Classification

Ekhine Irurozki

Classification

- Supervised learning: n features, a categorical class variable Y
- A classifier is a ML algorithm for classification. Training, learning,..
- Approaches:
  - Separation of the instances of each class in the feature space
  - Learning the distribution of the data

X_1	X_2		X_{n-1}	X_n	Y
3	6	...	5	9	1
5	1	...	5	6	0
4	6	...	5	5	0
7	89	...	23	85	1
3	435	...	3	1	1
...	...	...	...	...	...
...	...	...	...	...	...
8	1	...	77	321	0
9	8	...	6	8	1
4	77	...	3	132	0
8	9	...	1	8	0
9	8	...	4	8	?

OvO - OvA

- Some classification algorithms can only distinguish between two classes, how can we use them in multi class problems?
    - One vs One strategy: train k(k-1)/2 binary classifiers. At prediction stage we will assign a new observation to the class that wins in more classifiers
    - One vs All (or One vs Rest) strategy: one classifier per class. The outcome must be numerical, representing the confidence on it

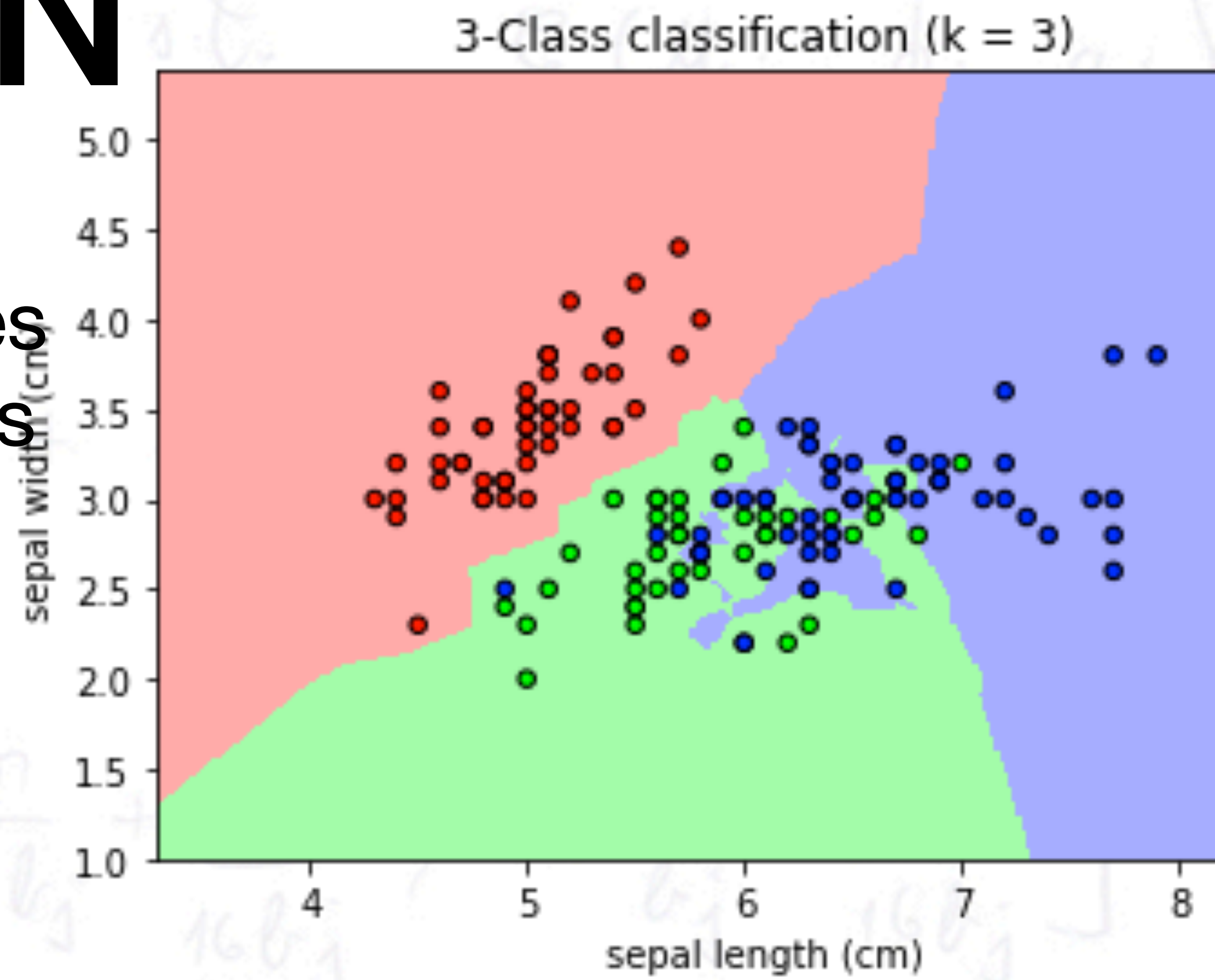
Outline

- K Nearest Neighbour, KNN
- Decision trees, bagging, random forests, boosting
- Logistic regression
- Discriminant analysis: LDA, QDA, GaussianNB
- Support Vector Machines, SVM

Outline

- **K Nearest Neighbour, KNN**
- Decision trees, bagging, random forests, boosting
- Logistic regression
- Discriminant analysis: LDA, QDA, GaussianNB
- Support Vector Machines, SVM

KNN



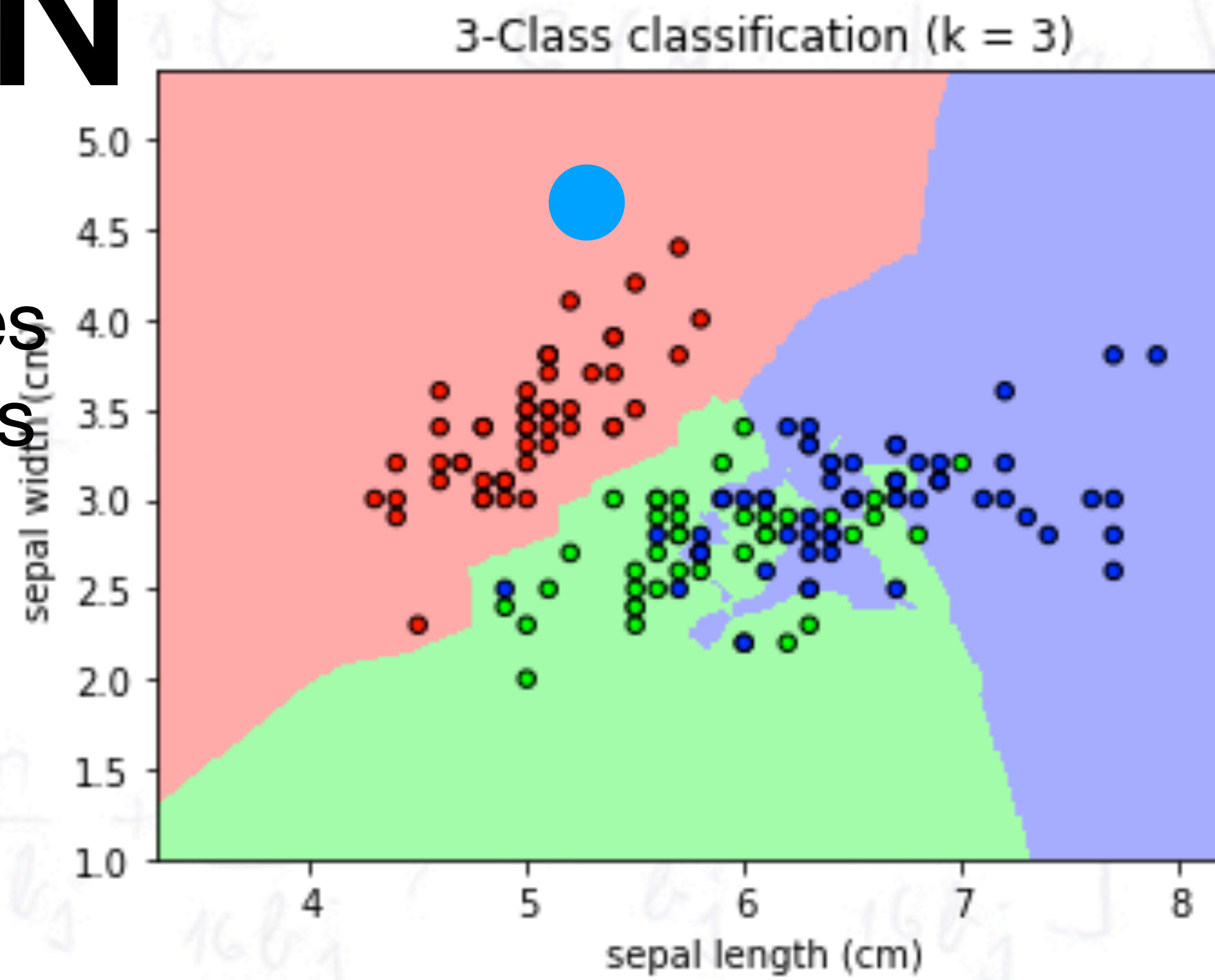
- It is a *lazy* algorithm which does not learn any model and makes computations in classification time

- Given a dataset *D* a new instance for prediction *X*

    - Let D' be the *k* closest instances to *X* on *D*

    - Assign *X* to the most popular class on *D'*

KNN



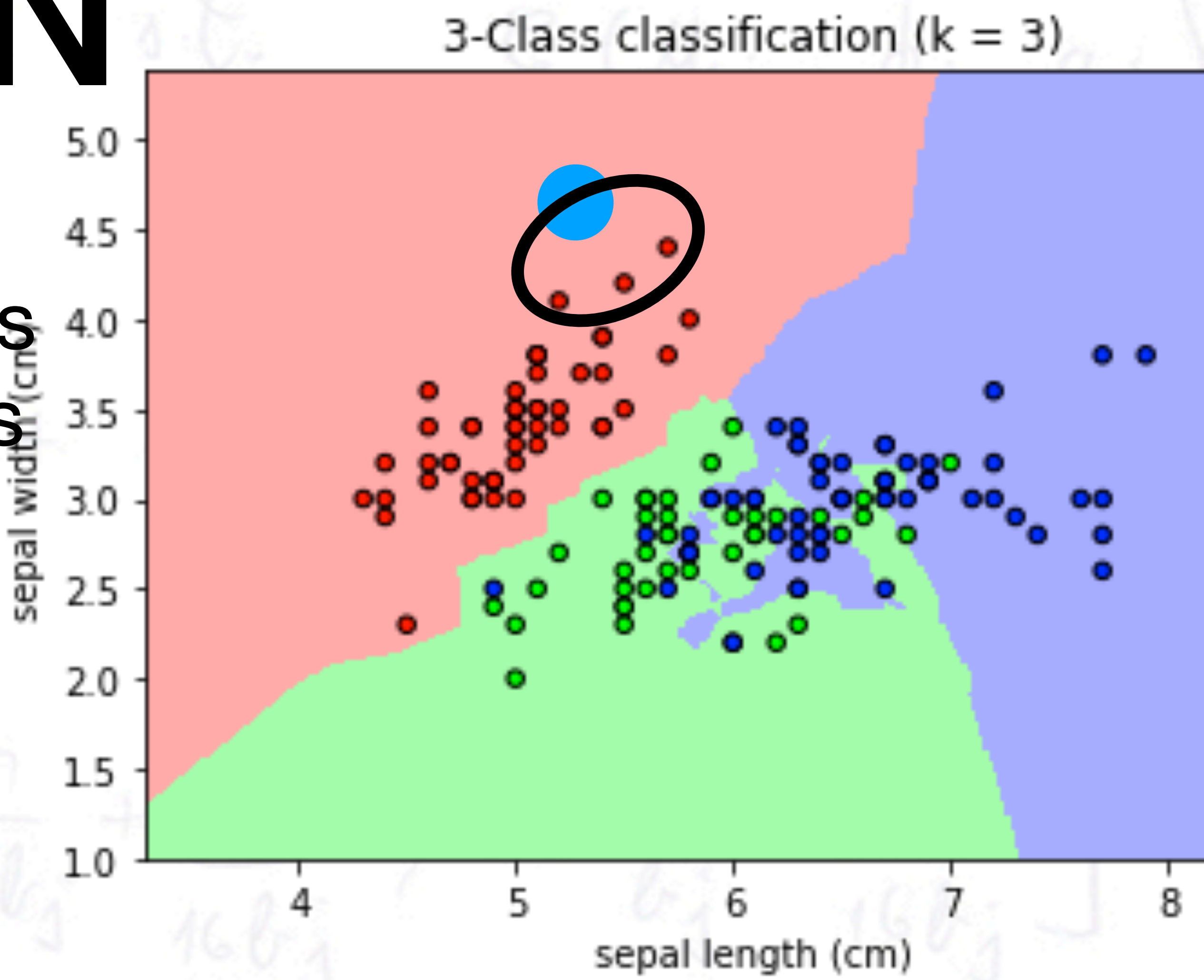
- It is a *lazy* algorithm which does not learn any model and makes computations in classification time

- Given a dataset *D* a new instance for prediction *X*

  - Let D’ be the *k* closest instances to *X* on *D*

  - Assign *X* to the most popular class on *D’*

KNN



- It is a *lazy* algorithm which does not learn any model and makes computations in classification time

- Given a dataset *D* a new instance for prediction *X*

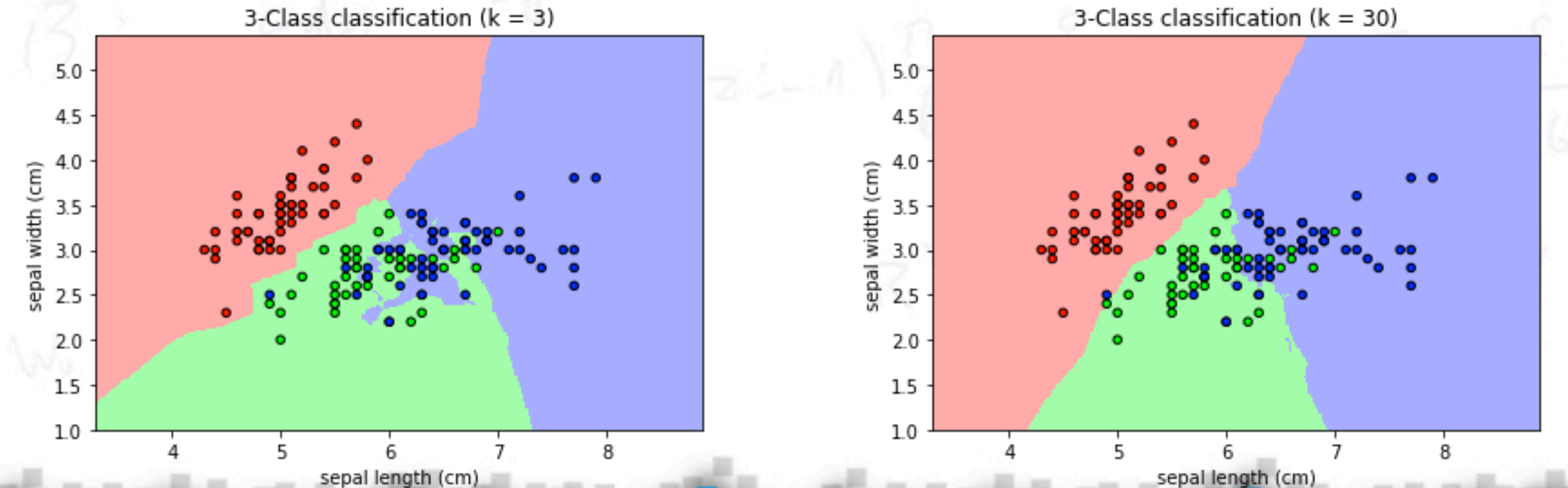
  - Let D' be the *k* closest instances to *X* on *D*

  - Assign *X* to the most popular class on *D'*

KNN

- What happens as we change k? The decision boundaries get smoother and we control overfitting

- How do we choose k?



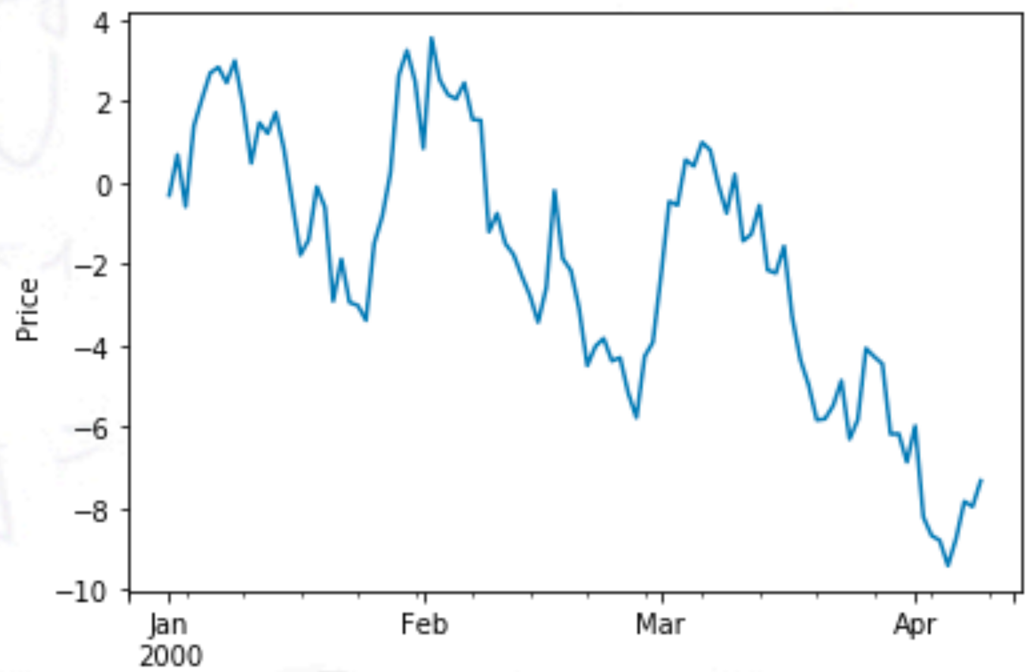
KNN

- If we can define similarity measures for the features we can use the KNN
- Time series, a series of data points indexed sequentially

