Machine Learning – Assignment 3 Neural Network

[redacted] [redacted]

Preprocessing: Data is normalized (z normalization) since features have different range of scales and the data is split into train, validation and test data set, where parameter decisions for the optimal neural network model is made from performance on validation set during training. **Stop criteria for the neural network training is based on maximum iterations or epoch where the validation cost/error is minimum**. As the configuration of network change, the convergence or stop criteria change too. So, every change made in the network like adding variables or changing network layers or hidden nodes we need to use results of validation data to decide best parameters for training along with stop criteria (minimum error epoch). Number of neurons in output layer is fixed to 4 as there are four levels of categories (A, B, C, D) in the data.

Data after Z-Normalization:

	all_mcqs_avg_n20	all_NBME_avg_n4	CBSE_01	CBSE_02
count	7.200000e+01	7.200000e+01	7.200000e+01	7.200000e+01
mean	1.018475e-15	-1.392405e-15	-1.148772e-16	1.511137e-16
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
min	-2.124762e+00	-2.102311e+00	-2.509294e+00	-1.914751e+00
25%	-7.012907e-01	-6.290565e-01	-8.255179e-01	-6.162072e-01
50%	-7.588434e-02	4.443150e-02	1.637004e-02	-2.104122e-02
75%	7.605487e-01	8.862915e-01	6.898804e-01	6.011778e-01
max	2.065077e+00	2.022802e+00	2.710412e+00	2.197305e+00

Data split size for each label:

Train data: Level [0, 1, 2, 3], train_size_per_level [15, 33, 21, 3]

Validation data: Level [0, 1, 2, 3], validation_size_per_level [7, 7, 3, 1]

Test data: Level [0, 1, 2, 3], Test_size_per_level [9, 6, 6, 1]

1. **Neural Network with BSOM Dataset (100 points):**

a. Can you implement a backpropagation Neural Network using variables 'all_mcqs_avg_n20', 'all_NBME_avg_n4', 'CBSE_01', and 'CBSE_02' to classify 'LEVEL' with a single hidden layer and 5 hidden nodes?

Yes, backpropagation is implemented for given variables, the stop criteria is defined using the cost function graph where validation cost/error is minimum.

Model configuration:

Initially we assume learning rate alpha=0.5, with weights initialization as random within epsilon range [+0.001 -0.001]. From Figure 1 minimum validation cost is at epoch 565. So, taking epoch 565 as stop criteria which has min cost **1. 6672745782412186.**Model has 1 input layer, 1 hidden layer (5 neurons) and 1 output Layer.

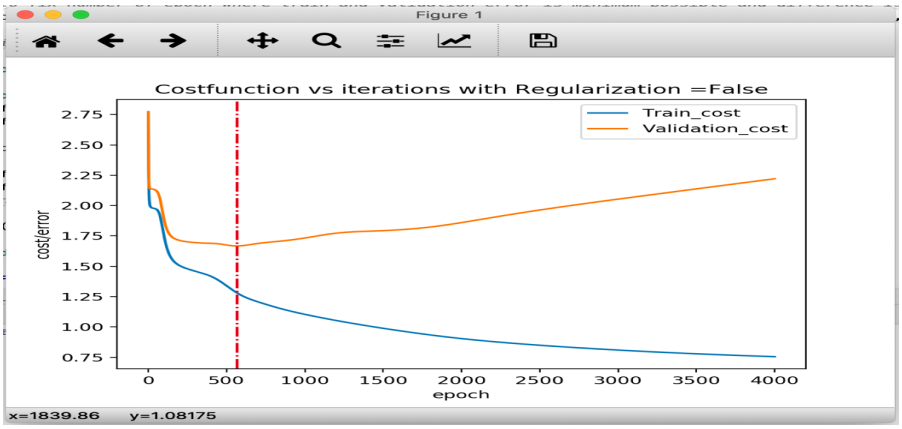


Figure 1:epoch vs error for train and validation data

Analysis of Model: During training, the metrics on validation and train is given in Figure 2. The train and validation **weighted F1-score are 0.67 and 0.61** appears like model is not overfitting.

Model Metrics:

```
TrainPrediction
[[10  5  0  0]
 [ 2 24  7  0]
 [ 0  6 15  0]
 [ 0  0  3  0]]
              precision    recall  f1-score   support

           0       0.83      0.67      0.74        15
           1       0.69      0.73      0.71        33
           2       0.60      0.71      0.65        21
           3       0.00      0.00      0.00         3

   micro avg       0.68      0.68      0.68        72
   macro avg       0.53      0.53      0.52        72
weighted avg       0.66      0.68      0.67        72
```

```
validation predictions
[[4 3 0 0]
 [0 5 2 0]
 [0 1 2 0]
 [0 0 1 0]]
              precision    recall  f1-score   support

           0       1.00      0.57      0.73         7
           1       0.56      0.71      0.63         7
           2       0.40      0.67      0.50         3
           3       0.00      0.00      0.00         1

   micro avg       0.61      0.61      0.61        18
   macro avg       0.49      0.49      0.46        18
weighted avg       0.67      0.61      0.61        18
```

Figure 2 :Train and Validation Confusion matrix and F1-score

b. Test different number of hidden nodes (at least 4 more options) with a single hidden layer and compare the performance. Does the highest number of hidden nodes have the highest performance?

Solution: Highest number of hidden nodes did not have highest performance.

Analysis: Highest performance among different options of hidden nodes is considered w.r.t minimum validation cost because **the stability of neural network is estimated by error/cost.** The model with **minimal error and highest accuracy/f1-score** is considered as high performer. From Table 1 and Figure 3 to 9, it is evident that the model is **performing better for number of hidden nodes=2**. This result is solely depending on the data used for modeling the neural network.

Even for number of hidden nodes = 3 the performance is nearly same, however number of parameters required to train increase with increase in number of hidden nodes. To keep model less complex, we choose hidden nodes = 2 with single hidden layer.

