COL 774: Machine Learning
Assignment 3

BRAJ RAJ NAGAR

2022AIB2682

aib222682@scai.iitd.ac.in

1. Decision Trees, Random Forests and Gradient Boosted Trees:

Dataset 1:

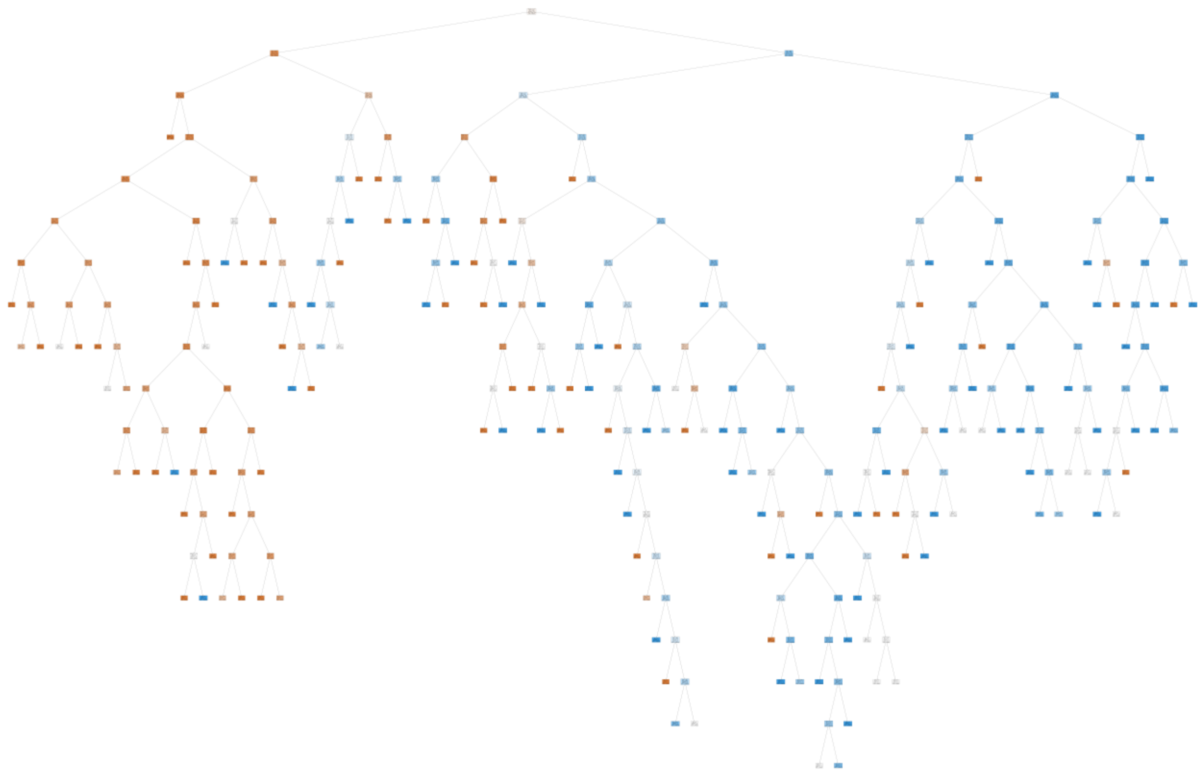
a. Decision Tree Construction and Visualization:

Missing values is ignored in examples.

Training accuracy: 92.52747252747253%

Test accuracy: 68.77470355731226 %

Validation accuracy: 76.03305785123967%



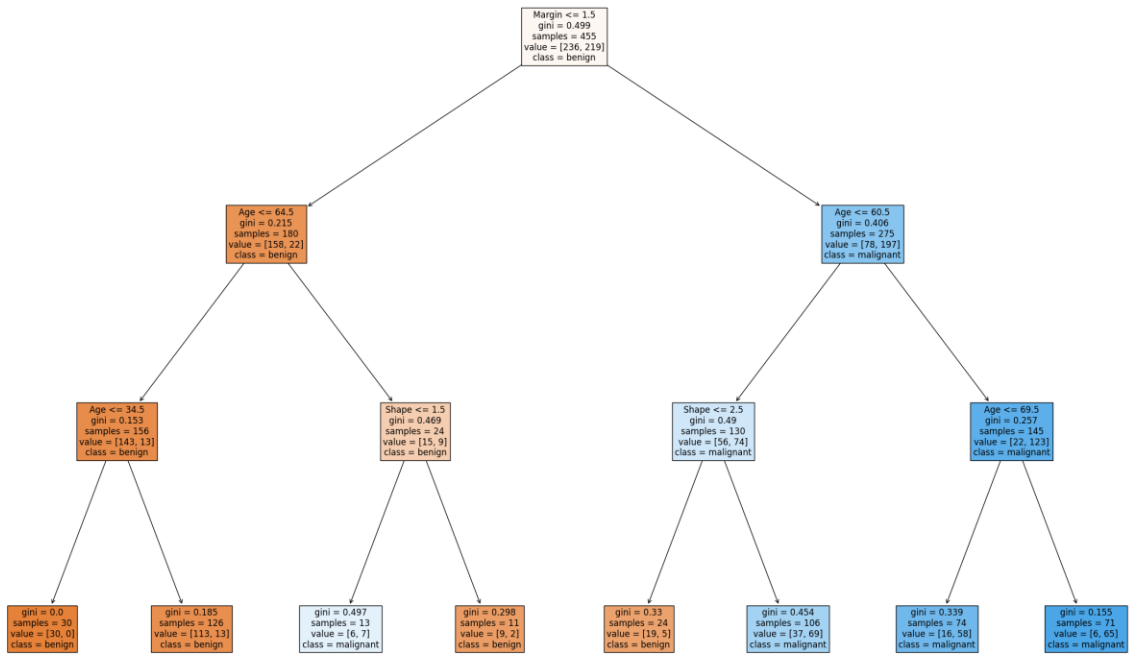
b. Decision Tree Grid Search

Optimal parameter found in grid search: {'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 4}

