Project

Dian Rong, Jing Yu
CSC311 Fall 2023

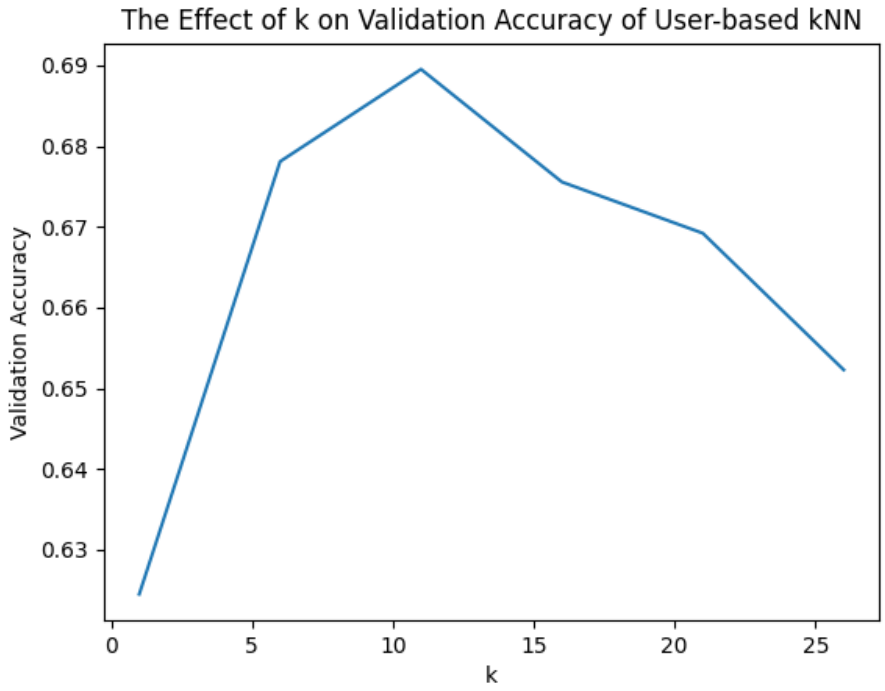
December 5, 2023

1 Part A

1.1 k-Nearest Neighbor.

(a)

- the validation accuracy for user-based kNN with k=1 is 0.6244707874682472.
- the validation accuracy for user-based kNN with k=6 is 0.6780976573525261.
- the validation accuracy for user-based kNN with k=11 is 0.6895286480383855.
- the validation accuracy for user-based kNN with k=16 is 0.6755574372001129.
- the validation accuracy for user-based kNN with k=21 is 0.6692068868190799.
- the validation accuracy for user-based kNN with k=26 is 0.6522720858029918.

