COL774 Assignment 3

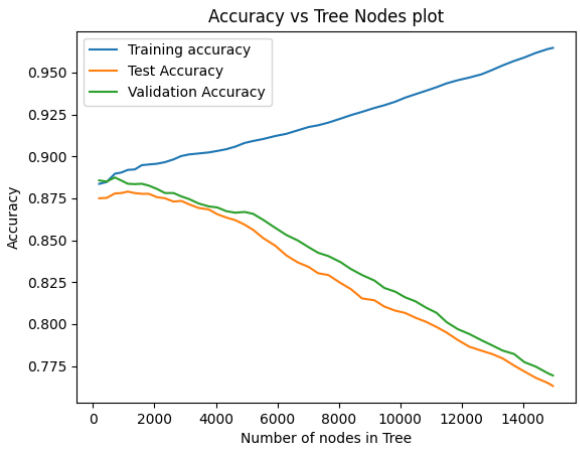
Decision Trees:

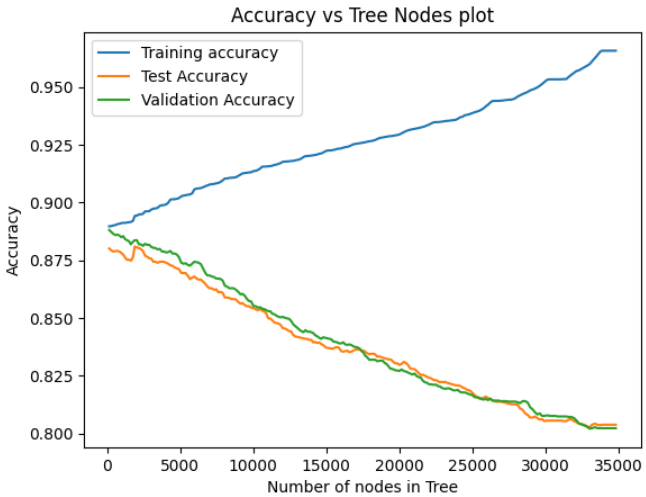
1. (a) In part a, we have done classification using both oneHot encoding, and multiLabel classification. It was found that the tree has higher number of nodes in case of oneHot encoding due to larger number of classes. As a result, accuracy obtained for oneHotEncoding is better than for multiclass classification.

For Multiclass classification:
Train accuracy: 0.9647201946472019
Validation accuracy: 0.7693498452012384
Test accuracy: 0.7631055076310551

For OneHot classification:
Train accuracy: 0.965743198407432
Validation accuracy: 0.8022998673153472
Test accuracy: 0.8038044680380447

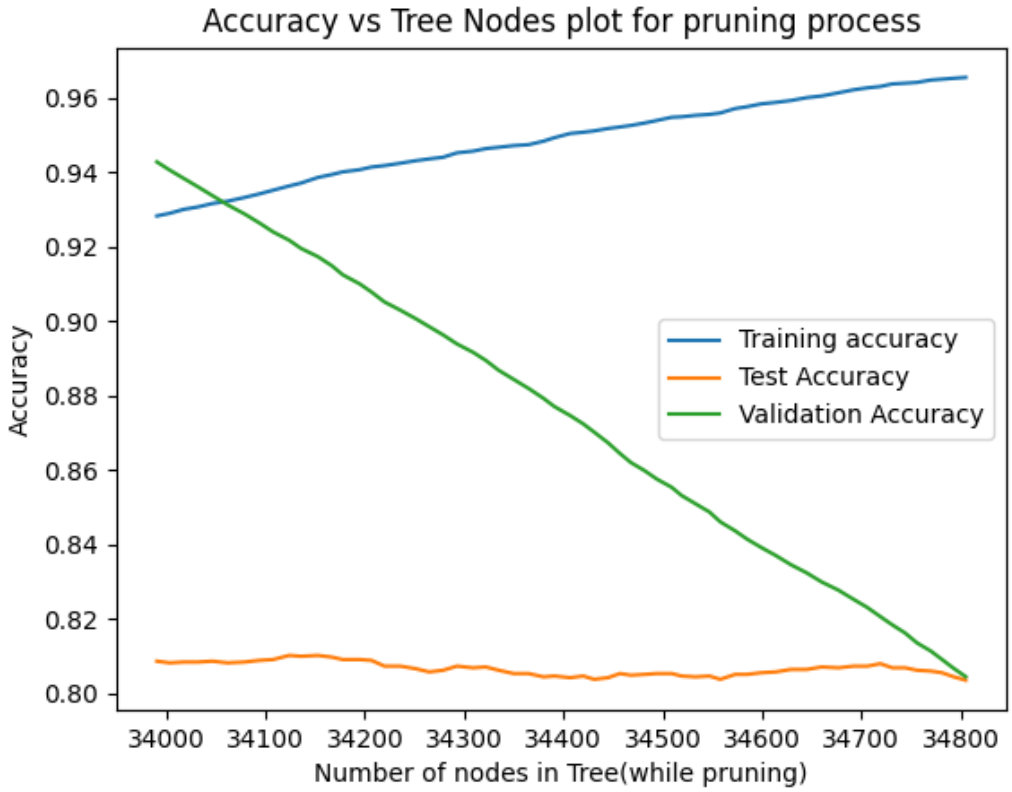
From the graphs below, we see that the training accuracy increases as the tree is grown while the testing, validation accuracies decrease after slightly increasing. This shows that we are massively overfitting our decision tree and some other optimizations such as early stopping, pruning etc. should be performed to improve out validation/test accuracies.





