**Assignment – 3**

**COL774 Machine Learning**

Anirudha Kulkarni

**Q1 Decision Tree**

a)

**Datapoints:**Training accuracy

Testing accuracy

Validation accuracy

[0.8835987613359876, 0.8964001327140013, 0.8964001327140013, 0.8968701614687016, 0.8985843839858438, 0.9023999115239991, 0.9035611590356116, 0.905966600309666, 0.9072660915726609, 0.9088697190886972, 0.9194591904445919, 0.934555408095554, 0.9568679495686795, 0.9791528422915284, 0.9954656049546561]

[0.8752488387524884, 0.8880778588807786, 0.8880778588807786, 0.8885202388852024, 0.893607608936076, 0.8962618889626189, 0.8953771289537713, 0.8958195089581951, 0.896040698960407, 0.8953771289537713, 0.8940499889404999, 0.8865295288652952, 0.880557398805574, 0.8750276487502765, 0.8723733687237337]

[0.8861123396727112, 0.891640866873065, 0.891640866873065, 0.8929677134011499, 0.8898717381689518, 0.8929677134011499, 0.8942945599292349, 0.897390535161433, 0.8978328173374613, 0.8971693940734189, 0.9038036267138434, 0.8962848297213623, 0.8889871738168952, 0.8803626713843432, 0.8721804511278195]

[3, 7, 15, 27, 49, 81, 127, 191, 269, 367, 1375, 3607, 6863, 10047, 12335]

Multi way

[0.8835987613359876, 0.8964001327140013, 0.8967595664675957, 0.8985567352355673, 0.9011004202610042, 0.9083443928334439, 0.9170537491705375, 0.9261778367617783, 0.9370437956204379, 0.9484074319840743, 0.9910141561601415, 1.0, 1.0, 1.0, 1.0]

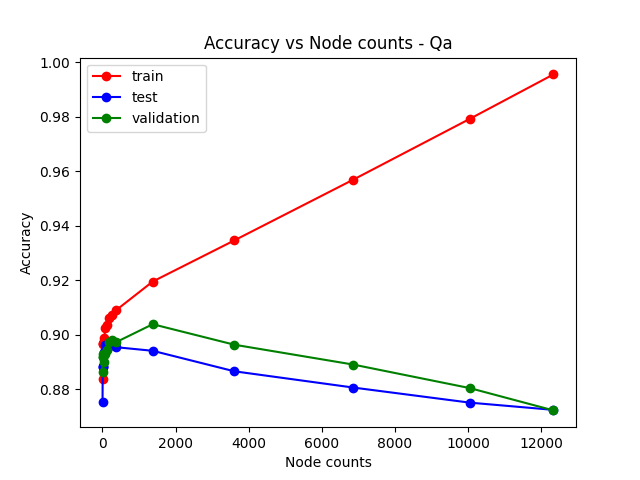
[0.8752488387524884, 0.8880778588807786, 0.8882990488829905, 0.8865295288652952, 0.8867507188675072, 0.8927228489272285, 0.8889626188896262, 0.8818845388188454, 0.876133598761336, 0.8763547887635479, 0.8639681486396815, 0.8613138686131386, 0.8613138686131386, 0.8613138686131386, 0.8613138686131386]

[0.8861123396727112, 0.891640866873065, 0.8909774436090225, 0.8911985846970367, 0.8925254312251216, 0.8896505970809376, 0.8898717381689518, 0.888766032728881, 0.8874391862007961, 0.8854489164086687, 0.8721804511278195, 0.8695267580716497, 0.8695267580716497, 0.8695267580716497, 0.8695267580716497]

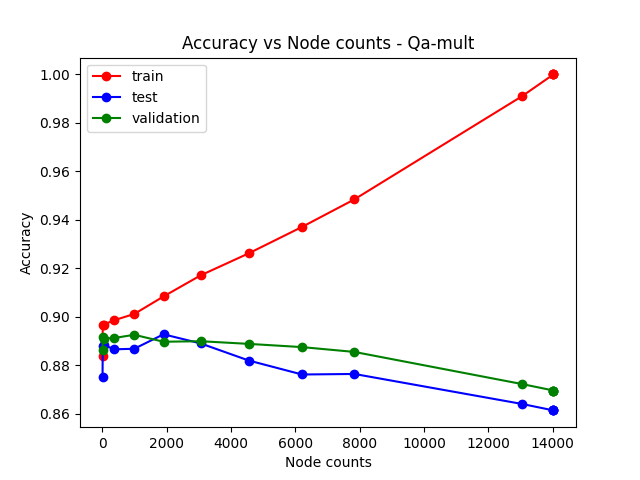
[3, 9, 49, 367, 988, 1899, 3052, 4553, 6204, 7831, 13045, 14020, 14020, 14020, 14020]

