Predicting NBA Shot Success

Rylen Grundy

2025-02-12

Load Packages

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.4.2
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.4.2
```

```
##
## Attaching package: 'dplyr'
```

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

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

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.3
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.3
```

```
## Loading required package: lattice
```

```
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.4.3
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library(keras)
```

```
## Warning: package 'keras' was built under R version 4.4.3
```

```
library(recipes)
```

```
## Warning: package 'recipes' was built under R version 4.4.2
```

```
##
## Attaching package: 'recipes'
```

```
## The following object is masked from 'package:stats':
##
##     step
```

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.4.2
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

Load and Create Data Set

Loading all the datasets of shot data from years

2004-2024

```
NBA_2004_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2004_Shots.cs
v')
NBA_2005_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2005_Shots.cs
v')
NBA_2006_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2006_Shots.cs
v')
NBA_2007_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2007_Shots.cs
v')
NBA_2008_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2008_Shots.cs
v')
NBA_2009_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2009_Shots.cs
v')
NBA_2010_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2010_Shots.cs
v')
NBA_2011_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2011_Shots.cs
v')
NBA_2012_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2012_Shots.cs
v')
NBA_2013_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2013_Shots.cs
v')
NBA_2014_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2014_Shots.cs
v')
NBA_2015_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2015_Shots.cs
v')
NBA_2016_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2016_Shots.cs
v')
NBA_2017_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2017_Shots.cs
v')
NBA_2018_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2018_Shots.cs
v')
NBA_2019_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2019_Shots.cs
v')
NBA_2020_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2020_Shots.cs
v')
NBA_2021_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2021_Shots.cs
v')
NBA_2022_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2022_Shots.cs
v')
NBA_2023_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2023_Shots.cs
v')
NBA_2024_Shots = read.csv('C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData/NBA_2024_Shots.cs
v')
```

Combine all of the years into one dataset.

```
shots = list.files(path = "C:/Users/rgrun/Spring 2025/Senior Thesis/ShotData", pattern = "*.cs
v", full.names = TRUE) %>%
  lapply(read.csv)%>%
  bind_rows()
shots = as.data.frame(shots)
```

Inspect and clean data

Inspect Variable Types

```
glimpse(shots)
```

```
## Rows: 4,231,262
## Columns: 26
## $ SEASON_1       <int> 2004, 2004, 2004, 2004, 2004, 2004, 2004, 2004, 2004, 2…
## $ SEASON_2       <chr> "2003-04", "2003-04", "2003-04", "2003-04", "2003-04", …
## $ TEAM_ID        <int> 1610612747, 1610612757, 1610612747, 1610612757, 1610612…
## $ TEAM_NAME      <chr> "Los Angeles Lakers", "Portland Trail Blazers", "Los An…
## $ PLAYER_ID      <int> 977, 757, 977, 757, 757, 2567, 757, 977, 1544, 977, 221…
## $ PLAYER_NAME    <chr> "Kobe Bryant", "Damon Stoudamire", "Kobe Bryant", "Damo…
## $ POSITION_GROUP <chr> "G", "G", "G", "G", "G", "C", "G", "G", "F", "G", "F", …
## $ POSITION       <chr> "SG", "PG", "SG", "PG", "PG", "C", "PG", "SG", "PF", "S…
## $ GAME_DATE      <chr> "04-14-2004", "04-14-2004", "04-14-2004", "04-14-2004",…
## $ GAME_ID        <int> 20301187, 20301187, 20301187, 20301187, 20301187, 20301…
## $ HOME_TEAM      <chr> "POR", "POR", "POR", "POR", "POR", "POR", "POR", "POR",…
## $ AWAY_TEAM      <chr> "LAL", "LAL", "LAL", "LAL", "LAL", "LAL", "LAL", "LAL",…
## $ EVENT_TYPE     <chr> "Made Shot", "Made Shot", "Missed Shot", "Made Shot", "…
## $ SHOT_MADE      <lgl> TRUE, TRUE, FALSE, TRUE, FALSE, TRUE, TRUE, TRUE, TRUE,…
## $ ACTION_TYPE    <chr> "Jump Shot", "Driving Layup Shot", "Jump Shot", "Jump S…
## $ SHOT_TYPE      <chr> "3PT Field Goal", "2PT Field Goal", "2PT Field Goal", "…
## $ BASIC_ZONE     <chr> "Above the Break 3", "Restricted Area", "Mid-Range", "M…
## $ ZONE_NAME      <chr> "Left Side Center", "Center", "Left Side Center", "Left…
## $ ZONE_ABB       <chr> "LC", "C", "LC", "L", "R", "C", "RC", "C", "C", "RC", "…
## $ ZONE_RANGE     <chr> "24+ ft.", "Less Than 8 ft.", "16-24 ft.", "16-24 ft.",…
## $ LOC_X          <dbl> 20.0, 0.0, 13.3, 16.4, -15.8, 0.0, -15.8, -1.5, -1.0, -…
## $ LOC_Y          <dbl> 21.35, 5.25, 24.45, 13.95, 7.85, 5.25, 23.15, 29.95, 5.…
## $ SHOT_DISTANCE  <int> 25, 0, 23, 18, 16, 0, 23, 24, 1, 18, 9, 24, 0, 3, 24, 1…
## $ QUARTER        <int> 6, 6, 6, 6, 6, 6, 6, 6, 4, 6, 6, 4, 6, 4, 4, 6, 4, 4, 6…
## $ MINS_LEFT      <int> 0, 0, 0, 0, 0, 1, 1, 1, 0, 2, 2, 0, 3, 0, 0, 3, 0, 1, 4…
## $ SECS_LEFT      <int> 0, 2, 9, 31, 55, 12, 25, 42, 13, 27, 52, 15, 31, 21, 38…
```

Check for missing values

```
colSums(is.na(shots))
```

```
##       SEASON_1       SEASON_2        TEAM_ID      TEAM_NAME      PLAYER_ID 
##              0              0              0              0              0 
##    PLAYER_NAME POSITION_GROUP       POSITION      GAME_DATE        GAME_ID 
##              0           7930           7930              0              0 
##      HOME_TEAM      AWAY_TEAM     EVENT_TYPE      SHOT_MADE    ACTION_TYPE 
##              0              0              0              0              0 
##      SHOT_TYPE     BASIC_ZONE      ZONE_NAME       ZONE_ABB     ZONE_RANGE 
##              0              0              0              0              0 
##          LOC_X          LOC_Y  SHOT_DISTANCE        QUARTER      MINS_LEFT 
##              0              0              0              0              0 
##      SECS_LEFT 
##              0
```

Position Group and Position have 7930 missing values. Because this is such a small fraction of the total data, I think it's best if these observations are deleted that way we can still attempt to use Position as a predictor.

```
shots = shots %>%
  filter(!is.na(POSITION))
```

