SHOT BY SHOT: CAN WE PREDICT NBA SHOT SUCCESS?

By: Jonathan Beyene

Overview:

In the game of basketball two teams compete head to head to determine which team can successfully score more points than the other. The most important aspect of the game is a player's ability to score, which involves successfully shooting a basketball into a net. In that sense, it would be extremely informative to a player, team, and overall basketball organization if they were able to predict the outcome of a certain shot selection. The goal of this project is to utilize machine learning models to see how well we can predict the outcome of a basketball shot. This involves using various classification models, conducting hyperparameter tuning, and identifying various important features to optimize our model's predictive accuracy in identifying whether a basketball shot is a make or a miss.

Related Work:

Albert Tan posted an article on the website Medium titled "Predicting the Result of an NBA Shot: Machine Learning Analysis Project". Here he looked at shooting data from the 2014-2015 NBA season and used features such as 'Shot Clock', 'Dribbles', 'Touch Time', 'Shot Distance', and 'Closest Defender Distance'. He trained seven different models with his best performing model being an Optimal Decision Tree Classifier with ADABoost using the following hyperparameters: max_depth = 6, min_samples_leaf = 17, n_estimators = 10, and learning rate = 0.19. This model ended up with an accuracy of 61.9% using these features.

How I plan to differ from this work is by utilizing different features. While I had hoped to include features such as ‘Closest Defender Distance’ I was unable to find good data for this metric and instead decided to capture defense through a team's defensive rating. Along with that, I decided to expand my feature space to include a player's shooting percentages, and other features such as one-hot encoding the type of shot action.

Another piece of relevant work I looked at was “Analysis of Machine Learning Models Predicting Basketball Shot Success” by Max Murakami-Moses. In this project many of the features found align with the features I plan on using, such as the shot action type, the minutes and seconds remaining in the quarter, and the location of the shot. Where I plan to differ is incorporating more features and using different model selection. In this work, the idea of defense is not present in the features, so I hope to include this aspect and see if it helps increase the performance. I also intend on incorporating a player's shooting percentages which I also hope improves accuracy. This project's best model had an accuracy of 65.1%, and although we are using different initial datasets, I hope that my additional feature implementation to my initial basketball shot dataset will have a higher accuracy.

Relevant Data:

In the beginning of my project attempt I utilized a dataset from Kaggle which scraped play-by-play data from the website “Basketball References” which contained every play from the 2015-2016 NBA season to January 20th of the 2020-2021 NBA season. The play-by-play dataset had a total of 3040524 data entries, however after subsetting the data for only rows where the play involved a player shooting the basketball, we ended with a total of 1144001 data values. I pivoted away from this dataset as it turned out to be very difficult to work with in correspondence to the other basketball datasets I intended to use. The biggest issue with this dataset was the formatting of the player names. For example, a player named “Trae Young” would be formatted as “T. Young”. This became an issue when I attempted to merge this play-by-play dataset with a player statistics dataset as some players could have the same abbreviated names. The player “Trae Young” and the player “Thaddeus Young” would both be formatted as “T. Young” and this resulted in problems with the merging.

The dataset I used for this project was found on github, as someone had already scraped shot data from NBA.com from the 2003-04 season to the 2022-23 season. This dataset included relevant numerical features such as ‘SHOT_DISTANCE’, ‘LOC_X’, ‘LOC_Y’, ‘MINS_LEFT’, and ‘SECS_LEFT’. It also contained categorical features such as ‘SHOT_TYPE’, ‘BASIC_ZONE’, ‘ZONE_RANGE’, and ‘ACTION_TYPE’, however ultimately I only included ‘ACTION_TYPE’. I decided to use ‘ACTION_TYPE’ as it contained a high level categorical of the type of shot. I deemed this is important as it brought in the idea of shot difficulty hierarchy with some shots being more difficult and therefore resulting in misses.

