UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Gandia

Propuesta de un sistema de apoyo a las decisiones para el proceso de draft y formación de equipos en partidas de League of Legends

Trabajo Fin de Grado

Grado en Tecnologías Interactivas

AUTOR/A: Canut Domínguez, Carlos María

Tutor/a: Sánchez Anguix, Víctor

Cotutor/a: Alberola Oltra, Juan Miguel

CURSO ACADÉMICO: 2023/2024

RESUMEN

Carlos Maria Canut Dominguez

Julio/2023

Tutores: Victor Sánchez Anguix
Juan Miguel Alberola Oltra

Departamento: Tecnologias Interactivas

La Inteligencia Artificial (IA) está revolucionando diversos campos, incluido el videojuego competitivo. League of Legends, un destacado juego de Arena de Batalla en Línea Multijugador (MOBA), sirve como punto central de este estudio. Utilizando datos de partidas clasificatorias públicas y partidas profesionales, presentamos un sistema de recomendación de selección de campeones impulsado por IA. Este sistema aprovecha datos históricos y técnicas avanzadas de aprendizaje automático para ofrecer recomendaciones precisas de campeones, capacitando a los equipos profesionales en la toma estratégica de decisiones durante la fase de selección. A medida que la IA continúa transformando el panorama de los deportes electrónicos, este documento explora su aplicación dentro de League of Legends, arrojando luz sobre la evolución de la dinámica de los videojuegos competitivos.

Palabras clave: league of legends; e-sports; analítica deportiva; inteligencia artificial; sistemas de apoyo a la decisión

ABSTRACT

Carlos Maria Canut Dominguez

July/2023

Advisors: Victor Sánchez Anguix
Juan Miguel Alberola Oltra

Department: Tecnologias Interactivas

Artificial Intelligence (AI) is revolutionizing various domains, including competitive video gaming. League of Legends, a prominent Multiplayer Online Battle Arena (MOBA) game, serves as the focal point of this study. Utilizing data from public ranked games and professional matches, we introduce an AI-driven champion selection recommendation system. This system leverages historical data and advanced machine learning techniques to provide precise champion recommendations, empowering professional teams in strategic decision-making during the draft phase. As AI continues to reshape the esports landscape, this document explores its application within League of Legends, shedding light on the evolving dynamics of competitive gaming.

Keywords: League of Legends; e-sports; sports analytics; artificial intelligence; decision support systems

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Los videojuegos de competición están avanzando a pasos agigantados; sacar datos de la evolución de estos se ha convertido en una fuente invaluable de información estratégica y un recurso esencial para el crecimiento continuo de la industria.

La Inteligencia Artificial (IA) para videojuegos, también conocida como IA de Juegos, ha sido objeto de estudio activo durante décadas. En los últimos años hemos sido testigos del éxito de agentes de IA en diversos tipos de juegos, incluyendo juegos de mesa como Go, la serie de juegos Atari, juegos de disparos en primera persona (FPS) como Counter Strike, videojuegos como Super Smash Bros, juegos de cartas como el Poker, entre otros. En la actualidad, los videojuegos de estrategia captan la atención al capturar la naturaleza del mundo real. Por ejemplo, en 2019, AlphaStar, una IA creada para jugar al juego de StarCraft II logró alcanzar el nivel de gran maestro (26).

Los juegos Multiplayer Online Battle Arena (MOBA)(28) presentan un fuerte vínculo con el análisis de datos y la IA debido a su naturaleza altamente competitiva y compleja. Estos juegos involucran enfrentamientos entre dos equipos de jugadores, cada uno controlando un héroe con habilidades únicas en un campo de batalla virtual. Dado que los MOBA combinan estrategia, coordinación de equipos y toma de decisiones en tiempo real, generan una gran cantidad de datos durante cada partida.

La vasta cantidad de datos generados en los MOBA ofrece oportunidades para aplicar técnicas avanzadas de análisis de datos e IA. El análisis de partidas pasadas puede proporcionar información valiosa sobre las estrategias más efectivas, los movimientos de los jugadores y las decisiones tomadas en diferentes situaciones del juego. Estos datos históricos permiten identificar patrones y tendencias que pueden ser utilizados para optimizar las estrategias futuras, como en el estudio realizado sobre el juego DotA 2 para la predicción de campeones utilizando un modelo de redes bayesianas(29), o el utilizado usando un modelo de Monte Carlo de búsqueda en árbol(30) en el juego de Honor of Kings.

Uno de los juegos del género MOBA más relevantes dentro de la escena competitiva en la actualidad, League of Legends (1), está demostrando en los últimos años un importante interés en

crear plataformas y realizar un análisis más exhaustivo a los datos de manera que puedan ayudar a detectar patrones y tendencias. Como es el ejemplo del proyecto realizado por parte del famoso equipo Cloud 9, equipo 6 veces campeón de la primera división estadounidense (LCS), junto con Microsoft para tratar de utilizar la analitica de datos para entender el juego de una manera más efectiva (27).

La elección de campeones en League of Legends es un proceso altamente estratégico que involucra no solo las preferencias individuales de los jugadores, sino también la sinergia entre las habilidades de los campeones de un equipo y su capacidad para contrarrestar a los oponentes. Con más de 150 campeones únicos y un vasto espacio de posibilidades tácticas, la tarea de seleccionar un equipo balanceado y eficiente es un desafío que puede determinar el curso de una partida. Un ejemplo sobre la importancia de una selección de campeones en equipos profesionales y una buena preparación para esto es que los jugadores tienen la opción de practicar los campeones que van a jugar e ir mejor entrenados, además de que la elección de estos campeones decidirá el estilo de juego que deberá adoptar el equipo en base a la composición que elijan.

Este trabajo de fin de grado se centra en proporcionar una aplicación web que ayude en la fase de selección de campeones de League of Legends a equipos profesionales haciendo recomendaciones, para esto busca una combinación de datos extraídos de partidas tanto competitivas como de jugadores de alto nivel que juegan día a día en servidores públicos.

1.1. Objetivos del proyecto

El objetivo de este proyecto es el desarrollo de una aplicación web que ofrezca recomendaciones para la selección de campeones en League of Legends, diseñado con un enfoque particular en la comunidad de equipos profesionales. Reconociendo la importancia de las decisiones tácticas en el ámbito competitivo, nuestro sistema se nutre de datos provenientes de partidas que juegan los jugadores en los servidores abiertos a todo el público en el modo de juego clasificatorio, el cual, cada partida que se juega otorga puntos de ranking si se gana o resta puntos de ranking si se pierde, también conocida como cola en solitario, y partidas competitivas para brindar recomendaciones precisas y efectivas usando datos históricos y técnicas de Machine Learning (ML). Este objetivo se podría concretar en los siguientes sub-objetivos:

1. Analizar y entender el funcionamiento del League of Legends.
2. Uso de datos de partidas de cola de emparejamiento en solitario y partidas profesionales en el sistema de recomendación de campeones.
3. Crear una herramienta de diseño sencillo e intuitivo para la simulación de la fase de selección de campeón junto con las recomendaciones.

A continuación, se irá exponiendo el desarrollo realizado para alcanzar estos objetivos.

1.2. Estructura del documento

Capítulo I: Introducción En el primer capítulo se realiza una breve explicación de los objetivos del trabajo y la estructura del documento.

Capítulo II: Marco teórico En el segundo capítulo se expone el funcionamiento de los esports y del League of Legends como juego de competición, así como una explicación de cada uno de los algoritmos utilizados para la creación del modelo de recomendaciones.

Capítulo III: Propuesta En el tercer capítulo se expone la propuesta planteando primero un análisis de los requisitos expuestos por el cliente.

Capítulo IV: Implementación del sistema de recomendaciones En este capítulo se expone el desarrollo realizado para la creación de las recomendaciones desde la extracción de datos hasta el algoritmo utilizado para las recomendaciones.

Capítulo V: Implementación de la Aplicación Web El quinto capítulo expone el desarrollo y estructura planteados para la aplicación web que hace uso del algoritmo de recomendaciones para ofrecerlas al cliente.

Capítulo VI: Conclusiones Capítulo en el que se realiza una valoración final del proyecto y futuras mejoras.

Capítulo VII: Bibliografía Por último, en este capítulo se exponen las referencias consultadas para el trabajo.

Capítulo 2

Marco Teórico

En este capítulo se realiza una breve descripción del funcionamiento del videojuego League of Legends, los deportes electrónicos, también conocidos como esports y el desarrollo del League of Legends como Esport y su estado actual. Por último, se exponen herramientas existentes y su funcionamiento.

2.1. Esports

Los esports son competiciones de videojuegos organizadas en un formato profesional. En estas competiciones, jugadores individuales o equipos compiten entre sí en diferentes títulos de videojuegos en línea.

Los esports han ganado una enorme popularidad en las últimas décadas, únicamente generado por competiciones de videojuegos, han llegado aproximadamente a 1.400 millones de dólares estadounidenses(3) y se han convertido en un fenómeno mundial con una base de seguidores cada vez mayor rondando aproximadamente los 270 millones de espectadores en 2022 (4).

Los eventos de esports se llevan a cabo tanto en línea como en locaciones físicas, atrayendo a una audiencia masiva a través de transmisiones en vivo por plataformas de streaming y televisión. Los espectadores pueden seguir las competiciones desde la comodidad de sus hogares o asistir a estadios llenos de fanáticos para vivir la emoción en persona.

Al igual que en los deportes tradicionales, los esports cuentan con jugadores profesionales que se dedican a tiempo completo a entrenar y perfeccionar sus habilidades. Muchos de ellos cuentan con contratos con equipos y patrocinadores, y pueden ganar premios en metálico significativos en los torneos más grandes, una aproximación del sueldo de un jugador profesional podemos obtenerla de las declaraciones de Hal Biagas, director ejecutivo de la LCS (liga profesional americana de League of Legends), el cual afirmó que los jugadores a ese nivel tenían sueldos promedio de más de 410.000 dólares en 2022 (5).

Los esports abarcan una amplia variedad de géneros de videojuegos, como los MOBA (Multiplayer Online Battle Arena), los juegos de disparos en primera persona (FPS), los juegos de lucha,

los juegos de estrategia en tiempo real (RTS) y otros. Algunos de los títulos más populares en los esports incluyen League of Legends, Dota 2, Counter-Strike: Global Offensive, Overwatch, Fortnite, entre otros.

La comunidad de esports es apasionada y diversa, con jugadores y seguidores de todas las edades y procedencias. Los eventos de esports han adquirido gran relevancia en los últimos años, teniendo la final del mundial de League of Legends valores como los de 5 millones de espectadores viendo en directo la gran final de 2018 (6). Además, han generado oportunidades de carrera no solo para jugadores profesionales, sino también para comentaristas, entrenadores, analistas y personal de producción.

En resumen, los esports son una forma de competición profesional basada en videojuegos, que ha logrado consolidarse como un fenómeno cultural y deportivo importante en todo el mundo. Con un crecimiento continuo y una comunidad apasionada, los esports representan una industria en constante expansión y una fuente de entretenimiento para millones de personas.

2.2. League of Legends

League of Legends es un videojuego multijugador en línea del género MOBA desarrollado y publicado por Riot Games (1). Fue lanzado por primera vez el 27 de octubre de 2009. El juego fue creado por Marc Merrill y Brandon Beck, inspirados por el mapa personalizado Defense of the Ancients (DotA) del juego de estrategia en tiempo real Warcraft III: The Frozen Throne(2).

El éxito de DotA como mod dentro de Warcraft III inspiró la idea de crear un juego independiente con un concepto similar con gráficos mejorados, una interfaz de usuario más amigable y mecánicas de juego adicionales. Así es como nació League of Legends como sucesor de DotA, tomando elementos principales del género MOBA y con un fuerte énfasis en el juego competitivo.

En League of Legends, los jugadores asumen el papel de campeones, cada uno con sus propias habilidades y estilos de juego distintivos. Estos se dividen en dos equipos de 5 jugadores, y el objetivo principal es destruir el Nexo de la base enemiga, estructura fortificada ubicada dentro de su base (ver figura 2.1).

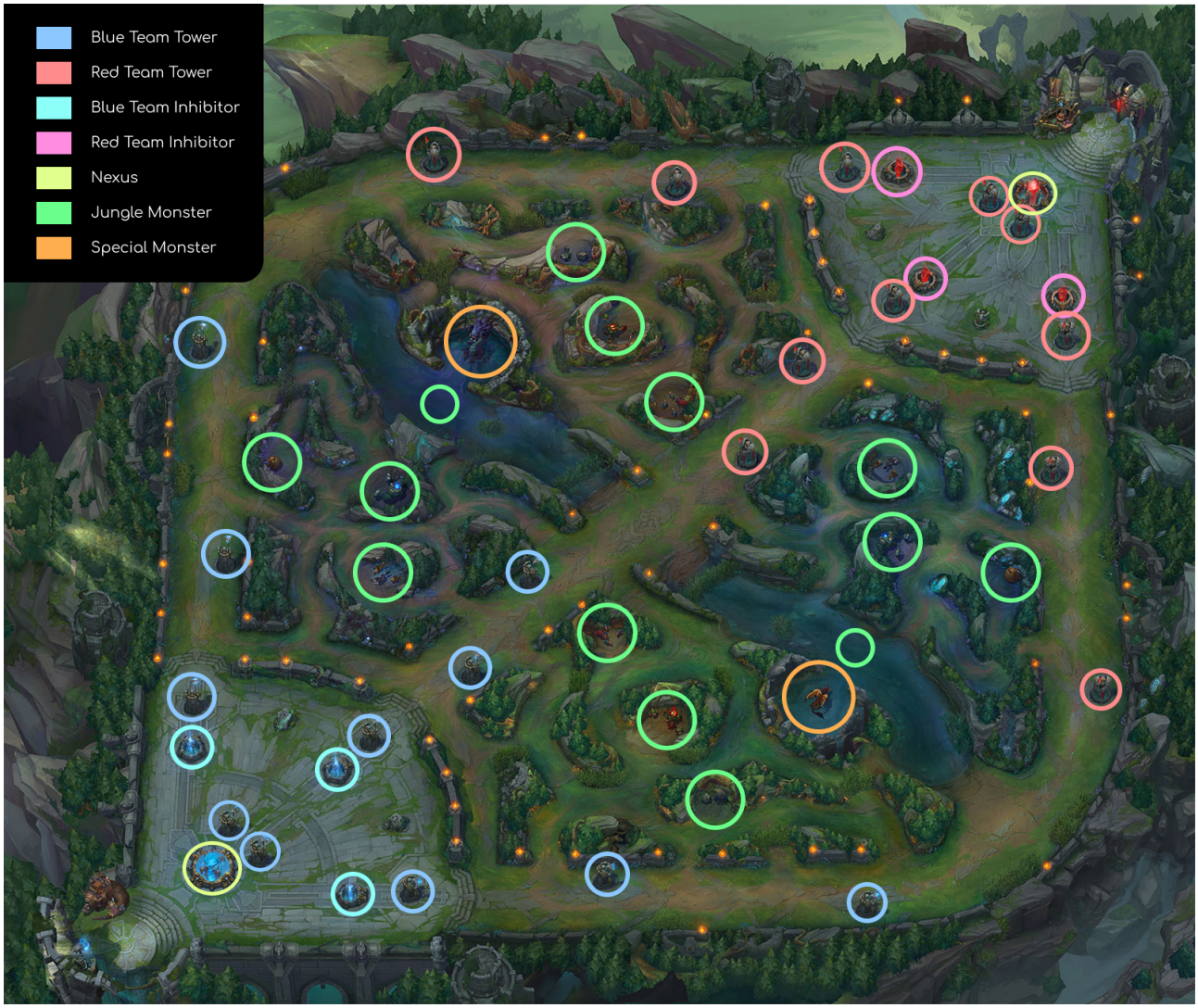


Figura 2.1: Objetivos del mapa de League of Legends. Fuente: recurso propio

El mapa de League of Legends, como se muestra en la Figura 2.2, se divide en 3 carriles: superior, central e inferior, además de una zona llamada jungla que se encuentra entre cada uno de los carriles donde aparecen monstruos los cuales ofrecen recompensas al eliminarlos (figura 2.1).