Check again to see if any variables have missing values

```
colSums(is.na(shots))
```

```
##       SEASON_1       SEASON_2        TEAM_ID      TEAM_NAME      PLAYER_ID 
##              0              0              0              0              0 
##    PLAYER_NAME POSITION_GROUP       POSITION      GAME_DATE        GAME_ID 
##              0              0              0              0              0 
##      HOME_TEAM      AWAY_TEAM     EVENT_TYPE      SHOT_MADE    ACTION_TYPE 
##              0              0              0              0              0 
##      SHOT_TYPE     BASIC_ZONE      ZONE_NAME       ZONE_ABB     ZONE_RANGE 
##              0              0              0              0              0 
##          LOC_X          LOC_Y  SHOT_DISTANCE        QUARTER      MINS_LEFT 
##              0              0              0              0              0 
##      SECS_LEFT 
##              0
```

Check Team Names and Team ID

I know some team names and cities have change over this time period, so checking to see how the team ID's compare is necessary.

```
team_id_changes <- shots %>%
  group_by(TEAM_ID) %>%
  summarize(unique_names = paste(unique(TEAM_NAME), collapse = ", "),
            name_count = n_distinct(TEAM_NAME)) %>%
  filter(name_count > 1) # Only keep team_ids with multiple names

print(team_id_changes)
```

```
## # A tibble: 5 × 3
##      TEAM_ID unique_names                                                  name_count
##        <int> <chr>                                                              <int>
## 1 1610612740 New Orleans Hornets, New Orleans/Oklahoma City Hornets,…               3
## 2 1610612746 Los Angeles Clippers, LA Clippers                                      2
## 3 1610612751 New Jersey Nets, Brooklyn Nets                                         2
## 4 1610612760 Seattle SuperSonics, Oklahoma City Thunder                             2
## 5 1610612766 Charlotte Bobcats, Charlotte Hornets                                   2
```

As the results show, some teams have gone through name and city changes but they remain with the same team ID. For this reason I will be using team ID as the unique identifier for teams when comparing shots across time periods.

Change all values of LA Clippers in the team_name variable to Los Angeles Clippers for continuity

```
shots = shots %>%
  mutate(TEAM_NAME = case_when(
    TEAM_NAME == "LA Clippers" ~ "Los Angeles Clippers",
    TRUE ~ TEAM_NAME
  ))
```

Rename Columns for Clarity

```
shots = dplyr::rename(shots,  SHOT_DISTANCE_FT = SHOT_DISTANCE)
```

Remove redundant columns

```
shots = shots %>% select(-POSITION, -EVENT_TYPE, -ZONE_ABB)
```

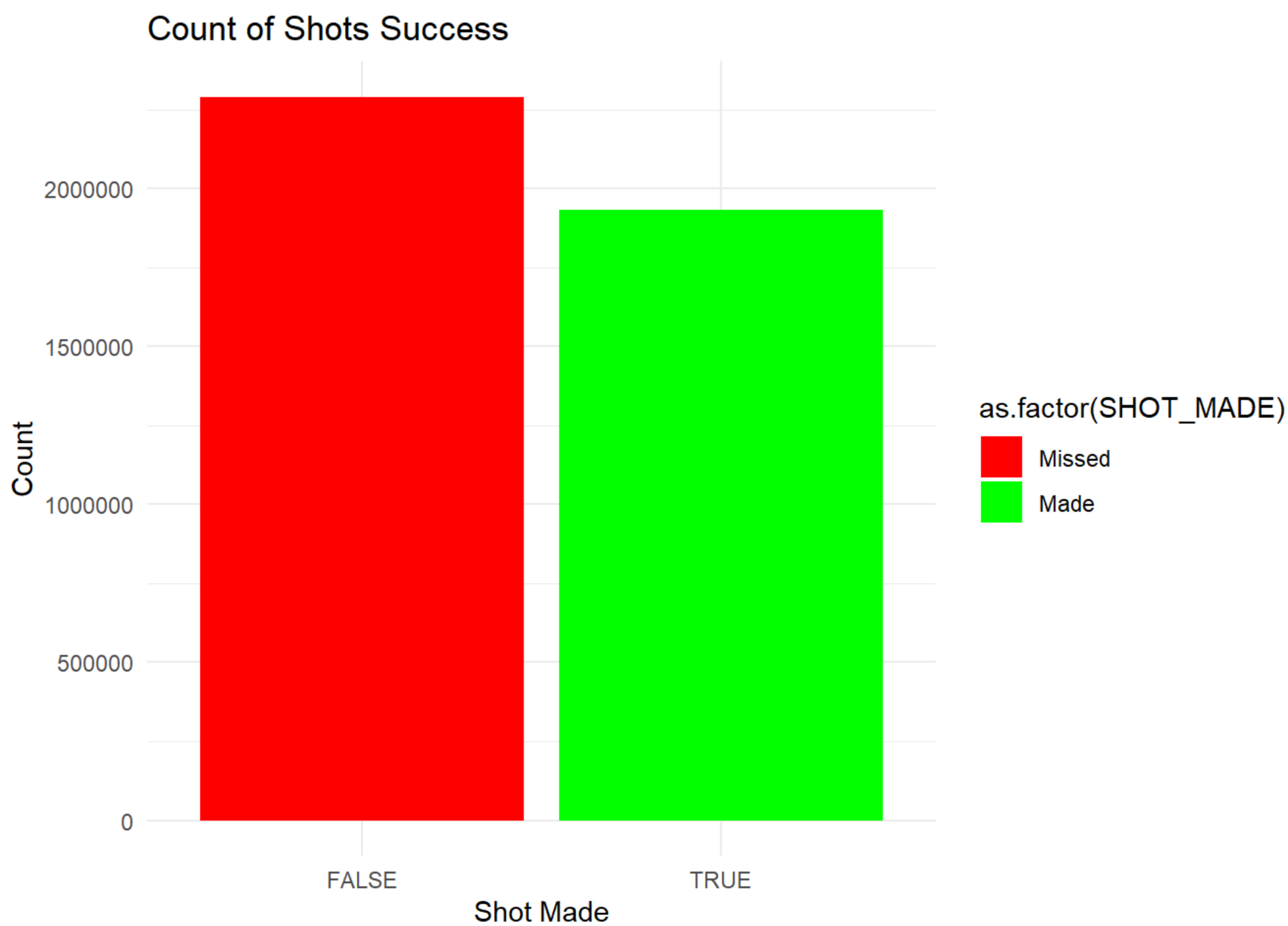
Conduct Exploratory Data Analysis (EDA)

