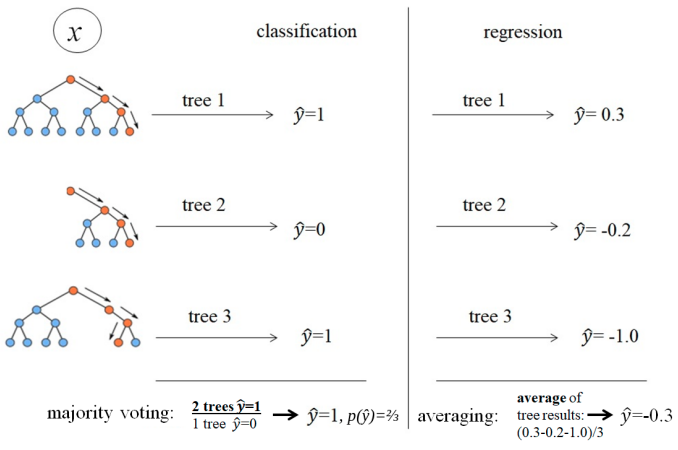
Einführung in das statistische Lernen

Random Forests: Bagging Ensembles



Learning goals

- Understand the basic idea of bagging
- Be able to explain the connection of bagging and bootstrap
- Understand how a prediction is computed for bagging
- Understand why bagging improves the predictive power

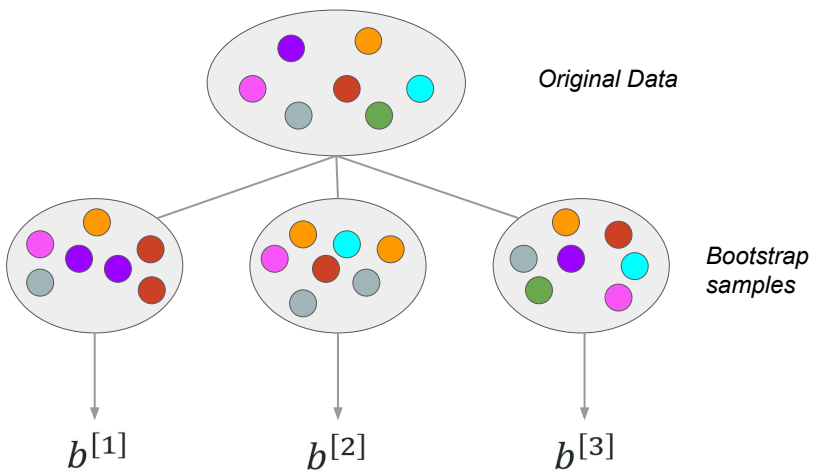
BAGGING

- Bagging is short for **B**ootstrap **Agg**regation.
- It's an **ensemble method**, i.e., it combines many models into one big "meta-model"
- Such model ensembles often work much better than their members alone would.
- The constituent models of an ensemble are called **base learners**

BAGGING

In a **bagging** ensemble, all base learners are of the same type. The only difference between the models is the data they are trained on. Specifically, we train base learners $b^{[m]}(\mathbf{x}), m = 1, \ldots, M$ on M **bootstrap** samples of training data $\mathcal{D}$:

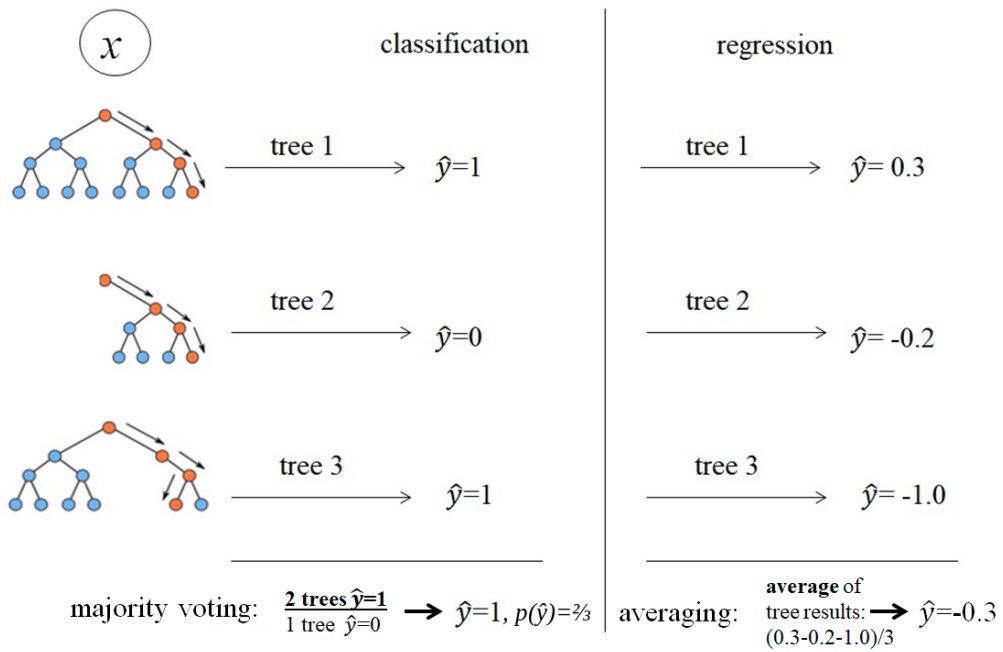
- Draw n observations from $\mathcal{D}$ with replacement
- Fit the base learner on each of the M bootstrap samples to get models $\hat{f}(x) = \hat{b}^{[m]}(\mathbf{x}), m = 1, \ldots, M$



BAGGING

Aggregate the predictions of the M fitted base learners to get the **ensemble model** $\hat{f}^{[M]}(\mathbf{x})$:

- Aggregate via averaging (regression) or majority voting (classification)
- Posterior class probabilities $\hat{\pi}_k(\mathbf{x})$ can be estimated by calculating predicted class frequencies over the ensemble



WHY/WHEN DOES BAGGING HELP?

