Support vector machines and tree-based methods

Federica Eduati

Eindhoven University of Technology
Department of Biomedical Engineering

2024

Learning goals

At the end of this lecture you will be able to:

- Explain the formulation of support vector machines (SVM) for classification problems
- Explain The formulation of tree-based methods for classification and regression problems
- Illustrate the application of SVM and tree-based methods to case studies and interpret the results.

Material

- Chapters 9, 12 and 15 from Friedman et al., The Elements of Statistical Learning

Overview

- Support vector machines
    - Maximal margin classifiers
    - Support vector classifiers
    - Support vector machines
- Tree-based methods
    - Decision trees
    - Bagging
    - Random-forests
    - Boosting

What is a hyperplane

Classification problem:

Find a *hyperplane* that separates the two classes in the feature space.

In p dimensions a *hyperplane* is a flat affine subspace of dimension $p-1$ with general equation:

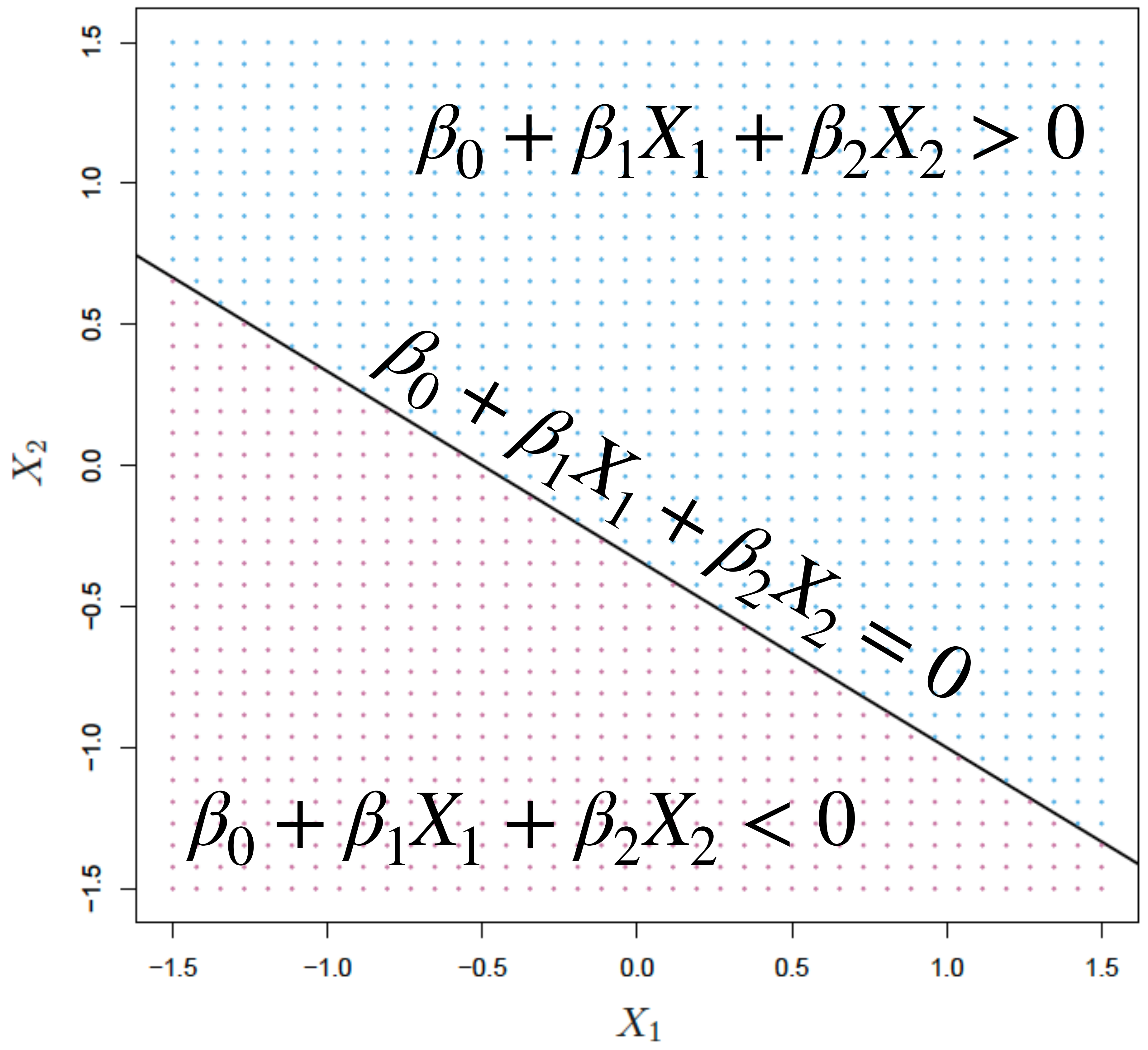
$$f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = x^T \beta + \beta_0 = 0$$

Where:

- $\beta_0 = 0$ only if the hyperplane goes through the origin.
- The vector $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ is a unit vector $(\|\beta\| = 1)$ orthogonal to the surface of the hyperplane.

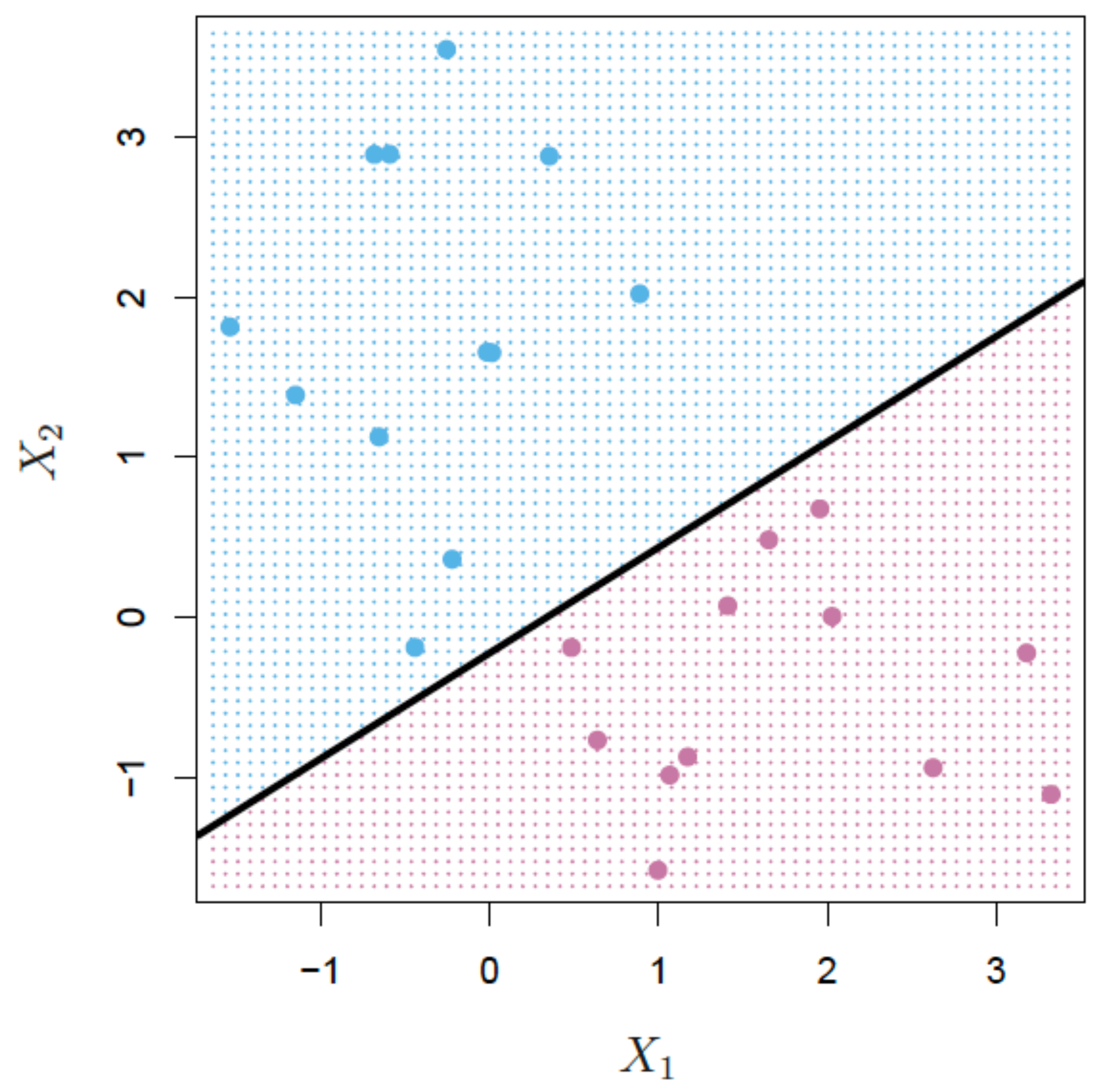
A hyperplane in two dimensions

In two dimensions the hyperplane is a flat one-dimensional subspace, i.e. a line.



Classifying using a separating hyperplane

Suppose to have a training data of N pairs $(x_1, y_1), (x_2, y_3), \ldots, (x_N, y_N)$ where each $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$ and $y_i \in \{-1, 1\}$



We want to find the separating hyperplane

$f(x) = x^T\beta + \beta_0 = 0$

such that:

$x^T\beta + \beta_0 > 0$ for $y_i = 1$ (blue points)

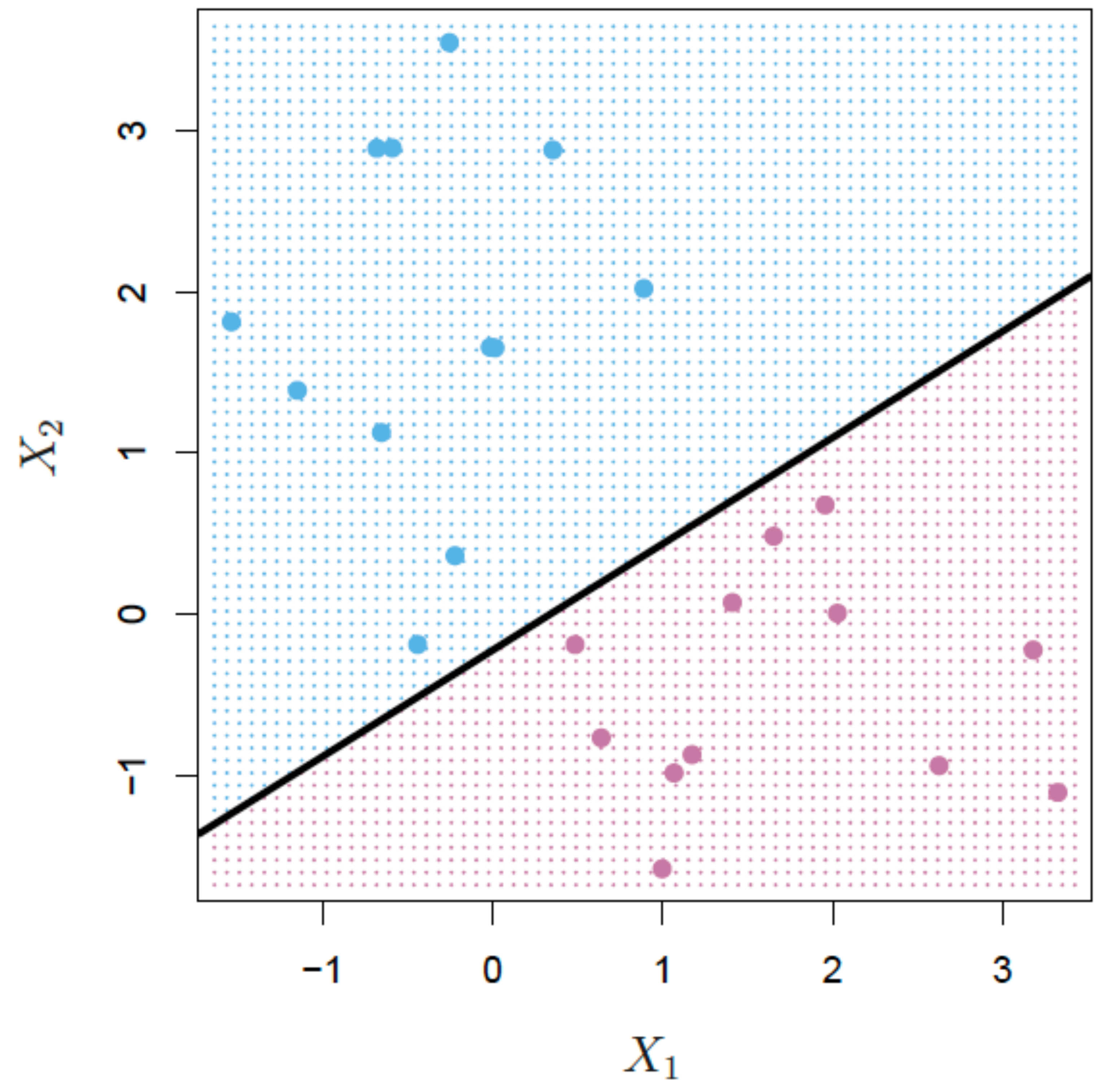
$x^T\beta + \beta_0 < 0$ for $y_i = -1$ (purple points)

Equivalently

$y_i f(x_i) > 0 \quad \forall i$

Classifying using a separating hyperplane

Suppose to have a training data of N pairs $(x_1, y_1), (x_2, y_3), \ldots, (x_N, y_N)$ where each $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$ and $y_i \in \{-1, 1\}$



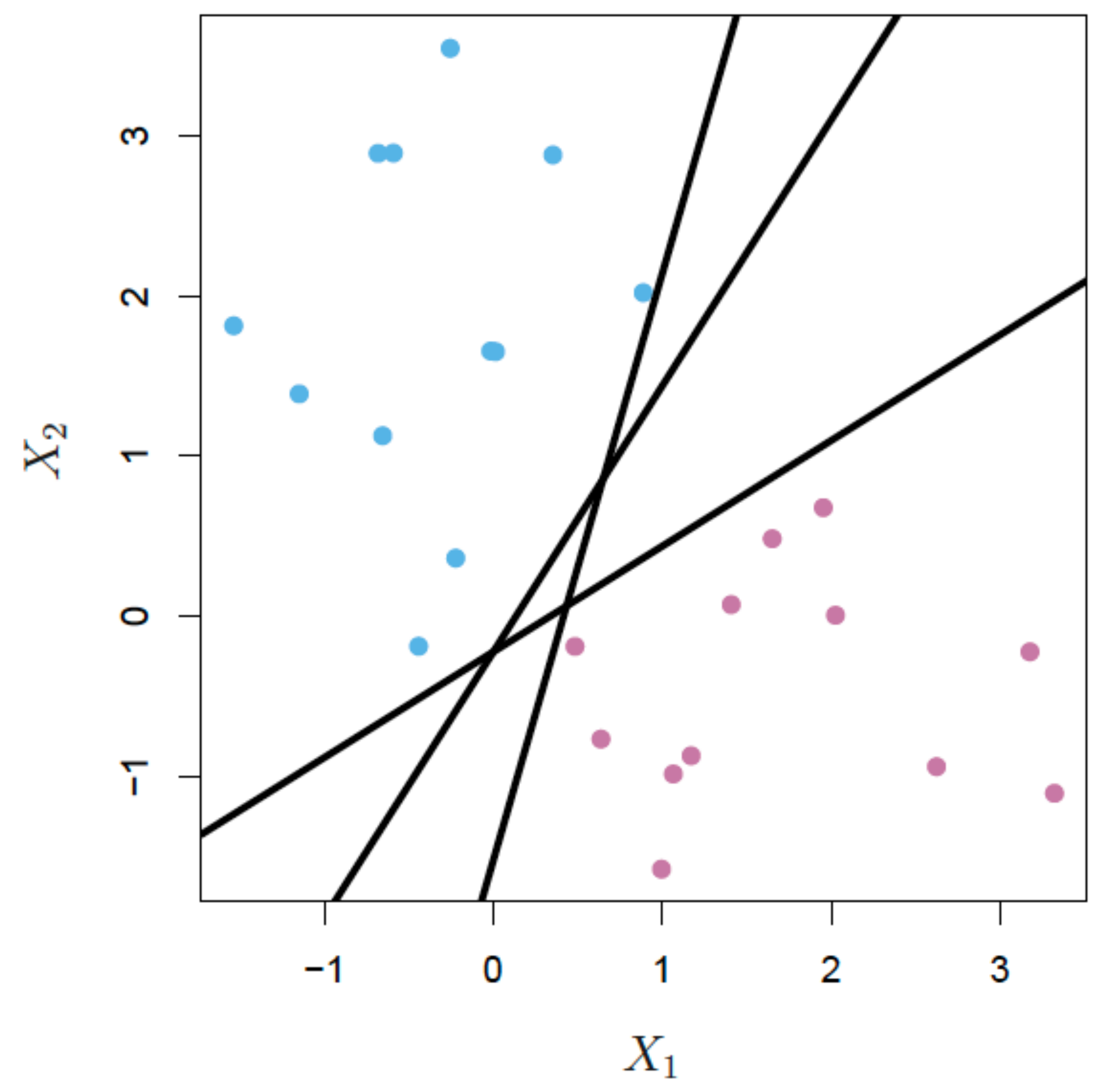
For a new observation $x^* = (x_1^*, x_2^*, \ldots, x_p^*)^T$, the class will be assigned based on the sign:

$f(x^*) > 0$ —> class 1

$f(x^*) < 0$ —> class -1

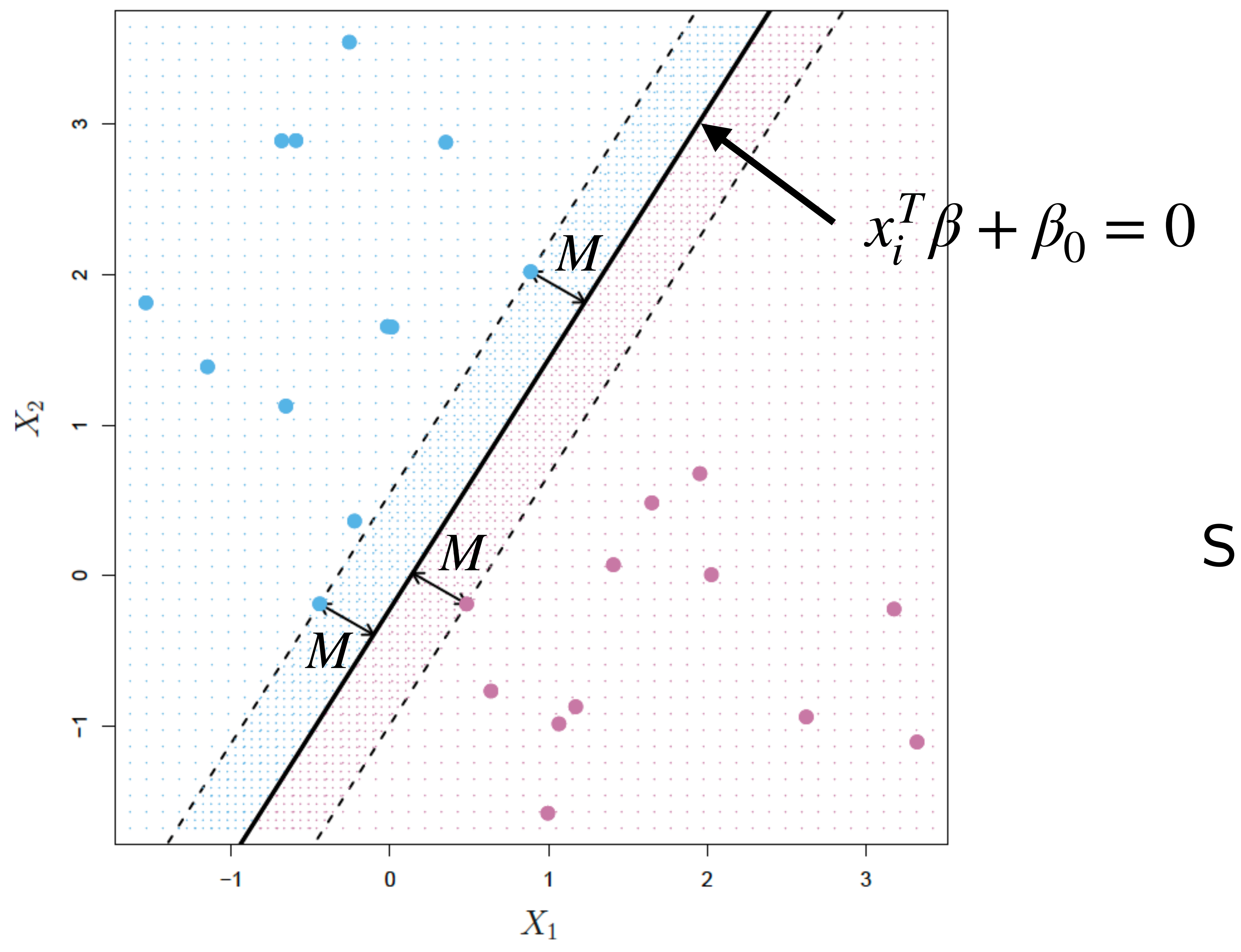
Classifying using a separating hyperplane

If the classes are perfectly separable, there are generally multiple hyperplanes that can separate them.



Maximal margin classifier

The *maximal margin classifier* is the one with the biggest margin between the two classes.



$$\max_{\beta,\beta_0,\|\beta\|=1} \quad M$$

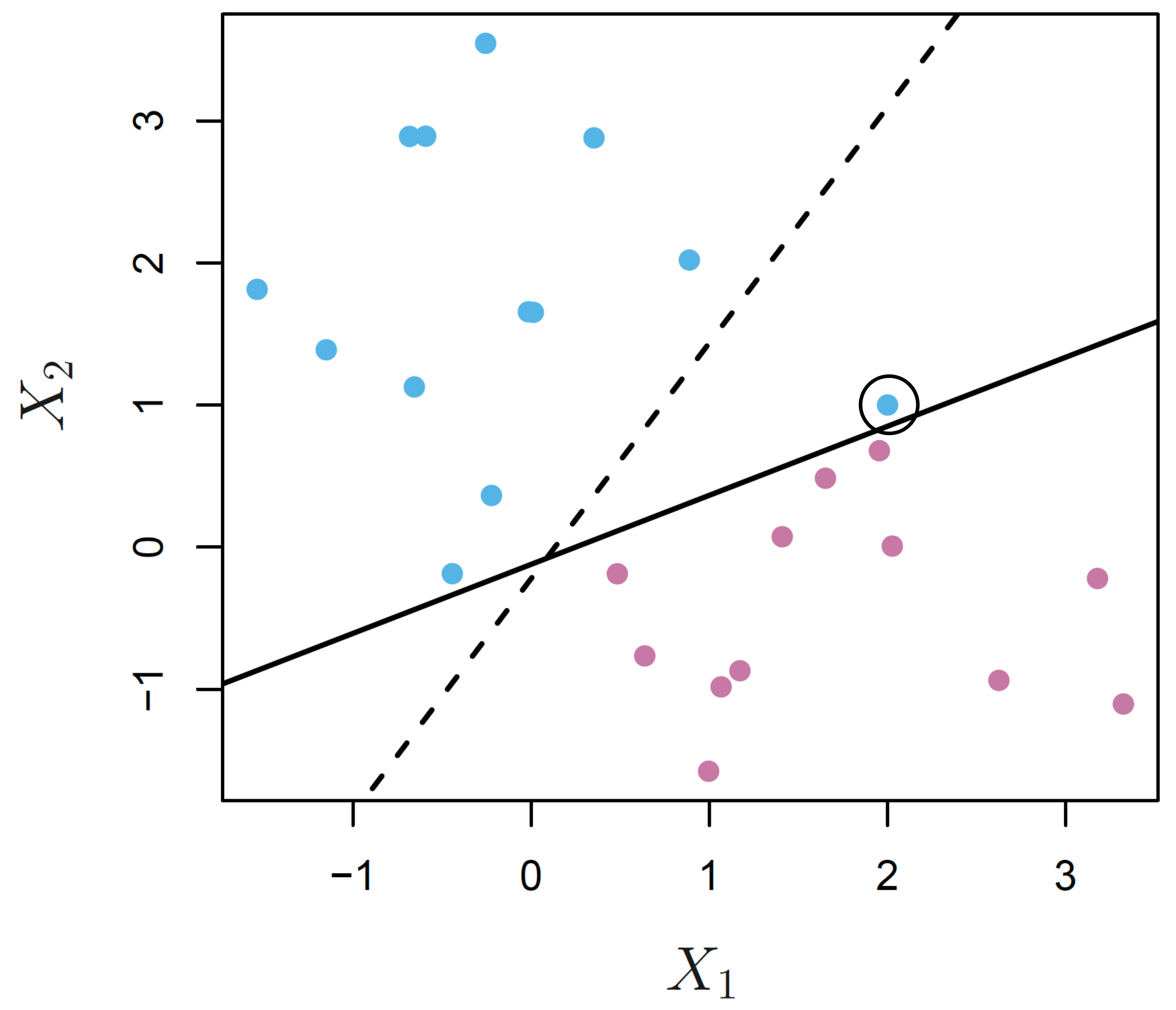
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq M, i = 1,\ldots,N$$

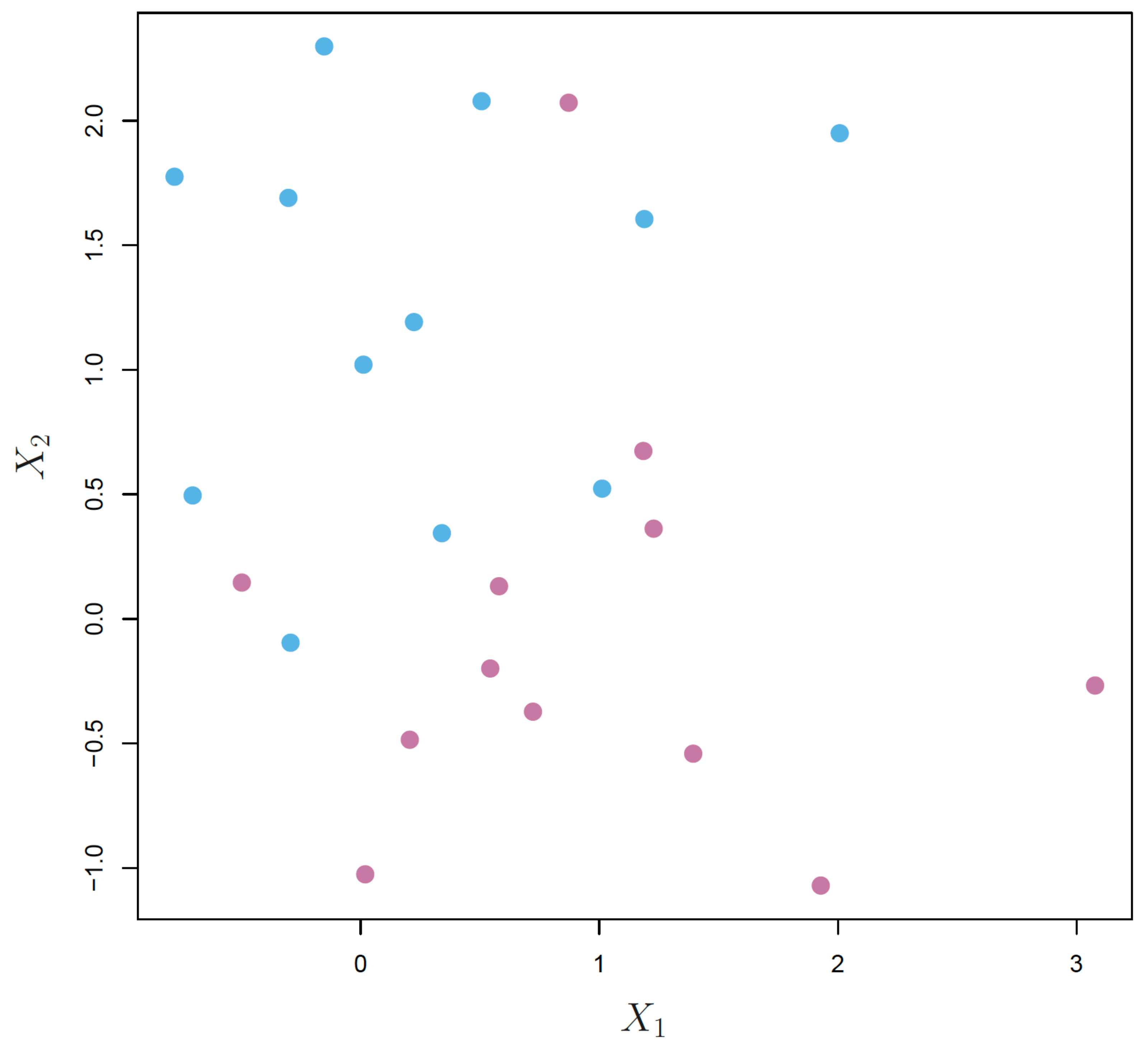
Each observation is on the correct side of the hyperplane and at least a distance M from the hyperplane

Noisy or non-separable data

The maximal margin classifier has issues in case of:

- Noisy data with outliers leading to poor solutions (left panel - just added one data point to the previous example)
- Data non-separable by linear boundary (right panel)





Support vector classifier

The *support vector classifier* provides a solution by maximising a *soft* margin (regularisation).

$$\max_{\beta, \beta_0, \|\beta\|=1} \quad M$$

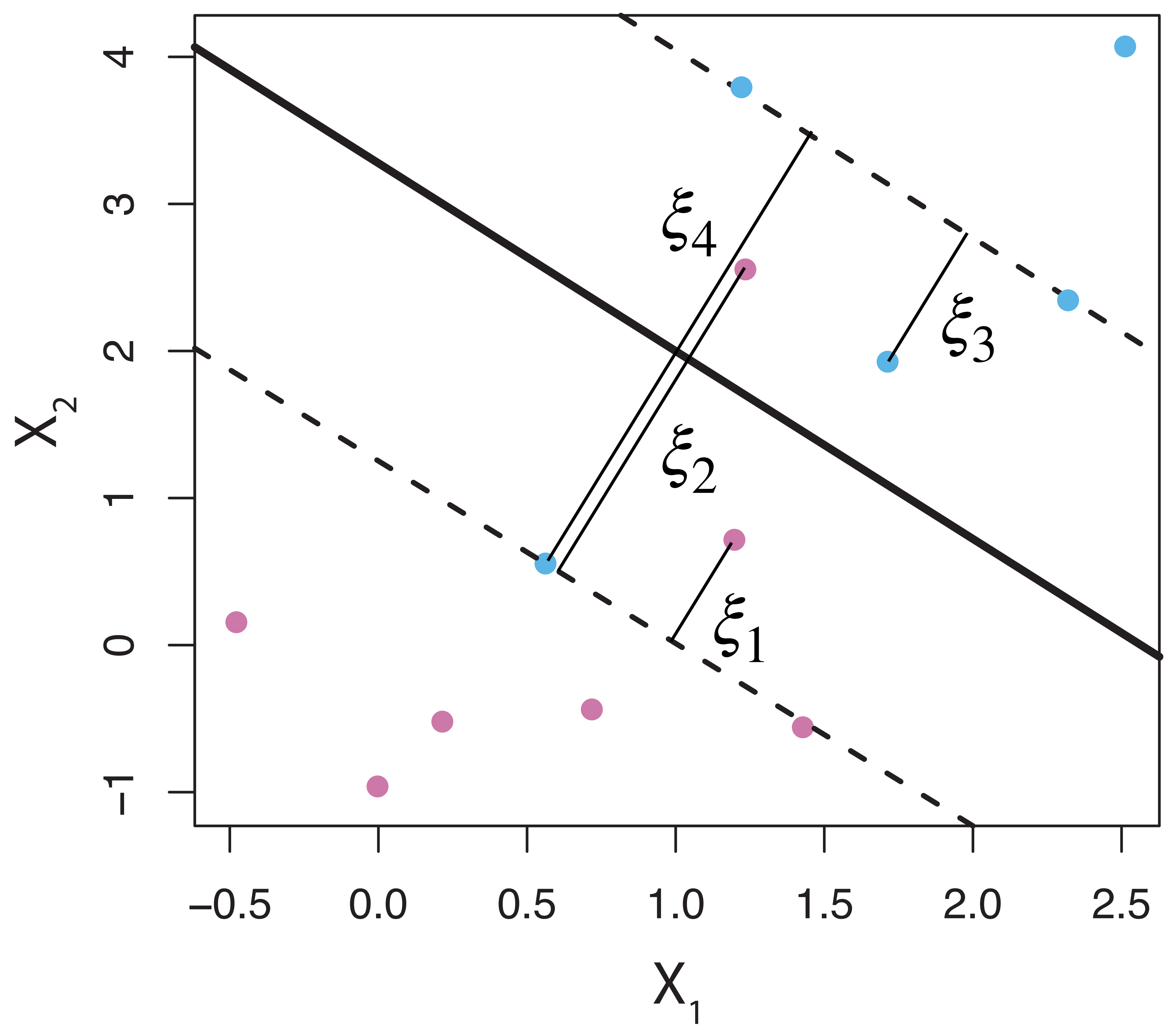
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq M(1 - \xi_i), i = 1, \ldots, N$$

Where:

- $\xi_i \geq 0$ are called *slack variables.*
- $\sum_{i=1}^{N} \xi_i \leq K$, where K is a non-negative constant (tuning parameter) that defines the budget we allow for the total amount of slack.

