# Equipo A2SJ - Entregable 3: Informe final
# *GamePredRater*

Minería de Datos

Agustín Mora Acosta
Agustin.Mora1@uclm.es
UCLM
Ciudad Real, Spain

Andrés González Díaz
andres.gonzalez8@alu.uclm.es
UCLM
Ciudad Real, Spain

Sergio Sevilla Ballesteros
Sergio.Sevilla@alu.uclm.es
UCLM
Ciudad Real, Spain

José Ángel Villamor Carrillo
JoseAngel.Villamor@alu.uclm.es
UCLM
Ciudad Real, Spain

## 1 INTRODUCCIÓN

Este documento se corresponde con el informe del trabajo propuesto por el equipo de trabajo A2SJ para la asignatura Minería de Datos perteneciente al Grado en Ingeniería Informática de la Universidad de Castilla-La Mancha.

El proyecto ha recibido el nombre de *'GamePredRater'*, con origen en el objetivo de presentar un modelo capaz de predecir la valoración que tendrá un videojuego, partiendo de algunos datos previos a la venta de este como su precio, su género o su descripción. Para alcanzar dicho objetivo se pondrán en práctica las diferentes técnicas y algoritmos que nos ofrece la asignatura, siguiendo las etapas que conforman el Proceso KDD.

A lo largo de este informe se irá viendo las diferentes tareas que ha realizado el equipo en cada fase del *Proceso KDD*. Las fases de dicho proceso son selección de datos, preprocesamiento de datos, transformación de los datos, minería de datos (la cual conlleva con ella diferentes pasos) y por último interpretación de los datos. Estas fases se realizan de forma secuencial e iterativa.

Seguir estos pasos a la hora de desarrollar el proyecto asegurará un trabajo correctamente realizado y organizado.

Además, el equipo de trabajo ha creado un repositorio en GitHub, donde puedes ver más detalladamente todo el estudio llevado a cabo. Además de diversa información que puedes obtener en la wiki.

## 2 COMPRENSIÓN DEL DOMINIO

La tecnología ha experimentando en los últimos años una explosión tremenda en la sociedad, y uno de los sectores en los que más uso se hace de la tecnología, es en la forma en la que nos entrenemos y pasamos nuestro tiempo. Hace años, era impensable tanto el tener acceso a miles de películas y series como el ponerte en la piel de un futbolista de élite.

Esto último es posible gracias a la aparición de los videojuegos, uno de los sectores más destacados a día de hoy en la inmersión de la tecnología en nuestras vidas. Sin embargo, no todos los juegos consiguen alcanzar una gran gratitud en su público, ¿por qué? ¿cuáles son los aspectos que influyen a la hora de que un videojuego tenga éxito sobre sus competidores? Esto es lo que queremos averiguar en nuestro estudio.

### 2.1 Planteamiento de la hipótesis y de los objetivos a perseguir

Cómo ya se ha mencionado el objetivo y solución que se busca dar en este proyecto es un modelo con capacidad de predecir el rating de un videojuego. El fin de dicho modelo se basa en poder realizar funciones de soporte a las plataformas de desarrollo y venta de videojuegos; dando una visión de las futuras ventas que se deben prever dada las características del juego, tales como plataforma o género.

Para llevar a cabo dicho modelo se deben considerar diversos factores como el conocimiento del dominio, los distintos datos que se muestran y su forma de hacerlo.

Además no podemos olvidar que el modelo es susceptible a la metodologías y las decisiones tomadas a lo largo del proceso. Decisiones como el procedimiento a seguir durante el preprocesado y la selección de datos, las técnicas usadas durante la reducción y proyección de los datos las cuales están pendientes de los objetivos propuestos del proyecto, la elección de funciones y algoritmos de *Data Mining* e incluso la interpretación final de los datos, relaciones y patrones obtenidos.

### 2.2 Tecnologías a aplicar

A la hora de realizar este estudio se va a usar el lenguaje de programación *Python*, un lenguaje de programación multiparadigma que destaca por su versatilidad, simplicidad y rapidez de desarrollo.

Además, el equipo que va a realizar este estudio usará *Google Colab*, que permitirá crear cuadernos. Un cuaderno es un documento en el cual se puede combinar código ejecutable de *Python* y texto enriquecido. La gran ventaja de *Google Colab* es que permite compartir contenido fácilmente entre los componentes del grupo, además de conceder acceso gratuito a *GPUs*.

Una de las grandes ventajas de usar *Python* para la Minería de Datos son las numerosas librerías con las que cuenta este lenguaje, destacando las librerías *NumPy* y *Pandas*.

La librería *NumPy* da soporte a la creación de vectores y matrices multidimensionales, además de aportar una serie de funciones matemáticas que permitirán operar con los vectores y las matrices. El uso de la librería *NumPy* aporta a *Python* una funcionalidad comparable a la de *MATLAB*. La librería *Pandas*, permite la lectura y escritura de archivos *CSV*, además de ayudar a mostrar los datos de forma más visual y cómoda.

Se utilizará también la librería *scikit-learn*, que permite realizar tanto entrenamientos de modelos, como selección de características, optimización de modelos, etc.

También se utilizará la herramienta *BigML*, que permite entrenar modelos y evaluarlos de manera sencilla.

### 2.3 Beneficios de la propuesta

El gran beneficio de esta propuesta para el equipo que lo lleva a cabo, es comprender cuales son los principales aspectos que hacen que un juego sea atractivo para la comunidad de jugadores, y el uso que se haga de la minería de datos para este propósito.

A lo largo de este estudio, investigando sobre el sector de los videojuegos, se conocerán diferentes comunidades, como por ejemplo SteamSpy, una comunidad que ofrece información sobre las ventas de los videojuegos de la plataforma *Steam*.

Además, al ser un sector que crece cada día, seguro que el resultado de este estudio podrá tener múltiples aplicaciones en el futuro.

### 2.4 Antecedentes o trabajos similares

Haciendo una búsqueda por Internet, se encuentran sobre todo trabajos o proyectos enfocados con la Minería de Datos y las ventas de los videojuegos, sobre todo en la predicción de cuantos se van a vender [1] [2].

Hay un trabajo muy similar al que desarrollado en el documento [3], la única diferencia es que en el trabajo citado se enfoca en predecir el porcentaje de posibilidad que tiene un videojuego para ser un *hit* o ser muy vendido en el mercado, y el nuestro se enfoca en cual será la valoración que le darán los usuarios al videojuego. Otro trabajo similar al anterior es este [4], el cual analiza cuales son las características que permiten que un videojuego sea un taquillazo.

Aún así, se encontró un trabajo de Minería de Datos [5] con un conjunto de datos que recogen las opiniones de la gente de diferentes juegos en la plataforma de Steam, utilizando diferentes técnicas de aprendizaje automático, como Árboles de Regresión y Clasificación o Redes Neuronales Artificiales. Este trabajo es parecido al de este proyecto, solamente este se enfoca en cómo puede afectar la valoración del juego socialmente, teniendo en cuenta factores como los amigos del usuario de la valoración del conjunto de datos.

## 3 SELECCIÓN DE LOS DATOS

### 3.1 Descripción de la fuente de datos

Los datos que serán utilizados para llevar a cabo este proyecto han sido extraídos de la página web de *Kaggle*, una conocida comunidad de científicos de datos que cuenta con un amplio catálogo de *datasets* y retos. En concreto el dataset utilizado proporcionado por un usario de la comunidad ya mencionada, provienen de la plataforma de distribución de videojuegos *Steam*, y contienen información sobre varios aspectos de los videojuegos ofertados en la tienda de la plataforma. Se pueden dividir los datos que serán utilizados en lo siguiente:

- Un *dataset* principal que contiene las principales características y todo lo necesario para sacar el *target* a predecir.
- Varios *datasets* de apoyo que servirán para ampliar los datos del dataset principal, y así poder extraer más valor de los datos para afrontar el reto propuesto.

