**Problem Statement and Dataset**

This documentation iterates the process and reasoning behind the Team 13 code for the Machine Learning 2 Team Presentation. The dataset, named Kobe Bryant Shot Selection, houses 30,697 basketball shots, characteristics about each shot, and whether or not Kobe Bryant made or missed the hoop. The dataset is available on the Kaggle website with the objective to predict which shots successfully made it through the net.

Here are the available columns from the Kaggle dataset:

**Data Dictionary**

| **Field Name** | **Field Type** | **Field Category** | **Definition** |
| --- | --- | --- | --- |
| action\_type | VARCHAR | Explanatory | A description of the action performed combined with the shot type |
| combined\_shot\_type | VARCHAR | Explanatory | The name of the shot type performed |
| game\_event\_id | NUMERIC | Explanatory | Paired identification for the game and event |
| game\_id | NUMERIC | Explanatory | The unique id assigned to each game matchup |
| lat | FLOAT | Explanatory | Team 13 suspects the latitude is of Los Angeles at 34.0522° N |
| loc\_x | NUMERIC | Explanatory | The location on the basketball half-court corresponds to the x-axis where 0 references the basketball net. The value becomes greater toward the right; the value is negative and gets smaller toward the left. |
| loc\_y | NUMERIC | Explanatory | The location on the basketball half-court corresponds to the y-axis where 0 references the basketball net. The value of the y-axis gets greater as Kobe moves toward the middle of the basketball court. |
| lon | FLOAT | Explanatory | Team 13 suspects the longitude is of Los Angeles at 118.2437° W |
| minutes\_remaining | NUMERIC | Explanatory | The minutes remaining when the shot was attempted in the quarter. Values between 0 to 11 minutes (inclusive). |
| period | NUMERIC | Explanatory | The quarter in the game when the shot was attempted. Values are between 1 to 4 (inclusive). |
| playoffs | NUMERIC | Explanatory | Whether or not the shot was attempted during the playoffs or not. 0 = Not in playoffs. 1 = playoffs. |
| season | DATE | Explanatory | The season when the shot was attempted. |
| seconds\_remaining | NUMERIC | Explanatory | The seconds remaining when the shot was attempted. Values between 0 and 59 (inclusive). |
| shot\_distance | NUMERIC | Explanatory | The distance between the hoop and Kobe when the shot was attempted in feet. Values between 0 and 79 (inclusive). |
| shot\_made\_flag | NUMERIC | To Predict | Whether or not the shot Kobe took was a make or a miss. 0 = miss; 1 = make. The Y variable that we are attempting to predict. |
| shot\_type | VARCHAR | Explanatory | Distinction between a 2 point field goal or 3 point field goal attempt. |
| shot\_zone\_area | VARCHAR | Explanatory | Description of where the shot was attempted. Values include Center, Left Side, Right Side etc. |
| shot\_zone\_basic | VARCHAR | Explanatory | Description of where the shot was attempted but more granular compared to shot\_zone\_area. Values include Mid-Range, In The Paint, Above the Break 3. |
| shot\_zone\_range | VARCHAR | Explanatory | The distance between where the shot was attempted and the net; however, unlike shot\_distance, shot\_zone\_range are buckets. |
| team\_id | NUMERIC | Explanatory | The unique id of what team Kobe Bryant was on when the shot was committed. The Los Angeles Lakers. |
| team\_name | VARCHAR | Explanatory | The name of the team Kobe Bryant was on when the shot was committed. The Los Angeles Lakers. |
| game\_date | DATE | Explanatory | The date when the shot was attempted. |
| matchup | VARCHAR | Explanatory | The abbreviation of the team Kobe Bryant was on versus the opponent team when the shot was attempted. |
| opponent | VARCHAR | Explanatory | The abbreviation of the opponent team when the shot was attempted. |
| shot\_id | NUMERIC | Explanatory | The unique id assigned to each shot attempted. |

**Data Exploration**

Now that the team reviewed and understood the data available from the Kaggle dataset, the team can explore the data. Some of the columns are unique identifiers or names which cannot predict whether the shot was a make or a miss; therefore, the team removes them from the R dataframe.