Visualizing Shot Success Rate

Calculate overall shot success rate

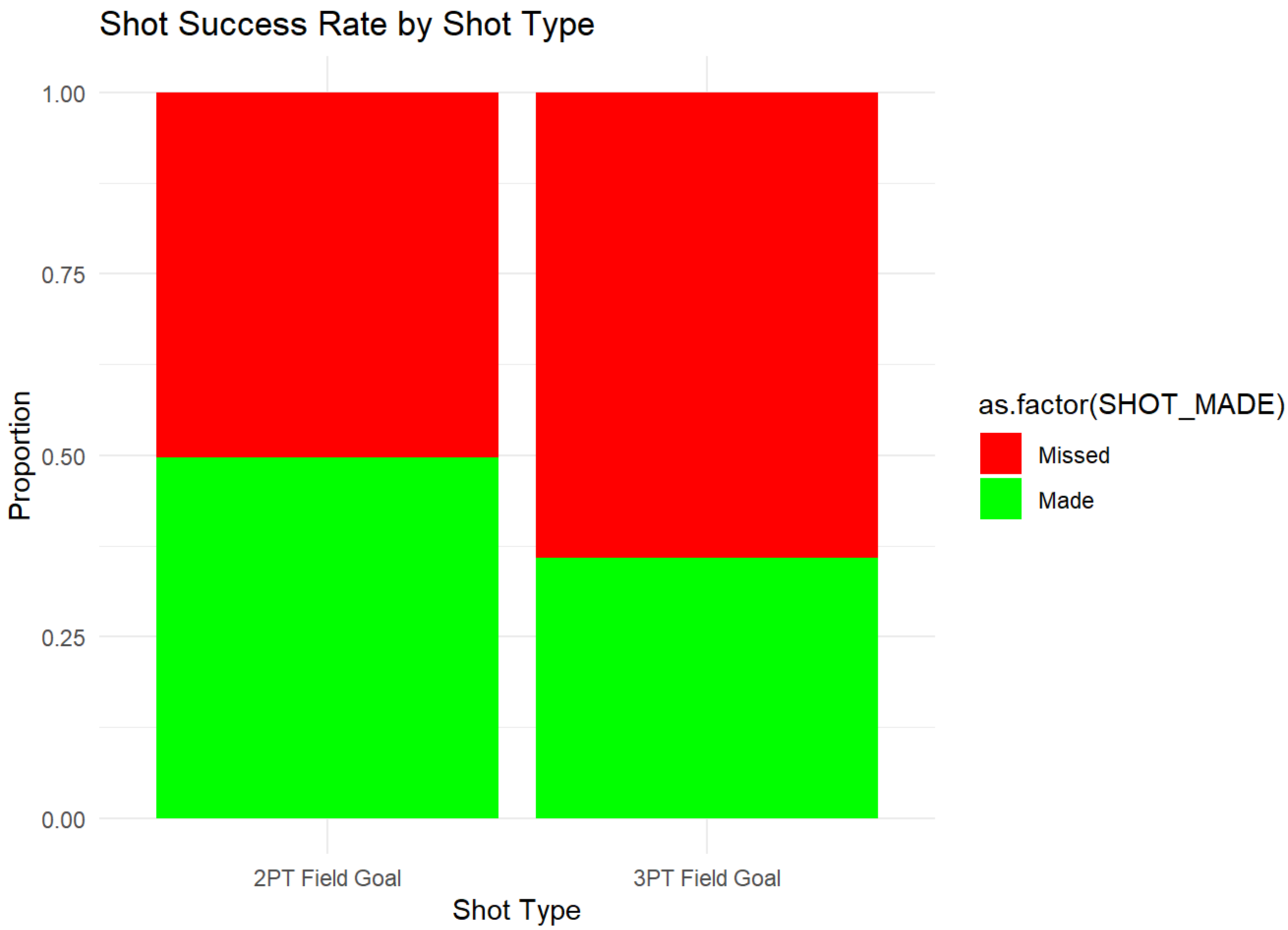
```
# Calculate overall shot success rate
shot_success = shots %>%
  group_by(SHOT_MADE) %>%
  summarise(count = n())

# Bar plot of shot success
ggplot(shot_success, aes(x = as.factor(SHOT_MADE), y = count, fill = as.factor(SHOT_MADE))) +
  geom_bar(stat = "identity") +
  labs(title = "Count of Shots Success", x = "Shot Made", y = "Count") +
  scale_fill_manual(values = c("red", "green"), labels = c("Missed", "Made")) +
  theme_minimal()
```



Shot Success by Shot Type (2pt vs 3pt)

```
ggplot(shots, aes(x = SHOT_TYPE, fill = as.factor(SHOT_MADE))) +
  geom_bar(position = "fill") +
  labs(title = "Shot Success Rate by Shot Type", x = "Shot Type", y = "Proportion") +
  scale_fill_manual(values = c("red", "green"), labels = c("Missed", "Made")) +
  theme_minimal()
```



