Decision Trees and Random Forests

-Devendra Pratap Yadav

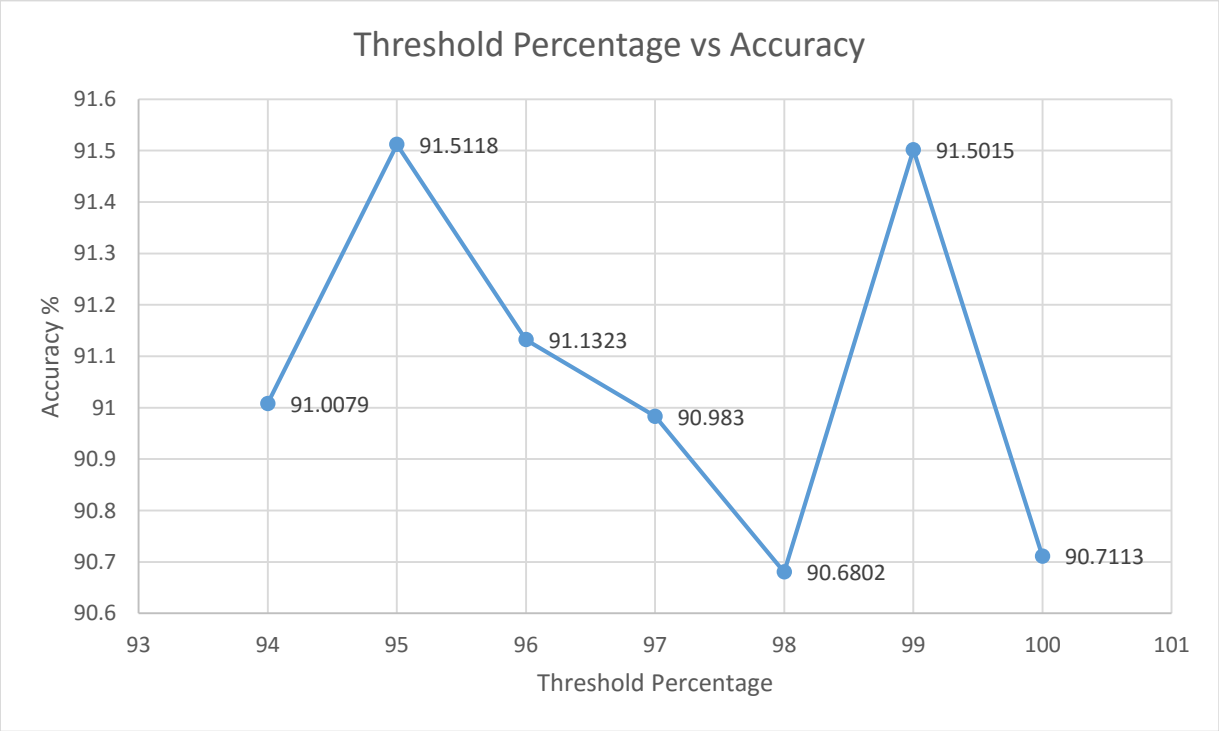
We have implemented the ID3 algorithm for constructing Decision Trees in C++.
The nodes of the tree are a struct containing Attribute Label and Value according to which it was split, along with a list of attributes used so far by parent nodes, list of children of the node and class labels for the data points represented by the subtree under the current node.

For each of the experiments, we present the results using graphs and discuss them.

