Assignment 1

**Jun Wang**

**Date:2022-11-9**

# Task 1 - Predicting the Read

## **Description of the problem**

In this task, we need to predict given a (user,book) pair from ‘pairs Read.csv’ whether the user would read the book (0 or 1). Accuracy will be measured in terms of the categorization accuracy (fraction of correct predictions). The test set has been constructed such that exactly 50% of the pairs correspond to read books and the other 50% do not.

## **Dataset**



200,000 ratings to be used for training. This data should be used for the ‘read

prediction’ (both classes) and ‘rating prediction’ **(CSE258 only)** tasks. It is not necessary to use *all*

ratings for training, for example if doing so proves too computationally intensive.

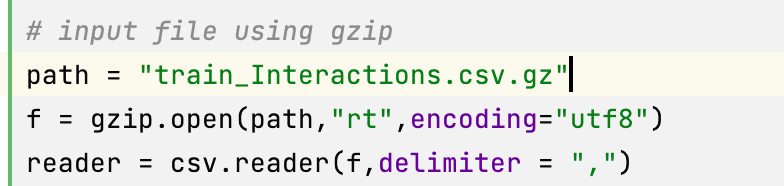
**userID** The ID of the user. This is a hashed user identifier from Goodreads.

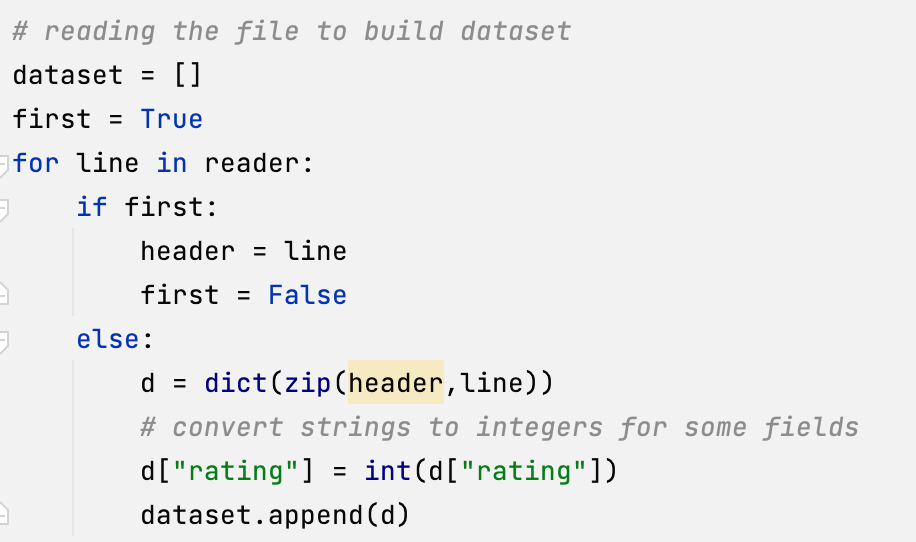
**bookID** The ID of the book. This is a hashed book identifier from Goodreads.

**rating** The star rating of the user’s review.

* **Preprocess for the dataset**

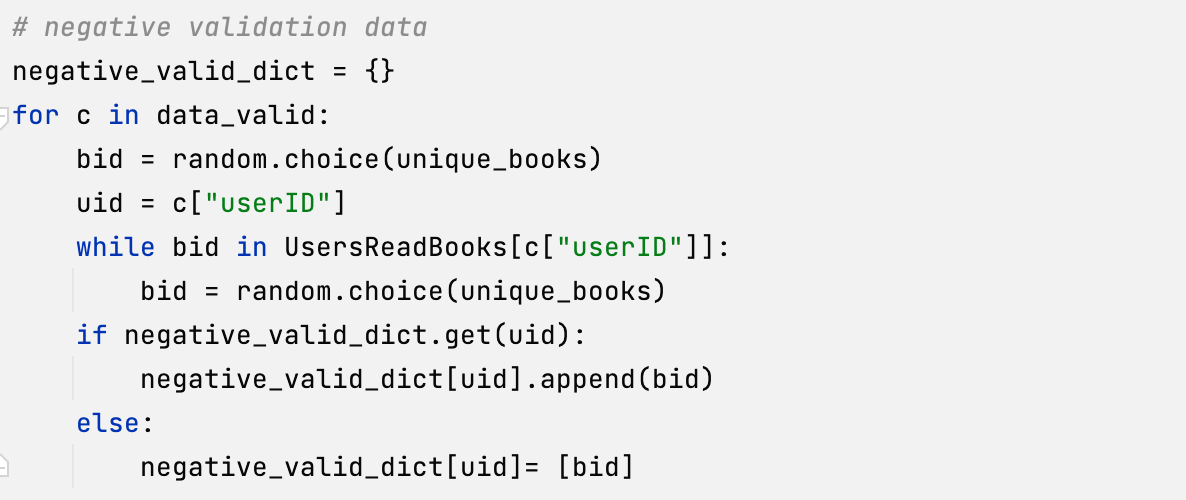
Frist, read the dataset and split them into train set and valid set:



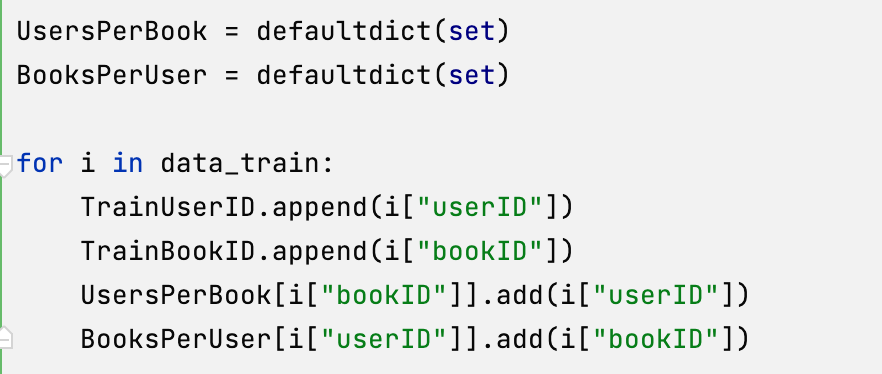


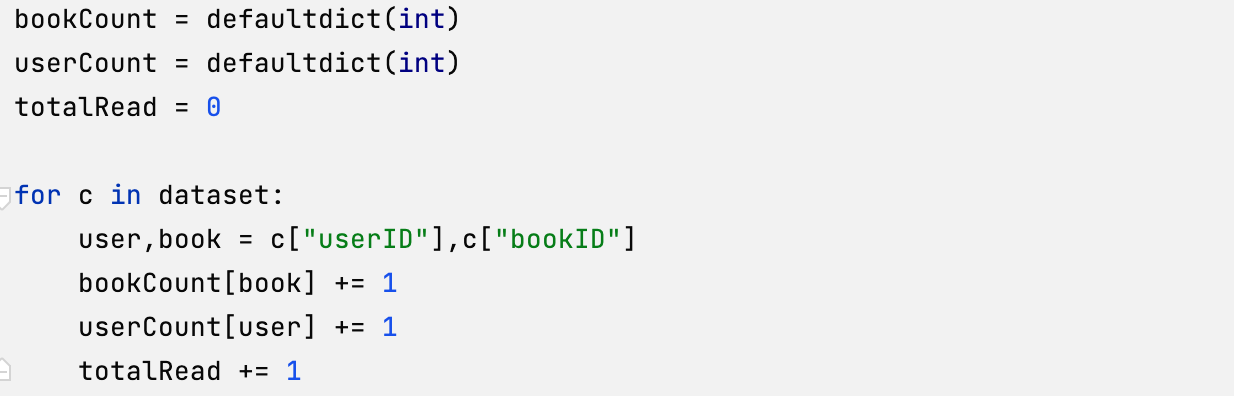


There is a problem here: since all the dataset is positive, we need to build some negative dataset for the pairs.



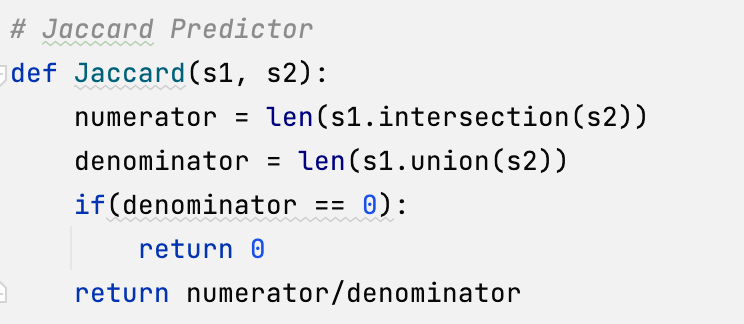
Then, prepare the neccessary data structure:





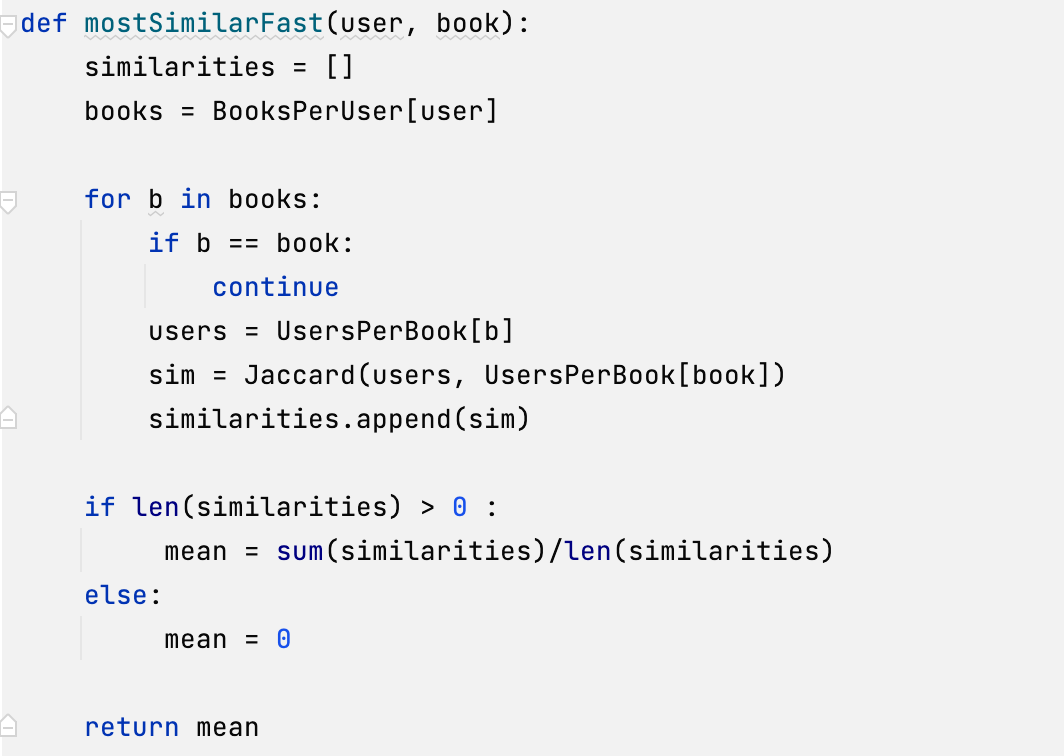
## **Methods**

In homework3, we notice that 75% can be obtained based on Book popularity. So we should make the best use of it. Also we have the Jaccard similarity, which ranges from 0 to 1. Here is the idea, we can create a new variable c = Jaccard similarity \* book counts. Based on the new variable, we can make a list and sort them. If the pair (u,b)’s c is in the top x percent of the ranking, we predict y to be “1”. Otherwise, we predict y to be 0.



Based on the similarity, we have lots of things to improve. That is what similarity we should choose, and whether the mean or max or min of the similarity list we should choose.

For the first questions, since it is the ranking for items, we should calculate the similarity based on items. In the following function, we can get a 79% accuaracy.

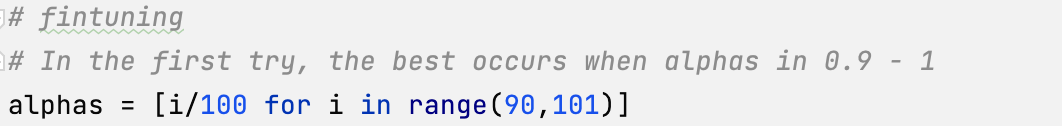


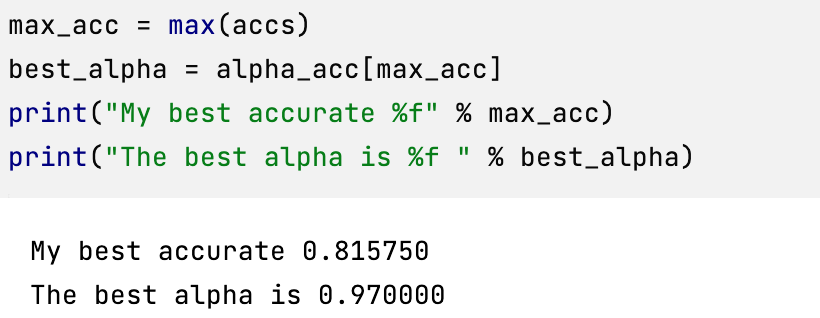
But, in some case, the item is not seen. We need to use the infomation from the users, so we can interchange user and item and set a factor alpha for them.



