Evaluating Player Performance:A Statistical Journey through Dota2

Introduction:

Defense of the Ancients (DotA) is an online multiplayer strategy game that engages players in a five-versus-five battle. Originally a custom game mode from Blizzard Entertainment's Warcraft 3, released in 2003, DotA has paved the way for the "Multiplayer Online Battle Arena" genre. The game is complex, featuring numerous variables and intricacies. The primary goal for each team is to destroy the opponent's ancient. This game is managed by a sophisticated ranking and matchmaking system and has garnered a player base of millions worldwide, including myself. Having played Dota2 since 2010, with a hiatus from 2016 to late 2019, I have found the game to be both competitive and potentially addictive. As of the latest update, Dota 2 features 124 unique heroes.

Problem Statement/Topic:

In this paper, I intend to analyze my own behavior and performance within DotA, exploring how various factors may contribute to the outcomes of my games. Given the team-based nature of DotA, individual performance metrics may not fully encapsulate the likelihood of winning a match. Other factors, such as matchmaking rating, item choices, and economic aspects of the game, also play critical roles, though some of these elements fall outside the scope of my current objective. By examining the relationship between match outcomes and various gameplay elements, this analysis aims to uncover patterns and relationships that may inform and validate my gaming intuition. My central research question focuses on identifying which aspects of my gameplay correlate with winning or losing matches.

The Data:

Since millions of players worldwide engage in Dota 2, and an understanding of player behavior and performance can enhance the gaming experience, inform game balance, and potentially offer strategies for improving team performance. This analysis not only holds personal significance but also contributes to the broader understanding of player dynamics in online multiplayer games. This project presents a data science challenge, requiring the collection, cleaning, and analysis of a large and complex dataset to extract meaningful insights. The primary dataset, Match_Data, comprises 495 rows and 357 columns, representing games played since 2019. However, the dataset's complexity is amplified by the presence of nested data frames within these columns, resulting in an extensive array of over seven thousand variables. Prior to analysis, it is crucial to define clear objectives to streamline the data cleaning and manipulation process, focusing on variables most likely to provide insights into gaming performance.

Research Questions:

1. How has my performance in Dota 2 evolved over time, and are there particular areas of improvement or decline?
2. Which heroes correlate with my best and worst performances, and what factors contribute to these patterns?
3. How does match duration influence my performance, and do I excel in shorter or longer games?
4. Does the Hero played impact my gameplay outcomes?
5. Is there a correlation between my performance and the frequency of my gameplay?
6. Does playing as Radiant or Dire influence match outcomes?
7. Are there identifiable player behaviors that correlate with a higher likelihood of winning?
8. What role do communication and teamwork play in securing a win?

Approach:

Data Collection:

Utilizing PyCharm and the OpenDota API, I have compiled match data and saved it as JSON files. This process involved multiple API calls and iterations over each match ID to retrieve detailed match data.

Data Cleaning and Preprocessing:

My initial focus will be on variable selection, aiming to isolate those most pertinent to my research questions. This process will include exploratory data analysis (EDA) to identify patterns or anomalies and may necessitate additional data retrieval from OpenDota.

Summary Statistics:

1. Descriptive Statistics: Calculate the mean, median, mode, minimum, maximum, range, quartiles, and standard deviation for continuous variables (e.g., Duration, Kills, Deaths, Assists, etc.). This will provide a sense of the central tendency, spread, and distribution of the data.
2. Frequency Counts: For categorical variables like Lane_role, Primary_attr, and Attack_type, produce frequency counts to understand the distribution of categories.
3. Correlation Analysis: Assess the correlation between continuous variables, especially between the performance metrics (e.g., Kills, Deaths, Assists, etc.) and the outcome variable (Win).

Data Visualization:

- Bar charts, scatter plots, and heat maps to represent performance distributions, correlations, and factor interactions Tables to summarize key statistics and findings.

- Histograms: Plot histograms for continuous variables to visualize their distribution and identify any skewness or outliers.
- Box Plots: Use box plots to identify outliers and understand the spread of the data for variables like Duration, Kills, Deaths, etc.
- Scatter Plots: Create scatter plots to visualize relationships between continuous variables, such as Kills vs. Win, Deaths vs. Win, or Gold_per_min vs. Win.
- Heat Map: Create a correlation heat map to visualize the strength and direction of relationships between continuous variables.

Model/Method Recommendation:

The primary model utilized in this analysis is logistic regression, chosen for its effectiveness in binary classification problems, such as predicting match outcomes (win or loss). Logistic regression is well-suited for this task due to its ability to handle multiple predictor variables and provide probabilities for the outcomes. By fitting a logistic regression model, I aim to identify significant predictors of winning a match, which include in-game metrics such as kills, deaths, assists, net worth, and various others.

Required Packages:

.dplyr, tidyr (for data manipulation and cleaning) .ggplot2 (for data visualization) .jsonlite (for loading JSON data) .Knitter (Facilitates customizable table creation in R Markdown) .Potentially additional packages as the analysis progresses

Detailed Variable Descriptions:

1. Match_id: A unique identifier generated by Steam for each match played.
2. Account_Id: Your unique Steam account identifier.

3. Duration: The total time the match lasted, measured in minutes.
4. Is_Radiant: A boolean variable indicating whether your team was the Radiant team (True) or the Dire team (False).
5. Win: A binary outcome of the match where 1 represents a win and 0 represents a loss.
6. Kills: The total number of enemy heroes you killed during the match.
7. Deaths: The number of times you were killed during the match, either by enemy heroes, units, neutral creeps, or through suicide.
8. Assists: The number of assists you made, helping your teammates secure kills.
9. KDA: The Kill-Death-Assist ratio calculated as (Kills+Assists)/Deaths
10. Last Hits: The total number of creeps you successfully delivered the final blow to, securing gold and experience.
11. Denies: The total number of allied creeps you killed, denying the enemy team gold and experience.
12. Net_worth: The total value of all items and gold possessed by your hero at the end of the game.
13. Total_gold: The total amount of gold earned during the game, including gold lost due to deaths and buybacks.
14. Gold_per_min: The average rate at which you earned gold throughout the match.
15. Total_xp: The total amount of experience earned during the match.
16. XP_per_min: The average rate at which you earned experience throughout the match.
17. Lane_efficiency: A measure from 0 to 1 of how efficiently you played your lane, based on how much gold and experience you out-earned your opponent in the same lane.
18. Lane_role: A categorical variable indicating the role you played, such as carry, mid lane, off lane, support, or hard support.
19. Hero_damage: The total amount of damage you dealt to enemy heroes during the match.
20. Abandon: A binary variable indicating whether any player abandoned the game (1 for yes, 0 for no).
21. Firstblood_claimed: A binary variable indicating whether you secured the first kill of the game (1 for yes, 0 for no)..
22. Localized_name: The name of the hero you played, to be later renamed to 'Hero'.
23. Primary_attr: The primary attribute of your hero, categorized as agility (agi), intelligence (int), or strength (str).
24. Attack_type: The basic attack type of your hero, categorized as melee or ranged.
25. Lost_gold: The total amount of gold lost in the game, primarily due to deaths and buybacks.

Step 1:Cleaning The Data

```
library(jsonlite)
library(dplyr)
library(purrr)
library(tidyr)
library(knitr)
library(caret)
library(car)
library(ggplot2)
```

I will start by exploring the variables, unnesting nested data frames when needed, merging data sets, dropping, and creating new columns. I intend to create a new data frame with limited variables I would like to work with. There is always a possibility of adding new and deleting existing variables during the analysis.

```
# Define the relative paths to the data files
data_dir <- "../data"

# Load the JSON files
Player_data <- fromJSON(file.path(data_dir, "matche_since_2019.json"))
matches_data <- fromJSON(file.path(data_dir, "parsed_matche_since_2019.json"))
Heroes <- fromJSON(file.path(data_dir, "Heroes.json"))
```

```r
#Deleting and renaming Hero name column for more clarity
Heroes <- Heroes %>% select(-starts_with("name"))
Heroes <- Heroes %>%
  rename(name = localized_name)
kable(head(Heroes))
```

id	name	primary_attr	attack_type	roles	legs
1	Anti-Mage	agi	Melee	Carry , Escape, Nuker	2
2	Axe	str	Melee	Initiator, Durable , Disabler , Carry	2
3	Bane	all	Ranged	Support , Disabler, Nuker , Durable	4
4	Bloodseeker	agi	Melee	Carry , Disabler , Nuker , Initiator	2
5	Crystal Maiden	int	Ranged	Support , Disabler, Nuker	2
6	Drow Ranger	agi	Ranged	Carry , Disabler, Pusher	2

