COL 774
ASSIGNMENT 3

QUESTION 1 (Decision Trees and Random Forests)

1A

To run :

```
python3 do_question1a.py 1  "decision_tree/decision_tree/train.csv"
"decision_tree/decision_tree/val.csv" "decision_tree/decision_tree/test.csv" "output1a.txt"
```

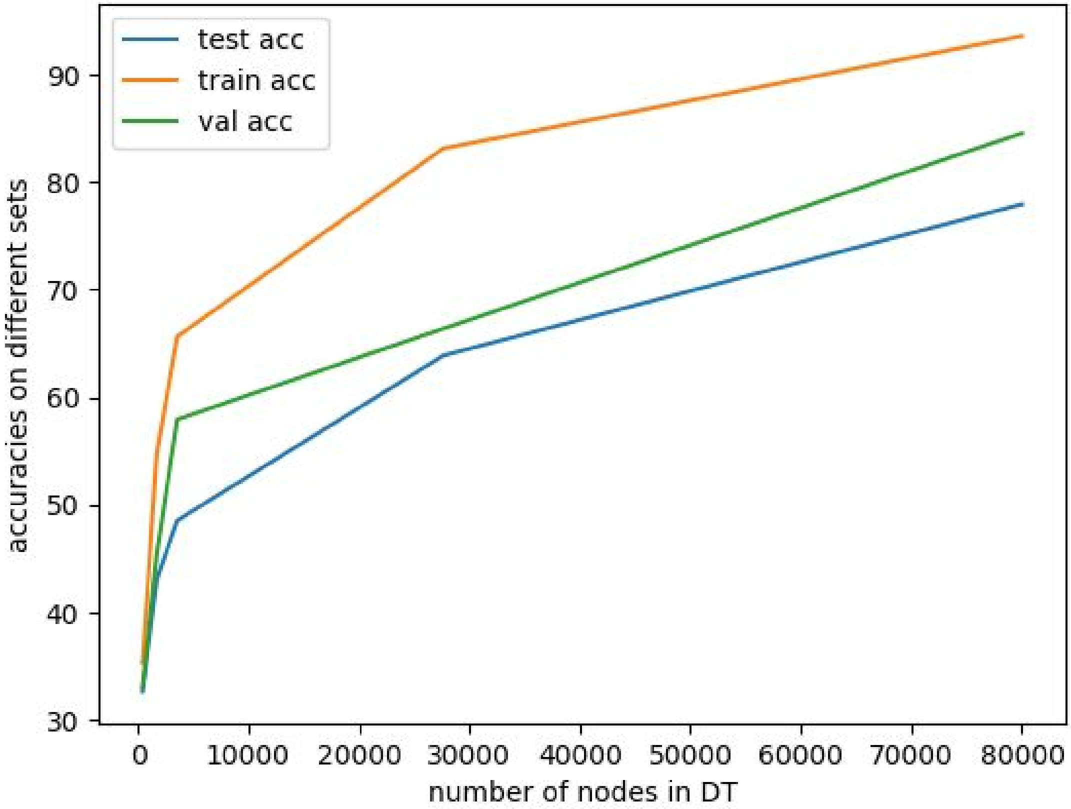
[python3 check1a.py] : for graph

At each node, I select the attribute which results in maximum decrease in the entropy of the class variable (i.e. has the highest mutual information with respect to the class variable). Entropy is calculated using p*log(p).

In this part, there is no pruning. The number of nodes expanded and corresponding accuracies obtained are :

Number of nodes = [412,1708,3558,27654,82046]
Test set accuracy = [32.69,43.07,48.56,63.88,77.93]
Training set accuracy = [35.32,54.76,65.65,83.12,93.56]
Validation set accuracy = [33.09,45.44,57.94,66.38,84.53]

Variation of accuracy with increasing number of nodes in the Decision Tree : (1a.png)



1B

To run :

```
python3 do_question1b.py 2 "decision_tree/decision_tree/train.csv"
"decision_tree/decision_tree/val.csv" "decision_tree/decision_tree/test.csv" "output1b.txt"
```

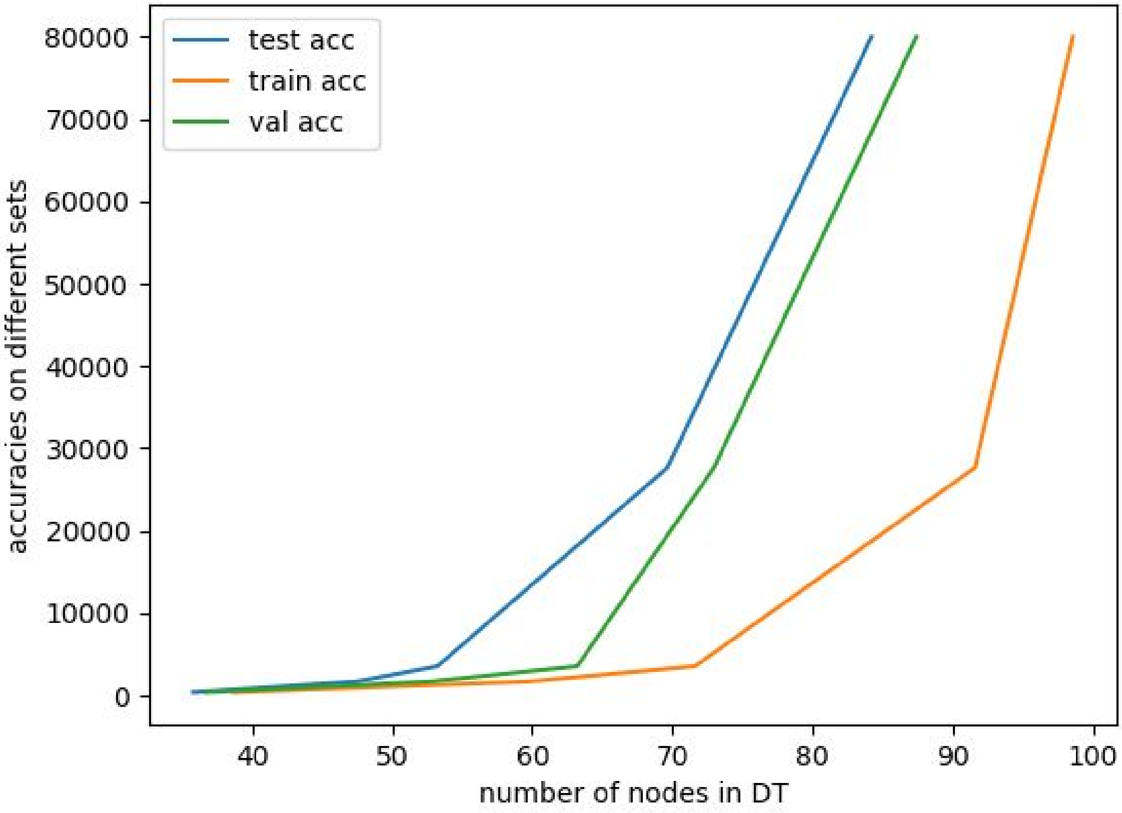
[python3 check1b.py] : for graph

In this part, pruning is applied. First, the decision tree is grown to full size i.e. 82046 nodes and then post-pruning is performed on the validation set. In post-pruning, I pruned the nodes of the tree (and sub-tree below them) by iteratively picking a node to prune so that resultant tree gives maximum increase in accuracy on the validation set. So, among all the nodes in the tree, I pruned that particular node such that pruning it (and the subtree below it) resulted in maximum increase in accuracy over the validation set. This was repeated until any further pruning did not improve the accuracy over the validation set.

