Applying Machine Learning to E-Sports

Ritwik Katiyar and John Chmielowiec

Abstract: The popularity of electronic sports has steadily grown in recent years. E-sports can provide financial and social opportunities to players and team owners. The games played in e-sports are usually highly accessible to fans. As the popularity of e-sports has grown, so too has the amount of data available on the games played. This paper explores some of that data in regards to a specific game, League of Legends. Can interesting observations about the current state of the game be drawn from the data? Yes. Specifically, combinations of heroes frequently picked together can be observed. Can the data highlight ways in which players can improve their game? Yes. The data is able to show which features contribute to widely used performance metrics and players can take actions to improve their performance in the contributing features and the final metric. Can the winner of future matches be predicted based on their recorded game data? Partially. The dataset examined here is a good start to predicting the winner of a match, but more data is needed to successfully model a game as complex as League of Legends. This paper shows evidence of how data analytics will play a significant role in future professional e-sports competitions.

Keywords: e-sports; apriori; correlation matrix; elastic net, lasso, logistic regression, random forest, neural network

1. Introduction

Electronic sports, better known as e-sports, are competitions between players using video games as the test of skill. Many aspects of e-sports are borrowed from traditional sports competitions, including monetary rewards for winning competitions and spectating of games by fans.

Currently in e-sports, one of the most highly competitive games is League of Legends. League of Legends (LoL) is defined as a MOBA (Multiplayer Online Battle Arena.) This type of game consists of two teams of five players that share a common objective of destroying the opposing team's base. Each player controls one character, here called a champion, with abilities that no other character possesses. During the game, players will become more powerful by earning experience to improve the character's abilities and by purchasing items with a resource called gold. The most common way to earn both experience and gold is through direct conflict with the opposing team, resulting in a highly interactive game. The game has a character pool consisting of one hundred sixty-three champions and over one hundred fifty different items. The huge amount of tactical and strategic depth present forms a complex game with a high skill ceiling for the players. The e-sport format for League of Legends is similar to that of many sports leagues. More information about the game can be found at the link provided in the appendix. (Riot)

League's format divides each region of the world into servers, and each server maintains its own league and teams. The teams compete against one another throughout the season or split. A split refers to a schedule format implemented in a variety of sports leagues. In a split, the season is divided into two parts, with the winners of both halves playing each other at the end for the championship and to gain the regional title. Once a team attains a regional title then, they qualify for a worldwide tournament involving other winning teams from other regions. This purpose of this tournament is to decide which team is the best in the world. Professional League of Legends competition consists of many different regions and each region is ranked according to its representative team's performance at prior worlds events. Currently, the Tier 1 regions are North America, China, Korea, and Europe. There are many smaller regions such as Brazil, Tukey, Southeast Asia, Australia, etc. Winning the world's tournament can prove profitable for the teams and players. The Worlds series draws in millions of viewers each year and a victory at Worlds can earn the team a grand prize of $2.25 million, brand deals, and the support of fans. (Iyer)

The last few decades has seen the increased use of analytics in traditional sports. It is not surprising that various e-sports teams have started adopting the use of analytics. Professional League of Legends teams are one such group. A team's analytics and coaching staff work in conjunction to assist with player improvement and growth. This report aims to mimic the responsibilities of a data scientist that may be employed by one of the top League of Legends teams with the goal of optimizing player performance as well as predicting the odds of winning using various machine learning techniques.

Three different topics will be explored in this effort.

First, drafting champions before a game in League of Legends is an important aspect of determining whether a team will win or lose. Before a game begins, the players get an opportunity to pick the champions they will be playing and ban champions they do not wish to play against. This is a crucial part of the strategy in the game. Banning or picking away a champion an opposing player is skilled at can result in putting the opposing team at a disadvantage. Similarly, giving a player access to a champion they are skilled at playing by picking said champion can make the difference between a win or a loss. To make sure players are practicing the right champions and combinations of champions, it is pertinent to know what champions work well with one another. The data from the most recent split will be analyzed to gain insight as to what champions players are currently playing the most and what champions are being paired together to generate a game-winning team composition. Once players understand the dynamics of picking and banning champions, they can be focused on in-game performance.

Secondly, the dataset may be able to reveal what factors are contributing to performance metrics. One common performance metric that is used is Kills/Death/Assists (KDA.) KDA is similar to the number of goals scored, penalties taken, and assists given by a player in hockey. It simply sums the number of kills and assists a player achieved and divides that number by amounts of deaths a player had. If a player never died, the death count is set to one to avoid dividing by zero. The features of the dataset will be examined to see which have the most impact in predicting a player's KDA. If these features can be discovered, players could be coached in ways to adjust their play to improve their KDA and consequently give them a better chance at winning their matches.

The final goal of this analysis is to predict a match's winner before the game has even started. Developing a complex model that can make an accurate prediction of a winner can take months or years of work. The attempt detailed below will simply be to scratch the surface of this problem. Specifically, a model will be generated to predict the likelihood that a given team will win any match and another model will be used to predict the outcome of a matchup between two specified teams.

2. Dataset and Methods

The dataset for this project was acquired from https://oracleselixir.com. The website offers information on every game played during a season of competition. The data set contains:

- A summary of information for every match played, including data on which champions are played during the match and which champions the players have banned from participating in that match.
- Metrics for every individual player's performance within a given match.
- A team's overall performance stats.

The data set contains 123 features and 72,553 observations. Data from the most recent season's split will be used to generate models. (Data)

A Quick Summary of the data:

Gameid	**datacompleteness**	**league**	**playoffs**
ESPORTSTMNT06_2753012	complete	LFL2	0
ESPORTSTMNT06_2753012	complete	LFL2	0
ESPORTSTMNT06_2753012	complete	LFL2	0
ESPORTSTMNT06_2753012	complete	LFL2	0
ESPORTSTMNT06_2753012	complete	LFL2	0

Date	**game**	**participantid**	**side**	**position**	**playername**	**...**
2023-01-10 17:07:16	1	1	Blue	top	Wylenz	...
2023-01-10 17:07:16	1	2	Blue	jng	Julbu	...
2023-01-10 17:07:16	1	3	Blue	mid	Sintax	...
2023-01-10 17:07:16	1	4	Blue	bot	Axelent	...
2023-01-10 17:07:16	1	5	Blue	sup	Wixo	...

opp_csat15	golddiffat15	xpdiffat15	csdiffat15	killsat15	assistsat15
131.0	322.0	263.0	12.0	0.0	0.0
117.0	-357.0	-1323.0	-43.0	0.0	0.0
162.0	-479.0	-324.0	-26.0	0.0	0.0
122.0	200.0	292.0	20.0	0.0	0.0
3.0	-216.0	-579.0	0.0	0.0	0.0

deathsat15	opp_killsat15	opp_assistsat15	opp_deathsat15
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	1.0	0.0	0.0
1.0	0.0	1.0	0.0

Before any questions could be answered, steps needed to be taken to ensure the dataset did not contain any missing values or errors that would cause issues later. The first step was to separate the data into team data and player data. Next, missing values were searched for. Several missing values were found in the data associated with the League of Legends Professional League (LPL) and the League of Legends Development League (LDL). Due to issues with the League of Legends API, the website Oracle-elixir was unable to obtain complete data from the aforementioned leagues. Some of the planned analysis relied upon just a subset of the data (e.g. the choice of selected champion by each player.) Given this, the player data was further separated into complete player data and partial player data. Doing so enabled the use of the full dataset for certain parts of the analysis. The partial dataset was used for other parts of the analysis. Finally, a few instances of unusual data was found within the series of features related to champion bans. The seemingly duplicate ban error only occurred for two games, so those two games were removed from the dataset.

Since there are multiple questions to explore, the project has been segmented into parts where each part focuses on a specific question.

Part 1: Associations between champions played.

The first investigation to be done is looking at the associations between champions played. What champions are often picked together? What is the most popular team composition so far? The Apriori algorithm, which applies association rules and is an unsupervised algorithm, was utilized to receive the answers.

Association rules can also be referred to as, “market basket analysis”, which looks for associative patterns based on frequency within a dataset. This method was first applied in a commercial analysis of market products which aimed to look at the frequency of items that are bought together.

Apriori, and by extension the association rules, require that the data be one hot encoded; This means that additional “dummy” variables need to be created. A variable is created for every value in a feature. If the original value matches one of the new variables, its value is set to 1. All other dummy variables are set to a value of 0. This procedure was performed on the champion feature.

Once the data has been transformed, association rules are then applied. Where Champions = {champion 1, champion 2, … champion k} and where all distinct champions are involved or picked together in the game and where cardinality K = |Champion| is the total number of champions. An association is then created where A → B such that A, B are a subset of champions and are disjointed.

Some of the rules that will be used to look for interactions are as follows:

- Support: Support is represented as (A → B) = Pr (A ∩ B) and is a measure of the strength of association between two events.
- Confidence: Confidence is represented as (A → B) = Pr (B|A) = Pr (A ∩ B)/ Pr(A) and functions as a measure of the reliability of the rule (i.e., does A give B regardless of frequency)
- Lift: represented as (A → B) = Pr (B|A)/Pr (B) = Pr (A ∩ B) / Pr (A) Pr (B) which functions as a measure of association between two item sets. Meaning values greater than one indicate a positive correlation, values less than one indicated a negative correlation, and values of zero represent no correlation.
- Conviction: Represented as (A ∪ B) = [1 – Pr (B)] / [1 – Pr (B|A)] = Pr (BC)Pr (A) / Pr (BC ∩ A) serves as a measure of the implication of strength of the rule from statistical independence. Conviction of 1 implies A and B are independent.

Part 2: Features that affect Player KDA.

For this problem, the features that best explain KDA will be searched for. Since KDA is a continuous variable, regression algorithms will be applied in an attempt to determine which features most affect the KDA of a player. Identifying and interpreting such features can help focus the player in improving specific areas which will ultimately aid in improving the player's performance.

The algorithm that will be applied first is the Elastic Net regression algorithm. This algorithm is generally considered to be a middle ground between Ridge and Lasso models. The main reason for implementing Elastic Net as the first algorithm is that there is a strong possibility that some of the features may be strongly co-related. Elastic net mathematically is an improvement upon the regular residual square of sums (liner regression) algorithm. It improves upon it by adding a net cost function that, based on the tuning parameter, can make the algorithm function as Lasso or Ridge or a mix between the two. Cross-validation can also used to try and determine the best turning parameters.

$$J(\theta) = MSE(\theta) + r\alpha \sum_{i=1}^{n} |\theta| + \frac{1-r}{2}\alpha \sum_{i=1}^{n} \theta_i^2$$

Equation 1: The mathematical representation of the Elastic Net algorithm. (Geron)

Lasso would be an ideal choice to use after Elastic Net since the goal is to interpret the results and try to find how KDA relates to various other player performance metrics and features within the dataset. Lasso and Ridge are also an improvement upon the liner regression algorithm. Ideally, the use of Ridge regression would be avoided. The primary reason for this is that Lasso is far easier to interpret since Lasso employs the use of a shrinkage penalty parameter that rewards coefficients close to zero. Lasso can effectively perform the variable selection by effectively shrinking the parameters to zero. Ridge, however, would only shrink the coefficients closer to zero and not force any of them to be equal to zero. This is harder to interpret since it can be difficult to know if a feature is important or not. (Geron)

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right) + \lambda\sum_{j=1}^{p}\beta_j^2 \qquad\qquad \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right) + \lambda\sum_{j=1}^{p}|\beta_j|$$

Equation 2: The Equation on the right shows the Lasso equation compared to the Ridge equation on the left. It can be seen that the two algorithms are similar. However, one has a bonus of rewarding coefficients closer to zero. (Geron)

Part 3: win/loss prediction and features that determine win/loss.

