MARCH 22, 2017

APPLYING DATA MINING TECHNIQUES TO THE NBA

ANDREW ZHANG
PSTAT231
Thursday 6pm

Abstract

Accuracy has always been a point that many athletes have trained for during their entire careers. What are some of the potential factors that determine if a shot is made of missed? Is there even a way to predict if a shot goes in or misses? Using classification methods such as LDA, QDA, KNN, Decision Tree, and SVM, we can find the accuracy of these methods in predicting missed or made shots. After testing all 5 methods, we can conclude that Decision Tree was the best as it produced the highest accuracy rate amongst all of the methods.

Introduction

The data in this project is an NBA shot-log that contains variables such as shot result, shot distance, time holding the ball, number of dribbles, closest defender distance, and who is shooting the ball. While it is interesting to see what factors determine whether or not a shot goes into the hoop or misses, we can use this data to study what makes a great shooter based on number of shots made or what makes a great defender based on the number of shots missed. Although these questions are interesting, the one that this project addresses is whether or not we can predict shots made or missed and which method provides the best accuracy rate. After running through these methods, it seems like Decision Tree produced the best result as the accuracy rate was around 77.8% while the misclassification error was around 22.2%. Overall, the results of the project seemed positive in that the variables provided by the data were sufficient enough in predicting shot results.

Data

The data consists of shot data from various players including Kobe Bryant, LeBron James, Stephen Curry, and Russell Westbrook from the 2015 NBA season. Along with players, it includes data such as whether or not the shot made or missed, number of dribbles prior to shooting, whether it was a home or an away game, and the margin of victory. All in all, the original data contains 21 variables, however, this had to reduced as some of the variables played little part in affecting shot results.

Preprocessing

Prior to doing any work with the data, we must first consider the importance of each variable. Factors such as shot distance and closest defender distance will clearly play a part in determining the shot result, but factors such as win margin and game id are hard to quantify and play a lesser role in determining shot result. Therefore, these variables must be removed.

```
head(shots)

##    GAME_ID                  MATCHUP LOCATION W FINAL_MARGIN SHOT_NUMBER
## 1 21400899 MAR 04, 2015 - CHA @ BKN        A W           24           1
## 2 21400899 MAR 04, 2015 - CHA @ BKN        A W           24           2
## 3 21400899 MAR 04, 2015 - CHA @ BKN        A W           24           3
## 4 21400899 MAR 04, 2015 - CHA @ BKN        A W           24           4
## 5 21400899 MAR 04, 2015 - CHA @ BKN        A W           24           5
## 6 21400899 MAR 04, 2015 - CHA @ BKN        A W           24           6
##   PERIOD GAME_CLOCK SHOT_CLOCK DRIBBLES TOUCH_TIME SHOT_DIST PTS_TYPE
## 1      1       1:09       10.8        2        1.9       7.7        2
## 2      1       0:14        3.4        0        0.8      28.2        3
## 3      1       0:00         NA        3        2.7      10.1        2
## 4      2      11:47       10.3         2        1.9       17.2         2
## 5      2      10:34       10.9         2        2.7        3.7         2
## 6      2       8:15        9.1         2        4.4       18.4         2
##   SHOT_RESULT  CLOSEST_DEFENDER CLOSEST_DEFENDER_PLAYER_ID CLOSE_DEF_DIST
## 1        made     Anderson, Alan                     101187            1.3
## 2      missed Bogdanovic, Bojan                      202711            6.1
## 3      missed Bogdanovic, Bojan                      202711            0.9
## 4      missed      Brown, Markel                     203900            3.4
## 5      missed    Young, Thaddeus                     201152            1.1
## 6      missed    Williams, Deron                     101114            2.6
##   FGM PTS   player_name player_id
## 1   1   2 brian roberts    203148
## 2   0   0 brian roberts    203148
## 3   0   0 brian roberts    203148
## 4   0   0 brian roberts    203148
## 5   0   0 brian roberts    203148
## 6   0   0 brian roberts    203148
```

Table 1

```
head(agg_shots)

##   Location Shot_Number Period Shot_Clock Dribbles Touch_Time Shot_Distance
## 1        A           1      1       10.8        2        1.9           7.7
## 2        A           2      1        3.4        0        0.8          28.2
## 3        A           3      1         NA        3        2.7          10.1
## 4        A           4      2       10.3        2        1.9          17.2
## 5        A           5      2       10.9        2        2.7           3.7
## 6        A           6      2        9.1        2        4.4          18.4
##   Pts_Type Shot_Result Close_Def_Dist   Player_Name
## 1        2        made            1.3 brian roberts
## 2        3      missed            6.1 brian roberts
## 3        2      missed            0.9 brian roberts
## 4        2      missed            3.4 brian roberts
## 5        2      missed            1.1 brian roberts
## 6        2      missed            2.6 brian Roberts
```