In one sentence:

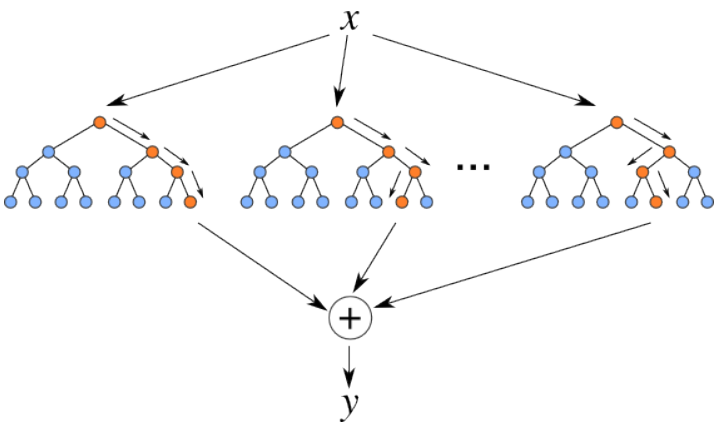
Because the variability of the average of the predictions of many base learner models is smaller than the variability of the predictions from one such base learner model.
If the error of a base learner model is mostly due to (random) variability and not due to structural reasons, combining many such base learners by bagging helps reducing this variability.

BAGGING: SYNOPSIS

- Basic idea: fit the same model repeatedly on many **bootstrap** replications of the training data set and **aggregate** the results
- Gains performance by reducing the variance of predictions, but (slightly) increases the bias: it reuses training data many times, so small mistakes can get amplified.
- Works best for unstable/high-variance base learners, where small changes in the training set can cause large changes in predictions: e.g., CART, neural networks, step-wise/forward/backward variable selection for regression
- Works best if base learners' predictions are only weakly correlated: they don't all make the same mistakes.
- Can degrade performance for stable methods like k-NN, LDA, Naive Bayes, linear regression

Einführung in das statistische Lernen

Random Forest: Introduction



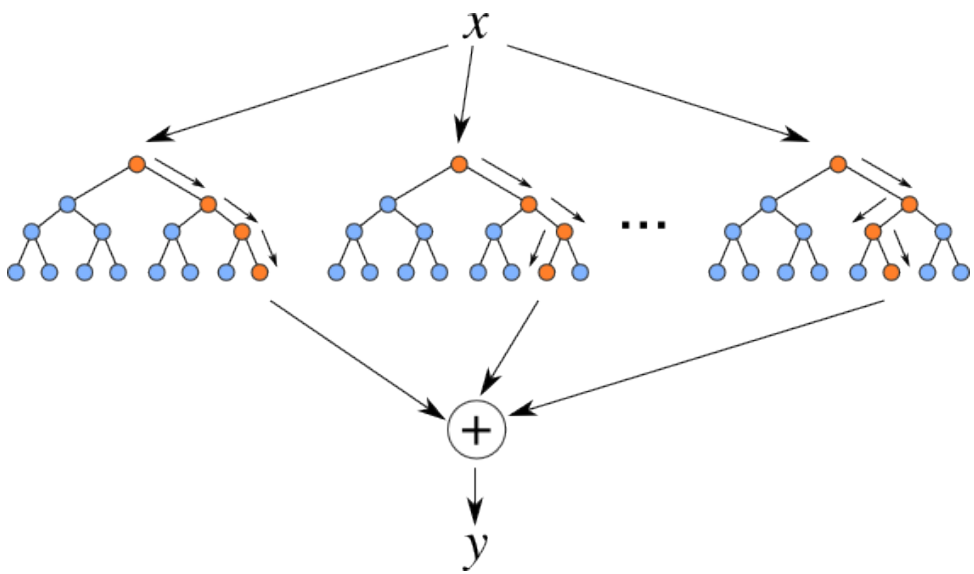
Learning goals

- Know how random forests are defined by extending the idea of bagging
- Understand that the goal is to decorrelate the trees
- Understand that the out-of-bag error is a way to obtain unbiased estimates of the generalization error during training

RANDOM FORESTS

Modification of bagging for trees proposed by Breiman (2001):

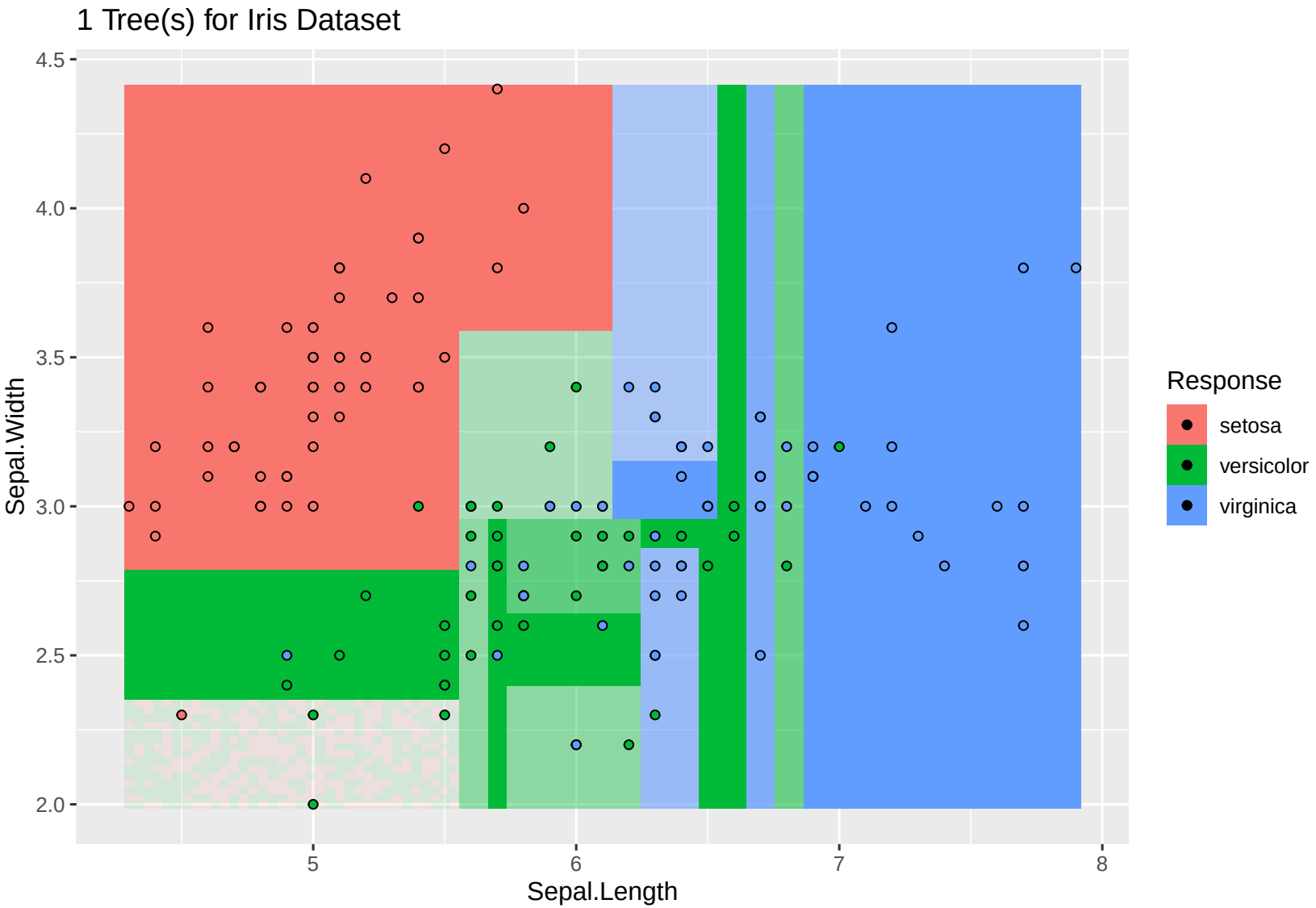
- Tree base learners on bootstrap samples of the data
- Uses **decorrelated** trees by randomizing splits (see below)
- Tree base learners are usually fully expanded, without aggressive early stopping or pruning, to **increase variance of the ensemble**