The left graph is for MultiClassification, while the right one is for OneHotEncoding.

(b) Post pruning was done in bottom up manner(post-order traversal). A node is pruned if for the validation set, it gives number of accurate classifications as leaf node rather than an internal node. From the graph below, we observe that the testing accuracy does not change, while the validation accuracy increases continuously and the training accuracy decreases. This is because, we are pruning the tree according to the validation data. Since, number of nodes split according to training data are being reduced, training accuracy decreases. There is slight increase in test accuracy indicating that pruning does not cause significant improvements on unseen data.



The above graph should be viewed from right to left, as pruning decreases nodes. We see a significant change in validation accuracy because pruning Is allowed to proceed till the topmost level.

Before pruning (using OneHot)
Train accuracy: 0.965743198407432
Validation accuracy: 0.8022998673153472
Test accuracy: 0.8038044680380447

After pruning:
Final Train Accuracy: 0.9282238442822385
Final Validation Accuracy: 0.9427244582043344
Final Test Accuracy: 0.8086706480867065

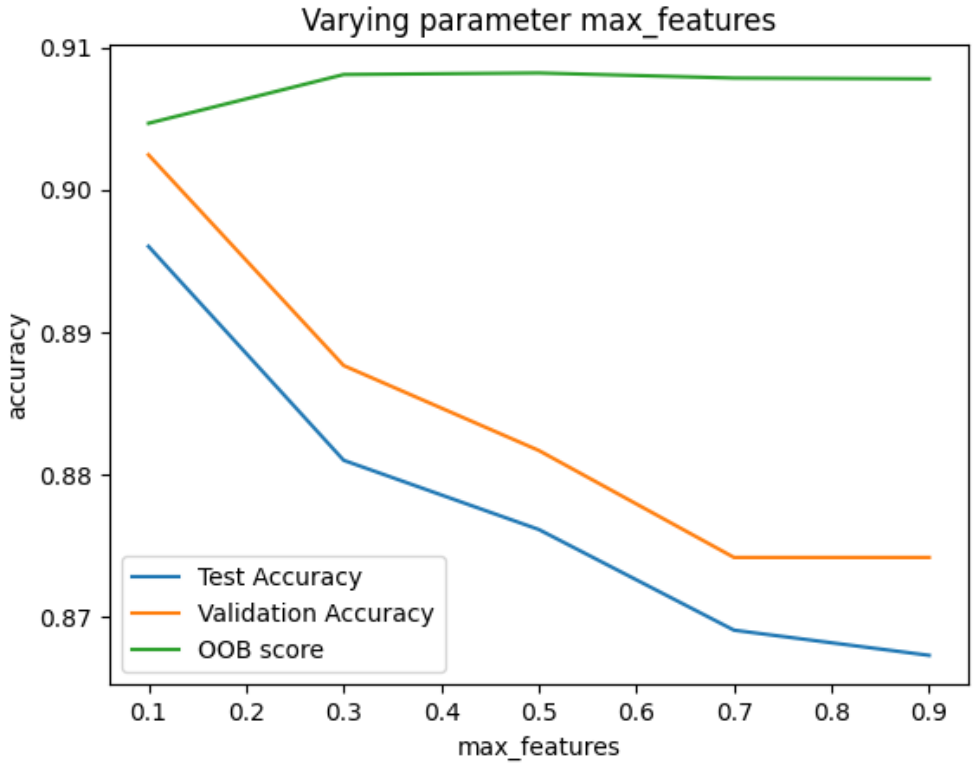
(c) The results obtained using random forest classifier are much better than using simple Decision Trees because we are using fraction of features for different trees which makes it less prone to overfit. Also, using large number of trees makes the model more robust and less sensitive to high variances in data.

