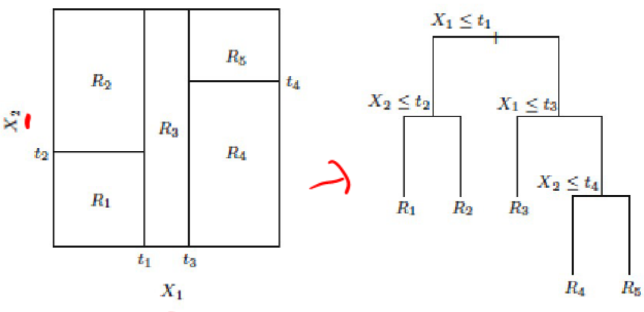
Lecture 7

Classification And Regression Trees (CART)
Bagging and Random Forests

Lecture 6 Review:

- CART: Classification And Regression Trees
  - 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
  - Breaks the predictor X space into non-overlapping sub-spaces, calls these $\boxed{R_m}$
  - Build an initial large tree then prune the tree such that the final tree has smallest prediction error

T_0



- Exploring this within NMES, predicting log(totalexp+1) and big expenditure

Regression Trees

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

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

E(Y|X)

- R_m are selected to minimize

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

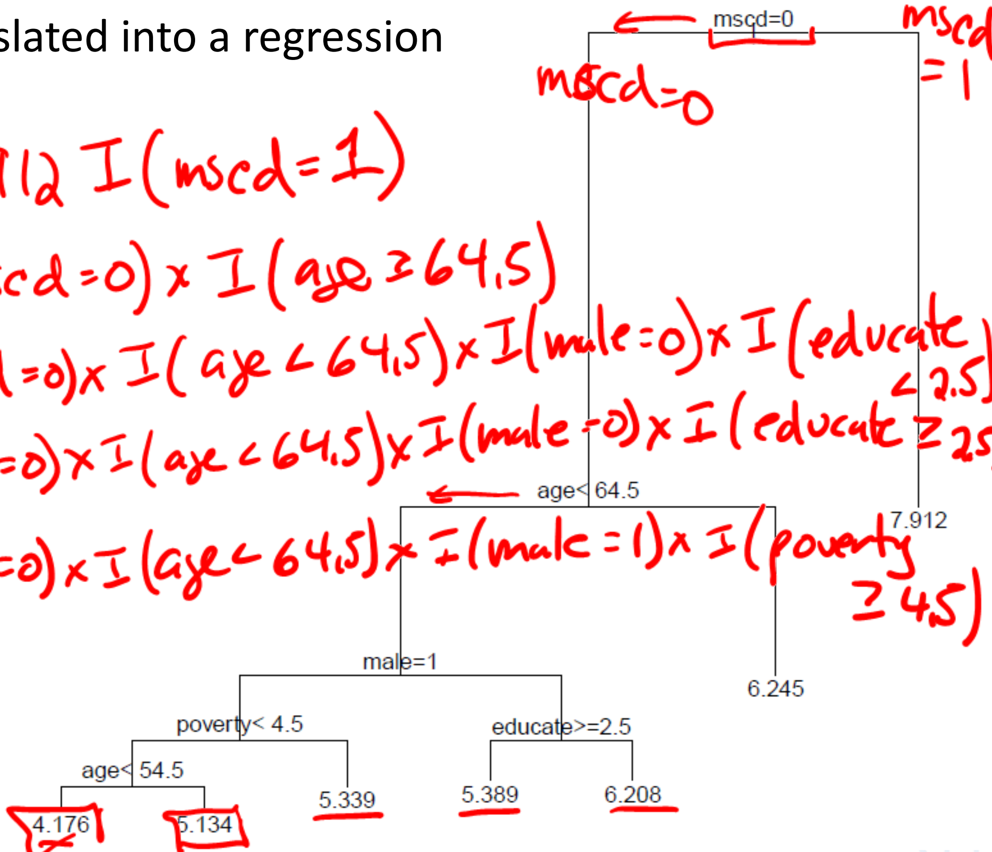
- c_m are estimated via the mean Y in R_m

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

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

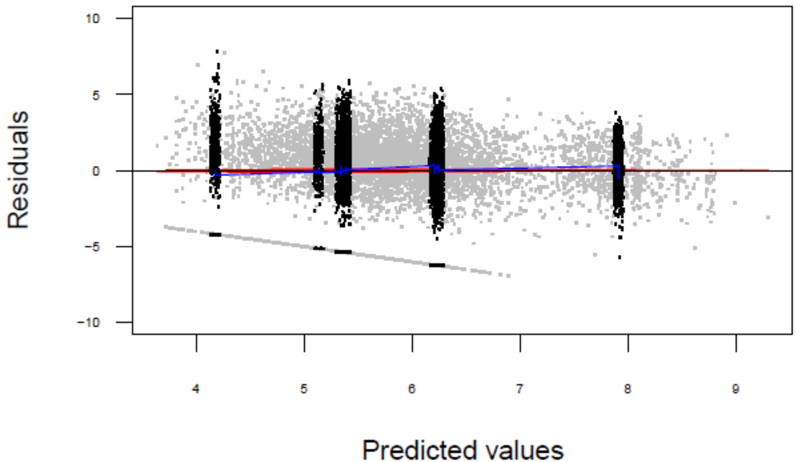
A CART can be translated into a regression model! You do….

