

Implementación de una técnica de aprendizaje máquina sin el uso de un framework para la predicción de calificaciones de videojuegos en la plataforma Steam.

LUIS ÁNGEL CRUZ GARCÍA¹

Correo institucional: A01736345@tec.mx

¹ Tecnológico de Monterrey, Escuela de Ingeniería y Ciencias, Epigmenio González 500 Fracc. San Pablo 76130 Querétaro, Qro, México.

Primera revisión 24 de agosto del 2024

En este reporte se describirán los procesos llevados a cabo para realizar una técnica de aprendizaje máquina con el fin de comprender cómo es que dicho proceso funciona y con el objetivo de lograr un modelo eficaz que dé una predicción lo más acertada posible para la calificación o rating de un videojuego recibido en la plataforma llamada Steam, pasando desde la etapa de la elección de la base de datos hasta la estimación de la eficacia de dicho modelo.

1. Introducción

Las técnicas de aprendizaje máquina, mayormente conocidas por su nombre en inglés (Machine learning), son técnicas que surgieron como parte de la inteligencia artificial, teniendo en común que se basan en el análisis de datos para lograr un aprendizaje a base de ello y así implementarse en una tarea en específico, una de estas técnicas es el aprendizaje supervisado, la cual tiene como punto principal la búsqueda de predicciones, este aprendizaje también cuenta con implementaciones distintas dependiendo de la situación que se quiera analizar, en este reporte se implementará un problema de regresión lineal, es decir, se buscarán predecir valores numéricos, valores que van cambiando en un rango constantemente y que son continuos. Para ello se utilizarán diferentes métodos y herramientas que puedan ir mejorando el modelo y las predicciones.

Hablando acerca de la situación planteada, Steam es una plataforma en línea en donde diferentes usuarios de todo el mundo pueden adquirir y utilizar productos relacionados con videojuegos, contando con un sistema de reseñas, calificaciones, logros, etc. Utilizando estos criterios se le es asignado un rating general a un producto, el cual será el valor que se buscará predecir. Para esto se evaluarán qué datos son más importantes o cuáles deben de ser limpiados y transformados, además, se irán modificando diferentes elementos para alcanzar el mejor resultado posible sin el uso de frameworks que puedan facilitar el proceso pero evitarían una mayor comprensión y descripción del mismo.

2. Obtención del dataset

La base de datos fue obtenida de la plataforma Kaggle, en la cual podemos encontrar diferentes tipos de elementos relacionados con el aprendizaje máquina, en este caso utilizaremos los recursos llamados Steam Releases (1). Durante esta fase en un proceso más complejo, es donde se podría tomar la información mediante diferentes técnicas, por otra parte, también se debe de comprobar que la información obtenida es verídica y/o proviene de fuentes confiables. En

este caso específico se considerará que la información del dataset cuenta con estas características sin hacer un énfasis muy específico en esta etapa ya que no forma parte de los principales objetivos del proyecto, además de que no se maneja información sensible.

Uno de los elementos a destacar es que también se debe de identificar si existe documentación que ayude a entender mejor la información, en este caso, no se brinda información relevante, sin embargo, Kaggle muestra en resumen del dataset, que aunque es poco detallado, es de ayuda para identificar que se cuenta con alrededor de 66,000 instancias y 20 columnas, de donde se obtendrán los atributos y el label. También se puede ver que cuenta con una columna de id predefinida.

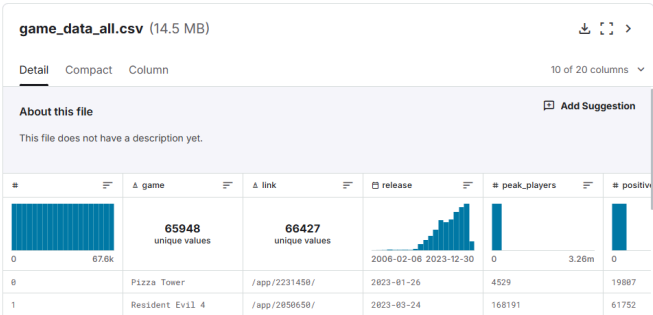


Fig. 1. Información inicial tomada de Kaggle

3. Primer análisis del dataset

A. Relevancia de los datos

Lo primero que se realizó fue guardar el dataset como un dataframe dentro de un código de Python, el cual será en el que se implementará todo el código, se utilizará esta estructura de datos debido a la versatilidad que nos brinda en su manejo y también debido a que al ser de las estructuras más utilizadas para este tipo de ejercicios existen muchas herramientas de Python diseñadas para su manejo, así como documentación de