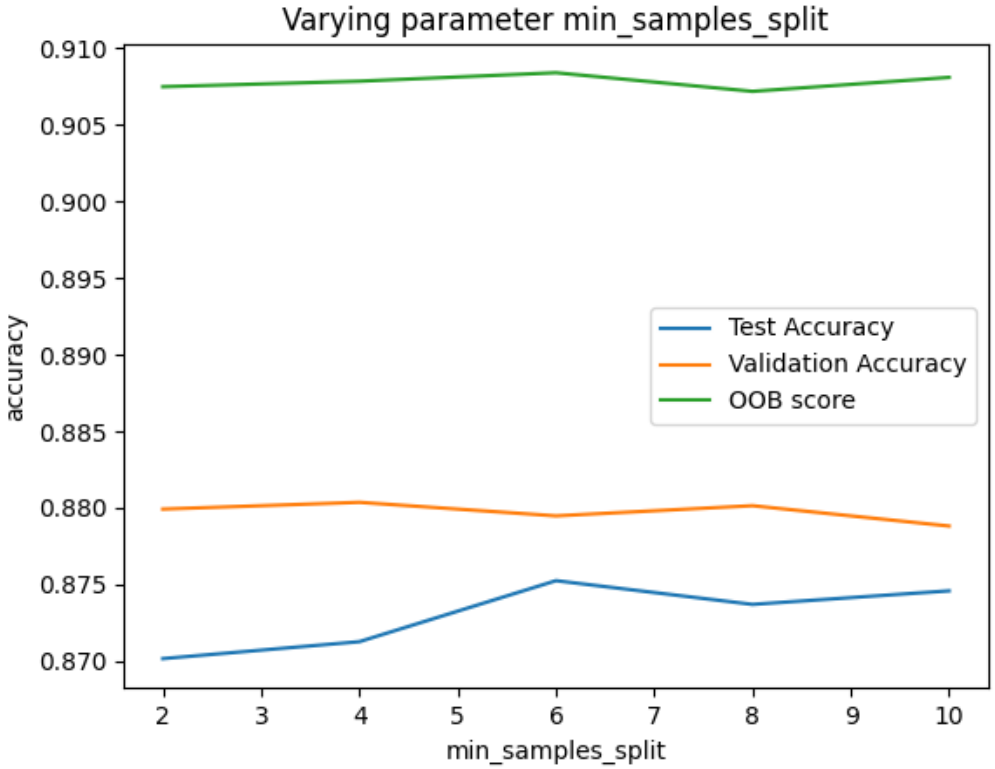
For this part, after performing grid search on 125 possible combinations of decision tree parameters. (n_estimators: {50, 150, 250, 350, 450}, max_features: {0.1, 0.3, 0.5, 0.7, 0.9}, min_samples_split: {2, 4, 6, 8, 10}). Best parameters obtained are:

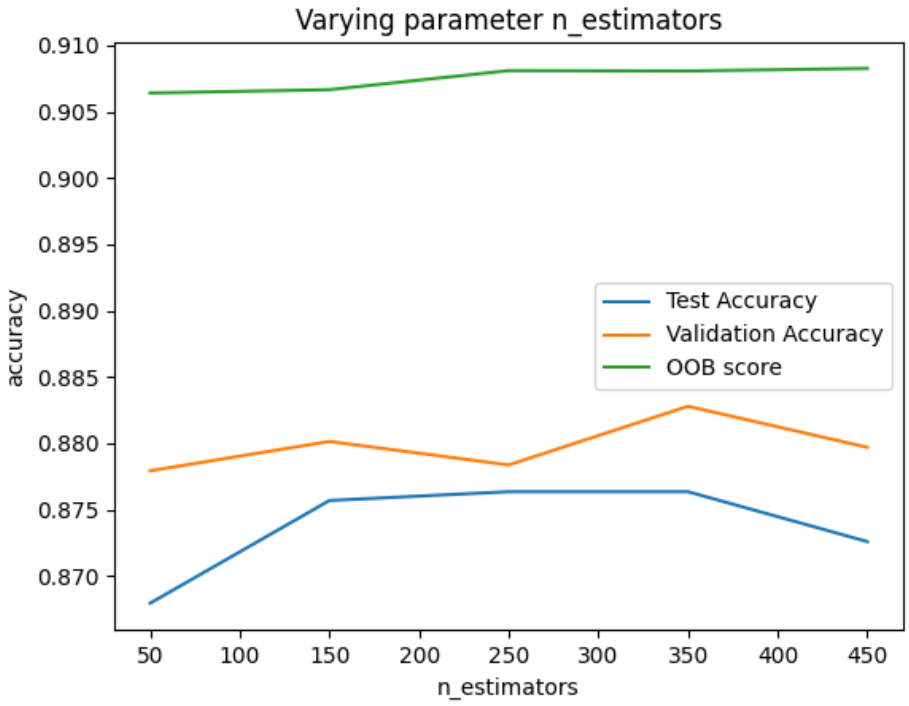
n_estimators: 350, max_features: 0.5, min_samples_split: 10
Best Parameter OOB Score: 0.9088697190886972
Training Accuracy: 0.9780745410307454
Test Accuracy: 0.8756912187569121
Validation Accuracy: 0.883016364440513

As we can see, the accuracies are much higher. Another important thing to note is that OOB score is close to the actual validation/test accuracy. This is because OOB score is calculated on unseen train data while training Decision Trees. This helps us not overfit our models and perform well on other unseen test data

(d)
Keeping Base parameters as n_estimators: 350, max_features: 0.5, min_samples_split: 10
We vary all 3 parameters once keeping other 2 values the same. The value ranges are same as in part (c).







From the above graphs, we can see that the peaks are still at n_estimators: 350, min_samples_split: 6, max_features: 0.5 for OOB score parameter. However, we see that the accuracies are most sensitive to change in min_features, i.e. the number of features to consider in each individual tree. Considering minimum number of features 0.1 gives best results on train, test sets because even moderately high values cause overfitting. For min_samples_split(minimum number of samples reaching a node to consider splitting it), we see roughly no changes in accuracy with min_samples_split as well as for n_estimators apart from random variations.

