COL774:A3

Sidharth Agarwal 2019CS50661

October 30, 2022

1 Libraries Used:

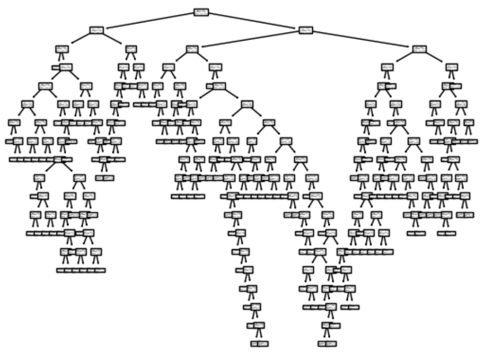
matplotlib, numpy, pandas, lightbgm, xgboost, sklearn, nltk

Random Forests Dataset 1

1.1.a

The results are as follows after training by removing the missing data point rows.

- Training Accuracy : 0.923
- Testing Accuracy : 0.683
- Validation Accuracy : 0.768



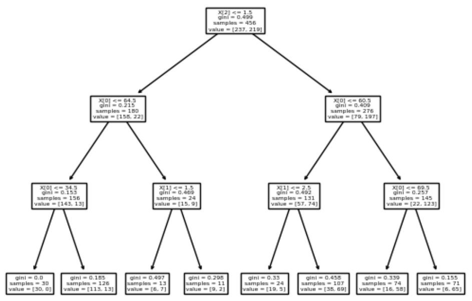
1.1.b

I did a grid search on the following parameters

('criterion' :['gini', 'entropy'], 'max_depth' : range(1,11), 'min_samples_split': range(2,11), 'min_samples_leaf': range(1,11))

Now the optimal parameters obtained after grid search were 'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 10, 'min_samples_split': 2

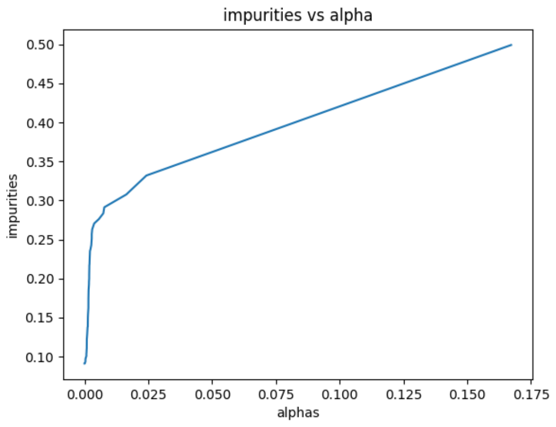
- Training Accuracy : 0.811
- Testing Accuracy : 0.754
- Validation Accuracy : 0.876

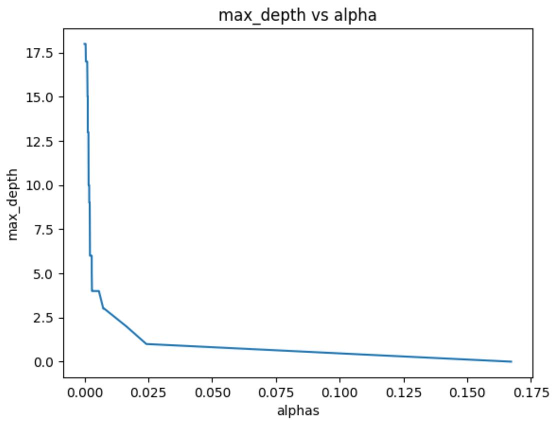


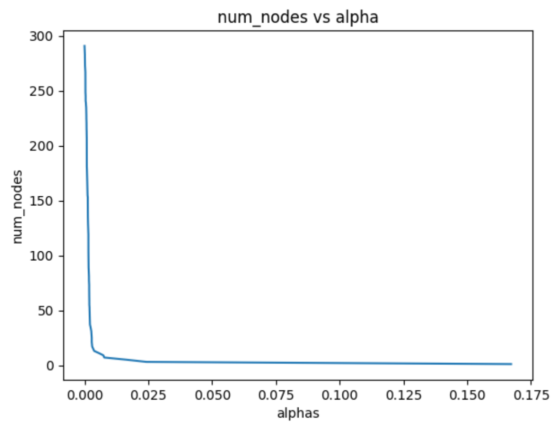
So it can clearly be observed that the tree obtained in this part is smaller in total size as well as in total depth. So this results in less over-fitting over the training data, which is also consistent with our experimental results.

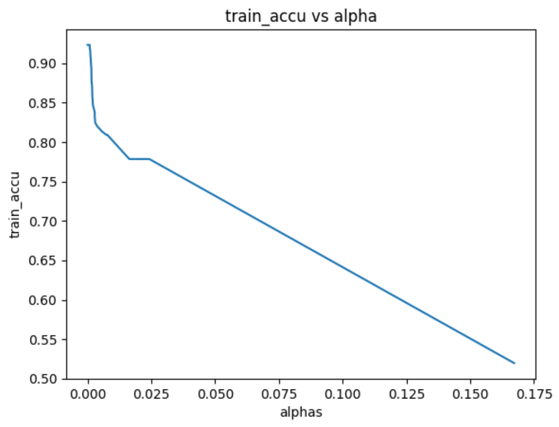
1.1.c

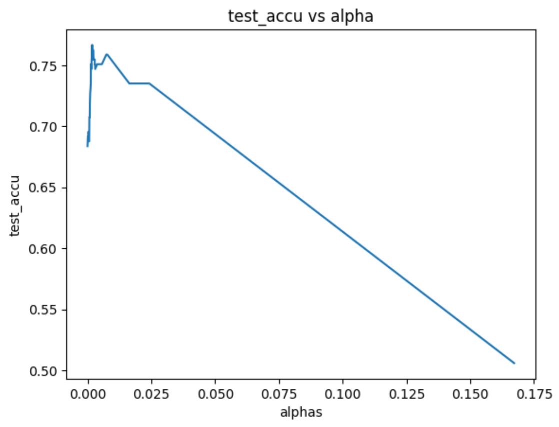
The graphs obtained are as follows..

