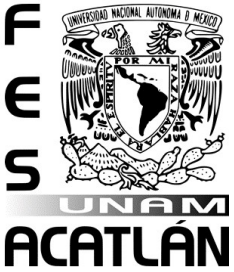


Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Acatlán

Diplomado Ciencia de Datos

Proyecto final

Moduló 2

Ambrocio Loreto Luis Manuel

Índice

1. Introducción

Durante las últimas décadas, con el auge de Youtube, Amazon, Netflix y muchos otros servicios web similares, los sistemas de recomendación han ido ocupando cada vez más lugar en nuestras vidas. Desde el comercio electrónico (sugerir a los compradores artículos que podrían interesarles) hasta la publicidad en línea (sugerir a los usuarios los contenidos adecuados que coincidan con sus preferencias), los sistemas de recomendación son hoy inevitables en nuestros viajes diarios en línea.

De manera muy general, los sistemas de recomendación son algoritmos destinados a sugerir elementos relevantes a los usuarios (artículos como películas para ver, textos para leer, productos para comprar o cualquier otra cosa, dependiendo de las industrias).

Los sistemas de recomendación son realmente críticos en algunas industrias, ya que pueden generar una gran cantidad de ingresos cuando son eficientes o también ser una forma de diferenciarse significativamente de la competencia.

El objetivo al final del diplomado es tener un sistema de recomendación que ayude a las personas a descubrir nuevos libros basado en sus gustos, libros leídos, etc. entre la información más importante que se requiere para entrenar un sistema de recomendación se encuentra:

- Tener información sobre calificaciones que usuarios hacen a los libros
- Tener formas de agrupar a los libros y/o usuarios

Lo que se busca en esta parte del proyecto es crear 2 modelos Machine Learning que nos ayuden en el sistema de recomendación y que cumplan con los siguientes objetivos:

- Tener una representación numérica de la reseña que el usuario dio a un libro
- Tener toda la columna de la categoría del libro llena (actualmente tiene muchos datos sin información)

2. Dataset

El conjunto de datos contiene reseñas de libros de amazon, fue obtenido en https://www.kaggle.com/data books-reviews, Este conjunto de datos contiene 2 archivos, como se ve en la siguiente figura:

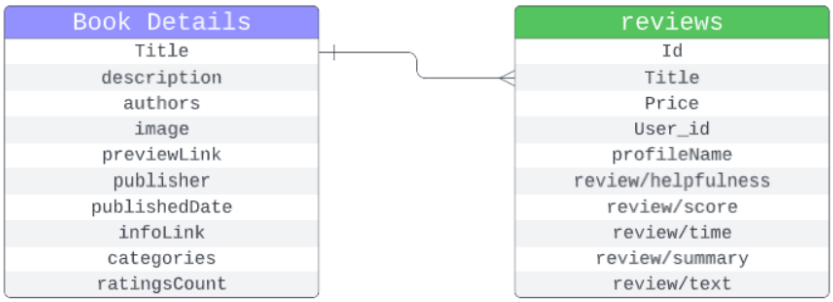


Figura 1: Diagrama de datos.

El primer archivo archivo de reseñas contiene 3 millones de comentarios de usuarios en 212404 libros únicos. El conjunto de datos es parte de la revisión de Amazon, el conjunto de

datos contiene reseñas de productos y metadatos de Amazon,las reseñas se obtuvieron entre mayo de 1996 y julio de 2014, este archivo tiene estos atributos

Feature	**Description**
id	El id del libro
Title	El título del libro
Price	El precio del libro
User_id	Id del usuario que calificó el libro
profileName	Nombre del usuario que calificó el libro
review/helpfulness	Calificación de utilidad de la reseña
review/score	Calificación de 0 a 5 para el libro
review/time	Momento de la reseña
review/summary	El resumen de una reseña de texto
review/text	El texto completo de una reseña

Cuadro 1: Descripción de las características del dataframe de reseñas.

El segundo archivo, el archivo Detalles de libros , contiene información detallada sobre 212404 libros únicos. El archivo se crea utilizando la API de Google Books para obtener información detallada sobre los libros que calificó en el primer archivo. Este archivo contiene estos atributos

Característica	**Descripción**
Title	Título del libro
Descripción	Descripción del libro
authors	Nombre de los autores del libro
image	URL de la portada del libro
previewLink	Enlace para acceder a este libro en Google Books
publisher	Nombre del editor del libro
publishedDate	Fecha de publicación del libro
infoLink	Enlace para obtener más información sobre el libro en Google Books
categories	Géneros de libros
ratingsCount	Calificación promedio del libro

Cuadro 2: Descripción de las características de un libro.

Nos interesa tener historia por usuario de interacciones con libros, entre más reseñas haya hecho es mejor, por tal motivo no excluimos a aquellos usuarios con poca reseñas, si el usuario tenía 3 reseñas o mas es un usuario que consideramos para entrenar nuestros modelos, los demás los omitimos, nos quedamos con 1,454,613 reseñas, eliminamos 838,658 usuarios de aprox 1,000,000

3. Limpieza de datos

3.1. Duplicidad

Primero se revisó duplicados por id usuario y id libro, para tener una reseña por libro, después se reviso que cada titulo de libro tuviera exactamente un id libro, se observo que para un mismo libro había casos donde existían más de un identificador, se dejo un único id por titulo de libro, se hizo una limpieza básica del libro y se modio distancia de palabras para validar que no hubieran títulos que se refirieran al mismo libro pero con una variación de unas cuantas letras, el proceso era muy pesado así que se simplifico primero filtrando los títulos

similares, se calculo la similitud usando la distancia coseno de la vectorización tfidf, teniendo títulos similares ahora si se calculo la distancia levenshtein y se unificaron aquellos libros con una distancia pequeña, después de esto se valido de nuevo que cada titulo de libro tuviera solo un id y que cada usuario calificara solo una vez a un libro, hasta este punto nos quedamos con 206,843 libros y 1,055,165 reseñas.