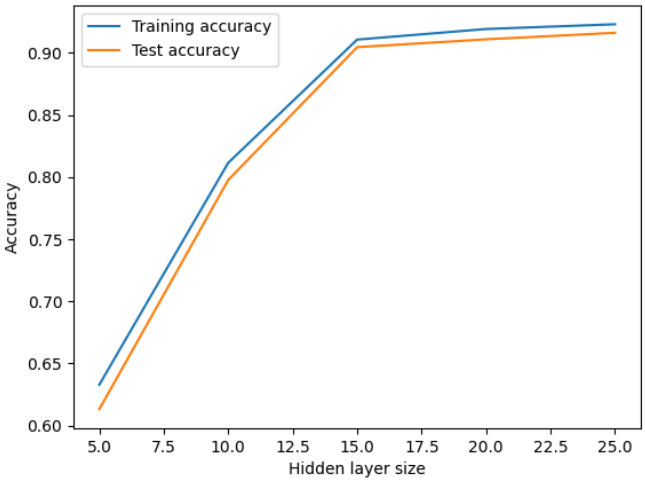
Neural Networks:

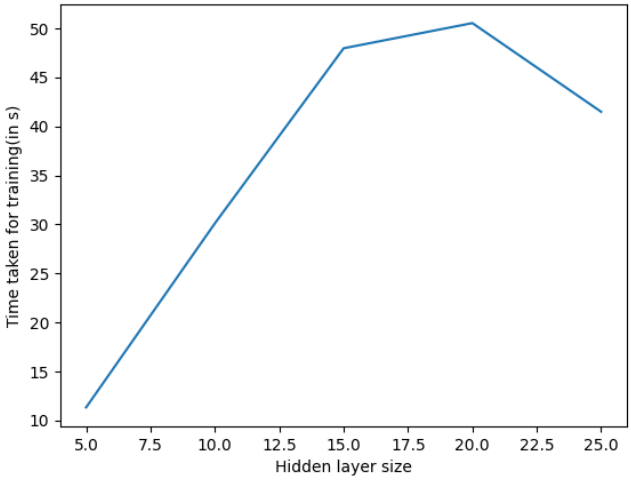
(a)One Hot Encoding has been done to preprocess inputs.

(b) We have initialized our weights according to He initialization. W and b vectors have been takn separately for ease in writing functions and equations. Vectorization has been used wherever possible.

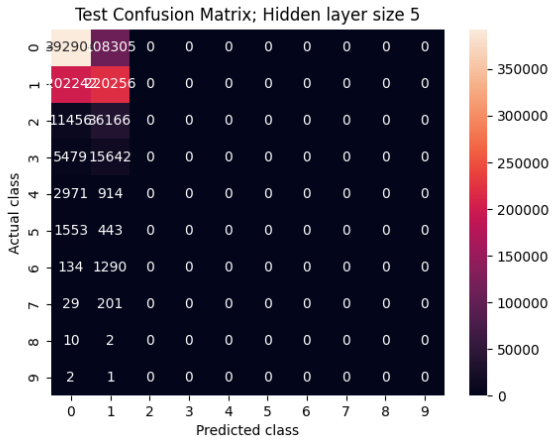
The neural networks are based upon SGD. Initialization of minibatch size M, hidden_layer_sizes, input features n, output classes r and activation(relu or sigmoid) is done.

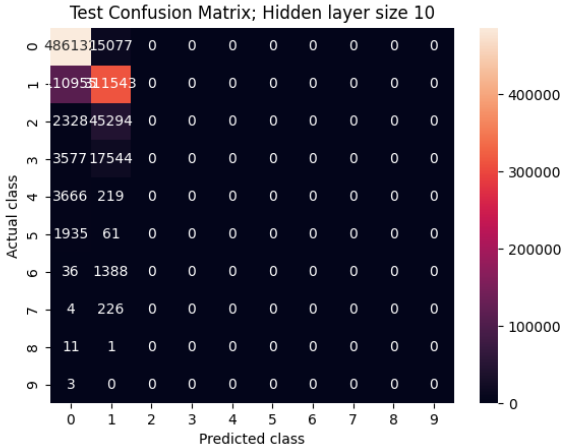
(c) On using M = 100, learning_rate = 0.1(fixed), plots obtained are shown below

