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* × $R(T_1) = \alpha$

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))
```

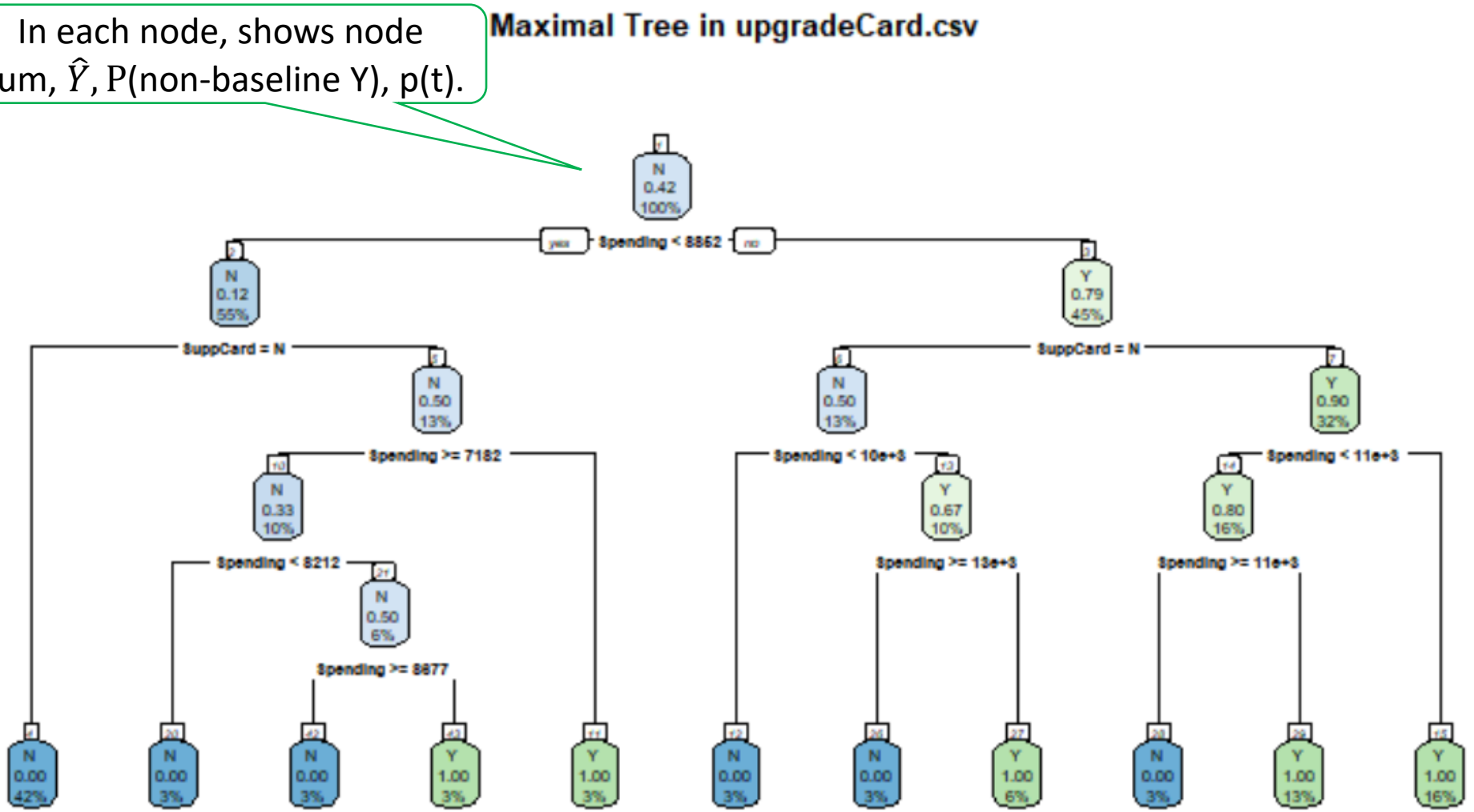
Default = 20. Changed to 2 due small sample size. “The minimum number of observations that must exist in a node in order for a split to be attempted.” -- ?rpart.control

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

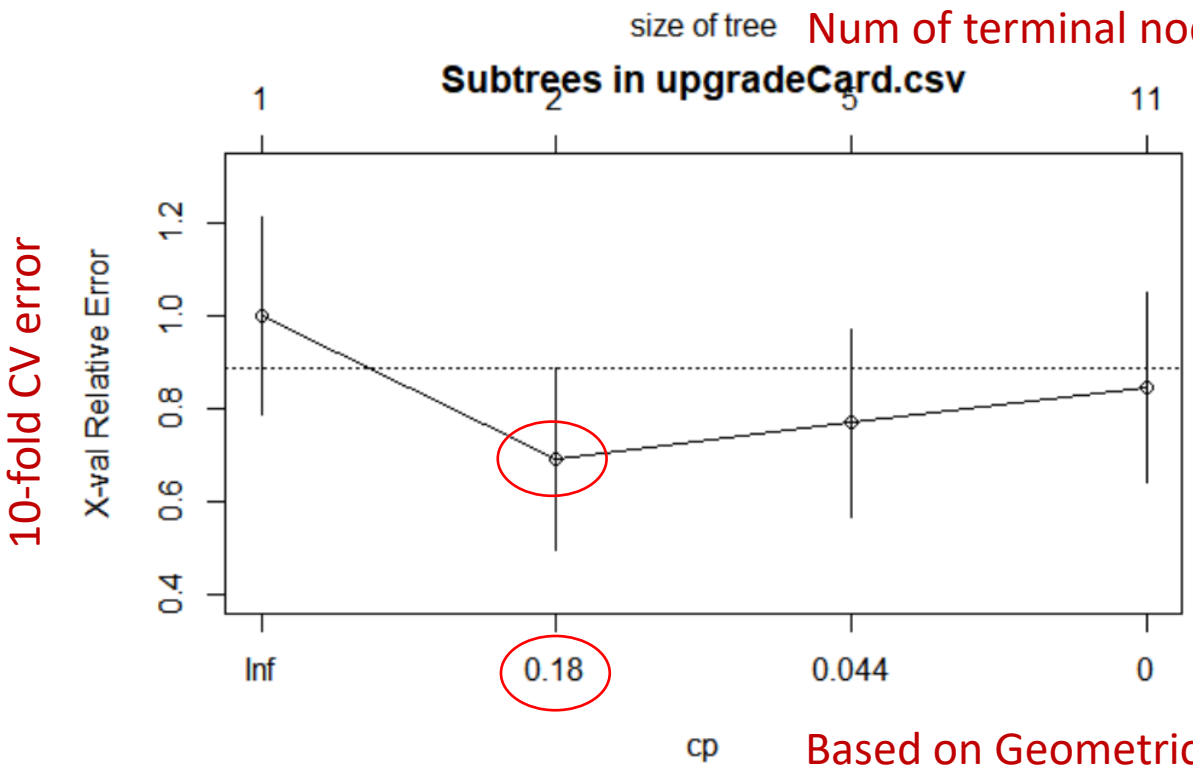
10-fold CV error

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")
```



