World Cup Predictions Final Report

By: Nicholas McCubbin, Shijie Chen, Nikolas Maldonado

Overview

The main data set we used is from the FIFA database. It contains every match between international teams from 1992 to March of 2022. There are 433 games recorded in this dataset and 36 different variables recorded. We also incorporated 4 other datasets for some of our modeling techniques that included the schedules and the groups of each World Cup since 1992, team ranks, and player ratings.

The objective of our predictive model is to predict the outcome of the group stages of the world cup. While the data was mostly ready, it still required some cleaning for use in our model. This included changing data types to meet the information in the tables. We also created features:

- Is the game a World Cup game? (Yes/No)
- One feature used for predictions was the win or loss of any world cup game. We predicted on the average rank of the game played which was the 2022 FIFA world cup rankings of both teams divided by 2.
- A rank difference variable that was the difference between the home teams rank and the away teams' rank.
- The difference in rating of the top three players of the home team minus the top-three players of the away team for offense, defense, midfield, and the highest rated goalkeeper that each team had.

We trained different types of models including

- Logistic regression initially that predicted 67% accuracy on the training and holdout data. This was the highest performing model.
- We used a decision tree also which predicted to 66% accuracy. This model was slightly less accurate.
- A random forest and a gradient boosted model. The accuracy of these models was significantly lower.
- Naïve Bayes. This model had very low accuracy.

Since the accuracy of the logistic regression was the highest, we decided to use this as the predictive model for the outcome of the World Cup matches. We combined four different data sets that included all international matches since 1992 as well as the schedule and the groups of the World Cup. The other data sets included the ranks of the World Cup for the

ratings of the corresponding positions. We averaged the ratings from the last five years. The reason that we did this is because players from five years ago are most likely not on the team anymore. We wanted to have an accurate representation of the current team since the games dated back to 1992. The predictors used in this model were:

- Average rank of each team
- Average player-score of the top 3 offenders on each team
- Average player-score of the top 3 defenders on each team
- Average player-score of the top 3 midfielders on each team
- Average player-score of the top rated goalkeeper on each team

The outcomes of the group stages that the logistic regression model is show in a table below. A W or L in the Win_or_loss column indicates a win or a loss for the home team in that matchup. These predictions showed a 67% accuracy on the holdout data, which gives us confidence that the model can predict real world outcomes reasonably well for international matches.

Home	Away	Win_or_loss
Senegal	Netherlands	L
USA	Wales	L
France	Australia	W
Denmark	Tunisia	W
Mexico	Poland	L
Argentina	Saudi Arabia	W
Belgium	Canada	W
Spain	Costa Rica	W
Germany	Japan	W
Morocco	Croatia	L
Switzerland	Cameroon	L
Portugal	Ghana	W
Brazil	Serbia	W
Netherlands	Ecuador	W
England	USA	W
Tunisia	Australia	L
Poland	Saudi Arabia	W
France	Denmark	W
Argentina	Mexico	W
Japan	Costa Rica	W
Belgium	Morocco	L
Croatia	Canada	L
Spain	Germany	L
Cameroon	Serbia	L
Brazil	Switzerland	L
Portugal	Uruguay	L
Wales	England	L
Ecuador	Senegal	L
Australia	Denmark	L
Tunisia	France	L
Poland	Argentina	L
Saudi Arabia	Mexico	L
Croatia	Belgium	L
Canada	Morocco	L
Japan	Spain	L
Costa Rica	Germany	L
Ghana	Uruguay	L

In the following pages is the R code and output for our models. The commented lines explain what each chunk of code is performing.