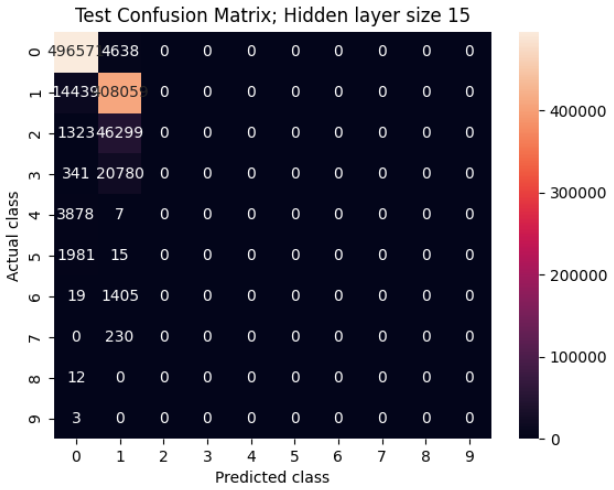


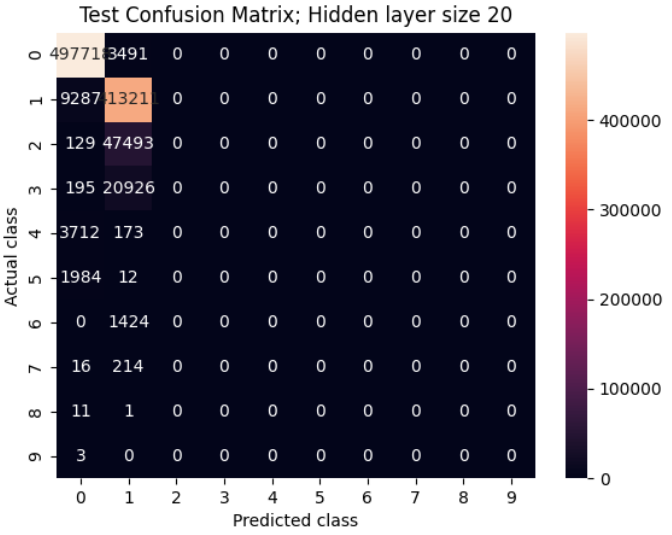


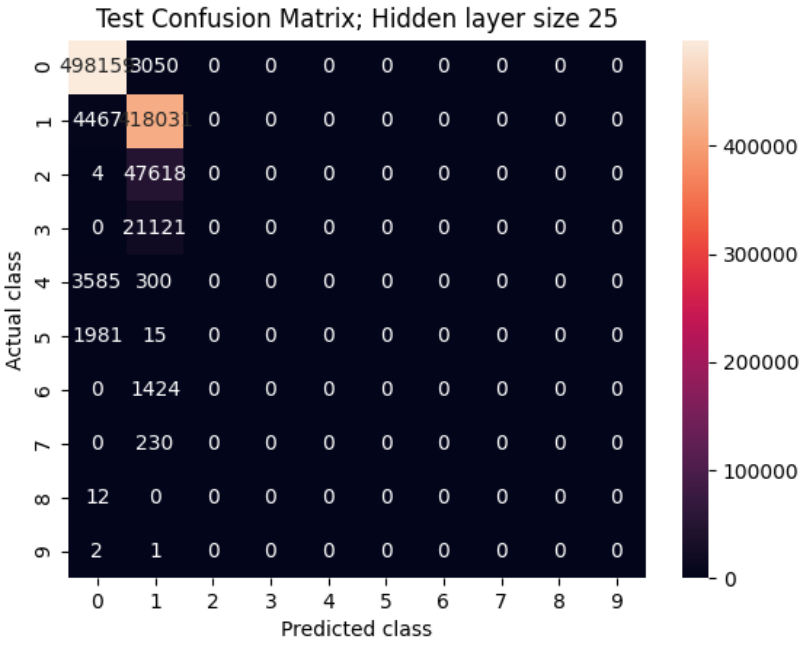
Confusion matrices:











As we can see from the confusion matrices, the data is always predicted between with class 0 or 1. This is because our hidden layer sizes are not large enough to get good results. More number of hidden layers classify more number of examples in the correct class for class 0, 1.

The results obtained are:
Hidden_layer_size: 5
Epoch 318/10000
Training accuracy: 0.6328668532586965
Test accuracy: 0.61316
Time to Train: 11.3383s

Hidden_layer_size: 10
Done epoch 807/10000
Training accuracy: 0.811515393842463
Test accuracy: 0.797675
Time to Train: 30.1122s

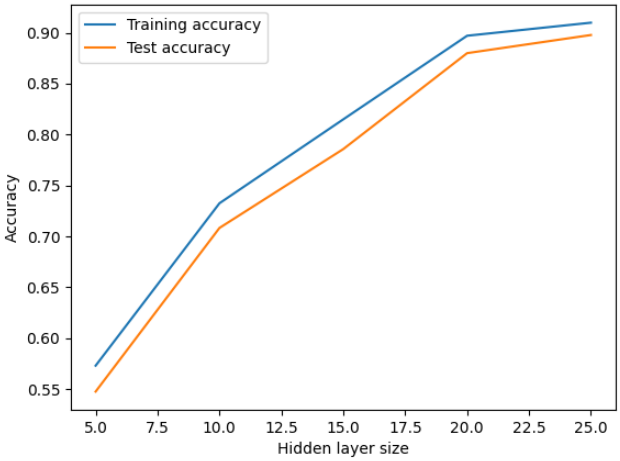
Hidden_layer_size: 15
Done epoch 1080/10000
Training accuracy: 0.9107157137145142
Test accuracy: 0.90463
Time to Train: 47.9712s