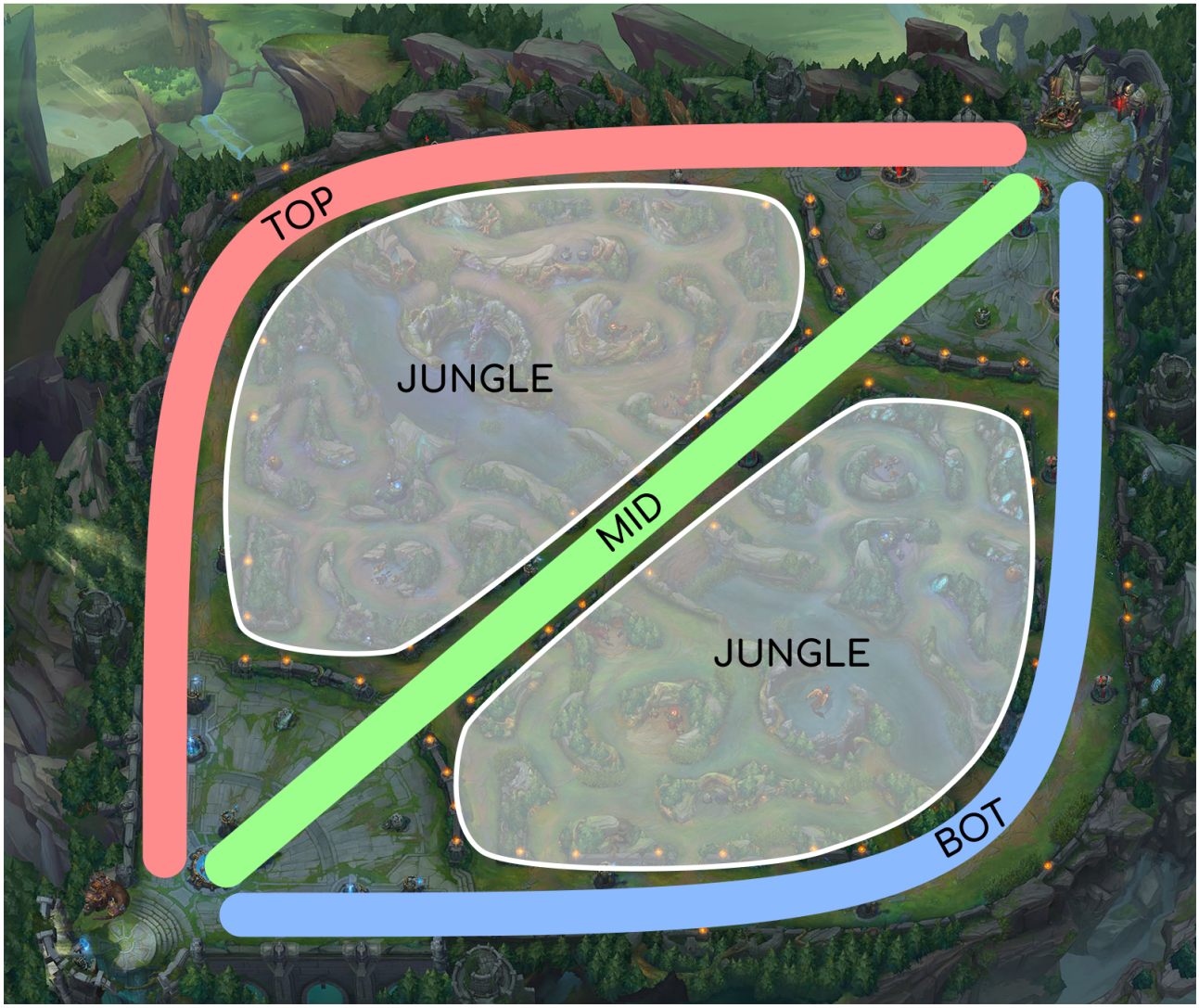


Figura 2.2: Carriles League of Legends. Fuente: recurso propio

Estos carriles y jungla sirven también para definir cada uno de los diferentes roles que los jugadores asumen al comenzar una partida y suelen mantenerse en estas posiciones hasta cierto minuto en la partida, cuando comienzan a agruparse de diferente manera en base a la situación, estos han sido establecidos a través de los años y se dividen de la siguiente manera, a la línea superior de TOP acude un único jugador por equipo al igual que en el MID, en la línea de BOT suelen acudir dos jugadores, siendo uno de estos el que suele aportar el mayor daño al equipo y otro el cual tiene un rol de soporte hacia su compañero de línea, por último, en la jungla acude un jugador con la función principal de eliminar los monstruos que allí se encuentran y aportar soporte a cada uno de los carriles cuando es necesario.

De manera regular se generan pequeños monstruos llamados súbditos los cuales defienden su respectiva base desde cada uno de los nexos los cuales avanzas a través de los carriles hacia la base enemiga.

Los jugadores deben realizar estrategias utilizando sus campeones, cooperando entre ellos junto con los súbditos para poder obtener ventajas y finalmente destruir el nexo enemigo.

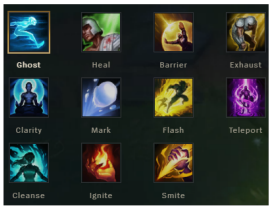
En lo que respecta al desarrollo de una partida hay un proceso el cual se repite y podríamos diferenciar en dos fases.

La primera fase se llama la selección de campeón, esta ocurre por turnos, donde los jugadores una vez emparejados en equipos de 5 seleccionan por turnos cada uno de los campeones que jugarán en esa partida, además de elegir dos habilidades llamadas hechizos de invocador los cuales son independientes del campeón seleccionado y una combinación de runas, las cuales son mejoras con las que cada jugador contará al comienzo de la partida.

La segunda fase es la partida en sí mismo, en esta los jugadores una vez han seleccionado sus campeones con sus respectivos hechizos de invocador y runas (ver Figura 2.3), aparecen en su base con una pequeña cantidad de oro y la posibilidad de desbloquear una de las 4 habilidades con las que cuenta su campeón, una vez salgan de la base, cada jugador acudirá a su respectivo carril a defender su base y obtener ventaja para así destruir la base enemiga.

Runas

Hechizos de Invocador



Objetos

Figura 2.3: Objetos, runas y Hechizos de Invocador. Fuente: cliente de escritorio de League of Legends

El juego cuenta con actualizaciones constantes las cuales suelen ocurrir cada 2 semanas en las que se introducen nuevos campeones, cosméticos como aspectos nuevos para ciertos campeones, cambios en mecánicas de juego, ajustes en campeones, objetos y runas (ver Figura 2.4).

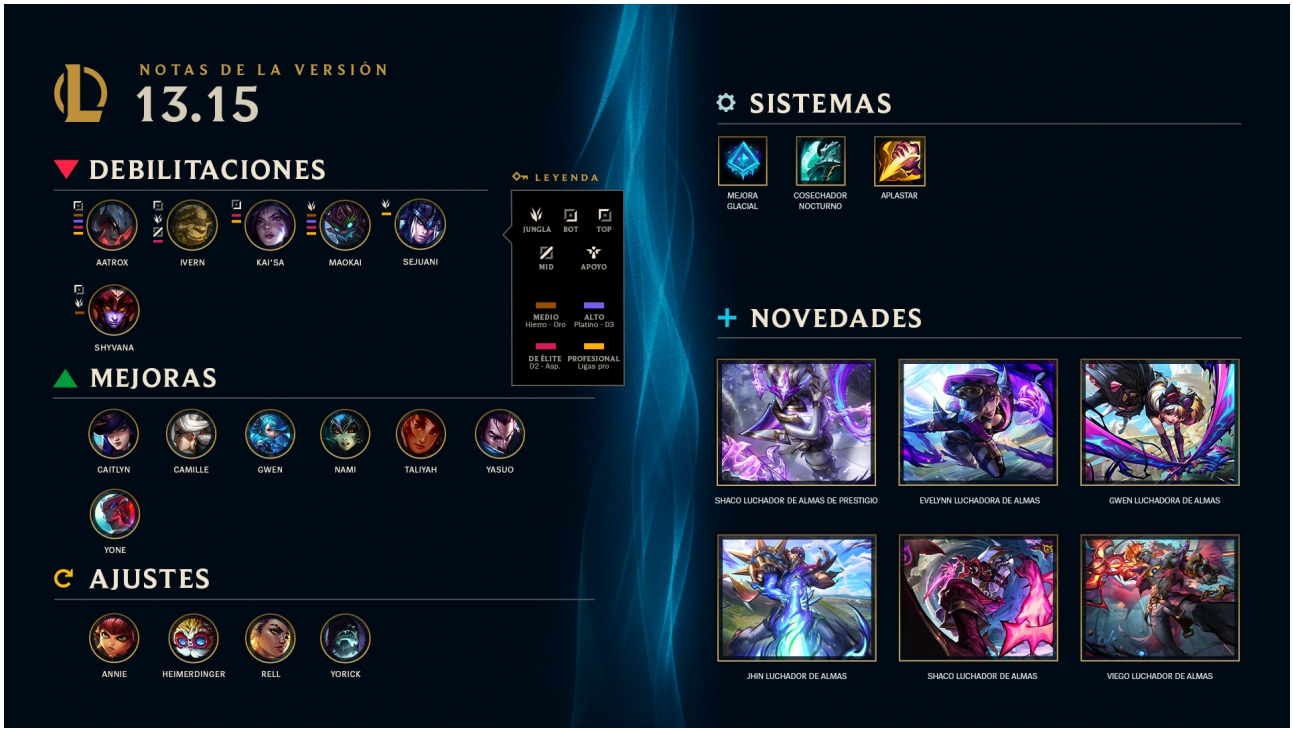


Figura 2.4: Notas de parche. Fuente: `https://www.leagueoflegends.com/en-gb/news/tags/patch-notes/`

2.3. League of Legends como Esport

Desde su lanzamiento en 2009, League of Legends fue diseñado con la intención de fomentar la competición y el juego en equipo, lo que ayudó a su adaptación como esport. A medida que ganaba popularidad, Riot Games, la compañía desarrolladora comenzó a organizar torneos y competiciones a nivel regional y mundial. Atrayendo jugadores y equipos profesionales de todo el mundo.

Un punto de inflexión para el League of Legends como Esport ocurrió en la segunda temporada de League of Legends, específicamente con el campeonato mundial. Este torneo, celebrado en octubre de 2012, fue el primer campeonato mundial de League of Legends a gran escala, donde se repartieron 2 millones de dólares en premios (7). En este, los mejores equipos de Europa, Estados Unidos, China y Corea del Sur lucharon por alzarse con el título de campeones mundiales.

Esta final marcó un hito importante en la historia de League of Legends como Esport, ya que alcanzó un pico de 1.1 millones de espectadores únicamente en la final (8), ayudando a que este se convirtiera en uno de los deportes electrónicos más populares y exitosos del mundo.

Después del mundial de 2012, League of Legends siguió creciendo como esport de renombre mundial. El éxito del mundial de la temporada 2 aumentó la popularidad del juego y atrajo a una base de jugadores más amplia, llegando a aumentar la base de jugadores activos por más de 8 llegando a los 12 millones solo en 2012 (7), lo que llevó a Riot Games a expandir y consolidar un ecosistema competitivo del juego.

A partir de entonces, Riot Games estableció un sistema de ligas regionales por todo el mundo, como la League of Legends Championship Series (LCS) en América del Norte y Europa (actualmente LEC), la League of Legends Pro League (LPL) en China y la League of Legends Champions Korea (LCK) en Corea del Sur (ver Figura 2.5). Estas ligas profesionales ofrecían a los equipos un marco de competiciones regulares y la oportunidad de clasificarse a torneos más prestigiosos como es el campeonato mundial.

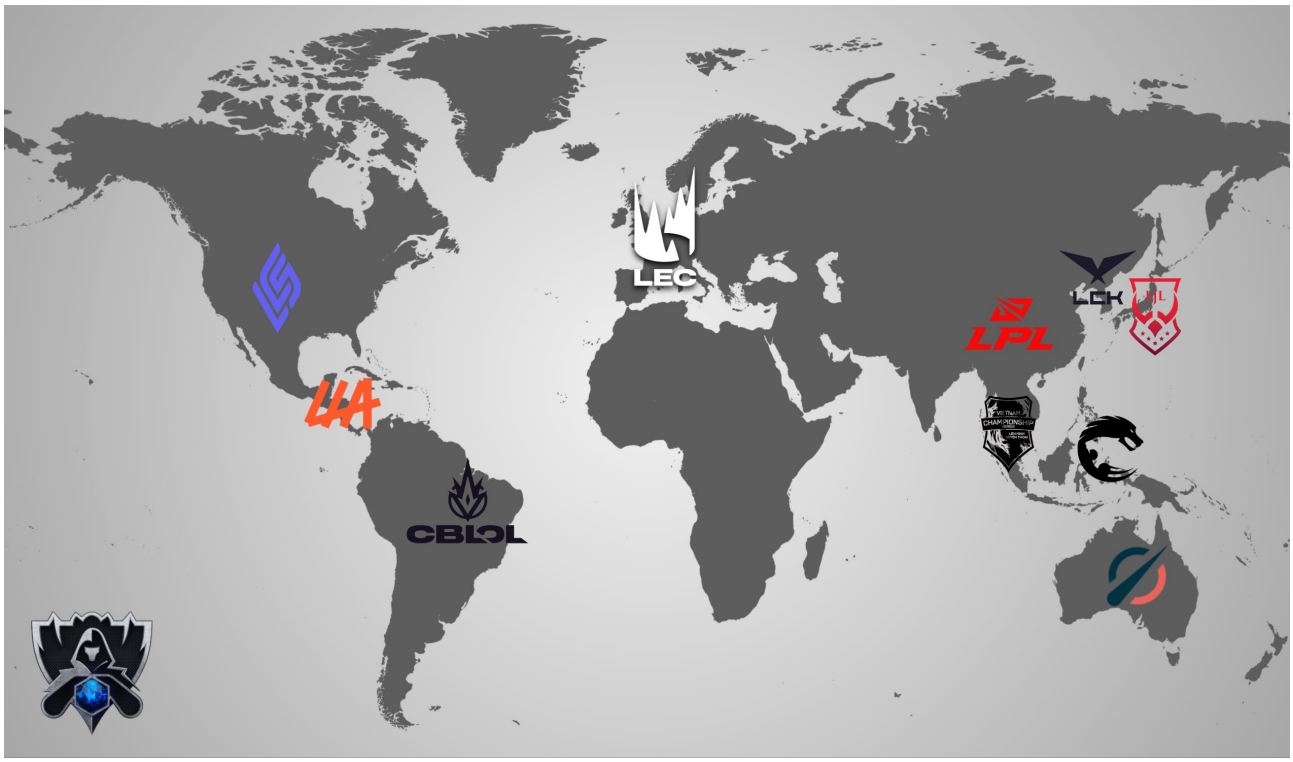


Figura 2.5: Ligas Internacionales. Fuente: `https://www.reddit.com/r/leagueoflegends/comments/djumtq/map_logos_the_120_teams_in_the_lol_esport/`

Para poder acceder a estas ligas las cuales ofrecían acceso a la escena internacional, se crearon también ligas regionales, en el caso de Europa se crearon las ERL (European Regional Leagues) (ver Figura 2.6), estas ligas regionales, cuentan con un ecosistema de ligas semi-profesionales y amateur mediante las cuales cualquier organización puede competir para conseguir una plaza en una de estas ERL.