### 3.2 *Dataset* principal

El *dataset* denominado '*steam*' cuenta con 18 *features* y 27.033 registros. Cada registro se corresponde a un videojuego de la tienda de la plataforma, y contiene diferentes datos básicos sobre este, a continuación se va a describir cada una de las features incluidas en este *dataset*:
- **appid**: Identificador único entero de cada juego.
- **name**: Titulo del juego.
- **release_date**: Fecha de salida del juego en formato YYYY-MM-DD.
- **english**: Esta variable toma el valor 1 si el juego soporta el inglés como idioma.
- **developer**: Nombre o nombres de las compañías involucradas en el desarrollo del juego. (Si son varios vienen delimitados por puntos y comas)
- **publisher**: Nombre o nombres de las compañías involucradas en la publicación y distribución del juego. (Si son varios vienen delimitados por puntos y comas)
- **platforms**: Lista de plataformas que soporta el videojuego separadas por punto y coma.
- **required_age**: Mínima edad requerida para jugar al título, acorde con el estándar PEGI UK. Aquellos con un valor de 0 en este campo no han sido clasificados.
- **categories**: Lista separada por puntos y coma de las categorías a las que pertenece el juego. Algunos ejemplos de estas categorías son: Single-player, Steam Achivements, Multi-player... etc.
- **genres**: Lista separada por puntos y coma de los géneros a los que pertenece el juego. Algunos ejemplos de estos géneros son: Action, Indie, Casual... etc
- **steamspy_tags**: Lista separada por puntos y coma de las steamspy tags más votadas por la comunidad para este juego. Son similares a los géneros, pero son votados por la comunidad. Algunos ejemplos de estas tags son: Action, Adventure, Indie... etc
- **achievements**: Número de logros que pueden ser adquiridos ingame, en caso de que los tenga.
- **positive_ratings**: Número de valoraciones positivas del juego.

**Table 1: Dataset Principal**

| name | release_date | ... | positive_ratings | negative_ratings | owners | price |
|---|---|---|---|---|---|---|
| Counter-Strike | 2000-11-01 | ... | 124534 | 3339 | 10000000-20000000 | 7.19 |
| Team Fortress Classic | 1999-04-01 | ... | 3318 | 633 | 5000000-10000000 | 3.99 |
| Day of Defeat | 2003-05-01 | ... | 3416 | 398 | 5000000-10000000 | 3.99 |
| Deathmatch Classic | 2001-06-01 | ... | 1273 | 267 | 5000000-10000000 | 3.99 |
| Half-Life: Opposing Force | 1999-11-01 | ... | 5250 | 288 | 5000000-10000000 | 3.99 |

- **negative_ratings**: Número de valoraciones negativas del juego.
- **average_playtime**: Media del tiempo de juego por los usuarios.
- **owners**: Número de usuarios propietarios del juego. Viene definido por un intervalo, como por ejemplo 20000-50000.
- **price**: Precio actual del título en la tienda de la plataforma en libras esterlinas.

### 3.3 *Datasets* de apoyo

**Dataset 1**
El primer dataset de apoyo se denomina '*steam description data*' y cuenta con 4 *features* y 27.033 registros. Cada registro se corresponde a un videojuego al igual que en el *dataset* principal, y contiene diferentes descripciones en lenguaje natural de los títulos. En muchos de los videojuegos, todas las descripciones son iguales. Contiene las siguientes variables:
- **appid**: Identificador único de cada juego. Coincide con los identificadores del *dataset* principal, por lo que es sencillo mergearlos.
- **detailed_description**: Descripción detallada del juego.
- **about_the_game**: Otra descripción del juego, aunque suele coincidir con la descripción detallada.
- **short_description**: Descripción corta del juego.

**Dataset 2**
Este segundo *dataset* de apoyo se denomina '*steamspy tag data*', y contiene 372 columnas y 27.033 registros. Cada columna de este *dataset* se corresponde con un *tag* (Exceptuando la primera que se 3 corresponde con el *appid*), y toma como valor el número de votos de usuarios de la comunidad a esa determinada *tag* para un juego. Este *dataset* se corresponde con la feature que encontramos en el *dataset* principal *steamspy tags*, dando así más información que dicha columna, ya que también aporta el número de votos que recibió cada *tag*.

**Dataset 3**
Este conjunto de datos llamado '*steam requirements data*' contiene información sobre los requisitos necesarios para jugar cada juego del *dataset* principal. Contiene 27319 registros, y un total de 6 *features* o columnas. Los valores de la mayoría de las columnas son texto plano, con una estructura similar a la de un *JSON* y a lenguajes *HTML5*.
- **appid**: Identificador único de cada juego. Coincide con los identificadores del *dataset* principal al igual que en los otros datasets de apoyo.
- **pc_requirements**: Requisitos del juego en ordenador.
- **mac_requirements**: Requisitos del juego en ordenador, en concreto para MAC.
- **linux_requirements**: Requisitos del juego en ordenador, en concreto para Linux.
- **minimum**: Requisitos mínimos para poder jugar, extraído de la columna **pc_requirements**.
- **recommended**: Requisitos recomendados para poder jugar, extraído de la columna .

**Dataset 4** El último dataset de apoyo se denomina *'hours played'* y cuenta con 5 features y 200.000 registros. Las columnas de este dataset hacen referencia a las siguientes características:
- **usuario**: Hace referencia al código de un usuario de Steam.
- **nombre**: Hace refencia al nombre de un videojuego.
- **tipo**: Existen dos tipos, *'purchase'*, que señala que el videojuego ha sido comprado, y *'play'*, que indica que el dato de horas es válido, ya que el tipo *'purchase'* siempre asigna el valor 1.0 a la feature horas.
- **horas**: Número de horas que el usuario ha jugado al videojuego.
- **0**: Esta columna no tiene valor alguno, todos sus valores son 0.

Para poner en contexto, un registro de este dataset hace referencia al número de horas que ha jugado un usuario a un determinado videojuego, por tanto esta información podrá ser de mucha utilidad en este estudio.

## 4 PREPROCESAMIENTO Y TRANSFORMACIÓN DE DATOS

El objetivo de este apartado es preprocesar los datasets presentados previamente con la finalidad de obtener una tarjeta de datos final y definitiva que nos permita trabajar con las diferentes técnicas y modelos necesarios para extraer conocimiento de los datos.

### 4.1 Dataset principal

Nuestro dataset principal contiene un gran número de features importantes para nuestra tarea de predicción de ratings, pero estas features necesitan ser preprocesadas y transformadas. Además de esto, necesitamos extraer lo que será nuestra variable objetivo a partir de este dataset, es decir, lo que queremos predecir.

**TRATAMIENTO DE VALORES VACÍOS**
El dataset principal no contaba con valores vacíos en ninguna de las columnas, por lo que no ha sido necesario realizar ningún tipo de tratamiento.

**Table 2: Dataset de apoyo: Descripciones de los juegos**

| steam_appid | detailed_description | about_the_game | short_description |
|---|---|---|---|
| 10 | Play the world's number 1 online ... | Play the world's number 1 online ... | Play the world's number 1 online ... |
| 20 | One of the most popular online act... | One of the most popular online act... | One of the most popular online act... |
| 30 | Enlist in an intense brand of Axi... | Enlist in an intense brand of Axi... | Enlist in an intense brand of Axi... |
| 40 | Enjoy fast-paced multiplayer gamin... | Enjoy fast-paced multiplayer gamin... | Enjoy fast-paced multiplayer gamin... |
| 50 | Return to the Black Mesa Research ... | Return to the Black Mesa Research ... | Return to the Black Mesa Research ... |

**Table 3: Dataset de apoyo: Tags de Steamspy**

| appid | 1980s | 1990s | 2.5d | 2d | 2d_fighter | 360_video | 3d | 3d_platformer | 3d_vision | 4_player_local | 4x | 6dof | atv | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 144 | 564 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| 20 | 0 | 71 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| 40 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| 50 | 0 | 77 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |

**Table 4: Dataset de apoyo: Requisitos del juego**

| steam_appid | pc_requirements | ... | minimum |
|---|---|---|---|
| 10 | {'minimum': '\r\n\t\t\t<p>Minimum:</st... | ... | 500 mhz processor, 96mb ram, 16mb video card, ... |
| 20 | {'minimum': '\r\n\t\t\t<p>Minimum:</st... | ... | 500 mhz processor, 96mb ram, 16mb video card, ... |
| 30 | {'minimum': '\r\n\t\t\t<p>Minimum:</st... | ... | 500 mhz processor, 96mb ram, 16mb video card, ... |
| 40 | {'minimum': '\r\n\t\t\t<p>Minimum:</st... | ... | 500 mhz processor, 96mb ram, 16mb video card, ... |
| 50 | {'minimum': '\r\n\t\t\t<p>Minimum:</st... | ... | 500 mhz processor, 96mb ram, 16mb video card, ... |

**Table 5: Dataset de apoyo: Horas jugadas a juegos**

| ! usuario | nombre | tipo | horas | 0 |
|---|---|---|---|---|
| 151603712 | The Elder Scrolls V Skyrim | purchase | 1.0 | 0 |
| 151603712 | The Elder Scrolls V Skyrim | play | 273.0 | 0 |
| 151603712 | Fallout 4 | purchase | 1.0 | 0 |
| 151603712 | Fallout 4 | play | 87.0 | 0 |
| 151603712 | Spore | purchase | 1.0 | 0 |

**VARIABLE OBJETIVO**

Para el cálculo de la variable objetivo se ha tomado la división de el número de valoraciones positivas entre el número total de valoraciones del juego en cuestión. Se ha escogido este valor con el fin de normalizar y tratar de forma equitativa aquellos juegos que tienen un gran número de valoraciones y aquellos que tienen un número reducido de estas.

Tras calcular dicha variable, sacamos algunas gráficas para estudiar la distribución de esta y así poder conocer un poco mejor los datos. El histograma de esta variable objetivo, que podemos ver en la parte superior de la figura 1, nos indica que ocurre algo extraño, puesto que existen un gran número de juegos cuyo valor de la variable objetivo es 1 (Todas las valoraciones son positivas), 0 (Todas las valoraciones son negativas) o 0.5 (Exactamente la mitad de las valoraciones son positivas), datos que no parecen reflejar la realidad de lo que se pretende predecir. Tras investigar el por qué de esta distribución, nos dimos cuenta de que esto se debía a que había bastantes juegos que contaban con muy pocas valoraciones, haciendo que los valores comentados anteriormente fueran más habituales de lo que deberían. Para solucionar esto, filtramos el dataset por aquellos juegos que contasen al menos con 20 valoraciones, obteniendo una distribución mucho más balanceada que podemos ver en la parte inferior de la figura anteriormente mencionada. Tras este filtrado nos quedamos únicamente con 16807 registros de los 27075 con los que contaba el dataset original.

**PREPROCESADO/TRANSFORMACIÓN DEL RESTO DE VARIABLES**

Una vez teníamos calculada la variable objetivo, preprocesamos y transformamos el resto de features del dataset:

- **Columna 'english':** Esta columna toma el valor 1 en el 98% de los registros, y además no parece tener ningún tipo de correlación con la variable objetivo, por lo que se ha dedicidido desecharla.
- **Columnas 'developer' y 'publisher':** Estas columnas toman un gran número diferente de valores categóricos, impidiendonos transformar estas columnas en una columna por valor. Hemos considerado que la mejor opción para transformar

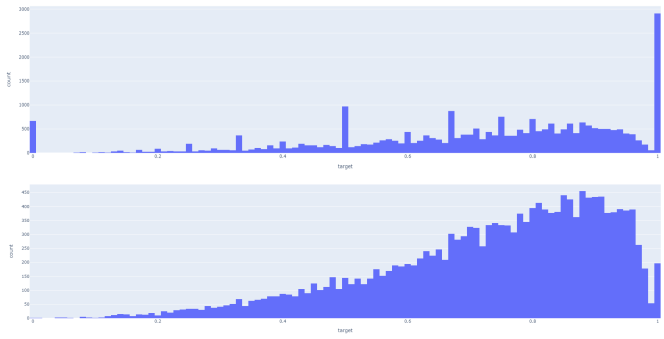

**Figure 1: Distribución de la variable objetivo antes y después del filtrado por número de valoraciones**

estas columnas es convertirlas en una variable numérica entera que represente el número de juegos desarrollados por el estudio que aparezca en la columna *'developer'*, y que suele coincidir en la mayoría de los casos con el que aparece en la columna *'publisher'*.

- **Columna 'platforms':** Esta columna contiene información acerca de aquellas plataformas para las que está disponible el juego separadas por el carácter ';' en caso de que esté disponible en varias plataformas. La transformación que hemos aplicado ha sido crear una columna por cada una de las 3 plataformas que podían aparecer en esta columna, y marcarlas a 1 o 0 dependiendo de sí aparecían en la columna o no.
- **Columna 'required_age':** A esta columna no le hemos aplicado ninguna transformación puesto que esta columna contiene valores numéricos enteros.
- **Columnas 'categories' y 'genres':** Estas dos columnas son similares a *'platforms'*, puesto que contienen las categorías y los géneros a los que pertenece un juego respectivamente, separados por el carácter ';' en caso de que pertenezca a varios. Al igual que con *'platforms'*, para cada uno de las categorías/géneros que pueden aparecer en estas variables se ha creado una nueva columna que puede tomar el valor 0 o 1 dependiendo si dicho juego pertenece a esa categoría/género.

  La diferencia con la columna *'platforms'* está en que con esta columna solo añadíamos 3 nuevas columnas, pero sin embargo al hacer este proceso con las columnas *'categories'* y *'genres'* añadimos más de 50 nuevas columnas. Además, algunas de estas columnas solo toman el valor 1 en un número mínimo de registros (En algunos casos en menos del 1% de registros). Para solucionar este problema de dimensionalidad, lo que hicimos fue mirar el porcentaje de ocurrencias del valor '1' por columna, desechando aquellas columnas en las que dicho porcentaje fuera menor que el 2% o mayor que el 60% en el caso de que fueran columnas de categorías, y desechando aquellas columnas en las que dicho porcentaje fuera menor que el 1% para las columnas que se correspondían con géneros. Con este filtrado de columnass, reducimos el número de columnas generadas a partir de la columna *'categories'* de 29 a 20, y el número de columnas generadas a partir de la columna *'genres'* de 27 a 16.
- **Columna 'steamspy_tags':** Hemos decidido desechar esta columna debido a que disponemos de la misma información que nos aporta esta columna pero de una forma más detallada en el dataset de apoyo de Tags de Steamspy.
- **Columna 'achievements':** Esta columna contiene el número de logors que tiene el juego en Steam, información que puede ser de importancia para nuestro objetivo debido a que existe un pérfil de jugador que se centra en conseguir dichos logros. Esta columna no hace falta que la preprocesemos puesto que viene en formato numérico entero.
- **Columna 'owners':** Esta columna contiene rangos de valores enteros que expresan el número de jugadores que han adquirido el juego de forma aproximada. Para transformar esta columna hemos decidido que convertir cada uno de estos rangos en una columna nueva no sería lo correcto, puesto que de esta forma perderíamos la ordinalidad de esta variable. Para conservar dicha ordinalidad, hemos convertido cada rango que aparece en esta variable en el valor medio del rango, haciendo que esta columna acabe siendo numérica entera. Por ejemplo, el valor '0-10' ha sido transformado en el valor numérico entero 5.
- **Columna 'price':** ' Esta columna consideramos que es muy importante, ya que el precio es una característica de mucho peso a la hora de valorar un juego. En nuestro dataset es de tipo float, por lo que no ha sido necesario hacerle ningún tipo de preproceso o transformación.

## 4.2 Dataset de apoyo: horas jugadas

A la hora de recibir un videojuego una review, un aspecto que puede ser muy significativo es estudiar el tiempo que han dedicado los usuarios a jugarlo. El dataset principal cuenta con las columnas media y mediana, de gran utilidad, pero con un gran problema, la cantidad de registros cuyos valores para ambas características es 0. En total, de los 27075 videojuegos con los que se trabaja en el dataset principal, 20905 no cuentan con un valor para la media y la mediana.

