COL774
Assignment 3

Decision Trees and Neural Networks

Aryan Dua
2020CS50475

October 30, 2022

1 Question 1

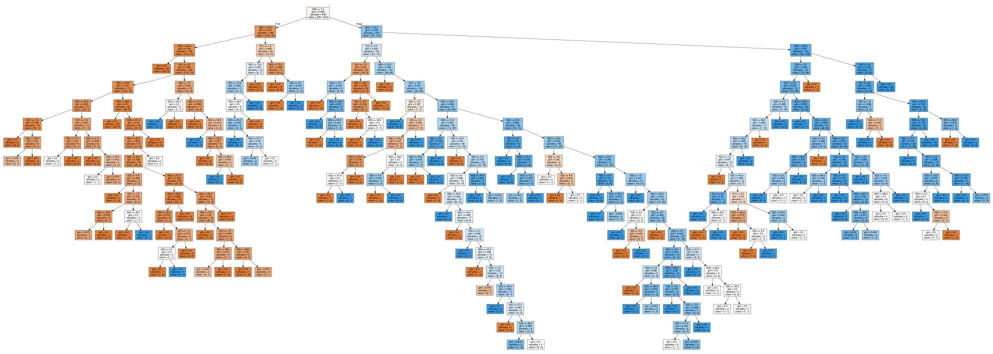
The libraries used for this question are: nltk, scipy, sklearn, xgboost, lightgbm, matplotlib, pandas, numpy, math and sys.

1.1 Dataset 1

a) Here are the results:

```
Results for Part a:
Training Accuracy is: 92.32456
Validation Accuracy is: 76.03306
Test Accuracy is: 69.56522
```

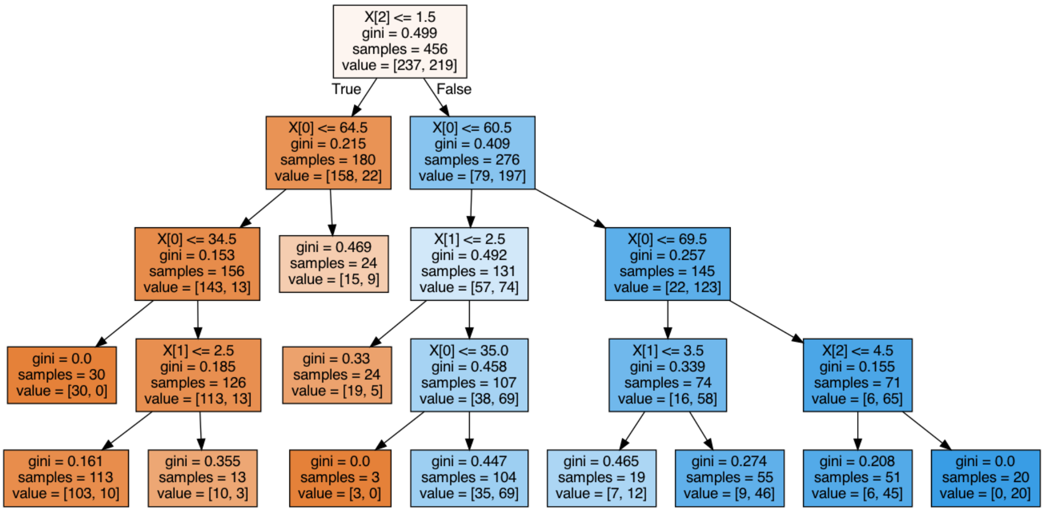
Here is the tree:



b) Here are the results:

```
Results for Part b:
Training Accuracy is: 81.93669
Validation Accuracy is: 85.92593
Test Accuracy is: 77.08333
```

Here is the tree:

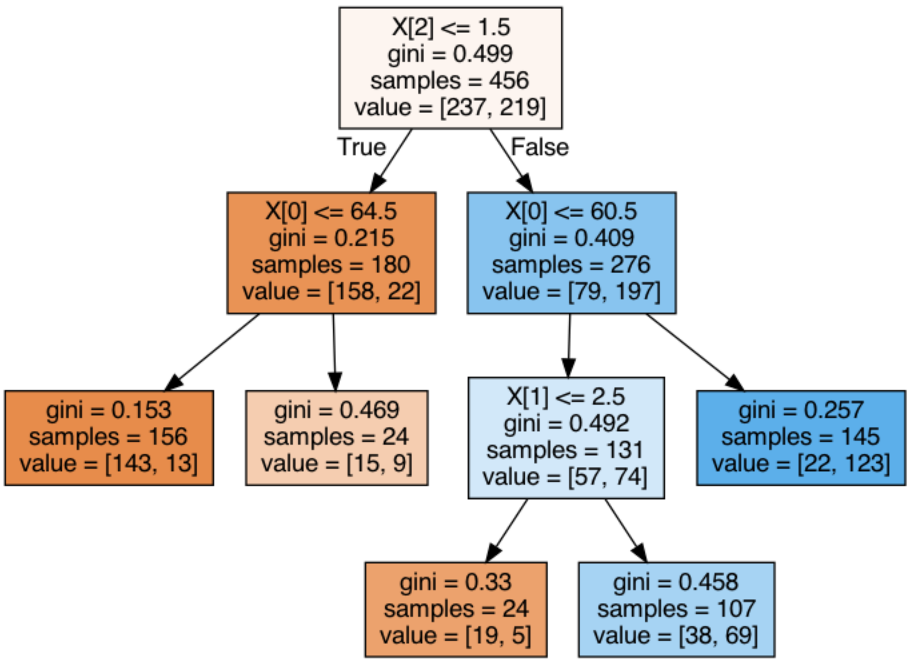


The major difference was that the size of the tree greatly reduced. In part a, the tree was huge, but in part b it very small. This effect was due to regularisation. The earlier model was overfitting the training data, but when we gave the parameters ”max-depth”, ”min-samples-split” and ”min-samples-leaf” some specific values, the overfitting reduced, and became a well-fit model.

