# M.Sc. in Data Science
# Recommender Systems: 2022
# Term Project Description

The project will consist of two parts. Part 1 is about developing multiple Recommender Systems for Amazon products. Part 2 is about the application of Federated Learning (FL) in Recommender Systems.

**Team size:** This is a team project. You can form a team of up to 4 people, but not more than that. The total number of points for both parts is 60.

## PART 1: Recommender System for Amazon products - 30 points + 5 points Bonus (Step D.5)

### General background

You need to have a good understanding of the material in Units 3-5 of the Lectures.

### Dataset

Amazon product review data. https://nijianmo.github.io/amazon/index.html

Read the dataset description carefully and download one of the "small" 5-core category datasets. We recommend that you use the "Digital Music" 5-core dataset (http://deepyeti.ucsd.edu/jianmo/amazon/categoryFilesSmall/Digital_Music_5.json.gz) which consists of 169,781 reviews, but you are free to use a larger category dataset depending on your computational resources. Also, please download the respective "metadata" dataset for the category you chose. For the "Digital Music" products metadata, the download link is: http://deepyeti.ucsd.edu/jianmo/amazon/metaFiles2/meta_Digital_Music.json.gz

### Indicative Steps in your approach

Part 1 of the projects sonsits of 4 sub-parts (A-D). We will implement the following types of Recommender systems:

A. A **Content-based Recommender System** based on product metadata, in order to make relevant recommendations to users. Below is an indicative list of steps to follow:

Step A.1: Use Python and its libraries to pre-process and merge the datasets.

Step A.2: Similarly to Unit 3 of the Lecture slides on Content-based Recommender Systems, construct "item profiles" for the products that consist of multiple attributes from the products' metadata. You should at least use the "title" and "description" of each product as features utilizing TF-IDF scores, as well as one more metadata attribute such as "price", "salesRank", "also_buy", etc. You are free to use more metadata attributes to create "richer" item profiles.

Step A.3: Create two similarity matrices using the Jaccard and the Cosine similarity metrics to measure the similarity between products, based on the "item profiles" you have created. Which of the 2 similarity measures is more appropriate for your "item profiles"? (explain briefly in your report).

Step A.4: Find the top-5 products for each user based on her past ratings.

Step A.5: Use one of the two similarity matrices to make 5 product recommendations to each user, based on the products she liked in the past (do not forget to exclude already rated movies from the recommendation list).

Step A.6: Compare the recommendation lists with the top-rated products for 10 random users to validate whether your content-based RecSys indeed makes relevant recommendations (briefly discuss the validation results in your report).

B. A **Collaborative-Filtering (CF)** based Recommender System to predict product ratings, with the following indicative steps:

Step B.1: Use Python and its libraries to pre-process the datasets.

Step B.2: Utilize the Surprise library (http://surpriselib.com/) to test and compare 2 item-item CF methods, 2 user-user CF methods, and one SVD variant method, using RMSE and 5-fold cross-validation.

Step B.3: Discuss the validation results. Which model achieved the lowest RMSE?

Step B.4: Make 5 unseen product recommendations to each user based on the predicted ratings using the best-performing model.

C. A **Hybrid Recommender System** based on both systems A and B above to predict product ratings, with the following steps:

Step C.1: Use the Content-based RecSys you created in question A and the corresponding recommendation list for each user and combine it with the predicted rating recommendation list from B so as to create a final "*hybrid*" recommendation list for each user, consisting of 10 products. You are free to improvise on choosing how to combine/aggregate the two recommender systems, e.g. rank the recommendations into a single recommendation list for each user based on the average rank of the two standalone recommendations lists.

Step C.2: Briefly discuss your choices and the recommendations made for the same 10 random users from A. Did the recommendations change? Did they become more relevant?

D. A **Deep Learning (Deep Matrix Factorization)-** based Recommender System to predict product ratings. The steps here can be as follows:

Step D.1: Use Python and its libraries to pre-process the datasets and create training and test sets with an 80-20 ratio.

Step D.2: Use the Tensorflow library to create a Deep Matrix Factorization neural network that gets the user and item embeddings as an input and outputs the predicted rating.

Step D.3: Choose the appropriate parameters, e.g. loss function, for your model. Briefly explain why you made those choices in your report.

Step D.4: Compare two neural network variants, a Multilayer perceptron (with Dense layers) and a Convolutional Neural Network (CNN). Which one scores the lowest Root Mean Squared Error (RMSE) on the test set?

Step D.5 (bonus 5 points): Create a Neural Network (with any architecture you think is the most appropriate) that also includes the "item profile" you calculated in A for each product. So, the input feature vector now contains the user and item embeddings to calculate the latent factors, as well as the "item profile" features.
Do you see any performance increase compared to using just the user and item embeddings?

## PART 2: Federated Learning – 30 points + 5 points Bonus (Step 8)

We will implement a simple Recommender System based on Federated Learning (FL).

### General background

You need to have a basic understanding of the material in Unit 6 of the lecture set. In FL, the basic concept is that the datasets are distributed across users.

- Link https://en.wikipedia.org/wiki/Federated_learning contains useful information.
- See the link: https://blog.insightdatascience.com/explicit-matrix-factorization-als-sgd-and-all-that-jazz-b00e4d9b21ea

### Papers/material to study

a) https://arxiv.org/abs/1602.05629 - the method of Federated Averaging (FedAvg)
b) https://arxiv.org/abs/1812.06127 - the method of Federated Proximation (FedProx) – especially sections 1-3
c) https://arxiv.org/pdf/2004.04256.pdf - Federated Learning for Matrix Factorization.
d) https://www.tensorflow.org/federated/tutorials/custom_federated_algorithms_2 - a sample implementation of FedAvg.

