Final Project

Diagnostic questions answers prediction

With matrix factorization algorithms

Introduction

Online education services, such as Khan Academy and Coursera, provide a broader audience with access to high-quality education. On these platforms, students can learn new materials by watching a lecture, reading, and talking to instructors in a forum

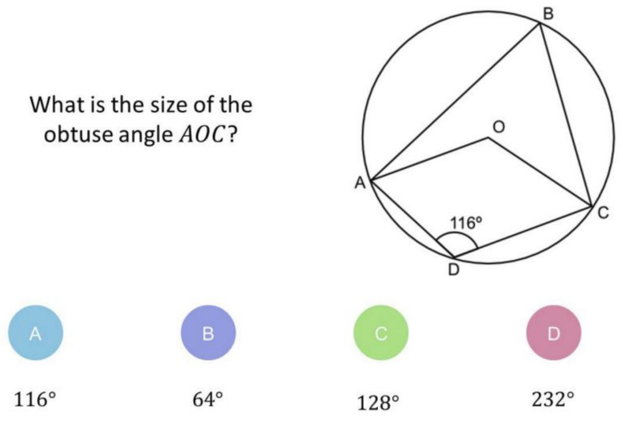
However, one disadvantage of the online platform is that it is challenging to measure students’ understanding of the course material. To deal with this issue, many online education platforms include an assessment component to ensure that students understand the core topics

(Taken from the final project CSC311 handout)

Diagnostic questions

The assessment component is often composed of diagnostic questions,

each a multiple choice question with one correct answer


What is the size of the obtuse angle AOC?

A 116°

B 64°

C 128°

D 232°

Figure 1: An example diagnostic question [1].

The diagnostic question is designed so that each of the incorrect answers highlights a common misconception. When students incorrectly answer the diagnostic question, it reveals the nature of their misconception and, by understanding these misconceptions, the platform can offer additional guidance to help resolve them

Settings for the final project

In this project, we built machine learning algorithms to predict whether a student can correctly answer a specific diagnostic question based on the student's previous answers to other questions and other students' responses

Each class member selected a set of algorithms trying to achieve the best result on the problem using their set of algorithms. We measured predictions with **accuracy** metric to compare the results

I have chosen **matrix factorization** that seek to find a low rank approximation to a sparse user-item interaction matrix

Matrix Factorization

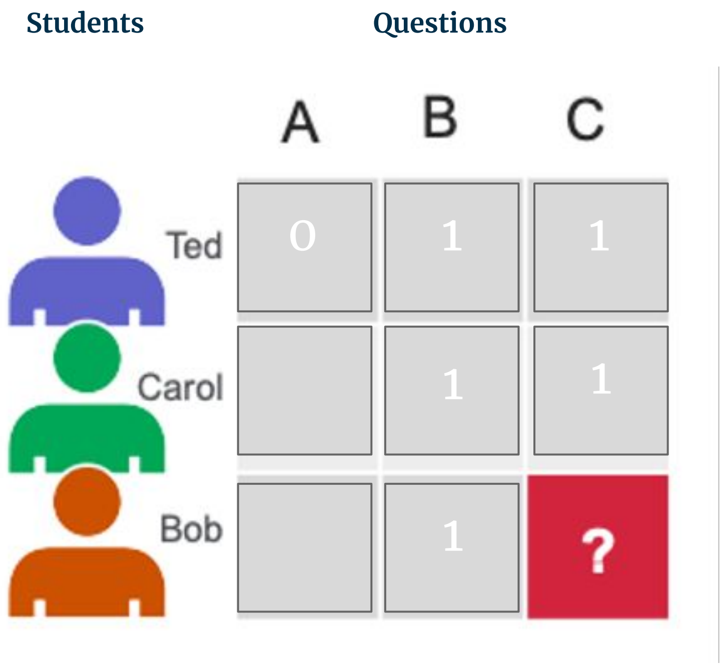
Students of the same type of group knowing/ missing one concept indicate that other students of this group might know/miss this concept too. These groups are based on latent factors

Algorithms:

- Singular Value Decomposition (SVD)
- Logistic Matrix Factorization (LMF)
- Non-negative Matrix Factorization (NMF)

Hyperparameters

- Number of latent features
- Regularization
- Learning Rate
- Appropriate initialization
- Loss function



Nature of data

- Binary
- Explicit
- With a big amount of missing values:
  only 5.9% of the entries are filled with values