la misma. Seguido de esto, encontramos los primeros datos que nos pueden dar una idea de qué cosas debemos limpiar, para esto se usó la función `info()`, dando el siguiente resultado:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 67571 entries, 0 to 67570
Data columns (total 20 columns):
#   Column              Non-Null Count  Dtype  
---  --
0   Unnamed: 0          67571 non-null  int64  
1   game                67571 non-null  object  
2   link                67571 non-null  object  
3   release             67571 non-null  object  
4   peak_players        67571 non-null  int64  
5   positive_reviews    67571 non-null  int64  
6   negative_reviews    67571 non-null  int64  
7   total_reviews       67571 non-null  int64  
8   rating              67571 non-null  float64
9   primary_genre       67561 non-null  object  
10  store_genres        67514 non-null  object  
11  publisher           67129 non-null  object  
12  developer           67445 non-null  object  
13  detected_technologies 60265 non-null  object  
14  store_asset_mod_time 67275 non-null  object  
15  review_percentage   47767 non-null  float64
16  players_right_now    67565 non-null  object  
17  24_hour_peak        67565 non-null  object  
18  all_time_peak       67571 non-null  int64  
19  all_time_peak_date   67565 non-null  object  
dtypes: float64(2), int64(6), object(12)
memory usage: 7.2+ MB
```

Fig. 2. Información del dataframe dada por Python

Lo que podemos observar es que cuenta con datos tanto numéricos como de tipo objeto, en donde podrían haber fechas o strings. Ya se pueden identificar las primeras columnas a eliminar, siendo la llamada "Unnamed 0" que es el id definido de cada dato, se eliminará debido a que Python asigna un id propio al momento de formar el dataframe. Por otra parte, se eliminarán todos los datos que tengan que ver con fechas o con información que desde primera vista se puede apreciar que no serán útiles, como "game" que contiene el nombre del juego, "link" que contiene el enlace de los juegos en Steam, además, se eliminará "store_genre" debido a que analizando los primeros datos con la función `head()` se puede observar que son categorías muy específicas y que realmente no terminarían teniendo un gran impacto en el modelo, por este mismo motivo tampoco se considerará el atributo "detected_technologies", por otra parte, se conservará "main_genre" que también cuenta con categorías, pero que a diferencia del atributo anterior, si puede llegar a tener alguna relación, ya que puede haber más juegos que compartan la misma característica. Mientras que para los que hacen referencias a empresas, como "developer" o "publisher" no se tenía suficiente información para saber si conservarlas o descartarlas, lo que hice fue contar los valores únicos de cada una de los atributos, dando 34224 para publisher y 39395 para developer, ambas son más de la mitad de los datos totales, lo que hace muy probable que o que muchos juegos formen parte de las mismas empresas y las demás solo tengan un juego en el dataset o que cada empresa tenga alrededor de 2 juegos, en cualquiera de los dos casos no se encontraría una relación útil y significativa para el modelo, por lo que también se descartaron.

B. Transformaciones iniciales

En la figura 2 se puede notar es que tanto "players_right_now" como "24_hour_peak" son de tipo objeto a pesar de tener datos numéricos, y es que como también pude notar en los primeros datos del dataframe, algunos números cuentan con comas, por lo que se eliminarán con la función `replace()` para después poder cambiar el tipo de dato a float con la función `astype()`.

C. Datos nulos

Con la función `isna()` y `sum()` pude observar que habían 19804 datos nulos en "review_percentage", al ser un número de datos considerable, consideré que era mejor conservar esos datos por lo que reemplacé esos datos nulos por la media de la columna.

Por otra parte, eliminé aquellos elementos que tenían datos nulos en las demás columnas, ya que solo eran 6 datos.