3.2. Completitud

Primero se reviso el dataset de reviews, se identifico que la columna del precio de los libros tenia más del 80 % de los datos nulos por lo que esta columna se elimino , después se encontraron 40 registros donde el titulo era nulo, sin el titulo del libro no podíamos acceder a los datos del libro, se consideró que estos datos son importantes y por eso se tuvieron que eliminar los registros, en el nombre del usuario los nulos se sustituyo por 'no information' , los nulos en review_summary y review_summary se eliminaron ya que son características importantes y no hay forma de imputarlos. Después se reviso el dataset de la información de los libros, de nuevo, los registros donde el titulo es nulo se eliminaron , la columna rating_count tenia más del 75 % de registros nulos, esta columna se elimino, después se eliminaron los registros donde más de la mitad de las columnas fueran null, esto porque esos registros no tendrían información valiosa, t_description, t_authors y v_categories son datos que consideramos importantes, eliminamos los casos donde todos estos datos fueran faltantes, los datos faltantes de las columnas d_published_date, t_publisher, t_authors y t_description se llenaron con 'no information', las columnas t_info_link, t_image , t_preview_link no tenían mucha información, pero no aportaban mucha información así que se eliminaron

3.3. Precisión/Orden

Primero se revisó que no hubiera variables unitarias, se revisó principalmente que el tipo de dato fuera el correcto, se revisaron todas las variables pero solo vamos a mencionar las variables que se modificaron, la variable review_helpfulness estaba como tipo de dato cadena y formato era $\frac{n_u}{n_c}$ donde n_u es la cantidad de usuarios que les fue útil la reseña y n_u es la cantidad de usuarios que calificaron la reseña, por ejemplo si 10 personas calificaron una reseña y a 5 les fue útil la reseña entonces el dato era $\frac{5}{10}$, pero nos es más útil esto como un valor numérico, es decir hacer la división de la expresión, sin embargo había un problema porque no es lo mismo 1/1 que 100/100 (ambos tendrían un review_helpfulness de 1 pero al segundo lo calificaron 100 veces), para solucionar esto se agrega la variable c_n_cal_review que contiene las veces que se califico la reseña y review_helpfulness se paso a numérica haciendo la división , de esa forma tendríamos el porcentaje de usuarios que les gusto la reseña en una variable, y en la otra la cantidad de usuarios que califico la reseña.

Posteriormente se reviso las variables de tipo fecha, para d_published_date la variable era de tipo cadena y esto era porque había fechas en diferentes formatos y no se podían pasar a date automáticamente, los diferentes formatos eran:

- YYYY-MM-DD
- YYYY-MM-DD HH:mm:SS
- YYYY
- YYYY-MM
- YYY*

Se pasó el formato a un formato único de YYYY-MM-DD y luego se paso a tipo de dato date, los formatos que no pertenecían a ninguno de estos 5 se pasaron a Null, d_review_time contenía números en lugar de fechas, pero el numero correspondía a los segundos después de 1970, esto se paso a fecha y quedo en el formato correcto.

3.4. Consistencia

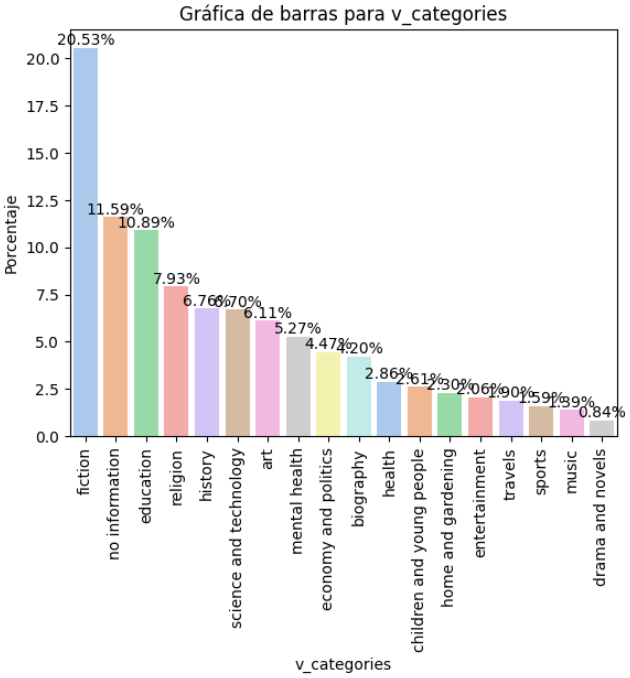
De nuevo, solo se mencionan las variables que recibieron una modificación, como mencionamos anteriormente, la variable c_review_helpfulness se modifico para que fuera una proporción, es producto de una división y no deben de existir valores mayores a 1, se encontraron valores mayores a 1, se dedujo que el denominador y numerador estaban invertidos al momento de calcularse y se calculo el inverso para obtener el valor correcto en estos casos, después se reviso d_review_time y como mencionamos antes esta variable corresponde a la fecha del review, el conjunto de datos fue recolectado de 1996 a 2014 y se encontraron valores menores y mayores a estas fechas, dado que no podemos saber la fecha correcta para estos casos procedimos a eliminarlos, se eliminaron porque la fecha de review es una dato importante a la hora de crea el modelo de recomendación, para la fecha de publicación del libro encontramos algo similar, había fechas mayores al 2014 y esto era imposible, es correcto tener fechas de publicación menores a 1996 pero es incorrecto tener fechas de publicación mayores a 2014, estos registros se pusieron como 'sin information', se tomo esta decisión porque no afecta mucho el no tener el dato exacto, después se detecto que para la fecha de publicación del libro había muchos datos atípicos, entonces procedimos a normalizar la variable y volverla categórica, las categorías que se crearon fueron las siguientes y estan balanceadas:

- no information
- (1681-12-31 , 1984-01-01]
- (1984-01-01, 1996-08-01]
- (1996-08-01, 2001-01-01]
- (2001-01-01, 2005-01-01]
- (2005-01-01, 2014-12-31]

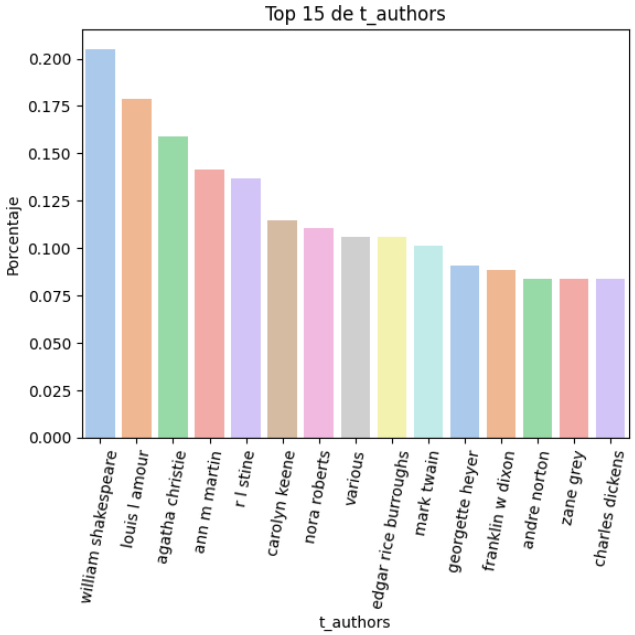
Después para la variable de categorías del libro se hizo una limpieza básica, se unificaron categorías mediante distancia de palabras tomando un umbral de 2 letras de diferencia, es decir si tenían 1 o 2 letras de diferencia se unificaban, y después se normalizo la variable, por ejemplo, todas las subcategorias de libros de religión se clasificaron solo como religión, para la variable de autor se hizo una limpieza básica y se se unificaron los autores similares con distancia de palabras y un umbral de 3 palabras, para las otras variables de texto solo se hizo una limpieza de texto y se quitaron stop words, finalmente con las variables ya limpiezas se volvió a validar duplicados por para los títulos de los libros, al final de toda esta limpieza de libros acabamos con 126810 libros y 1054894 reviews.

