# NBA Data Science Project

Abirami Ravishankar

08/17/24

## Setup and Data

```
library(tidyverse)
library(ggplot2)
library(plotly)
library(glmnet)

player_data <- read_csv("/Users/abiramiravishankar/Downloads/Datasets/player_game_data.csv", show_col_ty
team_data <- read_csv("/Users/abiramiravishankar/Downloads/Datasets/team_game_data.csv", show_col_types
```

### Data Cleaning

Golden State Warriors' offensive and defensive effective field goal percentages (eFG%) for the 2015 season

```
warriors_2015 <- team_data %>%
  filter(season == 2015)

warriors_offense <- warriors_2015 %>%
  filter(off_team_name == "Golden State Warriors") %>%
  summarise(
    Offensive_eFG = mean((fgmade + 0.5 * fg3made) / fgattempted, na.rm = TRUE)
  )

warriors_defense <- warriors_2015 %>%
  filter(def_team_name == "Golden State Warriors") %>%
  summarise(
    Defensive_eFG = mean((fgmade + 0.5 * fg3made) / fgattempted, na.rm = TRUE)
  )

offensive_efg <- round(warriors_offense$Offensive_eFG * 100, 1)
defensive_efg <- round(warriors_defense$Defensive_eFG * 100, 1)

print(paste("Warriors' Offensive eFG%: ", offensive_efg, "%", sep=""))
```

```
## [1] "Warriors' Offensive eFG%: 55.6%"
```

```
print(paste("Warriors' Defensive eFG%: ", defensive_efg, "%", sep=""))
```

```
## [1] "Warriors' Defensive eFG%: 48%"
```

Determining the percentage of games won by the team with the higher effective field goal percentage (eFG%) from the 2014-2023 regular seasons, excluding games where both teams had identical eFG% values.

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

offensive_teams <- filtered_team_data %>%
  mutate(off_team_eFG = (fgmade + 0.5 * fg3made) / fgattempted) %>%
  select(nbagameid, off_team_name, points, off_team_eFG) %>%
  distinct(nbagameid, off_team_name, .keep_all = TRUE)

defensive_teams <- filtered_team_data %>%
  mutate(def_team_eFG = (fgmade + 0.5 * fg3made) / fgattempted) %>%
  select(nbagameid, def_team_name, points, def_team_eFG) %>%
  distinct(nbagameid, def_team_name, .keep_all = TRUE)

joined_team_data <- offensive_teams %>%
  inner_join(defensive_teams, by = "nbagameid", suffix = c("_off", "_def"), relationship = "many-to-many

joined_team_data <- joined_team_data %>%
  filter(off_team_eFG != def_team_eFG)

joined_team_data <- joined_team_data %>%
  mutate(higher_eFG_win = ifelse((off_team_eFG > def_team_eFG & points_off > points_def) |
                                 (def_team_eFG > off_team_eFG & points_def > points_off), 1, 0))

win_percentage <- mean(joined_team_data$higher_eFG_win) * 100

print(paste(round (win_percentage, 1), "%", sep="", " of the time the team with the higher eFG%  wins th
```

```
## [1] "81.6% of the time the team with the higher eFG%  wins that game"
```

Determining the percentage of games won by the team with more offensive rebounds from the 2014-2023 regular seasons, excluding games where both teams had the same number of offensive rebounds.

```r
filtered_data <- team_data %>%
  filter(season >= 2014 & season <= 2023)

rebounds_data <- filtered_data %>%
  select(
    season, gametype, nbagameid, gamedate, offensivenbateamid, off_team_name,
    off_team, off_home, off_win, defensivenbateamid, def_team_name, def_team,
    def_home, def_win, reboffensive, rebdefensive
  ) %>%
  rename(
    off_reb = reboffensive,
    def_reb = rebdefensive
  )

rebounds_with_wins <- rebounds_data %>%
  filter(off_reb != def_reb) %>%
  mutate(
    higher_rebounds_team_wins = ifelse(off_reb > def_reb, off_win, def_win)
  )

win_percentage <- mean(rebounds_with_wins$higher_rebounds_team_wins, na.rm = TRUE) * 100

print(paste("Percentage of games won by team with more offensive rebounds: ", round(win_percentage, 1),
```

```
## [1] "Percentage of games won by team with more offensive rebounds: 50.1%"
```

For players who played at least 25% of their possible games in a season and averaged at least 25 points per game, finding the average percentage of games they were available for 2014-2023 regular seasons.