Number of hidden nodes with single layer	Validation minimum cost	stopping epoch	Weighted f1 score on validation data	Weighted f1 score on train data
1	1.7015744722164292	691	0.54	0.65
2	**1.5908250781037774**	**672**	**0.65**	**0.69**
3	1.6018114223898001	686	0.65	0.67
4	1.6517269752438943	648	0.61	0.67
5	1.6672745782412186	565	0.61	0.67
6	1.6346716137789334	618	0.65	0.71
7	1.70673534067146	274	0.47	0.70

Table 1: Information about minimum cost and associated f1-scores for different number of neurons

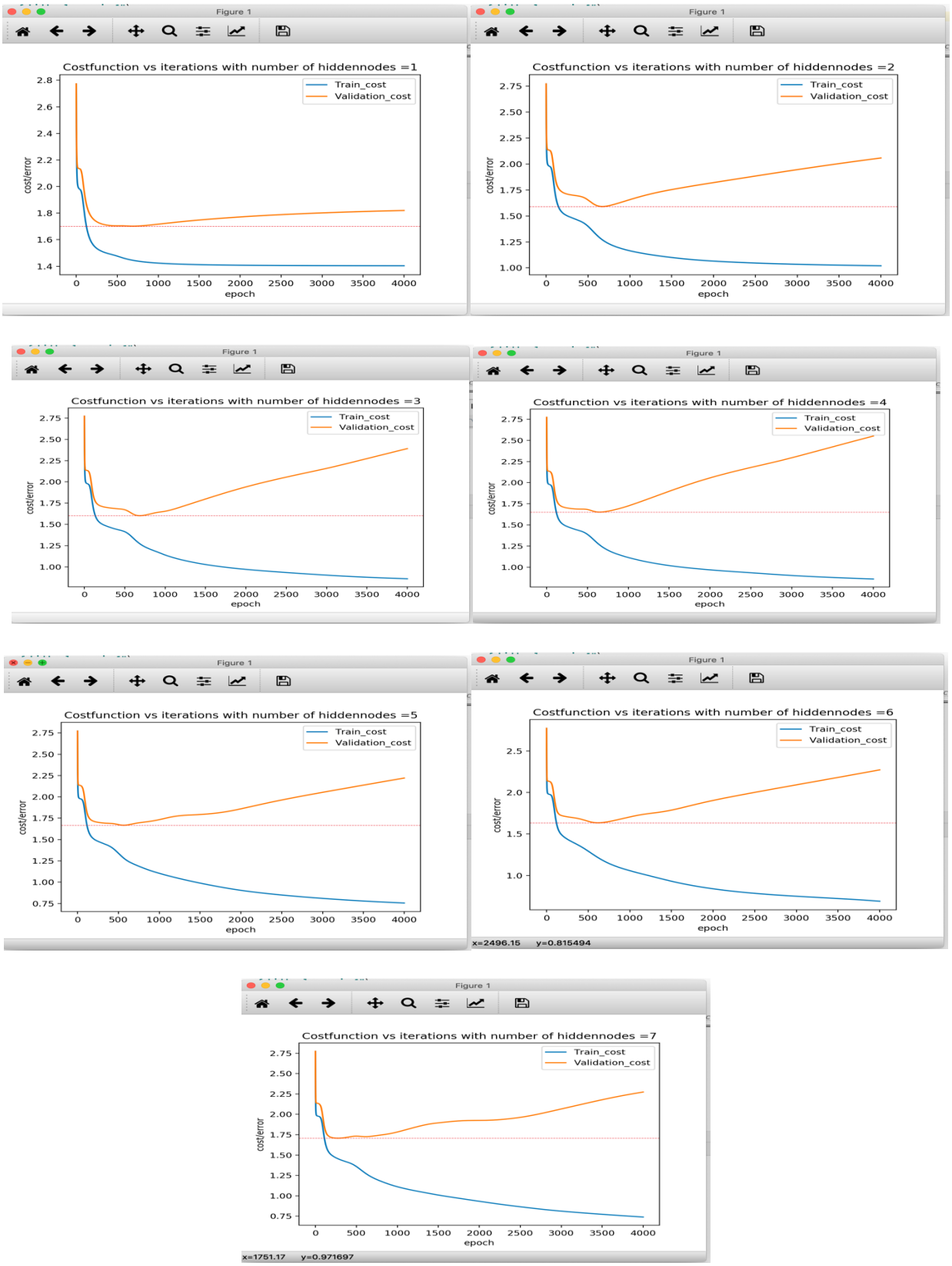


Figure 3 to 9: epoch vs error for neuron numbers 1,2,3,4,5,6,7 in single layer

c.Does adding more hidden layers improve performance? Compare at least 3 different configurations.

Solution: Performance measure is not improved when number of hidden layers is increased.

Note: Every hidden layer has same number of neurons.

From Table 2 as number of layers increase, the error is increasing for every addition of hidden layer with total number of neurons per model @each layer is 2,3,4. So, we can fix number of hidden nodes to 2 in each additional hidden layer. From Figure 12 and 13, cost function graph has reached saturation sooner with high error when number of hidden layers =3,4. In **Figure 10 and 11** we can observe nearly same minimal error, but for hidden nodes=2 and hidden layers = 2 **convergence is slow compared to model with hidden layer =1**. Hence **Model with single hidden layer with 2 hidden nodes is optimal here.**

neurons in each layer = 2 Number of hidden layers excluding output =	**Validation minimum cost**	**epoch**	**Weighted f1-score validation**	**Weighted f1-score train**
1	**1.5908250781037774**	**672**	**0.65**	**0.69**
2	1.6071847564288564	2198	0.65	0.75
3	2.137552865828954	23	0.22	0.29
4	2.1375475717526253	23	0.22	0.29
Neurons in each layer = 4 No of Hidden layers =				
1	1.6517269752438943	648	0.61	0.67
2	1.686585451530541	1987	0.65	0.72
3	2.1377319088832154	17	0.22	0.29
Neurons in each layer =3 No of Hidden layers=				
1	1.6018114223898001	686	0.65	0.67
2	1.6549265089471474	1984	0.65	0.73
3	2.1376544401253215	20	0.22	0.29

Table 2: Info about minimum cost and associated f1-scores for different number of neurons with different number of hidden layers

