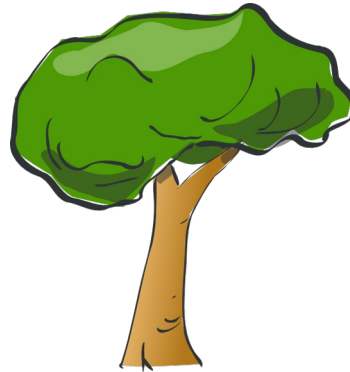*Blockkurs:*
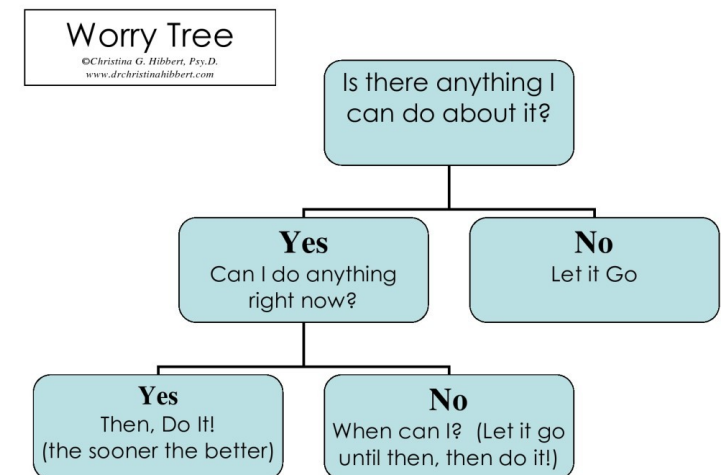**Introduction to Machine Learning for Psychologists**

Decision Trees

Yannick Rothacher

Zürich, FS2025

What is a decision tree?
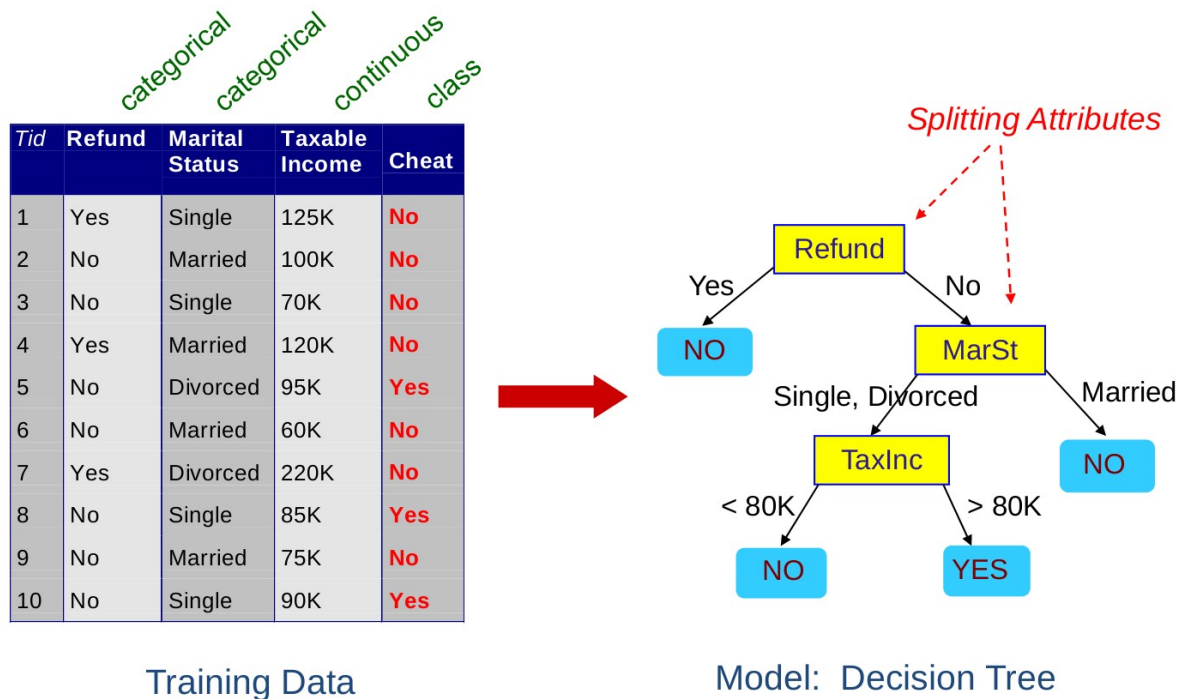
▷ Generally speaking, a **decision tree** is a diagram, which helps us determining a course of action (e.g. "should I worry?")

▷ We can use decision trees for **classification** or **regression**

  ▷ Such trees are the result of what is referred to as **recursive partitioning**

▷ Let's look at an example of a decision tree used for **classification**...
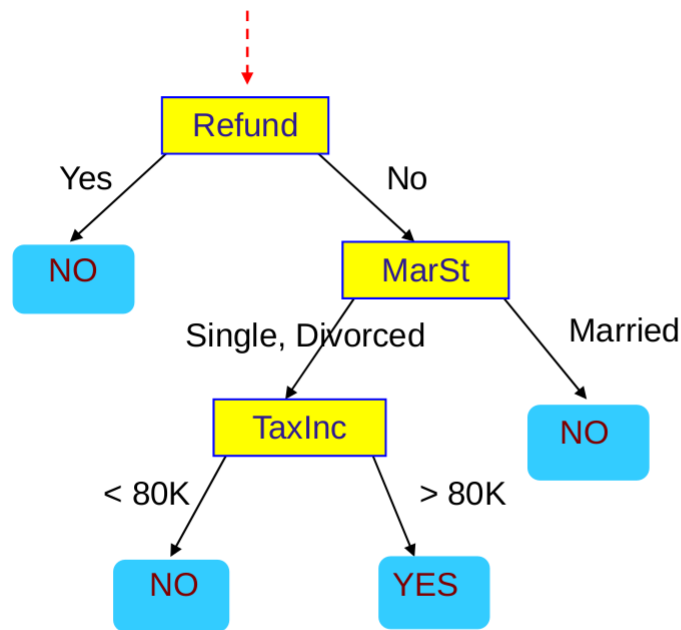
Decision tree – classification

▶ We want to predict whether a person will cheat in his/her tax declaration based on some training data

▶ "**Cheat**" is the target variable (two levels: yes/no)

▶ The generated decision tree uses the three predictor variables to model whether someone will cheat or not:

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data

Model: Decision Tree

Decision tree – classification

▶ Use the generated tree to predict outcome of a **new observation**

▶ Work through the tree for prediction:

Test Data

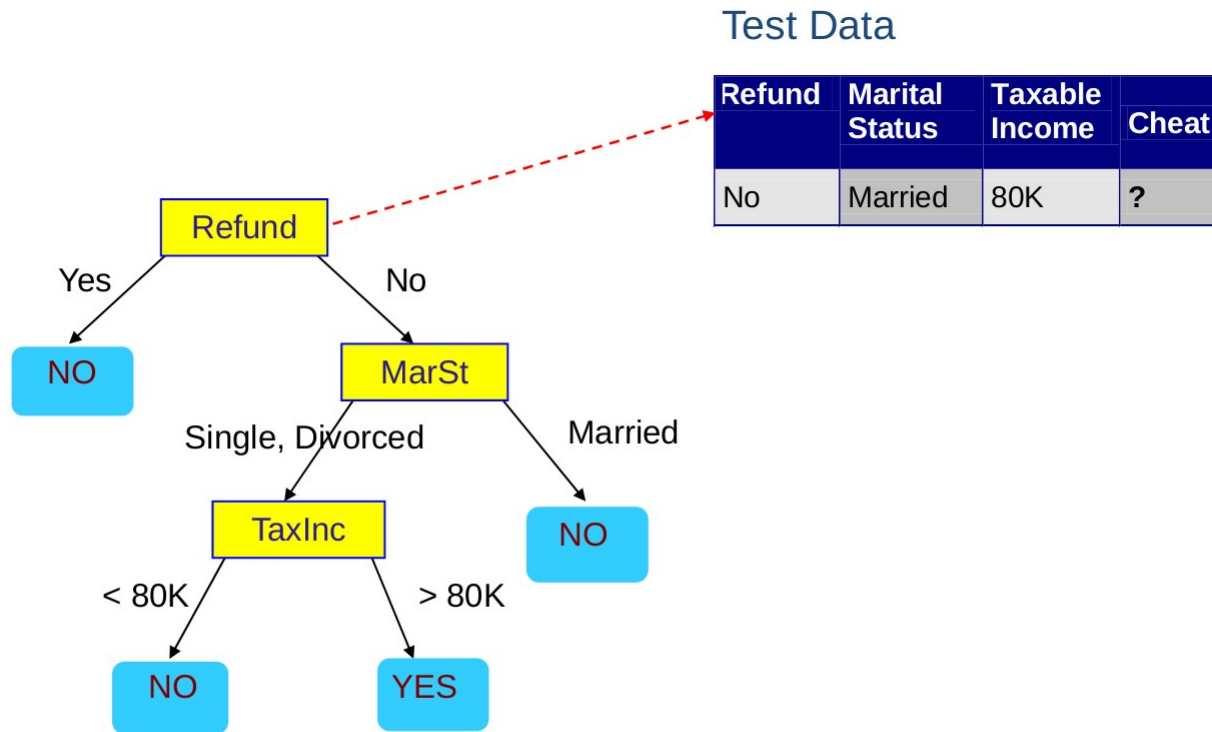| Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Start from the root of tree.

