Report on Decision Tree Pruning in R

Course: CSE 837

Submitted to

Kishan Kumar Ganguly

Lecturer

Institute of Information Technology

University of Dhaka

Submitted by

Tulshi Chandra Das
BSSE0811

Institute of Information Technology

University of Dhaka

Date: 03-10-2019

1. Introduction

In a decision tree algorithm one of the questions arises about the optimal dimension of the final tree. A tree which exceeds the training data and generalizes it badly for new samples. A small tree could not collect significant structural data about the sample area. It is, however, difficult to say when a tree algorithm should stop, because the number of a single additional node will reduce dramatically. It is called the horizon effect issue. A popular approach is to develop the tree until a small amount of cases are included in each node and then use pruning to extract nodes which do nothing further. Pruning should decrease the tree volume without decreasing the predictive precision as evaluated by cross validation systems. The measurements used to optimize efficiency vary in many methods for tree sliding. Pruning is a method for machine learning and search algorithms, which decreases the volume of decision trees through removal of tree segments that provide little classification authority. It decreases the finishing classification's complexity and thereby improves the predictive precision of the overfit.

2. Objectives

The objective of this report is to understand the significance of pruning decision tree. When decision tree is triggered, many branches represent noise-related anomalies in training data. This overflow issue occurs when the learning algorithm develops hypotheses to decrease the exercise setting mistake at the expense of enhanced test set mistakes. To solve this overfitting issue, cutting is needed. Tape a decision tree is a key step towards optimizing the efficiency of computing and classification accuracy. Pruning usually reduces tree size, avoids unnecessary complexity, and helps prevent overfitting of data sets when new data are classified.

3. Methodology

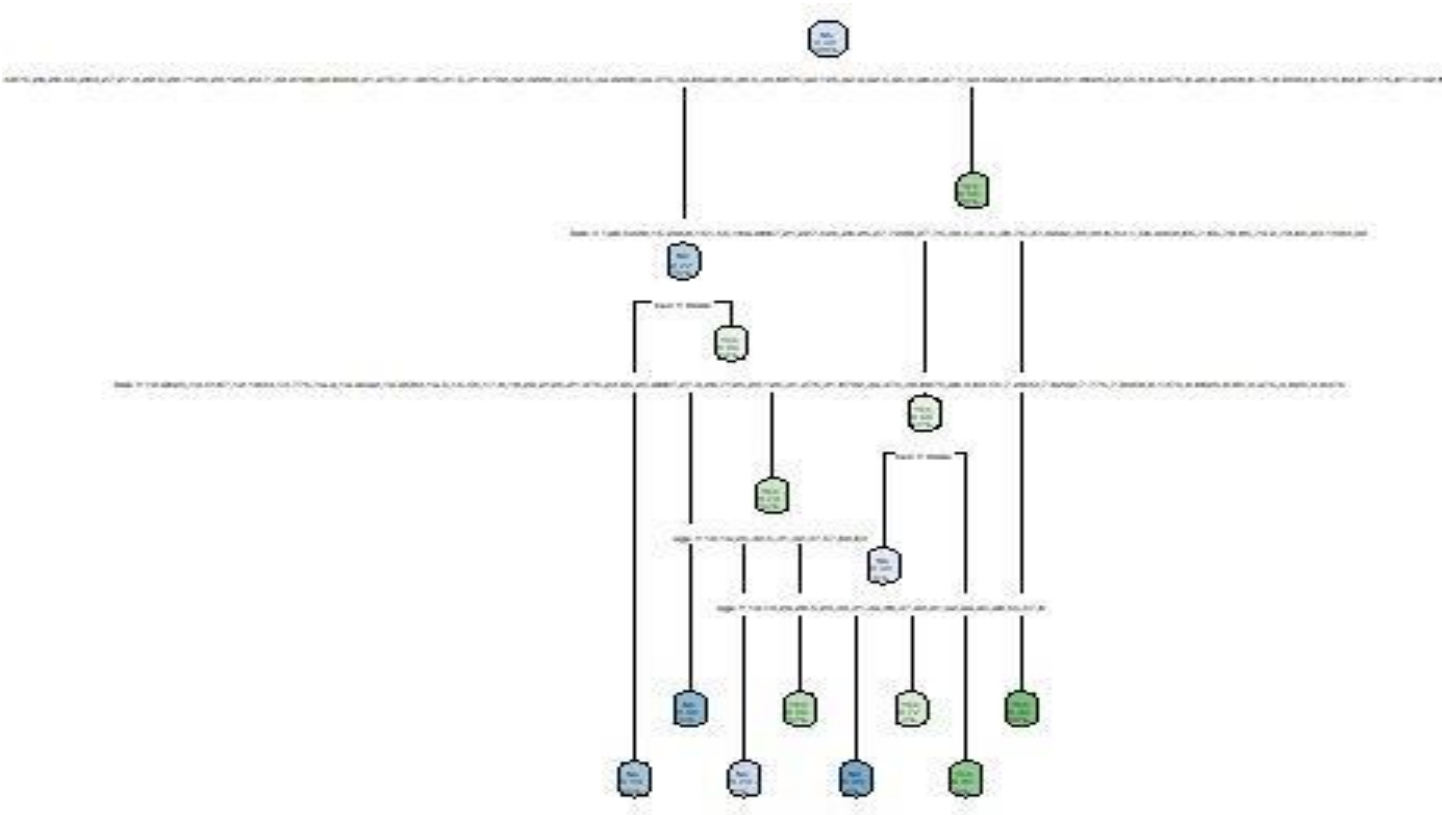
Post-pruning is also known as backward pruning. In this, first Generate the decision tree and then remove non-significant branches. Post-pruning a decision tree implies that we begin by generating the (complete) tree and then adjust it with the aim of improving the classification accuracy on unseen instances. There are two principal methods of doing this. One method that is widely used begins by converting the tree to an equivalent set of rules. Another commonly used approach aims to retain the decision tree but to replace some of its subtrees by leaf nodes, thus converting a complete tree to a smaller pruned one which predicts the classification of unseen instances at least as accurately. There are various methods for the post pruning.

4. Result and Discussion

4.1 pruning the Decision tree

```
lmp <- rpart(survived~., data =  data_train, method = 'class', control = rpart.control(minsplit = 4,minbucket = round(5/ 3), maxdepth = 3,cp=0.001))
print("After pruning tree")
predict_unseen <-predict(lmp,data_test, type = 'class')
table_mat_afterPruning <- table(data_test$survived, predict_unseen)
table_mat_afterPruning
rpart.plot(lmp,type=3,cex = .7, box.palette = lmp('red', 'green'), fallen.leaves = TRUE)
accuracy_Test_afterPruning <- sum(diag(table_mat_afterPruning)) / sum(table_mat_afterPruning)
print(paste('Accuracy for test after pruning', accuracy_Test_afterPruning))
```

Tree After pruning:



Confusion matrix before pruning:

```
    predict_unseen
      No Yes
No   121  40
Yes   21  80
```

Accuracy before pruning:

```
[1] "Accuracy for test before pruning 0.767175572519084"
```

Confusion matrix after pruning:

```
      predict_unseen
        No Yes
  No   125  36
  Yes   24  77
```

Accuracy after pruning:

```
[1] "Accuracy for test after pruning 0.770992366412214"
```

5. Conclusion

In post pruning, first decision tree is induced and then after it is pruned where as in pre pruning; nodes are not expanded during the building phase. As a result, fewer nodes are expanded during the building phase, and thus the complexity of constructing the decision tree is reduced.