Random forest will be used to accomplish this task by first generating a feature importance plot. Random Forest is an algorithm that itself is a collection of decision trees. A decision tree is non-parametric supervised learning algorithm that is better suited for classification tasks. The decision tree algorithm picks a feature that reduces the gini-score of the model the most. The Gini- score is defined as:

$$G_i = 1 - \sum_{k-1}^{n} P_{i,k}^2$$

Where Pi, k is the ratio of class k instances among the training instances in the ith node. For simplicity, a feature that reduces the gini score the most becomes the primary node and splits the tree. This process then continues until a desired threshold or until the gini-score is zero and a pure node is acquired.

Random forest builds upon this idea except it looks for an optimal feature in a random subset of features to build numerous decision trees. It then compiles the results together. These results can be used to try and see which features among the 123 features are the most important in determining if the team won or lost the game. Only the features that appeared important are used to prevent overloading the models with unnecessary features. This reduces computation time.

Once this plot has been generated using Random Forest, a threshold is applied to select important variables. Logistic regression is then applied to the selected features to predict the probability of a team winning. This probability is used to try and predict if one team would win against another team. In this comparison, the team with a higher probability based on their past performance wins.

Logistic regression uses the maximum likelihood method to find the probability of a particular classification task. The logistic function produces an S-shaped curve also known as the sigmoid curve that lies between zero and one. Mathematically, logistic regression looks like this:

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

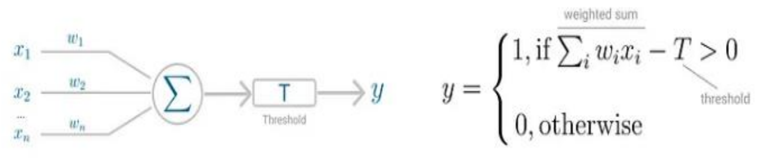
The equation can also be rewritten to display the log odds (the odds of a particular event occurring.) Values closer to 1 predict a win, and values closer to 0 predict a loss. Since there are multiple variables, the formula would have added coefficients up to the p (p is the number of variables that are selected for the model).

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 \dots \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 \dots \beta_p X_p}}$$

Once the model is complete, it is possible to try to predict the odds of a team winning a game based on their average performance in the past series of games. However, log odds will not be able to account for factors such as who the opposing team is or if the probability should go down or up based on the players currently playing in the team. To try to account for this complexity a neural network will be utilized. (Gareth)

A standard Multi-layer Perceptron classifier was chosen for this purpose. A neural network attempts to mimic the human brain's neural network. Where a stimulus activates a receptor which can then activate a neural network; this in turn fires off an effector which results in a response. This process can bounce between receptors until a proper response is generated.

A perceptron is widely considered the simplest artificial neural network (ANN). To put it simply, the input for the perceptron gets combined in a weighted sum, and if the weighted sum reaches a threshold, then the perception produces a result. This threshold is often referred to as an activation function or step function. This step function then assigns the output value based on what the step function is.



$$y = \begin{cases} 1, \text{if } \underbrace{\sum_i w_i x_i}_{\text{weighted sum}} - \underbrace{T}_{\text{threshold}} > 0 \\ 0, \text{otherwise} \end{cases}$$

Image 1: The image shows the basic functionality of a perceptron. (Siddiqui)

Among the various step functions available, the sigmoid (logistic) and tanh functions were selected for use. Sigmoid function focused is a logistic function that selects numbers between 0 to 1 the basic formula for this step function is as follows:

$$\boldsymbol{g(z) = \frac{1}{1 + e^{-z}}}$$

The graph produced by such a function represents an s shape. By comparison, the tanh function works similarly however, instead of the values ranging from 0 to 1 the values now range from 1 to -1. The formula can be represented as:

$$\boldsymbol{g(z) = \frac{2}{1 + e^{-2z}}}$$

This activation function also produces an s shape like the sigmoid function. These functions were selected since the task is to classify a result where the value can be 0 or 1. (Or 1 and -1, with 0 or negative values representing a loss). Adding to this is the concept a multi-layer perceptron. A multi-layer perceptron is simply multiple layers comprised of multiple perceptrons which are all located between the input and output layers.

Finally, the multilayer perception uses backpropagation (going through the algorithm once forward and then backward) gradient descent. This means it first makes a prediction and then measures the error. It proceeds to go through each layer in reverse to measure the error and tweak the connection weights to reduce error. 5. (Siddiqui)

3. Results and Discussion

3.1 Exploratory Data Analysis

This purpose of this section is to highlight the various data analyses and visualizations done before any attempt to answer key questions was done. Please refer to Appendix A for various plots and data visualizations and explorations that provide extra insight into the data.

3.2. Part-1: Associations between champions played

3.2.1: Exploratory Data Analysis (EDA):

Before the application of the Apriori algorithm, the champions were sorted by the number of times each champion was played. The plot below showcases the top 5 most frequently played champions and the 5 least frequently played champions.

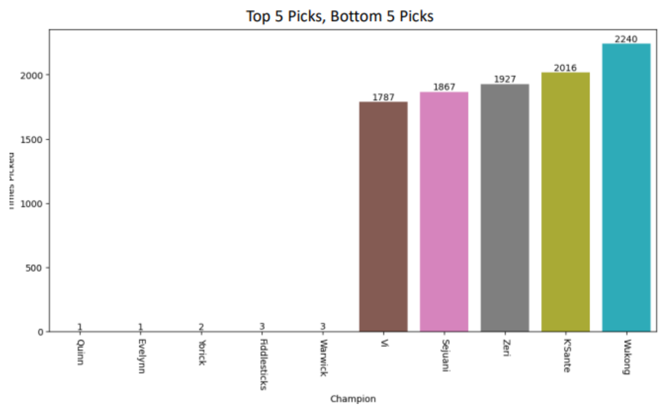


Image 2: Shows the 5 champions picked the least on the left and 5 champions picked the most in all the games within the dataset.

3.2.2 Support (Threshold 0.05)

A key thing to note based on the exploratory data analysis is that Zeri was observed as the champion that appeared with the highest support and association threshold. The expected results were to show relations with Wukong, K'sante, Vi, etc. Initially the results were sorted by the support and with the goal of getting values with high support. This yielded the 3 bidirectional pairings made by apriori meaning these champion pairings appeared most frequently among all the games. Nami-Lucian was higher than Lulu-Zeri followed by Rakan Xayah. However, this only gives a limited number of pairings. To look for more associations, it was decided to lower the support threshold to 0.025 and see if more interactions could be found. (For more details, refer to Table B1 in the apendix.)

3.2.3 Lift and Confidence (Threshold 0.025)

Other metrics can be relied on after lowering the threshold to include results with lower support values. Confidence is a metric or a measure of the reliability of the rule (i.e. how much probability-wise was A explained by C. e.g. when Nami → Lucian.) Lift explains the strength of co-relation. Sorting by those metrics and lowering the support threshold to include even more sets of observations resulted in observed values that may not have high support but have good confidence and lift. It can be noted that, along with the previously observed sets, there are new interactions such as (Thresh, Aphelious), (Rakhan, Wukong) and (Jayce, Makoai). Only the first 4 observations seem to have relevant confidence levels. It can also be observed that some associations are finally being that that involve the most frequently played champions. However, the lift for all of the values is higher than 1. This suggests that they might be worthwhile relationships. (Refer to Table B2 in the appendix for more details.)

3.2.4 More Lift and Confidence (Threshold 0.01)

After observing some of the more common sets it was decided to look for hidden or unique interactions. To do this, the support threshold was lowered even more. Results were again sorted by confidence and lift. Sets can now be observed that have more than one value per set such as (Nami, Gnar) → (Lucian). Interestingly by using these interactions, it is possible to observe various other champions that pair well with Nami and Lucian such as Gnar, Azie, Sejuani, Maokai, etc. It was also possible to note some more unique pairs such as Lux → Caitlyn in addition to sets involving Lulu and Zeri along with the champions such as Maoki and Sejuani. It can be noted that the observations are valid since the confidence is over 0.6, and the lift values are far over 1. However, it should be taken into account that the support values are lower than the previous values that were observed. (Refer to table B3 in the apendix for more details.) It was also noted that a number of the interactions seem to contain Lucian and Nami. These two are the champions that occurred the most together. Another decision was made to limit the confidence to remove Lucian and Nami and focus on other interactions without the two specified champions. To do so, the confidence was limited from 0.81 to 0.2 and the results sorted by conviction values. (Refer to Table B3 in the appendix for further informtaion.)

3.2.5 Conviction Only (Threshold 0.01)

After restricting and sorting by conviction, far more interesting relations can be seen, mostly involving (Zeri → Lulu) and (Xayah → Rakan). This makes sense as they were the second most occurring combination of champions, after (Lucian → Nami). Interactions can be observed involving Zeri → Lulu and champions like Maoki, K'sante, Sejuani, Vi, etc similar to the obseratons of the same champions appeared with Lucian and Nami. However, there are some unique interactions that can be noted, such as (Heimerdinger → Varus) and (Karma → Ezreal). These interactions are also valid since the conviction and lift values are higher than 1 and are considered to be incresting relations. (Refer to Table B4 in the appendix for further information.)

3.2.6 Lift Only (Threshold 0.01)

In the previous test, the sorting of rules had been done by Conviction only. The final test was to sort by only lift to look for any relevant interactions that may have been missed. However, no new interactions were able to be observed. It was notable that Caitlyn Lux and Ezreal Karma had uniquely high lift values. Since lift is a measure of association between two items, it can be assumed that Ezreal → Karma and Cailyn → Lux have a strong association with one another compared to the other values within the table. (Refer to Table B5 in the appendix for additional information.)

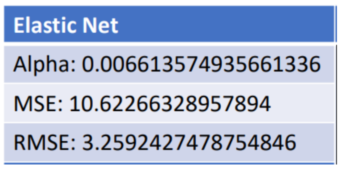
3.3. Part-2: Features that affect Player KDA.