Training accuracy: 81.31868131868132%

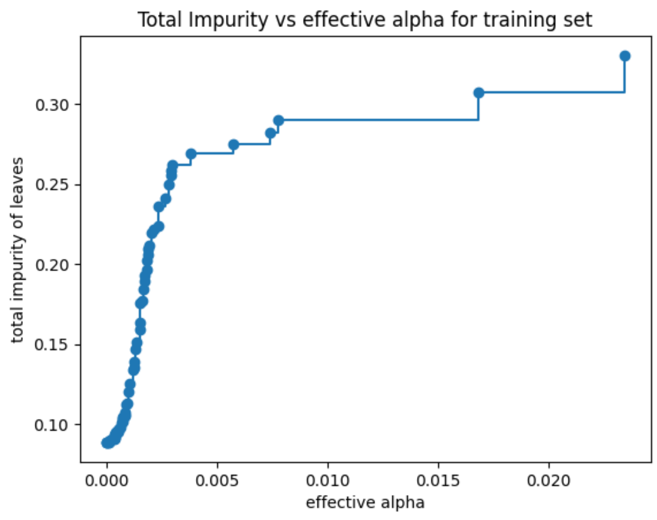
Test accuracy: 75.49407114624506%

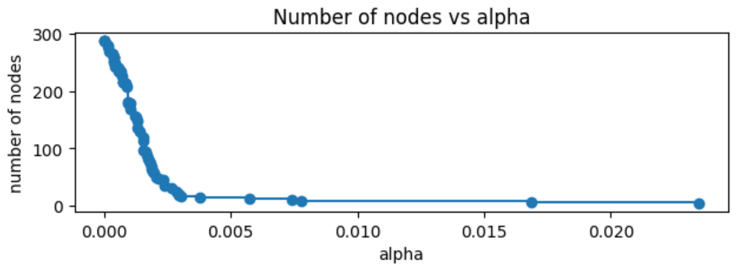
Validation accuracy: 87.60330578512396%

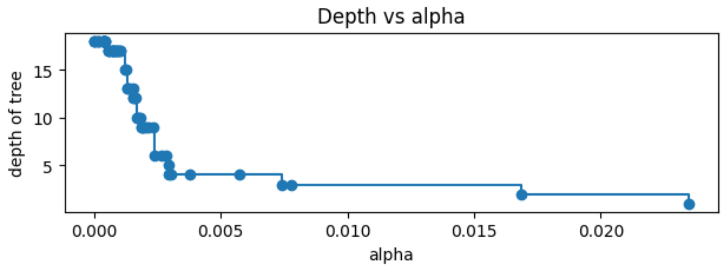


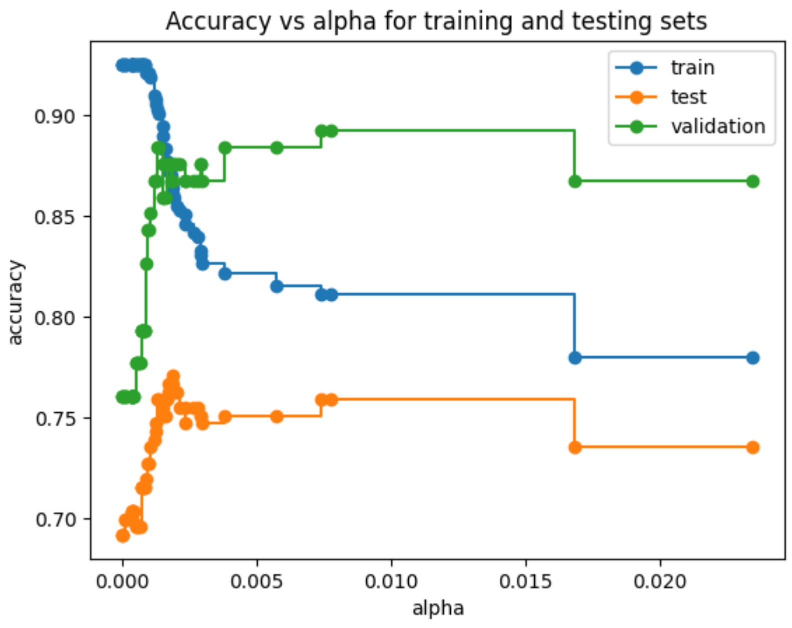
Observation:
With optimal parameters test and validation accuracy improved. But train data accuracy is decreased which indicate that in previous part model was overfitting.

c. **Decision Tree Post Pruning (Cost Complexity Pruning):**





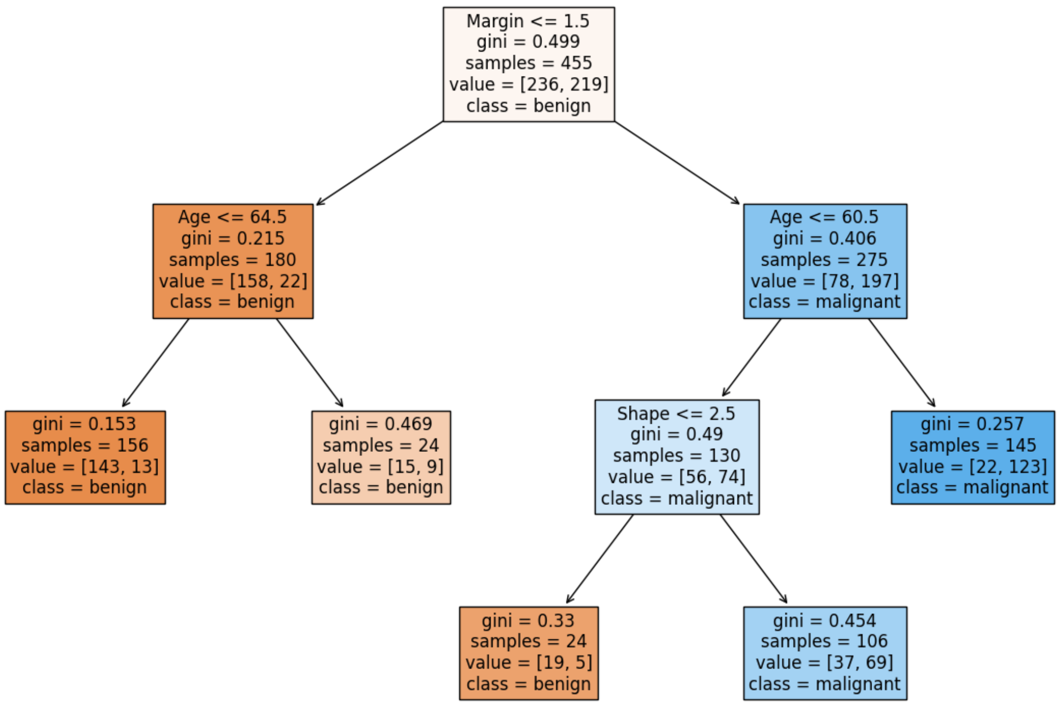




Observations:
Effective alpha increases total impurities of leaves increases.
Effective alpha increases no. of nodes decreases.
Effective alpha increases depth of the tree decreases.
Accuracy is maximum only at optimal value of effective alpha.

Optimal value of alpha: 0.007387057387057339

Train accuracy with respect to the best tree: 81.0989010989011%
Test accuracy with respect to the best tree: 75.8893280632411%
Validation accuracy with respect to the best tree: 89.25619834710744%



Comparison with part (a) and part (b):
Tree in this has lesser depth compare to part (a) and (b) due to cost complexity pruning. And accuracy is also improved because cost complexity pruning avoid overfitting of data.

d. Random Forests:
Out-of-bag accuracy (as explained in the class) to tune to the optimal values for these parameters.

Optimal set of parameters: {'max_features': 1, 'min_samples_split': 7, 'n_estimators': 51, 'oob_score': True}

Train accuracy: 87.25274725274725%
Out-of-bag accuracy: 78.46153846153847%
Test accuracy: 76.6798418972332%
Validation accuracy: 84.29752066115702%

e. Missing Data Imputation:
Simple imputer from sklearn library used to replace the missing value with median of that data group.
Results of Part (a) to (d) after imputation:
Part (a)
Train accuracy: 91.80633147113594%
Test accuracy: 72.91666666666666%
Validation accuracy: 73.33333333333333%

Part (b)
Train accuracy: 81.56424581005587%
Test accuracy: 80.90277777777778%
Validation accuracy: 87.40740740740741%

Part (c)
Train accuracy with respect to the best tree: 8212290502793296%
Test accuracy with respect to the best tree: 79.86111111111112%
Validation accuracy with respect to the best tree: 86.66666666666667%

Part (D)
Train accuracy: 82.30912476722533%
Out-of-bag accuracy: 79.70231914157149%
Test accuracy: 80.55555555555556%
Validation accuracy: 85.92592592592593%

Observation:
Test set accuracy is improved in every part after imputation. Simply because now we have trained our model with more information by imputing. Although in some cases validation accuracy is less compare to other parts maybe because imputation function (median) may not be the best choice for validation data set.

f. **Gradient Boosted Trees:**
XGBoost (Extreme Gradient Boosting) is used with the goal to construct a model with less bias, and better predictive performance.