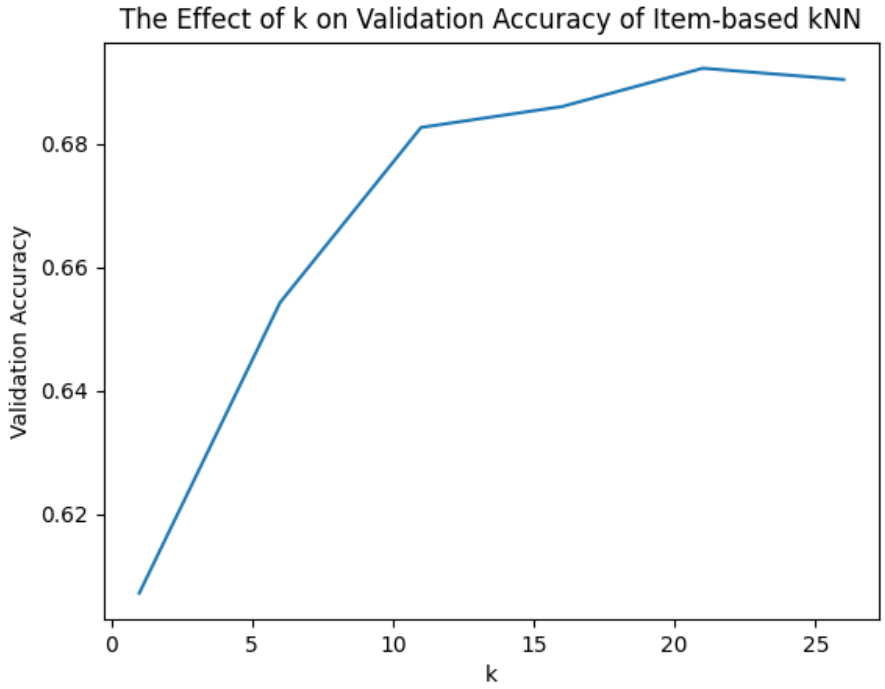


From k=1 to k=11, as k increases, validation accuracy also increases rapidly, however the function peaks at k=11 and for higher values of k, the validation accuracy decreases.

(b) The k* with the highest validation accuracy with user-based kNN is 11 and has a test accuracy of 0.6841659610499576.

(c) The underlying assumption is that if other students do the same on question A and question B (i.e. if a student gets question A right, then they also get question B right and if a student gets question A wrong, they also get question B wrong), then a specific student's correctness (correct or incorrect) on question A is the same as their correctness on question B.

- the validation accuracy for item-based kNN with k=1 is 0.607112616426757.
- the validation accuracy for item-based kNN with k=6 is 0.6542478125882021.
- the validation accuracy for item-based kNN with k=11 is 0.6826136042901496.
- the validation accuracy for item-based kNN with k=16 is 0.6860005644933672.
- the validation accuracy for item-based kNN with k=21 is 0.6922099915325995.
- the validation accuracy for item-based kNN with k=26 is 0.69037538808919.



From k=1 to k=21, as k increases, so does validation accuracy, however the function peaks at k=21 and for higher values of k, the validation accuracy starts to decrease. Additionally, the increase in validation accuracy starts to plateau around k=10.

The k* with the highest validation accuracy with item-based kNN is 21 and has a test accuracy of 0.6816257408975445.

(d) The test accuracy of user-based filtering is 0.6841659610499576 whereas the test accuracy of item-based filtering is 0.6816257408975445 so user-based filtering performs better.

(e) (a) The kNN doesn't take into account when answers were submitted. This means recent results have the same impact on predictions as old, potentially outdated results. In the extreme case, this implies that if a student takes a 3 year gap between using the same account, the model will predict that the student will make the same mistakes as they did 3 years ago when they may have learned a lot.

(b) Since the kNN weighs each question uniformly, an answer in the same subtopic as the original question has the the same effect on the prediction as an answer in a different subtopic. This is especially important since from the curse of dimensionality, all points tend to appear equidistant, causing the kNN to compute distances based on irrelevant dimensions.

1.2 Item Response Theory.

(a) Let $i \in \{0, 1, ..., 541\}$ refers to the id of 542 students. Let $j \in \{0, 1, ..., 1773\}$ refers to the id of 1774 questions.

$$p(\mathbf{C}|\boldsymbol{\theta}, \boldsymbol{\beta}) = \prod_{i=0}^{541} \prod_{j=0}^{1773} \left[\left(\frac{exp(\theta_i - \beta_j)}{1 + exp(\theta_i - \beta_j)} \right)^{c_{ij}} + \left(\frac{1}{1 + exp(\theta_i - \beta_j)} \right)^{1 - c_{ij}} \right]$$

Log-likelihood:

$$\begin{aligned} L &= logp(\mathbf{C}|\boldsymbol{\theta}, \boldsymbol{\beta}) \\ &= \sum_{i=0}^{541} \sum_{j=0}^{1773} \left[c_{ij} log\left(\frac{exp(\theta_i - \beta_j)}{1 + exp(\theta_i - \beta_j)} \right) + (1 - c_{ij}) log\left(\frac{1}{1 + exp(\theta_i - \beta_j)} \right) \right] \\ &= \sum_{i=0}^{541} \sum_{j=0}^{1773} \left[c_{ij} [(\theta_i - \beta_j) - log(1 + exp(\theta_i - \beta_j))] - (1 - c_{ij}) log(1 + exp(\theta_i - \beta_j)) \right] \\ &= \sum_{i=0}^{541} \sum_{j=0}^{1773} \left[c_{ij} (\theta_i - \beta_j) - log(1 + exp(\theta_i - \beta_j)) \right] \end{aligned}$$