Number of nodes expanded and corresponding accuracies :

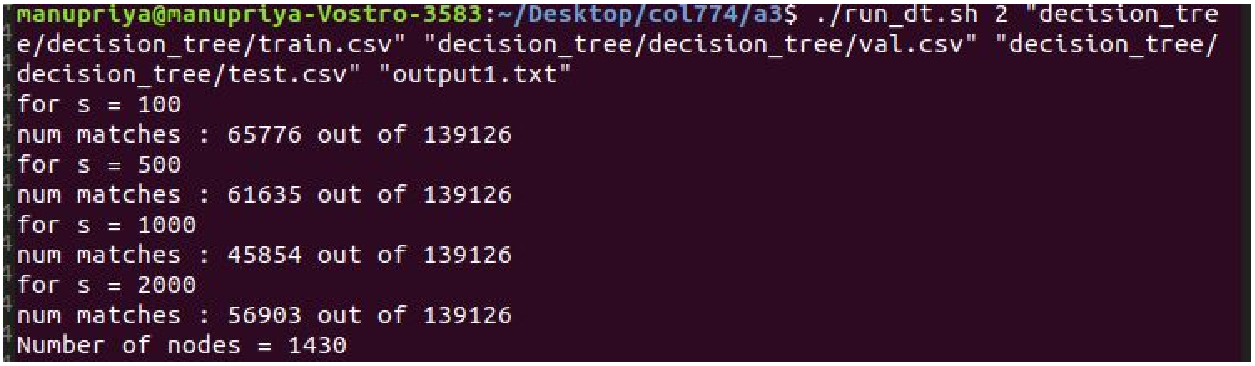
Number of nodes = [412,1708,3558,27654,80000]
Test set accuracy = [35.76,47.45,53.22,69.65,84.23]
Training set accuracy = [38.65,59.75,71.61,91.62,98.60]
Validation set accuracy = [36.69,52.74,63.21,72.97,87.43]

Plot (1b.png) :



We can notice that post pruning, all accuracies improve but it takes considerably more time. The number of nodes required is also less

Logs :



```
manupriya@manupriya-Vostro-3583:~/Desktop/col774/a3$ ./run_dt.sh 2 "decision_tre
e/decision_tree/train.csv" "decision_tree/decision_tree/val.csv" "decision_tree/
decision_tree/test.csv" "output1.txt"
for s = 100
num matches : 65776 out of 139126
for s = 500
num matches : 61635 out of 139126
for s = 1000
num matches : 45854 out of 139126
for s = 2000
num matches : 56903 out of 139126
Number of nodes = 1430
```

1C

To run :
python3 do_question1c.py "decision_tree/decision_tree/train.csv" "decision_tree/decision_tree/val.csv" "decision_tree/decision_tree/test.csv" "output1c.txt"

The predictions for test set are in output1c.txt

The optimal parameters obtained after thorough testing are :
N_estimators = 450
Max_features = 0.7
Min_samples_split = 2

After running on these :
Accuracy on training set = 99.43 %
Accuracy on test set = 96.65 %
Accuracy on validation set = 96.78 %

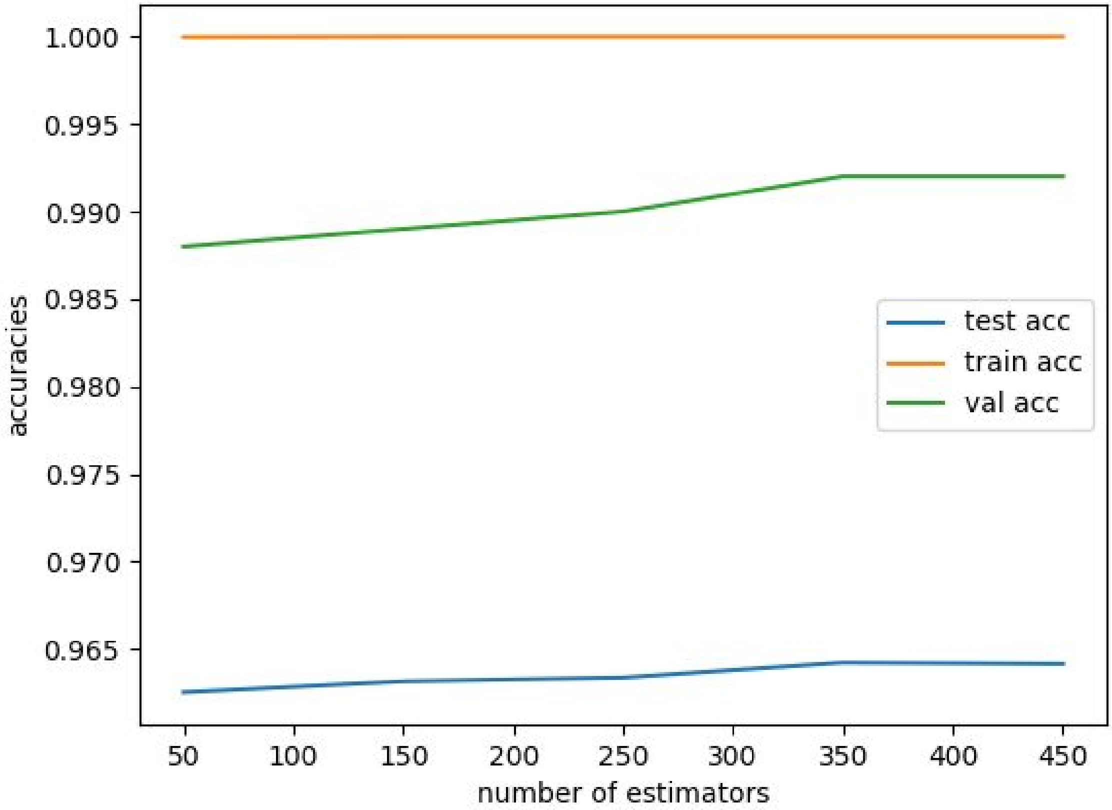
WE notice that the accuracies are higher than the ones obtained from the Decision Tree and time taken by both techniques is almost same. So, this library utility outperforms the the Decision Tree I built.

1D

To run :
python3 do_question1d.py "decision_tree/decision_tree/train.csv" "decision_tree/decision_tree/val.csv" "decision_tree/decision_tree/test.csv" "output1d.txt"

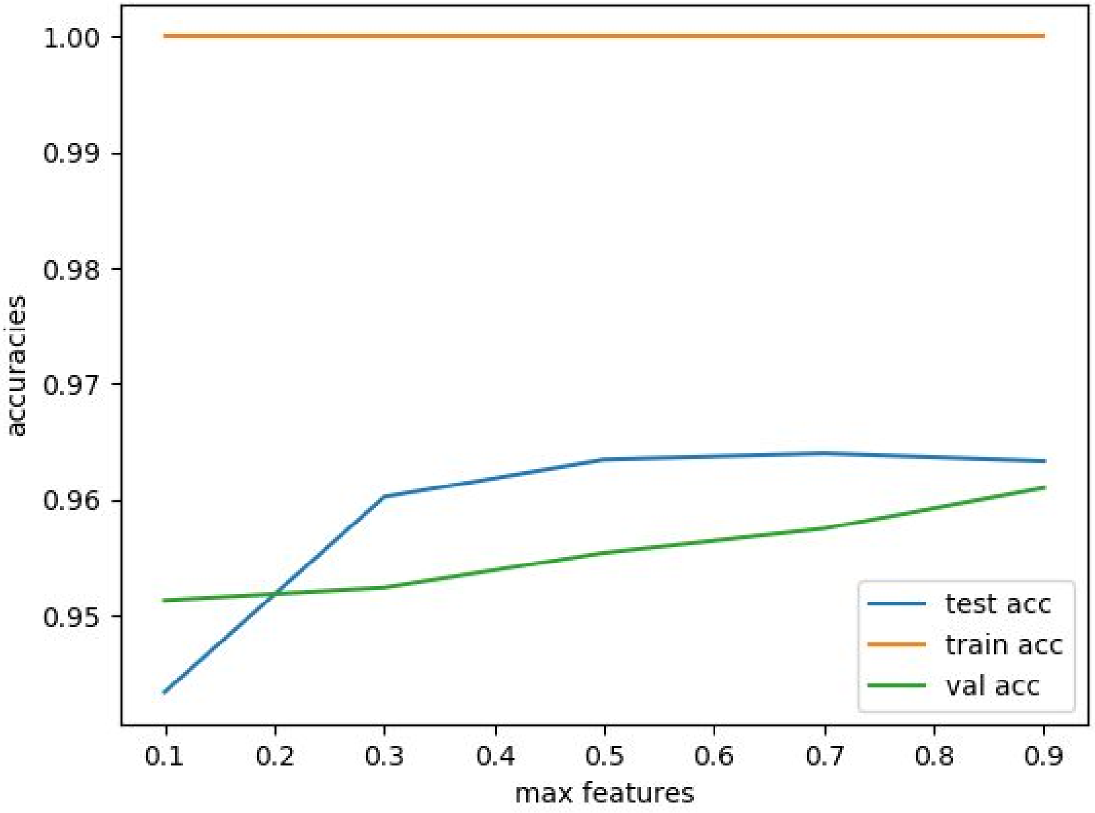
Graphs : [python3 check1dp1.py] , [python3 check1dp2.py] , [python3 check1dp3.py]

