Lecture6 Handout

Elizabeth Colantuoni

4/12/2021

I. Objectives and acknowledgements

In this lecture we will explore the *basics* of classification and regression trees (CART) as an alternative to constructing parametric linear and logistic regression models.

Upon completion of this session, you will be able to do the following:

- Understand, explain and use classification and regression trees (CART) as a complementary approach to regression analysis with binary outcome
- Predict medical expenditures and the occurence of big expenditures from MSCD and other explanatory variables using both logistic regression and CART
- Use ROC curves to measure the quality of prediction and compare quality of prediction across prediction approaches

This lecture was adapted from a lectures prepared by Shannon Wongvibulsin and Scott Zeger in 140.653 and 140.654 years past. Notation and definitions from Chapter 9 of HTF.

II. Set up

For illustration purposes, we will continue to work with the NMES dataset with a focus on building predictions models for medical expenditures, both mean expenditures (continuous variable, regression tree) and the risk of having a big expenditure (binary variable, classification tree).

```
load('./nmes.rdata')
d1 = nmes
d1[d1=='.'] = NA

## Create the necessary variables:
d1$bigexp=ifelse(d1$totalexp>1000,1,0)
d1$mscd=ifelse(d1$lc5+d1$chd5>0,1,0)

dat=data.frame(e=log(d1$totalexp+1),age=d1$lastage,
  mscd=factor(d1$mscd),beltuse=as.numeric(d1$beltuse),
  educate=as.numeric(d1$educate),married=factor(d1$marital),
  poverty=as.numeric(d1$povstalb),male=factor(d1$male))
dat = dat[complete.cases(dat),]

dat1=data.frame(big=factor(d1$bigexp),age=d1$lastage,
  mscd=factor(d1$mscd),beltuse=factor(d1$beltuse),
  educate=factor(d1$educate),
```

```
  poverty=factor(d1$povstalb),male=factor(d1$male))
dat1 = dat1[complete.cases(dat1),]
```

III. Introduction to CART

A. History

Classification and regression trees (CART) were introduced in 1984 by Leo Breiman, Professor of Statistics at UC Berkeley, Richard Olshen, Jerry Friedman and others. Breiman spent the rest of his career developing important statistical learning (SL) (aka "machine learning - ML") tools including "bagging", "boosting" and "random forests". You can encounter Leo Breiman at https://www.youtube.com/watch?v=JIXeMj1zwVU.

An overview of his approach to statistical learning, offered as a challenge to the conservative wing of his field, is Breiman, L. (2001). "Statistical Modeling: the Two Cultures". Statistical Science. 16 (3): 199–215. doi:10.1214/ss/1009213725. JSTOR 2676681. An excellent treatment of CART and other SL methods for classification and regression are given in Chapters 9-15 in our text book by Hastie, Tibshirani and Friedman. Details about the use of the R function *rpart* are in the Mayo Clinic Tech Report by Therneau and Atkinson on the course website.

For the basics of CART, the notes that follow are derived from Chapter 9.2.2 of HTF.

B. CART as binary splitting procedure

CART is a simple algorithm to minimize a measure of prediction error. For example, in the problem of predicting medical expenditures Y, we will use the mean squared error (MSE) as that measure. The CART prediction function is the mean value c_m within each of M non-overlapping rectangular X sub-spaces, $R_1, \ldots, R_M$ that the algorithm identifies. That is, CART approximates $E(Y|X)$ by a step function

$$f(X) = \sum_{m=1}^{M} c_m I(X \in R_m).$$

and the algorithm seeks to miminize

$$\sum_{i=1}^{n} \{y - \sum_{m=1}^{M} c_m I(X \in R_m)\}^2$$

where c_m is estimated by $\bar{y}_m$, the mean response for all the observations in R_m.

$$\hat{c}_m = ave(y_i | x_i \in R_m)$$

How do we know how to define R_m, i.e. which variable gets split and where to put the splits?

CART uses a greedy algorithm! Starting with all of the data, consider a splitting variable j and split point s, and define the pair of half-planes:

$$R_1(j,s) = \{X | X_j \leq s\} \text{ and } R_2(j,s) = \{X | X_j > s\}$$

Then we seek the splitting variable j and split point s that solve:

$$\min_{j,s} \left[\min_{c1} \sum_{x_i \in R_1(j,s)} (y_i - c1)^2 + \min_{c2} \sum_{x_i \in R_2(j,s)} (y_i - c2)^2 \right]$$

For any choice j and s, the inner minimization is solved by:

$$\hat{c}_1 = ave(y_i|x_i \in R_1) \text{ and } \hat{c}_2 = ave(y_i|x_i \in R_2)$$

Having found the best split, the process is repeated in each of the two resulting splits and so on.

