UNIT 4

Random Forests™

JOHANNES KEPLER
UNIVERSITY LINZ

BIOINF

DECISION TREES: INTRODUCTION

- A decision tree is a classifier that classifies samples “by asking questions successively”; each non-leaf node corresponds to a question, each leaf corresponds to a final prediction.
- Decision tree learning is concerned with partitioning the training data hierarchically such that the leaf nodes are hopefully homogeneous in terms of the target class.
- Decision trees have mainly been designed for categorical data, but they can also be applied to numerical features.
- Decision trees are traditionally used for classification (binary and multi-class), but regression is possible, too.

DECISION TREE LEARNING

- All decision tree learning algorithms are recursive, depth-first search algorithms that perform hierarchical splits.
- There are three main design issues:
  1. Splitting criterion: which splits to choose?
  2. Stopping criterion: when to stop further growing of the tree?
  3. Pruning: whether/how to collapse unnecessarily deep sub-trees?

  The two latter are especially relevant for adjusting the complexity of decision trees (underfitting vs. overfitting).

DECISION TREE LEARNING (cont'd)

1. Given: training set $\mathbf{Z} = \{(\mathbf{x}^i, y^i) \mid i = 1, \ldots, l\}$
2. Call DecTree($\mathbf{Z}$, Root, $\{$all possible splits$\}$)
3. DecTree($\mathbf{Z}$, N, I)
   a. If stopping criterion is fulfilled, exit.
   b. Determine split $i \in I$ such that splitting criterion is maximal.
   c. Divide $\mathbf{Z}$ into disjoint subsets $\mathbf{Z}_{i,j}$ according to the split i.
   d. For all j such that $\mathbf{Z}_{i,j} \neq \emptyset$
      - Generate new node N_j
      - Call DecTree($\mathbf{Z}_{i,j}$, N_j, $I \backslash \{i\}$)

SPLITTING: CATEGORICAL FEATURES

Binary split: $\mathbf{Z}_L = \{(\mathbf{x}, y) \in \mathbf{Z} \mid x_i = c\}$, $\mathbf{Z}_R = \{(\mathbf{x}, y) \in \mathbf{Z} \mid x_i \neq c\}$
Split according to entire feature: $\mathbf{Z}_j = \{(\mathbf{x}, y) \in \mathbf{Z} \mid x_i = c_j\}$

Typically, all possible splits w.r.t. all features are considered, but either binary or entire feature splits.

Example: consider a categorical attribute i with values {A, B, C}.

SPLITTING: CATEGORICAL FEATURES

Binary split: $\mathbf{Z}_L = \{(\mathbf{x}, y) \in \mathbf{Z} \mid x_i = c\}$, $\mathbf{Z}_R = \{(\mathbf{x}, y) \in \mathbf{Z} \mid x_i \neq c\}$
Split according to entire feature: $\mathbf{Z}_j = \{(\mathbf{x}, y) \in \mathbf{Z} \mid x_i = c_j\}$

Typically, all possible splits w.r.t. all features are considered, but either binary or entire feature splits.

Example: consider a categorical attribute i with values $\{$A, B, C$\}$.

Binary splits:

$$\begin{tikzcd}
& \mathbf{Z} \quad x_i = \text{A?} \arrow[dl, "\text{yes}"'] \arrow[dr, "\text{no}"] & \\
\{(\mathbf{x}, y) \mid x_i = \text{A}\} & & \{(\mathbf{x}, y) \mid x_i \neq \text{A}\}
\end{tikzcd}$$

$$\begin{tikzcd}
& \mathbf{Z} \quad x_i = \text{B?} \arrow[dl, "\text{yes}"'] \arrow[dr, "\text{no}"] & \\
\{(\mathbf{x}, y) \mid x_i = \text{B}\} & & \{(\mathbf{x}, y) \mid x_i \neq \text{B}\}
\end{tikzcd}$$

$$\begin{tikzcd}
& \mathbf{Z} \quad x_i = \text{C?} \arrow[dl, "\text{yes}"'] \arrow[dr, "\text{no}"] & \\
\{(\mathbf{x}, y) \mid x_i = \text{C}\} & & \{(\mathbf{x}, y) \mid x_i \neq \text{C}\}
\end{tikzcd}$$

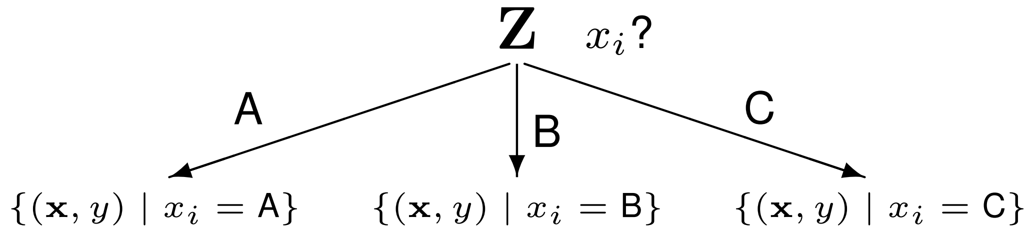
SPLITTING: CATEGORICAL FEATURES

Binary split: $\mathbf{Z}_L = \{(\mathbf{x}, y) \in \mathbf{Z} \mid x_i = c\}, \mathbf{Z}_R = \{(\mathbf{x}, y) \in \mathbf{Z} \mid x_i \neq c\}$
Split according to entire feature: $\mathbf{Z}_j = \{(\mathbf{x}, y) \in \mathbf{Z} \mid x_i = c_j\}$

Typically, all possible splits w.r.t. all features are considered, but either binary or entire feature splits.

