NBA Shot Prediction

Ryan Le
UC San Diego
r9le@ucsd.edu

James Zeng
UC San Diego
jaz021@ucsd.edu

1. Introduction

The National Basketball Association, or NBA, is one of the most popular sports leagues today. With increasing popularity and competition, the NBA has started to incorporate technology into their game to propel them into the future. Teams like the Houston Rockets and the Milwaukee Bucks utilize player and team analytics to bolster their offense by limiting the amount two-point shots taken and maximizing high percentage three-point shots. Where teams have historically done very well through fundamental basketball strategies such as ball movement, cutting, and pacing, the Bucks and the Rockets used analytics to create modern offenses that spread the floor with quick attacks to the basket and strategically placed shooters around the three-point line [5][6]. Using player analytics and statistical modelling has greatly improved the performance of both teams.

This paper examines the application of machine learning algorithms to predict the success of shots attempted by NBA players. A basketball shot is very complex with countless variables affecting it. Because of this, it is fairly difficult to predict if a shot will score with high accuracy. Factors like balance, footwork or shooting form can greatly affect the outcome of a shot. However, by tackling this problem, we hope to determine the importance of particular features affecting a basketball shot, which can be useful for aspiring basketball players, coaches and organizations.

2. Datasets

The dataset we will be primarily using is a dataset created from the basketball shot logs from the 2014-15 NBA season available through Kaggle, which consists of 128069 shot entries with relevant data pertaining to each one [1]. Each row in this dataset represents a single shot.

Furthermore, we will be using a dataset of player statistics from the 2014-15 season to supplement our models using player data.

2.1 Shot Dataset

2.1.1 Data Features

The features in this dataset are as follows:

Feature	Description
game_id	The ID of the game
matchup	The date and teams involved in the game
location	Home or away game for the current player
win/lose	Outcome of game
final_margin	Difference in final score
shot_number	The nth shot a player has taken this game
period	The quarter of basketball a shot is taken
game_clock	The current time elapsed in the game
shot_clock	The current time of the shot clock at the shot. In the NBA, a shot clock is 24 seconds
dribbles	# of dribbles prior to shot
touch_time	# seconds player touches ball prior to shot
pts_type	Whether shot is a 2-point or 3-point shot

shot_result	String for make or miss
closest_defender	Name of closest defender
closest_defender_id	Id of closest defender
close_def_dist	Distance in feet of closest defender
fgm	Whether or not shot was made (0 or 1)
pts	# points earned from shot (0, 2, or 3)
player_name	Name of player who shot the ball
player_id	ID of player who shot the ball

2.1.2 Data cleanup

Before we explored the data to search for relevant features, we decided to clean up the data to remove redundancy. We found that there were 4 pairs of redundant features that essentially conveyed the same information: "game_id" and "matchup", "closest_defender" and "closest_defender_id", "shot_result" and "fgm", and "player_name" & "player_id". To remove redundancy, we decided to keep the features that had integer representation.

We then converted the features "location" and "win/lose" into binary integer features where 1 would represent a home game/a win and 0 would represent an away game/a loss. Additionally, we decided to merge the "period" and "game_clock" into a "game_time" By starting our new feature at 0 and translating our "game_clock" and "period" to calculate an elapsed game time, we can represent all quarters, as well as overtime values.

2.1.3 Errors/Missing Data

In the dataset, we noticed that there were some errors in the dataset regarding the "touch_time" and some other features. There were various instances of "touch_time" where the data was negative, which shouldn't be possible. Additionally some features had missing data points. Whenever we encountered these invalid data points, we changed their values to zero.

2.1.4 Data Exploration

2.1.4.1 Data Assumptions

When exploring the data, we came in with various assumptions to help guide our feature design.

From previous research, we found that, typically, the greatest factors to shot accuracy is the player himself. Therefore, features such as "game_id", "matchup", "location", etc. would probably not have much effect on the outcome of the shot, so we did not examine these features in depth. Therefore, the main challenge of creating our model is figuring out which of these features would be the greatest predictor for a made/missed shot.

Firstly, two features we believed could have strong correlations to shot accuracy are "dribbles" and "touch_time". A high number of dribbles might indicate that the player is well guarded to the extent that they have a difficult time to make a shot. In turn, a high touch time could indicate a high number of dribbles or holding the ball for a long time.

Other features that could have strong positive correlations to shot accuracy are "shot_clock", "shot_dist", and "final_margin".

Typically, if a team's offense is going well, shots attempted early in the shot clock should have a higher chance of scoring because of a lesser sense of urgency that can lead to a relaxed shot.