c) The 4 plots have been saved in the folder Plots. The validation split was used to determine the best value of ccp_alpha. Here are the results:

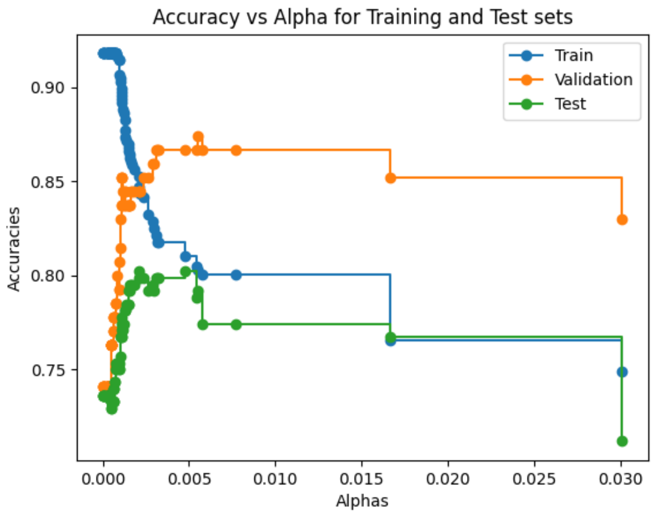
```
Results for Part c:
Training Accuracy is: 80.92105
Validation Accuracy is: 89.2562
Test Accuracy is: 75.88933
Best Alpha = 0.00737
```

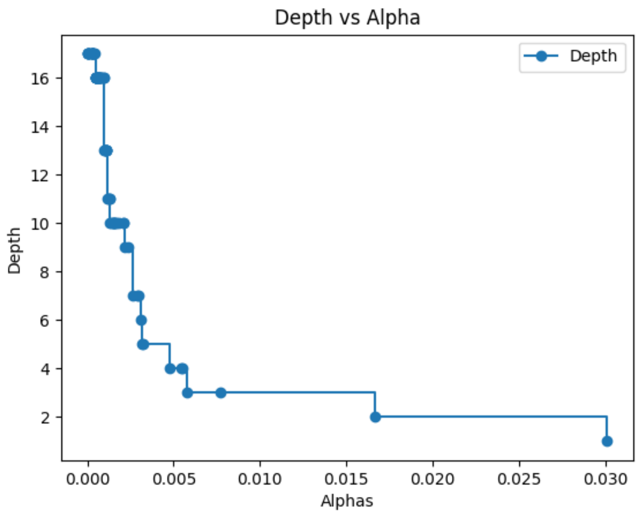
Here is the tree:

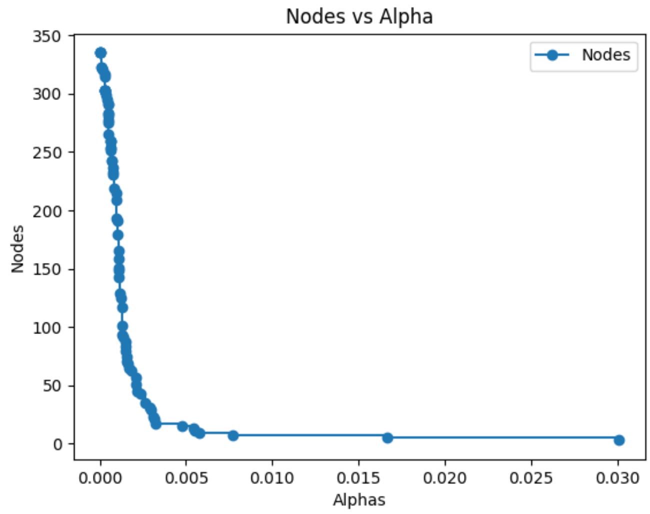


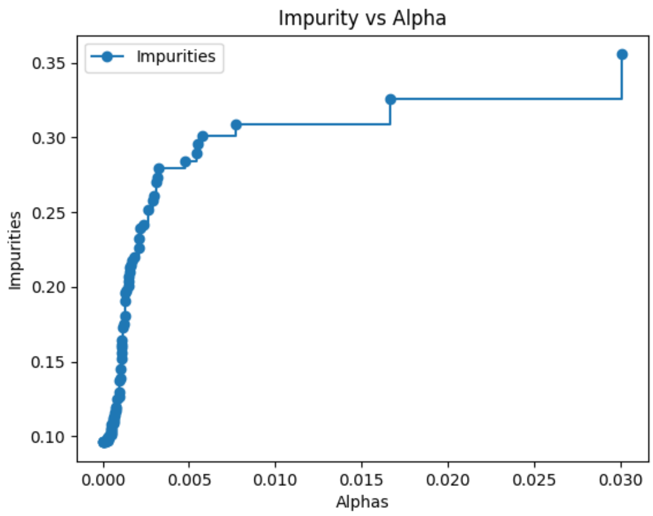
The trees in part a) and c) vary greatly in size, because a overfits the training data and c is a well-fit model, but the trees in parts b) and c) are quite small, so they regularise the model a lot. The tree in part c is the smallest one of all, it has just 9 nodes in total.