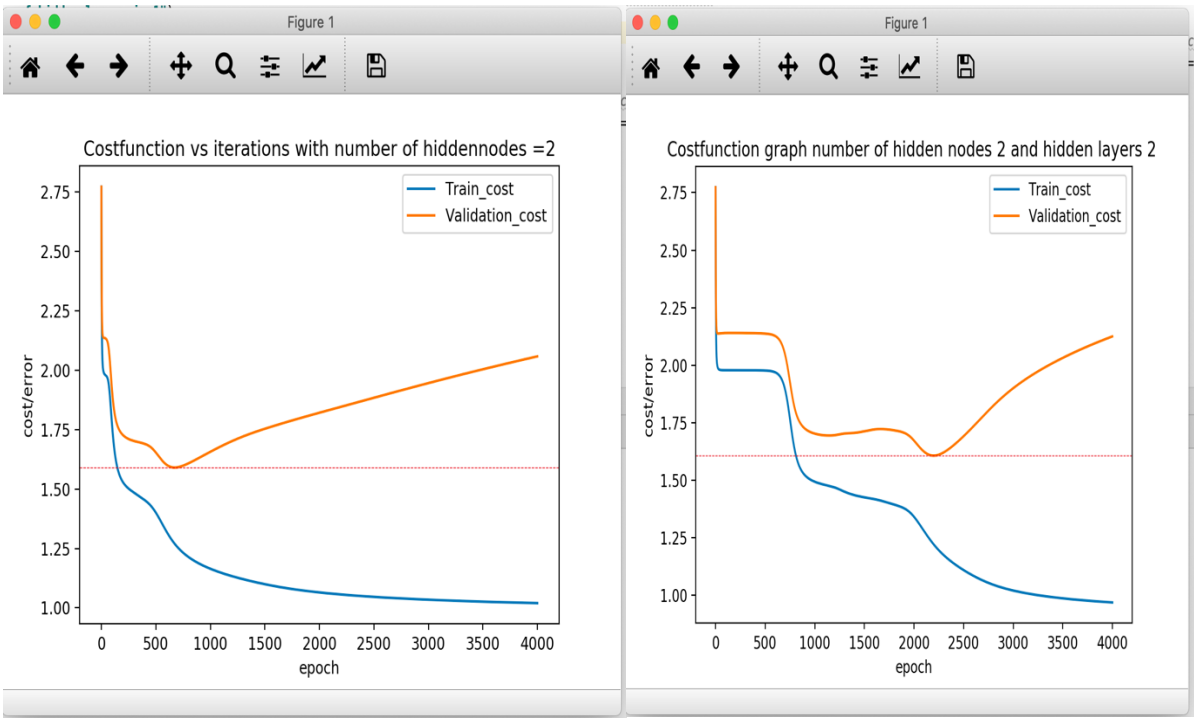


Figure 10 and 11

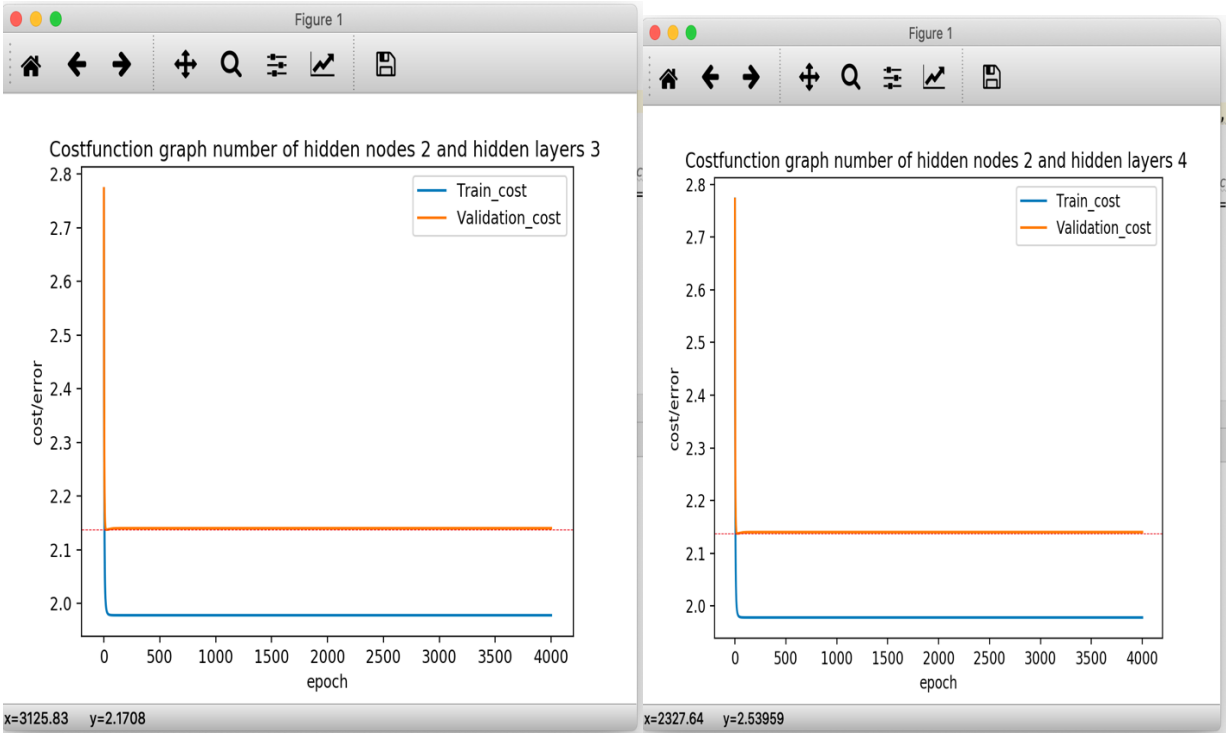


Figure 12 and 13

Figure 10 to 13: epoch vs error number of neurons= 2 and hidden layers=1,2,3,4

d.Evaluate performance for each case using metrics (such as confusion matrix, precision, recall, F1 scores and AUC). Which metrics are better in comparing model performances for this classification problem and why?

Solution: weighted average scores of precisions, recall and f1-scores are better in this classification problem.

Metric Analysis: The Nature of the given data is **highly imbalanced** in which samples of level 'D' is very low. Hence during classification if we want to give weights for class prediction ability based on number of data supports for the classes, we have to look at AUC of precision recall curve and weighted average precision, recall and f1-score in which class imbalances are handled. However, AUC here is calculated against TPR and FPR (1-specificity) (not weighted) which ignores other classes over majority class. So, we use weighted average to select best model for imbalanced data set.

Case 1(Model 1): 1 hidden layer with 5 hidden nodes alpha=0.5 epoch/ iterations=565.

Case 2(Model 2): 1 hidden layer with 2 hidden nodes alpha =0.5 epoch/iterations=672

We now evaluate the performance of the models from above scenarios on test data

Models Analysis: From figure 14 and 15 F1-score of Model 1 is 0.67, Model 2 is 0.67 and the model is not over trained (over fit) as train and test accuracy difference is lower. Model 1 predicted more B's correctly (figure 14 highlighted in green) than Model 2 but in predicting LEVEL C Model 2(figure 15 highlighted in green) did the better job. So, we consider overall score for evaluating the model performance.