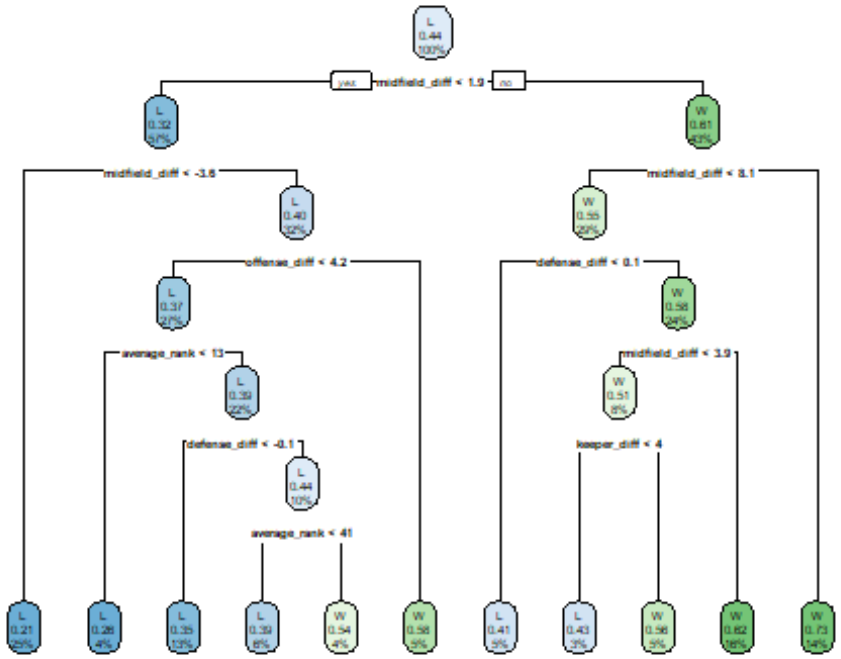
```
# Load libraries
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(readr)
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(rpart)
library(ggplot2)
library(arules)

## Loading required package: Matrix

##
## Attaching package: 'arules'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following objects are masked from 'package:base':
##
##     abbreviate, write

library(rpart.plot)
library(lubridate)

## Loading required package: timechange

##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:arules':
##
##     intersect, setdiff, union

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

# Load data
teams <- read_csv("teams.csv", col_names = TRUE, col_types = cols(.default =
"c"))
rankings <- read_csv("rankings.csv", col_names = TRUE, col_types =
cols(.default = "c"))
ratios <- read_csv("ratios.csv", col_names = TRUE, col_types = cols(.default
= "c"))
international_matches <- read_csv("international_matches.csv", col_names =
TRUE, col_types = cols(.default = "c"))

rankings <- data.frame(rankings)
international_matches$home_team_fifa_rank<-
as.numeric(international_matches$home_team_fifa_rank)
international_matches$away_team_fifa_rank <-
as.numeric(international_matches$away_team_fifa_rank)
international_matches$home_team_total_fifa_points <-
as.numeric(international_matches$home_team_total_fifa_points)
international_matches$away_team_total_fifa_points <-
as.numeric(international_matches$away_team_total_fifa_points)
international_matches$home_team_score <-
as.numeric(international_matches$home_team_score)
international_matches$away_team_score <-
as.numeric(international_matches$away_team_score)
international_matches$away_team_score <-
as.numeric(international_matches$away_team_score)

international_matches <- international_matches %>%
  mutate(rank_difference = home_team_fifa_rank - away_team_fifa_rank,
         average_rank = (home_team_fifa_rank + away_team_fifa_rank) / 2,
         point_difference = home_team_total_fifa_points -
away_team_total_fifa_points,
         is_stake = tournament != "Friendly",
         worldcup_game = grepl("FIFA World Cup", tournament),
         score_diff = as.numeric(home_team_score) -
as.numeric(away_team_score),
         offense_diff = as.numeric(home_team_mean_offense_score) -
as.numeric(away_team_mean_offense_score),
         defense_diff = as.numeric(home_team_mean_defense_score) -
as.numeric(away_team_mean_defense_score),
         midfield_diff = as.numeric(home_team_mean_midfield_score) -
as.numeric(away_team_mean_midfield_score),
         keeper_diff = as.numeric(home_team_goalkeeper_score) -
```

```
as.numeric(away_team_goalkeeper_score),
         win_or_loss = ifelse(score_diff > 0,"W","L") )

international_matches$home_team_goalkeeper_score <-
as.numeric(international_matches$home_team_goalkeeper_score)
international_matches$away_team_goalkeeper_score <-
as.numeric(international_matches$away_team_goalkeeper_score)
international_matches$home_team_mean_defense_score <-
as.numeric(international_matches$home_team_mean_defense_score)
international_matches$away_team_mean_defense_score <-
as.numeric(international_matches$away_team_mean_defense_score)
international_matches$home_team_mean_offense_score <-
as.numeric(international_matches$home_team_mean_offense_score)
international_matches$away_team_mean_offense_score <-
as.numeric(international_matches$away_team_mean_offense_score)
international_matches$home_team_mean_midfield_score <-
as.numeric(international_matches$home_team_mean_midfield_score)
international_matches$away_team_mean_midfield_score <-
as.numeric(international_matches$away_team_mean_midfield_score)

international_matches <- na.omit(international_matches)

set.seed(42)
rnd <- runif(nrow(international_matches))
train <- international_matches[rnd < 0.7,]
holdout <- international_matches[rnd >= 0.7,]


logreg_model <- train(factor(win_or_loss) ~ average_rank + rank_difference +
keeper_diff + midfield_diff + offense_diff + defense_diff,
                      data = train,
                      method = "glm",
                      family = "binomial")


logreg_predictions <- predict(logreg_model, holdout)

train$win_or_loss <- factor(train$win_or_loss)
logreg_predictions <- as.factor(logreg_predictions)
holdout$win_or_loss <- factor(holdout$win_or_loss)

levels(logreg_predictions) <- levels(holdout$win_or_loss)
levels(logreg_predictions) <- levels(train$win_or_loss)

performance <- confusionMatrix(logreg_predictions, holdout$win_or_loss)
performance

## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   L   W
##          L 572 284
##          W 158 326
##
##                Accuracy : 0.6701
##                  95% CI : (0.6443, 0.6953)
##     No Information Rate : 0.5448
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3235
