League of Legends Analysis

Team Members: Alexa, Damon, Yvanka, & Vlad

Introduction

Technological advancements over the past few decades have led to computers being more accessible, and thus video games to being more popular. The gaming industry has also seen a shift as people are more willing to go out and watch others play eSports, rather than just play them at home alone. ESports, or electronic sports, pertains to a competition that is typically contended between professional players at organized events. The eSports community has exponentially grown over the last several years. ESports players can make a seven-figure salary, teenagers can get college scholarships for gaming, and tickets to watch competitive eSports tournaments are often sold out (Rosenberg and Hill, 2016). Despite the rise of this industry, which according to NewZoo and CNBC is worth around $140 billion, there has not been much discussion about how to optimize professional eSports teams. This research report explored the inner workings of League of Legends and found which classes of Champions could potentially lead to a higher chance at winning if selected before the match begun. This report also addresses six factors which contribute to the overall KDA (Kills/Deaths/Assists) of an eSports athlete.

Through our research, we found that the estimated eSports market revenue increased from $1.8 billion in 2017 to approximately $3.5 billion by 2021 (Cottrell,2018). Because of the exponentially growing eSports market, we decided to provide strategic consultation services for professional eSports competition teams to optimize their performance. Therefore, our main business motivation/goals are as follows:

1. Help eSports team coaches/managers formulate the optimal team composition of Champion classes to maximize their chances of winning (*win rates*) in tournaments
2. Increase players awareness of other in-game tactics that are KDA contributors

Datasets Description & Manipulation

Our League of Legends (LoL) dataset includes 1,000,000 observations within Ranked Games from an existing dataset on Kaggle. The data we used from this data set included, a player's id, a champion's id, whether they won or lost their game, kills, deaths, assists, total damage dealt to champions, total damage taken, and vision score. These observations are all related to an individual player. League of Legends is a team sport consisting of 5 players per

team and a one-one-one team match system. Therefore, we decided to create a column to indicate which team a player belonged to, as well as the match a player participated in.

While these games were taken from Ranked Matches, which makes the game more competitive, in some cases a player's computer can fail or they can 'rage quit' and stop playing the match. This was indicative to us if a player's kills + deaths + assists were equal to 0. It is nearly impossible to contribute nothing within a Ranked match that usually lasts anywhere from 30 to 80 minutes. Therefore, we first found which players AFK'd, or did not participate, from the game. Then, we eliminated all the matches that these players belong to as we did not think it a fair match if not all players were present. This eliminated about 200,000 observations out of the 1,000,000 we started with (Figure 1).

We also needed to create a column that indicated what class a champion belonged within. Classes included: Fighter, Mage, Controller, Marksman, Tank, and Slayer. Each classification has a different role within a game which is important in strategizing how to win. A fighter will deal high damage to enemies, but also withstand a fair amount of damage. Mages are good with long-distance attacks, burst damage, and mainly utilize their ability power levels. Controllers use CC, or crowd control, to weaken the enemies' ability to fight. Such tactics can include slowing an enemies run speed, stunning/paralyzing them, lowering the attacks, etc. Marksman, similar to mages, will use long-distance attacks. However, instead of ability power, they focus on using attack damage. Tank champions will be able to withstand a lot of damage and serve as a shield for their teammates. Finally, slayers have very high attack damage and can quickly kill enemies. However, they have a low defense and can also be killed fairly quickly. We placed all champions into the best possible fit for their classes.

*To make our points clear, we will address the following questions case by case.

Problem 1

Methodologies

To analyze our first motivation - to help eSports team coaches/managers formulate the optimal team composition of Champion classes and maximize their chances of winning, we utilized a logistic regression. The model we chose to use in order to portray this analysis was a logistic regression. We chose this model because we wanted to perform predictive analysis and our dependent variable is dichotomous, win rate. We define winning a match as having a win rate over 0.50 as it would give a player a competitive edge over their opponent. Before we ran

this model, we evenly split our data into train and test datasets. Our predictors (input variables) for this logistic regression are: *the number of Fighters, Marksmen, Mages, Controllers, Slayers,* and *Tanks*. Our response (output variable) is *win rate* for each classification of champion (Figure 2).

Results and Conclusion

Using our first model(Linear Regression on *win rates*), we found that for every increase in 1 Fighter per team, the win rate for that team goes up by 5.56%**, for every increase in 1 Controller per team, the win rate for that team goes up by 6.02%**, and for every increase in 1 Marksman per team, the win rate for that team goes down by 7.99%*** .("**" denoting moderate significance, and "***" denoting prominent significance)

This data was interesting because our hypothesis was that the more tank champions per team, the higher the win rate would be. This is because a tank champion is very hard to kill, so the other team would make less gold by killing less champions. It would also be hard for the opponent to complete in-game objectives if all of the tanks were still alive. However, the rest of the data makes sense from a LoL perspective. Marksman are not usually great in late game and usually are just important early game. Therefore, having more of them would hinder your win rate. On the other hand, having a fighter to sustain damage and deal a lot of damage to enemy champions, as well as having a controller to aid the fighter would be beneficial to a team in the late game trying to win. Therefore, our output makes sense that *win rates* would go up if you have more fighters or controllers.

