PRÁCTICA DE SISTEMAS DE RECOMENDACIÓN

Universidade da Coruña
Adrián García Vázquez
Correo: adrian.gvazquez@udc.es

Introducción

En esta memoria se documenta el desarrollo de la práctica del bloque de sistemas de recomendación en la asignatura de RIWS. La práctica se centra en la implementación de modelos de recomendación utilizando el paquete `scikit-surprise`, abarcando desde la carga y exploración de los datos hasta la aplicación de varios algoritmos y la evaluación de sus resultados.

Paso 1: Descarga y extracción del dataset

Descripción

En este paso, se descarga el dataset de la URL proporcionada, se guarda localmente y se descomprime para obtener el archivo `ratings.csv`. Este archivo contiene las valoraciones de los usuarios sobre diversas películas.

Análisis del Resultado

Después de ejecutar esta función, el archivo `ratings.csv` se encuentra en la carpeta `ml-latest-small`. Esto permite que se pueda cargar en un `DataFrame` para su análisis posterior.

Paso 2: Carga de los datos en un DataFrame

Descripción

Cargamos el archivo `ratings.csv` en un `DataFrame` utilizando la librería pandas. Esto facilita el manejo y la exploración de los datos.

Análisis del Resultado

Los datos se encuentran dentro del rango mencionado y esperado. No existen números duplicados ni valores vacíos(NA).

Paso 3: Eliminación de películas y usuarios con pocas puntuaciones

Descripción

Eliminamos los películas (movieId) con menos de 10 puntuaciones, ya que es importante garantizar que cada película tenga un volumen significativo de interacciones para análisis confiables. Posteriormente, aplicamos el mismo criterio a los usuarios (userId) eliminando aquellos con menos de 10 puntuaciones. Esto ayuda a mantener únicamente a los usuarios más activos en el análisis.

Análisis del Resultado

Películas restantes: Calculamos el número de películas que superan el umbral. Usuarios restantes: Obtenemos cuántos usuarios permanecen tras aplicar ambos filtros. Puntuaciones restantes: El tamaño final del dataset tras estos pasos.

Paso 4: Histograma del número de puntuaciones por usuario y película

Descripción

Generamos dos histogramas para analizar la distribución de las puntuaciones:

```
Número de puntuaciones por usuario: Mostramos cuántos usuarios realizaron un
cierto número de puntuaciones.
Número de puntuaciones por película: Mostramos cuántos películas recibieron un
cierto número de puntuaciones.
```

Análisis del Resultado

En el histograma del número de puntuaciones por usuario, podemos observar que los usuarios tienden a dar menos de 200 valoraciones. Para el histograma del número de puntuaciones por película, observamos que los películas en general suelen tener pocas puntuaciones.

Paso 5: Histograma de la media de puntuaciones por usuario y película

Descripción

Creamos dos histogramas para analizar la distribución de las medias de puntuación:

```
Media de puntuaciones por usuario: Calculamos la media de las puntuaciones dadas
por cada usuario y representamos su distribución.
Media de puntuaciones por película: Calculamos la media de las puntuaciones
recibidas por cada película y representamos su distribución.
```

Análisis del Resultado

La tendencia de los usuarios es a valorar las películas entre valores de 3.0 y 4.5. Esto no nos indica mucho sobre el comportamiento de los usuarios, pues puede que esté sesgado la calidad del catálogo al que los usuarios acceden. Por otra parte, la tendencia de la puntuación por película también varía mayormente entre 2.5 y 4.0. Las películas no suelen llevar puntuaciones extremas, es muy difícil obtener un 1 o un 5.

Paso 6: Diagrama de barras de la distribución de las puntuaciones

Descripción

Representamos en un diagrama de barras la cantidad de veces que se ha dado cada puntuación.

Análisis del Resultado

Podemos observar que las puntuaciones más usadas varían entre 3 y 4. Esto nos indica que los usuarios tienden a valorar productos de forma "neutra", dado que los valores entre 3 y 4 son valores correctos al nivel de satisfacción tanto de usuario como de ítem.

