# Universidad San Jorge

# Escuela de Arquitectura y Tecnología

# Grado en Diseño y Desarrollo de Videojuegos

# Proyecto Final

# Utilidad de las estadísticas públicas para determinar el equipo ganador de una partida de League of Legends

**Autor del proyecto: Carlos Molina Quílez**

**Director del proyecto: África Domingo Montes**

**Zaragoza, 26 de junio de 2023**

universidad SANJORGE
GRUPO SANVALERO

Este trabajo constituye parte de mi candidatura para la obtención del título de Graduado en Ingeniería Informática por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido.

Doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma

Fecha

26 de junio de 2023

## Dedicatoria y Agradecimiento

Este trabajo está dedicado en primer lugar a mis padres, quienes han luchado porque mi hermano y yo tuviésemos más oportunidades que ellos y me han apoyado en todas las decisiones que he tomado. A mi hermano, por haberme ayudado todas las veces que le he necesitado, y no han sido pocas.

A todos mis amigos, por apoyarme durante todos estos años sin pedir nada a cambio... bueno.

Finalmente, a mi tutora académica África Domingo, cuyo inmenso esfuerzo y dedicación han llevado este proyecto a un nivel que no habría sido capaz de alcanzar yo solo.

# Contenidos

## Resumen

League of Legends (LoL) es uno de los videojuegos MOBA (multiplayer online battle arena) más jugados del mundo. Uno de sus modos de juegos es el modo competitivo o clasificatorio, donde dos equipos de cinco jugadores con un nivel de habilidad similares se enfrentan entre ellos controlando diferentes personajes del juego que se denominan campeones. En este trabajo presento un estudio de cómo utilizar y filtrar los datos públicos de las partidas para crear un dataset y prepararlo para, utilizando distintos modelos de machine learning (aprendizaje automático), predecir el equipo ganador de una partida a partir de los campeones que forman cada uno de ellos.

## Abstract

League of Legends (LoL) is one of the most played MOBAS (multiplayer online battle arena) in the world. One of their game modes is competitive ranked play, where players with similar skill level face each other controlling characters called champions. In this paper I present a study of how the public data from these champions can be used to create a dataset and prepare it to create machine learning models that are able to predict the winning team of a game taking into account the composition (the champions that form a team) of each team.

# 1. Introducción

Los videojuegos han experimentado una evolución constante desde su creación en la década de 1950, desde los primeros juegos en consolas hasta la actualidad en donde existen juegos en línea con gráficos y experiencias de realidad virtual de alta calidad. La tecnología ha mejorado y ha permitido una mayor inmersión en el juego y una experiencia más realista, y el surgimiento de los juegos en línea ha cambiado la forma en que los jugadores interactúan entre ellos.

Los MOBA (Multiplayer Online Battle Arena) son juegos en línea donde los jugadores controlan un personaje en un mapa virtual y compiten contra otros jugadores para destruir la base enemiga. Los jugadores deben trabajar en equipo y tomar decisiones estratégicas mientras adquieren habilidades y objetos para fortalecer a su personaje. Los MOBA se han vuelto muy populares en la última década y son uno de los géneros más jugados en el mundo de los juegos en línea, además, como podemos observar en la Ilustración 1 [1], sus ingresos incrementaron un 500% entre 2010 y 2015, llegando a facturar más de 500 millones de dólares. Actualmente estos ingresos han aumentado exponencialmente, siendo el League of Legends el MOBA con mayores beneficios, en 2020 generó 1750 millones de dólares [2].

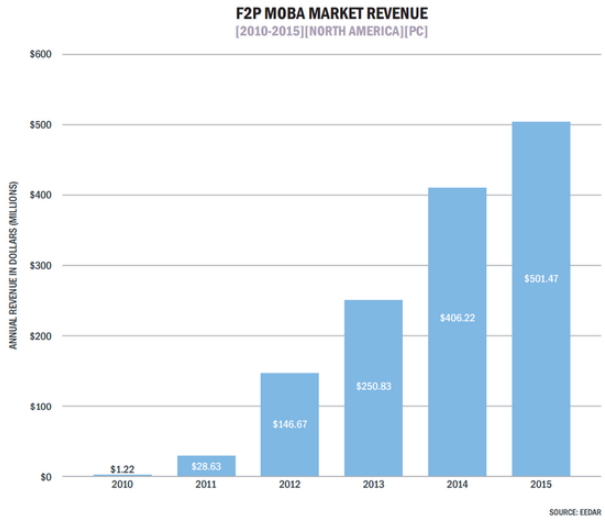


*Ilustración 1. Ingresos del mercado de los MOBAS 2010-2015*

Millones de jugadores de todo el mundo se enfrentan diariamente en juegos como League of Legends, Dota 2 o Smite, en busca de la victoria. Pero ¿qué es lo que hace que unos jugadores sean más exitosos que otros en estos juegos? ¿cuáles son las claves del éxito para ganar y subir clasificaciones? ¿Cuáles son las mejores estrategias y por qué? ¿Podemos inferir los

resultados de una partida a partir de los datos existentes?... En este trabajo se realiza un estudio en el que, a partir de datos públicos de League of Legends, se construye un dataset a partir del cual se predice el equipo ganador de una partida utilizando la composición de los dos equipos.

Esta memoria se organiza de la siguiente manera: Tras esta primera sección introductoria se describe, en la segunda sección, cómo es el videojuego que vamos a analizar y del que vamos a extraer y explotar datos: League of Legends. En la tercera sección se analizan diferentes trabajos académicos que presentan estudios similares al que se quiere abordar. En la cuarta sección se recogen los objetivos que se pretenden alcanzar con la realización de este proyecto y en la quinta sección se presenta la metodología seleccionada para realizar este proyecto y alcanzar los objetivos propuestos. En la sexta sección se recoge el estudio estadístico realizado en este trabajo que incluye tanto el análisis de las técnicas de machine learning que se emplearán, como la descripción de cómo se ha realizado la extracción de los datos para crear y procesar un dataset y la creación de los modelos de machine learning utilizando las técnicas mencionadas, así como su ejecución y la presentación de resultados. En la séptima sección discuten los resultados obtenidos en la sección anterior, comparándo nuestros resultados con los de otros trabajos previos, analizando la aportación de este trabajo a la comunidad y la desviación que hemos observado respecto a la metodología. En la octava sección se presenta un estudio económico del proyecto en el que además de estimar los costes de su realización, se incluye una propuesta de su posible explotación. Finalmente, en la novena y última sección, hablaré de las conclusiones del proyecto.

## 2. League of Legends (LoL)

League of Legends es uno de los MOBA más populares y jugados en el mundo. Desarrollado por Riot Games, el juego se lanzó en 2009 y ha ganado una gran base de seguidores gracias a su accesibilidad (es gratuito), combinación de estrategia, habilidad y juego en equipo. Como en otros MOBAS, dos equipos, en este caso de cinco jugadores, compiten por destruir la base enemiga.


*Ilustración 2. Logo de League of Legends.*

League of Legends cuenta con una amplia variedad de campeones, que son los distintos personajes que cada jugador puede seleccionar para jugar una partida, cada uno de ellos cuenta con habilidades únicas y un estilo de juego diferente, lo que lo hace atractivo para una gran variedad de jugadores.

Las diferencias entre los campeones son varias, en primer lugar, sus habilidades. No existe ningún campeón que tenga la misma habilidad que otro, sí que hay similitudes entre ellas. Como aquellas que cumplen la misma función, por ejemplo, existen varias habilidades que ralentizan el avance del campeón enemigo, son similares, pero tanto sus nombres como sus efectos visuales son distintos.

La otra diferencia son las estadísticas de los campeones. Los campeones poseen características como vida, daño, armadura, daño mágico, rango de ataque, etc. Estas son distintas dependiendo del tipo de campeón. Por ejemplo, un campeón de tipo "tanque" tendrá más vida y armadura que uno de tipo "asesino", y este tendrá más daño que uno de tipo "tirador" que tendrá más rango.

Una característica común de todos los campeones es la experiencia y los niveles, los campeones empiezan la partida a nivel 1. A lo largo de la partida todos los campeones ganan experiencia de distintas fuentes, y cuando esta llega a una cantidad específica, el campeón sube de nivel, mejorando sus características y habilidades, hasta un máximo de nivel 18, a partir del nivel 18 ese campeón no ganará experiencia.

Además, el juego tiene una comunidad y una escena competitiva activa, con varias ligas tanto como regionales, como continentales, y cada año hay un evento conocido como "Worlds" donde equipos de todo el mundo compiten por el título de campeón del mundo.

Antes de comenzar una partida, los jugadores de los dos equipos pasan por una fase de "draft" donde cada equipo escoge los campeones que van a querer jugar en la partida, se divide en dos fases:

- Fase the "baneo": Durante esta fase, cada jugador puede bloquear un campeón, impidiendo que se puedan jugar en la partida. Esto se suele hacer para prevenir que el equipo enemigo escoja un personaje que el jugador cree que les podría dar una ventaja a su oponente en la partida. En total se eliminan 10 campeones, 5 cada equipo, y cualquier jugador puede eliminar el campeón escogido sin ninguna restricción de orden.
- Fase the "pickeo": Durante esta fase, cada equipo se turna para seleccionar campeones siguiendo un orden fijo. El primer jugador del primer equipo (denominado equipo azul) elegirá el personaje que quiere jugar, después, los dos primeros jugadores del equipo enemigo (equipo rojo) elegirán sus respectivos campeones. Luego será el turno de los siguientes dos jugadores del otro equipo (azul), así hasta llegar al último jugador del equipo rojo, que escogerá su campeón y la partida comenzará.

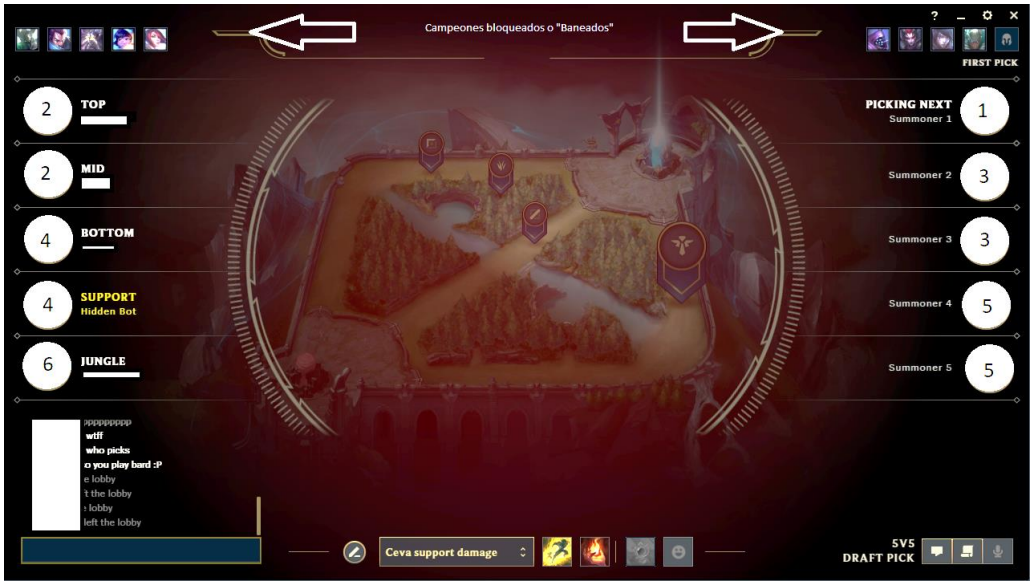


*Ilustración 3 Fase de 'draft'. Elaboración propia*

Una partida de League of Legends se juega en un mapa llamado Grieta del Invocador, está dividido en tres líneas (Top lane, Mid lane y Bot lane) y cada línea está separada por vegetación a la que se le denomina jungla. Los jugadores se dividen de la siguiente manera: un jugador se centra en la línea de arriba (Top), otro se centra en la línea del medio (Mid), dos jugadores van a la línea de abajo (Bot) y el último jugador vaga por la jungla.

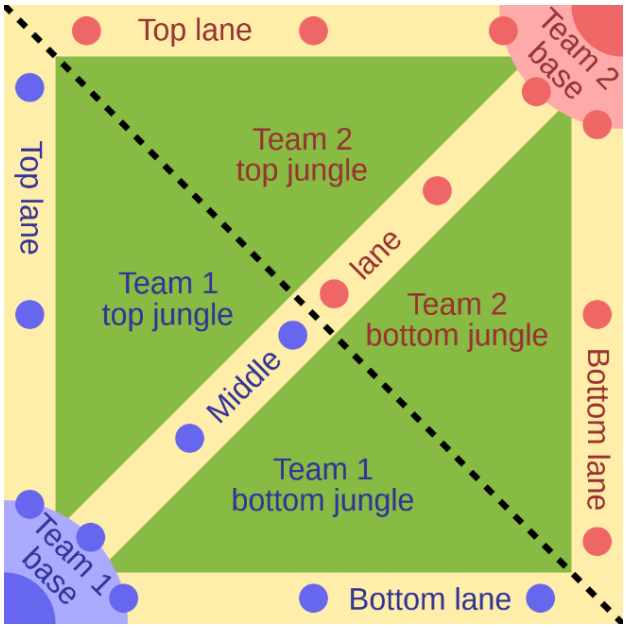


*Ilustración 4. Mapa explicativo de League of Legends.*

Los puntos rojos y azules representan torretas de cada equipo, es necesario romper todas las torretas de al menos una línea para dejar la base enemiga al descubierto.

Cada uno de los 5 jugadores tiene un rol único que pueden ser:

- Top laner: Es el jugador que lucha en la línea de arriba en el mapa, los campeones comúnmente jugados en esta línea son aquellos con mucha vida y defensa para absorber daño por su equipo (tanques o "bruisers").
- Jungler: Es el jugador que se mueve por la jungla, matando monstruos que habitan en ella y brindando apoyo a aquellas líneas que lo necesiten, hay muchos campeones distintos que pueden ser jugados en este rol, desde tanques con vida y defensa hasta asesinos con mucho daño en poco tiempo.
- Mid laner: Este jugador es responsable de la línea en el centro del mapa, y se encarga de controlarla, suelen jugarse magos o asesinos con gran potencial de daño.
- ADC (Attack Damage Carry): Este jugador es responsable de hacer el máximo daño posible al enemigo y suelen jugarse tiradores con ataques de larga distancia. Se centra en la calle inferior (Bot) junto al último rol.

- Support (Soporte): El rol de support, como su nombre indica, se encarga de ayudar a otros miembros del equipo, ya sea curando o haciendo que el equipo enemigo tenga problemas para alcanzar al suyo.

Es imprescindible que cada jugador entienda su rol y lo cumpla para que su equipo salga victorioso.

Al comenzar una partida, cada uno de los equipos aparece en sus respectivas bases, aparecen también unas entidades no jugables llamadas súbditos (los cuáles son NPC, o non-player character), cuya función es avanzar por sus respectivas líneas. Si un campeón mata a un súbdito del equipo enemigo recibe oro y experiencia. Al llegar a una cantidad específica de experiencia, los campeones suben de nivel, mejorando sus habilidades y características de ataque, defensa, etc. hasta un máximo de 18 niveles.

Matar a un campeón enemigo, es decir derrotar a otro jugador, otorgará al que lo haya asesinado de una cantidad de oro y experiencia muy superior al que se recibe por matar a un súbdito. Los monstruos que habitan en la jungla también otorgan experiencia y oro al morir. Con el oro recibido se pueden comprar objetos, hasta un máximo de seis, y estos fortalecen al campeón que los haya comprado. Existen también otros objetivos como torretas o monstruos legendarios que requieren del apoyo de varios integrantes del equipo para acabar con ellos y otorgan varias mejoras a todo el equipo.

La partida acaba cuando uno de los equipos destruye la base enemiga (llamada Nexo).