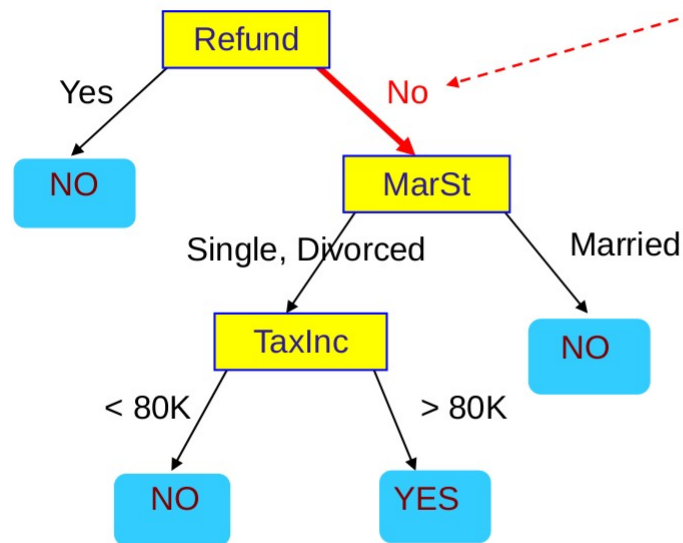**Source**: Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Published by Addison Wesley.

Decision tree – classification

▶ Use the generated tree to predict outcome of a **new observation**

▶ Work through the tree for prediction:

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Source: Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Published by Addison Wesley.

Decision tree – classification

▶ Use the generated tree to predict outcome of a **new observation**

▶ Work through the tree for prediction:

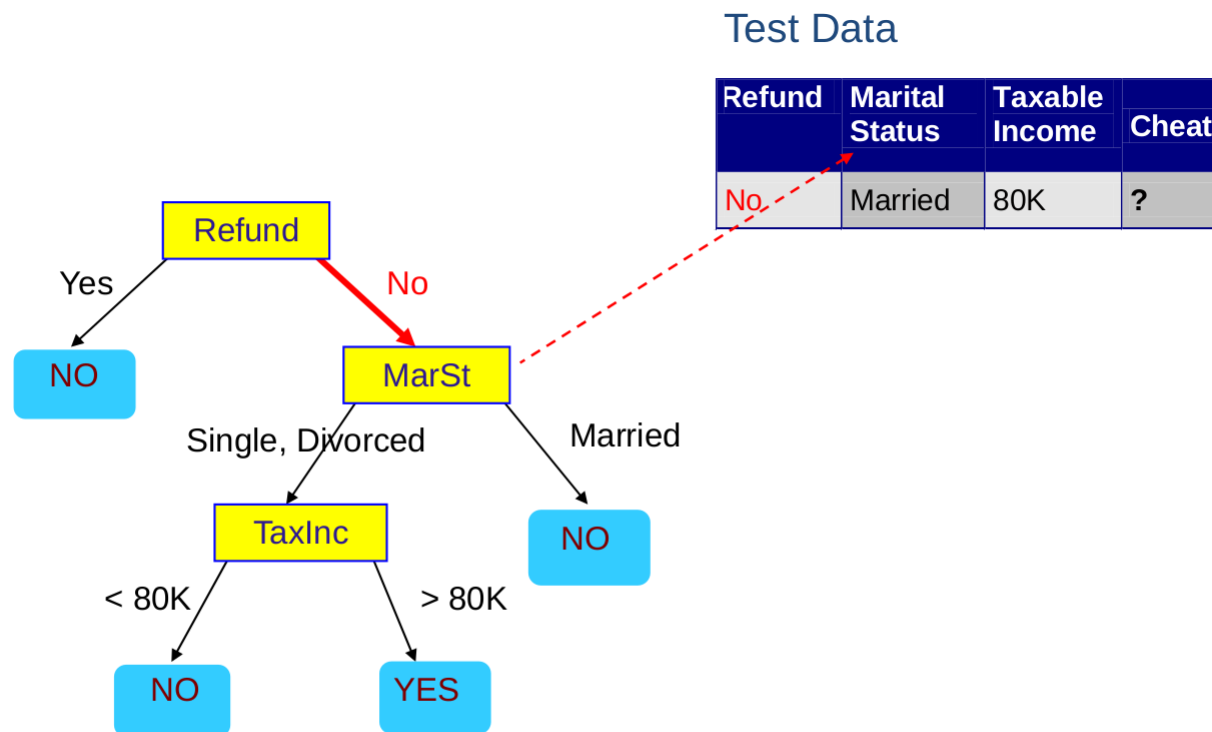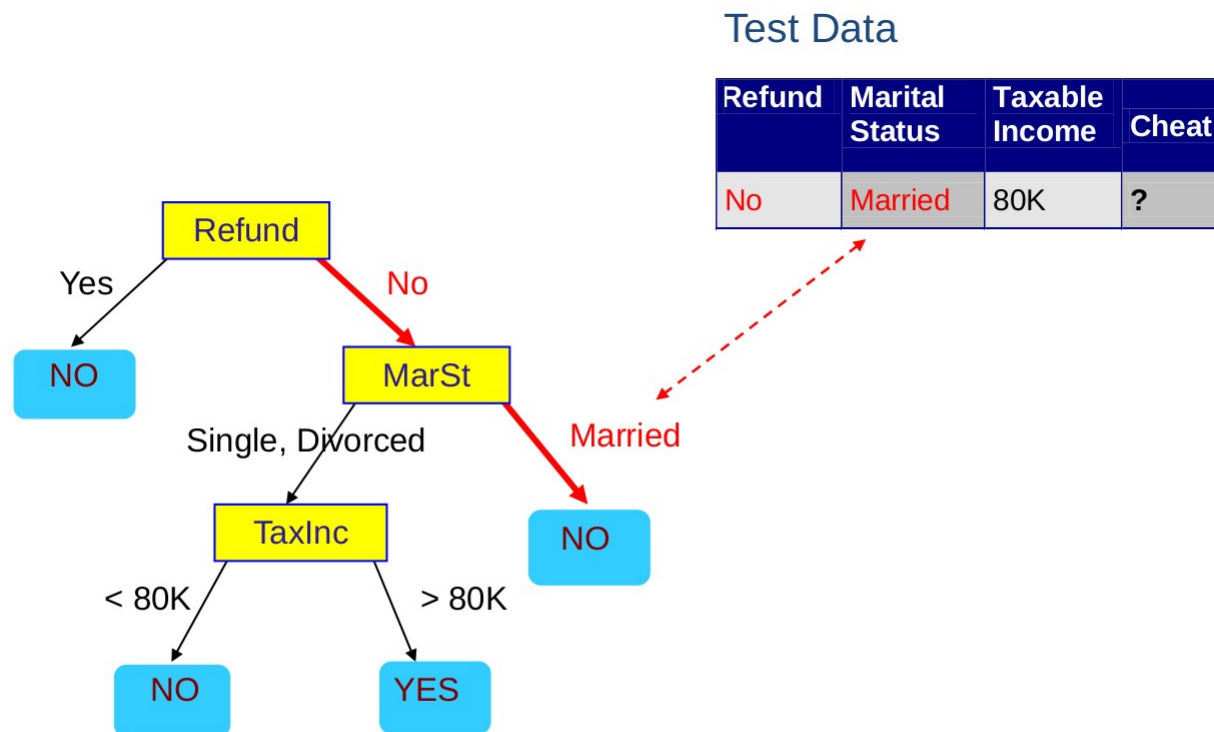Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