Tree has higher number of nodes in case of oneHot encoding due to larger number of classes

**One Hot encoded**:

****

**Multi way division:**

****

**Comment:**

**1. One hot encoding vs Multi way division: One hot encoding performs slightly better than multi way division. Multi way division produces less number of nodes compared with one hot encoding even at same depth.**

**2. As number of nodes increase the train accuracy increases but the validation and test accuracies increase upto a point and then start to decline. This is a case of overfitting. The model fits data correctly till 1000 nodes but after that it fits the noise as well and hence lead to overfit and testing accuracy declines.**

b)

One hot

[0.9933919486839194, 0.9795122760451228, 0.9662132271621323, 0.9472185357221854, 0.9310163680601636, 0.9006856890068569]

[0.8723733687237337, 0.8779031187790312, 0.880336208803362, 0.8867507188675072, 0.8891838088918381, 0.8845388188453882]

[0.8737284387439186, 0.8788146837682441, 0.8841220698805838, 0.8863334807607254, 0.891640866873065, 0.8923042901371074]

Multi

prune\_train\_list=[0.9933919486839194, 0.9795122760451228, 0.9662132271621323, 0.9472185357221854, 0.9310163680601636, 0.9006856890068569]

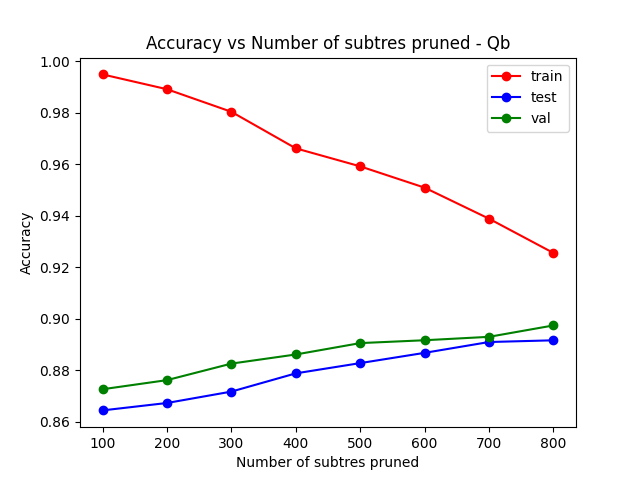
prune\_test\_list=[0.8723733687237337, 0.8779031187790312, 0.880336208803362, 0.8867507188675072, 0.8891838088918381, 0.8845388188453882]

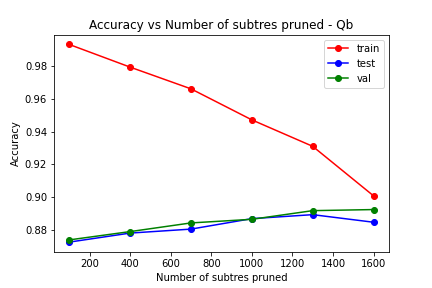
prune\_val\_list=[0.8737284387439186, 0.8788146837682441, 0.8841220698805838, 0.8863334807607254, 0.891640866873065, 0.8923042901371074]

Test accuracy after maximum possible pruning

[0.8752488387524884, 0.8880778588807786, 0.8880778588807786, 0.8885202388852024, 0.8885202388852024, 0.8889626188896262, 0.8885202388852024, 0.8958195089581951, 0.8953771289537713, 0.8942711789427118, 0.8955983189559832, 0.8958195089581951, 0.8880778588807786, 0.8752488387524884, 0.8752488387524884]

Comparing Accuracy vs Number of trees pruned for **Multi way division**



Comparing Accuracy vs Number of trees pruned for **One hot encoded**

**Test accuracy after maximum possible pruning**

Chart, line chart

Description automatically generated

**Comments:**

**1. increase in pruning increases the validation accuracy as expected as we pruned in way to increase validation accuracy.**

