Data Science Project

Sparsh Jain

08/05/24

Introduction

Basketball analytics has evolved significantly, leveraging data to enhance team strategies and predict game outcomes. In this project, I delve into an extensive basketball dataset to uncover key performance metrics and forecast playoff dynamics. Our analysis encompasses critical metrics such as offensive and defensive rates, providing insights into team strengths and weaknesses. Utilizing these metrics, I developed predictive models to estimate win percentages for teams in the playoffs, offering a data-driven approach to assessing team success. Additionally, I built a model to predict the number of games between two teams in the playoffs, adding a layer of foresight to playoff predictions. This project aims to demonstrate the power of data analytics in sports and provide valuable tools for teams and analysts to make informed decisions during the crucial playoff season.

Note:

Throughout this document, any `season` column represents the year each season started. For example, the 2015-16 season will be in the dataset as 2015. For most of the rest of the project, we will refer to a season by just this number (e.g. 2015) instead of the full text (e.g. 2015-16).

Setup and Data

```
library(tidyverse)
library(caret)
options(warn=-1)
# Note, you will likely have to change these paths. If your data is in the same folder as this project,
# the paths will likely be fixed for you by deleting ../../Data/awards_project/ from each string.
player_data <- read_csv("player_game_data.csv")
team_data <- read_csv("team_game_data.csv")
```

Part 1 – Data Cleaning

Question 1

QUESTION: What was the Warriors' Team offensive and defensive eFG% in the 2015-16 regular season? Remember that this is in the data as the 2015 season.

```
# Here and for all future questions, feel free to add as many code chunks as you like. Do NOT put echo =

warriors_offense <- team_data %>% filter(season == 2015 & off_team == 'GSW')  #filtering the data
```

```r
warriors_defense <- team_data %>% filter(season == 2015 & def_team == 'GSW')

# Calculating offensive eFG%
warriors_offense <- warriors_offense %>%
  mutate(FGM = fg2made + fg3made,
         threePM = fg3made,
         FGA = fgattempted,
         eFG = (FGM + 0.5 * threePM) / FGA)

# Calculating defensive eFG%
warriors_defense <- warriors_defense %>%
  mutate(FGM = fg2made + fg3made,
         threePM = fg3made,
         FGA = fgattempted,
         def_eFG = (FGM + 0.5 * threePM) / FGA)

# Calculate the overall offensive eFG%
  overall_offensive_efg <- round(100*mean(warriors_offense$eFG, na.rm = TRUE),1)

# Calculate the overall defensive eFG%
overall_defensive_efg <- round(100*mean(warriors_defense$def_eFG, na.rm = TRUE),1)

# Print the results
cat("Offensive eFG% for the Warriors in the 2015-16 season:", overall_offensive_efg, "%\n")
```

```
## Offensive eFG% for the Warriors in the 2015-16 season: 55.6 %
```

```r
cat("Defensive eFG% for the Warriors in the 2015-16 season:", overall_defensive_efg, "%\n")
```

```
## Defensive eFG% for the Warriors in the 2015-16 season: 48 %
```

ANSWER 1:

Offensive: 55.6% eFG
Defensive: 48.0% eFG

Question 2

QUESTION: What percent of the time does the team with the higher eFG% in a given game win that game? Use games from the 2014-2023 regular seasons. If the two teams have an exactly equal eFG%, remove that game from the calculation.

```r
regular_season_data <- team_data %>%
  filter(season >= 2014 & season <= 2023 & gametype == 2)  #filtering data

regular2_season_data <- regular_season_data %>%
         mutate(off_eFG = (fgmade + 0.5 * fg3made) / fgattempted) %>%
  select(nbagameid,
         off_win,
         fgmade,
         fg3made,
         fgattempted,
```

```
         off_eFG) %>%
          group_by(nbagameid) %>%
          filter(off_eFG > min(off_eFG))  # removing teams with equal or lesser efg%

S = sum(regular2_season_data$off_win)

perc = round(100 * S/nrow(regular2_season_data),1)


cat("Percentage of the time the team with the higher eFG% wins:", perc, "%\n")
```

```
## Percentage of the time the team with the higher eFG% wins: 81.6 %
```

ANSWER 2:

81.6%

Question 3

QUESTION: What percent of the time does the team with more offensive rebounds in a given game win that game? Use games from the 2014-2023 regular seasons. If the two teams have an exactly equal number of offensive rebounds, remove that game from the calculation.

```
regular_season_data <- team_data %>%
  filter(season >= 2014 & season <= 2023 & gametype == 2)

regular3_season_data <- regular_season_data %>%
  select(nbagameid,
         off_win,
         reboffensive) %>%
          group_by(nbagameid) %>%
          filter(reboffensive > min(reboffensive))

S = sum(regular3_season_data$off_win)

perc = round(100 * S/nrow(regular3_season_data),1)

cat("Percentage of the time the team with more offensive rebounds wins:", perc, "%\n")
```

```
## Percentage of the time the team with more offensive rebounds wins: 46.2 %
```

ANSWER 3:

46.2%

Question 4

QUESTION: Do you have any theories as to why the answer to question 3 is lower than the answer to question 2?

ANSWER 4:

Higher eFG% results in more efficient scoring per shot attempt, indicating improved shot selection and shooting abilities. Higher offensive rebounds, on the other hand, result in more possessions but do not always lead to better efficiency. Defensive impact, game strategy, and opponent strengths all influence which component results in a higher win %.

Question 5

QUESTION: Look at players who played at least 25% of their possible games in a season and scored at least 25 points per game played. Of those player-seasons, what percent of games were they available for on average? Use games from the 2014-2023 regular seasons.

For example:

- Ja Morant does not count in the 2023-24 season, as he played just 9 out of 82 games this year, even though he scored 25.1 points per game.
- Chet Holmgren does not count in the 2023-24 season, as he played all 82 games this year but scored 16.5 points per game.
- LeBron James does count in the 2023-24 season, as he played 71 games and scored 25.7 points per game.

```
df=player_data %>% filter(season >= 2014 & season <= 2023 & gametype == 2) %>%
  select(seconds,nbapersonid,missed,season,nbapersonid,points) %>%
  group_by(season,nbapersonid) %>%
  summarise(sum_missed = sum(missed), sum_played = sum(seconds > 0), games = sum(seconds >= 0), games_p
            av_pts = sum(points)/sum_played) %>%
  mutate(avg_points = coalesce(av_pts, 0), games_per=coalesce(games_per, 0)) %>%
  filter(games_per>0.25, avg_points>25) %>%
  mutate(avail_per=(games-sum_missed)/games)
```

```
## `summarise()` has grouped output by 'season'. You can override using the
## `.groups` argument.
```

```
S_games= sum(df$avail_per)
num = nrow(df)

perc = round(S_games/num * 100,1)
cat(perc,"% of games were played by them and was available on average.")
```

```
## 82.9 % of games were played by them and was available on average.
```

ANSWER 5:

82.9% of games

Question 6

QUESTION: What % of playoff series are won by the team with home court advantage? Give your answer by round. Use playoffs series from the 2014-2022 seasons. Remember that the 2023 playoffs took place during the 2022 season (i.e. 2022-23 season).

```r
# Creating function to return the playoffs round matches for any season.

round_matches <- function(team_data,year1,year2) {
playoff_matches <- team_data %>% filter(season>=year1 & season<=year2 & gametype==4)

team_counts = playoff_matches %>% group_by(season,off_team_name) %>%
  summarise(Count = n_distinct(def_team_name),.groups="drop")       # Counts of each team in playoffs

R1 <-team_counts %>% filter(Count == 1)                #Teams played round 1.

R1_matches = playoff_matches %>% inner_join(R1,by=c('off_team_name','season')) %>%
  select(season,off_team_name,nbagameid,off_win,off_home,def_team_name,def_win,def_home) %>%
  mutate(home_win = off_win * off_home + def_win * def_home)        # Filtering Round 1 matches.


playoff_matches = playoff_matches %>% filter(!(nbagameid %in% R1_matches$nbagameid))   # Removing round

R2 <-team_counts %>% filter(Count == 2)                 #Teams played round 2.

R2_matches = playoff_matches %>% inner_join(R2,by=c('off_team_name','season')) %>%
  select(season,off_team_name,nbagameid,off_win,off_home,def_team_name,def_win,def_home) %>%
  mutate(home_win = off_win * off_home + def_win * def_home)        # Filtering Round 2 matches.


playoff_matches = playoff_matches %>% filter(!(nbagameid %in% R2_matches$nbagameid))    # Removing roun

R3 <-team_counts %>% filter(Count == 3)               #Teams played conference finals

R3_matches = playoff_matches %>% inner_join(R3,by=c('off_team_name','season')) %>%
  select(season,off_team_name,nbagameid,off_win,off_home,def_team_name,def_win,def_home) %>%
  mutate(home_win = off_win * off_home + def_win * def_home) # Filtering Conference final matches.



playoff_matches = playoff_matches %>% filter(!(nbagameid %in% R3_matches$nbagameid)) %>%
  select(off_team_name,nbagameid,off_win,off_home,def_team_name,def_win,def_home) %>%
  mutate(home_win = off_win * off_home + def_win * def_home)                          # Removing confe



return(list(R1=R1_matches,R2=R2_matches,R3=R3_matches,R4=playoff_matches))
}

all_matches=round_matches(team_data,2013,2021)

R1_perc <- round(sum(all_matches$R1$home_win)*100/nrow(all_matches$R1),1)
cat("\nRound 1:",R1_perc,"%")
```

```
##
## Round 1: 59 %
```

```r
R2_perc <- round(sum(all_matches$R2$home_win)*100/nrow(all_matches$R2),1)
cat("\nRound 2:",R2_perc,"%")
```

```
##
## Round 2: 58.2 %
```

```
R3_perc <- round(sum(all_matches$R3$home_win)*100/nrow(all_matches$R3),1)
cat("\nConference Finals:",R3_perc,"%")
```

```
##
## Conference Finals: 63.7 %
```

```
R4_perc <- round(sum(all_matches$R4$home_win)*100/nrow(all_matches$R4),1)
cat("\nFinals:",R4_perc,"%")
```

```
##
## Finals: 52.9 %
```

ANSWER 6:

Round 1: 59.0%
Round 2: 58.2%
Conference Finals: 63.7%
Finals: 52.9%

Question 7

QUESTION: Among teams that had at least a +5.0 net rating in the regular season, what percent of them made the second round of the playoffs the following year? Among those teams, what percent of their top 5 total minutes played players (regular season) in the +5.0 net rating season played in that 2nd round playoffs series? Use the 2014-2021 regular seasons to determine the +5 teams and the 2015-2022 seasons of playoffs data.

For example, the Thunder had a better than +5 net rating in the 2023 season. If we make the 2nd round of the playoffs next season (2024-25), we would qualify for this question. Our top 5 minutes played players this season were Shai Gilgeous-Alexander, Chet Holmgren, Luguentz Dort, Jalen Williams, and Josh Giddey. If three of them play in a hypothetical 2nd round series next season, it would count as 3/5 for this question.

Hint: The definition for net rating is in the data dictionary.

```
team_regular_off <- team_data %>% filter(season>=2014 & season<=2021 & gametype==2) %>%
  mutate(ORTG=points*100/possessions)              # Filtering seasons and calculating ORTG for each ma

team_net5 <- team_regular_off %>%
  inner_join(team_regular_off,by=c("nbagameid","off_team_name"="def_team_name"),suffix=c("","_opponent")
  mutate(NET=ORTG-ORTG_opponent) %>%
  group_by(season,off_team_name) %>%
  summarize(NET=mean(NET)) %>% filter(NET>5)      # Filtering teams that had +5 Net rating in the previ
```

```
## `summarise()` has grouped output by 'season'. You can override using the
## `.groups` argument.
```

```r
second_round_teams <- team_data %>% filter(season>=2014 & season<=2021 & gametype==4) %>%
  group_by(off_team_name,season) %>%
  summarise(unique = n_distinct(def_team),.groups="drop") %>% filter(unique>=2)            # Filtering

qualified_teams <- team_net5 %>%
  inner_join(second_round_teams, by = c("off_team_name" = "off_team_name", "season" = "season"))   # Fi

Percentage=round(nrow(qualified_teams)*100/nrow(team_net5),1)
cat("\nPercent of +5.0 net rating teams making the 2nd round next year:",Percentage)
```

```
##
## Percent of +5.0 net rating teams making the 2nd round next year: 90.9
```

```r
#--------------------------------------Player's Analysis-------------------------------------------

top_players= player_data %>% filter(season>=2014 & season<=2021 & gametype==2) %>%
  semi_join(qualified_teams,by=c("season","team_name"="off_team_name")) %>%
  group_by(season,team_name,nbapersonid) %>%
  summarise(total_played=sum(seconds)) %>%
  mutate(rank = rank(desc(total_played))) %>%
  filter(rank <= 5)                                          # Filtering top players of the qualified
```

```
## `summarise()` has grouped output by 'season', 'team_name'. You can override
## using the `.groups` argument.
```

```r
all_matches=round_matches(team_data,2014,2021)$R2            # Filtering Round 2 matches for those te

playoffs_players= player_data %>%
  inner_join(all_matches,by=c("nbagameid")) %>%
  semi_join(top_players,by=c("team_name","nbapersonid","season.x"="season")) %>%
  group_by(season.x,nbapersonid,team_name) %>%
  summarise(sum_sec=sum(seconds))                            # Filtering out the players that satisfi
```

```
## `summarise()` has grouped output by 'season.x', 'nbapersonid'. You can override
## using the `.groups` argument.
```

```r
played = sum(playoffs_players$sum_sec>0)
percent=round(played*100/nrow(playoffs_players),1)

cat("\nPercent of top 5 minutes played players who played in those 2nd round series:",percent)
```

```
##
## Percent of top 5 minutes played players who played in those 2nd round series: 98.7
```

ANSWER 7:

Percent of +5.0 net rating teams making the 2nd round next year: 90.9%
Percent of top 5 minutes played players who played in those 2nd round series: 98.7%

Part 2 – Playoffs Series Modeling

For this part, you will work to fit a model that predicts the winner and the number of games in a playoffs series between any given two teams.

Include, as part of your answer:

- A brief written overview of how your model works, targeted towards a decision maker in the front office without a strong statistical background.
- What you view as the strengths and weaknesses of your model.
- How you'd address the weaknesses if you had more time and/or more data.
- Apply your model to the 2024 NBA playoffs (2023 season) and create a high quality visual (a table, a plot, or a plotly) showing the 16 teams' (that made the first round) chances of advancing to each round.

ANSWER:

This is an intentionally open ended question, and there are multiple approaches we can take. Here are a few notes and specifications:

1. My final output includes the probability of each team winning the series. For example: "Team A has a 30% chance to win and team B has a 70% chance." instead of "Team B will win." I am also predicting the number of games in the series.
2. I have only used data available prior to the start of the series. For example, I am not using a team's stats from the 2016-17 season to predict a playoffs series from the 2015-16 season.

