CSC311Project

Lantian Zhang, Boxuan Fang

April 5, 2024

Part A

1 Collaborative filtering with k-Nearest Neighbor

(a)

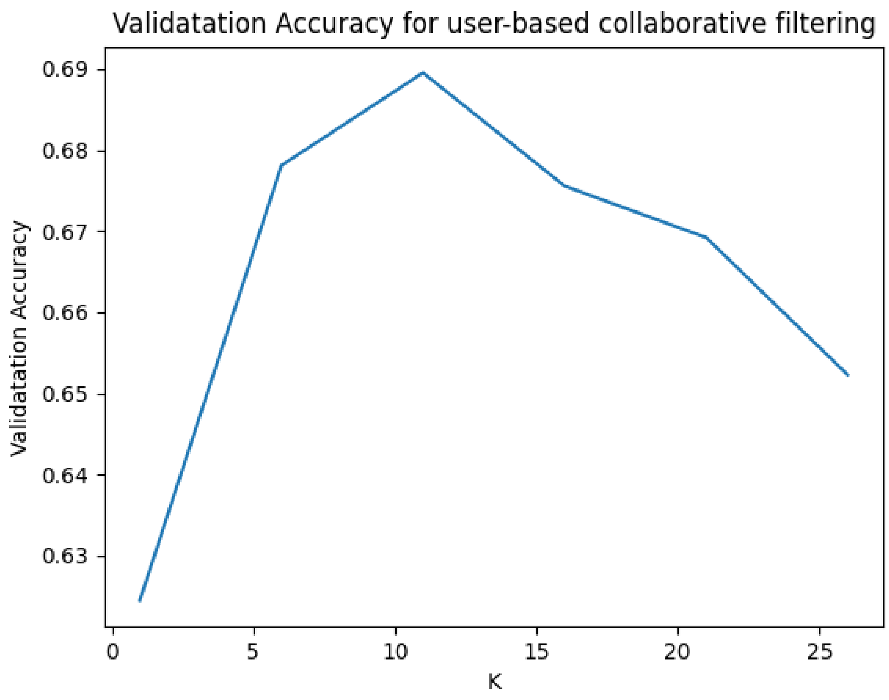

Figure 1: Validation Accuracy for user-based collaborative filtering

The k value we selected is 11. The test accuracy with k = 11 is 0.6841659610499576.

(b)

The underlying assumption on item-based collaborative filtering:

If question A is answered correctly and incorrectly by the other students the same as question B, specific students' correctness on question A matches that on question B.

(c)

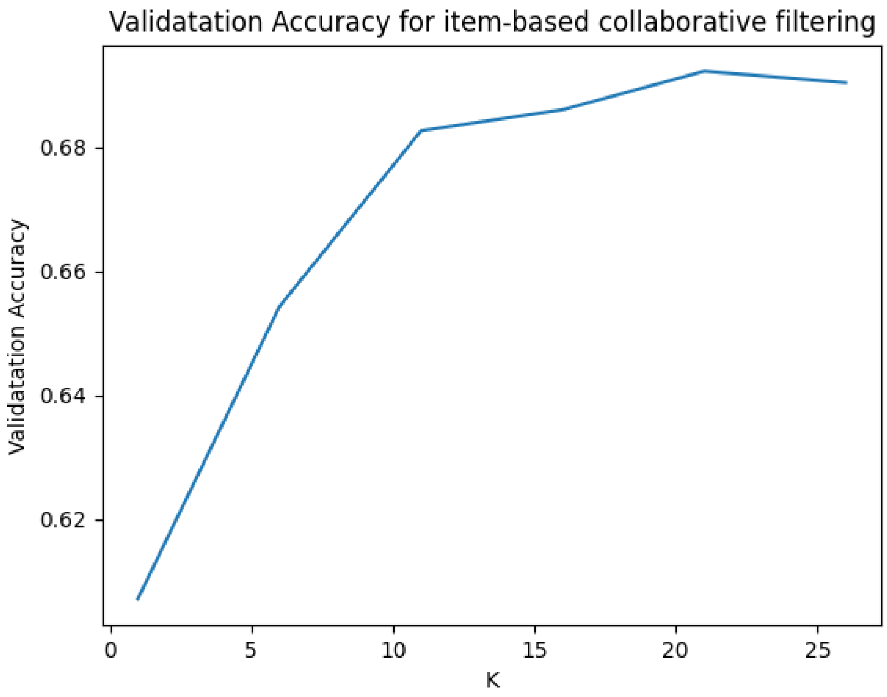

Figure 2: Validation Accuracy for item-based collaborative filtering

The k value we selected is 21. The test accuracy with k = 21 is 0.6816257408975445.