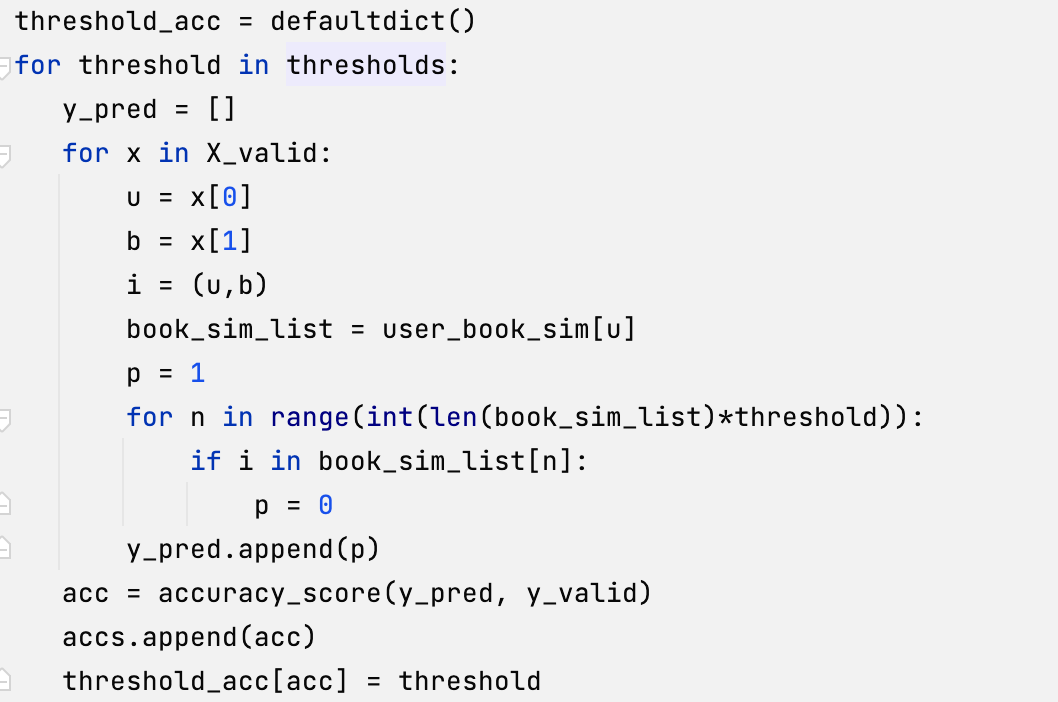
mean = alpha\*mean1 + (1-alpha)\*mean2

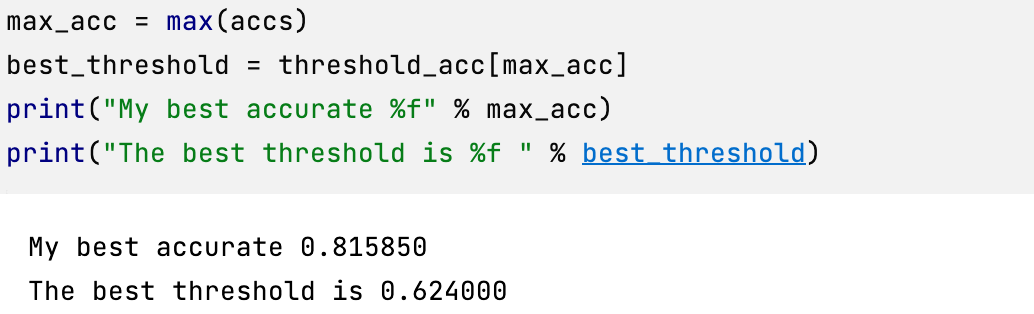
**Fintuning** the alphas: Get a 81% accuracy and best alpha is 0.97 which makes sense that the unseen item situation is rare.





**Also fintuning** the thresholds:





# Task 2 - Predicting the Rate

## **Description of the problem**

Predict people’s star ratings as accurately as possible, for those (user,item) pairs in ‘pairs Rating.txt’. Accuracy will be measured in terms of the *mean-squared error* (MSE).