Gracias al dataset denominado *'hours_played.csv'*, se podría minimizar la cantidad de registros con valores a 0.

**LIMPIEZA DEL DATASET DE APOYO**
Antes de realizar cualquier cálculo con este dataset, el primer paso es depurar su información, eliminando aquellas características y registros que no sean de utilidad.

Las únicas tres columnas que son de utilidad son el nombre del videojuego, horas y tipo, esta última servirá para filtrar los registros de tipo *'play'*, ya que los *'purchase'* hacen referencia a la compra del videojuego, y el valor de horas es 1.0, y no es válido. Tras el filtrado, se podrá eliminar la columna tipo.

**CÁLCULOS CON EL DATASET DE APOYO**
Una vez el dataset solo cuenta con las columnas nombre y horas, se procede a calcular la media y la mediana, haciendo uso de la función *DataFrame.groupby* de la librería Pandas, que permitirá agrupar los

diferentes registros que coincidan en nombre por media, mediana y desviación típica, esta última, será una nueva columna, que no estaba en el dataset principal.

Tras realizar los agrupamientos con *DataFrame.groupby*, se puede ver que el total de juegos que maneja nuestro dataset de apoyo son 3600.

**COMPARACIÓN DE AMBOS DATASETS**
El primer paso es comprobar cuál es la cantidad de videojuegos de los que se podrá obtener la media y la mediana gracias a este dataset de apoyo. El gran problema es que no se cuenta con algún identificador que relacione el dataset principal con el de apoyo. Por tanto, se procede a relacionarlos mediante el nombre del videojuego, aunque esto podría suponer que algún dato aprovechable se pierda.

De los 3600 juegos del dataset de apoyo, 1724 juegos coinciden en nombre con los del dataset principal, y de estos, 416 tienen valor 0 en la media y mediana. Para guardar los datos de forma organizada, se crea un nuevo dataframe, que contará con las columnas:

- **appid**: Permitirá relacionar el dataframe con el dataset principal.
- **average_playtime**: Columna que guardará la media de las horas jugadas.
- **median_playtime**: Columna que guardará la mediana de las horas jugadas.
- **standard_deviation**: Almacenará la media de la desviación típica de las horas jugadas de cada juego.

**Columna media y mediana**
Para almacenar estas dos características de los registros, habrá que comprobar si el videojuego en cuestión cuenta con un dato distinto de 0 en el dataset principal, en caso afirmativo, permanecerá dicho valor, en caso negativo, se buscará dicho videojuego en el dataset de apoyo, si está, almacenaremos dicho valor, si tampoco se puede encontrar, se mantendrá el valor a 0.
**Columna desviación típica**
Al ser una columna que no aparecía en el dataset principal, se aprovechará el cálculo de la desviación típica de los 1724 registros del dataset de apoyo que coinciden con el dataset principal. Para el resto de registros, el valor asignado será 0.

**EXPORTACIÓN DE DATOS**
Los datos se guardarán en un nuevo dataframe en el cual, a diferencia del dataset de apoyo en su estado original, si se guarda un id para relacionar este nuevo dataset con el principal de forma sencilla.

## 4.3 Dataset de apoyo: requisitos del juego

A la hora de compra y elección de un juego, una característica que suele pasar desapercibida entre los consumidores de juegos de consolas son las características de mínimas y recomendadas para disfrutar de estos medios. Sin embargo, entre los jugadores de pc la plataforma y los requisitos técnicos mínimos y recomendados de cada juego suelen ser una característica esencial en la que muchos consumidores del medio presta atención para saber si realmente podrán disfrutar y sacarle el máximo partido a los juegos. El dataset principal cuenta con una columna que recoge las plataformas en las que el videojuego es soportado, algo esencial para un usuario de cara a conocer de antemano si podrá disfrutar de dicho juego con su equipo.

En total, de los 27075 videojuegos con los que se trabaja en el dataset principal, 13185 no cuentan con unos requisitos recomendados. Además, entre los registros existentes existen varios formatos de colección de las características hardware, mezclando idiomas y lenguajes.

**LIMPIEZA DEL DATASET DE APOYO**
Antes de realizar cualquier cálculo con este dataset, el primer paso es depurar su información, eliminando aquellas características y registros que no sean de utilidad. La columna de recommended cuenta únicamente con 13890 las cuales son copia de la columna pc_requirements, por lo que no aporta información, de está forma se decide eliminarla y prestar atención a la columna de requisitos mínimos y la de requisitos mínimos y recomendados para usuarios de PC como fuente principal de extracción de información. Debido a la dificultad de análisis previamente mencionada nos centraremos únicamente en las características que suele ser las más decisivas y que los usuarios más atención prestan; siendo algunas de estas la RAM y el Procesador. Una vez el dataset solo cuenta con las columnas necesarias y que hemos seleccionado las características a las que prestar atención generamos un traductor y una serie de métodos para la limpieza de los registros. Durante el preprocesado de los registros diferenciamos tres fases:
- **Limpieza y Traducción**: El registro es traducido al inglés y se eliminan composiciones sintácticas, tales como etiquetas HTML.
- **Análisis de la RAM**: Se analiza buscando expresiones regulares comunes que puedan aportar información de la ram, siendo algunas de las expresiones y palabras reservadas “RAM”, “MB” o “GB”.
- **mac_ requirements** : Se analiza buscando expresiones regulares comunes que puedan aportar información del procesador en especial a la velocidad de este, siendo algunas de las expresiones y palabras reservadas “HZ” o “GHZ”.

Está etapa se repite 2 veces, siendo la 2ª pasada dirigida a los valores nulos. Tras está etapa y con la información extraída se tratan los valores nulos rellenándose mediante la media. Además se calcula la media, mediana y la desviación típica de estos valores.

**Columna RAM y RAM_MBytes**
Nos aportan un medio de almacenar la RAM requerida, debido a que tienen distintas unidades (siendo estas megabytes y gigabytes) se transforman todas a la misma unidad y se guarda en RAM_MBytes.

**Columna Processor y Processor MHZ**
Para almacenar estas dos características de los registros, habrá que comprobar si el videojuego en cuestión cuenta con la velocidad del procesador o el nombre de este, si cuenta se expresan en la misma unidad y almacena en Processor MHZ.

**EXPORTACIÓN DE DATOS**
Se procede al almacenamiento de todos los datos en un nuevo

dataset, donde también se guardará el id que permita establecer la relación entre este dataset con el dataset principal de forma sencilla.

## 4.4 Dataset de apoyo: descripciones del juego

Una de las cosas que hacen que un producto se venda o no es su descripción, ya que sugiere como va a ser este producto. Para los videojuegos no es distinto, y por tanto, es interesante extraer información de las descripciones encontradas en el dataset steam_description_data. Este dataset, como se había indicado previamente, cuenta con 4 columnas, una que corresponde al id del juego, y otras tres que describen el correspondiente juego. Lo que se quiere obtener a través de este conjunto de datos es información correspondiente de cada juego, como podrían ser las palabras fundamentales, el número de palabras por descripción, el número de oraciones, un análisis de sentimientos de las descripciones, etc.

**SELECCIÓN DE DATOS**

Por tanto, lo primero que hay que hacer es saber con qué información hay que quedarse. Por ello, hay que estudiar las tres columnas con texto, y concluir si hay que descartar información, o bien unir toda la información de cada juego en un única columna, sumando todos los valores de cada columna. Por eso primero se procedió a eliminar las etiquetas html que aparecían en las descripciones, ya que podrían incomodar a la hora de realizar las comprobaciones posteriores.

