

CART - continued

In this lecture, we will go over:

Pruning a tree, Prediction using a tree

Classification trees

In the last lecture, Classification and Regression Tree methodology was introduced. Decision trees have following features:

- Nonparametric, do not require normality
- Can be used for various data types – continuous, categorical (ordinal/nominal/binary)
- Can identify important variables, interactions, outliers.

Three main steps are involved:

Splitting

- select a variable and a split using Gini index(classification) or SS (regression)
- Split data into 2 subsets; process repeated on the two subsets (reason why this is called the *greedy* algorithm)
- Continue until a large tree with nodes containing a small number of similar observations are obtained

Pruning

Uses a cost complexity measure (CP) to trim the tree

Tree Selection

Use X-validation / test data to select a tree. Tree with small X-validation error are preferred, and so are smaller tree with comparable accuracy.

- Pruning is used to avoid the problem of overfitting.
- In backward pruning, a tree is grown until all possible leaf nodes have been reached, and then the tree is pruned.
- Post-pruning has been shown to increase accuracy up to 25%.

Pruning is done using a cost complexity measure:

$$R_{CP} = R + CP \times T$$

where

R = tree risk (SS_residuals for regression,
% mis-classification for classification)

T = # (terminal nodes)

CP = complexity parameter

Tree Selection is done using 10-fold X-validation:

Split data into 10 random subsets – D_j , $j = 1, 2, \dots, 10$

Leave out D_1 , compute tree on D_j , $j = 2, \dots, 10$, evaluate using D_1

Repeat above by leaving out D_2, D_3, \dots, D_{10}

Add error across 10 subsets to get X-validation error rate.

R-library rpart

- rpart – fits the tree model
- rpart.control – used to specify parameters
- summary.rpart- outputs summary of fitted model
- snip.rpart – interactive pruning
- prune.rpart – pruning
- plot.rpart and text.rpart – plotting tree

Example 1:

```
# we will use the data cu.summary of library rpart
# to illustrate tree-pruning.
# load library rpart
library(rpart)
car <- cu.summary # data is in this library
```

head(car)

			Price	Country	Reliability	Mileage	Type
Acura	Integra	4	11950	Japan	Muchbetter	NA	Small
Dodge	Colt	4	6851	Japan	<NA>	NA	Small
Dodge	Omni	4	6995	USA	Muchworse	NA	Small
Eagle	Summit	4	8895	USA	better	33	Small
Ford	Escort	4	7402	USA	worse	33	Small
Ford	Festiva	4	6319	Korea	better	37	Small

```
# grow(fit) regression tree
tree <- rpart(Mileage~Price + Country +
Reliability + Type, method="anova",
data=cu.summary)

print(tree)
```

Regression tree:

```
rpart(formula = Mileage ~ Price + Country + Reliability + Type,
data = cu.summary, method = "anova")
```

Variables actually used in tree construction:

```
[1] Price Type
```

Root node error: $1354.6/60 = 22.576$

n=60 (57 observations deleted due to missingness)

	CP	nsplit	rel error	xerror	xstd
1	0.622885	0	1.00000	1.03846	0.178102
2	0.132061	1	0.37711	0.59928	0.117844
3	0.025441	2	0.24505	0.41922	0.091326
4	0.011604	3	0.21961	0.39572	0.082687
5	0.010000	4	0.20801	0.42330	0.086660

Explanation of output:

- X-validation splits data into k subsets (rpart default k = 10)
- Each of k subsets is left out 1-by-1, model fitted to remaining data and used to predict the 'left-out' data.
- At the k-th *fold*, subset k = test data, remaining = training (*test data is also referred to as OUT-OF-BAG data*)

