CSC311 Summer 2025 Final Project

Swetha Poneasan, Natalia Tabja, Luis Calderon

August 8, 2025

1 k-nearest neighbors

(a)

The following table shows the accuracy values for each value of k:

k	Validation Accuracy
1	0.6245
6	0.6781
11	0.6897
16	0.6753
21	0.6681
26	0.6503

Below is the plot of each k value with their corresponding validation accuracy values. The graph peaks at k=11 and the validation accuracy decreases as k gets larger than 11.

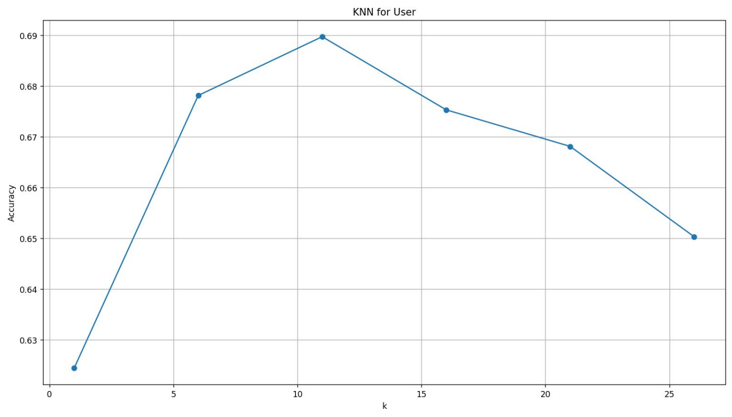
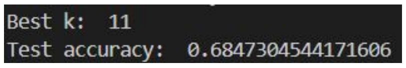


Figure 1: Plot of k values and validation accuracies for user-based KNN

(b)

Out of all tested k values, the best k value is k*=11 and the final test accuracy is 0.6847 as per the image below.



```
Best k:  11
Test accuracy:  0.6847304544171606
```

(c)

The underlying assumption of item-based collaborative filtering is that if a user answered a question in a certain way, it is highly likely that they answer similar questions in the same way.

The following table shows the accuracy values for each value of k for item-based collaborative filtering:

k	Validation Accuracy
1	0.6071
6	0.6544
11	0.6832
16	0.6863
21	0.6916
26	0.6902

Below is the plot of each k value with their corresponding validation accuracy values. The validation accuracy increases until its peak at k*=21, and slightly decreases at k=26.

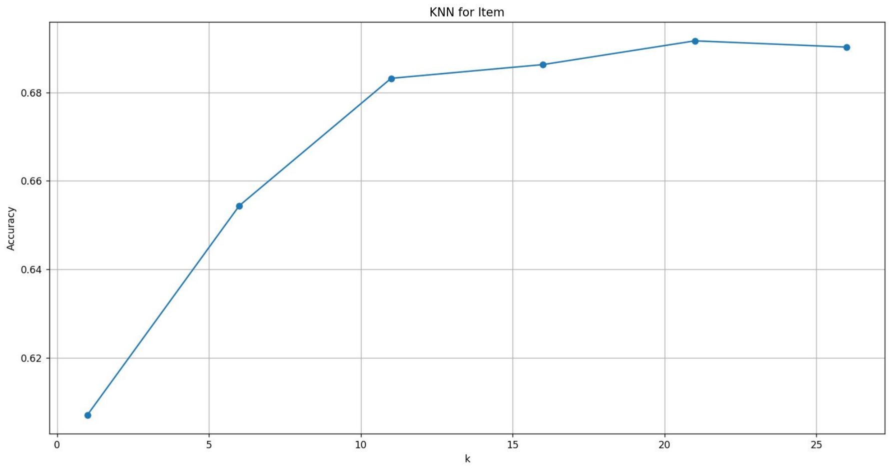
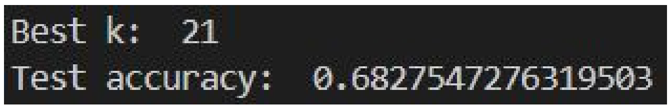


Figure 2: Plot of k values and validation accuracies for Item-based KNN

Out of all tested k values, the best k value is k* = 21 and the final test accuracy is 0.6828 as per the image below.



```
Best k:  21
Test accuracy:  0.6827547276319503
```

(d)

From our above results, the user-based collaborative filtering had final test accuracy of 0.6847 and the item-based collaborative filtering had final test accuracy of 0.6828. Based on these final test accuracy values, the user-based collaborative filtering performs slightly better, but both filtering methods are very similar.

(e)

1. The student and question matrix is very sparse, which makes it difficult to find neighbours that have similar responses. Specifically, there are not enough shared responses to find reliable neighbours when a student or question only has a few answers. This reduces the effectiveness of KNN for this specific problem. Additionally, sparsity may mean that distances are based on only a few observations which can increase noise when computing the similarity for both item-based and user-based collaborative filtering.