I wanted to implement a player's shooting statistics as features, so the second dataset I used was a player statistic dataset. To do this, I went to Kaggle and looked for comprehensive datasets that included a player’s name, their shooting percentages, the team they played on for those given shooting percentages, and the season in which they shot with those shooting statistics. While many of the datasets on Kaggle claimed to be comprehensive and include all players and their stats, I found that many were missing complete careers for some NBA players and others were missing random years for a given player with no statistics recorded. I ended up finding a dataset on a website called data.world, and although it wasn't perfect, it was much more comprehensive than any other of the player statistic datasets I had found. This data set contained features such as 'Player', 'Pos', 'MP', 'FG%', '3P%', '2P%', 'eFG', and 'Season'.

As mentioned above, the player statistic dataset wasn't fully complete, therefore when merging the player statistic dataset with the shot dataset resulted in null values. I took the most straightforward approach to tackling this problem by simply dropping these missing rows. If I were to come back to this project and improve it, for the players with missing seasons, I would average their previous season stats and fill those values in for the missing values. For the players with entire missing careers, I would create an "average nba player" profile which took all the shooting percentages and averaged it across all players and use those values to fill the missing values. Also to prevent data leakage, when merging these two datasets, I used the individual players shooting percentages of the previous season as features for the current shot. For example, if one of the shot data points was Lebron James shooting in 2017, the associated shooting percentages to this data point would be Lebron James's 2016 shooting averages.

For the most part, in a basketball game most shots are contested, so I wanted to incorporate some type of defensive feature in my dataset. Since my shot dataset did not include the nearest defender to the shot or the name of the nearest defender I decided to use the opposing teams defensive rating instead of an individual player. A team's defensive rating is an NBA advanced statistics and calculated using the opponents score and possession metrics (see Variable Appendix for formula). Since I limited the shooting dataset to only incorporate data points after the NBA merger occurred, finalizing the number of teams to 30 (occurred in 2004) I needed to find defensive team statistics for all the years post 2004. Unfortunately I was unable to find a team defensive rating dataset on kaggle or anywhere else online through my search. Since the span of years wasn't extremely extensive and for each season I would only need 30 data points as there are only 30 NBA teams, I decided to take the simplest approach first, and manually input the team defensive ratings from NBA.com in an excel spreadsheet. While I could have attempted to web scrape the data from NBA.com I made the decision that manually inputting it would be much more time efficient.

After merging the shot, player statistic, and defensive datasets I was left with a final dataset containing 2,947,521 data points. This dataset contained a mixture of both numerical and categorical features, so in order to utilize the categorical data, I one-hot encoded these features. The categorical features that were one-hot encoded include 'SHOT_MADE', and 'ACTION_TYPE' which resulted in my dataset having 83 feature columns and 1 target column.

From my domain knowledge of basketball, I choose these features as I believe that they have an impact on the event of a shot going in. The position of the player when they shoot the ball intuitively seems significant as if they are extremely far from the basket the likelihood of the shot going in decreases. I included the players shooting percentages as if the player taking the shot is a good shooter the models should account for that when predicting if the shot will go in. I used features like 'MP', 'MINS_LEFT', and 'SECS_LEFT' in hopes to capture intangible factors such as player fatigue. I also included the players positions to try to capture playstyle. For example, point guards usually take shots farther from the basket and have higher 3P% therefore I wanted the model to be able to take positional shot selection into account. Finally, I included the opposing teams defensive rating as shots are usually defended and I hoped to capture defensive pressure and its effect on whether a shot is a make or miss.

During my exploratory data analysis I explored how the different features are correlated with the result of a shot being a make (binary:1) or a miss (binary:0). The most correlated feature was ‘Driving Hook Shot’ which was one-hot encoded into the dataset and describes whether the shot was a driving hook shot or not. My basketball domain knowledge led me to bring in shooting percentages into the original dataset, and after completing the correlation analysis, ‘FG%’, ‘2P%’, ‘3P%’, and ‘eFG%’ are all within the twenty most correlated features with the target variable. To my surprise many of these one-hot encoded shot action features were in the top twenty most correlated features. To further explore this, I looked at the top twenty of these action types to see which was the most common, with ‘Driving Hook Shot’ being the most common type of shot by a landslide. In addition, I was curious to see which of these shot action types had the highest and lowest conversion rates in this dataset. After exploration, I found that the ‘Driving Hook Shot’ had the highest conversion rate and the ‘Driving Finger Roll Shot’ had the lowest conversion.

