COL 774 : Assignment 3

Prakhar Ganesh : 2015CS10245

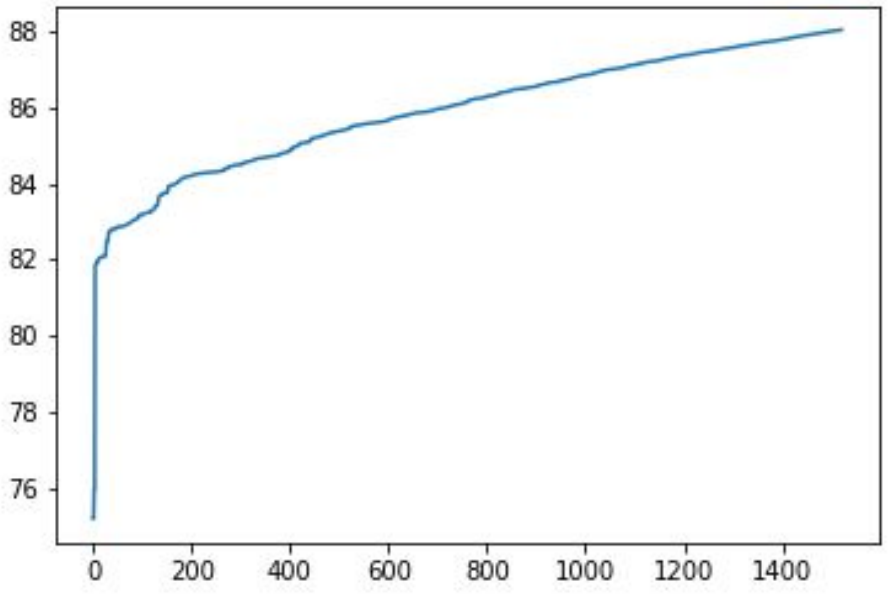
1. **Decision Trees:**

   a. Growing a Decision Tree

   Total Number of Nodes = 1521

   Training Dataset ->

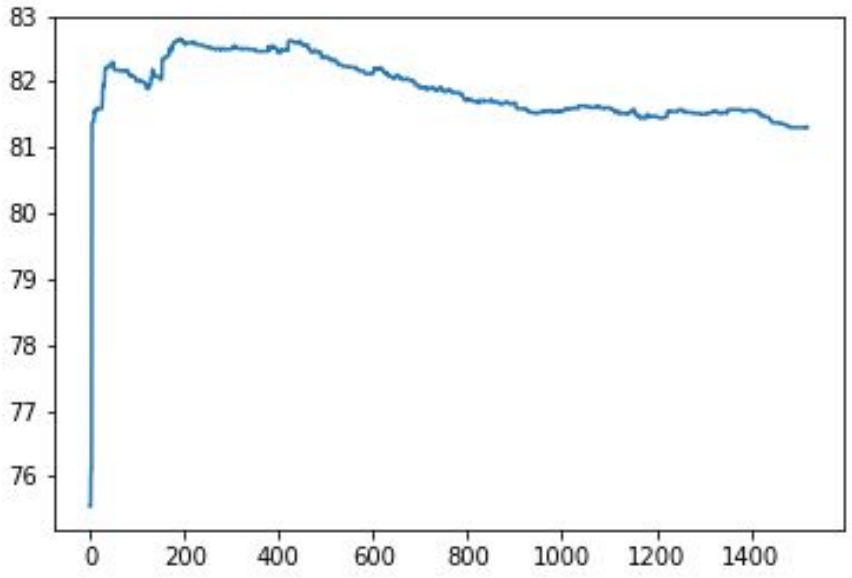
   Final Accuracy = 88.05%



Clearly since we are growing our decision tree based on the information gain that we get on the training dataset, its accuracy will monotonically increase with the number of nodes. The increase will be at different rates at different points dependent on the gain obtained at that point.

Test Dataset ->

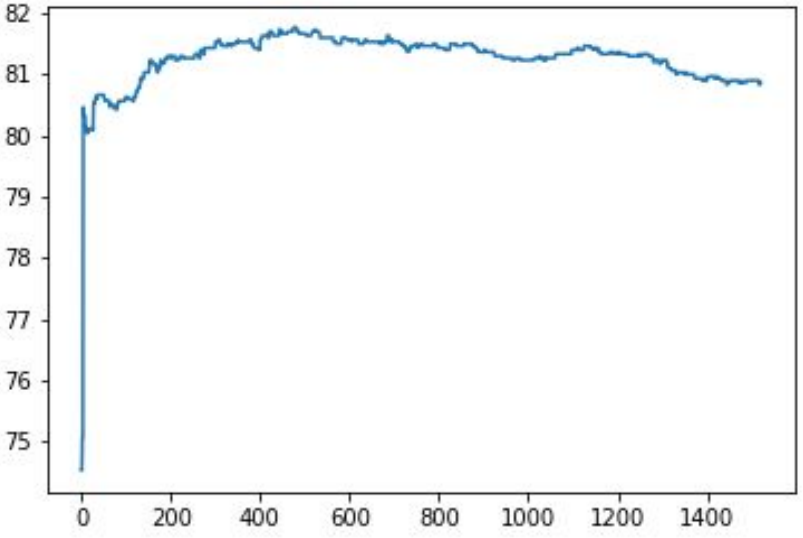
Final Accuracy = 81.30%



Assuming similar behaviour in the test dataset and the training dataset, we can definitely see an initial increase in the accuracy but since the fine graining of the tree is actually done based on the training set, once the test dataset reaches around its maximum accuracy, it will fluctuate.