   The matrix sparsity will also cause issues when a new student or question is added to the dataset. New users do not have any prior responses, so there will be no user-based overlap in the matrix. Similarly, new questions will have no existing information on student performance, so there will be no item-based overlap, which can lead to unreliable predictions when using KNN.

2. KNN must compute similarities using item- or user-based collaborative filtering between the target user and question and every other user and question in the dataset. There are a large number of users and questions, which makes KNN computationally expensive and slow to run when KNN compares each student against every other student.

2 Item Response Theory

(a)

Let $z_{ij} = e^{\theta_i - \beta_j}$. Then

$$p(c_{ij} = 1 \mid \theta_i, \beta_j) = \frac{z_{ij}}{1 + z_{ij}}.$$

Because of conditional independence, we can express the joint likelihood as a product over all i, j:

$$P(C \mid \theta, \beta) = \prod_i \prod_j \Big(p(c_{ij} = 1 \mid \theta_i, \beta_j)\Big)^{c_{ij}} \Big(1 - p(c_{ij} = 1 \mid \theta_i, \beta_j)\Big)^{1-c_{ij}} = \prod_{i,j} \Big(\tfrac{z_{ij}}{1+z_{ij}}\Big)^{c_{ij}} \Big(\tfrac{1}{1+z_{ij}}\Big)^{1-c_{ij}}.$$

Taking logarithms gives the log-likelihood:

$$\ell(\theta, \beta) = \log P(C \mid \theta, \beta) = \sum_{i,j} \Big[c_{ij} \log \frac{z_{ij}}{1 + z_{ij}} + (1 - c_{ij}) \log \frac{1}{1 + z_{ij}}\Big] = \sum_{i,j} \Big[c_{ij} \log z_{ij} - \log(1 + z_{ij})\Big].$$

Substitute $z_{ij} = e^{\theta_i - \beta_j}$:

$$\ell(\theta, \beta) = \sum_{i,j} \Big[c_{ij}(\theta_i - \beta_j) - \log\big(1 + e^{\theta_i - \beta_j}\big)\Big].$$

Finally, differentiating with respect to each parameter gives

$$\frac{\partial \ell}{\partial \theta_i} = \sum_j \Big(c_{ij} - \frac{e^{\theta_i - \beta_j}}{1 + e^{\theta_i - \beta_j}}\Big), \qquad \frac{\partial \ell}{\partial \beta_j} = \sum_i \Big(\frac{e^{\theta_i - \beta_j}}{1 + e^{\theta_i - \beta_j}} - c_{ij}\Big).$$

(b)

- See functions implemented in item_response.py
- Hyperparameters selected: Learning rate of 0.01 with 200 iterations.
- With our chosen hyperparameters, we will report the training curve that shows the training and validation log-likelihoods as a function of iterations.

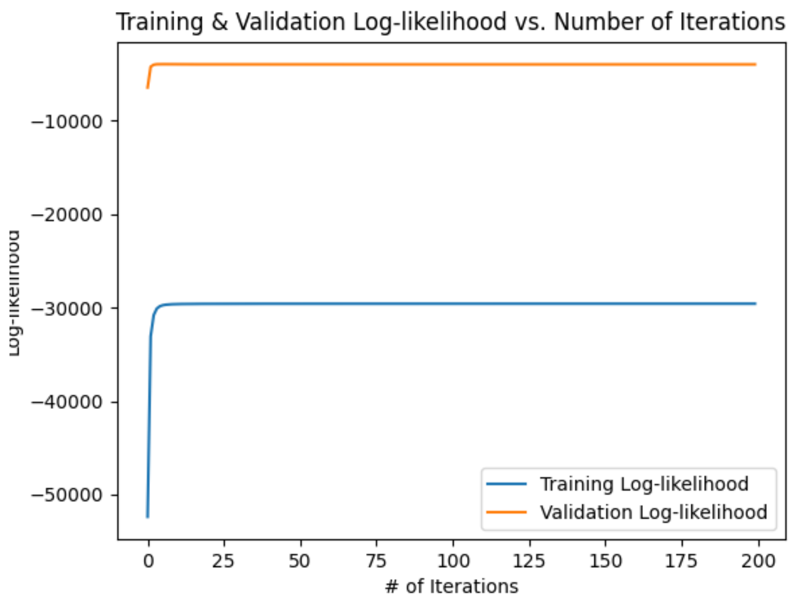
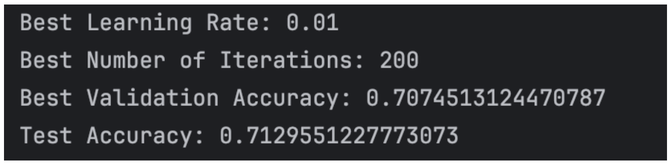


Figure 3: Training and Validation log-likelihoods as a function of iterations.

