Classification and Regression Trees

EPSY 887: Data Science Institute

Jason Bryer

https://github.com/jbryer/EPSY887DataScience
jason@bryer.org

October 7, 2014

Agenda

Agenda

Classification and Regression Trees (CART)

The goal of CART methods is to find best predictor in X of some outcome, y. CART methods do this recursively using the following procedures:

1. Find the best predictor in X for y.
2. Split the data into two based upon that predictor.
3. Repeat 1 and 2 with the split datasets until a stopping criteria has been reached.

Classification and Regression Trees (CART)

The goal of CART methods is to find best predictor in X of some outcome, y. CART methods do this recursively using the following procedures:

1. Find the best predictor in X for y.
2. Split the data into two based upon that predictor.
3. Repeat 1 and 2 with the split datasets until a stopping criteria has been reached.

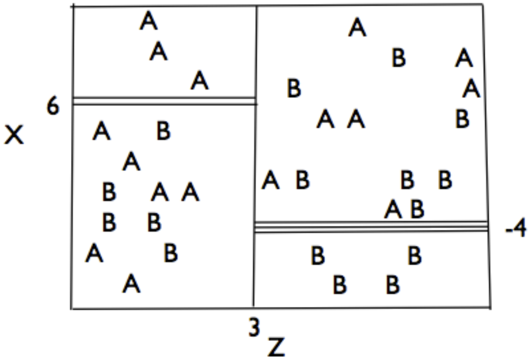
There are a number of possible stopping criteria including:

- Only one data point remains.
- All data points have the same outcome value.
- No predictor can be found that sufficiently splits the data.

Recursive Partitioning Logic of CART

Consider the scatterplot to the right with the following characteristics:

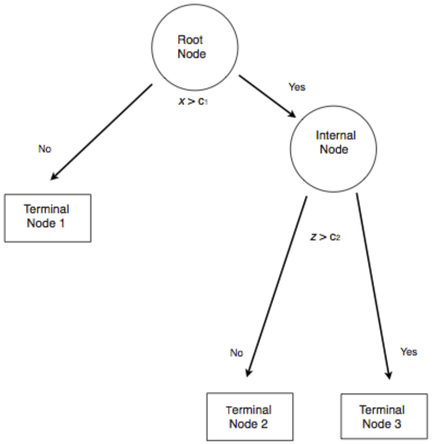
- Binary outcome, G, coded “A” or “B”.
- Two predictors, x and z
- The vertical line at $z = 3$ creates the first partition.
- The double horizontal line at $x = -4$ creates the second partition.
- The triple horizontal line at $x = 6$ creates the third partition.



Recursive Partitioning of a Binary Outcome
(where G = A or B and predictors are Z and X)

Tree Structure

- The root node contains the full dataset.
- The data are split into two mutually exclusive pieces. Cases where $x > c_i$ go to the right, cases where $x <= c_i$ go to the left.
- Those that go to the left reach a terminal node.
- Those on the right are split into two mutually exclusive pieces. Cases where $z > c_2$ go to the right and terminal node 3; cases where $z <= c_2$ go to the left and terminal node 2.



Sum of Squared Errors

The sum of squared errors for a tree *T* is:

$$S = \sum_{c \in leaves(T)} \sum_{i \in c} (y - m_c)^2$$

Where, $m_c = \frac{1}{n} \sum_{i \in c} y_i$, the prediction for leaf *c*.

Sum of Squared Errors

The sum of squared errors for a tree *T* is:

$$S = \sum_{c \in leaves(T)} \sum_{i \in c} (y - m_c)^2$$

Where, $m_c = \frac{1}{n} \sum_{i \in c} y_i$, the prediction for leaf *c*.

Or, alternatively written as:

$$S = \sum_{c \in leaves(T)} n_c V_c$$

Where V_c is the within-leave variance of leaf *c*.

Sum of Squared Errors

The sum of squared errors for a tree T is:

$$S = \sum_{c \in leaves(T)} \sum_{i \in c} (y - m_c)^2$$

Where, $m_c = \frac{1}{n} \sum_{i \in c} y_i$, the prediction for leaf c.

