CSC311 Project Final Report

Group Members:
Zixuan Zeng 1008533419
...
...

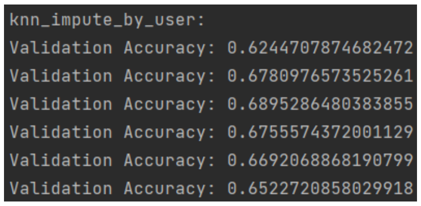
August 7, 2024

Note: Contributions and LLM are at the end of this report.

Part A

Q1 - KNN

1.(a)



```
knn_impute_by_user:
Validation Accuracy: 0.6244707874682472
Validation Accuracy: 0.6780976573525261
Validation Accuracy: 0.6895286480383855
Validation Accuracy: 0.6755574372001129
Validation Accuracy: 0.6692068868190799
Validation Accuracy: 0.6522720858029918
```

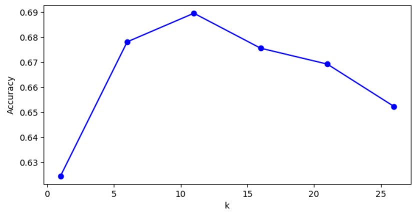
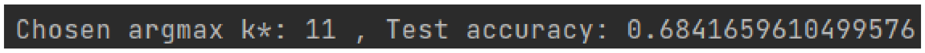


Figure 1: Accuracy vs k for KNN Impute by User

1.(b)

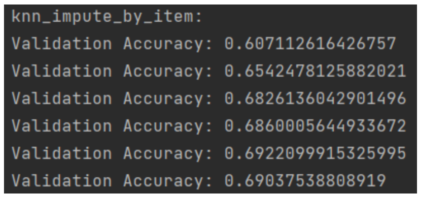


```
Chosen argmax k*: 11 , Test accuracy: 0.6841659610499576
```

Figure 2: Test accuracy with k*

1.(c)

The underlying assumption is that if question A is answered similarly by many students as question B, A's predicted response from specific students matches that of question B.



```
knn_impute_by_item:
Validation Accuracy: 0.607112616426757
Validation Accuracy: 0.6542478125882021
Validation Accuracy: 0.6826136042901496
Validation Accuracy: 0.6860005644933672
Validation Accuracy: 0.6922099915325995
Validation Accuracy: 0.69037538808919
```

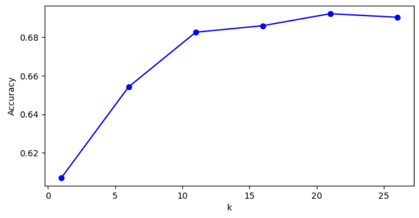
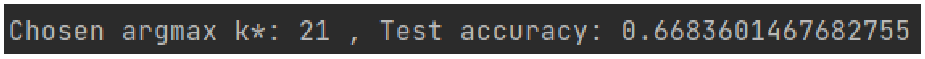


Figure 3: Accuracy vs k for KNN Impute by Item



```
Chosen argmax k*: 21 , Test accuracy: 0.6683601467682755
```

Figure 4: Test Accuracy with k*

1.(d)

The test accuracy for the user-based method (0.6842) is higher than that for the item-based method (0.6684). Therefore, the user-based collaborative filtering method performs better than the item-based collaborative filtering method in this case.

1.(e)

- Computationally expensive. KNN practically has no training process. With large datasets, as the number of students/questions grow, the time required to compute the distances and to identify the nearest neighbors at test time grows significantly.

- Curse of Dimensionality. When the sparse_matrix has too many missing values, it's hard to find good nearest neighbors, since most points will be about the same distances. This affects the prediction accuracy.