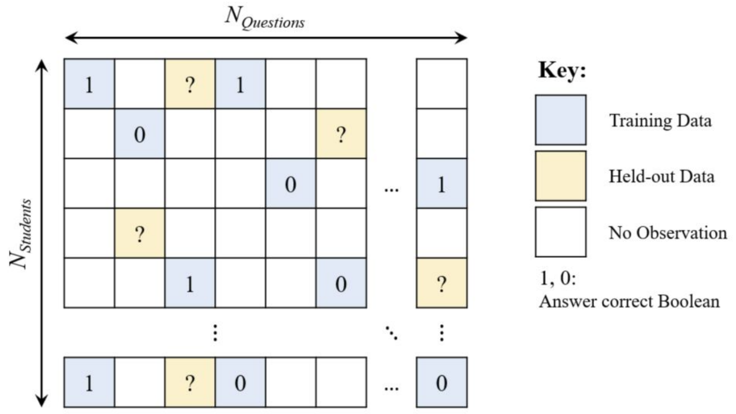
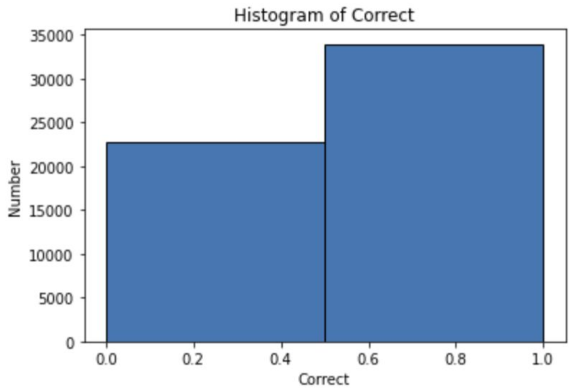


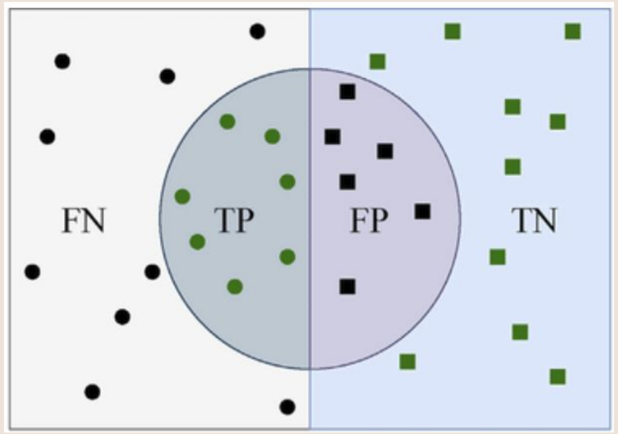
Figure 2: An example sparse matrix [1].