The distance of the shot could also affect a shot's probability of scoring as well because shots generated closer to the basket should be easier/

Furthermore, the final margin of the game could also be a good indicator of a made shot as the greater the win, the higher the likelihood that a shot favoring them went in.

With these assumptions, we explored the relationships between these features (and other potentially relevant features) and the percentage of shots made for each value of that respective feature.

2.1.4.2 Exploration Results

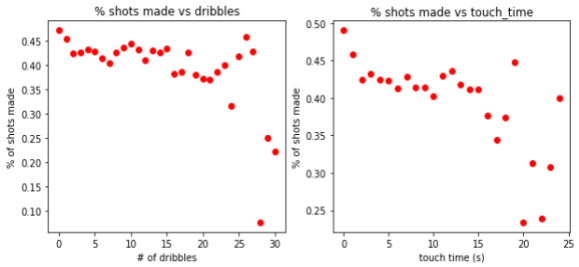

Figure 1

From the figures above, we can see a correlation between dribbles and touch time, shown by the downward trends in the two graphs. With the results of these graphs, our previous assumptions made about these features seem to be valid.

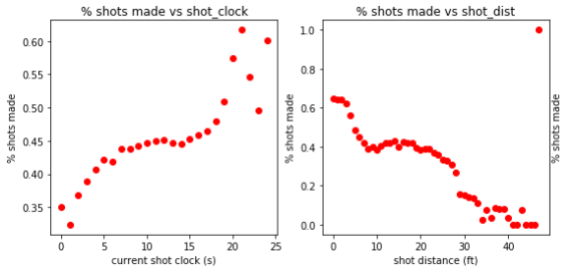

Figure 2

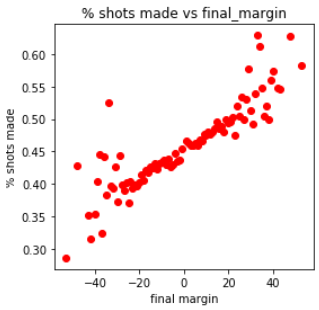

Figure 3

The graphs from Figure 2 and Figure 3 represent the features that we found the strongest correlations with percentage of shots made. Looking at the strong positive correlation from the left graph in Figure 2 and the graph from Figure 3, our assumptions made in the previous section appear to be valid. Likewise, looking at the correlation in the right graph in Figure 2, our assumptions about shot distance also appear to be correct.

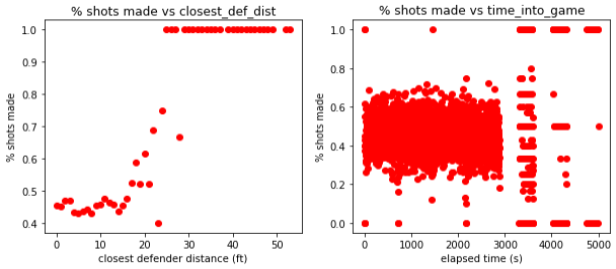

Figure 4

Two features that we thought could have been useful include the closest defender distance and the time into the game, as seen in Figure 4. However, we can see that the graph on the left contains a lot of outliers which could hinder the performance of our model, while the graph on the left does not have a clear correlation that would implore us to select it as a feature.

2.2 Player Dataset

One potential problem we faced is that we have no individual information on the players themselves. Using just the shot dataset, an NBA superstar has the same chance of making a shot as an average NBA player. This should not be the case. In order to try and have our model predict this as well, we created a player dataset from the 2014-2015 NBA season using basketball-reference.com [2].

2.2.1 Data Features

feature	Description
player	The player's name
pos	The player's position
fg	The player's field goal %

3p	The player's 3-pointer %
2p	The player's 2-pointer %
efg	The player's effective field goal %
ppg	The player's points per game

The feature we felt was most important is effective field goal percentage (eFG%). This feature takes into account the value of the 3-point shot, so it is a more accurate indicator of accuracy than using the other accuracy measures.

3. Predictive Task

Using this dataset, our task was to implement a model that can predict whether or not a shot is made based on the relevant features we have determined. Since we are predicting a binary outcome, we can measure our performance by calculating the accuracy of our results. We will attempt to use Logistic Regression, Support Vector Machines and Random Forests to implement our models. We will also create a simple baseline model to compare its results to the results of our other models.

4. Models

4.1 Baseline Model

Before trying out more complex models, we decided to create a baseline model that represents a simple predictor that should obtain a fair accuracy. A basketball shot has two outcomes: made or missed. Since this is merely a binary outcome, our baseline model just randomly outputs 1 or 0 for each feature given. Running the baseline on our test dataset provided accuracies around 50% as expected. To measure the quality of our models in the future, we will compare their performances to this simple, unsupervised baseline.

