


Spring 2022

## Predicting League of Legends Ranked Games Outcome

Ngoc Linh Chi Nguyen  
*Bard College*

Follow this and additional works at: [https://digitalcommons.bard.edu/senproj\\_s2022](https://digitalcommons.bard.edu/senproj_s2022)

 Part of the [Databases and Information Systems Commons](#), [Numerical Analysis and Scientific Computing Commons](#), [Other Computer Sciences Commons](#), [Sports Studies Commons](#), and the [Theory and Algorithms Commons](#)



This work is licensed under a [Creative Commons Attribution-Noncommercial-No Derivative Works 4.0 License](#).

---

### Recommended Citation

Nguyen, Ngoc Linh Chi, "Predicting League of Legends Ranked Games Outcome" (2022). *Senior Projects Spring 2022*. 267.

[https://digitalcommons.bard.edu/senproj\\_s2022/267](https://digitalcommons.bard.edu/senproj_s2022/267)

This Open Access is brought to you for free and open access by the Bard Undergraduate Senior Projects at Bard Digital Commons. It has been accepted for inclusion in Senior Projects Spring 2022 by an authorized administrator of Bard Digital Commons. For more information, please contact [digitalcommons@bard.edu](mailto:digitalcommons@bard.edu).

# Predicting League of Legends Ranked Games Outcome

Senior Project Submitted to  
The Division of Science, Mathematics, and Computing  
of Bard College

by

Ngoc Linh Chi Nguyen

Annandale-on-Hudson, New York  
May 2022



## **Acknowledgements**

I want to thank my advisor Sven Anderson and Kerri-Ann Norton, who help and guild me through this project. I would also like to thank my friend Abigail Shane Lulman, who helped me through this paper.

Special thanks to my family and my partner Chase Flores for being there to motivate me through my four years at Bard.



# Table of Contents

<b>Abstract</b>	<b>5</b>
<b>Chapter 1: Introduction</b>	<b>6</b>
1.1 Competitive Esports	6
1.2 Introduction to League of Legends	7
1.2.1 Ranked Games	7
1.2.2 Game mechanics	9
1.3 Introduction to Riot API and data	12
1.4 Machine Learning to predict games outcome	13
1.5 Players' decision-making process	15
<b>Chapter 2: Methodology</b>	<b>17</b>
2.1 Data collection	17
2.1.1 Collection process	17
2.1.2 Variable Descriptions	20
2.2 Machine learning models	22
2.2.1 Logistic Regression	23
2.2.2 Random Forest	24
2.2.3 Support Vector Machine (SVM)	25
<b>Chapter 3: Results</b>	<b>27</b>
3.1 Data Overview	27
3.2 Models Overview	28
3.2.1 Logistic Regression	29
Coefficients:	33
3.2.2 Random Forest	39
Feature Importances:	44
3.2.3 Support Vector Machine (SVM)	46
3.3 All Models Summary	47
<b>Chapter 4: Discussion</b>	<b>49</b>
4.1 Reflection	49
4.2 Improvement	50
<b>Reference</b>	<b>52</b>
<b>Appendix</b>	<b>58</b>

## Abstract

League of Legends (LoL) is the one of most popular multiplayer online battle arena (MOBA) games in the world. For LoL, the most competitive way to evaluate a player's skill level, below the professional Esports level, is competitive ranked games. These ranked games utilize a matchmaking system based on the player's ranks to form a fair team for each game. However, a rank game's outcome cannot necessarily be predicted using just players' ranks, there are a significant number of different variables impacting a rank game depending on how well each team plays. In this paper, I propose a method to predict rank game outcomes based on several different variables that would be collected 14minutes into the games. Using three different machine learning algorithms: Logistic Regression, Random Forest, and Support Vector Machine, I found that game outcomes can be predicted with 79.46% accuracy with all the collected data and newly added variables. I also use the model to show the importance of each variable in helping players make winning decisions. The results show that the difference in gold between the two teams has the most impact on the final result of the games. The skill difference in the three groups can be further examined with extra variables that were not explored during the study.

# Chapter 1: Introduction

## 1.1 Competitive Esports

Since its release in 2009, League of Legends (LoL) has been one of the most popular multiplayer online battle arena (MOBA) game titles out there (How Many People Play, 2021). In 2020, the game had 115 million League of Legends monthly active players, and its popularity has led to the successful production of its music videos, virtual K-pop bands, comics books, lore stories, and an upcoming animated series. The company, Riot Games, also uses the success of LoL to have several spin-off games, including a mobile version of LoL, Wild Rift, and a digital card game that is based in the same world.

LoL is often referred to as the world's largest esports game, with multiple regional, domestic, and international tournaments (League of Legends in Esports - Wikipedia, 2021). There are 12 regional League of Legends Championship Series composed of North America (LCS), Europe (LEC), China (LPL), Korea (LCK), Southeast Asia (PCS and VCS), Brazil (CBLOL), Russia (LCL), Japan (LJL), Latin America (LLA), Turkey (TCL), and Oceania (LCO). Each region has two main competitions per season, which are the spring and summer split. The best teams from each region then have a chance to compete in the Mid-Season Invitational (MSI), which occurs in between the spring and summer split and the League of Legends World Championship, hosted at the end of the season. The 2019 League of Legends World Championship prize pool amounted to an astonishing \$6.45 million with over 100 million unique viewers, peaking at a concurrent viewership of 44 million. All of these events have also



been broadcast on live streaming websites like Twitch<sup>1</sup>, YouTube, Bilibili, and even on ESPN, a cable television sports channel.

With both fame and fortune at stake, the outcome of a single game would decide players' careers and paths. With this in mind, the ability to predict the game outcome before the game ends can be useful and impactful to the Esports industry. The machine learning model can be useful to provide feedback to players, so they can improve their skills, a potentially great engagement tool to attract audiences that are both familiar or unfamiliar with the game, and last but not least, a potential tool that could outperform human prediction in the betting of Esports, could provide money-making opportunities. In this study, I proposed to use statistics and machine learning during a game to predict the outcome of the game before it ends.

## **1.2 Introduction to League of Legends**

### **1.2.1 Ranked Games**

Published in 2009, League of Legends (LOL) grew rapidly throughout the years and took over the gaming and esports industry by storm. The game is a Multiplayer Online Battle Arena (MOBA), which means that there are multiple players in one game using the internet to connect (League of Legends - Wikipedia, 2021). In standard games, two teams compete against each other, each team has five players, and each player controls a single character called a “Champion” and engages in real-time combat with the enemies. The goal of the game is to progressively and quite linearly progress into the enemy territory, destroying their defense structures along the way with the final objective being the destruction of the Nexus, the enemy's core and final structure. Upon the destruction of the Nexus, that team is declared victorious.

---

<sup>1</sup> <https://www.twitch.tv/>

The game has different game modes. For the time being, I will only focus on ranked mode, the most competitive aspect of the game. While other game modes have no impact on your rank status, ranked games, on the other hand, will give you League Points (LP or MMR), in what is called the League system. This system matches players of a similar skill level to play with and against each other. League of Legends utilizes two different matchmaking functions based on the size of your matchmaking party. In this project, I will only be analyzing the matchmaking system that allows up to two people to queue and search for a match together (solo/duo).

The system comprises nine different tiers which rank from lowest to the highest skill level of players: Iron, Bronze, Gold, Platinum, Diamond, Master, Grandmaster, Challenger. Each tier from Iron to Diamond has four divisions, indicated by a Roman numeral from IV (lowest) to I (highest). For Master, Grandmaster and Challenger, they are instead exclusively reliant on LP and the population of players within these rank tiers, known as Apex tiers.



Figure 1: Rank representative picture. Contained 9 different tiers which rank from lowest to the highest. (LoL Ranks 2020, 2022)

The game developer, Riot Games, also updates the game every couple of months for balance and changes into the game. These changes are influenced by player performance in different skill levels: Average (Iron IV – Gold I), Skilled (Platinum IV – Diamond III), Elite (Diamond II – Challenger), and Pro (Top 5 Pro Regions: LPL, LCK, LEC, LCS, LMS). In this paper, we will only focus on the Elite, Skilled, and Average.

### 1.2.2 Game mechanics

Games begin with the champion selection screen, where each player on each team gets to choose their champions from 157 different characters<sup>2</sup>. During specific modes (ranked and normal draft), each player can ban a champion before the selection process begins. After the champion selection, the games start with all ten champions at level one in their home base (Nexus). The map for ranked games is called Summoner's Rift which has three main lanes,

<sup>2</sup> As of October 2021

spawning minions waves, jungle camps, structures (eleven turrets, three inhibitors, one nexus) for each team, and neutral jungle camps (neutral objectives).

In a standard ranked game, each player on a team will fulfill each of these five positions: top laner, jungler, mid laner, attack damage carry (adc), and support. Each champion has five abilities: one passive and four active. Gold and experience are crucial winning factors. During the game, each player can gain gold and experience by last hitting minions; taking jungle camps, neutral objectives, or enemy structures; killing enemy champions; and finally, they also get a small passive gold income through the passing of time during the game.



*Figure 2: Summoner's Rift map with notation for objectives and players' roles. (A Map of the League of Legends, 2021)*

Characters can level up from gaining experience, each level-up will give their character a stats boost and a choice to advance one of their abilities (max at level eighteen). Additionally, using gold, players can buy items for their champions to increase their stats like health, damage, etc., giving them more chances to gain more experience and gold with their current champions.

In addition to gold and experience, there are multiple other game-winning factors that contribute to the outcome of a ranked game. First, since the map is not lit (only when a champion walks into range of a part of the map they can see that part), vision control is very important to obtain pieces of information. Vision control is a term for warding, a mechanic in LoL that allows you to put down an object called “ward” to provide vision to an area for your entire team. Second, neutral objects also contribute to the win condition of a game. When your team takes specific neutral monsters they also give certain stats buffs to your whole team (dragons, barons, and rift herald), and thus help to increase your chance to get more gold, experience, and even more objectives.

The game ends when a team’s Nexus is destroyed. You cannot destroy the Nexus without first destroying an inhibitor from a lane, and you can not destroy an inhibitor without first destroying the turrets in order (this only applies to each individual lane). Hence, destroying the enemy's structures is also important for winning the game.

On the other hand, taking the outer turret (or tier one) turret also gives you gold for each plate , then split to all nearby allies. There are five plates to each tier one turret, each consisting of 1000 health, making these turrets harder to take down. These plates disappear at 14-minutes into the game with all the gold you could have gotten from them. This means that you want to take down as many plates as possible before the 14 minute mark to give you and your team the optimal gold advantage.

