Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Querétaro

Profesor:
Benjamín Valdés Aguirre

Tc3002B.301

Desarrollo de aplicaciones avanzadas de ciencias computacionales

Book Recommendation Algorithm

Presenta

Olimpia Helena García Huerta

Querétaro, Querétaro 02 de Junio de 2025

Índice

Book Recommendation Algorithm

Resumen:

En este estudio se exploran diversas técnicas de aprendizaje automático para construir un sistema de recomendación de libros, utilizando metadatos y calificaciones de usuarios provenientes de la plataforma Goodreads. Inicialmente, se implementaron dos enfoques de clasificación: Árbol de Decisión y K-Nearest Neighbors (KNN), evaluando su efectividad para predecir categorías de calificación derivadas de puntajes promedio. El preprocesamiento de datos incluyó codificación multi-hot para variables categóricas como géneros y autores, así como normalización Z-score para características numéricas como número de páginas y cantidad de reseñas. La variable objetivo se discretizó en cuatro clases de calificación (de "Malo" a "Excelente"), habilitando la clasificación categórica.

Posteriormente, se adoptó un enfoque de regresión mediante redes neuronales artificiales (ANN) para predecir directamente el puntaje promedio. Se entrenaron dos modelos con distintos conjuntos de características, pero con una arquitectura común (64-32-1 neuronas y activación ReLU). A pesar del entrenamiento y validación exhaustivos, los modelos neuronales mostraron bajo rendimiento, con R^2 negativo y MAE superior a 2.0, indicando que fueron menos efectivos que predictores simples

Introducción:

Goodreads es una red social para lectores, que busca ayudarlos a descubrir más libros, mantener el progreso de sus lecturas y obtener recomendaciones.[2]

Sin embargo, como lector quieres obtener una recomendación basada en un libro que te encantó o en varios que te recomendaron, y ahí

reside nuestra problemática a resolver, crear un sistema de recomendación de libros basado en libros previamente leídos.

Para esto, se han empleado distintos métodos para lograr proporcionar recomendaciones relevantes[6], dos de ellos son KNN (K-Nearest Neighbor) un algoritmo basado en instancias que clasifica o realiza recomendaciones en función de la similitud entre elementos. Para un nuevo dato, el algoritmo identifica los K vecinos más cercanos dentro del conjunto de entrenamiento utilizando métricas de distancia (en nuestro caso la euclidiana). La decisión final se toma considerando la mayoría de clases o preferencias entre esos vecinos. [7]

Por otro lado, el Decision Tree Classifier es un modelo predictivo estructurado como un árbol, donde cada nodo interno representa una prueba sobre un atributo, cada rama una salida de la prueba, y cada hoja una etiqueta de clase o decisión final. El árbol se construye a partir de datos históricos, aprendiendo reglas de decisión que permiten clasificar nuevos ejemplos de manera eficiente.[8]

Estos son los dos algoritmos con los que empezamos este proyecto.

Metodología

A continuación, se describe el procedimiento bajo el cual se realizó el proyecto. Primero, se procedió con la investigación y recolección de datos en Kaggle, un repositorio online.

DataSet:

Este conjunto de datos ofrece una colección completa de información proveniente de GoodReads, una plataforma popular para que los lectores descubran y reseñen libros. Incluye información sobre libros, detalles de los libros y reseñas de usuarios, ofreciendo perspectivas sobre el mundo literario. [1]

Columnas:[1]

1. book_id: Un identificador único para cada libro en el conjunto de datos.
2. cover_image_uri: La URI (Identificador Uniforme de Recursos) o URL que apunta a la imagen de portada del libro.
3. book_title: El título del libro.
4. book_details: Detalles adicionales sobre el libro, como un resumen, sinopsis del argumento u otra información descriptiva.
5. format: El formato del libro, como tapa dura, libro de bolsillo, libro electrónico, audiolibro, etc.
6. publication_info: Información sobre la publicación del libro, incluyendo la editorial, la fecha de publicación y cualquier otro dato relevante.
7. authorlink: La URI o URL que dirige a más información sobre el autor, si está disponible.
8. author: El nombre del autor o los autores del libro.
9. num_pages: La cantidad de páginas que tiene el libro.
10. genres: El género al que pertenece el libro, puede ser uno o varios

