**ISTANBUL TECHNICAL UNIVERSITY ★ FACULTY OF MANAGEMENT**

# AI APPLICATION FOR PICKING HEROES IN LEAGUE OF LEGENDS

**B.Sc. THESIS**

**Gamze Nur KARAGÖZ**
**Begüm ÜSTÜN**

**Department of Industrial Engineering**

**FEBRUARY 2021**

**ISTANBUL TECHNICAL UNIVERSITY ★ FACULTY OF MANAGEMENT**

# AI APPLICATION FOR PICKING HEROES IN LEAGUE OF LEGENDS

**B.Sc. THESIS**

**Gamze Nur KARAGÖZ**
**070160106**
**Begüm ÜSTÜN**
**070160044**

**Department of Industrial Engineering**

**Thesis Advisor: Dr. Hüseyin Kutay Tinç**

**FEBRUARY 2021**

**ISTANBUL TEKNİK ÜNİVERSİTESİ ★ İŞLETME FAKÜLTESİ**

# LEAGUE OF LEGENDS ŞAMPİYON SEÇİMİNDE YAPAY ZEKA UYGULAMALARI

**LİSANS TEZİ**

**Gamze Nur KARAGÖZ**
**070160106**
**Begüm ÜSTÜN**
**070160044**

**Endüstri Mühendisliği Anabilim Dalı**

**Tez Danışmanı: Dr. Hüseyin Kutay TİNÇ**

**ŞUBAT 2021**

***To our family,***

**FOREWORD**

This thesis is written for Graduation Thesis at the Istanbul Technical University. Since the gaming community is growing day by day drastically, we as an engineering students thought we can contribute them with our study and knowledge. We also realized that there are lack of studies in this field so our main purpose in this thesis is to guide e-spor players at their gaming experiences. I would like to thank in the first place to my project advisor Hüseyin Kutay Tinç Ph.D. and all my friends for their valuable advices and emotional supports.

February 2021

Begüm ÜSTÜN
(Industrial Engineering Student)

Gamze Nur Karagöz
(Industrial Engineering Student)

**TABLE OF CONTENTS**

**Page**

**ABBREVIATIONS**

**LOL** **:** League of Legends
**MOBA** **:** Multiple Online Battle Arena
**XP** **:** Experience
**RP** **:** Riot Points
**GG** **:** Good Game
**AI** **:** Artificial Intelligence
**RFE** **:** Recursive Feature Elimination
**GBT** **:** Gradient Boosted Trees
**LR** **:** Logistic Regression
**RNN** **:** Recurrent Neural Networks
**LSTM** **:** Long Short-Term Memory
**GRU** **:** Gated Recurrent Units
**PPO** **:** Proximal Policy Optimization
**DOTA2** **:** Defense of the Ancients 2
**SVM** **:** Support Vector Machine
**AHP** **:** Analytic hierarchy process
**TOPSIS** **:** Technique for Order of Preference by Similarity to Ideal Solution
**PROMETHEE:** Preference ranking organization method for enrichment evaluation

**SYMBOLS**

**C** : Current choice

**N** : Choice number

**T** : The team which make champion selection

$\mathbf{C_t}$ : Choice types which are ban or pick

$\mathbf{C_{i-1}}$ : The earlier champion selection

**LIST OF TABLES**

**Page**

**LIST OF FIGURES**

**Page**

## AI APPLICATION FOR PICKING HEROES IN LEAGUE OF LEGENDS

## SUMMARY

Nowadays internet has become popular therefore the gaming industry is in demand. Everyday many new games appear and most of them are being played by a considerable group of people. League of Legends is one of the most popular games in the world. In this thesis, it is aimed to create an artificial intelligence that will support players in the picking phase of the game and provide them the suggestion of champions which will increase the win rate most. League of Legends is a MOBA game and as similar to other MOBA games, it is played with 2 teams consisting of 5 players each. The map in the game is called "Summoner's Rift" and the main purpose is to reach opponent team's base and destroy their "Nexus". There are many variables in the game which affects the outcome of the game.
There are some systems in League of Legends which determine the level of a player. First one of them is the basic levelling system. When a player starts the game for the first time, they will be at level 1 and by playing the game they gain experience points which will provide them to reach higher levels. Other than this system, there are also a ranked game system in the game which is called the ELO system. In ELO system, a player will gain or lose points after their ranked games due to the outcome of the game and their own individual performances. The ranks in LoL are iron, bronze, silver, gold, platinum, diamond, grand master and challenger. Players will match with other players close to their ranks. When a player wants to play a game, they decide the lane they want to play in and start the match searching. When a match is found, firstly there is a ban phase where every player bans a champion. Then the picking phase starts. In this phase teams chose their heroes by the order of 1:2:2:2:2:1. There are nearly 150 heroes in LoL, therefore deciding which hero to chose is a difficult process. Every champion has its unique abilities and roles. A player should pay attention to many variables while picking their hero. For example, the lane they will play at, the champions the opponent team has chosen, the champions that their own team has chosen and their champions' health points, ability and attack damages, etc. will have a big role of the decision a player should make. In short, it is a hard decision connected to many variables. After the players have chosen their heroes, there are a phase where they can finalize their controls and adjust their spells and runes. After this phase ends, the game starts.

As mentioned before, there are many variables that affects the outcome of the game and some of them are the decisions made in the gameplay. In the Summoner's Rift there are 3 lanes and a jungle. In the top and middle lane there is 1 champion, in the bottom lane there are 2 champions and in the jungle there is 1 champion. Every champion in every lane has an important role in the game. There are bots in the game spawning in every lane by time. These bots are called minions. Minions have low health point and damage, however they can attack to opponent champions and block them. They can have a major affect on the fights and gameplay. By killing these minions, a champion can gain gold which they can spend it in market and buy items. There are many items in the market and every champion should pick the right items that will fit their character's role and aim.

There is a jungle in the game, as mentioned before, and in this jungle there are some creatures. It is the junglers duty to kill these creatures and by killing them they can provide their teammates extra buffs like damage or mana regeneration.
The main purpose in this thesis is to create an artificial intelligence that can make suggestions in the picking phase in League of Legends. When the previous research are examined, it is seen that there are lack of studies in this field. Generally the researches are about the outcome prediction and analysis of the game. However, when previous researches are evaluated, the best algorithm for this thesis is seen to be the MinMax algorithm. By using the data provided from Riot Games API, it is aimed to build a MinMax algorithm using Python to create the artificial intelligence for League of Legends.

# LEAGUE OF LEGENDS ŞAMPİYON SEÇİMİNDE YAPAY ZEKA UYGULAMALARI

## ÖZET

Günümüzde internetin hızlıca yaygınlaşmasından dolayı oyun endüstrisi oldukça rağbet görmektedir. Her gün piyasaya onlarca oyun çıkmakta ve bunların çoğu oldukça fazla oyuncu tarafından oynanmaktadır. League of Legends dünyadaki en popüler oyunlar arasında üst sırada yerini almış bir oyundur. Bu tezin amacı, League of Legends oyununda oyunculara şampiyon seçimlerinde destek olacak, onlara en yüksek kazanma ihtimalini sağlayan şampiyonları önerecek bir yapay zeka uygulaması oluşturmaktır. League of Legends bir moba tarzı oyundur ve tüm moba oyunlarında olduğu gibi 5'er kişiden oluşan 2 takım karşı karşıya oynar. "Sihirdar Vadisi" adlı bir haritası vardır ve asıl amaç karşı takımın merkezinde bulunan "Nexus" kulelerini yok edip oyunu kazanmaktır. Oyunda bir çok değişken bulunmaktadır. Bunların bazıları oyun öncesindeki kararlara bağlıyken bazıları oyun sırasındaki kararlara bağlı olarak değişir. Bu tezde oyun öncesindeki kararlar üzerinde çalışma yapılmıştır.

League of Legends'da oyunucunun aşamasını belirleyen birkaç sistem vardır. Bunlardan ilki seviye sistemidir. Bu aslında her oyunda bulunan bir sistemdir. Bir oyuncu oyuna ilk defa başladığında 1.seviye olarak sayılır ve oyun oynadıkça deneyim puanları kazanarak gittikçe daha yüksek seviyelere erişir. Bu temel seviye sisteminin dışında LoL'de bir de derece sistemi vardır. Buna ELO sıralama sistemi de denir. Bu sistemde oyuncu dereceli oyunu sonrasında, oyunun sonucuna, yani oyuncunun takımının oyunu kazanıp kaybetmesine göre, ve oyuncunun oyun içerisindeki rolüne uygun bireysel performansı ölçülerek lig puanı kazanıp kaybetmesi ile belirli bir lig seviyesine yerleştirilir. League of Legends'daki lig seviyeleri; demir, bronz, gümüş, altın, platin, elmas, ustalık aşaması ve üstü olarak sıralanabilir. Bu sıralamada demir en düşük derece ve ustalık aşaması ve üstü en yüksek derecedir. Oyuncular dereceli oyunları kendi derecelerine en yakın oyuncular ile oynarlar. League of Legends oyununa girmeden önce oyun ararken, bir oyuncu oynamak istediği konumu seçer ve ona göre oyun aramayı başlatır. Oyuncunun tercihine göre konumandırıldığı bir oyun bulunur. Daha sonra seçim aşaması diye adlandırdığımız kısım başlar. Bu aşamda öncelikle her oyuncu birer şampiyon yasaklarlar. Daha sonra seçim aşaması başlar. Seçim 1:2:2:2:2:1 şeklinde ilerler. Özetle, bu aşamada oyuncu hangi şampiyonu oynayacağını seçer. Ancak League of Legends'da yaklaşık 150 şampiyon vardır ve günden güne bu sayı artmaktadır. Her şampiyonun kendi özgü özellikleri, yetenekleri ve konumları vardır. Oyuncu şampiyon tercihi yaparken birçok şeye dikkat etmelidir. Bunlar, kullanıcının konumlandığı yer, yasaklanan şampiyonlar, kendi takım arkadaşlarının rolleri, yetenkleri, hasar, can, zırh ve büyü direnci değerleri, rakip takımın seçtiği şampiyonlar ve seçmesi olası olan şampiyonlar olarak örnek verilebilir. Kısacası bu tercih bir çok değişkene bağlıdır ve en yüksek kazanma ihtimalini sağlayacak tercihi yapmak zordur. Oyuncular şampiyon seçimlerini yaptıktan sonra son kontrollerini yapmak ve büyü ve rünlerini ayarlamak için belirli bir süreleri vardır. Bu süre de dolduktan sonra oyun başlar.

Bahsedildiği üzere, oyunun gidişatını ve sonucunu etkileyen bir çok değişken vardır ve bunlardan bazıları da oyun başladıktan sonra olanlardır. Tezde daha detaylı

bahsedildiği üzere, oyun haritasında 3 farklı patika ve bir de orman vardır. Üst ve orta patikada 1 şampiyon, alt patikada 2 şampiyon ve ormanda da 1 şampiyon oynar. Her patikadaki şampiyonların rolleri kendi konumlarına özgüdür ve oyun için eşit derecede önemlidir. Her patikada belirli bir zamanda bir ortaya çıkan botlar vardır, bunlara minyon denir. Bu botların hasar ve can değerleri düşüktür ve karşı takımın kulelerine doğru yol alırlar. Önlerine çıkan şeylere saldırırlar, bu yüzden minyonların da oyundaki rolleri önemlidir çünkü bir savaş esnasında karşı takıma ekstradan hasar vurup onların yolunu engelleyebilirler. Minyonları ve bazı orman canavarlarını kestikçe oyuncular para kazanırlar ve oyundaki markette bu parayı eşyalar almak için kullanabilirler. Bu eşyalara şampiyonun rolüne göre karar verilir ve eşya aldıkça şampiyon güçlenir. Doğru eşya seçmek de oyunu kazanmak için oldukça önem taşımaktadır.

Bahsedildiği üzere oyunda bir de orman vardır ve burada bazı yaratıklar bulunur. Bu yaratıkları kesmek ormancı rolündeki oyuncunun görevidir. Bazı yaratıklar oyun için çok önemidir çünkü kesen kişiye veya tüm takım arkadaşlarına ekstra hasar, can, vb. eklentiler sağlayabilir. Bu yaratıklardan bazıları; ejderler, Baron Nashor, Vadi'nin alameti olarak sıralanabilir.

Bahsedildiği üzere tezin amacı, LoL'de şampiyon seçiminde öneriler yapabilen bir yapay zeka uygulaması oluşturmaktır. LoL ve benzer oyunlar üzerindeki çalışmalar incelendiğinde, bu konuda yeterli çalışma olmadığı fark edilmiştir. Genel anlamda maç sonucu tahmin etmek ve analiz yapmak üzerine çalışmalar yapılmış olsa da, bu tezin amacına yönelik birkaç araştırma da bulunmuştur. Bu araştırmalar incelendiğinde tezin amacına yönelik en uygun algoritmanın MinMax algoritması olduğuna karar verilmiştir. Python'da bir MinMax algoritması çalışılarak, Riot Games veri tabanından sağlanan veriler üzerinden istenilen yapay zekanın oluşturulması amaçlanmıştır.

# 1. INTRODUCTION

Technology has improved significantly throughout the last decade. Thanks to these advancements, computer and internet usage has become more accessible among the majority of the society. Therefore, computer games have become more popular in recent years. Hall et al (2015) assert that as stated in Entertainment Software Association, video games are played by more than half of the people in the United States and the median age for gamers is thirty one years. Also, women players create the forty-eight percent of the gaming community.. League of Legends,which is part of the MOBA game genre, is one of the most popular games worldwide which has twelve million active daily players (Shores, et al, 2014). Due to complexity of the game and often updates from developers, interest in LoL is increasing day by day. Thus, there is a need for further and scientific research for the gameplay and more. Unfortunately, since this is a rather newer game, there is a lack of research about it.

## 1.1 Purpose of Thesis

The purpose of this thesis is to provide the most pleasant experience for LoL players by developing an artificial intelligence that guides them in the picking heroes phase. Even though there are some websites and applications that offer users basic statistical information about game data to also guide them in the picking phase; our aim is to integrate this process with the live game and provide more accurate suggestions than the previous ones.

In this scope, a literature review has been done on the MOBA games, general information about LoL, retrieving data from Riot Games API and methodologies that have been used in similar studies.

## 1.2 Literature Review

### 1.2.1 MOBA games

MOBA games which is the shortening of “Multiplayer Online Battle Arena” is a game genre. Generally there are different heroes and characters according to their abilities and features.. Even though MOBA games are a part of real-time strategy games,they differentiate from them since they lack constructing functions. In MOBA games, there are two teams of 5 players each, every player plays a unique role so coordination between team players and improvement of every role each is very crucial to decide the coherent tactic (Mora-Cantallops & Sicilia, 2018). The map differs in every MOBA game but all of them have 3 paths consisting of towers and turrets. The team to exceed all of the towers and turrets and destroy the opponent team’s base tower wins the game. There are many MOBA games such as League of Legends, Dota2, Heroes of the Storm, etc. The first MOBA game was Warcraft III: Reign of Chaos which was released in 2002. It has changed the focal point of games into unique and compelling heroes rather than mineral gathering, base constructing and ammunition that were the fundamentals in real-time strategy games (Ferrari, 2013).

### 1.2.2 League of legends and background

League of Legends is a team based MOBA game that was released by Riot Games in 2009. Since then, the game gained popularity rapidly and became the most popular MOBA game of all time. There are two teams that compete with each other. Each team consists of five players. LoL takes place in a map called “Summoner’s Rift”, which can be seen in the Figure 1.1. Teams are aimed to destroy their opponent’s base.

**Figure 1.1 :** Summoner's rift map.

League of Legends is an active game, each seven to twenty-one days, the game officials at Riot Games release a patch (Adams & Walker, 2017). Due to these updates, the course of the game constantly changes. Regardless of a game, the most suitable and popular team build for winning is called Meta. Thus, e-sport players find Meta very crucial for their gameplay (Agha, 2015). LoL players decide their hero picks, items, runes and spells by these changes in Meta, therefore it is crucial for a player to keep up with the Meta in order to succeed in the game. For instance, latest meta game champions can be seen from Figure 1.2.


**Figure 1.2 :** Meta game champions.

The game takes place in the multiverse and the primary universe is called “Runeterra Prime”. Universes involve champions and each champion has its own unique story that connects them with each other.differently.

### 1.2.3 Experience

Experience (also known as XP) is a game metric which provides players to level up after achieving certain amounts of experience. Players can gain XPs by playing games and completing daily and weekly missions.