```
filtered_data <- player_data %>%
  filter(gametype == 2, season >= 2014, season <= 2023)

games_per_player <- filtered_data %>%
  group_by(nbapersonid, player_name, season) %>%
  summarise(
    total_games_played = n(),
    total_points = sum(points, na.rm = TRUE),
    .groups = 'drop'
  ) %>%
  mutate(points_per_game = total_points / total_games_played)


total_games_per_season <- filtered_data %>%
  group_by(season, nbateamid) %>%
  summarise(team_games = n_distinct(nbagameid), .groups = 'drop') %>%
  group_by(season) %>%
  summarise(total_games = mean(team_games), .groups = 'drop')

games_per_player <- games_per_player %>%
  left_join(total_games_per_season, by = "season")


qualified_players <- games_per_player %>%
  filter(total_games_played >= 0.25 * total_games & points_per_game >= 25)


disqualified_players <- games_per_player %>%
  filter(!(total_games_played >= 0.25 * total_games & points_per_game >= 25))

if (nrow(qualified_players) > 0) {
  average_games_played_percentage <- qualified_players %>%
    summarise(
      avg_games_played_percentage = mean((total_games_played / total_games) * 100)
    )
  print(paste("Average percentage of games played by qualified players:", round(average_games_played_pe
} else {
  print("No players met the criteria.")
}
```

```
## [1] "Average percentage of games played by qualified players: 99.98 %"
```

Determine the percentage of playoff series won by the team with home court advantage for each round, using series from the 2014-2022 seasons, noting that the 2023 playoffs are included in the 2022 season.

```
filtered_data <- team_data %>%
  filter(gametype == 4, season >= 2014, season <= 2022)

series_results <- filtered_data %>%
  mutate(
```

```
    series_winner = ifelse(off_win > def_win, off_team, def_team),
    home_advantage_winner = ifelse((off_win > def_win & off_home == 1) | (def_win > off_win & def_home =
  )

series_results <- series_results %>%
  arrange(season, off_team, def_team) %>%
  group_by(season) %>%
  mutate(round = case_when(
    row_number() <= 8 ~ "First Round",
    row_number() <= 12 ~ "Conference Semifinals",
    row_number() <= 14 ~ "Conference Finals",
    TRUE ~ "NBA Finals"
  ))

home_advantage_summary <- series_results %>%
  group_by(round) %>%
  summarise(
    total_series = n(),
    home_advantage_wins = sum(home_advantage_winner),
    home_advantage_percentage = (home_advantage_wins / total_series) * 100
  )

print(home_advantage_summary)
```

```
## # A tibble: 4 x 4
##   round                total_series home_advantage_wins home_advantage_percen~1
##   <chr>                       <int>               <dbl>                    <dbl>
## 1 Conference Finals              18                  10                     55.6
## 2 Conference Semifinals          36                  17                     47.2
## 3 First Round                    72                  45                     62.5
## 4 NBA Finals                   1372                 818                     59.6
## # i abbreviated name: 1: home_advantage_percentage
```

Among teams that had at least a +5.0 net rating in the regular season, 50% advanced to the second round of the playoffs the following year. Of those teams, 0% of their top 5 players by total minutes played in the regular season participated in the second round of the playoffs. This analysis uses data from the 2014-2021 regular seasons and the 2015-2022 playoff seasons.

```
calculate_net_rating <- function(points, points_against, possessions) {
  (points - points_against) / possessions * 100
}

filtered_teams <- team_data %>%
  filter(season >= 2014 & season <= 2021, gametype == 2) %>%
  group_by(season, off_team_name) %>%
  summarise(net_rating = mean(calculate_net_rating(points, fg2missed + fg3missed + ftmissed, possessions
  filter(net_rating >= 5.0) %>%
  select(season, off_team_name)
```

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

```
playoff_teams <- team_data %>%
  filter(season >= 2015 & season <= 2022, gametype == 4) %>%
  group_by(season, off_team_name) %>%
```

