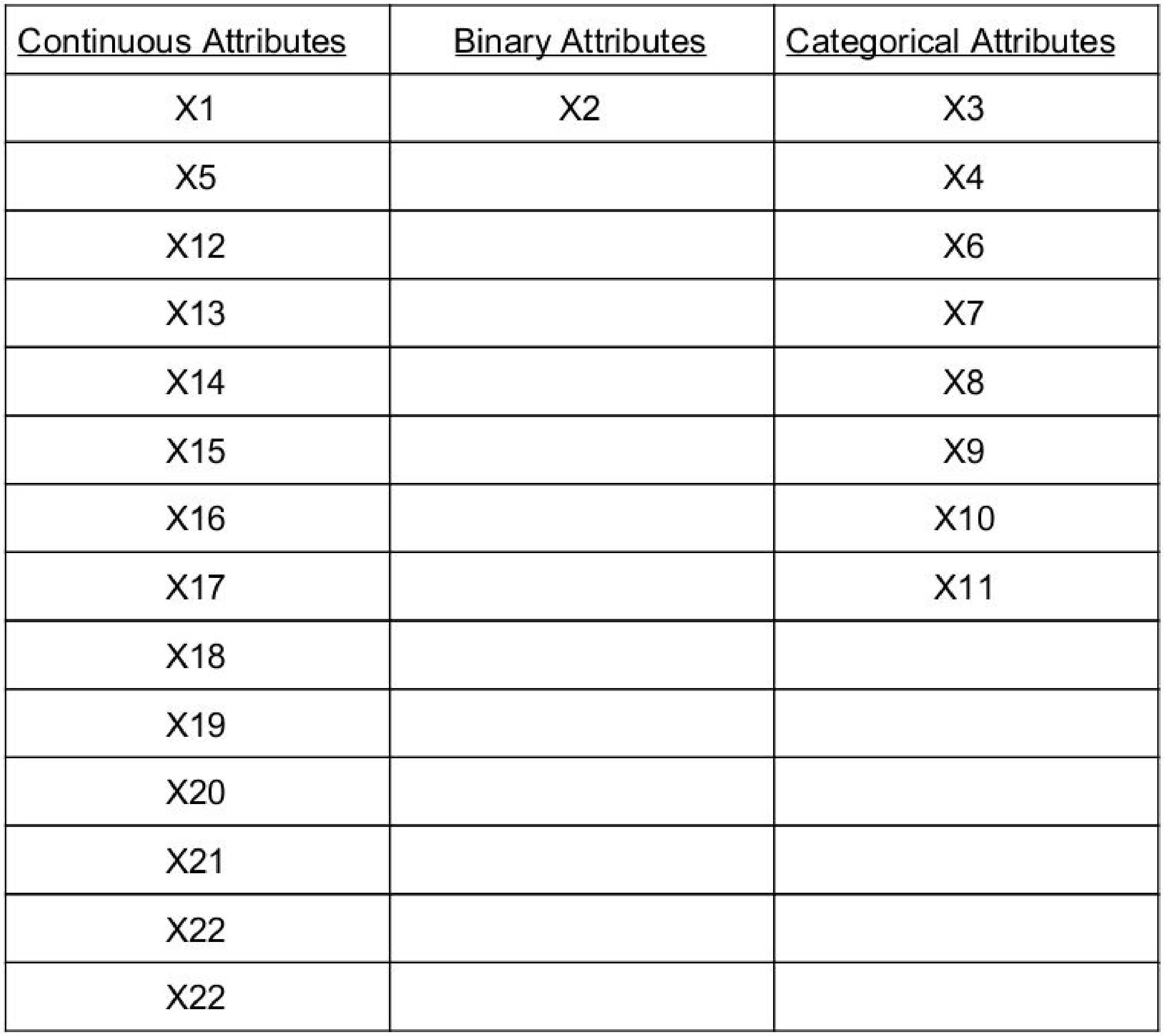
MACHINE LEARNING
COL-774

Assignment 3
Report file

Submitted by:
Anshu Bansal
2018MCS2142

Question1.(Decision Tree)



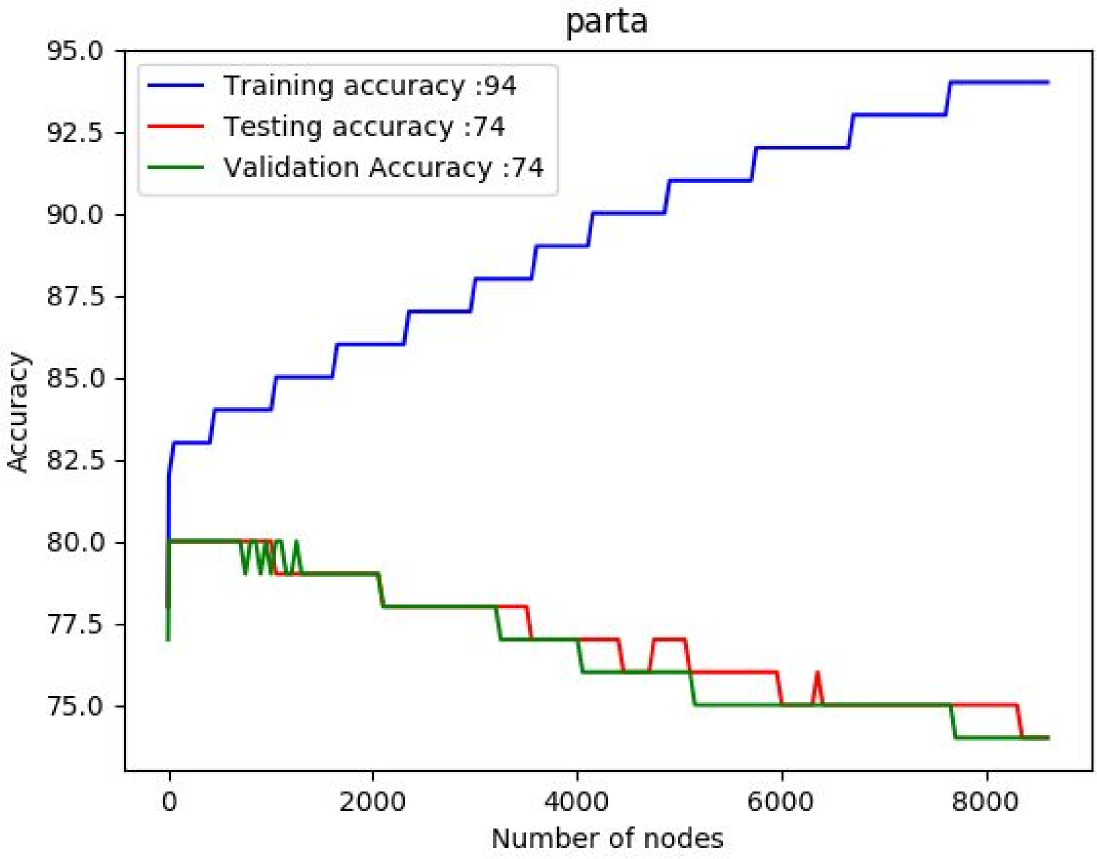
Continuous Attributes	Binary Attributes	Categorical Attributes
X1	X2	X3
X5		X4
X12		X6
X13		X7
X14		X8
X15		X9
X16		X10
X17		X11
X18		
X19		
X20		
X21		
X22		
X22		

Table 1: Different types of attributes in the data set

For continuous attributes, I converted them to binary based on whether the value is greater than the median threshold or not.
For binary attr, I did boolean(two-way) split.
For categorical, I did the multi-way split.

a. Accuracies against number of nodes in the tree as tree grows

BFS Growth is used
('accuracy Training Data', 94.65555555555555)
('accuracy Testing Data', 74.98333333333333)
('accuracy Validation Data', 74.31666666666666)
Number of nodes - 9257



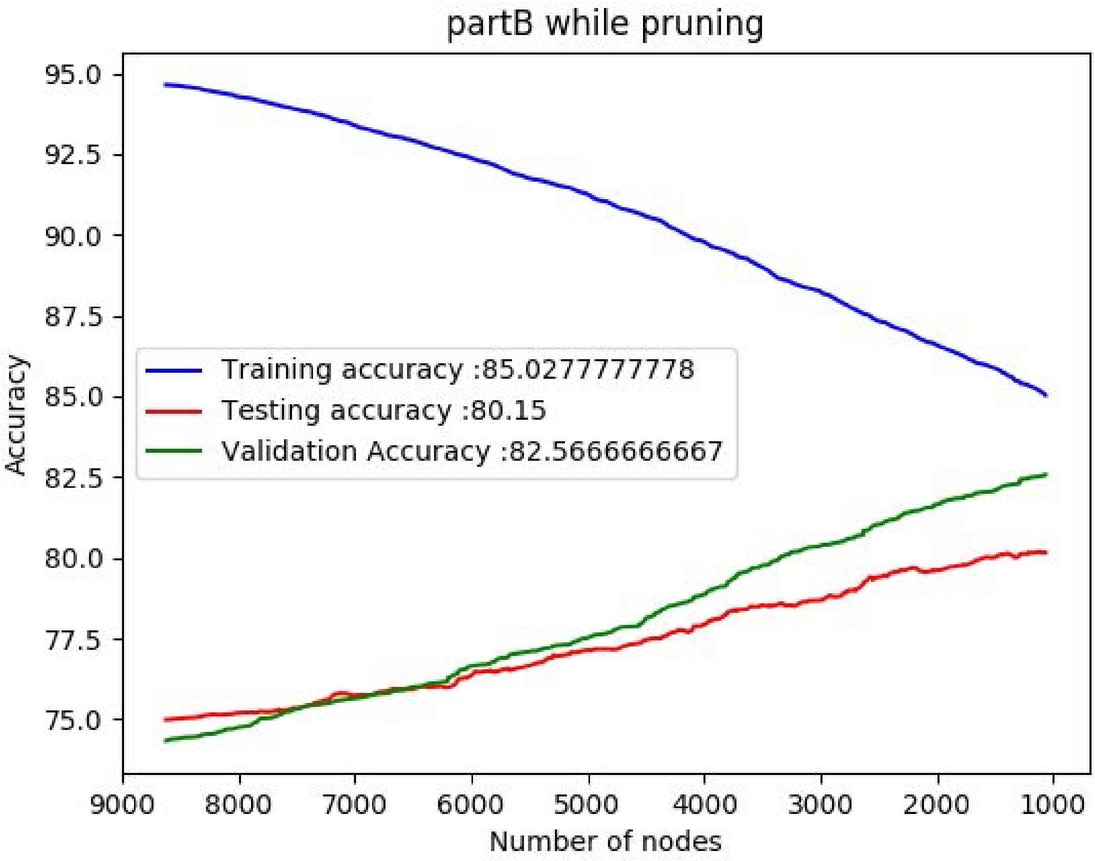
Observations:

Decision Tree with a single node predicts the majority class giving the accuracy of ~78% . As number of nodes increases, Training accuracy increases while Testing and Validation accuracies decreases i.e. overfitting happens.

b. Post pruning based on validation set

BFS growth

```
('accuracy Training Data', 85.0277777778)
('accuracy Testing Data', 80.15)
('accuracy Validation Data', 82.5666667)
Number of Nodes :  993
```

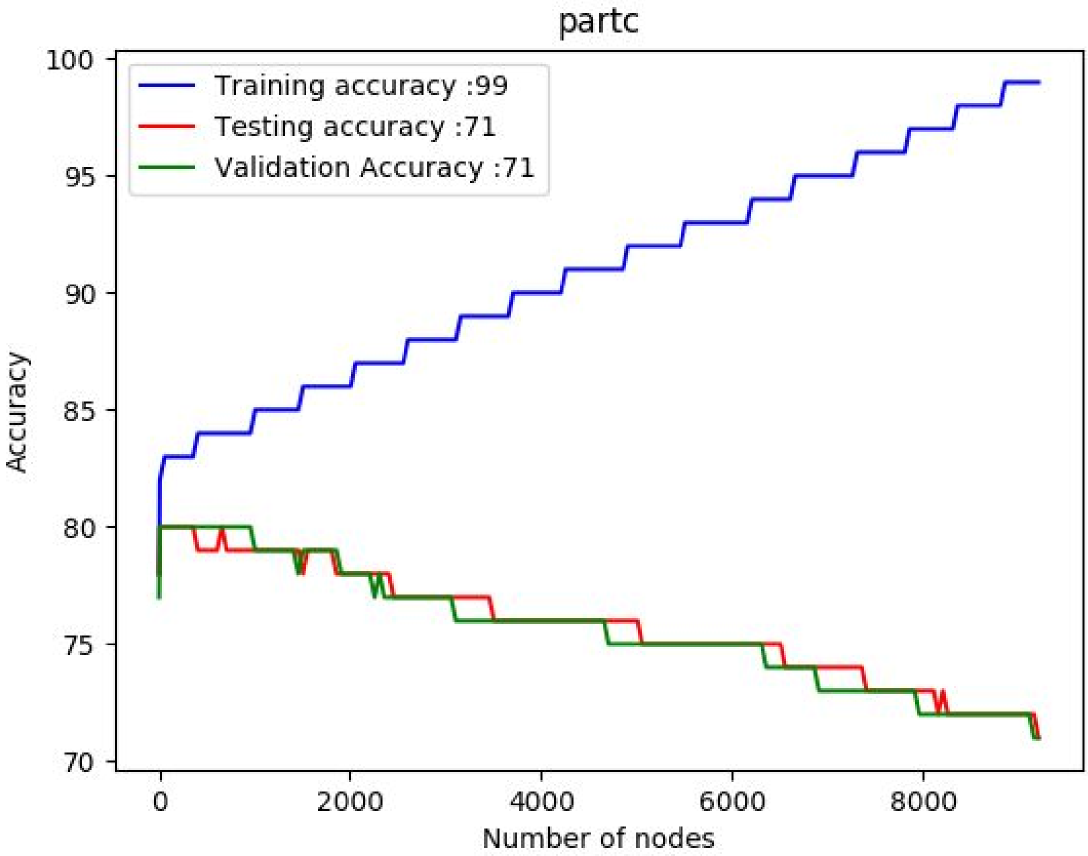


Observations:

Pruning decreases number of nodes from 9257 to 993 improving validation accuracy from 74% to 82% and testing accuracy from 75% to 80%. And hence helps in generalizing well while reducing overfitting.

c. Using medians dynamically (without pruning)

```
Number of nodes : 9849
('accuracy Training Data', 99.9)
('accuracy Testing Data', 71.98333333333333)
('accuracy Validation Data', 71.91666666666667)
```



Numerical Attributes split multiple times in a branch:

```
(1, [120000.0, 50000.0, 70000.0, 80000.0, 95000.0, 110000.0])
(5, [36.0, 41.5, 48.0, 50.0, 51.0, 54.0])
(12, [107948.0, 34711.5, 59307.0, 46385.0])
(13, [1053.0, 2380.0, 3468.0, 3816.0])
(14, [46896.5, 34027.0, 44291.0, 45766.0, 46106.0])
(15, [40385.0, 48097.0, 48635.0])
(16, [39369.0, 45794.0, 41296.0])
```

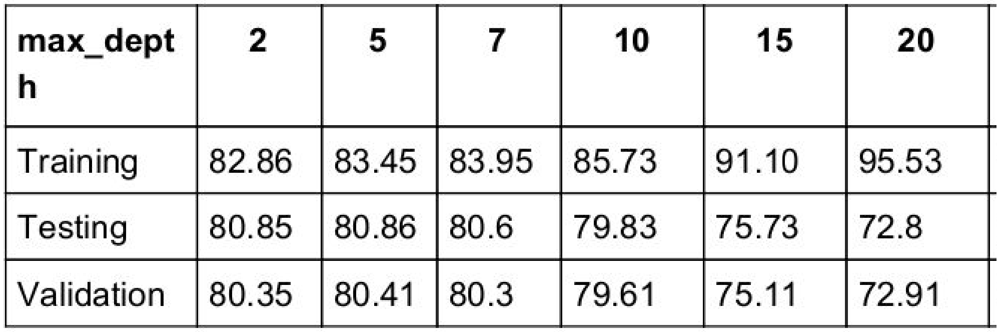
```
(17, [15500.5, 18929.0, 17132.0])
(18, [1300.0, 1211.0])
(19, [1500.0, 1602.0, 1803.0])
(20, [1500.0, 1287.0, 1058.0])
(21, [2882.5, 3360.0, 5000.0])
(22, [3000.0, 938.5])
(23, [1000.0, 202.0])
```

Observations:
The training accuracy boosts to ~99 while testing and validation set accuracy is ~72% which shows how bad it overfits the data. Also as same attributes are split multiple times based on median, number of nodes of tree is increased.

