Overview of Classification & Regression Tree (CART)

CART

Based on Chew C. H. (2020) textbook: AI, Analytics and Data Science. Vol 1., Chap 8.

Objective

1. What is the CART model? [The essential features]
2. Using the CART Model:
   - Decision Rules.
   - Model Predictions.
   - Overall Error.
   - Variable Importance.
3. Generating the CART Model:
   - Phrase 1: Growing the Tree to the Max.
   - Phrase 2: Pruning the Tree to the Min.
   - Selecting the Optimal Tree.

Key Concepts in Phrase 1: Growing Tree to the Maximum

Categorical Y	Continuous Y

• Node: $\hat{Y} = majority$
• Misclassification Error
  • Node level
  • Tree level
• Node Purity
• Gini Index
• Entropy
	• Node: $\hat{Y} = \bar{Y}$
	• SSE at the node level
	• MSE at the Tree level

Surrogates

Key Concepts in Phrase 2: Pruning Tree to the Minimum

- What is pruning?
- Complexity Penalty [Hyperparameter]:
  - Textbook: α
  - Rpart package: cp
- Total Cost of using the Model = Model Prediction Error + Complexity Penalty

Node Purity, Misclassification Error and Gini Index (for Categorical Y only)

CART

Based on Chew C. H. (2020) textbook: AI, Analytics and Data Science. Vol 1., Chap 8.

Optimal Tree Selection

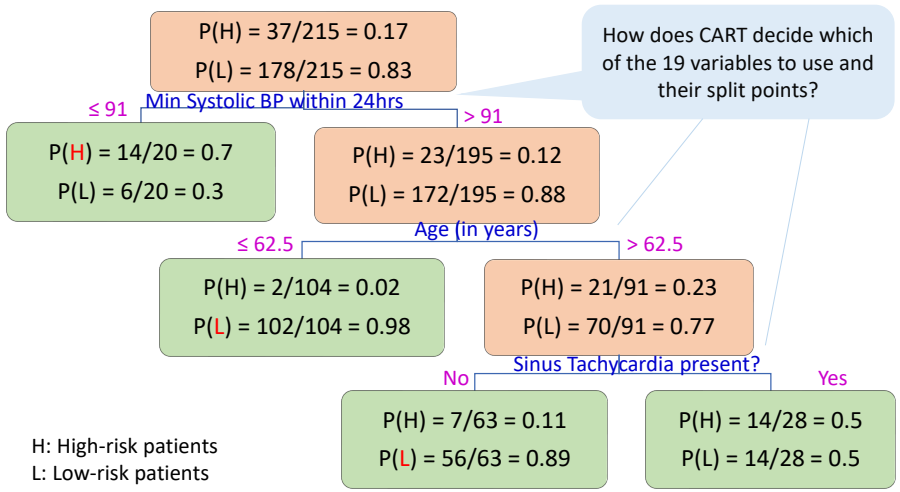
- 10 fold Cross Validation (CV)
- 1 Standard Error Rule

Next: The Invention of CART

- Understand the circumstances that led to the invention of the CART model.
- Learn the key features that make CART so attractive to business users.

CART Results for Heart Attack Prognosis

Source: Breiman, Friedman, Olshen, and Stone (1983). Classification and Regression Trees. Wadsworth.



H: High-risk patients
L: Low-risk patients

- There are 4 terminal nodes (i.e. 4 decision rules).
- Model predictions at terminal nodes are based on majority.
- Misclassification error at a terminal node = minority proportion.

Key Concepts in Phrase 1: Growing Tree to the Maximum

Categorical Y	Continuous Y

• Node: $\hat{Y} = majority$
• Misclassification Error
  • Node level
  • Tree level
• Node Purity
• Gini Index
• Entropy
	• Node: $\hat{Y} = \bar{Y}$
	• SSE at the node level
	• MSE at the Tree level