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

$$\frac{\partial L}{\partial \theta_i} = \sum_{j=0}^{1773} \left[c_{ij} - \frac{exp(\theta_i - \beta_j)}{1 + exp(\theta_i - \beta_j)} \right]$$

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

$$\frac{\partial L}{\partial \beta_j} = \sum_{i=0}^{541} \left[\frac{exp(\theta_i - \beta_j)}{1 + exp(\theta_i - \beta_j)} - c_{ij} \right]$$

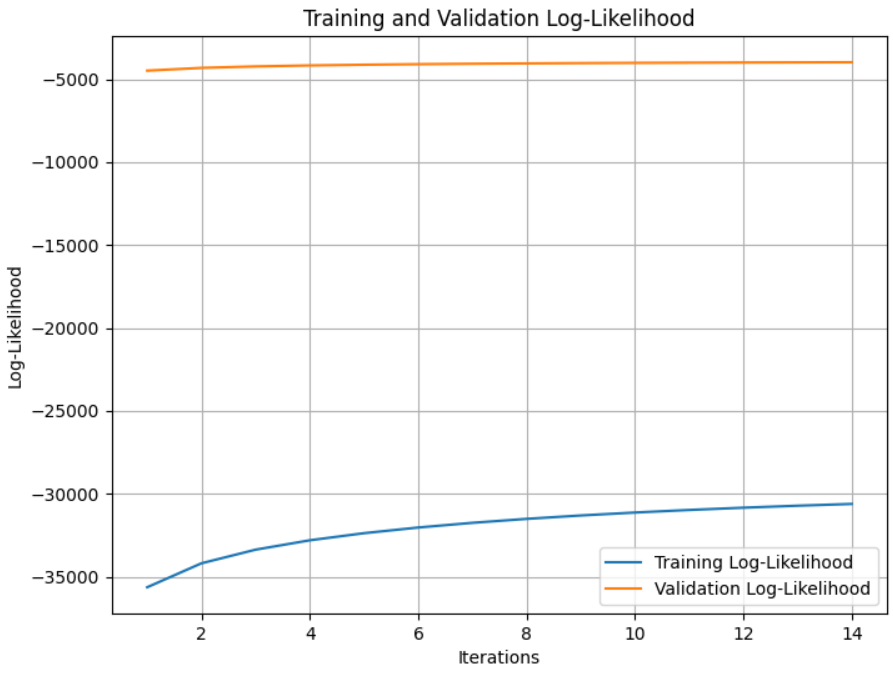
(b) Implementation in the file item_response.py

After tuning hyperparameters, we choose the best hyperparameter set according to the best validation accuracy.

Hyperparameters chosen:

- Learning rate = 0.01
- Number of iterations = 14

This is the training curve that shows the training and validation log-likelihoods as a function of iteration.



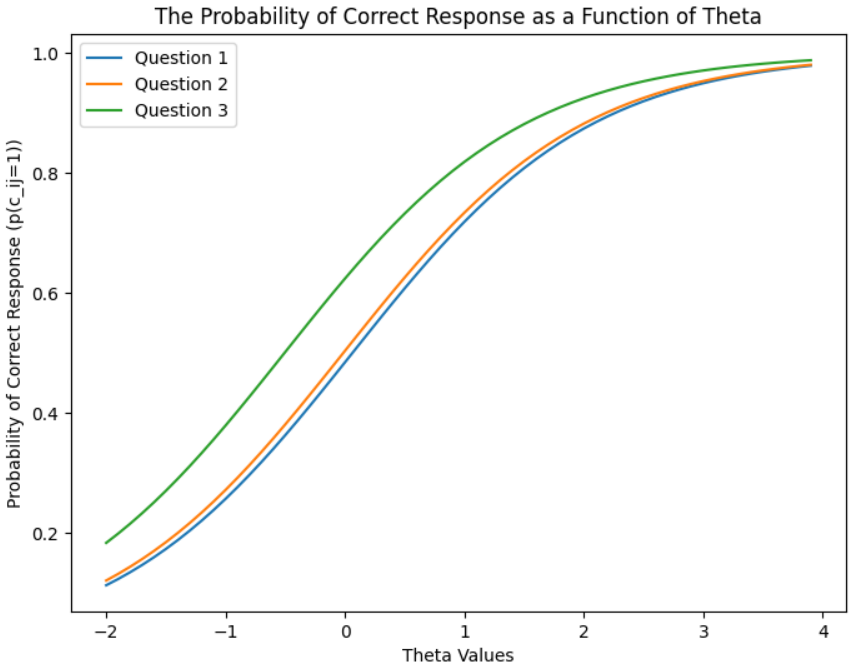
(c) Final Validation Accuracy: 0.7085802991814846