Optimal set of parameters: {'max_depth': 4, 'n_estimators': 20, 'subsample': 0.1}

Train accuracy: 79.51582867783985%
Test accuracy: 75.69444444444444%
Validation accuracy: 85.92592592592593%

Observations:
XGBoost provides better accuracy compare part (a) and even missing data imputation part (a) because XGBoost uses the approach of weak learner which gives less bias and more variance which is very helpful to avoid overfitting.

Dataset 2:

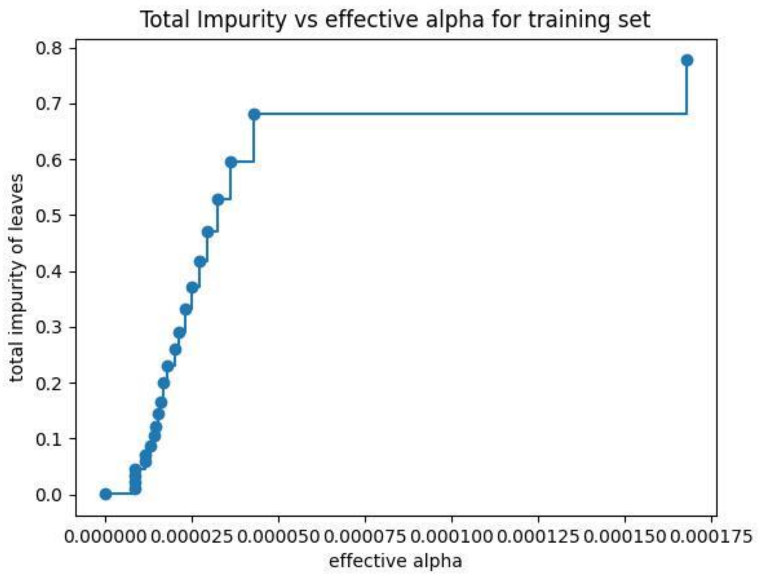
a. **Decision Tree Construction:**
Train accuracy: 99.94863074361428%
Test accuracy: 57.49730312837109%
Validation accuracy: 57.75899481287069%

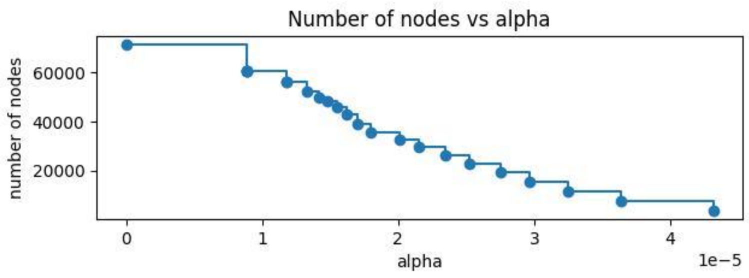
b. **Decision Tree Grid Search:**

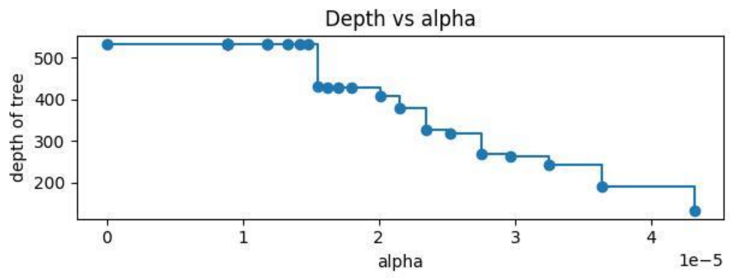
Optimal parameter found in grid search: {'max_depth': 50, 'min_samples_leaf': 1, 'min_samples_split': 2}

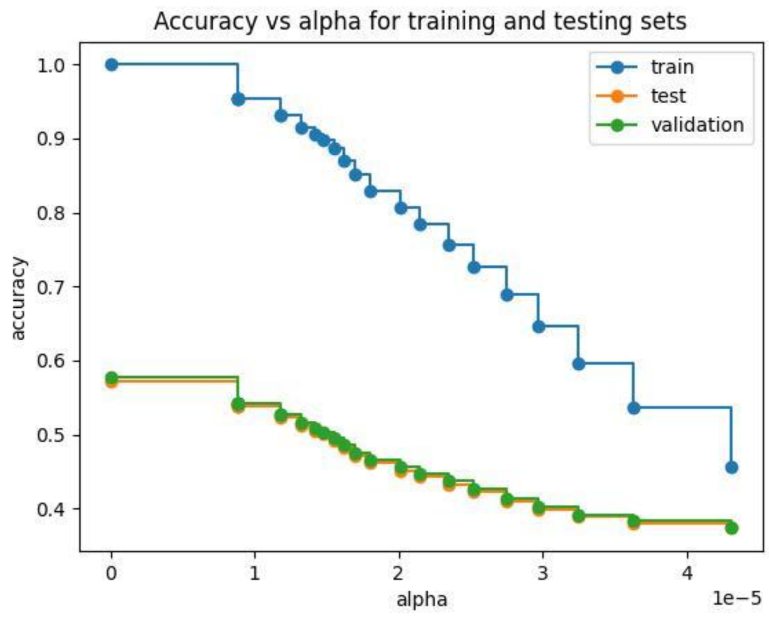
Train accuracy: 74.84943493817976%
Test accuracy: 48.0098947290109%
Validation accuracy: 48.221703279671%

c. Decision Tree Post Pruning:









Optimal value of alpha: 0

Train accuracy with respect to the best tree: 99.94863074361428%
Test accuracy with respect to the best tree: 57.25923446043968%
Validation accuracy with respect to the best tree: 57.74866188596582%

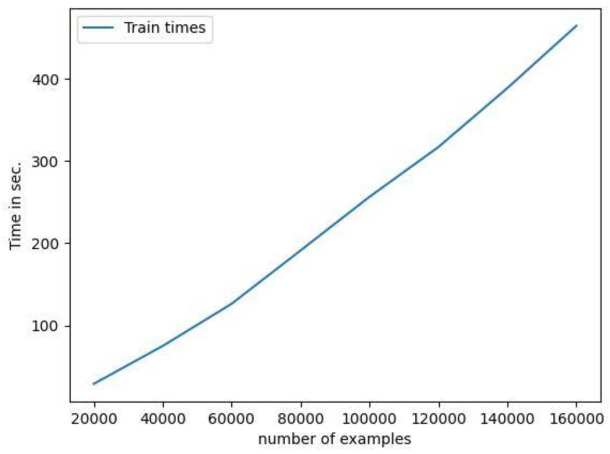
d. **Random Forests:**
max_features=0.4, min_samples_split=10, n_estimators=100
Train accuracy: 0.9964395791263684
Test accuracy: 0.6385448052672693
Validation accuracy: 0.6413441071317861

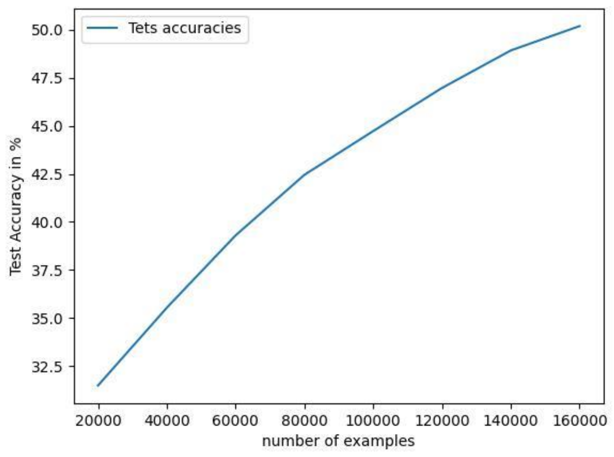
e. **XGBoost:**
At default parameters
Train accuracy: 83.25096538774932 %
Test accuracy: 55.241230517427375 %
Validation accuracy: 55.252226745748004 %

f. **LightGBM:**
At default parameters
Train accuracy: 53.55333545895774%
Test accuracy: 43.98132648885913%
Validation accuracy: 44.206327884436547%