KNN

- If we can define similarity measures for the features we can use the KNN

- Time series, a series of data points indexed sequentially

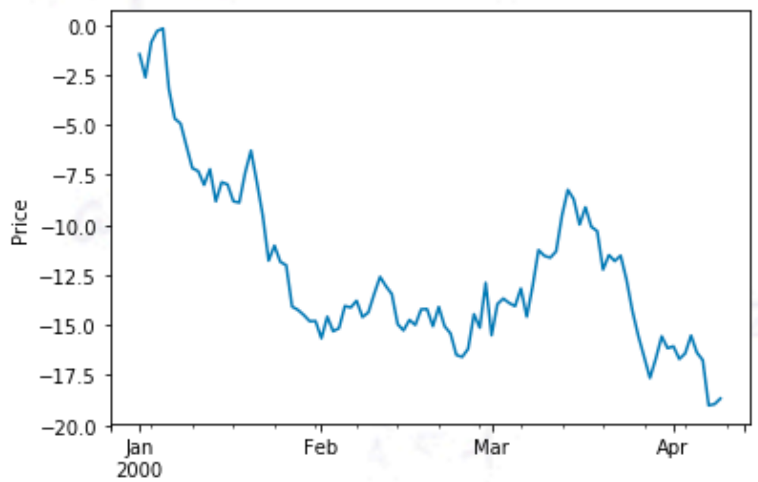


KNN - TS classification

Instance | **Class**

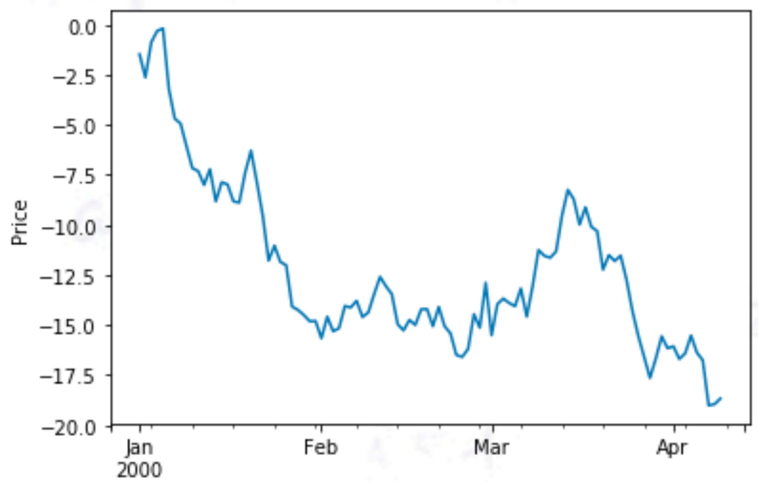
KNN - TS classification

Instance **Class**



KNN - TS classification

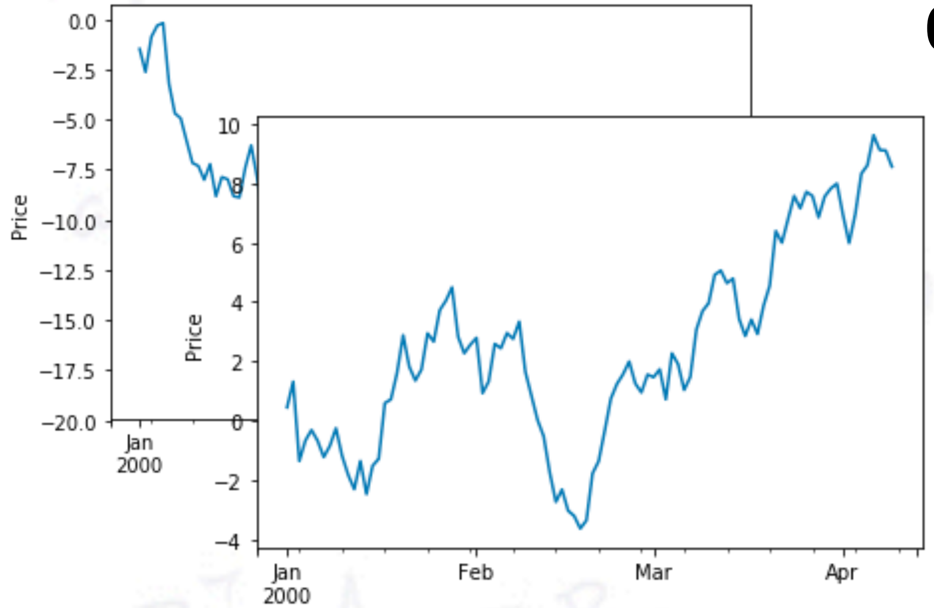
Instance **Class**



0

KNN - TS classification

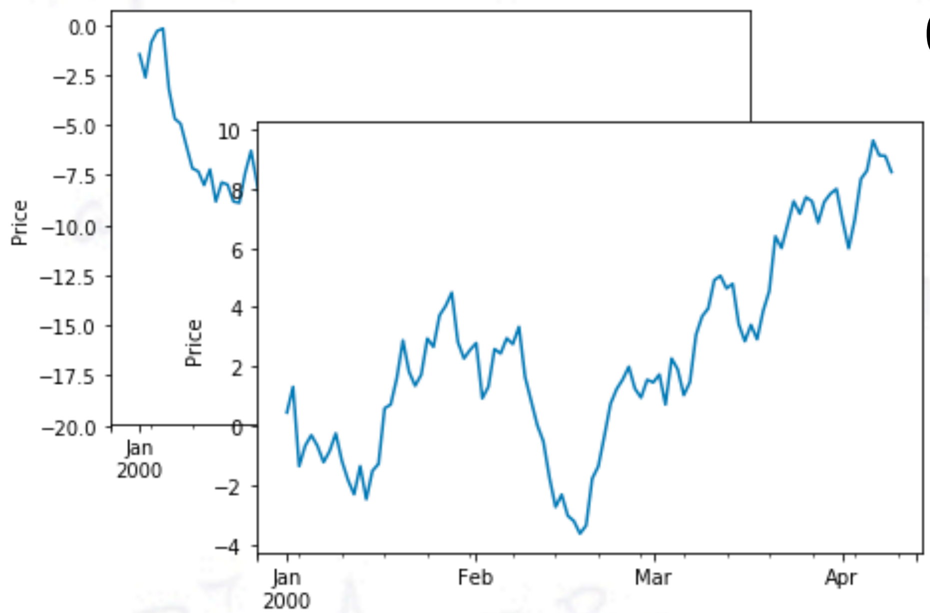
Instance **Class**



0

KNN - TS classification

Instance **Class**

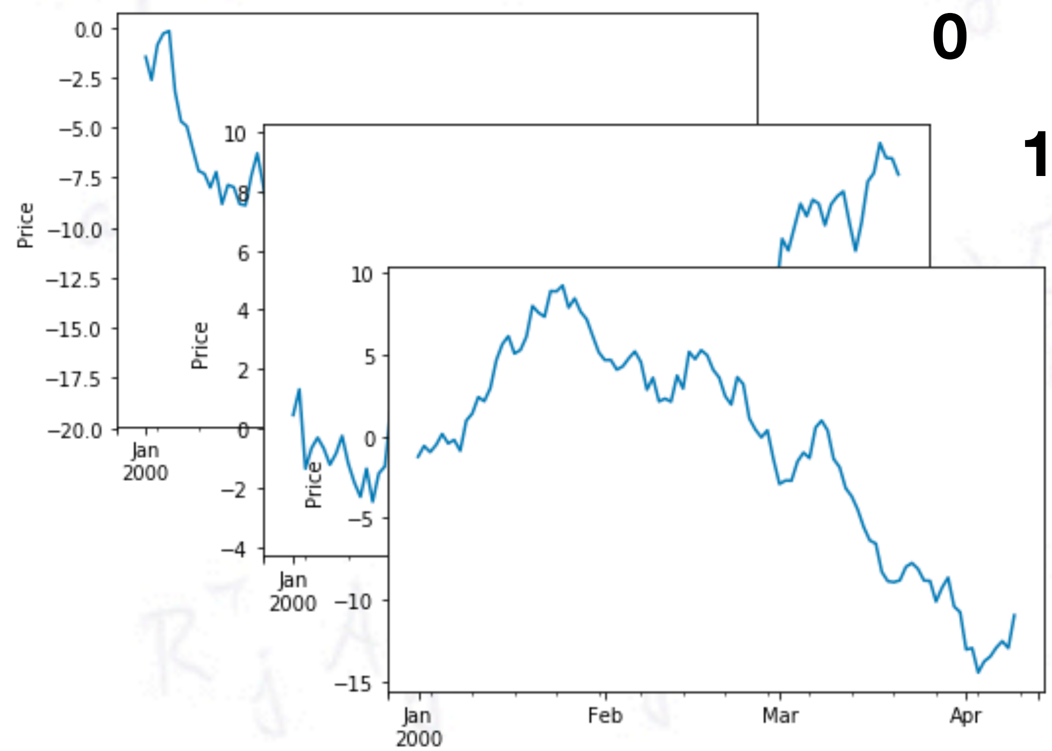


0

1

KNN - TS classification

Instance **Class**

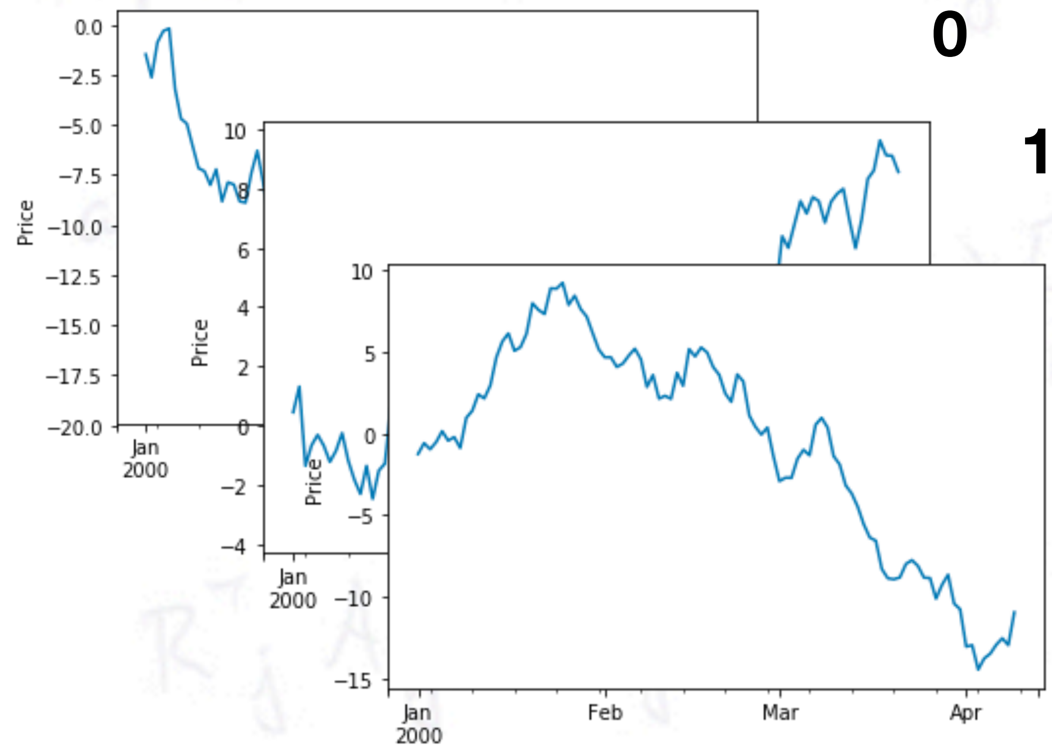


0

1

KNN - TS classification

Instance **Class**



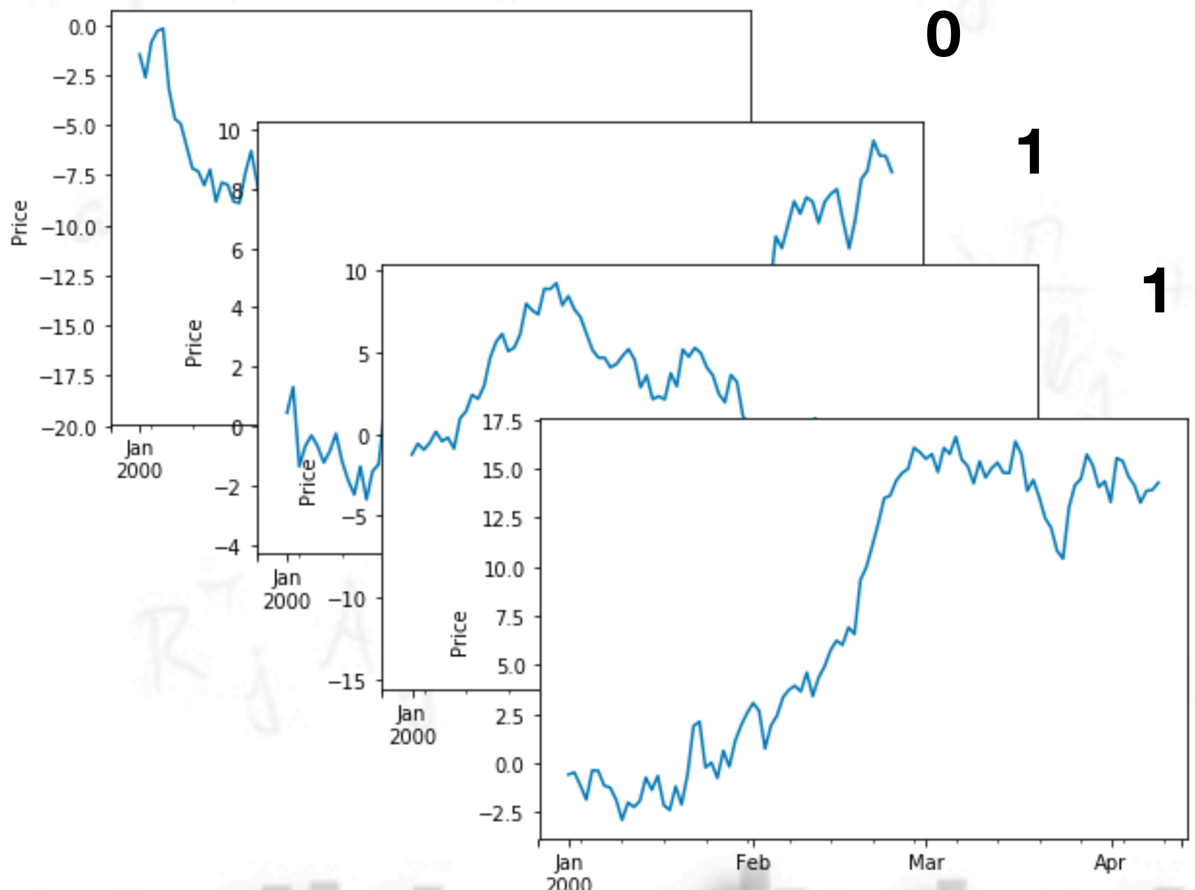
0

1

1

KNN - TS classification

Instance **Class**

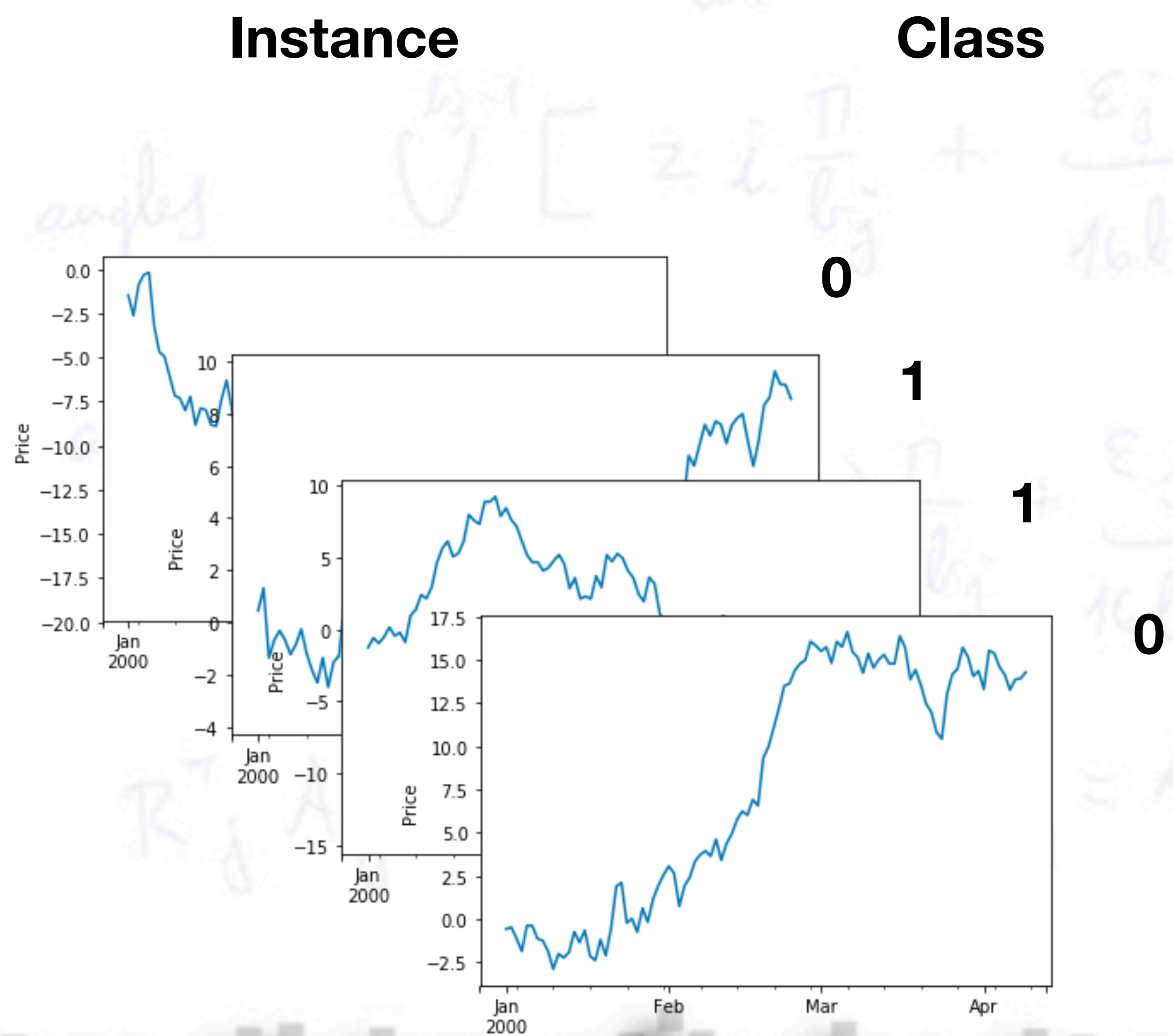


0

1

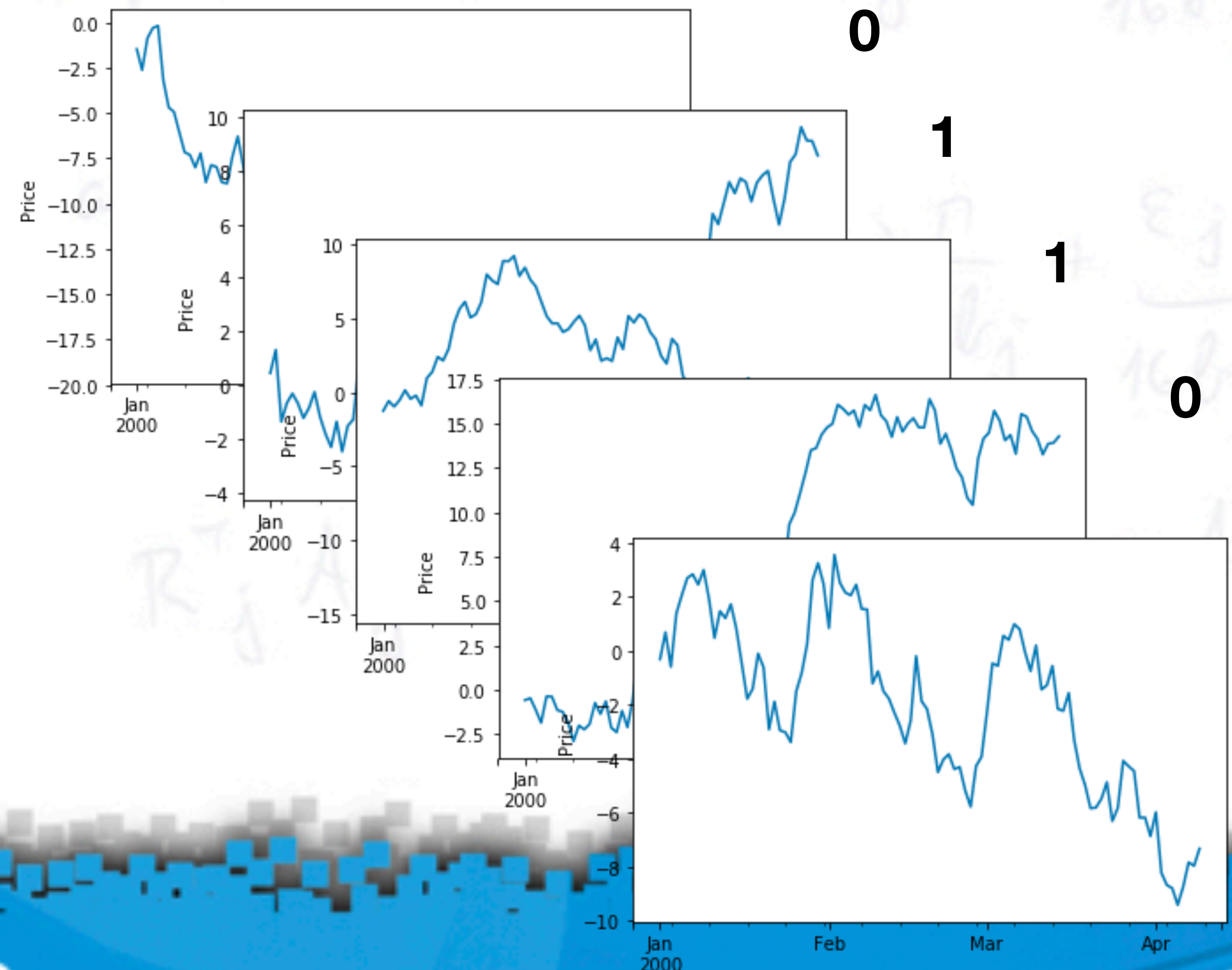
1

KNN - TS classification



KNN - TS classification

Instance **Class**



0

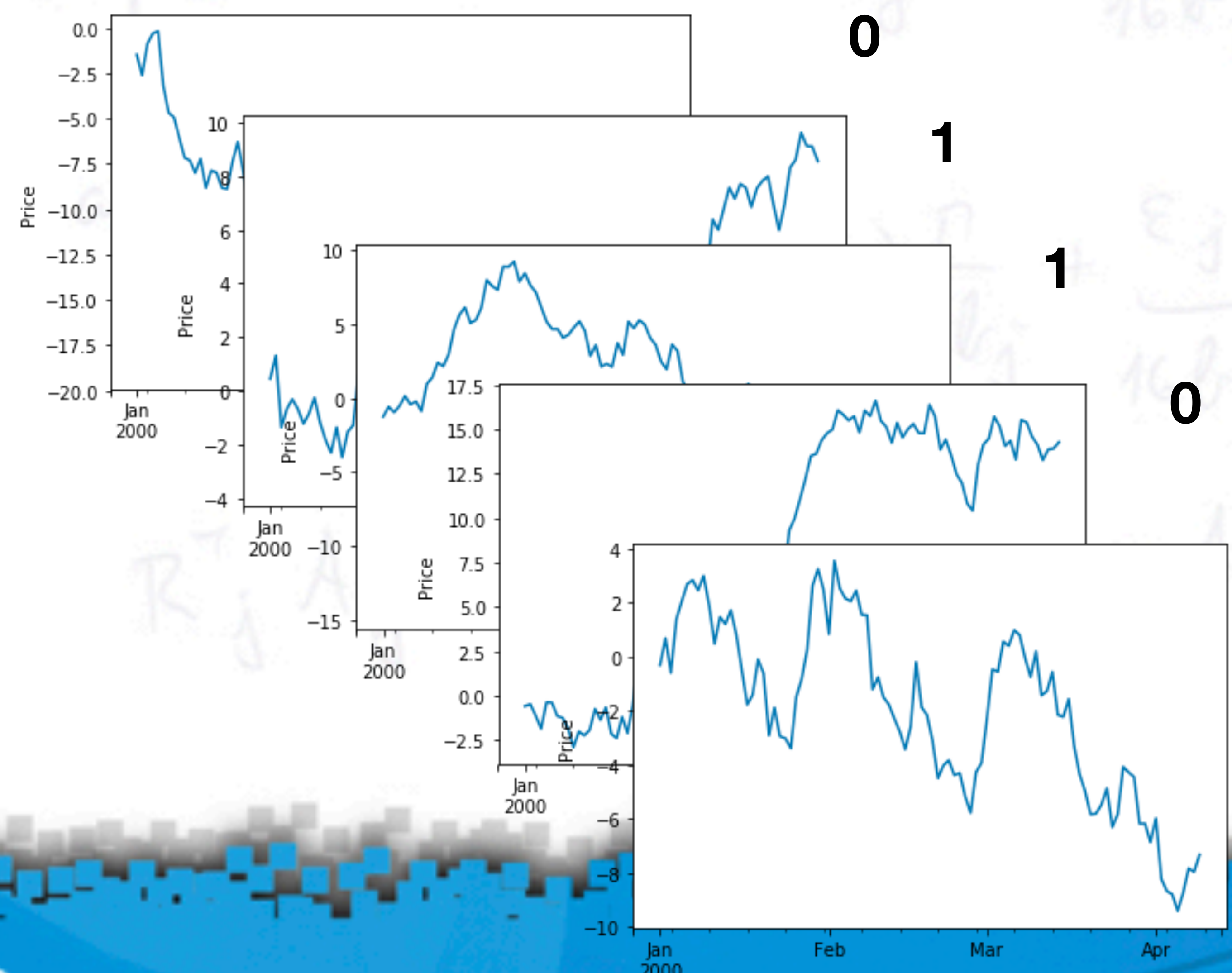
1

1

0

KNN - TS classification

Instance **Class**



0

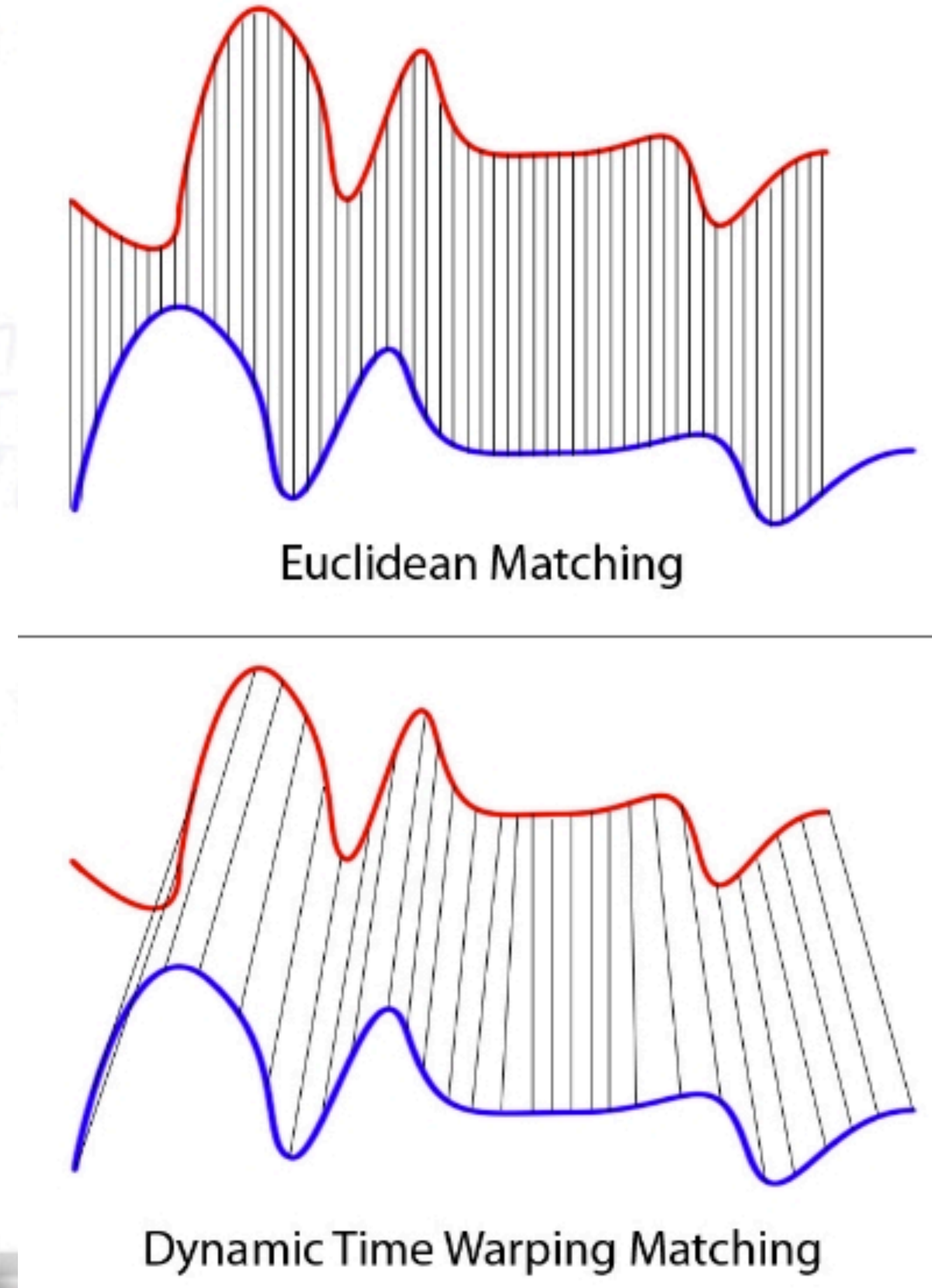
1

1

0