ANSWER : I have divided this task to two different models. Both the models used for the task was Logistic Classification.

```
colnames(team_data)
```

```
##  [1] "season"             "gametype"           "nbagameid"
##  [4] "gamedate"           "offensivenbateamid" "off_team_name"
##  [7] "off_team"           "off_home"           "off_win"
## [10] "defensivenbateamid" "def_team_name"      "def_team"
## [13] "def_home"           "def_win"            "fg2made"
## [16] "fg2missed"          "fg2attempted"       "fg3made"
## [19] "fg3missed"          "fg3attempted"       "fgmade"
## [22] "fgmissed"           "fgattempted"        "ftmade"
## [25] "ftmissed"           "ftattempted"        "reboffensive"
## [28] "rebdefensive"       "reboundchance"      "assists"
## [31] "stealsagainst"      "turnovers"          "blocksagainst"
## [34] "defensivefouls"     "offensivefouls"     "shootingfoulsdrawn"
## [37] "possessions"        "points"             "shotattempts"
## [40] "andones"            "shotattemptpoints"
```

```
match_data= team_data %>% inner_join(team_data,by=c("nbagameid"),suffix=c("","_opponent")) %>% filter(!
  distinct(nbagameid, .keep_all=TRUE) %>%
  select(-gamedate,-offensivenbateamid,-off_team,-defensivenbateamid,-def_team,-def_win,-def_home,-seas
         -offensivenbateamid_opponent,-gamedate_opponent,-off_team_name_opponent,-off_team_opponent,-of
         -defensivenbateamid_opponent,-def_team_name_opponent,-def_team_opponent,-def_home_opponent,-de
colnames(match_data)
```

```
##  [1] "season"                    "gametype"
##  [3] "nbagameid"                 "off_team_name"
##  [5] "off_home"                  "off_win"
##  [7] "def_team_name"             "fg2made"
##  [9] "fg2missed"                 "fg2attempted"
## [11] "fg3made"                   "fg3missed"
## [13] "fg3attempted"              "fgmade"
## [15] "fgmissed"                  "fgattempted"
## [17] "ftmade"                    "ftmissed"
## [19] "ftattempted"               "reboffensive"
## [21] "rebdefensive"              "reboundchance"
## [23] "assists"                   "stealsagainst"
## [25] "turnovers"                 "blocksagainst"
## [27] "defensivefouls"            "offensivefouls"
## [29] "shootingfoulsdrawn"        "possessions"
## [31] "points"                    "shotattempts"
## [33] "andones"                   "shotattemptpoints"
## [35] "fg2made_opponent"          "fg2missed_opponent"
## [37] "fg2attempted_opponent"     "fg3made_opponent"
## [39] "fg3missed_opponent"        "fg3attempted_opponent"
## [41] "fgmade_opponent"           "fgmissed_opponent"
## [43] "fgattempted_opponent"      "ftmade_opponent"
## [45] "ftmissed_opponent"         "ftattempted_opponent"
## [47] "reboffensive_opponent"     "rebdefensive_opponent"
## [49] "reboundchance_opponent"    "assists_opponent"
## [51] "stealsagainst_opponent"    "turnovers_opponent"
## [53] "blocksagainst_opponent"    "defensivefouls_opponent"
## [55] "offensivefouls_opponent"   "shootingfoulsdrawn_opponent"
## [57] "possessions_opponent"      "points_opponent"
## [59] "shotattempts_opponent"     "andones_opponent"
## [61] "shotattemptpoints_opponent"
```

