Data Science Project

Rohit Rajkumar Dhonukshe

06/29/24

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

```
# install.packages("tidyverse")
# install.packages("readr")
# install.packages("zoo")
# install.packages("plotly")
# install.packages("caret")
# install.packages("e1071")
# install.packages("corrplot")
# install.packages("car")
# install.packages("glmnet")
```

```
library(tidyverse)
library(readr)
library(zoo)
library(plotly)
library(plotly)
library(caret)
library(e1071)
library(corrplot)
library(car)
library(glmnet)
```

```
options(readr.show_col_types = FALSE)

# Read the player game data
player_data <- read_csv("~/player_game_data.csv")

# Read the team game data
team_data <- read_csv("~/team_game_data.csv")
```

Part 1 – Data Cleaning

In this section, you're going to work to answer questions using data from both team and player stats. All provided stats are on the game level.

We check for NA values in the data sets and drops the rows containing NA values to clean the data.

```
# Count NA values in team_data
sum(is.na(team_data))
```

```
## [1] 0
```

```
# Count NA values in player_data
sum(is.na(player_data))
```

```
## [1] 4
```

```
# Find rows with any NA values
na_rows <- which(rowSums(is.na(player_data)) > 0)
player_data_with_na <- player_data[na_rows, ]

# Print the rows with NA values
print(player_data_with_na)
```

```
## # A tibble: 4 × 59
##   nbagameid gamedate   season gametype nbapersonid player_name  nbateamid team
##       <dbl> <chr>       <dbl>    <dbl>       <dbl> <chr>            <dbl> <chr>
## 1  40900131 17-04-2010   2009        4     3216458 <NA>        1610612738 BOS
## 2  21400621 19-01-2015   2014        2      203296 <NA>        1610612758 SAC
## 3  21500512 04-01-2016   2015        2        6274 <NA>        1610612755 PHI
## 4  21701147 01-04-2018   2017        2  1962937848 <NA>        1610612737 ATL
## # ℹ 51 more variables: team_name <chr>, opposingnbateamid <dbl>,
## #   opp_team <chr>, opp_team_name <chr>, starter <dbl>, missed <dbl>,
## #   seconds <dbl>, points <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, rebdefensive <dbl>,
## #   offensivereboundchances <dbl>, defensivereboundchances <dbl>, …
```

The variable ‘player_name’ contains NA values, so we remove the corresponding rows from the dataset.

```
# Dropping rows with NA values from player_data
player_data <- na.omit(player_data)
```

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.

Solution

First, let’s define a function to calculate the effective field goal percentage (eFG%).

$$\text{eFG\%} = \frac{\text{FG2M} + 0.5 \times \text{FG3M}}{\text{FGA}}$$

```
# Function to calculate eFG%
calculate_eFG <- function(fg2m, fg3m, fga) {
  eFG <- (fg2m + 0.5 * fg3m) / fga
  return(eFG)
}
```

Calculate Offensive eFG% for the Warriors in the 2015-16 Regular Season

```
# Filter the data for Warriors' offensive stats in the 2015 regular season
warriors_off_2015 <- team_data %>%
  filter(off_team == 'GSW' & season == 2015 & gametype == 2)

# Calculate total field goals made and attempted
total_fg2made_off <- sum(warriors_off_2015$fg2made)
total_fg3made_off <- sum(warriors_off_2015$fg3made)
total_fga_off <- sum(warriors_off_2015$fgattempted)

# Calculate offensive eFG%
offensive_eFG <- calculate_eFG(total_fg2made_off, total_fg3made_off, total_fga_off)

# Print the offensive eFG% (converted to percentage)
offensive_eFG_percent <- offensive_eFG * 100
print(paste("Offensive eFG%:", round(offensive_eFG_percent, 1)))
```

```
## [1] "Offensive eFG%: 41.2"
```

Calculate Defensive eFG% for the Warriors in the 2015-16 Regular Season

```
# Filter the data for Warriors' defensive stats in the 2015 regular season
warriors_def_2015 <- team_data %>%
  filter(def_team_name == 'Golden State Warriors' & season == 2015 & gametype == 2)

# Calculate total field goals made and attempted by opponents
total_fg2made_def <- sum(warriors_def_2015$fg2made)
total_fg3made_def <- sum(warriors_def_2015$fg3made)
total_fga_def <- sum(warriors_def_2015$fgattempted)

# Calculate defensive eFG%
defensive_eFG <- calculate_eFG(total_fg2made_def, total_fg3made_def, total_fga_def)

# Print the defensive eFG% (converted to percentage)
defensive_eFG_percent <- defensive_eFG * 100
print(paste("Defensive eFG%:", round(defensive_eFG_percent, 1)))
```

```
## [1] "Defensive eFG%: 39.1"
```

ANSWER 1:

Offensive: 41.2% eFG
Defensive: 39.1% 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.

Solution

First, we filter the data for the 2014-2023 regular seasons and then calculate the game effective field goal percentage (eFG%) for each team.

```
# Filter data for the 2014-2023 regular seasons and calculate eFG% for each game
efg_data <- team_data %>%
  filter(season >= 2014 & season <= 2023 & gametype == 2)%>%
  mutate(game_efg = 100*(fg2made + 0.5 * fg3made) / fgattempted)
```

Exclude games where both teams have the same eFG%.

```
# Remove games where both teams have the same eFG%
efg_data <- efg_data %>%
  group_by(nbagameid) %>%
  filter(n_distinct(game_efg) > 1) %>%
  ungroup()
```

Determine the winning team based on the higher eFG%.

```
# Determine if the team with the higher eFG% won the game
result <- efg_data %>%
  group_by(nbagameid) %>%
  arrange(desc(game_efg)) %>%
  summarise(high_efg_win = first(off_win))
```

Finally, we calculate the percentage of games where the team with the higher eFG% won.

```
# Calculate the percentage of games won by the team with the higher eFG%
percent_efg_win <- sum(result$high_efg_win) / nrow(result) * 100

# Print the result
print(paste("Percentage of games won by the team with the higher eFG%:", round(percent_efg_wi
n, 1), "%"))
```

```
## [1] "Percentage of games won by the team with the higher eFG%: 73 %"
```

ANSWER 2:

73.0%

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.

Solution

First, we filter the data for the 2014-2023 regular seasons.

```
# Filter data for the 2014-2023 regular seasons
off_reb_data <- team_data %>%
  filter(season >= 2014 & season <= 2023 & gametype == 2)
```

Exclude games where both teams have the same number of offensive rebounds.

```
# Remove games where both teams have the same number of offensive rebounds
off_reb_data <- off_reb_data %>%
  group_by(nbagameid) %>%
  filter(n_distinct(reboffensive) > 1) %>%
  ungroup()
```

Determine the winning team based on the higher number of offensive rebounds.

```
# Determine if the team with the higher number of offensive rebounds won the game
result <- off_reb_data %>%
  group_by(nbagameid) %>%
  arrange(desc(reboffensive)) %>%
  summarise(high_offreb_win = first(off_win))
```