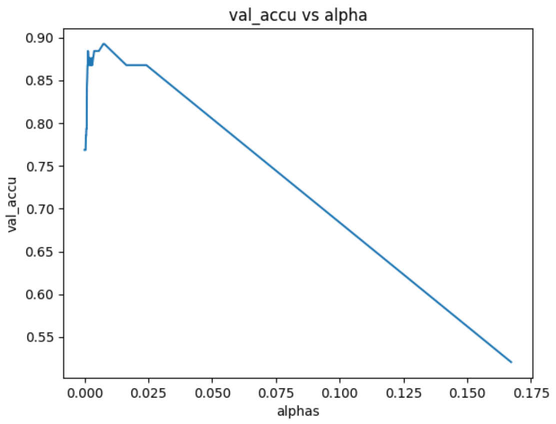





num_nodes vs alpha
num_nodes
alphas
0
50
100
150
200
250
300
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175


train_accu vs alpha
train_accu
alphas
0.50
0.55
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175

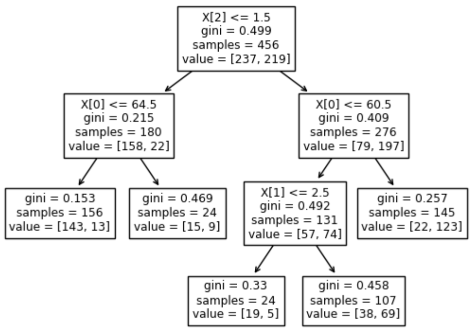

test_accu vs alpha
test_accu
alphas
0.50
0.55
0.60
0.65
0.70
0.75
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175



It can be observed that purning always results in less overfitting so training accuracies always decrease. Now at the same time it causes validation/testing accuracy to increase for some pruning which also later starts decreasing because after too much pruning the model looses its complexity to fit the data.

Now the accuracies for the best classifier according to validation accuracy is

- Training Accuracy : 0.809
- Testing Accuracy : 0.758
- Validation Accuracy : 0.892



The difference of this tree from the tree in part b is that it is a litter bit less complex(i.e less max depth and smaller size), and hence kinda underfits the data a little by giving us less training accuracy and slightly more validation accuracy.

1.1.d

Now tuning the best parameters in grid search using out of bag accuracy over the following range of parametes

'criterion' :['gini', 'entropy'], 'n_estimators' : [50,100,150,200], 'max_features' : ["sqrt", "log2", None], 'min_samples_split': [2,4,6,8,10]

We get the optimal parameters to be 'criterion': 'entropy', 'max_features': None, 'min_samples_split': 10, 'n_estimators': 50

- Training Accuracy : 0.870
- Testing Accuracy : 0.770

- Validation Accuracy : 0.876
- Out of Bag Accuracy : 0.763

1.1.e

Now for using the data impute technique as

MEAN:

Part a

- Training Accuracy : 0.933
- Testing Accuracy : 0.708
- Validation Accuracy : 0.755

Part b
Now doing a grid search on the same range of parameters we get the optimal parameters for this technique to be 'criterion': 'entropy', 'max_depth': 8, 'min_samples_leaf': 2, 'min_samples_split': 9. With the acuracies as follows.

- Training Accuracy : 0.849
- Testing Accuracy : 0.777
- Validation Accuracy : 0.844

Part c
Now the accuracies for the best classifier according to the validation accuracy is

- Training Accuracy : 0.811
- Testing Accuracy : 0.770
- Validation Accuracy : 0.874

Part d
Now grid searching for the same set of range of parameters, we get the optimal parameters to be 'criterion': 'entropy', 'max_features': None, 'min_samples_split': 10, 'n_estimators': 50

- Training Accuracy : 0.869
- Testing Accuracy : 0.784
- Validation Accuracy : 0.851
- OOB Score : 0.744

MEDIAN

: **Part a**

- Training Accuracy : 0.918
- Testing Accuracy : 0.732
- Validation Accuracy : 0.740

Part b
Now doing a grid search on the same range of parameters we get the optimal parameters for this technique to be 'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 3, 'min_samples_split': 2. With the acuracies as follows.

- Training Accuracy : 0.815
- Testing Accuracy : 0.809
- Validation Accuracy : 0.874

Part c
Now the accuracies for the best classifier according to the validation accuracy is

- Training Accuracy : 0.803
- Testing Accuracy : 0.791
- Validation Accuracy : 0.874

Part d
Now grid searching for the same set of range of parameters, we get the optimal parameters to be 'criterion': 'entropy', 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 50

- Training Accuracy : 0.856
- Testing Accuracy : 0.781
- Validation Accuracy : 0.851
- OOB Score : 0.754

Now it can be clearly observed that all the three no imputing, median and mean perform almost the same, with the mean performing very slightly better than the other two in general.

1.1.f

Now doing grid search over the given parameters range for the XGBoost classifier. We get the optimal parameters to be 'max_depth': 5, 'n_estimators': 10, 'subsample': 0.4. With the accuracies as follows.

- Training Accuracy : 0.828
- Testing Accuracy : 0.762
- Validation Accuracy : 0.884

Part 1 Dataset 2

For this dataset I also didn't consider words which appeared in the whole collection for less than 2 times, just to decrease the computation time and the accuracy was unaffected by the same.

1.2.a

The accuracies obtained are as follows.

