

# CSE 158/258, MGTA 461, DSC 256, Fall 2024: Assignment 1

## Instructions

In this assignment you will build **recommender systems** to make predictions related to book reviews from *Goodreads*.

Submissions will take the form of prediction files uploaded to gradescope, where their test set performance will be evaluated on a leaderboard. **Most of your grade will be determined by ‘absolute’ cutoffs; the leaderboard ranking will only determine enough of your assignment grade to make the assignment FUN.**

The assignment is due **Monday, Nov 18**, though make sure you upload solutions to the leaderboard regularly.

You should submit two files:

**writeup.txt** a brief, plain-text description of your solutions to each task; please prepare this adequately in advance of the submission deadline; this is only intended to help us follow your code and *does not need to be detailed*.

**assignment1.py** A python file containing working code for your solutions. The autograder *will not execute your code*; this file is required so that we can assign partial grades in the event of incorrect solutions, check for plagiarism, etc. Your solution should **clearly document which sections correspond to each task**. We may occasionally run code to confirm that your outputs match submitted answers, so **please ensure that your code generates the submitted answers.**<sup>1</sup>

Along with two files corresponding to your predictions:

**predictions\_Read.csv, predictions\_Rating.csv** Files containing your predictions for each (test) instance. The provided baseline code demonstrates how to generate valid output files.

This assignment should be completed **individually**. To begin, download the files for this assignment from: <http://cseweb.ucsd.edu/classes/fa24/cse258-b/files/assignment1.tar.gz>

## Files

**train\_Interactions.csv.gz** 200,000 ratings to be used for training. This data should be used for both 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.

**pairs\_Read.csv** Pairs on which you are to predict whether a book was read.

**pairs\_Rating.csv** Pairs (userIDs and bookIDs) on which you are to predict ratings.

**baselines.py** A simple baseline for each task, described below.

Please do not try to collect these reviews from Goodreads, or to reverse-engineer the hashing function I used to anonymize the data. Doing so will not be easier than successfully completing the assignment. **We will execute code for any solution suspected of violating the competition rules to confirm that it generates valid output; all code will be run through a plagiarism detector.**

## Tasks

You are expected to complete the following tasks:

**Read prediction** 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.

---

<sup>1</sup>Don’t worry too much about dependencies if importing non-standard libraries.

**Rating prediction** 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).

A competition page has been set up on gradescope to keep track of your results compared to those of other members of the class. The leaderboard will show your results on *half of* the test data, but your ultimate score will depend on your predictions across the *whole* dataset.

## Grading and Evaluation

This assignment is worth 22% of your grade. You will be graded on the following aspects. Each of the two tasks is worth 10 marks (i.e., 10% of your grade), plus 2 marks for the written report.

- Your ability to obtain a solution which outperforms the leaderboard baselines on *the unseen portion of* the test data (5 marks for each task). Obtaining full marks requires a solution which is substantially better than baseline performance.
- Your ranking for each of the tasks compared to other students in the class (3 marks for each task).
- Obtain a solution which performs well on *the seen portion of* the test data (i.e., the leaderboard). This is a consolation prize in case you overfit to the leaderboard. (2 mark for each task).

Finally, your written report should describe the approaches you took to each of the tasks. To obtain good performance, you should not need to invent new approaches (though you are more than welcome to!) but rather you will be graded based on your decision to apply reasonable approaches to each of the given tasks. You should generally get full marks for the report as long as it describes your solution. 1-2 paragraphs should be sufficient for each task (2 marks total).

## Baselines

Simple baselines have been provided for each of the tasks. These are included in 'baselines.py' among the files above. They are mostly intended to demonstrate how the data is processed and prepared for submission to gradescope. These baselines operate as follows:

**Read prediction** Find the most popular books that account for 50% of interactions in the training data. Return '1' whenever such a book is seen at test time, '0' otherwise.

**Rating prediction** Return the global average rating, or the user's average if we have seen them before in the training data.

Running 'baselines.py' produces files containing predicted outputs (these outputs can be uploaded to gradescope). Your submission files should have the same format.