```
peak_players          0
positive_reviews      0
negative_reviews      0
total_reviews         0
rating               0
review_percentage    19804
players_right_now     6
24_hour_peak         6
all_time_peak        0
dtype: int64
```

Fig. 3. Datos nulos obtenidos

D. Datos en extremos

Realicé gráficas que compararan cada uno de los atributos con "rating" para de esta manera poder ver si había tendencias lineales y para poder localizar datos que estuvieran muy alejados en cuanto a su valor respecto a los demás, obteniendo la siguiente imagen.

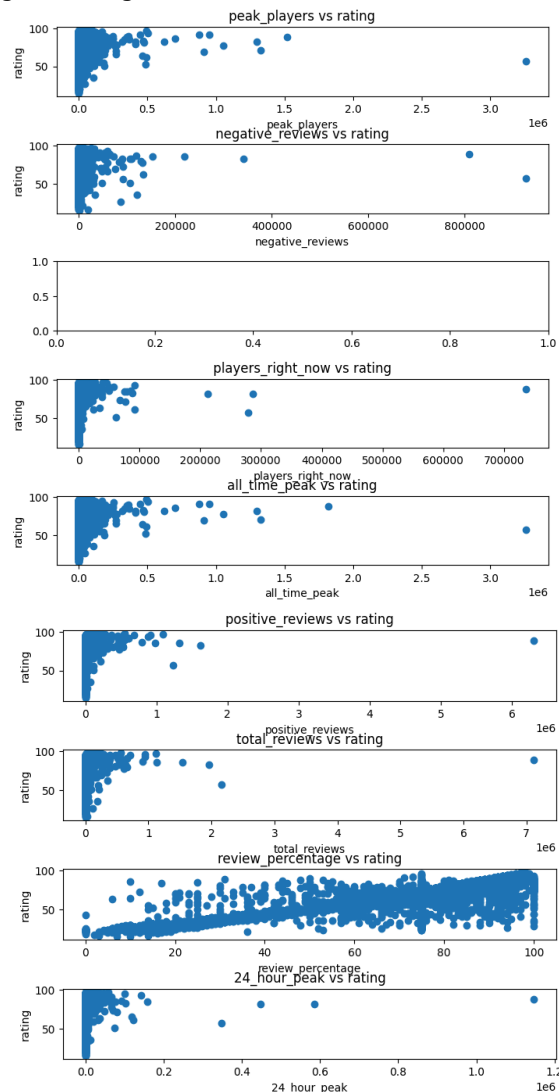


Fig. 4. Gráficas de los atributos contra rating

A primera instancia parece que todas las gráficas tienen una relación que podría ser lineal, al menos en cierta medida, todos con una pendiente positiva, sin embargo, en donde se nota de manera más clara la relación es con “review_percentage”.

Por otra parte, esta figura también es de utilidad para localizar valores que podrían ser errores o exclusiones, los cuales están presentes en su mayoría.

Por ejemplo, eliminar todos aquellos elementos que tienen un “peak_players” menor o igual a 0, ya que significaría que nunca ha sido jugado. También se borraron aquellas cantidades de reseñas muy positivas o muy negativas, ya que estas suelen ser o por polémicas o por situaciones muy específicas, como que un juego se haya vuelto popular por algún evento o que realmente sea demasiado bueno, pero son cosas que pasan con muy poca frecuencia, también utilicé esta lógica para las columnas que son de un “peak” de jugadores, obteniendo al final las siguientes gráficas:

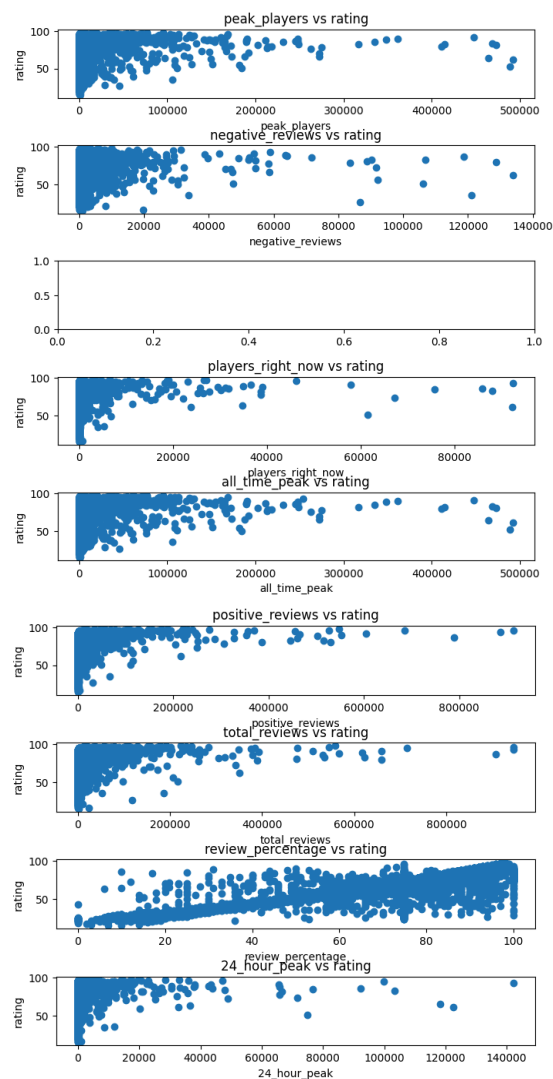


Fig. 5. Gráficas de los atributos modificados contra rating

Se pueden notar gráficas un poco más uniformes, en el sentido de que se tienen rangos de valores más claros, donde si bien aún existen algunos valores en los extremos, estos ya no se encuentran tan alejados, además de que están mejor distribuidos a diferencia de los que se encontraban al inicio.

4. One hot encoding

Como se mencionó anteriormente, se tiene aún un atributo categórico llamado “main_genre”, sin embargo, este cuenta con datos de tipo object, por lo que deberán ser cambiados a datos numéricos, para ello utilizaremos en conjunto la técnica one hot encoding, la cual separa los datos categóricos de n tipos diferentes en n atributos nuevos para el dataframe, en este caso, representando únicamente con el número 1 en caso de que pertenezca a esa categoría y con el número 0 en caso de que no pertenezca, la codificación de este procedimiento se puede apreciar en la función one_hot_encoding, la cual utiliza la función get_dummies que nos permite encontrar los valores categóricos de una columna dada. Después de obtener las nuevas columnas únicamente se concatenan a nuestro dataframe original, sin olvidar eliminar la columna inicial categórica, ya que ya no se utilizará.

5. Eligiendo pruebas

Lo siguiente que se hizo fue utilizar la técnica de split para entrenar y estimar el rendimiento del algoritmo que se explicará en pasos siguientes, en datos no considerados en su aprendizaje, esta técnica consiste en dividir el conjunto de datos (2). Una de las ventajas de esta técnica es que es eficiente para grandes cantidades de datos y si bien este caso no cuenta con demasiadas instancias, si podría resultar pesado para algunas computadoras, por lo que se aprovecha dicha ventaja.

