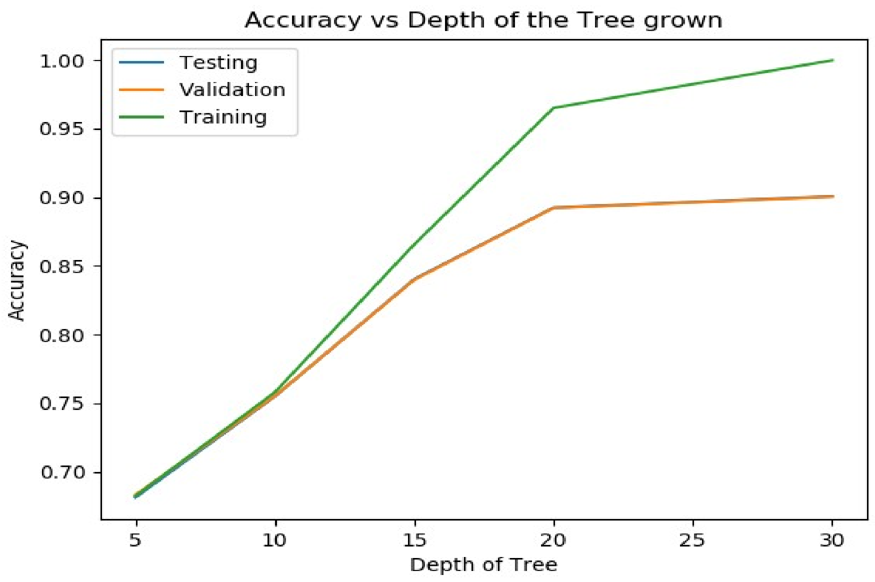
COL 774 – MACHINE LEARNING
ASSIGNMENT 3

UTSAV DEEP
2018CS10396

1. DECISION TREES (AND RANDOM FORESTS)

Part (a) Decision Tree Construction

Implementation Details: I have grown the tree in depth first manner. And all the observation have been made on the depth of the tree which is a reflection on the number of nodes.



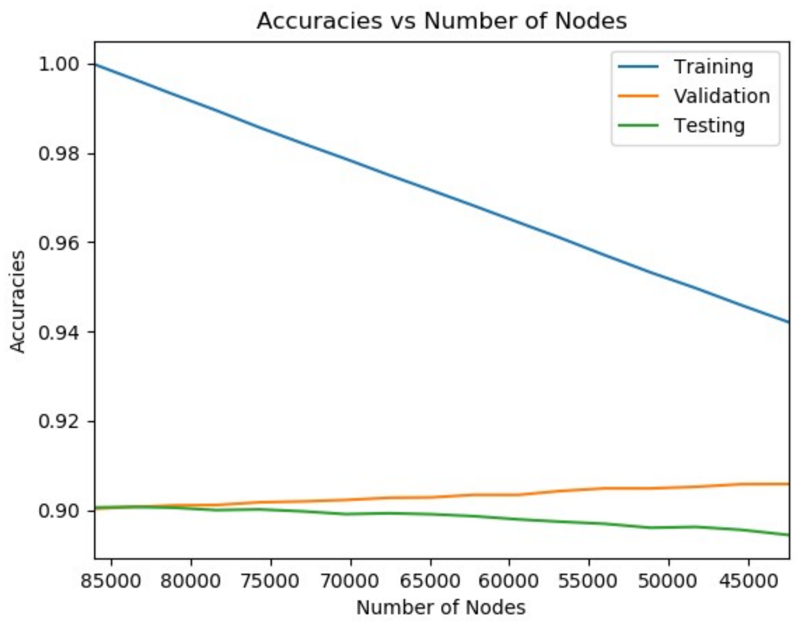
On growing the whole tree:
Max depth = 31
Total number of nodes = 86071
Training time = 9 minutes for full tree.

Training set accuracy = 99.97973775746095 %
Validation set accuracy = 90.03133849891465 %
Test set accuracy = 90.05991941871276 %

Observations:
> From the plots it can be observed that the training accuracy, validation accuracy and the test accuracy, all three increase as we increase the depth of the tree(and hence the number of nodes).
> However, the increase is maximum for the training set.
> This shows that there is a lot of over-fitting going on in the data which happens a lot in decision trees. We use random forests to remove this over fitting.

Part (b) Decision Tree Post Pruning

Implementation Details: Pruning is used to remove over fitting in the decision tree. In every pruning step nodes are pruned such that maximum increase is seen in the validation set accuracy.



Total number of nodes After Pruning = 42289 nodes
Reduction in number of nodes = 86071 – 42289 = 43782 nodes
Training time = 10 minutes.

