Analysis of NBA shot logs from the 2014-15 season

Shreyas Rajesh
A53324553
s1rajesh@ucsd.edu
University of California, San Diego
San Diego, CA

Ashish Gupta
A53326010
asg007@ucsd.edu
University of California, San Diego
San Diego, CA

1 INTRODUCTION

In this paper, we study and analyse basketball data for the NBA season 2014-15. This paper debunks various theories like the hot-hand hypothesis, where a player shot accuracy increases if he has made many successful attempts in the same game. This paper also sheds light on the clutch gene hypothesis which addresses the myth that a player performs significantly better than others in end of game situations in terms of efficiency and the points scored. This paper also proposes multiple predictive modelling techniques to predict if a shot would be 'made' or 'missed' given a certain set of parameters. To achieve this, we use a combination of two datasets, one of which includes every single shot log in the season with features like the time the shot was made, result of the shot, player who made shot, the closest defender, the distance from closest defender, the shot number and other game specific details. The other dataset depicts player specific information including the height, weight, age and other details for all players playing in the same season. A complete dataset was obtained by an inner join on these datasets. Using this complete dataset, we build a logistic regression model and a neural network model in Keras, assessing the usage of the features involved in each model and discussing their results. We also discuss a recommendation model using latent factor algorithm to suggest the best defender for a particular player making the shot. As part of our factual test of the hot-hand hypothesis, we find proof that the myth is in fact true and we visualise our findings to interpret the result in a coherent manner. Overall, this paper provides a comprehensive analysis into basketball data for better insights and understanding of the game.

2 EXPLORATORY ANALYSIS

This section briefly discusses the properties of the data and interesting finding we observed on this particular dataset. The dataset we used was procured courtesy of kaggle user Dansbecker. His extensive dataset has helped us perform a detailed exploratory analysis on NBA shot logs to understand the shooting of NBA players better. This dataset contains a total of 128069 shot attempts taken during the 2014-15 NBA season, along with 20 features related to these shot attempts, including but not limited to, location, closest defender, shot number, final margin, game clock, shot clock, dribbles, touch time and shot result. To support and be able to provide valuable insights into this dataset we added valuable features to it using another dataset collected by kaggle user Omri Goldstein. This dataset contains scraped data from Basketball Reference and contains all details corresponding to each player in the 2014-15 including but not limited to season statistics, like minutes played, field goals made, field goals attempted as well as personal statistics of the player like age, weight, height etc. Our eventual dataset therefore contained shot related information, as well as overall season statistics and personal statistics corresponding to the player and his closest defender on every shot. All combined our dataset contained 30 features corresponding to every shot attempt and 128069 shot attempts in all. We further cleaned the datasets and replaced NaN values and blanks with appropriate values for ease of use throughout this project. A couple of important limitations of the dataset however include the fact that, despite including a large number of shot attempts this dataset does not cover all games across the entire season and also includes only 281 players of the total 492 active players that season[3] and includes only field goals(no free throws).

2.1 Dataset Overview

Here we first discuss some basic statistics and properties of the dataset. The first immediate question that comes to mind when

```
'data.frame':   128069 obs. of  31 variables:
 $ GAME_ID                 : int  21400899 21400899 21400899 21400899 21400899 21400899 21400899 21400899 21400899 21400890 ...
 $ MATCHUP                 : Factor w/ 1808 levels "DEC 01, 2014 - DEN @ UTA",..: 1291 1291 1291 1291 1291 1291 1291 1291 1291 1277 ...
 $ LOCATION                : Factor w/ 2 levels "A","H": 1 1 1 1 1 1 1 1 1 2 ...
 $ W                       : Factor w/ 2 levels "L","W": 2 2 2 2 2 2 2 2 2 2 ...
 $ FINAL_MARGIN            : int  24 24 24 24 24 24 24 24 24 1 ...
 $ SHOT_NUMBER             : int  1 2 3 4 5 6 7 8 9 1 ...
 $ PERIOD                  : int  1 1 1 2 2 2 4 4 4 2 ...
 $ GAME_CLOCK              : Factor w/ 719 levels "0:00","0:01",..: 70 15 1 228 155 615 136 600 434 213 ...
 $ SHOT_CLOCK              : num  10.8 3.4 NA 10.3 10.9 9.1 14.5 3.4 12.4 17.4 ...
 $ DRIBBLES                : int  2 0 3 2 2 2 11 3 0 0 ...
 $ TOUCH_TIME              : num  1.9 0.8 2.7 1.9 2.7 4.4 9 2.5 0.8 1.1 ...
 $ SHOT_DIST               : num  7.7 28.2 10.1 17.2 3.7 18.4 20.7 3.5 24.6 22.4 ...
 $ PTS_TYPE                : int  2 3 2 2 2 2 2 3 3 ...
 $ SHOT_RESULT             : Factor w/ 2 levels "made","missed": 1 2 2 2 2 2 2 1 2 2 ...
 $ CLOSEST_DEFENDER        : Factor w/ 474 levels "Acy, Quincy",..: 15 51 51 62 472 457 219 352 314 132 ...
 $ CLOSEST_DEFENDER_PLAYER_ID: int  101187 202711 202711 203900 201152 101114 101127 203486 202721 201961 ...
 $ CLOSE_DEF_DIST          : num  1.3 6.1 0.9 3.4 1.1 2.6 6.1 2.1 7.3 19.8 ...
 $ FGM                     : int  1 0 0 0 0 0 0 1 0 0 ...
 $ PTS                     : int  2 0 0 0 0 0 0 2 0 0 ...
 $ player_name             : Factor w/ 281 levels "aaron brooks",..: 36 36 36 36 36 36 36 36 36 36 ...
 $ player_id               : int  203148 203148 203148 203148 203148 203148 203148 203148 203148 203148 ...
 $ Defender_Name           : Factor w/ 474 levels "  nene","a.j price",..: 9 47 47 313 431 111 200 318 100 463 ...
 $ Player_Height           : num  182 182 182 182 182 ...
 $ Defender_Height         : num  195 200 200 188 0 ...
 $ Height_Diff             : num  -12.5 -17.5 -17.5 -5 182.5 ...
 $ Player_Weight           : num  77.8 77.8 77.8 77.8 77.8 ...
 $ Defender_Weight         : num  99 97.2 97.2 85.5 0 ...
 $ Player_Age              : int  30 30 30 30 30 30 30 30 30 30 ...
 $ Defender_Age            : int  33 26 26 23 0 0 32 25 24 28 ...
 $ Player_Pos              : Factor w/ 6 levels "0","C","PF","PG",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ Defender_Pos            : Factor w/ 6 levels "0","C","PF","PG",..: 6 5 5 6 1 1 4 2 4 6 ...
```

