PSTAT 131 Final Project Report

Calvin Nguyen (5900147)

6/2/2020

Introduction

In this project, I will be taking a data set that contains information on the shots that basketball player Kobe Bryant took throughout his career. The data describes different qualities about each shot that Kobe took and whether that shot went in or not. How can we use machine learning methods to make a prediction on whether a shot goes in or not?

Data

My data comes from the "Kobe Bryant Shot Selection" data set on Kaggle. This dataset contains observations with different variables that contribute to whether or not Kobe made a shot. In the original dataset, there are a total of 30697 observations, with 24 predictor variables and 1 response variable.

```
## 'data.frame':    30697 obs. of  25 variables:
##  $ action_type       : Factor w/ 57 levels "Alley Oop Dunk Shot",..: 27 27 27 27 6 27 28 27 27 42 ..
##  $ combined_shot_type: Factor w/ 6 levels "Bank Shot","Dunk",..: 4 4 4 4 2 4 5 4 4 4 ...
##  $ game_event_id     : int  10 12 35 43 155 244 251 254 265 294 ...
##  $ game_id           : int  20000012 20000012 20000012 20000012 20000012 20000012 20000012 20000012 2
##  $ lat               : num  34 34 33.9 33.9 34 ...
##  $ loc_x             : int  167 -157 -101 138 0 -145 0 1 -65 -33 ...
##  $ loc_y             : int  72 0 135 175 0 -11 0 28 108 125 ...
##  $ lon               : num  -118 -118 -118 -118 -118 ...
##  $ minutes_remaining : int  10 10 7 6 6 9 8 8 6 3 ...
##  $ period            : int  1 1 1 1 2 3 3 3 3 3 ...
##  $ playoffs          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ season            : Factor w/ 20 levels "1996-97","1997-98",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ seconds_remaining : int  27 22 45 52 19 32 52 5 12 36 ...
##  $ shot_distance     : int  18 15 16 22 0 14 0 2 12 12 ...
##  $ shot_made_flag    : int  NA 0 1 0 1 0 1 NA 1 0 ...
##  $ shot_type         : Factor w/ 2 levels "2PT Field Goal",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ shot_zone_area    : Factor w/ 6 levels "Back Court(BC)",..: 6 4 3 5 2 4 2 2 4 2 ...
##  $ shot_zone_basic   : Factor w/ 7 levels "Above the Break 3",..: 5 5 5 5 6 5 6 6 3 3 ...
##  $ shot_zone_range   : Factor w/ 5 levels "16-24 ft.","24+ ft.",..: 1 3 1 1 5 3 5 5 3 3 ...
##  $ team_id           : int  1610612747 1610612747 1610612747 1610612747 1610612747 1610612747 161061
##  $ team_name         : Factor w/ 1 level "Los Angeles Lakers": 1 1 1 1 1 1 1 1 1 1 ...
##  $ game_date         : Factor w/ 1559 levels "1996-11-03","1996-11-05",..: 311 311 311 311 311 311 3
##  $ matchup           : Factor w/ 74 levels "LAL @ ATL","LAL @ BKN",..: 29 29 29 29 29 29 29 29 29 29
##  $ opponent          : Factor w/ 33 levels "ATL","BKN","BOS",..: 26 26 26 26 26 26 26 26 26 26 ...
##  $ shot_id           : int  1 2 3 4 5 6 7 8 9 10 ...
```

There are 10 categorical variables: action_type, combined_shot_type, shot_type, shot_zone_area, shot_zone_basic, shot_zone_range, team_name, matchup, year, and opponent. There are 13 numerical

predictors: game_event_id, game_id, lat, loc_x, loc_y, lon, minutes_remaining, period, season, seconds_remaining, shot_distance, team_id, and shot_id. The remaining predictor is a binary predictor: playoffs.

Exploratory Graphics

Next, I created different graphics and charts in order to visualize and explore the data. This helped me understand what kind of information was given by the predictor variables. These graphics can also help explain some of the collinearity in the data, as many of the predictor variables are either related or contain similar information.

```
## Warning: Removed 69 rows containing missing values (geom_point).
```

Figure 1: Shots Made(Blue) vs. Shots Missed(Red)

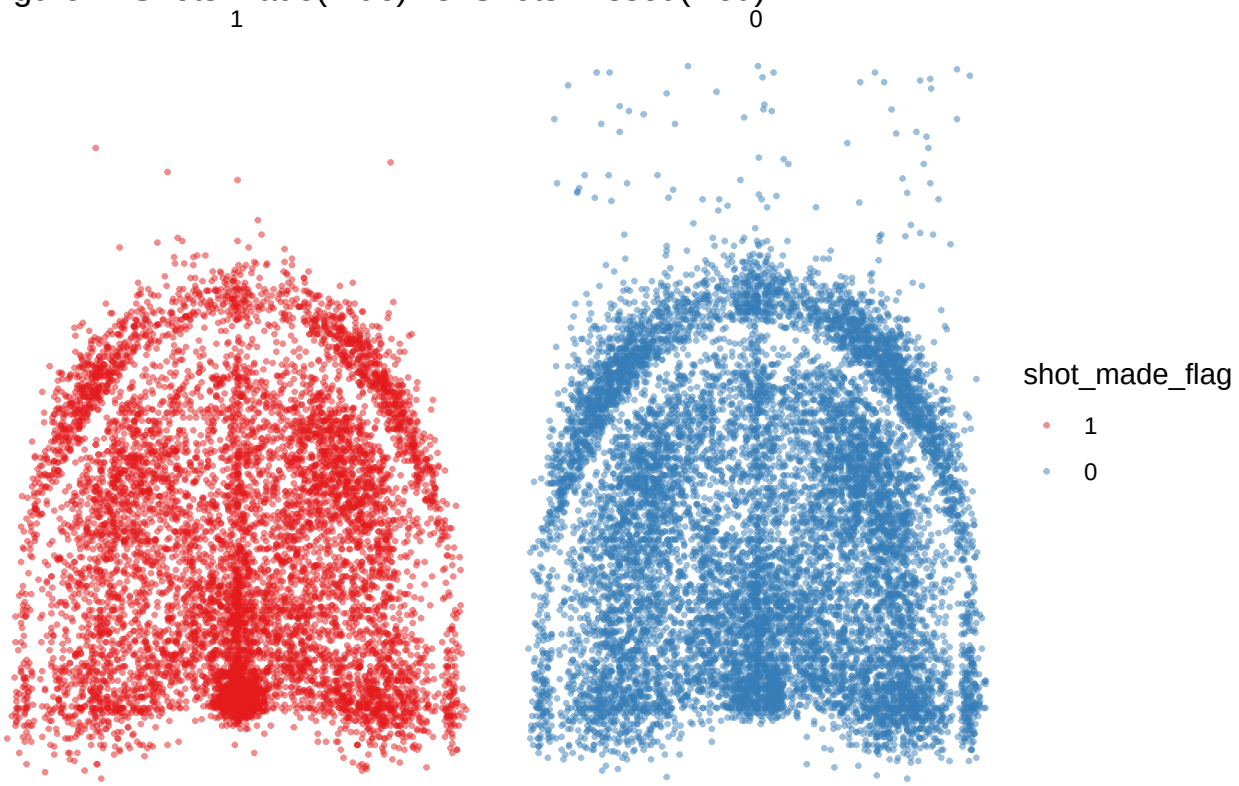


Figure 1 displays made shots and missed shots. There are 11465 makes and 14232 misses in the dataset. In examining the number of misses versus the number of makes, we can see that there is a slight class imbalance. There are many more observations of misses than makes in the data set.

```
## Warning: Removed 105 rows containing missing values (geom_point).
```