División del DataSet

Se dividió en 80% para datos de entrenamiento, que a su vez se dividió en 80% para train y 20% para validación. El 20% restante del dataset fue utilizado para test

Encoding:

Género:

Se utilizó multi-hot encoding:

Cuando una fila puede tener más de una clase al mismo tiempo (como múltiples géneros por libro)[3], se usa multi-hot encoding. Cada género se convierte entonces en una columna binaria (1 si el libro pertenece a ese género, 0 si no)

1. Limpieza de datos (clean_data):
   Se utilizó _ast.literal_eval_ para convertir las cadenas de texto que representaban listas en listas reales de Python. Luego, se eliminaron los espacios adicionales al inicio y al final de cada género, asegurando así una limpieza adecuada. Todos los géneros únicos fueron almacenados en un conjunto (set), lo que garantizó que no hubiera repeticiones.

2. Ordenamiento de géneros:
   Una vez identificados los géneros únicos, estos se ordenaron alfabéticamente. Aunque no es un paso obligatorio, esta organización se decidió para facilitar la estructura del dataset y su posterior análisis.

3. Codificación (hot_encoder):
   Para cada género único, se creó una nueva columna en el DataFrame con el mismo nombre del género. En cada una de estas columnas, se asignó un 1 si ese género estaba presente en la lista de géneros del libro correspondiente, o un 0 si no lo estaba. Este enfoque permitió representar múltiples géneros para un mismo libro de forma binaria.

4. Eliminación de la columna original genres:
   Finalmente, se eliminó la columna original genres, ya que toda su información había sido distribuida en las nuevas columnas binarias, una por cada género.

Autor:

Se utilizó multi-hot encoding para los autores:

1. Limpieza de datos (clean_authors):
   Se creó una función llamada clean_authors para normalizar los

nombres de los autores. Esta función realiza dos tareas principales:

   - Eliminación de espacios innecesarios: Se usa .strip() para quitar los espacios al inicio y al final de cada nombre de autor.
   - Manejo de errores: Si un campo de autor está vacío o es NaN, se reemplaza con la etiqueta 'Unknown'.

2. Todos los autores únicos se almacenan en un conjunto (set) para evitar repeticiones y se devuelven como una lista.
3. Ordenamiento de autores:
   Una vez obtenida la lista de autores únicos, se ordenan alfabéticamente usando sort(). Esto se realiza para mantener una estructura clara y ordenada del dataset, especialmente útil para análisis posteriores o visualización.
4. Codificación (author_hot_encoder):
   Se creó una nueva columna binaria por cada autor único. La función author_hot_encoder evalúa si el nombre del autor en cada fila coincide exactamente con el autor de la columna correspondiente:
   - Si coincide, se asigna un 1.
   - Si no coincide se asigna un 0.
5. Eliminación de la columna original author:
   Tras la codificación, la columna original author se elimina , ya que su información ahora se encuentra distribuida en las nuevas columnas binarias.

Escalamiento:

Se utilizó normalización Z-Score:

También conocida como estandarización. Este método transforma cada valor numérico restando la media de su columna y dividiendo entre su desviación estándar. Como resultado, cada característica numérica queda centrada en una media de 0 y una desviación estándar de 1. Esto ayuda a evitar que las variables con escalas mayores dominen el modelo.[4]

1. Identificación de columnas numéricas
   Se analizaron los tipos de datos del Data Frame y se seleccionaron todas las columnas numéricas. Las columnas no numéricas se excluyeron del proceso de escalado para evitar errores

2. Se estandarizaron los datos
   Se usó la librería de SkLearn de Standard Scaler para realizar la estandarización de los datos

3. Reconstrucción del DataFrame
   Una vez estandarizadas las columnas numéricas, se reconstruyó el Data Frame combinando estos datos con las columnas no numéricas originales, como los títulos de libros, autores y descripciones. Esto permite conservar toda la información.

