NBA Shot Prediction

Alex Fung, Patrick Osborne, Tony Lee, Viswesh Krishnamurthy

14/03/2020

ABSTRACT

The use of various predictive metrics in sports has been occurring as long as human beings have watched each other compete. Initially this started out as simple gut feeling or a subjective assessment of the competitors – the bets usually go to the bigger fighter! In more recent times, humanity has refined its predictive power with the advent of statistics and logical decision making (famously put to use in Major League Baseball, as shown in the film Moneyball). With the advent of the information age and the possibility for large-scale data analytics and machine learning, the National Basketball Association has decided to puruse this analysis to better understand player matchups (defender vs offender) and to assess & optimize shooter performance.

BUSINESS UNDERSTANDING

Detailed statistics are already available to the NBA as these have been tracked for many years, supporting classic statistical decision making. The goal is to run both unsupervised and supervised machine learning algorithms on the statistical data available. In technical terms, we aim to deliver predictive metrics for threat/benefit level at an individual player level, as well as an interactive application that identifies the likelihood of a shot landing from a specific offender shooting from a specific position on the court, against a specific defender located a certain distance away. This will allow coaches to run limited scenarios in the predictive model, to inform both their practice routines and to assist in making strategic decisions during live games.

As an example, consider the matchup of Lebron James on offence and Serge Ibaka on defence. Let's assume that Lebron typically tries to shoot from top of the key, and is being defended by Serge Ibaka, 5 feet away. The model takes these discrete inputs and outputs a real-world percentage success of 10.5% (example). If the average shooting success rate is 30%, we can identify this as a bad shot, and encourage Lebron to pass in these situations.

DATA UNDERSTANDING

We begin with understanding each feature available in the data. The available data set is data of all shots attempted at NBA games between 2014 and 2015. For each shot attempted, the most important outcome of that attempt, whether the shot was made or missed is available. This is seen in 2 columns SHOT_RESULTS and FGM. FGM stands for “Field Goal Made”. In support of this outcome, there are a number of other data points to be seen, like the player who attempted the shot and who defended the sho, how far away was the defender, how far away from the basket was the shot attempted, was the match at home or away etc. With that data understanding, let’s look at the “head” of the data.

GAME_ID	MATCHUP	LOCATION	W	FINAL_MARGIN	SHOT_NUMBER	PERIOD	GAME_CLOCK
21400899	MAR 04, 2015 - CHA @ BKN	A	W	24	1	1	1:09
21400899	MAR 04, 2015 - CHA @ BKN	A	W	24	2	1	0:14
21400899	MAR 04, 2015 - CHA @ BKN	A	W	24	3	1	0:00
21400899	MAR 04, 2015 - CHA @ BKN	A	W	24	4	2	11:47
21400899	MAR 04, 2015 - CHA @ BKN	A	W	24	5	2	10:34
21400899	MAR 04, 2015 - CHA @ BKN	A	W	24	6	2	8:15

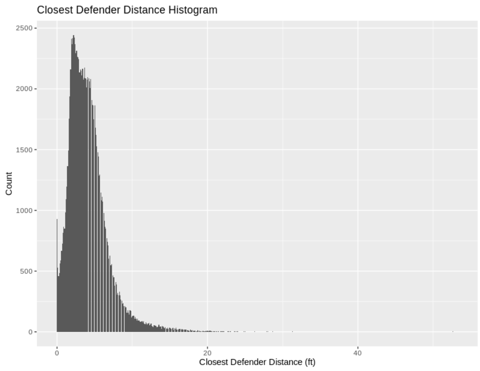
SHOT_CLOCK	DRIBBLES	TOUCH_TIME	SHOT_DIST	PTS_TYPE	SHOT_RESULT	CLOSEST_DEFENDER
10.80	2	1.90	7.70	2	made	Anderson, Alan
3.40	0	0.80	28.20	3	missed	Bogdanovic, Bojan
	3	2.70	10.10	2	missed	Bogdanovic, Bojan
10.30	2	1.90	17.20	2	missed	Brown, Markel
10.90	2	2.70	3.70	2	missed	Young, Thaddeus
9.10	2	4.40	18.40	2	missed	Williams, Deron

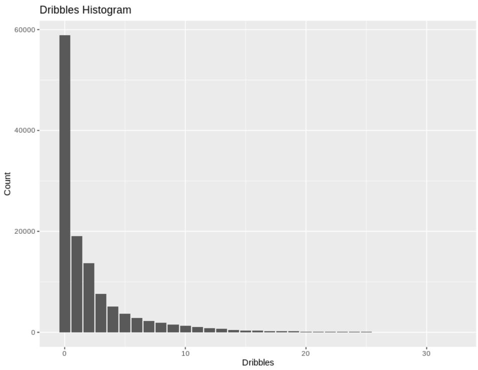
Table 1: Data Dictionary - NBA Data

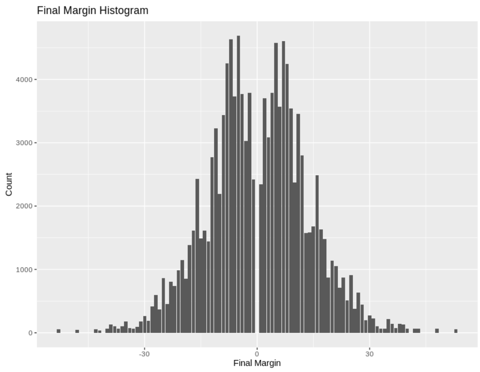
Feature	Feature.Description
GAME_ID	A unique ID for each game
MATCHUP	Shows the date of the match and the teams the match is between
LOCATION	A - Away, H - Home. Shows the location of the match with respect to the first team in the match up column
W	Win or Loss. W means win and L means Loss, with respect to the first team in the match up column
FINAL_MARGIN	Points difference between the teams at the end of the game
SHOT_NUMBER	To be read in conjunction with the PERIOD column. Indicates the shot number in a given game period
PERIOD	Indicates the game period
GAME_CLOCK	Time elapsed since the period commenced. This datset shows the time at which the shot was attempted. Max 12 minutes per period
SHOT_CLOCK	The length of time for a given shot in seconds. Max - 24 seconds is a rule
DRIBBLES	The number of times the ball was dribbled before the shot was attempted
TOUCH_TIME	The length of time a player touched the ball
SHOT_DIST	The distance from which a shot was attempted. Distance in feet
PTS_TYPE	Points awarded if a shot was made. 2 pointer or 3 pointer shots
SHOT_RESULT	Indicates whether the shot was made or missed
CLOSEST_DEFENDER	Shows the name of the player that was the closest defender
CLOSEST_DEFENDER_PLAYER_ID	Unique ID of the closest defender
CLOSE_DEF_DIST	Distance of the closest defender in feet
FGM	An abbreviation for FIELD GOALS MADE. A proxy for SHOT_RESULT, 0 indicates missed shot and 1 indicates shot made
PTS	Points awarded for shots made
player_name	Name of the player who attempted the shot
player_id	Unique ID of the player who attempted the shot