C. Illustration of binary splitting / model

```
include_graphics("images/figure9.2.jpg")
```

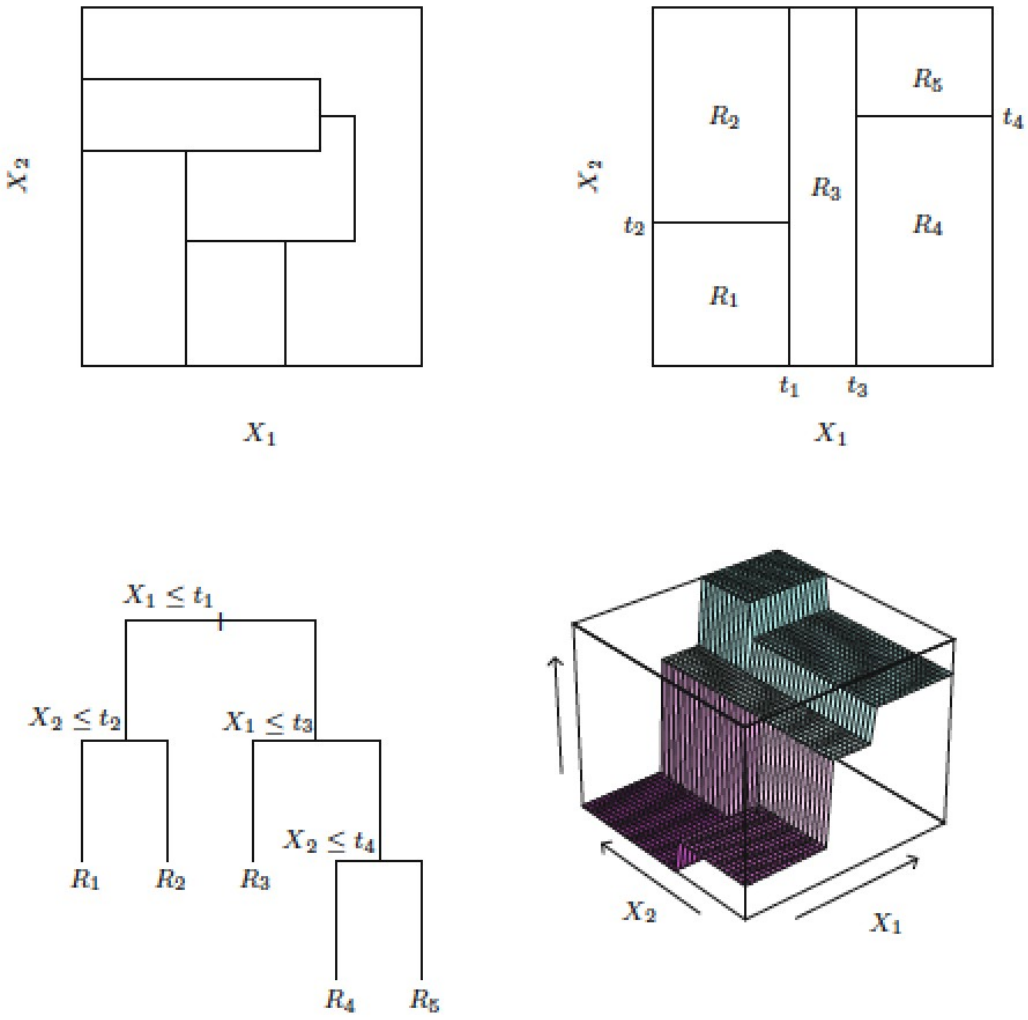


FIGURE 9.2. *Partitions and CART. Top right panel shows a partition of a two-dimensional feature space by recursive binary splitting, as used in CART, applied to some fake data. Top left panel shows a general partition that cannot be obtained from recursive binary splitting. Bottom left panel shows the tree corresponding to the partition in the top right panel, and a perspective plot of the prediction surface appears in the bottom right panel.*

D. When do we know how to stop splitting?

But when does the algorithm stop?

The tree size, i.e. the number of groups M, is controlled by user-set parameters. It is recommended to first grow a large tree, call this tree T_0, determining M by stopping the process only when a minimum node size (say 5) is reached. Subsequently, this large tree is pruned using *cost-complexity pruning*. For example, working backwards from the large tree, keep only the splits that yield improvement in MSE.

What is the *complexity parameter*? In CART, they refer to the complexity parameter as "cp". Define a subtree $T \subset T_0$ to be any tree that can be obtained by pruning T_0, that is, collapsing any number of its internal (non-terminal) nodes. We index terminal nodes by m,with node m representing region R_m. Let $|T|$ denote the number of terminal nodes in T. Letting

$$N_m = \#\{x_i \in R_m\}$$

$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$$

$$Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$$

Then the cost complexity parameter is:

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|$$

The CART procedure finds the tree $T_\alpha \subset T_0$ that minimizes $C_\alpha(T)$ for each α. The parameter $\alpha \geq 0$ defines the trade-off between tree size and the goodness of fit. Larger values (smaller) of α result in smaller (larger) trees. You can show that for each α there is a unique smallest subtree T_α that minimizes $C_\alpha(T)$. See details in HTF on how these are trees are identified. The procedure yeilds an estimate of α based on a cross-validated procedure and for each α a corresponding optimal tree T_α. The final tree can be selected among the T_α that yeilds the smallest cross-validated sums of squares.