Figure 1: Features of the dataset

we think about a shot log dataset, is which players are attempting the most shots, and what the variation is across the players in the dataset. If the dataset is sufficiently large we would expect to see a normal distribution and is roughly what we observe despite the dataset being small. However, we see that the distribution is right skewed which means that a few players in the dataset took way more shots compared to the rest of the league. Also, as expected James Harden(a high volume shooter) had the highest field goal attempts at 1044. The league average is shown by the red line(456 attempts), which turns out to around 5.5 shot attempts per game. The above analysis was performed completely using the original dataset.

Another interesting exploration of the dataset was to look at the plots of player age against the shot distance. It is a common notion in the NBA that as players get older they begin to shot

Figure 2: Histogram of shot attempts per player

a lot more from the outside(i.e. farther from the basket) as they don't have the athletic ability to finish in traffic and explode to the rim on a regular basis. A classic example of a player that follows this pattern is the famous Kobe Bryant. We believe that this is a hypothesis worth testing. We initially observed no major correlation however, it is also important to note that this applies a lot more to smaller and quicker guards as compared to larger post players. Since post players, would infact possibly move closer to the rim(e.g. to catch lobs), to reduce movement and strength required to score on other defenders of their size. So we repeated this task with only point guards and shooting guards(perimeter players) and the plot in figure 3 provides the results and confirms the hypothesis partially. Although, this plot conforms with our expectation our

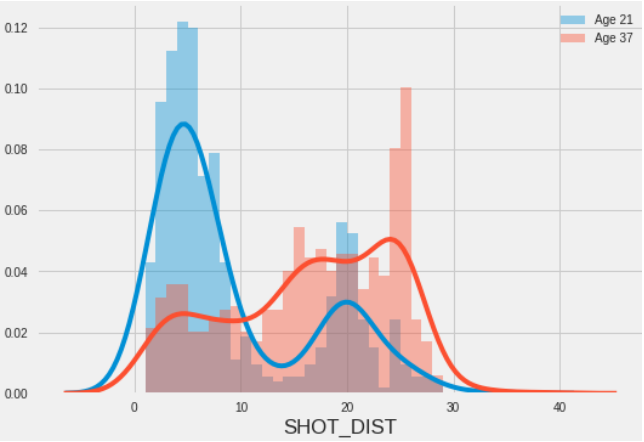

Figure 3: Guards shooting distance distribution

sample size(especially after considering fixed ages and only 2 out of 5 positions) is too small to say that this generalises to the entire NBA and across multiple seasons.

2.2 Klay's Hot Hand

There has been a constant debate about the hot hand in sports especially basketball that has a large number of rallies per game increasing the scope for players to get hot. A hot hand basically refers to the ability of a player to shoot the ball better as they make more shots consecutively. This theory has been largely contradicted throughout the history of the sport right from the 1980s[10] until as recent as 2014[2] so much so that it was called the hot hand fallacy[4] yet people continue to work on this problem. However, in 2018[5] there was a study that came out supporting this hypothesis arguing that all earlier methods of measurement were flawed in the same basic way due to the law of small numbers. And further, as fans of the Golden State Warriors, Klay Thompson is a classic example of a player that seems to catch fire when he shoots. He is one of the few players that goes on unbelievable hot streaks, making even skeptics believe in its existence. This extremely interesting debate, made us want to find out for ourselves. We used the shot log data to determine whether the shooting percentages of a player increases as their number of makes increase. We filter and select players that have atleast 500 shot attempts and atleast 1 instance of 5 makes in a row. We noticed as shown in figure 4 that as the player makes more shots their shooting percentage can possibly go up, thus essentially giving them a "hot hand" which describes the phenomenon of better shooting as they make more shots consecutively. Further as

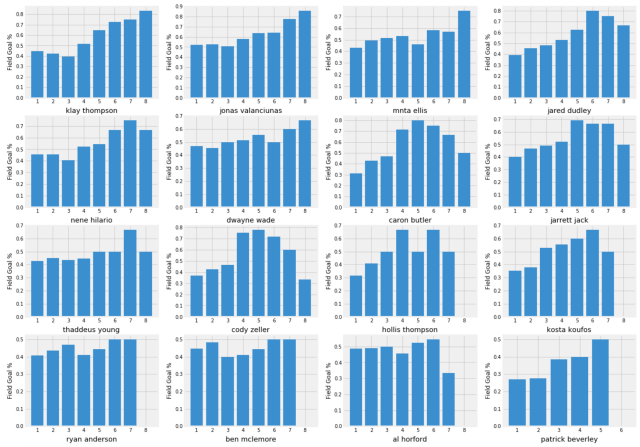

Figure 4: Shooting percentage against number of consecutive makes(for players who have atleast 500 attempts, 1 instance of 5 consecutive makes and have an overall upward trend)

expected this is most obviously the case with Klay Thompson, who's shooting percentage rises to a whopping 80% after 7 straight makes. We also noticed however that this is a rare phenomenon expressed only in 16 cases out of the 281 players in our dataset and this analysis isn't perfect since we have a fairly small sample space. Players have very few occasions in their career that they score more than 6 consecutive baskets(even high volume high percentage shooters) let alone one season in this dataset. For example, James Harden to had the most attempts in the 2014-15 season never made more than 4 consecutive shots. Thus, this argument is not bulletproof

but rather an interesting finding within the confines of this dataset and that much more of an accomplishment for players that have achieved this feat.