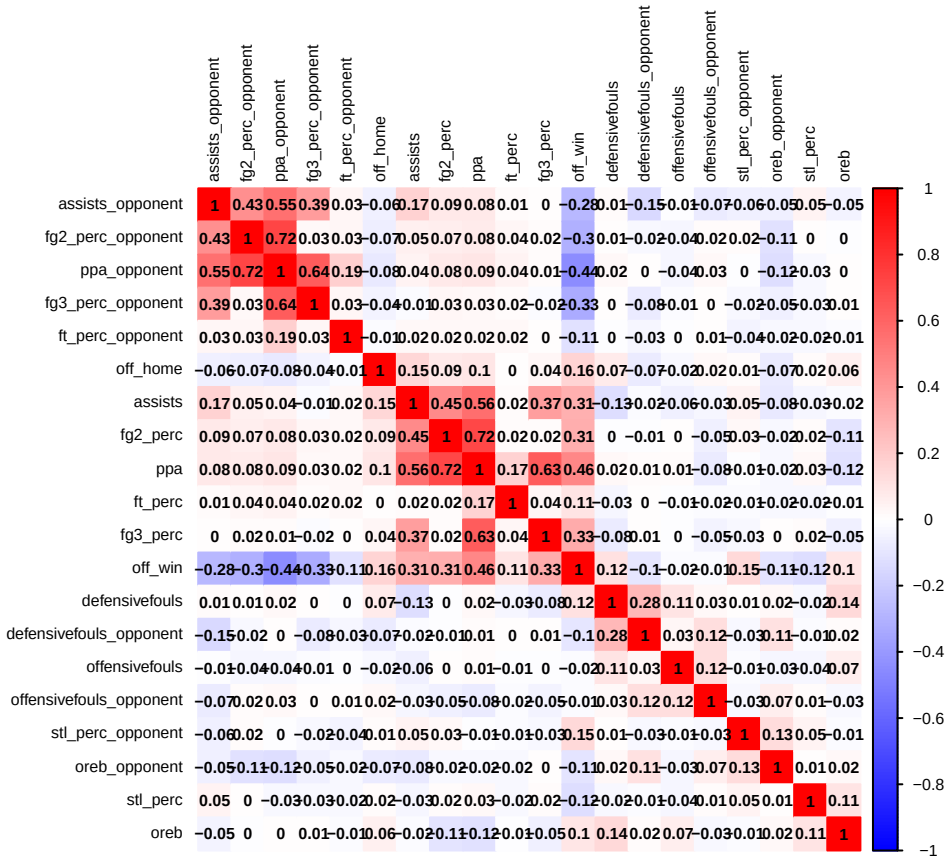
```
match_data_upd=match_data %>% mutate(fg2_perc=fg2made/fg2attempted, fg3_perc=fg3made/fg3attempted, ft_p
                                     ppa=shotattemptpoints/shotattempts, stl_perc=stealsagainst/possessi
                                     oreb=reboffensive/reboundchance,
                                     fg2_perc_opponent=fg2made_opponent/fg2attempted_opponent,
                                     fg3_perc_opponent=fg3made_opponent/fg3attempted_opponent, ft_perc_
                                     ppa_opponent=shotattemptpoints_opponent/shotattempts_opponent,
                                     stl_perc_opponent=stealsagainst_opponent/possessions_opponent,
                                     oreb_opponent=reboffensive_opponent/reboundchance_opponent,
                                    ) %>%
  select(season,gametype,off_team_name,def_team_name,off_home,fg2_perc,fg3_perc,ft_perc,ppa,stl_perc,
         oreb,assists,defensivefouls,offensivefouls,
         fg2_perc_opponent,fg3_perc_opponent,ft_perc_opponent,ppa_opponent,stl_perc_opponent,
         oreb_opponent,assists_opponent,defensivefouls_opponent,offensivefouls_opponent,
         off_win)
```

```
test_data=match_data_upd %>% filter(season<=2022 & season>=2021)
train_data=match_data_upd %>% filter(season<2021) %>% select(-season,-gametype,-off_team_name,-def_team
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
# Calculate the correlation matrix
corr_matrix <- cor(train_data)
options(repr.plot.width = 10, repr.plot.height = 10)
# Generate the heatmap
corrplot(corr_matrix, method = "color", type = "full",
         tl.col = "black", tl.cex = 0.5, # Reduce text label size
         number.cex = 0.5, # Reduce correlation coefficient size
         addCoef.col = "black", # Add correlation coefficient
         col = colorRampPalette(c("blue", "white", "red"))(200),
         order = "hclust", # Cluster similar variables together
         cl.cex = 0.5)
```




```
logitModel <- glm(off_win ~ .-1, data = train_data, family = binomial)
summary(logitModel)
```

```
##
```

```
## Call:
## glm(formula = off_win ~ . - 1, family = binomial, data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3768  -0.1362   0.0004   0.1534   2.9873
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## off_home                   0.32753    0.08967   3.653 0.000259 ***
## fg2_perc                   8.34183    1.66837   5.000 5.73e-07 ***
## fg3_perc                   9.13889    1.19377   7.655 1.93e-14 ***
## ft_perc                    4.41427    0.45624   9.675  < 2e-16 ***
## ppa                       25.86965    1.45281  17.807  < 2e-16 ***
## stl_perc                 -51.85661    1.97640 -26.238  < 2e-16 ***
## oreb                      21.49364    0.81959  26.225  < 2e-16 ***
## assists                    0.05105    0.01092   4.675 2.94e-06 ***
## defensivefouls             0.21974    0.01281  17.159  < 2e-16 ***
## offensivefouls            -0.50852    0.03373 -15.076  < 2e-16 ***
## fg2_perc_opponent        -10.94092    1.82551  -5.993 2.06e-09 ***
## fg3_perc_opponent         -9.51689    1.22051  -7.797 6.32e-15 ***
## ft_perc_opponent          -4.36315    0.47738  -9.140  < 2e-16 ***
## ppa_opponent             -24.66508    1.50145 -16.428  < 2e-16 ***
## stl_perc_opponent         52.33845    1.98613  26.352  < 2e-16 ***
## oreb_opponent            -21.61688    0.80560 -26.833  < 2e-16 ***
## assists_opponent          -0.06225    0.01101  -5.653 1.58e-08 ***
## defensivefouls_opponent   -0.20961    0.01271 -16.490  < 2e-16 ***
## offensivefouls_opponent    0.47114    0.03337  14.118  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13462.3  on 9711  degrees of freedom
## Residual deviance:  3430.5  on 9692  degrees of freedom
## AIC: 3468.5
##
## Number of Fisher Scoring iterations: 8
```

```
test_data_2022= test_data %>% filter(season=="2022" & gametype == 4)
test_data_2021=test_data %>% filter(season=="2021" & gametype== 4)

stats_2022 = test_data %>% filter(season==2022 & gametype == 2) %>% select(-season,-gametype,-def_team_
  group_by(off_team_name,off_home) %>% summarize(across(where(is.numeric), mean, .names = "avg_{col}"))
  select(-contains("_opponent"))
```

```
## `summarise()` has grouped output by 'off_team_name'. You can override using the
## `.groups` argument.
```

```
test_data_2022_playoff= test_data_2022 %>% distinct(off_team_name,def_team_name,off_home, .keep_all=TRU
  select(off_team_name,def_team_name,off_home,off_win) %>%
  inner_join(stats_2022,by=c("off_team_name","off_home")) %>%
  inner_join(stats_2022,by=c("def_team_name"="off_team_name"),suffix=c("","_opponent")) %>% filter(off_
```

```
  select(-off_home_opponent,-off_team_name,-def_team_name,-avg_off_win,-avg_off_win_opponent) %>%
  rename_with(~ gsub("avg_", "", .x), .cols = everything())

test_2022_y=test_data_2022_playoff$off_win
test_data_2022_playoff = test_data_2022_playoff %>% select(-off_win)
```

```
stats_2021 = test_data %>% filter(season==2021 & gametype == 2) %>% select(-season,-gametype,-def_team_
  group_by(off_team_name,off_home) %>% summarize(across(where(is.numeric), mean, .names = "avg_{col}"))
  select(-contains("_opponent"))
```

```
## `summarise()` has grouped output by 'off_team_name'. You can override using the
## `.groups` argument.
```

```
test_data_2021_playoff= test_data_2021 %>% distinct(off_team_name,def_team_name,off_home, .keep_all=TRU
  select(off_team_name,def_team_name,off_home,off_win) %>%
  inner_join(stats_2021,by=c("off_team_name","off_home")) %>%
  inner_join(stats_2021,by=c("def_team_name"="off_team_name"),suffix=c("","_opponent")) %>% filter(off_
  select(-off_home_opponent,-off_team_name,-def_team_name,-avg_off_win,-avg_off_win_opponent) %>%
  rename_with(~ gsub("avg_", "", .x), .cols = everything())

test_2021_y=test_data_2021_playoff$off_win
test_data_2021_playoff = test_data_2021_playoff %>% select(-off_win)
```

```
test2022_matches_played=test_data_2022 %>% group_by(off_team_name,def_team_name) %>% summarize(num_matc
```

```
## `summarise()` has grouped output by 'off_team_name'. You can override using the
## `.groups` argument.
```

```
predicted_probabilities <- predict(logitModel, newdata = test_data_2022_playoff , type = "response")
predicted_classes_improved <- ifelse(predicted_probabilities > 0.6, 1, 0)

# Confusion matrix
confMatrixImproved <- confusionMatrix(as.factor(predicted_classes_improved), as.factor(test_2022_y))
print(confMatrixImproved)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 21 13
##          1  3 11
##
##                Accuracy : 0.6667
##                  95% CI : (0.5159, 0.796)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.01465
##
##                   Kappa : 0.3333
##
##  Mcnemar's Test P-Value : 0.02445
##
```

```
##             Sensitivity : 0.8750
##             Specificity : 0.4583
##          Pos Pred Value : 0.6176
##          Neg Pred Value : 0.7857
##              Prevalence : 0.5000
##          Detection Rate : 0.4375
##    Detection Prevalence : 0.7083
##       Balanced Accuracy : 0.6667
##
##        'Positive' Class : 0
##
```

```
results_2022 = data.frame(actual=as.factor(test_2022_y), predicted= as.factor(predicted_classes_improve
```

```
cross_entropy_loss_2022 <- -mean(test_2022_y * log(predicted_probabilities) + (1 - test_2022_y) * log(1
cat(cross_entropy_loss_2022)
```

```
## 0.8681511
```

```
predicted_probabilities_2021 <- predict(logitModel, newdata = test_data_2021_playoff , type = "response
predicted_classes_improved_2021 <- ifelse(predicted_probabilities_2021 > 0.6, 1, 0)

# Confusion matrix
confMatrixImproved_2021 <- confusionMatrix(as.factor(predicted_classes_improved_2021), as.factor(test_2
print(confMatrixImproved_2021)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 8 8
##          1 7 7
##
##                Accuracy : 0.5
##                  95% CI : (0.313, 0.687)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.5722
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.5333
##             Specificity : 0.4667
##          Pos Pred Value : 0.5000
##          Neg Pred Value : 0.5000
##              Prevalence : 0.5000
##          Detection Rate : 0.2667
##    Detection Prevalence : 0.5333
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

```
cross_entropy_loss_2021 <- -mean(test_2021_y * log(predicted_probabilities_2021) + (1 - test_2021_y) * l
cross_entropy_loss_2021
```

```
## [1] 0.8945466
```

```
results_2021 = data.frame(actual=as.factor(test_2021_y), predicted= as.factor(predicted_classes_improved
                          prob=as.factor(predicted_probabilities_2021*100))
```

##MODEL 1: This model basically predicts the winner of the series for the playoff matches. I used Logistic Classification for this task. This takes the season performance of the teams into consideration and predicts the winner of the series by analyzing the season stats like field goal percentage, offensive rebounds percentage, fouls etc for both the teams. This model is trained on every matches of the previous seasons and playoffs, while predicting the winners, it takes the current seasons stats as an input.

Strength: The model definitely works on giving more importance to the recent form rather than their previous records. it also gives the Home-Advantage to the teams and predicts winner based on that parameter This model does not takes the future stats for predicting the current season prediction.

Weakness: It does not give the importance to player stats and also the injuries of top players during playoffs.

If I had more time and data, I would mainly focus on creating more parameters which I didn't in this task because of less data. I had to restrict my parameters according to the data available.

Accuracy of Model 1: 58.3%

#THE Number of GAMES in PLAYOFFS:

For predicting the number of games between two teams in the playoffs, I used Logistic Multi-Classification model. This model divides the season data and playoff data separately. The season stats is taken as independent parameters and the number of games played by the teams in the playoffs as the dependent variable. This model had lesser data available as the season and playoff data was present for only few seasons.

Strength: The model definitely works on giving more importance to the recent form rather than their previous records. It also gives the Home-Advantage to the teams and predicts winner based on that parameter. This model does not takes the future stats for predicting the current season prediction. Also, uses regularization techniques to prevent over-fitting the data.

Weakness: Similar to previous model,it does not give the importance to player stats and also the injuries of top players during playoffs.

If I had more time and data, I would mainly focus on creating more parameters which I didn't in this task because of less data. I had to restrict my parameters according to the data available. I would have also considered the player importance factor to predicting the number of games. i would have also included parameters which considers results from previous matches of that series. Parameters like Head-to-Head win can also be considered with more data.