Analysis/Modeling:

Since my question is exploring how well we can predict if a basketball shot will go in, this is a classification problem with binary results, since a shot can either be a make or a miss. Since this is a classification problem we will be focusing on classification models. For selecting the hyperparameters, I had a large dataset and the tuning process took an extremely long time so I came up with two solution: either take a random subsample of the data and use a grid search then make the claim that the optimal hyperparameters in the subsetted data is representative of the entire data space or use all the data and use a randomized search. I used the latter option, as I was fortunate enough to have a lot of data and wanted to use it all. The randomized search has relatively similar results to a grid search and allows me to utilize all the data while speeding up the tuning process which is ultimately why I opted for this route.

Logistic Regression:

Logistic Regression is a classification model that can be used to make binary predictions. In the case of my project, we can model a shot going in as a 1 and a shot missing as a 0, and structure the problem into a binary setting. Since I am working with time series data, to avoid data leakage in the tuning process I used an incremental year tuning strategy. During the hyperparameter process, I made the distinct decision to use L1 Regularization only due to the size of my dataset. My data has 83 feature columns so I wanted to implement L1 Regularization to narrow down the important features. I also standardized the data before using it in the modeling as it sped up the fitting process. The hyperparameter space I explored contained the 'solver', 'max_iter,' and 'C' parameters and I used a randomized search to pick the optimal parameters. Since I am using a large dataset, the hyperparameter tuning process took a great amount of time using a regular grid search of the hyperparameter space, so I made the decision to use a randomized search as it gave me similar results and sped up the overall process. The final parameters used in modeling included {'solver': 'saga', 'max_iter': 10000, 'C': 0.001}.

Decision Tree:

A Decision Tree can be used in a classification setting and splits nodes by a specific decision rule in order to classify a datapoint. For the training process, I used an incremental tuning process as mentioned above to avoid data leakage. The hyperparameter space I explored contained the ‘max_leaf_nodes’, ‘max_features’, ‘criterion’, and ‘ccp_alpha’ parameters and I used a randomized search to pick the best parameters. As noted in the Logistic Regression section, I utilized a random search as it sped up the hyperparameter tuning process. For the final iteration of this model, I used the following hyperparameters: {'max_leaf_nodes': 20, 'max_features': 'log2', 'criterion': 'log_loss', 'ccp_alpha': 0.001}

XGBoost:

XGBoost is an ensemble method utilizing Decision Trees, where each new Decision Tree is fit on the residual error of the previous tree. Since my dataset is susceptible to data leakage due to basketball shot data being considered time series data, I utilized an incremental year tuning process. I would train in the first year, and use the second year as the validation set. Once a year has been used and seen in the validation set, in the next iteration of tuning I would include it in the training set and expand my validation set. I continued this process until I noticed that the optimal hyperparameters no longer changed. I explored the hyperparameter space that included ‘n_estimators’, ‘max_leaves’, and ‘learning_rate’, and used a randomized search to find the optimal parameters. The parameters used in my modeling were {'n_estimators': 500, 'max_leaves': 10, 'learning_rate': 0.1}.

Results:

After completing the hyperparameter tuning and modeling I was left with the following results (See Data Appendix). For the dataset in use for this project, the average FG% over the whole dataset was 45.58%. Subtracting this percentage from one we get 54.42% which is the average percentage of missed shots in this dataset. If we had an extremely simple model that predicted that every shot would be a miss, intuitively this model would have an accuracy of 54.42% therefore I used this as a benchmark for measuring the goodness of my model.