2.3 Kobe's Clutch Gene

Another highly debated and interesting discussion is about some players being better clutch players than others(i.e. their shot percentage is higher than normal in late game situations when the game is on the line). Generally, having to shoot in a situation that determines whether the team wins or loses the game is one of extreme stress and anxiety thus reducing the performance of the player. However, many argue that some players, step up in such occasions and improve their performance carrying their team to victory more often than not. The most commonly referenced player to do this is Kobe Bryant[1]. Watching his games, certainly does make it feel that way, however we wanted to test it ourselves to know for sure using this shot data. We generated a scatter plot of the performance(points scored) in the last two minutes of games that were eventually decided by less than 8 points against the number of such shots attempted. This was our choice of definition for "clutch" situations. We then fit a line to the data to determine the league average for this dataset as shown in red in figure 5. To our

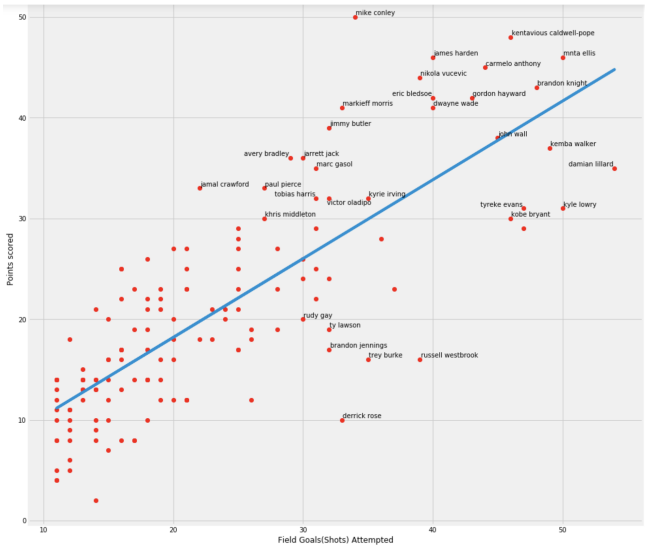

Figure 5: Scatter plot of points scored against shots attempted by each player in "clutch" situations and the league average(red line) determined by linear fit

surprise Kobe Bryant did not exist among the top players at all. In fact, he has a worse shooting percentage and significantly lower than the league average in points scored for the number of attempts he has. We believe there are a couple of reasons for this. Firstly, Kobe Bryant was nowhere near his prime years in 2014-15, he was very close to retirement and hence his athleticism had deproved meaning more fatigue in late game situations. Secondly, historically too, other literature shows that Kobe's clutch performances are wrongly attributed to his mentality when it is a product sheerly of the number of such situations and such attempts he has taken which as we observe even in his waning years is quite high compared to the rest of the league. Hence, he is below league average on points scored, however, has a very high volume that creates an impression that he is a very clutch player by the few times he performed extraordinarily. However, as expected we note the most efficient players in late game situations were Mike Conley and James Harden who had high volume and scored more than league average, and conversly Damian Lillard who had the highest attempts in such situations, fell below league average in points scored. Derrick Rose too, recovering from injury although still attributed with his MVP season had more attempts than half of the league but shot extremely poorly and therefore the typical "inverse clutch" player and someone who succumbs to the pressure of the moment(atleast in this data and 2014-15 season). This analysis bodes well with what we know from watching the NBA and these players in general are the ones shooting in late game situations in the 2014-15 NBA season.

3 TASK SELECTION AND FEATURE ENGINEERING

3.1 Lights out shooting

With a couple of interesting findings in the data, we debated the best tasks to perform on the dataset. We noticed that the most obvious task to accomplish on this data is simply a shot prediction task(i.e. will the player make or miss the shot given all the information about the player and situation). Given that the dataset is small and sports in general being extremely unpredictable and dependent on too many factors(large number of them are unquantifiable, e.g player's mood, team morale etc) apart from the features under consideration. However, acknowledging this, we still believe that this is a task that can be accomplished with a certain degree of accuracy and therefore put this to test. This can be really useful to NBA teams to determine who their best scorers are and who should be attempting the most shots in each situation. We worked on using all the features available to us to predict the shooting percentage of a player in that situation. This is identical to performing a binary classification task and using the score associated with the make as the shooting percentage in that situation. We further use these classifiers to determine which features are most highly correllated to the shot prediction determining what are the most important factors in making/missing a shot in any situation. A simple accuracy measurement of the classifier along with a confusion matrix for the predictions will provide us with a measure of the quality of our model and hence we use these metrics to evaluate our model. Looking at the correlation between the features we can also tell whether it matches our intuition of what we believe to be important factors in making or missing a shot.