- Training Accuracy : 0.999
- Testing Accuracy : 0.0.572
- Validation Accuracy : 0.573

The time taken for training was 67.5 seconds

1.2.b

Now I did a grid search on the following range of parameters

'max_depth' : [2,4,8], 'min_samples_split': [2,4,8], 'min_samples_leaf': [2,4,8]

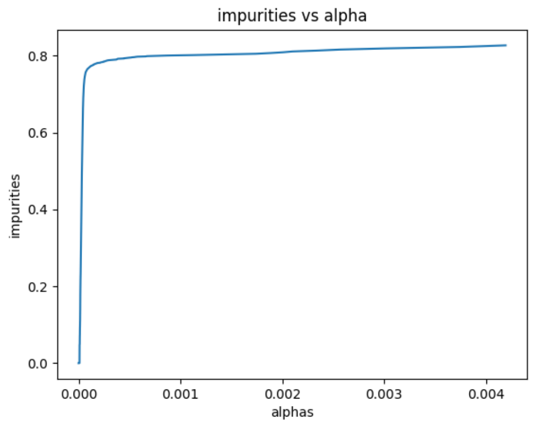
The optimal parameters obtained are as follows: 'max_depth': 8, 'min_samples_leaf': 2, 'min_samples_split': 2

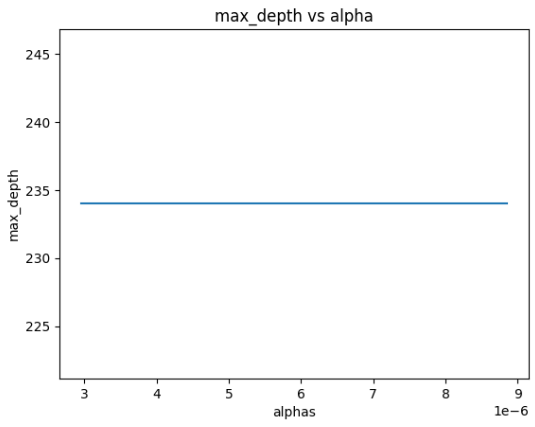
- Training Accuracy : 0.343
- Testing Accuracy : 0.337
- Validation Accuracy : 0.342

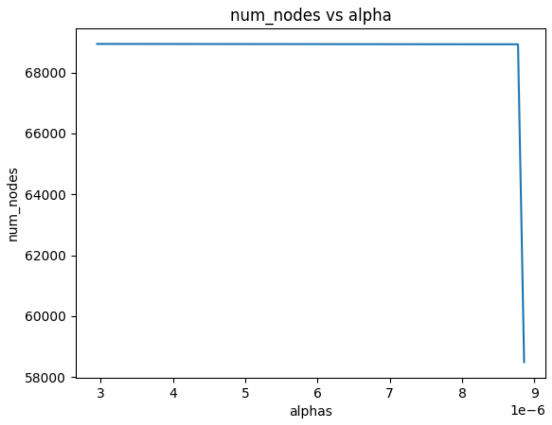
As we can clearly see that all the performance accuracies have decreased. This clearly indicates that the model is no longer complex enough to fit the data. The total time taken for part b was 501.3s.

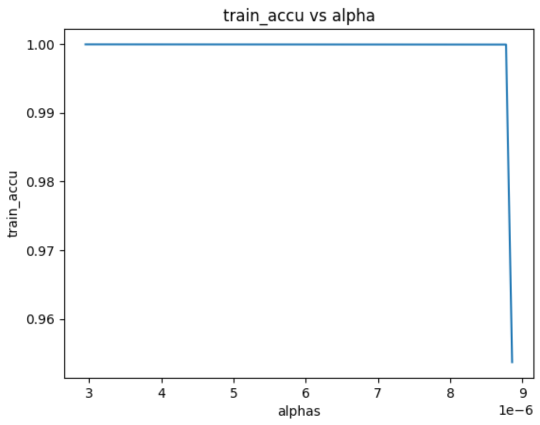
1.2.c

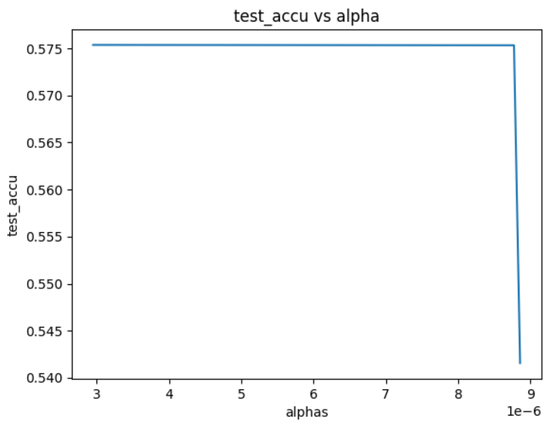
So since in this dataset this was taking more time than expected so after manually testing for various alphas I decided to plot the graphs only for 8 alphas which lied in the range of 1e-5 to 1e-6, to reduce computation time.

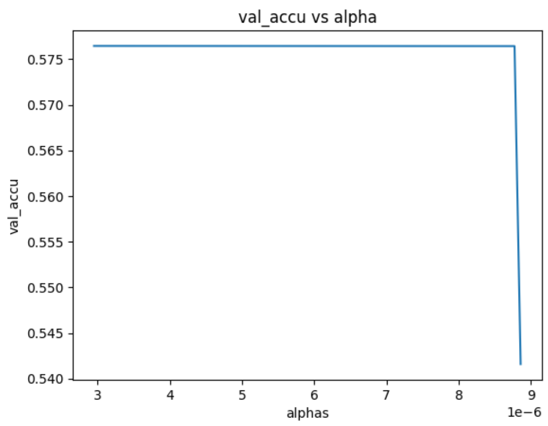





num_nodes vs alpha
num_nodes
68000
66000
64000
62000
60000
58000
3
4
5
6
7
8
9
alphas
1e−6


train_accu vs alpha
train_accu
1.00
0.99
0.98
0.97
0.96
3
4
5
6
7
8
9
alphas
1e−6


test_accu vs alpha
test_accu
0.575
0.570
0.565
0.560
0.555
0.550
0.545
0.540
3
4
5
6
7
8
9
alphas
1e−6



Now the accuracies for the best classifier according to validation accuracy is

- Training Accuracy : 0.999
- Testing Accuracy : 0.575
- Validation Accuracy : 0.574

1.2.d

After running the random forests for a subset of parameters because of limited computation power, I found the optimal parameters 'max_features': 0.7, 'min_samples_split': 6, 'n_estimators': 50, with results as below.