(d)

User-based collaborative filtering is better because it has a higher test accuracy.

(e)

Two potential limitations of kNN: (we included three limitations)

1. It can't make predictions for a student who has not answered any question or a question that has not been answered by any student because in this case, we don't know which data points are the k nearest neighbors.
2. It requires huge computation work when the training set is very large because kNN must compute the distances from one data point to all other ones to find the k-nearest neighbors for each prediction.
3. If the numbers of students and questions are very large, all data points are far away from each other in an extremely high-dimensional space. Then the closest point is not very predictive.

2 The Item Response Theory Model

(a)

Let N, M ∈ ℕ. Assume that there are N students and M questions in the training date set.

$$\log p(\mathbf{C}|\boldsymbol{\theta}, \boldsymbol{\beta}) = \sum_{i=1}^{N}\sum_{j=1}^{M}[c_{ij}(\theta_i - \beta_j) - log(1 + exp(\theta_i - \beta_j))]$$

$$\frac{\partial \log\, p(\mathbf{C}|\boldsymbol{\theta},\boldsymbol{\beta})}{\partial \theta_i} = \sum_{j=1}^{M}[c_{ij} - \frac{exp(\theta_i - \beta_j)}{1+exp(\theta_i - \beta_j)}]$$

$$\frac{\partial \log\, p(\mathbf{C}|\boldsymbol{\theta},\boldsymbol{\beta})}{\partial \beta_j} = \sum_{i=1}^{N}[-c_{ij} + \frac{exp(\theta_i - \beta_j)}{1+exp(\theta_i - \beta_j)}]$$

Let $\mathbf{A} = \boldsymbol{\theta}\mathbf{1}_{1\times M} - \mathbf{1}_{N\times 1}\boldsymbol{\beta}^T$.

$$\log\, p(\mathbf{C}|\boldsymbol{\theta},\boldsymbol{\beta}) = \mathbf{1}_{1\times N}(\mathbf{C}\odot\mathbf{A} - log(1+exp(\mathbf{A})))\mathbf{1}_{M\times 1}$$

$$\frac{\partial \log\, p(\mathbf{C}|\boldsymbol{\theta},\boldsymbol{\beta})}{\partial \boldsymbol{\theta}} = (\mathbf{C} - \frac{exp(\mathbf{A})}{1+exp(\mathbf{A})})\mathbf{1}_{M\times 1}$$

$$\frac{\partial \log\, p(\mathbf{C}|\boldsymbol{\theta},\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = (-\mathbf{C} + \frac{exp(\mathbf{A})}{1+exp(\mathbf{A})})^T\mathbf{1}_{N\times 1}$$

(b)

See the code in `item_response.py`.

(c)

The learning rate we selected is 0.025. The number of iterations we selected is 5.