Limitations

In order to check how accurate our model is we first predicted the probability of winning for each participant based on our logistic regression. Then, we assigned a value of 0 for a probability below or equal to 0.50 and a value of 1 for a probability above 0.50. This indicates whether a player was more likely to win the game or lose. Afterwards, we took the mean of loses or wins, which are identified as a 0 or 1, and got 0.5114. Therefore, our model is 51.14% accurate. We also used the stargazer package in order to get a general sense of the breakdown of wins, and champ classes (Figure 3).

While this model indicates probability of winning based on class selection of champs, we believe there are a few improvements we can make to our model in order to improve it in the future. Firstly, we only take into consideration class selection per team. Our model would be

more accurate if we included class countertypes and how champion classifications would perform based on the opposing teams' composition of champions. Furthermore, while we did identify each champion's class, we did not take into consideration their tier level. Each champion's tier level is indicative of how powerful they can become in game. For example, god tier champion is known to win more games than an average tier champion. One limitation of our model is the champion classification method. We assigned each champion into one of six classifications, however some champions fit into more than one of the main six classifications.

Problem 2

The question we address in this section is amongst the six in-game metrics (identified below), which one contributes to a team's KDA the most. A more detailed explanation of the variables is as follows:

$$* \ KDA \ = \ \frac{Kills \ + Assists}{Deaths}$$

* Contributing factors:

Single Factor:

- Vision score (*VS*): [total # of minutes the team had wards spying on the enemy team] + [total # of minutes the team prevented the enemy team from placing wards on the map)
- Total damage dealt to champions (*TDC*): total damage dealt to enemy players by the team
- Total damage taken (*TDT*): total damage sustained by the team from all sources, such as enemy champions, creeps (NPCs, or non-playable characters) and turrets

Interaction between (including all possible combination of the independent variables above):

- *VS* and *TDC*,
- *VS* and *TDT*,
- *TDC* and *TDT*

We then went ahead and built our multiple linear regression model to explain the above relationships as follows:

KDA= β_0 + β_1* VS + β_2* TDC+ β_3* TDT+ β_4 * VS * TDC + β_5* VS * TDT + β_6* DTC * TDT

Methodologies:

Before we could implement our multiple linear regression model in R, we needed to do some more cleaning of the data previously used in Question 1. More specifically, we needed to find the sum of each team's KDAs, vision scores, total damages dealt to champions, and total damages taken – on the *stats1* table of our Kaggle dataset. The way we did it is by:

1. Including an extra column for each variable, which would later hold the totals for each current row and subsequent 4 rows of data (thus denoting the totals per team of 5 players)
2. Calculating those totals by using the Excel built-in ROW() and MOD() functions, which allowed us to only calculate the totals for the row numbers where their difference with the original row number was divisible by 5 (see *DA_Project_Group8_MLR_Data_Cleaning.csv*)

Next, we imported the cleaned data into R with the following variables: *teamKills, teamAssists, teamDeaths, teamVS, teamTDC, teamTDT.* We then deleted the rows containing all zeros (essentially all rows in between the team totals). This left us with only 193,860 data points (down from the original 969,300).

After that, we normalized the values in all columns by subtracting the mean value per column from each individual value and diving the difference by the standard deviation per column. We then created a new *teamKDA* column by summing the *teamKills* and *teamAssists* columns and dividing the sum by *teamDeaths.* Lastly, we added a column to hold an interaction term for all possible combinations of our independent variables, that is:

- *teamVS*teamTDC* (*VS_TDC* column in R data frame),
- *teamVS*teamTDT* (*VS_TDT* column in R data frame),
- *teamTDT*teamTDC* (*TDT_TDC* column in R data frame)

Results and Conclusion:

We ran our multiple linear regression on *teamKDA* as the response variable and the rest as prediction variables. We obtained the following output from the model-displayed in Appendix Figure 4)

We further performed Best Subset Selection on the above model (allowing up to 6 predictors) but found that the best subset of 6 predictors would still include all the 6 predictors from our original model. Therefore, no changes were made to our original model.

In conclusion, we saw that *TDT_TDC* was the biggest contributor to *teamKDA.* This makes intuitive sense since the more damage the team deals to enemy champions, and the more damage the team takes during the match, the higher KDA the team is going to achieve.

However, a less intuitive conclusion was that *teamVS* and *teamTDC* on their own were not statistically significant contributors to *teamKDA* – but that *VS_TDC* and *VS_TDT* were. This means that if the team's vision score grows together with the total damage dealt to enemy champions, as well as the total damage taken, only then there is going to be a statistically significant increase in the team's KDA. Intuitively, this signals that increasing the team's vision score or total damage dealt to enemy champions alone will not achieve a higher KDA. Another counter-intuitive conclusion was that *VS_TDC* was a negative contributor to *teamKDA,* while *VS_TDT* was a positive one. The only logical interpretation we could think of was that placing own wards or destroying enemy wards on the map made sense only if the team is not engaging too much with the enemy champions in the process. In other words, taking damage while dealing with wards (equivalent to dealing with vision score) is allowed, yet returning damage would actually distract the team from focusing on the wards.