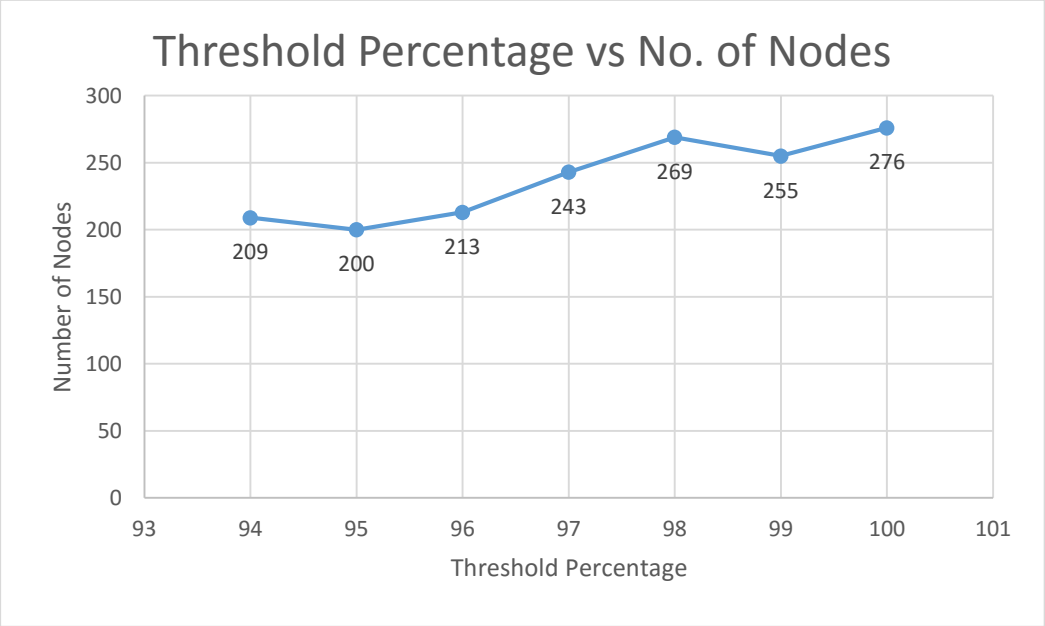
EXPERIMENT 1

We consider a random subset of 1000 data points from the given data to train the Decision Tree.Due to small number of '1' class labels (~400) for total data (~5800), the accuracy of the tree varies based on how many '1' label data points are selected. Hence, all the data presented for a given experiment is average of 10 test runs.

We apply early stopping criteria in the form of "Threshold Percentage". If more than X% of data points being considered at a node have the same class label 'a', then we label that node as a terminal node with class label 'a' and don't split it further.
We vary the Threshold Percentage from 94% to 100% and plot the results. Below 94%, we often get a tree with a single node since most data points have '0' class label.



We observe a peak at 95% stopping criteria and then the accuracy goes down. No stopping criteria has lower accuracy as it is more complex and overfits the data.



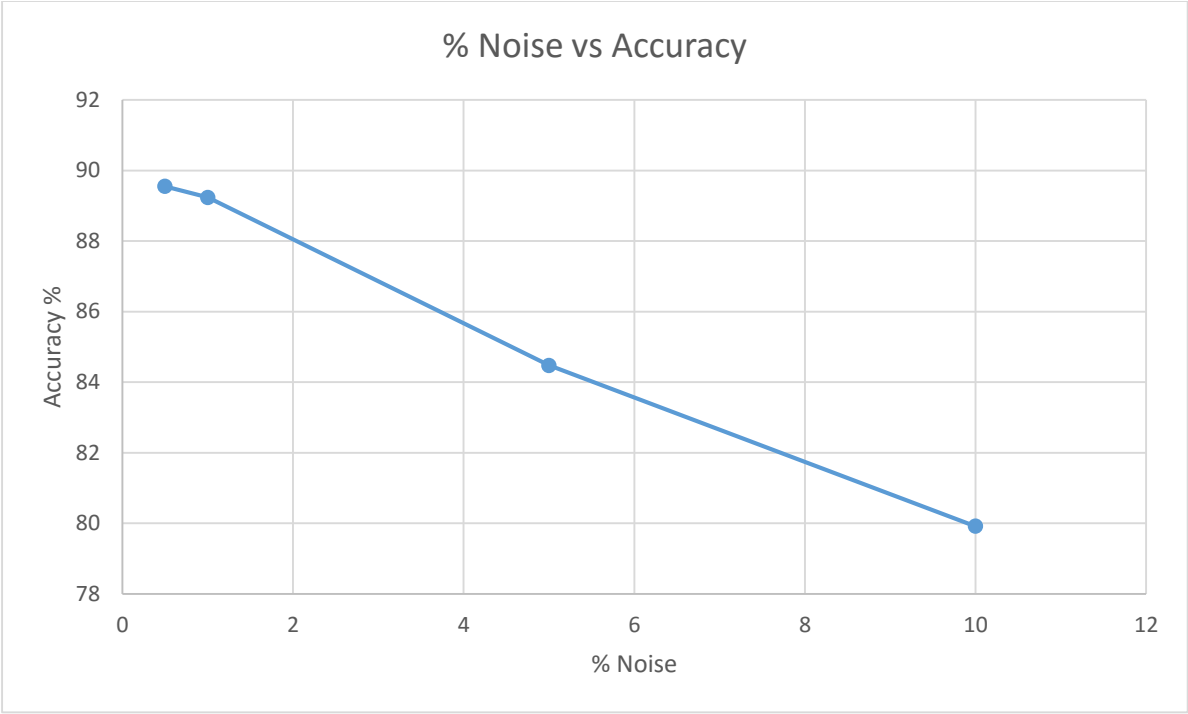
The number of nodes also increases as the threshold is increased as a more complex tree is formed.