Or, alternatively written as:

$$S = \sum_{c \in leaves(T)} n_c V_c$$

Where V_c is the within-leave variance of leaf c.

Or goal then is to find splits that minimize S.

Advantages of CART Methods

- Making predictions is fast.
- It is easy to understand what variables are important in making predictions.
- Trees can be grown with data containing missingness. For rows where we cannot reach a leaf node, we can still make a prediction by averaging the leaves in the sub-tree we do reach.
- The resulting model will inherently include interaction effects.
- There are many reliable algorithms available.

Agenda

Califonia Real Estate

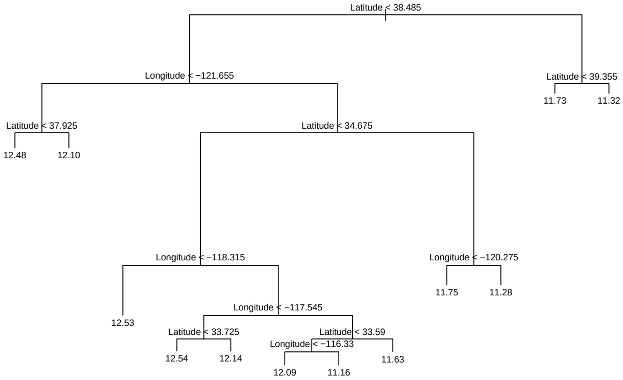
In this example we will predict the median California house price from the house's longitude and latitude.

```
> names(calif)
```

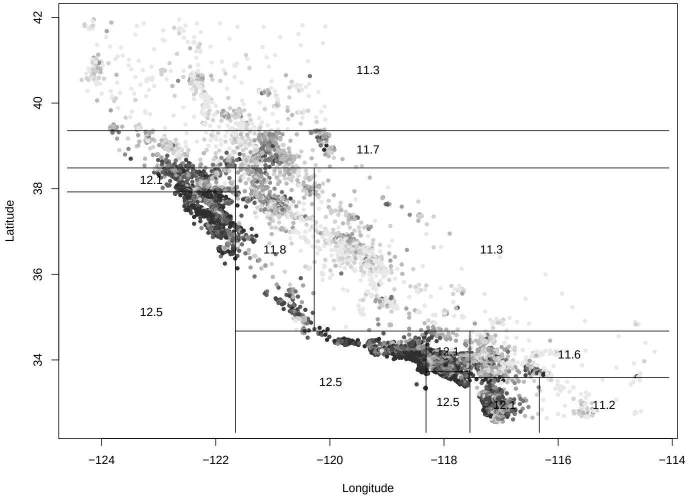
```
[1] "MedianHouseValue" "MedianIncome"
[3] "MedianHouseAge"   "TotalRooms"
[5] "TotalBedrooms"    "Population"
[7] "Households"       "Latitude"
[9] "Longitude"
```

Califonia Real Estate: Tree 1

```
> treefit <- tree(log(MedianHouseValue) ~ Longitude + Latitude,
    data=calif)
> plot(treefit); text(treefit, cex=0.75)
```



Califonia Real Estate: Tree 1



Califonia Real Estate: Tree 1

```
> summary(treefit)

Regression tree:
tree(formula = log(MedianHouseValue) ~ Longitude + Latitude,
    data = calif)
Number of terminal nodes:  12
Residual mean deviance:  0.1662 = 3429 / 20630
Distribution of residuals:
    Min.   1st Qu.    Median      Mean   3rd Qu.
-2.75900 -0.26080 -0.01359  0.00000  0.26310
    Max.
 1.84100
```

Here “deviance” is the mean squared error, or root-mean-square error of 0.41.

