Análisis y Reporte sobre el desempeño del modelo - Portafolio de Análisis

Ana Daniela López Dávila | A00831568

Introducción

El uso de algoritmos de ML se ha vuelto cada vez más accesible y eficiente gracias al uso de frameworks y bibliotecas especializadas. Estas herramientas ofrecen una gran variedad de algoritmos de ML, como redes neuronales, árboles de decisión, regresión lineal, entre otros, que pueden ser implementados y personalizados. Una de las aplicaciones de estos algoritmos se encuentran en los sistemas de recomendación.

Un sistema de recomendación tiene como objetivo proporcionar a los usuarios recomendaciones personalizadas sobre productos, servicios o contenido que puedan interesarles. Estos sistemas utilizan técnicas avanzadas para analizar el comportamiento del usuario, la información del producto y otros datos relevantes para generar recomendaciones precisas.

En este proyecto, se planea construir un sistema de recomendación de películas personalizado utilizando un modelo predictivo (Random Forest) y un conjunto de datos de MovieLens. En este reporte se describe paso a paso el proceso de construcción del sistema, incluyendo el desarrollo y mejoramiento del modelo utilizado.

Datos utilizados

Los conjuntos de datos de MovieLens fueron recopilados por el Proyecto de Investigación GroupLens en la Universidad de Minnesota. Estos datos proporcionan una valiosa fuente de información sobre calificaciones (en una escala de 1-5) de películas y preferencias de los usuarios. Cada usuario ha calificado al menos 20 películas, y además, se incluye información demográfica básica de los usuarios, que abarca datos como la edad, género, ocupación y código postal.

*Los datos se encuentran en https://grouplens.org/datasets/movielens/100k/

Descripción de los conjunto de datos

- *u_data*

Contiene el conjunto de datos completo, con 100,000 calificaciones otorgadas. Cada calificación se compone de un usuario, una película, una calificación (en una escala de 1-5) y una marca de tiempo en formato de segundos Unix desde el 1 de enero de 1970.

Columnas: user id, item id, rating, timestamp
Cantidad de datos: 100,000 registros

- *u_genre*

Proporciona una lista de géneros cinematográficos utilizados en las películas del conjunto de datos.

Columna: genres
Cantidad de datos: 19 géneros

- *u_item*

Contiene información detallada sobre las películas, incluyendo su título, fecha de lanzamiento, URL de IMDb y géneros. Los géneros se representan como una serie de campos binarios donde 1 indica que la película pertenece a un género específico.

Columnas: movie id, movie title, release date, video release date, IMDb URL, unknown, Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western
Cantidad de datos: 1,682 registros de películas

- *u_info*

Enumera el número de usuarios, películas y calificaciones presentes en el conjunto de datos.

- *u_user*

Contiene información demográfica sobre los usuarios, incluyendo su edad, género, ocupación y código postal.

Columnas: user id, age, gender, occupation, zip code
Cantidad de datos: 943 usuarios

Importar y combinar datos

Se comenzó importando cada conjunto de datos utilizando la función *read_csv* de Pandas y especificando las columnas apropiadas. Una vez que se importaron los datos, se combinan

para obtener un conjunto de datos completo y relevante para nuestro proyecto. Utilizamos las funciones de fusión *merge* y manipulación de datos de Pandas para realizar esta operación. El resultado final es un DataFrame que contiene la siguiente información:

```
----------* Datos combinados con éxito

       user_id  movie_id  age gender genre_name  rating
0          196       242   49      M     Comedy       3
1          196       393   49      M     Comedy       4
2          196       381   49      M     Comedy       4
3          196       251   49      M     Comedy       3
4          196       655   49      M  Adventure       5
...        ...       ...  ...    ...        ...     ...
99995      941       919   20      M  Adventure       5
99996      941       273   20      M     Action       3
99997      941         1   20      M  Animation       5
99998      941       294   20      M     Comedy       4
99999      941      1007   20      M     Comedy       4

[100000 rows x 6 columns]

 Estos son los datos que utilizaremos para entrenar nuestros modelos.
```

Dividir datos (Entrenamiento y validación)

Para poder evaluar el rendimiento de los modelos a probar, es fundamental dividir nuestro conjunto de datos combinado. Como primer paso, se definió las características X y y. Nuestra variable objetivo es el *rating* que el usuario le da a las películas, la demás información es utilizada como características o *features* en los modelos.