Table 2

```
head(comp_agg_shots)

##   Location Shot_Number Period Shot_Clock Dribbles Touch_Time Shot_Distance
## 1        A           5      2       15.0        0        6.7          26.4
## 2        A          12      3       19.7        3        3.0          24.5
## 3        H           5      2       16.0        1        3.8           4.8
## 4        H          12      4       13.5       12        9.7           3.4
## 5        H           5      1        6.0        0        3.7          24.8
## 6        H          12      4       18.0        8        6.3          19.3
##   Pts_Type Shot_Result Close_Def_Dist Player_Name
## 1        3      missed            5.7 kobe bryant
## 2        3      missed            3.1 kobe bryant
## 3        2      missed            0.9 kobe bryant
## 4        2        made             3.3 kobe bryant
## 5        3      missed             4.5 kobe bryant
## 6        2      missed             4.9 kobe bryant
```

Table 3

Table 1 shows 21 different variables from the original data. There are also around 128,000 observations. Table 2 shows 11 different variables after the first phase of preprocessing. This step included filtering out any unnecessary variable. This table also contains around 128,000 observations. Table 3 shows 11 different variables with only around 829 observations. This table reflects both the removal of unnecessary variables and only including shot data for 7 players rather than the entire NBA.

```
summary(comp_agg_shots)

##  Location  Shot_Number       Period        Shot_Clock
##  A:431    Min.   : 1.00   Min.   :1.000   Min.   : 0.000
##  H:398    1st Qu.: 5.00   1st Qu.:1.000   1st Qu.: 8.775
##           Median : 9.00   Median :2.000   Median :13.650
##           Mean   :10.23   Mean   :2.437   Mean   :13.361
##           3rd Qu.:15.00   3rd Qu.:3.000   3rd Qu.:18.000
##           Max.   :37.00   Max.   :5.000   Max.   :24.000
##                                           NA's   :33
##     Dribbles        Touch_Time     Shot_Distance      Pts_Type
##  Min.   : 0.000   Min.   :-0.400   Min.   : 0.10   Min.   :2.000
##  1st Qu.: 0.000   1st Qu.: 1.100   1st Qu.: 4.60   1st Qu.:2.000
##  Median : 2.000   Median : 3.100   Median :13.40   Median :2.000
##  Mean   : 3.691   Mean   : 4.341   Mean   :13.43   Mean   :2.217
##  3rd Qu.: 6.000   3rd Qu.: 6.300   3rd Qu.:21.70   3rd Qu.:2.000
##  Max.   :25.000   Max.   :21.200   Max.   :42.70   Max.   :3.000
##
##  Shot_Result  Close_Def_Dist              Player_Name
##  made  :390   Min.   : 0.000   lebron james     :140
##  missed:439   1st Qu.: 2.200   russell westbrook:138
##               Median : 3.500   stephen curry    :138
##               Mean   : 3.813   kyrie irving     :135
##               3rd Qu.: 4.800   anthony davis    :120
##               Max.   :19.600   kobe bryant      :102
##                                (Other)          : 56
```