Califonia Real Estate: Tree 2, Reduce Minimum Deviance

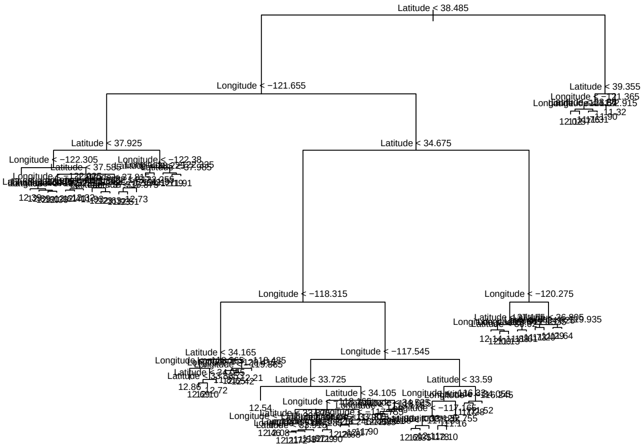
We can increase the fit but changing the stopping criteria with the `mindev` parameter.

```
> treefit2 <- tree(log(MedianHouseValue) ~ Longitude + Latitude,
     data=calif, mindev=.001)
> summary(treefit2)
```

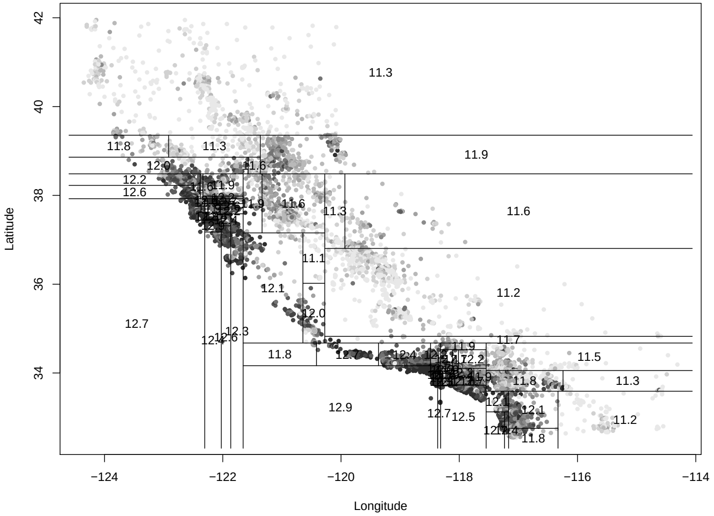
```
Regression tree:
tree(formula = log(MedianHouseValue) ~ Longitude + Latitude,
    data = calif, mindev = 0.001)
Number of terminal nodes:  68
Residual mean deviance:  0.1052 = 2164 / 20570
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.
-2.94700 -0.19790 -0.01872  0.00000  0.19970
    Max.
 1.60600
```

With the larger tree we now have a root-mean-square error of 0.32.

Califonia Real Estate: Tree 2, Reduce Minimum Deviance



Califonia Real Estate: Tree 2, Reduce Minimum Deviance



Califonia Real Estate: Tree 3, Include All Variables

However, we can get a better fitting model by including the other variables.

```
> treefit3 <- tree(log(MedianHouseValue) ~ ., data=calif)
> summary(treefit3)
```

```
Regression tree:
tree(formula = log(MedianHouseValue) ~ ., data = calif)
Variables actually used in tree construction:
[1] "MedianIncome"   "Latitude"
[3] "Longitude"      "MedianHouseAge"
Number of terminal nodes:  15
Residual mean deviance:  0.1321 = 2724 / 20620
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.
-2.86000 -0.22650 -0.01475  0.00000  0.20740
    Max.
 2.03900

>
```

With all the available variables, the root-mean-square error is 0.11.

Agenda

Titanic Example[1]

survived	Survived Titanic sinking
sex	Gender
pclass	Passenger class
age	Age at sailing
sibsp	Number of siblings or spouses aboard.

```
> names(titanic3)
```

```
 [1] "pclass"    "survived"  "name"
 [4] "sex"       "age"       "sibsp"
 [7] "parch"     "ticket"    "fare"
[10] "cabin"     "embarked"  "boat"
[13] "body"      "home.dest"
```