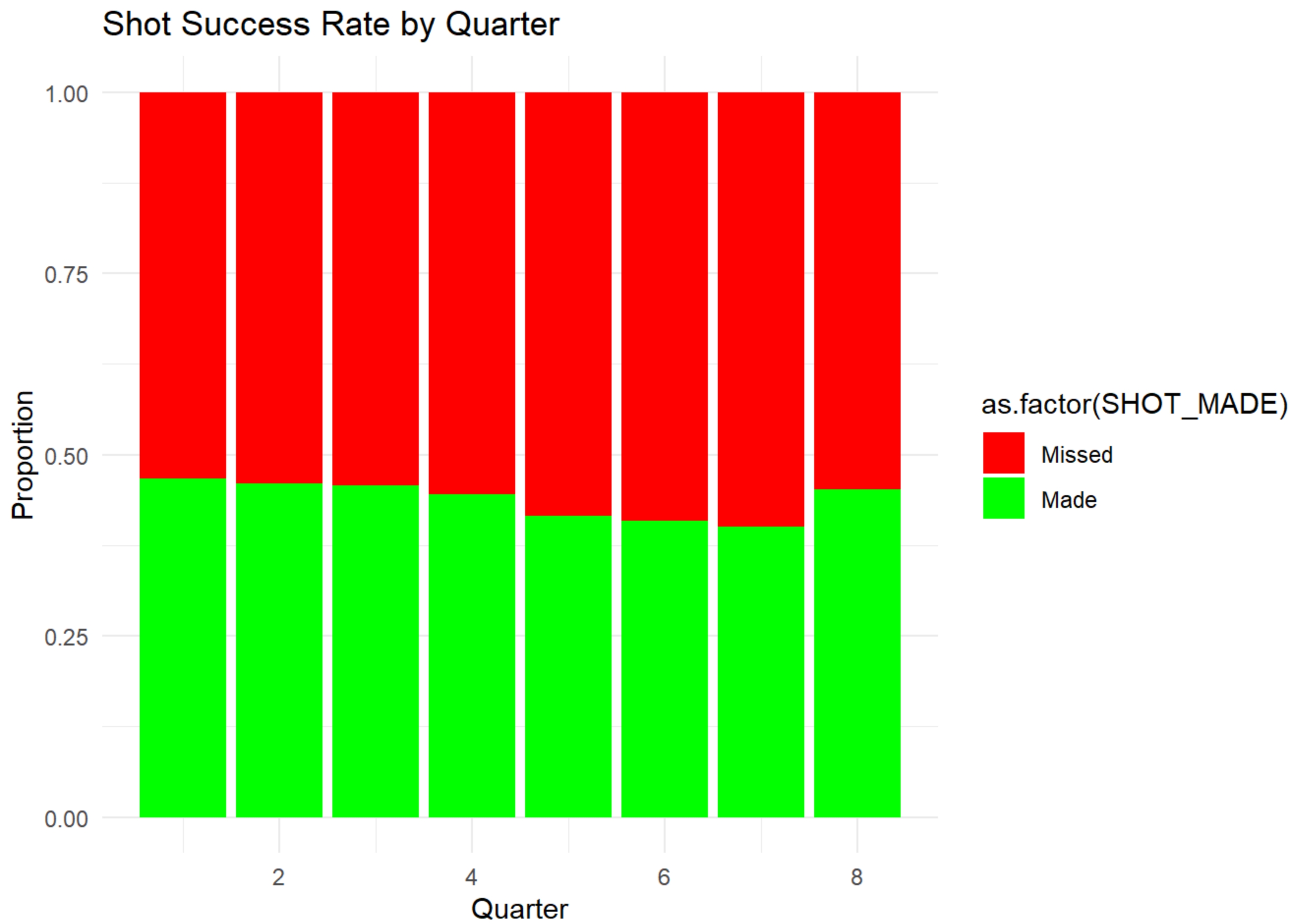
Shot Success by Distance

```
# Shot Success by distance
ggplot(shots, aes(x = SHOT_DISTANCE_FT, fill = SHOT_MADE)) +
  geom_histogram(binwidth = 2, position = "dodge") +
  labs(title = "Shot Success by Distance", x = "Shot Distance (ft)", y = "Count") +
  scale_fill_manual(values = c("red", "green"))
```

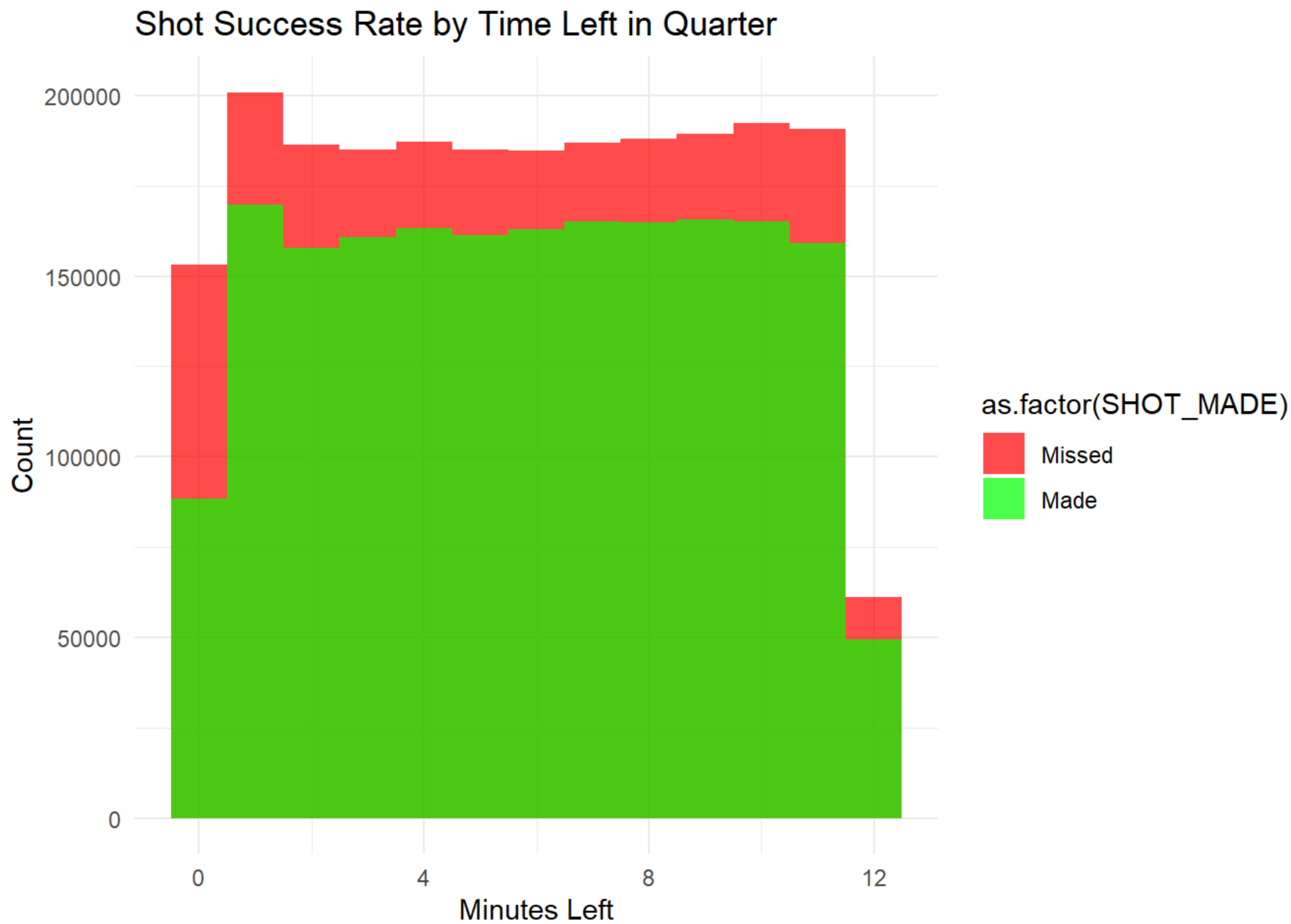
Identifying Key Patterns

Shot Success by Game Time (Quarter and Time Left)

```
ggplot(shots, aes(x = QUARTER, fill = as.factor(SHOT_MADE))) +
  geom_bar(position = "fill") +
  labs(title = "Shot Success Rate by Quarter", x = "Quarter", y = "Proportion") +
  scale_fill_manual(values = c("red", "green"), labels = c("Missed", "Made")) +
  theme_minimal()
```




```
ggplot(shots, aes(x = ((MINS_LEFT * 60) + SECS_LEFT)/60, fill = as.factor(SHOT_MADE))) +
  geom_histogram(binwidth = 1, position = "identity", alpha = 0.7) +
  labs(title = "Shot Success Rate by Time Left in Quarter", x = "Minutes Left", y = "Count") +
  scale_fill_manual(values = c("red", "green"), labels = c("Missed", "Made")) +
  theme_minimal()
```