Example: consider a categorical attribute i with values {A, B, C}.

Split of entire feature:


$\mathbf{Z}$ x_i?
A
B
C
$\{(\mathbf{x}, y) \mid x_i = \text{A}\}$
$\{(\mathbf{x}, y) \mid x_i = \text{B}\}$
$\{(\mathbf{x}, y) \mid x_i = \text{C}\}$

SPLITTING: NUMERICAL FEATURES

Apply threshold c to i-th feature, i.e.

$$\mathbf{Z}_L = \{(\mathbf{x}, y) \mid x_i < c\}, \mathbf{Z}_R = \{(\mathbf{x}, y) \mid x_i \geq c\}.$$

Typically, all possible splits w.r.t. all features are considered, where thresholds are chosen as mean values of “neighboring” values occurring in the data.

Example: if the values 0, 0.1, 0.4, 0.8, and 1.2 occur in the data for a numerical feature, the following thresholds are considered: 0.05, 0.25, 0.6, and 1. For large data sets with many different values for each feature, this is unpractical. Instead, fine-grained binnings of the ranges of the features can be considered.

COMMON SPLITTING CRITERIA

Classification:

- Information gain
- Gini impurity (gain)

Regression:

- variance reduction

INFORMATION GAIN

For any (sub)set of data $\mathbf{Z}$, let us define the relative proportions of samples belonging to the k-th class (of classes $1, \ldots, M$) as

$$p_k(\mathbf{Z}) = \frac{|\{(\mathbf{x}, y) \in \mathbf{Z} \mid y = k\}|}{|\mathbf{Z}|}.$$

Then the *entropy* of $\mathbf{Z}$ w.r.t. the target is defined as

$$H(\mathbf{Z}) = -\sum_{k=1}^{M} p_k(\mathbf{Z}) \cdot \log\big(p_k(\mathbf{Z})\big).$$

The entropy is maximal if the classes are uniformly distributed in the set $\mathbf{Z}$. It is 0 (minimal) if all samples of $\mathbf{Z}$ belong to one single class.

INFORMATION GAIN (cont'd)

The *information gain* of employing the i-th split for partitioning $\mathbf{Z}$ into sets $\mathbf{Z}_{i,1}, \ldots, \mathbf{Z}_{i,K_i}$ is then defined as

$$g_E(\mathbf{Z}, i) = H(\mathbf{Z}) - \sum_{j=1}^{K_i} \frac{|\mathbf{Z}_{i,j}|}{|\mathbf{Z}|} \cdot H(\mathbf{Z}_{i,j}).$$

Comments:

- Typically, in each step, all possible splits are considered and the one with the highest information gain is selected.
- The information gain is nothing else but the *Kullback-Leibler divergence*.
- Information gain is the standard splitting criterion in the decision tree algorithms *ID3*, *C4.5*, and *C5.0*.

GINI IMPURITY (GAIN)

With the notations above, the *Gini impurity* of $\mathbf{Z}$ is defined as

$$I_G(\mathbf{Z}) = \sum_{k=1}^{M} p_k(\mathbf{Z}) \cdot \big(1 - p_k(\mathbf{Z})\big) = 1 - \sum_{k=1}^{M} p_k(\mathbf{Z})^2$$

This value is maximal if the classes are uniformly distributed in the set $\mathbf{Z}$. It is 0 (minimal) if all samples of $\mathbf{Z}$ belong to one single class.
The *Gini (impurity) gain* of employing the i-th split for partitioning $\mathbf{Z}$ into sets $\mathbf{Z}_{i,1}, \ldots, \mathbf{Z}_{i,K_i}$ is then defined as

$$g_G(\mathbf{Z}, i) = I_G(\mathbf{Z}) - \sum_{j=1}^{K_i} \frac{|\mathbf{Z}_{i,j}|}{|\mathbf{Z}|} \cdot I_G(\mathbf{Z}_{i,j}).$$

This is the standard splitting criterion employed by the decision tree algorithm *CART* for classification.

VARIANCE REDUCTION

For any (sub)set of data $\mathbf{Z} = \{(\mathbf{x}^i, y^i) \mid i = 1, \ldots, l\}$, let us denote the vector of target values as $\mathbf{y}(\mathbf{Z}) = (y^1, \ldots, y^l)$. The *reduction of variance* of employing the i-th split for partitioning $\mathbf{Z}$ into sets $\mathbf{Z}_{i,1}, \ldots, \mathbf{Z}_{i,K_i}$ is then defined as

$$g_V(\mathbf{Z}, i) = \mathrm{Var}\big(\mathbf{y}(\mathbf{Z})\big) - \sum_{j=1}^{K_i} \mathrm{Var}\big(\mathbf{y}(\mathbf{Z}_{i,j})\big).$$

This is the standard splitting criterion employed by the decision tree algorithm *CART* for regression.

STOPPING CRITERIA: WHEN TO QUIT FURTHER SPLITTING?

- Maximum depth reached
- Minimum number of samples in current node under limit
- Splitting gain below limit

[details: see literature / software documentation]

PRUNING DECISION TREES

Reduced error pruning: recursively remove nodes the removal of which does not lead to a decrease of prediction performance (on training set or validation set).

Cost-complexity pruning: recursively remove the sub-tree the removal of which leads to the smallest increase of error per removed leaf. This is repeated until only the root tree remains. Of the entire sequence of trees, the tree is used as final model that gives the best prediction performance (on training set or validation set).