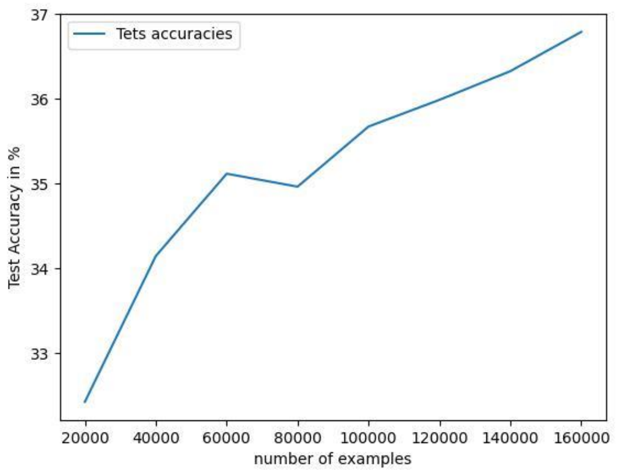
g. **Training with Varying amount of data:**
n ∈ {20k, 40k, 60k, 80k, 100k, 120k, 140k, 160k}

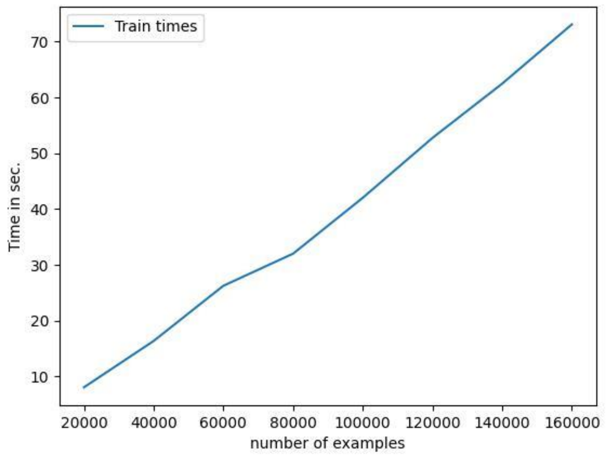
Part (a):


Train times
Time in sec.
400
300
200
100
20000
40000
60000
80000
100000
120000
140000
160000
number of examples

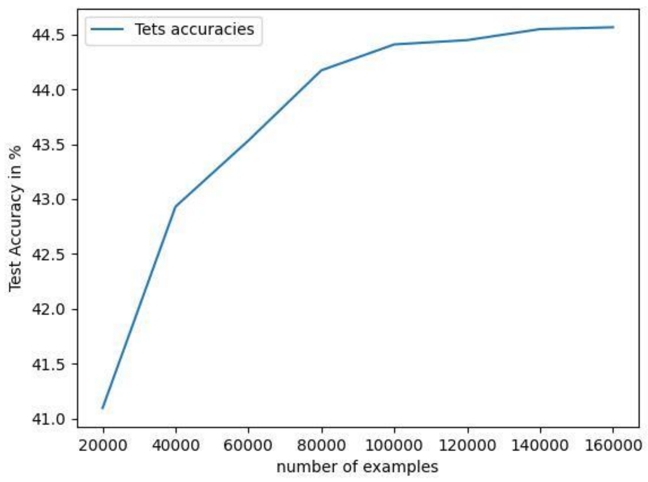

Tets accuracies
Test Accuracy in %
50.0
47.5
45.0
42.5
40.0
37.5
35.0
32.5
20000
40000
60000
80000
100000
120000
140000
160000
number of examples

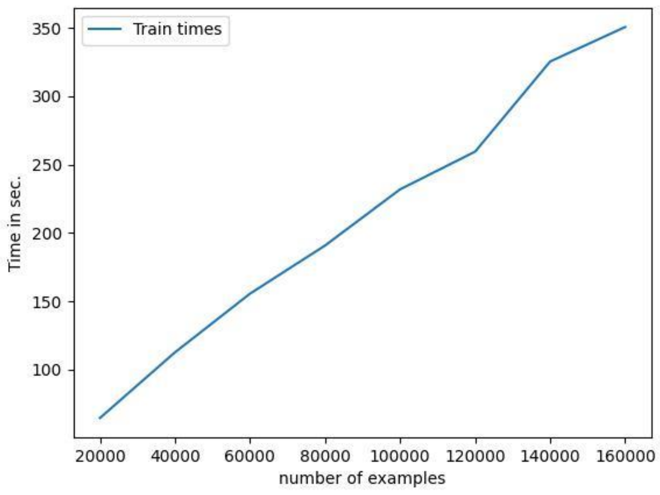
Part (b):


Tets accuracies
Test Accuracy in %
number of examples
37
36
35
34
33
20000
40000
60000
80000
100000
120000
140000
160000


Train times
Time in sec.
number of examples
70
60
50
40
30
20
10
20000
40000
60000
80000
100000
120000
140000
160000

Part (f):


Tets accuracies
Test Accuracy in %
number of examples
44.5
44.0
43.5
43.0
42.5
42.0
41.5
41.0
20000
40000
60000
80000
100000
120000
140000
160000


Train times
Time in sec.
350
300
250
200
150
100
20000
40000
60000
80000
100000
120000
140000
160000
number of examples

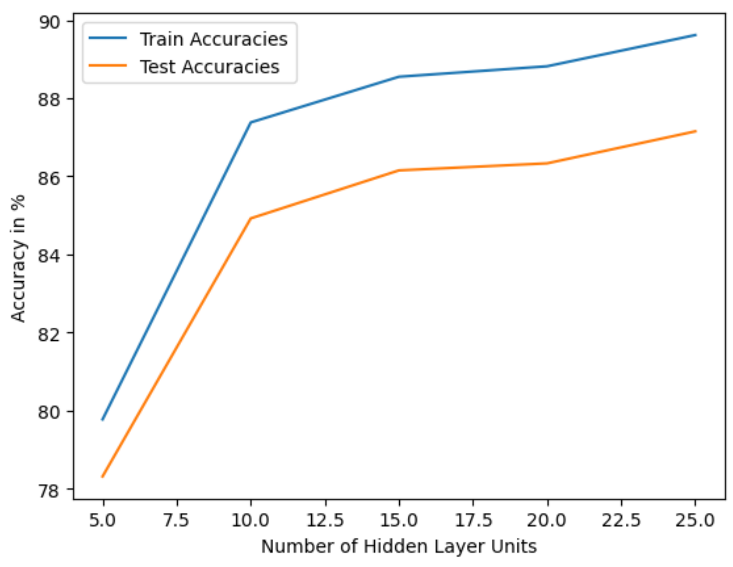
2. Neural Networks:

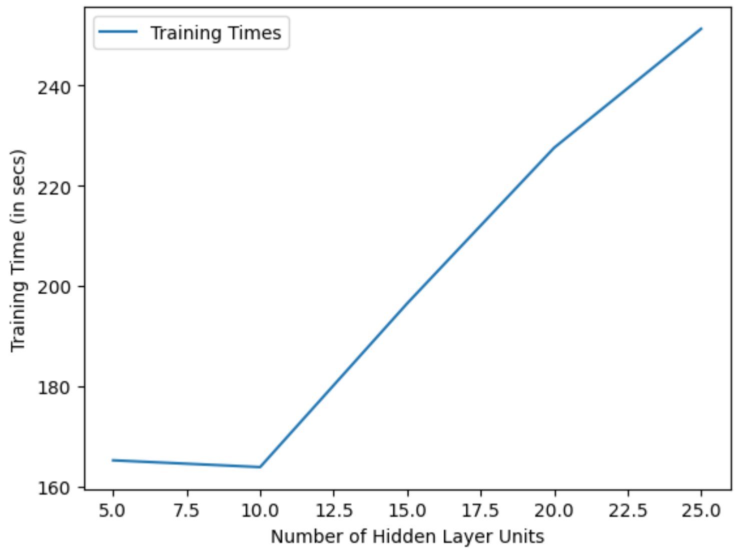
a. Neural Network algorithm has been implemented from scratch for Multi-class Classification. In this outputs are converted into one hot encoded form. We have used Mini-Batch Stochastic Gradient Descent to train the network. A fully connected architecture is assumed while implemented the algorithm.
The implementation is generic for following parameters:
(a) Mini-Batch Size
(b) Number of features/attributes
(c) Hidden Layer Architecture : List of numbers denoting the number of perceptrons in the corresponding hidden layer.
(d) Number of target classes

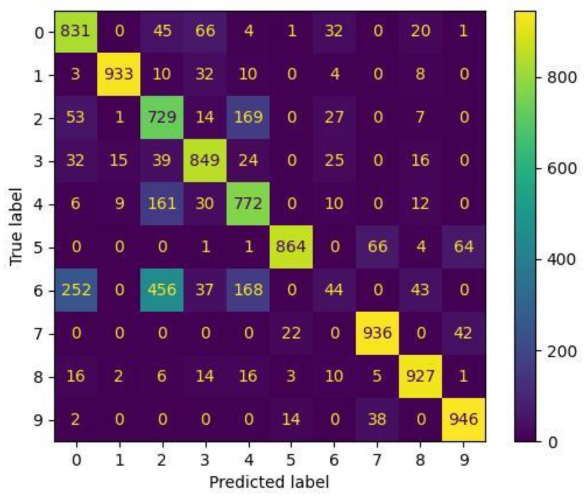
b. To train the network following parameter are fixed.
Number of Hidden Layer = 1
$\eta = 0.1$
Hidden Layer Units = [5, 10, 15, 20, 25]
Mini Batch Sizer = 100

Convergence Criteria: (current average cost – previous average cost) < 5e-5 and no. of epochs > 50

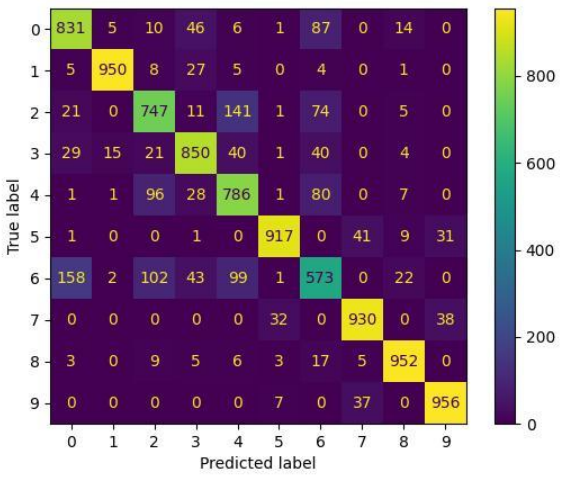
```
Train accuracies....
Train accuracy for hidden layer of 5 units is: 79.77166666666666 %
Train accuracy for hidden layer of 10 units is: 87.37833333333333 %
Train accuracy for hidden layer of 15 units is: 88.54833333333333 %
Train accuracy for hidden layer of 20 units is: 88.81666666666666 %
Train accuracy for hidden layer of 25 units is: 89.61666666666666 %

Test accuracies....
Test accuracy for hidden layer of 5 units is: 78.31 %
Test accuracy for hidden layer of 10 units is: 84.92 %
Test accuracy for hidden layer of 15 units is: 86.15 %
Test accuracy for hidden layer of 20 units is: 86.33 %
Test accuracy for hidden layer of 25 units is: 87.15 %

Training Time....
Training time for hidden layer of 5 units is: 165.18345522880554 sec
Training time for hidden layer of 10 units is: 163.8339102268219 sec
Training time for hidden layer of 15 units is: 196.49265480041504 sec
Training time for hidden layer of 20 units is: 227.5731236934662 sec
Training time for hidden layer of 25 units is: 251.32813835144043 sec
```



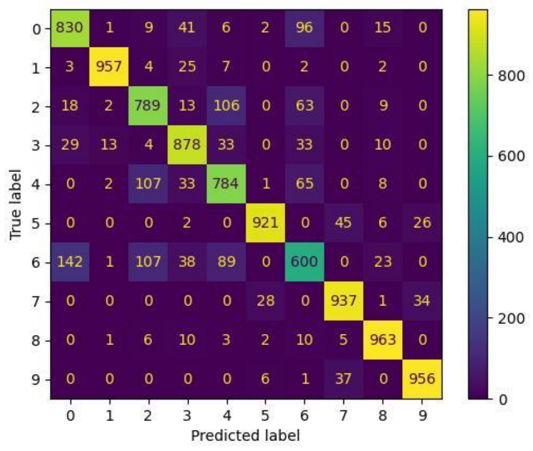




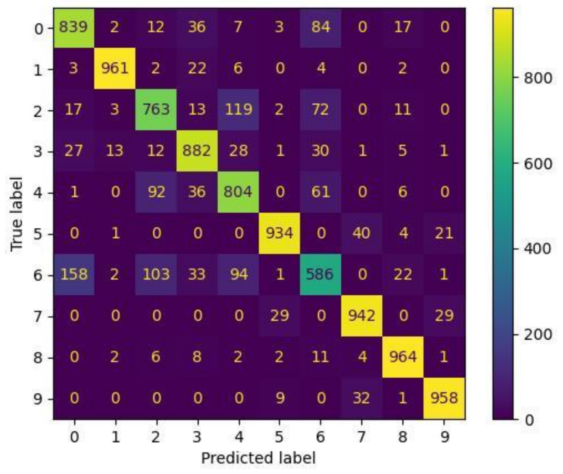
Confusion Matrix for hidden layer of 5 units



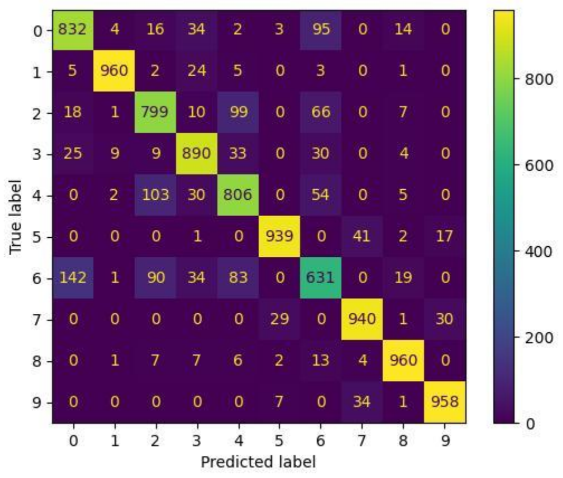
Confusion Matrix for hidden layer of 10 units