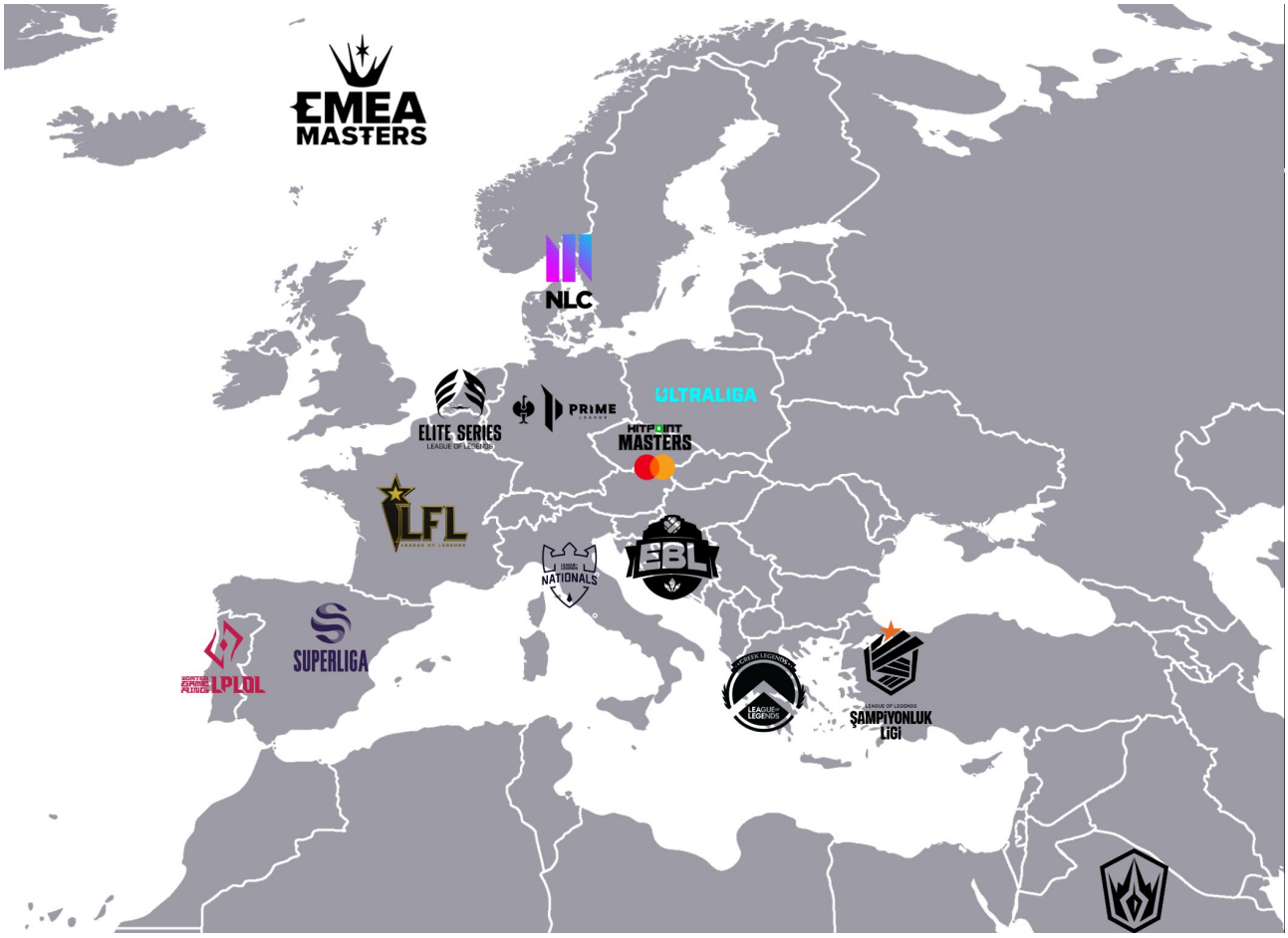


Figura 2.6: Ligas Regionales Europeas. Fuente: `https://www.reddit.com/r/leagueoflegends/comments/djumtq/map_logos_the_120_teams_in_the_lol_esport/`

Como conclusión, los esports son un sector que mueve millones de personas tanto a nivel profesional como amateur, y que el uso de herramientas para la mejora del rendimiento y funcionamiento de los equipos profesionales, consigue fomentar el mejor desarrollo de la competición.

2.4. Plataformas para League of Legends

En los últimos años, numerosas plataformas han surgido con el objetivo de ayudar a jugadores a mejorar su nivel dentro del juego y entender mejor el funcionamiento de este. Este tipo de herramientas surgen tanto para jugadores amateur como profesionales con el objetivo de ayudar a mejorar a los jugadores. En este apartado analizaremos este tipo de herramientas enfocadas a League of Legends.

Existen diversas plataformas enfocadas en League of Legends, podríamos diferenciar en dos categorías:

1. Plataformas acompañantes para mejorar individualmente
2. Plataformas para League of Legends competitivo

Las plataformas acompañantes se centran en ayudar a mejorar de manera individual a un jugador, mostrándole a este los errores que comete y en qué aspectos de la partida debe enfocarse en base al rol y campeón que juega, esto lo consiguen mediante plataformas que muestran el rendimiento del jugador una vez la partida ha terminado y tambien a traves de software que el jugador instala y que genera recomendaciones y observaciones para mejorar en tiempo real. Todos estas recomendaciones y observaciones, además de los análisis de rendimiento se realizan comparando al jugador con otros jugadores de mayor nivel o profesionales y haciendo análisis y obteniendo insights en los datos mediantes análisis de datos y modelos estadísticos (ver Figura 2.7).

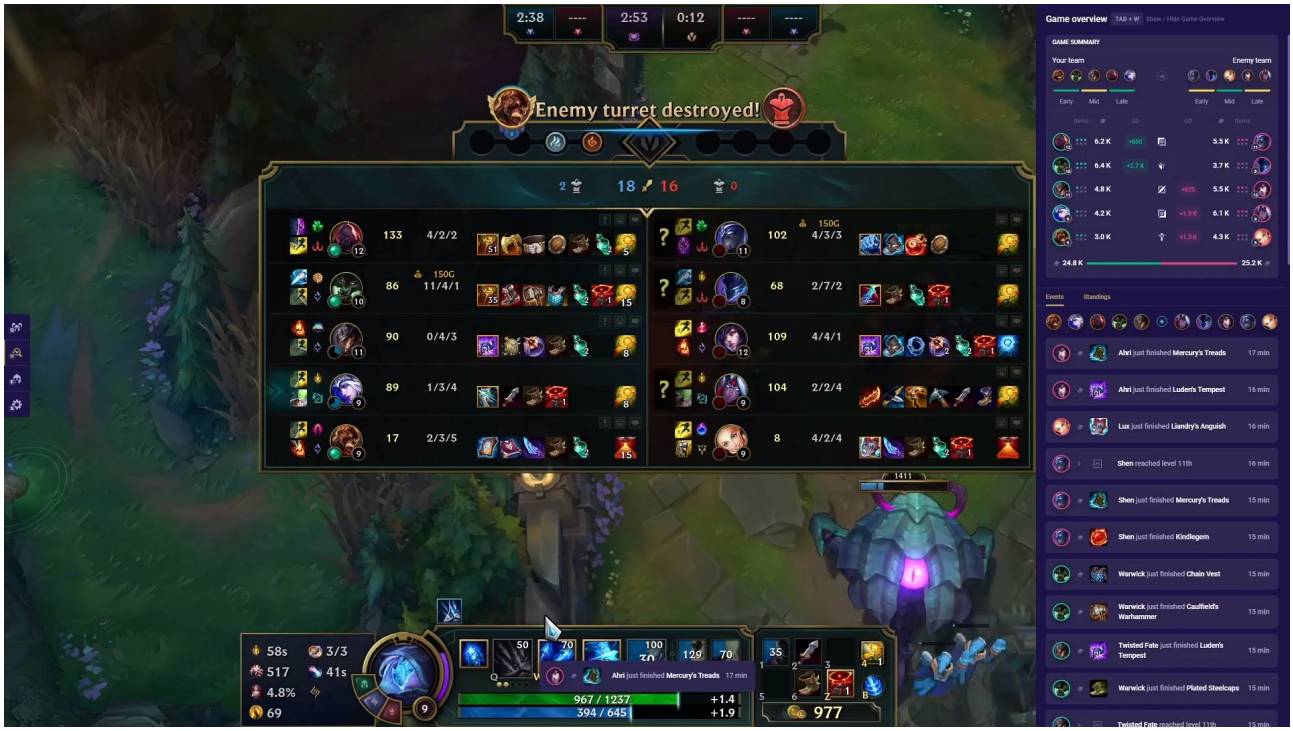


Figura 2.7: Recomendaciones en partida mobalytics. Fuente: `https://mobalytics.gg/`

Ejemplos de este tipo de plataformas serían Mobalytics(31) , op.gg(32) y u.gg(33).

Por otro lado, existen las plataformas para League of Legends competitivo, estas se enfocan en mostrar los datos de las partidas que se juegan en las competiciones profesionales y poder llegar a hacer un análisis mucho más detallado y personalizado, dentro de este tipo de plataformas que suelen usarse para consultar el rendimiento de equipos y jugadores profesionales podemos diferenciar entre dos tipos, las cuales son usadas por parte de aficionados y las que hacen uso equipos profesionales y comentaristas. Ejemplos de plataformas para todos los públicos son Factorgg (9), Leaguepedia (10) y Games of Legends (14) entre otras.

En los últimos años podemos ver el crecimiento de estas plataformas para competitivo, estas añaden funcionalidades para llevar un seguimiento de entrenamientos además del seguimiento más preciso de equipos, de manera que proporcionan a los equipos con más información para preparar partidos y analizar su rendimiento en entrenamientos.

Ejemplos de este tipo de plataformas serían DeepLol Pro (15) y Shadowgg (16), las cuales ofrecen tanto datos de partidas competitivas como datos de las partidas que juega el equipo de manera privada contra otros equipos para entrenar.

Todo este tipo de herramientas ofrecen a los jugadores y equipos una manera en la que analizar el juego y poder mejorar, pero tienen algo en común, y esto es que se trata de herramientas únicamente para consultar datos, en ninguna ofrecen un planteamiento mediante el cual se haga uso de una IA para ofrecer algo que no solo ayude a consultar, sino que ofrezcan posibles soluciones a problemas como es el caso de la selección de campeones, de ahí lo que plantea este trabajo.

2.5. Algoritmos para sistema de recomendación

Para el desarrollo de un correcto modelo de clustering con un gran conjunto de atributos, se recomienda realizar una reducción de dimensionalidad del conjunto de datos previamente a aplicar el clustering, dado que esto puede ser beneficioso por varias razones como son:

1. Maldición de la dimensionalidad: A medida que aumenta el número de dimensiones, los datos se vuelven más dispersos, lo que dificulta encontrar patrones o agrupaciones.

2. Mejora de la eficiencia: Al reducir la dimensionalidad, se puede reducir significativamente la complejidad computacional y mejorar la eficiencia del proceso de agrupamiento.

3. Reducción del ruido: La existencia de datos irrelevantes o ruidosos puede obstaculizar el proceso de agrupamiento, el uso de técnicas de reducción de dimensionalidad permite que el algoritmo de agrupamiento se centre en los aspectos más relevantes de los datos.

4. Evitar el Overfitting: Al contar con un alto número de características puede generar el sobreajuste de los grupos haciendo que estos tengan un rendimiento deficiente al ajustarse de manera extremadamente detallada a los datos, al aplicar reducción de dimensionalidad ayudamos a reducir el riesgo extrayendo la información más importante y eliminando características redundantes.

Un serie de técnicas que se utilizan para esta reducción de dimensionalidad son, el análisis de componentes principales (PCA)(17), la incrustación estocástica de vecinos t-Distribuidos (t-SNE)(19) y la aproximación uniforme de variedad (UMAP)(18).

El Análisis de Componentes Principales (PCA) tiene como objetivo transformar un conjunto de variables correlacionadas en un nuevo conjunto de variables no correlacionadas llamadas componentes principales que capturan la mayor parte de la variabilidad de los datos.

La Incrustación Estocástica de vecinos t-Distribuidos (t-SNE) busca transformar los puntos de datos que tenemos en una alta dimensión a una de menor dimensión preservando las relaciones entre estos puntos, para esto construye distribuciones de probabilidad para medir las similitudes

entre los puntos en el espacio original y proyectarlos en el espacio de menor dimensión minimizando la divergencia entre las distribuciones.

Por último, la Aproximación Uniforme de Variedad (UMAP) se trata de un algoritmo similar a t-SNE, aunque difiere en ciertos puntos como son que t-SNE hace uso de distribuciones de probabilidad para medir las similitudes entre datos, mientras que UMAP utiliza un marco matemático para preservar tanto estructuras locales como globales, este también ofrece más flexibilidad en la incrustación, debido a que t-SNE puede llegar a no gestionar tan bien el uso de variables categóricas. Por otro lado, UMAP cuenta también con una ventaja a la hora de gestionar conjuntos de datos más grandes de manera más escalable y eficiente.

A partir de esta reducción de dimensionalidad se pueden plantear diversos algoritmos de clustering. Algoritmos de clustering planteados pueden ser K-Means(20), OPTICS(21) y DBSCAN(22).

K-Means es un algoritmo de clustering que busca dividir un conjunto de datos en K grupos o clusters. Utiliza la distancia entre puntos para asignar cada punto al grupo cuyo centroide (promedio) está más cercano.

DBSCAN es un algoritmo de clustering basado en la densidad que agrupa puntos cercanos en clusters y puede identificar regiones de baja densidad como ruido.

OPTICS es un algoritmo de clustering que permite identificar la estructura de los clusters en conjuntos de datos más complejos. Calcula las relaciones de distancia entre puntos conectados por densidad y crea un gráfico de alcance que revela cómo están conectados.

También cabe mencionar la técnica utilizada para la selección de la mejor recomendación basada en frecuencia condicional con búsqueda exhaustiva, en esta se establece un orden de prioridad para las elecciones, y se selecciona la opción más repetida de acuerdo a este orden.

Capítulo 3

Propuesta

En el siguiente capítulo se plantea el diseño de la propuesta de este trabajo de fin de grado. Primero, para definir la solución, se consultó a una serie de 3 analistas de League of Legends, los cuales están encargados de proporcionar a entrenadores con toda la información necesaria para la preparación de partidos, por lo cual pueden ofrecer un punto de vista de las necesidades de los entrenadores.

3.1. Análisis de requisitos

En esta sección se hablará de cada uno de los requisitos planteados tras realizar una serie de 3 reuniones con los analistas de SK Gaming en LEC (equipo de la primera división europea) y TSM en LCS (equipo de primera división estadounidense) donde se consultó cómo se podría ayudar a la preparación de partidos.