Table 4

Summary statistics for comp_agg_shots

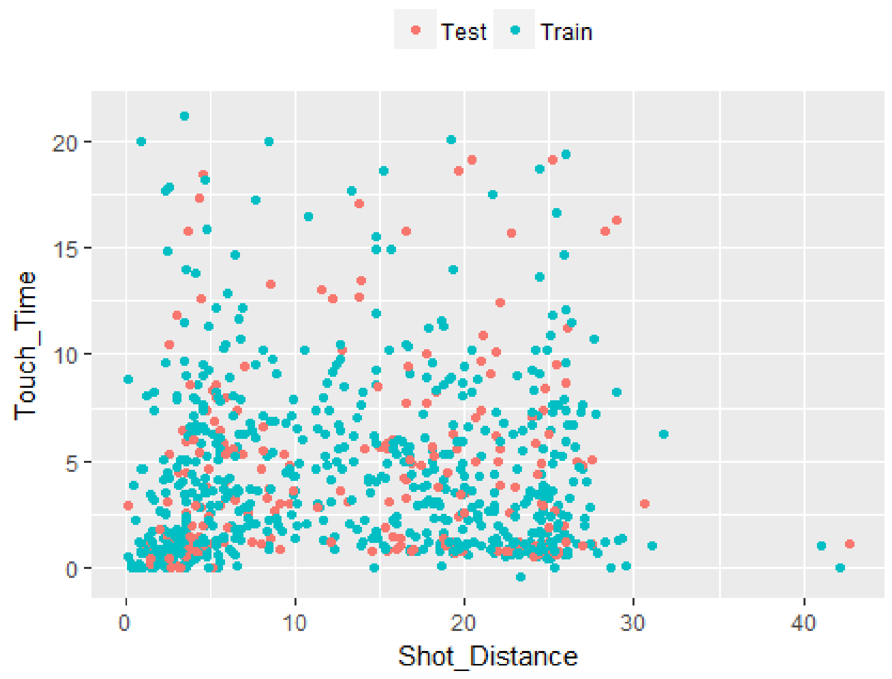

Figure 1

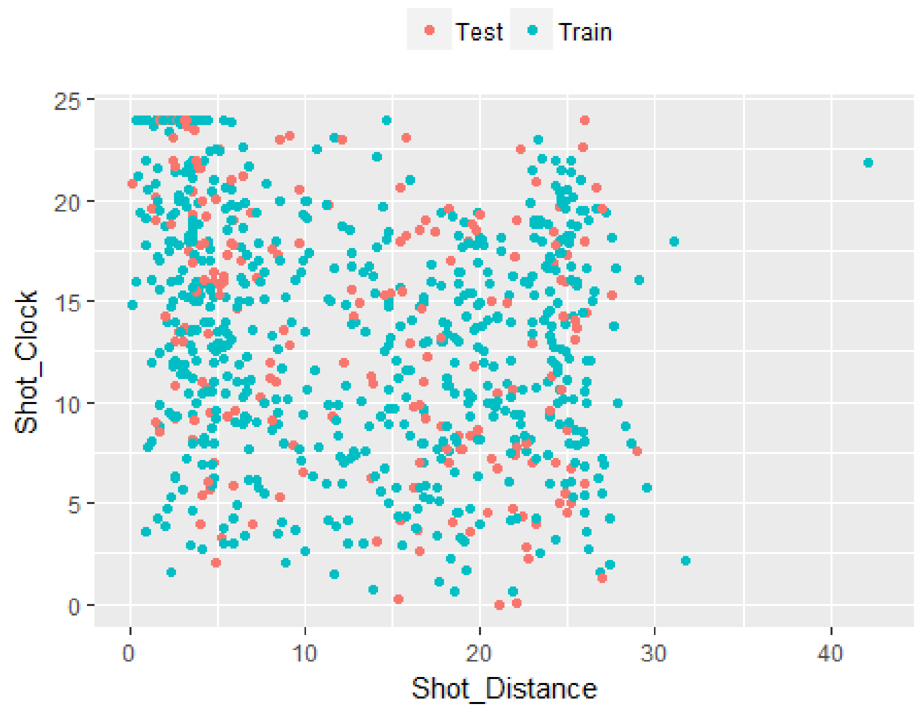

Figure 2

Figures 1 and 2 are correlation plots that map out potential interactions between two variables. These plots look at shot distance, shot clock, and touch time. We can see in figure 1, the closer the shot distance is to the rim, the less the players hold on to the ball. These quick shots could be attributed to maybe having an open shot or numerous defenders forcing the shooter to shoot. In figure 2, it seems that the shot clock doesn't seem to affect the distance of the shot all too much. We can speculate further as to why shot distance and shot clock may play an important role in whether the shot is a make or miss.

PCA

Before applying any sort of tests to the aggregated data set, we can use PCA in order to check whether or not the variables selected are correlated in determining shot results.

```
str(comp_agg_shots)

## 'data.frame':    829 obs. of  11 variables:
##  $ Location     : Factor w/ 2 levels "A","H": 1 1 2 2 2 2 2 1 2 2 ...
##  $ Shot_Number  : int  5 12 5 12 5 12 19 7 2 9 ...
##  $ Period       : int  2 3 2 4 1 4 4 3 1 3 ...
##  $ Shot_Clock   : num  15 19.7 16 13.5 6 18 NA 7.6 17.9 6 ...
##  $ Dribbles     : int  0 3 1 12 0 8 4 4 6 5 ...
##  $ Touch_Time   : num  6.7 3 3.8 9.7 3.7 6.3 5.1 5 6.4 7.4 ...
##  $ Shot_Distance : num  26.4 24.5 4.8 3.4 24.8 19.3 27.5 13.4 25.1 11.2 ..
.
##  $ Pts_Type     : int  3 3 2 2 3 2 3 2 3 2 ...
##  $ Shot_Result  : Factor w/ 2 levels "made","missed": 2 2 2 1 2 2 2 2 1 1
...
##  $ Close_Def_Dist: num  5.7 3.1 0.9 3.3 4.5 4.9 3.2 4.1 3.4 1.7 ...
##  $ Player_Name  : Factor w/ 281 levels "aaron brooks",..: 159 159 159 159
159 159 159 159 159 159 ...
```

Table 5

```
str(comp_agg_shots)

## 'data.frame':    829 obs. of  11 variables:
##  $ Location     : num  1 1 2 2 2 2 2 1 2 2 ...
##  $ Shot_Number  : int  5 12 5 12 5 12 19 7 2 9 ...
##  $ Period       : int  2 3 2 4 1 4 4 3 1 3 ...
##  $ Shot_Clock   : num  15 19.7 16 13.5 6 18 0 7.6 17.9 6 ...
##  $ Dribbles     : int  0 3 1 12 0 8 4 4 6 5 ...
##  $ Touch_Time   : num  6.7 3 3.8 9.7 3.7 6.3 5.1 5 6.4 7.4 ...
##  $ Shot_Distance : num  26.4 24.5 4.8 3.4 24.8 19.3 27.5 13.4 25.1 11.2
...
##  $ Pts_Type     : int  3 3 2 2 3 2 3 2 3 2 ...
##  $ Shot_Result  : num  2 2 2 1 2 2 2 2 1 1 ...
##  $ Close_Def_Dist: num  5.7 3.1 0.9 3.3 4.5 4.9 3.2 4.1 3.4 1.7 ...
##  $ Player_Name  : num  159 159 159 159 159 159 159 159 159 159 ...
```

Table 6

Table 5 reveals what type each variable is in the dataset. As we can see, Location and Player_Name are both characters, therefore they cannot be processed by PCA as PCA only accepts numerical variables. For this we convert both variables to numerics where in Location, away is recognized by 1 and home is recognized as 2. Player_Name is given numeric values based on their alphabetical place in the original data. Table 6 shows a dataset suitable for PCA.