Final Test Accuracy: 0.7019475021168501

(d) For questions $j_1 = 1, j_2 = 2, j_3 = 3$, here is the plot that shows the probability of the correct response $p(c_{ij} = 1)$ as a function of θ given a question j.

The shape of these curves is similar to a sigmoid curve, which is s-shape. These curves indicate how difficult the questions are in relation to the students' abilities.

Steeper curves imply greater sensitivity of a question to changes in ability, while flatter curves suggest less sensitivity.



1.3 Neural Networks.

(a) Differences between ALS and neural networks:

1. They use different approaches. ALS is a matrix factorization method, while neural networks, including autoencoders, are deep learning models.

2. Neural networks are more complex and flexible than ALS. Neural networks can capture nonlinear patterns and relationships within data, while ALS is a linear factorization method.

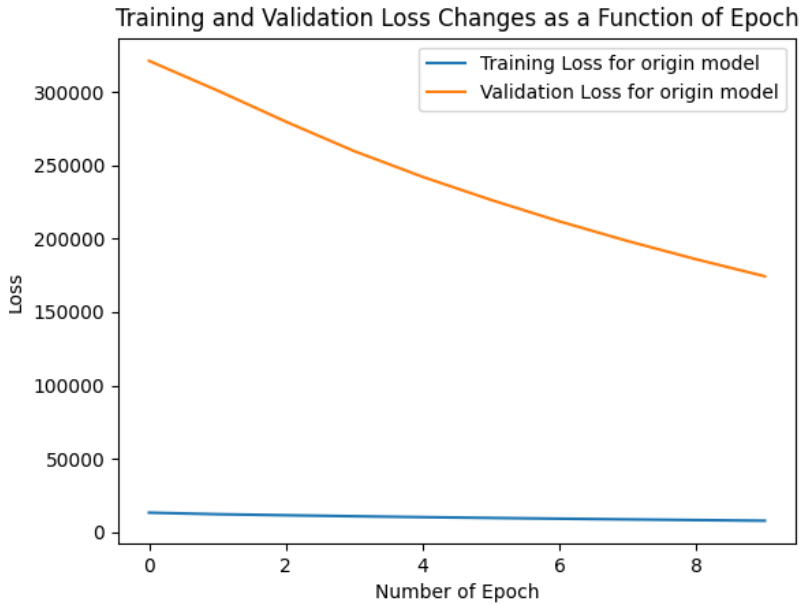
3. They use different optimization methods. ALS iteratively optimizes for user and item latent factors, while neural networks employ gradient-based optimization techniques to update weights and biases across layers.

(b) Implementation in the file neural_network.py

(c) Without regularization, we choose hyperparameters with the highest validation accuracy of 0.6852949477843635:

- $k^* = 50$
- learning rate = 0.05
- number of epoch = 10

(d) This is the plot that shows training and validation objectives changes as a function of epoch. The final test accuracy is 0.6756985605419137 for $k^* = 50$.



(e) After tuning the regularization penalty, we choose λ to be 0.001.

The Final Validation Accuracy: 0.6820491109229466

The Final Test Accuracy: 0.6785210273779283

The performances of these two models seem to be similar. It doesn't perform better with the regularization penalty.