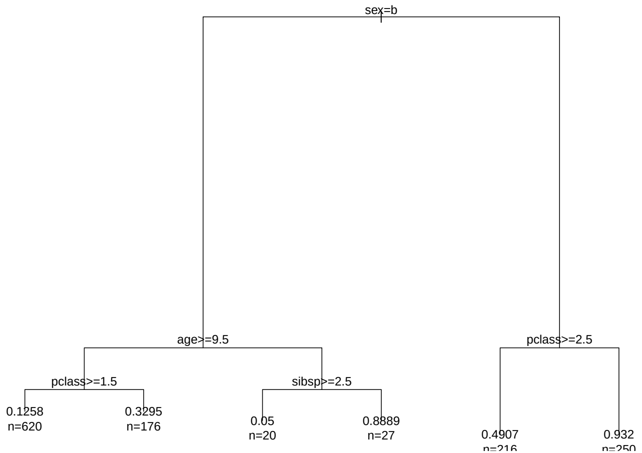
[1]Data available from Vanderbilt University

Classification using rpart

```
> (titanic.rpart <- rpart(survived ~ pclass + sex + age + sibsp,
   data=titanic3))

n= 1309

node), split, n, deviance, yval
      * denotes terminal node

 1) root 1309 309.014500 0.3819710
   2) sex=male 843 130.251500 0.1909846
     4) age>=9.5 796 112.763800 0.1708543
       8) pclass>=1.5 620  68.187100 0.1258065 *
       9) pclass< 1.5 176  38.886360 0.3295455 *
     5) age< 9.5 47  11.702130 0.5319149
      10) sibsp>=2.5 20   0.950000 0.0500000 *
      11) sibsp< 2.5 27   2.666667 0.8888889 *
   3) sex=female 466  92.388410 0.7274678
     6) pclass>=2.5 216  53.981480 0.4907407 *
     7) pclass< 2.5 250  15.844000 0.9320000 *
```

Classification using rpart

```
> plot(titanic.rpart); text(titanic.rpart, use.n=TRUE, cex=1)
```

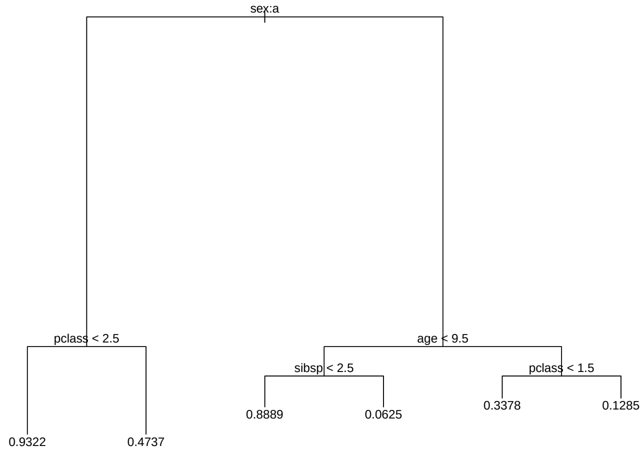


Classification using tree

```
> (titanic.tree <- tree(survived ~ pclass + sex + age + sibsp,
   data=titanic3))

node), split, n, deviance, yval
      * denotes terminal node

 1) root 1046 252.7000 0.4082
   2) sex: female 388  72.2500 0.7526
     4) pclass < 2.5 236  14.9200 0.9322 *
     5) pclass > 2.5 152  37.8900 0.4737 *
   3) sex: male 658 107.3000 0.2052
     6) age < 9.5 43  10.4700 0.5814
      12) sibsp < 2.5 27   2.6670 0.8889 *
      13) sibsp > 2.5 16   0.9375 0.0625 *
     7) age > 9.5 615  90.3300 0.1789
      14) pclass < 1.5 148  33.1100 0.3378 *
      15) pclass > 1.5 467  52.2900 0.1285 *
```

Classification using `tree`

```
> plot(titanic.tree); text(titanic.tree, cex=1)
```

