Supervised learning

C. Maugis-Rabusseau & P. Neuvial

IMT

2021-07-01

Credits

Many ideas and illustrations borrowed from

- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112, p. 18). New York: springer.
- Albert, M., Besse, P. Laurent, B. (2021). *Cours de Machine learning, INSA GMM 4è année (2020-2021)*
- Villa-Vialaneix, N. (2011). *Machine Learning; formation INRA pour la plate-forme Biostat*

Case study: SAheart data

```r
data(SAheart, package = "bestglm")
dim(SAheart)
```

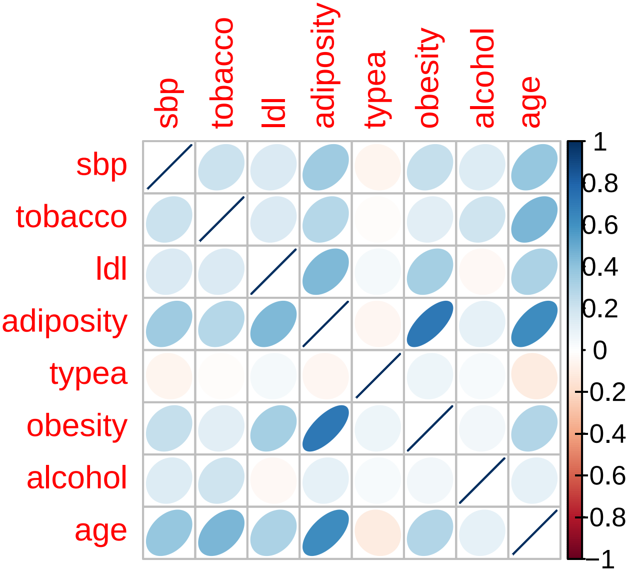
```
[1] 462  10
```

```r
SAheart$chd <- as.factor(SAheart$chd)
```

A retrospective sample of 462 males in a heart-disease high-risk region of the Western Cape, South Africa.

1. sbp : systolic blood pressure
2. tobacco : cumulative tobacco (kg)
3. ldl : low density lipoprotein cholesterol
4. adiposity : body adiposity index (?)
5. famhist : family history of heart disease (Present, Absent)
6. typea : type-A behavior
7. obesity : body mass index
8. alcohol : current alcohol consumption
9. age : age at onset
10. *chd* : response, coronary heart disease (0,1)

SAheart data: quantitative variables



SAheart data: qualitative variables

Coronary heart disease (0/1) vs family history (Absent/Present)

```
   Absent Present
0     206      96
1      64      96
```

Outline

- Motivation: beyond linear models
- k nearest neighbors
- Cross-validation
- Classification and Regression Trees
- Random Forests

Note: we will focus on classification but the same ideas apply to regression

caret: One R Prediction Package to rule them all?

"The caret package (short for Classification And REgression Training) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for:

- data splitting
- pre-processing
- feature selection
- model tuning using resampling
- variable importance estimation"

Currently 238 available models (for classification and/or regression)!

Links:

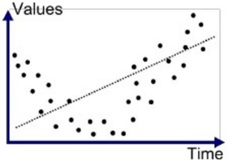
- https://topepo.github.io/caret
- https://topepo.github.io/caret/available-models.html
- https://topepo.github.io/caret/train-models-by-tag.html
- https://github.com/topepo/caret/blob/master/models/files/

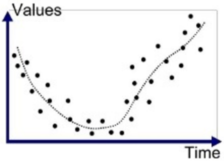
Motivation: beyond linear models

Beyond linear models for regression

Regression setting: $Y = f(X) + \epsilon$, X and Y quantitative

Sometimes linear regression is not flexible enough

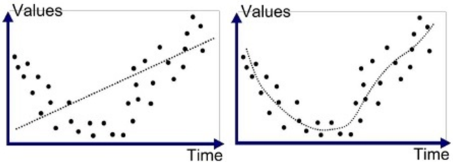




Beyond linear models for regression

Regression setting: $Y = f(X) + \epsilon$, X and Y quantitative

Sometimes linear regression is not flexible enough



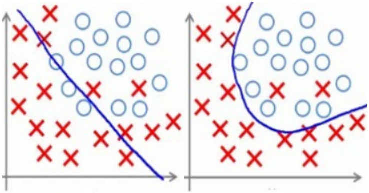
Examples of more flexible methods:

- polynomial regression, splines, local regression, generalized additive models
- regression trees, random forests, support vector regression

Beyond linear models for classification

Classification setting: $Y = f(X) + \epsilon$, X quantitative and Y qualitative

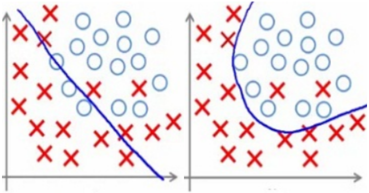
Sometimes a linear detection boundary is not flexible enough



Beyond linear models for classification

Classification setting: $Y = f(X) + \epsilon$, X quantitative and Y qualitative

Sometimes a linear detection boundary is not flexible enough



Examples of classification methods with linear detection boundary:

- logistic regression, linear discriminant analysis, linear SVM

Examples of classification methods with non-linear detection boundary:

- k **nearest-neighbors**, quadratic discriminant analysis, non-linear SVM, **classification trees**, **random forests**

K nearest neighbors

K-nearest neighbors

To predict the class of x_0:

- define $\mathcal{N}(x_0)$: the K points that are closest to x_0
- calculate $P(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}(x_0)} \mathbb{1}_{y_i = j}$
- choose the j with maximum $P(Y = j|X = x_0)$

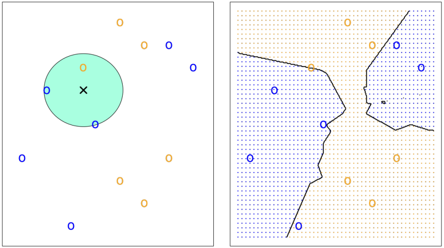


Figure 1: KNN with $K = 3$.

K-nearest neighbors

To predict the class of x_0:

- define $\mathcal{N}(x_0)$: the K points that are closest to x_0
- calculate $P(Y=j|X=x_0) = \frac{1}{K}\sum_{i\in\mathcal{N}(x_0)} \mathbb{1}_{y_i=j}$
- choose the j with maximum $P(Y=j|X=x_0)$

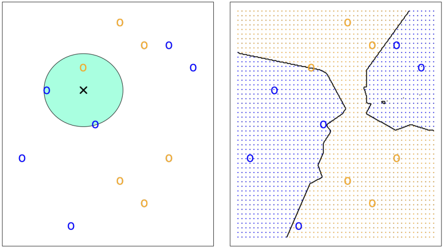


Figure 1: KNN with $K=3$.

Main issue: how should K be chosen?

K-NN with R (take 1)

```r
library(class)
```

Train the model

```r
fit <- knn(train = SAheart[, quant_var],
           test = SAheart[, quant_var],
           cl = SAheart$chd, k = 5)
```

Confusion matrix

```r
table(fit, cl = SAheart$chd)
```

```
   cl
fit   0   1
  0 260  86
  1  42  74
```

K-NN with R (take 1)

```r
library(class)
```

Train the model

```r
fit <- knn(train = SAheart[, quant_var],
           test = SAheart[, quant_var],
           cl = SAheart$chd, k = 5)
```

Confusion matrix

```r
table(fit, cl = SAheart$chd)
```

```
   cl
fit   0   1
  0 260  86
  1  42  74
```

Warning

This not what we whould do! See “Take 2”

K-NN with R (take 1, $K = 1$)

```r
fit <- knn(train = SAheart[, quant_var],
           test = SAheart[, quant_var], cl = SAheart$chd, k = 1)
table(fit, cl = SAheart$chd)
```

```
   cl
fit   0   1
  0 302   0
  1   0 160
```

Perfect classification!?

K-NN with R (take 1, $K = 1$)

```r
fit <- knn(train = SAheart[, quant_var],
           test = SAheart[, quant_var], cl = SAheart$chd, k = 1)
table(fit, cl = SAheart$chd)
```

```
   cl
fit   0   1
  0 302   0
  1   0 160
```

Perfect classification!?

Yes... on the training sample. This is called **overfitting**

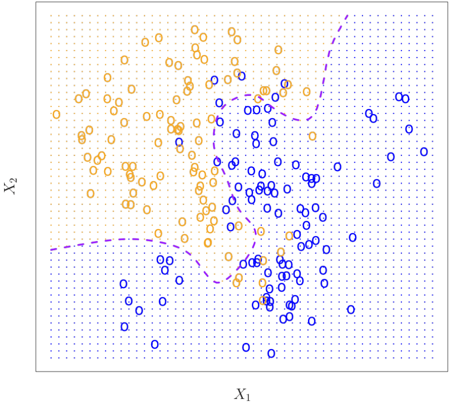
Warning

This not what we whould do! See “Take 2”

Pedagogical example in two dimensions

Simulated data (ISLR book)

- two groups: orange and blue
- optimal decision boundary: purple lines (“Bayes classifier”)



$K = 100$: underfitting

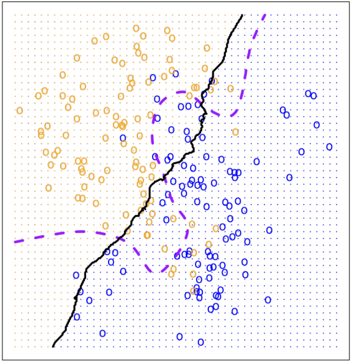


Figure 2: kNN with $K = 100$.

- many errors on the training sample
- poor estimation of the detection boundary (not flexible enough)

$K = 1$: overfitting

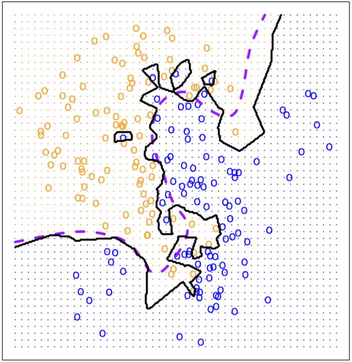


Figure 3: KNN with $K = 1$.

- no errors on the training sample
- poor estimation of the detection boundary (too flexible)

$K = 10$: correct fit

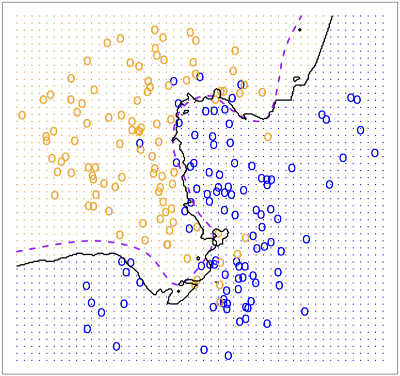
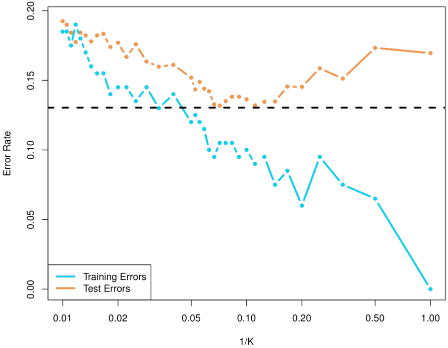


Figure 4: KNN with $K = 10$.

- few errors on the training sample
- correct estimation of the detection boundary

Major issue of model selection: how to choose K?

Classification error on the sample $(X_i, Y_i)_{i\in S}$: $E(K) = \frac{1}{|S|}\sum_{i\in S} \mathbb{1}_{Y_i \neq \hat{f}_K(X_i)}$



- training error: $E(K)$ and $\hat{f}_K$ are estimated on the same sample
  - cannot be used for model selection!
- test error: $E(K)$ and $\hat{f}_K$ are estimated on independent samples
  - how to estimate it in absence of a validation sample? **cross-validation**

K-NN with R (take 2)

$K = 1$

```r
fit <- knn(train = training[, quant_var],
           test = testing[, quant_var],
           cl = training$chd, k = 1)
table(fit, cl = testing$chd)
```

```
   cl
fit  0  1
  0 55 30
  1 20 10
```

$K = 5$

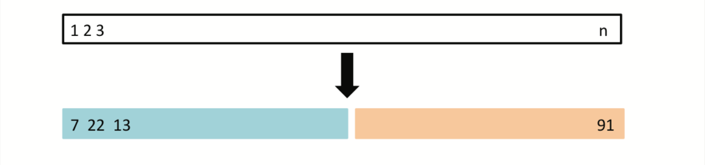
```r
fit <- knn(train = training[, quant_var],
           test = testing[, quant_var],
           cl = training$chd, k = 5)
table(fit, cl = testing$chd)
```

```
   cl
fit  0  1
  0 59 25
  1 16 15
```

Cross-validation

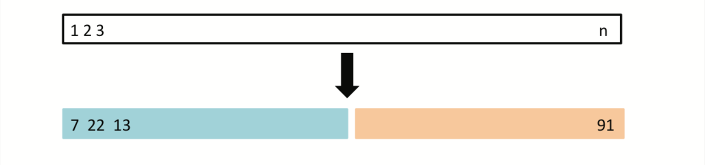
Hold out sample

In absence of test set we can build one from the original sample of n examples by splitting it between a "training set" and a "validation set":



Hold out sample

In absence of test set we can build one from the original sample of n examples by splitting it between a “training set” and a “validation set”:



Drawbacks

- variability of the error estimation
- training on fewer observations may yield to overestimating the test error rate for the model fit on the entire data set.

Hold-out sample with R

```r
library(caret)
set.seed(998)

dat <- SAheart
y <- dat$chd

inTraining <- createDataPartition(y, p = .75, list = FALSE)

training <- dat[ inTraining,]
testing  <- dat[-inTraining,]
```

Hold-out sample with R

```
library(caret)
set.seed(998)

dat <- SAheart
y <- dat$chd

inTraining <- createDataPartition(y, p = .75, list = FALSE)

training <- dat[ inTraining,]
testing  <- dat[-inTraining,]
```

By construction the proportion of 0/1 is preserved:

```
mean(training$chd==0)
```

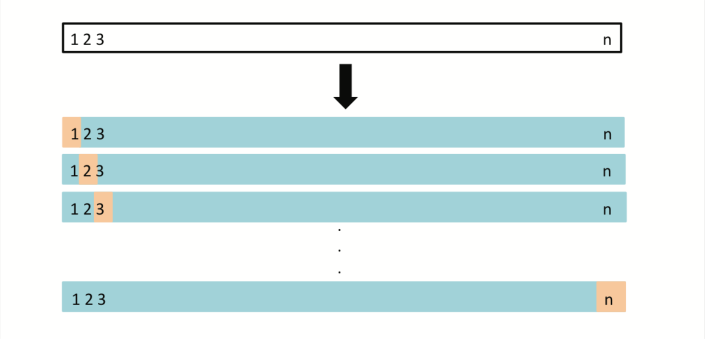
```
[1] 0.6541787
```

```
mean(testing$chd==0)
```

```
[1] 0.6521739
```

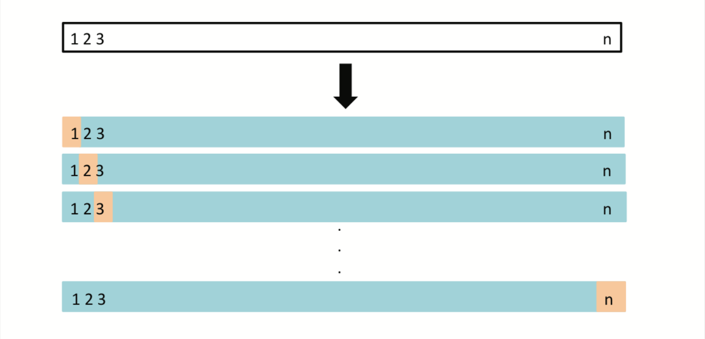
Leave-one-out cross-validation

Create n training sets of size $n-1$, each holding only one example out:



Leave-one-out cross-validation

Create n training sets of size $n-1$, each holding only one example out:

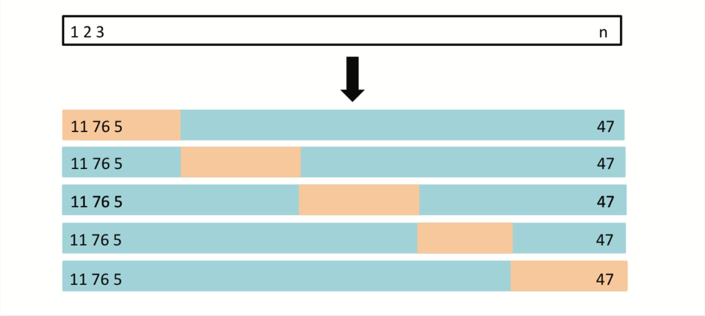


Comparison to hold-out sample

- [+] no variability of the error estimation
- [+] no or very litte overestimation of the test error rate for the model fit on the entire data set.
- [-] (possibly) time-consuming: n models of size $n-1$ to fit!

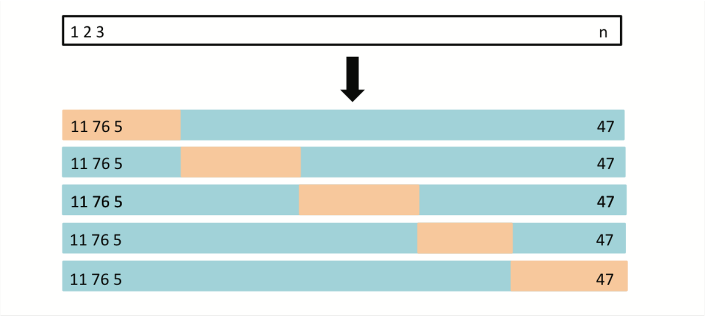
k-fold cross-validation

- split the orginal data in *k* groups
- each group (in turn) is a validation set and the rest is the training set



k-fold cross-validation

- split the orginal data in k groups
- each group (in turn) is a validation set and the rest is the training set



Comparison to leave-one-out

- leave-one-out $=$ n-fold cross-validation
- [+] only k models of size $n-k$ to fit

How should k be chosen? Usually $k = 5$ or $10\ldots$

Cross-validation in R using caret

Leave-one out

```r
train_control <- trainControl(method = "LOOCV")
```

k-fold cross-validation (here with $k = 10$)

```r
train_control <- trainControl(method = "cv",
                              number = 10)
```

k fold CV with K-NN

(Watch your k! Sorry about the notation with k and K...)

```r
model_knn <- train(form = chd ~ .,
                   data = training,
                   trControl = train_control,
                   # tuneGrid = expand.grid(k = c(1:10)*4),
                   method = "knn")
model_knn
```

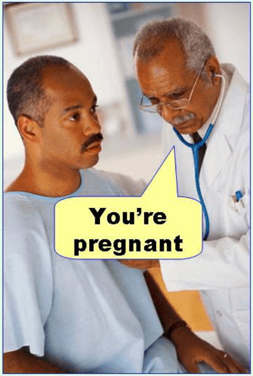
```
k-Nearest Neighbors

347 samples
  9 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 313, 313, 312, 312, 312, 312, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.6254622  0.1087459
  7  0.6457143  0.1406069
  9  0.6457983  0.1334024

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
```

ROC curves

True and false positives

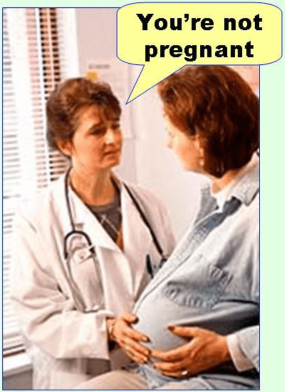
Type I error

(false positive)



Type II error

(false negative)



True and false positive rates

The output of a binary classification method is $\hat{Y}_i = \hat{f}(X_i) \in \{0, 1\}$.

It may be interpreted in terms of true and false positives:

- False Positive Rate: FPR$= \frac{\sum_{i=1}^n \mathbb{1}_{\hat{Y}_i=1 \& Y_i=0}}{\sum_{i=1}^n \mathbb{1}_{Y_i=0}}$
- True Positive Rate: TPR$= \frac{\sum_{i=1}^n \mathbb{1}_{\hat{Y}_i=1 \& Y_i=1}}{\sum_{i=1}^n \mathbb{1}_{Y_i=1}}$

True and false positive rates

The output of a binary classification method is $\hat{Y}_i = \hat{f}(X_i) \in \{0, 1\}$.

It may be interpreted in terms of true and false positives:

- False Positive Rate: FPR$= \frac{\sum_{i=1}^n \mathbb{1}_{\hat{Y}_i=1 \& Y_i=0}}{\sum_{i=1}^n \mathbb{1}_{Y_i=0}}$
- True Positive Rate: TPR$= \frac{\sum_{i=1}^n \mathbb{1}_{\hat{Y}_i=1 \& Y_i=1}}{\sum_{i=1}^n \mathbb{1}_{Y_i=1}}$

In fact $\hat{Y}_i$ is generally obtained by thesholding a probability:

$$\hat{Y}_i = 1 \quad \iff P(Y = 1 | X = X_i) > 0.5$$

True and false positive rates

The output of a binary classification method is $\hat{Y}_i = \hat{f}(X_i) \in \{0, 1\}$.

It may be interpreted in terms of true and false positives:

- False Positive Rate: FPR$= \frac{\sum_{i=1}^n \mathbb{1}_{\hat{Y}_i=1 \& Y_i=0}}{\sum_{i=1}^n \mathbb{1}_{Y_i=0}}$
- True Positive Rate: TPR$= \frac{\sum_{i=1}^n \mathbb{1}_{\hat{Y}_i=1 \& Y_i=1}}{\sum_{i=1}^n \mathbb{1}_{Y_i=1}}$

In fact $\hat{Y}_i$ is generally obtained by thesholding a probability:

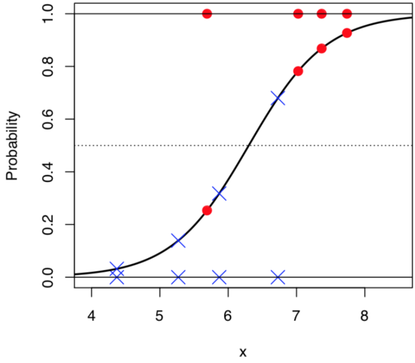
$$\hat{Y}_i = 1 \quad \iff P(Y = 1|X = X_i) > 0.5$$

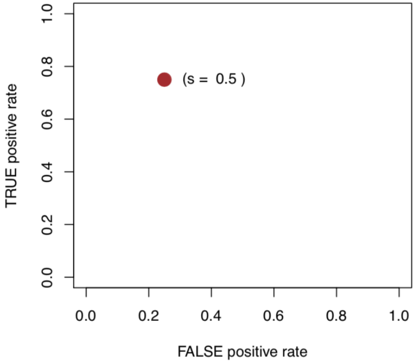
Other values of the threshold would yield different $\hat{Y}_i$ (and FPR, TPR):

$$\hat{Y}_i(s) = 1 \quad \iff P(Y = 1|X = X_i) > s$$

Receiver Operating Characteristic (ROC) curve

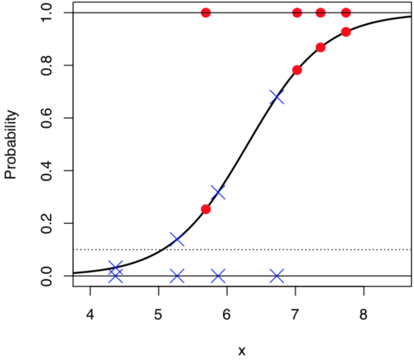
ROC curve: $\{(\mathrm{TPR}(s), \mathrm{FPR}(s)), s \in [0, 1]\}$

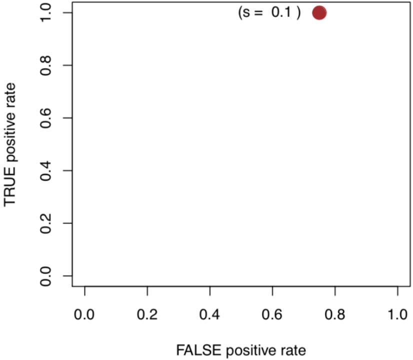




Receiver Operating Characteristic (ROC) curve

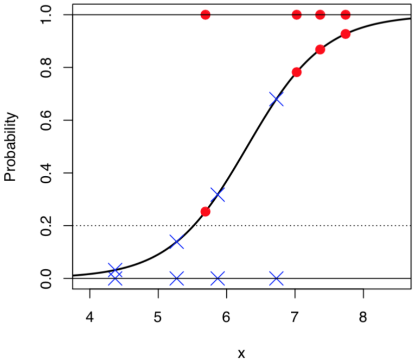
ROC curve: $\{(\mathrm{TPR}(s), \mathrm{FPR}(s)), s \in [0,1]\}$

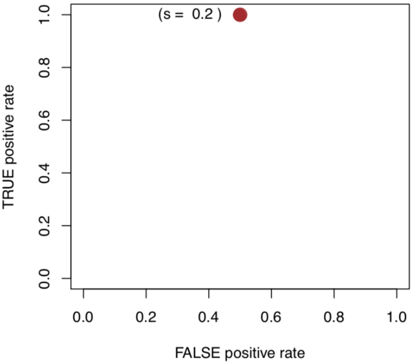




Receiver Operating Characteristic (ROC) curve

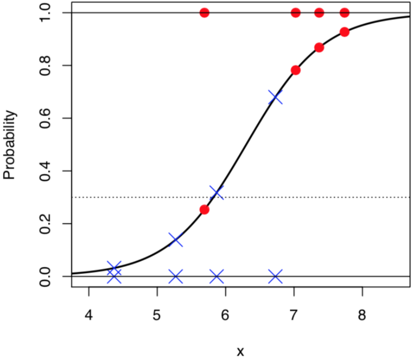
ROC curve: $\{(\mathrm{TPR}(s), \mathrm{FPR}(s)), s \in [0,1]\}$

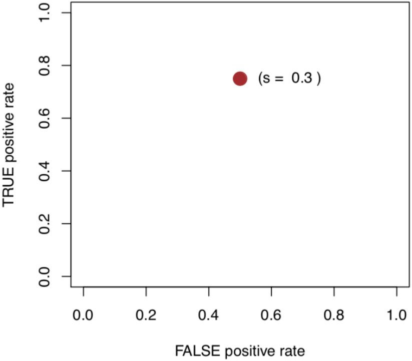




Receiver Operating Characteristic (ROC) curve

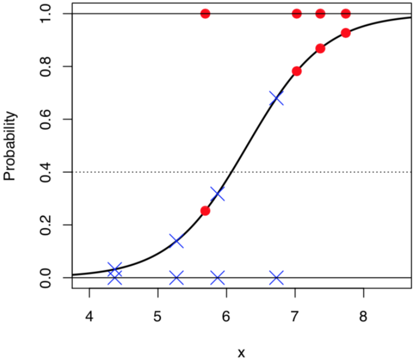
ROC curve: $\{(\mathrm{TPR}(s), \mathrm{FPR}(s)), s \in [0,1]\}$

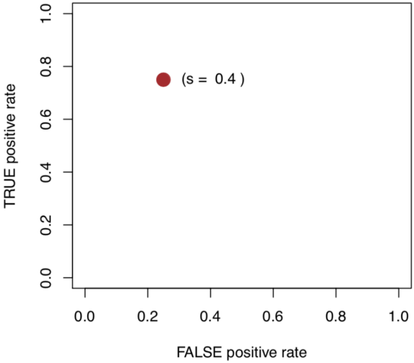




Receiver Operating Characteristic (ROC) curve

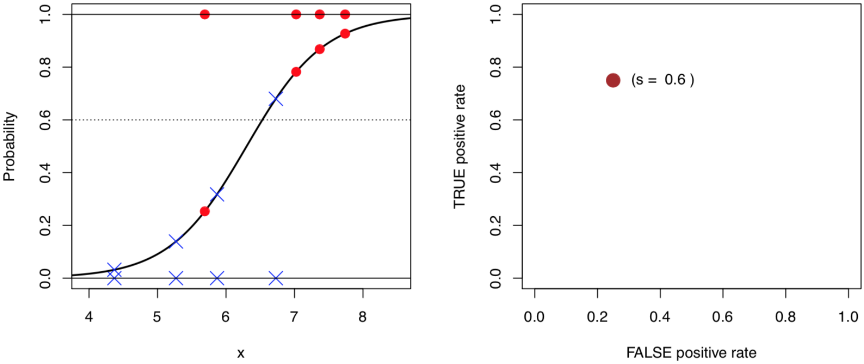
ROC curve: $\{(\mathrm{TPR}(s), \mathrm{FPR}(s)), s \in [0, 1]\}$





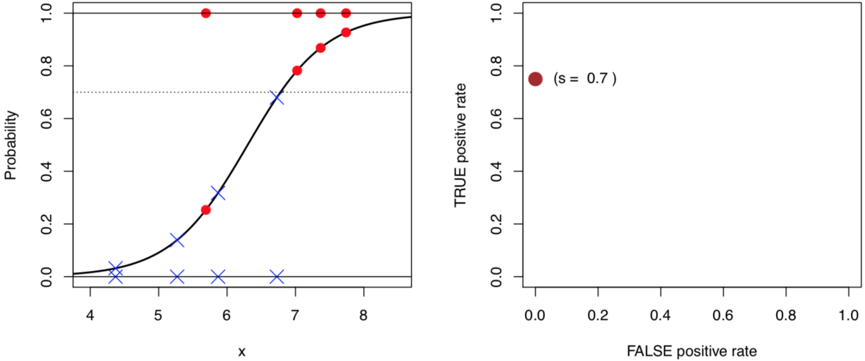
Receiver Operating Characteristic (ROC) curve

ROC curve: $\{(\mathrm{TPR}(s), \mathrm{FPR}(s)), s \in [0,1]\}$



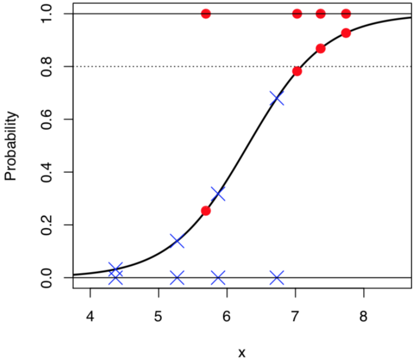
Receiver Operating Characteristic (ROC) curve

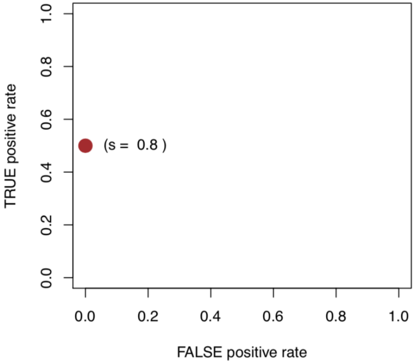
ROC curve: $\{(\mathrm{TPR}(s), \mathrm{FPR}(s)), s \in [0, 1]\}$



Receiver Operating Characteristic (ROC) curve

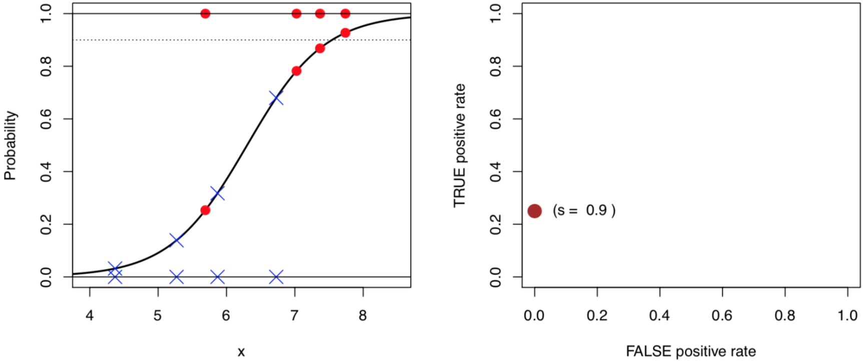
ROC curve: $\{(\mathrm{TPR}(s), \mathrm{FPR}(s)), s \in [0,1]\}$





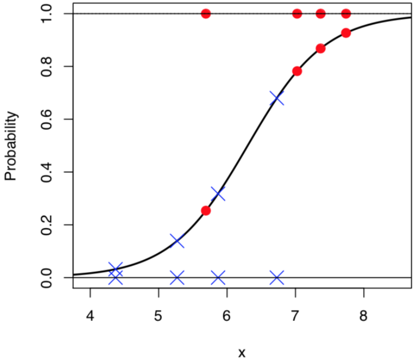
Receiver Operating Characteristic (ROC) curve

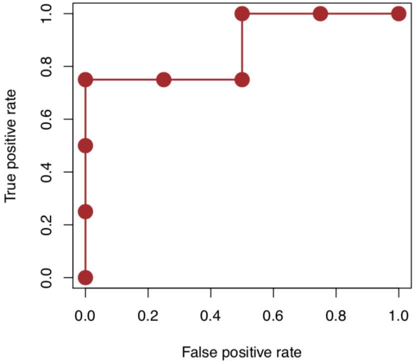
ROC curve: $\{(\mathrm{TPR}(s), \mathrm{FPR}(s)), s \in [0,1]\}$



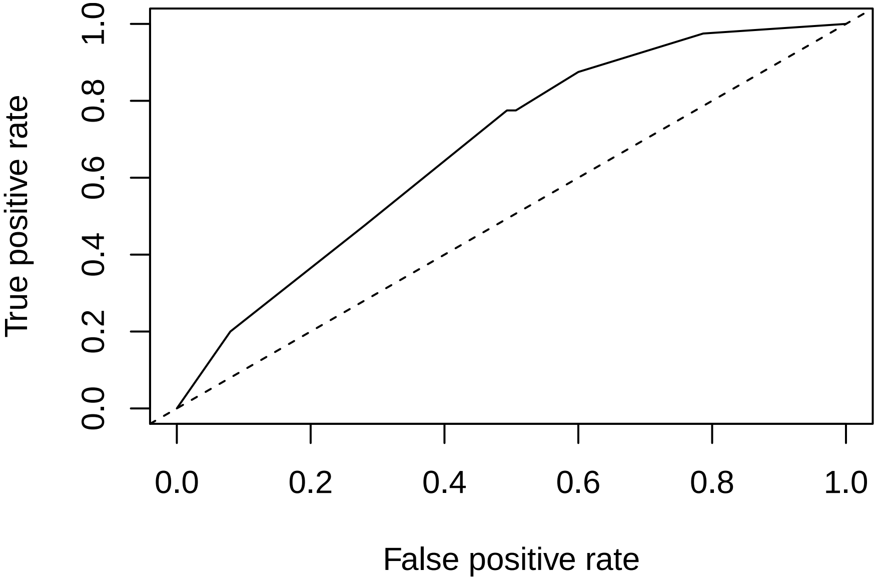
Receiver Operating Characteristic (ROC) curve

ROC curve: $\{(\mathrm{TPR}(s), \mathrm{FPR}(s)), s \in [0, 1]\}$

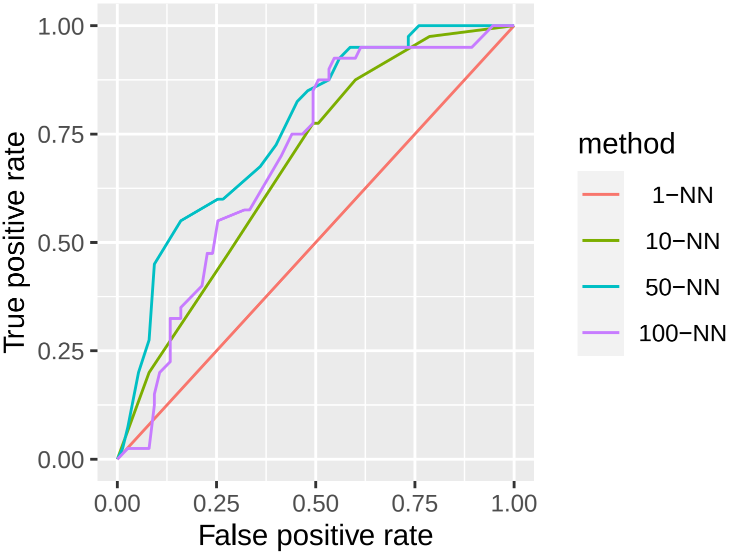




ROC curve with R



Comparing hold-out performance for varying k



Classification and Regression Trees

Background

Reference

Breiman, Friedman, Olshen, Stone(1984). Classification and regression trees. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software.

Idea

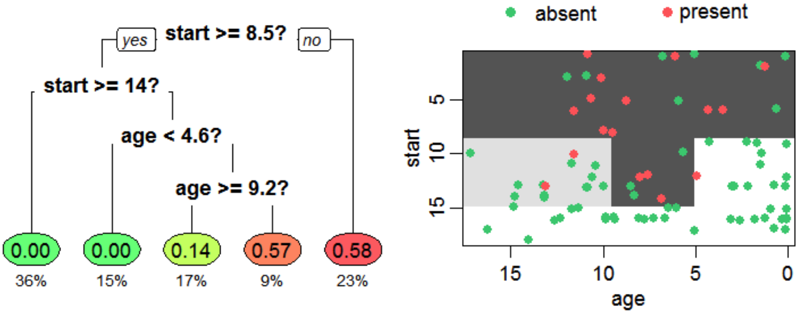
Segmenting the predictor space (X) into a number of simple regions according to the value of Y

Features

- can be used either for regression or for classification
- accounts for non-linearities (non-parametric)
- easy to represent and interpret
- easy to mix quantitative and qualitative predictors

Example (from Wikipedia)

- Y: indicator of presence of kyphosis after spinal surgery
- X_1: age of the patient
- X_2: vertebra at which surgery was started



Left: the colored leaves show the probability of kyphosis after spinal surgery, and percentage of patients in the leaf. Right: the background color indicates the empirical probability of kyphosis after surgery.

Image source: Stephen Milborrow, CC BY-SA 4.0

CART: general idea

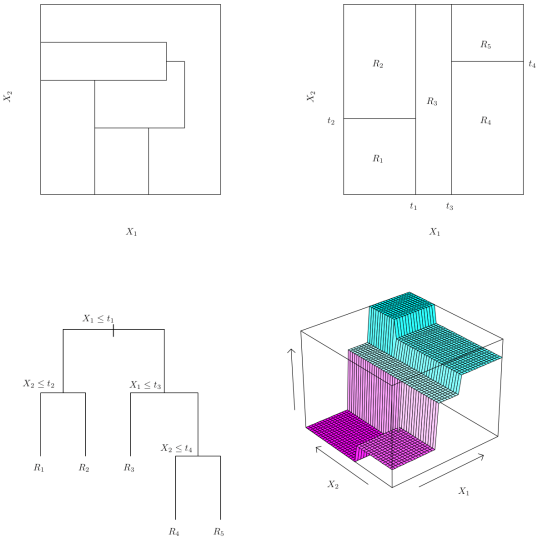
Tree-building process

- Divide the input into a *partition*, i.e. J distinct and non-overlapping regions, R_1, R_2, ..., R_J
- In practice: use *recursive binary splitting* to obtain the partition
- Caveat: greedy approach (too complex to find the best partition)

Prediction

If observation i falls into R_j, predict Y_i as the most commonly occurring class of training observations in R_j (or mean response value in R_j in the case of a regression tree).

CART is greedy



The top left tree cannot be obtained by CART!

Recursive binary splitting

Find best split

- candidate splits: $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ for all possible j and s
- best split defined as maximizing either the Gini index G (a.k.a. node purity) or the deviance D (or cross-entropy):

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \qquad D = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$$

where $\hat{p}_{mk}$ is the proportion of training observations from class k in region m.

Recursion

- split one of the previously identified regions
- stopping criterion: do not split nodes with less than a given number of observations

Pruning

The need for pruning

Influence of the size of the tree/of the partition:

- too coarse partition (e.g. one single leave): unspecific nodes, **underfitting**
- too fine partition (e.g. one observation per leave): too specific nodes, **overfitting** (poor preductions on a test set)

⇒ need to control the complexity of the tree, measured by its size $|T|$

Cost complexity pruning

1. Grow a large tree T_0
2. For a given value of define the "best subtree" with complexity parameter α as the subtree $T(\alpha) \subset T_0$ minimizing $D(T) + \alpha|T|$
3. Choose α by *cross-validation*

Note on the form of the optimization criterion

$$\mathrm{Min}_T\ D(T) + \alpha|T|$$

This is an instance of a *penalized criterion* incorporating both

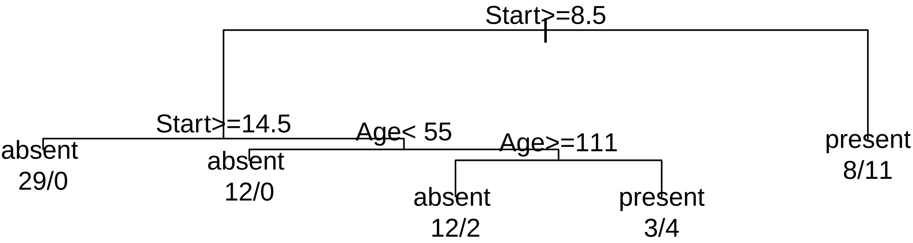
- $D(T)$: quantifies the fit (or adjustment) of the model to the observations
- $|T|$: quantifies the complexity of the model

See other model selection criterion (e.g. in regression)

- AIC, BIC, Cp
- lasso, ridge
- ...

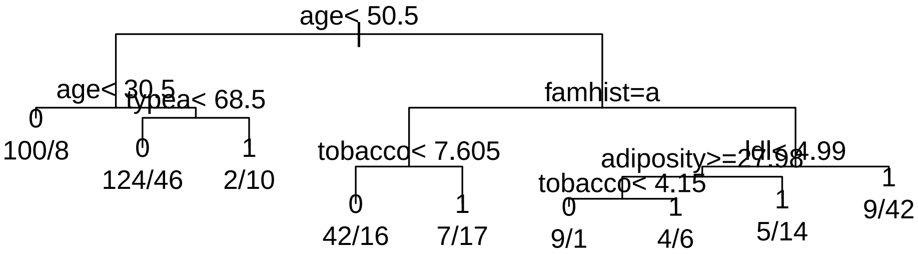
CART with R: kyphosis data

```r
library("rpart")
fit <- rpart(Kyphosis ~ Age + Number + Start, data = kyphosis)
par(xpd = NA)
plot(fit)
text(fit, use.n = TRUE)
```



CART with R: SAheart

```r
library("rpart")
fit <- rpart(chd ~ ., data = SAheart)
par(xpd = NA)
plot(fit)
text(fit, use.n = TRUE)
```



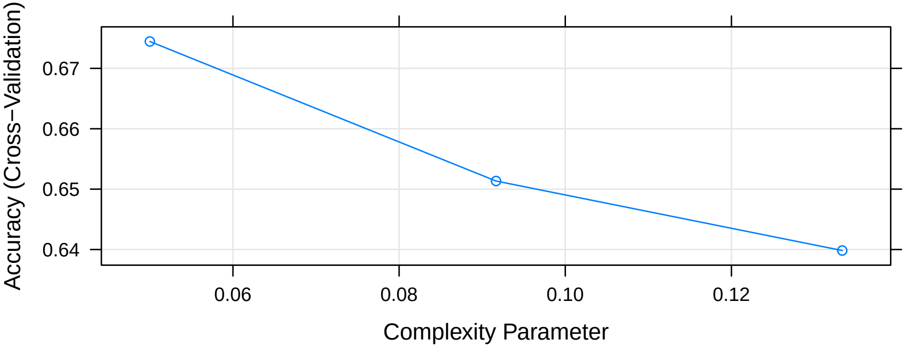
CART with R via caret: model

```
model_rpart <- train(chd ~., data = training,
                     method = "rpart",
                     trControl = train_control)
model_rpart
```

```
CART

347 samples
  9 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 313, 312, 312, 312, 312, 312, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.05000000  0.6744538  0.24929648
  0.09166667  0.6513445  0.17398255
  0.13333333  0.6398319  0.02489056

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.05.
```

CART with R via caret: parameters

```r
plot(model_rpart)
```



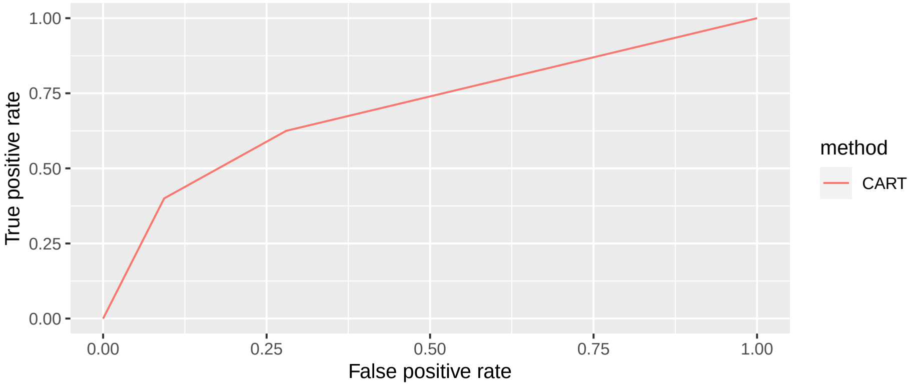
CART with R via caret: confusion matrix

```r
pred <- predict(model_rpart, newdata = testing)
table(pred, testing$chd)
```

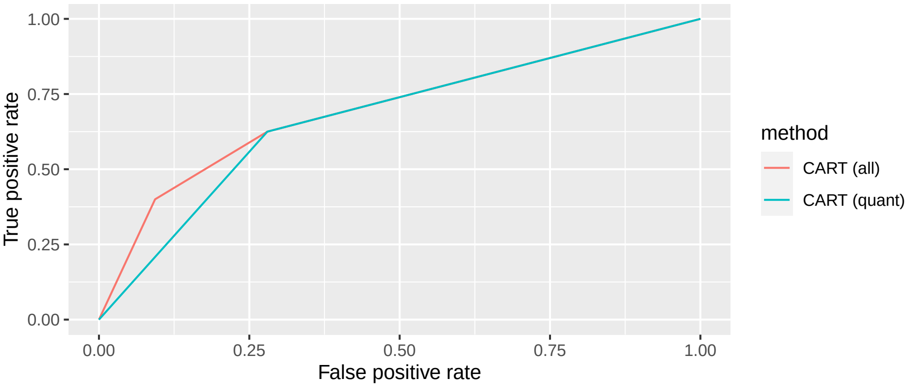
```
pred  0  1
   0 68 24
   1  7 16
```

CART with R via caret: ROC curve

```r
prob <- predict(model_rpart, newdata = testing, type = "prob")[["1"]
roc_rpart <- tidy_perf(prob, testing$chd, method = "CART")
ggplot(roc_rpart,  aes(x = `False positive rate`,
                    y = `True positive rate`,
                    color = method)) +
    geom_line()
```



CART with R via caret: ROC curves



Conclusions on CART

Why CART is widely used

- applies to both classification and regression
- trees are easy to visualize and interpret
- accomodate both quanti and quali variables, and missing values
- efficient algorithm to grow and prune trees

Some limitations

- instability: little robustness to the learning sample
- poor prediction accuracy

$\Rightarrow$ hence the need for *aggregating* trees!

Random Forests

From trees to forests

References

- R. Genuer, J.-M. Poggi. *Les forêts aléatoires avec R*. Presses Universitaires de Rennes 2019
- R. Genuer, J.-M. Poggi. *Random Forests with R*. Springer 2020

Idea

- Fix the instability of CART by growing multiple trees and *aggregating* the results.
- Statistical ingredient: Bootstrap Aggregation or *bagging*

Bagging

General goal

Reduce the variance of a statistical learning method

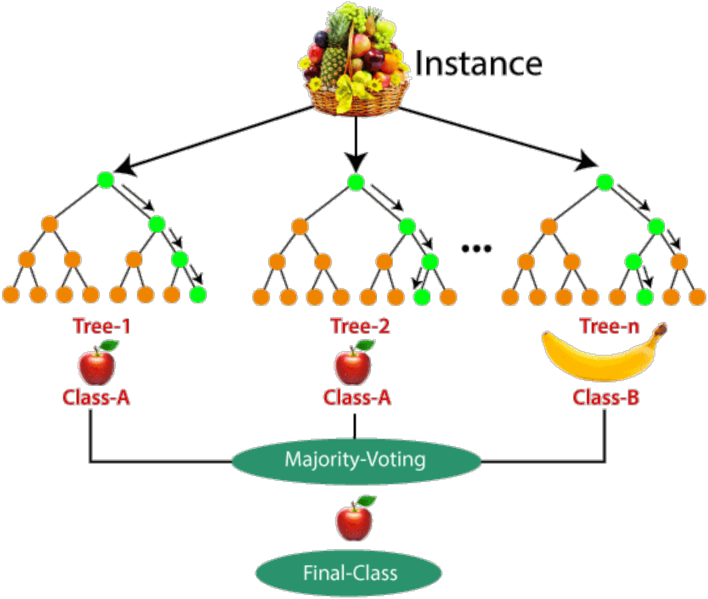
Hint

If $(Z_1, \dots, Z_n)$ are n *independent* observations with common variance denoted by σ^2, then the variance of their mean $\bar{Z} = n^{-1} \sum_{i=1}^{n} Z_i$ is σ^2/n.

Algorithm

1. Grow B trees, each on a *bootstrap sample* of the data (sample n observations with replacement)
2. Prediction on a test observation is obtained by a *majority vote* of the B trees

Illustration of Bagging



Out-of-Bag (OOB) Error Estimation

Idea

Make use of observations i not included in a particular sample (but for which we have both X_i and Y_i):

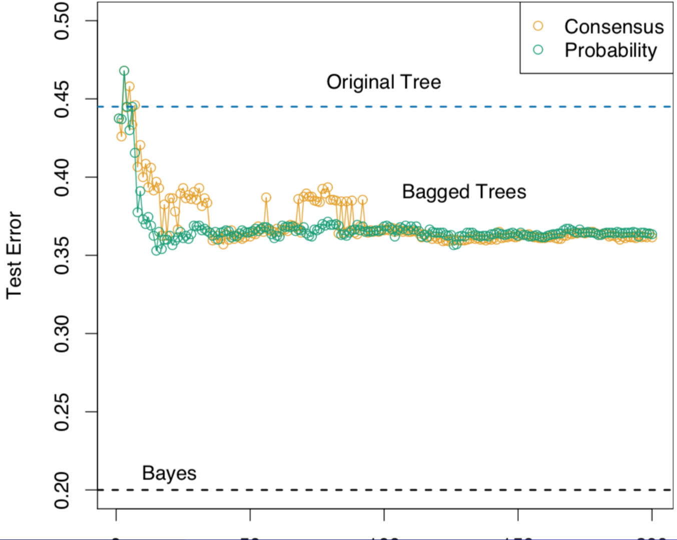
- for each i average individual predictions on OOB data
- compare to Y_i

Comment

- On average, $\approx 1/3$ of observations are not included in a particular bootstrap sample

Bagging: simulated example

Source: Elements of Statistical Learning (chapter 10)



Random forests

Limitation of bagged trees

bootstrap samples are correlated, so their aggregation may not reduce the variance as much as we hope.

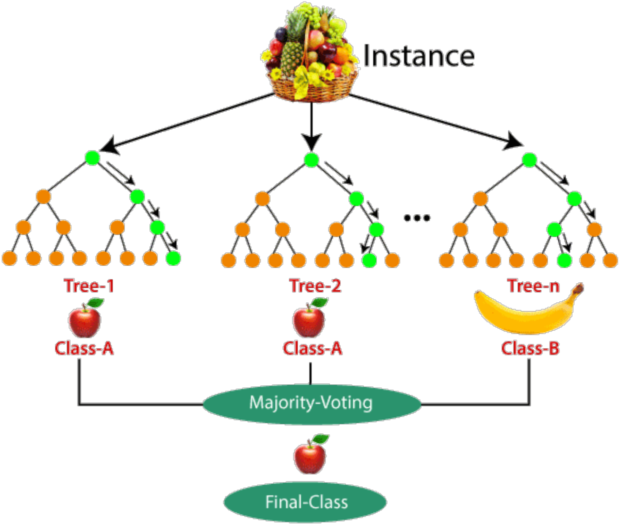
Difference between bagged trees and random forests

- candidates at each split restricted to a *random selection* of $m < p$ variables
- typically: $m \approx \sqrt{p}$

Parameters

- B (`n_tree`): number of trees (bootstrap samples) in the forest
- m (`m_try`): number of candidate variables for each split

Illustration of random forests



Bagging and random forests: gene expression data

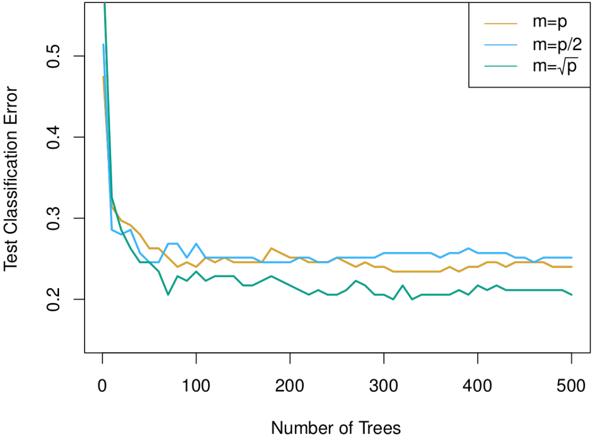
Gene expression data set

- $n = 349$ patients
- $p = 500$ genes (with highest variance)
- $(Y_i)_{1 \leq i \leq n}$: indicator of cancer (vs normal) patient
- $(X_{ij})_{1 \leq i \leq n, 1 \leq j \leq p}$: expression of gene j in patient i

Pipeline

- randomly divide the observations into a training and a test set
- apply random forests to the training set for three different values of the number of splitting variables m:
  - $m = p$ (bagging)
  - $m = p/2$
  - $m = \sqrt{p}$

Bagging and random forests: gene expression data



Error rate of CART: ≈ 0.45

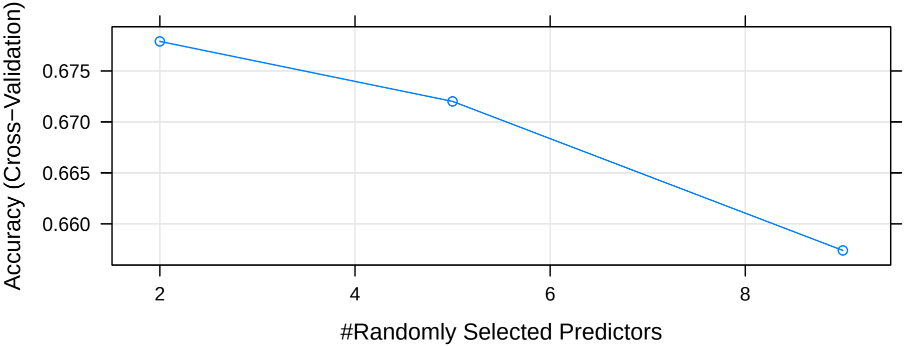
Random Forests with R via caret: model

```r
model_rf <- train(chd ~., data = training,
                  method = "rf",
                  trControl = train_control)
model_rf
```

```
Random Forest

347 samples
  9 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 312, 313, 313, 313, 312, 312, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.6778992  0.2411942
  5     0.6720168  0.2372066
  9     0.6573950  0.2053311

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

Random Forests with R via caret: parameters

```
plot(model_rf)
```



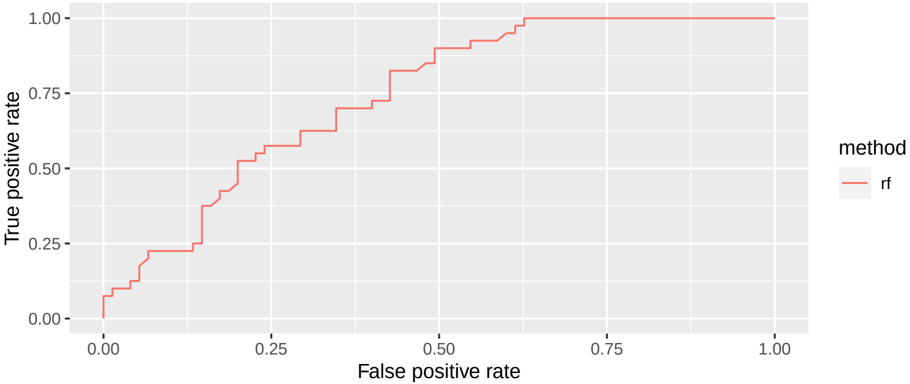
Random Forests with R via caret: confusion matrix

```r
pred <- predict(model_rf, newdata = testing)
table(pred, testing$chd)
```

```
pred  0  1
   0 62 24
   1 13 16
```

Random Forests with R via caret: ROC curve

```
prob <- predict(model_rf, newdata = testing, type = "prob")[["1"]]
roc_rf <- tidy_perf(prob, testing$chd, method = "rf")
ggplot(roc_rf,  aes(x = `False positive rate`,
                    y = `True positive rate`,
                    color = method)) +
    geom_line()
```



Variable importance

Random Forests typically have very good classification performance. However, the obtained (aggregated) predictor is hard to interpret.

Quantifying variable importance

For predictor X_i:

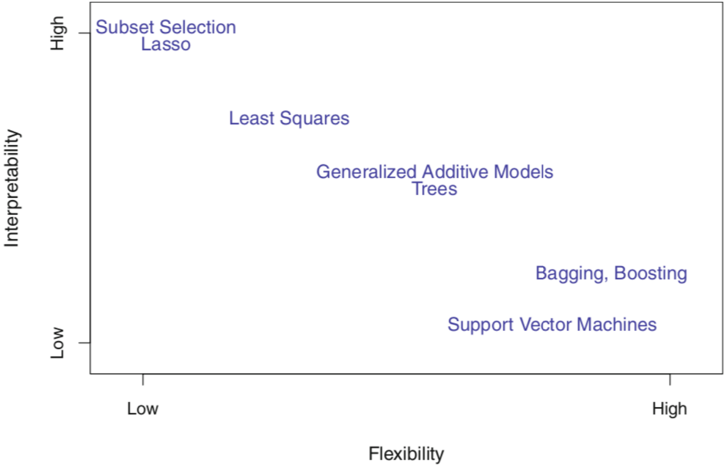
- for each tree: total amount by which the Gini index/deviance is decreased by splits over X_i
- average over all B trees

Summary: CART vs random forests

Method	Simplicity/interpretability	Accuracy
CART	++	–
Random forests/Boosting	-	++

- Not discussed: boosting (R package `gbm`): trees are grown *sequentially* using information from previously grown ones
- Variable importance measures can help improving interpretability for random forests and boosting

Tradeoff b/w model interpretability and flexibility



Adding more competitors: logistic regression

```r
model_logistic <- train(form = chd ~ .,
                        data = training,
                        trControl = train_control,
                        method = "glm",
                        family = "binomial")
prob <- predict(model_logistic, newdata = testing,
                type = "prob")[["1"]]
roc_logistic <- tidy_perf(prob, testing$chd, method = "Logisti
```

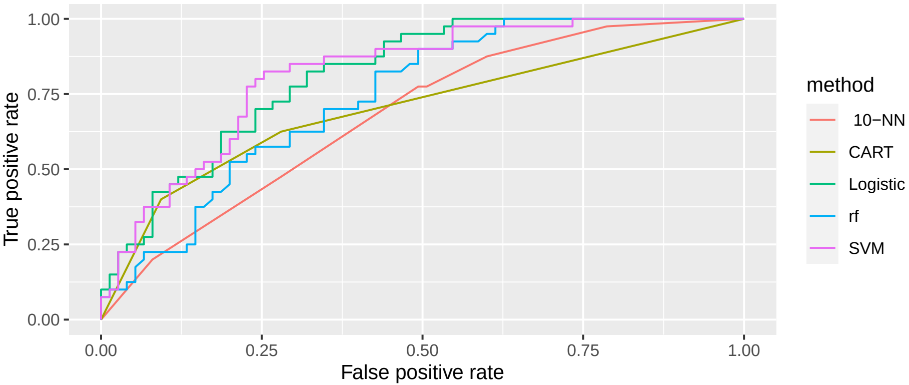
Note: no tuning parameter here!

Adding more competitors: linear SVM

```
model_svm <- train(form = chd ~ .,
                   data = training,
                   trControl = train_control,
                   method = "svmLinear2",
                   probability = TRUE)
prob <- predict(model_svm, newdata = testing,
                type = "prob")[["1"]]
roc_svm <- tidy_perf(prob, testing$chd, method = "SVM")
```

SVM = Support Vector Machine

Gobal comparison


True positive rate
False positive rate
method
10−NN
CART
Logistic
rf
SVM