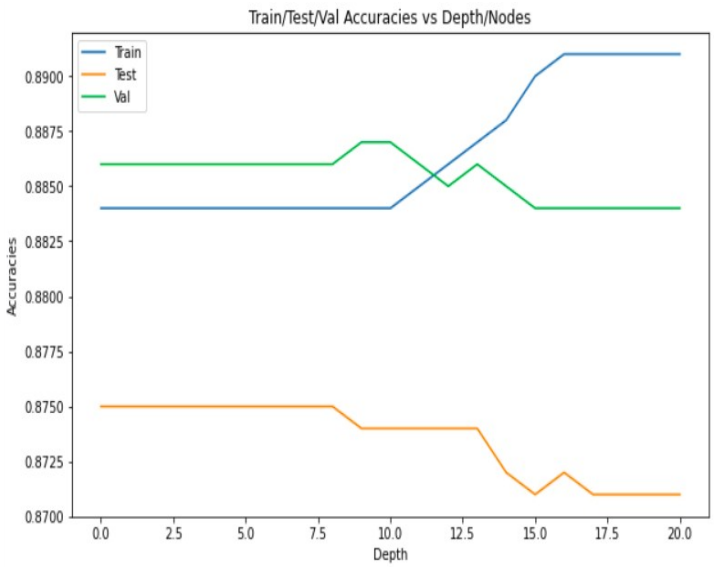
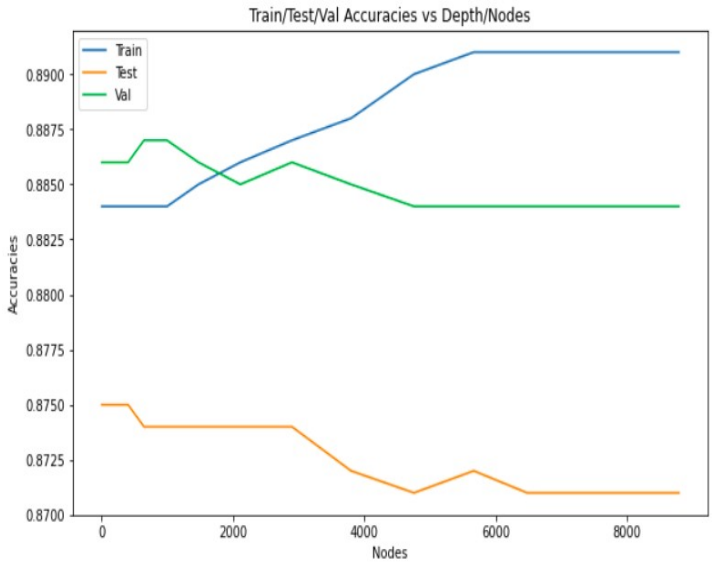
COL774 – Machine Learning (Assignment-3)

Prakhar Aggarwal
2019CS50441

Part 1 (Decision Trees and Random Forests)

(a) (i) Accuracy analysis for depth and nodes (multi-way split)





For depth 20:

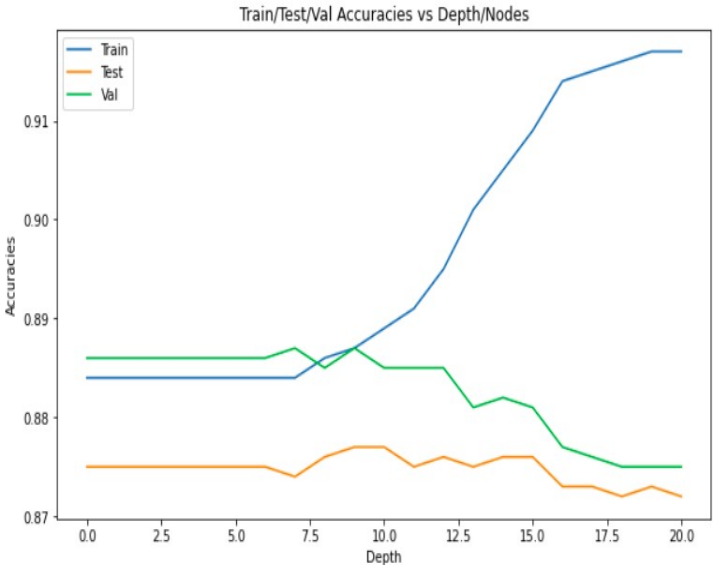
Training accuracy: 89.1%
Testing accuracy: 87.1%
Validation accuracy: 88.4%

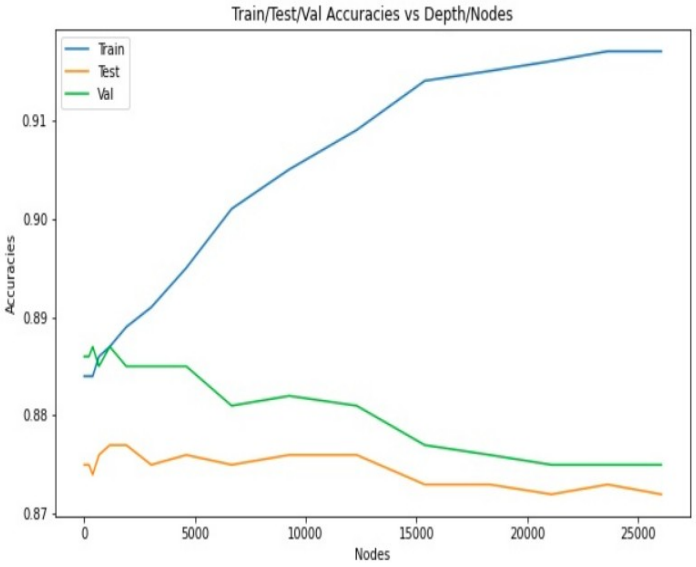
As can be seen from the curve, the model tends to overfit for higher depth since training accuracy is continuously increasing whereas the testing as well as the validation accuracy is decresing.