* **Preprocess for the dataset**

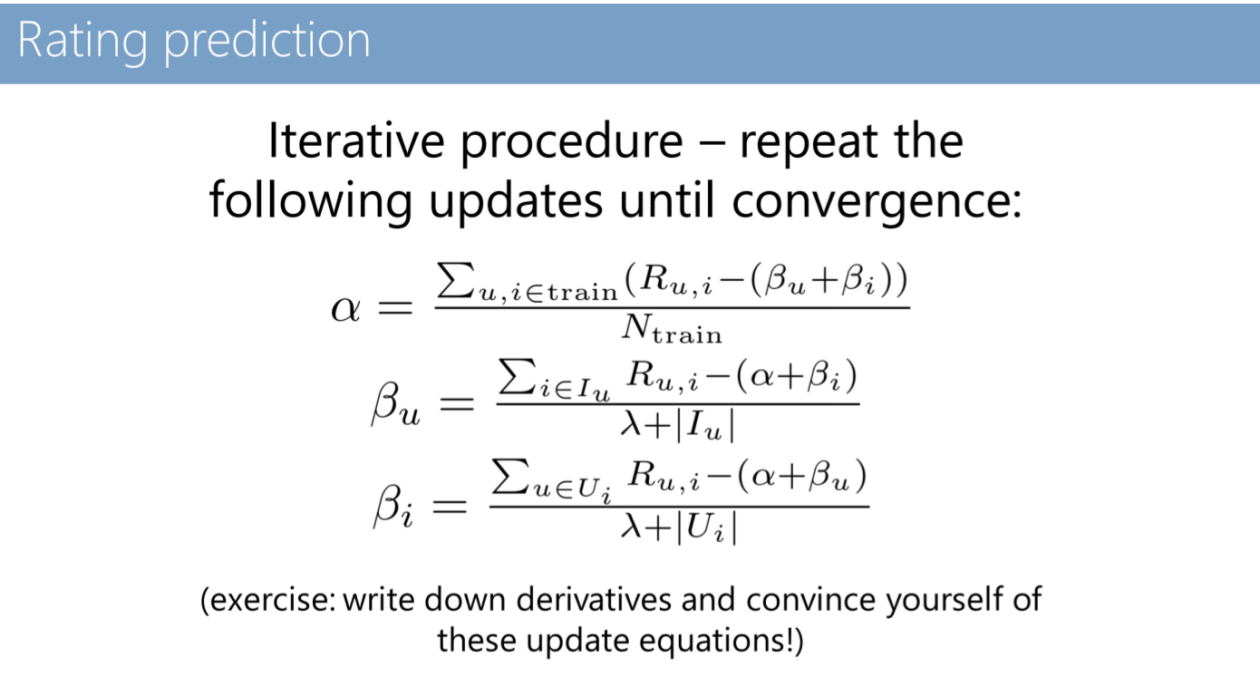
Frist, read the dataset and split them into train set and valid set like the task 1.

