DESAUTELS | McGill
Faculty of Management
Faculté de gestion

MGSC 661 - FINAL PROJECT

Soccer Match Prediction Model

Shehryar HUSSAIN - 260986031

Saad TARIQ - 261008634

Table of Contents

1.0 Introduction

Football (soccer) is the most played and most watched sport in the world today (1). According to Fédération Internationale de Football Association (FIFA), there are approximately 265 million people in the world who play football and roughly 3.5 billion people in the world who consider themselves as football fans (2). This popularity is expected to increase every year with more and more advancements in the game. This game is also of particular interest to data analysts and experts who want to predict the outcomes of football games before the match finishes. Due to the increasing number of bets being placed during the football games, these experts want to know what factors can determine the outcome of the game. However, it is often hard to predict the outcomes of football games, given the uncertain and constantly evolving nature of the game. It is really hard to call out the winner before the 90 minute mark. This is also evident from the 2015/2016 season when Leicester City won the English Premier League which had one in five thousand odds of winning at the start of the league (3).

A football game has a lot of parameters associated with it, the strength of the overall team, their sub-strengths in terms of their defensive acumen or their attacking approach, the overall shots that they take at the goal and the average age of the players in the team, all of these factors contribute to the outcome of the game in one way or another. To this day, no one has been able to say for sure, that a team of certain attributes would be prevalent over the other team every single time. If they could, the betting industry would have been long gone!

The aim of this project is to understand which attributes of a team contribute to their overall success, and the importance of each attribute in doing so as well. Also, using a classification model, we have tried to predict the outcome of each game, given certain team attributes as input. The outcomes are multi-class and have been labeled as HOME (Home Team Wins), AWAY (Away Team Wins) and DRAW. We collected the data for our model through multiple disparate data sources, going through data from multiple English Premier League seasons along with looking at individual team statistics published by FIFA and merging them together in order to create our primary dataset.

The model started off with data visualization, where we looked at each predictor and their skewness in order to determine whether the distribution in data would be detrimental to the accuracy of our model. We looked at collinearity and correlations between various predictors and integrated those into our model, which did not result in an overfitted model nor with a one with a very low accuracy.

2.0 Data Description

We performed exploratory analysis to understand the attributes of data. To visualize the distribution of each predictor, we plotted the boxplot diagram to determine the range and outlier values. Histograms were also plotted in order to check the skewness and distribution of the dataset.

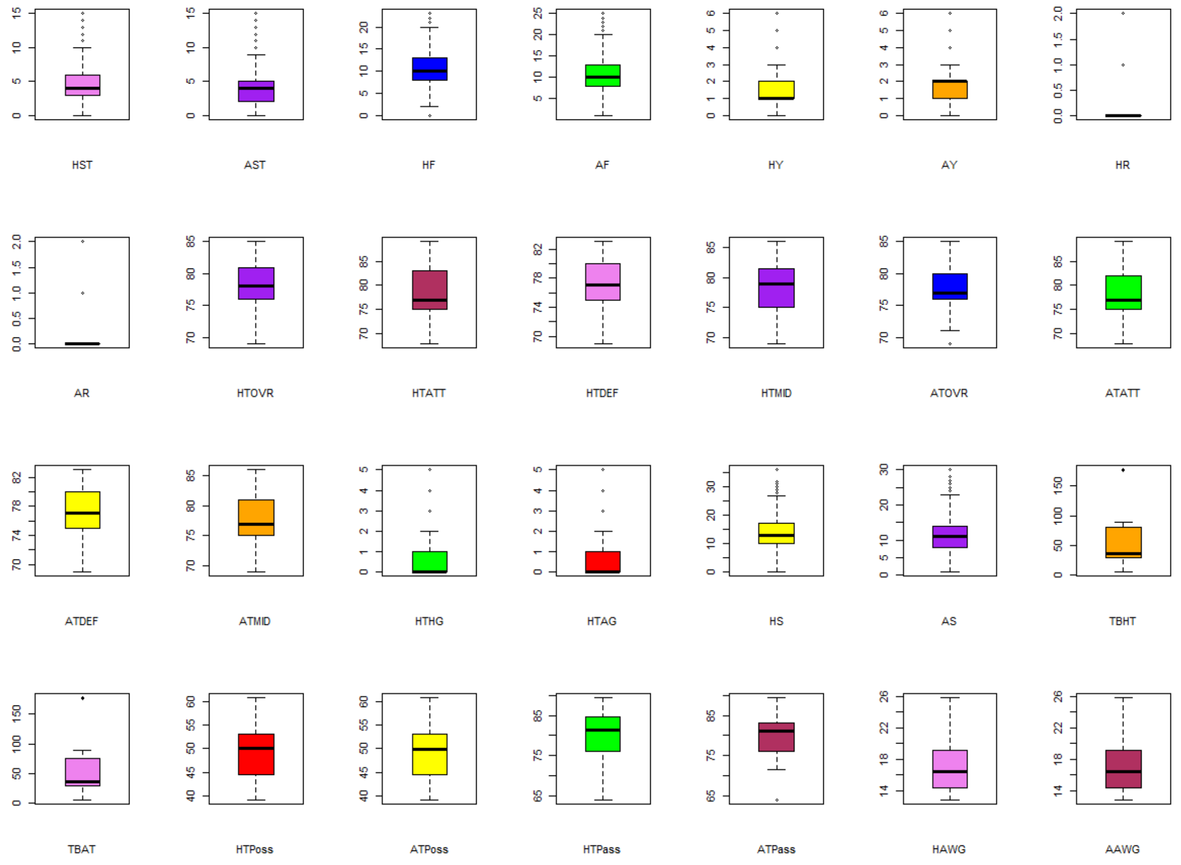


Fig 2.1 - Box Plots of all Variables

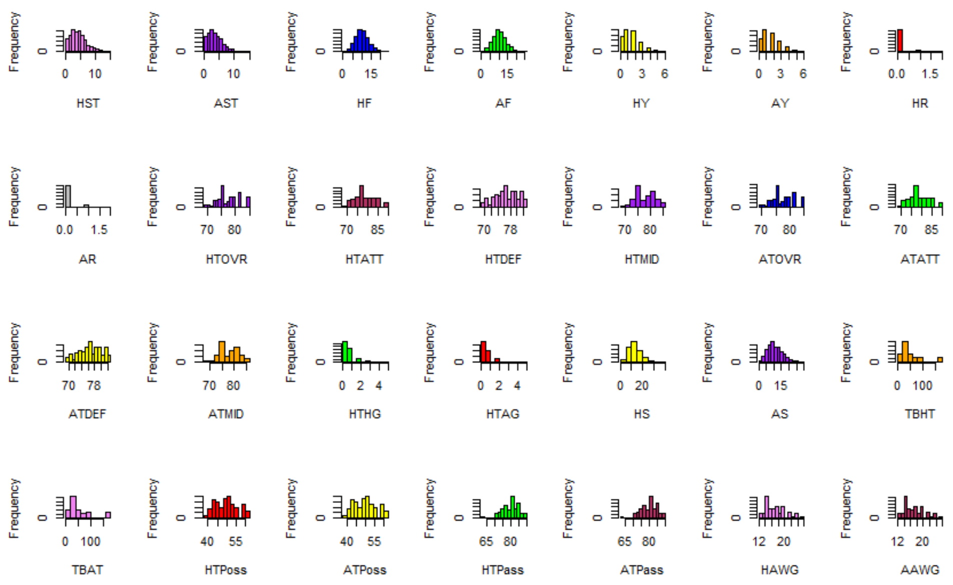


Fig 2.2 - Histograms of all Variables

From the plots above, you can see that there are only two predictors with uneven distribution, i.e. AR and HR, which is explained by the fact that mostly players do not receive a red card during a match and the overall occurrence of such an event is very isolated compared to the overall final result of the match.