Typical LoL ranked games usually last around 30 to 45 minutes, and each game can be divided into three main phases: the laning phase, mid-game, and late-game. Usually, the laning phase is around the first 10 to 15 minutes, where players try to gather experience and gold by farming (killing minions, jungle camps) in their own lanes. These are the crucial foundation for what an outcome of a game would look like. During this phase is where teams and players establish their foundation and gather advantages to use during mid and late games. At 14-minutes is where turret plating falls off, where the game is shifting into mid-game. Also, at 15 minutes is where your team can start a surrender vote where an anonymous vote is made to whether they can withdraw from the game and accept a loss to their match. This is where we want to make our prediction using all the information that we have gathered from the first 14 minutes.

### **1.3 Introduction to Riot API and data**

For their gaming community, Riot Games created Riot API (Application Program Interface), which is a REST API that allows developers to get data from Riots' games such as League of Legends, Valorant, Legends of Runeterra, Teamfight Tactics (Riot Developer Portal, 2022). Developers can use data from their games and create independent applications and websites for the gaming community. To do this, developers have to register their products for a permanent API key. The Riot API for League of Legends gives you information about the status of the server, players' details (name, level, profile icon, account ids), players' match history and match details, players' current game status, and more. Examples of some of the websites and applications created by outside developers for League of Legends are websites that help players view their stats (winrate, match history, specific champions' win rates, etc.) and applications that help new players to adjust their champions' runes (pre-match data) and recommend items to

build for the champion that they are playing. Some applications even give you information about your opponents during the loading of the game to help you better understand your opponent's gameplay. The most popular websites include op.gg and u.gg, where they present data about individual summoners, their match history, and their current game being played along with their opponents' information (if applicable).

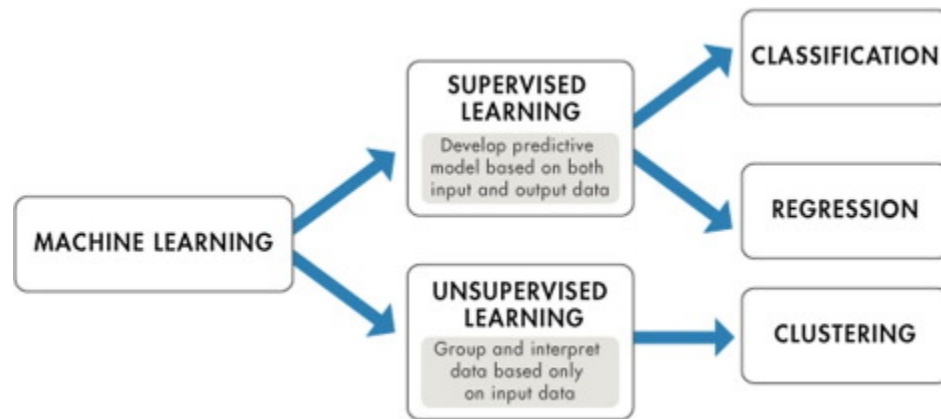
For our study, we will look at specific data of players' games at the 14-minute mark. Using these data, we will predict the result with some of the factors that impact the game outcome that have been mentioned before. We will take those data at the 14-minute mark from both teams and use them to predict. All of the data we will consider for each team includes: gold difference between the two teams, total gold, average level (from 1 to 18), minions killed, jungle minions killed, the total damage to champions, structures destroyed, dragons killed, herald/baron killed, elite monsters kill, the team total kill/death/assist (KDA), wards placed, and wards destroyed.

#### **1.4 Machine Learning to predict games outcome**

In the past, there have been many attempts to predict the outcomes of certain sports games. When machine learning and AI started to develop more and more, they became popular tools used to predict sports and Esports outcomes. In the world of sports, using machine learning to predict the outcome of matches has been popular. This is also true for the world of Esports. The amount of work to build a model that can successfully predict outcomes of Esports games is increasing (Li & Fu, 2021) .

One of the common machine learning algorithms used in sports is classification, which involves predicting a target variable in previously unseen data. The goals of classification are to predict a target variable by building a model based on a training dataset, then use that model to

predict the value of the target variable in the test dataset. This is called supervised learning, where this phase is guided toward the target variable while building the model for prediction.



*Figure 3: Supervised learning versus unsupervised learning (Bunker & Thabtah, 2019)*

Several predictors use pre-match data (before the match begins) to predict the outcome, which prevents them from having accurate results for non-professional games since pre-match data sometimes have little to no impact on non-professional Esports environments.

For League of Legends predictors, some researchers and predictors use pre-match data and data at the 10-minute mark to predict the outcome. There are various models and research that have been built to predict the outcome of games, including similar league of legends researches and reports such as predicting the game outcomes at the 10-minute mark, predicting based on player-champion experience, or predicting the winner in a live professional match (Cabrera, n.d.; Do et al., 2021; Hodge et al., 2019). Most of the previous research used multiple classification algorithms such as Logistic Regression, Decision Trees, Support Vector Machine, Ensemble Algorithms, k-Nearest Neighbors, Gradient Boosting, and Deep Neural Network. These showed results ranging from 70.61% to 75.1%.



For the Cabrera, n.d. article, the author use data at the 10-minute mark and he collected the data through Riot API as well. The data collected was similar to mine but he did not separate the red and blue teams, but instead combined the data for both teams and found the difference between the two. He also collected data from three other regions besides North America. This study has four machine learning models: Logistic Regression, Decision Tree, Support Vector Machine and Ensemble Algorithms. Additionally, the Do et al., 2021 article uses the player-champion experience which is how much experience the players have on the champions that they are using in the game as data for predicting the outcomes.

However, for this paper, I decided to use data at the 14-minute mark with no pre-match data or live prediction. This is because pre-match (before the match begins) and live data (using data when the game is happening) are harder to collect due to the availability on the Riot API, where the game publishes most of its data. This also prevents them from having accurate results for non-professional games since pre-match data sometimes have little to no impact on non-professional Esports environments. On the other hand, some of the researchers choose to predict using the 10-minute mark instead of the 14-minute mark. This is because there are online data available for the 10-minute mark of the game already collected on open sources sites which helps them speed up the process. I chose the 14-minute mark as discussed in Section 1.2.2 (Game Mechanics).

### **1.5 Players' decision-making process**

Research has shown that individuals playing video games increase their behavior, cognitive skills, time management, and decision-making process. Some analyses indicate that depending on the type of video gameplay, they enhance the domains of top-down attention and spatial cognition, with encouraging signs for perception (Bediou et al., 2018; Reynaldo et al.,

2021). Our study involves the MOBA games League of Legends which is considered a strategy game with a teamwork co-op purpose. The game involves significant instant decision-making throughout a stressful high-pace environment where each decision a player makes can change the outcome for them, and their team. In a competitive gaming scene, a player usually experiences multiple decision-making processes that affect their competition for thousands of dollars, their whole career as a professional gamer, and on top of that, doing so in front of more than 100 million viewers. This helps indicate the importance of decision-making during the games, and in this study, we will look at factors in the game that are most important for players to make these decisions.

## **Chapter 2: Methodology**

### **2.1 Data collection**

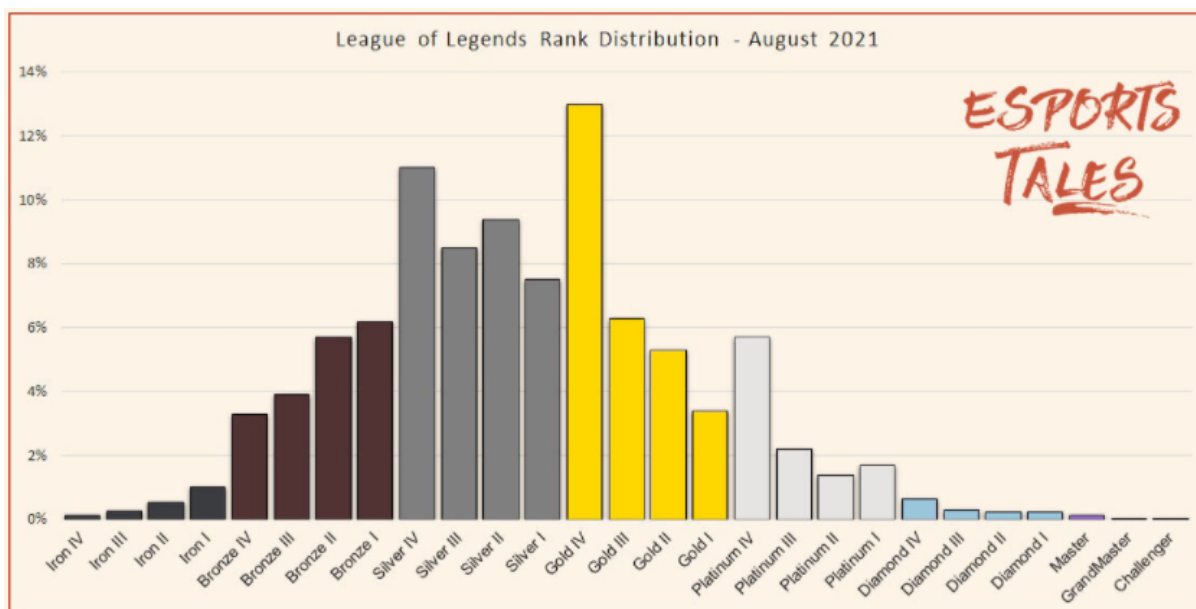
The Riot API, created by Riot Games, has multiple versions. The basic key of the Riot API allows 50 requests per minute (one request every 1.2 seconds).

The data was collected from September to November of 2021. The game's current patch note at that time was patch 11.18 to patch 11.23. Using the Riot API, I have access to data files that could be used to extract matches id, then use those matches id to extract detailed information about each match and store them in excel files.

#### **2.1.1 Collection process**

The collection process started by dividing the player population ranked into three categories mentioned in the introduction:

- Average: players rank from Iron IV - Gold I (86.4% of the population)
- Skilled: players rank from Platinum IV - Diamond III (13% of the population)
- Elite: players rank from Diamond II - Challenger (0.6% of the population)



*Figure 4: League of Legends Rank Distribution in August 2021 (Vincenzo, 2021)*

Using this separation, we can more accurately predict ranked games outcome based on their skill level. This is because different skilled players behave differently during games, therefore the variable that affects our outcome could weigh differently as the way the player behaves and progresses throughout the game is very different. For example, during a game of League of Legends, if players were to be in a team that is in a losing scenario and their teams are playing at a great disadvantage, a lower ranked player in Bronze or Silver might keep playing that game even though the current situation in the game might not look good for them. This is because they either do not know enough information to conclude that the game is not winnable for their team or they know that in lower-ranked, people tend to make mistakes and that they can exploit it to come back and win the game even with a large disadvantage. However, in the same scenario for a higher ranked player, they would either know that the game situation is not winnable anymore as they are currently at a huge disadvantage and that it is not easily reversible as players in their ranks can easily use the advantage to end the game. Hence, the higher-ranked

players sometimes might decide to surrender a game early to save time and move on to the next game.