Esta fue elegida porque se planea usar un modelo de clasificación, aunque aún no se ha definido cuál. Dado que muchos algoritmos de clasificación son sensibles a la escala (por ejemplo, SVM, KNN, regresión logística), la estandarización asegura que todas las características contribuyan de manera equitativa al proceso de aprendizaje.[5]

Target:

Posteriormente nos preguntamos qué tan buenos son los libros que se están recomendando.

Por lo que se procedió a crear la clase objetivo Rating_class la cual se muestra a continuación:

Rango rating	Clase	Descripción
<= 2.5	0	Malo
>2.5 y <=2.9	1	Regular
>3 y <=4.0	2	Bueno
>4.0	3	Excelente

Esto con el fin de predecir en base al Rango del Rating que tan Excelente o Malo es un libro.

Para esto tuvimos que desescalar la columna average_rating y convertirla en la original. Con estos datos creamos la columna rating class y la llenamos con las clases de las distintas categorías usando cut para seleccionar los intervalos, bins son los rangos y labels son las clases que van a quedar en la columna,

Se convirtió la columna rating_class a int y se verificó la distribución de las clases, dando como resultado un DataSet algo desbalanceado:

rating_class	Cantidad
0	33
1	669
2	5348
3	6930

KNN

Se usó KNN para recomendar libros a partir de un ID dado, esto posteriormente se cambiará para que lo que se ingrese sea un título y se busque el ID asociado al mismo.

Metodología:

Se recibe el ID del libro, el DF escalado, el DF que contiene los títulos y descripción indexados por el ID del DF numérico y se

calculan los 10 vecinos más cercanos, una vez realizado esto, se buscan los ID's en el DF de texto y se despliegan en pantalla.

Implementación:

Decision Tree:

El primer modelo implementado siguiendo la metodología descrita por Duhan y Arunachalam[6] fue un árbol de decisión.

Metodología:

Se desarrolló una función denominada entrenar_decision_tree para construir y entrenar el árbol utilizando la librería sklearn.tree.DecisionTreeClassifier. Esta función recibe como entrada los conjuntos de datos de entrenamiento y prueba, así como las etiquetas correspondientes (X_train, X_test, y_train, y_test), y permite ajustar la profundidad máxima del árbol (max_depth), que por defecto se establece en 10. El modelo utiliza el criterio de entropía, como se menciona en el paper[6], para evaluar la calidad de las divisiones internas del árbol. Este enfoque sigue una estrategia de aprendizaje codiciosa (greedy), maximizando la ganancia de información en cada nodo.

Esto se hizo ya que los clasificadores basados en árboles de decisión destacan en sistemas de recomendación por su:

- Alto rendimiento en clasificación supervisada.
- Capacidad de manejar datos categóricos y numéricos.
- Velocidad de entrenamiento y predicción.

KNN Classification:

El segundo enfoque implementado para la clasificación de preferencias de usuario fue el algoritmo K-Nearest Neighbors (KNN) de clasificación.

Metodología:

Se desarrolló una función llamada entrenar_knn que construye un modelo KNeighborsClassifier de la biblioteca sklearn.neighbors. La función recibe como parámetros los datos de entrenamiento (X_train, y_train) y el número de vecinos n_neighbors, que por defecto se define en 10. En este estudio se entrenaron dos modelos con diferentes subconjuntos de datos, utilizando 5 vecinos (n_neighbors=5) para reflejar una clasificación basada en similitud más localizada. Esto se alinea con lo propuesto por Duhan y Arunachalam [6], quienes aplican el algoritmo KNN en su sistema de recomendación para calcular similitud entre libros.

Su uso se justifica por su simplicidad y efectividad en entornos donde las relaciones entre instancias pueden modelarse mediante medidas de similitud como la distancia Euclídeana. En su implementación, el algoritmo fue capaz de alcanzar una precisión de validación del 88.85% con una puntuación F1 de 0.9991

Resultados:

Para evaluar el desempeño de los modelos de clasificación aplicados (Decision Tree y K-Nearest Neighbors), se utilizaron tres métricas: accuracy, F1-score y MAE (error absoluto medio). Se compararon dos variantes por modelo:

- **Modelos "1":** entrenados sin las columnas average_rating y rating_class, ya que esta última es la variable objetivo.
- **Modelos "2":** entrenados eliminando además otras columnas relacionadas con calificación, para analizar su impacto predictivo.