Q2 - IRT

2.(a)

$$p(c_{ij} = 1|\theta_i, \beta_j) = \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} = \sigma(\theta_i - \beta_j)$$

log-likelihood:

$$\log p(C|\theta, \beta) = \sum_i \sum_j [c_{ij}(\theta_i - \beta_j) - \log(1 + \exp(\theta_i - \beta_j))]$$

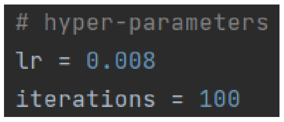
The derivative of the log-likelihood with respect to θ_i:

$$\frac{\partial \log p(C|\theta, \beta)}{\partial \theta_i} = \sum_j [c_{ij} - \sigma(\theta_i - \beta_j)]$$

The derivative of the log-likelihood with respect to β_j:

$$\frac{\partial \log p(C|\theta, \beta)}{\partial \beta_j} = \sum_i [-c_{ij} + \sigma(\theta_i - \beta_j)]$$

2.(b)



```
# hyper-parameters
lr = 0.008
iterations = 100
```

Figure 5: Hyper-Parameters

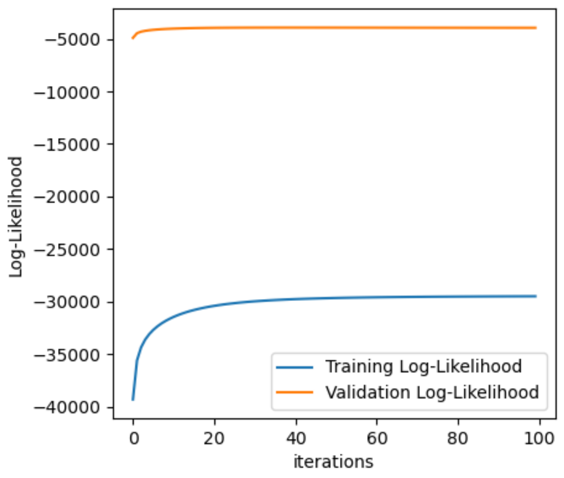
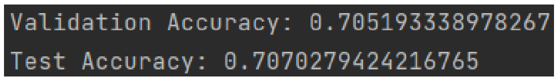


Figure 6: Training and Validation Log-Likelihoods vs Iteration

2.(c)



```
Validation Accuracy: 0.705193338978267
Test Accuracy: 0.7070279424216765
```

Figure 7: Final Validation & Test Accuracy

2.(d)

The three curves are all in S shape, as the sigmoid function. They represent the probability of correct responses as a function of student ability θ.
It shows that students with a high ability have a high probability of answering correctly.
Also, question with a high difficulty is skewed to the right, meaning it has a lower probability of being answered correctly.

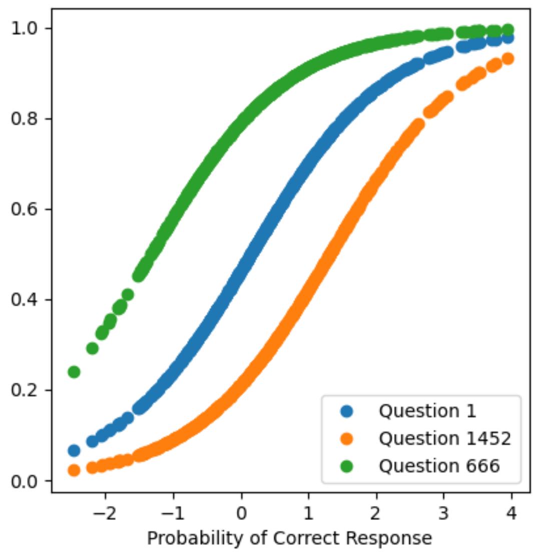
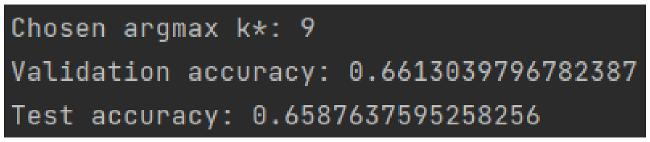


Figure 8: Probability of Correct Response vs Theta

Q3 - (i) Option 1: Matrix Factorization

3(i).(a)



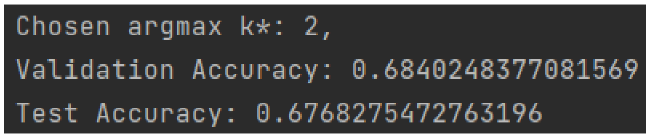
```
Chosen argmax k*: 9
Validation accuracy: 0.6613039796782387
Test accuracy: 0.6587637595258256
```

Figure 9: SVD: Final Validation & Test Performance with chosen k

3(i).(b)

Filling the missing values with averages or zeros does not accurately reflect the data's underlying structure. There will be loss or distorsion of information.