Here are the four graphs:






Nodes vs Alpha
Nodes
Alphas
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0
50
100
150
200
250
300
350


Impurity vs Alpha
Impurities
Alphas
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.10
0.15
0.20
0.25
0.30
0.35

d) Here are the results:

```
Results for Part d:
Training Accuracy is: 79.14339
Validation Accuracy is: 80.74074
Test Accuracy is: 75.69444
Out of bag Accuracy: 73.37058
```

e) Here are the results:

```
Results for 'mean' aggregate function:
Results for Part a:
Training Accuracy is: 93.29609
Validation Accuracy is: 76.2963
Test Accuracy is: 70.48611

Results for Part b:
Training Accuracy is: 81.93669
Validation Accuracy is: 86.66667
Test Accuracy is: 79.16667

Results for Part c:
Training Accuracy is: 81.19181
Validation Accuracy is: 88.14815
Test Accuracy is: 79.16667
Best Alpha = 0.00381

Results for Part d:
Training Accuracy is: 79.14339
Validation Accuracy is: 80.74074
Test Accuracy is: 75.69444
Out of bag Accuracy: 73.37058

Results for 'median' aggregate function:
Results for Part a:
Training Accuracy is: 91.80633
Validation Accuracy is: 74.07407
Test Accuracy is: 73.61111

Results for Part b:
Training Accuracy is: 81.37803
Validation Accuracy is: 86.66667
Test Accuracy is: 80.55556

Results for Part c:
Training Accuracy is: 80.26071
Validation Accuracy is: 87.40741
Test Accuracy is: 79.16667
Best Alpha = 0.00552

Results for Part d:
Training Accuracy is: 79.51583
Validation Accuracy is: 85.92593
Test Accuracy is: 78.47222
Out of bag Accuracy: 77.46741
```

My observation is, when using the mean aggregate function, the training accuracies tend to be better, but when using the median aggregate function, the test accuracies are better. Therefore, the more generalisable metric is to use the median as the aggregate function.

f) Here are the results:

```
Results for Part f:
Training Accuracy is: 80.81937
Validation Accuracy is: 85.92593
Test Accuracy is: 78.81944
```

1.2 Dataset 2

a) Here are the results:

```
Results for Part a:
Training Accuracy is: 100.0
Validation Accuracy is: 58.10618
Test Accuracy is: 57.80233
```

b) Here are the results:

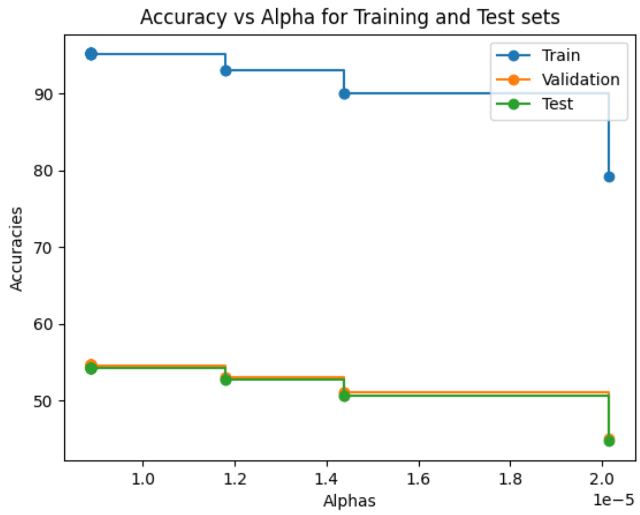
```
Results for Part b:
Training Accuracy is: 47.07461
Validation Accuracy is: 36.64883
Test Accuracy is: 36.29803
```

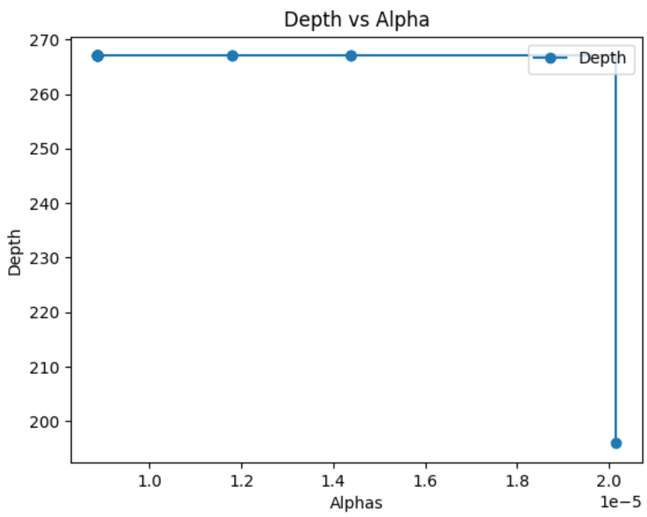
c) Here are the results:

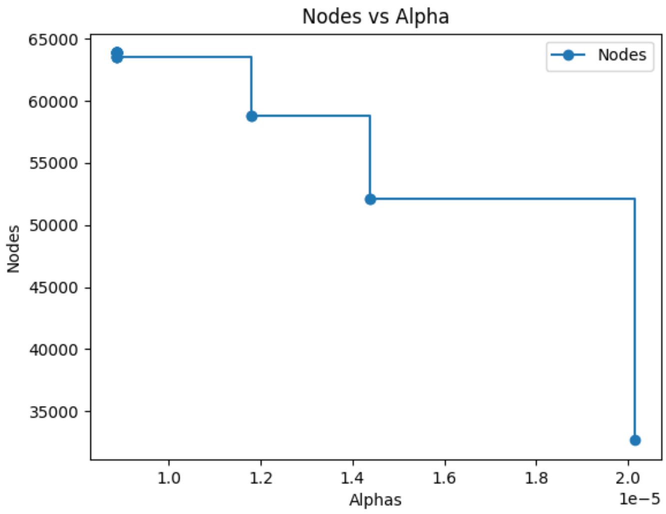
```
Results for Part c:
Training Accuracy is: 95.09158
Validation Accuracy is: 54.59712
Test Accuracy is: 54.19782
Best Alpha = 1e-05
```

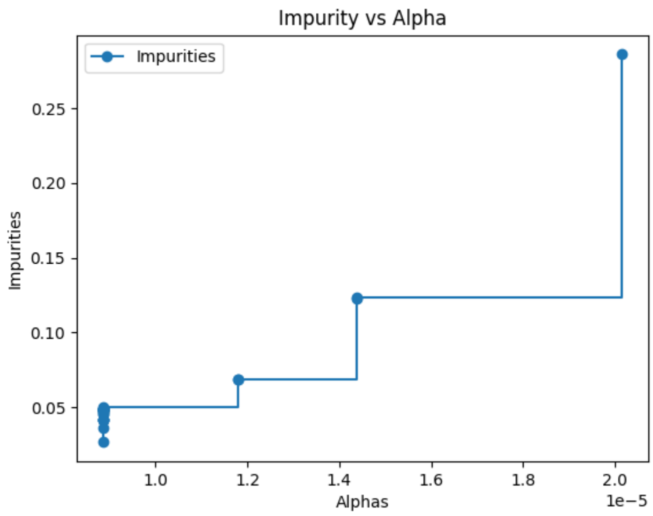
Since the dataset was too big, I used the Golden Search Method to find the optimal alpha.

Here are the four graphs:






Nodes vs Alpha
Nodes
65000
60000
55000
50000
45000
40000
35000
1.0
1.2
1.4
1.6
1.8
2.0
1e−5
Alphas


Impurity vs Alpha
Impurities
0.25
0.20
0.15
0.10
0.05
1.0
1.2
1.4
1.6
1.8
2.0
1e−5
Alphas

d) Here are the results:

```
Results for Part d:
Training Accuracy is: 31.61246
Validation Accuracy is: 31.61049
Test Accuracy is: 31.64825
Out of bag Accuracy: 31.61246
```

e) Here are the results:

```
Results for Part e:
Training Accuracy is: 100.0
Validation Accuracy is: 65.10777
Test Accuracy is: 65.20106
```

f) Here are the results:

```
Results for Part f:
Training Accuracy is: 40.85
Validation Accuracy is: 38.30
Test Accuracy is: 38.14
```

Running time for parts -

(a) Part a - 97.17 seconds

(b) Part b - 6924 seconds

(c) Part c - 3561.5 seconds

(d) Part d - 25147 seconds

(e) Part e - 135176 seconds

(f) Part f - 258 seconds

g) The running time for part g was too much, it did not complete its run. I have written the code and submitted it.

2 Question 2

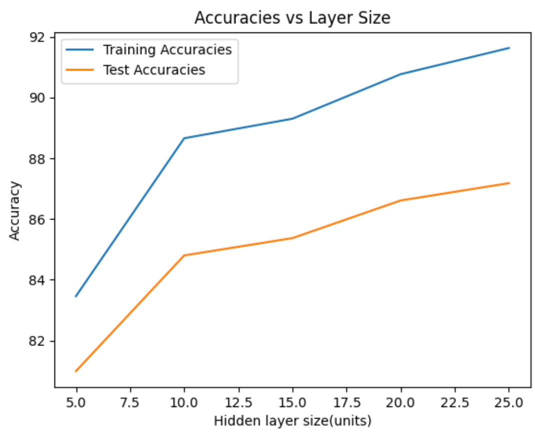
The libraries used for this part are: sklearn, matplotlib, numpy, pandas and sys.

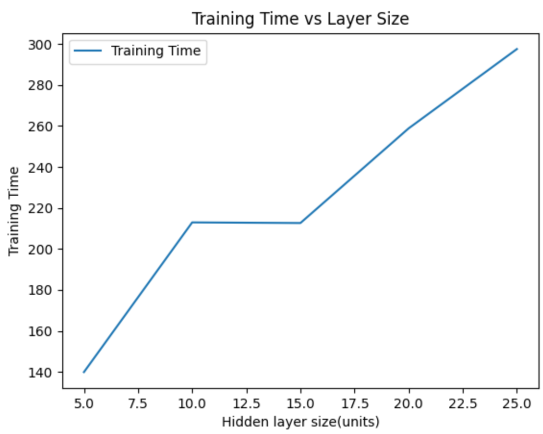
a) There are no results to show for part A. The neural network architecture for generalised parameters was built in this part.

b) The 5 confusion matrices for each of the hidden units value have been stored in a folder named 'ConfusionMatrices'. The relevant images for this part are those with Part b in their names.