In order to check collinearity and correlations between our predictors, we created a collinearity matrix for all the predictors in our dataset (Appendix 2). The predictors with a collinearity of greater than 0.75 were rejected. Below is the collinearity matrix of our finalized dataset.

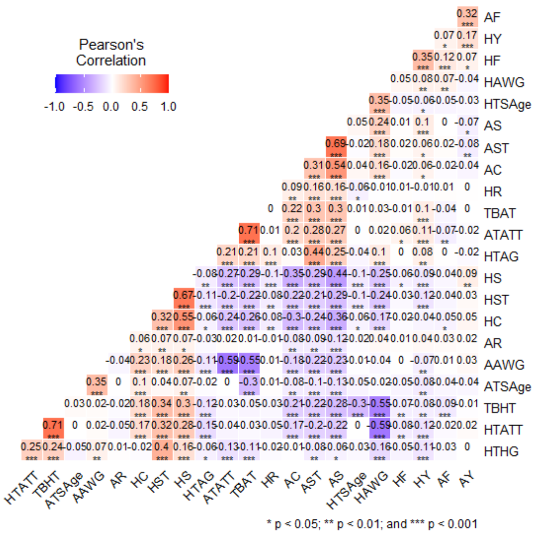


Fig 2.3 - Correlation Matrix of Final Datase

3.0 Model Selection

There were a number of candidate models that we ran in order to understand the behavior of the dataset under different modeling techniques. Our analysis and selection of the optimum model included:

1. Creating Testing and Training datasets.
2. Choosing Predictors using feature selection.
3. Running different Models to check Accuracy.
4. Hyperparameter Tuning.

3.1. Creating testing and training datasets:

In order to check the accuracy of our model, it was important that we had a training and testing dataset. This was to ensure that all the candidate models ran through the same training and testing datasets in order to check their accuracy. We divided our main dataset into two, on the basis of 66.6 and 33.4 ratios.

3.2. Choosing Predictors:

We employed an iterative predictor selection method by implementing a series of simple Random Forest feature selection models, in order to determine the most important predictors for our models. The model was first run integrating all the predictors in the dataset in order to find the relative importance and contribution of each predictor towards the accuracy of our model, this was done through the ***Mean Decrease In Accuracy*** parameter. The predictors found to be having a negative impact on the accuracy of the model were removed and accuracies of all models were calculated, after which the next Random Forest model was run. The process was repeated until we found all the predictors that contributed positively to the accuracy of our model. The predictors finalized in this stage (given below) were used in all the candidate models in order to test their relative performance.

HST | AST | HF | HC | AC | HY | AY | AR | HR | HTATT | ATATT | HTHG | HTAG | HS | AS | TBHT | TBAT | HAWG | AAWG

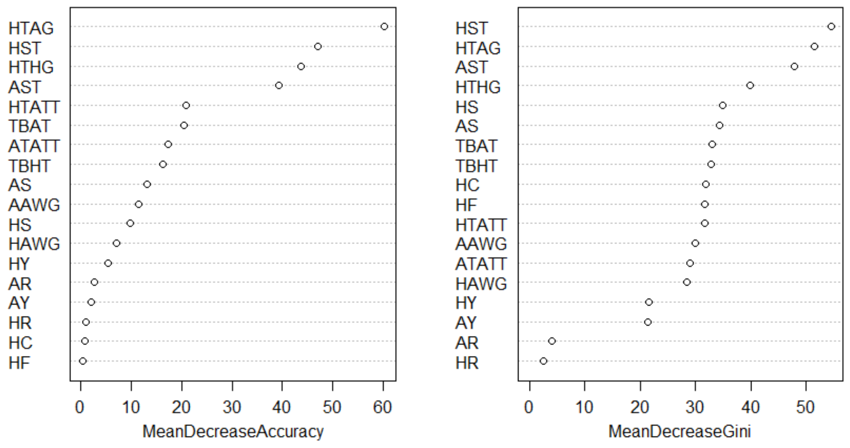


Fig 3.1 - Relative Importance of Predictors

3.3. Iterating Different Models:

We ran a number of different candidate models before settling and deciding on the most optimum model for our needs, in terms of complexity and accuracy. Since our target variable was a multi-class variable, that meant that we could not use simple Logistic Regression, so hence we moved onto Linear Discriminant Analysis.

Multiple Linear Discriminant Models were run using the predictors finalized at each stage and the results regarding the accuracy of the model were recorded. The accuracy was calculated using a Confusion Matrix, which reported the True Positives, True Negatives, False Positives and False Negatives. This data was used to calculate the overall accuracy of the model. We repeated the same process with a Quadratic Discriminant Model, integrating all relevant predictors into our model. The results of the accuracy of this model were also recorded. However, due to a huge difference in the accuracy of the Linear Discriminant Model and Quadratic Discriminant Model, we removed the latter one because of its significantly low accuracy.

Next, we ran a Decision Tree model, by first overfitting our tree to the dataset by setting a very small CP Value (Appendix 3). We then found the optimum value of the CP Value that reduced the error in our fully grown tree and would help against overfitting as well. The plot of Decision

Tree Error (X-val Relative Error) vs CP Value is given below along with the more optimized decision tree.

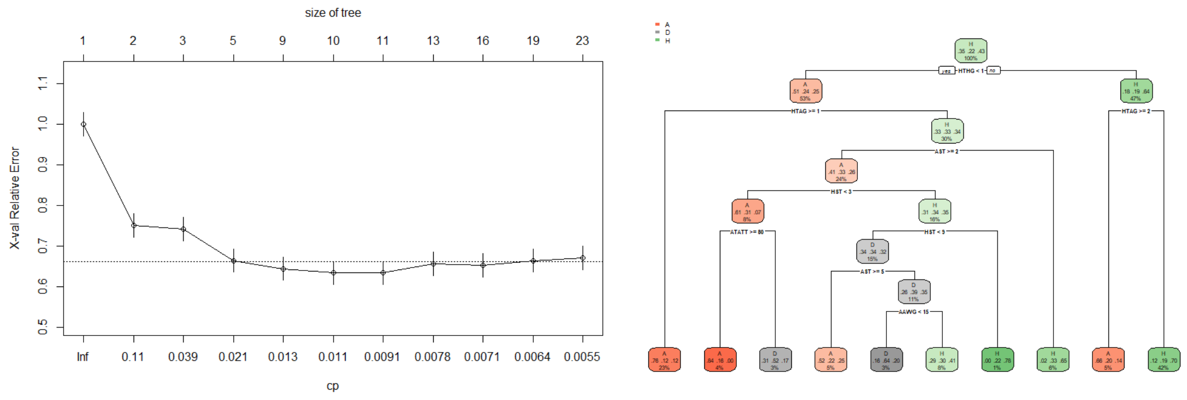


Fig 3.2 - Plot of Optimum CP Value (Left), Decision Tree based on Optimum CP Value (Right)

Next, we ran a Random Forest Model, using the same predictors we finalized during the early stages of our project. The initial model was run using the default values of the hyperparameters, except for *ntrees* which was put as equal to 1303. The model was trained and the accuracy of the model relative to the testing dataset was evaluated. Since this model performed marginally better than all our previous models, we decided to conduct hyperparameter tuning on this model to see the extent to which we can increase its performance. This was one of our finalized models.

