Instructions

In this assignment you will build **recommender systems** to make predictions related to video game reviews from *Steam*.

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 20**, 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.**[1]

Along with two files corresponding to your predictions:

`predictions Played.csv`, `predictions Hours.csv` Files containing your predictions for each (test) instance (you should submit two of the above three files). The provided baseline code demonstrates how to generate valid output files.

To begin, download the files for this assignment from:
`https://cseweb.ucsd.edu/classes/fa23/cse258-a/files/assignment1.tar.gz`

Files

train.json.gz 175,000 instances to be used for training. This data should be used for both the 'play prediction' and 'time played prediction' tasks. It is not necessary to use *all* observations for training, for example if doing so proves too computationally intensive.

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

gameID The ID of the game. This is a hashed game identifier from Steam.

text Text of the user's review of the game.

date Date when the review was entered.

hours How many hours the user played the game.

hours transformed $\log_2(\text{hours}+1)$. **This transformed value is the one we are trying to predict.**

pairs Played.csv Pairs on which you are to predict whether a game was played.

pairs Hours.csv Pairs (userIDs and gameIDs) on which you are to predict time played..

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

Please do not try to collect these reviews from Steam, 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 run the code of any solution suspected of violating the competition rules, and you may be penalized if your code does produce your submitted solution**.

[1]Don't worry too much about dependencies if importing non-standard libraries.

Tasks

You are expected to complete the following tasks:

Play prediction Predict given a (user,game) pair from 'pairs_Played.csv' whether the user would play the game (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 played games and the other 50% do not.

Time played prediction Predict how long a person will play a game (transformed as $\log_2(\text{hours} + 1)$, for those (user,game) pairs in 'pairs_Hours.csv'. Accuracy will be measured in terms of the *mean-squared error* (MSE).

A competition page has been set up on Kaggle 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 outperforms the baselines 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 (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:

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

Time played prediction Return the global average time, 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.