Cuando ya se han dejado los textos con etiquetas, se procede a comparar cada una de las columnas. La columna detailed_description y about_the_game contienen prácticamente los mismo valores, por tanto la segunda columna es descartada. Por tanto, ya solo quedaría saber qué hacer con la columna short_description. Comparando ambas descripciones, la detallada y la corta, se comprueba que ambas columnas no son iguales, de momento ninguna de las dos pueden ser descartadas. Escogiendo descripciones al azar, se detecta que la descripción corta aparece dentro de la detallada, así que se procede a comprobar si todas las descripciones cortas se encuentran dentro de las detalladas. Realizando esto se ha obteniendo un resultado mejor, pero no el que se esperaba; tan solo un 30% de las descripciones cortas están contenidas dentro de la descripción detallada. Pero aún así, al fijarse en otras descripciones al azar que no eran iguales, se veía que ambas tenían prácticamente la misma información, solo que la detallada era más extensa que la corta. Por ello, y haciendo una prueba final, se detecta que más de la mitad de las descripciones cortas tienen al menos una subcadena en la descripción detallada que coincide con la descripción corta. Sabiendo esto y viendo casos sueltos, se decide que los únicos datos con los que se van a trabajar son los de la columna detailed_description.

**PREPROCESAMIENTO DE LOS DATOS**

Una vez escogida los datos con los que se van a trabajar, se han de comprobar antes de realizar el procesamiento del lenguaje natural, comprobando que no haya ninguna descripción vacía y sobre todo que la información recogida sea en el idioma que se está trabajando, en este caso el inglés. Una vez ya se tiene los datos preparados para trabajar, ya se procede al procesamiento

Aún así, hay que realizar una limpieza de ruido de todas las descripciones, eliminando primeramente información innecesaria, como son las comillas, las comas, los puntos... además de algunas palabras como podrían ser las que corresponden a los correos, a las páginas webs, etc. Una vez se haya eliminado el máximo ruido posible, se ha de tokenizar todas las palabras y eliminar las *stops words* (the, and, at...) es decir, las palabras que apenas aporten valor a la oración, como pueden ser los adverbios, artículos, etc. Tras esto, se lematizan los tokens, para obtener la raíz de las palabras, y así poder manejar la información más fácilmente.

**PROCESAMIENTO DEL TEXTO**

Una vez ya se tienen las descripciones limpias, se procede a obtener los valores tfidf. Esto es ver la importancia o el porcentaje de veces que aparecen cada una de las palabras dentro de todos los tokens que se están estudiando. Aunque antes de realizar esto, primeramente se va a realizar un conteo de las palabras: un total de 150.000 palabras distintas, un número muy poco manejable. Por tanto, se obtiene el valor tfidf de las palabras que tengan un valor entre 0.01 y 0.25, para que no sean ni muy poco comunes ni demasiado comunes. Con esto, obtenemos una matriz con todos los valores correspondientes para cada descripción, en la cual las columnas corresponden a las palabras. Se cuenta por tanto con un total de 1778 palabras, un registro mucho más manejable.

Una vez se hayan obtenido los valores de *TFIDF*, ya se buscan otras características extras que se pueden extraer de los textos para poder aportar más información, como sería el Part of Speech (la clasificación sintáctica de cada palabra), el número de palabras y oraciones por descripción y un análisis de sentimientos. Este último indica los valores de sentimientos que tiene la descripción, aportando una cifra del 0 al 1 para positivo, negativo y neutral. Este último apartado es el más descartable de todos, ya que prácticamente todas las descripciones son neutrales o un poco positivas, pero esta característica realmente no aporta nada diferencial.

**EXPORTACIÓN DE DATOS**

Todas las columnas conseguidas en este dataset con las que se vayan a trabajar en el dataset principal serán guardadas en un nuevo dataframe, para posteriormente hacer el correspondiente merge con el dataset principal.

## 4.5 Obtención de la tarjeta de datos

Una vez contabamos con el preproceso del dataset principal y de todos los correspondientes datasets de apoyo, fusionamos todos los datasets para obtener una tarjeta de datos definitiva, la cual cuenta con 16807 registros y un total de 2220 columnas. Esta tarjeta de datos ha sido normalizada, obteniendo así un dataset en el que todas las columnas tienen valores entre 0 y 1.

## 4.6 Selección de caracteristicas

Antes de lidiar con los algoritmos, tenemos que solucionar un gran inconveniente de nuestra tarjeta de datos, la gran dimensionalidad de esta. Para esto, hemos llevado a cabo una selección de características con la finalidad de reducir en gran medida las características de nuestra tarjeta de datos. Para ello, a la hora de llevar a cabo esta

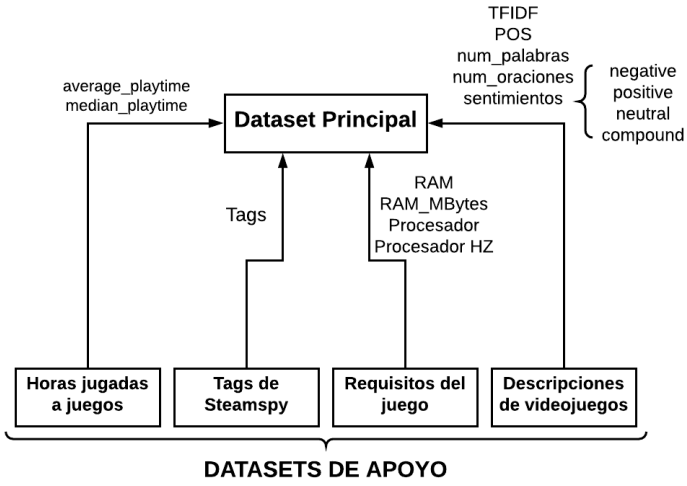


**Figure 2: Diagrama de relaciones entre datasets**

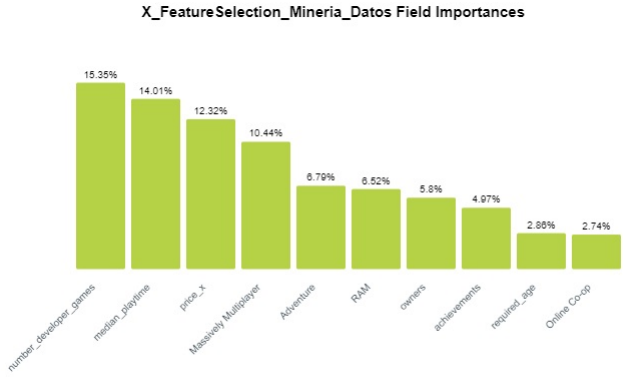


**Figure 3: Variables con mayor importancia para el árbol de decisión construido en BigML**

selección de características, se dividirán en dos categorías todas las características del conjunto de datos. Por un lado las descripciones de los videojuegos y las etiquetas de la comunidad de SteamSpy, es decir, las caracerísticas para las que se tuvo que usar *Natural Languague Processing*, y por el otro lado, el resto de características, entre las que podemos encontrar los géneros o otra información directa del juego como por ejemplo, el precio.

**DESCRIPCIONES Y STEAMSPY TAGS**

En este subconjunto de características se cuenta con muchas más características que en el otro subconjunto, con un total de 1801. Lo primero a tener en cuenta a la hora de llevar a cabo la selección de estas características, es que pueden existir palabras repetidas, por lo que habrá que comprobarlo, y en caso de que existan, convertir esas dos características en una sola, haciendo la media de ambas para obtener el valor final. Una vez no tengamos características repetidas, se llevará a cabo la selección de características mediante el uso del método *SelectKBest*, que se quedará con las 200 mejores características de nuestro conjunto de datos.

**RESTO DE CARACTERÍSTICAS**

Sobre este conjunto de características, en el que podemos encontrar características como el precio, el género o el número de juegos desarrollado por la desarrolladora, se ha llevado a cabo una selección más manual.

Para este proceso, lo primero que realizamos fue crear un árbol de decisión sencillo en *BigML* utilizando únicamente nuestro subconjunto de características, el cual nos permitió extraer la importancia de las características más relevantes a la hora de construir dicho árbol. La importancia de las características más relevantes se puede ver en la Figura 3 y será considerada más adelante para tomar decisiones.