**2. Increase in pruning decreases training accuracy significantly but this is counteracted by increase in test accuracy which increases with validation accuracy. This is expected as the noise is learnt less as we start to prune and hence lead to fitting model to the underlying patterns only**

**3. Test accuracy decreases with more number of total nodes in the tree but is more when done with pruning than without pruning.**

**C)**

Optimal parameters: n\_estimators: 350, max\_features:0.3, min\_samples\_split: 10

oob score: 0.8907597876575979

Train accuracy: 0.9064642778146428

Test accuracy: 0.8843176288431763

Val accuracy: 0.8900928792569659

Previous model after pruning:

Train accuracy: 0.9006856890068569

Test accuracy: 0.8845388188453882

Val accuracy: 0.8923042901371074

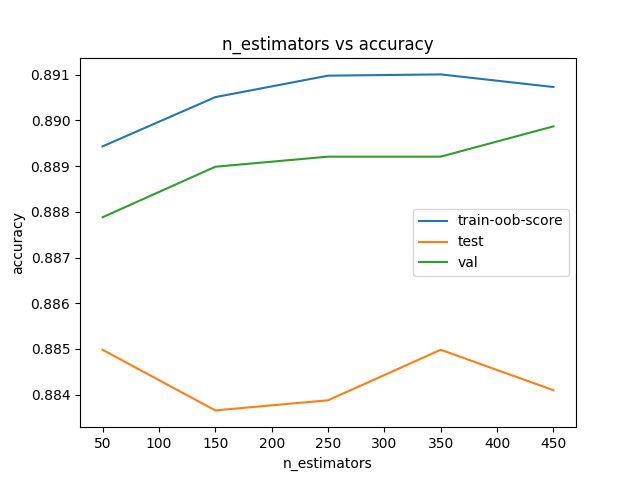
**Comments:**

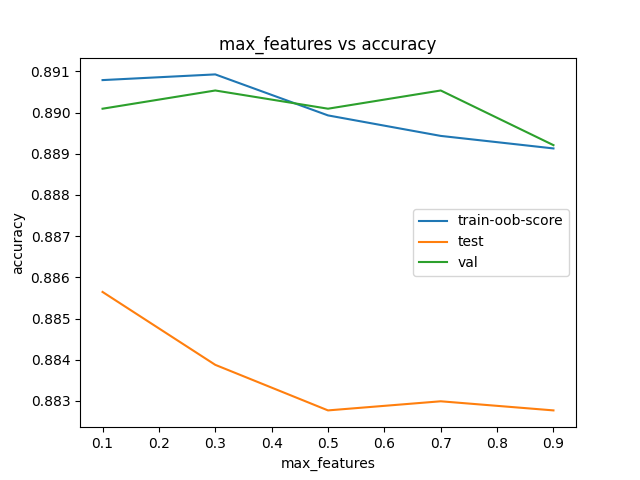
**1. Random Forest classifier performs better than using simple Decision Tree. Also, using large number of trees makes the model more robust and less sensitive to high variances in data.**

**2. Both performs almost similarly with Random Forest performing better in training and Pruning performing better in testing and validation. Still the difference is less than 0.02 percent**

**3. out of bag score is a good indicator of performance on test and validation data. Though the train accuracy is higher than oob socre, test accuracy and validation accuracy match with oob score**

**d)**

****

****

**Chart, line chart

Description automatically generated**

**Observations:**

We see the peak at optimal parameters in oob score in each graph but test and validation accuracies vary over a limit.

In case of n\_estimators we see the test and validation follows similar trend as oob score and are maximum at 350 estimators.

In case of max\_features test accuracy decreases as we increase the max\_features. Though the optimal is at 0.3 and validation also occurs at 0.3, test accuracy is maximum at 0.1. This can be attributed to noise as the difference in accuracies is less than 0.1%

In case of min\_samples\_split we see validation and train accuracy is max at 10 and test accuracy decreases by 0.1% after 9. Still the overall trend is upwards and matches with oob score trend

**Sensitivity:**

n\_estimators is the least sensitive to absolute parameter change. It varies over by 0.1 percent over entire range of n\_estimator. max\_feature is the moderately sensitive to absolute parameter change. It varies by 0.3 percent over entire range. min\_samples\_split is most sensitive with 0.4% variation across range.