A popular NBA basketball metric is the 2 Point Field Goal (2PT FG%) and 3 Point Field Goal Percentage (3PT FG%). These metrics showcase the accuracy of the player when making 2 pointers versus 3 pointers. Kobe Bryant’s 2PT FG% is greater than the NBA average at 47.7% compared to 45.2%. Alternatively, the NBA 3PT FG% is higher than Kobe’s average at 35.4% and 32.9% respectively. This demonstrates that Kobe has a higher accuracy at 2 pointers compared to 3 pointers; therefore, the team will consider the shot\_type variable in the R model.

Kobe Bryant’s accuracy decreases over the quarters in the game with quarter 1 having the highest accuracy rate of 46.5% while quarter 4 is at 41.3%. This decline in accuracy can be attributed to greater stress later in the game. The team will also include the period variable in the R model because of this large difference between the first and fourth quarters.

The location where the shot was attempted is crucial to predict whether the shot was a make or miss. In the tableau dashboard included in the GitHub, the heat map shows all the successful shots made by Kobe Bryant. Team 13 notices greater volume of successes at the net (dunks), center shots, right center, and right edge. It seems like Kobe prefers the right side of the court more than the left; therefore, columns like shot\_zone\_area, and shot\_zone\_basic will be considered. Shot\_zone\_range will not be considered because shot\_zone\_basic already captures the distance and we desire a parsimonious model.

The opponent variable houses more than 32 teams; therefore, R throws an error because the default maximum number of levels for a factor variable in R is 32. With the error and for parsimonious efforts, the team decided to also remove opponent and matchup from the dataframe used to conduct machine learning.

**R File**

The Kaggle dataset is a CSV file so team 13 reads in the csv file as a data frame with string as factors and header set to true. However, the shot\_made\_flag are numerical values of 0 and 1 so we must call the as.factor command because R by default considers this column as numerical data. Moreover, instead of having separate minute and second remaining columns, we calculated the total time remaining in the quarter when the shot was attempted and named this new column time\_remaining. We removed undesired features as discussed in the data exploration section. This is accomplished by subtracting out the vector of columns in the subset command. Moreover, some other factors have been altered – the opponent variable has 33 categories but this is because team names have changed; therefore, the names have been fixed. The kaggle dataset has purposeful NULL shot\_made\_flag values because the goal is to correctly predict the outcome of the shot. Therefore, these rows with null values have been removed from the considered dataset subset. A proportion of the dataset is considered as the training subset while the remaining observations are the testing set.

*Code: Initialize and Read Data*

rm(list = ls())

library(randomForest)

library(e1071)

library(gbm)

#import data set and separate into train/test based on 0/1 or NA in shot\_made\_flag field

shooting <- read.csv("KobeBryantRIP.csv", header = T, stringsAsFactors = T)

#add feature

shooting$time\_remaining <- shooting$minutes\_remaining\*60 + shooting$seconds\_remaining

shooting$post\_achilles <- as.factor(ifelse(shooting$shot\_id >= 21003, 1, 0))

#modify features

shooting$shot\_made\_flag <- as.factor(shooting$shot\_made\_flag)

shooting$opponent[shooting$opponent == "NJN"] <- "BKN"

shooting$opponent[shooting$opponent == "VAN"] <- "MEM"

shooting$opponent[shooting$opponent == "NOH"] <- "CHA"

shooting$opponent[shooting$opponent == "SEA"] <- "OKC"

shooting$opponent <- as.character(shooting$opponent)

shooting$opponent <- as.factor(shooting$opponent)

#remove features

shooting <- subset(shooting, select = -c(game\_event\_id, game\_id, lat, lon, loc\_x, loc\_y,

minutes\_remaining, seconds\_remaining, team\_id,

team\_name, game\_date, matchup, shot\_id, action\_type))