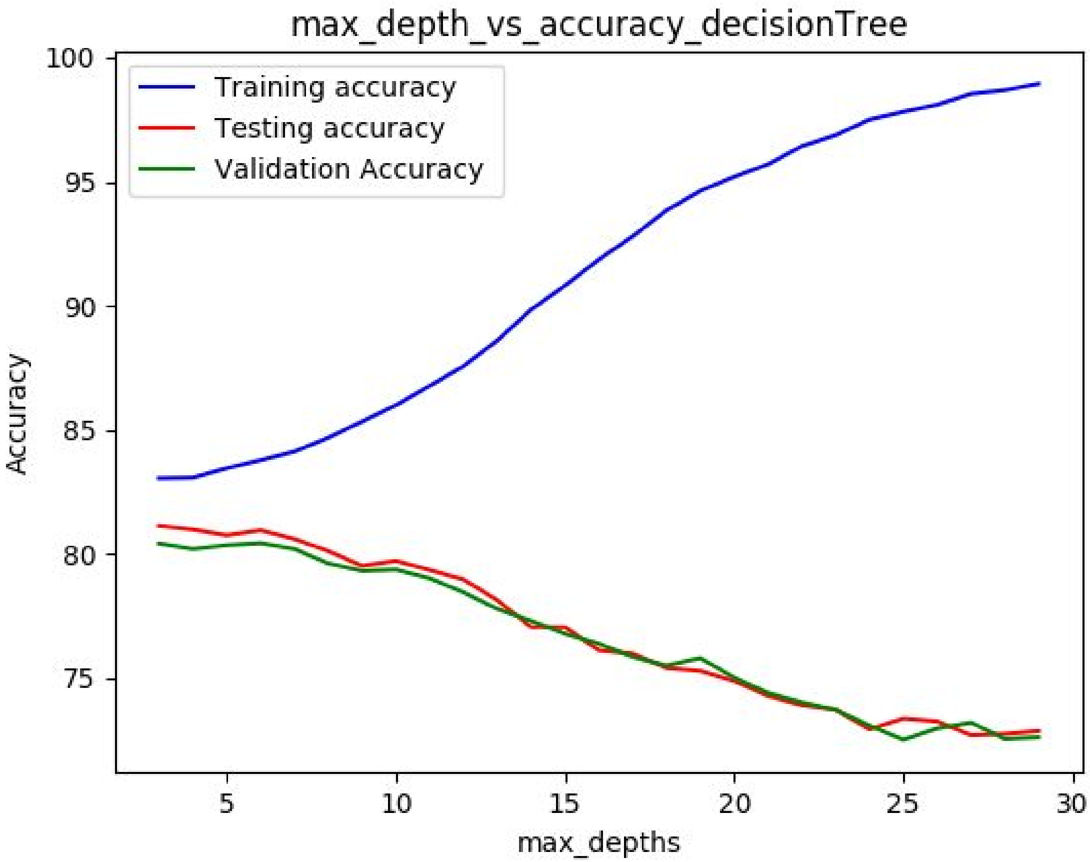
d. Using Sklearn library
Scikit-learn implementation :
(i) min_sample_leaf : A split at any depth will only be considered if it leaves at least min_sample_leaf samples in both left and right branches.
node.
(ii) min_sample_split : Min samples required to split an internal
(iii) max_depth : Max height of the tree.

With default parameters
(max_depth=None,min_samples_split=2,min_samples_leaf=1)

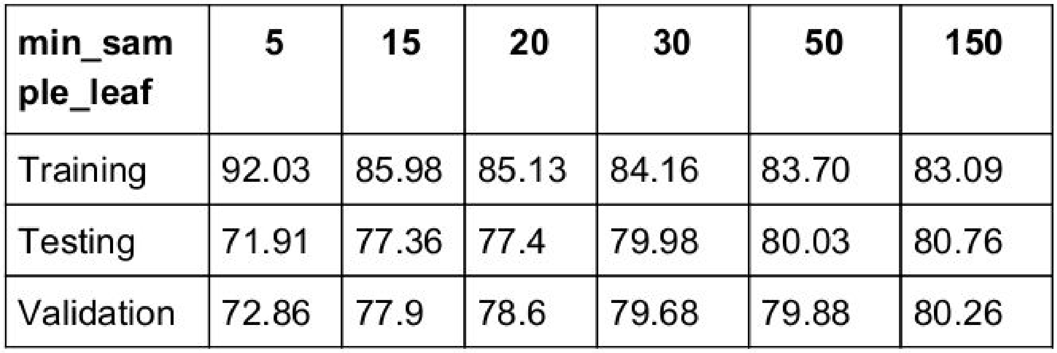
```
(' Accuracy on train : ', 99.96111111111111)
(' Accuracy on valid : ', 72.08333333333333)
(' Accuracy on test : ', 72.11666666666666)
```



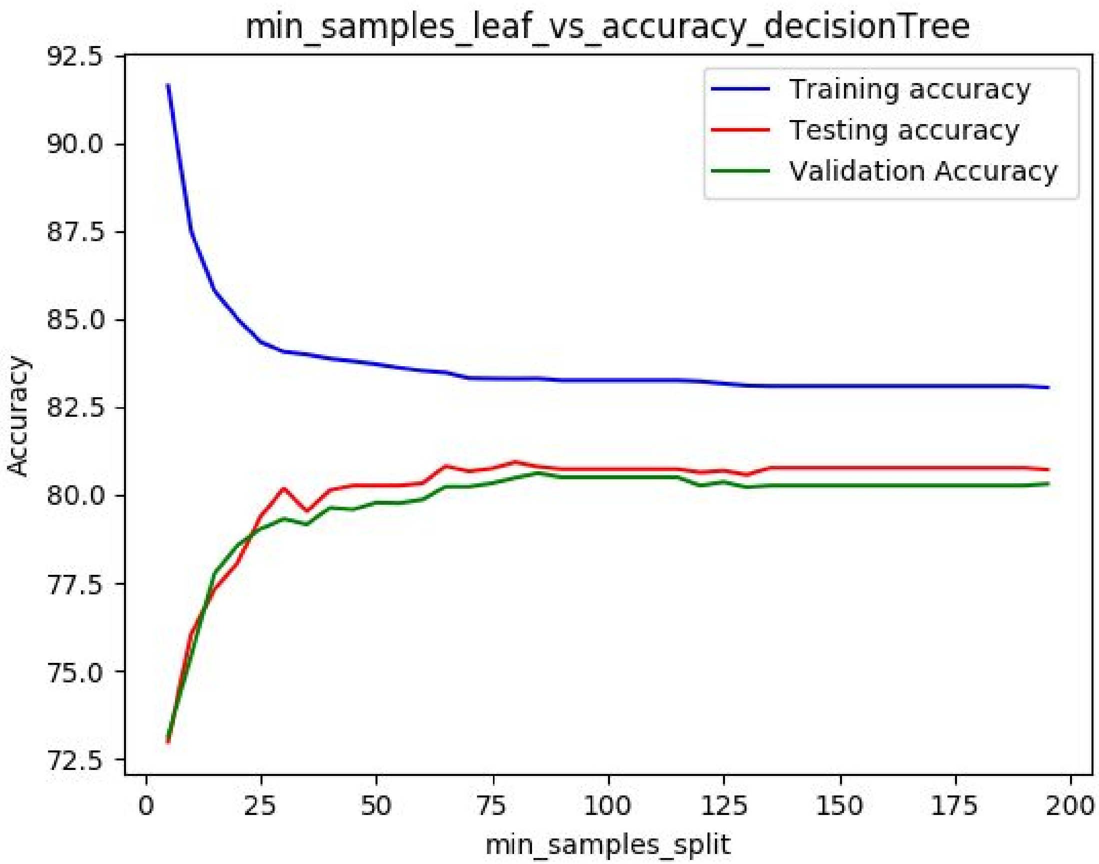
max_depth	2	5	7	10	15	20
Training	82.86	83.45	83.95	85.73	91.10	95.53
Testing	80.85	80.86	80.6	79.83	75.73	72.8
Validation	80.35	80.41	80.3	79.61	75.11	72.91



Validation accuracy drops with increase in max_depth.

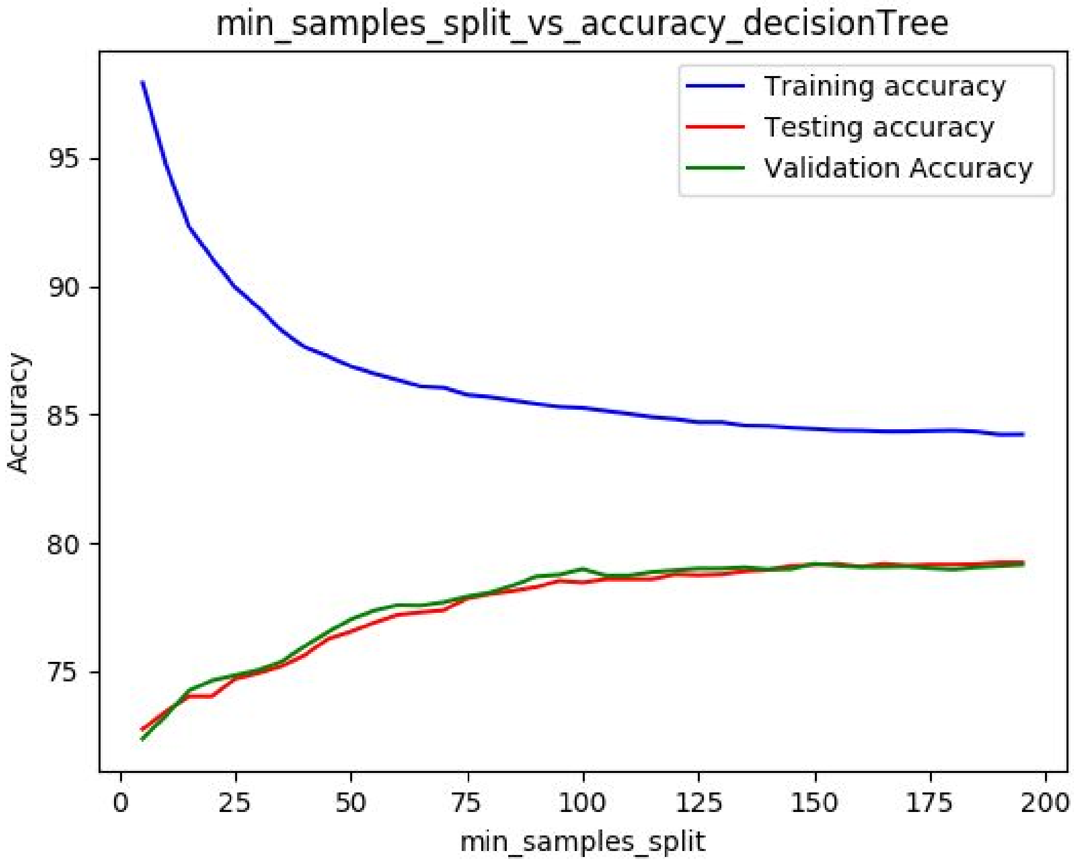


min_sample_leaf	**5**	**15**	**20**	**30**	**50**	**150**
Training	92.03	85.98	85.13	84.16	83.70	83.09
Testing	71.91	77.36	77.4	79.98	80.03	80.76
Validation	72.86	77.9	78.6	79.68	79.88	80.26



Validation accuracy increases with increase in min_samples_leaf.

min_sample_split	**2**	**5**	**10**	**20**	**50**	**100**
Training	100	98.23	94.95	91.20	87.01	85.26
Testing	70.71	70.31	72.38	73.33	75.56	77.6
Validation	71.26	70.45	72.85	73.16	75.95	78.35



Validation accuracy increases with increase in min_samples_split.

After running grid parameter search, parameters with best validation accuracy are:

```
	{'min_samples_split': 95, 'max_depth': 19, 'min_samples_leaf': 85}
	(‘validation Accuracy : ‘ , 80.61666666666667 )
	('Accuracy over training :', 83.30555555555556)
	('Accuracy over testing :', 80.80000000000001)
```

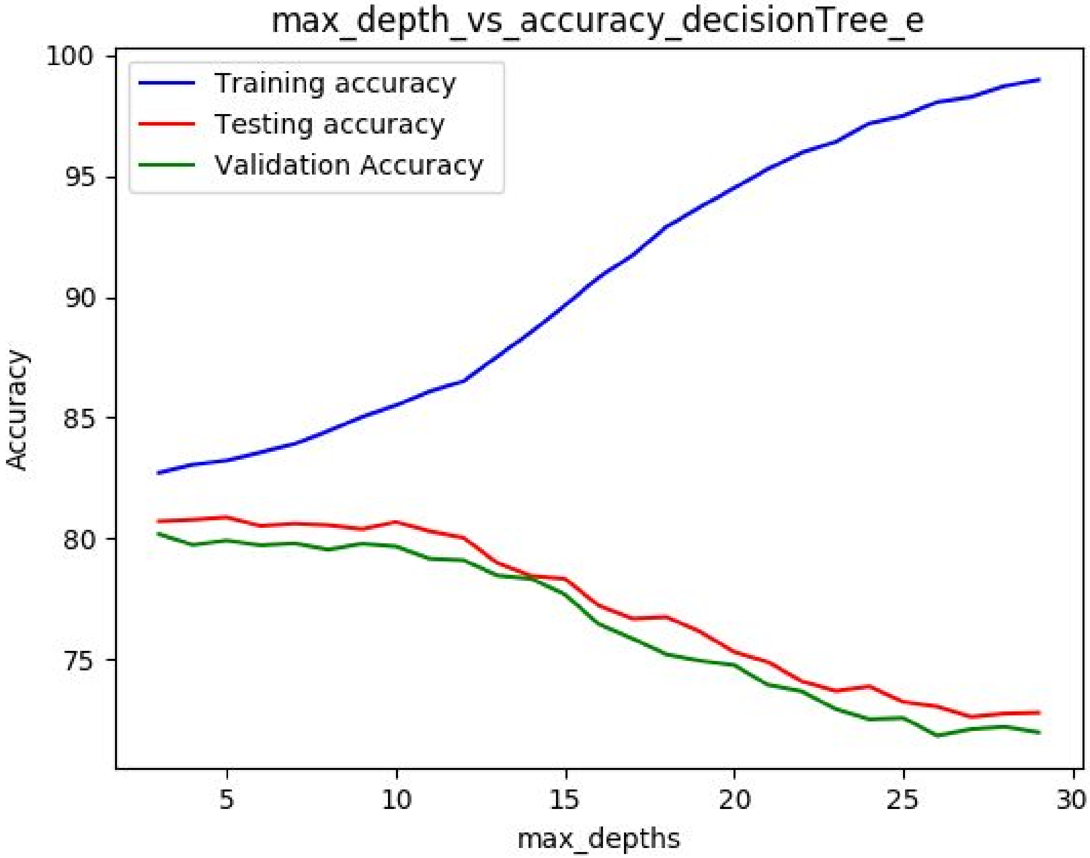
Observations:
Training accuracy decreases than part c and is almost same as of part b. However, testing and validation accuracy is increased than part c and is almost same as that in part b. Therefore, the results it produces are close to the results produced by post pruning.

e. Using one hot encoding
with best parameters from part c
{'min_samples_split': 95, 'max_depth': 19, 'min_samples_leaf': 85}

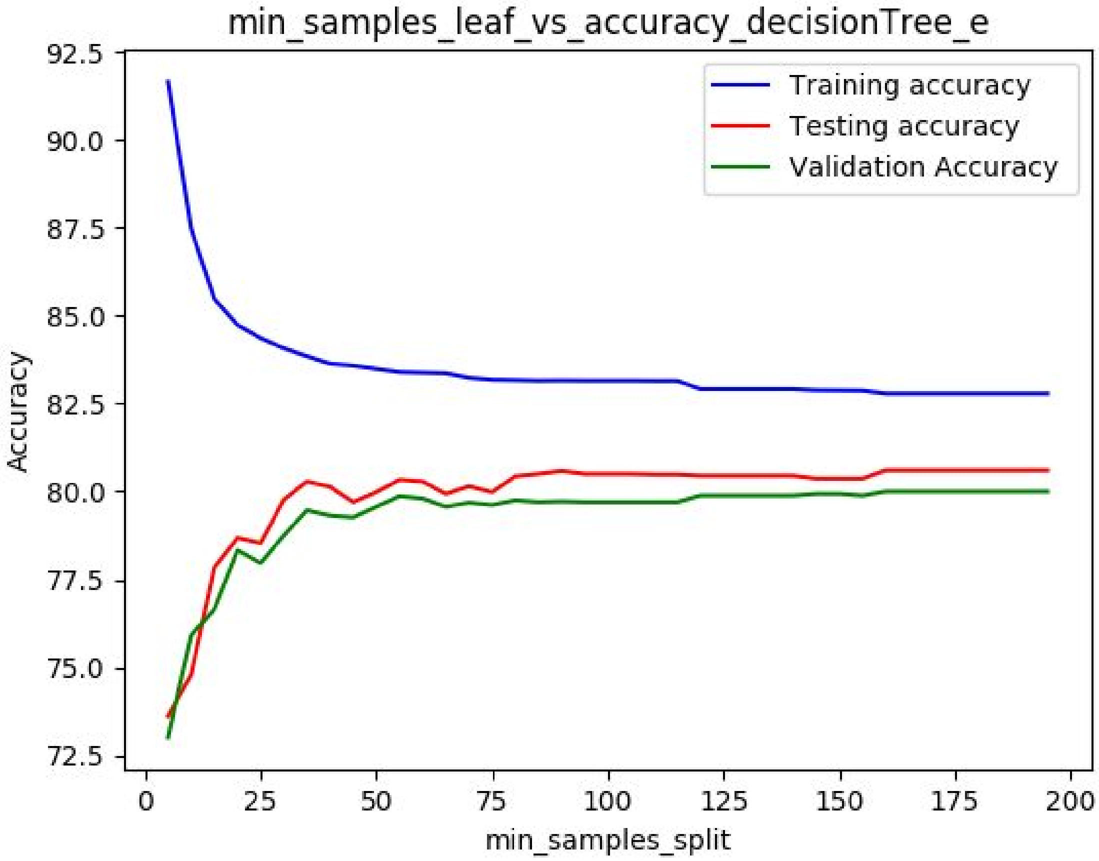
```
('Accuracy over training :', 83.15)
('Accuracy over Validation :', 79.7)
('Accuracy over testing :', 80.5)
```

with default parameters:
{(max_depth=None,min_samples_split=2,min_samples_leaf=1) }