### 1.2.4 Level system

For a new LoL player to play a ranked match, he/she needs to reach to the 30th level. To level up, players must gain a certain amount of XPs.

### 1.2.5 Blue essence and riot points

Blue Essence is a currency in the game that is used to unlock some League of Legends content like purchase of champions, hextech crafting materials, additional rune pages, chromas, exclusive contents in the Essence Emporium. Also, it can be used for changing a player's summoner name. Blue essence can be acquired from leveling up and first win of the day. Riot points (also known as RP) is another currency which unlocks very similar content to blue essence. Some of the unlocks are purchasing champions, hextech crafting materials, additional rune pages, chromas, bundles, emotes, ward skins, summoner icons and champion skins. Unlike blue essence, RP cannot be earned in the game via gaining experience. It has to be purchased with real-life currencies. For instance, champion skins of League of Legends can be bought by getting RP from the market for a price range between two and forty dollars (Agha, 2015). Furthermore, Agha (2015) stated that according to Tassi, in 2014 Riot Games has gained nearly one billion dollars via these small purchases.

### 1.2.6 Champions

Champions are the characters that are controlled by players in the LoL. Each champion has unique features and abilities. Champions’ health points, manas, attack

damages and speeds, ability powers, armors, magic resistances, movement speeds and ranges differ from another according to champions' classes. Since LoL is a metagame, the amount of champions and their attributes update at every patch of the game. When the recent status is examined it can be seen that there are 153 champions where some of them can be seen in Figure 1.3.


**Figure 1.3 :** List of some of the champions in LoL.

#### 1.2.6.1 Champion classes

The champions have classes like controller, fighter, mage, marksman, slayer, tank and specialist. Classes depict a certain pattern that is affected by the player's actions and reactions to the surroundings. These classes are also known as roles in the game where they have splitted into categories in a broader way as it can be seen from Figure 1.4. In the beginning of the game, players make first and second role preferences and the game assigns that player to one of that role.

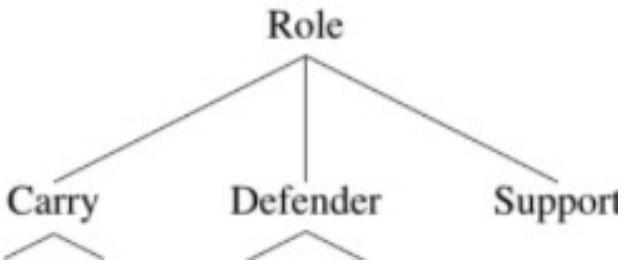



**Figure 1.4 :** Champion roles (Shores et al, 2014).

### 1.2.6.2 Controller

Controllers are defending by casting and managing the battlefield by protecting and providing opportunities for their teammates. This class is also known as "Supports" in the game. Controllers have two subclasses like catchers and enchanters. The distribution of controller's functions can be seen from Figure 1.5 and the meaning of the symbols in Figure 1.5 to 1.10 can be seen in Table 1.1.

**Figure 1.5 :** Distribution of a controller's functions.

**Table 1.1 :** Meaning of the symbols in Figure 1.5 to Figure 1.10.

| Symbol | Meaning |
|---|---|
| Foot | Mobility |
| Swords | Ability to deal with damage |
| Shield | Toughness |
| @ | Control |
| Cup | Utility |

#### 1.2.6.3 Catcher

Catchers speciality is in locking down their rivals. They can create intense zones to lock the whole battlefield as well. Bard, Blitzcrank, Ivern, Jhin and Lux are examples of champions that are part of catcher class.

#### 1.2.6.4 Enchanter

Enchanters defend their allies from dangers so that they can improve their allies' effectiveness. As a weakness, they do not damage much as others and they are slow in movement due to their low speed. Janna, Karma, Lulu and Nami are the examples of champions that are part of the enchanters class.

#### 1.2.6.5 Fighter

Fighters which are also known as bruisers that are mastered at surviving and dealing damage. Fighters have two subclasses which are divers and juggernauts. The distribution of fighter's functions can be seen from Figure 1.6.

**Figure 1.6 :** Distribution of a fighter's functions.

#### 1.2.6.6 Diver

Even though divers have lower durability than juggernauts, due to their higher mobility they are very good at singling out primary enemies to blitz toward, making those enemies to encounter with them instantly. Camille, Diana, Elise, Hecarim and Irella are given examples for the divers class.

#### 1.2.6.7 Juggernaut

This is the only subclass that is superior at both taking and dealing serious amounts of damage. Despite this power, they have a very limited mobility and low range. Some champions like Aatrox, Darius, Dr. Mundo and Garen fall into this class.

#### 1.2.6.8 Mage

Mages are offensive casters that burn down their rivals through their potent spells. They are very good at dealing with marksmen but they also can be shut down by assassins very easily. Mages have three subclasses like artillery, battlemage and burst. The distribution of mage's functions can be seen from Figure 1.7.

**Figure 1.7 :** Distribution of a mage's functions.

#### 1.2.6.9 Artillery

Artilleries excel at range. They use this advantage to destroy their rivals from far distant places. Unfortunately, this class has very limited mobility so they are badly damaged when enemies are finally closing in on them. Lux, Jayce, Varus and Zoe can be given examples for this class.

#### 1.2.6.10 Battlemage

This subclass is also called Warlocks. Compared to artillery, battlemages have shorter range and higher mobility. They stay at the fight longer but they cause less damage than others. Anivia, Karthus and Morgana are examples for this class.

#### 1.2.6.11 Burst

Burst Mages lock down their defenseless targets for singling out them. After locking down their targets, burst mages cause calamitous barrage of damage from distance.

On the other hand, burst mages suffer from stronger targets who can dodge their first damage. Ahri, Annie and Lux are a couple examples of heroes that are part of this subclass.

#### 1.2.6.12 Marksman

Marksmen are also known as “Carries” in the game. Although being very fragile, which makes them very defenseless to burst damage, marksmen have great range so they can deal with sustained damage at range. They can be very good at taking down their opponents when they are in a safe position. Examples for champions which fall into this class can be given as Aphelios, Jhin, Jinx and Miss Fortune. The distribution of marksman’s functions can be seen from Figure 1.8.

**Figure 1.8 :** Distribution of a marksman's functions.

#### 1.2.6.13 Slayer

Slayers are also known as “Assassins”. Champions who are part of this class are very nimble, damage prioritized melees that look to destroy their targets very fast. They are excellent at taking down marksmen and mages. On the other hand, they struggle while fighting with tanks and fighters. Slayers have two subclasses called Assassins and Skirmishers. The distribution of slayer’s functions can be seen from Figure 1.9.

**Figure 1.9 :** Distribution of a slayer's functions.

#### 1.2.6.14 Assassin

Assassins' speciality is in creeping in enemy lines with their peerless mobility. They put themselves in dangerous positions since they are melee fighters and use their defense methods to prevent getting damage. Akali, Pyke and Zed are a couple examples of the champions that are part of the subclass.

**Skirmisher**

This class is also called Duelists Skirmishers' goal is to destroy any nearby enemy approaches. Since Skirmishers cannot close in on high-prioritized targets due to their lack of high-end burst damage, they have strong defending tools to survive in the battle. Champions like Fiora, Jax and Yasuo are part of the Skirmisher subclass.

#### 1.2.6.15 Tank

Tanks main purpose is using their strong melee skill at disrupting their rivals and diverting focus to themselves. Thanks to this method, tanks' allies can be free from the dangers. Tanks have two subclasses like Vanguards and Wardens. The distribution of tank's functions can be seen from Figure 1.10.

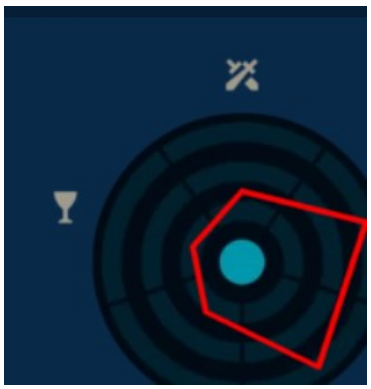


**Figure 1.10 :** Distribution of a tank's functions.

#### 1.2.6.16 Vanguard

Vanguards' specialty is at getting the action started while leading the charge in the team. Allstar, Leona and Sion can be given examples for this class.

#### 1.2.6.17 Warden

Unlike Vanguards, which are offending tanks, Wardens are defending tanks that stand steadfast, seeking to hold the line by persistently locking down any on-comers

who try to pass them. Braum, Shen and Tarc are the champions that are part of this subclass.

#### 1.2.6.18 Specialist

This is the class where outsiders of the fellow classes are grouped. Due to this situation, specialists are a very diverse group.

#### 1.2.6.19 Champion abilities

Champion ability is a  unique action that is performed by a champion and it is only available for that selected champion. A champion's kits is made up by these unique actions. Champion kit has a three tiered structure whereas the first one is innate. Second one is called basic abilities which can be controlled with "Q" , "W" and "E" keys at the keyboard. These abilities can be improved with ranking up. The third and final one is called ultimate and it is controlled with the "R" key in the keyboard. Ultimates are a champion's most powerful and unique ability which has the capability of changing the whole game. While using abilities, champions spend manas and also there is a time limit to use that ability again.

### 1.2.7 Skins

Skins are items that provide alternative looks to champions They can be bought via usage of RPs (Riot Points). Skins have tiers like ultimate, mythic, legendary, epic, standard and budget. These tiers are decided by their RP costs. The champion Lux can be seen in Figure 1.11 without any skin and in Figure 1.12 with a skin called "Elementalist Lux".

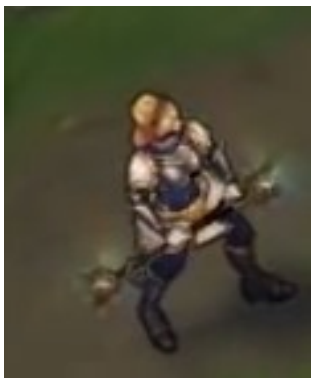


**Figure 1.11 :** The champion "Lux" without any skin.

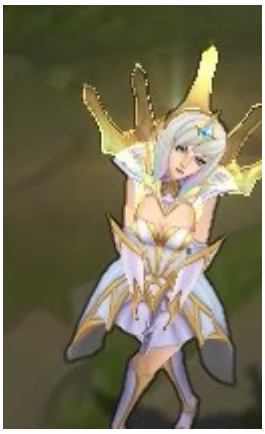


**Figure 1.12 :** The champion "Lux" with a skin.

### 1.2.8 Matchmaking

Basically, matchmaking is the automated process where players are matched with other players. The goal of the matchmaking process is to provide fairness and competitiveness in the match. First, the game decides the player's strength. Then, it determines eligible opponents for that player. Finally, the system completes the match in the fairest way possible. According to Decelle et al, the study on fifty-two games, regardless of the ranks, average players wait less than sixty seconds before the matchmaking phase and it is addressed that nearly quarter of them are unbalanced (2015).

#### 1.2.8.1 ELO

For determining similar talent leveled chess opponents, the Elo rating system has been discovered by Arpad Elo (Veron et al, 2014). Basic idea behind the ELO system is it uses Mathematics for predicting the game result between two players. League of Legends uses a modified version of this ELO system. The rating system in LoL goes like: iron, bronze, silver, gold, platinum, diamond, grand master and challenger where bronze being lowest and diamond highest rank. Nearly, seventy percent to ninety percent of the players are positioned mainly in silver ELO, bronze and gold ELOs consecutively. Players in challenger or master ELOs form a group of approximately 0.1 percent of total players. Moreover, only 200 players are part of challenger ELO (Mora-Cantallops & Sicilia, 2018).

### 1.2.8.2 Lanes

Lanes are the main paths where minions follow the Nexus. Paths have planted structures like Turrets which pushes away opponent minion advances. In lanes, these consecutive structures have to be destroyed in order to destroy the Nexus. In Summoner's Rift, there are three lanes called top, mid and bottom, which can be seen from Figure 1.13. Couple roles can be played in each lane.

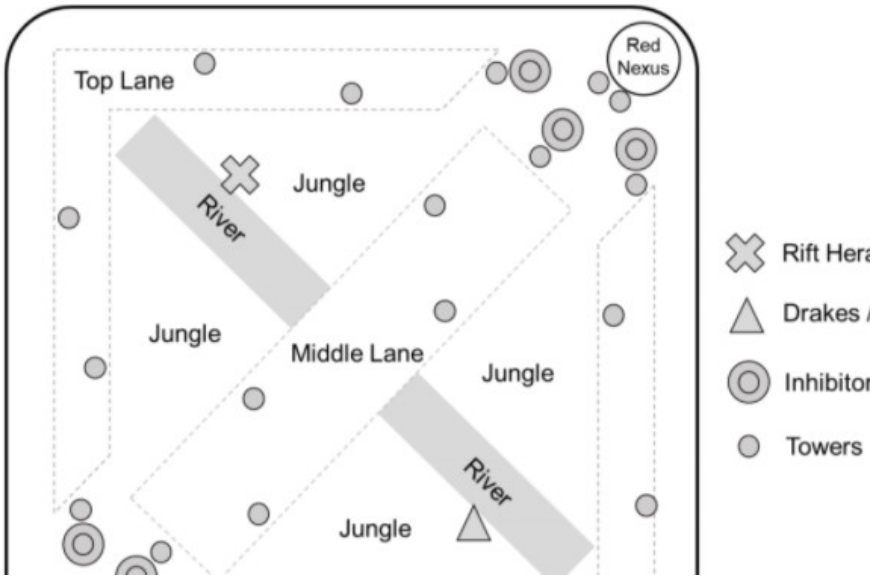



**Figure 1.13 :** Lanes in summoner's rift (Novak et al, 2019).

**Solo/top**

This lane's role priorities dragons, therefore the player in here has to isolate her/himself from the rest of the team.

**Mid**

Since it has a short distance, this lane provides the highest gold and experience earnings. Champions like mages which have bad defense are a good choice for this lane.

**Bottom**

Support and Carry roles are played in this lane. Like top lane, dragons are a priority here.

**Jungle**

Jungle is the area where it left after defining top,bottom and mid lanes. Rather than the others, it is a safer environment to earn golds This is the area where players gank(ambush) enemies.

### 1.2.9 Draft phase

Draft phase is the phase where the matchmaking is done and players get ready to start the game by banning and picking their heroes and adjusting their runes and spells. After the matchmaking is done, an alert appears on the player's screen that says "Match found" with buttons "Accept" and "Decline". If a player clicks on the "Accept" button, it leads him/her to the pick/ban phase.

#### 1.2.9.1 Teams

As mentioned before, there are 2 opponent teams in LoL consisting of 5 players each. One team is called "The Blue Team" and the other is called "The Red Team".

#### 1.2.9.2 Ban phase

Before the ban phase, there is a 15 second phase that enables players to show their intended heroes to their team in order to prevent other players from banning or picking their intended heroes. In the ban phase every team player decides and picks a hero to ban. The banned heroes can not be picked in that game. Therefore the ban phase is a beneficial phase for a player to make tactics about his/her game. For instance if a player plans to pick a hero, he/she can ban the hero's counter hero and increases the chance of his/her team's victory.

The ban phase is a blind ban phase which is 27 seconds. Only those in the same team can see each other's bans. And two players in the same team can not ban the same hero. Also if a player fails to ban a hero, it will result in the player automatically banning none. After 0 to 10 heroes are banned, bans are revealed to both teams for 5 seconds and the pick phase starts.

#### 1.2.9.3 Pick phase

In the pick phase, the order is 1:2:2:2:2:1. The duration for the pick phase is 27 seconds in 6 turns. The players in the teams are ordered randomly. The first player from the blue team picks the first hero, then 2 players from the red team picks their

heroes simultaneously and it continues like this. Among 153 heroes, a player should choose a hero. A banned hero and a chosen hero by another player can not be chosen.

The hero pick phase is an important step of the game because it affects the whole gameplay. There are many variables in hero picking. First of them is competence. A player should know the hero and its spells before he/she picks it. Another important variable is that a player should notice the opponent team's picks and decide according to them. Besides the heroes that are chosen, a team's compatibility is important. As mentioned before, there are different roles in LoL and a team should be a combination of different roles. For instance if a team has no tank heroes, it is likely that they will die early during team fights. Therefore during the pick phase, decisions must be made carefully in order to maximize the chance of victory.

#### 1.2.9.4 Finalization of draft phase

After every player picks their heroes, there is a phase of finalization which is 30 seconds. In this phase, players can exchange heroes, adjust their spells and runes and make tactics through the game chat.