3.3.1 Exploratory Data Analysis

Since there are a lot of features it can be difficult to individually look at every single feature and analyze the details. Instead, it was decded to look at the features that pertain to KDA (the target variable) and each other. (Refer to the Correlation matrix provided in the appendix section C: Image C1) Initially, this matrix showed how correlated the data is and how there seem to be several redundant variables within the dataset. An example of redundant variables is, wards placed, wards killed, and control wards bought. These variables can be explained by vision score (which is a score calculated using the mentioned variables.) Similarly, total gold, earned gold, earned gold share, gold spent, minion kills, and monster kills can all be explained by cspm (creep score per minute) and earned gold. After eliminating these variables, a new correlation matrix was constructed which still contained several high correlations between variables. However, many of the higher correlations have been removed at this point. (Refer to Image C2 in the Appendix.) The relations that KDA has with the features can not be observed. Some of the positive correlations are result, and team kpm (kills per minute) which seem to have the highest positive correlation (67% and 64%). Some negative correlations are opp_baron (opponent obtaining the baron buff), damagetakenperminute (damage taken per minute), and playing the top lane position (-18,-19, and -10 percent respectively.) Multiple models were examined to see if clearer results than the correlation matrix could offered.

3.3.1 Elastic Net

One again the Elastic Net was chosen as the first choice because when the varaibles are strongly corelated it is better to run the Elastic Net model. Before the Elastic Net model is chosen, the alpha value needed to determined. This was done by running cross validation.



Elastic Net
Alpha: 0.006613574935661336
MSE: 10.62266328957894
RMSE: 3.2592427478754846

Table 3.3.1.1: This table shows the alpha value almost close to zero and that the Mean squared error (MSE) as well as root mean squared error (RMSE) is

relativly high. The R-squared for this value was also 0.571 suggesting that only about 57% of the varaince was being explained.

Based on the above results, it was decided to apply Lasso since the alpha value was so close to zero. It was thought that perhaps better results could be obtained by simply running Lasso. The root mean squared error (RMSE) was also examined because the root mean squared error informs how much the residuals differ from the actual score on average. This tells a little more detail about the performance or accuracy of the model compared to the mean squared error.

3.3.2 Lasso

Lasso (1st Model)
Alpha:0.00330678
MSE: 10.61955
RMSE: 3.25924

Table 3.3.2.2: This table shows the alpha value of the first lasso model based on the cross validation as well as the mean squared error and the root mean squared error.

It can noted that the mean squared error, the root mean squared error, and the root mean squared error have all decreased a bit. However, the error rate is still a little high. Since Lasso is outperforming the Elastic Net model, it was decided to fit the whole data set with Lasso to further analyze the results.

3.3.3 Lasso – Whole Dataset

Lasso (2nd Model)
Alpha: 0.0033067
MSE: 10.45945
RMSE: 3.234107
R-squared: 0.571

Table 3.3.3.3: The MSE and RMSE have reduced even more once the complete data set was run.

Once again, it is important to note that these values are still high. The R-squared value for Lasso is observerd to be essentially the same as the Elastic net model. Only 57% of the variance is explained. However, for basic interpretation, this is okay. At the very least, the R-squared results suggest the model is at least performing better than 50%. The next task is to look at the coefficients generated by the model to observe if any new details were able to be discovered that were not explained by the co-relation matrix.

gamelength	0.016469
result	1.514470
firstbloodkill	0.028964
firstbloodassist	0.184970
firstbloodvictim	-0.329507
team kpm	1.233766
barons	0.082316
opp_barons	-0.075856
inhibitors	0.012062
dpm	0.156015
damagetakenperminute	-1.037783
visionscore	0.062009
earned gpm	1.795120
cspm	-0.830665
golddiffat10	0.077308
position_bot	-0.346279
position_jng	0.604199
position_mid	-0.000000
position_sup	0.300962
position_top	-0.068717

Table 3.3.3.4: Shows the coeficents produced by the lasso model.

Looking at the coefficients, one thing that can be observed is that the earned gpm (gold per minute) and damage taken per minute features have a strong impact on determining KDA. The correlation matrix has observable values of 38% and -19% for gpm and damage taken per minute. Even though -19% is the strongest negative correlation, there are far more positive correlations that the model was expected to favor. Moreover, players playing the support as well as jungle roles seem to have a strong positive impact (0.604 and 0.3 respectively) on the game while players playing the bottom (bot) position have a significantly negative impact. With both top and middle (mid) positions playing a negligible role, it was suspected that support and jungle are unique roles that work far differently than the traditional roles of mid, bot, and top. Support and jungle positions were removed from the features to see if different results would occur. This yielded the following results:

3.3.4 Removed Support and jungle Players

Lasso (2nd Model)
Alpha: 0.0033067
MSE: 9.6188348
RMSE: 3.101424
R-Squared: 0.585

Table 3.3.4.5 With the jungle and support players removed, themodel is able to perform a lot better with an increase in R-squared, MSE, and RMSE scores.

It is hard to determine what could have contributed to the improvement of the score. One hypothesis is that the support and jungle players play the game differently enough from the top, middle, and bottom players. The difference in the way the positions are played resulted in reducing the performance of the model overall. The observed coefficients should show any differences from the previous model.

gamelength	-0.084609
result	1.515701
firstbloodkill	0.042931
firstbloodassist	0.171845
firstbloodvictim	-0.320791
team kpm	1.173139
barons	0.083743
opp_barons	-0.034667

inhibitors	0.016006
dpm	0.256964
damagetakenperminute	-0.695696
visionscore	0.196611
earned gpm	1.136409
cspm	-0.181922
golddiffat10	0.029694
position_bot	-0.221354
position_mid	0.131389
position_top	-0.000000

Table 3.3.4.6: shows the coeffiencts and the values that the model placed for each variable when it comes to determining the KDA of a player.

Some of the key changes that are observed are the fact that the position top has now been set to have no influence of KDA (0.000) compared to the previous model's value which indicated the mid position as not an influence to the model (0.000). Moreover, the values for damage taken per minute have significantly been reduced (-1.03 to -0.69.) The length of the game now has a negative correlation (-0.08) compared to the previous model that suggested a slight positive correlation (0.016). Vision score increased from 0.06 to 0.19. This was surprising since traditionally support and jungle roles are responsible for vision control. However, the fact that the value of the vision score went up suggests that players with high KDA generally have a good vision score. Players can be guided to improve their KDA metric with more confidence with the observed changes.

3.4 Part-3: : win/loss prediction and features that determine win loss.

3.4.1 Exploratory Data Analysis

This graph looks at win differences between the red and the blue side.

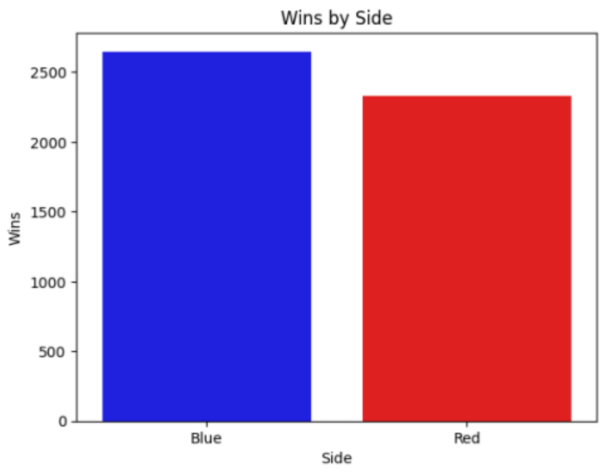


Image 2: Shows the number of games that resulted in a winning team being blue side vs red side.

Even though the difference between the red and blue sides is minor there is a possibility that minor errors could occur since the data isn't perfectly balanced. This is evidence for the standardization of the data. There is even more exploratory analysis provided in section D of the appendix.

3.4.2 Random Forest

Once some of the early data analysis was complete, random forest was used to reduce several features that do not contribute significantly to determining the win or loss of a team. To do, the threshold was set to 0.02 as it provided the least number of variables without removing any variable that may be extremely important to the game and determining winners. The full list of variables and their importance impurity score is in the Appendix D section.

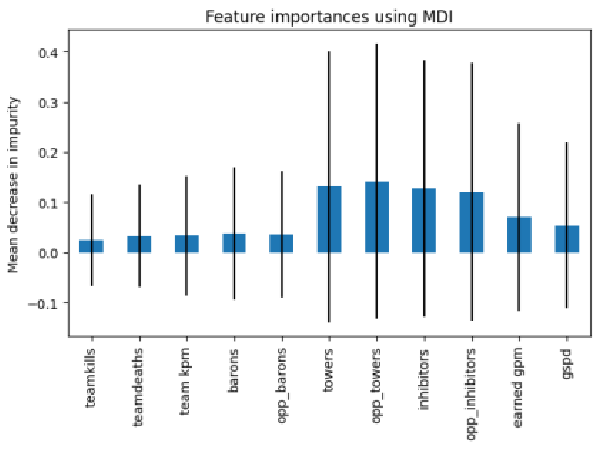
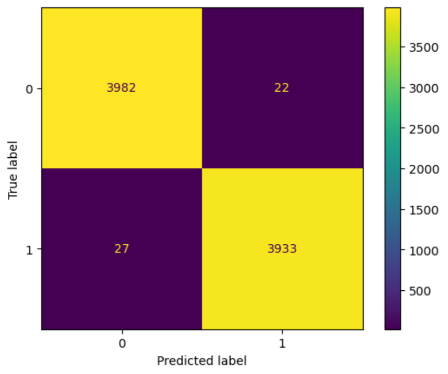


Image 3: Random Forest importance plot results with the threshold set to 0.02. The purpose of this is get variables that have a high impact.

3.4.3 Logistic Regression

After running logistic regression, a confusion matrix was constructed and roc plotted to evaluate the performance of the model.



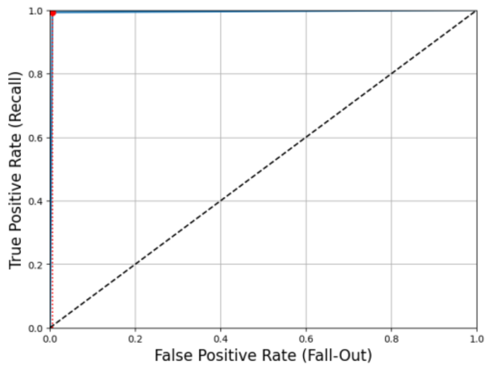
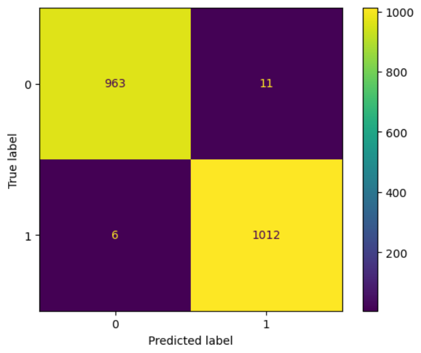


Image 4 and 5: confusion matrix for predicting the results of the training set, observe that the model has performed extremely well. Accuracy score is almost a 1.

Since the results looked too good, the possibility that the model may be overfitted was high. However, the test set produced the following confusion matrix and ROC plot.