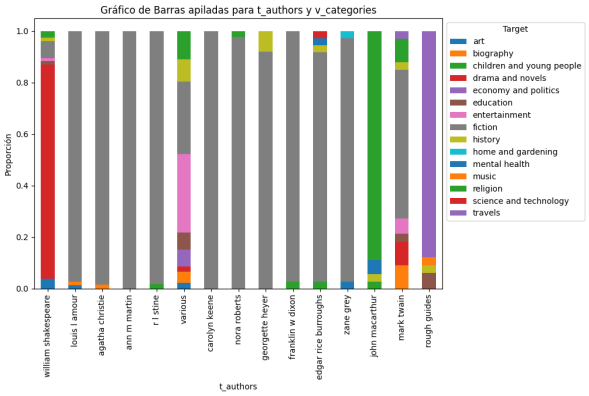
4. Modelo para predecir la categoría del libro

El propósito del primer modelos es crear un modelo para predecir la categoría de un libro basado en su descripción, titulo, autor, editorial y/o fecha esto porque tenemos muchos casos donde la categoría no es conocida y queremos rellenar esa información con la predicción de un modelo, en esta parte solo usaremos el dataset de los datos de los libros. Primero empezamos revisando un gráfico de barras para ver como se distribuye nuestra target.

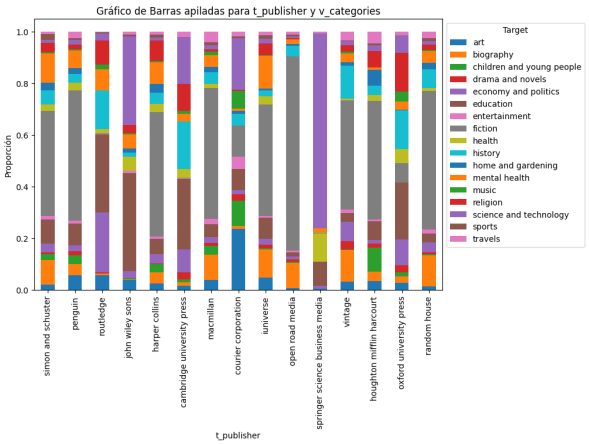


Confirmamos que hay una gran cantidad de datos sin información, además vemos cuales son las categorías que más consumen las personas, vemos que la target esta des-balanceada, revisemos algunas gráficas de el top de autores y como se distribuye la categoría dentro de cada autor.

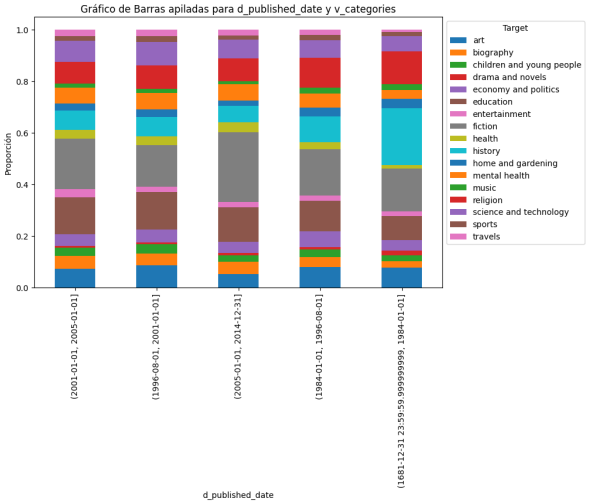




En esta ultima gráfica podemos ver que los autores (por lo menos el top 15) si ayudan a predecir la categoría del libro, por ejemplo William Shakespeare publicó libros principalmente de Drama y novelas, mientras que Louis L'Amour publicaba libros principalmente de Ficción, para la editorial y fecha de publicación podemos encontrar algo similar.



Para las editoriales vemos que algunas son más de ciencia y tecnología como la editorial springer science bussines media, u otras son más de ficción como la editorial pengüin.



Para la fecha también hay diferencias entre cada rango, por ejemplo en el rango de 1681 a 1984 sobresalen un poco más los libros de historia mientras que en el rango de 2005 a 2014 sobresalen un poco más los libros de ficción, hicimos también bolsas de palabras para las diferentes categorías para ver si entre cada categorías sobresalían diferentes palabras, encontramos que si hay diferencias de la bolsa de palabras entre cada categoría, pero solo mostraremos para 3 categorías ya que son 17 categorías.

Ficción:

Figura 2: Bolsa de palabras de la descripción para libros de ficción

Educación:

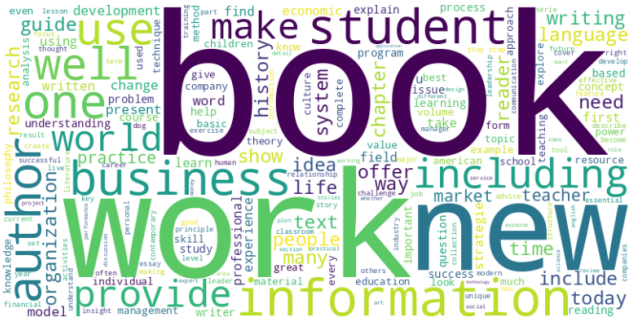


Figura 3: Bolsa de palabras de la descripción para libros de educación

Religión:

Figura 4: Bolsa de palabras de la descripción para libros de religión

En los 3 casos sobresalen palabras diferentes, por ejemplo para los libros de ficción sobresalen palabras como love, life, family, para libros de educación sobresalen libros como book, work, student, information, y para lo libros religiosos hay palabras como god, christian, bible etc.

Después procedimos a armar el conjunto de datos que sirvió para entrenar el modelo, se construyo con los datos que tienen información de su categoría, los que no tienen este dato no los tomamos en cuenta, también conservemos solo los datos donde venga la descripción, si no viene los omitimos ya que sera nuestra principal información para hacer la predicción.

4.1. Modelo 1

Empezamos con un modelo sencillo, usando solo naive bayes y usando solo la variable t_description, para tener un formato numérico se utilizo TF-IDF con los siguientes parámetros.

ngram_range=(1, 3):

Este parámetro determina qué combinaciones de palabras (n-gramas) se deben considerar al construir el vocabulario. En este caso, (1, 3) significa que se considerarán unigramas, bigramas y trigramas.

min_df=10:

Este parámetro establece el número mínimo de documentos en los que un término debe aparecer para ser incluido en el vocabulario. En este caso, solo se incluirán términos que aparecen en al menos 10 documentos.

max_features=100:

Este parámetro limita el número máximo de características (términos) que se incluirán en el vocabulario. Aquí, se limita a 100 términos con mayor TF-IDF.

