UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería Informática

eSports, un enfoque basado en inteligencia artificial

eSports, an artificial intelligence-based approach

Realizado por
Miguel Torres Gómez

Tutorizado por
José del Campo Ávila

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, junio de 2021

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA INFORMÁTICA

eSports, un enfoque basado en inteligencia artificial

eSports, an artificial intelligence-based approach

Realizado por
Miguel Torres Gómez

Tutorizado por
José del Campo Ávila

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2021

Fecha defensa: julio de 2021

Resumen

Los denominados *eSports* han visto incrementada su popularidad en los últimos años debido al creciente interés tanto del público que asiste a sus numerosos eventos como inversores que ven en el juego una oportunidad única. Este trabajo trata de analizar y clasificar los resultados de los partidos del juego *Counter Strike: Global Offensive* (CS:GO) mediante el uso de técnicas de aprendizaje computacional (*machine learning*, por sus siglas en inglés). El objetivo consistirá en analizar las grabaciones de los partidos citados, que se encuentran almacenadas en numerosas páginas web, correspondientes a una de las ligas profesionales de este videojuego, denominada *ESL PRO LEAGUE*. Se analizarán entorno a 400 partidas de los jugadores de más alto nivel usando librerías de código abierto con el objetivo de extraer información, y así posteriormente, hacer uso de diferentes programas empleados en el ámbito de la minería de datos para realizar una limpieza de datos, crear modelos y extraer conclusiones. Los resultados muestran diversas reglas extraídas que pueden servir de ayuda a los equipos para modificar su estilo de juego con el objetivo de alzarse con la victoria, además, de diferentes estadísticas que muestran claramente un desbalanceo que debería ser solventado por los creadores del juego.

Palabras clave: Minería de datos, Inteligencia Artificial, Deportes electrónicos

Abstract

The so-called eSports have seen an increase in popularity in recent years due to the growing interest of both the public attending its numerous events and investors who see the game as a unique opportunity. This work aims to analyse and classify the results of Counter Strike: Global Offensive (CS:GO) matches by using computational learning techniques. The objective will be to analyse the recordings of the aforementioned matches, which are stored on numerous websites, corresponding to one of the professional leagues of this video game, called ESL PRO LEAGUE. Around 400 replays of the highest level players will be analysed using open source libraries with the aim of extracting information, and then making use of different programmes used in the field of data mining to perform data cleaning, create models and draw conclusions, such as Weka or Rapid Miner. The results show several extracted rules that can help teams to modify their play style in order to win, as well as different statistics that clearly show an imbalance that should be solved by the creators of the game.

Keywords: Data mining, Artificial Intelligence, eSports

Índice

1 Introducción

En este capítulo se introducen los razonamientos sobre los cuales se ha elegido y desarrollado el trabajo en este ámbito, además de una breve explicación sobre las tecnologías, conceptos y metodología aplicados en este.

1.1. Motivación

Debido a la rápida evolución de nuevas tecnologías, cada año se desarrollan una gran variedad de software y entre estos, muchos de ellos destinados al entretenimiento, como por ejemplo, es el caso de los videojuegos. Estos están destinados a todo tipo de público y ofrecen multitud de géneros distintos, incentivando la creación de una industria que cada vez suscita mayor interés, el de las competiciones electrónicas (*eSports*). Cada año se crean nuevas competiciones que ofrecen un premio mayor y por esto la competitividad entre clubes y equipos se ha visto incrementada considerablemente. Es por esto que es realmente útil analizar el funcionamiento del juego, lo que se conoce como *"meta-analysis"*, además de entender el estilo de juego de tus rivales para conseguir la victoria. Como hemos dicho la cantidad de información es cada vez mayor y se hace necesario utilizar herramientas y metodologías relacionadas con la minería de datos y aprendizaje computacional con el objetivo de extraer información y mejorar la forma de jugar de los equipos.

En los últimos años y debido al auge de plataformas para compartir contenido en línea como *Youtube* o *Twitch*, visualizar contenido nunca había sido tan fácil, tal y como se observa en la Figura 1, la evolución de los espectadores en estas plataformas no ha hecho más que aumentar.

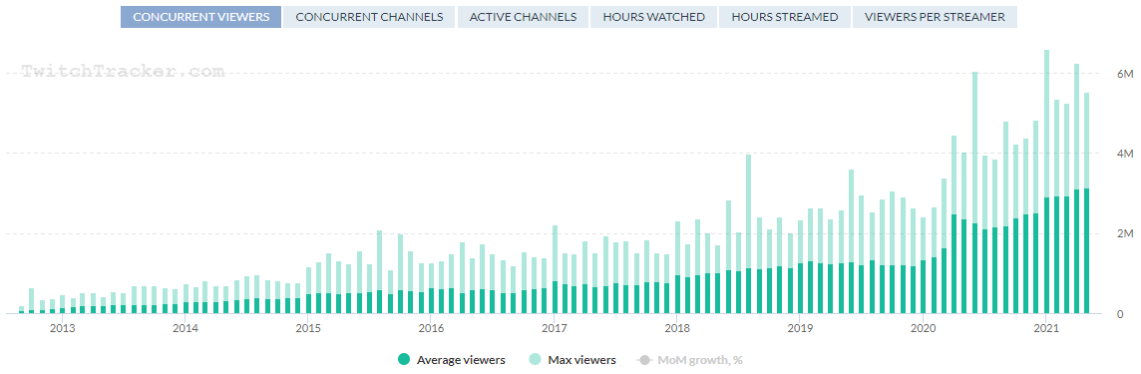

Figura 1: Numero de espectadores concurrentes por año en Twitch [16]

En dichas plataformas, la mayor cantidad de contenido que se consume esta relacionada con los videojuegos, en el caso que nos ocupa, se tratará de analizar la información del juego *Counter Strike:Global Offensive (CSGO)*. Dicho juego tiene una media de 100.000 espectadores diarios y millones de horas visualizadas al día, tal y como se observa en la Figura 2.

En definitiva, los *eSports* se han convertido en un fenómeno mediático y están aquí para quedarse.

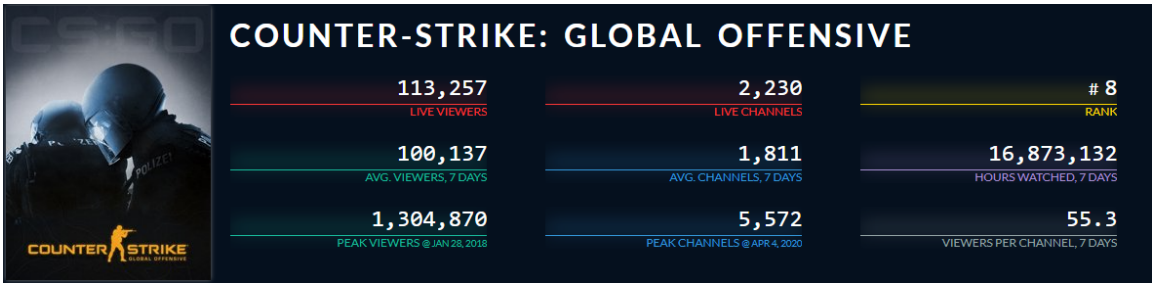

Figura 2: Estadísticas del videojuego CSGO en Twitch[6]

1.2. Objetivos

El trabajo consistirá en desarrollar modelos que extraigan conocimiento y que nos permitan determinar patrones de interés e incluso realizar predicciones (Arboles de decisión, Random Forest, Redes neuronales, Máquinas de soporte vectorial, etc.). A partir de información acerca de partidos, competiciones y el desempeño de los jugadores en dichos torneos.

El trabajo busca extraer patrones en el conjunto de datos, el cual se obtendrá tras la comprensión y preparación de datos a partir de las partidas grabadas previamente del videojuego

y haciendo uso de librerías *open-source*, para posteriormente: seleccionar, eliminar, mezclar y limpiar dicha información mediante técnicas de minería de datos, que nos permita identificar estrategias a seguir, como por ejemplo, el estilo de juego que mejor se adapte al encuentro (agresivo o defensivo), con el objetivo de ayudar a los equipos a alcanzar el éxito con mayor frecuencia. Se busca extraer información relevante sobre el estado actual del juego, como por ejemplo, datos sobre su equilibrio (se desarrollará más adelante en la sección 3.5) y atributos destacados. También se busca comparar, por ejemplo a modo de tabla, diversos modelos de predicción y clasificarlos según su rendimiento, coste y tiempo.

1.3. Minería de datos e inteligencia artificial

Como se ha comentando hasta el momento, el trabajo se encuadra dentro del ámbito de la minería de datos e inteligencia artificial los cuales forman parte de la inteligencia artificial, sin embargo, antes de entrar a comentar las distintas fases del proyecto y su desarrollo en general es necesario definir y explicar estos dos términos.

- **Inteligencia artificial**: se define como la capacidad de un sistema de interpretar correctamente los datos externos, de aprender de ellos y de utilizar esos aprendizajes para lograr objetivos y tareas específicas mediante una adaptación flexible [12]. El objetivo científico central de la inteligencia computacional es comprender los principios que hacen posible el comportamiento inteligente, en sistemas naturales o artificiales. La hipótesis principal es que el razonamiento es computación. El objetivo central de la ingeniería es especificar métodos para el diseño de artefactos útiles e inteligentes [5].

- **Minería de datos**: para hablar de este concepto, es necesario entender que el descubrimiento de conocimientos en las bases de datos (KDD) es un análisis exploratorio y automático y el modelizado de grandes repositorios de datos. El KDD es el proceso organizado de identificación de patrones válidos, novedosos, útiles y comprensibles a partir de conjuntos de datos grandes y complejos. La minería de datos (DM) es el núcleo del proceso de KDD, e implica la inferencia de algoritmos que exploran los datos, desarrollan el modelo y descubren patrones previamente desconocidos. El modelo se utiliza para la comprensión de los fenómenos a partir de los datos, el análisis y la predicción [9].

1.4. Tecnologías usadas

Principalmente, durante el desarrollo del trabajo se han utilizado herramientas de minería de datos e inteligencia artificial para extraer los modelos y lenguajes de programación para realizar la comprensión y preparación de datos. En concreto son los siguientes.

- **C Sharp**: es un lenguaje de programación multiparadigma desarrollado y estandarizado por la empresa Microsoft como parte de su plataforma .NET [3]. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes. Dicho lenguaje de programación se ha utilizado para realizar la comprensión y preparación de datos, tal como se ha dicho y aunque es parecido en su paradigma orientado a objetos a Java, se ha utilizado un enfoque basado en eventos. De tal forma que cuando un evento se invoca se realiza la recopilación de datos necesarios de dicho evento para posteriormente crear un conjunto de datos con todos los atributos recolectados y relacionados. En concreto los eventos forman parte del juego y nos sirven como referencia durante la ejecución de la partida (se desarrollará más adelante en la sección 3.2).

- **Weka**: es una colección de algoritmos de aprendizaje automático para tareas de minería de datos. Contiene herramientas para la preparación de datos, clasificación, regresión, agrupación, minería de reglas de asociación y visualización [17]. Weka es un software de código abierto publicado bajo la licencia pública general GNU. Se ha utilizado en la fase de modelizado y evaluación, ya que provee una gran cantidad de algoritmos para realizar dicha comprensión y preparación y favorece la recolección de estadísticas que han resultado muy útiles, y se han mostrado realmente interesantes en los resultados.

- **RapidMiner**: es un programa para el análisis y minería de datos. Permite el desarrollo de procesos de análisis de datos mediante el encadenamiento de operadores a través de un entorno gráfico. Se usa en investigación, educación, capacitación, creación rápida de prototipos y en aplicaciones empresariales. Se distribuye bajo licencia AGPL. RapidMiner proporciona más de 500 operadores orientados al análisis de datos, incluyendo los necesarios para realizar operaciones de entrada y salida, preprocesamiento de datos y visualización. También permite utilizar los algoritmos incluidos en Weka [15]. RapidMi-

ner facilita enormemente el trabajo al poseer una interfaz gráfica realmente intuitiva. Se ha realizado una comparación de diversos algoritmos para comprobar sus rendimientos con el conjunto de datos. Se trata de una de las soluciones más utilizadas en el campo de la ciencia de datos, tal y como se aprecia en la Figura 3.