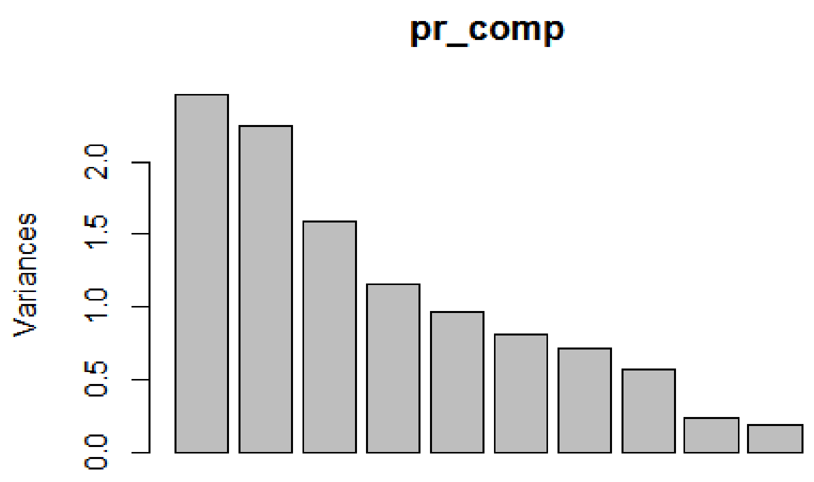

Figure 3

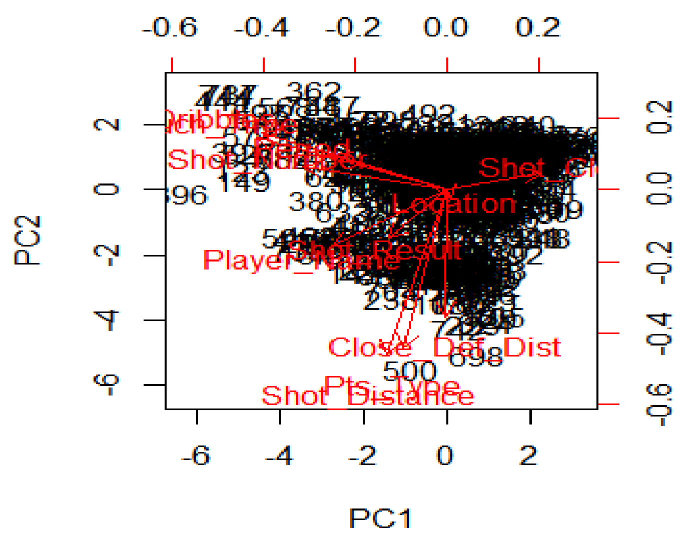

Figure 4

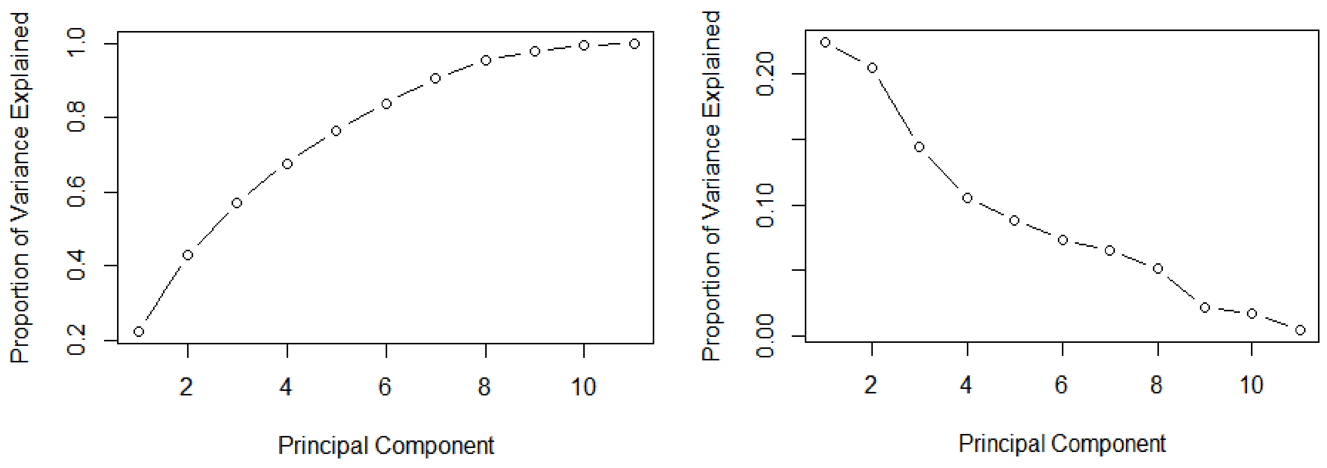

Figure 5

Figure 3 shows the distribution of the principal components for the training set, These plots are all based on training and test data which was split 75% to 25%. Figure 4 shows the results of the biplot of the PCA, while Figure 5 shows the scree plots for the Principal Components. Looking at the scree plot, it is hard to determine which variables are unnecessary as much of the variance is explained by all PC's. This confirms that all the variables we have selected are viable factors in determining shot results.