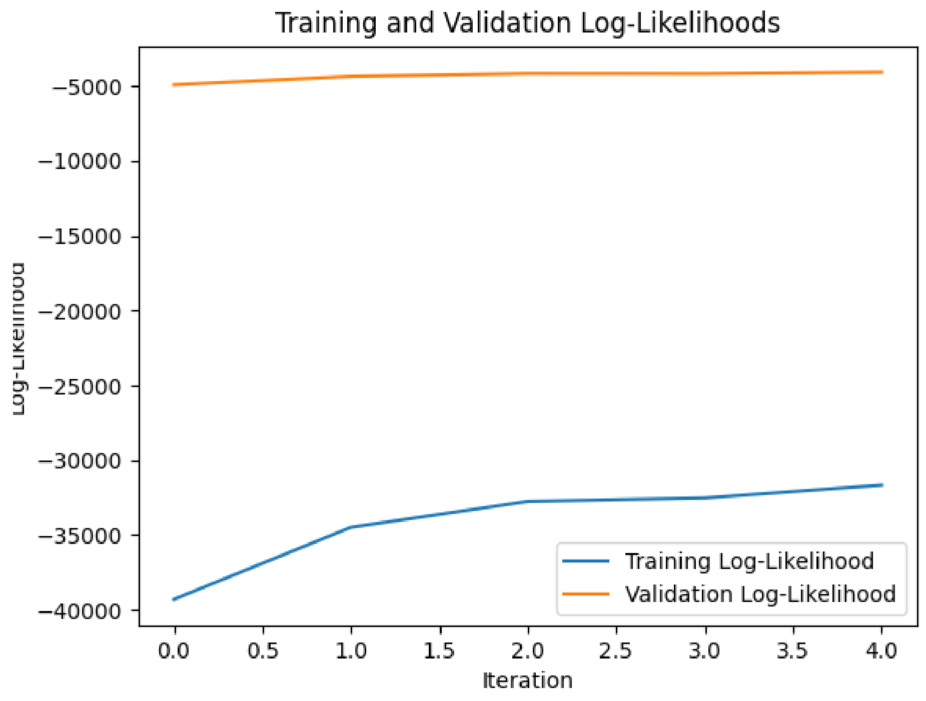

Figure 3: Training and Validation Log-Likelihoods over Iterations

The validation accuracy of our final model is 0.7095681625740897. The test accuracy of our final model is 0.7008185153824442.

(d)

We selected questions 1, 2, 3.

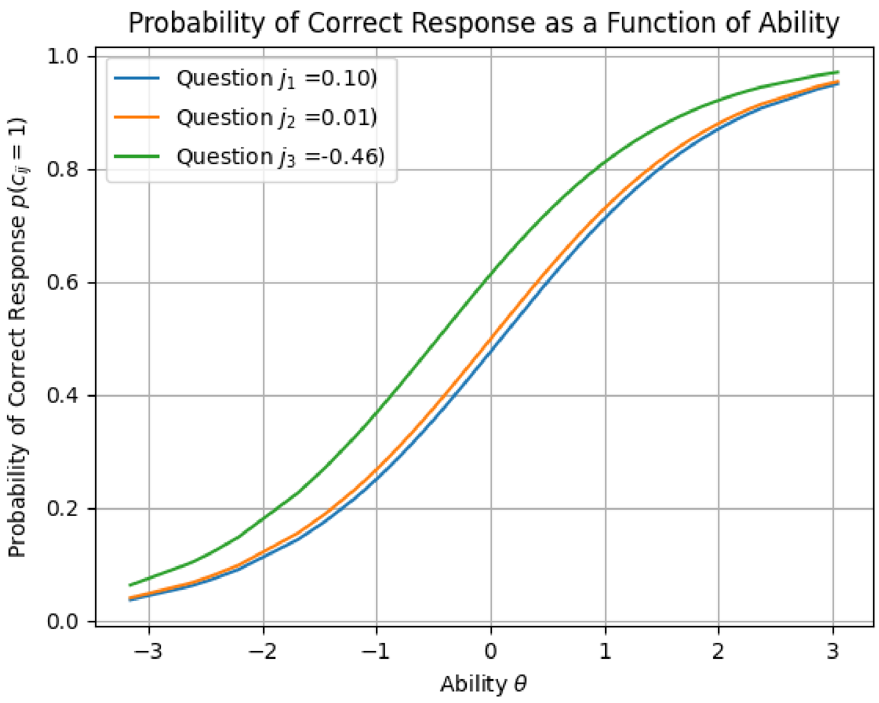

Figure 4: Probability of Correct Response as a Function of Ability

Comment on the shapes of the curves and what these curves represent:

The curve with a higher difficulty is above the one with a lower difficulty in the plot. It indicates that the harder the question is, controlling on student's ability, the smaller the probability that the student will answer it correctly.

Each curve is increasing. It indicates that by controlling the difficulty of the question, students with higher ability will have a greater chance of answering it correctly.

3 Matrix Factorization OR Neural Networks

We selected Matrix Factorization

(a)

The k value we selected is 9. The validation accuracy with k = 9 is 0.6613039796782387. The test accuracy with k = 9 is 0.6587637595258256.

(b)

One limitation of SVD for this task:

The methodology we used here for filling in the missing entries before implementing SVD is that we take the mean of observed entries for every column, which is the probability that question j will be answered correctly by an average student, and fill in the missing entries using this column mean. However, each entry corresponds to a specific student and a specific question. For each missing entry, the column mean used to fill in the entry does not consider the ability level of the corresponding student, so the filling entries will be biased.

(c)

See the code in `matrix_factorization.py`.

(d)

The k value we selected is 2. The learning rate we selected is 0.02. The number of iterations we selected is 10.