Los jugadores deben trabajar juntos, comunicarse y tomar decisiones estratégicas para conseguir llegar a la victoria. El resultado de una partida puede ser influenciado por muchos factores, incluyendo la composición un equipo (los campeones), el rendimiento personal, y el uso de trabajo en equipo y estrategia.[3]

Existen dos modos de juego, el modo normal y el modo clasificatorio. En el modo clasificatorio, al acabar una partida los jugadores ganan o pierden "puntos de liga" dependiendo del resultado de la partida. Estos puntos de liga influyen en su clasificación y rango en el sistema de clasificación de League of Legends. En este sistema existen distintas ligas y divisiones, ordenadas de: IRON, SILVER, GOLD, PLATINUM, DIAMOND, MASTER, GRANDMASTER y CHALLENGER. Excepto las tres últimas ligas, todas tienen cuatro divisiones, siendo IV la más baja y I la más alta.

Los jugadores son asignados a una liga y división al completar su primera partida clasificatoria, y a partir de ahí deben ganar partidas para ganar puntos y subir divisiones y ligas. Por ejemplo, un jugador en Diamante III, el cual tiene 90 puntos de liga, si gana la siguiente partida y

obtiene 20 puntos, subirá de Diamante III 90 pl (puntos de liga) a Diamante II 10 pl, ya que es necesario superar 100 puntos para subir a la siguiente división. Si un jugador está en la división I y alcanza 100 puntos, deberá superar una promoción para poder ascender a la siguiente liga. Estas promociones consisten en jugar una serie de tres partidas y ganar dos de ellas para ascender. Por ejemplo, si un jugador en Silver I 100 pl gana dos partidas de su promoción, ascenderá a Gold IV 0 pl, y si pierde dos partidas se quedará en Silver I 75 pl. A partir de Master las cosas son distintas, en lugar de tener que llegar a 100 pl para ascender, no hay un límite de puntos, y para ascender de liga es necesario llegar a un número de puntos que varía dependiendo del número de jugadores en la liga actual, así que es posible que se necesiten 300 puntos o 700 puntos. Lo mismo ocurre para los jugadores de Grandmaster que quieren ascender a Challenger.

## 2.1. Estadísticas y datos públicos

Las estadísticas de los campeones o jugadores de League of Legends son públicas y se pueden recopilar y analizar de muchas maneras. Hay varios sitios web que ofrecen estadísticas en tiempo real y datos históricos sobre jugadores individuales, equipos y campeones. Estas estadísticas incluyen información sobre victorias y derrotas, porcentaje de asesinatos, asistencias, KDA (kills, deaths, assists) y mucho más. Estos datos se pueden utilizar para hacer análisis de rendimiento individual y de equipo, identificar fortalezas y debilidades, y tomar decisiones informadas sobre cómo mejorar al juego. Además, los datos sobre los campeones y sus estadísticas se pueden utilizar para elegir un equipo y estrategia ganadora en una partida. En la ilustración 5 podemos ver una captura de la página web U.GG [4], que utiliza estos datos públicos para crear una lista de los mejores campeones.

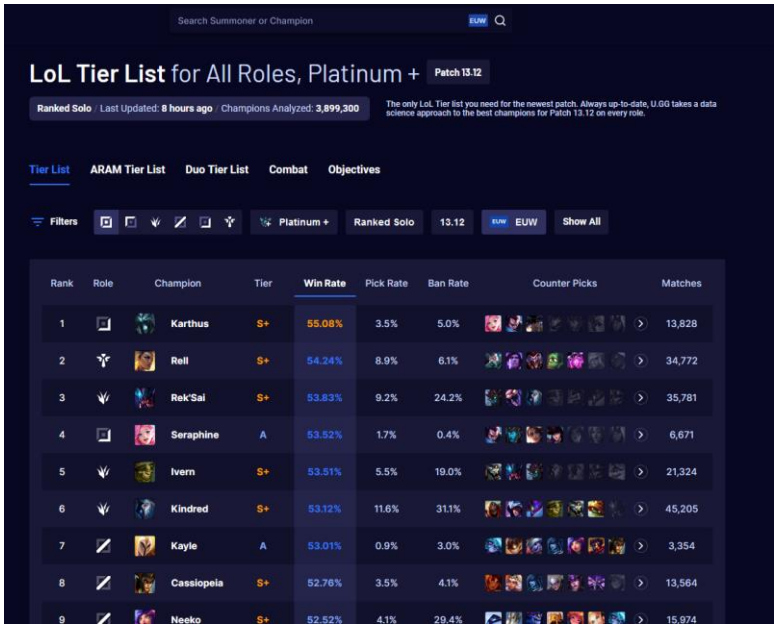


*Ilustración 5. Captura del sitio web U.GG.*

Al ser un juego multijugador en línea, ~~el~~ League of Legends recibe cambios ya sea en campeones (bajarle la vida a un personaje, subirle el daño de ataque a otro, etc), objetos (son comprados por los jugadores con oro y mejoran a su personaje) o al juego en sí (por ejemplo, modificando los objetivos secundarios). Estos cambios ocurren cada varias semanas y son conocidos como "parches".

Estos "parches" hacen que las estadísticas varíen respecto a los parches anteriores, y pueden hacer que campeones y/o objetos que anteriormente eran muy utilizados sean olvidados por completo. Los parches hacen que este "meta" esté en constante cambio, lo cual hace que el juego sea muy diverso.

Las estadísticas que varían a medida que los parches ocurren son:

- "WinRate": El "WinRate" es el porcentaje de victorias de un campeón (Número de victorias del campeón / Número de partidas jugadas del campeón).
- "PickRate": El "PickRate" es el porcentaje de partidas en las que el campeón se ha jugado (Número de partidas jugadas del campeón / Número de partidas totales).
- "BanRate": El "BanRate" es el porcentaje de partidas en las que el campeón ha sido bloqueado (Número de partidas en las que el campeón fue bloqueado / Número de partidas totales).

## 3. Estado del arte

Los MOBAS (Multiplayer Online Battle Arena) son uno de los géneros de videojuegos más populares y competitivos. Millones de jugadores de todo el mundo se enfrentan diariamente en juegos como League of Legends, Dota 2 o Smite, en busca de la victoria. Pero, ¿qué es lo que hace que unos jugadores sean más exitosos que otros en estos juegos? ¿Cuáles son las claves del éxito para ganar y subir clasificaciones? ¿Cuáles son las mejores estrategias y por qué? Desde hace años, numerosos estudios han tratado de responder a estas preguntas, analizando las estadísticas, comportamientos y factores que influyen en el éxito en los MOBAS. Tras estudiar varios trabajos, los hemos divido por tipos de análisis que realizan:

Un tipo de análisis que podemos observar en [5], [6], [7] y [8] es el de los jugadores, y cómo sus características como tipo de personalidad, rendimiento, etc, afectan a las probabilidades de ganar de un equipo.

Otro tipo de análisis es el de los personajes jugables de los distintos juegos [9], [10], [11], [12], [13], [14], [15], [16], y de qué manera afectan sus estadísticas a las probabilidades de ganar. No todos los análisis de los personajes se centran en las probabilidades de victoria, por ejemplo [13] hace un análisis de la popularidad de los personajes y cómo varía entre los distintos rangos de clasificación.

Varios trabajos crean también distintos modelos de "teambuilding" o creación de equipos [9], [11], [12], [13], [16], [17] y [18] donde establecen un sistema de elección de personajes jugables que aumente la probabilidad de victoria del equipo y/o de predicción de equipo ganador dependiendo de varias condiciones.

Dado el origen y los objetivos de nuestro trabajo, nos enfocamos en el análisis de aquellos trabajos que tratan el "teambuilding" ya que es especialmente relevante para nuestro campo de estudio y los objetivos que queremos lograr. Los trabajos comentados anteriormente abordan la cuestión del "teambuilding" desde distintas perspectivas, y a continuación hay una explicación de cada uno de ellos.

Agarwala y Pearce [9] utilizaron los datos de 40.000 partidas para estudiar cómo la composición de un equipo de Dota2, un moba muy similar al League of Legends, afecta a sus datos de éxito en una partida. Utilizaron regresión logísitica, k-means algoritm y PCA para el tratamiento de los datos. Consiguieron dos modelos predictivos con un 57% y 62% de precisión respectivamente.

No consiguieron un modelo en el que clasificaran los datos por divisiones y su agrupación de datos no respondía claramente a estilos de juego.

Semenov y Romov [11] utilizaron datos de cinco millones de partidas para crear varios modelos de predicción de victoria de un equipo usando algoritmos de aprendizaje automático (machine learning) incluyendo Regresión Logística, Árboles de Decisión, Random Forest y Gradient Boosting. El modelo final con mayor precisión que desarrollaron es capaz de predecir el equipo ganador analizando la composición de los personajes de cada equipo con una precisión del 70%.

Zhengxing Chen [12] implementó el algoritmo de búsqueda heurístico Monte Carlo Tree Search para crear un modelo de selección de los campeones óptimos para un equipo que aumente las probabilidades de victoria utilizando datos de 3 millones de partidas con 111 campeones diferentes con un 53.77% de precisión.

Daniel Gourdeau y Louis Archambault [13] analizaron datos de 7630 partidas, realizaron una división de campeones en "clusters" o grupos dependiendo de su función mediante el algoritmo "Ward Hierarchical Clustering Algorithm" para crear modelo de selección del mejor campeón para un equipo que más aumente sus probabilidades de victoria según su composición con un 84.3% de precisión para el juego Heroes of the Storm y un 68.3% para Dota 2.

Tiffany D. Do, Seong Ioi Wang [16] utilizaron datos de 5000 partidas y distintos tipos de machine learning para crear un modelo que fuese capaz de determinar cuál de dos equipos tiene mayor probabilidad de victoria teniendo en cuenta la habilidad individual de los jugadores con el campeón que estén jugando, usando estadísticas como el "winrate" con ese campeón consiguieron que su modelo alcanzase un 75.1% de precisión.

Iuliia Porokhnenko, Petr Polezhaevy Alexander Shukhman [17] entrenaron varios modelos de machine learning utilizando datos de 56691 partidas del videojuego Dota 2 como Gradient Boosting Classification (GBC), Random Forest Classification (RFC), Linear Regresion (LR), Support Vector Classification (SVC), etc. y concluyeron que el mejor modelo que predijiese el equipo ganador entre dos dependiendo de los campeones elegidos por cada equipo es el de SVC y LR con un 77.39% de precisión.

Conley y Perry [18] se centraron en la creación de un sistema de recomendación de héroes para el juego Dota 2 utilizaron datos de partidas públicas y algoritmos de machine learning

como Regresión logística o K-Nearest Neighbors. El modelo final era capaz de recomendar varios personajes que aumentaban la probabilidad de victoria del equipo con una precisión del 68%.

En la Tabla 1 se recogen los datos más importantes de los trabajos más relevantes para este proyecto, como las herramientas que utilizan para sus estudios, los datos que utilizan, sobre que juego hacen el análisis, si clasifican los datos por categoría y los resultados que han obtenido.  A partir de su análisis podremos acotar mejor el alcance del análisis estadístico a realizar en el proyecto.

| Referencias | MOBA title | Herramienta estadística | Datos que utilizan/variables | Clasificación por categoría de jugador | Resultados |
|---|---|---|---|---|---|
| [9]<br>Atish Agarwala<br>Michael Pearce | Dota 2 | Regresión lineal.<br>Normalización de estadísticas de los héroes.<br>K-means algorithm (cluster de datos).<br>PCA | Datos de 40.00 partidas:<br>Utilizan la composición de héros de cada equipo. | No clasifica, utiliza todos los datos de todas las categorías | Cómo la composición de un equipo afecta a su tasa de éxito en una partida.<br>Predicción del 57% de precision para 7 componentes de PCA y del 62% para el modelo completo. |
| [10]<br>Choong-Soo Lee<br>Ivan Ramler | LoL | Calculan el uso de campeones mediante una fórmula que han creado<br>$U_{ctd} = \frac{M_{ctd}}{\sum_{c=1}^{C} W_{ctd}/5} \cdot 100$<br>Donde Mctd son las partidas donde el campeón c en la clasificacion t en el día d y el sumatorio de Wctd/5 es el número de victorias. C representa el el número total de campeones jugados en el día d. | Datos de campeones como popularidad, porcentaje de victorias, ratio de asesinatos-muertes-asistencias, bloqueos de campeones, objetos y clasificación de jugadores, extraído todo de la página web LoL DB Gameguyz. | Utilizan los datos de las distintas categorías y de partidas normales. | Cada semana varios campeones son gratis y se pueden jugar sin ser comprados, esto hace que aumente el uso de esos campeones en partidas normales. Siendo los campeones más caros los que aumentan más su uso durante la rotación gratuita. |
| [11]<br>Aleksandr Semenov<br>Peter Romov<br>Sergey Korolev<br>Daniil Yashkov | Dota 2 | Regresión logística.<br>Naive Bayes.<br>Factorization Machines<br>Gradient Boosting of Decision Trees<br>Modelo que predice la mejor composición de un equipo para vencer a otro | Datos de 5 millones de partidas de distintos modos de juego y distinto nivel de habilidad. | Dividen los datos en Habilidad Normal, Alta y Muy Alta | Aumentaron la precisión de su modelo respecto a otros modelos ya existentes.<br>0.7 XGBoost<br>0.68 LogReg<br>0.68 Bayes |
| [6]<br>Ilya Makarov<br>Dmitry Savostyanov<br>Boris Litvyakov<br>Dmitry I. Ignatov | Dota 2 | Regresión logística.<br>Árboles de decisión.<br>Diseño de un modelo que prediga el equipo ganador evaluando el rendimiento de cada jugador. | Datos de partidas profesionales como cantidad de oro y experiencia obtenida por jugador, asesinatos, compras de objetos. De cada jugador, no campeón. | Solo utilizan partidas profesionales | El modelo ya existente "TrueSkill" solo obtuvo 0.72 de precisión, y el modelo creado que combina también "TrueSkill" aumentó a 0.92. |
| [12]<br>Zhengxing Chen<br>Truong-Huy Nguyen<br>Yuyu Xu<br>Christopher Amat | Dota 2 | Monte Carlo Tree Search.<br>Crear un modelo que seleccione la mejor composición de campeones que tenga mayor probabilidad de victoria. | Datos de 3 millones de partidas con 111 campeones diferentes. | Nivel de habilidad medio | EL Monte Carlo Tree Search se puede utilizar para diseñar un sistema de selección de campeones sin tener en cuenta la habilidad individual de los jugadores |

| | | | | | |
|---|---|---|---|---|---|
| [13]<br>Daniel Gourdeau<br>Louis Archambault | HotS, Dota 2 | Red neuronal (Neural Network), en particular "Discrete generative adversial network" o GAN.<br>Modelo de selección de campeones, dan más o menos peso a los campeones dependiendo de sus roles.<br>Ward hierarchical clustering algorithm | Datos de 7630 partidas, unos 110074 ejemplos de elecciones y bloqueos de campeones. | Partidas profesionales | Se puede hacer uso de una "Discrete generative adversial network" para crear un modelo de selección de personajes. |
| [15]<br>Alexander Dallmann<br>Johannes Kohlmann<br>Daniel Zoller<br>Andreas Hotho | Dota 2 | Red neuronal, en particular SIR model.<br>Regresión logística.<br>Modelo neuronal<br>"Multilayer-Perceptron"<br>GRU (Recurrent Neural Network)<br>NARM (Neural Attentive Recommendation Model) | No especifica el número de partidas.<br>Datos de los objetos comprados en las partidas, así como la composición de los equipos. | Partidas clasificatorias, pero no especifica nivel de habilidad | El modelo GRU alcanzó un 0.78 de precisión.<br>Cuanta precisión tiene la recomendación de objetos |
| [16]<br>Tiffany D. Do<br>Seong Ioi Wang<br>Dylan S. Yu<br>Matthew McMillian<br>Ryan McMahan | LoL | Support vector classifier.<br>K-Nearest neighbors.<br>Random Forest Trees.<br>Gradient Boosting.<br>Deep Neural Networks (Keras).<br>Crear un modelo que prediga el equipo ganador teniendo en cuenta la experiencia de los jugadores con su personaje. | Utilizan datos de 5000 partidas y de los jugadores de cada partida (máximo 50000 jugadores si no se repite ninguno).<br>De los jugadores extraen el porcentaje de victoria del campeón que juegan y sus puntos de maestría (que es un número que aumenta a más experiencia tenga el jugador con ese campeón) | No especifica clasificación. | Consiguieron que su modelo alcanzase un 75.1% de precisión.<br><br>Cómo el conocimiento y experiencia del campeón escogido por cada jugador influye en la probabilidad de victoria. |
| [18]<br>Kevin Conley<br>Daniel Perry | Dota 2 | Regresión logística.<br>K-Nearest Neighbors.<br>Modelo que predice la probabilidad de victoria dependiendo de la composición. | Datos de 56691 partidas extraídos de la API oficial. | El nivel de habilidad de las partidas era "muy alto".<br>Incluyen la composición de campeones de cada equipo. | Viendo la composición de un equipo se puede usar el modelo para recomendar campeones contra ese equipo.<br><br>El modelo de KNN alcanzó un 67.43% de precisión. |
| [17]<br>Iuliia Porokhnenko<br>Petr Polezhaev<br>Alexander Shukhman | Dota 2 | Gradient Boosting Classification (GBC).<br>Random Forest Classification (RFC).<br>XGBoost Classification (XGBC).<br>Regresión logística.<br>SVC.<br>CBC.<br>Red Neuronal. | Datos de 56690 partidas obtenidos a través de "OpenDota API".<br>Dividiendo las partidas en grupos por modo de juego. | Se escogieron partidas con un nivel de habilidad alto. | Crearon varios modelos que calculan el equipo ganador en una partida dependiendo de la composición.<br>El mejor es el de Regresión Lineal y SVC con un 77.39% de precisión para los dos modelos, con el de Regresión siendo más rápido que el SVC. |

*Tabla 1. Trabajos analizados.*

Los objetivos que se han establecido para este trabajo fin de grado en la fase de propuesta de proyecto (ver propuesta de proyecto en Anexo I), son:

- Describir el videojuego donde se aplicará el análisis estadístico y las bases de datos de dónde se obtendrán los datos a tratar.
- Conocer el estado del arte en las estadísticas empleadas en el análisis de videojuegos.
- Realizar un estudio estadístico a partir del cual podamos concluir si a partir de los datos públicos de un videojuego podemos extraer información útil sobre el juego.
- Evaluar los beneficios, inconvenientes y limitaciones que supone la utilización de estadística y datos públicos para obtener información útil sobre un videojuego.
- Obtener experiencia en el manejo de grandes cantidades de datos a través de herramientas estadísticas en un contexto real.