4.2 Logistic Regression

Since we are producing a binary outcome with our predictor, using logistic regression was a natural first start because of its ability predict binary outcomes using probabilities. Since we saw some features had some clear trends for what makes a shot go in, we figured logistic regression would be able to correctly classify most of the easy examples.

We implemented our logistic regression model using sklearn.linear_model's logistic regression function. We then decided to create two models of logistic regression to test their respective performances:

L1 regularization (popularly known as "Lasso Regression") represented with this equation:

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

L2 regularization (popularly known as "Ridge Regression") represented with this equation:

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

We tested various regularization constants (C-value for the library) from 0.01 to 100 and found that a constant value of 0.01 provided the best accuracy for both methods.

4.3 Support Vector Machine

Another model we decided to use is a support vector machine using sklearn's LinearSVC function. We predicted that using an SVM might produce a higher accuracy than logistic regression because of they are not affected by outliers like logistic regression might be. This could be useful for some features in our feature design as features such as "shot_dist" and "final_margin" could have many outliers from "Hail Mary" shots at the end of quarters and blowout games. However, one tradeoff of using a SVM is the training time is often much longer compared to using logistic regression.

4.4 Random Forest

Random forest is an ensemble learning method that creates multiple decision trees using random

subsets of features and uses the outcome of all trees to make a final prediction. In our case, we are using random forest classification, from sklearn.ensemble's library, so the final prediction is the mode of all tree predictions. Random forests might be more advantageous than logistic regression and support vector machines because each tree is trained on a random subset of the training set, so it would be robust to overfitting.

To optimize our random forest algorithm, we had to tweak the parameters of the forest, mainly: 'n_estimators' (number of trees in the forest) and 'max_depth' (the maximum depth of each tree). After several different combinations, we found the best accuracy using a forest of 100 trees with a maximum depth of 20.

5. Experiments and Results

When conducting the experiments, every model was trained on a 95% random subset of the data and tested on the remaining 5%. We decided to test our models using four different feature designs:

- F1 = ['shot_clock', 'shot_dist', 'touch_time', 'final_margin']
- F2 = ['shot_clock', 'shot_dist', 'touch_time', 'final_margin', 'efg']
- F3 = ['shot_clock', 'shot_dist', 'touch_time', 'final_margin', 'efg_z_overall']
- F4 = ['shot_clock', 'shot_dist', 'touch_time', 'final_margin', 'efg_z_position']

5.1 F1: No Player Data

Before we incorporated the use of player data in our model, we wanted to see how accurate our predictor would be with using just the top features of our shot data. The features we used for this experiment are { 'shot_clock', 'shot_dist', 'touch_time' and 'final_margin' }, as they had the strongest correlations with shot accuracy. Testing this feature design with each of our models, our results can be seen in Figure 5.

	Training Accuracy	Test Accuracy
Log Reg L1	0.604	0.605
Log Reg L2	0.604	0.605
SVM	0.604	0.606
RF	0.637	0.600

Figure 5

From these results, we can see that all methods performed similarly in terms of accuracy, with the SVM coming out on top with the test set. However, one stark difference we do notice is the training accuracy of the random forest is greater than that of all other models.

5.2 F2: Using eFG%

While the previous results with only using the shot data performed considerably better than the baseline, we felt that incorporating player data into our model would provide good insight into whether a shot will be made as every player has a different skill level. Since there are two different types of shots in basketball (the 2-point shot and the 3-point shot) and every player has different shot tendencies (some might prefer only 3-pointers etc.), we decided that the best metric to use to rate a player's shooting efficiency is effective field goal percentage (eFG%). eFG% is calculated by the following equation [4]:

$$\frac{(field\ goals\ made + 0.5 * 3\ point\ field\ goals\ made)}{field\ goals\ attempted}$$

This metric allowed us to take into consideration the added value of the three point shot. A player shooting a 3-point shot could shoot at a lower percentage than a player who primarily attempts 2-point shots, but since the 3-point shot provides more points, that extra point should be taken into

consideration. When running our model to include eFG%, we achieved the results shown by Figure 6.

	Training Accuracy	Test Accuracy
Log Reg L1	0.604	0.605
Log Reg L2	0.604	0.606
SVM	0.604	0.607
RF	0.638	0.615

Figure 6

Using this new feature design, we do not see much improvement in the logistic regression models and the support vector machine, while the random forest saw a significant 1.5% increase.

5.3 F3: Using eFG% z-score

The results of using eFG% alone might have been due to the fact that there is only a small range of values for eFG% for all players. To add more weight to eFG%'s that are greater than league average, we used the z-score the eFG%, calculated with the following equation:

$$\frac{eFG\% - \ mean\ of\ all\ eFG\%}{standard\ devation\ of\ all\ eFG\%}$$

The results with this feature for this feature design is seen in Figure 7.

