



JOHNS HOPKINS
BLOOMBERG SCHOOL
of PUBLIC HEALTH

Lecture 6

Prediction/classification using logistic regression models Classification And Regression Trees (CART)

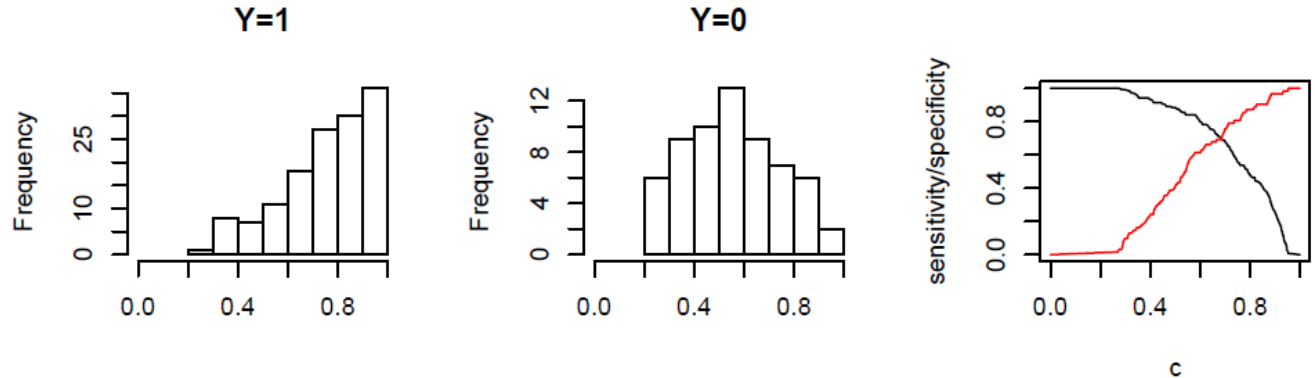
Lecture 5 Review: Prediction/classification

- Data: $(Y_1, X_1), \dots, (Y_n, X_n)$ where X_i is a $(p + 1) \times 1$ vector of exposures/predictors.
 - Model: $\text{logit}[Pr(Y_i = 1|X_i)] = X_i^T \beta$
 - Fit the Model: $\hat{\beta} \rightarrow \hat{\mu}_i = \frac{\exp(X_i^T \hat{\beta})}{1 + \exp(X_i^T \hat{\beta})}$
-
- ▶ Define a classification rule:
-
-
-
-
-
-
-
-
-
-
- ▶ Define sensitivity and specificity based on the classification rule:



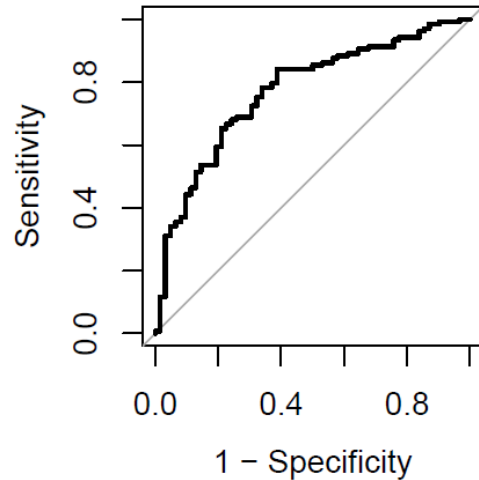
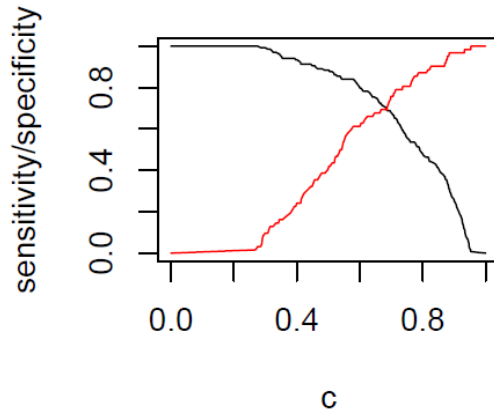
Defining and evaluating the classifier

- ▶ Set c so we can maximize both sensitivity and specificity
- ▶ Plot sens and spec as a function of c