Finally, we calculate the percentage of games where the team with more offensive rebounds won.

```
# Calculate the percentage of games won by the team with more offensive rebounds
percent_offreb_win <- sum(result$high_offreb_win) / nrow(result) * 100

# Print the result
print(paste("Percentage of games won by the team with more offensive rebounds:", round(percen
t_offreb_win, 1), "%"))
```

```
## [1] "Percentage of games won by the team with more offensive rebounds: 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? Try to be clear and concise with your answer.

ANSWER 4:

The reason why the percentage of games won by the team with more offensive rebounds (question 3) is lower than the percentage of games won by the team with the higher eFG% (question 2) likely stems from the fundamental differences in the impact of these statistics on the outcome of a game.

Effective Field Goal Percentage (eFG%)
Direct Scoring Impact: eFG% directly measures shooting efficiency, taking into account the added value of three-pointers. Teams with higher eFG% are more likely to score efficiently, which directly contributes to winning games.
Correlation with Winning: High eFG% is strongly correlated with winning because it reflects the team's ability to score points effectively. Teams that shoot better generally win more games because scoring is the primary objective in basketball.

Offensive Rebounds
Indirect Impact: While offensive rebounds provide additional scoring opportunities by allowing a team to maintain possession, they do not directly translate to points. The team still needs to convert these extra possessions into successful shots.
Defensive Trade-offs: Teams that focus heavily on offensive rebounding might compromise their transition defense, allowing the opposing team to score easier baskets on fast breaks. This can offset the advantage gained from additional offensive possessions.
Overall Game Context: Offensive rebounding is just one aspect of the game. A team can win despite having fewer offensive rebounds if it excels in other areas such as defense, free-throw shooting, or minimizing turnovers.

In summary, eFG% has a more direct and significant impact on a team's ability to win games because it reflects shooting efficiency and scoring, which are crucial to winning. Offensive rebounds, while valuable, do not guarantee points and can be offset by weaknesses in other areas of the game. This likely explains why the percentage of games won by the team with the higher eFG% is higher than the percentage of games won by the team with more offensive rebounds.

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.

Solution

First, we filter the data for the 2014-2023 regular seasons.

```
# Filter data for the 2014-2023 regular seasons
av_data <- player_data %>%
  filter(season >= 2014 & season <= 2023 & gametype == 2)
```

Identify players who played at least 25% of their possible games in a season.

```
# Summarize the number of games played and total games in a season for each player
players_with_min_games <- av_data %>%
  group_by(player_name, season) %>%
  summarise(
    games_played = sum(ifelse(missed == 0, 1, 0)),
    season_total_games = n()
  ) %>%
  filter(games_played / season_total_games >= 0.25) %>%
  ungroup()
```

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

Filter the players who scored at least 25 points per game among those identified in the previous step.

```
# Calculate average points per game for players who meet the minimum games requirement
high_scorers <- av_data %>%
  inner_join(players_with_min_games, by = c('player_name', 'season')) %>%
  filter(missed == 0) %>%
  group_by(player_name, season) %>%
  summarise(avg_points = mean(points)) %>%
  filter(avg_points >= 25.0) %>%
  select(player_name, season)
```

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

Finally, we calculate the percentage of games these high-scoring players were available for on average.

```
# Calculate the percentage of games players were available
availability_percentage <- av_data %>%
  inner_join(high_scorers, by = c('player_name', 'season')) %>%
  group_by(player_name, season) %>%
  summarise(
    available_games = sum(missed == 0),
    total_games = n()
  ) %>%
  summarise(
    total_available_games = sum(available_games),
    total_season_games = sum(total_games)
  ) %>%
  summarise(percentage_available = sum(total_available_games) / sum(total_season_games) * 10
0)
```

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

```
# Print the result
print(paste("Average percentage of games available for high-scoring players:", round(availabi
lity_percentage$percentage_available, 1), "%"))
```

```
## [1] "Average percentage of games available for high-scoring players: 83.4 %"
```

ANSWER 5:

83.4% 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).

SOlution

Filter the data for the 2014-2022 seasons and identify the round of the playoff series.

```
# Filter data for playoff games from 2014 to 2022 and identify the round of the series
playoff <- team_data %>%
  filter(gametype == 4 & season >= 2014 & season <= 2022) %>%
  mutate(round = substr(as.character(nbagameid), 6, 6))
```

Define a function to calculate the win percentage by round for teams with home court advantage and calculate the win %

```
# Function to calculate win percentage by round for home court advantage teams
calculate_home_court_win_percentage <- function(playoff_data, round_number) {
  playoff_data %>%
    filter(round == round_number & off_home == 1) %>%
    summarise(
      wins = sum(off_win),
      games = n()
    ) %>%
    mutate(win_percentage = round(wins / games * 100, 1))
}

# Calculate win percentages by round
win_percentage_round1 <- calculate_home_court_win_percentage(playoff, 1)
win_percentage_round2 <- calculate_home_court_win_percentage(playoff, 2)
win_percentage_conf_final <- calculate_home_court_win_percentage(playoff, 3)
win_percentage_final <- calculate_home_court_win_percentage(playoff, 4)

# Print results
print(win_percentage_round1)
```

```
## # A tibble: 1 × 3
##    wins games win_percentage
##   <dbl> <int>          <dbl>
## 1   232   386           60.1
```

```
print(win_percentage_round2)
```

```
## # A tibble: 1 × 3
##    wins games win_percentage
##   <dbl> <int>          <dbl>
## 1   127   211           60.2
```

```
print(win_percentage_conf_final)
```

```
## # A tibble: 1 × 3
##    wins games win_percentage
##   <dbl> <int>          <dbl>
## 1    59   101           58.4
```

```
print(win_percentage_final)
```

```
## # A tibble: 1 × 3
##    wins games win_percentage
##   <dbl> <int>          <dbl>
## 1    27    51           52.9
```

ANSWER 6:

Round 1: 60.1%
Round 2: 60.2%
Conference Finals: 58.4%
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.

Solution

Calculate Offensive and Defensive Ratings

```
offensive_ratings <- team_data %>%
  filter(season >= 2014 & season <= 2021) %>%
  mutate(off_rating = 100 * (points / possessions)) %>%
  group_by(season, off_team) %>%
  summarise(offensive_rating = mean(off_rating, na.rm = TRUE))
```

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

```
defensive_ratings <- team_data %>%
  filter(season >= 2014 & season <= 2021) %>%
  mutate(def_rating = 100 * (points / possessions)) %>%
  group_by(season, def_team) %>%
  summarise(defensive_rating = mean(def_rating, na.rm = TRUE))
```

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

Calculate Net Rating and Filter Teams

```
net_ratings <- offensive_ratings %>%
  inner_join(defensive_ratings, by = c("season", "off_team" = "def_team")) %>%
  mutate(net_rating = offensive_rating - defensive_rating, next_season = season + 1) %>%
  filter(net_rating >= 5.0)
```

Identify Teams Making the Second Round of Playoffs

```
playoff_games <- team_data %>%
  filter(gametype == 4 & season >= 2015 & season <= 2022) %>%
  mutate(round = substr(as.character(nbagameid), 6, 6))

second_round_teams <- playoff_games %>%
  filter(round == 2) %>%
  distinct(season, off_team, gametype)

teams_with_net_rating <- net_ratings %>%
  left_join(second_round_teams, by = c("next_season" = "season", "off_team")) %>%
  mutate(made_second_round = ifelse(!is.na(gametype), 1, 0))

percentage_made_second_round <- round(sum(teams_with_net_rating$made_second_round) / nrow(tea
ms_with_net_rating) * 100,1)
print(paste("Percentage of teams with +5.0 net rating making the second round:", percentage_m
ade_second_round))
```

```
## [1] "Percentage of teams with +5.0 net rating making the second round: 63.6"
```

Filter Players and Calculate Minutes Played

```
regular_season_players <- player_data %>%
  filter(gametype == 2) %>%
  inner_join(teams_with_net_rating %>% filter(made_second_round == 1), by = c("season", "tea
m" = "off_team"))

players_minutes <- regular_season_players %>%
  group_by(season, team, player_name) %>%
  summarise(total_minutes = sum(seconds) / 60, .groups = 'drop')

top_5_players <- players_minutes %>%
  group_by(season, team) %>%
  slice_max(order_by = total_minutes, n = 5) %>%
  ungroup()

top_5_players <- top_5_players %>%
  mutate(next_season = season + 1)
```

Identify Top Players Playing in the Next Season’s Second Round

