Andrew Kozma
Akash Nadan
05 Dec 2016

ORIE 4741 Final Report: Predicting NBA Basketball Shots

Introduction

Basketball is a sport played by two teams, each with five players. The team with the most points at the end of 60 minutes wins the game. To score points, you must shoot a basketball, which is about the size of a pumpkin, into a hoop 10 feet off the ground (one on each end for each team). The ball falls down through the hoop, and the opposing five players try to prevent you from scoring. Every shot is 2 points unless it is made from past a line that is approximately a 20 foot radius away from the hoop. You can pass the ball to whomever you want and you must dribble the ball when you move with the ball. The NBA stands for the "National Basketball Association" and contains 30 teams of arguably the most skilled players in the world.

Data

We have a dataset consisting of features from NBA shots. It consists of every shot taken by every NBA player from October 28, 2014 to March 4, 2015. Each row, or example of the dataset represents a single shot, and each column, or feature, of our dataset contains information about that shot. Furthermore, our dataset can be conceptualized into two sets. One is data from each shot, such as the distance from the hoop or the number of points scored from the shot. The other is data from the game that the shot was taken in. This includes whether the player's team won the game and what the final margin was. All the features are summarized below in a table.

Data from each game	**Data from each shot**
The game number	Shot number
The opposing team	Period number
Home or away	Game clock
Win or loss	Shot clock
Final score of the game	Number of dribbles before shooting
	Time held onto the ball before shooting
	Distance of the shot
	2 point or 3 point shot
	Distance of closest defender
	Who is the closest defender
	Was the shot made
	Which player took the shot

Problem

We are interested in figuring out what factors influence a player's ability to make a shot in the NBA, and whether or not we can use these factors to predict whether or not a player will make a shot. An average or beginner basketball player can improve their shooting by practicing more, but since these players are already so skilled, we expect different game situations to have effects on their ability to make shots, and will try to identify the game features that have the greatest effect.

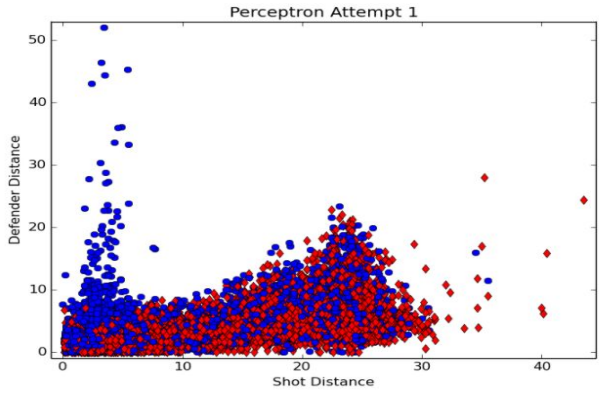
Approach

We tried to solve this problem using approaches that we learned in class. Our first step was to determine what the most significant features were in making a shot. For this we decided it was necessary to focus only on the features of our dataset containing information about the shot. We made an assumption that data from each game, such as the result of the game and the opposing team, would be insignificant in determining if a shot during that game was made. Thus, we ran a linear regression using R and selected the six most significant features from the dataset based on their P-values. These are the time on the shot clock, the number of dribbles before shooting, the amount of time a player touched the ball before shooting, the distance of a shot, whether the shot is a 2-pointer or 3-pointer, and the distance of the closest defender. We thought these features would be useful too because they are things that players have some control over when taking a shot. They can decide to hold on to the ball for an extra second, take another dribble, take a step back away from the defender, or line up behind the 3-point arc, to name a few examples. Thus, these features will be useful in predicting whether or not a player will make a shot.

Analysis

1. Perceptron

The first model we were interested in running on our data was the perceptron. The perceptron algorithm is an algorithm used to classify the data into two groups, which for our project is whether or not a shot is made. For the perceptron to work, the data must be linearly separable, and in order to figure that out, we decided to make a plot of the defender distance versus the shot distance. This plot is shown below.