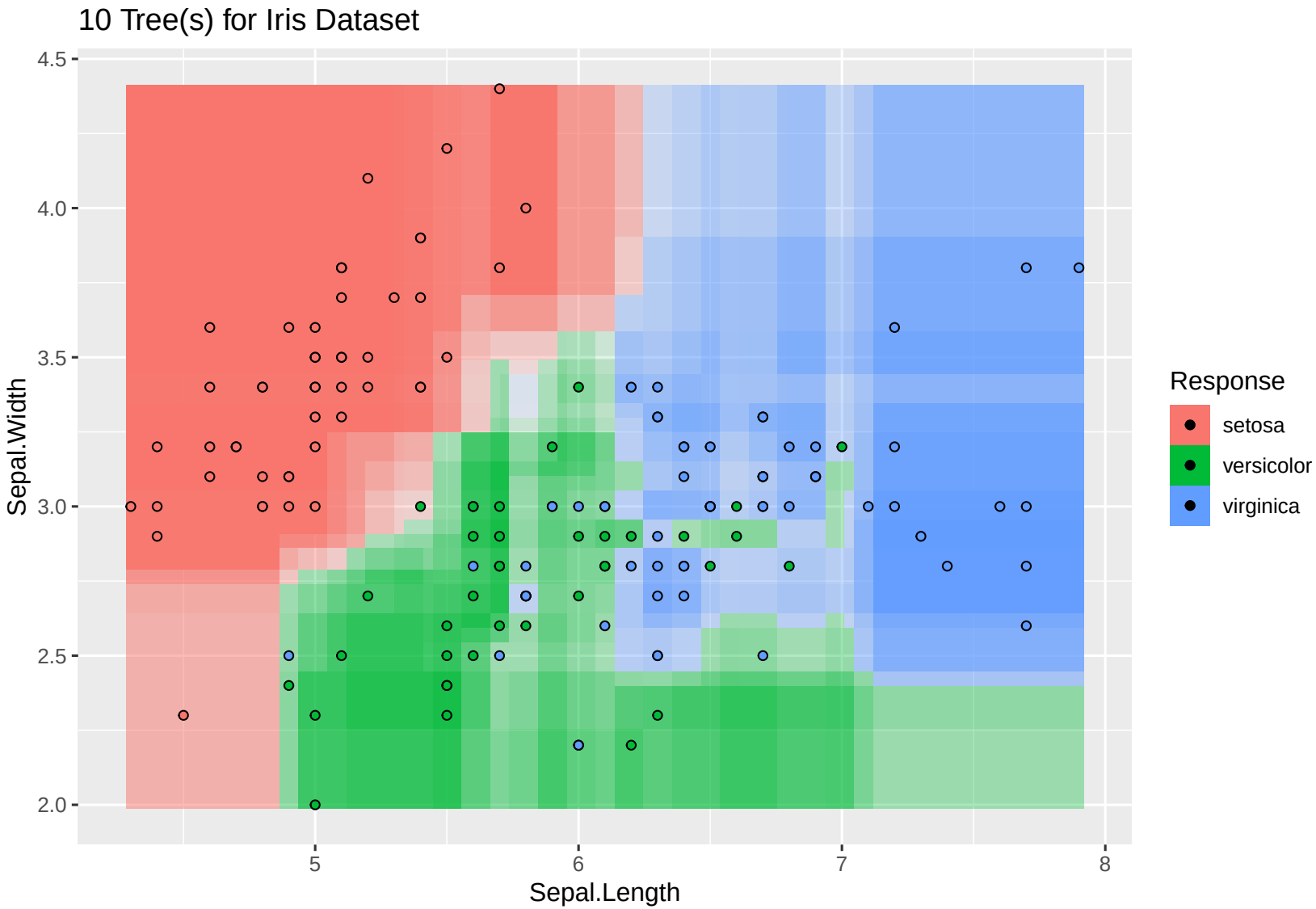
RANDOM FEATURE SAMPLING

- From our analysis of bagging risk we can see that decorrelating trees improves the ensemble
- Simple randomized approach:
  At each node of each tree, randomly draw mtry $\leq p$ candidate features to consider for splitting. Recommended values:
  - Classification: mtry $= \lfloor \sqrt{p} \rfloor$
  - Regression: mtry $= \lfloor p/3 \rfloor$