Confusion Matrix for hidden layer of 15 units



Confusion Matrix for hidden layer of 20 units



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	832	4	16	34	2	3	95	0	14	0
1	5	960	2	24	5	0	3	0	1	0
2	18	1	799	10	99	0	66	0	7	0
3	25	9	9	890	33	0	30	0	4	0
4	0	2	103	30	806	0	54	0	5	0
5	0	0	0	1	0	939	0	41	2	17
6	142	1	90	34	83	0	631	0	19	0
7	0	0	0	0	0	29	0	940	1	30
8	0	1	7	7	6	2	13	4	960	0
9	0	0	0	0	0	7	0	34	1	958

Confusion Matrix for hidden layer of 25 units

Conclusion:
For large hidden layer units, time taken to train the model is increasing substantially as we can see it from the graphs above.
Accuracy increases as the no. of hidden layer units increases.

c. Using Adaptive Learning Rate

Learning rate (η) would get updated after every epoch

$$\eta_e = \eta_0/\sqrt{e}$$

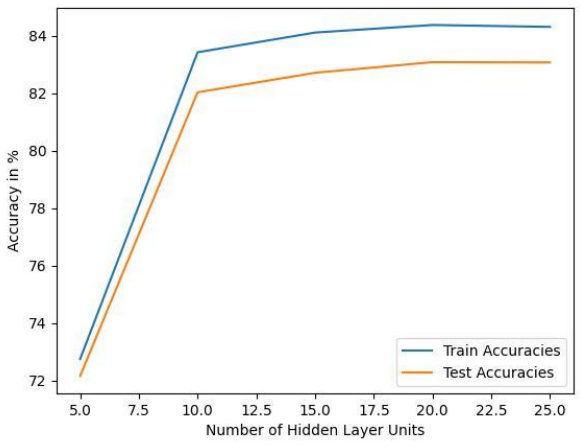
Where $\eta_0 = 0.1$ and e is the current epoch number.

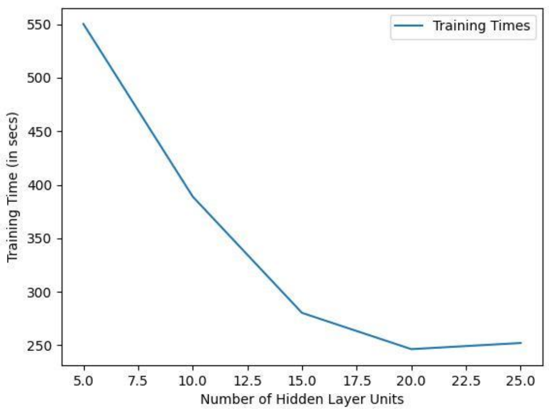
Convergence criteria: (current average cost – previous average cost) < 5e-5 and no. of epochs > 50

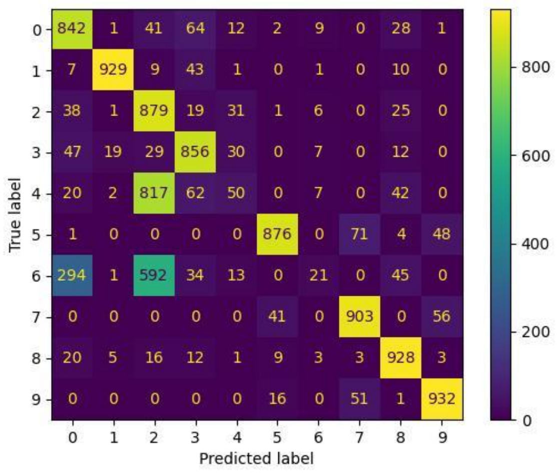
Train accuracies....
Train accuracy for hidden layer of 5 units is: 72.75%
Train accuracy for hidden layer of 10 units is: 83.42333333333333%
Train accuracy for hidden layer of 15 units is: 84.11%
Train accuracy for hidden layer of 20 units is: 84.37333333333333%
Train accuracy for hidden layer of 25 units is: 84.30666666666666%

Test accuracies....
Test accuracy for hidden layer of 5 units is: 72.16%
Test accuracy for hidden layer of 10 units is: 82.03%
Test accuracy for hidden layer of 15 units is: 82.71%
Test accuracy for hidden layer of 20 units is: 83.08%
Test accuracy for hidden layer of 25 units is: 83.07%

Training Time....
Training time for hidden layer of 5 units is: 550.1844565868378 sec
Training time for hidden layer of 10 units is: 388.9296541213989 sec
Training time for hidden layer of 15 units is: 280.2832624912262 sec
Training time for hidden layer of 20 units is: 246.40840601921082 sec
Training time for hidden layer of 25 units is: 252.13452792167664 sec

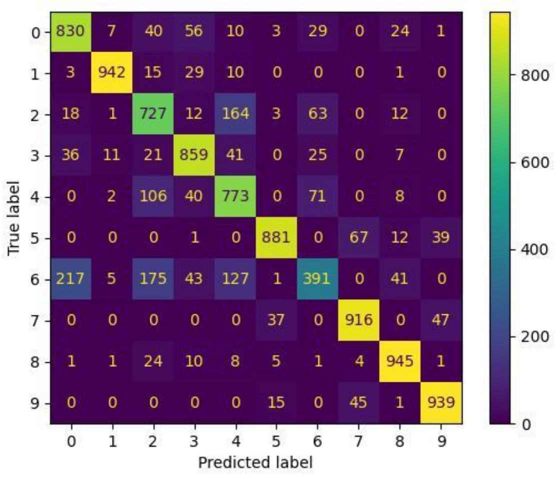






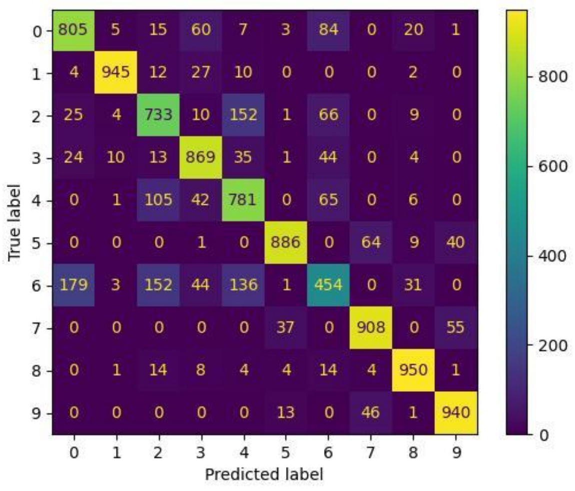
	0	1	2	3	4	5	6	7	8	9
0	842	1	41	64	12	2	9	0	28	1
1	7	929	9	43	1	0	1	0	10	0
2	38	1	879	19	31	1	6	0	25	0
3	47	19	29	856	30	0	7	0	12	0
4	20	2	817	62	50	0	7	0	42	0
5	1	0	0	0	0	876	0	71	4	48
6	294	1	592	34	13	0	21	0	45	0
7	0	0	0	0	0	41	0	903	0	56
8	20	5	16	12	1	9	3	3	928	3
9	0	0	0	0	0	16	0	51	1	932