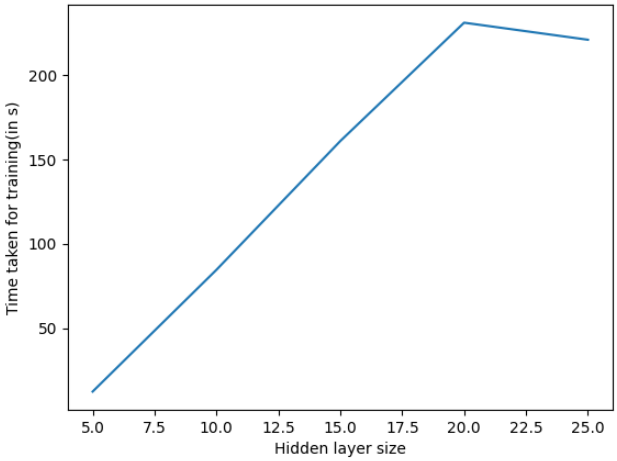
Hidden_layer_size: 20
Done epoch 1177/10000
Training accuracy: 0.9192323070771692
Test accuracy: 0.910929
Time to Train: 50.5404s

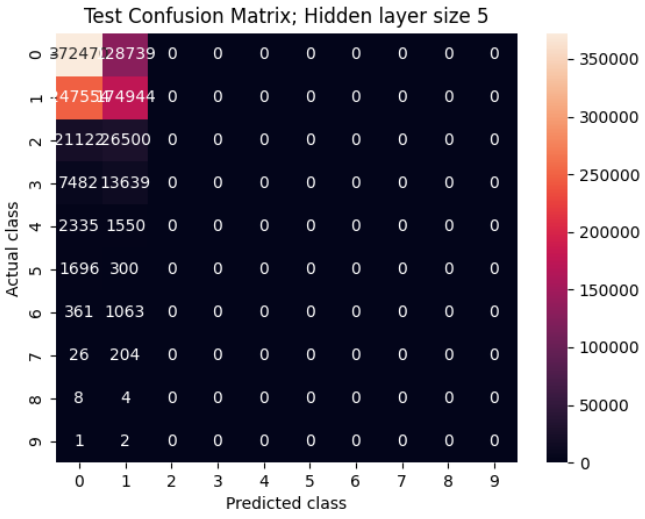
Hidden_layer_size: 25
Done epoch 909/10000
Done Iteration 5/5
Training accuracy: 0.9230707716913235
Test accuracy: 0.91619
Time to Train: 41.4938s

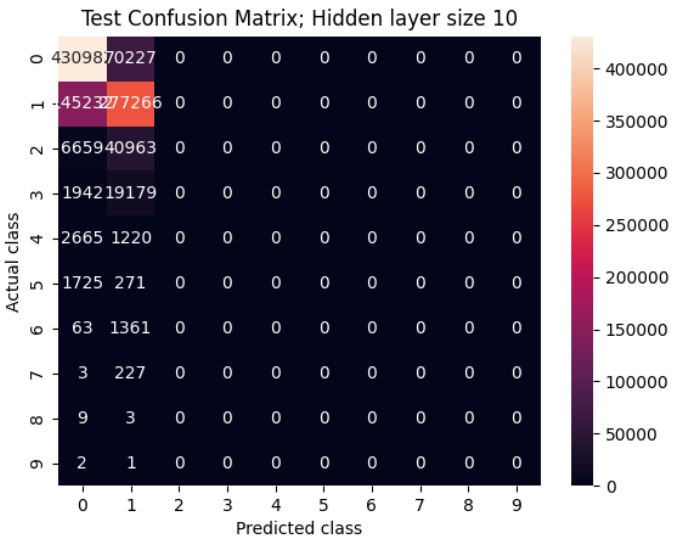
(d)

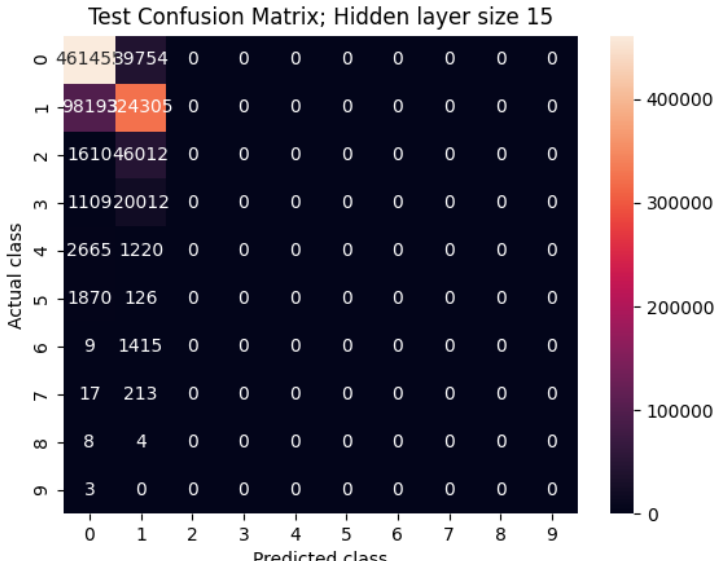
In this part, it was found that for the same initial learning rates as part(b), error threshold should be kept lower to obtain a similar output of accuracy. Also, for same value of threshold, time taken for training is higher as learning rate is decreasing with time. The below data is obtained using the same threshold as above. We can see that accuracies are slightly less than part(d). Adaptive learning does not make conergence faster in our case, rather it increase time to reach convergence.