Defining and evaluating the classifier

- ▶ Evaluate the entire model upon which the classifier is based
 - ▶ Receiver Operating Characteristic (ROC) curve
 - ▶ Plot sens vs. 1-spec for each c



Classification and Regression Trees (CART)

- ▶ So far, we have considered using a logistic regression model to define a classifier.
 - ▶ This approach requires that we build the regression model, i.e. we know the key predictors, including functional form for continuous variables and important interactions, etc.
- ▶ Instead of building a logistic regression model for developing a classifier, we will consider a classification and regression tree.
 - ▶ Removes the need for us to specify the model.
- ▶ We will give a very brief introduction to statistical learning methods
 - ▶ Classification and regression trees (CART) (today)
 - ▶ Random forests (Thursday)
 - ▶ 3rd term Machine Learning course by Vadim
 - ▶ Chapter 9 in HTF text
- ▶ Classification and regression trees (CART)
 - ▶ Leo Breiman, Professor of Statistics at UC Berkeley, Richard Olshen, Jerry Friedman and others in 1984
 - ▶ Breiman spent the rest of his career developing important statistical learning (SL) (aka “machine learning - ML”) tools including “bagging”, “boosting” and “random forests”.

CART

- ▶ Data will be partitioned into a
 - ▶ Training set: used to construct the model
 - ▶ Test/validation set: used to evaluate the quality/fit of the model (compared to competing models)
- ▶ Goal is to generate predictions for linear/continuous or categorical (binary or many than 2 categories) responses
- ▶ Algorithm goal: minimize a measure of prediction error
- ▶ Break the predictor space into M non-overlapping rectangular predictor sub-spaces
 - ▶ assign a predicted value within each subspace
 - ▶ Linear outcome: predicted value in each subspace is the sample mean
 - ▶ Binary outcome: predicted value in each subspace is most frequently occurring outcome (0 vs. 1) in that subspace



Regression Trees

- ▶ Approximates $E(Y|X)$ via a step function!

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

- ▶ R_m are selected to minimize

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

- ▶ c_m are estimated via the mean Y in R_m

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



Regression Trees

- ▶ CART uses a greedy algorithm

- ▶ To find the first split:

- ▶ Search over all predictors X_j and split points s

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

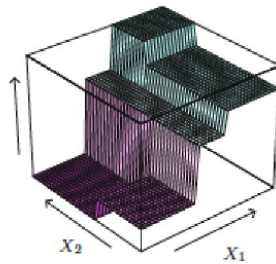
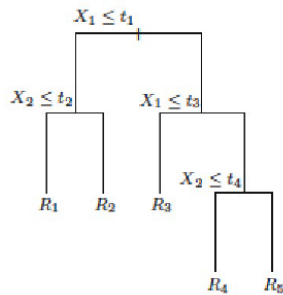
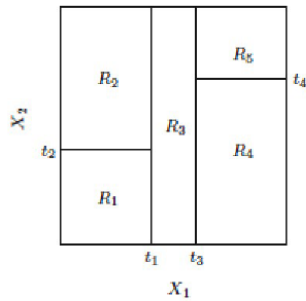
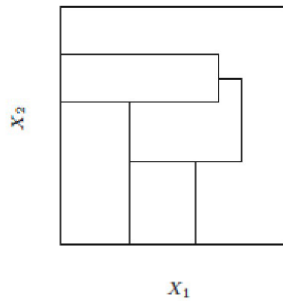
- ▶ Find j and s that satisfy:

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

- ▶ Repeat



HTF Figure 9.2



Regression Trees: When do we stop?

- ▶ Tree size, i.e. number of groups M , is determined by user-specified parameters
- ▶ Recommendation is to start by growing a large tree, call this T_0 , to determine M by setting a minimum node size (e.g. each node must include at least 5 observations).
 - ▶ Then “prune” the tree using a “cost complexity parameter”



Regression Trees: Cost complexity parameter

- ▶ Called “cp” in the output
- ▶ Define a subtree T which is contained in T_0
 - ▶ Obtained by collapsing internal nodes
 - ▶ $|T|$ denotes the number of terminal nodes in subtree T

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

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

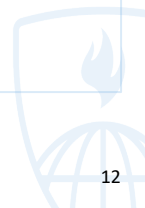
$$Q_m(T) = \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$$