```
match_data_upd2=match_data %>% mutate(fg2_perc=fg2made/fg2attempted, fg3_perc=fg3made/fg3attempted, ft_
                                      ppa=shotattemptpoints/shotattempts, stl_perc=stealsagainst/possessi
                                      oreb=reboffensive/reboundchance, DRTG=points_opponent/possessions_
                                      fg2_perc_opponent=fg2made_opponent/fg2attempted_opponent,
                                      fg3_perc_opponent=fg3made_opponent/fg3attempted_opponent, ft_perc_
                                      ppa_opponent=shotattemptpoints_opponent/shotattempts_opponent,
                                      stl_perc_opponent=stealsagainst_opponent/possessions_opponent,
                                      ORTG_opponent=points_opponent/possessions_opponent, DRTG_opponent=p
                                      oreb_opponent=reboffensive_opponent/reboundchance_opponent
                                     ) %>%
```

```
  select(season,gametype,off_team_name,def_team_name,off_home,fg2_perc,fg3_perc,ft_perc,ppa,stl_perc,,OR
         oreb,assists,defensivefouls,offensivefouls,
         fg2_perc_opponent,fg3_perc_opponent,ft_perc_opponent,ppa_opponent,stl_perc_opponent,
         ORTG_opponent,DRTG_opponent,blocksagainst_opponent, andones_opponent,
         oreb_opponent,assists_opponent,defensivefouls_opponent,offensivefouls_opponent,
         off_win)
season_data=match_data_upd2 %>% filter(season>="2014" & gametype==2)

playoff_df <- team_data %>% filter(season >= 2014 & gametype == 4) %>%
  group_by(season, off_team_name, def_team_name) %>%
  summarise(count = n(), .groups = "drop") %>%
  rowwise() %>%
  mutate(
    Sorted1 = min(off_team_name, def_team_name),
    Sorted2 = max(off_team_name, def_team_name)
  ) %>%
  ungroup() %>%
  select(off_team_name = Sorted1, def_team_name = Sorted2, count, season) %>%
  distinct()



stats2_seasonwise = season_data %>% select(-def_team_name) %>%
  group_by(season,off_team_name) %>% summarize(across(where(is.numeric), mean, .names = "avg_{col}")) %>
  select(-contains("_opponent"))
```

```
## `summarise()` has grouped output by 'season'. You can override using the
## `.groups` argument.
```

```
final_data2= playoff_df %>%
  left_join(stats2_seasonwise,by=c("off_team_name","season")) %>%
  left_join(stats2_seasonwise,by=c("def_team_name"="off_team_name","season"),suffix=c("","_opponent")) %
  rename_with(~ gsub("avg_", "", .x), .cols = everything()) %>% drop_na() %>% select(-count, count)



train_data = final_data2 %>% filter(season<=2020) %>% select(-season,-gametype,-off_home,-off_home_oppon
test_data= final_data2 %>% filter(season>=2021 & season<=2022) %>% select(-season,-gametype,-off_home,-o
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-8
```

```r
# Prepare training data
x_train <- as.matrix(train_data[, -29])  # Exclude the response variable
y_train <- as.factor(train_data$count)

# Prepare test data
x_test <- as.matrix(test_data[, -29])    # Exclude the response variable
y_test <- as.factor(test_data$count)

# Fit Lasso (L1) Regularized Multinomial Logistic Regression
model_lasso <- cv.glmnet(x_train, y_train, family = "multinomial", alpha = 1)

# Fit Ridge (L2) Regularized Multinomial Logistic Regression
model_ridge <- cv.glmnet(x_train, y_train, family = "multinomial", alpha = 0)

# Select the best lambda for Lasso (L1) regularization
best_lambda_lasso <- model_lasso$lambda.min

# Select the best lambda for Ridge (L2) regularization
best_lambda_ridge <- model_ridge$lambda.min

# Predictions using the best lambda for Lasso (L1) and Ridge (L2) models on test data
predictions_lasso <- predict(model_lasso, newx = x_test, s = best_lambda_lasso, type = "class")
predictions_ridge <- predict(model_ridge, newx = x_test, s = best_lambda_ridge, type = "class")

# Evaluate accuracy or other metrics on test data
accuracy_lasso <- mean(predictions_lasso == y_test)
accuracy_ridge <- mean(predictions_ridge == y_test)

# Print results
print("Lasso (L1) Regularization:")
```

```
## [1] "Lasso (L1) Regularization:"
```

```r
print(paste("Best Lambda:", best_lambda_lasso))
```

```
## [1] "Best Lambda: 0.120607046479078"
```

```r
print(paste("Accuracy on Test Data:", accuracy_lasso))
```

```
## [1] "Accuracy on Test Data: 0.4"
```

```r
print("Ridge (L2) Regularization:")
```

```
## [1] "Ridge (L2) Regularization:"
```

```r
print(paste("Best Lambda:", best_lambda_ridge))
```

```
## [1] "Best Lambda: 120.607046479078"
```

```r
print(paste("Accuracy on Test Data:", accuracy_ridge))
```

```
## [1] "Accuracy on Test Data: 0.4"
```

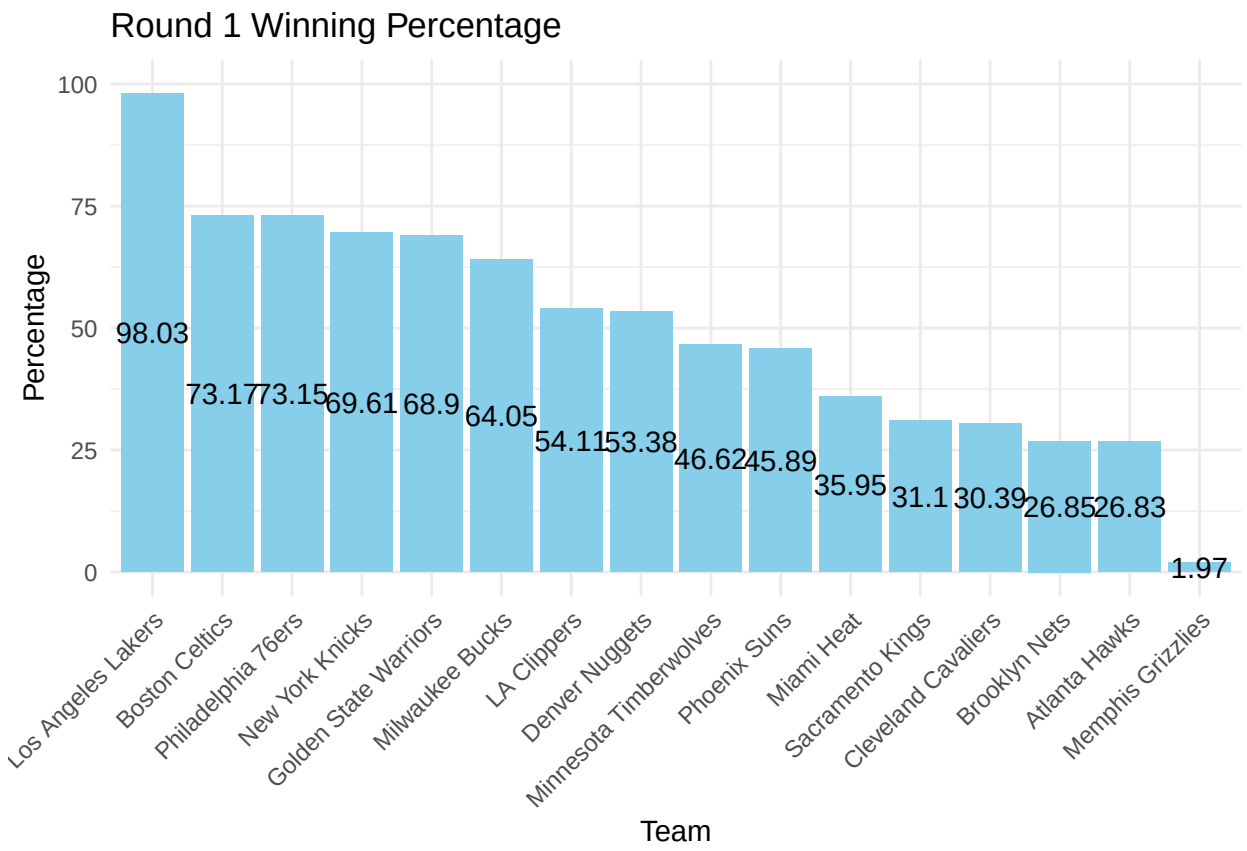
```r
teams_2024 <- c("Milwaukee Bucks","Miami Heat","Cleveland Cavaliers","New York Knicks","Philadelphia 76
test_data_2024= team_data %>% filter(season==2023 & gametype == 2 & (off_team_name %in% teams_2024)) %>%
  select(-gametype,-nbagameid,-offensivenbateamid,-defensivenbateamid,-def_home) %>%
  mutate(fg2_perc=fg2made/fg2attempted, fg3_perc=fg3made/fg3attempted, ft_perc=ifelse(ftattempted==0, 0
                                    ppa=shotattemptpoints/shotattempts, stl_perc=stealsagainst/possess
                                    oreb=reboffensive/reboundchance) %>% group_by(season,off_team_name
  summarize(across(where(is.numeric), mean)) %>% select(season,off_team_name,off_home,fg2_perc,fg3_perc
```