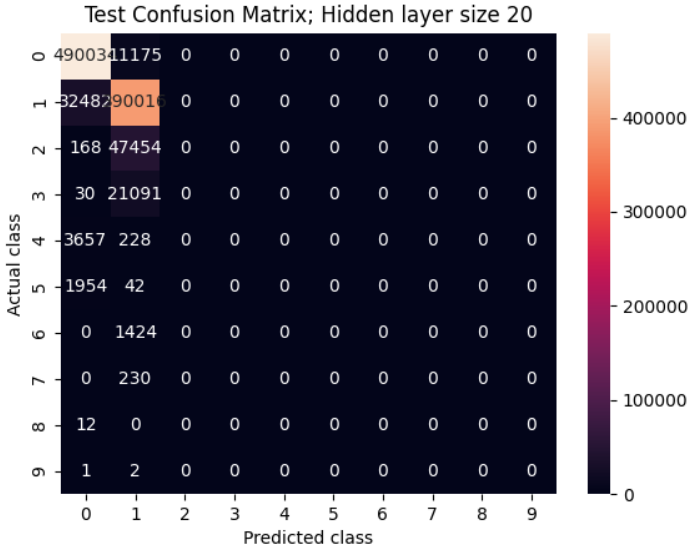


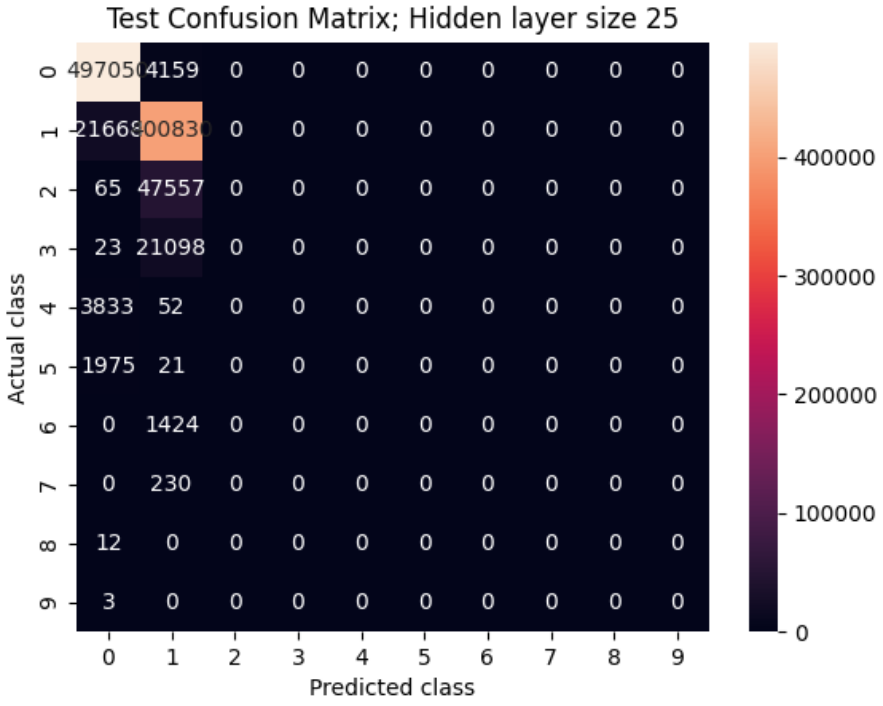












Results obtained are:

Done epoch 365/10000

Done Iteration 1/5

Training accuracy: 0.572890843662535

Test accuracy: 0.547414

Time to Train: 12.5053s

Done epoch 2208/10000
Done Iteration 2/5
Training accuracy: 0.7325069972011196
Test accuracy: 0.708248
Time to Train: 84.6611s

Done epoch 3246/10000
Done Iteration 3/5
Training accuracy: 0.8149940023990404
Test accuracy: 0.78576
Time to Train: 160.9211s

Done epoch 4758/10000
Done Iteration 4/5
Training accuracy: 0.8971611355457817
Test accuracy: 0.88005
Time to Train: 231.0549s

Done epoch 4691/10000
Done Iteration 5/5
Training accuracy: 0.9099960015993602
Test accuracy: 0.89788
Time to Train: 220.9450s

As we can see, time taken to converge is more and accuracies are also reduced slightly for same threshold. Hence, in part(e), we use slightly smaller threshold (1e-6: change in cost between 2 epochs, previously 1e-5).

(e) For part e, we are running data for constant threshold for error change in epoch(= 1e-6) for all cases. Adaptive Learning Rate is considered in all cases Also, due to imbalance in the data, accuracies for different cases are very similar but test accuracies can be seeen to slighly vary. The results obtained are given here:

Training relu model with hidden layers [100, 100]
Done epoch 224/10000
Time to train: 39.9936s
Training Accuracy: 0.92331
Test Accuracy: 0.923686