Tras el análisis de los datos disponibles del juego y de los análisis hechos previamente por otros investigadores, nos planteamos los siguientes objetivos concretos para acotar el análisis estadístico presentado en este trabajo:

- Seleccionar aquellos métodos que nos permitan realizar una predicción de cuál será el equipo ganador dada la colección de campeones seleccionados.
- Crear un data set apropiado para su análisis a partir de los datos públicos del videojuego Leage of Legends.
- Aplicar al menos tres de los métodos para realizar predicciones que se utilizan en los trabajos previos analizados y comparar los resultados obtenidos con los publicados por otros autores.

Propongo también un cambio de título de "Utilidad de las estadísticas públicas para determinar los mejores personajes de un MOBA" a "Utilidad de las estadísticas públicas para determinar el equipo ganador de una partida de League of Legends" para que el título refleje mejor el análisis estadístico que se realiza en el trabajo.

# 4. Metodología

Una metodología en el contexto de desarrollo software es un conjunto de prácticas, técnicas y herramientas que se utilizan para planificar, ejecutar y controlar un proyecto de manera efectiva [19], [20]. Es necesario definir una metodología porque proporciona un marco de trabajo claro y coherente para el equipo de proyecto y ayuda a garantizar que el proyecto se entregue a tiempo, dentro del presupuesto y con calidad [21]. Elegir la metodología más adecuada depende de varios factores, como el tamaño y la complejidad del proyecto, el tamaño y los integrantes del equipo, el tipo de industria y los requisitos del cliente [21] [22]. Existen muchas metodologías que se utilizan en la gestión de proyectos, pero todas se pueden clasificar en dos tipos, tradicionales o ágiles [23]:

- Metodologías tradicionales: estas metodologías se basan en una planificación exhaustiva y detallada al comienzo del proyecto. Definen con precisión los objetivos y requisitos del proyecto antes de empezar a trabajar. Algunas metodologías tradicionales incluyen la metodología de cascada o Waterfall y la metodología de desarrollo en V.

- Metodologías ágiles: estas metodologías se basan en un enfoque más flexible y adaptativo de la gestión de proyectos. En lugar de planificar todo el proyecto de antemano, se dividen en ciclos de variada duración (de una a varias semanas) llamados "sprints". En estos "sprints" se realizan distintas tareas que se definen antes de comenzar el "sprint". Algunas metodologías ágiles incluyen Scrum, Kanban y Extreme Programming (XP).

Dentro de las metodologías tradicionales, Waterfall [23], [24] se basa en un enfoque secuencial y lineal. Este modelo sigue una secuencia de fases o etapas, donde cada etapa se termina antes de comenzar la siguiente. Esta metodología es adecuada para proyectos bien definidos, con objetivos y alcances claramente establecidos, con plazos definidos y sin cambios significativos en los requisitos a lo largo del proyecto. La metodología Waterfall es rígida y no es adecuada para proyectos en los que la incertidumbre y los cambios son comunes.

La otra metodología tradicional, metodología de desarrollo en V, se basa en la idea de visualizar el flujo de trabajo del proyecto en forma de una letra "V". En la rama izquierda, se encuentran las actividades relacionadas con el diseño del proyecto, mientras que en la rama derecha se encuentran las actividades relacionadas con las pruebas y la verificación del proyecto. Al igual que la metodología Waterfall, este tipo de metodología es ideal para proyectos con los objetivos y requerimientos definidos antes de comenzar el proyecto, y sin cambios significativos a lo largo del desarrollo del proyecto.

Las características principales de las metodologías ágiles nombradas anteriormente serían:

- Scrum[25]: Scrum se basa en una serie de roles, eventos, artefactos y reglas que ayudan a los equipos a colaborar y a trabajar juntos de manera efectiva. Scrum se divide en sprints, que duran entre una y cuatro semanas, y cada sprint termina con una revisión y una retrospectiva. Esta metodología es adecuada para proyectos de software donde los requisitos cambian constantemente y los objetivos no están completamente definidos.

- Kanban[26]: Kanban se centra en la visualización y limitación del trabajo en progreso (WIP). Kanban se basa en la idea de que el trabajo debe ser visible y que el equipo debe trabajar en una cantidad limitada de tareas a la vez para maximizar la eficiencia y minimizar los retrasos. Kanban se divide en columnas que representan las etapas del flujo de trabajo, y utiliza tarjetas o notas adhesivas para visualizar el trabajo en cada etapa. Esta metodología es adecuada para proyectos con requisitos estables y donde la eficiencia y la capacidad de respuesta son importantes.

- XP[27]: Extreme Programming (XP) se centra en la comunicación y colaboración entre el equipo de desarrollo y el cliente, y se basa en cinco valores: comunicación, simplicidad, retroalimentación, valentía y respeto. XP incluye prácticas como pruebas automatizadas, integración continua y programación en parejas. Esta metodología es adecuada para proyectos de software que requieren una alta calidad, donde los requisitos cambian con frecuencia y la comunicación con el cliente es importante.[26]

## 4.1. Análisis del proyecto

En este proyecto, el equipo está formado por una sola persona, el alumno, el cliente es la tutora del proyecto. Se realizará durante el curso escolar, lo cual significa que habrá períodos donde se tengan que atrasar tareas a realizar debido a exámenes u otros trabajos de asignaturas que el alumno esté cursando. No habrá reuniones diarias ni presupuesto a considerar. El objetivo principal del proyecto es hacer un análisis estadístico de los personajes jugables del videojuego League of Legends para obtener el que sería el mejor equipo. Conforme se avance en el desarrollo del proyecto, será necesario refinar y ajustar el objetivo, puesto que el grado de incertidumbre inicial sobre los datos que se podrán utilizar, la información que nos pueden ofrecer, o las herramientas que podemos utilizar para conseguirlo es alto.

## 4.2. Metodología seleccionada

Teniendo en cuenta las características del proyecto, considero metodologías que incluyan criterios como la flexibilidad y la adaptabilidad entre otros, por lo tanto, debemos descartar metodologías tradicionales como waterfall y utilizar una metodología ágil.

Dentro de las metodologías ágiles, las características principales se pueden dividir en tres grupos, roles, eventos y artefactos. Los roles son las funciones que los distintos integrantes del equipo desempeñan (cliente, desarrolladores, mánager, etc.), los eventos son sucesos típicamente iterativos y regulares, diseñados para proporcionar estructura y ritmo al proceso de desarrollo (Sprints, reuniones diarias o "daylies", etc.), por último, los artefactos son aquellas piezas de trabajo o documentación que se utilizan para organizar y rastrear el progreso (backlog, lista de tareas, etc.).

De las metodologías ágiles ya mencionadas (Scrum, Kanban y XP), presento aquí un análisis de sus principales diferencias y similitudes en cuanto a roles, eventos y artefactos:
Scrum [28]:

- Roles: Scrum define tres roles principales: el Product Owner (representa la voz del cliente), el Scrum Master (líder del equipo y encargado de eliminar cualquier obstáculo) y el Equipo de Desarrollo (se encarga de realizar las tareas previstas).
- Eventos: Scrum define varios eventos estructurados, como la Reunión de Planificación del Sprint (se definen las tareas que formarán parte de cada sprint), la Reunión Diaria del Scrum (se informa sobre lo que se hizo el día anterior, lo que se hará ahora y los obstáculos que han ido surgiendo), la Revisión del Sprint (valoración que se realiza al final de cada sprint) y la Retrospectiva de sprint (conclusiones, mejoras y recomendaciones para el siguiente sprint).
- Artefactos: Scrum define tres artefactos principales: la Pila del Producto (documento central con todos los elementos necesarios para la ejecución de este), la Pila del Sprint (objetivos y/o tareas pendientes por hacer o cumplir en el sprint, sino se cumplen se trasladan al siguiente) y el Incremento (manera de medir el progreso. Para Scrum, es esencial que cada iteración tenga un incremento).

Kanban [26]:

- Roles: Kanban no define roles específicos, aunque se espera que los miembros del equipo colaboren y se involucren activamente en el proceso de desarrollo.
- Eventos: Kanban no define eventos específicos, aunque es común que el equipo tenga reuniones diarias para revisar el estado del tablero Kanban y asegurar que hay una "mejora continua" (mejora continua del proceso de desarrollo a través de la retroalimentación continua y la optimización del flujo de trabajo).

- Artefactos: Kanban define dos artefactos principales: el Tablero Kanban (herramienta visual utilizada para administrar y visualizar el flujo de trabajo) y la Lista de Trabajo (herramienta para visualizar y gestionar las tareas individuales que forman parte del proceso de producción o desarrollo).

XP [27]:

- Roles: XP define cinco roles principales: el Cliente (define las funcionalidades y requisitos del software), el Programador (implementa las funcionalidades definidas por el cliente), el Tester (asegura la calidad del software a través de pruebas continuas), el Coach (asesora al equipo en la implementación de prácticas y valores de XP) y el Tracker (gestiona la información del proyecto y la comunicación entre los miembros del equipo).
- Eventos: XP define varios eventos, como el Juego de Planificación (define y prioriza las funcionalidades para la iteración actual), las Reuniones Diarias (actualiza y sincroniza el equipo sobre el progreso y obstáculos) y las Revisiones de la Iteración (presentación del trabajo completado al cliente para su retroalimentación).
- Artefactos: XP define varios artefactos, como las Historias de Usuario (descripciones de funcionalidades desde la perspectiva del usuario), las Tarjetas de Tareas (lista de tareas necesarias para implementar cada Historia de Usuario) y los Valores (principios que guían el trabajo del equipo, incluyendo Comunicación, Simplicidad y Feedback).

Después de revisar las características de estas tres metodologías, he decidido que no quiero limitarme a elegir una sola metodología. En su lugar, voy a seleccionar aquellas características que considero más útiles para mi proyecto. Al tomar lo mejor de cada metodología y adaptarlas a nuestro equipo y proyecto, espero crear un proceso de desarrollo ágil y eficiente que nos permita lograr nuestros objetivos de manera efectiva.

De Kanban voy a escoger la opción no tener roles definidos, ya que el equipo es muy pequeño. Sin embargo, me gustaría incorporar los sprints de 2-3 semanas de Scrum, ya que esto permitiría una mejor planificación y organización del trabajo, aunque sin tener reuniones diarias. De XP voy a elegir las Tarjetas de Tareas que se realizarán al principio de cada sprint, sin necesidad de crear Historias de Usuario, simplemente una lista de tareas para tener orden en el proceso de desarrollo.

Finalmente, me gustaría enfocarme en la mejora continua de Kanban para asegurarme de que el proceso de desarrollo esté siempre mejorando y ajustándose a nuestras necesidades específicas. Para ello, en cada sprint mejoraremos la memoria del proyecto volcando todo el trabajo realizado durante el sprint dentro de esta y recogiendo en ese documento todos los

aspectos de mejora que se detectan al final de cada iteración por medio de comentarios y anotaciones en el documento.

Después de cada una de las reuniones para la siguiente reunión realizábamos un listado de las cosas que teníamos pendientes para la siguiente a modo de acta y programábamos la siguiente reunión. Todas las reuniones se planificaban dentro de Microsoft TEAMS en una fecha con un listado de tareas pendientes. En las siguientes ilustraciones (ilustraciones 5, 6 y 7) aparecen ejemplos de la información asociada a cada reunión, nuestras Tarjetas de Tareas.