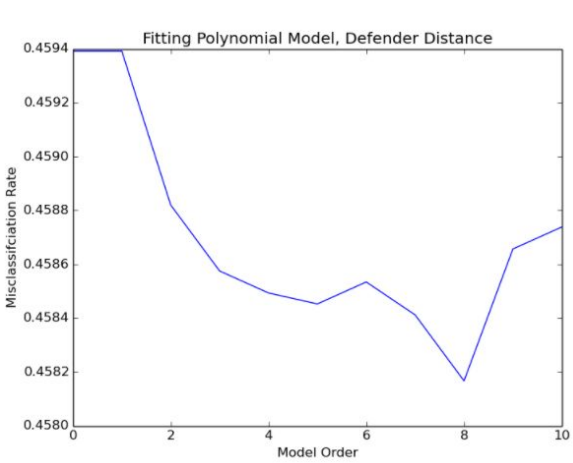


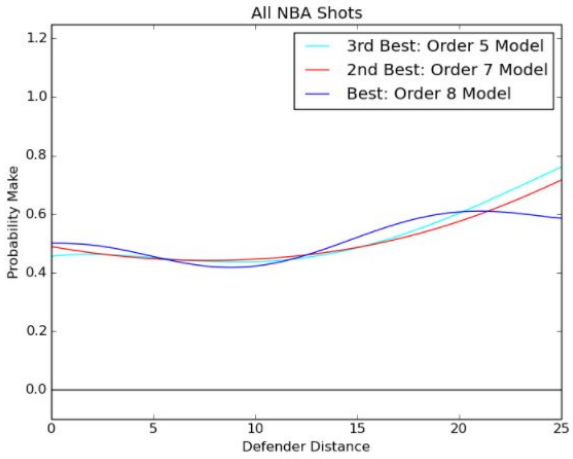
The blue represents shots that are made and the red points represent the shots missed. Because the points overlap so much, the data is not linearly separable. It is reassuring that the data is not linearly separable because there is no clear situation where players will definitely either make or miss a shot. Thus, the perceptron will not converge

nor find a solution to the problem of determining whether or not a shot is made. More complicated measures are needed to analyze this problem.

2. Polynomial Regression

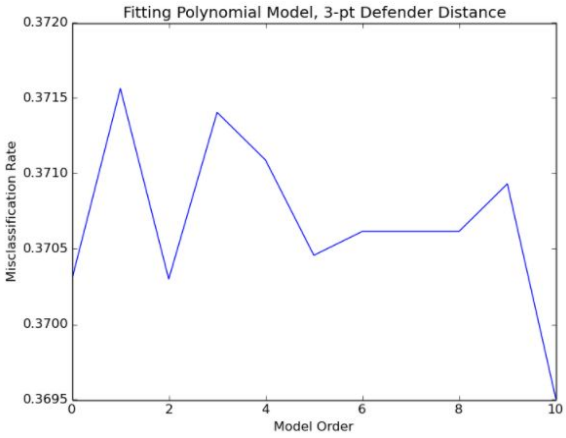
The next approach we tried was to fit a polynomial model to various features of our data. The polynomial model takes one feature and fits different orders of that feature to determine if a shot is made or not. A first order model is a linear model, second order is a polynomial, third is a cubic, and so on. The first feature we chose to fit a polynomial model to was the defender distance. We figured that the range of the polynomial model would be the probability that a shot is made given a distance from the closest defender. We created a Vandermonde matrix to store the various orders of the features, fit a model for orders 1 to 10 and calculated the misclassification rate for each of the models. The following plot on the left shows the misclassification rate on the test set for each of the models.