EXPERIMENT 2

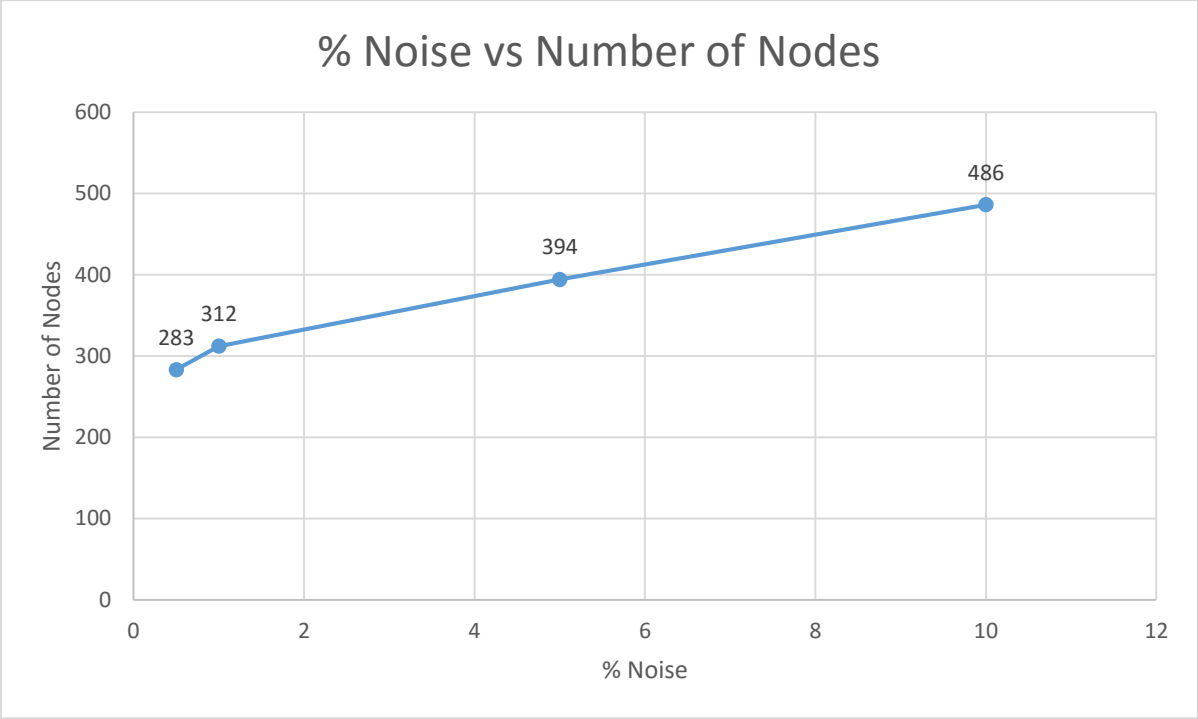
We add noise to the data provided and measure the change in accuracy along with the number of nodes in the tree.

To add noise, we can swap the labels of a percentage of the data. However, this leads to very low noise added to the actual data since lots of labels are '0'. So, most of the labels are swapped '0' with '0' and no significant change in data occurs and negligible change is accuracy is observed.

Hence, instead of swapping, we add noise by inverting the class label of the data points. This adds noise properly and gives good results.