3(i).(c)(d)(e)



```
Chosen argmax k*: 2,
Validation Accuracy: 0.6840248377081569
Test Accuracy: 0.6768275472763196
```

Figure 10: ALS: Final Validation & Test Performance with chosen k

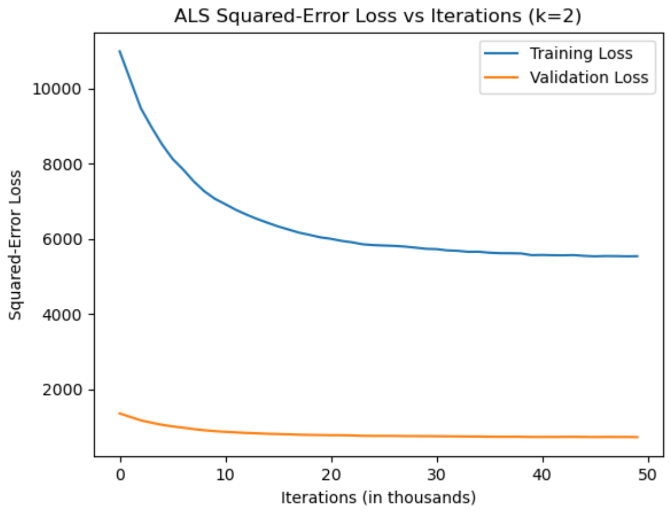


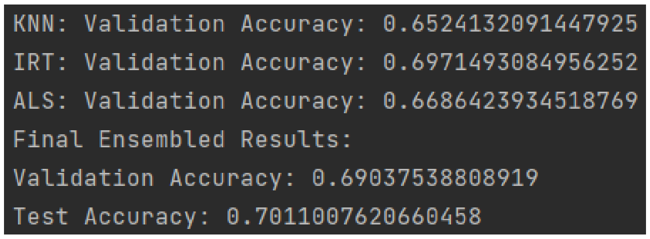
Figure 11: ALS: Squared-Error Loss vs Iterations

Q4 - Ensemble

Ensemble Process:

1. For each separate model, create different subsets of the training data of the same size of the original training data by sampling with replacement.
2. Train KNN, IRT, ALS. Report validation accuracy of each model. Use the models to generate predictions for validation and test data.
3. Using the three sets of predictions, calculate the mean of predictions.
4. Evaluate the ensembled prediction for final ensembled validation and test accuracy.

The ensemble does achieve better performance than individual models based on the final accuracy results. Because ensemble approach reduces the variance by combining the three models, which improved generalization.



```
KNN: Validation Accuracy: 0.6524132091447925
IRT: Validation Accuracy: 0.6971493084956252
ALS: Validation Accuracy: 0.6686423934518769
Final Ensembled Results:
Validation Accuracy: 0.69037538808919
Test Accuracy: 0.7011007620660458
```

Figure 12: Individual Model & Final Ensemble Accuracies

Part B

Note: This part consists of an ensemble of two models, created by two team members with separate algorithm analysis.

Modified ALS - Zixuan Zeng

Q1 - Formal Description:

- Base Model: Matrix Factorization with ALS
  The base model of Matrix Factorization with ALS captures some nuanced properties of users and questions.
  However, it initializes both matrices U and Z with values selected uniformly at random between 0 and $\frac{1}{np.sqrt(k)}$, for applying SGD to fit data into the model for the derivation of the k principal components.
  The randomness in latent factor initialization can lead to poor prediction quality and convergence to local minima.
  Also, This method lacks interpretability. Since we have no knowledge over the desired demensionality of PCA, along with the non-deterministic nature of the algorithm, the argmax k* fluctuates significantly.