(c)

With the implemented code, we obtained a final validation accuracy of 0.7075 and a final test accuracy of approximately 0.7130.



```
Best Learning Rate: 0.01
Best Number of Iterations: 200
Best Validation Accuracy: 0.7074513124470787
Test Accuracy: 0.7129551227773073
```

(d)

Each curve is the logistic function

$$p(c_{ij} = 1 \mid \theta_i, \beta_j) = \frac{1}{1 + e^{-(\theta_i - \beta_j)}}.$$

- **Inflection point at** $p = 0.5$: Occurs exactly where $\theta_i = \beta_j$, marking the item's *difficulty*.
- **Maximum information at the inflection**: The derivative

  $$\frac{dp}{d\theta} = p(\theta)\,[1 - p(\theta)]$$

  is maximized at $p = 0.5$, so small changes in ability cause the largest change in probability, making the item most *informative* around its difficulty.
- **S-shape**: Probability grows slowly at low ability, accelerates around $\theta = \beta_j$, then plateaus at high ability—reflecting that items yield the most information near their difficulty and little at the extremes.

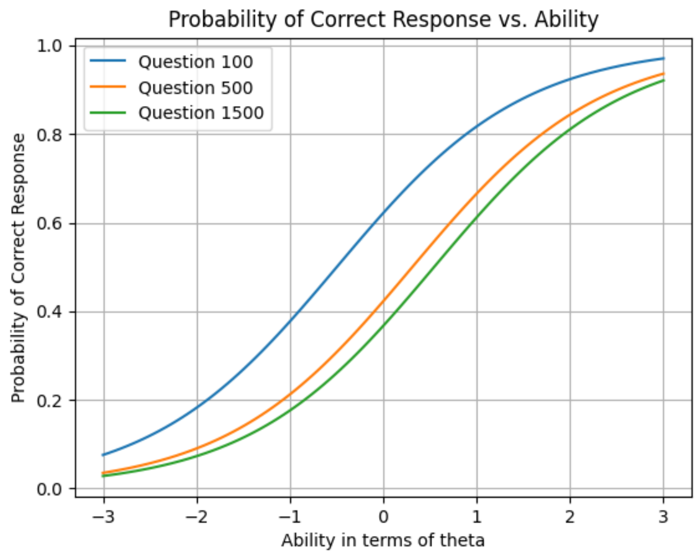


Figure 4: Probability of correct response vs. ability for items with discrimination parameters 100, 500, and 1500.

- **Horizontal shift with equal slopes**: In the 1PL model all curves have the same steepness but are shifted by β_j:
  - Blue (Q100), β_{100} smallest $\rightarrow$ *easiest*: students with lower-than-average ability ($\theta < 0$) have a fairly high probability of answering correctly.
  - Orange (Q500), moderate shift $\rightarrow$ *moderate*.
  - Green (Q1500), β_{1500} largest $\rightarrow$ *hardest*: students need a higher-than-average ability ($\theta > 0$) to have a high probability of answering correctly.
- **Flat tails**: As $\theta \rightarrow -\infty$, $p \rightarrow 0$; as $\theta \rightarrow +\infty$, $p \rightarrow 1$. Very low- or high-ability students provide little additional information on that item.

3 Neural Networks

(a) ALS vs Neural Networks

In ALS, we aim to find two matrices U and V such that the original response matrix $R \approx UV^\top$. We first hold V fixed and solve for U to minimize reconstruction error, then hold U fixed and solve for V. This is efficient to implement because each subproblem reduces to a least-squares minimization, which has a closed-form solution. The algorithm is called Alternating Least Squares because we alternate between updating U and V until convergence.

Three key differences between ALS and NNs are the following:

1. **Model architecture:** ALS optimizes just two latent matrices, whereas neural networks can include multiple hidden layers with many units per layer, enabling more expressive nonlinear modeling.
2. **Training structure:** ALS alternates between optimizing users (holding items fixed) and items (holding users fixed). Neural networks are trained end-to-end via a full forward and backward pass through the entire model.

3. **Optimization method and computational cost:** ALS relies on closed-form updates via least-squares minimization and does not require gradients. Neural networks, in contrast, use gradient-based optimization (e.g., stochastic gradient descent) and backpropagation, which can be computationally more intensive and sensitive to hyperparameters like learning rate.

(b) Implementing the AutoEncoder

We begin with a student's response vector $\mathbf{v} \in \mathbb{R}^d$, where each entry is either 1 (correct), 0 (incorrect), or missing (NaN). The encoder compresses this input into a lower-dimensional latent representation $\mathbf{z} \in \mathbb{R}^k$, where $k < d$. This latent vector is intended summarize the student's overall ability or profile.