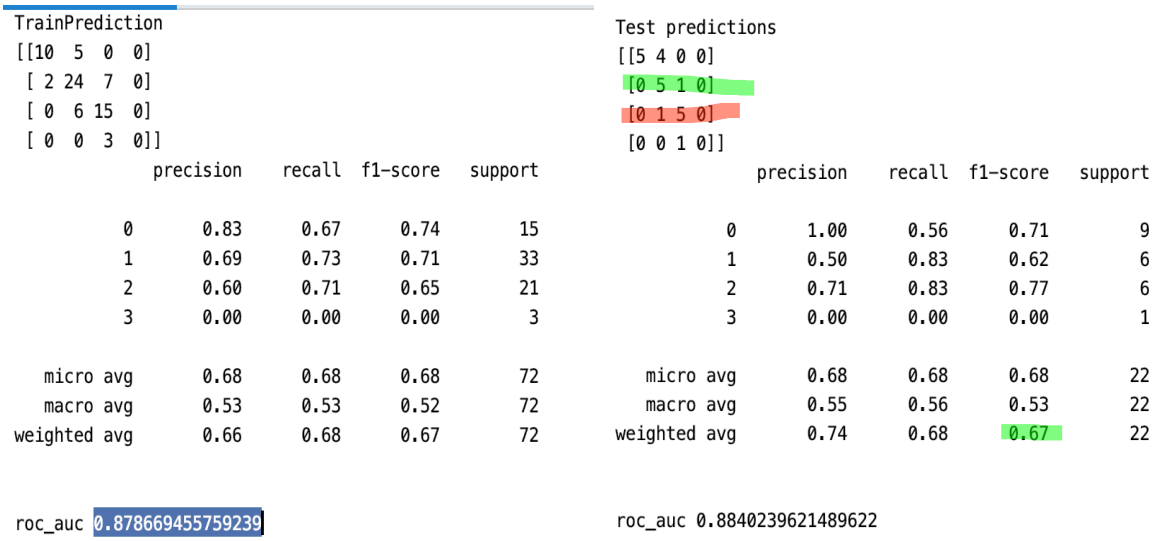
In Table 3 weighted F1-score of two Models differ by 0.02 and AUC differ by 0.01 only. The score doesn't differ much for single layer 2 hidden neuron Model and single layer 5 hidden neuron Model, however **parameter required to train for Model 1 is (25+24=49) and for Model 2 is (10+12=22).So preferred to choose "Model 2" with lower parameters (A single layer 2 hidden neurons) as "Best Model".**

Model	**Weighted average Precision**		**Weighted average recall**		**Weighted average f1-score**		**AUC**	
	Test	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**
Model 1	0.74	0.66	0.68	0.68	0.67	0.67	0.884023962148962	0.878669455759239
Model 2	0.73	0.69	0.68	0.71	0.67	0.69	0.891836462148962	0.878489491095494

Table 3: weighted average Precision, Recall and f1-score along with AUC on Test and Train data for Model 1 and Model 2

Confusion matrix and Precision, recall, f1 scores:

Model 1:



TrainPrediction

```
[[10  5  0  0]
 [ 2 24  7  0]
 [ 0  6 15  0]
 [ 0  0  3  0]]
```

	precision	recall	f1-score	support
0	0.83	0.67	0.74	15
1	0.69	0.73	0.71	33
2	0.60	0.71	0.65	21
3	0.00	0.00	0.00	3
micro avg	0.68	0.68	0.68	72
macro avg	0.53	0.53	0.52	72
weighted avg	0.66	0.68	0.67	72

roc_auc 0.878669455759239

Test predictions

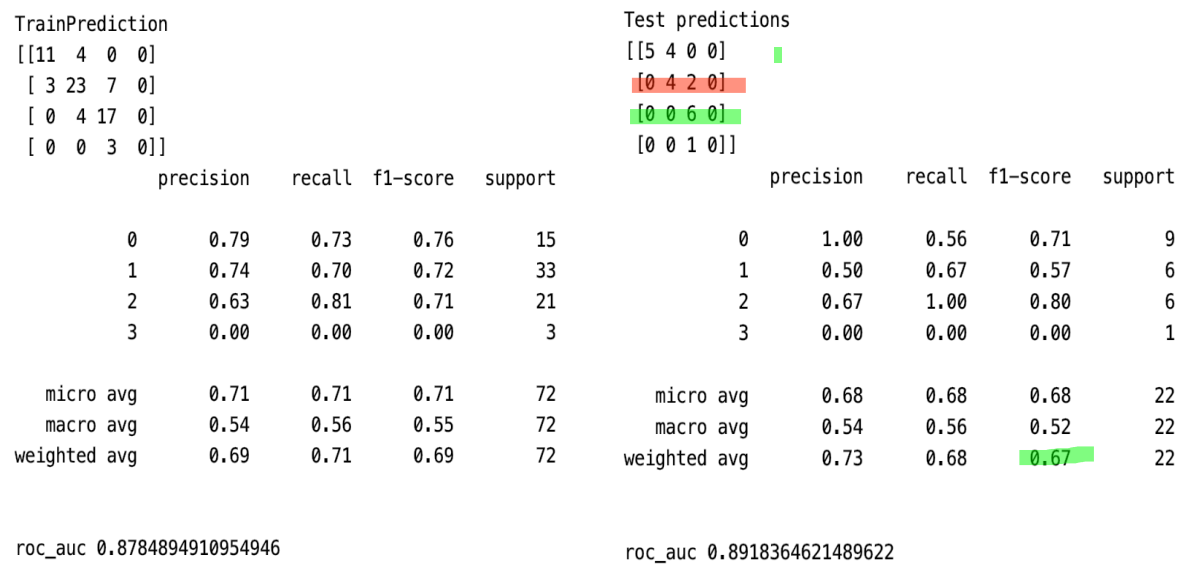
```
[[5 4 0 0]
 [0 5 1 0]
 [0 1 5 0]
 [0 0 1 0]]
```

	precision	recall	f1-score	support
0	1.00	0.56	0.71	9
1	0.50	0.83	0.62	6
2	0.71	0.83	0.77	6
3	0.00	0.00	0.00	1
micro avg	0.68	0.68	0.68	22
macro avg	0.55	0.56	0.53	22
weighted avg	0.74	0.68	0.67	22

roc_auc 0.8840239621489622

Figure 14: Confusion matrix, precision, recall, f1-score and auc value for single layer 5 hidden neuron Model

Model 2:



TrainPrediction

```
[[11  4  0  0]
 [ 3 23  7  0]
 [ 0  4 17  0]
 [ 0  0  3  0]]
```

	precision	recall	f1-score	support
0	0.79	0.73	0.76	15
1	0.74	0.70	0.72	33
2	0.63	0.81	0.71	21
3	0.00	0.00	0.00	3
micro avg	0.71	0.71	0.71	72
macro avg	0.54	0.56	0.55	72
weighted avg	0.69	0.71	0.69	72

roc_auc 0.8784894910954946

Test predictions

```
[[5 4 0 0]
 [0 4 2 0]
 [0 0 6 0]
 [0 0 1 0]]
```

	precision	recall	f1-score	support
0	1.00	0.56	0.71	9
1	0.50	0.67	0.57	6
2	0.67	1.00	0.80	6
3	0.00	0.00	0.00	1
micro avg	0.68	0.68	0.68	22
macro avg	0.54	0.56	0.52	22
weighted avg	0.73	0.68	0.67	22

roc_auc 0.8918364621489622

Figure 15: Confusion matrix, precision, recall, f1-score and auc value for single layer 2 hidden neuron Model

2. **Regularization (30 points):**

a. Does regularization improve the performance for the best model in Question 1? Test at least 5 different regularization values to support your answer.

Solution: Since the convergence condition / stop criteria in our experiment is epoch where validation cost is minimal. Regularization did not decrease the cost/error with increase in lambda. **Regularization did not improve the model performance** for given stop criteria.