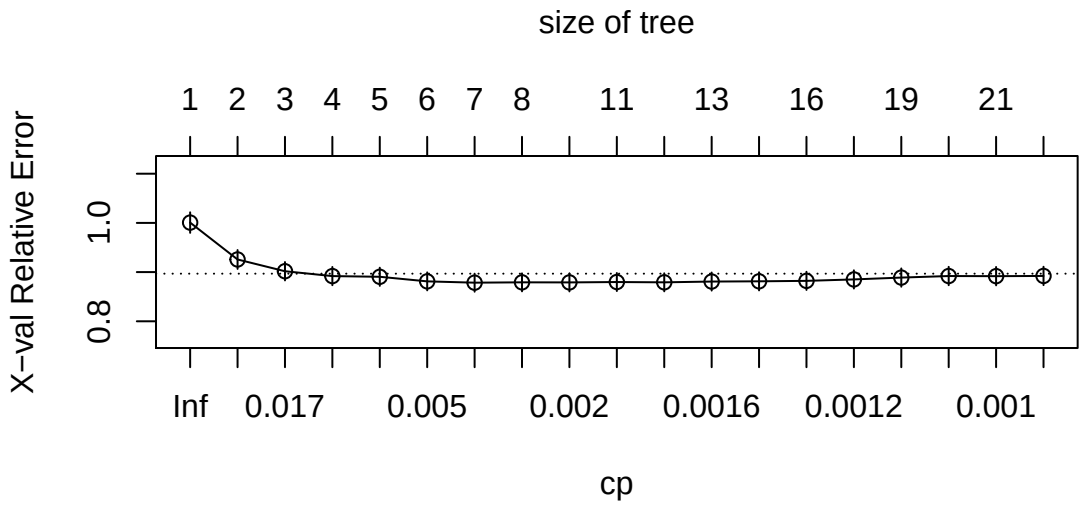
IV. Predicting medical expenditures

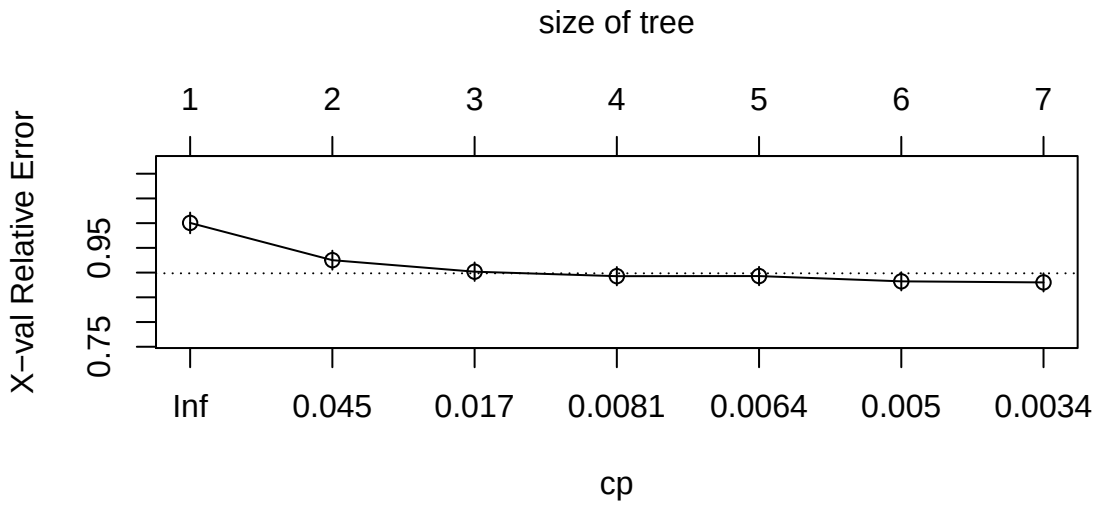
We will demonstrate the use of regression trees by predicting the mean of the $log(totalexp + 1)$. We will create a training and testing/validation dataset, grow a large tree, and identify the *complexity parameter* that yeilds the lowest cross-validated MSE which results in the selected tree.

```
set.seed(123454321)
dat.train=dat[train<-sample(1:nrow(dat),floor(nrow(dat)/2)),]
dat.test=dat[-train,]
```

```
## Fit a first tree settting the "cp" parameter to 0.001
tree0=rpart(e~.,data=dat.train,method="anova",control=rpart.control(minsize=20,cp=.001))
par(mfrow=c(2,1),mar=c(5,5,5,1))
plotcp(tree0)

tree=rpart(e~.,data=dat.train,method="anova",control=rpart.control(minsize=20,cp=.0030))
plotcp(tree)
```