Validation Dataset ->

Final Accuracy = 80.86%

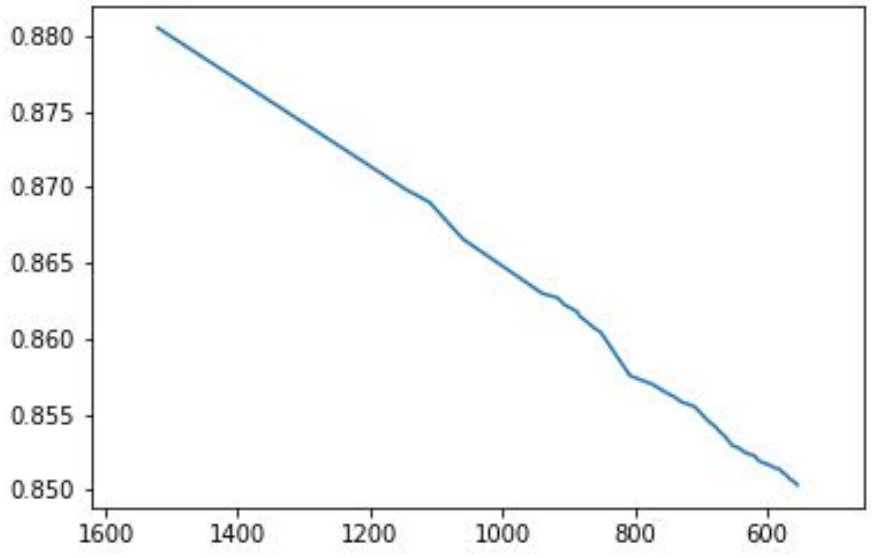


Since we are not using the validation set in this part to help the growing of the tree in any particular way, we can say that both test dataset and validation dataset will have similar behaviour. That is, as explained earlier, we will see an initial increase in the accuracy but once the validation dataset reaches around its maximum accuracy, it will fluctuate.

b. Pruning

Final Number of Nodes after Pruning = 554

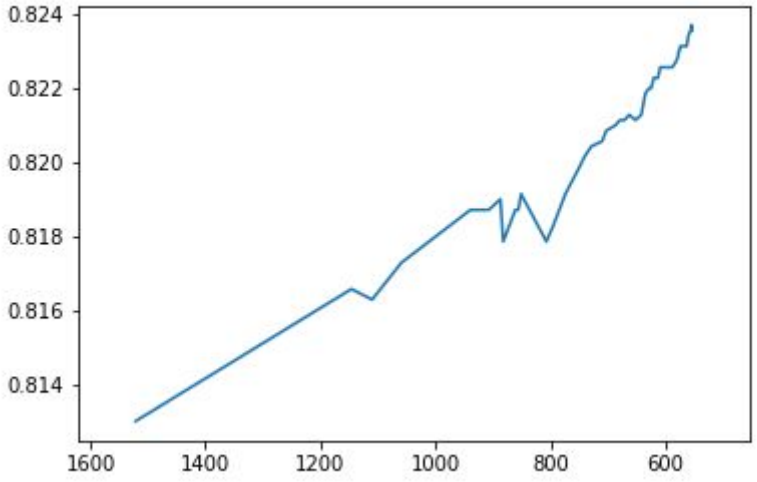
Training Dataset ->
Final Accuracy = 85.03%



Since the decision tree was first grown using the training dataset, at every step we growed our tree when we saw information gain for the training dataset. This clearly means that any pruning done using a different dataset, validation set, will decrease the accuracy on the training dataset.

Test Dataset ->

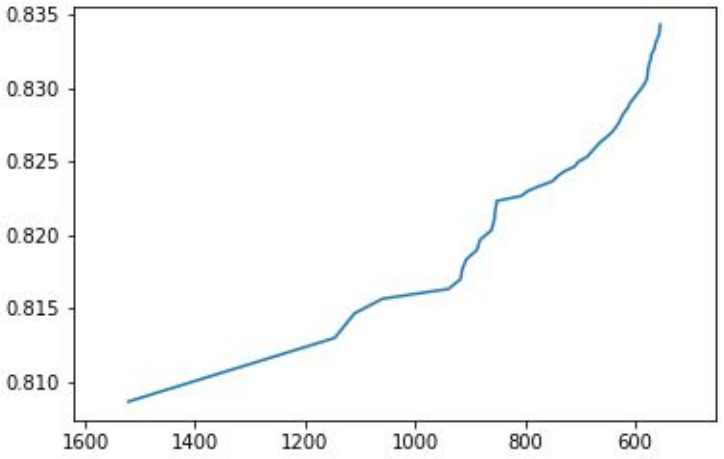
Final Accuracy = 82.35%



Since the tree grown using the training set is later pruned using the validation set, it comprised best information out of both the datasets and thus is a better learned model to understand the data distribution that the one where no pruning was done and thus the accuracy on the test dataset increases.

Validation Dataset ->

Final Accuracy = 83.43%



Since the pruning is done based on correctly classifying examples in the validation dataset, clearly the accuracy on validation will increase with more and more pruning.