Analysis: It is evident from Figure 17 (group of graphs for different values of lambda) that the overfit started post minimal validation cost epoch.Initially if we consider number of iterations the model should be trained to as 4000 ,from Figure 17 we can say that Regularization reduced overfit ,but did not improved the performace of model as the cost tend to increase with increase in lambda.From **Table 4 and Figure 16 the cost/error tend to increase on both train and validation with increase in lambda making model underfit at higher lambda**.Although overfit reduced with increase in lambda ,the cost/error also increasing(low accuracy). So, **Regularization did not improve the Model performance.**

lambda	Validation minimum cost	Stopping epoch for chosen lambda
10	2.1363034588826424	81
0.7	1.8467649060741864	1002
0.6	1.8274568358345802	999
0.5	1.8063122491393315	1067
0.2	1.7073835246919715	747
0.1	1.657476816486276	699
0	**1.5908250781037774**	672
0.01	1.598546261414624	674

Table 4: minimal cost/error for different values of lambda and epoch at which minimal is achieved.

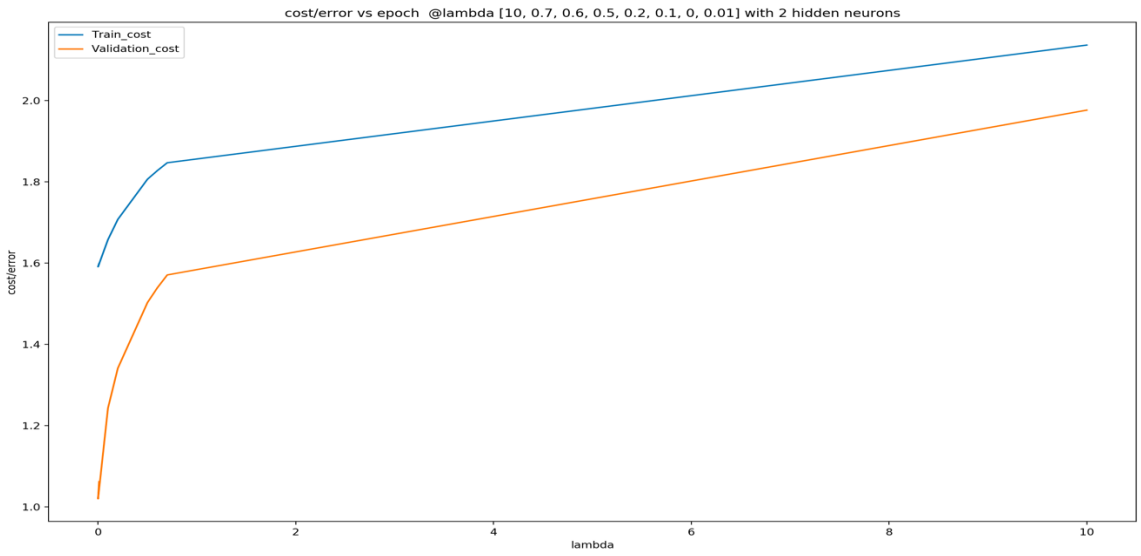


Figure 16: Trend of train and validation cost with increase in Lambda.

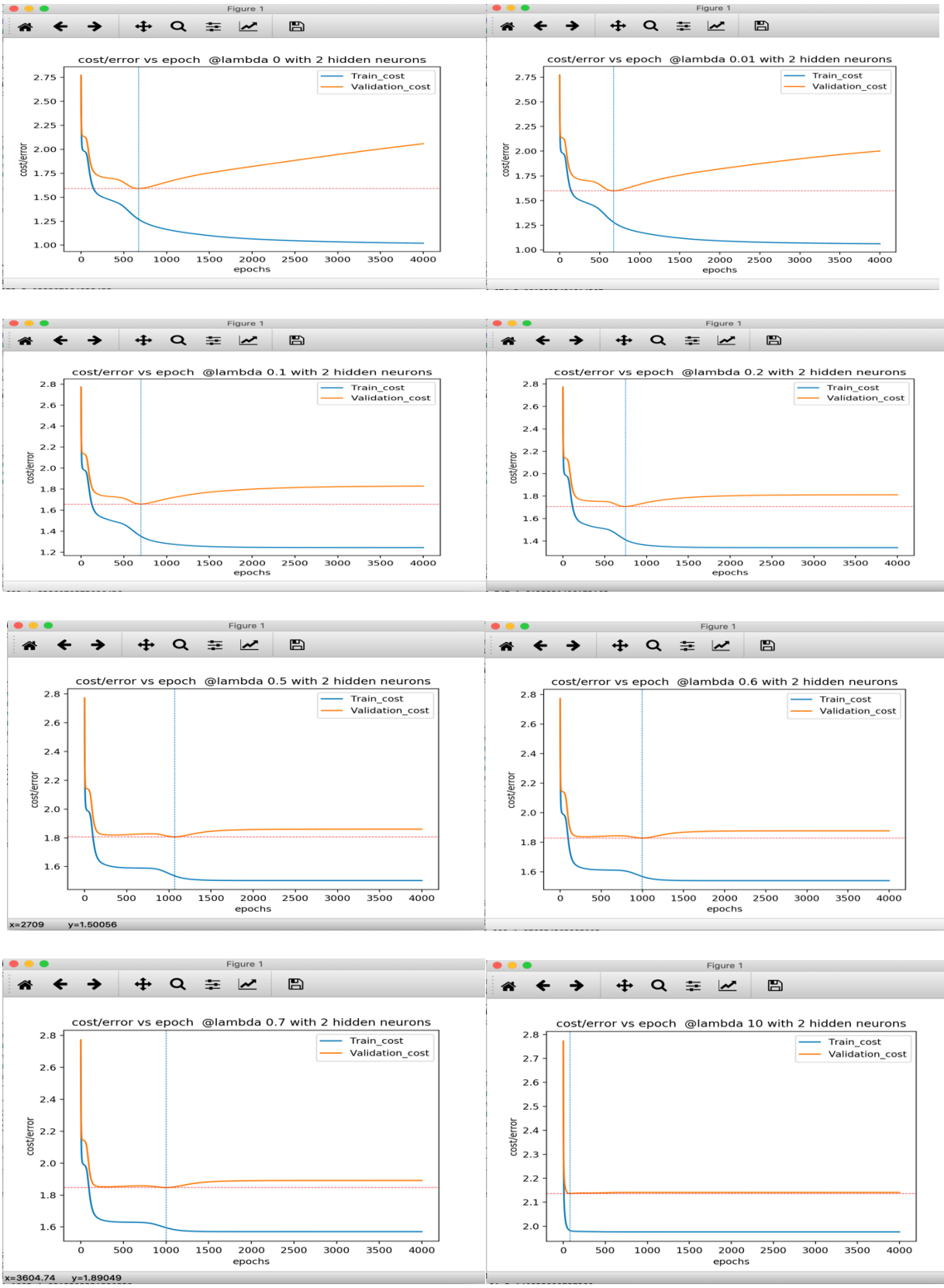


Figure 17(Group of 8 graphs) that show decrease in overfit and increase in error, with increase in Lambda at iteration 4000

b. Can you test the performance of your neural network using random initialization of weights vs. all weights equal to the same value? Is symmetry breakage a critical issue here?