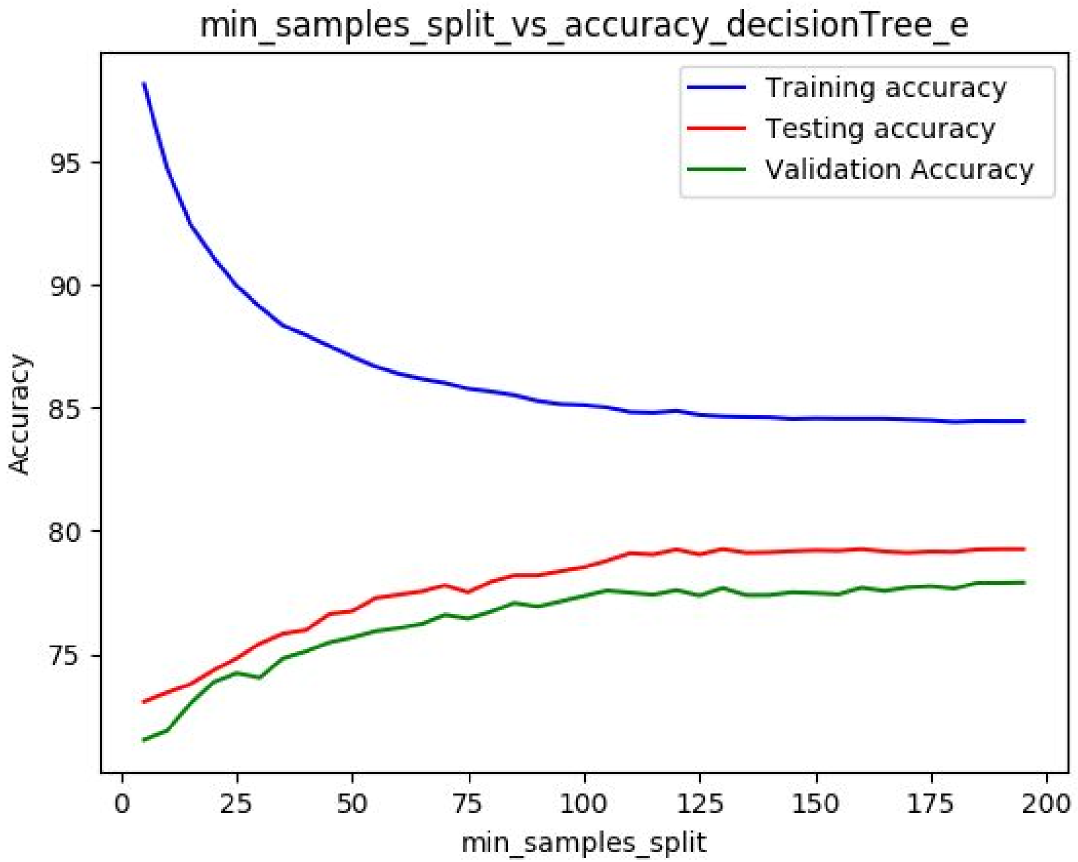
```
(' Accuracy on train : ', 99.96111111111111)
(' Accuracy on valid : ', 71.7)
(' Accuracy on test : ', 72.8)
```



Validation accuracy decreases with increase in max_depth.



Validation accuracy increases with increase in min_samples_leaf



Validation accuracy increases with increase in min_samples_leaf.
Using grid parameter search:
{'min_samples_split': 95, 'max_depth': 5, 'min_samples_leaf': 55}
(‘validation accuracy :’, 80.15)
('Accuracy over training :', 83.08333333333333)
('Accuracy over testing :', 81.0)

Observations:
Training accuracy decreases than part c and is almost same as of part b & d. However, testing and validation accuracy is increased than part c and is almost same as that in part b & d. Therefore, one hot encoding is not showing improvement over the accuracies.

f. Random Forest using sklearn

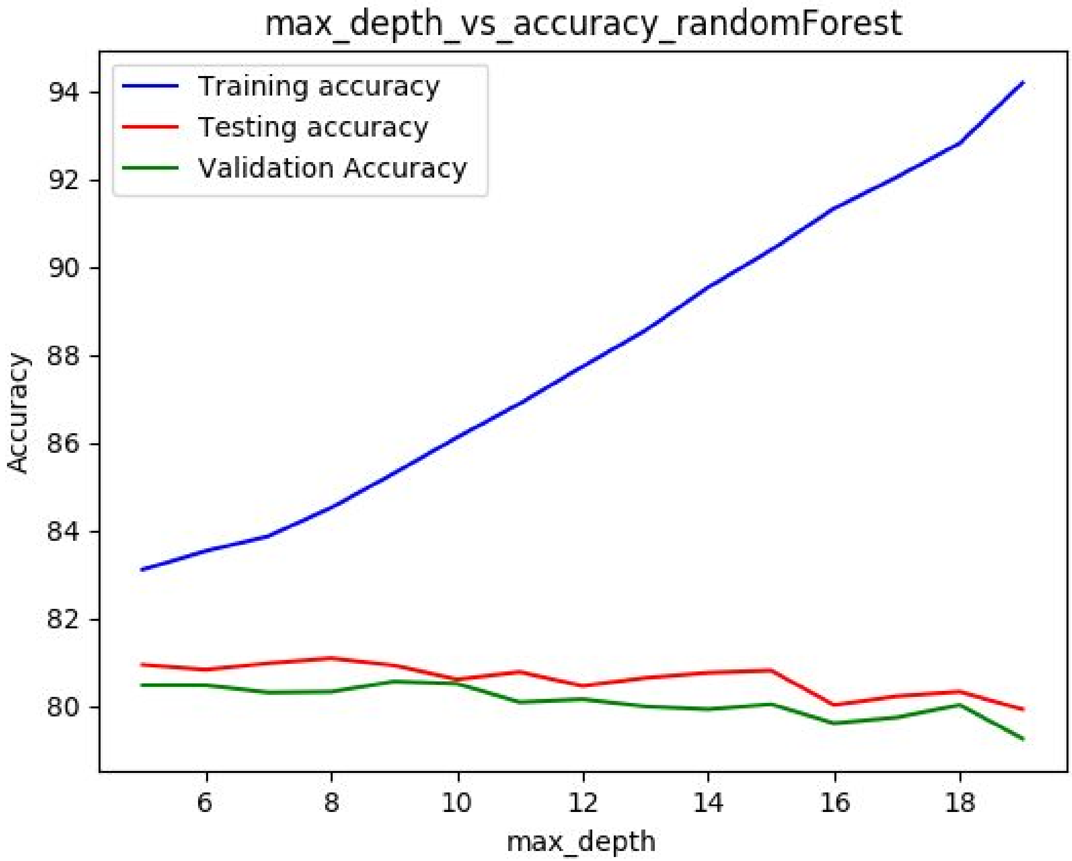
Using default parameters: 'max_features': auto, 'n_estimators': 10, 'bootstrap': True, 'max_depth': None

with default parameters: bootstrap = True

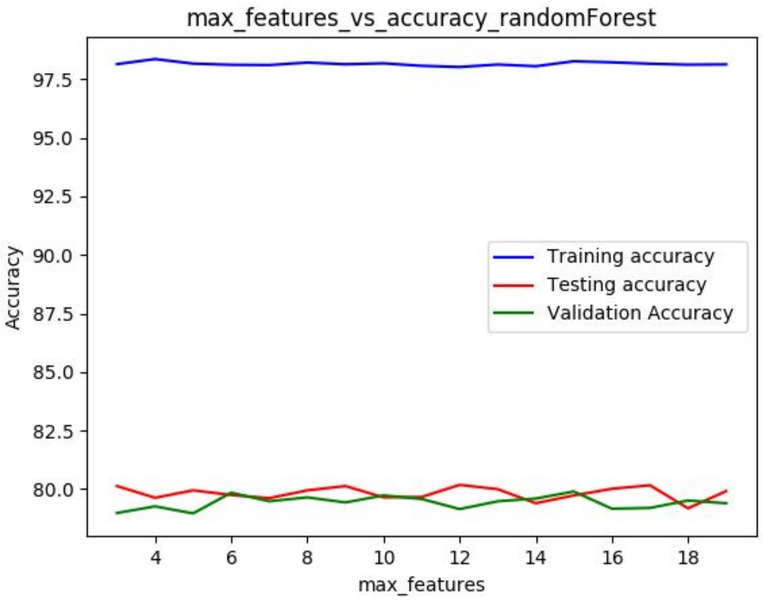
```
(' Accuracy on train : ', 98.36111111111111)
(' Accuracy on valid : ', 79.26666666666667)
(' Accuracy on test : ', 79.63333333333334)
```

with default parameters and bootstrap = False:

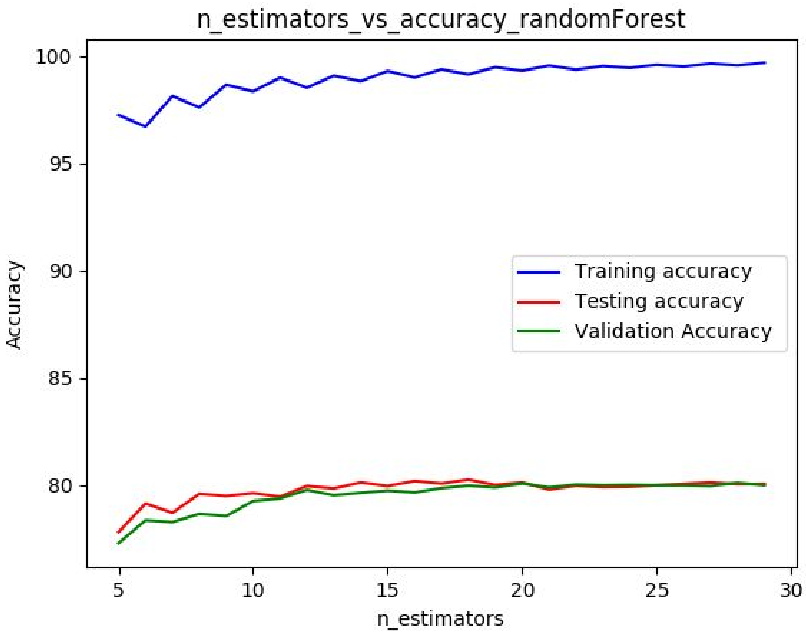
```
(' Accuracy on train : ', 99.96111111111111)
(' Accuracy on valid : ', 79.61666666666667)
(' Accuracy on test : ', 79.21666666666667)
```



Validation accuracy decreases at some values and increases at some i.e. no fixed pattern is there.



On max_features also, validation accuracy increases at some values and decreases at some i.e. no fixed pattern is followed.



Validation accuracy increases with n_estimators i.e. number of trees in forest.

Best parameters using grid search:

```
{'max_features': 4, 'n_estimators': 7, 'bootstrap': True, 'max_depth': 8}
(‘validation accuracy:’, 80.63333333333334 )
('Accuracy over training :', 84.48333333333333)
('Accuracy over testing :', 80.91666666666667)
```

Observations :

Training accuracy decreases than part c and is almost same as of part b, d & e. However, testing and validation accuracy is increased than part c and is almost same as that in part b, d & e. Therefore, random forest generalizes quite well as done by post-pruning.

Question2.(Neural Network)

a. The link for the one-hot encoding of train and test data is as follows:
https://drive.google.com/open?id=1FL6RSb1uUyYtjRmrTCMTIVFrpQn-cmgs

In all the question, used batchsize = 100

b. Neural Network implemented
Following accuracies are according to the parameters :
With 25 neuron and single hidden layer, sigmoid activation function , constant learning rate
Eta = 0.1
Epochs = 1500
Error threshold = 10 ** -16 (absolute difference between old error and new error)
Used two criteria for stopping the convergence i.e.
either max epochs reached or error threshold reached

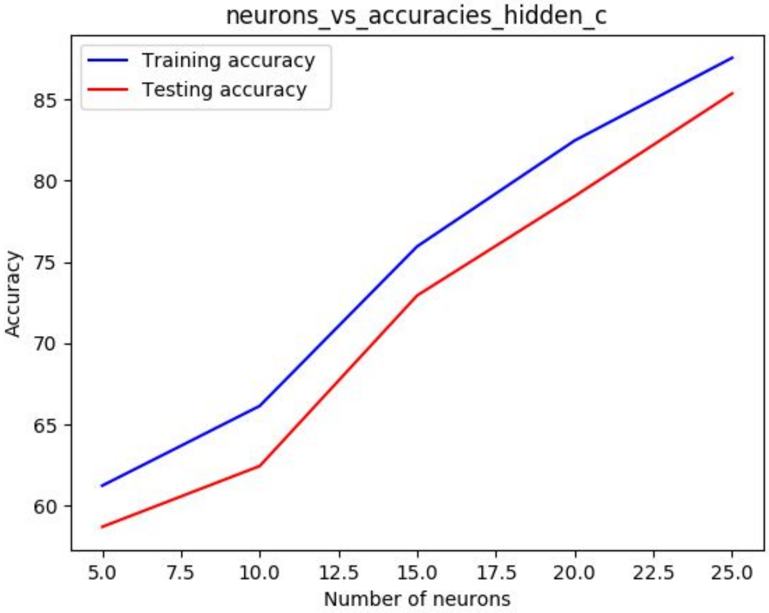
```
('Accuracy On training :', 92.80727708916433)
('Accuracy On testing :', 92.5209)
```

c. Single hidden layer.
single hidden layer, sigmoid activation function , constant learning rate
The neural network was tested with a single hidden layer and by varying number of units in that layer.
Number of neurons: [5, 10, 15, 20, 25]
Eta = 0.1
Stopping criteria :
Epochs = 1500
Error threshold = 10 ** -16 (absolute difference between old error and new error)

Following are training and testing accuracies:

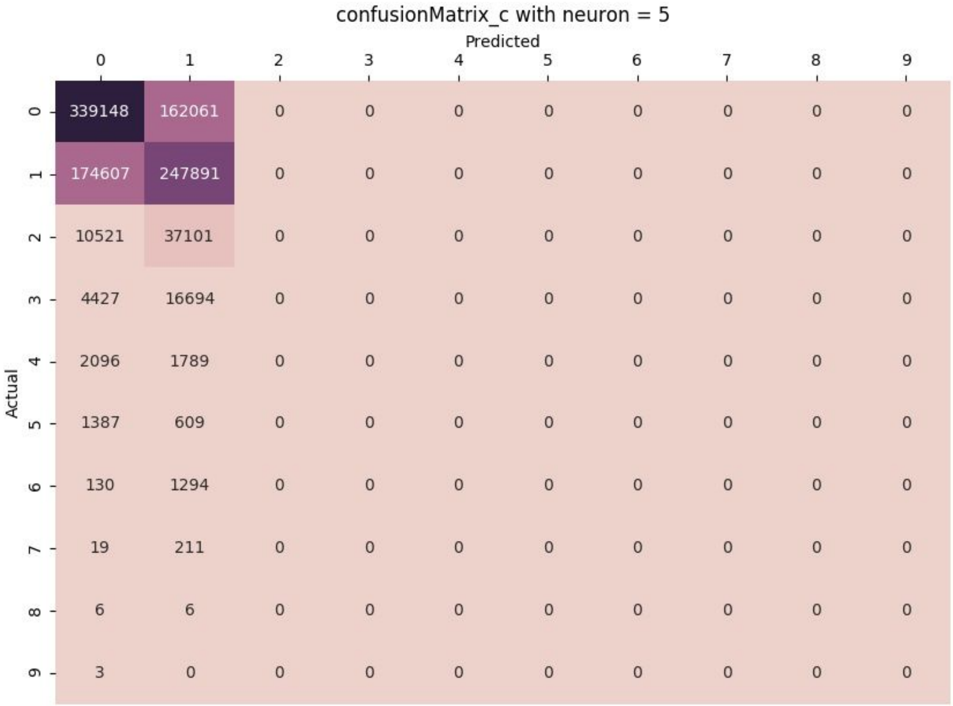
```
[5, 10, 15, 20, 25],
{'test': [58.7039, 62.4453, 72.9424, 79.0528, 85.3738],
'train': [61.239504198320674, 66.14554178328669, 75.96161535385846,
82.4750099960016, 87.56497401039584]}
```

Execution time for training = [2223.289870024, 3286.122042894, 4350.20216608, 5410.647963047, 7474.370280981]



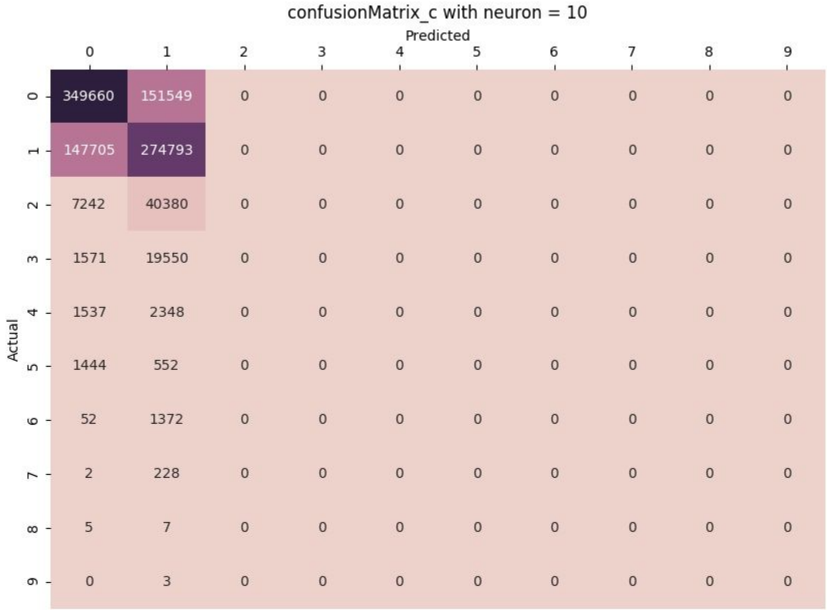
By increasing the number of units in the hidden layer accuracy has gone up. This may be because of the fact that with more neurons we get more parameters and our model learns better. But if we increase it by large number, the model may overfit.