Max validation accuracy is observed for depth = 9, after this point, test as well as validation accuracy tends to decrease.

Accuracies at depth = 9
Training accuracy: 88.4%
Testing accuracy: 87.4%
Validation accuracy: 88.7%

(a) (ii) Accuracy analysis with depth and nodes (one hot encoding)





For depth 20:

Training accuracy: 91.7%
Testing accuracy: 87.2%
Validation accuracy: 87.5%

As can be seen from the curve, the model tends to overfit for higher depth since training accuracy is continuously increasing whereas the testing as well as the validation accuracy is decresing.

Max validation accuracy for one hot encoding is also observed for depth = 9, after this point, test as well as validation accuracy tends to decrease.

Accuracies at depth = 9
Training accuracy: 88.7%
Testing accuracy: 87.7%
Validation accuracy: 88.7%

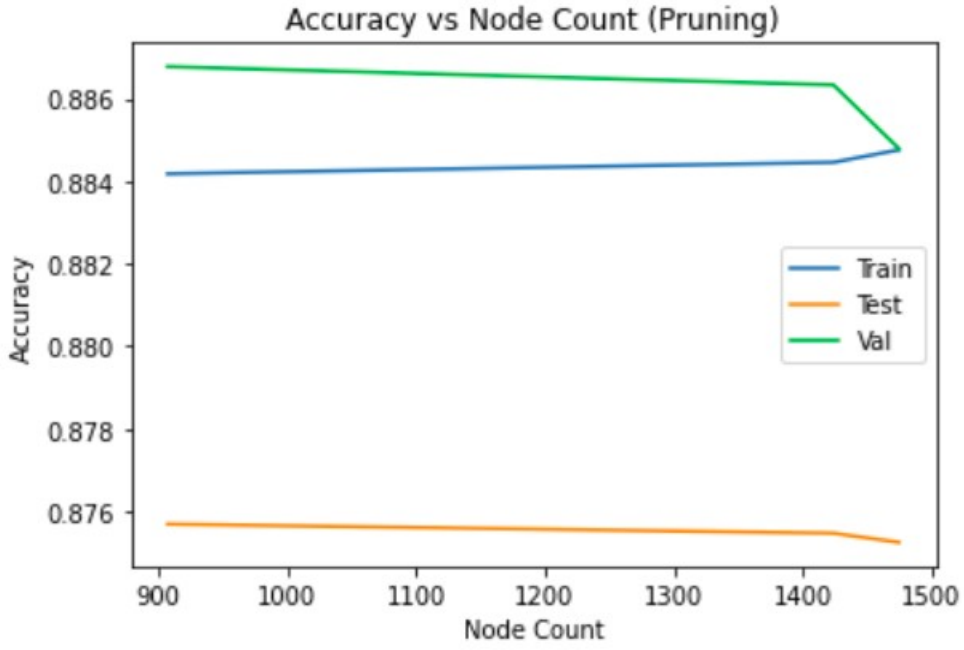
Comparing the results for depth 20 for multi-way split and one hot encoding, we get that one-hot encoding performs slightly better than multi-way split.

Also, the number of nodes generated by one-hot encoding is alot more than that generated by multi-way split. Reason being, once a categorical attribute is used for splitting in multi-way split, it won’t be used again, whereas in the case of one hot encoding, we might split over same categorical feature multiple times.

(b) Post pruning

For this part, depth was set to 9 (because of good results in previous part). Using bredth first search, all nodes as well as the node count was computed. Out of all nodes, best node (node which results in max increase in validation accuracy) was picked and pruned.
It takes 3 iterations to maximize the validation accuracy. Number of nodes reduces as:

Node Count	Training Accuracy	Test Accuracy	Validation Accuracy
1475	88.476%	87.524%	88.478%
1424	88.445%	87.547%	88.633%
907	88.417%	87.569%	88.677%



Observations:

- We can see from the above graph, there is a increase of 0.2% in validation accuracy as node count updates from 1475 to 907.
- Train accuracy decreases with decrease in node count; i.e. the overfitted model is getting corrected.
- There is an increase of 0.04% in test accuracy.

(c) Random Forests

Best parameters obtained are:

n_estimators: 350
max_features: 0.3
min_samples_split: 10
max_depth: 20

Accuracies obtained for these parameters:

Training Accuracy: 97.102%
Test Accuracy: 90.134%
Validation Accuracy: 90.601%

Out of bag score (oob_score): 0.9081232

Accuracy comparison:

Accuracies	Part-b (Post pruning)	Part-c (RandomForestClassifier)
Training Accuracy	88.417%	97.102%
Test Accuracy	87.569%	90.134%
Validation Accuracy	88.677%	90.601%

On comparing this with the values obtained in part b, we can clearly observe that the RandomForestClassifier outperforms Decision tree with pruning with good margin.