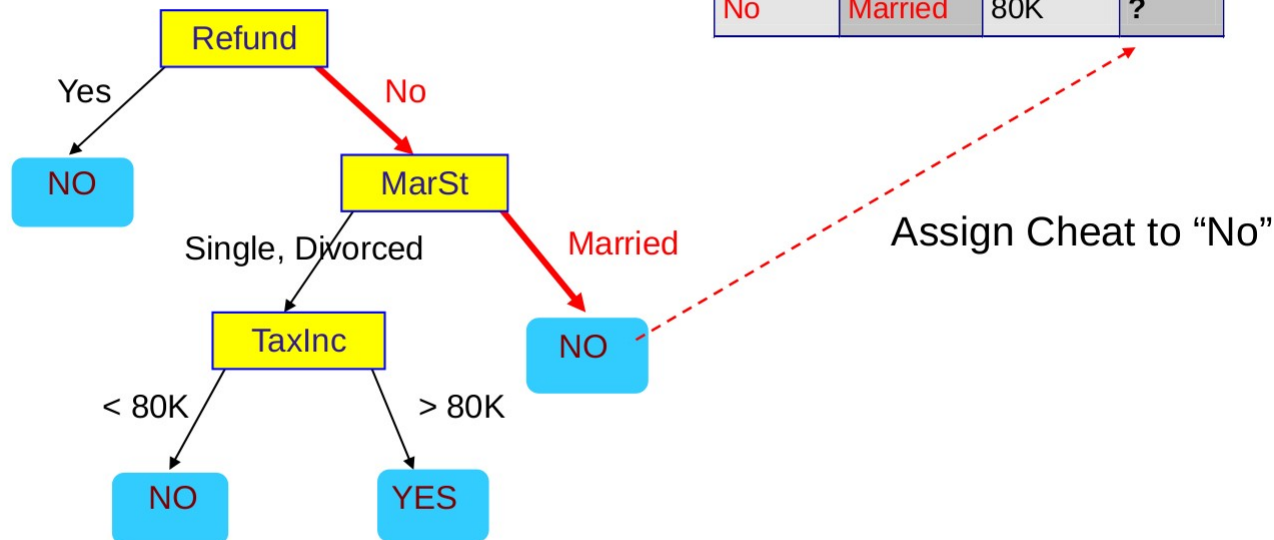**Source**: Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Published by Addison Wesley.

Decision tree – classification

▶ Use the generated tree to predict outcome of a **new observation**

▶ Work through the tree for prediction:

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Decision tree – classification

▶ Use the generated tree to predict outcome of a **new observation**
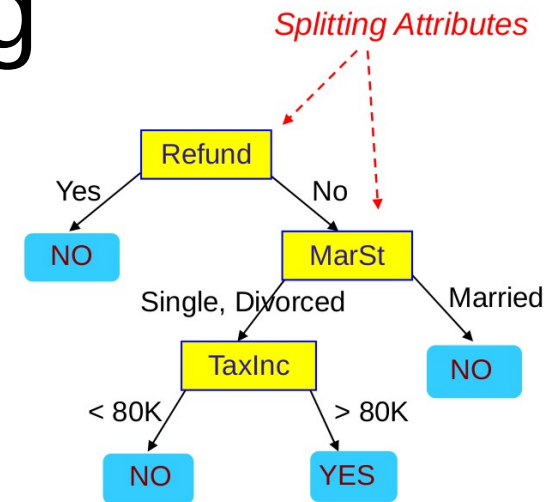
▶ Work through the tree for prediction:

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Source: Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Published by Addison Wesley.

Decision tree – classification

▶ Use the generated tree to predict outcome of a **new observation**

▶ Work through the tree for prediction:

Test Data
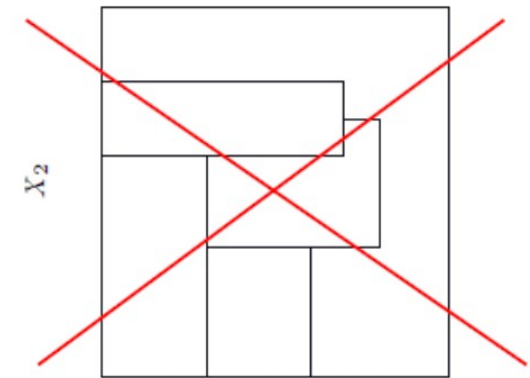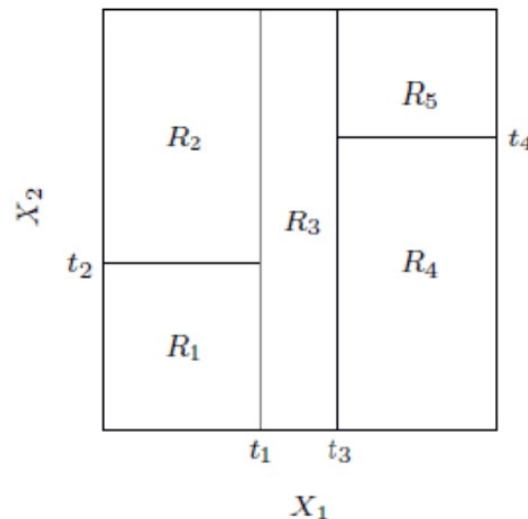
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to "No"

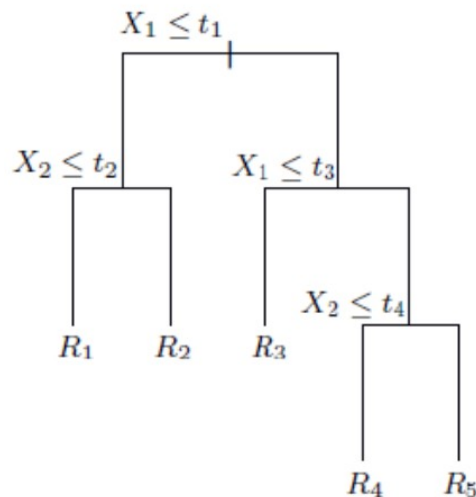**Source**: Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Published by Addison Wesley.

Decision tree – partitioning

- A classic decision tree splits the data at each node based on **one predictor variable**