Data analysis and graphics using R: an example-based approach

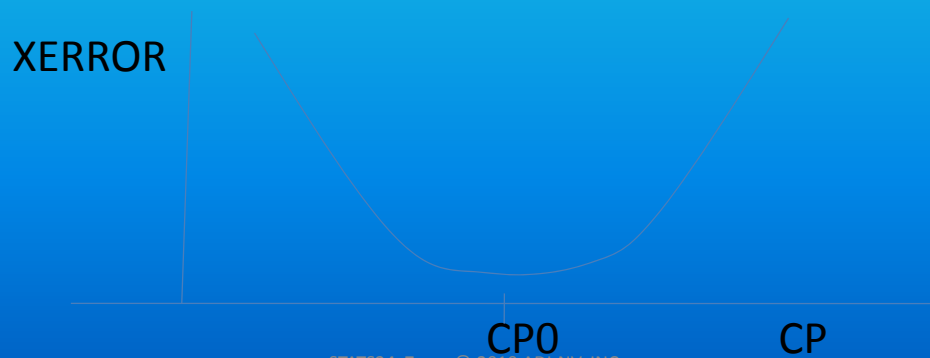
By John Hilary Maindonald, John Braun

- In regression modeling,
prediction error = $SS_{\text{Residuals}}$
- In classification schemes, we look at %(mis-classification) or % (correct classification).

average error = $\text{total error} / \#(\text{observations})$

The Cost-Complexity parameter (CP)

- The # of splits is controlled in by CP that puts a cost on each splits. Splitting is stopped when increase in cost is not worth the reduction in lack-of-fit.
- High CP leads to a small tree; low CP results in a complex tree.



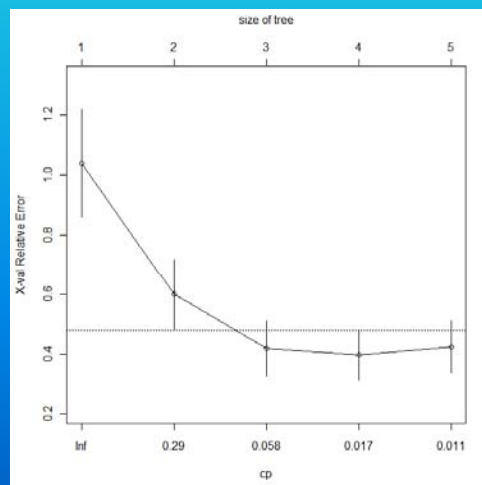
STATS24x7.com © 2010 ADI-NV, INC

15

From CP-table on slide 5, $CP0 \approx .012$

```
# plot x-validation results  
plotcp(tree)
```

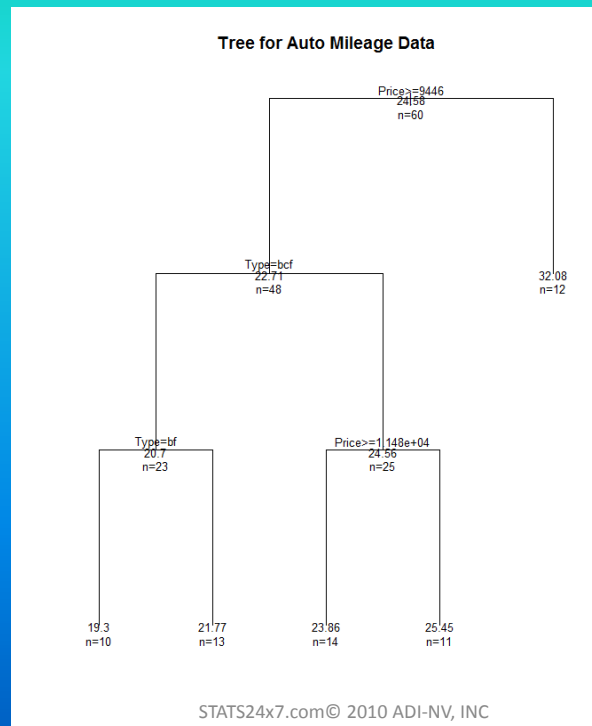
```
# summary(tree) will give detailed information.
```