	Training Accuracy	Test Accuracy
Log Reg L1	0.604	0.612
Log Reg L2	0.604	0.612
SVM	0.604	0.612
RF	0.639	0.615

Figure 7

Again, we see incremental improvement with both regressions and the SVM increasing about 0.5% and the Random Forest performing about the same as the previous feature design.

5.4 F4: Using eFG% z-score per Position

While using the z-score of eFG% improved our accuracies, we wanted to see if using the z-score of the player relative to his position would improve the performance of our model. In basketball, a player's position greatly affects the role the player has in the offense. For example, a player with the center position generally shoots the ball when close to the basket which could result in a higher eFG% in comparison to a shooting guard who would take more outside shots.

With this feature design, our results can be seen in Figure 8.

	Training Accuracy	Test Accuracy
Log Reg L1	0.604	0.612
Log Reg L2	0.604	0.612
SVM	0.604	0.612
RF	0.638	0.613

Figure 8

Using this feature design very marginally increased the performances of the first three models by ten-thousandths of a percent, while also decreasing the performance of the Random Forest model.

6. Relevant Work

For relevant work on this problem, many decided to use data solely on data around the moment of the shot and not data about the player as we have. Common techniques used include logistic regression and random forests as we have used and convolutional neural networks in other works.

In "Predicting NBA Shots", Meehan utilizes the same dataset as we have and uses many methods

such as logistic regression, SVM, naive bayes, random forests, and boosting [9]. Tuning each model with optimal parameters that he found, he was able to achieve a maximum accuracy of 68%. He concludes that random forests and boosting are the most effective in predicting a shot due to the limited features in this particular dataset. However, it is also important to note that Meehan was also able to achieve similar accuracies to our models when only using the features from the shot dataset with optimized parameters in his models.

Other work that tackled the problem of NBA shot prediction utilized the NBA's SportVu tracking camera data which records the position of the players, ball and referee 25 times per second [3]. One work, "NBA Shot Prediction and Analysis" by Cen, Chase, Pena-Lobel and Silberwasser utilizes a dataset from the 2013-14 NBA season that includes features like a player's position on the court during the shot as well as other similar features from our dataset, like closest defender distance and touch time [7]. Using this data, they were able to achieve a 65% accuracy.

Using the same dataset, Harmon, Lucey and Klabjan create image representations of the position of the ball, and players on offense and defense in the 5 seconds prior to a shot to predict the outcome of a shot [6]. Utilizing these image representations, they were able to factor in opposing defensive players in their predictions. Using a Convolutional neural network along with a feed forward network, they were able to achieve a 61.5% accuracy.

7. Conclusion

Ultimately, after running three feature sets on four different models, we were able to produce results hovering around 60-61% test set accuracy. While results on the test set were similar amongst all 4 models, the Random Forests model performed the best on all feature sets that included player data. Additionally, the feature set that simply used the value of a player's eFG% performed best when compared to the feature set that used the z-scores of players' eFG%'s. This invalidated our assumption that the relative value of a player's eFG% to his peers is a better predictor of a shot going in. This could be because a player's performance compared to his peers might not be the most important factor when it comes to the outcome of a single shot.

However, while including the player data improved the performances of each model, there was only about a 1% improvement across the board. This demonstrates that, while player data might be slightly helpful, the various factors in the moment of the shot are probably more predictive of whether or not the shot would go in, as seen in related works. Future work to improve the performance of our model could include data from the exact moment of the shot, such as data used from SprotVu's tracking camera as used in models from the previously cited literature.

8. References

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

[2] Basketball-Reference 2014-15 Player Stats. https://www.basketball-reference.com/leagues/NBA_2015_per_game.html, 2015.

[3] https://en.wikipedia.org/wiki/SportVU

[4] https://www.basketball-reference.com/about/glossary

[5] Houston Rockets Offense Explained. https://www.theringer.com/nba/2018/2/13/17005988/houston-rockets-offense-explained, 2018.

[6] Milwaukee Bucks Highlight Offense. https://www.sbnation.com/nba/2018/10/18/17992662/milwaukee-bucks-highlights-offense, 2018.

[7] R. Cen, H. Chase, C. Pena-Lobel, D. Silberwasser. NBA Shot Prediction and Analysis, 2017.

[8] M. Harmon, P. Lucey, and D. Klabjan. Predicting shot making in basketball learnt

from adversarial multiagent trajectories, 2016
[9] B. Meehan. Predicting NBA Shots, 2017.