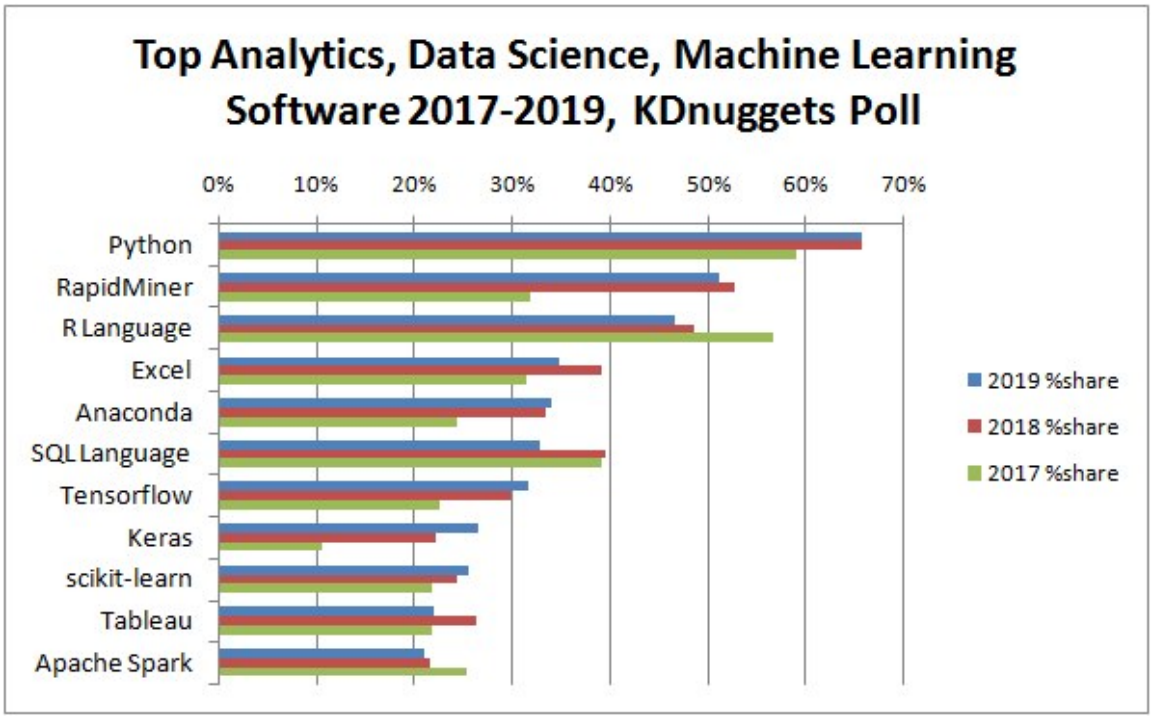

Figura 3: Plataformas de minería de datos más utilizadas entre los años 2017-2019 [14]

1.5. Metodología

En la realización del trabajo ha sido necesario establecer un proceso organizado a través de la metodología CRISP-DM (Cross Industry Standard Process for Data Mining). CRISP-DM es un modelo de proceso para la minería de datos que ofrece una visión general del ciclo de vida de un proyecto de minería de datos. Contiene las fases de un proyecto, sus respectivas tareas y las relaciones entre estas tareas. Las relaciones pueden existir entre cualquier tarea de minería de datos en función de los objetivos, los antecedentes y el interés del usuario y, sobre todo, de los datos. El ciclo de vida de un proyecto de minería de datos consta de seis fases, que se muestran en la Figura 4.

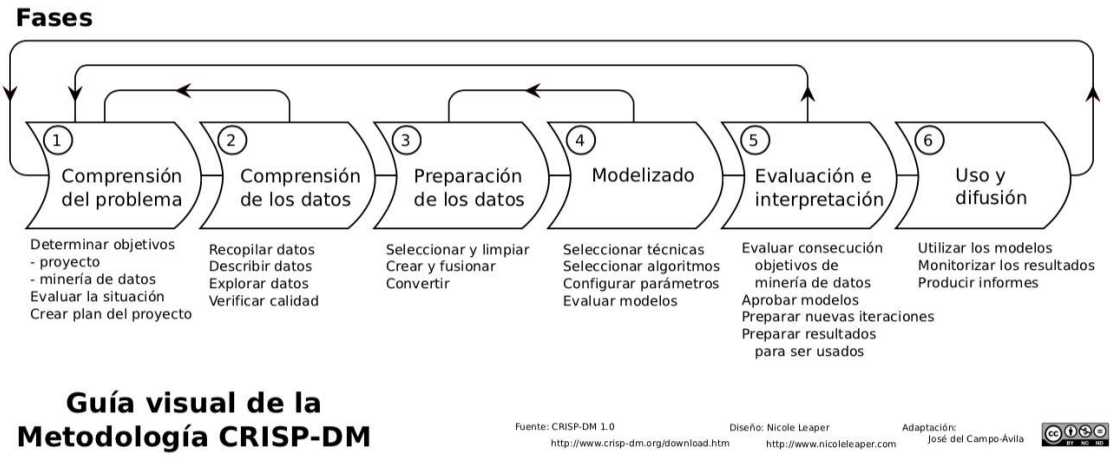

Figura 4: Diseño de la metodología CRISP-DM

La secuencia de las fases no es rígida. Siempre es necesario avanzar y retroceder entre las distintas fases. El resultado de cada fase determina qué fase, o una tarea concreta de una fase, debe realizarse a continuación. Las flechas indican las dependencias más importantes y frecuentes entre fases. El círculo exterior de la figura 4 simboliza la naturaleza cíclica de la propia minería de datos. La extracción de datos no termina una vez que se despliega una solución. Las lecciones aprendidas durante el proceso y de la solución desplegada pueden desencadenar nuevas cuestiones de negocio. Los procesos posteriores de datos se beneficiarán de las experiencias de los anteriores. A continuación, describimos brevemente cada fase [4].

- **Fase I. Comprensión del problema**: esta fase inicial, se centra en la comprensión de los objetivos y requisitos del proyecto para luego convertir este conocimiento en una definición del problema de minería de datos y un plan preliminar diseñado para lograr los objetivos.

- **Fase II. Comprensión de los datos**: en esta fase, se comienza con la recopilación inicial de datos y continúa con actividades que permiten familiarizarse con los datos, identificar los problemas de calidad de los datos, descubrir las primeras percepciones de los datos y/o detectar subconjuntos interesantes para formar hipótesis sobre la información oculta.

- **Fase III. Preparación de los datos**: en esta fase, se cubren todas las actividades necesarias para construir el conjunto de datos final (datos que se introducirán en las herra-

mientas de modelización) a partir de los datos brutos iniciales. Es probable que las tareas de preparación de datos se realicen varias veces y no en un orden determinado. Las tareas incluyen la selección de tablas, registros y atributos, así como la transformación y limpieza de los datos para las herramientas de modelización.

- **Fase IV. Modelizado**: en esta fase, se seleccionan y aplican diversas técnicas de modelización y se calibran sus parámetros hasta alcanzar los valores óptimos. Normalmente, existen varias técnicas para el mismo tipo de problema de minería de datos. Algunas técnicas tienen requisitos específicos sobre la forma de los datos. Por lo tanto, a menudo es necesario volver a la fase de preparación de los datos.

- **Fase V. Evaluación e interpretación**: en esta fase, se han construido uno o varios modelos que parecen tener una alta calidad desde la perspectiva del análisis de datos. Antes de proceder al despliegue final del modelo, es importante el papel de un experto ya que debe ser quien evalúe y revise a fondo los pasos ejecutados para estar seguro de que el modelo alcanza adecuadamente los objetivos. Al final de esta, debe tomarse una decisión sobre el uso de los resultados de la minería de datos.

- **Fase VI. Uso y difusión**: en esta fase, el conocimiento obtenido tendrá que organizarse y presentarse para que el cliente pueda usarlo. Dependiendo de los requisitos, la fase de desarrollo puede ser tan simple como la generación de un informe o tan compleja como la realización periódica y quizás automatizada de un proceso de análisis de datos en la organización (en nuestro caso particular esta fase no se desarrollará).

2

Counter Strike:Global Offensive

En este capítulo se hablará sobre las ideas básicas en las que se fundamenta el juego, el objetivo es ayudar a comprender mejor ciertos aspectos sobre este que pueden resultar en un principio difíciles de comprender, pero que son necesarios para el correcto entendimiento del desarrollo del trabajo.

2.1. Género

Counter Strike: Global Offensive es un juego de tipo *shooter*, estos se basan en la idea de que todos los jugadores portan un arma (normalmente de fuego) que pueden disparar a voluntad, en este sentido, existen numerosos subgéneros que tratan el tema desde diferentes puntos de vista como: la perspectiva, realismo, número de personajes, etc.

El objetivo del juego puede ser tremendamente variado pero en la mayoría podemos encontrar un factor en común: acabar con la vida del contrincante.

2.2. Estructura

La configuración del juego es de una sencillez relativa, cada partida esta compuesta por dos equipos, 10 jugadores (5 por equipo), 30 rondas y un mapa, escenario donde se desarrolla la partida.

La partida suele tener una duración de entre unos 45-60 minutos. Cada equipo forma parte de un bando (***CounterTerrorist*** *(equipo* ***CT****)* o ***Terrorist*** *(equipo* ***T****)*), el equipo que gana debe alzarse con la victoria en 16 de esas 30 rondas, sin embargo, cuando en la partida han trans-

currido 15 rondas (la mitad de la partida) ambos equipos cambian al bando contrario jugando desde una perspectiva diferente.

Como se ha comentado, el juego esta compuesto por 30 rondas, es decir, los equipos pueden llegar a un resultado de empate. Para ello se dispone de un tiempo extra cuyo objetivo es desempatar la partida y finalmente declarar a un vencedor.

2.3. Funcionamiento

En cada mapa existen dos zonas o perímetros donde se tiene que acceder para activar un dispositivo explosivo o bomba. Tal como se ha mencionado anteriormente en cada mapa existen varias zonas de importancia que podemos resumir en las siguientes 4:

- **Zona de aparición del equipo CT**: se trata de la parte del escenario o mapa donde aparecen los jugadores del equipo *CounterTerrorist (CT)*, podemos observar dicha zona en el círculo azul de la Figura 5.
- **Zona de aparición del equipo T**: se trata de la parte del escenario o mapa donde aparecen los jugadores del equipo *Terrorist (T)*, podemos observar dicha zona en el círculo amarillo de la Figura 5.
- **Zona de bomba A**: se trata de la parte de una de las dos partes del escenario o mapa donde se debe colocar la bomba, podemos observar dicha zona en el círculo rojo inferior de la Figura 5.
- **Zona de bomba B**: se trata de la parte de una de las dos partes del escenario o mapa donde se debe colocar la bomba, podemos observar dicha zona en el círculo rojo superior de la Figura 5.

Figura 5: Diseño del mapa *de_inferno*

A continuación conviene explicar el papel que desempeña cada bando en la partida:

- **CounterTerrorist**: Este bando es el encargado de defender, es decir, su objetivo será bloquear la entrada del bando contrario a estos perímetros o zonas para activar la bomba. Si no consiguen realizar el bloqueo siempre pueden acabar con el equipo rival y desactivar el dispositivo explosivo.
- **Terrorist**: Este bando es el encargado de atacar, es decir, su objetivo será acceder a un perímetro o zona para activar la bomba. Si no consiguen hacerla explotar siempre pueden acabar con el equipo rival.

La ronda terminará cuando; a uno de los bandos no le queden jugadores vivos, en cuyo caso ganará el equipo contrario (al acabar con la vida de un jugador, este no vuelve a jugar hasta la ronda siguiente), o bien, la bomba haya explotado (en este caso ganará el equipo Terrorist) o se haya desactivado (en este caso ganará el equipo CounterTerrorist) o haya terminado el tiempo de la ronda (en este caso ganará el equipo CounterTerrorist).