However, to perform this task, we need to first engineer all the features available to us to optimise the performance of the classifier on this dataset. The table below describes each column of the dataset, the corresponding value generated to feed as input(engineered feature) to the classifier and the dimension of this feature for each datapoint. We must first note a couple of points, we have not chosen all the columns. We have dropped a few columns that we believed will have no correlation on the output and yet make our model significantly more complex(e.g Matchup) or are redundant features(Field Goal%). Secondly, the features mentioned

Feature	Technique	Dimension
Location	Binary	1
Game result(W)	Binary	1
Final Margin	Int	1
Shot number	Int	1
Period	One hot encoding	4
Game Clock	Float(in seconds)	1
Shot Clock	Float	1
# of Dribbles	Int	1
Touch Time	Float	1
Shot Distance	Float	1
Closest Distance Defender	Float	1
Points Type**	One hot encoding	2
Field Goals Made	Int	1
Points(Total in game)	Int	1
Height Difference**	Float	1
Weight Difference**	Float	1
Player Age	Int	1
Defender Age	Int	1
Player Position	One hot encoding	5
Defender Position	One hot encoding	5
Player Name**	One hot encoding	281
Defender Name**	One hot encoding	487
Shot Result(Label)	Binary	1

Table 1: This table contains all the features along with how they were engineering before training the various models along with their dimensions.(These features were used only in some models/attempts as discussed later)**

in the table are the features that we chose after trial and error with other variations and realised these to perform the best on the dataset under consideration. From our previous exploratory analysis of the dataset as well as our understanding of the game, despite having created and attempted training the model with all the features in the dataset, we expected a few features to certainly help more than others. These features, were mainly shot related features, like shot distance, touch time, number of dribbles etc. and hence kept these features in all models and attempts with this dataset. We have discussed the relation between the features and the peformance of the model using the regression coefficients in a later part of this paper.

3.2 Lockdown defense

Since, we have information of every shot and their corresponding closest defender on each shot, another interesting task to perform on this dataset, was a recommendation task that involves recommending the best defender for every offensive player. This is a very interesting and valuable prediction as it has large scope in helping teams identifying the best matchups for players from previous shot data. However, this is a very small sample space, containing only 281 offensive players and very few cases of the same defender guarding them repeatedly given the plethora of situations in games. However, we still put this task to test to see if we are able to generate any correlation in the data without very high expectations. Since we don't have any test data for the task, we use the validation split to compare the best predicted defenders from our training set to the best defenders based on actual performance from the validation data. This isn't ideal since the validation could potentially contain players and/or defenders that were never seen in the training data, however we have tried to avoid this by grouping the players and then shuffling the data to ensure some representation of each player in the training data. Since, this was the best possible method within the means of the dataset we progress with this.

The feature engineering for this task takes slightly more effort to encode and we used the shot logs to generate offensive player, defensive player pairs and a corresponding rating that is esentially the shot efficiency(field goal percentage) of all shots taken by that offensive player while being guarded by that defensive player. We use this pairwise information of shot efficiency as our metric to determine the quality of defenders. We compute the errors using the mean squared error(MSE) of the predicted shot efficiency compared to the actual shot percentage on the validation data. The mean squared error is given by,

$$MSE = \frac{1}{N}(predicted\ shot\% - actual\ shot\%)^2 \quad (1)$$

where N is the number of datapoints under consideration and the actual shot efficiency is the label of the datapoint. Finally, we analyse these eventual recommendations of defenders from our model by comparing them with the labels and our prior knowledge of the NBA 2014-15 season.

4 SHOT PREDICTION MODELS

In this section we deal with various methods for shot predictions(i.e will the shot be made or missed) on this dataset. As discussed in the previous section, we are performing a binary classification task to analyse shot prediction. First off, the first traditional model to use for a binary classification task is naturally logisitic regression. So we attempted logistic regression as done by others [1] performing similar analysis on this type of data. We compared various features, attempted various combinations of features and methods of encoding the features. Here we discuss only the best ones with mention of the other attempts and potential reasons of failure. We also, use the results of the logistic regression to analyse the regression coefficients to determine the most important features.

We further use a more powerful Multi-Layer Perceptron Neural Network model, to see if the performance improves on the dataset and to obtain a better understanding of whether the limitations in accuracy are due to the complexity of the model or due to the dataset. We discuss our various attempts with the deep learning model as well and finally discuss the results of both models. Since this is a classification task we can't consider a lot of the other models we have learnt in this class, except maybe SVM. However, we expect SVM to perform very similar to logistic regression however, SVM takes much longer to fit and struggles especially in high dimensional features and due to lack of resources and time to properfly implement, tune hyperparameters and test we restrained from including the SVM model.

4.1 Logistic Regression Model

For the Logistic regression model we consider all possible features initially and then used trial and error to determine the best features

using the regression coefficients, and model accuracy. If the model accuracy improved it was simple to definitely keep those features. Further features whose regression coefficients were close to 0 when compared to the rest of the features were also discarded or reused in a different feature encoding technique. Eventually we settled for a feature vector of 22 dimensions to obtain the model with best performance on this dataset. We strongly believe these are not essentially the only features that "matter", but rather the best correlated features in this dataset, thus helping the model the most. The other features, either negatively affect the model accuracy or do not affect the model accuracy at all and have very small regression coefficients. Finally, we also performed a grid search and determined the best value for the regularization factor(C) to optimize the performance of the model. The table below shows the regression coefficients corresponding to the columns chosen in the model with highest accuracy. The starred columns are one hot encoded and hence contain more than one regression coefficient however, for the sake of this plot to show their importance relative to other features we have considered the absolute average of their coefficients. This clearly shows us that certain features, are highly