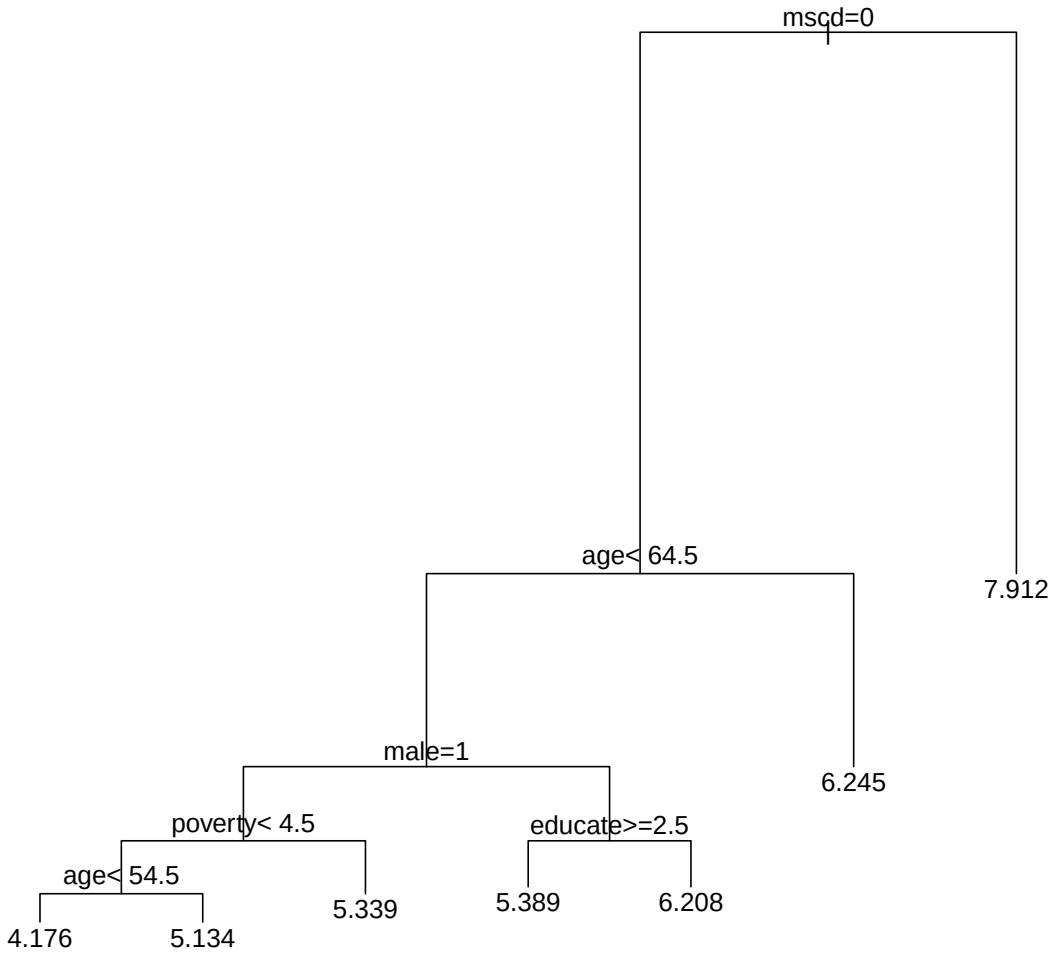

```
## Print the cptable
tree$cptable
```

```
##           CP nsplit rel error    xerror       xstd
## 1 0.075345979      0 1.0000000 1.0004711 0.02089362
## 2 0.026861159      1 0.9246540 0.9251182 0.01917377
## 3 0.010320801      2 0.8977929 0.9018284 0.01866638
## 4 0.006407304      3 0.8874721 0.8927497 0.01854963
## 5 0.006319377      4 0.8810648 0.8929794 0.01855866
## 6 0.003940150      5 0.8747454 0.8822941 0.01838046
## 7 0.003000000      6 0.8708052 0.8801729 0.01836491
```

```
## Print the variable importance
tree$variable.importance
```

```
##      mscd       age   educate      male   married   poverty   beltuse
## 3109.1510 1282.8392  461.4138  425.8878  289.6406  260.7690   22.2820
```

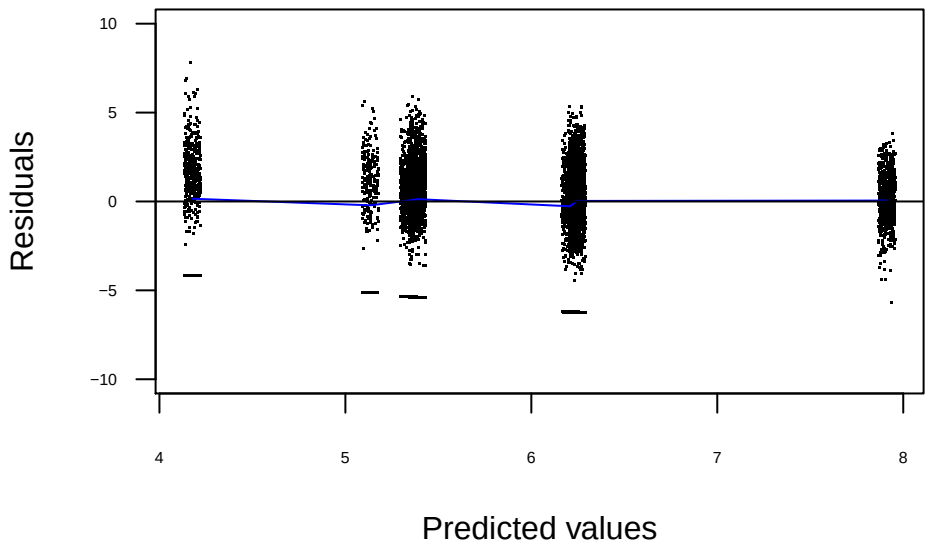
```r
## Plot the tree and add labels
plot(tree)
text(tree,pretty=3)
```



Now let's predict $log(totalexp + 1)$ for the test data set that was left out when the tree was trained.

```
# Generate predicted values
tree.yhat=predict(tree,newdata=dat.test,na.action=na.pass)
# Compute the residuals
res.tree.test=dat.test$e-tree.yhat
# Compute the MSE = sums of squared residuals / n for the
# test/validation dataset
mse.tree.test=sum(res.tree.test^2)/length(res.tree.test)

# Plot the residuals vs. predicted values
o=order(tree.yhat)
par(mar=c(4,4,1,1))
plot(jitter(tree.yhat,factor=6),jitter(res.tree.test,factor=6),
     pch=".",xlab="Predicted values",
     ylab="Residuals",ylim=c(-10,10),las=1,cex.axis=0.5)
lines(tree.yhat[o],predict(lm(res.tree.test~ns(tree.yhat,5)))[o],
      type="l",col="blue");abline(h=0,col="black")
```