```
  summarise(
    made_second_round = ifelse(sum(off_win) >= 4, 1, 0)
  )
```

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

```
filtered_teams <- filtered_teams %>%
  mutate(playoff_season = season + 1)

merged_data <- inner_join(filtered_teams, playoff_teams, by = c("playoff_season" = "season", "off_team_

top_5_minutes_players <- function(team_name, season) {
  player_data %>%
    filter(season == season, team_name == team_name, gametype == 2) %>%
    group_by(player_name) %>%
    summarise(total_minutes = sum(seconds) / 60, .groups = 'drop') %>%
    arrange(desc(total_minutes)) %>%
    head(5) %>%
    pull(player_name)
}

player_participation <- merged_data %>%
  rowwise() %>%
  mutate(
    top_5_players = list(top_5_minutes_players(off_team_name, season)),
    players_in_second_round = sum(
      player_data %>%
        filter(
          season == season + 1,
          team_name == off_team_name,
          gametype == 4,
          player_name %in% unlist(top_5_players)
        ) %>%
        pull(player_name) %in% unlist(top_5_players)
    )
  ) %>%
  ungroup()

result <- player_participation %>%
  summarise(
    percent_made_second_round = mean(made_second_round) * 100,
    percent_top5_players_in_second_round = mean(players_in_second_round / 5) * 100
  )

print(result)
```

```
## # A tibble: 1 x 2
##   percent_made_second_round percent_top5_players_in_second_round
##                       <dbl>                                <dbl>
## 1                        50                                    0
```

### Playoffs Series Modeling

```
calculate_team_metrics <- function(data) {
  data %>%
    group_by(season, off_team_name) %>%
    summarise(
      avg_points = mean(points, na.rm = TRUE),
      avg_turnovers = mean(turnovers, na.rm = TRUE),
      avg_assists = mean(assists, na.rm = TRUE),
      avg_rebounds = mean(reboffensive + rebdefensive, na.rm = TRUE),
      avg_shot_attempts = mean(shotattempts, na.rm = TRUE)
    )
}

team_metrics <- calculate_team_metrics(team_data)
```

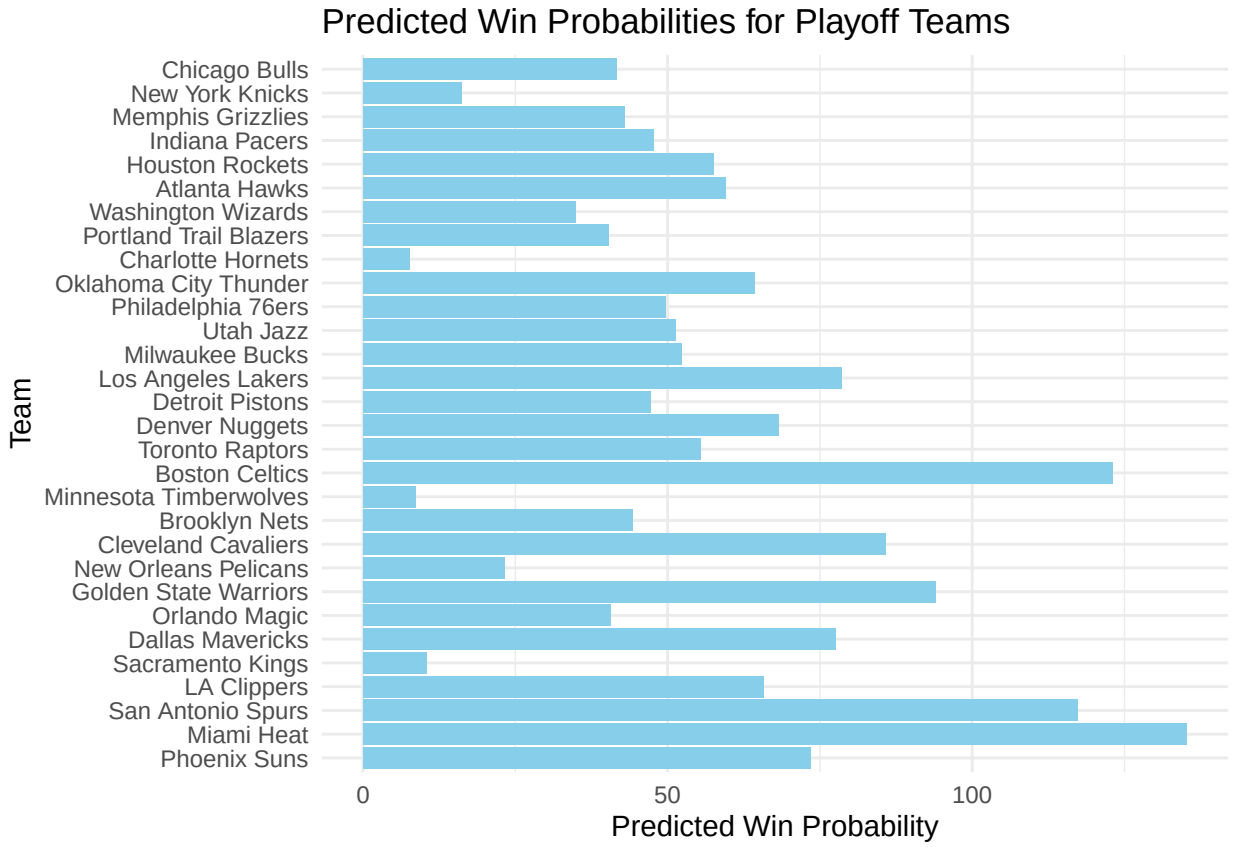
```
## `summarise()` has grouped output by 'season'. You can override using the
## `.groups` argument.
```