Varying number of estimators :



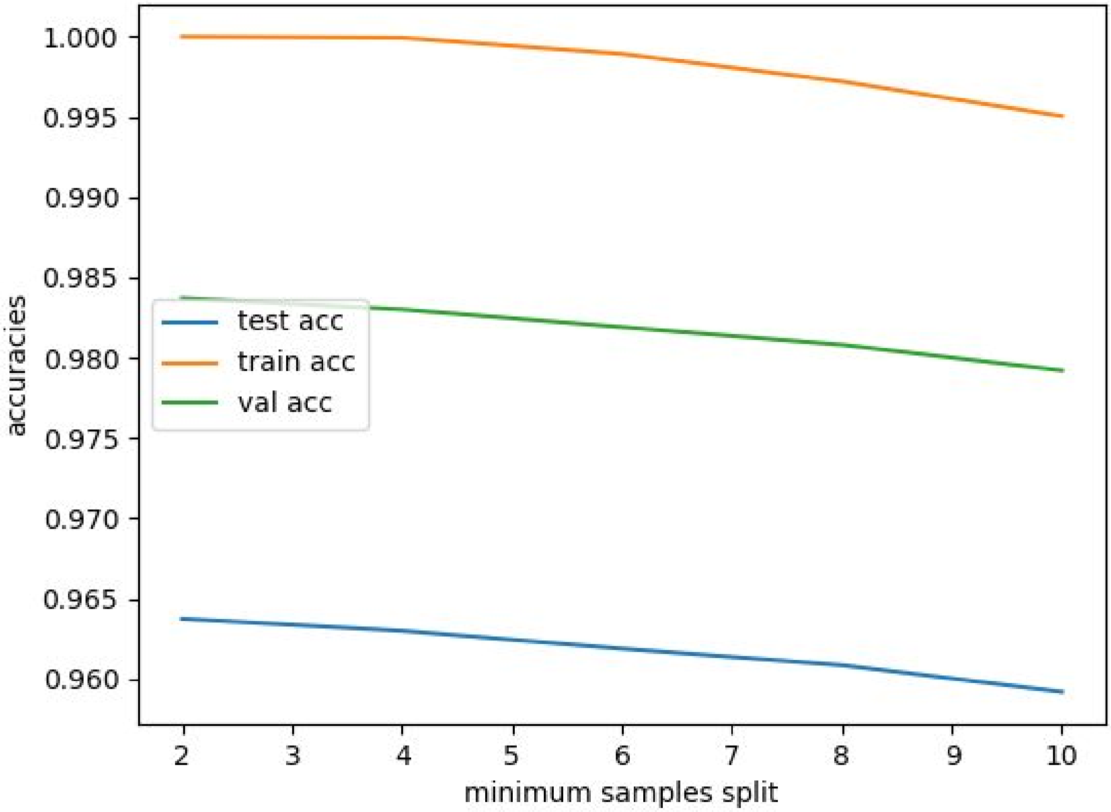
Hence, the classifier is not very sensitive to the number of estimators.

Varying Max Features :



Hence, the classifier is a little sensitive to the parameter of max_features.

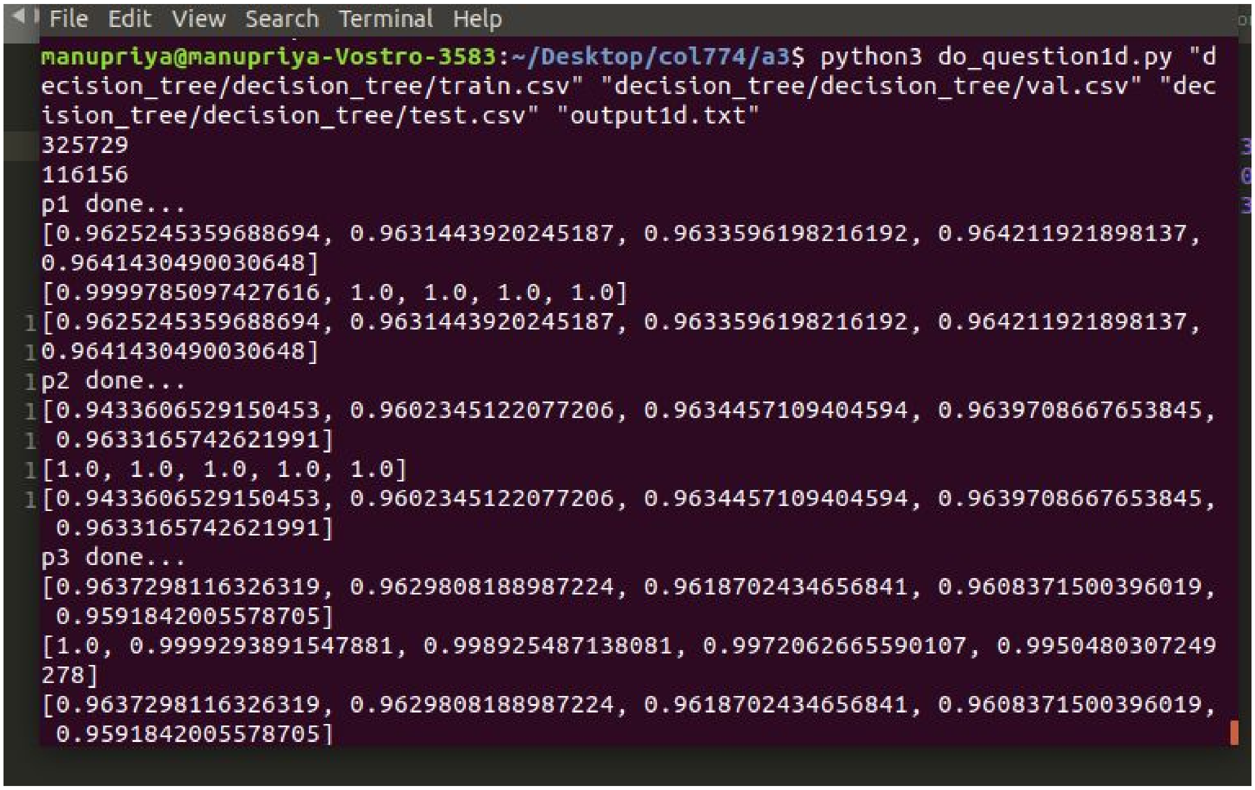
Varying Minimum Samples Split :



Hence, the classifier is very sensitive to the parameter of min_samples_split.