1.4 Ensemble

The ensemble process was as follows:

1. Bootstrap the training dictionary to create 3 different samples
2. Create one k-Nearest Neighbours (k-NN) model, one Item-Response Theory (IRT) Model, and one Neural Network (NN) using the hyperparameters selected in parts (a)-(c).
3. Train the k-NN model on sample one, IRT on sample two, and NN on sample three
4. Create a list of sets of weights that each sum up to 3 (ex. [[1, 1, 1], [0.5, 0.5, 2], ...]) to generate the predictions
5. For each value in the validation set, generate a prediction from each model. The final prediction is then equal to whether their weighted average is greater than 0.5.

   In other words: final prediction $= \mathbb{I}[\frac{\sum_{i=0}^{2} w_i * p_i}{3} >= 0.5]$ where w_i, and p_i are the weight of model i and prediction of model i ($\in \{0, 1\}$) respectively

6. Sum up the accuracy of the validation set to get the validation accuracy of the set of weights

7. Choose the set of weights (w*) with the highest validation accuracy ([0.75, 1.5, 0.75] with 0.7002540220152413)

8. Test the ensemble on w* (test accuracy is 0.6979960485464296)

We did not obtain a better performance using the ensemble. Since the models are all trained on the same features, the models are too similar and their errors are correlated.

2 Part B

2.1 Justification

We think our algorithm in 1.4 may be underfitting because its training accuracy is low at 0.7091227420265311 and the final validation accuracy and test accuracy are similar at 0.7002540220152413 and 0.6979960485464296 respectively.

Therefore, we are adding complexity to our ensemble algorithm by training it on more features and adding parameters. Additionally our new base models will have more variation than the previous base models, so ensembling them will be more useful.

2.2 Diagram of Our Model

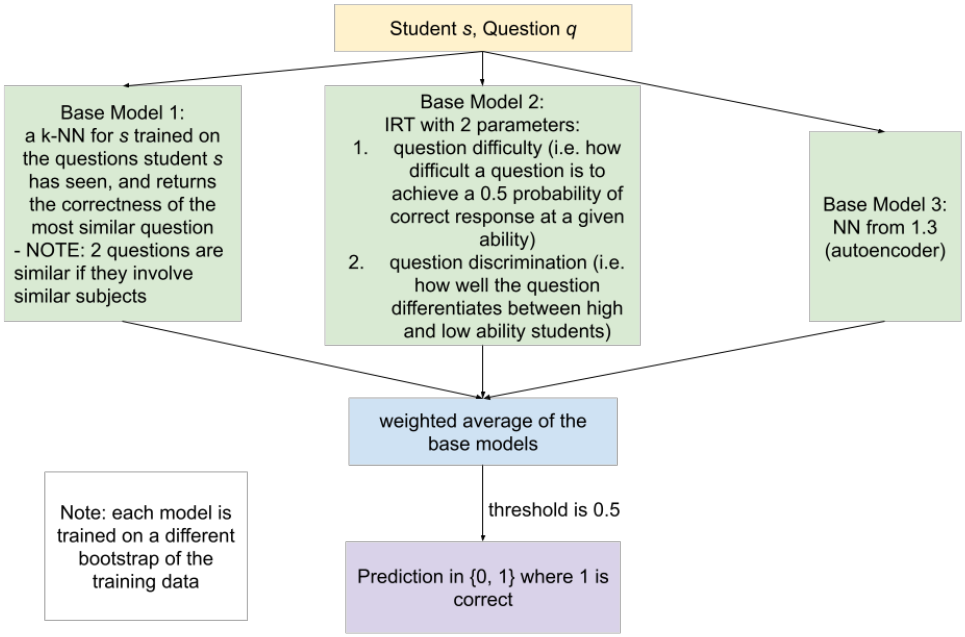


Figure 1: Diagram of our Model

Note: the primary data is of users and questions they got correct or incorrect.
Link to original figure

2.3 k-NN Modification

In this subject-based k-NN algorithm, given a user, the kNN finds the closest question that the user answered, and predicts the correctness based on the closest question's correctness.

The core underlying assumption is that if question A has the same subjects as question B, student S's correctness on A matches student S's correctness on B.

Note, subjects are coded as binary variables so if question q has subject i but not subject j, $q[i] = 1$ and $q[j] = 0$, and there are 388 subjects.