Para llevar a cabo la división de los datos en conjuntos de entrenamiento y prueba, utilizamos la función *train_test_split* de la biblioteca Scikit-Learn. Definimos que el 20% de nuestros datos se utilizará como conjunto de prueba, mientras que el 80% restante se utilizará como conjunto de entrenamiento. También especificamos una semilla aleatoria, *random_state,* para garantizar la reproducibilidad de nuestros resultados.

Debido que los modelos a probar requieren que todas las características sean numéricas, codificamos las características categóricas 'gender' y 'genre_name' utilizando la codificación one-hot. Esto se logra mediante la función get_dummies de Pandas, que crea columnas binarias para cada categoría posible.

Entrenar modelos

Árbol de decisión

El primer modelo desarrollado fue un Árbol de Decisión con una profundidad máxima de 5.

```
Evaluación del primer modelo: Árbol de decisión

Error Cuadrático Medio (MSE): 1.159606552218771
Raíz del Error Cuadrático Medio (RMSE): 1.0768502923892305
Coeficiente de Determinación (R2): 0.07900849501151119
```

Los resultados sugieren un rendimiento moderado. El MSE indica que las predicciones tienen una desviación significativa respecto a los valores reales. El RMSE indica que, en promedio, las predicciones tienen un error alrededor de 1.08 unidades según la escala de calificación. Y, por último, el Coeficiente de Determinación es menor al 10%. Esto sugiere que el modelo no está capturando bien la variabilidad de los datos y, por ende, su capacidad de predecir es limitada.

Random Forest

El segundo modelo fue un Bosque Aleatorio, Random Forest, con 300 estimadores.

```
Evaluación del segundo modelo: Random Forest

Error Cuadrático Medio (MSE): 1.0617706355555554
Raíz del Error Cuadrático Medio (RMSE): 1.0304225519443737
Coeficiente de Determinación (R2): 0.15671247827822876
```

Se observa una mejora en comparación con el Árbol de Decisión. El MSE disminuyó, sin embargo aún sigue siendo un tanto significativo. el RMSE también se redujo, indicando que las predicciones mantienen un error alrededor de 1.03 unidades, en promedio. El Coeficiente de Determinación aumentó a más de 15%, lo que indica una mejor capacidad del modelo para explicar la variabilidad en los datos.

Red Neuronal

El tercer modelo fue una Red Neuronal, la cual utilizó los datos en una misma escala para aumentar su rendimiento. Para lograr esto, se utiliza *StandardScaler* para estandarizar las características, lo que significa que se centran alrededor de cero y tienen una desviación estándar de uno.

Posterior a esto se definió la arquitectura o estructura de la Red Neuronal. La red neuronal consta de una capa de entrada y una capa de salida. Primero se tiene una capa oculta con 16 unidades y función de activación ReLU, Rectified Linear Unit, como primera capa oculta. Luego, para predecir el valor continuo de la calificación de la película, se agrega una capa de salida con una sola unidad y activación lineal. La activación lineal permite que la red neuronal genere salidas continuas en lugar de categorías discretas.

```
Model: "my_sequential_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 hiddenlayer1 (Dense)        (None, 16)                368

 outputlayer (Dense)         (None, 1)                 17

=================================================================
Total params: 385
Trainable params: 385
Non-trainable params: 0
_________________________________________________________________
----------* Creación del tercer modelo con éxito
```