##                                lo
##  Mcnemar's Test P-Value : 2.754e-09
##
##             Sensitivity : 0.7836
##             Specificity : 0.5344
##          Pos Pred Value : 0.6682
##          Neg Pred Value : 0.6736
##              Prevalence : 0.5448
##          Detection Rate : 0.4269
##    Detection Prevalence : 0.6388
##       Balanced Accuracy : 0.6590
##
##        'Positive' Class : L
##
```

```
dt_model <- rpart(factor(win_or_loss) ~ rank_difference
                                      + keeper_diff
                                      + midfield_diff
                                      + offense_diff
                                      + defense_diff
                                      + average_rank,
                          data = train,
                          method = "class",
                          control = list(minsplit = 100,
                                         cp =.0015985,
                                         minbucket = 100))


rpart.plot(dt_model)
```




```
summary(dt_model)

## Call:
## rpart(formula = factor(win_or_loss) ~ rank_difference + keeper_diff +
##     midfield_diff + offense_diff + defense_diff + average_rank,
##     data = train, method = "class", control = list(minsplit = 100,
##         cp = 0.0015985, minbucket = 100))
##   n= 2963
##
##            CP nsplit rel error    xerror       xstd
## 1 0.209923664      0 1.0000000 1.0000000 0.02063645
## 2 0.009923664      1 0.7900763 0.8458015 0.02010503
## 3 0.008778626      3 0.7702290 0.8229008 0.01999070
## 4 0.005343511      5 0.7526718 0.8061069 0.01990078
## 5 0.002290076      7 0.7419847 0.7961832 0.01984521
## 6 0.001598500     10 0.7351145 0.8007634 0.01987109
##
## Variable importance
##   midfield_diff    defense_diff    offense_diff rank_difference
keeper_diff
##              34              20              18              13
13
##    average_rank
##               2
##
## Node number 1: 2963 observations,    complexity param=0.2099237
##   predicted class=L  expected loss=0.4421195  P(node) =1
```

```
##     class counts:  1653  1310
##    probabilities: 0.558 0.442
##   left son=2 (1688 obs) right son=3 (1275 obs)
##   Primary splits:
##       midfield_diff   < 1.9    to the left,  improve=122.93300, (0
missing)
##       defense_diff    < 1.35   to the left,  improve=122.37990, (0
missing)
##       offense_diff    < 2.85   to the left,  improve=121.00810, (0
missing)
##       rank_difference < -2.5   to the right, improve=101.53370, (0
missing)
##       keeper_diff     < 1.5    to the left,  improve= 77.59214, (0
missing)
##   Surrogate splits:
##       defense_diff    < 2.35   to the left,  agree=0.822, adj=0.586, (0
split)
##       offense_diff    < 1.65   to the left,  agree=0.802, adj=0.540, (0
split)
##       rank_difference < -3.5   to the right, agree=0.754, adj=0.428, (0
split)
##       keeper_diff     < 3.5    to the left,  agree=0.738, adj=0.392, (0
split)
##       average_rank    < 1.75   to the right, agree=0.570, adj=0.002, (0
split)
##
## Node number 2: 1688 observations,    complexity param=0.008778626
##   predicted class=L  expected loss=0.3169431  P(node) =0.5696929
##     class counts:  1153   535
##    probabilities: 0.683 0.317
##   left son=4 (747 obs) right son=5 (941 obs)
##   Primary splits:
##       midfield_diff   < -3.65  to the left,  improve=30.55094, (0 missing)
##       offense_diff    < -4.85  to the left,  improve=24.29903, (0 missing)
##       defense_diff    < -2.35  to the left,  improve=22.23986, (0 missing)
##       rank_difference < 9.5    to the right, improve=18.01059, (0 missing)
##       keeper_diff     < -2.5   to the left,  improve=14.76210, (0 missing)
##   Surrogate splits:
##       defense_diff    < -5.1   to the left,  agree=0.757, adj=0.451, (0
split)
##       offense_diff    < -5.15  to the left,  agree=0.732, adj=0.395, (0
split)
##       keeper_diff     < -3.5   to the left,  agree=0.701, adj=0.325, (0
split)
##       rank_difference < 12.5   to the right, agree=0.697, adj=0.315, (0
split)
##       average_rank    < 68.75  to the right, agree=0.562, adj=0.009, (0
split)
##
## Node number 3: 1275 observations,    complexity param=0.009923664
```

```
##   predicted class=W  expected loss=0.3921569  P(node) =0.4303071
##     class counts:   500   775
##    probabilities: 0.392 0.608
##   left son=6 (862 obs) right son=7 (413 obs)
##   Primary splits:
##       midfield_diff   < 8.1    to the left,  improve=17.170370, (0
missing)
##       defense_diff    < 8.75   to the left,  improve=16.091640, (0
missing)
##       offense_diff    < 2.15   to the left,  improve=12.829870, (0
missing)
##       rank_difference < -11.5  to the right, improve=10.415840, (0
missing)