El tiempo de cada ronda es de 2 minutos, que podemos dividir en las siguientes fases:

- **Tiempo de compra**: cada bando dispone de los primeros 15 segundos para comprar equipamiento y así prepararse para el enfrentamiento.
- **Tiempo de ronda**: tras estos 15 segundos el tiempo restante se dedicará al desarrollo de la ronda, que terminará de cualquiera de las formas que se ha explicado anteriormente.
- **Tiempo para desactivar la bomba**: sea cual sea el tiempo restante de la ronda, tras activar la bomba (esto puede o no ocurrir), se establecerán 30 segundos para que el equipo CounterTerrorist alcance la zona y la desactive. El tiempo para desactivarla es de 10 segundos, que pueden ser reducidos a 5 si se hace uso de un kit antiexplosivo, que han tenido que adquirir durante el tiempo de compra.

2.4. Economía y roles

La economía y los roles que desempeña cada jugador dentro de cada equipo es realmente importante y a veces confusa, por ello es necesario entrar en detalle.

- **Economía**: cada bando cuenta con una cantidad de dinero inicial para realizar compras de armas, blindaje, granadas, etc. Con el objetivo de estar mejor equipados en el siguiente combate. Cuando un jugador acaba con la vida de otro obtiene una recompensa de dinero, además, el bando vencedor de la ronda será bonificado con un cantidad extra.
- **Roles**: en las partidas profesionales cada jugador suele desempeñar un rol distinto.
    1. **Líder**: se encarga de organizar y dirigir al equipo.
    2. **Apoyo**: se encarga de ofrecer apoyo, normalmente mediante el uso de granadas.
    3. **Francotirador**: se encarga de utilizar armas de tipo francotirador.
    4. **Punta de lanza**: se encarga de realizar la primera baja en la ronda, suele ser el primero en acceder a las zonas para colocar la bomba.
    5. **Acechador**: se encarga de cortar las rotaciones del equipo rival (movimientos que realiza el otro equipo para ayudar a sus compañeros).

2.5. Grabaciones

Al finalizar cada partida, se genera una grabación de esta en una de las carpetas donde se instaló el juego (esto ocurre tanto en torneos como en cualquier otra partida ya que se guarda en el propio ordenador). Dicha grabación se trata de un fichero en formato .DEM, compuesto por una cabecera con información sobre la fecha, mapa, protocolos e información sobre la partida y el motor del juego.

El resto del archivo esta compuesto por la información de cada una de las rondas que se jugaron en la partida. Son estos archivos los que se utilizan para realizar la captación y posterior análisis de los datos.

3

Desarrollo y fases del trabajo

En este capítulo se explican y desarrollan las diferentes fases del proyecto aplicadas a nuestro problema en particular, atendiendo a la metodología empleada.

3.1. Comprensión del problema

Para empezar, en esta primera parte resulta necesario comprender los objetivos y estudiar los requisitos imprescindibles para posteriormente transformar los datos o conocimientos extraídos y lograr la consecución de estos objetivos, es decir, primero hay que dejar claro que el objetivo será encontrar reglas, patrones y datos estadísticos que describan el funcionamiento y estado actual del juego ya que este varía con el tiempo según las diferentes actualizaciones que se realizan cada año.

Es precisamente esto lo que se pretende abordar, se estudiará cada año desde 2019 hasta 2021 para conocer los cambios de paradigma del juego, esto es, la forma que tienen los jugadores de participar en el y el comportamiento de estos durante la partida.

Concretamente se tiene como objetivo encontrar información que explique la relación que existe entre los diferentes bandos de la partida (ver sección 2.3), el ganador de la ronda y el mapa jugado (ver sección 2.3), también se busca comparar diferentes modelos aplicados a nuestro problema para encontrar cual de ellos se adapta mejor (precisión) y cual de ellos es más rápido y con un menor coste computacional para utilizarlo como posible modelo predictivo y así conocer el ganador de una contienda.

Otro objetivo fundamental, comentado previamente, es observar los comportamientos (agresivo o defensivo por ejemplo) que toman los jugadores durante la partida y cuáles son los recursos (armas, granadas, blindaje, etc.) de los que más uso hacen durante el desarrollo del

encuentro.

Por otra parte, resulta también necesario definir una serie de requisitos para construir posteriormente un plan del proyecto, para ello lo más básico que se necesita es una fuente apropiada de datos de la que partir para posteriormente realizar una fase de preparación de datos, modelizado y evaluación e interpretación.

Por tanto, se ha realizado una búsqueda exhaustiva de las diferentes fuentes de información que pueden resultar útiles a la hora de realizar todos los pasos citados anteriormente, principalmente se han identificado tres tipos de fuente que se explican a continuación.

- **Páginas web**: esta fuente de datos es la que resulta más sencilla e intuitiva de entender, se trata de páginas que ofrecen una cantidad enorme de datos sobre el videojuego en cuestión, incluso, desde el año 2014 (año de salida oficial del juego).

  Sin embargo, esta fuente de datos aunque muy útil se descartó (aunque no por completo) para el trabajo, puesto que sería necesario realizar la extracción de datos de forma automática a través de peticiones a la página web, esta técnica se conoce como *web scrapping*, la desventaja de esta técnica es que no esta bien vista por parte de los dueños de las páginas web ya que aumentaría mucho el número de peticiones por segundo, el tráfico web y añadiría demasiado estrés a sus servidores.

  En este sentido, el web scrapping tampoco es ilegal puesto que las disputas legales siempre se han resuelto ha favor de los usuarios que utilizan la técnica, además de que en España no existen impedimentos legales para realizarla [1], puesto que al final se adquiere de una forma legítima datos que son puestos a disposición del público en una web de Internet.

  El motivo de no usar esta técnica como se ha explicado anteriormente es la de evitar tanto problemas legales como problemas con los dueños de las web. Esta fuente de datos no se terminó de descartar pues se utilizará para descargar las grabaciones en bruto de las partidas a analizar.

- **Interfaces de programación de aplicaciones (APIs)**: esta fuente de datos consiste en la utilización de una librería basada en peticiones a una base de datos (esta sería por tanto la fuente de información), gracias a esto se puede programar de forma sencilla

un software dedicado a la comprensión y preparación de datos sin entrar en problemas legales.

Sin embargo, esta idea también se descartó para nuestro problema pues la variedad de datos y el numero de peticiones no son suficientes. Existen versiones mucho mejores y más completas, sin embargo, requieren de un desembolso económico en base a suscripciones considerablemente grande.

- **Archivos .DEM**: esta tercera y última fuente de datos es la que finalmente se utiliza durante el desarrollo del trabajo. Estos archivos con extensión .DEM tienen una estructura sencilla tal y como se explica en la sección 2.5.

  Esta fuente de datos es la fuente más original y menos sesgada que se puede consultar ya que se trata de la propia partida en sí misma. La recopilación de datos de esta fuente se realizará a través de librerías *open-source* puesto que facilitan enormemente el trabajo con este tipo de archivos, no siendo necesario el estudio profundo de su estructura de datos.

Una vez que se han explicado las posibles opciones sobre las fuentes de datos y se ha seleccionado una de ellas para su uso (grabaciones a través de librerías open-source), ya disponemos de una base de la que partir a la hora de realizar nuestro plan para el proyecto.

En concreto el plan consistirá en identificar qué librería hay que utilizar y qué lenguaje de programación emplear para realizar las siguientes fases de la metodología aplicada. Posteriormente será necesario recopilar una gran cantidad de grabaciones de partidas jugadas para después utilizar la mencionada librería en las siguientes fases, llegados a este punto es necesario entender que datos se recopilarán y porqué (esta parte se desarrollará en la sección 3.2).

Una vez que ya tenemos tanto los datos originales que vamos a emplear (grabaciones) como la herramienta, como una idea de los datos a extraer, se construirá un software a partir de la librería para realizar la extracción y preparación de los datos. Una vez tengamos el software listo procederemos a extraer la información que necesitamos y cuando esta parte esté completada, se prepararán estos datos con el objetivo de obtener un conjunto de datos final (esta parte se desarrollará en la sección 3.3).

El siguiente paso será crear nuestros modelos (esta parte se desarrollará en la sección 3.4)

para finalmente evaluarlos e interpretarlos (esta parte se desarrollará en la sección 3.5). Como se puede observar la mayoría de fases de este plan se corresponden con la metodología empleada, que se resume de una forma más visual en la figura 6 (nótese que el plan para el proyecto no es exactamente lo mismo que la metodología empleada).

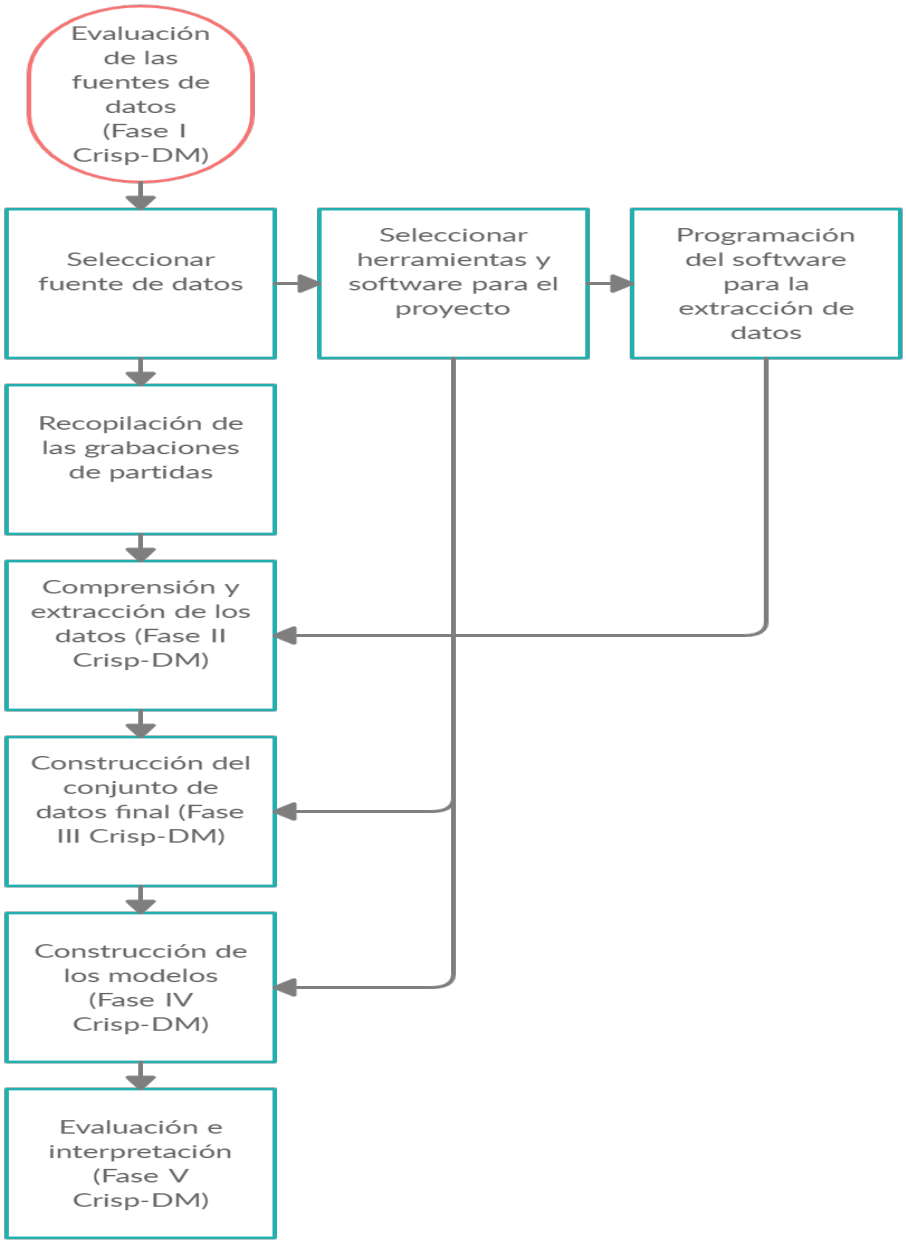

Figura 6: Diseño del plan para el proyecto

3.2. Comprensión de los datos

En esta fase, ya se han recolectado una gran cantidad de partidas, en concreto, se han recopilado 400 grabaciones que ocupan un espacio total de 150GB. Estas grabaciones suponen a priori, un conjunto de datos lo suficientemente grande como para poder inferir conclusiones y extraer patrones, esto es el objetivo final del trabajo.