STATS24x7.com © 2010 ADI-NV, INC

16

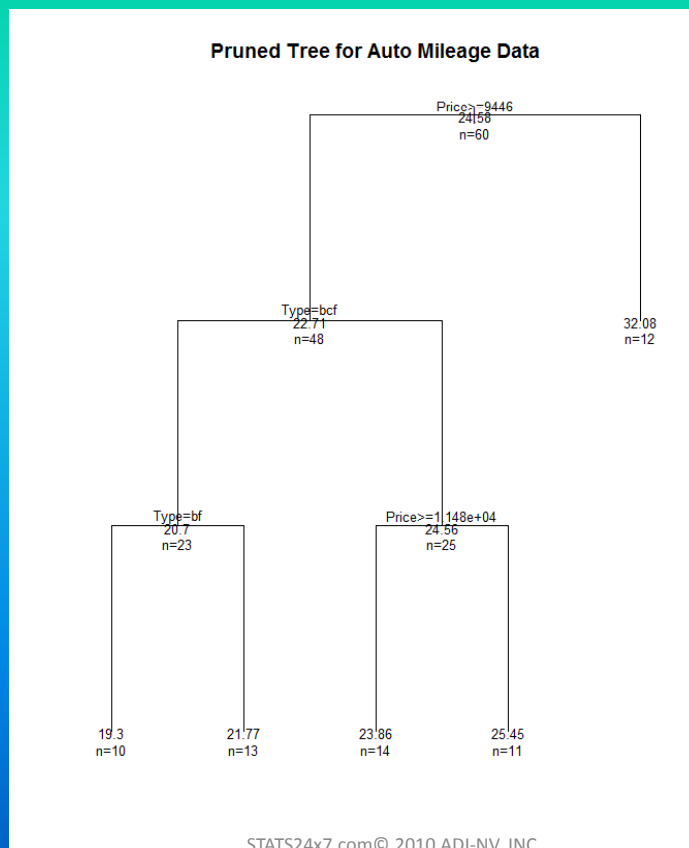
```
# plot the tree
plot(tree, uniform=TRUE, main = "Tree for Auto Mileage Data")
text(tree, use.n=TRUE, all=TRUE, cex=.8)
```



17

```
# prune the tree using CP = CP0 = 0.01160389)
Pruned_tree <- prune(tree, cp = 0.01160389)
plot(Pruned_tree, uniform=TRUE, main = "Pruned Tree
for Auto Mileage Data")
text(Pruned_tree, use.n=TRUE, all=TRUE, cex=.8)
```

Above produces the pruned tree on the next slide
(turns out to be the same tree as the original tree on
slide 10)