The final iteration of my Logistic Regression Model had a testing accuracy of 64.00% and the final iteration of my Decision Tree Model had a testing accuracy of 59.46%. My best performing model was the XGBoost model with a testing accuracy of 65.28%, showing a 10.86% boost in accuracy when compared to the benchmark. After looking at the feature importances for this model, it seemed like the categorical description of the action of the shot were the most important features which I found shocking, with the most important feature being whether or not a shot was a 'Driving Hook Shot'. After investigating why these categorical descriptions had such a big impact, the relationship eventually began evident. For example, in this dataset there were 1,379,367 attempts at a driving hook shot with only 475,897 resulting in a make, which resulted in a 34.50% conversion rate. Another important categorical shot action feature was 'Jump Shot' where there were 20811 attempts with 18797 of them resulting in a made shot, having a conversion rate of 90.32%. Based on this, it is easy to say that if the shot attempt was a 'Jump Shot' it will most likely be classified as a made shot, while if the shot was a 'driving hook shot' it will most likely be classified as a missed shot.

While my XGBoosted model has a 10.86% increase in accuracy when compared to the benchmark, this model as it stands should not be used yet in practice and can be adjusted. I was surprised to see that my domain knowledge that led me to include features such as a players shooting percentages across different types of shots and the defensive rating of the opposing teams did not translate into the model. While I don’t believe that my domain knowledge is incorrect, I have come to the conclusion that my implementation of these features was too general. If I were to expand on this project, I would look for a dataset that included player shooting statistics in a specific zone (Ex: Left zone, Restricted Area, etc.). Some players shoot better in specific areas on the court and less efficiently in others, and by narrowing the area for shooting percentage, it may result in better accuracy in shot prediction. I would also get more specific in my implementation of defense. I would instead try to find a dataset that included the closest defensive player to the shot, and try to append their personalized defensive rating instead of the teams defensive rating. For example, some players are more defensively gifted and may cause the shooter more difficulty in converting a shot compared to others. This methodology could narrow down the effects of defensive pressure and potentially improve results.

DATA APPENDIX:

Logistic Regression			
Hyperparameters	Feature Space	Data Range (Years)	Accuracy Score
Penalty: L1 Solver: Saga Max_iter: 10000 C: 0.001	SHOT_DISTANCE	Train: 2004 Val: 2005	Train: 59.42% Val: 59.65%
Penalty: L1 Solver: Saga Max_iter: 10000 C: 0.001	ALL FEATURES	Train: 2004 Val: 2005	Train: 64.17% Val: 63.56%
Penalty: L1 Solver: Saga Max_iter: 10000 C: 0.001	ALL FEATURES	Train: 2004-05 Val: 2006-07	Train: 63.87% Val: 64.41%
Penalty: L1 Solver: Saga Max_iter: 10000 C: 0.001	ALL FEATURES	Train: 2004-07 Val: 2008-11	Train: 64.16% Val: 63.89%
Penalty: L1 Solver: Saga Max_iter: 10000 C: 0.001	ALL FEATURES	Train: 2004-17 Test: 2018-21	Train: 63.58% Test: 64.00%
Decision Tree			
Hyperparameters	Feature Space	Data Range (Years)	Accuracy Score
Max_leaf_nodes: 20 Max_features: log2 Criterion: log_loss Ccp_alpha: 0.001	SHOT_DISTANCE	Train: 2004 Val: 2005	Train: 60.86% Val: 61.04%
Max_leaf_nodes: 20 Max_features: log2 Criterion: log_loss Ccp_alpha: 0.001	ALL FEATURES	Train: 2004 Val: 2005	Train: 60.65% Val: 60.70%
Max_leaf_nodes: 20 Max_features: log2 Criterion: log_loss	ALL FEATURES	Train: 2004-05 Val: 2006-07	Train: 60.87% Val: 60.65%