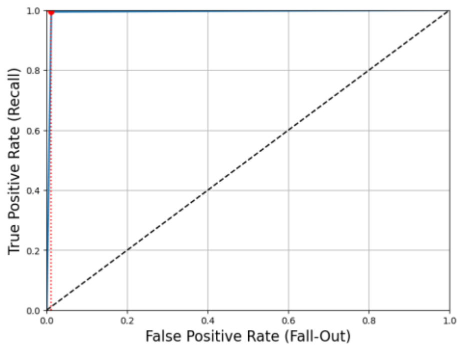


Image 6 and 7: The images show the confusion matrix and the roc plot for the test set.

This confusion matrix is roughly as good as the confusion matrix provided by the training set predictions. The accuracy for both the training and test sets are below.

Train - Accuracy	0.99384
Test - Accuracy	0.9914

Table 3.4.3.1: Shows the accuracy scores between two models.

The model was overfit. However, the overall accuracy has only lowered by 0.002. This was good enough to move forward with the model and try some predictions.

Before any predictions are made, it is important to understand what the model has learned. The model has learned what team stats determine a winner. If the model is given the average state of a team and asked to predict, then the model would give the odds of the team winning based on their mean performance in the past. These odds can then used to judge which team would win a match. As an example, the model was given the average statistic for the top two teams in league (T1 and Gen.G.) The results were as follows:

Probability of win for Team 1 (Gen.G): 0.971753
Probability of win for Team 2 (T1): 0.998100

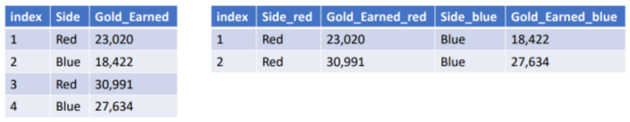
Since T1 has a higher probability or odds of winning, it can be said that T1 will win the game. However, this has certain drawbacks since the model doesn't understand that it is evaluating a game, nor can it account for variations since a predicted score for a team will remain the same. Additionally, it can be incredibly difficult for an upcoming team to stand out, since if the mean statistics of the team aren't good the model would determine the team's odds of winning to be poor, even though the team may have won the last few games.

3.4.4 Neural Network

To try to account for this complexity, a neural network was employed.

Initial tests with neural networks yielded poor results. Since the neural network was running with the same dataset used for logistic regression, it was only processing data for a single team at a time. The resulting model didn't understand that the game consisted of two teams per match.

To combat this, the data set was transformed such that the data for two teams were combined into one record. Effectively turning a long data set into a wide data set. The following image provides an example of one of the features (gold earned) and how it got transformed.



index	Side	Gold_Earned
1	Red	23,020
2	Blue	18,422
3	Red	30,991
4	Blue	27,634

index	Side_red	Gold_Earned_red	Side_blue	Gold_Earned_blue
1	Red	23,020	Blue	18,422
2	Red	30,991	Blue	27,634

Image 8: Shows the transformation of the data.

The neural network ran with the transformed data. The grid search cross-validation technique was utilized to find the best parameters for the model. The following parameters were determined to be the most optimal:

{'alpha': 0.001, 'hidden_layer_sizes': (500, 250), 'solver': 'adam'}

The alpha value is higher than the default or standard 1e-5 Which means that the cross-validation algorithm suggests that the model may be overfitting. The lead to the need to lower the alpha score to 0.001. The training and testing scores for the model were examined. They showed that the model was overfitting once again with the train score being 1.0 (100% accuracy) and the test score being 0.9959. However, the testing results weren't vastly different from the training score. An attempt was made to test some predictions by comparing them to data not in the initial dataset. The model was employed to predict the winner of a tournament. The tournament was a real tournament that occurred in April 2023 in the North American region. In this tournament, six teams played in a double-elimination format.

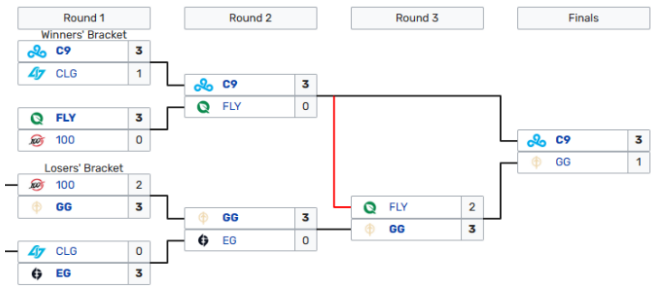
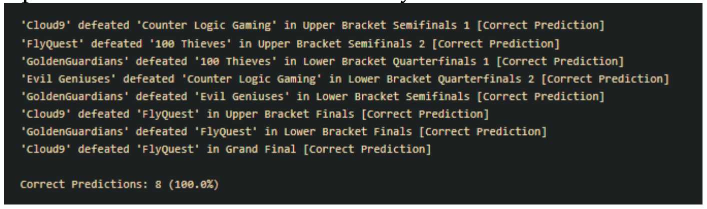


Image 9: Shows the actual results of the tournament that was used to test the second neural network.

Could this new model predict the winner of this tournament? How accurate would the prediction be? The model was able to predict all the outcomes correctly.



```
'Cloud9' defeated 'Counter Logic Gaming' in Upper Bracket Semifinals 1 [Correct Prediction]
'FlyQuest' defeated '100 Thieves' in Upper Bracket Semifinals 2 [Correct Prediction]
'GoldenGuardians' defeated '100 Thieves' in Lower Bracket Quarterfinals 1 [Correct Prediction]
'Evil Geniuses' defeated 'Counter Logic Gaming' in Lower Bracket Quarterfinals 2 [Correct Prediction]
'GoldenGuardians' defeated 'Evil Geniuses' in Lower Bracket Semifinals [Correct Prediction]
'Cloud9' defeated 'FlyQuest' in Upper Bracket Finals [Correct Prediction]
'GoldenGuardians' defeated 'FlyQuest' in Lower Bracket Finals [Correct Prediction]
'Cloud9' defeated 'FlyQuest' in Grand Final [Correct Prediction]

Correct Predictions: 8 (100.0%)
```

Image 10: Shows the model predicting the results correctly.

However, subsequent runs of the code show that the model was not predicting all the results perfectly every time. Due to the nature of the neural network algorithm, it can be difficult to understand how the algorithm learned and predicted the scores perfectly. In some of the other runs of the algorithm, the model couldn't maintain the 100% accuracy of the tournament predictions. Frequently, the accuracy would be 37.5%, 50%, or 62.5%.

One theory as to why the neural network has such a wide variety of accuracy is that too much or too little of the relevant data was being included or excluded from the training and test sets. This could result in the model being overfit or underfit. As this was just a single test, further testing and development of the neural network would be required to comfortably describe the efficacy of the model.

The model was also used to predict a winner between Gen.G and T1 as was done in the logistic regression test. The neural network's results were largely consistent with the logistic regression results and was able to predict T1 winning over Gen.G.

5. Conclusions

From the results of the testing performed for this report, the following conclusions can be drawn. The players have been playing champions like, (Lucian → Nami), (Zeri → Lulu), and (Xayah → Rakan). Apriori was able to uncover further unique interactions such as (Himerdinger → Varus), (Nami, Vi → Lucian), etc. Their results can provide a team with ways to accurately draft their team's champions for their players by making sure that the best champions are picked and even provide a framework for characters the enemy team may pick or a composition they would try and construct.

The KDA analysis indicates that players should focus on roaming the map, maintaining a good vision score as well as placing higher importance on generating gold via fighting and grouping with their team and not as much on passively generating a lead through a high creep score (via killing monsters or minions).

The importance plot generated from random forest highlighted some of the more important aspects of determining if a team would win or lose a game. Those features are the number of kills, barons, towers, inhibitors, earned gold, and average gold spent percentage difference (gspd). With the use of these features, the logistic regression was able to generate probabilities of a team winning or losing a game. However, those probabilities are limited and can be difficult for a team to change. If a team starts to perform better in the last 10 games the average of their 50 games would not be able to show a significant change in the performance making the predictions very static and resistant to change. The neural networks provided more flexibility. However, the models were unstable, and the accuracy of the model ranged from 37.5% to 100%. Through further testing, and possibly including various other features and parameters that may be available in other datasets, the overall performance of the model could be improved. Regardless, the algorithms used in the report provide a strong foundation and a framework for future research.

Author Contributions: Both authors contributed equally towards the success of the project.

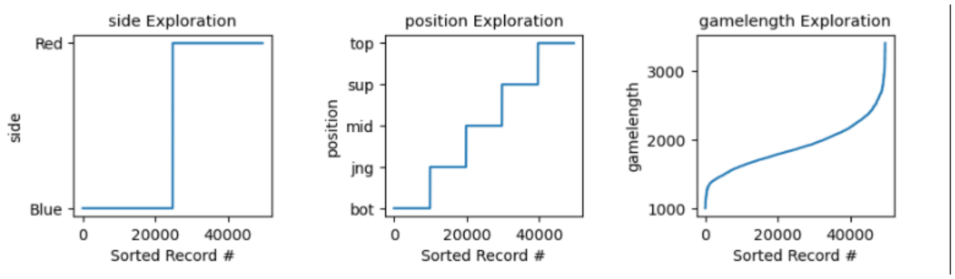
Data Availability Statement: The data was obtained from oracleselixir.com and is publicly available for all users.

Appendix A

What is League of Legends?

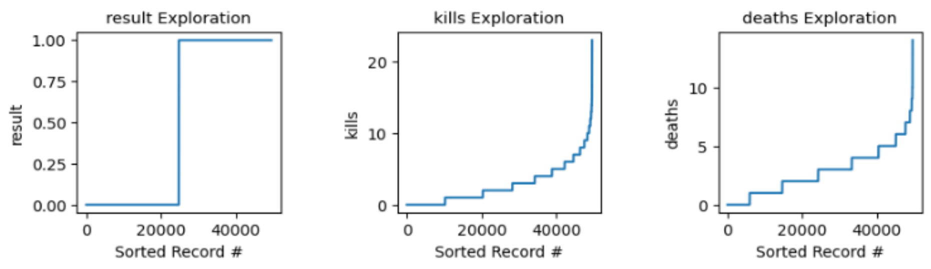
https://www.youtube.com/watch?v=BGtROJeMPeE&ab_channel=LeagueofLegends

Plot A1



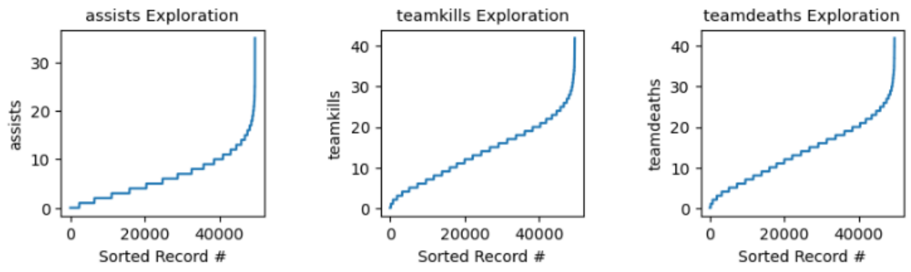
Plot A1: Looking at the plots from left to right, the first plot shows the distribution of red and blue team players within the data set, along with player positions and the game length. Note that the first two plots from the left are evenly divided among each field which is what is expected indicating that the data was successfully cleaned. Analyzing the game length, it can be noted that most of the data is centered around the 2000 second or (33 minutes) mark..