(e)

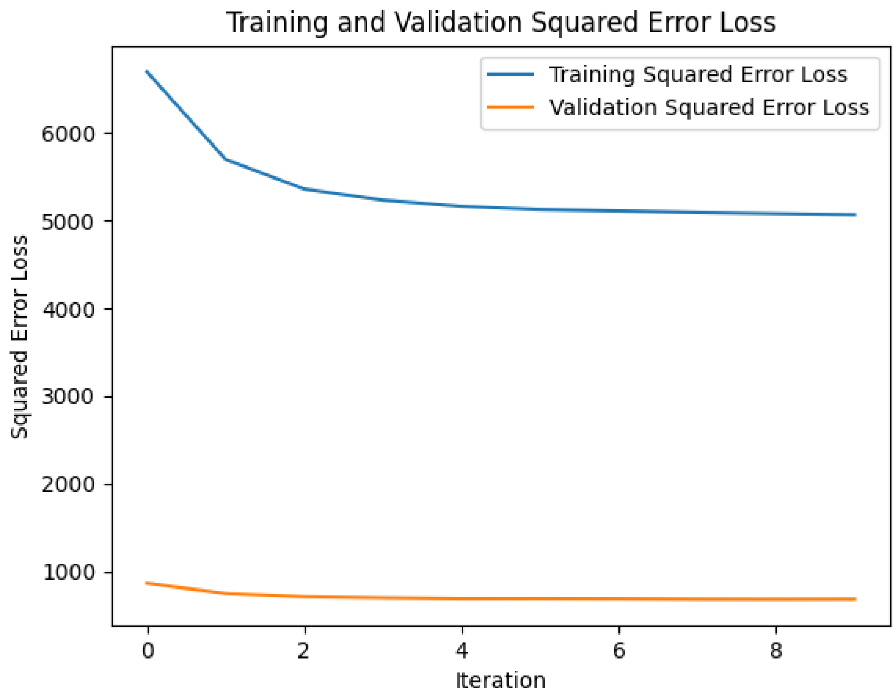

Figure 5: Training and Validation Squared Error Loss over Iteration

The validation accuracy for the final model is 0.7071690657634773. The test accuracy for the final model is 0.698278295230031.

Part B

1. Formally Describe My Modified Algorithm

We are focusing on improving the singular-value decomposition (SVD) model's performance by modifying the way we fill in the missing entries in the sparse matrix before actually performing the SVD algorithm.

In the Baseline SVD model, The way it fills in the missing entries in the sparse matrix is that it takes the mean of the observed entries for each column and fills in the missing entries (by column) using each of these values, meaning that it calculates the probability of correctness for each question $P(Question_j = 1)$ and fill in the missing entries using these probabilities. For example, if our sparse matrix for training looks like this:

	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6
S_1	0	1	?	0	?	1
S_2	?	1	1	?	0	0
S_3	?	?	?	0	0	0
S_4	1	?	?	1	?	?
S_5	?	0	?	0	1	?

Table 1: Example Sparse Matrix

This is an example of 6 questions and 5 students with missing entries in the sparse matrix. Based on the baseline SVD algorithm, we fill in the sparse matrix by its column means:

	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6
S_1	0	1	1	0	0.33	1
S_2	0.5	1	1	0.25	0	0
S_3	0.5	0.67	1	0	0	0
S_4	1	0.67	1	1	0.33	0.33
S_5	0.5	0	1	0	1	0.33

Table 2: Example Sparse Matrix Filled

However, in this case, using the baseline algorithm, we are treating every student the same, regardless of whether or not some of them might be a good student who is able to perform better than others in these bunch of diagnostic questions. The same logic for the opposite, where there might be some students who are unfamiliar with these contents and will perform worse than the others. Considering this situation, we need to improve the methodology for filling in the missing entries so that we can somewhat get a more accurate matrix to further perform the SVD algorithm.

Let's first reconsider the probability of answering j^{th} question correct , i.e., $P(Q_j = 1)$. The higher value means that there is a higher chance that students will answer this j^{th} question correctly. So, we can think of the complexity of question j = β_j (0 for the hardest and 1 for the easiest) and $\beta_j = P(Q_j = 1)$. Also, on the other side, if we calculate the row means across every question, we are calculating the probability that student i will answer questions correctly $P(S_i = 1)$, which can be thought as the student i^{th} ability θ_i (0 for untrained student and 1 for well-trained student) and $\theta_i = P(S_i = 1)$. Given this, we can come up with the new parameter μ_{ij}, where:

$$\mu_{ij} = \frac{\beta_j + \theta_i}{2}$$

to mathematically describe the concurrent effect of the ability of student i and the difficulty of question j on the correctness of student i's answer to question j.