The decoder then reconstructs $\mathbf{v}$ from $\mathbf{z}$, attempting to predict the full set of responses—including the missing entries—to see if the model can accurately reconstruct the answers me already know. If it does well on the known answers, we trust that its predictions for the missing responses are likely to be accurate as well. During training, we minimize the mean squared error between the reconstruction and the original response vector, but only on the observed (non-NaN) entries.

Our forward pass implements the following structure:

$$\hat{\mathbf{v}} = \sigma\left(\mathbf{W}^{(2)} \cdot \sigma\left(\mathbf{W}^{(1)}\mathbf{v} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}\right)$$

which corresponds to the line of code:

```
out = F.sigmoid(self.h(F.sigmoid(self.g(inputs))))
```

The input first passes through the encoder (a linear transformation followed by a sigmoid), producing the latent vector $\mathbf{z}$. This is then passed through the decoder (another linear transformation and sigmoid) to produce the predicted output vector $\hat{\mathbf{v}} \in \mathbb{R}^d$.

(c) Choosing the Hyperparameters

Trial	k	Learning Rate	Epochs	**Validation Accuracy**
1	10	0.05	20	0.6850
2	50	0.05	20	0.6825
3	50	0.05	30	0.6780
4	50	0.01	20	0.6836
5	**50**	**0.01**	**40**	**0.6884**
6	100	0.01	40	0.6849
7	200	0.01	30	0.6788
8	500	0.01	30	0.6712

Table 1: Validation accuracy for various combinations of k, learning rate, and training epochs, focusing on $k = 50$.

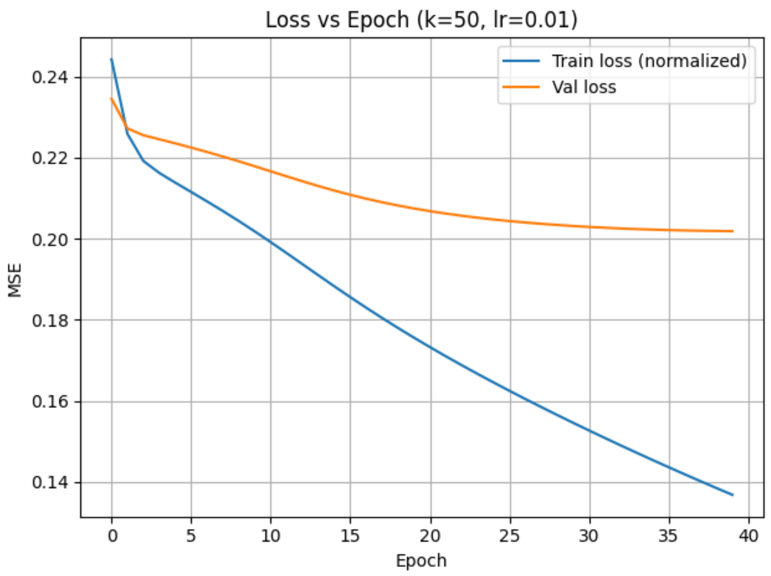
We performed a grid search over:

- latent dimension $k \in \{10, 50, 100, 200, 500\}$
- learning rate $\in \{0.001, 0.01, 0.05, 0.1\}$
- number of epochs $\in \{20, 30, 40\}$

Each configuration was trained once and evaluated on the validation set. We selected the combination ($k = 50,\ \text{lr} = 0.01,\ \text{epochs} = 40$) as it achieved the highest validation accuracy of 0.6884.

(d) Final Training, Validation, and Test Performance

We retrained the model using the selected hyperparameters ($k = 50$, learning rate $= 0.01$, epochs $= 40$) and recorded the training and validation MSE losses over epochs. Since the training loss is the raw sum of squared errors over all training samples, while the validation loss is computed as the mean squared error per validation point, we normalized the training loss by dividing it by the number of samples to make both curves directly comparable:



The training loss decreases steadily, while the validation loss flattens around epoch 35, suggesting convergence without overfitting. The validation accuracy remained stable around 0.685 ± 0.002, indicating consistent generalization performance across epochs.

After retraining with the chosen configuration, the model achieved a test accuracy of **0.6912** on the public test set, which aligns with the validation results.

(e) Effect of L_2 Regularization

To assess the impact of L_2 regularization, we introduced a penalty term $\frac{\lambda}{2}\|W\|^2$ to the objective function and evaluated different values of $\lambda \in \{0.001, 0.01, 0.1, 1\}$ while keeping the best previously chosen hyperparameters fixed ($k = 50$, learning rate $= 0.01$, epochs $= 40$).