Therefore, the main takeaway from our model for professional League of Legends players is as follows: **In order to maximize your team's KDA, place own wards or destroy enemy wards on the map while staying away from full-on engaging with the enemy.**

Limitations:

As you can see from the model output, its adjusted R^2 was very low (less than 1%). The implication here is that our model explains only a small portion in the variation of the team KDA in the past League of Legends ranked matches.

While the conclusions from our model make a lot of logical sense (which in our opinion is more important than the accuracy of the model as such), there is a way to better the accuracy of our model in the future. This can be done by collecting more data on the potential factors influencing the team KDA. Even though we were only restricted to in-game metrics in our data, if more variables could be included on the actual psychological state of players in ranked matches (such as on the scale from 1 to 10, how focused the team was on a given match), the performance of our model could improve drastically.

References

1. Cottrell, C., McMillen, N., & Harris, B. (2018). “Sport psychology in a virtual world: Considerations for practitioners working in eSports”. Journal of Sport Psychology In Action: 1-9.
2. Esportsearnings.com. (2019). “Highest Overall Earnings - Esports Player Rankings: Esports Earnings”. Available at: https://www.esportsearnings.com/players.
3. Pannekeet, J. (2019). “Global Esports Economy Will Top $1 Billion for the First Time in 2019”. NewZoo. Available at: https://newzoo.com/insights/articles/newzoo-global-esports-economy-will-top-1-billion-for-the-first-time-in-2019/.
4. Kaggle: Your Home for Data Science. (2019). Retrieved 12 December 2019, from http://www.kaggle.com/

Appendix

```r
#Create a MatchID
Stats$match_index <- c(0, rep(1:(nrow(Stats)-1)%/%10))
Stats_Test$match_index <- c(0, rep(1:(nrow(Stats)-1)%/%10))

#Create a new column for aggregate KDA
Stats$AFK <- Stats$kills + Stats$deaths + Stats$assists

#Find those who did not afk in the game, or who had a aggregrate KDA score of > 0 (cant be negative)
Stats_Test <- Stats[!(Stats$AFK == 0),]

#Delete all matches which don't have 10 participants
Stats_Final <- Stats_Test[Stats_Test$match_index %in% names(which(table(Stats_Test$match_index) > 9))
```

Figure 1

```
Call:
glm(formula = Win ~ NumFighter + NumMarksman + NumController +
    NumMage + NumSlayer + NumTank, family = quasibinomial, data = Overview.train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.262  -1.183  -1.079   1.171   1.341

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.09080    0.08907  -1.019  0.30802
NumFighter     0.05562    0.02102   2.646  0.00815 **
NumMarksman   -0.07989    0.02261  -3.534  0.00041 ***
NumController  0.06024    0.02094   2.877  0.00402 **
NumMage        0.03467    0.02054   1.688  0.09149 .
NumSlayer     -0.01155    0.02299  -0.502  0.61543
NumTank        0.03707    0.02232   1.661  0.09677 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasibinomial family taken to be 1.00005)

    Null deviance: 134372  on 96930  degrees of freedom
Residual deviance: 134311  on 96924  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 3
```

Figure 2

```
> stargazer(Overview, type="text", median=TRUE)

================================================================================
Statistic        N       Mean     St. Dev.  Min Pctl(25)  Median  Pctl(75)    Max
--------------------------------------------------------------------------------
Team          193,862 96,930.500 55,963.280  0  48,465.2 96,930.5 145,395.8 193,861
Win           193,862   0.500      0.500     0     0       0.6        1        1
NumFighter    193,862   1.226      0.719     0     1        1         2        4
NumMarksman   193,862   0.985      0.328     0     1        1         1        4
NumController 193,862   0.833      0.562     0     0        1         1        4
NumMage       193,862   1.140      0.696     0     1        1         2        4
NumSlayer     193,862   0.305      0.514     0     0        0         1        3
NumTank       193,862   0.373      0.564     0     0        0         1        4
--------------------------------------------------------------------------------
```

Figure 3

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.72960    0.02703  26.992  < 2e-16 ***
teamVS      -0.03673    0.02759  -1.331  0.18308
teamTDC      0.06961    0.04791   1.453  0.14626
teamTDT     -0.21157    0.04745  -4.459 8.24e-06 ***
VS_TDC      -0.12789    0.04265  -2.998  0.00271 **
VS_TDT       0.11468    0.04247   2.700  0.00693 **
TDT_TDC      0.38443    0.01711  22.471  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.21 on 193853 degrees of freedom
Multiple R-squared:  0.003248,  Adjusted R-squared:  0.003217
F-statistic: 105.3 on 6 and 193853 DF,  p-value: < 2.2e-16
```

Figure 4