Cabe aclarar que este modelo se compila utilizando el optimizador Adam con una tasa de aprendizaje de 0.001 y la función de pérdida de error cuadrático medio (MSE) para minimizar la diferencia entre las predicciones y las calificaciones reales. En cuanto al entrenamiento, se especifica el número de épocas a 100. Durante el entrenamiento, se realiza la validación utilizando el 15% de los datos de entrenamiento para evaluar el rendimiento del modelo en datos no vistos.

```
Evaluación del cuarto modelo: Red Neuronal

625/625 [==============================] - 0s 182us/step
Error Cuadrático Medio (MSE): 1.1790244052980738
Raíz del Error Cuadrático Medio (RMSE): 1.0858289024050123
Coeficiente de Determinación (R2): 0.06358630056380477
```

El tercer modelo, una Red Neuronal sugiere que podría haber margen para mejorar su rendimiento. Sin embargo, dentro de este proyecto se planea trabajar con una Red Neuronal para conocer y explorar el mejoramiento de este tipo de modelos.

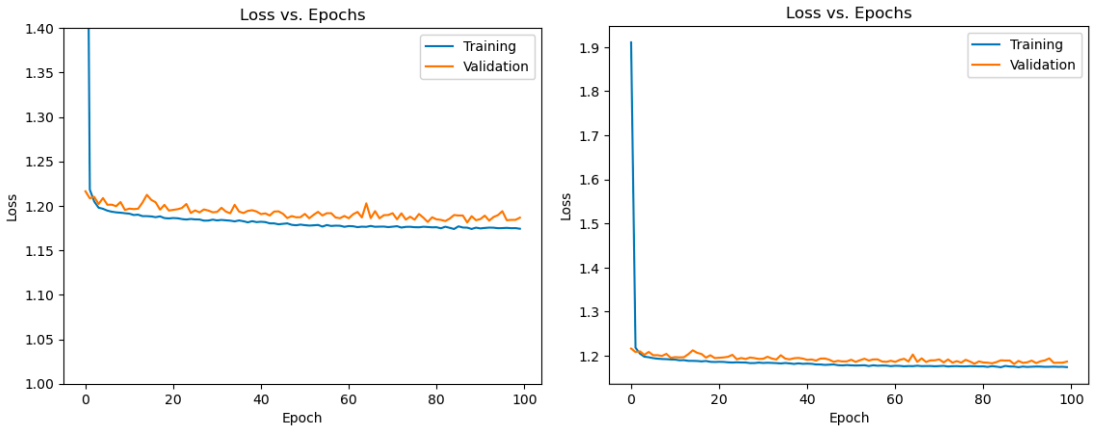
Selección de modelo

La decisión de trabajar con una Red Neuronal fue tomada por su capacidad de abordar la complejidad de los datos con patrones sutiles y el potencial que se le ve al modelo para

mejorar mediante distintas técnicas. Además, se toma en cuenta que con 100,000 registros es suficiente para entrenar una Red Neuronal y poder trabajar con ella.

Análisis del modelo original

Antes de mejorar la Red Neuronal, se realizó un análisis sobre el bias, varianza y el ajuste del modelo. Para esto se graficó la curva de aprendizaje a través de las epochs del entrenamiento.



Observaciones:

Nos damos cuenta que el modelo presenta un comportamiento de **underfitting**:

- Ambas pérdidas disminuyen a lo largo de las epochs, esto nos da una señal positiva acerca del comportamiento del modelo.
- La diferencia entre las pérdidas de entrenamiento y validación es pequeña, lo que indica que el modelo generaliza bastante bien y no está sobreajustando en gran medida a los datos de entrenamiento.
- La pérdida es de valor significativo y esto representa un **sesgo alto** y bajo ajuste de modelo, lo que indica que el modelo es demasiado simple para capturar la complejidad de los datos y es posible que el modelo necesite una mayor capacidad o una arquitectura más compleja para mejorar su rendimiento.