Las grabaciones se han recopilado de la página web HLTV [7], la cual nos provee de dichas partidas desde el año 2014. En concreto, las partidas recolectadas pertenecen a la *ESL Pro League* [10], es decir, la liga profesional de *CSGO* más reconocida y donde participan los mejores equipos del mundo.

A continuación y antes de desarrollar el software que permitirá extraer los datos de esta fuente de información en bruto, es necesario explicar qué atributos se recolectarán y porqué. En este sentido, se ha escogido la liga profesional mencionada anteriormente con la finalidad de evitar información poco relevante o sesgada derivada de por ejemplo, una extracción de datos de partidas de jugadores *amateur*.

Esto se debe a que los jugadores profesionales entrenan mucho más y su perspectiva del juego es superior, además y como se ha comentado anteriormente cada equipo esta compuesto por 5 jugadores y cada uno de estos desempeña un rol distinto (esto suele ser diferente en el caso de competiciones *amateur*) lo que permite estudiar mejor el impacto de cada uno de estos roles durante la partida.

Considerando la información suministrada por los expertos en la materia se ha seleccionado hasta 41 atributos que pueden resultar útiles para el estudio que se va a llevar a cabo.

Estos atributos están explicados uno a uno en la sección 6, estos atributos hacen referencia al dinero del que disponen los jugadores, supervivientes de cada bando, tiempo empleado en la ronda, rondas ganadas, etc. Es importante destacar que el atributo *roundWinner* será el atributo de clase ya que proporciona información sobre el ganador de la ronda. Este atributo será nuestro atributo objetivo puesto que el interés del estudio es analizar qué planteamientos deben seguir los jugadores y qué condiciones deben ocurrir para ganar la ronda.

Una vez que se han explicado los atributos que se pretenden recoger y además se ha comentado porqué se han escogido partidas relacionadas con el ámbito profesional, a continuación

es necesario definir que librería y lenguaje de programación se utilizarán para desarrollar el software que realice la extracción de datos.

La librería que se ha escogido se denomina *demoInfo* [11]. Se trata de una librería con licencia *MIT* (la cual nos permite hacer uso de ella sin problemas legales). Esta librería nos facilita leer los archivos con extensión .DEM (grabaciones de las partidas). Está escrito en el lenguaje C#, orientado a objetos y basado en eventos, es decir, la librería es capaz de monitorizar los eventos que ocurren durante el desarrollo de la partida.

Un evento no es más que una situación relevante que se realiza en una ronda, por ejemplo: lanzar granadas, activar la bomba, desactivar la bomba, inicio de la ronda, final de la ronda, eliminación de un jugador, etc.

El desarrollo del software consistirá por tanto, en crear un flujo de aplicación que nos permita leer todos estos eventos por cada ronda que se juegue en la partida y extraer la información que consideremos relevante, en este caso, el objetivo será extraer los 41 atributos citados anteriormente.

Para comenzar a explicar el software desarrollado, es necesario, entender el funcionamiento del programa principal, es decir, aquel que se ejecutará primero. Tal y como se observa en la Figura 7 podemos ver tres zonas importantes en la clase *main*. La primera de ellas (resaltada en azul) se encarga de crear el objeto para analizar las grabaciones (este objeto es una instancia de la clase creada para llevar a cabo el análisis mediante la monitorización de eventos).

Por otra parte, la segunda zona del código (resaltada en verde) se encarga de recorrer un directorio donde se almacenan las partidas, posteriormente se crean 4 variables que guardan el nombre de grabaciones distintas, el motivo de esto se observa en la siguiente zona (resaltada en rojo) en la que se ve la ejecución de cada una de esas partidas mediante la función *ExecuteParser* almacenando en una ruta en concreto el archivo con extensión .csv creado tras analizar la grabación por completo. Como hemos comentado las partidas se analizan de 4 en 4, haciendo uso de la librería *Parallel* mediante la función *invoke*.

El paralelismo resulta muy necesario en este trabajo ya que permite extraer toda la información consumiendo muchísimo menos tiempo, en concreto, la eficiencia paso de ser de 4-5 minutos por cada partida, a 10 minutos por cada 4 partidas (no se entrará en detalles de como funciona esta librería puesto que no es el objetivo del trabajo, pero puede consultarse en el siguiente recurso [13]).

Después de extraer la información de todas las grabaciones, es necesario unirlas en un solo archivo con extensión .csv para realizar el estudio del conjunto de datos completo, esta es la función de la segunda parte del programa principal o *main*, sin embargo, esto se desarrollará en la sección 3.3.

```
class Driver
{
    0 referencias
    static void Main(string[] args)
    {
        //Object that extracts the data from the replays
        CSGODemoParser csgoDemoParser = new CSGODemoParser();

        // Process the list of files (.DEM) found in the directory. Ref: https://docs.microsoft.com/en-us/dotnet/api/system.io.directory.ge
        string[] fileEntries = Directory.GetFiles(@"D:\Demos-2020");
        for (int i = 0; i < fileEntries.Length; i+=4){
            Console.WriteLine("[+] Processing demos "+(i+1)+"-"+(i+4)+"/"+ fileEntries.Length); //Im processing the replays 4 by 4 but it d

            String firstDemo = fileEntries[i];
            String secondDemo = fileEntries[i+1];
            String thirdDemo = fileEntries[i+2];
            String fourthDemo = fileEntries[i+3];

            //Execute the program in parallel using 4 cores of my computer (again, it depends on the computer).
            //The index [13..] is to avoid the complete name of the demo's directory so it can find the correct path
            Parallel.Invoke(() => { csgoDemoParser.ExecuteParser(@"D:\ResultadosDemos-2020" + firstDemo[13..] + ".csv", firstDemo); },
                            () => { csgoDemoParser.ExecuteParser(@"D:\ResultadosDemos-2020" + secondDemo[13..] + ".csv", secondDemo); },
                            () => { csgoDemoParser.ExecuteParser(@"D:\ResultadosDemos-2020" + thirdDemo[13..] + ".csv", thirdDemo); },
                            () => { csgoDemoParser.ExecuteParser(@"D:\ResultadosDemos-2020" + fourthDemo[13..] + ".csv", fourthDemo); });
        }
```

Figura 7: Primera parte del código principal de la aplicación

Ahora que se ha establecido el flujo principal, es necesario conocer el interior del método que realiza el análisis, es decir, la función *ExecuteParser*.

Tal y como se observa en la Figura 8 se distinguen de nuevo 3 zonas, la primera de ellas (resaltada en azul) se encarga de crear un puntero a la ruta donde se van a escribir los datos recogidos (el método *GenerateCSVHeader* escribe la primera fila en el conjunto de datos, es decir, el nombre de los atributos), posteriormente se crea un objeto denominado *parser* (este objeto se proporciona en la librería con la que se trabaja). En la siguiente zona (resaltada en verde) se observa como se utiliza la función *ParseHeader* para omitir la parte de la cabecera de los archivos .DEM (para más información consular la sección 2.5).

Seguidamente se crea un objeto denominado *matchStatus* (tercera zona resaltada en rojo)

el cual permite representar el estado de cada ronda, esto es por ejemplo: número de jugadores vivos, granadas lanzadas, bombas desactivadas, etc. (se hablará de esta clase más adelante).

Tal y como se observa, se definen todos los eventos necesarios para realizar la extracción de datos, estos eventos los proporciona la librería para programar el flujo de aplicación (este flujo se puede observar dentro de cada evento en el código que se adjunta junto con este documento, no se explicará aquí debido a su extensión).

Finalmente se utiliza la función *ParseToEnd*, la cual nos permite monitorizar ronda a ronda los eventos de la partida.

```
public CSGODemoParser() { }

4 referencias
public void ExecuteParser(String outputPath,String demoPath)
{
    StreamWriter outputStream = new StreamWriter(outputPath);
    outputStream.WriteLine(GenerateCSVHeader());

    //Object to parse the file.dem to retrieve the data
    DemoParser parser = new DemoParser(File.OpenRead(demoPath));
    //Parse the first bytes of the header in order to skip them
    parser.ParseHeader();

    //Object to represent the state of the match in each round
    MatchStatus matchStatus = new MatchStatus(parser.Map);

    //Events
    parser.MatchStarted += (sender, e) => ...

    parser.LastRoundHalf += (sender, e) => ...

    parser.RoundStart += (sender, e) => ...

    parser.PlayerKilled += (object sender, PlayerKilledEventArgs e) => ...

    parser.FreezetimeEnded += (sender, e) => ...

    parser.BombPlanted += (sender, e) => ...

    parser.RoundEnd += (sender, e) => ...

    parser.RoundOfficiallyEnd += (sender, e) =>...


    //Parse up to the end of the demo
    parser.ParseToEnd();
    //close the stream
    outputStream.Close();
}
```

Figura 8: Código de la clase CSGODemoParser

A continuación, en la Figura 9 se observan algunas funciones de la clase *matchStatus* (zona resaltada en azul).

- **MatchStatus**: se trata del constructor de la clase, su función es instanciar los atributos de esta.
- **SetCompetence**: permite establecer el valor para el atributo *roundCompetence* (para más información consultar la sección 6).
- **ClearRoundInfo**: permite reiniciar los valores de los atributos después de cada ronda.
- **FeedManagement**: permite analizar la retroalimentación (ver Figura 10) que obtenemos en cada ronda obtenida a partir del enfrentamiento entre los jugadores (zona resaltada en verde).
- **ToString**: se utiliza para añadir al conjunto de datos los valores analizados por cada ronda.

Es necesario añadir también que los 41 atributos se encuentran creados como variables en esta misma clase para llevar a cabo el análisis (no se observa en la figura 9 dada su extensión).

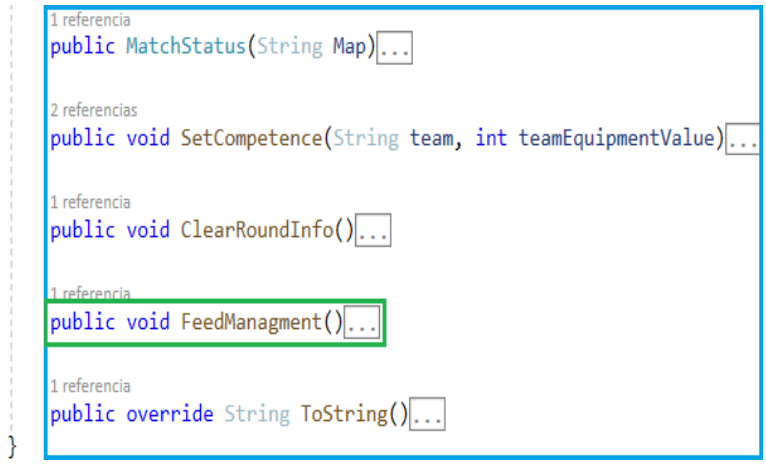

Figura 9: Código de la clase MatchStatus

El método *FeedManagement* utiliza una lista de objetos denominados *FeedRegister* para representar cada uno de los registros de la Figura 10.

Figura 10: Ejemplo de *KillFeed* de una ronda

La clase se puede observar en la Figura 11, es muy intuitiva ya que solo dispone de algunos atributos y un constructor explicados en la propia imagen.

```
public class FeedRegister
{
    public Player killer; //The person who made the kill
    public Player victim; //The person who was killed
    public string weaponClass; //The class of the weapon (rifle, sniper, etc...)
    public string weaponName; //The name of the weapon (ak47, glock, etc...)
    public bool headshot; //If the kill was a headshot
    public bool flashed; //If the person that died was blinded
    public float timeOfTheKill; //How long until the person died

    1 referencia
    public FeedRegister(Player killer, Player victim, string weaponClass, string weaponName, bool headshot, bool flashed, float timeOfTheKill)
    {
        this.killer = killer;
        this.victim = victim;
        this.weaponClass = weaponClass;
        this.weaponName = weaponName;
        this.headshot = headshot;
        this.flashed = flashed;
        this.timeOfTheKill = timeOfTheKill;
    }
}
```

Figura 11: Código de la clase FeedRegister

En este punto del trabajo, ya se ha creado el software necesario para la extracción de los datos. Al ejecutar el programa para analizar todas las partidas se generan tres ficheros (uno por cada año desde 2019 hasta 2021) con extensión .csv.

