Predicting NBA Shots

How well can we predict if a shot is made or missed?

Virginia Baskin  
 DSCI 303  
Rice University   
 vfb1@rice.edu

Harry Golen  
 DSCI 303  
 Rice University   
golen@rice.edu

ABSTRACT

The use of data analytics and machine learning has fundamentally changed the way basketball is played professionally. Optimized models can predict, with some accuracy, whether certain factors will influence the likelihood of a shot, or field goal, will be made. Any information that can be gained from EDA and modeling is extremely helpful for this competitive industry, so the aim of this present project is to predict if an NBA shot will be made or missed based off of easily recorded information about the game. The goal of this study is to predict if a shot is made correctly at least 60% of the time and find the best model to do so using model evaluations like accuracy score but also training and testing time complexity, since basketball is such a quick moving sport.

1 INTRODUCTION

In all of the history of professional basketball, there has only been one player unanimously voted Most Valuable Player (MVP): Steph Curry in 2016. Curry earned this honor not just through dominating the league offensively, but because he changed the way the game of basketball is played. Until players like Curry, his Golden State Warriors teammate Klay Thompson, and Houston Rockets guard James Harden came onto the scene, the National Basketball Association (NBA) had historically been ruled by inside scoring. Most of the greatest players of all time were centers or power forwards, and the strategy of basketball was mostly built around trying to get open shots near the basket (layups) or in the “midrange.” However, when Curry, the greatest shooter of all time, gained stardom, the league began to change. The Warriors popularized a style of basketball that was built around shooting three-pointers, and their revolution was built on statistics. Using the very simple math that three points are worth more than two points, they strayed away from shooting shots in the midrange, where the expected points per shot was at its lowest, and tried to take as many threes and layups as possible.

Ever since Curry and the Warriors changed basketball, the whole league has been attempting to quantify shot quality. By knowing which shots are most likely to produce the most points, teams can strategize around shooting only the best shots, and they can train their defense to allow shots that would be unlikely to go in while preventing those most likely to hurt their team. Many factors could potentially affect the likelihood of a shot to go in: which player is shooting it, how many players are defending it, the quality of the defense, the distance of the shot, the conditions in the game such as whether the team shooting is winning or losing, the location of the game, and more. More recently, NBA teams have employed high-tech ball- and player-tracking technology that collects data on players’ locations and velocities as well as data on the basketball itself.

NBA teams have created statistical departments that leverage their data to attempt to calculate what kinds of shots are most likely to go in (as well as initiate other data-related projects), and we similarly oriented our project around trying to predict shot data. Because the NBA is such a competitive league, no team would share their data or their models with the public, so we don’t have all the data at our disposal that a real NBA team would. Our dataset, which comes from Kaggle, is composed of all the shots taken in the 2014-15 season. It includes some basic player tracking data, such as shot distance and the distance to the closest defender, but it doesn’t have all the data a team would have at its disposal. Through our research, we found that NBA teams have used neural networks to predict whether shots would go in, and we use a neural network as well (in addition to several other candidate models). In this paper, we first describe some characteristics of our dataset and some notable conclusions we found through Exploratory Data Analysis (EDA), then we test a baseline model for our data followed by a description of our candidate (more sophisticated) models, and then we finally conclude our project by advocating for one model and describing our findings.

1.2 Problem Statement

Can we predict whether a shot will be made based on variables like closest defender, shot distance, shot type, time left on clock, previous number of shots, etc.?

Our goal is to predict whether a shot will be made with at least 60% accuracy. Even the best shooters miss wide-open shots sometimes (because a lot of shooting is random physics), so we know we won’t be perfect, but we believe 60% accuracy would be an impressive addition to basketball analytics.

1.2 Related Work

As mentioned above, the most cutting-edge machine learning for NBA shot analysis is performed by the professional’s team’s own statisticians and is kept under strict lock and key. There are, however, many instances of independent fans attempting to predict shots as personal projects, much like we do in this paper. We did not rely on nor reference these past projects, instead opting to explore and model the data independent of outside influence. We did read articles about the process of basketball analytics from NBA Stuffer, a reputable stats-reference that delivers unique metrics and NBA analytics content. In particular, the company highlights the use of neural networks in predicting shot quality, which motivated us to include neural networks in our model as well.

2 METHODS

We first did EDA, which involved understanding our data, cleaning it, engineering features, performing correlation analysis, and feature selection.