Observations:

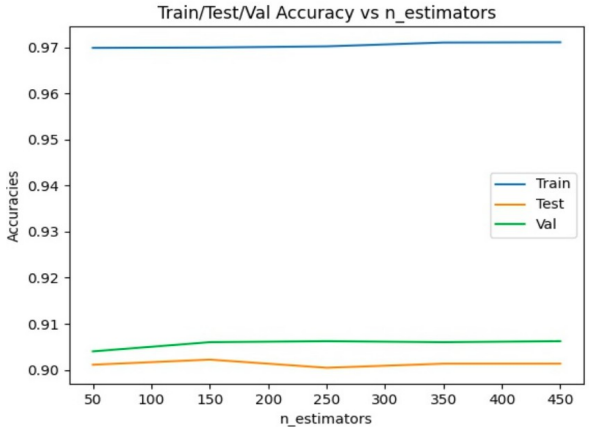
- Categorical features are split in the way; if job can have 3 different values, then 3 new features are added and the feature job itself is removed.
- On comparing these results with previous parts, we can say that Random forest classifier works better than Decision trees. Reason for improvement is that having multiple trees makes the model less sensitive to high variance.
- Out of bag score is computed on the data that was not necessarily used in the analysis of the model. It is a good measure of performance on test and validation data.

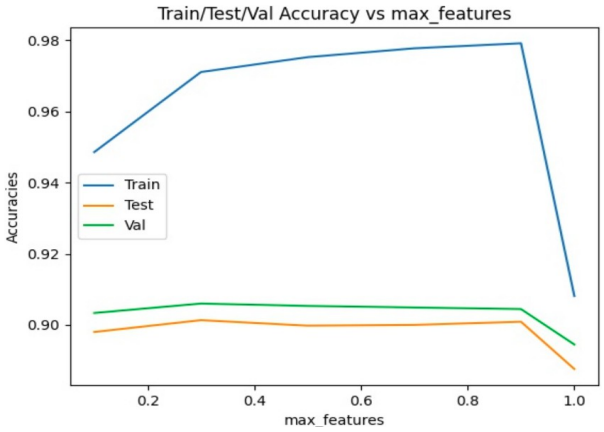
(d) Parameter Sensitivity Analysis

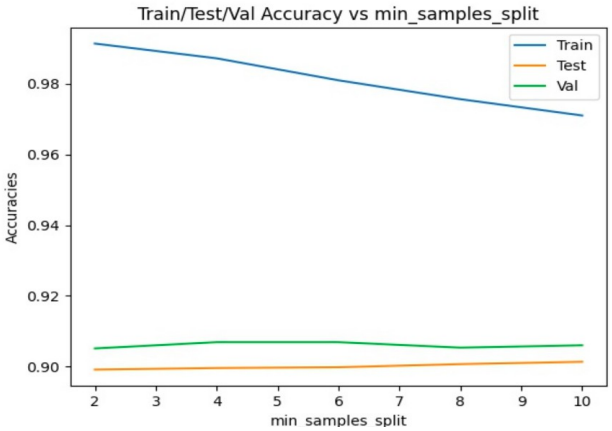
Best parameters obtained are:

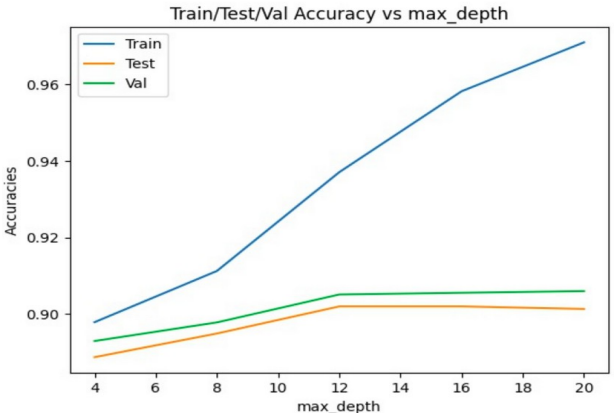
n_estimators: 350
max_features: 0.3
min_samples_split: 10
max_depth: 20

Train/Test/Val accuracy analysis is done by fixing all but one feature.









Observations:

- In case of n_estimators, there is not much change in accuracy with change in n_estimators from 50 to 450.
- For Max features, accuracies slighlty change as we go from 0.1 to 1. We can see from the graph, max validation accuracy is for max_feature = 0.3
- For min_samples_split, max test accuracy is obtained at value 10. Though there is not much change in the accuracy values.
- We can see from the graph, there is reasonable change in train/test/val accuracies with increase in depth.

Parameter Sensitivity:

- n_estimators and min_samples_split are least sensitive to parameter change. Variation in test accuracy is only 0.02% for n_estimators and 0.2% for min_samples_split.
- There is a variation of 1.2% for max_features.
- Reasonable change in accuracy is observed for max_depth. A variation of 1.4% in test/val accuracy is observed.