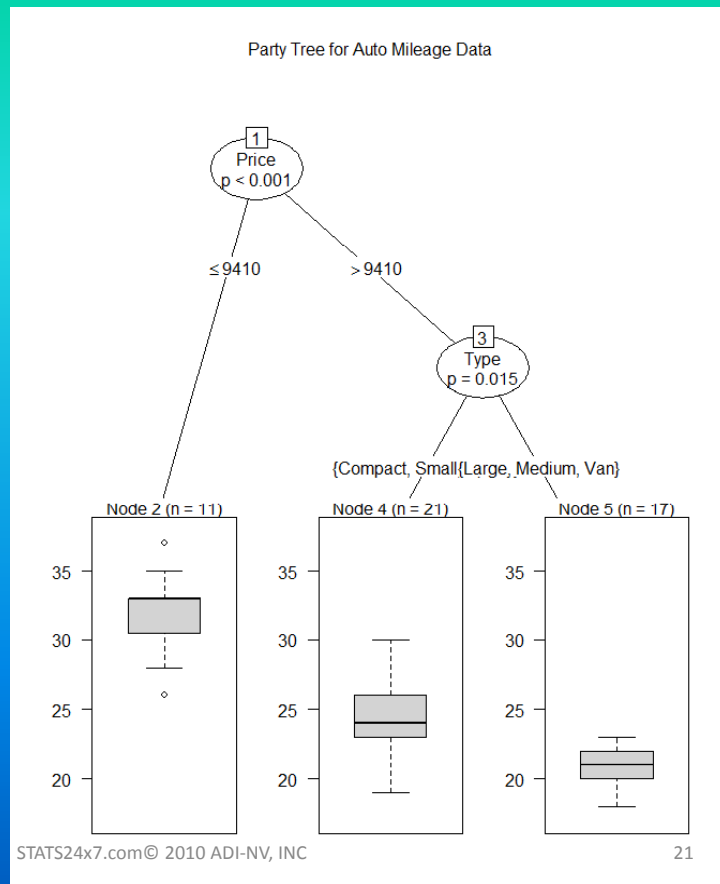


19

the library party can also be used to get a
classification tree

```
library(party)
tree2 <- ctree(Mileage~Price + Country + Reliability +
Type, data=na.omit(cu.summary))
plot(tree2, main = "Party Tree for Auto Mileage Data")
```

Regression tree for the auto mileage data from package rparty



Example 2: Boston housing data

```
library(MASS)  
library(rpart)
```

```
boston <-  
read.csv("K:/TEACH/DataMining_Fall2009/Data/bostonhousing.csv",  
header=TRUE)
```

```
boston.tree1 <- rpart(boston$MEDV ~ ., method="anova",  
data=boston, control=rpart.control(cp=.0001))
```

```
summary(boston.tree1) # part of output shown on next 2 slides
```

	CP	nsplit	relerror	xerror	xstd
1	0.452744	0	1	1.00376	0.082951
2	0.171172	1	0.54726	0.62267	0.056205
3	0.071658	2	0.37608	0.43498	0.046558
4	0.036164	3	0.30443	0.34667	0.041835
5	0.033369	4	0.26826	0.34455	0.042497
6	0.026613	5	0.23489	0.32982	0.042491
7	0.015851	6	0.20828	0.2987	0.040039
8	0.008245	7	0.19243	0.27099	0.035942
9	0.007265	8	0.18418	0.2448	0.031118
10	0.006931	9	0.17692	0.2464	0.031142
11	0.006126	10	0.16999	0.24681	0.031149
12	0.004805	11	0.16386	0.23788	0.030804
13	0.004561	12	0.15905	0.23904	0.029659
14	0.003941	13	0.15449	0.24033	0.02974
15	0.003316	14	0.15055	0.23825	0.029921
16	0.003121	15	0.14724	0.23335	0.02992
17	0.002246	16	0.14412	0.22916	0.029966
18	0.002235	18	0.13962	0.2317	0.029981
19	0.002172	19	0.13739	0.23241	0.029986

	CP	nsplit	relerror	xerror	xstd
20	0.001934	20	0.13522	0.2325	0.029975
21	0.001717	21	0.13328	0.23127	0.029911
22	0.001444	22	0.13157	0.22863	0.028511
23	0.00141	23	0.13012	0.22964	0.028523
24	0.001364	24	0.12871	0.23008	0.028522
25	0.001278	25	0.12735	0.22908	0.028587
26	0.001247	26	0.12607	0.22882	0.028591
27	0.001137	28	0.12358	0.22833	0.028607
28	0.000963	29	0.12244	0.229	0.02866
29	0.000849	30	0.12148	0.22741	0.028659
30	0.00071	31	0.12063	0.22612	0.028719
31	0.000588	32	0.11992	0.22623	0.028709
32	0.000512	33	0.11933	0.22477	0.028705
33	0.000379	34	0.11882	0.22433	0.028709
34	0.000372	35	0.11844	0.22398	0.028709
35	0.000344	36	0.11807	0.22389	0.028713
36	0.000331	37	0.11772	0.22386	0.028713
37	0.000227	39	0.11706	0.22429	0.02871
38	0.000196	40	0.11683	0.22458	0.028778

The complexity table above shows that:

Tree with 27 nodes gives min xerror

Trees with 9, 10, 11, or 12 nodes are within 1 sd of minimum ($.2283 + .0286 = .2569$).