First, because the Riot API does not have a function built in to retrieve an arbitrary amount of games or players, I manually collected 27 random summoner names, each for each rank division using the 'leaderboard' section of the u.gg website that lists all players on each server by rank. In this case, I will be using data from the North American server only. We will call them seeds as they will be used as the beginning data to collect more player information from the API. Then using the API, I collected necessary data and filled them in for each seed: id, account Id, puuid, summoner level. The 'puuid' variable here is important as it is a unique primary key that identifies players. With that in mind, I then used each player in the seed list to collect a number of players around the same ranked level using their match history, eliminating any duplicates. This process was repeated until a fixed number of players were found for each group. Noted that the Average group has the highest number of players, and Elite has the lowest because of the population distribution I mention earlier:

- Elite: 2753 players
- Skilled: 9037 players
- Average: 15023 players

A similar process was performed to collect the matches Id. I selected random games from each player's match history with a random number of games being collected ranging from 1 to 5 games per player, eliminating duplicates. Then repeat the process until a desired sample of matches was found:

- Elite: 10005 matches
- Skilled: 30003 matches

- Average: 75003 matches

Once the list of matches was found, they were then used to extract data at 14 minutes into the game. Here I have to use multiple Jupyter notebooks to retrieve the data because the API limits our calls per minute so we have to run multiple kernels (using a different API key) to speed up the process. After combining the files for each group, I then filter the data as some of the collected information could contain errors. I did this by graphing each variable and then eliminating the out-of-bound value on both ends (either it is too low or too high for the variable to be correct). After I filtered the data, this is the final filtered data available for use:

- Elite: 9595 matches
- Skilled: 25220 matches
- Average: 62398 matches

### 2.1.2 Variable Descriptions

All the variables were extracted from the 14-minute mark of the game, in which case is frame 14 in the JSON file that the match info was in. Here is a list of variables that were extracted from the API:

- *blueWins*: the value is 0 or 1 indicating the winning team
- *blueWardsPlaced*, *redWardsPlaced*: each teams number of ward places
- *blueWardsDestroyed*, *redWardsDestroyed*: each team number of wards killed
- *blueKills*, *redKills*: each team number of kills
- *blueDeaths*, *redDeaths*: each team number of deaths
- *blueAssists*, *redAssists*: each team number of assists
- *blueEliteMonsters*, *redEliteMonsters*: each team number of elite monsters killed (dragon, baron or heralds)

- *blueDragons, redDragons*: each team number of dragons killed
- *blueHeraldsBarons, redHeraldsBarons*: each team number of heralds or barons killed
- *blueTowersDestroyed, redTowersDestroyed*: each team number of turrets takedown
- *blueTotalGold, redTotalGold*: each team total gold earned
- *blueTotalDamageToChampions, redTotalDamageToChampions*: each team's total damage done to opponents champions
- *blueAvgLevel, redAvgLevel*: each team's average level (min 1, max 18)
- *blueTotalMinionsKilled, redTotalMinionsKilled*: each team's total number of creeps killed
- *blueTotalJungleMinionsKilled, redTotalJungleMinionsKilled*: each team total number of jungle camps killed
- *blueGoldDiff, redGoldDiff\**: gold different on each team

\*I decided to remove the redGoldDiff after collecting it because the value is the same as the blueGoldDiff.

After collecting all the variables, I also added new variables to create an extra set of data. These newly added variables are non-linear combinations of data that were in the original data set. The new data set comprises the old data and the newly added combination:

- *blue/redAvgLevel*: ratio of the blue team and red team average level
- *blue/redTotalDamageToChampions*: ratio of blue and red team total damage to opponents champions
- *blue/redKills*: ratio of blue and red team kills obtained

I added these variables because the ratio between the two teams may help to determine the difference in advantage during the stage of the games. These features represent each game's

situation and help us determine whether certain important information can predict match outcomes.

I also check the bias of the data before starting to split data. The general metric related to win rate by teams is balanced with blue sides having higher win rate in Elite and Average groups but with less than 0.5% difference. Overall, the blue side has a 50.01 % win rate for 97213 games for this data sample. This is non-bias data if the result was guessed, it would be a 50% chance of being correct.

<b>Groups</b>	<b># Matches</b>	<b>Blue Win Rate</b>
Elite	9595	50.20 %
Skilled	25220	49.50 %
Average	62398	50.33 %
Total	97213	50.01 %

*Table 1: Winrates bias checking results*

## **2.2 Machine learning models**

Predicting the outcome of a particular game is considered a classification problem as the end goal is to predict which team will win: either the blue side wins, the red side loses or vice versa. Therefore, I chose to use classification algorithms such as Logistic Regression, Random Forest, and Support Vector Machine (SVM).

In addition, I separate 10% of the original data for testing purposes. The remaining 90% of the data is used for cross-validation to split data into training samples and validation. Cross-validation means partitioning the data into separate folds (in this case 5 folds), and then each fold will take turns as the validation sample while the other folds will be combined into a



training sample. This step is repeated  $k$  times where  $k$  is the number of folds that exist.

Afterward, the results are reported as the average results of all folds.

Variables such as 'blueTotalGold' and 'blueAvgLevel' have very different ranges. This magnitude difference biases many learning algorithms, lowering their overall performance. Therefore, I used the StandardScaler in Scikit-learn to rescale all variables. The StandardScaler standardizes a variable to a z-score by calculating the means then taking each variable, subtracting the mean, and dividing by the standard deviation. This process results in a distribution having the same shape as the original distribution, but with every variable having a standard deviation and variance equal to 1 and a mean of the distribution of 0.

### **2.2.1 Logistic Regression**

Logistic Regression, a variant of linear regression, is a supervised learning classification algorithm that is used to predict a specified target binary variable (Building A Logistic Regression, 2022). The algorithm uses a log of odds as the dependent variable and then predicts the probability of occurrence of a binary variable utilizing a logit function. It also describes and estimates the relationship between a dependent binary variable and other independent variables through maximum likelihood, a common learning algorithm used by a variety of machine learning algorithms.

In this case, I am using Scikit-learn's linear Logistic Regression model (Sklearn.Linear\_model.LogisticRegression, 2022). I ran the model three times with different penalties: No regularization with the 'sag' solver, L1 regularization with the 'liblinear' solver, and L2 regularization with the 'saga' solver. They all have a 'max\_iter' of 1000. After that, I take the fold of data that has the highest accuracy in the L1 penalty model and extract the coefficient of the variables. This is because L1 tends to shrink coefficients to zero whereas L2

tends to shrink coefficients evenly. This makes L1 useful for feature selection; we use L1 to measure how strongly a variable affects model prediction.

### **2.2.2 Random Forest**

Random Forest is a supervised algorithm that is widely used both in classification and regression problems. The algorithm is based on ensemble learning, which combines multiple classifiers to solve a complex problem. Ensemble is a machine learning model that combines the prediction of two or more models. The ensemble member makes predictions and then they may be combined using statistics or more refined methods that help each member learn how much to trust each other and under what conditions. With that in mind, ensembles help improve the predictive performance of classifiers because they reduce the variance component of the prediction error by adding bias.

The Random Forest algorithm involved decision trees which are non-parametric supervised learning techniques used for classification and regression. In the Decision trees, we start at the root of the tree to predict a class label for a record (Decision Tree Algorithm, 2022). Then we compare the values of the root feature to the records'. Then, for each comparison, we follow the branch corresponding to the value and hop to the next node. The decision tree then classifies the feature by sorting them down the tree from the root to some leaf or terminal node that provided the classification for the feature. Each node in the tree acts as an example of some attributes, and each edge descending from the node fits the possible solution to the case. Recursively, this process is repeated for every subtree rooted at the new node. They are used to create models that predict the value of target variables by creating data features from learning simple decision rules. They categorize and make predictions based on previous answers to a set

of questions. Decision trees are a type of supervised learning, meaning that the model is trained and then tested on a set of data that contains the desired categorization.

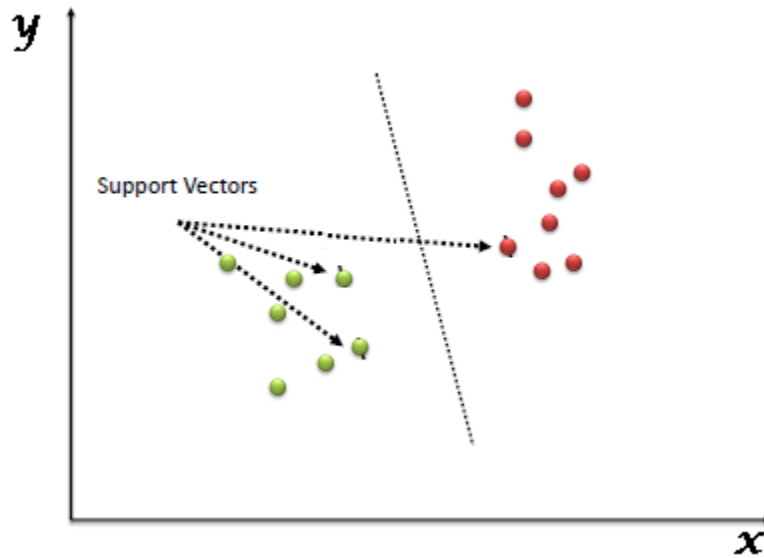
Random Forest builds decision trees on different samples and takes either the majority vote in the classification case or the average vote for regression. This means it contains a number of decision trees on various subsets of the given datasets and then takes the majority or average to improve the predictive accuracy of that dataset.

In this case, I am using Scikit-learn's ensemble Random Forest Classifier (Sklearn.Ensemble.RandomForestClassifier, 2022). After that, I also find the important feature of the model for each group by using the 'feature\_importances\_' attribute in the built-in Scikit-learn's ensemble Random Forest Classifier. The attribute returned an array with the impurity-based feature importances. This will help us to determine the variable that is most and least influential to the final prediction.

### **2.2.3 Support Vector Machine (SVM)**

Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression (SVM | Support Vector Machine, 2022). It is mostly used in classification which is what would be applicable to our problem. In SVM, each data item is plotted as a point in n-dimensional space, where n is the number of features, with each feature's value being the value of a particular coordinate. After that, classification is performed by finding the hyper-plane that differentiates the two classes which are demonstrated in Figure 15. Basically, the objective of the SVM is to find a hyperplane that distinctly classifies the data points in an n-dimensional space. To do this there are many possible hyperplanes that could be chosen, but the main goal is to find a plane that has the maximum margin, which is the maximum

distance between data points of both classes. This help provides some reinforcement for future data points that can be classified with more certainty.



*Figure 15: The hyper-plane that differentiates the two classes in SVM (SVM | Support Vector Machine, 2022)*

In this study, I used Scikit-learn's Support Vector Classification (1.4. Support Vector Machines, 2022). I trained the model with the 'linear' and 'rbf' kernel. I decided to go with the 'linear' kernel because the general yielded results are better for this kernel.