Part 2 (Neural Networks)

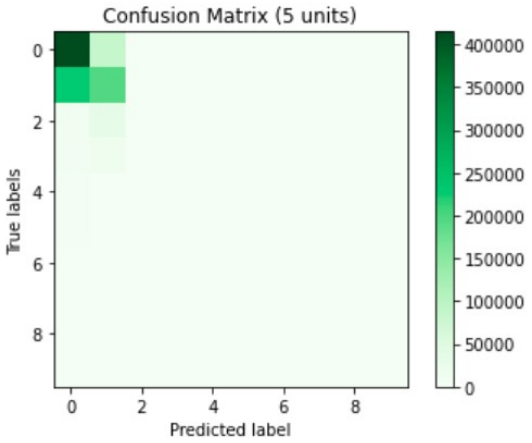
(a) In this part, categorical features are transformed to their one hot encoding representation. The new dataset obtained contains 85 features.

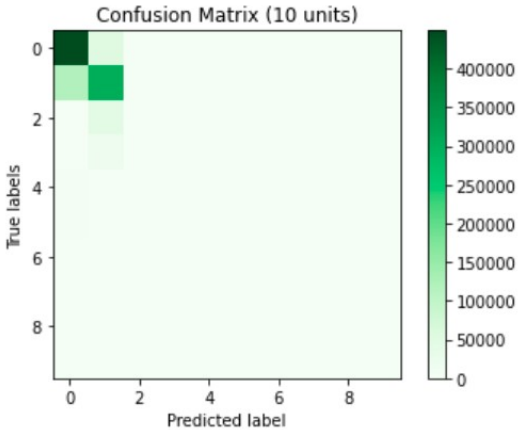
(b) In this part, a neural network model was constructed from scratch. Forward propagation, backward propagation were coded from first principles. Mini-batch SGD was used to faster the learning process.
The model is general enough to take the following values as parameters:
Mini-batch size, Number of features, Hidden layer architecture, Number of target classes, learning rate, epsilon, Number of Epochs.

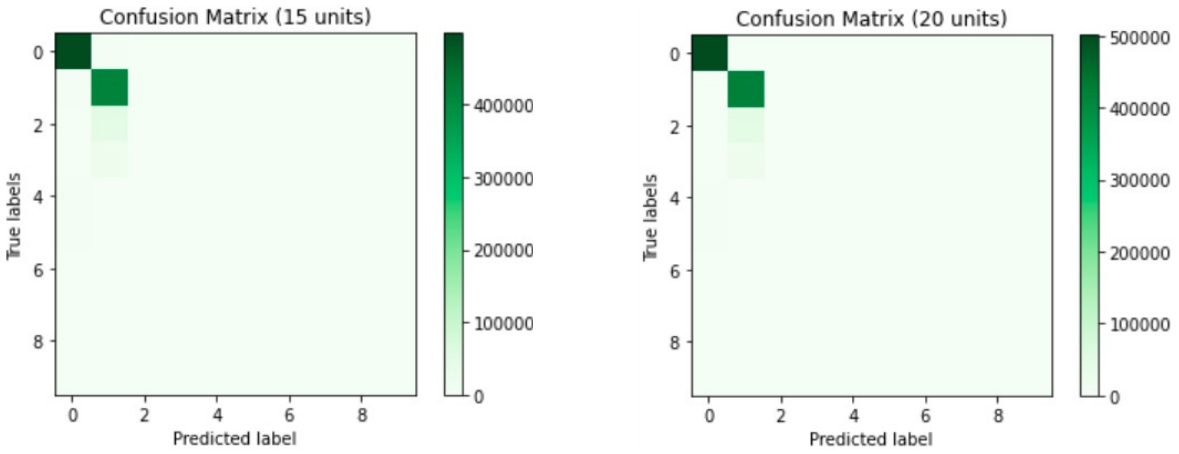
(c) Convergence Criteria:

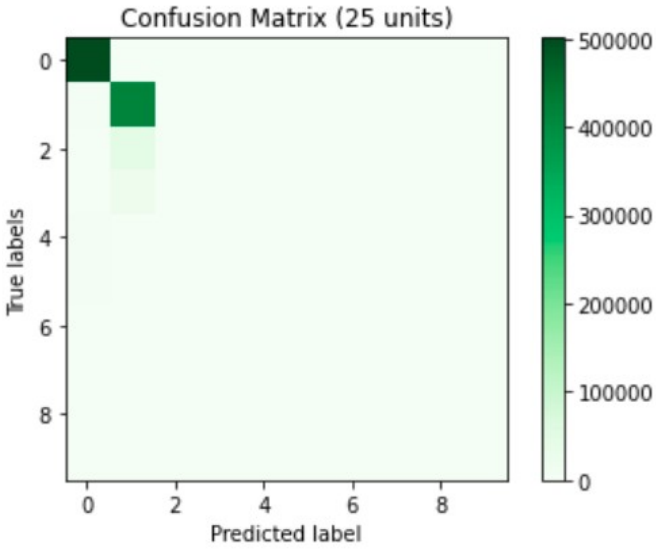
- There is a hard bound on the number of the epochs (1000)
- There is a variable epsilon which looks at the difference of errors of consecutive epochs and stops if this difference is less than epsilon (0.0001)
- We have a variable MIN_LOSS which helps in avoiding convergence in a case where error difference of consecutive epochs is very small but error of a particular epoch is high.
- Learning rate: 0.1