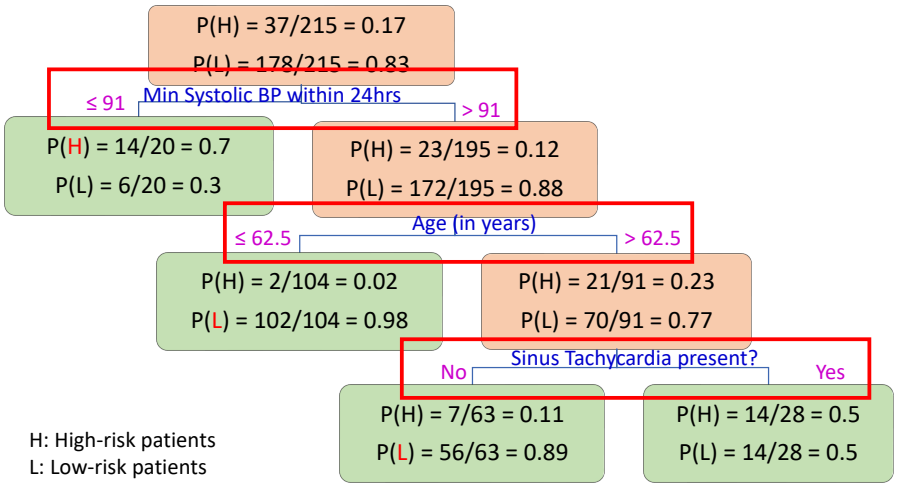
Surrogates	

Choosing the Best Splitting Variable and Best Split Point

- At each node during growth phrase, CART must consider and test:
  - all X variables and,
  - all possible values in that X variable
  - to determine the best binary split.
- A split is good if it results in purer child nodes, on average.
- The best split will result in the purest possible child nodes, on average.
- The theoretical purest child node is the node with 100% in one Y category (i.e. 0% in all other Y categories).
- The theoretical most impure child node is the node with the same proportion in each of the Y category (i.e. uniform distribution for Y).
- CART model prediction accuracy directly affected by node purity.

CART Results for Heart Attack Prognosis

Source: Breiman, Friedman, Olshen, and Stone (1983). Classification and Regression Trees. Wadsworth.



H: High-risk patients
L: Low-risk patients

- A split is defined by a specific X variable and a specific value.
- Only the best split is used (and shown) to generate 2 child nodes.
- Result is purer when averaged across the 2 child nodes.

To automate the search for the best split, need to specify the selection criteria in terms of a formula

- Formulas to determine the best binary split:
  - Gini Index, or
  - Entropy.
- Gini index is preferred by Prof Leo Breiman.
  - Default setting in many software including Rpart package.
- Misclassification Error is not a good formula to determine the best binary split.
  - Set as an exercise question.

Example: Y = category A or B.

1. Best case: Highest purity occurs when 0% cat A and 100% cat B, or 100% cat A and 0% cat B.
2. Worst case: Highest impurity occurs when 50% cat A and 50% cat B.
3. Preference: Impurity of {40% cat A, 60% cat B > Impurity of {30% cat A, 70% cat B}
4. Symmetry: Impurity of {40% cat A, 60% cat B} = Impurity of {60% cat A, 40% cat B}

- All formulas used to determine the best split must demonstrate the above results.

Entropy, Gini and Misclassification Error

Let $p(y \mid t)$ denote the proportion of cases belonging to class y at tree node t.

The three measures can be simplified if y has only two possible outcomes {0, 1}.

Let p represent p(y = 1 | t), then 1 – p represent p(y = 0 | t). The 3 impurity measures become:

Entropy(t) =

If Y has k categories

$$-\sum_{y=0}^{k} p(y \mid t)\log_2 p(y \mid t)$$

If Y has only 2 categories

$$= -\left[p(0 \mid t)\log_2 p(0 \mid t) + p(1 \mid t)\log_2 p(1 \mid t)\right] = -\left[(1-p)\log_2(1-p) + p\log_2 p\right]$$