Training set accuracy after pruning = 94.1964025309383 %
Validation set accuracy after pruning = 90.58335609447551 %
Test set accuracy after pruning = 89.43489789593305 %

Observations:
> As expected the validation test accuracy increases as more and more nodes are pruned.

> However the increase is very small because the validation accuracy was already increasing with depth as observed in part a.
> Thus the accuracy could not change by drastic amounts after pruing.
> As were are reducing the over-fitting and removing the nodes we built for the training data set, it is normal to observe that the training data set accuracy continuously decreases as more and more nodes are pruned.
> The test data accuracy is also observed to decrease as more and more nodes are pruned. The test accuracy decreased by about 0.5 % due to pruning.

Part (c): Random Forests

Implementation Details:
> I used the Grid search method which tries out all the possibilities to give the best attributes.
> The method used was clf = GridSearchCV(rfc, parameters) where parameters is the dictionary containing all the values that I wanted to search for.
> The out of the bag accuracy was fond using the oob_score_ attribute of the Random forest classifier.
> clf.best_params_ attribute was used to get the best parameters for the model. The output was
{'n_estimators': 450, ‘max_features’: 0.7, ‘min_samples_split’ : 2}

By using the out-of- the bag accuracy model, the best parameters found were:
n_estimators : 450
max_features : 0.7
min_samples_split : 2

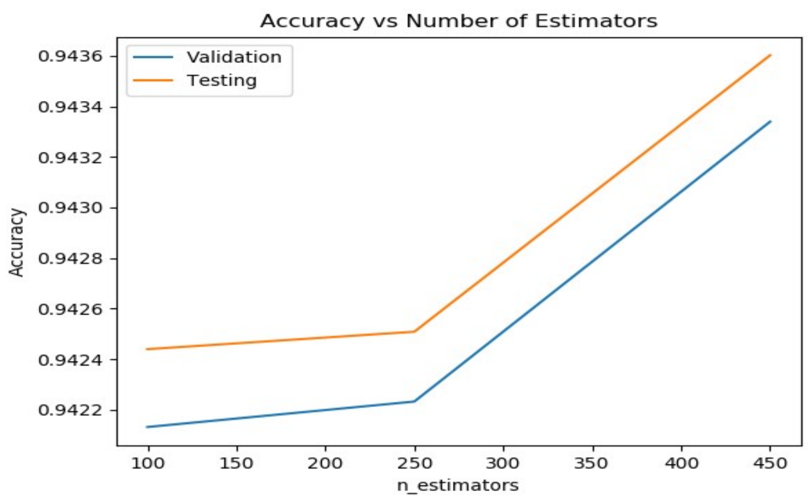
The Out of the Bag accuracy for these parameters = 96.35555937604573 %
Training set accuracy = 100.0 %
Validation set accuracy = 96.34503974814197 %
Test set accuracy = 96.40139123248046 %

Comparision with accracies obtained in part (b):
> All the three values, training set accuracy, validation set accuracy and the test set accuracy are better in random forests case that in the post pruning case.
> All the accuracies are considerably higher than the pruning case
> The validation and the test accuracy are very close to the training data accuracy, thus hinting that the over-fitting has been removed by the Random Forest.
> With this we can conclude that Random Forests are a very good way to remove the over-fitting problem in the Decision Tree model.

Part (d) Parameter Sensitivity Analysis for Random Forests.

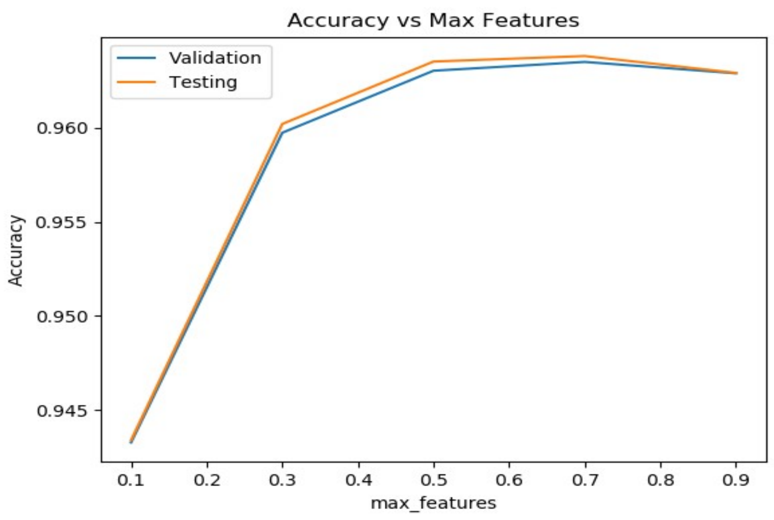
Observations regarding sensitivity of each parameter:

=> Varying n_estimators keeping others constant:



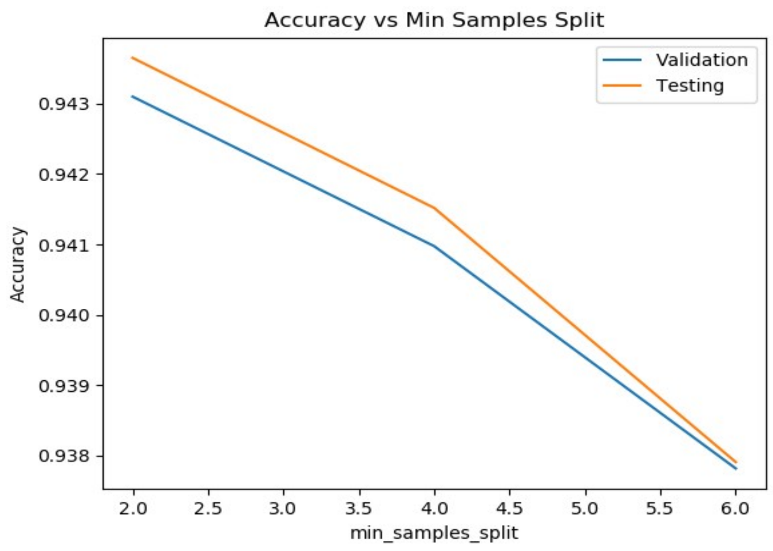
> We can see that the accuracy increases over both validation and test set rapidly.
> However the change in accuracy is not much.
> From the data observed we can conclude that the model is not very sensitive to the number of estimators as the accruacy increase happens in the 3rd decimal.
> It is also observed that the training accuracy for 100,250,350,450 estimators is 100% in each case.

=> Varying max_features keeping others constant:



> I varied max_features with the values: [0.1, 0.3, 0.5, 0.7, 0.9]
> The best test and validation accuracies were found for value 0.7 as expected from the grid search.
> From the graph we can see that the accuracy depends upon the number of max features.
> However, near the maximum value, varying the max_features value does not change the accuracies much.
> Therefore it can be concluded that this model is not very sensitive to max_features near the best value.
> It is also observed that the training accuracy for all values is 100% in each case.

=> Varying min_samples_split others constant:



> It can be observed that the accuracies of both test data and validation data change upon changing the min_samples_split value.
> The best accuracies are for the value 2 as observed after grid search.
> Changing the min_samples_split value causes the accuracies to change however the change only occurs after 3 point in the decimal value.
> The model is very senisitive to the min_samples_split value as it can be observed from all the graphs in above.

> Among n_estimators, max_features and min_samples_split, the model looks to be most sensitive to min_samples_split, then to n_estimators and finally to max_features.

2. NEURAL NETWORKS

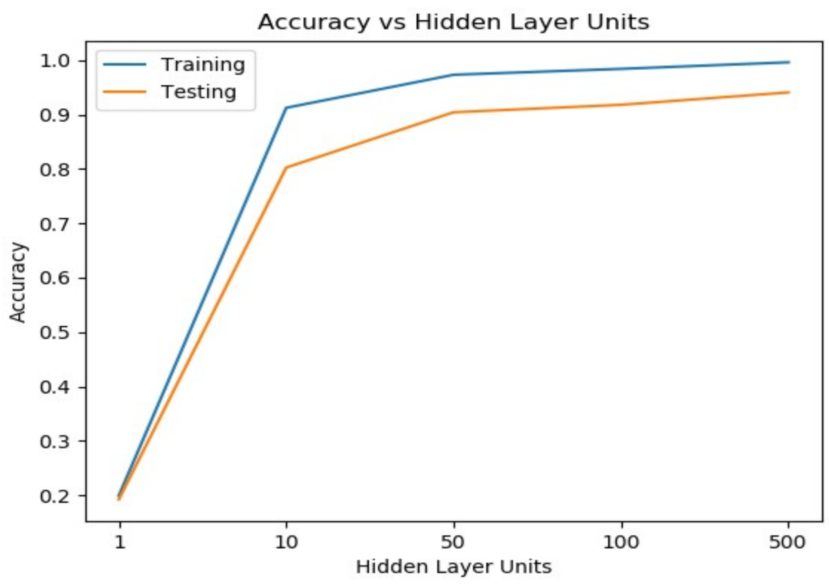
Part(a) Neural Network implementation

Implementation Details:
> One hot encoding was done to the output parameters to convert integer y into a vector.
> The format of the training X was modified to fit the requirements.
> The architecture was assumed to be fully connected as asked.

Part (b) Single Layer

> I set the learning rate to 0.1 and used mini-batch size of 100.