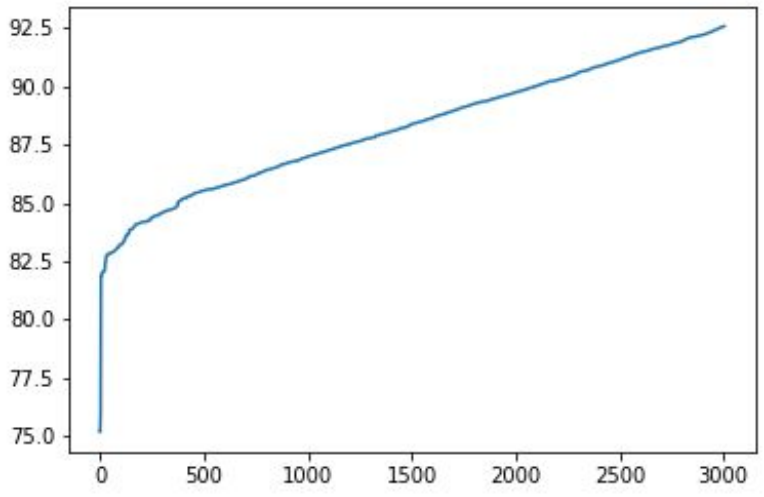
c. Median on the fly

The attributes which were split multiple times in a branch and the median values at the branch with maximum number of splits ->

1. Age
   40.0
   49.0
   53.0
   56.0
   58.0
   60.0
   61.0
2. Fnlwgt
   171931.0
   116774.5
   144556.0
   155016.5
   163958.5
3. Education Number
   10.0
   12.0
4. Hour per Week
   40.0
   50.0
   55.0
   60.0

Training Dataset ->

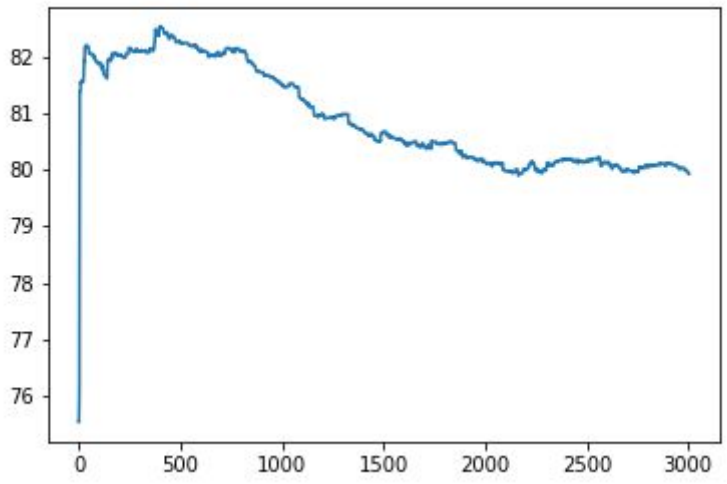
Final Accuracy = 92.58%



Using median calculation on the fly instead of doing it beforehand clearly would give a better distribution of data at every node at which that particular attribute is used in division. Thus the training accuracy increases even further.

Test Dataset ->

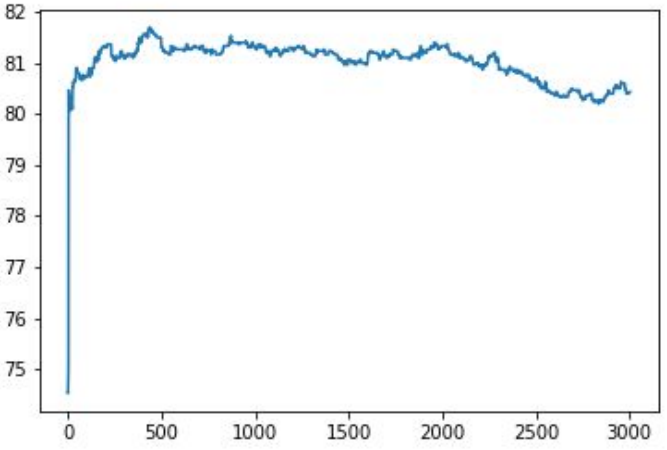
Final Accuracy = 79.92%



Since the medians are calculated at every node using the distribution of the training dataset, it is possible that the model may have overfitted itself to the training dataset and thus such changes in the decision tree may actually decrease the test accuracy.

Validation Dataset ->

Final Accuracy = 80.43%

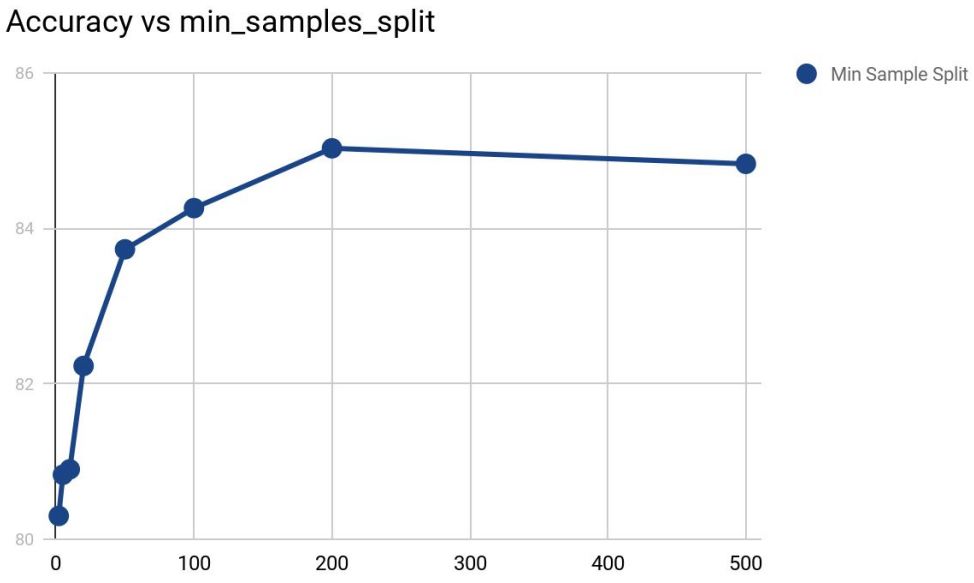


Same logic as used for test dataset works for validation dataset also but in this particular case, it looks like possibly the distribution of training and validation datasets maybe match a little because of which the testing accuracy has not deteriorated substantially.