Posterior a la primera reunión se concluyó que la sección donde más se podría ayudar en la preparación de partidos es la selección de campeones. Para ello, se plantearon diversos requerimientos para la creación de una solución. Estos fueron divididos en historias de usuarios las cuales podemos observar en la siguiente tabla (ver Tabla 3.1).

Cuadro 3.1: Historias de usuario

Historia de usuario	Detalles
Como entrenador, quiero realizar simulaciones de selección de campeón.	La herramienta deberá simular una selección de campeón.
Como entrenador, quiero tener flexibilidad a la hora de detallar cada uno de los campeones.	La herramienta deberá dejar seleccionar el rol y campeón en cada turno de la simulación.
Como entrenador, deseo recibir recomendaciones en base a la simulación planteada.	La herramienta deberá ofrecer la opción de recibir recomendaciones en cualquier momento de la simulación.
Como entrenador, quiero recibir recomendaciones que utilicen información actualizada del juego.	Todas las recomendaciones deberán basarse en partidas extraídas con los últimos cambios que haya recibido el juego.

3.2. Arquitectura de la solución

Una vez enumerados todos los requisitos, se decidió que la mejor implementación para que esta solución sea sencilla de usar por entrenadores además de poder realizar actualizaciones regulares sería mediante la creación de una aplicación web a través de la cual se utilizará el modelo creado mediante diferentes técnicas de machine learning para realizar las recomendaciones, para esto se planteó el siguiente esquema de la solución (ver Figura 3.1).

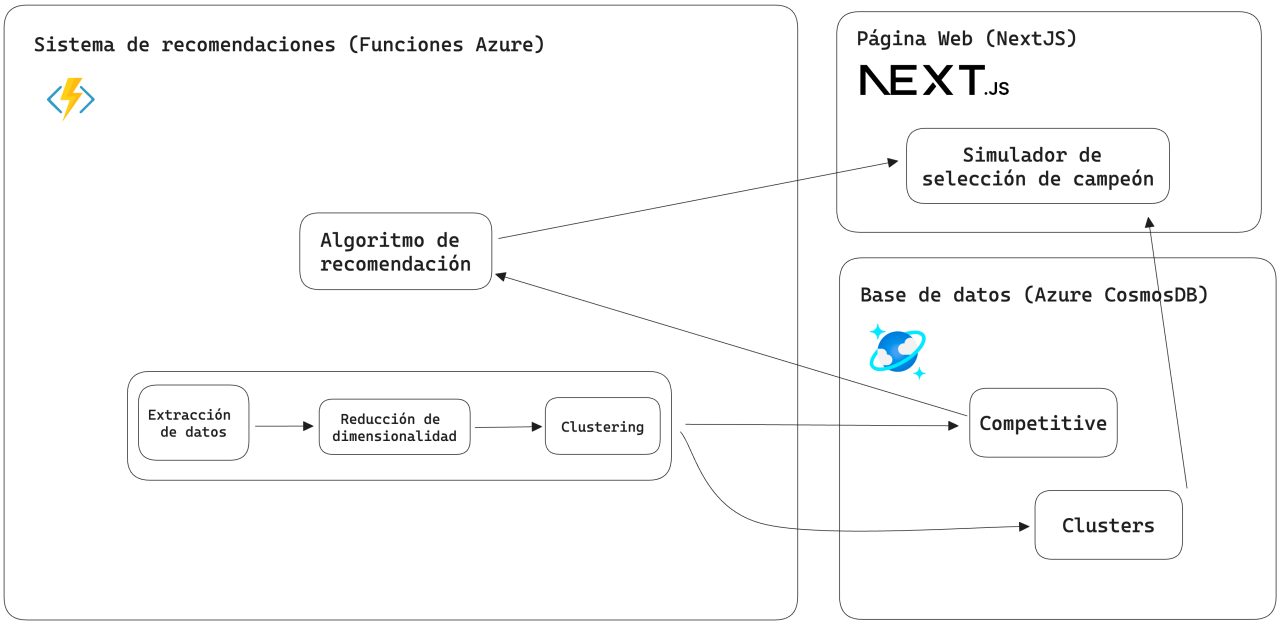


Figura 3.1: Arquitectura de la solución. Fuente: recurso propio

En la arquitectura se pueden observar 3 elementos, por un lado el **Sistema de recomendacio-**

nes se trata de las funciones de extracción y procesado de datos y el algoritmo de recomendación utilizado.

La **Aplicación Web** se trata de la web la cual se utilizará para poder hacer uso del algoritmo de recomendación de una manera cómoda y sencilla.

Por último, la **Base de datos** es donde almacenaremos los dos conjuntos de datos, Competitive será donde se guarden los datos históricos con los clusters ya aplicados a cada uno de los campeones y Clusters donde se guarden las traducciones existentes entre cada campeón y el cluster al que pertenecen.

Arquitectura aplicación web

Una vez decidido que la solución se visualizará mediante una aplicación web, el siguiente paso es plantear que arquitectura seguirá la misma, desde el servidor donde se realizan los procesos de extracción y preparación de datos, la inserción y extracción de estos en una base de datos, hasta el cliente web el cual consumirá los datos de la base de datos conectándose al servidor.

Debido a que nuestra aplicación únicamente consumirá datos desde el cliente para la petición de recomendaciones, se ha decidido hacer uso de una arquitectura sin servidor, de manera que se evitan los problemas de la creación de una infraestructura compleja, podemos desarrollar de manera rápida y reducir costes ya que solo se hace uso de recursos en el momento que se necesitan.

En la siguiente figura se muestra el planteamiento de aplicación web empleado para este proyecto (ver Figura 3.2). Para el cliente se ha realizado una implementación mediante Nextjs, alojándo este en Vercel.

Para la base de datos se ha usado Cosmos DB en Azure, y siguiendo con el servicio en la nube de Microsoft se ha implementado funciones las cuales son llamadas desde el cliente para obtener recomendaciones de campeones.

Por último, la base de datos es actualizada de manera regular mediante funciones sin servidor alojadas en el mismo servicio de Azure Serverless Functions que se ejecutan diariamente para actualizar los datos usados para las recomendaciones.

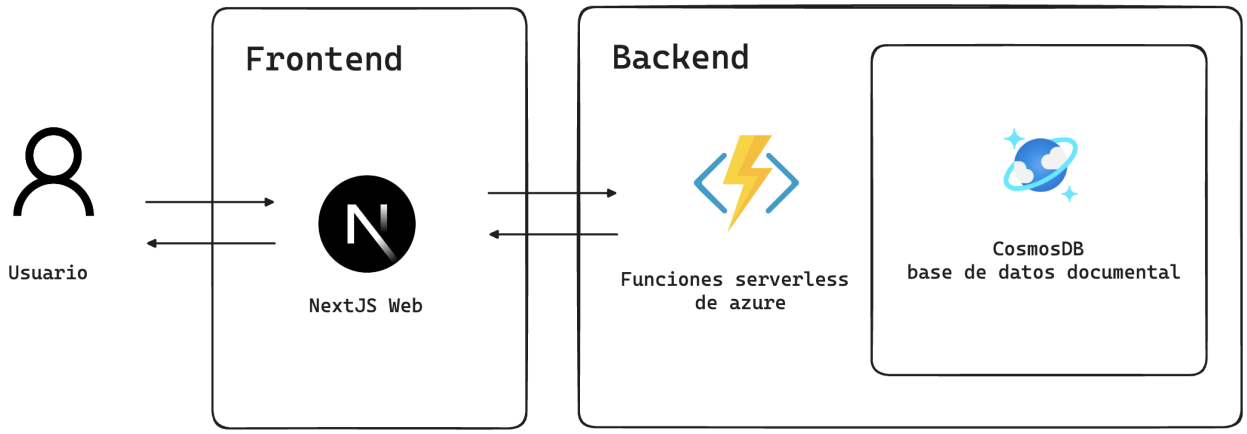


Figura 3.2: Arquitectura de la página web. Fuente: recurso propio

3.3. Sistema de recomendación de campeones

Una de las principales metas de la solución planteada es contar con un buen algoritmo para recomendación de campeones, para ello se realizará primero una extracción de datos, posteriormente una reducción de dimensionalidad junto con un clustering, seguido de un algoritmo para recomendaciones (ver Figura 3.3).

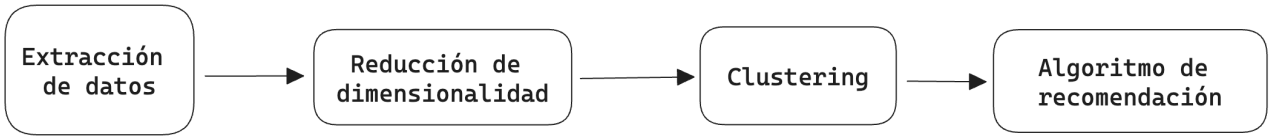


Figura 3.3: Sistema de recomendación de campeones. Fuente: recurso propio

Extracción de datos

La primera necesidad para poder realizar las recomendaciones es contar con un conjunto de datos suficientemente grande como para poder contar con información de todos los campeones, para esto se plantea el uso de un conjunto de datos de cola en solitario, del cual se extraen las estadísticas que nos ofrece la API pública de Riot Games.

Una vez extraída esta información, se procede a la extracción del segundo conjunto de datos, este de partidas profesionales, de este conjunto únicamente se extraen los datos de la selección de campeones y el orden de la selección de estos.

Reducción de Dimensionalidad y Clustering

Una vez extraídos los dos conjuntos, se utiliza el conjunto de cola en solitario donde contamos con más partidas para realizar primero una reducción de dimensionalidad para reducir el número

de atributos con los que trabajamos, y un clustering, de manera que podemos utilizar los clusters obtenidos en el segundo conjunto de datos en el cual haremos nuestras recomendaciones.

Algoritmo de recomendación

Por último, una vez que contamos con el conjunto de datos preparado, hacemos uso del algoritmo de la frecuencia condicional con búsqueda exhaustiva para tratar de encontrar la selección de campeón más parecida a la entrada que tenga nuestro sistema de recomendación de campeones.

3.4. Diseño de la interfaz de usuario

Para la creación de cada una de las vistas de la aplicación web, después de haber definido cada uno de los requisitos, se procedió a hacer uso de la herramienta Figma para la creación de cada una de las vistas, de manera que primero se hacía el diseño en Figma para posteriormente presentar los diseños a los clientes, obtener feedback y así realizar los cambios/mejoras oportunas. Una vez validado todo el diseño, se ponía en marcha la fase de implementación de los diseños.

El diseño se basa en el ya existente para la selección de campeón de League of Legends (ver Figura 3.4).

Figura 3.4: Captura de selección de campeones en League of Legends. Fuente: cliente de escritorio de League of Legends

Se realizó un diseño de la vista inicial, en esta se muestra con la misma estructura que en la fase de selección de campeón de League of Legends cada uno de los campeones a seleccionar y

bloquear, mostrando a la izquierda los correspondientes al equipo azul y a la derecha los del equipo rojo (ver Figura 3.5).

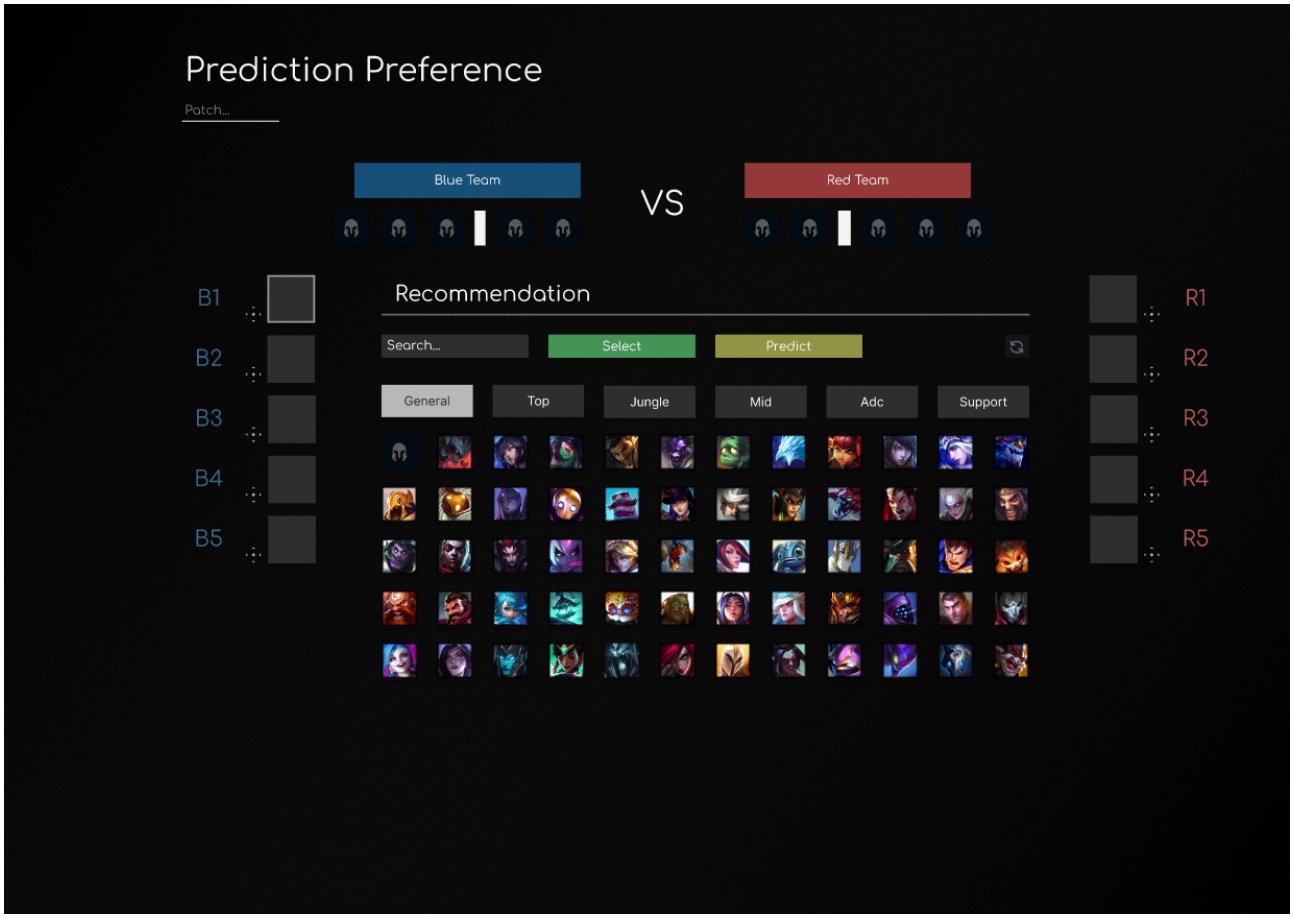


Figura 3.5: Interfaz de usuario página inicial en figma. Fuente: recurso propio

Primero podemos observar en la esquina superior izquierda un campo editable ”patch...”, este mostrará el parche del juego, es decir, en qué versión del juego se realiza la predicción, automáticamente seleccionará la última versión disponible, aunque puede ser editado a versiones antiguas.

En la zona central se puede observar una serie de botones además de una serie de fichas de campeones las cuales son clicables.