En este caso utilicé el 80% de los datos para mi conjunto de entrenamiento y el otro 20% para el conjunto de prueba, este proceso se puede ver detallado en la función split del código realizado, en donde primero los valores son desordenados para después ser separados en el índice que corresponde del 0 al 80% y del 80% al 100%, también cabe destacar que se debe hacer tanto con X como con y, que en este caso será el rating de cada videojuego, mientras que X será todos los atributos que restan tras la depuración y tras el one hot encoding, a continuación se muestra un ejemplo de los datos de X train:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 38202 entries, 37698 to 27745
Data columns (total 36 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   peak_players                          38202 non-null  int64
1   positive_reviews                      38202 non-null  int64
2   negative_reviews                      38202 non-null  int64
3   total_reviews                        38202 non-null  int64
4   review_percentage                    38202 non-null  float64
5   players_right_now                    38202 non-null  float64
6   24_hour_peak                         38202 non-null  float64
7   all_time_peak                        38202 non-null  int64
8   Action (1)                           38202 non-null  float64
9   Adventure (25)                       38202 non-null  float64
10  Audio Production (52)                 38202 non-null  float64
11  Casual (4)                           38202 non-null  float64
12  Design & Illustration (53)           38202 non-null  float64
13  Early Access (70)                    38202 non-null  float64
14  Education (54)                       38202 non-null  float64
15  Free to Play (37)                    38202 non-null  float64
16  Game Development (60)                38202 non-null  float64
17  Gore (74)                            38202 non-null  float64
18  Indie (23)                           38202 non-null  float64
19  Massively Multiplayer (29)            38202 non-null  float64
20  Nudity (72)                           38202 non-null  float64
21  RPG (3)                              38202 non-null  float64
22  Racing (9)                           38202 non-null  float64
23  Sexual Content (71)                   38202 non-null  float64
24  Simulation (28)                       38202 non-null  float64
25  Sports (18)                           38202 non-null  float64
26  Strategy (2)                          38202 non-null  float64
27  Unknown Genre (0)                     38202 non-null  float64
28  Unknown Genre (21)                    38202 non-null  float64
29  Unknown Genre (33)                    38202 non-null  float64
30  Unknown Genre (34)                    38202 non-null  float64
31  Unknown Genre (6)                     38202 non-null  float64
32  Utilities (57)                        38202 non-null  float64
33  Video Production (58)                  38202 non-null  float64
34  Violent (73)                           38202 non-null  float64
35  Web Publishing (59)                   38202 non-null  float64
dtypes: float64(31), int64(5)
memory usage: 10.8 MB
```