Figure 2: Shot Types

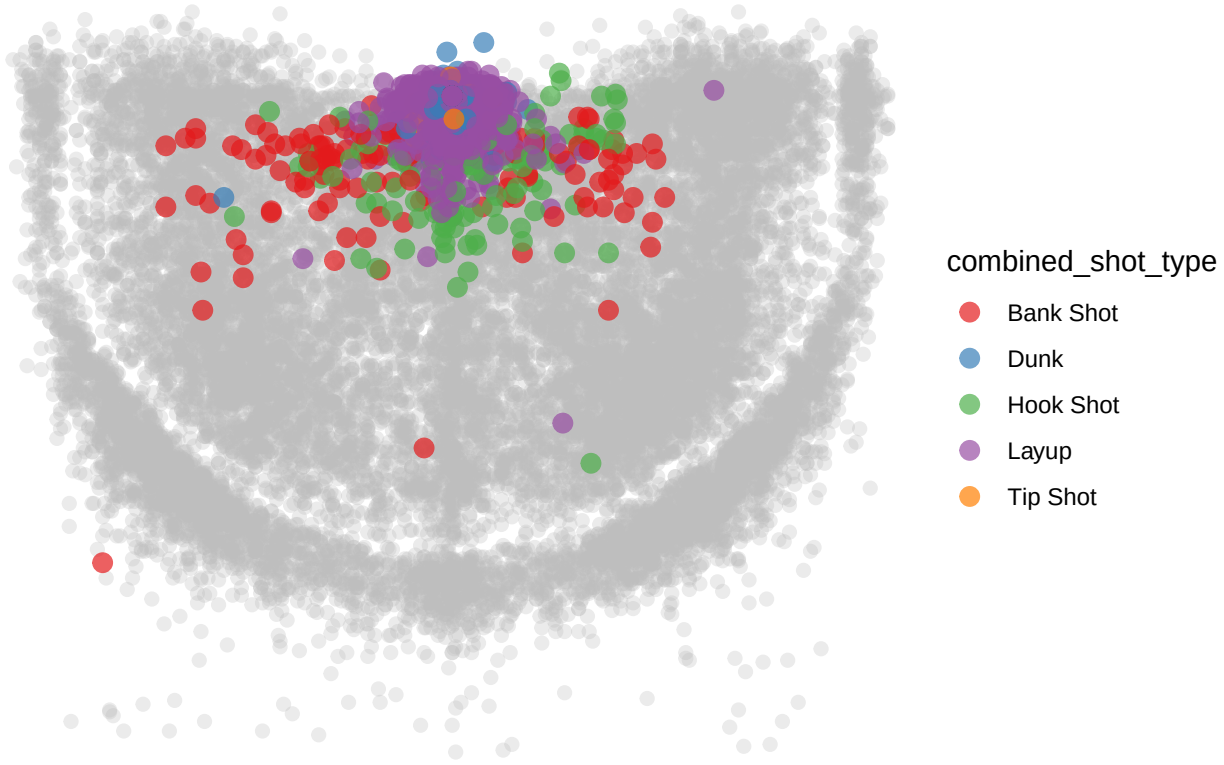


Figure 2 shows the different kinds of shots that Kobe took and their locations on the floor. Since a majority of shots taken were jump shots, the jump shots are colored grey for easier visualization. By comparing the grey plotted points (jump shots) to the other types of shots, we can see that there are significantly more jump shots in comparison to all other types of shots. From this plot, we can see that jump shots vary significantly in where they are taken from, while the rest of the shots are closer to the basket.

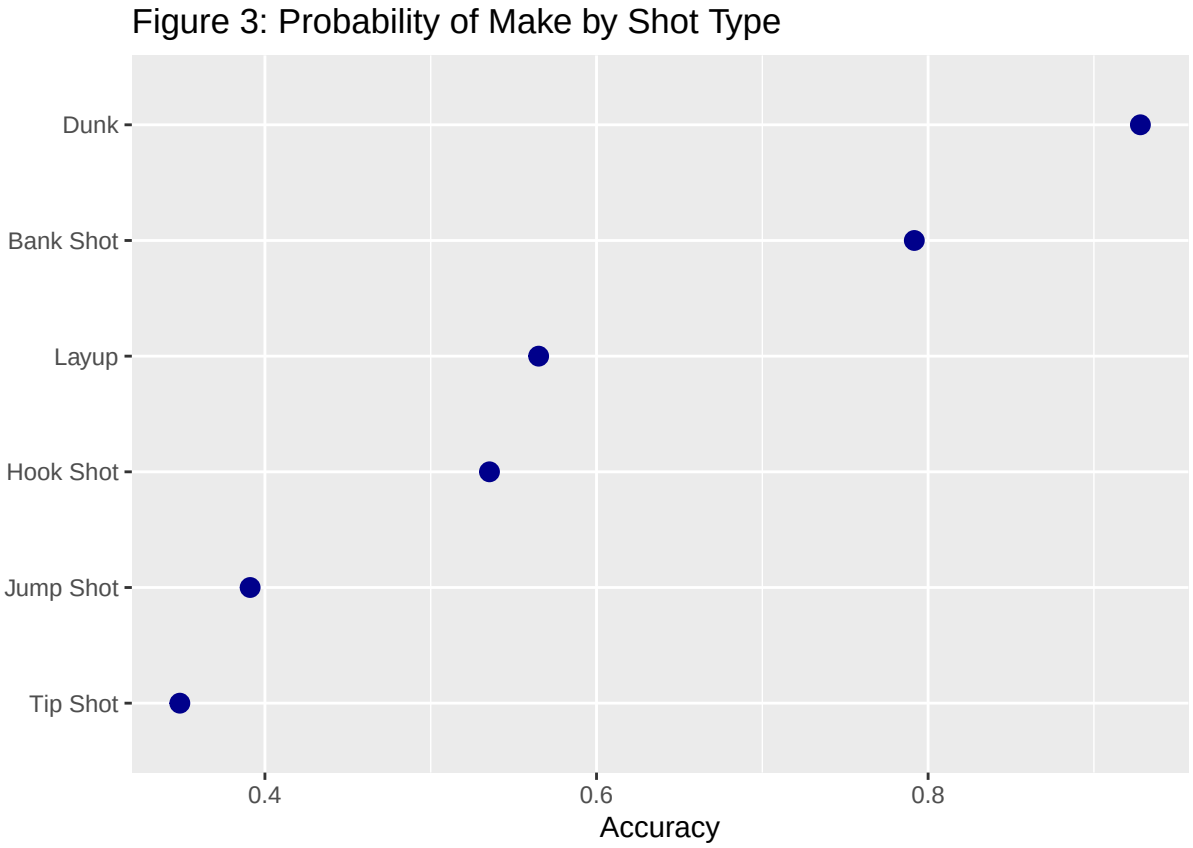


Figure 3 shows how accurate Kobe was on different types of shots, with accuracy in terms of probability of making a shot. We can see that the chances of making a shot varies a lot between shot types. This might indicate that shot type is an important predictor in predicting whether a shot goes in or not, because the outcome variable might be heavily dependent on the type of shot that is taken. We can also see that the shots that are described as dunks have much higher probabilities of being makes than other shots. This might affect our models, because there is a class imbalance within shot types. There are substantially more makes than misses for a shot that is a dunk, while there are significantly more misses than makes for shots that are jump shots.