- For the case of only **two predictor variables** we can visualize this in a 2D-coordinate system:
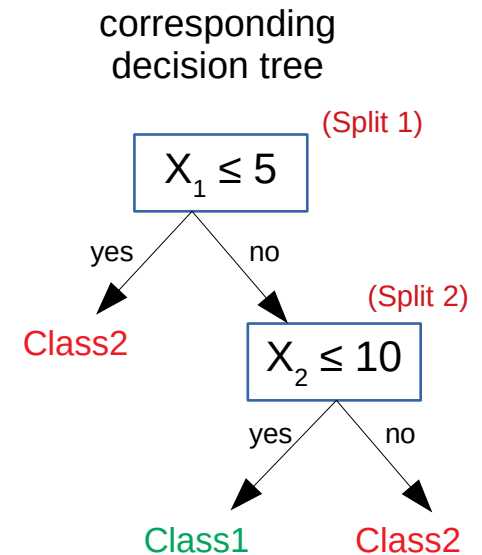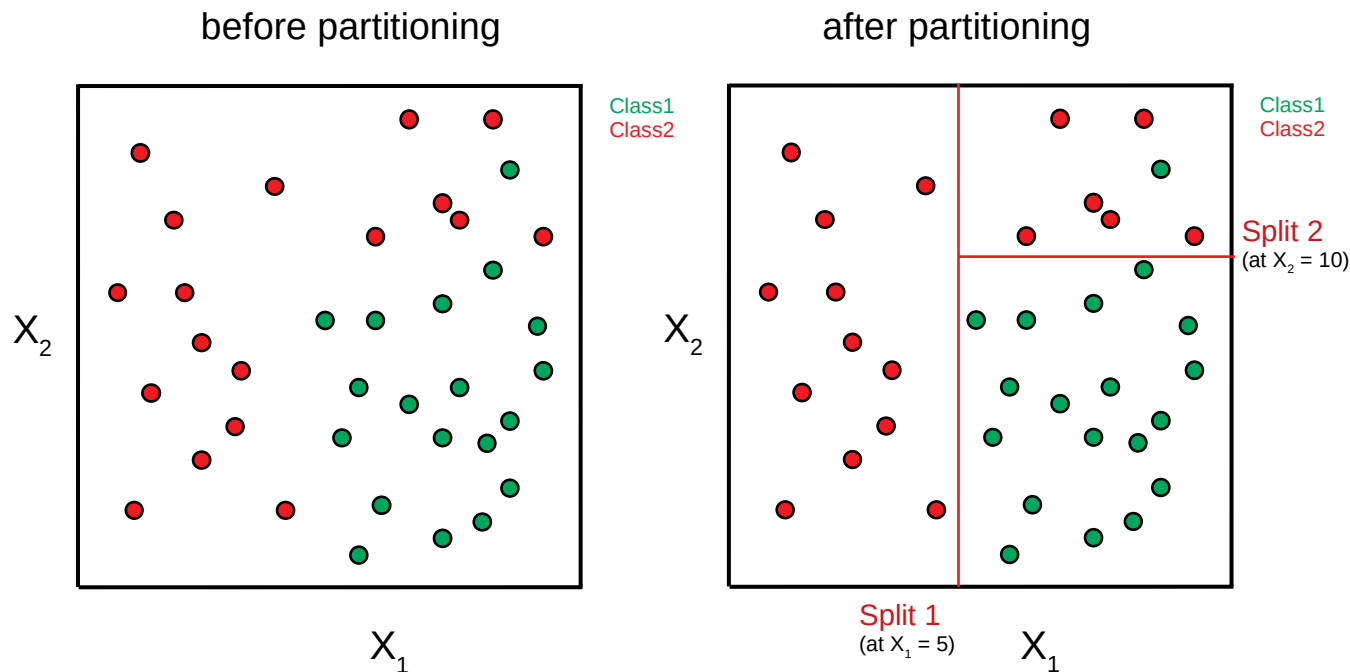
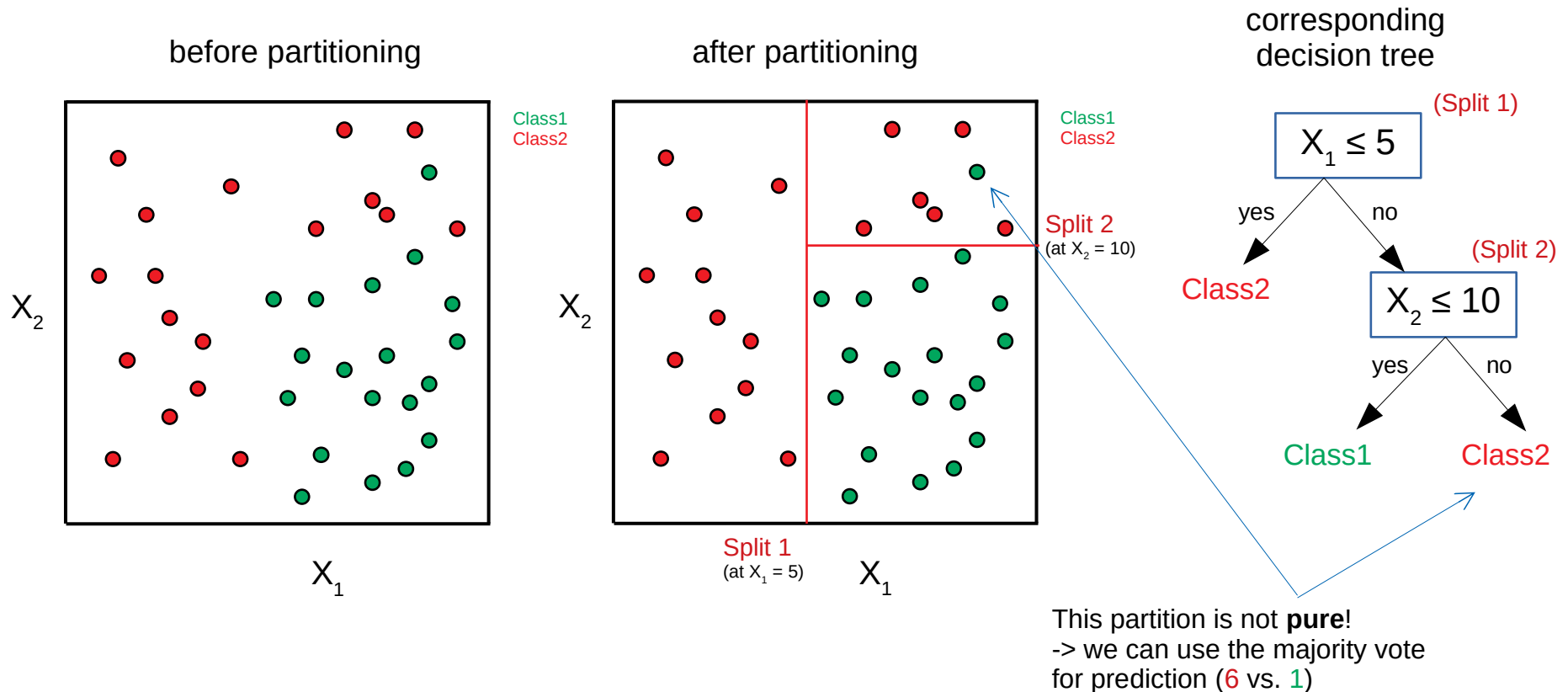*Splitting Attributes*

Model: Decision Tree

Only straight lines!

Decision tree – partitioning

▶ A classic decision tree splits the data at each node based on **one predictor variable**

▶ For the case of only **two predictor variables** we can visualize this in a 2D-coordinate system:

corresponding decision tree

Decision tree – partitioning

▶ A classic decision tree splits the data at each node based on **one predictor variable**

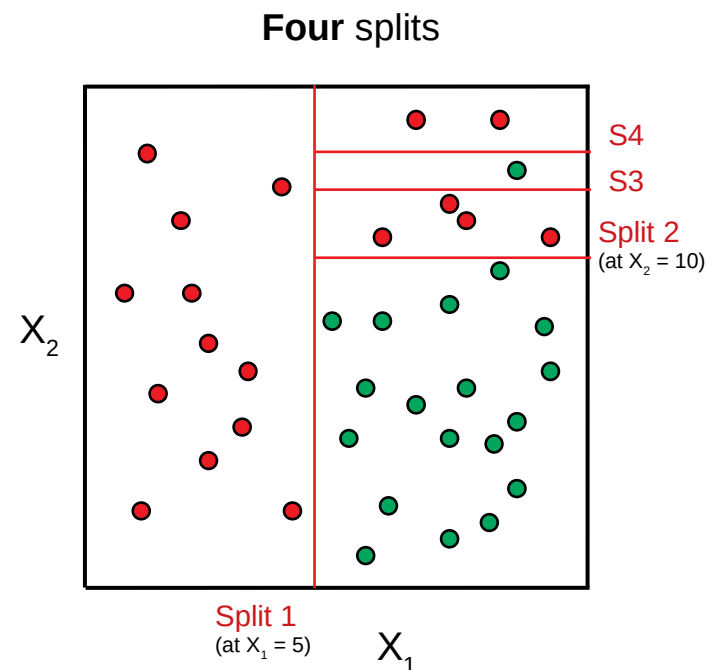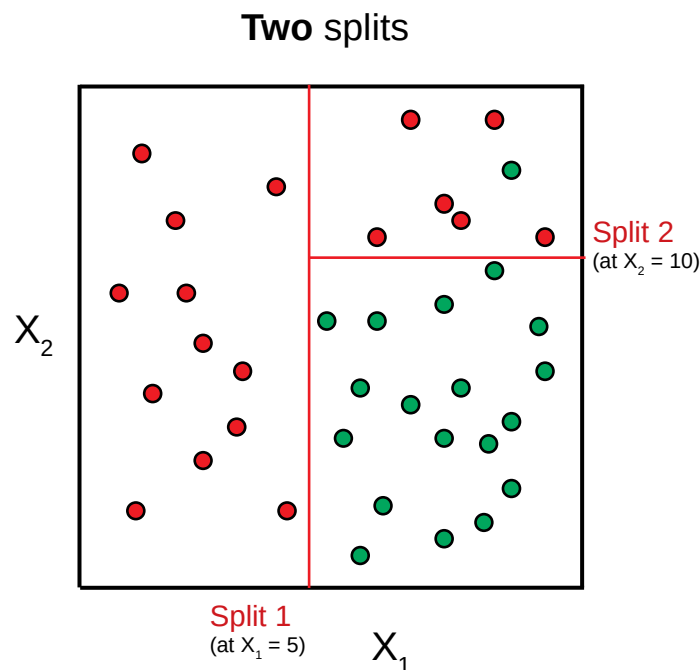▶ For the case of only **two predictor variables** we can visualize this in a 2D-coordinate system:

before partitioning

after partitioning

corresponding decision tree

Class1
Class2

Split 2
(at X_2 = 10)

Split 1
(at X_1 = 5)

(Split 1)

$X_1 \leq 5$

yes   no

Class2

(Split 2)

$X_2 \leq 10$

yes   no

Class1   Class2

This partition is not **pure**!
-> we can use the majority vote for prediction (6 vs. 1)

Decision trees – over- and underfitting

► How **deep** should a tree be allowed to grow?

  ▷ We can grow a tree until there are only pure partitions left (right figure below)

  ▷ Same problem of **over- and underfitting** like with the KNN-classifier!

► Which one of the trees below is better?