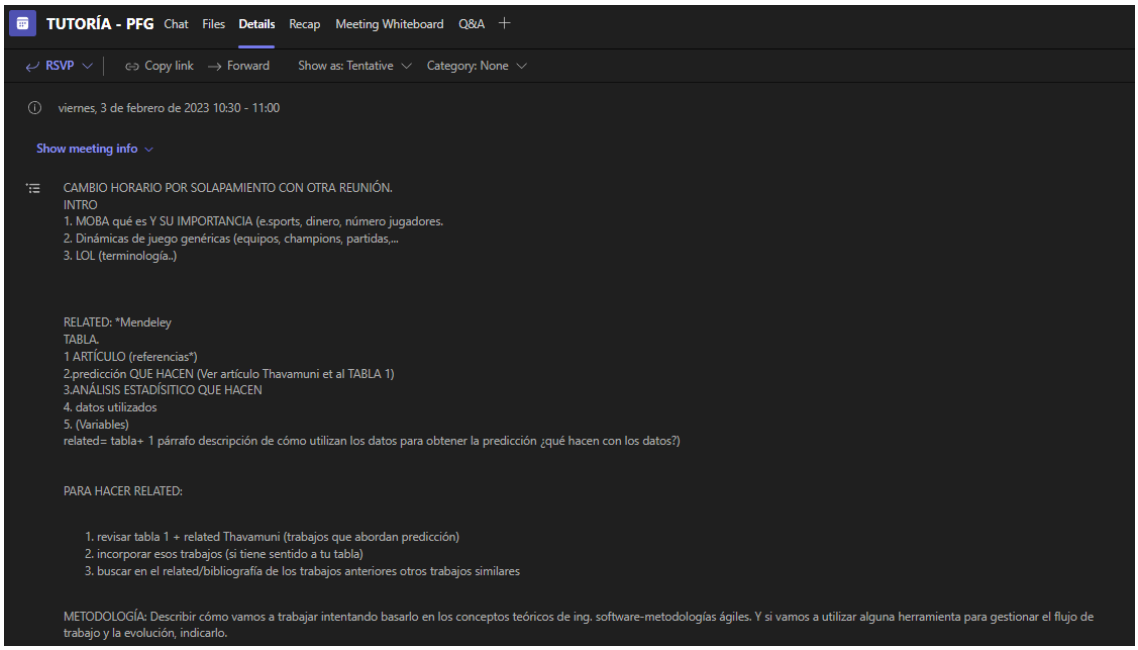


*Ilustración 6. Reunión del Sprint 1.*

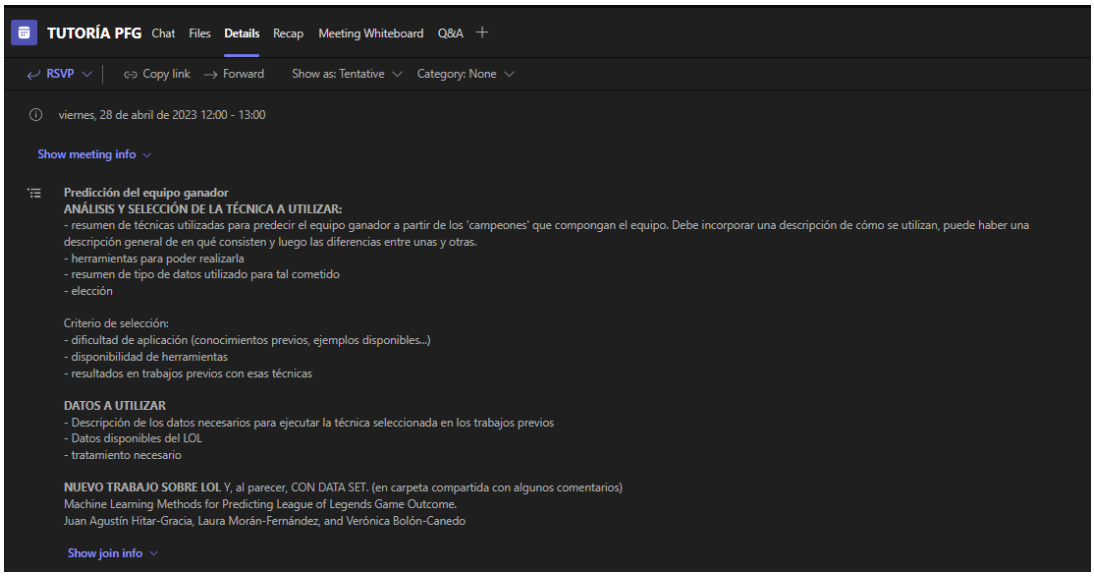


*Ilustración 7. Reunión del Sprint 7.*

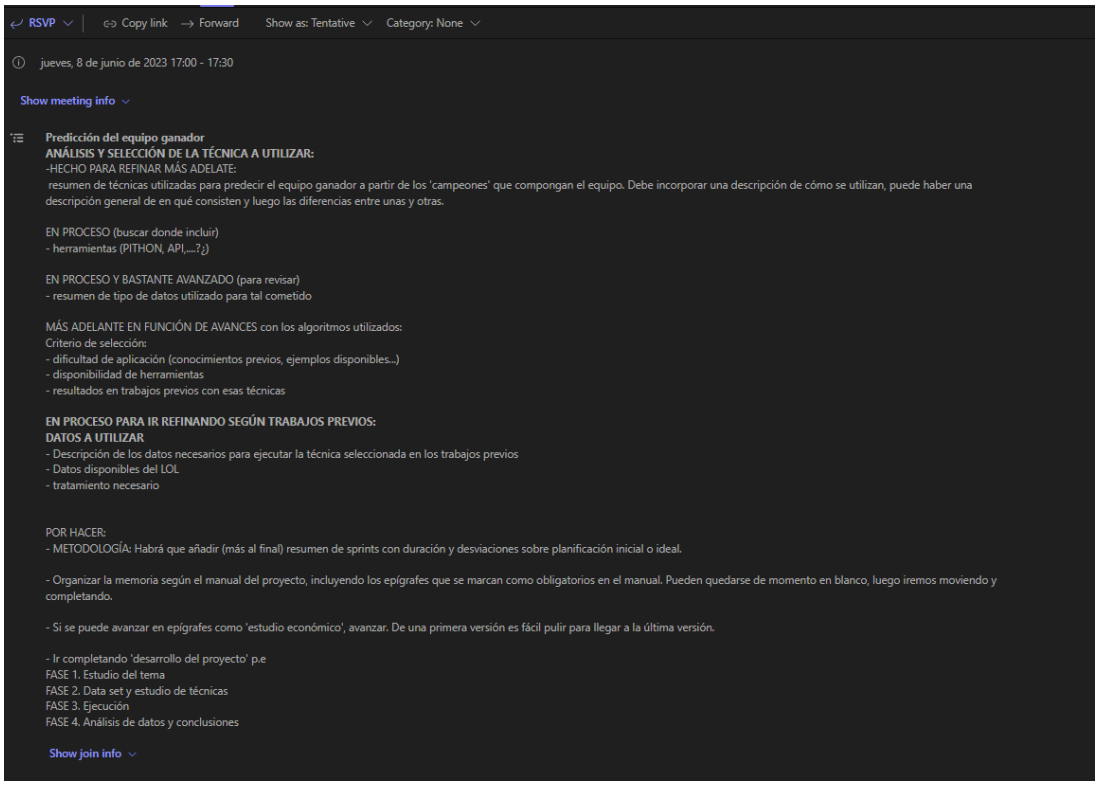


*Ilustración 8. Reunión del Sprint 10.*

Hemos realizado finalmente 12 iteraciones entre el 3 de Febrero y el 23 de Junio, aunque hubo un trabajo previo de investigación antes de la preparación de la memoria que incluyó tareas como la realización de la propuesta y la búsqueda de trabajos previos entre otras.

# 5. Estudio estadístico

## 5.1. Técnicas

Como hemos visto en el estado del arte, estas son las técnicas de machine learning más utilizadas.Todas estas técnicas de machine learning han sido utilizadas en los trabajos previos, y todas tienen ventajas y desventajas. He decidido que comenzaré utilizando Logistic Regression (LR) para el diseño del modelo de predicción del equipo ganador de League of Legends. Una de las razones por las que he elegido LR sobre otras técnicas para comenzar es debido a la experiencia previa que tengo trabajando con este modelo de regresión. Además, LR es un modelo simple pero poderoso que puede utilizarse en problemas de clasificación binaria y multiclase, lo que lo hace adecuado para nuestro caso de estudio. También utilizaré el resto de las técnicas analizadas y así podré comparar los resultados de las diferentes técnicas para este problema para determinar cuál de ellas puede resultar más apropiada.

## 5.2. Dataset

A la hora de escoger dataset para realizar el análisis estadístico existen varias opciones: utilizar un dataset utilizado en alguno de los proyectos que he estudiado antes, utilizar un dataset público o crear un data set propio. Los dataset de los trabajos previos sobre LOL [16] y [10] no están disponibles. El dataset de [16] es privado y fue extraído manualmente de la API de Riot Games y el dataset de [10] fue extraído de una página web que ya no existe [10]. Respecto a los data set públicos, desafortunadamente, a pesar de existir diferentes opciones en páginas web como la de Kaggle [29], no encontré ningún dataset que pudiera ser utilizado para este trabajo puesto que, aunque hay muchos conjuntos de datos, ninguno tenía justo lo queríamos, por ejemplo [30], [31] tienen una cantidad considerable de datos, pero la tabla no tiene todos los atributos que necesito, [32] guardaba datos de partidas profesionales, pero son pocos los datos y, como el primer ejemplo, la tabla no tenía todos los atributos que busco. Por tanto, teniendo en cuenta que dentro de los objetivos del proyecto se consideró crear nuestro propio data set, la opción seleccionada es la de utilizar un data set propio creado desde cero. Para ello utilizaré la API oficial de Riot Games para extraer los datos públicos del juego, siguiendo un procedimiento similar al seguido por Do y otros [16] en su trabajo.

Utilizando la API oficial de Riot Games voy a obtener miles de partidas divididas por nivel de habilidad utilizando el sistema de "elo" del League of Legends que divide por categorías a los jugadores en IRON, BRONZE, SILVER, GOLD, PLATINUM, DIAMOND, MASTER, GRANDMASTER, CHALLENGER.

Este dataset contendrá información sobre los campeones seleccionados por cada equipo, así como sobre el resultado de la partida y otros datos relevantes. En concreto, el dataset tendrá las siguientes columnas: match_id, champion_name, champion_role, team, outcome, rank, kills, deaths, assists y gameDuration.

- match_id: ID de la partida
- champion_name: Nombre del campeón
- champion_role: Rol del campeón en la partida
- team: Equipo al que pertenece el campeón
- outcome: Resultado de la partida, equipo ganador
- rank: Nivel de habilidad de los jugadores
- kills: Número de asesinatos que hizo el campeón en la partida
- deaths: Número de muertes que tuvo el campeón en la partida
- assists: Número de asistencias que dio el campeón en la partida
- gameDuration: Duración de la partida en segundos

### 5.2.1. Extracción de Dataset

Para realizar el proceso de extracción del dataset utilicé el lenguaje de programación PHP, SQL para hacer operaciones en la base de datos, y utilicé phpMyAdmin para crear una base de datos y todas las tablas necesarias. Para la creación del script de php utilicé la aplicación SublimeText. Decidí utilizar estas herramientas por dos razones, la primera es que son todas gratuitas y la segunda es que tengo experiencia previa con estas herramientas.

El proceso de extracción de los datos en bruto a través de la API oficial duró un par de semanas. Para acceder a las funciones de la API es necesaria la adquisición de una llave, y estas están restringidas a 100 accesos cada 2 minutos, y como la cantidad de información de partidas que tenía que extraer era 210.950 partidas, y solo podía acceder a 100 de esas partidas cada 2 minutos, la cantidad de tiempo necesario para analizarlas todo fue considerable.

Este proceso consistió en llenar varias tablas:

- "summonersdatat": En esta tabla extraemos todos los jugadores de todas las clasificaciones de Europa.
- "matchesids": En esta tabla extraemos todas las partidas jugadas por todos los jugadores de toda Europa.
- "matchinformation": En esta tabla extraemos los datos de las partidas jugadas.

Para extraer datos de la API de Riot Games es necesario utilizar diversos endpoints que podemos encontrar en la página web de developer de Riot Games [33], un endpoint es una URL donde una aplicación o programa puede acceder a ciertos datos o funciones de una API. Cada endpoint se define con una ruta (la URL) y un método HTTP (GET, POST, PUT, DELETE, etc.), en el caso de la API de Riot Games, solo hay endpoints con métodos GET, que se utilizan para obtener datos, acceder a funciones, etc.

Para llenar la primera tabla ("summonersdatat") utilicé el endpoint de la API de Riot Games /lol/league/v4/entries/{queue}/{tier}/{division}, cuando un endpoint incluye una palabra entre corchetes "{}" significa que debemos sustituirlo por unos datos específicos, en este caso estas son las opciones posibles:

- "queue": "RANKED_SOLO_5x5", "RANKED_FLEX_SR", "RANKED_FLEX_TT", en este caso solo nos interesa la primera opción, la de clasficatorias.
- "tier": "."IRON", "BRONZE", "SILVER", "GOLD", "PLATINUM", "DIAMOND".
- "division": "I", "II", "III", "IV"

En la ilustración 8 está el código utilizado en php para crear los tiers y las divisiones, los "highTier" son aquellos que no tienen divisiones como MASTER, GRANDMASTER y CHALLENGERS, y necesitan hacer uso de otro endpoint que describo más adelante.

```php
$tiers = array("IRON", "BRONZE", "SILVER", "GOLD", "PLATINUM", "DIAMOND");
$divisions = array("I", "II", "III", "IV");
$highTier = array("master", "grandmaster", "challenger");
```

*Ilustración 9. Creación de tiers y divisiones.*

Ahora simplemente hacemos dos bucles para iterar por todos los tiers y divisiones y extraer todos los jugadores como podemos ver en la siguiente ilustración:

```php
foreach ($tiers as $tier) {
    $url="";
    foreach ($divisions as $division) {
        $url = "https://$region.api.riotgames.com/lol/league/v4/entries/RANKED_SOLO_5x5/$tier/$division?api_key=$apiKey";
        $response = file_get_contents($url);
        $data = json_decode($response, true);
        //print("<pre>".print_r($data,true)."</pre>");
        $counter++;
        if($counter == 100){
            $counter = 0;
            sleep($sleepDuration);
        }
```

*Ilustración 10. Bucles para extraer jugadores.*

Los bucles de la Ilustración 9 hará que iteremos desde IRON I, IRON II, IRON III, etc. Hasta DIAMOND IV. Está también la URL que he comentado anteriormente, en el primer caso la URL será esta:

"*https://euw1.api.riotgames.com/lol/league/v4/entries/RANKED_SOLO_5x5/IRON/I?api_key=APIKEY*", siendo "APIKEY" la llave gratuita que nos de la página web. Una vez hacemos la request, compruebo si hemos hecho 100 peticiones, ya que como ese es el límite cada dos minutos, si se cumple tenemos que parar el programa dos minutos. Ahora que hemos terminado la request, hemos obtenido todos los jugadores de ese tier y podemos insertarlos en la base de datos siguiendo el código de la siguiente ilustración:

```
foreach ($data as $entry) {
    $queueType = $entry['queueType'];
    $tier = $entry['tier'];
    $rank = $entry['rank'];
    $summonerId = $entry['summonerId'];
    $summonerName = $entry['summonerName'];
    $leaguePoints = $entry['leaguePoints'];
    $wins = $entry['wins'];
    $losses = $entry['losses'];
    // Insert or update summoner league in database
    $query = "INSERT INTO summonersdatat (queueType, tier, rank, summonerId, summonerName, leaguePoints, wins, loses) VALUES ('$
        queueType', '$tier', '$rank', '$summonerId', '$summonerName', $leaguePoints, $wins, $losses) ON DUPLICATE KEY UPDATE
        leaguePoints=$leaguePoints, wins=$wins, loses=$losses";
    $result = $mysqli->query($query);
}
```

*Ilustración 11. Insertar jugadores en la database.*

Esto funcionará con todos los tiers excepto los de MASTER, GRANDMASTER y CHALLENGER, ya que estos no tienen divisiones del I al IV y hay que utilizar un endpoint distinto: */lol/league/v4/{tier}leagues/by-queue/{queue}.* Podemos ver la utilización de este endpoint en la siguiente ilustración:

```
foreach ($highTier as $tier) {
    $url = "https://$region.api.riotgames.com/lol/league/v4/${tier}leagues/by-queue/RANKED_SOLO_5x5?api_key=$apiKey";
    $response = file_get_contents($url);
    $data = json_decode($response, true);
```

*Ilustración 12. Endpoint para high tier.*

Una vez la petición descrita en la ilustración 11 ha sido mandada a la API, podemos insertar los datos recibidos en la respuesta a la tabla:

```
foreach ($data['entries'] as $entry) {
    $queueType = 'RANKED_SOLO_5x5';
    switch ($tier) {
        case 'master':
            $tier = 'MASTER';
            break;
        case 'grandmaster':
            $tier = 'GRANDMASTER';
            break;
        case 'challenger':
            $tier = 'CHALLENGER';
            break;
    }
    $rank = $entry['rank'];
    $summonerId = $entry['summonerId'];
    $summonerName = $entry['summonerName'];
    $leaguePoints = $entry['leaguePoints'];
    $wins = $entry['wins'];
    $losses = $entry['losses'];
    // Insert or update summoner league in database
    $query = "INSERT INTO summonersdatat (queueType, tier, rank, summonerId, summonerName, leaguePoints, wins, loses) VALUES ('$queueType'
        , '$tier', '$rank', '$summonerId', '$summonerName', $leaguePoints, $wins, $losses) ON DUPLICATE KEY UPDATE leaguePoints=$
        leaguePoints, wins=$wins, loses=$losses";
    $result = $mysqli->query($query);
}
```

*Ilustración 13. Insertar jugadores en la database 2.*

Una vez ha terminado la ejecución del código de la ilustración 12, la tabla estará llena:

| IdSumData | tier | rank | summonerName | summonerId |
|---|---|---|---|---|
| 4921 | IRON | I | PrivyDecree | 06FKLkR5ZVNRG_gCig2kwYac6icXdpO1 |
| 4922 | IRON | I | Samoussa 1st | FePTnVSPNvLARixXmTNdgwRIXoECNO |
| 4923 | IRON | I | nightooi | jA_T46tzfMYqkTwbxNXvtmzhStqnFgE7vF |
| 4924 | IRON | I | Biwoe iYD | qu5UfjCQlvLV-xzSwB8jHTbMBC_tFv8lAw |
| 4925 | IRON | I | e1m1ha | oxMtEM-0JREdMiGU-PJ5jUd09QuLdp3U |
| 4926 | IRON | I | 7DiamonD7 | jMvTzK5RcxGRsgxWWfeKVQUc6-q_8xr |
| 4927 | IRON | I | Rufynello | G1w20TTPxgFFcm9tnNdEt-lumE4Dz71N |
| 4928 | IRON | I | Rockwolf78 | uljQjL3JiC2LAqfZYm2_k-TNSm4IO65W0 |
| 4929 | IRON | I | Silverskale | UEkOIN8MTbJKujkhvT0uvalgzTrj3VDsr6C |
| 4930 | IRON | I | tuiguillunt | QGbfYS_MstSmryadQjRlyQFODmY6UC |
| 4931 | IRON | I | Walgore999 | ka9vVstuFEn41dLWKeKV1BR3Jy0ENgT |
| 4932 | IRON | I | Sunila | uBlkkuLy06uZxrcT85mQ8Vkg-qAETgOR |

*Ilustración 14. Parte de la tabla "summonersdatat".*

La ilustración 13 muestra once jugadores y su información dentro de la tabla "summonersdatat". La tabla guarda en total datos de 20875 jugadores.