For 5 neurons in single hidden layer


confusionMatrix_c with neuron = 5

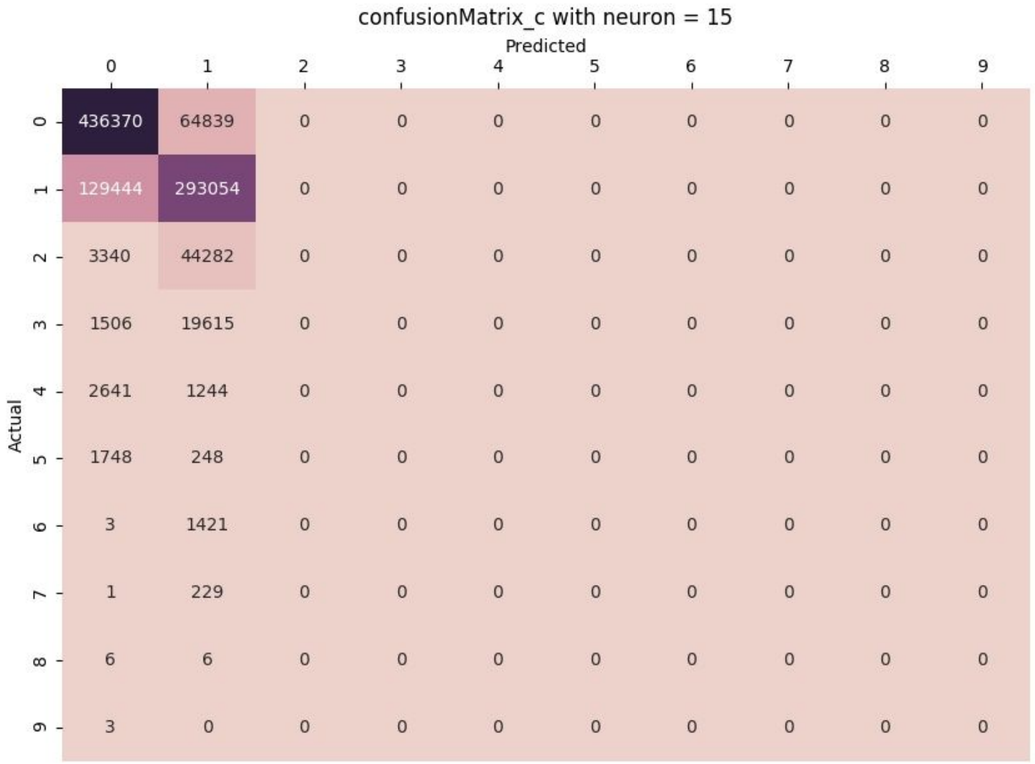
	0	1	2	3	4	5	6	7	8	9
0	339148	162061	0	0	0	0	0	0	0	0
1	174607	247891	0	0	0	0	0	0	0	0
2	10521	37101	0	0	0	0	0	0	0	0
3	4427	16694	0	0	0	0	0	0	0	0
4	2096	1789	0	0	0	0	0	0	0	0
5	1387	609	0	0	0	0	0	0	0	0
6	130	1294	0	0	0	0	0	0	0	0
7	19	211	0	0	0	0	0	0	0	0
8	6	6	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

For 10 neurons in hidden layer


confusionMatrix_c with neuron = 10

	0	1	2	3	4	5	6	7	8	9
0	349660	151549	0	0	0	0	0	0	0	0
1	147705	274793	0	0	0	0	0	0	0	0
2	7242	40380	0	0	0	0	0	0	0	0
3	1571	19550	0	0	0	0	0	0	0	0
4	1537	2348	0	0	0	0	0	0	0	0
5	1444	552	0	0	0	0	0	0	0	0
6	52	1372	0	0	0	0	0	0	0	0
7	2	228	0	0	0	0	0	0	0	0
8	5	7	0	0	0	0	0	0	0	0
9	0	3	0	0	0	0	0	0	0	0

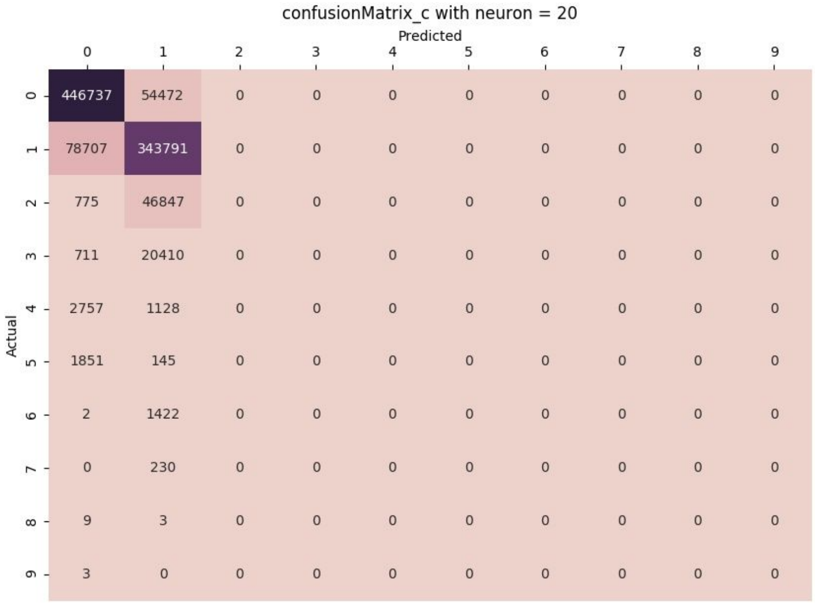
For 15 neurons in hidden layer



confusionMatrix_c with neuron = 15

	0	1	2	3	4	5	6	7	8	9
0	436370	64839	0	0	0	0	0	0	0	0
1	129444	293054	0	0	0	0	0	0	0	0
2	3340	44282	0	0	0	0	0	0	0	0
3	1506	19615	0	0	0	0	0	0	0	0
4	2641	1244	0	0	0	0	0	0	0	0
5	1748	248	0	0	0	0	0	0	0	0
6	3	1421	0	0	0	0	0	0	0	0
7	1	229	0	0	0	0	0	0	0	0
8	6	6	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

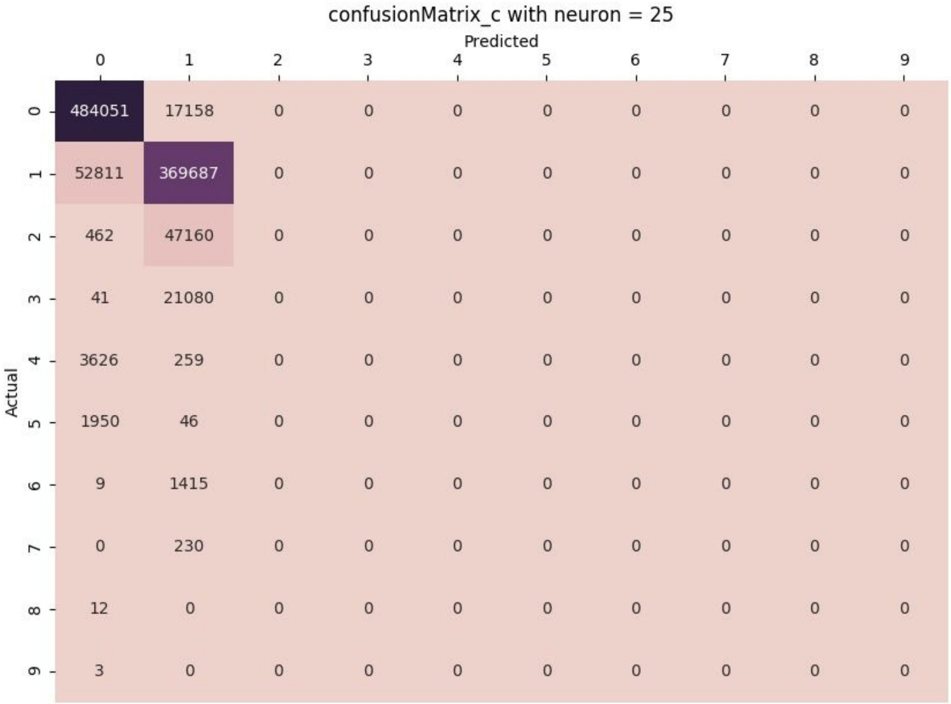
For 20 neurons in hidden layer



confusionMatrix_c with neuron = 20

	0	1	2	3	4	5	6	7	8	9
0	446737	54472	0	0	0	0	0	0	0	0
1	78707	343791	0	0	0	0	0	0	0	0
2	775	46847	0	0	0	0	0	0	0	0
3	711	20410	0	0	0	0	0	0	0	0
4	2757	1128	0	0	0	0	0	0	0	0
5	1851	145	0	0	0	0	0	0	0	0
6	2	1422	0	0	0	0	0	0	0	0
7	0	230	0	0	0	0	0	0	0	0
8	9	3	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

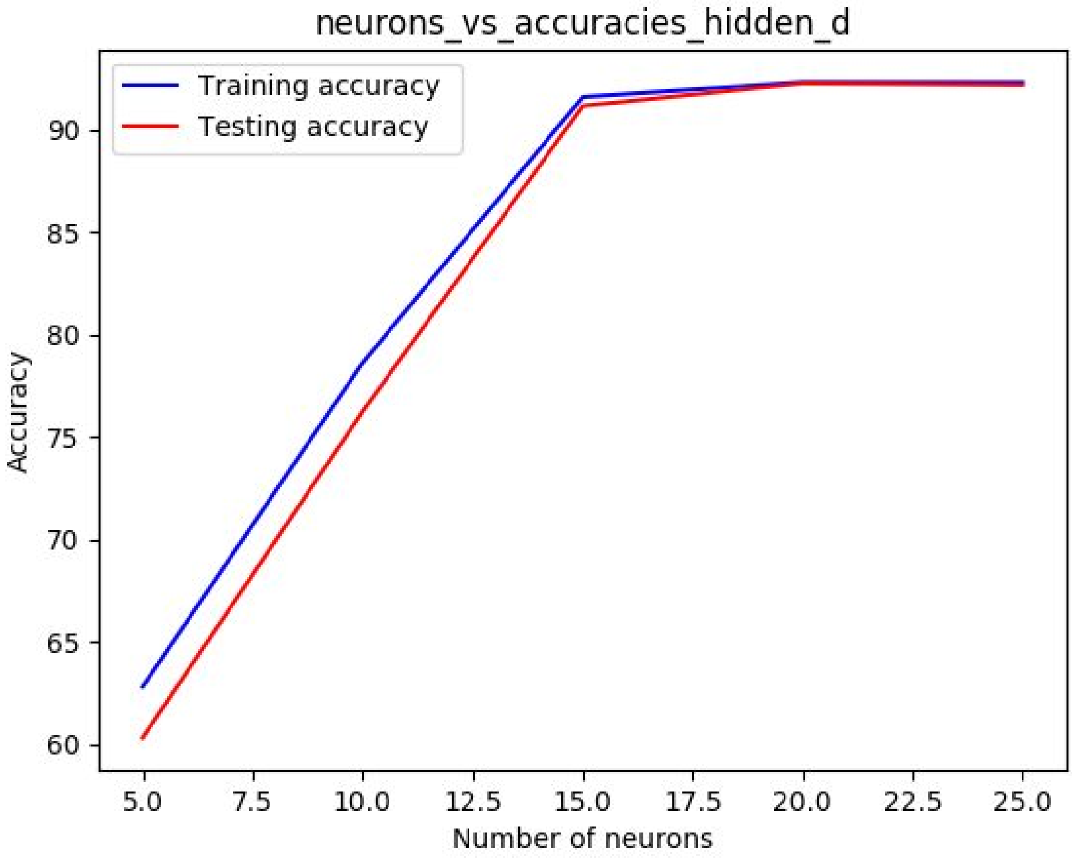
For 25 neurons in hidden layer



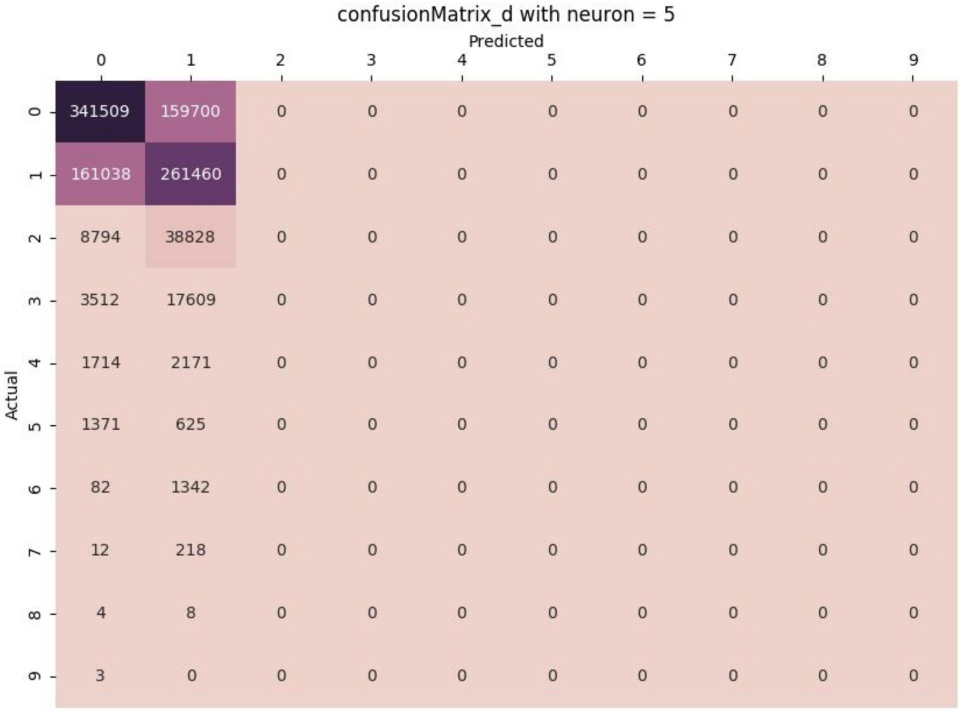
	0	1	2	3	4	5	6	7	8	9
0	484051	17158	0	0	0	0	0	0	0	0
1	52811	369687	0	0	0	0	0	0	0	0
2	462	47160	0	0	0	0	0	0	0	0
3	41	21080	0	0	0	0	0	0	0	0
4	3626	259	0	0	0	0	0	0	0	0
5	1950	46	0	0	0	0	0	0	0	0
6	9	1415	0	0	0	0	0	0	0	0
7	0	230	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

d. 2 hidden layers and same neurons in both of them
two hidden layer, sigmoid activation function , constant learning rate
The neural network was tested with a single hidden layer and by varying number of units in that layer.
Number of neurons: [5, 10, 15, 20, 25]
Eta = 0.1
Stopping criteria :
Epochs = 1500
Error threshold = 10 ** -16 (absolute difference between old error and new error)
Following are testing and training accuracies:
Neurons = [5, 10, 15, 20, 25],
{'test': [60.2969, 76.2335, 91.17, 92.271, 92.1917],
'train': [62.78688524590164, 78.6125549780088, 91.60335865653738, 92.33106757297081, 92.32307077169132]

Execution time for training = [3487.135553122, 5689.330428123, 6866.216413021, 8008.11416101, 10414.18355107]



For 5 5 neurons in two hidden layers

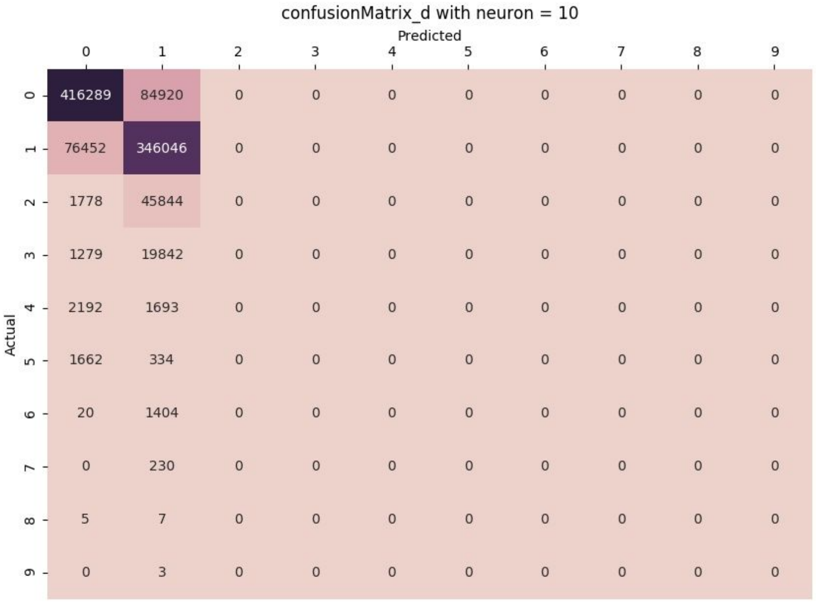


confusionMatrix_d with neuron = 5

Predicted

Actual	0	1	2	3	4	5	6	7	8	9
0	341509	159700	0	0	0	0	0	0	0	0
1	161038	261460	0	0	0	0	0	0	0	0
2	8794	38828	0	0	0	0	0	0	0	0
3	3512	17609	0	0	0	0	0	0	0	0
4	1714	2171	0	0	0	0	0	0	0	0
5	1371	625	0	0	0	0	0	0	0	0
6	82	1342	0	0	0	0	0	0	0	0
7	12	218	0	0	0	0	0	0	0	0
8	4	8	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