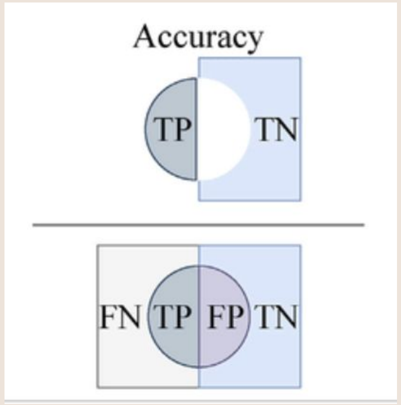
- Slightly unbalanced

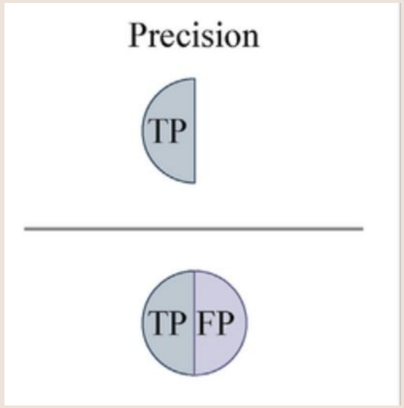
Metrics

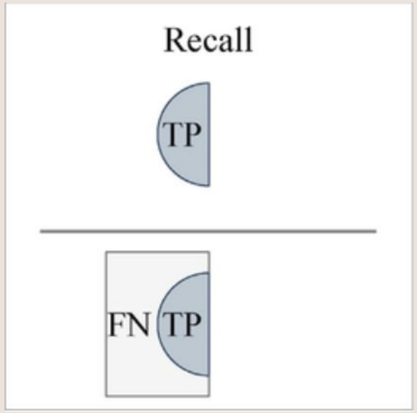
for binary classification



The metric for this project





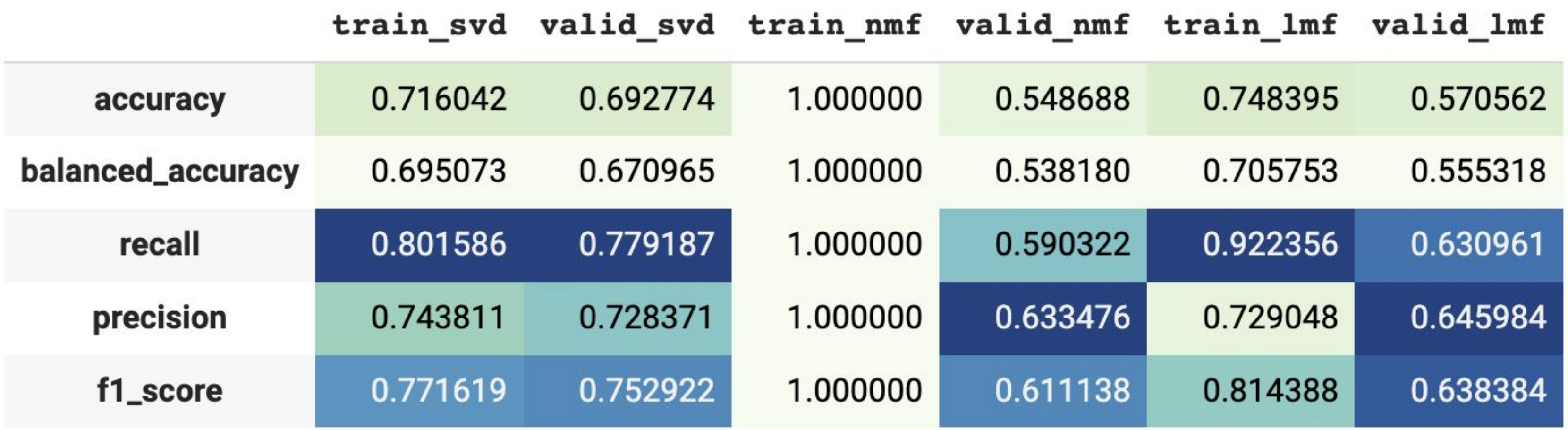


F1 Score

$$2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Baseline models

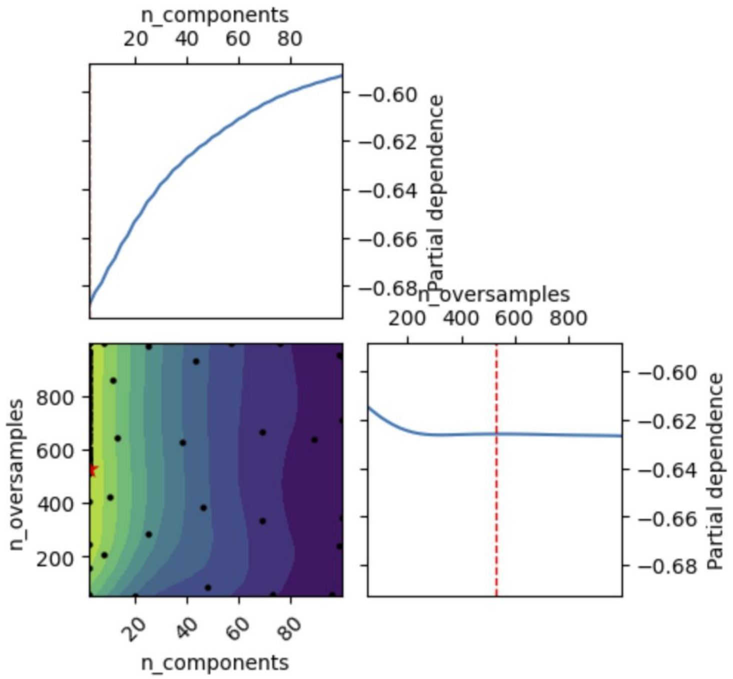
With default hyperparameters

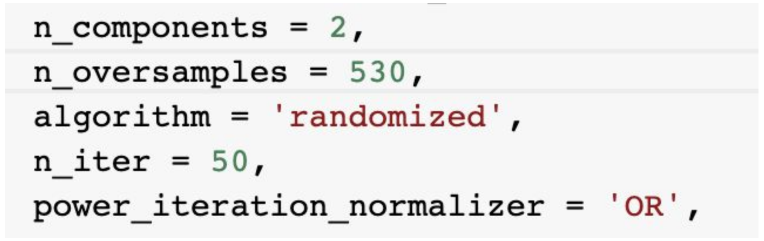


	train_svd	valid_svd	train_nmf	valid_nmf	train_lmf	valid_lmf
accuracy	0.716042	0.692774	1.000000	0.548688	0.748395	0.570562
balanced_accuracy	0.695073	0.670965	1.000000	0.538180	0.705753	0.555318
recall	0.801586	0.779187	1.000000	0.590322	0.922356	0.630961
precision	0.743811	0.728371	1.000000	0.633476	0.729048	0.645984
f1_score	0.771619	0.752922	1.000000	0.611138	0.814388	0.638384