λ	Validation Accuracy	Test Accuracy
0.001	**0.6904**	**0.6861**
0.01	0.6797	0.6757
0.1	0.6250	0.6246
1.0	0.6228	0.6257

We found that a small amount of regularization ($\lambda = 0.001$ to 0.01) had little effect on validation performance. However, increasing λ beyond this threshold significantly reduced both validation and test accuracy. For instance, at $\lambda = 1.000$, both metrics dropped by approximately 10% relative to their values at $\lambda = 0.001$. This suggests that additional regularization degraded performance, implying the model was not overfitting and may have already been near optimal capacity (as evidenced by the solid validation accuracies in Table 1).

Overall, regularization had a mild effect in this setting. The best performance with regularization was obtained at $\lambda = 0.001$, which still underperformed compared to the unregularized model.

4 Ensemble

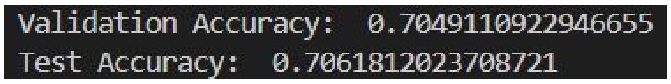
To implement bagging ensemble, we used logistic regression as the base model for all three ensemble members. In order to convert our data into a format that could be used for logistic regression, we created a feature matrix for each of the training, validation, and test sets. The matrices were built of one-hot vectors for each student-question pair, which set a 1 at the index corresponding to the student and another 1 at the index corresponding to the question.

Logistic regression was chosen for all three base models because it is quick to train and is computationally cheap, which allows for good performance. Additionally, we used one-hot vectors for our students and questions when meant that each input became a large vector which had only 2 non-zero entries, resulting in the data being very sparse. This caused the input space to be large relative to the number of training samples, which increased variance since many features (students and questions) appeared only a few times in the data. This made the bagging ensemble beneficial, since it could reduce the variance across all three base models by averaging their predictions.

For each base model, we generated a bootstrap sample from the training data by randomly sampling with replacement to create three training sets using resample(x_train, y_train). Then, we trained a logistic regression model on each of the three bootstrapped datasets using LogisticRegression from sklearn.linear_model. We obtained predictions for each of the models on the validation and test sets to compute the final validation and test accuracies. To create the final ensemble prediction, we averaged the predicted probabilities from each of the three base models. We used predict_proba to get the probability that each student answered each question correctly. Then, we used np.mean to average the predictions for the three models before thresholding.

The final validation and test accuracy are as follows:

- Final validation accuracy: 0.7049
- Test accuracy: 0.7062



```
Validation Accuracy:  0.7049110922946655
Test Accuracy:  0.7061812023708721
```

The hyperparameters used are:

- max_iter = 200, which ensures that the sparse data is able to be fit by logistic regression. It also ensures that we will have convergence.

Here are the individual accuracies when bagging is not used on each of the individual logistic regression base models:

Model	**Validation Accuracy**	**Test Accuracy**
Model 1	0.7005	0.7022
Model 2	0.6995	0.7031
Model 3	0.6953	0.7031

These results show that compared to using a single logistic regression model, using the ensemble provides more stable predictions and slightly improves the accuracy. This is because the ensemble can combine multiple models trained on the data to reduce its overall variance. By averaging the predictions, it is also able to reduce overfitting.

5 Part B

Introduction

The 1PL IRT model lacks flexibility to capture item informativeness. We address this by extending it to a 2PL model with clustering priors.

1. Formal Description

We propose a 2PL IRT extension with clustering priors in four stages:

(1) Item Features: Construct a standardized matrix $X \in \mathbb{R}^{J \times p}$ from available item metadata (e.g., one-hot topic encodings, summary stats).

(2) PCA: Apply SVD to X, keeping top d components ($\approx 90\%$ variance explained): $Z = \tilde{X} V_{:,1:d}$.

(3) Clustering: Run K-means (10 restarts) on rows of Z to assign each item j to a cluster $g(j) \in \{1, \dots, K\}$.

(4) Cluster-Regularized 2PL: We define the response probability as

$$p_{ij} = \sigma\big(\alpha_j(\theta_i - \beta_j)\big), \quad \sigma(x) = \frac{1}{1+e^{-x}},$$

with priors:

$$\alpha_j \sim \mathcal{N}(\mu_k^{(\alpha)}, \tau_\alpha^2), \quad \beta_j \sim \mathcal{N}(\mu_k^{(\beta)}, \tau_\beta^2), \quad \text{for } g(j) = k.$$

Our loss is the negative log-posterior:

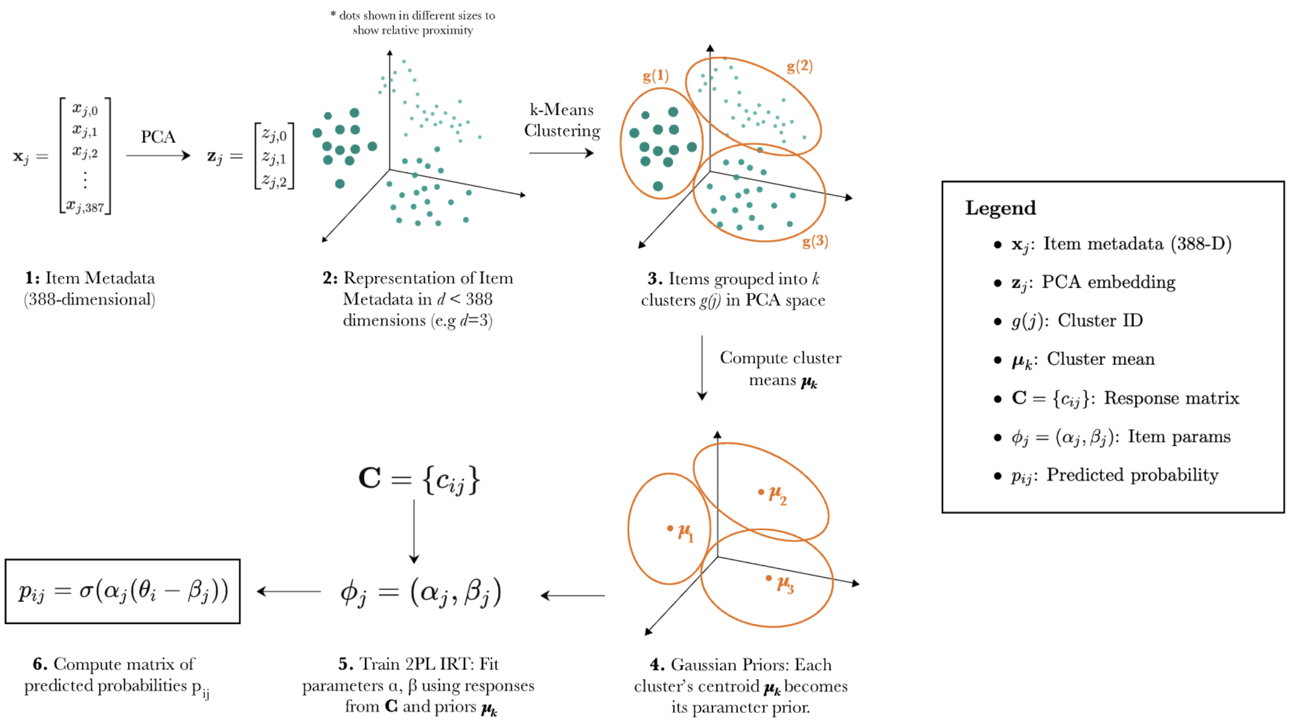
$$\mathcal{L} = -\sum_{i,j} \big[C_{ij} \log p_{ij} + (1 - C_{ij}) \log(1 - p_{ij})\big] + \sum_j \left[\frac{(\alpha_j - \mu_{g(j)}^{(\alpha)})^2}{2\tau_\alpha^2} + \frac{(\beta_j - \mu_{g(j)}^{(\beta)})^2}{2\tau_\beta^2}\right].$$

Algorithm 1 Alternating Adam updates for 2PL IRT with Clustering Priors

```
1:  Initialize: Fit 1PL to get $\theta_i^{(0)}, \beta_j^{(0)}$; set $\alpha_j^{(0)} = 1$; compute $\mu_k^{(\alpha,0)}, \mu_k^{(\beta,0)}$
2:  repeat
3:      for each student $i$ do
4:          $\theta_i \leftarrow \theta_i - \eta \frac{\partial \mathcal{L}}{\partial \theta_i}$
5:      end for
6:      for each item $j$ do
7:          $\alpha_j \leftarrow \alpha_j - \eta \frac{\partial \mathcal{L}}{\partial \alpha_j}$
8:          $\beta_j \leftarrow \beta_j - \eta \frac{\partial \mathcal{L}}{\partial \beta_j}$
9:      end for
10:     for each cluster $k$ do
11:         $\mu_k^{(\alpha)} \leftarrow \frac{1}{|\mathcal{G}_k|} \sum_{j \in \mathcal{G}_k} \alpha_j$
12:         $\mu_k^{(\beta)} \leftarrow \frac{1}{|\mathcal{G}_k|} \sum_{j \in \mathcal{G}_k} \beta_j$
13:     end for
14: until convergence on held-out log-likelihood
```

Why our proposed method should be expected to perform better: Adding item–specific discrimination α_j reduces 1PL under-fitting, while cluster-based priors shrink (α_j, β_j) toward shared means to curb over-fitting. PCA plus k-means supplies the structure for those priors, and an Adam optimiser—warm-started from 1PL estimates—ensures fast, stable training.