A continuación, se presenta un resumen de las métricas obtenidas:

Modelo	Accuracy (val)	F1-Score (val)	MAE (val)	Accuracy (test)	F1-Score (test)	MAE (test)

Decision Tree 1	0.9106	0.9106	0.0897	0.8934	0.8938	0.1079
Decision Tree 2	0.5135	0.5384	0.5689	0.5600	0.5809	0.5140
KNN 1	0.5547	0.5357	0.4507	0.5516	0.5315	0.4561
KNN 2	0.5901	0.5795	0.4191	0.7564	0.7457	0.2517

Las conclusiones que podemos obtener de estos resultados son:

- El mejor modelo general fue Decision Tree 1, con una accuracy superior al 89% en test, y el menor error (MAE ≈ 0.11).
- El modelo Decision Tree 2, al eliminar las variables relacionadas con calificación, muestra una caída sustancial en todas las métricas.
- En KNN, el modelo KNN 2 (sin rating) supera a KNN 1 en test, pero no alcanza el rendimiento del árbol de decisión.

El impacto de eliminar las variables de rating en el modelo 2 fue mayor en los árboles de decisión que en KNN, reflejando su importancia como características predictivas, o la posibilidad de un sobreajuste.

En comparación con el trabajo de Duhan y Arunachalam[6], su Decision Tree Classifier alcanzó una precisión de validación del 99.05% y un F1-score de 1.0, en un entorno experimental más controlado y posiblemente con clases más balanceadas. En nuestro caso, aunque las métricas son mucho menores, los resultados del modelo Decision Tree 1 siguen siendo muy competitivos, especialmente considerando que se trabajó con un conjunto de datos

modificado y sin aplicar técnicas avanzadas de balanceo o ajuste de hiper parámetros.

Asimismo, el paper destaca que KNN logró un F1-score de 0.9991, mientras que nuestros modelos oscilaron entre 0.5315 y 0.7457, lo que sugiere que el desempeño del algoritmo KNN puede ser altamente sensible a la calidad y estructura del conjunto de datos, así como a la normalización y selección de vecinos.

Nuevo planteamiento del problema:

Ahora, nos enfrentamos al problema de realmente predecir el rating, ya no clasificarlo. Por lo que se usará una red neuronal de regresión para saber que tan bueno es un libro según su autor, número de páginas, género, número de ratings y número de reviews. Para esto, se usó el mismo Dataset previamente pre-procesado.

Red Neuronal:

Se implementó una Red Neuronal de regresión, siguiendo una arquitectura estándar basada en capas densas. Este modelo se empleó para evaluar su rendimiento predictivo frente a otras técnicas más tradicionales, como los árboles de decisión.

Metodología:

Se desarrollaron dos modelos de red neuronal utilizando la API Sequential de Keras. Ambos modelos comparten una arquitectura similar pero fueron entrenados con distintos conjuntos de datos (X_train, X1_train, etc.).

La arquitectura empleada en ambos modelos consiste en:

- Una capa de entrada con 64 neuronas y función de activación ReLU.
- Una capa oculta intermedia con 32 neuronas, también con activación ReLU.

- Una capa de salida con una sola neurona para problemas de regresión

Ambos modelos se compilaron utilizando el optimizador Adam, y se utilizó el error cuadrático medio (MSE) como función de pérdida, apropiada para problemas de regresión.

Para asegurar que se guardará el mejor modelo durante el entrenamiento, se incorporó un *callback* ModelCheckpoint que monitoriza la pérdida de validación (val_loss). El mejor modelo (con menor pérdida de validación) se guarda en un archivo .keras.

Ambos modelos se entrenaron durante 150 épocas y un batch de 32, posteriormente los modelos se guardaron con model.save() y se cargaron de nuevo con load_model(), lo que permite reutilizar los modelos entrenados sin volver a correrlos.

Su uso se justifica, ya que las redes neuronales son altamente flexibles para capturar relaciones no lineales complejas entre variables y se adaptan bien a problemas de regresión con múltiples variables de entrada.[9]

Resultados:

Los resultados evidencian un rendimiento significativamente inferior en los modelos desarrollados en este trabajo (NN1 y NN2) en comparación con el modelo de YuMin et al.[10] quienes desarrollaron un enfoque de predicción de calificaciones de películas utilizando una arquitectura de red neuronal alimentada por métricas compuestas y enriquecidas.