Exploring the dimensions

```r
#Checking Rows and variables in this data set.
dim(Player_data)
```

[1] 495 16

```r
#Checking dimension instead of printing a table which will reflect as few pages when knitted.
dim(matches_data)
```

[1] 495 45

```r
# In game chat messages data
Chat_df <- map_df(matches_data$chat, ~ .x)
kable(head(Chat_df))
```

time	type	key	slot	player_slot
-75	chatwheel	56001	3	3
-75	chatwheel	56001	3	3
47	chatwheel	11	7	130
84	chatwheel	2	7	130
91	chatwheel	7	7	130
93	chatwheel	7	7	130

```r
#Dropping empty columns that are not needed.
matches_data <- matches_data %>% select(-starts_with("cosmetics"))
matches_data <- matches_data %>% select(-starts_with("all_word"))
```

```r
#unesting a nested data frame within matches_data
players_df <- map_df(matches_data$players, ~ .x)
```

```r
#Creating New data frame with the required variables and filtering by my personal account Id
new_data_frame <- players_df %>%
  filter(account_id == 191944840) %>%
  select(match_id, account_id, duration, isRadiant, win, hero_id, kills, deaths,
         assists, kda, last_hits, denies, net_worth, total_gold, gold_per_min,total_xp, xp_per_min,
         hero_damage)
dim(new_data_frame)
```

[1] 495 18

```r
#joining data from two different data frames by hero id
new_data_frame <- left_join(new_data_frame, Heroes, by = c("hero_id" = "id"))
```

```r
#Removing irrelevant columns
new_data_frame <- new_data_frame %>%
  select(-legs, -roles)
#Adding New Column(variable) for lost gold(due to death or buybacks)
new_data_frame <- new_data_frame %>%
  mutate(gold_lost = total_gold - net_worth)
#changing duration from second format to minute for readability
new_data_frame <- new_data_frame %>%
  mutate(duration = duration / 60)
dim(new_data_frame)
```

[1] 495 22

Thus far the cleaning process involved several steps:

Filtering Data: As shown in R code, filtered out the matches where my account is involved using filter(account_id == my_ID). Data Type Conversions: Ensure all variables are of the correct data type. Creating New Variables: As needed for the analysis, create new variables (e.g., gold_lost). Dropping Irrelevant Variables: Remove variables that are not relevant to my analysis to simplify the dataset. Merging Dataframes: Unnesting and Joining the hero attributes with match data based on hero_id.

This new data frame includes the most essential variables.

Step 2: Plotting

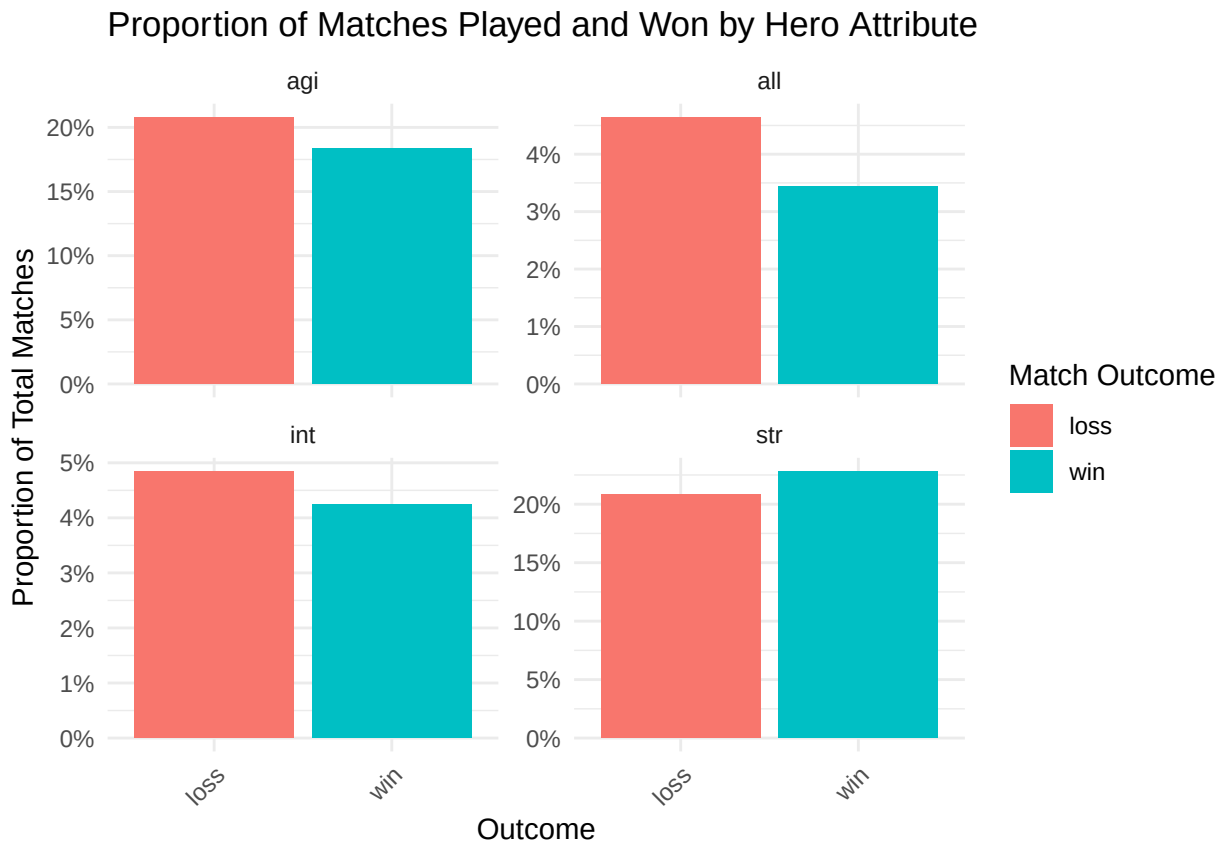
Plotting: Summarizing, Grouping and Plotting Data to Answer Key Questions.