- Modified ALS:
  My proposed modified model combines ALS with content-based approach to handle the existing problems.
  It incorporates the question metadata, using the subjects of each question as additional factors.
  Introduce new matrices: $U_s \in \mathbb{R}^{n \times subjects}$ and $Z_s \in \mathbb{R}^{m \times subjects}$, where U_s is also initialized by sampling uniformly at random, but Z_s is initialized to a sparse matrix with entries filled using question-subjects metadata, set at $\frac{1}{np.sqrt(num\_subjects)}$

Prediction given by:

$$R_{ij} \approx U^T Z + U_s^T Z_s$$

Squared Error Loss:

$$L = \frac{1}{2} \sum_{(n,m) \in O} (C_{nm} - (U^T Z + U_s^T Z_s))^2$$

Update Rules for SGD:

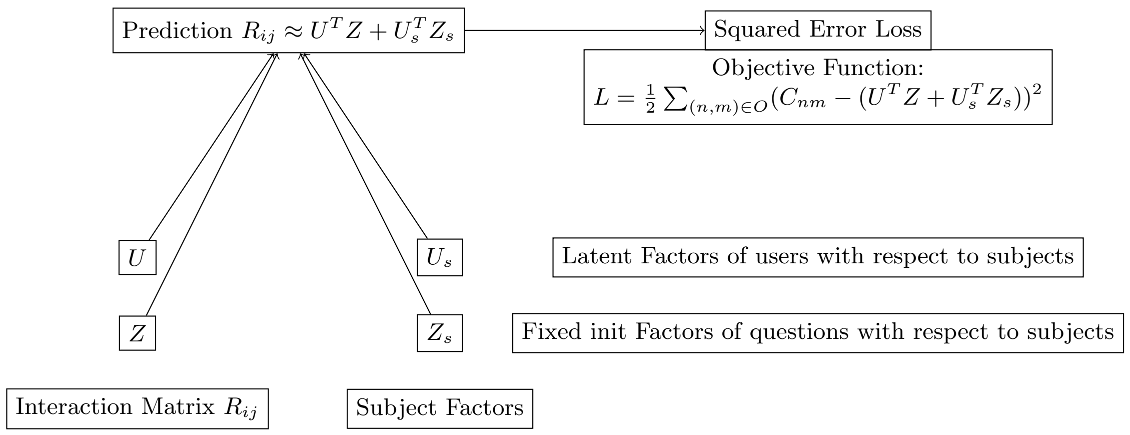
$$U_i \leftarrow U_i - \alpha \frac{\partial L}{\partial U_i} = -(C_{ij} - (U^T Z + U_s^T Z_s))Z_j$$

$$Z_j \leftarrow Z_j - \alpha \frac{\partial L}{\partial U_i} = -(C_{ij} - (U^T C + U_s^T Z_s))U_i$$

$$U_{si} \leftarrow U_{si} - \alpha \frac{\partial L}{\partial U_{si}} = -(C_{ij} - (U^T Z + U_s^T Z_s))Z_{sj}$$

$$Z_{sj} \leftarrow Z_{sj} - \alpha \frac{\partial L}{\partial Z_{sj}} = -(C_{ij} - (U^T Z + U_s^T Z_s))U_{si}$$

Q2 - Figure or Diagram:



Prediction $R_{ij} \approx U^T Z + U_s^T Z_s$
Squared Error Loss
Objective Function: $L = \frac{1}{2}\sum_{(n,m)\in O}(C_{nm} - (U^T Z + U_s^T Z_s))^2$
U
U_s
Z
Z_s
Latent Factors of users with respect to subjects
Fixed init Factors of questions with respect to subjects
Interaction Matrix R_{ij}
Subject Factors

Figure 13: Diagram of the Modified ALS Model showing the interaction between matrices U and Z, U_s and Z_s, the prediction formula, and the squared error loss function.