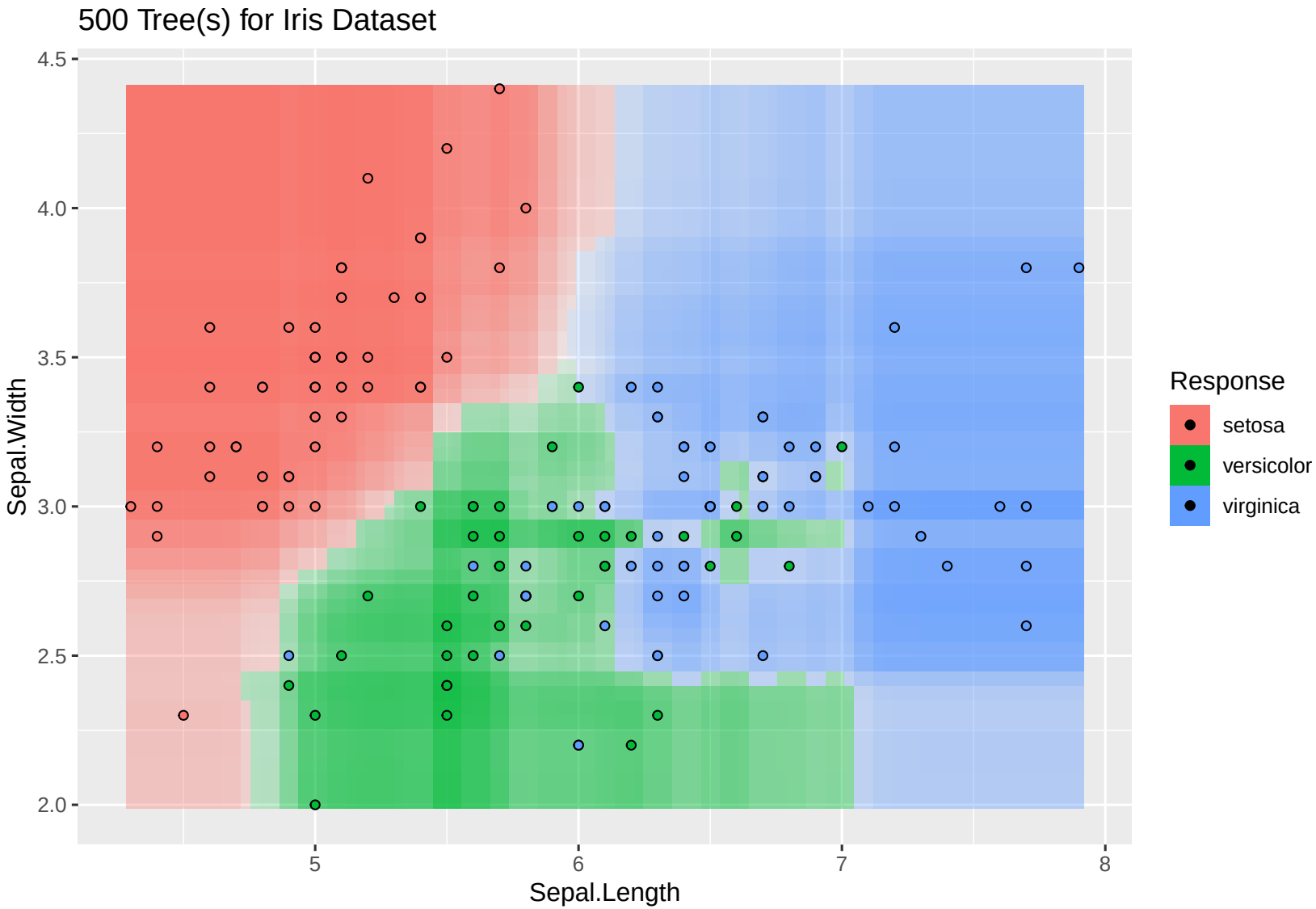
EFFECT OF ENSEMBLE SIZE



EFFECT OF ENSEMBLE SIZE

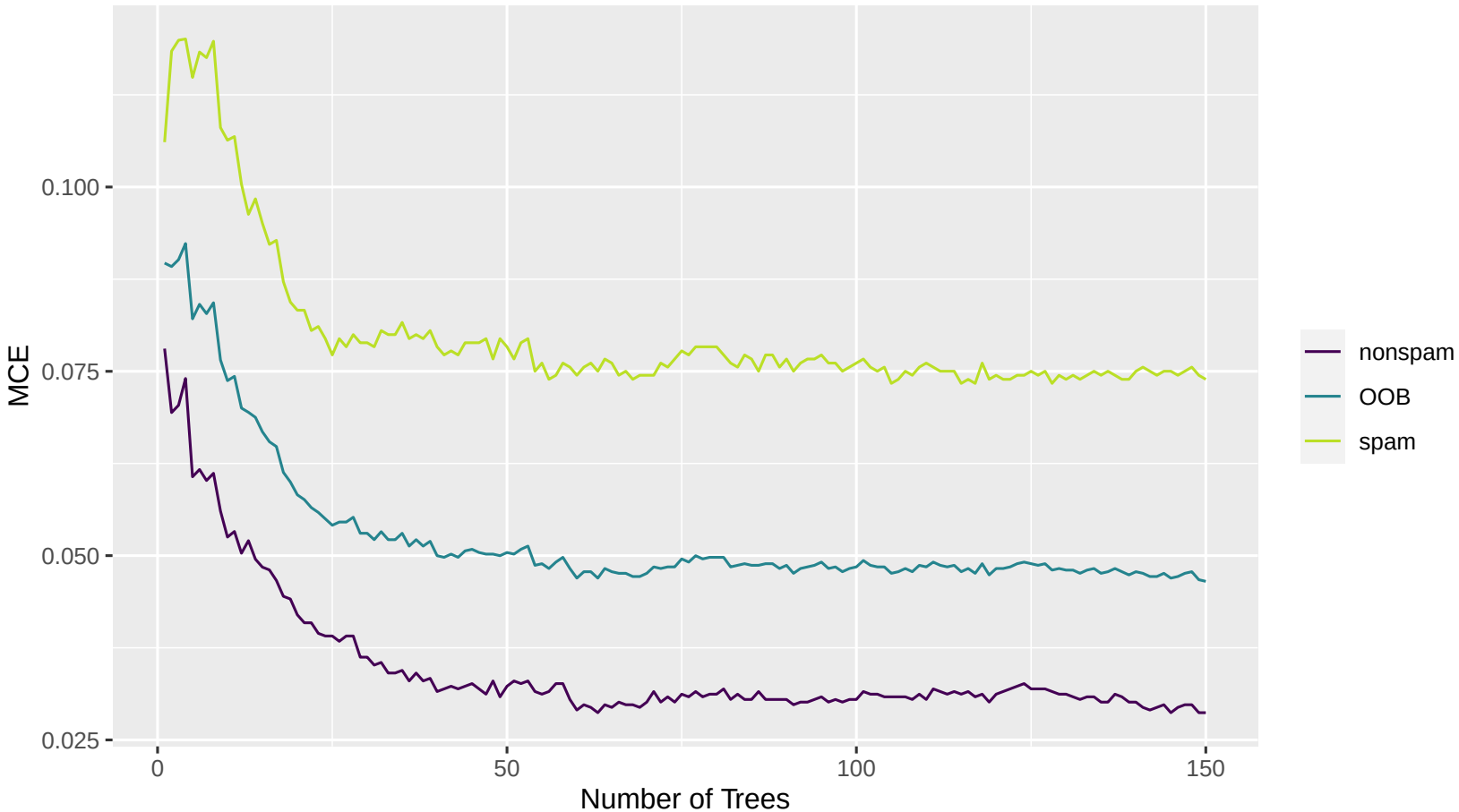


EFFECT OF ENSEMBLE SIZE

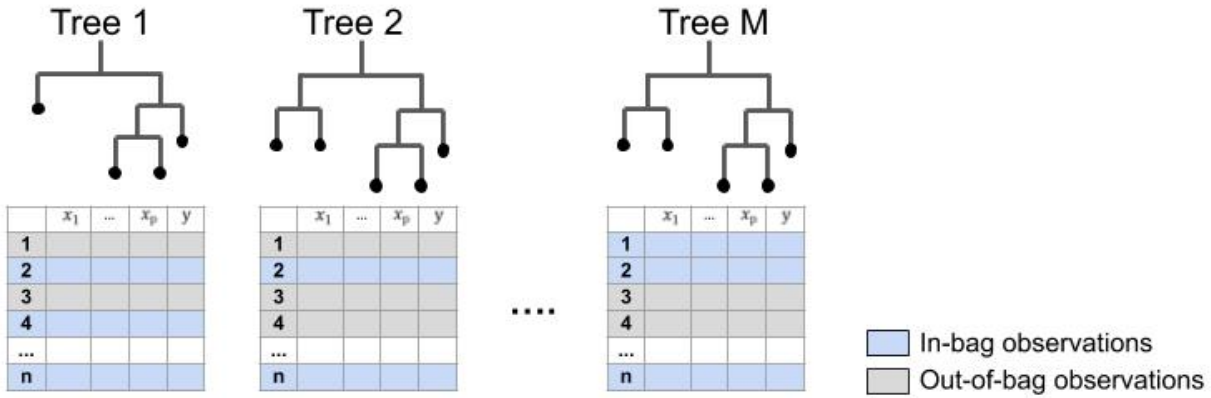


OUT-OF-BAG ERROR ESTIMATE

With the RF it is possible to obtain unbiased estimates of the generalization error directly during training, based on the out-of-bag observations for each tree:



OUT-OF-BAG ERROR ESTIMATE



- For an estimation of the generalization error, we exploit the fact that the i-th observation acts as unseen test point for all trees in which it is OOB.
- Let $\text{OOB}^{[m]}$ denote the index set $\left\{ i \in \{1, \ldots, n\} | (\mathbf{x}^{(i)}, y^{(i)}) \text{ is OOB for } b^{[m]}(\mathbf{x}) \right\}$.
- The number of trees for which the i-th observation is OOB is then given by $S_{\text{OOB}}^{(i)} = \sum_{m=1}^{M} \mathbb{I}(i \in \text{OOB}^{[m]})$.
- We can compute the average over predictions $\hat{y}^{(i)[m]}$ from trees $b^{[m]}(\mathbf{x})$ that have observation i in their OOB data to obtain an ensemble prediction.
- The average loss of these ensemble OOB predictions over all n observations yields an estimate for the generalization error.