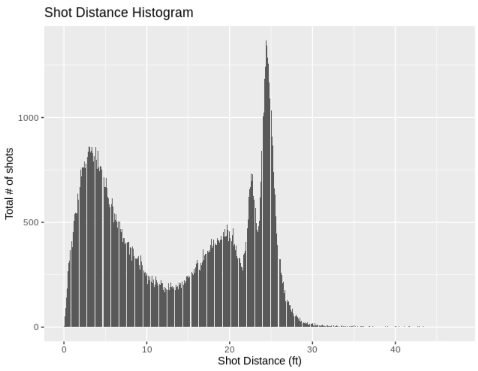
Plots

We attempt to further understand the data using the following plots. A histogram of the “closest defender distance” shows that a majority of the shots were defended from within 5 feet of the player attempting the shot and it is safe to say that more than 90% of the shots were defended from within 10 feet. The “dribbles count” shows that most of the shots were attempted soon after getting the ball and that more than 80% of the shots were attempted within 3 dribbles. “Final Margin” histogram shows that most matches were won or lost within a 15 point margin. The “Shot distance” histogram shows that most of the shots were attempted from “top of the key” and followed by 2 to 4 feet range from the basket









DATA PREPARATION

SHOT_CLOCK

Looking at the data, some of the NA values need to be dealt with. The “SHOT_CLOCK” column has some NA values and the assumption is that the SHOT_CLOCK was equal to the GAME_CLOCK and therefore it may not be recorded. For such cases, the GAME_CLOCK is assumed to be equal to SHOT_CLOCK.

```
cleanData <- initialData
gameClock <- as.vector(second(fast_strptime(cleanData$GAME_CLOCK, "%M:%S"))) +
  as.vector(minute(fast_strptime(cleanData$GAME_CLOCK, "%M:%S"))) * 60
shotClock <- is.na(initialData$SHOT_CLOCK)
for(i in 1:length(gameClock)){
  if(shotClock[i] & gameClock[i] < 25){
    cleanData$SHOT_CLOCK[i] <- gameClock[i]
  }
}
```

Names

To further handle player names in this exercise, all names are standardized to read as “First Name” followed by “Last Name”. A custom function was written to achieve this result.

```
nameformatreverse <- function(s) {
  fname <- str_extract(s, "^\\w+")
  lname <- str_extract(s, "\\w+$")
  s <- paste(lname, fname, sep = ", ")
}
```

All Shooter & Defender names are then put through the function to standardize names

```
shooterName <- cleanNoNAData$player_name
shooterName <- toupper(shooterName)
shooterName <- nameformatreverse(shooterName)
```

```
cleanNoNAData$player_name <- shooterName
cleanNoNAData$CLOSEST_DEFENDER <- toupper(cleanNoNAData$CLOSEST_DEFENDER)
cleanNoNAData$CLOSEST_DEFENDER <- gsub("[.]", "", cleanNoNAData$CLOSEST_DEFENDER)
```

Game Clock

It makes best sense to have the GAME_CLOCK expressed in seconds.

```
cleanNoNASecondsClockData <- cleanNoNAData
cleanNoNASecondsClockData$GAME_CLOCK <-
  as.vector(second(fast_strptime(cleanNoNAData$GAME_CLOCK, "%M:%S"))) +
  as.vector(minute(fast_strptime(cleanNoNAData$GAME_CLOCK, "%M:%S"))) * 60
```

Touch time

Any row that has TOUCH_TIME less than 0.1 seconds is not right and hence are omitted

```
cleanNoNASecondsClockData <- cleanNoNASecondsClockData[cleanNoNASecondsClockData$TOUCH_TIME > 0, ]
```

MODELLING

K-Means Clustering

The Elbow method is a popular, non computation intensive process of determining the most optimal number of clusters for a dataset by looking at a dropoff of variance. The other methods, eg, Bayesian Inference which we ran is more computation intensive, and produced optimal clusters that didnt agree with the visual Elbow method.Therefore, after plotting and analyzing a few different features against each other and highlighting the clusters by colouring the datapoints, we find that 3 clusters is likely the best compromise for the important features, namely, shot distance and closest defender distance.

To perform clustering, only the numeric columns from the data are selected.

