University of Toronto
CSC311, Fall 2023

Final Project:

Aryaman Arora - 1007858873
Spencer Perkins - 1008102955

Contributions:

Aryaman Arora:

1. Created Final Report Template.
2. Item Response Theory:
   (a) Calculations:
      i. Likelihood.
      ii. Log-Likelihood.
      iii. Derivative of Log-Likelihood With Respect to θ_i.
      iv. Derivative of Log-Likelihood With Respect to β_j.
   (b) Code Implementation of IRT.
   (c) IRT Report Section.
3. Part B:
   (a) Improved Autoencoder Neural Network Code Implementation:
      i. Building.
      ii. Training.
      iii. Testing.
      iv. Optimization.
   (b) Formal Description Report Section.
   (c) Comparison:
      i. Hypothesis/Extension Testing.
      ii. Report Section.
   (d) Limitations Report Section.

Spencer Perkins:

1. k-Nearest Neighbors:
   (a) KNN Code Implementation.
   (b) KNN Report Section.
2. Autoencoder Neural Network:
   (a) NN Code Implementation.

   (b) NN Report Section.

3. Ensemble:
   (a) Ensemble Code Implementation.
   (b) Ensemble Report Section.
4. Part B:
   (a) Part B Formal Description Report Section.
   (b) Part B Diagram Creation.

Part A:

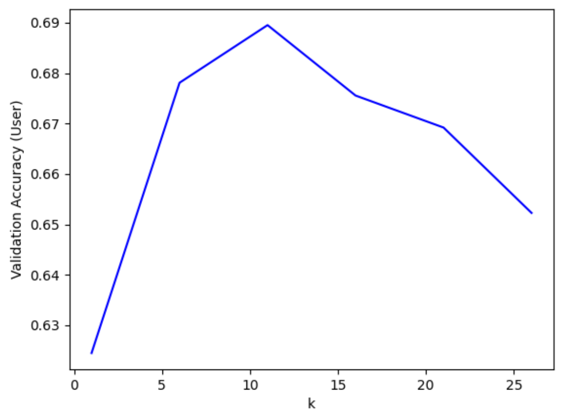
1. **k-Nearest Neighbor:**

   (a) Here is the plot and accuracy of the user-based collaborative filtering kNN model on the validation data as a function of k:

   Accuracies:

   k = 1 User-based Validation Accuracy: 0.6244707874682472
   k = 6 User-based Validation Accuracy: 0.6780976573525261
   k = 11 User-based Validation Accuracy: 0.6895286480383855
   k = 16 User-based Validation Accuracy: 0.6755574372001129
   k = 21 User-based Validation Accuracy: 0.6692068868190799
   k = 26 User-based Validation Accuracy: 0.6522720858029918

   Plot:



   (b) Our final test accuracy for our chosen $k*$ value, $k* = 11$, is 0.6841659610499576.

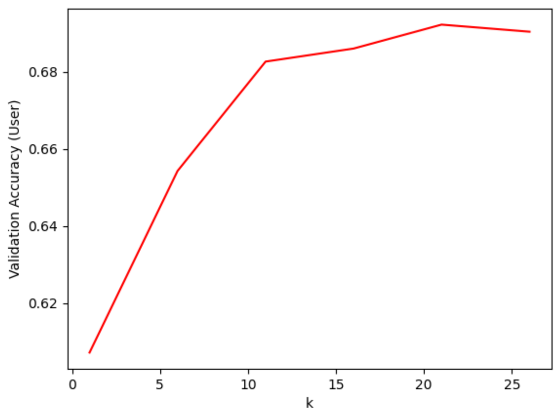
   (c) The underlying assumption on item-based collaborative filtering is that if question A is answered similarly to question B by multiple students, that the answers for other questions answered by the same students will also be similar.

   Here is the plot and accuracies of the item-based collaborative filtering kNN model on the validation data as a function of k:

   Accuracies:

k = 1 Item-based Validation Accuracy: 0.607112616426757
k = 6 Item-based Validation Accuracy: 0.6542478125882021
k = 11 Item-based Validation Accuracy: 0.6826136042901496
k = 16 Item-based Validation Accuracy: 0.6860005644933672
k = 21 Item-based Validation Accuracy: 0.6922099915325995
k = 26 Item-based Validation Accuracy: 0.69037538808919

Plot:



Our final test accuracy for our chosen $k*$ value, $k* = 21$, is 0.6816257408975445.