Decision Tree

Now that we have finished all preprocessing and dimensionality reduction, we can begin testing methods to determine which is the most accurate at predicting shot results. Decision tree uses the tree function to determine which variables seemingly influence the result of make of miss. In this portion, we tested 3 different trees, an ordinary tree, a tree with more specific nodes, and the most optimal tree to see which yielded the best accuracy rate and smallest misclassification.

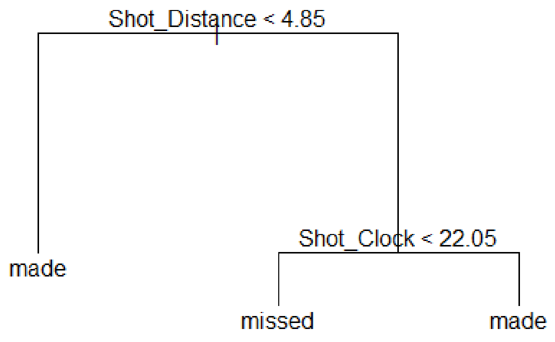

Figure 6

```
##                    Shot_test
## shot.tree.pred1 made missed
##            made    40      23
##            missed  56      89
```

```
#Accuracy
sum(diag(error1))/sum(error1)
## [1] 0.6201923

#Misclassification Error
1-sum(diag(error1))/sum(error1)

## [1] 0.3798077
```

Table 7

From the first tree, we can see that the algorithm determined that most made/missed shots were determined by Shot_Distance, Touch_Time, and Shot_Clock. Table 7 shows that this model is 62% accurate with a 37.9% misclassification rate.

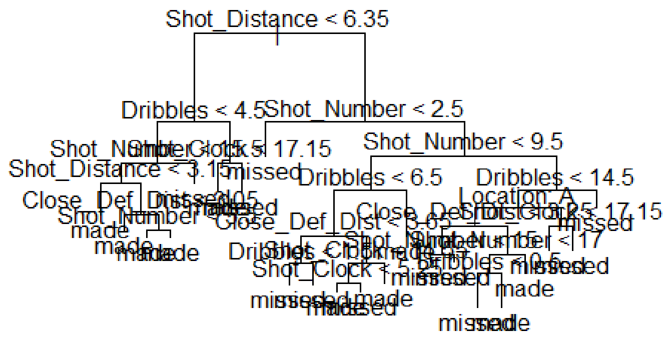

Figure 7

```
##                   Shot_test
## shot.tree.pred2 made missed
##             made     75     25
##             missed   21     87

#Accuracy
sum(diag(error2))/sum(error2)

## [1] 0.7788462

#Train Error Rate
1-sum(diag(error2))/sum(error2)

## [1] 0.2211538
```

Table 8

Figure 7 represents the larger tree with more nodes as the conditions set in the tree function were more specific. As you can see, it is harder to distinguish which variables play a part in determining the shot result. However, this tree yields an accuracy of 77.9% and a misclassification error of 22.1%. This can probably be attributed to the fact that made and missed shots are determined by multiple factors rather than a few. The more conditions, the more accurate.

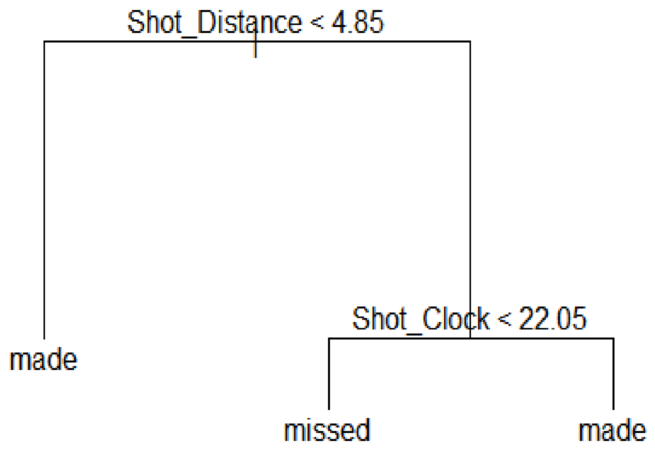

Figure 8

```
##   made missed
##     63    145

#Accuracy
sum(diag(error3))/sum(error3) #62.0
%

## [1] 0.6201923

#Train Error
1-sum(diag(error3))/sum(error3) #38
.0%

## [1] 0.3798077
```

Table 8

Figure 8 shows the most optimal tree based on the data presented. It is very similar to Figure 6 in terms of simplicity and variables used. The accuracy rate for this model is 62% while the misclassification error is 37.9%, which again is very similar to that of Figure 6. What's interesting is that Figure 7 has a better accuracy and misclassification error than the supposed most optimal tree. This could be because the algorithm generalizes what variables determine miss and makes, but the larger tree uses specific conditions to determine miss and makes.