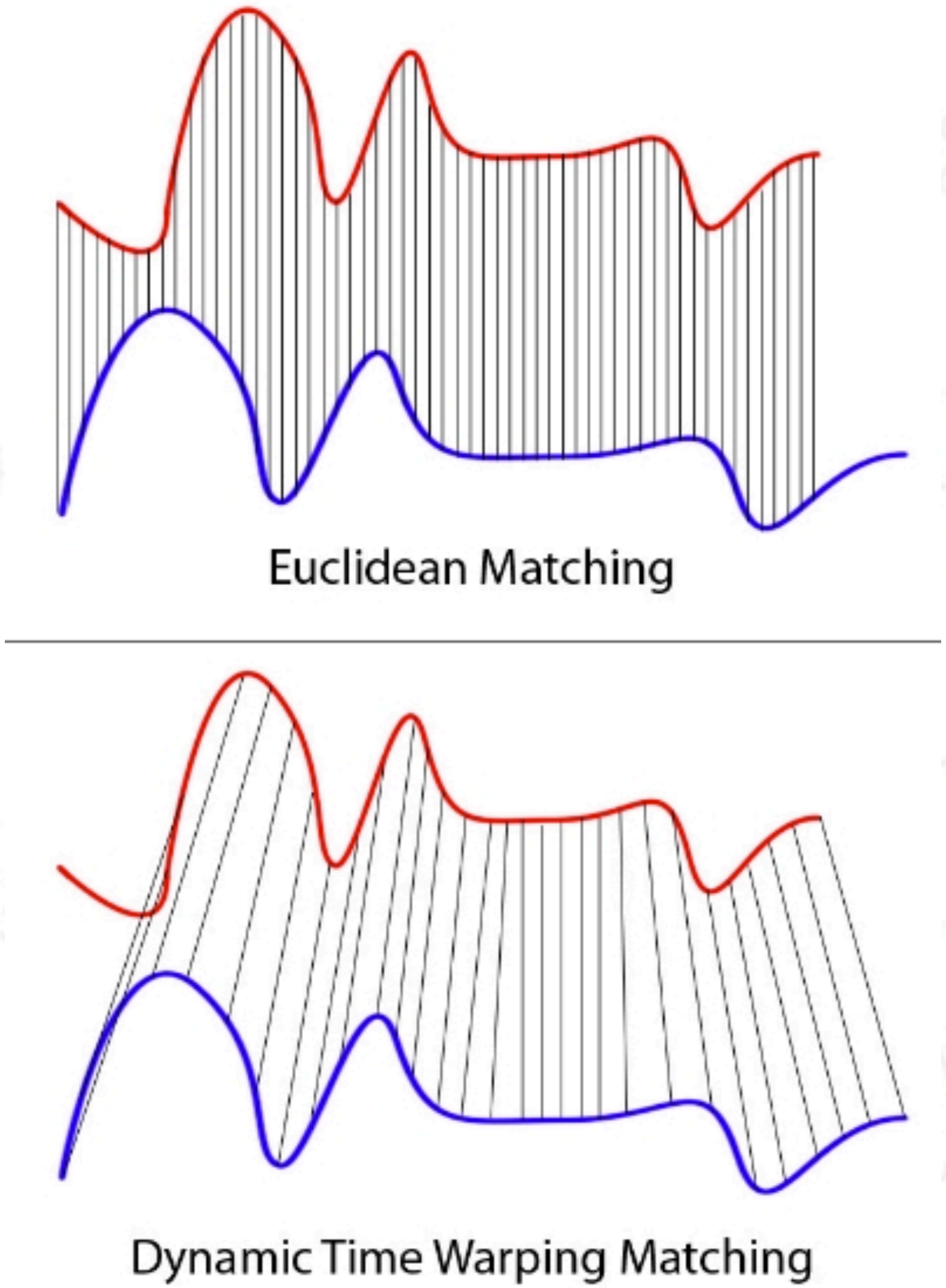
?

KNN - TS distances



KNN for time series

Dynamic time warping, DTW, a distance for time series



KNN

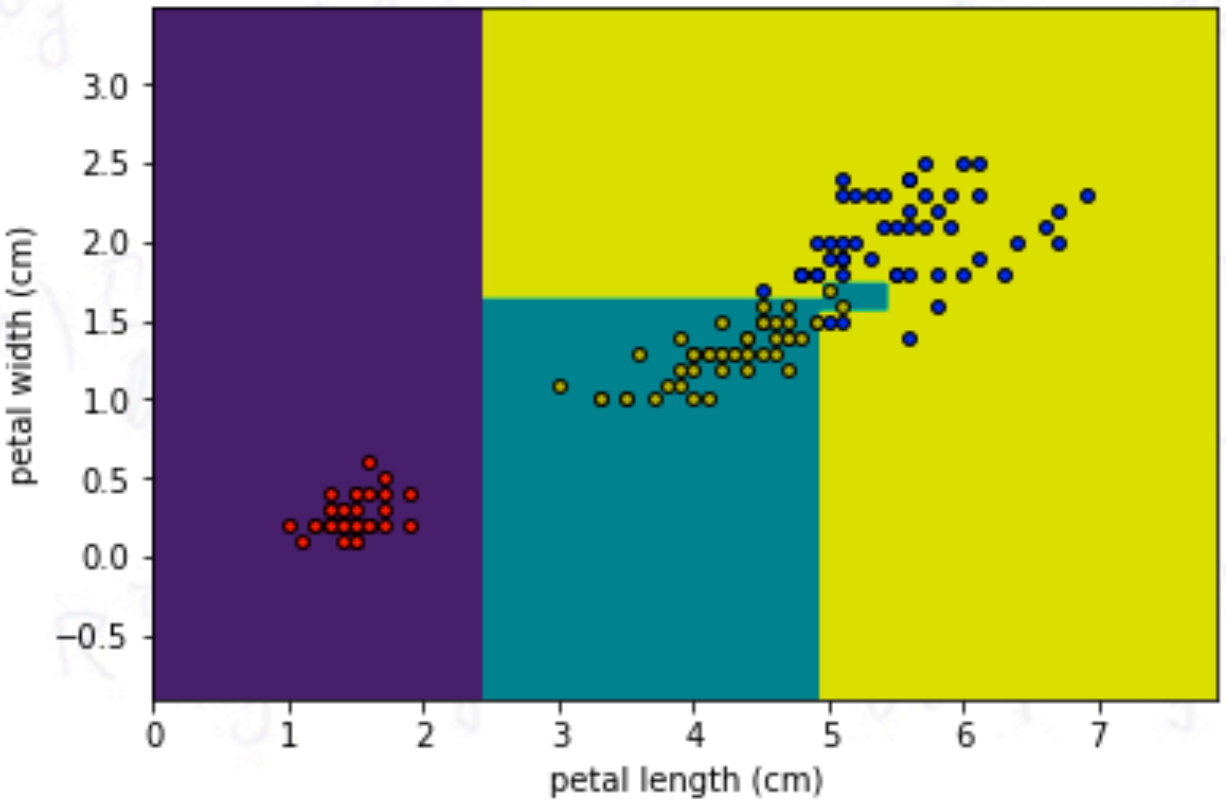
- Easy to understand and implement
- Computationally efficient in general
- Defining *similarities*
- The first thing that you should try when approaching a ML problem
- The curse of dimensionality: Nearest neighbours tend to be far away in high dimensions

Outline

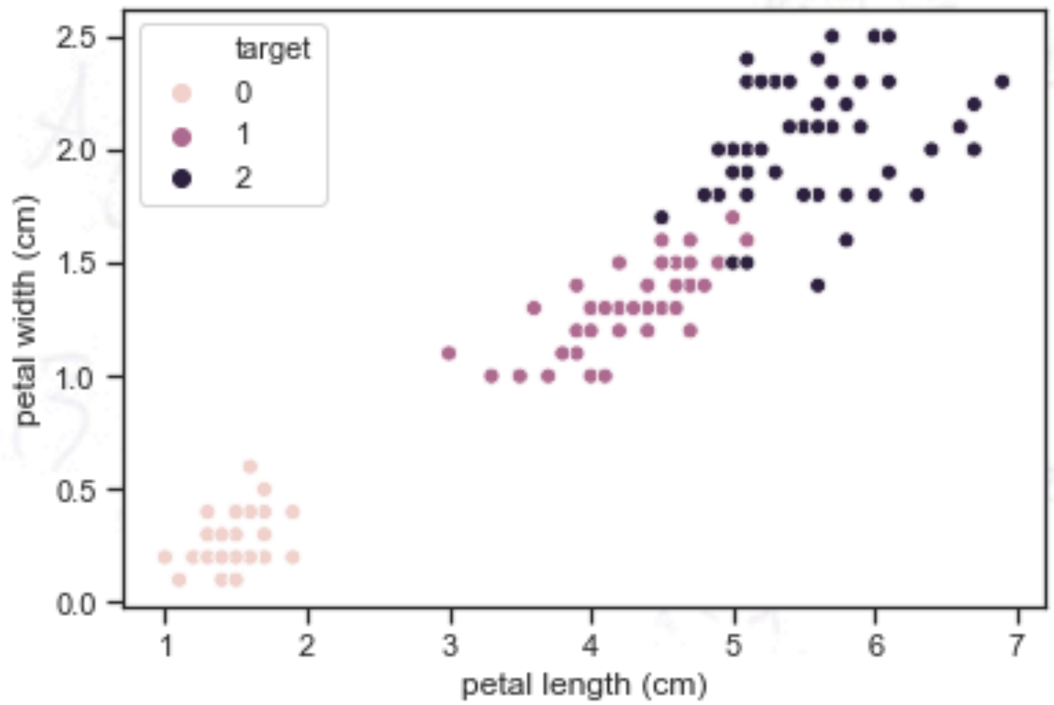
- K Nearest Neighbour, KNN
- **Decision trees, bagging, random forests, boosting**
- Logistic regression
- Discriminant analysis: LDA, QDA, GaussianNB
- Support Vector Machines, SVM

Decision trees

- Divide the feature space into high dimensional rectangles
- Find the boxes that minimise the classification error
- Infeasible!

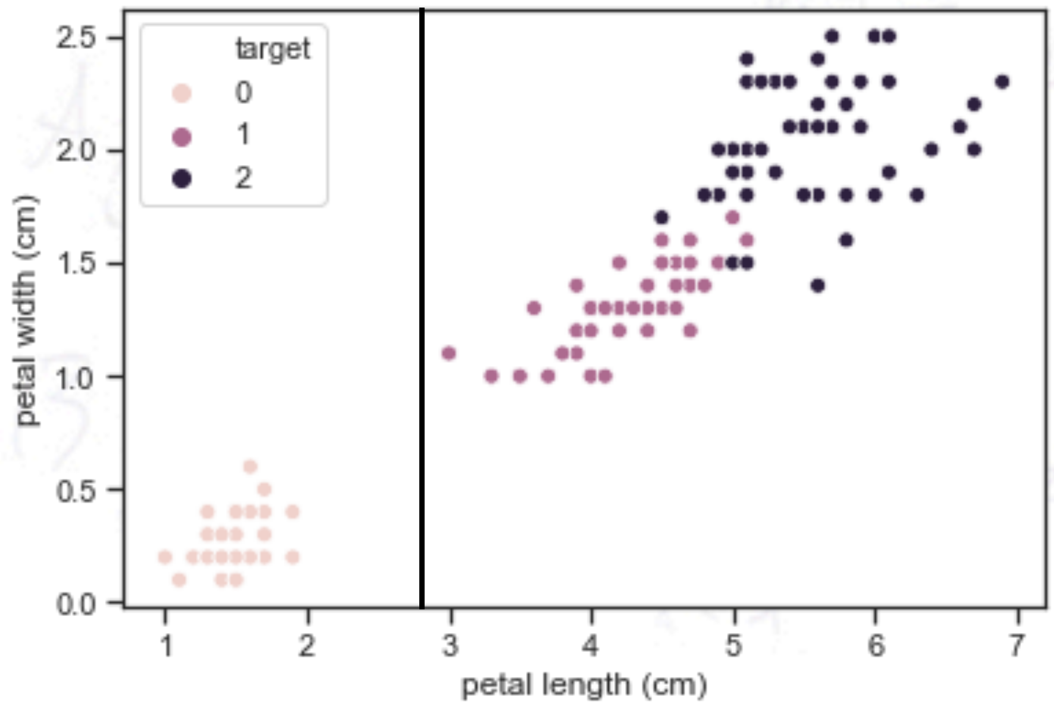


Decision trees



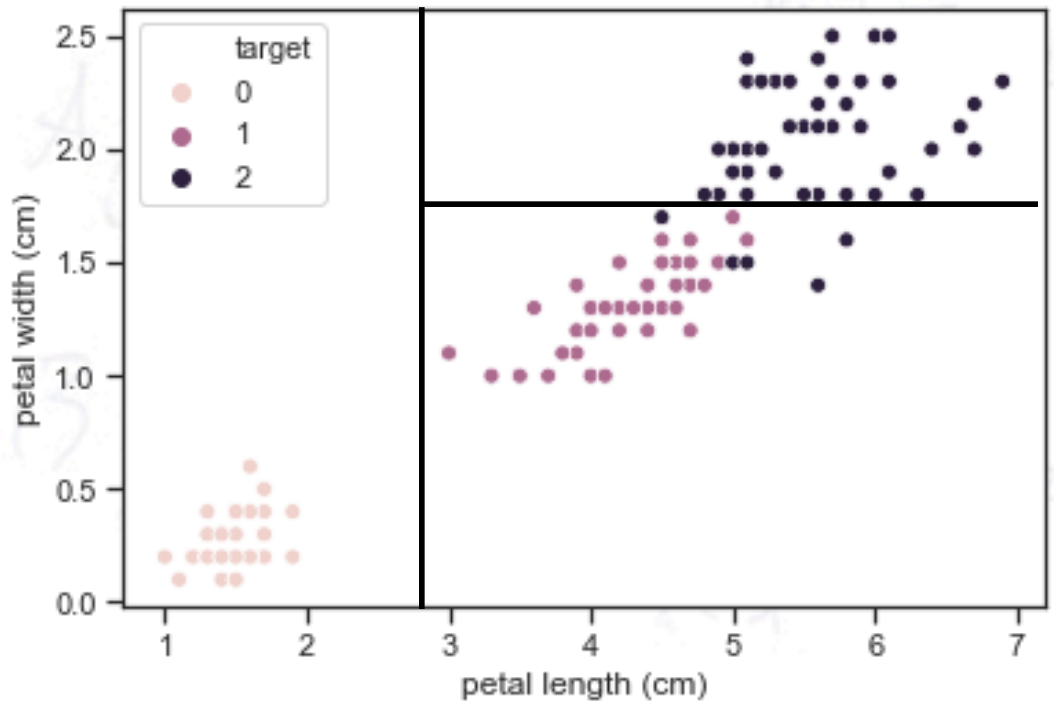
- Recursive binary splitting: iteratively partition the feature space.
- Error measured as $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$ where $\hat{p}_{mk}$ is the proportion of observations in the kth class in the mth region

