ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA
Universitat Rovira i Virgili

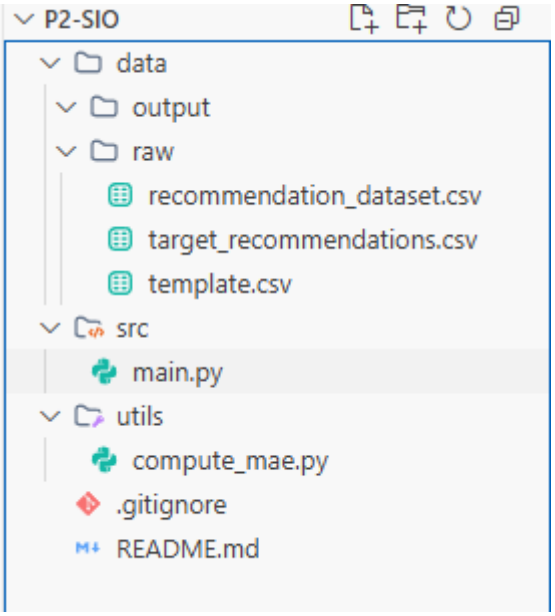
PRACTICA 2
SIO

Autores: Ton Llop, Alfonso Sánchez
Profesor: Edgar Batista

1. Base del proyecto y limpieza de datos

Esta práctica consiste en analizar y entender los mecanismos de recomendación en base a los diferentes datos que se obtienen de los usuarios. El objetivo es implementar técnicas de filtrado colaborativo para inferir y predecir las valoraciones desconocidas (representadas con el valor 99) dentro de una matriz dispersa de 73421 usuarios y 100 restaurantes. La finalidad última es maximizar la precisión del sistema de recomendación, minimizando el Error Absoluto Medio (MAE) entre las predicciones generadas y las valoraciones reales ocultas.

Para ello se ha elaborado el siguiente esquema de proyecto (scripts) para trabajar:



Para que los algoritmos funcionen correctamente hemos realizado este proceso para limpiar los datos:

1. Sustitución de todos los valores 99 por NaN, utilizando la función load_dataset() del módulo loadData.py.
   Esto es imprescindible porque un 99 influiría erróneamente en la media, la varianza y la medida de similitud entre usuarios o ítems.
2. Conversión de todas las columnas a tipo numérico para garantizar compatibilidad con librerías como NumPy y Surprise.
3. Obtención de una matriz Usuario × Restaurante con valores reales y NaN, adecuada para aplicar técnicas de predicción basadas en similitudes o modelos latentes.

Esta limpieza de datos garantiza cualquier cálculo estadístico que hagamos luego (medias, correlaciones, distancias, similitudes, etc.) refleje únicamente información válida y no valores no necesarios.

2. Los baselines

Los **modelos baseline** (línea base) son fundamentales por su simplicidad, ya que definen el umbral mínimo de error aceptable. Si modelos más sofisticados, como KNN o SVD, no logran mejorar el MAE (Error Absoluto Medio) de estos modelos de referencia, se considera que son ineficaces.

Actualmente, contamos con dos modelos baseline: la **media global** y la **media por usuario**.

2.1 Global mean

Esta es la predicción más simple posible. Asume que la valoración de un usuario para un restaurante desconocido es igual a la media de todas las valoraciones de todos los usuarios y de todos los restaurantes en el dataset.

La fórmula sería:

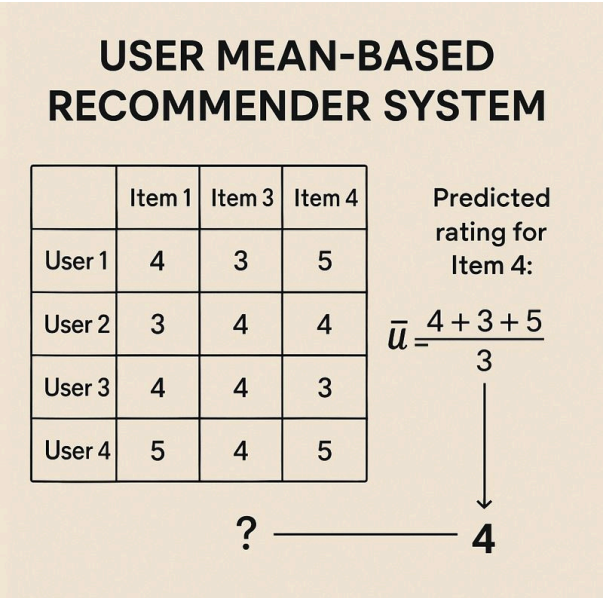
$$\hat{r}_{u,i} = \mu$$

Donde Rui es la predicción para el usuario u en el ítem i, y μ es la media de todas las valoraciones.