# Chapter 3: Results

## 3.1 Data Overview

I used the original data set collected by the API to analyze our dataset. Since the original data set have a large number of features, I use the Seaborn Heatmap functions and Matplotlib to create a heatmap for better visualization of these data (Seaborn.Heatmap, 2022). Here I only show the heatmap of the combination of all three groups: Elite, Skilled, Average.

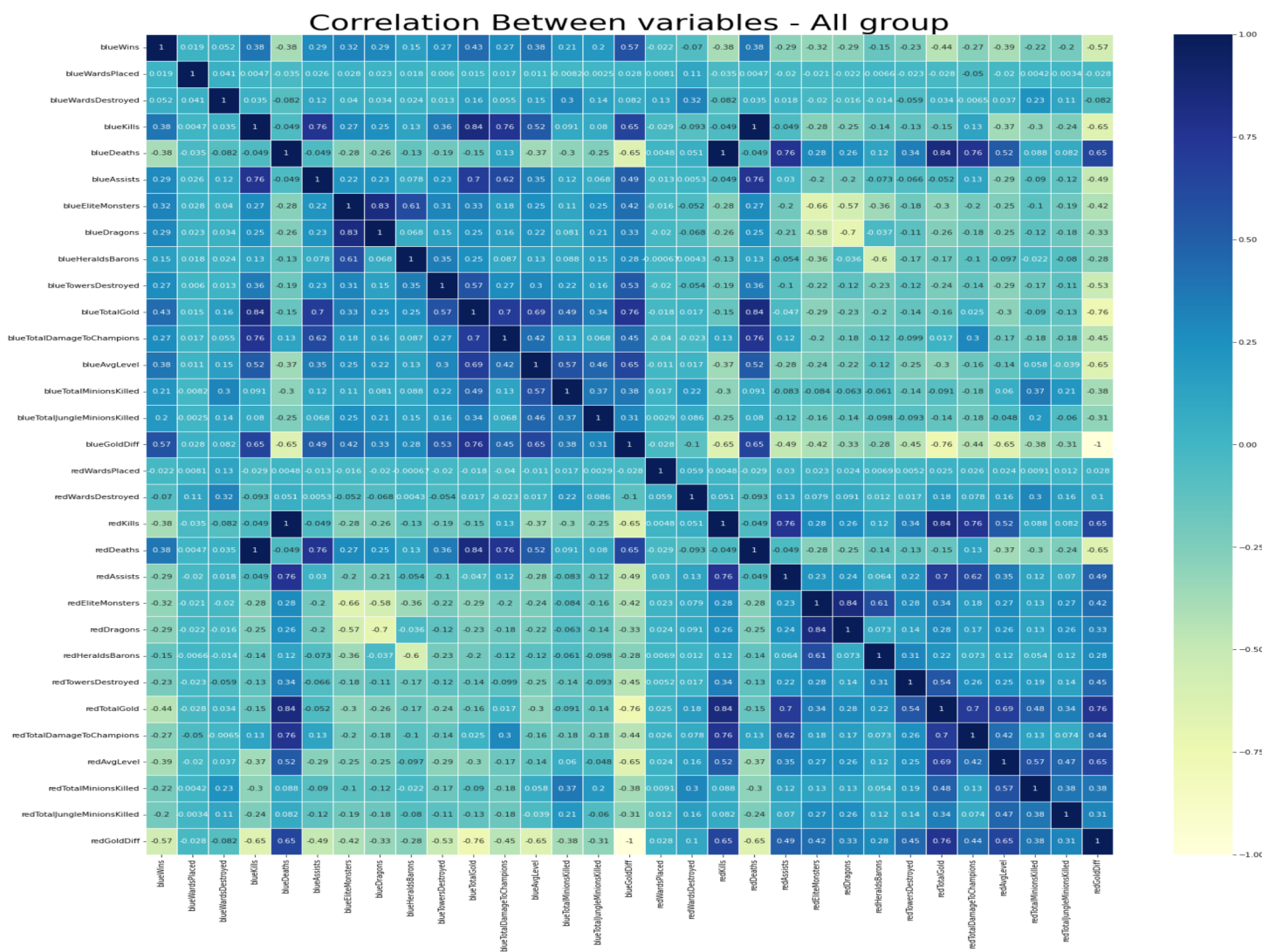


Figure 13: Correlation between all variables in the Old Variables group for all three groups combines (Elite, Skilled, Average)

In Figure 13, we can see that the correlations between some of the variables are noticeable. There is a positive correlation between total gold on each team and the resources collected by that team and vice versa where there is a negative correlation between the total gold and the enemy time resources. This makes sense because most resources are converted into gold to spend in the game to create more advantage to take more resources. Additionally, there is also a clear correlation between the “redGoldDiff”, gold different for red team, and all the variable containing resources for red team and vice versa for blue team. This also further indicates that gold is an important factor and that it is the core to gain other resources.

### 3.2 Models Overview

As mentioned before, I separated the original dataset into 10% testing data and the rest is used for validation and training purposes. After training and validating the models, I imported the testing data set that was held out of training and ran the set through the model that output the most accurate results.

The results contain accuracy, precision, recall, and F1 score. Accuracy is the percentage shown the model predicted correctly compared to the total number of predictions made (Accuracy, Precision, Recall or F1, 2022).

$$Accuracy = \frac{\# \text{ of correct predictions}}{\# \text{ of total predictions}}$$

*Equation 1: Accuracy formula*

The Precision shows how accurate the model is, which means how often the model is correct when it makes a prediction.

$$\textit{Precision} = \frac{\# \textit{ of True Positives}}{\# \textit{ of True Positives} + \# \textit{ of False Positives}}$$

*Equation 2: Precision formula*

The recall score is the percentage ability of the model to identify the correct answer in the given data set.

$$\textit{Recall} = \frac{\# \textit{ of True Positives}}{\# \textit{ of True Positives} + \# \textit{ of False Negatives}}$$

*Equation 3: Recall formula*

Lastly, the F1 score is a combination of the precision and recall score. It is a harmonic mean of precision and recall. It reaches its best value at 1 and worst at 0.

$$\textit{F1 score} = 2 * \frac{\textit{Precision} * \textit{Recall}}{\textit{Precision} + \textit{Recall}}$$

*Equation 4: F1 score formula*

### **3.2.1 Logistic Regression**

For Logistic Regression, I ran the model before I added the three non-linear combinations (blue/redAvgLevel, blue/redTotalDamageToChampions, blue/redKills) variables and, then again after I added the new variables, each time applying no penalty, L1, or L2 regularization.

The Logistic Regression model is a linear model for classification (Python Logistic Regression Tutorial, 2022). In the Scikit-learn Logistic Regression model, the probabilities that represent the possible results are modeled using a logistic function. The Logistic Function is a sigmoid curve with the equation:

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

*Equation 5: Logistic Function or Logistic Curve*

Where  $x_0$  is the  $x$  value of the sigmoid's midpoint;  $L$  is the curve's maximum value;  $k$  is the logistic growth rate or steepness of the curve.

Regularization is a procedure used to prevent overfitting issues (L1 and L2 Regularization Methods, 2022). It adds a regularization, or penalty, term to the logistic equation to stop overfitting the model. L1 regularization is called Lasso Regression and L2 is known as the Ridge Regression. Ridge Regression adds a squared magnitude of coefficient as a penalty term to the loss function:

$$\sum_{i=1}^n (Y_i - \sum_{j=1}^p X_{ij} \beta_j)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

*Equation 6: Cost Function with L2 regularization (Ridge Regression)*

Lasso Regression adds an absolute value of the magnitude of coefficient as a penalty term to the loss function:

$$\sum_{i=1}^n (Y_i - \sum_{j=1}^p X_{ij} \beta_j)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

*Equation 7: Cost Function with L1 regularization (Lasso Regression)*



Here,  $\lambda \sum_{j=1}^p \beta_j^2$  is the part that represents the L2 regularization element in Equation 6 and

$\lambda \sum_{j=1}^p |\beta_j|$  is the part that describes the L1 regularization component in Equation 7. The lambda is

important here because it will affect the weight if it is large and could lead to under-fitting the model. Hence, how lambda is chosen is important. The big difference between these two penalties is mentioned in the method of Logistic Regression, which is that the Lasso shrinks the less important feature's coefficient to zero so it would work better for feature selection in the case of a large number of features.

Afterward, I used the fold having the highest accuracy for the L1 regularization model to examine variable coefficients. This is used to determine the importance of the variable in the prediction model. A coefficient describes the size and direction of the relationship between the predictors and the dependent variable. It signifies how much the means of the response variable changes compared to the independent variable while other variables in the model are held constant. With the coefficient, we can see the relative importance of a variable when used by the model to predict classification.

The results are recorded as we take an average of each fold and then take the average of those with the three runs with no regularization, L1, and L2. The training, validation, and testing data results accuracy are shown in the tables below:

**Training Data (accuracy):**

<b>Group</b>	<b>Original variables (29 variables):</b>	<b>New variables added (32 variables):</b>
Elite	78.16 %	78.10 %

Skilled	76.22 %	76.19 %
Average	76.27 %	76.27 %

*Table 2: Logistic Regression's Accuracy training data result.*

**Validation Data (accuracy):**

<b>Group</b>	<b>Original variables (29 variables):</b>	<b>New variables added (32 variables):</b>
Elite	77.95 %	77.88 %
Skilled	76.00 %	76.02 %
Average	76.26 %	76.20 %

*Table 3: Logistic Regression's Accuracy validation data result.*

**Testing Data (accuracy):**

<b>Group</b>	<b>Original variables (29 variables):</b>	<b>New variables added (32 variables):</b>
Elite	79.25 %	79.15 %
Skilled	74.66 %	74.31 %
Average	76.97 %	77.62 %

*Table 4: Logistic Regression's Accuracy testing data result.*

The training and validation data accuracy results are averaged throughout the cases of no regularization, L1 regularization, and L2 regularization. There is no significant difference in the original variables' results and the new variables added results in all data sets and groups. On the other hand, the consistency of train/valid/test also indicates models are not overtrained or undertrained. The results for the testing data set are best for the Elite and Average group compared to the Skilled group which dropped 1.71 % compared to the validation data.

Additionally, for the original data set, the precision score was higher than the accuracy in all groups. The recall and F1 scores were not significantly different from the accuracy and did not have a different pattern.

The testing data results precision is shown in the below table:

**Testing Data (precision):**

<b>Group</b>	<b>Original variables (29 variables):</b>	<b>New variables added (32 variables):</b>
Elite	80.17 %	80.38 %
Skilled	77.36 %	76.85 %
Average	78.23 %	78.43 %

*Table 5: Logistic Regression's Precision testing data result.*

Overall, in all three groups for the new variables data set, the results for the Logistic regression model have an average accuracy of 77.03%, an average precision of 78.55%, and an average recall score of 75.32%, and an average F1 score of 76.89%. We can conclude that adding these three variables to the dataset did not make a significant impact on the result accuracy of the model.