Después se hizo una búsqueda de parámetros para buscar el mejor tipo de modelo y parámetros para Naive Bayes, el mejor modelo fue un Bernoulli Naive Bayes con los siguientes hiperparametros.

alpha = 1.0294723681580236

Este es el parámetro de suavizado que se aplica a las probabilidades de las características. Un valor de α mayor implica un suavizado más fuerte.

binarize =0.1770397105876216

Este parámetro establece un umbral para binarizar (convertir en 0 o 1) las características. Si `binarize` no es `None`, las características se binarizan según si son mayores que el umbral proporcionado. Antes de pasar a ver las métricas del modelo recordemos los siguientes conceptos:

- **Verdaderos positivos:** Es el número de casos positivos que el modelo ha clasificado correctamente como positivos.
- **Verdaderos negativos:** Es el número de casos negativos que el modelo ha clasificado correctamente como negativos.
- **Falsos positivos:** Es el número de casos negativos que el modelo ha clasificado incorrectamente como positivos.
- **Falsos negativos:** Es el número de casos positivos que el modelo ha clasificado incorrectamente como negativos.

Accuracy :

- **Definición:** El Accuracy mide cuántas de las predicciones totales son correctas, es decir, la proporción de predicciones correctas entre todas las predicciones realizadas.
- **Fórmula:** Accuracy $= \frac{\text{Número de predicciones correctas}}{\text{Número total de predicciones}}$
- **Ejemplo:** Si tienes 90 predicciones correctas de un total de 100, la exactitud sería del 90

Precision:

- **Definición:** La Precision mide la proporción de predicciones positivas que fueron verdaderamente positivas. En otras palabras, se enfoca en la calidad de las predicciones positivas.
- **Fórmula:** $\text{Precision} = \frac{\text{Verdaderos positivos}}{\text{Verdaderos positivos + Falsos positivos}}$
- **Ejemplo:** Si tienes 80 verdaderos positivos y 20 falsos positivos, la precisión sería del 80

Recall :

- **Definición:** El Recall mide la proporción de instancias positivas que fueron correctamente identificadas por el modelo. Se enfoca en la capacidad del modelo para capturar todos los casos positivos.
- **Fórmula:** $\text{Recall} = \frac{\text{Verdaderos positivos}}{\text{Verdaderos positivos + Falsos negativos}}$
- **Ejemplo:** Si tienes 80 verdaderos positivos y 10 falsos negativos, la recuperación sería del 80

F1-Score (Puntuación F1):

- **Definición:** El F1-score es una métrica que combina precision y recall en un solo valor. Es particularmente útil cuando hay un desequilibrio entre las clases.
- **Fórmula:** $\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision + Recall}}$.
- **Ejemplo:** Si tienes una precision del 80

Dicho lo anterior pasamos a ver las métricas de este primer modelo:

Cuadro 3: Resultados de entrenamiento y prueba

Métrica	**Train Mean**	**Train Std**	**Test Mean**	**Test Std**
Accuracy	0.385738	0.002992	0.385360	0.005367
Precision	0.264530	0.010965	0.270825	0.012171
Recall	0.211319	0.002482	0.211621	0.003865
F1-Score	0.214749	0.003079	0.215551	0.004232

El resultado no es alto para ninguna de las 3 métricas y no se ve ningún sobreajuste, hay que considerar que para este primer modelo se creó un modelo sencillo y solo una variable, además tenemos 17 clases, por lo que el resultado no es tan malo y sabemos que la variable si es predicativa, por ultimo se muestra la matriz de confusión:

Figura 5: Matriz de confusión para el modelo 1

4.2. Modelo 2

El modelo 2 de nuevo es un modelo sencillo, un Naive Bayes, el objetivo es ver que tan predictivas son las variables t_publisher t_authors y d_published_date, como vimos en las gráficas estas variables si nos pueden aportar información, nos apoyamos en la moda de cada valor para ver si con esa información se podía predecir la categoría del libro, es decir, para un autor, por ejemplo William Shakespeare, se tomo la moda de la categoría a la que pertenecen sus libros, esa categoría se sustituyo en todos los registros donde el autor sea William Shakespeare, se hizo lo mismo para la editorial y fecha de publicación, después esos valores se codificaron a One hot encoding para pasar una entrada numérica al modelo, se hizo una búsqueda de parámetros e hiper-parámetros para encontrar el menor error , el mejor modelo es un Bernoulli Naive Bayes con alpha = 0.03527638531315791 y binarize = 0.04634927790753873, a continuación se muestran las métricas obtenidas.

Cuadro 4: Resultados de entrenamiento y prueba

Métrica	**Train Mean**	**Train Std**	**Test Mean**	**Test Std**
Accuracy	0.535074	0.002910	0.442173	0.004950
Precision	0.652156	0.005873	0.512278	0.009886
Recall	0.439422	0.005980	0.317319	0.005458
F1-Score	0.501286	0.005876	0.362730	0.006128

De nuevo, confirmamos que estas variables son predicativas y que nos dio un resultado mejor al azar, incluso mejor al resultado obtenido anteriormente, sin embargo buscamos un mejor modelo, mostramos la matriz de confusión en la siguiente imagen.

Figura 6: Matriz de confusión para el modelo 2

4.3. Modelo 3

El modelo 3 de nuevo, es un Naive Bayes y tiene como objetivo ver que tan predictora es la variable del titulo por si sola, es similar al modelo 1 en donde se considero solo la descripción, el tratamiento que esta variable recibió igual es el mismo al quitar stop words y haciendo una limpieza básica, y después se hizo una vectorización TF-IDF con los siguientes parámetros:

- ngram range=(1, 3)
- min df=10
- max features=100

El mejor modelo es un Bernoulli Naive Bayes con alpha = 0.7607364110014678 y binarize = 0.19385043856097783, a continuación se muestran las métricas obtenidas.

Cuadro 5: Resultados de entrenamiento y prueba

Métrica	**Train Mean**	**Train Std**	**Test Mean**	**Test Std**
Accuracy	0.368384	0.003029	0.368460	0.005183
Precision	0.334051	0.005322	0.332047	0.013437
Recall	0.218941	0.003029	0.216004	0.004986
F1-Score	0.239908	0.003610	0.235424	0.006414

Consideran nuestra métrica de más interés el F1-Scores vemos que es ligeramente mejor al primer modelo pero no mejor que el segundo, aún así vemos que solo el titulo nos puede ayudar a predecir la categoría del libro aunque las métricas no son las mejores, mostramos la matriz de confusión, en las 3 matrices vistas hasta ahora el principal error visto es que se predice ficción como la categoría y la categoría correcta es alguna otra, esto podría deberse a que la target no esta balanceada, no vamos a hacer over-sampling o sub-sampling porque prediríamos muchos datos o agregaríamos mucho sesgo (según la elección del método) .