Creating a multifaceted histogram that shows games played versus games won, separated by hero attribute. This will help visualize how often I play and win with heroes of different primary attributes (Agility, Intelligence, Strength, all_universal.

```r
# Create a summary of games played and games won by hero attribute
games_summary <- new_data_frame %>%
  group_by(primary_attr) %>%
  summarise(Games_Played = n(),
            Games_Won = sum(win)) %>%
  gather(key = "Outcome", value = "Count", -primary_attr)
```

```r
# Add a column for losses for convenience
total_games <- nrow(new_data_frame)
new_data_frame <- new_data_frame %>%
  mutate(loss = ifelse(win == 1, 0, 1))

# Melt the data for 'win' and 'loss' columns to long format
long_data <- new_data_frame %>%
  gather(key = "Outcome", value = "Count", win, loss)

# Calculate the proportions for wins and losses
long_data <- long_data %>%
  group_by(primary_attr, Outcome) %>%
  summarise(Total = sum(Count)) %>%
```

```r
  ungroup() %>%
  mutate(Proportion = Total / total_games)

# Plot the data with facets for each hero attribute
ggplot(long_data, aes(x = Outcome, y = Proportion, fill = Outcome)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ primary_attr, scales = "free_y") +
  scale_y_continuous(labels = scales::percent) +
  labs(x = "Outcome", y = "Proportion of Total Matches",
       fill = "Match Outcome",
       title = "Proportion of Matches Played and Won by Hero Attribute") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

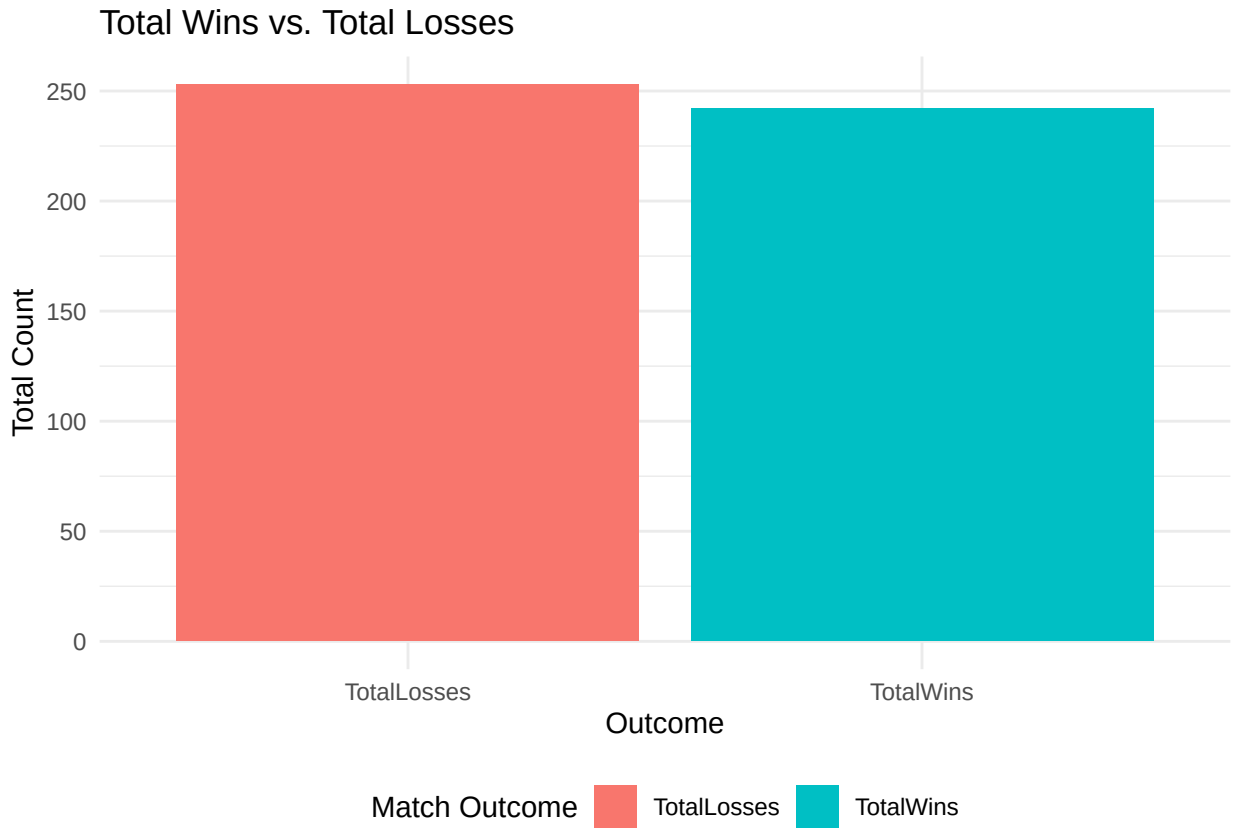


I only have a higher win-than-loss percentage by playing Strength Heroes. Maybe I should consider a different Hobby :)

Plotting Total win VS Tolal loss for perspective

```r
# Summarize wins and losses
summary_df <- new_data_frame %>%
  summarise(TotalWins = sum(win),
            TotalLosses = sum(loss)) %>%
  gather(key = "Outcome", value = "Count", TotalWins, TotalLosses)

# Plot the histogram
ggplot(summary_df, aes(x = Outcome, y = Count, fill = Outcome)) +
```

```
  geom_bar(stat = "identity") +
  scale_y_continuous(labels = scales::comma) +
  labs(x = "Outcome", y = "Total Count",
       fill = "Match Outcome",
       title = "Total Wins vs. Total Losses") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

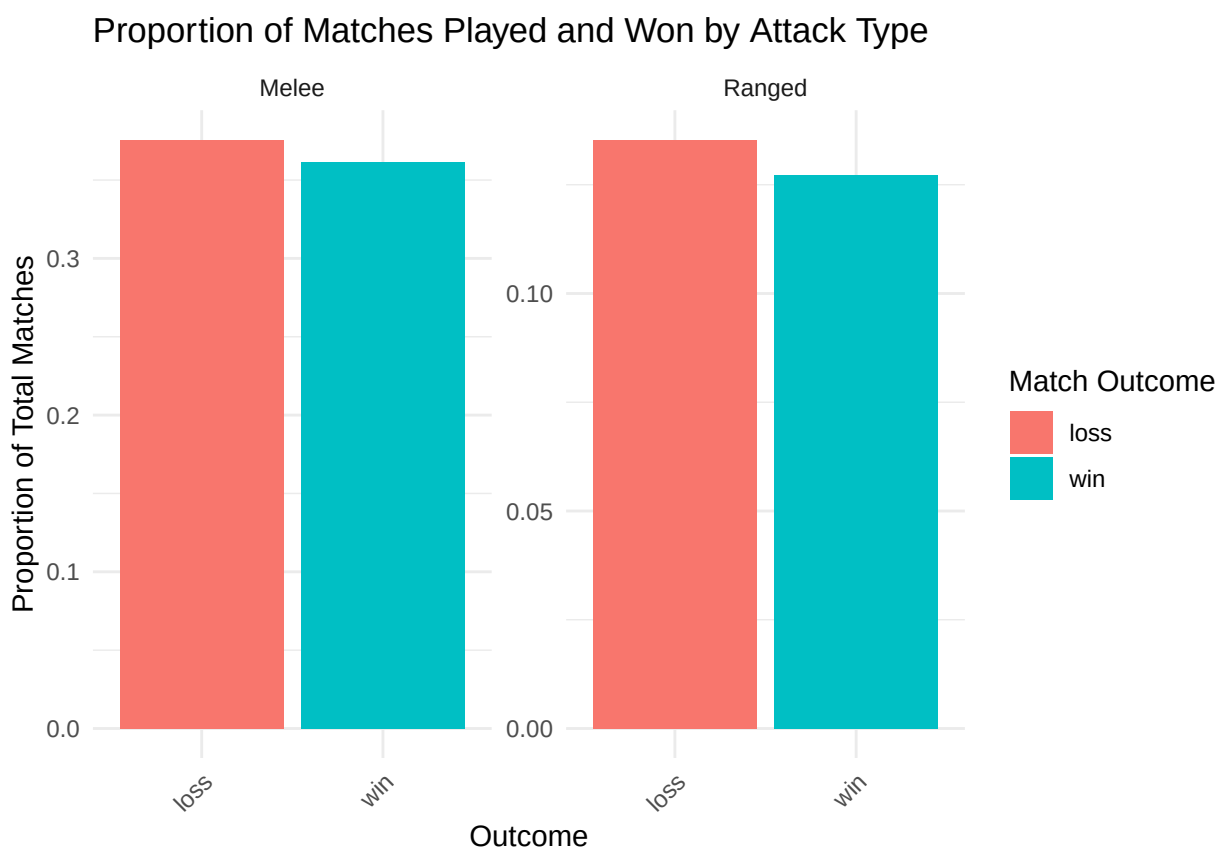


I often hear the phrase "Delete Dota," after seeing my win percentage, I think I should. :(

