**Write R Scripts or use R to perform any mathematical operations while solving the following problems.**

**Note:** The complete solutions to the assignment are split between this document and the Assignment6-Classification1.R file.

**Problem 1: Applying CART, C4.5 and NaiveBayes Algorithms**

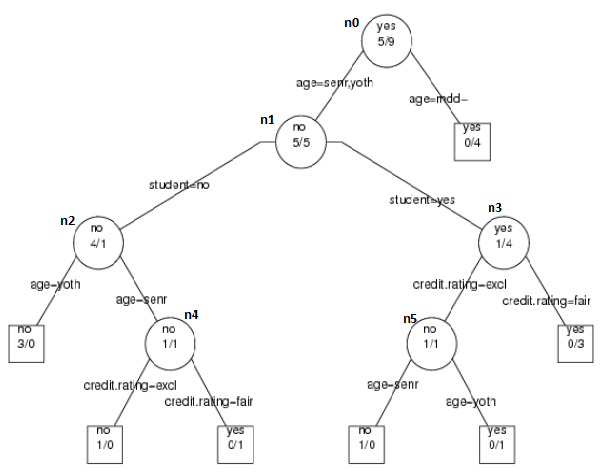
Given the following training data with 4 categorical variables and 1 target variable

****

**Do the following:**

a. Build a decision tree using CART algorithm manually without any pre and post pruning.

**[Ans:]** In the below diagram, ‘a/b’ in node denotes the count of ‘0/1’, ie, ‘non-buyer/buyer’. Also, the 3 age values are renamed to youth, middle-aged, and senior. *See the Assignment6-Classification1.R file for details of the manual calculations.*

**[Ans:]**

b. Predict the class of following test observation using the tree you constructed in part-a:

age<=30, income=medium, student=yes, creditrating=fair

**[Ans:]** We will reach the "root>age=youth,senior>student=yes>credit=fair" path in the tree. So, the outcome will be ‘buyer’

c. Prune the tree built in step using cp parameter for values 0.01, 0.05 and 0.08.

Show the resulting trees you got after pruning.

**[Ans:]**

|  |  |  |  |
| --- | --- | --- | --- |
| **cp=0.01** | **Prune Error** | **Unpruned error** | **Gain** |
| n0 | (5/14)\*(14/14) + (0.01\*1) = 0.367 | (0/3)\*(3/14) + (0/1)\*(1/14) + (0/1)\*(1/14) + (0/1)\*(1/14) + (0/1)\*(1/14) + (0/3)\*(3/14) + (0.01\*7) = 0.07 | -0.297 |
| n1 | (5/10)\*(10/14) + (0.01\*1) = 0.367 | 0 + (0.01\*6) = 0.06 | -0.307 |
| n2 | (1/5)\*(5/14) + (0.01\*1) = 0.0814 | 0 + (0.01\*3) = 0.03 | -0.0514 |
| n3 | (1/5)\*(5/14) + (0.01\*1) = 0.0814 | 0 + (0.01\*3) = 0.03 | -0.0514 |
| n4 | (1/2)\*(2/14) + (0.01\*1) = 0.0814 | 0 + (0.01\*2) = 0.02 | -0.0614 |
| n5 | (1/2)\*(2/14) + (0.01\*1) = 0.0814 | 0 + (0.01\*2) = 0.02 | -0.0614 |

There is no positive gain from collapsing any of the sub-trees. The unpruned error is less compared to the pruned error at each node. So, no pruning required for cp=0.01.