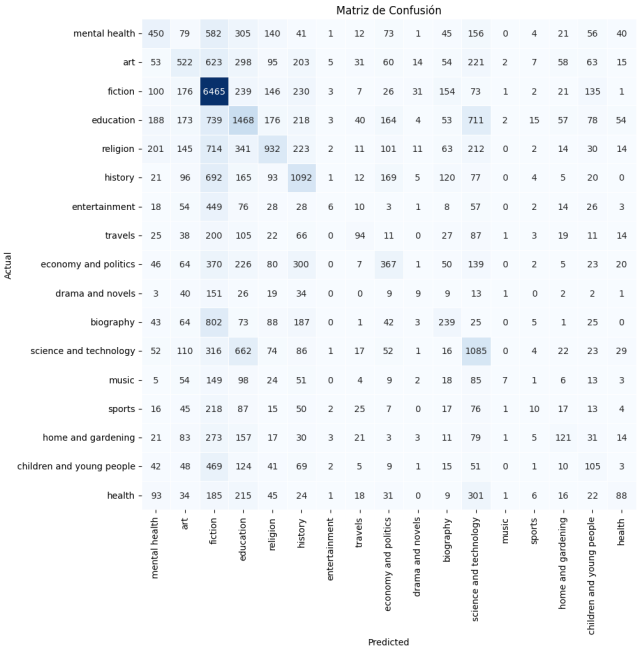


Figura 7: Matriz de confusión para el modelo 3

4.4. Modelo 4 y 5

Visto lo anterior se procedió a crear 2 modelos con todas las variables utilizadas en los modelos anteriores, para ambos modelos la entrada fue la misma pero cambio el tipo de modelo, en las transformaciones se hicieron las mismas que en los modelos anteriores, el único cambio que hubo fue en el parámetro max features=100 de la vectorización TF-IDF, se unieron todas las variables (ya transformadas) obtenidas y se hizo una búsqueda de hiper-parámetros para ambos modelos, para el modelo 4 se busco ajustar un random forest y los mejores parámetros son los siguientes:

- **max_depth = 26** La profundidad máxima de cada árbol en el bosque. En este caso, se ha establecido en 26, lo que significa que cada árbol en el bosque puede tener un máximo de 26 niveles.
- **max_features ='sqrt'** El número máximo de características a considerar al dividir un nodo. En este caso, se utiliza 'sqrt', lo que significa que el número de características consideradas en cada división es la raíz cuadrada del número total de características.
- **min_samples_leaf = 0.012** El número mínimo de muestras requeridas para formar una hoja en el árbol. En este caso, se ha establecido en 0.012199238402872157, lo que indica una fracción del total de muestras.
- **min_samples_split = 0.047** El número mínimo de muestras requeridas para dividir un nodo interno. En este caso, se ha establecido en 0.04702093862738248, lo que indica una fracción del total de muestras.
- **n_estimators = 250** El número de árboles en el bosque. En este caso, se ha establecido en 250, lo que significa que se están utilizando 250 árboles en el bosque.

En el modelo 5 se busco ajustar un Gradient Boosting Classifier, los parámetros óptimos fueran los siguientes

- **learning_rate = 0.146** La tasa de aprendizaje, que controla la contribución de cada árbol al modelo. En este caso, se ha establecido en 0.14653120428735267.

- **max_depth = 30**
- **max_features = 'log2'**
- **min_samples_leaf = 0.118**
- **min_samples_split = 0.921**
- **n_estimators = 150**

A continuación se muestras las métricas obtenidas en ambos modelos.

Cuadro 6: Resultados de entrenamiento y prueba en el modelo 4

Métrica	**Train Mean**	**Train Std**	**Test Mean**	**Test Std**
Accuracy	0.425433	0.002093	0.408679	0.005007
Precision	0.186989	0.005843	0.185402	0.010858
Recall	0.181174	0.001283	0.171684	0.003087
F1-Score	0.154154	0.001405	0.147472	0.003047

Cuadro 7: Resultados de entrenamiento y prueba en el modelo 5

Métrica	**Train Mean**	**Train Std**	**Test Mean**	**Test Std**
Accuracy	0.320212	0.006857	0.313799	0.007468
Precision	0.101032	0.020614	0.092752	0.017800
Recall	0.111555	0.005699	0.107987	0.005844
F1-Score	0.078701	0.007517	0.077507	0.006160

Podemos ver que los dos modelos a pesar de ser más robustos tiene un peor rendimiento que nuestros modelos anteriores, se esperaría que al combinar todas las variables y meterlas a un modelo más robusto el modelo mejoraría aunque sea solo un poco, pero paso lo contrario, se tuvo un modelo peor, esto podría deberse a que al agregar muchas variables se agrega también mucho ruido y/o multi-colinealidad, en el próximo modelo haremos selección de características para tratar de mitigar esto, en las siguientes imágenes vemos las matrices de confusión y vemos que la predicción es peor que en los modelos anteriores.