Decision trees



- Recursive binary splitting: iteratively partition the feature space.
- Error measured as
  $$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \text{ where } \hat{p}_{mk}$$
  is the proportion of observations in the kth class in the mth region

Decision trees



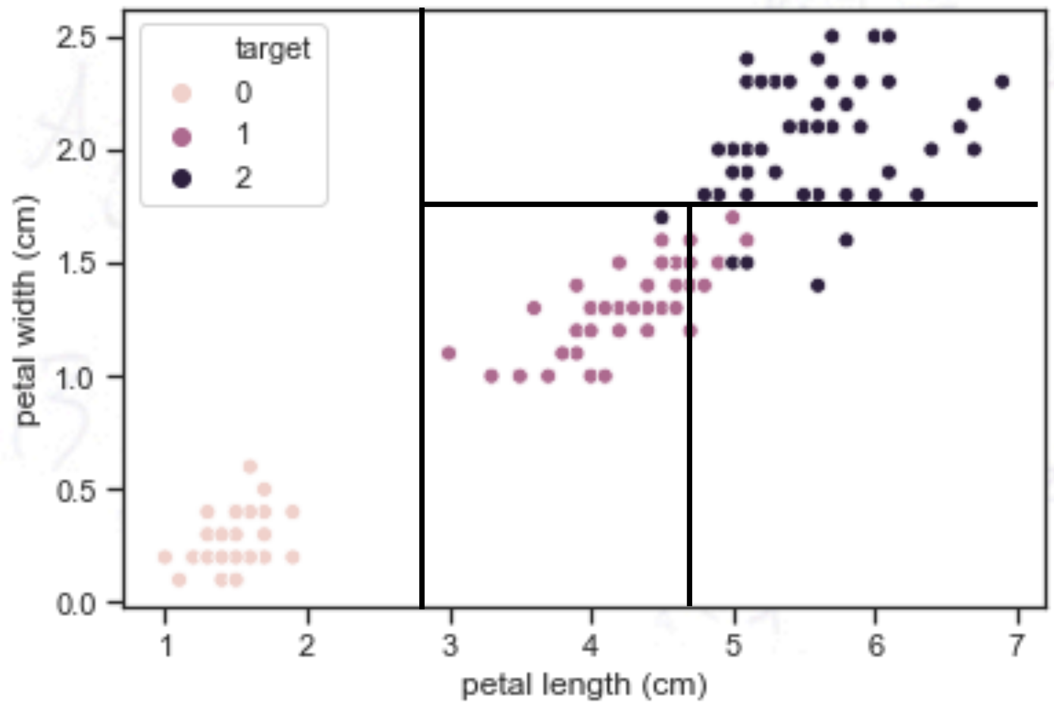
- Recursive binary splitting: iteratively partition the feature space.

- Error measured as

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \text{ where } \hat{p}_{mk}$$

is the proportion of observations in the kth class in the mth region

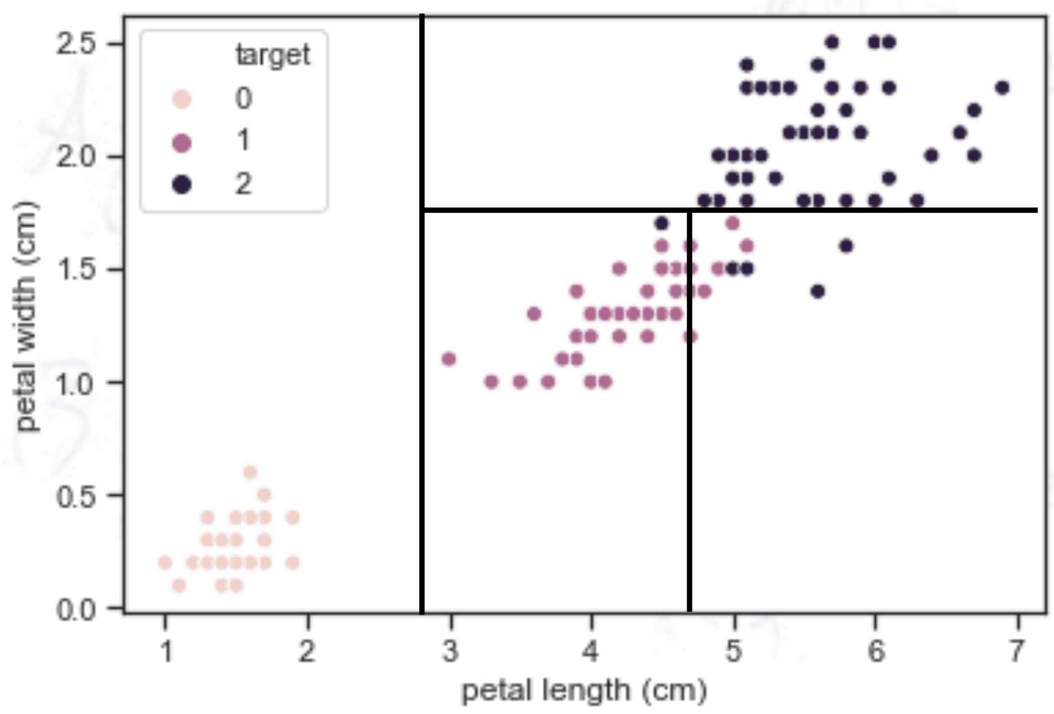
Decision trees

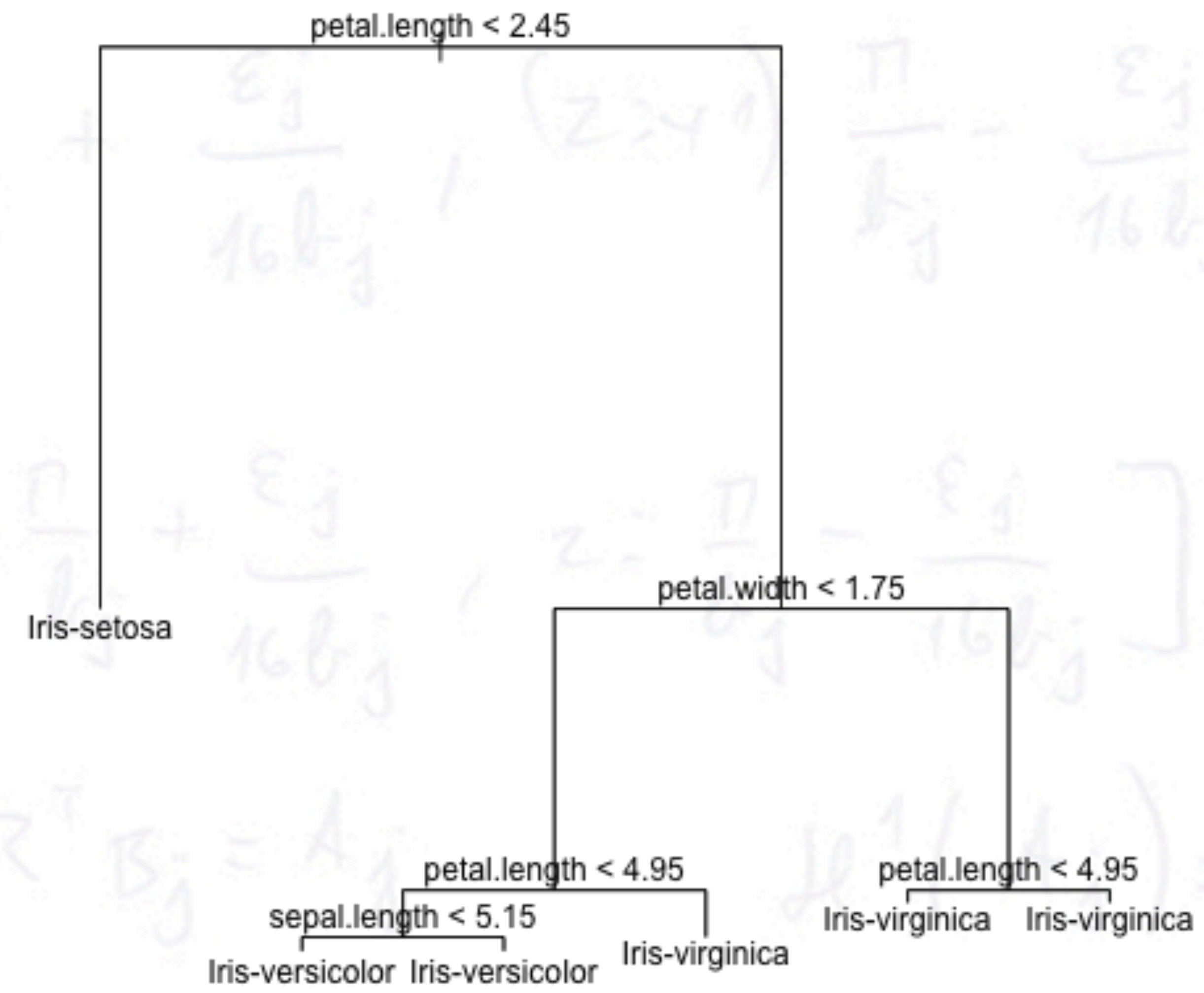


- Recursive binary splitting: iteratively partition the feature space.

- Error measured as
$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \text{ where } \hat{p}_{mk}$$
is the proportion of observations in the kth class in the mth region

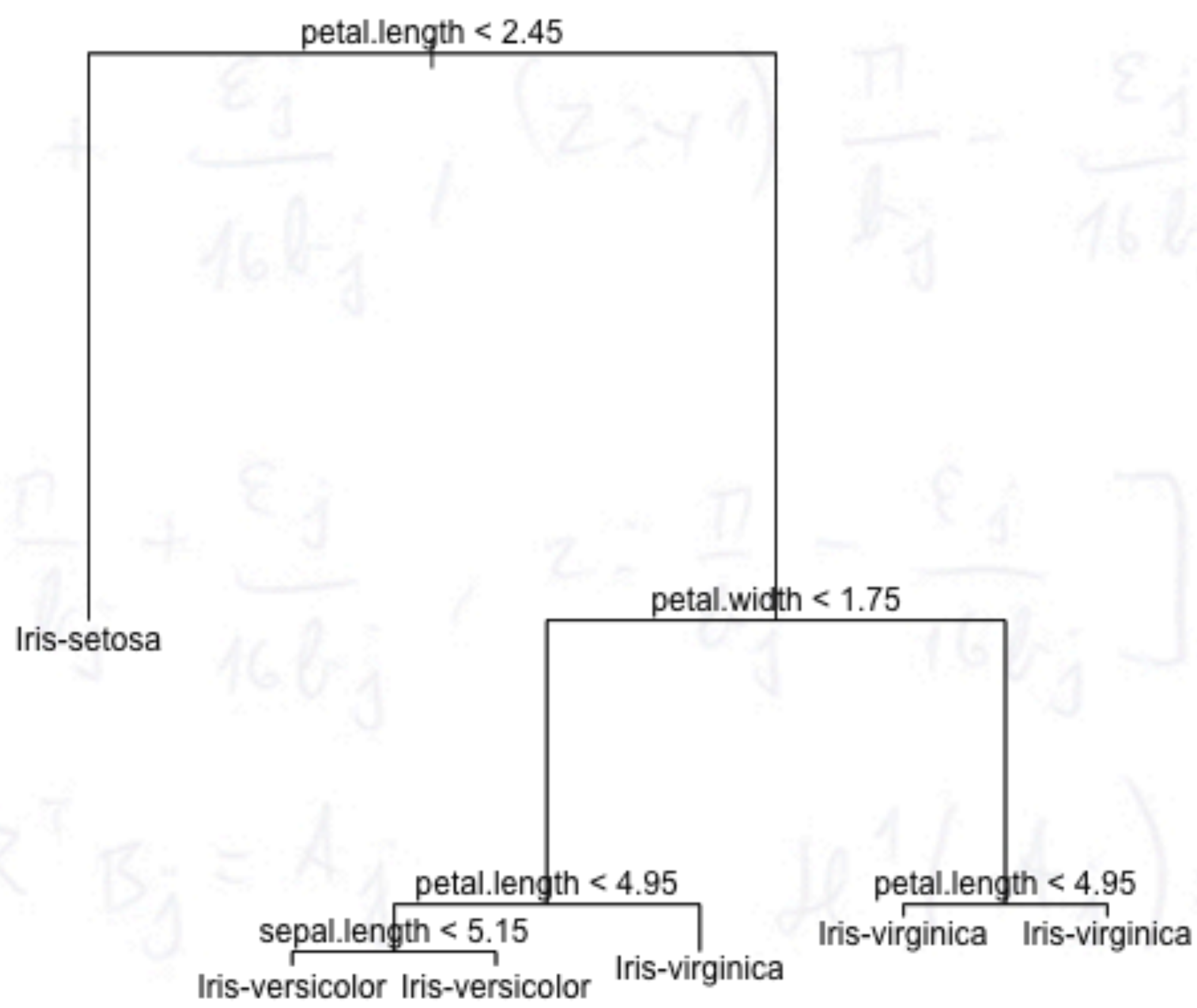
Decision trees





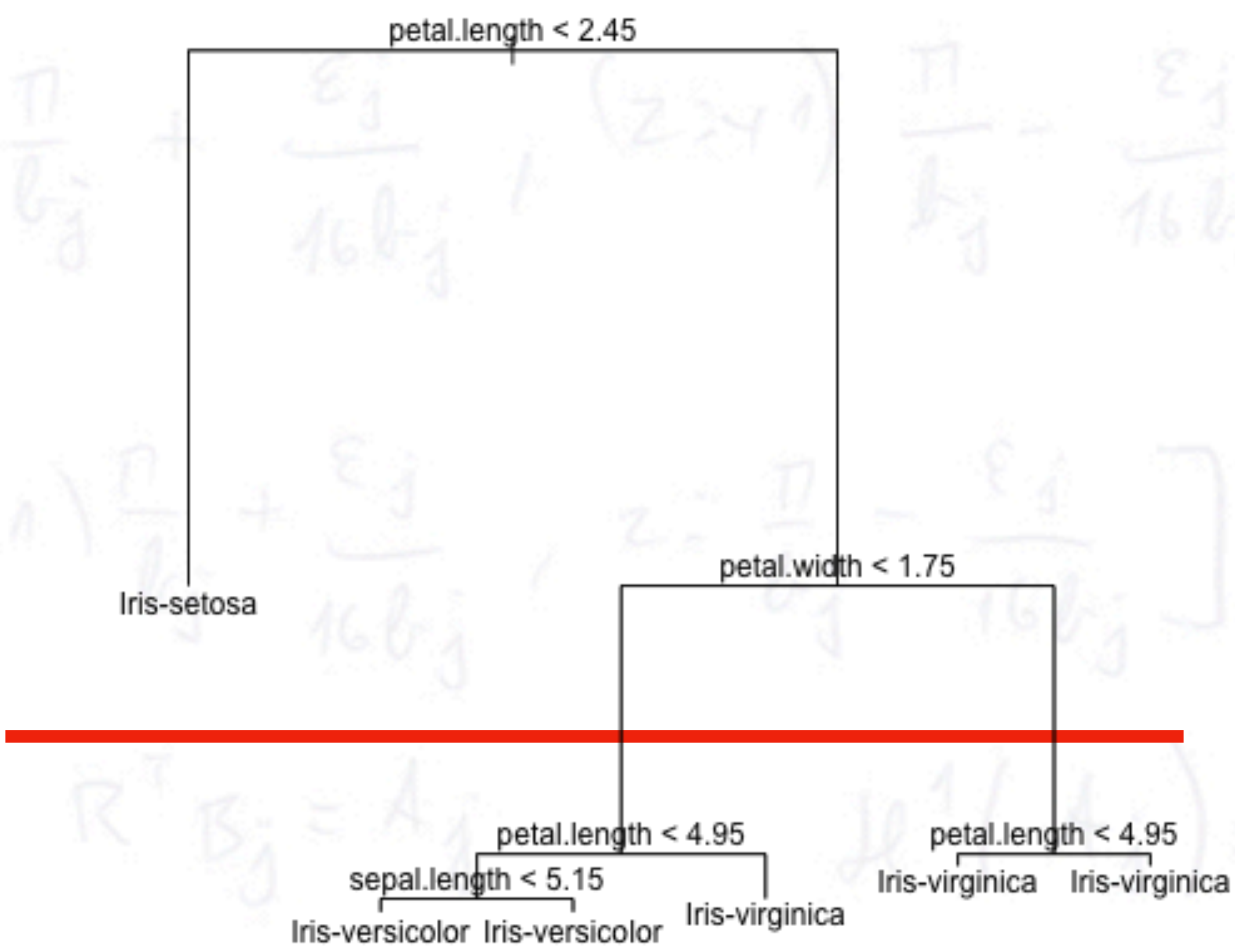
Decision trees

- Deep trees lead to high variance and overfitting
- When do we stop growing the tree?
    - iterate until each region contains no more than *k* samples
    - prune the tree, regularization



Decision trees

- Deep trees lead to high variance and overfitting
- When do we stop growing the tree?
  - iterate until each region contains no more than *k* samples
  - prune the tree, regularization

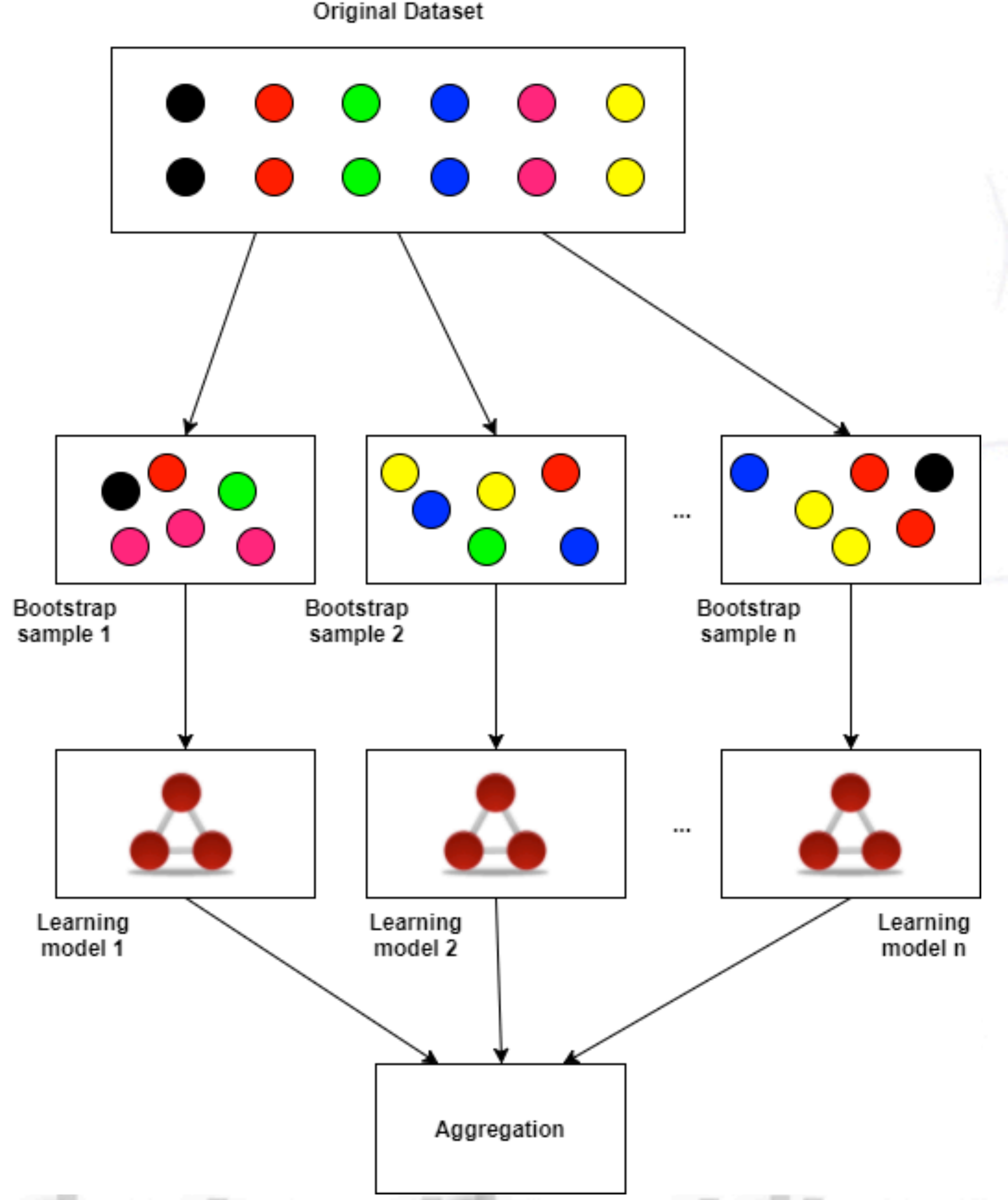


Decision trees

- Simple and interpretable. It is easy to understand by an expert
- Not competitive with modern algorithms
- High variance
- Can be a building block for other techniques