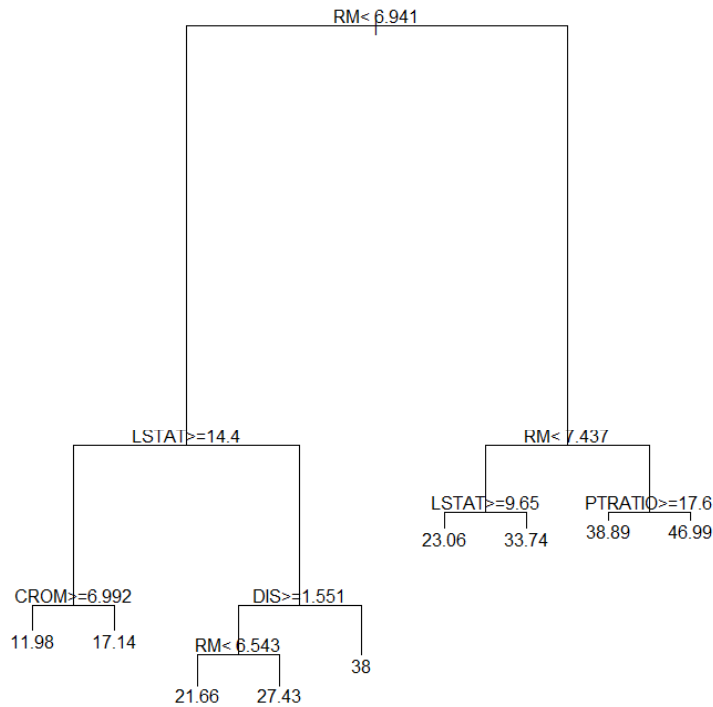
Using 1 sd rule, we will use CP inside (.007, .008).

```
pruned.boston 1<- prune(boston.tree1, cp = .0075)
```

```
plot(pruned.boston1, main = "Pruned (CP = .0075) Boston Housing  
Data Tree")
```

```
text(pruned.boston1)
```

Pruned (CP = .0075) Boston Housing Data Tree



STATS24x7.com© 2010 ADI-NV, INC

27

From the above tree, we see that:

Important split is by RM (# rooms/dwelling)

2nd split on LSTAT (% lower status of population) is also quite important.

Houses with high RM have more average value .

Houses with small RM and low status in population with high crime rate are cheaper.

STATS24x7.com© 2010 ADI-NV, INC

28

Prediction Using the Fitted Tree

```
# min xerror rule
pruned.boston2 <- prune(boston.tree1, cp = .0011)

# predict using both models
boston.predict1 <- predict(pruned.boston1)
boston.predict2 <- predict(pruned.boston2)

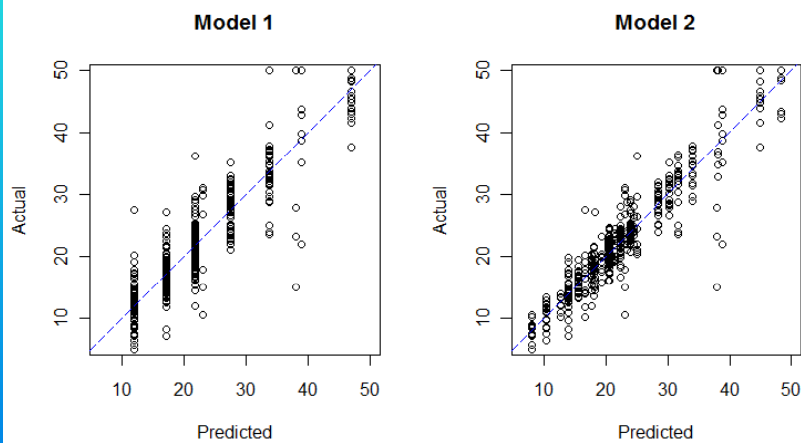
boston.pred.mat <- cbind(boston$MEDV,
boston.predict1, boston.predict2)
boston.pred.mat <- data.frame(boston.pred.mat)
names(boston.pred.mat) <- c("MEDV", "pred1", "pred2")
cor(boston.pred.mat)
```