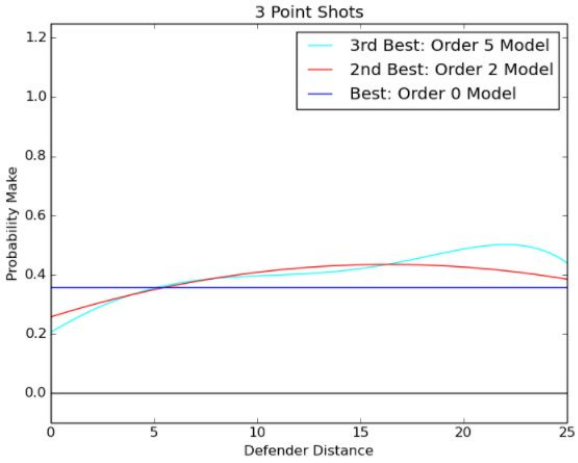




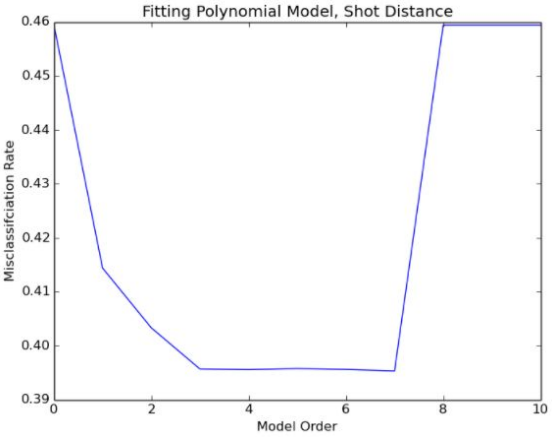
The polynomial model with the lowest misclassification rate on the data was the 8th order model. Thus to help visualize the results of this model, we plotted (above on the right) the resulting model against the distance from the closest defender. Since the predictions are either 0 or 1, the model predicts the probability of making that shot given a defender distance. It would not make sense to plot the results of the shot on this same graph because it would just contain a bunch of points at probability = 0 and 1.

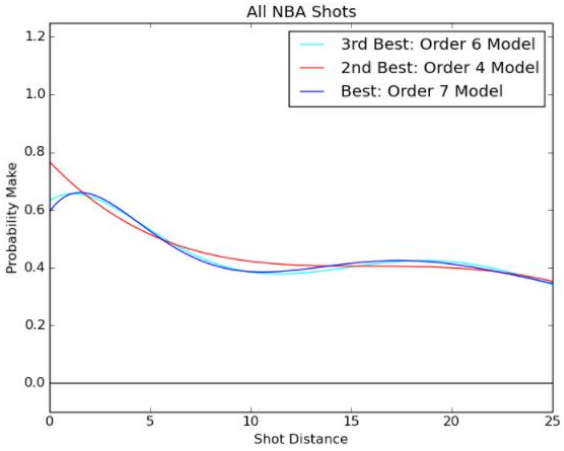
Just as a further investigation we decided to see how polynomial models with defender distance worked for only 3-pointers. We disregarded the model of 10th order because we felt that it overfit the data. Consequently, the best order was the 0th order, which predicts a constant probability a 3-pointer would be made no matter the defender distance. This would mean that the presence of a defender does not affect their ability to make a shot. The model with the second lowest misclassification rate, the second order model, as seen in the graph below on the right, predicts that one's shooting percentage actually increases slightly as the defender is farther and farther away. This is more consistent with one's intuition; it is easier to make a shot with people farther away.





Similarly, we ran polynomial models with shot distance from the hoop as the independent variable, and found that the 7th order model was the best at predicting whether or not the shot was made (seen in tables below).





We decided to run the polynomial models just as an initial assessment. The polynomial models only take into account one predictor variable at a time, but the following models that we ran take into account the entire array of significant predictor variables.

3. Loss Functions

Since this is a classification problem, we next chose to approach the problem by using different loss functions with the goal of classifying whether or not a shot would be made. We chose to find the parameters that would minimize our loss function plus the regularizer under the six features of the input space. All the models were fit to find the parameters using the training set, and then the misclassification rates were calculated using the test set.

The first model we chose was a quadratic loss with no regularizer, or the least squares problem. A quadratic loss is not the most ideal for a classification problem, nevertheless, it was run anyways in order to get an idea for how it performs on the data.