Figure 4: Probability of Make by Shot Area

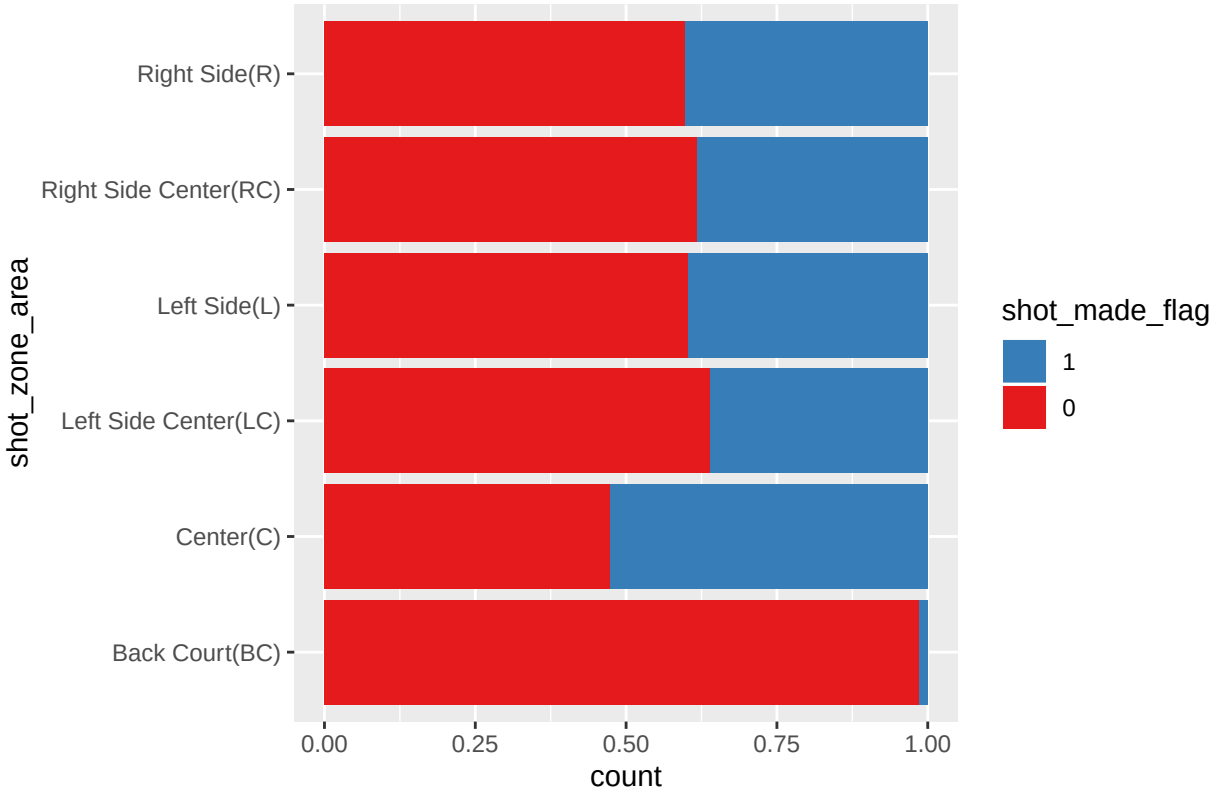
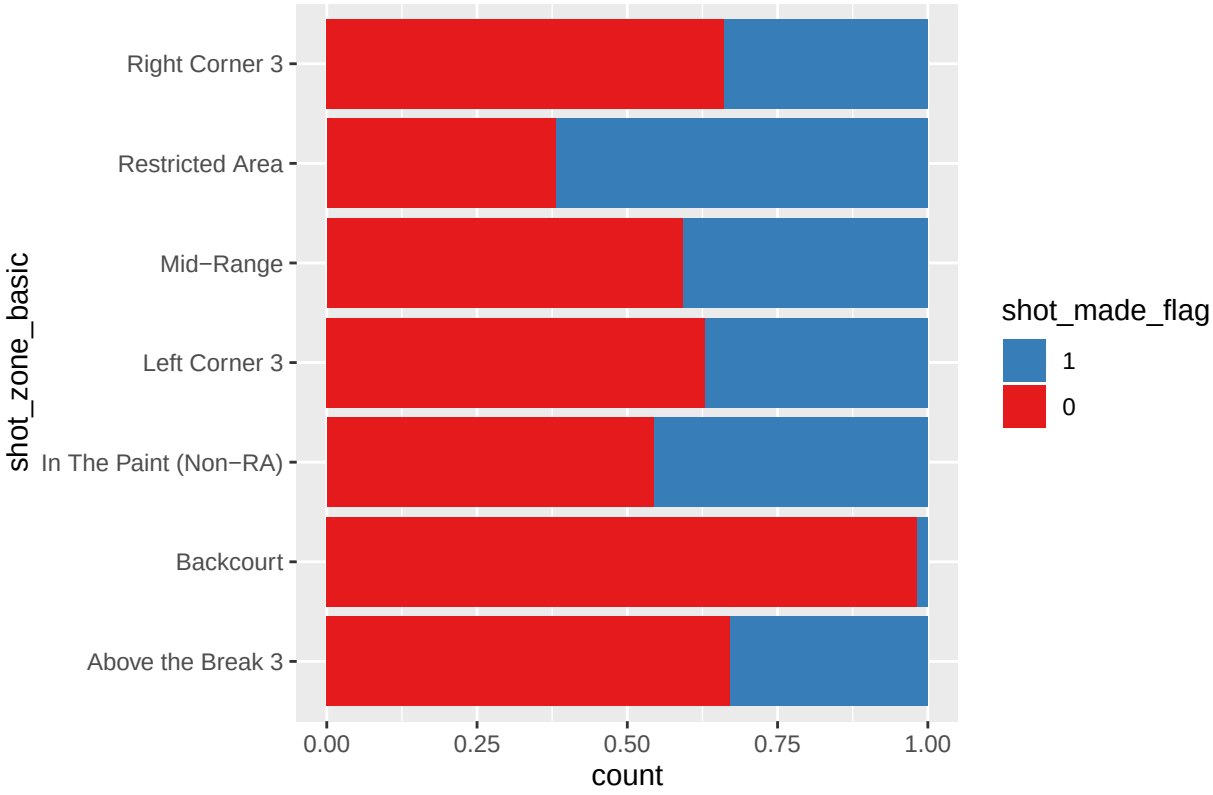
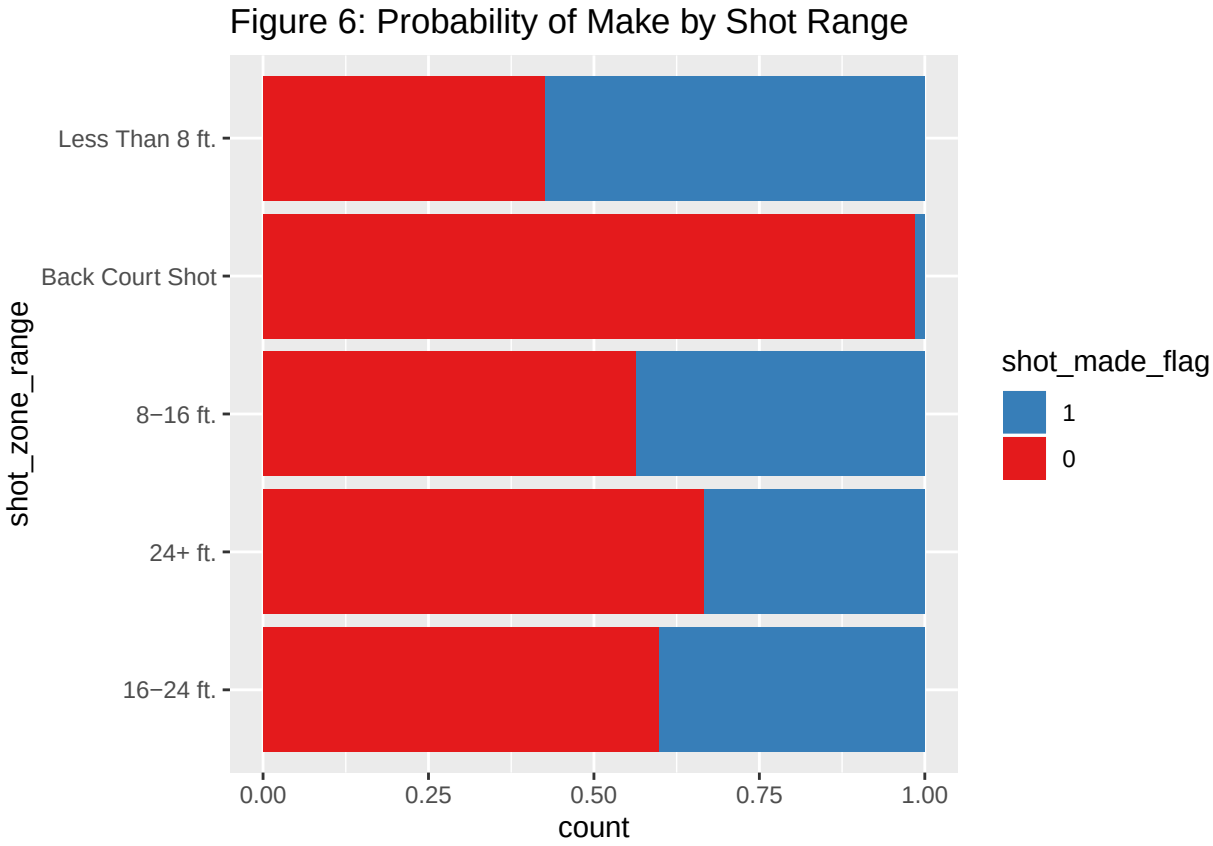


Figure 5: Probability of Make by Shot Location





Figures 4-6 show accuracy according to shot zones. It seems that these three variables contain the same amount of variability. The factors that are most variable are Back Court Shots.