La primera fila de botones muestra un campo de texto ”Search....[el] cual filtra las fichas de campeones mostrando únicamente las que contengan la cadena de texto en su nombre. A continuación aparecen botones de ”Select”para seleccionar el campeón previamente clicado, el botón de ”Predict.[el] cual puede ser usado en cualquier momento y calculará cuál es la mejor recomendación de campeones y los mostrará justo debajo del texto Recommendationçomo se muestra en la segunda figura. Al fondo se muestra un botón con un icono de dos flechas rotando, este puede usarse en cualquier momento y reiniciará el estado de la simulación.

La segunda fila de botones muestra cada uno de los roles además de uno previamente seleccionado llamado ”General”. Estos botones sirven para hacer una selección de campeón para un rol

específico o, en el caso de usar la opción ”General”hacer una selección sin rol definido.

Una vez se ha dado click en el botón de ”Predict”, la vista añadirá una lista de campeones justo debajo del título Recommendation”, esta lista de campeones se actualizará cada vez que se realice un cambio en la simulación y se de click en el botón, a continuación se muestra la vista al dar click al botón de ”Predict”(ver Figura 3.6).

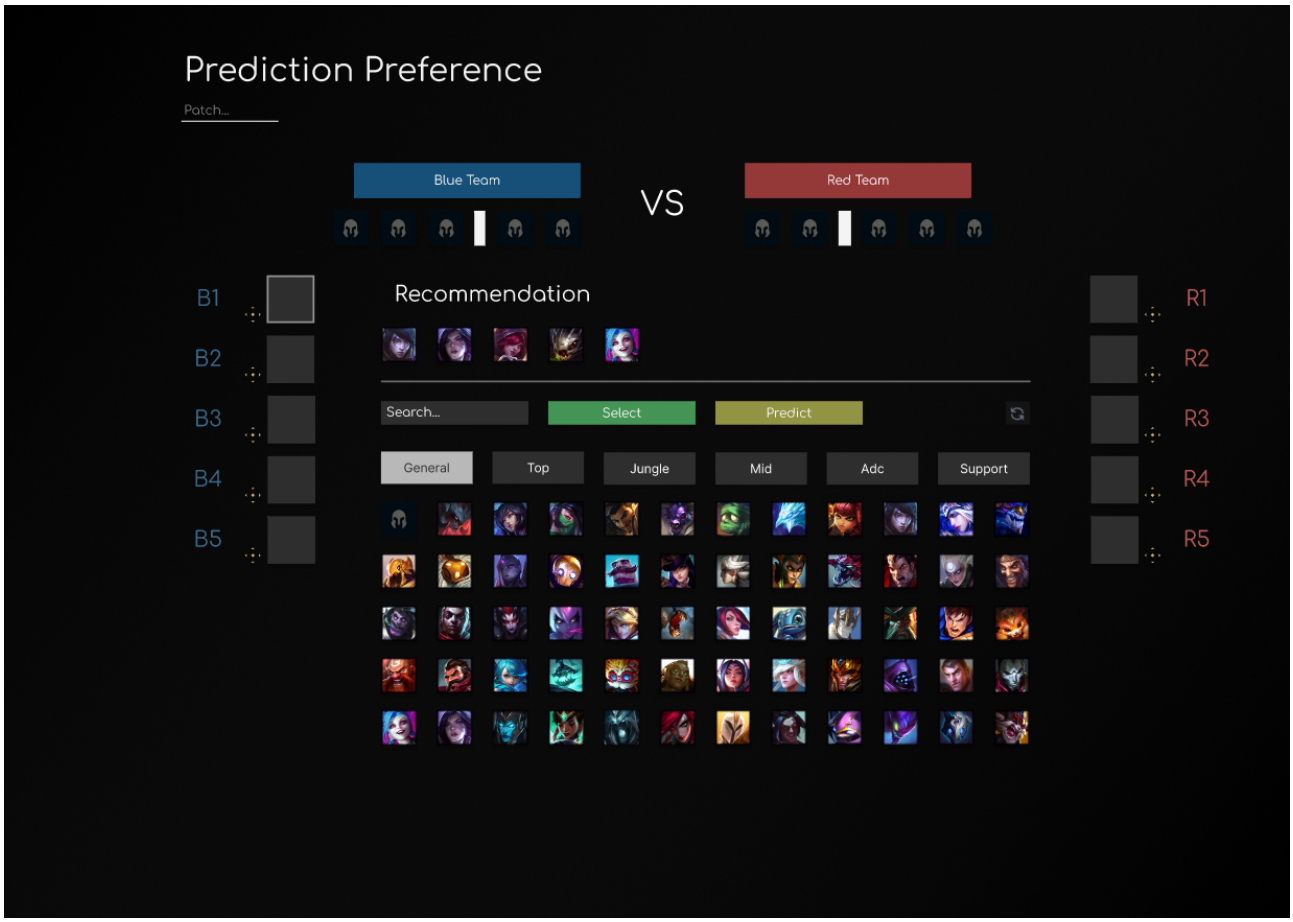


Figura 3.6: Interfaz de usuario página con recomendación en figma. Fuente: recurso propio

Por último, comentar que la simulación sigue el mismo orden de selección de campeón, este va alternando de manera un tanto irregular en selecciones del equipo rojo y el equipo azul, en la siguiente figura se muestra cual es el orden (ver Figura 3.7).

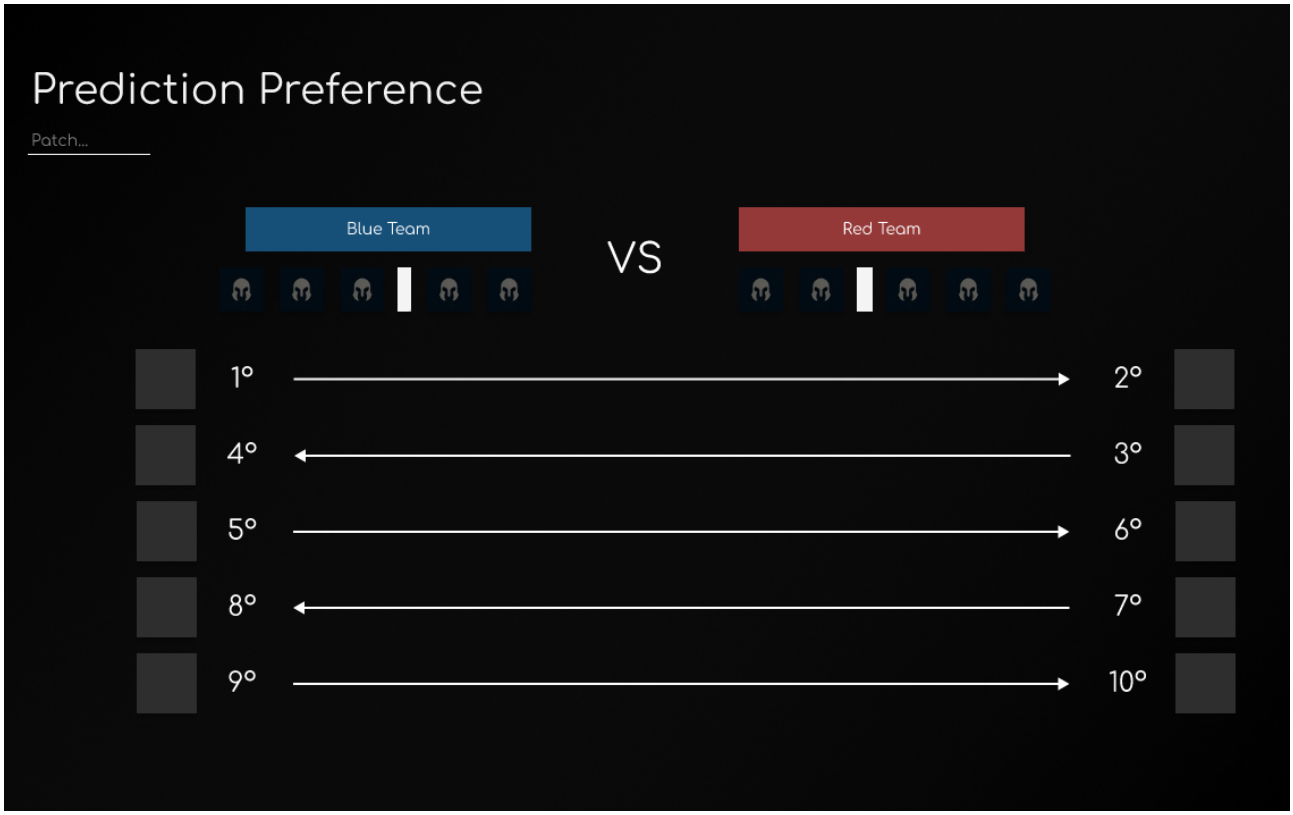


Figura 3.7: Orden de selección de campeón. Fuente: recurso propio

Capítulo 4

Implementación del sistema de recomendaciones

En esta sección se explica la extracción y análisis realizado para obtener el conjunto de datos y el algoritmo de recomendaciones el cual se utiliza posteriormente, previamente a las recomendaciones se ha realizado un clustering para realizar las recomendaciones en base a grupos de campeones y no a campeones individuales debido a que no se contaba con datos suficiente de cada uno de los campeones.

4.1. Conjuntos de datos

El conjunto de datos planteado tiene que consistir en uno el cual cuente con información de la selección de campeones manteniendo el orden de esta, por lo que se ha optado por extraer datos de partidas de competición profesionales. 1281 partidas del año 2023 fueron extraídas de la wiki de Esports de League of Legends(10) haciendo uso del api pública mediante la cual se hacen consultas con sintaxis similar a SQL a las bases de datos Cargo que ofrecen desde la wiki (34), para esto se siguieron los siguientes pasos:

Primero se extraen las ligas que cumplen las condiciones que buscamos y los identificadores para cada uno de los campeones para poder traducir los nombres de los campeones en ids para evitar problemas.

```
site = mwclient.Site('lol.fandom.com',path='/')

leagues_query = site.api('cargoquery',
    limit='max',
    tables='Tournaments=T',
    fields='T.Name, T.OverviewPage, T.League, T.Year, T.LeagueIconKey, T.Region,
        T.TournamentLevel, T.IsOfficial, T.DateStart',
```

```
    where="Year = 2023 AND IsOfficial = 1 AND TournamentLevel = 'Primary' AND
        DateStart <= '" + current_date + "'")


champions_response = site.api('cargoquery',
    limit = 'max',
    tables = "Champions=C",
    fields = "C.Name , C.KeyInteger"
)
```

Una vez teniendo las ligas, procedemos a recorrer cada una de las ligas e iterar para obtener las partidas de todo 2023, al hacer esto, primero buscamos los parches de cada una de las partidas para así poder identificar las partidas.

```
patch_response = site.api('cargoquery',
    limit = 'max',
    tables = "Tournaments=T, ScoreboardTeams=ST, ScoreboardGames=SG",
    join_on = "ST.OverviewPage = T.OverviewPage, ST.GameId = SG.GameId",
    fields = "ST.GameId, SG.Patch, SG.DateTime_UTC",
    where = "SG.DateTime_UTC > '" + str(last_date) + "' AND T.Name = '" + league +
        "'",
    order_by = "SG.DateTime_UTC"
)
```

Posteriormente extraemos información de cada jugador recorriendo las partidas existentes en cada liga, para esto se hace uso de la siguiente llamada.

```
players_response = site.api('cargoquery',
    limit = 'max',
    tables = "Tournaments=T, ScoreboardPlayers=SP",
    join_on = "SP.OverviewPage = T.OverviewPage",
    fields = "SP.OverviewPage, SP.GameTeamId, SP.GameId, SP.DateTime_UTC, SP.Link,
        SP.PlayerWin, SP.Champion, SP.Role, SP.Side , SP.Runes, SP.Team, SP.TeamVs,
        SP.KeystoneMastery, SP.KeystoneRune, SP.PrimaryTree, SP.SecondaryTree,
        SP.Items, SP.Trinket, SP.SummonerSpells",
    where = "SP.DateTime_UTC > '" + str(last_date) + "' AND T.name='" + str(league)
        + "'",
    order_by = "SP.DateTime_UTC"
)
```

Y por último, recorremos las partidas sin duplicados, para esto primero crearemos un DataFrame mediante la libreria de Pandas de Python para así obtener una lista de los identificadores de partida sin duplicados.

```
total_games = [ game['title'] for game in players_response]
league_games_df = pd.DataFrame(total_games)
unique_league_games = pd.unique(league_games_df['GameId'])
```

Una vez tenemos esta lista, la recorreremos y obtendremos todas las estadísticas que queramos de todas estas partidas, las cuales podemos encontrar en los parametros de PJM.StatsPage y PJM.TimelinePage.

```
match_response = site.api(
    action = 'cargoquery',
    limit = 'max',
    tables = "MatchScheduleGame=MSG, PostgameJsonMetadata=PJM",
    fields = "MSG.RiotPlatformGameId, MSG.GameId, MSG.Blue, MSG.Red, PJM.StatsPage,
        PJM.TimelinePage",
    where= "MSG.GameId='" + game_id + "'",
    join_on = "MSG.RiotPlatformGameId = PJM.RiotPlatformGameId"
)
```

Para una descripción más detallada de este proceso, todo el código se encuentra en el repositorio con la extracción de datos disponible en `https://github.com/CarlosCanut/tfg_recommender/tree/main`.

Después de extraer los datos de cada uno de los campeones de las 1281 partidas se procedió a analizar el total de campeones de los cuales se había obtenido información para el posterior clustering. Aunque se contó con partidas de 140 campeones, de 44 de los campeones no se contaba ni siquiera con información de más de 10 partidas, lo cual dificultaría la creación de clusters precisos. Debido a tener tan solo datos suficientes del 58,9 % del total de campeones existentes, se planteó utilizar un procedimiento para tratar de abarcar el mayor número de campeones dentro de este conjunto de datos competitivos.

La solución planteada es agrupar cada uno de los campeones en diferentes grupos, de manera que las recomendaciones se harán en base a los grupos de campeones y no a cada campeón de manera individual (ver Figura 4.1).

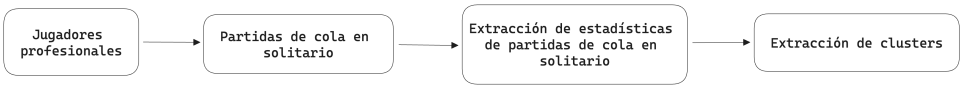


Figura 4.1: Orden de pasos para extracción de clusters. Fuente: recurso propio

Para poder encontrar datos de cada uno de los campeones se hizo uso de los datos de partidas de jugadores del más alto nivel en la cola de emparejamiento en solitario, esta cola de emparejamiento es la que utilizan los jugadores para emparejarse con gente de un nivel similar y poder practicar contra jugadores de todo el continente, los jugadores seleccionados para obtener datos de sus partidas han sido todos los que se encuentran compitiendo en la temporada 2023 en la liga aspirante en los servidores Europeo y Coreano.

Para la extracción de datos se hizo uso de la API pública que Riot Games ofrece para obtener datos de cada partida que se juegue en un servidor público, como es el caso de la cola de emparejamiento en solitario. De esta se obtuvieron datos de 32625 partidas de los mejores jugadores de los servidores de Europa y Corea.