```
playoff_participation <- player_data %>%
  filter(gametype == 4) %>%
  inner_join(teams_with_net_rating %>% filter(made_second_round == 1), by = c("season" = "nex
t_season", "team" = "off_team")) %>%
  mutate(round = substr(as.character(nbagameid), 6, 6)) %>%
  filter(round == 2) %>%
  distinct(season, team, player_name, gametype.x)

top_players_playoff <- top_5_players %>%
  left_join(playoff_participation, by = c("next_season" = "season", "team", "player_name")) %
>%
  mutate(played_in_second_round = ifelse(!is.na(gametype.x), 1, 0))

percentage_played_in_second_round <- round(sum(top_players_playoff$played_in_second_round) /
nrow(top_players_playoff) * 100,1)
print(paste("Percentage of top players playing in next season's second round:", percentage_pl
ayed_in_second_round))
```

```
## [1] "Percentage of top players playing in next season's second round: 79"
```

ANSWER 7:

Percent of +5.0 net rating teams making the 2nd round next year: 63.6%
Percent of top 5 minutes played players who played in those 2nd round series: 79.0%

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.

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

1. Your final output must include 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.” You must also predict the number of games in the series. This can be probabilistic or a point estimate.
2. You may use any data provided in this project, but please do not bring in any external sources of data.
3. You can only use data available prior to the start of the series. For example, you can’t use a team’s stats from the 2016-17 season to predict a playoffs series from the 2015-16 season.
4. The best models are explainable and lead to actionable insights around team and roster construction. We’re more interested in your thought process and critical thinking than we are in specific modeling techniques. Using smart features is more important than using fancy mathematical machinery.
5. 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.

Solution

Data Preparation

We completed the data cleaning process previously

Feature Engineering and Model Preparation

The below code is written to create a dataset for predicting the outcomes of NBA playoff series. The steps include calculating advanced statistics, aggregating player data, calculating rolling averages, and preparing the final dataset for modeling.

Feature Engineering for Team Data

First, we calculate advanced statistics for each team game. These metrics include Player Points per Attempt (PPA), Turnover Percentage (TOV), Offensive Rating (ORTG), Field Goal Percentage (FGP), Three-Point Percentage (TPP), Free Throw Percentage (FTP) and Effective Field Goal (EFG)

```
# Calculate advanced statistics for each team game
team_data_enhanced <- team_data %>%
  mutate(
    ppa = shotattemptpoints / shotattempts,
    tov = turnovers / (possessions / 100),
    rtg = points / (possessions / 100),
    fgp = fgmade / fgattempted,
    tpp = fg3made / fg3attempted,
    ftp = ftmade / ftattempted,
    game_efg = (fg2made + 0.5 * fg3made) / fgattempted
  )
```

Feature Engineering for Player Data

Next, we calculate individual player metrics for each game. These metrics include PPA, Usage (USG), Assist Ratio (AST), Offensive and Defensive Rebound Ratios (OREB and DREB), Turnover Ratio (TOV), Steal Ratio (STL), Block Ratio (BLK), Effective Field Goal Percentage (EFG), and True Shooting Percentage (TS).

```
# Calculate individual player metrics
player_data_enhanced <- player_data %>%
  filter(seconds != 0) %>%
  mutate(
    PPA = shotattemptpoints / shotattempts,
    USG = (shotattempts + turnovers) / (teamshotattempts + teamturnovers),
    AST = assists / (teamfgmade - (fg3made + fg2made)),
    OREB = reboffensive / offensivereboundchances,
    DREB = rebdefensive / defensivereboundchances,
    TOV = turnovers / (shotattempts + turnovers),
    STL = steals / (defensivepossessions + 1),  # normalize to avoid INF
    BLK = blocks / (opponentteamfg2attempted + 1),  # normalize to avoid INF
    EFG = (fg2made + 0.5 * fg3made) / fgattempted,
    TS = points / (2 * (fgattempted + 0.44 * ftattempted))
  )
```

Aggregating Player Data

We then aggregate the player metrics to calculate average metrics for each team in each game.

```
# Summarize player metrics for each game
team_avg_player_metrics <- player_data_enhanced %>%
  group_by(nbagameid, gamedate, season, gametype, team, opp_team) %>%
  summarise(
    avg_PPA = mean(PPA, na.rm = TRUE),
    avg_USG = mean(USG, na.rm = TRUE),
    avg_AST = mean(AST, na.rm = TRUE),
    avg_OREB = mean(OREB, na.rm = TRUE),
    avg_DREB = mean(DREB, na.rm = TRUE),
    avg_TOV = mean(TOV, na.rm = TRUE),
    avg_STL = mean(STL, na.rm = TRUE),
    avg_BLK = mean(BLK, na.rm = TRUE),
    avg_EFG = mean(EFG, na.rm = TRUE),
    avg_TS = mean(TS, na.rm = TRUE)
  )
```

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

Combining Team and Player Data

We join the aggregated player metrics with the enhanced team data.

```
# Combine team and player data
combined_data <- team_data_enhanced %>%
  inner_join(team_avg_player_metrics, by = c('season', 'gametype', 'nbagameid', 'off_team' =
'team'))%>%
  select(season, gametype, nbagameid, gamedate.x, off_team, off_home, off_win, def_team, def_
home, def_win, points, game_efg, ppa, tov, rtg, fgp, tpp, ftp, avg_PPA, avg_USG, avg_AST, avg
_OREB, avg_DREB, avg_TOV, avg_STL, avg_BLK, avg_EFG, avg_TS)
```

```
selected_columns <- c("off_win","points","game_efg","ppa","tov","rtg","fgp","tpp","ftp","avg_
PPA","avg_USG","avg_AST", "avg_OREB","avg_DREB","avg_TOV","avg_STL","avg_BLK","avg_EFG","avg_
TS")

calc_rolling_avg <- function(data, selected_columns, window_size = 5) {
  data %>%
    arrange(gamedate.x) %>%
    mutate(across(all_of(selected_columns), ~ rollmeanr(., window_size, fill = NA, align = "r
ight"), .names = "rollavg_{col}")) %>%
    filter(row_number() > window_size) %>%  # Exclude the first 'window_size' rows
    mutate(across(starts_with("rollavg_"),
                  ~ lag(., 1),
                  .names = "prev_{col}"))  # Exclude the current game from the rolling averag
e
}

# Calculate rolling averages for home and away games separately
home_games <- combined_data  %>%
  filter(off_home == 1) %>%
  group_by(season, off_team) %>%
  group_modify(~ calc_rolling_avg(.x, selected_columns))

away_games <- combined_data  %>%
  filter(off_home == 0) %>%
  group_by(season, off_team) %>%
  group_modify(~ calc_rolling_avg(.x, selected_columns))

# Combine home and away rolling averages
rolling_avg_data  <- bind_rows(home_games, away_games) %>%
  arrange(nbagameid)
```

Creating Final Dataset for Modeling

We join the rolling averages for both teams in each game to create a comprehensive dataset for modeling. This includes metrics for both the offensive and defensive teams.