For example: For student 3 and question 1, the missing entry will be filled by 0.25 instead of 0.5.

$$\mu_{3,1} = \frac{0.5 + 0}{2} = 0.25$$

For a well-trained student (with high θ) and a hard question (with low β), the μ_{ij} we calculated here is able to produce a value that is in between, which makes sense since a good student might also answer wrong in a tricky question. Whereas for a not-that-well-trained student (with somehow low θ) and an easy question (with high β), he might have a chance to answer it correctly since it is an entry-level question.

We then use the μ_{ij} to fill in the missing entries to get a non-sparse matrix to perform the SVD algorithm. In this way, we can get a more 'accurate' complete matrix with less biased estimations for missing entries. This is why we expect our modified algorithm to perform better.

The code of the modified version of SVD is in `modified_svd.py`.

2. Figure or Diagram

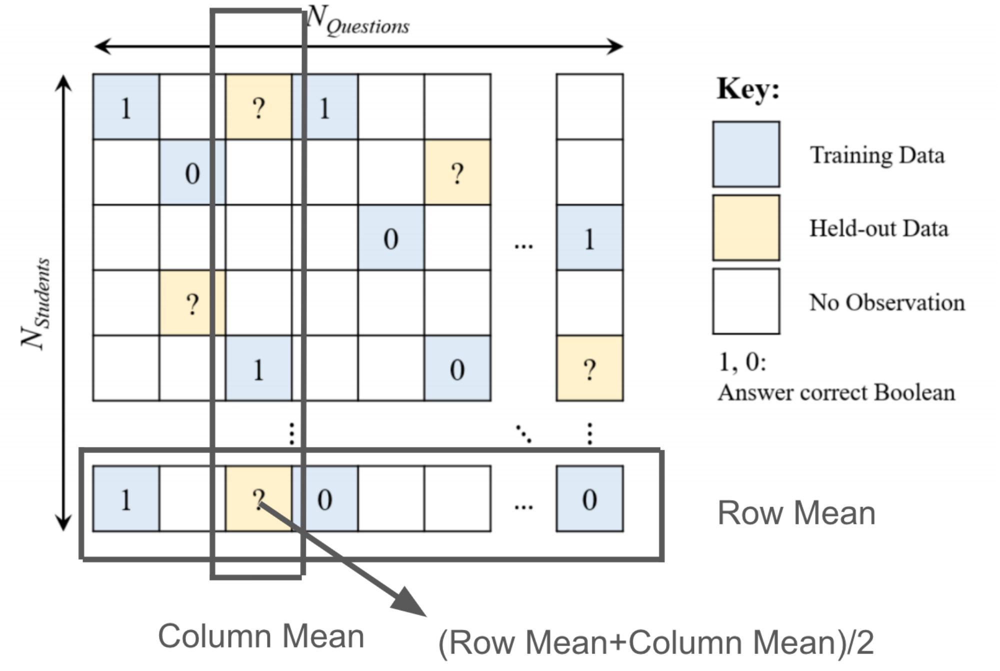

Figure 6: Modified Algorithm Illustration

This Figure is from the instruction of CSC 311 Winter 2024 Final Project.