Refinamiento del modelo

Debido al comportamiento de underfitting del modelo, se decidió construir una estructura de red neuronal un poco más compleja para visualizar el comportamiento del modelo con la misma data. La nueva estructura consiste de una capa de entrada con 16 unidades y una función de activación ReLU. Seguido de esta capa, se agregó la técnica de dropout con una tasa del 30% para prevenir el sobreajuste. Entre las capas ocultas se tiene que:

- Las capas ocultas 2 y 3, tienen 32 unidades cada una y utilizan regularización L2 para controlar la complejidad del modelo.
- Las capas ocultas 4, 5, 6 y 7, tienen respectivamente 64, 64, 128 y 128 unidades y utilizan regularización L2.

Asimismo, se utilizo Batch Normalization después de la primera capa oculta para estabilizar y acelerar el entrenamiento. Por último, se tiene la capa de salida con 1 unidad y una función de activación lineal.

```
Model: "my_sequential_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 hiddenlayer1 (Dense)        (None, 16)                368

 dropout_1 (Dropout)         (None, 16)                0

 hiddenlayer2 (Dense)        (None, 32)                544

 hiddenlayer3 (Dense)        (None, 32)                1056

 batch_normalization1 (Batch  (None, 32)               128
 Normalization)

 hiddenlayer4 (Dense)        (None, 64)                2112

 hiddenlayer5 (Dense)        (None, 64)                4160

 hiddenlayer6 (Dense)        (None, 128)               8320

 hiddenlayer7 (Dense)        (None, 128)               16512

 outputlayer (Dense)         (None, 1)                 129
```

Este modelo mejorado utiliza el mismo optimizador Adam con la misma tasa de aprendizaje al momento de compilar el modelo. La diferencia radica en el entrenamiento, pues se especifica un número de 600 epochs y se añade el callback de early stopping, que detendrá el entrenamiento si no se observa una mejora significativa en la métrica de pérdida durante 40 épocas consecutivas en modo de minimización. Durante el entrenamiento se visualiza que el modelo llega hasta la época número 131.

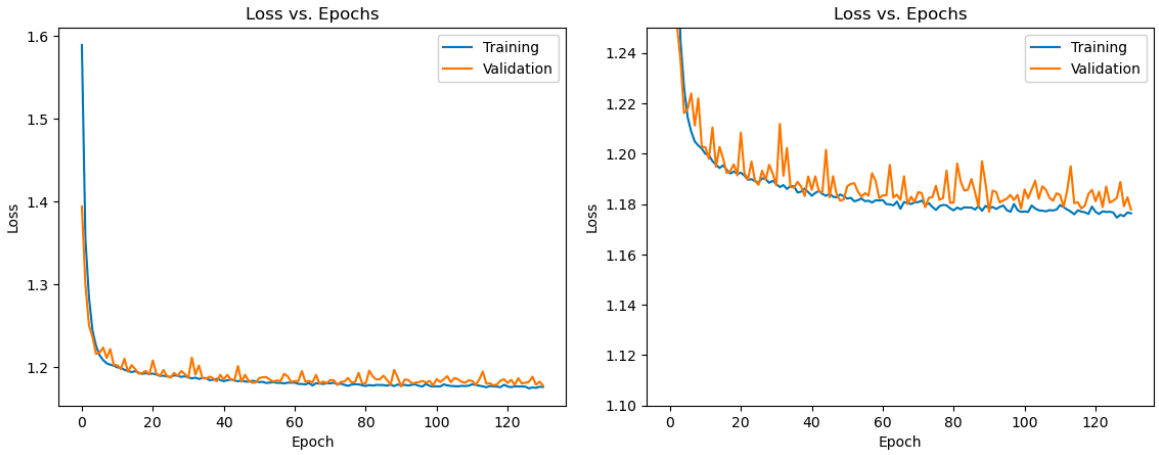
```
Evaluación del modelo: Red Neuronal Mejorado

  1/625 [..............................] - ETA: 6s
625/625 [==============================] - 0s 264us/step
Error Cuadrático Medio (MSE): 1.1628009420225314
Raíz del Error Cuadrático Medio (RMSE): 1.0783324821327285
Coeficiente de Determinación (R2): 0.07647142253011163
```

En cuanto a las métricas asociadas al rendimiento del modelo, se ve una ligera mejora a comparación del modelo de Red Neuronal original.