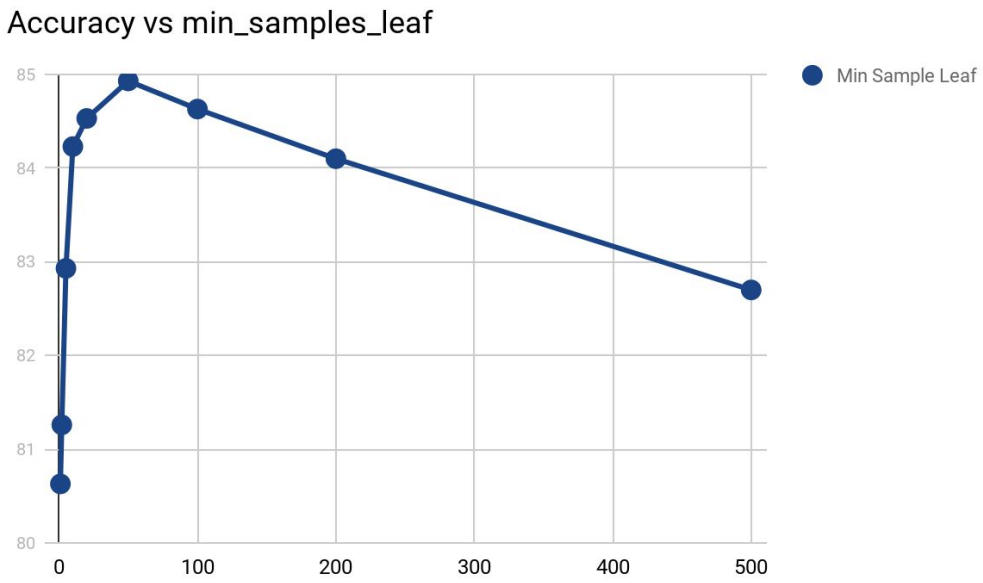
d. Using sklearn to make decision trees

We used the standard sklearn libraries to implement the decision tree on our training dataset and then observed the trend of the accuracy on the validation dataset by varying different parameters (keeping the rest constant and default). Later we did a grid search to obtain the parameters giving the best validation accuracy.

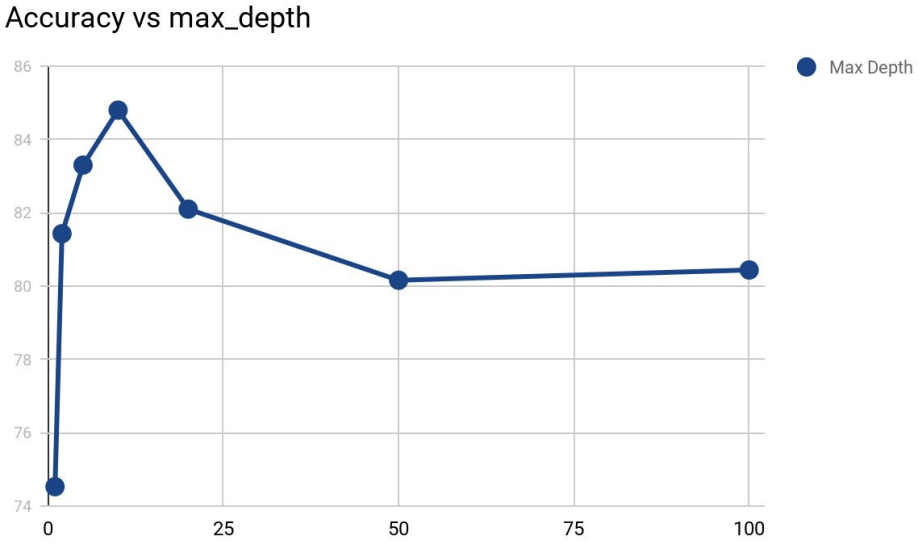
Accuracy behaviour vs number of minimum samples required to break a node.



Accuracy behaviour vs number of minimum samples required to make a leaf node.



Accuracy behaviour vs the maximum depth of the decision tree allowed.



Finally, after a grid search, the final parameters that were chosen were,
min_samples_split=200
min_samples_leaf=1
max_depth=20

Training Dataset ->

Final Accuracy = 86.52%

Test Dataset ->

Final Accuracy = 84.75%

Validation Dataset ->

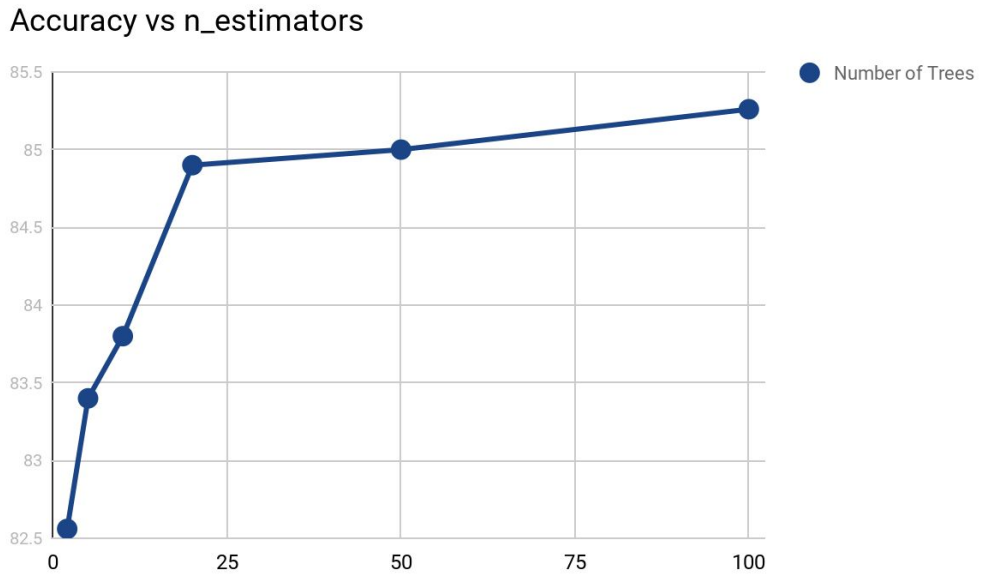
Final Accuracy = 85.26%

Clearly the Test Dataset accuracy (which should be the criteria of judgement since in both the methods, it was not used in any form) is better in this case than in part(b) (the accuracy was 82.35%).