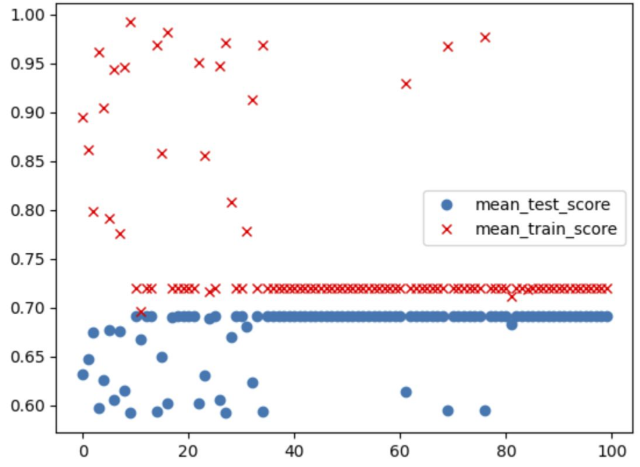
Hyperparameters turning. SVD

Bayesian optimization over hyper parameters





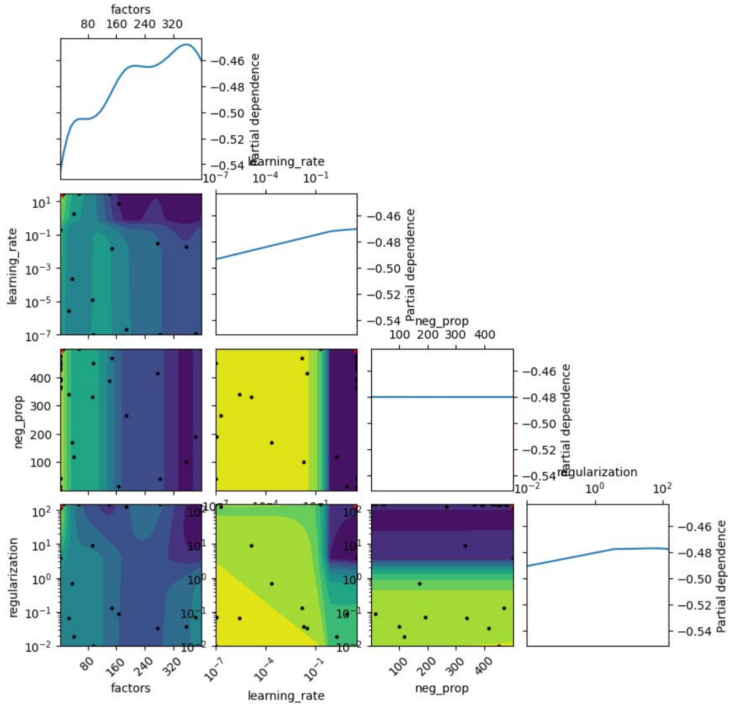
```
n_components = 2,
n_oversamples = 530,
algorithm = 'randomized',
n_iter = 50,
power_iteration_normalizer = 'OR',
```



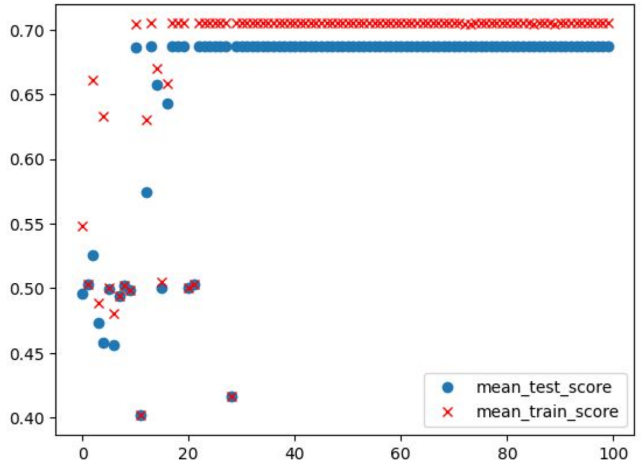
Number of iterations of bayesian optimization

