# Project Part II: Build a Recommendation System!

We've already conducted some data pre-processing on our dataset last time in **project part 1**. Now, you can build a recommendation system based on the dataset, by ***collaborative filtering***(協同過濾) technique! (particularly, ***user-based*** method). The concept of collaborative filtering is very easy and straightforward. If two people have mostly common experiences, then what a person likes (dislikes) very likely works on the other person as well. That is to say, they probably share the same preferences. We can retrieve the common histories of these users, and calculate the similarities between them. Thus, we can recommend an item to a user if the item is highly rated over other users with similar tastes.

You are asked to build a user-based recommendation system based on the datasets we provided last time (or other dataset that satisfies the requirements described in project part 1). I will guide you through the whole analysis processes by asking you the following questions. Please answer them in a **10-page (at most)** reports.

## Guidelines and Questions

The first step to execute the collaborative filtering algorithm is to **find the users that share the same tastes.** So, you need to construct a huge table which records every users' opinion (rating) to every item.
And then, you need to calculate the similarity between these users.
**Q1.** (15%) How do you calculate the similarity? Do you use *Jaccard similarity? Pearson coefficient?* or *Cosine similarity?* Or maybe, the similarity measure you used is none of the above, then what is it? Please explain why you choose the metric. Also, describe how you calculate the similarity between any 2 users, as well as how you implement it in your code.
**Q2.** (15%) Aside from the ratings, do you use other features to help deriving the similarity? If you do, how do you use them and how do you implement in your code? And if you don't, what features do you think might be helpful in calculating similarity? How could you fit the features in the procedure?

Now, after you computed all the similarities, you can pick out a group of users (say, 10 users) who are the most similar to the current user. We don't simply choose the highest one. Instead, we choose the *k* users with highest similarity, for that the degree of confidence is way bigger.

The next step will be for each user in testing (or validation) set, list out all the items that the user hasn't tried out while other users highly similar to him(her) have used it. Then, we pick the most high-rated ones as our recommendations from these items. We then successfully make a recommendation!

**Q3**. (25%) Please briefly explain the implementation regarding the procedures mentioned above in your code. That is, how you pick out the most similar *k* users to a certain user? How you filter out those items that the user has not experienced but the other similar *k* users have tried? Finally, among these items, how you make the final recommendation?

Please note that due to the form of testing dataset, your system should not only be able to *make a recommendation on item*, but also need to have ability to **estimate the ratings**, given the user and the item. You need to come up with a reasonable estimation method on these ratings. You are welcomed to refer to any papers.

**Q4**. (15%) Please describe how you estimate the ratings, given a user and an item, such as explaining it by formula. Also, briefly explain how you accomplish the estimations in your code.

**Hint**: I will give you an example as the baseline: the following method is proposed by this paper: https://link.springer.com/chapter/10.1007/978-3-319-90092-6_10.

Given a user $u$ and an item $i$, we can build a rating estimation model:

$$\widehat{Rating}(u,i) = \mu + b_u + b_i$$

, where $\mu$ is the average rating in the system.

$$\mu = \frac{\sum_{r_{ui} \in R} r_{ui}}{|R|}$$

, $b_i$ is the item bias representing if an item is, on average, rated better or worse than average.

$$b_i = \frac{\sum_{u \in U_i} (r_{ui} - \mu)}{|U_i|}$$

and $b_u$ is the user bias representing whether the user tends to rate high or low on average.

$$b_u = \frac{\sum_{i \in I_i} (r_{ui} - b_i - \mu)}{|I_u|}$$

However, note that when the user *u* or the item *i* have only very few ratings, the predictions can be very unreliable. One way to fix this is to introduce a damping term *β* to the denominator of the computations. Motivated by Bayesian statistics, this term will shrink the bias terms towards zero when the number of ratings for an item is small while having a negligible effect when the number of ratings is large. The modified formulas look like this:

$$b_i = \frac{\sum_{u \in U_i}(r_{ui} - \mu)}{|U_i| + \beta}$$
$$b_u = \frac{\sum_{i \in I_i}(r_{ui} - b_i - \mu)}{|I_u| + \beta}$$

The values of this damping parameter that have been used in the past range from 5~25. But this is only for reference. It is recommended that you re-tuned for your sys.

As easy as collaborative filtering technique, there is a drawback though. If the user to be examined is a ***new user***, which means that he(she) does not have many histories, this method would probably fail to deliver good recommendations. It becomes way harder to find matching users by similarity if we simply have no clue what the user might favor.