**Hero exchange**

Players can exchange their heroes after the pick phase if both players own both champions. Trades can be made multiple times. The players who will pick early in the picking phase can hold heroes for their teammates which will choose later. This method is used to secure high tier heroes early in the pick phase and prevent the other team to take them. For heroes like Yasuo, Zed, etc., this method is commonly applied.

**Runes**

Runes are additional features that can customize a champion and improve its power. There are 2 parts of runes; keystone rune and 5 secondary runes. Runes can be adjusted before the match and during the draft phase. There are also standard runes in the game which are pre-prepared by Riot games. There are 5 main rune paths which are; Precision, Domination, Sorcery, Resolve and Inspiration. These runes can be adjusted according to a champions function and also metagame.

**Spells**

Spells are extra abilities that can be chosen by the player in the draft phase. 2 spells can be chosen for a champion in order to grant more damage, instant health, mobility, etc. These spells, which can be seen from Figure 1.14, are Heal, Ghost, Barrier, Exhaust, Flash, Teleport, Smite, Cleanse and Ignite.

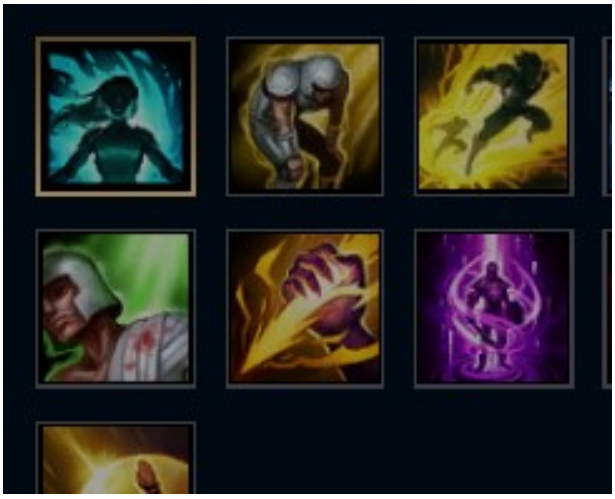


**Figure 1.14 :** Spells.

### 1.2.10 Gameplay

Gameplay is the most important factor that affects the end game and win rate. According to the champions chosen, the gameplay takes shape. Duration of a standard LoL match is thirty one minutes. However, it varies between twenty and over sixty minutes. (Shores et al, 2014). We can split the gameplay in 3 main parts; early game, middle game and end game.

In the early game, the main purpose is to farm. By farming, champions can gain gold, buy items and get more powerful than their opponents. In the early game the aim is to dominate the enemy team and prevent them from farming and getting stronger.

In the middle game, the main purpose is to destroy enemy turrets. In the middle game, one-to-one and team fights start. It is crucial to win these fights. Also in the middle game, dragons, rift herald and baron nashor can be killed which will provide extra advantage to a team.

In the end game, the main purpose is to destroy enemy inhibitors and nexusus. Therefore pushing lanes and dominating the opponent team is very important.

**Table 1.2 :** Timescales and temporal factors of gameplay (Reitmen, 2018).

| Timescales | Temporal Factors of Gameplay |
|---|---|
| Less than a second to few seconds | Animation durations |
| A few seconds to a minute | Ability cooldowns, time needed for objectives |
| A minute to a few minutes | Spell cooldowns |
| Dozens of minutes | Global game time |
| Variable | Travel speeds, rate of gold and experience gain |

#### 1.2.10.1 Minions

Minions are units sent by Nexus. They are generated as a wave every 30 seconds. They are controlled by the artificial intelligence of the game and their aim is to push the lane and attack any unit they see. Their damage is not high but they have big importance in the game. There are 4 types of minions; melee, caster, siege and super minions, where they can be seen in Figure 1.15, Figure 1.16, Figure 1.17 and Figure 1.18 respectively. By killing these minions a player can gain 21, 14, 60-90 -which depends on the minute of the game- and 60-90 golds respectively.

**Figure 1.15 :** A melee minion.

**Figure 1.16 :** A caster minion.

**Figure 1.17 :** A siege minion.

**Figure 1.18 :** A super minion.

#### 1.2.10.2 Jungle creatures

Apart from the lanes, there are other creatures in the jungle in Summoner's Rift which can be killed for XP and gold. The player who is playing the jungle role will kill these creatures and contribute to his/hers team.

**Dragons**

One of the important jungle creatures is dragons. Dragons spawn in the map periodically, killing them and gaining the extra power that is depending on the dragon's type is a very important factor that affects the whole gameplay. As Wong's study stated that in a LoL match where gold amounts are equal, there will be a sixty-seven percent chance of a team to be victorious if they have one dragon more. Besides, for a team that has one less dragon the chance for being victorious is forty-six percent (Wong, 2019).

There are 5 types of dragons in the game which are Cloud Drake, Mountain Drake, Infernal Drake, Ocean Drake and Elder Drake.

In the table below, the gains of dragon slays can be seen.

**Table 1.3 :** Drakes and their buffs.

| Name of the Drake | Buff |
|---|---|
| Cloud Drake | +3/4,5/6% bonus movement speed, tripled when out of combat. |
| Mountain Drake | +16/23/30% bonus true damage to turrets and epic monsters. |
| Infernal Drake | +10/17/24% attack damage and ability power. |
| Ocean Drake | Restores 6/9/12% of missing healt and mana every 5 seconds unless any damage is taken in the last 8 seconds. |
| Elder Drake | Increases other dragon's buff by 50% while also granting a burning effect on basic attacks. |

**Rift herald**

Rift Herald is another jungle creature which can be seen from Figure 1.19. It spawns in the jungle of Summoner's Rift and by killing this creature a player can get an item called "Eye of the Herald". Eye of the Herald can be used in order to call the creature to any lane for it to push the lane and attack turrets. By capturing Eye of the Herald, a team can gain advantage.

**Figure 1.19 :** Rift herald.

**Baron nashor**

Baron Nashor is the most powerful creature in the Summoner's Rift which can be seen from Figure 1.20. It has more health rather than other creatures however by killing the Baron Nashor a team gains extra ad, extra ap and a decrease in the respawn duration. Therefore in many LoL matches, both teams fight for the Baron.

**Figure 1.20 :** Baron nashor.

### 1.2.10.3 Market

In League of Legends, as mentioned before there is a system which is supported by gold. There is a market where players can buy items that will empower them when they are in their base. One champion can get 6 items at maximum. There are many items in the market, which some of them can be seen from Figure 1.21, therefore every hero's recommended items are different. A player can examine the opponent team's gameplay and buy items that will counter them.

As mentioned before, League of Legends is an active game with constant updates, therefore sometimes some items are more popular for certain champions. Players should follow the metagame to succeed in the game.


**Figure 1.21 :** Market in LoL.

### 1.2.11 After game phase

After a team destroys the opponent team's Nexus, after game phase starts. In the after game phase players can admire their team mate by clicking buttons "GG", "Stayed cool" and "Good leader". Moreover, players can monitor the statistics of the game which they can analyze afterwards to improve themselves.

### 1.2.12 Previous researchs

Even though, there are similar researchs about AI applications in MOBA games these are mostly related with either outcome prediction or analysis of the game statistics. There is a need for further and detailed AI application studies at MOBA games especially in League of Legends since as it mentioned before it is a rather popular and recent game. Some of the researchs can be seen in Table 1.4.

**Table 1.4 :** Related Researchs About LoL and Other MOBA Games.

| Focused Topic | Recommend Method | Reference |
|---|---|---|
| Extracting more detailed and continuous data from Lol matches | Big Data | (Kim et al, 2020) |
| Predicting winnings in LoL | Recursive Feature Elimination(RFE), Gini importance | (Ani et al, 2019) |
| Predicting the game Outcome in LoL | Logistic Regression, Gradient Boosting, Gradient Boosting Trees | (Lin, 2016) |
| Forecasting the game Outcome in LoL | Recurrent Neural Networks(RNN), LSTM, GRU | (Silva et al 2018) |
| Developing a AI bot for solo matches in LoL | LSTM, | (Lokohare et al, 2020) |
| Predicting outcome in DOTA2 | Naïve Bayes Method, Logistic Regression, Random Forest, SVM | (Wang et al, 2018) |
| Finding best character choice in LoL | AHP, TOPSIS, PROMETHEE | (Ercan et al, 2016) |
| Analyzing drafting phase's effects on game outcome in DOTA2 | Bayes Nets, Long Short Term Memory(LSTM), RNN | (Summerville et al, 2016) |
| Recommendation for hero picking at DOTA2 | Feature Vector, Logistic Regression, kNN | (Conley and Perry, 2013) |
| Choosing best move in card battle game | MinMax Algorithm | (Permana, 2019) |

| | | |
|---|---|---|
| Assisting players at drafting phase in LoL | MinxMax Algorithm | (Oliviera et al, 2017) |

When these previous researchs are evaluated, it is convenient to mention Kim et al's study first. Kim et al (2020) conducted a study to build a service that analyses LoL matches and presents information which involves the game's detailed evaluation and the contribution of a player to the team. They were inspired from the site called "OP.GG" which has the similar function, however they realised that data collection of OP.GG is discrete and more detailed continuous data is needed for a better analysis. Therefore, they designed a data collection system that provides them a big amount of data, and defined some variables such as; split scores, rotation scores, etc. By using these scores and further stats, they created a system. The table 1.5 is the comparison of OP.GG and the service Kim et al provided.

**Table 1.5 :** Features of OP.GG and authors' service (Kim et al, 2020).

| Feature | OP.GG | Kim et al's Service |
|---|---|---|
| Providing basic per-game information | O | O |
| Providing Riot API based statistics | O | O |
| Providing additional data analysis | X | O |
| Providing statistics per champion | O | X |
| Providing in game event information | O | O |

In conclusion, due to lack of data collection which is restricted by League of Legends API, the system is not efficient as they wanted. However, as a result of the study, it is possible for a player to analyse and improve her/his gameplay by using the authors' tool, where its interface can be seen at Figure 1.22.

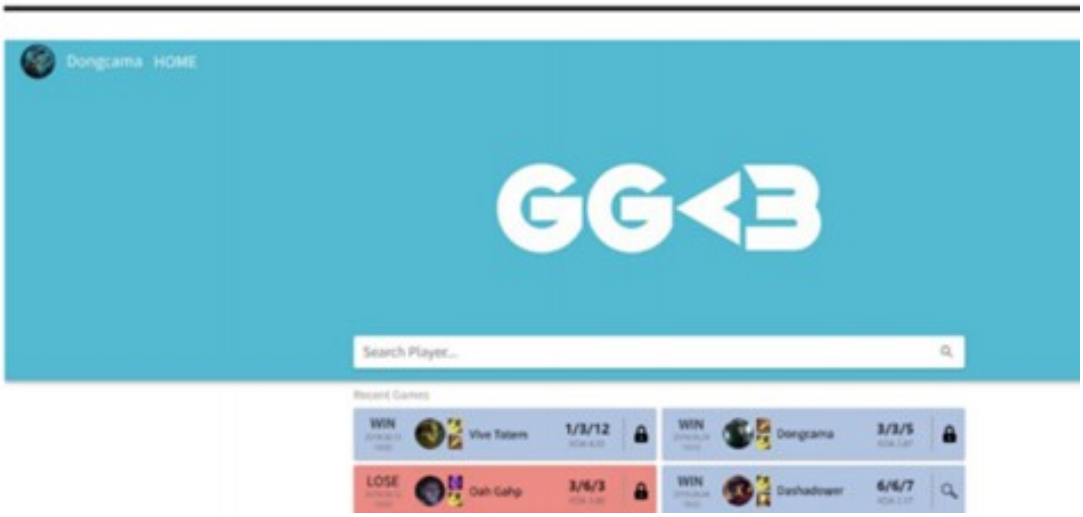



**Figure 1.22 :** Main client interface (Kim et al, 2020).

Furthermore, there are some studies which focus on predicting a LoL matches outcome. One of these studies was conducted by Ani et al (2019) to predict winnings at League of Legends while using various methods. It contains 1500 matched datasets where sixty percent of it is used for training and forty percent of it is used for testing purposes. They first tried to select appropriate features with the usage of Recursive Feature Elimination(RFE). Then they used the Gini importance measure to rank the top three aspects. This part is divided into pre and within game parts After the feature selection part, they focused on the ensemble part where they used random forest, adaboosting, gradient boosting, and extreme gradient boosting methods respectively. The accuracy results of pre-match, within match and combined are 95.52%, 98.18%, 99.75% with random forest, 57.22%, 96.31%, 96.23% with adaboost, 65.67%, 96.82%, 97.01% with gradient boosting and 65,12%, 96.83%, 97.21% with extreme gradient boosting respectively. According to this study, it can be conducted that the random forest approach will be the best of all other methods that is used in this research.

**M**oreover, as Lin (2016) states in his study, his pre-match information was picked heroes, summoner spells, mastery points of each hero and the winner team of the match. Also, he took total damage dealt per player, obtained the first kill of the game or not, kills to deaths ratio, gold spread within a team through mean and standard deviations, difference in gold between two players in a single lane etc. He used Gradient Boosting method and Gradient Boosted Trees with Logistic Regression methods for outcome predictions In this study they evaluated their train and test

errors. While all methods' train errors were either zero or almost zero,pre-match gradient boosting and gradient boosted tree with logistic regression methods' test error are 0.5 approximately. On the other hand, it can be said that both in game gradient boosting and gradie%nt boosted tree with logistic regression methods were very succesful due to their almost zero error rates.

Another example for outcome prediction in League of Legends can be examplified as Silve et al's studies. Silva et al (2018) studied to forecast the conclusion of a League of Legends match by using different types of Recurrent Neural Networks and data from Kaggle, which involves 7621 LoL matches between 2015-2018 and shows every information in an exact moment. The dataset involves both general information, which are "Teams involved", "Result", "Selected Champions", etc., and the states on every minute in the match, which are "Minute a unit was demolished", "Minute a hero was killed", etc. Three types of RNNs were involved in the first phase of the study, to decide the most accurate method: RNN, LSTM and GRU. For the first experiment 1/3 of the games were selected for the test, 2/3 for training. After the experiment when they examined the accuracy and standard deviation results on each method, which can be seen from Table 1.6, since Simple RNN has the highest accuracy and lowest standard deviation it is the most suitable method for this research.

**Table 1.6 :** Results comparing different RNNs for the time interval 10-15 (Silva et al, 2018).

| Network | SimpleRNN | GRU | LSTM |
|---|---|---|---|
| Accuracy | 76.29% | 72.92% | 71.63% |
| Standard Deviation | 1.16% | 1.39% | 1.63% |

For the second phase of the experiment, different time intervals from the match are tested by using Simple RNN to gain the highest accuracy in predicting the game outcome. Five different time intervals are used: 0-5, 5-10, 10-15, 15-20, 20-25. It is seen that, in the late game the gap between the winning and the losing team is bigger therefore by using the late game data it is possible to predict the outcome with the highest accuracy.

Another example of LSTM method in LoL is Lokahare et al's study. Lokohare et al (2020), studied an artificial intelligence project for League of Legends that aims to train a bot in a custom 1v1 game and succeed it to imitate a player. In their study, they utilized from LTSM and Proximal Policy Optimization methods. Due to their goal, they created a custom game in MidLane only. Their first player was Ashe and as their enemy they used Veigar. Ashe was only allowed to use one skill, "Volley". Nearly 70,000 detailed frames from 50 games were defined as dataset which involves both keyboard and mouse movements and presses. In addition to LTSM model, in PPO model many iterations were made in order to decrease poor choices, it was trained for 3 days. In conclusion, both models works considerably well as an artificial intelligence, however if models were trained more, the result would be more accurate.

DOTA2 is another game which is a part of MOBA game genre. Since League of Legends and DOTA2 have many similarities, especially in drafting phase and gameplay, researchs about DOTA2 are significant resources which will guide future works in AI studies in League of Legends. Wang et al has made a study to predict outcome in DOTA2 with the help of machine learning methods. Since high skill ability is very important to have high accuracy, they chose the highest ranked Chinese player's matches for their dataset. They also benefited from this players' competitors' match datas too. Eventually they managed to get 38629 high level match data. First they grouped heroes' features into 3 main categories which can be seen from Figure 1.23 (Wang et al, 2018).