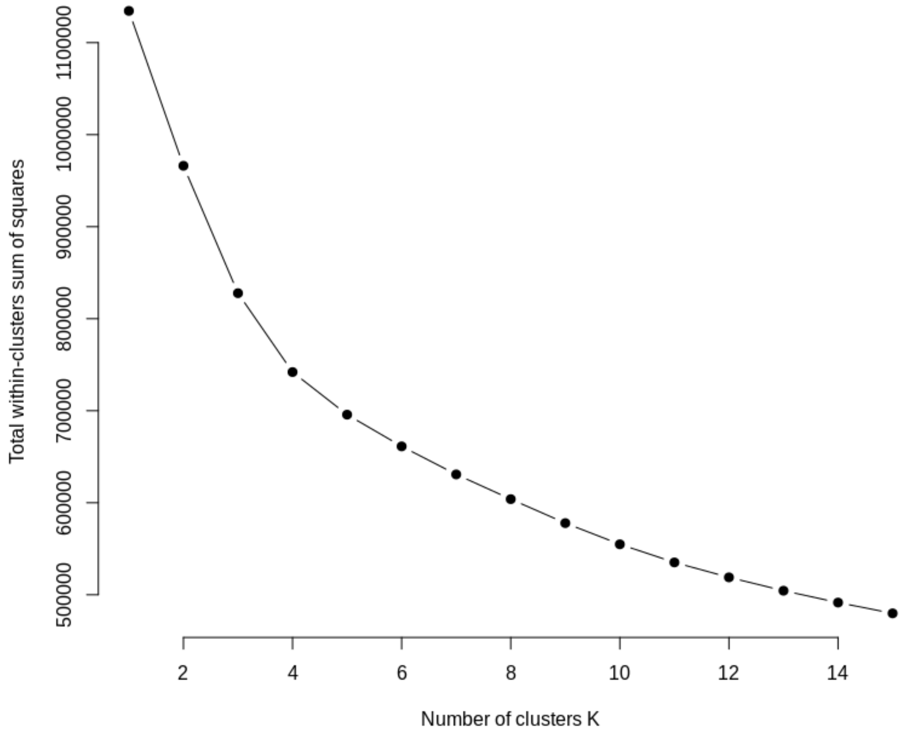
```
kdataunscaled <- cleanNoNASecondsClockData[, c("SHOT_NUMBER", "PERIOD",
                                               "GAME_CLOCK", "SHOT_CLOCK", "DRIBBLES",
                                               "TOUCH_TIME", "SHOT_DIST", "CLOSE_DEF_DIST")]
kdata <- scale(kdataunscaled)
```

We use the ‘Elbow’ method to determine the ideal number of clusters

```
set.seed(123)
# Compute and plot wss for k = 1 to k = 15
k.max <- 10
```

```
wss <- sapply(1:k.max, function(k){kmeans(kdata, k, nstart=50,iter.max = 15 )$tot.withinss})
wss
plot(1:k.max, wss,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
```

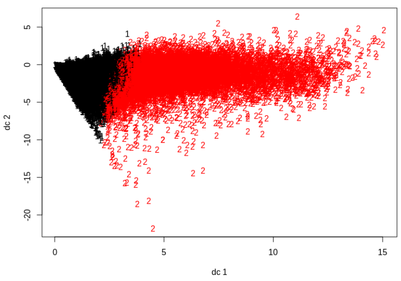
The “number of clusters” vs “sum of squares” plot helps us identify the ‘Elbow’ and decide on the right number of clusters. From the plot, we will try creating clusters with k = 2,3 & 4

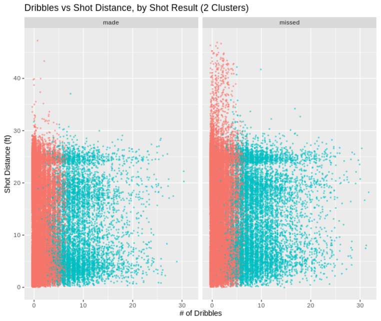


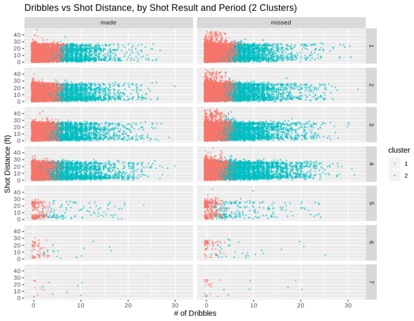
Bayesian Inference Criterion for k means to validate choice from Elbow Method

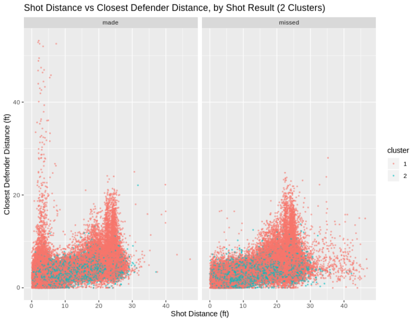
```
d_clust <- Mclust(as.matrix(kdata), G=1:10,
                  modelNames = mclust.options("emModelNames"))
d_clust$BIC
plot(d_clust)

# Let us apply kmeans for k=2 clusters
kmm.2 <- kmeans(kdata, 2, nstart = 50, iter.max = 15)
# Let us apply kmeans for k=3 clusters
kmm.3 <- kmeans(kdata, 3, nstart = 50, iter.max = 15)
# Let us apply kmeans for k=3 clusters
kmm.4 <- kmeans(kdata, 4, nstart = 50, iter.max = 15)
# We keep number of iter.max=15 to ensure the algorithm converges and nstart=50 to
# Ensure that atleast 50 random sets are choosen
kmm.2
kmm.3
kmm.4

# Plot the clusters
clusplot(kdataunscaled, kmm.3$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)

# Centroid Plot against 1st 2 discriminant functions
plotcluster(kdataunscaled, kmm.2$cluster)
plotcluster(kdataunscaled, kmm.3$cluster)
plotcluster(kdataunscaled, kmm.4$cluster)
```