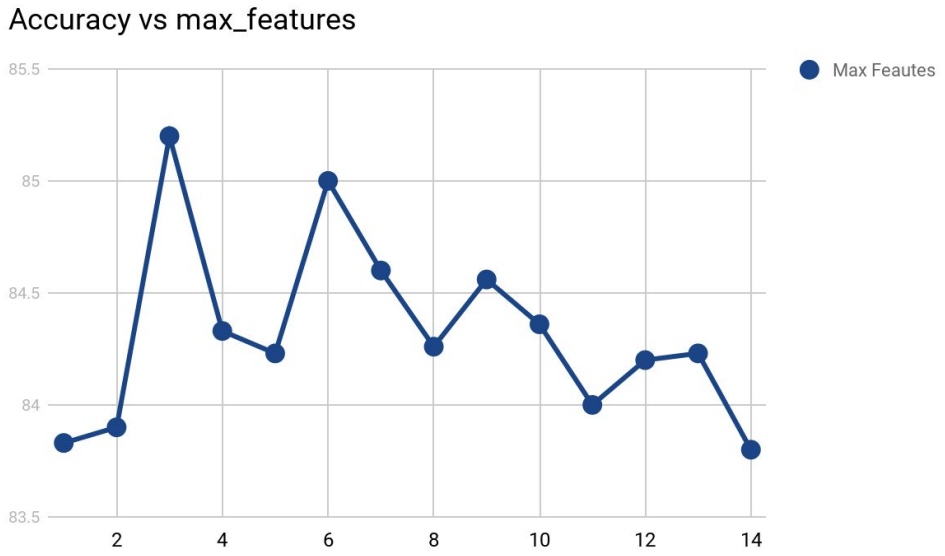
e. Random Forests

We used the standard sklearn libraries to implement the random forest on our training dataset and then observed the trend of the accuracy on the validation dataset by varying different parameters (keeping the rest constant and default). Later we did a grid search to obtain the parameters giving the best validation accuracy.

Accuracy behaviour vs Number of Trees in the Random Forest



Accuracy behaviour vs Number of Max features used for best split



Accuracy behaviour vs whether bootstrap samples were used while building the tree or not

With bootstrap -> 83.8%
Without bootstrap -> 80.46%

Finally, after a grid search, the final parameters that were chosen were,
n_estimators=100
max_features=4
bootstrap=True

Training Dataset ->
    Final Accuracy = 99.99%
Test Dataset ->
    Final Accuracy = 84.92%
Validation Dataset ->
    Final Accuracy = 85.43%

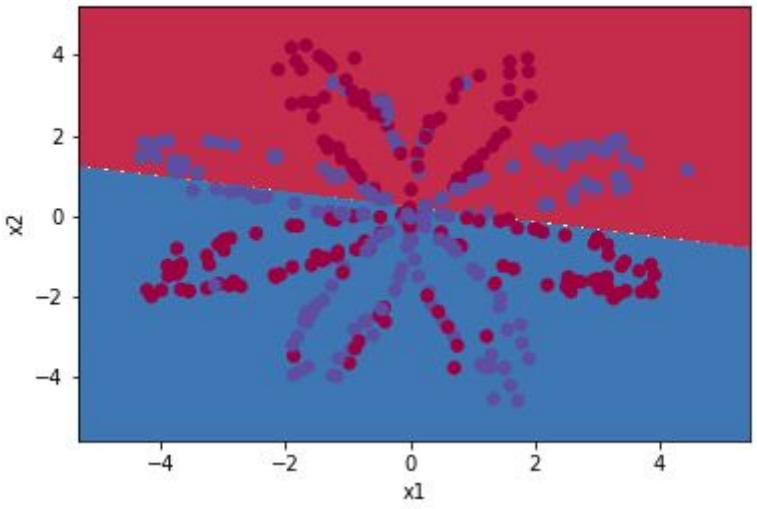
Clearly the Test Dataset accuracy (which should be the criteria of judgement since in both the methods, it was not used in any form) is just slightly better in this case than in part(b) (the accuracy was 82.35%) or part(c)(the accuracy was 84.75%).