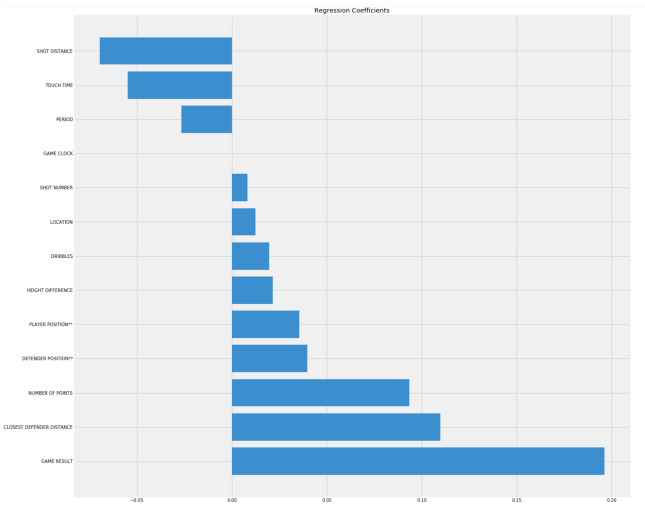

Figure 6: Feature importance to shot classification

influential of the outcome of the shot and some less so. We also see that some features are negatively correlated to the prediction of the shot. However, all these results match intuitive understanding of the sport and our expectation. For example, as a player shoots farther away from the basket(SHOT_DIST) the likelihood of making the basket reduces and as the game goes on the shot accuracy drops due to fatigue as well as the pressure of late game scenarios. However, it is slightly counter-intuitive and quite interesting to note that the touch time(time the ball is in players hand before shot) negatively affects the outcome of the shot. One would expect having longer to take time and shoot the ball would result in a more accurate shot, however, one could come up with reasons to justify why it is otherwise. We believe, that this probably happens because that players who shoot the ball immediately without holding onto it for too long are probably already open and hence no defender close by thus improving increasing accuracy and having a longer touch time, probably means the defender has time to contest the shot as well as the possibility of the player being stuck without options and hence holding the ball for longer before throwing it up in a more desperate attempt thus reducing accuracy.

The most important features are expected, are the game result(which means that if the team won each player probably shot better which is why they won), how close their closest defender was, and the number of points they had in the game(which means they had a better individual game). Finally, one might expect the game clock to have some effect on the shot outcome but as we observe it has extremely low influence on the shot outcome, telling us that the time in the quarter of the game doesn't matter as much as which quarter it is itself. We must also note that game result is a feature that cannot be obtained for shot predicted at the time of the attempt and thus might not be the best feature to choose, in case of a realtime analysis or deployment in the real world. We also expected the one hot encoded player name and/or the one hot encoded defender name to have a major contribution to the accuracy of the data, however, we observed that this not only drastically increases the complexity of the model(281 new dimensions in the feature vector) but also provides no improvement in accuracy. This could be due to lack of sufficient data or very minor correlation between the players and shots made in this specific dataset and the same explanation goes for the defender names as well. We discuss the accuracy and the conclusions from our results in the last section of the paper.

4.2 Multi-layer Perceptron Model

In this section, we use Keras in python to build a sequential predictive model using a dense neural network. Neural networks generally perform better when the real-valued input and output variables are to be scaled to a fixed range. For this problem, each of the input variables and the target variable have a Gaussian distribution; therefore, standardizing the data in this case is desirable. We chose to use a Multi-layer Perceptron model to perform this classification task. We performed the logistic regression and the deep learning task independently to not bias our features based on the performance in the other case. We chose various features to determine which ones provide the best model performance. However, we observed as expected that most of the features that work for the logistic regression are the ones that work in this case as well. Numeric features like shots made, points, height diff, shot distance and closest def distance are fit and transformed using the MinMaxScaler() function in python. This helps ensure that all the features is in the same scale for better model training. Some of the other features like period and player position are one hot encoded using the inbuilt functions in Keras. This is done since the values in these vectors have a small range, discrete values and after trial and error we observe the model performs better with these one hot encoded features.

The model we chose has one hidden layer with 16 nodes and uses the rectified linear activation function (ReLU). The output layer has 1 node, given the one real-value to be predicted and based on this output it is easily classified. In a neural network, the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for

that input. The rectified linear activation function is a piecewise linear function that will output the input directly if is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. In order to use stochastic gradient descent with backpropagation of errors to train deep neural networks, an activation function is needed that looks and acts like a linear function, but is, in fact, a nonlinear function allowing complex relationships in the data to be learned. The function must also provide more sensitivity to the activation sum input and avoid easy saturation. The rectified linear activation function is a simple calculation that returns the value provided as input directly, or the value 0.0 if the input is 0.0 or less. We can describe this function g() mathematically using the max() function over the set of 0.0 and the input z; for example: g(z) = max{0, z}

The function is linear for values greater than zero, meaning it has a lot of the desirable properties of a linear activation function when training a neural network using backpropagation. Yet, it is a nonlinear function as negative values are always output as zero.

We compile the model using a standard 'adam' optimiser performing stochastic gradient descent and testing the loss by 'binary crossentropy'. This is also called the log loss in classification as described in the equation below in our case.

$$Loss = -\frac{1}{N}\sum_{i=0}^{1} y_i log(\sigma(\hat{y}_i)) + (1 - y_i) log(1 - \sigma(\hat{y}_i)) \qquad (2)$$

where y is the true label, $\hat{y}$ is the prediction corresponding to that datapoint, and σ refers to the sigmoid function.

We split the dataset of 128069 records into training and test set where the training data using an approximately 90-10 split. We train the model using the train data predictors and the response variables repeating this for 100 epochs and setting 20% of the training data aside as the validation set. Given the stochastic nature of the training algorithm, specific results may vary on the test data. Here we generated two models, one with identical features to our logistic regression case, and one with modified features that contained mostly similar features however a couple of extra features and more one hot encoding of the features. In the first case(Model A) we use a simple single hidden layer with 10 nodes and we see that the training doesn't improve beyond a point despite adding more layers and doesn't even overfit to the data, showing us that for those features the dataset allows for only this high an accuracy. We have included the loss curve for the data in this case below.