Antes de llenar la siguiente tabla es necesario añadir una columna a la tabla "summonersdatat" que guarde el valor de una variable llamada ppuid, que es un número interno que tiene cada jugador y que utilizando los endpoints anteriores no podemos acceder a ese dato. Necesitamos hacer uso del *endpoint /lol/summoner/v4/summoners/by-name/{summonerName}*, sustituyendo *summonerName* por el nombre del jugador podemos acceder a ese dato y agregarlo a la tabla como indica el código de la siguiente ilustración:

```
$query = "SELECT summonerName FROM summonersdatat WHERE ppuid=''";
$result = $mysqli->query($query);
while ($row = $result->fetch_assoc()) {
    $summonerName = $row['summonerName'];
    if (strpos($summonerName, ' ') !== false) {
        $summonerName = str_replace(' ', '%20', $summonerName);
    }
    var_dump($row);
    echo "<br>";
    $urlSum = "https://$region.api.riotgames.com/lol/summoner/v4/summoners/by-name/".$summonerName."?api_key=$apiKey";
    $responseSum = file_get_contents($urlSum);
    print("<pre>".print_r($responseSum,true)."</pre>");
    if (strpos($http_response_header[0], "404") !== false) {
        $queryDelete = "DELETE FROM summonersdatat WHERE summonerName='".$row['summonerName']."'";
        $resultDelete = $mysqli->query($queryDelete);
    }
    $dataSum = json_decode($responseSum, true);
    $queryUpdate = "UPDATE summonersdatat SET ppuid='".$dataSum['puuid']."' WHERE summonerName='".$row['summonerName']."';";
    $resultUpdate = $mysqli->query($queryUpdate);
    $counter++;
```

*Ilustración 15. Extraer el ppuid de los jugadores.*

Acabado este primer proceso, podemos pasar a llenar la segunda tabla, "matchesids". Esta tabla guarda los identificadores de las partidas que ha jugado cada jugador, y para poder acceder al historial de partidas de un jugador necesitamos su ppuid y el endpoint /lol/match/v5/matches/by-puuid/{ppuid}. Una vez hacemos la request podemos insertar los datos en la tabla como indica la siguiente ilustración:

```php
$query = "SELECT * FROM summonersdatat";
$result = mysqli_query($mysqli, $query);
while ($row = mysqli_fetch_assoc($result)) {
    $counter++;
    if($counter == 100){
        $counter = 0;
        sleep($sleepDuration);
    }
    $puuid = $row['ppuid'];
    $tier = $row['tier'];
    $url="https://europe.api.riotgames.com/lol/match/v5/matches/by-puuid/$puuid/ids?start=0&count=20&api_key=$apiKey";
    $responseSum = file_get_contents($url);
    $dataSum = json_decode($responseSum, true);
    foreach ($dataSum as $matchId) {
        $queryId = "INSERT INTO matchesids (MatchId, Tier) VALUES ('$matchId', '$tier')";
        $resultId = $mysqli->query($queryId);
    }
}
```

*Ilustración 16. Extraer datos a la tabla "matchesids".*

| IdMatch | MatchId | Tier |
|---|---|---|
| 2662 | EUW1_6365816370 | IRON |
| 2663 | EUW1_6365640463 | IRON |
| 2664 | EUW1_6365596699 | IRON |
| 2665 | EUW1_6365520778 | IRON |
| 2666 | EUW1_6365493111 | IRON |
| 2667 | EUW1_6364304214 | IRON |
| 2668 | EUW1_6364295830 | IRON |
| 2669 | EUW1_6364281756 | IRON |

*Ilustración 17. Parte de la tabla "matchesids".*

La ilustración 16 muestra siete partidas de la tabla "matchesids". En esta tabla tenemos guardados 210800 partidas, su identificador y el nivel de habilidad de la partida.

Una vez haya terminado la ejecución, podemos pasar a llenar la última tabla, "matchinformation". Esta es la tabla que vamos a utilizar como dataset y la he explicado a fondo en el apartado anterior.

Para llenar esta tabla utilizaremos el endpoint /lol/match/v5/matches/{matchId} para extraer la información de la partida, y almacenaremos la información que queremos insertar en la tabla en variables pertinentes. Las ilustraciones 17 y 18 muestran el código necesario para realizar estas operaciones:

```php
$query = "SELECT * FROM matchesids WHERE Checked2 != 1";
$result = mysqli_query($mysqli, $query);
while ($row = mysqli_fetch_assoc($result)) {
    $counter++;
    if($counter == 100){
        $counter = 0;
        sleep($sleepDuration);
    }
    $matchId = $row['MatchId'];
    $tier = $row['Tier'];
    $urlMatch = "https://europe.api.riotgames.com/lol/match/v5/matches/$matchId?api_key=$apiKey";
    $responseMatch = file_get_contents($urlMatch);
    $dataMatch = json_decode($responseMatch, true);
```

*Ilustración 18. Endpoint para llenar la tabla "matchinformation".*

```php
$match_duration = $dataMatch['info']['gameDuration'];
$players = $dataMatch['info']['participants'];
//print("<pre>".print_r($players,true)."</pre>");
foreach ($players as $player) {
$champion_name = $player['championName'];
$champion_role = $player['teamPosition'];
$kills = $player['kills'];
$deaths = $player['deaths'];
$assists = $player['assists'];
if ($player['teamId'] == 100) {
    $team_id = 1;
    if ($player['win']) {
        $outcome = 1;
    }
    else{
        $outcome = 2;
    }
}
else{
    $team_id = 2;
    if ($player['win']) {
        $outcome = 2;
    }
    else{
        $outcome = 1;
    }
```

*Ilustración 19. Almacenar información en variables.*

Por último, insertamos en la tabla la información y marcamos la partida como comprobada usando estas consultas de sql descritas en la siguiente ilustración.

```php
$query = "INSERT INTO matchinformation (matchid, champion_name, champion_role, team, outcome, rank, kills,
    deaths, assists, gameDuration) VALUES ('$matchId', '$champion_name', '$champion_role', '$team_id', '$
    outcome', '$tier', '$kills', '$deaths', '$assists', '$match_duration')";
$result2 = $mysqli->query($query);

}
$queryUpdate = "UPDATE matchesids SET Checked2 = 1 WHERE MatchId = '$matchId';";
$resultUpdate = $mysqli->query($queryUpdate);
```

*Ilustración 20. Llenar la tabla "matchinformation".*

| MatchId ▲ 1 | champion_name | champion_role | team | outcome | rank | kills | deaths | assists | gameDuration |
|---|---|---|---|---|---|---|---|---|---|
| 3352614578 | Darius | TOP | 1 | 2 | DIAMOND | 1 | 8 | 4 | 1435 |
| 3352614578 | Nidalee | JUNGLE | 1 | 2 | DIAMOND | 2 | 7 | 4 | 1435 |
| 3352614578 | Cassiopeia | MIDDLE | 1 | 2 | DIAMOND | 3 | 5 | 4 | 1435 |
| 3352614578 | Senna | BOTTOM | 1 | 2 | DIAMOND | 1 | 5 | 1 | 1435 |
| 3352614578 | Blitzcrank | UTILITY | 1 | 2 | DIAMOND | 3 | 4 | 2 | 1435 |
| 3352614578 | Gnar | TOP | 2 | 2 | DIAMOND | 7 | 6 | 5 | 1435 |
| 3352614578 | JarvanIV | JUNGLE | 2 | 2 | DIAMOND | 8 | 0 | 11 | 1435 |
| 3352614578 | Lux | MIDDLE | 2 | 2 | DIAMOND | 3 | 3 | 16 | 1435 |
| 3352614578 | Zeri | BOTTOM | 2 | 2 | DIAMOND | 11 | 1 | 4 | 1435 |
| 3352614578 | Lulu | UTILITY | 2 | 2 | DIAMOND | 0 | 0 | 18 | 1435 |
| 3352637456 | Yone | TOP | 1 | 2 | DIAMOND | 4 | 7 | 9 | 1927 |
| 3352637456 | Viego | JUNGLE | 1 | 2 | DIAMOND | 5 | 9 | 7 | 1927 |
| 3352637456 | Yasuo | MIDDLE | 1 | 2 | DIAMOND | 7 | 12 | 6 | 1927 |

*Ilustración 21. Parte de la tabla "matchinformation".*

Esta ilustración muestra cómo queda una partida guardada dentro de la tabla (MatchId 3352614578) y parte de otra partida (MatchId 3352637456).

### 5.2.2. Procesamiento del dataset

Una vez extraídos los datos y el dataset está completo, es hora de procesar el dataset para eliminar la mayor cantidad de ruido y modificarlo para que pueda ser utilizado por un modelo de machine learning.

El ruido corresponde a partidas que no son apropiadas incluir en el estudio, puesto que corresponden a partidas no finalizadas o partidas en las que se han producido fenómenos anómalos. En trabajos previos como en [16], se eliminaban las partidas con una duración menor de un determinado número de minutos para eliminar los resultados de partidas no válidas. La duración media de una partida de League of Legends varía entre los 25 y los 30 según el nivel de habilidad, por lo tanto, aquellas partidas que duren menos de 20 minutos serán eliminadas. Las razones por las cuales una partida dure menos de 20 minutos son variadas, puede ser que la diferencia de habilidad entre los dos equipos sea muy grande, puede ser que un jugador de un equipo se haya desconectado, puede ser que uno o varios jugadores de un equipo sean "smurfs" (jugadores de rango más alto que están jugando en una cuenta de rango bajo, por ejemplo: un jugador con una cuenta en Diamante está jugando una partida en una cuenta en Oro).

Durante el filtrado de estos datos, observé un problema que había con el atributo de la duración de las partidas, y es que este atributo no es fiable, ya que noté como existían muchas entradas donde la duración de la partida no cuadraba con el número de asesinatos, muertes y asistencias de la partida. Por ejemplo, según la respuesta de la API de Riot Games, ocurrió una partida la cuál duró cinco minutos y ocurrieron más de 30 asesinatos entre los dos equipos. Esta es una partida imposible, ya que, al morir un jugador, este debe esperar varios segundos (entre 10 segundos y 50, dependiendo de varios factores como la duración actual de la partida, el número de asesinatos globales, cantidad de oro del jugador, etc.) antes de reaparecer. Si se tiene en cuenta también el tiempo que se necesita para volver a la batalla, es simplemente imposible que la partida durase cinco minutos. Al igual que este ejemplo, encontré muchas otras incongruencias sobre la relación de tiempo de partida y número de asesinatos, por lo tanto, ese atributo no se puede utilizar para filtrar los datos.

Buscando otras maneras de filtrar los datos, encontré la manera de filtrar aquellas partidas en las que una o varias personas se hubieran desconectado, y esta es eliminar aquellas partidas en las que haya uno o más jugadores cuya puntuación sea de cero asesinatos, cero muertes y cero asistencias. Filtrar aquellas partidas donde haya smurfs es más complejo, pero buscando irregularidades en partidas encontré partidas donde había jugadores con más de 40 asesinatos, encontré un jugador que en una partida había acabado con 100 asesinatos, estas partidas las voy a eliminar también ya que es un claro ejemplo de una partida con un "smurf". Es muy complicado eliminar todas las partidas en las que haya "smurfs", pero si eliminamos aquellas en las que haya una persona con más de 40 asesinatos podremos acabar con un buen número de partidas injustas. Por tanto, para eliminar estas partidas, primero seleccione aquellas entradas donde hubiera jugadores con cero asesinatos, cero muertes y cero asistencias (para los desconectados) y elimine todas las entradas cuya "MatchId" coincidiese con esta, ya que para eliminar una partida donde hubiese un jugador desconectado, es necesario eliminar también el resto de jugadores de esa partida. Para eliminar las partidas con "smurfs" el proceso es muy similar, seleccionando esta vez aquellas entradas cuyos asesinatos superasen los 40. En la siguiente ilustración podemos ver estos procesos de eliminación:

```php
function eliminateEntriesWithZeroKDA($mysqli){
    $query = "SELECT * FROM `matchinformation` WHERE kills=0 AND deaths=0 AND assists=0;";
    $result = $mysqli->query($query);
    $lastId = '';
    while ($row = mysqli_fetch_assoc($result)) {
        $matchId = $row['MatchId'];
        if ($lastId != $matchId) {
            $queryDelete = "DELETE FROM `matchinformation` WHERE MatchId = '$matchId';";
            $resultDelete = $mysqli->query($queryDelete);
            $lastId = $matchId;
        }
    }
}
function eliminateEntriesWithOverXKills($mysqli){
    $query = "SELECT * FROM `matchinformation` WHERE kills >= 40;";
    $result = $mysqli->query($query);
    $lastId = '';
    while ($row = mysqli_fetch_assoc($result)) {
        $matchId = $row['MatchId'];
        if ($lastId != $matchId) {
            $queryDelete = "DELETE FROM `matchinformation` WHERE MatchId = '$matchId';";
            $resultDelete = $mysqli->query($queryDelete);

            $lastId = $matchId;
        }
    }
}
```

*Ilustración 221. Eliminar entradas con desconectados y "smurfs".*

Una vez hemos eliminado la mayor cantidad de ruido posible, es hora de modificar el dataset para que pueda ser utilizado por un modelo de machine learning. Para que un modelo de machine learning pueda utilizar el dataset, se debe realizar un preprocesamiento de los datos para convertir las variables categóricas en variables numéricas. En el caso de los campeones y roles, se puede utilizar la técnica de one-hot encoding para crear columnas binarias para cada campeón y para cada rol, lo que resultaría en una tabla con 162 columnas para los campeones, 5 columnas para los roles y el resto de las variables igual.

El one-hot encoding es una técnica de codificación utilizada en el procesamiento de datos que convierte una variable categórica en una matriz numérica binaria. En esta técnica, se crea una columna separada para cada categoría única en la variable categórica, y en cada fila se marca con un "1" en la columna correspondiente a la categoría y con un "0" en todas las demás columnas.

Por ejemplo, en el caso de los roles, tenemos cinco opciones: “top”, “jungle”, “mid”, “bottom” y “support”, como podemos ver en la ilustración 22.

| MatchId | champion_name | champion_role |
|---|---|---|
| EUW1_6365816370 | Kayn | TOP |
| EUW1_6365816370 | Nunu | JUNGLE |
| EUW1_6365816370 | Lissandra | SUPPORT |
| EUW1_6365816370 | Aphelios | BOTTOM |
| EUW1_6365816370 | Akshan | MIDDLE |

*Ilustración 232. Imagen de una tabla de la database donde guardamos los datos de partidas.*

Esto no funcionará para un modelo de machine learning, ya que necesitan números exclusivamente, por lo tanto, utilizando one-hot encoding dividiremos los roles en las siguientes columnas descritas en la ilustración 23:

| role_TOP | role_JUNGLE | role_MIDDLE | role_BOTTOM | role_SUPPORT |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |

*Ilustración 243. Ejemplo de cómo se vería la tabla tras el procesamiento.*

Lo mismo ocurrirá con los nombres de los campeones, que, en este caso, al ser 162 los campeones posibles, significará que tendremos 162 columnas.