Solution: Symmetric breakage is not a critical issue here.

Analysis: All Zero's and random initiation of weights on a network (single layer 2 hidden neurons) symmetric breakage is not a critical issue, because If the weights are zero, complexity of the whole deep net would be the same as that of a single neuron. Since our base model has only two neurons, optimal cost function or error might not have a huge difference which can be observed in Table 5 below.

For iterations 4000:

Weight initialization 2 hidden single layer	Minimum validation cost	Minimum @epoch
Zero initialization	1.6899912784791489	481
Random initialization	1.5908250781037774	672

Table 5: minimal cost/error for Zero (all weights) and random initiation of weights and their optimal convergence epoch.

Now we would further analyze the performance for two types of weights initialization:

From Figure 18 and 19 we can observe the train loss is no longer decreasing beyond optimal convergence point when all weights are initialized to zero compared to random weights. If we have taken iterations for being optimal as 4000 symmetric breakage could be a critical issue here. However, our optimal convergence condition (epoch @ minimal cost/error) is different, so symmetric breakage is not an issue. Table 6 shows the metric on train and validation date at optimal convergence epoch. It can be analyzed that **validation f1-scores are same but there is overfit in model with random initialization of weights if iterations = 4000**.

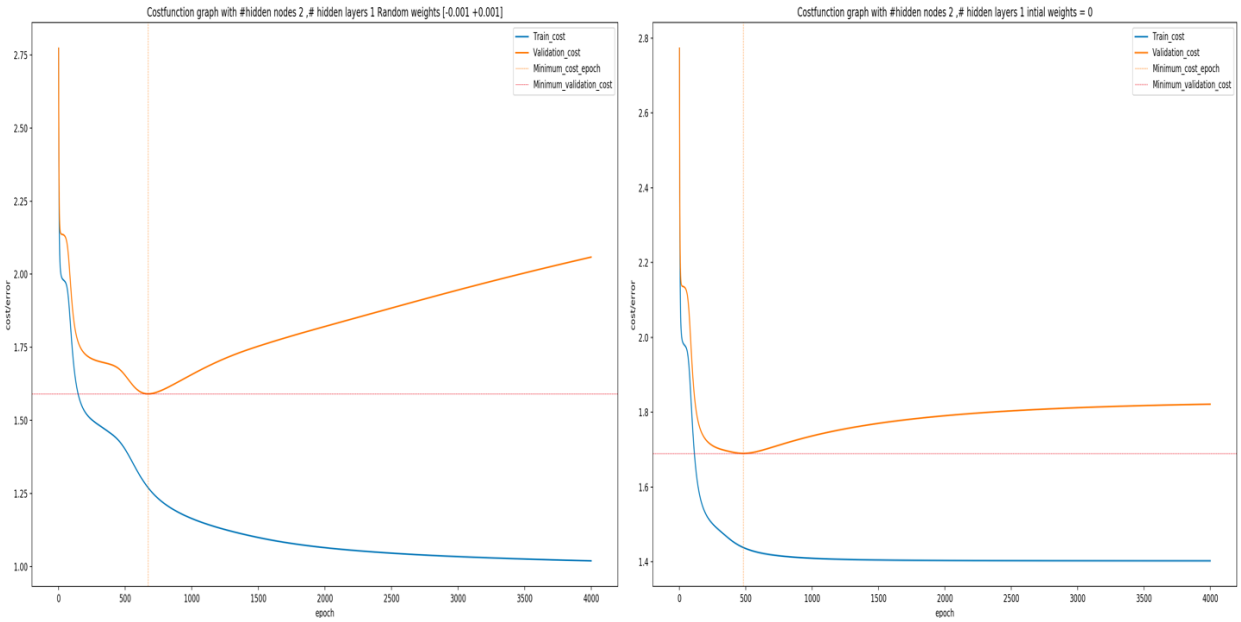


Figure 18: train & validation error vs epoch graph Random initialization weights [-0.001 to 0.001], Figure 19: All zero weights initialized error graph

Weights initialization	Weighted average Precision		Weighted average recall		Weighted average f1-score	
2 hidden neurons	**Validation**	**Train**	**Validation**	**Train**	**Validation**	**Train**
Random initialization [-0.001 to +0.001]	0.74	0.79	0.67	0.79	0.66	0.78
All Zero initialization [0 0]	0.72	0.67	0.67	0.69	0.66	0.68

Table 6: weighted metrics on train and validation.

c. Evaluate performance for each case using metrics (such as confusion matrix, precision, recall, F1 scores and AUC).

Evaluation on test data with minimum epoch as stop criteria. From Figure 19 and 20, the train confusion matrix of model with equal weights is able to predict more A's compared to case 1. But have more False Negative of B's than the model with random weights (case 1). But the generalization prediction of LEVELs A, B, C, D on test data is same for both type of weight initialization.

Case1: Non regularized random initialization (too small weights)

TrainPrediction

```
[[11  4  0  0]
 [ 3 23  7  0]
 [ 0  4 17  0]
 [ 0  0  3  0]]
```

	precision	recall	f1-score	support
0	0.79	0.73	0.76	15
1	0.74	0.70	0.72	33
2	0.63	0.81	0.71	21
3	0.00	0.00	0.00	3
micro avg	0.71	0.71	0.71	72
macro avg	0.54	0.56	0.55	72
weighted avg	0.69	0.71	0.69	72

roc_auc 0.8784894910954946

Test predictions

```
[[5 4 0 0]
 [0 4 2 0]
 [0 0 6 0]
 [0 0 1 0]]
```

	precision	recall	f1-score	support
0	1.00	0.56	0.71	9
1	0.50	0.67	0.57	6
2	0.67	1.00	0.80	6
3	0.00	0.00	0.00	1
micro avg	0.68	0.68	0.68	22
macro avg	0.54	0.56	0.52	22
weighted avg	0.73	0.68	0.67	22

roc_auc 0.8918364621489622

Figure 19: Confusion matrix, precision, recall and f1-score for train and validation (random weight initialization)

Case 2: Non regularized zero initialization (same weights)