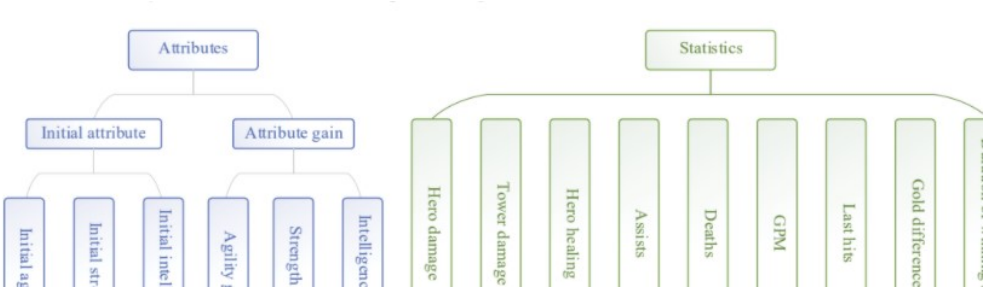



**Figure 1.23 :** The champions' features (Wang et al).

Naive Bayes Model is used to vector of length 115 to represent a heroes' draft with only value of 0, 1, and -1 in the vector at every attribute where heroe is chosen by Radiant vector gets 1 value which is shown in the equation (1.1) and equation (1.2),

whereas it gets -1 when it is picked by Dire and the value becomes 0 when it is not picked any of them which can be seen from equation (1.3).

$$x_i = \begin{cases} \mathbf{1}, & \textbf{if hero is picked by Radiant} \\ \mathbf{0}, & \textbf{if hero is not picked} \end{cases} \quad (1.1)$$

$$x_{i+144} = \begin{cases} \mathbf{1}, & \textbf{if hero is picked by Radiant} \\ \mathbf{0}, & \textbf{if hero is not picked} \end{cases} \quad (1.2)$$

$$x_i = \begin{cases} \mathbf{1}, & \textbf{if hero is picked by Radiant} \\ \mathbf{0}, & \textbf{if hero is not picked} \\ \mathbf{-1}, & \textbf{if the hero is picked by Dire} \end{cases} \quad (1.3)$$

After defining vectors, they created eigen vectors for the method that they used which can be seen in Figure 1.25.

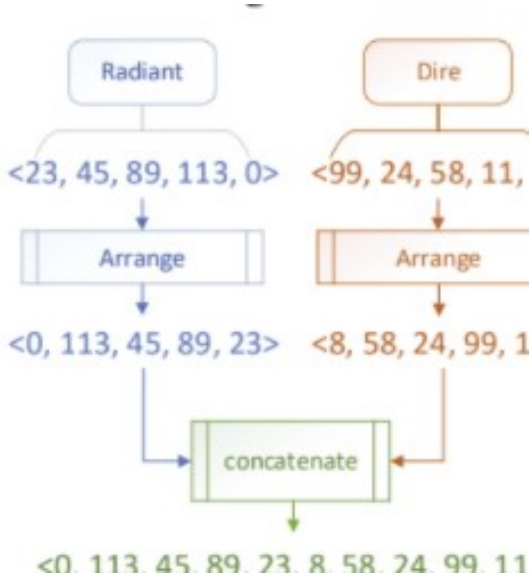



**Figure 1.24 :** Method that shows heroes' draft (Wang et al, 2018).

In the experiment part, they have used logistic regression, random forests and SVM. The results of this study can be seen in the Table 1.7.

**Table 1.7 :** Logistic Regression, Random Forests and SVM Results on data set 2 (Wang et al).

| Claasifier | Classification | Precision | Recall | n-score | support |
|---|---|---|---|---|---|
| Logistic Regression | -1 | 0.7297 | 0.77826 | 0.75321 | 3202 |
| | 1 | 0.7016 | 0.644 | 0.671 | 2593 |
| | total | 0.717 | 0.718 | 0.716 | 5795 |
| Random Forests | -1 | 0.716 | 0.718 | 0.740 | 3202 |
| | 1 | 0.68457 | 0.62437 | 0.653 | 2593 |
| | total | 0.701 | 0.703 | 0.701 | 5795 |
| SVM | -1 | 0.728 | 0.797 | 0.7611 | 3202 |
| | 1 | 0.716 | 0.63286 | 0.67199 | 2593 |
| | total | 0.722 | 0.7235 | 0.72123 | 5795 |

Besides the researchs about outcome predictions, some researchers have made studies which focus on the impact of the picking phase on the outcome and how to suggest champions in the draft phase to maximize the victory chance. For instance Ercan et al (2016), have studied to find the best character choice with the usage of multicriteria decision making methods like Analytic hierarchy process(AHP), TOPSIS and PROMETHEE. They had two scenarios while one is at Summoner's Rift and the other is in Howling Abyss modes. Summoner Rift's scenario can be seen in the Table 1.8.

**Table 1.8 :** Summoner's rift team composition scenario (Ercan et al, 2016).


For Summoner's Rift, AHP results can be seen in the table 1.9.

**Table 1.9 :** Finding Alternative Priorities of Champions (Ercan et al, 2016).

| Seçenekler | Kriterler | Kriterlerin Öncelik Vektörleri | Seçeneklerin Öncelik Vektörleri | Öncelik Vektörlerinin Çarpımı | Seçenek Önceliği | Ön |
|---|---|---|---|---|---|---|
| Katarina | Alanındaki Rakibe Karşı Durum | 0,5579 | 0,0348 | 0,0194 | 0,0893 | |
| | Rakip Takımdaki Karakterlere Karşı Durum | 0,1219 | 0,0654 | 0,0080 | | |
| | Hareketlilik | 0,0569 | 0,5028 | 0,0286 | | |
| | Takım Arkadaşlarının Büyüleriyle Uyumu | 0,2633 | 0,1265 | 0,0333 | | |
| Yasuo | Alanındaki Rakibe Karşı Durum | 0,5579 | 0,5028 | 0,2805 | 0,4481 | |
| | Rakip Takımdaki Karakterlere Karşı Durum | 0,1219 | 0,2571 | 0,0313 | | |
| | Hareketlilik | 0,0569 | 0,0678 | 0,0039 | | |
| | Takım Arkadaşlarının Büyüleriyle Uyumu | 0,2633 | 0,5028 | 0,1324 | | |
| LeBlanc | Alanındaki Rakibe Karşı Durum | 0,5579 | 0,2602 | 0,1452 | 0,2325 | |
| | Rakip Takımdaki Karakterlere Karşı Durum | 0,1219 | 0,5028 | 0,0613 | | |
| | Hareketlilik | 0,0569 | 0,2602 | 0,0148 | | |
| | Takım Arkadaşlarının Büyüleriyle Uyumu | 0,2633 | 0,0428 | 0,0113 | | |
| Zed | Alanındaki Rakibe Karşı Durum | 0,5579 | 0,1344 | 0,0750 | 0,1153 | |
| | Rakip Takımdaki Karakterlere Karşı Durum | 0,1219 | 0,1265 | 0,0154 | | |
| | Hareketlilik | 0,0569 | 0,1344 | 0,0076 | | |
| | Takım Arkadaşlarının Büyüleriyle Uyumu | 0,2633 | 0,0654 | 0,0172 | | |

For Summoner's Rift, TOPSIS results can be seen in the tables 1.10 and 1.11.

**Table 1.10 :** The decision matrix (Ercan et al, 2016).

| | ARKD | RTKKD | Hareketlilik | TA |
|---|---|---|---|---|
| Katerina | 0,0331 | 0,0136 | 0,0262 | 0,0 |
| Yasuo | 0,4780 | 0,0536 | 0,0035 | 0,1 |
| LeBlanc | 0,2474 | 0,1049 | 0,0135 | 0,0 |
| Zed | 0,1278 | 0,0264 | 0,0070 | 0,0 |

**Table 1.11 :** Evaluations of the alternatives (Ercan et al, 2016).

ığı ($S_i$) hesaplanarak bu uzaklıklar kullanılarak her b
ı ($C_i^*$) bulunmuştur. Hesaplanan değerler Tablo 8.'de v

***Tablo 8. Alternatiflerin Değerlendirme Sonuç***

| Sıra | Alternatifler | $Si^*$ | $Si^-$ | $Ci^*$ |
|---|---|---|---|---|
| 1 | Yasuo | 0,056 | 0,4627 | 0,892 |
| 2 | LeBlanc | 0,26 | 0,2346 | 0,4743 |

For Summoner's Rift, PROMOTHEE results can be seen in the table 1.12.

**Table 1.12 :** PROMETHEE result (Ercan et al, 2016).

PROMETHEE Flow Table

| Rank | action | Phi | Phi+ |
|---|---|---|---|
| 1 | Yasuo | 0,6900 | 0,7050 |
| 2 | LeBlanc | 0,1550 | 0,4150 |
| 3 | Zed | -0,0350 | 0,2650 |
| 4 | Galio | -0,1450 | 0,2700 |

As a result, it can be said that at Summoner's Rift the best character will be Yasuo. When the same methods applied at Howling Abyss, researchers have found that the best character for it will be Yasuo as well but since player knowledge is very crucial at Howling Abyss, they have decided that Jİnx will be the best choice for Howling Abyss.

Summerville et al (2016) studied the drafting phase of DotA2, which is another MOBA game that is very similar to League of Legends, to see its effect on the game result. The study contained a dataset of 1518 matches. They used bayes nets and Long Short-Term Memory Recurrent Neural Networks for their analysis. In Bayes Nets analysis, they created 3 different bayes nets. Full topology can be seen in the figure 1.26.

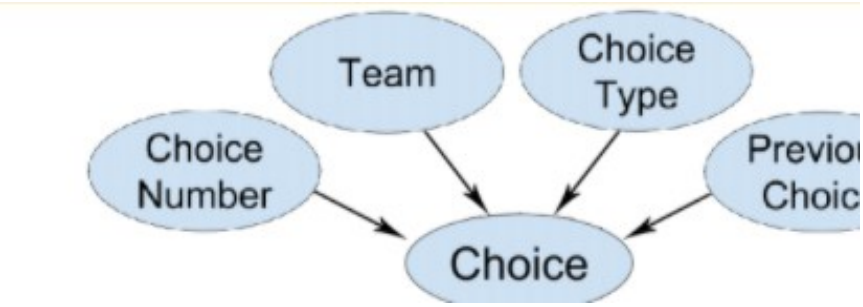



**Figure 1.25 :** Topology of the full bayes net (Summerville et al, 2016).

In mathematical terms this network represents can be shown in equation (1.4).

$$P\,r(C|N, T, C_t, C_{i-1}) \quad (1.4)$$

In LSTM analysis, the training is done on a One-Hot encoding. Categorical Cross-Entropy loss is used for the training of the model. The encoding consists of 137 words which are a mixture of the: 111 champions, 2 pick or ban selection, 2 teams which are A and B, 20 digits, 1 unique beginning word, 1 unique null word, every selection in draft consists of 4 words, type of selection which are ban or pick, selection number (1-20), teams, champions.

The accuracy results of this both techniques can be seen from table 1.13.

**Table 1.13 :** Success Rates of the Methods (Summerville et al, 2016).

| Method | Avg. Eval. Success% | Avg. Test Success% |
|---|---|---|
| Bayes Net- Nulligram | 4.93 | 4.95 |
| Bayes Net- Choice Number | 9.91 | 10.05 |
| Bayes Net- Full | 10.55 | 11.48 |
| LSTM | 13.42 | 11.94* |

At this research the best result is found with the LSTM method.

Conley and Perry's study is very similar to the previous one. In this work, they exported 56,691 matches and used 90% of them for the training set and remaining for the test set. They used feature vector method to identify 106 heroes and the winning team which can be seen from the equations (1.5), (1.6) and (1.7).

$$x_i = \begin{cases} 1, & \text{if a radiant player played as the hero with id i} \\ 0, & \text{otherwise} \end{cases} \tag{1.5}$$

$$x_{108+i} = \begin{cases} 1, & \text{if a dire player played as the hero with id i} \\ 0, & \text{otherwise} \end{cases} \tag{1.6}$$

$$y = \begin{cases} 1, & \text{if a radiant team won} \\ 0, & \text{otherwise} \end{cases} \tag{1.7}$$

First they used logistic regression to do their analysis. Then, they used k-nearest neighbors to enhance their system to not only detect winnings while those heroes are present but show heroes' relationship with each other. The team used this combined

distance and weighting function for knn calculations which can be seen in equation (1.8).

$$w_i=\left(\frac{\sum_{j=1}^{216} \mathbf{AND(qj,xj)}}{\mathbf{NUM\_IN\_QUERY}}\right)^d \quad (1.8)$$

After evaluating them one by one, they decided to choose dimension value for knn as 4 to have best accuracy. Later, they used these data to build a website which recommends heroes to the players.

After researching the machine learning techniques, some algorithms stepped forth. The most suitable one for our study is the MinMax algorithm. This algorithm is mostly used in zero-sum games such as chess and card battle games. For instance, Permana (2019) studied a Min-Max model for a card battle game, which consists of 2 players and in his study he created an NPC(Non player character) against a player. In order for NPC to make the most proper move, he used the Min Max algorithm. In this card battle game, there are different types of cards such as;

Willowisp (HP 5, ATK 0, Cost 1)

Succubus (HP 5, ATK 5, Cost 3)

Cockatrice (HP 8, ATK 3, Cost 3)

Windspirit (HP 6, ATK 2, Cost 2)

Dragon (HP 19, ATK 7, Cost 7)

Firespirit (HP 4, ATK 3, Cost 2)

Every step of the game should be calculated and decided by the NPC. If the first action belongs to NPC, according to the study it acts randomly. The next act of NPC will be decided due to its opponents moves.

**Table 1.14 :** Computations of MiniMax algorithm scenario (Permana, 2019).

| No | Area 1 | Area 2 | Area 3 | Area 4 | Area 5 | Damage | HP Left |
|---|---|---|---|---|---|---|---|
| 1 | Willowisp | - | Windspirit | - | - | 33.3% | 91.65% |
| 2 | Windspirit | - | Willowisp | - | - | 10% | 90% |
| 3 | Willowisp | - | Firespirit | - | - | 50% | 100% |
| 4 | Firespirit | - | Willowisp | - | - | 15% | 90% |
| 5 | Succubus | - | - | - | - | 25% | 50% |
| 6 | - | - | Succubus | - | - | 50% | 100% |
| 7 | Cockatrice | - | - | - | - | 15% | 50% |
| 8 | - | - | Cockatrice | - | - | 50% | 100% |
| 9 | Willowisp | - | - | - | - | 0% | 50% |
| 10 | | - | Willowisp | - | - | 0% | 40% |
| 11 | Firespirit | - | - | - | - | 15% | 50% |
| 12 | - | - | Firespirit | - | - | 50% | 100% |
| 13 | Windspirit | - | - | - | - | 10% | 50% |
| 14 | - | - | Windspirit | - | - | 33.3% | 87.5% |

In the Table 1.17, there are some calculations for different moves and decisions made by NPC. Damage in the table means the damage given to the opponent by NPC and HP Left value represents the health point value of NPC, therefore we aim to maximize damage and hp values. NPC chooses the best action so move 3,6,8 or 12 will be chosen because all of them have the most HP Left value and Damage value.

Oliveira et al (2017), also tried to use MinMax algorithm to build automatic team compositions in League of Legends. They developed a tool to assist League of Legend players in the draft phase by using linear regression and MinMax algorithms. First they developed the MinMax tree with some restrictions such as prioritising meta-game characters and considering the draft phase's selection mode which is 1:2:2:2:2:1. To calculate a possible team's potential win rate, the dataset from Brazillian League of Legends Championship was used. Moreover, with alpha-beta optimization, their tool suggests the best team composition to the players.

## 1.3 Hypothesis

After reviewing related works of AI applications in League of Legends, it is determined that using logistic regression will be ideal to guide players through the picking phase to find the optimal choice for them. First, a big amount of data which will contain every information of a LoL match will be retrieved from Riot Games API. This study group will involve high skilled players which are mainly in platinum, diamond and master ELOs. The algorithm will be created in Python with

the usage of this data. In conclusion, it is aimed to suggest optimal character choices that will guide players in the picking phase.

### 1.4 Methodology

After the determination of the algorithm which will be used in this thesis, it is necessary to determine the path of the study. The first step is to gather data from Riot API and parse the data with needed variables. Then one hot encoding will be applied to these variables since they are all categorical values. Using these categorical variables, a logistic regression model will be built. This model will provide a win rate. Using this team composition win rate and additional parameters, it will be possible to create a new win rate which will be used to suggest a champion for a player.