*Note: 0*log(0) is defined as 0.*

If Y has k categories

Gini(t) $= 1 - \sum_{y=0}^{k}[p(y \mid t)]^2$

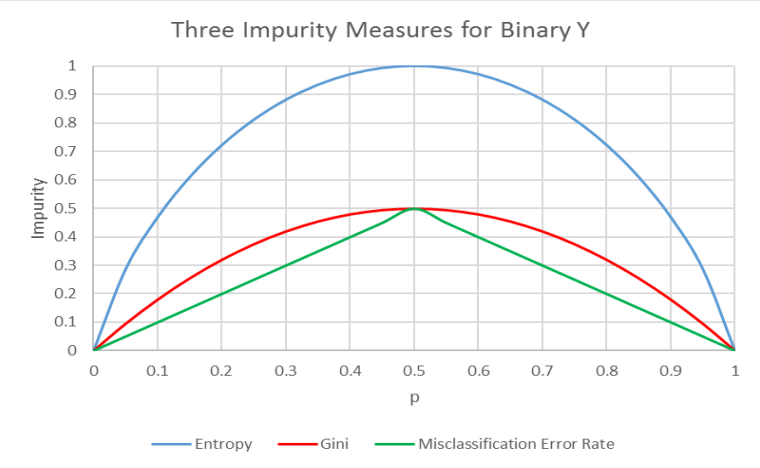
If Y has only 2 categories

$$= 1 - \left[p(0 \mid t)^2 + p(1 \mid t)^2\right] = 1 - \left[(1-p)^2 + p^2\right] = 2p(1-p)$$

[Majority Rule] Misclassification error, r(t) = $1 - \max\{p(y \mid t)\} = 1 - \max\{p, 1-p\}$

If Y has k categories

If Y has only 2 categories

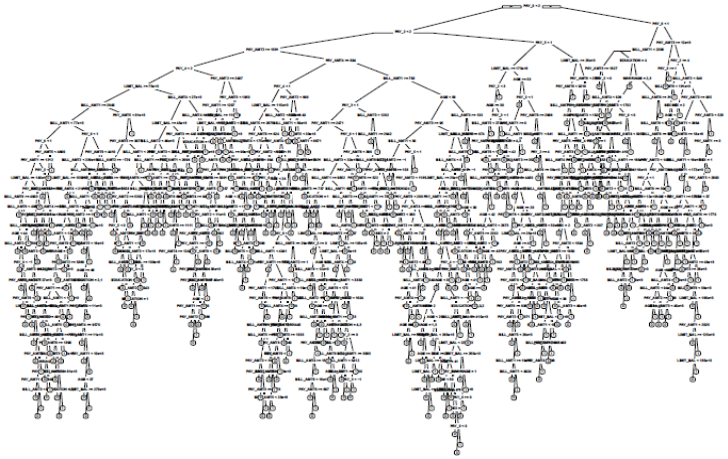


- Gini and Misclassification range from 0 (best) to 0.5 (worst).
- Entropy range from 0 (best) to 1 (worst).
- A numerical exercise with a small dataset will clarify how to use these formulas to determine the best split.

Growing the Tree to the maximum

- At each step, the best splitting variable and it's best split point is found, and applied to create two child nodes.
- The process continues, until a (lenient) stopping criteria is met.
- The result is a very large tree, which is likely to suffer from overfitting.
- Pruning is then required to prune the Tree to the minimum (root node) in Phrase 2.
  - Pruning generates a sequence of smaller and smaller Trees.
  - The optimal Tree is somewhere between the maximal Tree and the minimal Tree.

A Maximal Tree generated from a dataset with 21,000 cases and 23 X variables to predict Credit Risk



Next: Pruning the Tree

- Understand the Pruning process.
- Weakest Link Pruning.
  - Define the strength of a node's link to all its descendants.
- Completely determined by the data
  - No human judgement.
  - Pure Machine Learning.

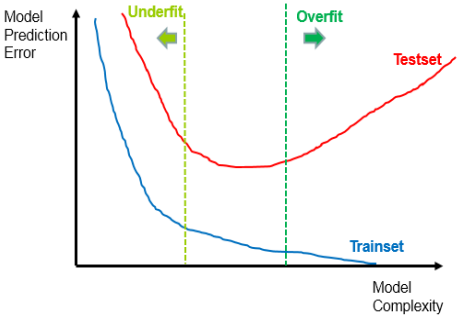
Complexity Penalty Parameter and Pruning the Tree

CART

Based on Chew C. H. (2020) textbook: AI, Analytics and Data Science. Vol 1., Chap 8.