##       keeper_diff     < 3.5    to the left,  improve= 7.733211, (0
missing)
##   Surrogate splits:
##       defense_diff    < 9.65   to the left,  agree=0.760, adj=0.259, (0
split)
##       offense_diff    < 10.35  to the left,  agree=0.740, adj=0.196, (0
split)
##       keeper_diff     < 14.5   to the left,  agree=0.738, adj=0.191, (0
split)
##       rank_difference < -46.5  to the right, agree=0.731, adj=0.169, (0
split)
##
## Node number 4: 747 observations
##   predicted class=L  expected loss=0.210174  P(node) =0.2521093
##     class counts:   590   157
##    probabilities: 0.790 0.210
##
## Node number 5: 941 observations,    complexity param=0.008778626
##   predicted class=L  expected loss=0.4017003  P(node) =0.3175835
##     class counts:   563   378
##    probabilities: 0.598 0.402
##   left son=10 (794 obs) right son=11 (147 obs)
##   Primary splits:
##       offense_diff    < 4.15   to the left,  improve=10.858210, (0
missing)
##       rank_difference < -23.5  to the right, improve= 9.058257, (0
missing)
##       defense_diff    < 1.55   to the left,  improve= 5.148539, (0
missing)
##       average_rank    < 12.75  to the left,  improve= 3.402199, (0
missing)
##       keeper_diff     < 7.5    to the left,  improve= 2.426125, (0
missing)
##   Surrogate splits:
##       rank_difference < -49.5  to the right, agree=0.854, adj=0.068, (0
split)
##       defense_diff    < 6.25   to the left,  agree=0.851, adj=0.048, (0
```

```
split)
##
## Node number 6: 862 observations,    complexity param=0.009923664
##   predicted class=W  expected loss=0.4489559  P(node) =0.2909214
##     class counts:   387   475
##    probabilities: 0.449 0.551
##   left son=12 (140 obs) right son=13 (722 obs)
##   Primary splits:
##       defense_diff    < 0.1    to the left,  improve=6.922406, (0 missing)
##       offense_diff    < 2.15   to the left,  improve=5.598561, (0 missing)
##       midfield_diff   < 4.65   to the left,  improve=5.045066, (0 missing)
##       rank_difference < -25.5  to the right, improve=4.665881, (0 missing)
##       average_rank    < 40.25  to the left,  improve=2.586462, (0 missing)
##   Surrogate splits:
##       keeper_diff     < -10.5  to the left,  agree=0.849, adj=0.071, (0
split)
##       offense_diff    < -5.15  to the left,  agree=0.849, adj=0.071, (0
split)
##       rank_difference < 42     to the right, agree=0.843, adj=0.036, (0
split)
##       average_rank    < 101.25 to the right, agree=0.840, adj=0.014, (0
split)
##
## Node number 7: 413 observations
##   predicted class=W  expected loss=0.2736077  P(node) =0.1393858
##     class counts:   113   300
##    probabilities: 0.274 0.726
##
## Node number 10: 794 observations,    complexity param=0.002290076
##   predicted class=L  expected loss=0.3690176  P(node) =0.2679717
##     class counts:   501   293
##    probabilities: 0.631 0.369
##   left son=20 (128 obs) right son=21 (666 obs)
##   Primary splits:
##       average_rank    < 12.75  to the left,  improve=3.774296, (0 missing)
##       rank_difference < -13.5  to the right, improve=2.968642, (0 missing)
##       defense_diff    < 1.45   to the left,  improve=2.942841, (0 missing)
##       keeper_diff     < 2.5    to the left,  improve=1.558993, (0 missing)
##       offense_diff    < -0.85  to the left,  improve=1.334272, (0 missing)
##
## Node number 11: 147 observations
##   predicted class=W  expected loss=0.4217687  P(node) =0.04961188
##     class counts:    62    85
##    probabilities: 0.422 0.578
##
## Node number 12: 140 observations
##   predicted class=L  expected loss=0.4071429  P(node) =0.04724941
##     class counts:    83    57
##    probabilities: 0.593 0.407
##
```

```
## Node number 13: 722 observations,    complexity param=0.005343511
##   predicted class=W  expected loss=0.4210526  P(node) =0.243672
##     class counts:   304   418
##    probabilities: 0.421 0.579
##   left son=26 (250 obs) right son=27 (472 obs)
##   Primary splits:
##       midfield_diff   < 3.9    to the left,  improve=3.849797, (0 missing)
##       offense_diff    < 2.15   to the left,  improve=2.668874, (0 missing)
##       defense_diff    < 7.45   to the left,  improve=2.642726, (0 missing)
##       average_rank    < 40.25  to the left,  improve=2.638201, (0 missing)
##       rank_difference < -39.5  to the right, improve=2.454461, (0 missing)
##   Surrogate splits:
##       rank_difference < 3.5    to the right, agree=0.675, adj=0.060, (0
split)
##       offense_diff    < -1.5   to the left,  agree=0.673, adj=0.056, (0
split)
##       defense_diff    < 1.25   to the left,  agree=0.673, adj=0.056, (0
split)
##       average_rank    < 5.75   to the left,  agree=0.662, adj=0.024, (0
split)