Hemos obtenido un MAE de 4.43 donde tenemos un error alto indicando que el dataset es disperso y que las preferencias de los usuarios no son uniformes.

2.2. User mean

Al igual que la media global ahora introducimos la primera capa de personalización. En este caso se asume que la valoración de un usuario para cualquier restaurante desconocido es igual a la media de sus propias valoraciones. Se basa en la idea de que hay usuarios generosos (siempre votan alto) o usuarios críticos (siempre votan bajo.



$$\hat{r}_{u,i} = \mu_u$$

El problema es que si un usuario no ha votado nada, nunca podremos predecir su próxima votación.

Hemos obtenido en este caso un MAE de 3.75, donde hemos reducido drásticamente comparado con la media global.

3. Filtrado Colaborativo: Item-KNN con Baseline

3.1 Definición

El algoritmo Item-KNN es una técnica de filtrado colaborativo donde la predicción para un ítem desconocido se obtiene analizando únicamente los ítems más parecidos a él entre aquellos que el usuario ya ha valorado.
La idea fundamental es: “Los restaurantes similares deberían recibir valoraciones similares por parte de un mismo usuario”
Para evitar tendencias propias de usuarios que puntúan de forma sistemáticamente alta o baja, o a restaurantes que suelen recibir valoraciones muy positivas o muy negativas, se utiliza además un baseline, que descompone cada valoración en:

- una media global μ,
- una tendencia individual del usuario b_u,
- una tendencia propia del restaurante b_i.

Esto permite separar la parte estructural del comportamiento y dejar al KNN la tarea de predecir únicamente la desviación respecto a ese patrón.

3.2 Implementación

Para implementarlo en calc.py hemos usado estas funciones:

- compute_baseline()
- compute_residuals()
- compute_item_similarity()
- predict_single_item_knn_baseline()
- predict_item_knn_baseline()

Nuestro sistema:

1. Carga el dataset, sustituyendo 99 → NaN (módulo loadData.py).
2. Calcula la estimación inicial $\mu + b_u + b_i$ con regularización (λ).
3. Obtiene los residuos.
4. Construye una matriz de similitud entre restaurantes basada en correlación Pearson con shrinkage.
5. Para cada pareja User–Restaurant del fichero target, predice usando los k ítems similares que el usuario sí valoró.
6. Estas predicciones se limitan al rango [-10, 10].

3.3 Parámetros clave

Parámetro	Función	Motivo
k	Nº de ítems vecinos.	k pequeño → vecinos fiables; k grande → riesgo. Se encontró óptimo alrededor de **12**.

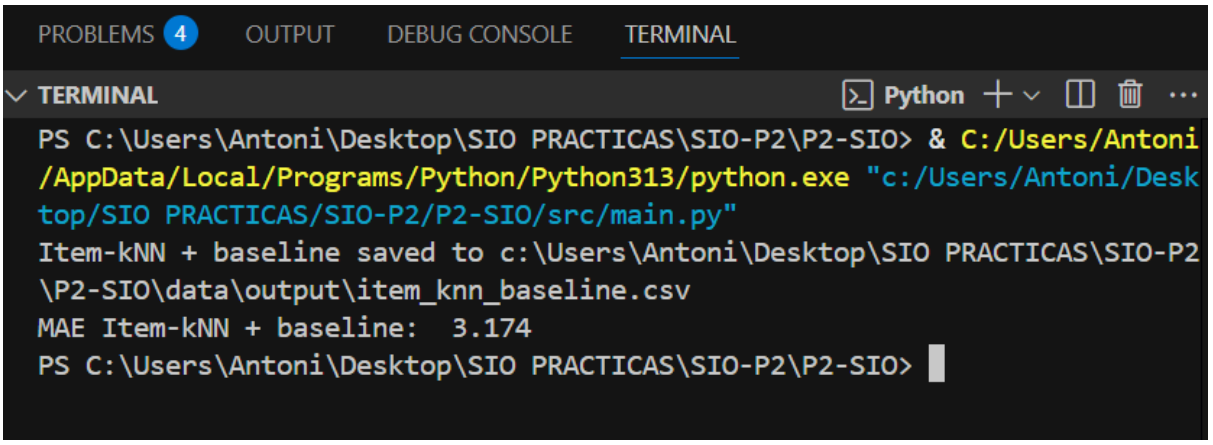
min_common	Mínimo de usuarios en común.	Evita similitudes espurias. Usamos **15** por equilibrio entre cobertura y robustez.
shrinkage_sim	Penalización de similitudes poco confiables.	Reduce correlaciones infladas cuando hay pocos datos. Valor óptimo **10–20**.
lambda_user, lambda_item	Ajuste y regularización de las tendencias	Evita sobreajuste en usuarios o ítems con pocos ratings. Valores típicos: $\lambda_u = 15$, $\lambda_i = 25$.