So, min_samples_split affects the accuracy and hence the model, the most.

Logs :



```
manupriya@manupriya-Vostro-3583:~/Desktop/col774/a3$ python3 do_question1d.py "d
ecision_tree/decision_tree/train.csv" "decision_tree/decision_tree/val.csv" "dec
ision_tree/decision_tree/test.csv" "output1d.txt"
325729
116156
p1 done...
[0.9625245359688694, 0.9631443920245187, 0.9633596198216192, 0.964211921898137,
0.9641430490030648]
[0.9999785097427616, 1.0, 1.0, 1.0, 1.0]
[0.9625245359688694, 0.9631443920245187, 0.9633596198216192, 0.964211921898137,
0.9641430490030648]
p2 done...
[0.9433606529150453, 0.9602345122077206, 0.9634457109404594, 0.9639708667653845,
 0.9633165742621991]
[1.0, 1.0, 1.0, 1.0, 1.0]
[0.9433606529150453, 0.9602345122077206, 0.9634457109404594, 0.9639708667653845,
 0.9633165742621991]
p3 done...
[0.9637298116326319, 0.9629808188987224, 0.9618702434656841, 0.9608371500396019,
 0.9591842005578705]
[1.0, 0.9999293891547881, 0.998925487138081, 0.9972062665590107, 0.9950480307249
278]
[0.9637298116326319, 0.9629808188987224, 0.9618702434656841, 0.9608371500396019,
 0.9591842005578705]
```

QUESTION 2 (Neural Networks)

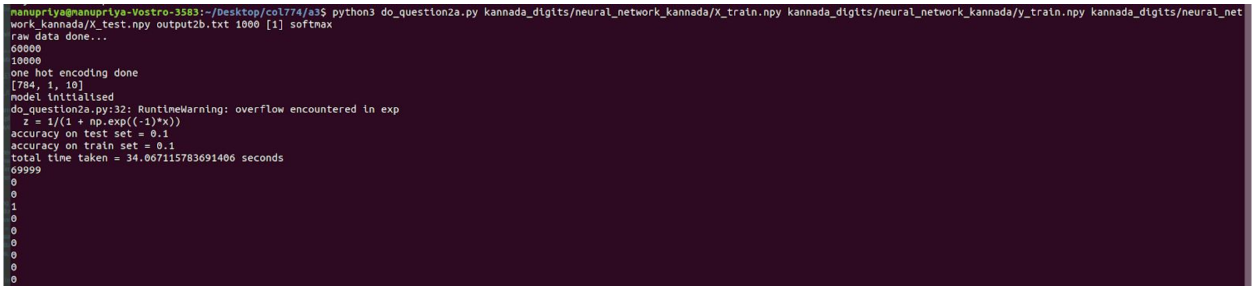
2A

To run :
python3 do_question2a.py kannada_digits/neural_network_kannada/X_train.npy kannada_digits/neural_network_kannada/y_train.npy kannada_digits/neural_network_kannada/X_test.npy output2a.txt 1000 [100] softmax

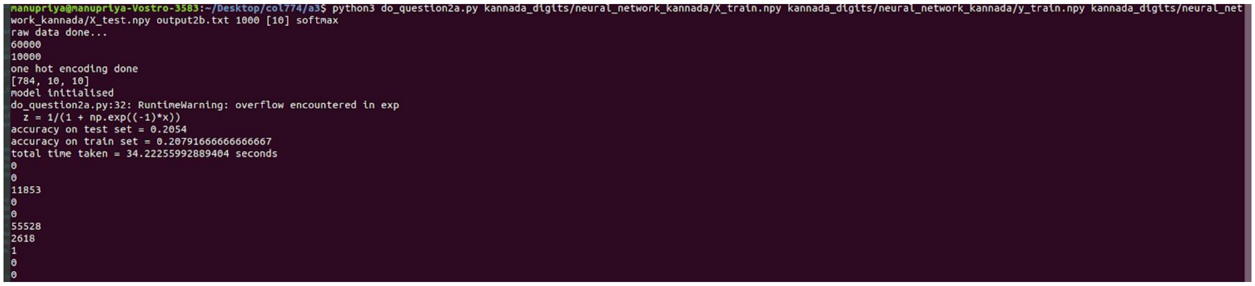
The predicted labels are in output2a.txt

The inferred class label is simply the label having the highest probability as output by the network.
Learning rate is set at 0.1 for the autograder but actually on my machine I used 0.001.
Number of epochs set at 100.
Stopping criteria : J-J’ <= 0.05

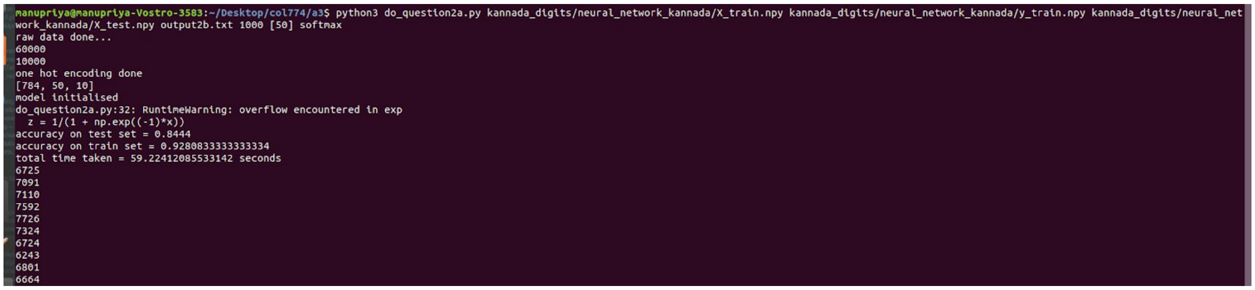
Screenshots for different hidden lists :



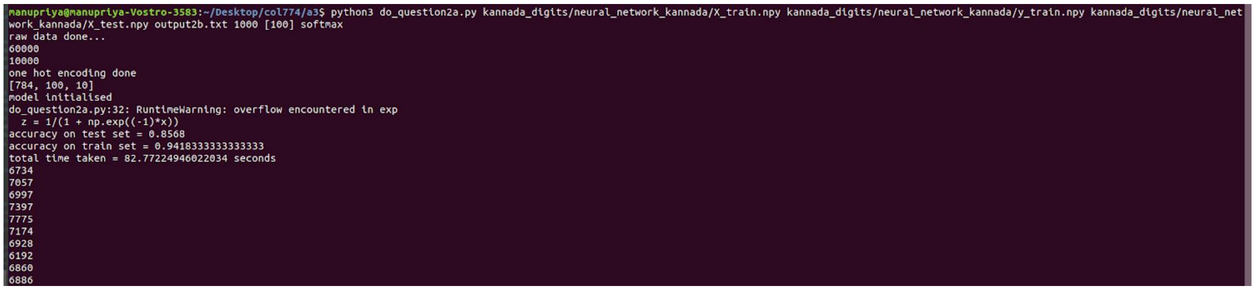
```
manupriya@manupriya-Vostro-3583:~/Desktop/col774/a3$ python3 do_question2a.py kannada_digits/neural_network_kannada/X_train.npy kannada_digits/neural_network_kannada/y_train.npy kannada_digits/neural_net
work_kannada/X_test.npy output2b.txt 1000 [1] softmax
raw data done...
60000
10000
one hot encoding done
[784, 1, 10]
model initialised
do_question2a.py:32: RuntimeWarning: overflow encountered in exp
  z = 1/(1 + np.exp((-1)*x))
accuracy on test set = 0.1
accuracy on train set = 0.1
total time taken = 34.067115783691406 seconds
69999
0
0
1
0
0
0
0
0
0
```