Para la extracción de datos mediante la API pública de Riot Games se siguió el siguiente proceso:

Primero se obtienen los jugadores de la liga aspirante.

```
challenger_league = watcher.league.challenger_by_queue(region['id'],
    "RANKED_SOLO_5x5")
challenger_players = [entry['summonerName'] for entry in
    challenger_league['entries']]
```

Una vez tenemos a estos jugadores, procedemos a extraer el historial de las últimas 100 partidas de cada uno de estos.

```
watcher.match.matchlist_by_puuid(region, puuid, count=100)
```

Una vez tenemos estas partidas, extraemos los datos de estas partidas, para esto primero se obtienen los objetos que nos ofrece la api, por un lado un objeto con los datos finales de partida, y otro con los datos minuto a minuto de cada partida. Una vez tenemos estos datos, obtenemos las estadísticas que buscabamos.

```
game_data = watcher.match.by_id(region, game_id)
timeline_data = watcher.match.timeline_by_match(region, game_id)
game_stats = get_postgame_data(game_id, game_data, timeline_data)
game_region_stats = game_region_stats + game_stats
```

Una vez almacenadas todas las partidas, agruparemos estas en archivos .csv para su posterior uso, esto lo hacemos mediante la libreria de pandas con Python.

```
region_df = pd.DataFrame(game_region_stats)
region_df.to_csv( path + region + "_stats.csv" )
```

Para un resumen detallado del proceso seguido para realizar esto, podemos encontrarlo en el repositorio de github disponible en `https://github.com/CarlosCanut/tfg_recommender/tree/`

`main`.

De estas partidas se obtuvieron datos de 163 campeones, de los cuales se cuentan con todos con más de 10 partidas por campeón, con este conjunto de datos contamos ya con información del 100 % de los campeones presentes en el juego, por lo que se procedió a extraer todas las estadísticas que ofrece la API de estas partidas, finalmente se seleccionaron 120 de cada uno de los jugadores participantes en cada partida, estas estadísticas fueron seleccionadas basandonos en diferentes categorias para tratar de obtener la máxima información de cada partida, siendo estas las siguientes:

1. Información de partida: Aquí englobamos todo lo que describa la situación de partida en la que se encuentra el jugador, campeón que utiliza, lado del mapa...
2. Objetivos: En esta categoría englobamos todo lo referente a estructuras y monstruos neutrales eliminados.
3. Recursos: Todo lo referente a la obtención de oro y experiencia en partida.
4. Daño: Todas las estadísticas relacionadas con daño recibido o inflingido.
5. Posicion: En esta categoría incluimos los porcentajes de tiempo en cada uno de los carriles y lo relacionado con la posición en el mapa.

Todas las estadísticas se describen en el documento anexo (ver Anexo **Estadísticas extraídas de partidas de cola en solitario**).

4.2. Extracción de datos

La preparación de los clusters utilizados para las recomendaciones se han desarrollado, como indicado previamente, realizando primero una extracción de los datos para los conjuntos de partidas de cola en solitario, aplicando posteriormente algoritmos de reducción de dimensionalidad y de clustering, para así finalmente poder realizar los clusters que nos ayudarán en la generación de recomendaciones. Para el desarrollo desde la extracción de datos hasta la creación de clusters, se ha hecho uso del lenguaje Python y se ha creado un repositorio de GitHub específico para todo el proceso, en este se ha definido un script principal mediante el cual se hacía uso de diversas funciones definidas para cada uno de los apartados (ver Figura 4.2).

```
624  if __name__ == '__main__':
625      extract_soloq_games()
626      general_clustering()
627      clustering_testing_soloq_games()
628      clustering_soloq_games()
629      extract_competitive_games()
630      generate_competitive_clustered_dataset_by_order_pick()
```

Figura 4.2: Orden de funciones de extracción y generación de clusters. Fuente: recurso propio

Para la extracción de datos, se han realizado dos conjuntos de datos, uno enfocado en obtener datos del máximo número de campeones posibles para la creación de los clusters, y otro que, mediante el uso de los clusters creados, realice las recomendaciones basándose en partidas competitivas.

Conjunto de datos de partidas en solitario

El primer conjunto de datos, utilizado para crear los clusters, se ha creado utilizando datos de la cola en solitario a través de la api pública de Riot Games. En cuanto al proceso seguido, como se ha descrito previamente, primero se han recopilado cada uno de los jugadores existentes en aspirante, rango donde se encuentran los mejores jugadores, haciendo uso del endpoint /lol/league/v4/challengerleagues/by-queue/queue", para poder almacenar a cada uno de los jugadores la api nos ofrece de cada cuenta su id global (PUUID), almacenaremos este para no perder el rastro de las cuentas si estas sufren algún cambio de región o de nombre. En la Figura 4.3 se muestra como se ha almacenado por región a cada jugador en un fichero json para posteriormente usar este para la extracción de datos.

Figura 4.3: puuids jugadores cola en solitario. Fuente: recurso propio

Después de obtener todos los jugadores, se procede a obtener las últimas partidas de cada jugador, para esto lo que hacemos es obtener las últimas 100 partidas de cada jugador, juntar todas las partidas de cada uno de los jugadores para eliminar partidas duplicadas y extraer información de cada una de las partidas. De manera que se generan ficheros csv para cada una de las regiones registradas con los datos de todas las partidas, almacenando las estadísticas previamente enumeradas (ver Figura 4.4).

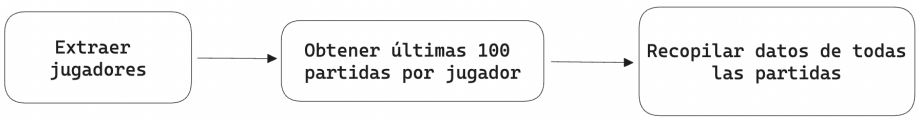


Figura 4.4: Extracción de datos. Fuente: recurso propio

Reducción de dimensionalidad y Clustering

Posteriormente a tener todas las estadísticas de partidas de jugadores de alto nivel, se procede a obtener los diferentes clusters a los que vamos a asignar cada uno de los campeones, para esto, primero vamos a basarnos en los propios roles en los que se reparten los campeones de League of Legends en una partida, estos son los siguientes:

1. Top: Encargado de ir al carril superior.
2. Jungla: Campeones sin carril del mapa decidido, estos se encargan de eliminar monstruos neutrales que aparecen en el mapa y asistir en los diferentes carriles.
3. Medio/Mid: Encargado de ir al carril central.
4. Ad carry/Bottom: Encargado de la línea inferior junto con el Soporte.
5. Soporte/Utility: Encargado de asistir en el carril inferior y en otros carriles si fuera necesario de manera puntual.

Vamos a tratar de obtener más información de estos, para lo cual realizaremos un primer clustering utilizando los campeones sin filtrar por ningún tipo de rol. Para esto vamos a utilizar los datos de partidas de la versión 13.10 de League of Legends de la cual hemos obtenido previamente, aplicaremos primero un rápido análisis mediante el método del codo para tratar de detectar el valor de 'k' para k means óptimo (ver Figura 4.5).

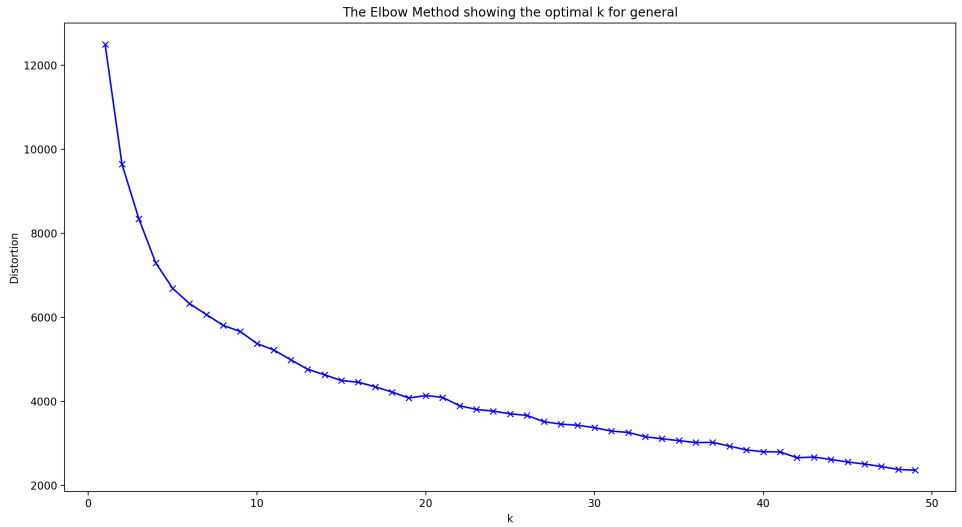


Figura 4.5: Método del codo para clustering general. Fuente: recurso propio

Una vez realizado este análisis, al no encontrar un valor de k claro el cual utilizar, utilizaremos las diferentes técnicas de reducción de dimensionalidad y clustering para encontrar cual es la mejor combinación de parámetros y algoritmos y aplicaremos estos para generar el mejor clustering y comparar la similitud de este con los roles ya existentes en League of Legends (ver Tabla 4.1).

Cuadro 4.1: Clustering General

reduction	clustering	reduction param	clustering param	silhouette_avg	calinski_harabasz	davies_bouldin
umap	kmeans	3	5	0.63	545.41	0.50
umap	kmeans	9	5	0.63	492.10	0.51
umap	kmeans	5	5	0.62	515.31	0.52
umap	kmeans	7	5	0.61	483.09	0.52
umap	kmeans	2	5	0.61	642.70	0.50

Una vez realizado, puede verse que la mejor combinación siempre opta por un clustering usando k means con una k de 5, es decir, generando 5 grupos, los cuales coinciden con los 5 roles. Dado que estamos tratando en cierto modo de hacer una comparativa de los roles con este primer clustering. Vamos a comprobar de este clustering que campeones han sido agrupados en clusters similares diferenciando estos en base al rol en el que más partidas se han jugado con cada uno. En la siguiente tabla se muestra el número de campeones pertenecientes a cada grupo (columnas) y de estos, mostramos el rol en el que más se han jugado (filas). Puede verse el cluster más común para cada uno de los roles (ver Tabla 4.2).

Cuadro 4.2: Relación entre roles y clustering general

role	3	1	0	2	4	most_common_group
MIDDLE	24	1	0	0	6	3
JUNGLE	3	1	28	0	0	0
TOP	3	32	0	0	1	1
UTILITY	0	0	1	27	0	2
BOTTOM	0	0	0	1	19	4

Así podemos ver como cada uno de estos clusters se considera principalmente un rol, y existen ciertos campeones que aparecen en un grupo aún siendo principalmente jugados en otro, esto se debe a que hay ciertos campeones con un estilo de juego un tanto atípico los cuales suelen tener conductas similares a las de otros roles en ciertos aspectos del juego debido a la naturaleza de estos. Por lo que para concluir podemos ver que un primer clustering nos arroja información general sobre los roles, y estos clusters pueden ser utilizados en el futuro para tratar casos en los que no queramos definir un rol concreto para un campeón y podamos hacer uso de estos grupos que toman en cuenta no solo los roles, sino también el comportamiento a un nivel más preciso de cada campeón.

Conjunto de partidas competitivas

Una vez obtenidos los grupos necesarios para poder clasificar cada campeón de manera general y en base a su rol, pasamos a la extracción de datos de partidas competitivas en función al orden

de selección y rol de cada uno de los campeones.

Primero, para la extracción de datos, se hará mediante el uso de la API pública de la wiki de League of Legends se van a extraer datos de la selección de campeón de todos los partidos de ligas profesionales jugados en 2023.

Para extraer información de la wiki de League of Legends hay que hacer uso del sistema que tienen implementado para extraer información de partidas competitivas, este funciona mediante un wrapper de SQL llamado Cargo (11), para esto hemos usado la librería llamada mwclient (13) de python con la cual podemos hacer llamadas y extraer datos de las partidas, para esto hemos creado un proceso por el cual, tomando en cuenta la limitación de partidas que podemos obtener por llamada de 500 partidas, hemos hecho múltiples llamadas y obtenido los datos de todas las partidas.

Para esto primero se han obtenido los datos de estas partidas y posteriormente se han agrupado todas en un único conjunto de datos. De este conjunto final se han extraído los datos de cada campeón y su rol en partida en base al orden en la selección de campeón y agrupados por partida añadiendo una columna donde se indica el lado del mapa ganador, por último se ha transformado el id de cada campeón en el cluster al que pertenecen (ver Tabla 4.3).

Cuadro 4.3: Partidas competitivas con clusters

game_id	**side_winner**	**cluster1**	**...**	**cluster n**	**role1**	**...**	**role n**
3087499	100	2	...	0	utility	...	role
3086486	100	3	...	0	mid	...	role
3087532	100	2	...	0	utility	...	role

4.3. Procesamiento de datos

Después de haber realizado la extracción de datos, contamos con 120 atributos que hacen referencia a cada una de las estadísticas extraídas de un total de 32625 partidas.

Primero se ha planteado el uso de una serie de algoritmos de reducción de dimensionalidad, estos son Análisis de Componentes Principales (PCA), Incrustación Estocástica de vecinos t-Distribuidos (t-SNE) y Aproximación y Proyección Uniforme de Variedad (UMAP).

A partir de la reducción de dimensionalidad se han planteado diversos algoritmos de clustering para posteriormente poder realizar las recomendaciones en base a los grupos generados. Los algoritmos de clustering planteados para este problema han sido K-Means, OPTICS y DBSCAN.

Después de haber explicado el funcionamiento de estos algoritmos, se procedió a la normalización de los datos y a la aplicación de estos algoritmos con diferentes combinaciones de parámetros para tratar de encontrar la mejor combinación posible.

Para la elección de la mejor combinación de algoritmos y parámetros se hace uso de unas métricas mediante las cuales se busca encontrar la mayor distancia entre cada uno de los grupos.

4.4. Optimización de Clustering

Una vez realizado la reducción de dimensionalidad y el clustering, para tratar de encontrar el mejor clustering realizado a cada uno de los roles, se decidió hacer uso de métricas para la evaluación de los diferentes clusters, las elegidas para esto fueron el Índice de Calinski-Harabasz(23), el Índice de Davies-Bouldin(24) y el coeficiente de silueta(25).

La búsqueda del mejor clustering se llevó realizando combinaciones de diferentes algoritmos con distintos valores, estos luego fueron comparados entre sí haciendo uso de las métricas de evaluación previamente mencionadas, de entre estos se realizó una clasificación y se seleccionó el que mejor puntuación obtuviera para cada rol, en las siguiente tablas se muestran las 5 mejores combinaciones para cada rol, Top (ver Tabla 4.4), Jungle (ver Tabla 4.5), Mid (ver Tabla 4.6), Bottom (ver Tabla 4.7), Utility (ver Tabla 4.8).