Este proceso lo hice utilizando el lenguaje de programación Python y la herramienta Visual Studio Code porque es gratuita. En la ilustración 24 se describe el código necesario para realizar el one-hot encoding en las columnas de "champion_name" y "champion_role".

```
from sklearn.preprocessing import OneHotEncoder
# One-hot encode the categorical columns
encoder = OneHotEncoder(sparse_output=False)
encoded_features = encoder.fit_transform(data[['champion_name', 'champion_role']])
encoded_df = pd.DataFrame(encoded_features, columns=encoder.get_feature_names_out(['champion_name', 'champion_role']))

# Drop the original categorical columns and add the encoded ones
data = data.drop(['champion_name', 'champion_role'], axis=1)
data = pd.concat([data, encoded_df], axis=1)
```

*Ilustración 254. Código para realizar one-hot encoding.*

Una vez está realizado el encoding, el siguiente paso es definir nuestras variables "X" e "y":

- La variable "X" es el conjunto de datos conocidos como los predictores. El modelo la utilizará para predecir el valor de la variable objetivo "y".
- "y" es la variable objetivo que el modelo está intentando predecir.

En nuestro caso la variable objetivo "y" es la columna de "outcome", mientras que la variable "X" es el resto de las columnas, en la siguiente ilustración defino estas variables:

```
# Define X and y
X = data.drop('outcome', axis=1)
y = data['outcome']
```

*Ilustración 265. Definición de las variables X e y.*

Después de haber realizado el encoding y definido las variables, se pueden dividir los datos en entrenamiento y testing:

```
# Split the data into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

*Ilustración 276. División en training y testing.*

La variable "random_state" que aparece en la ilustración 25 es una semilla que se utiliza para generar números aleatorios, al darle un valor aseguramos que los números aleatorios generados por el código serán siempre los mismos cada vez que lo ejecutamos. Es decir, los sets de entrenamiento y testing serán siempre los mismos a menos que cambiemos el número 42.

## 5.3. Clasificación del dataset

Una vez todo el dataset fue filtrado, lo dividí para los distintos experimentos según el nivel de habilidad en:

- Nivel de habilidad bajo: IRON, BRONZE, SILVER (14500 partidas)
- Nivel de habilidad medio: GOLD, PLATINUM, DIAMOND (15000 partidas)
- Nivel de habilidad alto: MASTER, GRANDMASTER, CHALLENGER (20000 partidas)

Aparte de dividirlos por nivel de habilidad, hice otro dataset el cual no tiene los datos de los roles de los jugadores, para comprobar como influyen los roles a la precisión de los modelos.

También dejé una copia del dataset sin separar para observar cómo se comportarían los modelos al no dividir por nivel de habilidad.

Esta es la lista de datasets que obtuve para la ejecución:

- DatasetLowSkill - Nivel de habilidad bajo
- DatasetLowSkillNoRoles - Nivel de habilidad bajo sin roles
- DatasetMidSkill - Nivel de habilidad medio
- DatasetMidSkillNoRoles - Nivel de habilidad medio sin roles
- DatasetHighSkill - Nivel de habilidad alto
- DatasetHighSkillNoRoles - Nivel de habilidad alto sin roles

- DatasetFull - Dataset sin separar
- DatasetFullNoRoles - Dataset sin separar sin roles

## 5.4. Ejecución

Para la ejecución de los métodos de Machine Learning utilicé el lenguaje de programación Python y la aplicación Visual Studio Code porque tengo experiencia con su utilización, ya que la he utilizado en otras materias, además es gratuita.

Teniendo preparados los datasets, se pueden crear todos los modelos de machine learning que he mencionado anteriormente.

### 5.4.1. Regresión logística

Para crear el modelo de regresión logística primero necesitamos importar el modelo de la biblioteca sklearn en Python.

Después creamos una instancia del modelo vacío. Ahora necesitamos entrenarlo con nuestros datos. Para ello alimentamos al modelo con nuestros datos de entrada ("X_train") y de salida ("y_train"). Durante el entrenamiento el modelo intenta encontrar la mejor línea o hiperplano que más se ajusta a nuestros datos para minimizar el error de predicción.

Con "mejor línea", la mejor manera de explicarlo es imaginar que estas intentando separar puntos rojos de puntos azules en un gráfico. La "mejor línea" es aquella que mejor separa los puntos azules de los rojos. En espacios con más dimensiones esta línea se convierte en un hiperplano. En nuestro proyecto los puntos rojos podrían ser aquellas partidas donde el outcome es victoria del equipo rojo (1) y los puntos azules aquellas partidas donde el outcome es victoria del equipo azul (2).

En otras palabras, la regresión logística trata de definir un límite de decisión tal que la mayoría de los puntos rojos estén a un lado y la mayoría de los puntos azules estén al otro.

Una vez el modelo ha terminado el entrenamiento, podemos utilizarlo para que haga predicciones con los datos de prueba "X_test" y almacenaremos estos resultados en "y_pred".

Por último, evaluaremos el modelo utilizando la función de accuracy_score, donde compararemos los resultados de las predicciones que ha hecho el modelo en "y_pred" con los verdaderos valores de "y_test", esta función nos dará un valor entre 0 y 1 y corresponde a la precisión del modelo.

En la ilustración 27 podemos ver este proceso.

```
from sklearn.linear_model import LogisticRegression
# Create a logistic regression model
model = LogisticRegression()
# Train the model
model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)
# Print the accuracy
print('Accuracy:', accuracy_score(y_test, y_pred))
```

*Ilustración 287. Regresión Logística.*

### 5.4.2. Árbol de decisión

El árbol de decisión es muy similar a la regresión logística, lo único que cambia es lo que tenemos que importar de la biblioteca de sklearn y la manera de crear el modelo. Este modelo creará una estructura de árbol donde cada nodo representa una característica (por ejemplo, "kills"), cada enlace entre nodos representa una decisión (por ejemplo, si "kills" es mayor a 10) y cada hoja es una predicción de resultado (ganar o perder).

El árbol de decisión navegará por este árbol basándose en las características de entrada hasta que llega a una hoja y hace una predicción. Por ejemplo, podría decir "Si el campeón es 'X', y el rol del campeón es 'Y', y el equipo tuvo más de 10 asesinatos, entonces predicen que el equipo ganará la partida".

Este proceso se repetirá para todas las partidas del dataset y el algoritmo aprenderá qué camino es el mejor para predecir el resultado de un partido.

En la ilustración 28 podemos ver el proceso de creación, entrenamiento y testing del modelo:

```
from sklearn.tree import DecisionTreeClassifier
# Crea un modelo de árbol de decisión
model = DecisionTreeClassifier()
# Entrena el modelo
model.fit(X_train, y_train)
# Realiza predicciones
y_pred = model.predict(X_test)
# Imprime la precisión
print('Precision:', accuracy_score(y_test, y_pred))
```

*Ilustración 298. Árbol de decisión.*

### 5.4.3. Random Forest

El método Random forest es similar al árbol de decisión, la única diferencia es que, a la hora de crear el modelo, es necesario incluir la variable de "n_estimators", que representa el número de árboles que se deben construir en el bosque. Aumentar este número hace que se puedan obtener resultados más precisos, sacrificando eficiencia computacional. En mi caso utilicé 100 árboles, ya que, al subir a 200 árboles, el coste computacional aumentaba exponencialmente y la mejora en los resultados no era grande (aumentaba la precisión unas décimas, de 79.1 a 79.2, mientras que el tiempo de ejecución aumentaba unos diez minutos).

El modelo creará 100 árboles utilizando subconjuntos aleatorios del conjunto de datos de entrenamiento. Esto significa que para cada árbol individual en el bosque está seleccionando aleatoriamente una muestra de los datos de entrenamiento con reemplazo (es decir, un dato puede ser seleccionado más de una vez) para entrenar ese árbol. También selecciona aleatoriamente un subconjunto de las características para cada árbol. Esta aleatorización adicional ayuda a evitar el sobreajuste al no permitir que ningún árbol se "ajuste" demasiado a los datos de entrenamiento, y también permite que los árboles sean "expertos" en diferentes aspectos de los datos.

Por ejemplo, si tuviéramos 5 partidas y creásemos 3 árboles, así se podrían crear estos árboles:

1. Árbol 1, entrenado en las partidas 1, 2, 2, 4, utilizando "kills" y "assists".
2. Árbol 2, entrenado en las partidas 2, 4, 5, 5, utilizando "deaths" y "assists".
3. Árbol 3, entrenado en las partidas 1, 1, 2, 2, 3, 3, 5, utilizando "kills" y "deaths".

En la ilustración 28 podemos observar el código necesario para crear, entrenar y testear el modelo:

```
from sklearn.ensemble import RandomForestClassifier
# Crea un modelo de Random Forest
model = RandomForestClassifier(n_estimators=100, random_state=42)
# Entrena el modelo
model.fit(X_train, y_train)
# Realiza predicciones
y_pred = model.predict(X_test)
# Imprime la precisión
print('Accuracy:', accuracy_score(y_test, y_pred))
```

*Ilustración 309. Random forest.*

### 5.4.4. Gradient Boosting Classifier (GBC)

El modelo de Gradient Boosting Clasifier es similar al Random Forest en el sentido de que utiliza arboles de decisión también, pero en lugar de hacerlos en paralelo, los hace de manera secuencial.

Primero el algoritmo entrena un primer árbol de decisión llamado árbol débil con los datos.

Luego, el algoritmo calcula los residuos de este primer árbol, que son la diferencia entre las predicciones del árbol y los valores reales de los datos.

A continuación, el algoritmo entrena un segundo árbol, pero en lugar de predecir la variable objetivo, intenta predecir los residuos del primer árbol, es decir, intenta corregir los errores del árbol anterior.

Este proceso se repetirá hasta que se cree el número de árboles especificado en la variable de "n_estimators", en caso de que no se especifique este dato, se crearán por defecto 100 árboles.

En la ilustración 29 podemos observar la creación, entrenamiento y testeo del modelo:

```
from sklearn.ensemble import GradientBoostingClassifier
# Crea un modelo de Gradient Boosting
model = GradientBoostingClassifier(random_state=42)
# Entrena el modelo
model.fit(X_train, y_train)
# Realiza predicciones
y_pred = model.predict(X_test)
# Imprime la precisión
print('Accuracy:', accuracy_score(y_test, y_pred))
```

*Ilustración 3031. Gradient Boosting Classifier.*

### 5.4.5. K-Nearest Neighbors (kNN)

El algoritmo k-Nearest Neighbors (kNN) funciona prediciendo el "outcome" de una partida basándose en la información de las 'k' partidas más similares del conjunto de entrenamiento. Esto significa que predecir el resultado de una partida, el algoritmo KNN mira las 'k' partidas más cercanas del conjunto de datos de entrenamiento (es decir, las partidas que tienen características más similares a la partida que estamos tratando de clasificar).

Estas características pueden incluir los campeones seleccionados, el rol de los campeones y las estadísticas de los jugadores (como asesinatos, muertes y asistencias), que previamente normalizamos y codificamos en variables numéricas.

Una vez que el algoritmo ha identificado las 'k' partidas más cercanas, hace una votación entre ellas para predecir el resultado de la partida actual. Por ejemplo, si k=3, y las tres partidas más cercanas terminaron en dos victorias y una derrota, entonces el algoritmo predecirá que la partida actual será una victoria.

La implementación de este modelo es muy similar al resto, la única diferencia es que antes de crear el modelo es necesario normalizar los datos de entrenamiento y testeo. Esto se debe a que kNN mide la distancia entre puntos. La otra cosa a tener en cuenta es el número de vecinos que queremos que compruebe, a mayor número de vecinos, mayor precisión. En mi caso voy a utilizar cinco. En la siguiente ilustración se puede ver esta implementación:

```
#Normalizacion de los datos de test y training
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

#Creacion, entrenamiento y testeo del modelo
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Imprime el informe de clasificación
print(classification_report(y_test, y_pred))
# Imprime la precisión
print('Precision:', accuracy_score(y_test, y_pred))
```

*Ilustración 32. Implementación del modelo kNN.*

### 5.4.6. Naive Bayes

El algoritmo de Naive Bayes predice el resultado basándose en la evidencia proporcionada por las características de la partida.

Para cada una de estas características, el modelo Naive Bayes calcula la probabilidad condicional de que la partida sea una victoria o una derrota, dado el valor de esa característica. El modelo luego asume que cada una de estas características contribuye de forma independiente a la probabilidad final, de ahí su nombre de "naive" o ingenuo en español.

Esto significa que la presencia de un campeón específico en una partida no cambiará la influencia de los roles de los campeones en el resultado de la partida, por ejemplo. Este es un supuesto muy fuerte y a menudo no es cierto en situaciones reales.

Este es un algoritmo muy rápido y eficiente, pero sus predicciones pueden ser menos precisas si las características del conjunto de datos no son realmente independientes.

En la ilustración 33 se puede ver la implementación de este método.

```
# Crea un modelo de Naive Bayes
model = GaussianNB()
# Entrena el modelo
model.fit(X_train, y_train)
# Realiza predicciones
y_pred = model.predict(X_test)
# Imprime la precisión
print('Precision:', accuracy_score(y_test, y_pred))