Plot A2



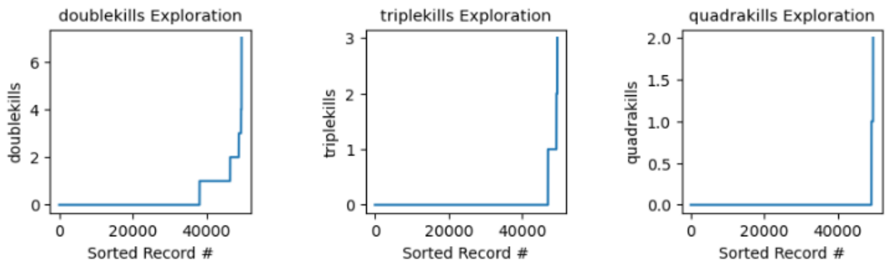
Plot A2: From left to right, note that the first plot shows the distribution of the results within the data set and once again the result looks evenly divided. It is observable that kills and deaths are not continuous variables. (Hence the steps) and can note that the number of kills is generally under 10 and deaths are under 5 suggesting the data is skewed.

Plot A3



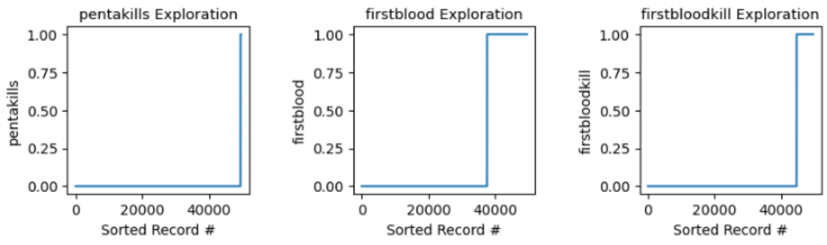
Plot A3: From left to right, observe that assists, teamkills and team deaths are all non-continues variables which is why there is a step-like plot. However, most of the variables are under 10 assists, and under 30 teamkills and team deaths.

Plot A4



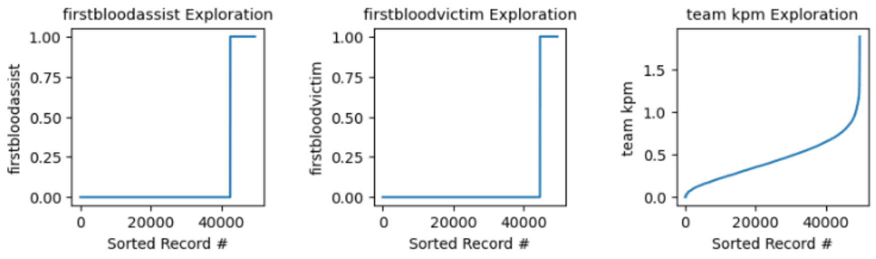
Plot A4: From left to right, observe that there are very few players that have multiple double kills or triple kills, and extremely few players even obtained quadra kills.

Plot A5



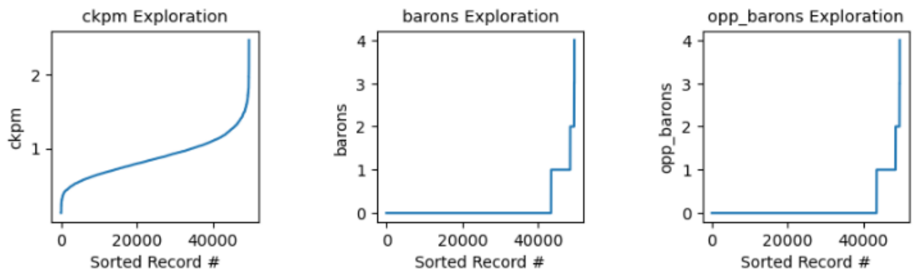
Plot A5: From left to right, observe that there are barely any penta kills that were obtained by any of the players. Interestingly, less than 1/9 of players got credit for a first blood kill. This makes sense since only 1 in 10 players in the game will be the player to get the first kill of the game.

Plot A6



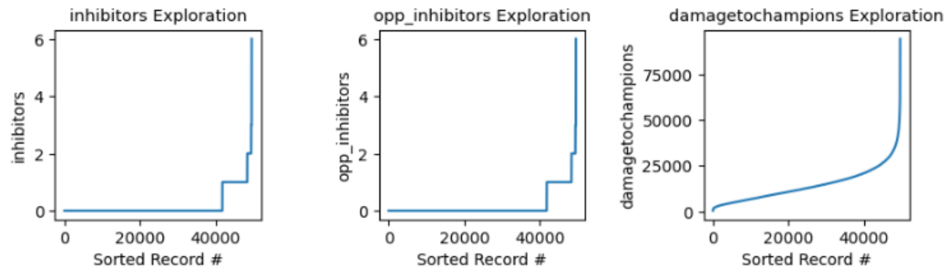
Plot A6: Note from left to right once again that only about 1/9 of the players earn first blood. Also observe the team kills per minute and note that very few teams ever went past 1.5 kills per minute during a game.

Plot A7



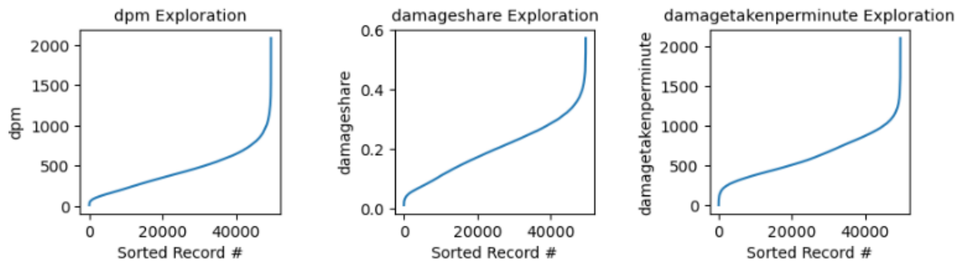
Plot A7 Note going left to right that Ckpm (Average combined kills per minute (team kills + opponent kills) is almost normally distributed as most of the values are around one ckpm. Also observable is that baron is an objective that isn't obtained every game, and there are only a few games where baron is an objective taken by a team.

Plot A8



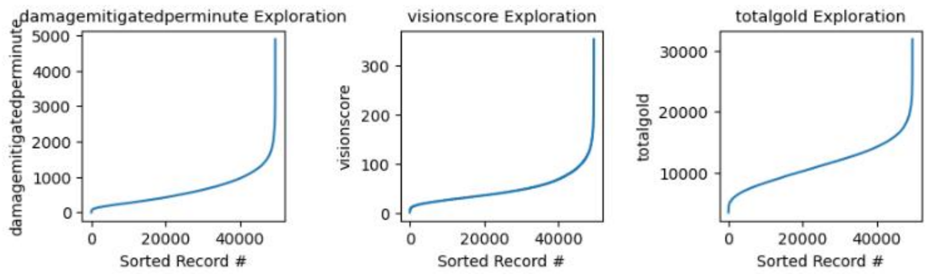
Plot A8: Going from left to right, note that very few games have 6 inhibitors obtained. Winning teams seem to obtain about 1 or 2 inhibitors per game. There are very few players that have an extremely high damage-to-champion ratio in their games. Generally, the values are under 25,000 damage. This is relatively low, but it makes sense since in professional leagues, one or two decisive fights generally determine the course of the game.

Plot A9



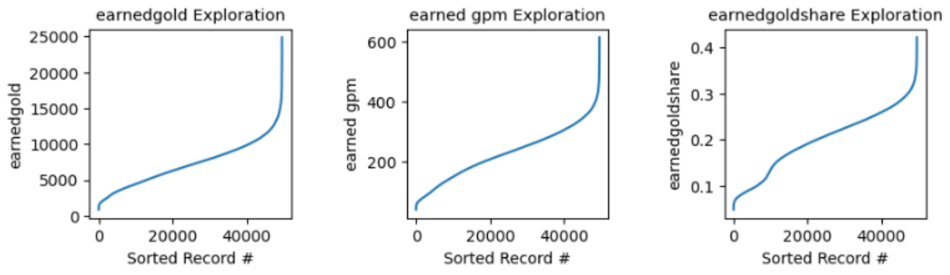
Plot A9: Note that all three variables look just about normally distributed with few instances of the extreme values. For example, DPM (damage per minute) and damage share seem to be a bit skewed. However, the number of variables that are outliers seem to be very few compared to the overall data set. Also note that otherwise all three variables here are normally distributed and don't have any odd missing values or extreme differences among them.

Plot A10



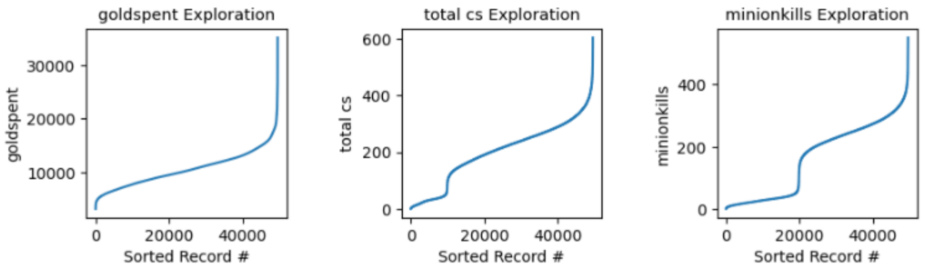
Plot A10: Going from left to right, observe that the majority of the values for damage mitigated per minute, vision score, and total gold have similar plots, where most values seem to lie under 2,000, 100, and 20,000 respectively. There are very few outliers at the high end.

Plot A11

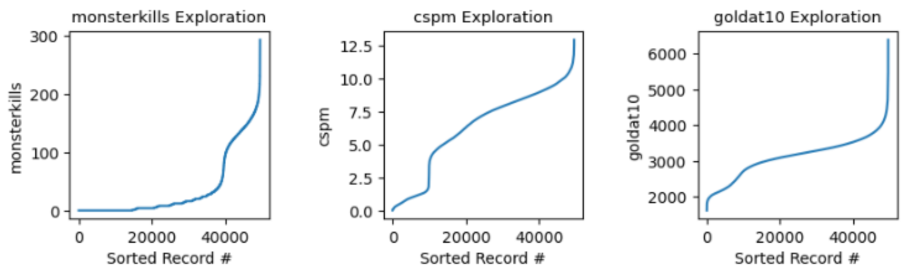


Plot A11: From left to right, observe that all the gold-related stats seem very similar to one another. Except for the earned gold share (which is a feature that explains the amount of gold compared to the other players in the team) which seem to have a dip early in the graph. Further analysis would need to be done to explain the slight abnormality.