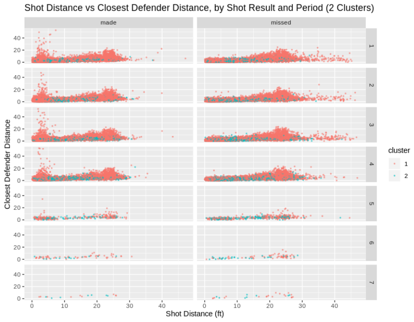
K2 plots



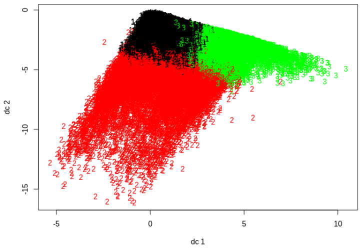


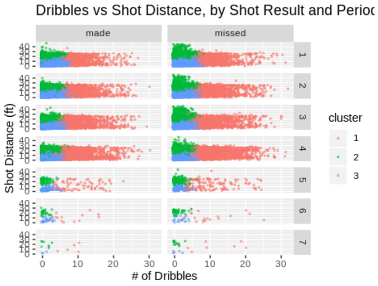


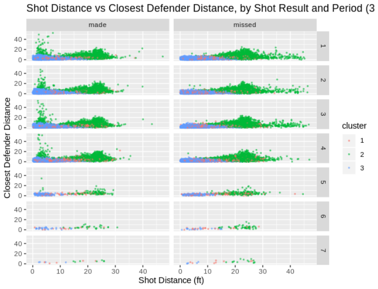


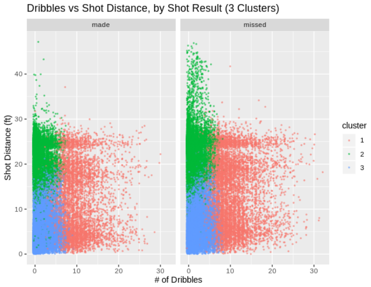


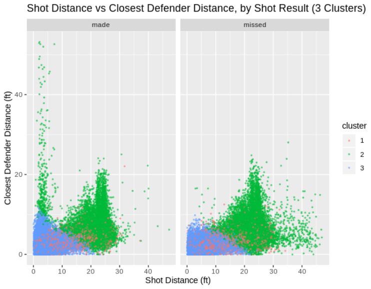
k3 plots


dc 1
dc 2
-5
0
5
10
0
-5
-10
-15

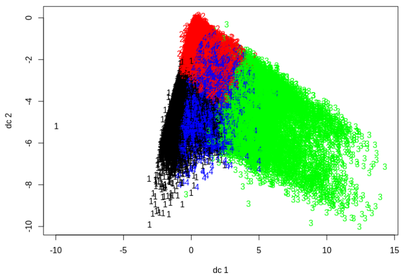

Dribbles vs Shot Distance, by Shot Result and Perioc
made
missed
Shot Distance (ft)
# of Dribbles
cluster
1
2
3
1
2
3
4
5
6
7
0
10
20
30
40

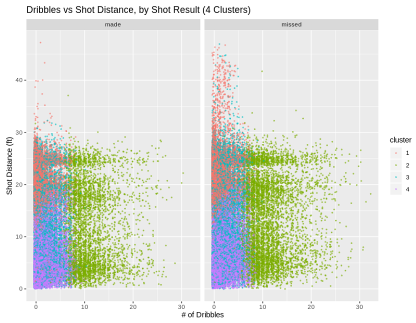

Shot Distance vs Closest Defender Distance, by Shot Result and Period (3
made
missed
Closest Defender Distance
Shot Distance (ft)
cluster
1
2
3
1
2
3
4
5
6
7
0
10
20
30
40

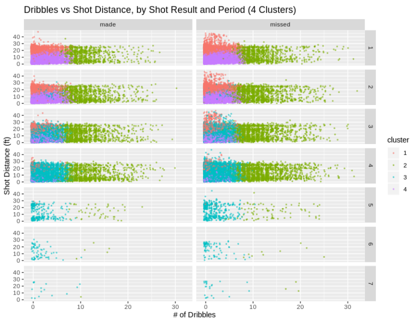

Dribbles vs Shot Distance, by Shot Result (3 Clusters)
made
missed
Shot Distance (ft)
# of Dribbles
cluster
1
2
3
0
10
20
30
40

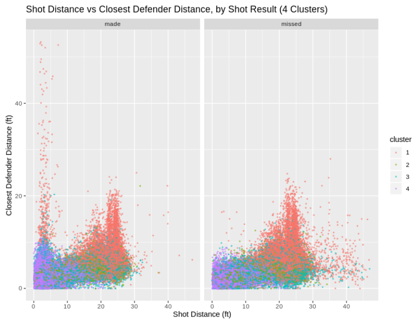

Shot Distance vs Closest Defender Distance, by Shot Result (3 Clusters)
made
missed
Closest Defender Distance (ft)
Shot Distance (ft)
cluster
1
2
3
0
10
20
30
40