## 2. DESIGN/MODELLING

### 2.1 Riot API

As mentioned in the methodology part, the first step in this study is to gather data. This necessary data will be gathered from RIOT API with the provided API code. RIOT API's interface can be seen from the Figure 9.1.

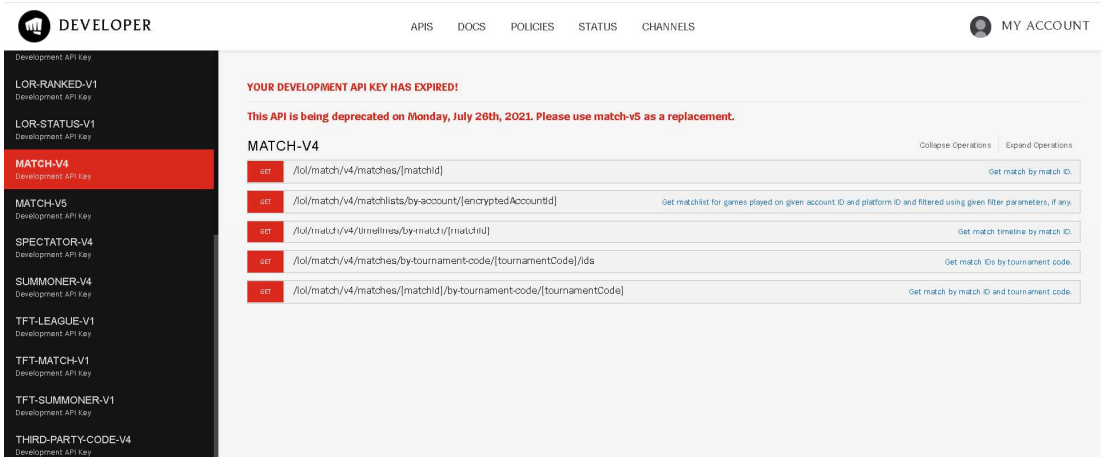



**Figure 2.1 :** Riot API interface.

From the “MATCH-V4” database, all the information needed for this project can be provided. With the usage of high elo match ids, a big amount of data will be gathered.

### 2.2 JSON Format

After gathering the needed amount of data, which is in the JSON format, this data should be converted to csv format in order to be used in a logistic regression model. JSON stands for “JavaScript Object Notation”. It is an easy text format for storage and transportation of data. An example of JSON format can be seen from the Figure 2.2.

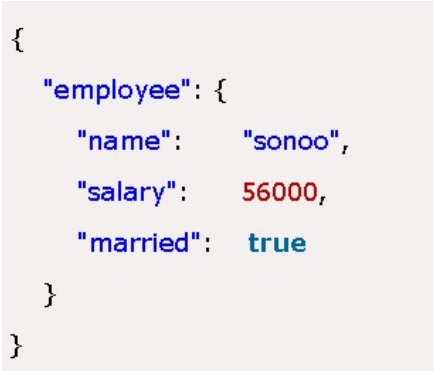



**Figure 2.2:** JSON format example.

By using Python, the data will be eliminated and converted into a csv file. The final state of the data can be seen from Figure 2.3.

| championid1,championid2,championid3,championid4,championid5,championid6,championid7,championid8,championid9,championid10,t1w |
|---|
| 81,67,3,121,71,119,2,63,72,12,1 |
| 17,67,127,48,126,39,100,63,123,98,0 |
| 50,2,118,72,12,147,153,127,126,71,0 |
| 148,83,57,112,93,81,100,63,137,25,0 |
| 96,20,39,121,43,54,56,129,123,25,0 |
| 39,30,147,137,93,81,109,75,21,3,0 |
| 3,122,107,72,49,119,9,4,121,25,0 |
| 69,130,147,137,49,127,153,98,48,12,0 |
| 83,153,137,127,16,44,11,81,72,98,0 |
| 146,88,147,127,12,39,100,75,126,78,0 |

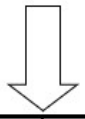
**Figure 2.3 :** Data after converted from JSON.

### 2.3 One Hot Encoding

After the conversion of data and other operations, it can be seen that the data still needs some arrangements. The values like “2” means that the champion with id 2 is taken by that player. Therefore all the values are categorical at this point. The best

way to use categorical variables for a logistic regression model will be one hot encoding. One hot encoding is a process where all the categorical variables are eliminated and a binary number for them is added. This process provides smaller sized data therefore the algorithm will be able to predict a more accurate result.An example of one hot encoding for a smaller size of data can be seen from the Figure 2.4.

| Ahri | Kai'Sa | Soraka | Warwick | Aatrox |
|---|---|---|---|---|
| Warwick | Syndra | Draven | Aatrox | Bard |



| Ahri | Kai'Sa | Soraka | Warwick | Aatrox | Syndra | Draven | Bard |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

**Figure 2.4 :** One hot encoding example.

In the first table there are 2 team compositions. After applying one hot encoding to this data, a smaller sized data which includes the same information is created. 1 specifies existent, 0 specifies nonexistent.

## 2.4 Logistic Regression

Logistic regression is a prediction model which is used for predicting a binary dependent variable. Since in this study, win rates are either 0 or 1, a logistic regression model would be appropriate. After arranging data, it will be fitted into a logistic regression model.

## 2.5 Suggestion

For the suggestion part, the model will require some inputs. In this model, it is assumed that the last player picking in the picking phase is using this model, and the suggestion is made due to previously picked champions. The player's own champion pool, their own win rate with every different champion and the champions' own

general win rate in their positions are considered as well. With the combination of these different parameters, an accurate champion suggestion will be made with the model.

### 2.6 General Flow of The Design

General flow of this studies design can be seen from the Figure 2.5.

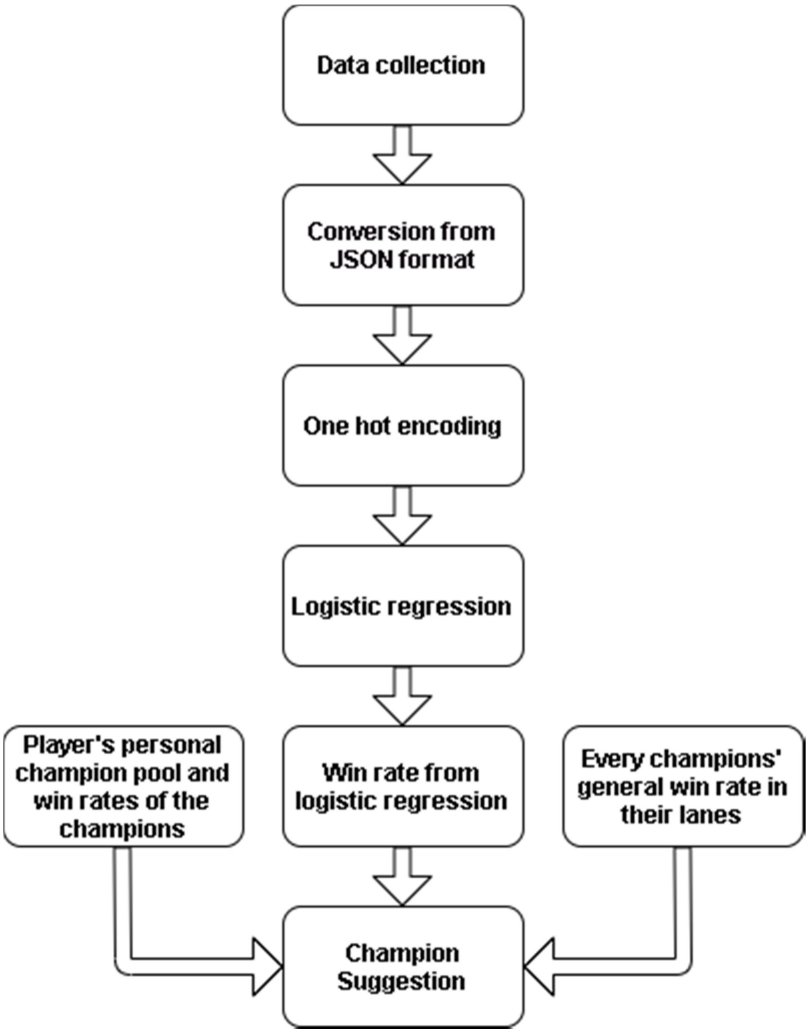



**Figure 2.5 :** General flow of the study.

## 3. APPLICATION

The application part consists of 4 main categories like data collection, feature engineering and data editing, one hot encoding and logistic regression.

### 3.1 Data Collection

Data that is needed for the analysis is gathered from Riot Games API under match-V4 section. This data can be collected with matchid code, region of the server and a developer key. Therefore, we applied for a developer key from the site which can be seen in Figure 3.1 and we acquired 170,000 matchids from kaggle.com with the high elos like diamond, challenger and master. This data mainly focuses on Korean (KR) server games but there are also around 10,000 amounts of data from West Europe server (EUW1) as well.

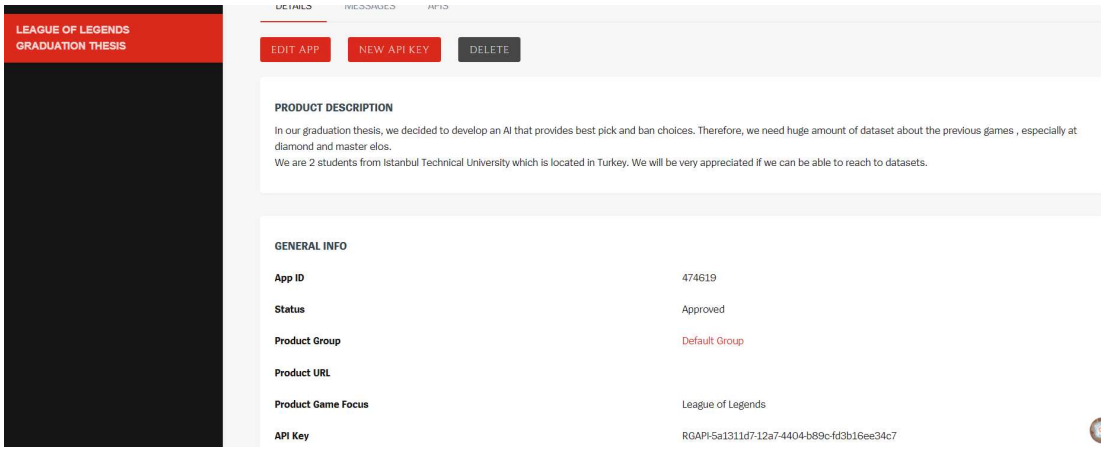



**Figure 3.1 :** Our developer key at Riot API.

The data we collected is json type and it consists 11 main parts. These are match, participant identities, player, team statistics, team bans, participants, rune, participant statistics, participant timeline and mastery. For instance, participant statistics part of the data can be seen in Figure 3.2.


**Figure 3.2 :** Example of the participant part of the data.

The Riot Developer site has a rate limit for retrieving data with 100 requests every 2 minutes. Therefore, we imported requests from library and put the limit in our code. Also, due to time limit and large data size, we only exported 10,000 data to set our program as it can be seen from Figure 3.3. Then we exported rest of 160,000 data step by step and added them to our analysis.

```
5]: import csv
    import time
    import requests
    import pandas as pd
```

```
7]: game_ids = list()
    with open("eu_10k_games.csv") as csv_file:
        spam = csv.reader(csv_file, delimiter=",")
        for row in spam:
            game_ids.append(row[0])

    num_of_games = len(game_ids)
    print(f"There are {num_of_games} game ids.")

    There are 9880 game ids.
```

```
3]: qps = 100/120 # 100 query per 120 seconds
```

**Figure 3.3 :** Data collection from API.

After setting the time limit, we entered our developer key and sent get requests. We also created a status code for the occurred errors where these can be seen in Figure 3.4.

```
count = 0

for gid in game_ids[1:]: # first row is ignored since it is header
    if count % (num_of_games // 100) == 0:
        print(f"Progress: {round(count/num_of_games,3) * 100} %")
    count += 1
    # create the query
    q = f"https://euw1.api.riotgames.com/lol/match/v4/matches/{gid}?api_key=RGAPI-5a1311d7-12a7-440

    # send get request
    r = requests.get(q)
    time.sleep(qps)

    # if error occurs
    while r.status_code != 200:
        print(f"Error in gid {gid}. {r.status_code}, {q}")
        time.sleep(qps)
        r = requests.get(q)
        error_count += 1
        if error_count == 5:
            corrupted_game_ids.append(gid)
```

**Figure 3.4 :** Data collection from API (continued).

Finally, we converted the data frame to a csv file and showed the “gameid”s that we couldn’t retrieve where it can be seen in Figure 3.5.

```
In [85]: # save DataFrame as .csv file
         df.to_csv("all_games9-10.csv")

In [86]: print("These ids could not retrieved: ", corrupted_game_ids)
         These ids could not retrieved:  []

In [68]: df.size
Out[68]: 1998
```

**Figure 3.5 :** Conversion of the data frame to a csv file.

### 3.2 Feature Engineering and Data Editing

The data which is gathered and converted it into csv file like mentioned earlier is json type and it consists 11 main parts. Due to these parts having too many statistics, data size was very high and it was unusable. Therefore, we decided to simplify the data for better and more sustainable dataset which only has the statistics that is needed for the analysis.

Firstly, we imported pandas, json, csv and sys from the python library. Then we opened our csv file which consists the collected dataset. This file has 2 columns like game_id where id of each game is written and data where all information about that game is written where can be seen in Figure 3.6

```
In [33]: import pandas as pd
         import json
         import csv
         import sys

In [34]: df = pd.read_csv("all_games9-10.csv")
         df
```

Out[34]:

| | Unnamed: 0 | game_id | data |
|---|---|---|---|
| 0 | 0 | 4225086418 | {"gameId":4225086418,"platformId":"KR","gameCr... |
| 1 | 1 | 4224499502 | {"gameId":4224499502,"platformId":"KR","gameCr... |
| 2 | 2 | 4224299083 | {"gameId":4224299083,"platformId":"KR","gameCr... |
| 3 | 3 | 4224292080 | {"gameId":4224292080,"platformId":"KR","gameCr... |
| 4 | 4 | 4222953957 | {"gameId":4222953957,"platformId":"KR","gameCr... |
| ... | ... | ... | ... |
| 1994 | 1994 | 4228822566 | {"gameId":4228822566,"platformId":"KR","gameCr... |
| 1995 | 1995 | 4226414661 | {"gameId":4226414661,"platformId":"KR","gameCr... |

**Figure 3.6 :** Imported libraries and the data file of the collected games.

Secondly, we only took data column from our data frame and defined it as game_data. which can be seen in Figure 3.7.

```
In [35]: game_data = df["data"]

In [37]: game_data

Out[37]: 0       {"gameId":4225086418,"platformId":"KR","gameCr.
         1       {"gameId":4224499502,"platformId":"KR","gameCr.
         2       {"gameId":4224299083,"platformId":"KR","gameCr.
         3       {"gameId":4224292080,"platformId":"KR","gameCr.
         4       {"gameId":4222953957,"platformId":"KR","gameCr.
                                      ...
         1994    {"gameId":4228822566,"platformId":"KR","gameCr.
         1995    {"gameId":4226414661,"platformId":"KR","gameCr.
         1996    {"gameId":4224274716,"platformId":"KR","gameCr.
```

**Figure 3.7 :** New data frame called "game_data".

Thirdly, we created a new csv file to write our only needed features in it from the data. For analysing the effect of the team composition on the game outcome, we only need the chosen champions by all 10 players and the game's outcome. These features can be found under the "participants" part, so we wrote a code that writes the needed data into this csv file. Thanks to json.loads we loaded all games and with using nested loop with the range of amount of the data we have in the file and thanks to

json.dumps we chosed the each 10 champion id's that are played in 1 game and we defined it as "championids". Then we defined win or lose situation of each game as "win". We added both "championids" and "win" to "data_list" where these codes can be seen in Figure 3.8.

```
In [45]: dumped_games = open('all_games_dumped.csv', 'w')

In [46]: csv_writer = csv.writer(dumped_games)

In [47]: header=("championid1","championid2","championid3","championid4","championid5","championid6","championid7"
         csv_writer.writerow(header)

Out[47]: 126

In [48]:
         for i in range(1999):
             game = json.loads(game_data[i])
             data_list=[]
             for a in range(0,10):
                 json.dumps(game['participants'])

                 championids=game['participants'][a]['championId']

                 data_list.append(championids)
```