	MEDV	pred1	pred2
MEDV	1.0000000	0.9032262	0.9367825
pred1	0.9032262	1.0000000	0.9641792
pred2	0.9367825	0.9641792	1.0000000

Model 2 (min xerror) has a higher correlation with observed MEDV, but has 27 nodes.

```
# plot observed vs predicted by the two models
```

```
par(mfrow=c(1,2), pty="s")  
with(boston.pred.mat,{  
  eqscplot(boston.predict1,MEDV,  
    xlim=range(boston.predict1,boston.predict2),ylab="Actual",  
    xlab="Predicted",main="Model 1")  
  abline(0,1,col="blue",lty=5)  
  eqscplot(boston.predict2,MEDV,  
    xlim=range(boston.predict1,boston.predict2),ylab="Actual",  
    xlab="Predicted",main="Model 2")  
  abline(0,1,col="blue",lty=5)  
  par(mfrow=c(1,1))  
})
```

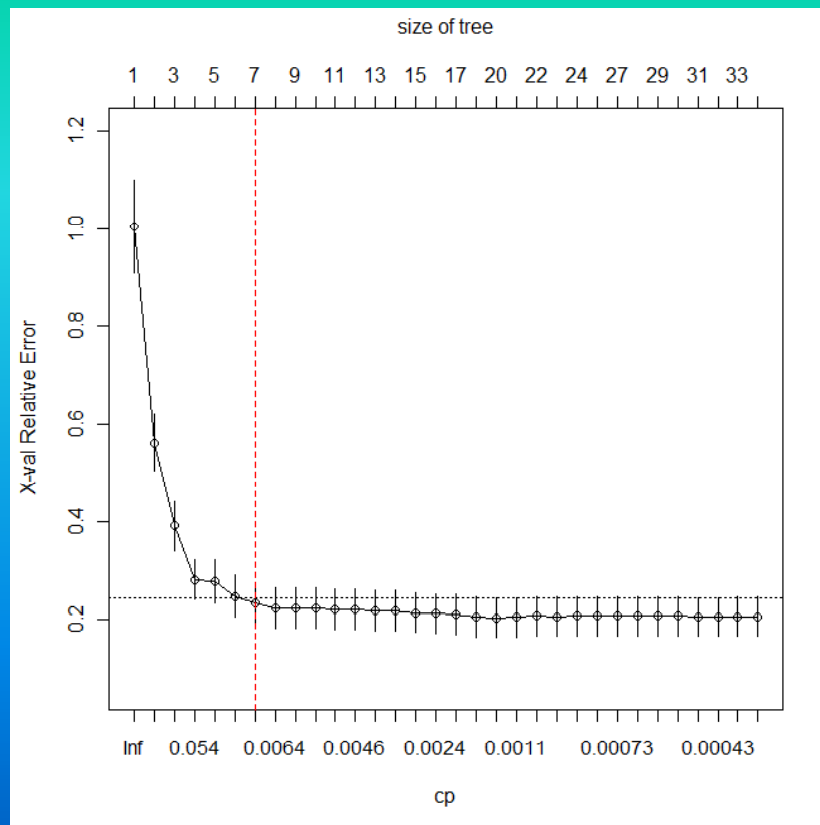


Evaluate Tree Model Using Training/Test Data

```
n <- nrow(boston)
# sample 75% of data
boston.samp <- sample(n, round(n*.75))
btrain <- boston[boston.samp,]
btest <- boston[-boston.samp,]

btrain.rp <- rpart(btrain$MEDV ~ ., method="anova",
data=btrain, control=rpart.control(cp=.0001))
summary(btrain.rp)

plotcp(btrain.rp)
abline(v=7,lty=2,col="red")
```



```
prune.btrain <- prune(btrain.rp, cp=.01)
plot(prune.btrain)
text(prune.btrain)
```