Two splits

Split 2 (at X_2 = 10)

X_2

Split 1 (at X_1 = 5) X_1

Four splits

S4
S3
Split 2 (at X_2 = 10)

X_2

Split 1 (at X_1 = 5) X_1

Recap: Test-error

▶ Like in the case of the KNN-classifier we want to minimize the **test-error**

▶ Evaluation methods like e.g. **cross-validation** can of course also be applied to decision trees

  ▷ Prevent **over-fitting**: Fully grown trees can be "**pruned**" (cut shorter) depending on whether a branch improves the fit to data evaluated by cross-validation
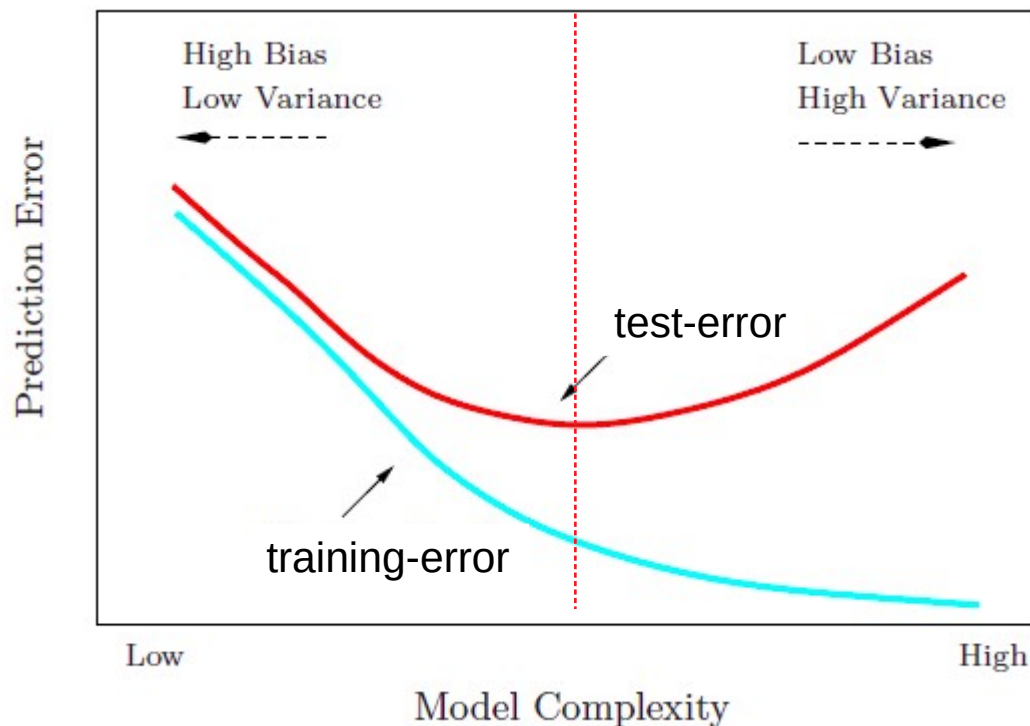
How are the partitions generated?

▶ Building a decision tree from scratch involves the following two questions at each node:

  ▷ **Which** variable should be used for the next split?

  ▷ **Where** along the chosen variable should we split?

▶ How can we decide what the **best variable** and **split-location** is?

▶ We will look at two different methods to solve these questions

  ▷ 1) classic partitioning based on **impurity measure** (e.g. Gini-index)

  ▷ 2) bias-free partitioning based on **significance tests**
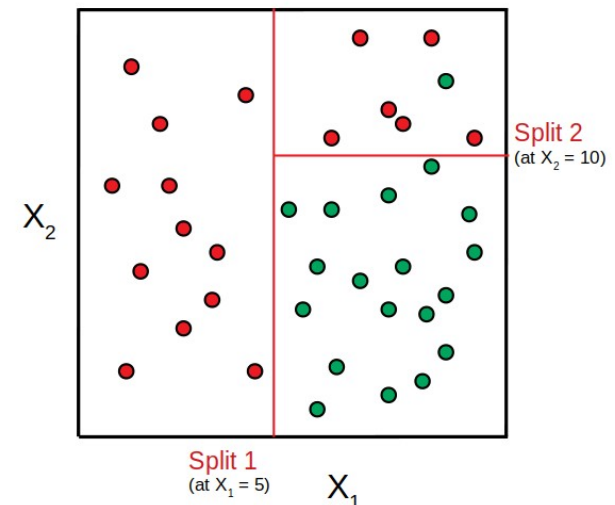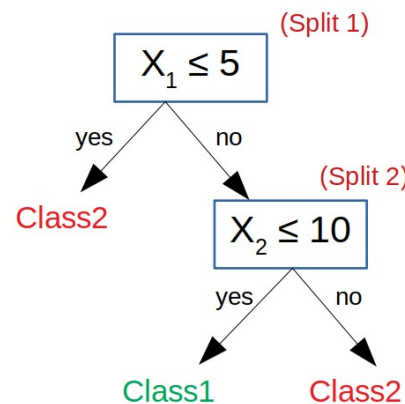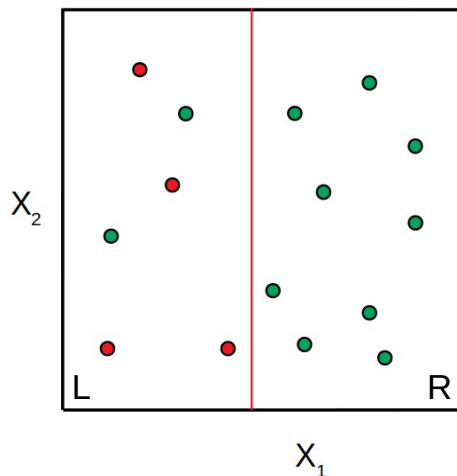
Classification trees – impurity measure

▶ **Main idea**: Try to find the split, which reduces the **impurity** of the data the most

  ▷ How can we measure "impurity"?

▶ The Gini index is one of the most common impurity measures

$$Gini = 1 - \sum_i^C p_i^2$$

Entropy is another common impurity measure

$$Entropy = -\sum_i^C p_i \log_2(p_i)$$

Example

X_2

X_1

$Gini_{root} = 1 - ((11/15)^2 + (4/15)^2) = $ **0.391**

$Gini_L = 1 - ((2/6)^2 + (4/6)^2) = $ **0.444**
$Gini_R = 1 - ((0/9)^2 + (9/9)^2) = $ **0**

Gini-reduction (weighted by number of observations in each partition):

$Gini\_decrease = Gini_{root} - (6/15)*Gini_L - (9/15)*Gini_R$

Classification trees – impurity measure

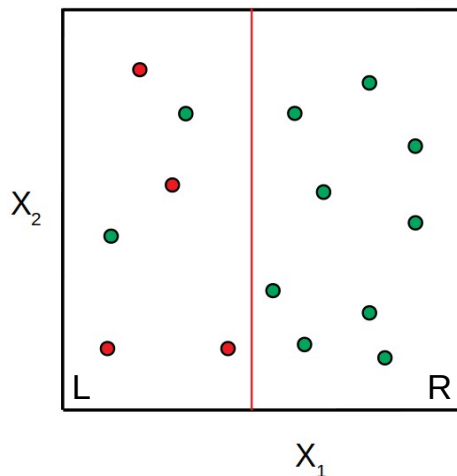▶ **Main idea**: Try to find the split, which reduces the **impurity** of the data the most

 ▷ How can we measure "impurity"?

▶ The Gini index is one of the most common impurity measures

$$Gini = 1 - \sum_{i}^{C} p_i^2$$

Entropy is another common impurity measure

$$Entropy = -\sum_{i}^{C} p_i \log_2(p_i)$$

Example

$Gini_{root} = 1 - ((11/15)^2 + (4/15)^2) = \textbf{0.391}$

$Gini_L = 1 - ((2/6)^2 + (4/6)^2) = \textbf{0.444}$
$Gini_R = 1 - ((0/9)^2 + (9/9)^2) = \textbf{0}$

Gini-reduction (weighted by number of observations in each partition):

$Gini\_decrease = Gini_{root} - (6/15)*Gini_L - (9/15)*Gini_R$

→ Choose the split with the largest Gini-reduction

Partition algorithm (impurity measure based)

$$Gini = 1 - \sum_i^C p_i^2$$

▶ Start with a **single region** (encompassing all data)

▶ Iterate through the following steps:

For each region **R**

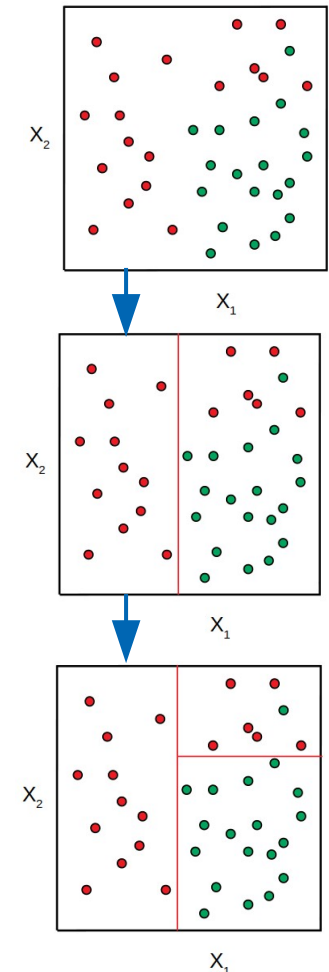　For each variable $\mathbf{x}_i$ in **R**

　　For each possible split $\mathbf{s}_i$ of $\mathbf{x}_i$

　　　Record impurity decrease

→ Choose ($\mathbf{x}_i$, $\mathbf{s}_i$) which gives maximum impurity decrease

→ Replace **R** with $\mathbf{R}_R$ and $\mathbf{R}_L$

▶ Stop splitting either based on a **stopping rule** or when there is no more impurity reduction possible

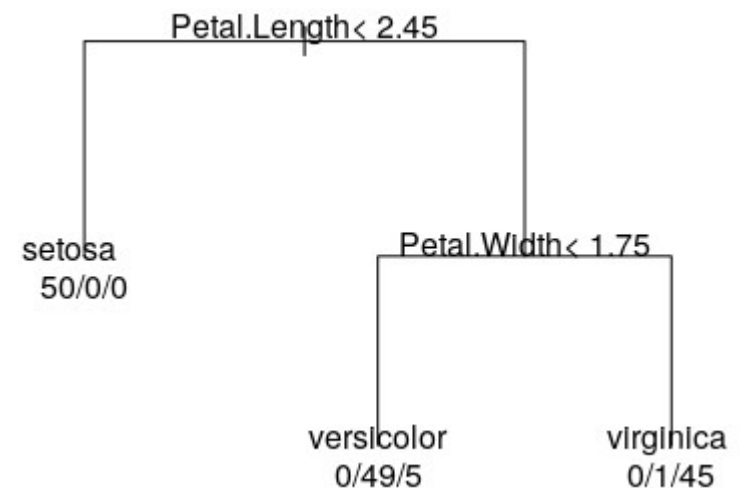　▶ Stopping rule example: Stop splitting when a branch contains less than a certain percentage of the data

Decision tree in R – rpart

▶ **rpart** is an R-package to fit decision trees (impurity measure based)

▶ Here an example of fitting a (classification) tree to the "Iris" data set:

```
library(rpart)
tree.iris <- rpart(Species ~., data=iris)  # default uses Gini-index
plot(tree.iris, margin = 0.1)
text(tree.iris, use.n = TRUE)
# Predict training data (use type = 'prob' to get probabilities):
pred.tree <- predict(tree.iris, iris, type = 'class')
# Confusion matrix:
(confT <- table(pred.tree, iris$Species))
pred.tree    setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         49         5
  virginica       0          1        45
# Training-error:
diag(confT) <- 0
missCount <- sum(confT)
(trainErr <- missCount/nrow(iris))
[1] 0.04
```

Decision tree in R – rpart

▶ **rpart** is an R-package to fit decision trees (impurity measure based)

▶ Here an example of fitting a (classification) tree to the "Iris" data set:

What is this?

```
library(rpart)
tree.iris <- rpart(Species ~., data=iris)  # default uses Gini-index
plot(tree.iris, margin = 0.1)
text(tree.iris, use.n = TRUE)
# Predict training data (use type = 'prob' to get probabilities):
pred.tree <- predict(tree.iris, iris, type = 'class')
# Confusion matrix:
(confT <- table(pred.tree, iris$Species))
pred.tree    setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         49         5
  virginica       0          1        45
# Training-error:
diag(confT) <- 0
missCount <- sum(confT)
(trainErr <- missCount/nrow(iris))
[1] 0.04
```
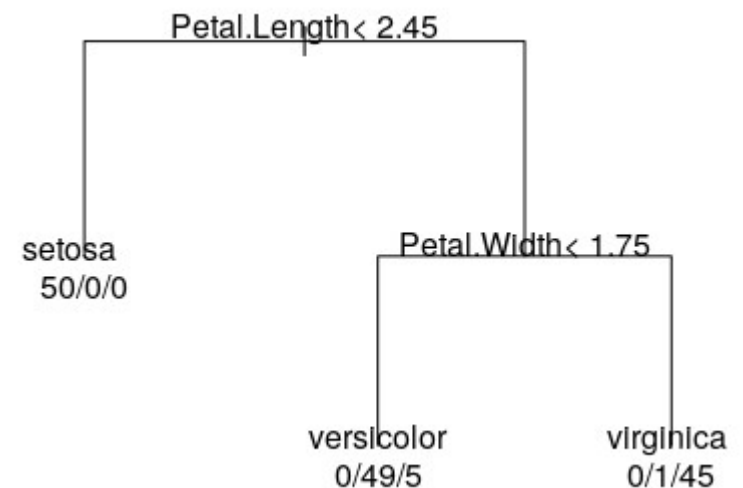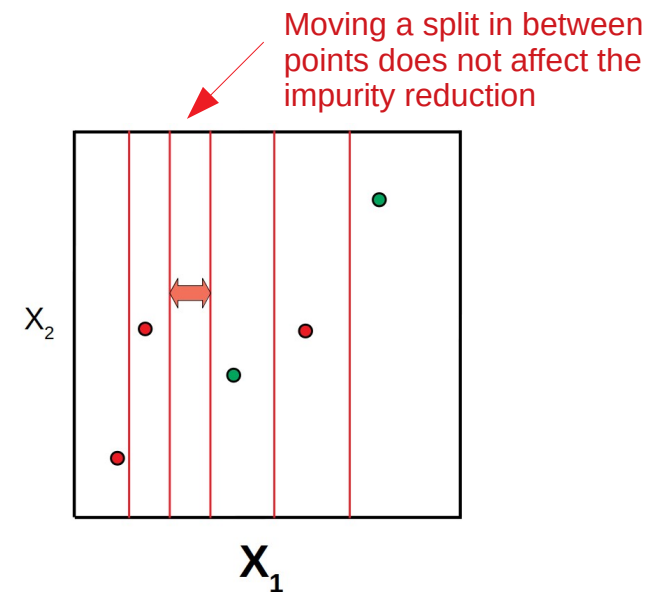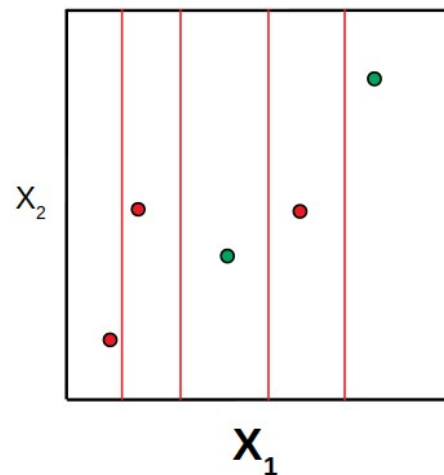
Problems with rpart

▷ Using an impurity measure based approach to select splits suffers from an **inherent bias** in certain cases

  ▷ This bias is related to the **number of splits** that are possible for each variable

▷ How many splits are there per variable?

  ▷ For **continuous** variable (numeric):

Moving a split in between points does not affect the impurity reduction
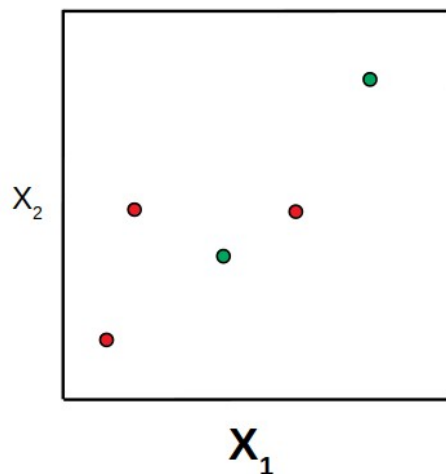
  ▷ For the five data points above, there are only **four ways** to separate the data (along X_1)

▷ What about **other types** of variables?

Problems with rpart

▶ We can also include **categorical variables** (factors) as predictors in a decision tree

▶ How many splits are possible for factors?