Hyperparameters turning. LMF

Bayesian optimization over hyper parameters



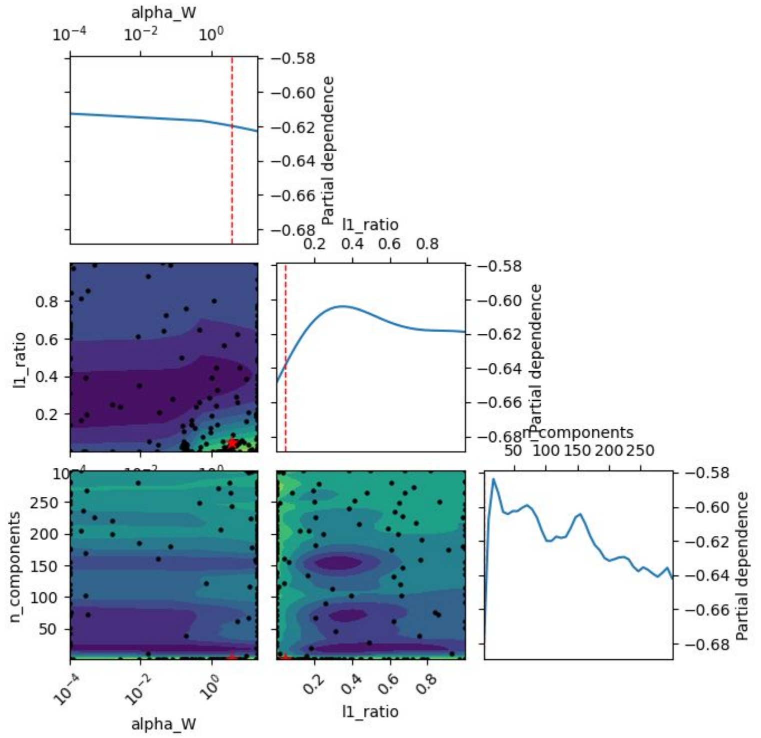
```
factors = 2,
regularization = 150,
learning_rate = 2.25,
neg_prop = 500,
iterations = 50,
```



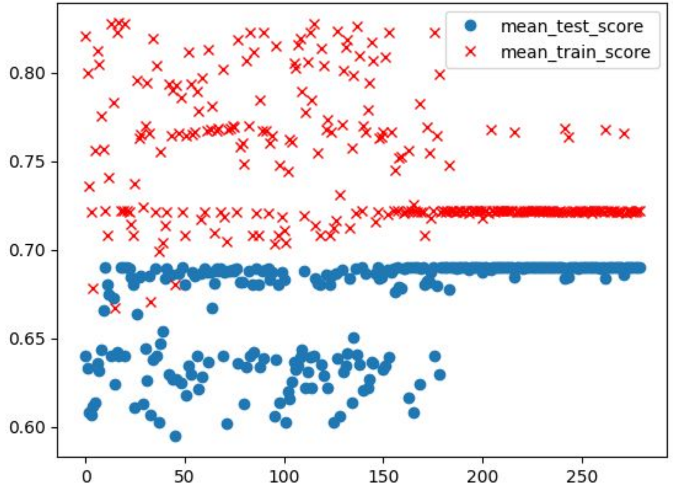
Number of iterations of bayesian optimization