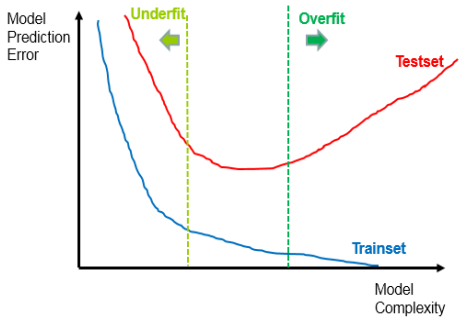
The Key to Pruning is to incorporate Complexity Penalty explicitly into Total Cost of using the Model

- The risk of Overfitting is omnipresent in all models.
- Overfitting is far less obvious than underfitting.
- The model cannot see the Testset during Tree growing and Tree Pruning.



The Key to Pruning is to incorporate Complexity Penalty explicitly into Total Cost of using the Model

- Observe the trade-off when Overfitting begins.
- Overfitting:
  - As Model increases in Complexity,
  - Trainset error decreases,
  - Testset error increases.
- Such trade-off can be incorporated into a formula.



Trade-offs incorporated into a formula via a Complexity Penalty Parameter α

Total Cost of CART = Misclassification Error + Total Complexity Cost

$$R_\alpha(T) = R(T) + \alpha\,|\,T\,|$$

Note:

In Breiman (1984) textbook, Complexity of tree |T| = number of terminal nodes.

- As model complexity |T| increases, R(T) decreases but α × |T| increases.
- α is a positive constant.
  - The penalty per unit model complexity.

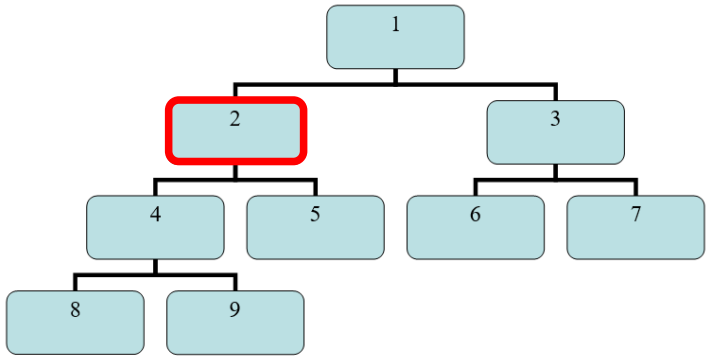
Pruning the Maximal Tree with Complexity Penalty

- Given a big tree that probably overfits, need to "prune" and simplify the tree by applying penalty for tree complexity.
- The bigger the penalty, the smaller the tree.
- The complexity of the tree can be measured by:
  - Number of terminal nodes (leaves), or
  - Number of splits
- What do we mean by "pruning the tree"?

Pruning the Tree (Before)

Tree Diagram BEFORE Pruning Node 2.

Terminal Nodes are 5, 6, 7, 8, 9. Internal nodes are 1, 2, 3, 4.

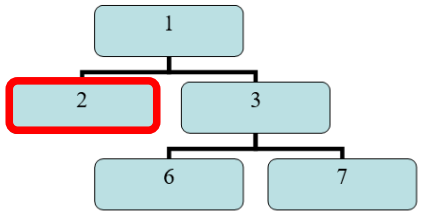


- Did you notice the node labelling convention? Implications?

Pruning the Tree (After)

Tree Diagram AFTER Pruning Node 2.

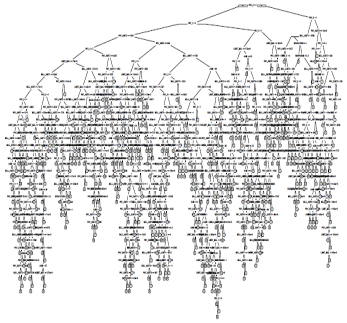
Terminal Nodes are 2, 6, 7. Internal nodes are 1, 3.



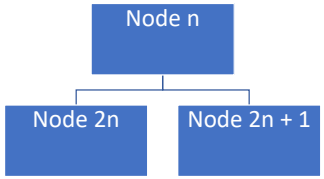
Thus, pruning at node t means all descendants of node t are cut off.

Node t still remains in the tree and becomes a terminal node.