Preliminary Feature Importance

Convert Categorical Variables to Factors

```
shots = shots %>%
  mutate(across(where(is.character), as.factor))
```

Convert Team_ID to Factor

```
shots$TEAM_ID = as.factor(shots$TEAM_ID)
```

Inspect Variables again

```
glimpse(shots)
```

```
## Rows: 4,223,332
## Columns: 23
## $ SEASON_1         <int> 2004, 2004, 2004, 2004, 2004, 2004, 2004, 2004, 2004,…
## $ SEASON_2         <fct> 2003-04, 2003-04, 2003-04, 2003-04, 2003-04, 2003-04,…
## $ TEAM_ID          <fct> 1610612747, 1610612757, 1610612747, 1610612757, 16106…
## $ TEAM_NAME        <fct> Los Angeles Lakers, Portland Trail Blazers, Los Angel…
## $ PLAYER_ID        <int> 977, 757, 977, 757, 757, 2567, 757, 977, 1544, 977, 2…
## $ PLAYER_NAME      <fct> Kobe Bryant, Damon Stoudamire, Kobe Bryant, Damon Sto…
## $ POSITION_GROUP   <fct> G, G, G, G, G, C, G, G, F, G, F, F, G, G, G, F, G, F,…
## $ GAME_DATE        <fct> 04-14-2004, 04-14-2004, 04-14-2004, 04-14-2004, 04-14…
## $ GAME_ID          <int> 20301187, 20301187, 20301187, 20301187, 20301187, 203…
## $ HOME_TEAM        <fct> POR, POR, POR, POR, POR, POR, POR, POR, BOS, POR, POR…
## $ AWAY_TEAM        <fct> LAL, LAL, LAL, LAL, LAL, LAL, LAL, LAL, ATL, LAL, LAL…
## $ SHOT_MADE        <lgl> TRUE, TRUE, FALSE, TRUE, FALSE, TRUE, TRUE, TRUE, TRU…
## $ ACTION_TYPE      <fct> Jump Shot, Driving Layup Shot, Jump Shot, Jump Shot, …
## $ SHOT_TYPE        <fct> 3PT Field Goal, 2PT Field Goal, 2PT Field Goal, 2PT F…
## $ BASIC_ZONE       <fct> Above the Break 3, Restricted Area, Mid-Range, Mid-Ra…
## $ ZONE_NAME        <fct> Left Side Center, Center, Left Side Center, Left Side…
## $ ZONE_RANGE       <fct> 24+ ft., Less Than 8 ft., 16-24 ft., 16-24 ft., 16-24…
## $ LOC_X            <dbl> 20.0, 0.0, 13.3, 16.4, -15.8, 0.0, -15.8, -1.5, -1.0,…
## $ LOC_Y            <dbl> 21.35, 5.25, 24.45, 13.95, 7.85, 5.25, 23.15, 29.95, …
## $ SHOT_DISTANCE_FT <int> 25, 0, 23, 18, 16, 0, 23, 24, 1, 18, 9, 24, 0, 3, 24,…
## $ QUARTER          <int> 6, 6, 6, 6, 6, 6, 6, 6, 4, 6, 6, 4, 6, 4, 4, 6, 4, 4,…
## $ MINS_LEFT        <int> 0, 0, 0, 0, 0, 1, 1, 1, 0, 2, 2, 0, 3, 0, 0, 3, 0, 1,…
## $ SECS_LEFT        <int> 0, 2, 9, 31, 55, 12, 25, 42, 13, 27, 52, 15, 31, 21, …
```

Engineer Features

Engineer a Home vs. Away Indicator

```
# Check to see all the team names and abbreviations
sort(unique(shots$HOME_TEAM))
```

```
##  [1] ATL BKN BOS CHA CHI CLE DAL DEN DET GSW HOU IND LAC LAL MEM MIA MIL MIN NJN
## [20] NOH NOK NOP NYK OKC ORL PHI PHX POR SAC SAS SEA TOR UTA WAS
## 34 Levels: ATL BKN BOS CHA CHI CLE DAL DEN DET GSW HOU IND LAC LAL MEM ... WAS
```

```
sort(unique(shots$TEAM_NAME))
```

```
##  [1] Atlanta Hawks                       Boston Celtics
##  [3] Brooklyn Nets                       Charlotte Bobcats
##  [5] Charlotte Hornets                   Chicago Bulls
##  [7] Cleveland Cavaliers                 Dallas Mavericks
##  [9] Denver Nuggets                      Detroit Pistons
## [11] Golden State Warriors               Houston Rockets
## [13] Indiana Pacers                      Los Angeles Clippers
## [15] Los Angeles Lakers                  Memphis Grizzlies
## [17] Miami Heat                          Milwaukee Bucks
## [19] Minnesota Timberwolves              New Jersey Nets
## [21] New Orleans Hornets                 New Orleans Pelicans
## [23] New Orleans/Oklahoma City Hornets New York Knicks
## [25] Oklahoma City Thunder               Orlando Magic
## [27] Philadelphia 76ers                  Phoenix Suns
## [29] Portland Trail Blazers              Sacramento Kings
## [31] San Antonio Spurs                   Seattle SuperSonics
## [33] Toronto Raptors                     Utah Jazz
## [35] Washington Wizards
## 35 Levels: Atlanta Hawks Boston Celtics Brooklyn Nets ... Washington Wizards
```

```
# Map all team names to their respective abbreviation
team_mapping = data.frame(
  team_name = c("Atlanta Hawks", "Boston Celtics", "Brooklyn Nets", "Charlotte Bobcats", "Charlotte Hornets", "Chicago Bulls", "Cleveland Cavaliers", "Dallas Mavericks", "Denver Nuggets", "Detroit Pistons", "Golden State Warriors", "Houston Rockets", "Indiana Pacers", "Los Angeles Clippers", "Los Angeles Lakers", "Memphis Grizzlies", "Miami Heat", "Milwaukee Bucks", "Minnesota Timberwolves", "New Jersey Nets", "New Orleans Hornets", "New Orleans Pelicans", "New Orleans/Oklahoma City Hornets", "New York Knicks", "Oklahoma City Thunder", "Orlando Magic", "Philadelphia 76ers", "Phoenix Suns", "Portland Trail Blazers", "Sacramento Kings", "San Antonio Spurs", "Seattle SuperSonics", "Toronto Raptors", "Utah Jazz", "Washington Wizards"),
  team_abbreviation = c("ATL", "BOS", "BKN", "CHA", "CHA", "CHI", "CLE", "DAL", "DEN", "DET", "GSW", "HOU", "IND", "LAC", "LAL", "MEM", "MIA", "MIL", "MIN", "NJN", "NOH", "NOP", "NOK", "NYK", "OKC", "ORL", "PHI", "PHX", "POR", "SAC", "SAS", "SEA", "TOR", "UTA", "WAS")
)

# Join the mapping to shots based on the full team name
shots = shots %>%
  left_join(team_mapping, by = c("TEAM_NAME" = "team_name"))

# Create the 'is_home' feature based on home team abbreviation
shots = shots %>%
  mutate(Is_Home = ifelse(team_abbreviation == HOME_TEAM, 1, 0))
```