Q3 - Comparison / Demonstration:

Proposed hypothesis:

The modified ALS improves prediction accuracy due to better optimization strategies: incorporating new metadata as factors that influence the predictions.

Since the new algorithm without this feature is just the base ALS model, we will just test against the base ALS model using separate argmax hyperparameters.

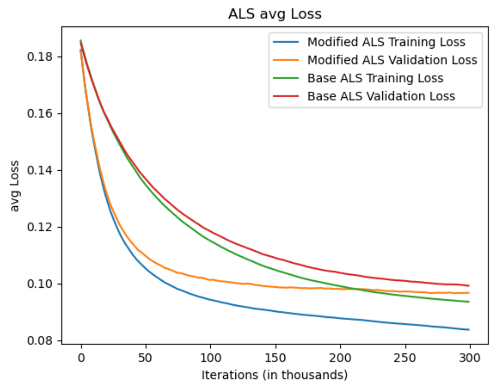
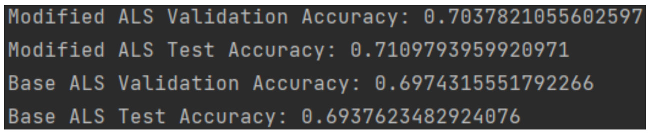


Figure 14: Average Loss vs Iterations: Modified ALS vs Base ALS



```
Modified ALS Validation Accuracy: 0.7037821055602597
Modified ALS Test Accuracy: 0.7109793959920971
Base ALS Validation Accuracy: 0.6974315551792266
Base ALS Test Accuracy: 0.6937623482924076
```

Figure 15: Validation & Test Accuracy: Modified ALS vs Base ALS

As seen in the plot, the modified ALS (as blue and yellow lines) has a lower squared loss error (averaged by len(train_data)).

It also has better validation accuracy of 0.704 and test accuracy of 0.711, compared to the base ALS model's validation accuracy of 0.697 and test accuracy of 0.694.

Therefore the modified ALS does have a better optimization.

Q4 - Limitations:

- The modified model assumes that subject metadata and latent factors capture all relevant aspects of user-question interactions, which may not always be the case.

- The reason is that this model emphasizes on the importance and correct catagorization of subject metadata and correctness of turning user-question interactions into latent factors. This method overlooks other important factors such as student metadata, contexual info, etc...

- To extend the current implementation to mitigate these limitations, we could integrate additional features or context-aware factors into the model to capture more nuanced interactions.

Hybrid KNN - Hao Liu

Q1 - Formal Description:

The Hybrid KNN model combines both user-based and item-based K-Nearest Neighbors (KNN) algorithms to improve the accuracy of predicting students' responses to diagnostic questions. Below is the detailed definition and methodology of the hybrid KNN model.

Formula Definition:

1. User-based KNN Imputation:
   Given a sparse matrix, where x_{ij} represents the response of student i to question j (0 for incorrect, 1 for correct, and NaN for missing), we can use userbased KNN to fill in the missing values. For each missing value x_{ij}, we find the k most similar students to student i and use their answers to fill in the missing value. Let N_i represent the k nearest neighbors of student i. The imputed value is calculated as:
   $$\hat{x}_{ij}^{user} = \frac{1}{k} \sum_{u \in N_i} x_{uj}$$

2. Item-based KNN Imputation:
   Similarly, we can use item-based KNN to fill in the missing values. For each missing value x_{ij}, we find the k most similar questions to question j and use their responses to fill in the missing value. Let N_i represent the k nearest neighbors of question j. The imputed value is calculated as:
   $$\hat{x}_{ij}^{item} = \frac{1}{k} \sum_{v \in N_j} x_{vj}$$

3. Hybrid Imputation:
   The final imputed value is a weighted combination of the user-based and item-based imputations:
   $$\hat{x}_{ij}^{hybrid} = \alpha \hat{x}_{ij}^{user} + (1 - \alpha) \hat{x}_{ij}^{item}$$
   where α is a user-defined weight parameter to balance the user-based and item-based imputations.