Confusion Matrix for hidden layer of 5 units



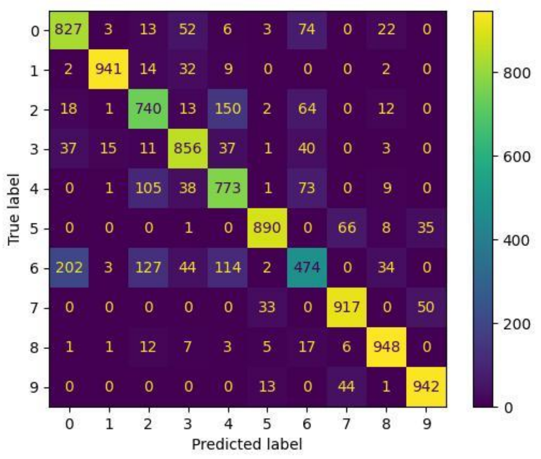
True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	830	7	40	56	10	3	29	0	24	1
1	3	942	15	29	10	0	0	0	1	0
2	18	1	727	12	164	3	63	0	12	0
3	36	11	21	859	41	0	25	0	7	0
4	0	2	106	40	773	0	71	0	8	0
5	0	0	0	1	0	881	0	67	12	39
6	217	5	175	43	127	1	391	0	41	0
7	0	0	0	0	0	37	0	916	0	47
8	1	1	24	10	8	5	1	4	945	1
9	0	0	0	0	0	15	0	45	1	939

Confusion Matrix for hidden layer of 10 units



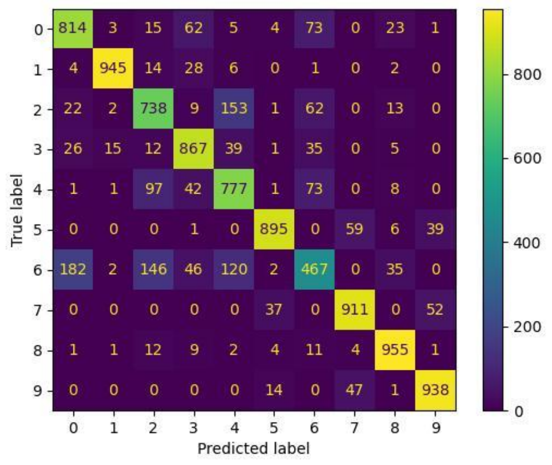
True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	805	5	15	60	7	3	84	0	20	1
1	4	945	12	27	10	0	0	0	2	0
2	25	4	733	10	152	1	66	0	9	0
3	24	10	13	869	35	1	44	0	4	0
4	0	1	105	42	781	0	65	0	6	0
5	0	0	0	1	0	886	0	64	9	40
6	179	3	152	44	136	1	454	0	31	0
7	0	0	0	0	0	37	0	908	0	55
8	0	1	14	8	4	4	14	4	950	1
9	0	0	0	0	0	13	0	46	1	940

Confusion Matrix for hidden layer of 15 units



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	827	3	13	52	6	3	74	0	22	0
1	2	941	14	32	9	0	0	0	2	0
2	18	1	740	13	150	2	64	0	12	0
3	37	15	11	856	37	1	40	0	3	0
4	0	1	105	38	773	1	73	0	9	0
5	0	0	0	1	0	890	0	66	8	35
6	202	3	127	44	114	2	474	0	34	0
7	0	0	0	0	0	33	0	917	0	50
8	1	1	12	7	3	5	17	6	948	0
9	0	0	0	0	0	13	0	44	1	942

Confusion Matrix for hidden layer of 20 units



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	814	3	15	62	5	4	73	0	23	1
1	4	945	14	28	6	0	1	0	2	0
2	22	2	738	9	153	1	62	0	13	0
3	26	15	12	867	39	1	35	0	5	0
4	1	1	97	42	777	1	73	0	8	0
5	0	0	0	1	0	895	0	59	6	39
6	182	2	146	46	120	2	467	0	35	0
7	0	0	0	0	0	37	0	911	0	52
8	1	1	12	9	2	4	11	4	955	1
9	0	0	0	0	0	14	0	47	1	938

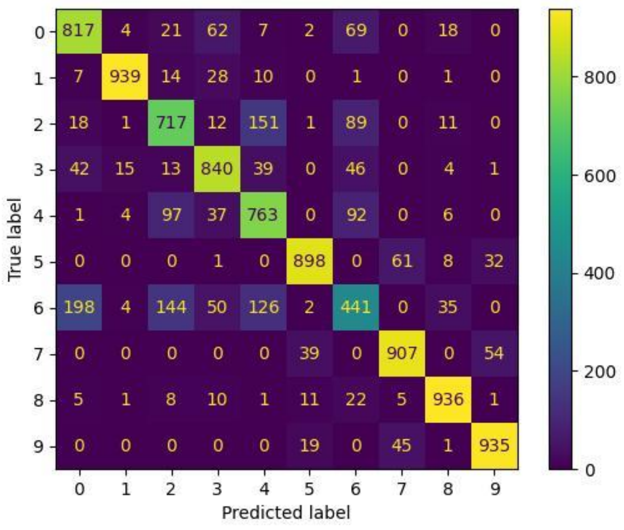
Confusion Matrix for hidden layer of 25 units

Observations:
Network trained using adaptive learning rate takes more time than what it had taken in non-adaptive learning.
Accuracies are slightly less than what we had earlier for each hidden layer units.
Adaptive learning rate has taken more time than non-adaptive learning rate as learning rate keeps on decreasing with each epoch and that is leading to its slow convergence.

d. ReLU and sigmoid activation unit comparison:
A network is implemented with 2 hidden layers with 100 units each.

Results with sigmoid Activation Units

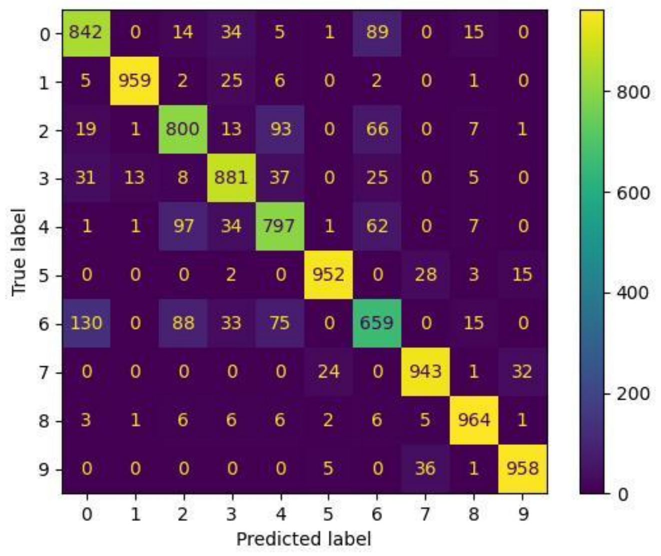
Train accuracy for hidden layer of [100, 100] units is: 83.45833333333333%
Test accuracy for hidden layer of [100, 100] units is: 81.93%
Training time for hidden layer of [100, 100] units is: 872.1873600482941 sec



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	817	4	21	62	7	2	69	0	18	0
1	7	939	14	28	10	0	1	0	1	0
2	18	1	717	12	151	1	89	0	11	0
3	42	15	13	840	39	0	46	0	4	1
4	1	4	97	37	763	0	92	0	6	0
5	0	0	0	1	0	898	0	61	8	32
6	198	4	144	50	126	2	441	0	35	0
7	0	0	0	0	0	39	0	907	0	54
8	5	1	8	10	1	11	22	5	936	1
9	0	0	0	0	0	19	0	45	1	935

Result with ReLu Activation Units

Train accuracy for hidden layer of [100, 100] units is: 90.275%
Test accuracy for hidden layer of [100, 100] units is: 87.55%
Training time for hidden layer of [100, 100] units is: 1426.159077167511 sec



Observations:
ReLu activation function for hidden layers gives better results.
Multiple hidden layers gives better accuracies compare to single hidden layer in part (b).