Creating a multifaceted histogram that shows games played versus games won, separated by hero attack type, will help visualize how often I play and win with heroes of different attack types.

```
#Calculate the proportions for wins and losses
long_data2 <- new_data_frame %>%
  gather(key = "Outcome", value = "Count", win, loss) %>%
  group_by(attack_type, Outcome) %>%
  summarise(Total = sum(Count)) %>%
  ungroup() %>%
  mutate(Proportion = Total / sum(Total))  # Calculate proportions

# Create the plot
ggplot(long_data2, aes(x = Outcome, y = Proportion, fill = Outcome)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ attack_type, scales = "free_y") +
  labs(x = "Outcome", y = "Proportion of Total Matches",
```

```
       fill = "Match Outcome",
       title = "Proportion of Matches Played and Won by Attack Type") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

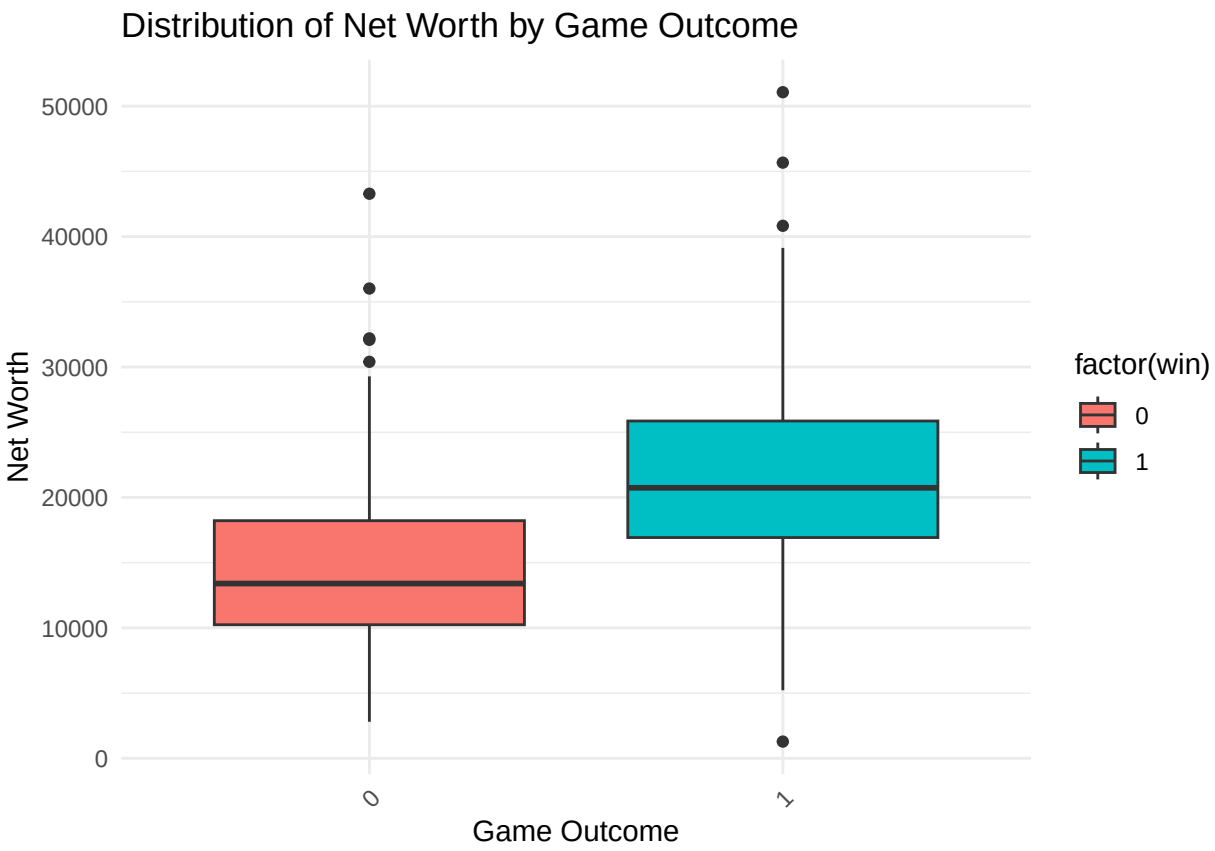


Now a multifaceted plot combining attack_type and primary_attr

```
#Calculate the proportions for wins and losses
long_data3 <- new_data_frame %>%
  gather(key = "Outcome", value = "Count", win, loss) %>%
  group_by(primary_attr, attack_type, Outcome) %>%
  summarise(Total = sum(Count), .groups = 'drop') %>%
  ungroup() %>%
  mutate(Proportion = Total / sum(Total))

# Create the plot
ggplot(long_data3, aes(x = attack_type, y = Proportion, fill = Outcome)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ primary_attr, scales = "free") +
  labs(x = "Attack Type", y = "Proportion of Total Matches",
       fill = "Match Outcome",
       title = "Proportion of Matches Won and Lost by Attack Type within each Primary Attribute") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

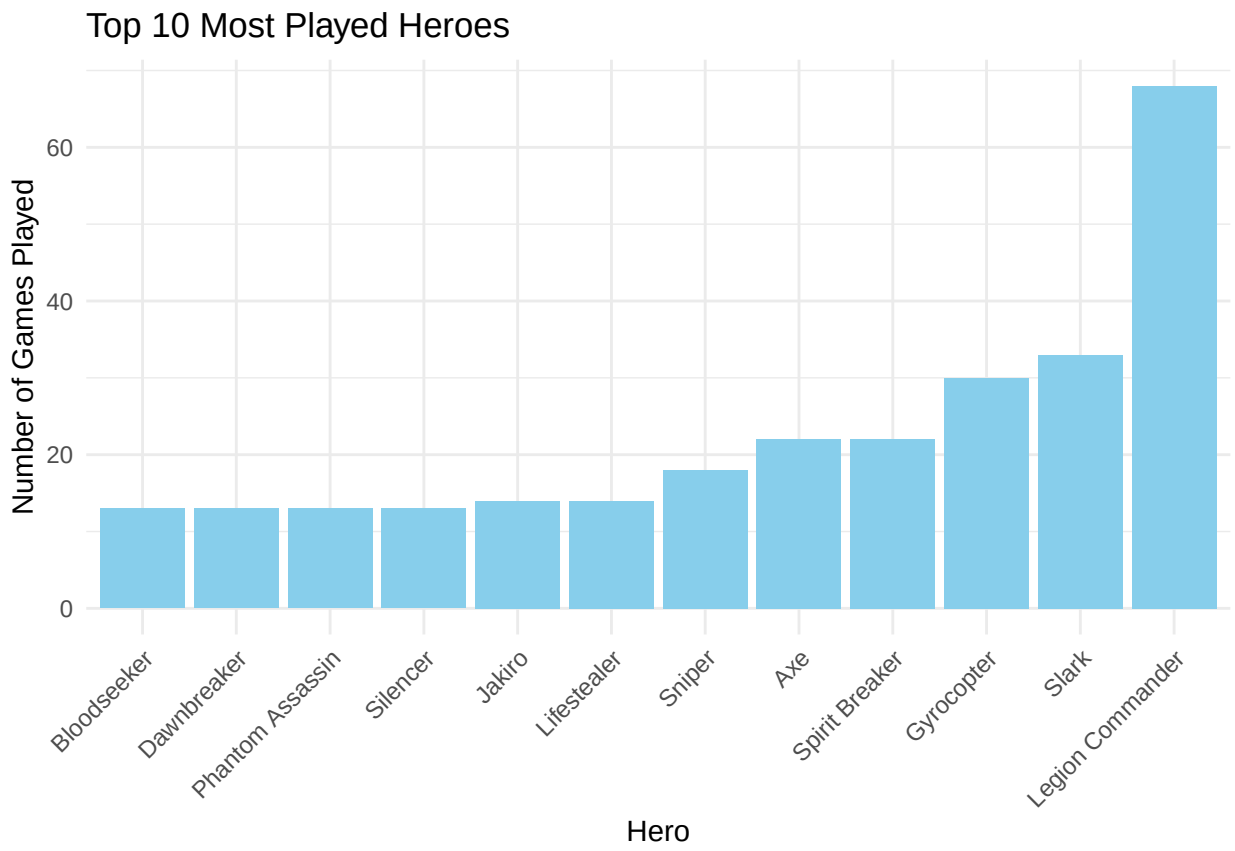
Box plot of Net Worth by Game Outcome

```
ggplot(new_data_frame, aes(x = factor(win), y = net_worth, fill = factor(win))) +
  geom_boxplot() +
  labs(x = "Game Outcome", y = "Net Worth",
       title = "Distribution of Net Worth by Game Outcome") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

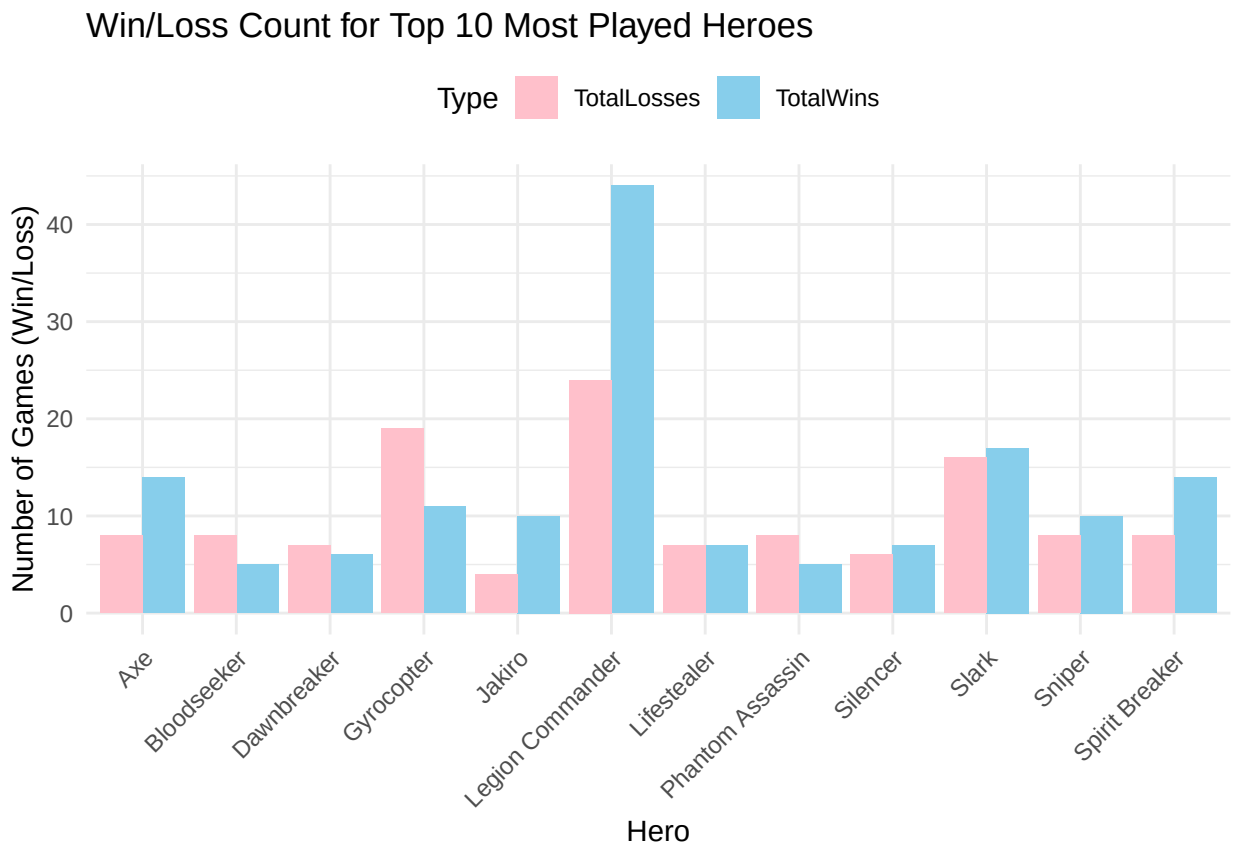


Top 10 most played heroes.