```
## `summarise()` has grouped output by 'season', 'off_team_name'. You can override
## using the `.groups` argument.
```

```r
round1_2024 <- data.frame(
  off_team_name = c("Milwaukee Bucks", "Milwaukee Bucks", "Cleveland Cavaliers", "Cleveland Cavaliers",
  def_team_name = c("Miami Heat","Miami Heat","New York Knicks","New York Knicks","Brooklyn Nets","Brook
  off_home=rep(c(0, 1), times = 8)
)

round1_m_2024 <- round1_2024 %>% inner_join(test_data_2024,by=c("off_team_name","off_home")) %>%
  inner_join(test_data_2024,by=c("def_team_name"="off_team_name"),suffix=c("","_opponent")) %>% filter(
  select(-off_home_opponent,-season,-season_opponent)

train_data_2024=round1_m_2024 %>% select(-off_team_name,-def_team_name)

predicted_probabilities_2024 <- predict(logitModel, newdata = train_data_2024 , type = "response")
predicted_classes_2024 <- ifelse(predicted_probabilities_2024 > 0.6, 1, 0)

results_2024 = data.frame(round1_2024, prob=as.numeric(as.character(as.factor(predicted_probabilities_2

results_2024_upd= results_2024 %>% group_by(off_team_name,def_team_name) %>% summarize(prob_win=round(m
```

```
## `summarise()` has grouped output by 'off_team_name'. You can override using the
## `.groups` argument.
```

```r
temp= data.frame(off_team_name=results_2024_upd$def_team_name, def_team_name=results_2024_upd$off_team_
results_2024_upd <- rbind(results_2024_upd, temp) %>% select(-def_team_name)

library(ggplot2)
ggplot(results_2024_upd, aes(x = reorder(off_team_name,-prob_win), y = prob_win)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Round 1 Winning Percentage", x = "Team", y = "Percentage") +
  geom_text(aes(label = prob_win), position = position_stack(vjust = 0.5), color = "black") +
  ylim(0, 100) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



##ROUND 2

```
round2_2024 <- data.frame(
  off_team_name = c("Milwaukee Bucks", "Milwaukee Bucks", "Philadelphia 76ers","Philadelphia 76ers","Den
  def_team_name = c("New York Knicks","New York Knicks","Boston Celtics","Boston Celtics","LA Clippers"
  off_home=rep(c(0, 1), times = 4)
)
```

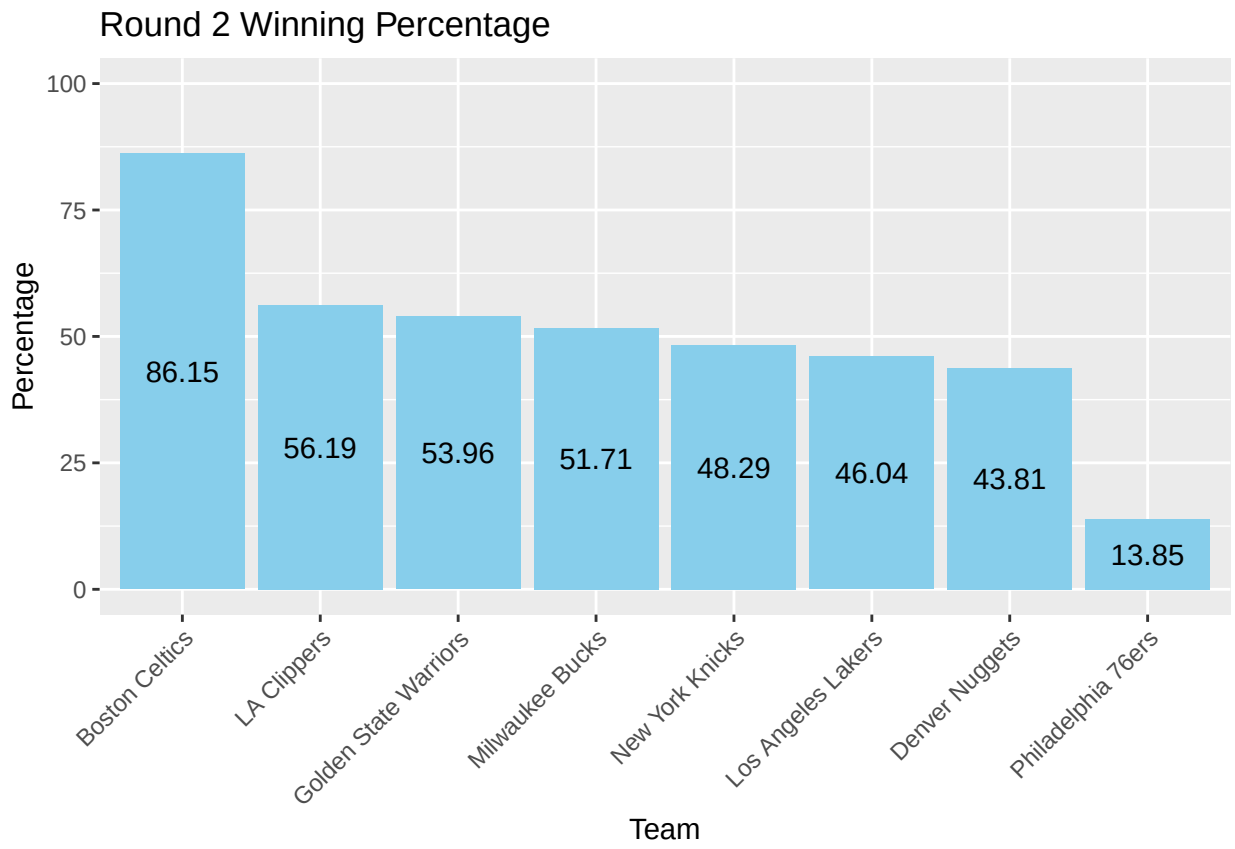
```
round2_m_2024 <- round2_2024 %>% inner_join(test_data_2024,by=c("off_team_name","off_home")) %>%
  inner_join(test_data_2024,by=c("def_team_name"="off_team_name"),suffix=c("","_opponent")) %>% filter(
  select(-off_home_opponent,-season,-season_opponent)

train2_data_2024=round2_m_2024 %>% select(-off_team_name,-def_team_name)
```

```
predicted_probabilities_R2_2024 <- predict(logitModel, newdata = train2_data_2024 , type = "response")
predicted_classes_R2_2024 <- ifelse(predicted_probabilities_R2_2024 > 0.6, 1, 0)

results_R2_2024 = data.frame(round2_m_2024, prob=as.numeric(as.character(as.factor(predicted_probabilit

results_R2_2024_upd= results_R2_2024 %>% group_by(off_team_name,def_team_name) %>% summarize(prob_win=r
```

```
## `summarise()` has grouped output by 'off_team_name'. You can override using the
## `.groups` argument.
```

```
temp= data.frame(off_team_name=results_R2_2024_upd$def_team_name, def_team_name=results_R2_2024_upd$off_
results_R2_2024_upd <- rbind(results_R2_2024_upd, temp) %>% select(-def_team_name)
```

```
ggplot(results_R2_2024_upd, aes(x = reorder(off_team_name,-prob_win), y = prob_win)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  geom_text(aes(label = prob_win), position = position_stack(vjust = 0.5), color = "black")+
  labs(title = "Round 2 Winning Percentage", x = "Team", y = "Percentage") +
  ylim(0, 100)  +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



#Conference Finals