Cuadro 4.4: Clustering Top

reduction	**clustering**	**reduction param**	**clustering param**	**silhouette_avg**	**calinski_harabasz**	**davies_bouldin**
umap	kmeans	2	4	0.46	77.65	0.74
umap	kmeans	2	3	0.45	70.44	0.77
umap	kmeans	2	5	0.44	78.24	0.79
umap	kmeans	2	6	0.44	77.93	0.70
umap	kmeans	5	3	0.42	56.24	0.89

Cuadro 4.5: Clustering Jungle

reduction	**clustering**	**reduction param**	**clustering param**	**silhouette_avg**	**calinski_harabasz**	**davies_bouldin**
umap	kmeans	2	4	0.42	48.22	0.73
umap	kmeans	2	7	0.39	51.95	0.74
umap	kmeans	2	3	0.39	47.78	0.88
umap	kmeans	2	8	0.38	52.29	0.73
umap	kmeans	2	9	0.38	52.48	0.71

Cuadro 4.6: Clustering Mid

reduction	clustering	reduction param	clustering param	silhouette_avg	calinski_harabasz	davies_bouldin
umap	kmeans	2	4	0.46	60.00	0.66
umap	kmeans	2	5	0.45	62.14	0.69
umap	kmeans	3	4	0.44	51.33	0.75
umap	kmeans	9	4	0.43	43.74	0.74
umap	kmeans	5	4	0.43	40.62	0.77

Cuadro 4.7: Clustering Bottom

reduction	clustering	reduction param	clustering param	silhouette_avg	calinski_harabasz	davies_bouldin
umap	kmeans	2	3	0.42	24.06	0.77
pca	kmeans	0.80	3	0.41	5.90	0.38
umap	kmeans	2	4	0.38	25.17	0.76
umap	kmeans	2	6	0.37	22.00	0.67
umap	kmeans	10	6	0.34	10.93	0.88

Cuadro 4.8: Clustering Utility

reduction	clustering	reduction param	clustering param	silhouette_avg	calinski_harabasz	davies_bouldin
umap	kmeans	10	3	0.48	33.50	0.68
umap	kmeans	6	3	0.48	30.80	0.72
umap	kmeans	3	3	0.48	33.48	0.65
umap	kmeans	5	3	0.47	32.54	0.70
umap	kmeans	2	3	0.47	38.12	0.69

Una vez repartidos todos los campeones en base a su cluster por rol, obtenemos la distribución de campeones por rol que se muestra en la Figura 4.6, el siguiente paso es analizar estos clusters y comprobar si estos muestran algún tipo de relación.

Figura 4.6: Clusters de campeones por rol. Fuente: recurso propio

Para analizar más a fondo los clusters, se han seleccionado una serie de estadísticas de nuestro conjunto de datos para un pequeño análisis y visualización, estas tratan de mostrar el funcionamiento de los campeones de cada cluster, de manera que podamos analizar si existen diferencias significativas entre estos.

En la línea de Top (ver Figura 4.7) observando los datos, podemos ver que el grupo 1 es el que más daño recibe, debido a que suelen ser campeones enfocados en tener una buena defensa y poder aguantar golpes, en cuanto al grupo 0 y el 2 son los que más daño realizan en partida, esto se debe a que son un estilo de campeón que, en el caso de jugarse suele enfocarse en hacer el máximo daño, en el caso del grupo 2 mediante habilidades de daño en área.

En la línea de jungla podemos observar que los grupos 0 y 2 son de campeones que utilizan daño mágico.

En la línea de mid podemos observar diferencias en el grupo 2, siendo este enfocado en daño físico y no daño mágico, siendo este mismo grupo también característico por su daño a estructuras.

En el rol de adc (bottom) no se presentan diferencias significativas entre los 3 grupos.

En el rol de support (utility) podemos observar una diferencia en el grupo 2 en cuanto al daño que este hace, siendo significativamente superior, y el grupo 0 el cual es el único el cual muestra un valor notorio en el valor de curaciones.

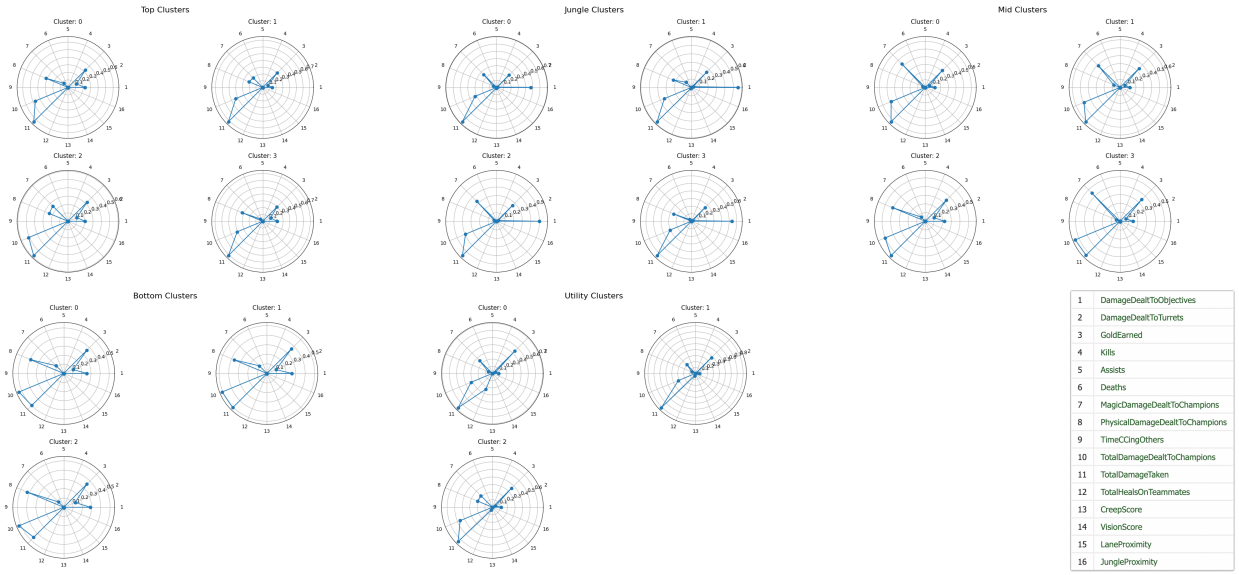


Figura 4.7: Estadísticas normalizadas por cluster. Fuente: recurso propio

4.5. Algoritmo de recomendación de campeones

En este apartado se va a plantear el proceso mediante el cual, una vez obtenido el conjunto de datos de partidas competitivas con datos de draft con los respectivos cluster de cada campeón y rol, se realiza la recomendación en base a una selección de campeón inacabada.

El primer paso para la recomendación es el de transformar el conjunto de datos de manera que consigamos un conjunto el cual sea sencillo de manipular para las recomendaciones. Primero se han agrupado cada uno de los clusters por rol, nuestro objetivo es poder diferencias los clusters de cada rol de manera global entre todos los roles, al repetir etiquetas en cada rol (dado que el etiquetado de los clusters de cada rol comienzan en 0), para solucionar esto se han actualizado las etiquetas, enumerando de nuevo los clusters por rol comenzando desde el rol de top hasta support, sin comenzar a contar de nuevo cada vez que cambiamos de rol (ver Tabla 4.9).

Cuadro 4.9: Clusters adaptados para algoritmo de recomendación

	top	**jungla**	**mid**	**ad carry**	**support**
antiguos cluster	0,1,2,3	0,1,2,3	0,1,2,3	0,1,2	0,1,2
nuevos cluster	0,1,2,3	4,5,6,7	8,9,10,11	12,13,14	15,16,17

Una vez actualizado el conjunto de datos, el siguiente paso será realizar una búsqueda utilizando el draft a predecir y encontrar cuál será la mejor elección para el siguiente turno utilizando el conjunto de datos. Para esto, se buscan drafts que coincidan con el mismo orden de selección y seleccionar el que cuente con una mayor frecuencia dentro de este subgrupo, utilizando como subgrupo el

conjunto completo de todas las posibles selecciones si no se encontrara ningún caso similar de draft.

Capítulo 5

Implementación de la Aplicación Web

Siguiendo los requisitos expuestos previamente, se explicará cómo se ha implementado la aplicación web.

Para esto se señalará cada una de las funcionalidades implementadas con las que se ha cumplido. Después, se desplegará por partes, de la extracción de datos se expondrá el proceso para la extracción de clusters, posteriormente la generación de recomendaciones, y de la parte de front-end se indicará la estructura de ficheros y componentes utilizados.

Por último, de la parte de back-end se expondrá la estructura de ficheros, el despliegue en azure para la base de datos y funciones utilizadas a modo de API por la web para la obtención de las recomendaciones (ver Figura 5.1).

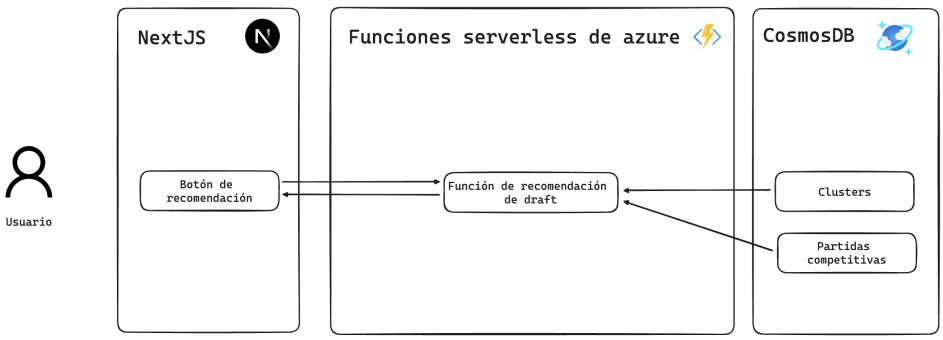


Figura 5.1: Esquema de obtención de recomendaciones. Fuente: recurso propio

5.1. Backend

El Backend podemos dividirlo en dos secciones, el servidor y la base de datos, tanto nuestro servidor como base de datos han sido creados mediante servicios en la nube de Microsoft Azure, siendo el servidor gestionado mediante Azure Serverless Functions (funciones sin servidor de azure) y la base de datos una base de datos Cosmos.

Azure Serverless Functions

El servicio de Serverless Functions de Azure funciona de manera que se alojan funciones las cuales pueden consumir recursos de otros servicios de Azure, como es por ejemplo la base de datos de Cosmos.

Este servicio nos sirve como capa de comunicación entre el cliente web y la base de datos, en nuestro caso hemos creado una función llamada GetRecommendation() para realizar la recomendación de draft, de manera que a esta se envía mediante una petición POST donde se almacena en el cuerpo la selección de campeón la cual se quiere obtener una recomendación y esta nos devuelve la recomendación, y otra función encargada de enviar cada uno de los clusters generados, el campeón y rol al que pertenecen llamada GetClusters() (ver Figura 5.2).

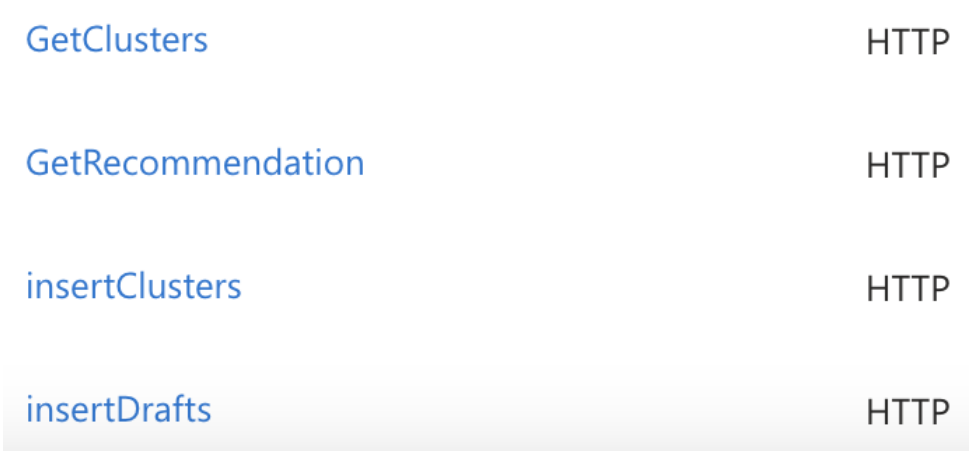


Figura 5.2: Funciones serverless de azure. Fuente: recurso propio

Estas funciones son insertClusters() para actualizar la información de los diferentes clusters extrayendo datos de cola en solitario y insertDrafts(), la cual está encargada de actualizar las partidas de competitivo haciendo uso de los clusters (ver Figura 5.2).

Para contar con los datos actualizados, como ya se ha mencionado se han creado dos funciones encargadas de actualizar los datos almacenados en la base de datos, a continuación explicamos como funcionan esta mismas.

Primero, la función de insertClusters() obtiene el contenedor dentro de Cosmos DB donde se almacenan los clusters, y almacena estos en la base de datos como un nuevo item.

```
client = CosmosClient.from_connection_string(connection_string)
database_name = 'Drafts'
container_name = 'clusters'
database = client.get_database_client(database_name)
container = database.get_container_client(container_name)

# Convert the DataFrame to a list of dictionaries
data = df.to_dict(orient='records')
```

```
# Insert each dictionary into Cosmos DB
for item in data:
    logging.info(item)
    item['id'] = str(uuid.uuid4()) # Generate 'id' field
    container.create_item(body=item)
```

A continuación, la función de insertDrafts() sirve para insertar cada una de las selecciones de campeón, para esto se sigue el mismo proceso que en la función de insertClusters() pero almacenando los datos en el contenedor de competitive.

```
# Create a Cosmos DB client and retrieve the container
client = CosmosClient.from_connection_string(connection_string)
database_name = 'Drafts'
container_name = 'competitive'
database = client.get_database_client(database_name)
container = database.get_container_client(container_name)

# Convert the DataFrame to a list of dictionaries
data = df.to_dict(orient='records')

# Insert each dictionary into Cosmos DB
for item in data:
    logging.info(item)
    item['id'] = str(uuid.uuid4()) # Generate 'id' field
    container.create_item(body=item)
```

Para la función de getClusters(), ya que esta función se utiliza para obtener cada uno de los clusters y los campeones pertenecientes a estos, esta únicamente devueve los datos en formato json.

```
items = []
for document in doc:
    items.append(document.to_dict())

df = pd.DataFrame(items)

return func.HttpResponse(df.to_json(orient="records"), mimetype="application/json")
```

Por último, la función de GetRecommendations() hace una busqueda del draft con más similitu-

des al draft planteado y devuelve el siguiente campeón para seleccionar.

Todas estas funciones puedes encontrarse con más detalle en el repositorio con el siguiente enlace `https://github.com/CarlosCanut/tfg_azure/tree/main`

Cosmos Database

Para poder realizar las recomendaciones de la manera más rápida y eficiente, se ha optado por hacer uso de una base de datos documental que nos ofreciera una rápida implementación y un esquema de datos flexible, para esto hemos usado el servicio de azure de Cosmos DB creando una base de datos mongodb, en esta almacenamos los datos de drafts competitivos, así como los clusters a los que pertenece cada campeón, de manera que podemos hacer uso de estos datos para realizar las recomendaciones, para el almacenamiento de estos datos se ha optado por generar dos contenedores dentro de una misma base de datos, uno de ellos llamado **clusters** encargado de almacenar las traducciónes entre el id de campeón junto a su rol y el cluster al que pertenecen y otro encargado de almacenar los datos de selección de campeón de partidas profesionales mediante los clusters de cada rol llamado **competitive** (ver Figura 5.3).