La ejecución del programa ha durado alrededor de 30 horas debido a fallos en el flujo de la aplicación que han tenido que ser resueltos, además, alrededor del 10 % de las grabaciones se encontraban corruptas y el análisis no se realizó correctamente por lo que se invirtió tiempo extra para solucionar los problemas y volver a realizar la extracción, aunque algunas partidas no pudieron recuperarse.

En total se han obtenido 9283 registros entre los diferentes archivos, es decir, ya disponemos de un número considerablemente grande de información como para proceder con la siguiente fase.

3.3. Preparación de los datos

Como se ha hablado al inicio de este capítulo, se busca analizar el conjunto de datos de cada año desde 2019 hasta 2021 y también el conjunto de datos unificado para extraer conclusiones y alcanzar los objetivos propuestos.

Antes de realizar esta última tarea es necesario revisar los datos extraídos y eliminar los datos anómalos, es decir, aquellos que debido a inconsistencias con las grabaciones, datos corruptos, atributos con valores *N/A* u otras razones, no se extrajeron correctamente. Tras realizar este paso el numero de registros totales entre los 3 conjuntos de datos es de 9202.

Para realizar la unión de los 3 conjuntos de datos en uno solo se utiliza la última parte del código del programa principal del software desarrollado y explicado en la fase previa.

Tal y como se observa en la Figura 12, existen 2 zonas que resultan interesantes comentar. Para empezar, la primera zona (resaltada en azul) se encarga de establecer la ruta origen de los conjuntos de datos extraídos anteriormente (además de iterar sobre este directorio) y la ruta destino del conjunto de datos unido. Por otra parte, la segunda zona (resaltada en verde) se encarga de unir los ficheros respetando la cabecera.

```
Console.WriteLine("[+] Merging all the .csv files!");

//// Process the list of files (.csv) found in the directory. Ref: https://chris.koester.io/index.php/2017/01/27/combine-csv-files/
string sourceFolder = @"D:\DataSets";
string destinationFile = @"D:\DataSets\DataSet.csv";

// Specify paths to combine all the .csv files
fileEntries = Directory.GetFiles(sourceFolder);
StreamWriter outputStream = new StreamWriter(destinationFile, true);

//Merging all the files
for (int i = 0; i < fileEntries.Length; i++){
    string file = fileEntries[i];
    string[] lines = File.ReadAllLines(file);

    if (i > 0){
        lines = lines.Skip(1).ToArray(); // Skip header row for all but first file
    }

    foreach (string line in lines){
        outputStream.WriteLine(line);
    }
}
outputStream.Close();

//At this point of the code the data (.DEM) has been gathering in a file called "DataSet.csv"
Console.WriteLine("[+] The dataset is ready!");
```

Figura 12: Segunda parte del código principal de la aplicación

A continuación, es necesario realizar una limpieza de los datos para poder obtener los conjuntos de datos que se utilizará en la siguiente fase de modelizado. Para llevar a cabo esta tarea se ha utilizado el software de *RapidMiner* (para más información consultar sección 1.4).

Gracias a este software se puede realizar la tarea de limpieza de una forma mucho más sencilla e intuitiva tal y como se observa en la Figura 13. En este apartado de la herramienta (*TurboPrep*, resaltado en azul) se realizará este proceso con cada uno de los conjuntos de datos (en la Figura 13 se ilustra el ejemplo con el del año 2019) mediante la opción de *Auto Cleansing* (resaltado en rojo).

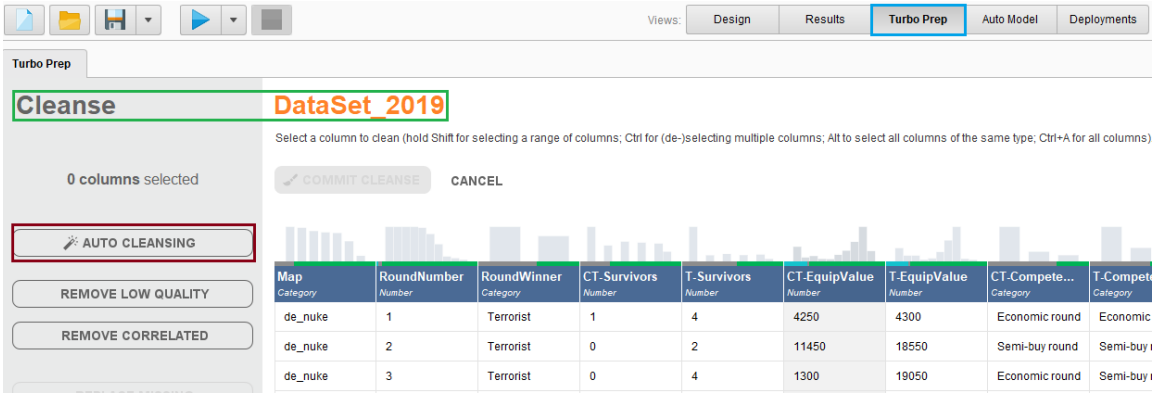

Figura 13: Funcionalidad de limpieza del software RapidMiner

Una vez se accede a la opción, esta presenta una interfaz muy intuitiva que guía a través de unos pasos para realizar la limpieza. Tal y como se observa en la Figura 14, la herramienta resalta algunos atributos que son muy estables, es decir, que en la gran mayoría de registros poseen el mismo valor y que por tanto no resultan de utilidad, así que se procederá a eliminar dichas columnas.

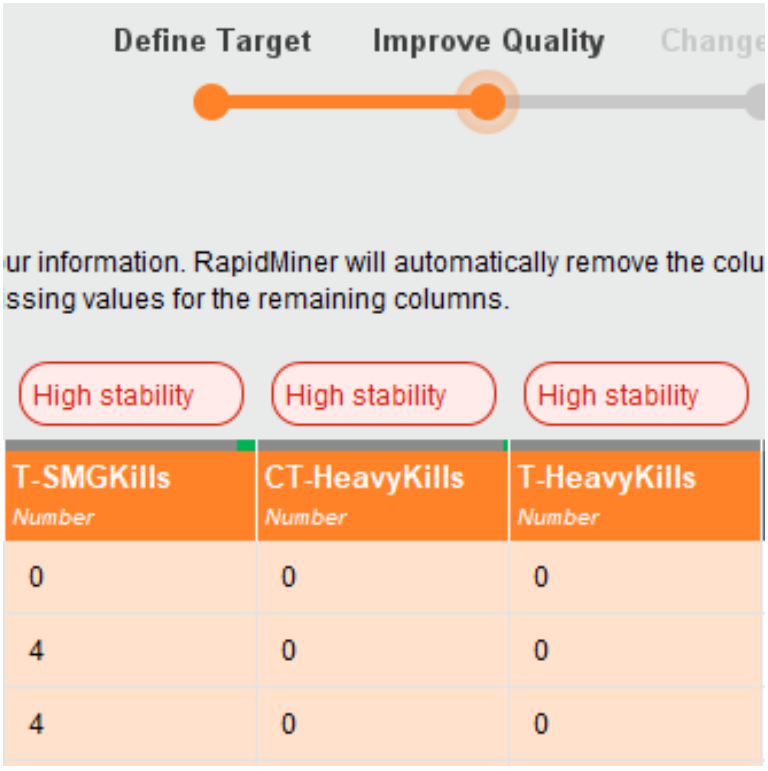

Figura 14: Guía para realizar la limpieza de datos

La opción anterior ayuda a prescindir de los atributos que no explican nada debido a su estabilidad, sin embargo, el proceso de limpieza aún no ha terminado porque es necesario también eliminar aquellos atributos que presenten una correlación muy alta.

Para este propósito vamos a utilizar la opción de *Auto Model* que nos proporciona también *RapidMiner*. Esta opción es la que se utiliza en la siguiente fase pero también se seleccionan aquellas columnas con un grado de correlación muy alto y que son susceptibles de ser eliminadas. Este proceso se puede observar en la Figura 15 (no todos los atributos con esta característica han sido eliminados, solo aquellos que sugerían el software y el experto).

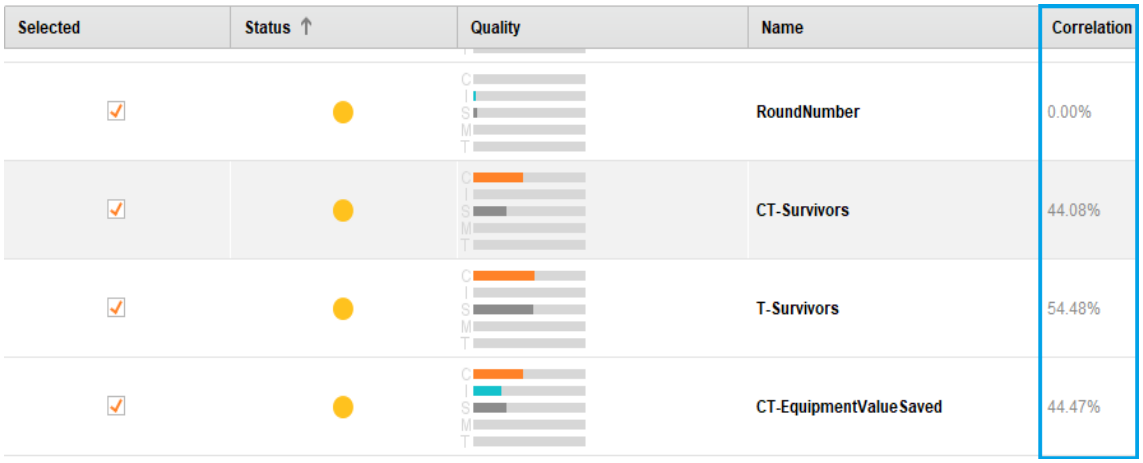

Figura 15: Selección de atributos debido a su correlación

Tras toda la transformación llevada a cabo con anterioridad, el número de atributos se ha reducido desde 41 hasta 25, es decir, ahora los conjuntos de datos están preparados para ser utilizados en la siguiente fase.

3.4. Modelizado

En este punto del trabajo, ya disponemos de los conjuntos de datos listos para ser introducidos en la etapa de modelizado.

Para empezar, se utilizará la herramienta *Weka*, ya que ofrece información estadística interesante sobre los distintos conjuntos de datos.

En particular, si se introduce en la herramienta los conjuntos de datos extraídos para cada año se puede ver un patrón interesante. Tal y como se observa en la Figura 16, existe un escenario de los 7 posibles que se pueden jugar en la competición en el que se participa mucho

menos, este es el caso de *de_vertigo*.

Como se ha comentado, esto es una norma a través de los 3 años que se han analizado, sin embargo y como es lógico, el uso de dicho escenario ha ido creciendo periódicamente desde 2019 hasta 2021 aunque sigue siendo muy poco jugado en entornos profesionales comparado con el resto de los 6 mapas. Habiéndose jugado en 667 rondas de las 9202 del conjunto de datos total, esto representa solo un 7,24 %.