**Figure 3.8** Using json.loads and j.son.dumps to clear the data.

As it can be seen from the figure 3.9, the file named "all_games_dumped.csv" contains only the needed data from the dataset with the headers like championid1, championid2, championid3, championid4, championid5, championid6, championid7, championid8, championid9, championid10, t1w where t1w indicates "Team 1 Won". t1w is a Boolean variable and it is true when first is won and it is false when the first lost.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | championid1,championid2,championid3,championid4,ch | | | | |
| 2 | | | | | |
| 3 | 30,91,523,412,86,104,57,235,223,142,True | | | | |
| 4 | | | | | |
| 5 | 39,111,142,236,245,91,110,78,76,432,True | | | | |
| 6 | | | | | |
| 7 | 22,12,38,104,266,516,64,429,412,7,True | | | | |
| 8 | | | | | |
| 9 | 5,68,50,555,59,875,432,105,81,245,True | | | | |
| 10 | | | | | |
| 11 | 39,134,523,64,25,142,497,21,164,58,True | | | | |

**Figure 3.9 :** Couple rows from "all_games_dumped.csv" file.

In the all_games_dumped we have a set of champion IDs and the outcome of the game in boolean form. We wanted to apply one hot encoding to our data, however our champion IDs were random. For instance, there were champions with champion IDs such as 497 and 875 but currently we only have 154 champions in the game. That means there are large gaps between the champion IDs and that would be a problem for one hot encoding. We have arranged the champion IDs from 1 to 154 according to our new list which can be seen from Figure 3.10. For example, Sett's original ID was 875, we have changed it to 152.

| | A | B | C |
|---|---|---|---|
| 1 | API ID | CHAMP | NEW ID |
| 128 | 236 | Lucian | 127 |
| 129 | 238 | Zed | 128 |
| 130 | 240 | Kled | 129 |
| 131 | 245 | Ekko | 130 |
| 132 | 246 | Kiyana | 131 |
| 133 | 254 | Vi | 132 |
| 134 | 266 | Aatrox | 133 |
| 135 | 267 | Nami | 134 |
| 136 | 268 | Azir | 135 |
| 137 | 350 | Yuumi | 136 |
| 138 | 360 | Samira | 137 |
| 139 | 412 | Thresh | 138 |
| 140 | 420 | Illaoi | 139 |
| 141 | 421 | Rek'Sai | 140 |

**Figure 3.10 :** New champion id arrangement.

Another problem with the data was win or lose situation being in Boolean form. We used find and change in Microsoft excel to fix this to 0 when first team lost and 1 to when first team is won. After simplifying and arranging our data, as it can be seen from Figure 3.11, the data is became appropriate for the usage of one hot encoding on it.

| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | championid1,championid2,championid3,championid4,championid5,championid6,championid7,championid8,championid9,championid10,t1w | | | | | | | | | | | | |
| 2 | 93,80,127,67,50,26,126,73,140,63,0 | | | | | | | | | | | | |
| 3 | 4,136,66,112,152,92,55,49,107,39,0 | | | | | | | | | | | | |
| 4 | 140,75,127,43,45,112,115,81,107,124,0 | | | | | | | | | | | | |
| 5 | 60,107,146,143,142,92,138,54,130,128,0 | | | | | | | | | | | | |
| 6 | 85,133,130,111,21,48,77,63,126,32,0 | | | | | | | | | | | | |
| 7 | 18,101,63,152,136,107,143,21,52,142,1 | | | | | | | | | | | | |
| 8 | 58,94,142,44,109,131,95,84,83,149,1 | | | | | | | | | | | | |
| 9 | 152,55,75,21,39,130,101,147,82,98,0 | | | | | | | | | | | | |
| 10 | 30,41,78,145,116,99,55,97,136,142,0 | | | | | | | | | | | | |
| 11 | 145,70,151,103,144,119,62,102,138,142,1 | | | | | | | | | | | | |
| 12 | 144,55,135,76,121,21,152,71,138,130,0 | | | | | | | | | | | | |
| 13 | 126,116,146,67,149,107,84,147,119,56,0 | | | | | | | | | | | | |
| 14 | 120,60,71,72,4,144,115,18,103,107,1 | | | | | | | | | | | | |
| 15 | 11,151,18,14,117,24,29,3,134,61,0 | | | | | | | | | | | | |
| 16 | 152,105,133,138,112,148,25,92,39,19,1 | | | | | | | | | | | | |

**Figure 3.11 :** Simplified data with new champion ids and win or loss.

### 3.3 One Hot Encoding

Since all these 10 picks are dummy variables, they should be defined to prevent wrong results before model training. There are couple ways to define dummy variables. The most common way to is using get_dummies from pandas library. Unfortunately, since there are 154 champions in the recent meta this means 154*10 dummy variables for our model. This amount of variables is not usable for creating a logical model so we decided apply one hot encoding to our data instead. With one hot encoding, variables only can get values as 1,0 or -1 where 1 represents hero is picked by the first team, -1 shows that the champion is picked by the second team and 0 represent that the champion is not picked by anyone. Thanks to one hot encoding we reduced our dummy variable amount from 1540 to 154 where each variable represent each hero in the game.

We started writing to code with installing keras and tensorflow libraries to python. Then we imported pandas, numpy, keras.utils and csv libraries which can be seen in Figure 3.12.

```
_version >=  3.6 ->google-auth<2,>=1.6.3->

In [3]: import pandas as pd
        from numpy import array
        from keras.utils import to_categorical
        import csv
```

**Figure 3.12 :** Importing necessary libraries for one hot encoding process.

We opened our final dataset which consists 170,000 different game data as a new dataframe and dropped the “t1w” column since it shows result of the game. We arrayed the new data frame and created “encoded” with applying to_categorical function from keras.utils on the new dataframe. This function successfully applied one hot encoding on the data frame.Then, we opened a new csv file to write this new encoded data in it. Codes of this can be seen in Figure 3.13.

```
In [9]: df=pd.read_csv('170k_son.csv')
        df2=df.drop(['t1w'],axis=1)
        df2=array(df2)

In [10]: encoded = to_categorical(df2)
         data_file5 = open('onehotencoding_son_2905.csv', 'w'
```

**Figure 3.13 :** One hot encoding application to the new data set.

In Figure 3.14, creation of 10 different lists for each pick can be seen.

```
In [11]: list1=[]
         list2=[]
         list3=[]
         list4=[]
         list5=[]
         list6=[]
         list7=[]
         list8=[]
         list9=[]
```

**Figure 3.14 :** 10 new lists for each pick in the game.

Finally, a nested loop is created in the range of dataset size and first picks are appended to list1, second picks are appended to list2, third lists are appended to list3, fourth lists are appended to list4, fifth picks are appended to list 5, sixth picks are appended to list6, seventh picks are appended to list7, eighth picks are appended to list8, ninth picks are appended to list9 and tenth picks are appended to list10 from encoded. As it mentioned earlier, opposion teams’ picks are shown with -1, therefore we added “-“ in front of the encoded in fifth, sixth, seventh, eighth, ninth and tenth pick. We arrayed all the lists and created a eleventh list called list11 where we

summed all these lists together. At last, we wrote this new list to our csv file. All list arrangement can be seen in Figure 3.15.

```
In [12]: for k in range(171856):
             list1=encoded[k][0]
             list2=encoded[k][1]
             list3=encoded[k][2]
             list4=encoded[k][3]
             list5=encoded[k][4]
             list6=-encoded[k][5]
             list7=-encoded[k][6]
             list8=-encoded[k][7]
             list9=-encoded[k][8]
             list10=-encoded[k][9]
             list1= array(list1)
             list2= array(list2)
             list3= array(list3)
             list4= array(list4)
             list5= array(list5)
             list6= array(list6)
             list7= array(list7)
             list8= array(list8)
             list9= array(list9)
             list10=array(list10)
             list11=list1+list2+list3+list4+list5+list6+list7+list8+list9+list10
             list11=array(list11)
             csv_writer.writerow(list11)
```

**Figure 3.15 :** Final stage at one hot encoding.

In one hot encoding values of 1, 0 and -1 are float and there is a need for turning them into integers. We did this on excel with turning csv file to a xlsx file and used decrease decimal. Afterwards, we added our “t1w” column to this file. We named this new file as 170k_integer_excel.xlsx.

### 3.4 Personal & Champion Win Rates

For personal win rate calculation, data is collected from the user named “RenektoNSaL”. His every ranked matchfrom 2018 to 2021 where he played top players are involved in the calculation of his win rate. The reason behind only choosing top players is this player’s play pattern mainly focuses on that lane. Some of statistics of “RenektoNSaL” can be seen in Figure 3.16.

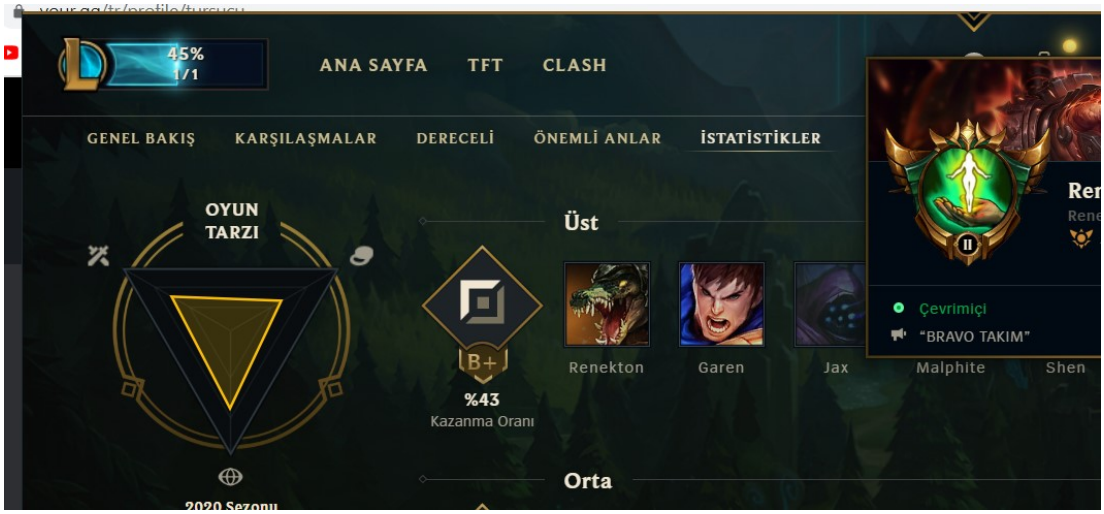


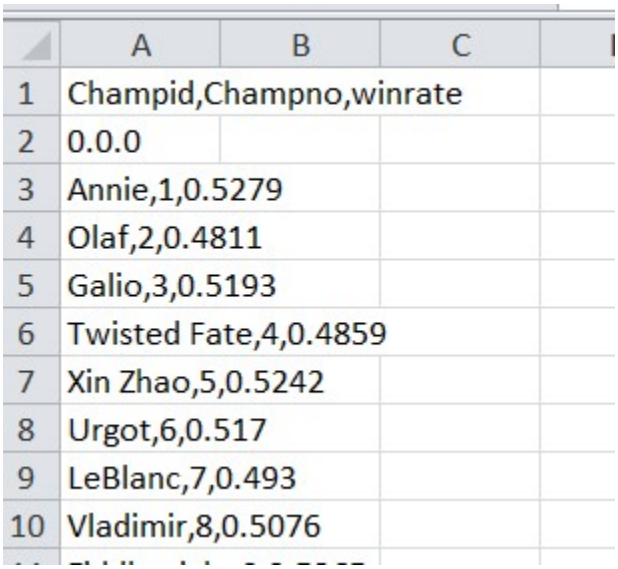

**Figure 3.16 :** Player "RenektoNsaL's statistics page in the game interface.

Personal win rates are collected in a csv file called “personal_win_rate.csv” which can be seen in Figure 3.17. There are 3 columns like “champion” which shows the champions that are in person’s champion pool, “winrate” which shows person’s win rate with that champion and “champid” where it represents that champion’s id number.

| | A | B | C |
|---|---|---|---|
| 1 | Champion,Winrate,Champid | | |
| 2 | Renekton,0.6027083333333 | | |
| 3 | Garen,0.49375,77 | | |
| 4 | Malphite,0.2,50 | | |
| 5 | Jax,0.67,24 | | |

**Figure 3.17 :** “personal_win_rate_csv” file.

There is also a need for champion’s win rate for our modelling. Therefore we also acquired that data from op.gg and created an csv file for the champion’s win rates which can be seen in Figure 3.18.



| | A | B | C |
|---|---|---|---|
| 1 | Champid,Champno,winrate | | |
| 2 | 0.0.0 | | |
| 3 | Annie,1,0.5279 | | |
| 4 | Olaf,2,0.4811 | | |
| 5 | Galio,3,0.5193 | | |
| 6 | Twisted Fate,4,0.4859 | | |
| 7 | Xin Zhao,5,0.5242 | | |
| 8 | Urgot,6,0.517 | | |
| 9 | LeBlanc,7,0.493 | | |
| 10 | Vladimir,8,0.5076 | | |

**Figure 3.18 :** Each champion's win rate is acquired by op.gg.

While writing the code, we started with importing csv and pandas libraries and defined our data frame from personal_win_rate.csv file where it can be seen in Figure 3.19.

```
In [47]: import csv
         import pandas as pd

         df1=pd.read_csv('personal_win_ra
         x=(df1["Champid"])
         y=(df1["Winrate"])
         df1
```

Out[47]:

| | Champion | Winrate | Champid |
|---|---|---|---|
| 0 | Renekton | 0.602708 | 54 |
| 1 | Garen | 0.493750 | 77 |
| 2 | Malphite | 0.200000 | 50 |
| 3 | Jax | 0.670000 | 24 |

**Figure 3.19 :** Importing libraries and defining the data frame.

In our system, we decided to evaluate first 5 champions from individual's champion pools so we created "champ1", "champ2", "champ3", "champ4" and "champ5" where they show first 5 champion ids respectively and "champ1_winr",

“champ2_winr”, “champ3_winr”, “champ4_winr”, “champ5_winr” to show player’s win rate with those champion. The codes for this can be seen in Figure. 3.20.

```
In [48]: champ1=x[0]
         champ1_winr=y[0]
         champ2=x[1]
         champ2_winr=y[1]
         champ3=x[2]
         champ3_winr=y[2]
         champ4=x[3]
         champ4_winr=y[3]
         champ5=x[4]
```

**Figure 3.20 :** Defining champ1, 2, 3, 4 and 5 with the usage of data frame.

Since we needed to merge personal and champion win rates, we opened a new csv file called “personal_and_champion_win_rates.csv” and we defined champion_win_rates.csv file as a data frame in the Figure 3.21.

```
In [49]: data_file = open('personal_and_champion_win_rates.csv', 'w', newli
         csv_writer = csv.writer(data_file)
         header=("championid","winrate","championwinrate")
         csv_writer.writerow(header)

Out[49]: 36

In [50]: df=pd.read_csv('champion_win_rates.csv')
         df
```

Out[50]:

| | Champid | Champno | winrate |
|---|---|---|---|
| **0** | 0.0.0 | NaN | NaN |
| **1** | Annie | 1.0 | 0.5279 |
| **2** | Olaf | 2.0 | 0.4811 |
| **3** | Galio | 3.0 | 0.5193 |
| **4** | Twisted Fate | 4.0 | 0.4859 |
| **...** | ... | ... | ... |
| **150** | Rell | 150.0 | 0.4903 |
| **151** | Pyke | 151.0 | 0.4979 |

**Figure 3.21 :** Defining the csv file that contains champion win rates as a new data frame.

Finally, we created 5 lists for each champion in the player's pool and added those champion ids, personal and champion win rates to each list. Then we used csv_writer to add these lists as new rows to our csv file in the Figure 3.22.

```
In [5]: list1=[]
        list1.append(0)
        list1.append(champ1)
        list1.append(champ1_winr)
        list1.append(df["winrate"][champ1])

        list2=[]
        list2.append(1)
        list2.append(champ2)
        list2.append(champ2_winr)
        list2.append(df["winrate"][champ2])

        list3=[]
        list3.append(2)
        list3.append(champ3)
        list3.append(champ3_winr)
        list3.append(df["winrate"][champ3])

        list4=[]
        list4.append(3)
        list4.append(champ4)
        list4.append(champ4_winr)
        list4.append(df["winrate"][champ4])

        list5=[]
        list5.append(4)
        list5.append(champ5)
        list5.append(champ5_winr)
        list5.append(df["winrate"][champ5])

        csv_writer.writerow(list1)
        csv_writer.writerow(list2)
        csv_writer.writerow(list3)
        csv_writer.writerow(list4)
        csv_writer.writerow(list5)
```