As expected, addition of noise reduces the accuracy of our Decision Tree. The accuracy for 0.5% and 1% noise is similar to the case without noise. As we add 5% and 10% noise, the accuracy reduces dramatically. This can be attributed to the incorrect labels for data instances produced by inverting the class label.



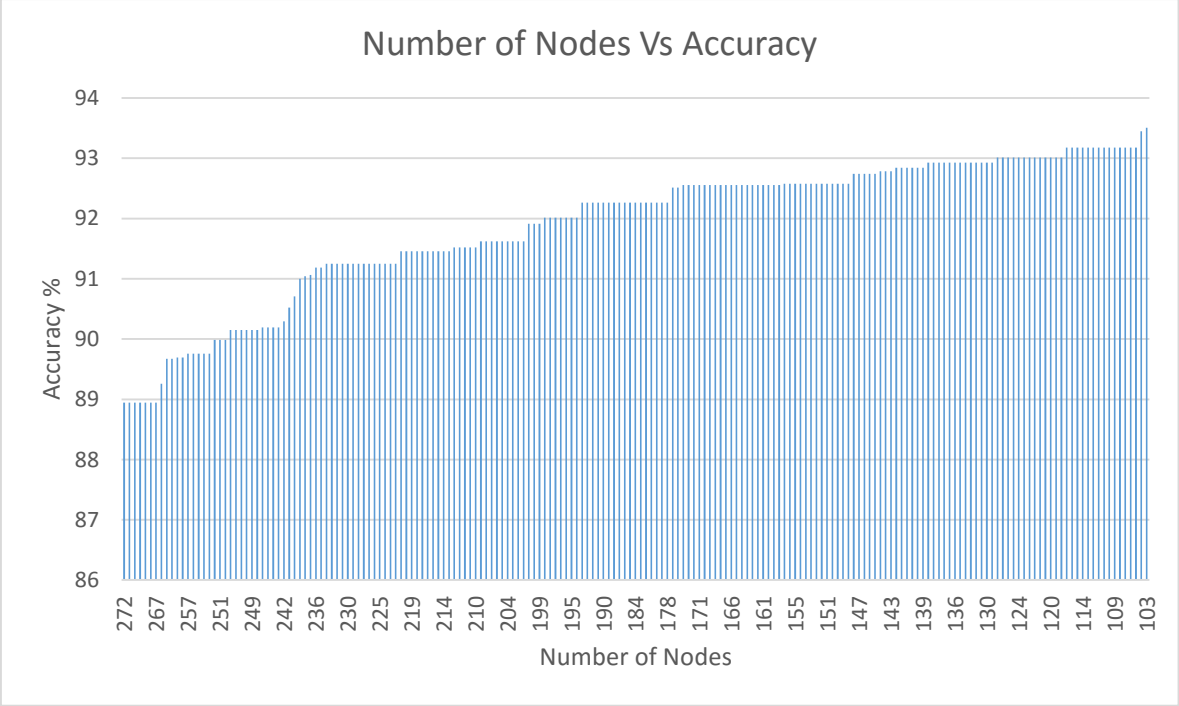
More noise means more variation in labels which earlier used to be purely of a single class. Adding noise reduces the possibility of a node having all data points belonging to a single class. Hence, nodes which were pure earlier now need to be split further. We can observe that the total number of nodes in the tree increases as we add more noise.

If large noise is present in the training data, then Decision Trees are not effective method for classification. The trees tend to become more complex, prone to overfitting, and perform poorly on test data.

EXPERIMENT 3

To reduce overfitting and reduce the complexity of the tree for better generalization, we perform the pruning operation. Here, we analyze each node in the tree and determine if it is needed or not. We compare the accuracy of the unmodified tree with the tree in which current node and its subtree is removed.
If the accuracy increases by removing the node, we remove the node and its subtree. We do this process on all the nodes in the tree. Hence, the final pruned tree is less complex, has fewer nodes, and performs better in the classification task.