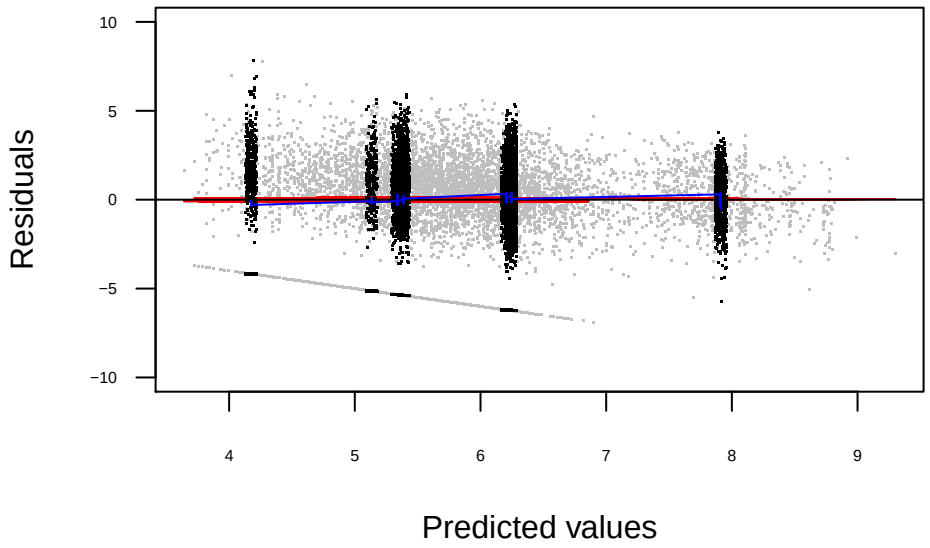


In addition, we can compare the prediction from the regression tree to a parametric model we construct.

```
# Fit a parametric model, using the training data
model=lm(data=dat.train,e~ns(age,2)*male*mscd + as.factor(poverty)*as.factor(educate))
# Get predictions, residuals based on test/validation sample
model.yhat=predict(model,newdata=dat.test)
res.model.test=dat.test$e-model.yhat
# Compute the MSE for the parametric model
mse.model.test=sum(res.model.test^2)/length(res.model.test)

# Compare the residuals vs. predicted values for
# the regression tree and parametric model
par(mar=c(4,4,1,1))
plot(model.yhat,res.model.test,
     pch=".",xlab="Predicted values",col="gray",
     ylab="Residuals",ylim=c(-10,10),las=1,cex.axis=0.5)
lines(model.yhat[o],predict(lm(res.model.test~ns(tree.yhat,5)))[o],
      type="l",col="red");abline(h=0,col="black")
```

```
points(jitter(tree.yhat,factor=6),jitter(res.tree.test,factor=6),pch=".",col="black")
o=order(tree.yhat)
lines(tree.yhat[o],predict(lm(res.tree.test~ns(model.yhat,5)))[o],pch="-",col="blue")
```



The means squared errors for the models are CART: 5.806; Linear Model: 5.68

V. Some comments about CART

Some comments about CART to read further about in THF Chapter 9.2.4:

- interactions at the core
- must produce rectangles in X space
- specific tree, but not necessarily predictions, are unstable to perturbations in X - makes interpretation of tree unreliable
- poorly represents smooth functional relationships
- has natural extensions to the GLM family
- handles missing data reasonably well through surrogate variables
- tends to favor variable selection for factors with lots of levels

VI. Classification Trees

A. Splitting and pruning criteria

The procedure for constructing the classification tree is the same as the regression tree *except* we need to replace the sums of squared residuals criteria used for node splitting and pruning with a measure more appropriate for classification.

For regression trees, we use the impurity measure

$$Q_m(T) = \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$$

For classification trees, define

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

Observations in node m are classified in category k based on the category k with highest $\hat{p}_{mk}$.

Different measures of node impurity $Q_m(T)$ include the misclassification error, Gini index and cross-entropy or deviaince. When we are constructing a classification tree for a binary response where $p = Pr(Y = 1)$, these measures are:

$$\text{Misclassification error: } 1 - max(p, 1-p)$$

$$\text{Gini index: } 2p(1-p)$$

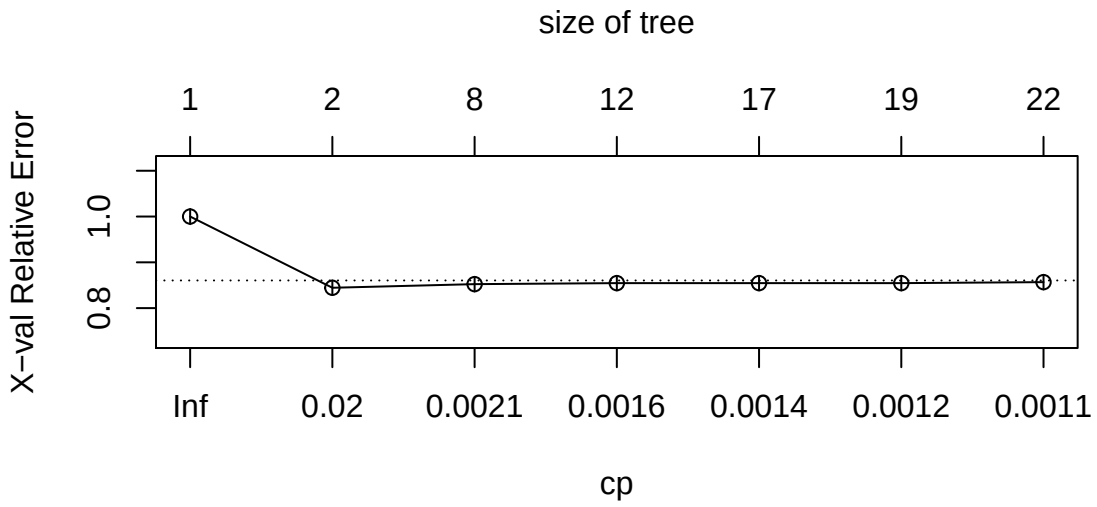
$$\text{Cross entropy or deviance: } -plog(p) - (1-p)log(1-p)$$