In the other case(Model B), we use a slightly more complex model with a single hidden layer but 16 hidden nodes, and notice that adding a few features differently encoded led to the mode performing exceptionally well on the training data and overfitting to provide close to 99% accuracy and only 54% on the validation set. However, it is important to note that these same features do not perform as well using logistic regression due to lack of complexity of logistic regression and hence weren't chosen for the logistic regression case. We saw this as a breakthrough, and with the addition of the regularising parameter, the individual values of weight matrices decrease and move closer to each other leading to simpler models. Therefore, it reduces overfitting to quite an extent. We select a value of 0.00001

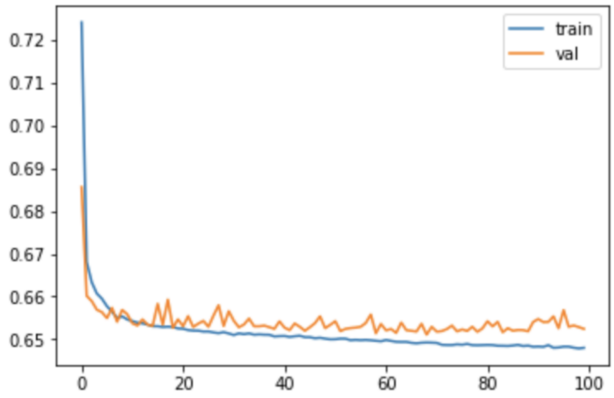

Figure 7: Training and validation loss curves for model A

as the kernel regulariser in the compile model stage and re-train the model to get to accuracy a high train accuracy of around 96% however, in this case obtaining a test accuracy of 90.74% as well. We analyse the results in more detail in the last section however, brought it up here to justify our eventual choice of model. The loss

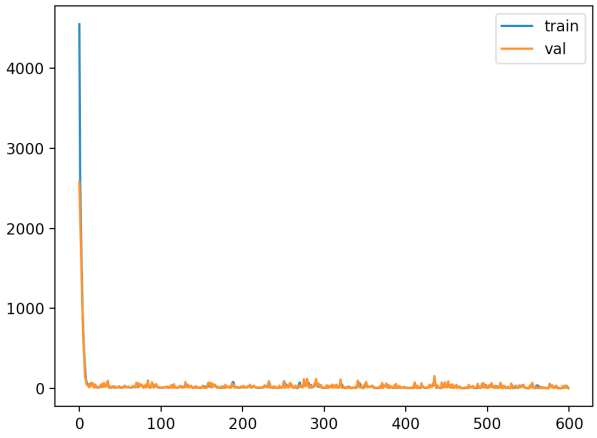

Figure 8: Training and validation loss curves for model B

function for both validation and the train data is as shown in the graph above. There is a steep decline in loss initially and remains more or less constant as the model progresses to train for 1000 epochs. We initially noticed that the loss function decreases and then stabilizes as expected, however there were random spikes in the loss function(i.e. few epochs had very high loss despite overall downward trend). We diagnosed the problem to be the missing values in the data for player height/weight and player age which were features that we didn't consider in the earlier models. We fixed this issue by removing such instances from the dataset. We lose around 1000 datapoints however, it is still a better decision, than letting the noise exist in the data. Although, this model performs well on the train, validation and test data, we believe that it could have overfitted onto this dataset, and since we don't have any data

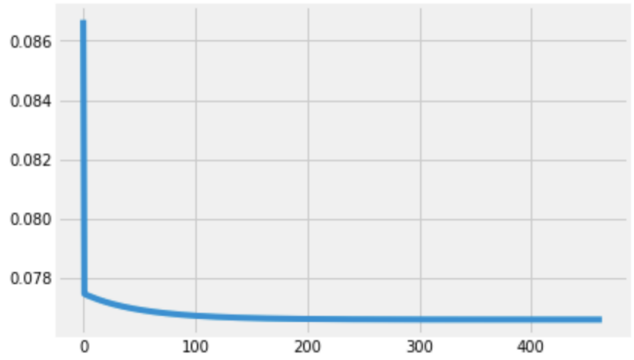

Figure 9: Variation of Mean Squared Error with the number of iterations of the update rule

from outside this dataset, it is hard to confirm this hypothesis and we hence optimistically assume that the model hasn not overfitted.

4.3 Recommendation system

In this section we discuss the recommendation task of predicting the best defenders for any given offensive player by scoring each pair of players based on shot efficiency of offensive player when being guarded by that defender. As discussed in the previous section, we generated player-defender pairs along with a corresponding score, that was determined by dividing the number of shots made by the number of shots attempted by that player when being guarded by said defender(shot%). This task turns out to be very similar in structure to a rating prediction task since, we eventually seek to determine the scores corresponding to all pairs(even ones not present in the dataset) and use that information to rank the defenders and identify the best one for each offensive player. We therefore use a latent factor model to complete this task. We begin with a simple latent factor model given by,

$$efficiency(p, d) = \alpha + \beta_p + \beta_d \tag{3}$$

where p and d represent the player and defender respectively. Here the β terms model the bias of each player and defender above the mean respectively, and alpha models the mean over all players and defenders. We also use a regularization factor λ while computing the cost function and look to minimise this cost function using a Mean Squared Error loss and do so using the iterative update rule until we achieve convergence. The plot below shows the variation in Mean Squared Error loss with each iteration. We analysed the results however, and didn't achieve much success and hence also attempted a more complex latent factor model including the γ parameter. The equation for this model becomes,

$$efficiency(p, d) = \alpha + \beta_p + \beta_d + \gamma_p \cdot \gamma_d \tag{4}$$

This model, in this case is better since it contains the cross terms that models the biases of player and defender scores for the pairs thus helping our model perform this scoring task better. We also include two regularisation terms λ_1, λ_2 corresponding to the β and γ terms respectively and repeat the same process of optimisation using the scipy optimise function.

However, this model didn't provide a significant improvement in the performance on this dataset as is discussed in the results. We noticed very small improvements despite ardent and strenous efforts to determine the best initialisation and hyperparameter values for the task. The results are discussed in greater detail in the final section.

5 LITERATURE REVIEW

In this section we briefly discuss the work that others have done in this field more specifically using this dataset or to solve this exact problem wherever possible. As already discussed briefly in the exploratory analysis of the data there is a lot of work previously done regarding the hot hand hypothesis[2, 5, 10] and the clutch gene hypothesis[1] and most involve simple statistical analysis and experimentation with no machine learning involved. We therefore, focus more on papers that implement machine learning modelling similar to ours, thus also helping us understand the current state of the art benchmark for relevant work.