# Imprime el informe de clasificación
print(classification_report(y_test, y_pred))
```

*Ilustración 33. Implementación del modelo Naive Bayes.*

## 5.5. Resultados

En la siguiente tabla podemos observar el rendimiento de los distintos modelos para el dataset con los roles incluidos:

| Modelo | Precisión nivel de habilidad bajo | Precisión nivel de habilidad medio | Precisión nivel de habilidad alto | Sin clasificación |
|---|---|---|---|---|
| **Regresión Logística** | 50.02% | 50.03% | 51.34% | 50.04% |
| **Árbol de decisión** | 70.8% | 83.2% | 78.86% | 76.32% |
| **Random Forest** | 77.0% | 78.4% | 79.3% | 77.91% |
| **Gradient Boosting** | 77.1% | 77.3% | 75.0% | 78.28% |
| **K-Nearest Neighbors** | 73.5% | 75.5% | 76.6% | 75.9% |
| **Naive Bayes** | 50.7% | 49.8% | 51.3% | 50.3% |

*Tabla 2. Precisión de cada modelo con roles.*

En esta tabla podemos observar el rendimiento de cada modelo, pero esta vez sin considerar los roles:

| Modelo | Precisión nivel de habilidad bajo | Precisión nivel de habilidad medio | Precisión nivel de habilidad alto | Sin clasificación |
|---|---|---|---|---|
| **Regresión Logística** | 50.02% | 50.3% | 51.34% | 50.04% |
| **Árbol de decisión** | 70.9% | 83.5% | 79.1% | 76.5% |
| **Random Forest** | 76.9% | 78.4% | 79.1% | 77.6% |
| **Gradient Boosting** | 77.1% | 76.2% | 75.0% | 78.3% |
| **K-Nearest Neighbors** | 73.9% | 75.7% | 76.7% | 75.9% |
| **Naive Bayes** | 50.7% | 49.8% | 51.3% | 50.3% |

*Tabla 3. Precisión de cada modelo sin roles.*

Al aplicar el modelo de regresión logística y el de Naive Bayes observé como tenían una precisión baja (un 50% de acierto, que es similar al que obtendríamos con un método aleatorio). La razón de esto es que la regresión logística es una técnica que asume que existe una relación lineal entre las variables de entrada y la variable de salida [34]. Como esta relación no es lineal, los modelos que pueden capturar relaciones no lineales (como Random Forest, Decision Trees y Gradient Boosting) funcionan mejor. Para el caso de Naive Bayes, el problema principal es la estrecha relación que hay entre las distintas características, lo que hace que un algoritmo como Naive Bayes que asume que son independientes entre ellas tenga una peor precisión.

Como podemos ver en las tablas anteriores (Tabla 2 y Tabla 3), el resto de modelos tienen una precisión relativamente alta (>70%) para cualquiera de los datasets utilizados. El Gradient Boosting es el más preciso para nivel de habilidad bajo, el árbol de decisión para nivel de habilidad medio, y el Random Forest para nivel de habilidad alto.

También se puede observar que incluir los roles en el dataset no genera grandes mejoras en la predicción. El mayor aumento de precisión en la predicción se da con el método Gradient Boosting para el dataset de habilidad medio, que solo aumenta un 0.9% al incluir los roles de cada personaje.

# 6. Resultados del trabajo

## 6.1. Comparación de resultados con resultados previos

Comparando los resultados obtenidos en las ejecuciones de los diferentes métodos empleados con los resultados presentados en los trabajos previos analizados, podemos concluir que nuestros resultados son muy similares a los suyos. En el caso de [17], utilizaron partidas de nivel de habilidad alto y obtuvieron un 77.39% de precisión, en nuestro caso, con partidas de nivel de habilidad alto hemos conseguido un 79.3% de precisión utilizando Random Forest. [17], [17] y [9] no especifican el nivel de habilidad, pero consiguieron un 78%, 75.1% y 62% de precisión respectivamente, en nuestro caso, sin clasificar datos, nuestra precisión más alta es de 78.37% con Gradient Boosting. Es necesario recalcar que estos trabajos son distintos al nuestro, ellos tienen en cuenta otras variables, por ejemplo [17] utilizó datos de los objetos comprados, [17] utilizó los datos de los jugadores (no solo de los campeones como hacemos aquí), los únicos trabajos que son similares a este son [17] y [9], ya que utilizan únicamente las composiciones de los equipos para predecir el equipo ganador. Los resultados que obtenemos con regresión logísticas son peores que los de otros trabajos, lo cual significa que se debe estudiar cómo mejorar estos resultados.

## 6.2. Aportación de este trabajo a la comunidad

Este trabajo aporta un estudio en la línea del que podemos encontrar en trabajos como [17] y [9] lo que nos permite contrastar si nuestros resultados son similares. Además, generamos un data set original desde los datos públicos, y este dataset está preparado para ser usado por otros que deseen realizar estudios similares. Además, hemos incluido diferentes tipos de tratamientos. Por un lado, hemos dividido el dataset en categorías, las hemos evaluado y hemos descubierto que no hay mucha diferencia entre las predicciones en función de la categoría. Esto contrasta con otros trabajos que sí veían que a los rangos altos había menos precisión que en rangos bajos.

También hemos hecho pruebas incluyendo los roles de los campeones en las partidas, y sin incluirlos, y hemos llegado a la conclusión que la inclusión de los roles no afecta mucho a la precisión de los modelos. Es muy probable que no hayamos detectado diferencias de incluir los roles o no porque es conocido por todos los jugadores que hay unos campeones que se juegan para un rol y hay otros que no se juegan para ese rol y en nuestro dataset no hay muchas partidas en las que un campeón apropiado para una línea en particular por ejemplo para la línea de mid, sea jugado en una línea distinta como por ejemplo top. Estos datos prácticamente no están en el dataset y por eso considero que haber añadido el rol no es muy determinante

porque ya el campeón y su rol de alguna manera en las partidas está determinado y se elige a sabiendas de que el personaje es adecuado para ese rol.

Añadir que el dataset generado este proyecto, así como sus clasificaciones en categorías y todos los scripts con la ejecución de los distintos modelos utilizados se han añadido al proyecto en forma de material adicional.

## 6.3. Desviación de la metodología

Como no teníamos una planificación cerrada, no hay una desviación como tal, pero si es cierto que las primeras iteraciones fuimos más lento de lo que nos habría gustado y por el contrario en las últimas hemos sido más ágiles. En un principio nos planteamos la misma cantidad de trabajo entre dos reuniones, sin embargo, en las primeras iteraciones, la compatibilización del proyecto con otras actividades como el seguimiento de otras materias han hecho que en esos primeros sprints los avances hayan sido más lento. Otra cosa que ha ralentizado el avance es la lectura y el análisis de tantos trabajos previos y artículos científicos los cuales me requirieron más tiempo del esperado. Por otra parte, los últimos sprints han sido mucho más productivos, una vez el resto de las materias estaban terminadas y el dataset estaba completo, la cantidad de trabajo que he podido abordar ha sido mucho mayor. Igualmente, como la íbamos adaptando en cada sprint no ha habido mucho problema en ajustarse.

| Sprints | Fecha | Tiempo Ideal (h) | Tiempo Real (h) | Tema |
|---|---|---|---|---|
| Sprint 1 | 03-feb | 20 | 10 | Introducción (MOBA, LOL, DINAMICAS). |
| Sprint 2 | 17-feb | 30 | 40 | Comienzo de related work (lectura de trabajos). |
| Sprint 3 | 02-mar | 30 | 20 | Continuación de related work. |
| Sprint 4 | 17-mar | 30 | 10 | Continuación de related y empezamos metodología. |
| Sprint 5 | 31-mar | 30 | 20 | Continuación de related y metodología y comenzar con herramientas estadísticas. |
| Sprint 6 | 14-abr | 30 | 10 | Técnicas de machine learning existentes. |
| Sprint 7 | 28-abr | 25 | 25 | Analisis y selección de técnicas a usar, creacion del dataset. |
| Sprint 8 | 12-may | 25 | 15 | Continuación con la creacion del dataset |
| Sprint 9 | 02-jun | 25 | 40 | Tratamiento del dataset, herramientas para utilizarlo. |
| Sprint 10 | 08-jun | 25 | 60 | Ejecución de los modelos con herramientas. |
| Sprint 11 | 20-jun | 25 | 50 | Tratamiento de resultados, estudio económico, otras revisiones. |
| Sprint 12 | 23-jun | 25 | 50 | Conclusiones, terminar revisiones. |

*Tabla 4. Sprints con fechas, horas y tema.*

En la tabla 4 se puede observar un resumen de todos los sprints realizados con sus fechas, el tiempo ideal que querría haber dedicado a cada uno versus el tiempo real que he podido dedicar, así como un pequeño resumen de qué se hizo en el sprint.

El siguiente gráfico representa la progresión temporal de nuestro proyecto. Podemos ver dos líneas distintas: La línea azul representa el planning ideal que diseñamos al comienzo del proyecto. Por otro lado, la línea roja representa el planning real que seguimos. Esta línea nos muestra la realidad de la implementación del proyecto, reflejando cómo durante la mitad del proyecto las iteraciones tuvieron menos horas de las previstas debido principalmente a la necesidad de tener que compaginar el proyecto con otras materias. Una vez esas materias fueron superadas (más o menos a partir de mitad de mayo o séptimo sprint) podemos observar como el tiempo dedicado a cada sprint aumentó bastante.

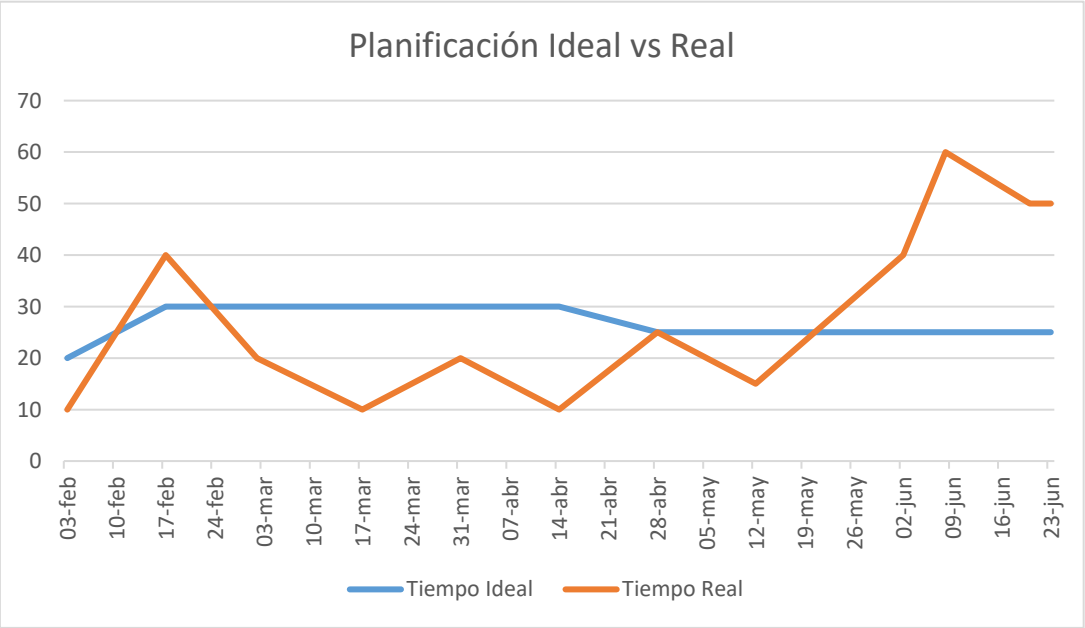


*Ilustración 34. Grafico comparativo de tiempo ideal vs real.*

# 7. Estudio Económico

## 7.1. Costes de desarrollo

Teniendo en cuenta que el único integrante del equipo es un ingeniero de software junior, y los salarios de estos trabajadores suelen ser alrededor de 23.973,88 [35] euros al año, y el proyecto ha durado unos cuatro meses y medio, podemos determinar que los costes de desarrollo han sido unos 9000 euros (4.5 meses son unos 0.375 años, teniendo 24000 euros al año podemos calcular que el coste son 9000 euros).Costes de material

Para el desarrollo del proyecto utilicé un potente ordenador que pudiese ejecutar los distintos programas y scripts, las especificaciones de este ordenador son: un procesador Intel(R) Core(TM) i7-9700 CPU, una tarjeta gráfica NVIDIA GeForce RTX 3070, 16 GB de RAM y Window 10 de sistema operativo.

Junto con el ordenador, utilicé dos monitores, ASUS VZ24EHE 23.8" y LG 24MK430H-B 23.8", un ratón Razer Naga X y un teclado Forgeon Clutch RGB 60% Switch Blue.

Una cosa a tener en cuenta es que el coste de este material es durante el desarrollo del proyecto, y como el proyecto ha durado 4.5 meses presento en la siguiente tabla el coste estimado por mes:

| Nombre | Coste (€) | Coste por mes (€) |
|---|---|---|
| Ordenador | 2000 | 750 |
| ASUS VZ24EHE | 120.99 | 45.37 |
| LG 24MK430H-B | 99 | 37 |
| Razer Naga X | 79.99 | 29.99 |
| Forgeon Clutch | 104.99 | 39.37 |
| **Total** | **2404.97** | **895.73** |

*Tabla 5. Coste materiales.*

Los programas utilizados fueron Visual Studio Code para la realización del código, para la creación de la base de datos y el dataset utilicé phpMyAdmin, XAMPP y la API de Riot Games. Todos son gratuitos. Para escribir de la memoria utilicé Word, cuya licencia fue gratuita al comprar el ordenador.

## 7.2. Coste Total

| | **Coste (€)** |
|---|---|
| Desarrollo | 9000 |
| Material | 895.73 |
| **Total** | **9895.73** |

*Tabla 6. Coste total.*

## 7.3. Perspectiva empresarial

La perspectiva empresarial más sencilla sería crear una página web, ya que el coste tanto de diseño web, desarrollo web y mantenimiento de la página puede ser hecho por mí. El coste monetario vendría a la hora de comprar un alojamiento web y un nombre de dominio, estos suelen variar de precio, pero normalmente están a unos 10€. Suponiendo que comprase el Hosting web de SiteGround [36] por 12€ al mes, y un dominio en NameCheap [37] por 6.98€ al año, el coste ascendería a 150.98€ al año.

La manera más rápida de amortizar este coste sería incluir publicidad en la página web utilizando proveedores de publicidad en línea como Google AdSense [38]. Si utilizásemos esta manera, necesitaríamos generar unos 14€ al mes para amortizar los costes.

El dinero que se puede ganar con Google AdSense depende de varios factores como el número de visitantes de la página, el porcentaje de usuarios que hacen clic en los anuncios (tasa de clics) y el coste promedio por clic (CPC).

El CPC varía dependiendo de la industria y la ubicación geográfica, pero suele oscilar entre 0.10€ y 2.0€. Siguiendo el estudio de [39], que investigaron el CPC en todas las comunidades españolas, vamos a suponer que el CPC es de 0.91€. Para la tasa de clics vamos a utilizar los datos del articulo [40], que estudiaron entre varias cosas la tasa de clics media de todos los sectores del mercado, y llegaron a la conclusión que la tasa de clics es del 2.62%.

Con un coste por clic (CPC) de 0.91€ y una tasa de clics del 2.62%, podemos calcular los ingresos potenciales de la siguiente manera:

- Como he comentado antes, necesitaríamos generar unos 14€ al mes. Esto se traduce en alrededor de 1538 clics al mes (14€/0.91€ por clic).
- Con una tasa de clics del 2.62%, necesitaríamos alrededor de 58755 visitas por mes (1538/0.0262).
- Por lo tanto, si consiguiéramos más de 58755 visitas al mes estaríamos ganando dinero.

Utilizando la página ActivePlayer [41] podemos ver cómo actualmente hay 150 millones de jugadores del League of Legends en todo el mundo. Por lo tanto, necesitaríamos que al mes visitasen la página el 0.039% de jugadores del LoL (58755 / 150M * 100).

Si redujésemos los jugadores en lugar de todo el mundo a toda Europa, que cuenta con unos 35 millones de jugadores aproximadamente según un artículo en la página GetDroidTips [42], necesitaríamos que al mes visitasen la página un 0.168% (58755 / 35M * 100).

# 8. Conclusiones

## 8.1. Grado cumplimiento objetivos

Para evaluar el progreso y éxito de este proyecto, es fundamental revisar en qué medida se han cumplido los objetivos establecidos al inicio. Voy a comparar los resultados obtenidos con cada uno de los objetivos planteados originalmente, permitiendo así evaluar el grado de cumplimiento de estos.

El primero de los objetivos, conocer el estado del arte en las estadísticas empleadas en el análisis de videojuegos sí que lo he cumplido porque en el apartado "2. Estado del arte" podemos observar el análisis a fondo realizado sobre numerosos trabajos previos.

El segundo de los objetivos, la descripción del videojuego lo he cumplido también, he descrito completamente el videojuego en la introducción del proyecto.

El tercer objetivo, realización de un estudio estadístico, también lo considero cumplido. Como se puede observar, el proyecto consiste en la explotación de los datos públicos de un videojuego.

El cuarto objetivo, sobre la evaluación de los beneficios, inconvenientes y limitaciones, también se ha abordado durante el desarrollo del proyecto. Los beneficios han sido explicados en el apartado "6.2. Aportación de este trabajo a la comunidad", y los inconvenientes y limitaciones han sido expuestos en el apartado "5.2 Dataset" donde hablé de los problemas a la hora de encontrar datasets o las limitaciones de la utilización de un API público.

Finalmente, el quinto objetivo, referente a la obtención de experiencia en el manejo de grandes cantidades de datos a través de herramientas estadísticas también lo considero alcanzado. La realización de este proyecto me ha facilitado el poder adquirir experiencia en la utilización de este tipo de datos tanto en la obtención del dataset como su explotación a posteriori utilizando técnicas de Machine Learning.

## 8.2. Valoración personal

Cuando me enfrente a este problema pensé que haría cosas diferentes a las que he acabado haciendo, pensaba que haríamos otro tipo de estudio, pero conforme el proyecto ha ido avanzando, y sobre todo después de estudiar los proyectos similares, empezó a tomar una forma que no era la esperada, pero igualmente la sensación que he tenido después de finalizar el proyecto es la de que he aprendido muchísimo y he podido aplicar muchos de los conocimientos que he ido adquiriendo durante toda la carrera. Si que es cierto que no he conseguido crear una herramienta o aplicación que sea capaz de predecir el resultado al darle dos equipos, o que sea capaz de recomendar campeones como hacían otros trabajos. Pero he entendido mucho mejor el problema que he abordado y me veo con muchas más habilidades a