For 10 10 neurons in two hidden layers

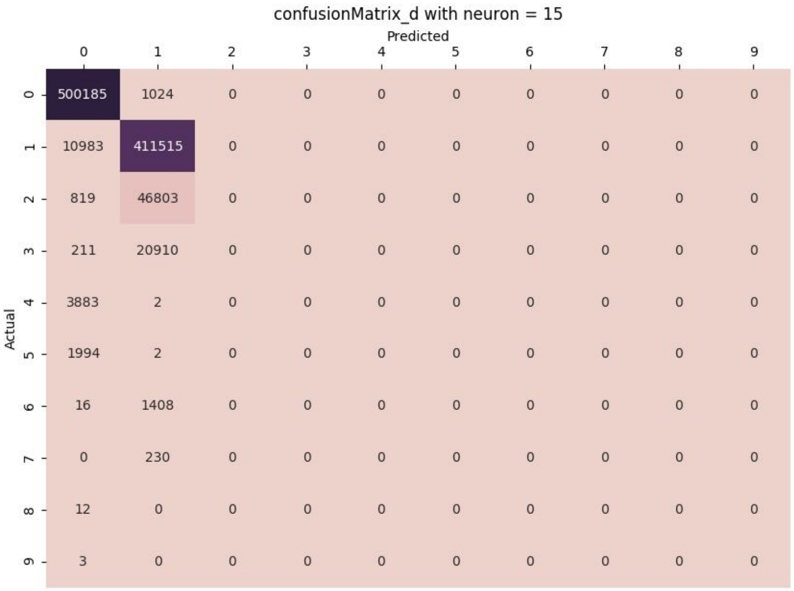


confusionMatrix_d with neuron = 10

Predicted

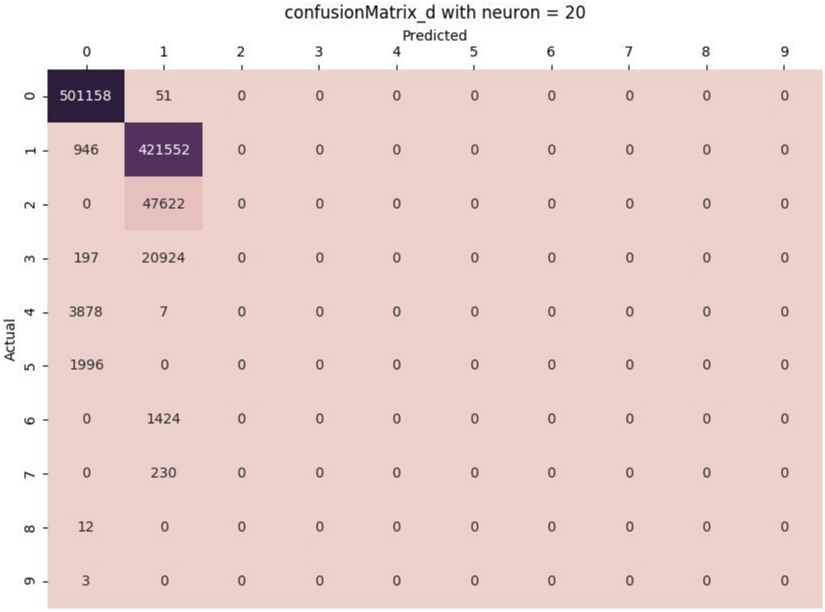
Actual	0	1	2	3	4	5	6	7	8	9
0	416289	84920	0	0	0	0	0	0	0	0
1	76452	346046	0	0	0	0	0	0	0	0
2	1778	45844	0	0	0	0	0	0	0	0
3	1279	19842	0	0	0	0	0	0	0	0
4	2192	1693	0	0	0	0	0	0	0	0
5	1662	334	0	0	0	0	0	0	0	0
6	20	1404	0	0	0	0	0	0	0	0
7	0	230	0	0	0	0	0	0	0	0
8	5	7	0	0	0	0	0	0	0	0
9	0	3	0	0	0	0	0	0	0	0

For 15 15 neurons in two hidden layers


confusionMatrix_d with neuron = 15
Predicted
Actual

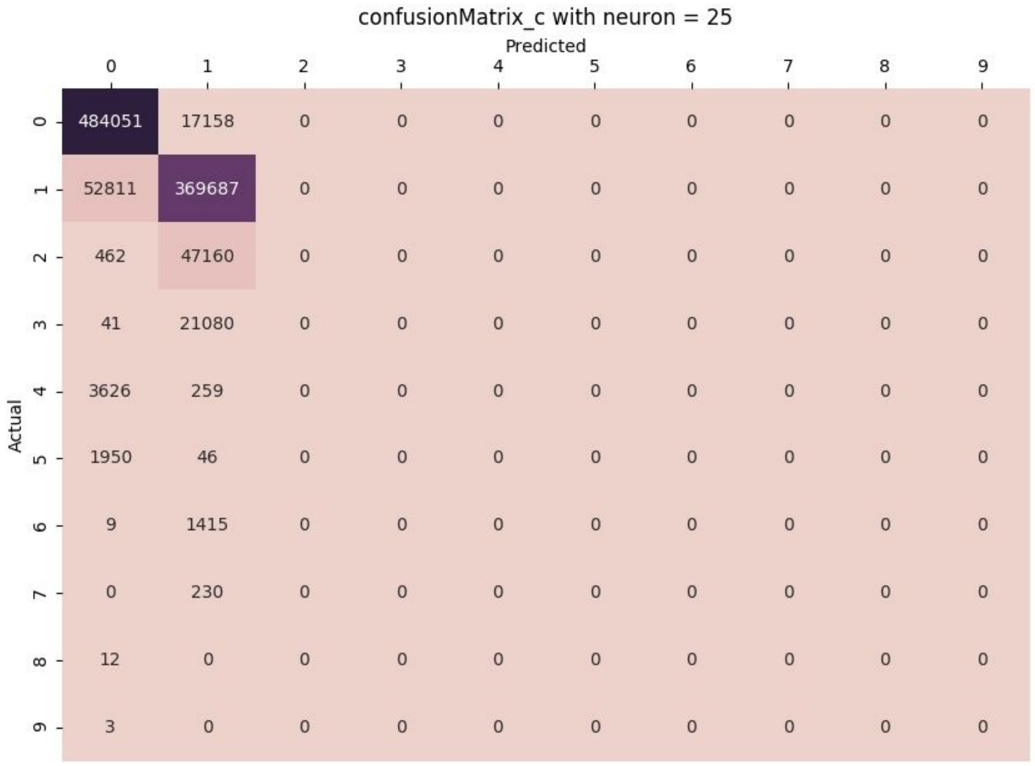
Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	500185	1024	0	0	0	0	0	0	0	0
1	10983	411515	0	0	0	0	0	0	0	0
2	819	46803	0	0	0	0	0	0	0	0
3	211	20910	0	0	0	0	0	0	0	0
4	3883	2	0	0	0	0	0	0	0	0
5	1994	2	0	0	0	0	0	0	0	0
6	16	1408	0	0	0	0	0	0	0	0
7	0	230	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

For 20 20 neurons in two hidden layers


confusionMatrix_d with neuron = 20
Predicted
Actual

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	501158	51	0	0	0	0	0	0	0	0
1	946	421552	0	0	0	0	0	0	0	0
2	0	47622	0	0	0	0	0	0	0	0
3	197	20924	0	0	0	0	0	0	0	0
4	3878	7	0	0	0	0	0	0	0	0
5	1996	0	0	0	0	0	0	0	0	0
6	0	1424	0	0	0	0	0	0	0	0
7	0	230	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

For 25 25 neurons in two hidden layers



confusionMatrix_c with neuron = 25

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	484051	17158	0	0	0	0	0	0	0	0
1	52811	369687	0	0	0	0	0	0	0	0
2	462	47160	0	0	0	0	0	0	0	0
3	41	21080	0	0	0	0	0	0	0	0
4	3626	259	0	0	0	0	0	0	0	0
5	1950	46	0	0	0	0	0	0	0	0
6	9	1415	0	0	0	0	0	0	0	0
7	0	230	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

e. Adaptive learning rate

There wasn’t any improvement in the accuracy when adaptive learning with tol = 10 ** -4 was used. Some accuracies remained same as earlier while some became even worse.

Single and Double hidden layer, sigmoid activation function , adaptive learning rate

The neural network was tested with a single hidden layer then double hidden layer and by varying number of units in that layer.

Number of neurons: [5, 10, 15, 20, 25]

Eta = 0.1

Stopping criteria :

Epochs = 800

Threshold = 10 ** -25

i) Single Hidden Layer

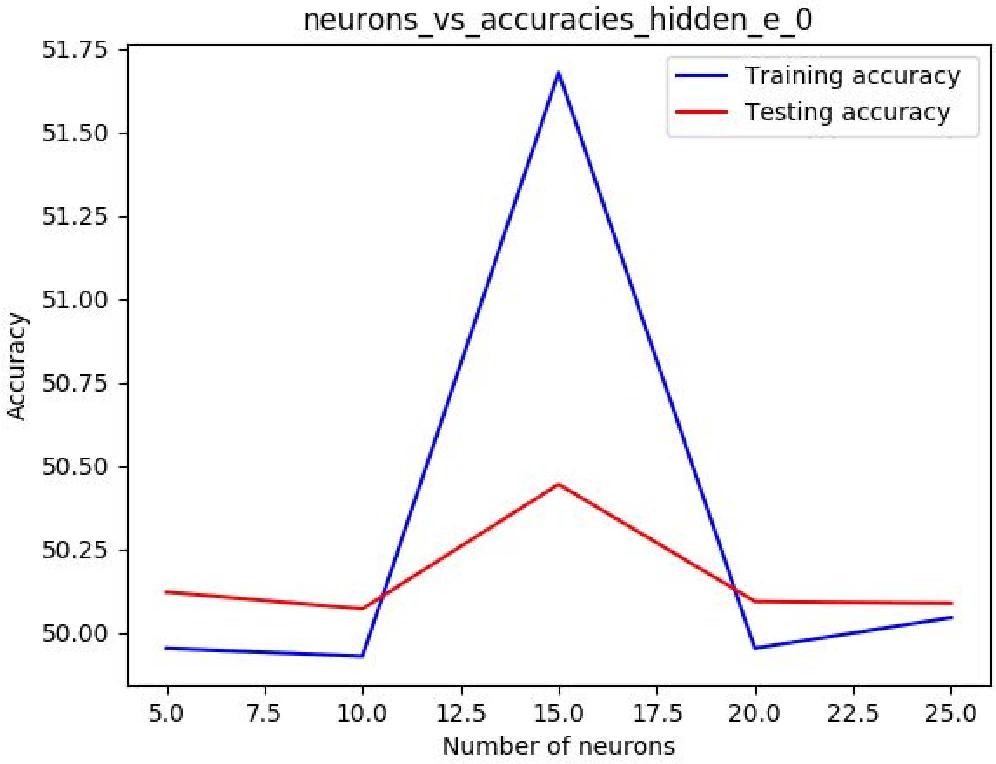
Following are testing and training accuracies:

Neurons = [5, 10, 15, 20, 25],

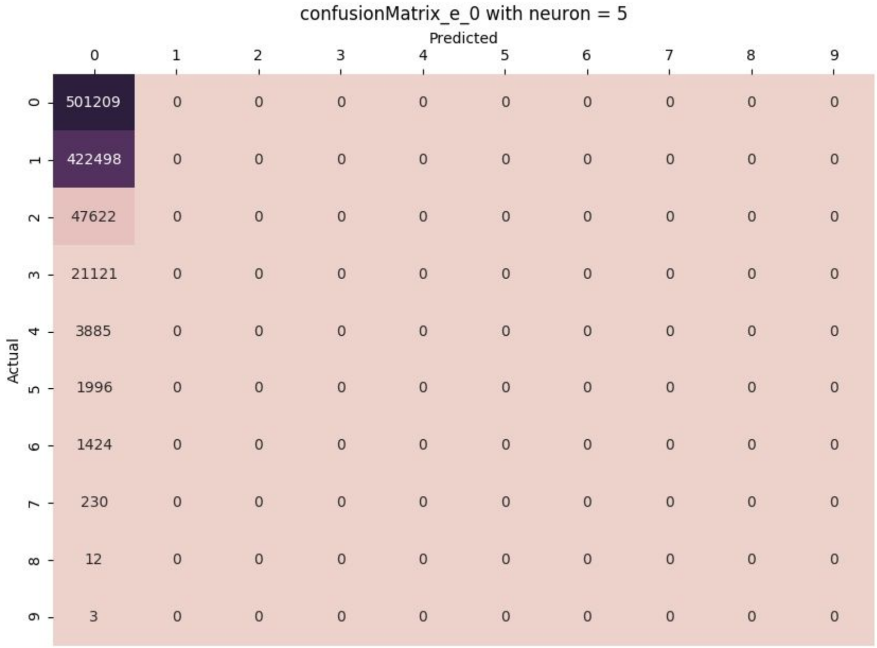
{'test': [50.1209, 50.0705, 50.4442, 50.093,50.0873],

'train': [49.95201919232307, 49.9280287884846,51.67932826869252, 49.95201919232307 ,50.043982407037184]

Execution time for training = [353.9656729698, 464.0275249481, 1134.224796057, 881.2413668633, 1108.448998928]



For 5 neurons in single layer

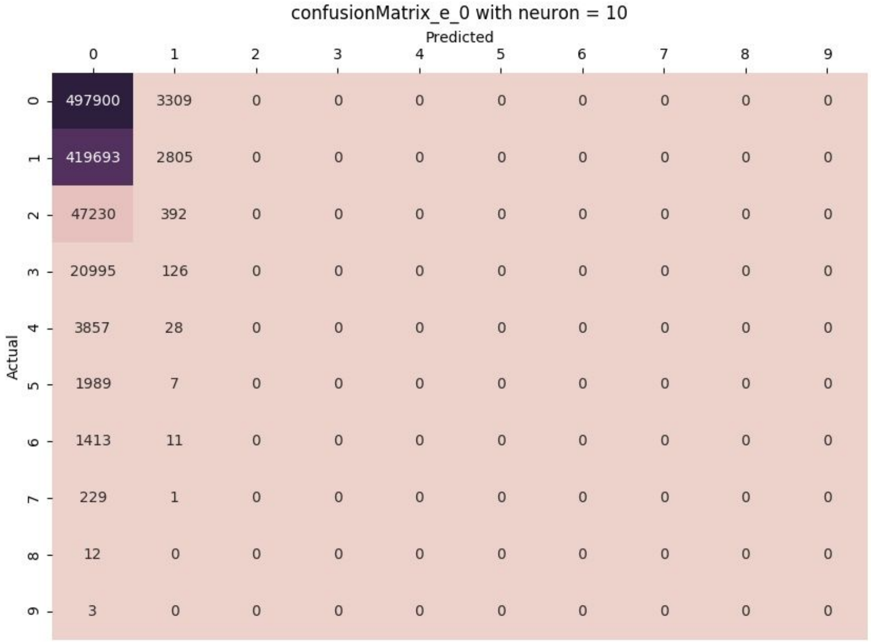


confusionMatrix_e_0 with neuron = 5

Predicted

Actual	0	1	2	3	4	5	6	7	8	9
0	501209	0	0	0	0	0	0	0	0	0
1	422498	0	0	0	0	0	0	0	0	0
2	47622	0	0	0	0	0	0	0	0	0
3	21121	0	0	0	0	0	0	0	0	0
4	3885	0	0	0	0	0	0	0	0	0
5	1996	0	0	0	0	0	0	0	0	0
6	1424	0	0	0	0	0	0	0	0	0
7	230	0	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

For 10 neurons in single layer

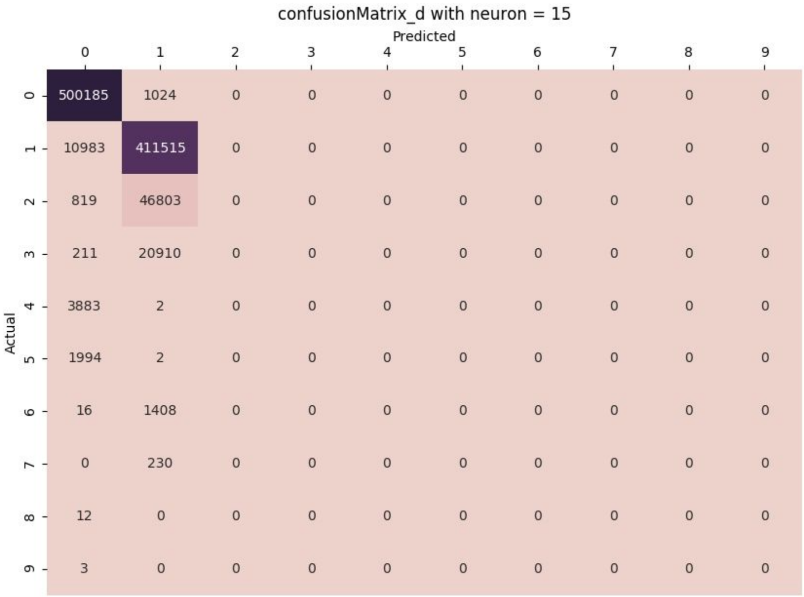


confusionMatrix_e_0 with neuron = 10

Predicted

Actual	0	1	2	3	4	5	6	7	8	9
0	497900	3309	0	0	0	0	0	0	0	0
1	419693	2805	0	0	0	0	0	0	0	0
2	47230	392	0	0	0	0	0	0	0	0
3	20995	126	0	0	0	0	0	0	0	0
4	3857	28	0	0	0	0	0	0	0	0
5	1989	7	0	0	0	0	0	0	0	0
6	1413	11	0	0	0	0	0	0	0	0
7	229	1	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