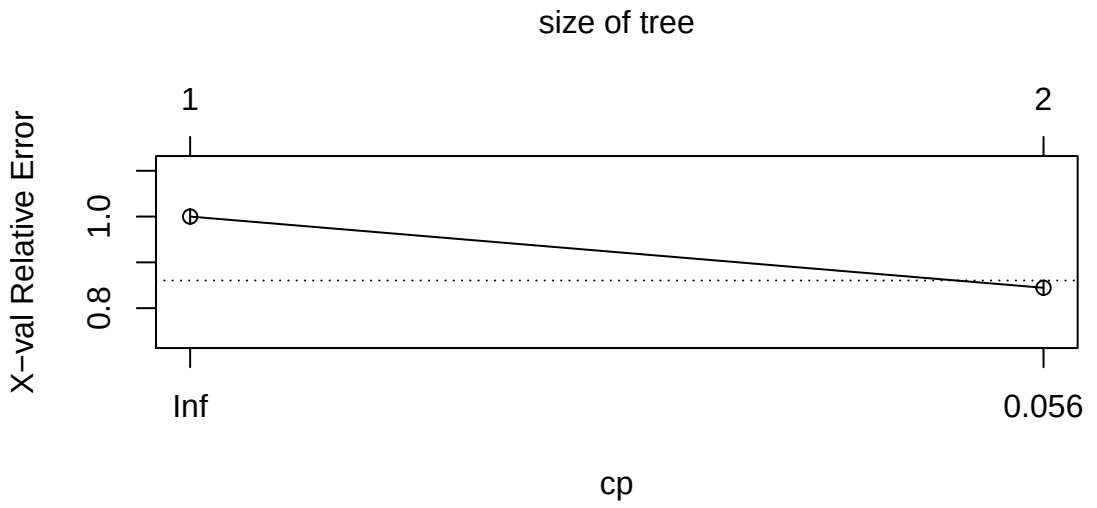
The Gini index or cross-entropy impurity measures are often used to construct the tree where the misclassification error is used for tree pruning.

B. Example: NMES

Lets repeat the CART analysis with a binary outcome, big expenditures, defined to be those above $1,000 per year

```
set.seed(123454321)
dat1.train=dat1[train<-sample(1:nrow(dat),floor(nrow(dat)/2)),]
dat1.test=dat1[-train,]
tree0=rpart(big~.,data=dat1.train,method="class",control=rpart.control(minsize=20,cp=.001))
par(mfrow=c(2,1),mar=c(5,5,5,1))
plotcp(tree0)
tree=rpart(big~.,data=dat1.train,method="class",control=rpart.control(minsize=20,cp=.02))
plotcp(tree)
```





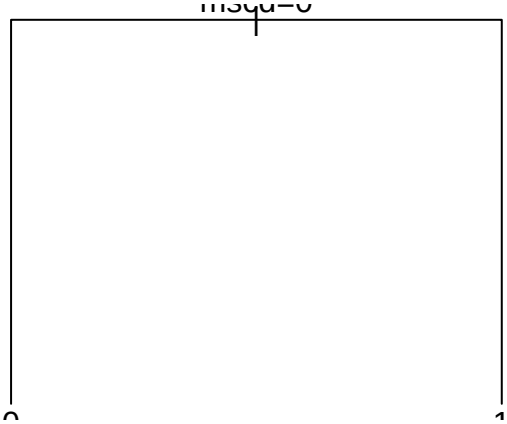

```
## Print the cptable
tree$cptable
```

```
##          CP nsplit rel error    xerror       xstd
## 1 0.1555751      0 1.0000000 1.0000000 0.01667776
## 2 0.0200000      1 0.8444249 0.8444249 0.01600726
```

```
## Print the variable importance
tree$variable.importance
```

```
##     mscd
## 231.0501
```

```
plot(tree)
text(tree,pretty=3)
```