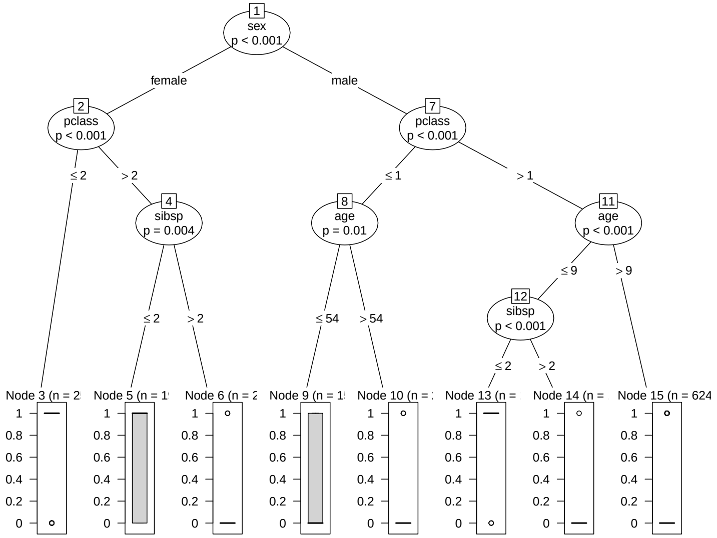


Classification using ctree

```
> (titanic.ctree <- ctree(survived ~ pclass + sex + age + sibsp,
   data=titanic3))

 Conditional inference tree with 8 terminal nodes

Response:  survived
Inputs:  pclass, sex, age, sibsp
Number of observations:  1309

1) sex == {female}; criterion = 1, statistic = 365.607
  2) pclass <= 2; criterion = 1, statistic = 105.161
    3)*  weights = 250
  2) pclass > 2
    4) sibsp <= 2; criterion = 0.996, statistic = 10.8
      5)*  weights = 195
    4) sibsp > 2
      6)*  weights = 21
1) sex == {male}
  7) pclass <= 1; criterion = 1, statistic = 24.611
    8) age <= 54; criterion = 0.99, statistic = 9.079
      9)*  weights = 151
    8) age > 54
      10)*  weights = 28
  7) pclass > 1
    11) age <= 9; criterion = 1, statistic = 25.406
      12) sibsp <= 2; criterion = 1, statistic = 22.192
        13)*  weights = 24
      12) sibsp > 2
        14)*  weights = 16
    11) age > 9
      15)*  weights = 624
```

Classification using ctree

```
> plot(titanic.ctree)
```



Receiver Operating Characteristic (ROC) Graphs

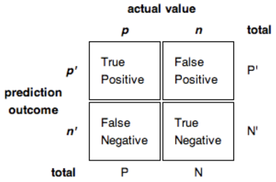
In a classification model, outcomes are either as positive (p) or negative (n). There are then four possible outcomes:

true positive (TP) The outcome from a prediction is p and the actual value is also p

false positive (FP) The actual value is n

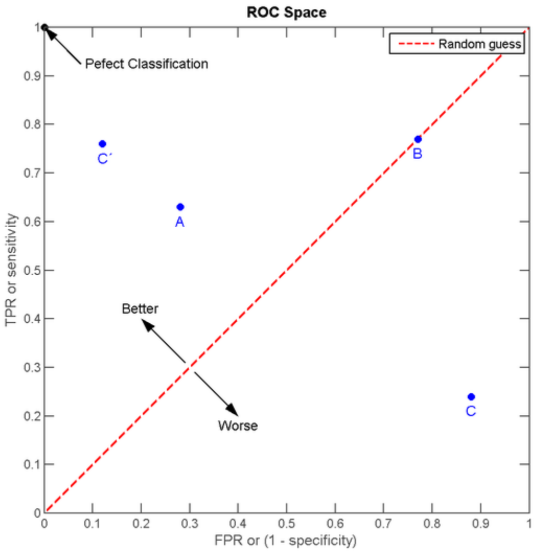
true negative TN) Both the prediction outcome and the actual value are n.

false negative (FN) The prediction outcome is n while the actual value is p.



		actual value		
		p	n	total
prediction outcome	p'	True Positive	False Positive	P'
	n'	False Negative	True Negative	N'
	total	P	N	