```
# Calculate the number of games for each hero
hero_play_count <- new_data_frame %>%
  count(name) %>%
  arrange(desc(n)) %>%
  top_n(10, n)  # Select the top 10 most played heroes

# Create the bar chart
ggplot(hero_play_count, aes(x = reorder(name, n), y = n, fill = name)) +
  geom_bar(stat = "identity", position = "dodge", fill = "skyblue") +
  labs(x = "Hero", y = "Number of Games Played",
       title = "Top 10 Most Played Heroes") +

  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), legend.position = "none") +
  guides(fill = FALSE)  # Hide the legend, as it's redundant in this case
```

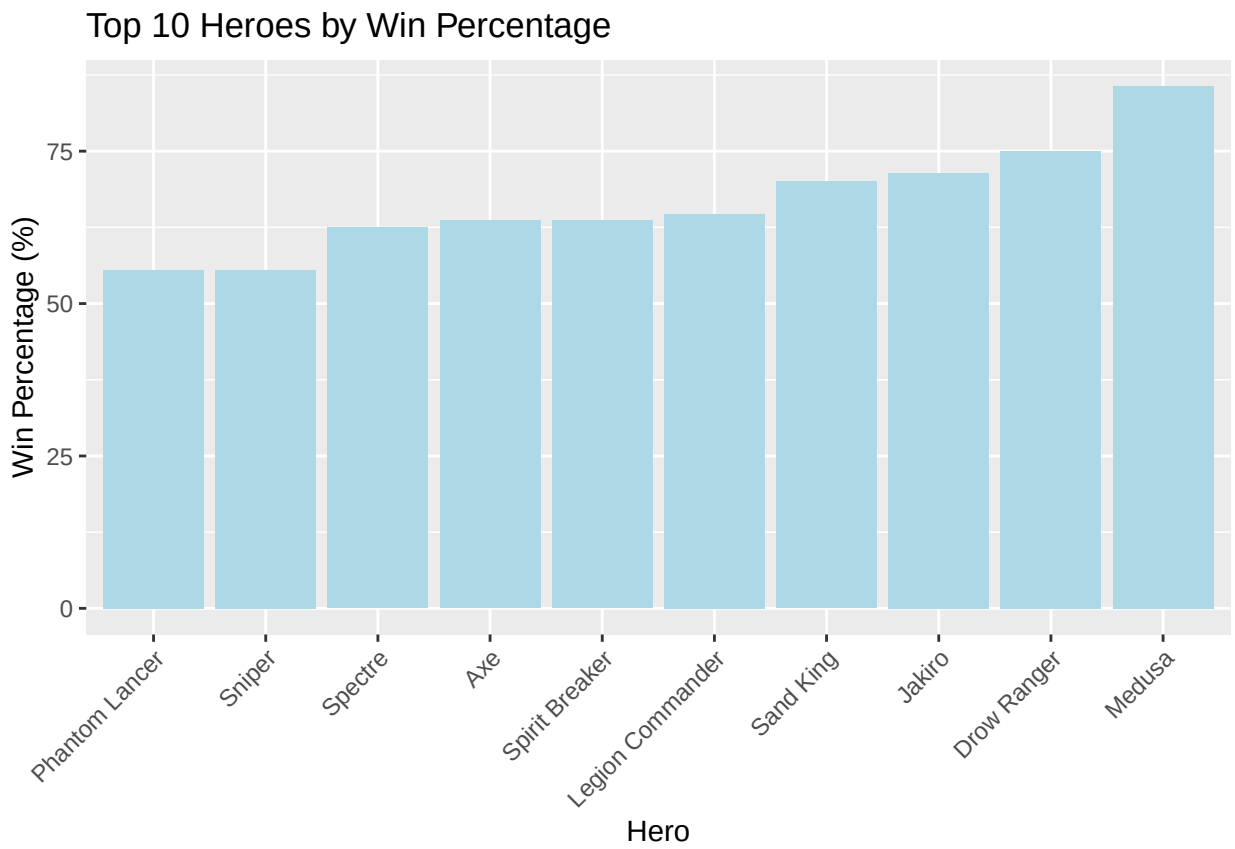



```r
# Calculate total wins and losses for each hero
total_wins_losses <- new_data_frame %>%
  filter(name %in% hero_play_count$name) %>%
  group_by(name) %>%
  summarise(TotalWins = sum(win), TotalLosses = sum(loss)) %>%
  ungroup()

# Merge with hero_play_count to get a combined data frame
combined_data <- merge(hero_play_count, total_wins_losses, by = "name")

# Prepare data for plotting
long_data_top_heroes <- combined_data %>%
  gather(key = "Outcome", value = "Count", TotalWins, TotalLosses)