Estos parámetros se ajustaron mediante pruebas sucesivas y evaluando el MAE mediante el script oficial computeMae.py.

3.4 Resultados

Tras varias optimizaciones hemos obtenido un MAE de **3.174**.
Este resultado supone una mejora respecto a:

- Media global (MAE ≈ 4.43)
- Media por usuario (MAE ≈ 3.75)
- SVD estándar (≈ 3.23)



4. Factorización de Matrices: SVD

4.1. Contexto

Para este modelo, se optó por abandonar el enfoque de vecindad utilizado por KNN en favor de una técnica de Factorización de Matrices. Este método, en particular el algoritmo SVD,

alcanzó reconocimiento global tras el Netflix Prize de 2009, donde fue fundamental para aumentar la precisión de las recomendaciones de la plataforma en un 10%.

4.2. Concepto: Una red neuronal matricial

Aunque su nombre proviene del álgebra lineal el SVD funciona de manera análoga a una red neuronal superficial.

El sistema no “mira” a quién se parece a quien. En su lugar, intenta “aprender” dos matrices de pesos.

1. Una matriz de usuarios.
2. Una matriz de restaurantes.

El objetivo es descomponer la matriz original (R) en el producto de estas dos matrices menores de Factores Latentes.

$$R \approx U \cdot V^T$$

4.3. Factores latentes

El modelo comprime la información en un número reducido de características ocultas (n_factors = 30).

- Estos factores capturan conceptos abstractos deducidos.
- Ejemplo: El “factor 1” podría estar detectando si un restaurante es “elegante”, mientras que el “factor 2” detecta si es “comida rápida”. Si un usuario tiene un valor alto en el “Factor 1”, el modelo predecirá una nota alta para restaurantes elegantes.

4.4. Entrenamiento y configuración

A diferencia del cálculo directo del k-NN, este modelo se entrena con los datos para poder realizar juegos las predicciones paso a paso.
Los parámetros que usamos son:

- Factores: 30 - Suficientes para capturar matrices sin sobreajuste.
- Épocas: 75 - Número de iteraciones de aprendizaje.
- Learning Rate: 0.001 - La velocidad a la que el modelo corrige sus errores
- Regularización: 0.2 - Evitar que los pesos crezcan demasiado.

4.5. Resultados

El modelo SVD alcanzó un MAE de 3.235. Aunque numéricamente sea inferior al KNN, su valor es que comete errores diferentes, lo que es útil para combinarse con el KNN.

5. SVD + KNN

5.1.

Tras evaluar los modelos individuales, observamos que el Item-KNN y el SVD ofrecian los mejores resultados, pero funcionaban de manera distinta:

Item-kNN (Enfoque Local): Es excelente encontrando similitudes específicas entre restaurantes vecinos. Funciona muy bien cuando hay valoraciones directas compartidas.

SVD (Enfoque Global): Es excelente capturando la estructura latente de toda la matriz. Funciona mejor generalizando patrones abstractos y rellenando huecos donde el kNN no encuentra suficientes vecinos.

Nuestra hipótesis fue que sus errores no estaban correlacionados: donde uno falla, el otro suele acertar. Por tanto, combinarlos mediante una técnica de ensemble (Linear Blending) debería reducir el error global.

5.2. Metodología de mezcla

Implementamos la suma ponderada de las predicciones de ambos modelos. La fórmula aplicada fue:

$$\hat{r}_{final} = \alpha \cdot \hat{r}_{kNN} + (1 - \alpha) \cdot \hat{r}_{SVD}$$

Para determinar el peso óptimo (alpha) realizamos una búsqueda iterativa. Creamos un optimizador que determinó que la mejor combinación otorgaba un peso del 60% al knn y un 40% al SVD. Esto tiene sentido, pues el kNN individual tenía un rendimiento ligeramente superior.

5.3. Validación y selección de modelos

Durante la fase de experimentación, intentamos integrar un tercer modelo: KNN con Normalización Z-Score. Sin embargo, al añadirlo a la ecuación, el MAE subió al 3.182. Esto nos llevó a una conclusión importante: **añadir complejidad no siempre garantiza mejores resultados.** El modelo Z-Score añade ruido en la predicción por lo que decidimos descartar este modelo.

5.4. Recorte de rango

Dado que el SVD es un modelo matemático puro. En ocasiones genera predicciones fuera del rango lógico (ej: 10.5 o -11.2). Como paso final, aplicamos una técnica de Clipping para restringir todas las predicciones al intervalo válido [-10, 10].

$$\hat{r}_{clipped} = \max(-10, \min(10, \hat{r}_{final}))$$

Esto fue crucial para eliminar el error residual de los valores extremos.