[details: see literature / software documentation]

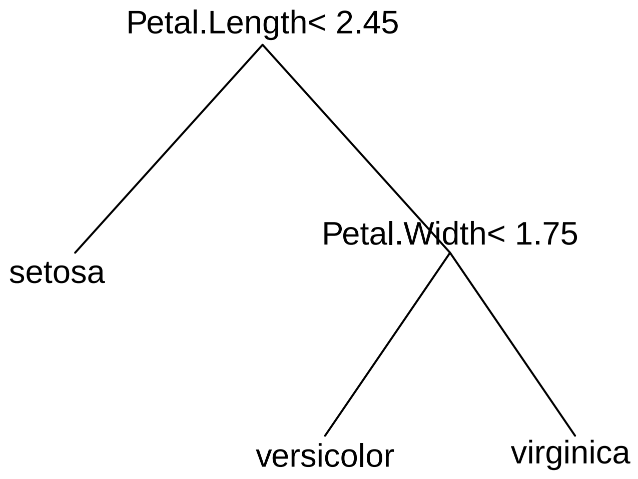
COMPUTING PREDICTIONS

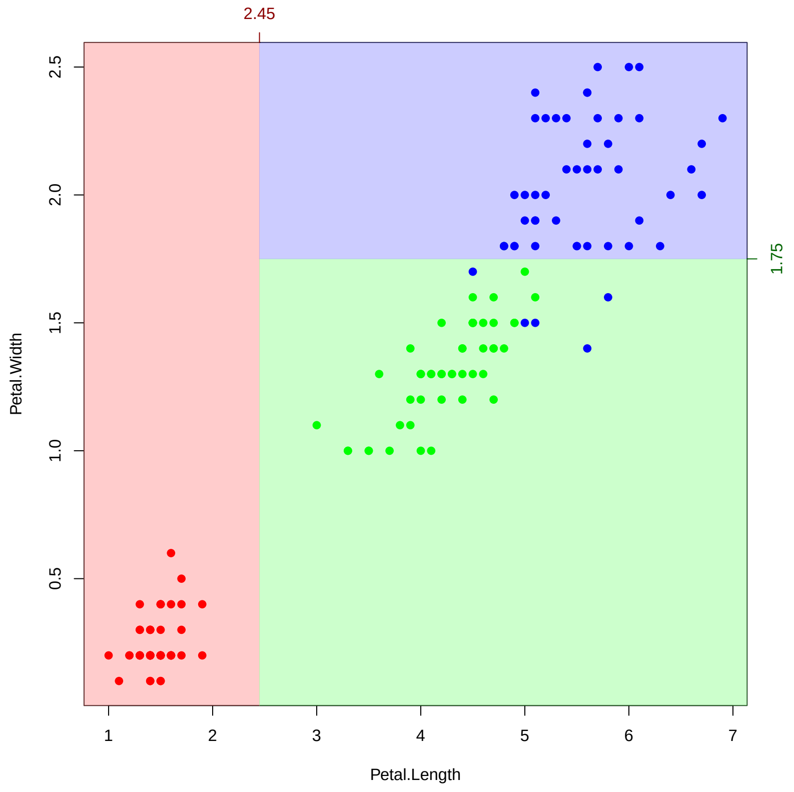
The tree recursively partitions/splits the training set into subsets, each of which is associated with a leaf node. This “assignment” of samples to leaf nodes is the basis for making predictions with decision trees: For a new input $\mathbf{x}$, traverse through the tree by answering the questions associated with each node until a leaf node is reached to which the sample $\mathbf{x}$ is assigned.

Classification: assign the class to the leaf node that appears most prominently among the training samples associated with this leaf node; alternatively: compute relative frequencies of classes in the leaf node and use these frequencies as estimates of the conditional probabilities $p(y = k \mid \mathbf{x})$.

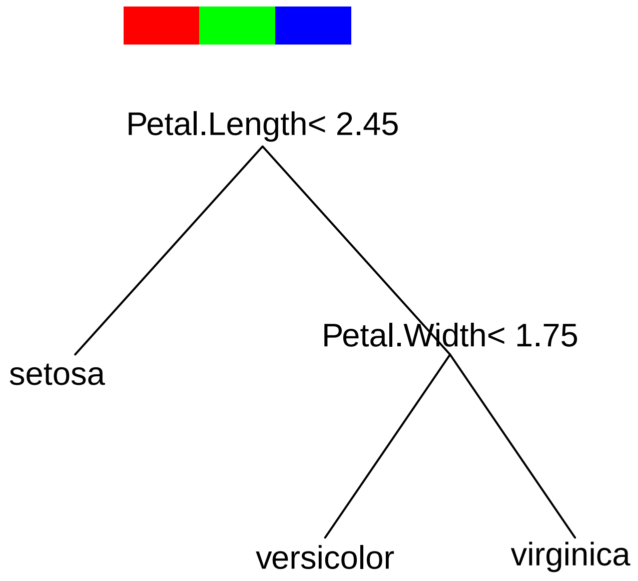
Regression: use the mean target value of samples associated with the leaf node.