**Neural Network:**

a) attached in the code

b) attached in the code

c)

|  |  |  |  |
| --- | --- | --- | --- |
| Units | Train time | Train accuracy | Test accuracy |
| 5 | 63.26 | 0.6457 | 0.6278 |
| 10 | 142.82 | 0.87 | 0.85 |
| 15 | 161.37 | 0.92 | 0.92 |
| 20 | 207.40 | 0.9233 | 0.9218 |
| 25 | 247.45 | 0.9233 | 0.9217 |

train accuracy and test accuracy vs units

Chart

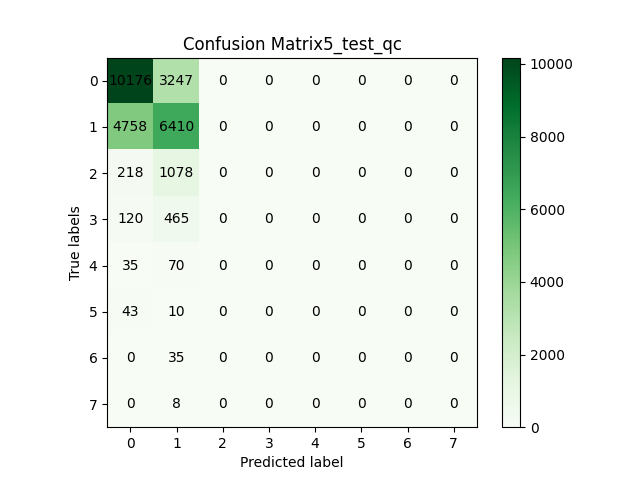
Description automatically generated

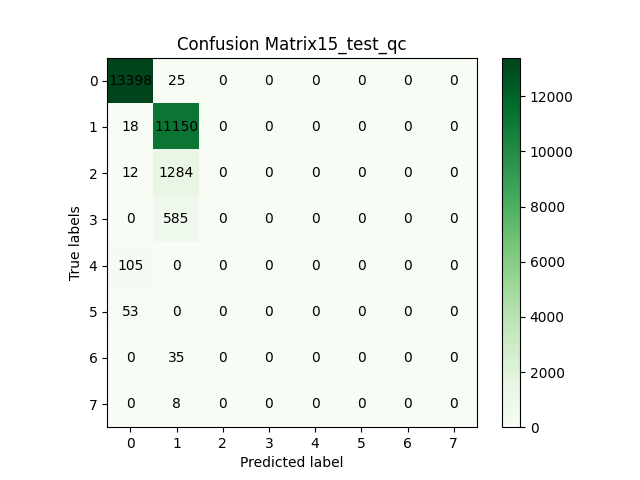
Time taken to train

Chart, line chart

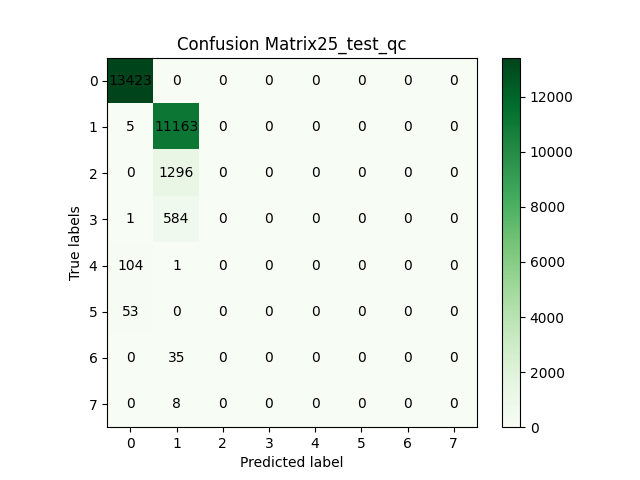
Description automatically generated

Stopping criteria: if change in train acc after an epoch is less than epsilon or if max iterations are exceeded where **Epsilon =** **10-5, Max iterations = 1400**