Fig. 6. Resumen de datos de X train

6. Escalamiento

Muchos de los datos que podemos encontrar cuentan con diferentes unidades de medidas, ya que representan cantidades específicas, por ejemplo el número de reseñas o el número de usuarios, y en su mayoría son números grandes, sin embargo, si existe diferencia entre ellos y más con la columna "review_percentage", que al ser un porcentaje solo se mueve entre 0 y 100, es por eso que se debe de aplicar una técnica de escalado para que todos los datos tengan impacto en el modelo y ninguno sea desapercibido debido a sus unidades.

Para este caso implementé un normalizado por media, también llamado Zscore o método de transformación, en el cual la media y la desviación estándar se calculan por separado para cada columna. Este da valores que suelen estar entre -3 y 3, aunque también depende de los valores utilizados y de que tanta desviación haya entre ellos. (3)

$$z = \frac{x - \text{mean}(x)}{\text{stdev}(x)} \quad (1)$$

La implementación de esta técnica se puede observar en la función `scaling`, en la cual se calcula la media y la desviación estándar de cada columna y se realiza la operación para cada valor aprovechando la versatilidad de python para no tener que declarar ciclos.

7. Regresión lineal para la predicción

La regresión lineal simple puede observarse en diversas ocasiones y con diferentes aplicaciones. Básicamente, lo que se hace es calcular la tasa de cambio que tendrá una variable respecto a otra, para así trazar una línea que represente su comportamiento. Dicha línea se utiliza para predecir valores de la variable dependiente basados en valores de la variable independiente, que en este caso no será una sola variable sino que varias.

A. Función de hipótesis

La función hipótesis es una función matemática que el modelo utiliza para hacer predicciones a base de los datos de entrada, que en este caso serán los valores de entrenamiento y dando como resultado el valor predicho. Como se mencionó anteriormente, en este modelo se usará una regresión lineal, por lo que la función hipótesis cuenta con la siguiente forma:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p + \epsilon. \quad (2)$$

En donde las β son los parámetros que representan la pendiente en una regresión, mientras que ϵ (épsilon) es el término de error, que se identificará como bias en el modelo. (4)

Como en este caso se tienen varias variables independientes, que son los atributos, la función hipótesis se vuelve más grande, por lo que se tendrán varias "X".

B. Función de costo

La función de costo es la herramienta que el modelo usa para evaluar qué tan bien está haciendo sus predicciones, mide el error entre las predicciones del modelo y los valores reales. El objetivo del modelo es minimizar este error. Una manera común de hacerlo es usando el error cuadrático medio (MSE, por sus siglas en inglés), que se calcula sumando los cuadrados de las diferencias entre las predicciones y los valores reales, y luego

dividiendo por el número de datos. Siendo de lo más populares en los problemas de regresión lineal y también siendo el que se implementó en el código y que se puede observar en la función GD que hace referencia a la función de optimización que se hablará más adelante.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (3)$$

En este caso y_i son los valores reales, mientras que \hat{y}_i son las predicciones.

C. Función de optimización

Para mejorar las predicciones, el modelo ajusta los parámetros β mediante un proceso llamado descenso por gradiente. Esto implica ajustar repetidamente los parámetros en pequeñas cantidades, siguiendo la dirección que reduce el error. En el código implementado, este ajuste se hace en un bucle que continúa hasta que el modelo encuentra los mejores valores para los coeficientes, estos se pueden ver representados como β_i en la ecuación 2, o hasta que alcanza un número máximo de intentos (epochs), que es lo que normalmente sucede, ya que con modelos que no son perfectamente lineales entonces no se suele encontrar valores tan exactos y el error suele estancarse. (5)