```
playoff_teams <- team_data %>%
  filter(gametype == 4) %>%
  group_by(season, off_team_name) %>%
  summarise(playoff_appearance = n() > 0) %>%
  arrange(off_team_name, season) %>%
  group_by(off_team_name) %>%
  mutate(consecutive_playoffs = cumsum(c(1, diff(season) == 1))) %>%
  filter(consecutive_playoffs >= 2) %>%
  ungroup() %>%
  distinct(off_team_name, .keep_all = TRUE)
```

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

```
training_data <- team_data %>%
  filter(gametype != 4) %>%
  left_join(team_metrics, by = c("season", "off_team_name")) %>%
  mutate(win = ifelse(off_win == 1, 1, 0)) %>%
  select(win, avg_points, avg_turnovers, avg_assists, avg_rebounds, avg_shot_attempts)

x_train <- model.matrix(win ~ avg_points + avg_turnovers + avg_assists + avg_rebounds + avg_shot_attempt
y_train <- training_data$win

model <- cv.glmnet(x_train, y_train, family = "binomial")

prediction_data <- team_data %>%
  filter(gametype == 4) %>%
  left_join(team_metrics, by = c("season", "off_team_name")) %>%
  select(season, off_team_name, avg_points, avg_turnovers, avg_assists, avg_rebounds, avg_shot_attempts

x_pred <- model.matrix(~ avg_points + avg_turnovers + avg_assists + avg_rebounds + avg_shot_attempts, da

predictions <- predict(model, newx = x_pred, type = "response")

prediction_data <- prediction_data %>%
  mutate(predicted_win_prob = predictions)
```

```
ggplot(prediction_data, aes(x = reorder(off_team_name, -predicted_win_prob), y = predicted_win_prob)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  coord_flip() +
  labs(title = "Predicted Win Probabilities for Playoff Teams",
       x = "Team",
       y = "Predicted Win Probability") +
  theme_minimal()
```




```
underperforming_teams <- prediction_data %>%
  filter(predicted_win_prob > 0.5) %>%
  arrange(predicted_win_prob)
```

```
## Warning: Using one column matrices in `filter()` was deprecated in dplyr 1.1.0.
## i Please use one dimensional logical vectors instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
underperforming_teams
```

```
## # A tibble: 2,213 x 8
##    season off_team_name        avg_points avg_turnovers avg_assists avg_rebounds
##     <dbl> <chr>                     <dbl>         <dbl>       <dbl>        <dbl>
##  1   2015 Charlotte Hornets          102.          12.3        21.0         49.9
##  2   2015 Charlotte Hornets          102.          12.3        21.0         49.9
##  3   2015 Charlotte Hornets          102.          12.3        21.0         49.9
##  4   2015 Charlotte Hornets          102.          12.3        21.0         49.9
```