**Q5.** (10%) To overcome the aforementioned downside of user-based collaborative filtering, what measures will you take? Do you implement the measure in your code? If so, how do you do that?

## System Testing and Demonstration

We will be testing your recommendations system by testing datasets we preserved. Also, according the schedule arranged by the teacher, we will ask you to demo how you execute your recommendation system. Details are given below:

**1. Book-Crossing Dataset**

This dataset contains 278,858 users with 1,149,780 ratings about 271,379 books. For those students using this dataset, your recommendation system would be tested by the 114,978 records preserved, and ranked by the accuracy to compare with other students. *±1 from the ground truth rating* would still be considered accurate.

**2. The Echo Nest Taste Profile Subset**

This dataset contains 1,019,318 users and 384,546 songs included in MSD, with each record presented as a *"user - song - play count"* triplet. There are totally 48,373,586 such triplets in the dataset. For those students using this dataset, the remaining 4,837,359 triplets (10%) would be preserved as test cases to examine your recommendation system. The performance will be ranked by the accuracy to compare with other students. *±10 from the ground truth counts* would still be considered accurate (might adjust).

**3. Other datasets**

If you are using your own dataset that ***at least*** has **a set of users** and at least **a set of items,** you should use **train_test_split()** to generate a testing subset. When handing in this **project part 2**, **you should hand in the testing set as well, including the ground truth column.** If the ratings are categorical values within a small range, like those in book-crossing dataset, we will consider *±1* from the ground truth rating accurate. Otherwise, if the ratings are some sorts of counts, like those in Echo Nest Taste profile subset, we will deem *±10* from the ground truth counts accurate. The performance will be compared with others who use their own datasets as well. (**Note: If you have to validate your performance, please separate out an extra validation subset from your training data! Do not peek the answers in testing dataset while you are training, and do not use some direct copy-paste technique from testing set in your predictions, or you will get 0!**)

**You should hand in the predicted ratings(counts) for all the given user-item pairs in testing dataset! Please store the predictions in a csv file, naming after your student ID** (see the hand-in rule section). **Also, please order you predictions as the user-item pairs listed in testing dataset. Do not include any header row or index column.**

**Q6**. (10%) of your score will be based on your **performance ranking**, compared with others using the same dataset. The higher-ranked 50% get a full 10%, while the lower-ranked 50% get 5% in this section.

---

**Q7**. (10%) The last 10% of your score in project part 2 is based on the **demonstration**. The tentative schedule of demo will be on **6/16**. (You can take back your answer sheets of both midterm and final exams the same day.) (If any rearrangement were made, we would inform you on E3 announcement.) The purpose of the demo is just to make sure that you actually built the system on your own, and made predictions on testing data in a proper way.

## Hand-in Rules *(Important!)*

*The project will be divided into 2 ~ 3 parts, each with full credits (100%). The final score of your project will be the arithmetic average of the grades of these parts.*
You shall hand in a report with **a maximum of 10 pages in pdf format**, as well as your **Python code.** You are allowed to write your report in either English or Chinese. While turning in, name **all** your files as your **STUDENT ID,** described as follows:

| | |
|---|---|
| **Code** | **310554004.py** or **310554004.ipynb** |
| **Report (in pdf)** | **310554004.pdf** |
| **Predictions (in csv)** | **310554004.csv** |
| **Testing data**, including **answers** ***(only for those using their own dataset)*** | **310554004_test.csv** |

Finally, the deadline of Project Part II will be **June 15, 23:59:59**.

## Penalty

1. You will get a **20% discount** for your grades for **late submission**. For those who don't even bother doing the make-up submission, you will get **0** points.

2. You will get **0 points** without any review if the files you handed in violate the **hand-in rules** listed in the previous page. So, feel free to ask me if you have any question regarding the hand-in format. **Don't risk your scores!**

3. Even if you submitted your code, report and predictions, **as long as you don't come to demo your system, you will still get 0 points without any compensation.** So, if you really have difficulty coming to demo the system that day (say, in a quarantine), please contact me or teacher as early as possible! We will arrange an online demo.

---

**Some announcement**: for my own convenience, if you want to ask me or other classmates questions, please ask on the **Q&A** 提問區 on E3 system. I can easily gather questions this way. You might also help others indirectly, since your question might probably be others classmates' question, too. If you know the answers to some questions on the discussion forum area, feel free to answer them for me. I will be grateful for that ♥

**Do not mail my G-mail account or my E3 account directly. I will not respond to you from now on.**