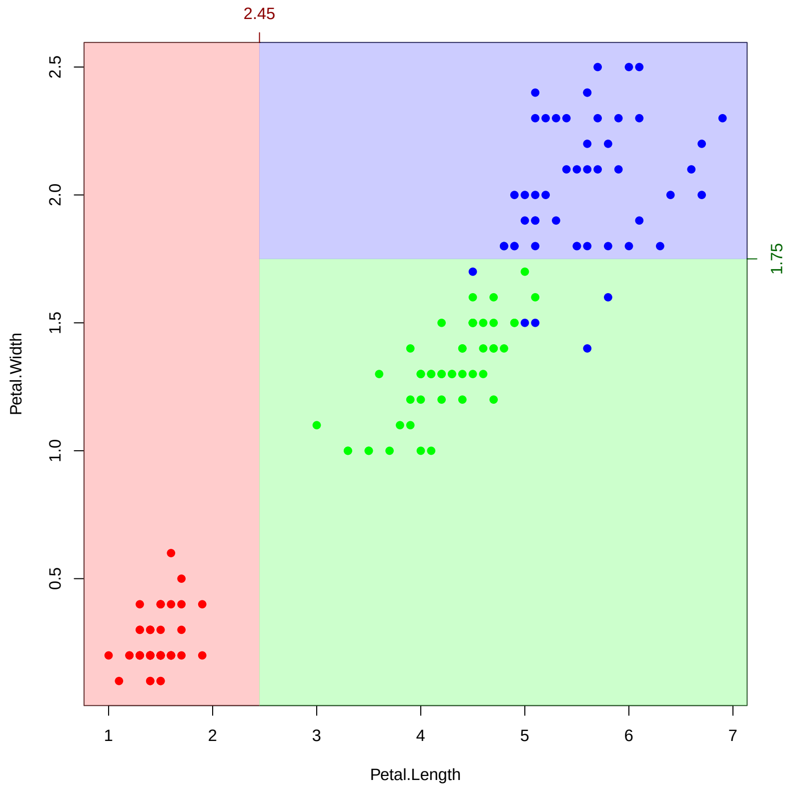
EXAMPLE: IRIS DATA SET



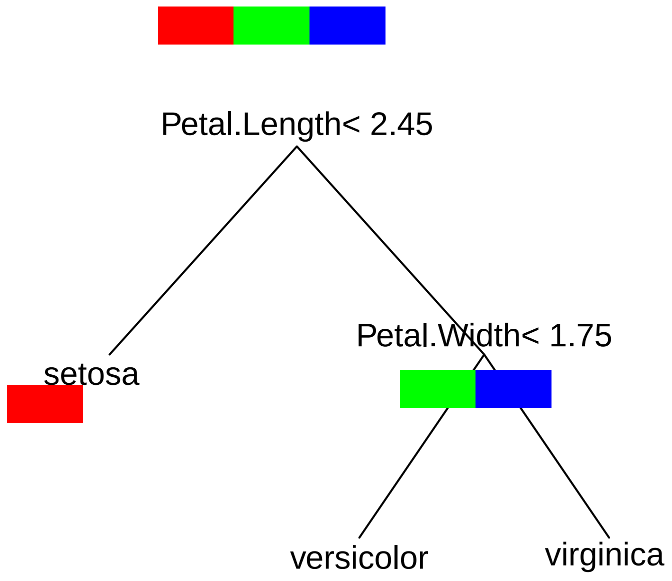


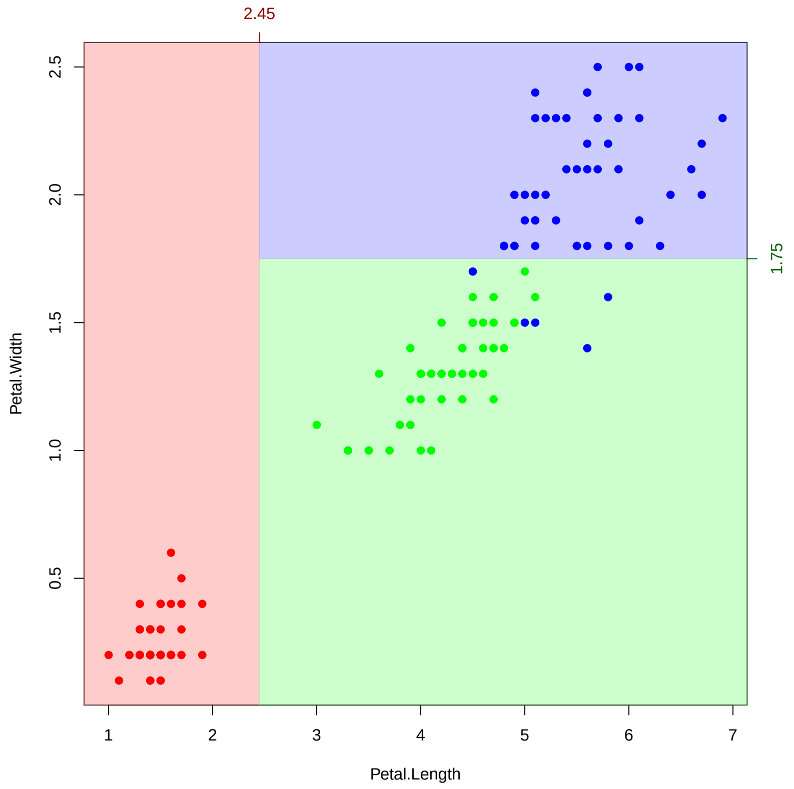
EXAMPLE: IRIS DATA SET



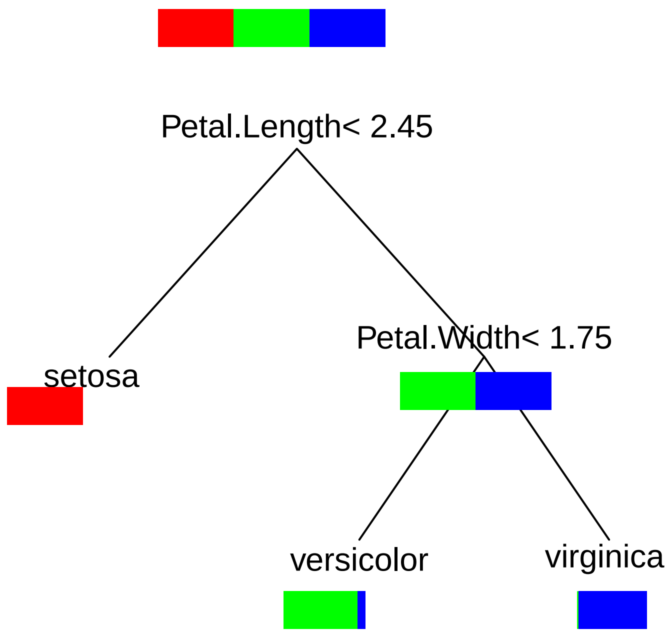


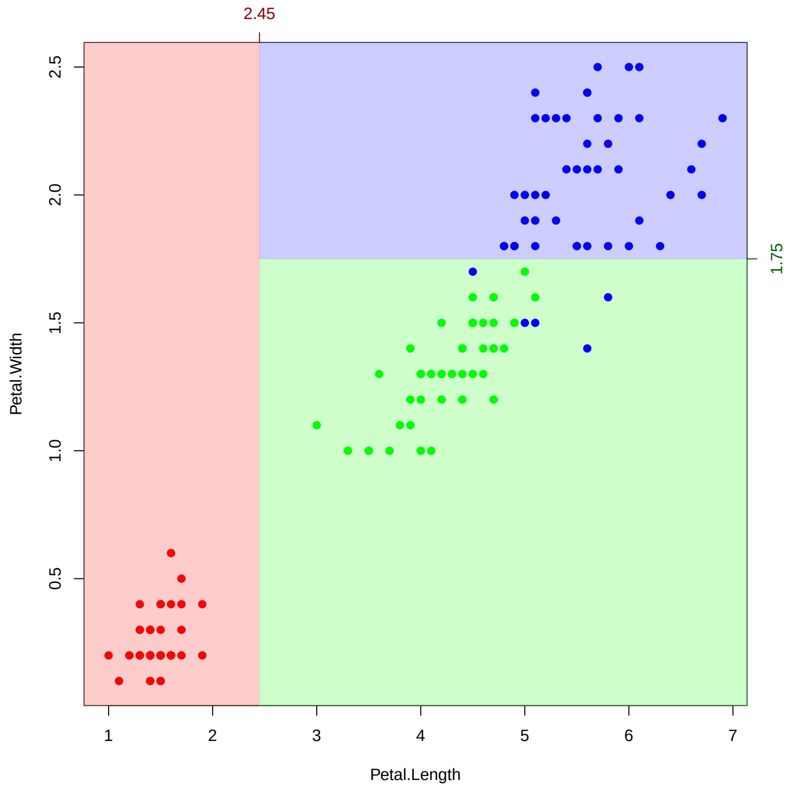