Finally, we used a variation of the Gradient Boosting Technique taught in class, a library called *'XGBoost'*. This was primarily done because the Gradient Boosting Method discussed in class only worked for regression and binary classification models, since we had a multi-class classification to perform, we thought it was better to use the appropriate library for this. While the traditional Gradient Boosting Method can be used by using the multinomial method, this method is outdated and may result in errors (as warned by R Studio itself). The XGBoosted Tree method was run using default parameters and we found the accuracy to be on par with the Random Forest method, but the speed of computation was much slower, so hence we decided to finalize Random Forest as the optimum model for our use case based on its speed and accuracy.

3.4. Hyperparameter tuning:

The hyperparameter of Random Forests, *ntrees* was measured against the OOB Error performance of the model, since the OOB Error is a measure of the accuracy of the model. This was used to find the optimal value of the *ntrees* hyperparameter. Next, in order to find the optimal value of *mtry* hyperparameter (which is the number of variables randomly sampled as candidates on each split of the tree), a graph was plotted of the value of *mtry* against OOB Error, given below.

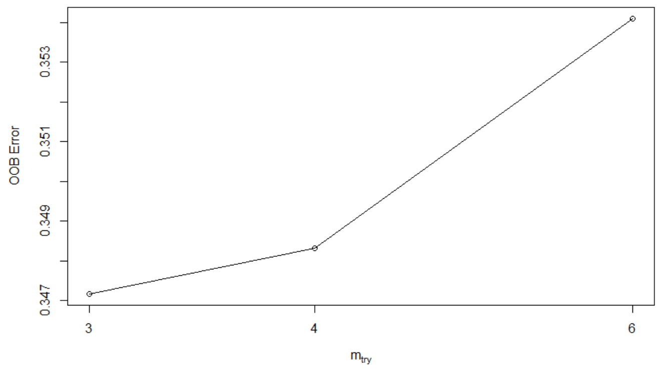


Fig 3.3 - Plot of value of mtry against OOB Error

The optimum value of *mtry* and *ntrees*, significantly improved the performance of the Random Forest model, improving our accuracy by almost 3.5 points in percentage.

4.0 Results

Prediction models with 100% accuracy are almost impossible in football because of numerous variables involved that can affect the final outcome of the match. Most of these variables are very difficult to measure and do not have data available online (such as player confidence, etc.). Therefore, the models that we make are limited by the access of data.

However, our model is still accurate enough to give a good idea about how the match result will vary based on our predictors. We were able to achieve an accuracy of 67.43% and based on our research, most of the models available online could not reach this level of accuracy.

Rank	Model	Accuracy	Time (Green = Fast, Red = Time-Consuming)
1	Random Forest	67.43%	🟩
2	XGBoost	67.2%	🟥
3	LDA	66.1%	🟩
4	QDA	61.7%	🟩
5	Decision Tree	60.1%	🟩

As we can see from the table above, random forest gave us the best results, i.e., it had the highest accuracy while it ran in real time too. On the other hand, although XGBoost had the most comparative accuracy as Random Forest in our final iteration, however, it takes a lot of time for XGBoost to run. Given the nature of the game and how quickly predictions can change, it is important to keep a model that updates as fast as possible. Hence, we reject XGBoost. Our third ranked model was LDA, which had a lower accuracy compared to random forest in all the iterations. Moreover, since random forest uses ensemble training by constructing multiple trees at the time of training, it is a more reliable model as it avoids overfitting when new variables are added. Overfitting is difficult to avoid in LDA and hence we reject that model too. Lastly, we also reject QDA and simple decision trees because of their low accuracy compared to the other models. In soccer, we need to make sure that we are as accurate as possible.

When we solve the model using random forest, our final confusion matrix is as follows:

	ACTUAL			
PREDICTED		A	D	H
	A	110	4	28
	D	41	17	52
	H	21	6	157

As we can see from the confusion matrix above, our model correctly predicts away win games by 64% ($\frac{110}{(110+41+21)}$), draw games by 63% ($\frac{17}{(4+17+6)}$), and home win games by 66% ($\frac{157}{(28+52+157)}$) . This means that approximately every 2 out of 3 games are predicted correctly based on our model, which is a good number given the extensive list of predictors that can be employed in soccer matches and given the existing models that are available online.

5.0 Classifications/Predictions & Conclusions

Using a Decision Tree to represent one of the many trees grown as a part of the Random Forest method, we can create an interpretation of our results and how they would look related to the real world data.

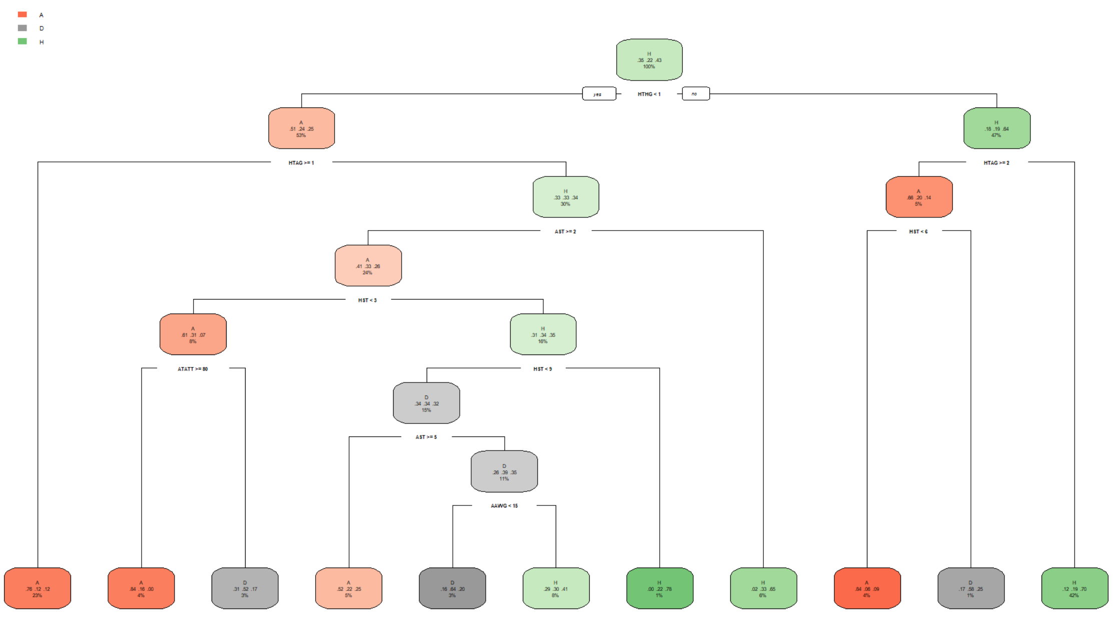


Fig 5.1 - One of the Decision Trees in Random Forest (Example)

For instance, our model predicts that if the home team's half time goals are greater than or equal to 1, and the away team's half time goals are less than 2, then there is a 70% chance that the home team will win. However, if the away team's half time goals are greater than or equal to 2 and the home team's shots on target are less than 6, then there is a 84% chance that the away team will win.