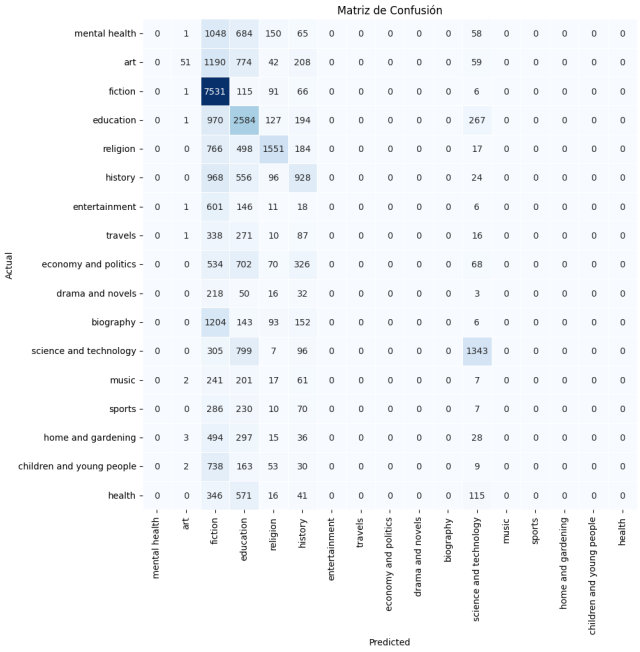


Figura 8: Matriz de confusión para el modelo 4

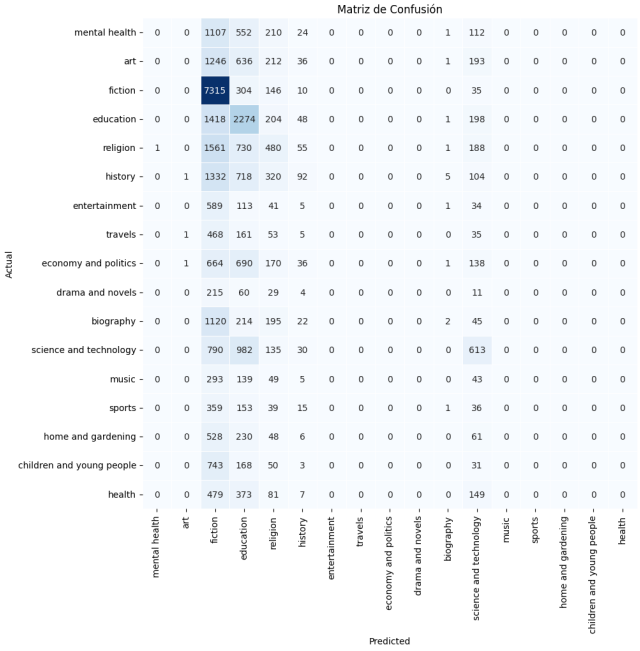


Figura 9: Matriz de confusión para el modelo 5

4.5. Modelo 6

Para mitigar el ruido y/o multicolinalidad que se agrega al agregar todas las variables se hizo una selección de características , se utilizo la información muta y para conservar solo variables que más importancia tenían, la información mutua es una medida de la dependencia entre dos variables aleatorias. En el contexto de la estadística, la información mutua cuantifica la cantidad de información que se obtiene sobre una variable aleatoria al conocer el valor de otra variable aleatoria. La información mutua es igual a cero si y solo si X e Y son independientes, lo que significa que el conocimiento del valor de una variable no proporciona información sobre la otra. Cuanto mayor sea el valor de la información mutua, mayor será la dependencia entre las dos variables. Se calculo la información mutua para todas las variables comparadas con la

target, se utilizo un umbral de 0.01, es decir se dejaron solo aquellas variables (ya vectorizadas) que tuvieran un valor mayor a este, se entrenaron varios modelos y se encontró que el mejor era un Ada Boost Classifier con los siguientes parámetros:

Parámetros de AdaBoost:

- **Algoritmo:SAMME.R** AdaBoost puede utilizar diferentes algoritmos de clasificación débil para construir el conjunto final de clasificadores. 'SAMME' y 'SAMME.R' son dos opciones comunes. 'SAMME' (Stagewise Additive Modeling using a Multiclass Exponential loss function) es el algoritmo original de AdaBoost para clasificación multiclase, mientras que 'SAMME.R' (SAMME Real) es una variante que utiliza estimadores de probabilidad reales en lugar de 1 o -1 como en 'SAMME'. 'SAMME.R' a menudo es preferido cuando
- **Tasa de Aprendizaje = 0.5238289272560606**
- **Número de Estimadores = 350**

Las métricas obtenidas son las siguientes:

Cuadro 8: Resultados de entrenamiento y prueba

Métrica	Train Mean	Train Std	Test Mean	Test Std
Accuracy	0.590893	0.003636	0.484292	0.007229
Precision	0.611327	0.006998	0.408673	0.015462
Recall	0.452497	0.006096	0.317151	0.010292
F1-Score	0.498641	0.006555	0.338284	0.011521

Los resultados están muy a la par con el modelo 2, el Accuracy y Recall son ligeramente mejores, pero la precision y F1-Score son ligeramente peores, la matriz de confusión no se ve considerablemente mejor o peor al modelo 2, hasta el momento estos 2 son nuestros mejores modelos.

Figura 10: Matriz de confusión para el modelo 6

4.6. Modelo 7 (Mejor Modelo)

Como tercer modelo se entreno una red neuronal artificial, se pasaron 3 diferentes tipos de entradas, la vectorización del titulo, la moda de autores, editorial y fecha de publicación en formato One hot encoding y la descripción vectorizada de la descripción, para este modelo no se utilizo la vectorización TF-IDF, se utilizó lo que se conoce como tokenizar, básicamente es identificar cada palabra con un id, y sustituir la palabra del texto por su id, por ejemplo si la entrada es 'me gusto el libro' una tokenización valida podría ser [1,4,28,12] , el vocabulario del tokenizer se definió como 50000, es decir tenemos 50000 id's para las diferentes palabras del texto, para las palabras que no alcanzaron a tener token (porque el vocabulario es de más de 50000 y solo puede haber uno por palabra) se les asigna uno por default que regularmente es 0, se estableció una secuencia de tokens de 300 para la descripción del libro y 30 para el titulo, esto quiere decir que cada secuencia de token de las descripciones tiene un tamaño de 300 y para los títulos tiene un tamaño de 30, si el numero de tokens es menor al tamaño de la secuencia se rellena con 0 a la derecha, si es mayor al tamaño se trunca a la derecha, una vez hecho esto se construyo la arquitectura la cual es la siguiente:

- **Capa de entrada:** Tres entradas diferentes:
    1. Descripción tokenizada seguida de una capa de embedding de dimensión 300.
    2. Título tokenizado seguido de una capa de embedding de dimensión 300.
    3. Autor, editorial y fecha en One-Hot Encoding.
- **Capa Oculta:** Concatenación de las tres entradas, seguida de una capa oculta con 64 unidades y función de activación ReLU.
- **Capa Dropout:** Capa de Dropout con parámetro 0.5 para regularización.
- **Capa de Salida:** Capa de salida con 17 unidades y función de activación softmax.
- **Optimizador:** Adam con tasa de aprendizaje (Learning Rate) de 0.00003.

Se entreno por 20 épocas y el resultado de las métricas fue el de el cuadro 9.

Cuadro 9: Resultados de entrenamiento y prueba

Métrica	**Train**	**Test**
Accuracy	0.993378	0.699530
Precision	0.992705	0.676590
Recall	0.986190	0.592201
F1-Score	0.989378	0.620761

Vemos un poco de sobreajuste aun con la capa de Dropout, se probaron varias arquitecturas y parámetros y esta fue la mejor, debido al tiempo de ejecución no se pudo hacer más pruebas con más arquitecturas y parametros para mitigar el sobreajuste, sin embargo, aun con sobreajuste se observa que las metricas del test superan a todos los modelos anteriores, el menor F1-score en el etst obtenido hasta el momento era de 0.36, por lo que decidimos escoger este modelo como nuestro mejor modelo, a continuación se observa la matriz de confusión la cual se ve mejor que en los modelo anteriores.

