# Using CV for assessment (I)

CROSS VALIDATION

How the learned artifact will behave on unseen data?

More precisely:
How an artifact learned with **this learning technique** will behave on unseen data?

# Using CV for assessment (II)

"This learning technique" $=$ BuildDecisionTree() with $k_{\text{min}} = 10$

1. repeat $k$ times
   1.1 BuildDecisionTree() with $k_{\text{min}} = 10$ on all but one slice
      - $\frac{k-1}{k}$ $\cancel{\frac{9}{10}}n$ observations in each $\mathbf{X}$ passed to BuildDecisionTree()

   1.2 compute classification error on left out slice
2. average computed classification errors

$k$

$\cancel{10}$ invocations of BuildDecisionTree()

# Using CV for assessment (III)

"This learning technique" $=$ BuildDecisionTree() with $k_{\text{min}}$ chosen automatically with a 10-fold CV

For assessing this technique, we do two nested CVs:

1. repeat $k$ times $=5$
   1.1 choose $k_{\text{min}}$ among $m$ values $=20$ with 10-CV (repeat BuildDecisionTree() 10$m$ times) $200$ on all but one slice
      - $\frac{k-1}{k}\frac{9}{10}n$ observations in each $\mathbf{X}$ passed to BuildDecisionTree()!
   1.2 compute classification error on left out slice
      - usually, a new tree is built on $\frac{k}{k-1}n$ observations
2. average computed classification errors

$(\overset{m}{10} + 1)k$ invocations of BuildDecisionTree()

# Using CV for assessment: “cheating”

“This learning technique” $=$ $\textsc{BuildDecisionTree}()$ with $k_{\text{min}}$ chosen automatically with a 10-fold CV

Using just one CV is cheating (cherry picking)!

- $k_{\text{min}}$ is chosen exactly to minimize error on the full dataset
- conceptually, this way of “fitting” $k_{\text{min}}$ is similar to the way we build the tree

Subsection 1

## Regression trees

# Regression with trees

Trees can be used for regression, instead of classification.

decision tree vs. regression tree

## Tree building: decision $\to$ regression

$\in \mathbb{R}^m$

**function** $\textsc{BuildDecisionTree}(\mathbf{X}, \mathbf{y})$
    **if** $\textsc{ShouldStop}(\mathbf{y})$ **then**
        $\hat{y} \leftarrow$ most common class in $\mathbf{y}$
        **return** new terminal node with $\hat{y}$
    **else**
        $(i, t) \leftarrow \textsc{BestBranch}(\mathbf{X}, \mathbf{y})$
        $n \leftarrow$ new branch node with $(i, t)$
        append child $\textsc{BuildDecisionTree}(\mathbf{X}|_{\mathbf{x}_i < t}, \mathbf{y}|_{\mathbf{x}_i < t})$ to $n$
        append child $\textsc{BuildDecisionTree}(\mathbf{X}|_{\mathbf{x}_i \geq t}, \mathbf{y}|_{\mathbf{x}_i \geq t})$ to $n$
        **return** $n$
    **end if**
**end function**

**Q:** what should we change?

# Tree building: decision $\rightarrow$ regression

**function** $\textsc{BuildDecisionTree}(\mathbf{X}, \mathbf{y})$
  **if** $\textsc{ShouldStop}(\mathbf{y})$ **then**
    $\hat{y} \leftarrow \bar{y}$ $\triangleright$ mean $\mathbf{y}$
    **return** new terminal node with $\hat{y}$
  **else**
    $(i, t) \leftarrow \textsc{BestBranch}(\mathbf{X}, \mathbf{y})$
    $n \leftarrow$ new branch node with $(i, t)$
    append child $\textsc{BuildDecisionTree}(\mathbf{X}|_{\mathbf{x}_i < t}, \mathbf{y}|_{\mathbf{x}_i < t})$ to $n$
    append child $\textsc{BuildDecisionTree}(\mathbf{X}|_{\mathbf{x}_i \geq t}, \mathbf{y}|_{\mathbf{x}_i \geq t})$ to $n$
    **return** $n$
  **end if**
**end function**

**Q:** what should we change?

# Best branch

**function** $\textsc{BestBranch}(\mathbf{X}, \mathbf{y})$
  $(i^\star, t^\star) \leftarrow \arg\min_{i,t} E(\mathbf{y}|_{\mathbf{x}_i \geq t}) + E(\mathbf{y}|_{\mathbf{x}_i < t})$
  **return** $(i^\star, t^\star)$
**end function**

**Q:** what should we change?

# Best branch

DIFFERENT

**function** BESTBRANCH($\mathbf{X}, \mathbf{y}$)

  $(i^\star, t^\star) \leftarrow \arg\min_{i,t} \sum_{y_i \in \mathbf{y}|_{\mathbf{x}_i \geq t}} (y_i - \bar{y})^2 + \sum_{y_i \in \mathbf{y}|_{\mathbf{x}_i < t}} (y_i - \bar{y})^2$

  **return** $(i^\star, t^\star)$

**end function**

elements of y

label of decision node

**Q:** what should we change?

Minimize sum of residual sum of squares (RSS) (the two $\bar{y}$ are different)

# Stopping criterion

**function** SHOULDSTOP($\mathbf{y}$)
    **if** $\mathbf{y}$ contains only one class **then**
        **return** true
    **else if** $|\mathbf{y}| < k_{\text{min}}$ **then**
        **return** true
    **else**
        **return** false
    **end if**
**end function**

**Q:** what should we change?

# Stopping criterion

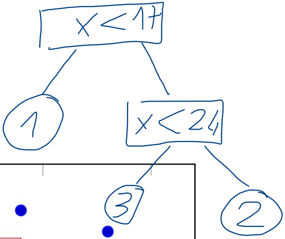
```
function ShouldStop(y)
    if RSS is 0 then
        return true          ⟶ RARE
    else if |y| < k_min then
        return true
    else
        return false
    end if
end function
```

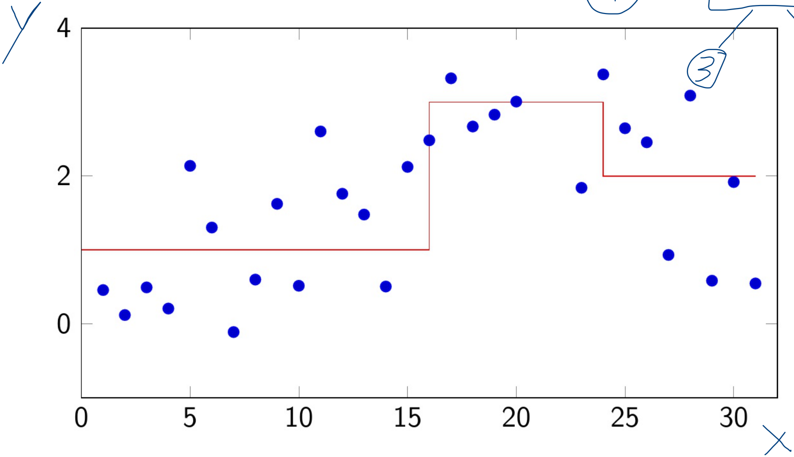
**Q:** what should we change?

# Interpretation

$p = 1$

$n = 29$



# Regression and overfitting

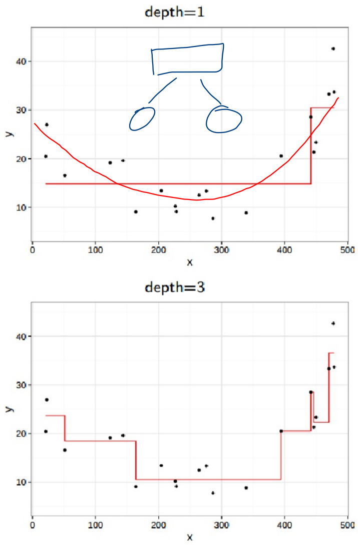



Image from F. Daolio

# Trees in summary

Pros:

- ▲ easily interpretable/explicable
- ▲ learning and regression/classification easily understandable
- ▲ can handle both numeric and categorical values

Cons:

- ▼ not so accurate (**Q:** always?)

# Tree accuracy?

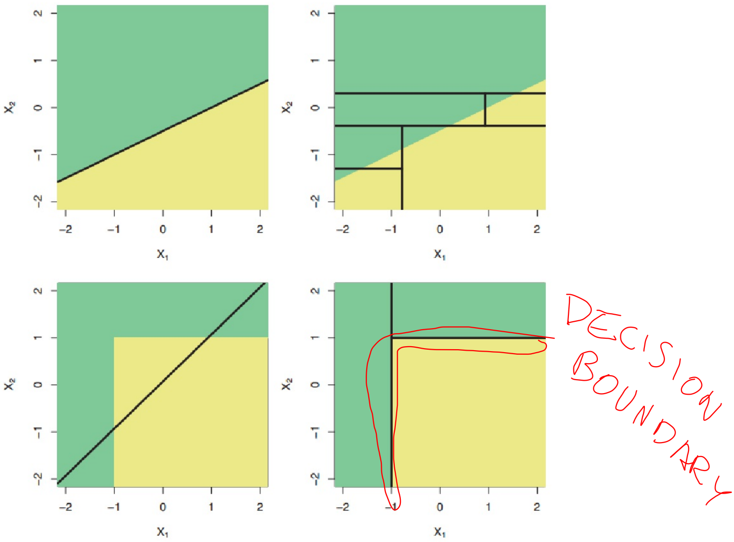



Image from An Introduction to Statistical Learning

# Lab: tree on iris (2 h)

CONFUSION MATRIX

- for each of the 5 variables in iris, predict it with the other 4
- which is the hardest to be predicted? why?

Packages: `tree`

Functions: `tree`, `prune.tree`, `predict.tree(t, type="class")`, `table`

LEARNING

X, y

Species ~ .

formula, d