used to create all our features was Pandas. We were able to use Pandas' CSV reading and writing capabilities to interact with our player tracking data in the form of dataframes. This was critical for us to filter our game files to build each feature. To make our visualizations, we interacted with our datasets and feature outputs in Jupyter notebooks and used matplotlib to make our visualizations in the notebook cells. Finally, to train and build our model, we used scikit-learn to fit regression models to our evaluator.

METHODOLOGY

We had access to very rich player tracking data. Figures 2a-2c show how fine tuned the player tracking data is. This level of detail allowed us to generate very descriptive features.

The first step in analyzing our data was to develop some utility functions that made analyzing the large player positional dataset easier. One utility function we created was determining the time of a shot. As mentioned before, we had access to a game file and events files. The events files had a tag for events that were attempted shots, and this included a time. However, after visually inspecting film, we discovered that these times were inaccurate, so we wrote our own script to determine when a shot was taken by utilizing a join of the game and events files. In order to do this, we selected all the times where the ball was nine feet off the ground and the event number associated with that time. We joined this data with the events file, where the event number was equal. Lastly, we selected rows where the tag indicated a shot, and the remaining list of times were our shot times. Once we had a list of times, this was vital for producing many of our features. Also, determining player location at the time of a shot became an easy lookup in the games files.

After developing utility functions to more efficiently store and query the player positional data, we worked on extracting features. We first used these features to create a shot classifier, with the first step of finding the instant on the court when a shot was taken. This way, we could have a clear snapshot of all the players' locations on the floor at the moment of a shot.

However, with a high level of variability the predictor model was not extremely valuable by itself. We realized it would be more useful for a team to see what features are important to an efficient shot. Thus, we then pivoted to an evaluation metric in which we were not bound to binary predictions, but rather a range that indicated how valuable a shot was (based on an expected value model). Additionally, we were able to add more features such as player skill and defender distance to enhance our initial model itself.

These are the following features our final classification and evaluation models were based on:

Number of offensive players outside arc
One of the features we decided to integrate into our model is the number of offensive players outside of the arc. We felt this would be a good feature to capture the spacing of the floor at the time of a shot.

Distance to nearest teammate
Another feature we used was the shooter's distance to the nearest teammate. The idea behind this feature was that to determine the impact that spacing had on shot quality. Moreover, closer distance could be indicative of play types like pick and rolls.

Bench ratio
We looked at the first ten players on the court at the start of the game and called them the starters while every other player was called a bench player. For each shot, we looked at the team who

was shooting and provided a ratio of number of bench players to the original five starters for that team. This was paired with whether the shot went in or not and was integrated into the feature matrix.

Distance to basket

Another feature incorporated into our model is the distance from shooter to the basket. Obviously, this feature had a strong influence on our model's predictive power as the distance a player is to the basket is one of the biggest factors in determining the difficulty of a shot. This was confirmed when examining the coefficients of our model, as the coefficient corresponding to this feature had the largest absolute value of 0.0295.

Distance to nearest defender

Another feature we used was the distance to the nearest defender from the shooter. This feature would have high predictive power because if a defender was close, the shot was likely to be contested whereas a farther defender would suggest an open, higher percentage shot.

Player shooting rating

While most of our features focus on distance and positioning, a key factor is still the skill level of the player himself. Using an outside dataset, we were first able to map player names to player id's. This enabled us to map the 2k data based on player name to our player tracking data based on player id. At the time of each shot, we looked at the shooter's location on the floor (inside the paint or not) and then mapped it to either his inside or outside shooting ability, respectively. We could then run this for every shot taken in the game, outputting shooter rating.

Catch and shoot

This was a feature that required a complex script that utilized time ranges to calculate if a shot was attempted off a catch and shoot or attempted off the dribble. For each shot attempt, we analyzed a 2.4 second window prior to the shot release. During this window, the player closest to the ball was tracked, and if the player closest to the ball changed within 2 second of the shot going off, the shot was counted as a catch and shoot. To do this, we wrote a utility function that would track the ball and player movement during a possession. We limited our search to the shooting team only, ensuring the offense could be the only player closest to the ball when the shot was counted as catch and shoot.