2. **Neural Networks:**

a. Logistic Regression Learner

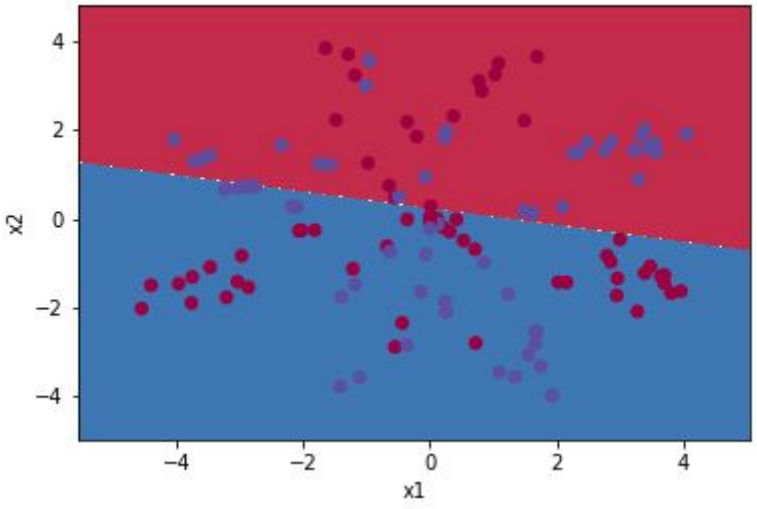
Using the Logistic Regression Model from the sklearn library, we got the following accuracies ->

Training Accuracy = 45.78



Clearly since the given dataset is not linearly separable, the logistic regression model will not be able to classify the given dataset properly. In the graph we can see that even though a line is made by the model which reduces the error to the minimum, still clearly it is not good enough thus giving us poor accuracy.

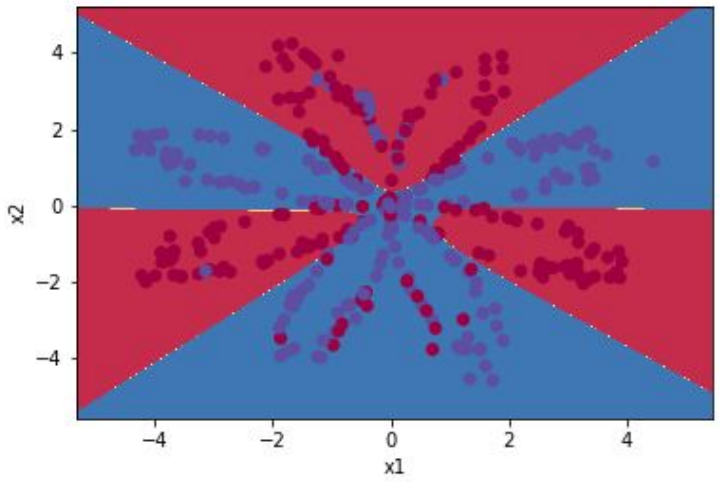
Test Accuracy = 38.33



b. Neural Network Implementation

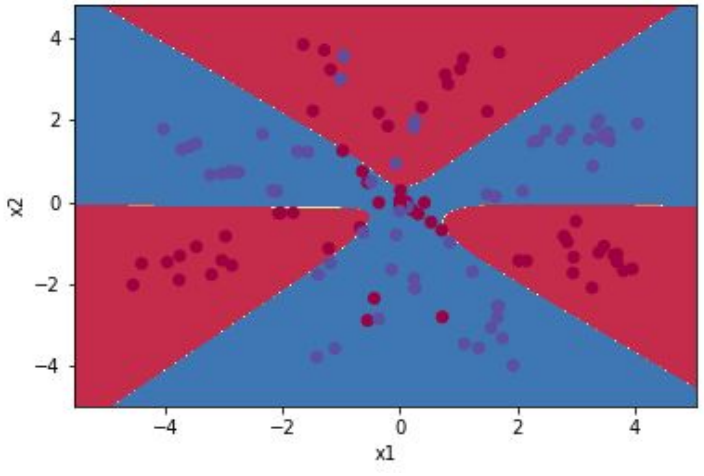
Using our implementation of the neural network and the following parameters ->
Stopping Criteria = delta(error) for every batch < 0.001
Hidden Layers = One hidden layer with 5 perceptrons
Batch size = # of inputs

Training Accuracy = 87.89%



Clearly since the neural network can learn more complex and non linear classifier boundaries than a linear regression model, it is able to learn the distribution better.

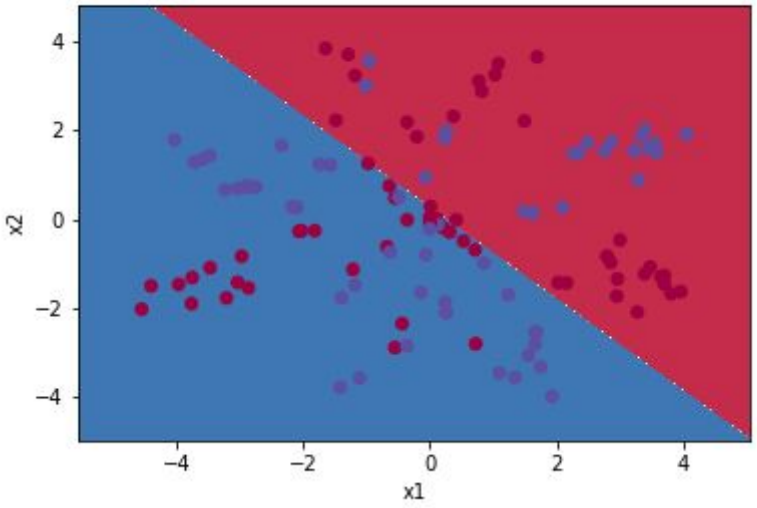
Test Accuracy = 84.16%



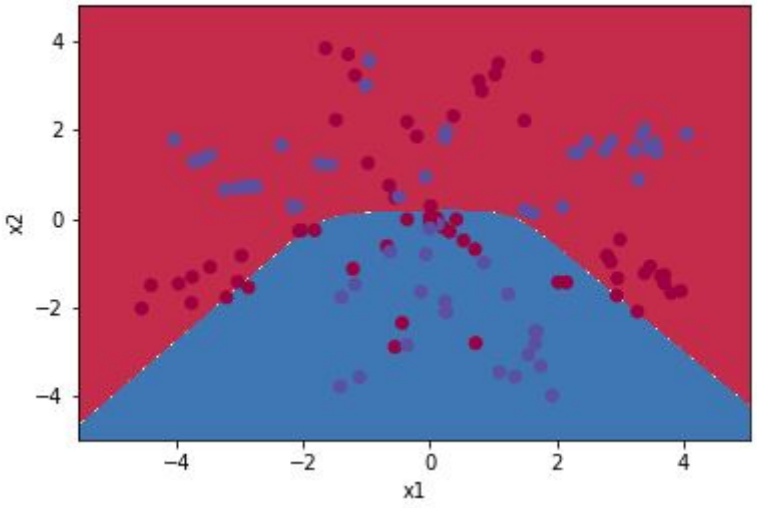
c. Changing Architecture

Constant parameters ->
Stopping Criteria = delta(error) for every batch < 0.001
Batch size = # of inputs