Figure 7: Probability of Make by Minutes Remaining

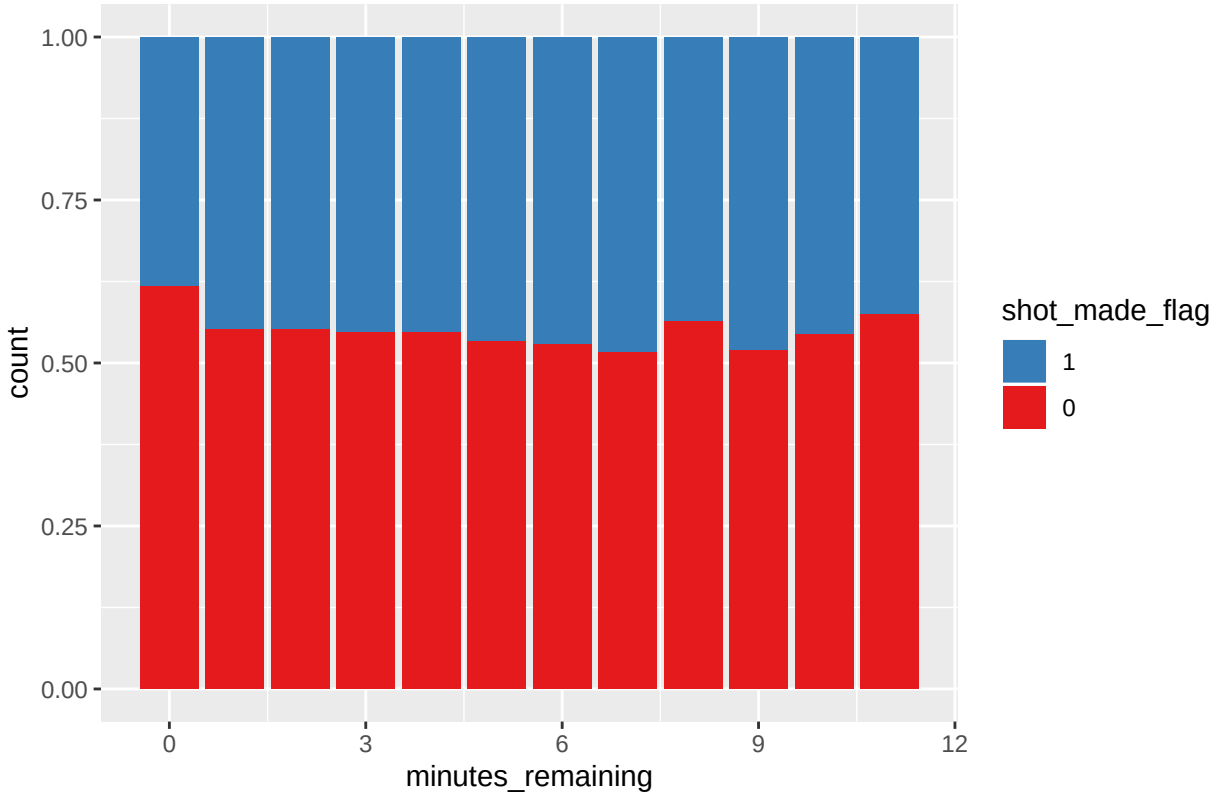


Figure 8: Probability of Make by Period

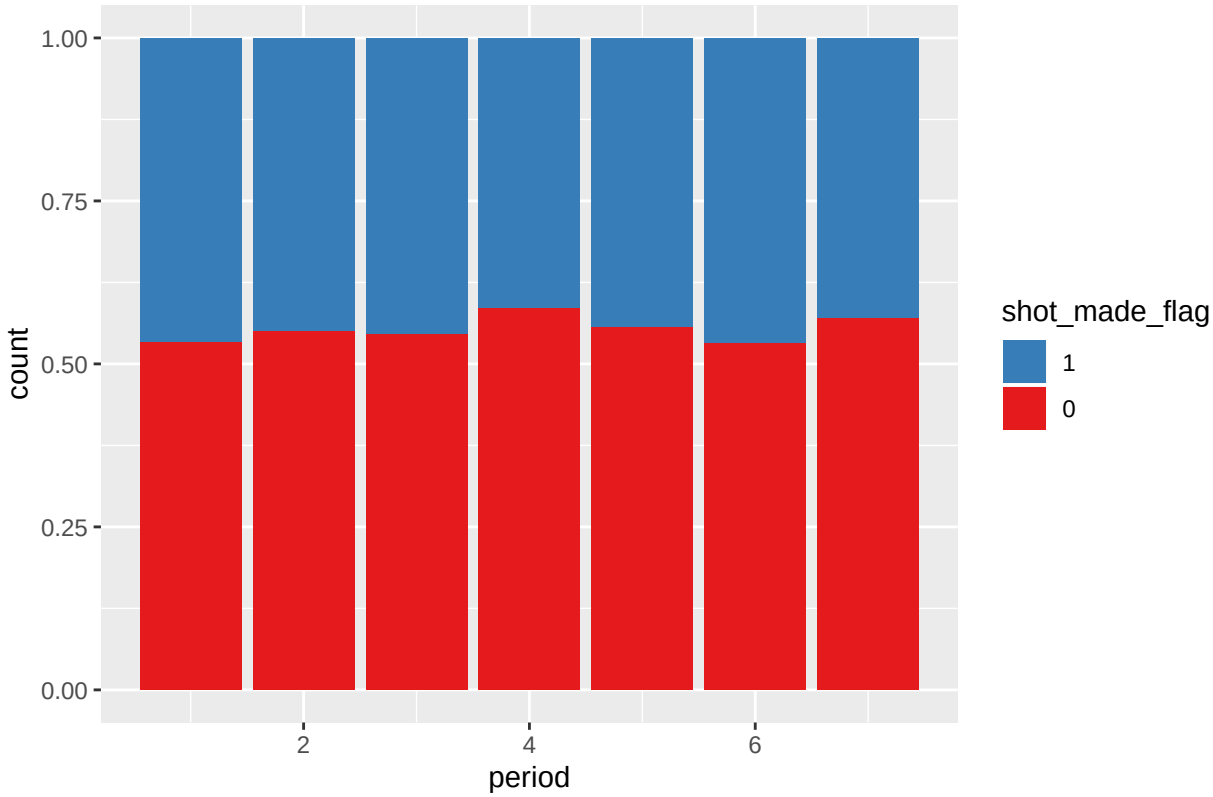
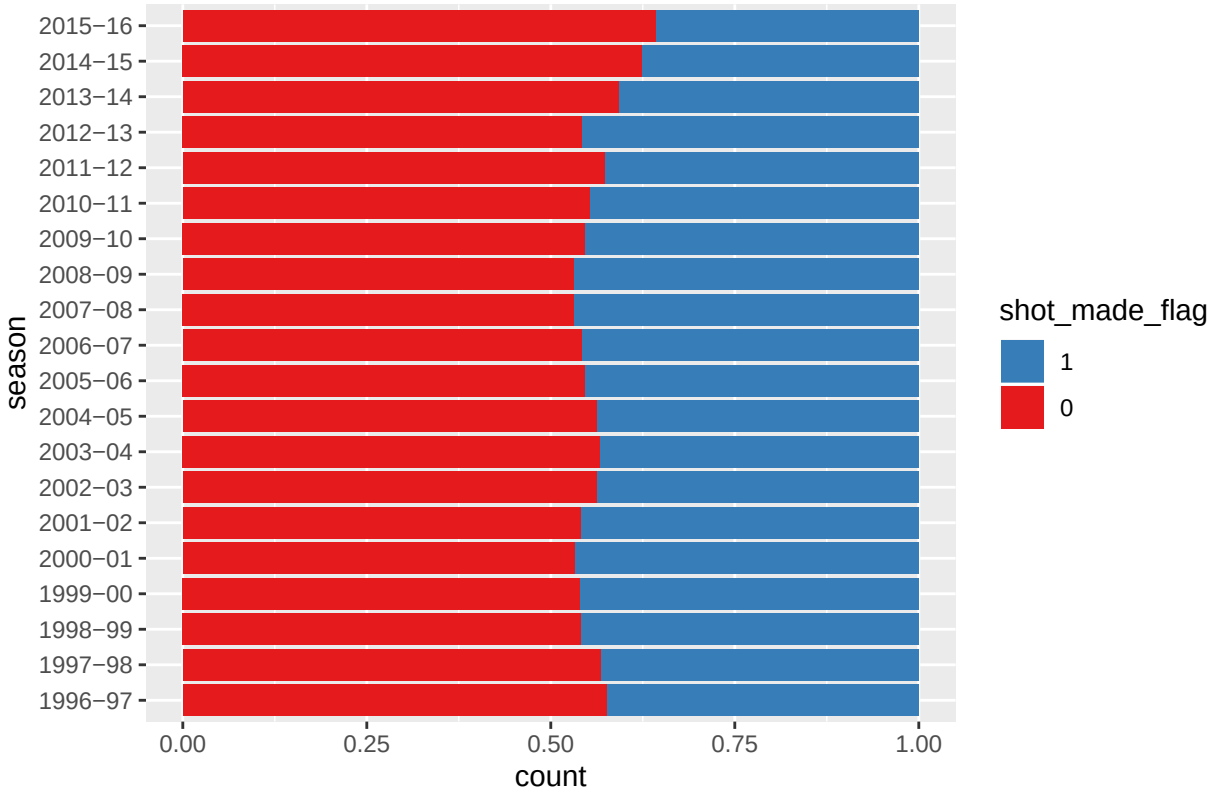


Figure 9: Probability of Make by Season



Figures 7-9 show accuracy according to minutes remaining in the game, period of the game, and the season. All of these variables relate to time, and there is very little variability in the data, so these variables might not provide adequate information to help us in our prediction.

```
## Warning: Removed 23 rows containing non-finite values (stat_count).
```

```
## Warning: Removed 3 rows containing missing values (geom_bar).
```