Plot A12

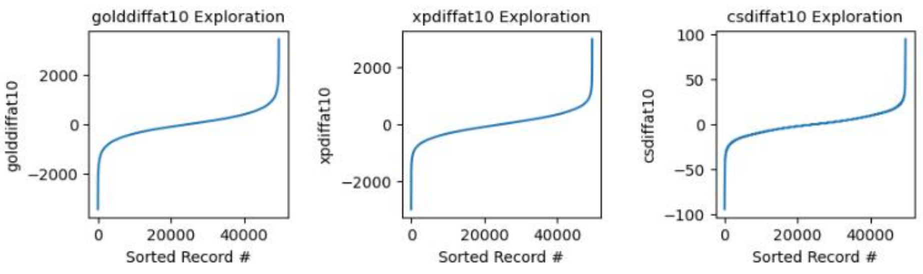


Plot A12: Observing from left to right, note that gold spent is once again like the plots in plot A11. However, notice an interesting dip in cs (creep score) and minion kills. This dip exists due to support players that don't generally get a lot of minion kills or cs score during the game. This suggests that in the later analysis, these players should be removed.



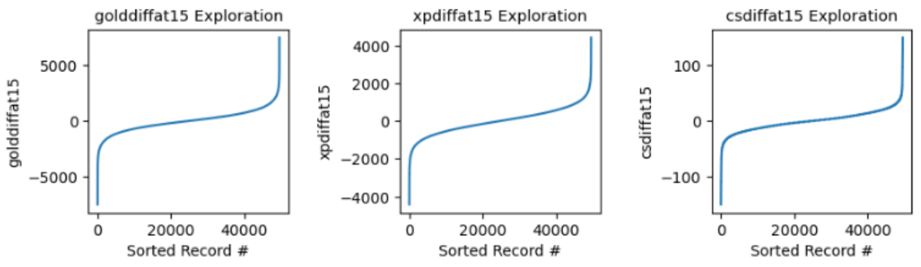
Plot A13: Observe here that the monster kills and cspm (creep score per minute) are two variables that have bizarre shapes. Again, the reason for this bizarre shape could be due to only jungle players farming monsters. cspm (creep score per minute) is bizarre, since as mentioned in plot A12, cs (creep score) is not earned by support players and that causes a dip in the plot.

Plot A14



Plot A14: Note that all of the values here are normally distributed where most of the values are between +2,000/-2,000, +2,000/-2,000, and +50/-50 respectively.

Plot A15



Plot A15: Shows once again a normally distributed variables where most values are between +5,000/-5,000, 2,000/-2,000 and 25/-25 respectively.

Appendix B

Table B1

	antecedents	consequents	support	confidence	lift	conviction
(Nami)	(Lucian)	0.093273	0.942949	9.732410	0.078915	0.989549

(Lucian)	(Nami)	0.096888	0.907772	9.732410	0.078915	0.993509
(Lulu)	(Zeri)	0.109137	0.600736	3.887804	0.048699	0.833781
(Rakan)	(Xayah)	0.118474	0.443220	3.678729	0.038236	0.826030
(Xayah)	(Rakan)	0.120482	0.435833	3.678729	0.038236	0.827916
(Zeri)	(Lulu)	0.154518	0.424301	3.887804	0.048699	0.878535

Table B1: Shows the Output for the first support threshold of 0.05. This is the max possible threshold that can be set for the dataset. The top three bidirectional combinations of (Nami, Lucian), (Lulu, Zeri), and (Xayah, Rakan) are noteworthy.

Table B2

antecedents	consequents	support	confidence	lift	leverage	conviction	zhangs_metric
(Nami)	(Lucian)	0.087952	0.942949	9.732410	0.078915	15.830028	0.989549
(Lucian)	(Nami)	0.087952	0.907772	9.732410	0.078915	9.831365	0.993509
(Lulu)	(Zeri)	0.065562	0.600736	3.887804	0.048699	2.117601	0.833781
(Thresh)	(Aphelios)	0.025703	0.504931	4.635127	0.020158	1.799879	0.826319
(Rakan)	(Xayah)	0.052510	0.443220	3.678729	0.038236	1.579652	0.826030
(Xayah)	(Rakan)	0.052510	0.435833	3.678729	0.038236	1.562528	0.827916
(Zeri)	(Lulu)	0.065562	0.424301	3.887804	0.048699	1.547448	0.878535
(Jayce)	(Maokai)	0.025904	0.271865	2.279273	0.014539	1.209560	0.620374
(Aphelios)	(Thresh)	0.025703	0.235945	4.635127	0.020158	1.242183	0.880134
(Rakan)	(Wukong)	0.026908	0.227119	1.257422	0.005509	1.060159	0.232236
(Xayah)	(Nautilus)	0.026406	0.219167	1.665065	0.010547	1.112111	0.454138
(Maokai)	(Jayce)	0.025904	0.217172	2.279273	0.014539	1.155705	0.637276
(Zeri)	(Wukong)	0.032028	0.207277	1.147573	0.004119	1.033625	0.152097
(Nautilus)	(Xayah)	0.026406	0.200610	1.665065	0.010547	1.100237	0.459967
(Zeri)	(K'Sante)	0.028112	0.181936	1.138245	0.003414	1.027011	0.143651
(Vi)	(Zeri)	0.026707	0.179245	1.160028	0.003684	1.030127	0.162105

(Wukong)	(Zeri)	0.032028	0.177321	1.147573	0.004119	1.027718	0.156943
(K'Sante)	(Zeri)	0.028112	0.175879	1.138245	0.003414	1.025920	0.144561
(Zeri)	(Vi)	0.026707	0.172840	1.160028	0.003684	1.028826	0.163164
(Zeri)	(Sejuani)	0.025904	0.167641	1.042915	0.001066	1.008288	0.048670
(Sejuani)	(Zeri)	0.025904	0.161149	1.042915	0.001066	1.007905	0.049031
(Wukong)	(Rakan)	0.026908	0.148972	1.257422	0.005509	1.035836	0.249850

Table B2: Shows the output for when the apriori threshold was set to 0.025 support values. These values were then sorted by confidence and lift more details on interpretation in results.

Table B3

antecedents	consequents	support	confidence	lift	leverage	conviction	zhangs_metric
(Nami, Gnar)	**(Lucian)**	**0.012048**	**0.960000**	**9.908394**	**0.010832**	**22.577811**	**0.910502**
(Azir, Nami)	**(Lucian)**	**0.010843**	**0.955752**	**9.864551**	**0.009744**	**20.410341**	**0.908939**
(Azir, Lucian)	**(Nami)**	**0.010843**	**0.947368**	**10.156932**	**0.009776**	**17.227811**	**0.911983**
(Sejuani, Nami)	**(Lucian)**	**0.013855**	**0.945205**	**9.755696**	**0.012435**	**16.481802**	**0.910848**
(Maokai, Lucian)	**(Nami)**	**0.011747**	**0.943548**	**10.115976**	**0.010586**	**16.062020**	**0.912507**
(Nami)	**(Lucian)**	**0.087952**	**0.942949**	**9.732410**	**0.078915**	**15.830028**	**0.989549**
(Vi, Nami)	**(Lucian)**	**0.016265**	**0.941860**	**9.721171**	**0.014592**	**15.533534**	**0.912897**
(Nami, Maokai)	**(Lucian)**	**0.011747**	**0.936000**	**9.660684**	**0.010531**	**14.111132**	**0.907882**
(Nami, Wukong)	**(Lucian)**	**0.015763**	**0.934524**	**9.645448**	**0.014129**	**13.792990**	**0.911702**
(Sejuani, Lucian)	**(Nami)**	**0.013855**	**0.932432**	**9.996800**	**0.012469**	**13.419558**	**0.913543**
(Nami, K'Sante)	**(Lucian)**	**0.016466**	**0.926554**	**9.563186**	**0.014744**	**12.296223**	**0.911633**
(Vi, Lucian)	**(Nami)**	**0.016265**	**0.920455**	**9.868382**	**0.014617**	**11.398853**	**0.914832**
(Gnar,	**(Nami)**	**0.012048**	**0.909091**	**9.746551**	**0.010812**	**9.973996**	**0.909453**

Lucian)							
(Lucian)	(Nami)	0.087952	0.907772	9.732410	0.078915	9.831365	0.993509
(K'Sante, Lucian)	(Nami)	0.016466	0.896175	9.608075	0.014752	8.733212	0.912690
(Wukong, Lucian)	(Nami)	0.015763	0.887006	9.509770	0.014105	8.024533	0.911035
(Lux)	(Caitlyn)	0.022088	0.817844	15.725338	0.020684	5.204282	0.962401
(Vi, Lulu)	(Zeri)	0.012450	0.673913	4.361387	0.009595	2.592811	0.785221
(Sejuani, Lulu)	(Zeri)	0.012651	0.656250	4.247076	0.009672	2.459584	0.779572
(Wukong, Lulu)	(Zeri)	0.012952	0.629268	4.072458	0.009771	2.280576	0.770303

Table B3: Shows the output for when the apriori support threshold was set to 0.01 support values. These values were then sorted by confidence and lift more details in the results.

Table B4

antecedents	consequents	support	confidence	lift	leverage	conviction	zhangs_metric
(Vi, Lulu)	(Zeri)	0.012450	0.673913	4.361387	0.009595	2.592811	0.785221
(Sejuani, Lulu)	(Zeri)	0.012651	0.656250	4.247076	0.009672	2.459584	0.779572
(Wukong, Lulu)	(Zeri)	0.012952	0.629268	4.072458	0.009771	2.280576	0.770303
(Lulu)	(Zeri)	0.065562	0.600736	3.887804	0.048699	2.117601	0.833781
(K'Sante, Lulu)	(Zeri)	0.011747	0.590909	3.824207	0.008675	2.066734	0.753487
(Thresh)	(Aphelios)	0.025703	0.504931	4.635127	0.020158	1.799879	0.826319
(Yuumi)	(Zeri)	0.018976	0.517808	3.351118	0.013313	1.753414	0.728281
(Sejuani, Zeri)	(Lulu)	0.012651	0.488372	4.474872	0.009824	1.741233	0.797180
(Xayah, Wukong)	(Rakan)	0.012149	0.487903	4.118234	0.009199	1.721405	0.776512
(Caitlyn)	(Lux)	0.022088	0.424710	15.725338	0.020684	1.691308	0.987781
(Vi, Zeri)	(Lulu)	0.012450	0.466165	4.271396	0.009535	1.668801	0.786900
(Ezreal)	(Karma)	0.020482	0.439655	5.933558	0.017030	1.652382	0.872095
(Wukong,	(Xayah)	0.012149	0.451493	3.747388	0.008907	1.603475	0.753420

Rakan)							
(Rakan)	**(Xayah)**	**0.052510**	**0.443220**	**3.678729**	**0.038236**	**1.579652**	**0.826030**
(Xayah)	**(Rakan)**	**0.052510**	**0.435833**	**3.678729**	**0.038236**	**1.562528**	**0.827916**
(Zeri)	**(Lulu)**	**0.065562**	**0.424301**	**3.887804**	**0.048699**	**1.547448**	**0.878535**
(Zeri, K'Sante)	**(Lulu)**	**0.011747**	**0.417857**	**3.828755**	**0.008679**	**1.530318**	**0.760189**
(Wukong, Zeri)	**(Lulu)**	**0.012952**	**0.404389**	**3.705346**	**0.009456**	**1.495713**	**0.754278**
(Heimerdinger)	**(Varus)**	**0.016566**	**0.421995**	**3.097324**	**0.011218**	**1.494373**	**0.704809**
(Karma)	**(Ezreal)**	**0.020482**	**0.276423**	**5.933558**	**0.017030**	**1.317639**	**0.898006**

Table B4: Shows the output for when the apriori support threshold was set to 0.01 support values. These values were then sorted by conviction, and the confidence was limited to 0.2 to 0.81 to remove Lucian and Nami-related interactions, more details in the results.