|  |  |  |  |
| --- | --- | --- | --- |
| **cp=0.05** | **Prune Error** | **Unpruned error** | **Gain** |
| n0 | (5/14)\*(14/14) + (0.05\*1) = 0.407 | (0/3)\*(3/14) + (0/1)\*(1/14) + (0/1)\*(1/14) + (0/1)\*(1/14) + (0/1)\*(1/14) + (0/3)\*(3/14) + (0.05\*7) = 0.35 | -0.297 |
| n1 | (5/10)\*(10/14) + (0.05\*1) = 0.407 | 0 + (0.05\*6) = 0.30 | -0.107 |
| n2 | (1/5)\*(5/14) + (0.05\*1) = 0.1214 | 0 + (0.05\*3) = 0.15 | 0.0286 |
| n3 | (1/5)\*(5/14) + (0.05\*1) = 0.1214 | 0 + (0.05\*3) = 0.15 | 0.0286 |
| n4 | (1/2)\*(2/14) + (0.05\*1) = 0.1214 | 0 + (0.05\*2) = 0.10 | -0.0214 |
| n5 | (1/2)\*(2/14) + (0.05\*1) = 0.1214 | 0 + (0.05\*2) = 0.10 | -0.0214 |

The sub-trees n2 and n3 have equal gain, and also reduce the same number of leaves. So, they are both collapsed.

# Tree still remains unchanged. Why?

rpart(buy ~ age + income + student + credit.rating, computer,control=rpart.control(cp=0.05,minsplit=2))

|  |  |  |  |
| --- | --- | --- | --- |
| **cp=0.08** | **Prune Error** | **Unpruned error** | **Gain** |
| n0 | (5/14)\*(14/14) + (0.08\*1) = 0.437 | (0/3)\*(3/14) + (0/1)\*(1/14) + (0/1)\*(1/14) + (0/1)\*(1/14) + (0/1)\*(1/14) + (0/3)\*(3/14) + (0.08\*7) = 0.56 | 0.123 |
| n1 | (5/10)\*(10/14) + (0.08\*1) = 0.437 | 0 + (0.08\*6) = 0.48 | 0.043 |
| n2 | (1/5)\*(5/14) + (0.08\*1) = 0.1514 | 0 + (0.08\*3) = 0.24 | 0.0886 |
| n3 | (1/5)\*(5/14) + (0.08\*1) = 0.1514 | 0 + (0.08\*3) = 0.24 | 0.0886 |
| n4 | (1/2)\*(2/14) + (0.08\*1) = 0.1514 | 0 + (0.08\*2) = 0.16 | 0.0086 |
| n5 | (1/2)\*(2/14) + (0.08\*1) = 0.1514 | 0 + (0.08\*2) = 0.16 | 0.0086 |

The sub-tree n0 has the greatest gain. So, it is collapsed during post-pruning.

# Tree still remains unchanged. Why?

rpart(buy ~ age + income + student + credit.rating, computer,control=rpart.control(cp=0.05,minsplit=2))

**Note:** *The pruning expected at cp=0.05 actually happens for cp=0.1. Why?*

*The pruning expected at cp=0.08 actually happens for cp=0.3. Why?*

**Problem 2: Applying Cost-complexity Pruning**

Given the following tree, apply the cost complexity pruning discussed in class for cp values of 0, 1/20, 1/10, 1/8, 1/3, 1. Do the following for each of cp value separately:

a. Compute the pruned and unpruned cost at every internal node.

b. Find out the pruned tree.



|  |  |  |  |
| --- | --- | --- | --- |
| **cp=0** | **Prune Error** | **Unpruned error** | **Gain** |
| T2 | (25/100)\*(100/100) + (0\*1) = 0.25 | (1/51)\*(51/100) + (5/26)\*(26/100) + (0/20)\*(20/100) + (0\*3) = 0.06 | -0.19 |
| T3 | (21/49)\*(49/100) + (0\*1) = 0.21 | (5/26)\*(26/100) + (0/20)\*(20/100) + (0\*2) = 0.05 | -0.16 |

No pruning done since unpruned error is less than prune error.

|  |  |  |  |
| --- | --- | --- | --- |
| **cp=1/20** | **Prune Error** | **Unpruned error** | **Gain** |
| T2 | (25/100)\*(100/100) + (1/20\*1) = 0.30 | (1/51)\*(51/100) + (5/26)\*(26/100) + (0/20)\*(20/100) + (1/20\*3) = 0.21 | -0.09 |
| T3 | (21/49)\*(49/100) + (1/20\*1) = 0.26 | (5/26)\*(26/100) + (0/20)\*(20/100) + (1/20\*2) = 0.15 | -0.11 |