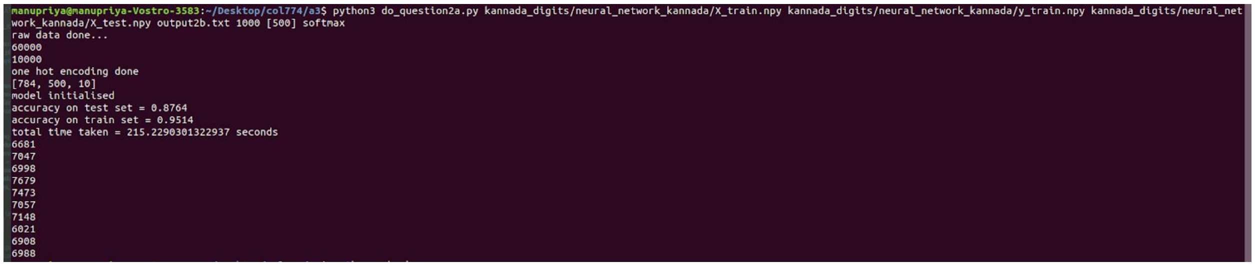
```
manupriya@manupriya-Vostro-3583:~/Desktop/col774/a3$ python3 do_question2a.py kannada_digits/neural_network_kannada/X_train.npy kannada_digits/neural_network_kannada/y_train.npy kannada_digits/neural_net
work_kannada/X_test.npy output2b.txt 1000 [10] softmax
raw data done...
60000
10000
one hot encoding done
[784, 10, 10]
model initialised
do_question2a.py:32: RuntimeWarning: overflow encountered in exp
  z = 1/(1 + np.exp((-1)*x))
accuracy on test set = 0.2054
accuracy on train set = 0.20791666666666667
total time taken = 34.22255992889404 seconds
0
0
11853
0
0
55528
2618
1
0
0
```



```
manupriya@manupriya-Vostro-3583:~/Desktop/col774/a3$ python3 do_question2a.py kannada_digits/neural_network_kannada/X_train.npy kannada_digits/neural_network_kannada/y_train.npy kannada_digits/neural_net
work_kannada/X_test.npy output2b.txt 1000 [50] softmax
raw data done...
60000
10000
one hot encoding done
[784, 50, 10]
model initialised
do_question2a.py:32: RuntimeWarning: overflow encountered in exp
  z = 1/(1 + np.exp((-1)*x))
accuracy on test set = 0.8444
accuracy on train set = 0.9280833333333334
total time taken = 59.22412085533142 seconds
6725
7091
7110
7592
7726
7324
6724
6243
6801
6664
```



```
manupriya@manupriya-Vostro-3583:~/Desktop/col774/a3$ python3 do_question2a.py kannada_digits/neural_network_kannada/X_train.npy kannada_digits/neural_network_kannada/y_train.npy kannada_digits/neural_net
work_kannada/X_test.npy output2b.txt 1000 [100] softmax
raw data done...
60000
10000
one hot encoding done
[784, 100, 10]
model initialised
do_question2a.py:32: RuntimeWarning: overflow encountered in exp
  z = 1/(1 + np.exp((-1)*x))
accuracy on test set = 0.8568
accuracy on train set = 0.9418333333333333
total time taken = 82.77224946022034 seconds
6734
7057
6997
7397
7775
7174
6928
6192
6860
6886
```



```
manupriya@manupriya-Vostro-3583:~/Desktop/col774/a3$ python3 do_question2a.py kannada_digits/neural_network_kannada/X_train.npy kannada_digits/neural_network_kannada/y_train.npy kannada_digits/neural_net
work_kannada/X_test.npy output2b.txt 1000 [500] softmax
raw data done...
60000
10000
one hot encoding done
[784, 500, 10]
model initialised
accuracy on test set = 0.8764
accuracy on train set = 0.9514
total time taken = 215.2290301322937 seconds
6681
7047
6998
7679
7473
7057
7148
6021
6908
6988
```

2B

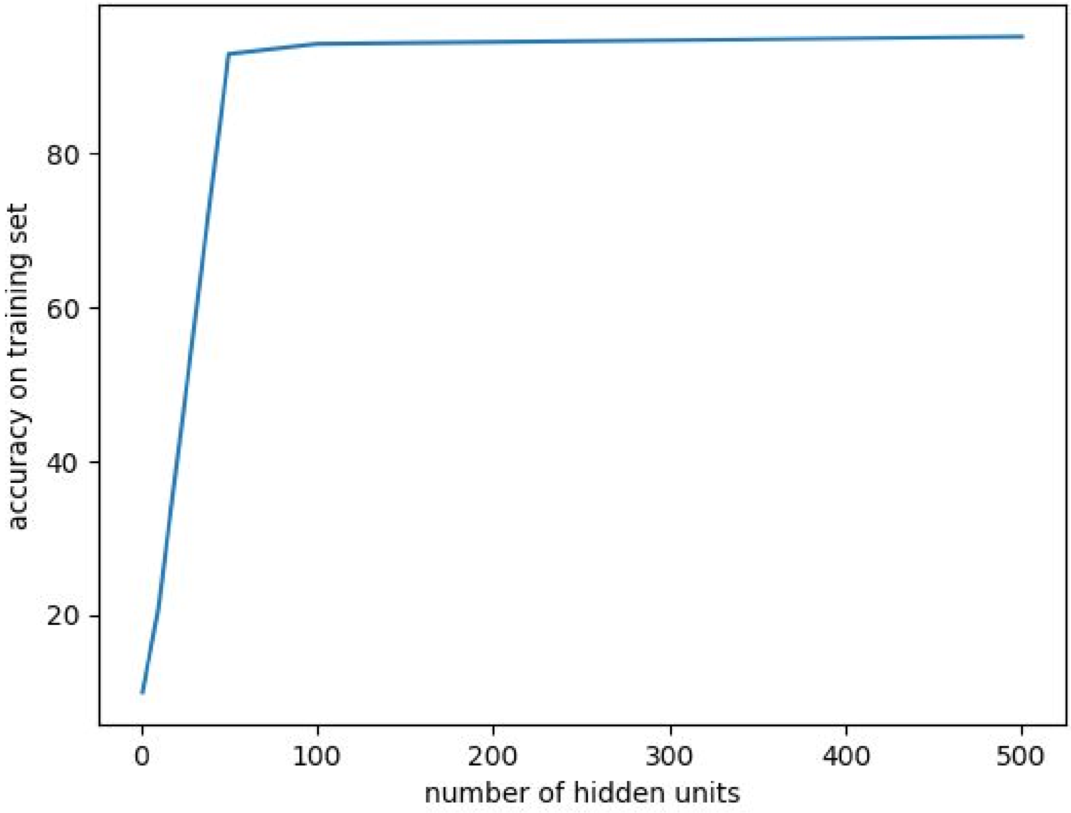
To run :

```
python3 do_question2b.py kannada_digits/neural_network_kannada/X_train.npy
kannada_digits/neural_network_kannada/y_train.npy
kannada_digits/neural_network_kannada/X_test.npy output2b.txt 1000 [100] softmax
```