##
## Node number 20: 128 observations
##   predicted class=L  expected loss=0.2578125  P(node) =0.04319946
##     class counts:    95    33
##    probabilities: 0.742 0.258
##
## Node number 21: 666 observations,    complexity param=0.002290076
##   predicted class=L  expected loss=0.3903904  P(node) =0.2247722
##     class counts:   406   260
##    probabilities: 0.610 0.390
##   left son=42 (371 obs) right son=43 (295 obs)
##   Primary splits:
##       defense_diff    < -0.1   to the left,  improve=3.051655, (0 missing)
##       rank_difference < 32.5   to the right, improve=2.731654, (0 missing)
##       offense_diff    < -2.65  to the left,  improve=1.871430, (0 missing)
##       keeper_diff     < 1.5    to the left,  improve=1.784387, (0 missing)
##       average_rank    < 41.25  to the left,  improve=1.505083, (0 missing)
##   Surrogate splits:
##       offense_diff    < -0.85  to the left,  agree=0.649, adj=0.207, (0
split)
##       rank_difference < 2.5    to the right, agree=0.611, adj=0.122, (0
split)
##       keeper_diff     < 4.5    to the left,  agree=0.611, adj=0.122, (0
split)
##       midfield_diff   < 1.25   to the left,  agree=0.581, adj=0.054, (0
split)
##       average_rank    < 17.25  to the right, agree=0.568, adj=0.024, (0
split)
##
## Node number 26: 250 observations,    complexity param=0.005343511
```

```
##   predicted class=W  expected loss=0.492  P(node) =0.08437395
##     class counts:   123   127
##    probabilities: 0.492 0.508
##   left son=52 (100 obs) right son=53 (150 obs)
##   Primary splits:
##       keeper_diff     < 3.5    to the left,  improve=2.028000, (0 missing)
##       rank_difference < -14.5  to the right, improve=1.487902, (0 missing)
##       offense_diff    < 4.65   to the left,  improve=1.487902, (0 missing)
##       defense_diff    < 4.45   to the left,  improve=1.416765, (0 missing)
##       average_rank    < 27.25  to the right, improve=1.405181, (0 missing)
##   Surrogate splits:
##       rank_difference < 5.5    to the right, agree=0.656, adj=0.14, (0
split)
##       defense_diff    < 2.1    to the left,  agree=0.620, adj=0.05, (0
split)
##       offense_diff    < 14.45  to the right, agree=0.608, adj=0.02, (0
split)
##       average_rank    < 3.25   to the left,  agree=0.604, adj=0.01, (0
split)
##
## Node number 27: 472 observations
##   predicted class=W  expected loss=0.3834746  P(node) =0.159298
##     class counts:   181   291
##    probabilities: 0.383 0.617
##
## Node number 42: 371 observations
##   predicted class=L  expected loss=0.3477089  P(node) =0.1252109
##     class counts:   242   129
##    probabilities: 0.652 0.348
##
## Node number 43: 295 observations,    complexity param=0.002290076
##   predicted class=L  expected loss=0.4440678  P(node) =0.09956126
##     class counts:   164   131
##    probabilities: 0.556 0.444
##   left son=86 (184 obs) right son=87 (111 obs)
##   Primary splits:
##       average_rank    < 40.75  to the left,  improve=3.312580, (0 missing)
##       keeper_diff     < 2.5    to the left,  improve=2.221677, (0 missing)
##       offense_diff    < -0.85  to the left,  improve=1.725997, (0 missing)
##       rank_difference < 5.5    to the right, improve=1.359397, (0 missing)
##       defense_diff    < 2.75   to the right, improve=0.757362, (0 missing)
##   Surrogate splits:
##       rank_difference < 27.5   to the left,  agree=0.675, adj=0.135, (0
split)
##       offense_diff    < 3.65   to the left,  agree=0.644, adj=0.054, (0
split)
##       defense_diff    < 5.25   to the left,  agree=0.641, adj=0.045, (0
split)
##       keeper_diff     < -12    to the right, agree=0.627, adj=0.009, (0
split)
```

```
##       midfield_diff   < -2.7   to the right, agree=0.627, adj=0.009, (0
split)
##
## Node number 52: 100 observations
##   predicted class=L  expected loss=0.43  P(node) =0.03374958
##     class counts:    57    43
##    probabilities: 0.570 0.430
##
## Node number 53: 150 observations
##   predicted class=W  expected loss=0.44  P(node) =0.05062437
##     class counts:    66    84
##    probabilities: 0.440 0.560
##
## Node number 86: 184 observations
##   predicted class=L  expected loss=0.3858696  P(node) =0.06209922
##     class counts:   113    71
##    probabilities: 0.614 0.386
##
## Node number 87: 111 observations
##   predicted class=W  expected loss=0.4594595  P(node) =0.03746203
##     class counts:    51    60
##    probabilities: 0.459 0.541
```

```
#Checking Training Accuracy
t_pred = predict(dt_model,holdout,type="class")
t = holdout$win_or_loss
accuracy = sum(t_pred == t)/length(t)
print(accuracy)
```

```
## [1] 0.6768657
```

```
#Checking Holdout Accuracy to see if Overfit
t_pred = predict(dt_model,train,type="class")
t = train$win_or_loss
accuracy1 = sum(t_pred == t)/length(t)
print(accuracy1)
```

```
## [1] 0.6749916
```