```
# Self-join to match each game with its opponent's data
modeling_data <- rolling_avg_data %>%
  inner_join(rolling_avg_data, by = c('season', 'gametype', 'nbagameid'), suffix = c("", "_de
f")) %>%
  filter(off_team != off_team_def)  # ensure teams are different
```

```
## Warning in inner_join(., rolling_avg_data, by = c("season", "gametype", : Detected an unex
pected many-to-many relationship between `x` and `y`.
## ℹ Row 8 of `x` matches multiple rows in `y`.
## ℹ Row 8 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
# Select relevant columns and calculate net rating
final_dataset <- modeling_data %>%
  select(
    season, off_team, gametype, nbagameid, gamedate.x, off_home, off_win, def_team, def_home,
def_win,
    starts_with("prev_rollavg")
  ) %>%
  drop_na() %>%
  mutate(
    prev_rollavg_netrtg = prev_rollavg_rtg - prev_rollavg_rtg_def,
    target = as.factor(off_win)
  )
```

Function to Prepare Data for Training and Testing

```
prepare_data <- function(data, final_season, gametype_filter, selected_columns) {
  # Filter data for the specified seasons and game type
  # Here we consider past 5 seasons data
  start_season = final_season - 5
  filtered_data <- data %>%
    filter(season <= final_season & season >=start_season & gametype == gametype_filter)

  # Select the relevant columns
  selected_data <- filtered_data[selected_columns]
  return(selected_data)
}
```

Function to Split Data into Training and Testing Sets

```
split_data <- function(data, target_col, split_ratio) {
  set.seed(123)  # Ensure reproducibility
  trainIndex <- createDataPartition(data[[target_col]], p = split_ratio, list = FALSE)
  trainData <- data[trainIndex, ]
  testData <- data[-trainIndex, ]
  return(list(trainData = trainData, testData = testData))
}
```

Function to Train Logistic Regression Model with Cross Validation

```
train_logistic_model <- function(train_data, target_col) {
  control <- trainControl(method = "cv", number = 10)  # 10-fold cross-validation
  set.seed(123)
  model <- train(as.formula(paste(target_col, "~ .")), data = train_data, method = "glm", fam
ily = binomial, trControl = control)
  return(model)
}
```

Function to Predict and Evaluate Model

```
predict_and_evaluate <- function(model, test_data, target_col) {
  predictions <- predict(model, newdata = test_data)
  probabilities <- predict(model, newdata = test_data, type = "prob")
  cm <- confusionMatrix(predictions, test_data[[target_col]])

  # Print results
  print("Confusion Matrix:")
  print(cm)

  return(list(predictions = predictions, probabilities = probabilities, confusion_matrix = c
m))
}
```

Model Execution Steps

```
# Define the columns to be used for prediction
selected_columns <- c("prev_rollavg_off_win", "prev_rollavg_off_win_def", "prev_rollavg_netrt
g",
                      "prev_rollavg_tpp", "prev_rollavg_ftp",
                      "prev_rollavg_avg_USG", "prev_rollavg_avg_AST", "prev_rollavg_avg_ORE
B",
                      "prev_rollavg_avg_DREB", "prev_rollavg_avg_TOV", "prev_rollavg_avg_ST
L",
                      "prev_rollavg_avg_BLK", "prev_rollavg_avg_EFG", "prev_rollavg_avg_TS",
                      "prev_rollavg_tpp_def", "prev_rollavg_ftp_def",
                      "prev_rollavg_avg_USG_def", "prev_rollavg_avg_AST_def", "prev_rollavg_a
vg_OREB_def",
                      "prev_rollavg_avg_DREB_def", "prev_rollavg_avg_TOV_def", "prev_rollavg_
avg_STL_def",
                      "prev_rollavg_avg_BLK_def", "prev_rollavg_avg_EFG_def", "prev_rollavg_a
vg_TS_def",
                      "target")
```

Here, I have chosen the playoffs of 2021 season for prediction

```
# Prepare the data
data <- prepare_data(final_dataset, final_season = 2021, gametype_filter = 2, selected_column
s)

# Split the data into training and testing sets
data_split <- split_data(data, target_col = "target", split_ratio = 0.8)
train_data <- data_split$trainData
test_data <- data_split$testData

# Train the logistic regression model
logistic_model <- train_logistic_model(train_data, target_col = "target")

# Evaluate the model on the testing data
evaluation_results <- predict_and_evaluate(logistic_model, test_data, target_col = "target")
```

```
## [1] "Confusion Matrix:"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 748 430
##          1 423 741
##
##                Accuracy : 0.6358
##                  95% CI : (0.6159, 0.6553)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.2716
##
##  Mcnemar's Test P-Value : 0.8372
##
##             Sensitivity : 0.6388
##             Specificity : 0.6328
##          Pos Pred Value : 0.6350
##          Neg Pred Value : 0.6366
##              Prevalence : 0.5000
##          Detection Rate : 0.3194
##    Detection Prevalence : 0.5030
##       Balanced Accuracy : 0.6358
##
##        'Positive' Class : 0
##
```

We can see that model trained on Regular seasons of (2016-2021) data has an accuracy of 63.5%

Now to predict who will win the playoff series , my strategy is to use both teams performance during the season, utilizing my models predictions for all games both teams played against other team and games played head-to-head against each other during the season .

Function to Predict Playoff Outcome Probability

```
predict_playoff_outcome <- function(playoff_season, team1, team2, model, data, selected_colum
ns) {
  # Filter data for the specified season and team1 games
  team1_data <- data %>%
    filter(season == playoff_season & gametype == 2 & off_team == team1 & def_team != team2)
  # Select the relevant columns for prediction
  team1_test_data <- team1_data[selected_columns]

  # Make probability predictions using the provided model
  probabilities1 <- predict(model, newdata = team1_test_data, type = "prob")
  # Create a data frame with predictions and actual outcomes
  probabilities1 <- data.frame(probabilities1)
  team1_df <- data.frame(Prob = probabilities1[2], Actual_Outcomes = team1_data$off_win)

  # For team2
  team2_data <- data %>%
    filter(season == playoff_season & gametype == 2 & off_team == team2 & def_team != team1)
  team2_test_data <- team2_data[selected_columns]
  probabilities2 <- predict(model, newdata = team2_test_data, type = "prob")
  probabilities2 <- data.frame(probabilities2)
  team2_df <- data.frame(Prob = probabilities2[2], Actual_Outcomes = team2_data$off_win)

  # For head-to-head games
  head_data <- data %>%
    filter(season == playoff_season & gametype == 2 & off_team == team1 & def_team == team2)
  head_test_data <- head_data[selected_columns]
  probabilitiesh <- predict(model, newdata = head_test_data, type = "prob")
  probabilitiesh <- data.frame(probabilitiesh)
  head_df <- data.frame(Prob = probabilitiesh[2], Actual_Outcomes = head_data$off_win)

  # For head-to-head games in previous season
  head_past_data <- data %>%
    filter(season == (playoff_season-1) & gametype == 2 & off_team == team1 & def_team == tea
m2)
  head_past_test_data <- head_past_data[selected_columns]
  probabilitieshp <- predict(model, newdata = head_past_test_data, type = "prob")
  probabilitieshp <- data.frame(probabilitieshp)
  head_past_df <- data.frame(Prob = probabilitieshp[2], Actual_Outcomes = head_past_data$off_
win)



  # Calculate column means
  team1_means <- colMeans(team1_df)
  team2_means <- colMeans(team2_df)
  head_means <- colMeans(head_df)
  head_past_means <- colMeans(head_past_df)

  # Create a summary data frame with appropriately named columns
  summary_df <- data.frame(
    Team1_Prob_Mean = team1_means["X1"],
    Team1_Actual_Outcomes_Mean = team1_means["Actual_Outcomes"],
    Team2_Prob_Mean = team2_means["X1"],
    Team2_Actual_Outcomes_Mean = team2_means["Actual_Outcomes"],
    Head_Prob_Mean = head_means["X1"],
    Head_Actual_Outcomes_Mean = head_means["Actual_Outcomes"],
```