In our code, we perform pruning using 'Reduced Error Pruning' method. We perform a DFS on the whole tree to traverse each node. We check if removing each node and its subtree is beneficial and prune the tree. We perform pruning in Top Down manner so that the largest possible subtree whose removal is beneficial is removed first rather than removing small subtrees at lower levels and then moving to upper level nodes. This method reduces the number of nodes better than a Bottom Up approach.



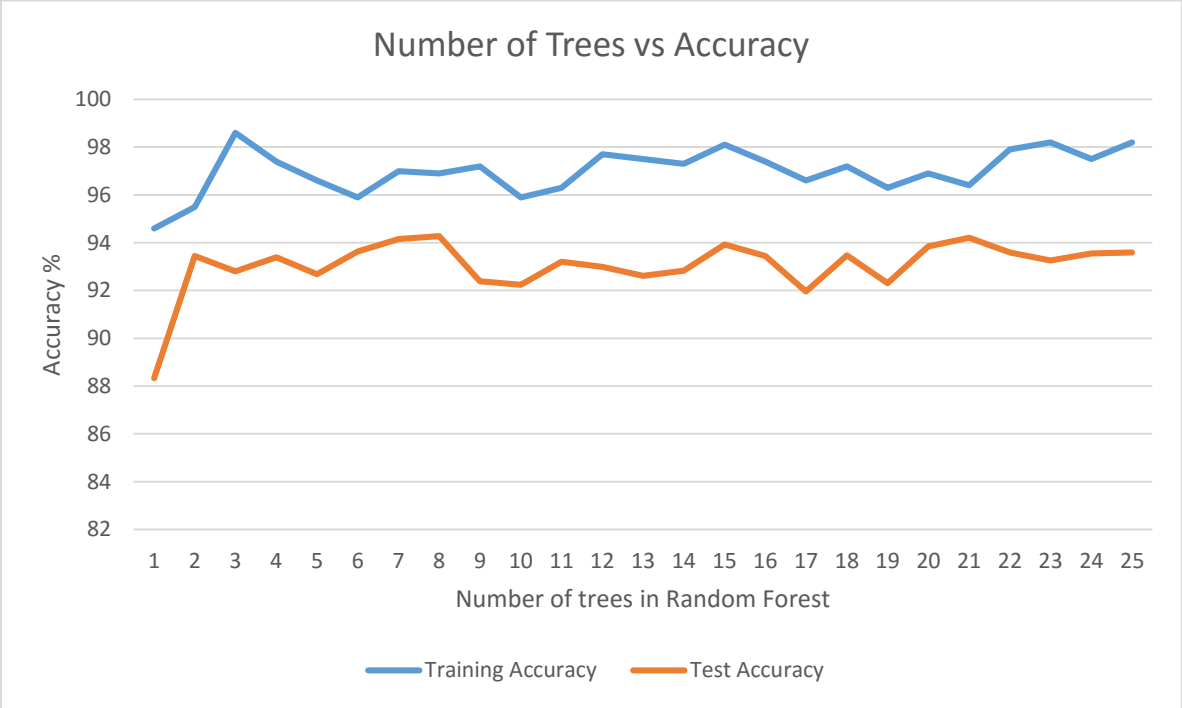
We observe that as we remove the nodes, the training accuracy increases.

Hence, pruning is an effective method to reduce overfitting and reduce the complexity of the tree.

EXPERIMENT 4

A useful method to reduce overfitting is to create multiple trees using a subset of original data and combine their result to get the final classification. This process is called creating 'Random Forests'. We select a random subset of features and use it to create a tree. We repeat the process and create large number of trees to obtain a random forest.

We use the process of feature bagging where we select random √D features out of total given D features for each tree. By decorrelating the trees, we get the result from several independent trees and combine them using Majority voting. Hence, odd number of trees is preferred to avoid ties in classification.



We observe that very small number of trees perform poorly since they do not cover all attributes used for classification. As the trees increase to a significant amount, most attributes are covered in multiple trees and we obtain increase in accuracy of our model.
We reach an equilibrium after a certain number of trees after which, the training and test accuracy remain constant with slight variation and adding more trees does not benefit the random forest.

Hence, we see that creating an ensemble of a large number of simple Decision Trees performs better than a single, complex tree created using the whole data.
Random Forests effectively reduce overfitting and provide better generalization than a single tree.