Adding the Average of the last 5 years stats to the teams DF

```
team_names <- unique(international_matches$home_team)
# Create an empty data frame to store the results
results_df <- data.frame()

# Loop through each team in the team_names list
for (team in team_names){
  # Filter the data to include only rows where the home_team is equal to the
current team
  filtered_data <- filter(international_matches, home_team == team)
```

```
  # Summarize the filtered data by calculating the mean of each relevant
column
  summarized_data <- filtered_data %>%
                     group_by(home_team) %>%
                     summarize(
                               home_team_mean_midfield_score =
mean(home_team_mean_midfield_score),
                               home_team_mean_defense_score =
mean(home_team_mean_defense_score),
                               home_team_goalkeeper_score =
mean(home_team_goalkeeper_score),
                               home_team_mean_offense_score =
mean(home_team_mean_offense_score))

  # Use bind_rows() to add the summarized data to the results data frame
  results_df <- bind_rows(results_df, summarized_data)
}



# For loop to grab ranks for every team from 2022 rankings

countries <- results_df$home_team
for (i in 1:length(countries)) {
  # Pull the rank for the current country
  results_df$rank[i] <- rankings %>%
    filter(year(rank_date) == 2022, country_full == countries[i]) %>%
    head(1) %>% summarise(as.numeric(rank))
}


# Renaming Columns in results_df

colnames(results_df) <- c( "Country","Midfield_score", "Defense_score"
,"Goalkeeper_score",
"Offense_score" , "rank")
```

