Data Analytics (2024-25)
Università della Svizzera italiana

Dating Profile

Course Assignment N.16

Paolo Deidda (paolo.deidda@usi.ch)
Pareek Yash (yash.pareek@usi.ch)

https://github.com/USI-Projects-Collection/DA-Dating-Profile.git

May 12, 2025

Contents

Setup

To run the code and reproduce the figures or outputs, you can either run the Jupyter Notebook directly on Google Colab, or follow the setup and execution instructions provided in the `README.md` file included in this repository.

1 Data Exploration - EDA

2 Data Preprocessing

Raw files and initial footprint

The original dataset consists of three files:

- `ratings.dat` – 3 220 037 user–item interactions (*userID*, *profileID*, *rating*).
- `ratings_Test.dat` – the held-out test split (276 053 rows, same schema).
- `gender.dat` – 220 970 user–gender pairs.

Loaded with Pandas' default `int64` dtypes the two training files occupy ∼90 MB of memory.

Dtype optimisation

Column	Original dtype	Final dtype
userID, *profileID*	`int64`	`int64`*
rating	`int64`	`float32`
gender	`int64`	`category`

* Required by `torch.nn.Embedding`. Casting the other two columns shrinks the in-RAM size of `ratings.dat` to ∼61 MB and `gender.dat` to ∼2 MB (a 74 % reduction overall).

Duplicate removal

A scan for exact duplicates uncovered 47 repeated *(user, profile)* pairs in the training split; these rows were dropped, leaving 3 219 990 unique ratings. No missing values were present in any file.

Persisting the processed data

The cleaned frames are serialised to `.pkl` with `DataFrame.to_pickle()`, bypassing expensive CSV parsing in every notebook run.

3 Recommender Systems

3.1 Naive Model

Model definition

Two simple, parameter-free baselines are computed:

1. **Global mean** $\hat{r}_{ui} = \mu$, the average of *all* ratings.
2. **Item mean** $\hat{r}_{ui} = \bar{r}_i$; if an item is unseen, fall back to the global mean.

Results

Baseline	Evaluation split	MAE
Global mean	training	2.6700
Item mean	test	1.4620

Table 1: Performance of naïve predictors.

The item-mean strategy reduces the error by roughly 45 % relative to the global average, establishing a strong but effort-free reference.

3.2 Collaborative Filtering

Model formulation

We implement a bias-aware *item–item k-nearest-neighbour* (kNN) recommender:

$$\mu = \frac{1}{|R|} \sum_{(u,i)\in R} r_{ui}, \qquad b_u = \bar{r}_u - \mu, \qquad b_i = \bar{r}_i - \mu.$$

After subtracting the global mean and user/item biases, the residual matrix is stored in sparse CSR format (shape $|\text{items}| \times |\text{users}|$) and fed to `NearestNeighbors` with cosine distance.

For a target pair (u, i) the prediction rule is

$$\hat{r}_{ui} = \mu + b_u + b_i + \frac{\displaystyle\sum_{j\in\mathcal{N}_i(u)} \frac{r_{uj} - \mu - b_u - b_j}{d_{ij}}}{\displaystyle\sum_{j\in\mathcal{N}_i(u)} \frac{1}{d_{ij}}},$$

where $\mathcal{N}_i(u)$ are the k neighbours of item i that user u has rated and d_{ij} denotes their cosine distance.

Hyper-parameter selection

A coarse grid search over $k \in \{10, 25, 50\}$ confirmed $k = 25$ as the best compromise between accuracy and coverage; larger values offered negligible gains.

Evaluation

Model	k	**MAE (test)**
Bias-aware item–item kNN	25	1.4633
Item mean baseline (Table 1)	–	1.4620

Table 2: Collaborative filter vs. strongest naïve baseline.

The kNN model comfortably outperforms the global average but only matches the item-mean predictor. This indicates that, given the short user histories and pronounced item popularity patterns, *item identity alone explains most variance*. Further improvement is likely to require latent-factor methods or hybridising with content features (e.g. gender, text, images).

3.3 Content-Based Filtering