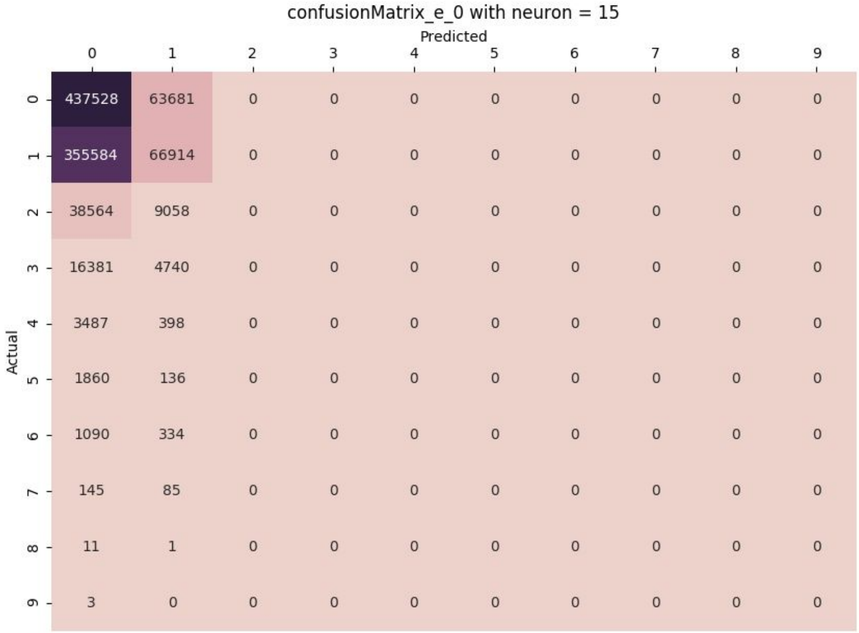
For 15 neurons in single layer



confusionMatrix_d with neuron = 15

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	500185	1024	0	0	0	0	0	0	0	0
1	10983	411515	0	0	0	0	0	0	0	0
2	819	46803	0	0	0	0	0	0	0	0
3	211	20910	0	0	0	0	0	0	0	0
4	3883	2	0	0	0	0	0	0	0	0
5	1994	2	0	0	0	0	0	0	0	0
6	16	1408	0	0	0	0	0	0	0	0
7	0	230	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

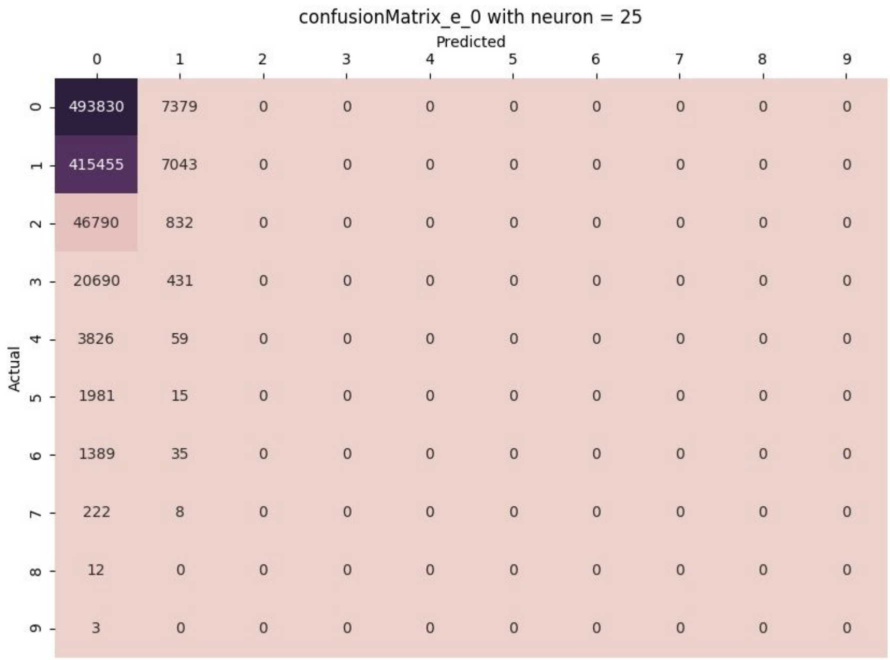
For 20 neurons in single layer



confusionMatrix_e_0 with neuron = 15

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	437528	63681	0	0	0	0	0	0	0	0
1	355584	66914	0	0	0	0	0	0	0	0
2	38564	9058	0	0	0	0	0	0	0	0
3	16381	4740	0	0	0	0	0	0	0	0
4	3487	398	0	0	0	0	0	0	0	0
5	1860	136	0	0	0	0	0	0	0	0
6	1090	334	0	0	0	0	0	0	0	0
7	145	85	0	0	0	0	0	0	0	0
8	11	1	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

For 25 neurons in single layer

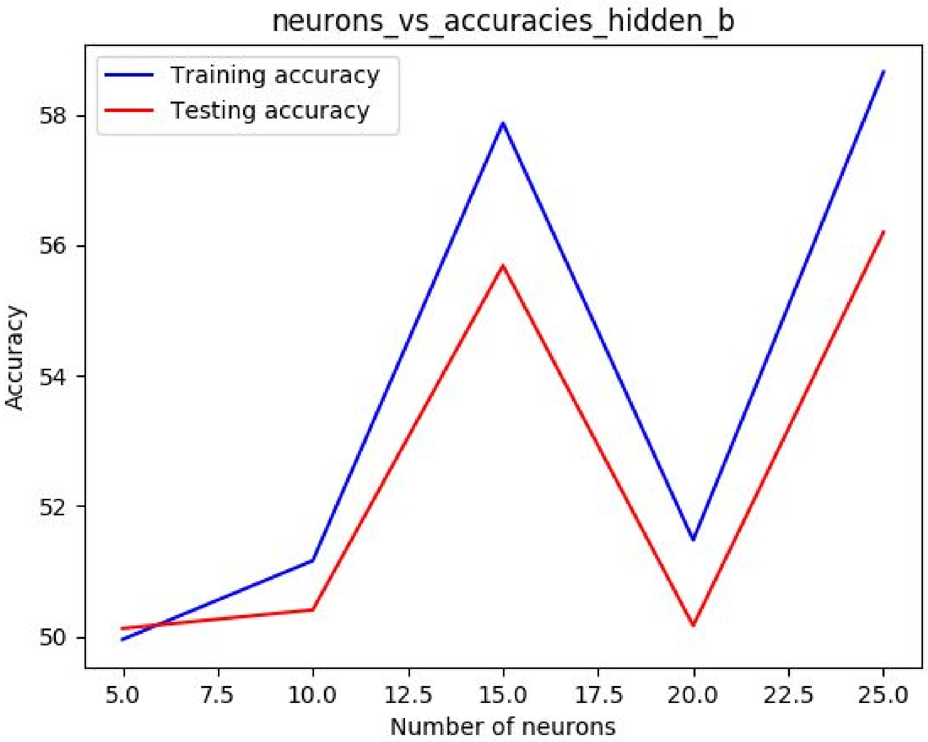


confusionMatrix_e_0 with neuron = 25

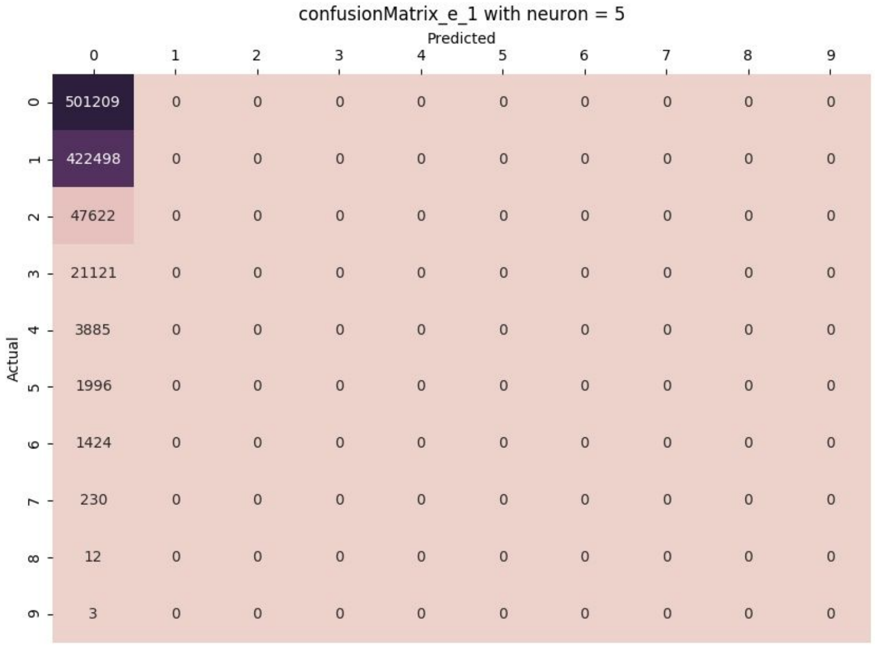
	0	1	2	3	4	5	6	7	8	9
0	493830	7379	0	0	0	0	0	0	0	0
1	415455	7043	0	0	0	0	0	0	0	0
2	46790	832	0	0	0	0	0	0	0	0
3	20690	431	0	0	0	0	0	0	0	0
4	3826	59	0	0	0	0	0	0	0	0
5	1981	15	0	0	0	0	0	0	0	0
6	1389	35	0	0	0	0	0	0	0	0
7	222	8	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

ii) For Two Hidden Layers:

Following are testing and training accuracies:
Neurons = [5, 10, 15, 20, 25],
{'test': [50.1209, 50.4047, 55.6858,50.1633, 56.196],
'train': [49.95201919232307, 51.15953618552579, 57.87285085965614, 51.47540983606557, 58.652538984406235]
Execution time for training = [173.7517058849, 591.768998861, 1229.940575123, 1493.387754202, 2595.921108961]



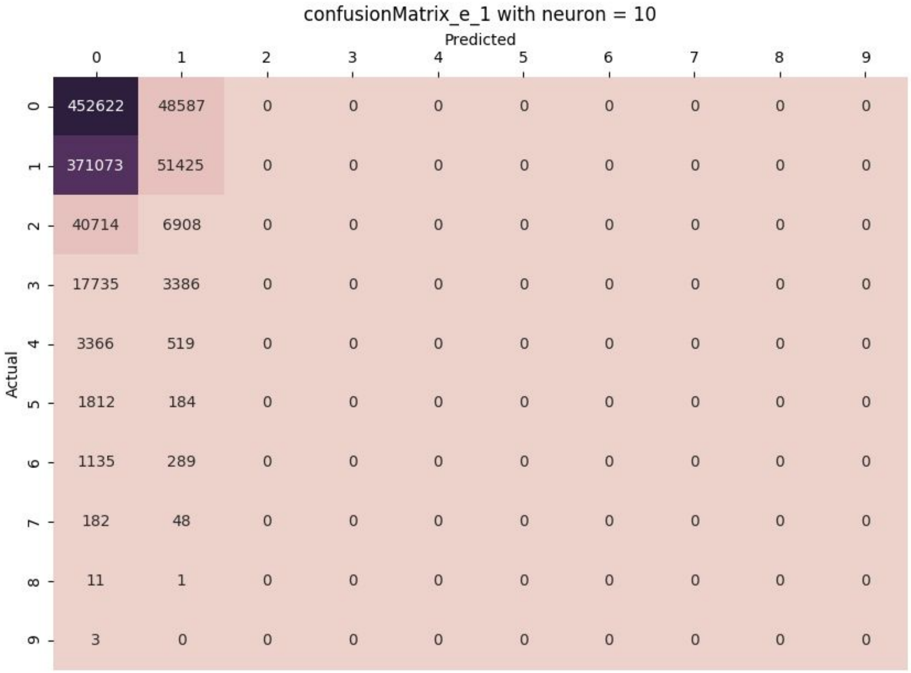
For 5 neurons in two layer



confusionMatrix_e_1 with neuron = 5

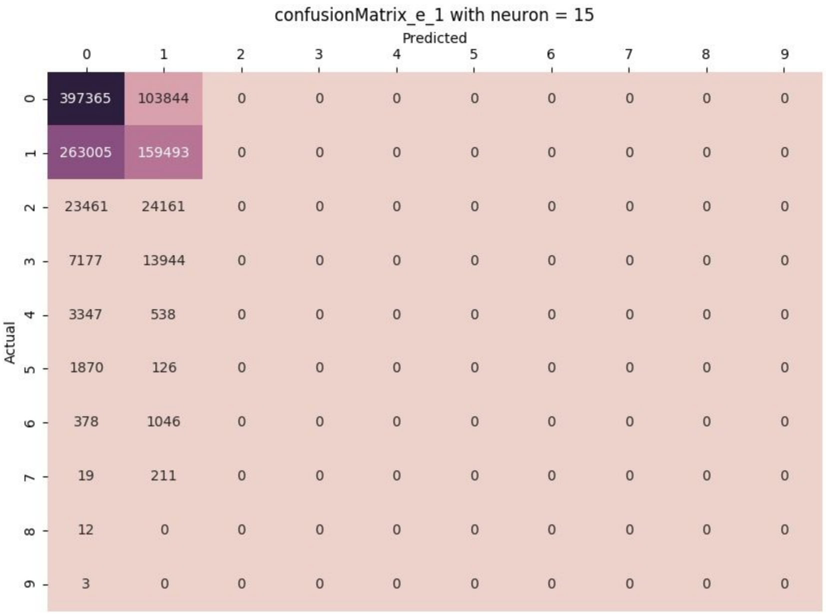
Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	501209	0	0	0	0	0	0	0	0	0
1	422498	0	0	0	0	0	0	0	0	0
2	47622	0	0	0	0	0	0	0	0	0
3	21121	0	0	0	0	0	0	0	0	0
4	3885	0	0	0	0	0	0	0	0	0
5	1996	0	0	0	0	0	0	0	0	0
6	1424	0	0	0	0	0	0	0	0	0
7	230	0	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

For 10 neurons in two layer


confusionMatrix_e_1 with neuron = 10

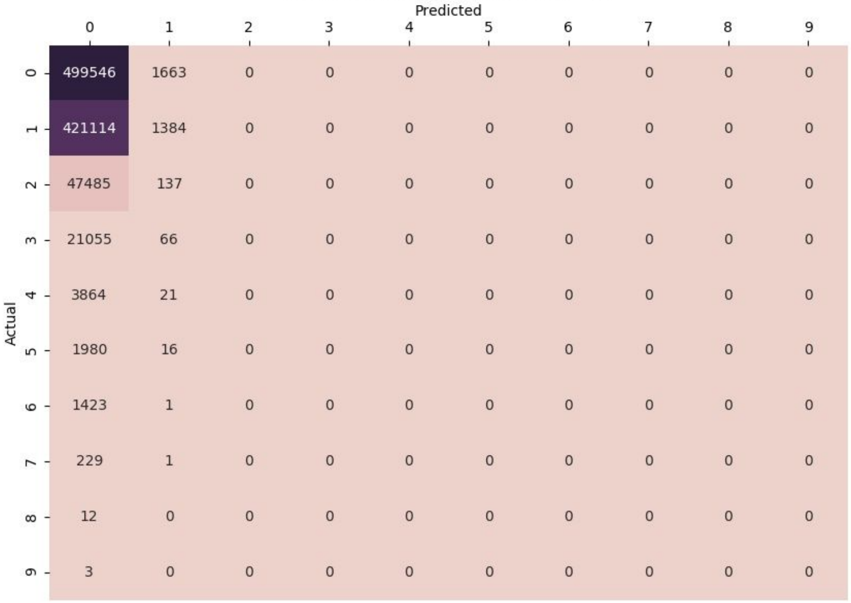
	0	1	2	3	4	5	6	7	8	9
0	452622	48587	0	0	0	0	0	0	0	0
1	371073	51425	0	0	0	0	0	0	0	0
2	40714	6908	0	0	0	0	0	0	0	0
3	17735	3386	0	0	0	0	0	0	0	0
4	3366	519	0	0	0	0	0	0	0	0
5	1812	184	0	0	0	0	0	0	0	0
6	1135	289	0	0	0	0	0	0	0	0
7	182	48	0	0	0	0	0	0	0	0
8	11	1	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

For 15 neurons in two layer


confusionMatrix_e_1 with neuron = 15

	0	1	2	3	4	5	6	7	8	9
0	397365	103844	0	0	0	0	0	0	0	0
1	263005	159493	0	0	0	0	0	0	0	0
2	23461	24161	0	0	0	0	0	0	0	0
3	7177	13944	0	0	0	0	0	0	0	0
4	3347	538	0	0	0	0	0	0	0	0
5	1870	126	0	0	0	0	0	0	0	0
6	378	1046	0	0	0	0	0	0	0	0
7	19	211	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