**Coefficients:**

The fold has the highest accuracy in the L1 regularization model and takes the coefficient of the variables for each group (Elite, Skilled, Average). The coefficient describes the size and direction of the connection between a predictor and the dependent variable. They are the number in which the values of the term are multiplied in the regression equation. Hence, a positive coefficient indicates that when the predictor increases, the response variable also increases. On the other hand, a negative sign shows that when the predictor increases, the response variable

decreases. We use the coefficients to help determine the impact of a predictor variable on the classification. The coefficient of a variable represents the change in the link function for each unit change in the predictor variable, while the other variables in the data set are held the same as constant. Basically, positive and negative coefficients have an impact on the results of the events either positive or negative. But the closer to 0 the coefficient is the more it implies that the impact of the predictor variable on the results is insignificant.

The coefficients for three groups with the original variables are shown in the graphs below:

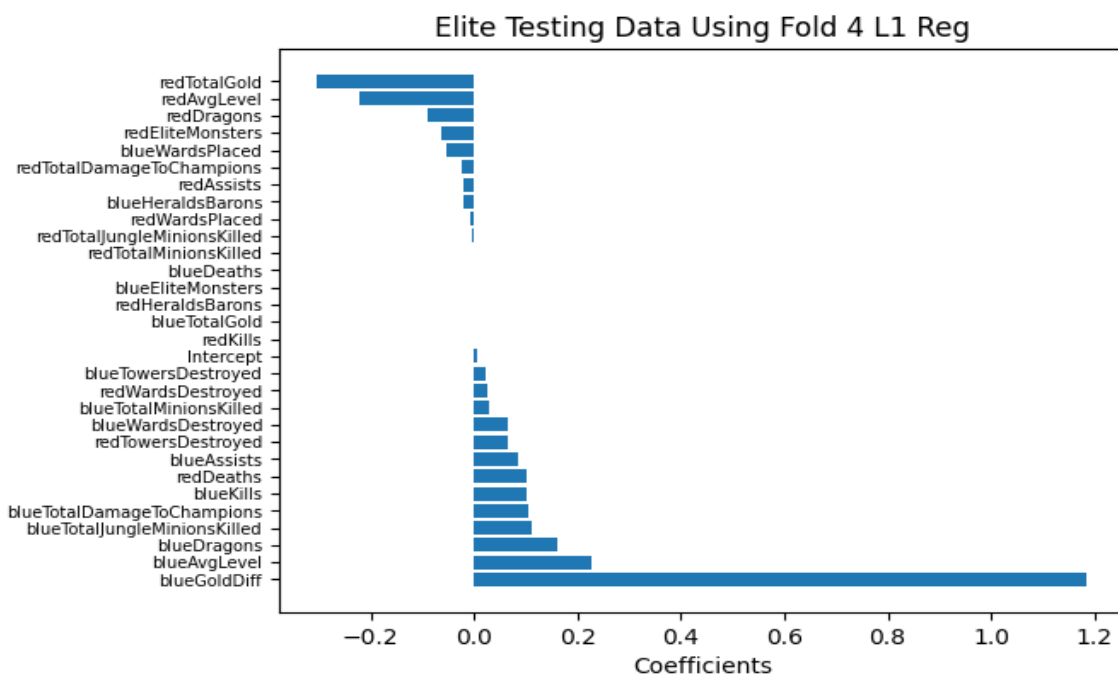


Figure 6: Elite group Logistic Regression with original testing data. The coefficient for Fold 4 with L1 regularization.



Figure 7: Skilled group Logistic Regression with original testing data. The coefficient for Fold 3 with L1 regularization.

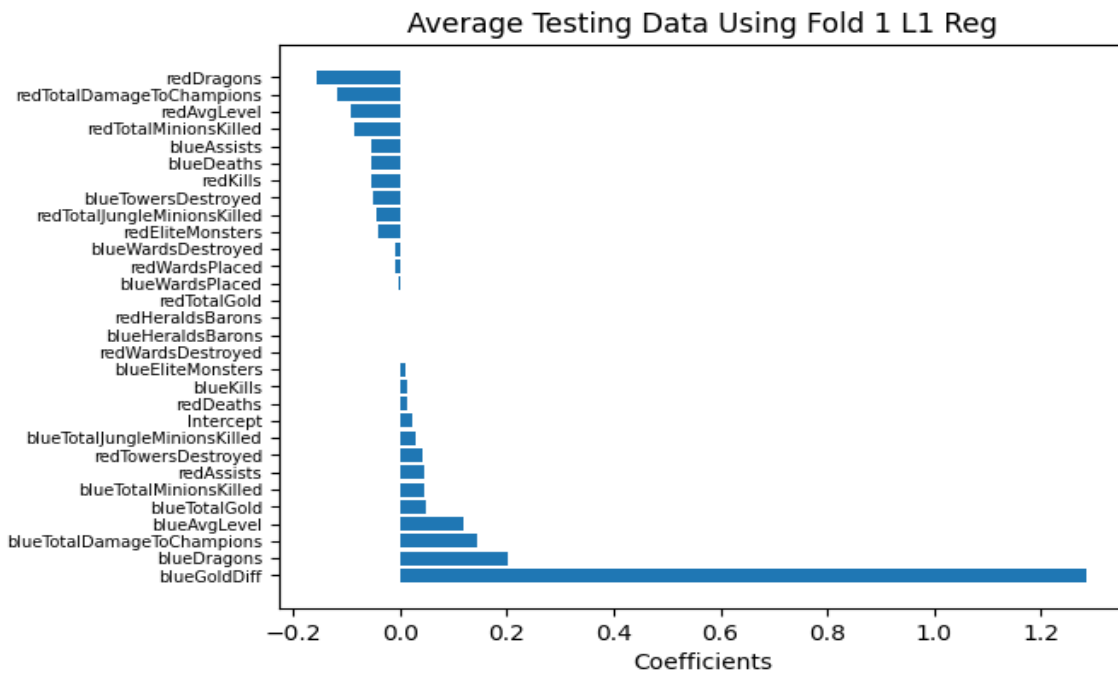


Figure 8: Average group Logistic Regression with original testing data. The coefficient for Fold 1 with L1 regularization.

In Figure 6, the results show in the Elite group with original data the coefficient that has the most impact is blueGoldDiff at 1.186 and the lowest impact is 0 for blueDeaths, blueEliteMonsters, blueTotalGold, redKills, and redHeraldsBarons. The second most impactful coefficient is redTotalGold at -0.3. Even though the blueTotalGold has a coefficient of 0, redTotalGold is -0.3. This means that the total gold from each team might have an impact but the difference between them is more important. Next, the coefficient results of the Skilled testing with original data with a similar highest coefficient is the blueGoldDiff at 0.997 and the second highest is redAvgLevel at -0.177. The figure shows blueHeraldsBarons, blueTotalGold, and redHeraldsBarons at 0 coefficient. Lastly, Figure 8 represents the average groups with original data with the highest coefficient at 1.286 is blueGoldDiff and the second-highest coefficient is blueDragons at 0.2. The variables that have no impact with a zero coefficient are blueHeraldsBarons, redTotalGold, and redHeraldsBarons.

The highest coefficient for the original testing data for all three groups is the same variable: blueGoldDiff. The variable coefficient for blueGoldDiff is significantly higher in all groups compared to all the other variables. The difference between blueGoldDiff coefficients to the next highest coefficient is 0.88 for the Elite group, 0.82 for the Skilled group, and 1.08 for the Average group. This variable represents the gold difference between two teams. This shows that the difference in gold between the two teams plays a major factor in the end results of the match. Since gold is the resource to use in the game to easily obtain other resources and it is the easiest resource to use and take advantage of, this makes sense. We also notice that variables like blueHeraldsBarons, and redHeraldsBarons have 0 coefficients in all three groups. These variables are not important in the process of predicting the results of the match. This means objectives Heralds and Barons have little to no impact on the game results before the 14-minute

mark. Additionally, all the variables associated with the red team have a negative impact on the results, as the predicted variable here is whether the blue team wins or not.

The coefficients for three groups with the newly added variables are shown in the figures below:



Figure 9: Elite group Logistic Regression with new variable added testing data. The coefficient for Fold 4 with L1 regularization.

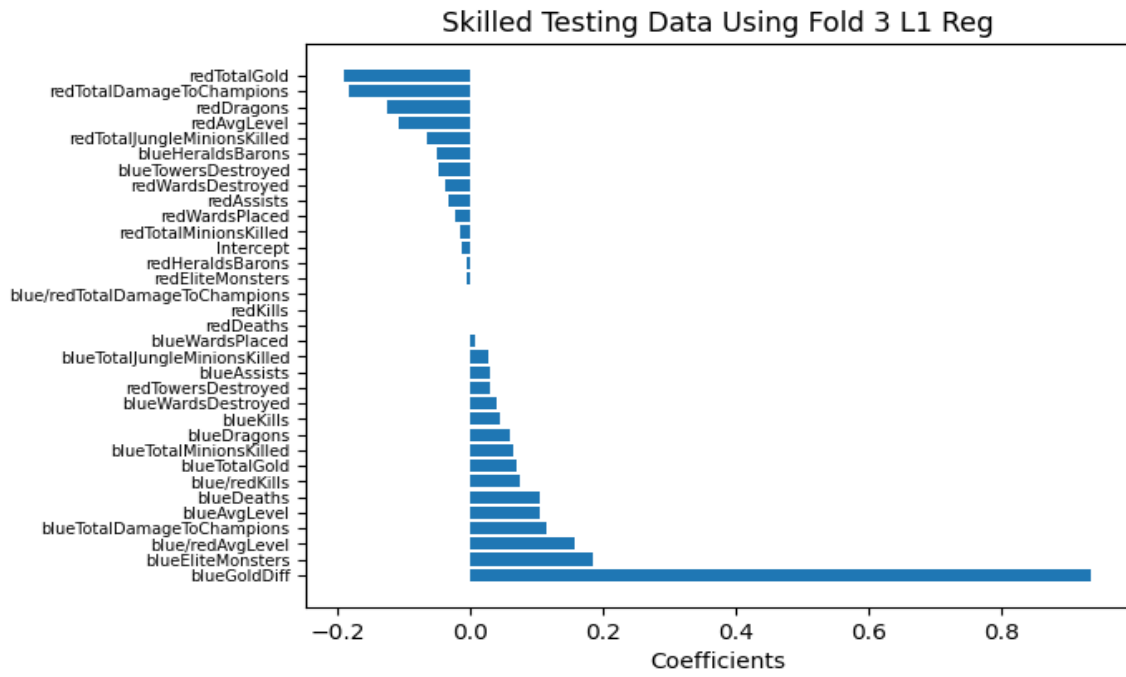


Figure 10: Skilled group Logistic Regression with new variable added testing data. The coefficient for Fold 3 with L1 regularization.