5.5. Resultado

Gracias a la sinergia entre la precisión local del kNN y la generalización global del SVD, junto con el ajuste de pesos y el clipping, logramos **nuestro mejor resultado: 3.143.**

Este valor representa una mejora sustancial respecto al baseline inicial (4.43) y supera el rendimiento de cualquier modelo individual por separado, confirmando el éxito de la estrategia híbrida.

6. Experimentos No Exitosos: KNN con Normalización Z-Score y Mezcla Triple

6.1 Definición

La normalización Z-Score es una forma de poner todas las valoraciones de un usuario en la misma escala, para que su manera particular de puntuar no afecte al cálculo de similitudes.
Cada usuario tiene su propio “estilo” al valorar:

- algunos siempre puntúan muy alto,
- otros muy bajo,
- y otros tienen valoraciones muy irregulares.

Z-Score intenta corregir estas diferencias para comparar usuarios e ítems de forma más justa. Lo que hace es convertir cada valoración en una medida que indica si esa nota está por encima, por debajo o cerca de lo que el usuario suele hacer normalmente.
En lugar de fijarse en el número exacto de la valoración, la normalización se fija en qué tan distinta es esa valoración respecto al comportamiento habitual del usuario. Así, las valoraciones de personas “generosas”, “estrictas” o “variables” pasan a una escala común.

La idea era corregir tendencias particulares de cada usuario: algunos tienden a puntuar muy alto o muy bajo, mientras que otros muestran una gran variación entre valoraciones.
La hipótesis inicial era que, al poner todas las valoraciones en una escala comparable, la detección de restaurantes similares sería más precisa.

6.2 Implementación y funcionamiento

La versión con Z-Score se implementó mediante el algoritmo KNNWithZScore de la librería Surprise. Para ello:

1. Se eliminaron previamente los valores marcados como 99.
2. Se entrenó un modelo basado en ítems utilizando métricas de similitud ajustadas con shrinkage.

3. Las predicciones para el conjunto target se obtuvieron directamente mediante el valor estimado del modelo.

Este enfoque se integró en el mismo flujo de trabajo que el Item-KNN tradicional, permitiendo comparar fácilmente los resultados.

6.3 Resultados

El rendimiento del modelo no cumplió las expectativas: **MAE = 3.239**
Este valor fue inferior al obtenido por el Item-KNN con baseline, demostrando que la normalización no aportaba mejoras en este entorno concreto.

6.4 Análisis del resultado

Las razones principales por las que este método no funcionó mejor incluyen:

- Usuarios con pocas valoraciones: su media y desviación estándar pueden no ser fiables, lo que provoca transformaciones incorrectas.
- La normalización puede exagerar variaciones pequeñas y afectar negativamente a la similitud entre ítems.
- Algunos comportamientos consistentes de los usuarios, que ayudan al KNN tradicional, quedan diluidos con la transformación.

6.5 Mezcla triple: KNN + SVD + KNN-ZScore

También se probó un modelo híbrido que combinaba tres enfoques distintos: KNN con baseline, SVD y el modelo con Z-Score. La intención era aprovechar las ventajas complementarias de cada uno.
Sin embargo, el resultado final fue: **MAE = 3.182**
Es decir, la inclusión del componente basado en Z-Score reducía la calidad del modelo combinado, impidiendo que alcanzara el rendimiento del híbrido simple (KNN + SVD), que sí ofrecía mejoras claras.

6.6 Conclusión final

Tras evaluar el comportamiento de los modelos, se concluyó que la variante con Z-Score no aportaba beneficios reales sino que disminuye la estabilidad del sistema. Al final decidimos mantener únicamente el modelo híbrido basado en KNN y SVD, que demostraba mejores resultados con una estructura más clara y manejable.

7. Tabla resumen y conclusiones

Finalmente esta ha sido la tabla de resultados obtenida:

Modelo / Algoritmo	MAE (Error Medio Absoluto)	Notas / Observaciones
Global Mean	**4.430**	Baseline simple (Media de todo el dataset).
User Mean	**3.750**	Baseline personalizado por usuario.
Item-kNN + Baseline	**3.174**	Filtrado Colaborativo (Mejor modelo individual).
SVD (Matrix Factorization)	**3.235**	Factores Latentes (30 factores, 75 épocas).
KNN With Z-Score	**3.239**	Normalización estadística (No mejoró el kNN base).
Ensemble (kNN + SVD + ZScore)	**3.182**	Mezcla de 3 modelos (El ZScore introdujo ruido).
Ensemble Final (kNN + SVD)	**3.143**	🏆 **MEJOR RESULTADO**

Esta práctica nos ha permitido experimentar con diferentes modelos y fórmulas para calcular el recomendado de usuario. Si bien es cierto que hoy en día se usan modelos de IA avanzados para las recomendaciones y que seguramente haciendo uso de ellos se podría mejorar mucho el resultado.