Engineer Feature for Time Elapsed

This variable will help provide a better understanding how much time in the game has elapsed when the shot was taken

```
# Calculate the total amount of seconds that have elapsed at the time of the shot

shots$Game_Sec_Elapsed <- ifelse(
  shots$QUARTER <= 4,
  ((shots$QUARTER - 1) * 12 * 60) + (12 * 60 - (shots$MINS_LEFT * 60 + shots$SECS_LEFT)),
  (4 * 12 * 60) + ((shots$QUARTER - 5) * 5 * 60) + (5 * 60 - (shots$MINS_LEFT * 60 + shots$SECS_
LEFT))
)
```

Engineer Shot_Made to also have a numeric representation

```
shots$SHOT_MADE_Numeric <- as.numeric(shots$SHOT_MADE)
```

Investigate how variables are related

Look at Numeric Variables

```
library(purrr)
```

```
## Warning: package 'purrr' was built under R version 4.4.2
```

```
##
## Attaching package: 'purrr'
```

```
## The following object is masked from 'package:data.table':
##
##     transpose
```

```
## The following object is masked from 'package:caret':
##
##     lift
```

```
data <- shots
target_var <- "SHOT_MADE_Numeric"

# Ensure the target variable is numeric
data[[target_var]] <- as.numeric(data[[target_var]])

# Get all numeric predictor variables (excluding target variable)
predictor_vars <- names(data) %>%
  setdiff(target_var) %>%
  keep(~ is.numeric(data[[.x]]))  # Keep only numeric predictors

# Run cor.test() for each predictor
cor_results <- predictor_vars %>%
  map_df(~ {
    # Ensure both the predictor and target are numeric
    predictor <- as.numeric(data[[.x]])
    target <- as.numeric(data[[target_var]])

    # Perform the correlation test
    test <- cor.test(predictor, target, use = "complete.obs")

    # Return results in a tibble
    tibble(Variable = .x, Correlation = test$estimate, P_value = test$p.value)
  })

# Print results
print(cor_results)
```

```
## # A tibble: 11 × 3
##    Variable         Correlation   P_value
##    <chr>                  <dbl>     <dbl>
##  1 SEASON_1             0.0118  5.73e-131
##  2 PLAYER_ID            0.00517 2.24e- 26
##  3 GAME_ID              0.0119  4.61e-131
##  4 LOC_X               -0.00360 1.49e- 13
##  5 LOC_Y               -0.141   0
##  6 SHOT_DISTANCE_FT    -0.200   0
##  7 QUARTER             -0.0161  2.20e-241
##  8 MINS_LEFT            0.0164  8.93e-249
##  9 SECS_LEFT            0.0156  8.05e-227
## 10 Is_Home              0.0105  2.18e-103
## 11 Game_Sec_Elapsed    -0.0197  0
```

Most variables have a weak correlation with whether a shot is made or not, but the p-values are extremely small which indicate statistical significance. SHOT_DISTANCE and LOC_Y stand out as they have slightly stronger correlations than other variables, meaning their relationship with the target variable is more meaningful.

Look at Categorical Variables

```
target_var2 = "SHOT_MADE"
categorical_vars = names(shots)[sapply(shots, is.factor)]

categorical_vars = setdiff(categorical_vars, target_var2)

chi_results = categorical_vars %>%
  map_df(~ {
    test = chisq.test(table(shots[[.x]], shots[[target_var2]]))
    tibble(Variable = .x, Chi_Square = test$statistic, P_Value = test$p.value)
  })
```

```
## Warning in chisq.test(table(shots[[.x]], shots[[target_var2]])): Chi-squared
## approximation may be incorrect
## Warning in chisq.test(table(shots[[.x]], shots[[target_var2]])): Chi-squared
## approximation may be incorrect
```

```
print(chi_results)
```

```
## # A tibble: 12 × 3
##    Variable       Chi_Square   P_Value
##    <chr>               <dbl>     <dbl>
##  1 SEASON_2            1162. 1.04e-233
##  2 TEAM_ID              675. 1.75e-123
##  3 PLAYER_NAME        39838. 0
##  4 POSITION_GROUP     15780. 0
##  5 GAME_DATE           4723. 2.07e- 50
##  6 HOME_TEAM            389. 2.24e- 62
##  7 AWAY_TEAM            238. 6.44e- 33
##  8 ACTION_TYPE       374271. 0
##  9 SHOT_TYPE          67056. 0
## 10 BASIC_ZONE        208915. 0
## 11 ZONE_NAME          96072. 0
## 12 ZONE_RANGE        152448. 0
```

All of the variables have small p-values which means they are all significantly associated with shot success. Action_type, shot_type, basic_zone and zone_name have the strongest relationships, which suggest that where and how a player shoots significantly affects success. Player_name and Position_group also have a strong relationship which makes sense as different players have different shooting abilities and tendencies. Variables such as team_name, season_2, home_team and away_team have weaker relationships with the target variables.

Select Relevant Variables for Model

```
df = shots %>% select(SEASON_2, TEAM_ID, PLAYER_NAME, POSITION_GROUP, ACTION_TYPE, BASIC_ZONE, S
HOT_DISTANCE_FT, Is_Home, Game_Sec_Elapsed, SHOT_MADE_Numeric)
```

Data Preprocessing

```
# Name the data for ease of use
data = df

# Define target variable
target_var = "SHOT_MADE_Numeric"

# One-hot encode POSITION_GROUP and BASIC_ZONE
recipe_prep <- recipe(SHOT_MADE_Numeric ~ ., data = data) %>%
  step_dummy(all_of(c("POSITION_GROUP", "BASIC_ZONE")), one_hot = TRUE) %>%
  prep(training = data)

data <- bake(recipe_prep, new_data = data)

# K-fold target encoding function
kfold_target_encode <- function(data, cat_vars, target_var, k = 5) {
  set.seed(123)
  folds <- createFolds(data[[target_var]], k = k, list = TRUE)

  for (var in cat_vars) {
    encoded_vals <- numeric(nrow(data))

    for (i in seq_along(folds)) {
      train_idx <- unlist(folds[-i])
      valid_idx <- folds[[i]]

      means <- data[train_idx, ] %>%
        group_by(across(all_of(var))) %>%
        summarise(mean_target = mean(.data[[target_var]], na.rm = TRUE), .groups = "drop")

      encoded_vals[valid_idx] <- data[valid_idx, ] %>%
        left_join(means, by = var) %>%
        pull(mean_target)
    }

    data[[var]] <- ifelse(is.na(encoded_vals), mean(data[[target_var]], na.rm = TRUE), encoded_v
als)
  }
  return(data)
}

# Apply K-fold target encoding
categorical_vars <- c("SEASON_2", "TEAM_ID", "PLAYER_NAME", "ACTION_TYPE")
data <- kfold_target_encode(data, categorical_vars, target_var)
```