2. Schematic of our Revised IRT Model



* dots shown in different sizes to show relative proximity
$\mathbf{x}_j = [x_{j,0}, x_{j,1}, x_{j,2}, \vdots, x_{j,387}]^T$
PCA
$\mathbf{z}_j = [z_{j,0}, z_{j,1}, z_{j,2}]^T$
k-Means Clustering
g(1)
g(2)
g(3)
1: Item Metadata (388-dimensional)
2: Representation of Item Metadata in $d < 388$ dimensions (e.g d=3)
3. Items grouped into k clusters $g(j)$ in PCA space
Compute cluster means $\boldsymbol{\mu}_k$
μ_1
μ_2
μ_3
4. Gaussian Priors: Each cluster's centroid $\boldsymbol{\mu}_k$ becomes its parameter prior.
$\mathbf{C} = \{c_{ij}\}$
$\phi_j = (\alpha_j, \beta_j)$
$p_{ij} = \sigma(\alpha_j(\theta_i - \beta_j))$
5. Train 2PL IRT: Fit parameters α, β using responses from **C** and priors $\boldsymbol{\mu}_k$
6. Compute matrix of predicted probabilities p_{ij}
Legend
$\mathbf{x}_j$: Item metadata (388-D)
$\mathbf{z}_j$: PCA embedding
$g(j)$: Cluster ID
$\boldsymbol{\mu}_k$: Cluster mean
$\mathbf{C} = \{c_{ij}\}$: Response matrix
$\phi_j = (\alpha_j, \beta_j)$: Item params
p_{ij}: Predicted probability

Figure 5: **Schematic overview of the model pipeline.** High-dimensional item metadata is reduced using PCA (1–2), clustered via k-means (3), and used to define Gaussian priors over item parameters (4). These priors guide training of a 2PL IRT model using the response matrix $\mathbf{C} = \{c_{ij}\}$ (5), which outputs predicted probabilities p_{ij} (6).

3. Comparison or Demonstration

Accuracy Comparison To compare our modified algorithm to the baseline model, we trained both models using a learning rate of 0.001 and 200 iterations as the hyperparameters, and computed their validation and test accuracies under these settings.

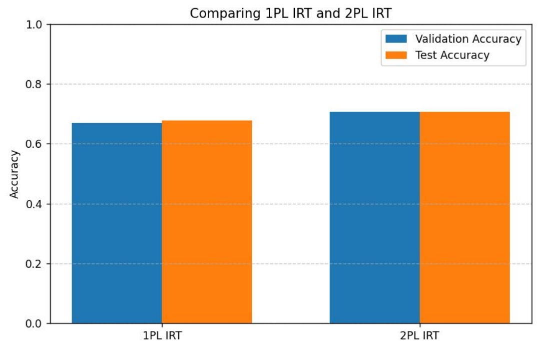


Figure 6: Accuracy values between 1PL and 2PL

Model	Val Acc	Test Acc
1PL IRT	0.6691	0.6782
2PL IRT	0.7060	0.7070

Table 2: Validation and test accuracies for 1PL vs. 2PL

Table 2 shows the validation and test accuracy values corresponding to Figure 6.

Model	Validation Accuracy	Test Accuracy
1PL IRT on full training set	0.5912	0.5955
2PL IRT on full training set	0.7075	0.7059
1PL IRT on 50% training set	0.5576	0.5456
2PL IRT on 50% training set	0.6943	0.6915

Table 3: Validation and test accuracies of both models under full and reduced training data

From our comparison, it is clear that our modified 2PL IRT algorithm achieves higher validation and test accuracies than our baseline model by around 3-4%, as shown in Table 2.

Hypothesis Testing and Experiment As per our argument on why our modified algorithm should help, our hypothesis is that the IRT 2PL algorithm performs better due to regularization since PCA+K-means discovers meaningful item groups.

1. Experiment Design We tested the above hypothesis by comparing 1PL IRT with 2PL IRT by completing the following steps. This experiment was done using a learning rate of 0.001 and 50 iterations.
   (a) Train both models on the full training dataset.
   (b) Train both models on a reduced dataset that contains 50% of the training data.
   (c) Compare the validation and test accuracies of both tests. If 2PL IRT is regularizing, it will have a larger advantage on the reduced dataset.
2. Experiment Output