- Training Accuracy : 0.997
- Testing Accuracy : 0.635
- Validation Accuracy : 0.636

1.2.e

After running it using XGBoost library I found the optimal parameters to be 'subsample': 0.5, 'max_depth': 60, 'n_estimators': 50, with results as below.

- Training Accuracy : 0.925
- Testing Accuracy : 0.608
- Validation Accuracy : 0.606

1.2.f

After using the lightGBM classifier and testing it for a only few(due to limited computation power) hyper-parameters I found the readings to be.

- Training Accuracy : 0.714
- Testing Accuracy : 0.414
- Validation Accuracy : 0.417
- Time Taken : 162.07 seconds

Now the time taken for part (a) i.e normal decision tree was 67 seconds, part (b) i.e grid search over decision tree classifier finished in 500 seconds. For part (c), in which we only cared about 8 models with different alphas, the total time taken came out to be approximately 25 minutes and here in lightgbm it was only 162.07 seconds.

1.2.g

I have written the code for the same, but due to the shortage of resources provided us, I was not able to run the experiments for the same.

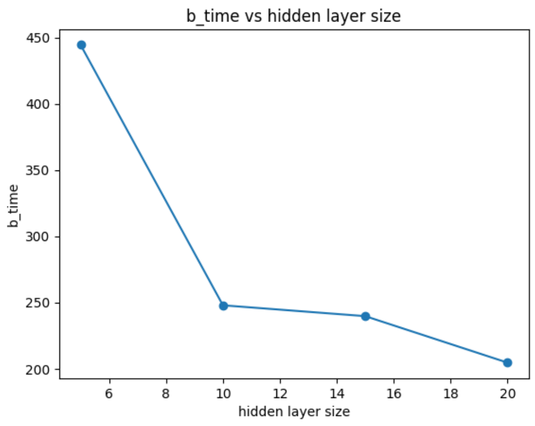
Part 2 Neural Network

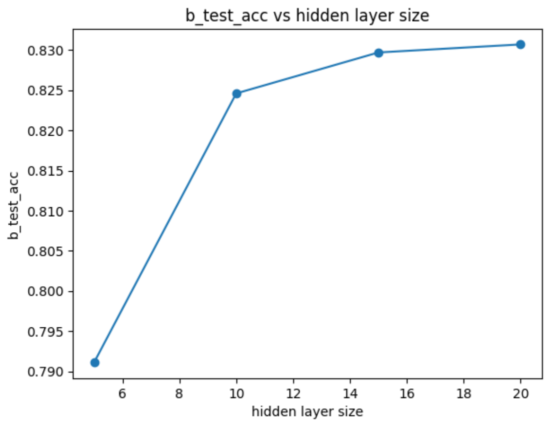
2.a

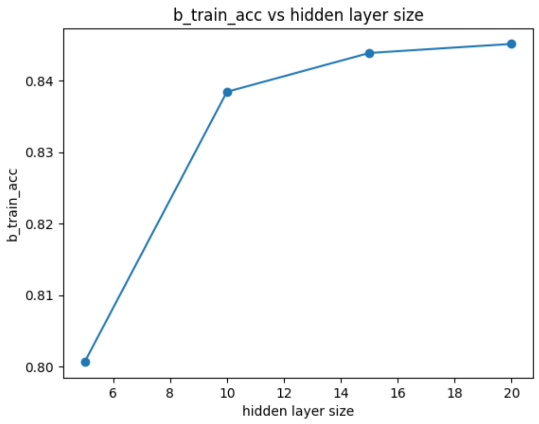
I have implemented the neural network architecture from scratch in my code.

2.b

The stopping criteria that I used was if the difference between the cost of after and before training for an epoch is less than epsilon or if the number of epochs trained have exceeded the max number of epochs allowed then I terminate. Using the learning rate to be 0.1, batch size to be 100 and maximum number of epochs to be 200.



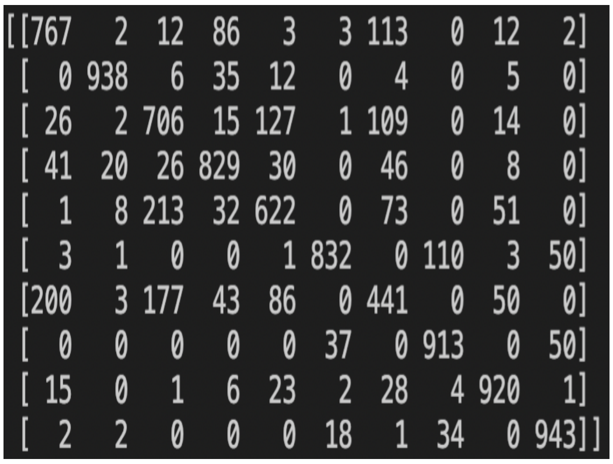




Now it can be noted that the training time decreases with the increase in hidden layer size, this is because as we add more perceptrons to the hidden layer we give the model more complexity, which results in model to easily fit on the data. Now this also increases the training and testing accuracy because of the same reason.

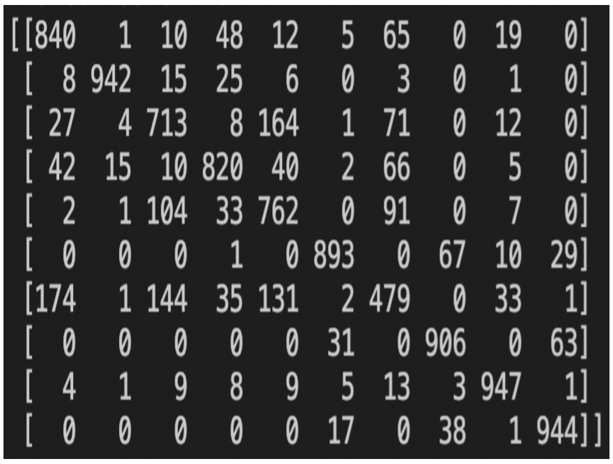
The confusion matrix obtained for hidden layer size(H) =