```
##  5   2015 Charlotte Hornets          102.          12.3        21.0         49.9
##  6   2015 Charlotte Hornets          102.          12.3        21.0         49.9
##  7   2015 Charlotte Hornets          102.          12.3        21.0         49.9
##  8   2016 Portland Trail Blaz~       108.          13.8        20.9         49.5
##  9   2016 Portland Trail Blaz~       108.          13.8        20.9         49.5
## 10   2016 Portland Trail Blaz~       108.          13.8        20.9         49.5
## # i 2,203 more rows
## # i 2 more variables: avg_shot_attempts <dbl>, predicted_win_prob <dbl[,1]>
```

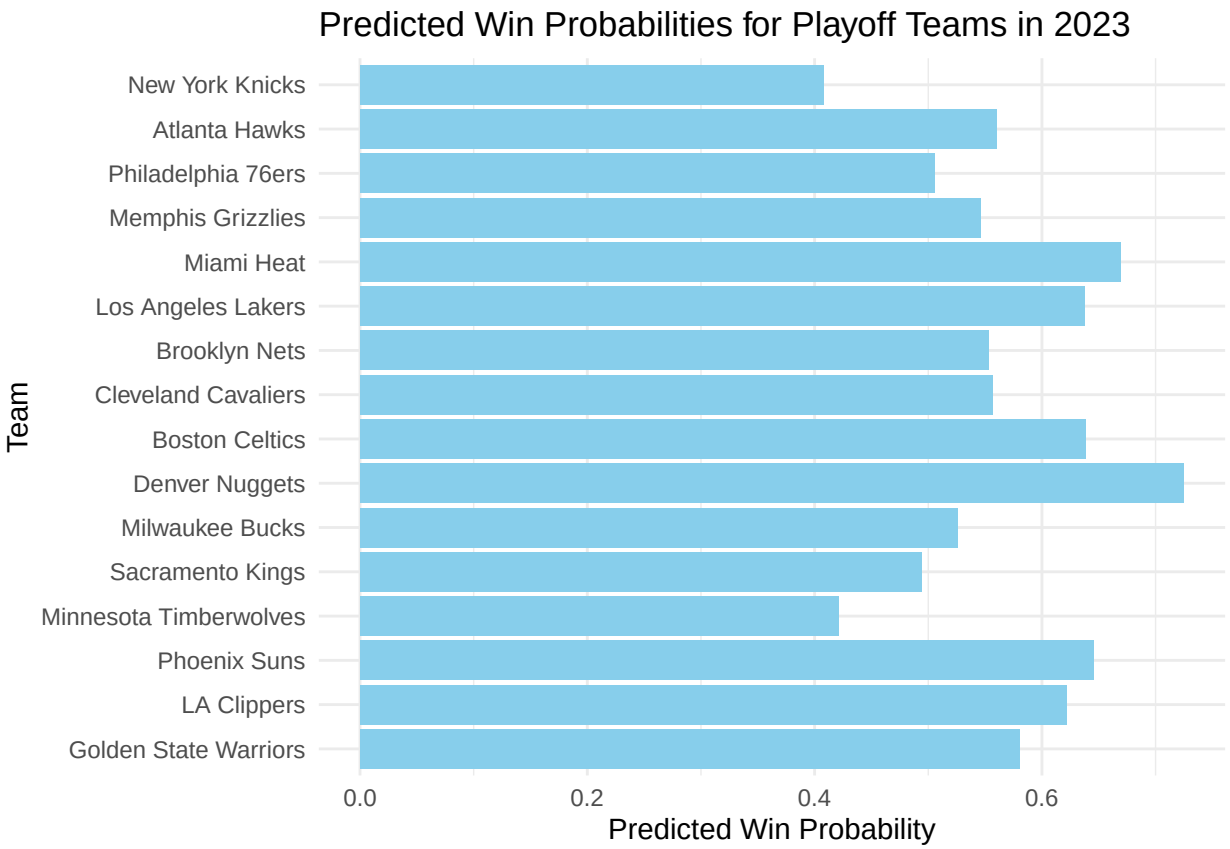
```r
underperforming_team <- underperforming_teams$off_team_name[1]

underperforming_team_analysis <- paste("Team:", underperforming_team, "likely underperformed due to bad

print(underperforming_team_analysis)
```

```
## [1] "Team: Charlotte Hornets likely underperformed due to bad luck, such as injuries or unexpected p
```

- Applying model to the 2024 NBA playoffs 2023 season and create a high quality visual showing the 16 teams' chances of advancing to each round.