Figure 10: Probability of Make by Distance

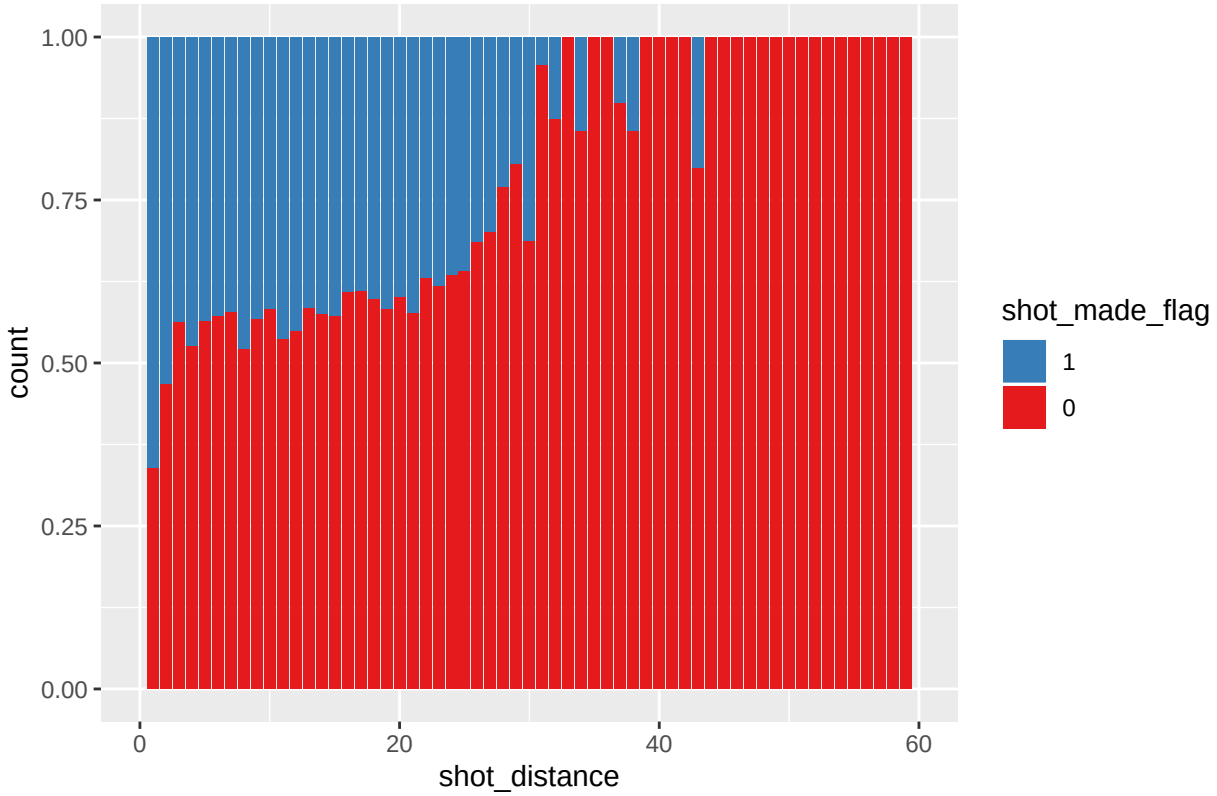


Figure 10 shows a chart describing shot accuracy by distance to the basket. We can see that there is considerable variability in this variable, so it might be a good variable to use in our prediction. As distance increases, shots tend to be less accurate, with some variability in between.

Data Completeness

Next, I explored the missingness in the dataset. I discovered that there were 5000 observations in the dataset that had missing values for the response variable shot_made_flag. I removed these 5000 observations and was left with 25697 observations.

Data Cleaning

Before I started building my models, I performed some data cleaning in order to change variables that would work with my models better and to remove unecessary variables. First, I combined the minutes_remaining and seconds_remaining variables into one variable for total time in seconds: time_remaining. I then transformed the response variable into "Make" corresponding to a 1 and "Miss" corresponding to a 0. This was to make the results of classifications more interpretable. Next, I transformed the categorical variables into dummy variables. Many of these categorical variables contained a high number of levels. For example, combined_shot_type had 6 levels. I turned each of these levels into a dummy variable. Lastly, I removed some unecessary variables. I removed game_event_id and game_id, because these were simply variables that ordered the number of games Kobe Bryant played in. With our models, using this kind of timeline variable isn't possible. The variable combined_shot_type was a more basic version of action_type, so I chose to remove action_type because it had 57 levels versus 6 levels for combined_action_type. The variables lon and lat describe the same information about shot location as loc_x and loc_y except in different measurement units, so they were removed. I removed team_id and team_name because they were the same value at every occurence. This is because Kobe played for one team his whole career, so variables relating to the team he played for did not change. Lastly, I removed matchup and opponent. These were variables describing the

teams that Kobe was playing against. This information would be useful in normal conditions; however teams change significantly throughout seasons and integrating these kinds of variables would require considerable preprocessing. I would have to use the matchup information to figure out how Kobe's percentages varied according to the team he played against and whether the team's roster in a given year affected his shooting. For that reason, I chose to not include these variables in my data set.

Methods

In this project, I will develop 3 classifiers to predict whether a shot went in or not. I will train a Classification Tree, a Logistic Regression Model, and a Random Forest Model. I divided 75% of the observations into the training set and the remaining 25% into the test set. This resulted in 19272 observations for the training set and 6425 observations for the test set.

In the Classification Tree, I will first build a complete classification tree on the training data. Next, I will use 10-fold cross-validation to choose the best level of tree complexity. This cross-validation method will give the number of terminal nodes that might lead to the lowest test error rate. I will then use this value to prune my tree and use the pruned tree to predict on the test data.

The Random Forest Model does not require cross-validation or any model selection processes.

In the Logistic Regression Model, I will first fit a logistic regression model to the training data and generate an ROC curve to measure my model's performance. Doing so will help me choose a threshold value to lower the model's error rate. After constructing the ROC curve, I will plot the False Negative Rate and False Positive Rate against the probability threshold values, ranging from 0 to 1. From this, I will select the probability threshold value that results in the smallest combined False Negative Rate and False Positive Rate. This will be determined by choosing the probability threshold value with the smallest euclidian distance between the (FNR,FPR) and (0,0) on the graph.

Classification Tree

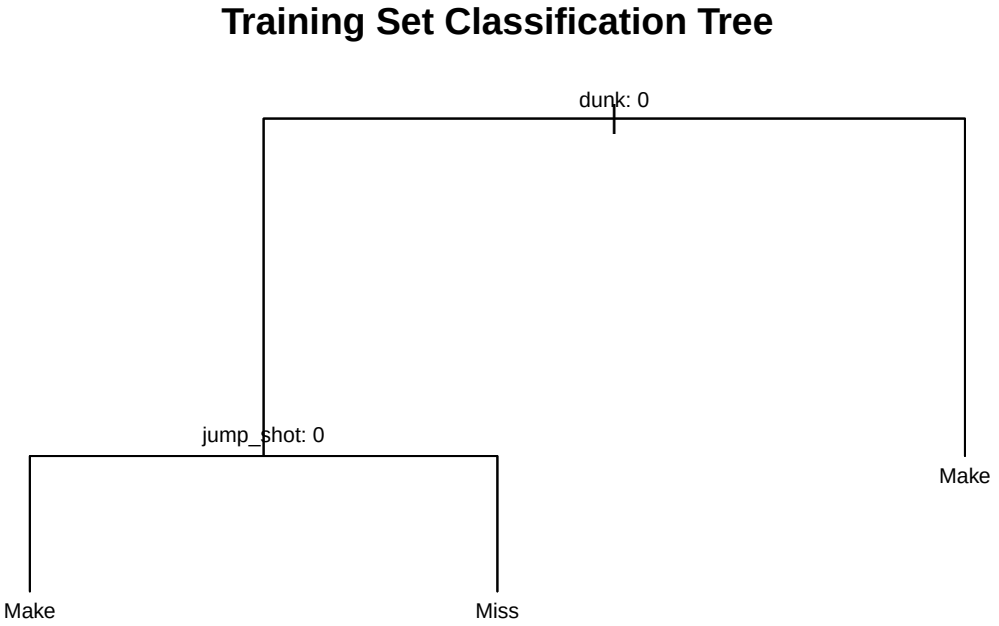
The first model I fit was a classification tree. I chose this model because my dataset contains a lot of predictor variables and a classification tree would make it easy to interpret how these variables were used to determine a classification. Classification trees divide the feature space into nonoverlapping regions. Observations in the same region are given the same classification. Splits in these trees are determined by choosing variables and cutpoint values that lead to the biggest increase in region impurity, which can be measured with a few different scores such as Classification Error, Gini Index, or Entropy.

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
##
## Classification tree:
## tree(formula = shot_made_flag ~ ., data = stats.train)
## Variables actually used in tree construction:
## [1] "dunk"      "jump_shot"
## Number of terminal nodes:  3
## Residual mean deviance:  1.32 = 25300 / 19300
## Misclassification error rate: 0.39 = 7515 / 19272
```

```
## $size
## [1] 3 2 1
##
## $dev
## [1] 7515 7953 8597
##
```

```
## $k
## [1] -Inf  438  644
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```




```
##
## yhat.test Make Miss
##      Make  989  544
##      Miss 1879 3013
```

I first fit a Decision Tree Model on my training data. This resulted in a tree with 3 terminal nodes. The first variable the tree was split on was dunk. If the shot taken was a dunk, the tree classified the observation as a make. If the shot was not a dunk, then the tree led to another split on the variable jump shot. This split then classified observations as a make if it was not a jump shot and a miss if it was a jump shot. Next, I performed 10-fold cross-validation to choose the optimal level of tree complexity and to determine if I needed to prune the tree. This method revealed that the best number of terminal nodes was 3 and so no pruning was done to the tree. Looking at the confusion matrix, the tree misclassified 1879 of makes as misses, leading to a 38.4% FNR. This is because it only took into account whether or not the shot was a jump shot or a dunk and did not use other predictor variables in classification. When used to predict on the test set observations, the test error rate was 37.7%.