```
    Head_Past_Prob_Mean = head_past_means["X1"],
    Head_Past_Actual_Outcomes_Mean = head_past_means["Actual_Outcomes"]
  )
  team1_weighted <- 0.65 * summary_df$Team1_Prob_Mean + 0.35 * summary_df$Team1_Actual_Outcom
es_Mean
  team2_weighted <- 0.65 * (1 - summary_df$Team2_Prob_Mean) + 0.35 * (1 - summary_df$Team2_Ac
tual_Outcomes_Mean)
  head_weighted <- 0.65 * summary_df$Head_Prob_Mean + 0.35 * summary_df$Head_Actual_Outcomes_
Mean
  head_past_weighted <- 0.65 * summary_df$Head_Past_Prob_Mean + 0.35 * summary_df$Head_Past_A
ctual_Outcomes_Mean

  # Calculate final probability using weighted average of weighted means
  # Provided the weights based on manual experimentation and based on importance of overall p
erformance of both teams during the regular season and head-to-head play outcomes during curr
ent and previous seasons.
  final_prob_team1 <- 0.35 * team1_weighted +
                      0.35 * team2_weighted +
                      0.20 * head_weighted +
                      0.10 * head_past_weighted
  # Return the summary data frame
  return(final_prob_team1)

}
```

Simulating the playoff series

```
simulate_series <- function(final_prob_team1) {
  # Initialize counters for wins and losses
  team1_wins <- 0
  team2_wins <- 0
  max_matches <- 7

  # Simulate each match
  for (match in 1:max_matches) {
    # Sample the outcome of the match based on the probability of Team 1 winning
    match_outcome <- sample(c("Team1", "Team2"), size = 1, prob = c(final_prob_team1, 1 - fin
al_prob_team1))

    # Update win counters based on the match outcome
    if (match_outcome == "Team1") {
      team1_wins <- team1_wins + 1
    } else {
      team2_wins <- team2_wins + 1
    }

    # Check if either team has won 4 matches
    if (team1_wins == 4 || team2_wins == 4) {
      break
    }
  }

  # Return the final series score
  return(data.frame(Team1_Wins = team1_wins, Team2_Wins = team2_wins))
}
```

```
playoff_season <- 2021
team1 <- "BOS"
team2 <- "GSW"

# Prepare the data
data <- prepare_data(final_dataset, final_season = playoff_season, gametype_filter = 2, selec
ted_columns)

# Split the data into training and testing sets
data_split <- split_data(data, target_col = "target", split_ratio = 0.8)
train_data <- data_split$trainData
test_data <- data_split$testData

# Train the logistic regression model
logistic_model <- train_logistic_model(train_data, target_col = "target")

result <- predict_playoff_outcome(playoff_season, team1, team2, logistic_model, final_datase
t, selected_columns)

series_result <- simulate_series(result)
cat('Team1:', team1, 'has a', round(100*result),'% chance of winning the series','\n')
```

```
## Team1: BOS has a 54 % chance of winning the series
```

```
cat('Team2:', team2, 'has a', round(100*(1-result)),'% chance of winning the series','\n')
```

```
## Team2: GSW has a 46 % chance of winning the series
```

```
cat('Games played =', series_result[1,1]+series_result[1,2], '\n')
```

```
## Games played = 5
```

```
cat('Series result :', series_result[1,1], '-',series_result[1,2] )
```

```
## Series result : 4 - 1
```

Summary of the Modeling Approach

Purpose:
The primary goal of the model is to predict the probability of each team winning a playoff series and the number of games the series will last.

Approach:

Data Utilization: The model uses data from the current and past four seasons (a total of five seasons) to train and make predictions. For example, predicting outcomes for the 2022 playoffs uses data from 2017-2022. Key Metrics: The model incorporates a wide range of advanced metrics, including Net rating, Usage rate, Assist rate, Rebounding rates, Turnover rate, Steal rate, Block rate, Effective field goal percentage, True shooting percentage.

Recent Performance: It uses rolling averages of key metrics from the past five home games for the home team and the past five away games for the away team to capture recent performance trends.

Prediction Method: A logistic regression model is used, achieving an accuracy of approximately 64% in predicting game outcomes.

Playoff Simulation: Using game-by-game predictions, the model simulates a seven-game playoff series to estimate each team's chances of winning the series and the likely number of games played.

Strengths and Weaknesses

Strengths:

Comprehensive Feature Representation: The model incorporates a wide range of advanced metrics, providing a detailed and comprehensive representation of team performance.
Recent Performance Trends: By using rolling averages, the model captures recent performance and momentum, which are crucial for predicting playoff outcomes. Adaptability to New Data: The model is designed to adapt to new data each season, ensuring it remains current and accurate.

Weaknesses:

Mid-Season Changes: The model does not fully account for mid-season changes such as trades, coaching changes, or significant shifts in team strategy.
Player Injuries: Major player injuries are not directly accounted for, which can significantly impact predictions.
Days of Rest: The model does not consider the number of rest days between games, which can affect player performance and game outcomes.

Addressing Weaknesses

Improvement Plan:

Incorporate Mid-Season Changes: The model can be adjusted to dynamically update for mid-season trades, coaching changes, and other significant alterations.

Player Injuries: Integrate a factor that accounts for injuries to key players, possibly through real-time injury reports and historical performance impact data.

Rest Days: Include a variable for days of rest to better predict performance fluctuations due to fatigue or recovery.

Model Experimentation: Although logistic regression has performed best (experimented with Decision Tress and Random Forest as well), experimenting with more complex models like neural networks can help improve accuracy and robustness.
By addressing these weaknesses with additional time and data, the model's predictive accuracy and reliability in real-world scenarios can be further enhanced.

Applying my model to the 2024 NBA playoffs (2023 season) and create a high quality visualization showing the 16 teams' (that made the first round) chances of advancing to each round.

Teams and Initial Matchups

```
playoff_teams <- list(
  "Eastern Conference" = c("MIL", "BOS", "PHI", "CLE", "NYK", "IND", "MIA", "ORL"),
  "Western Conference" = c("DEN", "DAL", "NOP", "PHX", "LAC", "OKC", "LAL", "MIN")
)

initial_matchups <- list(
  "Eastern Conference" = list(
    c("BOS", "MIA"),
    c("CLE", "ORL"),
    c("IND", "MIL"),
    c("PHI", "NYK")
  ),
  "Western Conference" = list(
    c("NOP", "OKC"),
    c("LAC", "DAL"),
    c("PHX", "MIN"),
    c("LAL", "DEN")
  )
)
```

```
playoff_season <- 2023

# Prepare the data
data <- prepare_data(final_dataset, final_season = playoff_season, gametype_filter = 2, selec
ted_columns)

# Split the data into training and testing sets
data_split <- split_data(data, target_col = "target", split_ratio = 0.8)
train_data <- data_split$trainData
test_data <- data_split$testData

# Train the logistic regression model
logistic_model <- train_logistic_model(train_data, target_col = "target")



# Main function
calculate_win_percentages <- function(playoff_season, team1, team2, logistic_model, final_dat
aset, selected_columns) {

  # Predict the playoff outcome
  result <- predict_playoff_outcome(playoff_season, team1, team2, logistic_model, final_datas
et, selected_columns)

  # Simulate the series
  series_result <- simulate_series(result)

  # Calculate and print the results
  team1_win_percentage <- round(100 * result)
  team2_win_percentage <- round(100 * (1 - result))
  games_played <- series_result[1, 1] + series_result[1, 2]
  series_result_str <- paste(series_result[1, 1], "-", series_result[1, 2])

  cat('Team1:', team1, 'has a', team1_win_percentage, '% chance of winning the series', '\n')
  cat('Team2:', team2, 'has a', team2_win_percentage, '% chance of winning the series', '\n')
  cat('Games played =', games_played, '\n')
  cat('Expected Series result :', series_result_str, '\n')