$$\hat{E}(e|X) = 7.912\, I(mscd=1)$$
$$+\ 6.245\, I(mscd=0)\times I(age \ge 64.5)$$
$$+\ 6.208\, I(mscd=0)\times I(age < 64.5)\times I(male=0)\times I(educate < 2.5)$$
$$+\ 5.389\, I(mscd=0)\times I(age < 64.5)\times I(male=0)\times I(educate \ge 2.5)$$
$$+\ 5.334\, I(mscd=0)\times I(age < 64.5)\times I(male=1)\times I(poverty \ge 4.5)$$



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

CART approach to missing data: surrogate variables

```
## Node number 1: 6151 observations,    complexity param=0.07534598
##   mean=5.908773, MSE=6.708663
##   left son=2 (5463 obs) right son=3 (688 obs)
##   Primary splits:
##       mscd    splits as  LR,          improve=0.075345980, (0 missing)
##       age     < 64.5 to the left,  improve=0.047221950, (0 missing)
##       male    splits as  RL,          improve=0.008848244, (0 missing)
##       married splits as  LRLLL,     improve=0.007339919, (0 missing)
##       beltuse < 2.5  to the left,  improve=0.005627789, (0 missing)
##   Surrogate splits:
##       age < 93.5 to the left,  agree=0.888, adj=0.003, (0 split)
##
## Node number 2: 5463 observations,    complexity param=0.02686116
##   mean=5.656467, MSE=6.668967
##   left son=4 (3446 obs) right son=5 (2017 obs)
##   Primary splits:
##       age     < 64.5 to the left,  improve=0.030424030, (0 missing)
##       male    splits as  RL,          improve=0.013146610, (0 missing)
##       beltuse < 2.5  to the left,  improve=0.006562917, (0 missing)
##       educate < 2.5  to the right, improve=0.006149474, (0 missing)
##       married splits as  LRLLL,     improve=0.004817598, (0 missing)
##   Surrogate splits:
##       married splits as  LRLLL,     agree=0.715, adj=0.229, (0 split)
##       educate < 3.5  to the left,  agree=0.658, adj=0.074, (0 split)
...
```

Classification trees

- Same procedure
  - Different goodness of fit criteria

y_i = Binary
$k = 0, 1$

For classification trees, define

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

R_m : $\hat{p}_{m0}$ $\hat{p}_{m1}$

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:

R_m : $\hat{p}_{m0} = .7$ $\hat{p}_{m1} = .3$
Classification $R_m = 0$
misclassification error $= .3$

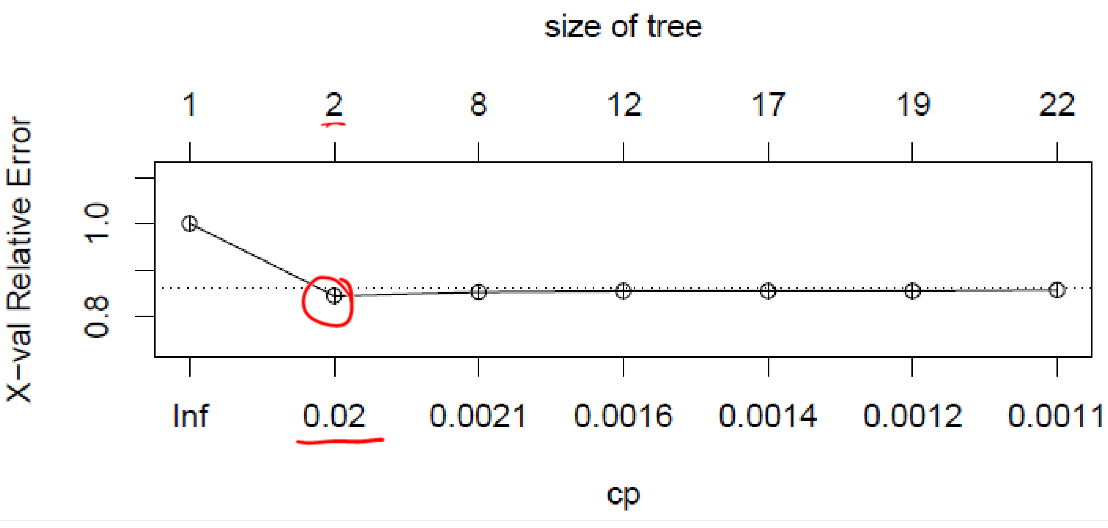
- Misclassification error: $1 - max(p, 1-p)$
- Gini index: $2p(1-p)$
- Cross entropy or deviance: $-plog(p) - (1-p)log(1-p)$