Calendar

Description automatically generatedCalendar

Description automatically generated

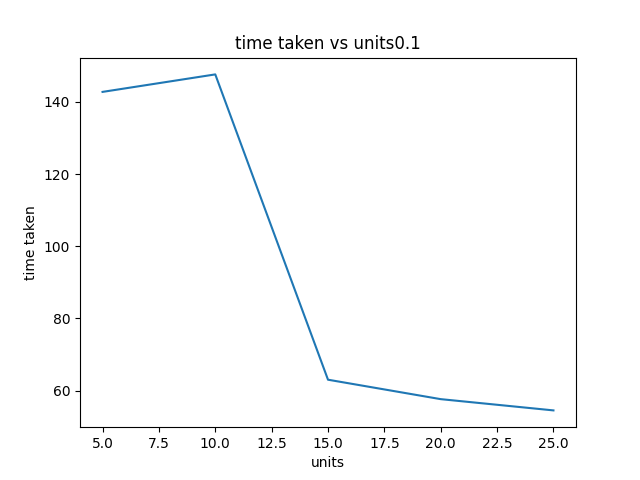
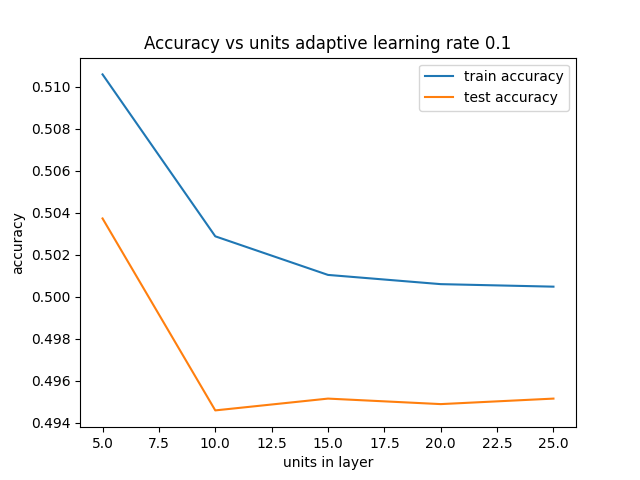


Observations:

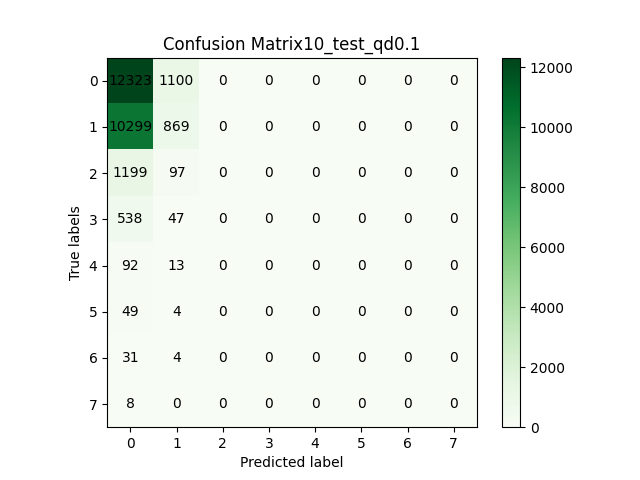
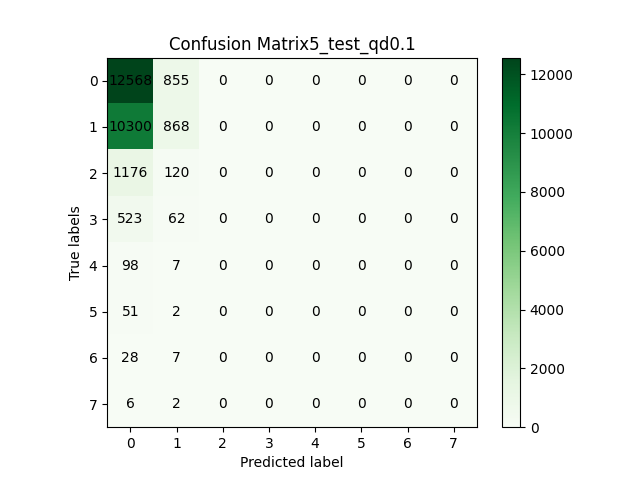
1. Accuracy increases as the number of units in the layer are increased. This is intuitive as more units allows more complex relation to learn. But there is not significant progress after 15 units in the layer. Handful of examples are predicted correctly but time required to train is significant.