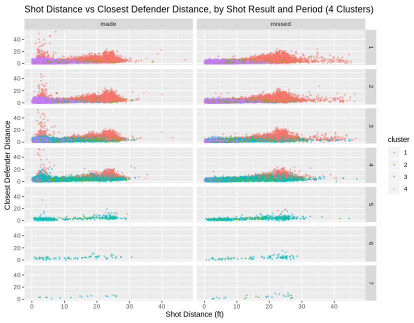
k4 plots







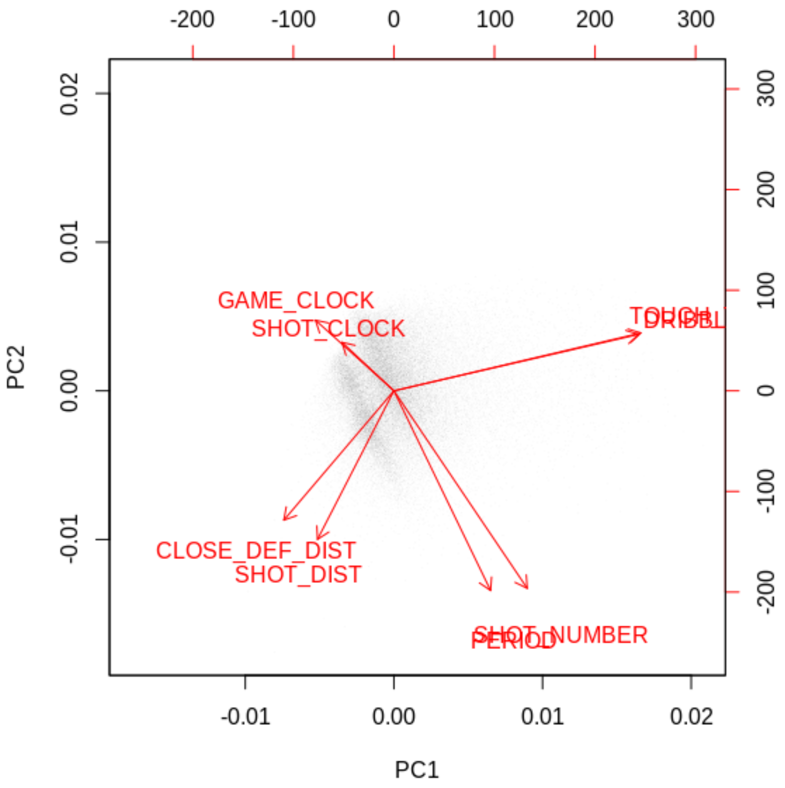




We chose a number of supervised machine learning models to predict the number of field goals made. We ran, evaluated, and compared the performance of the following models: * Decision Tree * Logistic Regression * GBM * GBM with PCA Before running the models, we converted the predictors and response variable to their correct data type. For example, numeric factors such as SHOT_DIST were explicitly converted into numeric, while categorical factors such as FGM were explicitly converted into categorical factors. Because the binary response variables of, 0 and 1, showcased a relatively balanced dataset, with the split being roughly 55%/45%, we did not need to undersample or oversample the data.

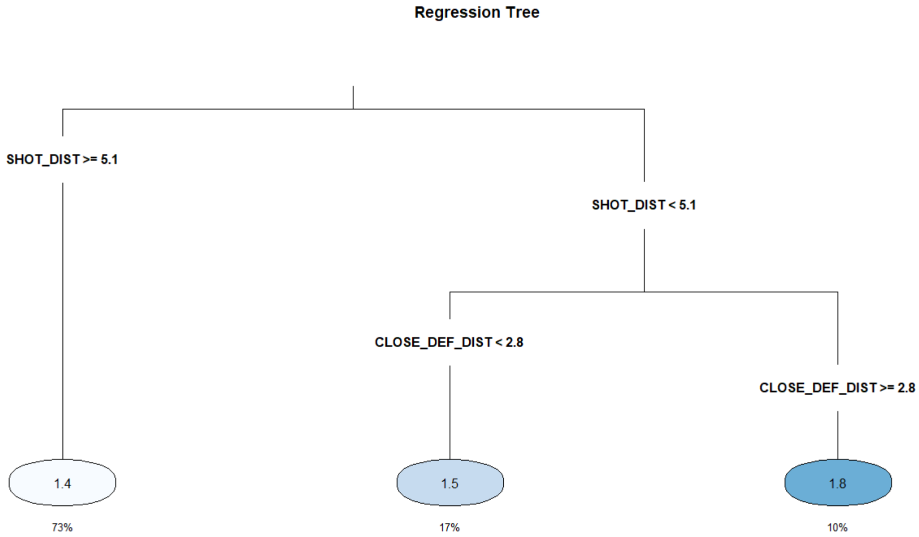
We also split the dataset into a training, and testing set. The split was set at 70% training, and 30% training, ensuring an equal

Principal Component Analysis

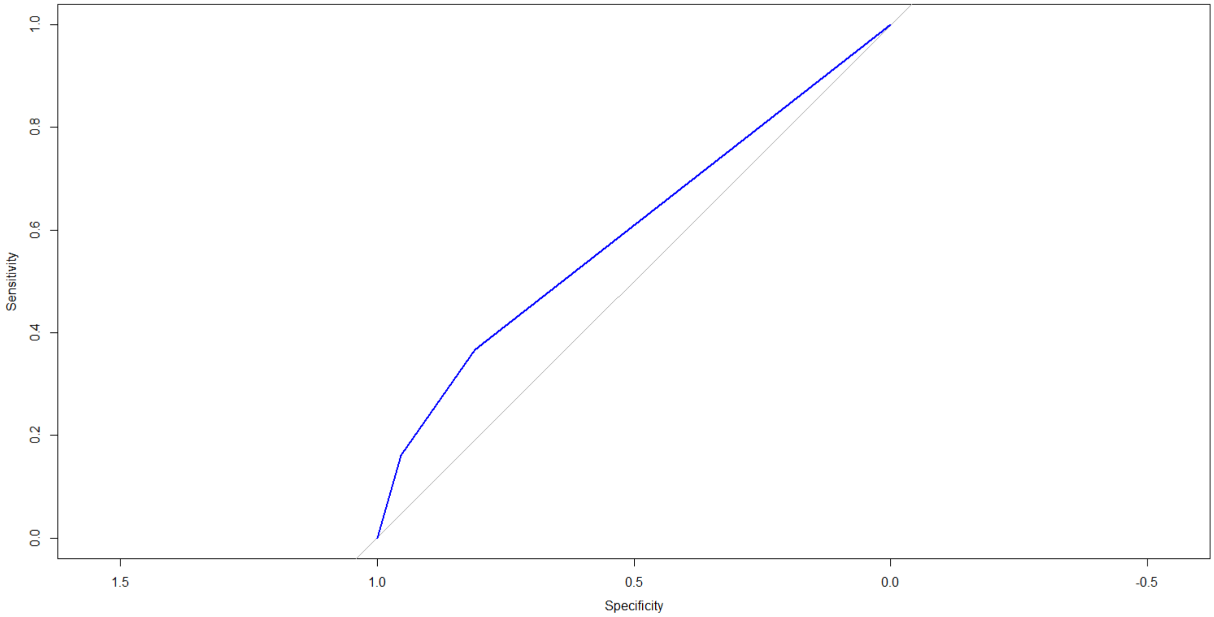


Decision Tree

The Decision tree algorithm is an algorithm that uses a tree-like data structure to make either predictions for regression, or classification problems. Given the business problem and context, a categorical variable decision tree was chosen as we wanted to classify, given the available data, whether or not an attempted shot made became a FGM (Field Goal Made); in this particular situation, there would be two categories for the response variable: either 0, denoting an attempted shot that missed, or 1, denoting an attempted shot that resulted in a field goal. A decision tree is suitable supervised machine learning algorithm because it is fairly easy to explain and visualize. For example, the following figure below is the generated decision tree diagram. Shot Distance, titled as SHOT_DIST, as well as the distance to the closest defender, titled as CLOSE_DEF_DIST, were the two most important variables in the decision tree, and of which the decisions are based upon.