EXAMPLE: IRIS DATA SET





EXAMPLE: IRIS DATA SET





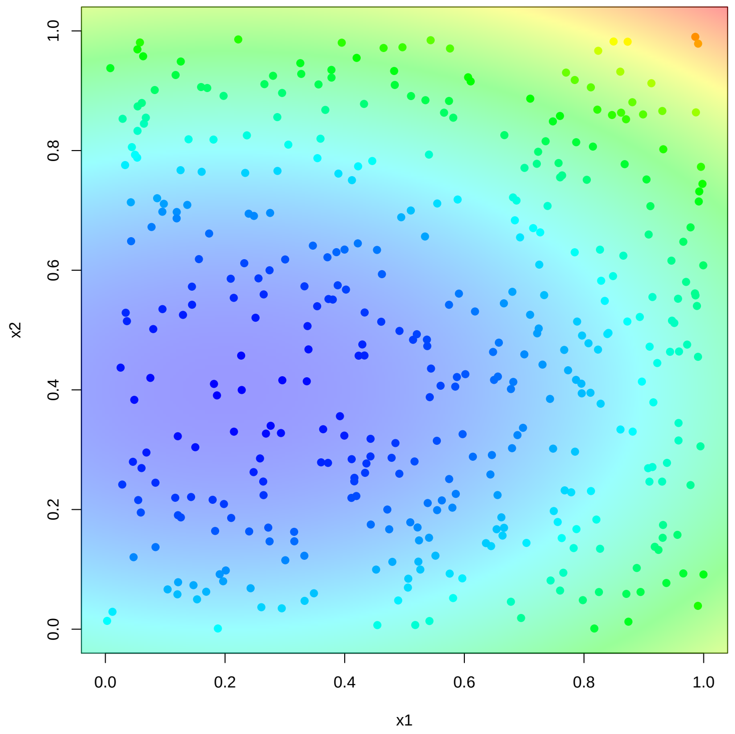
DECISION TREES AS RULE BASES

Every path from the tree root to a leaf can be interpreted as a rule, where each split corresponds to the fulfillment/non-fulfillment of a condition/predicate.