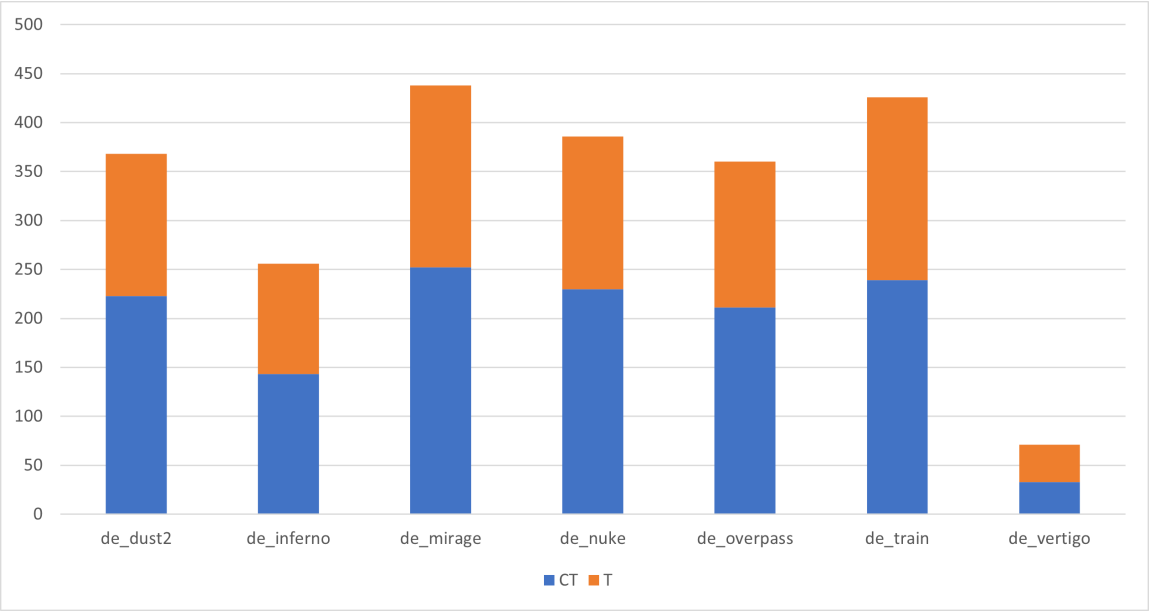

Figura 16: Partidas jugadas en cada mapa durante 2019

Con estos mismos datos estadísticos, se puede destacar otra característica que ha ido evolucionando y que resulta ser muy importante: el balanceo del juego.

Claramente en la Figura 16 anterior se observa como en determinados escenarios, como por ejemplo, *de_dust2* la barra azul tiene más peso que la barra roja. Esto quiere decir que el bando *CounterTerrorist* consigue un número mayor de victorias en las rondas que el bando *Terrorist* (alrededor de un 61 % en los datos extraídos de 2019), sin embargo, esto debería ser más bien cercano a un 50 % para que ambos bandos se encuentren en igualdad de condiciones.

En la Figura 17, se observa un cambio con respecto a lo dicho anteriormente, es decir, en 2019 existía un desequilibrio que más tarde (en los años 2020 y 2021) se corrigió hasta llegar a valores más cercanos al 50 %.

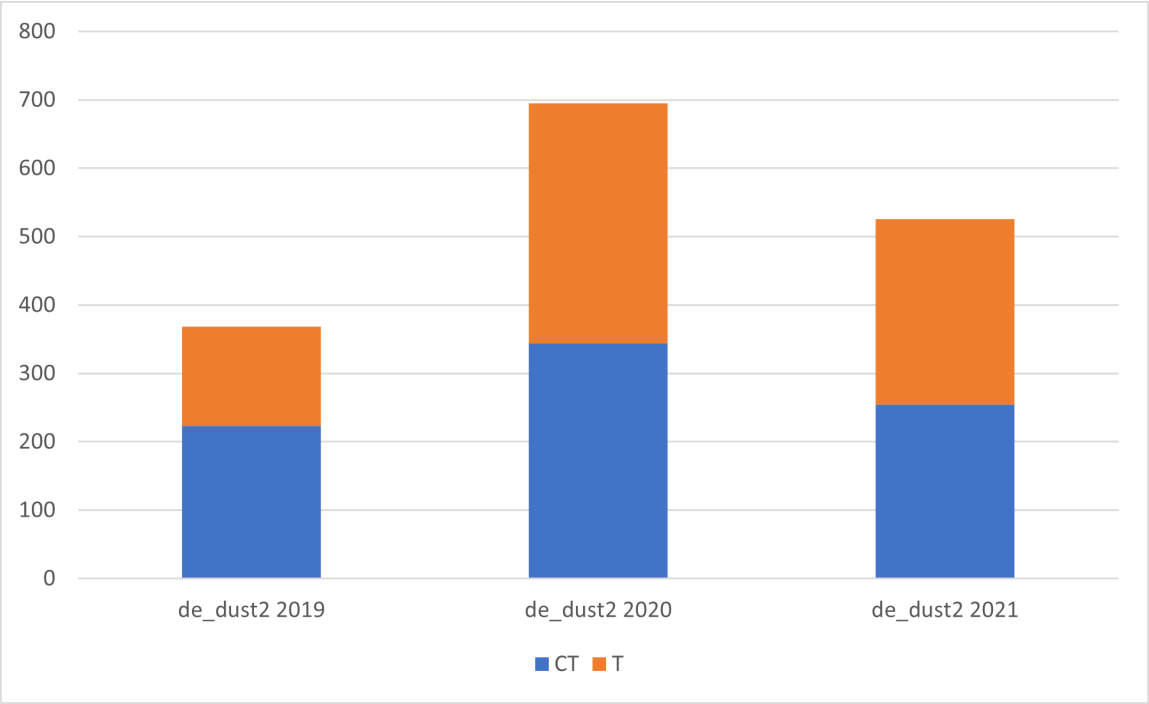

Figura 17: Partidas jugadas en el mapa *de_dust2*, 2019-2021

Este desequilibrio comentado anteriormente es algo que se da en la mayoría de mapas, es decir, un bando tiene más probabilidades de ganar la ronda por el simple hecho de pertenecer a el. Ambos equipos cambian de bando a la mitad del partido, sin embargo, psicológicamente puede ser de provecho empezar en el bando con ventaja a priori.

Como se ha comentado 2021 es de los 3 años analizados el que mayor equilibrio presenta en los escenarios.

Otro dato que resulta ser muy importante para los equipos esta relacionado con el atributo *OpeningKill*, es decir, qué bando realiza la primera eliminación en la ronda.

Tras analizar el conjunto de datos al completo se extrae la conclusión de que aquel bando que realice la primera baja en la ronda tendrá un 74,8 % de posibilidades de ganar dicha ronda. Esta información resulta de vital importancia y sobretodo se ve acentuada cuando ambos equipos se han equipado lo suficientemente bien para el enfrentamiento.

Por otro lado, se utilizará el programa *RapidMiner*, el cual usa algunas implementaciones de los algoritmos de los que dispone *Weka*. En estos algoritmos de decisión se ha empleado validación cruzada, utilizando el 90 % del conjunto como datos de entrenamiento. Y se han realizado 10 repeticiones (*Folds*).

Los algoritmos seleccionados para realizar el modelizado para esta fase son los siguientes.

- **NaiveBayes**: clasificador probabilístico fundamentado en el teorema de Bayes.
- **Logistic Regression**: análisis de regresión utilizado para predecir el resultado de una variable categórica.
- **Generalized Linear Lodel**: generalización flexible de la regresión lineal ordinaria.
- **Fast Large Margin**: clasificador lineal binario no probabilístico.
- **Deep Learning**: conjunto de algoritmos de aprendizaje automático para modelar abstracciones de alto nivel.
- **Decision Tree**: modelo de predicción mediante estructuras de datos basadas en árboles.
- **Random Forest**: combinación de un conjunto de árboles para clasificación y regresión.
- **Gradient Boosted Tree**: técnica de aprendizaje automático utilizado para el análisis de la regresión y clasificación estadística.

El objetivo de modelizar los datos con estos algoritmos es realizar una comparación entre ellos, sin embargo, antes de desarrollar este paso es necesario configurar los parámetros que se utilizarán en los algoritmos.

Tal y como se observa en la Figura 18, se han seleccionado los algoritmos anteriores, configurando también las opciones para optimizar los algoritmos automáticamente y el número de árboles que se generarán con los algoritmos de *Random Forest* y *Gradient Boosted Tree* (se han escogido 10 puesto que se utilizan varios conjuntos de datos).

Además, es importante destacar que se ha desactivado el algoritmo de Máquina de Soporte Vectorial puesto que la duración para extraer el modelo era considerablemente grande, incluso la propia herramienta advierte de esto.

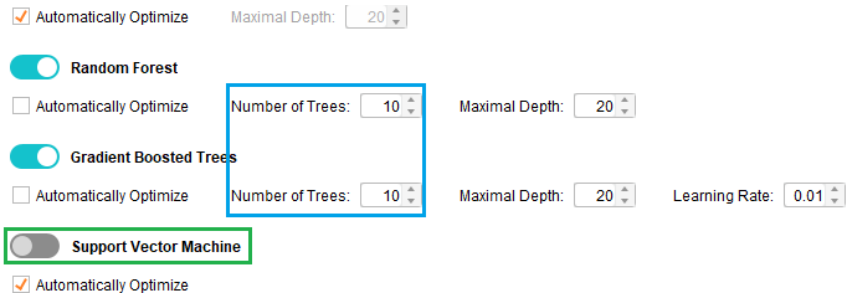

Figura 18: Selección de parámetros con la herramienta *Rapid Miner*

Una vez escogido los algoritmos y configurado algunos de sus parámetros, es necesario, configurar otras opciones sobre los propios datos y no sobre los modelos para poder llevar a cabo la evaluación.

Como se observa en la Figura 19, se ha de seleccionado la opción de eliminar columnas con un número de valores muy alto, esto debe realizarse de esta manera para no perder demasiada información que puede resultar útil de cara a extraer reglas y patrones.

También se han seleccionado las opciones sobre el análisis de las columnas puesto que esto nos permitirá llevar a cabo una comparación sobre las predicciones, esto es, comparar el coste, la precisión, etc.

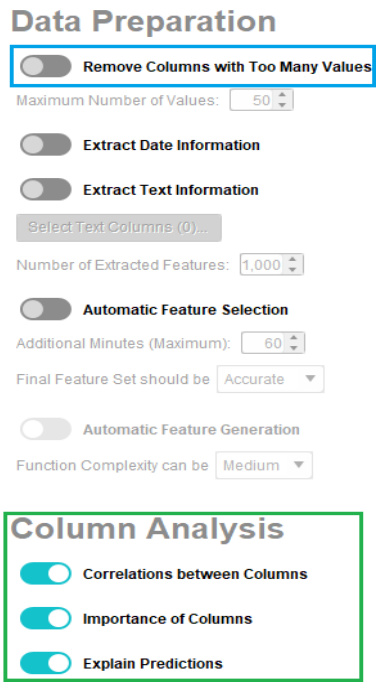

Figura 19: Selección de opciones sobre datos con la herramienta *Rapid Miner*

Tras realizar los pasos anteriores, en la fase de modelizado se han extraído 168 modelos y entre ellos destacan 2 algoritmos por encima del resto en cuanto a tiempo y precisión se trata.

En concreto, como podemos observar en las Figuras 20 y 21, el algoritmo *Gradient Boosted Tree* destaca por ser el que mayor precisión tiene. El tiempo necesario para llevar a cabo el modelizado también es algo a destacar, en este caso el mejor algoritmo es *Naive Bayes*. Pero su precisión es tan baja que no compensa.