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

Example: Predict a big expenditure

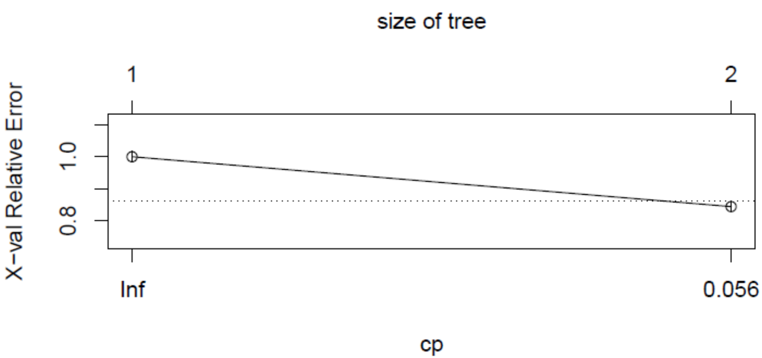
```
set.seed(123454321)
dat1.train=dat1[train<-sample(1:nrow(dat),floor(nrow(dat)/2)),]
dat1.test=dat1[-train,]
tree0=rpart(big~.,data=dat1.train,method="class",control=rpart.control(minsize=20,cp=.001))
par(mfrow=c(2,1),mar=c(5,5,5,1))
plotcp(tree0)
```

big = { 1 >1,000 ; 0 o/w }

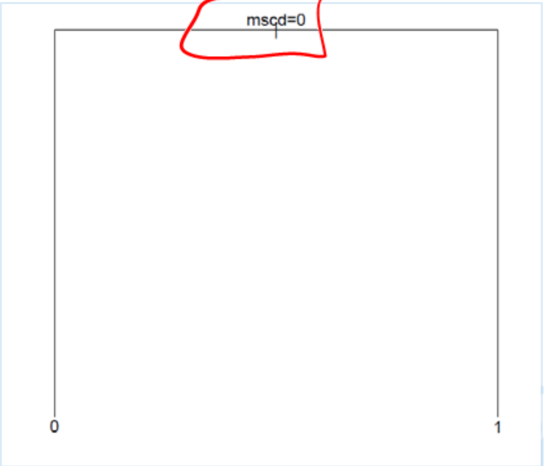


Example: Predict a big expenditure

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



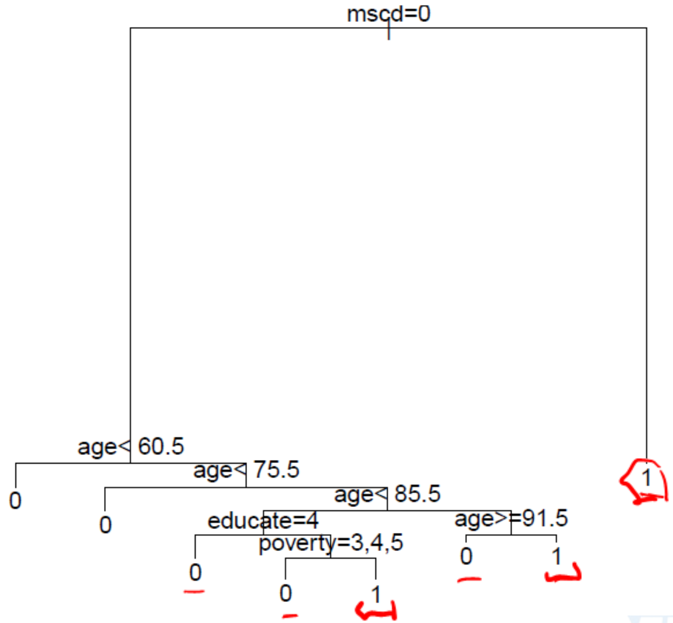

```
##           CP nsplit rel error    xerror       xstd
## 1 0.1555751      0 1.0000000 1.0000000 0.01667776
## 2 0.0200000      1 0.8444249 0.8444249 0.01600726
##
## Variable importance
## mscd
##  100
```



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")
```

minsize = 20

$\hat{Pr}(Y_i=0), \hat{Pr}(Y_i=1)$

$\hat{P}_{m1}$