### Datasets to use

There are two options. You can use any ONE (1) of the three datasets below, however the suggestion is to use Option 3 since we will see this dataset in the lab lectures.

- Option 1: Personality 2018 dataset from https://grouplens.org/datasets/personality-2018/. Download this dataset. We will definitely need the ratings.csv file. Read also the README.txt file. See whether/how you can use the other dataset (personality-data.csv).
- Option 2: The Book Crossings dataset. This is one of the datasets that (c) above uses. Download the dataset from: http://www2.informatik.uni-freiburg.de/~cziegler/BX/ (better in .csv format).
- Option 3: Movielens "latest small" dataset. https://grouplens.org/datasets/movielens/.

We will focus on the method "Latent factors (Matrix factorization) with baselines" to predict ratings of users for items. We will adjust the method to the FL setting.

**Indicative Steps in your approach**

Step 1: Formulate the optimization problem (similarly to the formulation in Unit 5). Your optimization problem will relate to the prediction error in the training dataset. Make sure you understand the unknowns of the problem, and what you are trying to optimize.

Step 2: Do pre-processing of the dataset if needed. Distinguish the training and test datasets.

Step 3: Decide on the environment on which you will implement your approach. There is more than one good solution for that. For example, Python and its packages (Jupiter Notebook, numpy etc) is good. Most probably, that will be more than enough.

Step 4: First, solve the optimization problem of Matrix Factorization with baselines FOR THE ENTIRE DATASET. Namely, assume that the entire dataset is all located at a single server. Use Stochastic Gradient Descent for this step. You can assume a small number of Latent Factors e.g. 2-4 for items and users. Try also with larger number of latent factors (e.g. 10 or 20). Do you see any difference?
Note: There are ready implementations for that. You can use code from implementations of these methods on the web, provided that you (i) always cite the source with the appropriate link/reference, (ii) will be able to explain its steps if needed during the final presentation.

Step 5: Divide the dataset into chunks (parts) and assign each part to a user. If needed, you can restrict the dataset to ~50-100 users, where each user has her own part of the dataset (own movies with their ratings).

Step 6: FL implementation. Implement the Federated Averaging (FedAvg) algorithm, proposed in https://arxiv.org/abs/1602.05629 for our recommender systems problem with Matrix Factorization and baselines. The nature of the optimization problem may result in some changes in the way FedAvg is designed and implemented. Paper (c) offers some good guidelines. However, you can choose not to follow the approach in that paper.

Step 7: Implement another FL algorithm, FedProx from in https://arxiv.org/abs/1812.06127. Do you see any difference? Why is that? Discuss.

Step 8 (Bonus 5 points): Implement FedRecon from https://arxiv.org/pdf/2102.03448.pdf . You will need to implement "Algorithm 1" in the paper. See 5.1.1 and C.1. Compare with FedAvg, FedProx.

Step 9: Decide on the evaluation metrics for Steps 6-8 and 4. These can be at least 2 of the metrics mentioned in papers (a)-(c) above, or some other metrics that you can think of yourself. Typical metrics for FL are training loss, test accuracy, and speed of convergence.

Step 10: Compare the results of Step 6 with those in Step 4. We are interested in the accuracy of rating prediction in the test dataset. These algorithms will differ in terms of the metrics in general.

**Project Deliverables**

- Project deliverables due: March 21, 2022. These are as follows:

- A ~20-page report for each team, describing the approach, findings, assumptions, choices made in the project, presenting results in plots, and providing ideas for future work. Details will follow.
- A 30-40-slide (~45min) presentation on the above. This will be presented at the end of the semester (March 21, 2021). All members of the team will present ~8-10mins each.
- Your implemented code on the above. If you have used code from other sources, clearly state only the source link in the report and the presentation. You do not need to include the code borrowed from other sources in your code.
- A single .zip file containing a) the report, b) the slide presentation, and c) the code (only the one implemented by you), will be uploaded in E-class (details on that will follow).

**Miscellanea**

- Help: This project will determine by itself your grade in this class. Therefore, in light of fairness, clarifications will be given only with regard to the questions themselves. Specific directions to a team on what to do, how to proceed with certain steps of the process, or whether something you do is correct or not, will not be provided.
- Work as a team: The size of the team can be up to 4 people. Therefore, it is important to choose your partners and to start the work as soon as possible. It is important for all to have a common basic understanding of the problem and approach. Splitting the work is good and in fact necessary to carry out this project. Clearly, it is not going to be easy to meet in person because of the pandemic but in the end, meeting and working online may be better and more efficient.
- Grading criterion: This is a class project, and as such, it is understandable (and expected) if in the end you do not achieve performance commensurate to that achieved in the research papers above. The grading will be performed on the basis of solutions that make sense, they work without bugs, and give satisfactory results. Also, the quality of the report, the quality/clarity of the presentation, and the answers to questions that may be asked during the presentation will shape the final grade.
- Novelty of the approach: Each of the two parts of the project is based on already existing work in the scientific literature. Specific guidelines for the steps are given so as to facilitate you in your work. However, the project is designed on purpose to be open-ended, giving some directions, but leaving certain decisions up to you. Thus, you are free to improvise and/or think of your own approach and algorithms.
- Use of material from the web etc. The area is very mature, and most probably you can find several implementations (esp. of FL) online. You can use material, provided that you cite the source and that you will be ready to explain it while presenting, if asked to.