OUT-OF-BAG ERROR ESTIMATE

- Compute the ensemble OOB prediction for each observation:

$$\hat{y}_{\text{OOB}}^{(i)} = \begin{cases} \frac{1}{S_{\text{OOB}}^{(i)}} \sum_{m=1}^{M} \mathbb{I}(i \in \text{OOB}^{[m]}) \cdot \hat{y}^{(i)[m]} & \text{in regression,} \\ \arg\max_{k \in \{1,\ldots,g\}} \frac{\sum_{m=1}^{M} \mathbb{I}(i \in \text{OOB}^{[m]}) \cdot \mathbb{I}(\hat{h}^{(i)[m]} = k)}{S_{\text{OOB}}^{(i)}} & \text{in classification.} \end{cases}$$

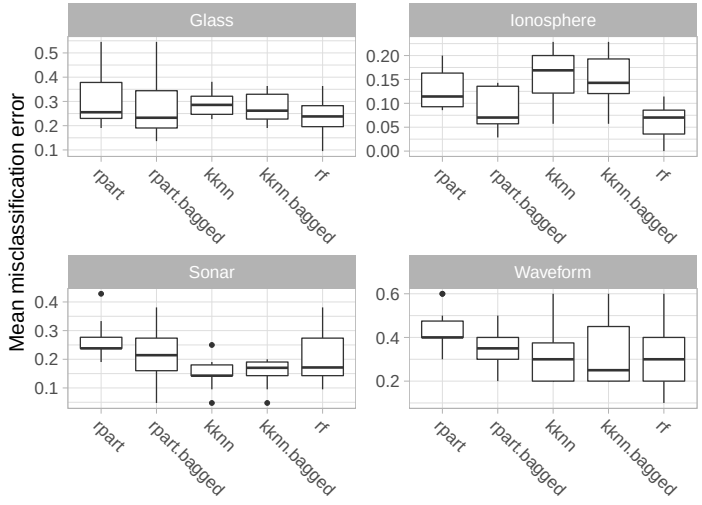
- Then, take the average of the resulting point-wise losses to estimate the OOB error of the forest:

$$\widehat{\text{err}}_{\text{OOB}} = \frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, \hat{y}_{\text{OOB}}^{(i)})$$

- Note that the use of class labels commands the use of 0-1 loss in classification (alternative formulations for other losses are possible).
- OOB size: $\mathbb{P}(i \in \text{OOB}^{[m]}) = \left(1 - \frac{1}{n}\right)^n \overset{n \to \infty}{\longrightarrow} \frac{1}{e} \approx 0.37$ for $i \in \{1, \ldots, n\}$.
- Similar to 3-CV, can be used for a quick model selection.