  # Return a list of results
  return(list(
    team1 = team1,
    team2 = team2,
    team1_win_percentage = team1_win_percentage,
    team2_win_percentage = team2_win_percentage,
    games_played = games_played,
    series_result = series_result_str
  ))
}
```

ROUND 1 Simulation - Eastern Conference

```
result <- calculate_win_percentages(2023, "BOS", "MIA", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: BOS has a 61 % chance of winning the series
## Team2: MIA has a 39 % chance of winning the series
## Games played = 5
## Expected Series result : 4 - 1
```

```
result <- calculate_win_percentages(2023, "CLE", "ORL", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: CLE has a 53 % chance of winning the series
## Team2: ORL has a 47 % chance of winning the series
## Games played = 7
## Expected Series result : 3 - 4
```

```
result <- calculate_win_percentages(2023, "IND", "MIL", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: IND has a 47 % chance of winning the series
## Team2: MIL has a 53 % chance of winning the series
## Games played = 6
## Expected Series result : 4 - 2
```

```
result <- calculate_win_percentages(2023, "PHI", "NYK", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: PHI has a 49 % chance of winning the series
## Team2: NYK has a 51 % chance of winning the series
## Games played = 7
## Expected Series result : 4 - 3
```

Conference Semifinals

```
result <- calculate_win_percentages(2023, "BOS", "CLE", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: BOS has a 57 % chance of winning the series
## Team2: CLE has a 43 % chance of winning the series
## Games played = 6
## Expected Series result : 4 - 2
```

```
result <- calculate_win_percentages(2023, "IND", "NYK", logistic_model, final_dataset, select
ed_columns)
```

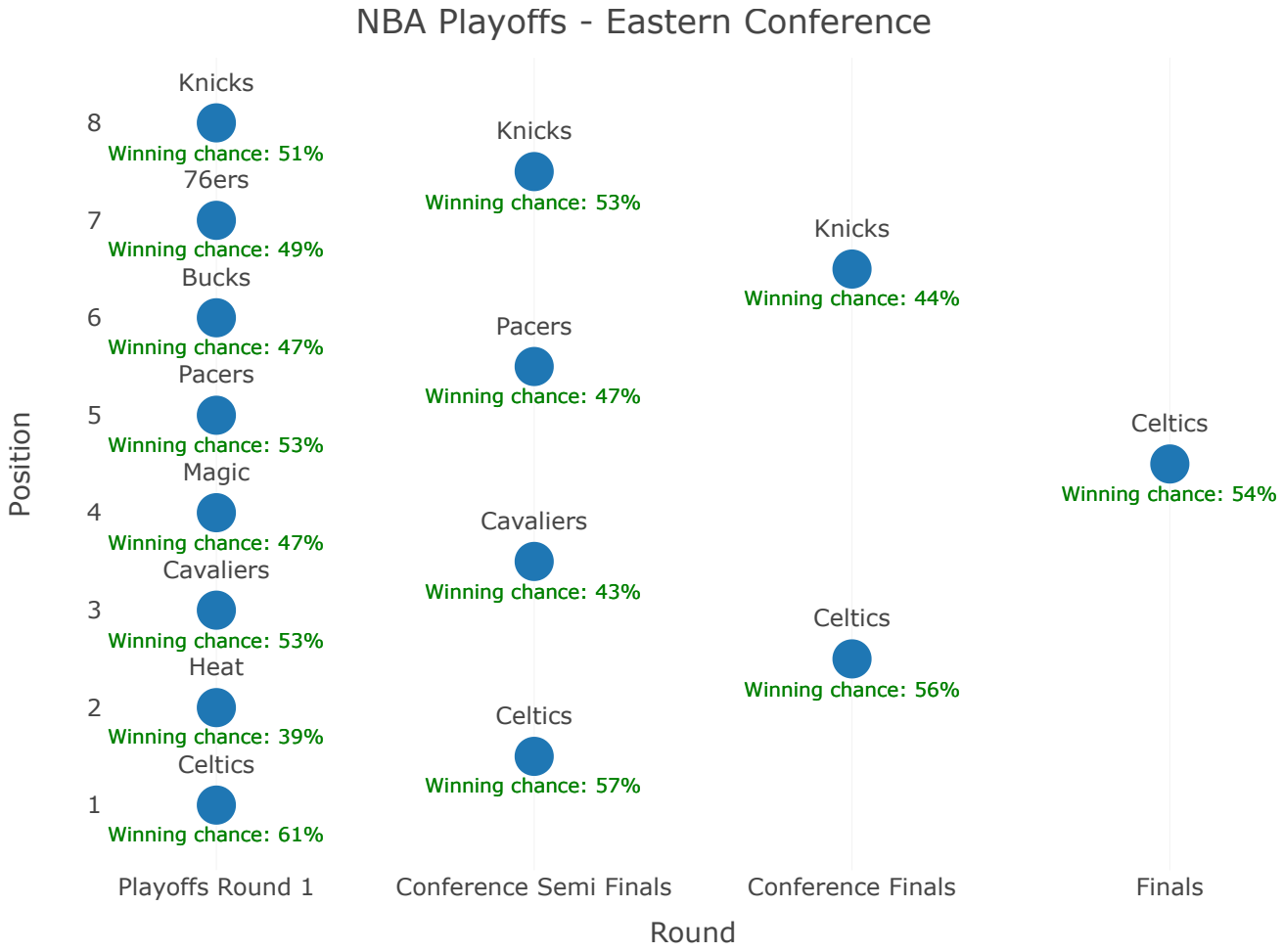
```
## Team1: IND has a 47 % chance of winning the series
## Team2: NYK has a 53 % chance of winning the series
## Games played = 7
## Expected Series result : 3 - 4
```

Conference Finals

```
result <- calculate_win_percentages(2023, "BOS", "NYK", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: BOS has a 56 % chance of winning the series
## Team2: NYK has a 44 % chance of winning the series
## Games played = 6
## Expected Series result : 4 - 2
```

```
# Define the positions for each team in the bracket
teams <- c("Celtics", "Heat", "Cavaliers", "Magic", "Pacers", "Bucks", "76ers", "Knicks",
           "Celtics", "Cavaliers", "Pacers", "Knicks",
           "Celtics", "Knicks",
           "Celtics")
rounds <- c(1, 1, 1, 1, 1, 1, 1, 1,
            2, 2, 2, 2,
            3, 3,
            4)
positions <- c(1, 2, 3, 4, 5, 6, 7, 8,
               1.5, 3.5, 5.5, 7.5,
               2.5, 6.5,
               4.5)

# Define the winning chances for each team
winning_chances <- c("61%", "39%", "53%", "47%", "53%", "47%", "49%", "51%",
                     "57%", "43%", "47%", "53%",
                     "56%", "44%",
                     "54%")

# Create a data frame for plotting
bracket_df <- data.frame(team = teams, round = rounds, position = positions, winning_chance =
winning_chances)

# Plot using plotly
p <- plot_ly()

# Add the nodes with team names
p <- p %>% add_trace(type = 'scatter', mode = 'markers+text',
                     x = bracket_df$round, y = bracket_df$position,
                     text = bracket_df$team, textposition = 'top center',
                     marker = list(size = 20),
                     showlegend = FALSE)

# Add annotations for winning chances
for (i in 1:nrow(bracket_df)) {
  # Add winning chances below team names
  p <- p %>% add_annotations(
    x = bracket_df$round[i],
    y = bracket_df$position[i] - 0.3,
    text = paste("Winning chance:", bracket_df$winning_chance[i]),
    showarrow = FALSE,
    font = list(size = 10, color = 'green')
  )
}