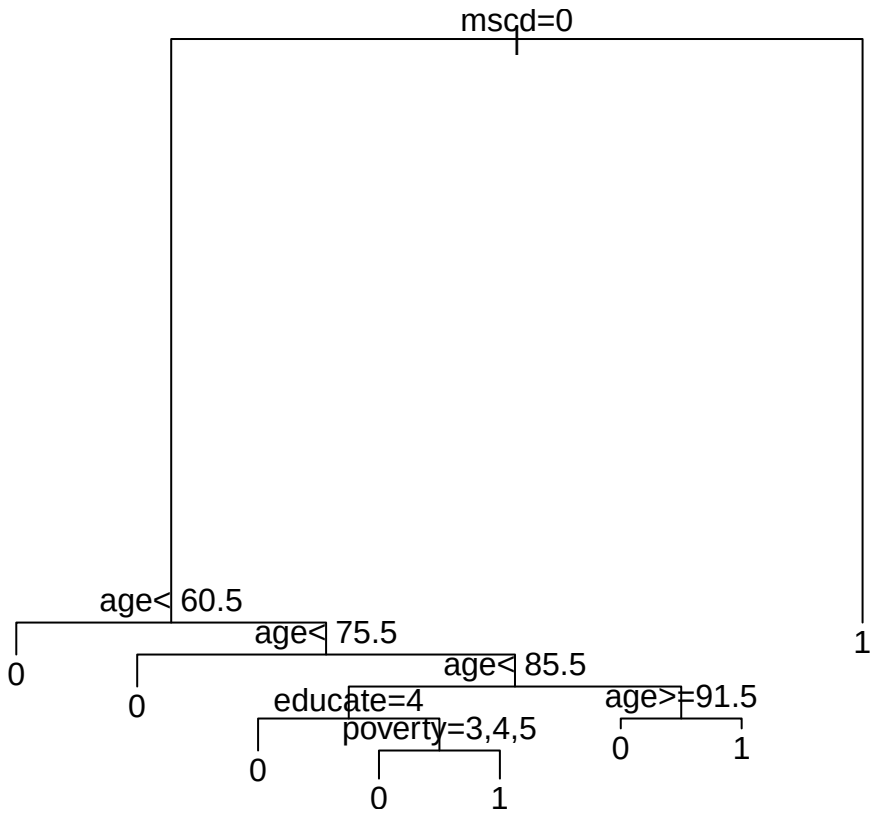


Modify the CART criteria to allow the tree to have fewer cases in the terminal nodes, e.g. minsize = 5.

```
treetest0=rpart(big~.,data=dat1.train,method="class",control=rpart.control(minsize=5,cp=.001))
printcp(treetest0)
```

```
##
## Classification tree:
## rpart(formula = big ~ ., data = dat1.train, method = "class",
##     control = rpart.control(minsize = 5, cp = 0.001))
##
## Variables actually used in tree construction:
## [1] age     beltuse educate male    mscd    poverty
##
## Root node error: 2269/6151 = 0.36888
##
## n= 6151
##
##          CP nsplit rel error  xerror     xstd
## 1 0.1555751      0   1.00000 1.00000 0.016678
## 2 0.0024974      1   0.84442 0.84442 0.016007
## 3 0.0017629      7   0.82680 0.84310 0.016000
## 4 0.0014691     11   0.81974 0.85544 0.016064
## 5 0.0013222     16   0.81225 0.85721 0.016073
## 6 0.0011753     18   0.80961 0.86029 0.016088
## 7 0.0010000     21   0.80608 0.85985 0.016086
```

```
bestcp <- treetest0$cptable[which.min(treetest0$cptable[,"xerror"]),"CP"]
tree.pruned <- prune(treetest0, cp = bestcp)
plot(tree.pruned);text(tree.pruned,pretty=3)
```




```
# comparison to logistic model
model=glm(data=dat1.train,big~ns(age,2)*male*mscd
          + as.factor(poverty)*as.factor(educate),
          family=binomial())

# Get the predicted values from the logistic regression model
model.yhat=predict.glm(model,newdata=dat1.test,type="response")
# The ROCR package requires that you first create a "prediction"
# object using the prediction function,
# this function takes the predicted probabilities + true values
pred.model.test=prediction(model.yhat,dat1.test$big)
# The performance function will compute several measures of
# performance for the classification scheme
# here we are selecting true positive rate, false positive rate
perf.model.test=performance(pred.model.test,"tpr","fpr")
# Use the performance object and ask for AUC
auc.model.test=performance(pred.model.test,"auc")
```

```
# Compare the performance of the parametric model
# to the classification tree
tree.yhat=as.vector(predict(tree,newdata=dat1.test,na.action=na.pass)[,2])
# Same as before, create the prediction object and
# compute auc
pred.tree.test=prediction(tree.yhat,dat1.test$big)
auc.tree.test=performance(pred.tree.test,"auc")

# Compare the performance of the parametric model
# to the classification tree
tree.yhat2=as.vector(predict(tree.pruned,newdata=dat1.test,na.action=na.pass)[,2])
# Same as before, create the prediction object and
# compute auc
pred.tree.test2=prediction(tree.yhat2,dat1.test$big)
auc.tree.test2=performance(pred.tree.test2,"auc")


auc.model.test@y.values[[1]]
```

```
## [1] 0.6762746
```

```
auc.tree.test@y.values[[1]]
```

```
## [1] 0.588122
```

```
auc.tree.test2@y.values[[1]]
```

```
## [1] 0.6463928
```

C. Additional example

Data is available from 1046 passengers from the Titanic. The binary response is whether or not the passenger survived. Possible predictors include: pclass (passenger class: 1st 2nd 3rd), sex (male vs. female), age in years (min 0.17, max 80), sibsp (number of siblings or spouses aboard: 0, . . . , 8), parch (number of parents or children aboard: 0, . . . , 6)

```
data(ptitanic)
set.seed(123)
tree <- rpart(survived ~ ., data = ptitanic, control = rpart.control(cp = 0.0001))
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = survived ~ ., data = ptitanic, control = rpart.control(cp = 1e-04))
##
## Variables actually used in tree construction:
## [1] age    parch  pclass sex    sibsp
##
## Root node error: 500/1309 = 0.38197
##
## n= 1309
##
##          CP nsplit rel error xerror     xstd
## 1 0.4240000      0     1.000  1.000 0.035158
## 2 0.0210000      1     0.576  0.576 0.029976
```