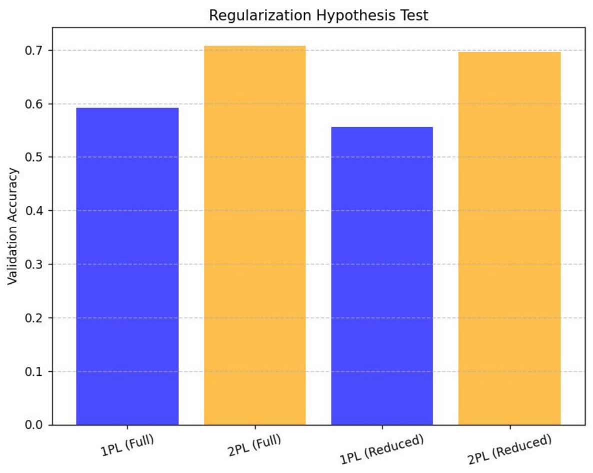


Figure 7: Validation accuracy for 1PL vs. 2PL on full and 50% training sets

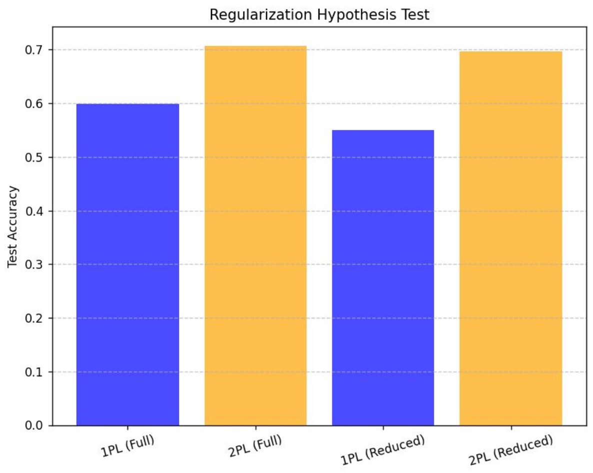


Figure 8: Test accuracy for 1PL vs. 2PL on full and 50% training sets

3. Analysis of Results

   As per our results, 2PL outperforms 1PL on both the full and reduced training datasets. When the dataset is reduced by 50%, 1PL suffers a drop from 0.5912 to 0.5576 on validation and 0.5955 to 0.5456 on test. On the other hand, 2PL has a much smaller drop: it drops from 0.7075 to 0.6943 on validation and 0.7059 to 0.6915 on test, as shown in Table 3. This is a clear sign of regularization, which shows that the cluster priors used stabilize α and β, which is able to reduce overfitting. This smaller decrease in accuracy for 2PL supports our hypothesis that PCA+K-means acts as a form of regularization by leveraging meaningful item groups.

4. Limitations

While our clustering-regularized 2PL IRT model improves accuracy over the baseline, it has several important limitations:

1. Dependence on metadata quality. Our approach relies on item metadata to construct PCA embeddings and define priors. If the metadata is sparse, uninformative, or misaligned with actual item properties (e.g., vague subject tags or incomplete features), the clusters may not reflect true question structure. This can lead to poor regularization, especially if unrelated items are grouped together.

2. Hard clustering limits flexibility. We assign each item to a single cluster using k-means. This ignores the possibility that a question might relate to multiple conceptual themes (e.g., both algebra and geometry), or that item characteristics vary continuously. As a result, the model may over-regularize items that don't cleanly fit into one group.

3. Risk of over-regularizing outliers. For atypical or high-variance questions, cluster-based priors can dominate the gradients and prevent the model from fitting those items accurately. This is especially problematic when questions behave differently from their peers but are still informative.

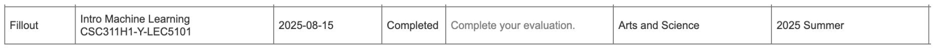
Possible extensions: Future work could explore soft clustering (e.g., Gaussian mixtures), adaptive regularization strengths, or learning the item groupings jointly with the IRT parameters instead of fixing them upfront. These could allow for more flexible and robust generalization.

Team Contributions

- **Luis Calderon** — Implemented the Item-Response Theory model (Part A, Q2) and wrote the formal description of the extended 2PL-with-clustering algorithm (Part B, Q1).
- **Natalia Tabja** — Designed and trained the neural-network autoencoder (Part A, Q3); created the schematic diagram and wrote the Limitations discussion (Part B, Q2 and Q4).
- **Swetha Poneasan** — Conducted the k-nearest neighbours experiments and bagging ensemble (Part A, Q1 and Q4); ran the comparison/regularisation experiments (Part B, Q3).

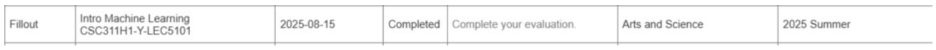
Verification of Submitting Course Evaluation

We confirm that all team members have submitted the course evaluation for this course.



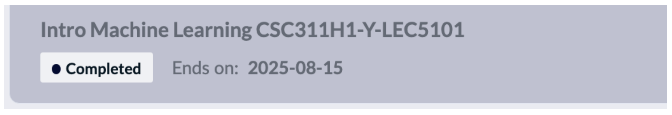
Fillout	Intro Machine Learning CSC311H1-Y-LEC5101	2025-08-15	Completed	Complete your evaluation.	Arts and Science	2025 Summer

Natalia's Course Evaluation Proof



Fillout	Intro Machine Learning CSC311H1-Y-LEC5101	2025-08-15	Completed	Complete your evaluation.	Arts and Science	2025 Summer

Swetha's Course Evaluation Proof



Intro Machine Learning CSC311H1-Y-LEC5101

● Completed Ends on: 2025-08-15

Luis's Course Evaluation Proof