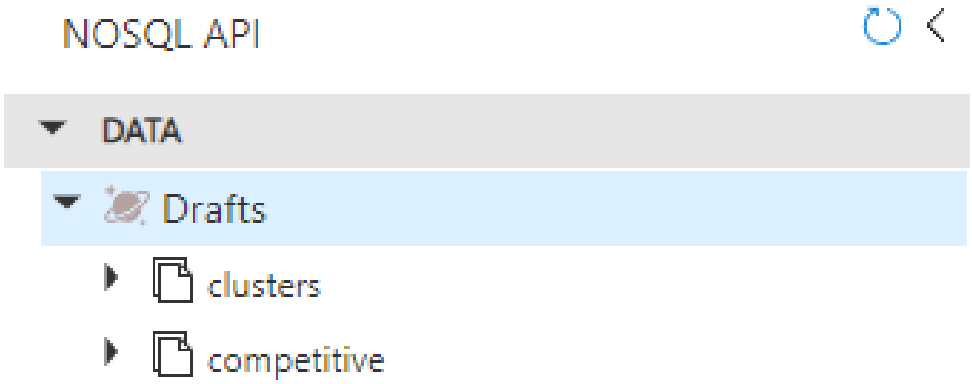


Figura 5.3: base de datos en cosmosdb. Fuente: recurso propio

5.2. Frontend

La implementación del frontend se ha realizado con NextJS, famoso framework de ReactJS, como se mencionó con anterioridad, una ventaja que ofrece utilizar un framework de ReactJS es el uso de componentes, estos funcionan de manera independiente, lo que ayuda a se realicen cargas de datos de manera aislada.

Otra ventaja que ofrece NextJS frente a otros frameworks es el renderizado de parte del servidor, esto ofrece la posibilidad de realizar carga de datos en el servidor.

Estructura de ficheros

Para la configuración inicial del código se ha creado un repositorio dedicado en GitHub, una vez creado el repositorio se ha creado la estructura inicial de la aplicación mediante el comando:

```
npm create-next-app recommender
```

Con este comando se generará la estructura que utilizaremos para nuestro proyecto, como características de NextJS, este ofrece un sistema de enrutamiento basado en archivos, de manera que cualquier archivo que se genere dentro de la carpeta /pages será tratado como una nueva ruta sin necesidad de configuración.

También se ha implementado Tailwindcss, un marco de diseño de CSS de alto nivel el cual permite realizar cambios de estilo mediante el uso de clases, este fue instalado haciendo uso del comando:

```
npm install -D tailwindcss postcss autoprefixer
```

Y posteriormente inicializandolo mediante:

```
npx tailwindcss init -p
```

A continuación podemos ver la estructura final.

Primero tenemos la carpeta /components que contiene todos los componentes que se van a reutilizar en la aplicación (ver Figura 5.4).

En la carpeta /node_modules encontramos todas las dependencias instaladas en el proyecto (ver Figura 5.4).

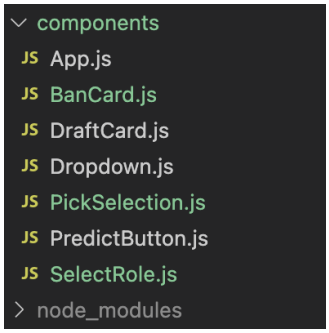


Figura 5.4: Estructura de ficheros /components /node modules. Fuente: recurso propio

El directorio /pages cuenta con el archivo /pages/_app.js y /pages/_document.js, estos están encargados de realizar una primera carga desde el servidor de la página que se enviará al cliente, en nuestro caso son los encargados de cargar el archivo de css global para el correcto funcionamiento de tailwindcss y de cargar las fuentes utilizadas en la web, descargando al cliente de estas tareas (ver Figura 5.5).

Por otro lado encontramos el archivo index.js, este se carga por parte del cliente a excepción de una función que encontramos en este archivo llamada getStaticProps(), esta es la encargada de definir una carga de datos la cual se realizará por parte del servidor, en esta cargamos imagenes de campeones y los clusters existentes para las posteriores recomendaciones, de esta manera se descarga la primera carga de la web (ver Figura 5.5).

Dentro de la misma carpeta de /pages podemos encontrar /pages/api, en esta es donde se encuentran todas nuestras conexiones hacia APIs de terceros, en esta se puede encontrar el fichero

de predict.js, aquí se define una ruta llamada /api/predict la cual hace de pasarela para conectar con nuestra API alojada en Azure, está llamada a nuestro servidor será realizada de manera encapsulada de manera que en el cliente se hará una llamada a un endpoint totalmente diferente cada vez que se utilice (ver Figura 5.5).

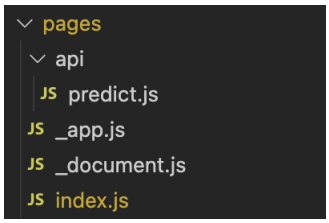


Figura 5.5: Estructura de ficheros /pages. Fuente: recurso propio

En la carpeta /public podemos encontrar todas las imágenes e iconos (ver Figura 5.6).

Dentro del directorio /styles encontramos el fichero globals.css, este es el encargado de cargar tailwindcss para su correcto funcionamiento (ver Figura 5.6).

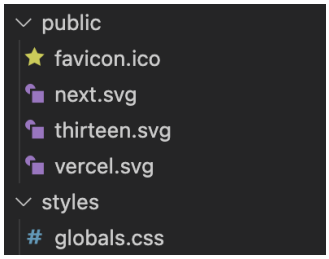


Figura 5.6: Estructura de ficheros /public /styles. Fuente: recurso propio

También podemos observar archivos enfocados en configuración, next.config.js para la configuración de NextJS, package-lock.json y package.json son archivos de metadatos para proyectos que utilizan NodeJS, postcss.config.js necesario para configurar tailwindcss, README.md utilizado para mostrar instrucciones y/o documentación del proyecto y tailwind.config.js el cual cuenta con la configuración de tailwindcss (ver Figura 5.7).

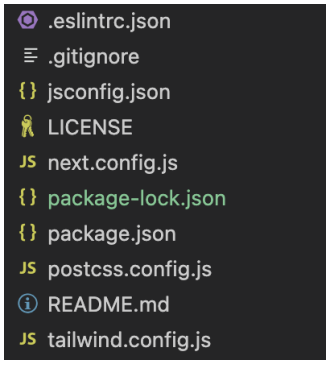


Figura 5.7: Estructura de ficheros configuración. Fuente: recurso propio

Una vez se ha terminado de implementar la aplicación, el siguiente paso es exportarla y desplegarla, para esto se ha utilizado vercel, mediante la misma interfaz de la página de vercel se ha

podido conectar nuestro repositorio de GitHub para que se realice un despliegue en un servidor de vercel generando una build de producción del proyecto mediante el comando:

```
npm run build
```

Capítulo 6

Conclusiones

Durante la realización de este proyecto se ha conseguido desarrollar una aplicación web que ofrece la posibilidad de agilizar y ofrecer un planteamiento efectivo a la preparación de partidos en el ámbito competitivo profesional del videojuego League of Legends, alcanzando todos las metas inicialmente planteadas. La aplicación web ha sido desplegada en los servidores de vercel y está disponible en el siguiente enlace: `https://carlos-canut-tfg.vercel.app/`, la extracción y procesado de los datos se ha realizado y publicado en un repositorio de github en el siguiente enlace: `https://github.com/CarlosCanut/tfg_recommender/tree/main`, y todas las funciones existentes en la plataforma de azure se encuentran previamente definidas y podemos encontrarlas en el siguiente repositorio: `https://github.com/CarlosCanut/tfg_azure`.

Se ha conseguido realizar un prototipo que cumple todos los requisitos planteados siendo esta una aplicación que simula una selección de campeones de League of Legends y ofrece recomendaciones manteniendo un diseño intuitivo y similar al ya existente en League of Legends.

La aplicación ha sido creada a partir de NextJS, NodeJS, Azure Serverless Functions mediante Python y Azure Cosmos DB.

Como líneas de trabajo futuro, se podría realizar modelos de recomendación basados en enfrentamientos entre dos equipos utilizando únicamente conjuntos de datos con presencia de estos equipos. También podrían probarse otros algoritmos a la hora de realizar las recomendaciones como podría ser el uso de Monte Carlos Tree Search (MCTS), o incluso crear un modelo de predicción de resultados de partida y generar simulaciones de manera que podamos alimentar nuestras recomendaciones de estas simulaciones y llegar a ofrecer recomendaciones más precisas.

También, en lo que respecta a la validación de la propuesta, se trató de realizar con usuarios profesionales, pero debido al difícil acceso a estos y el limitado tiempo no se pudo realizar, por lo que, como trabajo futuro se plantea la validación de la propuesta con un conjunto relevante de usuarios profesionales.

Como anotación final, comentar que dado el interés en poder llegar a implementar este sistema, se encuentra en estado de desarrollo un artículo científico donde se desarrolla la propuesta

basándose como punto inicial donde termina este mismo trabajo fin de grado, aplicando cada una de las propuestas de trabajo futuro previamente comentadas.

El diseño e implementación de un sistema completo que cuenta tanto con una creación de diseño visual, implementación de una aplicación web y la creación de un modelo de recomendaciones, ha supuesto un aprendizaje a nivel personal y todo un impulso a la formación recibida en el grado. Además del aprendizaje sobre el mundo de los deportes electrónicos y la posibilidad de haber creado un proyecto en un ámbito tan innovador y apasionante como este estoy seguro que todos estos conocimientos me ayudarán en el futuro en mi desarrollo profesional.

Bibliografía

[1] League of Legends. (s.f.). [en línea] Disponible en: https://www.leagueoflegends.com/es-es/ [Accedido el 3 de octubre de 2023].

[2] Warcraft III. (s.f.). [en línea] Disponible en: https://en.wikipedia.org/wiki/Warcraft_III:_The_Frozen_Throne [Accedido el 3 de octubre de 2023].

[3] Statista. (2023). Evolución anual de los ingresos anuales generados por las competiciones de videojuegos a nivel mundial entre 2022 y 2030. [en línea] Statista. Disponible en: [URL] [Accedido el 3 de octubre de 2023].

[4] Statista. (2023). Tamaño de la audiencia de los deportes electrónicos a nivel mundial de 2020 a 2025, por tipo de espectadores. [en línea] Statista. Disponible en: [URL] [Accedido el 3 de octubre de 2023].

[5] Studholme, B. (2023). El mercado salarial de los deportes electrónicos se encamina hacia una corrección. Digiday. [en línea] Disponible en: [URL] [Accedido el 3 de octubre de 2023].

[6] Esports Charts. (s.f.). [en línea] Disponible en: https://escharts.com/tournaments/lol/2022-world-championship [Accedido el 3 de octubre de 2023].

[7] Razablan. (2020). LoL: Worlds 2012, el nacimiento de la Copa del Invocador y el inicio de la dominación asiática. millenium.

[8] Wikipedia. (s.f.). League of Legends: Season 2 World Championship. [en línea] Disponible en: https://en.wikipedia.org/wiki/League_of_Legends:_Season_2_World_Championship#:~:text=Over%208%20million%20viewers%20tuned,in%20history%20at%20the%20time. [Accedido el 3 de octubre de 2023].

[9] FactorGG. (s.f.). [en línea] Disponible en: https://www.factor.gg/ [Accedido el 3 de octubre de 2023].

[10] Leaguepedia. (s.f.). League of Legends Esports Wiki. [en línea] Disponible en: https://lol.fandom.com/wiki/League_of_Legends_Esports_Wiki [Accedido el 3 de octubre de 2023].

[11] Cargo API. (s.f.). [en línea] Disponible en: https://lol.fandom.com/api.php?action=help&modules=cargoquery [Accedido el 3 de octubre de 2023].

[12] LolPros. (s.f.). [en línea] Disponible en: https://lolpros.gg/ [Accedido el 3 de octubre de 2023]

[13] Librería mwclient. (s.f.). [en línea] Disponible en: https://github.com/mwclient/mwclient [Accedido el 3 de octubre de 2023].

[14] Games of Legends. (s.f.). [en línea] Disponible en: https://gol.gg/esports/home/ [Accedido el 3 de octubre de 2023].

[15] DeepLol Pro. (s.f.). [en línea] Disponible en: https://pro.deeplol.gg/ [Accedido el 3 de octubre de 2023].

[16] ShadowGG. (s.f.). [en línea] Disponible en: https://shadow.gg/ [Accedido el 3 de octubre de 2023].

[17] Análisis de Componentes Principales. (s.f.). [en línea] Disponible en: https://es.wikipedia.org/wiki/An%C3%A1lisis_de_componentes_principales [Accedido el 3 de octubre de 2023].

[18] Aproximación y Proyección Uniforme de la Variedad para la Reducción de Dimensionalidad. (s.f.). [en línea] Disponible en: https://umap-learn.readthedocs.io/en/latest/ [Accedido el 3 de octubre de 2023].

[19] Incrustación Estocástica de Vecinos t-Distribuidos. (s.f.). [en línea] Disponible en: https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding [Accedido el 3 de octubre de 2023].

[20] K-Medias. (s.f.). [en línea] Disponible en: https://es.wikipedia.org/wiki/K-medias [Accedido el 3 de octubre de 2023].

[21] Ordenación de Puntos para Identificar la Estructura de Agrupamiento. (s.f.). [en línea] Disponible en: https://www.dbs.ifi.lmu.de/Publikationen/Papers/OPTICS.pdf [Accedido el 3 de octubre de 2023].

[22] Agrupamiento Espacial Basado en Densidad de Aplicaciones con Ruido. (s.f.). [en línea] Disponible en: https://es.wikipedia.org/wiki/DBSCAN [Accedido el 3 de octubre de 2023].

[23] Índice Calinski-Harabasz. (s.f.). [en línea] Disponible en: https://help.alteryx.com/2020.2/es/K-Centroids_Diagnostics.htm#:~:text=El%20%C3%ADndice%20Calinski%2DHarabasz%20se,puntos%20dentro%20de%20un%20cl%C3%BAster). [Accedido el 3 de octubre de 2023].

[24] Índice de Davies-Bouldin. (s.f.). [en línea] Disponible en: https://en.wikipedia.org/wiki/Davies%E2%80%93Bouldin_index [Accedido el 3 de octubre de 2023].

[25] Coeficiente de Silueta. (s.f.). [en línea] Disponible en: https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c [Accedido el 3 de octubre de 2023].

[26] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., ... Powell, R. (2019). Nivel Grandmaster en StarCraft II mediante aprendizaje por refuerzo multiagente. nature.

[27] Microsoft. (2020). Cloud9 conquista los deportes electrónicos mediante el análisis de datos en Azure. Microsoft.

[28] Arena de Batalla en Línea Multijugador. (s.f.). [en línea] Disponible en: https://es.wikipedia.org/wiki/Videojuego_multijugador_de_arena_de_batalla_en_l%C3%ADnea [Accedido el 3 de octubre de 2023].

[29] Summerville, A., Cook, M., Steenhuisen, B. (2016). Draft-Analysis of the Ancients: Predicción de selecciones en DotA 2 mediante aprendizaje automático. Informe Técnico AAAI WS-16-22100.

[30] Chen, S., Zhu, M., Ye, D., Zhang, W., Fu, Q., Yang, W. (2021). ¿Qué héroes elegir? Aprendizaje para seleccionar en juegos MOBA con redes neuronales y búsqueda de árboles. arXiv:2012.10171v4 [cs.AI].

[31] Mobalytics. (s.f.). [en línea] Disponible en: https://mobalytics.gg/ [Accedido el 3 de octubre de 2023].

[32] OP.GG. (s.f.). [en línea] Disponible en: https://www.op.gg/ [Accedido el 3 de octubre de 2023].

[33] U.GG. (s.f.). [en línea] Disponible en: https://u.gg/ [Accedido el 3 de octubre de 2023].

[34] Leaguepedia Cargo Tables. (s.f.). [en línea] Disponible en: https://lol.fandom.com/wiki/Special:CargoTables [Accedido el 3 de octubre de 2023].

[35] Similitud del Coseno. (s.f.). [en línea] Disponible en: https://es.wikipedia.org/wiki/Similitud_coseno [Accedido el 3 de octubre de 2023].