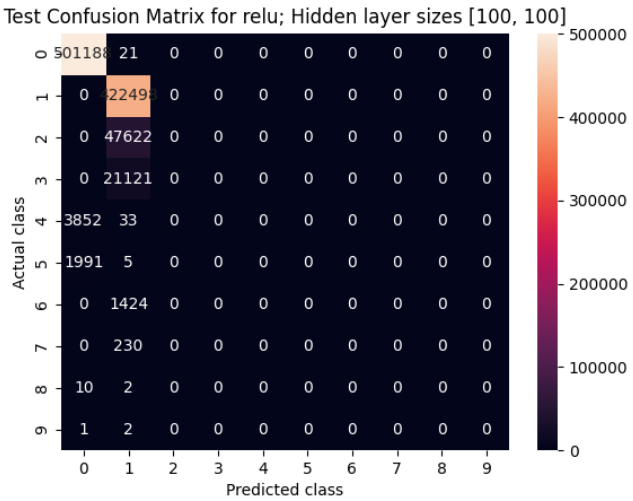
Training sigmoid model with hidden layers [100, 100]
Done epoch 6024/10000
Time to train: 1205.0065s
Training Accuracy: 0.92331
Test Accuracy: 0.923106

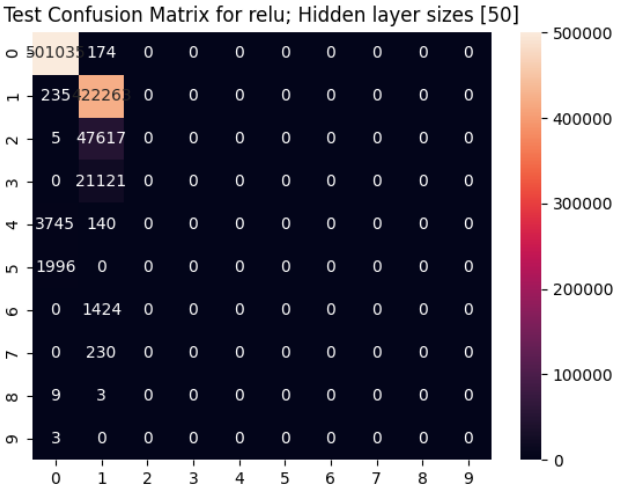
Training relu model with hidden layers [50]
Done epoch 4612/10000
Time to train: 1161.2367s
Training Accuracy: 0.92331
Test Accuracy: 0.923298

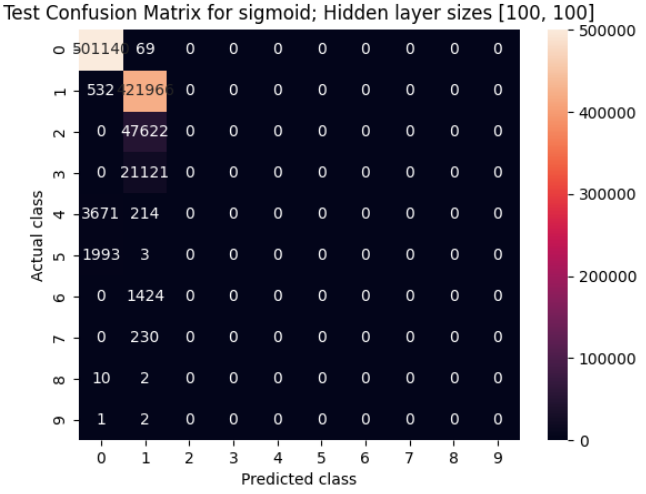
Training sigmoid model with hidden layers [50]
Done epoch 10000/10000
Time to train: 834.5422s
Training Accuracy: 0.92303
Test Accuracy: 0.912834

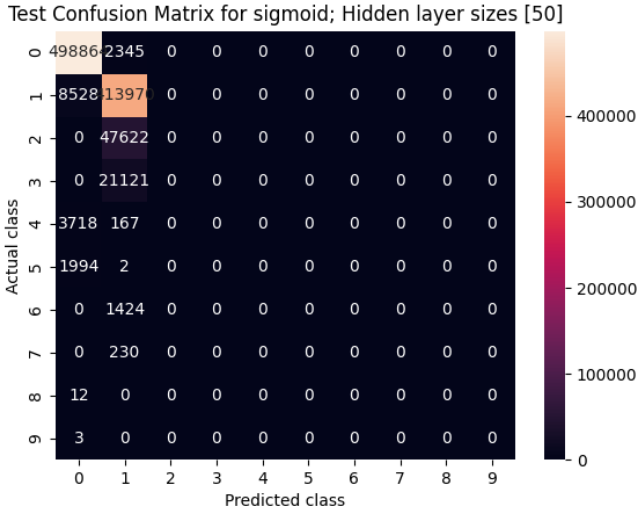
We can see the test accuracies slightly better for relu model, than sigmoid. Also, convergence for relu activation model is much faster than sigmoid in terms of training time and number of epochs. For the case of sigmoid with hidden layers = [50], test accuracy is less, because epochs have exceeded max_epochs before convergence.

Confusion matrices are shown below:









(f) Results for MLP classifier:
Training relu MLP Classifier for hidden layers [100, 100]...
Time to train: 23.1566s
Test accuracy Score: 0.986925

For sklearn MLP classifier with adaptive Learning Rate and minibatch size 100, we get Test accuracy as 0.986925, higher than our best score of 0.923686. The training time is also much lower due to convergence in small number of epochs and other optimizations such as multithreading.