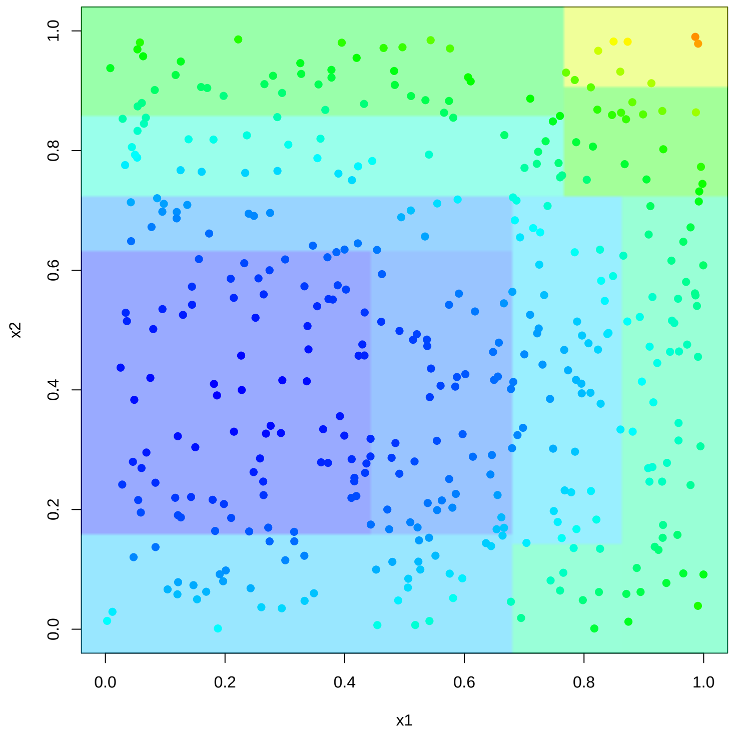
Example (Iris data set): the decision tree from the previous slide can be interpreted as follows:

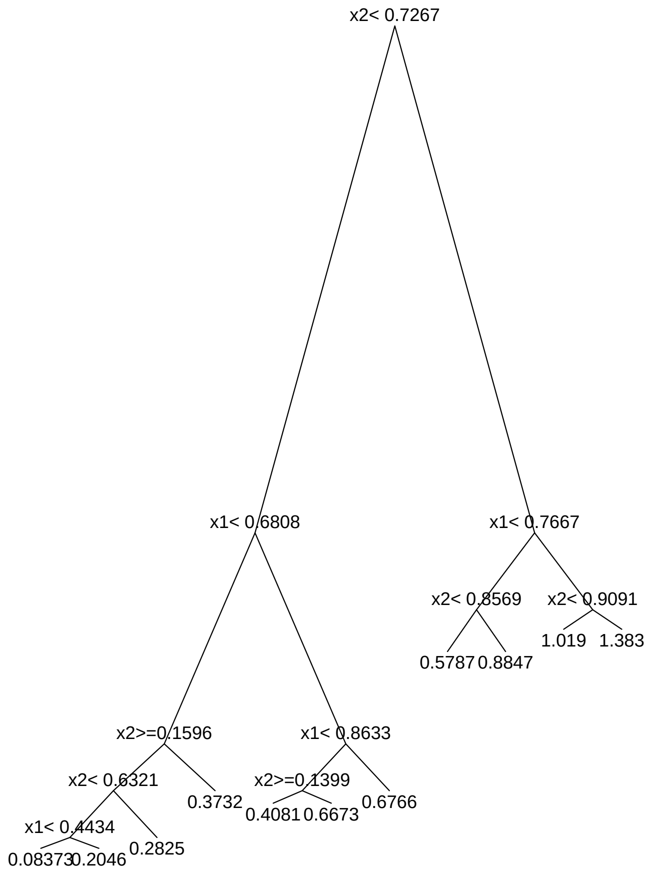
IF	Petal.Length < 2.45	**THEN**	Class $=$ setosa
IF	Petal.Length ≥ 2.45 & Petal.Width < 1.75	**THEN**	Class $=$ versicolor
IF	Petal.Length ≥ 2.45 & Petal.Width ≥ 1.75	**THEN**	Class $=$ virginica

EXAMPLE: REGRESSION DATA SET



EXAMPLE: REGRESSION DATA SET





ADVANTAGES OF DECISION TREES

- Simple and computationally efficient
- Built-in feature selection
- Interpretable models
- Can be applied to categorical and numerical attributes
- Scaling-invariant for numerical features
- Can applied both to classification (genuine multi-class support) and regression

DISADVANTAGES OF DECISION TREES

- Greedy splitting may lead to sub-optimal solutions. Finding an optimal tree is actually NP-complete.
- Only axis-parallel splits of numerical features (though there are variants that consider linear combinations of numerical features, too)
- Shallow trees are not accurate (high bias), deep trees overfit (high variance). This is the classical bias-variance trade-off like for any other machine learning method, but it is more difficult to address for decision trees (stopping criteria, pruning).