Implementing the Schedule

```
schedule <- read.csv("schedule.csv")

#Pulling the data
home_teams <- unique(international_matches$home_team)
for (team in home_teams){
schedule$Home_Offense[schedule$Home == team] <-
results_df$Offense_score[results_df$Country == team]
schedule$Home_Defense[schedule$Home == team] <-
results_df$Defense_score[results_df$Country == team]
schedule$Home_Midfield[schedule$Home == team] <-
```

```
results_df$Midfield_score[results_df$Country == team]
schedule$Home_Keeper[schedule$Home == team] <-
results_df$Goalkeeper_score[results_df$Country == team]
}


away_teams <- unique(international_matches$away_team)
for (team in away_teams){
schedule$Away_Offense[schedule$Away == team] <-
results_df$Offense_score[results_df$Country == team]
schedule$Away_Defense[schedule$Away == team] <-
results_df$Defense_score[results_df$Country == team]
schedule$Away_Midfield[schedule$Away == team] <-
results_df$Midfield_score[results_df$Country == team]
schedule$Away_Keeper[schedule$Away == team] <-
results_df$Goalkeeper_score[results_df$Country == team]
}
```

Re-Making Features

```
#Away ranks
countries <- schedule$Away
for (i in 1:length(countries)) {
  # Pull the rank for the current country
  schedule$rank_away[i] <- rankings %>%
    filter(year(rank_date) == 2022, country_full == countries[i]) %>%
    head(1) %>% summarise(as.numeric(rank))
}

#Home Ranks

countries <- schedule$Home
for (i in 1:length(countries)) {
  # Pull the rank for the current country
  schedule$rank_home[i] <- rankings %>%
    filter(year(rank_date) == 2022, country_full == countries[i]) %>%
    head(1) %>% summarise(as.numeric(rank))
}



schedule <- schedule[1:49,] %>%
  mutate(rank_difference = as.numeric(rank_home) - as.numeric(rank_away),
         average_rank = ( as.numeric(rank_home) + as.numeric(rank_away) / 2
),
         keeper_diff = Home_Keeper - Away_Keeper,
         offense_diff = Home_Offense - Away_Offense,
         defense_diff = Home_Defense - Away_Defense,
         midfield_diff = Home_Midfield - Away_Midfield)
```

```
schedule <- na.omit(schedule)



for (game in 1:length(schedule)){
      schedule$Win_or_loss[game] <- predict(logreg_model,schedule[game,])
}

schedule$Win_or_loss <- ifelse(schedule$Win_or_loss ==2,'W','L')
```