Stopping Criterion:
1) epochs > maxEpochs
   maxEpochs = 150
2) Loss < 10 4 for consequtive 10 updates



For hidden layer units = 1:
Training Accuracy = 19.915 %
Test Accuracy = 19.24 %
Time taken = 2 min 25sec

For hidden layer units = 10:
Training Accuracy = 91.22666666666667 %
Test Accuracy = 80.24 %
Time taken = 4 min

For hidden layer units = 50:
Training Accuracy = 97.30333333333333 %
Test Accuracy = 90.41 %
Time taken = 8 min 30 sec

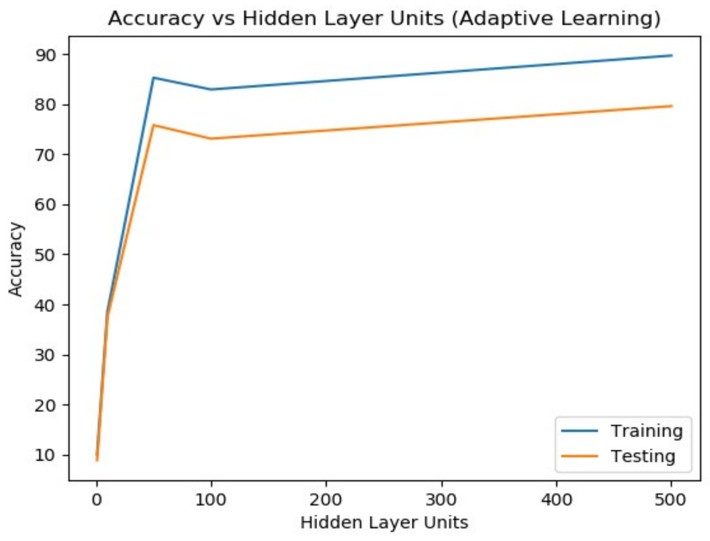
For hidden layer units = 100:
Training Accuracy = 98.385 %
Test Accuracy = 91.78 %
Time taken = 10 min 50 sec

For hidden layer units = 500:
Training Accuracy = 99.59333333333333 %
Test Accuracy = 94.06 %
Time taken = 19 min

Observations:
> It is observed that the accuracy of both training data and test data increases with the increase in number of hidden layer units.
> More the number of hidden layer units better is the prediction by the model.

Part (c) Adaptive learning

Stopping Criterion:
1) epochs > maxEpochs
    maxEpochs = 500
2) Loss < 10 4 for consequtive 10 updates



For hidden layer units = 1:
Training Accuracy = 10 %
Test Accuracy = 8.9 %
Time taken = 4 min

For hidden layer units = 10:
Training Accuracy = 38.533333333333336 %
Test Accuracy = 37.48 %
Time taken = 6 min 20 sec

For hidden layer units = 50:
Training Accuracy = 85.26833333333333 %
Test Accuracy = 75.8 %
Time taken = 12 min

For hidden layer units = 100:
Training Accuracy = 82.92166666666667 %
Test Accuracy = 73.1 %
Time taken = 16 min 40 sec

For hidden layer units = 500:
Training Accuracy = 89.68833333333334 %
Test Accuracy = 79.6 %
Time taken = 25 min

> Adaptive learning has increased the time taken to train the model for each case of number of hidden units.
> Not only that, even the accuracy has decreased when compared to its fixed counterpart.
> Adaptive learning has not helped in this model for the given data set.
> The accuracy has decreased for both test and validation sets.

Part (d) Double hidden Layer

> Implemented a network with 2 hidden layers with 100 units each.

Both layers Sigmoid case :
Training Accuracy = 98.035 %
Test Accuracy = 91.63 %

1 ReLU and 1 sigmoid case :

Training Accuracy = 98.751 %
Test Accuracy = 92.84 %

Comparing the above two cases:
> One ReLU and one Sigmoid case delivers better accuracy than both Sigmoid case.
> It can be observed that the accuracy has increased in the case of one ReLU and one sigmoid case when compared to the both sigmoid case.
> This is because ReLU introduces non-linearity in the network thus helping the model make better predictions.

Comparing with single layer case:
> The accuracy has significantly increased when compared to the single layer model.
> For 100 units single layer had accuracy of 91.78 % over the test data whereas the double layer had an accuracy of 92.84%.
> In general double hidden layer neural networks give more that 90% accuracy over any data set.
> This thus means that, having a double hidden layer neural network will give better predictions when compared to the single hidden layer neural network.

Part (e) Scikit-learn library

Training Accuracy = 99.2 %
Test Accuracy = 94.8 %

Comparing with part(d):
> The accuracy when compared to part(d) has increased for both training as well as test data set.
> The accuracy has increased by a very small amoumt in both cases although.
> This means that the implementation of the SkLearn library is similar to ours.

THANK YOU!