The Stopping Criterion is: When the difference between 2 consecutive cost function values is lesser than 1e-9.

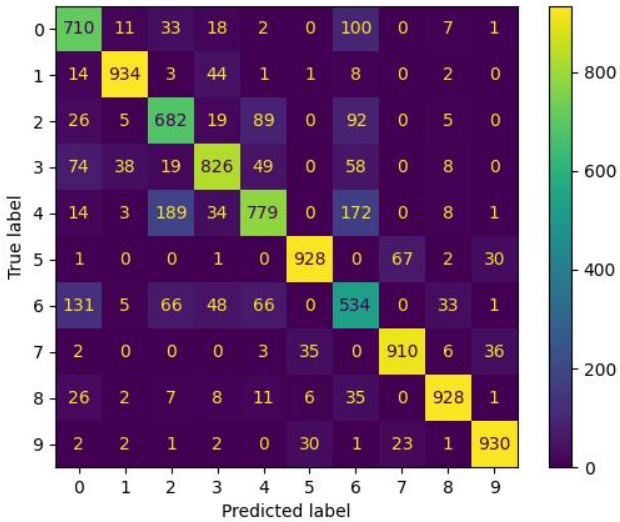
The graphs plotting the Accuracies vs Layer Size and Training Times vs Layer Size are as shown below.

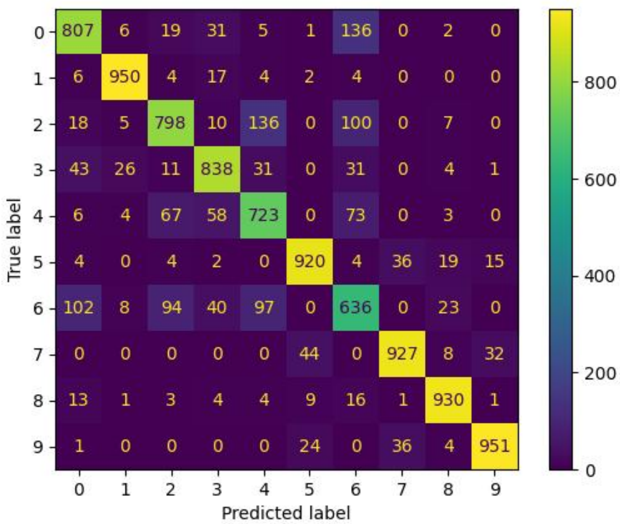


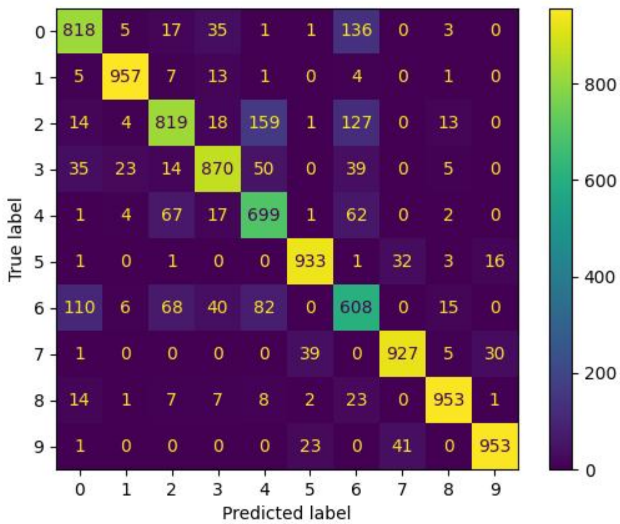


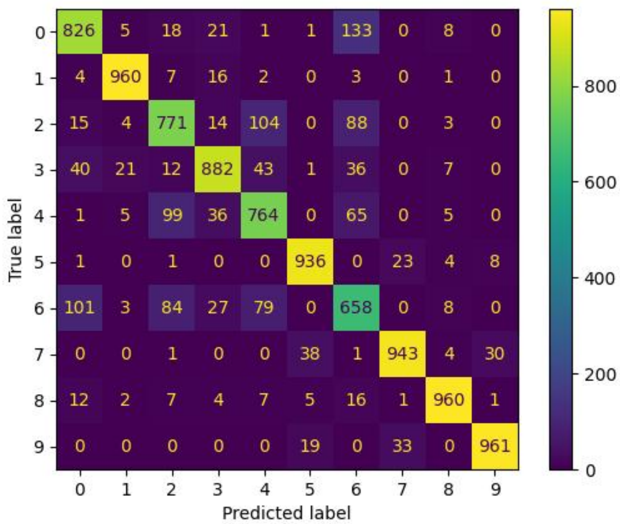
The Accuracies and Training Times are stored in the output file for this part.