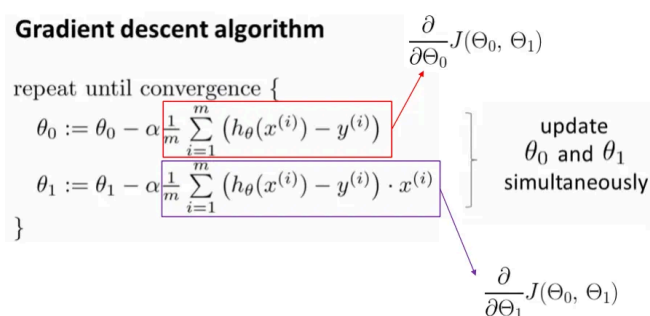


Fig. 7. Algoritmo de gradiente descendiente

En resumen, podemos decir que durante el proceso primero la función de hipótesis calcula las predicciones del modelo. Luego, se calcula el error entre estas predicciones y los valores reales. El gradiente se obtiene calculando la derivada de la función de costo respecto a los parámetros del modelo, esta derivada se aproxima usando la diferencia entre las predicciones y los valores reales, representados por el error. En el código se calcula el gradiente de la función de pérdida con respecto a cada parámetro del modelo mediante el producto punto de los parámetros con los errores, y se divide por el número de muestras para obtener el gradiente promedio, estas muestras son la m que podemos observar en la figura 7 que describe este algoritmo con ecuaciones.

D. Hiperparámetros

Los hiperparámetros son aquellos elementos que podemos cambiar de manera manual y que pueden afectar en el resultado del modelo, en este caso tenemos el learning rate, que se mencionó en el punto anterior, y el número de épocas. Para poder obtener un mejor resultado, fui variando en las combinaciones de estos valores. Tomando en cuenta los siguientes valores:

- Learning rate (alfa) de 0.0001, 0.001 y 0.1.
- Número de épocas de 500, 2000 y 4000.

Para ir viendo cómo cambian los valores, realicé iteraciones combinando los valores para después graficarlos, esto se

encuentra implementado en la función `test_hyper` y `plot_results` en el código, para al final obtener el siguiente resultado.

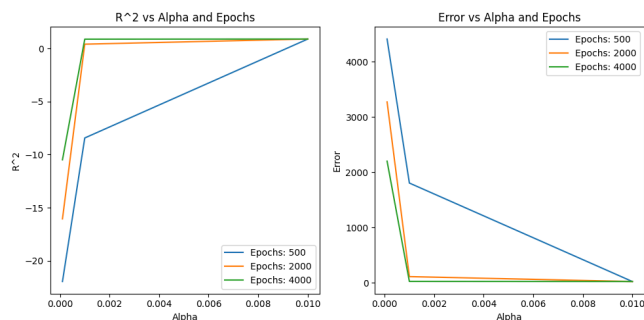


Fig. 8. Gráfica de R2 y Error variando los hiperparámetros

Lo que se puede observar en la figura 8 es que a medida de que se aumenta el número de épocas, se encuentra una mejora considerable, sin embargo, entre las 2000 y 4000 épocas no hay tanta diferencia a pesar de ser el doble, por lo que incrementar más la cantidad ya no significaba una mejora significativa, por otra parte, vemos que cuando el valor de alpha es muy pequeño, el descenso por gradiente se hace muy lento y requiere una gran cantidad de épocas para dar terminación en un resultado muy similar al que se obtiene con otros alphas, los otros dos valores de alpha se comportan de manera muy similar, sin embargo, con 0.001 tiende a ir más a la baja conforme se aumenten las épocas sin tender tanto a overfitting, por lo que será el valor elegido.

8. Resultados

A. Error

Lo que se busca es que el modelo pueda disminuir el error a través de las técnicas descritas anteriormente, y se puede apreciar en la figura 9 que se logró en buena medida, descendiendo de un valor muy alto hasta casi ser 0, y aunque es cierto que este valor como tal no es una buena medida para saber si el modelo está describiendo la mayoría de los datos, si nos puede ayudar a identificar que el error bajó considerablemente.