Paso 7: Creación de un objeto `surprise.Dataset` y validación cruzada con 5 folds

Descripción

Se crea un objeto `surprise.Dataset` a partir del DataFrame y se realiza validación cruzada con 5 folds utilizando la función `set_my_folds`.

Análisis del Resultado

1. **Creación del Dataset**: Convierte el DataFrame en un formato compatible con Surprise para usar sus algoritmos.
2. **Validación Cruzada con 5 Folds**: Evalúa el modelo en 5 particiones diferentes, entrenando y probando en cada una.
3. **Uso del parámetro `shuffle=True`**: Asegura que los datos se mezclen aleatoriamente antes de dividirse en los folds.

Paso 8: GridSearchCV para determinar los mejores hiperparámetros de KNNWithZScore

Descripción

En este paso, realizamos una búsqueda de hiperparámetros utilizando `GridSearchCV` para encontrar la mejor configuración de los parámetros `k` y `min_k` para el modelo **KNNWithZScore**. La búsqueda se realiza utilizando la métrica de similitud **Pearson** y el criterio de evaluación **MAE (Mean Absolute Error)**. La validación cruzada se realiza con 3 folds (cv=3) y se utiliza el parámetro `n_jobs=-1` para aprovechar todos los núcleos de la CPU.

Los valores de los hiperparámetros `k` y `min_k` que se exploran son:

- `k`: [25, 50, 75]
- `min_k`: [1, 3, 5]

Análisis del Resultado

El valor de **MAE** más bajo indica una mejor capacidad de predicción.

Combinación óptima de hiperparámetros

Tras realizar el proceso de `GridSearchCV`, los mejores hiperparámetros encontrados para el modelo **KNNWithZScore** fueron:

- `k = 50`
- `min_k = 3`

Estos valores resultaron en el **MAE más bajo** (aproximadamente 0.6438 en el fold 3). Esta combinación fue la que mejor desempeño tuvo en todos los folds probados, lo que indica que es una configuración robusta y generalizable.

Conclusión Final

La combinación de parámetros **k = 50** y **min_k = 3** es la mejor opción para el modelo **KNNWithZScore**, ya que proporciona el menor **MAE** en todos los folds evaluados. Este conjunto de parámetros debe ser utilizado consistentemente en todos los subconjuntos de test en los ejercicios 8 y 9, tal como se requiere en el enunciado.

Esta combinación de parámetros será utilizada en todos los subconjuntos de test de los ejercicios 8 y 9.

Resultados de la búsqueda de hiperparámetros para KNNWithZScore:

fold	mae	params	test_mae
0	0.659112	{'k': 50, 'min_k': 1}	0.650623
1	0.658839	{'k': 75, 'min_k': 3}	0.648446
2	0.658385	{'k': 75, 'min_k': 5}	0.649056
3	0.658440	{'k': 75, 'min_k': 3}	0.647342
4	0.659804	{'k': 75, 'min_k': 3}	0.646750

Paso 9: Comparación de algoritmos con la métrica MAE

Descripción

En este paso se comparan los tres algoritmos de recomendación en función de su capacidad para predecir las valoraciones, utilizando la métrica **MAE (Mean Absolute Error)**:

Análisis del Resultado

1. Comparación de los MAE

Al analizar los valores de **MAE** de cada algoritmo en los 5 folds, obtenemos los siguientes resultados (promedios):

- **NormalPredictor**: MAE promedio de 1.109.
- **KNNWithZScore**: MAE promedio de 0.648.
- **SVD**: MAE promedio de 0.648.

2. Análisis de los Resultados

- **NormalPredictor**: Este algoritmo utiliza un enfoque aleatorio para predecir las valoraciones de los usuarios, lo que genera un **MAE bastante alto** en comparación con los otros dos algoritmos. Esto se debe a que la predicción aleatoria rara vez coincide con las valoraciones reales de los usuarios.