RANDOM FORESTS

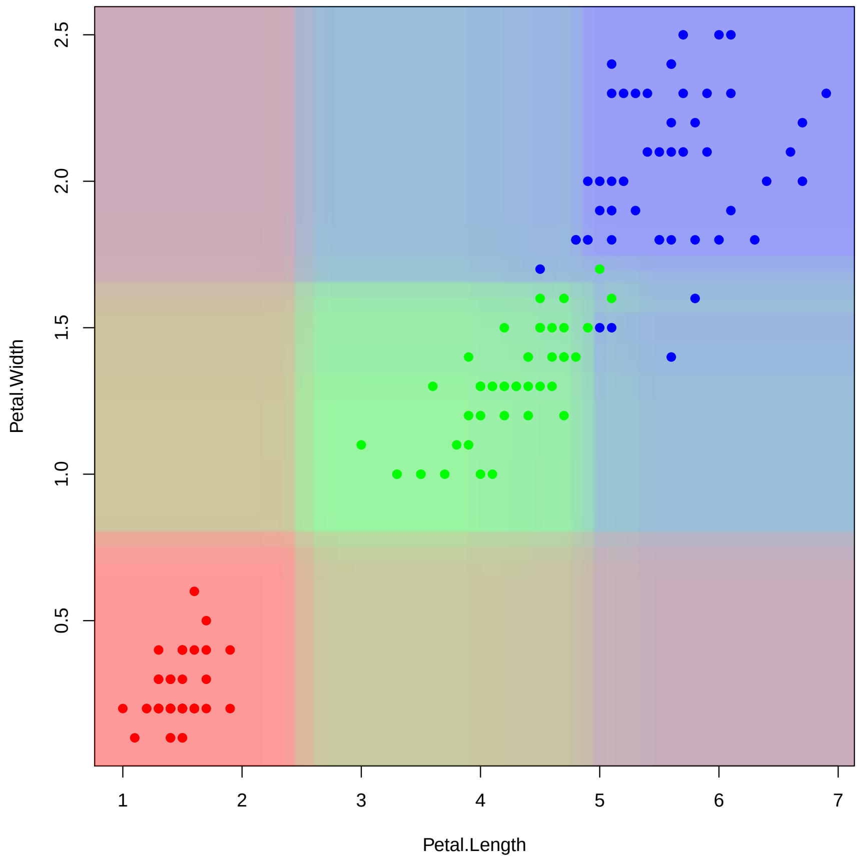
- Ensemble methods are quite common in machine learning:
  - Instead of a single model, multiple models are trained.
  - When making predictions, the results of these models are aggregated (e.g. averaged, voting, etc.).
- The motivation of aggregating multiple models is to reduce variance, i.e. to avoid overfitting.
- *Random forests* are ensembles of decision trees.

RANDOM FORESTS: MOST COMMON VARIANT

- Use CART (Classification and Regression Trees) for training the single trees, i.e. binary splits with Gini impurity gain (for classification) / variance reduction (for regression) as splitting criterion.
- For each tree, samples are chosen randomly from the training set (typically with replacement).
- For each split, only a sub-sample of randomly chosen features is considered.
- Trees are grown to full size and not pruned.

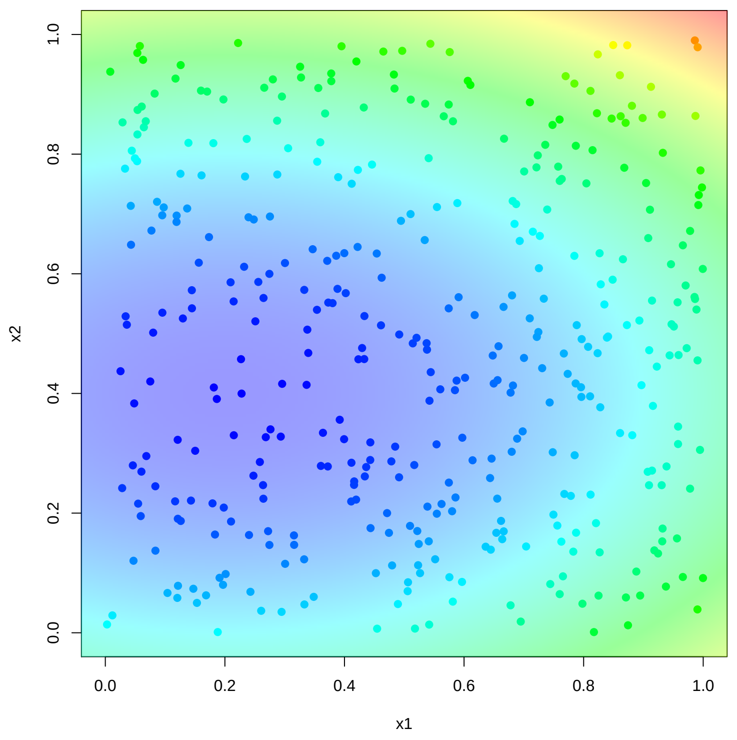
EXAMPLE: IRIS DATA SET

(1000 TREES)

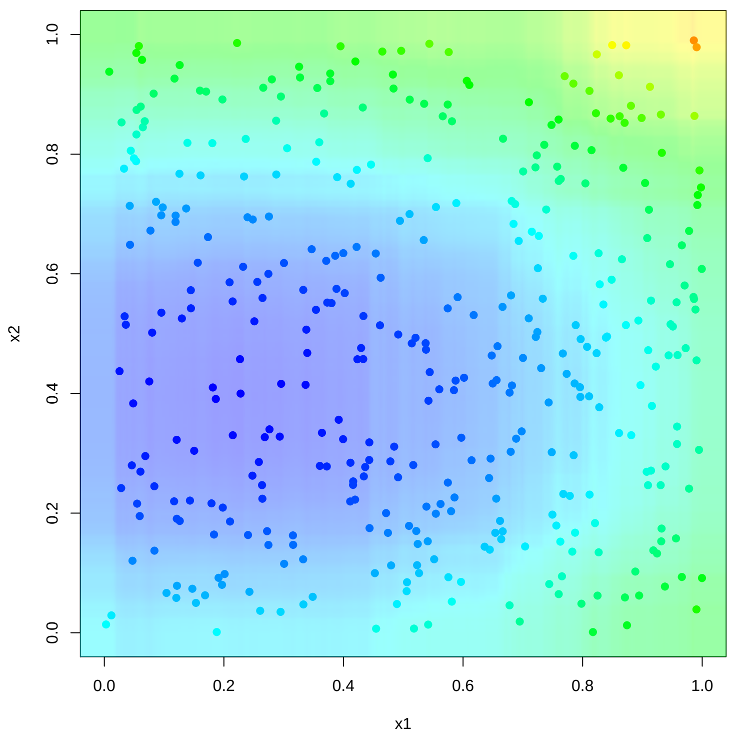


EXAMPLE: REGRESSION DATA SET

True function:



Prediction (1000 trees):



RANDOM FORESTS VS. OVERFITTING

- It can be proved (cf. Breiman) that random forests do not tend to overfit if the number of trees increases.
- This, however, does not mean that random forests cannot overfit generally. Like for any other machine learning method, training errors can be much smaller than the real generalization error (test error), especially if the number of trees is not large.