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



Regression Trees: Cost complexity parameter

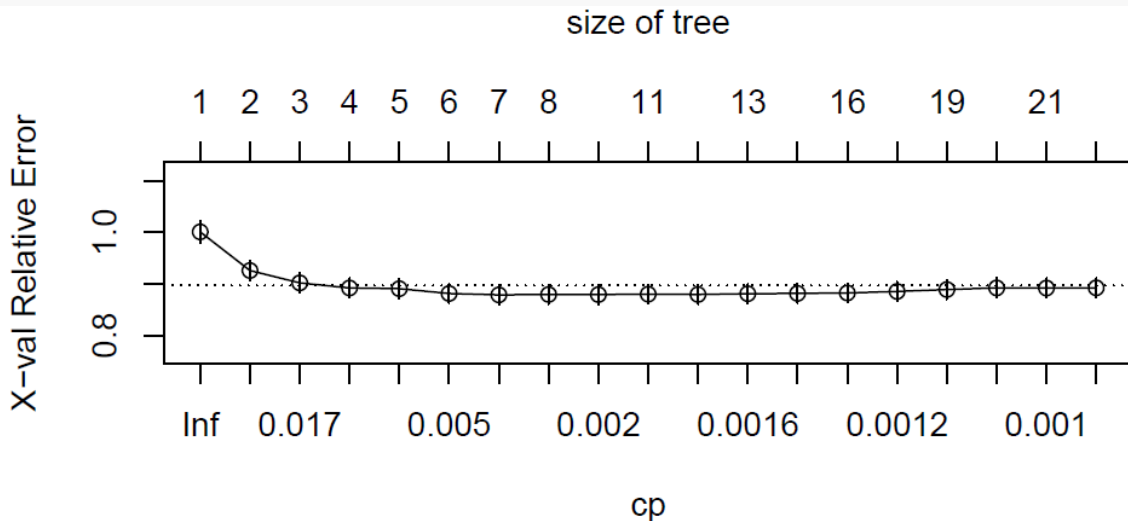
- ▶ CART finds the unique tree T_α that minimizes $C_\alpha(T)$ for each α .
- ▶ α defines the trade-off between tree size and goodness of fit.
- ▶ α values are estimated via a cross-validation procedure; see details in Chapter 9 of HTF
- ▶ Final tree can be selected among the T_α that yields the smallest cross-validated sums of squared error (MSE).



Example: Predict $\log(\text{expenditures} + 1)$ within NMES

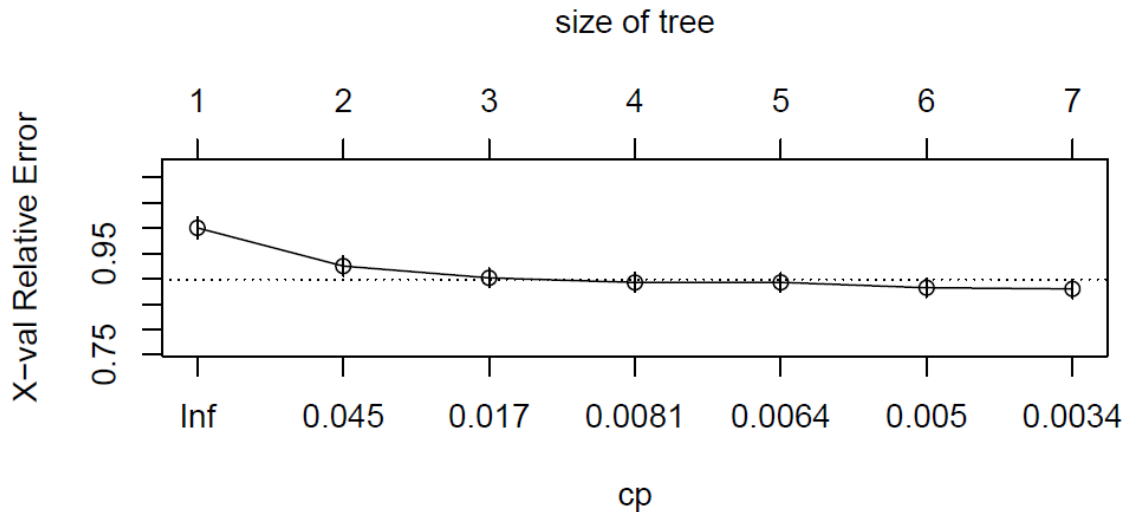
```
set.seed(123454321)
dat.train=dat[train<-sample(1:nrow(dat),floor(nrow(dat)/2)),]
dat.test=dat[-train,]
```

```
## Fit a first tree setting 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)
```