3. My Model versus Baseline Models

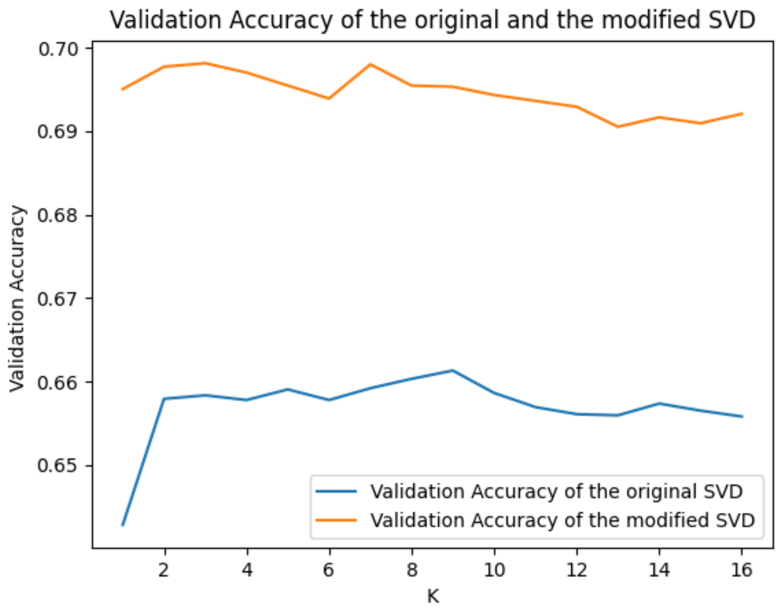

Figure 7: Validation Accuracy of the original and the modified SVD

According to Figure 2, which shows the modified model's validation accuracy versus the baseline model's validation accuracy, we can see that for the same k values, the modified model improved the validation accuracy by approximately 5 percent on average. Moreover, based on the validation accuracy, we tuned our hyperparameter k = 3 as it produces the highest validation accuracy.

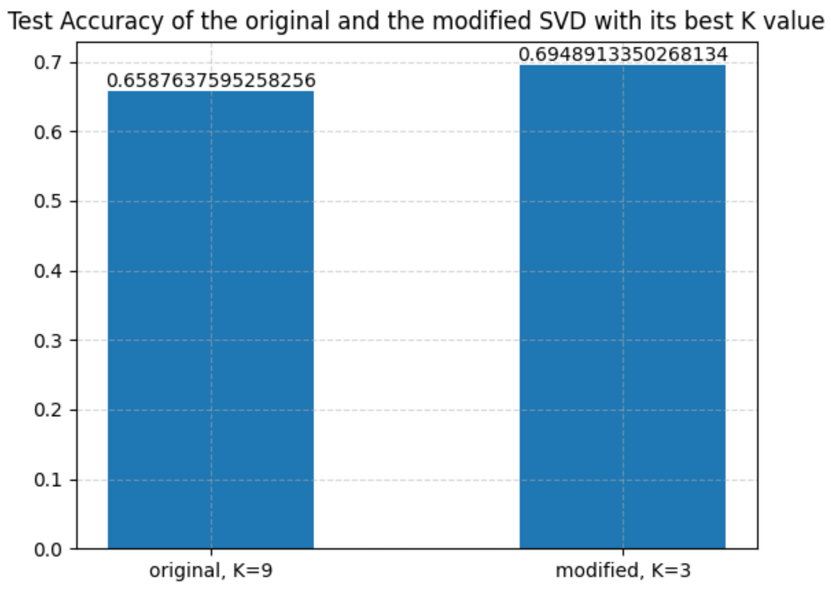

Figure 8: Test Accuracy of the original and the modified SVD with its best K value

According to Figure 3, Compared with the baseline model, the modified model improved the test accuracy by approximately 4 percent, which means that the modification to the baseline algorithm improved the prediction accuracy overall.

So our model performs better than the baseline one.

4. Explain My Model's Performance

The modified model has more predictive power than the baseline model since we produced a somehow 'more accurate' matrix that is able to reveal the complexity of the questions and students' abilities better. As a result, by performing the SVD algorithm afterward, it is likely to produce more accurate predictions.

5. Analyze One Limitation of My Model

Here we try to find a mathematical model to describe the relationship among the ability of a student, the difficulty of a diagnostic question, and the probability that this student will answer this question correctly. We simply model it with the average of the student's problem-solving ability and the difficulty of the question, which is used to quantify the concurrent effect of the two factors on the correctness of the question, but it might not be accurate. Because we don't have related expertise in this field, we cannot reasonably describe the dependency between the problem-solving ability of a student and the difficulty of a diagnostic question. The lack of proper interpretation ability may directly affect the logicality of the model.

Since we use the means to make estimations, under the setting that the sparse matrix is very sparse, where it has too many missing entries, the algorithm might perform very poorly as it cannot gather information effectively from both the student's side and the question's side and estimating with means will generate large errors.

If this scenario happens, the priority solution is to try our best to collect more data that can be used for training to further perform the algorithm more effectively.

However, for such a platform, we are more likely to have sparse matrices because most students only write a part of the questions, so this limitation exists.

Part C

Each team member's contributions: Lantian completed Question 1 of Part A, calculation and tuning hyper-parameters in Questions 2, 3 of Part A, and Questions 2, 3, 5 of Part B, code style and formatting; Boxuan completed Questions 2, 3 of Part A and Questions 1, 3, 4 of Part B. We confirm that we have average contributions.