Num of Units	5	10	15	20	25
Training Time	33.763s	37.343s	39.755s	42.797s	44.292s
Train Accuracy	62.687%	76.997%	91.975%	92.275%	92.225%
Test Accuracy	61.022%	75.169%	91.503%	92.225%	91.984%

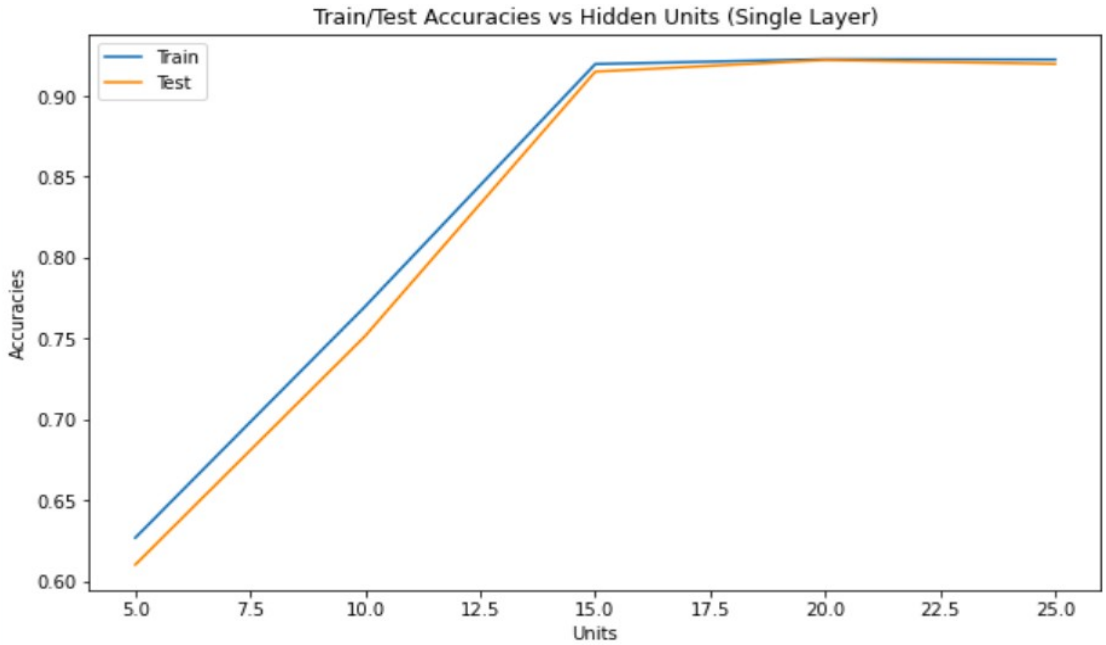








We can see from the below graph that in the case of 1 Hidden layer, both training as well as test accuracy increases on increasing the number of neurons. This can be explained as more neurons allow complex features to learn. There is not much change in accuracy as one goes from 20 to 25 neuron units.

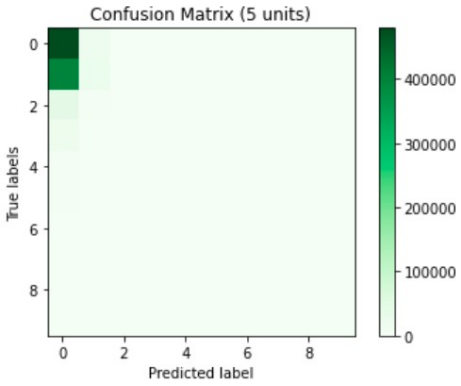


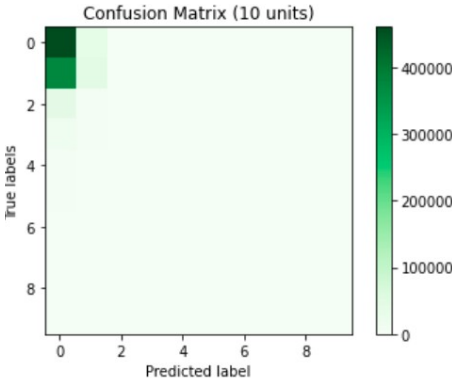
(d) Adaptive Learning

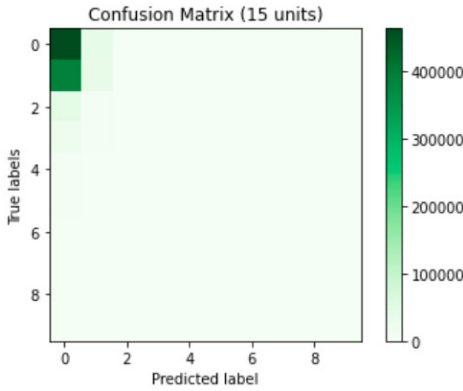
On running the model for 1000 epochs, following results are obtained:

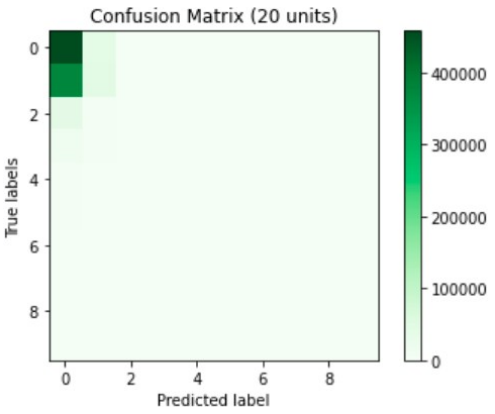
Num of Units	5	10	15	20	25
Training Time	36.952s	49.401s	63.192s	45.562s	50.782s
Train Accuracy	50.652%	51.347%	50.552%	51.303%	50.212%
Test Accuracy	50.39%	50.624%	49.807%	50.416%	49.475%

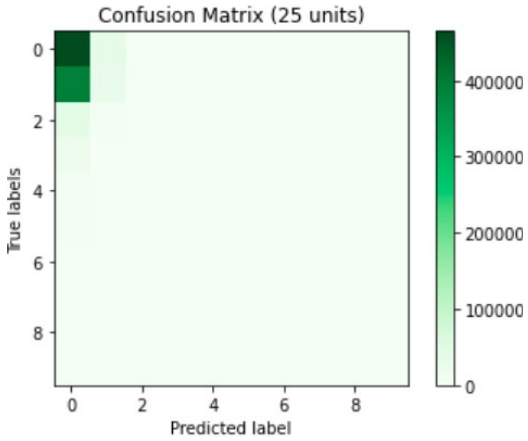
As we can see from the above table, accuracies obtained are way smaller than that obtained in the previous part. Also, there is no improvement in the accuracy with increase in neurons.

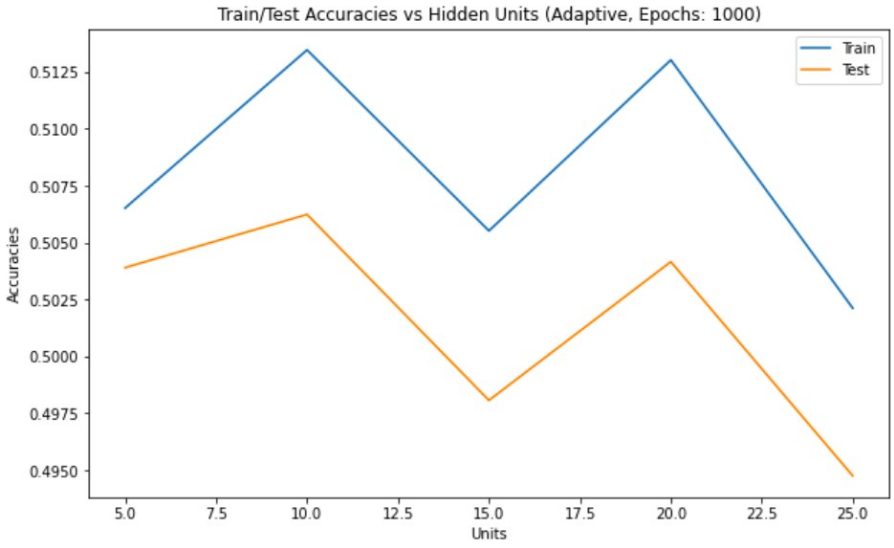






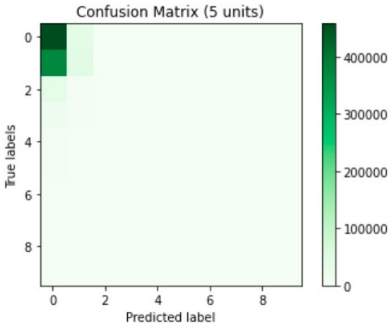


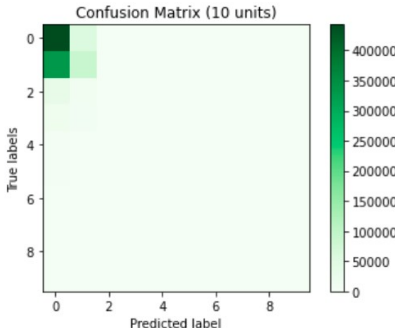


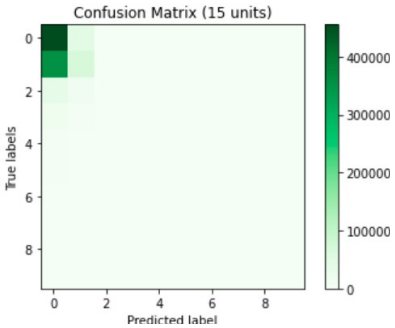


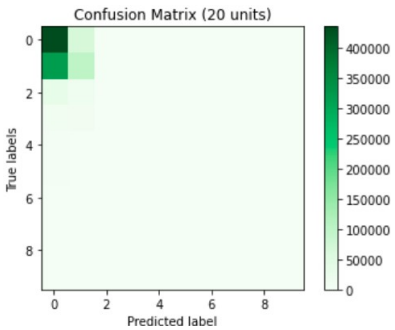
On changing the stopping criteria i.e. on increasing the number of epochs from 1000 to 2000, following results are obtained:

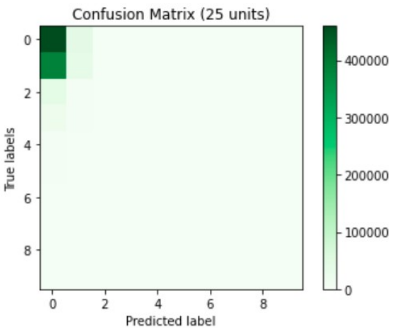
Num of Units	5	10	15	20	25
Training Time	76.398s	78.272s	88.15s	89.879s	95.56s
Train Accuracy	51.483%	54.294%	53.555%	54.95%	50.476%
Test Accuracy	50.812%	53.096%	52.333%	53.643%	49.665%

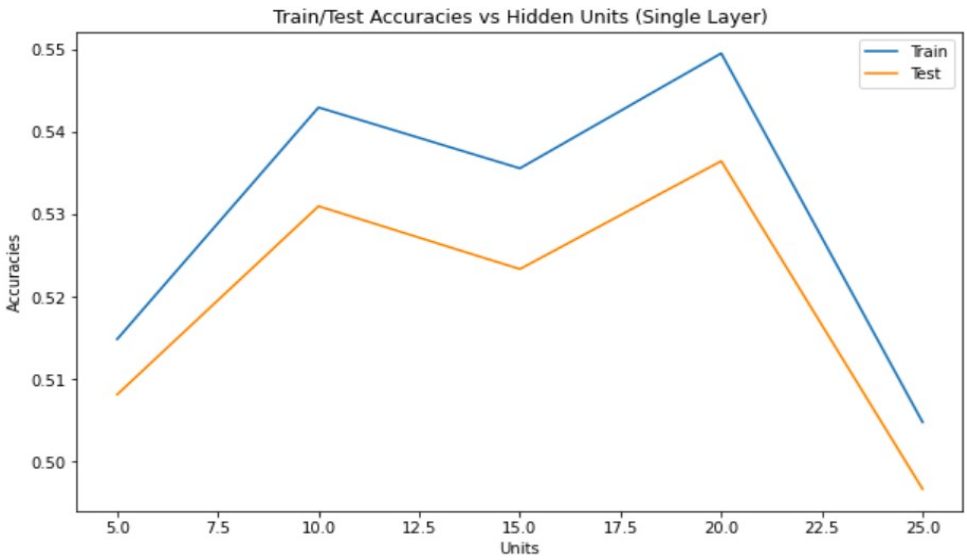












So, adaptive learning rate for the given data doesn't helps in decreasing the learning time and doesn't improve or even manage to keep the same accuracy as obtained with the fixed learning rate.

(e) Results obtained for different settings: a network with 2 hidden layers with 100 units each and sigmoid activation function:

Model	Training Time	Train Accuracy	Test Accuracy
2 Hidden layers (100,100), Sigmoid activation	1030.84s	99.78%	98.492%
2 Hidden layers (100,100), Sigmoid activation, Adaptive learning rate	728.832s	50.096%	49.559%
2 Hidden layers (100,100), Relu activation	917.689s	51.12%	50.289%
2 Hidden layers (100,100), Relu activation, Adaptive Learning Rate	583.187s	51.06%	50.436%

Confusion Matrix:

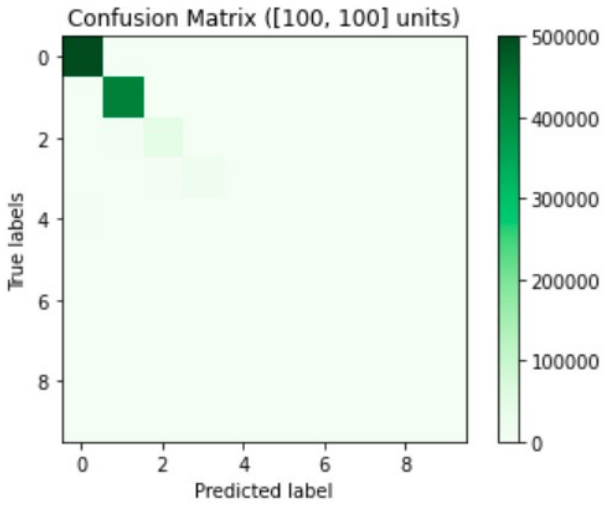


Figure 1: (100,100), Sigmoid

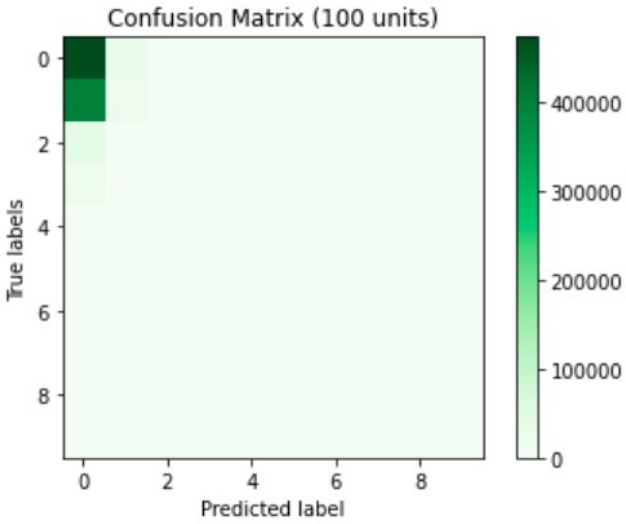


Figure 2: (100,100), Sigmoid, Adaptive

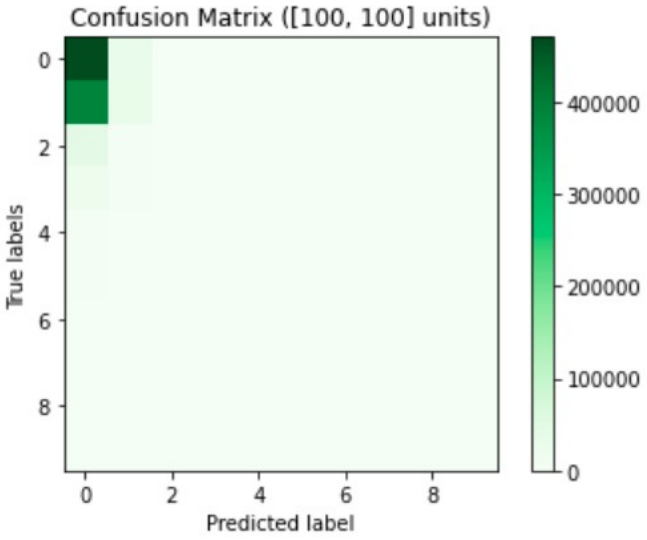


Figure 3: (100,100), Relu

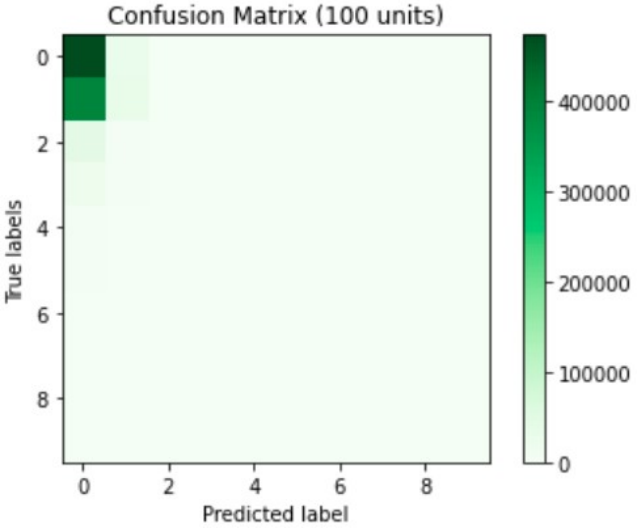


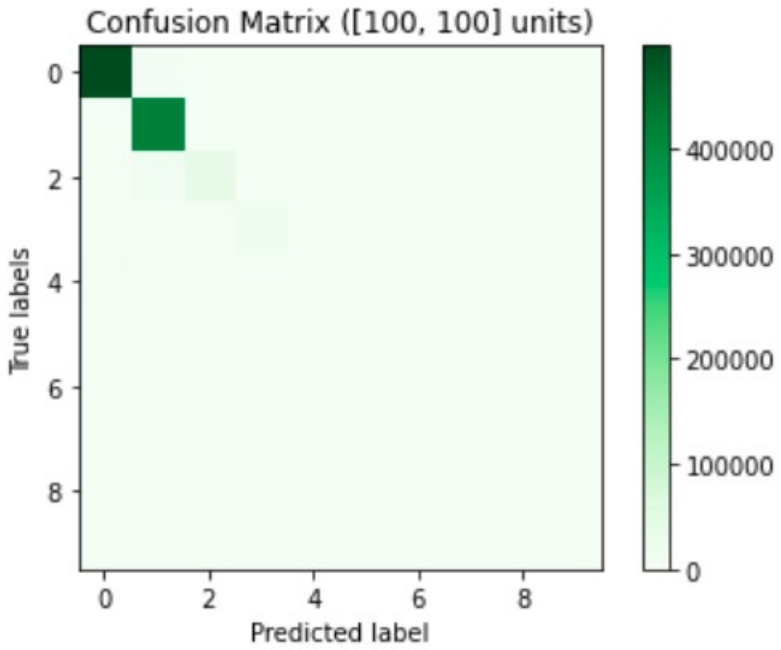
Figure 4: (100,100), Relu, Adaptive

Observations:

- In relu activation function, there was a problem of huge gradient updates due to which, some neurons were not able to learn the parameter. (nan values)
- Best accuracies are obtained in the case of ‘sigmoid’ activation with (85,100,100,10) neural architecture.
- There is large improvement in test accuracy as we go from 1 layer (20 units) with accuracy 92.225% to 2 layers (100, 100 units) with accuracy 98.492%.
- On looking at the Confusion matrices, we can infer that the model (sigmoid, (100,100)) learns some examples of class 3 as well as class 4, whereas model with single layer doesn’t.
- This is the primary reason for drastic improvement in train/test accuracy.

(f) MLP Classifier
Setting: Activation = Relu, Max_iterations = 1000, solver = sgd, learning_rate = “adaptive”, architecture = (100, 100)

Training Time: 1735.218s
Train Accuracy: 99.43223%
Test Accuracy: 97.3983%



We can observe that the accuracy obtained with the sklearn inbuilt MLP classifier with cross entropy loss for relu activation function is way better than that obtained with Mean squared loss. Though the time taken to train the model is alot more, but the results obtained are pretty good.
On comparing this with part e, our model performs slighlty better than the one learned by MLP classifier.