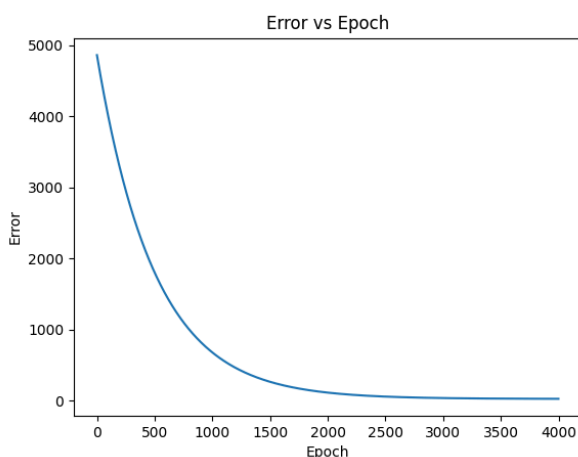


Fig. 9. Gráfica de Error contra época

B. R cuadrada

El R cuadrado es una métrica que mide la proporción de la variabilidad de los datos que el modelo es capaz de explicar. Este a diferencia del valor de error anterior, si nos ayuda a saber de manera directa qué tan funcional es el modelo.

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \quad (4)$$

La fórmula 4 calcula el R cuadrado, donde y_i es un valor real del label, \hat{y}_i es la predicción y \bar{y} es el promedio de los valores reales. Esta división de sumatorias al cuadrado se encuentra en la función `calc_rsquared` del código.

Para saber que tan eficiente es nuestro modelo debemos de hacer este cálculo tanto con los datos usados para el entrenamiento como con los datos reservados para las pruebas, ya que de esta manera se verá que no exista underfitting que es cuando gran parte de los valores no son explicados por el modelo, o que no haya overfitting que es cuando el modelo se encarga de memorizar y por lo mismo solo funciona con ciertos valores en concreto.

Los resultados obtenidos son los siguientes:

- Test R2: 0.8842607118643201
- Train R2: 0.8801323451303004

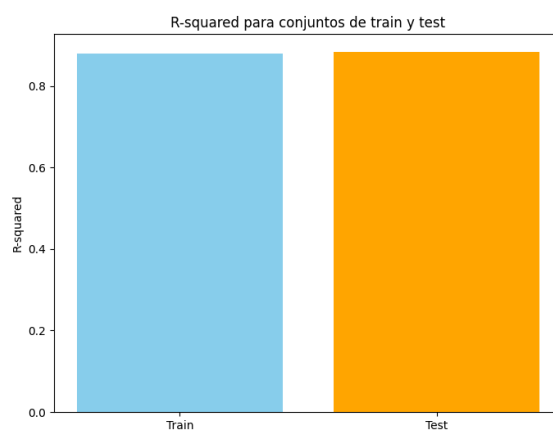


Fig. 10. Gráfica de R cuadrada en train y test

9. Evaluación de los resultados

Como se puede observar tanto en la figura 10 como en los valores exactos, el R cuadrado es muy similar para ambos casos, lo que significa que no existe una varianza muy grande independientemente del conjunto de datos utilizado, sin embargo, el valor ronda el 88%, por lo que no termina de explicar todos los valores del rating de los juegos, que es la variable a predecir, teniendo un sesgo medio aunque tendiendo ser bajo, al no explicar alrededor del 12% de los datos.

Este resultado también indica que no existe una condición de overfitting muy alta, por otra parte, aún podría ser mejorado el porcentaje de valores explicados por el modelo, es por eso que aún con una r cuadrada del 88% se podría considerar como underfitting en su ajuste.

Referencias

1. W Whim. (2023). Steam Score Prediction (1) [game_data_all.csv]. Kaggle. https://www.kaggle.com/datasets/whigmawhim/steam-releases/data?select=game_data_all.csv
2. Jason Brownlee. (2016). How to choose the right test options when evaluating machine learning algorithms. Machine Learning Mastery. <https://machinelearningmastery.com/how-to-choose-the-right-test-options-when-evaluating-machine-learning-algorithms/>
3. Microsoft. (2023.). Normalize data: Azure Machine Learning. Microsoft Learn.

<https://learn.microsoft.com/es-es/azure/machine-learning/component-reference/normalize-data?view=azureml-api-2>

4. Amazon Web Services. (2023.). ¿Qué es la regresión lineal? AWS.
<https://aws.amazon.com/es/what-is/linear-regression/>
5. Aprendizaje Maq. (2017). Regresión lineal con gradiente descendente. Medium.
<https://medium.com/@aprendizaje.maq/regresi%C3%B3n-lineal-con-gradiente-descendente-c3b5ca97e27c>