Given the classification above, as a Manager of the home team, they should try to aim for at least 6 shots on target in the whole match as that will lead to a 58% chance of getting a draw rather than losing to the away team.

Similarly, if the half time home goals are 0 and the away team's half time goals are greater than or equal to 1, then there is a 76% chance that the away team will win. However, if the half time score is a draw, the home team should try and restrict the away team from shooting more than 1 shot on target as this will lead to a 65% chance that the home team will win. However, if the away team manages to get more than 1 shot on target, then the home team should get at least 9 shots on target to have a 76% chance of winning. If this does not work, then as a manager, play defensive and restrict away team's shots on target to less than 5. This will increase the likelihood of getting a draw, if not a win.

In conclusion, as a Home Team Manager, try to:

1. Aim for at least 1 goal in the first half and not let the away team score more than 1 goal in the first half as well. This involves a well-thought strategy that is aggressive in attacking but at the same time ensures that the defense line is not easy to break.
2. If the away team scores, try to ensure that they just score 1 goal and do not shoot any more shots on target. Otherwise, try to ensure that your team takes at least 9 shots on target to counter that. This means that initially opt for a counter attacking strategy where the defense line is held, any opportunities of counter attacks are utilized well. If the away team still keeps attacking and shoots on target, try a more aggressive approach by playing a little more attacking.

The classification model can be used to strategize approaches while taking on different teams in order to maximize your chances of winning. It can help the managers decide whether or not they should play an attacking or a defensive style game, or even how they should change approaches based on the scoreboard of the half-time mark.