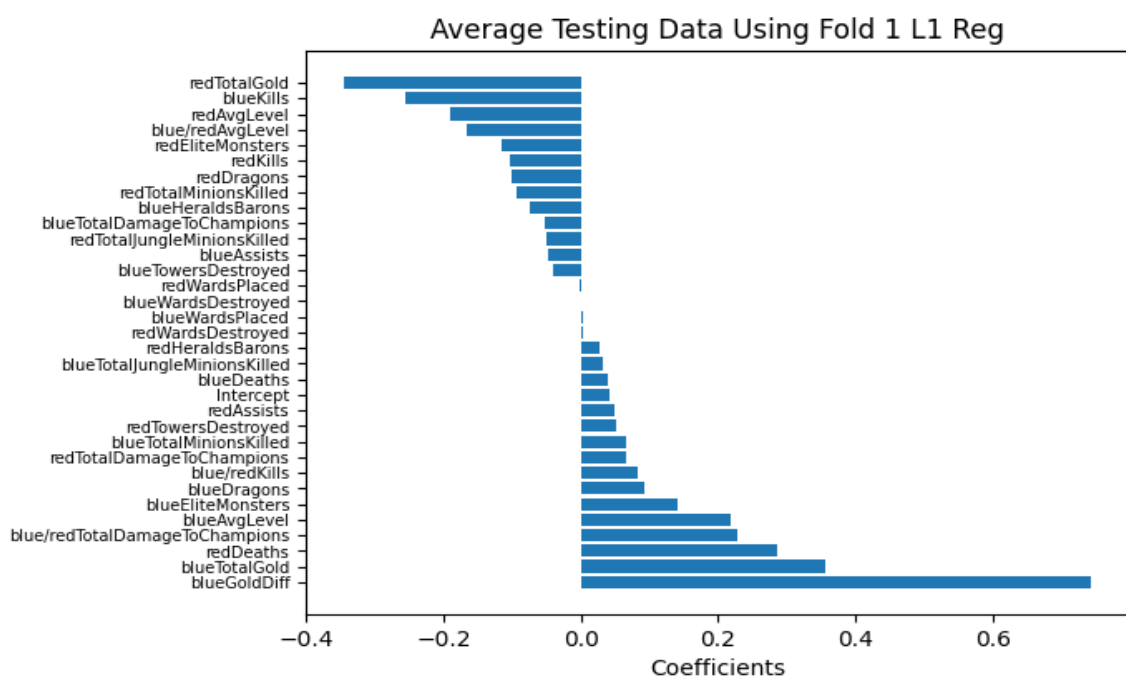


Figure 11: Average group Logistic Regression with new variable added testing data. The coefficient for Fold 1 with L1 regularization



In Figure 9, the results show in the Elite group with the newly added variable the coefficient that has the most impact is blueGoldDiff at 0.6. The second most impactful coefficient is redTotalGold at -0.519, which is not that much of a difference from the highest variable. The lowest impact is 0 for blueHeraldsBarons, blueDeaths, redAvgLevel, and redHeraldsBarons. Moreover, the coefficient results of the Skilled testing with newly added data are shown in Figure 10 with the highest coefficient being the blueGoldDiff at 0.936 and the second highest is redTotalGold at 0.357. The graph shows redKills, redDeaths, and blue/redTotalDamageToChampions is at little to 0 coefficients. Finally, Figure 11 indicates that the average group with newly added data with the highest coefficient at 0.745 is blueGoldDiff and the second-highest coefficient is blueTotalGold at 0.357. However, there are no 0 coefficient variables in this model meaning that every variable contributes to the final result of the prediction.

Overall, in the three groups, adding new variables does not change the most significant variable, which remains blueGoldDiff. This indicates that after adding the new variable the importance of the gold difference between the two teams to the results has not changed. However, the coefficient for the Elite and Average group is more spread out as most variables have some significantly close to the highest coefficient. This shows that adding the three new variables has helped the model balance the importance of each variable without making a big impact on the final accuracy of the model.

### **3.2.2 Random Forest**

The Random Forest model is also trained separately with the three groups: Elite, Skilled, and Average. I only used the newly added variable data set for this model. Afterward, I also take

the feature importance also known as variable importance of the model. This method is a built-in Scikit-learn implementation of the Random Forest classifier.

The Random Forest model is constructed with a set of decision trees, and each tree is a set of internal nodes and leaves (1.10. Decision Trees, 2022). In the internal node, the selected variable or feature is used to make decisions on how to separate the data set into two or more with similar decisions. These features are selected by some criterion which in the case of classification would be information gained from the decision. The feature importance is collected by the average importance of all trees in the forest. This lets us know what variables are important during the decision-making process of predicting the results, meaning determining the feature attribute the most to the predictive power of the model.

I ran the model multiple times with different parameters to find the most efficient model with the right parameters. The differences between the results were not significant. Figure 4 and Table 10 below show the different parameters used. As shown, changing all the variables: `nEstimators`, `minSampleSplit`, `minSampleLeaf`, `maxFeatures`, does not make a significant difference in the result of the model. Hence, I use the parameters that return the highest accuracy.

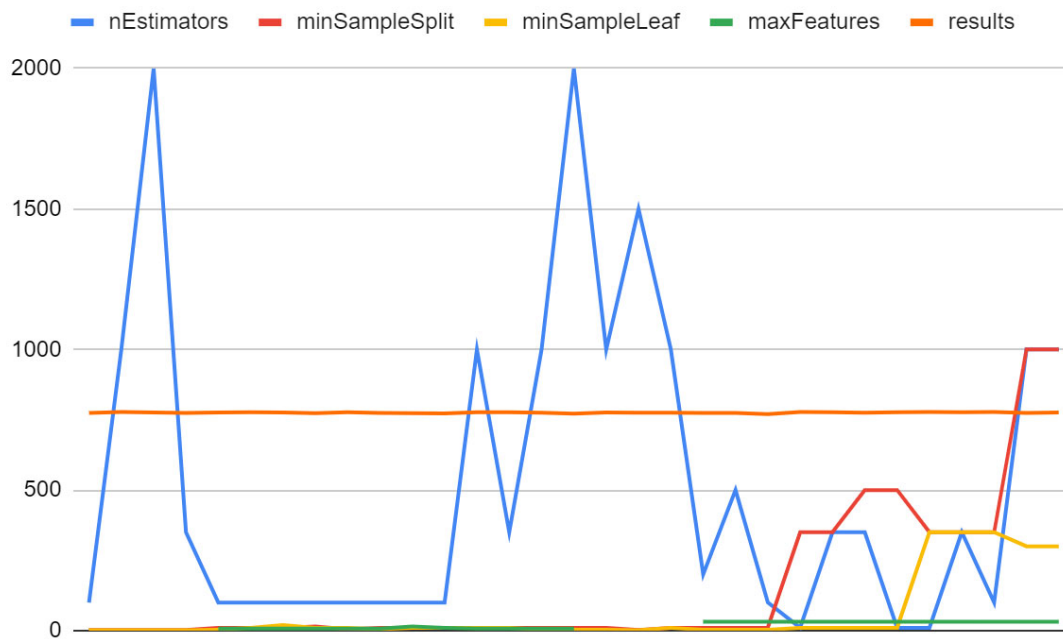


Figure 4: Parameters for optimizing Random Forest model (results in the table max at 1000%)

<b>nEstimators</b>	<b>minSample Split</b>	<b>minSample Leaf</b>	<b>maxFeatu res</b>	<b>results</b>
1000	2	1	auto	77.789
10	350	10	32	77.766
10	350	350	32	77.755
100	350	350	32	77.743
350	350	350	32	77.72
100	10	10	7	77.708
10	500	10	32	77.674
100	5	10	7	77.662
350	10	10	7	77.662
1000	10	10	7	77.65
350	350	10	32	77.65

1000	10	3	auto	77.627
100	10	3	7	77.604
100	10	20	7	77.604
2000	2	1	auto	77.593
1000	1000	300	32	77.569
350	500	10	32	77.546
1000	10	3	7	77.523
1500	2	1	auto	77.523
1000	10	10	auto	77.488
200	10	3	32	77.442
1000	1000	300	32	77.442
100	2	1	auto	77.43
350	2	1	auto	77.43
100	10	5	7	77.43
500	10	3	32	77.419
100	15	10	7	77.384
100	10	10	15	77.326
100	10	10	10	77.303
2000	10	3	7	77.21
100	10	3	32	77.002

*Table 10: Parameters for optimizing Random Forest model (results in the table max at 100%)*

Finally, the most effective parameters were: `n_estimators = 1000`, `min_samples_split= 2`, `min_sample_leaf= 1`, `max_features = 'auto'`. The training, validation, and testing data results of accuracy, which are averaged over the five folds, are shown in Table 6 below:

<b>Group</b>	<b>Training</b>	<b>Validation</b>	<b>Testing</b>
Elite	100 %	77.50 %	79.04 %

Skilled	100 %	75.60 %	79.15 %
Average	100 %	75.89 %	79.04 %

*Table 6: Random Forest's Accuracy training, validation, and testing data result.*

In Table 6, the training model looks overfitted with 100% accuracy in every group, but the validation and testing give consistent results. The validation for all three groups is slightly lower than the testing results. However, the difference between them is not significant where for the Elite group, validation accuracy is 77.5% and testing is 79.04%; the Skilled group validation accuracy is 75.6% and testing is 79.15%; and lastly, the Average group accuracy is 75.89% and testing is 79.04%. There is no significant difference between the testing and validation results from all three groups.

The precision, recall, and F1 score of the three groups for testing results are shown in Table 7.

<b>Group</b>	<b>Precision</b>	<b>Recall</b>	<b>F1 Score</b>
Elite	79.04 %	78.69 %	79.26 %
Skilled	81.44 %	76.43 %	78.86 %
Average	79.30 %	78.89 %	79.30 %

*Table 7: Random Forest's Precision, Recall, and F1 Score for testing data result.*

Table 7 shows that the precision score for the Skilled group is the highest among the three groups. This means that the model for the Skilled group is correct 81.44 % of the time when it makes a prediction. The Elite and the Average group have similar scores for precision at 79.04% and 79.3%. The table also shows that the Recall and F1 score does not have any meaningful difference between the groups.

**Feature Importances:**

I used the ‘feature\_importances\_’ attribute in the built-in Scikit-learn's ensemble Random Forest Classifier to find the important variable of the model for each group. The results will show us which variables have the most and least significant impact on the results of the predictor. For each group, I choose the highest accuracy testing result fold and select that fold to get the model feature importance.

The Feature Importance of the three groups with the Random Forest model is shown in Figure 12.