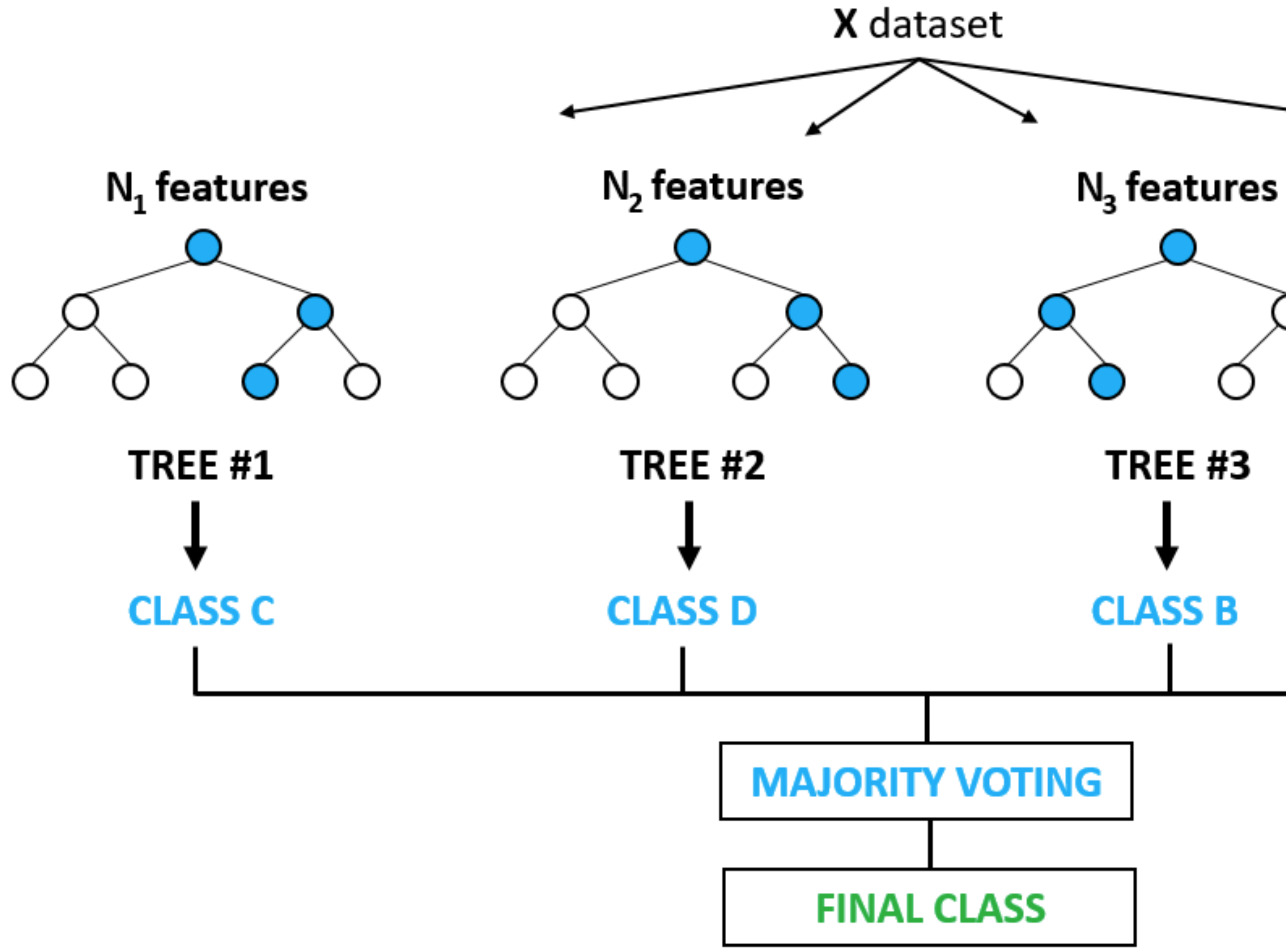
Bagging

- Bootstrap sample
- Grow a full tree
- Aggregate all results
- The mean decrease of Gini index by split for a given feature gives the variable importance



Random forests

- Bootstrap sample
- Select a subset of the features for each split
- Aggregate the results



Outline

- K Nearest Neighbour, KNN
- Decision trees, bagging, random forests, boosting
- **Logistic regression**
- Discriminant analysis: LDA, QDA, GaussianNB
- Support Vector Machines, SVM

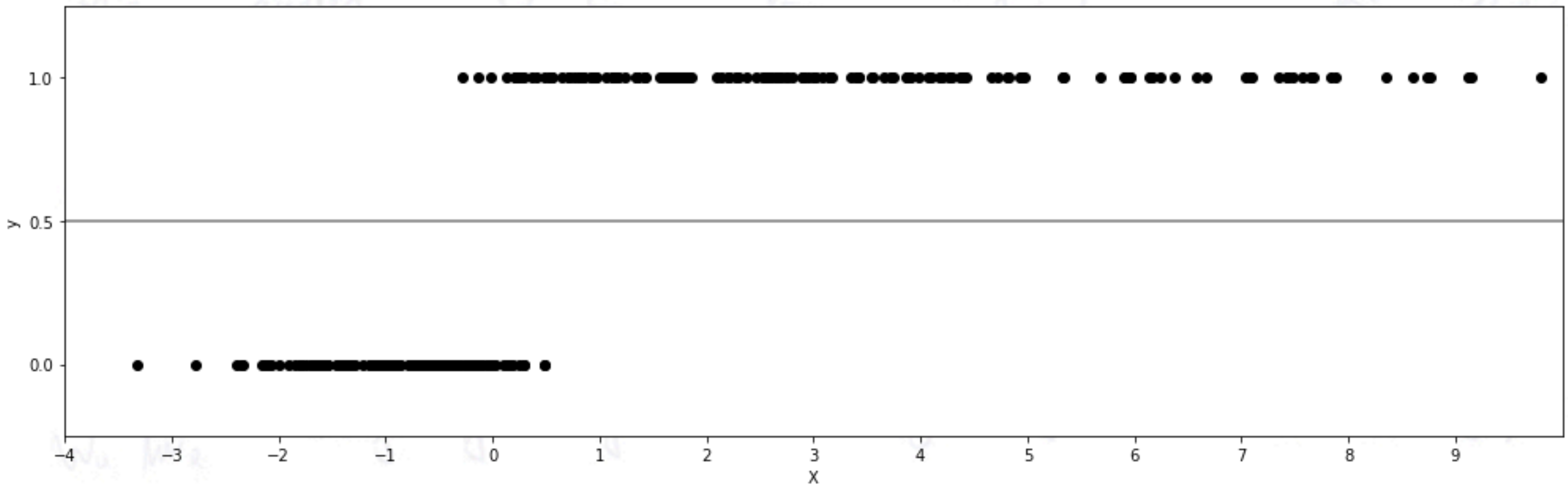
Logistic regression

- Binary class variable, Y
- Example in finance: X is the balance of a client and Y client's default
- Different approach from previous chapter: model the relation between X, the balance, and the probability of default of this client $p(X) = Pr(Y = 1 | X)$

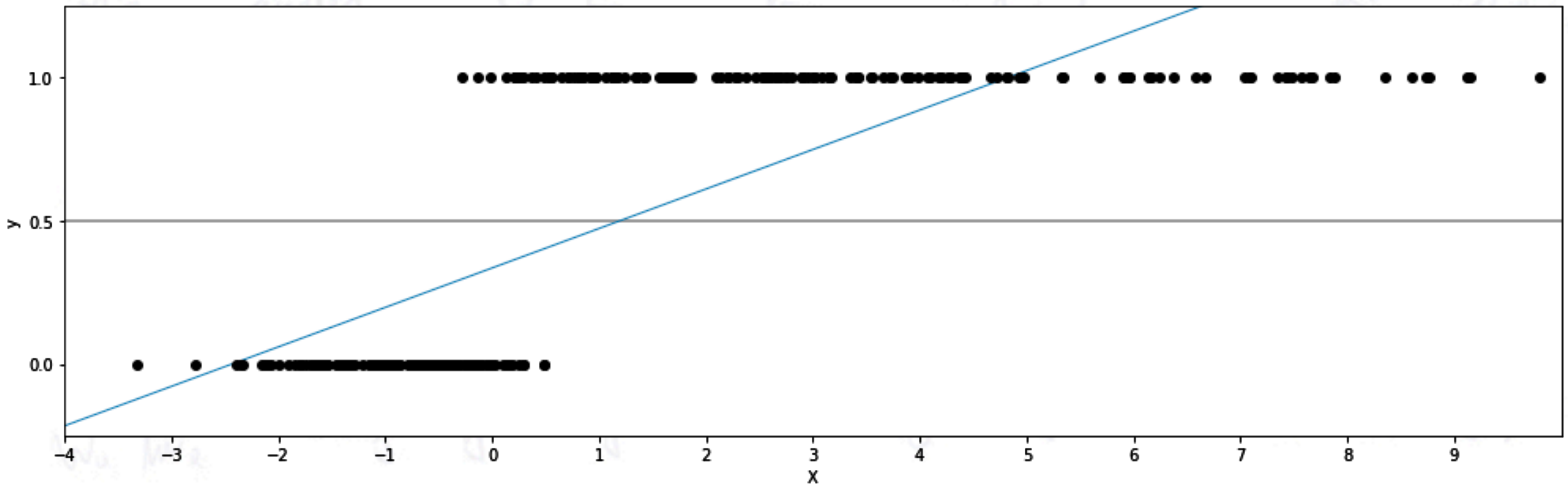
Logistic regression

- First approach is to use linear regression: $Y = \beta_0 + \beta_1 X$
  - LR is a very powerful solution in many context
  - Easy to fit
  - Interpretable

Logistic regression

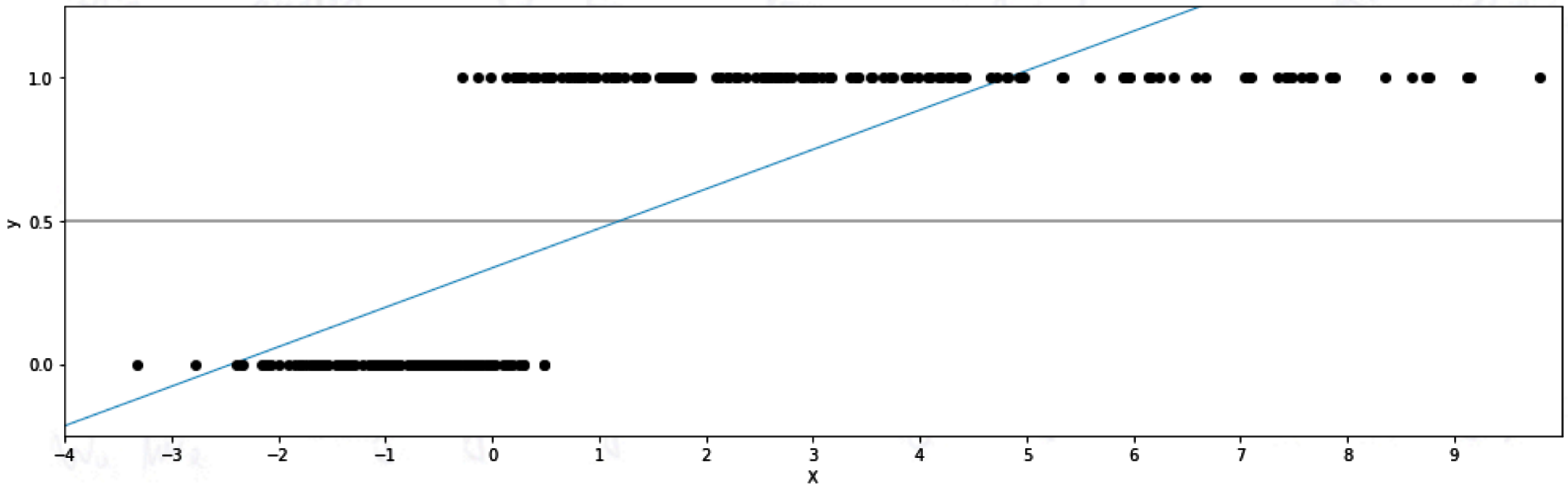


Logistic regression



Logistic regression

- Change the domain from $(-\infty, \infty)$ to $[0,1]$



Logistic regression

- One approach is to use the logistic function

$$p(X) = \frac{exp(\beta_0 + \beta_1 X)}{1 + exp(\beta_0 + \beta_1 X)}$$

- This is equivalen to $log \frac{p(X)}{1 - p(X)} = \beta_0 + \beta_1 X$

Logistic regression

- One approach is to use the logistic function

$$p(X) = \frac{exp(\beta_0 + \beta_1 X)}{1 + exp(\beta_0 + \beta_1 X)}$$

- This is equivalen to $log \frac{p(X)}{1 - p(X)} = \beta_0 + \beta_1 X$

Logistic regression

- One approach is to use the logistic function

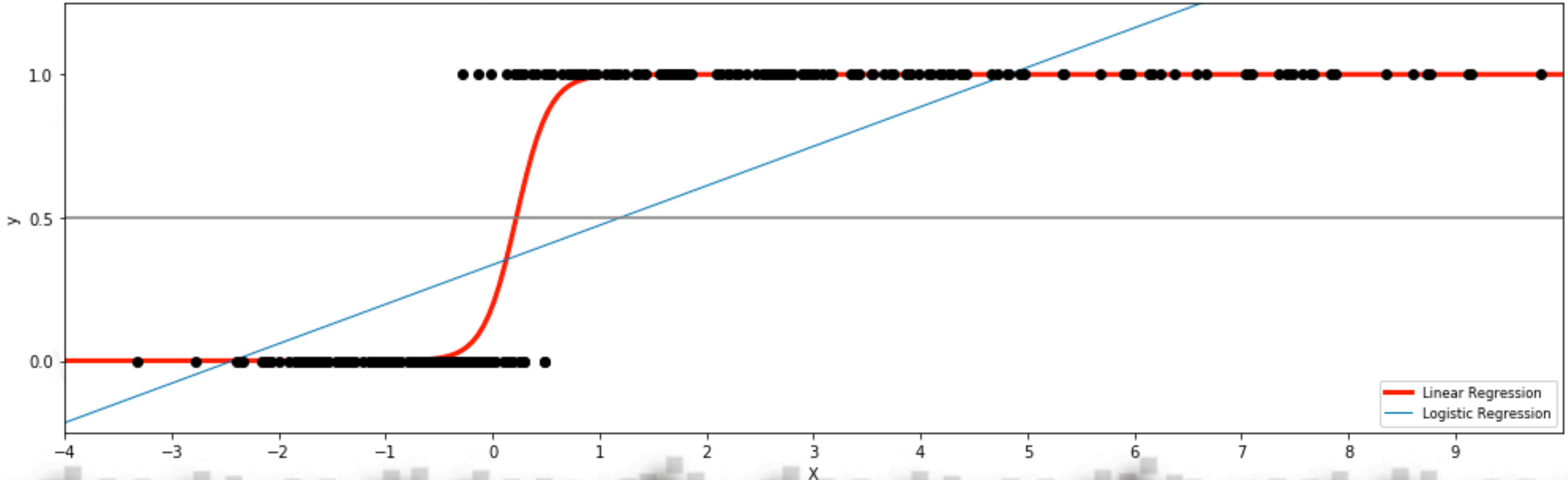
$$p(X) = \frac{exp(\beta_0 + \beta_1 X)}{1 + exp(\beta_0 + \beta_1 X)}$$

The logit is linear in *X*

- This is equivalen to $log \frac{p(X)}{1 - p(X)} = \beta_0 + \beta_1 X$

Logistic regression

- Making predictions is easy
- Maximum likelihood estimation of the coefficients efficient



Logistic regression

- When the classes are well separated it can be unstable: if there is a feature that separates clases perfectly the coefficients go up to infinity
- If the sample is small discriminant analysis is more accurate

Outline

- K Nearest Neighbour, KNN
- Decision trees, bagging, random forests, boosting
- Logistic regression
- **Discriminant analysis: LDA, QDA, GaussianNB**
- Support Vector Machines, SVM

Discriminant analysis

- $P(Y = k)$
- $P(X = x \mid Y = y)$: assume that they follow the Gaussian distribution with the same variance in each class (Linear Discriminant Analysis)
- $P(Y = k \mid X = x)$

X	Y
3	1
5	0
4	0
7	1
3	1
...	...
...	...
8	0
9	1
4	0
8	?