ROC Space

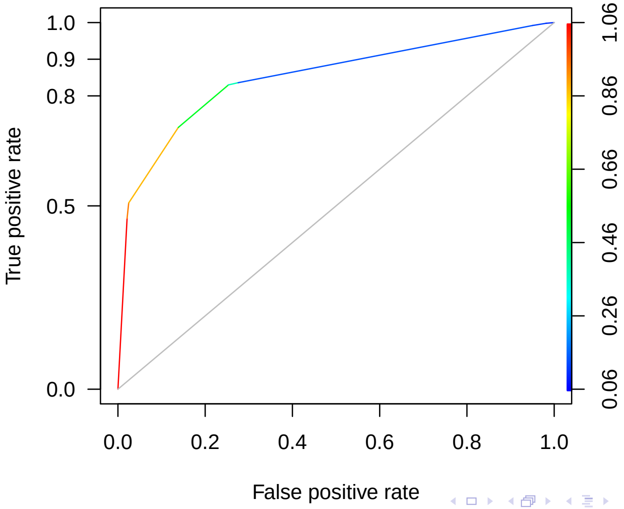

ROC Space
Random guess
Pefect Classification
C'
A
B
C
Better
Worse
TPR or sensitivity
FPR or (1 - specificity)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

ROCR

The `ROCR` package provides three functions to plot ROCs.

```
> titanic.pred <- predict(titanic.ctree)
> pred <- prediction(titanic.pred, as.integer(titanic3$survived))
> perf <- performance(pred, measure="tpr", x.measure="fpr")
> plot(perf, colorize=TRUE, yaxis.at=c(0,0.5,0.8,0.9,1),
   yaxis.las=1)
> lines(c(0,1), c(0,1), col="grey")
```

ROCR



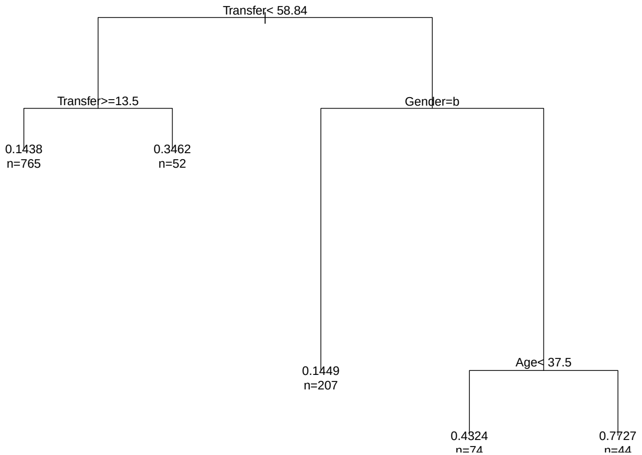
True positive rate
False positive rate
0.0
0.5
0.8
0.9
1.0
0.0
0.2
0.4
0.6
0.8
1.0
0.06
0.26
0.46
0.66
0.86
1.06

New Student Outreach Example

One issue with CART methods is that in many instances the models may over fit the data. In this example we will examine a new student outreach program.

```
> names(students)
 [1] "treat"       "Course"      "Grade"
 [4] "Gender"      "Ethnicity"   "Military"
 [7] "ESL"         "EdMother"    "EdFather"
[10] "Age"         "Employment"  "Income"
[13] "Transfer"    "GPA"         "GradeCode"
[16] "Level"       "ID"          "Treat"
> rp = rpart(Treat ~ Age + Ethnicity + Military + Gender +
   Employment + Transfer + ESL +
   EdMother + EdFather + Income, data=students)
> plot(rp); text(rp, cex=1, use.n=TRUE)
```

New Student Outreach Example



New Student Outreach Example

The `where` element contains the leaf node for each row used to grow the tree (note that for `ctree` use the `where` function). Over-fitting is generally not an issue in propensity score methods since we do not wish to generalize the results in phase I. However, we do need students from both the treatment and comparison groups in each leaf node.

```
> strata = factor(rp$where)
> table(strata, students$Treat)
```

```
strata FALSE TRUE
     3   655  110
     4    34   18
     6   177   30
     8    42   32
     9    10   34
```