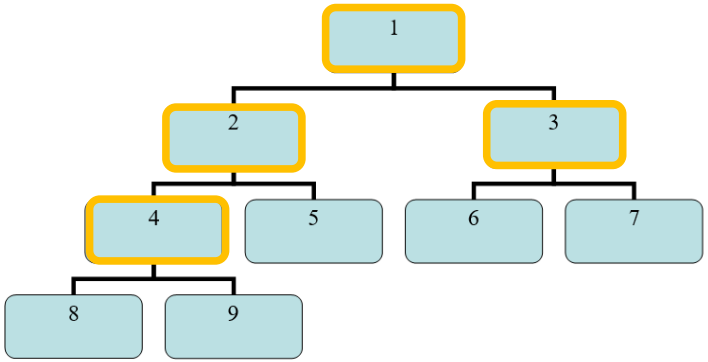
Node Labelling Convention



- Especially important if you have a big tree.
- A big tree cannot be visualized with the details at each node.
- By standardizing the node labelling convention, given any node, we know:
  - Who is the left child node (if any)
  - Who is the right child node (if any)
  - Who is the parent node (if any)
- i.e. no need to see the big tree to trace the ancestry.



Where to prune?



- How to determine the “best” node to prune away?
- Ans: Weakest link pruning.
- How to determine which link is weakest?

Total Cost, Adjusted for Model Complexity

Define $\tilde{T}$ to be the set of all **terminal nodes** in the tree T.

Using r(t) and p(t), where p(t) is the proportion of all cases in terminal node t, define the contribution of terminal node t to the total misclassification error as

$$R(t) = r(t) \times p(t) \qquad t \in \tilde{T}$$

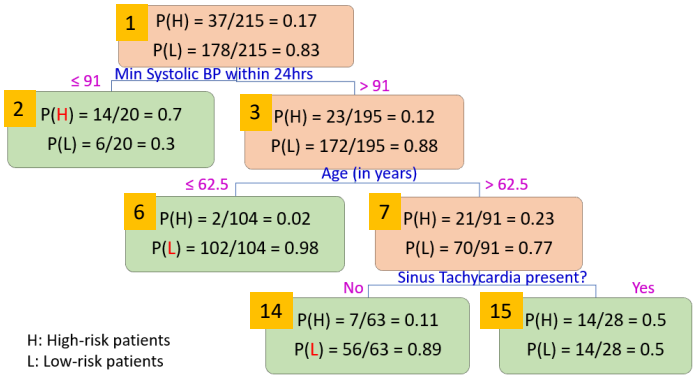
Overall Misclassification Error of the Tree T:

$$R(T) = \sum_{t \in \tilde{T}} R(t)$$

Complexity Adjusted Total Cost of the Tree T:

$$R_\alpha(T) = R(T) + \alpha |T|$$

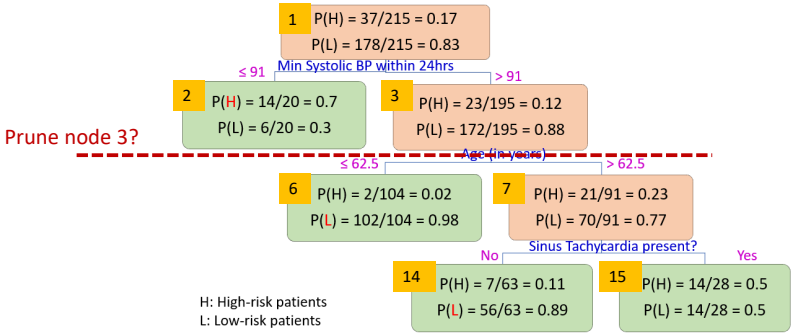
Where $|T|$ represents the **size of the tree** e.g. number of terminal nodes, and α represents the **penalty cost per unit complexity**. This is a costing mechanism to penalize complexity. Large α result in small tree.