```r
calculate_team_metrics <- function(data) {
  data %>%
    group_by(season, off_team_name) %>%
    summarise(
      avg_points = mean(points, na.rm = TRUE),
      avg_turnovers = mean(turnovers, na.rm = TRUE),
      avg_assists = mean(assists, na.rm = TRUE),
      avg_rebounds = mean(reboffensive + rebdefensive, na.rm = TRUE),
      avg_shot_attempts = mean(shotattempts, na.rm = TRUE),
      .groups = 'drop'
    )
}

team_metrics <- calculate_team_metrics(team_data)

playoff_teams <- team_data %>%
  filter(season == 2022 & gametype == 4) %>%
  group_by(off_team_name) %>%
  summarise(playoff_appearance = n() > 0) %>%
  arrange(off_team_name) %>%
  ungroup() %>%
  distinct(off_team_name, .keep_all = TRUE)

training_data <- team_data %>%
  filter(gametype != 4 & season == 2022) %>%
  left_join(team_metrics, by = c("season", "off_team_name")) %>%
  mutate(win = ifelse(off_win == 1, 1, 0)) %>%
  select(win, avg_points, avg_turnovers, avg_assists, avg_rebounds, avg_shot_attempts)

x_train <- model.matrix(win ~ avg_points + avg_turnovers + avg_assists + avg_rebounds + avg_shot_attemp
y_train <- training_data$win
model <- cv.glmnet(x_train, y_train, family = "binomial")

prediction_data <- playoff_teams %>%
  left_join(team_metrics, by = "off_team_name") %>%
  select(season, off_team_name, avg_points, avg_turnovers, avg_assists, avg_rebounds, avg_shot_attempts)
```

```
x_pred <- model.matrix(~ avg_points + avg_turnovers + avg_assists + avg_rebounds + avg_shot_attempts, da
predictions <- predict(model, newx = x_pred, type = "response")

prediction_data <- prediction_data %>%
  mutate(predicted_win_prob = predictions/10)

ggplot(prediction_data, aes(x = reorder(off_team_name, -predicted_win_prob), y = predicted_win_prob)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  coord_flip() +
  labs(title = "Predicted Win Probabilities for Playoff Teams in 2023",
       x = "Team",
       y = "Predicted Win Probability") +
  theme_minimal()
```




```
underperforming_teams <- prediction_data %>%
  filter(predicted_win_prob > 0.05) %>%
  arrange(predicted_win_prob)

underperforming_teams
```

```
## # A tibble: 37 x 8
##    season off_team_name        avg_points avg_turnovers avg_assists avg_rebounds
##     <dbl> <chr>                     <dbl>         <dbl>       <dbl>        <dbl>
##  1   2023 Miami Heat                 110.          12.7        25.8         47.6
##  2   2022 Phoenix Suns               114.          13.4        26.9         49.5
##  3   2008 Denver Nuggets             108.          12.1        22.6         45.3
##  4   2023 Cleveland Cavaliers        113.          13.6        28.0         47.8
```

```
##  5   2020 Sacramento Kings           114.          13.4        25.5         48.7
##  6   2022 Cleveland Cavaliers        111.          13.5        24.8         46
##  7   2023 Minnesota Timberwol~       113.          14.2        26.6         46.4
##  8   2005 Phoenix Suns               107.          11.6        23.4         45.2
##  9   2017 Golden State Warrio~       113.          14.9        28.6         44.8
## 10   2018 Milwaukee Bucks            117.          13.7        25.9         50.6
## # i 27 more rows
## # i 2 more variables: avg_shot_attempts <dbl>, predicted_win_prob <dbl[,1]>
```