Random Forest

```
##         OOB estimate of  error rate: 40.64%
## Confusion matrix:
##        made missed class.error
## made    145    141   0.4930070
## missed  102    210   0.3269231
```

Table 9

Following Decision Tree, we can use Random Forest to create an importance plot to identify which variables are the most important in determining shot results. This, in turn, can also help explain why the decision tree's accuracy was what it was. From Table 10, we can see that there is a 40% error rate, possibly resulting from an overgeneralization of the variables used to determine makes or misses.

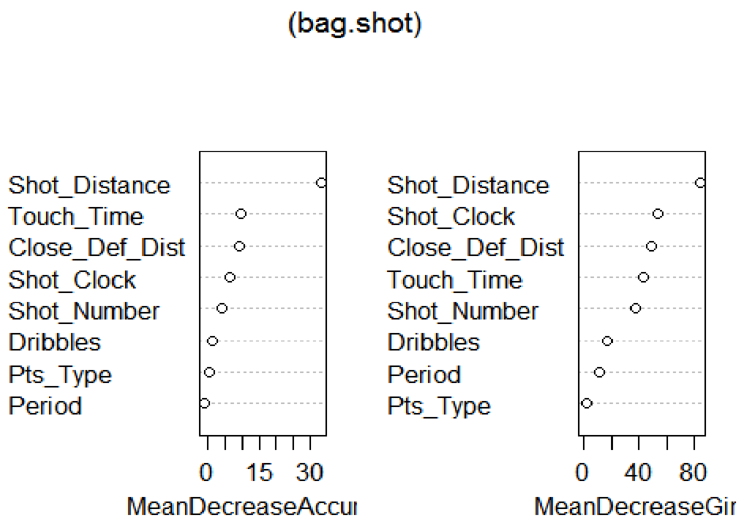

Figure 9

Figure 9 shows the importance plot for the variables utilized in the decision tree. As we can see, shot distance, shot clock, and touch time seemed to play an integral part in determining the outcome of the shot. This is reflected in Figures 6, 7, and 8. However, the depth in which each variable is analyzed in comparison with others plays an instrumental part in determining accuracy of the model as well.