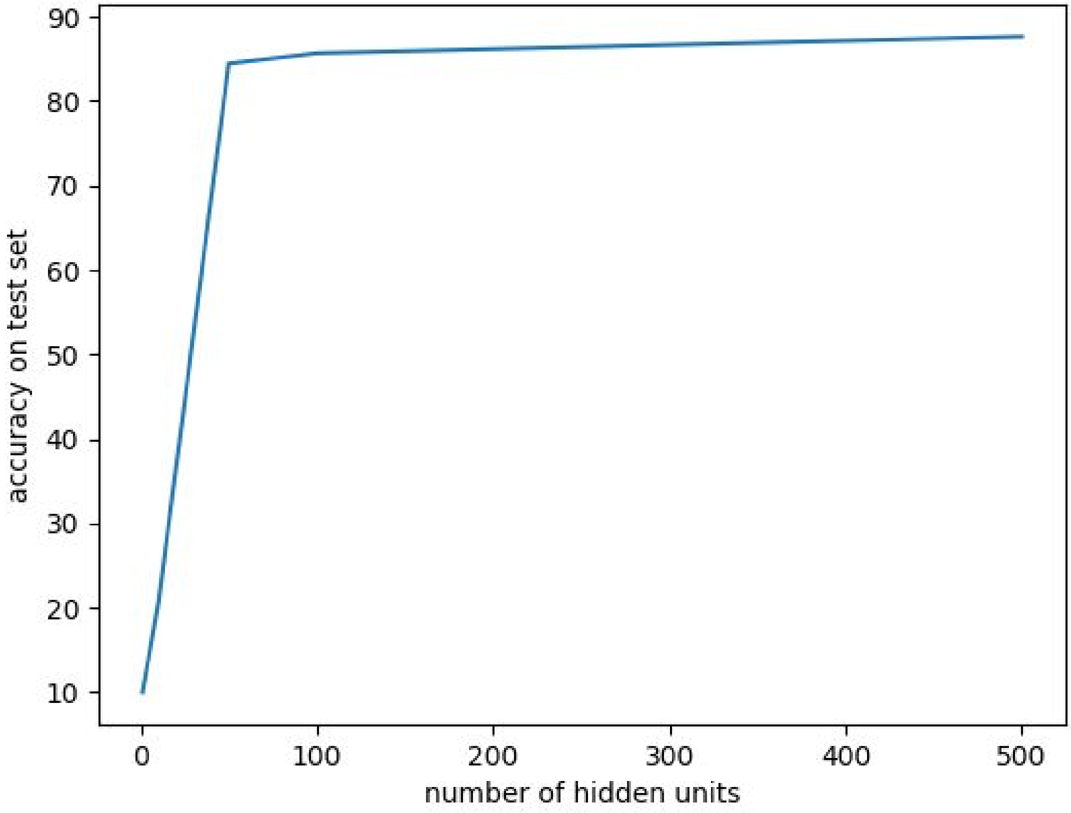
Learning fixed at 0.001

Using 1, 10, 50, 100 and 500 hidden units in the single hidden layer :

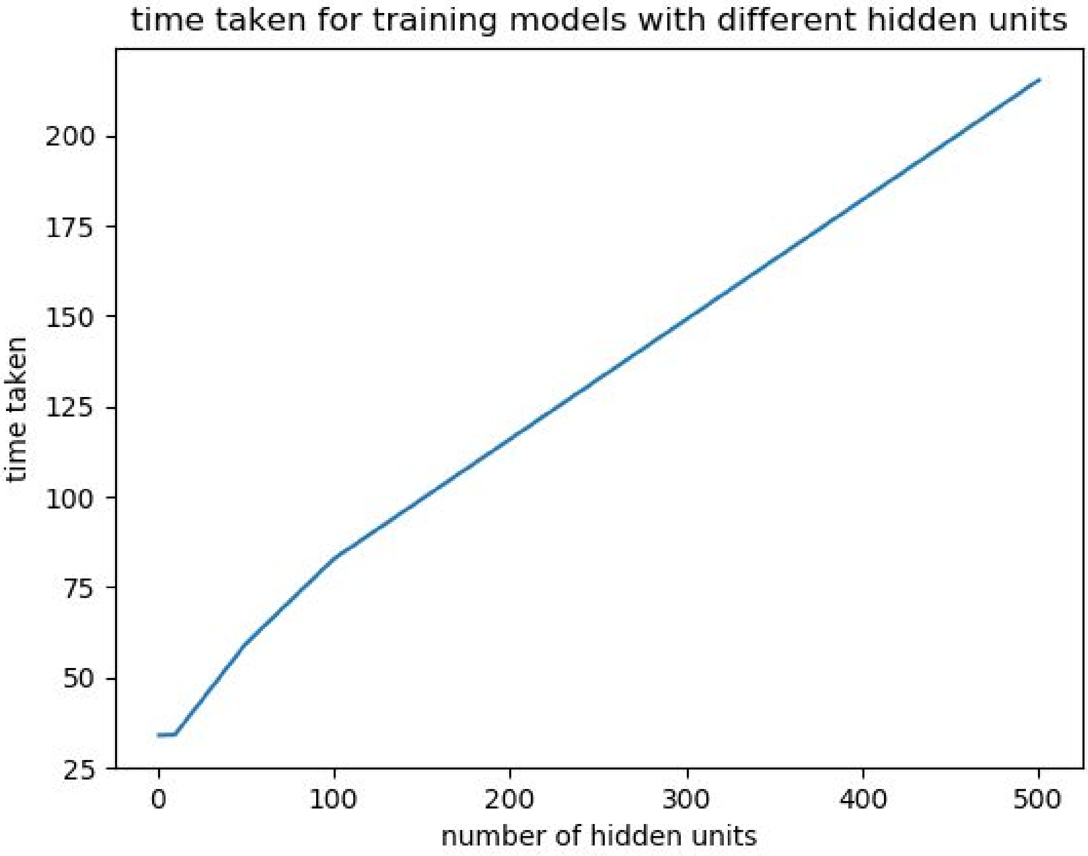
Plot for accuracy (for training set) vs number of hidden units (2b_acc1.png) :



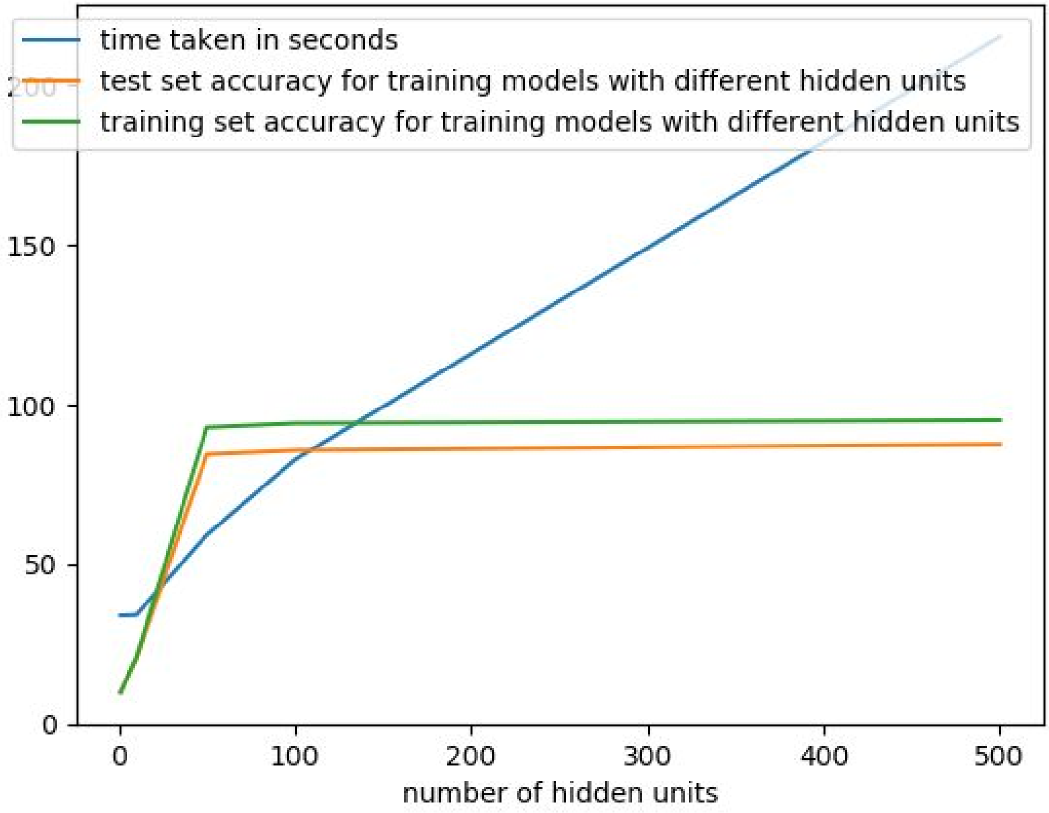
Plot for accuracy (for test set) vs number of hidden units (2b_acc2.png) :



Plot for time taken vs number of hidden units (2b_time.png) :



All 3 together (appropriately scaled) : [check2b.py]



We notice that accuracy generally increases for more number of hidden units and beyond 50, becomes close to 100% and then saturates.
More number of hidden units makes the model slower as it takes more time to train. It almost linearly varies with number of hidden units.