OUT-OF-BAG ESTIMATES

- Random forests allow for assessing the generalization performance on the basis of training data only.
- For each sample, the error can be computed by considering only those trees that have not used this sample in their training sub-sample.
- Then the overall out-of-bag error can be computed by averaging the out-of-bag errors of all samples.

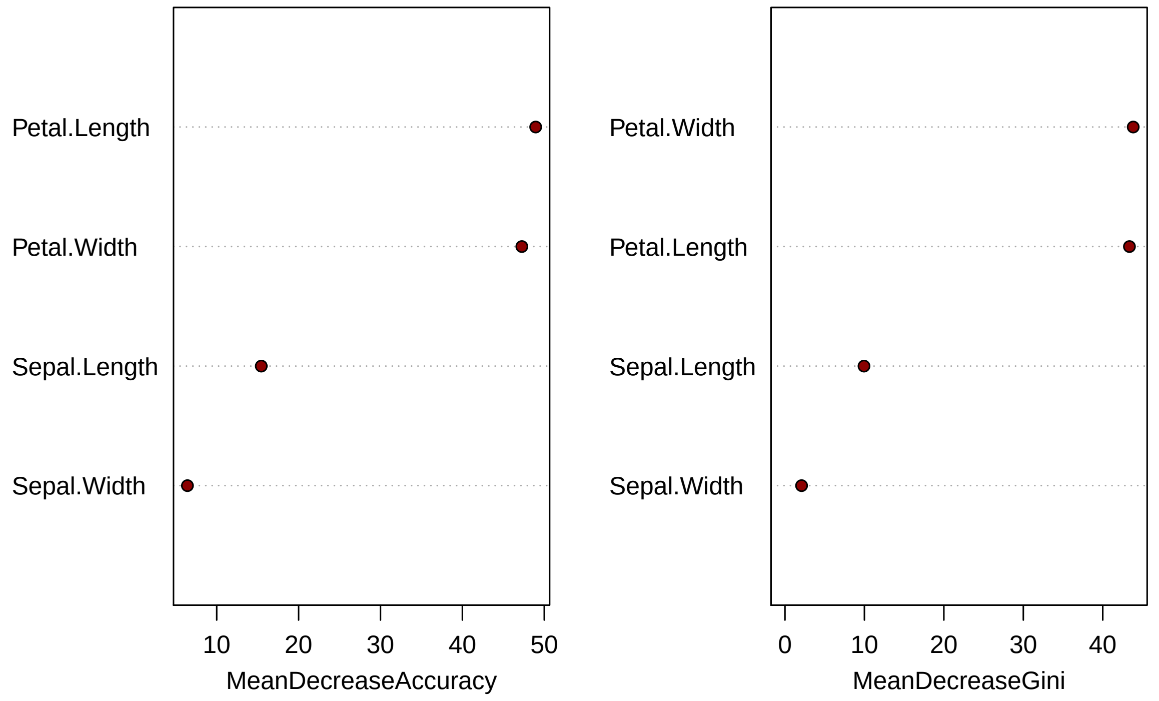
MEASURING VARIABLE IMPORTANCE

Mean Gini impurity decrease: for all features, average the Gini impurity gains of all splits in all trees that involve this feature;

Mean accuracy decrease:

1. Compute out-of-bag error for each sample.
2. For each feature separately, consider random permutations and compute the out-of-bag errors for the data set with the permuted feature.
3. Then the importance score is computed by averaging the differences before and after permuting the feature (upon normalization by the standard deviation of the differences).

EXAMPLE: VARIABLE IMPORTANCES FOR IRIS DATA SET (1000 TREES)



STRATIFICATION FOR UNBALANCED CLASS DISTRIBUTIONS

- The sub-samples used for training the individual trees are as (un)balanced as the original data set.
- Therefore, larger classes will receive more emphasis by the final random forest.
- A simple strategy to equalize the importance of classes and, thereby, to improve balanced accuracy instead of standard accuracy is to *stratify the classes* when sampling.
- **Example:** suppose we have a data set with 1000 samples, 900 of which are negative and 100 of which are positive. Then we can always draw 100 negative and 100 positive samples for training trees.

CONCLUDING REMARKS

- Random forests are a very powerful machine learning tools.
- Advantages:
  - $+$ Only few hyperparameters and easy to use
  - $+$ Built-in feature selection ($+$ variable importance measures)
  - $+$ Possibility to correct for unbalanced class distributions
  - $+$ Relatively robust against overfitting
  - $+$ Scale-invariant
  - $+$ Training and predictions can be parallelized

CONCLUDING REMARKS (cont'd)

- Disadvantages:
  - Computationally expensive
  - Not very interpretable (unlike single trees)
- That they are robust against overfitting does *not* mean that the training error is an estimate for the generalization error! Use out-of-bag errors or the test set method / cross validation to assess the generalization performance of random forests!