Ccp_alpha: 0.001			
Max_leaf_nodes: 20 Max_features: log2 Criterion: log_loss Ccp_alpha: 0.001	ALL FEATURES	Train: 2004-07 Val: 2007-11	Train: 60.59% Val: 61.08%
Max_leaf_nodes: 20 Max_features: log2 Criterion: log_loss Ccp_alpha: 0.001	ALL FEATURES	Train: 2004-17 Test: 2018-21	Train: 62.22% Test: 59.46%
XGBoost			
Hyperparameters	Feature Space	Data Range (Years)	Accuracy Score
N_estimators: 150 Max_leaves: 4 Learning_rate: 1	SHOT_DISTANCE	Train: 2004 Val: 2005	Train: 60.86% Val: 61.04%
N_estimators: 500 Max_leaves: 10 Learning_rate: 0.1	ALL FEATURES	Train: 2004 Val: 2005	Train: 65.19% Val: 63.92%
N_estimators: 500 Max_leaves: 10 Learning_rate: 0.1	ALL FEATURES	Train: 2004-05 Val: 2006-07	Train: 64.77% Val: 64.55%
N_estimators: 500 Max_leaves: 10 Learning_rate: 0.1	ALL FEATURES	Train: 2004-07 Val: 2008-11	Train: 64.83% Val: 64.70%
N_estimators: 500 Max_leaves: 10 Learning_rate: 0.1	ALL FEATURES	Train: 2004-17 Test: 2018-21	Train: 64.88% Test: 65.28%

Variable Appendix:

SHOT_DISTANCE: Distance of player to basket
LOC_X: Players x-position on the court ranging from -25 to 25
LOC_Y: Players y-position on the court ranging from 0 to 50
MINS_LEFT: Minutes left in the quarter
SECS_LEFT: Second left in the quarter
SHOT_TYPE: Type of shot (Either 2pt or 3pt)
BASIC_ZONE: Zone on court where shot was taken
ZONE_RANGE: Distance from the basket of zone
ACTION_TYPE: Description of shot (dunk, layup, finger roll,..etc)
MP: Minutes Played
FG%: Percentage of all shots (excluding free throws) that a player has made on the given season
3P%: Percentage of three-point shots a player has made on the given season
2P%: Percentage of two-point shots (inside the three-point line) a player has made on the given season
eFG%: A players effective field goal percentage which gives different weights to a players 3P% and 2P% as a three-point shot is intrinsically more difficult to make.
eFG% Formula: 2P% + 1.5*3P%
SHOT_ACTION: High level description of the shot (driving finger roll shot, tip dunk shot, etc..)

References:

1. A. Tan, “Predicting the result of an NBA shot: Machine learning analysis project,” *Medium*, 07-May-2021. [Online]. Available: https://ahtan-18882.medium.com/predicting-the-result-of-an-nba-shot-machine-learning-analysis-project-1989068b1c94. [Accessed: 14-Sep-2023]
2. Brown, Christophe. “Predicting the Probability of Scoring a Basket in the NBA Using Gradient Boosted Trees.” *Medium*, 11 Nov. 2021, towardsdatascience.com/predicting-the-probability-of-scoring-a-basket-in-the-nba-using-gradient-boosted-trees-c157390fb17. Accessed 17 Sept. 2023.
3. “Teams Defense: Stats.” Teams Defense | Stats | NBA.Com, www.nba.com/stats/teams/defense. Accessed 8 Dec. 2023.
4. DomSamangy. “DomSamangy/NBA_SHOTS_04_23: NBA Regular Season Shot Location Data from the 2003-04 Season to 2022-23 W/ Data Viz Example Code.” *GitHub*, github.com/DomSamangy/NBA_Shots_04_23. Accessed 8 Dec. 2023.
5. Avcontentteam. “Tree Based Algorithms: A Complete Tutorial from Scratch (in R & Python).” *Analytics Vidhya*, 2 Mar. 2023, www.analyticsvidhya.com/blog/2016/04/tree-based-algorithms-complete-tutorial-scratch-in-python/.
6. “NBA Player Stats - Dataset by Etocco.” *Data.World*, 26 May 2022, data.world/etocco/nba-player-stats.