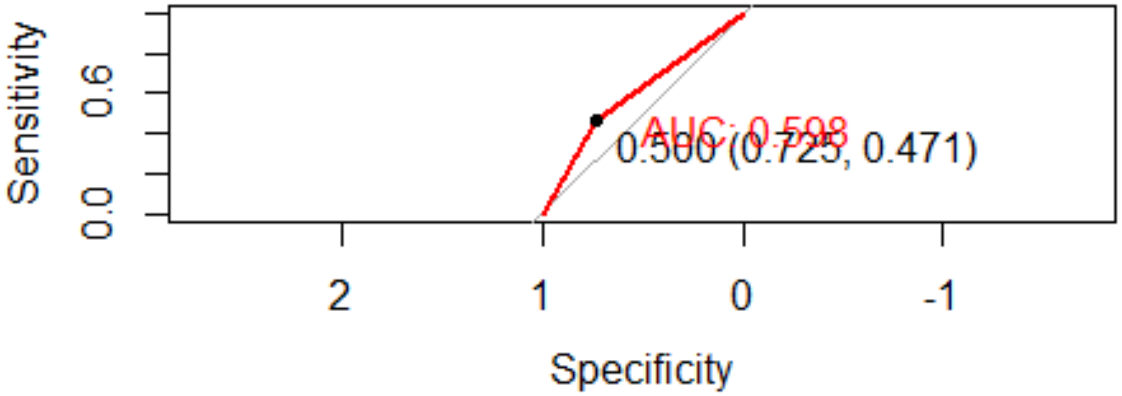


The ROC curve, and AUC of the decision tree algorithm showed similar results to the other models, as seen in the figure below. Nevertheless, it was decided not to use the decision tree algorithm as the model only showed 3 raw probabilities given the different permutations of SHOT__DIST and CLOSE__DEF__DIST. This would not have looked good on the Shiny app, as we wanted to show more probabilities on a more granular level, as there were other predictor variables that were not being used in the final model.



Logistic Regression

Logistic Regression is a parametric model used for binary classification, and it is based off the logit (logistic) function, hence the name. There are a few assumptions that were made with logistic regression: * There is a large enough dataset to make accurate predictions * Observations are independent of one another * Response variable is binary (0 or 1). * Predictor variables are related to logit function * Minimal multicollinearity among the predictor variables All assumptions except (4) and (5) are true, and due to the lack of time on our part, we were unable to ascertain whether the last two assumptions were, indeed, true. The results for the logistic regression showed similar results to other models. Listed below is the ROC curve/AUC results of the logistic regression model.



In terms of other metrics like accuracy, balanced accuracy, sensitivity, and specificity, the logistic regression model also showed similar results to the other models. Unfortunately, logistic regression took the longest amount of time to train compared to all the other models, at 2.5 hours. Listed below are the other metrics, as well as the runtime of the logistic regression model.

```
Confusion Matrix and Statistics

          Reference
Prediction    no   yes
       no  15018  9051
       yes  5695  8053

               Accuracy : 0.6101
                 95% CI : (0.6051, 0.615)
    No Information Rate : 0.5477
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1993

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.7251
            Specificity : 0.4708
         Pos Pred Value : 0.6240
         Neg Pred Value : 0.5858
             Prevalence : 0.5477
         Detection Rate : 0.3971
```

```
   Detection Prevalence : 0.6365
      Balanced Accuracy : 0.5979

       'Positive' Class : no
```

Stochastic Gradient Boosting (GBM)

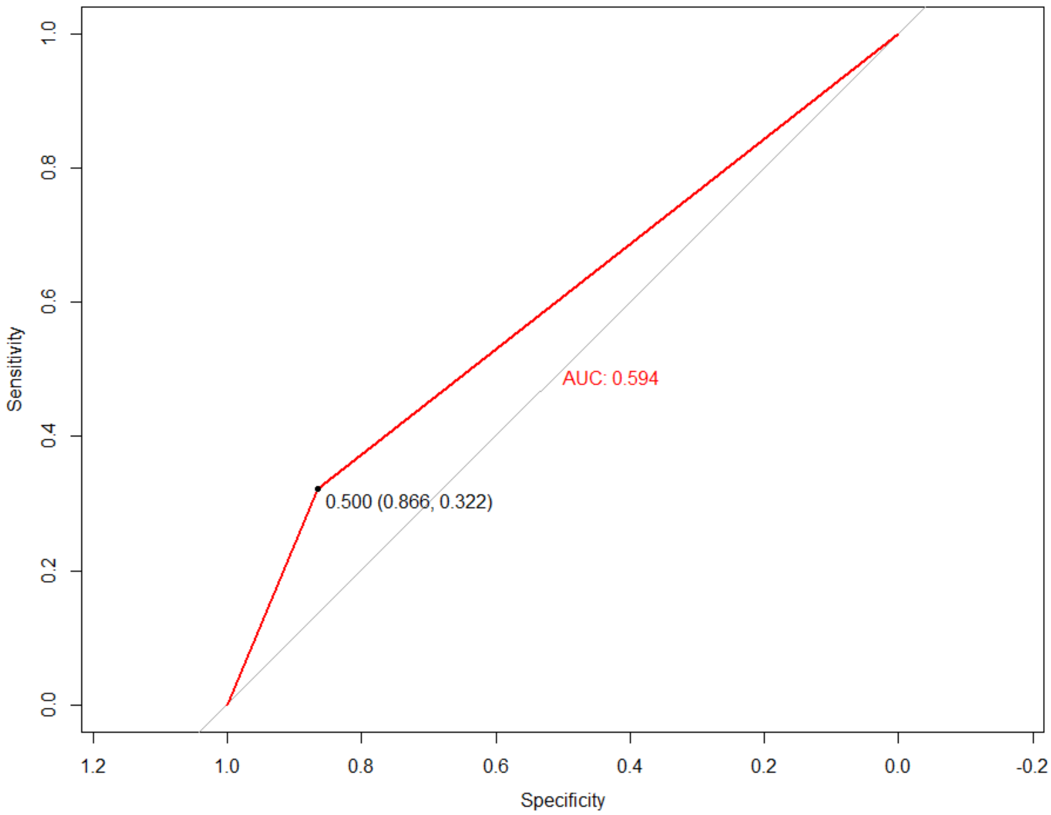
Stochastic Gradient Boosting is a relatively complex supervised learning algorithm. The algorithm continuously iterates through several trees, at one at a time, so that it can boost the performance of its weakest learners.