The resulting coefficients are shown in the table below, and produced the following misclassification rate: **Quad Loss = 0.3992**

We wanted the next models that we ran to have loss functions suitable for classification. Therefore, we chose to run logistic loss and hinge loss. First, we ran both these models to find parameters without any regularizations, and achieved the following misclassification rates: L**ogistic Loss = 0.4005 and Hinge Loss = 0.4096**

We then decided to add some regularization to the parameters. We thought it would be interesting to run the l1 regularizer on our data. Since this regularizer encourages sparsity in the fitted parameters, it would be perfect for the problem we are interested in. A player cannot take many factors into consideration when taking the shot; he just plays how he has always been playing. It would be nice if there were features that were more important than others in this model, so a player could have only a couple things to consider when trying to increase their chances of making a shot. This l1 regularizer was run with both a hinge loss and a logistic loss, and indeed achieved sparsity. The only coefficient that was nonzero was the shot distance for both types of losses. The model with 6 features is thus reduced to a model with one feature using this l1 regularization. The results are summarized in a table below. Furthermore, the following misclassification rates were observed: **Hinge Loss with l1 regularizer = 0.4594 and Logistic Loss with l1 regularizer = 0.4594**

We then decided to try a quadratic regularizer instead of an l1 regularizer with hinge loss. This model is called a Support Vector Machine (SVM). SVM models help avoid overfitting and provide a safety margin on the classification boundary. The SVM was run and achieved the following misclassification rate: **Hinge Loss with quad regularizer = 0.4001**

Similarly, we decided to try a quadratic regularizer with a logistic loss which produces a logistic regression model. A logistic regression model tends to have a low variances and is robust to noise in the data. The logistic model produces a prediction value in the format of a probability which is extremely useful in case we want to rank the different types of shots and their likelihood of success as a possible extension. However, for our current problem, after looping through various threshold values and measuring their training misclassification error, we decided to use a threshold of 0.50, which means that if the predicted probability was greater than 0.50 then we predicted this shot to be a make and if it was less than 0.50 then we predicted this shot to be a miss. The logistic regression model was run and achieved the following misclassification rate: **logistic regression = 0.3991**

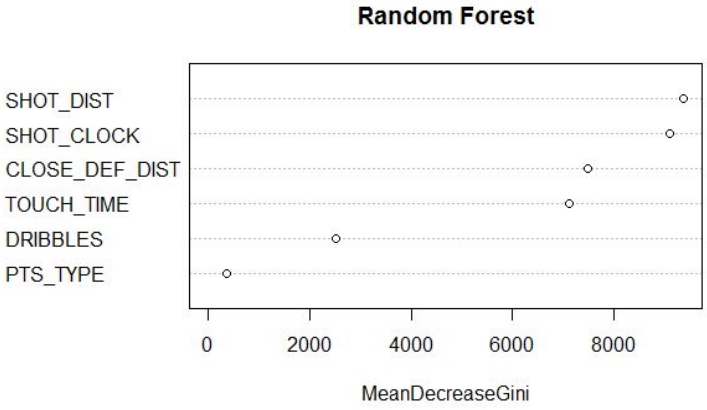
Below is a table of all the parameters from the combination of loss functions and regularizers that were run.

Results from Proximal Gradient With Loss Functions					
Feature Names	**Quadratic Loss, No Regularizer**	**Hinge Loss, L1 Regularizer**	**Logistic Loss, L1 Regularizer**	**Hinge Loss, Quadratic Regularizer**	**Logistic Loss, Quadratic Regularizer**
Shot Clock	0.0068	0.000	0.000	0.0407	0.0207
Dribbles	0.0123	0.000	0.000	-0.0131	-0.0072
Touch Time	-0.0267	0.000	0.000	-0.0194	-0.0115
Shot Dist	-0.0293	-0.0404	-0.0061	-0.0728	-0.0407
Points Type	0.0392	0.000	0.000	0.0198	0.0016
Closest Def. Dist	0.0442	0.000	0.000	0.0433	0.0340