Random Forest

The next model I fit was a Random Forest model. This model is an ensemble method and is an extentsion of the classification tree method. With a Random Forest, the model builds multiple trees that are not pruned. Each individual tree has high variance, but low bias. Predictions are made based on a majority vote of the prediction of each tree. When the predictions of these trees are averaged, the goal is to maintain the low bias, while decreasing variance.

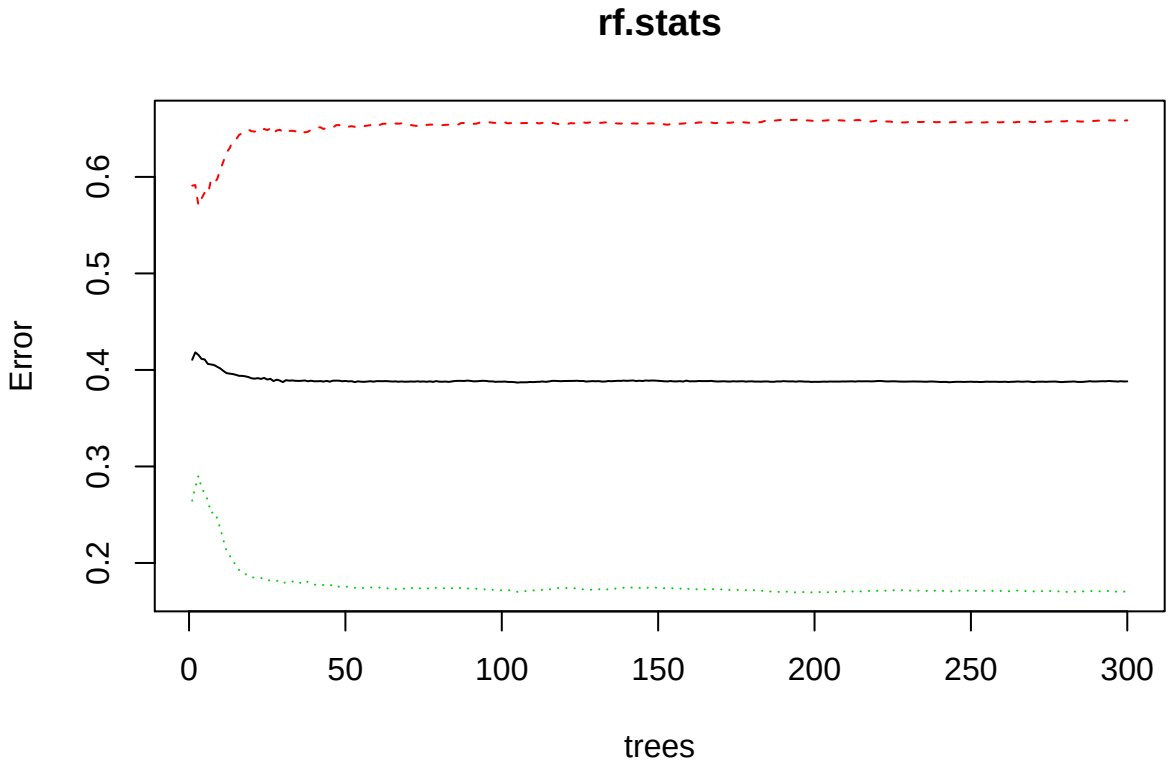
```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

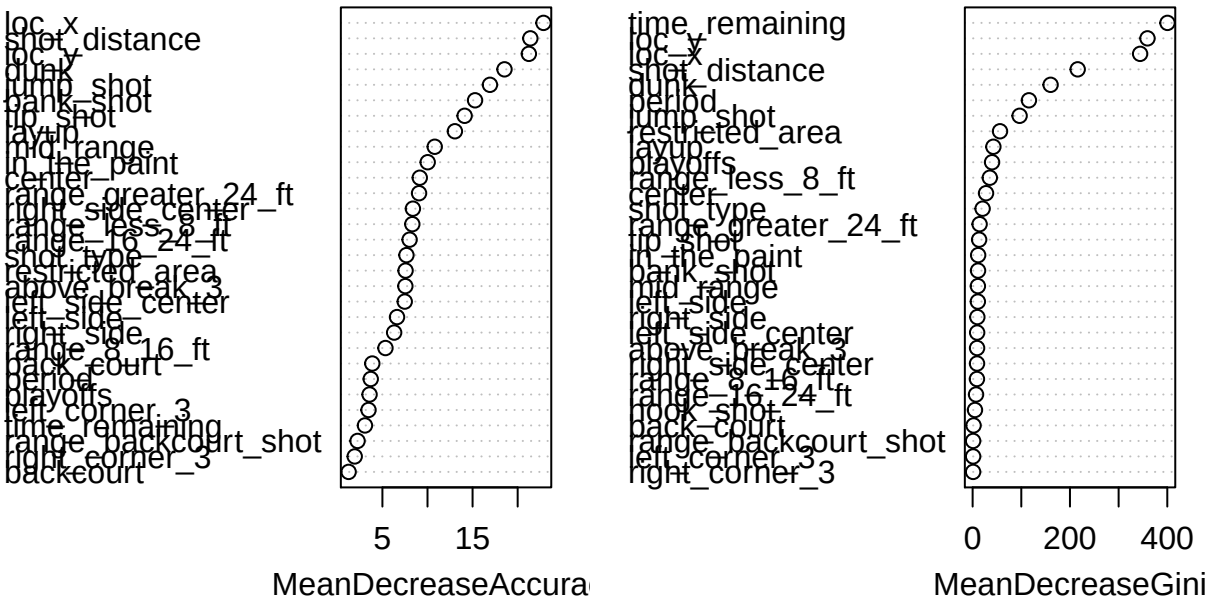
```
##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```



For classification there are typically $\sqrt{p}$ random variables considered for a split. My data set contains 31 predictor variables, so I chose 6 as the number of variables to consider for a split. The number of trees grown was 300. The plot above shows that as the number of trees increases, the test error(indicated by the green line) decreases slightly. The test error rate for this model was slightly higher than the test error rate for the classification tree at 38%.



The variable importance plot shows that the most important variables in terms of both model accuracy and Gini Index were shot_distance, loc_x, loc_y, dunk, and jump_shot. The Gini Index placed more emphasis on time variable predictors by placing importance on time_remaining and period as well.

Logistic Regression

The last model I fit was a Logistic Regression model. Logistic Regression uses maximimum likelihood to estimate coefficients and then uses these coefficients in the odds model. I thought Logistic Regression would be a good model for this data, because the response variable is binary.

```
##
## Call:
## glm(formula = shot_made_flag ~ ., family = binomial, data = stats.train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.645  -1.273   0.889   1.028   2.335
##
## Coefficients: (6 not defined because of singularities)
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)                1.26e-01   4.38e-01    0.29  0.77446
## loc_x                      8.40e-04   4.32e-04    1.95  0.05164 .
## loc_y                     -5.21e-05   5.24e-04   -0.10  0.92087
## period                     4.98e-02   1.31e-02    3.80  0.00015 ***
## playoffs1                  1.09e-02   4.27e-02    0.25  0.79951
## shot_distance              2.39e-02   1.02e-02    2.34  0.01948 *
## shot_type3PT Field Goal -8.99e-02   2.78e-01   -0.32  0.74697
## time_remaining             -2.20e-04   7.28e-05   -3.02  0.00250 **
## bank_shot1                 -1.74e+00   3.43e-01   -5.07  4.1e-07 ***
```