Einführung in das statistische Lernen

Random Forest: Benchmarking Trees, Forests, and Bagging K-NN



Learning goals

- Understand for which kind of learners bagging can improve predictive power

BENCHMARK: RANDOM FOREST VS. (BAGGED) CART VS. (BAGGED) K-NN

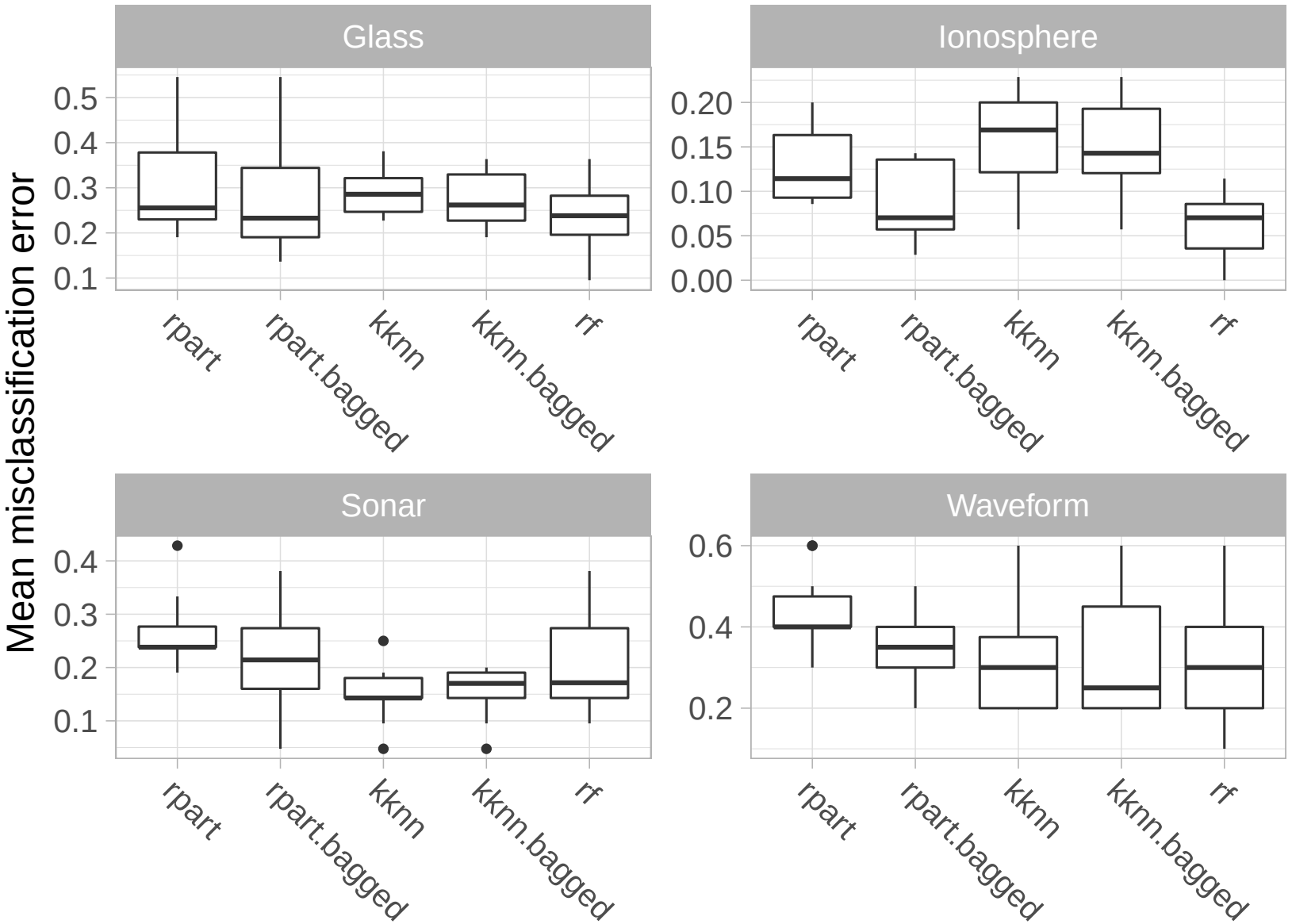
- Goal: Compare performance of random forest against (bagged) stable and (bagged) unstable methods
- Algorithms:
  - classification tree (CART, implemented in `rpart`, `max.depth`: 30, `min.split`: 20, `cp`: 0.01)
  - bagged classification tree using 50 bagging iterations (`bagged.rpart`)
  - k-nearest neighbors (k-NN, implemented in `kknn`, $k = 7$)
  - bagged k-nearest neighbors using 50 bagging iterations (`bagged.knn`)
  - random forest with 50 trees (implemented in `randomForest`)
- Method to evaluate performance: 10-fold cross-validation
- Performance measure: mean misclassification error on test sets

BENCHMARK: RANDOM FOREST VS. (BAGGED) CART VS. (BAGGED) K-NN

- Datasets from **mlbench**:

Name	Kind of data	n	p	Task
Glass	Glass identification data	214	10	Predict the type of glass (6 levels) on the basis of the chemical analysis of the glasses represented by the 10 features
Ionosphere	Radar data	351	35	Predict whether the radar returns show evidence of some type of structure in the ionosphere (“good”) or not (“bad”)
Sonar	Sonar data	208	61	Discriminate between sonar signals bounced off a metal cylinder (“M”) and those bounced off a cylindrical rock (“R”)
Waveform	Artificial data	100	21	Simulated 3-class problem which is considered to be a difficult pattern recognition problem. Each class is generated by the waveform generator.

BENCHMARK: RANDOM FOREST VS. (BAGGED) CART VS. (BAGGED) K-NN



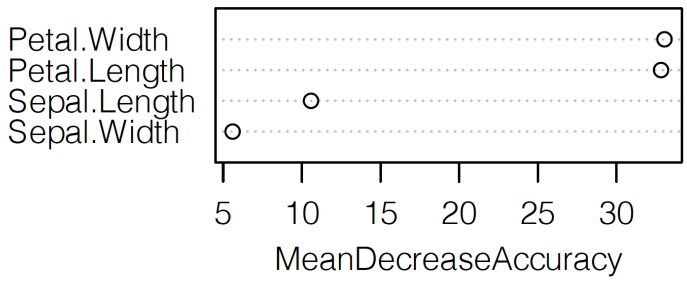
BENCHMARK: RANDOM FOREST VS. (BAGGED) CART VS. (BAGGED) K-NN

Bagging k-NN does not improve performance because:

- k-NN is stable w.r.t. perturbations
- In a 2-class problem, nearest-neighbor-based classification only changes under bagging if both
  - the nearest neighbor in the learning set is **not** in at least half of the bootstrap samples, but the probability that any given observation is in the bootstrap sample is 63%, which is greater than 50%,
  - and, simultaneously, the *new* nearest neighbor(s) all have a different label than the missing nearest neighbor in those bootstrap samples, which is unlikely for most regions of $\mathcal{X} \times \mathcal{Y}$.