e. Experiment by increasing the number of hidden layers:

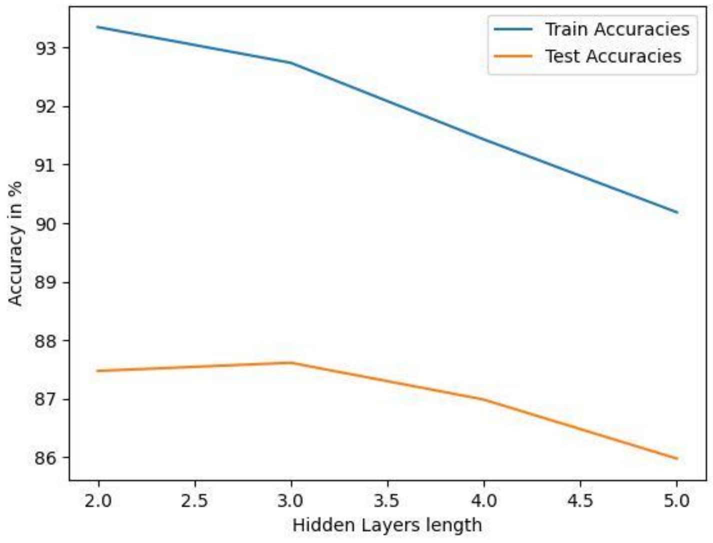
Hidden layers in network from 2 to 5. Keeping the number of hidden units per layer to be 50.

Accuracies with ReLu function for Hidden Layers:

Train accuracy for hidden layer of [50, 50] units is: 93.35333333333334 %
Train accuracy for hidden layer of [50, 50, 50] units is: 92.74166666666666 %
Train accuracy for hidden layer of [50, 50, 50, 50] units is: 91.43166666666667 %
Train accuracy for hidden layer of [50, 50, 50, 50, 50] units is: 90.185 %

Test accuracy for hidden layer of [50, 50] units is: 87.47 %
Test accuracy for hidden layer of [50, 50, 50] units is: 87.61 %
Test accuracy for hidden layer of [50, 50, 50, 50] units is: 86.98 %
Test accuracy for hidden layer of [50, 50, 50, 50, 50] units is: 85.97 %

Training time for hidden layer of [50, 50] units is: 489.35870456695557 sec
Training time for hidden layer of [50, 50, 50] units is: 412.4283878803253 sec
Training time for hidden layer of [50, 50, 50, 50] units is: 334.13640904426575 sec
Training time for hidden layer of [50, 50, 50, 50, 50] units is: 356.448077917099 sec

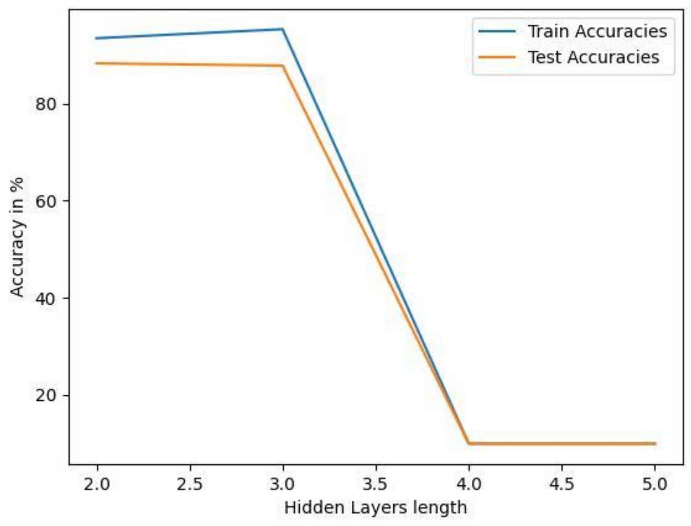


Accuracies with Sigmoid function for Hidden Layers:

Train accuracy for hidden layer of [50, 50] units is: 93.4 %
Train accuracy for hidden layer of [50, 50, 50] units is: 95.26 %
Train accuracy for hidden layer of [50, 50, 50, 50] units is: 10.0 %
Train accuracy for hidden layer of [50, 50, 50, 50, 50] units is: 10.0 %

Test accuracy for hidden layer of [50, 50] units is: 88.23 %
Test accuracy for hidden layer of [50, 50, 50] units is: 87.77000000000001 %
Test accuracy for hidden layer of [50, 50, 50, 50] units is: 10.0 %
Test accuracy for hidden layer of [50, 50, 50, 50, 50] units is: 10.0 %

Training time for hidden layer of [50, 50] units is: 663.4633610248566 sec
Training time for hidden layer of [50, 50, 50] units is: 986.2991437911987 sec
Training time for hidden layer of [50, 50, 50, 50] units is: 74.34880018234253 sec
Training time for hidden layer of [50, 50, 50, 50, 50] units is: 83.22205805778503 sec



Observations:

ReLu activation function more suitable and gives better results for multiple hidden layer neural network.
Sigmoid function in multiple hidden layers network suffers from vanishing gradient problem.
The gradient for the sigmoid function will saturate and when using the chain rule, it will shrink. By contrast, the derivative for ReLU is always 1 or 0.
Because of this as the no. of hidden layer increases we can see the drastic reduction in accuracies of model.

f. Cross-Entropy loss:

$$\mathbf{h} = \mathbf{w}^T\mathbf{X}$$

$$\text{Logistic regression: } \mathbf{z} = \sigma(\mathbf{h}) = \frac{1}{1+e^{-\mathbf{h}}}$$

$$\text{Cross-entropy loss: } J(\mathbf{w}) = -(\mathbf{y}log(\mathbf{z}) + (1-\mathbf{y})log(1-\mathbf{z}))$$

$$\text{Use chain rule: } \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial J(\mathbf{w})}{\partial \mathbf{z}}\frac{\partial \mathbf{z}}{\partial \mathbf{h}}\frac{\partial \mathbf{h}}{\partial \mathbf{w}}$$

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{z}} = -(\frac{\mathbf{y}}{\mathbf{z}} - \frac{1-\mathbf{y}}{1-\mathbf{z}}) = \frac{\mathbf{z}-\mathbf{y}}{\mathbf{z}(1-\mathbf{z})}$$

$$\frac{\partial \mathbf{z}}{\partial \mathbf{h}} = \mathbf{z}(1-\mathbf{z})$$

$$\frac{\partial \mathbf{h}}{\partial \mathbf{w}} = \mathbf{X}$$

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{X}^T(\mathbf{z}-\mathbf{y})$$

$$\text{Gradient descent: } \mathbf{w} = \mathbf{w} - \alpha\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$

Train accuracy for hidden layer of [50, 50, 50] units is: 92.43666666666667 %

Test accuracy for hidden layer of [50, 50, 50] units is: 87.83 %

Training time for hidden layer of [50, 50, 50] units is: 334.8329563140869 sec

Total no. of epochs for hidden layer of [50, 50, 50] units is: 85

g. MLPClassifier:

Parameters for MLPClassifier:
hidden_layer_sizes=(50,50,50)
activation="relu"
solver='sgd'
max_iter=85
learning_rate_init=0.1

Train accuracy: 93.49833333333333 %
Test accuracy: 87.42 %
Training time: 215.6363410949707 sec

Observation:
Accuracies in this part and part (f) are almost similar as we kept all parameters same. Which gives validation to my neural network model that I've build from scratch.

Libraries used for this assignment:

pandas

numpy

matplotlib

sklearn

time

sys

scipy

xgboost

lightgbm