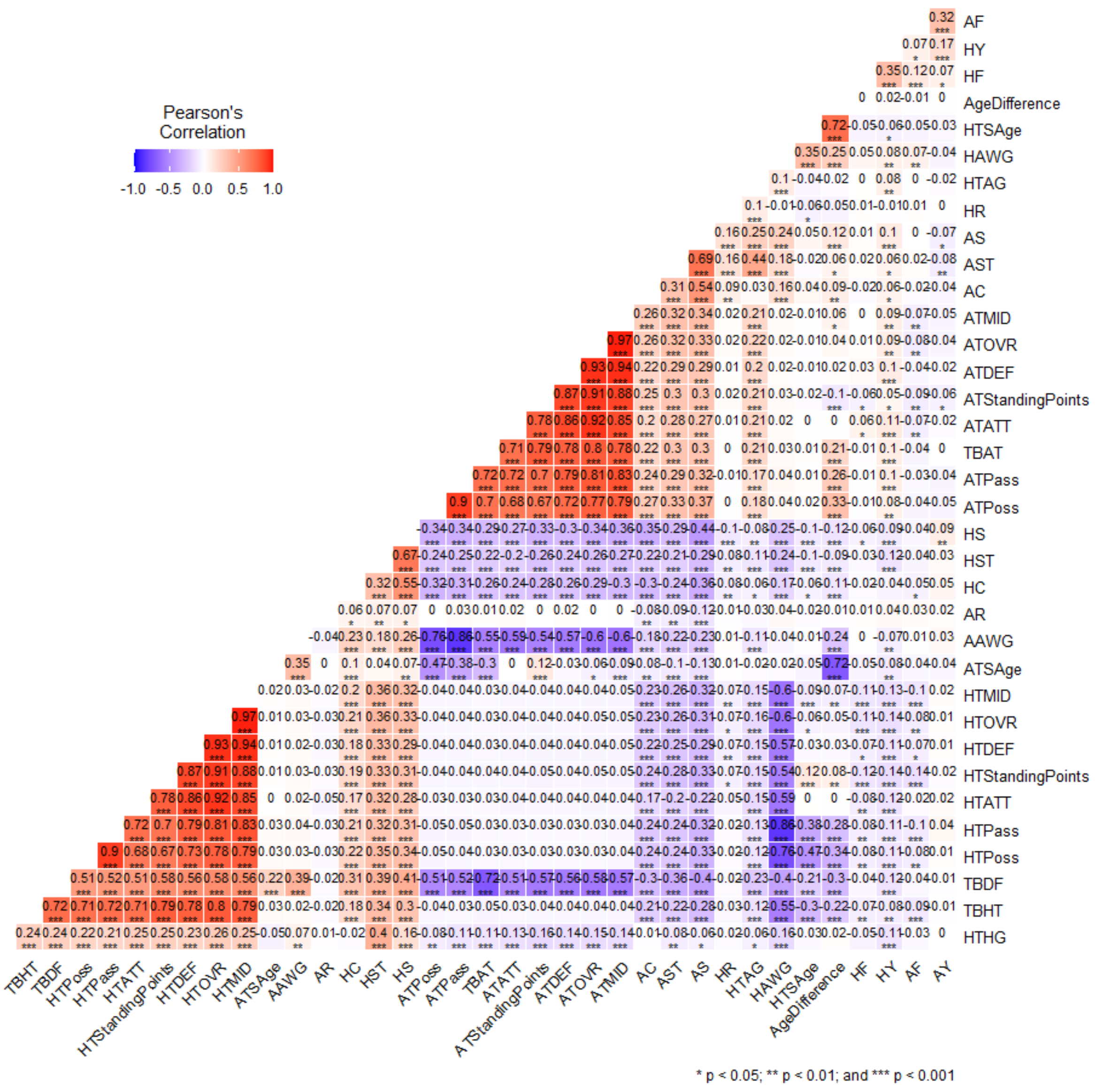
6.0 Appendix

6.1 Appendix 1: Data Dictionary

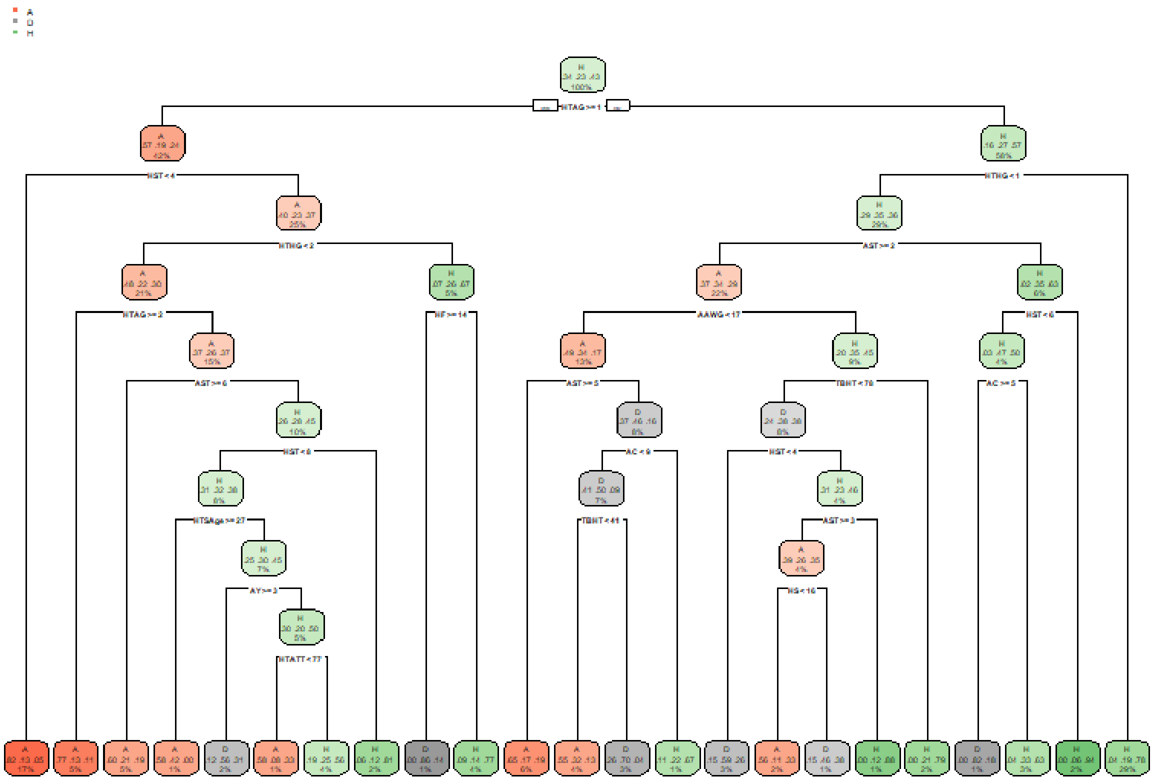
Predictor	Meaning
HST	Home Team Shots on Target
AST	Away Team Shots on Target
HF	Home Team fouls against Away Team
AF	Away Team fouls against Home Team
HY	Home Team Yellow Cards
AY	Away Team Yellow Cards
HR	Home Team Red Cards
AR	Away Team Red Cards
HTOVR	Home Team Overall Statistics
HTATT	Home Team Attack Statistics
HTDEF	Home Team Defensive Statistics
HTMID	Home Team Midfield Statistics
ATOVR	Away Team Overall Statistics
ATATT	Away Team Attack Statistics
ATDEF	Away Team Defensive Statistics
ATMID	Away Team Midfield Statistics
HTHG	Half Time Home Team Goals
HTAG	Half Time Away Team Goals
HS	Home Team Shots

AS	Away Team Shots
TBHT	Home Team Transfer Budget
TBAT	Away Team Transfer Budget
TBDF	Difference between Home and Away Team Transfer Budgets
HTPoss	Home Team Average Possession
ATPoss	Away Team Average Possession
HTPass	Home Team Average Passing Percentage
ATPass	Away Team Average Passing Percentage
HAWG	Home Team Average Air Balls Won
AAWG	Away Team Average Air Balls Won
HTStandingPoints	Home Team Current Table Standing Points (If the team is in the top the get points 5 to 2, if they're in the 5th or 6th position they get 1 point, the teams in 7th to the 17th position get 0 points, the teams in 18th to the 20th position get -1 to -3 points and those not in the Premier League anymore get -5 points)
ATStandingPoints	Away Team Current Table Standing Points (If the team is in the top the get points 5 to 2, if they're in the 5th or 6th position they get 1 point, the teams in 7th to the 17th position get 0 points, the teams in 18th to the 20th position get -1 to -3 points and those not in the Premier League anymore get -5 points)
HTRatingDiff	Difference between Home Team and Away Team points.
HTSAge	Home Team Average Player Age
ATSAge	Away Team Average Player Age
HTR	Half Time Result (Away, Home or Draw)
DFOVR	Difference in Overall Statistics between Home and Away Teams
DFATT	Difference in Attack Statistics between Home and Away Teams
DFMID	Difference in Midfield Statistics between Home and Away Teams
DFDEF	Difference in Defensive Statistics between Home and Away Teams

6.2. Appendix 2: Collinearity Matrix of all the Predictors in the Dataset.

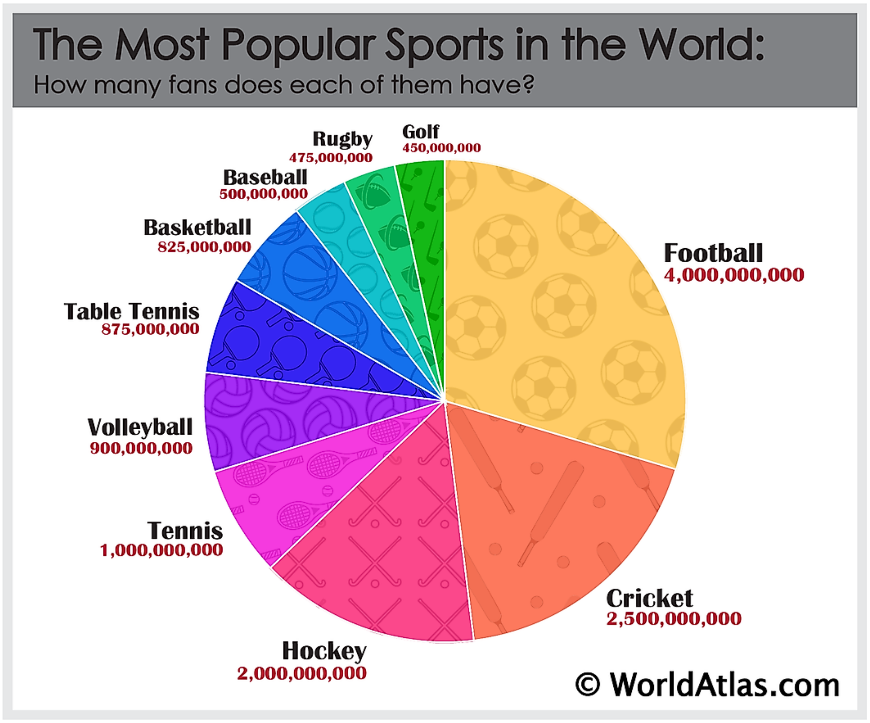


6.3. Appendix 3: Fully grown Decision Tree



6.4. Appendix 4: References & Research Sources

(1) https://www.worldatlas.com/articles/what-are-the-most-popular-sports-in-the-world.html



(2) https://www.allianz.com/en/about-us/sports-culture/football/allianz-football.html#:~:text=Football%20is%20the%20most%20popular,billion%20consider%20themselves%20football%20fans.
(3) https://www.skysports.com/football/news/11712/10261535/premier-league-2015-16-how-the-odds-changed-as-leicester-claimed-the-title
(4) https://www.football-data.co.uk/englandm.php
(5) https://www.football-data.co.uk/notes.txt
(6) https://towardsdatascience.com/machine-learning-algorithms-for-football-prediction-using-statistics-from-brazilian-championship-51b7d4ea0bc8
(7) https://github.com/Matheuskempa/My_Udacity_Capstone/blob/master/Treatment.ipynb
(8) https://www.researchgate.net/publication/335230415_A_Deep_Learning_Approach_to_Predict_Football_Match_Result
(9) https://www.whoscored.com/Regions/252/Tournaments/2/Seasons/8228/Stages/18685/TeamStatistics/England-Premier-League-2020-2021
(10) https://www.premierleague.com/stats
(11) https://www.fifaindex.com/teams/fifa21/