```
# 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")
```

minsize = 5

Example: Predict a big expenditure

- Compare the AUC values

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

Parametric model

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

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

→ 1st tree ⇒ mscd

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

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

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

→ 2nd tree
mscd, age, education

HEART: Foundations of Statistical Machine Learning

Class 8: November 8th, 2018

Shannon Wongvibulsin
Johns Hopkins School of Medicine
Biomedical Engineering Department
MD/PhD Candidate

Ensemble Methods

- **Bagging** (**B**ootstrap **Agg**regat**ing**)
- **Random Forest**

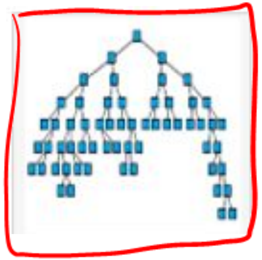


Image source:
https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjMmruX-cLeAhVrmuAKHXETDpQQjRx6BAgBEAU&url=https%3A%2F%2Fcloudblogs.microsoft.com%2Fmicrosoftsecure%2F2018%2F02%2F14%2Fhow-artificial-intelligence-stopped-an-emotet-outbreak%2F&psig=AOvVaw20oWhcs-VEjNXILgXX-cVW&ust=1541702921680011

Ensemble Methods

- **Bagging** (**B**ootstrap **Agg**regat**ing**)
- **Random Forest**

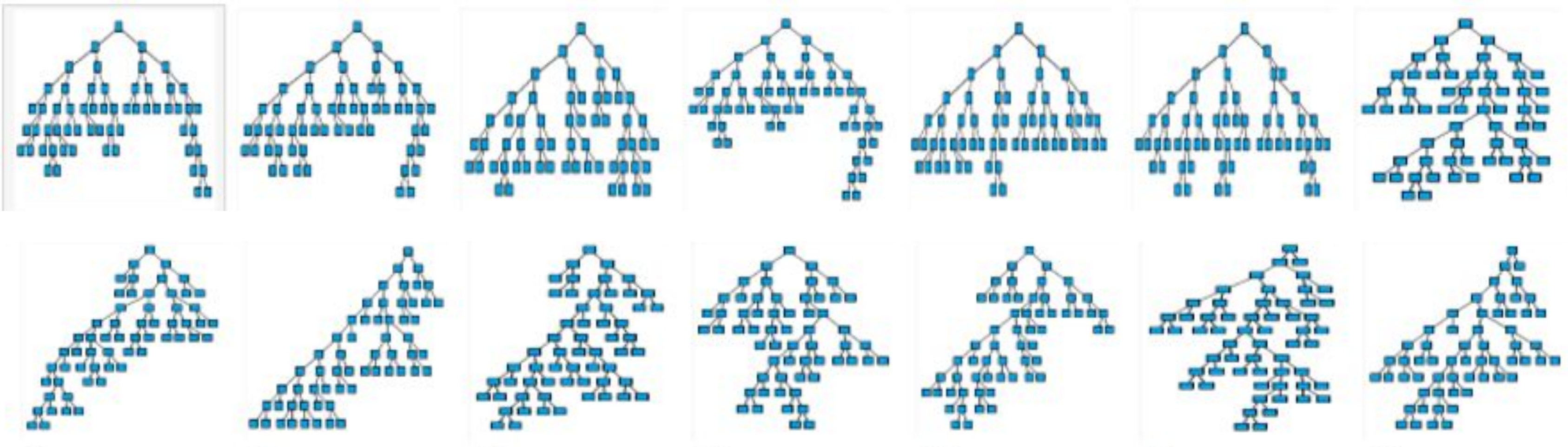


Image source:
https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjMmruX-cLeAhVrmuAKHXETDpQQjRx6BAgBEAU&url=https%3A%2F%2Fcloudblogs.microsoft.com%2Fmicrosoftsecure%2F2018%2F02%2F14%2Fhow-artificial-intelligence-stopped-an-emotet-outbreak%2F&psig=AOvVaw20oWhcs-VEjNXILgXX-cVW&ust=1541702921680011

Bootstrapping

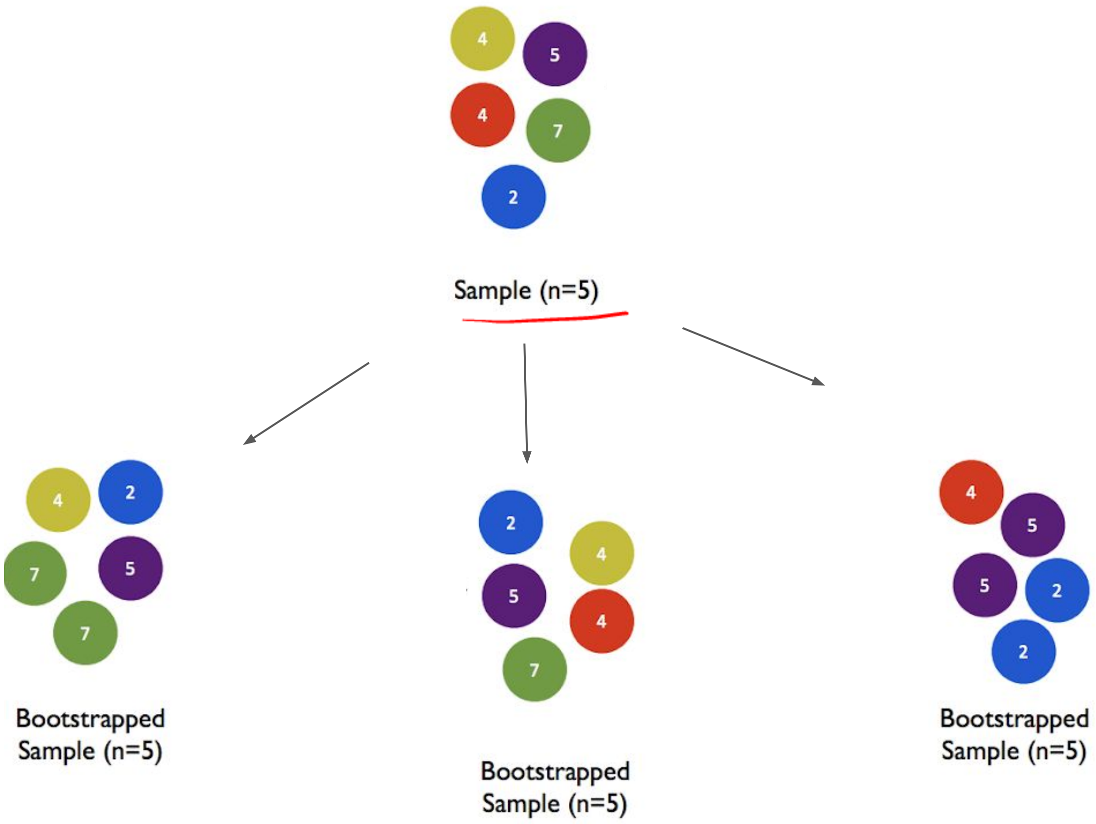


Image source: http://web.stanford.edu/class/psych252/tutorials/doBootstrapPrimer.pdf