Algorithm Pseudocode:

Hybrid KNN Imputation Algorithm

Algorithm 1 Hybrid KNN Imputation

```
Require: $\mathbf{X}$: Sparse matrix with missing values
Require: $k_{\text{user}}$: Number of neighbors for user-based KNN
Require: $k_{\text{item}}$: Number of neighbors for item-based KNN
Require: $\alpha$: Weight for user-based KNN
Ensure: $\hat{\mathbf{X}}$: Imputed matrix
1: Initialize $KNN_{\text{user}}$ with $k_{\text{user}}$ neighbors
2: Initialize $KNN_{\text{item}}$ with $k_{\text{item}}$ neighbors
3: $\mathbf{X}_{\text{user}} \leftarrow KNN_{\text{user}}$.fit_transform($\mathbf{X}$)        ▷ User-based KNN imputation
4: $\mathbf{X}_{\text{item}} \leftarrow KNN_{\text{item}}$.fit_transform($(\mathbf{X}^T)^T$)    ▷ Item-based KNN imputation
5: $\hat{\mathbf{X}} \leftarrow \alpha\mathbf{X}_{\text{user}} + (1-\alpha)\mathbf{X}_{\text{item}}$    ▷ Combine user-based and item-based
   imputations
6: return $\hat{\mathbf{X}}$
```

Figure 16: Algorithm Pseudocode

Q2 - Figure or Diagram:

The provided graph shows the validation accuracy of the hybrid KNN imputation algorithm as a function of the number of neighbors k. The analysis focuses on understanding the performance trends and determining the optimal k value for the hybrid KNN model.

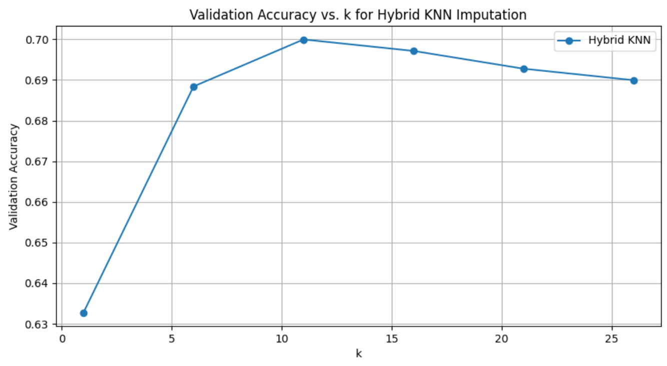


Figure 17: Hybrid KNN Imputation Results

Observations:

1. Accuracy Improvement with Increasing k: For small values of k (specifically k = 1), the validation accuracy is relatively low (approximately 0.63).

As k increases from 1 to 11, there is a significant improvement in validation accuracy, reaching a peak of approximately 0.70 at k = 11.

2. Optimal k: The highest validation accuracy is observed at k = 11. This indicates that using 11 neighbors provides the best balance between capturing sufficient information from similar users and items while avoiding overfitting to noise.

3. Decreasing Accuracy for Larger k: After k = 11, the validation accuracy begins to decrease gradually. This decline suggests that using too many neighbors may introduce less relevant or noisy data, which negatively impacts the model's performance.

Q3 - Comparison / Demonstration:

Through experiments, we compared the validation accuracy and test accuracy of the hybrid KNN model with the baseline models (user-based KNN and item-based KNN). The comparison was performed using different values of k (number of neighbors) to evaluate the performance of each model. Below are the observations and analysis based on the results.

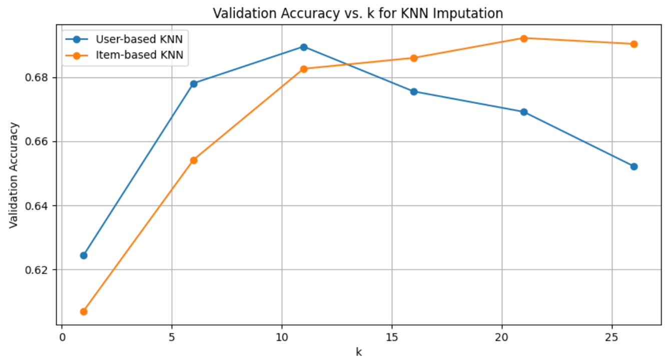


Figure 18: Validation Accuracy vs k for KNN Imputation

1. Comparison of Validation accuracy
The graph shows the validation accuracy for user-based KNN and item-based KNN as k varies.