Slack variables

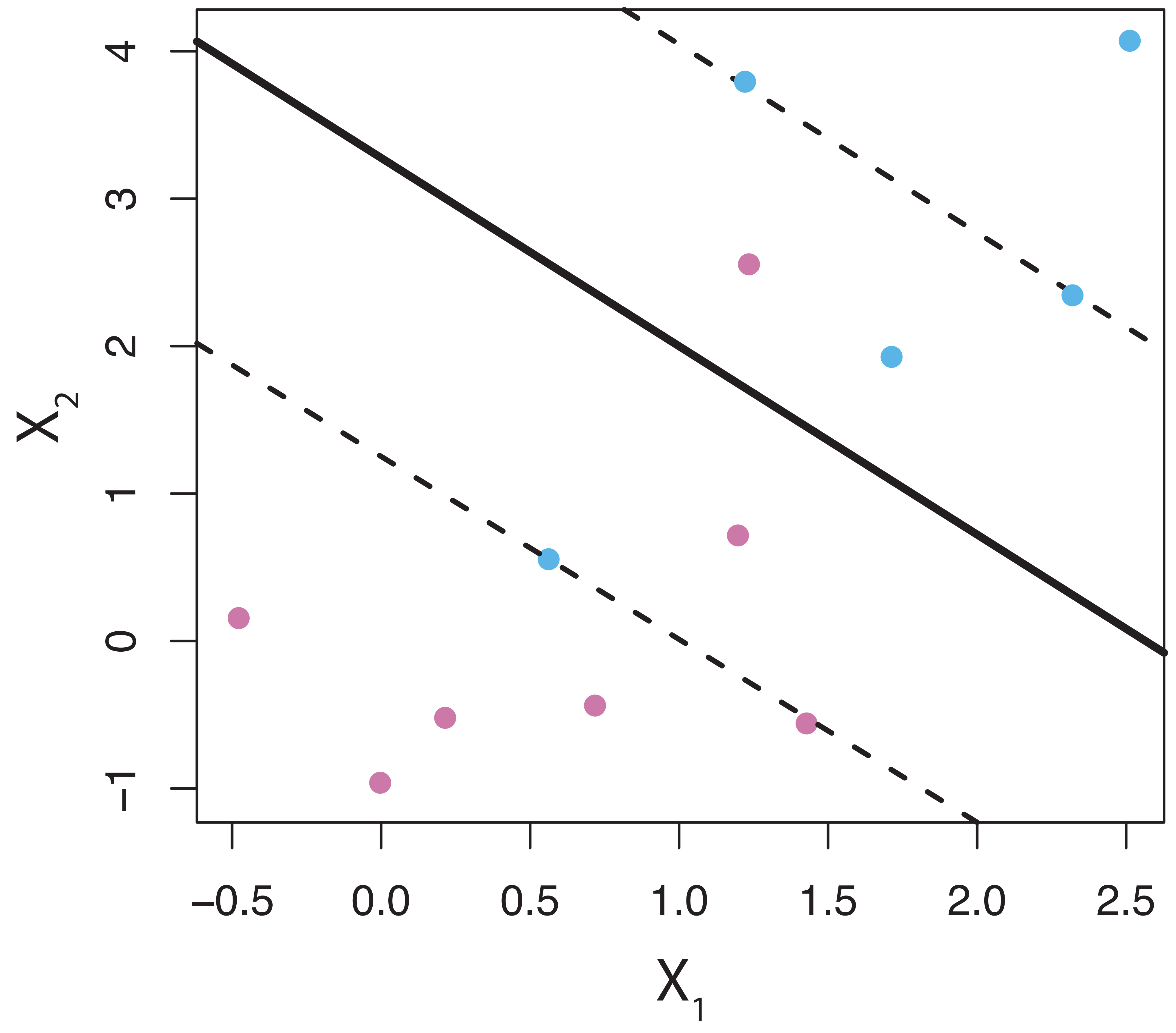
The *slack variables* $\xi_1, \ldots, \xi_N$ allow the corresponding observation i to be:



- $\xi_i = 0$ on the correct side of the margin
- $\xi_i > 0$ on the wrong side of the margin
- $\xi_i > 1$ on the wrong side of the hyperplane

Mentimeter question (www.menti.com code 3680 2274)

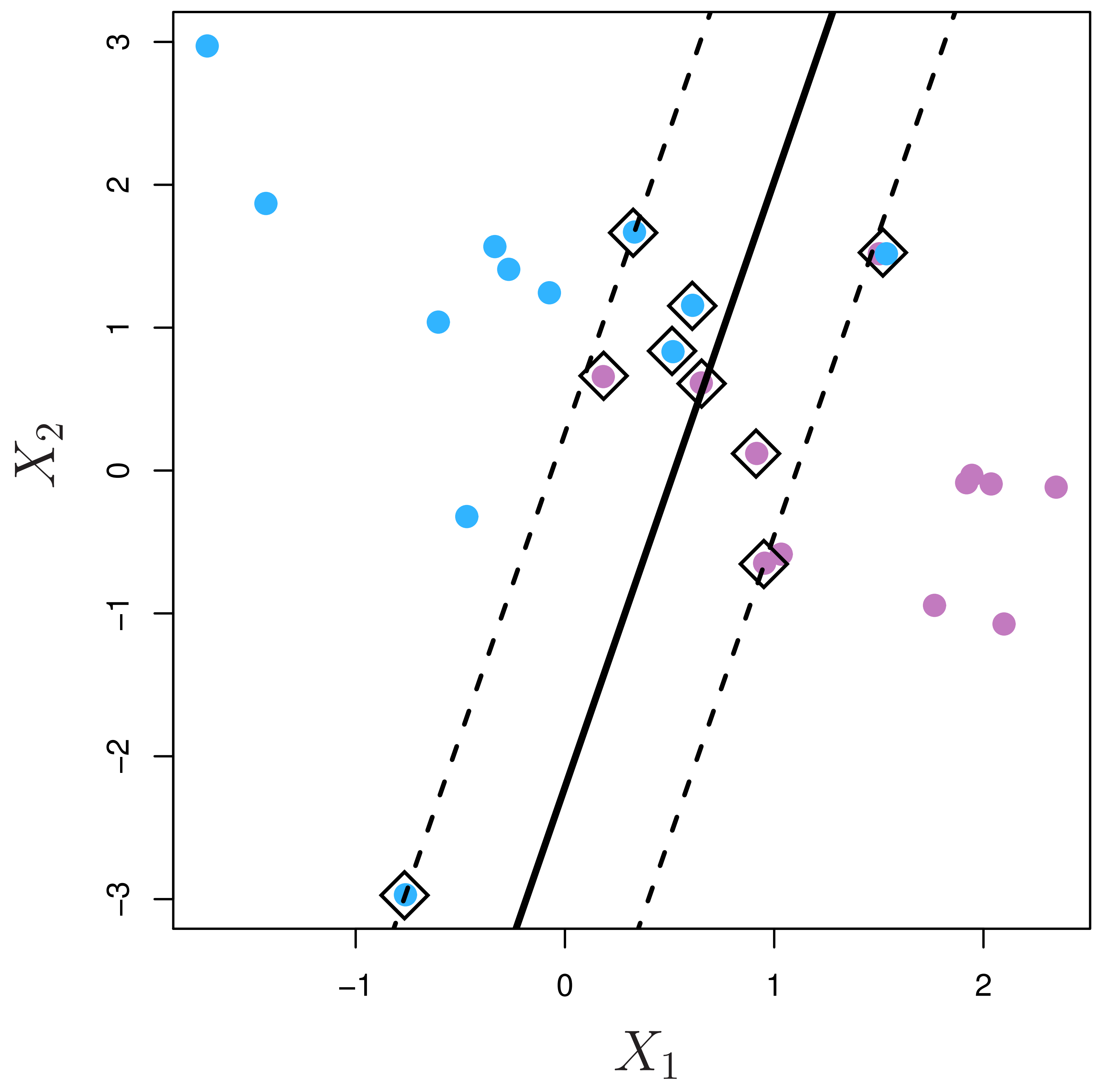
How many purple observations are on the wrong side of the margin?



Support vectors

Support vectors (marked with diamonds) are the observations that lie directly on the margins or on the wrong side of the margin.

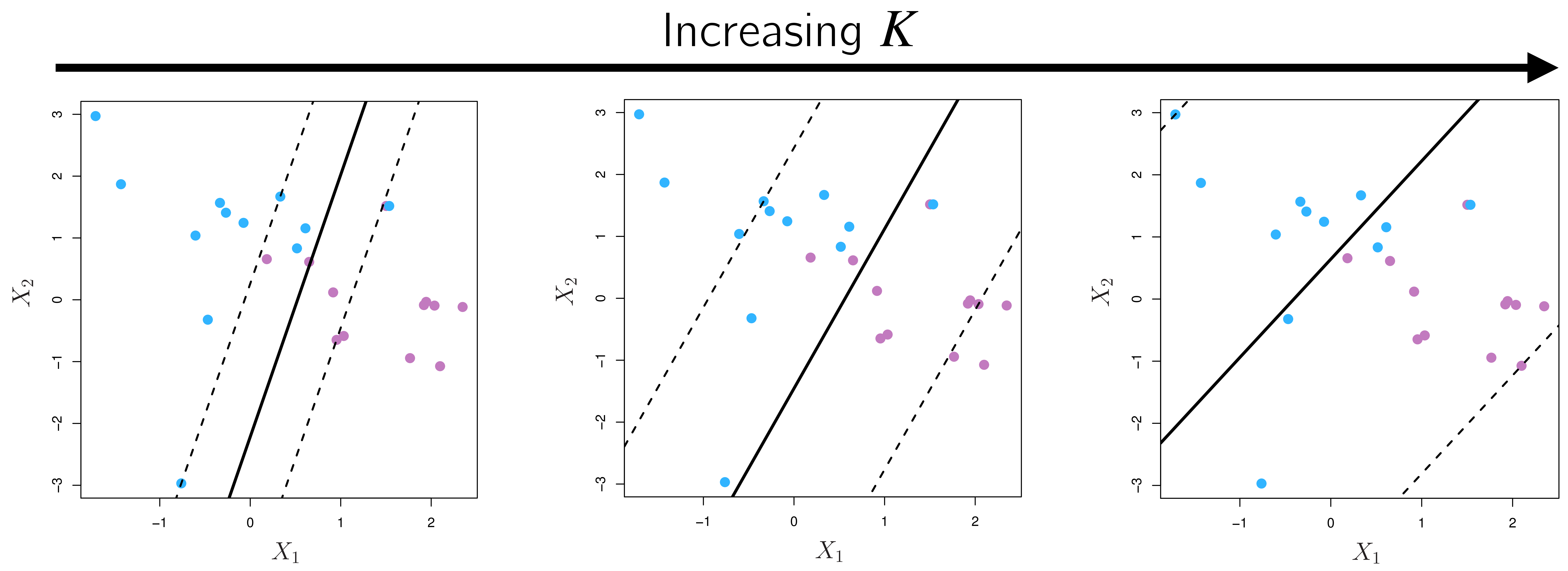
They are the only one that determine the orientation of the hyperplane.



Regularisation

The constant K is tunable and can be seen as a regularisation parameter

- $K = 0$ no budged for violation of the margin (maximum margin classifier)
- Increasing K allows more slack (wider margins, more regularisation)



Mentimeter question (www.menti.com code 3680 2274)

What is the effect of increasing K on the bias-variance trade-off?

- Decreases variance and increases bias
- Increases variance and decreases bias

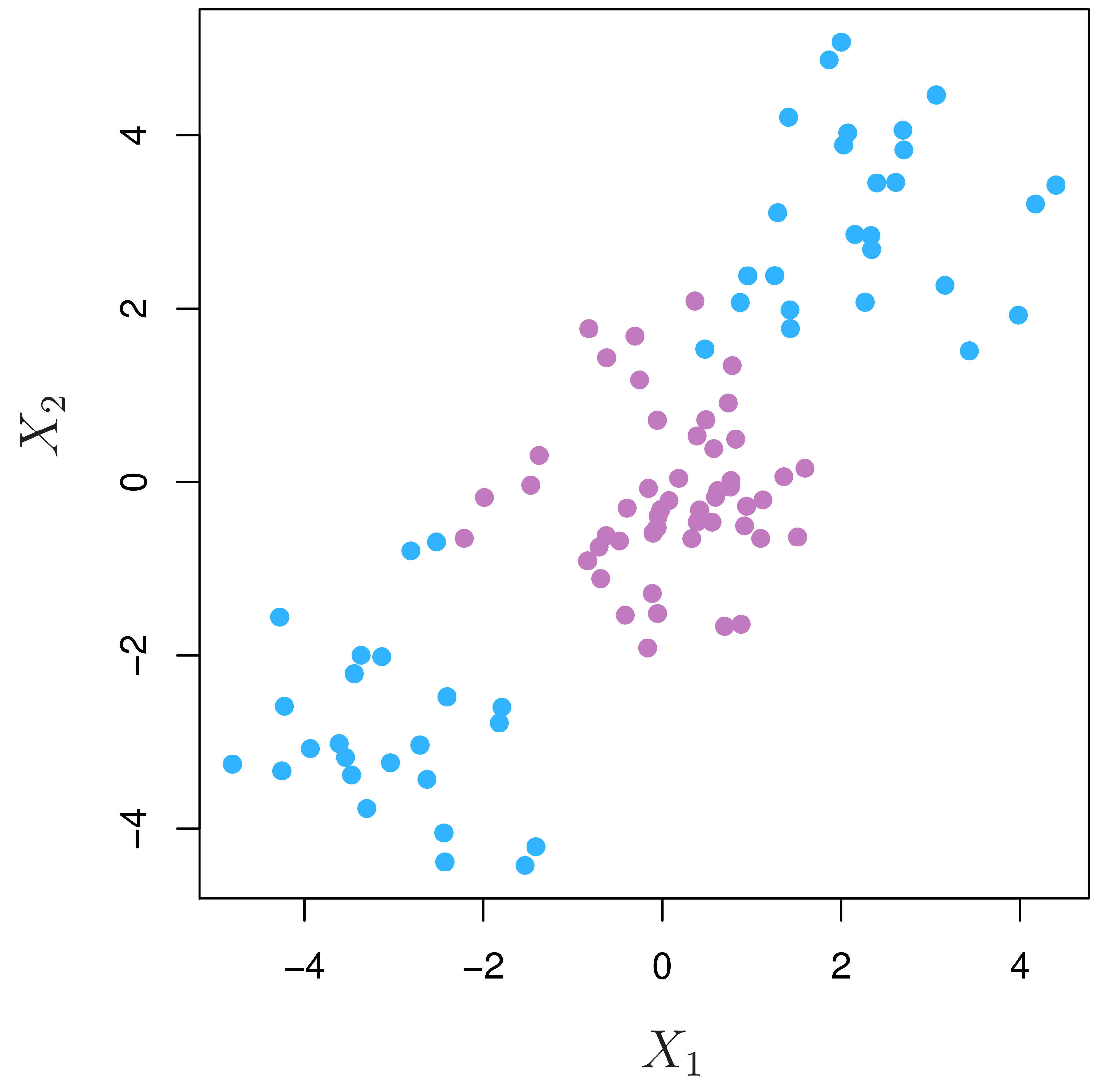
Effect of K on the bias-variance trade-off

K controls the bias-variance trade-off

- Small K —> narrow margins —> high fit to the data —> low bias, high variance
- Large K —> wide margins —> more violation allowed —> high bias, low variance

Mentimeter question (www.menti.com code 3680 2274)

Will a support vector classifier perform well in this example?



- Yes
- No

Classification with non-linear decision boundaries

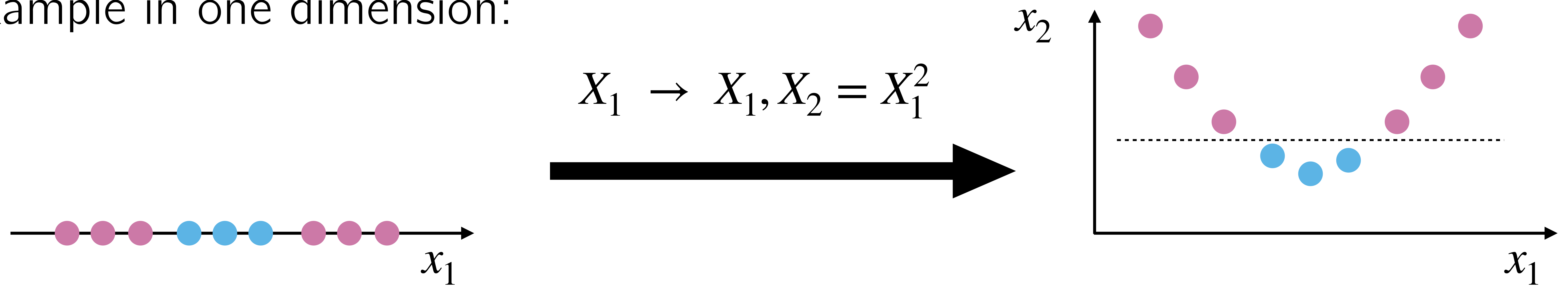
Idea: use of quadratic, cubic or higher order polynomial functions of the predictors to extend support vector classifier to handle non-linear class boundaries.

$$X_1, X_2 \;\rightarrow\; X_1, X_1^2, X_2, X_2^2, X_1X_2$$

$$f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_2 + \beta_4 X_2^2 + \beta_5 X_1 X_2$$

Example in one dimension:

$$X_1 \;\rightarrow\; X_1, X_2 = X_1^2$$



Problem: There are many ways to enlarge feature space, we could end up with a huge number of features (infeasible for high p).

Support vector machines

Support vector machines (SVM): extension of the support vector classifier which enlarges the feature space by using **kernels**.

The kernel approach is an efficient computational methodology to enlarge the feature space to accommodate non-linear boundary between classes.

Solutions using inner products

The solution of the support vector classifier problem involves only the inner products of the observations.

The *inner product* of two r-vectors a, b is $\langle a, b\rangle = \sum_{i=1}^{r} a_i b_i$

The *inner product* of two observations x_i, x_i' is given by:

$$\langle x_i, x_i' \rangle = \sum_{j=1}^{p} x_{ij} x_{ij}'$$

Then, the linear support vector classifier can be represented as:

$$f(x) = \beta_0 + \sum_{i=1}^{N} \alpha_i \langle x, x_i \rangle$$

where there are N parameters α_i, $i = 1, \ldots, N$, one per training observation.

Solutions using inner products

To estimate the parameters $\alpha_1, \ldots, \alpha_n$ and β_0 we need all the inner products $\langle x_i, x_i' \rangle$ between all pairs of training observations.

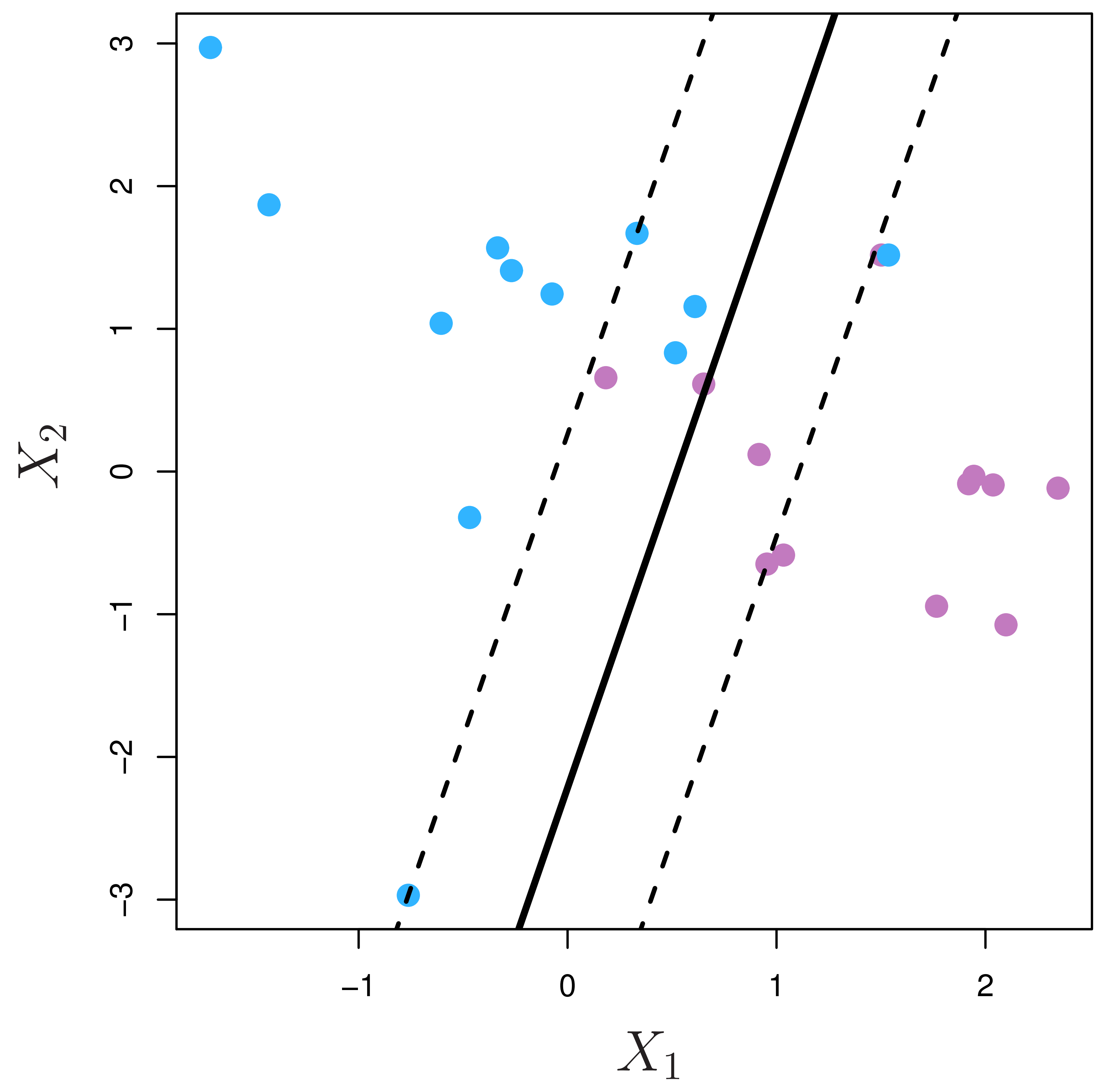
However, α_i is nonzero only for the support vectors, so if S is the collection of indices of these support points, we can rewrite $f(x)$ as:

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$$

Which involves fewer terms than before.

Mentimeter question (www.menti.com code 3680 2274)

How many α do we have to estimate in this example?



The kernel approach

The *kernel* can be seen as an abstraction of the inner product. It is a function that quantifies similarities between observations.

$$K(x_i, x_i')$$

We can then write the function $f(x)$ as:

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i)$$

In the case of the *support vector classifier* we have:

$$K(x_i, x_i') = \sum_{j=1}^{p} x_{ij} x_{ij}'$$

Linear kernel

Support vector machines

Support vector machines are an extension of the support vector classifier that make use of *non-linear kernels*.

$$K(x_i, x_i') = \left(1 + \sum_{j=1}^{p} x_{ij}x_{i'j}\right)^d$$

Polynomial kernel $(d > 1)$

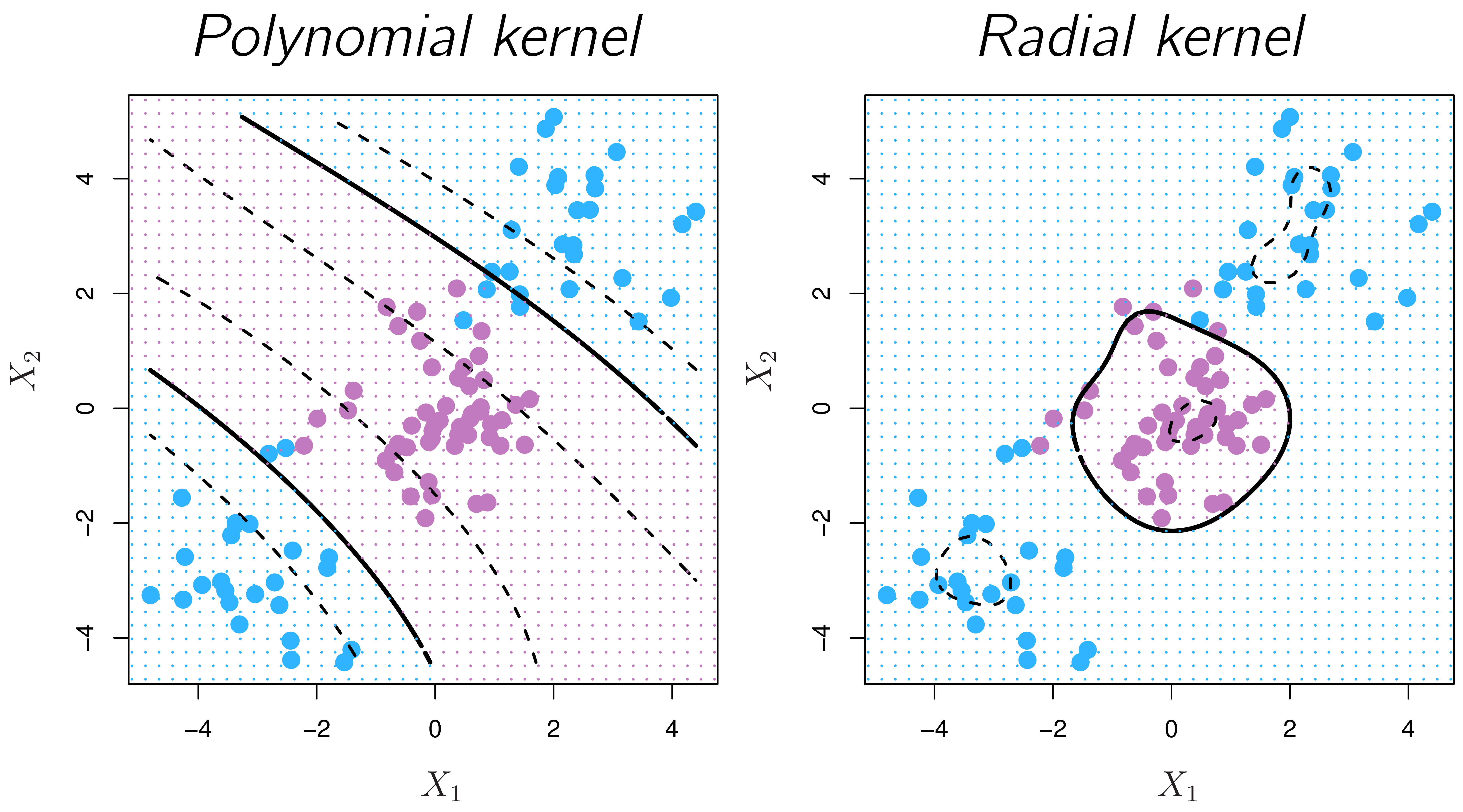
$$K(x_i, x_i') = exp\left(-\gamma \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2\right)$$

Radial kernel $(\gamma > 0)$

d and γ are tuning parameters.

Support vector machines

Example of applications of support vector machines.



Support vector machines

What is the advantage of using a kernel rather than simply enlarging the features space using functions of the original features?

- Computational advantage: we don't work in the enlarged feature space
- Automatically computes the inner product for high dimensional space of features.
- Avoid overfitting by automatically squashing down most dimensions (only based on the support vectors).

Extension to multi-class

So far, binary classification (two-class setting).

The SVM concept of separating hyperplanes does not lend itself to more than two classes.

Two approaches for extending SBM to $K > 2$ classes classification:

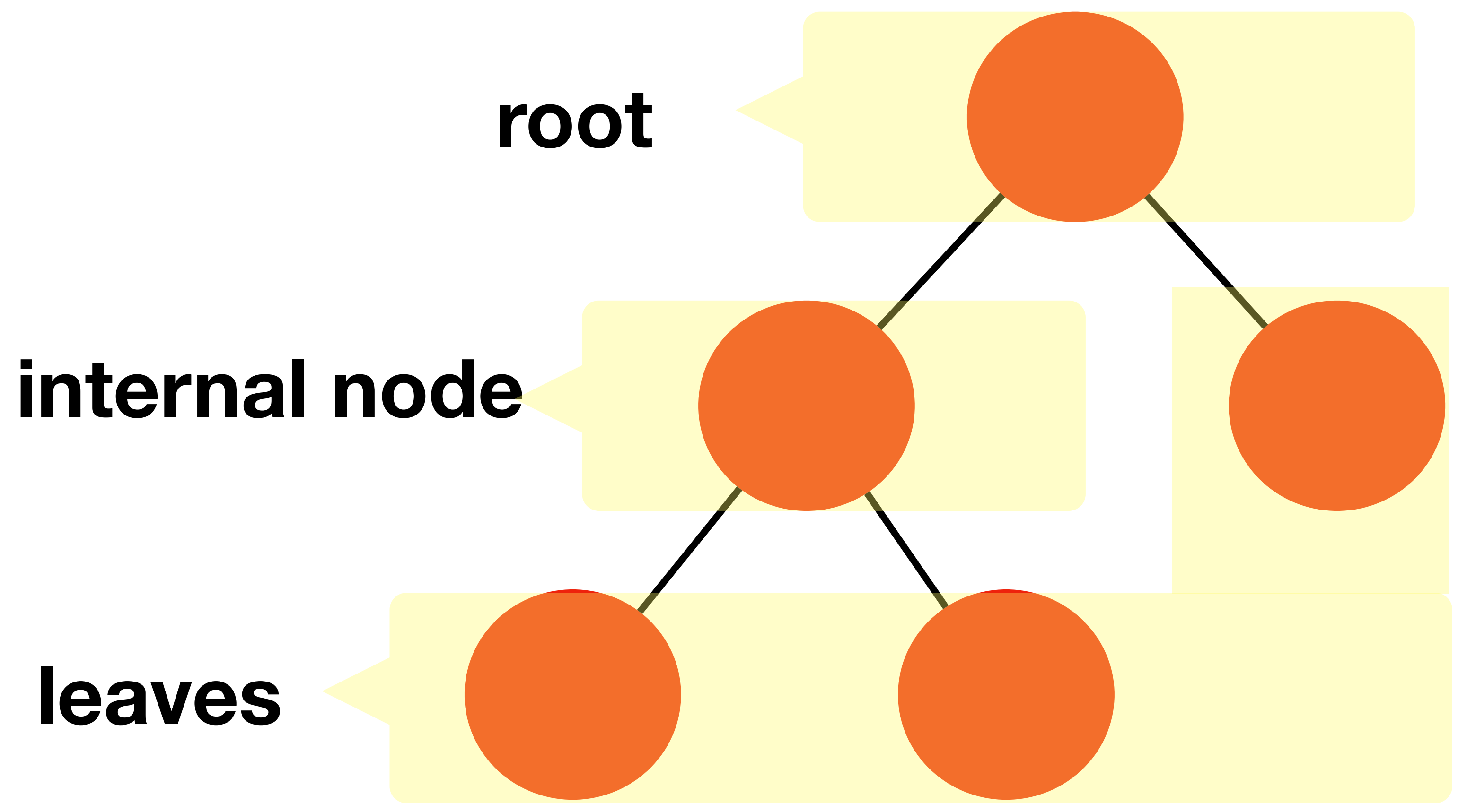
- *One-versus-one classification*: $\binom{K}{2}$ SVMs, comparing all pairs of classes. We assign the test observation to the class most frequently selected in the pairwise classification.
- *One-versus-all classification*: K SVMs, each time we compare a class K to the remaining K-1 classes. We assign the test observation to the class with the best discrimination rule.

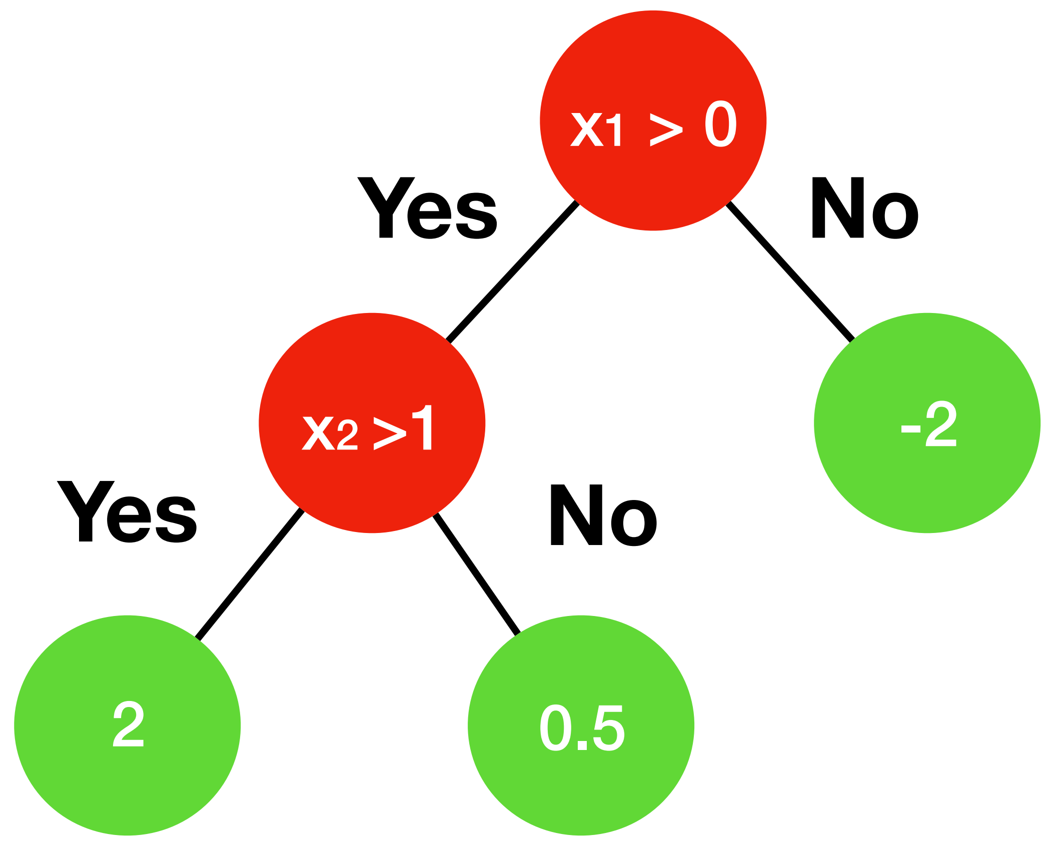
Summary

- **Separating Hyperplane:** A decision boundary that separates data points of different classes.
- **Maximising the Margin:**
  - **Hard Margin**: For the **Maximal Margin Classifier (MMC)**, perfectly separates data with no errors.
  - **Soft Margin**: For the **Support Vector Classifier (SVC)**, allows some misclassification to handle overlapping data.
- **Support Vectors:** Observations that lie on or inside the margins, or are misclassified. They are the only points that determine the hyperplane's position and orientation.
- **Regularisation**: Controls the trade-off between maximising the margin and allowing classification errors.
- **Linear and Non-linear Classification:** The **kernel trick** enables **Support Vector Machines (SVM)** to handle non-linear boundaries by mapping data into higher dimensions.

Tree-based methods

- Can be used for *regression* or *classification*.
- Work by partitioning the feature space into a set of rectangles by consecutive binary partitions.
- This can be summarised into a tree: decision-trees.

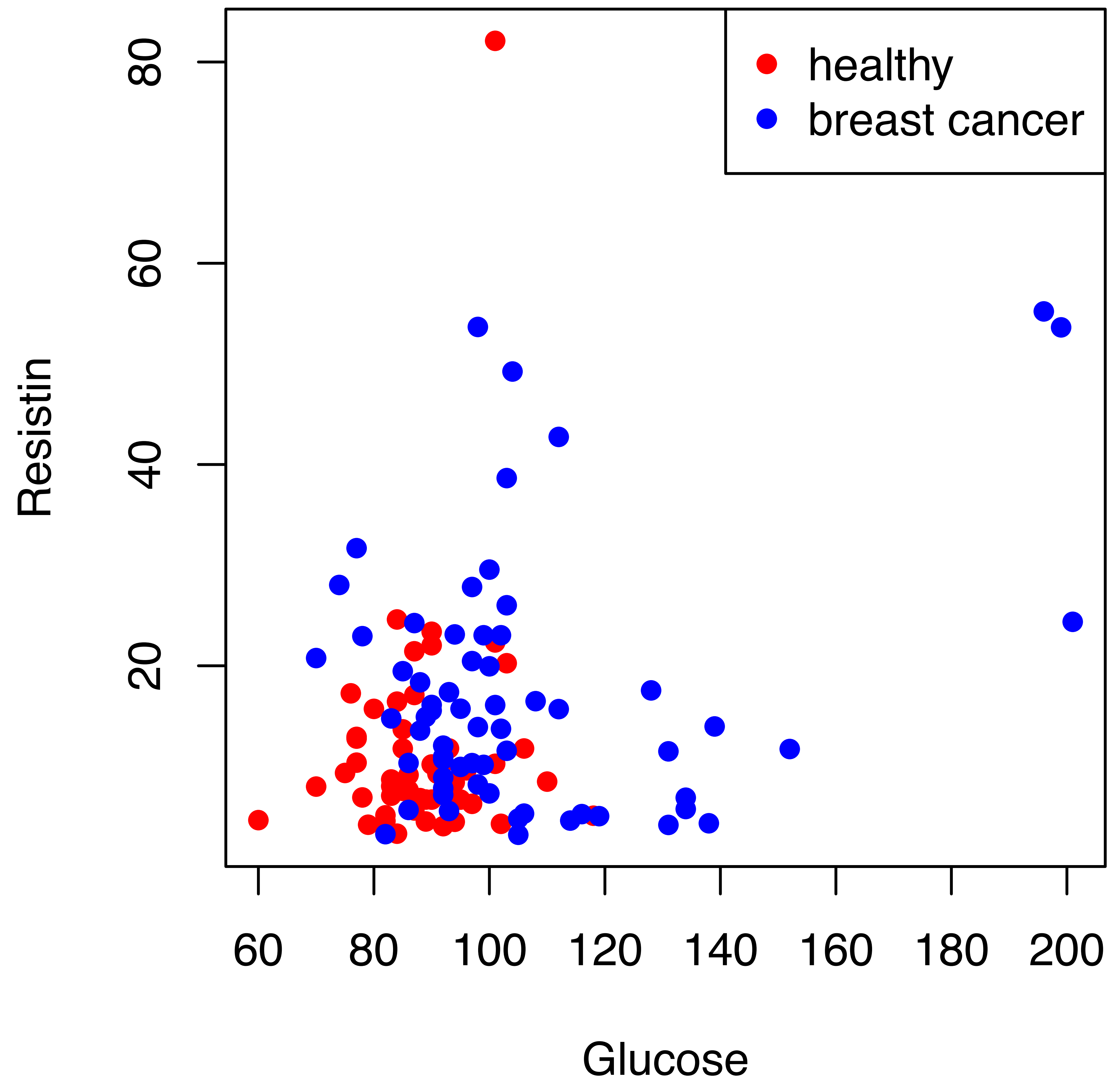




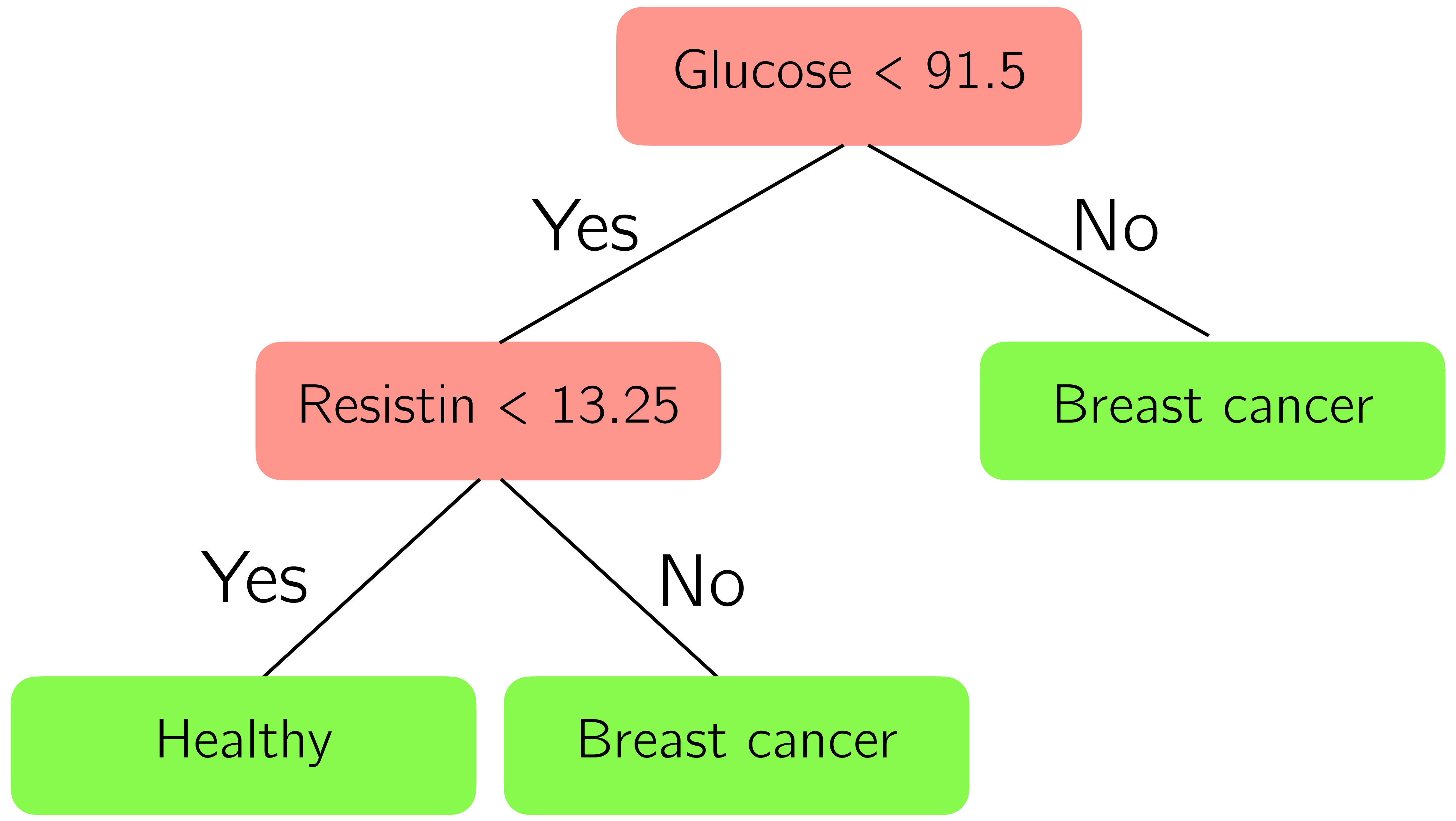
- **Splitting conditions**: root and internal nodes
- **Predictions**: leaves

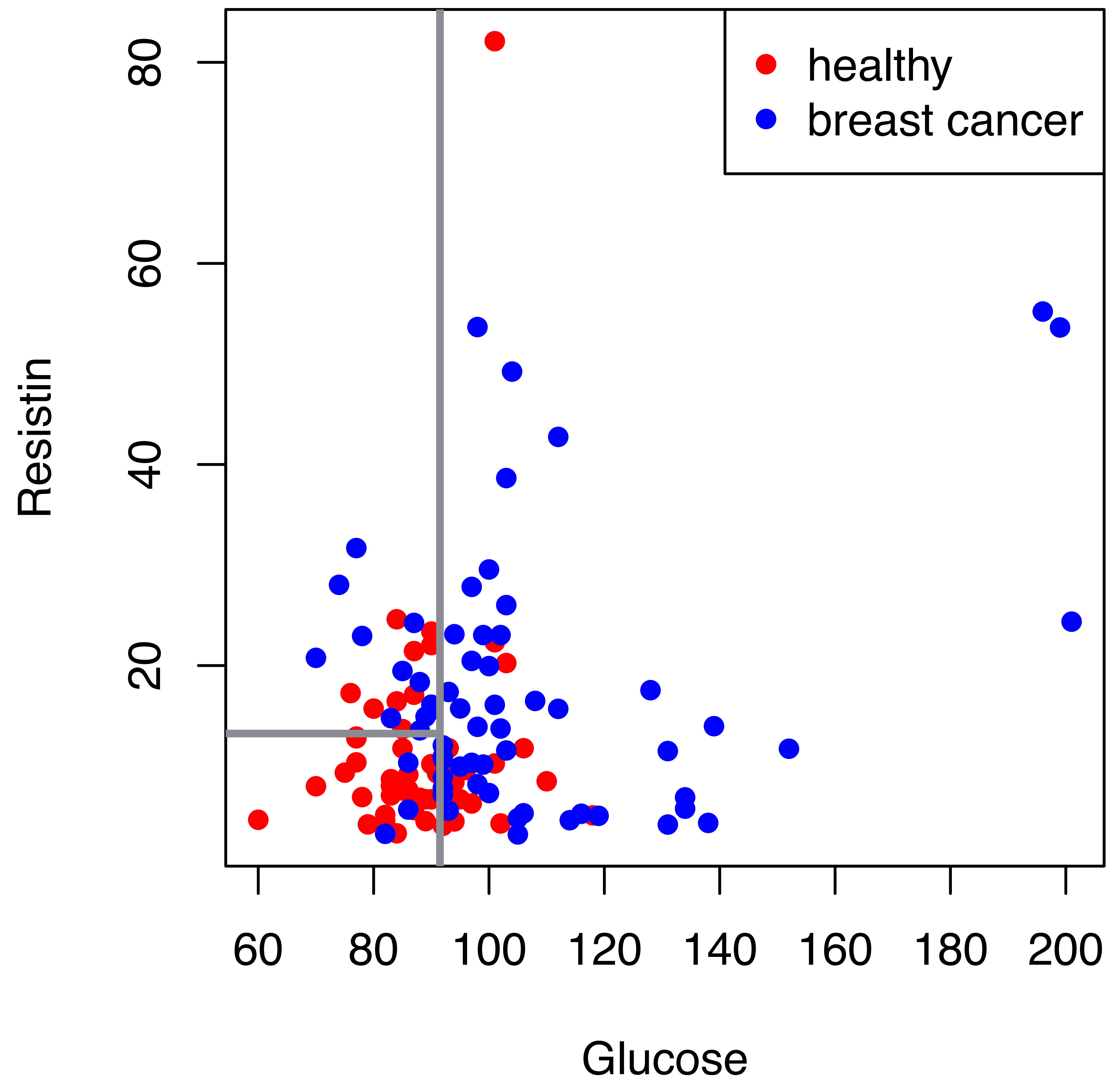
Example: breast cancer dataset

Coimbra breast cancer dataset (Patricio, M., et al, BMC Cancer, 2018)



Example: decision tree





The number of partitions correspond to the number of leaves.

How to build a decision tree

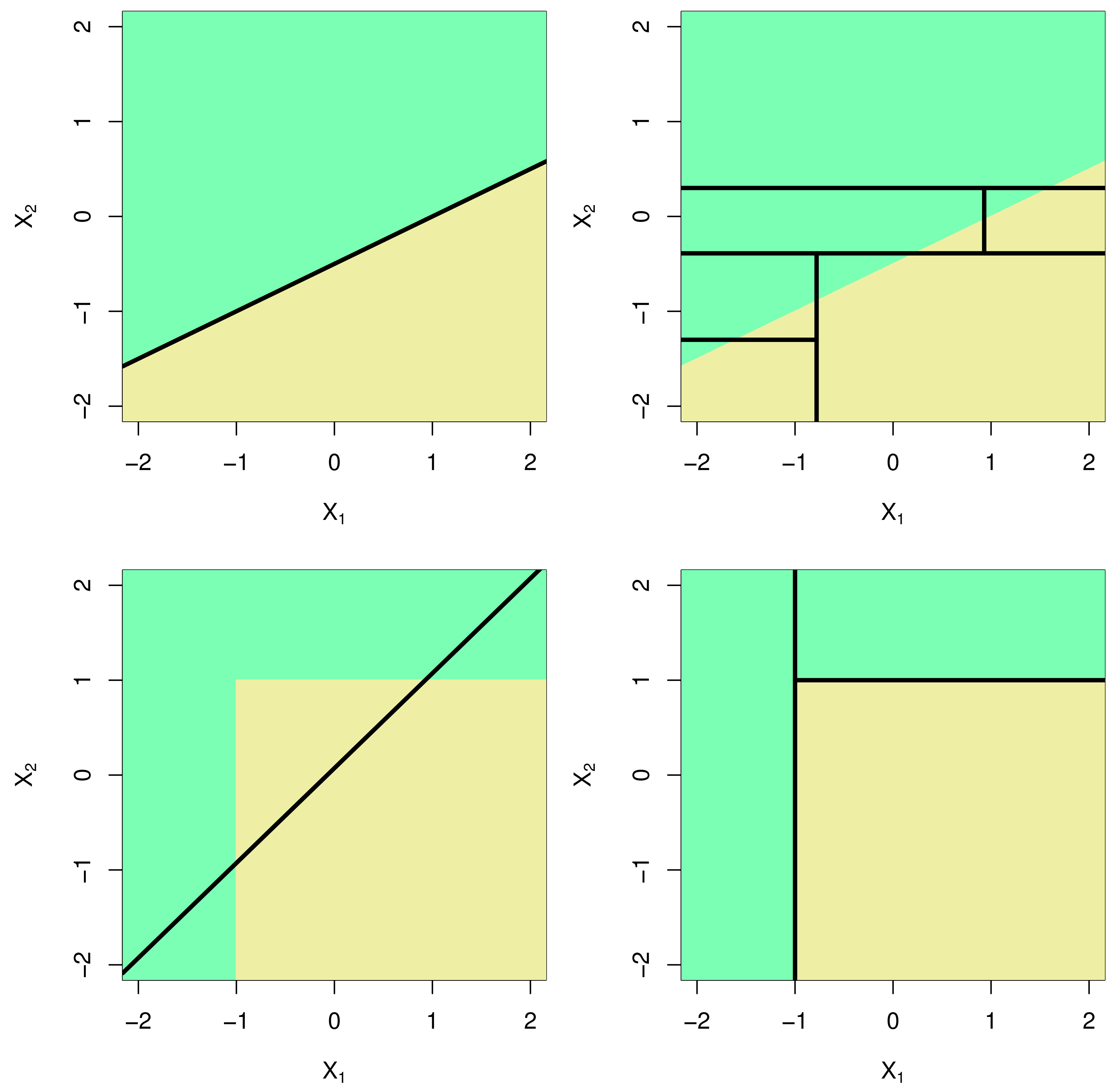
Consider a training data consisting in N observation pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, where each $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and p is the number of features.

The algorithm should build a decision tree that:

- Divides the predictor space in M non-overlapping regions R_j with $j = 1, \ldots, M$. These regions are high-dimensional rectangles (boxes).
- For each observation that fall in the same region R_m we make the same prediction, which is the average of the response values (for regression) or the most represented class (for classification) for the training observations in R_m.
- This partition should minimise the RSS (for regression) or the misclassification (for classification).

Mentimeter question (www.menti.com code 3680 2274)

For which ones the decision boundary was defined using a decision tree?

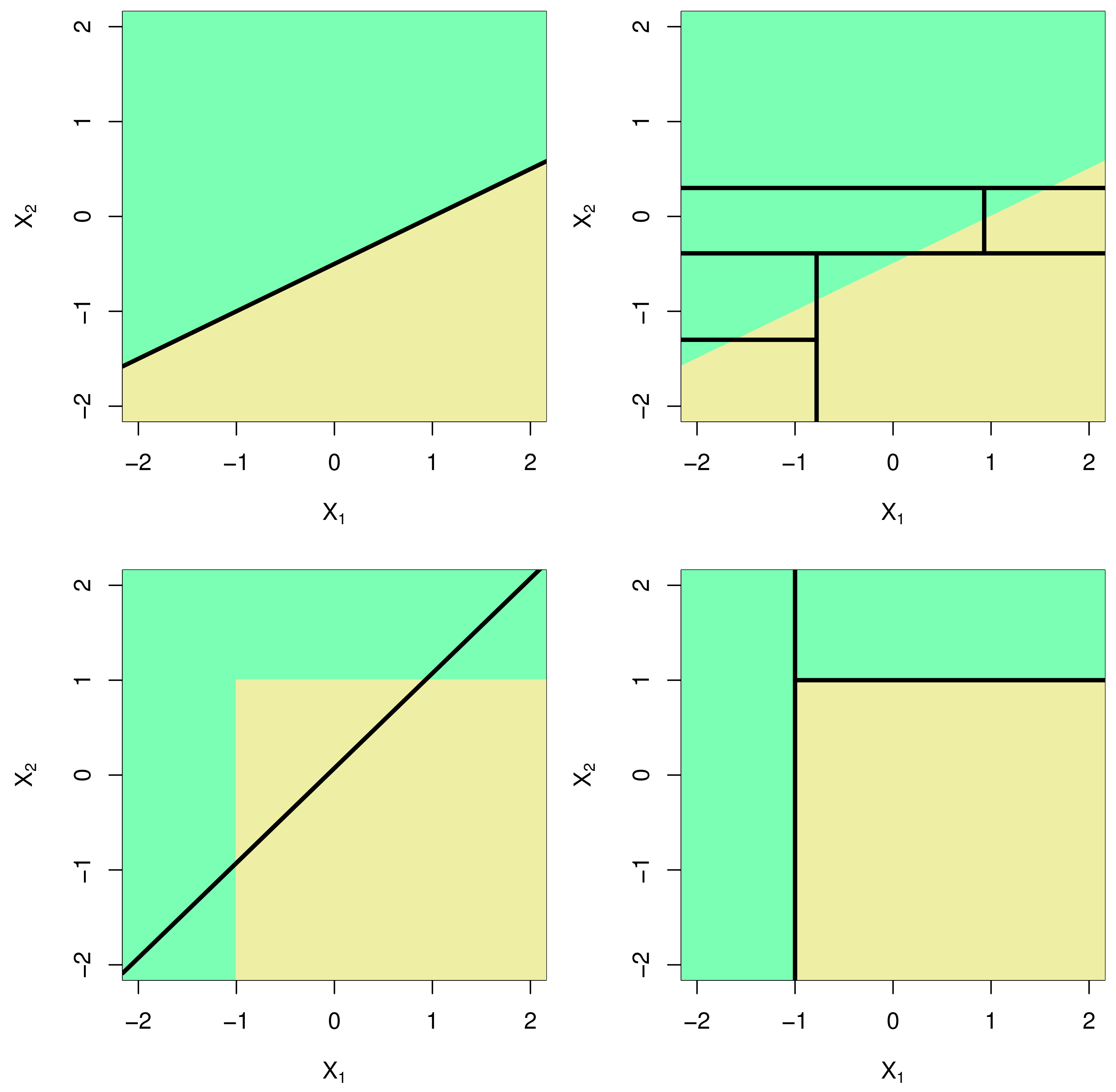


(multiple choices possible)

- A
- B
- C
- D

Mentimeter question (www.menti.com code 3680 2274)

Which scenarios are suitable for a decision tree?



(multiple choices possible)

- A
- B
- C
- D

How to build a decision tree

It is infeasible to evaluate every possible partition of the feature space.

We adopt a *recursive binary splitting* approach that is:

- *Top-down:* starts from the top of the tree and proceeds with subsequent splits, each split is indicated by two new branches.
- *Greedy:* it looks at the optimal splitting at that specific step of the tree, without looking ahead.

How to build a decision tree

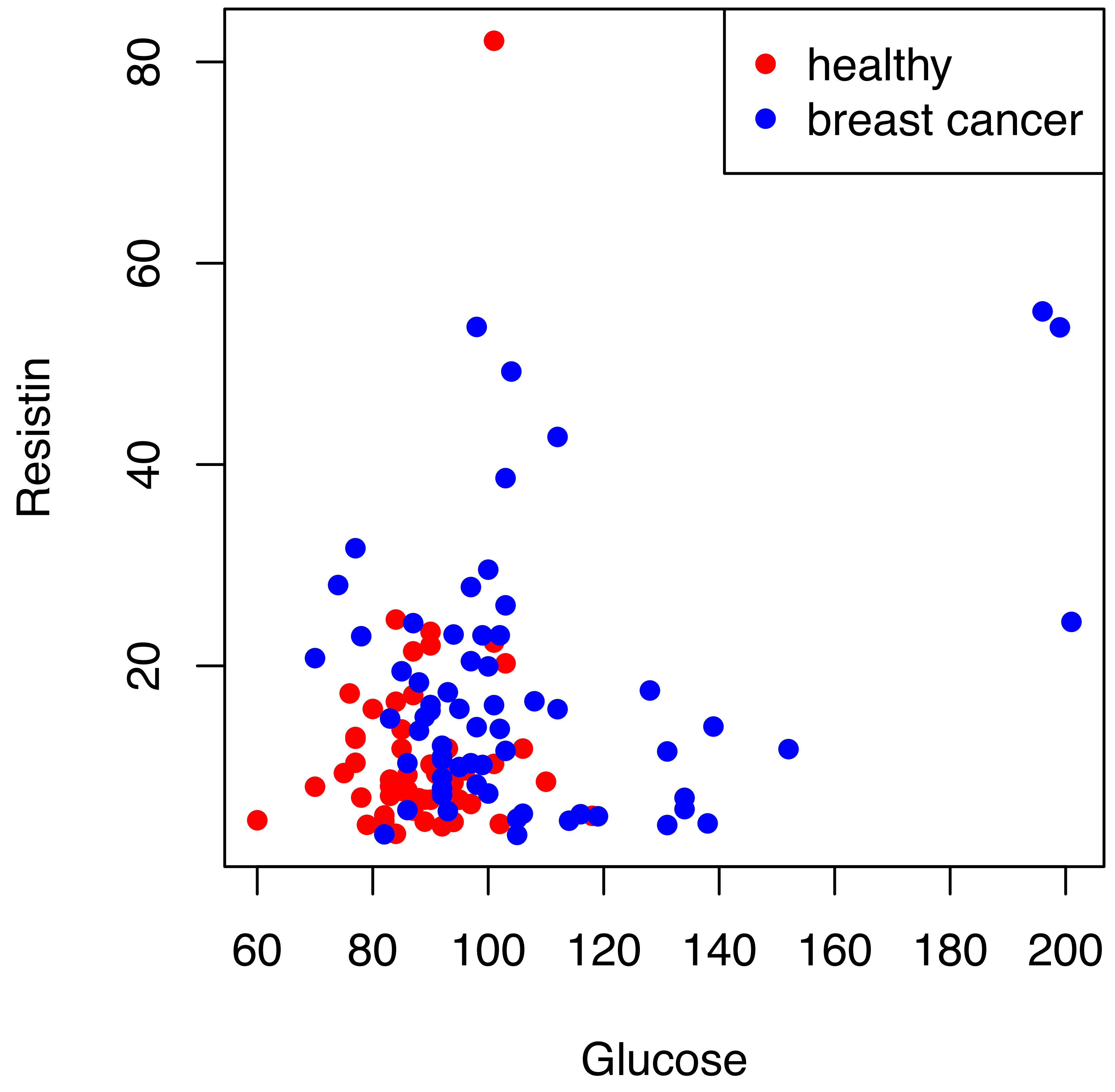
- We start from the top of the tree and we select the variable X_j and the splitting point s to define the pair of half-planes $R_1(j,s) = \{X|X_j < s\}$and $R_2(j,s) = \{X|X_j > s\}$ that leads to the greatest reduction of the cost function.

  Notation: $R_1(j,s) = \{X|X_j < s\}$ means the region of X is which X_j has a value less than s.

- For each of the resulting two regions, we repeat the procedure but looking only at the data in that half-plane.
- This continue until we reach a termination criterion (e.g. no region with more than 5 observations).

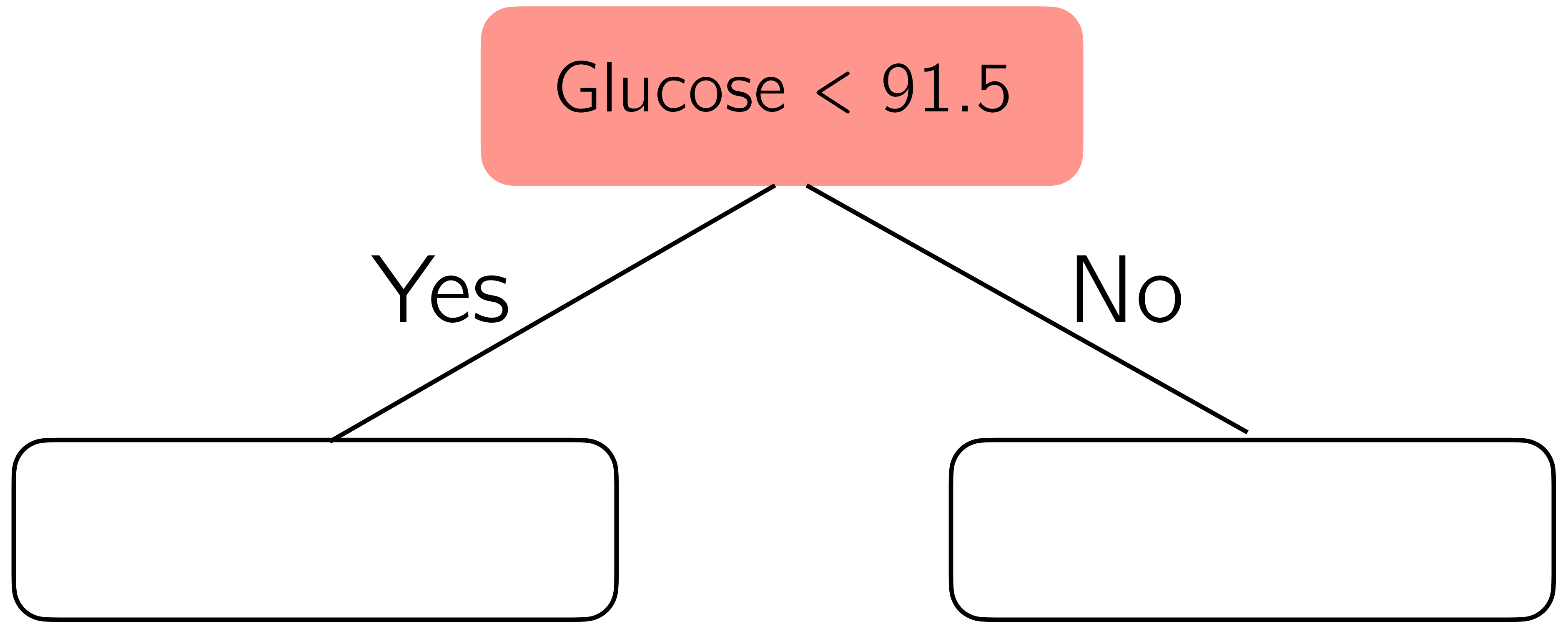
Example: how to build a decision-tree

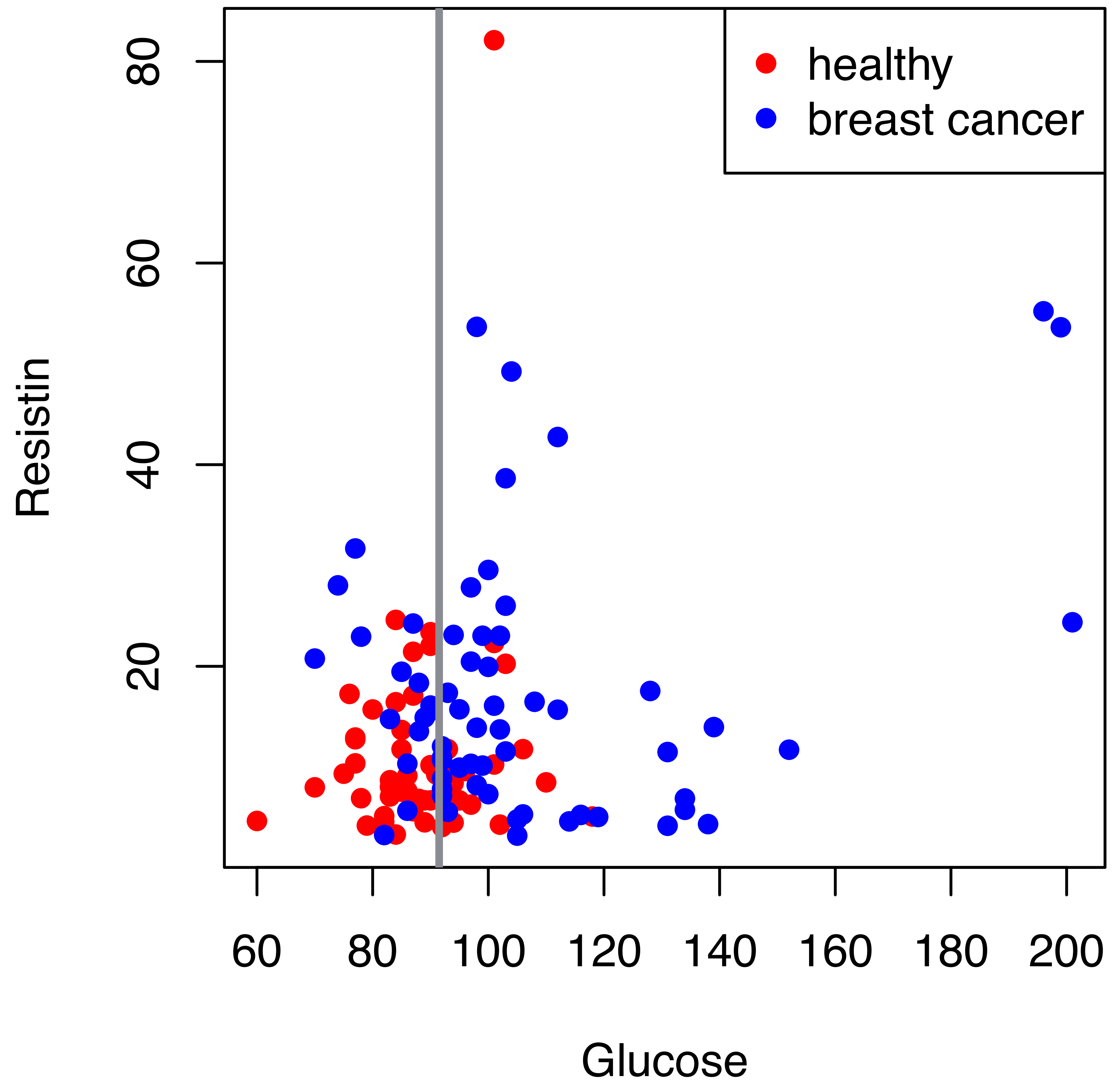
Chose which feature j and which splitting point s gives me the best partition (lowest misclassification error)



Example: how to build a decision-tree

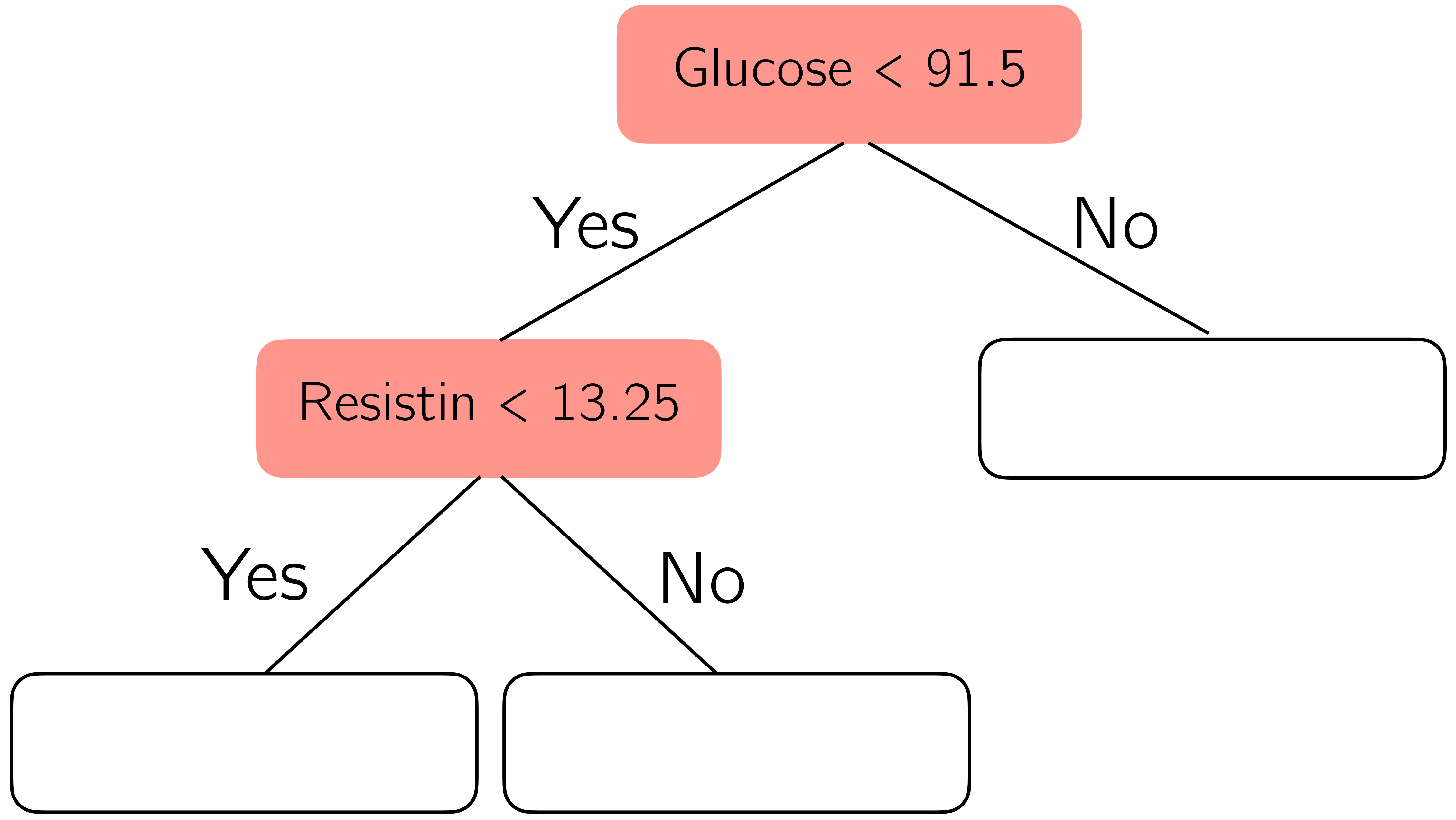
Chose which feature j and which splitting point s gives me the best partition (lowest misclassification error)

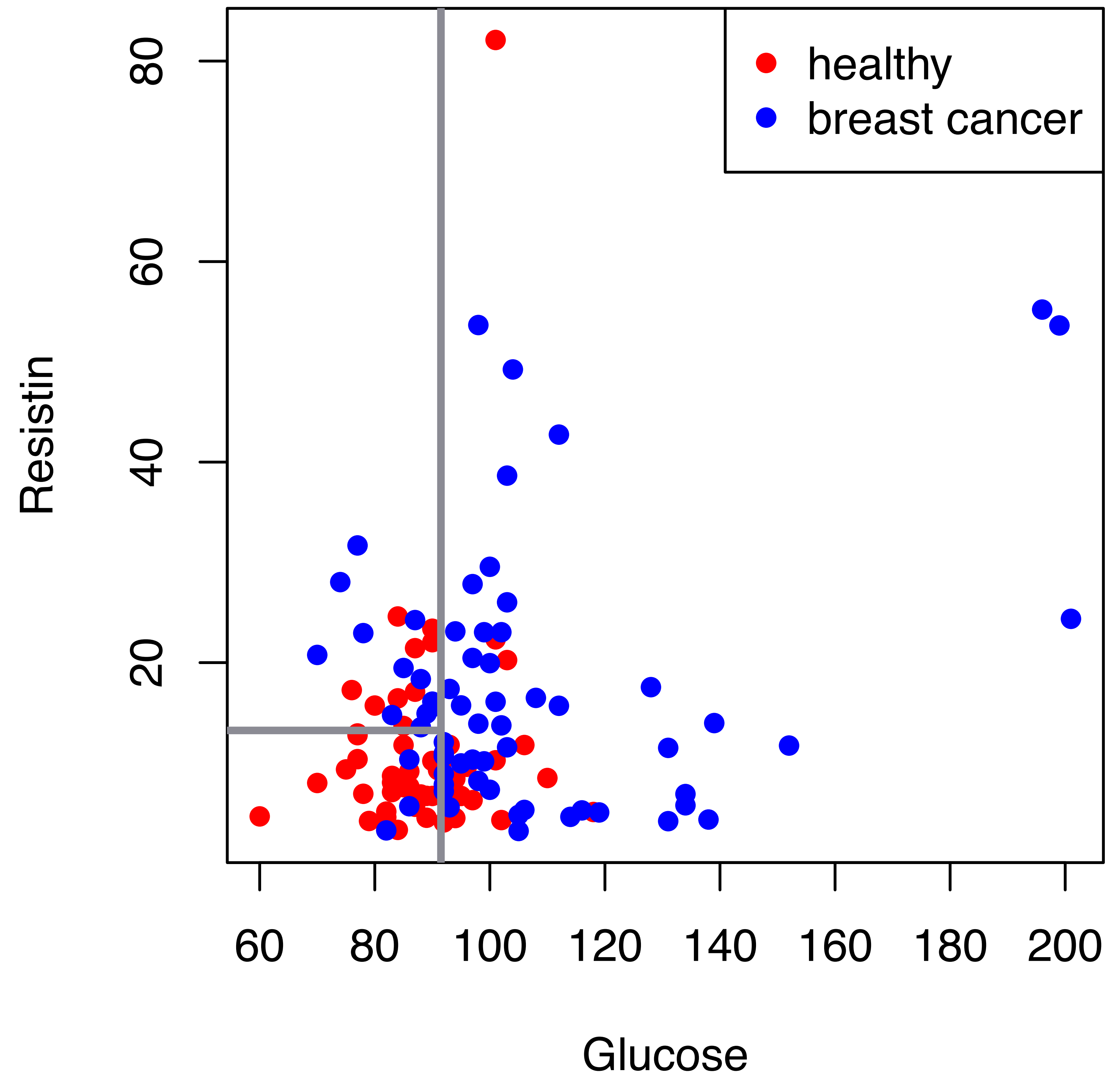




Example: how to build a decision-tree

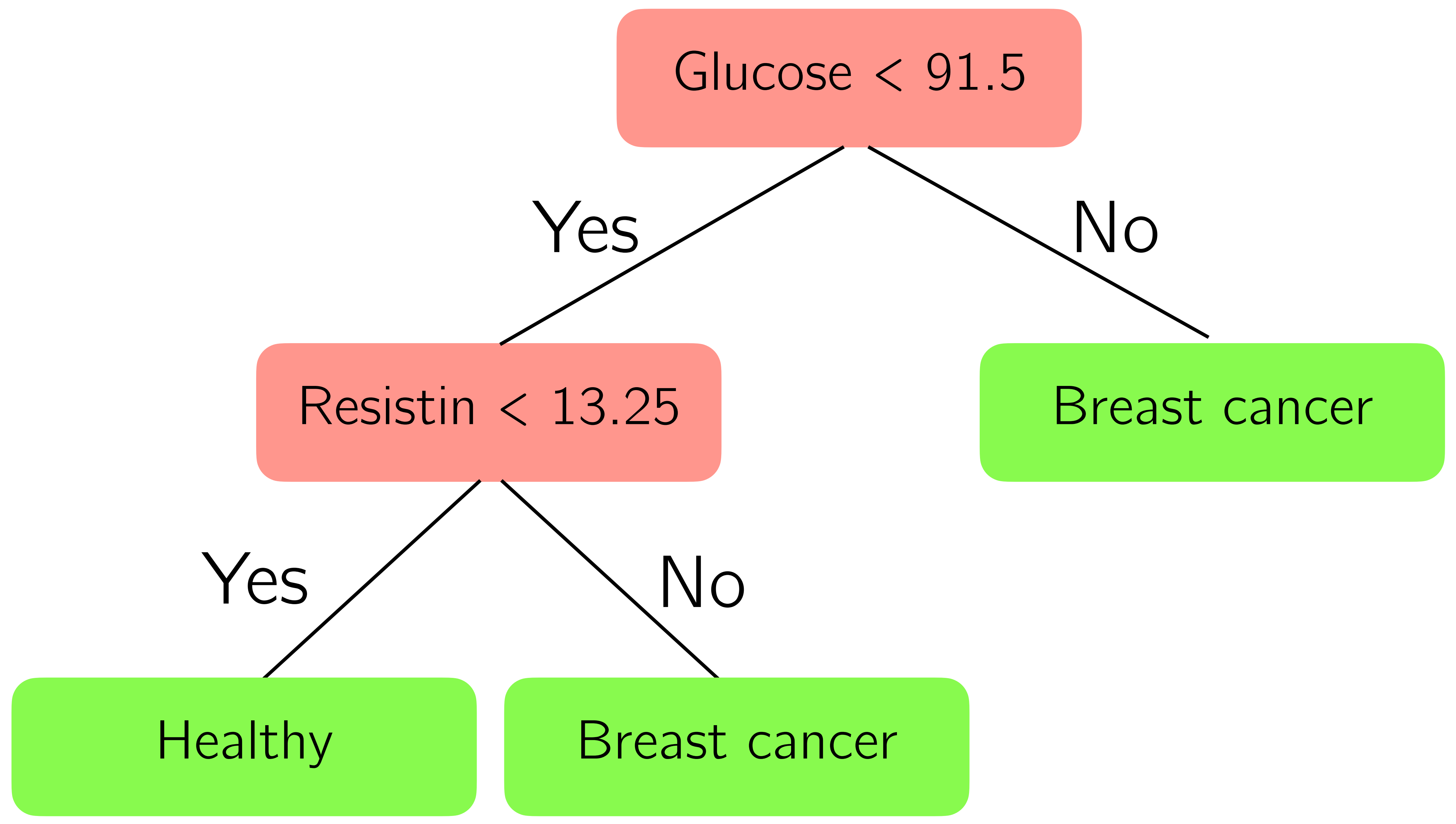
Chose which feature j and which splitting point s gives me the best partition (lowest misclassification error)