Con el fin de obtener un poco más de información que nos permita realizar esta selección, calculamos y mostramos la matriz de correlación entre las características, la cual podemos ver en la Figura 4. En dicha matriz, se puede ver con claridad que hay algunas características bastante correlacionadas entre sí, como por ejemplo *'Massively Multiplayer'* y *'MMO'*, o *'Violent'* y *'Gore'*. Al igual que con la importancia de las características, utilizaremos esta información más adelante para realizar la selección.

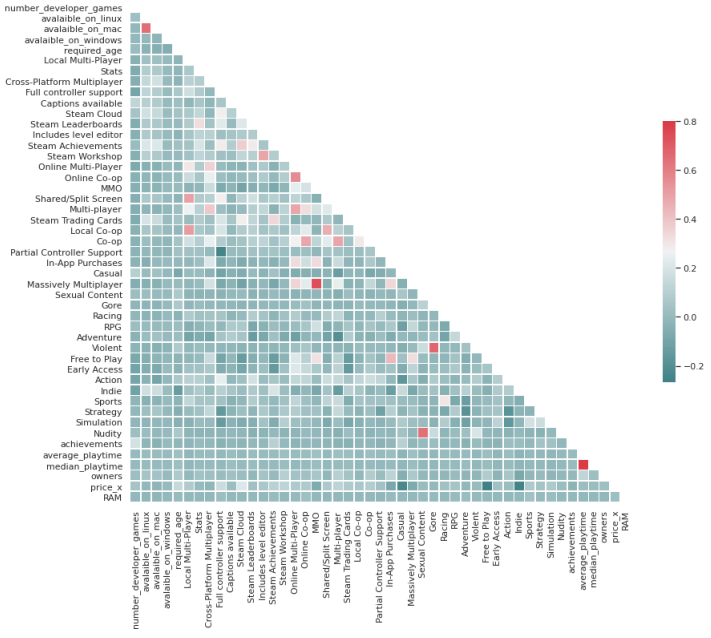


**Figure 4: Matriz de correlación entre las características del segundo subconjunto**

Por último, calculamos la correlación de cada una de las características con la variable objetivo. Esta correlación no era significativa para ninguna de las características, por lo que no fue utilizada para la selección.

Finalmente, para realizar la selección de características consideramos que lo mejor era únicamente seleccionar aquellas características que tomó en consideración el árbol de decisión construido en *BigML*. Para esto, utilizamos un diccionario con el nombre de dichas características y su importancia, el cual fue exportado directamente desde *BigML*. Entre estas características, encontramos algunas que tienen un alto grado de correlación entre ellas, pudiendo aportar cierta información redundante. Para solucionar esto, desechamos aquella característica de ambas que menor grado de importancia tenga. Por ejemplo, *'average_playtime'* y *'median_playtime'* aparecen entre las características con más importancia, pero entre ellas tienen un grado de correlación bastante alto, por lo que desecharemos *'average_playtime'* ya que tiene menor grado de importancia que *'median_playtime'*.

Tras esto, reducimos el número de este segundo subconjunto de características de 47 a 24.

### 4.7 División de la tarjeta de datos - Conjunto de entrenamiento y de test

Tras realizar la selección de características sobre la tarjeta de datos, dividimos nuestra trajeta de datos principalmente en dos conjuntos de datos, 'X' que contendrá unicamente las features, e 'y', el cual contendrá nuestra variable objetivo. Además de esta división, realizaremos a su vez una segunda división para diferenciar el conjuntos de datos de entrenamiento del de test, utilizando el 33% de los registros del conjunto de datos para test.

## 5 MINERÍA DE DATOS

### 5.1 Selección de la apropiada tarea de Minería de Datos

El problema originalmente, debido al tipo del target, que muestra valores continuos, es de regresión, sin embargo, se ha adaptado también a clasificación, para poder realizar modelos tanto en regresión y clasificación con el objetivo de encontrar el mejor modelo posible. Para ello, como se ha explicado anteriormente, el target tiene valores entre 0 y 1, por tanto, para poder crear clases debemos tener en cuenta dichos valores. Para hacer dicha clasificación para este conjunto de datos, se ha basado la clasificación que realiza la plataforma Steam con sus propias valoraciones de los juegos, que es la siguiente:

- 95 - 99% : Overhwelmingly Positive.
- 94 - 80% : Very Positive.
- 80 - 99% + few reviews: Positive.
- 70 - 79% : Mostly Positive.
- 40 - 69% : Mixed.
- 20 - 39% : Mostly Negative.
- 0 - 39% + rew reviews: Negative.
- 0 - 19% : Very Negative.

Al no tener en cuenta las reviews de los juegos, dichas clases tienen que ser adaptadas. Además, se ha tenido que adaptar esta clasificación respecto a los datos con los que se están trabajando, convirtiendo la clasificación original a la siguiente:

- 90 - 99% : Overhwelmingly Positive o clase 3.
- 80 - 89%: Very Positive o clase 2.
- 60 - 79%: Positive o clase 1.
- 0 - 59%: Negative o clase 0.

### 5.2 Elección del algoritmo de Minería de Datos

Para este apartado, se ha decidido abordarlo desde dos puntos de vistas diferentes: una usando algoritmos que sean traceables, es decir, que dada una entrada específica, se pueda saber su salida siguiendo el algoritmo; y utilizando algoritmos no traceables, más orientados a ser una "caja negra". Se ve oportuno hacer estos dos tipos de puntos de vistas para poder valorar ambas salidas de los diferentes modelos.

*5.2.1 Algoritmos traceables.* Se utilizará un árbol de decisión que sea lo suficientemente interpetable a la vez que tenga unos resultados sensatos, dando como salida la clasificación que se explicó en el apartado anterior.

*5.2.2 Algoritmos no traceables.* Se utilizará algoritmos como Arboles de decisión (menos interpretables que los anteriores), Random Forests y se intentarán optimizar para conseguir los mejores resultados, con las herramientas que ofrecen tanto BigML como la librería *scikit-learn*.

### 5.3 Aplicación del algoritmo

*5.3.1 Algoritmos traceables.* Como algoritmo traceable escogimos los árboles de decisiones implementados en la librería Scikit-Learn. Para optimizar los parametros de este modelo, probamos varias combinaciones de parámetros diferentes, evaluando cada modelo mediante validación cruzada, y quedándonos con el mejor de estos modelos. Los parámetros que se tuvieron en cuenta para este paso son los siguientes:

(1) **criterio**: La función para medir la calidad de una división. Los criterios admitidos son "gini" para la impureza de Gini y la "entropía" para la ganancia de información.
(2) **max_depth**: La profundidad máxima del arbol de decisiones.
(3) **min_samples_split**: El número mínimo de muestras necesarias para dividir un nodo interno
(4) **min_samples_leaf**: El número mínimo de muestras necesarias para estar en un nodo hoja (suavizado el efecto en regresión)
(5) **class_weight**: Pesos asociados a las clases. El modo "balanced" utiliza los valores de y para ajustar automáticamente los pesos inversamente proporcionales a las frecuencias de clase en los datos de entrada

*5.3.2 Algoritmos no traceables.* Primeramente, se realizó un algoritmo sencillo para cada uno de los diferentes target en *BigML*, utilizando el modelo Model que aporta dicha herramienta, que se basa en un Árbol de Decisión. Al realizar esto con ambos targets, se encontró que los valores dados en el de regresión, tanto de salida

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **0** | 0.45 | 0.14 | 0.21 | 1225 |
| **1** | 0.37 | 0.74 | 0.49 | 1890 |
| **2** | 0.32 | 0.12 | 0.18 | 1336 |
| **accuracy** | | | 0.36 | 5547 |
| **macro avg** | 0.37 | 0.32 | 0.29 | 5547 |
| **weighted avg** | 0.37 | 0.36 | 0.31 | 5547 |

**Table 6: Resultados del Arbol de decisiones**

como de evaluación, no eran para nada coherentes a lo que realmente se esperaba. Por tanto, se decidió hacer pruebas con el target de regresión utilizando los modelos de la librería *scikit-learn* junto a *Google Colab*, para determinar si había algún tipo de error en los datos. Aún así, los valores obtenidos con el target de clasificación si tenía sentido, por tanto se probó a realizar el equivalente a un Random Forest en *BigML*, un ensemble utilizando Arboles de decisiones y 10 iteraciones. Tras obtener dichos modelos, se decidió que sería interesante utilizar la herramienta *OptiML* con el conjunto de datos, para obtener diferentes modelos con distintas configuraciones y poder escoger el que más se adecúe al objetivo de esta práctica.

