Data Mining Assignment 3 – Recommender Systems

Problem Setting: Movie Recommendation

For this assignment, you will work with a dataset of movies **“movies.csv”** and user ratings **“ratings_train.csv”**. The goal is to develop a recommender system that suggests movies to users based on their preferences and past interactions. Recommender systems play a crucial role in various industries, including entertainment, e-commerce, and online content platforms. Your task is to build and evaluate different recommendation models and analyze their effectiveness.

Before implementing your recommendation models, you need to preprocess and combine these datasets. This includes handling missing values, encoding categorical features, and normalizing numerical attributes.

Task 1: Building Recommender Models and Evaluation

You must implement at least **two** different types of recommendation models. Utilise a **collaborative filtering** approach, such as user-based, also utilise a **matrix factorization** technique.

Once you have built your models, evaluate their performance using appropriate metrics. For rating predictions, root mean squared error (RMSE) can measure accuracy, while ranking-based evaluation can rely on precision@K, recall@K. Additionally, you may explore diversity and novelty measures to assess the variety of recommendations. After comparing the models, discuss their trade-offs in terms of accuracy, computational efficiency, and personalization.

Task 2: Making Predictions for Specific Users

Using your best-performing recommender model, generate personalized movie recommendations for each user of the dataset “**ratings_test.csv**”. Fill in the csv with the 10 best recommendations for each user (movieId), the order of the movies does not matter. Discuss the strategies you use and how you handle cold cases (new users that have no interactions yet) in the report.

Submission Details

- Implement your code in any language (Python recommended). Use libraries like:
    1. **pandas** for data manipulation
    2. **sklearn** for preprocessing and evaluation
    3. **surprise or lightfm** for recommendation models
    4. **matplotlib** or **seaborn** for visualizations
- Include your findings in a concise report of maximum 4 pages.
- Submit both the code, your results and the report(pdf) as a **.zip** file via Blackboard by **16/05/2025**.
- The ZIP file should include your full name like, **Lastname_Firstname-studentNumber.zip.**

Questions: In case you have any questions specific to the assignment, please send an email to nick.wils@uantwerpen.be

Evaluation Criteria

Your submission will be evaluated based on the following grading rubric.

	Less than 7	**7 to 10**	**10 to 13**	**14 to 17**	**18 to 20**
Writing Style	The report is very confusing; the writing style is below average.	At places the report is not very clear; there is a lack of clear structure.	The text is overall clear although some parts could be improved.	Most of the text is easy to follow, findings are explained in a clear way.	The text is very clearly and concisely written, illustrative examples and figures are not overused, but added where needed.
Task 1	Task was not completed.	Task was only completed in a minimal way. Incorrect or insufficient argumentation is given for the choices that were made. The result analysis is not clear.	Task was completed. Some of the explanations behind the methodology/results were okay, though some aspects are missing.	Task was completed and the motivation behind the methodology as well as the result analysis are clear. Some interesting points are made.	The motivation behind the methodology and the result analysis are excellent. Most decisions are backed either by numerical analyses, scientific papers or convincing arguments.
Task 2	Task was not completed.	Incorrect or insufficient argumentation is given for the performance estimate.	Effort was done to complete the task but some important aspects are missing.	Most of the task execution was done and motivated well. Only minor aspects have been overlooked.	Complete and thorough analysis was done to complete the task.
Code	Code raises many errors.	Code raises some errors or is very unclear.	Code runs but lacks clear structure and readability, only little documentation is given.	Code is readable and sufficiently documented.	Code is very readable and well documented. It is structured in a way that only by (un)commenting single lines, the code for the different tasks can be run.