Einführung in das statistische Lernen

Random Forests: Feature Importance



Learning goals

- Understand that the goal of defining variable importance is to enhance interpretability of the random forest
- Know definition of variable importance based on improvement in split criterion
- Know definition of variable importance based on permutations of OOB observations

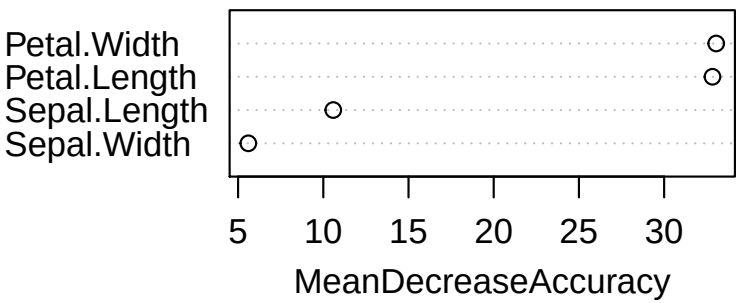
VARIABLE IMPORTANCE

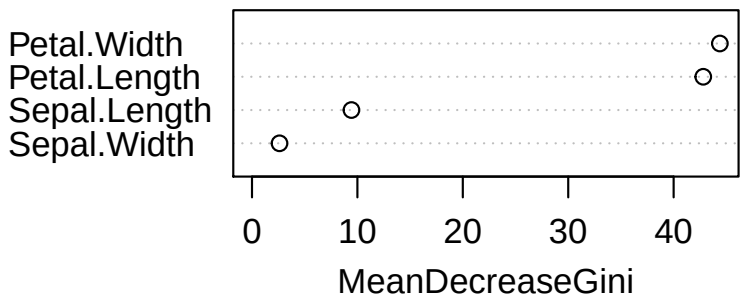
- Single trees are highly interpretable
- Random forests as ensembles of trees lose this feature
- Contributions of the different features to the model are difficult to evaluate
- Way out: variable importance measures
- Basic idea: by how much would the performance of the random forest decrease if a specific feature were removed or rendered useless?

VARIABLE IMPORTANCE

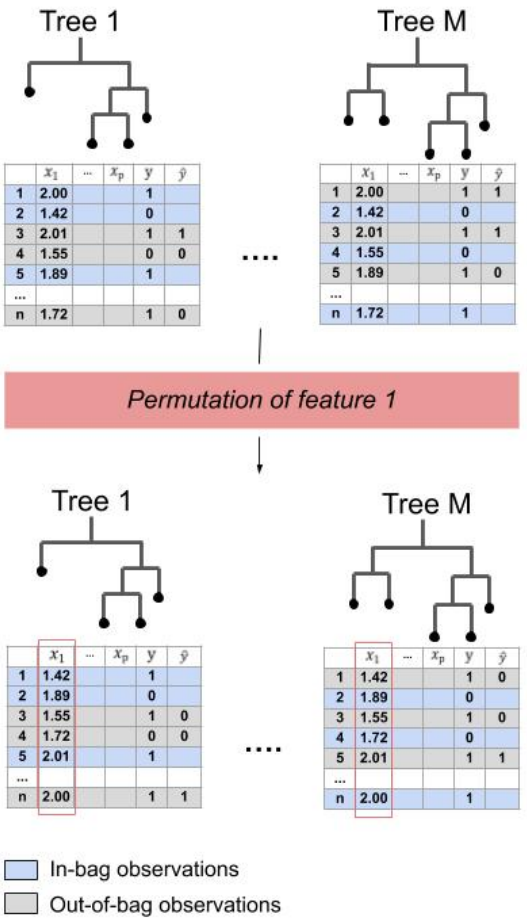
Measure based on improvement in split criterion

for features x_j, $j = 1$ to p **do**
  for tree base learners $\hat{b}^{[m]}(\mathbf{x})$, $m = 1$ to M **do**
    Find all nodes $\mathcal{N}$ in $\hat{b}^{[m]}(\mathbf{x})$ that use x_j.
    Compute improvement in splitting criterion achieved by them.
    Add up these improvements.
  end for
  Add up improvements over all trees to get feature importance of x_j.
end for





VARIABLE IMPORTANCE



Measure based on permutations of OOB observations

Estimate OOB error $\widehat{\text{err}}_{\text{OOB}}$.

for features x_j, $j = 1$ to p **do**

Perform permutation ψ_j on x_j to distort feature-target relation for x_j.

for distorted observations $(\mathbf{x}^{(i)}_{\psi_j}, y^{(i)})$, $i = 1$ to n **do**

Compute OOB prediction $\hat{y}^{(i)}_{\text{OOB},\psi_j}$.

Compute corresponding loss $L(y^{(i)}, \hat{y}^{(i)}_{\text{OOB},\psi_j})$.

end for

Estimate importance of j-th variable

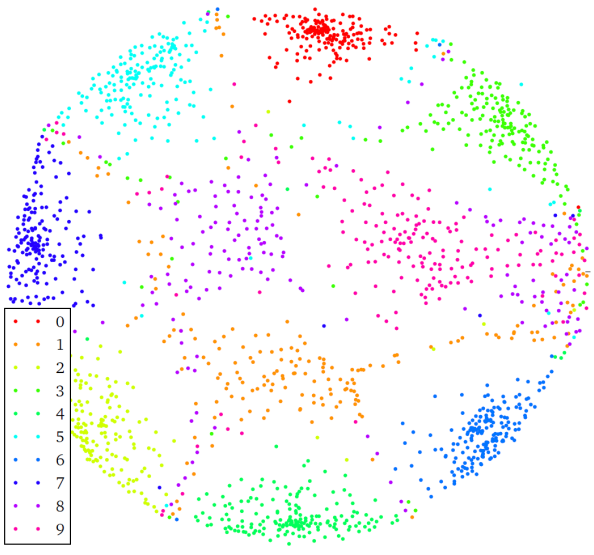
$$\widehat{\text{VI}}_j = \widehat{\text{err}}_{\text{OOB},\psi_j} - \widehat{\text{err}}_{\text{OOB}}$$

$$= \frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, \hat{y}^{(i)}_{\text{OOB},\psi_j}) - \widehat{\text{err}}_{\text{OOB}}.$$

end for

Einführung in das statistische Lernen

Random Forests: Proximities



Proximity plot for a 10-class handwritten digit classification task.

Learning goals

- Understand how a random forest can be used to define proximities of observations
- Know how proximities can be used for missing data, outliers, mislabeled data and a visualization of the forest

RANDOM FOREST PROXIMITIES

- One of the most useful tools in random forests
- A measure of similarity ("closeness" or "nearness") of observations derived from random forests
- Can be calculated for each pair of observations
- Definition:
  - The proximity between two observations $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ is calculated by measuring the number of times that these two observations are placed in the same terminal node of the same tree of the random forest, divided by the number of trees in the forest
  - The proximity of observations $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ can be written as $\text{prox}\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)$
  - The proximities form an intrinsic similarity measure between pairs of observations
- The proximities of all observations form a symmetric $n \times n$ matrix.