Shot Evaluator

Once we had our finalized shot classification model, we pursued creating a shot "evaluator." This evaluator calculated the expected value of a shot based on all of the features discussed above and normalized the rating on a scale from 0 to 1. This was done using the probabilities of a shot going in using our logistic regression classification model and multiplying that with the point value of a shot attempt.

Once we had our shot evaluator we were able to see the individual effect each feature had on shot quality.

ANALYSIS

After training our classification model on our positional data we were able to analyze the coefficients for each feature and the performance of our model. In terms of our classifier's performance we saw that it had an accuracy of 56.23% on test data, which we believe is a reasonably high score due to the inherent randomness involved with shooting a shot. The precision, recall, and f1-score of our classifier was 0.54, 0.56, and 0.51 respectively. In total our test data made up 30% of our entire dataset, which came out to a total of 2077 shots analyzed. Figure 3 shows more detailed metrics broken down by shots attempts that were a miss vs a make.

	Precision	Recall	f1-score	support
Miss (0)	0.58	0.85	0.69	1171
Make (1)	0.5	0.19	0.28	906
Avg / total	0.54	0.56	0.51	2077

Figure 3: Logistic Regression shot make/miss classifier Metrics

In order to better compare the relative strengths of each feature on whether a shot would go in or not we decided to normalize our input data. This was done using the mean and standard deviation of each of our features and feeding in this normalized data matrix into our logistic model. Doing so yielded an accuracy of 56.14%. Our final logistic regression models had the following coefficients:

Coefficient Value	**Normalized Feature Coefficient Value**	**Description of Feature**
0.01198379	0.01312017	Number of offensive players outside the arc
0.01023649	0.06028284	Distance between shooter and nearest teammate
-0.00544025	-0.00119036	Ratio of bench players to starters on offensive team
-0.02859703	-0.24119803	Distance from shooter to the basket
0.00654641	0.0286847	Distance from shooter to the nearest defender
0.00310266	0.02204522	Shot rating of the player
-0.01409697	-0.00725429	If shot is a 3 or not
-0.07044214	-0.02704075	If shot is off a catch and shoot or not

Figure 4: Coefficients and corresponding feature for classification model

After analyzing the performance of the logistic regression model, we saw that a lot of our probabilities of whether a shot went in were being concentrated close to 0.5. This is indeed a limitation of logistic regression, as it tends to require very extreme feature values to have probabilities close to 1. We tried to mitigate this issue using parameter tuning, but also realized that the inherent randomness of a shot limits our ability to predict it with a high degree of certainty.

Our analysis shows that distance from the shooter to the basket had the largest effect on shot quality, which logically makes sense as shots closer to the basket have a much higher chance of going in than farther shots. We also saw that the feature with the second largest effect on shot quality was distance between shooter and nearest teammate with a coefficient of 0.060. This indicates that spacing plays a large role in the predictive power of our classifier.

Another feature that had a strong effect on shot quality was the shot rating feature. This feature did not rely on positional data and rather used the 2k rating of a player's shot ability as a heuristic for our classification model. Figure 5 demonstrates the relationship between the shot evaluator and the 2k rating.