(d) Which method performs better depends on the k value, with $k >= 16$, item-based collaborative filtering performs better, with $k <= 11$, user-based collaborative filtering performs better.

(e) Here are two potential limitations of kNN for our task:

i. Scalability: kNN is computationally expensive, especially as the size of the dataset grows. As more user or question data is gathered, the model will take significantly longer for each prediction.

ii. Sparsity of Data: In collaborative filtering, the user-item interaction matrix often has many missing entries. If we have a scenario where a user hasn't answered many questions or a question hasn't been answered by many users, neighbors will be further apart and the model will give worse predictions.

2. Item Response Theory:

(a) Here are the steps we need to take in order to derive the log-likelihood, $\log(p(C|\theta, \beta))$, for all students and questions:

Let's assume that a student's result for a question c is a Bernoulli random variable for our two parameters, θ and β.

In the instructions for this assignment, we are given the probability that the question j is correctly answered by a student i:

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

However, we are not given the probability that the question j is incorrectly answered by a student i, so we must determine that before calculating the likelihood equation for Item Response Theory:

$$p(c_{ij} = 0|\theta_i, \beta_j) = 1 - \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)}$$

Now, we can succinctly write the two of these equations together to get our likelihood equation:

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

Assuming that $\{c_ij, ..., c_NM\}$ are independent and identically distributed, the joint probability of the outcome $\{c_ij, ..., c_NM\}$ is:

$$p(c_ij, ..., c_NM|\theta_i, \beta_j) = \Pi_{i=1}^N \Pi_{j=1}^M (\frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)})^{c_{ij}}(1 - \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)})^{1-c_{ij}}$$

Therefore, our likelihood function is:

$$L(\theta_i, \beta_j) = \Pi_{i=1}^N \Pi_{j=1}^M (\frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)})^{c_{ij}}(1 - \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)})^{1-c_{ij}}$$

Now that we have calculated our likelihood function, we can take the log of it to get our log-likelihood function:

$$l(\theta_i, \beta_j) = \log(L(\theta_i, \beta_j))$$

$$\Rightarrow \log(\Pi_{i=1}^N \Pi_{j=1}^M (\frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)})^{c_{ij}}(1 - \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)})^{1-c_{ij}})$$

$$\Rightarrow \sum_{i=1}^N \sum_{j=1}^M (c_{ij} \log(\frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)}) + (1 - c_{ij}) \log(1 - \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)}))$$

Therefore, our log-likelihood equation for all students and questions is:

$$\sum_{i=1}^N \sum_{j=1}^M (c_{ij} \log(\frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)}) + (1 - c_{ij}) \log(1 - \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)}))$$

Now that we have derived the log-likelihood, we will take the derivative of it with respect to θ_i and β_j:

Let us start by taking the derivative of the log-likelihood function with respect to θ_i:

$$\frac{dl}{d\theta_i} = \frac{d}{d\theta_i}(\sum_{i=1}^N \sum_{j=1}^M (c_{ij} \log(\frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)}) + (1 - c_{ij}) \log(1 - \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)})))$$

$$\Rightarrow \sum_{i=1}^N \sum_{j=1}^M (\frac{d}{d\theta_i} c_{ij} \log(\frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)}) + \frac{d}{d\theta_i}(1 - c_{ij}) \log(1 - \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)}))$$

$$\Rightarrow \sum_{j=1}^M c_{ij} - \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)}$$