Hyperparameters turning. NMF

Bayesian optimization over hyper parameters

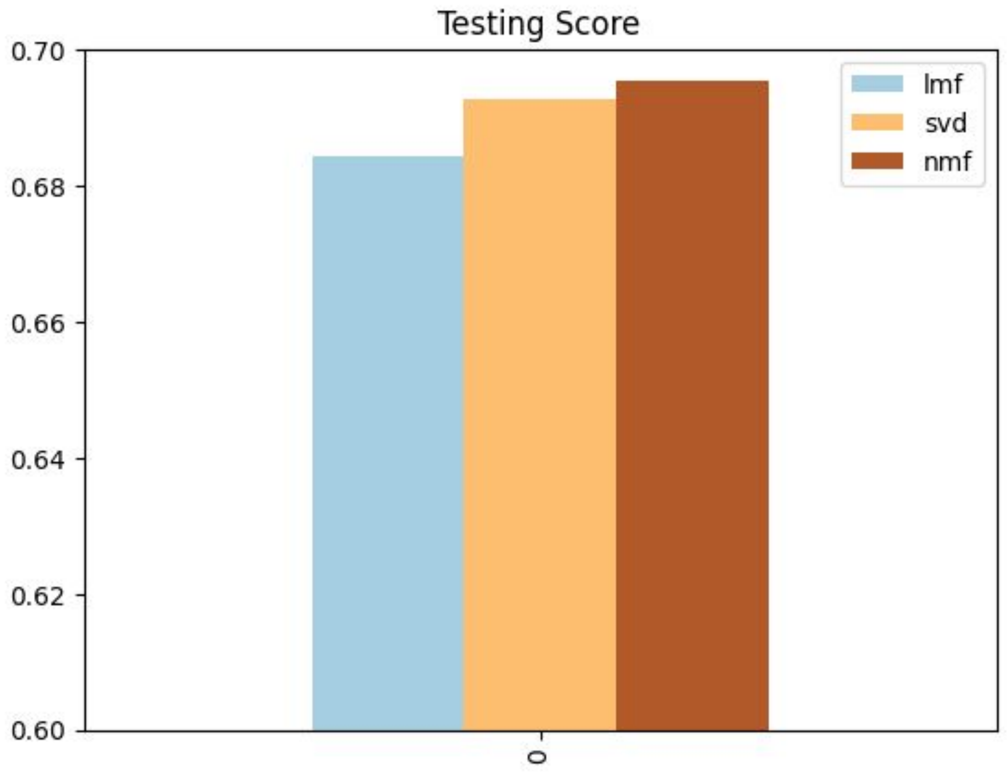


```
n_components = 2,
alpha_W = 3.622,
alpha_H = 0,
l1_ratio = 0.048,
init = 'random',
solver = 'mu',
beta_loss = 'kullback-leibler',
max_iter = 100000,
tol = 0.0001,
```



Number of iterations of bayesian optimization

Comparison of the results

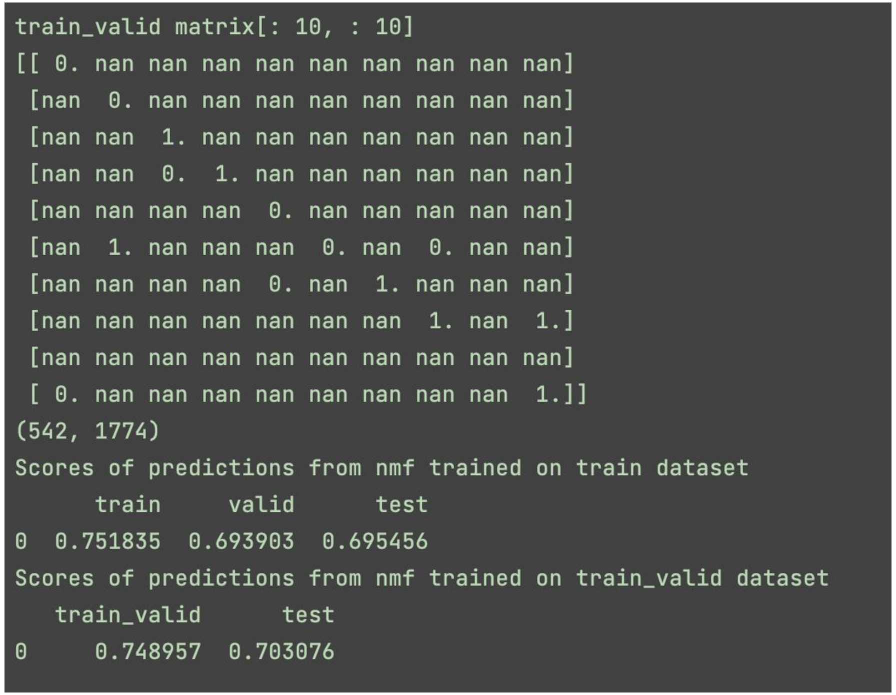


Result on NMF accepting missing values
Are better than previous results

NMF in principle can handle datasets with missing values by ignoring nans and focusing on the values present in a dataset while optimizes the lost function. However the sklean implementation of NMF didn’t support this case. I opened a feature request to handle nan in NMF - and the support has been added. I downloaded the source code and run the algorithm again in PyCharm. You can see the results on the right->

Model trained on a training dataset produces a consistent result but overfits the data