2C

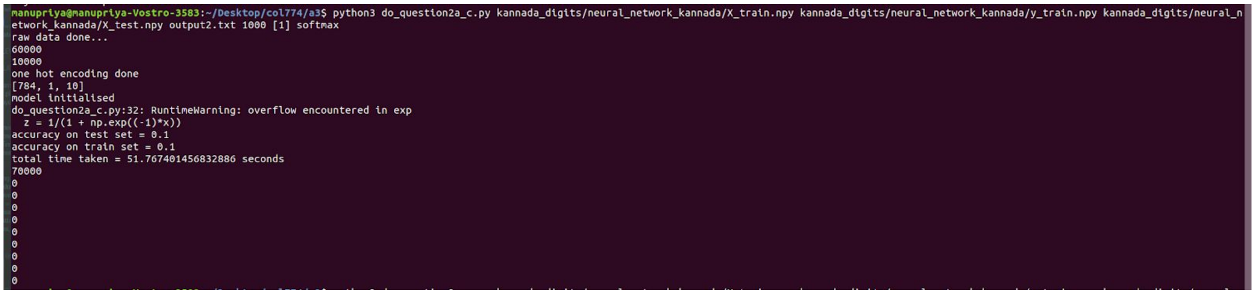
To run :

```
python3 do_question2c.py kannada_digits/neural_network_kannada/X_train.npy
kannada_digits/neural_network_kannada/y_train.npy
kannada_digits/neural_network_kannada/X_test.npy output2b.txt 1000 [100] softmax
```

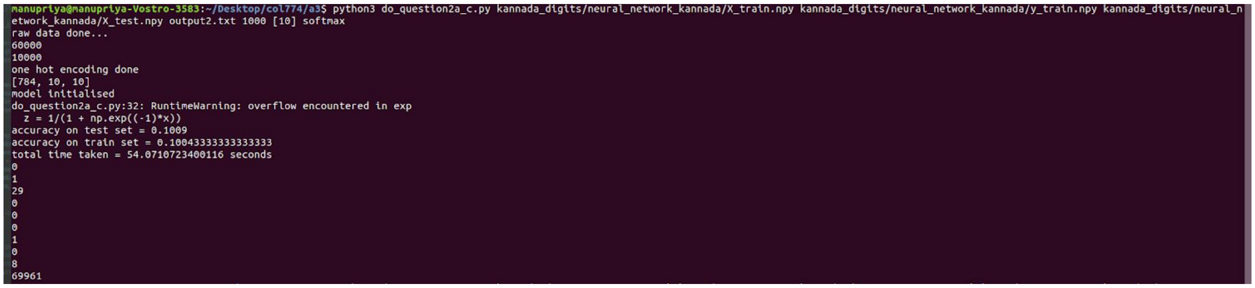
This is same as 2B except for the use of adaptive learning rate.
Neta = Neta0 / root(e)
Neta0 = 0.5 and e is the epoch number

Stopping criteria was kept the same

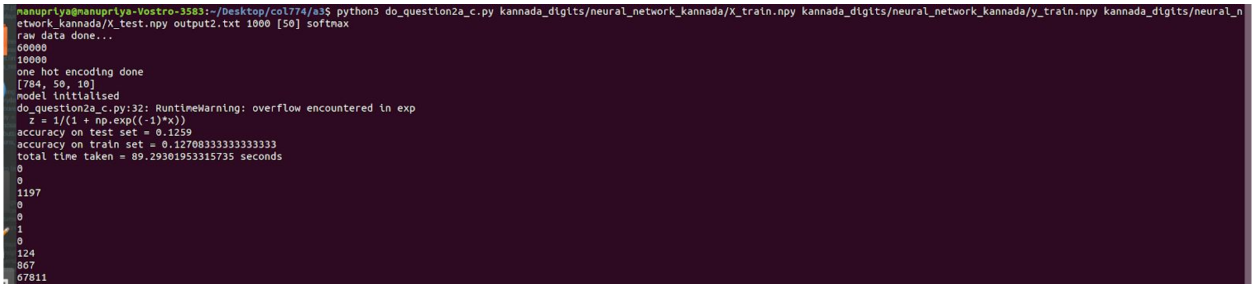
Logs :



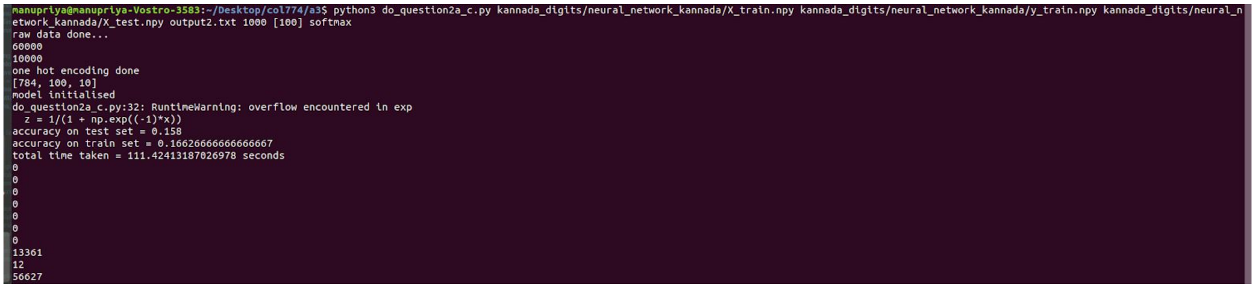
```
manupriya@manupriya-Vostro-3583:~/Desktop/col774/a3$ python3 do_question2a_c.py kannada_digits/neural_network_kannada/X_train.npy kannada_digits/neural_network_kannada/y_train.npy kannada_digits/neural_n
etwork_kannada/X_test.npy output2.txt 1000 [1] softmax
raw data done...
60000
10000
one hot encoding done
[784, 1, 10]
model initialised
do_question2a_c.py:32: RuntimeWarning: overflow encountered in exp
  z = 1/(1 + np.exp((-1)*x))
accuracy on test set = 0.1
accuracy on train set = 0.1
total time taken = 51.767401456832886 seconds
70000
0
0
0
0
0
0
0
0
0
```