Gradient Boosted Model

```
library(gbm)

## Loaded gbm 2.1.8.1

train$win_or_loss <- ifelse(train$win_or_loss == "W",1,0)
# Create the gbm model
gbm_model <- gbm(factor(win_or_loss) ~ average_rank + rank_difference +
keeper_diff + midfield_diff + offense_diff + defense_diff,
                 data = train)

## Distribution not specified, assuming bernoulli ...

# Make predictions on the holdout set
gbm_predictions <- predict(gbm_model, holdout, n.trees = 100)

# Convert the predictions to a factor and set the levels to match the holdout
set
gbm_predictions <- as.factor(gbm_predictions)
levels(gbm_predictions) <- levels(holdout$win_or_loss)

# Compute the performance of the model using a confusion matrix
performance <- confusionMatrix(gbm_predictions, holdout$win_or_loss)
performance

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   L   W
##          L 730 610
##          W   0   0
##
##                Accuracy : 0.5448
##                  95% CI : (0.5177, 0.5717)
##     No Information Rate : 0.5448
##     P-Value [Acc > NIR] : 0.5113
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
```

```
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.5448
##          Neg Pred Value :    NaN
##              Prevalence : 0.5448
##          Detection Rate : 0.5448
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : L
##
```

Random Forrest

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## 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
```

```
# Create the random forest model
rf_model <- randomForest(factor(win_or_loss) ~ average_rank + rank_difference
+ keeper_diff + midfield_diff + offense_diff + defense_diff,
                         data = train)

# Make predictions on the holdout set
rf_predictions <- predict(rf_model, holdout)

# Convert the predictions to a factor and set the levels to match the holdout
set
rf_predictions <- as.factor(rf_predictions)
levels(rf_predictions) <- levels(holdout$win_or_loss)


performance <- confusionMatrix(rf_predictions, holdout$win_or_loss)
performance
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   L   W
##          L 524 282
##          W 206 328
##
##                Accuracy : 0.6358
##                  95% CI : (0.6094, 0.6616)
##     No Information Rate : 0.5448
##     P-Value [Acc > NIR] : 9.089e-12
##
##                   Kappa : 0.2582
##
##  Mcnemar's Test P-Value : 0.0006861
##
##             Sensitivity : 0.7178
##             Specificity : 0.5377
##          Pos Pred Value : 0.6501
##          Neg Pred Value : 0.6142
##              Prevalence : 0.5448
##          Detection Rate : 0.3910
##    Detection Prevalence : 0.6015
##       Balanced Accuracy : 0.6278
##
##        'Positive' Class : L
##
```

Naive Bayes

```
library(e1071)


nb_model <- naiveBayes(factor(win_or_loss) ~ average_rank + rank_difference +
keeper_diff + midfield_diff + offense_diff + defense_diff,
                       data = train)


nb_predictions <- predict(nb_model, holdout)


nb_predictions <- as.factor(nb_predictions)
levels(nb_predictions) <- levels(factor(holdout$win_or_loss))
```