Bagging

Training Sample
nxp

n: number of observations
p: number of predictors

Create B bootstrap samples by sampling with replacement from the training sample

Breiman, Machine learning (2001)

Bagging

Training Sample
nxp

n: number of observations
p: number of predictors

Create B bootstrap samples by sampling with replacement from the training sample

Bootstrap Sample 1 (“In-Bag”)
nxp

“Out-of-Bag” Data

Breiman, Machine learning (2001)

Bagging

Training Sample
nxp

n: number of observations
p: number of predictors

Create B bootstrap samples by sampling with replacement from the training sample

Bootstrap Sample 1
(“In-Bag”)
nxp

“Out-of-Bag” Data

Bootstrap Sample 2
(“In-Bag”)
nxp

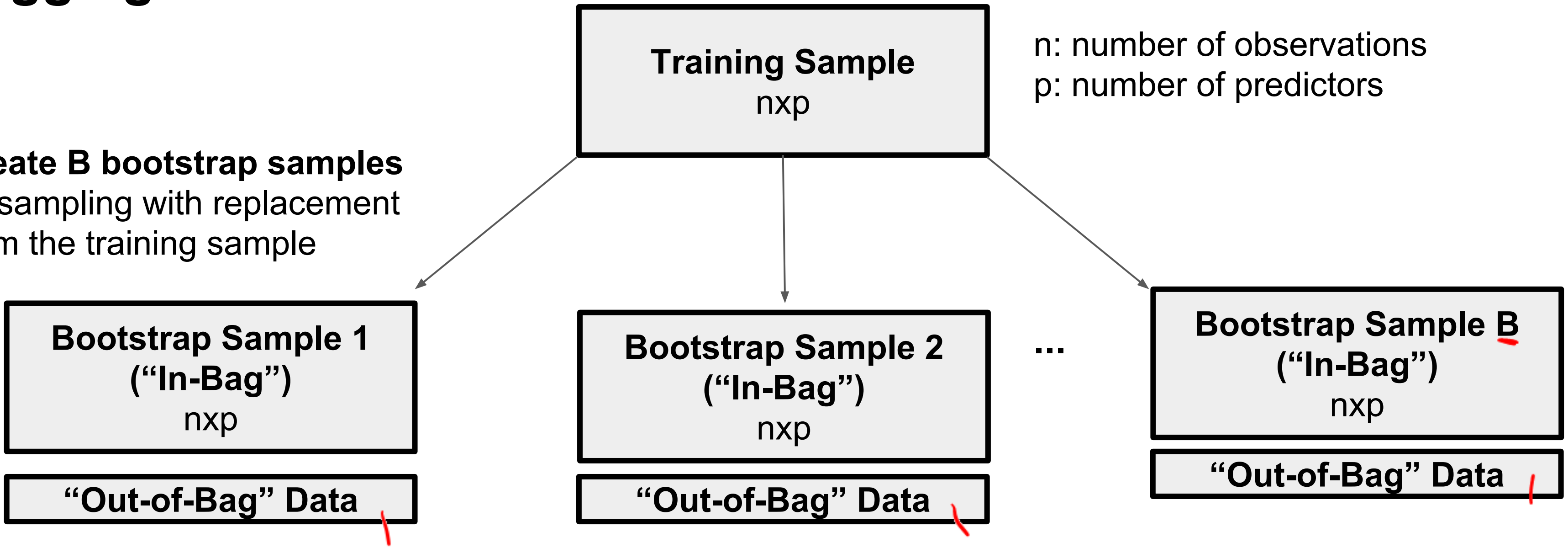
“Out-of-Bag” Data

Breiman, Machine learning (2001)

Bagging

n: number of observations
p: number of predictors

Create B bootstrap samples by sampling with replacement from the training sample



Breiman, Machine learning (2001)

Bagging

Training Sample
nxp

n: number of observations
p: number of predictors

Create B bootstrap samples by sampling with replacement from the training sample

Bootstrap Sample 1 ("In-Bag")
nxp

"Out-of-Bag" Data

Bootstrap Sample 2 ("In-Bag")
nxp

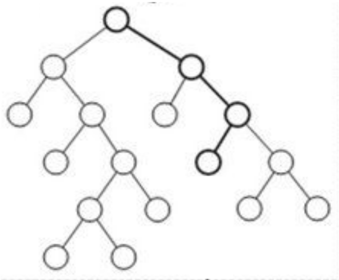
"Out-of-Bag" Data

...

Bootstrap Sample B ("In-Bag")
nxp

"Out-of-Bag" Data

Build one tree from each of the B bootstrap samples



Breiman, Machine learning (2001)

Random Forests

Training Sample
nxp

n: number of observations
p: number of predictors

Create B bootstrap samples by sampling with replacement from the training sample

Bootstrap Sample 1 ("In-Bag")
nxp

"Out-of-Bag" Data

Bootstrap Sample 2 ("In-Bag")