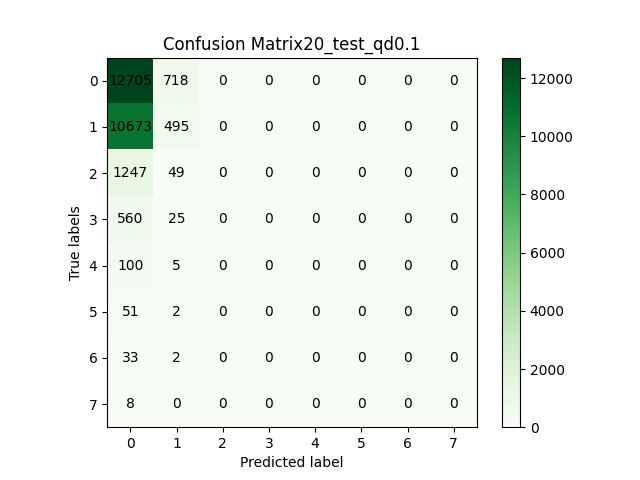
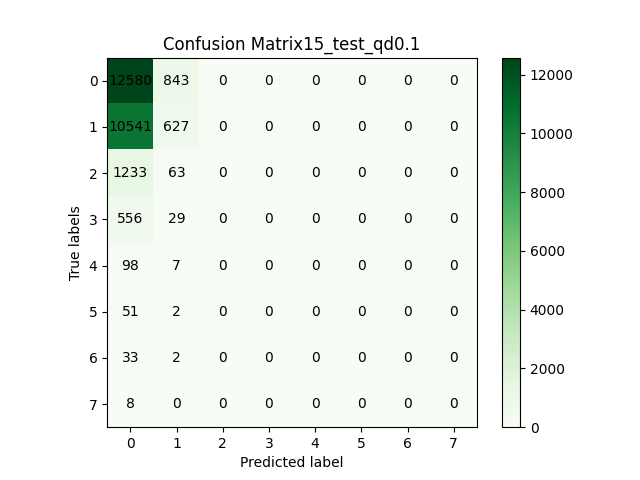
d)

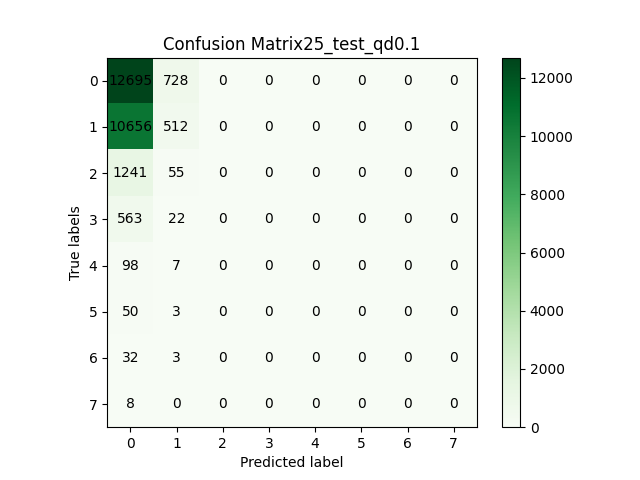
When alpha0 = 0.1



Stopping criteria: if change in train acc after an epoch is less than epsilon or if max iterations are exceeded where **Epsilon =** **10-6, Max iterations = 1400 as the progress is very slow**