New Student Outreach Example

Typically, you will want to select a tree size that minimizes the cross-validated error, the `xerror` column printed by `printcp`.

```
> printcp(rp)

Regression tree:
rpart(formula = Treat ~ Age + Ethnicity + Military + Gender +
    Employment + Transfer + ESL + EdMother + EdFather + Income,
    data = students)

Variables actually used in tree construction:
[1] Age      Gender   Transfer

Root node error: 180.06/1142 = 0.15767

n= 1142

        CP nsplit rel error  xerror     xstd
1 0.048261      0   1.00000 1.00131 0.045347
2 0.017746      2   0.90348 0.93146 0.045679
3 0.011073      3   0.88573 0.93167 0.046707
4 0.010000      4   0.87466 0.92843 0.047139
```

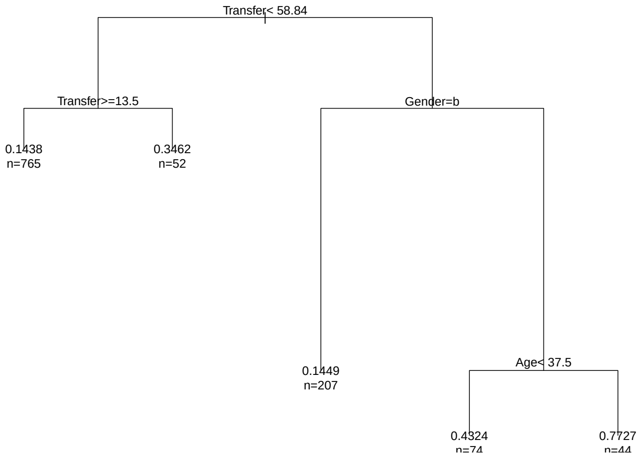
Or we can extract the smallest complexity parameter for the smallest cross-validated error.

```
> (cp4min <- rp$cptable[which.min(rp$cptable[,"xerror"]),"CP"])

[1] 0.01
```

New Student Outreach Example

```
> rp2 = prune(rp, cp=cp4min - .001)
> plot(rp2); text(rp2, use.n=TRUE, all=FALSE)
```

New Student Outreach Example



New Student Outreach Example

Crosstab of strata before pruning

```
> strata = factor(rp$where)
> table(strata, students$Treat)

strata FALSE TRUE
     3   655  110
     4    34   18
     6   177   30
     8    42   32
     9    10   34
```

Crosstab of strata after pruning

```
> strata2 = factor(rp2$where)
> table(strata2, students$Treat)

strata2 FALSE TRUE
      3   655  110
      4    34   18
      6   177   30
      8    42   32
      9    10   34
```

Titanic Revisited: Logistic Regression

Lets revisit the Titanic dataset and compare the tree method with logistic regression. First, we need to impute missing values. In this dataset only age is missing with about 20% missing.

```
> titanic.mice <- mice(titanic3[,c("pclass","sex","age","sibsp")])
> titanic.complete <- cbind(survived=titanic3$survived,
   complete(titanic.mice, 5))
```

Perform a logistic regression with the `glm` function.

```
> titanic.glm <- glm(survived ~ pclass + sex + age + sibsp,
       data=titanic.complete,
       family=binomial(logit))
```

Titanic Revisited: Logistic Regression

```
> summary(titanic.glm)

Call:
glm(formula = survived ~ pclass + sex + age + sibsp, family = binomial(logit),
    data = titanic.complete)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4758  -0.6667  -0.4443   0.6571   2.5105

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.46586    0.38392  11.632  < 2e-16 ***
pclass      -1.08367    0.10077 -10.754  < 2e-16 ***
sex2        -2.57730    0.15253 -16.897  < 2e-16 ***
age         -0.02823    0.00585  -4.827 1.39e-06 ***
sibsp       -0.30874    0.08287  -3.726 0.000195 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1741.0  on 1308  degrees of freedom
Residual deviance: 1223.2  on 1304  degrees of freedom
AIC: 1233.2

Number of Fisher Scoring iterations: 4
```