# Create the bar chart for total games, wins, and losses
ggplot(long_data_top_heroes, aes(x = name, y = Count, fill = Outcome)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Hero", y = "Number of Games (Win/Loss)",
       title = "Win/Loss Count for Top 10 Most Played Heroes") +
  scale_fill_manual(values = c("TotalWins" = "skyblue", "TotalLosses" = "pink")) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), legend.position = "top") +
  guides(fill = guide_legend(title = "Type"))
```



We can visually see my success rate for my most ten played heroes. However, considering the proportion, I have only succeeded by playing Axe, Legion Commander, and Spirit Breaker. Therefore, I want to check the top 10 heroes based on my win rate percentage for heroes played at least 5 games.

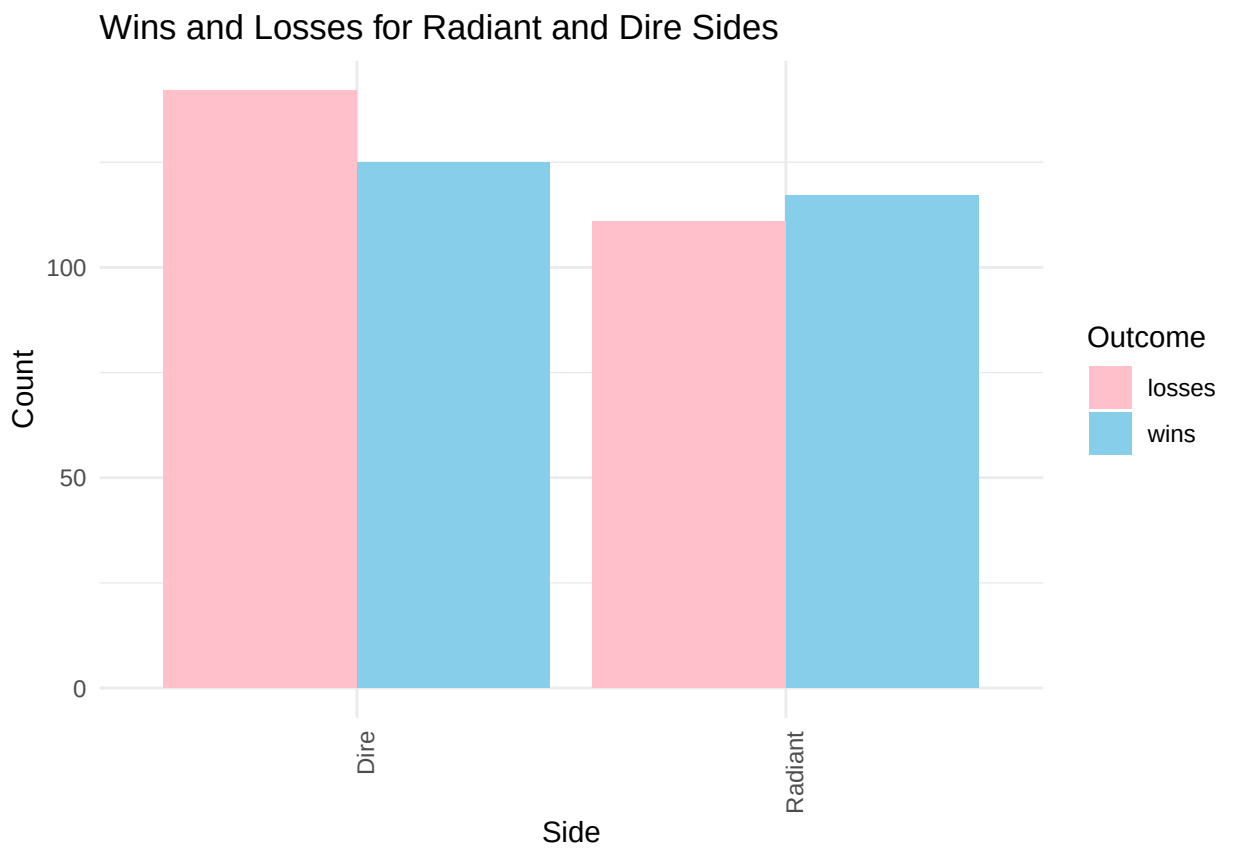
```
# Aggregate data to sum wins and count total games
aggregated_data <- new_data_frame %>%
  group_by(hero_id, name) %>%
  summarize(total_wins = sum(win),
            total_games = n(),
            win_percentage = (total_wins / total_games) * 100) %>%
  ungroup()

# Filter out heroes with a low number of games, sort, and select top 10
top_10_heroes <- aggregated_data %>%
  filter(total_games > 5) %>%
  arrange(desc(win_percentage)) %>%
  head(10)

ggplot(top_10_heroes, aes(x = reorder(name, win_percentage), y = win_percentage)) +
  geom_bar(stat = "identity", fill = "lightblue") +
  labs(title = "Top 10 Heroes by Win Percentage",
       x = "Hero",
       y = "Win Percentage (%)") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



So we notice that considering win percentage, Axe and Legion Commander become ranked five and six, and we see new heroes that did not come up on my most played list have higher win percentage in my case. Next, I want to plot the Win vs Loss count when I play as a Radiant versus when I play as Dire.

Team side played

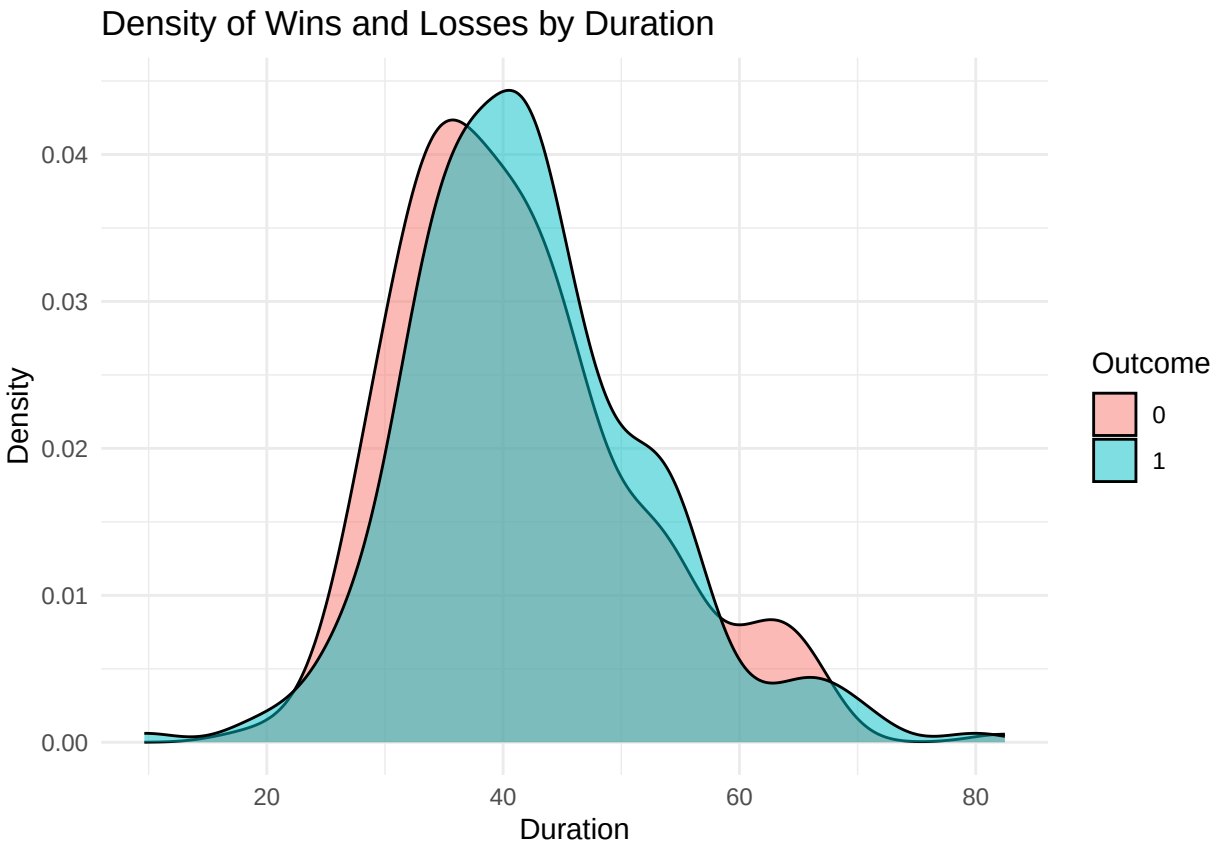
```
# Categorize data as Radiant or Dire
new_data_frame <- new_data_frame %>%
  mutate(side = ifelse(isRadiant, "Radiant", "Dire"))

# Calculate wins and losses
side_wins_losses <- new_data_frame %>%
  group_by(side) %>%
  summarize(wins = sum(win),
            losses = sum(1 - win)) %>%
  ungroup()

long_side_wins_losses <- side_wins_losses %>%
  gather(key = "Outcome", value = "Count", wins, losses)

ggplot(long_side_wins_losses, aes(x = side, y = Count, fill = Outcome)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Wins and Losses for Radiant and Dire Sides",
       x = "Side",
       y = "Count") +
  theme_minimal() +
  scale_fill_manual(values = c("wins" = "skyblue", "losses" = "pink")) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Excellent, I will start playing more on the Radiant side!

Lastly, I want to visualize the density of matches won by duration to understand if longer matches affect my performance and, therefore, the game outcome.