    - User-based KNN:
    The validation accuracy starts at approximately 0.62 for k=1 and increases as k increases. The highest validation accuracy of 0.689 is observed at k=11. Beyond k=11, the accuracy starts to decline, indicating potential overfitting or introduction of noise from less similar users.

- Item-based KNN:
  The validation accuracy starts at approximately 0.61 for k=1 and steadily increases as kkk increases. The highest validation accuracy of 0.692 is observed at k=21. Beyond k=21, the accuracy remains relatively stable, suggesting that item-based KNN is less sensitive to the number of neighbors compared to user-based KNN.
- Hybrid KNN:
  The hybrid KNN model achieves the highest validation accuracy of 0.70 at k=11, which is higher than the best validation accuracy of both user-based and item-based KNN models. This demonstrates that combining user-based and item-based imputations provides a more accurate imputation.

2. Comparison of Test Accuracy
   The test accuracy further validates the performance of the models.
   - User-based KNN:
     The best test accuracy of 0.684 is observed at k=11.
   - Item-based KNN:
     The best test accuracy of 0.681 is observed at k=21.
   - Hybrid KNN:
     The hybrid KNN model achieves a test accuracy of 0.692 at k=11, confirming that the hybrid approach is more effective in predicting missing values compared to using only user-based or item-based KNN.
3. Experimental Design for Hybrid KNN Performance Validation
   To validate the hybrid KNN model's performance, we designed experiments with different kkk values and weight combinations (α):
   - Varying k Values:
     We experimented with k values ranging from 1 to 26 to determine the optimal number of neighbors.
   - Varying Weight Combinations:
     We tested different weight combinations (α) to balance user-based and item-based imputations, e.g. α=0.3, 0.5, 0.7. The results indicated that the hybrid KNN model with 1k=11 and α=0.5 provided the best performance in terms of validation and test accuracy.

Q4 - Limitations:

1. Computational Efficiency
    - Computational Complexity:
      The hybrid KNN model's computational complexity increases with the number of neighbors kkk and the size of the dataset. Performing KNN imputation twice (once for users and once for items) adds to the computational load.
    - Memory Usage:
      The model requires significant memory to store and process the large sparse matrix, especially when the dataset is large.
2. Further Improvement Directions
    - Incorporating Additional Metadata:
      Introducing additional features such as user demographics (age, gender) and item difficulty levels can provide more context for the imputation, potentially improving accuracy.
    - Advanced Imputation Techniques:
      Exploring more advanced imputation techniques, such as matrix factorization or deep learning-based models (e.g., autoencoders), could further enhance the imputation accuracy. Implementing these models may address some of the limitations of KNN by leveraging latent factors or more complex patterns in the data.
    - Optimizing Weight Parameter(α):
      Further tuning of the weight parameter α used to balance user-based and item-based imputations might yield additional improvements in accuracy. Using optimization techniques like grid search or Bayesian optimization can help find the optimal α value efficiently. By addressing these limitations and exploring the suggested improvements, the hybrid KNN model can be made more robust and effective for practical applications.

End of Report

Note: The following pages contain Contributions and LLM.

Contributions:

Zixuan Zeng 1008533419

- Completed full part A, which is the current stable version.
  (It's worth noting that we decided to try out part A independently to get familiar with the project. We shared opinions on Part A question 3 Matrix Factorization about the derivation of gradients and latent factors update rules)
- Part B Modified Matrix Factorization ALS model.
- Part B ensemble.
  (Though the ensembled model has a slight lower test accuracy than the two separate models in part B.)
- Latex final report.

...

...

LLM:

- ChatGPT 4o:
  - python stats resample
  - sklearn resample
  - scipy sparse matrix
  - latex draw figure