- $\tilde{T} = \{2, 6, 14, 15\}$
- r(2) = 0.3, p(2) = 20/215, R(2) = 0.3*20/215 ≈ 0.027907
- r(6) = 0.02, p(6) = 104/215, R(6) = 0.02*104/125 ≈ 0.009674
- r(14) = 0.11, p(14) = 63/215, R(14) = 0.11*63/215 ≈ 0.03223
- r(15) = 0.5, p(15) = 28/215, R(15) = 0.5*28/215 ≈ 0.065116
- R(T) ≈ 0.1349
- $R_\alpha(T) = 0.1349 + 4\alpha$

Pruning triggered at certain values of alpha

- Only when alpha is big enough will the weakest link be pruned away.
- Chicken Rice Analogy:
  - Current cost in food centre: $3 per pack. Consume 10 packs per month.
  - What if cost increase to $3.10 per pack?
  - What if cost increase to $5 per pack?
  - What if cost increase to $5.20 per pack?
  - What if cost increase to $20 per pack?
- CART:
  - Current cost: α = 0 units per terminal node. i.e. no penalty for model complexity.
  - What if α = 0.1?
  - What if α = 0.3?
- As alpha increases, pruning is triggered only at certain few values of alpha, until only the root node is left.



- Is α = 0.0001 sufficient to prune node 3?
- Without pruning node 3: $R_\alpha(T) = 0.1349 + 4\alpha \approx 0.1353$
- If prune at node 3, node 3 becomes terminal node, and r(3) = 0.12, p(3) = 195/215, R(3) ≈ 0.1088, R(T) = R(2) + R(3) ≈ 0.1367, $R_\alpha(T) = 0.1367 + 2\alpha = 0.1369$
- Do not prune node 3 since total cost is lower without pruning.

- Is α = 0.02 sufficient to prune node 3?
- Without pruning node 3: $R_\alpha(T) = 0.1349 + 4\alpha \approx 0.2149$
- If prune at node 3, node 3 becomes terminal node, $R_\alpha(T) = 0.1367 + 2\alpha = 0.1767$
- Prune node 3 since total cost is lower with pruning.
- i.e. we compare the consequences of the two scenarios.

What minimum value of alpha will trigger the pruning?

- Compare the two equations from the two scenarios and solve for alpha.
- Total cost without pruning at node 3 = Total cost if prune at node 3

$$0.1349 + 4\alpha = 0.1367 + 2\alpha$$

$$2\alpha = 0.0018$$

$$\alpha = 0.0009$$

- α = 0.0009 is the minimum value of α required to prune at node 3. i.e. the prune trigger. [Note that this value is only an approximation as we used only 4 decimal places in the first equation.]
- Any value of alpha above the minimum value will also prune away node 3.
- Repeat the same analysis at all other internal nodes (i.e. nodes 1, 7) to get their prune triggers.
- The weakest link will be the lowest value among all the prune triggers. E.g. at node X.
- Prune away node X. Recompute R(T), $R_\alpha(T)$, and repeat the analysis at all internal nodes to get the second weakest link, and so on….
- The pruning sequence is thus specified and completely determined by the data.

Next: rpart demonstration on a dataset

- CART implemented in Rpackage rpart.
- Phrase 1: Grow Tree to max.
- Phrase 2: Prune Tree to min.
- Pruning sequence determines a sequence of sub trees.
- One of the sub-tree is the optimal tree for CART model prediction.
- How to use the CART model to make predictions.

Rpart Demo on Categorical Y

Dataset: upgradeCard.csv
Rscript: upgradeCard ver1.R

CART

Based on Chew C. H. (2020) textbook: AI, Analytics and Data Science. Vol 1., Chap 8.

Differences in the Total Cost Formulas

- Total Cost of CART = Misclassification Error + Total Complexity Cost
- Textbook formula:

$$R_\alpha(T) = R(T) + \alpha|T|$$

  - |T|: Number of terminal nodes.

- rpart formula:

$$R_{cp}(T) = R(T) + cp \times R(T_1) \times |T|$$

  - |T|: Number of splits. *Note: Number of terminal nodes = Number of splits + 1.*
  - $R(T_1)$: Root Node Error.
  - cp: Complexity Parameter. *Note:* $cp \times R(T_1) = \alpha$

Dataset with categorical Y:

- Predict if Customer will upgrade credit card or not?
- Two X variables:
  - Spending on credit card over last 12 months;
  - Supplementary card – Y/N?