Table B5

antecedents	consequents	support	confidence	lift	leverage	conviction	zhangs_metric
(Caitlyn)	(Lux)	0.022088	0.424710	15.725338	0.020684	1.691308	0.987781
(Ezreal)	(Karma)	0.020482	0.439655	5.933558	0.017030	1.652382	0.872095
(Karma)	(Ezreal)	0.020482	0.276423	5.933558	0.017030	1.317639	0.898006
(Thresh)	(Aphelios)	0.025703	0.504931	4.635127	0.020158	1.799879	0.826319
(Aphelios)	(Thresh)	0.025703	0.235945	4.635127	0.020158	1.242183	0.880134
(Sejuani, Zeri)	(Lulu)	0.012651	0.488372	4.474872	0.009824	1.741233	0.797180
(Vi, Lulu)	(Zeri)	0.012450	0.673913	4.361387	0.009595	2.592811	0.785221
(Vi, Zeri)	(Lulu)	0.012450	0.466165	4.271396	0.009535	1.668801	0.786900
(Sejuani, Lulu)	(Zeri)	0.012651	0.656250	4.247076	0.009672	2.459584	0.779572
(Xayah, Wukong)	(Rakan)	0.012149	0.487903	4.118234	0.009199	1.721405	0.776512
(Wukong, Lulu)	(Zeri)	0.012952	0.629268	4.072458	0.009771	2.280576	0.770303
(Lulu)	(Zeri)	0.065562	0.600736	3.887804	0.048699	2.117601	0.833781

Table B5: Shows the output for when the apriori support threshold was set to 0.01 support values. These values were then sorted by lift. and the confidence was still limited to 0.2 to 0.81 to remove Lucian and Nami-related interactions.

Appendix C

Image C1

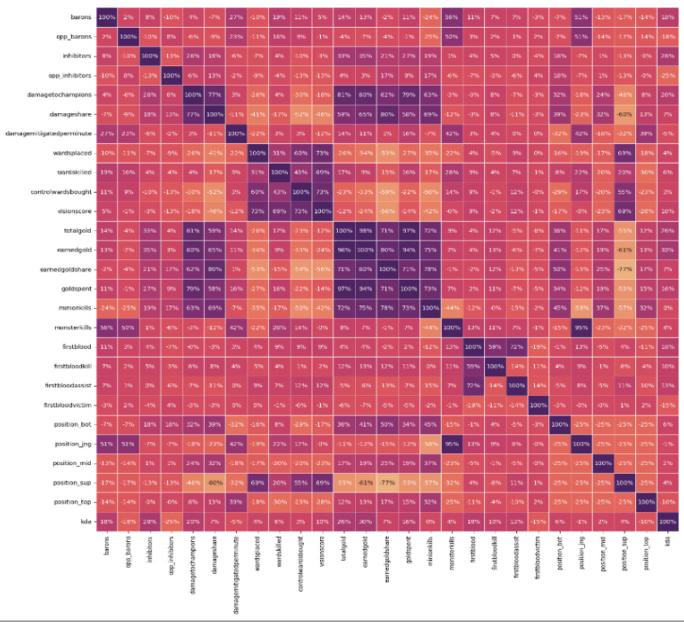


Image C1: This matrix shows the correlation between variables among themselves and KDA. The darker spots reveal a strong positive correlation, and the lighter spot shows a negative correlation. This plot shows how certain variables seem to be able to explain one another and variables that are redundant as a combination of two or more of them can be explained by one variable.

Image C2

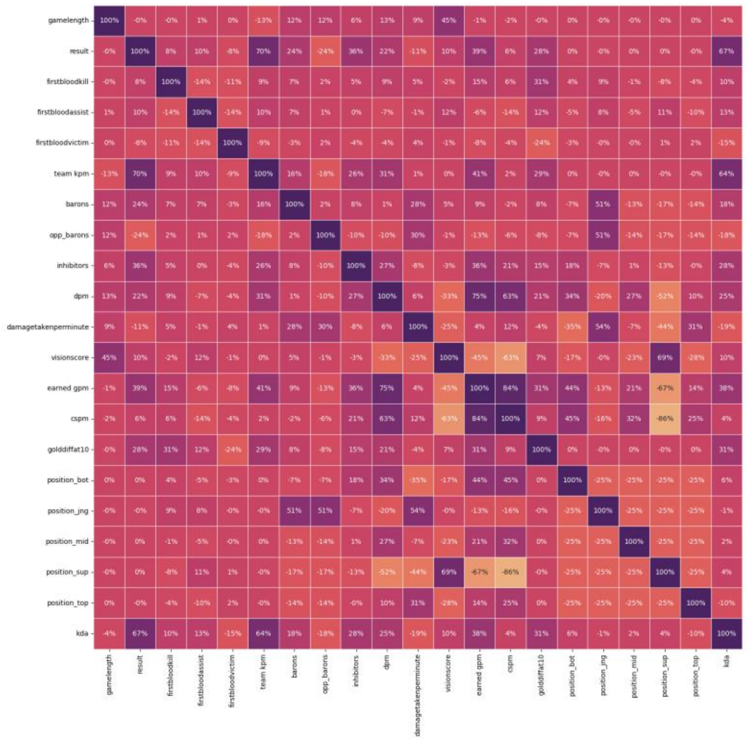


Image C2: In this new matrix redundant variables were removed. It is better because of the lack of extremes and should be easier to work with than the previous matrix. There are some interesting values in this matrix that will help to discern the relation between KDA to all other values. Notice that KDA has a negative correlation with game length and a strong positive correlation with the result of the game. Meaning if the result is a win, then the KDA would be high. The difference between the correlation matrix and the model is explained in detail in the results.

Image C3

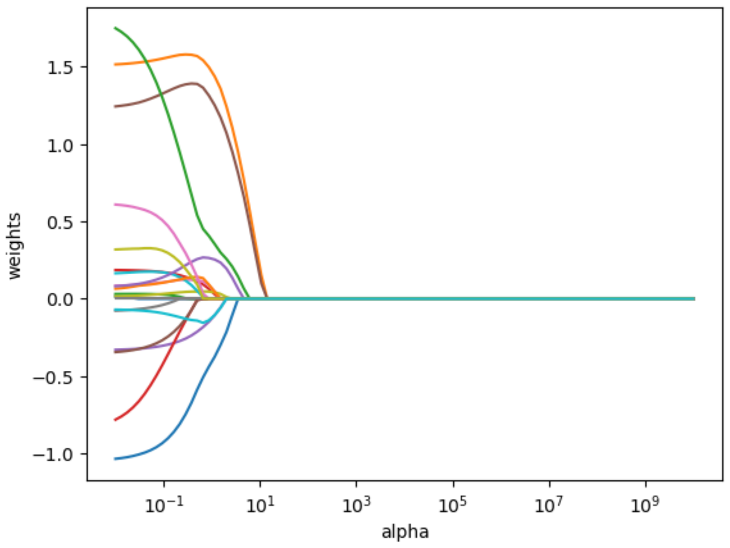


Image C3: shows the shrinkage of each variable within the dataset in the Elastic Net algorithm based on alpha value. Notice that by alpha 10^1 all are variables have effectively been reduced to zero.

Image C4

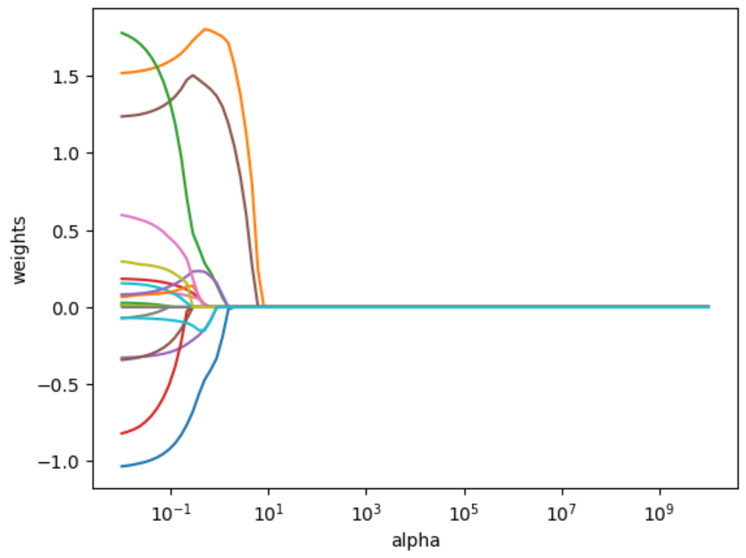


Image C4: Shows how the variables are shrunk using Lasso. Notice the graph is slightly different as the variables take a sharper dive toward zero compared to the elastic net.

Appendix D

Image D1

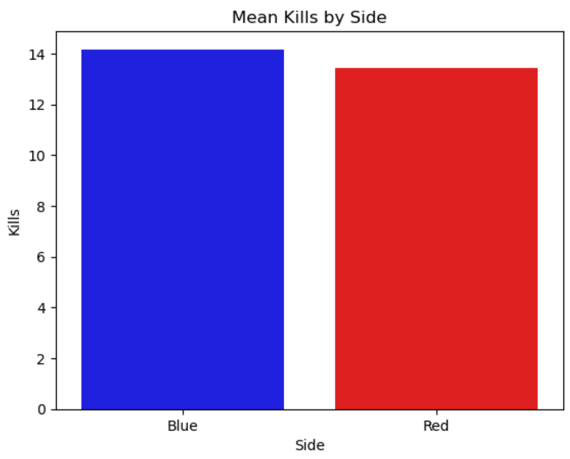


Image D1: Shows the kill disparity between the blue and red sides. A key thing to note is the red side seems to be a little lower, suggesting that being on the blue side is beneficial to a team in improving the number of kills a team earns. This also implies that since the number of kills is higher for the blue side, the number of deaths is greater on the red side. Similarly, the gold earned is higher on blue since kills generate gold. This disparity may also be explained simply by the fact that the blue side won more games than the red side, and the winning team usually averages more kills than the losing team.