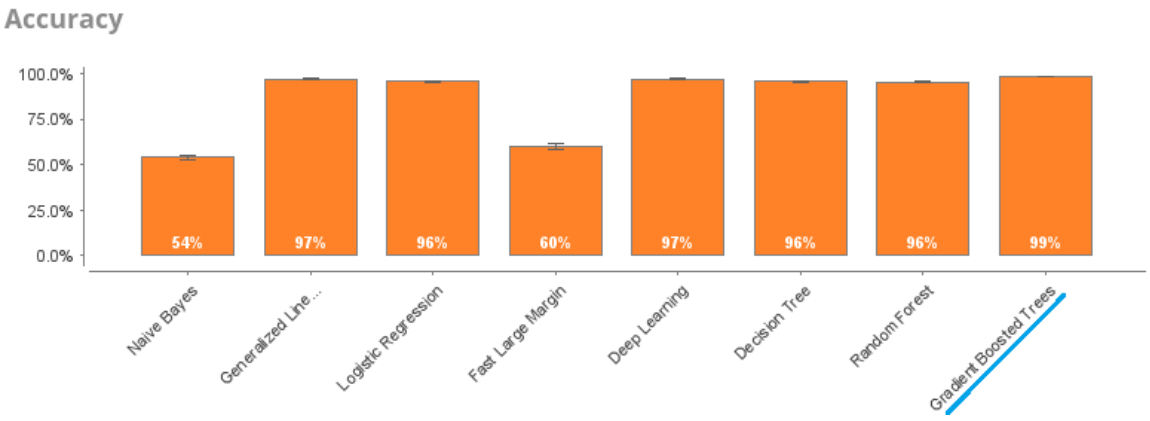

Figura 20: Comparación de algoritmos según la precisión

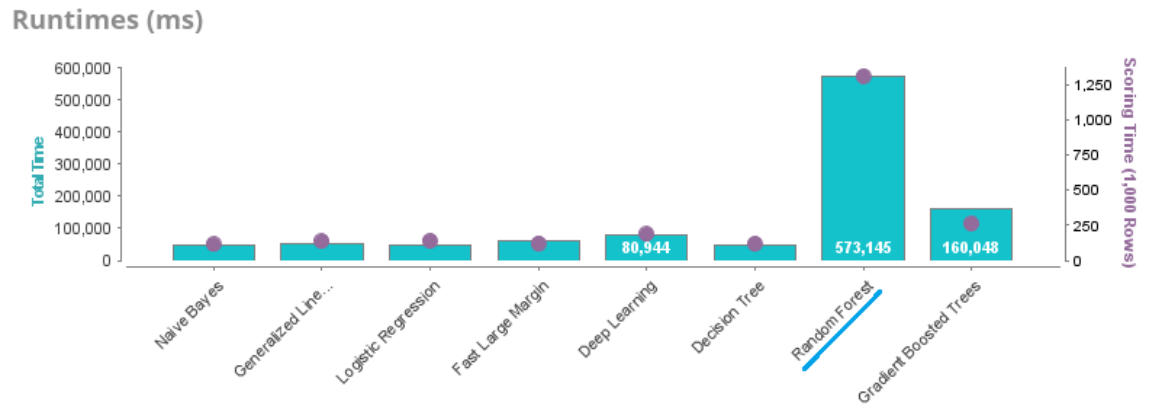

Figura 21: Comparación de algoritmos según el tiempo de modelizado

Para finalizar, en el Cuadro 1, se muestra una comparación entre los distintos parámetros obtenidos por los algoritmos empleados, es importante destacar que uno de los algoritmos que utilizaremos es el *Decision Tree* que aunque no resulta ser el mejor de todos sí que resulta ser rápido y poseer gran precisión, además nos permitirá extraer patrones de una forma comprensible.

Este término hace referencia al campo de la inteligencia artificial explicable (XAI), que es ampliamente reconocida como una característica crucial para el despliegue práctico de los modelos de IA. Se refiere a la utilización de técnicas en la aplicación de tecnología de inteligencia artificial (IA) tal que los resultados de los modelos generados a partir de estas pueden ser entendidos por humanos [2].

Model	Accuracy	Standard Deviation	Gains	Total Time	Training Time
Naive Bayes	54.1 %	1.3 %	0	48 s	4 ms
Generalized Linear Model	97.1 %	0.5 %	2,29	50 s	13 ms
Logistic Regression	95.9 %	0.5 %	2,18	48 s	19 ms
Fast Large Margin	60.2 %	1.4 %	294	1 min 1 s	154 ms
Deep Learning	97.2 %	0.6 %	2,274	1 min 20 s	453 ms
Decision Tree	95.8 %	0.6 %	2,228	49 s	11 ms
Random Forest	95.6 %	0.5 %	2,156	9 min 33 s	133 ms
Gradient Boosted Trees	98.7 %	0.2 %	2,316	2 min 40 s	290 ms

Cuadro 1: Tabla comparativa entre los algoritmos empleados

A continuación, se estudiará con la herramienta *RapidMiner* los modelos desarrollados anteriormente. En concreto se estudiaran los modelos extraídos mediante los algoritmos de *Random Forest* y *Gradient Boosted Trees*. El objetivo de esto es identificar los atributos raíz de los arboles generados por cada algoritmo para extraer conclusiones sobre su impacto en el juego y la evolución a lo largo de los años analizados.

Tras realizar este estudio, se destaca que, durante 2019 y 2020 los atributos más importantes para lograr la victoria por parte de los CT en una ronda son por un lado, *CT-SniperKill* (Número de eliminaciones con francotirador por parte del bando *CounterTerrorist*) lo que resulta lógico ya que el bando *CT* es el encargado de defender el mapa y por tanto el uso de armas a larga distancia es muy conveniente.

Por otro lado también se destaca el atributo *BombPlanted* (Activación de la bomba), es decir, esto nos da información sobre la táctica que debería seguir el bando *Terrorist*.

Además, en el año 2021 se produce un ligero cambio. El atributo *BombPlanted* sigue siendo muy importante para alcanzar la victoria por parte del bando *Terrorist*, sin embargo, para el

bando *CounterTerrorist* esto ha cambiado.

En este año resulta más importante el atributo *CT-RifleKills* o lo que es lo mismo, este bando parece obtener la victoria con mayor facilidad mediante el uso de armas a media distancia. Esto no quiere decir necesariamente que este bando tenga que cambiar la táctica y pasar a ser más agresivo, pero podría ser algo a tener en cuenta por los equipos.

Finalmente, se estudiará el algoritmo *Decision Tree* con el conjunto de datos total para así observar algunas reglas que puedan resultar de interés.

Del árbol resultante, puede extraerse una regla que resulta de mucho interés. Tal y como se observa en la Figura 22, una parte del árbol arroja información sobre la táctica que debería tomar un equipo cuando se encuentra en rondas con poco nivel adquisitivo, es decir, no disponen de mucho equipamiento (no se incluye el árbol entero dada su extensión).

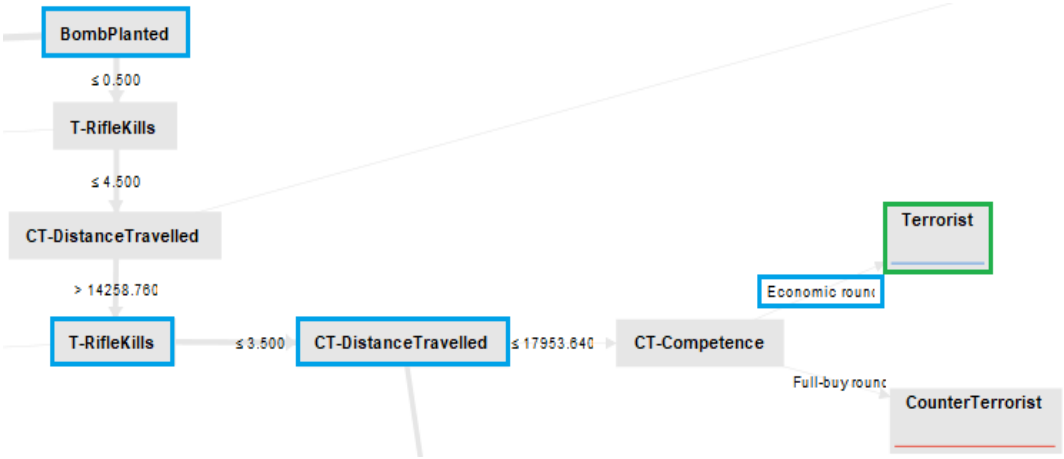

Figura 22: Regla extraída del árbol de decisión generado

En concreto, la regla nos dice que si la bomba no ha sido activada y además el número de bajas por rifle del equipo *Terrorist* es menor que 3 entonces, en este tipo de rondas el bando *CounterTerrorist* debería optar por ser más agresivo. Esto se evidencia por el atributo *CT-DistanceTravelled* que se encuentra en valores altos.

Una vez que se han extraído reglas, patrones y la información más relevante de la fase de modelizado, podemos proceder con la siguiente y última fase.

3.5. Evaluación e interpretación

En esta fase los expertos darán su opinión sobre los resultados y se evaluarán la consecución de objetivos dado el desarrollo de todas las fases anteriores.

Para empezar, en las Figuras 16 y 17 después de evaluarlas, se extraen diferentes conclusiones de las que pueden hacer uso los equipos de nivel profesional para enfocar sus estrategias.

Por un lado se evidencia cómo existe un escenario que claramente es mucho menos jugado que el resto. Esto es un resultado muy interesante ya que si se practicase mucho más el juego en equipo y las tácticas en este escenario concreto, podrían sorprender a los rivales y alzarse con la victoria.

Otra de las conclusiones que han quedado claras tras observar estas figuras, es la de que claramente existe un desbalanceo en los bandos de cada escenario. Por lo general el bando *CounterTerrorist* suele tener un mayor porcentaje de victorias en todos los escenarios jugados. Esto se debe en parte a que se encuentran posicionados más cerca de las zonas del mapa donde se debe activar el dispositivo explosivo.

Esto es algo lógico pues son el bando encargado de la defensa, sin embargo, las competiciones y sobretodo la compañía encargada del juego podría situar los puntos de aparición (ver sección 2.3) de este equipo ligeramente más lejos de las zonas comentadas anteriormente para que este bando tenga menos tiempo para posicionarse y por tanto exista una mayor igualdad.

Por otra parte, resulta lógico pensar que la propia topología de los mapas es la responsable de causar este desbalanceo. Esto se debe a la reducida cantidad de vías (normalmente 2) para acceder a las zonas importantes de los mapas. Por tanto, para solucionar en parte este desequilibrio sería necesario poner en práctica o bien otros mapas, o bien trabajar en los actuales haciéndolos más abiertos y espaciosos.

De todas maneras, los equipos deberían tener en cuenta este último dato y reforzar el bando *Terrorist* para obtener un mayor porcentaje de victorias en este bando y disfrutar de la ventaja que le proporciona el bando contrario al cambiar a mitad de partido.

Como se comentó en la sección 2.3, existen dos formas de ganar una ronda. O bien se acaba con el equipo rival, o bien se detona el dispositivo explosivo.

El resultado obtenido anteriormente donde observábamos que el atributo *BombPlanted* era muy importante resulta ser de mucha utilidad ya que suscita la idea de que es mejor seguir la

segunda vía que la primera.

En la fase anterior se habló sobre el atributo *OpeningKill* obtenido tras realizar el análisis estadístico con *Weka* y se observa como ambos bandos aumentan muchísimo sus probabilidades de ganar una ronda si consigue la primera eliminación.

Siendo interesante de por sí este último dato, resulta aún más útil cuando se pone en práctica con la táctica que se deriva de la regla obtenida tras analizar el árbol de decisión al final de la fase previa. Esta regla hacía referencia a que el bando *CounterTerrorist* debería ser agresivo (*CT-DistanceTravelled* alto) al principio de la ronda, cuando no se han producido eliminaciones (*T-RifleKills* bajo) y su nivel de equipamiento es bajo (*EconomicRound*).

La regla en sí lo que da a entender es que el equipo debe ser más agresivo al inicio de la ronda cuando no dispone de equipamiento suficiente porque suelen conseguir la primera eliminación, lo que aumenta sus probabilidades de victoria.

4

Conclusiones y Líneas Futuras

4.1. Conclusiones

Durante el desarrollo del trabajo se han cubierto los objetivos que se pretendían alcanzar.

Existen algunas reflexiones que se pueden llevar a cabo no solo sobre los datos obtenidos durante la experimentación, sino también sobre las herramientas empleadas en el mismo.

Para empezar, es necesario destacar que el uso de una metodología (CRISP-DM) ha sido realmente importante ya que ha facilitado la organización de los tiempos del trabajo y su separación en diferentes fases. Esto a permitido que en todo momento se tenga en mente la parte del trabajo que se esta desarrollando teniendo en cuenta las tareas a realizar y las herramientas a utilizar.

El uso de una metodología ha permitido también gestionar la relación entre coste y beneficio de los recursos invertidos (como el tiempo) en cada fase.