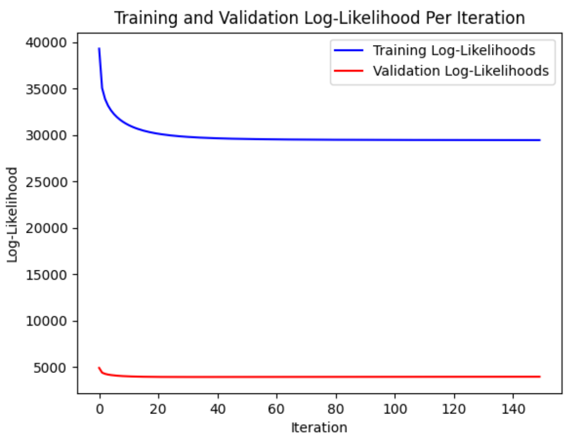
Now, let us take the derivative of the log-likelihood function with respect to β_j:

$$\frac{dl}{d\beta_j} = \frac{dl}{d\beta_j}(\sum_{i=1}^N \sum_{j=1}^M (c_{ij} \log(\frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)}) + (1 - c_{ij}) \log(1 - \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)})))$$

$$\Rightarrow \sum_{i=1}^N (\frac{dl}{d\beta_j} c_{ij} \log(\frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)}) + \frac{dl}{d\beta_j}(1 - c_{ij}) \log(1 - \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)}))$$

$$\Rightarrow \sum_{i=1}^N -c_{ij} + \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i-\beta_j)}$$

(b) Here is the training curve that shows the training and validation log-likelihoods as a function of iteration, where we had 150 iterations and a learning rate of 0.01:

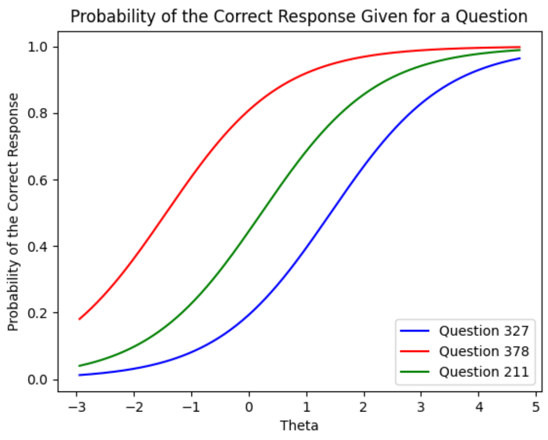


(c) Here is the final validation and test accuracies:

Accuracies:

Final Validation Accuracy: 0.7063223257126728
Final Test Accuracy: 0.707874682472481

(d) Here are the three curves showing the probability of the correct response $p(c_{ij} = 1)$ as a function of θ given a question j for the three questions we chose, $j_1 = 327$, $j_2 = 378$, and $j_3 = 211$:
Plot:



From the above graph, we can see that all of the lines are similar to the exponential function. We can also see that when $\theta \to -\inf$, the probabilities for all of the lines move towards 0, and when $\theta \to \inf$, the probabilities for all of the lines move towards 1. Each of these lines represent the probability a student will answer a question chosen from the dataset correctly given their current ability.

3. **Neural Networks:**

(a) Here are three differences between ALS and neural networks:

i. ALS and neural networks differ in that ALS tries to decompose the user-item interaction matrix into two lower-dimensional matrices representing users and items, while neural networks try to encode the entire input data into a number of latent dimensions which capture relationships within the data.

ii. They differ in how they train. ALS iteratively solves for the user and item matrices to minimize the reconstruction error, it relies on trying to solve a least-squares problem. Neural network iteratively use backpropagation and gradient descent to update it's weights over and over by computing the loss on the input data.

iii. ALS can only work with explicit feedback (users providing ratings for items), by filling in missing values in a sparse matrix. Neural Networks can also work with implicit feedback (Trying to infer user rating from data), making them ultimately more versatile.

(b) We implemented the AutoEncoder class that performs a forward pass of the autoencoder following the instructions in the docstring.

(c) We have trained the autoencoder using latent dimensions $k \in \{10, 50, 100, 200, 500\}$ and have tuned the optimization hyperparameters.
The $k*$ that has the highest validation accuracy is $k* = 10$.