# Customize layout
p <- p %>% layout(title = 'NBA Playoffs - Eastern Conference',
                  xaxis = list(title = 'Round', tickvals = 1:4, ticktext = c("Playoffs Round
1","Conference Semi Finals","Conference Finals","Finals")),
                  yaxis = list(title = 'Position', showgrid = FALSE, zeroline = FALSE),
                  showlegend = FALSE)

# Show plot
p
```



ROUND 1 Simulation - Western Conference

```
result <- calculate_win_percentages(2023, "NOP", "OKC", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: NOP has a 44 % chance of winning the series
## Team2: OKC has a 56 % chance of winning the series
## Games played = 6
## Expected Series result : 4 - 2
```

```
result <- calculate_win_percentages(2023, "LAC", "DAL", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: LAC has a 56 % chance of winning the series
## Team2: DAL has a 44 % chance of winning the series
## Games played = 6
## Expected Series result : 4 - 2
```

```
result <- calculate_win_percentages(2023, "PHX", "MIN", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: PHX has a 52 % chance of winning the series
## Team2: MIN has a 48 % chance of winning the series
## Games played = 7
## Expected Series result : 3 - 4
```

```
result <- calculate_win_percentages(2023, "LAL", "DEN", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: LAL has a 46 % chance of winning the series
## Team2: DEN has a 54 % chance of winning the series
## Games played = 7
## Expected Series result : 4 - 3
```

Conference Semifinals

```
result <- calculate_win_percentages(2023, "LAC", "OKC", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: LAC has a 48 % chance of winning the series
## Team2: OKC has a 52 % chance of winning the series
## Games played = 7
## Expected Series result : 4 - 3
```

```
result <- calculate_win_percentages(2023, "PHX", "DEN", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: PHX has a 49 % chance of winning the series
## Team2: DEN has a 51 % chance of winning the series
## Games played = 7
## Expected Series result : 4 - 3
```

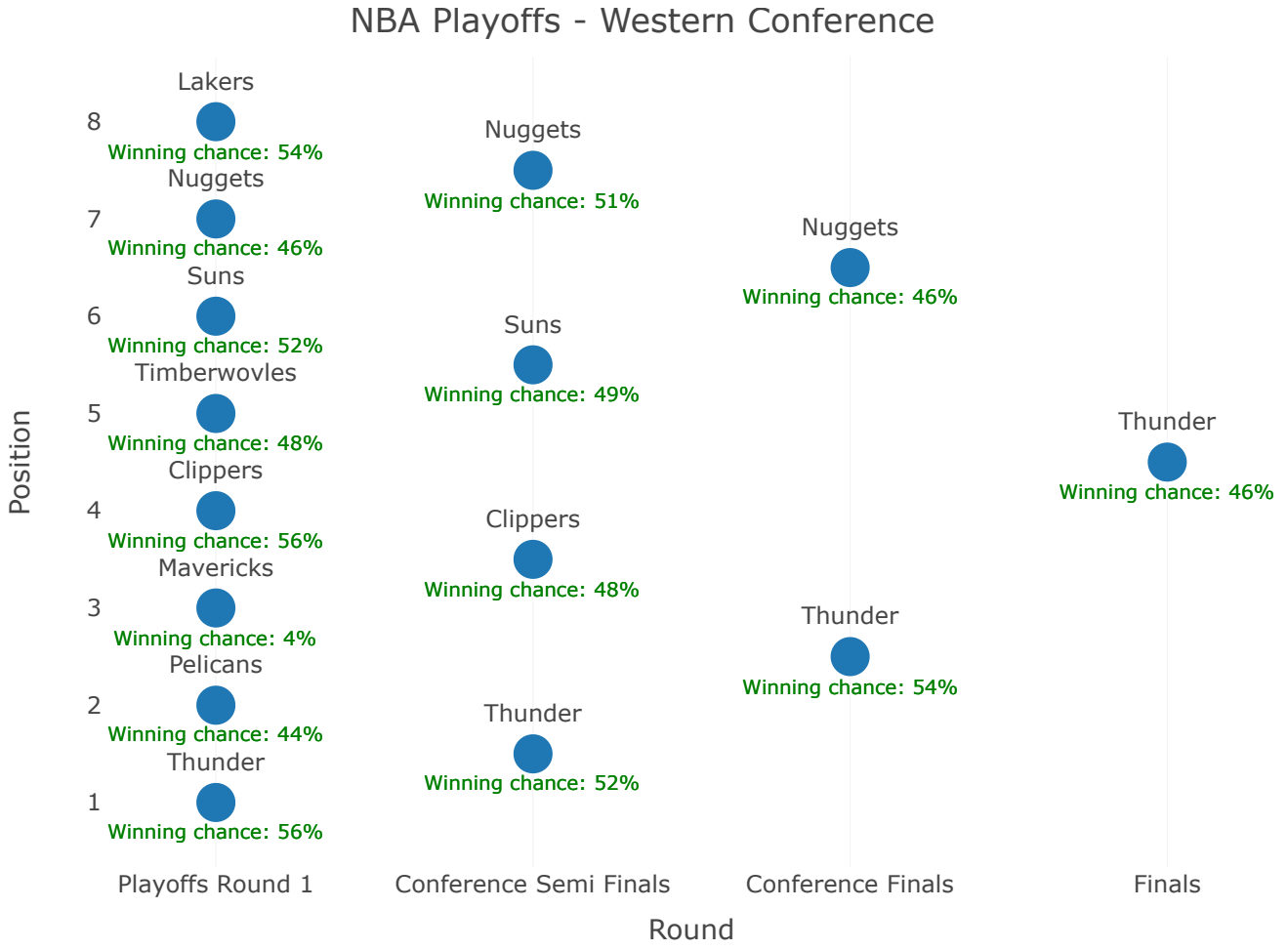
Conference Finals

```
result <- calculate_win_percentages(2023, "DEN", "OKC", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: DEN has a 46 % chance of winning the series
## Team2: OKC has a 54 % chance of winning the series
## Games played = 6
## Expected Series result : 2 - 4
```

```
library(plotly)

# Define the positions for each team in the bracket
teams <- c("Thunder", "Pelicans", "Mavericks", "Clippers", "Timberwovles", "Suns", "Nuggets",
"Lakers",
           "Thunder", "Clippers", "Suns", "Nuggets",
           "Thunder", "Nuggets",
           "Thunder")
rounds <- c(1, 1, 1, 1, 1, 1, 1, 1,
            2, 2, 2, 2,
            3, 3,
            4)
positions <- c(1, 2, 3, 4, 5, 6, 7, 8,
               1.5, 3.5, 5.5, 7.5,
               2.5, 6.5,
               4.5)

# Define the winning chances for each team
winning_chances <- c("56%", "44%", "4%", "56%", "48%", "52%", "46%", "54%",
                     "52%", "48%", "49%", "51%",
                     "54%", "46%",
                     "46%")

# Create a data frame for plotting
bracket_df <- data.frame(team = teams, round = rounds, position = positions, winning_chance =
winning_chances)

# Plot using plotly
p <- plot_ly()

# Add the nodes with team names
p <- p %>% add_trace(type = 'scatter', mode = 'markers+text',
                     x = bracket_df$round, y = bracket_df$position,
                     text = bracket_df$team, textposition = 'top center',
                     marker = list(size = 20),
                     showlegend = FALSE)

# Add annotations for winning chances
for (i in 1:nrow(bracket_df)) {
  # Add winning chances below team names
  p <- p %>% add_annotations(
    x = bracket_df$round[i],
    y = bracket_df$position[i] - 0.3,
    text = paste("Winning chance:", bracket_df$winning_chance[i]),
    showarrow = FALSE,
    font = list(size = 10, color = 'green')
  )
}

# Customize layout
p <- p %>% layout(title = 'NBA Playoffs - Western Conference',
                  xaxis = list(title = 'Round', tickvals = 1:4, ticktext = c("Playoffs Round
1","Conference Semi Finals","Conference Finals","Finals")),
                  yaxis = list(title = 'Position', showgrid = FALSE, zeroline = FALSE),
                  showlegend = FALSE)
```

```
# Show plot
p
```




```
result <- calculate_win_percentages(2023, "BOS", "OKC", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: BOS has a 54 % chance of winning the series
## Team2: OKC has a 46 % chance of winning the series
## Games played = 6
## Expected Series result : 4 - 2
```

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. If given more time and data, how would you use what you found to improve your model?

ANSWER :

Team 1: Boston Celtics

Competitive Window:

2021-2022 Season: Made it to the NBA Finals. 2022-2023 Season: Lost to the Miami Heat in the Eastern Conference Finals.