    ▷ E.g. **two**-level factor (levels: a,b)
       a **|** b split1

    ▷ E.g. **four**-level factor (levels: a,b,c,d):
       a **|** b,c,d split1
       b **|** a,c,d split2
       c **|** a,b,d split3
       d **|** a,b,c split4
       a,b **|** c,d   split5
       a,c **|** b,d   split6
       a,d **|** b,c   split7

    ▷ **k**-level factor: $\boxed{number\ of\ splits = 2^{(k-1)} - 1}$ → 10-levels: 511 splits

▶ With increasing number of levels we can apply an **exponentially growing** number of splits

Training Data

Model: Decision Tree
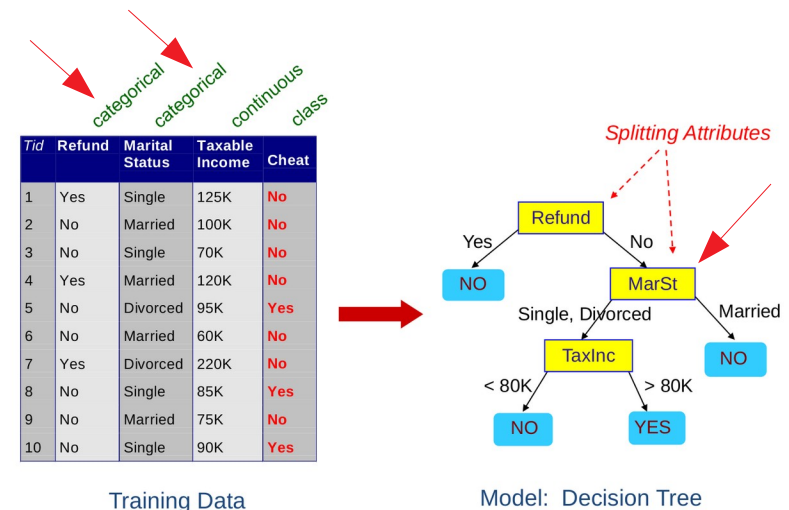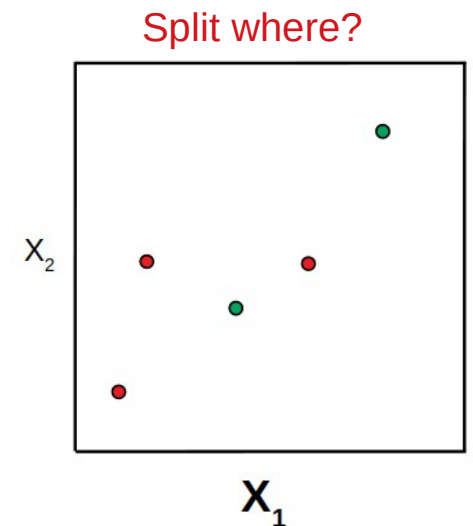
Source: Tan, Steinbach, Kumar

Problems with rpart

▷ With increasing number of levels we can apply an **exponentially growing** number of splits

▷ The difference in numbers of splits between continuous variables, factors with few levels and factors with many levels creates an issue of **multiple comparisons**

▷ Multiple comparisons problem in decision trees:

```
E.g. two-level factor (levels: a,b)
a I b  split1

E.g. four-level factor (levels: a,b,c,d):
a  I  b,c,d  split1
b  I  a,c,d  split2
c  I  a,b,d  split3
d  I  a,b,c  split4
ab I  c,d    split5
ac I  b,d    split6
ad I  b,c    split7
```

▷ Even if there is **no real relation** between predictors and the target variable, under certain circumstances multi-level factors will be preferably selected for splitting compared to numerical variables or factors with few levels because they get **more "chances"** to reach a high impurity-decrease (even if this decrease is only achieved by pure luck)

▷ More chances in the sense that more splits can be evaluated (more chances to get lucky)

▷ Ideally, we **do not want** our decision trees to be biased in their variable selection

Bias-free partitioning

- **Bias-free partitioning** has been proposed to solve this problem (e.g. see Hothorn et al. 2006)

  - Main idea: Use significance tests instead of impurity measures to select variable and next split

- How does it work?

  - The selection process is more clearly separated into

    1) Choosing the next splitting **variable** and

    2) Choosing the splitting **location** along the chosen variable

Split where?

Bias-free partitioning

▶ Choosing the next splitting variable:

   ▷ Test for **each variable** whether it has a **significant association** with the target variable (e.g. think of correlation test for numerical variables and of ANOVA for categorical variables). Collect the associated p-values (can be compared between tests)

   ▷ If no variable shows a significant p-value stop splitting (integrated **stopping rule**)

   ▷ Otherwise choose the variable with the **lowest p-value** for the next split

Bias-free partitioning

Actual implementation uses special permutation tests (called "**conditional inference tests**") that are available for all types of variables.

▶ Choosing the next splitting variable:

  ▷ Test for **each variable** whether it has a **significant association** with the target variable (e.g. think of correlation test for numerical variables and of ANOVA for categorical variables). Collect the associated p-values (can be compared between tests)

  ▷ If no variable shows a significant p-value stop splitting (integrated **stopping rule**)

  ▷ Otherwise choose the variable with the **lowest p-value** for the next split

Bias-free partitioning

▶ Choosing the next splitting variable:

 ▷ Test for **each variable** whether it has a **significant association** with the target variable (e.g. think of correlation test for numerical variables and of ANOVA for categorical variables). Collect the associated p-values (can be compared between tests)

 ▷ If no variable shows a significant p-value stop splitting (integrated **stopping rule**)

 ▷ Otherwise choose the variable with the **lowest p-value** for the next split

▶ Choosing the next splitting location:

 ▷ Calculate for each possible split (along the chosen variable) a **test-statistic** (expressing the difference between the created partitions, *could also use impurity measure*) and select split with the highest test-statistic

▶ Using this approach, there is **no bias** in the selection of different variable-types
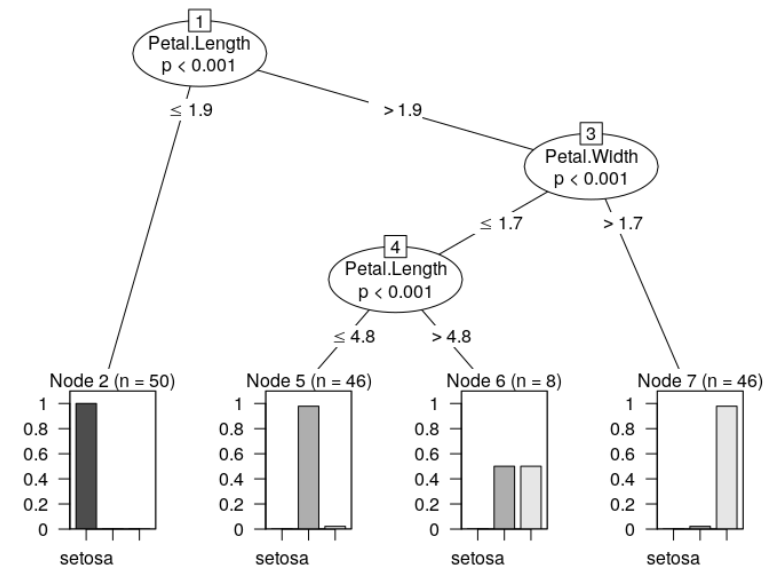
Bias-free partitioning in R

- ▶ "partykit" is an R-package which includes functions to fit bias-free (significance test based) decision trees (**ctree**()-function)

- ▶ Fitting a classification tree to the "Iris" data set

```
> library(partykit)
> ctree.iris <- ctree(Species ~., data=iris)   # fit tree
> plot(ctree.iris)
> pred.ctree <- predict(ctree.iris, newdata=iris, type='response')
> confT <- table(pred.ctree, iris$Species)   # Confusion matrix
```

```
> confT
pred.ctree    setosa versicolor virginica
   setosa         50          0         0
   versicolor      0         49         5
   virginica       0          1        45
```
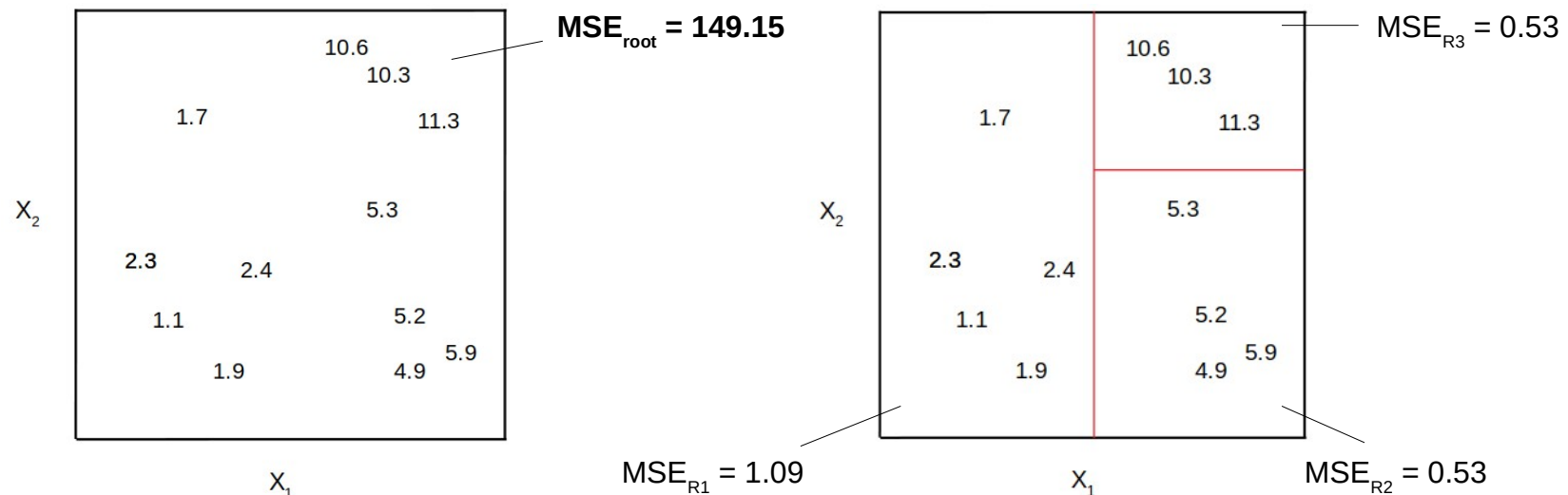
Decision trees for regression

▶ Decision trees can be used for classification (categorical target variable) and for **regression** (numerical target variable)

▶ The general principle behind building the tree structure does not change

    ▷ E.g. use **mean squared error** as "impurity measure" instead of Gini-index

$$MSE = \frac{1}{n} \sum_{i}^{n} (x_i - \bar{x})^2$$

    ▷ We try to find the splits which best reduce the mean squared error

▶ With **ctree** we can fit a bias-free regression tree (again based on significance tests)

MSE$_{root}$ **= 149.15**
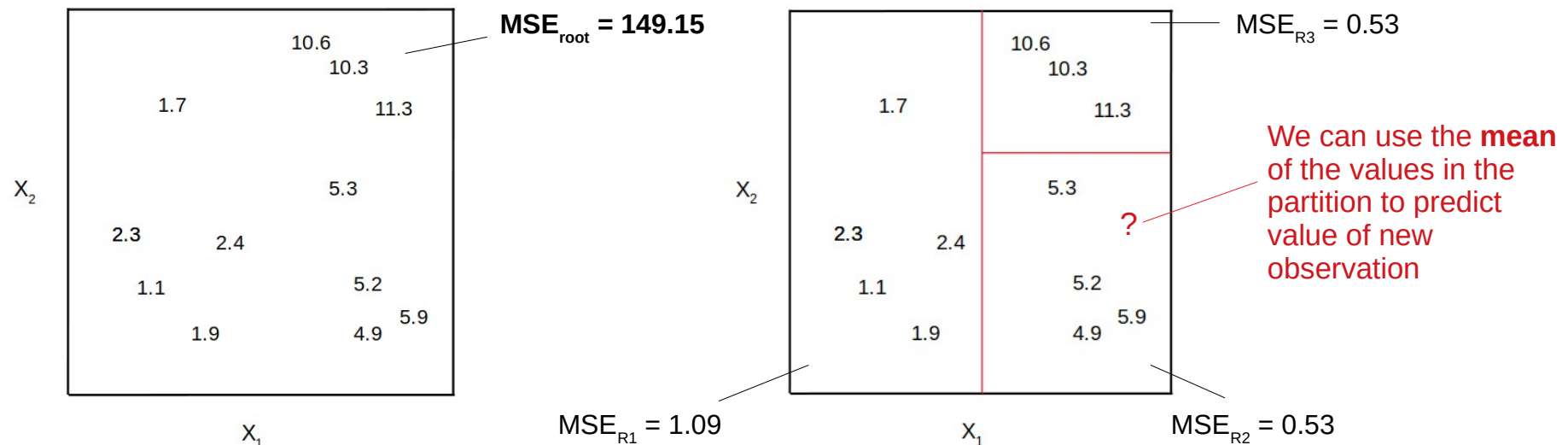
MSE$_{R3}$ = 0.53

MSE$_{R1}$ = 1.09

MSE$_{R2}$ = 0.53

Decision trees for regression

▷ Decision trees can be used for classification (categorical target variable) and for **regression** (numerical target variable)

▷ The general principle behind building the tree structure does not change

    ▷ E.g. use **mean squared error** as "impurity measure" instead of Gini-index

$$MSE = \frac{1}{n} \sum_{i}^{n} (x_i - \bar{x})^2$$

    ▷ We try to find the splits which best reduce the mean squared error

▷ With **ctree** we can fit a bias-free regression tree (again based on significance tests)

MSE$_{root}$ = 149.15

MSE$_{R3}$ = 0.53

We can use the **mean** of the values in the partition to predict value of new observation

MSE$_{R1}$ = 1.09

MSE$_{R2}$ = 0.53

Regression tree in R
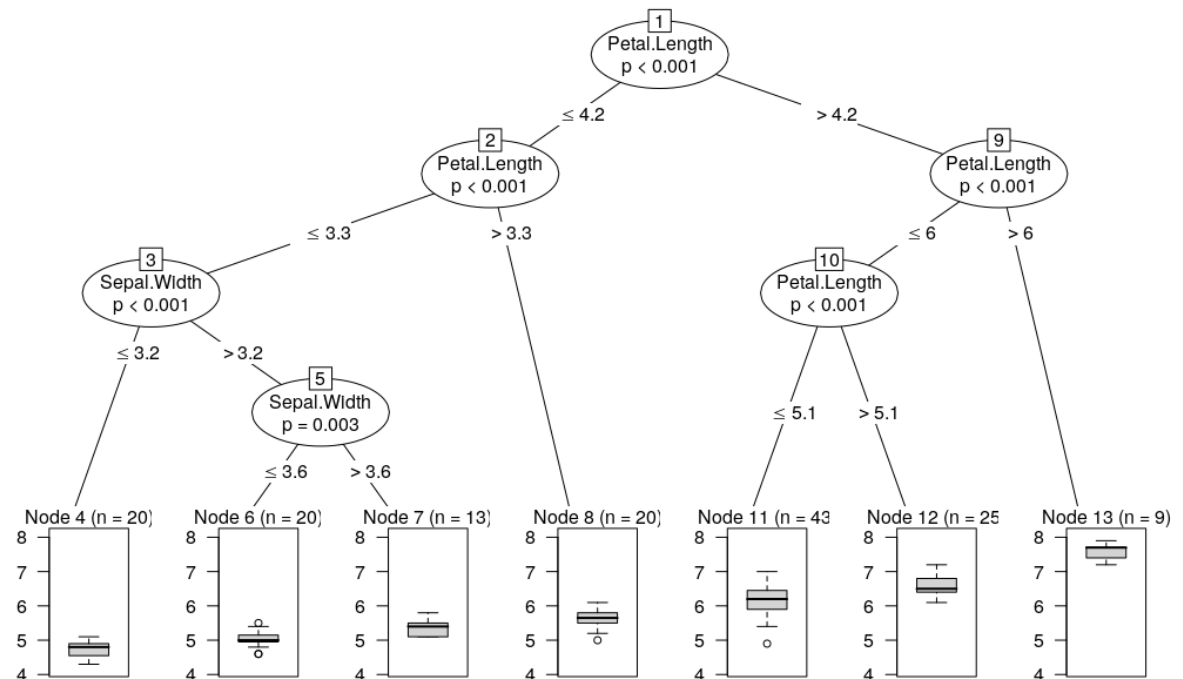
▶ Example of fitting a regression tree to the "Iris" data using ctree()

```r
library(partykit)
ctree.iris <- ctree(Sepal.Length ~., data=iris)  # fit regression tree
plot(ctree.iris)
# Get the predicted values (for training data):
pred.ctree <- predict(ctree.iris, newdata=iris, type='response')
```

Decision trees – Summary

- Advantages of decision trees:

  - Good interpretability

  - No assumptions regarding distributions

  - Can capture non-linear structures and complex interactions

- Disadvantages:

  - Single trees are instable (small changes in data can give different-looking tree, especially if underlying pattern is complex)

  - In case of not pruned rpart-trees: Tend to overfit

  - Needs a lot of data to capture linear structures

Caution: In case of different types of predictor variables use algorithm with bias-free variable selection (ctree in R)

Further reading

▶ Strobl, C., Malley, J., & Tutz, G. (2009). **An introduction to recursive partitioning: Rationale, application, and characteristics of classification and regression trees, bagging, and Random Forests**. *Psychological Methods*, 14 (4), 323–348. doi: 10.1037/a0016973