Discriminant analysis

- Model the distribution of X and use the Bayes theorem to obtain $P(Y|X)$:

$$P(Y=k\,|\,X=x) = \frac{P(X=x\,|\,Y=y)P(Y=k)}{P(X=x)} = \frac{\pi_k f_k(x)}{\sum_{i\leq k}\pi_i f_i(x)}$$

- Assign x to the class with maximum $P(Y|X)$
- We do not need the denominator
- Particularly accurate when the classes are normal

Discriminant analysis

- Model the distribution of X and use the Bayes theorem to obtain $P(Y|X)$:

density

$$P(Y=k|X=x) = \frac{P(X=x|Y=y)P(Y=k)}{P(X=x)} = \frac{\pi_k f_k(x)}{\sum_{i\leq k} \pi_i f_i(x)}$$

- Assign x to the class with maximum $P(Y|X)$

- We do not need the denominator

- Particularly accurate when the classes are normal

Discriminant analysis

- Model the distribution of X and use the Bayes theorem to obtain $P(Y|X)$:

prior **density**

$$P(Y=k|X=x) = \frac{P(X=x|Y=y)P(Y=k)}{P(X=x)} = \frac{\pi_k f_k(x)}{\sum_{i \leq k} \pi_i f_i(x)}$$

- Assign x to the class with maximum $P(Y|X)$

- We do not need the denominator

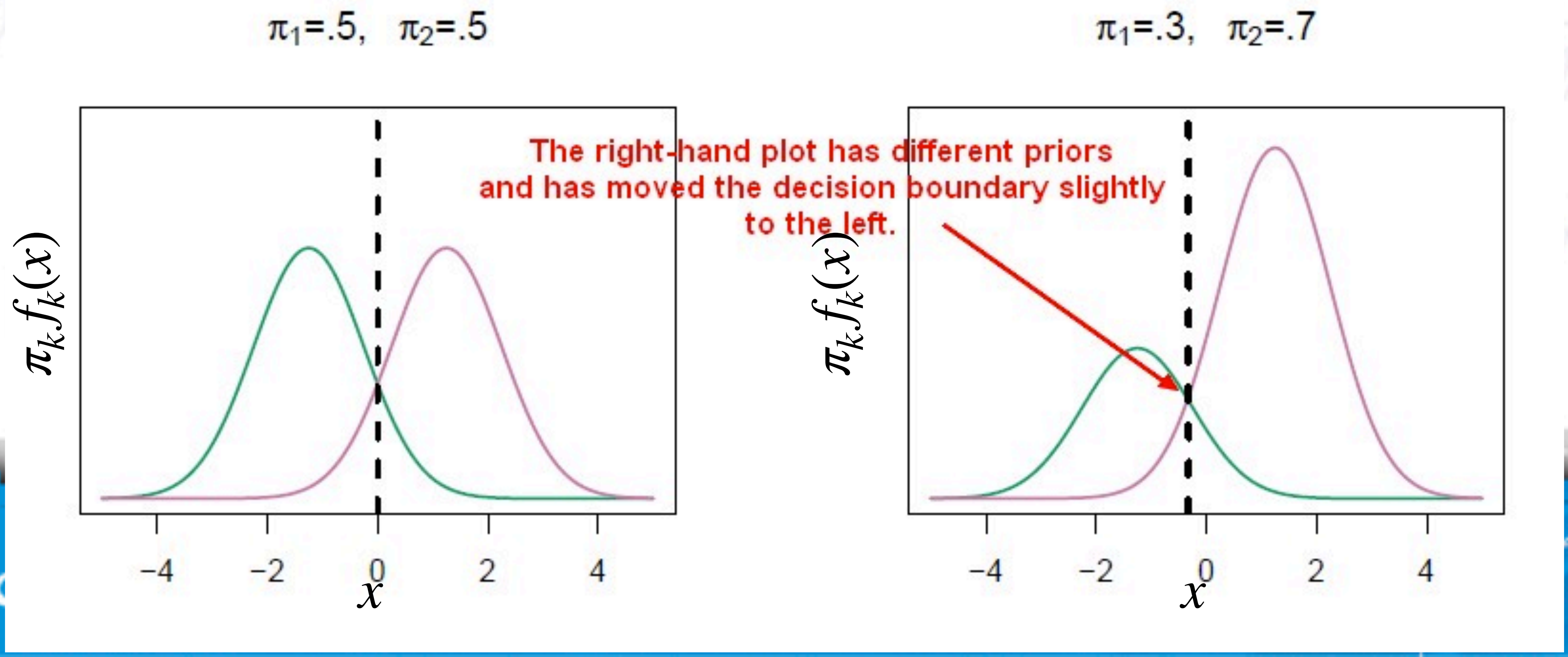
- Particularly accurate when the classes are normal

Discriminant analysis

- Model the distribution of X and use the Bayes theorem to obtain $P(Y|X)$:

$$P(Y=k|X=x) = \frac{P(X=x|Y=y)P(Y=k)}{P(X=x)} = \frac{\overbrace{\pi_k}^{\text{prior}} \overbrace{f_k(x)}^{\text{density}}}{\sum_{i\le k} \pi_i f_i(x)}$$

-
-
-



Linear Discriminant analysis

- If we assume that *f* is Gaussian, the Bayes classifier assign an observation to the class with maximum discriminant funciton value

$$\delta_k(x) = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(p_k)$$

- Linear function of x

- Learning from data implies estimating the means and the variances

Quadratic discriminant analysis

- $P(Y=k\,|\,X=x) = \dfrac{P(X=x\,|\,Y=y)P(Y=k)}{P(X=x)} = \dfrac{\pi_k f_k(x)}{\sum_{i\le k} \pi_i f_i(x)}$

- assume that the density follows a Gaussian distribution with different variance in each class

- Discriminant function $\delta_k(x)$ is quadratic on x

Quadratic discriminant analysis

- $$P(Y = k \mid X = x) = \frac{P(X = x \mid Y = y)P(Y = k)}{P(X = x)} = \frac{\pi_k f_k(x)}{\sum_{i \leq k} \pi_i f_i(x)}$$
- assume that the density follows a Gaussian distribution with different variance in each class
- Discriminant function $\delta_k(x)$ is quadratic on x

Quadratic discriminant analysis

- $P(Y=k \mid X=x) = \dfrac{P(X=x \mid Y=y)P(Y=k)}{P(X=x)} = \dfrac{\pi_k f_k(x)}{\sum_{i \leq k} \pi_i f_i(x)}$

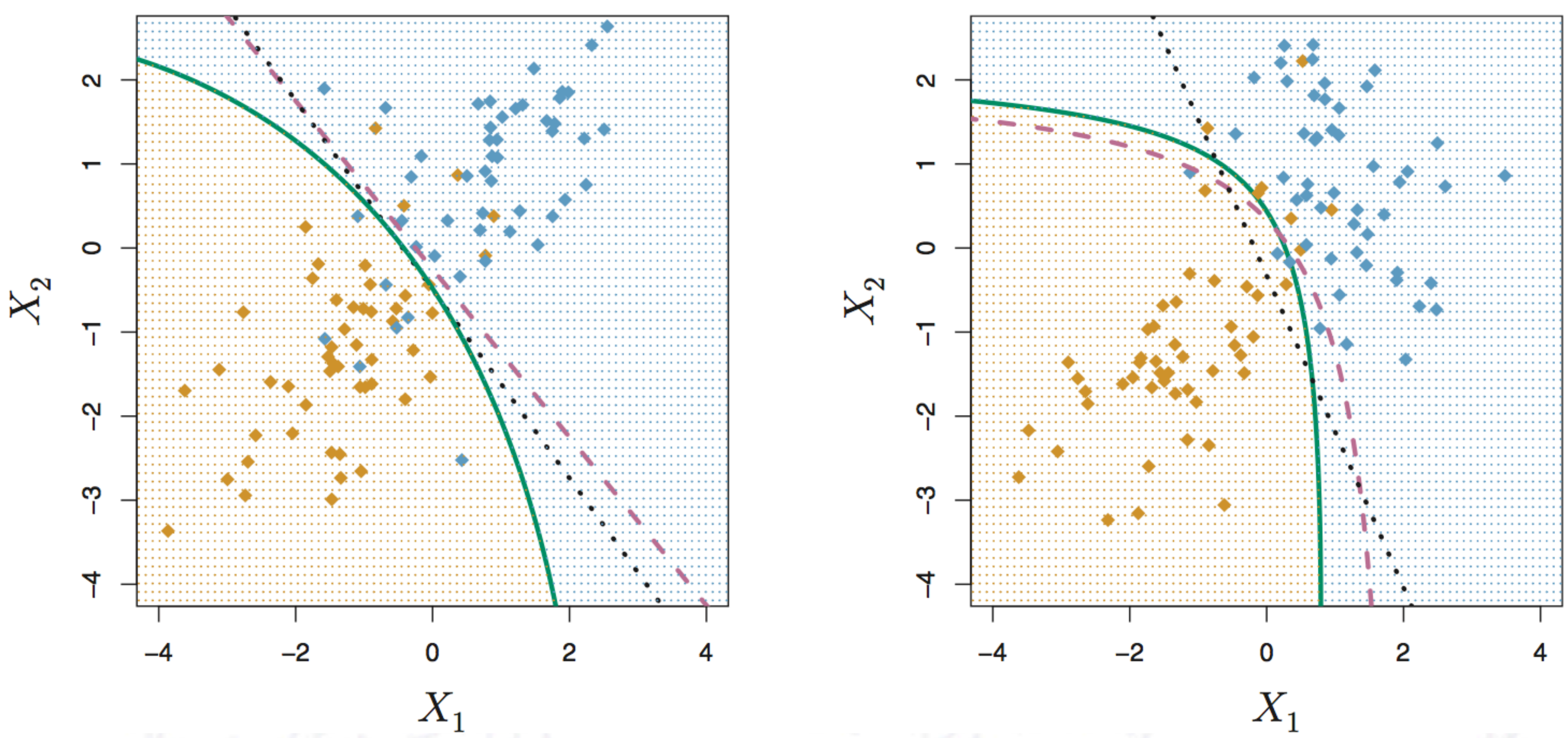
- assume that the density follows a Gaussian distribution with different variance in each class

- Discriminant function $\delta_k(x)$ is quadratic on x

Naive Bayes

- In each class the density factors into a product of densities, $f_k(x) = \prod_{j=1}^{p} f_{jk} x(j)$
- The assumptions are strong (conditionally independence in each of the classes) but for classification we are interested in finding the class for which the probability $P(Y = k \mid X = x) = \dfrac{P(X = x \mid Y = y) P(Y = k)}{P(X = x)}$ is maximized

LDA QDA



- Bayes, LDA and QDA decision boundaries

Outline

- K Nearest Neighbour, KNN
- Decision trees, bagging, random forests, boosting
- Logistic regression
- Discriminant analysis: LDA, QDA, GaussianNB
- **Support Vector Machines, SVM**

Separating hyperplanes

- Find decision boundaries that separate the data in different classes
- Two dimensional feature space in the example

Hyperplane

- A flat affine subspace of dimension *n-1*
- In a two dimensional space, a hyperplane is a line
- It has the form

$$\beta_0 + X_1\beta_1 + X_2\beta_2 + \ldots + X_n\beta_n = 0$$

- The vector ($\beta_1, \beta_2, \ldots, \beta_n$) is called a normal vector and goes from the origin in a direction orthogonal to the surface of the hyperplane

Hyperplane

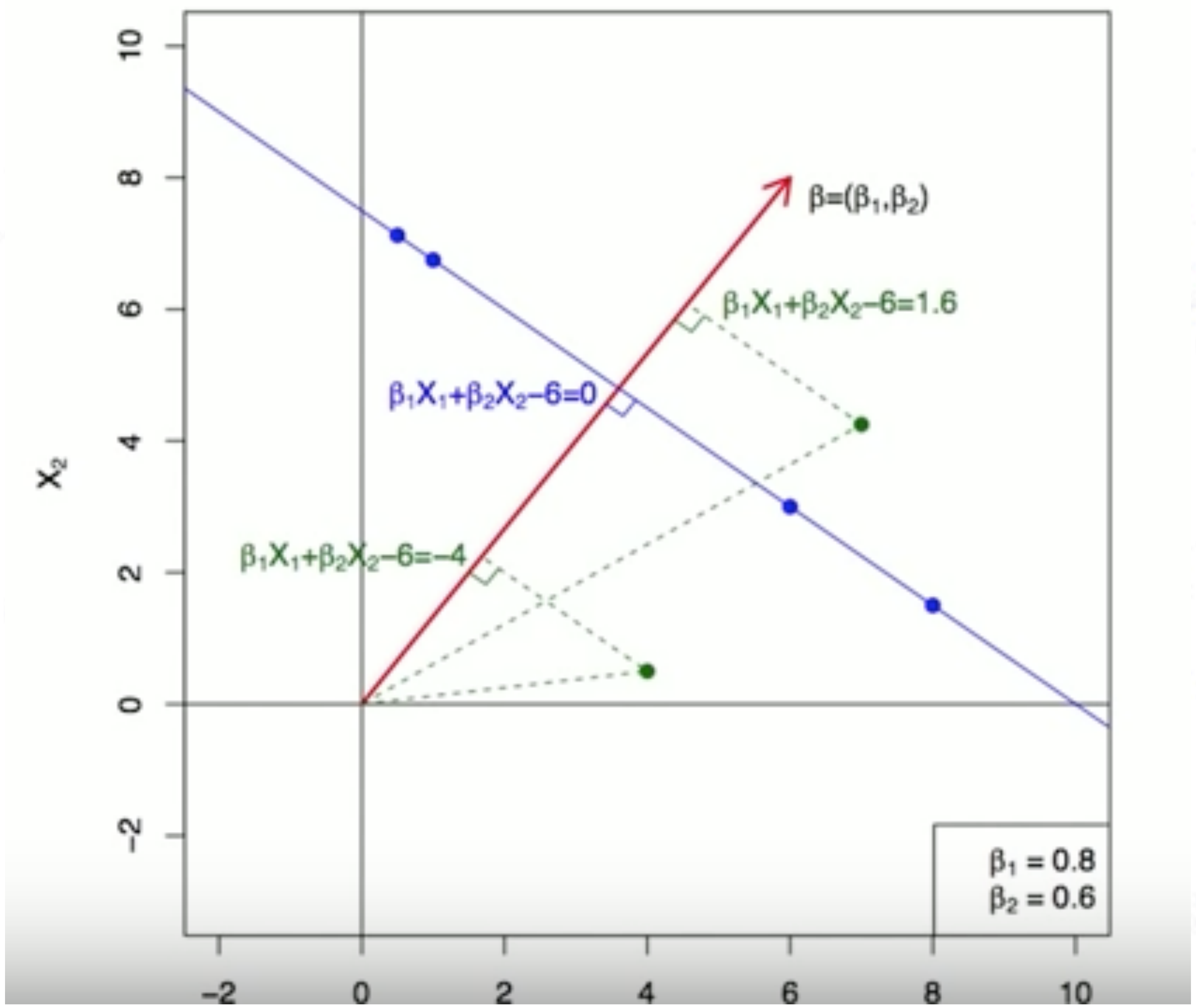
- Hyperplane, $\beta_0, \beta_1, \beta_2$

- Normal, β_1, β_2

    - $\beta_1^2 + \beta_2^2 = 1$

- Projections onto the normal and the distance to the hyperplane

- Can be positive, zero or negative



Hyperplane

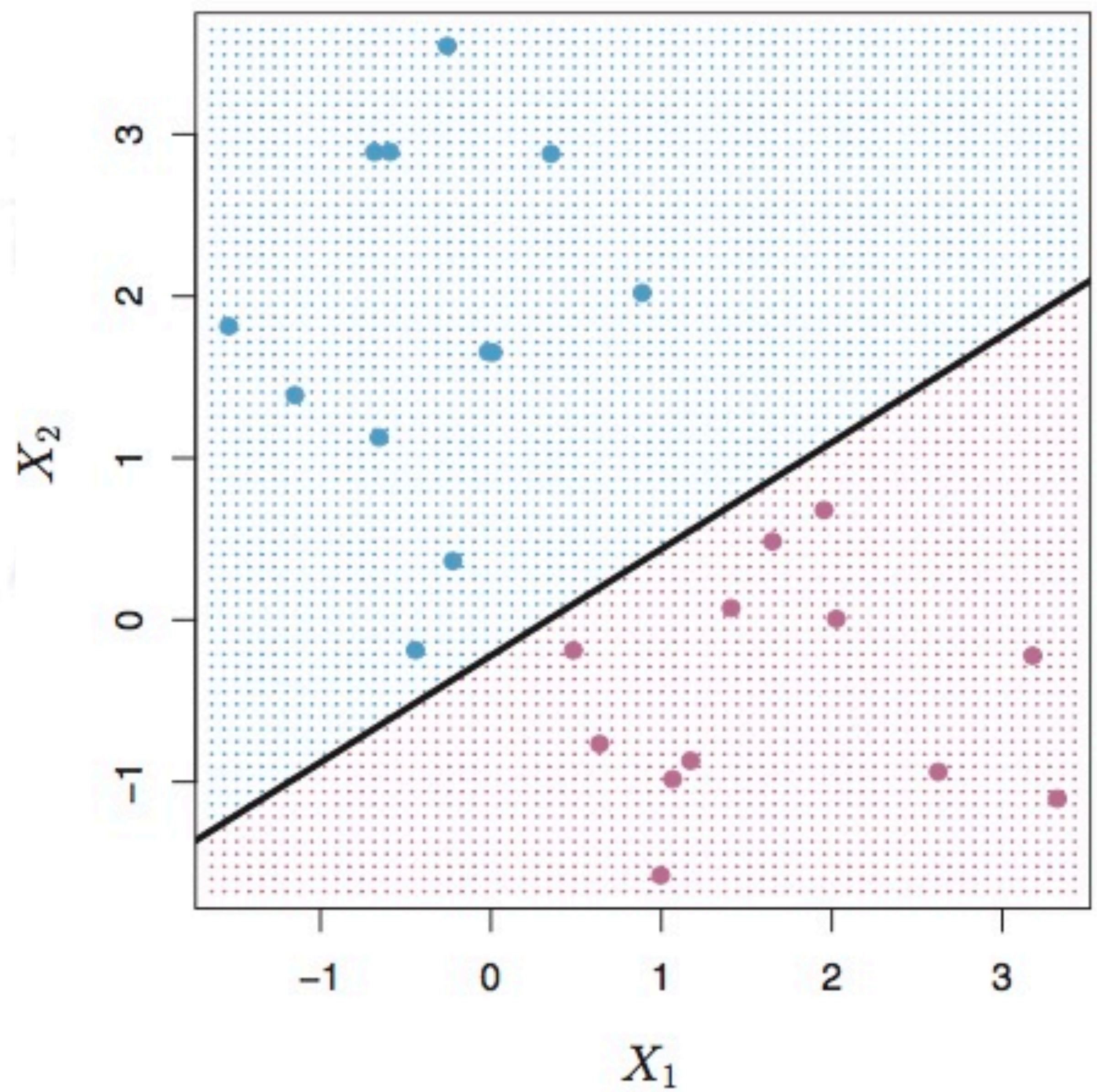
- For every blue point *i*

$$\beta_0 + \sum_j X_{ij}\beta_{ij} > 0$$

- For every purple point *i*

$$\beta_0 + \sum_j X_{ij}\beta_{ij} < 0$$

- Predictions are very easy
- Data has to be normalised



Hyperplane

- Assume that the class value of instance i is $y_i = \{-1, +1\}$

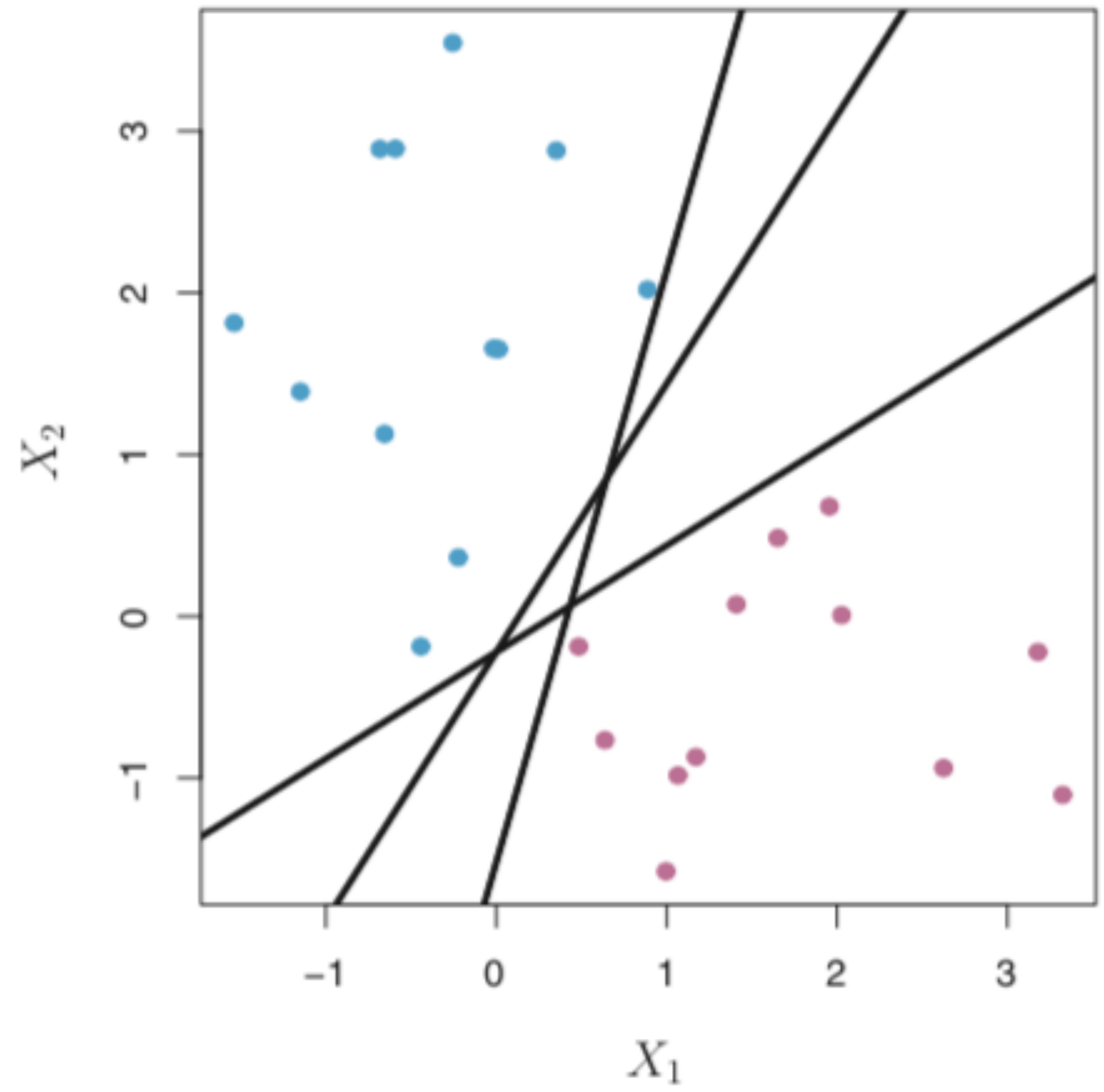
- Then for every separating hyperplane it holds that for every sample *i*

$$y_i * (\beta_0 + \sum_j X_{ij}\beta_{ij}) > 0$$

- This is the key to the learning algorithm

Max margin classifier

- Which separating Hyperplane do we choose?
- In order to reduce the variance we will prefer the hyperplane that is farther from the observations