```
round3_2024 <- data.frame(
  off_team_name = c("Milwaukee Bucks", "Milwaukee Bucks","LA Clippers","LA Clippers"),
  def_team_name = c("Boston Celtics","Boston Celtics","Golden State Warriors","Golden State Warriors"),
  off_home=rep(c(0, 1), times = 2)
)
```

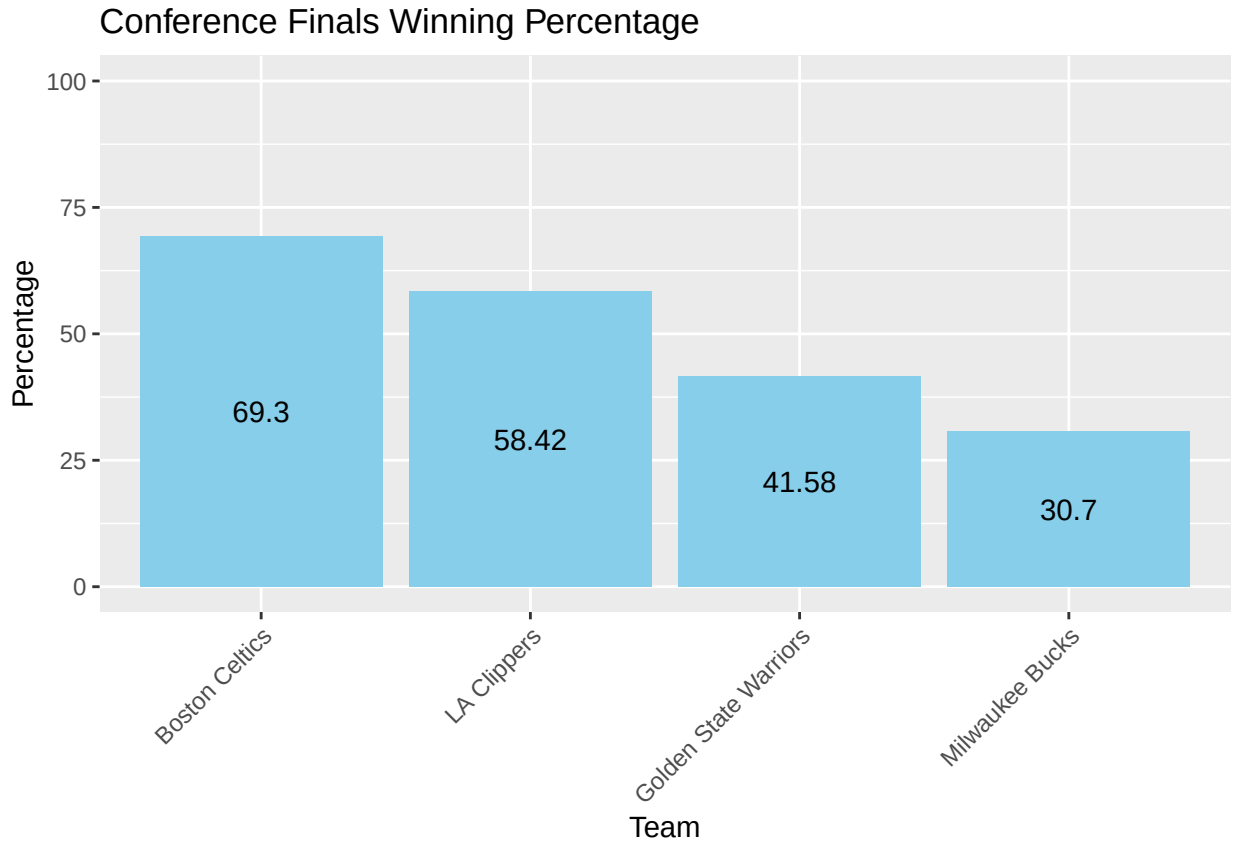
```
round3_m_2024 <- round3_2024 %>% inner_join(test_data_2024,by=c("off_team_name","off_home")) %>%
  inner_join(test_data_2024,by=c("def_team_name"="off_team_name"),suffix=c("","_opponent")) %>% filter(
  select(-off_home_opponent,-season,-season_opponent)

train3_data_2024=round3_m_2024 %>% select(-off_team_name,-def_team_name)
```

```
predicted_probabilities_R3_2024 <- predict(logitModel, newdata = train3_data_2024 , type = "response")
predicted_classes_R3_2024 <- ifelse(predicted_probabilities_R3_2024 > 0.6, 1, 0)

results_R3_2024 = data.frame(round3_m_2024, prob=as.numeric(as.character(as.factor(predicted_probabilit

results_R3_2024_upd= results_R3_2024 %>% group_by(off_team_name,def_team_name) %>% summarize(prob_win=r
```

```
## `summarise()` has grouped output by 'off_team_name'. You can override using the
## `.groups` argument.
```

```
temp= data.frame(off_team_name=results_R3_2024_upd$def_team_name, def_team_name=results_R3_2024_upd$off_
results_R3_2024_upd <- rbind(results_R3_2024_upd, temp) %>% select(-def_team_name)
```

```
ggplot(results_R3_2024_upd, aes(x = reorder(off_team_name,-prob_win), y = prob_win)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  geom_text(aes(label = prob_win), position = position_stack(vjust = 0.5), color = "black")+
  labs(title = "Conference Finals Winning Percentage", x = "Team", y = "Percentage") +
  ylim(0, 100)  +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



#Finals

```
round4_2024 <- data.frame(
  off_team_name = c("LA Clippers","LA Clippers"),
  def_team_name = c("Boston Celtics","Boston Celtics"),
```

```
  off_home=rep(c(0, 1))
)
```