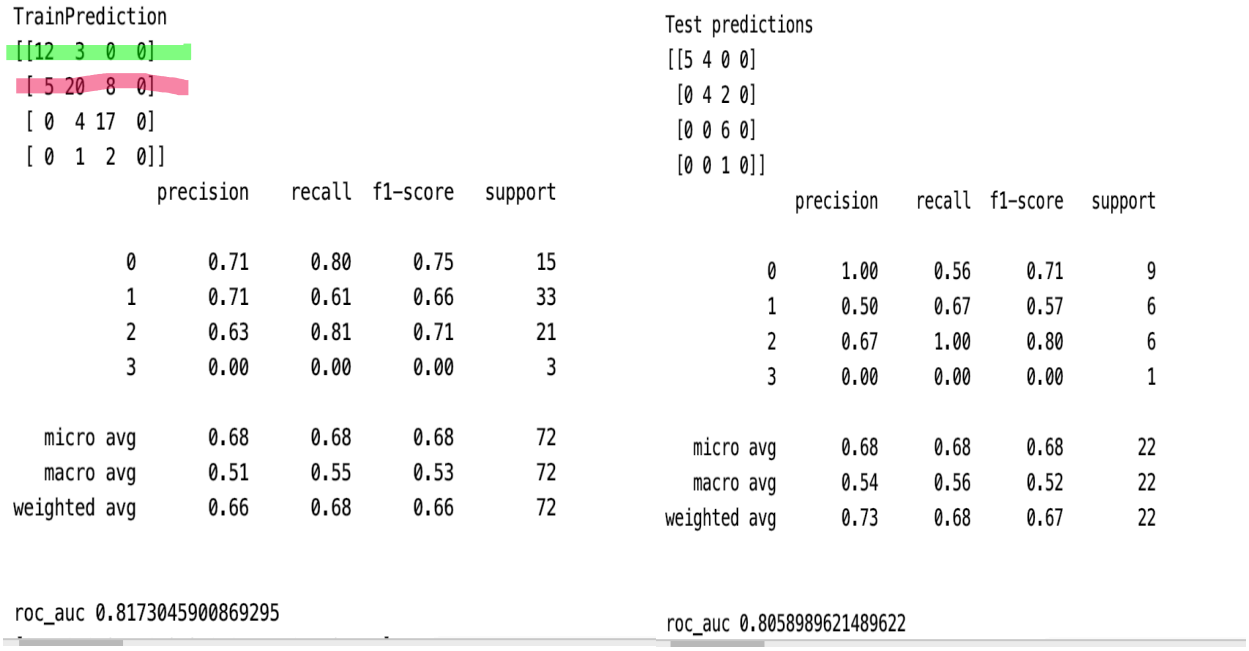


Figure 20: Confusion matrix, precision, recall and f1-score for train and validation (equal weight initialization =0)

Weights initialization	**Weighted average Precision**		**Weighted average recall**		**Weighted average f1-score**		**AUC**	
2 hidden neurons	Test validation	Train	Test validation	Train	Test validation	Train	Test validation	Train
Random initialization [-0.001 to +0.001]	0.72	0.79	0.68	0.79	0.67	0.78	0.8647931929181929	0.8900541524851661
	0.74		0.67		0.66		0.8307019777608013	
All Zero initialization [0 0]	0.73	0.67	0.68	0.69	0.67	0.68	0.8058989621489622	0.8085839841 79623
	0.72		0.67		0.65		0.7615822086410322	

Table 7: weighted metrics on train and test

From Table 7, In both the cases model does not overfit. There are variations in train metrics as stop criteria for each case is different but the performance on test data is same for both cases of weight initialization. The model accuracy is 67% for case 1 and case 2. However, in general it is good practice not to use zero weights initialization as gradients would undergo the exact same parameter updates making the concept of deep network useless. So, we choose model with **random initialization of weights (low weights) as best model for further exploration in rest of the analysis below.**

3. Neural Network with More Variables (20 points):

a. Add one more variable that you think might improve the performance for the best model in Question 2.

Initial model: Single layer 2 neuron network.

Initial Features: (`'all_mcqs_avg_n20'`,`'all_NBME_avg_n4'`,`'CBSE_01'`,`'CBSE_02'`)

Feature selection: Using **spearman correlation coefficien**t to find correlation with the target and other features. Randomly selecting 4 variables and finding correlation of the variables and choosing one which has high correlation with target and low correlation with other feature variables.

Randomly selected variables:1. HA_final, 2. HD_final ,3.SA_NBME. 4.STEP_1

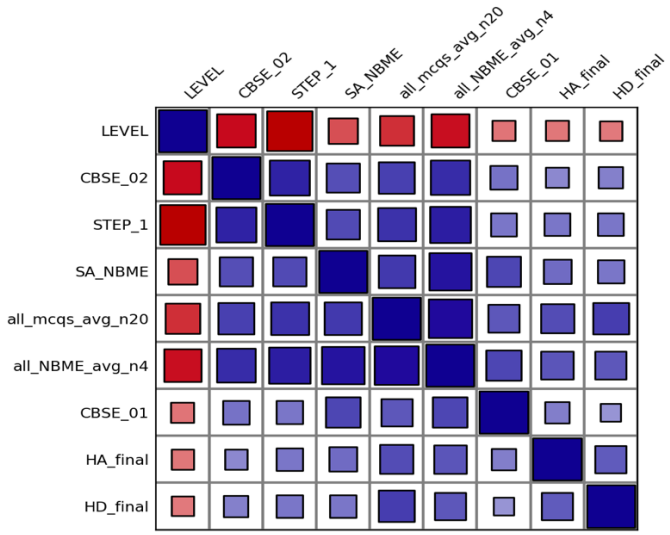


Figure 21: The correlation plot between target and features using Spearman

From the Figure 21 "**STEP_1" and "SA_NBME" has more correlation with target 'LEVEL'** variable among 4 randomly selected variable. Now we try to see if these selected variables have low correlation with other independent features.

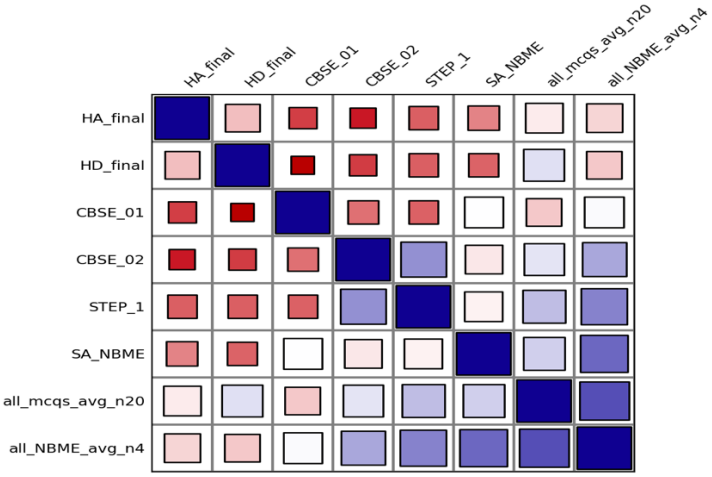


Figure 22: The correlation plot between feature variables

From Figure 22, HA_final and HD_final have less correlation with remaining features however from Figure 21 they have low correlation with target 'LEVEL' .In addition to this "STEP_1" and "SA_NBME" have more correlation with "all_NBME_avg_n4" only. They also have comparatively low correlation with other features so we can consider taking these two variables to build a model.