Table 1: Shows the model coefficients for various Loss Functions and Regularizers

4. Random Forest

Another model that we implemented for our classification problem was a Random Forest. We decided on implementing a Random Forest because they can be used for classification problems, prevent overfitting, and give estimates on variable importance. A Random Forest is developed by using a multitude of decision trees and then decides on the class of the input based on the mode of the classes predicted by the multiple decision trees. Because Random Forests uses many decision trees, overfitting is prevented, which is a common problem faced by single decision trees. Therefore we implemented our forest so that it would create 100 decision trees at each node and prevent overfitting the training data. After implementing the Random Forest model on our training data we used it to predict the outcomes of our test data and came up with the following misclassification rate: **Random Forest = 0.3990.**



Additionally, we used a the mean decrease in the Gini Index in order to identify which variables in the model have the greatest influence on the outcome of the shot. Based on our model we achieved an accuracy of 60.1% and found that both the distance of the shot from the hoop and the shot clock time were the most significant factors in deciding whether or not a shot was made.

5. K-Nearest Neighbors

We also developed a model using K-Nearest Neighbors (KNN) algorithm. The KNN algorithm predicts the class of a point based on the similarities between that point and its K closest neighbors. We decided to use this model because it is robust to noisy data and effective for large data sets, like the one we have. Because KNN determines the class of a point based on its similarities to other points, we started by normalizing our data. We normalize the data because the similarity between data points is based on Euclidean distance, and by normalizing the data we avoid skew produced by using varying scales for each feature. After normalizing the data, we built a KNN model based on our training data and then ran the model on our test data and came up with the following misclassification rate: **K-Nearest Neighbors = 0.4071**

Conclusion

Overall, we achieved the following misclassification rates from our models:

Misclassification Rates	
Model	**Misclassification Rate**
Quadratic Loss, No Regularizer	0.3992
Hinge Loss, No Regularizer	0.4096
Logistic Loss, No Regularizer	0.4005
Hinge Loss, L1 Regularizer	0.4594
Logistic Loss, L1 Regularizer	0.4594
Hinge Loss, Quadratic Regularizer (SVM)	0.4001
Logistic Loss, Quadratic Regularizer	0.3991
Random Forest	0.3990
K-Nearest Neighbors	0.4071

The Random Forest model produced the lowest misclassification rate of 0.3990. However, the Quadratic Loss with no regularizer and the Logistic Loss with quadratic regularizer also had extremely close misclassification rates. A few of the other models also had extremely close misclassification rates, and were only about 0.020 worse. Both of the models that used the L1 regularizer had the highest misclassification rates, and were approximately 0.0600 worse than the other models. If we had to recommend a model we would recommend using the Random Forest or the Quadratic Loss with no regularizer because they are computationally efficient and produced the lowest errors.

Based on our models we were able to identify that shot distance from the hoop, shot clock, and closest defender distance have the greatest influence on whether or not a

player will make their shot. We were able to determine these variables by analyzing their coefficients in each of the models, analyzing p-values, and using mean decrease in Gini Index. Furthermore, based on our analysis we found that of the shots being made from players, they have the highest percentage of success the closer they are to the hoop and the farther the closest defender is from them. These two ideas are things that are already known to the NBA, and many plays have been developed by teams in order to maximize these two criteria.

We feel good about the models that we recommended as the best predictors. Because all of the models have relatively close misclassification rates, this reconfirms that are models are doing as well as possible. However, because of the high misclassification rates, we are not willing to make changes about how NBA coaches advise their players to take certain shots. Also, NBA players are highly skilled athletes. They have developed certain tendencies and styles of play that can't be taught. They can alter the way they do certain things, but it is hard to change a habit that has been ingrained over years of practice. Making a basketball shot in the NBA has many extraneous variables that we are unable to account for, and therefore we cannot say that we would recommend NBA teams using our model in order to predict their shots, but overall the model does a sufficient job at analyzing an average NBA player.