```
result <- calculate_win_percentages(2022, "BOS", "MIA", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: BOS has a 56 % chance of winning the series
## Team2: MIA has a 44 % chance of winning the series
## Games played = 6
## Expected Series result : 4 - 2
```

```
play_bos <- team_data%>%
  filter(season == 2022 & gametype == 4 & off_team == 'BOS' & def_team == 'MIA')
play_bos
```

```
## # A tibble: 7 × 41
##   season gametype nbagameid gamedate   offensivenbateamid off_team_name off_team
##    <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>         <chr>
## 1   2022        4  42200301 2023-05-17         1610612738 Boston Celti… BOS
## 2   2022        4  42200302 2023-05-19         1610612738 Boston Celti… BOS
## 3   2022        4  42200303 2023-05-21         1610612738 Boston Celti… BOS
## 4   2022        4  42200304 2023-05-23         1610612738 Boston Celti… BOS
## 5   2022        4  42200305 2023-05-25         1610612738 Boston Celti… BOS
## 6   2022        4  42200306 2023-05-27         1610612738 Boston Celti… BOS
## 7   2022        4  42200307 2023-05-29         1610612738 Boston Celti… BOS
## # ℹ 34 more variables: off_home <dbl>, off_win <dbl>, defensivenbateamid <dbl>,
## #   def_team_name <chr>, def_team <chr>, def_home <dbl>, def_win <dbl>,
## #   fg2made <dbl>, fg2missed <dbl>, fg2attempted <dbl>, fg3made <dbl>,
## #   fg3missed <dbl>, fg3attempted <dbl>, fgmade <dbl>, fgmissed <dbl>,
## #   fgattempted <dbl>, ftmade <dbl>, ftmissed <dbl>, ftattempted <dbl>,
## #   reboffensive <dbl>, rebdefensive <dbl>, reboundchance <dbl>, assists <dbl>,
## #   stealsagainst <dbl>, turnovers <dbl>, blocksagainst <dbl>, …
```

```
play_mia <- team_data%>%
  filter(season == 2022 & gametype == 4 & off_team == 'MIA' & def_team == 'BOS')
play_mia
```

```
## # A tibble: 7 × 41
##   season gametype nbagameid gamedate   offensivenbateamid off_team_name off_team
##    <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>         <chr>
## 1   2022        4  42200301 2023-05-17         1610612748 Miami Heat    MIA
## 2   2022        4  42200302 2023-05-19         1610612748 Miami Heat    MIA
## 3   2022        4  42200303 2023-05-21         1610612748 Miami Heat    MIA
## 4   2022        4  42200304 2023-05-23         1610612748 Miami Heat    MIA
## 5   2022        4  42200305 2023-05-25         1610612748 Miami Heat    MIA
## 6   2022        4  42200306 2023-05-27         1610612748 Miami Heat    MIA
## 7   2022        4  42200307 2023-05-29         1610612748 Miami Heat    MIA
## # ℹ 34 more variables: off_home <dbl>, off_win <dbl>, defensivenbateamid <dbl>,
## #   def_team_name <chr>, def_team <chr>, def_home <dbl>, def_win <dbl>,
## #   fg2made <dbl>, fg2missed <dbl>, fg2attempted <dbl>, fg3made <dbl>,
## #   fg3missed <dbl>, fg3attempted <dbl>, fgmade <dbl>, fgmissed <dbl>,
## #   fgattempted <dbl>, ftmade <dbl>, ftmissed <dbl>, ftattempted <dbl>,
## #   reboffensive <dbl>, rebdefensive <dbl>, reboundchance <dbl>, assists <dbl>,
## #   stealsagainst <dbl>, turnovers <dbl>, blocksagainst <dbl>, …
```

Model predicts that Celtics have higher probability of winning the series.

Performance Analysis:

During the 2022-2023 season, the Celtics were expected to advance past the Miami Heat in the Eastern Conference Finals but lost the series.

Possible Reasons for Under performance:

Shooting Efficiency (3-Point Percentage): The Celtics had a significant drop in their 3-point shooting accuracy in key games of the series. For a team that relies heavily on perimeter shooting, this decline was critical.

Injuries:
Player Health: Key players may have suffered from injuries or were not at full strength, impacting their ability to perform in crucial games.

Close Games:
Overtime and Clutch Situations: As we can observe in the above data multiple games in the series were decided by a narrow margin. Such close outcomes often come down to individual moments rather than predictable performance trends.

Team 2: New York Knicks

Competitive Window:

2022-2023 Season: Made it to the Conference Semi Finals. 2022-2023 Season: Lost to the Indiana Pacers in the Eastern Conference SemiFinals.

```
result <- calculate_win_percentages(2023, "NYK", "IND", logistic_model, final_dataset, select
ed_columns)
```

```
## Team1: NYK has a 53 % chance of winning the series
## Team2: IND has a 47 % chance of winning the series
## Games played = 6
## Expected Series result : 2 - 4
```

Model predicts that Knicks have higher probability of winning the series.

Performance Analysis:

During the 2022-2023 season, the Knicks were expected to advance past the Indiana Pacers but lost the series. They were in a 3-3 series tie but lost the decisive Game 7.

Reasons for Underperformance:

Unpredictable Performance by Opponent: In Game 7, a Pacers player set a record for shots made, which was an extraordinary performance and an outlier.

Team Fatigue: By Game 7, fatigue could have set in, affecting the Knicks' ability to perform at their peak.
Pressure and Nerves: The high stakes of a Game 7 can lead to increased pressure on players, potentially impacting their performance.

Classification: Bad Luck: The exceptional performance by a Pacers player in Game 7 is a form of bad luck. Such outlier performances are rare and difficult to predict or account for in models.

Model Improvement:

To further refine the predictive model considering the insights from both the Boston Celtics and New York Knicks' underperformance, we can integrate the following enhancements:

Factor in Pressure and Fatigue: Add variables that assess the impact of pressure and fatigue on team performance, such as reduced shooting percentages or increased turnovers in decisive games.

Incorporate Shooting Variance: Add a factor to account for game-to-game shooting variance, particularly for three-point shooting and identify patterns in shooting performance, including hot and cold streaks.

Player Health Index: Develop a metric to track player health and injuries, incorporating their potential impact on performance.

Clutch Performance: Include metrics that capture team and player performance in clutch situations (last five minutes of a game within five points).

Incorporate Opponent Adjustments: Introduce a component that evaluates the adaptability of teams to opponent strategies, considering the effectiveness of defensive and offensive adjustments over a series.

Psychological and Situational Factors: Consider psychological metrics such as player confidence, historical performance in elimination games, and team morale. Incorporating these factors into the model may provide a more comprehensive view of how situational and psychological aspects impact playoff outcomes.

If given more time and data, by integrating these enhancements, the predictive model will be better equipped to account for the nuances of playoff basketball, including bad luck scenarios and factors not previously considered. This will improve the model's accuracy and reliability in forecasting playoff performance.