#split into train and test based on shot\_made\_flag being 1/0 or NA

shooting <- shooting[!is.na(shooting$shot\_made\_flag),]

trainIndices <- sample(1:nrow(shooting), 0.9\*nrow(shooting))

train <- shooting[trainIndices, ]

test <- shooting[-trainIndices, ]

**Model 1: SVM**

Support Vector Machine (SVM) separates data points into different classes by finding a hyperplane that separates observations into classes. For our project the classes would be make and miss; therefore, SVM could help us predict the shot outcome. The SVM model is trained on the training set with a linear kernel. Team 13 chose the linear kernel because we have a decent number of features; thus the linear method is best for managing higher dimensions. The train accuracy results in 61.3% which is better than random guessing at 50% though there is room for improvement. The test SVM results in 61.4%.

*Code: SVM Model*

#svm

svm.kobe <- svm(shot\_made\_flag ~ ., data = train, kernel = "linear", cost = 1)

yhat <- predict(svm.kobe, newdata = train)

confusion <- table(yhat, train$shot\_made\_flag)

sum(diag(confusion)) / sum(confusion)

#test svm

svm.test <- predict(svm.kobe, newdata = test)

mean(svm.test == test$shot\_made\_flag)

**Model 2: Bagging**

Bagging or Bootstrap Aggregation creates subsets of the training data and combines the predictions of each model as a finalized prediction. Team 13 chose to include a bagging approach to avoid overfitting and help compare models against one another for better quality. The training accuracy results in 99.7% accuracy which alarms us for overfitting issues; the validation set accuracy is the worst performing at 56.9% which serves our suspicions.

*Code: Bagging*

#bagging

bag.kobe <- randomForest(shot\_made\_flag ~ ., data = train, mtry = 10, importance = T)

yhat <- predict(bag.kobe, newdata = train)

confusion <- table(yhat, train$shot\_made\_flag)

sum(diag(confusion)) / sum(confusion)

#test bagging

bag.test <- predict(bag.kobe, newdata = test)

mean(bag.test == test$shot\_made\_flag)

**Model 3: Boosting**

Boosting is the act of adding more weak learners to eventually create a stronger learner which is more reliable and accurate compared to a sole learner. Since our objective is to correctly classify a make or miss, we use the Bernoulli distribution and then iterate through many models with differing interaction depths and shrinkage. Once the double for loop conducts all models, we select the model with greatest accuracy. Boosting train accuracy is 85.5% and results in a test validation accuracy is 54.7% which is the lowest of the three thus far.

Code: Boosting

#boosting

train$shot\_made\_flag <- as.numeric(train$shot\_made\_flag) - 1

boost.kobe <- gbm(shot\_made\_flag ~.,

data=train,

distribution= 'bernoulli',

n.trees = 5000,

interaction.depth = 5,

shrinkage = 0.5,

verbose = F)

yhat <- predict(boost.kobe, newdata=train, type = "response")

yhat <- ifelse(yhat > 0.5, 1, 0)

confusion <- table(yhat, train$shot\_made\_flag)

sum(diag(confusion)) / sum(confusion)

#test boosting

test$shot\_made\_flag <- as.numeric(test$shot\_made\_flag) - 1

boost.test <- predict(boost.kobe, newdata = test, type = "response")

boost.test <- ifelse(boost.test > 0.5, 1, 0)

mean(boost.test == test$shot\_made\_flag)

**Voting Best Model**

Finally, we analyze all three model’s performance and see how many shots each model predicted correctly. We wanted to incorporate all three models; therefore, the voting process says if the majority of models (⅔) predict make, then we predict make. To do this, we returned the test predictions as numeric and minused 1. We need to minus 1 because the classes in R are identified as 1 or 2 – in our data 0 or 1. Finally, we sum up the predictions for each case and if the sum is greater than 1, then that prediction is used. The result is around 66%.

*Code: Model Voting*

#model voting

svm.num <- as.numeric(svm.test) - 1

bag.num <- as.numeric(bag.test) - 1

boost.num <- as.numeric(boost.test) - 1

votes <- ifelse(svm.num + bag.num + boost.num > 1, 1, 0)

mean(votes == test$shot\_made\_flag)