Prepare the neccessary data stucture: beta\_user, beta\_book

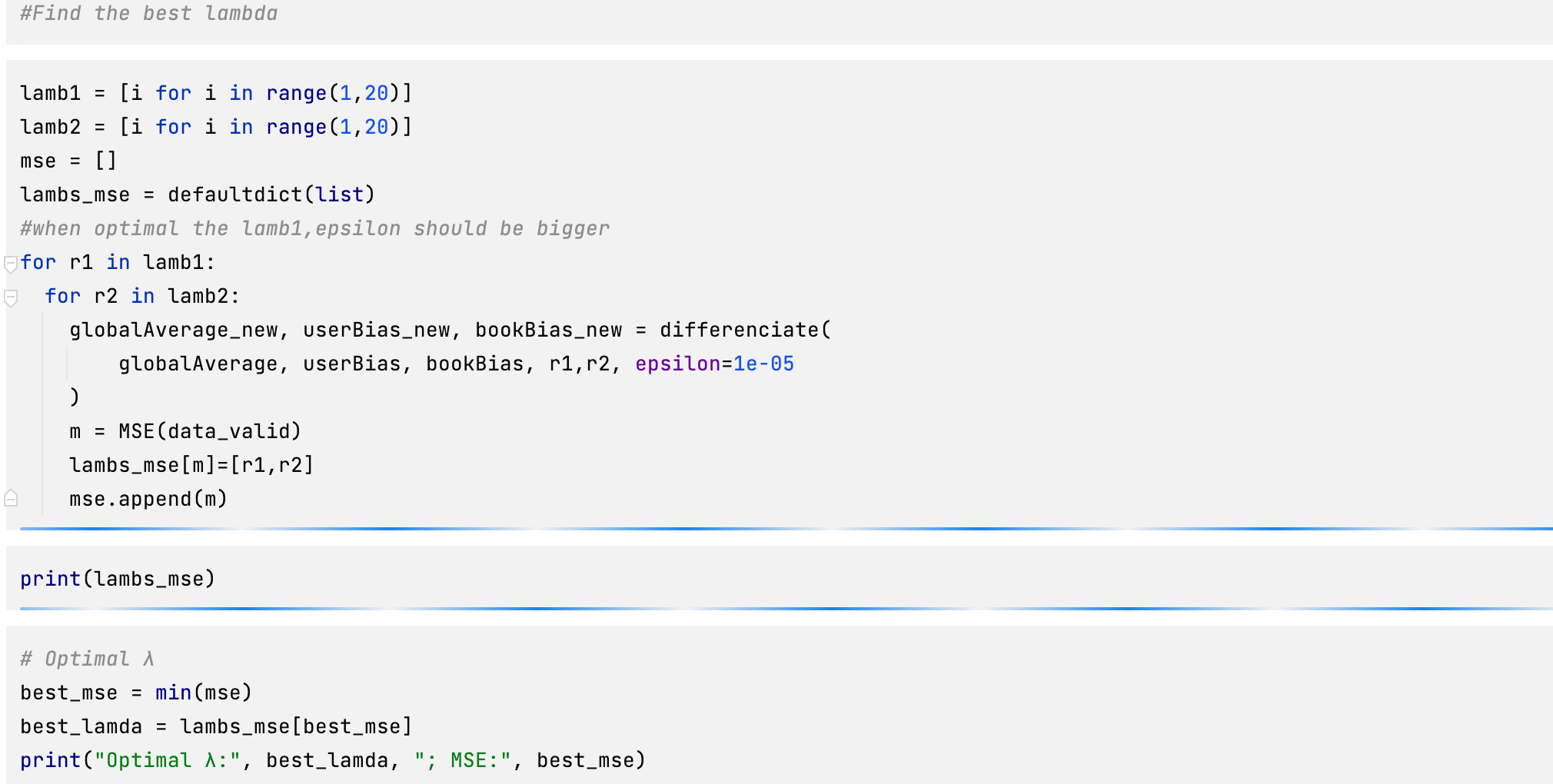


* **Methods: latent factor models**

Traditional one:

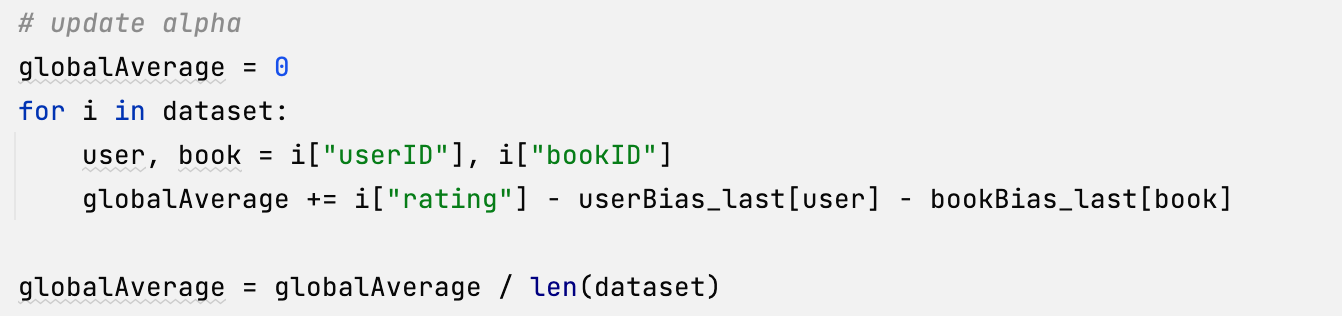


Here I use different lambdas for user-bias and book-bias, and finetune the factors based on these two lambda:

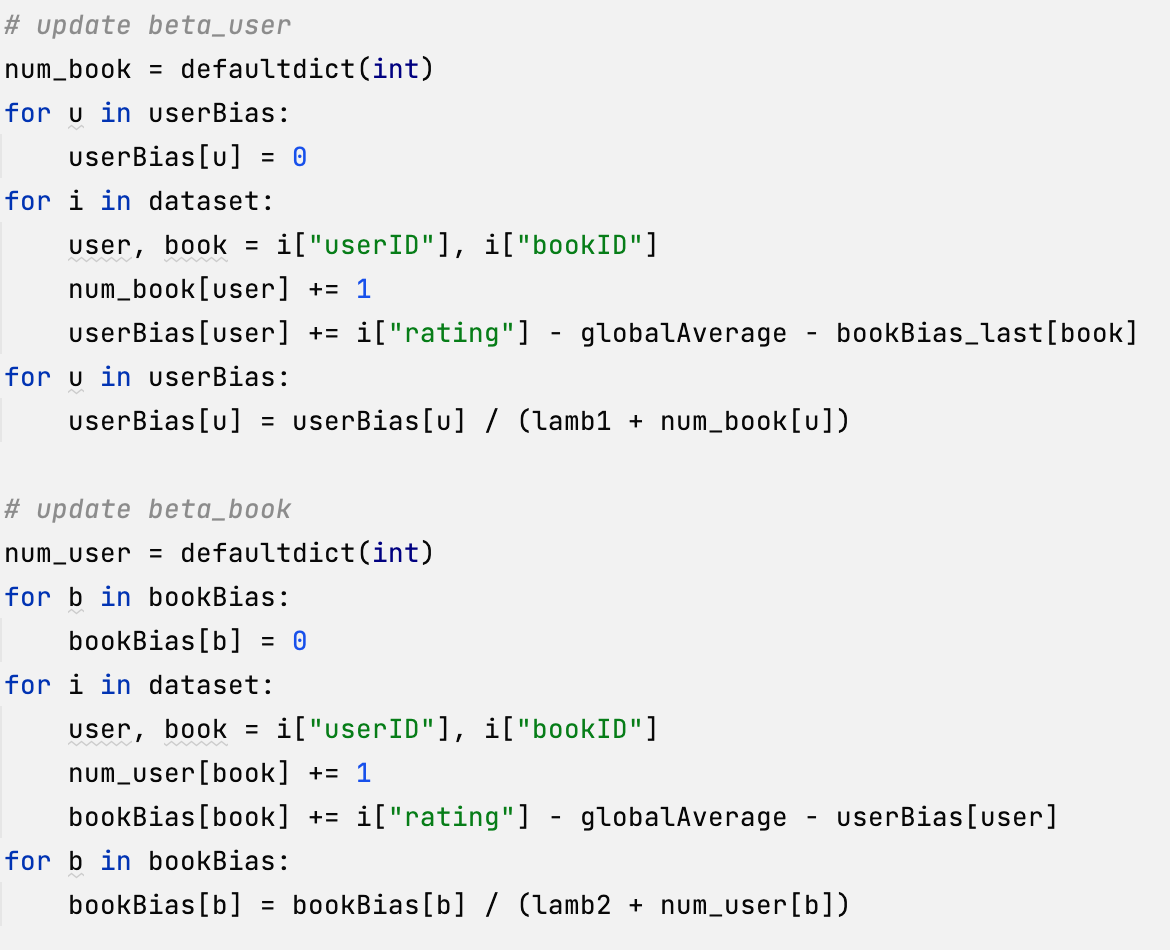


**updata alpha:**

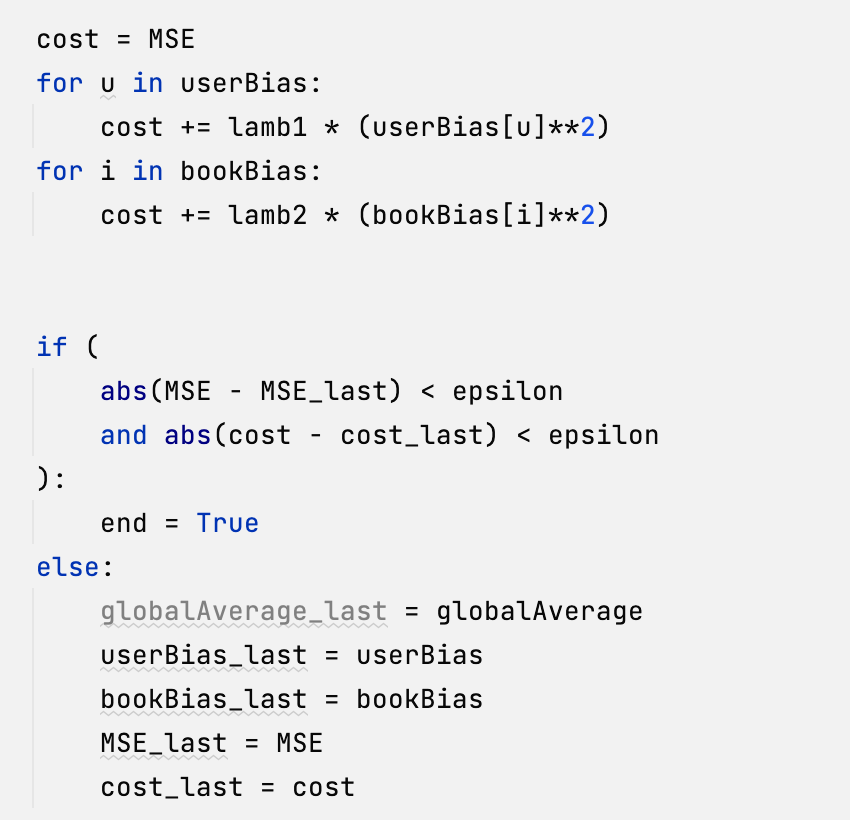
Here alpha = globalAverage:



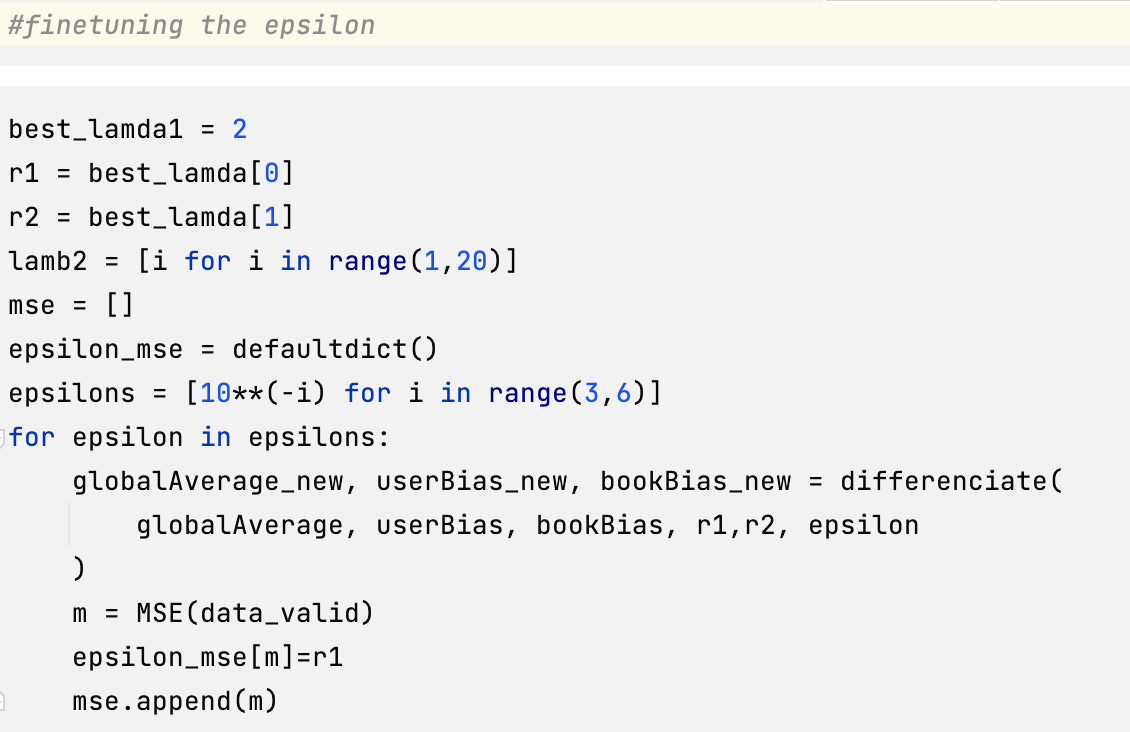
**updata beta:**



**End of iteration: set a bound**



**Fine-tune the bound**



Finally, use all the dataset to predict the rate.

I have also tried to use some SVD models to do it, but the results are not better than the latent factor one.