Para los modelos de regresión se siguió la misma filosofía que los modelos de clasificación: un modelo sencillo como es el Árbol de Decisión, uno más complejo como es un Random Forest y un modelo utilizando Boosting, que se basa en un Random Forest pero haciendo una selección de samples diferente a este. Uno de los tres modelos resultantes que se obtendran, será optimizado, para ello, se utilizará la función RandomizedSearch, con diferentes parámetros para obtener diferentes modelos.

## 5.4 Evaluación de los resultados

Tras seleccionar, entrenar y validar los modelos, vamos a revisar y comentar los resultados obtenidos:

*5.4.1 Algoritmos traceables.* Utilizando el informe de clasificación proporcionado por Scikit Learn, mostramos el rendimiento de nuestro modelo en la predicción en la tabla 6. Como resultado a todo este procedimiento mostramos el arbol resultante siendo la Figura 5.

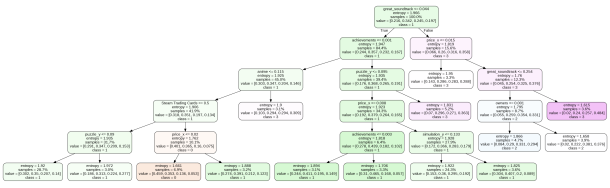

**Figure 5: Arbol de Decisión Completo**

El árbol lo podemos partir en 2 sub arboles partiendo del nodo raíz en función del *"soundtrack"* del videojuego, que como ya hemos visto toma gran importancia en los clientes a la hora de comprar un juego y de calificarlo. De está forma dividimos el árbol de decisión en dos sub-arboles, aquellos con una calificación de soundtrack > 0.044 y los que tienen una inferior.

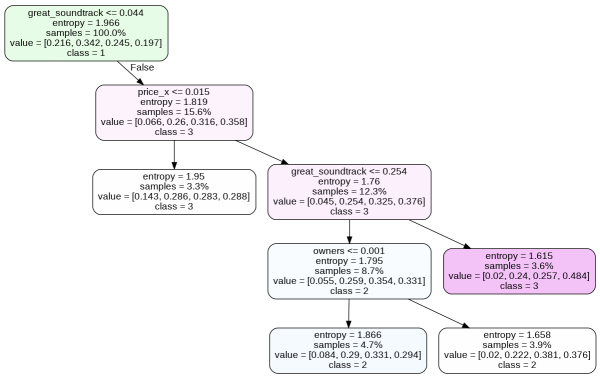


**Figure 6: Sub-Arbol Derecho**

En el primer sub-arbol (Figura 6) podemos observar que la clasificación del videojuego se mueve entre las clases 2 y 3 ("Very Positive" y " Overhwelmingly Positive") las cuales corresponden con las clases más positivas. quedando la decisión entre estas clases en función del precio y el soundtrack. Siendo de la clase 2 cuando la valoracion del sountrack sea menor o igual que 0.254 y el numero de propietarios menor o igual que 0.001.

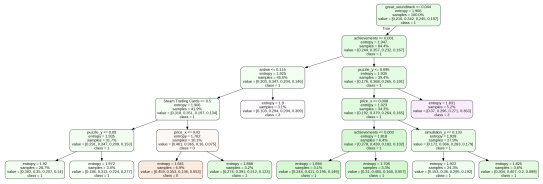

**Figure 7: Sub-Arbol Izquierdo**

El segundo sub-arbol (Figura 7) podemos observar que la clasificación es más compleja encontrando que los videojuegos mueve entre las clases 0, 1 y 3 ("Negative", "Positive" y "Overhwelmingly Positive"), para facilitar el entendimiento de este sub-arbol lo volvemos dividir segun su nodo raiz, el cual los separa en función del numero de logros que se pueden adquirir dentro del videojuego otra caracteristica que tiene peso a la hora de calificar y adquirir un videojuego. (Figura 8)

De esta forma que el juego tenga logros lo marca de forma positiva eliminando de este subconjuto la posibilidad de ser clasificado en la clase 0. Siendo así de clase 3 si el juego tiene relacion con puzzles, sino a las clase 1.

En el ultimo sub-arbol (Figura 9) nos encontramos con las clases 0, 1 y 3; donde si a pesar de no tener logros una parte de la comunidad se siente atraida por la cultura japonesa, los mangas y el anime de forma que si el juego esta ciertamente relacionado puede clasificarse como "Overhwelmingly Positive", sino las tendecias se

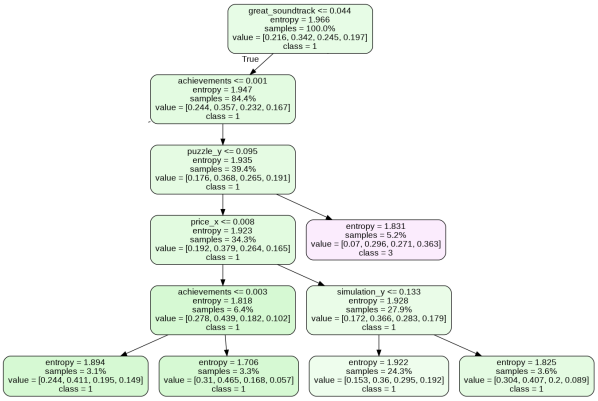


**Figure 8: Sub-Arbol Izquierdo-Drch**

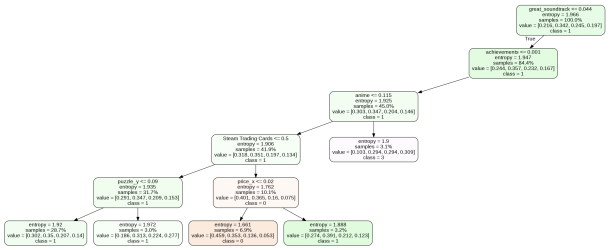

**Figure 9: Sub-Arbol Izquierdo-Izq**

dividen encontrando que si contiene la coleccion de cromos sacada por Steam y posee un precio bajo es probable que se trate de un juego de clase 0, en caso contrario a los mecionados los juegos quedan clasificados como clase 1.

*5.4.2 Algoritmos no traceables.* Para los algoritmos no traceables, vamos a estudiar tanto algoritmos de clasificación, como de regresión.

**CLASIFICACIÓN**

Como se nombró anteriormente, para realizar los algoritmos no traceables de clasificación, se ha usado la herramienta *BigML*, llevandose a cabo tres algoritmos, un Árbol de Decisión, un Random Forest, y por último, un OptiML, que creará multitud de modelos, y de estos, se escogerá el modelo que más precisión ofrezca.
Para analizar cada uno de los modelos resultantes, usaremos las medidas de precisión pertinentes.

**Árbol de Decisión**

En principio, el valor de exactitud del modelo no es alto, apenas un 27.7%, con un valor de F-measure de 0.2654.
Las características más importantes parecen apropiadas a un problema de este tipo, ya que nos encontramos con la importancia de la banda sonora del videojuego, su precio y diferentes aspectos relacionados con los gráficos que ofrece.

**Random Forest**

Se vuelve a obtener un modelo con una exactitud semejante al anterior, del 28.3% y una *F-measure* de 0.267.