H = 5



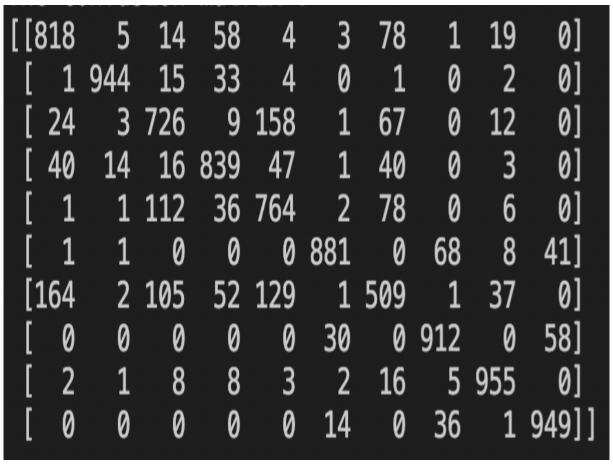
```
[[767   2  12  86   3   3 113   0  12   2]
 [  0 938   6  35  12   0   4   0   5   0]
 [ 26   2 706  15 127   1 109   0  14   0]
 [ 41  20  26 829  30   0  46   0   8   0]
 [  1   8 213  32 622   0  73   0  51   0]
 [  3   1   0   0   1 832   0 110   3  50]
 [200   3 177  43  86   0 441   0  50   0]
 [  0   0   0   0   0  37   0 913   0  50]
 [ 15   0   1   6  23   2  28   4 920   1]
 [  2   2   0   0   0  18   1  34   0 943]]
```

H = 10



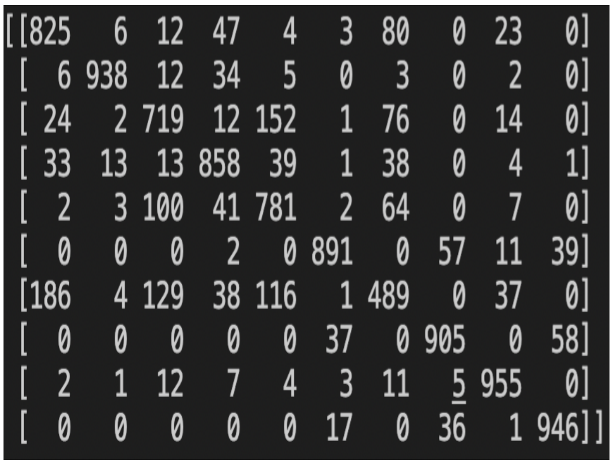
```
[[840   1  10  48  12   5  65   0  19   0]
 [  8 942  15  25   6   0   3   0   1   0]
 [ 27   4 713   8 164   1  71   0  12   0]
 [ 42  15  10 820  40   2  66   0   5   0]
 [  2   1 104  33 762   0  91   0   7   0]
 [  0   0   0   1   0 893   0  67  10  29]
 [174   1 144  35 131   2 479   0  33   1]
 [  0   0   0   0   0  31   0 906   0  63]
 [  4   1   9   8   9   5  13   3 947   1]
 [  0   0   0   0   0  17   0  38   1 944]]
```

H = 15



```
[[818   5  14  58   4   3  78   1  19   0]
 [  1 944  15  33   4   0   1   0   2   0]
 [ 24   3 726   9 158   1  67   0  12   0]
 [ 40  14  16 839  47   1  40   0   3   0]
 [  1   1 112  36 764   2  78   0   6   0]
 [  1   1   0   0   0 881   0  68   8  41]
 [164   2 105  52 129   1 509   1  37   0]
 [  0   0   0   0   0  30   0 912   0  58]
 [  2   1   8   8   3   2  16   5 955   0]
 [  0   0   0   0   0  14   0  36   1 949]]
```

H = 20

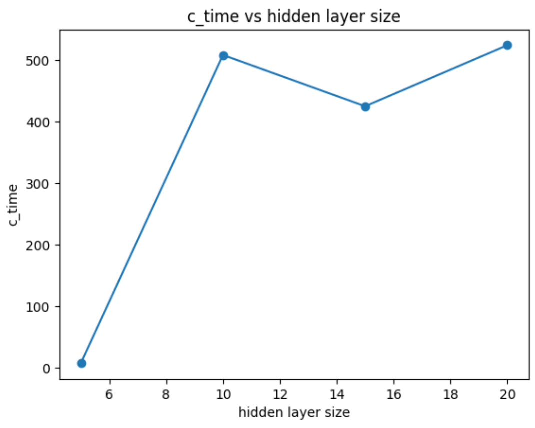


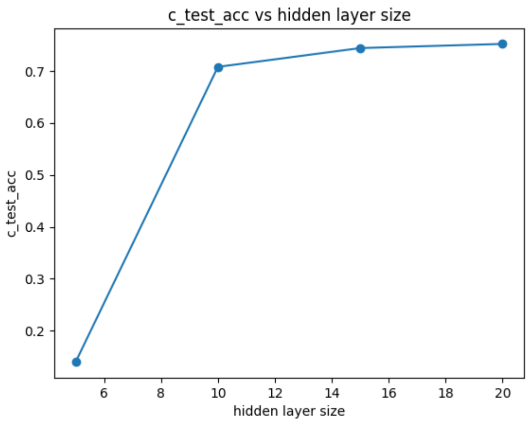
```
[[825   6  12  47   4   3  80   0  23   0]
 [  6 938  12  34   5   0   3   0   2   0]
 [ 24   2 719  12 152   1  76   0  14   0]
 [ 33  13  13 858  39   1  38   0   4   1]
 [  2   3 100  41 781   2  64   0   7   0]
 [  0   0   0   2   0 891   0  57  11  39]
 [186   4 129  38 116   1 489   0  37   0]
 [  0   0   0   0   0  37   0 905   0  58]
 [  2   1  12   7   4   3  11   5 955   0]
 [  0   0   0   0   0  17   0  36   1 946]]
```

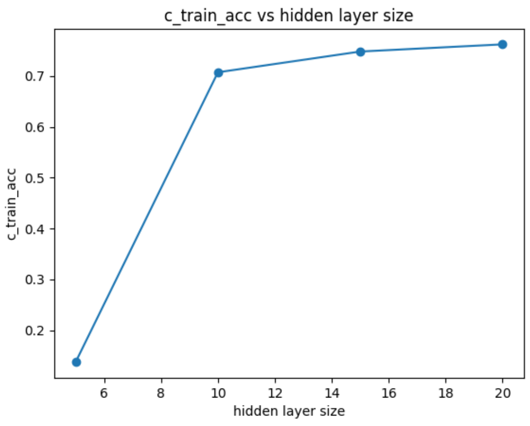
Now it can be noticed that as we increase the hidden layer size then the number of elements in the diagonal increases, resulting in more samples being correctly classified.