2.3.1 k-NN Algorithm

1. Of the Q questions student S has done, find k examples $\{q^{(i)}, c^{(i)}\}$ closest to the question instance q.
    - Note: Each question is represented as a vertex of its subjects
2. Classification output is majority class: $y* = \max_{c \in \{0,1\}} \sum_{i=1}^{k} \mathbb{I}(c = c^{(i)})$

2.4 IRT Modification

Modify IRT from one-parameter to two-parameter model by adding another parameter α_j, where α_j represents a discrimination of question j, which measures how well the question differentiates between high and low ability students.

2.4.1 IRT Algorithm

1. The probability that the question j is correctly answered by student i is formulated as:
$$p(c_{ij} = 1|\theta_i, \beta_j, \alpha_j) = \frac{exp(\alpha_j(\theta_i - \beta_j))}{1 + exp(\alpha_j(\theta_i - \beta_j))}$$
2. Calculate Log-likelihood, where N is the number of students and M is the number of questions:
$$L = logp(\mathbf{C}|\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\alpha}) = \sum_{i=1}^{N} \sum_{j=1}^{M} \Big[c_{ij}\alpha_j(\theta_i - \beta_j) - log\big(1 + exp(\alpha_j(\theta_i - \beta_j))\big) \Big]$$
3. Calculate the derivative of the log-likelihood with respect to θ_i, β_j, α_j:
$$\frac{\partial L}{\partial \theta_i} = \sum_{j=1}^{M} \big[c_{ij}\alpha_j - \frac{\alpha_j exp(\alpha_j(\theta_i - \beta_j))}{1 + exp(\alpha_j(\theta_i - \beta_j))} \big]$$
$$\frac{\partial L}{\partial \beta_j} = \sum_{i=1}^{N} \big[\frac{\alpha_j exp(\alpha_j(\theta_i - \beta_j))}{1 + exp(\alpha_j(\theta_i - \beta_j))} - c_{ij}\alpha_j \big]$$
$$\frac{\partial L}{\partial \alpha_j} = \sum_{i=1}^{N} \big[c_{ij}(\theta_i - \beta_j) - \frac{(\theta_i - \beta_j) exp(\alpha_j(\theta_i - \beta_j))}{1 + exp(\alpha_j(\theta_i - \beta_j))} \big]$$
4. Use gradient descent to find the parameters with maximum likelihood, initiating θ_i, β_j to be all zeros, α_j to be all ones, where γ refers to learning rate:
$$\theta_i \leftarrow \theta_i + \gamma \cdot \frac{\partial L}{\partial \theta_i} \qquad \beta_j \leftarrow \beta_j + \gamma \cdot \frac{\partial L}{\partial \beta_j} \qquad \alpha_j \leftarrow \alpha_j + \gamma \cdot \frac{\partial L}{\partial \alpha_j}$$

5. Tune the hyperparameters (different learning rates and number of iterations), to find the one with best validation accuracy.

2.5 Ensemble

Aside from changing the base models, we will use the same ensemble process.

1. Bootstrap the training dictionary (of the students, questions, and correctness) to create 3 different samples
2. Create one k-Nearest Neighbours (k-NN) model, one Item-Response Theory (IRT) Model, and one Neural Network (NN)
3. Train the k-NN model on sample one and the question metadata, IRT on sample two, and NN on sample three
4. Tune the hyperparameters for each model
5. Create a list of sets of weights that each sum up to 3 (ex. [[1, 1, 1], [0.5, 0.5, 2], ...]) to generate the predictions
6. For each value in the validation set, generate a prediction from each model. The final prediction is then equal to whether their weighted average is greater than 0.5.

   In other words: final prediction $= \mathbb{I}[\frac{\sum_{i=0}^{2} w_i * p_i}{3} >= 0.5]$ where w_i, and p_i are the weight of model i and prediction of model i ($\in \{0, 1\}$) respectively
7. Sum up the accuracy of the validation set to get the validation accuracy of the set of weights
8. Choose the set of weights (w*) with the highest validation accuracy
9. Test the ensemble on w*

2.6 Comparison

Model	Training Accuracy		Validation Accuracy		Test Accuracy	
	Previous	Current	Previous	Current	Previous	Current
k-NN	1.0	0.71024	0.68163	0.65086	0.68417	0.64352
IRT	0.73206	**0.73486**	0.70858	**0.71056**	0.70195	**0.70449**
NN	0.83917		0.68205		0.67852	
Ensemble Model	0.70912	**0.71500**	0.70025	0.69856	0.69799	**0.69913**

We think our original ensemble algorithm is underfitting, so we modified the algorithm to overcome it by adding complexity to the ensemble model with more parameters in the base models. We assume that ensemble performance will improve primarily due to increased

model diversity rather than overfitting or regularization since the only thing we change is the number of parameters to the models.