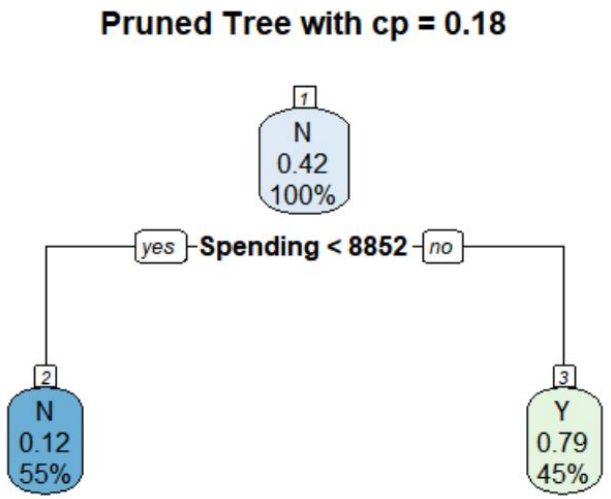
Get a specific subtree via prune() by pruning the maximal tree m2 with a specific value of cp

```
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
```

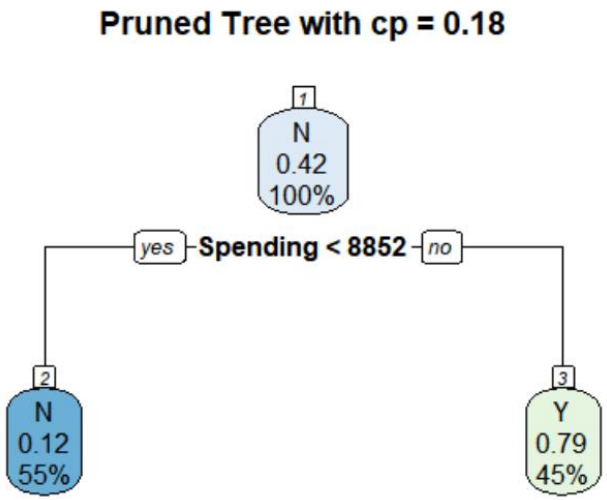


CART Model Predictions

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.