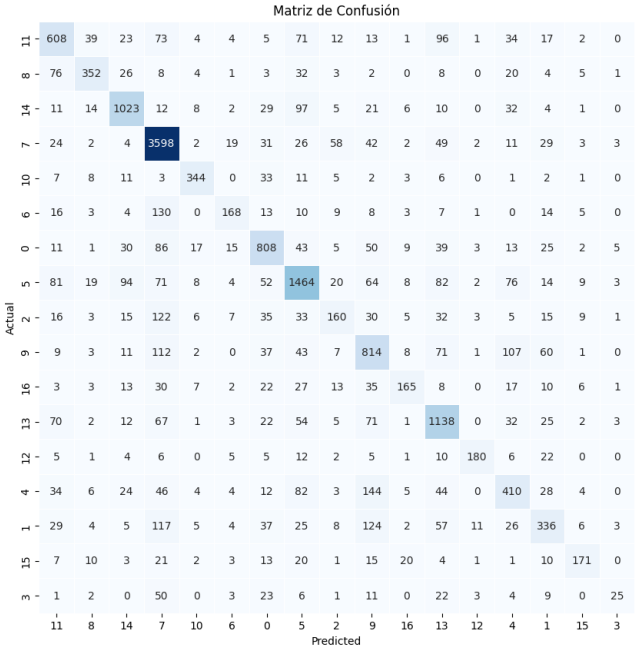
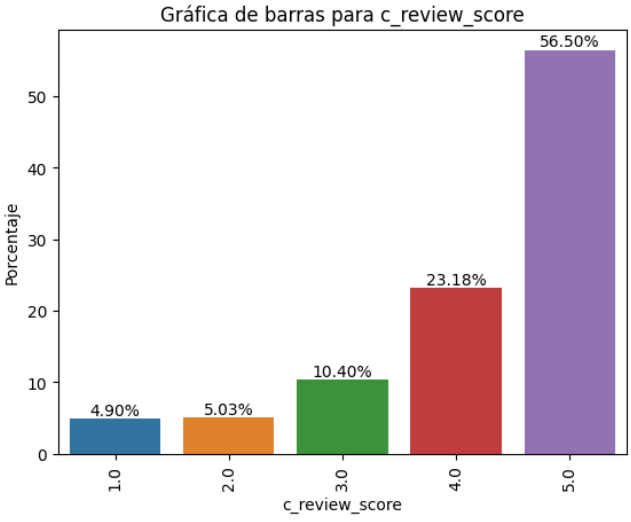


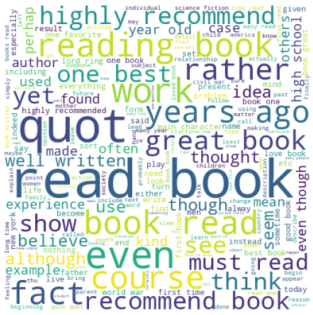
Figura 11: Matriz de confusión para el modelo 7

5. Predecir Score

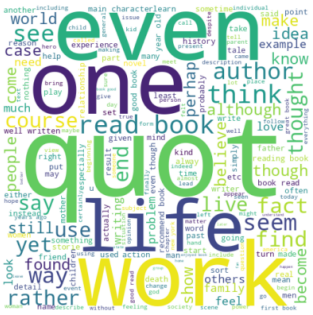
El objetivo en este modelo es predecir el score dado, basado en la reseña que dio el usuario, hay que recordar que nuestro modelo objetivo es un sistema de recomendación, el objetivo de este modelo es tener una representación numérica de la reseña que un cliente da a un libro, se tiene el score como representación numérica pero es muy simple y la reseña escrita puede aportar más información, entonces el objetivo es entrenar una red neuronal que tome como entrada el texto de la reseña y prediga el score que el usuario le dio al libro(de 1 a 5), después tomar la salida de ultima capa oculta de la red neuronal como nuestra representación numérica, a esto se le conoce a esto como embbeding, generalmente se utilizan los embbedings de palabras pero ahora queremos un embbeding de toda la frase, solo tomamos en cuenta los registros con información de score y review del usuario, se podrían tomar las otras variables o calcular nuevas, por ejemplo análisis de sentimiento, pero el objetivo es darle una representación al review y esto mejoraría el modelo pero ensuciaría nuestra representación del texto, hagamos una revisión rápida del EDA, en la siguiente gráfica se observa como se distribuye la target, se observa que la target no es balanceada, de nuevo, no haremos over-sampling ni sub-sampling.



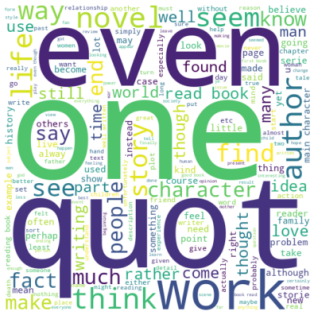
Vamos a gráficar las nubes de palabras para cada categoría.



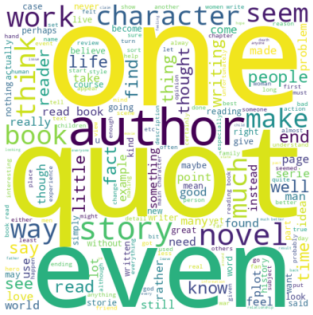
(a) Score de 5



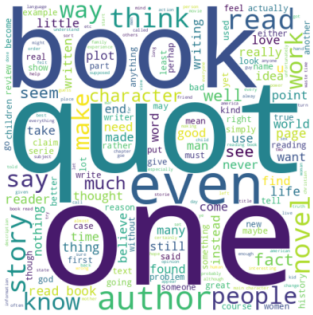
(b) Score de 4



(c) Score de 3



(d) Score de 2



(e) Score de 1

Figura 12: Nube de palabras para los diferentes score

Para este caso no se ve una diferencia significativa , vamos a seguir con los modelos.

5.1. Modelo 1 y 2

En estos modelos al igual que en la red neuronal anterior se tokenizaron las entradas de texto para que fueran secuencia de texto, los hiper-parametros fueron

- Tamaño del vocabulario: 10000
- Tamaño de la secuencia: 200

Después para el Modelo 1 la red neuronal tuvo la siguiente arquitectura:

- **Capa de entrada**: la review tokenizada
- **Capa embedding:** Capa embedding de tamaño 128
- **Capa Oculta:** Capa oculta de 64 unidades con función de activación Relu
- **Capa de Salida:** Capa de salida con 5 unidades y función de activación softmax.
- **Optimizador:** Adam.

Para el modelo 2 la arquitectura fue la siguiente:

- **Capa de entrada**: la review tokenizada
- **Capa embedding:** Capa embedding de tamaño 128
- **Capa Oculta:** Capa oculta de 64 unidades con función de activación Relu

- **Capa Oculta:** Capa oculta de 16 unidades con función de activación Relu
- **Capa de Salida:** Capa de salida con 5 unidades y función de activación softmax.
- **Optimizador:** Adam.