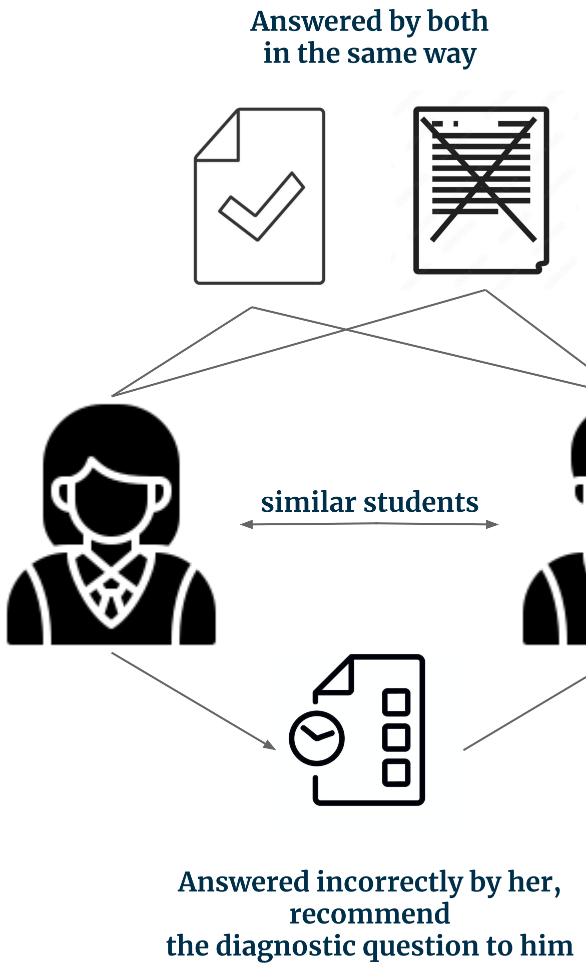
Model trained on training and validation datasets combined together produces a better result and less overfits the data. There more non zero entries we have, the better predictions



```
train_valid matrix[: 10, : 10]
[[ 0. nan nan nan nan nan nan nan nan nan]
 [nan  0. nan nan nan nan nan nan nan nan]
 [nan nan  1. nan nan nan nan nan nan nan]
 [nan nan  0.  1. nan nan nan nan nan nan]
 [nan nan nan nan  0. nan nan nan nan nan]
 [nan  1. nan nan nan  0. nan  0. nan nan]
 [nan nan nan nan  0. nan  1. nan nan nan]
 [nan nan nan nan nan nan nan  1. nan  1.]
 [nan nan nan nan nan nan nan nan nan nan]
 [ 0. nan nan nan nan nan nan nan nan  1.]]
(542, 1774)
Scores of predictions from nmf trained on train dataset
      train     valid      test
0  0.751835  0.693903  0.695456
Scores of predictions from nmf trained on train_valid dataset
   train_valid      test
0     0.748957  0.703076
```

Collaborative filtering

These predictions form the groundwork for many advanced customized tasks. For instance, using the predicted correctness, the online platform can automatically recommend a set of diagnostic questions of appropriate difficulty that fit the student's background and learning status


Answered by both in the same way
similar students
Answered incorrectly by her, recommend the diagnostic question to him