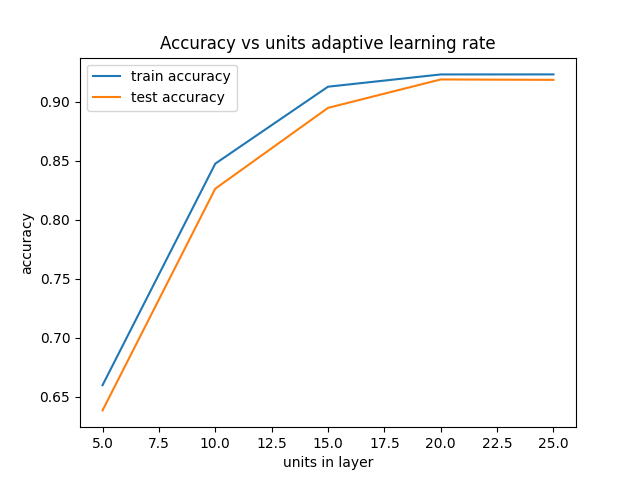


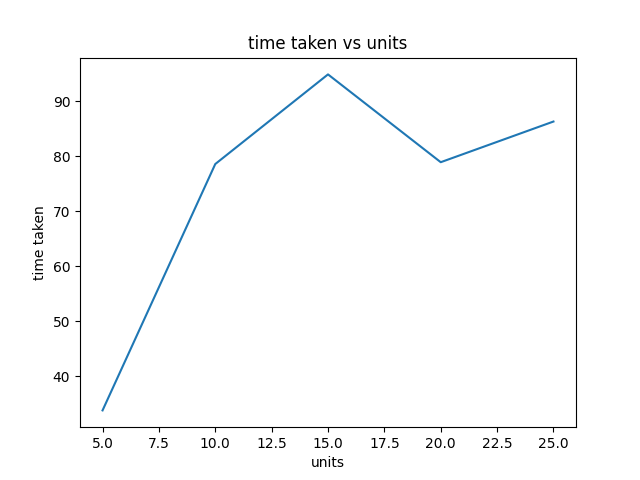


When alpha0 = 10

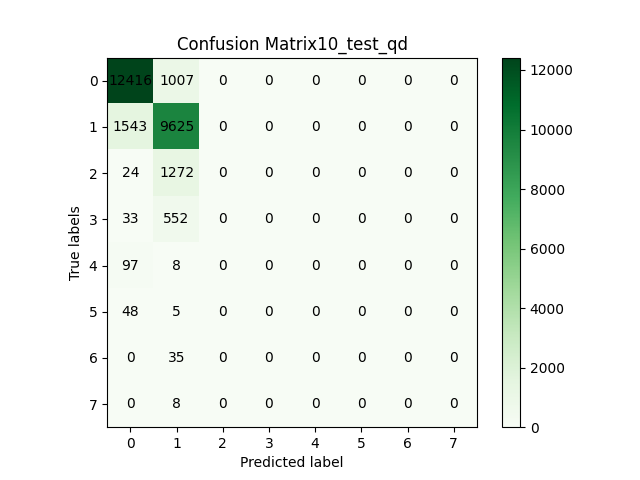
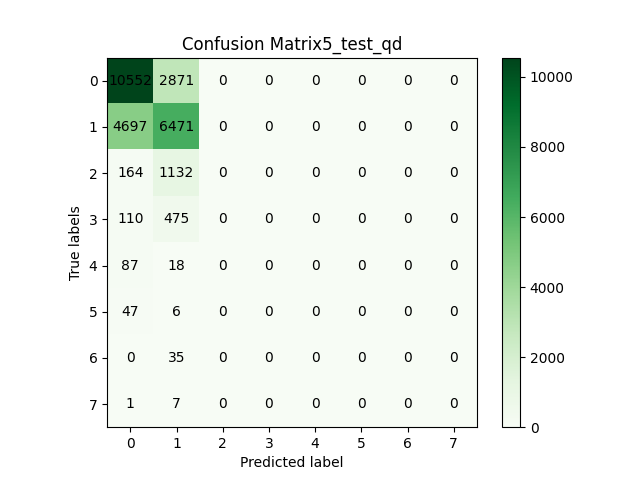
**Train Accuracies**: [0.6724110355857656, 0.8502199120351859, 0.9145541783286686, 0.9233106757297082, 0.9233106757297082]