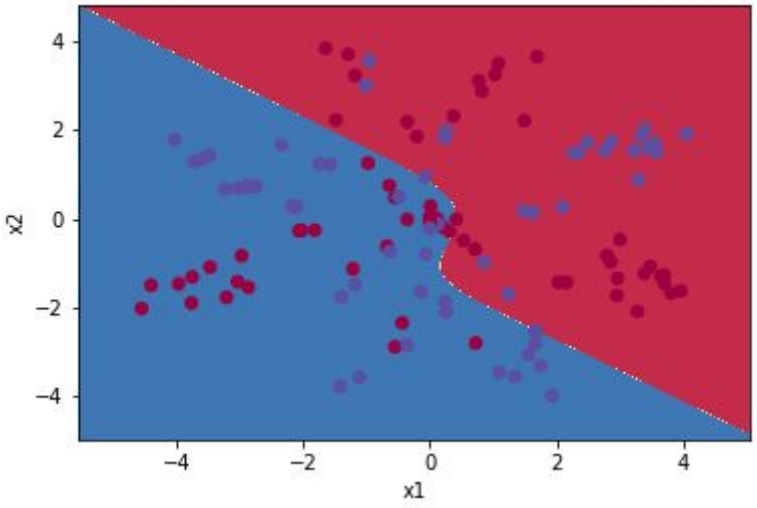
Hidden Layers = One hidden layer with 1 perceptrons
Training Accuracy = 63.15%
Test Accuracy = 56.66%



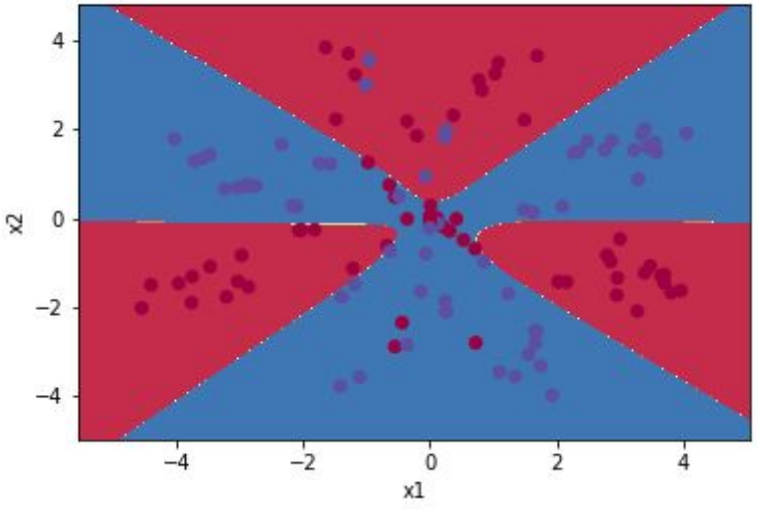
Hidden Layers = One hidden layer with 2 perceptrons
Training Accuracy = 53.21%
Test Accuracy = 52.5%



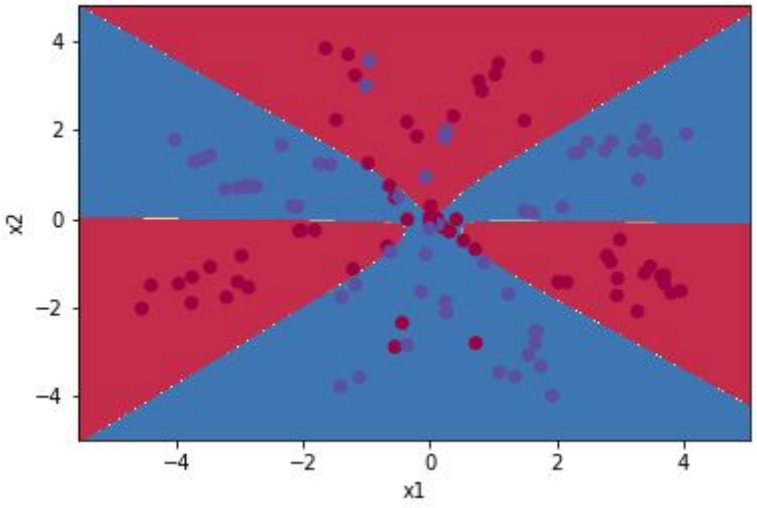
Hidden Layers = One hidden layer with 3 perceptrons
Training Accuracy = 60.26%
Test Accuracy = 54.16%