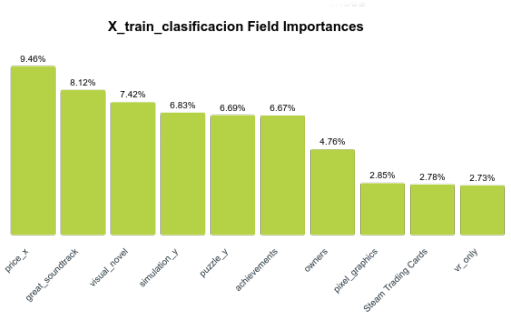


**Figure 10: Variables más importantes del Arbol de Decisión**

Uno de los aspectos más interesantes de este algoritmo, es que como característica más importante, nos encontramos con la categoría gramatical RB, que hace referencia a los adverbios, además, hay más categorías gramaticales entre las más importantes, como los verbos, complementos directos, etc. En principio, no parecen las características más adecuadas.

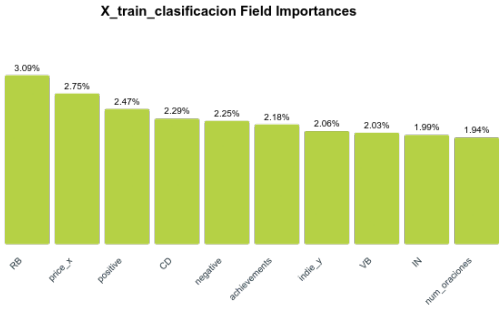


**Figure 11: Variables más importantes del Random Forest**

**Modelo optimizado**

Después de ver que los dos anteriores modelos no han obtenido una exactitud muy alta, procedemos a evaluar el resultado de poner en funcionamiento la herramienta *OptiML* ofrecida por *BigML*. De los modelos devueltos, nos quedamos con el que más exactitud nos ofrece, que es un bootstrap decision forest con 519 nodos.

Obtenemos ahora un modelo con un *accuracy* de un 42.1%, que mejora ampliamente los resultados de los dos modelos anteriores. Adjuntamos la matriz de confusión al completo de este modelo.

Sin duda, las características más importantes para este modelo tienen más sentido que las de los dos modelos anteriores. Además, se puede ver que escoge como características más importantes algunas que han sido utilizadas en el modelo traceable. Destaca por encima de todas la importancia que tiene la banda sonora de un videojuego a la hora de atraer a los jugadores, su precio, objetivos que tenga el videojuego y demás aspectos gráficos.

**REGRESIÓN**

Para evaluar la precisión los diferentes modelos resultantes usaremos las siguientes métricas de regresión: error absoluto medio

**Table 7: Matriz de confusión del modelo optimizado**

| | 0 | 1 | 2 | 3 | Actual | Recall |
|---|---|---|---|---|---|---|
| 0 | 156 | 294 | 18 | 19 | 487 | 32.03% |
| 1 | 91 | 563 | 67 | 78 | 799 | 70.46% |
| 2 | 40 | 329 | 83 | 99 | 551 | 15.06% |
| 3 | 17 | 193 | 59 | 146 | 415 | 35.18% |
| Predicted | 304 | 1,379 | 227 | 342 | 2,252 | **38.19%** |
| Precision | 51.32% | 40.83% | 36.56% | 42.69% | **42.85%** | **42.10%** |

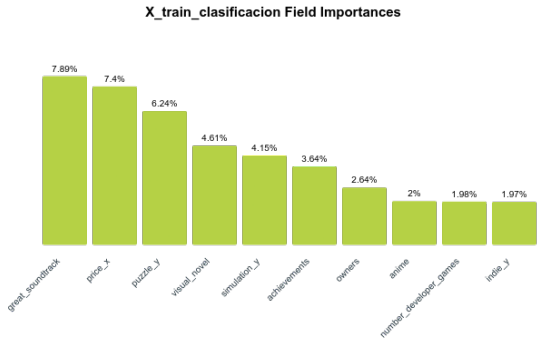


**Figure 12: Variables más importantes del modelo optimizado**

(MAE), error cuadrático medio (MSE) y la tasa de error por la raíz cuadrada de MSE (RMSE).
En primer lugar, vamos a evaluar tres modelos, resultantes de poner en funcionamiento los tres algoritmos nombrados anteriormente, para posteriormente, optimizar uno de ellos.

- **Árbol de Decisión:** Los resultados obtenidos son los siguientes:
  MAE: 0.1360682519136137
  MSE: 0.029887548332383947
  RMSE: 0.17288015598206738
- **Random Forest:** Los resultados obtenidos son los siguientes:
  MAE: 0.1271272507041679
  MSE: 0.02774853491992711
  RMSE: 0.16657891499204547
- **Boosting:** Los resultados obtenidos son los siguientes:
  MAE: 0.12275697388546472
  MSE: 0.026126786531047903
  RMSE: 0.16163782518658157

Los resultados son practicamente iguales, la diferencia es ínfima. Sí que se puede ver que los modelos con menos error son el de Random Forest y el de Boosting. Por tanto, se ha visto conveniente realizar una optimización a uno de los dos algoritmos, en este caso el Random Forest, aunque sea un poco peor (por casi nada) que el Boosting, y esto es debido, primeramente, por considerar que se haría más rápido, y además, se podría evitar el temido *overfitting*.

Dicho modelo se ha obtenido utilizando el método *RandomizedSearchCV*, que funciona de manera similar al *GridSearchCV*, pero no utilizando todos los parámetros, y especificando un número determinado de iteraciones. Para realizar esta optimización, al no contar con la herramienta de *OptiML*, se ha escogido un número determinado de características para poder realizar las distintas evaluaciones, como son el número de estimadores, la profundidad máxima, etc. Una vez terminado, se obtiene el mejor modelo, y se entrena, obteniendo con dicho modelo los siguientes valores de error:

- MAE: 0.12561357649782964
- MSE: 0.025556419945176195
- RMSE: 0.15986375431965869

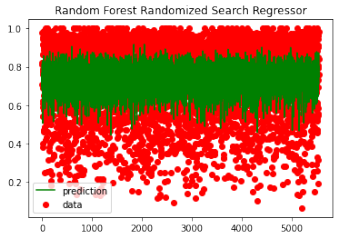


**Figure 13: Entrenamiento del RandomForest optimizado**

Se puede ver a simple vista que ha mejorado respecto de los anteriores modelos.

Como última prueba de este apartado, se ha procedido a comprobar el último modelo obtenido con un valor aleatorio del dataset de test. Para ello, se escoge un número aleatorio del 0 al número máximo de valores que hay en dicho dataset, y se obtiene tanto los valores del correspondiente juego en X_test, como el valor que tiene asignado dicho juego como target, en y_test. Una vez se tiene ambas cosas, se predice el valor del target con el modelo (previamente adaptando el array para poder permitir que se realice la predicción de un único valor), y una vez se tiene ya el valor predecido, se calcula los errores comparando este valor con el de y_test, consiguiendo los siguientes valores:

- MAE: 0.032900359161017634
- MSE: 0.001082433632923957
- RMSE: 0.032900359161017634

Como podemos ver, el modelo predice este único valor con bastante precisión.

## 6 APLICACIÓN DEL CONOCIMIENTO

Este proyecto podría servir para cualquier tipo de plataforma de videojuegos con catálogo online, ya sea el propio *Steam* o cualquier otra empresa, permitiendo obtener la valoración estimada de un videojuego. Esto podrá permitir maximizar ventas e incluso poder ajustar el precio del videojuego, dependiendo de la valoración que se haya obtenido. También le permitirá poder saber en qué clase de videojuegos poder invertir.

Además, también puede servir para ciertas empresas de desarrollo de videojuegos, que con este tipo de proyecto pueden saber por dónde enfocar sus próximos lanzamientos y que características debería reunir para poder conseguir la máxima valoración posible, que esto suele llevar a un gran número de ventas.

## 7 CONCLUSIONES DEL PROYECTO

Gracias a este estudio, se ha comprendido cual es el resultado de llevar a cabo el *Proceso KDD* y el porque este es necesario para realizar un estudio de minería de datos de forma eficiente y sofisticada.

Como principal línea de mejora que se podría aplicar a este proyecto, se debería de obtener modelos con una precisión superior. Para ello, se debería de volver a iterar en las diferentes fases del *Proceso KDD*.