Titanic Revisited: Logistic Regression

But from our tree methods it appears there is an interaction effect between class and gender.

```
> titanic.glm2 <- glm(survived ~ pclass + sex + pclass:sex + age + sibsp,
   data=titanic.complete, family=binomial(logit))
> summary(titanic.glm2)
```

```
Call:
glm(formula = survived ~ pclass + sex + pclass:sex + age + sibsp,
    family = binomial(logit), data = titanic.complete)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0358  -0.6537  -0.5128   0.4890   2.3767

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  7.009329   0.681902  10.279  < 2e-16 ***
pclass      -2.055083   0.224290  -9.163  < 2e-16 ***
sex2        -5.811581   0.646529  -8.989  < 2e-16 ***
age         -0.029838   0.006137  -4.862 1.16e-06 ***
sibsp       -0.296704   0.084347  -3.518 0.000435 ***
pclass:sex2  1.319960   0.240258   5.494 3.93e-08 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1741.0  on 1308  degrees of freedom
Residual deviance: 1184.6  on 1303  degrees of freedom
AIC: 1196.6

Number of Fisher Scoring iterations: 5
```

Agenda

Ensemble Methods

Ensemble methods use multiple models that are combined by weighting, or averaging, each individual model to provide an overall estimate. Each model is a random sample of the sample. Common ensemble methods include:

- *Boosting* - Each successive trees give extra weight to points incorreclty predicted by earlier trees. After all trees have been estimated, the prediction is determined by a weighted "vote" of all predictions (i.e. results of each individual tree model).
- *Bagging* - Each tree is estimated independent of other trees. A simple "majority vote" is take for the prediction.
- *Random Forests* - In addition to randomly sampling the data for each model, each split is selected from a random subset of all predictors.

Random Forests

The random forest algorithm works as follows:

1. Draw n_{tree} bootstrap samples from the original data.
2. For each bootstrap sample, grow an unpruned tree. At each node, randomly sample m_{try} predictors and choose the best split among those predictors selected[2].
3. Predict new data by aggregating the predictions of the n_{tree} trees (majority votes for classification, average for regression).

[2] Bagging is a special case of random forests where $m_{try} = p$, where *p* is the number of predictors

Random Forests

The random forest algorithm works as follows:

1. Draw n_{tree} bootstrap samples from the original data.
2. For each bootstrap sample, grow an unpruned tree. At each node, randomly sample m_{try} predictors and choose the best split among those predictors selected[2].
3. Predict new data by aggregating the predictions of the n_{tree} trees (majority votes for classification, average for regression).

Error rates are obtained as follows:

1. At each bootstrap iteration predict data not in the bootstrap sample (what Breiman calls "out-of-bag", or OOB, data) using the tree grown with the bootstrap sample.
2. Aggregate the OOB predictions. On average, each data point would be out-of-bag 36% of the times, so aggregate these predictions. The calculated error rate is called the OOB estimate of the error rate.

[2] Bagging is a special case of random forests where $m_{try} = p$, where *p* is the number of predictors

Random Forests: Titanic Revisited

```
> set.seed(2112)
> titanic.rf <- randomForest(factor(survived) ~ pclass + sex + age + sibsp,
       data=titanic.complete,
       ntree=5000,
       importance=TRUE)
> print(titanic.rf)

Call:
 randomForest(formula = factor(survived) ~ pclass + sex + age +      sibsp, data = titan
               Type of random forest: classification
                     Number of trees: 5000
No. of variables tried at each split: 2

        OOB estimate of  error rate: 19.94%
Confusion matrix:
    0   1 class.error
0 723  86   0.1063041
1 175 325   0.3500000

> importance(titanic.rf)

               0          1 MeanDecreaseAccuracy MeanDecreaseGini
pclass  94.50820 108.687422            130.46118         61.72316
sex    281.79319 354.620397            355.50247        167.08670
age     99.32642  53.592652            124.83594         67.34050
sibsp   79.64061  -8.590617             65.56774         19.59650
```

Agenda

Discussion

- CART Methods for Propensity Score Analysis
  - Overfitting?
  - Stratification
- CART Methods for Data Mining
  - Splitting datasets
  - Overfitting
- Missing Data
- Results Informing other Regression Methods
- Ensemble Method

Thank You

Jason Bryer (jason@bryer.org)
`http://jason.bryer.org`