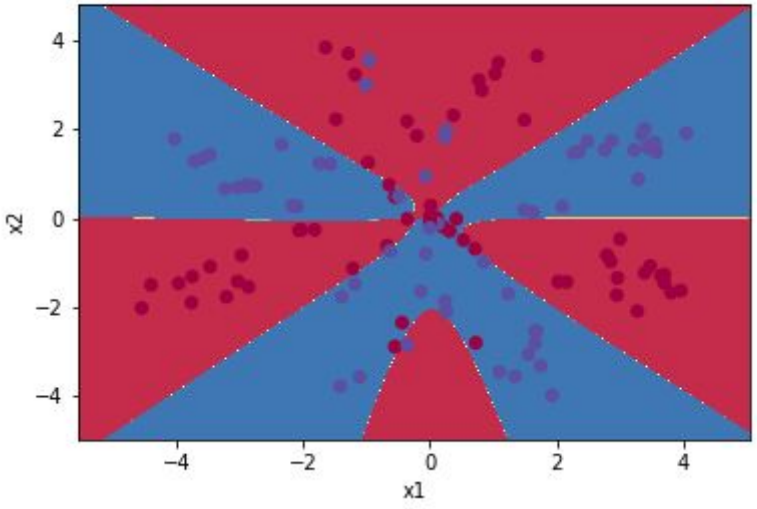
Hidden Layers = One hidden layer with 5 perceptrons
Training Accuracy = 86.33%
Test Accuracy = 81.16%



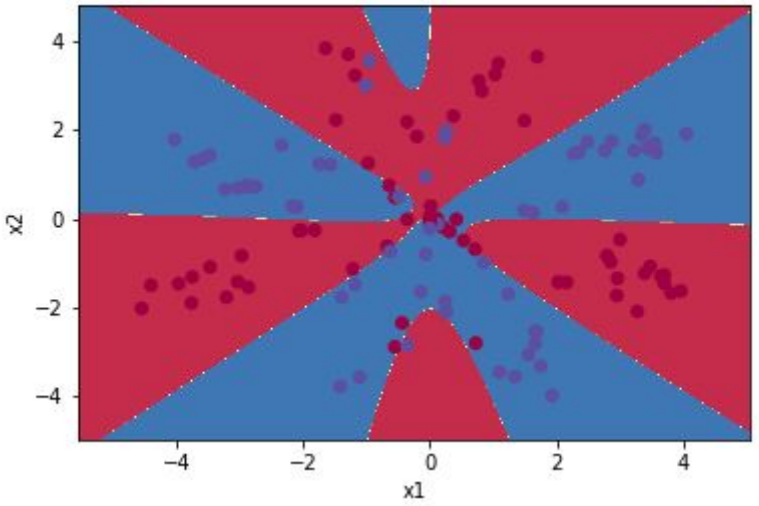
Hidden Layers = One hidden layer with 10 perceptrons
Training Accuracy = 87.63%
Test Accuracy = 81.66%



Hidden Layers = One hidden layer with 20 perceptrons
Training Accuracy = 88.68%
Test Accuracy = 82.5%

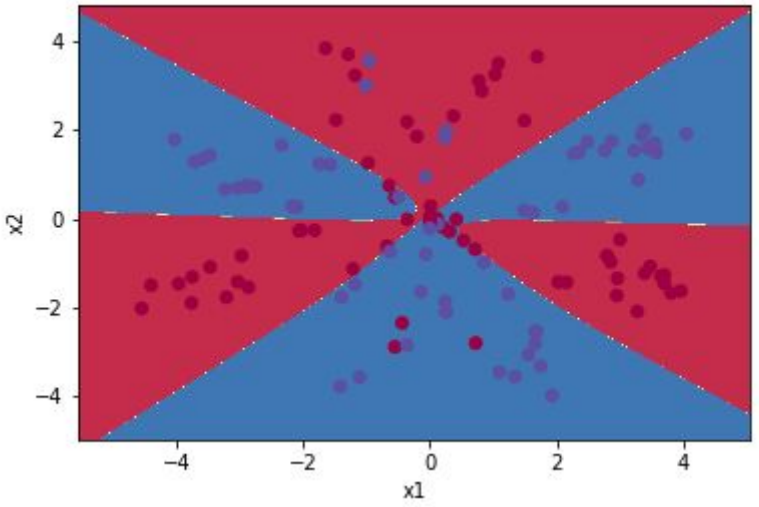


Hidden Layers = One hidden layer with 40 perceptrons
Training Accuracy = 89.21%
Test Accuracy = 82.5%



The decision boundary formed gets more and more complex, i.e. more and more curves are introduced every time we increase the number of perceptrons in the hidden layer. Although that does not mean that the classification is getting better because it is possible that the boundaries may overfit itself to the training dataset. The best # of perceptrons is possibly 20 because after that only training accuracy increases, i.e. the boundaries start to overfit.

Hidden Layers = Two hidden layers with 5 perceptrons each
Training Accuracy = 89.21%
Test Accuracy = 83.33%



More layers means more complexity in the decision which means better classification which can be seen. Even if that many layers are not required, with enough training, the corresponding weights can adjust such that the layers are redundant.

d. MNIST : LIBSVM

Training Accuracy = 99.87%
Test Accuracy = 98.33%

This shows that the data we have for the digits 6 and 8 is linearly separable and thus LIBSVM gives us very good accuracies

e. MNIST : Single Perceptron

Training Accuracy = 99.66%
Test Accuracy = 97.16%

Since a single perceptron with a sigmoid function at the output is basically working like a linear regression model, and since we noticed above that the data is linearly separable, we can explain the accuracies obtained above.

f. MNIST : Variable learning rate

Hidden Layer = Single with 100 perceptrons
Training Accuracy = 98.78%
Test Accuracy = 97.26%

When the function describing the dataset is not complex but we still use a lot of perceptrons and layers in our architecture, the model actually learns to make those perceptrons and layers redundant so that it can learn the simple classifier boundary (which is a linear classifier in this case). However to do so requires a lot of training and thus with less amount of training, the model may learn complex boundary which doesn't actually completely satisfy the distribution. Thus we can explain the slight drop in the accuracies as compared to using a single perceptron.