2.c

There didn't arise any need to change the stopping criteria. The plots obtained for using the adaptive learning rate are as follows.

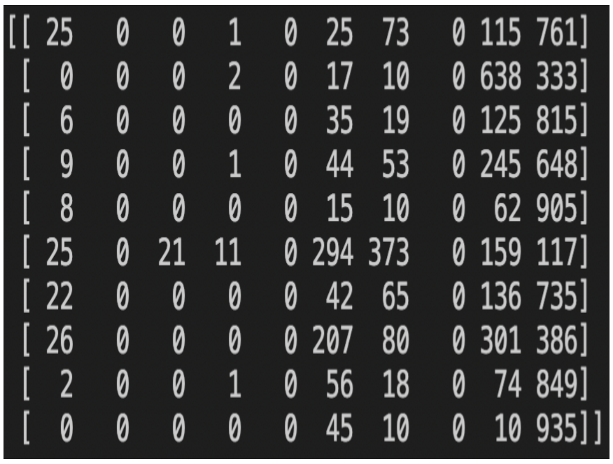




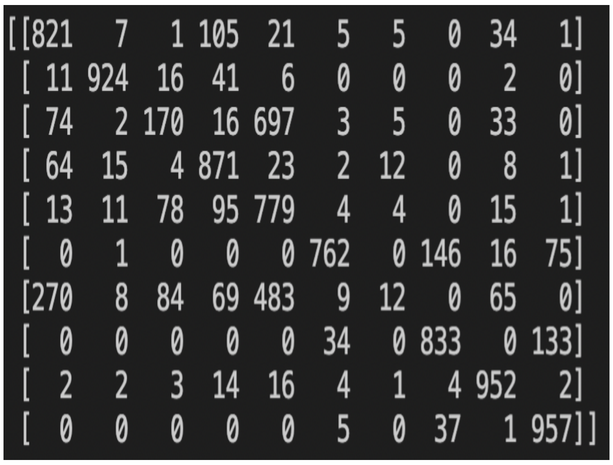


Here you can notice that the model with size 5 is not training at all and immediately converges, since the learning rate is not high enough to reflect a change in the cost. Now overall because of this adaptive learning rate the time taken turn out to be greater than that taken in the normal constant learning rate. This is because now after each epoch the change that is made to gradients is now smaller than that made previously. Also it is noticed that adaptive learning results in decreased accuracies for both test and train cases.
The confusion matrices obtained are as follows.

H = 5

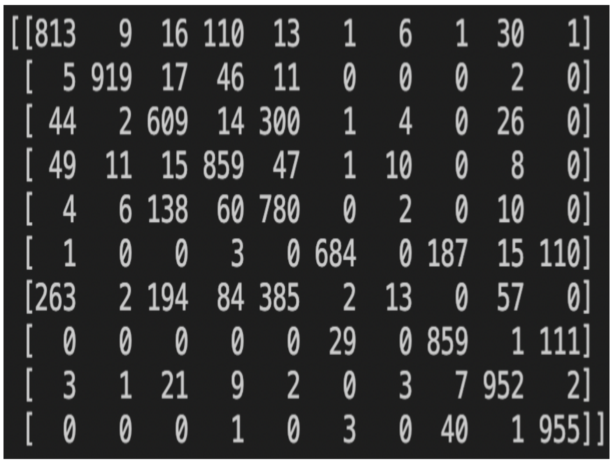

```
[[ 25   0   0   1   0  25  73   0 115 761]
 [  0   0   0   2   0  17  10   0 638 333]
 [  6   0   0   0   0  35  19   0 125 815]
 [  9   0   0   1   0  44  53   0 245 648]
 [  8   0   0   0   0  15  10   0  62 905]
 [ 25   0  21  11   0 294 373   0 159 117]
 [ 22   0   0   0   0  42  65   0 136 735]
 [ 26   0   0   0   0 207  80   0 301 386]
 [  2   0   0   1   0  56  18   0  74 849]
 [  0   0   0   0   0  45  10   0  10 935]]
```

H = 10



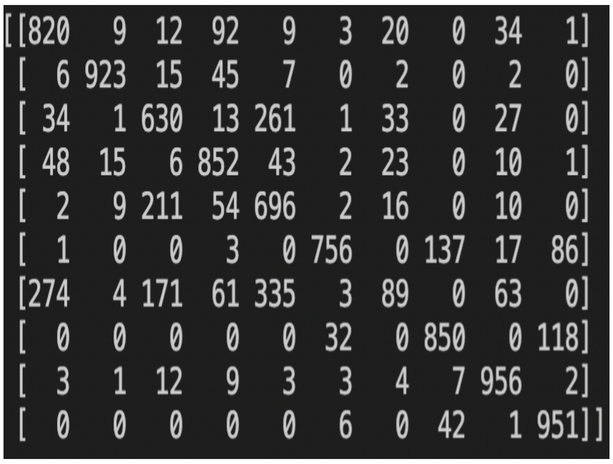
```
[[821   7   1 105  21   5   5   0  34   1]
 [ 11 924  16  41   6   0   0   0   2   0]
 [ 74   2 170  16 697   3   5   0  33   0]
 [ 64  15   4 871  23   2  12   0   8   1]
 [ 13  11  78  95 779   4   4   0  15   1]
 [  0   1   0   0   0 762   0 146  16  75]
 [270   8  84  69 483   9  12   0  65   0]
 [  0   0   0   0   0  34   0 833   0 133]
 [  2   2   3  14  16   4   1   4 952   2]
 [  0   0   0   0   0   5   0  37   1 957]]
```