Example: Predict $\log(\text{expenditures} + 1)$ within NMES

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

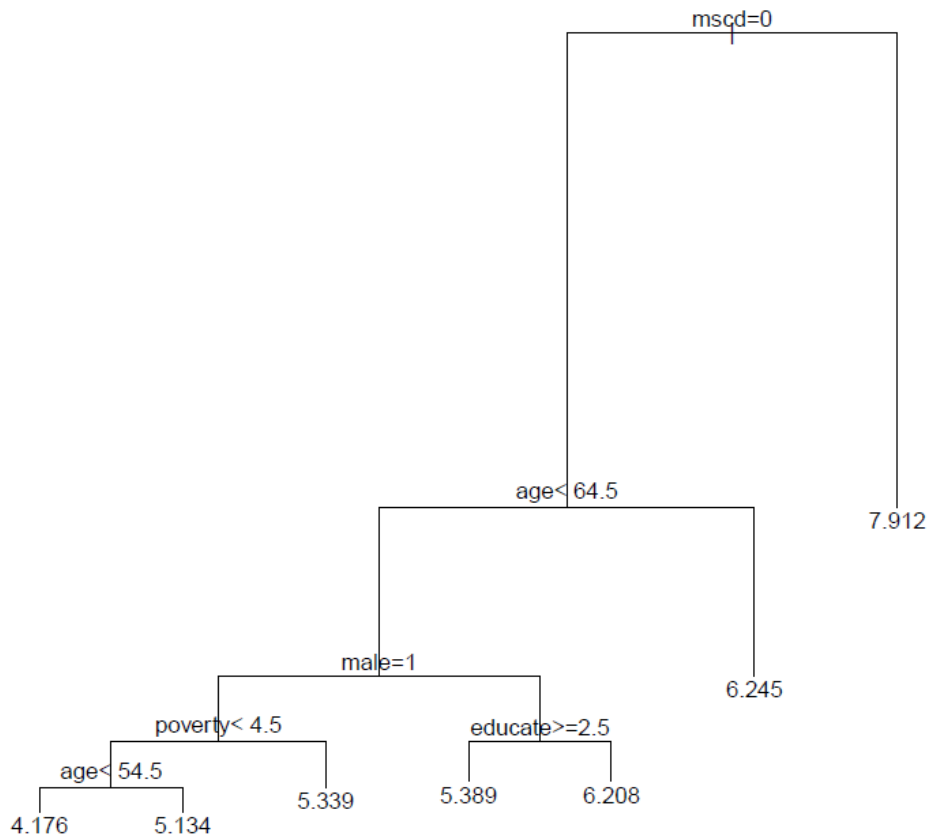


Example: Predict $\log(\text{expenditures} + 1)$ within NMES

```
##          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
##
## Variable importance
##      mscd      age educate      male married poverty
##        53       22       8        7        5        4
```

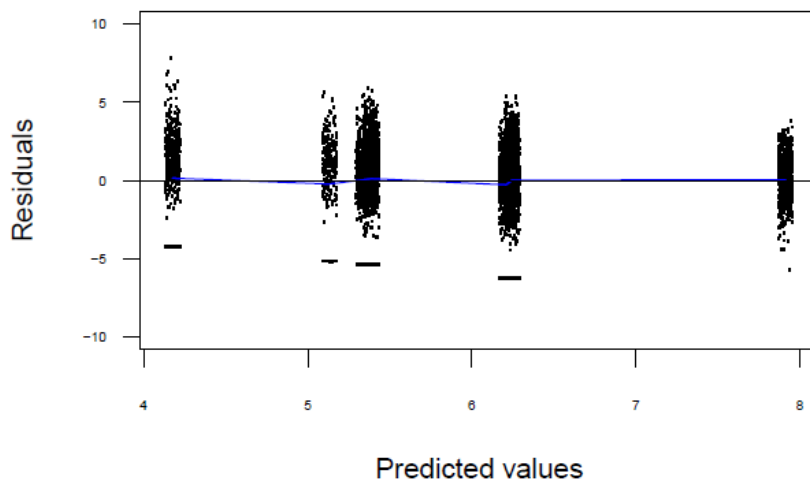
- ▶ Measure of importance: sum of the goodness of split measures for each split where the variable is the primary variable (plus additional piece related to whether this variable is a surrogate), scaled to total 100 across all variables