**Figure 3.22 :** Creation of 5 different lists for each champion in the pool.

### 3.5 Team Compositions

Since the system we build suggests last pick for the first team, the other selections are needed to be added to the system. This is also entered manually with asking the user about selections. We created a new csv file called "picks.csv". Then we created each team composition with the champions that are on the players' pool in the Figure 3.23.

```
In [1]: import csv
        data_file = open('picks.csv', 'w', newline='
        csv_writer = csv.writer(data_file)
        header=("championid1","championid2","champio
        csv_writer.writerow(header)
```

```
Out[1]: 126
```

```
In [2]: import pandas as pd
        df2=pd.read_csv('personal_and_champion_win_r
        X2=df2["championid"]
        df2
```

Out[2]:

| | championid | winrate | championwinrate |
|---|---|---|---|
| **0** | 54 | 0.602708 | 0.4881 |
| **1** | 77 | 0.493750 | 0.5050 |

**Figure 3.23 :** Creating the "picks.csv" file and defining new data frame from previous files.

Manual entries of the champion selections of other players' can be seen in Figure 3.24. In here, fifth pick is entered but then changed with the champions in the pool to later using in team composition win rate calculations. The reason 153 is added to the list is to prevent size differences of the lists while doing one hot encoding.

```
In [4]: list=[]
        list.append(0)
        pick1=int(input("first pick:"))
        list.append(pick1)
        pick2=int(input("second pick:"))
        list.append(pick2)
        pick3=int(input("third pick:"))
        list.append(pick3)
        pick4=int(input("fourth pick:"))
        list.append(pick4)
        pick5=int(input("fifth pick:"))
        list.append(pick5)
        pick6=int(input("sixth pick:"))
        list.append(pick6)
        pick7=int(input("seventh pick:"))
        list.append(pick7)
        pick8=int(input("eighth pick:"))
        list.append(pick8)
        pick9=int(input("nineth pick:"))
        list.append(pick9)
        pick10=int(input("tenth pick:"))
        list.append(pick10)
        list.append(153)

        first pick:15
        second pick:111
        third pick:13
        fourth pick:41
```

**Figure 3.24 :** Taking inputs from the user to get previous champion selections.

After taking inputs, we changed fifth pick with the heroes in the player's pool and wrote each row to the new file in the Figure 3.25.

```
In [5]: for i in range(5):
            list[5]=X2[i]
            csv_writer.writerow(l
```

**Figure 3.25 :** Changing fifth pick with the heroes in the pool.

### 3.6 Logistic Regression

First we imported required libraries to run a logistic regression, then we defined our data frame. As it can be seen in Figure 3.26, there are 154 variables and "win" column is the outcome that we want to predict where it gets 1 when the first team wins and 0 when it loses.

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.linear_model import LogisticRegression
        np.random.seed(0)
        from sklearn.model_selection import train_test_split

In [4]: df = pd.read_excel("170k_integer_excel.xlsx")

        df
```

Out[4]:

| | id1 | id2 | id3 | id4 | id5 | id6 | id7 | id8 | id9 | id10 | ... | id146 | id147 | id148 | id149 | id150 | id151 | id152 | id153 | id154 | win |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **168021** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **168022** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **168023** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **168024** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | ... | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| **168025** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -1 | 0 | 1 |

168026 rows × 155 columns

**Figure 3.26 :** Importing libraries and defining data frame for the logistic regression.

Afterwards, X and y of the model is defined in the Figure 3.27.

```
In [5]: X=df.drop(['win'],axis=1)
        y=(df['win'])
        print(df)
```

```
        id1  id2  id3  id4  id5  id6  id7  id8  id9  id10  ...  id146  id147  \
0         0    0    0    0    0    0    0    0    0     0  ...      0      0
1         0    0    0    1    0    0    0    0    0     0  ...      0      0
2         0    0    0    0    0    0    0    0    0     0  ...      0      0
3         0    0    0    0    0    0    0    0    0     0  ...      1      0
4         0    0    0    0    0    0    0    0    0     0  ...      0      0
...     ...  ...  ...  ...  ...  ...  ...  ...  ...   ...  ...    ...    ...
168021    0    0    0    1    0    0    0    0    0     0  ...      0      0
168022    0    0    0    0    0    0    0    0    0     0  ...      0      0
168023    0    0    0    0    0    0    0    0    0     0  ...      0      1
168024    0    0    0    0    0    0    0    0   -1     0  ...      1      0
168025    0    0    0    0    0    0    0    0    0     0  ...      0      0

        id148  id149  id150  id151  id152  id153  id154  win
0           0      0      0      0      0      0      0    0
1           0      0      0      0      1      0      0    0
2           0      0      0      0      0      0      0    0
3           0      0      0      0      0      0      0    0
4           0      0      0      0      0      0      0    0
...       ...    ...    ...    ...    ...    ...    ...  ...
168021      0      0      0      0      0      0      0    1
168022      0      0      0      0      0      0      0    1
168023      0      0      0      0      0      0      0    0
168024      0      1      0      1      0      0      0    1
168025      0      0      0      0     -1     -1      0    1

[168026 rows x 155 columns]
```

**Figure 3.27 :** Defining X and y for the logistic regression.

After defining X and y we splitted the train and tests where 20% of the data is the test set of the model in the Figure 3.28.

```
In [6]: X_train, X_test , y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
        print(y_test)

167668    1
132324    1
85757     1
135341    0
162337    0
         ..
30914     1
92431     0
36935     0
62757     0
43857     1
Name: win, Length: 33606, dtype: int64
```

**Figure 3.28 :** Splitting train and test sets.

Following train and test split, we fit our X_train into a logistic regression model which can be seen in Figure 3.29.

```
In [7]: log=LogisticRegression(solver='liblinear', random_state=0)
        log.fit(X_train, y_train)

Out[7]: LogisticRegression(random_state=0, solver='liblinear')
```

**Figure 3.29 :** Logistic regression model.

## 4. OUTPUT ANALYSIS

In output analysis, first we evaluated classification report and confusion matrix. Later, we created the artificial intellengence to according to those results.

### 4.1 Classification Report and Confusion Matrix

After running logistic we created classification report and the confusion matrix of the model. When classification report is checked in the Figure 4.1, it can be seen that the accuracy is 54% and when confusion matrix is checked in the Figure 4.2, it can be seen that from 33606 sized dataset, actual 1 valued 7750 ones and actual 0 valued 9124 ones were misclassified. Therefore, we deducted that team compositions effect on the game outcome was not that significant compared to other effects in the game.

```
In [8]: predictions = log.predict(X_test)
        print(predictions)
        from sklearn.metrics import classification_report
        print(classification_report(y_test, predictions))
```

```
[0 0 0 ... 1 1 1]
              precision    recall  f1-score   supp

           0       0.54      0.55      0.54     16
           1       0.55      0.54      0.54     16

    accuracy                           0.54     33
```

**Figure 4.1 :** Classification report of the logistic regression.

```
In [9]: from sklearn import metrics
        cnf_matrix = metrics.confusion_matrix(y_test, predictions)
        cnf_matrix
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline
        class_names=[0,1] # name  of classes
        fig, ax = plt.subplots()
        tick_marks = np.arange(len(class_names))
        plt.xticks(tick_marks, class_names)
        plt.yticks(tick_marks, class_names)
        # create heatmap
        sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnB
        ax.xaxis.set_label_position("top")
        plt.tight_layout()
        plt.title('Confusion matrix', y=1.1)
        plt.ylabel('Actual label')
        plt.xlabel('Predicted label')
```

```
Out[9]: Text(0.5, 257.44, 'Predicted label')
```

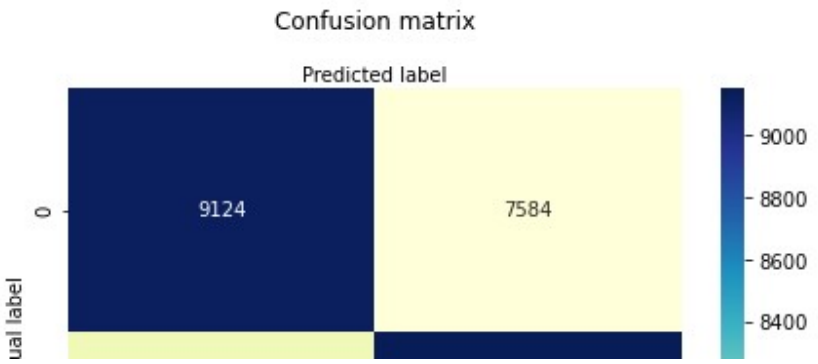



**Figure 4.2 :** Confusion matrix of the logistic regression.

### 4.2 Creation of the Champion Suggesting Aritifcial Intelligence

For creating the suggesting champions part for the players, first we imported the previous csv file name 'personal_and_champion_win_rates.csv' where it has player's champion pool heroes personal and own win rates.

```
In [13]: import pandas as pd
         df2=pd.read_csv('personal_and_champion_win_rates.csv')
         X2=df2["championid"]
         y2=(df2["winrate"])
         z2=(df2["championwinrate"])
         df2
```

Out[13]:

| | championid | winrate | championwinrate |
|---|---|---|---|
| 0 | 54 | 0.602708 | 0.4881 |
| 1 | 77 | 0.493750 | 0.5050 |
| 2 | 50 | 0.200000 | 0.4978 |
| 3 | 24 | 0.670000 | 0.5035 |
| 4 | 6 | 0.556667 | 0.5170 |

**Figure 4.3** Importing previously created csv file called "personal_and_champion_win_rates.csv" and defining each column as x,y and z respectively.

Afterwards, we imported the "picks.csv" to enter the 9 other picks at the game and arrayed them. Then, we encoded them. In here, 153 is written to prevent encoding to be shorter than showing all 154 possible picks.

```
In [16]: df=pd.read_csv('picks.csv')
         df=array(df)
         df

Out[16]: array([[ 15, 111,  13,  41,  54,   6,  34,  48,  52,  59, 153],
                [ 15, 111,  13,  41,  77,   6,  34,  48,  52,  59, 153],
                [ 15, 111,  13,  41,  50,   6,  34,  48,  52,  59, 153],
                [ 15, 111,  13,  41,  24,   6,  34,  48,  52,  59, 153],
                [ 15, 111,  13,  41,   6,   6,  34,  48,  52,  59, 153]],
               dtype=int64)

In [17]: encoded = to_categorical(df)
         encoded

Out[17]: array([[[0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
```

**Figure 4.4 :** Importing and encoding the "picks.csv".

Similar to previous, a nested loop is created in the range of dataset size and first picks are appended to list1, second picks are appended to list2, third lists are appended to list3, fourth lists are appended to list4, fifth picks are appended to list 5,

sixth picks are appended to list6, seventh picks are appended to list7, eighth picks are appended to list8, ninth picks are appended to list9 and tenth picks are appended to list10 from encoded. As it mentioned earlier, opposion teams' picks are shown with -1, therefore we added "-" in front of the encoded in fifth, sixth, seventh, eighth, ninth and tenth pick. We arrayed all the lists and created a eleventh list called list11 where we summed all these lists together. Unlike the previous study, in here there is also list12 for fixing the addition of champion 153 with multiplication of 0. At last, we predicted each compositions' outcome one by one in the Figure 4.5.

```
In [18]: output=[]
         for i in range(5):
             list1=encoded[i][0]
             list2=encoded[i][1]
             list3=encoded[i][2]
             list4=encoded[i][3]
             list5=encoded[i][4]
             list6=-encoded[i][5]
             list7=-encoded[i][6]
             list8=-encoded[i][7]
             list9=-encoded[i][8]
             list10=-encoded[i][9]
             list1= array(list1)
             list2= array(list2)
             list3= array(list3)
             list4= array(list4)
             list5= array(list5)
             list6= array(list6)
             list7= array(list7)
             list8= array(list8)
             list9= array(list9)
             list10=array(list10)
             list12=encoded[0][10]*0
             list12=array(list12)
             list11=list1+list2+list3+list4+list5+list6+list7+list8+list
             list11=array(list11)

             input = list11.reshape((1, -1))

             output1=log.predict_proba(input)
             output.append(output1)
```

**Figure 4.5 :** Calculations of each team composition's win rate.

In the final_list, it can be seen that overall win rate is calculated by each hero in the game pool of the player with a nested loop. Due to lower accuracy percentage, the effects of team composition is taken as 0.4, personal win rate's is taken as 0.4 and champion win rate is chosen as 0.2. All these calculations can be seen in Figure 4.6.

After calculating every possible outcome, we made system to choose maximum winning possibility and suggest that option to the player.

```
In [20]: final_list=[]
         for i in range(5):
             comp_win_rate=output[i][:,1]
             personal_win_rate=y2[i]
             champion_win_rate=z2[i]
             final=(0.4*comp_win_rate)+(0.4*personal_win_rate)+(0.2*champion_win_rate)
             final_list.append(final)

         final_list

Out[20]: [array([0.53395038]),
          array([0.49792141]),
          array([0.37361805]),
          array([0.55936884]),
          array([0.5119215])]

In [21]: max(final_list)

Out[21]: array([0.55936884])

In [22]: print("You must choose hero with id:",X2[final_list.index(max(final_list))])