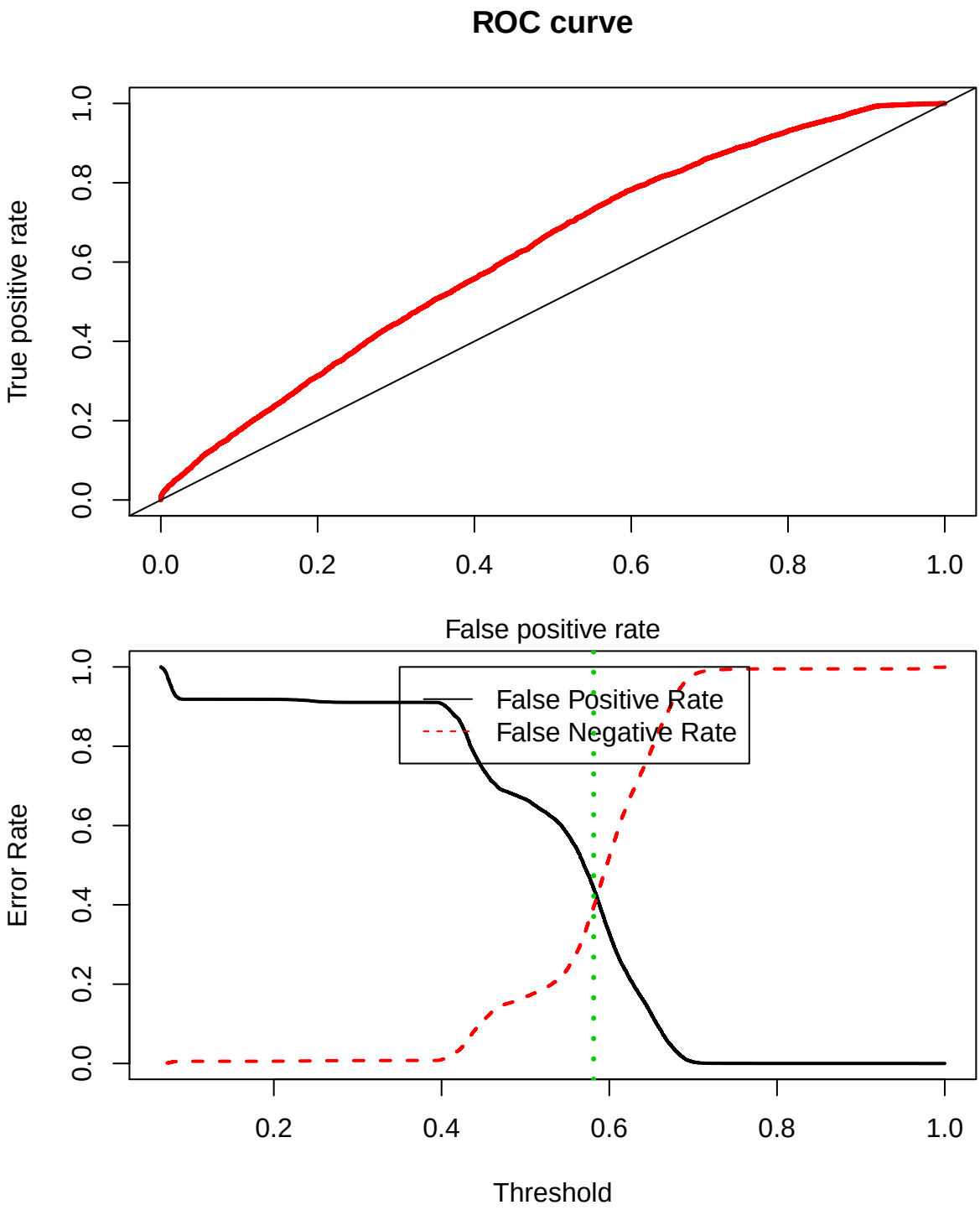
```
## dunk1                  -3.15e+00   2.36e-01  -13.32  < 2e-16 ***
## hook_shot1             -6.27e-01   3.04e-01   -2.06  0.03931 *
## jump_shot1             -2.66e-01   2.36e-01   -1.13  0.25942
## layup1                 -9.07e-01   1.96e-01   -4.63  3.6e-06 ***
## tip_shot1                     NA         NA      NA       NA
## back_court1             1.14e+01   1.08e+02    0.11  0.91599
## center1                -7.20e-02   9.92e-02   -0.73  0.46793
## left_side_center1       8.07e-02   1.41e-01    0.57  0.56607
## left_side1              2.46e-01   1.28e-01    1.91  0.05576 .
## right_side_center1     -2.34e-01   7.91e-02   -2.96  0.00308 **
## right_side1                   NA         NA      NA       NA
## above_break_31          3.24e-01   1.61e-01    2.01  0.04478 *
## backcourt1             -8.82e+00   1.08e+02   -0.08  0.93492
## in_the_paint1           3.47e-01   3.50e-01    0.99  0.32093
## left_corner_31          1.66e-01   2.32e-01    0.72  0.47458
## mid_range1              4.56e-01   3.41e-01    1.34  0.18127
## restricted_area1        5.37e-01   3.82e-01    1.41  0.15957
## right_corner_31               NA         NA      NA       NA
## range_16_24_ft1        -3.93e-01   1.33e-01   -2.95  0.00317 **
## range_greater_24_ft1          NA         NA      NA       NA
## range_8_16_ft1         -4.09e-01   9.45e-02   -4.33  1.5e-05 ***
## range_backcourt_shot1         NA         NA      NA       NA
## range_less_8_ft1              NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 26492  on 19271  degrees of freedom
## Residual deviance: 25100  on 19246  degrees of freedom
## AIC: 25152
##
## Number of Fisher Scoring iterations: 11
```

I first fit a logistic regression model to the training data. The output above shows a summary of the model, including coefficients for each variable. We can see that variables that have large coefficients are important to the model. For example, the shot's location, represented by loc_x and loc_y were significant variables.

Some variable coefficients can be interpreted as follows:

The variable period has a coefficient of 4.982e-02. This indicates that, for every one unit change in period, the log odds of a make increases by 4.982e-02, all other variables held constant.

The variable center1 has a coefficient of -7.20e-02. This indicates that if the shot is taken from the center zone, for every one unit change in center1, the log odds of a make decreases by 7.20e-02, all other variables held constant.



After fitting the model on the training data, I looked to find the best value for a probability threshold on which to make classifications. First, I constructed the ROC curve, which had an area under the curve of 0.625. I then obtained false positive rates and false negative rates from this curve and ploted these against error rate and threshold values between [0,1]. From the graph above, it can be seen that the best probability threshold value is 0.5814.

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
##       true
```

```
## pred   Make Miss
##   Make 1239 2126
##   Miss 1629 1431
```

```
## [1] 0.5844
```

I used the model to make probability predictions on the test set. These probabilities where then transformed into class labels using the optimal threshold found from the ROC curve. If an observation had a probability less than or equal to 0.5814, then it was labeled as a Miss and the observation was labeled as a Make otherwise. This resulted in a test error rate of 58.4%, which is significantly higher than both the error rates for the Classification Tree as well as the Random Forest. This might be due to how the model handled the dummification of the categorical variables. The model was not able to estimate coefficients for many of the dummy variables.

Conclusion

Evaluating model accuracy, the Classification Tree and the Random Forest Model performed much better than the Logistic Regression Model. The Classification Tree had a test error rate of 37.7% while the Random Forest Model had a test error rate of 38%, so the Classification Tree was marginally better. I have chosen the Classification Tree as the final model, because of it's interpretability. I can easily see which variables are being used to make classifications by plotting the model. This allowed me to confirm hypotheses about the predictor variables.

A test error rate of 38% is fairly high. I believe this error rate results from study limitations from the data. One issue I had was that I couldn't fully utilize the whole data set I was working with. The original dataset contained 24 predictor variables. I was only able to utilize about half of those variables. Many of the variables in the data set were also categorical variables with a very high number of levels. I attempted to counter this by creating dummy variables for the different levels of the categorical variables, but this might've affected my model's accuracy because the different levels were split up into variables that didn't capture all of the information of the original variable. This could've affected correlation between the predictor variables. Some categorical variables also had too many levels to use in the models. For example, action_type was a categorical variable with 57 levels and I could not use it in my models. Although the overall data set contained a fairly evenly distributed amount of Makes and Misses for the outcome variable, my models suffered from class imbalance within certain variables. For example, the variable combined_action_type contained the levels dunk and jump_shot. Almost every single shot that was a dunk was a Make. Jump shots, on the other hand had significantly lower observations that were Makes. The effect of this class imbalance is seen in the Classification Tree Model. It only used the variable combined_action_type and was heavily influenced by the high sucess rate of dunks and low sucess rate of jumpshots. In addition, my models suffered from high collinearity between predictor variables.

Appendix

```
## Reading in dataset
library(dplyr)
library(knitr)
basketball.stats <-read.csv("/Users/calvinnguyen/downloads/data.csv")
RNGkind(sample.kind = "Rejection")
set.seed(3)

## Variable summary
str(basketball.stats)

## Data Visualization
data <- read.csv("/Users/calvinnguyen/downloads/data.csv", stringsAsFactors = FALSE)
```

```
train <- data[!is.na(data$shot_made_flag),]
test <- data[is.na(data$shot_made_flag),]

train$shot_made_flag <- as.factor(train$shot_made_flag)

train$shot_made_flag <- factor(train$shot_made_flag, levels = c("1", "0"))

pplot <- function(feat) {
  feat <- substitute(feat)
  ggplot(data = train, aes_q(x = feat)) +
    geom_bar(aes(fill = shot_made_flag), stat = "count", position = "fill") +
    scale_fill_brewer(palette = "Set1", direction = -1) +
    ggtitle(paste("accuracy by", feat))