```
ggplot(new_data_frame, aes(x = duration, fill = factor(win))) +
  geom_density(alpha = 0.5) +
  labs(title = "Density of Wins and Losses by Duration",
       x = "Duration",
       y = "Density",
       fill = "Outcome") +
  theme_minimal()
```



The plot graph could be more informative; the outcomes are closely similar. Therefore, I can not conclude that I have a better chance of winning longer or shorter games.

Step3: Modeling

Correlation

In analyzing my gaming dataset, addressing collinearity among variables is crucial to avoid skewed interpretations. Looking forward, I aim to harness machine-learning techniques. Backward stepwise regression will help isolate the most impactful predictors of victory. Statistical analysis could uncover patterns in playing styles, and classification algorithms like logistic regression enable us to predict match outcomes with greater accuracy, providing valuable insights into the variables that govern game success.

First, I checked the remaining variables that include NA values in the data set:

```
na_counts <- colSums(is.na(new_data_frame))
na_counts
```

```
match_id   account_id     duration    isRadiant          win      hero_id
       0            0            0            0            0            0
   kills       deaths      assists          kda    last_hits       denies
       0            0            0            0            0            0
```

net_worth total_gold gold_per_min total_xp xp_per_min hero_damage 0 0 0 0 0 250 name primary_attr attack_type gold_lost loss side 0 0 0 0 0 0 Hero damage is an important predictive factor for wining, it seems that OpenDota was not saving these data in the older games, However, I would rather have them replaced

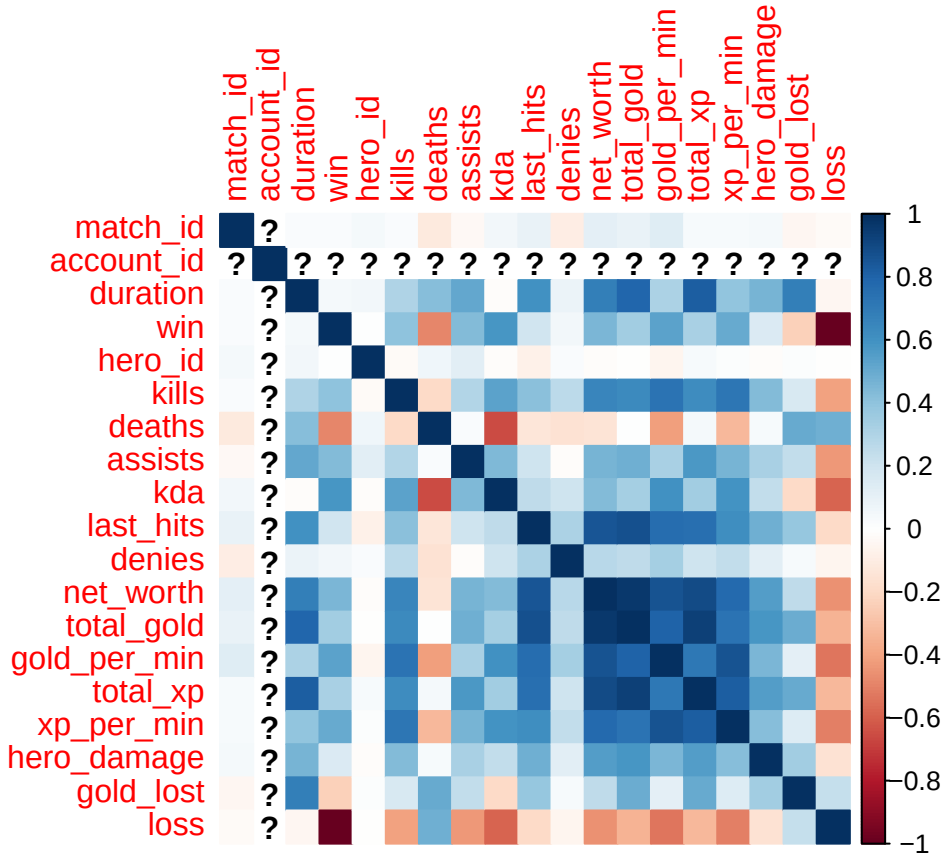
with the mean value of the remaining 245 games.

```
# Calculate the mean of the non-NA values
mean_hero_damage <- mean(new_data_frame$hero_damage, na.rm = TRUE)

# Replace NA values with the mean
new_data_frame$hero_damage[is.na(new_data_frame$hero_damage)] <- mean_hero_damage
```

The second step is to check for collinearity:

```
# Select only numeric columns

numeric_data <- new_data_frame %>%
  select(where(is.numeric))

# Compute the correlation matrix on the numeric data
correlation_matrix <- cor(numeric_data)
```

```
#visualizing collinearity
library(corrplot)
if (requireNamespace("corrplot", quietly = TRUE)) {
  corrplot::corrplot(correlation_matrix, method = "color")
}
```



The next step is fitting a predictive model:

Model:

Splitting data:

```
#Removing variables with perfect collinearity and insignificant correlation:
model_data_frame <- new_data_frame %>%
  select(-account_id, -loss, -match_id, -hero_id, -name)
# Ensure that 'win' is a factor
model_data_frame$win <- as.factor(new_data_frame$win)
```

```
# Set a seed for reproducibility
set.seed(123)

# Split the data into training and test sets
splitIndex <- createDataPartition(model_data_frame$win, p = 0.8, list = FALSE, times = 1)
train_data <- model_data_frame[splitIndex, ]
test_data <- model_data_frame[-splitIndex, ]
```

Fitting a Logistic model

```
# Create a formula for the full model
full_formula <- as.formula(paste("win ~", paste(names(train_data)[names(train_data) != "win"], collapse

# Fit the full logistic regression model
full_model <- glm(full_formula, data = train_data, family = binomial())

# Perform backward stepwise selection
final_model <- step(full_model, direction = "backward")
```

Start: AIC=246.32 win ~ duration + isRadiant + kills + deaths + assists + kda + last_hits + denies + net_worth + total_gold + gold_per_min + total_xp + xp_per_min + hero_damage + primary_attr + attack_type + gold_lost + side

Step: AIC=246.32 win ~ duration + isRadiant + kills + deaths + assists + kda + last_hits + denies + net_worth + total_gold + gold_per_min + total_xp + xp_per_min + hero_damage + primary_attr + attack_type + gold_lost

Step: AIC=246.32 win ~ duration + isRadiant + kills + deaths + assists + kda + last_hits + denies + net_worth + total_gold + gold_per_min + total_xp + xp_per_min + hero_damage + primary_attr + attack_type

```
           Df Deviance    AIC
```

- primary_attr 3 209.86 241.86
- assists 1 208.32 244.32
- deaths 1 208.41 244.41
- isRadiant 1 208.77 244.77
- hero_damage 1 209.64 245.64
- attack_type 1 209.89 245.89 208.32 246.32
- gold_per_min 1 210.32 246.32
- duration 1 210.70 246.70
- denies 1 210.76 246.76
- xp_per_min 1 213.03 249.03
- total_xp 1 213.55 249.55
- total_gold 1 214.78 250.78
- kda 1 214.99 250.99

- net_worth 1 218.74 254.74
- kills 1 224.19 260.19
- last_hits 1 230.34 266.34

Step: AIC=241.86 win ~ duration + isRadiant + kills + deaths + assists + kda + last_hits + denies + net_worth + total_gold + gold_per_min + total_xp + xp_per_min + hero_damage + attack_type

```
           Df Deviance    AIC
```

- assists 1 209.87 239.87
- deaths 1 209.91 239.91
- isRadiant 1 210.41 240.41
- hero_damage 1 211.09 241.09 209.86 241.86
- gold_per_min 1 212.15 242.15
- duration 1 212.22 242.22
- attack_type 1 212.22 242.22
- denies 1 212.57 242.57
- xp_per_min 1 215.43 245.43
- total_xp 1 215.86 245.86
- total_gold 1 216.82 246.82
- kda 1 217.21 247.21
- net_worth 1 220.14 250.14
- kills 1 227.27 257.27
- last_hits 1 233.03 263.03

Step: AIC=239.87 win ~ duration + isRadiant + kills + deaths + kda + last_hits + denies + net_worth + total_gold + gold_per_min + total_xp + xp_per_min + hero_damage + attack_type

```
           Df Deviance    AIC
```

- deaths 1 209.93 237.93
- isRadiant 1 210.41 238.41
- hero_damage 1 211.09 239.09 209.87 239.87
- gold_per_min 1 212.23 240.23
- attack_type 1 212.23 240.23
- duration 1 212.23 240.23
- denies 1 212.59 240.59
- xp_per_min 1 215.57 243.57
- total_xp 1 215.91 243.91
- total_gold 1 217.08 245.08
- net_worth 1 220.14 248.14
- kda 1 231.06 259.06
- last_hits 1 235.32 263.32
- kills 1 242.17 270.17

Step: AIC=237.93 win ~ duration + isRadiant + kills + kda + last_hits + denies + net_worth + total_gold + gold_per_min + total_xp + xp_per_min + hero_damage + attack_type