Collaborative part

Based on the information on answers from many students

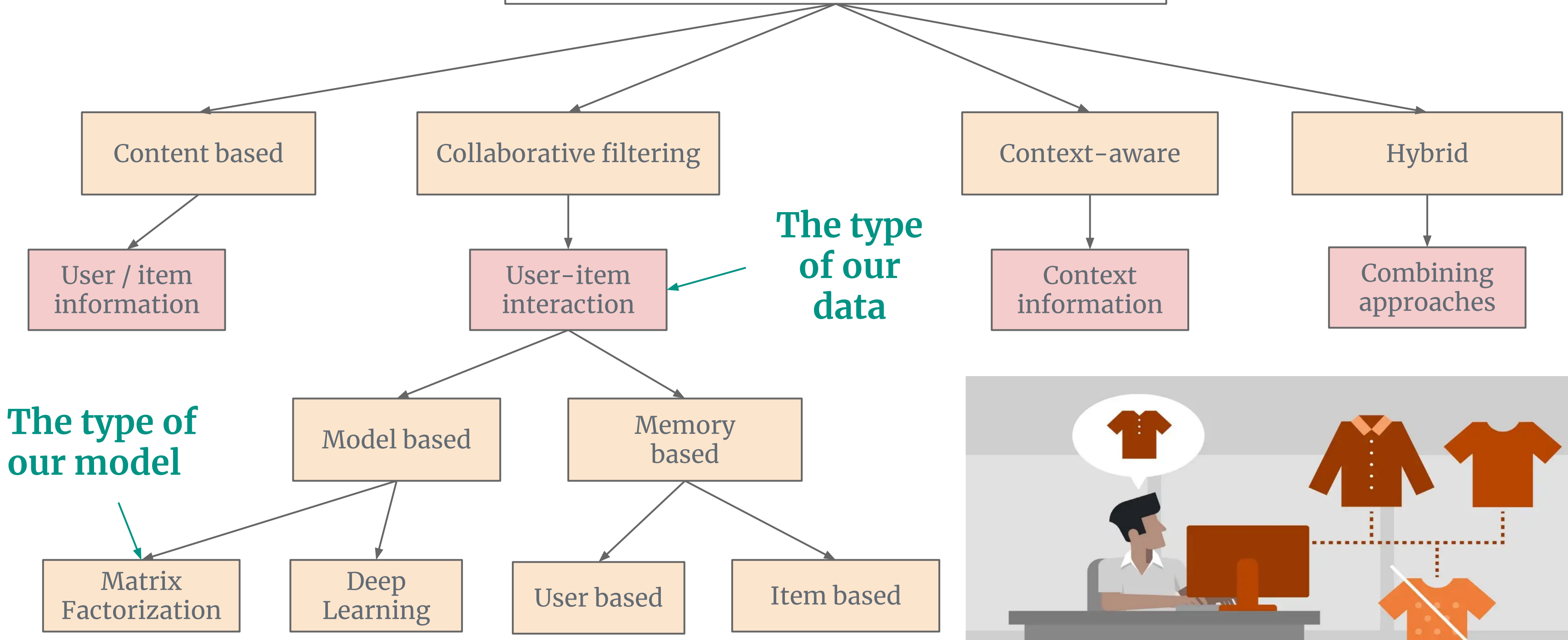
Filtering part

The algorithm recommends diagnostic questions

Framing the problem

As recommender system

Recommender system overview



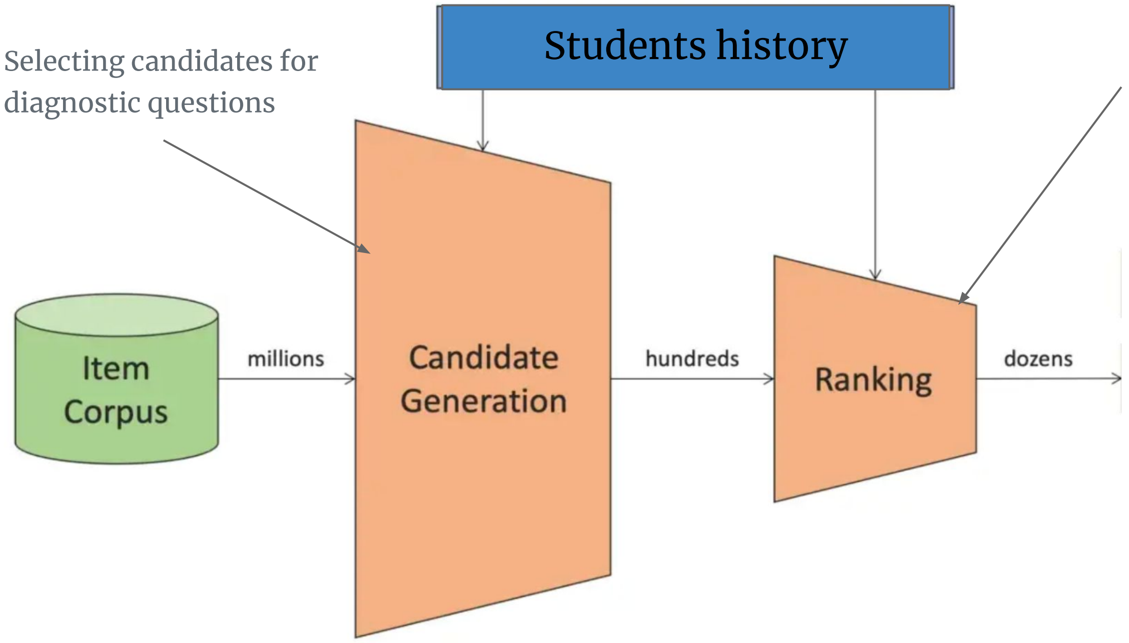
Accomplish recommendation in two steps

Done

Future steps

Selecting candidates for diagnostic questions

Sorting selected diagnostic questions in the order of expected discovery of the lack of knowledge



Question 1

Question 2

Question 3

.
.
.

2-stage Recommender System (inspired by YouTube)

Ranking

This project was focused on the correct / incorrect answer classification task however with the Matrix Factorization Algorithms we could perform the ranking task right away using metrics MAP@K, ARHR

Sequential recommendation

In real life students might improve their knowledge after taking additional courses so their answers done previously might not reflect their current state. We can improve our results by modeling the temporal dynamics of the data into sequential recommender system in the future

Hybrid recommender systems

Utilize question metadata, student demographic factors