Los valores negativos del coeficiente de determinación (R^2) sugieren que mi modelo es menos preciso que una predicción constante basada en la media de los datos.

Por el contrario, el modelo del Estado del Arte reporta una precisión significativamente mayor, con un MAE estimado inferior a 0.75 y resultados validados a través de una matriz de confusión.

Conclusiones:

En esta investigación se exploraron diferentes enfoques para la recomendación y clasificación de libros, comenzando con métodos tradicionales como árboles de decisión y KNN. Continuando con la implementación de una red neuronal orientada a regresión. A lo largo del proyecto, se aplicaron diversas técnicas de preprocesamiento, incluyendo multi-hot encoder, normalización Z-score y transformación del target en clases discretas, lo cual permitió estructurar los datos de manera adecuada para su uso en modelos de aprendizaje supervisado.

Los resultados muestran que el mejor desempeño en tareas de clasificación fue alcanzado por el modelo **Decision Tree 1**, con una precisión del 89.34% en el conjunto de prueba y un MAE de apenas 0.1079, superando tanto a los modelos KNN como a su versión reducida (Decision Tree 2). Este resultado confirma la capacidad de los árboles de decisión para capturar relaciones estructuradas y jerárquicas en los datos.

En contraste, los modelos de red neuronal empleados para regresión mostraron un desempeño considerablemente inferior. Con valores de R^2 negativos y errores absolutos medios superiores a 2.0, se concluye que la arquitectura utilizada no fue capaz de capturar adecuadamente las relaciones entre las variables predictoras y la calificación promedio de los libros. Este resultado sugiere que la calidad del preprocesamiento y la ingeniería de características tiene un impacto crucial en el rendimiento de modelos de redes neuronales, incluso más que la complejidad arquitectónica en sí.

Referencias:

[1] Grimm, “Books_Dataset_GoodReads(May 2024),” *Kaggle.com*, 2024. https://www.kaggle.com/datasets/dk123891/books-dataset-goodreadsmay-2024?select=Book_Details.csv (accessed May 19, 2025).

[2] “About Goodreads,” *Goodreads.com*, 2025. https://www.goodreads.com/about/us (accessed May 19, 2025).

[3]GeeksforGeeks, “An introduction to MultiLabel classification,” *GeeksforGeeks*, Jul. 15, 2020. https://www.geeksforgeeks.org/an-introduction-to-multilabel-classification/ (accessed May 19, 2025).

[4] GeeksforGeeks, “ZScore Normalization: Definition and Examples,” *GeeksforGeeks*, Jul. 26, 2024. https://www.geeksforgeeks.org/z-score-normalization-definition-and-examples/ (accessed May 19, 2025).

[5] I. Belcic, “Classification in Machine Learning,” *Ibm.com*, Oct. 15, 2024. https://www.ibm.com/think/topics/classification-machine-learning (accessed May 19, 2025).

[6] Easy Chair, “Book Recommendation System Using Machine Learning,” *Easychair.org*, 2023. https://easychair.org/publications/preprint/2V4r? (accessed May 26, 2025).

[7] IBM, “KNN,” *Ibm.com*, Oct. 04, 2021. https://www.ibm.com/mx-es/think/topics/knn (accessed Jun. 02, 2025).

[8] IBM, “Decision trees,” *Ibm.com*, Nov. 02, 2021. https://www.ibm.com/mx-es/think/topics/decision-trees (accessed Jun. 02, 2025).

[9] IBM, “Red neuronal,” *Ibm.com*, Oct. 06, 2021. https://www.ibm.com/mx-es/think/topics/neural-networks (accessed Jun. 02, 2025).

[10] S. YuMin, Z. Yuan, and Y. JinYao, “Neural Network Based Movie Rating Prediction,” *Proceedings of the 3rd International*

Conference on Big Data and Computing, vol. 18, no. ICBDC ’18, pp. 33–37, Apr. 2018, doi: https://doi.org/10.1145/3220199.3220204.