Shot Prediction is a task that requires a lot of effort in order to be done accurately since the features that we have considered are an extremely small subset of all the features that infact contribute to the accuracy of a shot, some of which are extremely hard to quantify in the first place. Felcen and Lucy's work[7] on shot prediction is extremely interesting however, they use pose estimation on the video of the shot attempt to determine the quality of a shot. One would expect their results to be significantly better than our models given the better features and larger datasets, however it is only 8 percentage points better than our Logistic Regression model and they obtain 69.8% using LR, showing how unpredictable it is, and that there is large amounts of noise in any dataset. Another interesting piece of work[6] discusses various models for shot prediction and interestingly enough used the identical dataset as our work. His results as expected quite closely mimic ours in terms of accuracy of the Logistic Regressor, however, his neural network for the same task, though he hasn't described its exact architecture seems to perform worse than our model. Given that its the work of the student we believe that it's possible that we came up with a better model of our own. Other works along these lines include the paper by Y. Chang[11] which provides a new metric for the measurement of shot quality. This work inspired us to come up with interesting metrics for the evaluation of our model, however, the limitations of our dataset didn't allow us to fully exploit and implement these ideas.

The recommendation task based on shot logs however, has a lot less relevant literature. This is potentially mainly due to the fact that the shot log data, is not the best method to determine defensive performance as well as matchups due to large variability and various other contributing factors in the real world. This is being alleviated first by generating advanced statistics like player defensive impact[9] which will definitely help improve and increase the potential of such recommendation algorithms in the future. It could also be due to the lack of correlation and therefore inability to generate substantial results that has led to this deficiency literature. However, there ar einteresting articles that discusses other similar recommendation algorithms in the context of the NBA. Raymond Wright's work[8] discusses recommender systems for the best type

of shot attempts for each player in the NBA(by classifying shots, into categories, like layups, hook shots, midrange jumpers etc.). As a result, we don't have any comparison baselines making it harder to analyse our results deterministically.

Most of the work related to this data in the NBA for shot prediction or defender/shooter recommendation involve fairly more advanced stats like tracking data, shooting arc, player pose, defender pose team play running etc., which make up an extremely essential part of the analysis and in comparison our analysis is an extremely basic one. However, we believe that these analysis are necessary to serve as the basis for these more complex feature analysis to ensure the right direction and this analysis can scale well with more data, and data across multiple seasons. We believe, this work is more of a baseline metric for all future work using shot logs in the field and the starting point for this type of work as well.

6 RESULTS AND CONCLUSION

In this section we discuss and compare the results of our various models and their impact in the larger scheme. In the shot prediction task, we considered two models, the logistic regression model and the deep learning model. We experimented with the various features to find which ones work best for that model choice. The table below shows the results from the various models.

Model	Feature	Train Acc	Val Acc	Test Acc
LogReg	22	62.71%	62.35%	62.18%
MLP(Model A)	22	62.11%	61.81%	62.41%
MLP(Model B)	137	98.41%	98.3%	90.74%

Table 2: This table contains train validation and test performance for the various models along with the feature dimensions in each model.)

We must note that the logistic regression model and the MLP(model A) as explained earlier were trained on the same features. This shows us that increasing the model complexity didn't solve the problem better which means that our initial suspicion that the problem with the logistic regression model was its inability to model/learn the complex features of the dataset was wrong and in fact, the dataset itself doesn't allow for the accuracy to cross this threshold with these features. These features, are not deterministic and highly correlated enough with the data for the model to perform better. This although, not ideal sits well with intuition in that just 22 features is nowhere close to enough to determine with high certainty the outcome of a shot. As we have seen with other literature in the area[7], even with extremely advanced features that include pose of the player extracted from videos the models don't provide an accuracy greater than 70%. As a result, we were quite convinced with the performance of our model, despite the low accuracy numbers.

However, we did notice, that our models weren't even overfitting to the data with just these features which we expected to happen since the dataset is fairly small. Hence we decided to attempt including more features and one hot encode more features to provide for more range of weights to be learnt and use a larger Multi-Layer Perceptron and repeat the task. We created model B which uses a 137 dimension feature dimension instead. This model, performs better and provides really high values of training and validation accuracy. This model, perfectly models this dataset and since all our training, validation and test data is selected from this dataset, provides very high accuracy on this data. This was more in line with what we expected when we started out with this task. These results, are basically explained by the fact that model B has now very effectively all the patterns in this dataset and therefore has high accuracy on all the three sets(train, validation, test) and we expect that it would continue to perform well for any more data from the 2014-15 NBA season, for the same 281 offensive players. However, will not generalise well to the entire NBA across other seasons and other players not included in this dataset. Although, we strongly believe this is the case, we have no way of confirming this since, we don't have access to a dataset that can capture these variations.

Having generated these models, we believe that they serve as fairly good baselines for the shot prediction using simple text/numeric features and also hold up against all the other work in this field using similar features. Models like this can be used to devise better models with more advanced features including non-text/numeric data(e.g. video analytics based features, x,y coordinates of location on the floor etc.) many of which people have begun to collect only as recent as 2 years ago. These higher accuracy measurements can then be used to help NBA teams and staff to determine which players thrive in which situations and make better substitutions at as low as a game to game level, as well as make better trades by choosing the right players that fit into their systems at a season level. This work therefore, has an unbelievably large scope and more time and resources to scrape our own data and analyse it better would have let us develop much better and more generic and scalable models for each of the tasks.