7.0 Code

```
############################ INSTALLING PACKAGES & CALLING LIBRARIES
##############################################

install.packages("readxl")
install.packages("dplyr")
install.packages("MASS")
install.packages("klaR")
install.packages("caret")
install.packages("mlbench")
install.packages("randomForest")
install.packages("tree")
install.packages("rpart")
install.packages("rpart.plot")
install.packages("xgboost")
library(randomForest)
library(mlbench)
library(MASS)
library(klaR)
library(caret)
library(dplyr)
library(readxl)
library(tree)
library(rpart)
library(rpart.plot)
library(xgboost)

##IMPORTING DATASET
data = read_excel("C:/Users/shehr/Desktop/MGSC 661/Final Exam/Football
Team Dataset MGSC661 - Final.xlsx")

data$FTR=as.factor(data$FTR)
```

```
attach(data)

############################ DATA EXPLORATION
##############################################
##Visualizing variables to check distribution, skewness and outliers
# Boxplots
par(mfrow = c(4,7))
boxplot(HST, xlab = "HST", col = "violet")
boxplot(AST, xlab = "AST", col = "purple")
boxplot(HF, xlab = "HF", col = "blue")
boxplot(AF, xlab = "AF", col = "green")
boxplot(HY, xlab = "HY", col = "yellow")
boxplot(AY, xlab = "AY", col = "orange")
boxplot(HR, xlab = "HR", col = "red")
boxplot(AR, xlab = "AR", col = "gray")
boxplot(HTOVR, xlab = "HTOVR", col = "purple")
boxplot(HTATT, xlab = "HTATT", col = "maroon")
boxplot(HTDEF, xlab = "HTDEF", col = "violet")
boxplot(HTMID, xlab = "HTMID", col = "purple")
boxplot(ATOVR, xlab = "ATOVR", col = "blue")
boxplot(ATATT, xlab = "ATATT", col = "green")
boxplot(ATDEF, xlab = "ATDEF", col = "yellow")
boxplot(ATMID, xlab = "ATMID", col = "orange")
boxplot(HTHG, xlab = "HTHG", col = "green")
boxplot(HTAG, xlab = "HTAG", col = "red")
boxplot(HS, xlab = "HS", col = "yellow")
boxplot(AS, xlab = "AS", col = "purple")
boxplot(TBHT, xlab = "TBHT", col = "orange")
boxplot(TBAT, xlab = "TBAT", col = "violet")
boxplot(HTPoss, xlab = "HTPoss", col = "red")
boxplot(ATPoss, xlab = "ATPoss", col = "yellow")
boxplot(HTPass, xlab = "HTPass", col = "green")
boxplot(ATPass, xlab = "ATPass", col = "maroon")
boxplot(HAWG, xlab = "HAWG", col = "violet")
boxplot(AAWG, xlab = "AAWG", col = "maroon")



# Histograms
par(mfrow = c(4,7))
```

```r
hist(HST, xlab = "HST", col = "violet", main = NULL)
hist(AST, xlab = "AST", col = "purple", main = NULL)
hist(HF, xlab = "HF", col = "blue", main = NULL)
hist(AF, xlab = "AF", col = "green", main = NULL)
hist(HY, xlab = "HY", col = "yellow", main = NULL)
hist(AY, xlab = "AY", col = "orange", main = NULL)
hist(HR, xlab = "HR", col = "red", main = NULL)
hist(AR, xlab = "AR", col = "gray", main = NULL)
hist(HTOVR, xlab = "HTOVR", col = "purple", main = NULL)
hist(HTATT, xlab = "HTATT", col = "maroon", main = NULL)
hist(HTDEF, xlab = "HTDEF", col = "violet", main = NULL)
hist(HTMID, xlab = "HTMID", col = "purple", main = NULL)
hist(ATOVR, xlab = "ATOVR", col = "blue", main = NULL)
hist(ATATT, xlab = "ATATT", col = "green", main = NULL)
hist(ATDEF, xlab = "ATDEF", col = "yellow", main = NULL)
hist(ATMID, xlab = "ATMID", col = "orange", main = NULL)
hist(HTHG, xlab = "HTHG", col = "green" , main = NULL)
hist(HTAG, xlab = "HTAG", col = "red", main = NULL)
hist(HS, xlab = "HS", col = "yellow", main = NULL)
hist(AS, xlab = "AS", col = "purple", main = NULL)
hist(TBHT, xlab = "TBHT", col = "orange", main = NULL)
hist(TBAT, xlab = "TBAT", col = "violet", main = NULL)
hist(HTPoss, xlab = "HTPoss", col = "red", main = NULL)
hist(ATPoss, xlab = "ATPoss", col = "yellow", main = NULL)
hist(HTPass, xlab = "HTPass", col = "green", main = NULL)
hist(ATPass, xlab = "ATPass", col = "maroon", main = NULL)
hist(HAWG, xlab = "HAWG", col = "violet", main = NULL)
hist(AAWG, xlab = "AAWG", col = "maroon", main = NULL)

par(mfrow = c(1, 1))
############################ DATA CLEANING
######################################################
#Step 1: Removing distinct variables and non-numerical variables
data = data[-c(1,2,3,29)]
#Step 2: Removing columns that are calculated from other columns and are
not necessary in our model
data = data[-c(19:22,36)]
#Moving target variable to the end of the dataset for ease in evaluation
data = data %>% relocate(FTR, .after = last_col())
```

```
##CORRELATION CHECK
# calculate correlation matrix
set.seed(7)

df = data[, c(1:36)]
correlationMatrix = cor(df)

print(correlationMatrix)

# Separating attributes with correlation > 0.75

highlyCorrelated = findCorrelation(correlationMatrix, cutoff = 0.75)

print(highlyCorrelated)

data1 = data[-highlyCorrelated]

#install.packages("metan")
library(metan)

corrl = corr_coef(df)
plot(corrl)

corrl = corr_coef(data1[, c(1:22)])
plot(corrl)


#Removing collinear (variables calculated from other columns. i.e.,
AgeDifference, TBDF)
data1 = data1[-c(19,22)]

############################# FINAL PREDICTORS
##################################################
#HST, AST, HF, HC, AC, HY, AY, AR, HR, HTATT, ATATT, HTHG, HTAG, HS, AS,
TBHT, TBAT, HAWG, AAWG

############################# FINAL MODEL
#########################################################
```

```
#FEATURE SELECTION

set.seed(2)
forest_in = randomForest(FTR~HST + AST + HF + AF + HC + AC + HY + AY + HR
+ AR + HTATT + ATATT + HTHG + HTAG + HTSAge +
                         ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG,
data = data1, ntree = 1140, importance = TRUE, na.action = na.omit)

importance(forest_in)
#This recommends removing AF so removed AF and ran all models

set.seed(2)
forest_re = randomForest(FTR~HST + AST + HF + HC + AC + HY + AY + HR + AR
+ HTATT + ATATT + HTHG + HTAG + HTSAge +
                         ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG,
data = data1, ntree = 1140, importance = TRUE, na.action = na.omit)

importance(forest_re)
#This recommends removing HTSAge & HR but that decreased accuracy when we
ran top 3 models

set.seed(6)
forest_re1 = randomForest(FTR~HST + AST + HF + HC + AC + HY + AY + AR +
HTATT + ATATT + HTHG + HTAG +
                            ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG,
data = data1, ntree = 1140, importance = TRUE, na.action = na.omit)

importance(forest_re1)
#This recommends removing ATSAge. Hence, removed both HTSAge & ATSAge
since both are doubtful variables and ran top 3 models

set.seed(104)
forest_re3 = randomForest(FTR~HST + AST + HF + HC + HY + AY + AR + HR +