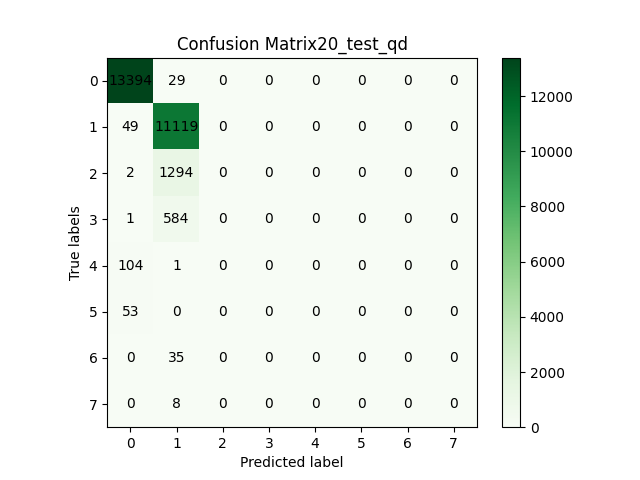
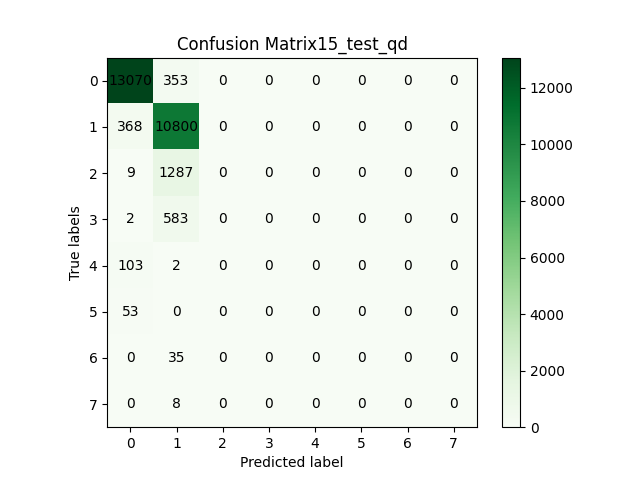
**Test Accuracies**: [0.649683200239943, 0.8287781651857684, 0.8956997713043152, 0.918269411014884, 0.9177445356727777]

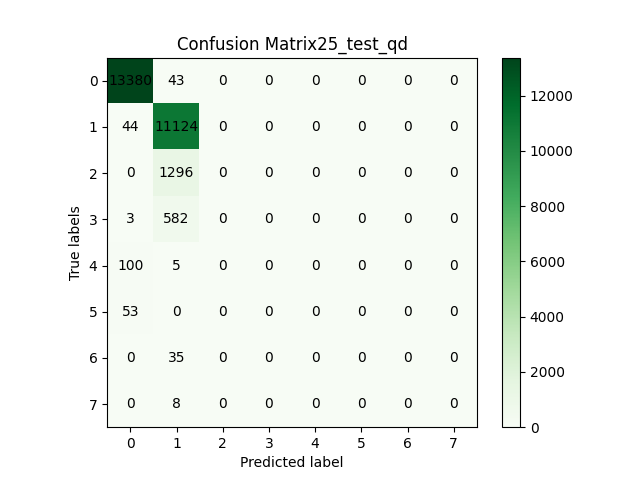




if change in train acc after an epoch is less than epsilon or if max iterations are exceeded where **Epsilon =** **10-5, Max iterations = 1400**







e)

|  |  |  |
| --- | --- | --- |
|  | Sigmoid | Relu |
| Train time | 442.49 | 51.66 |
| Training accuracy | 0.9233106757297082 | 0.9616953218712515 |
| Test accuracy | 0.9211562253964684 | 0.957035204139017 |

Relu:

Calendar

Description automatically generated

Sigmoid

Calendar

Description automatically generated

Comments:

1. Test accuracy and confusion matrix comparison:

Relu performs significantly better in terms of overall accuracy. Also, relu correctly classifies 3rd prediction where sigmoid didn’t predict any correctly. Difference in training time is also significant. Relu converges much faster

2. single hidden layer with sigmoid vs 2 layers

Single hidden layer could not achieve more than 92% accuracy in all the cases. Whereas 100,100 combinations provided more than 95% test accuracy. Also the accuracy was still high in single layer as only 2 predictions where majority. Single layer could not learn any prediction corresponding to class 3. Hence if the data was without heavy bias then there would have been larger difference in accuracies.

f)

Training time: 123.23423743247986

Test accuracy: 0.9603719116709781

Library implementation performed better than custom implementation. The best on test accuracy compares by 0.96 vs 0.957. Training time is also lower due to optimization such as multithreading and efficient use of data structures. The difference in accuracy might be due to difference in error function and stopping criteria and learning rate