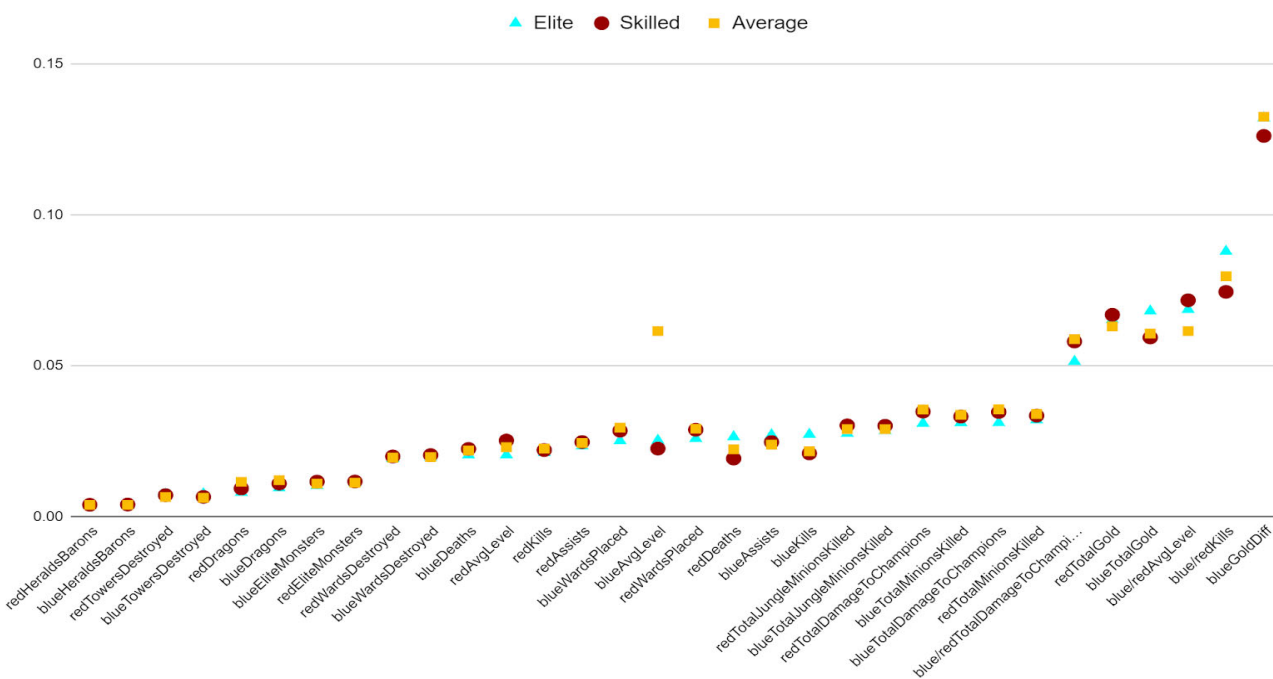


Figure 12: Elite (Fold 4), Skilled (Fold 2), and Average (Fold 1) group Random Forest testing data Feature Importance.

In all three groups, the most important variable that impacts the results is blueGoldDiff. This result is similar to the Logistic Regression variable coefficients where the gold difference

between the two teams has the most impact on the predictor. The second most important feature in all three groups is also the same which is the blue/redKills. This variable is the ratio between the total kills on each team.

Figure 12 suggests that the top six variables in all the groups are the same: blueGoldDiff, blue/redKills, blue/redAvgLevel, blueTotalGold, redTotalGold, blue/redTotalDamageToChampions. These variables have the Feature Importance scores ranging from 0.051 to 0.133. Other variables for all three groups range from 0.36 and below. Furthermore, not only are the most two important variables the same in all groups but the other four in the top six important variables are a mix of the same variables: redTotalGold, blueTotalGold, blue/redAvgLevel, blue/redTotalDamageToChampions. The variables “redTotalGold” and “blueTotalGold” indicate that the total gold from each team is impactful to the results. The “blue/redAvgLevel” is the ratio between team blue and red average levels. This implies that not only gold but experience, which is the factor that increases character level, also plays an important role in the decision-making of predicting the result. The “blue/redTotalDamageToChampions” is the ratio between blue and red team damage to champions, which is the total damage every five players on each team did to any of the opponent players. That means the more fighting or damage a team does to one another in the first 14-minute also plays a role in determining the end result.

Finally, from the figures, we also see that the six least impactful variables are all the same in all three groups: redHeraldsBarons, blueHeraldsBarons, blueTowersDestroyed, redTowersDestroyed, blueEliteMonsters, redEliteMonsters. All of these variables represent the total number of objectives (Heralds, Barons, Towers, Elite Monsters) killed for each team. These results make sense since at the 14-minute mark objectives are usually not the focus of the game.

They serve more like a bonus purpose or side goals for the team. However, the consistency of the level of importance of the variable for all groups, meaning features of importance are similar in all three groups, is unexpected. As the three groups represent three different skill levels, I would anticipate that the feature importance would be somewhat different for each group. But in this Random Forest model that is not the case.

### 3.2.3 Support Vector Machine (SVM)

The Support Vector Machine (SVM) model is trained to predict wins for the three main groups: Elite, Skilled, and Average. I use Scikit-learn's Support Vector Classification with a 'linear' kernel as mentioned in the Methods. I only used the newly added variable data set for this model. The table below shows the accuracy of the training, validation, and testing data which is taken averaged over the five folds:

<b>Group</b>	<b>Training</b>	<b>Validation</b>	<b>Testing</b>
Elite	78.24 %	78.00 %	79.15 %
Skilled	76.14 %	75.99 %	79.46 %
Average	76.29 %	76.24 %	78.73 %

*Table 8: SVM's Accuracy training, validation, and testing data result.*

In Table 8, the training and validation for all three groups do not have any significant difference, where the testing data yielded better results compared to the training and validation. For the Elite group, training and validation accuracy is 78.24% and 78.00%, whereas testing is 79.15% (around a 1.15 % difference). For the Skilled group, training and validation accuracy is 76.14% and 75.99%, whereas testing is 79.46% (about a 3.47% difference). Lastly, the Average group training and validation accuracy is 76.29% and 76.24%, whereas testing is 78.73% (about a 2.49% difference). Overall, the result is moderately better for testing data.



The precision, recall, and F1 score of the three groups for testing results are shown in the below table:

<b>Group</b>	<b>Precision</b>	<b>Recall</b>	<b>F1 Score</b>
Elite	80.00 %	78.48 %	79.30 %
Skilled	80.94 %	78.29 %	79.50 %
Average	78.73 %	79.10 %	79.10 %

*Table 9: SVM's Precision, Recall, and F1 Score for testing data result.*

Table 7 demonstrates that the precision score for the Elite and the Skilled group have similar scores for precision at 80.00% and 80.94%. This means that these models for the Elite and Skilled group are correct 80.00% and 80.94% of the time when it makes a prediction. The Average group has the lowest precision score among the three groups at 78.73%. The table also shows that the Recall and F1 score does not have any meaningful difference among the groups.

### **3.3 All Models Summary**

Three machine learning models are tested with the newly added variable data set and all three model types yield rather similar results. In Table 11, the Random Forest and SVM models have more consistent accuracy results for all three groups: Elite, Skilled and Average. The Logistic Regression model did as well on the Elite group but for the Skilled and Average group, the accuracy falls short compared to the other two models. The SVM and Logistic Regression tied for the highest accuracy results for the Elite group at 79.15%. The SVM did a better job for the Skilled group with the highest accuracy of 79.15%. Finally, the Random Forest has the highest accuracy for the Average group at 78.73%. The Random Forest showed 100% accuracy in the training set compared to Logistic Regression and SVM which has generalization results for training and testing. On the other hand, both Logistic Regression and Random Forest had the

‘coefficient’ and ‘feature importance’ attributes available so we could determine the most relevant behavior and variables for predicting the results.

<b>Group</b>	<b>Logistic Regression</b>	<b>Random Forest</b>	<b>SVM</b>
Elite	<b>79.15 %</b>	79.04 %	<b>79.15 %</b>
Skilled	74.31 %	79.15 %	<b>79.46 %</b>
Average	77.62 %	<b>79.04 %</b>	78.73 %

*Table 11: Testing data results Accuracy for Logistic Regression, Random Forest, and SVM for newly added variable data set.*

## Chapter 4: Discussion

### 4.1 Reflection

In this paper, I worked with an established Riot API to pull data from the first 14 minutes of games and used that data to predict League of Legends game outcomes. Using the Riot API has given me more information and knowledge about using an established API to extract data and using that data to get the specific data that I need in the API. Working with this data and API is also very interesting to me as I am passionate about this topic. The data collecting process was interesting yet frustrating at the same time as the API was slow to collect the data with limited calls per minute being made.

In the results, we did establish an accuracy of 79.56%, more than 4% better than the results reported in the research mentioned in Section 1.5 , which had the highest accuracy of 75.1% ( Do et al., 2021) in which they used pre-match data to predict the outcome. Furthermore, our models compared to a similar study by Cabrera that we discuss in the 1.5 section was overall generally better (Cabrera, n.d.). This study uses data from the 10-minute mark and had similar data with slightly difference representation of the data and also has similar model selections. Our results for the Logistic Regression model is 8.54% higher with theirs being 70.61%. Additionally, our Random Forest highest accuracy result is 79.15% which is 11.93% higher compared to theirs 67.22%. And for their SVM, they did not report the results but mentioned that SVM showed poor performance on the test samples and was not as good as their other models, which for our results, SVM have the highest accuracy at 79.46%.

In the Logistic Regression and Random Forest results, we identified some of the important variables that help players decide what aspect has the most value to winning the game. In both models, the most impactful feature is the gold difference between the two teams. This is because, just as explained in Section “1.2 Introduction to League of Legends”, most of everything obtained in the game is converted to gold and experience. Basically, the more gold you have and the more gold you deny your opponent, the more likely you are to win. However, in the Logistic Regression model, the coefficient for each group varies depending on the skill level which is expected. However, for Random Forest feature importance, the three groups have similar variable importance levels, which is surprising as each group has different skills, which would have impacted their decision-making during the game. This would lead to the expectation that the features’ importance in each group would somewhat be different. This is because, depending on the skill level, players will tend to consider the importance of each objective, or variable in the game more or less important, hence focusing more on those specific variables. This was not the case for Random Forest which could mean that even though the skills differences vary, the core value of the game in the early 14-minute mark could have the same structure of importance for each type of object or variable in the game.

#### **4.2 Improvement**

Using the Voting Classifier to help combine the models to improve the accuracy of the results is one of the models that I was hoping to try but ended up not because I did not have enough time. The idea is to potentially see if the results predicted by the models are correct for the same data. If they predict correct on different data, we can then combine the models to hopefully improve the performance of the models.