(d) With our chosen $k*$, $k* = 10$, with $lr = 0.1$ and $num\_epochs = 40$. here is the plot of how the training and validation objectives change as a function of epoch:
Plot:

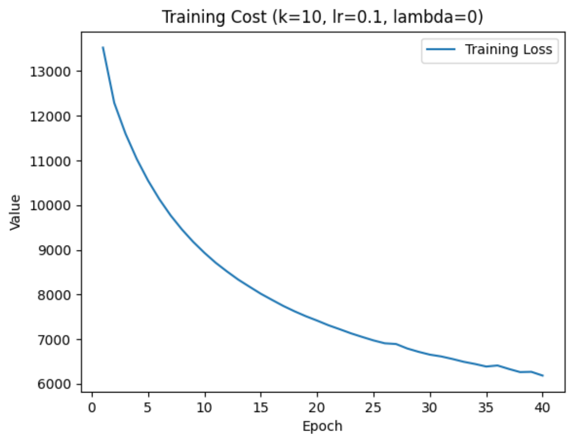


Figure 1: q3d1.png

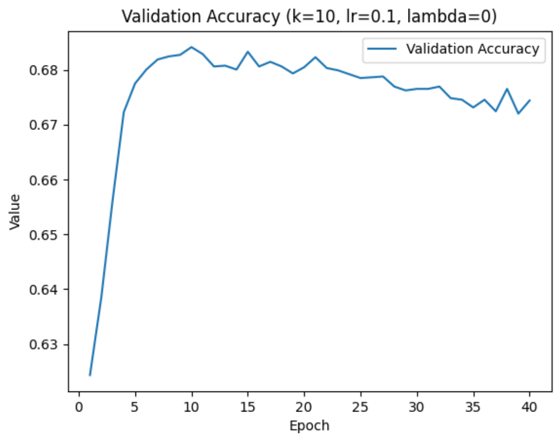


Figure 2: q3d2.png

Our final test accuracy was 0.6768 for our Auto Encoder Neural Network Model.

(e) We have modified the function train so that the objective adds the L_2 regularization.
Using the k and other hyperparameters selected from part (d), we have tuned the regularization penalty $\lambda \in \{0.001, 0.01, 0.1, 1\}$.
With our chosen λ, $\lambda = 0.0001$, here are the final validation and test accuracies we got:

Validation Accuracy: 0.6771
Test Accuracy: 0.6773

Our model seems to perform roughly the same with the regularization penalty, and as lambda goes up, the performance seemed to go down.

4. Ensemble:

The three base models we selected and trained using with bootstrapping the training set are all from Scikit-Learn and are their Decision Tree Classifer, Random Forest Classifier, and Gradient Boosting Classifer.

Here are the final valdiation and test accuracies we calculated:

Validation Accuracy: 0.6808

Test Accuracy: 0.6817

The ensemble process consisted of bootstrapping the training set seperatly for each of the three base models, training each of the three base models on their respective bootstrapped training sets, and then predicting the test set using each of the three base models. The final prediction for each student-question pair was the average, but functionally the mode of the three predictions from the base models.

We got slightly worse performance than the knn model, and roughly the same as the neural network. This is likely because the base models we chose are all very similar and are all tree-based models. If we had chosen more diverse base models, we likely would have gotten better performance.

Part B

1. Formal Description:

We made several changes to the Autoencoder Neural Network in Part A: 3. ii.

Metadata:

We added a metadata layer to the neural network. This layer takes in the metadata of the question's subject ids from *question_meta.csv*. This layer is then fed into the first hidden layer of the neural network.

This method is intended to increase the accuracy of the neural network by allowing it to learn the relationship between students and the subjects which they perform well on, not just the question they did well on. For example, in order to predict whether a student will answer a new question correctly or incorrectly, we can see how well the student has been doing on other questions with similar subjects, like Decimals or Percentages, instead of just basing the prediction on the similarity between question IDs.

This metadata is ran through a feature extractor to create a 3D input array of size (num_questions, num_subjects, num_users).

The inputs are finally embedded and concatenated before being fed into the model for training.

Loss:

We changed the loss function from the Mean Squared Error Loss Function to the Binary Cross Entropy Loss Function. The Binary Cross Entropy Loss Function excels in classification problems, like predicting whether a student can answer a question correctly or not, and when we can assume that the error follows a Binomial Distribution. The MSE Loss Function is better to use when we have an estimation problem and when we can assume that the error follows a Normal Distribution. Based on our experience building the base-models from Part A for this project, we can tell that the error follows more closely to a Binomial Distribution than a Normal Distribution, and as we are trying to build a model which can classify whether a student can answer a question correctly or incorrectly, the Binary Cross Entropy Loss Function was the better option to use.