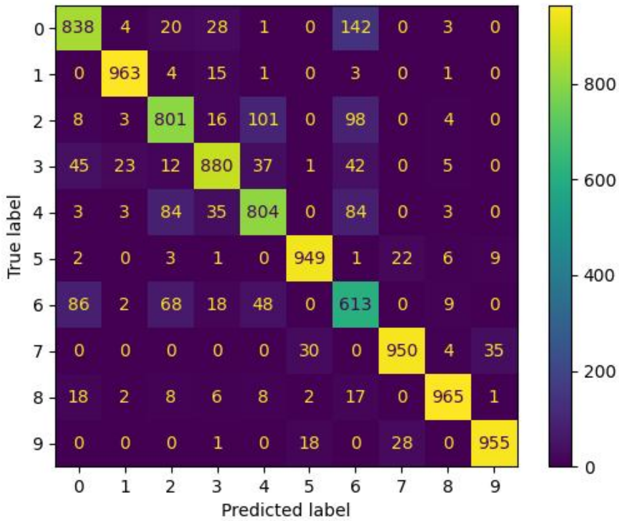
Here are the 5 confusion matrices, for number of hidden units = 5,10,15,20,25 respectively:





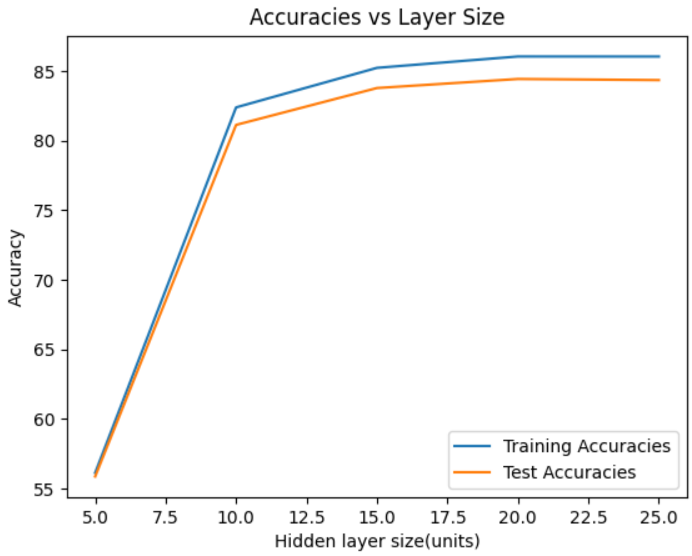

True label
Predicted label

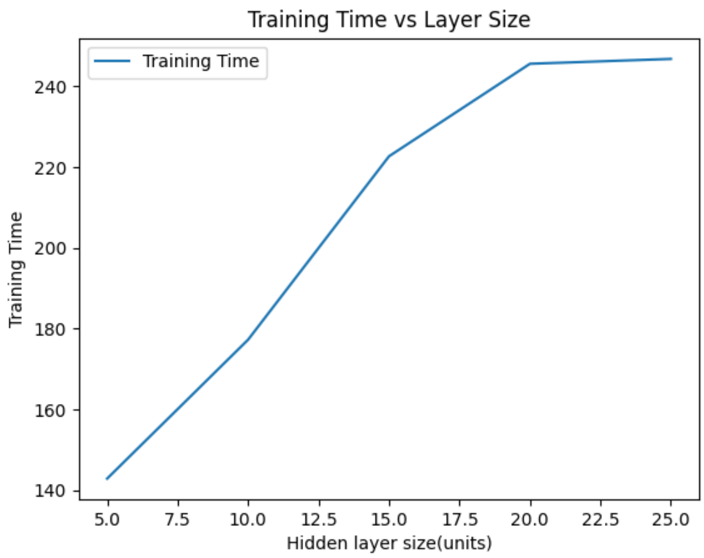

True label
Predicted label


True label
Predicted label
0
1
2
3
4
5
6
7
8
9
838 4 20 28 1 0 142 0 3 0
0 963 4 15 1 0 3 0 1 0
8 3 801 16 101 0 98 0 4 0
45 23 12 880 37 1 42 0 5 0
3 3 84 35 804 0 84 0 3 0
2 0 3 1 0 949 1 22 6 9
86 2 68 18 48 0 613 0 9 0
0 0 0 0 0 30 0 950 4 35
18 2 8 6 8 2 17 0 965 1
0 0 0 1 0 18 0 28 0 955
800
600
400
200
0

c) The training time actually became slower on the implementation of an adaptive learning rate. The Stopping Criterion is: When the difference between 2 consecutive cost function values is lesser than 1e-9.

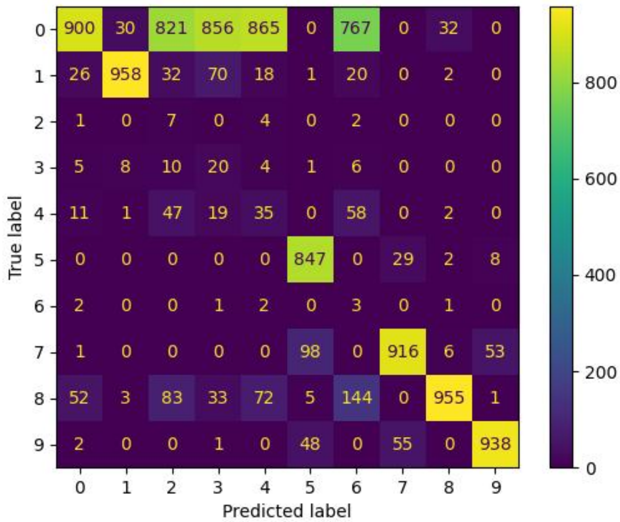
The graphs plotting the Accuracies vs Layer Size and Training Times vs Layer Size are as shown below.