2.1 EXPLORATORY DATA ANALYSIS

2.1.1 The Data

Our dataset was sourced from Kaggle.com and originally contained 21 columns and 128,069 observations. The columns are as follows: **game\_id**: unique identifying key for every game/match up, **matchup**: information about game, including: month day, year and abbreviated team names (ex: FEB 26, 2015 - CLE vs. GSW), **location**: “H” or “A” indicating whether game is home or away for the player, **w**: ‘W’ or “L” indicating whether player's team won or lost, **final\_margin**: by how much did the player's team win or lose (the score for the shooting player's team minus the score for his opponent's team at the end of that game), **shot\_number**: integer of how many shots this player has taken in this game so far, **period**: what period this shot occurred during, **game\_clock**: the time remaining in that period when shot occurred (min:sec), **shot\_clock**: the time on shot clock when shot occurred (in seconds) (0 to 24), **dribbles**: integer of how many times they dribbled the ball once in possession and before shot, **touch\_time**: time player had ball before shot, **shot\_dist**: distance from hoop in feet, **pts\_type**: whether the attempted shot was a 2 or 3 pointer (free throws not in this dataset), **shot\_result**: whether the shot was made or missed, **closest\_defender**: lastname, firstname of closest defender to the player, **closest\_defender\_player\_id**: unique key identifying closest defender above, **close\_def\_dist**: distance in feet of how far defender was from player, **fgm**: 1 or 0 indicating if field goal (shot) was made or not, **pts**: points awarded for shot (0, 2, or 3), **player\_name**: firstname lastname(s) of player who took the shot, and finally, **player\_id**: unique key identifying player who took the shot.

Upon observation, it is clear that there will be almost perfect correlation between some features, like **shot\_result**, **fgm**, and **pts**, as they all reflect whether a shot was made. Since our goal is to predict **fgm**, we immediately were aware to exclude **shot\_result** and **pts** from any future feature selection.

2.1.2 Cleaning

Some aspects of the features of the dataset needed cleaning or reformatting before we could begin anything else. The feature **touch\_times** has negative values, which simply does not make sense in the context of the problem. A player can touch the ball for 0 seconds, but for not negative seconds. We found there were only 312 observations with negative touch time, so we simply removed these anomalies from the dataset.

Some observations for **shot\_clock** were missing (NA), and after inspection we found that this missingness was not random. Whenever there was less than 24 seconds left in the period, the shot clock value was not recorded, so we imputed the value for **game\_clock** as shot clock in those cases. Additionally, **shot\_clock** and **game\_clock** were in different time units, (seconds versus mins:seconds), so **game\_clock** was converted to all seconds to match.

Lastly, features that represented binary outcomes (**w**, **location**) were converted to 0 and 1, with the “better” option represented as 1 and worse represented as 0 (e.g. away = 0, home = 1; loss = 0, win = 1).

2.1.3 Feature Engineering

Because of the nature of our dataset, we were excited about the opportunity for feature engineering. As we will see, feature engineering provided a significant number of the variables in our analysis and improved the performance of our models greatly.

The first set of features we engineered is related to the clock. **full\_game\_clock** is the total number of seconds remaining in the game (as opposed to **game\_clock**, which is only the number of seconds left in the quarter). **buzzer\_beater** is a binary variable that is 1 if there’s less than 1 second left on the shot clock, and 0 if not. **overtime** is a binary variable that equals 1 if the shot was taken in overtime, and 0 if not.

We also wanted to account for some unique types of shots that are taken in basketball. **clutch\_shot** is 1 if the shot was taken in the last two minutes of a game that was decided by 5 or fewer points, and 0 if not. **putback** is 1 if the shot was taken within 5 feet with greater than 23 seconds on the shot clock, and 0 if not.

Finally, we wanted to look at who was shooting and who was defending the shot. **player\_shooting** is a variable that we calculated by grouping by player and adding a column that is the proportion of all shots in the database taken by that player that were made (this is commonly known in the NBA as field goal percentage, or FG%). **defender\_quality** is the inverse, so it’s calculated by grouping by defender and adding the proportion of shots that were defended by that defender that were made.

2.1.5 Correlation Analysis & Feature Selection

After cleaning, adjusted skewness, and feature engineering, there were 15 correlated features with fgm above 0.01:

shot\_dist–0.192518

pts\_type–0.121745

player\_shooting –0.105316

defender\_quality–0.077712

cox\_touch\_time– 0.062989

final\_margin–0.058861

putback –0.058657

w–0.050329

buzzer\_beater–0.048777

cox\_dribbles–0.043376

full\_game\_clock–0.016397

period–0.014091

clutch\_shot–0.012559

game\_clock–0.011591

overtime–0.010588

Note how all of the engineered features are correlated with fgm, confirmation that our feature engineering was successful and added value that was not previously in the dataset that future models can use in their prediction.