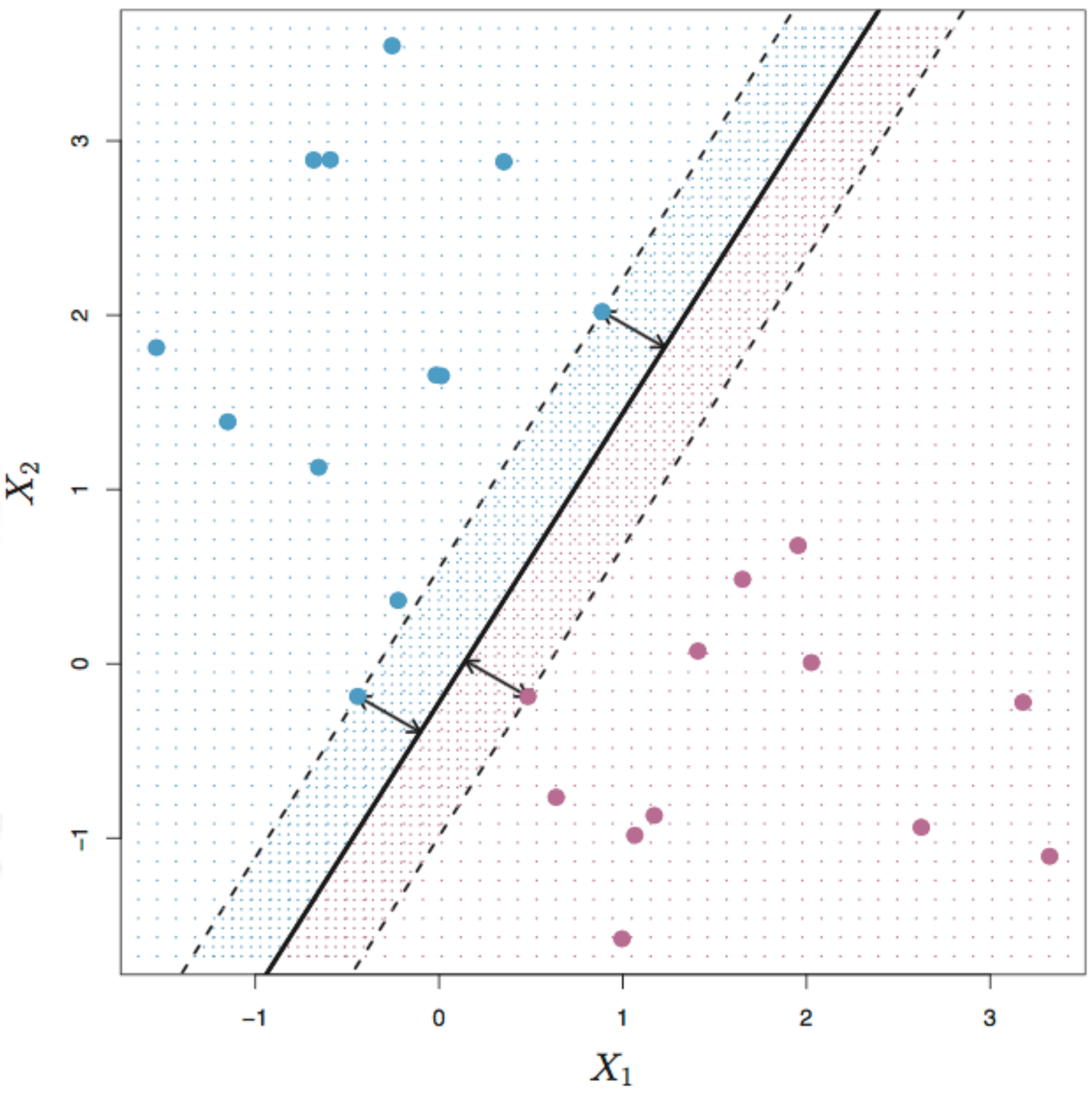
Max margin classifier

- The maximal margin classifier can be estimated as an optimisation problem

$$\arg \max_{\beta_0, \beta_1, \ldots, \beta_n} M$$

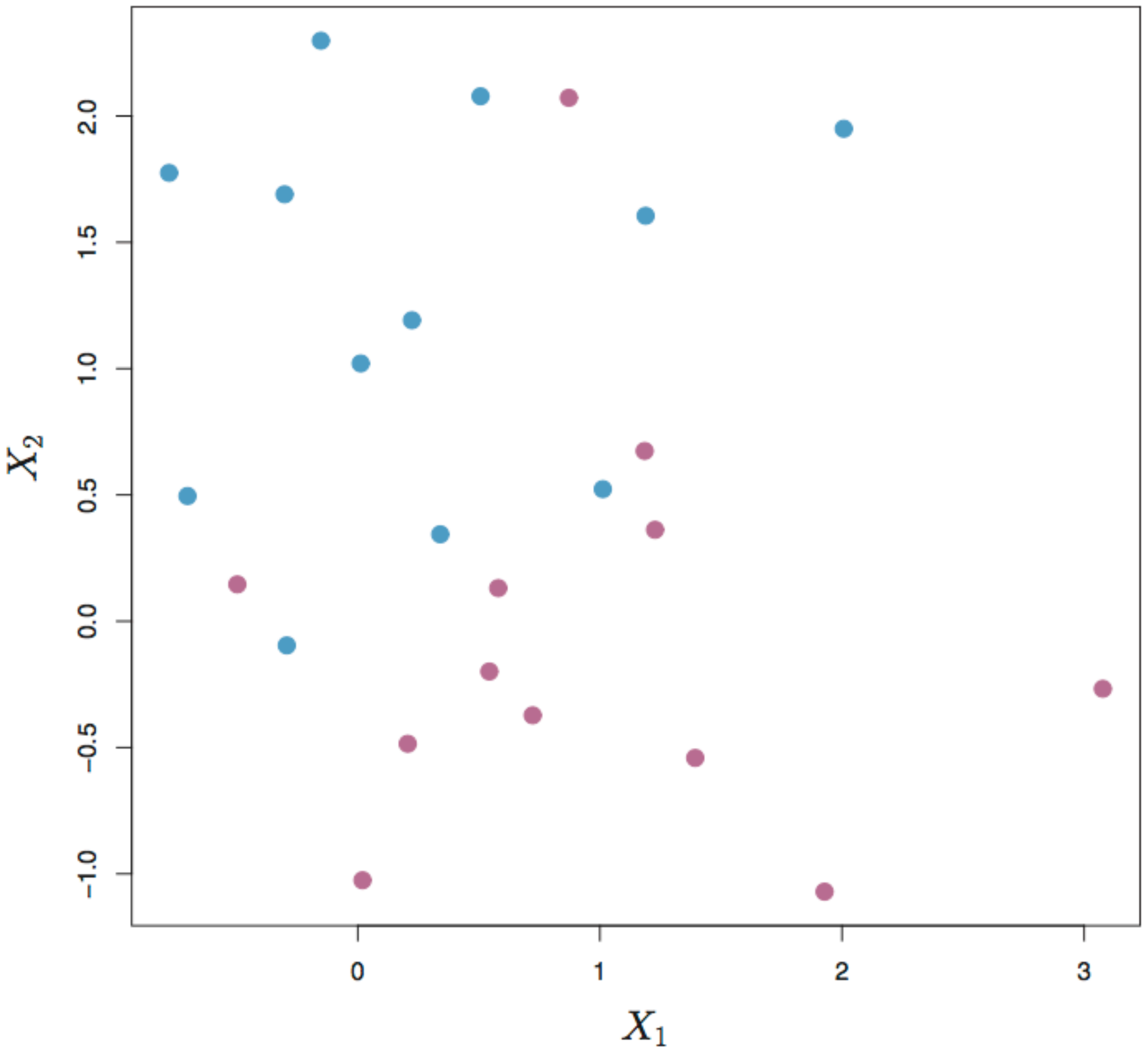
$$\text{subject to } \sum_{j=1}^{n} \beta_j^2 = 1,$$

$$y_i * (\beta_0 + \sum_j X_{ij}\beta_{ij}) \geq M$$



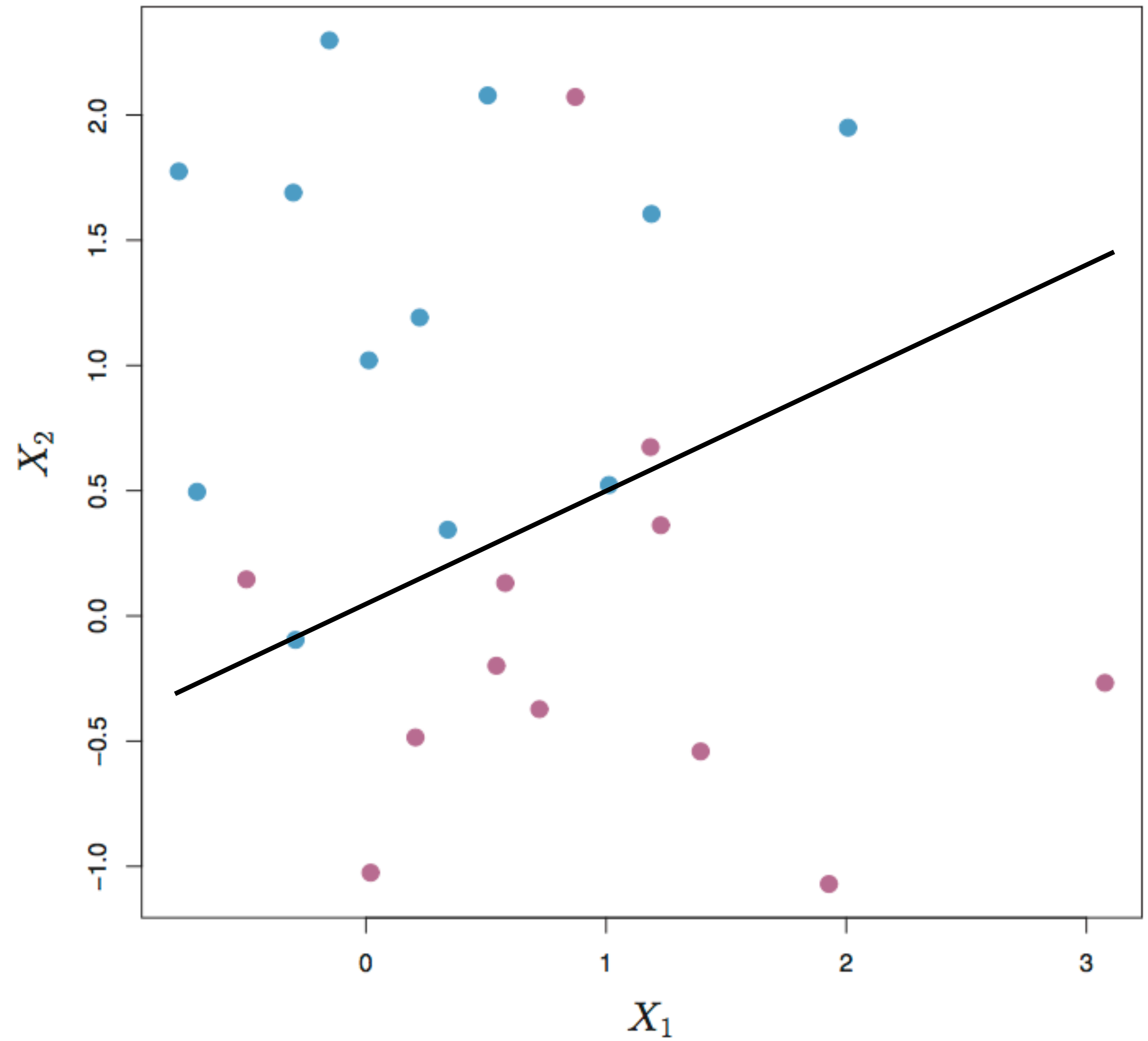
The non-separable case

- In most cases the data is not separable. In this case we can try to look for the hyperplane that almost separates the classes, the *support vector classifier*



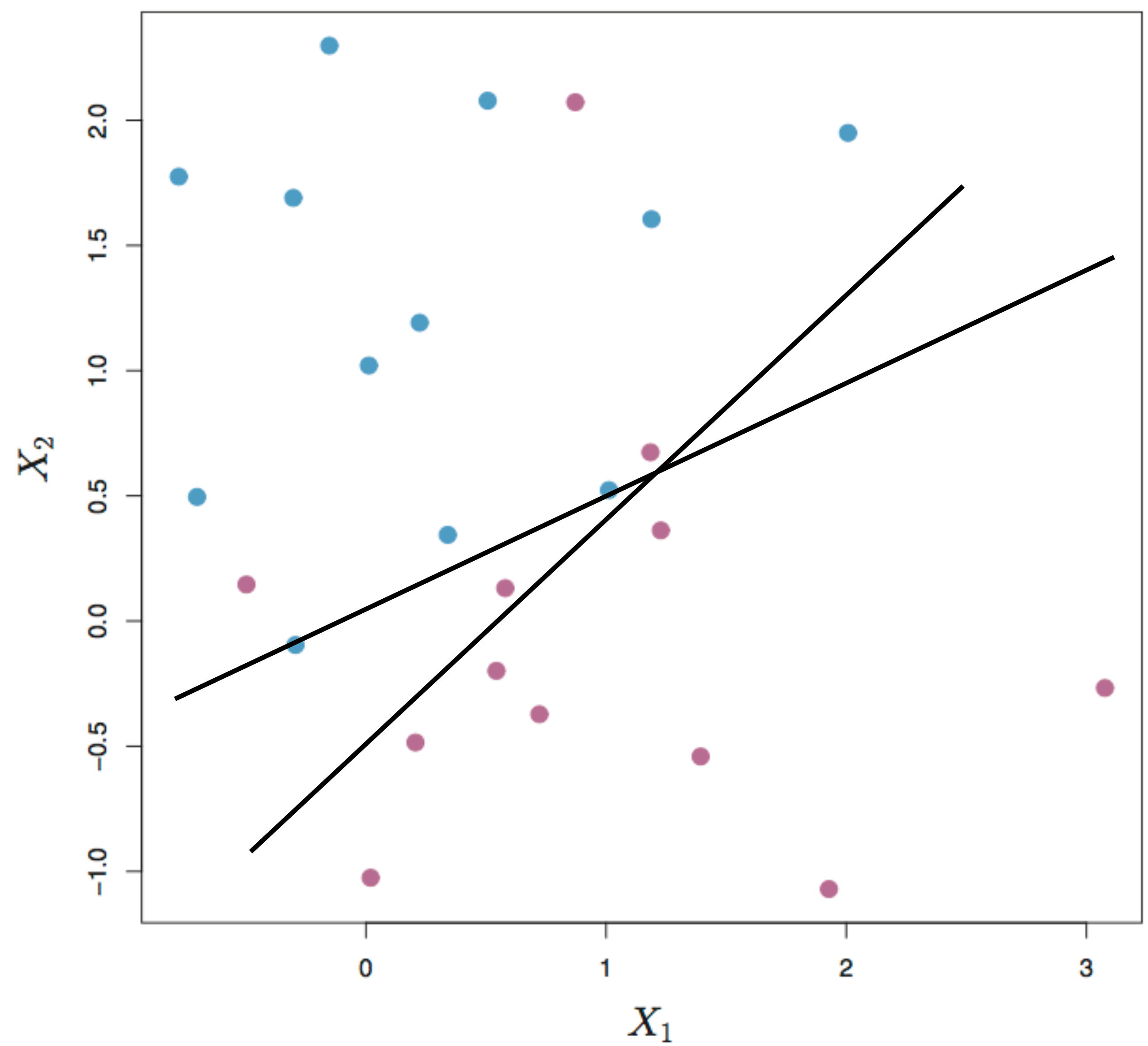
The non-separable case

- In most cases the data is not separable. In this case we can try to look for the hyperplane that almost separates the classes, the *support vector classifier*

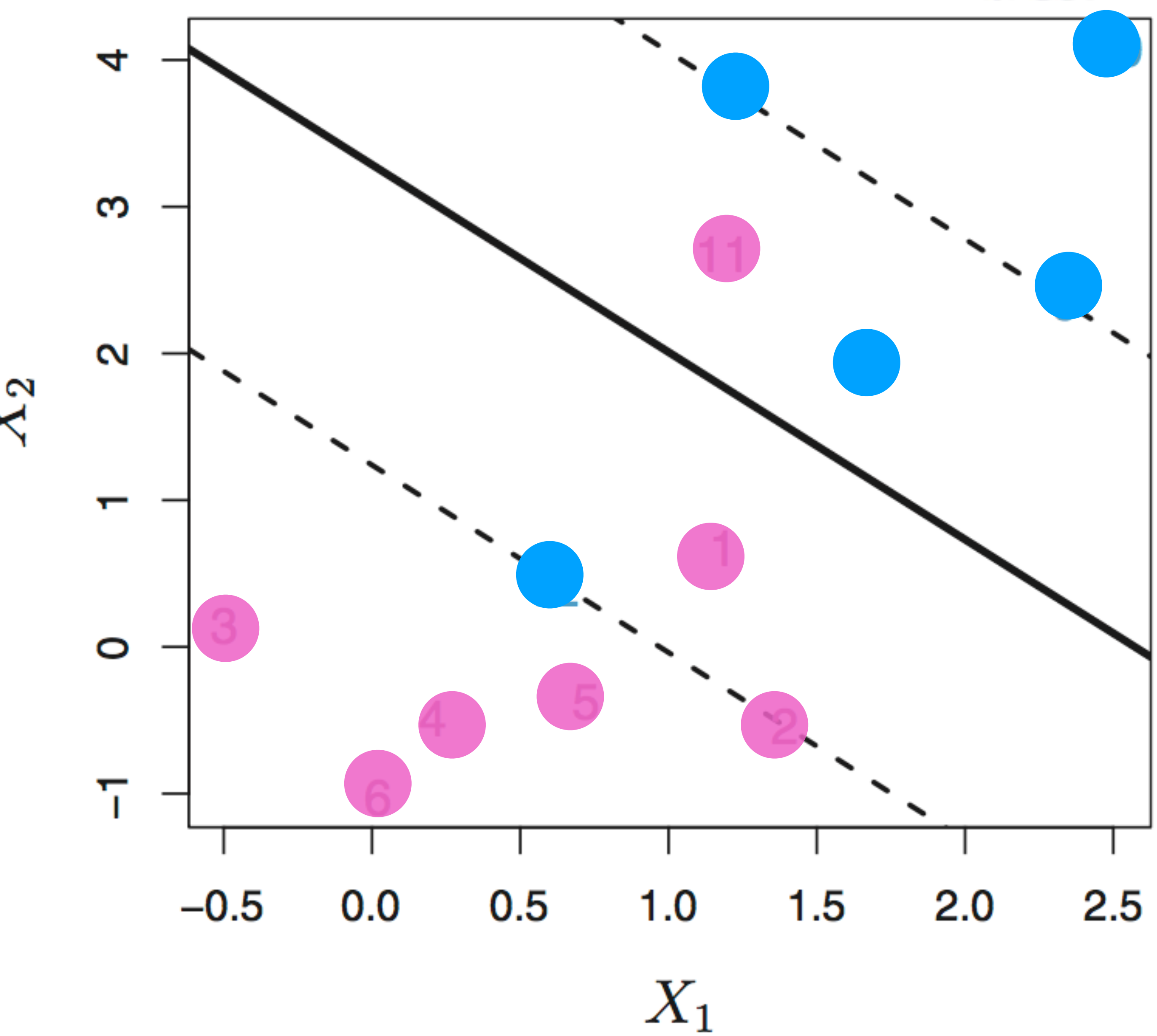


The non-separable case

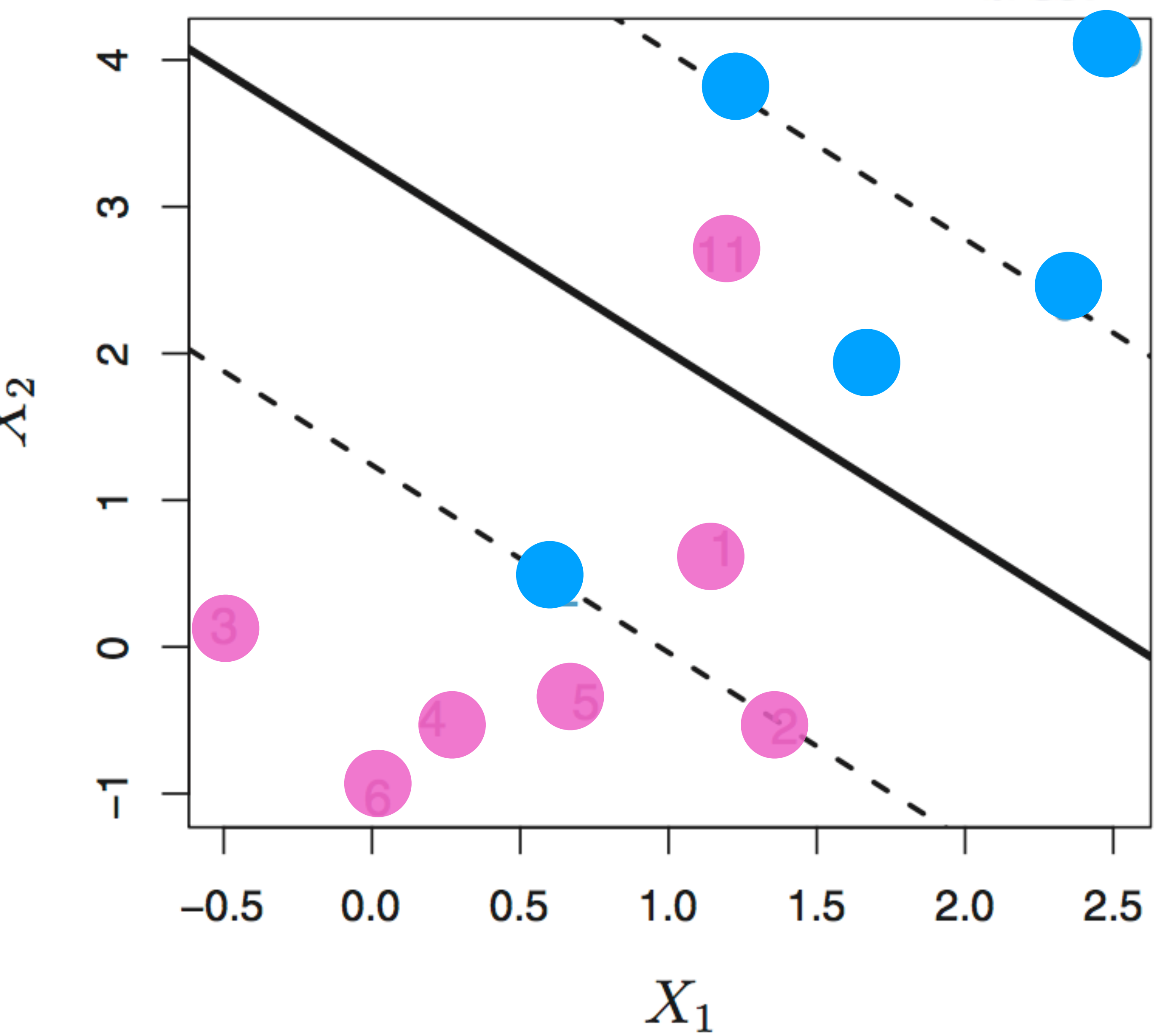
- In most cases the data is not separable. In this case we can try to look for the hyperplane that almost separates the classes, the *support vector classifier*