Por otro lado, sería buena idea modificar o incluso desarrollar otra librería para la comprensión y preparación de datos ya que durante el desarrollo del trabajo, la librería utilizada no tenía en cuenta ciertas características de las grabaciones de las partidas al no ser 100 % oficiales, esto se debe a que se analizó una liga profesional que no es propiedad de la compañía que creó el videojuego.

Por tanto las partidas contenían algunas modificaciones que tuvieron que tenerse en cuenta cuando se desarrolló el software, además de encontrarse con algunos archivos corruptos.

Por todo esto, crear un software que tenga estos detalles en cuenta y que incluso pueda analizar grabaciones de otros videojuegos hubiera facilitado mucho el trabajo.

Además, también se puso en práctica el paralelismo, técnica usada frecuentemente en tra-

bajos relacionados con la minería de datos dado el gran volumen de estos.

Se utilizó una librería externa creada para realizar estas funciones, sin embargo, esta librería dispone de opciones y realiza acciones para poder ejecutar el paralelismo en general. Si se hubiera implementado el paralelismo teniendo en cuenta únicamente las necesidades y características del software desarrollado, la fase de extracción de datos hubiera sido más rápida.

Durante la fase de modelizado, se estudiaron los algoritmos *Random Forest* y *Gradient Boosted Trees*, con el objetivo de identificar cambios de concepto. Esta parte llevó un tiempo considerable dada la gran cantidad de modelos que generan y combinan estos algoritmos, por tanto, automatizar la identificación de estos cambios de concepto analizando todos los árboles generados hubiera favorecido mucho a esta parte del trabajo.

Por todo lo expuesto, la minería de datos sobre competiciones deportivas electrónicas (*eSports*, por sus siglas en inglés) es un campo de investigación interesante y con proyección sobre el que merece la pena profundizar.

4.2. Líneas Futuras

Como trabajo futuro sería interesante llevar a cabo un estudio sobre algunas líneas que se plantean después haber realizado el estudio.

Para empezar, podría desarrollarse una aplicación que incorporé, o bien datos de este mismo trabajo, o bien nueva información para asesorar a los jugadores sobre las estrategias a tomar en base a sus rivales antes del comienzo de la partida.

Por una parte, todo lo expuesto en el trabajo ha partido desde la comprensión y preparación de datos de competiciones de la escena profesional, sin embargo, podría resultar útil estudiar también ligas de ámbito *amateur*.

El objetivo de esto puede ser por un lado, descubrir nuevas tácticas que pueden proporcionar jugadores no profesionales. Esto no suele ser lo común pero ya que la cantidad de jugadores profesionales es mucho menor que la cantidad de jugadores *amateur* o semi profesionales puede llegar a darse el caso.

También realizar este estudio podría ayudar a los jugadores más novatos o menos experimentados a mejorar sus habilidades en el juego.

Por otra parte y como segunda línea, se plantea realizar un estudio (quizás no tan técnico) sobre las características de los jugadores, esta idea surge porque en este videojuego en concreto

el número de hombres es muchísimo mayor que el número de mujeres.

A este respecto, *HLTV* (web que ofrece mucha información sobre el juego tratado en el trabajo) ofrece un ranking [8] donde clasifica a los mejores jugadores. En concreto en el ranking aparecen 743 jugadores distintos y el 100 % de ellos son hombres.

A primera vista y dado que las características de los deportes electrónicos son diferentes que los tradicionales (no existen a priori excesivas diferencias físicas entre sexos, sin embargo, puede que sí psicológicas) esta diferencia podría ser objeto de estudio.

Aunque puede que esto se deba a simplemente que las personas más atraídas por este tipo de videojuegos sean hombres y por tanto la cantidad de hombres que llegan a dedicarse profesionalmente al juego sea entonces muchísimo mayor por pura estadística.

Por último y no menos importante, existe la posibilidad de continuar con este mismo proyecto realizando más iteraciones (ya que así lo permite la metodología empleada) con el objetivo de conseguir otro tipo de resultados o utilizar otro enfoque.

5
Bibliografía

[1] *¿Es legal el scraping, crawling o raspado web en España?* URL: https://datstrats.com/blog/scraping-es-legal-espana/.

[2] Alejandro Barredo Arrieta y col. "Explainable Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". En: *Information Fusion* 58 (jun. de 2020), págs. 82-115. ISSN: 15662535. DOI: 10.1016/j.inffus.2019.12.012.

[3] *C Sharp - Wikipedia, la enciclopedia libre.* URL: https://es.wikipedia.org/wiki/C_Sharp.

[4] P. Chapman y col. "CRISP-DM 1.0: Step-by-step data mining guide". En: (2000).

[5] *Computational Intelligence and Knowledge 1.1 What Is Computational Intelligence?* Inf. téc. URL: https://www.cs.ubc.ca/~poole/ci/ch1.pdf.

[6] *Counter-Strike: Global Offensive - Twitch Statistics and Charts · TwitchTracker.* URL: https://bit.ly/3ci5xhm.

[7] *CS:GO News & Coverage | HLTV.org.* URL: https://www.hltv.org/.

[8] *CS:GO Player statistics database | HLTV.org.* URL: https://www.hltv.org/stats/players?startDate=all.

[9] *Data Mining and Knowledge Discovery Handbook.* Springer US, 2010. DOI: 10.1007/978-0-387-09823-4.

[10] *ESL Pro League - Season 14.* URL: https://pro.eslgaming.com/csgo/proleague/.

[11] *GitHub - StatsHelix/demoinfo: A library to analyze CS:GO demos in C#.* URL: https://github.com/StatsHelix/demoinfo.

[12] Andreas Kaplan y Michael Haenlein. "Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence". En: *Business Horizons* 62.1 (ene. de 2019), págs. 15-25. ISSN: 00076813. DOI: 10.1016/j.bushor.2018.08.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S0007681318301393.

[13] *Parallel.For Método (System.Threading.Tasks) | Microsoft Docs*. URL: https://docs.microsoft.com/es-es/dotnet/api/system.threading.tasks.parallel.for?view=net-5.0.

[14] *Python leads the 11 top Data Science, Machine Learning platforms: Trends and Analysis - KDnuggets*. URL: https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html.

[15] *RapidMiner - Wikipedia, la enciclopedia libre*. URL: https://es.wikipedia.org/wiki/RapidMiner.

[16] *Twitch Statistics & Charts · TwitchTracker*. URL: https://twitchtracker.com/statistics.

[17] *Weka 3 - Data Mining with Open Source Machine Learning Software in Java*. URL: https://www.cs.waikato.ac.nz/ml/weka/.

6

Anexo. Glosario

El presente anexo contiene la definición de cada uno de los 41 atributos analizados originalmente durante el desarrollo del trabajo.

1. **Map (Nominal)**: Nombre del escenario donde se desarrolla la partida. Los valores que puede tomar son los siguientes: ”de_dust2”, ”de_inferno”, ”de_mirage”, ”de_nuke”, ”de_overpass”, ”de_train”, ”de_vertigo”.

2. **RoundNumber (Numérico)**: Número de la ronda.

3. **RoundWinner (Numérico)**: Bando ganador de la ronda.

4. **CT-Survivors (Numérico)**: Número de supervivientes del bando *CounterTerrorist* al final de la ronda.

5. **T-Survivors (Numérico)**: Número de supervivientes del bando *Terrorist* al final de la ronda.

6. **CT-EquipValue (Numérico)**: Valor del equipamiento adquirido por parte del bando *CounterTerrorist*.

7. **T-EquipValue (Numérico)**: Valor del equipamiento adquirido por parte del bando *Terrorist*.

8. **CT-Competence (Nominal)**: Accesibilidad a equipamiento (dinero) disponible para el bando *CounterTerrorist* al inicio de la ronda. Los valores que puede tomar son los siguientes: .Economic round”, ”Full-buy round”, ”Semi-buy round”.

9. **T-Competence (Nominal)**: Accesibilidad a equipamiento (dinero) disponible para el bando *Terrorist* al inicio de la ronda. Los valores que puede tomar son los siguientes: .Economic round”, ”Full-buy round”, ”Semi-buy round”.

10. **CT-EquipmentValueSaved (Numérico)**: Valor del equipamiento guardado por parte del bando *CounterTerrorist*.

11. **T-EquipmentValueSaved (Numérico)**: Valor del equipamiento guardado por parte del bando *Terrorist*.

12. **CT-DistanceTravelled (Numérico)**: Distancia total viajada por parte del bando *CounterTerrorist*.

13. **T-DistanceTravelled (Numérico)**: Distancia total viajada por parte del bando *Terrorist*.

14. **BombPlanted (Booleano)**: Activación de la bomba.

15. **BombDefused (Booleano)**: Desactivación de la bomba.

16. **RoundTime (Numérico)**: Tiempo desde que empieza hasta que acaba la ronda.

17. **CT-AverageHP (Numérico)**: Media de salud de los jugadores del bando *CounterTerrorist* al final de la ronda.

18. **T-AverageHP (Numérico)**: Media de salud de los jugadores del bando *Terrorist* al final de la ronda.

19. **CT-AverageArmor (Numérico)**: Media de puntos de blindaje de los jugadores del bando *CounterTerrorist* al final de la ronda.

20. **T-AverageArmor (Numérico)**: Media de puntos de blindaje de los jugadores del bando *Terrorist* al final de la ronda.

21. **CT-PistolKills (Numérico)**: Número de eliminaciones con pistolas por parte del bando *CounterTerrorist*.

22. **T-PistolKills (Numérico)**: Número de eliminaciones con pistolas por parte del bando *Terrorist*.

23. **CT-SMGKills (Numérico)**: Número de eliminaciones con armas ligeras por parte del bando *CounterTerrorist*.

24. **T-SMGKills (Numérico)**: Número de eliminaciones con armas ligeras por parte del bando *Terrorist*.

25. **CT-HeavyKills (Numérico)**: Número de eliminaciones con armas pesadas por parte del bando *CounterTerrorist*.

26. **T-HeavyKills (Numérico)**: Número de eliminaciones con armas pesadas por parte del bando *Terrorist*.

27. **CT-RifleKills (Numérico)**: Número de eliminaciones con rifles por parte del bando *CounterTerrorist*.

28. **T-RifleKills (Numérico)**: Número de eliminaciones con rifles por parte del bando *Terrorist*.

29. **CT-SniperKills (Numérico)**: Número de eliminaciones con francotirador por parte del bando *CounterTerrorist*.

30. **T-SniperKills (Numérico)**: Número de eliminaciones con francotirador por parte del bando *Terrorist*.

31. **CT-GrenadeKills (Numérico)**: Número de eliminaciones con granadas por parte del bando *CounterTerrorist*.

32. **T-GrenadeKills (Numérico)**: Número de eliminaciones con granadas por parte del bando *Terrorist*.

33. **CT-HS % (Numérico)**: Porcentaje de acierto en la cabeza por parte del bando *CounterTerrorist*.

34. **T-HS % (Numérico)**: Porcentaje de acierto en la cabeza por parte del bando *Terrorist*.

35. **CT-FA (Numérico)**: Número de eliminaciones asistidas con una granada cegadora por parte del bando *CounterTerrorist*.

36. **T-FA (Numérico)**: Número de eliminaciones asistidas con una granada cegadora por parte del bando *Terrorist*.

37. **CT-DefuseKits (Numérico)**: Número de dispositivos antiexplosivos.

38. **OpeningKill (Nominal)**: Bando que realiza la primera eliminación en la ronda. Los valores que puede tomar son los siguientes: ”Terrorist”, ÇounterTerrorist”.

39. **OpeningKillTime (Numérico)**: Tiempo que transcurre desde que empieza la ronda hasta que se produce la primera eliminación.

40. **CT-TradeKills (Numérico)**: Número de eliminaciones intercambiadas por parte del bando *CounterTerrorist* (JugadorT A mata a JugadorCT B y JugadorCT C mata a JugadorT A).

41. **T-TradeKills (Numérico)**: Número de eliminaciones intercambiadas por parte del bando *Terrorist* (JugadorT A mata a JugadorCT B y JugadorCT C mata a JugadorT A).

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA