Classification tree, bagging, and random forest

Pavlo Mozharovskyi[1]

(with contributions of Laurent Rouviere[2] and Valentin Patilea[3])

[1]LTCI, Télécom Paris, Institut Polytechnique de Paris
[2]Université Rennes 2
[3]Ensai, CREST

Machine learning

Paris, March 12, 2022

Today

Classification tree

- Algorithm
- Tuning

Bagging

- Motivation
- Algorithm
- An example

Random forest

- Algorithm
- Interpretation
- Consistency results

Literature

Learning materials include but are not limited to:

- Hastie, T., Tibshirani, R., and Friedman, J. (2009).
  The Elements of Statistics Learning: Data Mining, Inference, and Prediction (Second Edition).
  Springer.
  - Section 8.7.
  - Section 9.2.
  - Chapter 15.

- Slides of the lecture.

- Biau, Devroye, Lugosi (2008).
  Consistency of random forests and other averaging classifiers.
  Journal of Machine Learning Research, 9, 2015–2033.

Binary supervised classification (reminder)

Notation:

- **Given:** for the random pair (X, Y) in $\mathbb{R}^d \times \{0, 1\}$ consisting of a random observation X and its random binary label Y (class), a sample of n i.i.d.: $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)$.
- **Goal:** predict the label of the new (unseen before) observation $\mathbf{x}$.
- **Method:** construct a classification rule:

$$g : \mathbb{R}^d \to \{0, 1\}, \mathbf{x} \mapsto g(\mathbf{x}),$$

  so $g(\mathbf{x})$ is the prediction of the label for observation $\mathbf{x}$.
- **Criterion:** of the performance of g is the **error probability**:

$$R(g) = \mathbb{P}[g(X) \neq Y] = \mathbb{E}[1(g(X) \neq Y)].$$

- **The best solution:** is to know the distribution of (X,Y):

$$g(\mathbf{x}) = 1(\mathbb{E}[Y|X = \mathbf{x}] > 0.5).$$

Contents

Contents

Classification tree (algorithm)

Growing a tree (training)

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.

Classification tree (algorithm)

Growing a tree (training)

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Measure of impurity $Q^{(m)}(T)$ for node m of tree T.

Classification tree (algorithm)

Growing a tree (training)

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Measure of impurity $Q^{(m)}(T)$ for node m of tree T.
- Stopping criteria $S^{(m)}(T)$ for node m of tree T.

Classification tree (algorithm)

Growing a tree (training)

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Measure of impurity $Q^{(m)}(T)$ for node m of tree T.
- Stopping criteria $S^{(m)}(T)$ for node m of tree T.

1. Define the root node by the region $R^{(0)}$ containing the entire sample, set $m = 0$.

Classification tree (algorithm)

Growing a tree (training)

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Measure of impurity $Q^{(m)}(T)$ for node m of tree T.
- Stopping criteria $S^{(m)}(T)$ for node m of tree T.

1. Define the root node by the region $R^{(0)}$ containing the entire sample, set $m = 0$.
2. If $S^{(m)}(T)$ is fulfilled then stop for this node (e.g., a lower bound for the # of obs. in a node).

Classification tree (algorithm)

Growing a tree (training)

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Measure of impurity $Q^{(m)}(T)$ for node m of tree T.
- Stopping criteria $S^{(m)}(T)$ for node m of tree T.

1. Define the root node by the region $R^{(0)}$ containing the entire sample, set $m = 0$.
2. If $S^{(m)}(T)$ is fulfilled then stop for this node (e.g., a lower bound for the # of obs. in a node).
3. Find a variable and a split (one-variable threshold) diving node region $R^{(m)}$ into two nodes with subregions $R^{(m_L)}$ and $R^{(m_R)}$ to minimize $Q^{(m)}(T)$.

Classification tree (algorithm)

Growing a tree (training)

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Measure of impurity $Q^{(m)}(T)$ for node m of tree T.
- Stopping criteria $S^{(m)}(T)$ for node m of tree T.

1. Define the root node by the region $R^{(0)}$ containing the entire sample, set $m = 0$.
2. If $S^{(m)}(T)$ is fulfilled then stop for this node (e.g., a lower bound for the # of obs. in a node).
3. Find a variable and a split (one-variable threshold) diving node region $R^{(m)}$ into two nodes with subregions $R^{(m_L)}$ and $R^{(m_R)}$ to minimize $Q^{(m)}(T)$.
4. Repeat steps 2–3 for all leaves until global stopping.

Classification tree (algorithm)

Growing a tree (training)

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Measure of impurity $Q^{(m)}(T)$ for node m of tree T.
- Stopping criteria $S^{(m)}(T)$ for node m of tree T.

1. Define the root node by the region $R^{(0)}$ containing the entire sample, set $m = 0$.
2. If $S^{(m)}(T)$ is fulfilled then stop for this node (e.g., a lower bound for the # of obs. in a node).
3. Find a variable and a split (one-variable threshold) diving node region $R^{(m)}$ into two nodes with subregions $R^{(m_L)}$ and $R^{(m_R)}$ to minimize $Q^{(m)}(T)$.
4. Repeat steps 2–3 for all leaves until global stopping.

Output: The tree T.

Classification tree – Descending the tree

- *Descend the tree* until a terminal node.

Classification tree – Descending the tree

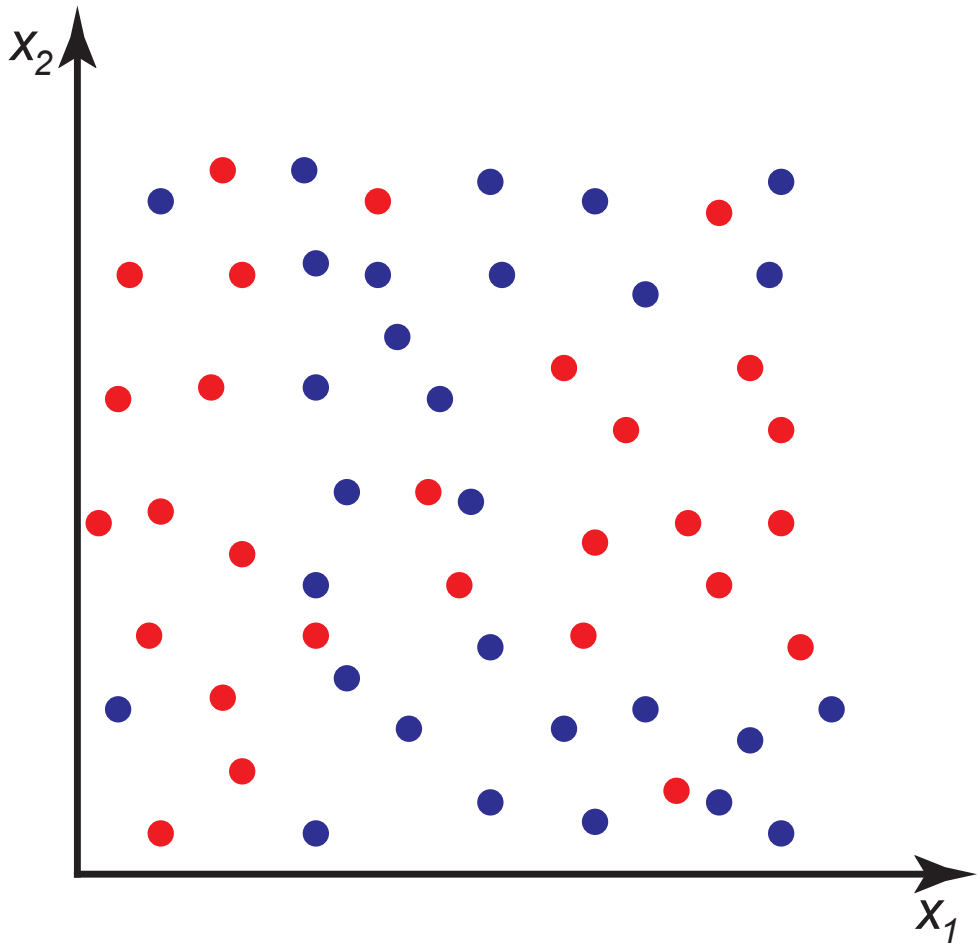
- *Descend the tree* until a terminal node.
- In each node m, classify the observations by choosing the *majority class*.

Classification tree – Descending the tree

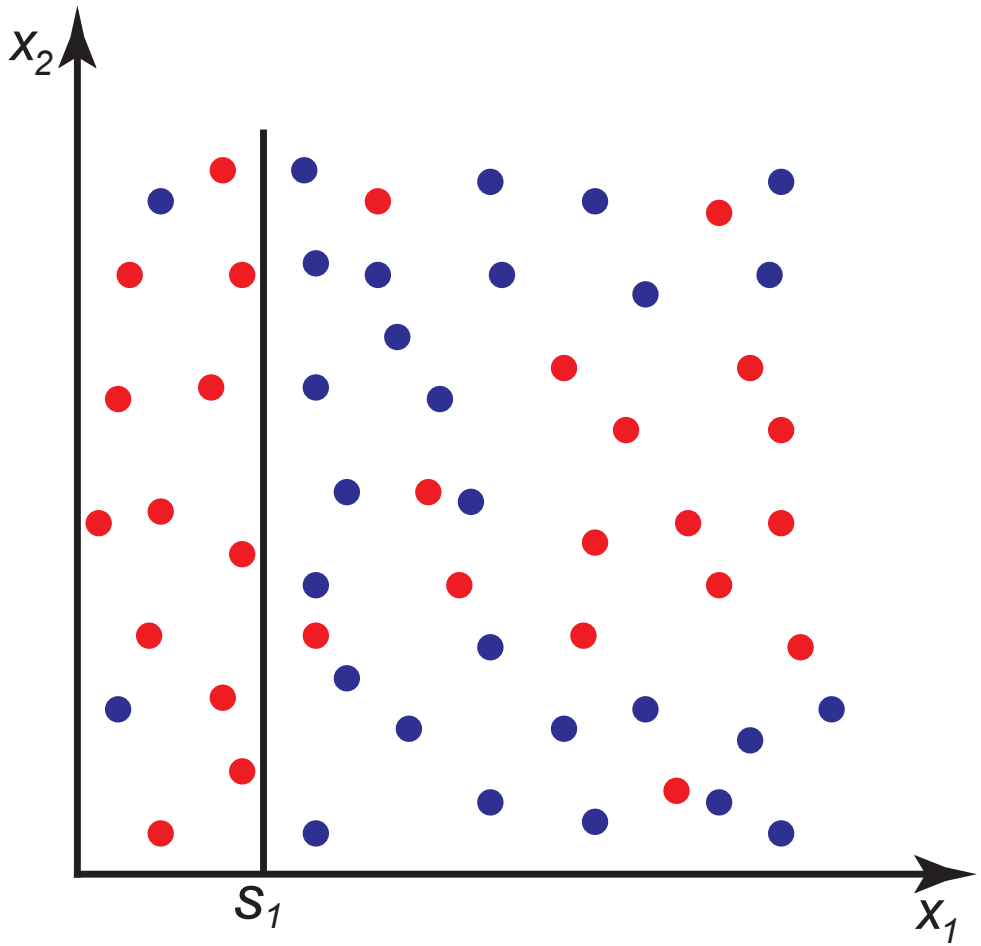
- *Descend the tree* until a terminal node.
- In each node m, classify the observations by choosing the *majority class*.
- That is, in node m *classify* the observations to class $c(m)$:

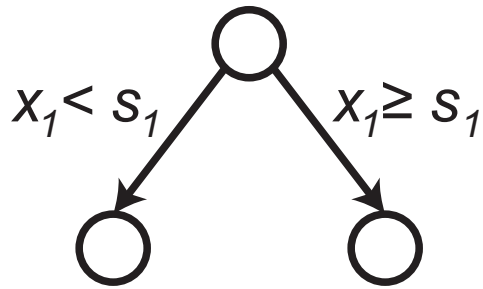
$$c(m) = \underset{k \in \{0,1\}}{\arg\max} \sum_{i \in R^{(m)}} I(y_i = k)\,.$$

Classification tree (illustration)

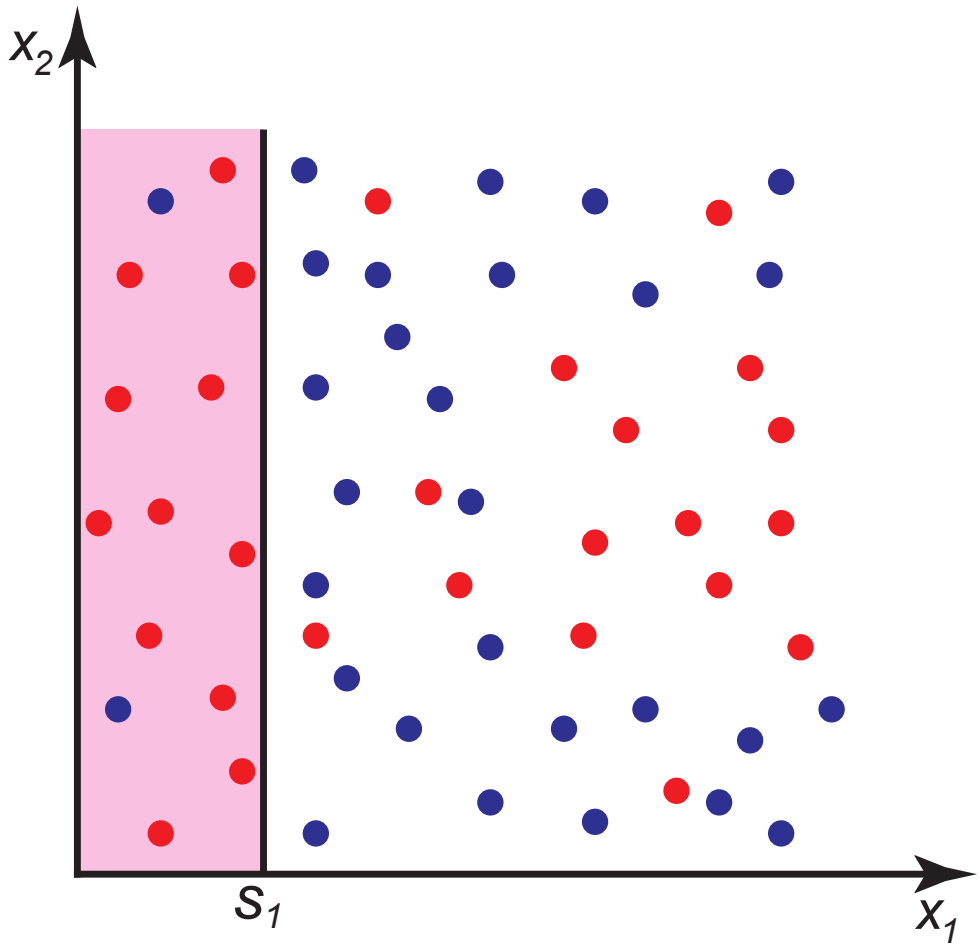


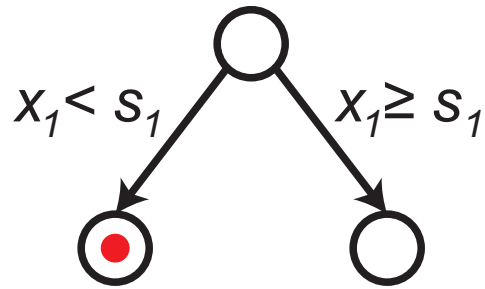
Classification tree (illustration)



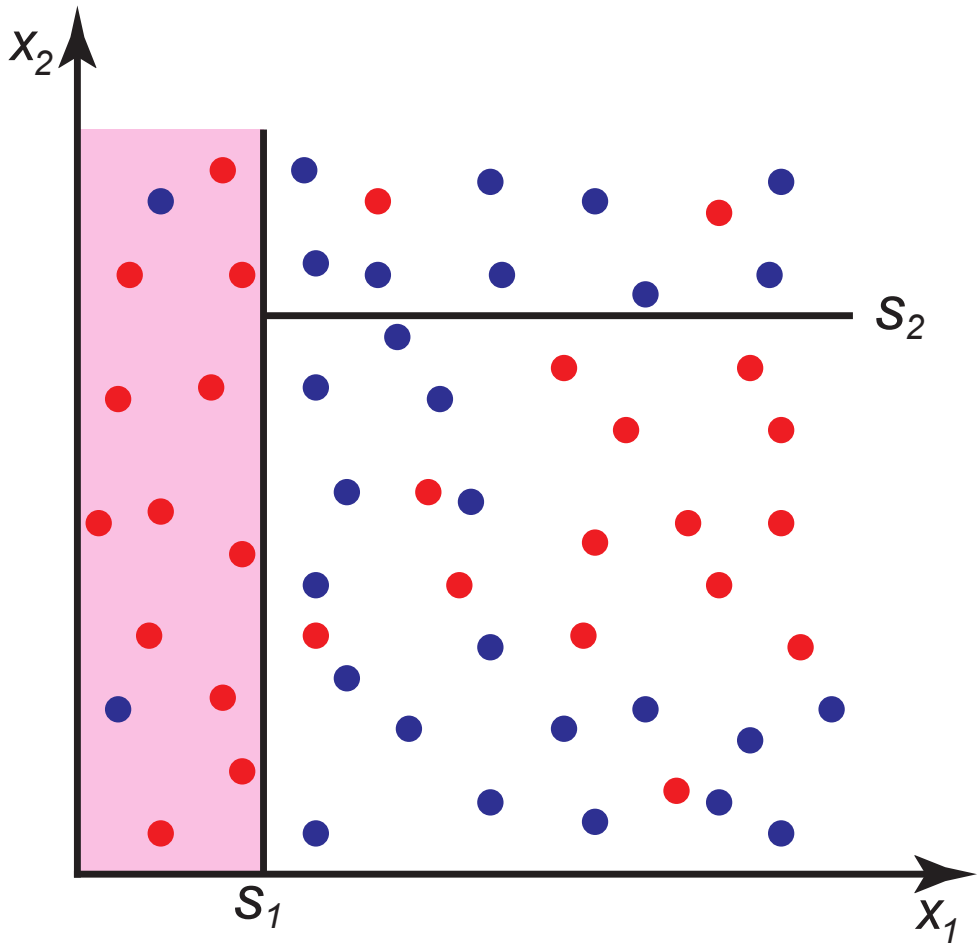


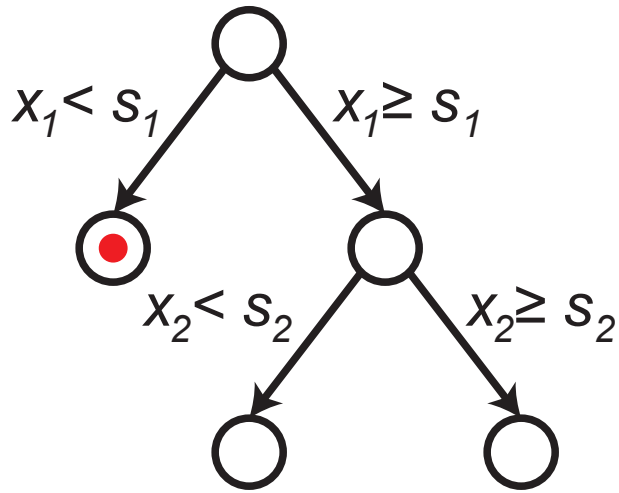
Classification tree (illustration)



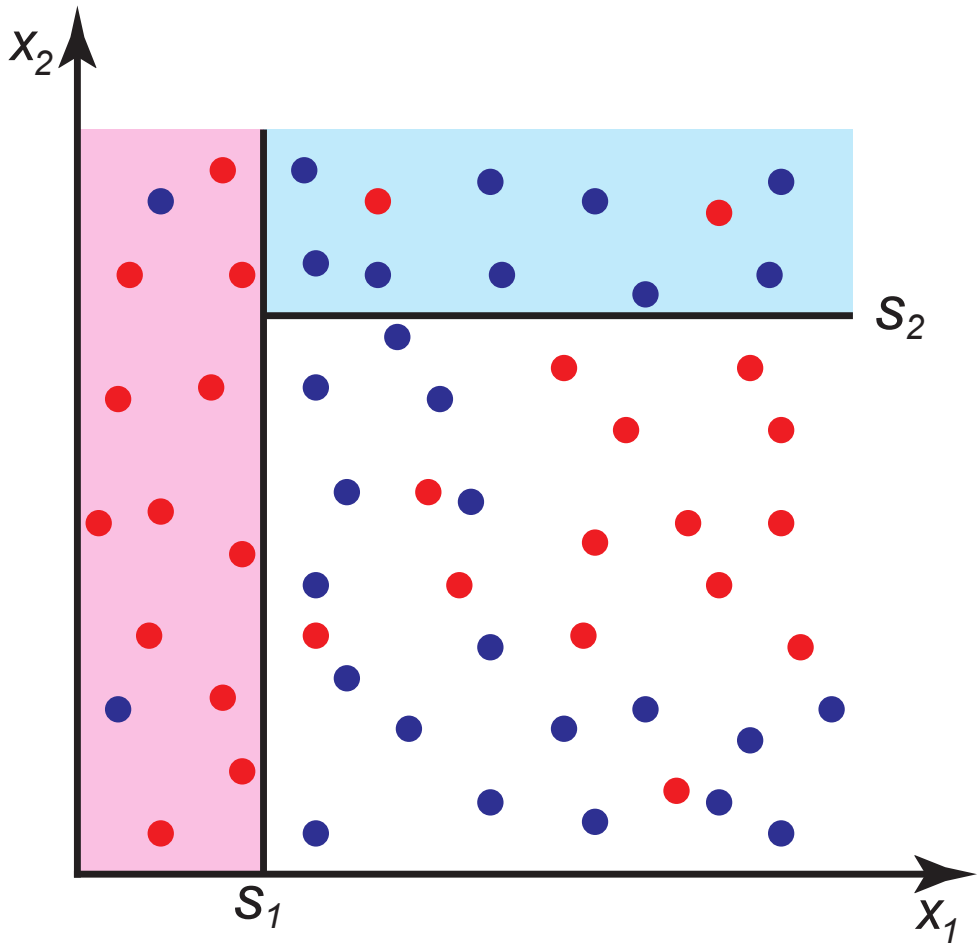


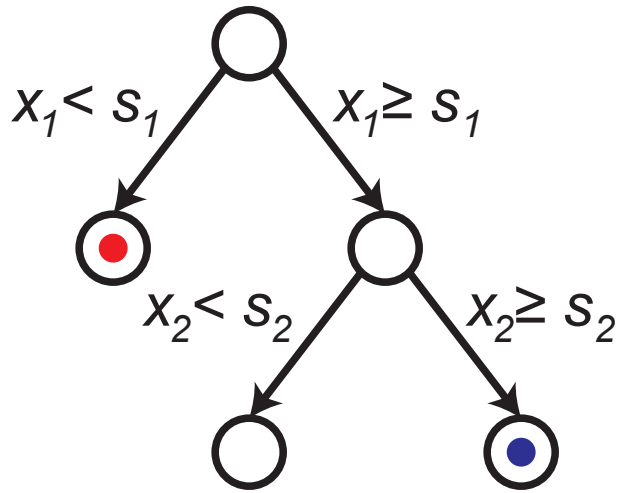
Classification tree (illustration)



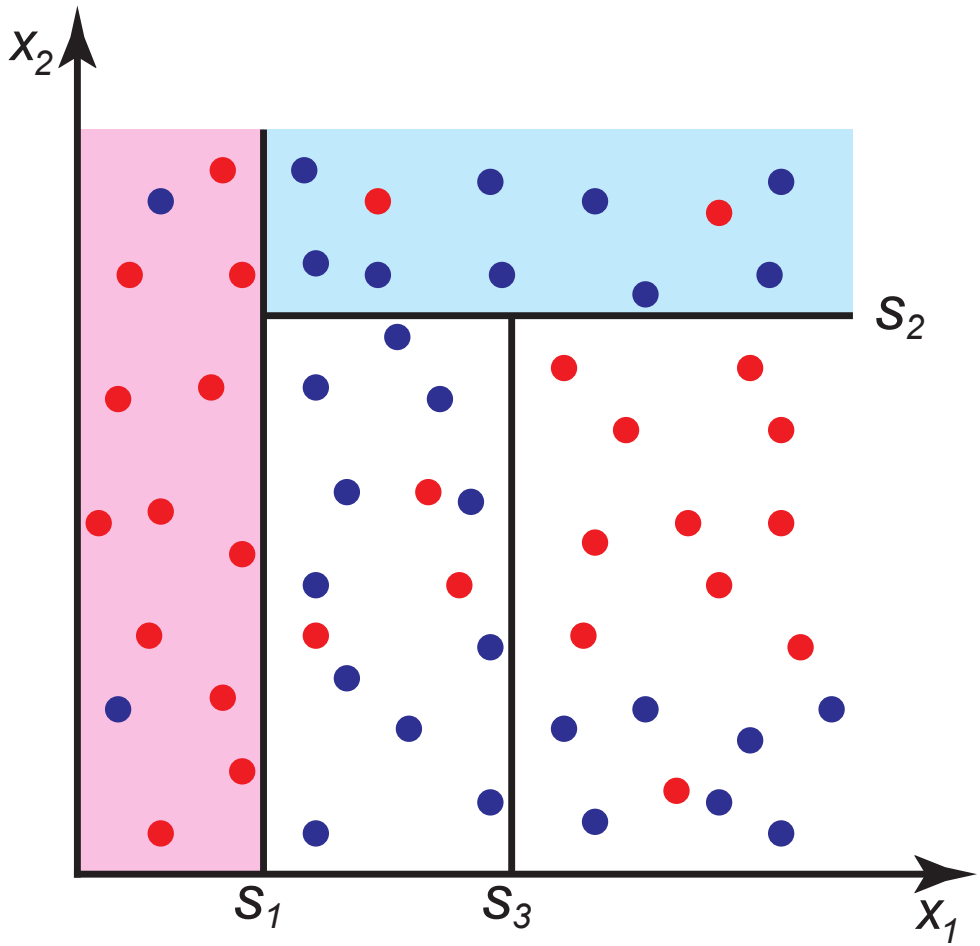


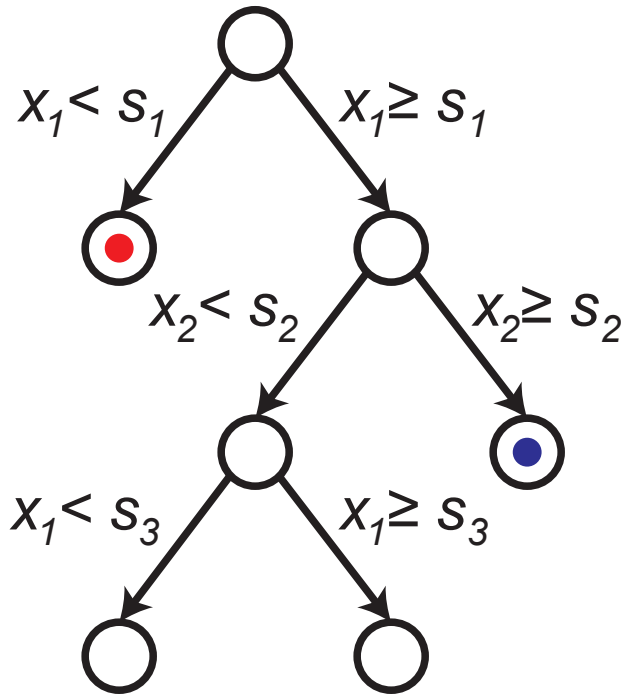
Classification tree (illustration)



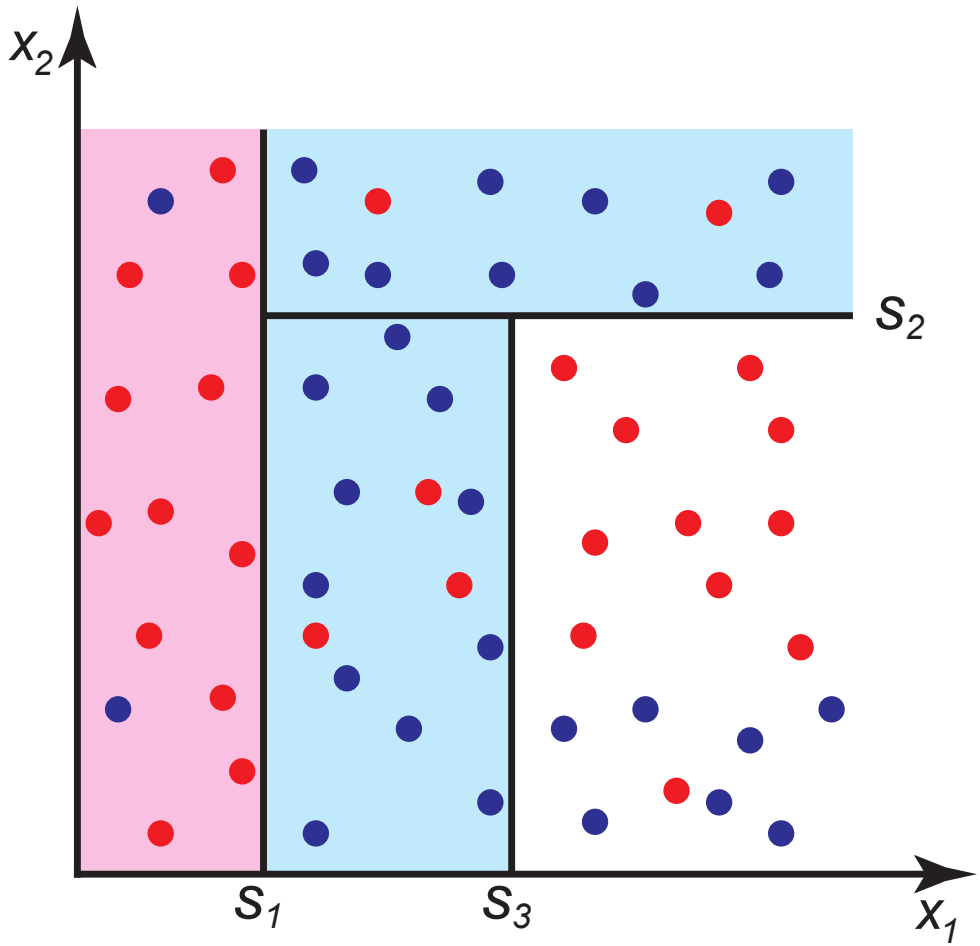


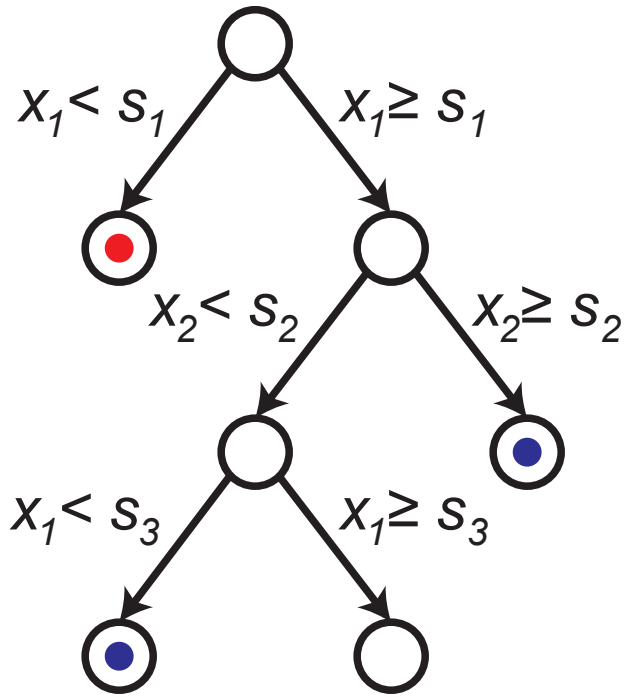
Classification tree (illustration)



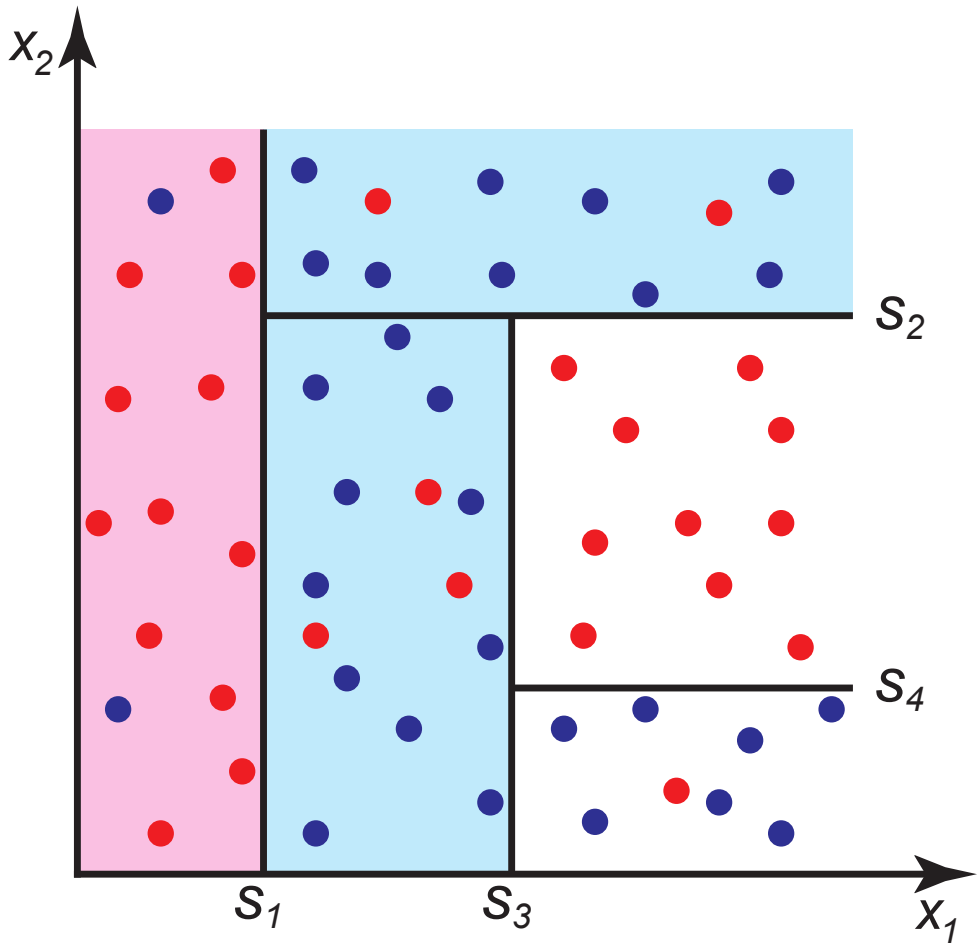


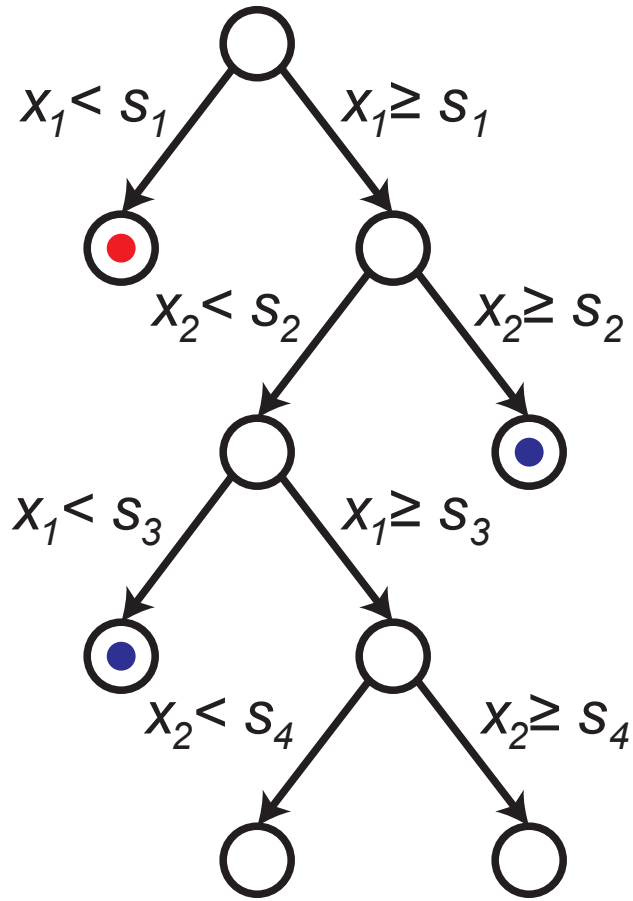
Classification tree (illustration)



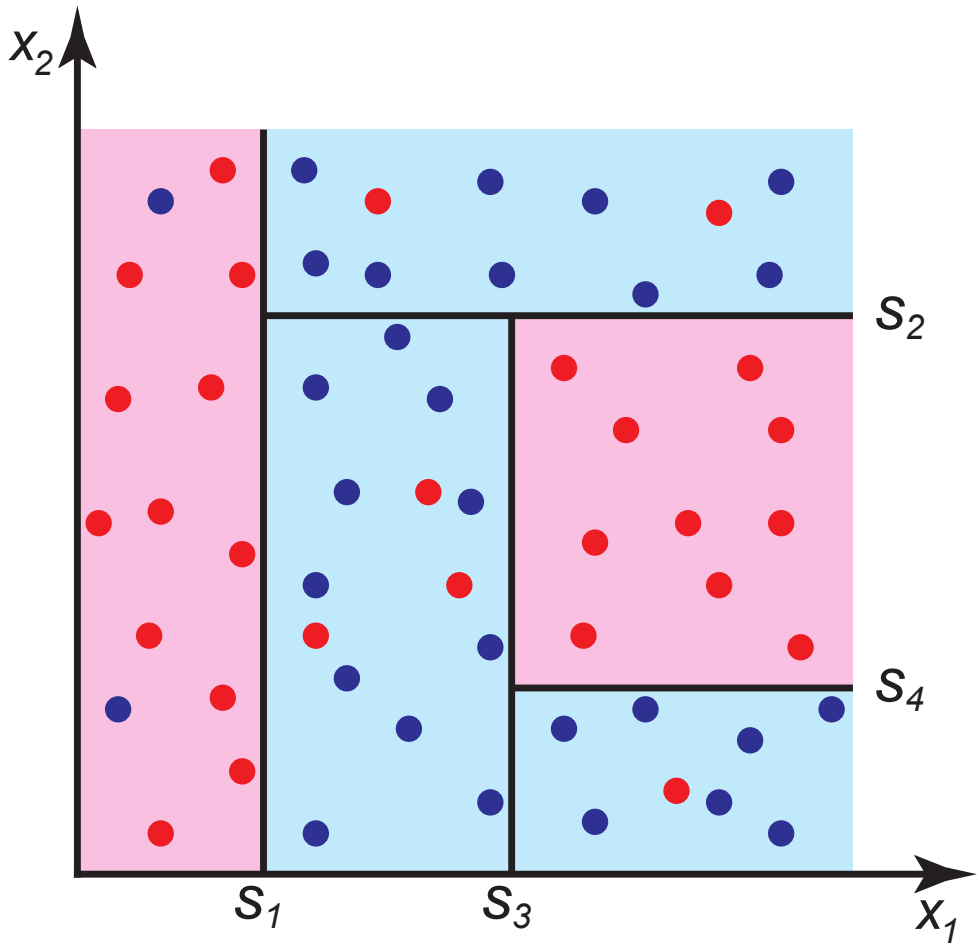


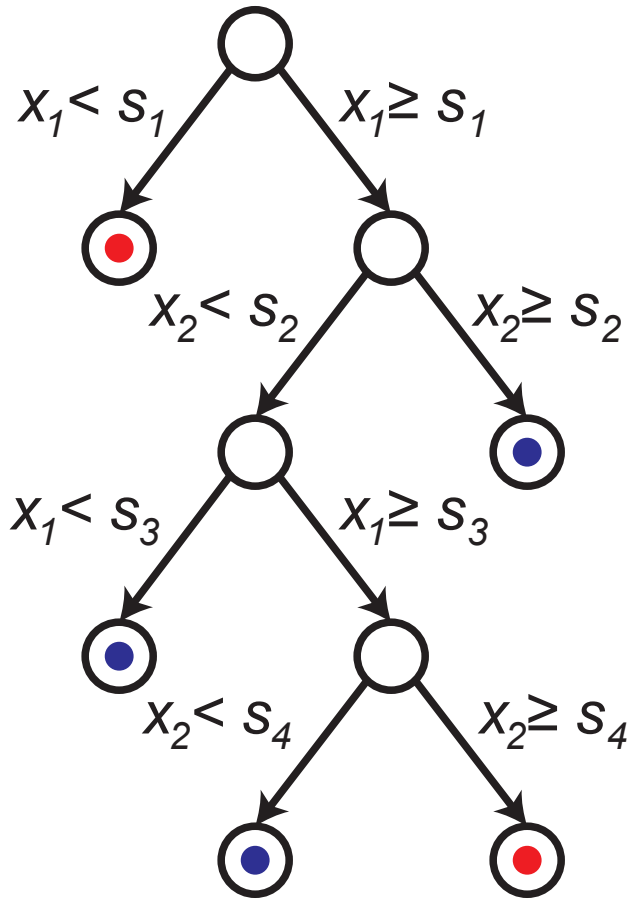
Classification tree (illustration)





Classification tree (illustration)





Classification tree: choice of impurity measure

Let $n^{(m)} = \#\{\mathbf{x} \,|\, \mathbf{x} \in R^{(m)}\}$ be the number of observations in region $R^{(m)}$. Then the classification accuracy of node m classifying to class k is

$$\hat{p}_k^{(m)} = \frac{1}{n^{(m)}} \sum_{\mathbf{x}_i \in R^{(m)}} I(y_i = k)\,.$$

Classification tree: choice of impurity measure

Let $n^{(m)} = \#\{\mathbf{x} \,|\, \mathbf{x} \in R^{(m)}\}$ be the number of observations in region $R^{(m)}$. Then the classification accuracy of node m classifying to class k is

$$\hat{p}_k^{(m)} = \frac{1}{n^{(m)}} \sum_{\mathbf{x}_i \in R^{(m)}} I(y_i = k)\,.$$

Possible choices for Q (the measure of impurity):

- Misclassification error:

$$Q^{(m)}(T) = \frac{n^{(m_L)}}{n^{(m)}}(1 - \hat{p}_{k(m_L)}^{(m_L)}) + \frac{n^{(m_R)}}{n^{(m)}}(1 - \hat{p}_{k(m_R)}^{(m_R)})\,,$$

Classification tree: choice of impurity measure

Let $n^{(m)} = \#\{\mathbf{x} \,|\, \mathbf{x} \in R^{(m)}\}$ be the number of observations in region $R^{(m)}$. Then the classification accuracy of node m classifying to class k is

$$\hat{p}_k^{(m)} = \frac{1}{n^{(m)}} \sum_{\mathbf{x}_i \in R^{(m)}} I(y_i = k)\,.$$

Possible choices for Q (the measure of impurity):

- Misclassification error:

$$Q^{(m)}(T) = \frac{n^{(m_L)}}{n^{(m)}}(1 - \hat{p}_{k(m_L)}^{(m_L)}) + \frac{n^{(m_R)}}{n^{(m)}}(1 - \hat{p}_{k(m_R)}^{(m_R)})\,,$$

- Gini index:

$$Q^{(m)}(T) = \frac{n^{(m_L)}}{n^{(m)}} 2\hat{p}_k^{(m_L)}(1 - \hat{p}_k^{(m_L)}) + \frac{n^{(m_R)}}{n^{(m)}} 2\hat{p}_k^{(m_R)}(1 - \hat{p}_k^{(m_R)})\,,$$

Classification tree: choice of impurity measure

Let $n^{(m)} = \#\{\mathbf{x} \,|\, \mathbf{x} \in R^{(m)}\}$ be the number of observations in region $R^{(m)}$. Then the classification accuracy of node m classifying to class k is

$$\hat{p}_k^{(m)} = \frac{1}{n^{(m)}} \sum_{\mathbf{x}_i \in R^{(m)}} I(y_i = k)\,.$$

Possible choices for Q (the measure of impurity):

- Misclassification error:

$$Q^{(m)}(T) = \frac{n^{(m_L)}}{n^{(m)}}(1 - \hat{p}_{k(m_L)}^{(m_L)}) + \frac{n^{(m_R)}}{n^{(m)}}(1 - \hat{p}_{k(m_R)}^{(m_R)})\,,$$

- Gini index:

$$Q^{(m)}(T) = \frac{n^{(m_L)}}{n^{(m)}} 2\hat{p}_k^{(m_L)}(1 - \hat{p}_k^{(m_L)}) + \frac{n^{(m_R)}}{n^{(m)}} 2\hat{p}_k^{(m_R)}(1 - \hat{p}_k^{(m_R)})\,,$$

- Cross-entropy (deviance):

$$\begin{aligned} Q^{(m)}(T) = \; - \; \Big\{ & \Big(\frac{n^{(m_L)}}{n^{(m)}} \big(\hat{p}_k^{(m_L)} \log \hat{p}_k^{(m_L)} + (1 - \hat{p}_k^{(m_L)}) \log(1 - \hat{p}_k^{(m_L)})\big) \\ & + \frac{n^{(m_R)}}{n^{(m)}} \big(\hat{p}_k^{(m_R)} \log \hat{p}_k^{(m_R)} + (1 - \hat{p}_k^{(m_R)}) \log(1 - \hat{p}_k^{(m_R)})\big) \Big\}\,. \end{aligned}$$

Maximizing the gain

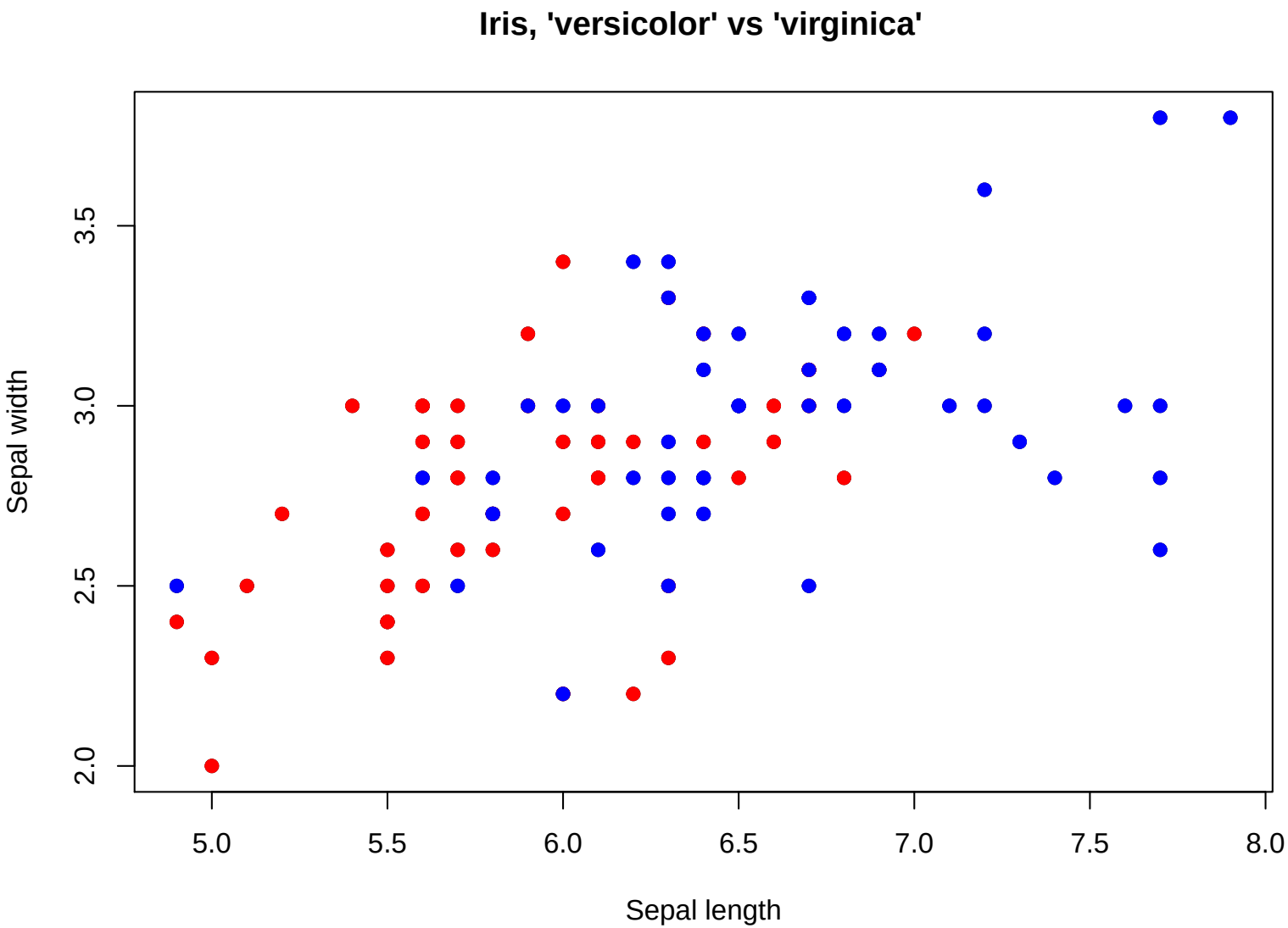
- Sometimes in the literature and in the textbooks, the minimization of $Q^{(m)}(T)$ is presented under the equivalent for of the *gain maximization*
- For instance, for the Gini index the gain is

$$2\hat{p}_k^{(m)}(1-\hat{p}_k^{(m)}) - Q_{Gini}^{(m)}(T)$$

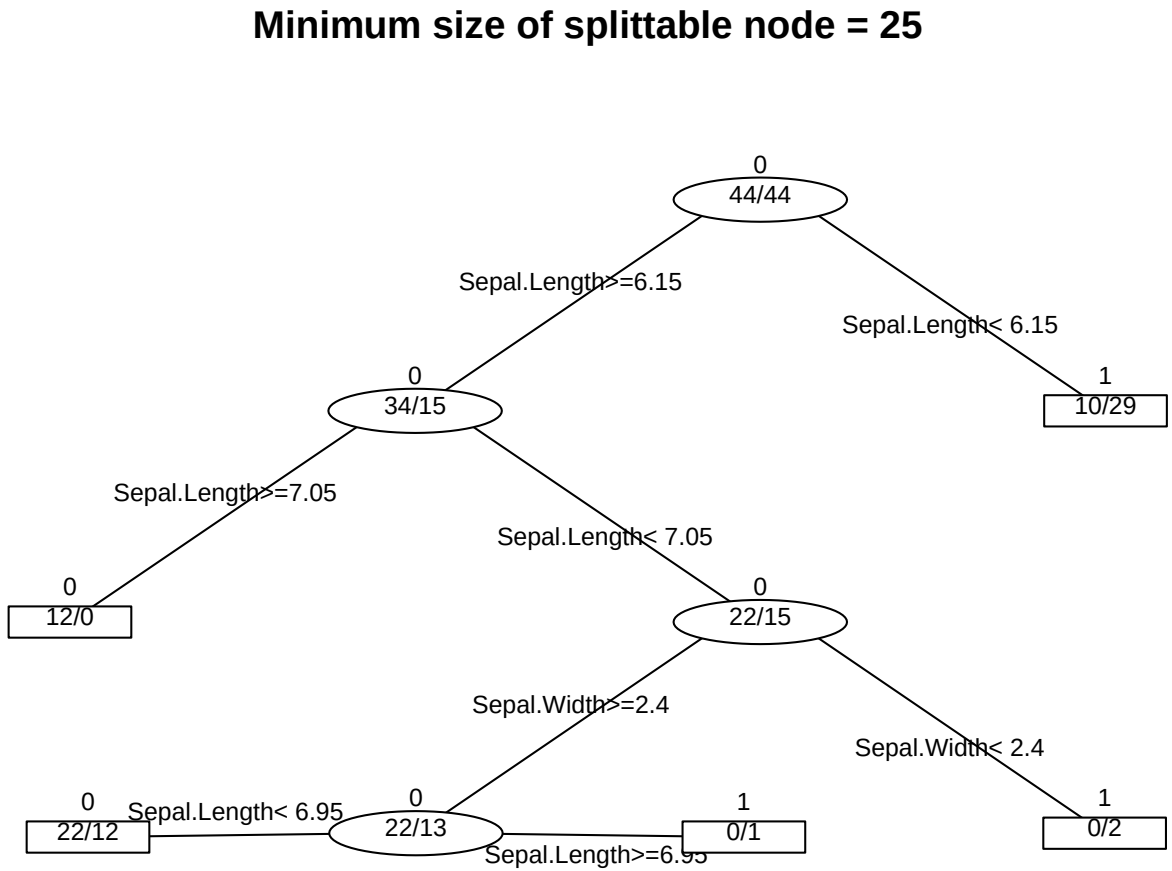
- For instance, for the deviance the gain is

$$\hat{p}_k^{(m)} \log \hat{p}_k^{(m)} + (1-\hat{p}_k^{(m)}) \log(1-\hat{p}_k^{(m)}) - Q_{deviance}^{(m)}(T)$$

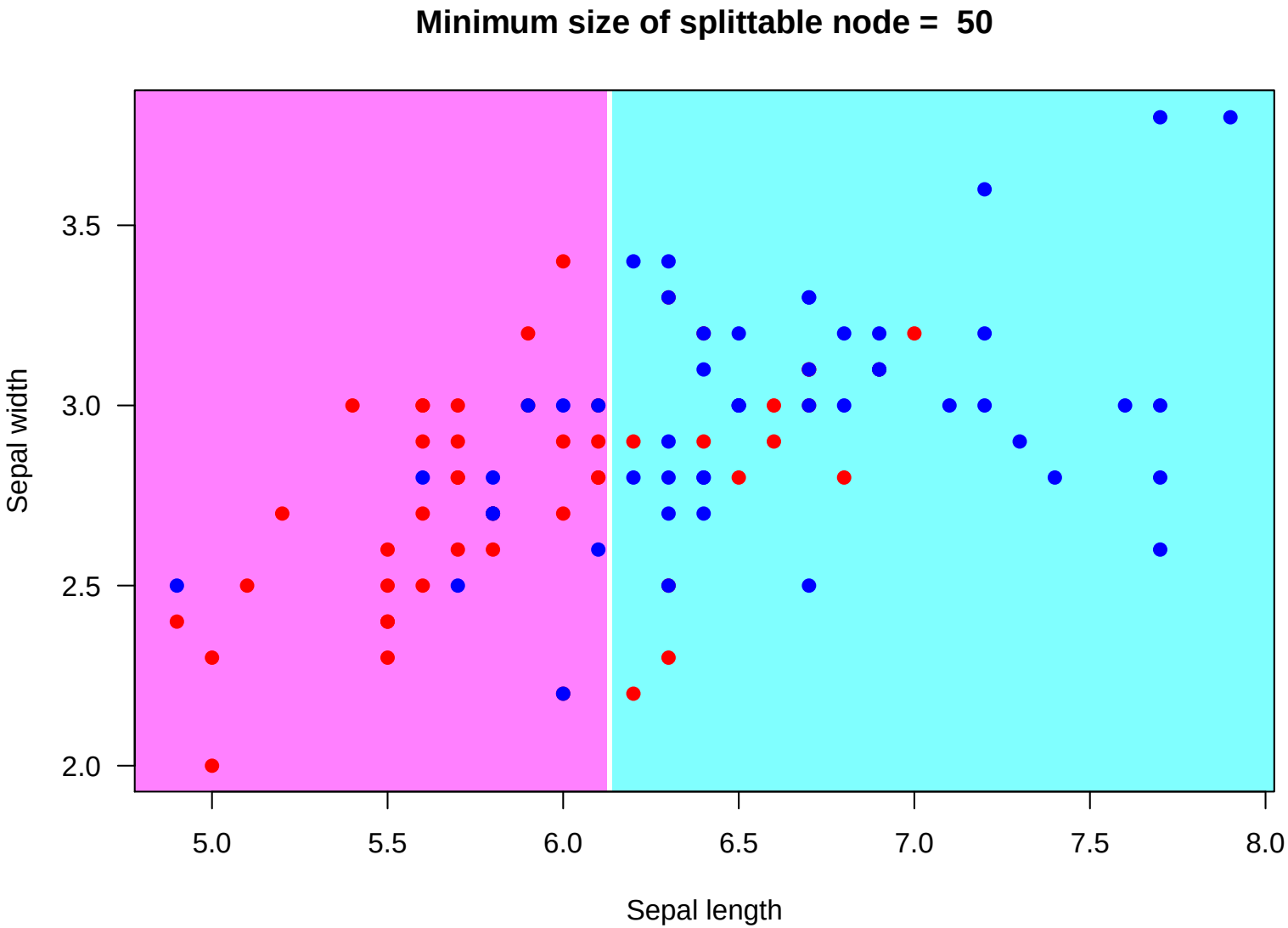
Classification tree: iris data



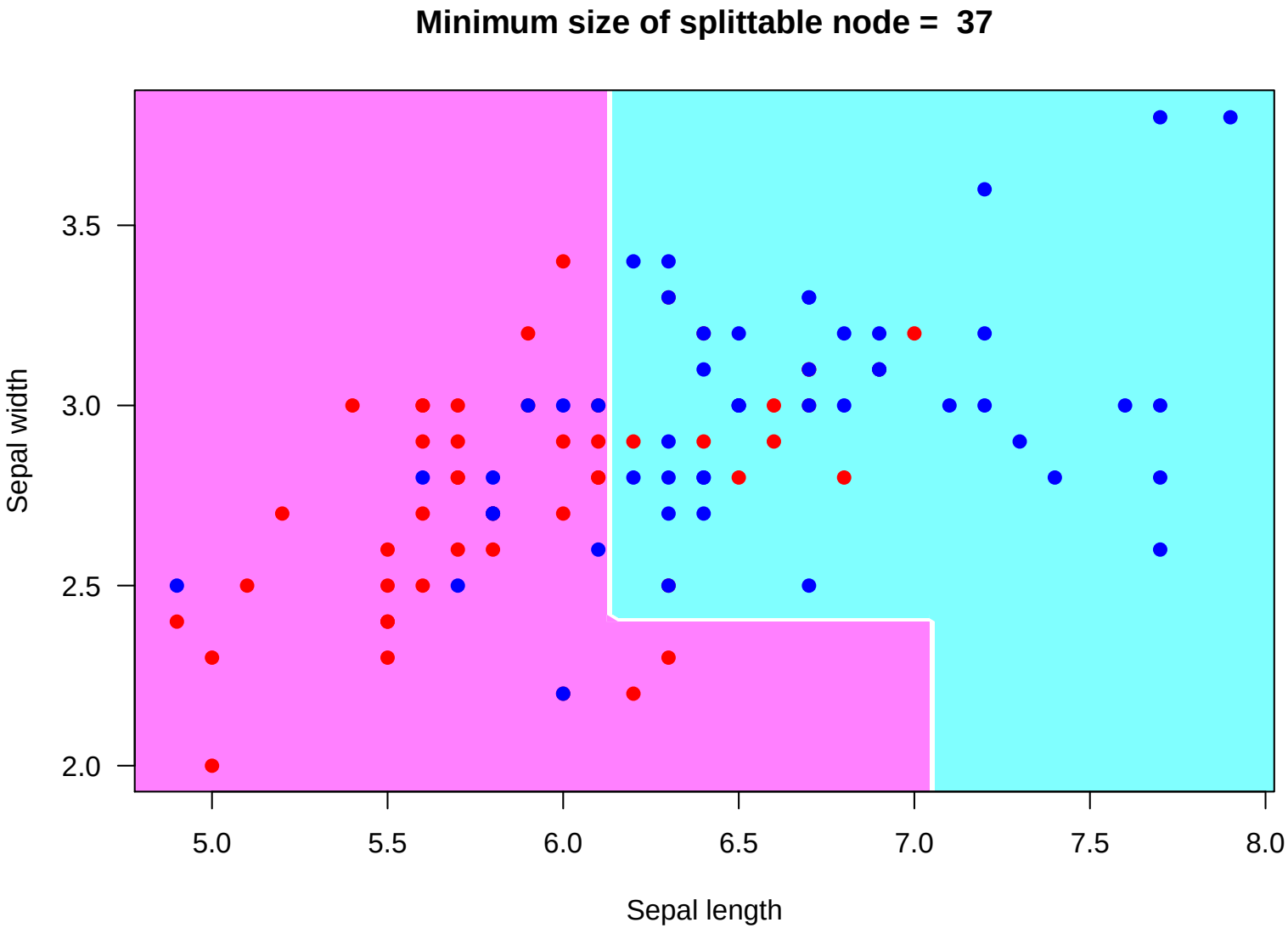
Classification tree: iris data



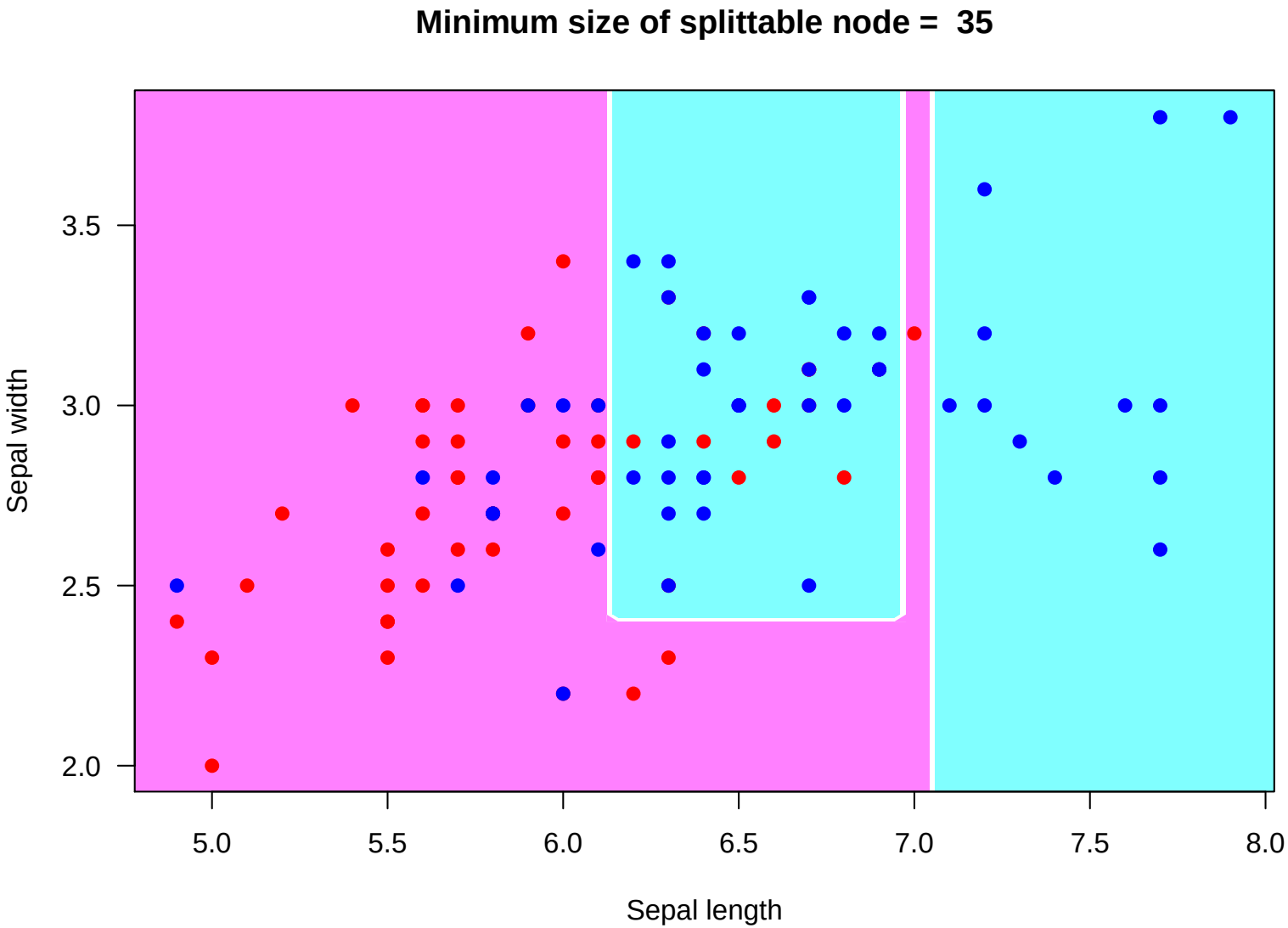
Classification tree: iris data



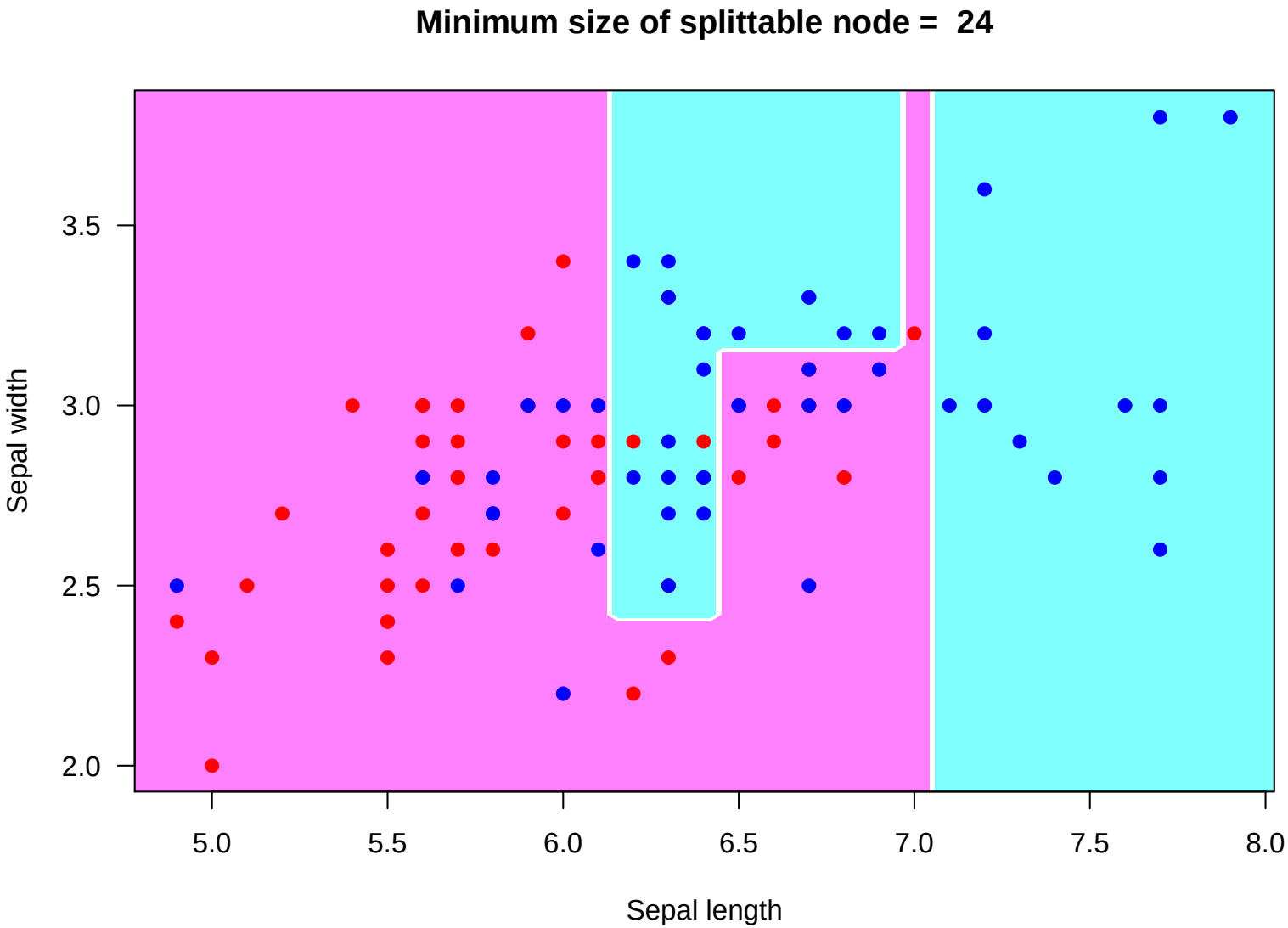
Classification tree: iris data



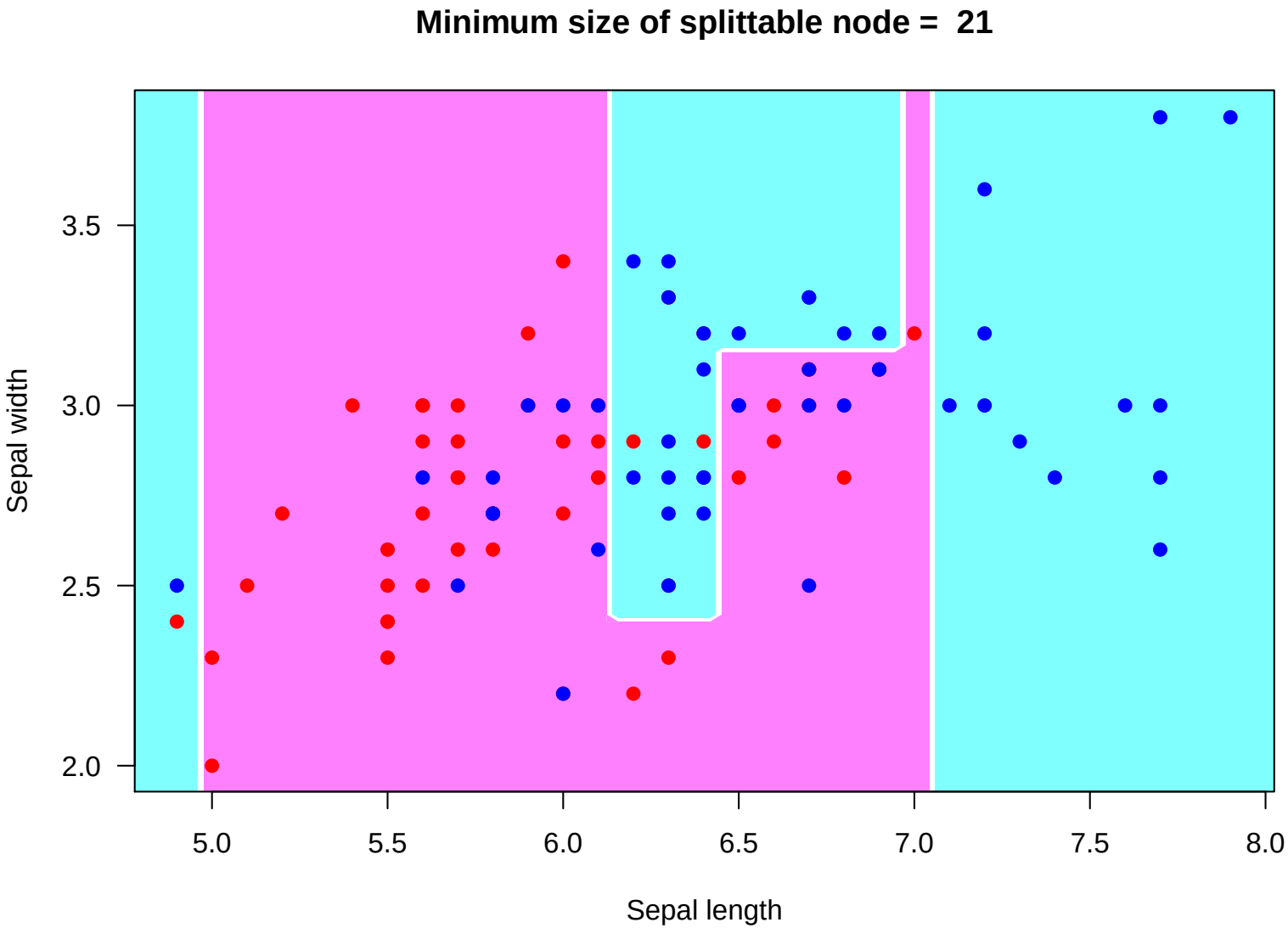
Classification tree: iris data



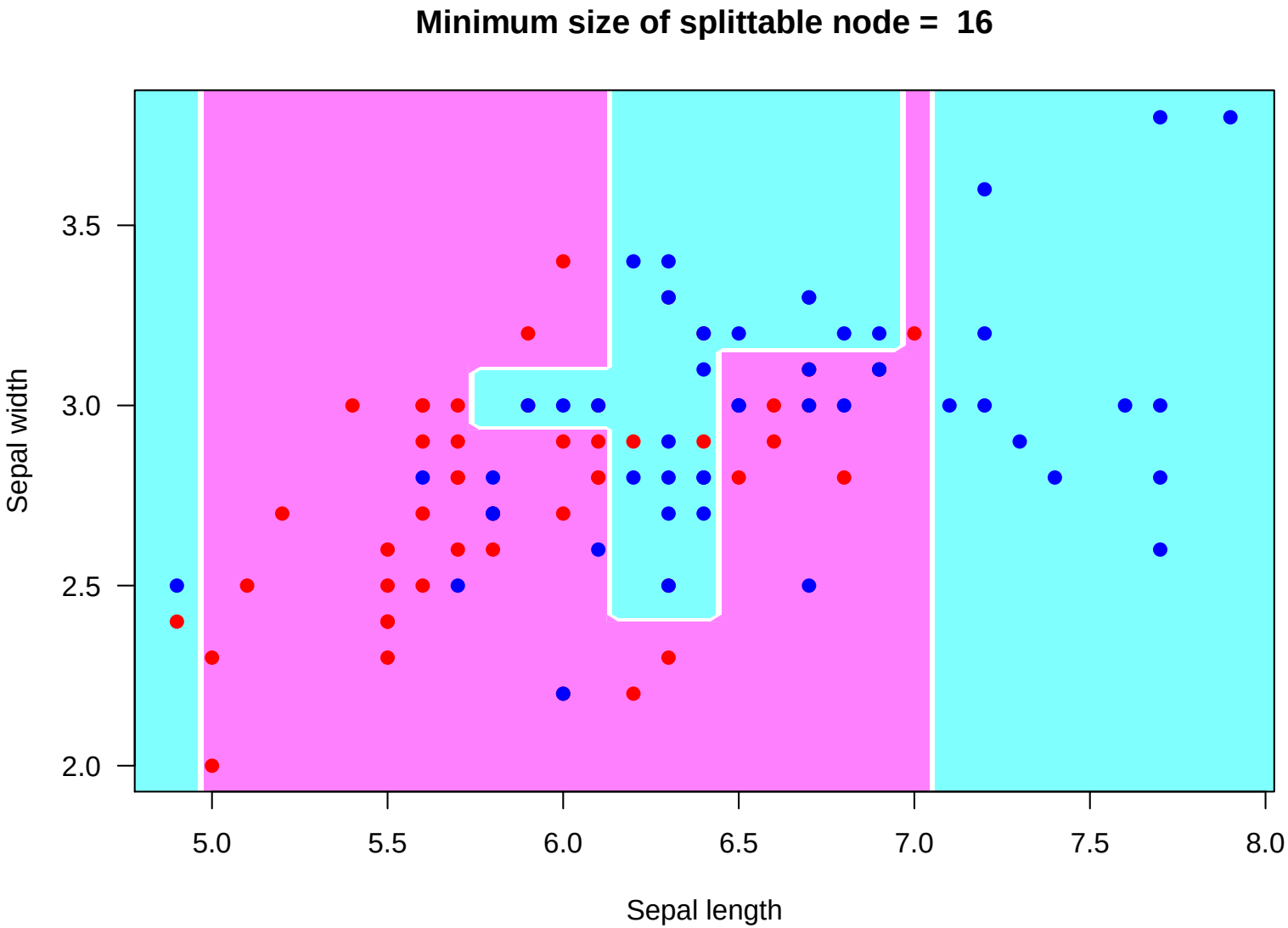
Classification tree: iris data



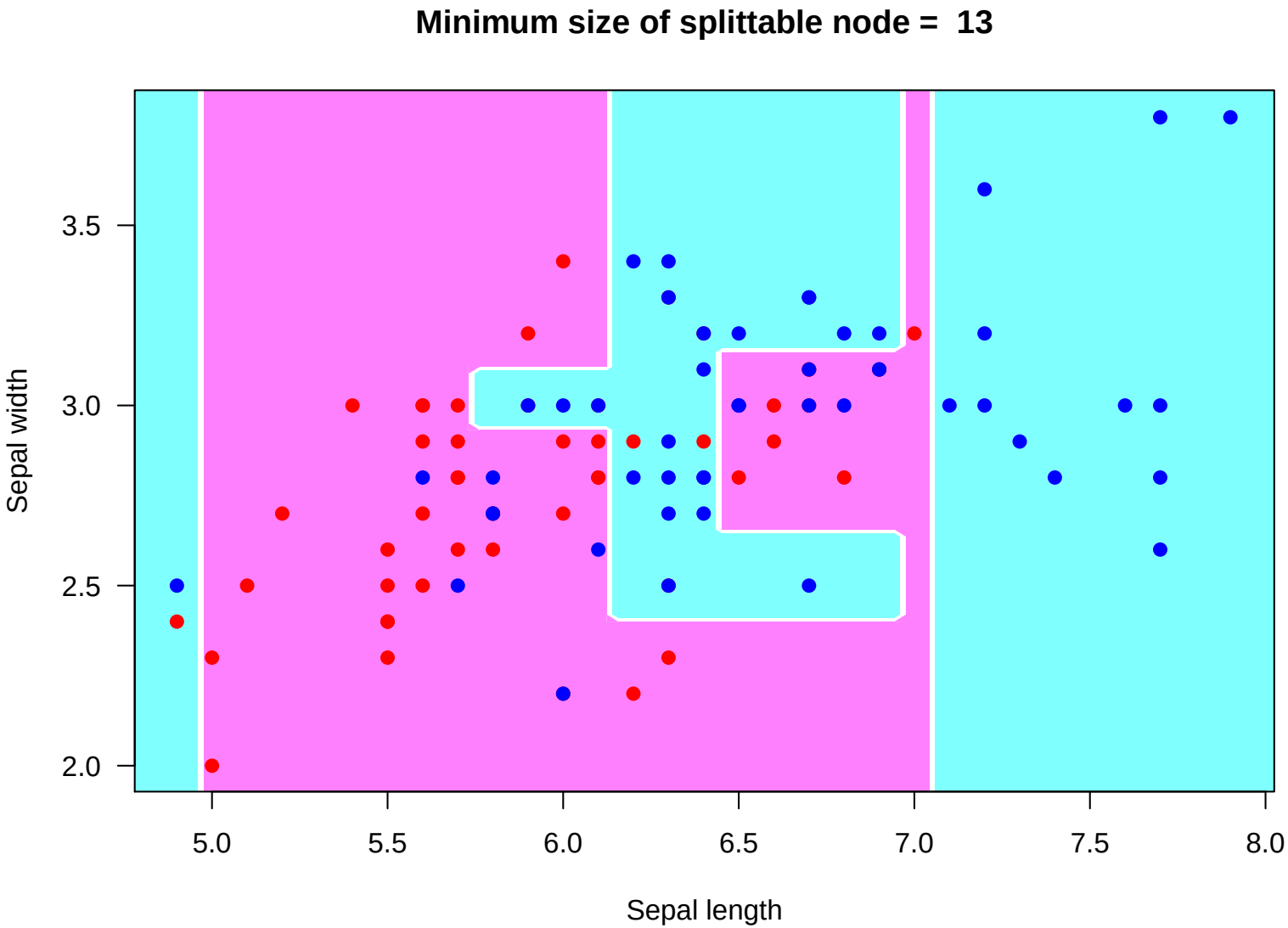
Classification tree: iris data



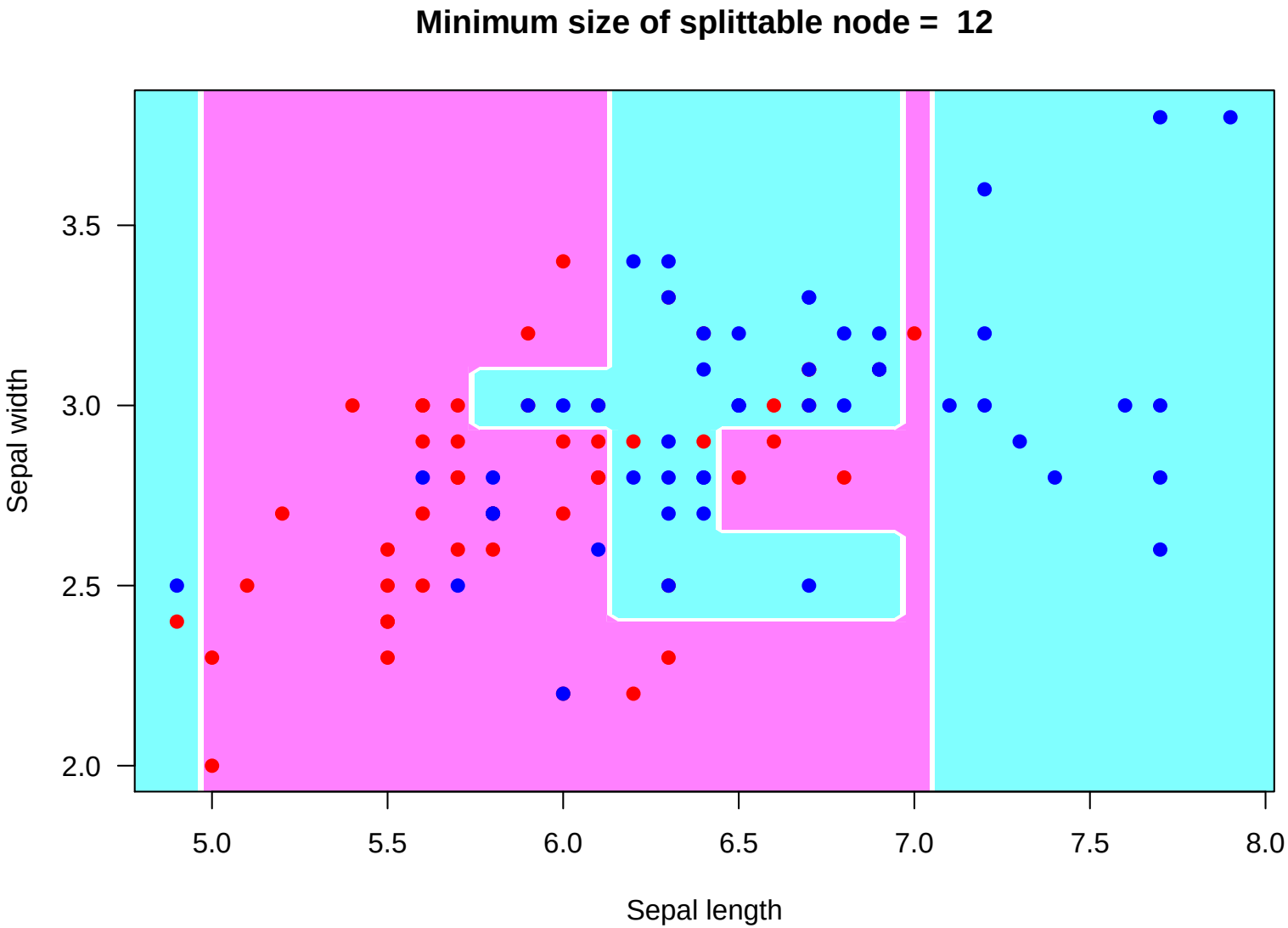
Classification tree: iris data



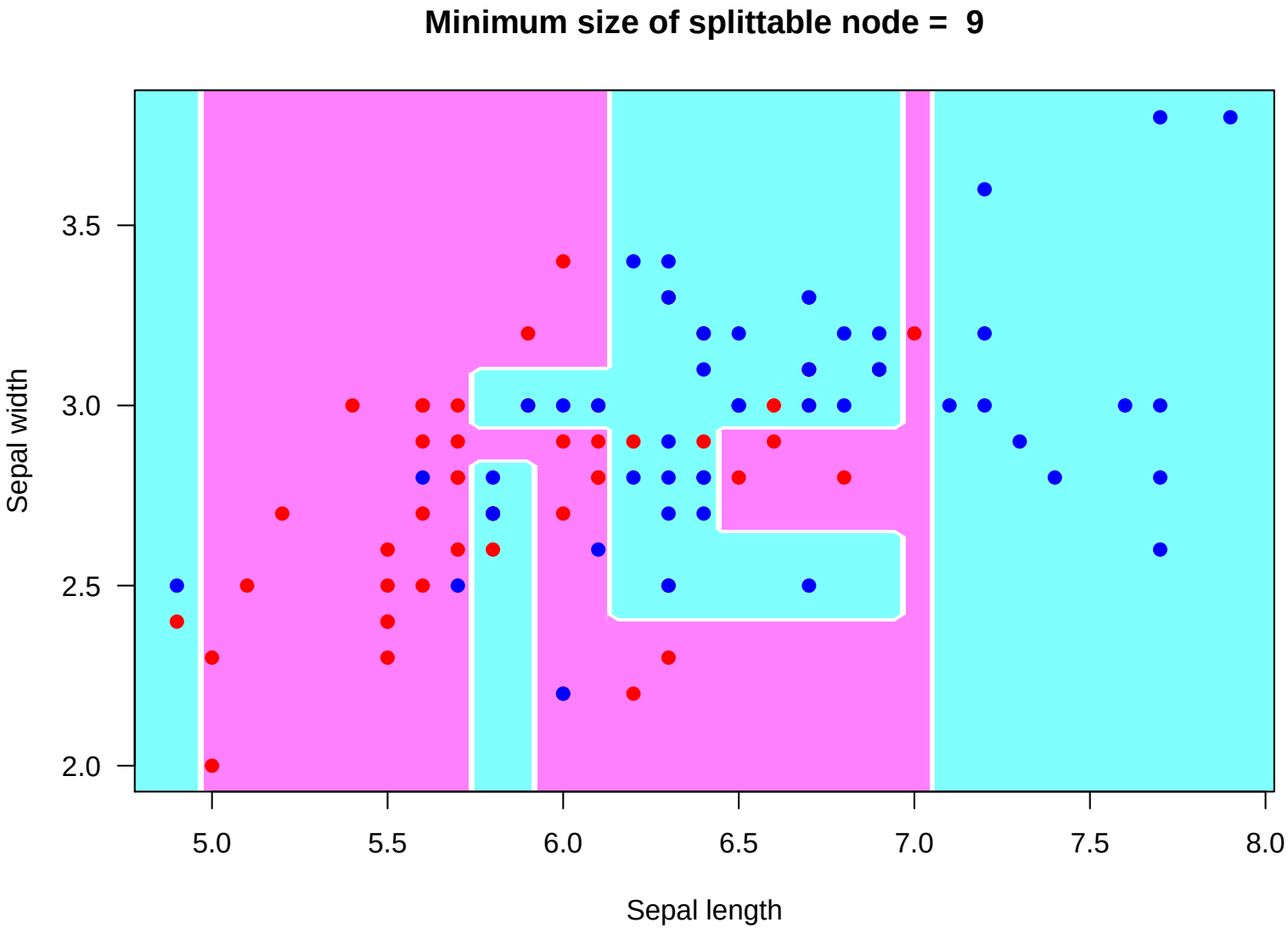
Classification tree: iris data



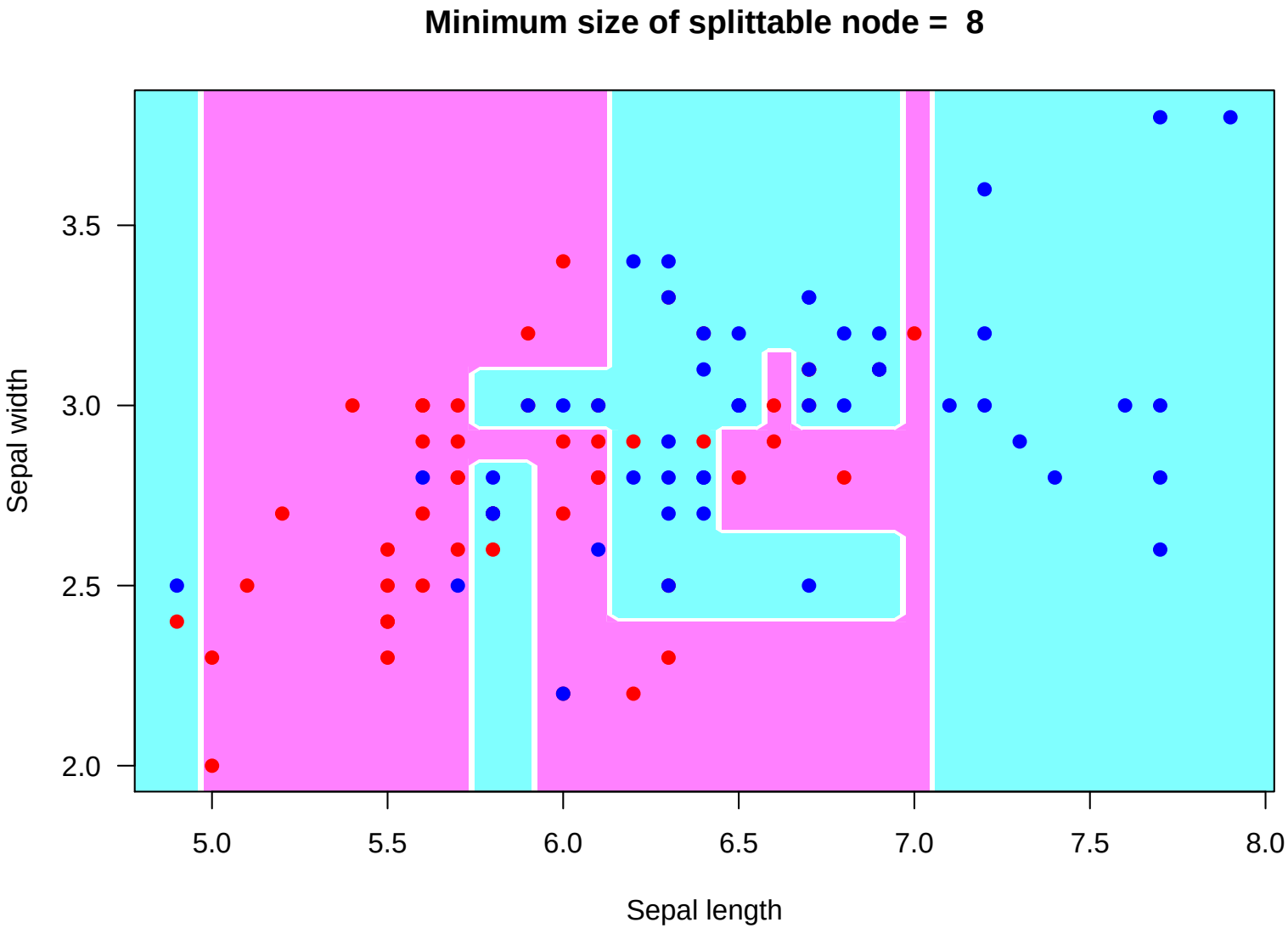
Classification tree: iris data



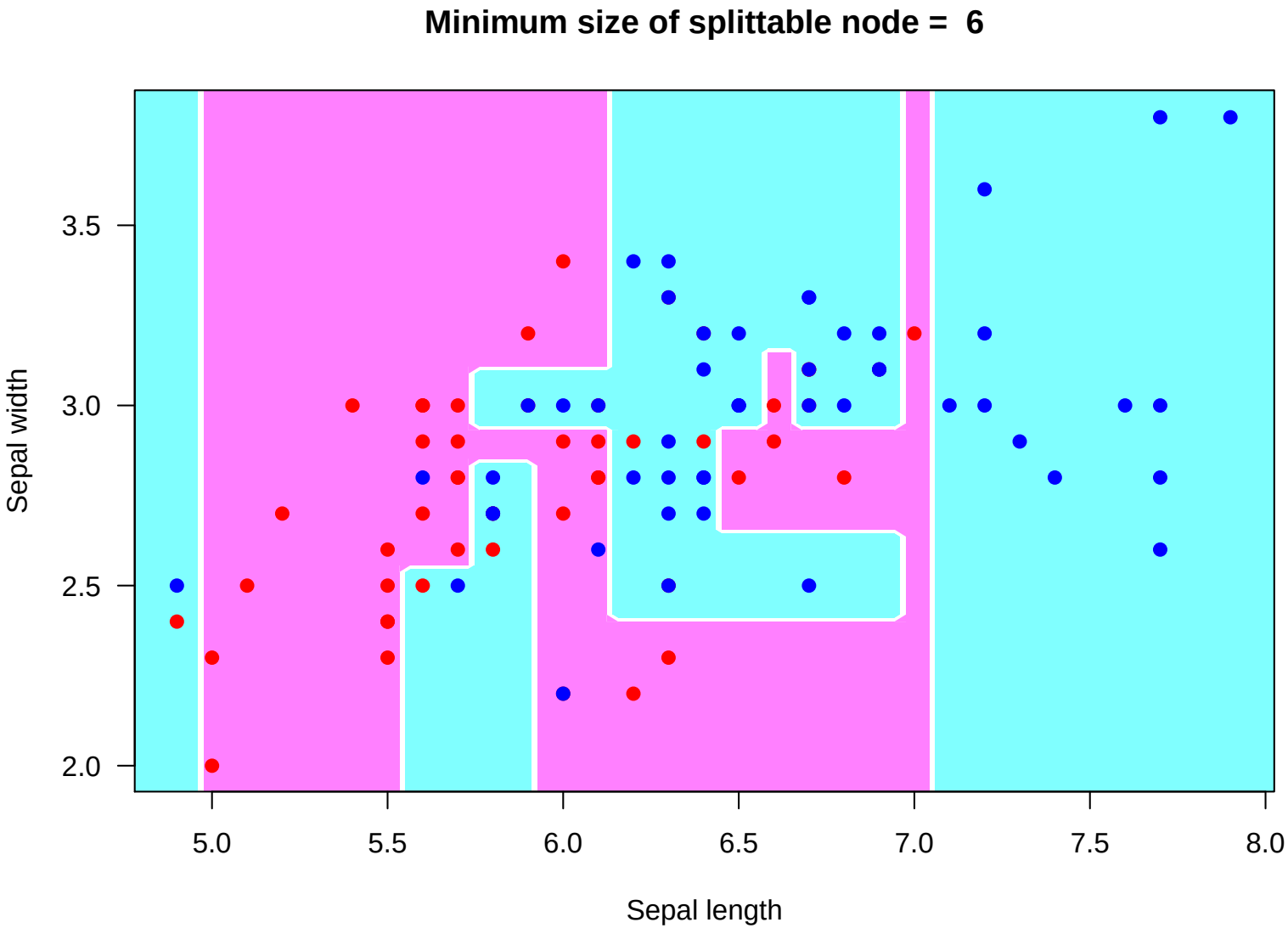
Classification tree: iris data



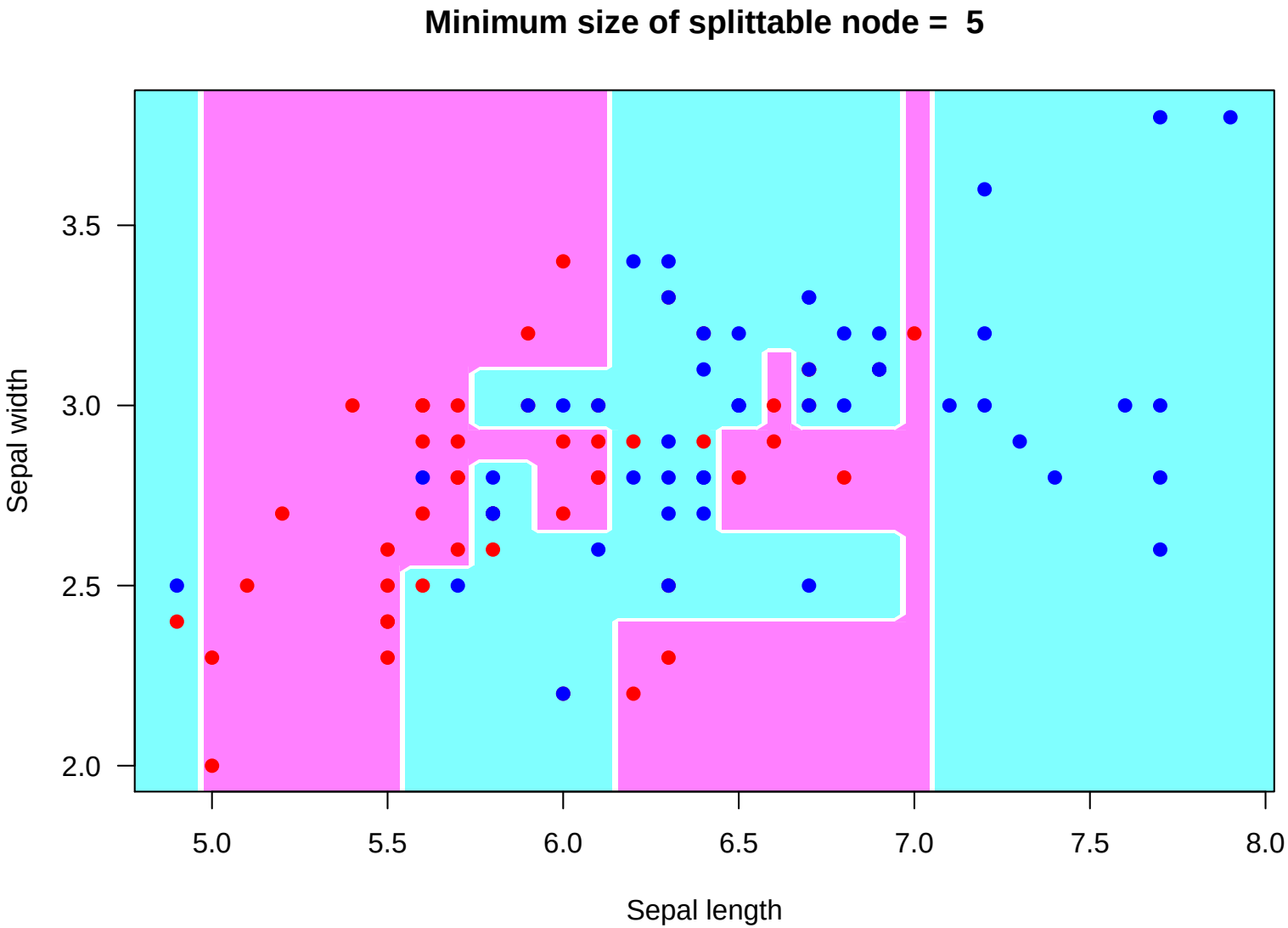
Classification tree: iris data



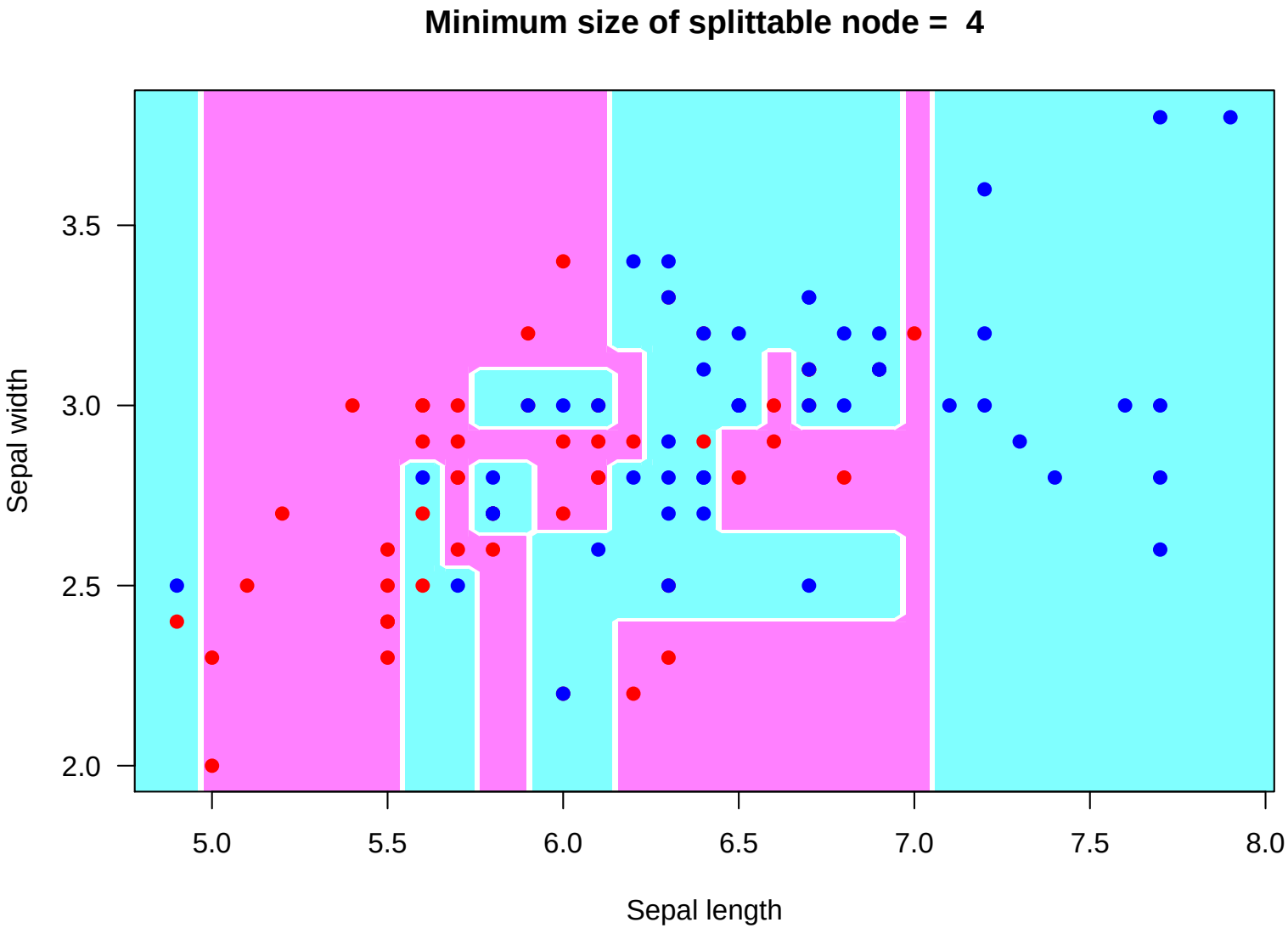
Classification tree: iris data



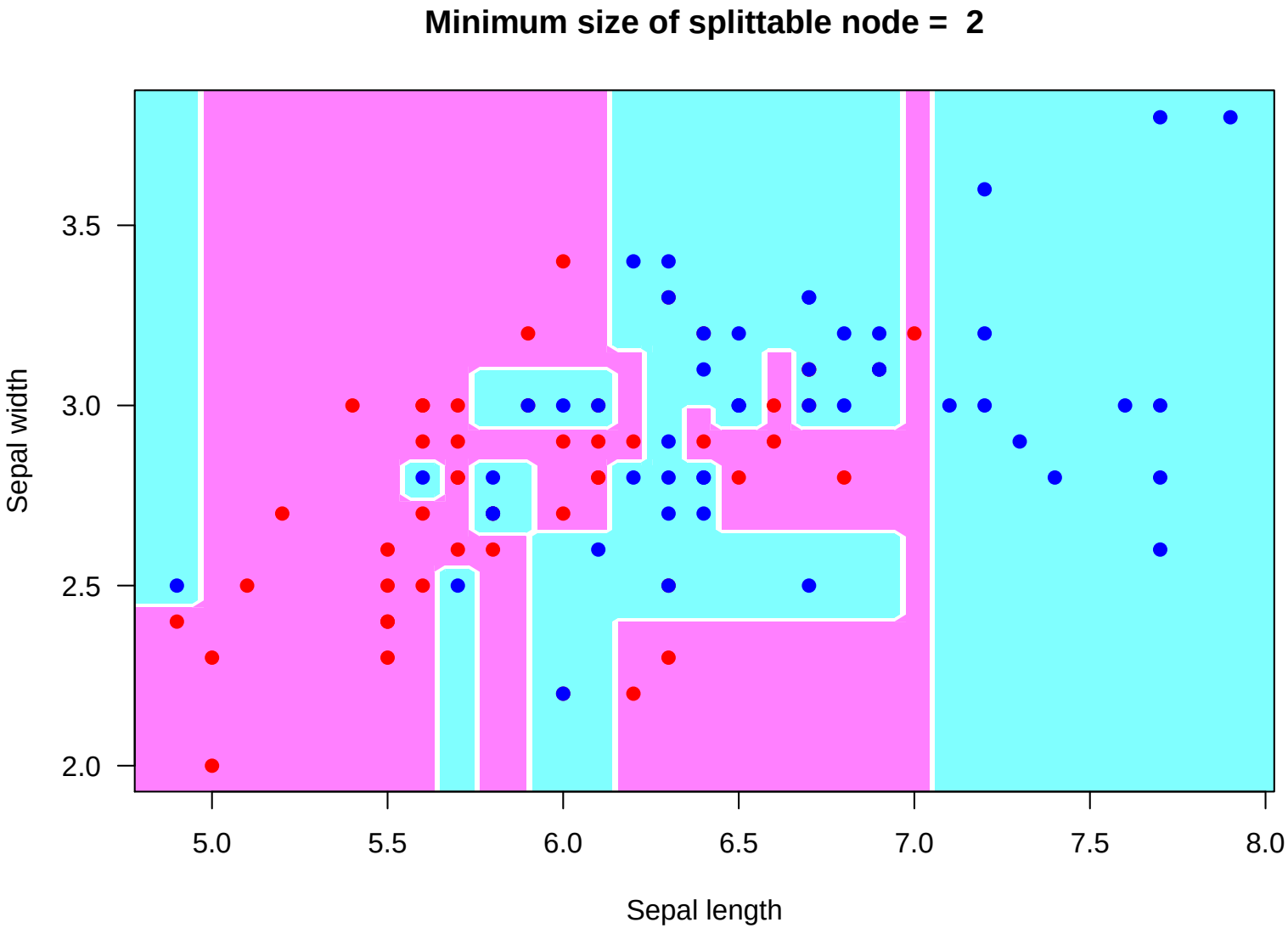
Classification tree: iris data



Classification tree: iris data



Classification tree: iris data



Contents

Classification tree: tuning and properties

- A possible (and wide-spread) choice for the stopping criteria S is to restrict **the number of points in region** $R^{(m)}$ to be split to some constant n_{min}:

$$S^{(m)}(T) = I(n^{(m)} < n_{min}) .$$

Classification tree: tuning and properties

- A possible (and wide-spread) choice for the stopping criteria S is to restrict **the number of points in region** $R^{(m)}$ to be split to some constant n_{min}:

$$S^{(m)}(T) = I(n^{(m)} < n_{min}).$$

- Often the classification tree is constructed in two stages:
  1. n_{min} **is set very small** and the tree is grown to T_0.

Classification tree: tuning and properties

- A possible (and wide-spread) choice for the stopping criteria S is to restrict **the number of points in region** $R^{(m)}$ to be split to some constant n_{min}:

$$S^{(m)}(T) = I(n^{(m)} < n_{min}) .$$

- Often the classification tree is constructed in two stages:
  1. n_{min} **is set very small** and the tree is grown to T_0.
  2. **Pruning** of tree (*i.e.*, collapsing any number of its non-terminal nodes) is conducted based on some parameter α, which consists in choosing a **subtree** $T \subset T_0$ that **minimizes** a **cost-complexity** criterion, *e.g.*

$$C_\alpha(T) = \sum_{m=1}^{\#T} n^{(m)} Q^{(m)}(T) + \alpha \#T ,$$

  where $\#T$ stands for the number of nodes in the tree.

Classification tree: tuning and properties

- A possible (and wide-spread) choice for the stopping criteria S is to restrict **the number of points in region** $R^{(m)}$ to be split to some constant n_{min}:

$$S^{(m)}(T) = I(n^{(m)} < n_{min}) .$$

- Often the classification tree is constructed in two stages:
  1. n_{min} **is set very small** and the tree is grown to T_0.
  2. **Pruning** of tree (*i.e.*, collapsing any number of its non-terminal nodes) is conducted based on some parameter α, which consists in choosing a **subtree** $T \subset T_0$ that **minimizes** a **cost-complexity** criterion, *e.g.*

$$C_\alpha(T) = \sum_{m=1}^{\#T} n^{(m)} Q^{(m)}(T) + \alpha \#T ,$$

where $\#T$ stands for the number of nodes in the tree.
For each α, it can shown that there is a unique smallest subtree T_α that minimizes $C_\alpha(T)$.
Pruning parameter α is chosen by the means of **cross-validation**.

Some more comments

- To handle *predictors with unordered values* when the outcome is of 0-1 type, one can simply order the predictor classes according to the proportion falling in outcome class 1.

Some more comments

- To handle *predictors with unordered values* when the outcome is of 0-1 type, one can simply order the predictor classes according to the proportion falling in outcome class 1.

- To handle *missing predictor values* there are two recommended approaches
  - if the predictor is categorical, create a new category 'missing';
  - a more general approach is the construction of surrogate variables (see Hastie *et al.* (2009, section 9.2) for the details).

Some more comments

- To handle *predictors with unordered values* when the outcome is of 0-1 type, one can simply order the predictor classes according to the proportion falling in outcome class 1.

- To handle *missing predictor values* there are two recommended approaches
  - if the predictor is categorical, create a new category 'missing';
  - a more general approach is the construction of surrogate variables (see Hastie *et al.* (2009, section 9.2) for the details).

- The key **advantage** of the classification tree is its **interpretability**, as the feature space partition is fully described by a single tree.

Some more comments

- To handle *predictors with unordered values* when the outcome is of 0-1 type, one can simply order the predictor classes according to the proportion falling in outcome class 1.

- To handle *missing predictor values* there are two recommended approaches
  - if the predictor is categorical, create a new category 'missing';
  - a more general approach is the construction of surrogate variables (see Hastie *et al.* (2009, section 9.2) for the details).

- The key **advantage** of the classification tree is its **interpretability**, as the feature space partition is fully described by a single tree.

- Some **disadvantage** are the *instability* (trees have high variance, a small change in the data can result in a quite different series of splits – see bagging for a solution) and the *difficulty in capturing additive structures*.

Contents

Contents

The key idea

- The "Wisdom of Crowds" (Surowiecki, 2004): The **collective knowledge** of a diverse and **independent** body of people typically exceeds the knowledge of any single individual, and can be harnessed by voting.

The key idea

- The "Wisdom of Crowds" (Surowiecki, 2004): The **collective knowledge** of a diverse and **independent** body of people typically exceeds the knowledge of any single individual, and can be harnessed by voting.

- **Bagging** implements this way of thinking standing for a range of methods following the general idea introduced by Léo Breiman (1996).

The key idea

- The "Wisdom of Crowds" (Surowiecki, 2004): The **collective knowledge** of a diverse and **independent** body of people typically exceeds the knowledge of any single individual, and can be harnessed by voting.

- **Bagging** implements this way of thinking standing for a range of methods following the general idea introduced by Léo Breiman (1996).

- **Bagging** is a shortcut for **B**ootstrap **Agg**regat**ing**.

The key idea

- The "Wisdom of Crowds" (Surowiecki, 2004): The **collective knowledge** of a diverse and **independent** body of people typically exceeds the knowledge of any single individual, and can be harnessed by voting.

- **Bagging** implements this way of thinking standing for a range of methods following the general idea introduced by Léo Breiman (1996).

- **Bagging** is a shortcut for **B**ootstrap **Agg**regat**ing**.

- The main idea is to construct a single estimator that consists of **a number of basic classifiers** (weak learners) (taught on a bootstrapped samples) aggregated by averaging (voting).

Motivation (regression)

- Consider the standard regression setting

$$Y = g(X) + \epsilon.$$

Motivation (regression)

- Consider the standard regression setting

$$Y = g(X) + \epsilon.$$

- The single bagged estimator

$$\hat{g}_B(\mathbf{x}) = \frac{1}{B}\sum_{k=1}^{B} g_k(\mathbf{x})$$

is the estimator of g obtained by aggregating estimators $g_1, ..., g_B$.

Motivation (regression)

- Consider the standard regression setting

$$Y = g(X) + \epsilon.$$

- The single bagged estimator

$$\hat{g}_B(\mathbf{x}) = \frac{1}{B}\sum_{k=1}^{B} g_k(\mathbf{x})$$

  is the estimator of g obtained by aggregating estimators $g_1, ..., g_B$.

- $g_k(\mathbf{x}) = g_k(\mathbf{x}; (X_1, Y_1), ..., (X_n, Y_n))$ as well as $\hat{g}_B(\mathbf{x}) = \hat{g}_B(\mathbf{x}; (X_1, Y_1), ..., (X_n, Y_n))$ are random variables.

Motivation (regression)

- Consider the standard regression setting

$$Y = g(X) + \epsilon.$$

- The single bagged estimator

$$\hat{g}_B(\mathbf{x}) = \frac{1}{B}\sum_{k=1}^{B} g_k(\mathbf{x})$$

  is the estimator of g obtained by aggregating estimators $g_1, ..., g_B$.

- $g_k(\mathbf{x}) = g_k(\mathbf{x}; (X_1, Y_1), ..., (X_n, Y_n))$ as well as $\hat{g}_B(\mathbf{x}) = \hat{g}_B(\mathbf{x}; (X_1, Y_1), ..., (X_n, Y_n))$ are random variables.

- One can measure the improvement of aggregating by comparing performance of $\hat{g}_B(\mathbf{x})$ and those of $g_k(x)$, $k = 1, ..., B$ in terms of **bias** and **variance**.

Bias and variance (regression)

- **Assumption (unfeasible):** Random variables $g_1, ..., g_B$ are i.i.d.

Bias and variance (regression)

- **Assumption (unfeasible):** Random variables $g_1, ..., g_B$ are i.i.d.

- **Bias:**

$$\mathbb{E}[\hat{g}_B(\mathbf{x})] = \mathbb{E}[g_k(\mathbf{x})]\,.$$

Bias and variance (regression)

- **Assumption (unfeasible):** Random variables $g_1, ..., g_B$ are i.i.d.

- **Bias:**

$$\mathbb{E}[\hat{g}_B(\mathbf{x})] = \mathbb{E}[g_k(\mathbf{x})]\,.$$

Conclusion

Aggregation does not modify the bias.

Bias and variance (regression)

- **Assumption (unfeasible):** Random variables $g_1, ..., g_B$ are i.i.d.

- **Bias:**

$$\mathbb{E}[\hat{g}_B(\mathbf{x})] = \mathbb{E}[g_k(\mathbf{x})]\,.$$

Conclusion

Aggregation does not modify the bias.

- **Variance:**

$$\mathbb{V}[\hat{g}_B(\mathbf{x})] = \frac{1}{B}\mathbb{V}[g_k(\mathbf{x})]\,.$$

Bias and variance (regression)

- **Assumption (unfeasible):** Random variables $g_1, ..., g_B$ are i.i.d.

- **Bias:**

$$\mathbb{E}[\hat{g}_B(\mathbf{x})] = \mathbb{E}[g_k(\mathbf{x})]\,.$$

Conclusion

Aggregation does not modify the bias.

- **Variance:**

$$\mathbb{V}[\hat{g}_B(\mathbf{x})] = \frac{1}{B}\mathbb{V}[g_k(\mathbf{x})]\,.$$

Conclusion

Aggregation reduces the variance
(the conclusion here is obtained under the unfeasible assumption of i.i.d. property of $g_1, ..., g_B$).

Motivation (classification)

- Let $g_1, ..., g_B$ be an ensemble of **basic classifiers**.

Motivation (classification)

- Let $g_1, ..., g_B$ be an ensemble of **basic classifiers**.
- **Assumption:** Each basic classifier has an independent error $\epsilon < 0.5$ for predicting the correct decision $y = 1$ for some value $\mathbf{x}$:

$$\mathbb{P}\big(g_k(\mathbf{x}) \neq 1\big) = \epsilon < 0.5 \quad \text{for} \quad k = 1, ..., B\,,$$

$$g_1(\mathbf{x}), ..., g_B(\mathbf{x}) \quad \text{are still assumed } \textbf{i.i.d}\,.$$

Motivation (classification)

- Let $g_1, ..., g_B$ be an ensemble of **basic classifiers**.
- **Assumption:** Each basic classifier has an independent error $\epsilon < 0.5$ for predicting the correct decision $y = 1$ for some value $\mathbf{x}$:

$$\mathbb{P}\big(g_k(\mathbf{x}) \neq 1\big) = \epsilon < 0.5 \quad \text{for} \quad k = 1, ..., B\,,$$
$$g_1(\mathbf{x}), ..., g_B(\mathbf{x}) \quad \text{are still assumed } \textbf{i.i.d}\,.$$

- Further, let the aggregated classifier be

$$g^{agg}(\mathbf{x}) = 1\Big(\frac{1}{B}\sum_k g_k(\mathbf{x}) > 0.5\Big)\,.$$

Motivation (classification)

- Let $g_1, ..., g_B$ be an ensemble of **basic classifiers**.
- **Assumption:** Each basic classifier has an independent error $\epsilon < 0.5$ for predicting the correct decision $y = 1$ for some value $\mathbf{x}$:

$$\mathbb{P}\big(g_k(\mathbf{x}) \neq 1\big) = \epsilon < 0.5 \quad \text{for} \quad k = 1, ..., B\,,$$
$$g_1(\mathbf{x}), ..., g_B(\mathbf{x}) \quad \text{are still assumed } \textbf{i.i.d}\,.$$

- Further, let the aggregated classifier be

$$g^{agg}(\mathbf{x}) = 1\Big(\frac{1}{B}\sum_k g_k(\mathbf{x}) > 0.5\Big)\,.$$

- Then $\sum_k g_k(\mathbf{x})$ will have binomial distribution

$$\sum_k g_k(\mathbf{x}) \sim Bin(B, 1-\epsilon)$$

Motivation (classification)

- Let $g_1, ..., g_B$ be an ensemble of **basic classifiers**.
- **Assumption:** Each basic classifier has an independent error $\epsilon < 0.5$ for predicting the correct decision $y = 1$ for some value $\mathbf{x}$:

$$\mathbb{P}\big(g_k(\mathbf{x}) \neq 1\big) = \epsilon < 0.5 \quad \text{for} \quad k = 1, ..., B\,,$$
$$g_1(\mathbf{x}), ..., g_B(\mathbf{x}) \quad \text{are still assumed } \textbf{i.i.d}\,.$$

- Further, let the aggregated classifier be

$$g^{agg}(\mathbf{x}) = 1\Big(\frac{1}{B}\sum_k g_k(\mathbf{x}) > 0.5\Big)\,.$$

- Then $\sum_k g_k(\mathbf{x})$ will have binomial distribution

$$\sum_k g_k(\mathbf{x}) \sim Bin(B, 1-\epsilon)$$

and classification error of $\mathbf{x}$ will decrease with increasing B:

$$\mathbb{P}\big(g^{agg}(\mathbf{x}) \neq 1\big) = \sum_{k=1}^{B/2} \binom{B}{k} (1-\epsilon)^k \epsilon^{B-k} \xrightarrow[B\to\infty]{}$$

Motivation (classification)

- Let $g_1, ..., g_B$ be an ensemble of **basic classifiers**.
- **Assumption:** Each basic classifier has an independent error $\epsilon < 0.5$ for predicting the correct decision $y = 1$ for some value $\mathbf{x}$:

$$\mathbb{P}\big(g_k(\mathbf{x}) \neq 1\big) = \epsilon < 0.5 \quad \text{for} \quad k = 1, ..., B\,,$$
$$g_1(\mathbf{x}), ..., g_B(\mathbf{x}) \quad \text{are still assumed } \textbf{i.i.d}\,.$$

- Further, let the aggregated classifier be

$$g^{agg}(\mathbf{x}) = 1\Big(\frac{1}{B}\sum_k g_k(\mathbf{x}) > 0.5\Big)\,.$$

- Then $\sum_k g_k(\mathbf{x})$ will have binomial distribution

$$\sum_k g_k(\mathbf{x}) \sim Bin(B, 1-\epsilon)$$

and classification error of $\mathbf{x}$ will decrease with increasing B:

$$\mathbb{P}\big(g^{agg}(\mathbf{x}) \neq 1\big) = \sum_{k=1}^{B/2} \binom{B}{k} (1-\epsilon)^k \epsilon^{B-k} \xrightarrow[B\to\infty]{} 0\,.$$

Motivation (classification)

Theorem (Chernoff-Hoeffding, Bernoulli scheme)

If $X_1, ..., X_n$ are i.i.d. random variables taking values in $\{0, 1\}$, then for any $\eta > 0$ it holds

$$\mathbb{P}\left(\mathbb{E}[X_i] - \frac{1}{n}\sum_{i=1}^{n} X_i > \eta\right) < \exp(-2\eta^2 n)\,.$$

Motivation (classification)

Theorem (Chernoff-Hoeffding, Bernoulli scheme)

If $X_1, ..., X_n$ are i.i.d. random variables taking values in $\{0, 1\}$, then for any $\eta > 0$ it holds

$$\mathbb{P}\Big(\mathbb{E}[X_i] - \frac{1}{n}\sum_{i=1}^{n} X_i > \eta\Big) < \exp(-2\eta^2 n)\,.$$

One can express classification error as

$$\mathbb{P}\big(g^{agg}(\mathbf{x}) \neq 1\big) = \mathbb{P}\Big(\frac{1}{B}\sum_{k=1}^{B} g_k(\mathbf{x}) < 0.5\Big)\,.$$

Motivation (classification)

Theorem (Chernoff-Hoeffding, Bernoulli scheme)

If $X_1, ..., X_n$ are i.i.d. random variables taking values in $\{0, 1\}$, then for any $\eta > 0$ it holds

$$\mathbb{P}\Big(\mathbb{E}[X_i] - \frac{1}{n}\sum_{i=1}^{n} X_i > \eta\Big) < \exp(-2\eta^2 n)\,.$$

One can express classification error as

$$\mathbb{P}\big(g^{agg}(\mathbf{x}) \neq 1\big) = \mathbb{P}\Big(\frac{1}{B}\sum_{k=1}^{B} g_k(\mathbf{x}) < 0.5\Big)\,.$$

By a sequence of simple transformations we obtain

$$\underbrace{(1-\epsilon)}_{\mathbb{E}[g_1(\mathbf{x})]} - \frac{1}{B}\sum_{k=1}^{B} g_k(\mathbf{x}) > 0.5 - \epsilon\,.$$

Motivation (classification)

Theorem (Chernoff-Hoeffding, Bernoulli scheme)

If $X_1, ..., X_n$ are i.i.d. random variables taking values in $\{0, 1\}$, then for any $\eta > 0$ it holds

$$\mathbb{P}\Big(\mathbb{E}[X_i] - \frac{1}{n}\sum_{i=1}^{n} X_i > \eta\Big) < \exp(-2\eta^2 n)\,.$$

One can express classification error as

$$\mathbb{P}\big(g^{agg}(\mathbf{x}) \neq 1\big) = \mathbb{P}\Big(\frac{1}{B}\sum_{k=1}^{B} g_k(\mathbf{x}) < 0.5\Big)\,.$$

By a sequence of simple transformations we obtain

$$\underbrace{(1-\epsilon)}_{\mathbb{E}[g_1(\mathbf{x})]} - \frac{1}{B}\sum_{k=1}^{B} g_k(\mathbf{x}) > 0.5 - \epsilon\,.$$

Applying the Chernoff-Hoeffding inequality gives

$$\mathbb{P}\big(g^{agg}(\mathbf{x}) \neq 1\big) < \exp\Big(-\frac{1}{2}B(1-2\epsilon)^2\Big)\,.$$

Motivation

- The derivations from above exploit the **assumption** that random variables $g_1(\mathbf{x}), ..., g_B(\mathbf{x})$ are **independent and identically distributed**.

Motivation

- The derivations from above exploit the **assumption** that random variables $g_1(\mathbf{x}), ..., g_B(\mathbf{x})$ are **independent and identically distributed**.

- As the classifiers $g_1, ..., g_B$ are constructed using the same training sample $\mathcal{D}_n$, the assumption of **independence** is not really credible.

Motivation

- The derivations from above exploit the **assumption** that random variables $g_1(\mathbf{x}), ..., g_B(\mathbf{x})$ are **independent and identically distributed**.

- As the classifiers $g_1, ..., g_B$ are constructed using the same training sample $\mathcal{D}_n$, the assumption of **independence** is not really credible.

- **Remark:** If the variables $g_1, \ldots, g_B$ identically distributed with variance σ^2, but not necessarily independent, with positive pairwise correlation ρ, then

$$\mathbb{V}\left[\frac{1}{B}\sum_{k=1}^{B} g_k\right] =$$

Motivation

- The derivations from above exploit the **assumption** that random variables $g_1(\mathbf{x}), ..., g_B(\mathbf{x})$ are **independent and identically distributed**.

- As the classifiers $g_1, ..., g_B$ are constructed using the same training sample $\mathcal{D}_n$, the assumption of **independence** is not really credible.

- **Remark:** If the variables $g_1, \ldots, g_B$ identically distributed with variance σ^2, but not necessarily independent, with positive pairwise correlation ρ, then

$$\mathbb{V}\left[\frac{1}{B}\sum_{k=1}^{B} g_k\right] = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 .$$

Motivation

- The derivations from above exploit the **assumption** that random variables $g_1(\mathbf{x}), ..., g_B(\mathbf{x})$ are **independent and identically distributed**.

- As the classifiers $g_1, ..., g_B$ are constructed using the same training sample $\mathcal{D}_n$, the assumption of **independence** is not really credible.

- **Remark:** If the variables $g_1, \ldots, g_B$ identically distributed with variance σ^2, but not necessarily independent, with positive pairwise correlation ρ, then

$$\mathbb{V}\left[\frac{1}{B}\sum_{k=1}^{B} g_k\right] = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 .$$

- The idea is thus to **introduce a source of randomness** into the sample used to train each single classifier g_k , $k = 1, ..., B$ to render the pairwise correlation of the g_k's as small as possible.

Motivation

- The derivations from above exploit the **assumption** that random variables $g_1(\mathbf{x}), ..., g_B(\mathbf{x})$ are **independent and identically distributed**.

- As the classifiers $g_1, ..., g_B$ are constructed using the same training sample $\mathcal{D}_n$, the assumption of **independence** is not really credible.

- **Remark:** If the variables $g_1, \ldots, g_B$ identically distributed with variance σ^2, but not necessarily independent, with positive pairwise correlation ρ, then

$$\mathbb{V}\left[\frac{1}{B}\sum_{k=1}^{B} g_k\right] = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 .$$

- The idea is thus to **introduce a source of randomness** into the sample used to train each single classifier g_k , $k = 1, ..., B$ to render the pairwise correlation of the g_k's as small as possible.

- Resort to the idea of the **bootstrap**.

Contents

Bagging (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.

Bagging (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Basic classifier $g(\cdot)$.

Bagging (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Basic classifier $g(\cdot)$.
- Number of estimators to aggregate B.

Bagging (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Basic classifier $g(\cdot)$.
- Number of estimators to aggregate B.

For $k = 1, ..., B$

Bagging (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Basic classifier $g(\cdot)$.
- Number of estimators to aggregate B.

For $k = 1, ..., B$

1. Draw a sample $\mathcal{D}^*_{n,k}$ from $\mathcal{D}_n$ using bootstrap.

Bagging (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Basic classifier $g(\cdot)$.
- Number of estimators to aggregate B.

For $k = 1, ..., B$

1. Draw a sample $\mathcal{D}^*_{n,k}$ from $\mathcal{D}_n$ using bootstrap.
2. Learn g^*_k on $\mathcal{D}^*_{n,k}$.

Bagging (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Basic classifier $g(\cdot)$.
- Number of estimators to aggregate B.

For $k = 1, ..., B$

1. Draw a sample $\mathcal{D}^*_{n,k}$ from $\mathcal{D}_n$ using bootstrap.
2. Learn g^*_k on $\mathcal{D}^*_{n,k}$.

Output: The aggregated classifier $g^{agg}(\cdot) = 1\Big(\frac{1}{B}\sum_{k=1}^{B} g^*_k(\cdot) > 0.5\Big)$.

Bagging (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Basic classifier $g(\cdot)$.
- Number of estimators to aggregate B.

For $k = 1, ..., B$

1. Draw a sample $\mathcal{D}^*_{n,k}$ from $\mathcal{D}_n$ using bootstrap.
2. Learn g^*_k on $\mathcal{D}^*_{n,k}$.

Output: The aggregated classifier $g^{agg}(\cdot) = 1\Big(\frac{1}{B}\sum_{k=1}^{B} g^*_k(\cdot) > 0.5\Big)$.

Classification

- Classify the new observation $\mathbf{x}$ as

$$g^{agg}(\mathbf{x}) = 1\Big(\frac{1}{B}\sum_{k=1}^{B} g^*_k(\mathbf{x}) > 0.5\Big).$$

Drawing bootstrap samples

- **Bootstrap** is a technique based on random sampling, which allows for estimating the sampling distribution of almost any statistics.

Drawing bootstrap samples

- **Bootstrap** is a technique based on random sampling, which allows for estimating the sampling distribution of almost any statistics.
- Bootstrap drawings are represented by B random variables θ_k, $k = 1, ..., B$.

Drawing bootstrap samples

- **Bootstrap** is a technique based on random sampling, which allows for estimating the sampling distribution of almost any statistics.
- Bootstrap drawings are represented by B random variables θ_k, $k = 1, ..., B$.
- In general for the sample consisting of n observations, two techniques are used to draw bootstrap samples:
  - draw (choose randomly) n **observations with replacements**,

Drawing bootstrap samples

- **Bootstrap** is a technique based on random sampling, which allows for estimating the sampling distribution of almost any statistics.
- Bootstrap drawings are represented by B random variables θ_k, $k = 1, ..., B$.
- In general for the sample consisting of n observations, two techniques are used to draw bootstrap samples:
  - draw (choose randomly) n **observations with replacements**,
  - draw (choose randomly) $l < n$ **observations without replacement**.

Drawing bootstrap samples

- **Bootstrap** is a technique based on random sampling, which allows for estimating the sampling distribution of almost any statistics.
- Bootstrap drawings are represented by B random variables θ_k, $k = 1, ..., B$.
- In general for the sample consisting of n observations, two techniques are used to draw bootstrap samples:
  - draw (choose randomly) n **observations with replacements**,
  - draw (choose randomly) $l < n$ **observations without replacement**.
- Thus aggregated classifiers contain two sources of randomness:

Drawing bootstrap samples

- **Bootstrap** is a technique based on random sampling, which allows for estimating the sampling distribution of almost any statistics.
- Bootstrap drawings are represented by B random variables θ_k, $k = 1, ..., B$.
- In general for the sample consisting of n observations, two techniques are used to draw bootstrap samples:
  - draw (choose randomly) n **observations with replacements**,
  - draw (choose randomly) $l < n$ **observations without replacement**.
- Thus aggregated classifiers contain two sources of randomness:
  - due to the $\mathcal{D}_n$ being a random draw from distribution of (X, Y),

Drawing bootstrap samples

- **Bootstrap** is a technique based on random sampling, which allows for estimating the sampling distribution of almost any statistics.
- Bootstrap drawings are represented by B random variables θ_k, $k = 1, ..., B$.
- In general for the sample consisting of n observations, two techniques are used to draw bootstrap samples:
  - draw (choose randomly) n **observations with replacements**,
  - draw (choose randomly) $l < n$ **observations without replacement**.
- Thus aggregated classifiers contain two sources of randomness:
  - due to the $\mathcal{D}_n$ being a random draw from distribution of (X, Y),
  - due to the bootstrap drawing.

Choice of the parameters

- There are two choices to be done:
  - **base classifier** $g(\cdot)$,
  - **number of** bootstrap **iterations** B.

Choice of the parameters

- There are two choices to be done:
  - **base classifier** $g(\cdot)$,
  - **number of** bootstrap **iterations** B.

- Under suitable conditions, given the original sample $\mathcal{D}_n$, by the law of large numbers we have, almost surely

$$\lim_{B\to\infty} g^{agg}(\mathbf{x}) \quad = \quad \lim_{B\to\infty} 1\Big(\frac{1}{B}\sum_{k=1}^{B} g_k^*(\mathbf{x}) > 0.5\Big)$$

Choice of the parameters

- There are two choices to be done:
  - **base classifier** $g(\cdot)$,
  - **number of** bootstrap **iterations** B.

- Under suitable conditions, given the original sample $\mathcal{D}_n$, by the law of large numbers we have, almost surely

$$\begin{aligned} \lim_{B\to\infty} g^{agg}(\mathbf{x}) &= \lim_{B\to\infty} 1\Big(\frac{1}{B}\sum_{k=1}^{B} g_k^*(\mathbf{x}) > 0.5\Big) \\ &= 1\Big(\lim_{B\to\infty} \frac{1}{B}\sum_{k=1}^{B} g_k^*(\mathbf{x}) > 0.5\Big) \end{aligned}$$

Choice of the parameters

- There are two choices to be done:
  - **base classifier** $g(\cdot)$,
  - **number of** bootstrap **iterations** B.

- Under suitable conditions, given the original sample $\mathcal{D}_n$, by the law of large numbers we have, almost surely

$$\begin{aligned}
\lim_{B\to\infty} g^{agg}(\mathbf{x}) &= \lim_{B\to\infty} 1\Big(\frac{1}{B}\sum_{k=1}^{B} g_k^*(\mathbf{x}) > 0.5\Big) \\
&= 1\Big(\lim_{B\to\infty} \frac{1}{B}\sum_{k=1}^{B} g_k^*(\mathbf{x}) > 0.5\Big) \\
&= 1\Big(\mathbb{E}^*[g_k^*(\mathbf{x}) \mid \mathcal{D}_n] > 0.5\Big)\,.
\end{aligned}$$

Choice of the parameters

- There are two choices to be done:
  - **base classifier** $g(\cdot)$,
  - **number of** bootstrap **iterations** B.

- Under suitable conditions, given the original sample $\mathcal{D}_n$, by the law of large numbers we have, almost surely

$$\begin{aligned}
\lim_{B\to\infty} g^{agg}(\mathbf{x}) &= \lim_{B\to\infty} 1\Big(\frac{1}{B}\sum_{k=1}^{B} g_k^*(\mathbf{x}) > 0.5\Big) \\
&= 1\Big(\lim_{B\to\infty} \frac{1}{B}\sum_{k=1}^{B} g_k^*(\mathbf{x}) > 0.5\Big) \\
&= 1\big(\mathbb{E}^*[g_k^*(\mathbf{x}) \mid \mathcal{D}_n] > 0.5\big)\,.
\end{aligned}$$

- So $g^{agg}(\mathbf{x})$ stabilizes with increasing B converging to the **bagging estimator** $1\big(\mathbb{E}^*[g_k^*(\mathbf{x}) \mid \mathcal{D}_n] > 0.5\big)$.

Choice of the parameters

- There are two choices to be done:
  - **base classifier** $g(\cdot)$,
  - **number of** bootstrap **iterations** B.

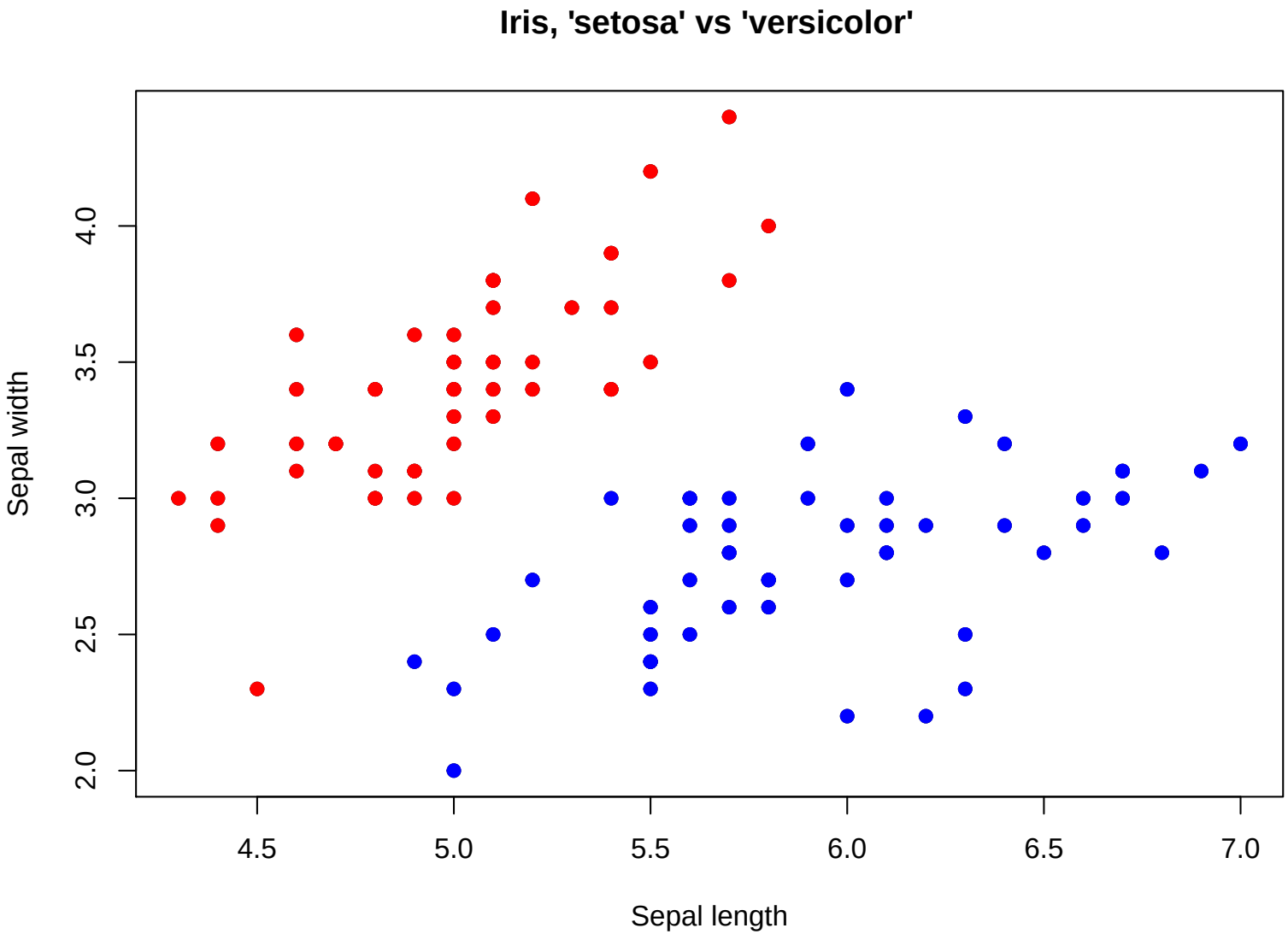
- Under suitable conditions, given the original sample $\mathcal{D}_n$, by the law of large numbers we have, almost surely

$$\begin{aligned}
\lim_{B\to\infty} g^{agg}(\mathbf{x}) &= \lim_{B\to\infty} 1\Big(\frac{1}{B}\sum_{k=1}^{B} g_k^*(\mathbf{x}) > 0.5\Big) \\
&= 1\Big(\lim_{B\to\infty} \frac{1}{B}\sum_{k=1}^{B} g_k^*(\mathbf{x}) > 0.5\Big) \\
&= 1\big(\mathbb{E}^*[g_k^*(\mathbf{x}) \mid \mathcal{D}_n] > 0.5\big)\,.
\end{aligned}$$

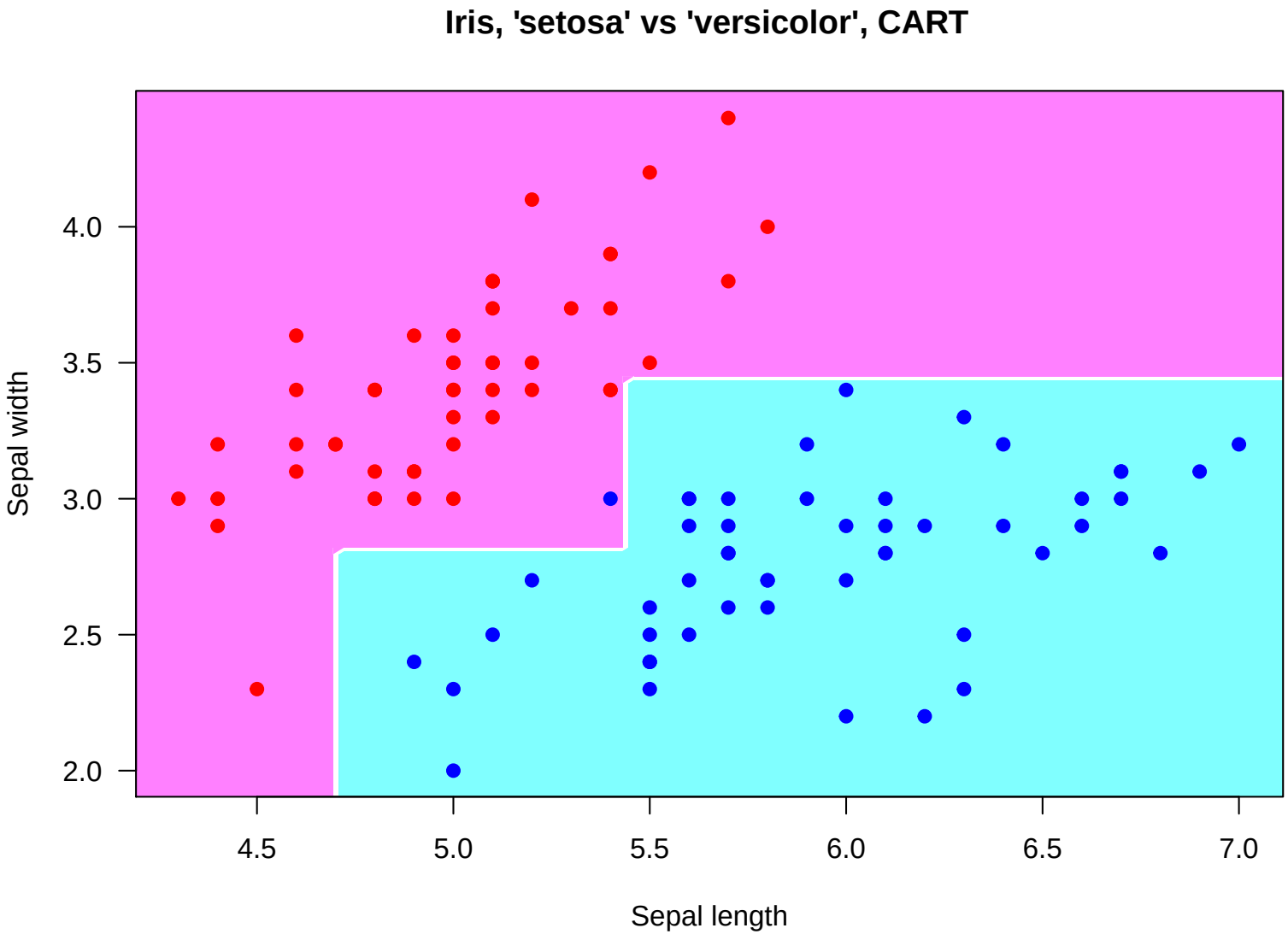
- So $g^{agg}(\mathbf{x})$ stabilizes with increasing B converging to the **bagging estimator** $1\big(\mathbb{E}^*[g_k^*(\mathbf{x}) \mid \mathcal{D}_n] > 0.5\big)$.

- Thus, B should be chosen as large as possible, regarding computational capabilities.

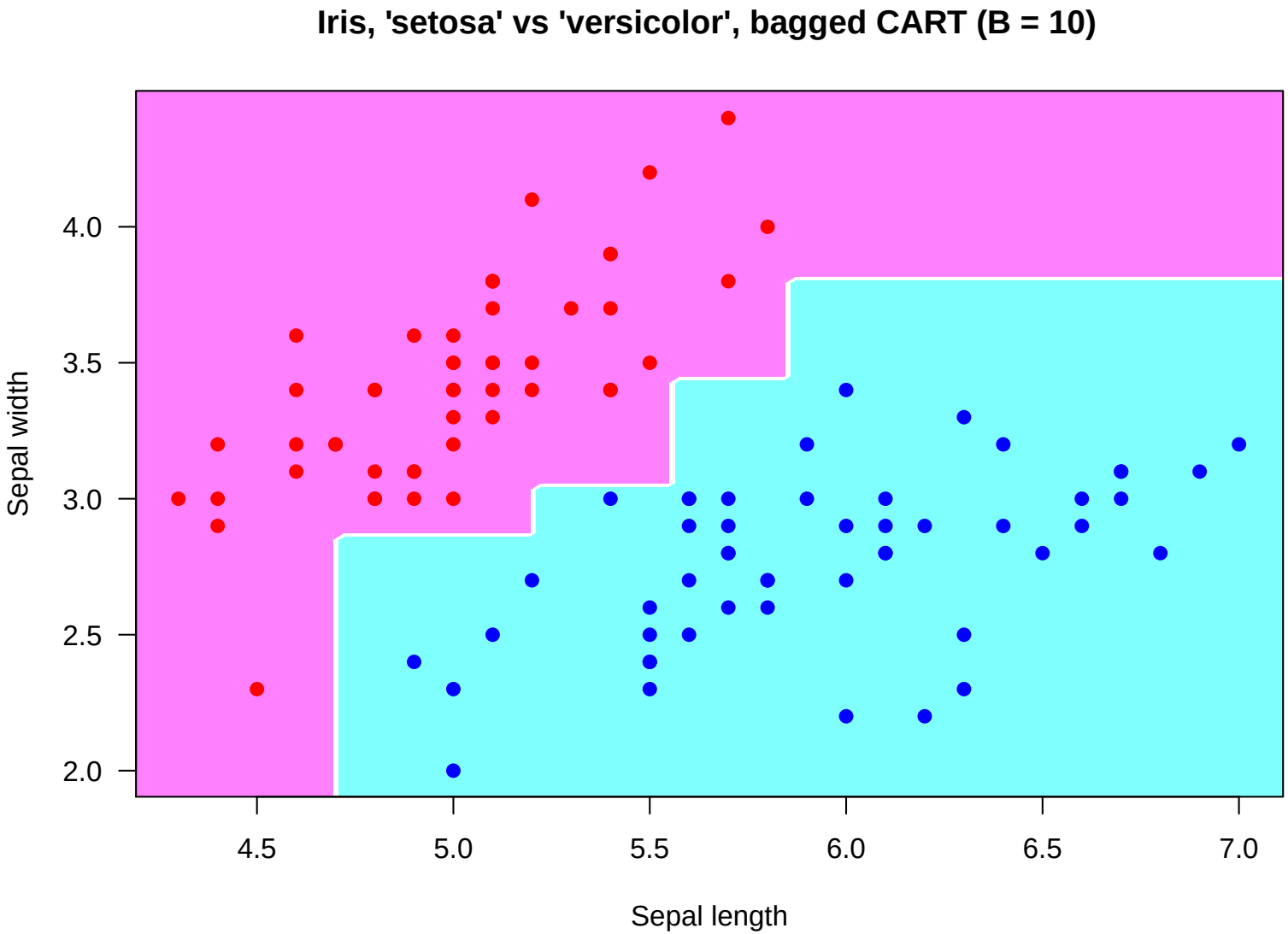
Bagging classification tree: iris data



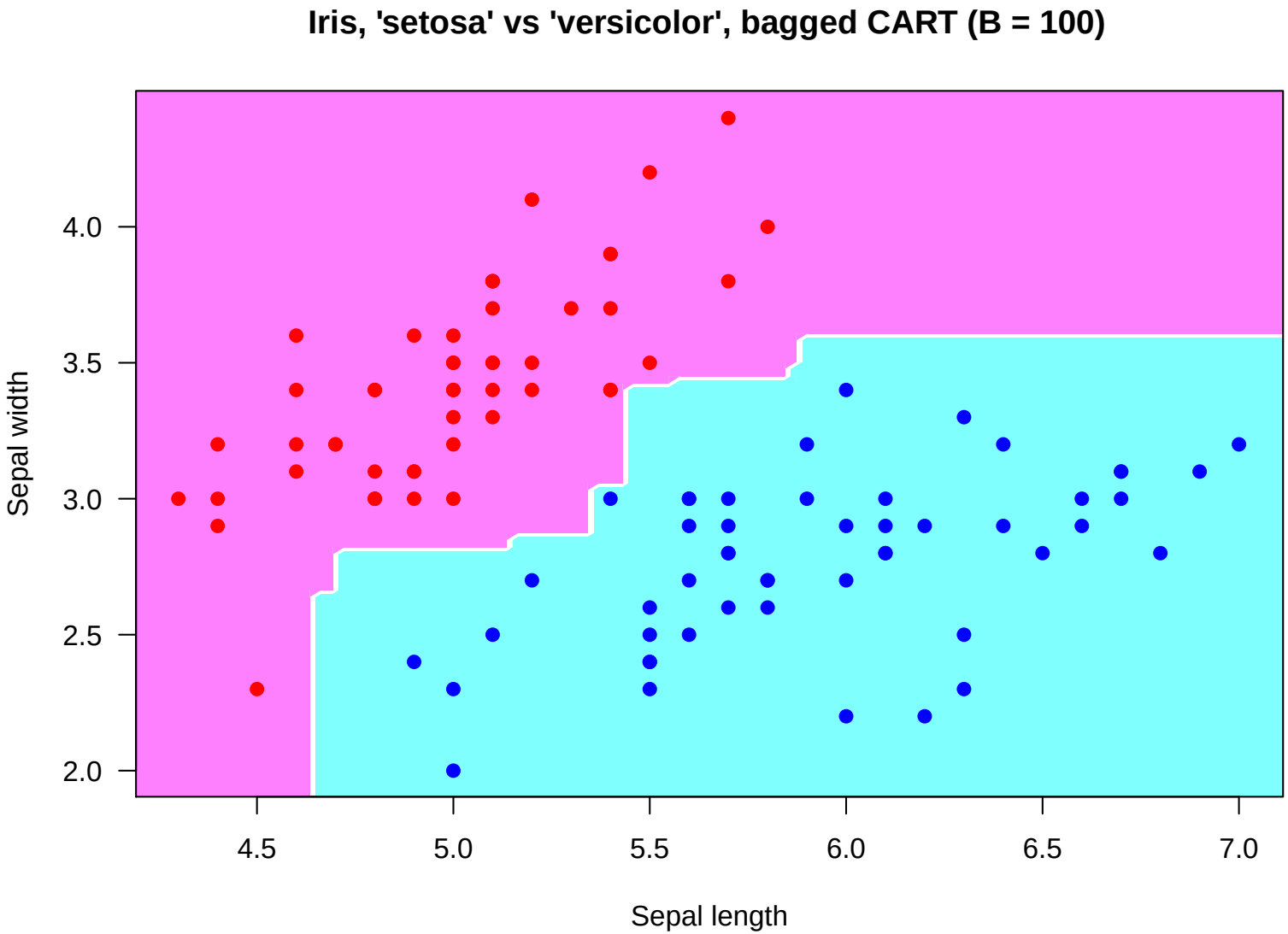
Bagging classification tree: iris data



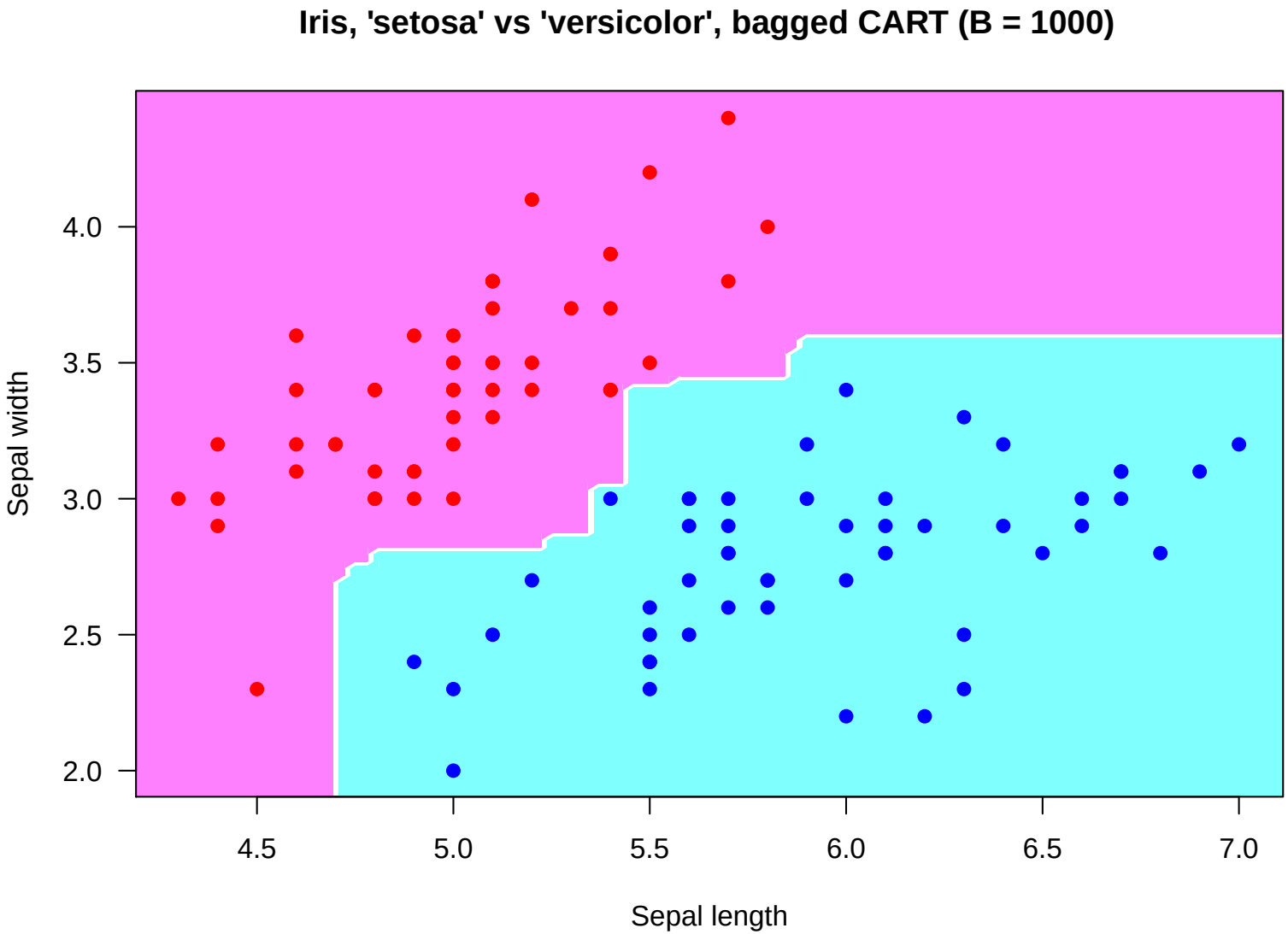
Bagging classification tree: iris data



Bagging classification tree: iris data



Bagging classification tree: iris data



Properties and recommendations

- Bagging a good classifier can make it better, bagging a bad classifier can make it worse.

Properties and recommendations

- Bagging a good classifier can make it better, bagging a bad classifier can make it worse.
- Significant improvement by bagging is not expected on large data sets because there bootstrap samples are very similar. Subsampling is expected to improve things.

Properties and recommendations

- Bagging a good classifier can make it better, bagging a bad classifier can make it worse.
- Significant improvement by bagging is not expected on large data sets because there bootstrap samples are very similar. Subsampling is expected to improve things.
- Bagging could be improved by using a robust location estimator instead a mean over the B bootstrapped classifiers. Taking the median yields the so-called *bragging* (Buhlmann, 2003). Trimmed means is another option.

Properties and recommendations

- Bagging a good classifier can make it better, bagging a bad classifier can make it worse.
- Significant improvement by bagging is not expected on large data sets because there bootstrap samples are very similar. Subsampling is expected to improve things.
- Bagging could be improved by using a robust location estimator instead a mean over the B bootstrapped classifiers. Taking the median yields the so-called *bragging* (Buhlmann, 2003). Trimmed means is another option.
- Bagging reduces interpretability of the classifier because any simple structure in the model is lost.

Contents

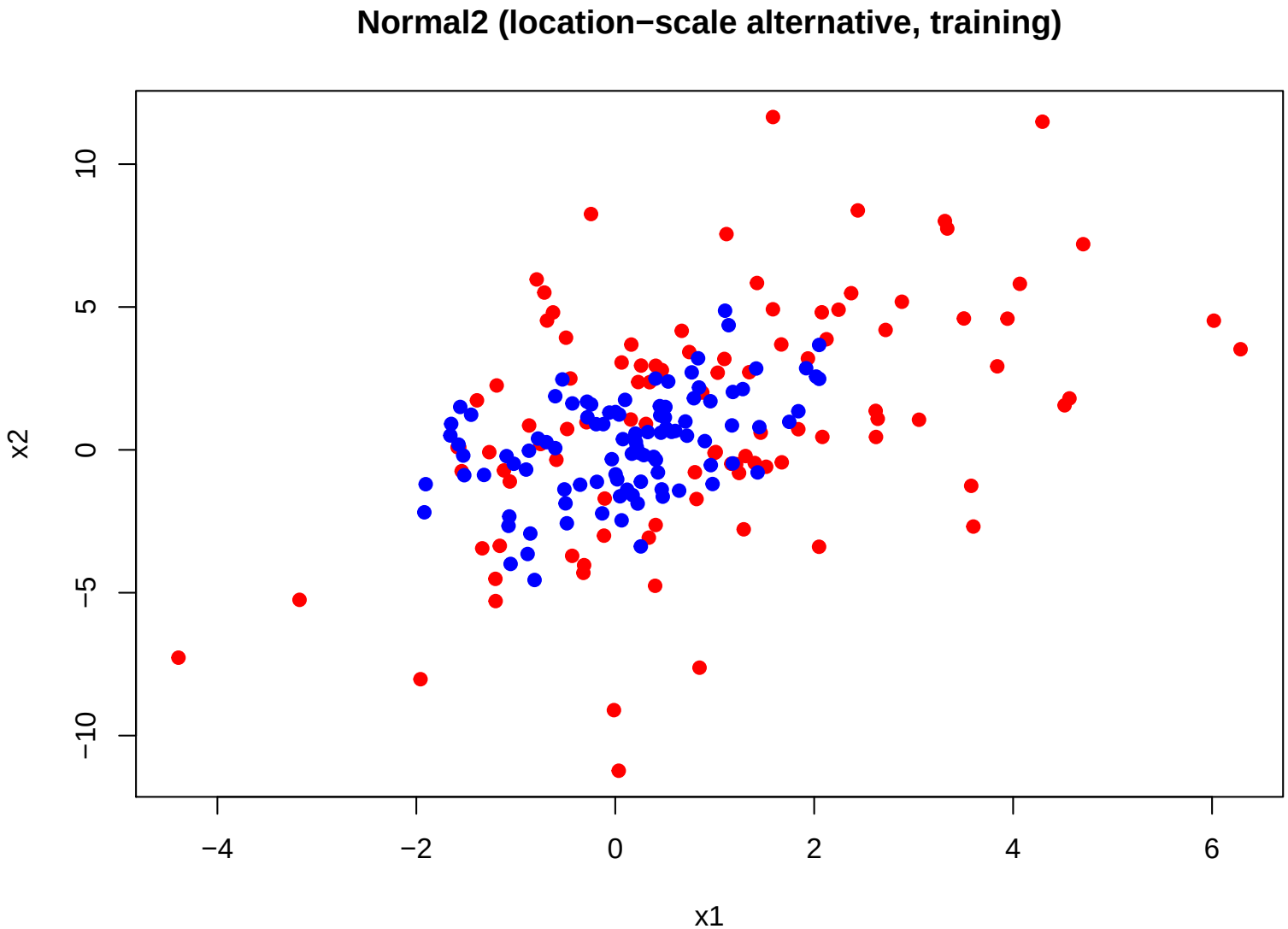
Normal2 data generation

- Generate independent copies of a bivariate normal vector:
  - for $Y = 0$ mean $(0, 0)$ and

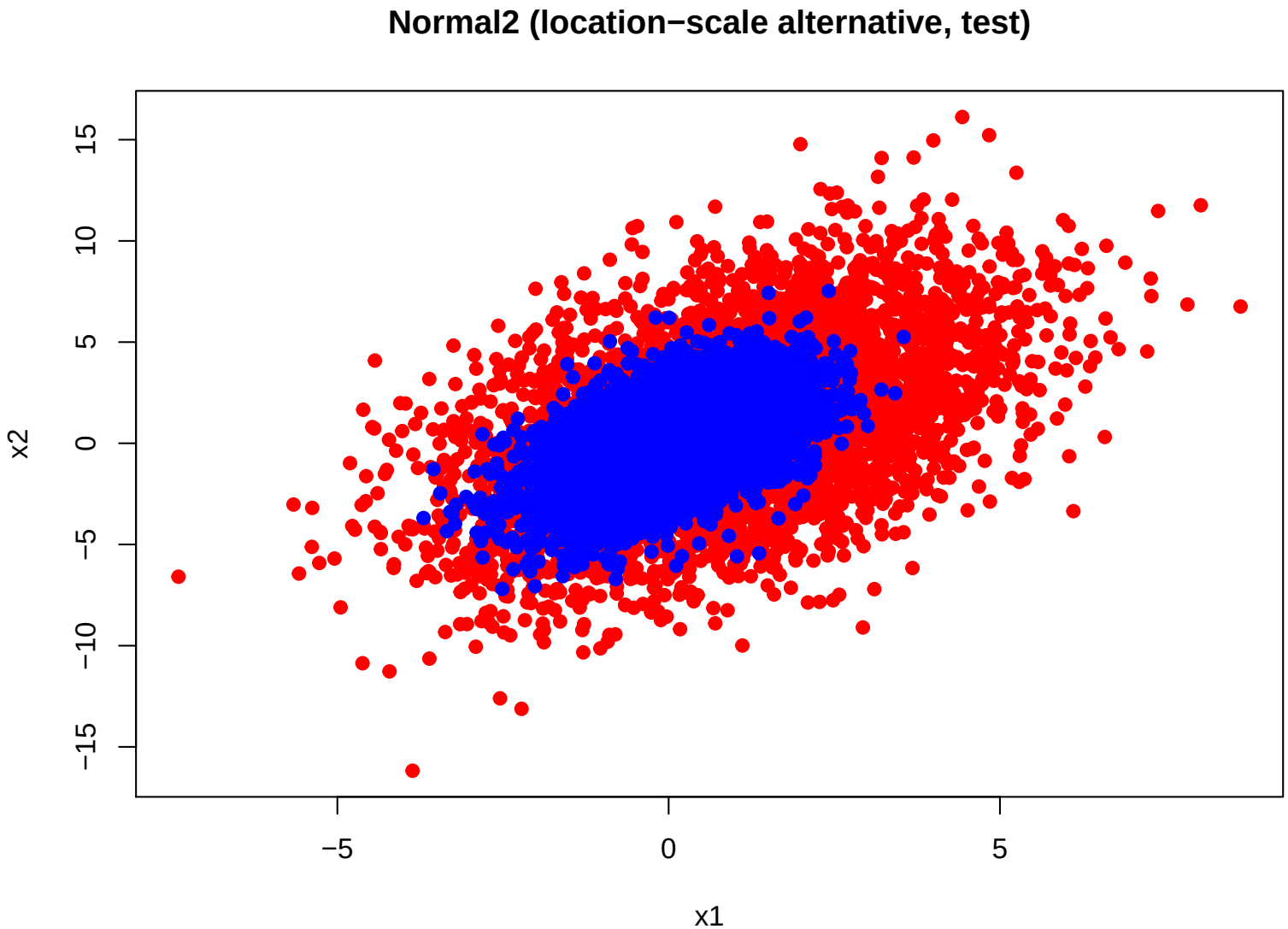
$$\Sigma = \begin{pmatrix} 1 & 1 \\ 1 & 4 \end{pmatrix}$$

  - for $Y = 1$ mean $(0, 0)$ and the variance is equal to 4Σ

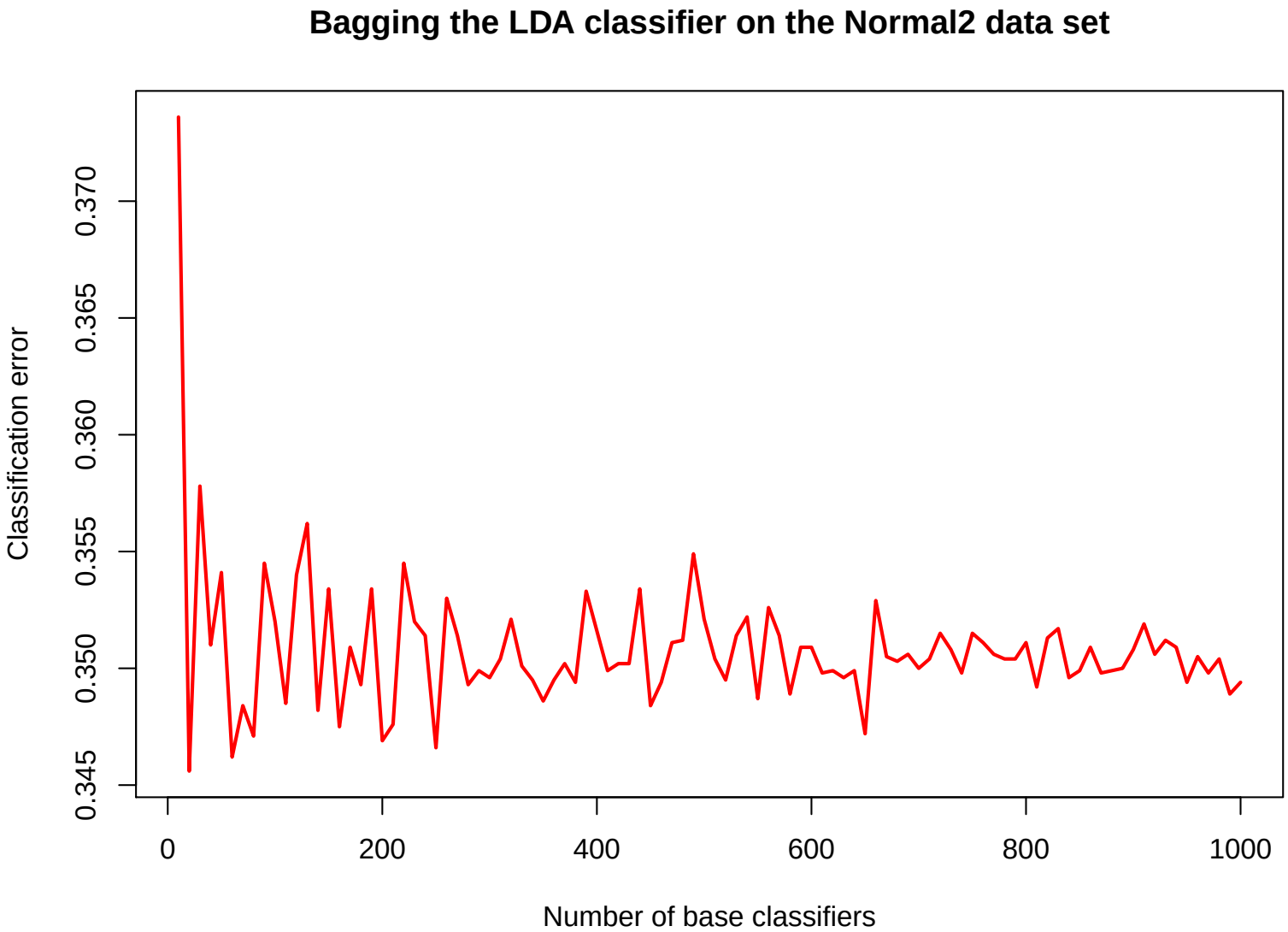
Normal2 data



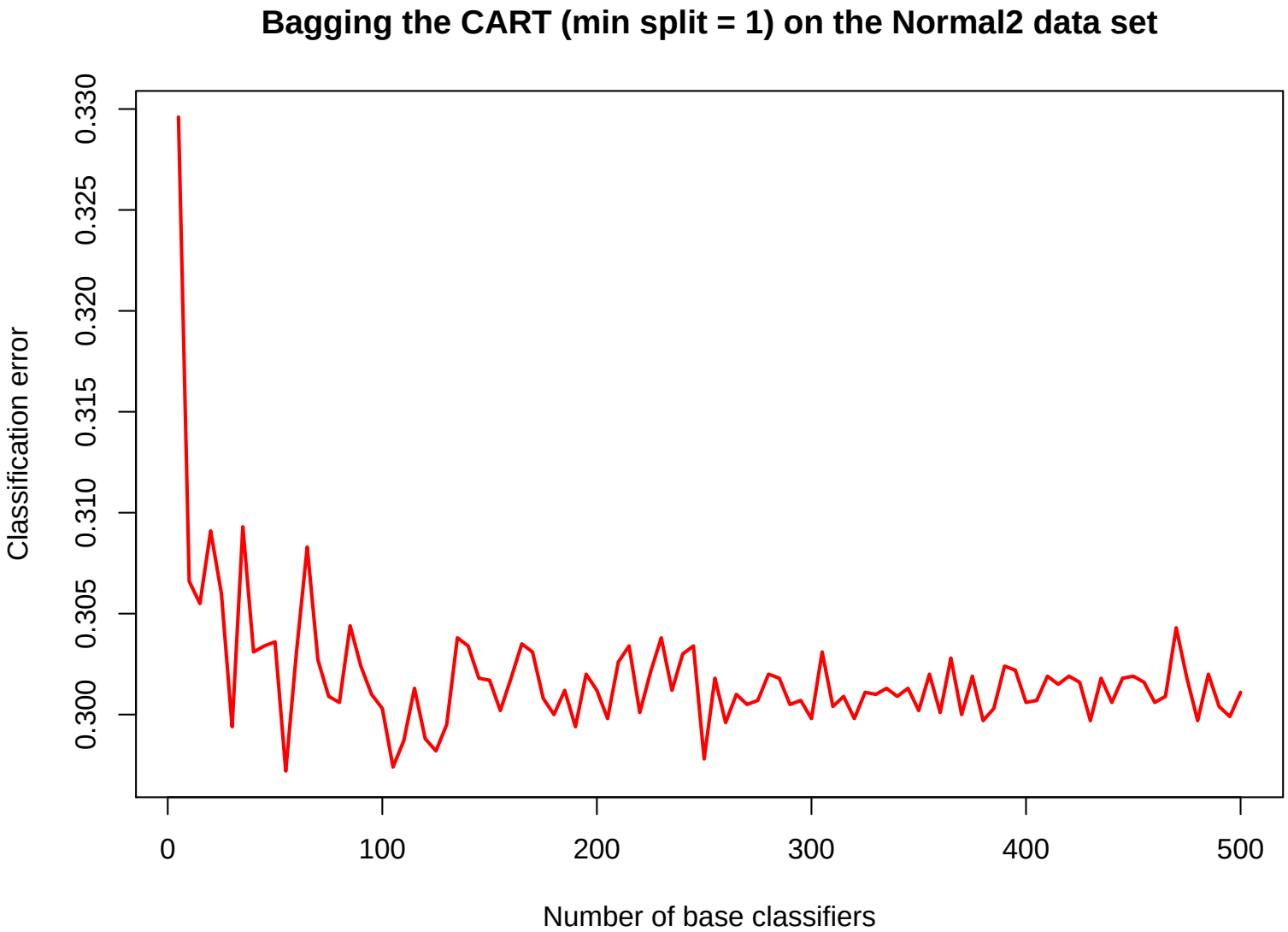
Normal2 data



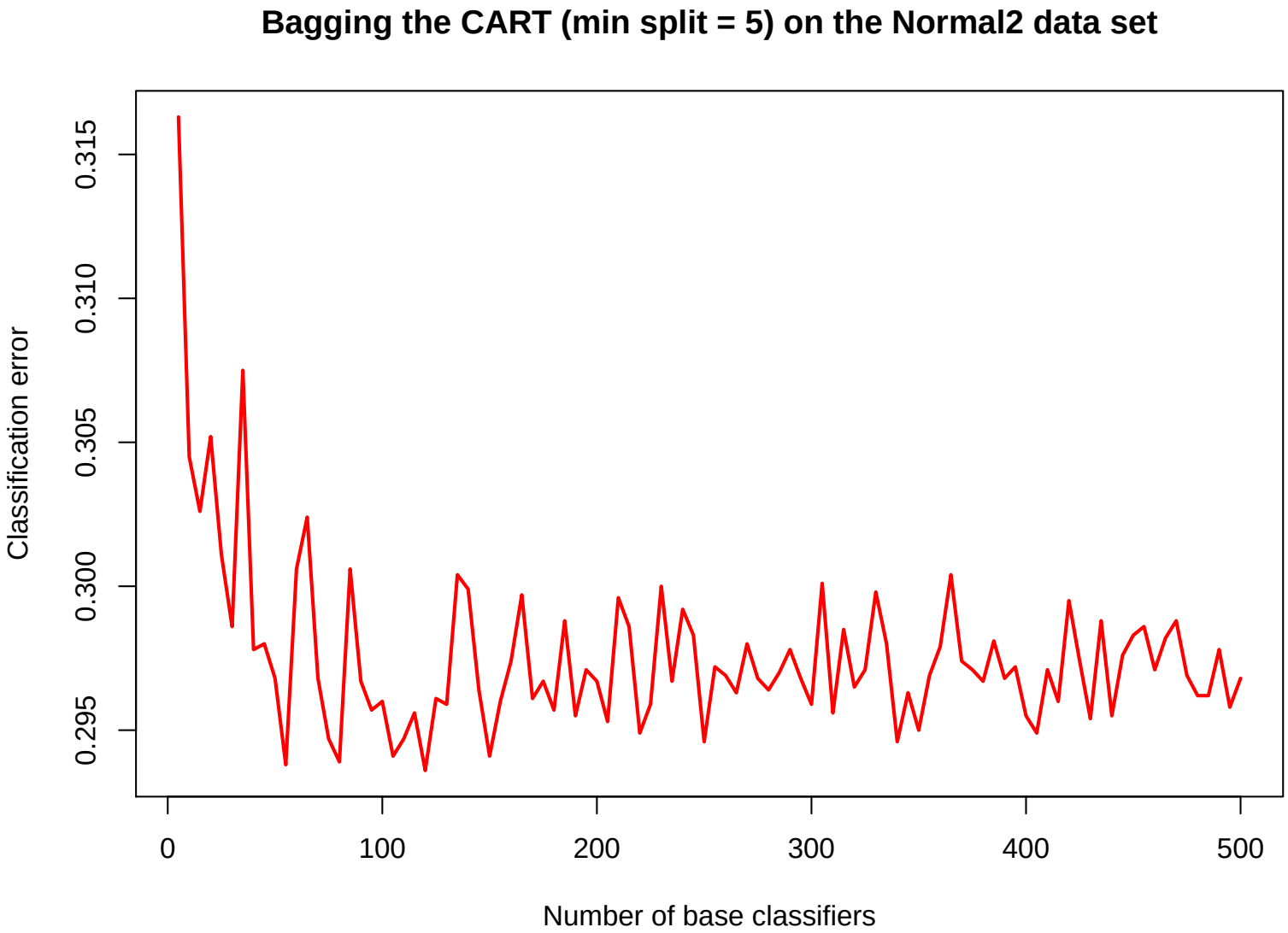
Bagging LDA: Normal2 data



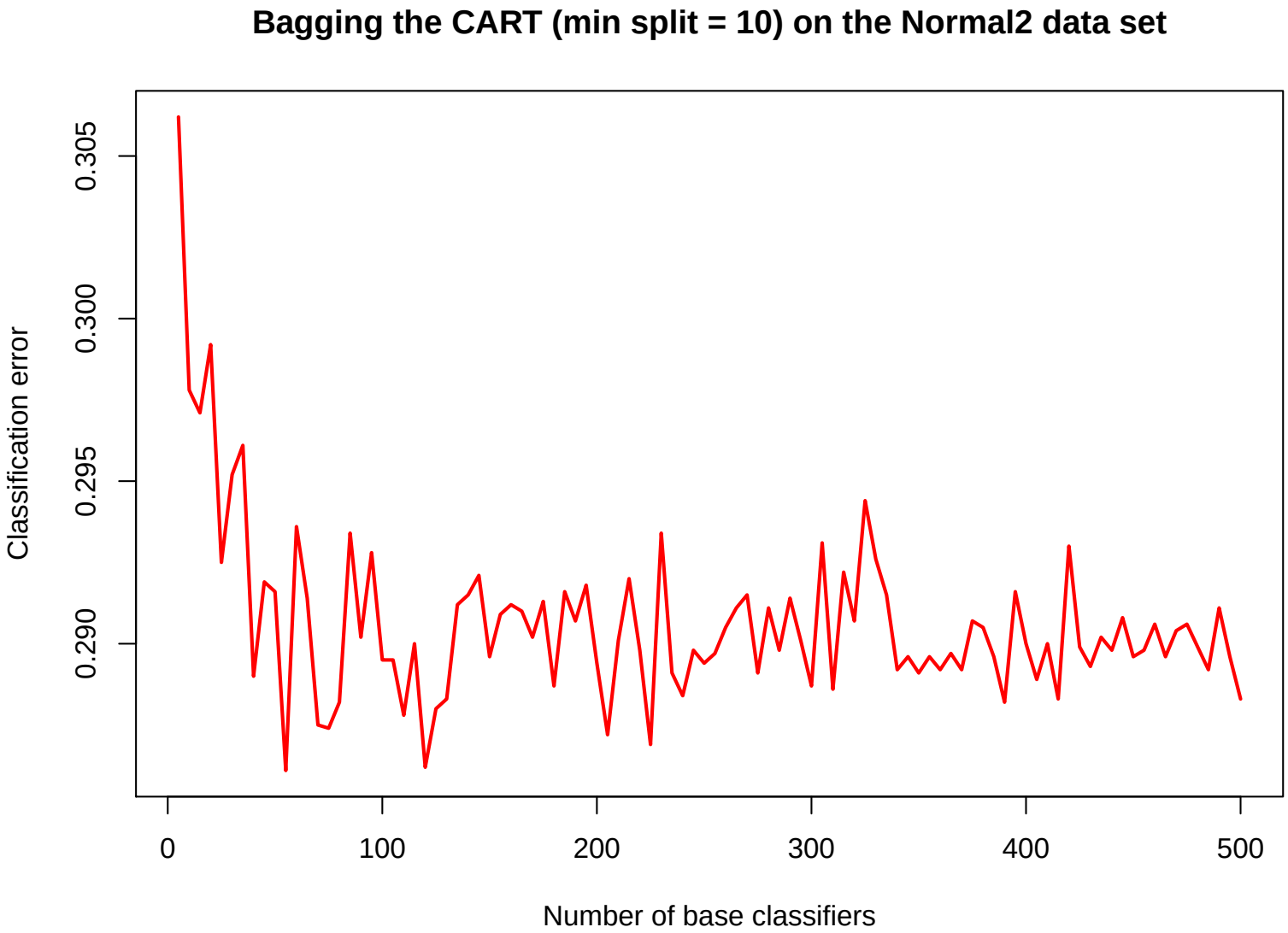
Bagging classification tree: Normal2 data



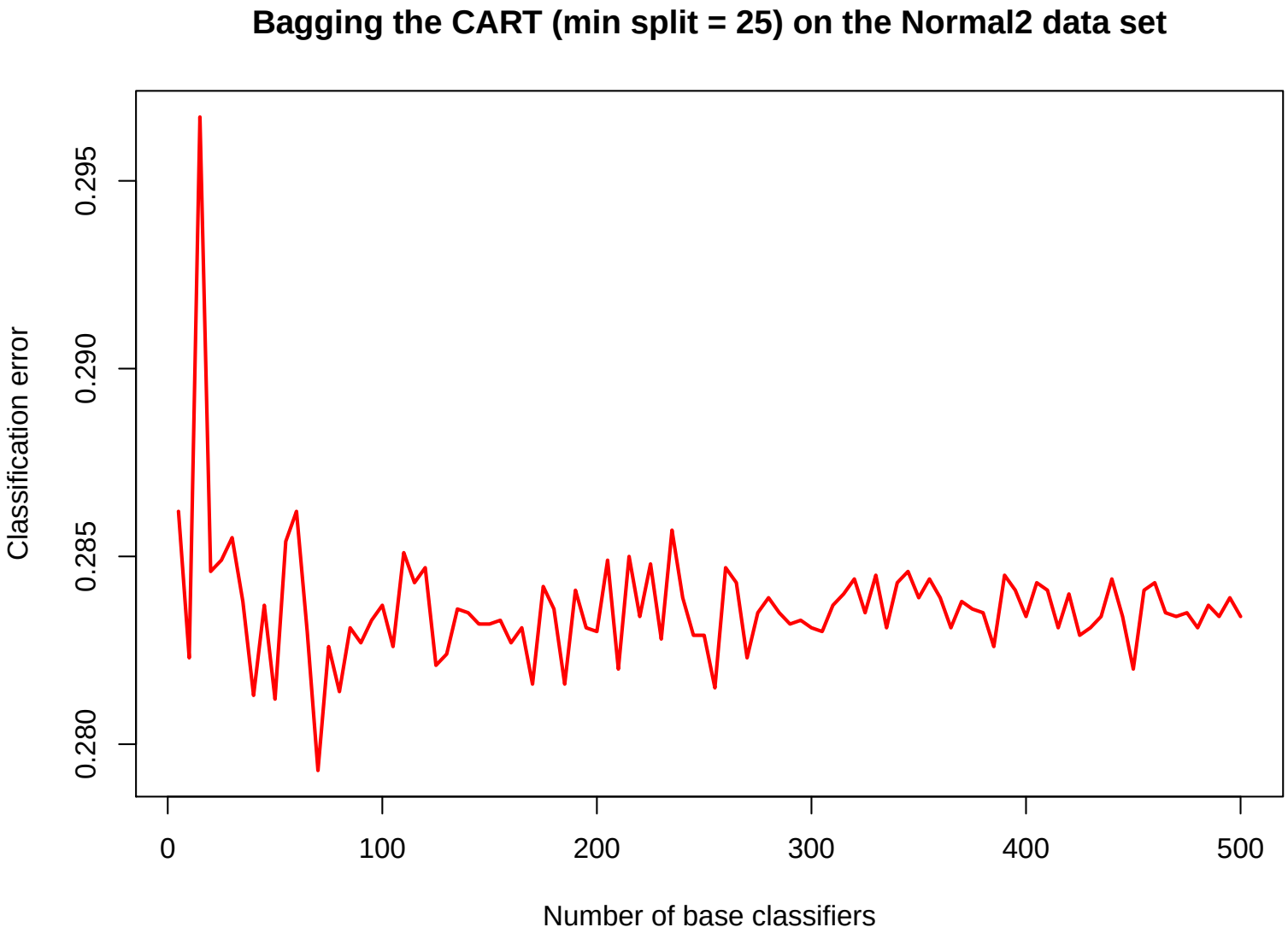
Bagging classification tree: Normal2 data



Bagging classification tree: Normal2 data



Bagging classification tree: Normal2 data



Contents

Contents

The key idea

- Random forest is a **collection of trees**.

The key idea

- Random forest is a **collection of trees**.

- Random forests have been introduced by **Léo Breiman** in the early 2000s. The following web-page is dedicated to random forests: `http://www.stat.berkeley.edu/~breiman/RandomForests/`

The key idea

- Random forest is a **collection of trees**.
- Random forests have been introduced by **Léo Breiman** in the early 2000s. The following web-page is dedicated to random forests: `http://www.stat.berkeley.edu/~breiman/RandomForests/`
- Random forests can be seen as a **modified version of bagging** as they also aggregate trees taught on the bootstrap samples.

The key idea

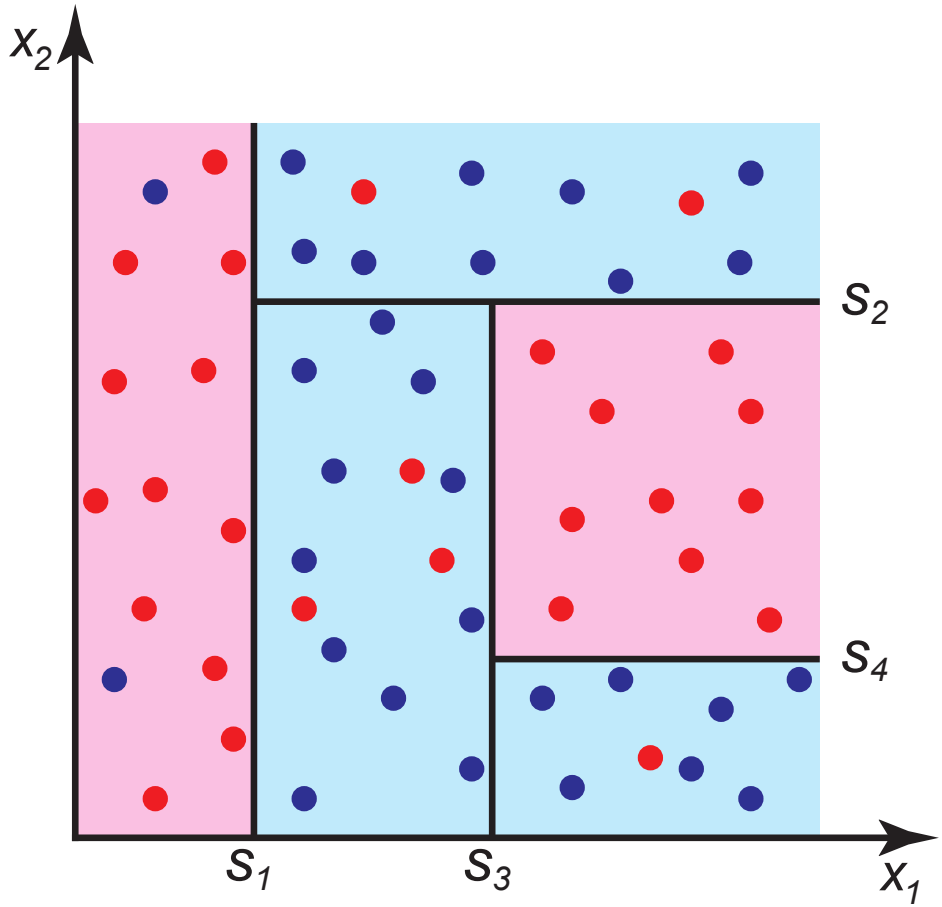
- Random forest is a **collection of trees**.
- Random forests have been introduced by **Léo Breiman** in the early 2000s. The following web-page is dedicated to random forests: `http://www.stat.berkeley.edu/~breiman/RandomForests/`
- Random forests can be seen as a **modified version of bagging** as they also aggregate trees taught on the bootstrap samples.
- However, random forests introduce a substantial modification of bagging that builds a large collection of *de-correlated* trees.

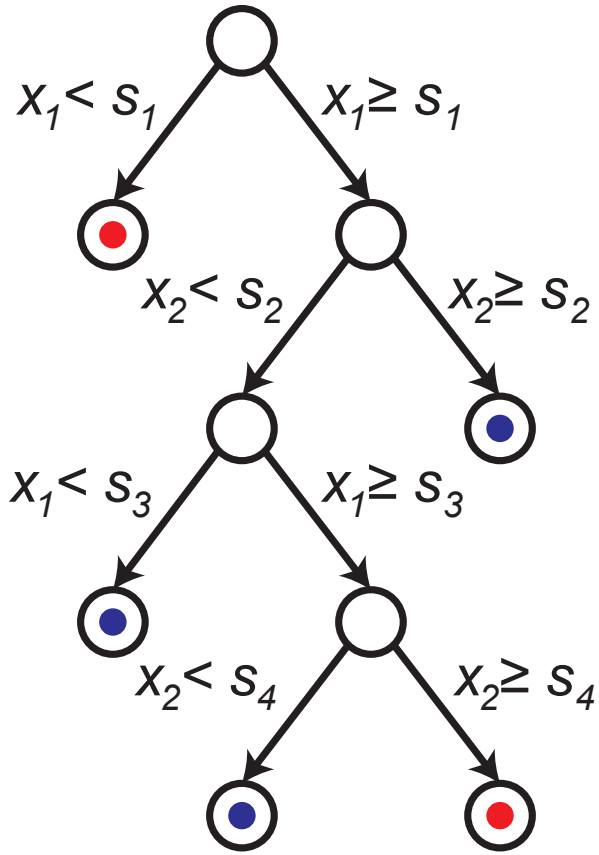
The key idea

- Random forest is a **collection of trees**.
- Random forests have been introduced by **Léo Breiman** in the early 2000s. The following web-page is dedicated to random forests: `http://www.stat.berkeley.edu/~breiman/RandomForests/`
- Random forests can be seen as a **modified version of bagging** as they also aggregate trees taught on the bootstrap samples.
- However, random forests introduce a substantial modification of bagging that builds a large collection of *de-correlated* trees.
- Let $T_k(\mathbf{x})$, $k = 1, ..., B$ be tree-similar classifiers ($T_k : \mathbb{R}^d \to \{0, 1\}$). The **random forest** classifier assigns new observation $\mathbf{x}$ by aggregating these:

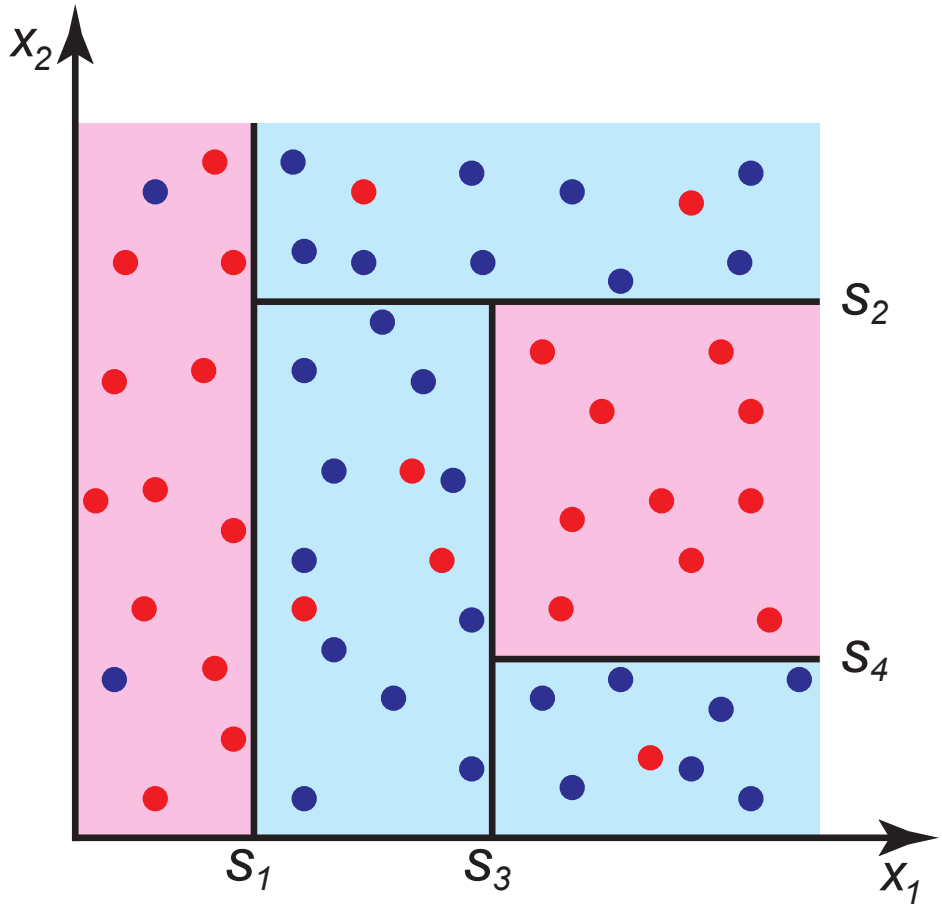
$$T^{RF}(\mathbf{x}) = 1\Big(\frac{1}{B}\sum_{k=1}^{B} T_k(\mathbf{x}) > 0.5\Big).$$

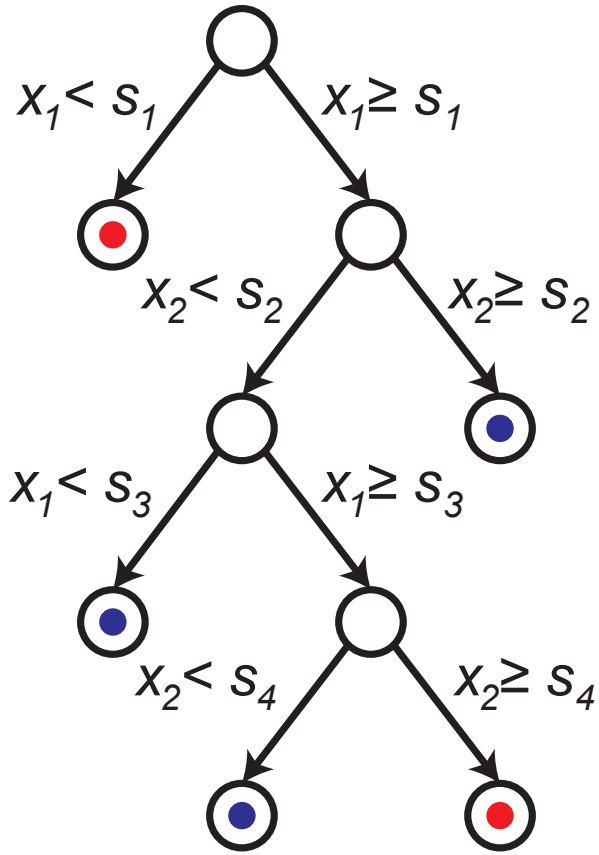
The key idea


x_2
s_2
s_4
s_1
s_3
x_1


$x_1 < s_1$
$x_1 \geq s_1$
$x_2 < s_2$
$x_2 \geq s_2$
$x_1 < s_3$
$x_1 \geq s_3$
$x_2 < s_4$
$x_2 \geq s_4$

The key idea





- **To reduce correlation between trees**, Breiman proposes
  - first, **randomly select** m variables out of all d variables,
  - next, pick the best variable/split-point among the m.

Random forests (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.

Random forests (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Number of trees B; minimum number of observations for a node to be split n_{min}; impurity criterion Q.

Random forests (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Number of trees B; minimum number of observations for a node to be split n_{min}; impurity criterion Q.
- number of variables to use when splitting $m \in \{1, ..., d\}$.

Random forests (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Number of trees B; minimum number of observations for a node to be split n_{min}; impurity criterion Q.
- number of variables to use when splitting $m \in \{1, ..., d\}$.

For $k = 1, ..., B$

Random forests (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Number of trees B; minimum number of observations for a node to be split n_{min}; impurity criterion Q.
- number of variables to use when splitting $m \in \{1, ..., d\}$.

For $k = 1, ..., B$

1. Draw a sample $\mathcal{D}^*_{n,k}$ from $\mathcal{D}_n$ using bootstrap.

Random forests (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Number of trees B; minimum number of observations for a node to be split n_{min}; impurity criterion Q.
- number of variables to use when splitting $m \in \{1, ..., d\}$.

For $k = 1, ..., B$

1. Draw a sample $\mathcal{D}^*_{n,k}$ from $\mathcal{D}_n$ using bootstrap.
2. Learn the classification tree T^*_k on $\mathcal{D}^*_{n,k}$; each time when splitting a node, search optimal variable among m variables randomly chosen out of all d variables.

Random forests (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Number of trees B; minimum number of observations for a node to be split n_{min}; impurity criterion Q.
- number of variables to use when splitting $m \in \{1, ..., d\}$.

For $k = 1, ..., B$

1. Draw a sample $\mathcal{D}^*_{n,k}$ from $\mathcal{D}_n$ using bootstrap.
2. Learn the classification tree T^*_k on $\mathcal{D}^*_{n,k}$; each time when splitting a node, search optimal variable among m variables randomly chosen out of all d variables.

Output: The aggregated classifier $T^{RF}(\cdot) = 1\Big(\frac{1}{B}\sum_{k=1}^{B} T^*_k(\cdot) > 0.5\Big)$.

Random forests (algorithm)

Training

Input:

- Training sample $\big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Number of trees B; minimum number of observations for a node to be split n_{min}; impurity criterion Q.
- number of variables to use when splitting $m \in \{1, ..., d\}$.

For $k = 1, ..., B$

1. Draw a sample $\mathcal{D}^*_{n,k}$ from $\mathcal{D}_n$ using bootstrap.
2. Learn the classification tree T^*_k on $\mathcal{D}^*_{n,k}$; each time when splitting a node, search optimal variable among m variables randomly chosen out of all d variables.

Output: The aggregated classifier $T^{RF}(\cdot) = 1\Big(\frac{1}{B}\sum_{k=1}^{B} T^*_k(\cdot) > 0.5\Big)$.

Classification

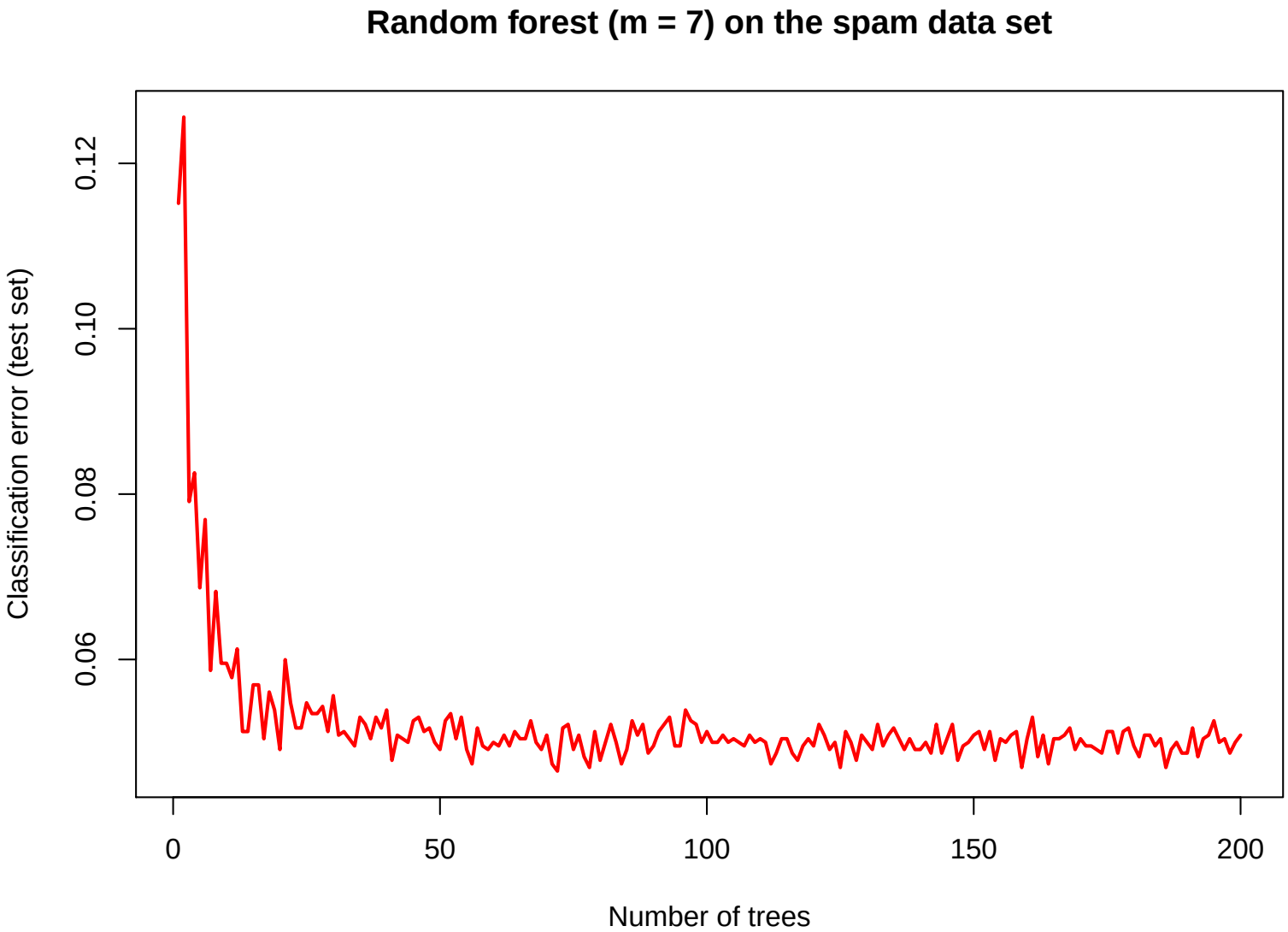
- Classify the new observation $\mathbf{x}$ as

$$T^{RF}(\mathbf{x}) = 1\Big(\frac{1}{B}\sum_{k=1}^{B} T^*_k(\mathbf{x}) > 0.5\Big).$$

Spam data

- See Hastie *et al.* (2009, ch.1)
- A standard data set consisting of information from 4601 email messages
- The purpose is to predict if the email message is a spam or not
- For all 4601 email messages, the following information is available
  - the true outcome (email type) email or spam is available
  - the relative frequencies of 57 of the most commonly occurring words and punctuation marks in the email message.

Random forests: spam data



Properties

- There are **two** introduced **sources of randomness**:

Properties

- There are **two** introduced **sources of randomness**:
  - B **bootstrap samples**,

Properties

- There are **two** introduced **sources of randomness**:
  - B **bootstrap samples**,
  - m **variables randomly chosen** out of d when splitting each tree node.

Properties

- There are **two** introduced **sources of randomness**:
  - B **bootstrap samples**,
  - m **variables randomly chosen** out of d when splitting each tree node.
- The method is simple, implementations are available in numerous software, *e.g.* `R`-package `randomForest`.

Properties

- There are **two** introduced **sources of randomness**:
  - B **bootstrap samples**,
  - m **variables randomly chosen** out of d when splitting each tree node.
- The method is simple, implementations are available in numerous software, *e.g.* `R`-package `randomForest`.
- The classifier is known for relatively **high speed** of training and classification.

Properties

- There are **two** introduced **sources of randomness**:
  - B **bootstrap samples**,
  - m **variables randomly chosen** out of d when splitting each tree node.
- The method is simple, implementations are available in numerous software, *e.g.* `R`-package `randomForest`.
- The classifier is known for relatively **high speed** of training and classification.
- The classifier is known for its relatively **precise prediction on complex data**, *i.e.* those including many variables, missing entries, *etc*.

Properties

- There are **two** introduced **sources of randomness**:
  - B **bootstrap samples**,
  - m **variables randomly chosen** out of d when splitting each tree node.
- The method is simple, implementations are available in numerous software, *e.g.* `R`-package `randomForest`.
- The classifier is known for relatively **high speed** of training and classification.
- The classifier is known for its relatively **precise prediction on complex data**, *i.e.* those including many variables, missing entries, *etc*.
- Classifier has **limited sensibility w.r.t. to the choice of parameters**: B, m, n_{min}.

Choice of number m of variables for a node

- Parameter m is related to the **dependence between** single **trees**.

Choice of number m of variables for a node

- Parameter m is related to the **dependence between** single **trees**.
- The **lower** is m:

Choice of number m of variables for a node

- Parameter m is related to the **dependence between** single **trees**.
- The **lower** is m:
  - to larger extent the variables at which to split each node **are chosen randomly**,
    thus the more **different** are single **trees**,
    thus the more **independent** are single **trees**,

Choice of number m of variables for a node

- Parameter m is related to the **dependence between** single **trees**.
- The **lower** is m:
  - to larger extent the variables at which to split each node **are chosen randomly**,
    thus the more **different** are single **trees**,
    thus the more **independent** are single **trees**,
  - the **lower** is the **prediction accuracy of** each single **tree**,
    and thus of the entire forest as well.

Choice of number m of variables for a node

- Parameter m is related to the **dependence between** single **trees**.
- The **lower** is m:
  - to larger extent the variables at which to split each node **are chosen randomly**,
    thus the more **different** are single **trees**,
    thus the more **independent** are single **trees**,
  - the **lower** is the **prediction accuracy of** each single **tree**,
    and thus of the entire forest as well.
- The **higher** is m: *vice versa*.

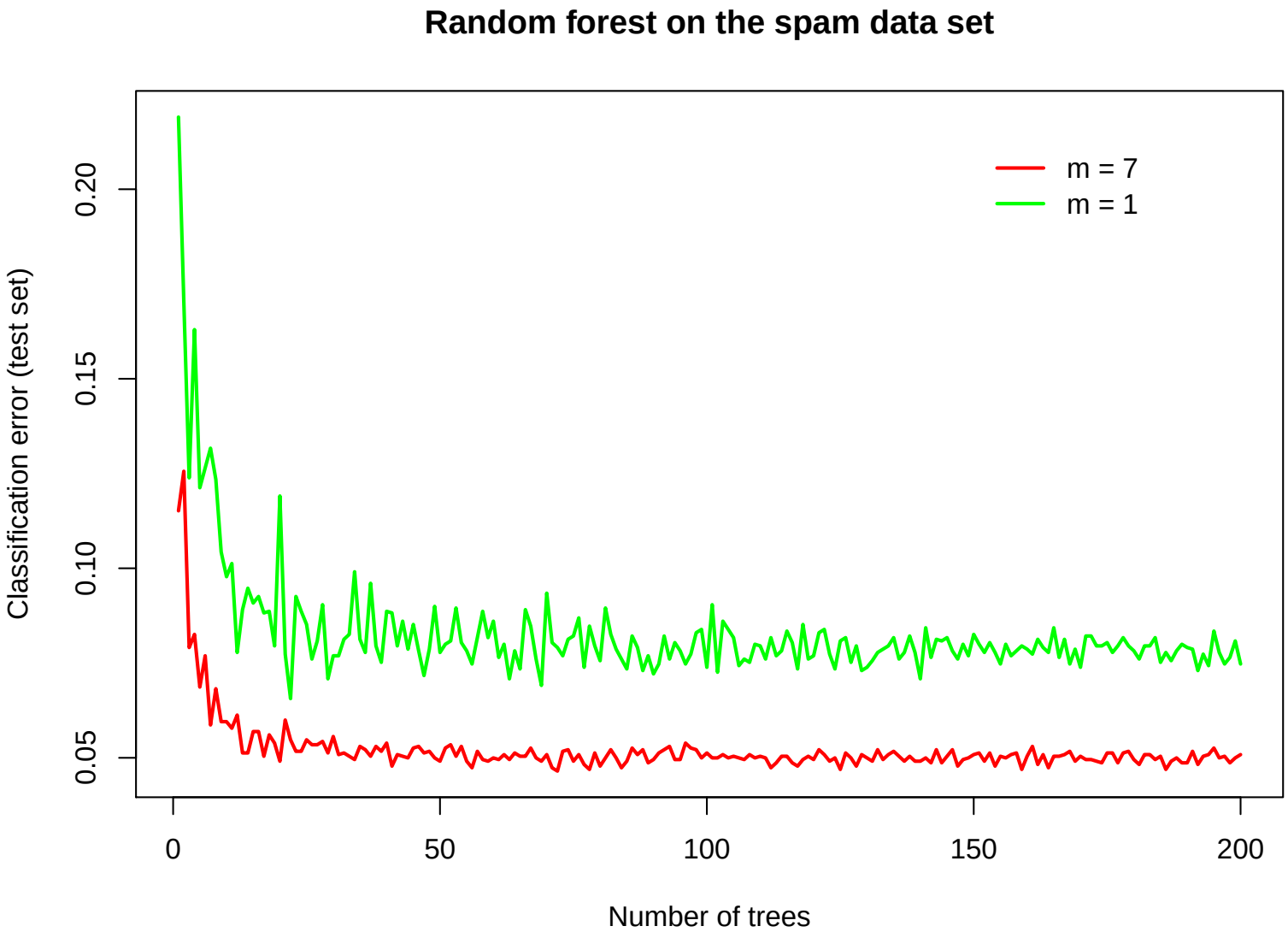
Choice of number m of variables for a node

- Parameter m is related to the **dependence between** single **trees**.
- The **lower** is m:
  - to larger extent the variables at which to split each node **are chosen randomly**,
    thus the more **different** are single **trees**,
    thus the more **independent** are single **trees**,
  - the **lower** is the **prediction accuracy of** each single **tree**,
    and thus of the entire forest as well.
- The **higher** is m: *vice versa*.
- It is recommended to check the performance of the random forests for **different choices of** m.

Choice of number m of variables for a node

- Parameter m is related to the **dependence between** single **trees**.
- The **lower** is m:
  - to larger extent the variables at which to split each node **are chosen randomly**,
    thus the more **different** are single **trees**,
    thus the more **independent** are single **trees**,
  - the **lower** is the **prediction accuracy of** each single **tree**,
    and thus of the entire forest as well.
- The **higher** is m: *vice versa*.
- It is recommended to check the performance of the random forests for **different choices of** m.
- The **inventors recommend** $m = \lfloor \sqrt{d} \rfloor$
  (the default value in `R`-package `randomForest`).

Random forests: spam data



Contents

Performance and interpretation

- It is desirable to measure the performance of the random forests, as of any other classification technique, in terms of the error probability:

$$R(T^{RF}) = \mathbb{P}\big(T^{RF}(X) \neq Y\big)\,.$$

Performance and interpretation

- It is desirable to measure the performance of the random forests, as of any other classification technique, in terms of the error probability:

$$R(T^{RF}) = \mathbb{P}\big(T^{RF}(X) \neq Y\big)\,.$$

- As usual, the error can be measured:

Performance and interpretation

- It is desirable to measure the performance of the random forests, as of any other classification technique, in terms of the error probability:

$$R(T^{RF}) = \mathbb{P}\left(T^{RF}(X) \neq Y\right).$$

- As usual, the error can be measured:
  - For a **probability distribution**: by a **simulation study**, *i.e.* train T^{RF} and measure its classification error for a number of simulated data sets.

Performance and interpretation

- It is desirable to measure the performance of the random forests, as of any other classification technique, in terms of the error probability:

$$R(T^{RF}) = \mathbb{P}\left(T^{RF}(X) \neq Y\right).$$

- As usual, the error can be measured:
  - For a **probability distribution**: by a **simulation study**, *i.e.* train T^{RF} and measure its classification error for a number of simulated data sets.
  - For **given data**: by **splitting data** into training and test subsets, by **iterating this splitting** if data are small, or by **cross-validation**.

Performance and interpretation

- It is desirable to measure the performance of the random forests, as of any other classification technique, in terms of the error probability:

$$R(T^{RF}) = \mathbb{P}\left(T^{RF}(X) \neq Y\right).$$

- As usual, the error can be measured:
  - For a **probability distribution**: by a **simulation study**, *i.e.* train T^{RF} and measure its classification error for a number of simulated data sets.
  - For **given data**: by **splitting data** into training and test subsets, by **iterating this splitting** if data are small, or by **cross-validation**.
- Random forests offer an additional possibility to directly estimate classification error exploiting the **out-of-bag (OOB)** principle.

Performance and interpretation

- It is desirable to measure the performance of the random forests, as of any other classification technique, in terms of the error probability:

$$R(T^{RF}) = \mathbb{P}\left(T^{RF}(X) \neq Y\right).$$

- As usual, the error can be measured:
  - For a **probability distribution**: by a **simulation study**, *i.e.* train T^{RF} and measure its classification error for a number of simulated data sets.
  - For **given data**: by **splitting data** into training and test subsets, by **iterating this splitting** if data are small, or by **cross-validation**.
- Random forests offer an additional possibility to directly estimate classification error exploiting the **out-of-bag (OOB)** principle.
- The same idea can be extended from sample points to variables allowing to measure **variable importance**.

Out-of-bag error

- For each pair $(\mathbf{x}_i, y_i)$ from $\mathcal{D}_n$, let I_i be the **set of indices of trees** whose bootstrap samples $\mathcal{D}_n^*$ **do not contain this observation**.

Out-of-bag error

- For each pair $(\mathbf{x}_i, y_i)$ from $\mathcal{D}_n$, let I_i be the **set of indices of trees** whose bootstrap samples $\mathcal{D}_n^*$ **do not contain this observation**.

- By these trees, observation $\mathbf{x}_i$ is then classified as

$$\hat{y}_i = \frac{1}{\#I_i} \sum_{k \in I_i} T_k^*(\mathbf{x}_i)\,.$$

Out-of-bag error

- For each pair $(\mathbf{x}_i, y_i)$ from $\mathcal{D}_n$, let I_i be the **set of indices of trees** whose bootstrap samples $\mathcal{D}_n^*$ **do not contain this observation**.

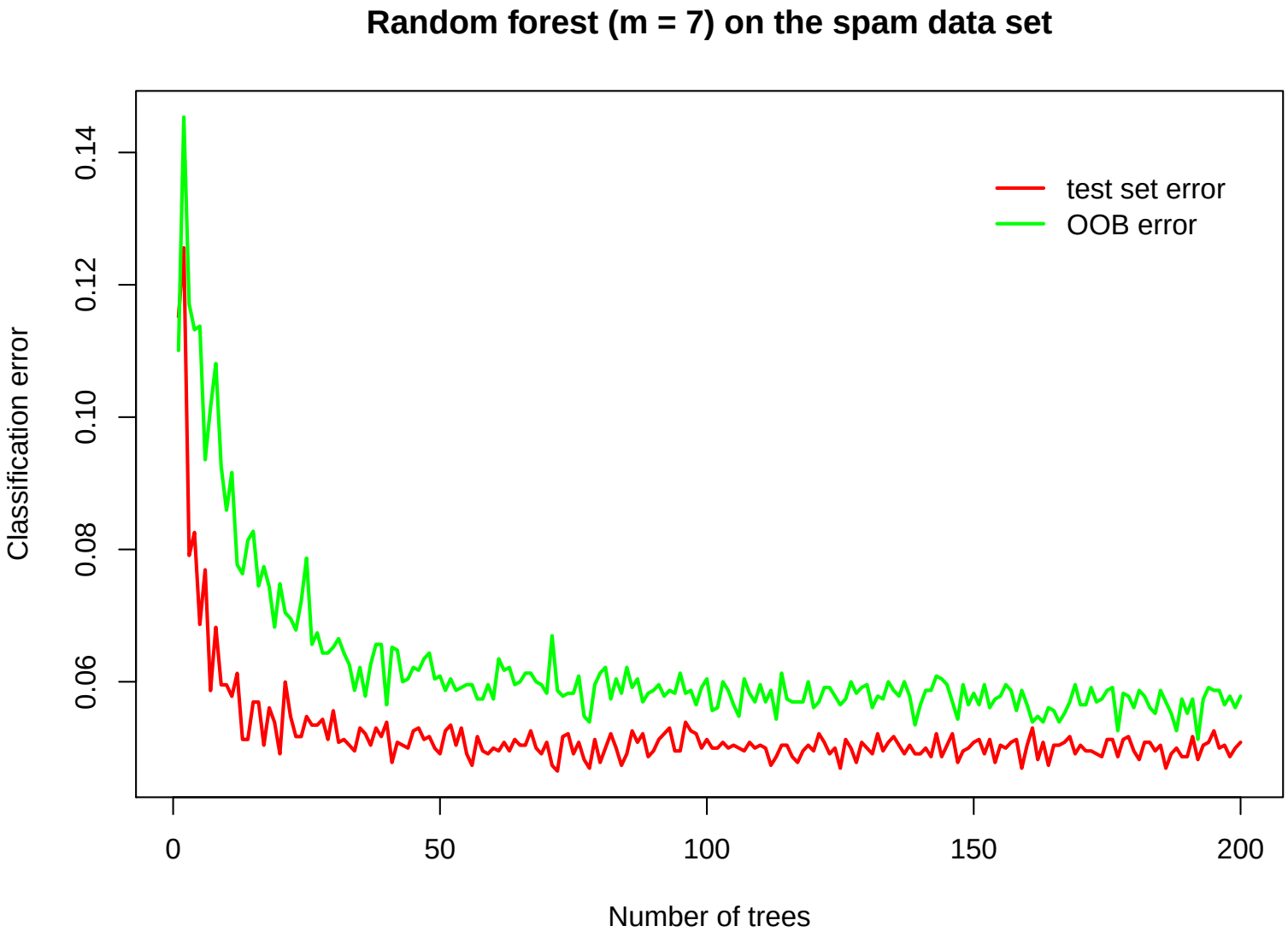
- By these trees, observation $\mathbf{x}_i$ is then classified as

$$\hat{y}_i = \frac{1}{\#I_i} \sum_{k \in I_i} T_k^*(\mathbf{x}_i) \,.$$

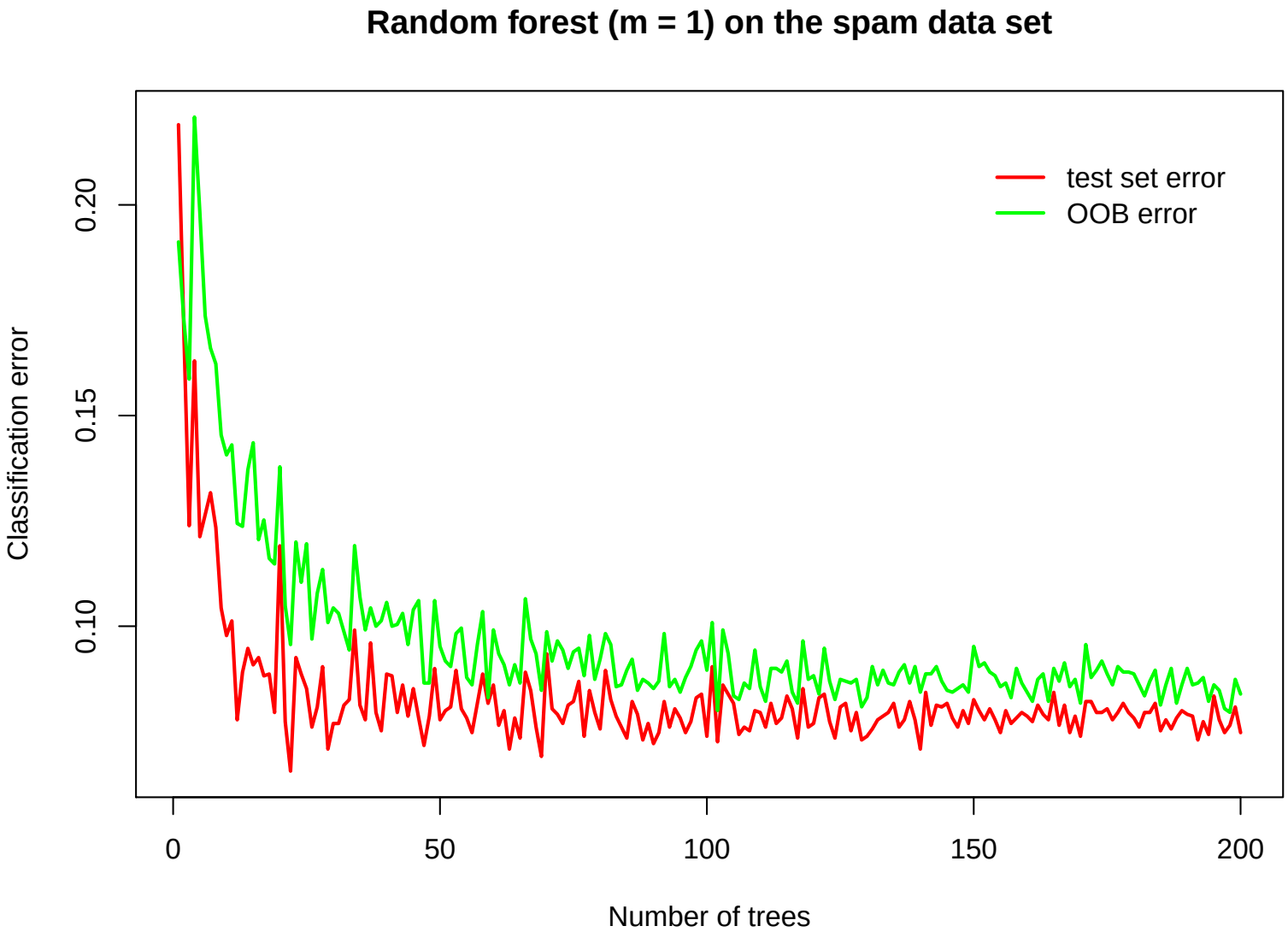
- Averaging over all observations $\mathbf{x}_i$, $i = 1, ..., n$ from $\mathcal{D}_n$ gives the out-of-bag estimate of the error rate:

$$R_{OOB} = \frac{1}{n} \sum_{i=1}^{n} 1\big(\hat{y}_i \neq y_i\big) \,.$$

Random forests: spam data



Random forests: spam data



Importance of a variable

- For a bootstrap sample $\mathcal{D}^*_{n,k}$, let $\mathcal{D}^{*-}_{n,k}$ be the **subset of training sample not contained in** $\mathcal{D}^*_{n,k}$, *i.e.* it holds $\mathcal{D}^*_{n,k} \cup \mathcal{D}^{*-}_{n,k} = \mathcal{D}_n$ and $\mathcal{D}^*_{n,k} \cap \mathcal{D}^{*-}_{n,k} = \emptyset$.

Importance of a variable

- For a bootstrap sample $\mathcal{D}^*_{n,k}$, let $\mathcal{D}^{*-}_{n,k}$ be the **subset of training sample not contained in** $\mathcal{D}^*_{n,k}$, *i.e.* it holds $\mathcal{D}^*_{n,k} \cup \mathcal{D}^{*-}_{n,k} = \mathcal{D}_n$ and $\mathcal{D}^*_{n,k} \cap \mathcal{D}^{*-}_{n,k} = \emptyset$.
- Then, let $R_{OOB(k)}$ be the classification error estimated on $\mathcal{D}^{*-}_{n,k}$:

$$R_{OOB(k)} = \frac{1}{\#\mathcal{D}^{*-}_{n,k}} \sum_{\mathbf{x} \in \mathcal{D}^{*-}_{n,k}} 1\left(T^*_k(\mathbf{x}) \neq y_i\right).$$

Importance of a variable

- For a bootstrap sample $\mathcal{D}^*_{n,k}$, let $\mathcal{D}^{*-}_{n,k}$ be the **subset of training sample not contained in** $\mathcal{D}^*_{n,k}$, *i.e.* it holds $\mathcal{D}^*_{n,k} \cup \mathcal{D}^{*-}_{n,k} = \mathcal{D}_n$ and $\mathcal{D}^*_{n,k} \cap \mathcal{D}^{*-}_{n,k} = \emptyset$.
- Then, let $R_{OOB(k)}$ be the classification error estimated on $\mathcal{D}^{*-}_{n,k}$:

$$R_{OOB(k)} = \frac{1}{\#\mathcal{D}^{*-}_{n,k}} \sum_{\mathbf{x} \in \mathcal{D}^{*-}_{n,k}} 1\left(T^*_k(\mathbf{x}) \neq y_i\right).$$

- Further, let $\mathcal{D}^{*-}_{n,k}(j)$ be the same subset $\mathcal{D}^{*-}_{n,k}$ where the **values of variable** $j \in \{1, ..., d\}$ have been **randomly perturbed**, and measure the error from above on this perturbed subset:

$$R_{OOB(k,j)} = \frac{1}{\#\mathcal{D}^{*-}_{n,k}(j)} \sum_{\mathbf{x} \in \mathcal{D}^{*-}_{n,k}(j)} 1\left(T^*_k(\mathbf{x}) \neq y_i\right).$$

Importance of a variable

- For a bootstrap sample $\mathcal{D}^*_{n,k}$, let $\mathcal{D}^{*-}_{n,k}$ be the **subset of training sample not contained in** $\mathcal{D}^*_{n,k}$, *i.e.* it holds $\mathcal{D}^*_{n,k} \cup \mathcal{D}^{*-}_{n,k} = \mathcal{D}_n$ and $\mathcal{D}^*_{n,k} \cap \mathcal{D}^{*-}_{n,k} = \emptyset$.
- Then, let $R_{OOB(k)}$ be the classification error estimated on $\mathcal{D}^{*-}_{n,k}$:

$$R_{OOB(k)} = \frac{1}{\#\mathcal{D}^{*-}_{n,k}} \sum_{\mathbf{x} \in \mathcal{D}^{*-}_{n,k}} 1\left(T^*_k(\mathbf{x}) \neq y_i\right).$$

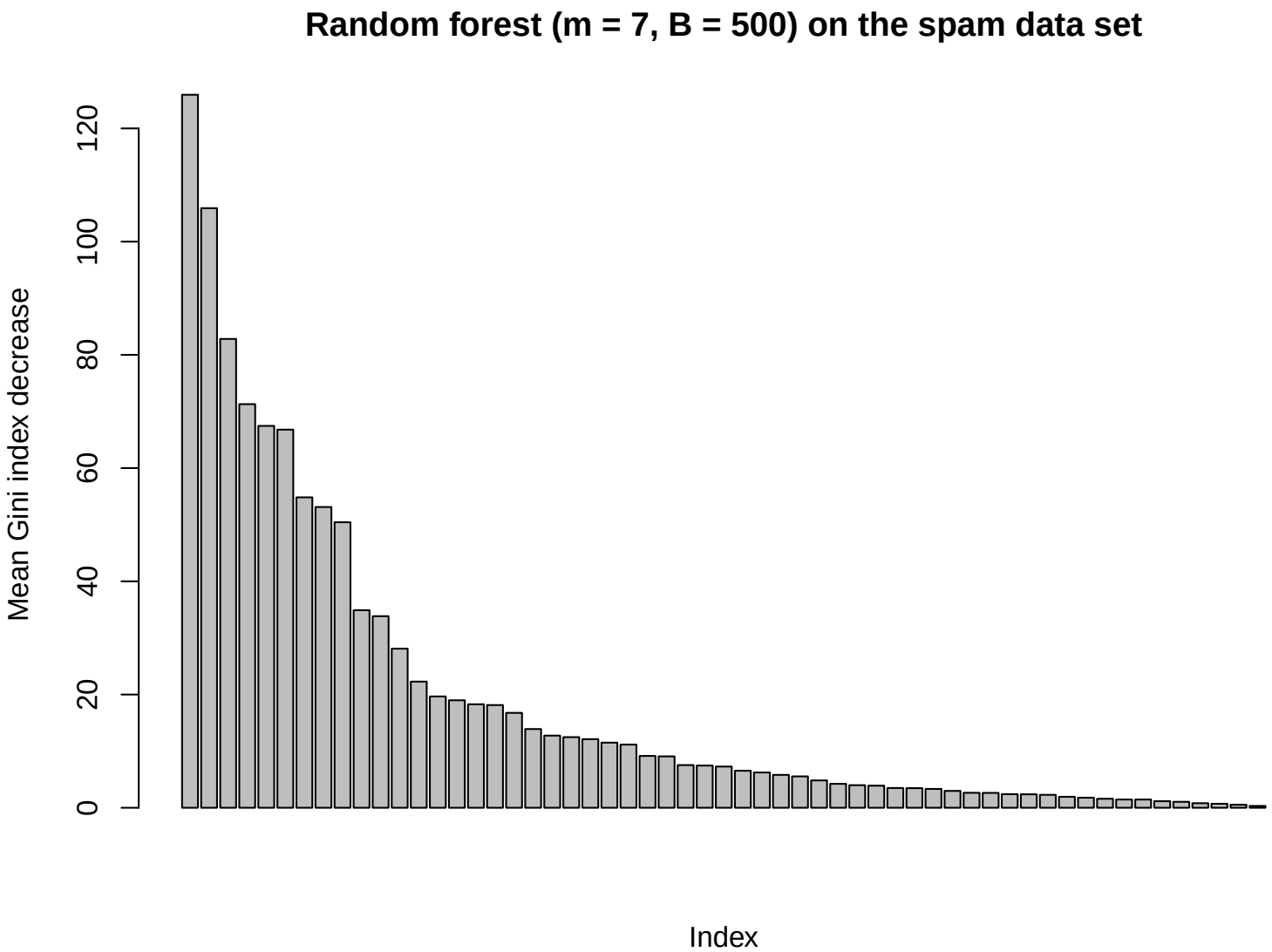
- Further, let $\mathcal{D}^{*-}_{n,k}(j)$ be the same subset $\mathcal{D}^{*-}_{n,k}$ where the **values of variable** $j \in \{1, ..., d\}$ have been **randomly perturbed**, and measure the error from above on this perturbed subset:

$$R_{OOB(k,j)} = \frac{1}{\#\mathcal{D}^{*-}_{n,k}(j)} \sum_{\mathbf{x} \in \mathcal{D}^{*-}_{n,k}(j)} 1\left(T^*_k(\mathbf{x}) \neq y_i\right).$$

- The **importance of variable** j can thus be measured (by averaging over all B trees) as:

$$Imp(X_j) = \frac{1}{B} \sum_{k=1}^{B} \left(R_{OOB(k,j)} - R_{OOB(k)}\right).$$

Random forests: importance of the variables using the Gini index decrease



Contents

Consistency of the purely random forest classifier

Define the **purely random tree classifier** T^{pr} as follows:

Consistency of the purely random forest classifier

Define the **purely random tree classifier** T^{pr} as follows:

- The support of X (and thus the root node of T^{pr}) is $[0,1]^d$.

Consistency of the purely random forest classifier

Define the **purely random tree classifier** T^{pr} as follows:

- The support of X (and thus the root node of T^{pr}) is $[0,1]^d$.
- At each step, the leaf is chosen uniformly at random among all existing leaves.

Consistency of the purely random forest classifier

Define the **purely random tree classifier** T^{pr} as follows:

- The support of X (and thus the root node of T^{pr}) is $[0,1]^d$.
- At each step, the leaf is chosen uniformly at random among all existing leaves.
- At each node, the split variable j is chosen uniformly at random among $1, ..., d$.

Consistency of the purely random forest classifier

Define the **purely random tree classifier** T^{pr} as follows:

- The support of X (and thus the root node of T^{pr}) is $[0,1]^d$.
- At each step, the leaf is chosen uniformly at random among all existing leaves.
- At each node, the split variable j is chosen uniformly at random among $1, ..., d$.
- The selected cell is split at a random location, chosen according to a uniform random variable on the length of the chosen side of the selected cell.

Consistency of the purely random forest classifier

Define the **purely random tree classifier** T^{pr} as follows:

- The support of X (and thus the root node of T^{pr}) is $[0,1]^d$.
- At each step, the leaf is chosen uniformly at random among all existing leaves.
- At each node, the split variable j is chosen uniformly at random among $1, ..., d$.
- The selected cell is split at a random location, chosen according to a uniform random variable on the length of the chosen side of the selected cell.
- The procedure is repeated k times where $k \geq 1$ is fixed in advance.

Consistency of the purely random forest classifier

Define the **purely random tree classifier** T^{pr} as follows:

- The support of X (and thus the root node of T^{pr}) is $[0,1]^d$.
- At each step, the leaf is chosen uniformly at random among all existing leaves.
- At each node, the split variable j is chosen uniformly at random among $1, ..., d$.
- The selected cell is split at a random location, chosen according to a uniform random variable on the length of the chosen side of the selected cell.
- The procedure is repeated k times where $k \geq 1$ is fixed in advance.
- The only data driven element is the class label of the leaf, chosen due to the majority of the observations contained in it.

Consistency of the purely random forest classifier

Define the **purely random tree classifier** T^{pr} as follows:

- The support of X (and thus the root node of T^{pr}) is $[0,1]^d$.
- At each step, the leaf is chosen uniformly at random among all existing leaves.
- At each node, the split variable j is chosen uniformly at random among $1, ..., d$.
- The selected cell is split at a random location, chosen according to a uniform random variable on the length of the chosen side of the selected cell.
- The procedure is repeated k times where $k \geq 1$ is fixed in advance.
- The only data driven element is the class label of the leaf, chosen due to the majority of the observations contained in it.

Theorem (Biau, Devroye, Lugosi, 2008; Th. 1)

Assume that the distribution of X is supported on $[0,1]^d$. Then the purely random forest classifier $T_B^{prRF} = 1\big(\frac{1}{B}\sum_{k=1}^{B} T^{pr}(\cdot, \mathcal{D}_n)\big)$ (as well as $\lim_{B\to\infty} T_B^{prRF}$) is consistent whenever $k \to \infty$ and $k/n \to 0$ as $k \to \infty$.

Consistency of the scale-invariant random forest classifier

Define the **scale-invariant random tree classifier** T^{si} as follows:

Consistency of the scale-invariant random forest classifier

Define the **scale-invariant random tree classifier** T^{si} as follows:

- Take the **purely random tree classifier**.

Consistency of the scale-invariant random forest classifier

Define the **scale-invariant random tree classifier** T^{si} as follows:

- Take the **purely random tree classifier**.
- Let the root node be the entire space $\mathbb{R}^d$.

Consistency of the scale-invariant random forest classifier

Define the **scale-invariant random tree classifier** T^{si} as follows:

- Take the **purely random tree classifier**.
- Let the root node be the entire space $\mathbb{R}^d$.
- Define the node-cutting procedure as follows:
  if the cell (node) m contains n_m points $\mathbf{x}_1, ..., \mathbf{x}_{n_m}$, then the random index I is chosen uniformly from the set $\{0, 1, ..., n_m\}$, and the cut is performed in the chosen variable between the points $\mathbf{x}_I$ and $\mathbf{x}_{I+1}$.

Consistency of the scale-invariant random forest classifier

Define the **scale-invariant random tree classifier** T^{si} as follows:

- Take the **purely random tree classifier**.
- Let the root node be the entire space $\mathbb{R}^d$.
- Define the node-cutting procedure as follows:
  if the cell (node) m contains n_m points $\mathbf{x}_1, ..., \mathbf{x}_{n_m}$, then the random index I is chosen uniformly from the set $\{0, 1, ..., n_m\}$, and the cut is performed in the chosen variable between the points $\mathbf{x}_I$ and $\mathbf{x}_{I+1}$.

Theorem (Biau, Devroye, Lugosi, 2008)

Assume that the distribution of X has non-atomic marginals in $\mathbb{R}^d$. Then the scale-invariant random forest classifier $T_B^{siRF} = 1\big(\frac{1}{B}\sum_{k=1}^{B} T^{si}(\cdot, \mathcal{D}_n)\big)$ (as well as $\lim_{B\to\infty} T_B^{siRF}$) is consistent whenever $k \to \infty$ and $\frac{k}{n} \to 0$ as $k \to \infty$.

Consistency of bagging

Remind:

- Bagging classifier:

$$g_B^{agg}(\mathbf{x}) = 1\Big(\frac{1}{B}\sum_{k=1}^{B} g_k^*(\mathbf{x}) > 0.5\Big)\,.$$

Consistency of bagging

Remind:

- Bagging classifier:

$$g_B^{agg}(\mathbf{x}) = 1\Big(\frac{1}{B}\sum_{k=1}^{B} g_k^*(\mathbf{x}) > 0.5\Big)\,.$$

- Averaged classifier (the limit of the bagging classifier):

$$\lim_{B\to\infty} g_B^{agg}(\mathbf{x}) = 1\big(\mathbb{E}^*[g_k^*(\mathbf{x}) \mid \mathcal{D}_n) > 0.5]\big)\,.$$

with θ being a random variable delivering a bootstrap sample of size $Bin(n, q_n)$ (without replacement), and $q_n \in [0, 1]$.

Consistency of bagging

Remind:

- Bagging classifier:

$$g_B^{agg}(\mathbf{x}) = 1\Big(\frac{1}{B}\sum_{k=1}^{B} g_k^*(\mathbf{x}) > 0.5\Big)\,.$$

- Averaged classifier (the limit of the bagging classifier):

$$\lim_{B\to\infty} g_B^{agg}(\mathbf{x}) = 1\big(\mathbb{E}^*[g_k^*(\mathbf{x}) \mid \mathcal{D}_n) > 0.5]\big)\,.$$

with θ being a random variable delivering a bootstrap sample of size $Bin(n, q_n)$ (without replacement), and $q_n \in [0, 1]$.

Theorem (Biau, Devroye, Lugosi, 2008; Th. 6)

Assume that the classifier g is consistent for a certain distribution (X, Y). Then the bagging classifier g_B^{agg} and its limit $1\big(\mathbb{E}[g_k^*(\mathbf{x}) \mid \mathcal{D}_n)] > 0.5\big)$ are also consistent if $nq_n \to \infty$ as $n \to \infty$.

Thank you for your attention!

And some more references

- Biau, G., Devroye, L., and Lugosi, G. (2008). Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9, 2015–2033.
- Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24, 123–140.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32.
- Buhlmann, P. (2003). Bagging, subbagging and bragging for improving some prediction algorithms. *Recent Advances and Trends in Nonparametric Statistics.* Akritas, M.G. and Politis, D.N. (Eds.). Elsevier, pp. 9–34.
- Györfi, L., Kohler, M., Krzyżak, A., Walk, H. (2002) *A distribution-free theory of nonparametric regression* Springer.
- Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistics Learning: Data Mining, Inference, and Prediction (Second Edition)*. Springer.
- Stone, C.J. (1977). Consistent nonparametric regression. *The Annals of Statistics*, 55(4), 595–645.
- Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*. CRC Press.