Split Data into Training and Testing Sets

```
set.seed(123)
trainIndex <- createDataPartition(data$SHOT_MADE_Numeric, p = 0.8, list = FALSE)
train_data <- data[trainIndex, ]
test_data <- data[-trainIndex, ]
```

Logisitc Regression Model

```
logistic_model <- glm(SHOT_MADE_Numeric ~ ., data = train_data, family = binomial)
logistic_preds <- predict(logistic_model, newdata = test_data, type = "response")
logistic_preds_class <- ifelse(logistic_preds > 0.5, 1, 0)
confusionMatrix(as.factor(logistic_preds_class), as.factor(test_data$SHOT_MADE_Numeric))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0 388519 245538
##          1  69611 140998
##
##                Accuracy : 0.6269
##                  95% CI : (0.6259, 0.6279)
##     No Information Rate : 0.5424
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2207
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8481
##             Specificity : 0.3648
##          Pos Pred Value : 0.6128
##          Neg Pred Value : 0.6695
##              Prevalence : 0.5424
##          Detection Rate : 0.4600
##    Detection Prevalence : 0.7507
##       Balanced Accuracy : 0.6064
##
##        'Positive' Class : 0
##
```

```
summary(logistic_model)
```

```
##
## Call:
## glm(formula = SHOT_MADE_Numeric ~ ., family = binomial, data = train_data)
##
## Coefficients: (2 not defined because of singularities)
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -8.055e-01  1.067e-01  -7.552 4.28e-14 ***
## SEASON_2                     -6.582e+00  1.456e-01 -45.215  < 2e-16 ***
## TEAM_ID                       2.321e+00  1.824e-01  12.725  < 2e-16 ***
## PLAYER_NAME                   1.023e+00  3.131e-02  32.682  < 2e-16 ***
## ACTION_TYPE                   4.796e+00  1.320e-02 363.361  < 2e-16 ***
## SHOT_DISTANCE_FT              9.293e-03  5.134e-04  18.101  < 2e-16 ***
## Is_Home                       1.998e-02  2.295e-03   8.706  < 2e-16 ***
## Game_Sec_Elapsed             -3.963e-05  1.367e-06 -28.986  < 2e-16 ***
## POSITION_GROUP_C              3.260e-02  4.250e-03   7.671 1.71e-14 ***
## POSITION_GROUP_F              1.615e-02  2.634e-03   6.133 8.62e-10 ***
## POSITION_GROUP_G                     NA         NA      NA       NA
## BASIC_ZONE_Above.the.Break.3 -2.309e-01  6.640e-03 -34.770  < 2e-16 ***
## BASIC_ZONE_Backcourt         -3.666e+00  7.861e-02 -46.636  < 2e-16 ***
## BASIC_ZONE_In.The.Paint..Non.RA. -3.126e-01  1.034e-02 -30.236  < 2e-16 ***
## BASIC_ZONE_Left.Corner.3     -7.515e-03  8.331e-03  -0.902 0.367026
## BASIC_ZONE_Mid.Range         -9.608e-02  7.224e-03 -13.300  < 2e-16 ***
## BASIC_ZONE_Restricted.Area   -4.634e-02  1.283e-02  -3.612 0.000304 ***
## BASIC_ZONE_Right.Corner.3            NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4659607  on 3378665  degrees of freedom
## Residual deviance: 4324385  on 3378650  degrees of freedom
## AIC: 4324417
##
## Number of Fisher Scoring iterations: 5
```

```
exp(coef(logistic_model))
```

```
##                    (Intercept)                         SEASON_2
##                   4.468816e-01                     1.384723e-03
##                        TEAM_ID                      PLAYER_NAME
##                   1.018225e+01                     2.782512e+00
##                    ACTION_TYPE                 SHOT_DISTANCE_FT
##                   1.210748e+02                     1.009337e+00
##                        Is_Home                 Game_Sec_Elapsed
##                   1.020178e+00                     9.999604e-01
##               POSITION_GROUP_C                 POSITION_GROUP_F
##                   1.033136e+00                     1.016284e+00
##               POSITION_GROUP_G     BASIC_ZONE_Above.the.Break.3
##                             NA                     7.938340e-01
##           BASIC_ZONE_Backcourt BASIC_ZONE_In.The.Paint..Non.RA.
##                   2.558337e-02                     7.315542e-01
##         BASIC_ZONE_Left.Corner.3             BASIC_ZONE_Mid.Range
##                   9.925134e-01                     9.083893e-01
##       BASIC_ZONE_Restricted.Area        BASIC_ZONE_Right.Corner.3
##                   9.547159e-01                               NA
```

Gradient Boosting Model

```
xgb_train <- xgb.DMatrix(data = as.matrix(train_data %>% select(-SHOT_MADE_Numeric)), label = tr
ain_data$SHOT_MADE_Numeric)
xgb_test <- xgb.DMatrix(data = as.matrix(test_data %>% select(-SHOT_MADE_Numeric)), label = test
_data$SHOT_MADE_Numeric)

xgb_model <- xgboost(data = xgb_train, max_depth = 6, eta = 0.1, nrounds = 100, objective = "bin
ary:logistic")
```