Our new loss function is defined as:

$$\text{BCE}(y, \hat{y}) = -\left(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})\right)$$

Our new objective function is:

$$\min_\theta \textstyle\sum_{v \in S} \text{BCE}(v, f(v; \theta))$$

Model Architecture:

We modified the architecture of the original model, which was two linear layers with sigmoids activation functions.

Our new model architecture is a sequence of Dense layers containing ReLU activation functions, Batch Normalization and Dropout layers in between. There is a final Sigmoid activation function at the end for the output layer.

Let v represent the input vector, which is a concatenation of the embeddings of questions, users, and subjects, and θ represent the model weights.

Let $W^{(i)}$ represent the weights of the i^{th} layer, and $b^{(i)}$ represent the bias of the i^{th} layer.

Let g represent the ReLU activation function, and h represent the Sigmoid activation function.

The function our improved model computes is defined as:

$$f(v;\theta) = h\left(W^{(16)} \cdot g\left(W^{(15)} \cdot \ldots g\left(W^{(1)} \cdot v + b^{(1)}\right) + b^{(15)}\right) + b^{(16)}\right)$$

Early Stopping:

We added early stopping to the model in the hopes of preventing any possible overfitting and shortening the training time of the model.

It stops after 7 epochs with no improvement in the validation accuracy, at which point it will restore the model weights to the epoch of the highest validation accuracy.

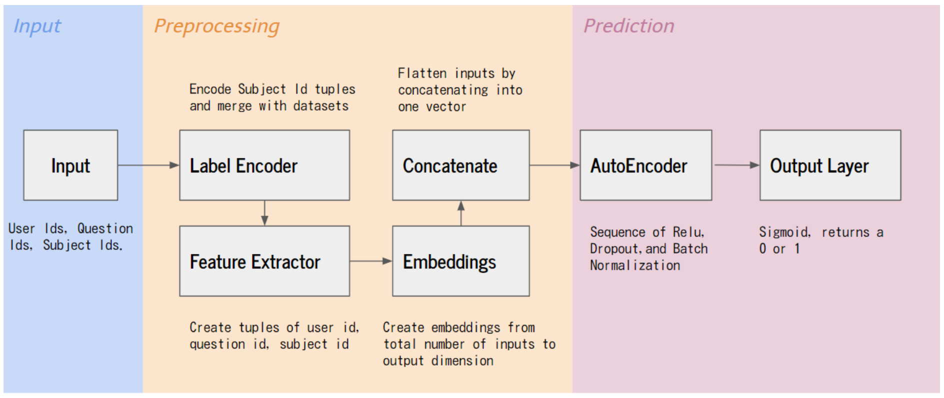
Optimzer:

Instead of using the Stochastic Gradient Descent Optimizer we used in our neural network model from Part A, we decided to use the Adam Optimizer to hopefully improve the optimization of our model.

The goal of this was to enhance the training efficiency and convergence speed. The Adam Optimizer combines the benefits of two extensions of Stochastic Gradient Descent: the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). The Adam Optimizer adjusts the learning rates for each parameter individually, allowing for more efficient training across different dimensions and scales.

2. **Diagram:**

Here is a visual representation of our Improved Autoencoder Neural Network:



Input
Preprocessing
Prediction
Encode Subject Id tuples and merge with datasets
Flatten inputs by concatenating into one vector
Input
Label Encoder
Concatenate
AutoEncoder
Output Layer
User Ids, Question Ids, Subject Ids.
Feature Extractor
Embeddings
Sequence of Relu, Dropout, and Batch Normalization
Sigmoid, returns a 0 or 1
Create tuples of user id, question id, subject id
Create embeddings from total number of inputs to output dimension

3. **Comparison:**

As written above, here are the accuracies for all of the base models we implemented:

k-Nearest Neighbor:

User-Based Collaborative Filtering: 0.6895

Item-Based Collaborative Filtering: 0.6816

Item Response Theory:

Validation Accuracy: 0.7063
Test Accuracy: 0.7079

Autoencoder Neural Network:

Validation Accuracy: 0.6771
Test Accuracy: 0.6773

Ensemble:

Validation Accuracy: 0.6808
Test Accuracy: 0.6817

With our improved Autoencoder Neural Network, here were the final accuracies we received:

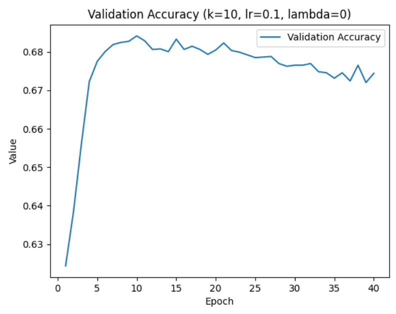
Improved Autoencoder Neural Network:

Validation Accuracy: 0.7042
Test Accuracy: 0.7059

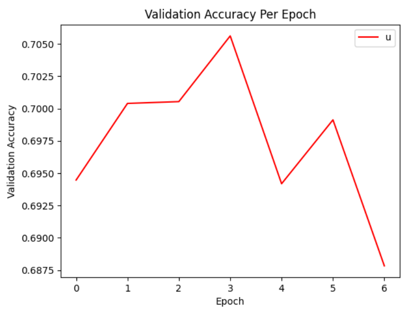
From these listed accuracies, we can see that the only model whose performance rivaled that of our improved model is the Item Response Theory model we built in Part A of the project.

However, in comparison to the original Autoencoder Neural Network we built in Part A, our new and improved Autoencoder Neural Network is around 3% better in terms of accuracy.

Here are the plots for the Original Autoencoder Neural Network and the Improved Autoencoder Neural Network:



V.S.



From these graphs, we can see that by utilizing Early Stopping in the Improved Autoencoder Neural Network, we got a higher accuracy on the model with far fewer epochs.

The way we are testing how our extensions to the Original Autoencoder Neural Network improve the base model design is by retraining our Improved Autoencoder Neural Network with the original optimizer, the SGD optimizer, first, and then, after switching back to the new optimizer, the Adam optimizer, we will retrain the Improved Autoencoder Neural Network with the original loss function, the Mean Squared Error Loss Function.

Here are the results from retraining the improved model with the original optimizer:

Validation Accuracy: 0.6719

Test Accuracy: 0.6571

Here are the results from retraining the improved model with the new optimizer and the old loss function:

Validation Accuracy: 0.7011

Test Accuracy: 0.7051

As we can see from this demonstration, we can tell that the extensions we implemented in the Improve Autoencoder Neural Network, the new optimizer (Adam) and the new loss function (Binary Crossentropy), have both improved the accuracy of the base model.

However, we can also tell that the change in optimizer played a larger role in improving the accuracy than the change in loss function. This is probably because the Adam optimizer utilizes two extensions of SGD: the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp).

4. **Limitations:**

Our improved Autoencoder Neural Network does have some limitations.

One of the biggest limitations for this improved model would be the amount of data it requires to make an accurate prediction. Though the accuracy of this improved model is higher than the base model from Part A, because we utilized the metadata from *question_meta.csv* for the training of the model, in order for it to make the best prediction possible for a given student and question ID, it will also require the subjects associated for this question ID to be input. Therefore, we would say that one of the large limitiations of this improved Autoencoder Neural Network model is that it requires the user ID, question ID, and subject metadata associated with the input question ID to make an accurate prediction.

Another limitation associated with the previous one we talked about is the fact that if our model is trying to predict whether a student can answer a question correctly or not based on the data it was trained on, but the student has not completed any questions with the same or similar subjects as the input question, then the model will be unable to make an accurate prediction as it has no data to base its prediction off of. The reason this limitation exists in not just our model, but really in all of the models we have built over the course of this project, is because the data all of these models were trained on does not contain how each and every student has answered each and every question in the database. This is very understandable as students can be of different education levels so they can't all answer the same questions, but we believe that one way to fix this issue is by padding the training data. By this, we mean that by adding rows containing students and questions they have not answered before and assigning the *is_correct* column to 0, we can provide more data to the model to make, hopefully, more accurate predictions.