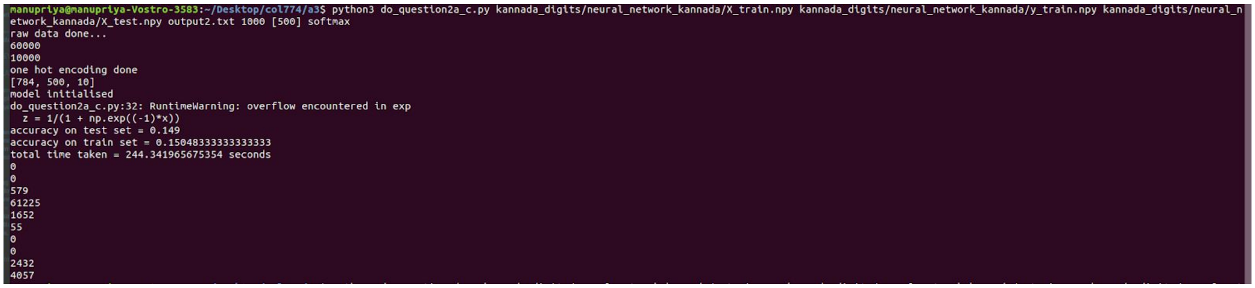
```
manupriya@manupriya-Vostro-3583:~/Desktop/col774/a3$ python3 do_question2a_c.py kannada_digits/neural_network_kannada/X_train.npy kannada_digits/neural_network_kannada/y_train.npy kannada_digits/neural_n
etwork_kannada/X_test.npy output2.txt 1000 [10] softmax
raw data done...
60000
10000
one hot encoding done
[784, 10, 10]
model initialised
do_question2a_c.py:32: RuntimeWarning: overflow encountered in exp
  z = 1/(1 + np.exp((-1)*x))
accuracy on test set = 0.1009
accuracy on train set = 0.10043333333333333
total time taken = 54.0710723400116 seconds
0
1
29
0
0
0
1
0
8
69961
```



```
manupriya@manupriya-Vostro-3583:~/Desktop/col774/a3$ python3 do_question2a_c.py kannada_digits/neural_network_kannada/X_train.npy kannada_digits/neural_network_kannada/y_train.npy kannada_digits/neural_n
etwork_kannada/X_test.npy output2.txt 1000 [50] softmax
raw data done...
60000
10000
one hot encoding done
[784, 50, 10]
model initialised
do_question2a_c.py:32: RuntimeWarning: overflow encountered in exp
  z = 1/(1 + np.exp((-1)*x))
accuracy on test set = 0.1259
accuracy on train set = 0.12708333333333333
total time taken = 89.29301953315735 seconds
0
0
1197
0
0
1
0
124
867
67811
```



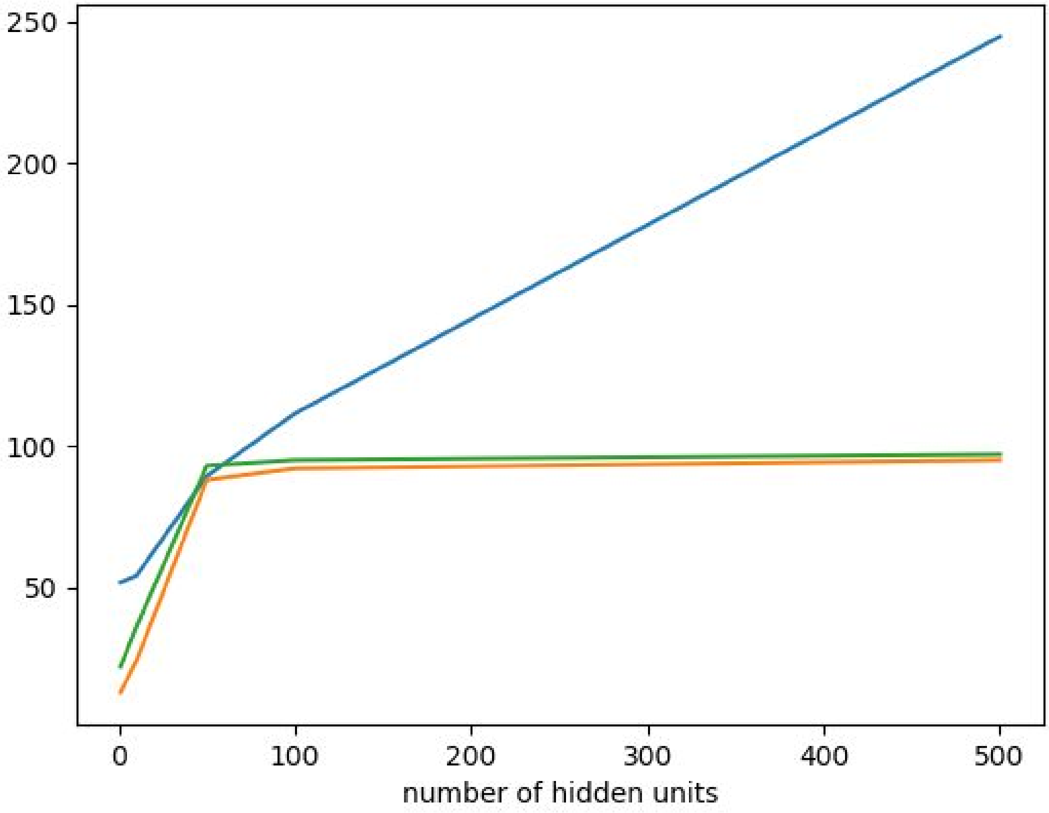
```
manupriya@manupriya-Vostro-3583:~/Desktop/col774/a3$ python3 do_question2a_c.py kannada_digits/neural_network_kannada/X_train.npy kannada_digits/neural_network_kannada/y_train.npy kannada_digits/neural_n
etwork_kannada/X_test.npy output2.txt 1000 [100] softmax
raw data done...
60000
10000
one hot encoding done
[784, 100, 10]
model initialised
do_question2a_c.py:32: RuntimeWarning: overflow encountered in exp
  z = 1/(1 + np.exp((-1)*x))
accuracy on test set = 0.158
accuracy on train set = 0.16626666666666667
total time taken = 111.42413187026978 seconds
0
0
0
0
0
0
0
13361
12
56627
```



```
manupriya@manupriya-Vostro-3583:~/Desktop/col774/a3$ python3 do_question2a_c.py kannada_digits/neural_network_kannada/X_train.npy kannada_digits/neural_network_kannada/y_train.npy kannada_digits/neural_n
etwork_kannada/X_test.npy output2.txt 1000 [500] softmax
raw data done...
60000
10000
one hot encoding done
[784, 500, 10]
model initialised
do_question2a_c.py:32: RuntimeWarning: overflow encountered in exp
  z = 1/(1 + np.exp((-1)*x))
accuracy on test set = 0.149
accuracy on train set = 0.15048333333333333
total time taken = 244.341965675354 seconds
0
0
579
61225
1652
55
0
0
2432
4057
```

Using 1, 10, 50, 100 and 500 hidden units in the single hidden layer :

All 3 together (appropriately scaled) : [check2c.py]



Adaptive learning rate increases training time by a small amount but increases accuracy on both training and test sets

2D

To run :

```
python3 do_question2d.py kannada_digits/neural_network_kannada/X_train.npy
kannada_digits/neural_network_kannada/y_train.npy
kannada_digits/neural_network_kannada/X_test.npy output2b.txt 1000 [100] relu
```

[the code in the file can be changed for relu/sigmoid]
Adaptive learning rate is used like in 2C.

For [100,100] hidden list :
ReLU gives accuracies:
Training set : 95.34
Test set : 91.05

Sigmoid gives accuracies :

Training set : 92.44
Test set : 89.16

While using sigmoid activation function and a single hidden layer, at n = 500 maximum accuracy was obtained which was around 95.45 on training set and 87.65 on test set. Hence, both the ReLU and Sigmoid functions applied on 100,100 hidden units are better models compared to the one with single hidden layer.

2E

Accuracy on test : 92.96 %
Accuracy on train : 99.725 %
Time taken = 134.67 seconds
Time taken is much less than the neural network implemented in 2D. Accuracy also outperforms the ones obtained in 2D.