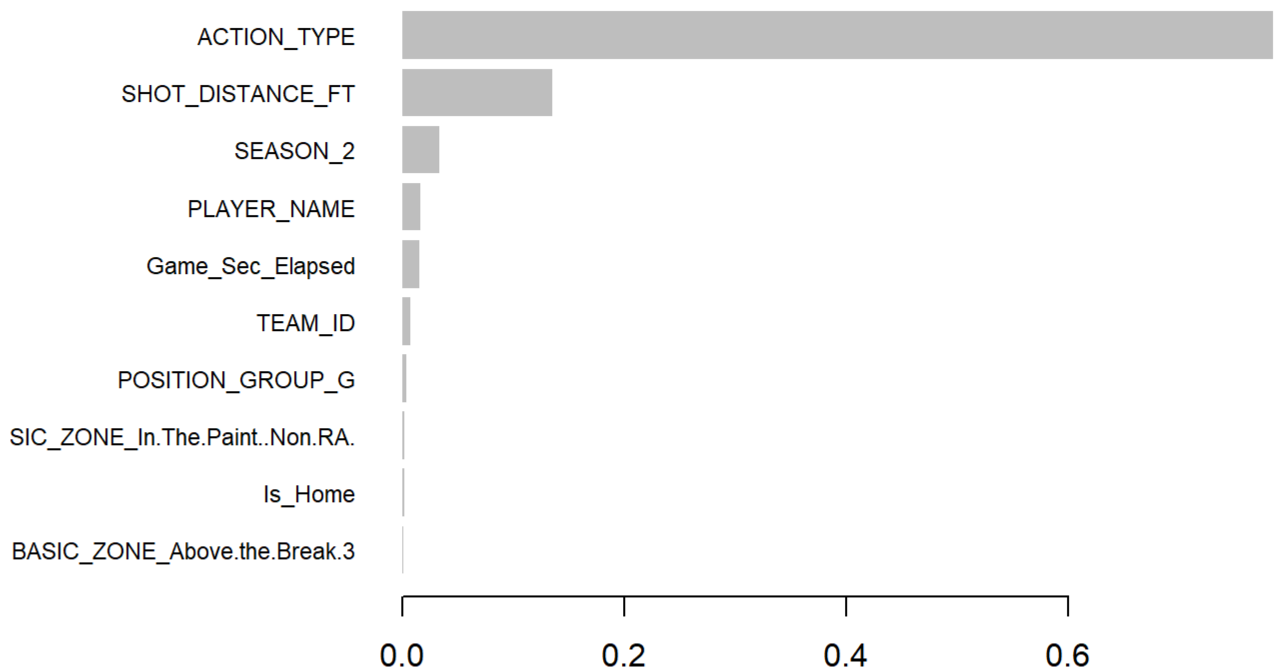
```
## [1]	train-logloss:0.682502
## [2]	train-logloss:0.673830
## [3]	train-logloss:0.666728
## [4]	train-logloss:0.660881
## [5]	train-logloss:0.656018
## [6]	train-logloss:0.651953
## [7]	train-logloss:0.648590
## [8]	train-logloss:0.645702
## [9]	train-logloss:0.643261
## [10]	train-logloss:0.641258
## [11]	train-logloss:0.639576
## [12]	train-logloss:0.638129
## [13]	train-logloss:0.636935
## [14]	train-logloss:0.635861
## [15]	train-logloss:0.635010
## [16]	train-logloss:0.634304
## [17]	train-logloss:0.633644
## [18]	train-logloss:0.633090
## [19]	train-logloss:0.632617
## [20]	train-logloss:0.632196
## [21]	train-logloss:0.631853
## [22]	train-logloss:0.631579
## [23]	train-logloss:0.631332
## [24]	train-logloss:0.631113
## [25]	train-logloss:0.630924
## [26]	train-logloss:0.630750
## [27]	train-logloss:0.630605
## [28]	train-logloss:0.630458
## [29]	train-logloss:0.630347
## [30]	train-logloss:0.630224
## [31]	train-logloss:0.630102
## [32]	train-logloss:0.630000
## [33]	train-logloss:0.629907
## [34]	train-logloss:0.629827
## [35]	train-logloss:0.629746
## [36]	train-logloss:0.629678
## [37]	train-logloss:0.629628
## [38]	train-logloss:0.629581
## [39]	train-logloss:0.629493
## [40]	train-logloss:0.629414
## [41]	train-logloss:0.629365
## [42]	train-logloss:0.629315
## [43]	train-logloss:0.629256
## [44]	train-logloss:0.629206
## [45]	train-logloss:0.629155
## [46]	train-logloss:0.629125
## [47]	train-logloss:0.629074
## [48]	train-logloss:0.629032
## [49]	train-logloss:0.628960
## [50]	train-logloss:0.628897
## [51]	train-logloss:0.628831
## [52]	train-logloss:0.628790
```

```
## [53] train-logloss:0.628684
## [54] train-logloss:0.628632
## [55] train-logloss:0.628596
## [56] train-logloss:0.628525
## [57] train-logloss:0.628479
## [58] train-logloss:0.628400
## [59] train-logloss:0.628343
## [60] train-logloss:0.628316
## [61] train-logloss:0.628273
## [62] train-logloss:0.628228
## [63] train-logloss:0.628161
## [64] train-logloss:0.628129
## [65] train-logloss:0.628111
## [66] train-logloss:0.628058
## [67] train-logloss:0.628022
## [68] train-logloss:0.627982
## [69] train-logloss:0.627935
## [70] train-logloss:0.627911
## [71] train-logloss:0.627868
## [72] train-logloss:0.627835
## [73] train-logloss:0.627810
## [74] train-logloss:0.627785
## [75] train-logloss:0.627744
## [76] train-logloss:0.627731
## [77] train-logloss:0.627707
## [78] train-logloss:0.627682
## [79] train-logloss:0.627663
## [80] train-logloss:0.627620
## [81] train-logloss:0.627594
## [82] train-logloss:0.627563
## [83] train-logloss:0.627547
## [84] train-logloss:0.627530
## [85] train-logloss:0.627515
## [86] train-logloss:0.627476
## [87] train-logloss:0.627453
## [88] train-logloss:0.627425
## [89] train-logloss:0.627414
## [90] train-logloss:0.627398
## [91] train-logloss:0.627373
## [92] train-logloss:0.627350
## [93] train-logloss:0.627319
## [94] train-logloss:0.627256
## [95] train-logloss:0.627232
## [96] train-logloss:0.627218
## [97] train-logloss:0.627176
## [98] train-logloss:0.627138
## [99] train-logloss:0.627132
## [100]    train-logloss:0.627110
```

```
xgb_preds <- predict(xgb_model, xgb_test)
xgb_preds_class <- ifelse(xgb_preds > 0.5, 1, 0)
confusionMatrix(as.factor(xgb_preds_class), as.factor(test_data$SHOT_MADE_Numeric))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0 381325 225605
##          1  76805 160931
##
##                Accuracy : 0.642
##                  95% CI : (0.641, 0.643)
##     No Information Rate : 0.5424
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2564
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8324
##             Specificity : 0.4163
##          Pos Pred Value : 0.6283
##          Neg Pred Value : 0.6769
##              Prevalence : 0.5424
##          Detection Rate : 0.4515
##    Detection Prevalence : 0.7185
##       Balanced Accuracy : 0.6243
##
##        'Positive' Class : 0
##
```

```
importance = xgb.importance(model = xgb_model)
xgb.plot.importance(importance, top_n = 10)
```


ACTION_TYPE
SHOT_DISTANCE_FT
SEASON_2
PLAYER_NAME
Game_Sec_Elapsed
TEAM_ID
POSITION_GROUP_G
SIC_ZONE_In.The.Paint..Non.RA.
Is_Home
BASIC_ZONE_Above.the.Break.3
0.0
0.2
0.4
0.6