HTATT + ATATT + HTHG + HTAG +
                            HS + AS + TBHT + TBAT + HAWG + AAWG, data =
train.data, ntree = 1140, importance = TRUE, na.action = na.omit)
```

```
importance(forest_re3)

varImpPlot(forest_re3)

#When we run random forest feature selection again, we see that all
predictors left increase accuracy of model
#Hence, we stop with the iterations and decide which is the best model to
choose

# SPLITTING INTO TEST & TRAINING SET

n = nrow(data1)
train.index = sample(n,floor(0.666 * n))
train.data = data1[train.index,]
test.data = data1[-train.index,]


##----------------------RANDOM FOREST---------------------------------
# Implementing Random Forests

set.seed(58)
forest1_re = randomForest(FTR~HST + AST + HF + HC + AC + HY + AY + AR +
HR + HTATT + ATATT + HTHG + HTAG +
                          HS + AS + TBHT + TBAT + HAWG + AAWG, data =
train.data, ntree = 1303, importance = TRUE, na.action = na.omit)

# Finding Optimal Value of mtry
set.seed(60)
mtry <- tuneRF(train.data[, c(1:3,5:18,21,22)], train.data$FTR,
ntreeTry=500,
                  stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
print(mtry)
print(best.m)

# Running Optimal Model

set.seed(55)
```

```
forest2_re = randomForest(FTR~HST + AST + HF + HC + AC + HY + AY + AR +
HR + HTATT + ATATT + HTHG + HTAG +
                          HS + AS + TBHT + TBAT + HAWG + AAWG, data =
train.data, ntree = 1140, importance = TRUE, na.action = na.omit, mtry =
best.m)

# Out of Sample Accuracy

predicted_test_re = predict(forest2_re, test.data, type = "class")
table(predicted_test_re)
confusionMatrix(test.data$FTR, predicted_test_re)

##Accuracy = 67.43%

## -AF, -(HTSAge & HR), -(HTSAge & ATSAge)
#LDA: 66.51%, 66.97%, 66.06%
#Random Forest: 66.97%, 67.2%, 67.43%
#XGB: 65.82%, 66.51%, 67.2%

#We see that XGB takes a lot of time to run and mostly has accuracy lower
than the other 2
#In football, predictions need to be made in real-time and as fast as
possible
#Since XGB has lower accuracy compared to others and also takes more time
to run, we reject XGB
#Since in all iterations, random forest has slightly higher accuracy than
LDA, we choose random forest
#Random forest also avoids overfitting if new variables are added and is
more reliable model

## FINAL TREE
set.seed(007)
tree = rpart(FTR~HST + AST + HF + HC + AC + HY + AY + AR + HR + HTATT +
ATATT + HTHG + HTAG +
             HS + AS + TBHT + TBAT + HAWG + AAWG,
             data = train.data, control = rpart.control(cp = 0.005),
na.action = na.omit)

printcp(tree)
```

```
plotcp(tree)
opt_cp = tree$cptable[which.min(tree$cptable[,'xerror']),'CP']

set.seed(008)
opt_tree = rpart(FTR~HST + AST + HF + HC + AC + HY + AY + AR + HR + HTATT
+ ATATT + HTHG + HTAG +
                 HS + AS + TBHT + TBAT + HAWG + AAWG,
                 data = train.data, control = rpart.control(cp = 0.01),
na.action = na.omit)
rpart.plot(opt_tree)

##### ITERATIONS THAT LEAD TO THE FINAL SOLUTION (FOR DETAILED ANALYSIS)
#########################


##FEATURE SELECTION USING RANDOM FOREST
set.seed(2)
forest_in = randomForest(FTR~HST + AST + HF + AF + HC + AC + HY + AY + HR
+ AR + HTATT + ATATT + HTHG + HTAG + HTSAge +
                        ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG,
data = data1, ntree = 1140, importance = TRUE, na.action = na.omit)

importance(forest_in)

#Removing AF since it has negative mean decrease accuracy

##———————————————LDA (LINEAR DISCRIMINANT
ANALYSIS)——————————————————————————
# Implementing a Linear Discriminant Analysis

lda = lda(FTR~HST + AST + HF + HC + AC + HY + AY + HR + AR + HTATT +
ATATT + HTHG + HTAG + HTSAge +
            ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG, data =
train.data)

# Out of Sample Accuracy
predicted_test = predict(lda, test.data)
table(predicted_test$class)
table(test.data$FTR)
```

```
confusionMatrix(test.data$FTR, predicted_test$class)

## Accuracy = 66.51%

##———————————————QDA (QUADRATIC DISCRIMINANT
ANALYSIS)——————————————————————————
# Implementing a Quadratic Discriminant Analysis

qda = qda(FTR~HST + AST + HF + HC + AC + HY + AY + HR + AR + HTATT +
ATATT + HTHG + HTAG + HTSAge +
            ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG,
          data = train.data)

# Out of Sample Accuracy
predicted_test = predict(qda, test.data)
table(predicted_test$class)
table(test.data$FTR)
confusionMatrix(test.data$FTR, predicted_test$class)

## Accuracy = 61.7%

##———————————————DECISION TREE———————————————————————————————————
# Implementing a Decision Tree

tree = rpart(FTR~HST + AST + HF + HC + AC + HY + AY + HR + AR + HTATT +
ATATT + HTHG + HTAG + HTSAge +
               ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG,
             data = train.data, control = rpart.control(cp = 0.005),
na.action = na.omit)

rpart.plot(tree)
printcp(tree)
plotcp(tree)
opt_cp = tree$cptable[which.min(tree$cptable[,'xerror']),'CP']

opt_tree = rpart(FTR~HST + AST + HF + HC + AC + HY + AY + HR + AR + HTATT
+ ATATT + HTHG + HTAG + HTSAge +
                   ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG,
                 data = train.data, control = rpart.control(cp = opt_cp),
na.action = na.omit)
rpart.plot(opt_tree)

summary(opt_tree)

# Out of Sample Accuracy
predicted_test = predict(opt_tree, test.data, type = 'class')
table(predicted_test)
table(test.data$FTR)
confusionMatrix(test.data$FTR, predicted_test)

## Accuracy = 60.09%

##——————————————————————RANDOM FOREST—————————————————————————————————
# Implementing Random Forests

set.seed(23)
forest1 = randomForest(FTR~HST + AST + HF + HC + AC + HY + AY + HR + AR +
HTATT + ATATT + HTHG + HTAG + HTSAge +
                       ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG, data
= train.data, ntree = 1140, importance = TRUE, na.action = na.omit)

# Finding Optimal Value of mtry

mtry <- tuneRF(data1[, c(1:3,5:22)], data1$FTR, ntreeTry=500,
               stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
print(mtry)
print(best.m)

# Running Optimal Model
set.seed(24)
forest2 = randomForest(FTR~HST + AST + HF + HC + AC + HY + AY + HR + AR +
HTATT + ATATT + HTHG + HTAG + HTSAge +
                         ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG,
data = train.data, ntree = 1140, importance = TRUE, na.action = na.omit,
mtry = best.m)
```

```