Support vector classifier



Support vector classifier



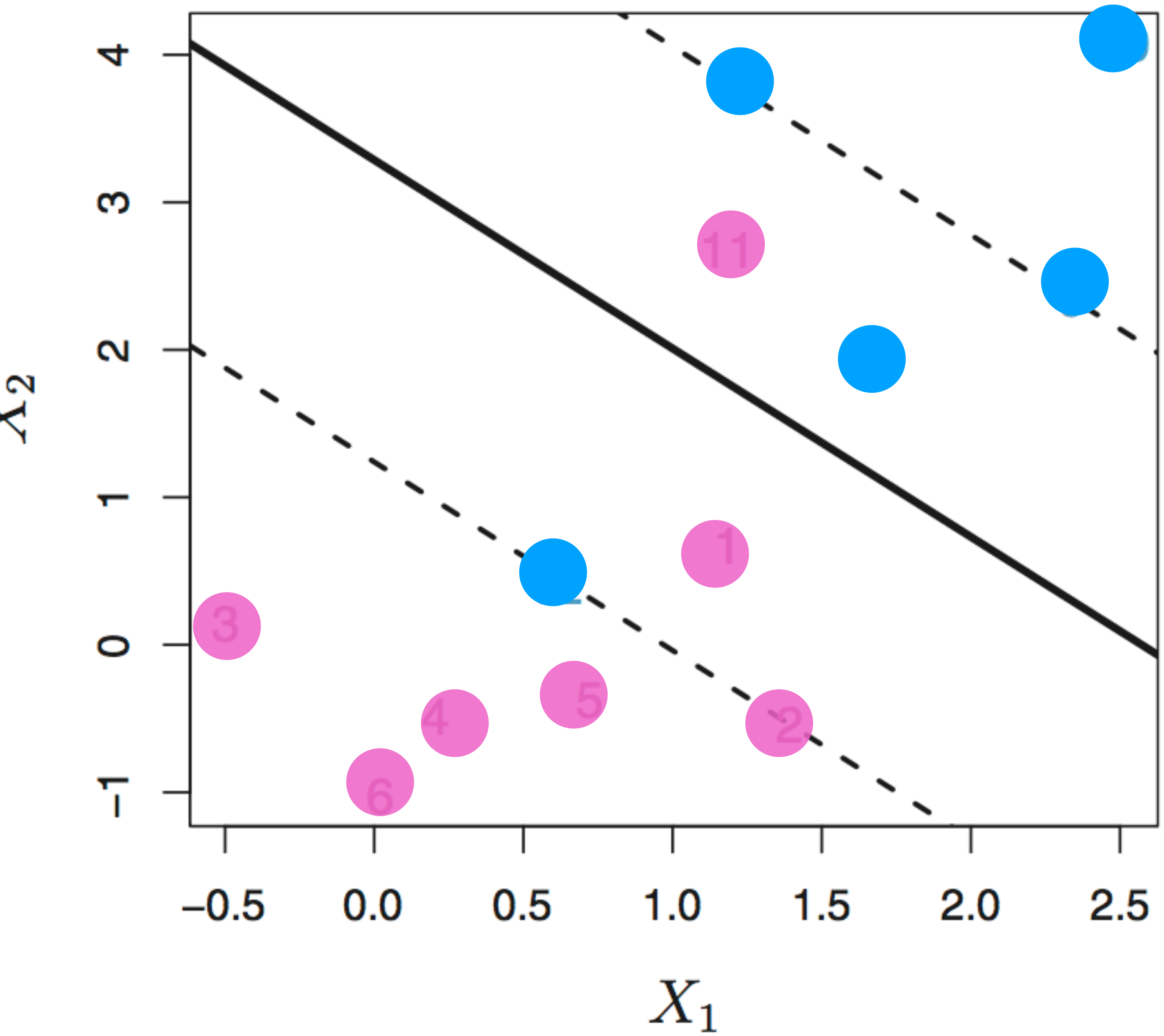
$$\arg \max_{\beta_0,\beta_1,\ldots,\beta_n,\,\epsilon_1,\ldots,\epsilon_n} M$$

$$\text{subject to } \sum_{j=1}^{n} \beta_j^2 = 1,$$

$$y_i * (\beta_0 + \sum_j X_{ij}\beta_{ij}) \geq M(1-\epsilon_i),$$

$$\epsilon_i \geq 0, \sum_i \epsilon_i \leq C$$

Support vector classifier



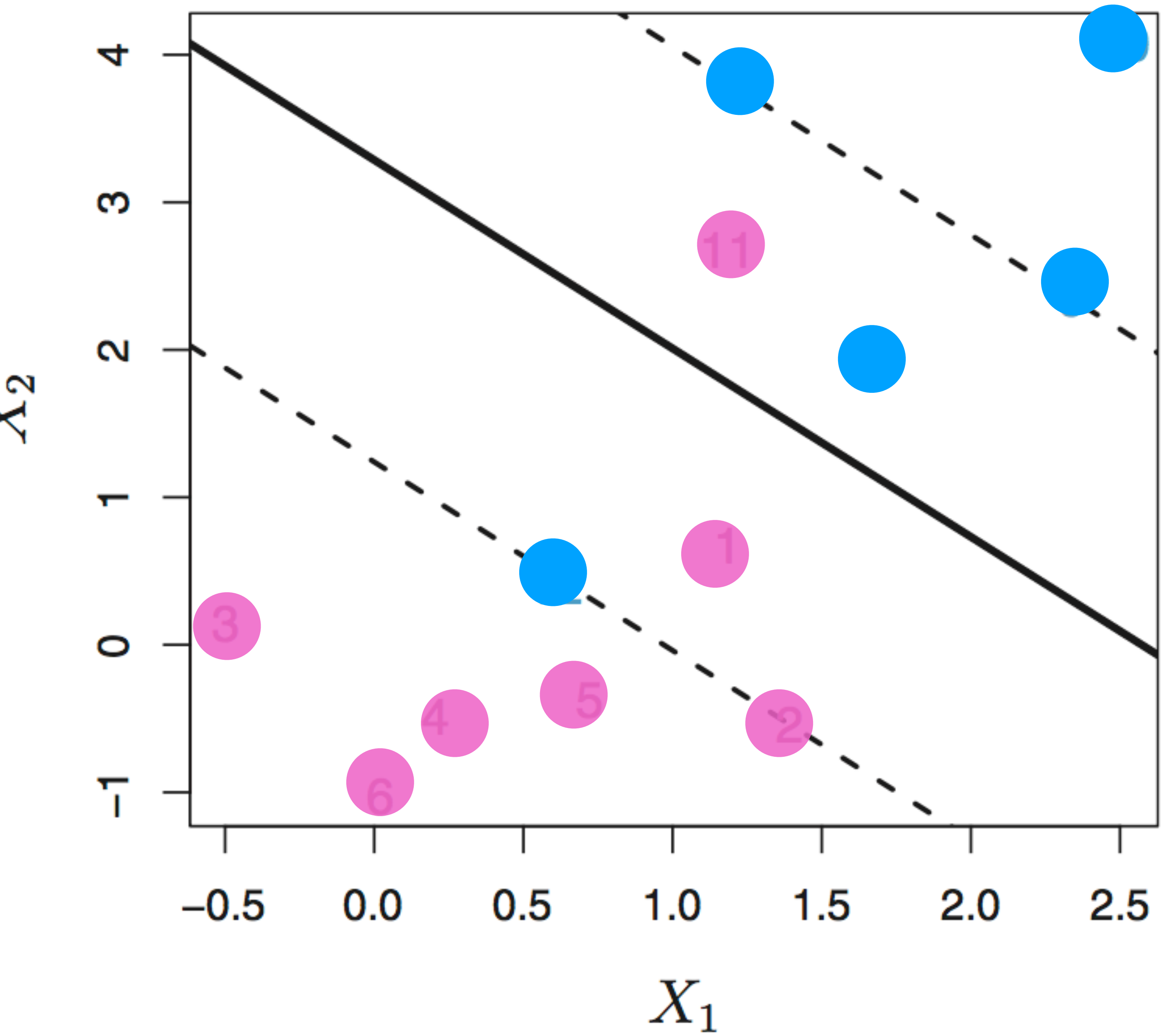
$$\arg \max_{\beta_0,\beta_1,\ldots,\beta_n,\epsilon_1,\ldots,\epsilon_n} M$$

$$\text{subject to} \sum_{j=1}^{n} \beta_j^2 = 1,$$

$$y_i * (\beta_0 + \sum_j X_{ij}\beta_{ij}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \sum_i \epsilon_i \leq C$$

Support vector classifier



$$\arg \max_{\beta_0,\beta_1,\ldots,\beta_n,\epsilon_1,\ldots,\epsilon_n} M$$

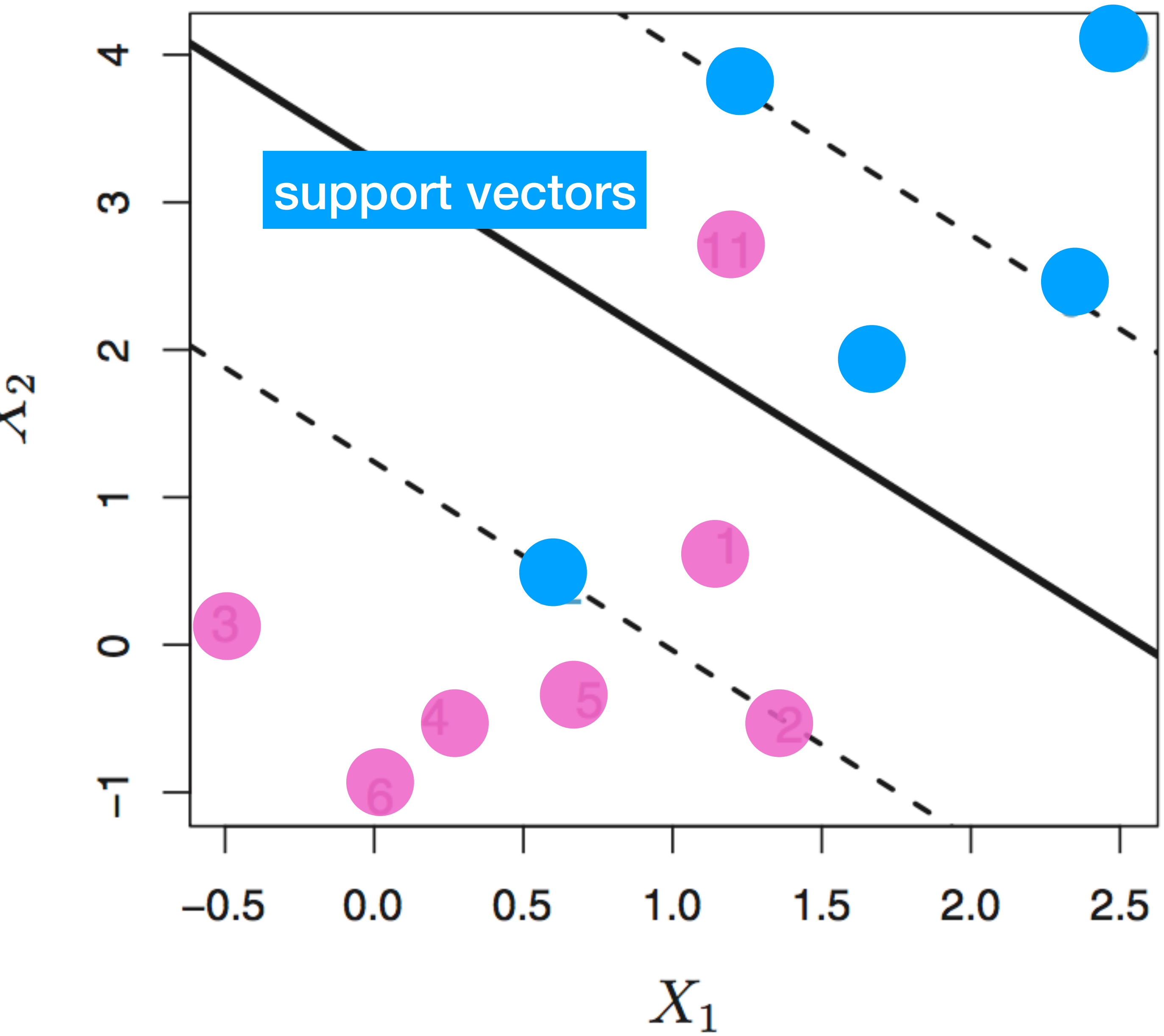
$$\text{subject to } \sum_{j=1}^{n} \beta_j^2 = 1,$$

$$y_i * (\beta_0 + \sum_j X_{ij}\beta_{ij}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \sum_i \epsilon_i \leq C$$

control overfitting

Support vector classifier



$$\arg \max_{\beta_0,\beta_1,\ldots,\beta_n,\epsilon_1,\ldots,\epsilon_n} M$$

$$\text{subject to } \sum_{j=1}^{n} \beta_j^2 = 1,$$

$$y_i * (\beta_0 + \sum_j X_{ij}\beta_{ij}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \sum_i \epsilon_i \leq C$$

control overfitting

SVC non linearities

- We can address non-linearities by enlarging the feature space as follows

$$X_1^2, X_1^3, X_1X_2, X_1X_2^2, \ldots$$

- Then we can fit

$$\beta_0 + X_1^2\beta_1 + X_1^3\beta_2 + X_1X_2\beta_3 + X_1X_2^2\beta_0 \ldots$$

SVM

- The SVC classifies a test instance to the size of the hyperplane it lies on. When K is the inner product it can also be expresses as

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

- To estimate it we just need the pairwise distance among observations. It happens that α_i is going to be non-zero just for the support vectors

- The support vector machine is an extension of the support vector classifier using kernels.

SVM kernels

- Kernels quantify the similarities between two observations

$$K(x_i, x_k) = \sum_j x_{ij} x_{kj}$$

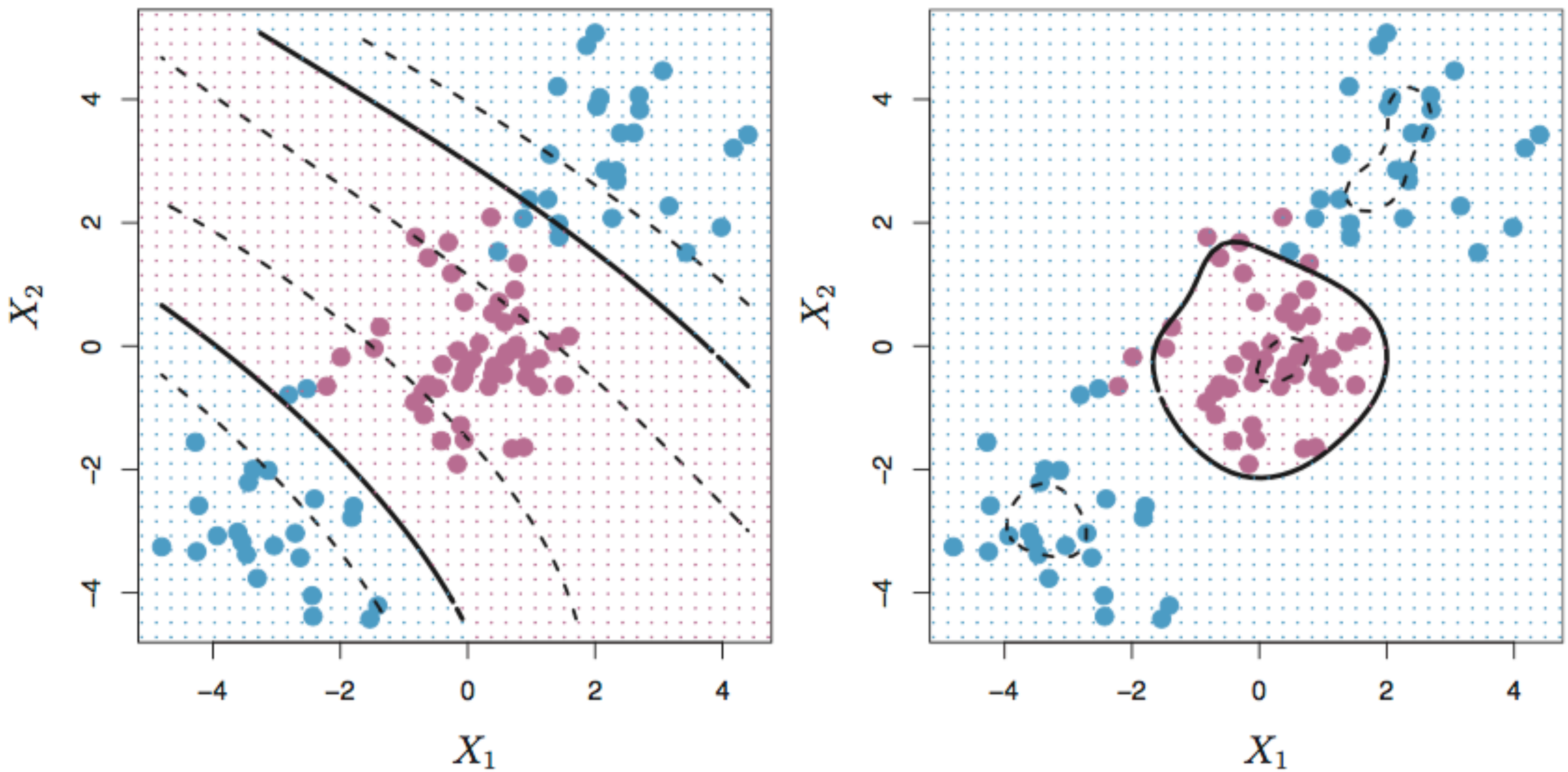
- Polynomial

$$K(x_i, x_k) = (1 + \sum_j x_{ij} x_{kj})^d$$

- Radial Kernel

$$K(x_i, x_k) = exp(-\lambda + \sum_j (x_{ij} x_{kj})^2)$$

SVM kernels



- Polynomial kernel of degree 3 Radial kernel