Image D3

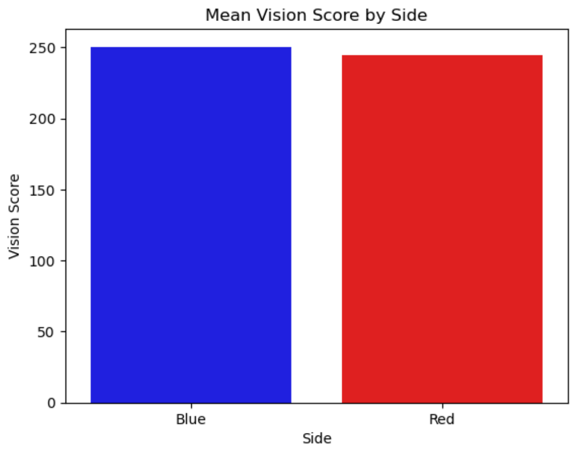


Image D3: An interesting thing to note here is that the players seem to ward a little more when they are on the blue side vs the red side. Perhaps this could be something the players should be encouraged to work on improving their vision score when on the red side of the map. This could also potentially reduce the kill differential mentioned before. The difference here may also be explained the difference in amount of games won.

Image D4

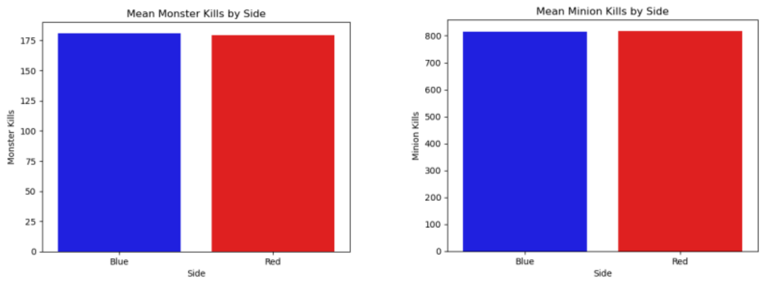
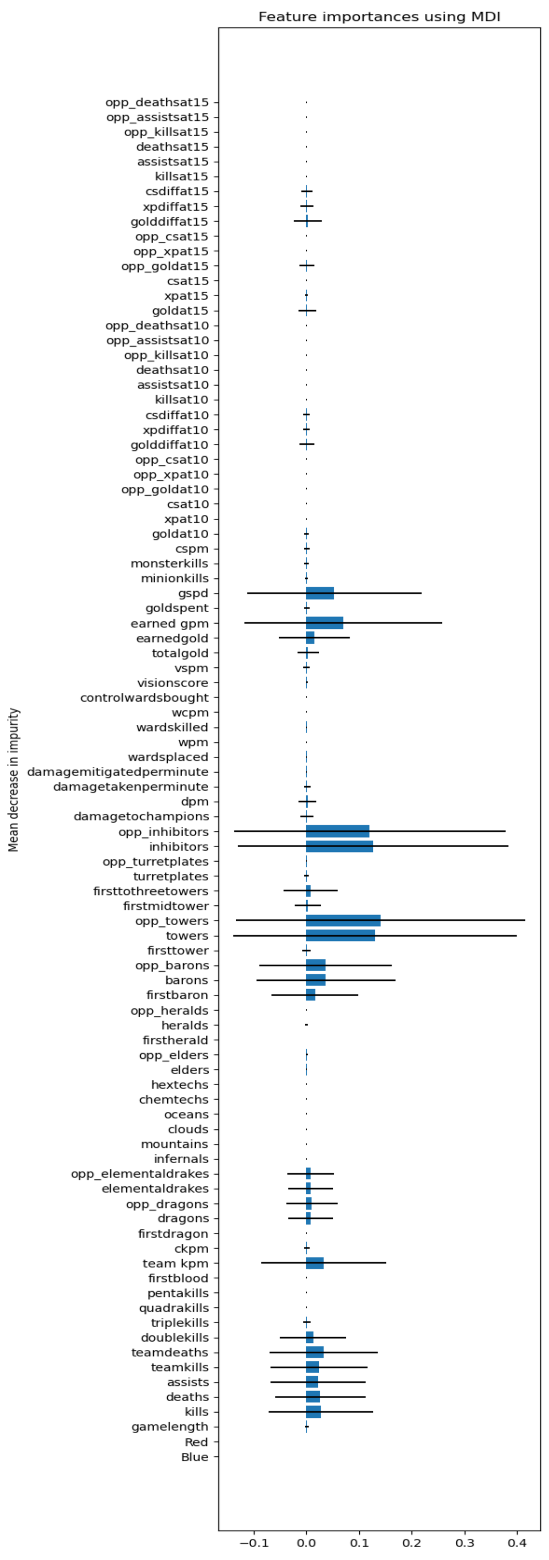


Image D4: Observe that the number of minion and monster kills are even between the two sides respectively.

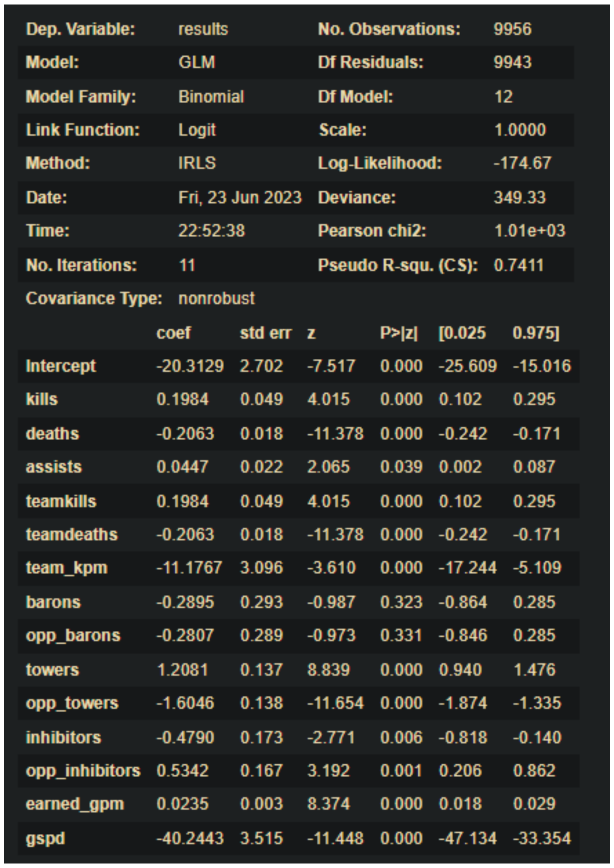
Image D5



Feature importances using MDI
Mean decrease in impurity
opp_deathsat15
opp_assistsat15
opp_killsat15
deathsat15
assistsat15
killsat15
csdiffat15
xpdiffat15
golddiffat15
opp_csat15
opp_xpat15
opp_goldat15
csat15
xpat15
goldat15
opp_deathsat10
opp_assistsat10
opp_killsat10
deathsat10
assistsat10
killsat10
csdiffat10
xpdiffat10
golddiffat10
opp_csat10
opp_xpat10
opp_goldat10
csat10
xpat10
goldat10
cspm
monsterkills
minionkills
gspd
goldspent
earned gpm
earnedgold
totalgold
vspm
visionscore
controlwardsbought
wcpm
wardskilled
wpm
wardsplaced
damagemitigatedperminute
damagetakenperminute
dpm
damagetochampions
opp_inhibitors
inhibitors
opp_turretplates
turretplates
firsttothreetowers
firstmidtower
opp_towers
towers
firsttower
opp_barons
barons
firstbaron
opp_heralds
heralds
firstherald
opp_elders
elders
hextechs
chemtechs
oceans
clouds
mountains
infernals
opp_elementaldrakes
elementaldrakes
opp_dragons
dragons
firstdragon
ckpm
team kpm
firstblood
pentakills
quadrakills
triplekills
doublekills
teamdeaths
teamkills
assists
deaths
kills
gamelength
Red
Blue
−0.1
0.0
0.1
0.2
0.3
0.4

Image D5: Shows the random forest importance plot for all the variables that the random forest algorithm ran on.

Image D6



Dep. Variable:	results	No. Observations:	9956
Model:	GLM	Df Residuals:	9943
Model Family:	Binomial	Df Model:	12
Link Function:	Logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-174.67
Date:	Fri, 23 Jun 2023	Deviance:	349.33
Time:	22:52:38	Pearson chi2:	1.01e+03
No. Iterations:	11	Pseudo R-squ. (CS):	0.7411
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-20.3129	2.702	-7.517	0.000	-25.609	-15.016
kills	0.1984	0.049	4.015	0.000	0.102	0.295
deaths	-0.2063	0.018	-11.378	0.000	-0.242	-0.171
assists	0.0447	0.022	2.065	0.039	0.002	0.087
teamkills	0.1984	0.049	4.015	0.000	0.102	0.295
teamdeaths	-0.2063	0.018	-11.378	0.000	-0.242	-0.171
team_kpm	-11.1767	3.096	-3.610	0.000	-17.244	-5.109
barons	-0.2895	0.293	-0.987	0.323	-0.864	0.285
opp_barons	-0.2807	0.289	-0.973	0.331	-0.846	0.285
towers	1.2081	0.137	8.839	0.000	0.940	1.476
opp_towers	-1.6046	0.138	-11.654	0.000	-1.874	-1.335
inhibitors	-0.4790	0.173	-2.771	0.006	-0.818	-0.140
opp_inhibitors	0.5342	0.167	3.192	0.001	0.206	0.862
earned_gpm	0.0235	0.003	8.374	0.000	0.018	0.029
gspd	-40.2443	3.515	-11.448	0.000	-47.134	-33.354

Image D6: The image shows the summary results for the regression plot that was constructed. Interestingly, observe that the p-values that indicate if the variables explain a win or a loss are all significant and under the 0.05 threshold except for baron, opp_baron, and assists. However, since the model provides good results, there was no need to remove non-significant variables.

References

1. (Riot) Games, Riot, director. *What Is League of Legends*. Youtube, 13 Oct. 2015, 1. https://www.youtube.com/watch?v=BGtROJeMPeE&ab_channel=LeagueofLegends. Accessed 26 June 2023.
2. (Iyer) Iyer, Ravi. *5 Esports Tournaments with Highest Expected Prize Pools in 2023*, es-charts.com/news/5-esports-tournaments-highest-expected-prize-pools-2023. Accessed 26 June 2023.
3. (Data) Sevenhuysen, Tim. “Oracle’s Elixir - Lol Esports Stats.” Oracle’s Elixir - LoL Esports Stats, oracleselixir.com/. Accessed 26 June 2023.
4. (Gareth) Gareth , James, et al. *An Introduction to Statistical Learning*. Springer, 2017.
5. (Geron) Géron, Aurélien. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd ed., O’Reilly Media, 2022.
6. (Siddiqui) Siddiqui, Shamoon. “How Would We Find a Better Activation Function than Relu?” *Medium*, 1 Jan. 2019, medium.com/shallow-thoughts-about-deep-learning/how-would-we-find-a-better-activation-function-than-relu-4409df217a5c.