Example: Predict $\log(\text{expenditure} + 1)$ within NMES



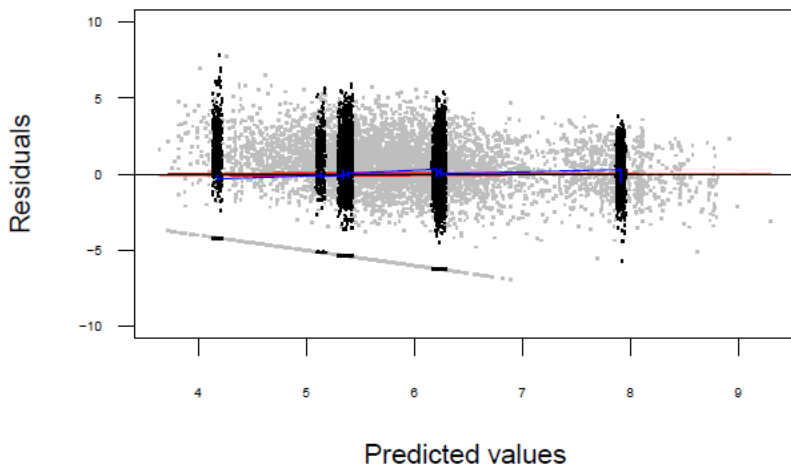
Regression trees: evaluating the fit of the model

```
# 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)
```



Regression trees: comparing to a parametric model

```
# 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)
```



MSE comparison:

CART: 5.806; Linear Model: 5.68

CART: General comments

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



Classification trees

- ▶ Same procedure
 - ▶ Different goodness of fit criteria

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 deviance. When we are constructing a classification tree for a binary response where $p = Pr(Y = 1)$, these measures are:

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

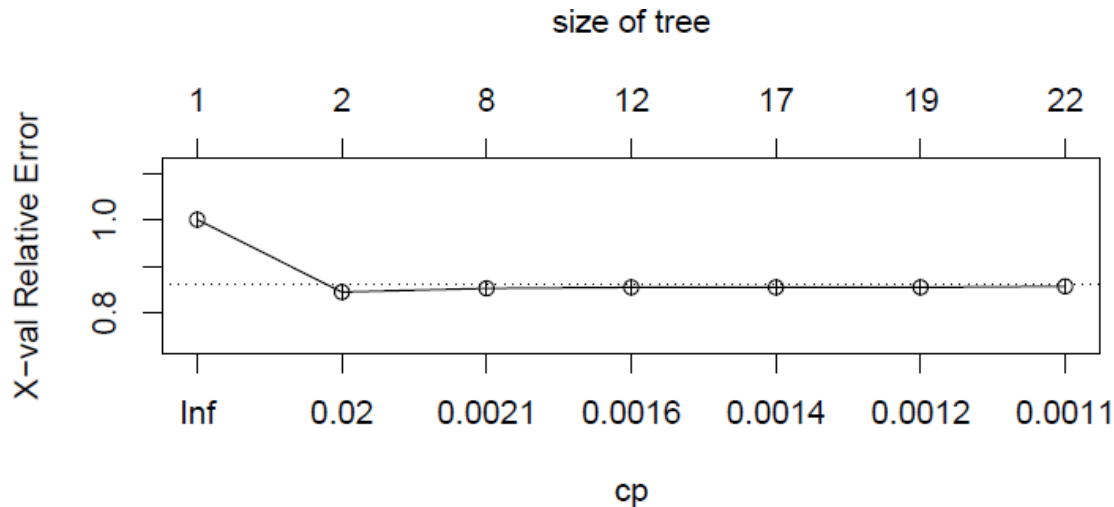
$$\text{Gini index: } 2p(1 - p)$$

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

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

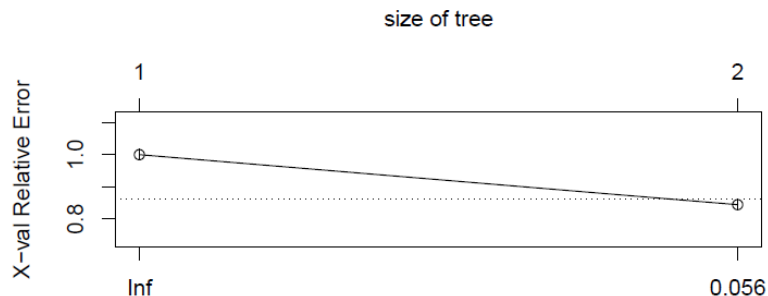
Example: Predict a big expenditure

```
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)
```

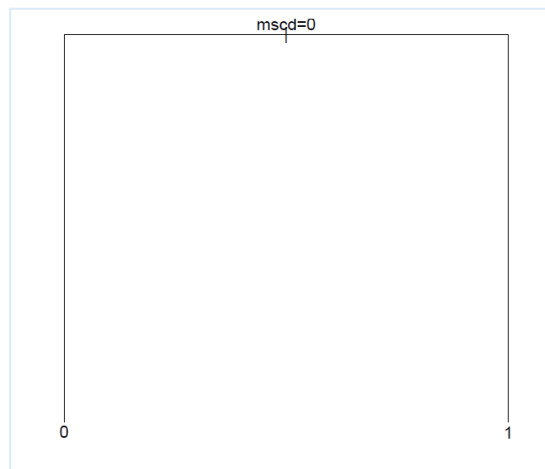


Example: Predict a big expenditure

```
tree=rpart(big~.,data=dat1.train,method="class",control=rpart.control(minsize=20,cp=.02))  
plotcp(tree)
```



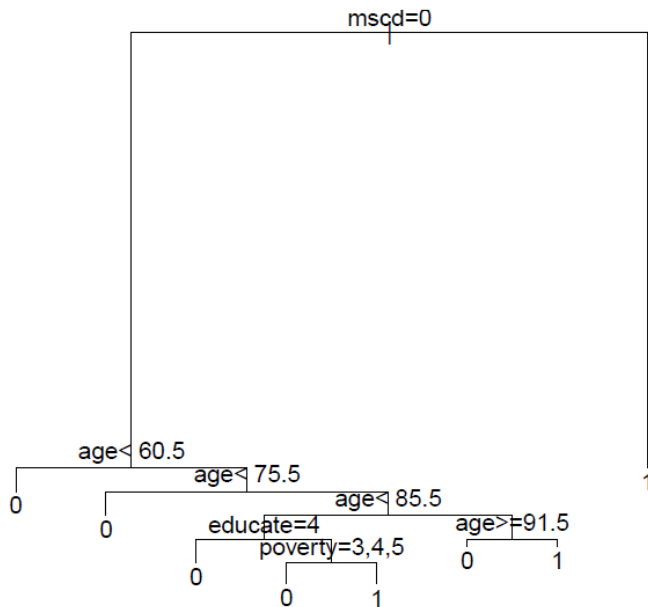
	cp				
##	CP	nsplit	rel error	xerror	xstd
## 1	0.1555751	0	1.0000000	1.0000000	0.01667776
## 2	0.0200000	1	0.8444249	0.8444249	0.01600726
##					
##	Variable importance				
##	mscd				
##	100				



Example: Predict a big expenditure

► Change the CART criteria

```
treetest0=rpart(big~.,data=dat1.train,method="class",control=rpart.control(minsize=5,cp=.001))  
bestcp <- treetest0$cptable[which.min(treetest0$cptable[, "xerror"]), "CP"]  
tree.pruned <- prune(treetest0, cp = bestcp)  
plot(tree.pruned);text(tree.pruned,pretty=3)
```



Example: Predict a big expenditure

- Compare to a parametric model

```
# 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")
```


Example: Predict a big expenditure

```
# 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")
```

Example: Predict a big expenditure

- ▶ Compare the AUC values

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

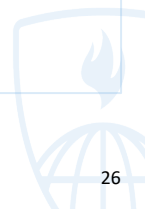
```
## [1] 0.6762746
```

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

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

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

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



Where to next?

- ▶ Based on some of the earlier limitations we pointed out, we will consider random forests!