When initially training the GBM model, we first decided to keep CLOSEST_DEFENDER_PLAYER_ID, and player_name, when training the GBM model, but the model itself was fairly big, and in terms of variable importance, both features were not that important unless if the players had a large enough data sample to draw meaningful conclusions from. For example, looking at the figure of output variable importance, one could see the top CLOSEST_DEFENDER and player_name names were highly regarded players from the 2015 season, or at least players who had a lot of playtime.

```
Confusion Matrix and Statistics

          Reference
Prediction    no   yes
       no  17477 11287
       yes  2703  5362

               Accuracy : 0.6201
                 95% CI : (0.6152, 0.6251)
    No Information Rate : 0.5479
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.197

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8661
            Specificity : 0.3221
         Pos Pred Value : 0.6076
         Neg Pred Value : 0.6648
             Prevalence : 0.5479
         Detection Rate : 0.4745
   Detection Prevalence : 0.7810
      Balanced Accuracy : 0.5941

       'Positive' Class : no
```

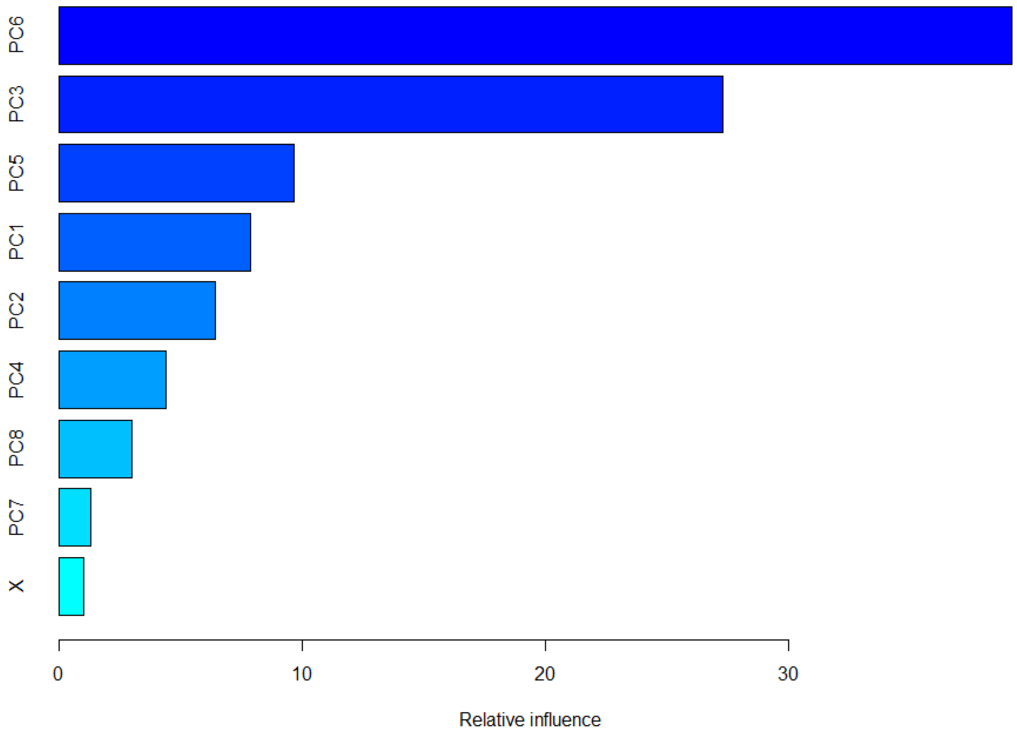
Also, including the features CLOSEST_DEFENDER_PLYAER_ID and player_name made the model size unnecessarily big for little to no improvement in the prediction of FGM. In relation to the size of the models, it also took a lot longer to train the model as well, at about 4 hours instead of the usual 1 hour. For hyper parameter optimization, we found that GBM produced the best results at with an interaction depth of 5, 40 trees, minimum of 10 observations in each node, and a shrinkage of 0.1. Using such hyper parameters offered a relatively fast training time of around 10 minutes

Listed below is the ROC curve, and AUC value for GBM without CLOSEST_DEFENDER_PLAYER_ID, and player_name predictor variables. Also listed below is the console output of training the GBM model, showing some of the metrics from the test dataset. Overall, the GBM model did the best compared to the other models, although not by a huge margin. It had an accuracy of 62.01%, a sensitivity of 0.8661 which were the highest of all the models. Unfortunately, its specificity was at the lower end compared to other models at 0.3221