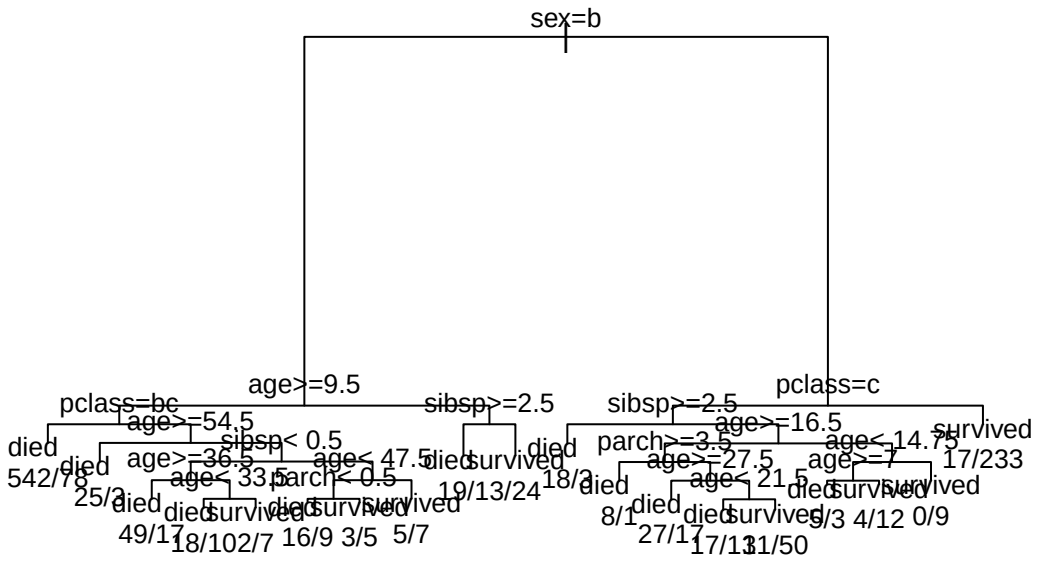
```
## 3 0.0150000      3     0.534  0.540 0.029279
## 4 0.0113333      5     0.504  0.520 0.028869
## 5 0.0025714      9     0.458  0.522 0.028911
## 6 0.0020000     16     0.440  0.520 0.028869
## 7 0.0001000     18     0.436  0.516 0.028785
```

```
bestcp <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]
tree.pruned <- prune(tree, cp = bestcp)

# confusion matrix (training data)
conf.matrix <- table(ptitanic$survived, predict(tree.pruned,type="class"))
rownames(conf.matrix) <- paste("Actual", rownames(conf.matrix), sep = ":")
colnames(conf.matrix) <- paste("Pred", colnames(conf.matrix), sep = ":")
print(conf.matrix)
```

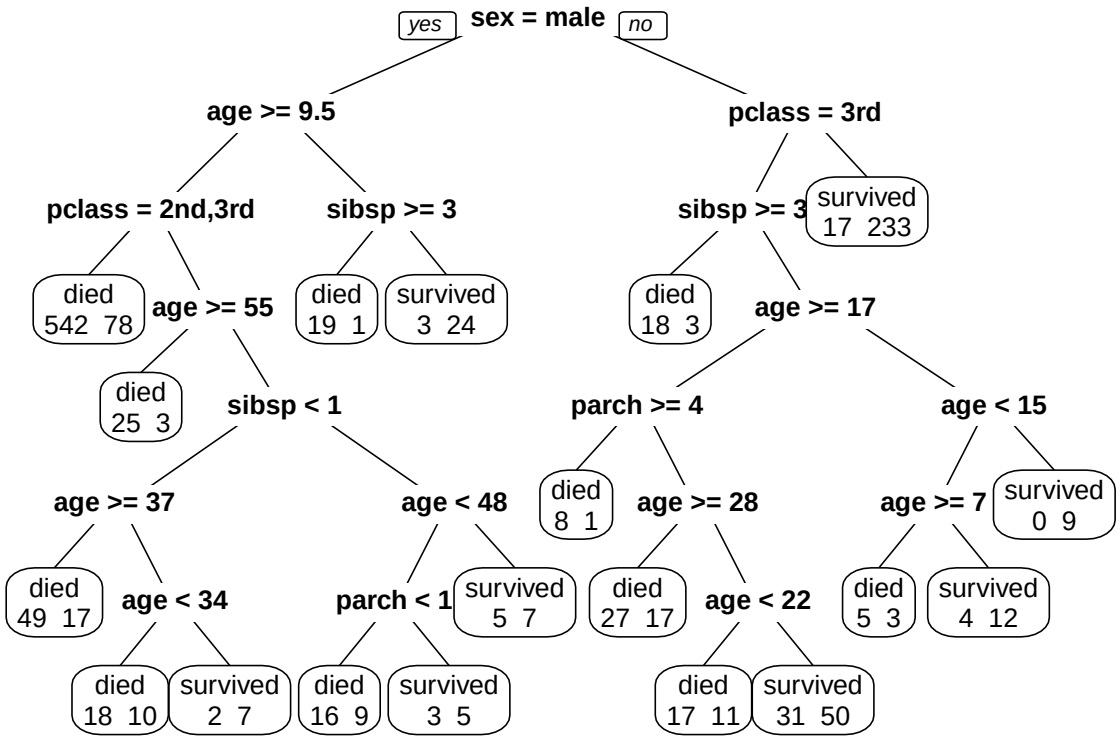
```
##
##                    Pred:died Pred:survived
##   Actual:died            744            65
##   Actual:survived        153           347
```

```
plot(tree.pruned)
text(tree.pruned, cex = 0.8, use.n = TRUE, xpd = TRUE)
```




```
# use.n = TRUE adds number of observations at each node
# xpd = TRUE keeps the labels from exteding outside the plot

prp(tree.pruned, faclen = 0, cex = 0.8, extra = 1)
```




```
# faclen = 0 means to use full names of the factor labels
# extra = 1 adds number of observations at each node; equivalent to using use.n = TRUE in plot.rpart
```