In order to reduce dimensionality and overfitting with our dataset, we attempted two types of feature selection. First, we used Sequential Backwards Selection (SBS) to select the top 12 group of features. In addition, we selected a different group of 19 features via domain knowledge and our correlation analysis. After testing via a baseline model, we ended up choosing the set of features we selected via domain knowledge and correlation analysis because that set had the best performance.

2.2 MODELING & RESULTS

For all of our modeling (and feature selection), we ensured we were not overfitting our data by splitting it into training, validation, and testing sets. 60% of the data was used for training, 20% for validation, and 20% for testing.

2.2.1 Baseline Model Results

First, we created a baseline logistic regression model with our cleaned dataset that hadn’t been adjusted for skewness nor had features engineered (to be able to evaluate the success of our preprocessing techniques). We normalized (standardized) the data and fit a basic logistic regression using Sklearn’s default parameters. We evaluated the model using the validation set, sequestering away the test set only to be used at the very end of our model building and tuning, as is good practice.

Baseline Accuracy: 0.6132

2.2.2 Candidate Models & Results

For our candidate models, we used the cleaned skew-adjusted dataset with our engineered features. We tested two different sets of features, one selected by sequential feature selection and the other from domain knowledge and what we discovered from EDA. After some initial testing, we found that the second subset of features, the domain knowledge set, resulted in slightly but consistently better accuracy scores on the validation set than the SBS subset. For our candidate models, we tuned hyperparameters using the GridSearchCV function from sklearn with a reasonable dictionary of parameters based on our knowledge of the algorithms.

**Test Accuracies:**

Random Forest: 0.61674

SVM: 0.61831

ANN: 0.62159

Naïve Bayes: 0.56226

KNN: 0.60320

2.2.3 Model Tuning via AWS

Some models were simply too computationally intensive to run on one laptop for days while still using that computer for other work. To address this issue, we created an AWS (Amazon Web Service) EC2 instance to run GridSearchCV in less time. To tune the ANN, what took at least over 5 and a half hours (that is just when I terminated the script on my computer), took EC2 22 minutes at 97% capacity. The instance type we used is “c5ad.8xlarge’, with 32vCPUs and 64 GB of memory.

To tune SVM’s hyperparameters, we were not so lucky. At first, I used a very powerful (and expensive) instance that tuned the model overnight, but then due to lack of foresight, I only output the validation accuracy and not the ideal hyperparameters. I was not able to run that instance again, due to cost, so I opted for the instance described above and ran a GridSearchCV. With fewer options, the resulting model was only slightly different than the default model, which was interesting, but overall SVM’s training time undermines any usefulness it could have in the world of basketball shot predictions.

DISCUSSION

Overall, we are encouraged by the results of our project. We set out to predict shot making with an accuracy of at least 60%, and almost all of our models exceeded that goal. Considering the practical ramifications of these types of models, our recommended final candidate is the Random Forest model. Although it did not have the absolute highest test accuracy, the two that had higher accuracy (SVM and ANN) both were extremely computationally expensive, and using the AWS cluster was difficult and expensive.

We believe that our research has potential applications in real-time during an NBA game (for example, a coach telling their players which shots they should try to take), in which case the Random Forest model has much more applicability. The ability to visualize the tree and see which factors were important is also relevant to this task.

With a project like this, there is so much more to working with data than creating models. You have to understand that dataset, how it was collected, what it represents, how different features interact with each other, if there are any anomalies (either by accident or by design), etc. By the time we got around to building models, we already had a good idea of what features are going to be important and contribute to our models‑‑so much so that our domain knowledge dataset consistently performed better against the sequential feature selection dataset.

REFERENCES

Data source: <https://www.kaggle.com/datasets/dansbecker/nba-shot-logs?resource=download>

Past projects: <https://hwchase17.github.io/sportvu/>

<https://www.the-iyrc.org/uploads/1/2/9/7/129787256/20_iyrc2020_35_final.pdf>

<http://cs229.stanford.edu/proj2017/final-reports/5132133.pdf>

Using neural networks in NBA predictions: <https://www.nbastuffer.com/analytics101/machine-learning-systems-in-sports/>

CONTRIBUTIONS

For EDA, Virginia completed data cleaning, visualizations, general explorations, skewness adjustments, and correlation analysis. Harry undertook feature engineering and feature selection. For modeling, Harry decided on and created the baseline model, Virginia made the candidate models with Harry’s help, and went on to tune and test all the models, using AWS when necessary for computationally expensive models. Virginia made most of the presentation slides with Harry contributing the rest. Virginia wrote sections of the final report, while Harry took on most sections of the report and additional tasks (formatting, editing, etc.).

GITHUB REPOSITORY

<https://github.com/virginiabaskin/NBAShots>