	A	B	C
1	Upgrade	Spending	SuppCard
2	N	8025	N
3	Y	8593	Y
4	N	1219	N
5	N	2032	N
6	N	3245	N
7	N	4012	N
8	N	5166	N
9	Y	10512	Y
10	N	10557	Y

```
> summary(custdata1.dt)
 Upgrade     Spending       SuppCard
 N:18    Min.    :   50   N:17
 Y:13    1st Qu.: 6358   Y:14
         Median : 8760
         Mean   : 8405
         3rd Qu.:10550
         Max.   :14804
```

rpart() with method = 'class' for categorical Y

Categorical Y: method = 'class'
Continuous Y: method = 'anova'

```
18  set.seed(2004)          # For randomisation in 10-fold CV.
19
20  # rpart() completes phrase 1 & 2 automatically.
21  # Change two default settings in rpart: minsplit and cp.
22  m2 <- rpart(Upgrade ~ Spending + SuppCard, data = custdata1.dt, method = 'class',
23              control = rpart.control(minsplit = 2, cp = 0))
```

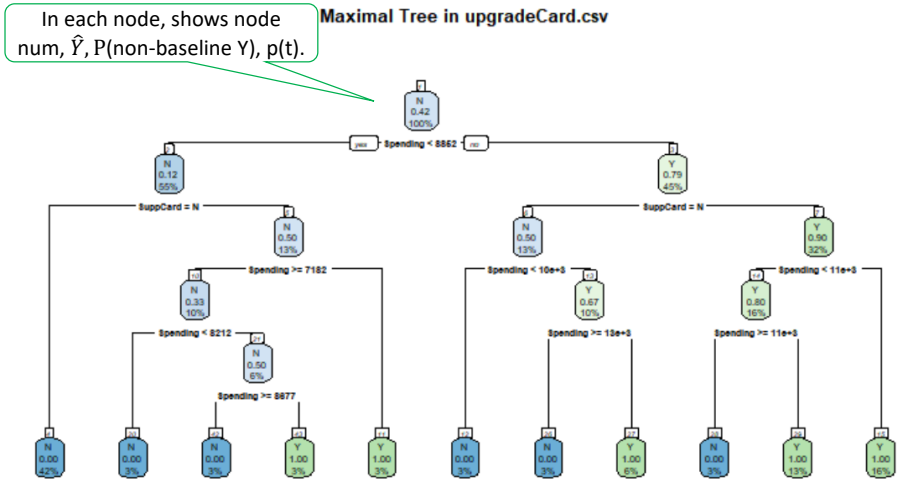
Default = 20. Changed to 2 due small sample size. "The minimum number of observations that must exist in a node in order for a split to be attempted." -- ?rpart.control

Default = 0.01. Changed to 0 to ensure grow tree to the max in phrase 1.

Plot the Tree Diagram with rpart.plot()

```
25  # plots the maximal tree and results.
26  rpart.plot(m2, nn= T, main = "Maximal Tree in upgradeCard.csv")
```

In each node, shows node num, $\hat{Y}$, P(non-baseline Y), p(t).




```
28  # prints the maximal tree m2 onto the console.
29  print(m2)
```

How to read the numbers:

```
> print(m2)
n= 31

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 31 13 N (0.5806452 0.4193548)
   2) Spending< 8851.5 17  2 N (0.8823529 0.1176471)
     4) SuppCard=N 13  0 N (1.0000000 0.0000000) *
     5) SuppCard=Y 4  2 N (0.5000000 0.5000000)
      10) Spending>=7182 3  1 N (0.6666667 0.3333333)
        20) Spending< 8212 1  0 N (1.0000000 0.0000000) *
        21) Spending>=8212 2  1 N (0.5000000 0.5000000)
          42) Spending>=8676.5 1  0 N (1.0000000 0.0000000) *
          43) Spending< 8676.5 1  0 Y (0.0000000 1.0000000) *
      11) Spending< 7182 1  0 Y (0.0000000 1.0000000) *
   3) Spending>=8851.5 14  3 Y (0.2142857 0.7857143)
     6) SuppCard=N 4  2 N (0.5000000 0.5000000)
      12) Spending< 10120 1  0 N (1.0000000 0.0000000) *
      13) Spending>=10120 3  1 Y (0.3333333 0.6666667)
        26) Spending>=12893.5 1  0 N (1.0000000 0.0000000) *
        27) Spending< 12893.5 2  0 Y (0.0000000 1.0000000) *
     7) SuppCard=Y 10  1 Y (0.1000000 0.9000000)
      14) Spending< 11496 5  1 Y (0.2000000 0.8000000)
        28) Spending>=10534.5 1  0 N (1.0000000 0.0000000) *
        29) Spending< 10534.5 4  0 Y (0.0000000 1.0000000) *
      15) Spending>=11496 5  0 Y (0.0000000 1.0000000) *
```

Node number 4:
Decision Rule:
Spending < 8851.5 and SuppCard = No.
13 cases of Upgrade = No, 0 cases of Upgrade = Yes.
Hence $\hat{Y}$ = No.
100% Upgrade = No, 0% Upgrade = Yes.
Is a terminal node.