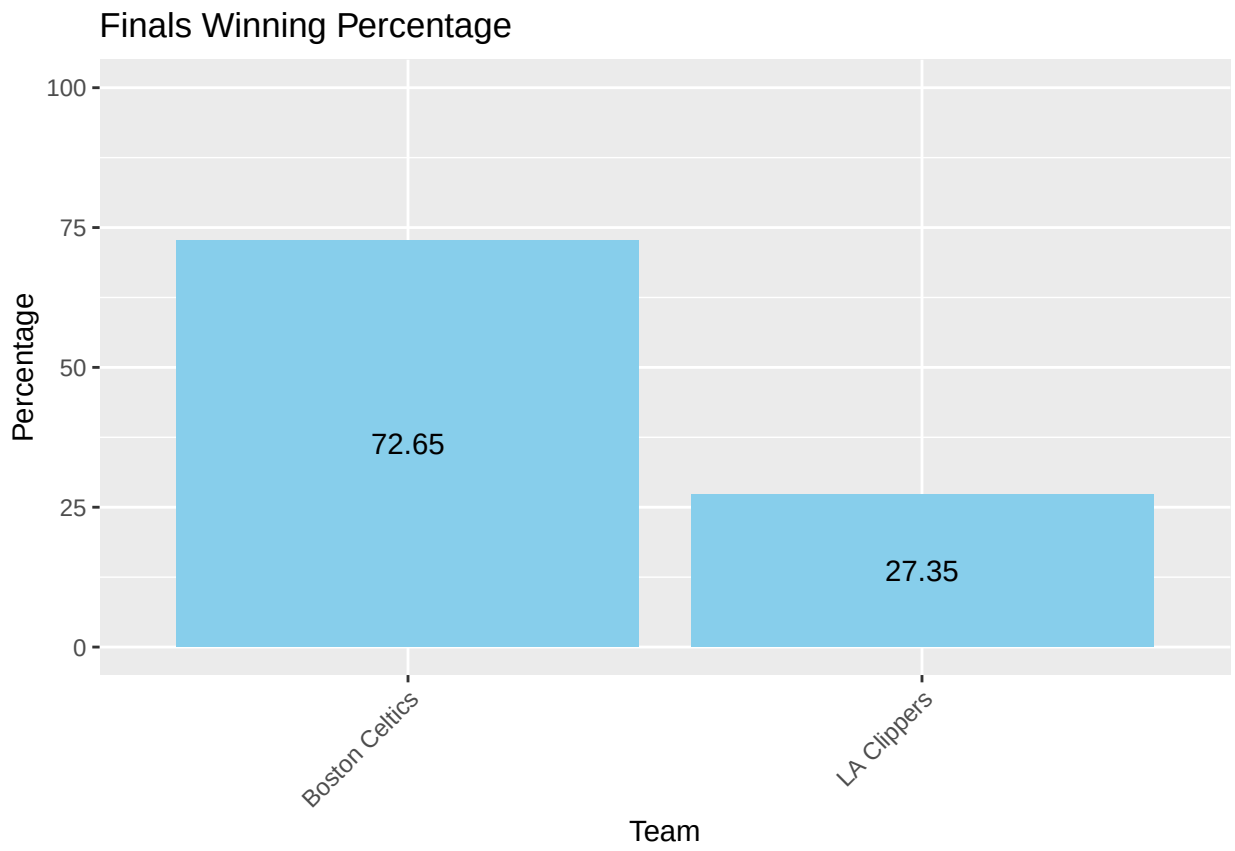
```
round4_m_2024 <- round4_2024 %>% inner_join(test_data_2024,by=c("off_team_name","off_home")) %>%
  inner_join(test_data_2024,by=c("def_team_name"="off_team_name"),suffix=c("","_opponent")) %>% filter(
  select(-off_home_opponent,-season,-season_opponent)

train4_data_2024=round4_m_2024 %>% select(-off_team_name,-def_team_name)
```

```
predicted_probabilities_R4_2024 <- predict(logitModel, newdata = train4_data_2024 , type = "response")
predicted_classes_R4_2024 <- ifelse(predicted_probabilities_R4_2024 > 0.6, 1, 0)

results_R4_2024 = data.frame(round4_m_2024, prob=as.numeric(as.character(as.factor(predicted_probabilit

results_R4_2024_upd= results_R4_2024 %>% group_by(off_team_name,def_team_name) %>% summarize(prob_win=r
```

```
## `summarise()` has grouped output by 'off_team_name'. You can override using the
## `.groups` argument.
```

```
temp= data.frame(off_team_name=results_R4_2024_upd$def_team_name, def_team_name=results_R4_2024_upd$off
results_R4_2024_upd <- rbind(results_R4_2024_upd, temp) %>% select(-def_team_name)
```

```
ggplot(results_R4_2024_upd, aes(x = reorder(off_team_name,-prob_win), y = prob_win)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  geom_text(aes(label = prob_win), position = position_stack(vjust = 0.5), color = "black")+
  labs(title = "Finals Winning Percentage", x = "Team", y = "Percentage") +
  ylim(0, 100) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Part 3 – Finding Insights from Your Model

Find two teams that had a competitive window of 2 or more consecutive seasons making the playoffs and that under performed your model's expectations for them, losing series they were expected to win. Why do you think that happened? Classify one of them as bad luck and one of them as relating to a cause not currently accounted for in your model.

```
team_finals <- team_data %>% filter(season >= 2014 & gametype == 4) %>%
  select(season, off_team_name, def_team_name) %>%
  group_by(season, off_team_name) %>%
  summarise(count = n_distinct(def_team_name), .groups = "drop") %>%
  filter(count == 1)
```

```
match_data_upd3 = team_data %>% filter((off_team_name=="Indiana Pacers" | off_team_name=="Oklahoma City
                                        gametype==4 & season>=2016 & season<=2019) %>%
  select(season,off_team_name,def_team_name,off_home,off_win)

teams= unique(match_data_upd3$`def_team_name`)
teams = c(teams,'Oklahoma City Thunder', 'Indiana Pacers')
test_data4= team_data %>% filter(season>=2016 & season<=2019 & gametype == 2 & (off_team_name %in% teams
  select(-gametype,-nbagameid,-offensivenbateamid,-defensivenbateamid,-def_home) %>%
  mutate(fg2_perc=fg2made/fg2attempted, fg3_perc=fg3made/fg3attempted, ft_perc=ftmade/ftattempted,
                                    ppa=shotattemptpoints/shotattempts, stl_perc=stealsagainst/possessi
                                    oreb=reboffensive/reboundchance) %>% group_by(season,off_team_name
  summarize(across(where(is.numeric), mean)) %>% select(season,off_team_name,off_home,fg2_perc,fg3_perc
```

```
## `summarise()` has grouped output by 'season', 'off_team_name'. You can override
## using the `.groups` argument.
```

```
train4_data = match_data_upd3 %>% inner_join(test_data4,by=c("season","off_team_name","off_home")) %>%
  inner_join(test_data4,by=c("def_team_name"="off_team_name","season"),suffix=c("","_opponent")) %>% fi
  select(-off_home_opponent,-off_win,off_win)
y3= train4_data$off_win
traindata3=train4_data %>% select(-season,-off_team_name,-def_team_name,-off_win)
```

```
predicted_probabilities3 <- predict(logitModel, newdata = traindata3 , type = "response")
predicted_classes3 <- ifelse(predicted_probabilities3 > 0.6, 1, 0)

# Confusion matrix
confMatrix3 <- confusionMatrix(as.factor(predicted_classes3), as.factor(y3))
print(confMatrix3)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 27  5
##          1  5  5
##
##                Accuracy : 0.7619
##                  95% CI : (0.6055, 0.8795)
##     No Information Rate : 0.7619
##     P-Value [Acc > NIR] : 0.5838
##
##                   Kappa : 0.3438
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.8438
##             Specificity : 0.5000
##          Pos Pred Value : 0.8438
##          Neg Pred Value : 0.5000
##              Prevalence : 0.7619
##          Detection Rate : 0.6429
##    Detection Prevalence : 0.7619
##       Balanced Accuracy : 0.6719
##
##        'Positive' Class : 0
##
```

```
results = data.frame(train4_data, predicted= as.factor(predicted_classes3),
                          prob=as.factor(predicted_probabilities3*100)) %>% select(season,off_team_name

final_teams= results %>% filter(predicted==1)
final_teams
```

```
##    season         off_team_name       def_team_name off_home off_win predicted
## 2    2019 Oklahoma City Thunder     Houston Rockets        1       1         1
```

```
## 4    2019 Oklahoma City Thunder     Houston Rockets        1       1         1
## 5    2019 Oklahoma City Thunder     Houston Rockets        1       1         1
## 10   2019        Indiana Pacers          Miami Heat        1       0         1
## 11   2019        Indiana Pacers          Miami Heat        1       0         1
## 17   2016        Indiana Pacers Cleveland Cavaliers        1       0         1
## 18   2016        Indiana Pacers Cleveland Cavaliers        1       0         1
## 29   2017 Oklahoma City Thunder           Utah Jazz        1       1         1
## 32   2017 Oklahoma City Thunder           Utah Jazz        1       1         1
## 33   2017 Oklahoma City Thunder           Utah Jazz        1       0         1
##                prob
## 2  66.9257664178397
## 4  66.9257664178397
## 5  66.9257664178397
## 10 65.5164602071347
## 11 65.5164602071347
## 17 67.3038622658172
## 18 67.3038622658172
## 29 79.8639036671788
## 32 79.8639036671788
## 33 79.8639036671788
```

ANSWER :

Oklahoma City Thunders had a bad luck with Utah Jazz in 2017. It lost the match which was predicted win by my model. My model has predicted correct prediction for the rest of the matches in for OKC in 2017. So I would give OKC having a bad day in that particular match(row Number 33).

Indiana Pacers, on the other hand, had lost all the matches that my predicted expected it to win. This difference can be accounted for the lesser number of parameters used in the models. IF I had more data and time, I would have also considered Head-to-Head matches and win-loss ratio. I would also try to include parameters for players stats for the both teams.