Análisis del modelo mejorado

Con el fin de comparar ambos modelos, se vuelve a realizar un análisis sobre el bias, varianza y ajuste del modelo mejorado. De igual forma se graficó la curva de aprendizaje a través de las epochs del entrenamiento.



Tomando en consideración las métricas y gráficas de curva de aprendizaje del modelo, se concluye que el modelo tiene una mejora en su ajuste a los datos. El grado de sesgo ha disminuido y la varianza es moderada pues el modelo no esta sobreajustado significativamente a los datos. Asimismo, el coeficiente de determinación nos indica que el modelo si explica una parte de la variabilidad de los datos pero no es la mayoría.

En resumen, el modelo de Red Neuronal Mejorado tiene un **bajo sesgo, una varianza moderada** y se ajusta aceptablemente bien a los datos de entrenamiento. Si bien, todavía hay margen de mejora para futuros proyectos. Por el momento, trabajaremos con el modelo mejorado para construir el sistema de recomendación.

Construcción del sistema de recomendación

Para construir el sistema de recomendación se desarrollaron cuatro funciones que dividen en secciones el funcionamiento del sistema.

1. Función 1. Obtener películas no calificadas por un usuario específico

En esta función se filtran todas las calificaciones realizadas por un usuario específico. De esas calificaciones realizadas, se identifican cuáles fueron los ID de las películas. Con la lista de los ID, se filtra la base de datos de películas para obtener aquellas películas que el usuario específico no ha calificado y, por ende, no ha visto.

Esta función nos permite tener guardadas los títulos, ID y género de cada película no vista por el usuario proporcionado.

2. Función 2. Generar dataset de atributos para realizar las predicciones

En esta función se combinan distintos conjuntos de datos para obtener las características correspondientes a cada registro, película no calificada. Se añade la columna con el ID del usuario, su edad y género. Y por último, se aplica la técnica one hoy encoding para pasar a dummies aquellas variables cualitativas como género de película.

Esta función nos permite tener guardados los atributos que serán utilizados para realizar nuestras predicciones y conocer qué calificación le pondría el usuario a la lista de películas no vistas. Todo con el fin de realizar recomendaciones.

3. Función 3. Realizar cinco recomendaciones por un usuario específico

En esta función se escalan los datos y se predice la calificación que el usuario dará. Asimismo, se obtienen los títulos de las películas para proporcionar una visualización atractiva y clara sobre las recomendaciones. El reordenar según la calificación más alta es lo que nos permite asegurarnos de que las recomendaciones son aquellas películas mejores calificadas por el modelo según los patrones encontrados.

4. Función 4. Sistema de recomendación completo

En esta última función se realiza todo el despliegue de las predicciones y recomendaciones.

Conclusión

El proyecto fue retador puesto que el manejo de datos fue fundamental para el entrenamiento y evaluación del modelo. Asimismo, se probaron distintos modelos para un mismo dataset y se reflexionó sobre sus comportamientos. En lo personal, me gustaría seguir trabajando con proyectos del estilo pues ponen a prueba mis habilidades de programación y mis conocimientos de ML. En las mejoras del sistema añadiría tener una base de datos actualizada puesto que las recomendaciones son películas antiguas.

En conclusión, se logró evaluar el bias, varianza y reconocer el tipo de ajuste del modelo. A su vez conocí una alternativa de sistema de recomendación haciendo uso de modelos supervisados.