Further studies can be conducted to improve the predictive results of this model by adding new aspects to the study. Because this study was only to find and predict which team is the winning team, only general information at the 14-minute mark regarding the objectives was used and collected. Furthermore, information could be collected and used such as data about each role and position difference between the two teams. We can use that information to compare the difference between the same role on each team to find the importance of each position and explore the variable that is impactful for each position.

Also, data on specific team composition (pre-match data) could also be used to explore the impact of champion selections and how team composition impacts the win rate of the match. Similarly, an analysis of players' statistics like rank or champions mastery (how familiar they are with the chosen champion) would help improve the accuracy of prediction of the match results, and how much would that impact the results.

Finally, the additional information about the whole game instead of only collecting 14 minutes of data would also contribute to the limitation of exploring important features and what actually impacts the game's final result.

## Reference

- 1.1. *Linear Models — scikit-learn 1.0.2 documentation.* (n.d.). Retrieved April 19, 2022, from [https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
- 1.4. *Support Vector Machines — scikit-learn 1.0.2 documentation.* (n.d.). Retrieved April 19, 2022, from <https://scikit-learn.org/stable/modules/svm.html#classification>
- 1.10. *Decision Trees — scikit-learn 1.0.2 documentation.* (n.d.). Retrieved April 12, 2022, from <https://scikit-learn.org/stable/modules/tree.html>
- A Map of the League of Legends game play in the classic mode.* | *Download Scientific Diagram.* (n.d.). Retrieved October 22, 2021, from [https://www.researchgate.net/figure/A-Map-of-the-League-of-Legends-game-play-in-the-classic-mode\\_fig1\\_319839481](https://www.researchgate.net/figure/A-Map-of-the-League-of-Legends-game-play-in-the-classic-mode_fig1_319839481)
- Accuracy, Precision, Recall or F1?* | *by Koo Ping Shung* | *Towards Data Science.* (n.d.). Retrieved April 19, 2022, from <https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9>
- Bediou, B., Adams, D. M., Mayer, R. E., Tipton, E., Green, C. S., & Bavelier, D. (2018). Meta-analysis of action video game impact on perceptual, attentional, and cognitive skills. *Psychological Bulletin*, 144(1), 77–110. <https://doi.org/10.1037/BUL0000130>
- Building A Logistic Regression in Python, Step by Step* | *by Susan Li* | *Towards Data Science.* (n.d.). Retrieved April 12, 2022, from <https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8>

- Bunker, R. P., & Thabtah, F. (2019). A machine learning framework for sport result prediction. *Applied Computing and Informatics*, 15(1), 27–33.  
<https://doi.org/10.1016/J.ACI.2017.09.005>
- Cabrera, M. S. (n.d.). *Predicting the winner of a League of Legends match at the 10-Minute Mark Capstone Project Machine Learning Engineer Nanodegree*.
- Coefficients for Binary Logistic Regression - Minitab Express*. (n.d.). Retrieved April 16, 2022, from  
<https://support.minitab.com/en-us/minitab-express/1/help-and-how-to/modeling-statistics/regression/how-to/binary-logistic-regression/interpret-the-results/all-statistics-and-graphs/coefficients/>
- Decision Tree Algorithm, Explained - KDnuggets*. (n.d.). Retrieved April 25, 2022, from  
<https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>
- Do, T. D., Wang, S. I., Yu, D. S., McMillian, M. G., & McMahan, R. P. (2021). Using Machine Learning to Predict Game Outcomes Based on Player-Champion Experience in League of Legends. *ACM International Conference Proceeding Series*.  
<https://doi.org/10.1145/3472538.3472579>
- Feature importances with a forest of trees — scikit-learn 1.0.2 documentation*. (n.d.). Retrieved April 12, 2022, from  
[https://scikit-learn.org/stable/auto\\_examples/ensemble/plot\\_forest\\_importances.html](https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html)
- Hodge, V., Devlin, S., Sephton, N., Block, F., Cowling, P., & Drachen, A. (2019). Win Prediction in Multi-Player Esports: Live Professional Match Prediction. *IEEE Transactions on Games*, 1–1. <https://doi.org/10.1109/TG.2019.2948469>

*Horizontal bar chart* — *Matplotlib 3.5.1 documentation*. (n.d.). Retrieved April 12, 2022,

from [https://matplotlib.org/stable/gallery/lines\\_bars\\_and\\_markers/barh.html](https://matplotlib.org/stable/gallery/lines_bars_and_markers/barh.html)

*How many people play League of Legends in 2021? [League of Legends player count]*.

(n.d.). Retrieved October 19, 2021, from

<https://techacake.com/league-of-legends-player-count/>

*How to Calculate Feature Importance With Python*. (n.d.). Retrieved April 12, 2022, from

<https://machinelearningmastery.com/calculate-feature-importance-with-python/>

*How To Use Riot API With Python. In this tutorial, I will use Riot API... | by Barney H. |*

*Towards Data Science*. (n.d.). Retrieved October 24, 2021, from

<https://towardsdatascience.com/how-to-use-riot-api-with-python-b93be82dbbd6>

*L1 and L2 Regularization Methods. Machine Learning | by Anuja Nagpal | Towards Data*

*Science*. (n.d.). Retrieved April 19, 2022, from

<https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c>

*League of Legends - Wikipedia*. (n.d.). Retrieved October 19, 2021, from

[https://en.wikipedia.org/wiki/League\\_of\\_Legends](https://en.wikipedia.org/wiki/League_of_Legends)

*League of Legends in esports - Wikipedia*. (n.d.). Retrieved October 20, 2021, from

[https://en.wikipedia.org/wiki/League\\_of\\_Legends\\_in\\_esports](https://en.wikipedia.org/wiki/League_of_Legends_in_esports)

*Logistic function - Wikipedia*. (n.d.). Retrieved April 19, 2022, from

[https://en.wikipedia.org/wiki/Logistic\\_function](https://en.wikipedia.org/wiki/Logistic_function)

*LOL Match Prediction Using Early Laning Phase Data | Machine Learning | by Jinhang*

*Jiang | Towards Data Science*. (n.d.). Retrieved October 22, 2021, from

<https://towardsdatascience.com/lol-match-prediction-using-early-laning-phase-data-machine-learning-4c13c12852fa>



*LoL Ranks 2020. League of Legends Ranking System Explained.* (n.d.). Retrieved April 25, 2022, from <https://egb.com/blog/league-of-legends-ranking-system-explained>

*pandas.plotting.scatter\_matrix — pandas 1.4.2 documentation.* (n.d.). Retrieved April 25, 2022, from

[https://pandas.pydata.org/docs/reference/api/pandas.plotting.scatter\\_matrix.html](https://pandas.pydata.org/docs/reference/api/pandas.plotting.scatter_matrix.html)

*Predicting e-sports winners with machine learning | by Bowen Yang | Insight.* (n.d.).

Retrieved October 20, 2021, from

<https://blog.insightdatascience.com/hero2vec-d42d6838c941>

Przybylski, A. K., & Wang, J. C. (2016). A large scale test of the gaming- enhancement hypothesis. *PeerJ*, 2016(11). <https://doi.org/10.7717/peerj.2710>

*Python Logistic Regression Tutorial with Sklearn & Scikit - DataCamp.* (n.d.). Retrieved April 12, 2022, from

<https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python>

*Random Forest Feature Importance Computed in 3 Ways with Python | MLJAR.* (n.d.).

Retrieved April 12, 2022, from

<https://mljar.com/blog/feature-importance-in-random-forest/>

*Rank (League of Legends) | League of Legends Wiki | Fandom.* (n.d.). Retrieved October 22, 2021, from [https://leagueoflegends.fandom.com/wiki/Rank\\_\(League\\_of\\_Legends\)](https://leagueoflegends.fandom.com/wiki/Rank_(League_of_Legends))

Reynaldo, C., Christian, R., Hosea, H., & Gunawan, A. A. S. (2021). Using Video Games to Improve Capabilities in Decision Making and Cognitive Skill: A Literature Review. *Procedia Computer Science*, 179, 211–221.

<https://doi.org/10.1016/J.PROCS.2020.12.027>

*Riot Developer Portal*. (n.d.). Retrieved April 25, 2022, from

<https://developer.riotgames.com/>

*seaborn.heatmap* — *seaborn 0.11.2 documentation*. (n.d.). Retrieved April 25, 2022, from

<https://seaborn.pydata.org/generated/seaborn.heatmap.html>

*sklearn.ensemble.RandomForestClassifier* — *scikit-learn 1.0.2 documentation*. (n.d.).

Retrieved April 12, 2022, from

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

*sklearn.linear\_model.LogisticRegression* — *scikit-learn 1.0.2 documentation*. (n.d.-a).

Retrieved April 19, 2022, from

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LogisticRegression.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

*sklearn.linear\_model.LogisticRegression* — *scikit-learn 1.0.2 documentation*. (n.d.-b).

Retrieved April 12, 2022, from

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LogisticRegression.html#sklearn.linear\\_model.LogisticRegression.predict](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.predict)

*sklearn.svm.SVC* — *scikit-learn 1.0.2 documentation*. (n.d.). Retrieved April 19, 2022, from

<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC>

*SVM | Support Vector Machine Algorithm in Machine Learning*. (n.d.-a). Retrieved April

25, 2022, from

<https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>

*SVM | Support Vector Machine Algorithm in Machine Learning*. (n.d.-b). Retrieved April 19, 2022, from

<https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>

*The F1 score | Towards Data Science*. (n.d.). Retrieved April 12, 2022, from

<https://towardsdatascience.com/the-f1-score-bec2bbc38aa6>

*What Is a Decision Tree?* (n.d.). Retrieved April 12, 2022, from

<https://www.mastersindatascience.org/learning/introduction-to-machine-learning-algorithms/decision-tree/>

*Why Use Ensemble Learning?* (n.d.). Retrieved April 19, 2022, from

<https://machinelearningmastery.com/why-use-ensemble-learning/>

Li, K., & Fu, M. (2021). Effectiveness of Machine Learning Methods on Gaming

Recommendation and Prediction. *ACM International Conference Proceeding Series*,

*PartF168982*. <https://doi.org/10.1145/3448734.3450491>

## Appendix

“[https://github.com/linhchi33a/NgocLinhChiNguyen\\_SeniorProject2022.git](https://github.com/linhchi33a/NgocLinhChiNguyen_SeniorProject2022.git)”

This is a link to Github that contains all files I made for my senior project. The github link contains data collected, data filtered, modified data in the “DataCollection” folder. This folder has two sub folders “Importantfiles” where are the official important data and Jupyter Notebook is kept and the “Temporaryfiles” where I keep all the excels files that is used temporarily to merge into bigger files. The links also have a folder called “Graph and Data” which contains pngs of all the graphs I made. Finally, the “MachineLearningModels” contains Jupyter Notebook files and html files of the machine models I ran for this study.