         You must choose hero with id: 24
```

**Figure 4.6 :** Calculation of final winning possibility for each 5 options.

## 5. CONCLUSION

## 6. REFERENCES


**Adams, A. Walker, S.** (2018), Real-world open-ended evolution: A league of legends adventure. *International Journal of Design & Nature and Ecodynamics*, 12(4), 458-469. doi: 10.2495/DNE-V12-N4-458-469

**Agha, B.** (2015). League of Legends Players and Esports (Master's thesis). McMaster University, Hamilton, Ontario.

**Ani, R., Harikumar, V.,Devan A. K., & Deepa, O. S.** (2019).Victory prediction in League of Legends using Feature Selection and Ensemble methods. *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, Madurai, India, 2019, pp. 74-77, doi: 10.1109/ICCS45141.2019.9065758

**Conley, K., & Perry, D.** (2013). How Does He Saw Me ? A Recommendation Engine for Picking Heroes in Dota 2. Retrieved from http://cs229.stanford.edu/proj2013/PerryConley-HowDoesHeSawMeARecommendationEngineForPickingHeroesInDota2.pdf

**Decelle, J.**, Hall, G., O'Donnell, L. (2015). The Importance of Matchmaking in League of Legends and its Effects on Users. Retrieved from http://web.cs.wpi.edu/~claypool/iqp/lol-match/report.pdf

**Ercan, M., Yıldırım, M., Oturak, Ç., Eren, T.** (2016). LEAGUE OF LEGENDS OYUNU KARAKTER SEÇİM PROBLEMİNİN ÇOK ÖLÇÜTLÜ KARAR VERME YÖNTEMLERİ İLE ÇÖZÜMÜ.*Uluslararası Eğitim Bilim ve Teknoloji Dergisi.*

**Ferrari, S.** (2013). From Generative to Conventional Play: MOBA and League of Legends. *DiGRA Conference*.

**Kim, S., Kim, D., Ahn, H. *et al.*** (2020). Implementation of user playstyle coaching using video processing and statistical methods in league of legends. *Multimed Tools Appl*. doi: https://doi.org/10.1007/s11042-020-09413-4

**Lin, L.** (2016). League of Legends Match Outcome Prediction. Retrieved from http://cs229.stanford.edu/proj2016/report/Lin-LeagueOfLegendsMatchOutcomePrediction-report.pdf

**Lohokare, A., Shah, A., & Zyda, M.** (2020). Deep Learning Bot for League of Legends. *Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment,* University of Southern California Los Angeles, California.

**Mora-Cantallops, M. Sicilia, M. A.** (2019). Team efficiency and network structure: The case of professional League of Legends. *Social Networks*, (Vol 58, pp 105-115).doi: https://doi.org/10.1016/j.socnet.2019.03.004

**Mora-Cantallops, M. Sicilia, M. A.** (2019). Player-centric networks in League of Legends. *Social Networks*, (Vol 55, pp 149-159).doi: https://doi.org/10.1016/j.socnet.2018.06.002

**Mora-Cantallops, M. Sicilia, M. A.** (2019). MOBA games: A literature review. *Entertainment Computing*, (Vol 26, pp 128-138).doi: https://doi.org/10.1016/j.entcom.2018.02.005

**Oliveira, V.D., Placides, B.J., Baffa, M.D., & Machado, A.F.** (2018). A Hybrid Approach To Build Automatic Team Composition In League of Legends. *Proceedings of SBGames 2017,* Instituto Federal de Educacao, Ciencia e Tecnologia do Sudeste de Minas Gerais, Departamento Academico da Ciencia da Computacao , Brasil, November 2nd - 4th, 2017.

**Permanana, S. D. H.** (2019). Implementation of Min Max Algorithm as Intelligent Agent on Card Battle Game. *International Journal of Information System & Technology* (Vol. 2, No. 2, pp. 53-58). doi:10.30645/ijstech.vv2i2.20

**Silva, A., Pappa, G., & Chaimowicz, L.** (2018). Continuous Outcome Prediction of League of Legends Competitive Matches Using Recurrent Neural Networks. *Proceedings of SBGames 2018,* Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil, October 29th – November 1st, 2018.

**Summerville, A.,Cook, M., Steenhuisen, B.** (2016). Draft-Analysis of the Ancients : Predicting Draft Picks in DotA 2 Using Machine Learning. *Experimental AI in Games: Papers from the AIIDE Workshop AAAI Technical Report WS-16-22.*

**Shores, K.B., He, Y., Swanenburg, K., Kraut, R., & Riedl, J.** (2014). The identification of deviance and its impact on retention in a multiplayer game. *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. doi: 10.1145/2531602.2531724.

**Veron, M., Marin, O., Monnet, S.** (2014). Matchmaking in multi-player on-line games: studying user traces to improve the user experience. *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop,* March 2014, pp 7-12. doi: 10.1145/2578260.2578265

**Wang, N., Li, L., Xiao, L., Yang, G., & Zhou, Y.** (2018). Outcome prediction of DOTA2 using machine learning methods. *Proceedings of 2018 International Conference on Mathematics and Artificial Intelligence*. doi: https://doi.org/10.1145/3208788.3208800

**Url1** <https://leagueoflegends.fandom.com/wiki/Minion_(League_of_Legends)>,data retrieved 29.12.2020

**Url-2** <https://lol.gamepedia.com/>, data retrieved 29.12.2020

**Url-3** <https://leagueoflegends.fandom.com/wiki/Mountain_Drake>, data retrieved 29.12.2020

**Url-4**
<https://leagueoflegends.fandom.com/wiki/Rift_Herald_(League_of_Legends)>, data retrieved 29.12.2020

**Url-5** <_https://oyunda.org/2020/06/league-of-legends-bir-sonraki-sezon-icin-yeni-bir-market-hazirliyor.html>, data retrieved 29.12.2020

**Url-6** < https://leagueoflegends.fandom.com/wiki/Champion_classes>, data retrieved 29.12.2020

## 7. APPENDICES

**APPENDIX A:** Data_collecting.ipynb

**APPENDIX B:** Data_editing.ipynb

**APPENDIX C:** One_hot_encoding.ipynb

**APPENDIX D:** Personal_champion_pool.ipynb

**APPENDIX E:** Team_compositions.ipynb

**APPENDIX F:** Logistic_regression_and_AI_creation.ipynb

## APPENDIX A

```
import csv
import time
import requests
import pandas as pd
```

In [82]:

```
game_ids = list()
with open("Challenger_Ranked_Games_9-10k.csv") as csv_file:
    spam = csv.reader(csv_file, delimiter=",")
    for row in spam:
        game_ids.append(row[0])

num_of_games = len(game_ids)
print(f"There are {num_of_games} game ids.")
```

```
There are 2000 game ids.
```

In [83]:

```
qps = 100/120 # 100 query per 120 seconds
error_count = 0
corrupted_game_ids = list()

df = pd.DataFrame(columns=["game_id", "data"])
```

In [84]:

```
count = 0

for gid in game_ids[1:]: # first row is ignored since it is header
    if count % (num_of_games // 100) == 0:
        print(f"Progress: {round(count/num_of_games,3) * 100} %")
    count += 1
    # create the query
    q =
f"https://kr.api.riotgames.com/lol/match/v4/matches/{gid}?api_key=RG
API-05bd6712-e319-4be2-9f25-5529b995b4f5"

    # send get request
    r = requests.get(q)
    time.sleep(qps)

    # if error occurs
    while r.status_code != 200:
        print(f"Error in gid {gid}. {r.status_code}, {q}")
        time.sleep(qps)
        r = requests.get(q)
        error_count += 1
        if error_count == 5:
            corrupted_game_ids.append(gid)
            break
    error_count = 0

    # add game id and game_data to DataFrame
    df = df.append({"game_id": gid, "data": r.text},
ignore_index=True)
```

In [85]:

```
# save DataFrame as .csv file
df.to_csv("all_games9-10.csv")
```

In [86]:

```
print("These ids could not retrieved: ", corrupted_game_ids)
```

```
These ids could not retrieved:  []
```

In [68]:

```
df.size
```

Out[68]:

```
1998
```

## APPENDIX B

```
import pandas as pd
import json
import csv
import sys
```

In [34]:

```
df = pd.read_csv("all_games9-10.csv")
df
game_data = df["data"]
```

In [37]:

```
game_data
```

Out[37]:

```
0       {"gameId":4225086418,"platformId":"KR","gameCr...
1       {"gameId":4224499502,"platformId":"KR","gameCr...
2       {"gameId":4224299083,"platformId":"KR","gameCr...
3       {"gameId":4224292080,"platformId":"KR","gameCr...
4       {"gameId":4222953957,"platformId":"KR","gameCr...
                              ...
1994    {"gameId":4228822566,"platformId":"KR","gameCr...
1995    {"gameId":4226414661,"platformId":"KR","gameCr...
1996    {"gameId":4224274716,"platformId":"KR","gameCr...
1997    {"gameId":4224196023,"platformId":"KR","gameCr...
1998    {"gameId":4224147107,"platformId":"KR","gameCr...
Name: data, Length: 1999, dtype: object
```

In [45]:

```
dumped_games = open('all_games_dumped.csv', 'w')
```

In [46]:

```
csv_writer = csv.writer(dumped_games)
```

In [47]:

```
header=("championid1","championid2","championid3","championid4","cha
mpionid5","championid6","championid7","championid8","championid9","c
hampionid10","t1W")
csv_writer.writerow(header)
```

Out[47]:

```
126
```

In [48]:

```
for i in range(1999):
    game = json.loads(game_data[i])
    data_list=[]
    for a in range(0,10):
        json.dumps(game['participants'])

        championids=game['participants'][a]['championId']

        data_list.append(championids)

    win=game['participants'][0]['stats']['win']
    data_list.append(win)
    csv_writer.writerow(data_list)
```

In [49]:

```
dumped_games.close()
```

## APPENDIX C

```
pip install keras
pip install tensorflow
import pandas as pd
from numpy import array
from keras.utils import to_categorical
import csv
```

In [9]:

```
df=pd.read_csv('170k_son.csv')
df2=df.drop(['t1w'],axis=1)
df2=array(df2)
```

In [10]:

```
encoded = to_categorical(df2)
data_file5 = open('onehotencoding_son_2905.csv', 'w', newline='')
csv_writer = csv.writer(data_file5)
```

In [11]:

```
list1=[]
list2=[]
list3=[]
list4=[]
list5=[]
list6=[]
list7=[]
list8=[]
list9=[]
list10=[]
```

In [12]:

```
for k in range(171856):
    list1=encoded[k][0]
    list2=encoded[k][1]
    list3=encoded[k][2]
    list4=encoded[k][3]
    list5=encoded[k][4]
    list6=-encoded[k][5]
    list7=-encoded[k][6]
    list8=-encoded[k][7]
    list9=-encoded[k][8]
    list10=-encoded[k][9]
    list1= array(list1)
    list2= array(list2)
    list3= array(list3)
    list4= array(list4)
    list5= array(list5)
    list6= array(list6)
    list7= array(list7)
    list8= array(list8)
    list9= array(list9)
    list10=array(list10)

list11=list1+list2+list3+list4+list5+list6+list7+list8+list9+list10
    list11=array(list11)
    csv_writer.writerow(list11)
deneme
```

In [13]:

```
data_file5.close()
```

## APPENDIX D

```
import csv
import pandas as pd

df1=pd.read_csv('personal_win_rate.csv')
x=(df1["Champid"])
y=(df1["Winrate"])
df1
champ1=x[0]
champ1_winr=y[0]
champ2=x[1]
champ2_winr=y[1]
champ3=x[2]
champ3_winr=y[2]
champ4=x[3]
champ4_winr=y[3]
champ5=x[4]
champ5_winr=y[4]
```

In [3]:

```
data_file = open('personal_and_champion_win_rates.csv', 'w',
newline='')
csv_writer = csv.writer(data_file)
header=("championid","winrate","championwinrate")
csv_writer.writerow(header)
```

Out[3]:

```
36
```

In [4]:

```
df=pd.read_csv('champion_win_rates.csv')
df
list1=[]
list1.append(0)
list1.append(champ1)
list1.append(champ1_winr)
list1.append(df["winrate"][champ1])

list2=[]
list2.append(1)
list2.append(champ2)
list2.append(champ2_winr)
list2.append(df["winrate"][champ2])

list3=[]
list3.append(2)
list3.append(champ3)
list3.append(champ3_winr)
list3.append(df["winrate"][champ3])

list4=[]
list4.append(3)
list4.append(champ4)
list4.append(champ4_winr)
list4.append(df["winrate"][champ4])

list5=[]
list5.append(4)
```

```
list5.append(champ5)
list5.append(champ5_winr)
list5.append(df["winrate"][champ5])

csv_writer.writerow(list1)
csv_writer.writerow(list2)
csv_writer.writerow(list3)
csv_writer.writerow(list4)
csv_writer.writerow(list5)
```

Out[5]:

```
data_file.close()
```

## APPENDIX E

```
import csv
data_file = open('picks.csv', 'w', newline='')
csv_writer = csv.writer(data_file)
header=("championid1","championid2","championid3","championid4","cha
mpionid5","championid6","championid7","championid8","championid9","c
hampionid10","152")
csv_writer.writerow(header)
```

Out[7]:

```
126
```

In [8]:

```
import pandas as pd
df2=pd.read_csv('personal_and_champion_win_rates.csv')
X2=df2["championid"]
df2
list=[]
list.append(0)
pick1=int(input("first pick:"))
list.append(pick1)
pick2=int(input("second pick:"))
list.append(pick2)
pick3=int(input("third pick:"))
list.append(pick3)
pick4=int(input("fourth pick:"))
list.append(pick4)
pick5=int(input("fifth pick:"))
list.append(pick5)
pick6=int(input("sixth pick:"))
list.append(pick6)
pick7=int(input("seventh pick:"))
list.append(pick7)
pick8=int(input("eighth pick:"))
list.append(pick8)
pick9=int(input("nineth pick:"))
list.append(pick9)
pick10=int(input("tenth pick:"))
list.append(pick10)
list.append(153)
first pick:15
second pick:111
third pick:13
fourth pick:41
fifth pick:31
sixth pick:6
seventh pick:34
eighth pick:48
nineth pick:52
tenth pick:59
```

In [10]:

```
for i in range(5):
    list[5]=X2[i]
    csv_writer.writerow(list)
```

In [11]:

```
data_file.close()
```

## APPENDIX F

In [4]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
np.random.seed(0)
from sklearn.model_selection import train_test_split

df = pd.read_excel("170k_integer_excel.xlsx")

df
X=df.drop(['win'],axis=1)
y=(df['win'])
print(df)
X_train, X_test , y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=1)
print(y_test)
log=LogisticRegression(solver='liblinear', random_state=0)
log.fit(X_train, y_train)
predictions = log.predict(X_test)
print(predictions)
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, predictions)
cnf_matrix
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu"
,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
import pandas as pd
df2=pd.read_csv('personal_and_champion_win_rates.csv')
X2=df2["championid"]
y2=(df2["winrate"])
z2=(df2["championwinrate"])
df2
from numpy import array
from keras.utils import to_categorical
import pandas as pd
list1=[]
list2=[]
list3=[]
list4=[]
```

```
list5=[]
list6=[]
list7=[]
list8=[]
list9=[]
list10=[]
list12=[]
```

In [16]:

```
df=pd.read_csv('picks.csv')
df=array(df)
df
```

Out[16]:

```
array([[ 15, 111,  13,  41,  54,   6,  34,  48,  52,  59, 153],
       [ 15, 111,  13,  41,  77,   6,  34,  48,  52,  59, 153],
       [ 15, 111,  13,  41,  50,   6,  34,  48,  52,  59, 153],
       [ 15, 111,  13,  41,  24,   6,  34,  48,  52,  59, 153],
       [ 15, 111,  13,  41,   6,   6,  34,  48,  52,  59, 153]],
      dtype=int64)
```

In [17]:

```
encoded = to_categorical(df)
encoded
```

Out[17]:

```
output=[]
for i in range(5):
    list1=encoded[i][0]
    list2=encoded[i][1]
    list3=encoded[i][2]
    list4=encoded[i][3]
    list5=encoded[i][4]
    list6=-encoded[i][5]
    list7=-encoded[i][6]
    list8=-encoded[i][7]
    list9=-encoded[i][8]
    list10=-encoded[i][9]
    list1= array(list1)
    list2= array(list2)
    list3= array(list3)
    list4= array(list4)
    list5= array(list5)
    list6= array(list6)
    list7= array(list7)
    list8= array(list8)
    list9= array(list9)
    list10=array(list10)
    list12=encoded[0][10]*0
    list12=array(list12)

list11=list1+list2+list3+list4+list5+list6+list7+list8+list9+list10+
list12
    list11=array(list11)

    input = list11.reshape((1, -1))

    output1=log.predict_proba(input)
    output.append(output1)

output
```

Out[18]:

```
[array([[0.51188239, 0.48811761]]),
 array([[0.50144648, 0.49855352]]),
 array([[0.51485487, 0.48514513]]),
 array([[0.5233279, 0.4766721]]),
 array([[0.53536292, 0.46463708]])]
```

In [20]:

```
final_list=[]
for i in range(5):
    comp_win_rate=output[i][:,1]
    personal_win_rate=y2[i]
    champion_win_rate=z2[i]

final=(0.4*comp_win_rate)+(0.4*personal_win_rate)+(0.2*champion_win_
rate)
    final_list.append(final)

final_list
```

Out[20]:

```
[array([0.53395038]),
 array([0.49792141]),
 array([0.37361805]),
 array([0.55936884]),
 array([0.5119215])]
```

In [21]:

```
max(final_list)
```

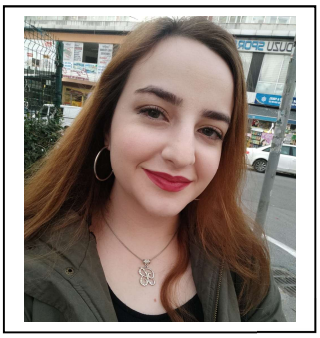
Out[21]:

```
array([0.55936884])
```

In [22]:

```
print("You must choose hero with
id:",X2[final_list.index(max(final_list))])
You must choose hero with id: 24
```

## 8. CURRICULUM VITAE



**Name Surname** **: Begüm Üstün**

**Place and Date of Birth** **: Istanbul 25.06.1998**

**E-Mail** **: ustunb16@itu.edu.tr**

**EDUCATION** **:**

- **B.Sc.** **:** 2021, Istanbul Technical University, Faculty of Management, Industrial Enginerring Department

**PROFESSIONAL EXPERIENCE AND REWARDS:**

- 2019 Production Intern at B/S/H/
- 2019 KSV Hackathlon Choice of Competitors Award
- 2019-2020 Supply Chain Project Student at B/S/H/
- 2020 Managemnt Intern at ELS Logistic
- 2020- Manufacturing Engineering Project Student at B/S/H/

**Name Surname** **: Gamze Nur Karagöz**

**Place and Date of Birth** **: Kayseri 03.03.1998**

**E-Mail** **: karagozga16@itu.edu.tr**

**EDUCATION** **:**

- **B.Sc.** **:** 2021, Istanbul Technical University, Faculty of Management, Industrial Enginerring Department

**PROFESSIONAL EXPERIENCE AND REWARDS:**

- 2019 Production Intern at Diversey
- 2019 Digital Marketing Intern at Migros Ticaret A.Ş.
- 2020 Marketing Intern at Mutlubiev/Cleanzy