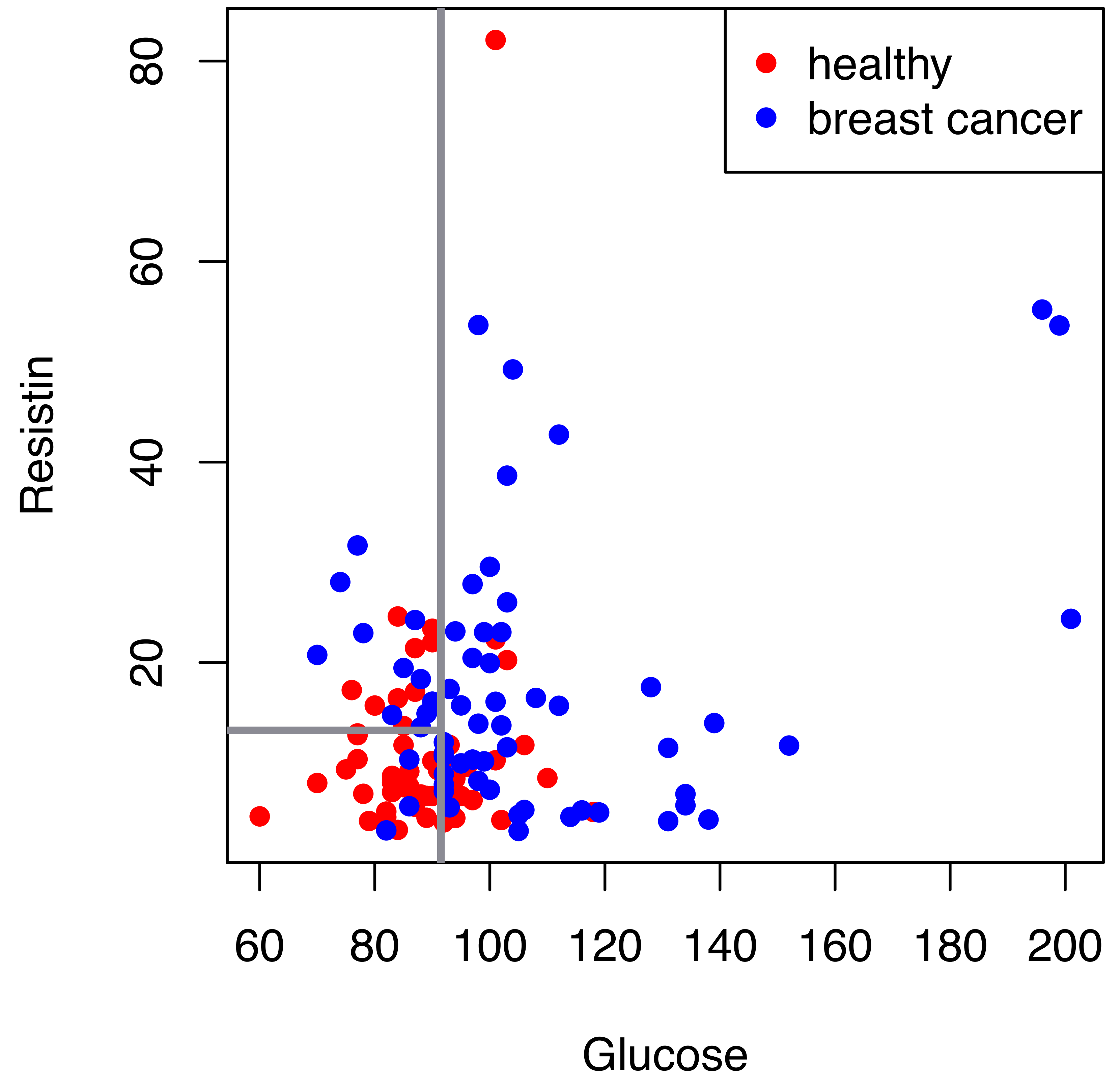




Example: how to build a decision-tree

When a termination criteria is reached, assign to each leave the most represented class.





Cost functions

- For regression: $$\sum_{j=1}^{J} \sum_{i: i \in R_j} (y_i - \hat{y}_{R_j})^2$$ where $$\hat{y}_{R_j} = \frac{1}{n_j} \sum_{i: i \in R_j} y_i$$

  Cost function: *Residual sum of squares (RSS)*

  Model prediction: average of the response values for the training observations in R_j

- For classification, considering $\hat{p}_{mk}$ as the proportion of training observations in region m that are from class k, we can define different metrics:

  - *Misclassification error:* $E = 1 - \max_k(\hat{p}_{mk})$
  - *Gini index:* $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
  - *Entropy:* $D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

  Preferred: more sensitive to node purity

Mentimeter question (www.menti.com code 3680 2274)

What happens for a **regression tree** if there is no termination criterion?

- The tree goes to infinity
- Each leaf has one observation only
- It is not possible to say without seeing the data

Mentimeter question (www.menti.com code 3680 2274)

What happens for a **classification tree** if there is no termination criterion?

- The tree goes to infinity
- Each leaf has one observation only
- It is not possible to say without seeing the data

Predictions using a decision-tree

What value will each leaf predict?

- For regression: the average of the training observations falling in the region R_m of the leaf m.
- For classification: the most occurring class in the region R_m of the leaf m.

Mentimeter question (www.menti.com code 3680 2274)

What is the risk of having a tree with a lot of leaves?

Tree pruning

Idea: build a large tree T_0 and then prune it back to a subtree T

This is done defining a cost complexity criterion:

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

Where:

- $|T|$ is the number of leaves in a subtree T
- α is a regularisation parameter.

The idea is shown for regression, can be extended to classification problems.

Pros and cons of decision trees

Pros:

- Easily interpretable and explainable
- Mirror human decision-making process (appealing for clinical decision making)
- Graphical representation

Cons:

- Poor prediction accuracy
- Sensitive to the training data

Solutions: combine different trees to derive a consensus prediction.

Combining a large number of trees can improve predictions at the price of losing a bit interpretability.

Bagging (or bootstrap aggregation)

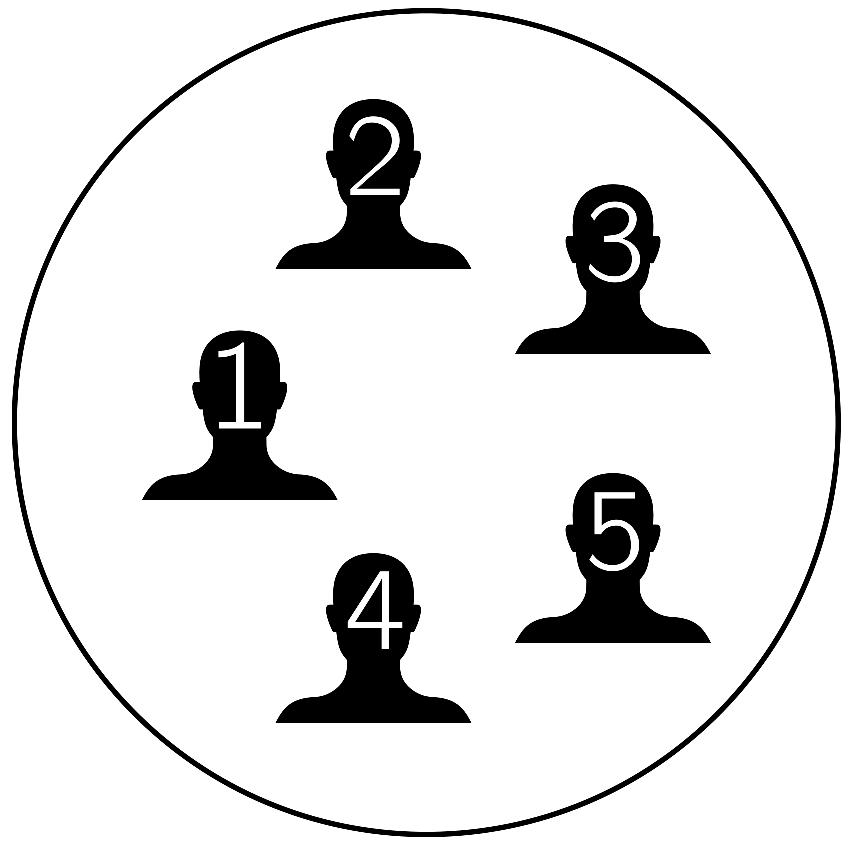
General concept: Given a set of n independent observations $Z_1, \ldots, Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is σ^2/n.

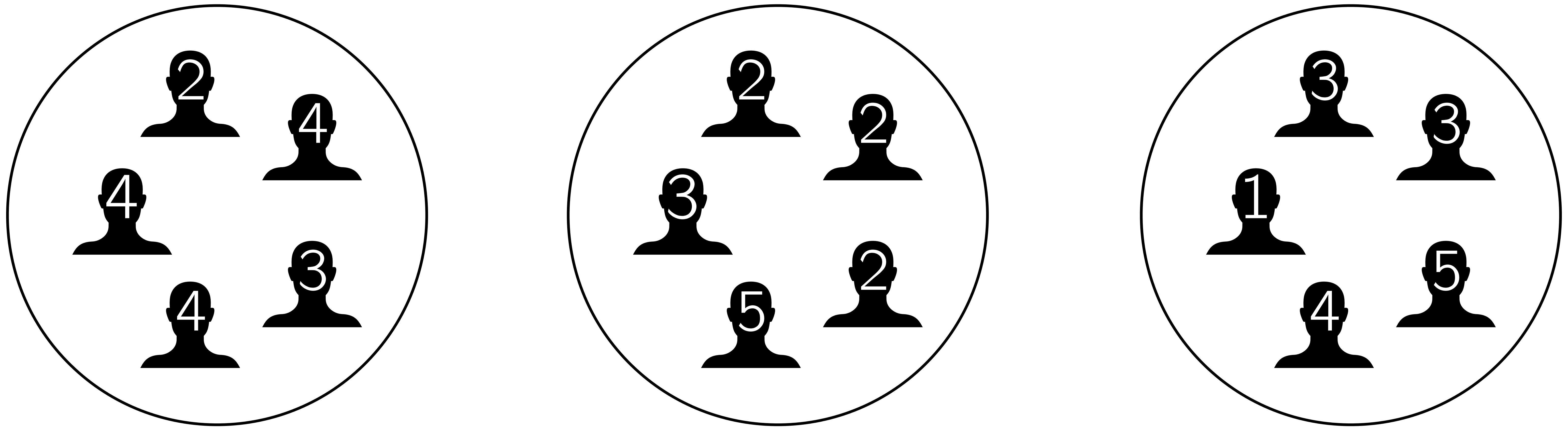
Idea: Instead of pruning big trees, build multiple independent big trees and average their predictions.

How to build independent trees with only one dataset?
With **bootstrap.**

Bootstrap

Repeatedly sampling (with replacement) the sample dataset to create many simulated datasets





Original sample sata

B sets of bootstrap samples ($B = 3$)

How to do bagging?

1. Generate B training datasets by bootstrapping (sampling with replacement).
2. Build a decision tree from each bootstrapped dataset (without pruning).
3. Obtain a final prediction by averaging (regression) or majority vote (classification)

Bagging reduces the variance without increasing the bias.

Out-of-bag error estimation

How to estimate the test error of a bagged model?

Idea: On average, each bagged tree makes use of about $2/3$ of the observations. We can use the remaining $1/3$, called *out-of-bag observations (OOB)*, to estimate the prediction error.

For each observation i we consider all the trees in which the observation was *OOB*. This yields to about $B/3$ predictions for that observation that can be averaged.

Variable importance measure

Problem: we reduce variance but at the price of losing interpretability.

Idea: Obtain a measure of the importance of each predictor looking at how much they decrease the cost function (e.g. RSS for regression, Gini index for classification) on average across the B trees.

Random forests

How can we improve performances over *bagging*?
Performing random subset selection of the features. This decorrelates the trees thus further reducing variance.

Compared to *bagging*, for ***random forests*** the procedure of building the trees from the bootstrapped datasets (step 2) changes as follow:

- *Bagging*: at each split the best feature for the split is selected across all the p features.
- *Random forests*: only a subset of m features is considered as possible candidate. A typical choice of m is $m \approx \sqrt{p}$.

Steps 1 and 3 remain unchanged.

Boosting

Differently from *bagging* and *random forests* (where *big* trees are build *independently*), with ***boosting*** *small* trees are grown *sequentially*.

- B trees are build sequentially, each with d splits
- Each tree fits a shrunken version of the residuals of the previous tree, compensating partially the bias of the previous tree.
- The higher the number of trees B, the smaller the bias and the higher the variance.

Choice of tuning parameters:

- B (number of trees): cross-validation.
- λ (shrinkage factor): typically 0.01 or 0.001 (note: small λ will require large B).
- d (number of splits for each tree): typically 1.

Summary

- **Decision Trees:** Recursive binary splitting of features to create a simple, interpretable model, prone to overfitting. Pruning can only partially address this problem.
- **Bagging:** Combines multiple decision trees trained on bootstrapped datasets to reduce variance.
- **Random Forests:** An extension of bagging that introduces randomness by selecting a subset of features at each split. De-correlates trees thus further reducing variance.
- **Boosting:** Sequentially builds trees, each focusing on correcting the errors of the previous ones. At each iteration bias is reduced but variance is increased.
- **Key trade-off:** ensemble methods (bagging, random forests, boosting) improve prediction accuracy but may sacrifice interpretability.

Questions?

Some figures where adapted from the book “An introduction to statistical learning” G. James, D. Witten, T. Hastie, R. Tibshirani, J. Taylor