- **KNNWithZScore**: Este algoritmo, utilizando la métrica de similitud **Pearson** y los mejores hiperparámetros encontrados en el proceso de validación, muestra un **MAE bajo y consistente** a lo largo de los 5 folds. Esto indica que el modelo tiene un buen rendimiento en general y es capaz de hacer predicciones precisas basándose en las similitudes entre usuarios.

- **SVD**: El algoritmo basado en **descomposición en valores singulares** (SVD) también muestra un **MAE similar al de KNNWithZScore**. Ambos algoritmos tienen un rendimiento comparable, ya que ambas técnicas están basadas en la identificación de patrones y relaciones en las valoraciones de los usuarios.

3. Conclusión

- **NormalPredictor**: MAE = 1.109 (alto).
- **KNNWithZScore**: MAE = 0.648 (bajo y consistente).
- **SVD**: MAE = 0.648 (bajo, pero más complejo).

Ambos, KNNWithZScore y SVD, se comportan mucho mejor que NormalPredictor según la métrica MAE. La diferencia entre KNNWithZScore y SVD es insignificante, por lo que cualquiera de los dos podría considerarse el mejor dependiendo de la implementación y del fold evaluado.

Resultados de MAE para los algoritmos comparados:

fold	NormalPredictor_MAE	KNNWithZScore_MAE	SVD_MAE
0	1.111199	0.651515	0.653589
1	1.107600	0.649001	0.649431
2	1.107057	0.643029	0.644146
3	1.113336	0.650528	0.648360
4	1.105283	0.648996	0.649866

MAE Promedio:

- **NormalPredictor**: 1.110
- **KNNWithZScore**: 0.649
- **SVD**: 0.649

Paso 10: Comparación de recomendaciones con tamaños 1, 2, 5 y 10 y umbral igual a 4

Descripción

En este paso se comparan los 3 algoritmos y los mismos conjuntos para observar cual es el rendimiento y el grado de recomendación de cada uno de ellos.

Análisis del Resultado

El gráfico de la **Precision-Recall Curve at k** muestra que:

- **NormalPredictor** tiene tanto precision como recall significativamente más bajos que los otros algoritmos, lo cual es esperado ya que sus predicciones son aleatorias.
- **SVD** supera ligeramente a **KNNWithZScore** en precision en la mayoría de los casos (especialmente para listas más pequeñas como tamaño 1), lo que sugiere que SVD es más efectivo para identificar elementos relevantes de manera más precisa.
- **KNNWithZScore**, aunque cercano a SVD, muestra un comportamiento más consistente en listas de tamaños mayores (5 y 10), con una leve mejora en recall.

Conclusión:

- Si el objetivo es maximizar la **precisión**, **SVD** es el mejor algoritmo.
- Si el objetivo es un balance entre precisión y recall en listas de mayor tamaño, **KNNWithZScore** podría ser preferido.

Resumen de la práctica

En resumen, los resultados clave del análisis son:

1. **KNNWithZScore** y **SVD** tienen un desempeño casi idéntico en términos de **MAE** (Mean Absolute Error), ambos mostrando un MAE promedio de aproximadamente **0.648**. Esto indica que ambos algoritmos son muy efectivos para hacer predicciones precisas basadas en las valoraciones de los usuarios.

2. **KNNWithZScore** es ligeramente más eficiente en **recall** cuando se trabaja con listas largas, mientras que **SVD** destaca en **precisión**. Esto sugiere que **KNNWithZScore** podría ser preferido cuando se prioriza la cobertura de recomendaciones, mientras que **SVD** sería más adecuado cuando se busca asegurar la calidad de las predicciones.

3. **NormalPredictor**, que utiliza un enfoque aleatorio, tiene un rendimiento mucho peor con un MAE promedio de **1.109**, indicando que su capacidad predictiva es limitada en comparación con los otros dos modelos.

Conclusiones:

- **Mejor en MAE**: **KNNWithZScore**, aunque **SVD** es casi igual.
- **Mejor para recomendaciones**: **SVD** en términos de precisión y **KNNWithZScore** en términos de recall para listas largas.