Prune Triggers and Pruning Sequence with printcp(). Can be used to identify optimal tree.

```
31  # prints out the pruning sequence and 10-fold CV errors, as a table.
32  printcp(m2)
```

```
Root node error: 13/31 = 0.41935

n= 31

        CP nsplit rel error  xerror     xstd
1 0.615385      0   1.00000 1.00000 0.21134
2 0.051282      1   0.38462 0.69231 0.19441
3 0.038462      4   0.23077 0.76923 0.20021
4 0.000000     10   0.00000 0.84615 0.20492
```

Prune Triggers

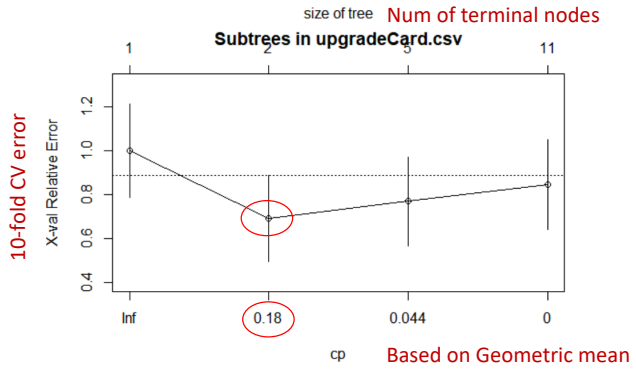
10-fold CV error

Pruning Sequence read from bottom up.

$$R_{cp}(T) = R(T) + cp \times R(T_1) \times |T|$$

Plot subtrees from the pruning sequence with plotcp(). Can be used to identify optimal tree.

```
34  # Display the pruning sequence and 10-fold CV errors, as a chart.
35  plotcp(m2, main = "Subtrees in upgradeCard.csv")
```



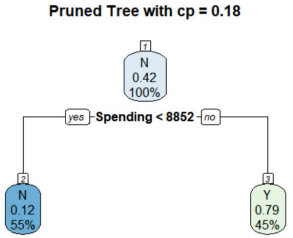
Get a specific subtree via prune() by pruning the maximal tree m2 with a specific value of cp

```
41  cp1 <- 0.18
42
43  m3 <- prune(m2, cp = cp1)
44
45  printcp(m3)
46
47  # plots the tree m3 pruned using cp1.
48  rpart.plot(m3, nn= T, main = "Pruned Tree with cp = 0.18")
```

```
Root node error: 13/31 = 0.41935

n= 31

        CP nsplit rel error  xerror     xstd
1 0.61538      0   1.00000 1.00000 0.21134
2 0.18000      1   0.38462 0.69231 0.19441
```

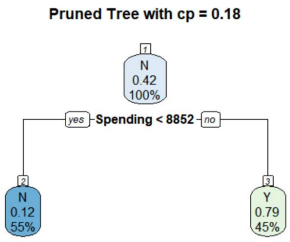


CART Model Predictions

CART can auto-handle missing values in trainset, testset or what-ifs.

```
50  # Test CART model m3 predictions
51  testcases <- data.frame(Spending = c(8000, 10000, NA), SuppCard <- c("Y", NA, NA))
52
53  cart.predict <- predict(m3, newdata = testcases, type = "class")
54
55  results <- data.frame(testcases, cart.predict)
```

	Spending	SuppCard	cart.predict
1	8000	Y	N
2	10000	*NA*	Y
3	*NA*	*NA*	N



Next Video Series: CART(Part 2)

- How CART automatically handle missing values – Surrogates.
- A better way to identify optimal tree.
  - The subtree with minimum CV error is an unstable solution.
- 10 fold cross validation.
- CART for continuous Y.