Las métricas fueron las siguientes:

Cuadro 10: Resultados de entrenamiento y prueba modelo 1

Métrica	**Train**	**Test**
Accuracy	0.957845	0.704225
Precision	0.945615	0.606871
Recall	0.941918	0.586768
F1-Score	0.943655	0.595892

Cuadro 11: Resultados de entrenamiento y prueba modelo 2

Métrica	**Train**	**Test**
Accuracy	0.980261	0.709299
Precision	0.970823	0.619608
Recall	0.970306	0.595535
F1-Score	0.970557	0.606791

Ambos modelo tienen métricas similares y presentan sobreajuste, en el siguiente modelo se intento mitigar el sibreajuste

5.2. Modelo 3

Para el tokenizer se utilizaron los siguientes parámetros:

- Tamaño del vocabulario: 200000
- Tamaño de la secuencia: 300

La arquitectura del modelo fue la siguiente:

- **Capa de entrada**: la review tokenizada
- **Capa embedding:** Capa embedding de tamaño 300, esta vez se descargi un modelo pre-entrenado word2vec se tomaron los embeddings que este modelo proporciona
- **Capa Oculta:** Capa oculta de 64 unidades con función de activación Relu
- **Capa Oculta:** Capa droput de 0.4 para el sobreajuste
- **Capa de Salida:** Capa de salida con 5 unidades y función de activación softmax.
- **Optimizador:** Adam con learning rate de 0.00002

El resultado fue el siguiente:
El sobreajuste se mitigo solo un poco, las metricas del test igual mejoraron un poco pero no es un cambio significativo, en la matriz de confusión nos dimos cuenta que el modelo tenia problemas al diferenciar entre score cercanos, por ejemplo entre 4 y 5, o entre 1 y 2, esto puede deberse a que no hay una regla estricta para calificar al libro, hay personas que tienen su forma

Cuadro 12: Resultados de entrenamiento y prueba

Métrica	Train	Test
Accuracy	0.970261	0.723279
Precision	0.961823	0.645136
Recall	0.930306	0.601242
F1-Score	0.940557	0.620523

de calificar y otras tiene una diferente, por ejemplo para un usuario seria ideal calificar con 5 solo los libros que para el son perfectos y con 4 libros que solo le gusten pero que no sean perfectos (para el) , para otro usuario podría ser suficiente que un libro le guste para calificarlo como 5, ambos podrían tener una opinión igual a un libro y calificarlos diferentes, en el siguiente modelo se normaliza para intentar mitigar esto

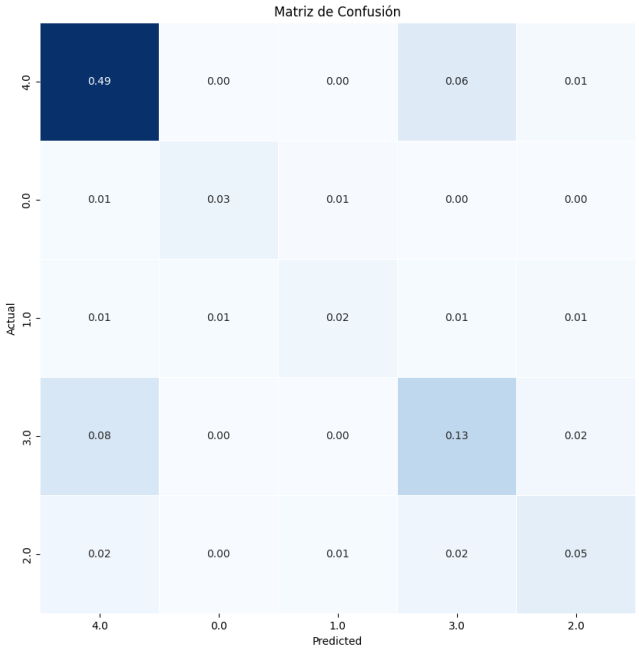


Figura 13: Matriz de confusión para el modelo 3

5.3. Modelo 4 (Mejor Modelo)

Se normalizo el score de la siguiente forma para minimizar el problema de distinguir entre score cercanos, que fue el problema que vimos en el modelo anterior, se normalizo de la siguiente forma:

- 1 y 2 como 0 (no le gusto)
- 3 como 1 (más o menos)
- 4 y 5 como 2 (le gusto)

Los parámetros fueron los mismos que en el modelo anterior solo cambio la capa de salida, fue de 3 en lugar de 5 el tamaño de la capa de salida, los resultados fueron los siguientes.

Cuadro 13: Resultados de entrenamiento y prueba

Métrica	**Train**	**Test**
Accuracy	0.994074	0.876361
Precision	0.991548	0.746756
Recall	0.984016	0.709525
F1-Score	0.987753	0.726846

Se observa todavía un poco de sobreajuste pero las métricas aumentaron considerablemente, por lo tanto tomamos a este como nuestro mejor modelo, a continuación se observa la matriz de confusión en donde confirmamos que el resultado es mejor que el modelo anterior.

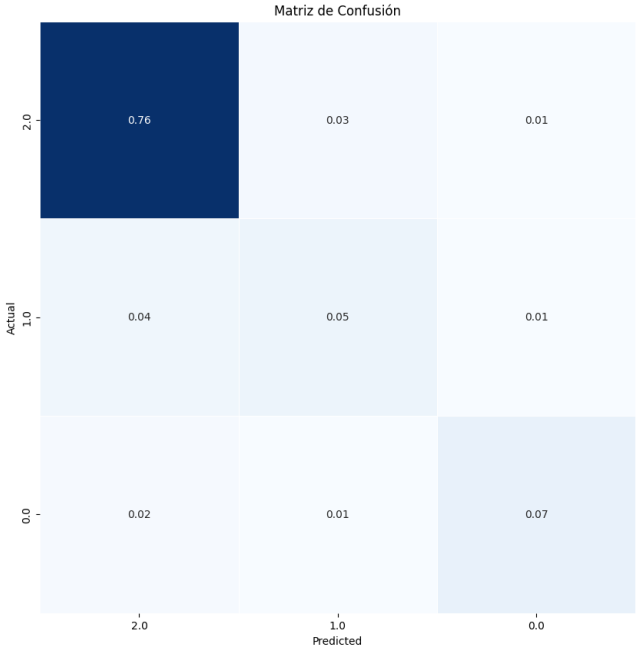


Figura 14: Matriz de confusión para el modelo 4

6. Pasos a seguir

Los modelos obtenidos son regulares, en ambos modelos el desafío principal es el procesamiento de texto, en este modulo no se vio muchas herramientas sofisticadas para el procesamiento de texto, en módulos posteriores se realizará una actualización de los modelos usando técnicas de procesamiento de lenguaje natural más avanzada, una vez obtenidos los mejores modelos se pasará a realizar el modelo de sistema de recomendación, que es el objetivo final del proyecto,