H = 15



```
[[813   9  16 110  13   1   6   1  30   1]
 [  5 919  17  46  11   0   0   0   2   0]
 [ 44   2 609  14 300   1   4   0  26   0]
 [ 49  11  15 859  47   1  10   0   8   0]
 [  4   6 138  60 780   0   2   0  10   0]
 [  1   0   0   3   0 684   0 187  15 110]
 [263   2 194  84 385   2  13   0  57   0]
 [  0   0   0   0   0  29   0 859   1 111]
 [  3   1  21   9   2   0   3   7 952   2]
 [  0   0   0   1   0   3   0  40   1 955]]
```

H = 20



```
[[820   9  12  92   9   3  20   0  34   1]
 [  6 923  15  45   7   0   2   0   2   0]
 [ 34   1 630  13 261   1  33   0  27   0]
 [ 48  15   6 852  43   2  23   0  10   1]
 [  2   9 211  54 696   2  16   0  10   0]
 [  1   0   0   3   0 756   0 137  17  86]
 [274   4 171  61 335   3  89   0  63   0]
 [  0   0   0   0   0  32   0 850   0 118]
 [  3   1  12   9   3   3   4   7 956   2]
 [  0   0   0   0   0   6   0  42   1 951]]
```

Now as can be observed in comparison to constant learning rate the number of diagnoal elements have slightly decreased in this case because of reduced accuracy.

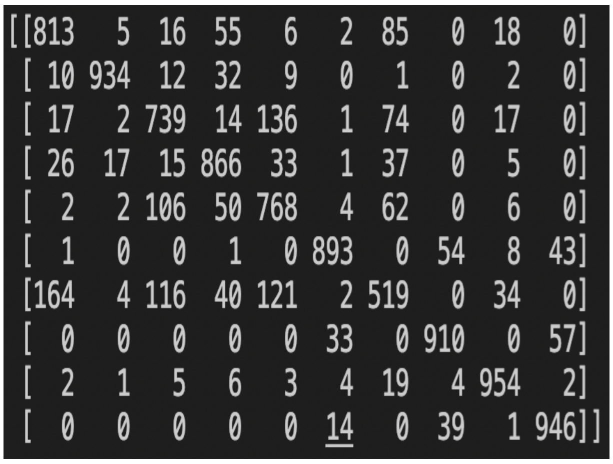
2.d

Now running for adaptive learning rates we obtain for learning rate type of

RELU:

- Training Time : 250.1s
- Training Accuracy : 0.847
- Testing Accuracy : 0.832

The confusion matrix obtained is as follows.

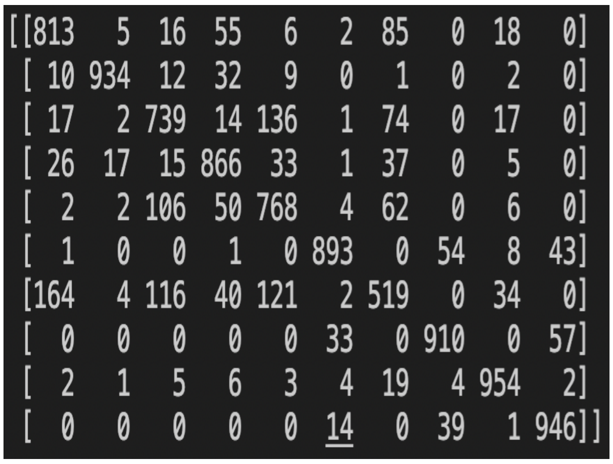


```
[[813   5  16  55   6   2  85   0  18   0]
 [ 10 934  12  32   9   0   1   0   2   0]
 [ 17   2 739  14 136   1  74   0  17   0]
 [ 26  17  15 866  33   1  37   0   5   0]
 [  2   2 106  50 768   4  62   0   6   0]
 [  1   0   0   1   0 893   0  54   8  43]
 [164   4 116  40 121   2 519   0  34   0]
 [  0   0   0   0   0  33   0 910   0  57]
 [  2   1   5   6   3   4  19   4 954   2]
 [  0   0   0   0   0  14   0  39   1 946]]
```

SIGMOID:

- Training Time : 1182.1s
- Training Accuracy : 0.739
- Testing Accuracy : 0.734

The confusion matrix obtained is as follows.



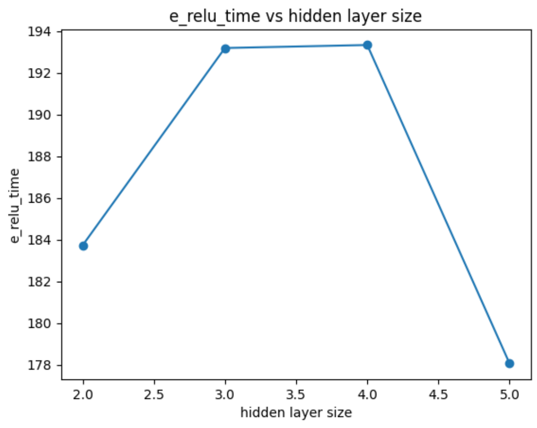
```
[[813   5  16  55   6   2  85   0  18   0]
 [ 10 934  12  32   9   0   1   0   2   0]
 [ 17   2 739  14 136   1  74   0  17   0]
 [ 26  17  15 866  33   1  37   0   5   0]
 [  2   2 106  50 768   4  62   0   6   0]
 [  1   0   0   1   0 893   0  54   8  43]
 [164   4 116  40 121   2 519   0  34   0]
 [  0   0   0   0   0  33   0 910   0  57]
 [  2   1   5   6   3   4  19   4 954   2]
 [  0   0   0   0   0  14   0  39   1 946]]
```