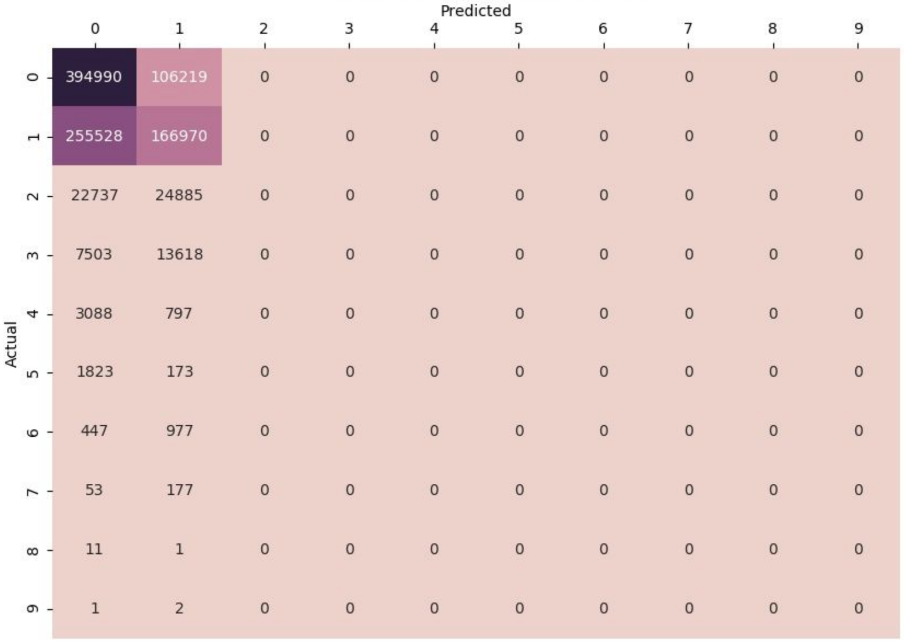
For 20 neurons in two layer



confusionMatrix_e_0 with neuron = 20

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	499546	1663	0	0	0	0	0	0	0	0
1	421114	1384	0	0	0	0	0	0	0	0
2	47485	137	0	0	0	0	0	0	0	0
3	21055	66	0	0	0	0	0	0	0	0
4	3864	21	0	0	0	0	0	0	0	0
5	1980	16	0	0	0	0	0	0	0	0
6	1423	1	0	0	0	0	0	0	0	0
7	229	1	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

For 25 neurons in two layer



confusionMatrix_e_1 with neuron = 25

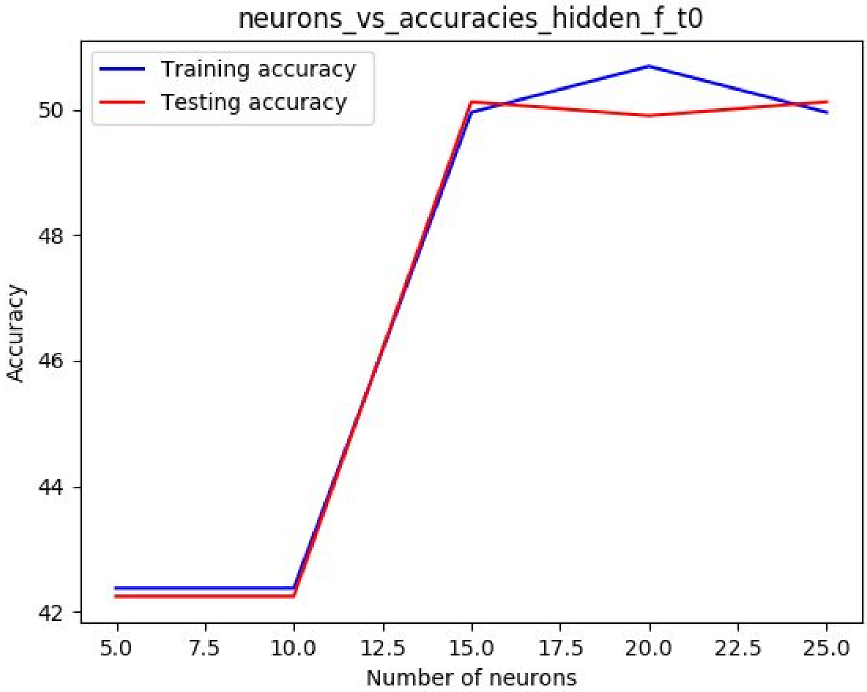
Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	394990	106219	0	0	0	0	0	0	0	0
1	255528	166970	0	0	0	0	0	0	0	0
2	22737	24885	0	0	0	0	0	0	0	0
3	7503	13618	0	0	0	0	0	0	0	0
4	3088	797	0	0	0	0	0	0	0	0
5	1823	173	0	0	0	0	0	0	0	0
6	447	977	0	0	0	0	0	0	0	0
7	53	177	0	0	0	0	0	0	0	0
8	11	1	0	0	0	0	0	0	0	0
9	1	2	0	0	0	0	0	0	0	0

F. Adaptive Learning Rate with Relu

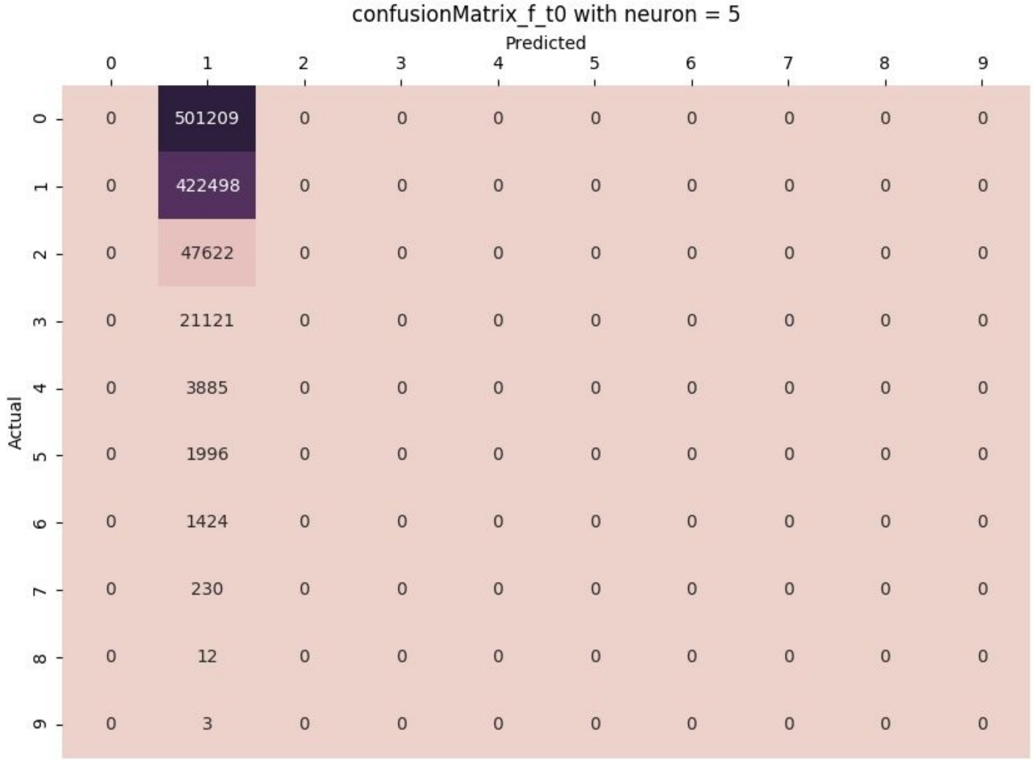
There wasn’t any improvement in the accuracy when adaptive learning with tol = 10 ** -4 was used. Some accuracies remained same as earlier while some became even worse.
Single and Double hidden layer, sigmoid activation function , adaptive learning rate
The neural network was tested with a single hidden layer then double hidden layer and by varying number of units in that layer.
Number of neurons: [5, 10, 15, 20, 25]
Eta = 0.1
Stopping criteria :
Epochs = 1500

i) Single Hidden Layer

Following are testing and training accuracies:
Neurons = [5, 10, 15, 20, 25],
{'test': [42.2498, 42.2498, 50.1209,49.8976, 50.1209],
'train': [42.37904838064774, 42.37904838064774 , 49.95201919232307, 50.683726509396244, 49.95201919232307]
Execution time for training = [1583.501597881, 2836.295443058, 4570.06251812 , 3842.302116871, 3839.308110952]

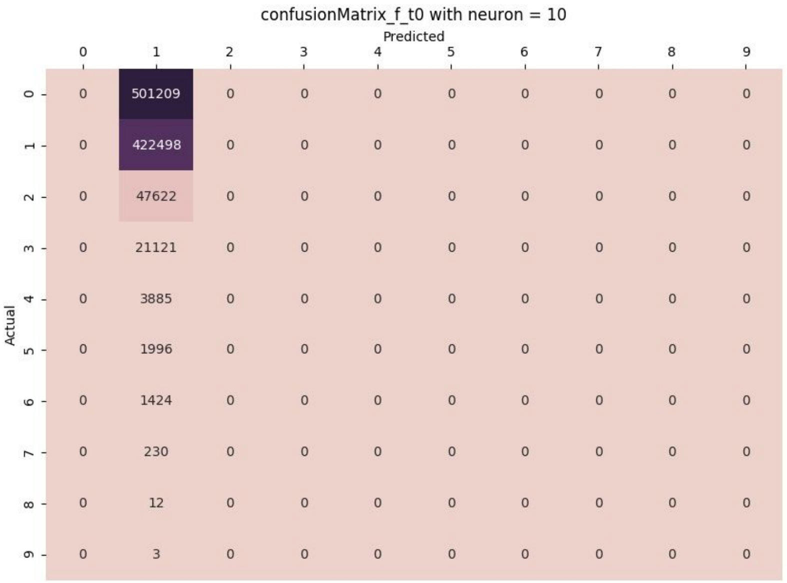

neurons_vs_accuracies_hidden_f_t0

For 5 layers :


confusionMatrix_f_t0 with neuron = 5

	0	1	2	3	4	5	6	7	8	9
0	0	501209	0	0	0	0	0	0	0	0
1	0	422498	0	0	0	0	0	0	0	0
2	0	47622	0	0	0	0	0	0	0	0
3	0	21121	0	0	0	0	0	0	0	0
4	0	3885	0	0	0	0	0	0	0	0
5	0	1996	0	0	0	0	0	0	0	0
6	0	1424	0	0	0	0	0	0	0	0
7	0	230	0	0	0	0	0	0	0	0
8	0	12	0	0	0	0	0	0	0	0
9	0	3	0	0	0	0	0	0	0	0

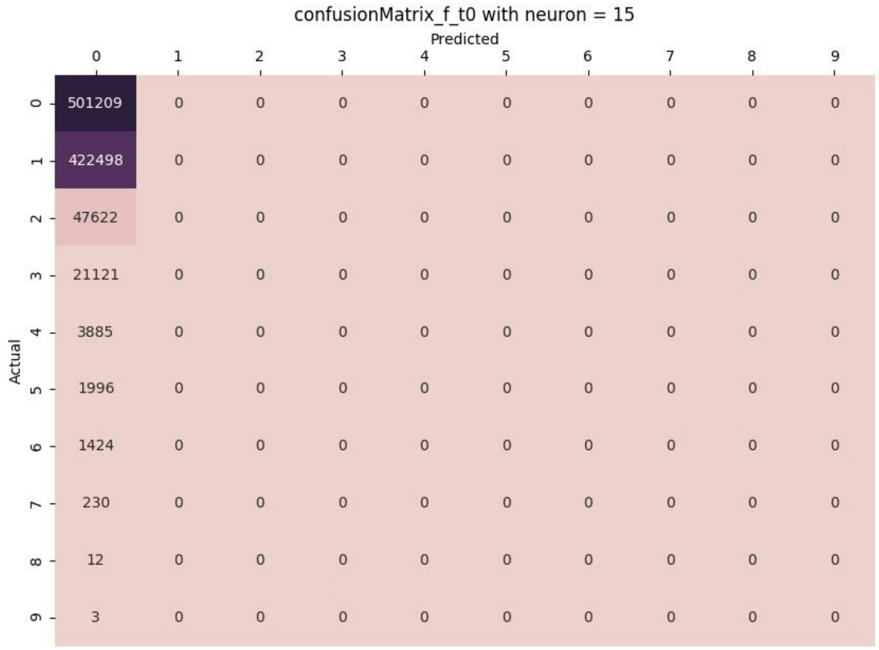
For 10 Layers:



confusionMatrix_f_t0 with neuron = 10

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	0	501209	0	0	0	0	0	0	0	0
1	0	422498	0	0	0	0	0	0	0	0
2	0	47622	0	0	0	0	0	0	0	0
3	0	21121	0	0	0	0	0	0	0	0
4	0	3885	0	0	0	0	0	0	0	0
5	0	1996	0	0	0	0	0	0	0	0
6	0	1424	0	0	0	0	0	0	0	0
7	0	230	0	0	0	0	0	0	0	0
8	0	12	0	0	0	0	0	0	0	0
9	0	3	0	0	0	0	0	0	0	0

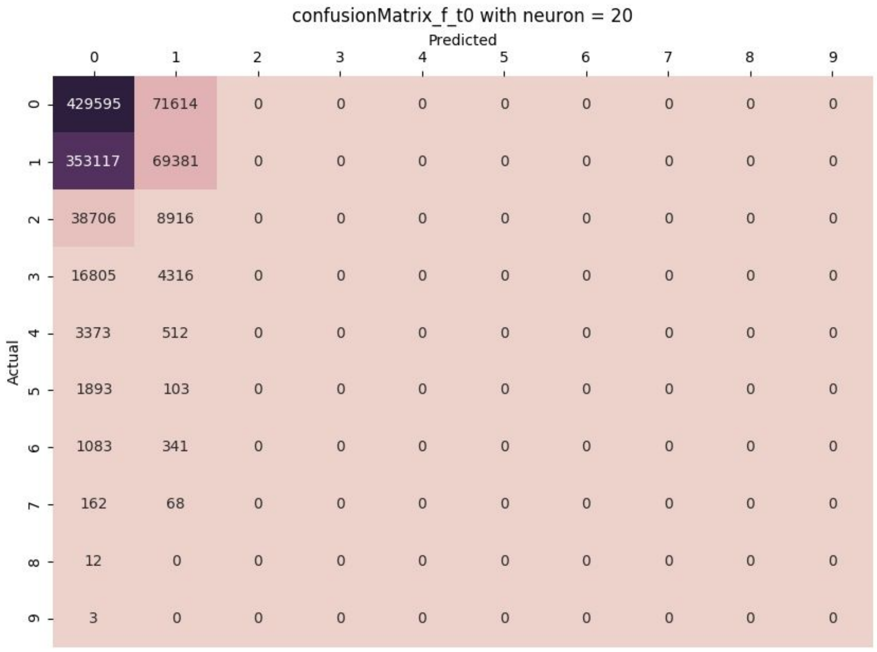
For 15 Layers:



confusionMatrix_f_t0 with neuron = 15

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	501209	0	0	0	0	0	0	0	0	0
1	422498	0	0	0	0	0	0	0	0	0
2	47622	0	0	0	0	0	0	0	0	0
3	21121	0	0	0	0	0	0	0	0	0
4	3885	0	0	0	0	0	0	0	0	0
5	1996	0	0	0	0	0	0	0	0	0
6	1424	0	0	0	0	0	0	0	0	0
7	230	0	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

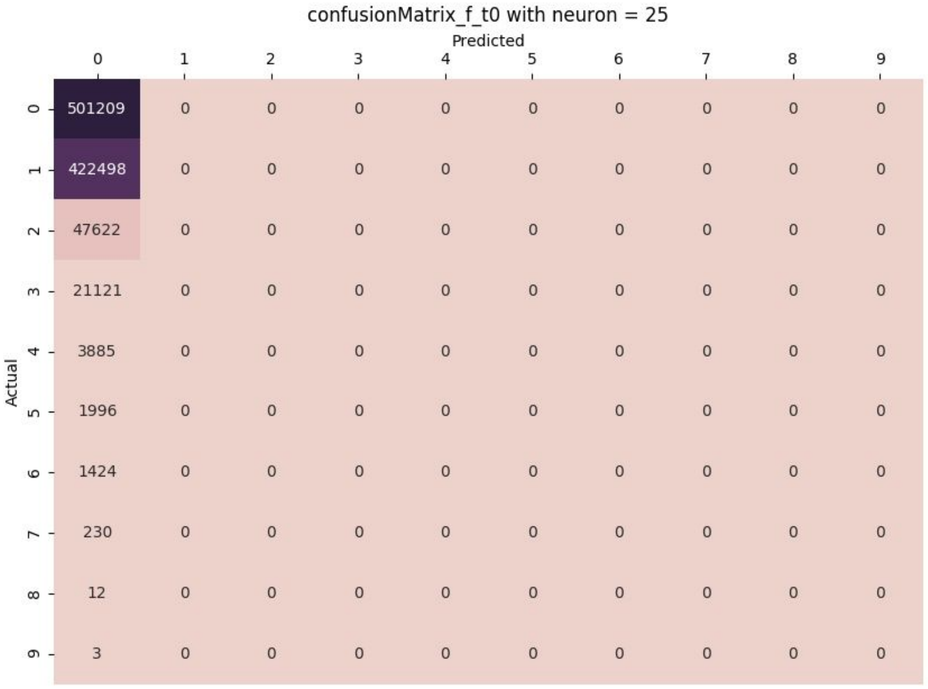
For 20 Layers:



confusionMatrix_f_t0 with neuron = 20

	0	1	2	3	4	5	6	7	8	9
0	429595	71614	0	0	0	0	0	0	0	0
1	353117	69381	0	0	0	0	0	0	0	0
2	38706	8916	0	0	0	0	0	0	0	0
3	16805	4316	0	0	0	0	0	0	0	0
4	3373	512	0	0	0	0	0	0	0	0
5	1893	103	0	0	0	0	0	0	0	0
6	1083	341	0	0	0	0	0	0	0	0
7	162	68	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