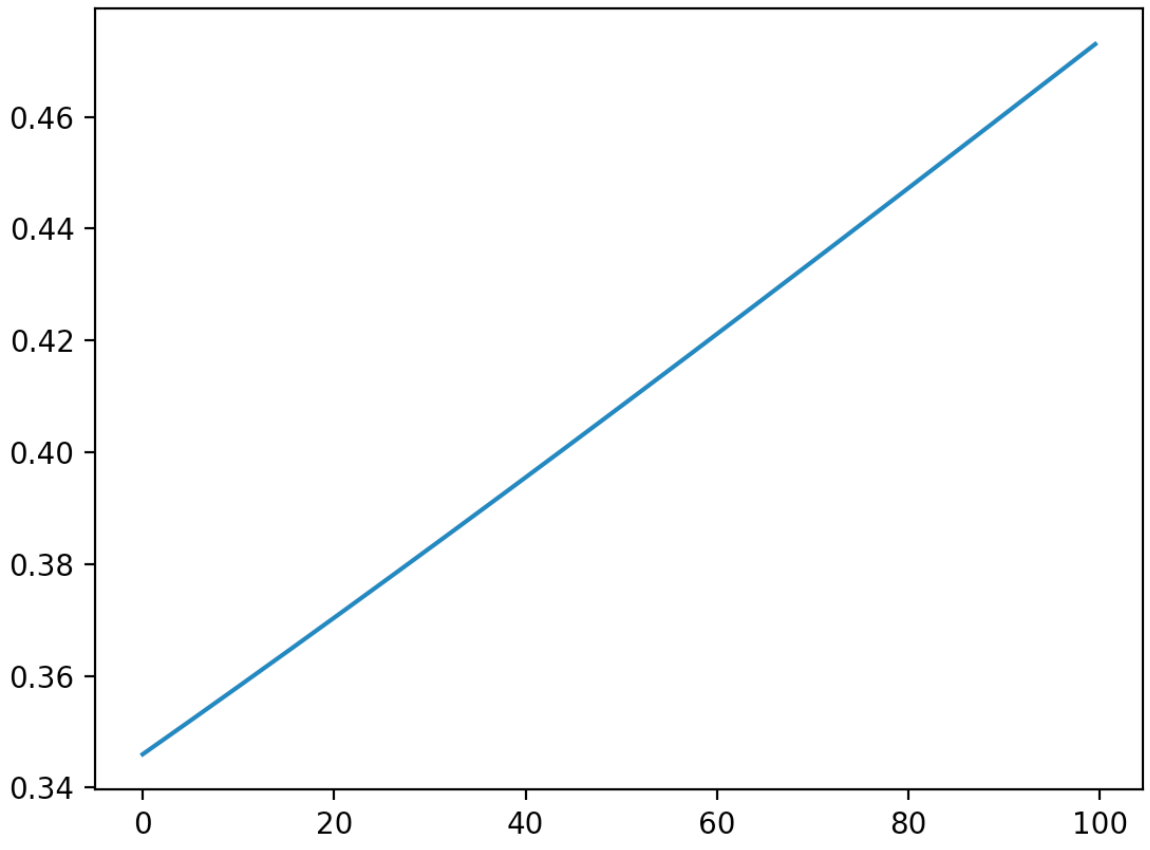

Figure 5: Shot Evaluator Score vs 2k Rating

Figure 5 clearly shows a strong linear relationship between 2k rating of a player's shooting ability and the value outputted by our shot evaluator. This validates the importance of shooter rating as a feature.

In order to visualize the relative strengths of each feature on predicting shot efficiency we plotted our normalized coefficient data as seen in figure 6.