nxp

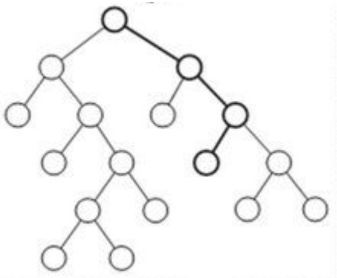
"Out-of-Bag" Data

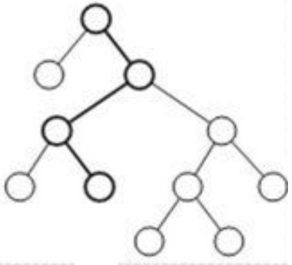
...

Bootstrap Sample B ("In-Bag")
nxp

"Out-of-Bag" Data

Build one tree from each of the B bootstrap samples





Breiman, Machine learning (2001)

Bagging

Training Sample
nxp

n: number of observations
p: number of predictors

Create B bootstrap samples by sampling with replacement from the training sample

Bootstrap Sample 1 ("In-Bag")
nxp

"Out-of-Bag" Data

Bootstrap Sample 2 ("In-Bag")
nxp

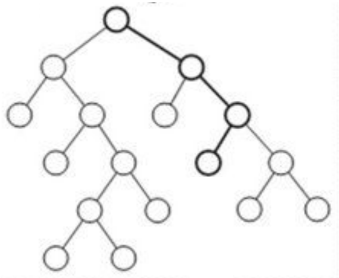
"Out-of-Bag" Data

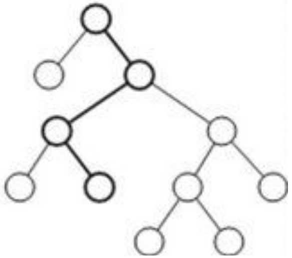
...

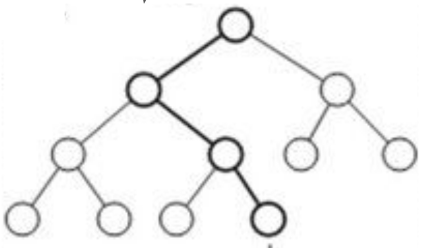
Bootstrap Sample B ("In-Bag")
nxp

"Out-of-Bag" Data

Build one tree from each of the B bootstrap samples



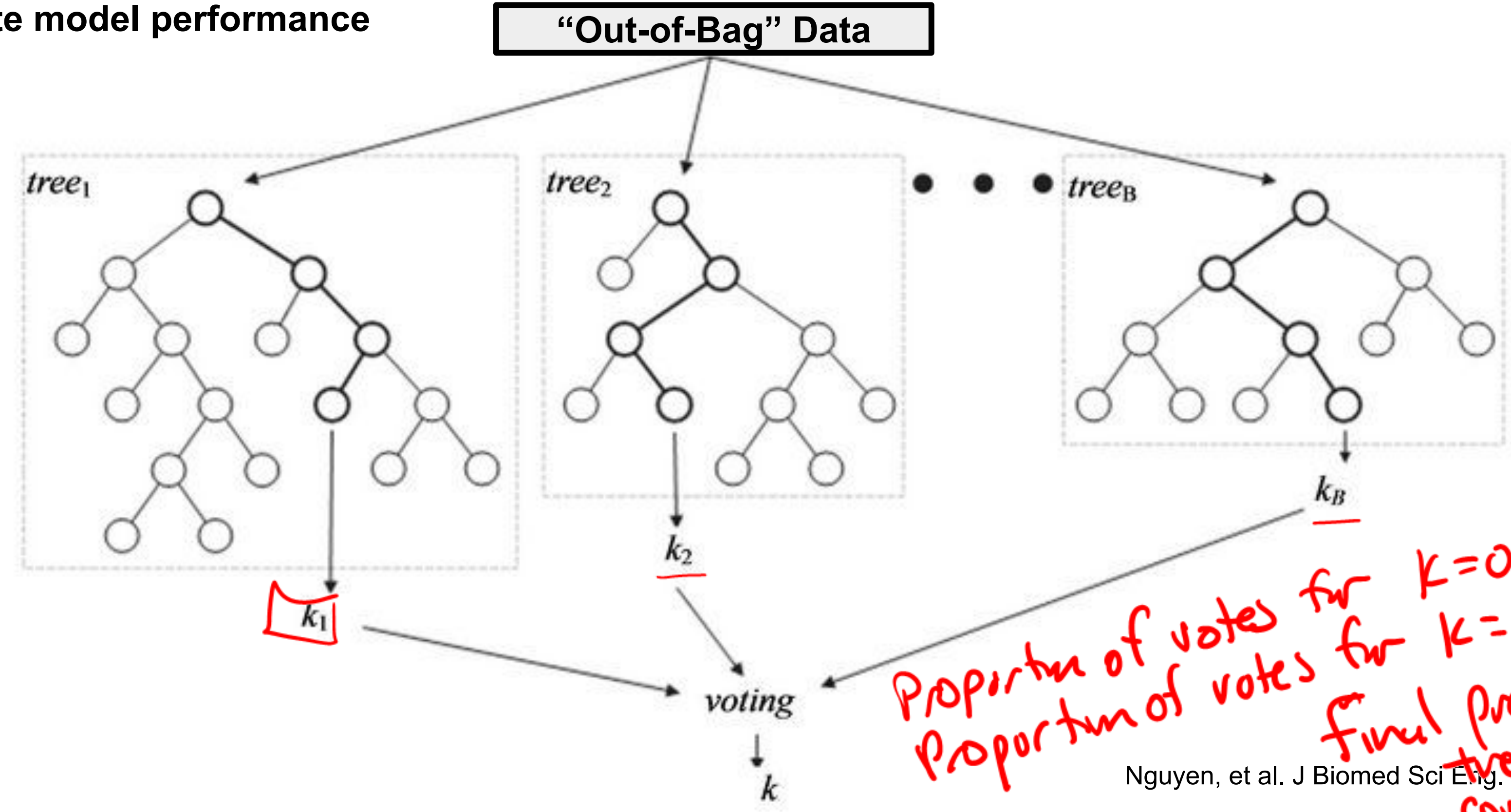




Breiman, Machine learning (2001)

Bagging

Use "Out-of-Bag" (OOB) data to estimate model performance



Y_i = binary

Proportion of votes for $K=0$,
Proportion of votes for $K=1$
final prediction as the most commonly

Nguyen, et al. J Biomed Sci Eng. (2013)

Bagging

Issue with decision trees: **high variance (overfitting)**

- Example:
  - Build decision trees on data split in random different ways
  - Decision trees give different results (high variance)

Addressing the high variance problem with bagging:

1. Build decision trees on B **bootstrap*** samples
2. Average predictions over all decision trees

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

Note: no pruning of trees

*Recall: **Bootstrapping**:

- Sample with replacement
- In-bag: ~⅔ of data
- Out-of-bag: ~⅓ of data

Potential Issue with Bagging

- **Correlated trees**
  - Example:
    - One very strong predictor