Model: 2 hidden layers with features
'all_mcqs_avg_n20', 'all_NBME_avg_n4', 'CBSE_01', 'CBSE_02','STEP_1', 'LEVEL'

Analysis: Adding variable 'STEP_1' performed better as cost is decreased which can be observed from table 7. However, when analyzed over validation vs Train from table 8((highlighted in green) it is clear that the Model is moving towards overfit. This can be avoided with regularization or early spotting than optimal convergence epoch 575.

Model	Cost function	epoch
Model 1 with 4 variables	1.5908250781037774	672
Model 2 with 5 variables 'STEP_1'	1.2684040978958793	575

Table 7: values of cost function for given and selected features

	Weighted average Precision		Weighted average recall		Weighted average f1-score	
2 hidden neurons	**Validation**	**Train**	**Validation**	**Train**	**Validation**	**Train**
'all_mcqs_avg_n20','all_NBME_avg_n4','CBSE_01','CBSE_02'	0.72	0.69	0.67	0.71	0.65	0.69
'all_mcqs_avg_n20','all_NBME_avg_n4','CBSE_01', 'CBSE_02','STEP_1'	0.79	0.92	0.78	0.96	0.75	0.94
'all_mcqs_avg_n20','all_NBME_avg_n4','CBSE_01','CBSE_02','STEP_1','SA_NBME'	0.79	0.92	0.78	0.96	0.75	0.94

Table 8: Train vs validation weighted f1-score metrics for given and selected features

b. Add another variable that you think might improve the performance into the model in Question 3a.

Analysis: Now add 'SA_NBME' to the model in 3a.

Model	cost function	Stop criteria
Model 1 with 4 variables	1.5908250781037774	672
Model 2 with 5 variables 'STEP_1'	**1.2684040978958793**	**575**
Model 2 with 6 variables 'STEP_1' and 'SA_NBME'	**1.2263942970754218**	**635**

Table 8: values of cost function for given and selected features

By adding another variable, From Table 8 the cost is decreased which is minimal among two earlier models. So, we can say that 'SA_NBME' contributed to model improvement. From table 8(highlighted in yellow) it is clear, that even this model is moving towards overfit. This can be avoided with regularization or early spotting than optimal convergence epoch 635.

compare performances of three models using metrics (such as confusion matrix, precision, recall, F1 scores and AUC).

Comparing three models:

From below table, we can observe the model performance is improved w.r.t test data. However, there is a trend towards overfitting. We can do early stopping or use regularization for overfit models.

From Figure 23 Model 2 has improved performance as it predicted LEVEL B correctly more when compared with Model 1 hence there is change in f1-score. So, we can say model 2 has 71% accuracy/f1-score.

Model 3 has improved performance as it predicted a greater number of LEVEL A correctly when compared to Model 1 and Model 2. Prediction on LEVEL B is same as Model 2. So final model accuracy/f1-score is 76%.

	Weighted average Precision		**Weighted average recall**		**Weighted average f1-score**		**AUC**	
2 hidden neurons	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**
'all_mcqs_avg_n20, 'all_NBME_avg_n4', 'CBSE_01', 'CBSE_02'	0.73	0.69	0.68	0.71	0.67	0.69	0.891836462148962	0.8784894910954946
'all_mcqs_avg_n20, 'all_NBME_avg_n4', 'CBSE_01, 'CBSE_02','STEP_1'	0.77	0.92	0.73	0.96	0.71	0.94	0.96875	0.9850998674528086
'all_mcqs_avg_n20','all_NBME_avg_n4','CBSE_01','CBSE_02','STEP_1','SA_NBME'	0.78	0.92	0.77	0.96	0.76	0.94	0.964009081196581	0.9852941176470589

Conclusion: Finally, we can conclude that the addition of new variables improved performance of model but failed to make prediction on LEVEL D.

Model 1:

TrainPrediction

```
[[11  4  0  0]
 [ 3 23  7  0]
 [ 0  4 17  0]
 [ 0  0  3  0]]
```

	precision	recall	f1-score	support
0	0.79	0.73	0.76	15
1	0.74	0.70	0.72	33
2	0.63	0.81	0.71	21
3	0.00	0.00	0.00	3
micro avg	0.71	0.71	0.71	72
macro avg	0.54	0.56	0.55	72
weighted avg	0.69	0.71	0.69	72

roc_auc 0.8784894910954946

Test predictions

```
[[5 4 0 0]
 [0 4 2 0]
 [0 0 6 0]
 [0 0 1 0]]
```

	precision	recall	f1-score	support
0	1.00	0.56	0.71	9
1	0.50	0.67	0.57	6
2	0.67	1.00	0.80	6
3	0.00	0.00	0.00	1
micro avg	0.68	0.68	0.68	22
macro avg	0.54	0.56	0.52	22
weighted avg	0.73	0.68	0.67	22

roc_auc 0.8918364621489622

Model 2:

TrainPrediction

```
[[15  0  0  0]
 [ 0 33  0  0]
 [ 0  0 21  0]
 [ 0  0  3  0]]
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	15
1	1.00	1.00	1.00	33
2	0.88	1.00	0.93	21
3	0.00	0.00	0.00	3
micro avg	0.96	0.96	0.96	72
macro avg	0.72	0.75	0.73	72
weighted avg	0.92	0.96	0.94	72

roc_auc 0.9850998674528086

Test predictions

```
[[5 4 0 0]
 [0 5 1 0]
 [0 0 6 0]
 [0 0 1 0]]
```

	precision	recall	f1-score	support
0	1.00	0.56	0.71	9
1	0.56	0.83	0.67	6
2	0.75	1.00	0.86	6
3	0.00	0.00	0.00	1
micro avg	0.73	0.73	0.73	22
macro avg	0.58	0.60	0.56	22
weighted avg	0.77	0.73	0.71	22

roc_auc 0.96875

Model 3:

TrainPrediction

```
[[15  0  0  0]
 [ 0 33  0  0]
 [ 0  0 21  0]
 [ 0  0  3  0]]
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	15
1	1.00	1.00	1.00	33
2	0.88	1.00	0.93	21
3	0.00	0.00	0.00	3
micro avg	0.96	0.96	0.96	72
macro avg	0.72	0.75	0.73	72
weighted avg	0.92	0.96	0.94	72

roc_auc 0.9852941176470589

Test predictions

```
[[6 3 0 0]
 [0 5 1 0]
 [0 0 6 0]
 [0 0 1 0]]
```

	precision	recall	f1-score	support
0	1.00	0.67	0.80	9
1	0.62	0.83	0.71	6
2	0.75	1.00	0.86	6
3	0.00	0.00	0.00	1
micro avg	0.77	0.77	0.77	22
macro avg	0.59	0.62	0.59	22
weighted avg	0.78	0.77	0.76	22

roc_auc 0.9640090811965812

Figure 23: confusion matrix, precision, recall, f1-score for All three Models