By comparing the new ensemble model with the baseline ensemble model, the new one demonstrates a marginal improvement in training and test accuracy. This suggests our new model doesn't underfit the data as much.

However, when comparing the individual modified models with individual baseline models, the accuracy of modified KNN went down, while the accuracy of modified IRT increased slightly. This suggests that although the overall ensemble model decreased its underfitting, the individual kNN's underfitting increased. This is likely because not all students have done many questions, causing certain users' kNNs to be small.

2.7 Limitations

1. Limited data availability:
    - Affecting KNN performance as outlined above
    - Insufficient discriminatory patterns in the data may hinder the effectiveness of adding the discrimination parameter to IRT model.
    - Don't account for contextual factors such as guessing or when the student answers causing the model to be more inaccurate for students who frequently guess or have taken a long break between completing questions
    - Possible extension: Investigate more informative features (ex. date of response) or use another model when there isn't much information on one input (ex. do item-based instead of user-based if the user hasn't done many questions)
2. Model complexity:
    - The addition of model complexity amplifies the risk of overfitting, impacting generalization to unseen data.
    - Possible extension: Implement regularization to mitigate overfitting; reduce dimensions of inputs (ex. reduce the number of subjects by combining similar ones)
3. Optimization limitation in ensemble:
    - The individual base models may perform differently when we combine them in ensemble model. However, we use the hyperparameters tuned in each individual base models for the ensemble one, which may not optimize the performance of each base model contribute to ensemble.
    - Possible extension: Use the same procedure except tune the hyperparameters of the base models together since Shahhosseini et al. suggests this could create a better performing ensemble (2017), i.e., choose the hyperparameters of the base models to maximize the performance of the ensemble model instead of the individual base models as we did here.

2.8 References

Jiang, Y., Yu, X., Cai, Y., & Tu, D. (2022). A multidimensional IRT model for ability-item-based guessing: the development of a two-parameter logistic extension model. Communications in Statistics: Simulation and Computation, 1–13. https://doi.org/10.1080/03610918.2022.2097694

Moghadamzadeh, A., Salehi, K., & Khodaie, E. (2011). A comparison the information functions of the item and test on one, two and three parametric model of the item response theory (irt). Procedia, Social and Behavioral Sciences, 29, 1359–1367. https://doi.org/10.1016/j.sbspro.2011.11.374

Olmuş, H., Nazman, E., & Erbaş, S. (2017). An evaluation of the two parameter (2-pl) irt models through a simulation study. Gazi University Journal of Science, 30(1), 235–249. https://dergipark.org.tr/en/pub/gujs/issue/28464/303387

Shahhosseini, M., Hu, G., & Pham, H. (2022). Optimizing ensemble weights and hyperparameters of machine learning models for regression problems. Machine Learning with Applications, 7(100251), 100251. https://doi.org/10.1016/j.mlwa.2022.100251

Thompson, N. (2018, November 29). What is the two parameter IRT model (2PL)? Assessment Systems. https://assess.com/what-is-the-two-parameter-irt-2pl-model/

Aiden Loe (2021, January 07). Intro to IRT https://aidenloe.github.io/introToIRT.html

3 Contributions

Part A:

- KNN: Dian
- IRT: Jing
- NN: Jing
- Ensemble:Dian

Part B:

- 2.1 Justification: Both
- 2.2 Diagram: Dian
- 2.3 k-NN: Dian
- 2.4 IRT: Jing
- 2.5 Ensemble: Dian
- 2.6 Comparison: Both
- 2.7 Limitations: Both
- 2.8 References: Both

Contributions: Both :)