      - All bagged trees will select strong predictor at top of tree
      - All bagged trees will be similar

How to decorrelate trees constructed from bootstrap samples?

Random Forest

Random Forests

Training Sample
nxp

n: number of observations
p: number of predictors

Create B bootstrap samples by sampling with replacement from the training sample

Bootstrap Sample 1
("In-Bag")
nxp

"Out-of-Bag" Data

Bootstrap Sample 2
("In-Bag")
nxp

"Out-of-Bag" Data

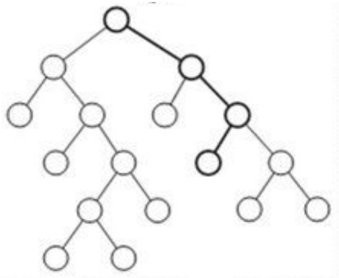
...

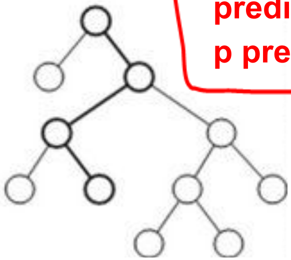
Bootstrap Sample B
("In-Bag")
nxp

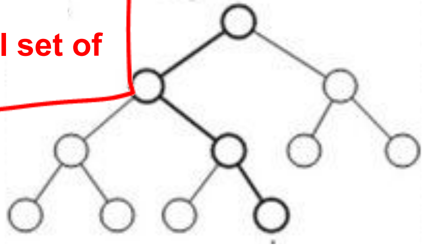
"Out-of-Bag" Data

Build one tree from each of the B bootstrap samples

At each split point, consider a random sample of m predictors from the full set of p predictors.







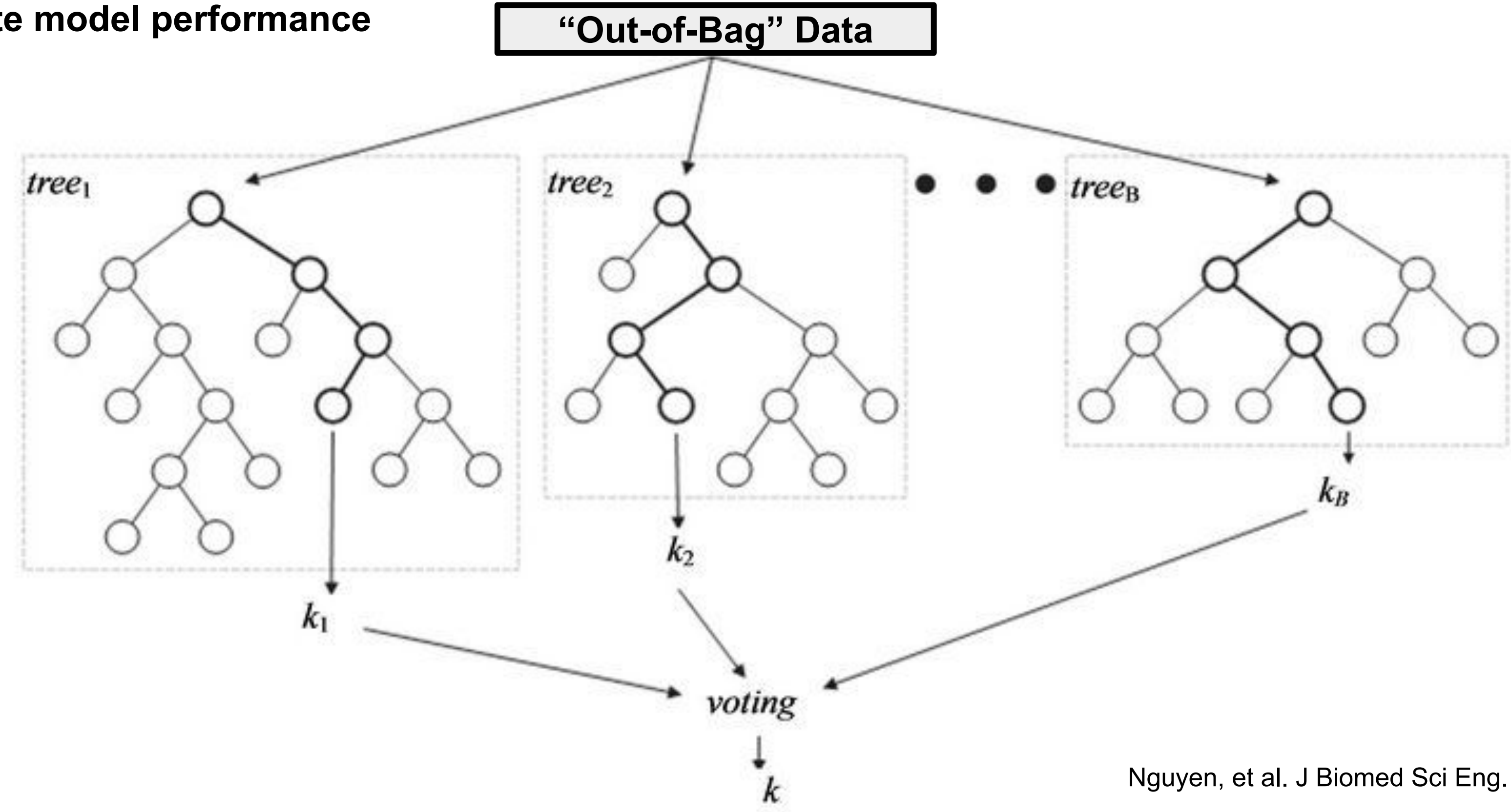
Breiman, Machine learning (2001)

Random Forest

- 1. Build decision trees on B **bootstrap** samples
    - **When a split is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors.**
        - Note:
            - $m \leq p$
            - If $m = p$, bagging
- 2. Average predictions over all decision trees

Random Forests

Use "Out-of-Bag" (OOB) data to estimate model performance



Nguyen, et al. J Biomed Sci Eng. (2013)

Random Forest Algorithm, HTF text

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\mathrm{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\mathrm{rf}}^B(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B$.

Random Forest Extras

- Missing values in the training data:
  - Random forests do not like missing values in the training data
  - First impute missing data:
    - Mean / mode replacement