# Out of Sample Accuracy

predicted_test = predict(forest2, test.data, type = "class")
table(predicted_test)
confusionMatrix(test.data$FTR, predicted_test)

## Accuracy = 67.2%

##———————————————————————XGBoost————————————————————————————————
# Implementing XGB

set.seed(123)
xgb = train(FTR~HST + AST + HF + HC + AC + HY + AY + HR + AR + HTATT +
ATATT + HTHG + HTAG + HTSAge +
              ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG, data =
train.data, method = "xgbTree", trControl = trainControl("repeatedcv",
number = 10))

# Out of Sample Accuracy
predicted_test = predict(xgb, test.data)
mean(predicted_test == test.data$FTR)

# Finding Relative Variable Importance

## Accuracy = 66.67%

### ACCURACY SUMMARY:
#LDA: 66.51%
#QDA: 61.7%
#Decision Trees: 60.09%
#Random Forest: 66.97%
#XGB: 65.82%

## We exclude QDA & Decision tree from our iterations because there is a
significant difference
## Between the accuracy % of these models and our top 3 models
### Now we experiment with our top 3 models i.e., LDA, XGB, and random
forest:
```

```
##FEATURE SELECTION USING RANDOM FOREST

set.seed(2)
forest_re = randomForest(FTR~HST + AST + HF + HC + AC + HY + AY + HR + AR
+ HTATT + ATATT + HTHG + HTAG + HTSAge +
                           ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG,
data = data1, ntree = 1140, importance = TRUE, na.action = na.omit)

forest_re
importance(forest_re)
##HTSAge & HR have negative coefficients which means they contribute
negatively to accuracy
###We re-run our top 3 models but this time excluding HTSAge & HR

##———————————————LDA (LINEAR DISCRIMINANT
ANALYSIS)——————————————————————————
# Implementing a Linear Discriminant Analysis
lda_re1 = lda(FTR~HST + AST + HF + HC + AC + HY + AY + AR + HTATT + ATATT
+ HTHG + HTAG +
                ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG,
          data = train.data)

# Out of Sample Accuracy
predicted_test_re = predict(lda_re1, test.data)
table(predicted_test_re$class)
table(test.data$FTR)
confusionMatrix(test.data$FTR, predicted_test_re$class)
## Accuracy = 66.97%

##——————————————————————RANDOM FOREST——————————————————————————————————
# Implementing Random Forests
# Finding Optimal Value of mtry

mtry_re1 <- tuneRF(data1[, c(1:3,5:8,10:18,20:22)], data1$FTR,
ntreeTry=500,
               stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
print(mtry_re1)
print(best.m)
```

```
# Running Optimal Model
set.seed(28)
forest1_re1 = randomForest(FTR~HST + AST + HF + HC + AC + HY + AY + AR +
HTATT + ATATT + HTHG + HTAG +
                             ATSAge + HS + AS + TBHT + TBAT + HAWG +
AAWG, data = train.data, ntree = 1140, importance = TRUE, na.action =
na.omit, mtry = best.m)

# Out of Sample Accuracy

predicted_test_re1 = predict(forest1_re1, test.data, type = "class")
table(predicted_test_re1)
confusionMatrix(test.data$FTR, predicted_test_re1)
##Accuracy = 67.2%

##———————————————————————XGBoost——————————————————————————————
# Implementing XGB

set.seed(125)
xgb_re1 = train(FTR~HST + AST + HF + HC + AC + HY + AY + AR + HTATT +
ATATT + HTHG + HTAG +
                  ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG, data =
train.data, method = "xgbTree", trControl = trainControl("repeatedcv",
number = 10))

# Out of Sample Accuracy
predicted_test = predict(xgb_re1, test.data)
mean(predicted_test == test.data$FTR)
##Accuracy = 66.51%

## ACCURACY SUMMARY
#LDA: 66.97%
#Random Forest: 67.2%
#XGB: 66.51%

set.seed(6)
forest_re1 = randomForest(FTR~HST + AST + HF + HC + AC + HY + AY + AR +
HTATT + ATATT + HTHG + HTAG +
                            ATSAge + HS + AS + TBHT + TBAT + HAWG + AAWG,
data = data1, ntree = 1140, importance = TRUE, na.action = na.omit)

importance(forest_re1)
##ATSAge has negative coefficient which means it might contribute
negatively to accuracy
###We re-run our top 3 models again but this time excluding ATSAge and
HTSAge, we include HR however:

##———————————————LDA (LINEAR DISCRIMINANT
ANALYSIS)——————————————————————————

lda_re = lda(FTR~HST + AST + HF + HC + AC + HY + AY + AR + HR + HTATT +
ATATT + HTHG + HTAG +
               HS + AS + TBHT + TBAT + HAWG + AAWG,
             data = train.data)

# Out of Sample Accuracy
predicted_test_re = predict(lda_re, test.data)
table(predicted_test_re$class)
table(test.data$FTR)
confusionMatrix(test.data$FTR, predicted_test_re$class)
## Accuracy = 66.06%

##——————————————————————RANDOM FOREST—————————————————————————————————
# Implementing Random Forests
# Finding Optimal Value of mtry

mtry_re <- tuneRF(data1[, c(1:3,5:18,21,22)], data1$FTR, ntreeTry=500,
                  stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
print(mtry_re)
print(best.m)

# Running Optimal Model
set.seed(58)
forest1_re = randomForest(FTR~HST + AST + HF + HC + AC + HY + AY + AR +
HR + HTATT + ATATT + HTHG + HTAG +
                            HS + AS + TBHT + TBAT + HAWG + AAWG, data =
train.data, ntree = 1140, importance = TRUE, na.action = na.omit, mtry =
best.m)

# Out of Sample Accuracy

predicted_test_re = predict(forest1_re, test.data, type = "class")
table(predicted_test_re)
confusionMatrix(test.data$FTR, predicted_test_re)
##Accuracy = 67.2%

##----------------------XGBoost------------------------------
# Implementing XGB

set.seed(130)
xgb_re = train(FTR~HST + AST + HF + HC + AC + HY + AY + AR + HR + HTATT +
ATATT + HTHG + HTAG +
               HS + AS + TBHT + TBAT + HAWG + AAWG, data = train.data,
method = "xgbTree", trControl = trainControl("repeatedcv", number = 10))

# Out of Sample Accuracy
predicted_test = predict(xgb_re, test.data)
mean(predicted_test == test.data$FTR)
##Accuracy = 67.2%

## ACCURACY SUMMARY: HTSAge & ATSAge
#LDA: 66.06%
#Random Forest: 67.2%
#XGB: 67.2%


####FINAL ACCURACIES#####

## -AF, -(HTSAge & HR), -(HTSAge & ATSAge)
#LDA: 66.51%, 66.97%, 66.06%
#Random Forest: 66.97%, 67.2%, 67.2%
#XGB: 65.82%, 66.51%, 67.2%

set.seed(104)
```

```
forest_re3 = randomForest(FTR~HST + AST + HF + HC + AC + HY + AY + AR +
HR + HTATT + ATATT + HTHG + HTAG +
                            HS + AS + TBHT + TBAT + HAWG + AAWG, data =
train.data, ntree = 1140, importance = TRUE, na.action = na.omit)

importance(forest_re3)

#When we run random forest feature selection again, we see that all
predictors left increase accuracy of model
#Hence, we stop with the iterations and decide which is the best model to
choose
```