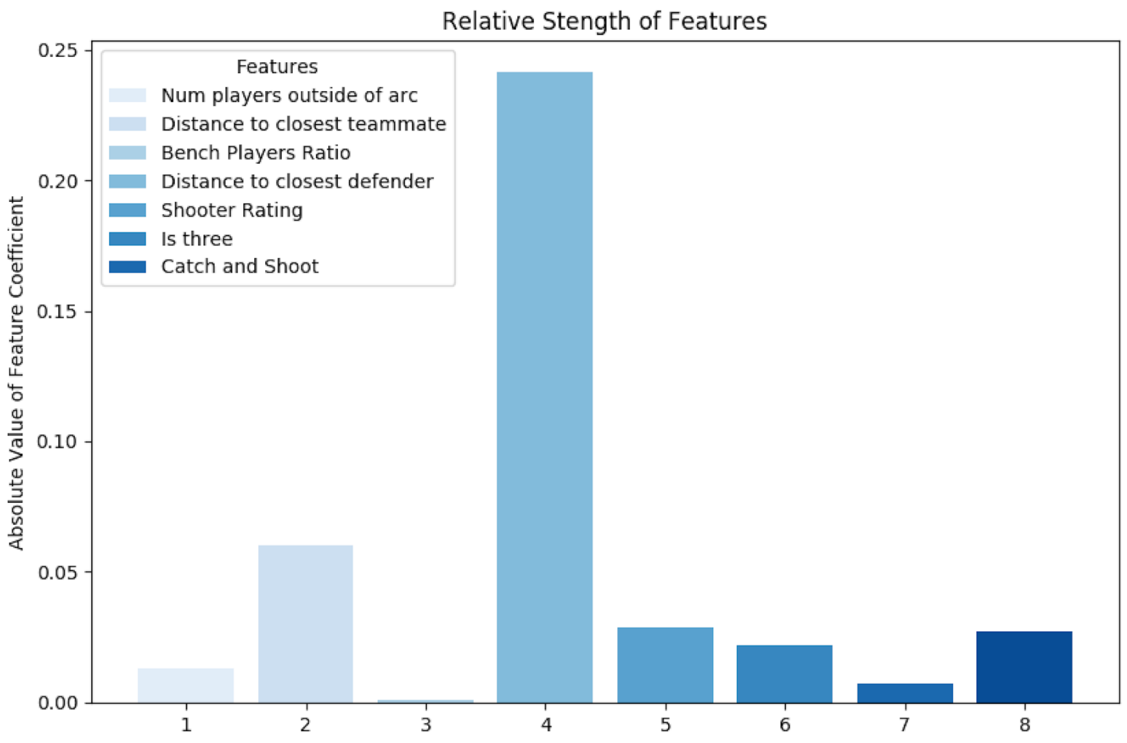

Figure 6: Relative Effect of Features on Shot Quality

After conducting our analysis we believe our classifier did a fairly good job of classifying a fundamentally random process. This is supported by a relatively high accuracy and coefficients for our features which appear to support basic assumptions about shot efficiency such as a closer shot being more efficient on average than a further shot.

ERROR

A significant chunk of error can be attributed to the quality of the data itself. One of the biggest data issues was inconsistency between timings in our player tracking data and over events file data. For example, each shot was taken at an earlier than in the player tracking file - based on the position of the ball - than what was reported in the events file. This misalignment, however, wasn't constant and had lots of variation. At first, we didn't realize but after getting some strange results, we watched the video recording of the game and noticed that the times of events in the actual game, and those stated in the events file did not line up.

Another data issue was duplication of rows in the player tracking data. We would find hundreds of duplicates at the same time stamp, making parsing through the file a challenge. A third issue was that, at times, sampling period wasn't uniform for rows in the player tracking file. It was supposed to be sampled at 0.04 seconds but there were many occurences where it was arbitrarily less than that. For all of these problems, sophisticated data cleaning scripts had to be written to mitigate the possible consequences.

CONCLUSION

After creating our model and analyzing its components and accuracy, we found that although our features were not very strongly correlated with shot efficiency, there was a clear ranking of relative importance between features. Most important, by a substantial margin, was distance from the shooter to the basket, which is an understandable result, given that, barring any other factors, shooting percentage is higher the closer one goes to the basket. Distance between the shooter and closest teammate also has an important role, which highlights the role of spacing in today's offense. Features whose coefficients had lesser magnitudes - signifying lower importance - include ratio of bench players to starters on the floor and number of offensive players outside the three point arc.

The overall test accuracy of our classifier was 56.23%, suggesting an upper limit to what can be determined by time-of-shot factors, alone. This suggests potential avenues for future work that takes into account possession-related factors like number of passes or off-ball movement into the model or even training on individual players. Furthermore, analysis can be done on the defensive side of the ball, to see how shot and possession based factors concentrated around defensive efforts impact the result of a shot. Still, the fact that we were able to have some success classifying shots based solely on player position data shows that player tracking data appears to have the potential to refine today's NBA offenses. Moreover, introducing an evaluator that can be used to quantitatively compare different shot attempts and allow a team to optimize for maximal value gives teams clear directions on what they can do with the large collection of data every game provides them.