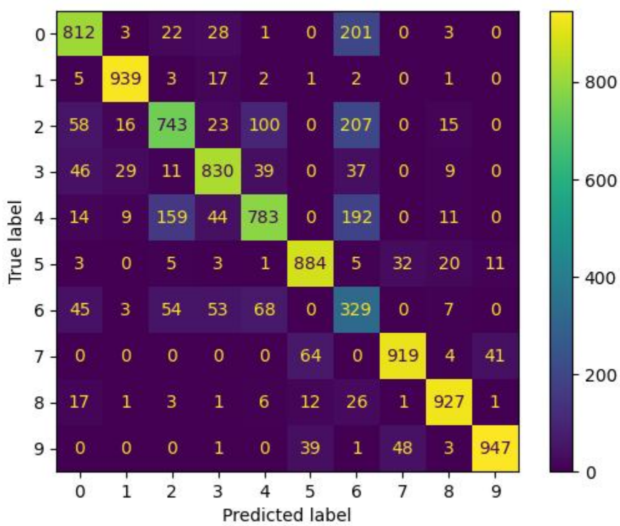


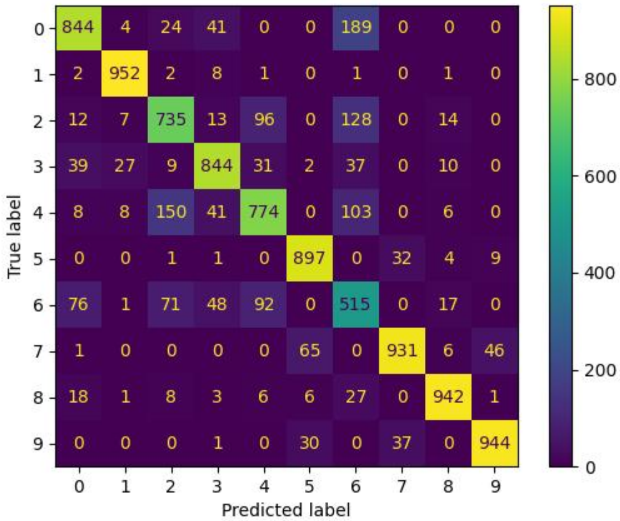


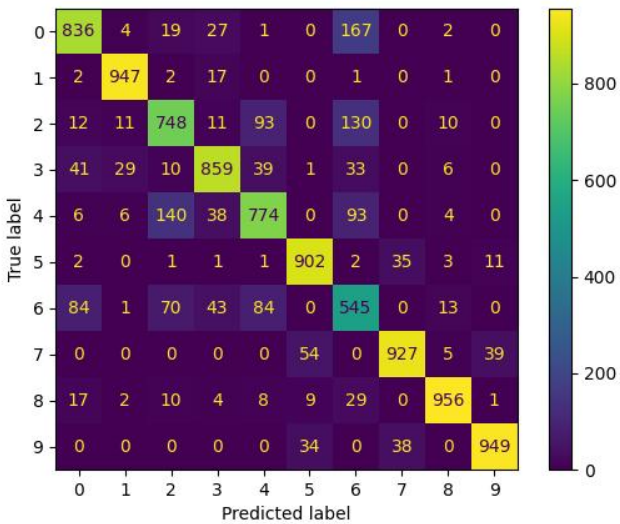
The Accuracies and Training Times are stored in the output file for this part.

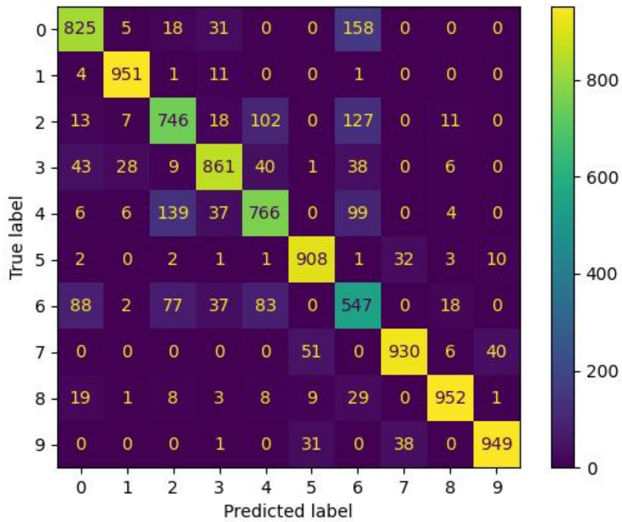
Here are the 5 confusion matrices, for number of hidden units = 5,10,15,20,25 respectively:






True label
Predicted label


True label
Predicted label

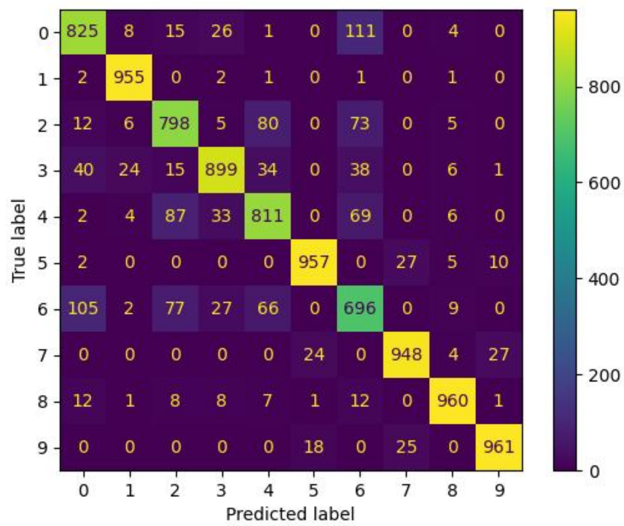

True label
Predicted label

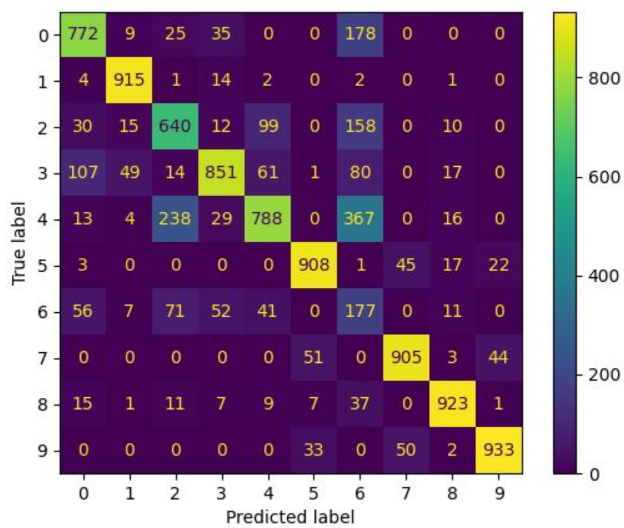
d) • The Accuracies of the architecture using Relu activation are: **92.45%(Training)** and **88.22%(Test)**

- The Accuracies of the architecture using Sigmoid activation are: **79.18%(Training)** and **78.12%(Test)**

The Relu activation works way better, and it was expected too, because of the vanishing gradient of the sigmoid at higher values.

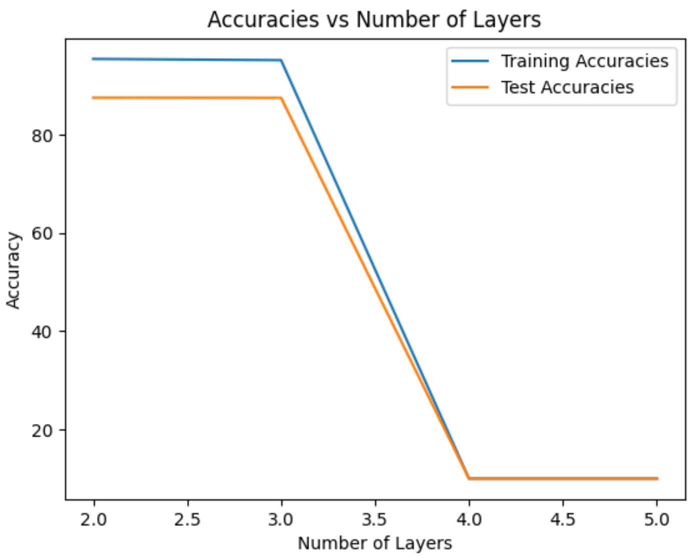
Here are the Confusion Matrices in the order: Relu, Sigmoid:



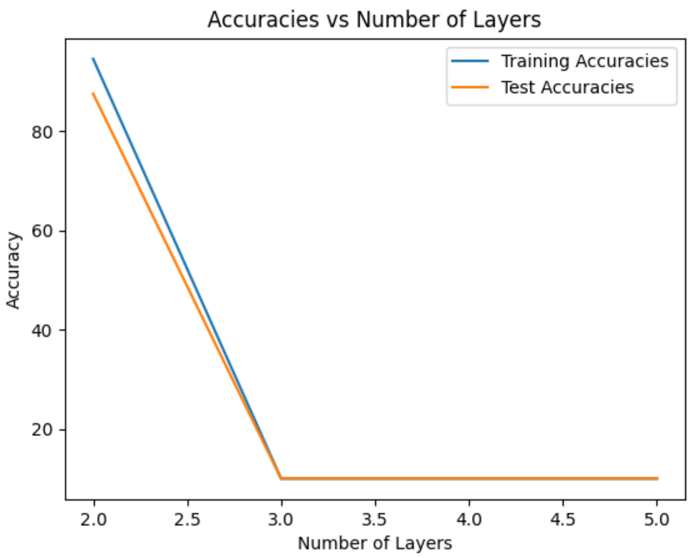


e) The best architecture turned out to be the one with Relu activation and **2** layers. It gave the highest training and test accuracies. Although the difference from the results of layer size = 3 was not significant , the former performed marginally better. Therefore, this is the architecture I shall use in parts f and g (layer size = 2).

- Graph of Accuracies vs Layers for Relu Activation



- Graph of Accuracies vs Layers for Sigmoid Activation



f)
- Derivative for BCE with respect to the output of each neuron in the last layer $\frac{dJ}{dO_L} = -\frac{y}{O_L} + \frac{1-y}{1-O_L}$.

  Now, this is multiplied with the derivative of the sigmoid function, i.e. $O_L * (1 - O_L)$. After this multiplication, it simplifies to $-y * (1 - O_L) + (1 - y) * O_L$. This avoid any divide by 0 errors.
- Training Accuracy is = **96.54%**
- Test Accuracy is = **87.05%**

g)
- Training Accuracy using MLPClassifier is = **91.65%**
- Test Accuracy using MLPClassifier is = **86.82%**

The performance of the library is actually worse than the performance of our architecture, in both the training and test sets.