    - Imputation via proximity; see these youtube videos which do a good job of giving the overview of how the imputation works

  https://www.youtube.com/watch?v=J4Wdy0Wc_xQ

  https://www.youtube.com/watch?v=nyxTdL_4Q-Q

- Missing values for testing/validation data:
  - Surrogate variables are used.

- Given the internal cross-validation that occurs, do we need to separate data into training and test/validation?
  - Not really! So long as you evaluate the utility of the random forest using the out-of-bag predictions/error!

  - In PS2, you will be using a training and test/validation because for learning we will ask you to compare the utility of a parametric model, a single CART and a random forest. But for applications where you will use the random forest, you don't need to separate the data.

Random Forest Extras

- Parameters that we control
  - Number of variables considered at each split, $\boxed{m}$
    - Classification tree: floor square-root p
    - Regression tree: floor p/3
  - Number of trees
  - Recommendation: To find m: set number of trees large (e.g. 500), identify minimum out-of-bag error for m = 1, 2, …, beyond default. After finding m: check to see if your forest is sensitive to number of trees by plotting MSE or out-of-bag error as a function of number of trees.

* minimum node size = 5

- Out of bag samples

  For each observation $z_i = (x_i, y_i)$, construct its random forest predictor by averaging only those trees corresponding to bootstrap samples in which z_i did not appear.

  - Out of bag error -> corresponds to a n-fold cross-validation

- Variable importance
  - Ranks each variable by summing up (over all trees) "improvement" in prediction error when variable is included