la hora de crear este tipo de aplicaciones o herramientas y que tipo de acciones debo seguir para hacerlos.

Durante este proyecto también he aprendido a utilizar otro tipo de herramientas como puede ser un gestor de bibliografía documental y a cómo extraer la información más relevante de papers académicos.

En general mi valoración de este proyecto es muy positiva y estoy muy satisfecho de los resultados obtenidos y con todo lo que he aprendido.

## 8.3. Mejoras y futuras aplicaciones del proyecto

Respecto a las mejoras sobre los métodos empleados, las enfocaría en dos partes. Por un lado, al comparar los resultados de la regresión logística con os obtenidos con otros trabajos vemos que no son similares y que además se acercan más a los resultados que obtendríamos con un método un aleatorio (50% de precisión). Una mejora clara para el futuro sería averiguar por qué la regresión logística produce esos resultados y cómo arreglarlo si se puede.

Por otra parte, mi idea es mejorar este proyecto automatizando este proceso con más datasets, porque en este caso solo he utilizado un dataset correspondiente a un parche y querría estudiar para cualquier parche. Así cuando Riot Games anuncie la salida de un nuevo parche, el programa sea capaz automáticamente de extraer el dataset de este nuevo parche y entrene los modelos. Aparte también que todo este proyecto esté disponible en una página web.

Si fuese capaz de automatizar estos procesos, habría también varias utilidades y aplicaciones de cara al futuro. Podría ser utilizado por equipos profesionales a la hora de formar estrategias y entrenar a sus equipos, por ejemplo pueden elegir dos equipos y predecir cuál de los dos ganaría, una vez el programa les de la solución, pueden poner a su equipo a jugar contra otro equipo con esa composición que le pusieron al programa para comprobar si es cierto, y si el programa acierte o se equivoque, esto puede llegar a conversaciones sobre por qué han ganado o perdido, así como incluir cambios en la composición del equipo para ver cómo se comporta el programa, viendo así qué campeones consiguen un cambio en la predicción del programa.

Jugadores no profesionales pueden utilizar este programa para varias cosas, una de ellas es para predecir si su equipo ganará la partida antes de empezar, y en el caso de que el programa le indique que va a perder la partida, el jugador puede "dodgear" la partida, esto ocurre cuando un jugador cierra el juego durante la selección de campeón, cuando la partida aún no ha comenzado.

Otro uso es para apostar en partidas profesionales como las de las ligas europeas, americanas, chinas o koreanas entre otras.

## 9. Bibliografía


[1] "The Growth of MOBA Popularity in Gaming - Gaming." http://stevb6686.weebly.com/bradley-20/the-growth-of-moba-popularity-in-gaming (accessed Feb. 14, 2023).

[2] "Solo en 2020 League of Legends generó más de 1.750 millones de dólares." https://es-us.finanzas.yahoo.com/noticias/2020-league-of-legends-gener%C3%B3-130000865.html (accessed Jun. 25, 2023).

[3] "How to Play - League of Legends." https://www.leagueoflegends.com/en-us/how-to-play/ (accessed Feb. 03, 2023).

[4] "U.GG LoL Tier List - Patch 13.12 Best Champions for All Roles in League of Legends." https://u.gg/lol/tier-list (accessed Jun. 26, 2023).

[5] Q. Li *et al.*, "A Visual Analytics Approach for Understanding Reasons behind Snowballing and Comeback in MOBA Games," *IEEE Trans Vis Comput Graph*, vol. 23, no. 1, pp. 211–220, Jan. 2017, doi: 10.1109/TVCG.2016.2598415.

[6] I. Makarov, D. Ignatov, D. Savostyanov, B. Litvyakov, and D. I. Ignatov, "Predicting Winning Team and Probabilistic Ratings in 'Dota 2' and 'Counter-Strike: Global Offensive' Video Games AIST: International Conference on Analysis of Images, Social Networks, and Texts View project Second International Workshop on Experimental Economics and Machine Learning View project Predicting Winning Team and Probabilistic Ratings in 'Dota 2' and 'Counter-Strike: Global Offensive' Video Games," 2018, doi: 10.1007/978-3-319-73013-4_17.

[7] Y. Ding, X. Hu, J. Li, J. Ye, F. Wang, and D. Zhang, "What Makes a Champion: The Behavioral and Neural Correlates of Expertise in Multiplayer Online Battle Arena Games," *https://doi.org/10.1080/10447318.2018.1461761*, vol. 34, no. 8, pp. 682–694, Aug. 2018, doi: 10.1080/10447318.2018.1461761.

[8] Z. Wang, A. Sapienza, A. Culotta, and E. Ferrara, "Personality and Behavior in Role-based Online Games," *2019 IEEE Conference on Games (CoG)*, vol. 2019-August, Aug. 2019, doi: 10.1109/CIG.2019.8848027.

[9] A. Agarwala and M. Pearce, "Learning Dota 2 Team Compositions".

[10] C.-S. Lee and I. Ramler, "Investigating the Impact of Game Features on Champion Usage in League of Legends", Accessed: Feb. 15, 2023. [Online]. Available: http://loldb.gameguyz.com/

[11] A. Semenov, P. Romov, S. Korolev, D. Yashkov, and K. Neklyudov, "Performance of machine learning algorithms in predicting game outcome from drafts in Dota 2," *Communications in Computer and Information Science*, vol. 661, pp. 26–37, 2017, doi: 10.1007/978-3-319-52920-2_3.

[12] Z. Chen *et al.*, "The Art of Drafting: A Team-Oriented Hero Recommendation System for Multiplayer Online Battle Arena Games," Jun. 2018, Accessed: Feb. 15, 2023. [Online]. Available: http://arxiv.org/abs/1806.10130

[13] D. Gourdeau and L. Archambault, "Discriminative neural network for hero selection in professional Heroes of the Storm and DOTA 2," *IEEE Trans Games*, 2019.

[14] G. Franco, M. Henrique Fonseca Ribeiro, and G. Comarela, "Towards an Interpretable Metric for DOTA 2 Players: An Unsupervised Learning Approach," *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 341–346, Oct. 2019, doi: 10.1109/BRACIS.2019.00067.

[15] A. Dallmann, J. Kohlmann, D. Zoller, and A. Hotho, "Sequential Item Recommendation in the MOBA Game Dota 2," *IEEE International Conference on Data Mining Workshops, ICDMW*, vol. 2021-December, pp. 10–17, 2021, doi: 10.1109/ICDMW53433.2021.00009.

[16] T. D. Do, S. Ioi Wang, D. S. Yu, M. G. McMillian, and R. P. McMahan, "Using Machine Learning to Predict Game Outcomes Based on Player-Champion Experience in League of Legends; Using Machine Learning to Predict Game Outcomes Based on Player-Champion Experience in League of Legends," 2021, doi: 10.1145/3472538.3472579.

[17] I. Porokhnenko, P. Polezhaev, and A. Shukhman, "Machine Learning Approaches to Choose Heroes in Dota 2".

[18] K. Conley and D. Perry, "How Does He Saw Me? A Recommendation Engine for Picking Heroes in Dota 2", Accessed: Mar. 01, 2023. [Online]. Available: http://www.youtube.com/watch?v=BAC9B9z

[19] Mariana. Pérez, "¿Qué es Metodología? » Definición, Tipos y Ejemplos 2021." https://conceptodefinicion.de/metodologia/ (accessed Mar. 28, 2023).

[20] Coelho Fabián, "Metodología: qué es, concepto y definición - Significados." https://www.significados.com/metodologia/ (accessed Mar. 28, 2023).

[21] "The Ultimate Guide to Project Management | Blog Wrike." https://www.wrike.com/blog/the-ultimate-guide-to-project-management/ (accessed Mar. 28, 2023).

[22] "Pulse of the Profession 2018: Success in Disruptive Times".

[23] E. Zavyalova, D. Sokolov, and A. Lisovskaya, "Agile vs traditional project management approaches: Comparing human resource management architectures," *International Journal of Organizational Analysis*, vol. 28, no. 5, pp. 1095–1112, Oct. 2020, doi: 10.1108/IJOA-08-2019-1857/FULL/XML.

[24] David Matthew, "Agile vs. Waterfall: Strengths and Weaknesses." https://www.simplilearn.com/waterfall-vs-agile-vs-devops-article?tag=agile%20vs%20waterfall%20methodology%20article (accessed Mar. 28, 2023).

[25] "What is Scrum Methodology in Project Management?" https://blog.planview.com/what-is-scrum-methodology-in-project-management/ (accessed Mar. 28, 2023).

[26] "What is Kanban Methodology and its Advantages?" https://blog.planview.com/what-is-kanban-methodology-and-its-advantages/ (accessed Mar. 28, 2023).

[27] "What is Extreme Programming (XP)? | Agile Alliance." https://www.agilealliance.org/glossary/xp/#q=~(infinite~false~filters~(postType~(~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'xp))~searchTerm~'~sort~false~sortDirection~'asc~page~1) (accessed Mar. 28, 2023).

[28] Bara Marc, "Roles, Eventos y Artefactos en la metodología Scrum | OBS Business School." https://www.obsbusiness.school/blog/roles-eventos-y-artefactos-en-la-metodologia-scrum (accessed Mar. 28, 2023).

[29] "Find Open Datasets and Machine Learning Projects | Kaggle." https://www.kaggle.com/datasets (accessed May 31, 2023).

[30] "(LoL) League of Legends Ranked Games | Kaggle." https://www.kaggle.com/datasets/datasnaek/league-of-legends?select=games.csv (accessed Jun. 07, 2023).

[31] "League of Legends Ranked Matches | Kaggle." https://www.kaggle.com/datasets/paololol/league-of-legends-ranked-matches?select=matches.csv (accessed Jun. 07, 2023).

[32] "League of Legends | Kaggle." https://www.kaggle.com/datasets/chuckephron/leagueoflegends?select=matchinfo.csv (accessed Jun. 07, 2023).

[33] "Riot Developer Portal." https://developer.riotgames.com/apis (accessed Jun. 21, 2023).

[34] Michael H. Kutner, Christopher J. Nachtsheim, John Neter, and William Li, "Applied linear statistical models." https://users.stat.ufl.edu/~winner/sta4211/ALSM_5Ed_Kutner.pdf (accessed Apr. 28, 2023).

[35] "BOE-A-2023-6346 Resolución de 27 de febrero de 2023, de la Dirección General de Trabajo, por la que se registra y publica el XX Convenio colectivo nacional de empresas de ingeniería; oficinas de estudios técnicos; inspección, supervisión y control técnico y de calidad." https://www.boe.es/diario_boe/txt.php?id=BOE-A-2023-6346 (accessed Jun. 22, 2023).

[36] "SiteGround – Alojamiento Web de Alta Calidad." https://siteground.es/ (accessed Jun. 22, 2023).

[37] "Buy a domain name - Register cheap domain names from $0.99 - Namecheap." https://www.namecheap.com/ (accessed Jun. 22, 2023).

[38] "Gana dinero con la monetización de tu sitio web - Google AdSense." https://adsense.google.com/start/ (accessed Jun. 22, 2023).

[39] "El CPC por países y por CCAA | Gorka Garmendia." https://www.gorkagarmendia.com/el-cpc-por-paises-por-ccaa/ (accessed Jun. 22, 2023).

[40] "Análisis comparativos y estadísticas de marketing por correo electrónico | Mailchimp." https://mailchimp.com/es/resources/email-marketing-benchmarks/ (accessed Jun. 22, 2023).

[41] "League of Legends Live Player Count and Statistics." https://activeplayer.io/league-of-legends/ (accessed Jun. 25, 2023).

[42] "League of Legends Player Count by Country / Region 2023." https://www.getdroidtips.com/league-of-legends-player-count/ (accessed Jun. 25, 2023).

## 10. Índice de ilustraciones

# 11. Índice de tablas

## Anexo I. Propuesta de Proyecto fin de grado original

| | |
|---|---|
| **Nombre alumno:** | Carlos Molina Quílez |
| **Titulación:** | Doble Grado Ingeniería Informática y Diseño y Desarrollo de Videojuegos |
| **Curso académico:** | 5º |

### TÍTULO DEL PROYECTO

Utilidad de las estadísticas públicas para determinar los mejores personajes de un MOBA

### DESCRIPCIÓN Y JUSTIFICACIÓN DEL TEMA A TRATAR

El juego League of Legends es un MOBA (Multiplayer Online Battle Arena) en el que cinco jugadores se enfrentan a otros cinco, cada uno manejando un personaje distinto y en el que cada equipo trabaja para destruir la base enemiga. Al ser un juego multijugador recibe "parches" constantemente. En estos parches se hacen distintos cambios, ya sea a personajes (bajarle la vida a un personaje, subirle el daño de ataque a otro, etc), objetos (son comprados por los jugadores con oro y mejoran a su personaje), o al juego en si (por ejemplo, modificando los objetivos secundarios). Estos parches salen cada varias semanas, 3-4 normalmente, y hacen que las estadísticas de los diferentes personajes cambien:

WinRate: % de victorias.

BanRate: % de baneo, que ocurre cuando un personaje es bloqueado antes de empezar la partida, lo que hace que dicho personaje no se pueda elegir.

PickRate: % de pickeo, las veces que se juega cada personaje.

Analizando las estadísticas de los jugadores se pueden obtener los personajes más populares y fuertes de cada parche. Sin embargo, son muchas las preguntas asociadas a estas estadísticas que nos podemos hacer:

¿Es posible, a partir de las estadísticas públicas predecir qué personajes serán los mejores para cada parche? ¿Están relacionadas las estadísticas públicas con las habilidades y características asignadas a cada personaje? ¿Se pueden predecir las estadísticas de los personajes partir de los cambios realizados cada parche?

Llevo años jugando este juego y siempre tenía las estadísticas a mano cuando quería subir rangos, pero al ser difíciles de entender a primera vista, dificultaban esta mejora. Pretendo mostrar cómo pueden ser utilizadas para determinar qué personajes son los mejores en un parche.

Realizar un trabajo de estas características me permitirá poner en práctica los conocimientos adquiridos a lo largo de la carrera, en particular los referentes a la estadística, aplicados a un caso real.

Asimismo, podré responder a muchas preguntas que me he hecho a lo largo de los años respecto a las estadísticas en este videojuego y me ayudará en caso de que en el futuro decida dedicarme a los eSports ya sea como entrenador o mánager de un equipo.

Con este trabajo pretendo también crear una manera de analizar estadísticas y sacar conclusiones concisas y reales que sean útiles para mejorar en el videojuego, tanto para jugadores casuales como equipos profesionales.

Para poder abordar este trabajo, en primer lugar, seleccionaremos y organizaremos los datos a tener en cuenta en el estudio y sus posibles clasificaciones. A partir de ahí definiremos preguntas más concretas que puedan ser respondidas a través de un análisis estadístico de los datos que nos ayude a explorar su potencialidad. En función de los datos seleccionados y de las preguntas concretas, seleccionaremos el análisis estadístico más adecuado y las herramientas más adecuadas para ejecutarlo. Finalmente, analizaremos los resultados del análisis estadístico y las amenazas de validez a nuestro trabajo.

**OBJETIVOS DEL PROYECTO**

Los objetivos del proyecto son:

- Conocer el estado del arte en las estadísticas empleadas en el análisis de videojuegos.
- Descripción del videojuego donde se aplicará el análisis estadístico y de las bases de datos de dónde se obtendrán los datos a tratar.
- Realizar un estudio estadístico a partir del cual podamos concluir si a partir de los datos públicos de un videojuego podemos extraer información útil sobre el juego
- Evaluar los beneficios, inconvenientes y limitaciones que supone la utilización de estadística y datos públicos para obtener información útil sobre un videojuego.
- Obtener experiencia en el manejo de grandes cantidades de datos a través de herramientas estadísticas en un contexto real.

**METODOLOGÍA**

La metodología se establecerá en las primeras fases del proyecto.

## PLANIFICACIÓN DE TAREAS

Aunque una vez definida la metodología del proyecto las tareas se definan con mayor precisión, la realización de este proyecto contendrá las siguientes tareas:

- Seleccionar y organizar los datos a tener en cuenta en el estudio y sus posibles clasificaciones.
- Definir preguntas de investigación concretas que puedan ser respondidas a través de un análisis estadístico de los datos.
- En función de los datos seleccionados y de las preguntas realizadas, seleccionar el análisis estadístico y las herramientas más adecuadas para ejecutarlo.
- Analizar los resultados del análisis estadístico y las amenazas de validez a nuestro trabajo.

## OBSERVACIONES ADICIONALES

La tutora de este proyecto sería África Domingo Montes que ha revisado esta propuesta.