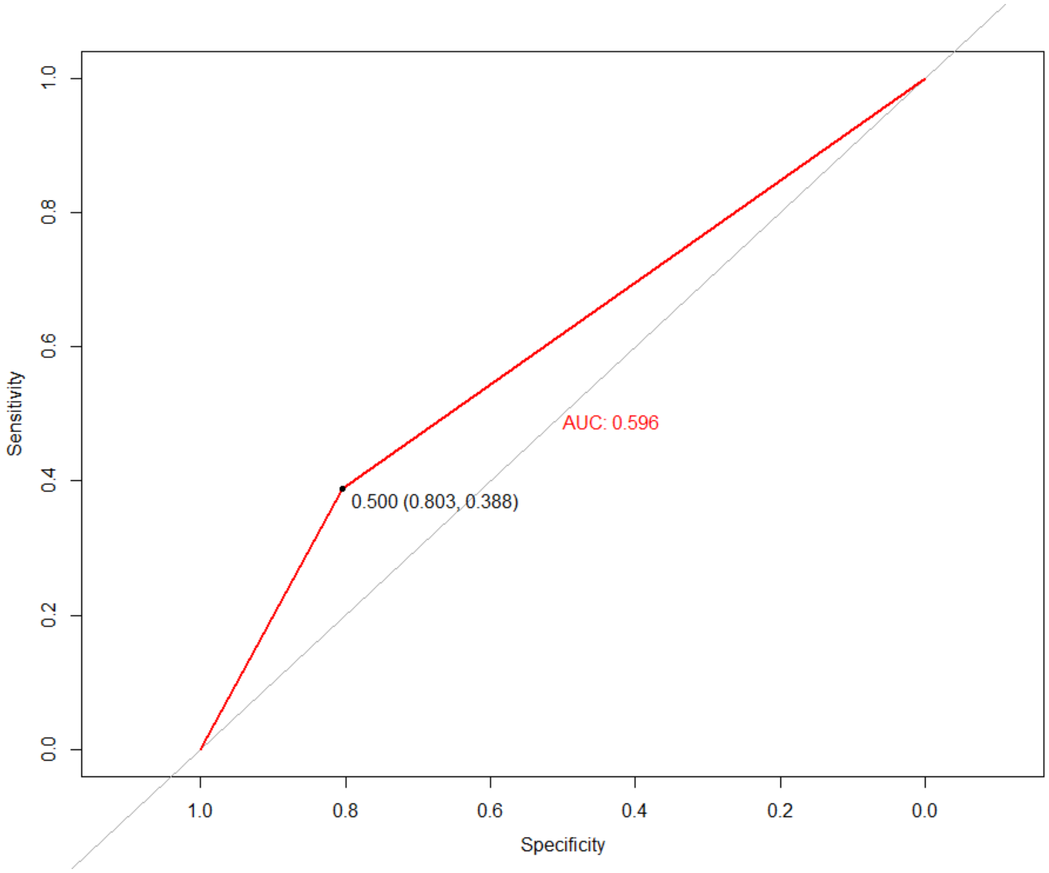


Stochastic Gradient Boosting (GBM) with PCA

We also chose to perform GBM with the results generated from PCA. We used all 8 of the components for the GBM model. Listed below was the variable importance of each component:



Listed below are the metrics of the GBM with PCA. Overall the GBM with PCA model was fairly competitive with the previous GBM model, as it scored 61.58% accuracy, and a 0.8035 sensitivity. Its specificity was a bit better than the previous GBM model at 0.3884. In terms of runtime both GBM models were fairly similar with a training duration of around 10 minutes.



```
- Fold5: shrinkage=0.1, interaction.depth=10, n.minobsinnode=10, n.trees=200
Aggregating results
Selecting tuning parameters
Fitting n.trees = 100, interaction.depth = 5, shrinkage = 0.1, n.minobsinnode = 10 on full training set
Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1        1.3695             nan     0.1000    0.0037
     2        1.3625             nan     0.1000    0.0035
     3        1.3568             nan     0.1000    0.0028
     4        1.3518             nan     0.1000    0.0025
     5        1.3474             nan     0.1000    0.0021
     6        1.3440             nan     0.1000    0.0017
     7        1.3408             nan     0.1000    0.0015
     8        1.3378             nan     0.1000    0.0013
     9        1.3350             nan     0.1000    0.0013
    10        1.3328             nan     0.1000    0.0010
    20        1.3178             nan     0.1000    0.0004
    40        1.3046             nan     0.1000    0.0001
    60        1.2993             nan     0.1000    0.0000
    80        1.2965             nan     0.1000    0.0000
   100        1.2944             nan     0.1000   -0.0000

> proc.time() - ptm_rf
   user  system elapsed
 217.28    0.20  217.50
>
```

```
Stochastic Gradient Boosting

Tuning parameter 'shrinkage' was held constant at a value of 0.1

Tuning parameter 'n.minobsinnode' was held constant at a value of 10
ROC was used to select the optimal model using the largest value.
The final values used for the model were n.trees = 100, interaction.depth =
 5, shrinkage = 0.1 and n.minobsinnode = 10.


Confusion Matrix and Statistics

          Reference
Prediction    no   yes
       no  16214 10182
       yes  3966  6467

               Accuracy : 0.6158
                 95% CI : (0.6109, 0.6208)
    No Information Rate : 0.5479
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1984

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8035
            Specificity : 0.3884
         Pos Pred Value : 0.6143
         Neg Pred Value : 0.6199
             Prevalence : 0.5479
         Detection Rate : 0.4403
   Detection Prevalence : 0.7167
      Balanced Accuracy : 0.5960

       'Positive' Class : no

>
> #summary of model
> summary(model_gbm)
    var   rel.inf
PC6 PC6 39.162726
PC3 PC3 27.261858
PC5 PC5  9.652336
PC1 PC1  7.877662
PC2 PC2  6.407543
PC4 PC4  4.380920
PC8 PC8  2.955187
PC7 PC7  1.287303
X     X  1.014465
```

EVALUATION

All models performed fairly similarly in terms of metrics, such as ROC, AUC, Accuracy, Balanced Accuracy, Sensitivity, and Specificity. In terms of overall performance, the models were not spectacular: they showed an accuracy, balanced accuracy, and AUC of around 0.60, sensitivity of around 70% to 80%, and a specificity of 30% to 40%. Nevertheless, there were some positives to be noted: although the model was only 10% better than chance, the sensitivity of all models was quite high at 70% to 80%. This meant that the model was fairly good at correctly predicting the percentage of actual positive cases (i.e FGM); on the flip side, all models had fairly poor specificity, meaning the models were not so good at correctly predicting the percentage of actual negative cases (i.e. shots that missed).