SVM

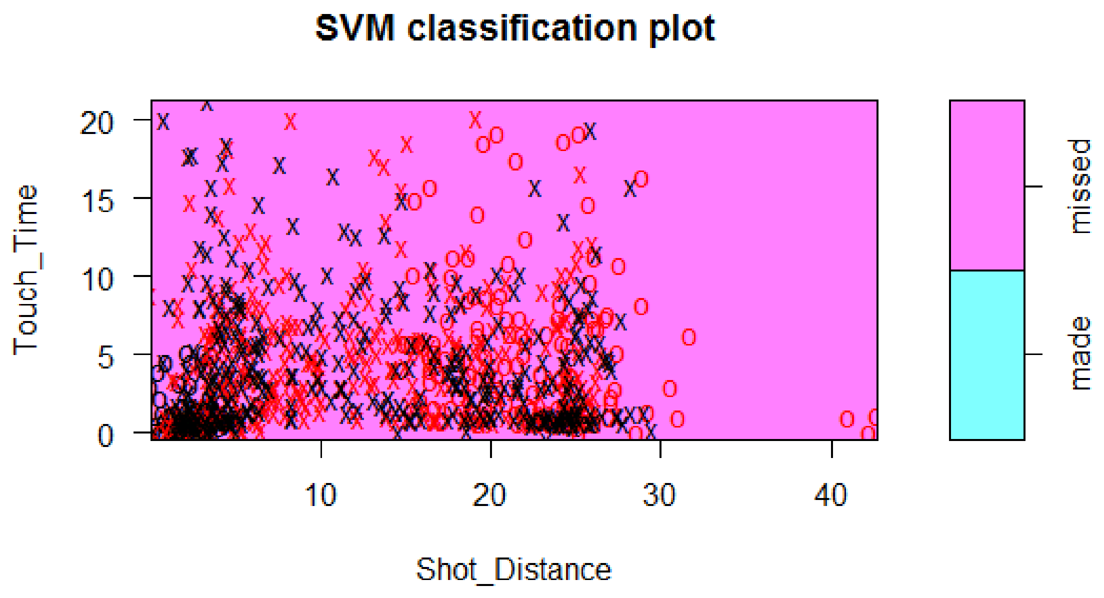

Figure 10

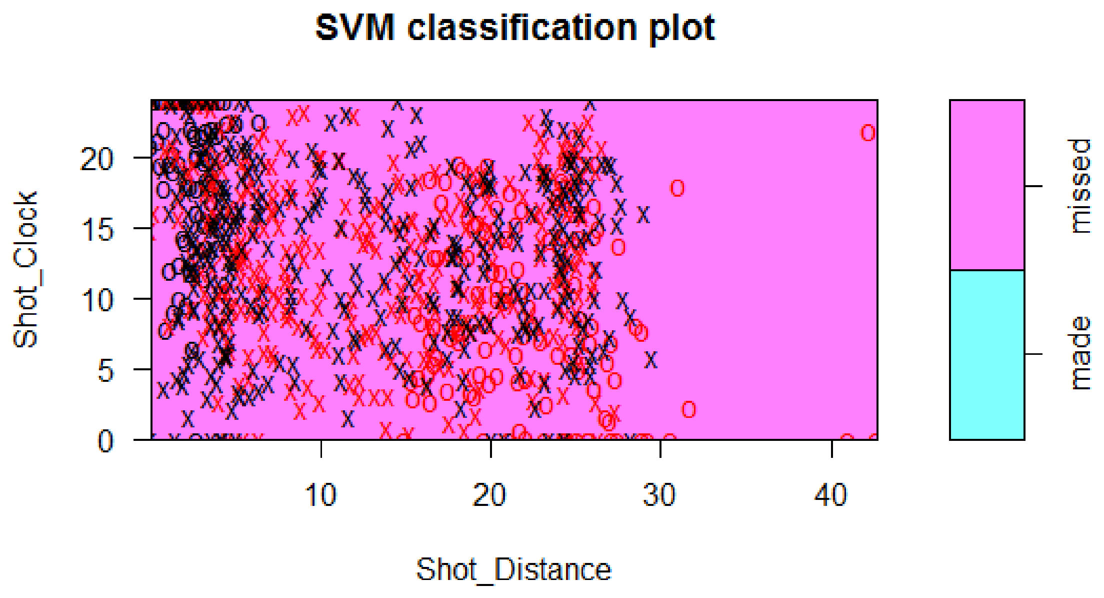

Figure 11

Following the importance plot, we can use the SVM function to create a classification plot for the most important variables denoted by the importance plot. In figure 10 and 11, we will analyze 3 different variables, shot distance, shot clock, and touch time. Through this plot, we can see the correlation between each of the variables.

```
##
## shot_svm_pred made  missed
##        made    219    130
##        missed  171    309

#Accuracy
sum(diag(svm_table)/sum(svm_table))

## [1] 0.6369119

#Test Error
1-sum(diag(svm_table)/sum(svm_table))

## [1] 0.3630881
```

Table 10

Table 11 shows the accuracy and test error of the SVM prediction. The accuracy is given at 63.7% while the misclassification error for the test set is around 36.3%.

K-Nearest Neighbor

The next test we are going to look at is the K-Nearest Neighbor test where we can determine accuracy based on how close points are to other points.

```
#Predict train set for k=5
pred.YTrain = knn(XTrain, XTrain, YTrain, k=5)
train$pred5 = pred.YTrain
conf.matrix5 = table(pred.YTrain, YTrain)

#Accuracy
sum(diag(conf.matrix5)/sum(conf.matrix5)) #71.3%

## [1] 0.7117552

#Train Error
1-sum(diag(conf.matrix5)/sum(conf.matrix5)) #28.7%

## [1] 0.2882448

#Predict train error for k=55
pred.YTrain = knn(XTrain, XTrain, YTrain, k=55)
train$pred55 = pred.YTrain
conf.matrix55 = table(pred.YTrain, YTrain)

#Accuracy
sum(diag(conf.matrix5)/sum(conf.matrix55)) #63.6%

## [1] 0.7117552

#Train Error
1-sum(diag(conf.matrix5)/sum(conf.matrix55)) #36.4%

## [1] 0.2882448
```

Table 11

Table 12 shows the sample accuracy and misclassification errors for the training error. The purpose of doing this is to randomly test different values of k as we are yet to know the best value of k.

```
#Predict test error for k=5
pred.YTest = knn(XTest, XTest, YTest, k=5)
test$pred5 = pred.YTest
conf.matrixtest5 = table(pred.YTest, YTest)

#Accuracy
sum(diag(conf.matrixtest5)/sum(conf.matrixtest5)) #77.4%

## [1] 0.7740385

#Test Error
1-sum(diag(conf.matrixtest5)/sum(conf.matrixtest5)) #22.5%

## [1] 0.2259615

#Predict test error for k=55
pred.YTest = knn(XTest, XTest, YTest, k=55)
test$pred55 = pred.YTest
conf.matrixtest55 = table(pred.YTest, YTest)

#Accuracy
sum(diag(conf.matrixtest55)/sum(conf.matrixtest55)) #62.98%

## [1] 0.6298077

#Test Error
1-sum(diag(conf.matrixtest55)/sum(conf.matrixtest55)) #37.0%

## [1] 0.3701923
```