RANDOM FOREST PROXIMITIES

- Algorithm:
  - Once a random forest has been trained, all of the training data is put through each tree (both in- and out-of-bag).
  - Every time two observations $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ end up in the same terminal node of a tree, their proximity is increased by one.
  - Once all data has been put through all trees and the proximities have been counted, the proximities are normalized by dividing them by the number of trees.

USING RANDOM FOREST PROXIMITIES

- Imputing missing data:
  1. Replace missing values for a given variable using the median of the non-missing values
  2. Get proximities
  3. Replace missing values in observation $\mathbf{x}^{(i)}$ by a weighted average of non-missing values, with weights proportional to the proximity between observation $\mathbf{x}^{(i)}$ and the observations with the non-missing values

  Steps 2 and 3 are then iterated a few times.
- Locating outliers:
  - An outlier is an observation whose proximities to all other observations are small
  - Measure of outlyingness can be computed for each observation in the training sample

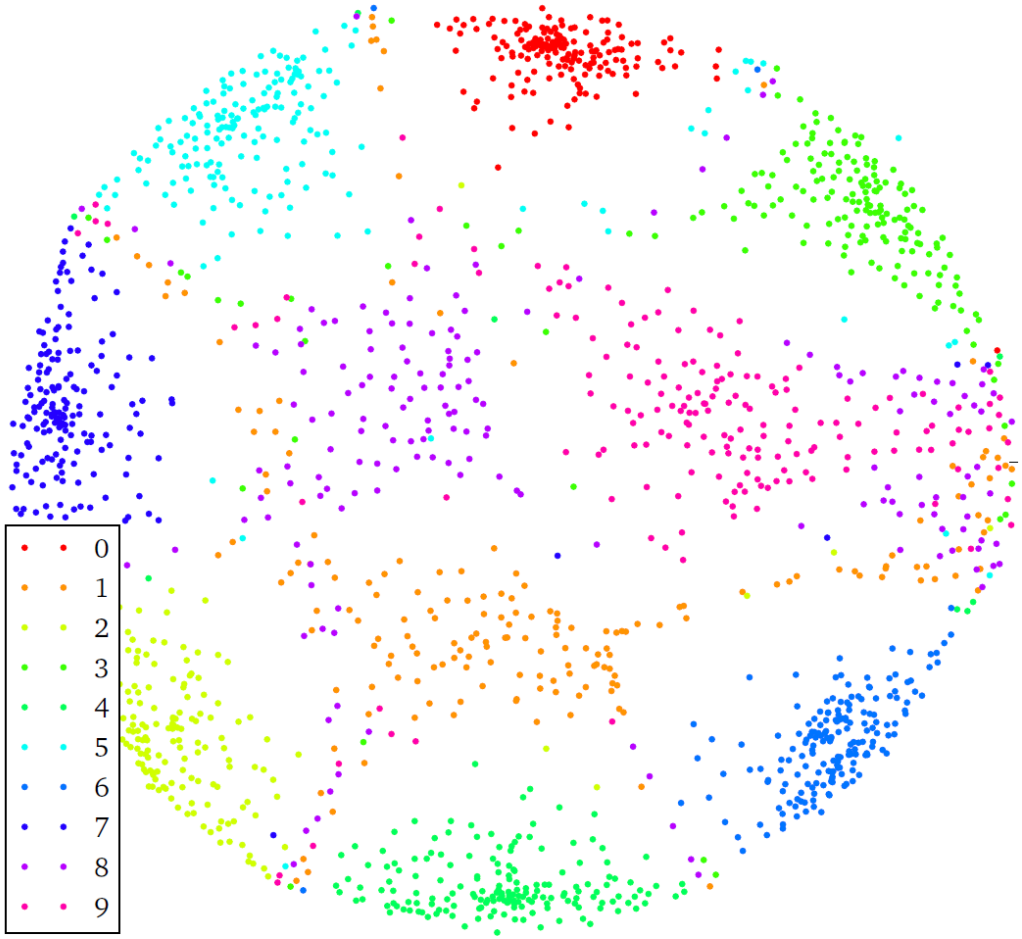
USING RANDOM FOREST PROXIMITIES

- If the measure is unusually large, the observation should be carefully inspected

- Identifying mislabeled data:
  - Instances in the training data set are sometimes labeled ambiguously or incorrectly, especially in “manually” created data sets.
  - Proximities can help in finding them: they often show up as outliers in terms of their proximity values.

- Visualizing the forest:
  - The values $1 - \text{prox}\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)$ can be thought of as distances in a high-dimensional space
  - They can be projected onto a low-dimensional space using metric multidimensional scaling (MDS)
  - Metric multidimensional scaling uses eigenvectors of a modified version of the proximity matrix to get scaling coordinates

USING RANDOM FOREST PROXIMITIES



Proximity plot for a 10-class handwritten digit classification task.

image from G. Louppe (2014) *Understanding Random Forests* `arXiv:1407.7502`.

USING RANDOM FOREST PROXIMITIES

- The figure depicts the proximity matrix learnt for a 10-class handwritten digit classification task
  - proximity matrix distances projected onto the plane using multidimensional scaling
  - samples from the same class form identifiable clusters, which suggests that they share a common structure
  - also shows the fact for which classes errors occur, e.g., digits 1 and 8 have high within-class variance and have overlaps with other classes

Einführung in das statistische Lernen

Random Forests: Advantages and Disadvantages

Learning goals

- Know advantages and disadvantages of random forests
- Be able to explain random forests in terms of hypothesis space, risk and optimization

RANDOM FOREST: ADVANTAGES

- All advantages of trees also apply to RF: not much preprocessing required, missing value handling, etc.
- Easy to parallelize
- Often works well (enough)
- Integrated variable importance
- Integrated estimation of generalization performance via OOB error
- Works well on high-dimensional data
- Works well on data with irrelevant “noise” variables
- Often not much tuning necessary

RANDOM FOREST: DISADVANTAGES

- Often sub-optimal for regression
- Same extrapolation problem as for trees
- Harder to interpret than trees (but many extra tools are nowadays available for interpreting RFs)
- Implementation can be memory-hungry
- Prediction is computationally demanding for large ensembles

RANDOM FOREST: SYNOPSIS

Hypothesis Space:
Random forest models are (sums of) step functions over rectangular partitions of (subspaces of) $\mathcal{X}$.
Their maximal complexity is controlled by the number of trees in the random forest ensemble and the stopping criteria for the constituent trees.

Risk:
Like trees, random forests can use any kind of loss function for regression or classification.

Optimization:
Exhaustive search over all (randomly selected!) candidate splits in each node of each tree to minimize the empirical risk in the child nodes.
Like all bagging methods, optimization can be done in parallel over the ensemble members.