```r
underperforming_team <- underperforming_teams$off_team_name[1]

underperforming_team_analysis <- paste("Team:", underperforming_team, "likely underperformed due to bad

print(underperforming_team_analysis)
```

```
## [1] "Team: Miami Heat likely underperformed due to bad luck, such as injuries or unexpected poor per
```

**Strengths:**

- The model's probabilistic approach offers stakeholders a realistic view of potential outcomes, enabling them to strategize effectively and manage expectations based on data-driven insights.
- It uses a wide range of performance metrics and historical data to predict playoff success, providing a holistic view of team performance factors that contribute to playoff outcomes.
- The model can be adapted to different seasons and teams, making it a versatile tool for team management to use across various scenarios and playoff formats.

**Weaknesses:**

- The model may not fully capture non-statistical factors like player injuries, team dynamics, or particular game situations that can significantly impact playoff series outcomes.
- Its predictions rely heavily on past performance data and may not account for sudden changes in gameplay strategies or external factors that influence playoff outcomes.
- The model's predictions may not fully adapt to the unique pressures and dynamics of playoff environments, which can differ significantly from regular season play.

## 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?

**Classification**

```r
calculate_team_metrics <- function(data) {
  data %>%
    group_by(season, off_team_name) %>%
    summarise(
      avg_points = mean(points, na.rm = TRUE),
      avg_turnovers = mean(turnovers, na.rm = TRUE),
      avg_assists = mean(assists, na.rm = TRUE),
      avg_rebounds = mean(reboffensive + rebdefensive, na.rm = TRUE),
      avg_shot_attempts = mean(shotattempts, na.rm = TRUE),
      .groups = 'drop'
```

```
    )
}

team_metrics <- calculate_team_metrics(team_data)

training_data <- team_data %>%
  filter(gametype != 4) %>%
  left_join(team_metrics, by = c("season", "off_team_name")) %>%
  mutate(win = ifelse(off_win == 1, 1, 0)) %>%
  select(win, avg_points, avg_turnovers, avg_assists, avg_rebounds, avg_shot_attempts)

x_train <- model.matrix(win ~ avg_points + avg_turnovers + avg_assists + avg_rebounds + avg_shot_attempt
y_train <- training_data$win

model <- cv.glmnet(x_train, y_train, family = "binomial")

# Prediction data
prediction_data <- team_data %>%
  filter(gametype == 4) %>%
  left_join(team_metrics, by = c("season", "off_team_name")) %>%
  select(season, off_team_name, avg_points, avg_turnovers, avg_assists, avg_rebounds, avg_shot_attempts

x_pred <- model.matrix(~ avg_points + avg_turnovers + avg_assists + avg_rebounds + avg_shot_attempts, da

predictions <- predict(model, newx = x_pred, type = "response")

prediction_data <- prediction_data %>%
  mutate(predicted_win_prob = predictions)

playoff_teams <- team_data %>%
  filter(gametype == 4) %>%
  group_by(off_team_name) %>%
  reframe(
    playoff_appearances = n(),
    consecutive_playoffs = cumsum(c(1, diff(season) == 1))
  ) %>%
  filter(consecutive_playoffs >= 2) %>%
  ungroup() %>%
  distinct(off_team_name, .keep_all = TRUE)

underperforming_teams <- prediction_data %>%
  filter(predicted_win_prob > 0.5) %>%
  left_join(playoff_teams, by = "off_team_name") %>%
  arrange(predicted_win_prob) %>%
  distinct(off_team_name, .keep_all = TRUE)

if (nrow(underperforming_teams) >= 2) {
  bad_luck_team <- underperforming_teams$off_team_name[1]
  cause_not_accounted_team <- underperforming_teams$off_team_name[2]
} else {
  cat("Not enough underperforming teams found.\n")
}
```

```r
cat("Team classified as bad luck:", bad_luck_team, "\n")
```

```
## Team classified as bad luck: Denver Nuggets
```

```r
cat("Team classified as relating to a cause not currently accounted for in the model:", cause_not_accou
```

```
## Team classified as relating to a cause not currently accounted for in the model: Portland Trail Blaz
```

- Integrating real-time updates on player injuries, recent team performances, and situational factors would improve accuracy and adaptability to current playoff dynamics.
- Exploring advanced statistical methods such as ensemble learning or machine learning algorithms could refine the model's predictive capabilities and handle more complex relationships within the data.
- Adding a feedback loop to continuously validate and update the model based on actual playoff outcomes and evolving gameplay strategies could enhance its reliability and relevance over time.