Table 12

Table 13 shows the test error and accuracy for the test data. As we can see, the test data accuracy rate is higher at 77.4% compared to the training data at 71% even when k=5. We can begin to suspect that a lower k value may prove to be the most optimal k for this test.

```
#Choosing optimal K for test
train.error.rate = NULL
test.error.rate = NULL

knn.wrap1 <- function(k, YTest){
  pred.YTest = knn.cv(XTest, YTest, k=k)
  mean(YTest != pred.YTest)
}

test.error.rate = sapply(1:50, knn.wrap1, YTest = YTest)

test.error.rate #best k is 5 at 0.3461
```

```
##  [1] 0.4086538 0.3942308 0.4134615 0.3942308 0.3461538 0.3942308 0.3750000
##  [8] 0.3846154 0.3942308 0.3798077 0.3798077 0.3942308 0.4038462 0.3942308
## [15] 0.3942308 0.4086538 0.4086538 0.3750000 0.3846154 0.3798077 0.4134615
## [22] 0.3798077 0.3653846 0.3605769 0.3605769 0.3653846 0.3701923 0.3750000
## [29] 0.3701923 0.3701923 0.3653846 0.3798077 0.3750000 0.3798077 0.3750000
## [36] 0.3653846 0.3846154 0.3990385 0.4038462 0.3798077 0.3894231 0.3990385
## [43] 0.3894231 0.3990385 0.3894231 0.4038462 0.3990385 0.3942308 0.3942308
## [50] 0.3798077

knnTest1 <- knn.cv(XTest, YTest, 5)
mean(knnTest1 != YTest) #0.3461

## [1] 0.3461538

conf.matrixtestk = table(knnTest1, YTest)

#Accuracy
sum(diag(conf.matrixtestk)/sum(conf.matrixtestk)) #65.38%

## [1] 0.6538462

#Test Error
1-sum(diag(conf.matrixtestk)/sum(conf.matrixtestk)) #34.6%

## [1] 0.3461538
```

Table 13

Table 14 depicts the code for determining the best k value for the test set. As we can see, from the list of test error rates, we select k=5 as it has the smallest error rate at 34.6%. These numbers differ slightly as the function used in this portion of the data involves k-cross validation. However, from this set of data, we can conclude that k=5 is the most optimal value for knn as it yields the best accuracy rate and smallest misclassification rate.

LDA

The third test done on the data set was Linear Discriminant Analysis, which involves the use of the lda function to predict values from the dataset.

```
set.seed(2)
shots.lda <- lda(Shot_Result ~., data = test, subset = train_ind)

shots.hat <- predict(shots.lda, test)
lda_test1 <- comp_agg_shots$Shot_Result[-train_ind]
tab_hats <- table(lda_test1, shots.hat$class)
tab_hats

##
## lda_test1  1  2
##    made   70 26
##    missed 21 91
```

```r
#Accuracy
sum(diag(tab_hats)/sum(tab_hats)) #77.4%

## [1] 0.7740385

#Test error for lda1
1-sum(diag(tab_hats)/sum(tab_hats)) #22.6%

## [1] 0.2259615
```

Table 14

Table 15 displays the code for conducting the lda test on the dataset. As seen above, the test yielded a relatively high accuracy rate at 77.4%

QDA

The final test on the data was Quadratic Discriminant Analysis, which like its counter part LDA, utilizes the qda function to predict values from the data set.

```r
set.seed(2)
shots.qda = qda(Shot_Result ~., data = test, subset = train_ind)

shots.hat2 = predict(shots.qda, test)
qda_test1 <- comp_agg_shots$Shot_Result[-train_ind]

tab_hats2 <- table(qda_test1, shots.hat$class)
tab_hats2

##
## qda_test1  1  2
##    made   70 26
##    missed 21 91

#Accuracy
sum(diag(tab_hats2)/sum(tab_hats2)) #77.4%

## [1] 0.7740385

#Test error for qda1
1-sum(diag(tab_hats2)/sum(tab_hats2)) #22.6

## [1] 0.2259615
```

Table 15

Table 16 shows the results of running the qda function on the data. The accuracy is given at 77.4% while the misclassification error for the test set is 22.6%. This is interesting as it is identical to its counterpart, LDA.

Conclusion

Based on the 4 tests presented, it seems that the most accurate one in predicting missed or made shots was the large decision tree. Given that the accuracy rate was 77.8%, it was slightly higher than the accuracy rates of all the other methods. This could be because most of the other methods generalize the factors that go into making or missing the shot, however, by utilizing a multitude of specific conditions and nodes, the decision tree is able to more accurately predict the shot result. Unlike most other data sets, variables such as the mental state of the player, the aggressiveness of the defenders, and other such factors are very difficult to quantify. For that reason, it becomes very hard to accurately discern what factors play a significant role in shot making other than variables that can be seen and counted. However, these results are quite positive considering that there were many factors that could've been added to the data set in order to make the predictions much more accurate. On that note, statistics such as historic field goal percentage for players, number of shots taken throughout a players career, and a scale that classifies each player's shooting could be added in order to make these predictions more accurate. For the future, we can consider creating a scale to determine which players can be classified as great shooters or which players can be classified as shooters that are great towards the end of close games.

References

1. https://www.kaggle.com/dansbecker/nba-shot-logs
2. Professor OH's PSTAT 131/231 Lectures