For 25 Layers:

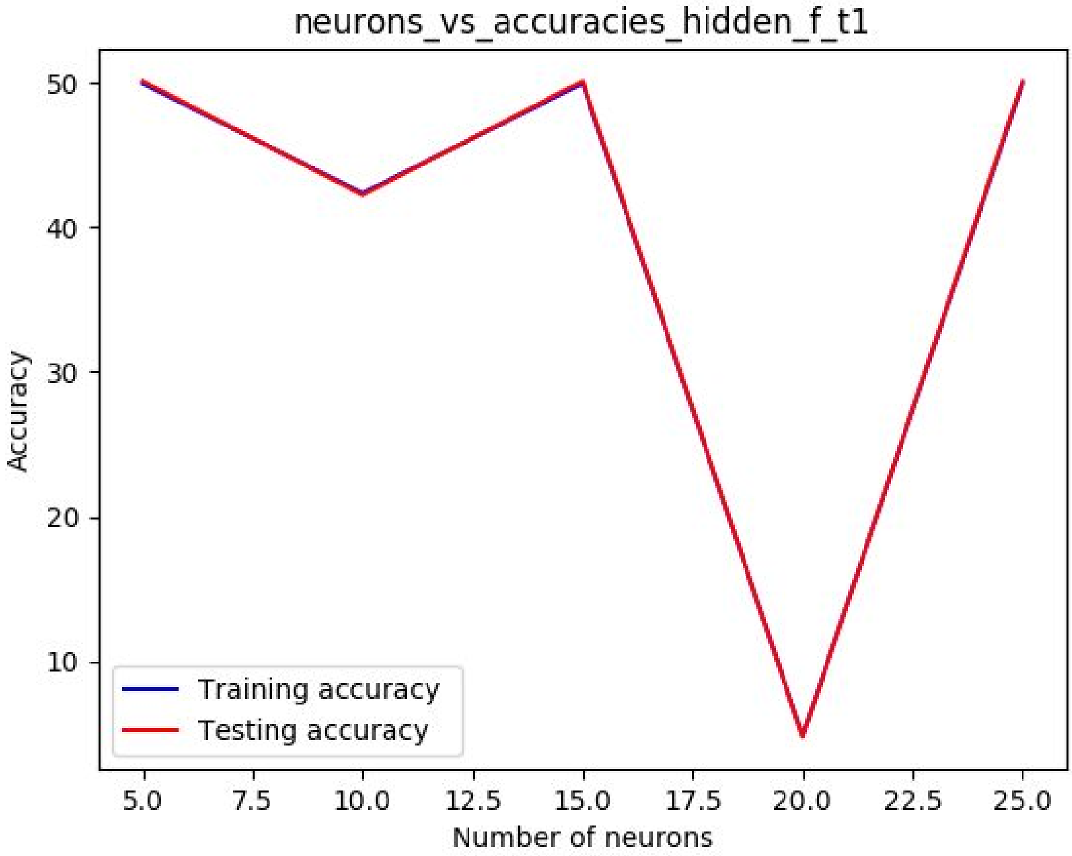


confusionMatrix_f_t0 with neuron = 25

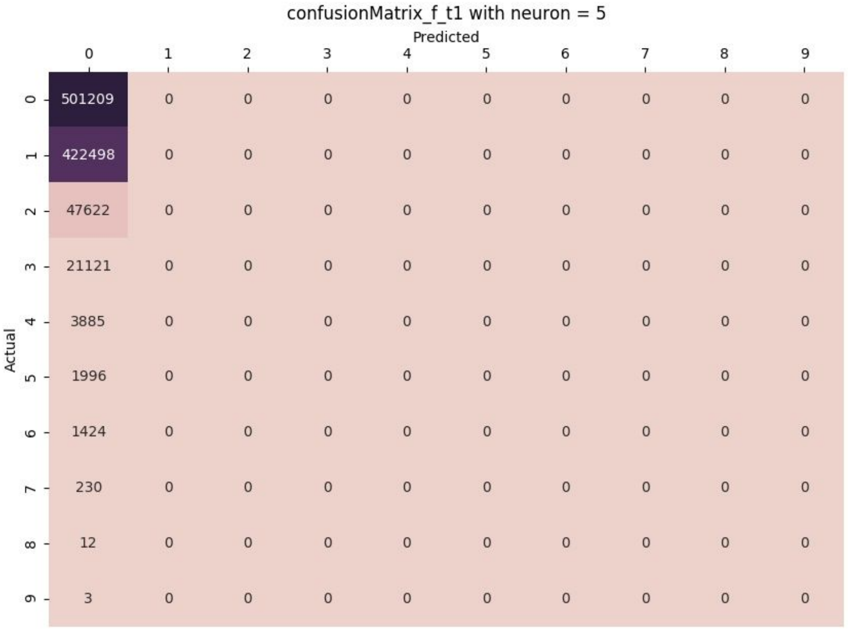
	0	1	2	3	4	5	6	7	8	9
0	501209	0	0	0	0	0	0	0	0	0
1	422498	0	0	0	0	0	0	0	0	0
2	47622	0	0	0	0	0	0	0	0	0
3	21121	0	0	0	0	0	0	0	0	0
4	3885	0	0	0	0	0	0	0	0	0
5	1996	0	0	0	0	0	0	0	0	0
6	1424	0	0	0	0	0	0	0	0	0
7	230	0	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

ii) Two Hidden Layers:

Following are testing and training accuracies:
Neurons = [5, 10, 15, 20, 25],
{'test': [50.1209,42.2498, 50.1209, 4.822071715313875, 4.7622, 50.1209
],
'train': [49.95201919232307, 42.37904838064774, 49.95201919232307,
49.95201919232307]
Execution time for training = [1825.282280922, 2589.579301119,
2821.268438816, 3837.023085117, 4831.529021025]



For 5 layers:

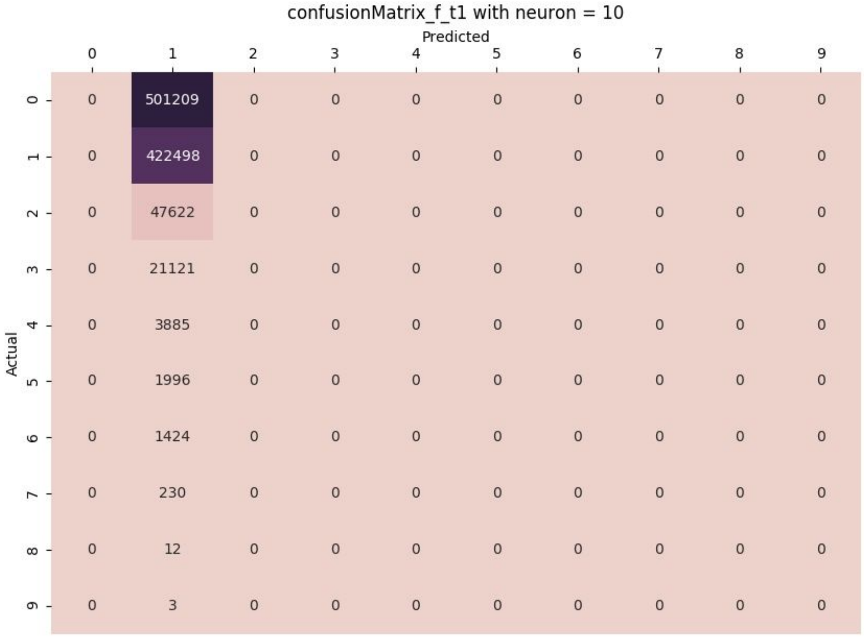


confusionMatrix_f_t1 with neuron = 5

Predicted

Actual	0	1	2	3	4	5	6	7	8	9
0	501209	0	0	0	0	0	0	0	0	0
1	422498	0	0	0	0	0	0	0	0	0
2	47622	0	0	0	0	0	0	0	0	0
3	21121	0	0	0	0	0	0	0	0	0
4	3885	0	0	0	0	0	0	0	0	0
5	1996	0	0	0	0	0	0	0	0	0
6	1424	0	0	0	0	0	0	0	0	0
7	230	0	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

For 10 Layers:

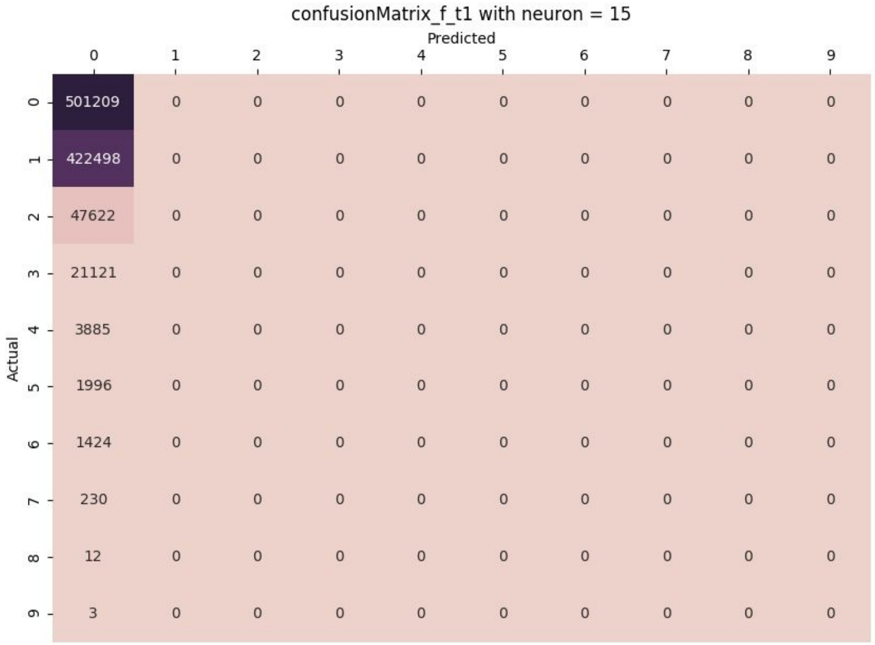


confusionMatrix_f_t1 with neuron = 10

Predicted

Actual	0	1	2	3	4	5	6	7	8	9
0	0	501209	0	0	0	0	0	0	0	0
1	0	422498	0	0	0	0	0	0	0	0
2	0	47622	0	0	0	0	0	0	0	0
3	0	21121	0	0	0	0	0	0	0	0
4	0	3885	0	0	0	0	0	0	0	0
5	0	1996	0	0	0	0	0	0	0	0
6	0	1424	0	0	0	0	0	0	0	0
7	0	230	0	0	0	0	0	0	0	0
8	0	12	0	0	0	0	0	0	0	0
9	0	3	0	0	0	0	0	0	0	0

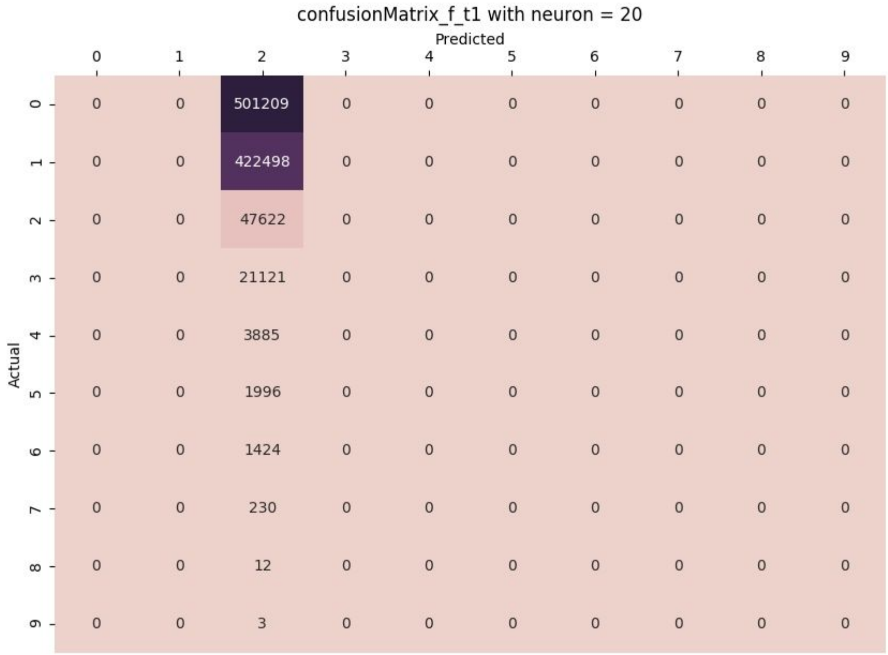
For 15 Layers:



confusionMatrix_f_t1 with neuron = 15

	0	1	2	3	4	5	6	7	8	9
0	501209	0	0	0	0	0	0	0	0	0
1	422498	0	0	0	0	0	0	0	0	0
2	47622	0	0	0	0	0	0	0	0	0
3	21121	0	0	0	0	0	0	0	0	0
4	3885	0	0	0	0	0	0	0	0	0
5	1996	0	0	0	0	0	0	0	0	0
6	1424	0	0	0	0	0	0	0	0	0
7	230	0	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0

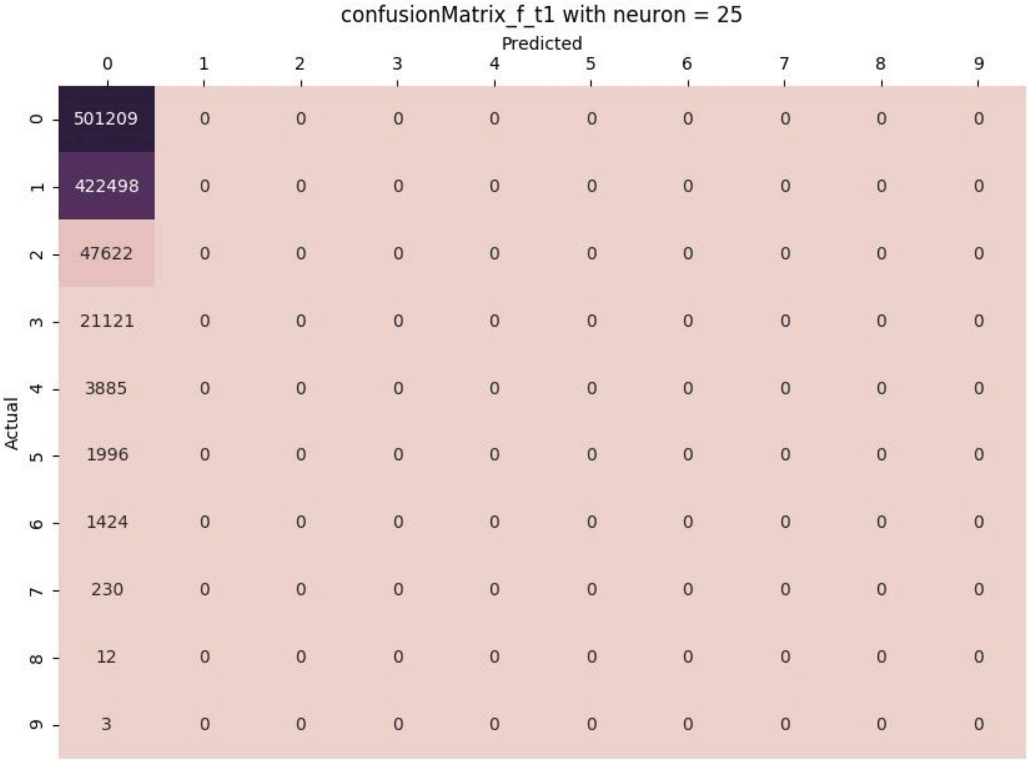
For 20 Layers:



confusionMatrix_f_t1 with neuron = 20

	0	1	2	3	4	5	6	7	8	9
0	0	0	501209	0	0	0	0	0	0	0
1	0	0	422498	0	0	0	0	0	0	0
2	0	0	47622	0	0	0	0	0	0	0
3	0	0	21121	0	0	0	0	0	0	0
4	0	0	3885	0	0	0	0	0	0	0
5	0	0	1996	0	0	0	0	0	0	0
6	0	0	1424	0	0	0	0	0	0	0
7	0	0	230	0	0	0	0	0	0	0
8	0	0	12	0	0	0	0	0	0	0
9	0	0	3	0	0	0	0	0	0	0

For 25 Layers:



confusionMatrix_f_t1 with neuron = 25

Predicted

Actual	0	1	2	3	4	5	6	7	8	9
0	501209	0	0	0	0	0	0	0	0	0
1	422498	0	0	0	0	0	0	0	0	0
2	47622	0	0	0	0	0	0	0	0	0
3	21121	0	0	0	0	0	0	0	0	0
4	3885	0	0	0	0	0	0	0	0	0
5	1996	0	0	0	0	0	0	0	0	0
6	1424	0	0	0	0	0	0	0	0	0
7	230	0	0	0	0	0	0	0	0	0
8	12	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0