The recommendation task, was much harder to analyse the results for. Although our mean squared error losses reduced and converged, we cannot evaluate exactly the performance based on players since small variations in the values causes the defenders to vary as shown in the table. Further, since the dataset is small, there is a much larger scope for outliers and small differences can affect the model drastically. Finally, the only way to analyse our model results is to look at the names of predicted best defender and compare it to the actual best defender based on the validation set and check if they are the same. We notice that although the scores for the predicted defenders and the actual defenders are quite close, the players are not the same in most cases, since a small difference in the score itself could lead to a change in the players since the variations are much smaller. A rating task contains ratings in steps of 1 or 0.5 making it slightly easier to determine however, in this case a 0.01% variation can lead to the name of the defender changing making it hard to analyse and do with a high accuracy. We have included below a list of the top 5 player defender pairs that had more than 10 attempts(so that we have a decent sample size, although this reduced the number of such players in the validation set to 17) along with their best defender and best predicted defender with the shot efficiency in the bracket. Note that, the shot efficiency scores are closely related however, the player names aren't always

Player	Model I	Model II
Paul Millsap	Carmelo Anthony(0.08)	Meyers Leonard(0.11)
Kobe Bryant	Jose Calderon(0.18)	Jeremy Lamb(0.16)
Deandre Jordan	Jared Sullinger(0.23)	C. Villanueva(0.24)
Solomon Hill	Tim Duncan(0.25)	Tim Duncan(0.24)
Jonas Valancunis	Rudy Gobert(0.27)	Jabari Parker(0.26)

Table 3: This table contains the top 5 defenders in the validation set against any player where Model II: Simple Latent factor model and Model I: Gamma included latent factor model. Further, the labels for these players are as follows from top to bottom: Gerald Henderson(0.10), Jose Calderon(0.18), Giannis A.(0.21), Tim Duncan(0.25), Rudy Gobert(0.27) respectively

the same. However we notice a couple of patterns here from a basketball standpoint that provides some confirmation that this task is performing fairly well. First, a lot of the defenders who show up in this list all-star defenders(i.e. top defenders in the league in that year) and are expected to hold their opponents to low shooting percentages. Second, the defenders and players in most pairs are generally from the same position which means that they are of similar physical nature making for a good matchup and explaining the good defense. In all, we believe that this task, due to lack of strong correlation might not have been the best choice of task for high accuracy, however, still shows some correlation and motivates us to believe there is larger scope for this work, provided there are more features(resources) and time available for this task.

In conclusion we would like to mention that these tasks although, fairly simple and straightforward looking, turned out to be quite the nightmare not only due to lack of strong enough correlation in the data(which is something we had expected) but also because of our choice of dataset. Since, we chose the dataset from Kaggle we were at its mercy to a large extent and had to deal with various issues like missing values, wrong values, unexpected zeros, NaN values etc., thus not only limiting the extent of progress of our work, but also affecting the accuracy and performance of our models, due to loss of data, and large amounts of noise in the data. We however, exploited the dataset to the best of our abilities and believe to have extracted the most out of it. And finally, sports itself as mentioned earlier, contains large amount of randomness and uncertainity especially in a high rally sport like basketball which is what makes it extremely thrilling and one of the most enjoyed forms of entertainment yet affects the ability to model various parts of basketball extremely accurately and use analytics to take make extremely decisive choices. However, as we have shown in this work, can still be used to perform a high level analysis that can always be used to guide and improve the quality of the sport especially in terms of the quality of play of all the teams in the NBA.

REFERENCES

[1] Chris Buetti. 2017. The Most Clutch Shot Clock Shooters in the NBA — A Statistical Approach. Retrieved Dec 2, 2019 from https://medium.com/@chrisbuetti/the-most-clutch-shot-clock-shooters-in-the-nba-a-statistical-approach-8982deda01e3

[2] Thomas Gilovich and Amos Tversky. 2012. The "Hot Hand": Statistical Reality or Cognitive Illusion? *CHANCE* 2, 4 (Sept. 2012), 31–34. https://doi.org/10.1080/09332480.1989.10554951

[3] https://www.basketball reference.com/. 2019. NBA Season Totals. Retrieved Dec 2, 2019 from https://www.basketball-reference.com/leagues/NBA_2015_totals.html

[4] https://www.wikipedia.org/. 2019. Hot Hand Fallacy. Retrieved Dec 2, 2019 from https://en.wikipedia.org/wiki/Hot_hand

[5] Adam Sanjurjo Joshua B. Miller. 2018. Surprised by the hot hand fallacy? The truth in the law of small numbers. *Econometrica* 86, 6 (Nov. 2018), 2019–2047. https://doi.org/10.2139/ssrn.2627354

[6] Brett Meehan. 2017. NBA Shot Prediction. Retrieved Dec 2, 2019 from http://cs229.stanford.edu/proj2017/final-reports/5132133.pdf

[7] Patrick Lucey Panna Felsen. 2017. "Body Shots": Analyzing Shooting Styles in the NBA using Body Pose. *MIT SLOAN, Sports Analytics Conference* (March 2017). http://www.sloansportsconference.com/wp-content/uploads/2017/02/1690.pdf

[8] Ilknur Kaynar-Kabul Raymond E. Wright, Jorge Silva. 2016. Shot Recommender System for NBA Coaches. (March 2016). https://doi.org/10.475/123_4

[9] stats.nba.com. 2019. Advanced defensive stats. Retrieved Dec 2, 2019 from https://stats.nba.com/players/defensive-impact/

[10] Robert Vallone Thomas Gilovich and Amos Tversky. 1985. The hot hand in basketball: On the misperception of random sequences. *Cognitive Psychology* 17, 3 (July 1985), 295–314. https://doi.org/10.1016/0010-0285(85)90010-6

[11] Jeff Su-Sheldon Kwok Tal Levy Adam Wexler Kevin Squire Second Spectrum Inc Yu-Han Chang, Rajiv Maheswaran. 2014. Quantifying Shot Quality in the NBA. *MIT SLOAN, Sports Analytics Conference* (Feb. 2014). http://www.sloansportsconference.com/wp-content/uploads/2014/02/2014-SSAC-Quantifying-Shot-Quality-in-the-NBA.pdf

CODE

Github Repository