No pruning done since unpruned error is less than prune error.

|  |  |  |  |
| --- | --- | --- | --- |
| **cp=1/10** | **Prune Error** | **Unpruned error** | **Gain** |
| T2 | (25/100)\*(100/100) + (1/10\*1) = 0.35 | (1/51)\*(51/100) + (5/26)\*(26/100) + (0/20)\*(20/100) + (1/10\*3) = 0.36 | 0.01 |
| T3 | (21/49)\*(49/100) + (1/10\*1) = 0.31 | (5/26)\*(26/100) + (0/20)\*(20/100) + (1/10\*2) = 0.25 | -0.06 |

T2 has prune error lesser than unpruned error. So, prune at T2.

|  |  |  |  |
| --- | --- | --- | --- |
| **cp=1/8** | **Prune Error** | **Unpruned error** | **Gain** |
| T2 | (25/100)\*(100/100) + (1/8\*1) = 0.375 | (1/51)\*(51/100) + (5/26)\*(26/100) + (0/20)\*(20/100) + (1/8\*3) = 0.435 | 0.06 |
| T3 | (21/49)\*(49/100) + (1/8\*1) = 0.335 | (5/26)\*(26/100) + (0/20)\*(20/100) + (1/8\*2) = 0.30 | -0.035 |

T2 has prune error lesser than unpruned error. So, prune at T2.

|  |  |  |  |
| --- | --- | --- | --- |
| **cp=1/3** | **Prune Error** | **Unpruned error** | **Gain** |
| T2 | (25/100)\*(100/100) + (1/3\*1) = 0.583 | (1/51)\*(51/100) + (5/26)\*(26/100) + (0/20)\*(20/100) + (1/3\*3) = 1.06 | 0.477 |
| T3 | (21/49)\*(49/100) + (1/3\*1) = 0.543 | (5/26)\*(26/100) + (0/20)\*(20/100) + (1/3\*2) = 0.716 | 0.173 |

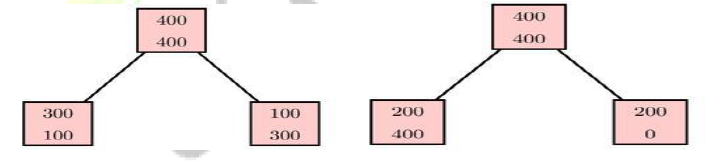
T2 gains more from pruning. So, prune at T2.

|  |  |  |  |
| --- | --- | --- | --- |
| **cp=1** | **Prune Error** | **Unpruned error** | **Gain** |
| T2 | (25/100)\*(100/100) + (1\*1) = 1.25 | (1/51)\*(51/100) + (5/26)\*(26/100) + (0/20)\*(20/100) + (1\*3) = 3.06 | 1.81 |
| T3 | (21/49)\*(49/100) + (1\*1) = 1.21 | (5/26)\*(26/100) + (0/20)\*(20/100) + (1\*2) = 2.05 | 0.84 |

T2 gains more from pruning. So, prune at T2.

**Problem 3: Impurity vs Misclassification Rate for tree growth**

Find the misclassification rate of following subtrees independently. Which measure do find useful to grow the tree: misclassification rate or impurity?



**[Ans:]**

Misclassification rate of left subtree = (100/400)\*(400/800) + (100/400)\*(400/800) = 0.25

Misclassification rate of right subtree = (200/600)\*(600/800) + (0/200)\*(200/800) = 0.25

We see that both the subtrees have the same misclassification rate. So, using the misclassification measure, we would not be able to differentiate between the two splits.

However, intuition suggests that split in the right sub-tree is better because it produces a ‘pure’ node. gini() is a measure of impurity used in CART.

Gini(left split) = gini(100/400, 300/400)\*(400/800) + gini(300/400, 100/400)\*(400/800) = 0.375

Gini(right split) = gini(200/600, 400/600)\*(600/800) + gini(200/200, 0/200)\*(200/800) = 0.333

Since, Gini(right split) is lower, the CART algorithm (which uses gini measure) would use this split, thus agreeing with our intuition.

So, I would use ‘impurity’ to grow trees.