}

# a plot to see position by feature
courtplot <- function(feat) {
  feat <- substitute(feat)
  train %>%
    ggplot(aes(x = lon, y = lat)) +
    geom_point(aes_q(color = feat), alpha = 0.7, size = 3) +
    ylim(c(33.7, 34.0883)) +
    scale_color_brewer(palette = "Set1") +
    theme_void() +
    ggtitle(paste(feat))
}

library(ggplot2)
ggplot(train, aes(x = loc_x, y = loc_y)) +
  geom_point(aes(color = shot_made_flag), alpha = 0.5, size = 0.5) +
  ylim(c(-50, 400)) +
  theme_void() +
  scale_color_brewer(palette = "Set1") +
  facet_grid(~ shot_made_flag) +
  labs(title = "Shots Made(Blue) vs. Shots Missed(Red)")

makes = nrow (filter(basketball.stats, shot_made_flag == 1))
misses = nrow (filter(basketball.stats, shot_made_flag == 0))

library(ggplot2)
ggplot() +
  geom_point(data = filter(train, combined_shot_type == "Jump Shot"),
             aes(x = lon, y = lat), color = "grey", alpha = 0.3, size = 2) +
  geom_point(data = filter(train, combined_shot_type != "Jump Shot"),
             aes(x = lon, y = lat,
                 color = combined_shot_type), alpha = 0.7, size = 3) +
  ylim(c(33.7, 34.0883)) +
  scale_color_brewer(palette = "Set1") +
  theme_void() +
  ggtitle("Shot Types")

prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
```

```r
as.data.frame.matrix(temp) -> temp
temp$shot <- rownames(temp)
ggplot(temp, aes(x = reorder(shot, `1`), y = 1)) +
  geom_point(aes(y = `1`), size = 3, color = " dark blue", stat = "identity") +
  coord_flip() +
  labs(y = "Accuracy", x = "", title = "Accuracy by Shot_type")

pplot(shot_zone_area) + coord_flip()

pplot(shot_zone_basic) + coord_flip()

pplot(shot_zone_range) + coord_flip()

pplot(minutes_remaining)

pplot(period)

pplot(season) + coord_flip()

## Missingness of dataset
missing.basketball = subset(basketball.stats,is.na(shot_made_flag))
summary(missing.basketball)
basketball.stats <- na.omit(basketball.stats)

## Preprocessing
library(tidyverse)

basketball.stats = basketball.stats %>%
  mutate(time_remaining = ((minutes_remaining*60)+ seconds_remaining))

basketball.stats$shot_made_flag <- as.factor(ifelse(basketball.stats$shot_made_flag == 1,"Make","Miss")

basketball.stats$bank_shot <- as.factor(ifelse(basketball.stats$combined_shot_type == "Bank Shot",1,0))
basketball.stats$dunk <- as.factor(ifelse(basketball.stats$combined_shot_type == "Dunk",1,0))
basketball.stats$hook_shot <- as.factor(ifelse(basketball.stats$combined_shot_type == "Hook Shot",1,0))
basketball.stats$jump_shot <- as.factor(ifelse(basketball.stats$combined_shot_type == "Jump Shot",1,0))
basketball.stats$layup <- as.factor(ifelse(basketball.stats$combined_shot_type == "Layup",1,0))
basketball.stats$tip_shot <- as.factor(ifelse(basketball.stats$combined_shot_type == "Tip Shot",1,0))

basketball.stats$back_court <- as.factor(ifelse(basketball.stats$shot_zone_area == "Back Court(BC)",1,0
basketball.stats$center <- as.factor(ifelse(basketball.stats$shot_zone_area == "Center(C)",1,0))
basketball.stats$left_side_center <- as.factor(ifelse(basketball.stats$shot_zone_area == "Left Side Cen
basketball.stats$left_side <- as.factor(ifelse(basketball.stats$shot_zone_area == "Left Side(L)",1,0))
basketball.stats$right_side_center <- as.factor(ifelse(basketball.stats$shot_zone_area == "Right Side C
basketball.stats$right_side <- as.factor(ifelse(basketball.stats$shot_zone_area == "Right Side(R)",1,0)

basketball.stats$above_break_3  <- as.factor(ifelse(basketball.stats$shot_zone_basic == "Above the Breal
basketball.stats$backcourt  <- as.factor(ifelse(basketball.stats$shot_zone_basic == "Backcourt",1,0))
basketball.stats$in_the_paint  <- as.factor(ifelse(basketball.stats$shot_zone_basic == "In The Paint (N
basketball.stats$left_corner_3 <- as.factor(ifelse(basketball.stats$shot_zone_basic == "Left Corner 3",
basketball.stats$mid_range <- as.factor(ifelse(basketball.stats$shot_zone_basic == "Mid-Range",1,0))
basketball.stats$restricted_area  <- as.factor(ifelse(basketball.stats$shot_zone_basic == "Restricted A
basketball.stats$right_corner_3  <- as.factor(ifelse(basketball.stats$shot_zone_basic == "Right Corner
```

```
basketball.stats$range_16_24_ft <- as.factor(ifelse(basketball.stats$shot_zone_range == "16-24 ft.",1,0)
basketball.stats$range_greater_24_ft <- as.factor(ifelse(basketball.stats$shot_zone_range == "24+ ft.",1
basketball.stats$range_8_16_ft <- as.factor(ifelse(basketball.stats$shot_zone_range == "8-16 ft.",1,0))
basketball.stats$range_backcourt_shot <- as.factor(ifelse(basketball.stats$shot_zone_range == "Back Cou
basketball.stats$range_less_8_ft <- as.factor(ifelse(basketball.stats$shot_zone_range == "Less Than 8 f

basketball.stats$playoffs <- as.factor(basketball.stats$playoffs)

basketball.stats = basketball.stats %>%
  select(-c(game_event_id,game_id,lat,lon,team_id,team_name,matchup,opponent, action_type, combined_sho

## splitting into training and test sets
RNGkind(sample.kind = "Rejection")
set.seed(3)

train = sample(1:nrow(basketball.stats), .75*nrow(basketball.stats))
stats.train = basketball.stats[train,]
stats.test = basketball.stats[-train,]

dim(stats.train)
dim(stats.test)

## Classification Tree Model Building

library(tree)
fit.tree <- tree(shot_made_flag ~., data = stats.train)
summary(fit.tree)

cv <- cv.tree(fit.tree, FUN = prune.misclass, K = 10)
cv

plot(fit.tree)
text(fit.tree, pretty = 0, cex = .7)
title("Training Set Classification Tree")


best.cv <- cv$size[which.min(cv$dev)]

yhat.test <- predict(fit.tree,stats.test, type = "class")

error <- table(yhat.test,stats.test$shot_made_flag)
error

tree.err = 1-sum(diag(error))/sum(error)

## Random Forest Model Building
library(randomForest)

rf.stats = randomForest(shot_made_flag~.,data = stats.train, mtry = 6, ntree = 300, importance = TRUE)

plot(rf.stats)

yhat.rf = predict(rf.stats, newdata = stats.test)
```

```r
rf.err = table(pred = yhat.rf, truth = stats.test$shot_made_flag)
rf.test.err = 1 - sum(diag(rf.err))/sum(rf.err)

varImpPlot(rf.stats)

## Logistic Regression Model Building
glm.fit <- glm(shot_made_flag~., data = stats.train, family = binomial)
summary(glm.fit)

prob.training = predict(glm.fit, type="response")

library(ROCR)

pred = prediction(prob.training, stats.train$shot_made_flag)

perf = performance(pred, measure="tpr", x.measure="fpr")

plot(perf, col=2, lwd=3, main="ROC curve")
abline(0,1)

auc = performance(pred, "auc")@y.values

# FPR
fpr = performance(pred, "fpr")@y.values[[1]]
cutoff = performance(pred, "fpr")@x.values[[1]]
# FNR
fnr = performance(pred,"fnr")@y.values[[1]]

rate = as.data.frame(cbind(Cutoff=cutoff, FPR=fpr, FNR=fnr))
rate$distance = sqrt((rate[,2])^2+(rate[,3])^2)

index = which.min(rate$distance)
best = rate$Cutoff[index]

matplot(cutoff, cbind(fpr,fnr), type="l",lwd=2, xlab="Threshold",ylab="Error Rate")

legend(0.35, 1, legend=c("False Positive Rate","False Negative Rate"),
col=c(1,2), lty=c(1,2))
abline(v=best, col=3, lty=3, lwd=3)

prob.test = predict(glm.fit, stats.test, type="response")

glm.test = stats.test %>%
  mutate(predMAKE = as.factor(ifelse(prob.test <= best, "Miss","Make")))

glm.err <- table(pred=glm.test$predMAKE, true=stats.test$shot_made_flag)
glm.err
1-sum(diag(glm.err))/sum(glm.err)
```

References

Kobe Bryant Shot Selection Dataset. Kaggle, 7 May 2020, [https://www.kaggle.com/c/kobe-bryant-shot-selection]

Exploring Kobe's Shots. Alexandru Papiu, 7 May 2020, [https://www.kaggle.com/apapiu/exploring-kobe-s-shots]