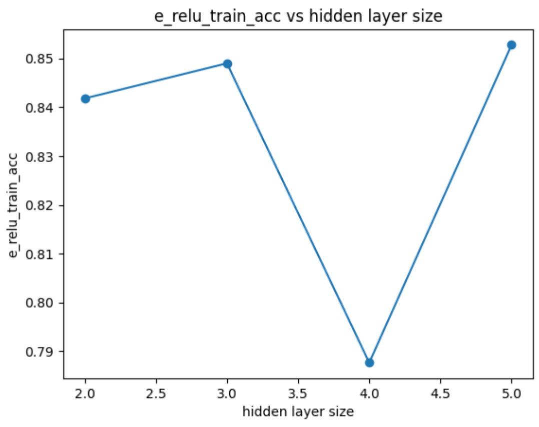
As we can clearly observe that the training time has increased going from relu to sigmoid, this is because it is easier to compute relu function itself as well as its derivative as compared to that in sigmoid. The accuracies have reduced both for training and testing as we go from relu to sigmoid., this is because sigmoid function is not able to handle vanishing gradients.

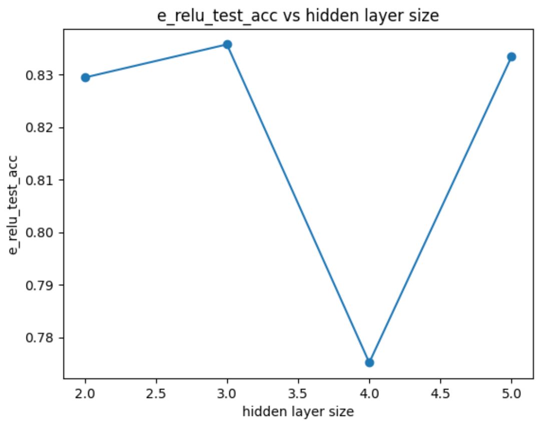
2.e

Now running the experiments for

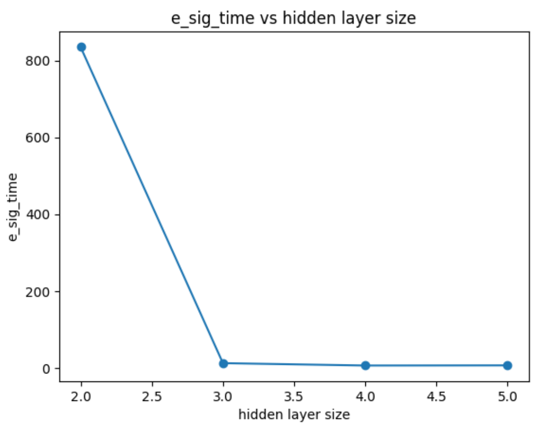
RELU:

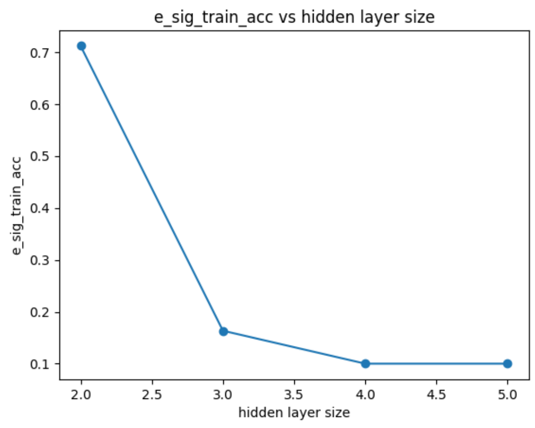

e_relu_time vs hidden layer size
e_relu_time
hidden layer size
194
192
190
188
186
184
182
180
178
2.0
2.5
3.0
3.5
4.0
4.5
5.0

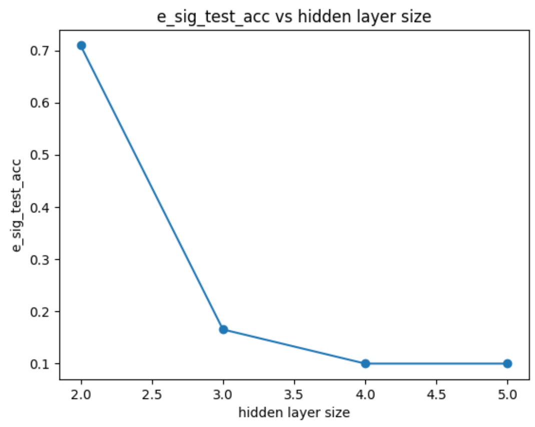

e_relu_train_acc vs hidden layer size
e_relu_train_acc
hidden layer size
0.85
0.84
0.83
0.82
0.81
0.80
0.79
2.0
2.5
3.0
3.5
4.0
4.5
5.0


e_relu_test_acc vs hidden layer size
e_relu_test_acc
hidden layer size
0.83
0.82
0.81
0.80
0.79
0.78
2.0
2.5
3.0
3.5
4.0
4.5
5.0

SIGMOID:







Now it can clearly be seen that relu consistently performs better than sigmoid, over all the architectures. Now coming to the best architecture among relu, the architecture with 3 hidden layers seems to give the best accuracy on the testing data, so we will go forward with the same. That is hidden layers being [50,50,50].

2.f

The change due to introduction of binary cross entropy was in final backpropagation layer and the cost function. The new final backpropagation layer came out to be.

$$\text{delta\_last} = (1/(\text{output - I - last\_out}))*(\text{last\_out})*(1\text{-last\_out})$$

The is also a change in the formula of the loss function and it came out to be

new_cost = sum((-y*log(forward(X)))/total_samples

Now the new accuracies came out to be came out to be very close as compared to the MSE loss and were as follows.

- Training Accuracy : 0.859
- Testing Accuracy : 0.847

2.g

Now using the hidden layers to be [50,50,50] and activation to be relu and rest the parameters same as given in the question. We get the accuracies to be

- Training Time : 530.6s
- Training Accuracy : 0.884
- Testing Accuracy : 0.845