```
           Df Deviance    AIC
```

- isRadiant 1 210.48 236.48
- hero_damage 1 211.16 237.16 209.93 237.93
- gold_per_min 1 212.30 238.30
- attack_type 1 212.32 238.32
- duration 1 212.70 238.70
- denies 1 212.70 238.70
- xp_per_min 1 215.60 241.60
- total_xp 1 215.94 241.94

- total_gold 1 217.10 243.10
- net_worth 1 221.15 247.15
- last_hits 1 241.10 267.10
- kills 1 245.55 271.55
- kda 1 262.27 288.27

Step: AIC=236.48 win ~ duration + kills + kda + last_hits + denies + net_worth + total_gold + gold_per_min + total_xp + xp_per_min + hero_damage + attack_type

```
            Df Deviance    AIC
```

- hero_damage 1 211.61 235.61 210.48 236.48
- attack_type 1 212.91 236.91
- gold_per_min 1 213.05 237.05
- denies 1 213.08 237.08
- duration 1 213.38 237.38
- xp_per_min 1 216.27 240.27
- total_xp 1 216.56 240.56
- total_gold 1 217.86 241.86
- net_worth 1 222.33 246.33
- last_hits 1 242.07 266.07
- kills 1 246.59 270.59
- kda 1 263.32 287.32

Step: AIC=235.61 win ~ duration + kills + kda + last_hits + denies + net_worth + total_gold + gold_per_min + total_xp + xp_per_min + attack_type

```
            Df Deviance    AIC
```

211.61 235.61 - gold_per_min 1 214.03 236.03 - duration 1 214.16 236.16 - attack_type 1 214.38 236.38 - denies 1 214.63 236.63 - xp_per_min 1 217.59 239.59 - total_xp 1 218.06 240.06 - total_gold 1 218.41 240.41 - net_worth 1 226.62 248.62 - last_hits 1 243.18 265.18 - kills 1 249.26 271.26 - kda 1 263.59 285.59

```
# Summary of the model
summary(final_model)
```

Call: glm(formula = win ~ duration + kills + kda + last_hits + denies + net_worth + total_gold + gold_per_min + total_xp + xp_per_min + attack_type, family = binomial(), data = train_data)

Coefficients: Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.8695947 2.9700883 -1.640 0.101100
duration -0.1253556 0.0785977 -1.595 0.110734
kills -0.3714326 0.0680793 -5.456 4.87e-08 ***kda 1.3771425 0.2369981 5.811 6.22e-09*** last_hits -0.0255473 0.0052266 -4.888 1.02e-06 ***denies -0.0717794 0.0421001 -1.705 0.088200 .***
net_worth 0.0004414 0.0001282 3.444 0.000574 total_gold 0.0008215 0.0003244 2.532 0.011334 *
gold_per_min -0.0189670 0.0123071 -1.541 0.123281
total_xp -0.0005594 0.0002273 -2.461 0.013844 *
xp_per_min 0.0221918 0.0092838 2.390 0.016831 *
attack_typeRanged 0.6797643 0.4125700 1.648 0.099428 .
— Signif. codes: 0 ‘***’ ***0.001*** ‘**’ *0.01* ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

```
Null deviance: 550.15  on 396  degrees of freedom
```

Residual deviance: 211.61 on 385 degrees of freedom AIC: 235.61

Number of Fisher Scoring iterations: 7 I have tried a few model iteration and so far This logistic regression model derived from the training data presents a notably more stable set of results compared to previous iterations. Key predictors such as kills, kda, last_hits, net_worth, total_gold, total_xp, and xp_per_min

demonstrate statistically significant associations with the probability of winning a match. The model's overall fit has improved, as evidenced by a lower residual deviance and a reduced AIC score, indicating enhanced predictive power. Interestingly, the variable attack_typeRanged shows a positive, albeit not statistically significant, relationship with winning, suggesting potential nuances in how different hero types influence game outcomes. With only seven iterations required for convergence, the model fitting process appears robust, paving the way for validation against the test dataset to evaluate the model's generalization and for further diagnostics to ensure the reliability of the predictors' effects.

Model Validation

```
# Predict probabilities on the test set
test_data$predicted_prob <- predict(final_model, newdata = test_data, type = "response")

# Convert probabilities to binary predictions based on a 0.5 threshold
test_data$predicted_class <- ifelse(test_data$predicted_prob > 0.5, 1, 0)
```

```
# Calculate the confusion matrix
conf_matrix <- confusionMatrix(as.factor(test_data$predicted_class), as.factor(test_data$win))

# Print the confusion matrix
print(conf_matrix)
```

Confusion Matrix and Statistics

```
      Reference
```

Prediction 0 1 0 48 12 1 2 36

```
           Accuracy : 0.8571
             95% CI : (0.7719, 0.9196)
No Information Rate : 0.5102
P-Value [Acc > NIR] : 5.065e-13

              Kappa : 0.713
```

Mcnemar's Test P-Value : 0.01616

```
        Sensitivity : 0.9600
        Specificity : 0.7500
     Pos Pred Value : 0.8000
     Neg Pred Value : 0.9474
         Prevalence : 0.5102
     Detection Rate : 0.4898
```

Detection Prevalence : 0.6122
Balanced Accuracy : 0.8550

```
   'Positive' Class : 0
```

```
# Print accuracy, precision, recall, and F1 score
accuracy <- conf_matrix$overall['Accuracy']
precision <- conf_matrix$byClass['Pos Pred Value']
recall <- conf_matrix$byClass['Sensitivity']
F1_score <- 2 * (precision * recall) / (precision + recall)

print(paste("Accuracy:", accuracy))
```

[1] “Accuracy: 0.857142857142857”

```
print(paste("Precision:", precision))
```

[1] “Precision: 0.8”

```
print(paste("Recall (Sensitivity):", recall))
```

[1] “Recall (Sensitivity): 0.96”

```
print(paste("F1 Score:", F1_score))
```

[1] “F1 Score: 0.872727272727273” The logistic regression model used in this project proved to be a powerful tool in predicting match outcomes based on key performance metrics. The model achieved an accuracy of 85.71%, significantly outperforming the baseline no information rate of 51.02%. Key predictors such as kills, KDA (Kill-Death-Assist ratio), last hits, net worth, total gold, total XP, and XP per minute were identified as significant contributors to the likelihood of winning a match.

The model's performance was validated against a test dataset, demonstrating high sensitivity (96%) and precision (80%), with an overall F1 score of 0.8727. These results indicate a strong capability of the model to predict match outcomes accurately.

Moving forward, further refinement of the model and additional diagnostic checks are essential to ensure its robustness and generalizability. Additionally, exploring time series analysis and natural language processing for in-game communication data could provide deeper insights into performance trends and the impact of teamwork on match outcomes.

Step4: Conclusion

Conclusion

In this analysis, I delved into understanding my gaming strategy and win probability in Dota 2, focusing on factors like the chosen heroes, their attack types, primary attributes, and team sides. Through thorough data collection, cleaning, and preprocessing, I built a comprehensive dataset representing my gameplay history. This dataset facilitated an extensive exploratory data analysis (EDA), which helped in identifying key variables influencing match outcomes.

Subsequently, I developed a predictive model using logistic regression, which demonstrated strong performance in predicting match outcomes based on key in-game metrics. The model's accuracy, sensitivity, precision, and F1 score were all notably high, indicating its effectiveness in identifying the factors that contribute to winning a match.

Despite the success of the model, some initial research questions remain unaddressed. For instance, a time series analysis could provide insights into the evolution of my performance over time, highlighting areas of improvement or decline. Although specific dates for each match are unavailable, this analysis could be approximated by grouping matches using their ascending match IDs.

Furthermore, analyzing the impact of communication and teamwork on securing wins requires a more intricate approach, ideally involving natural language processing (NLP) of in-game chat data. Unfortunately, due to the absence of voice chat data, this analysis has limitations.

The logistic regression model, while robust, requires additional diagnostic checks to confirm its accuracy and reliability. This step is crucial to ensure the model's generalizability beyond the current dataset.