Machine Learning Methods for Communication Networks and Systems

Francesco Musumeci

Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)

Politecnico di Milano, Milano, Italy

Part I – 7: Further ML algorithms

Outline

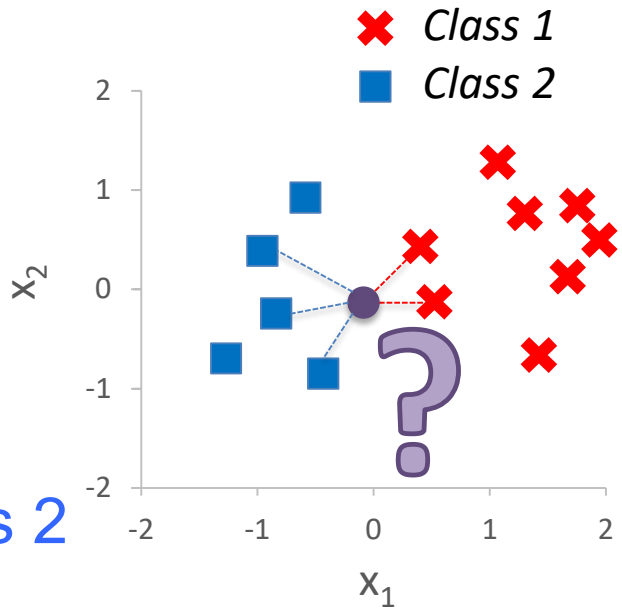
- K-Nearest Neighbors
- Tree-based methods
- Case Based Reasoning
- Anomaly detection

Outline

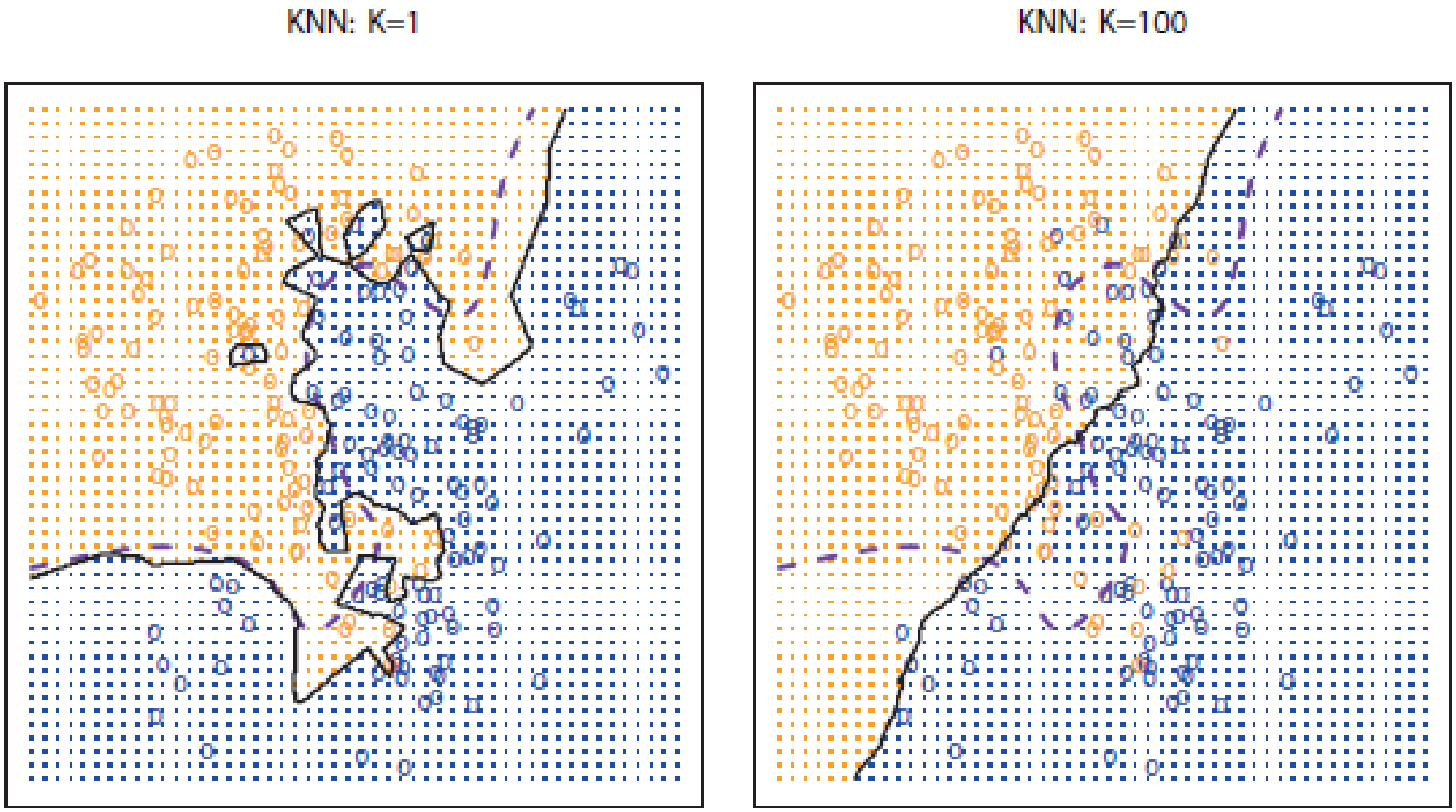
- K-Nearest Neighbors
- Tree-based methods
- Case Based Reasoning
- Anomaly detection

K-Nearest Neighbors

- Used for classification and regression
- Non-parametric method
- Decides based on the K nearest points in the dataset
  - no need for training phase
- Computationally complex for large datasets
- Need to choose K
  - Drives the bias/variance trade-off
- Example 1: classification (K=3)
  - Choose the most frequent class among the KNN → predict class 1
  - Changing the value of K (e.g. K=5) may affect the result → predict class 2



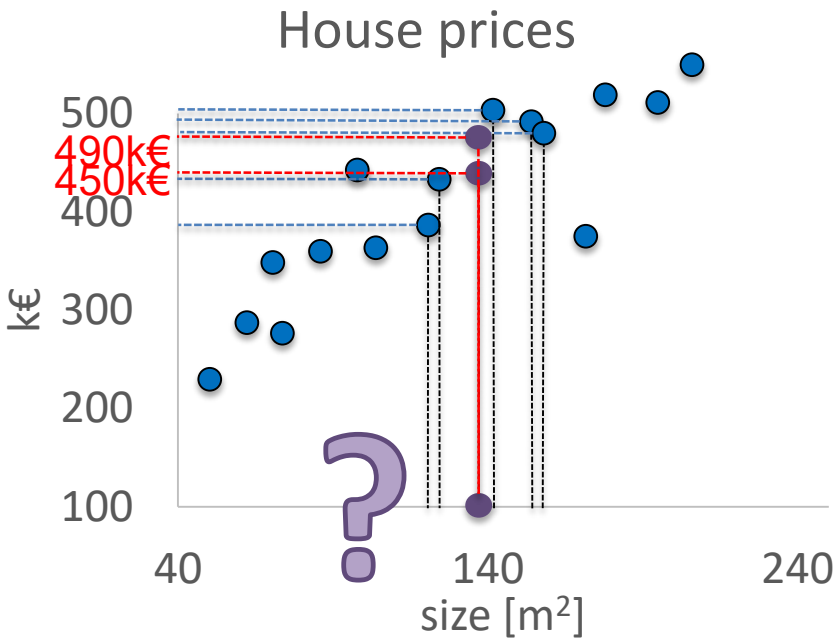
K-Nearest Neighbors
Classification



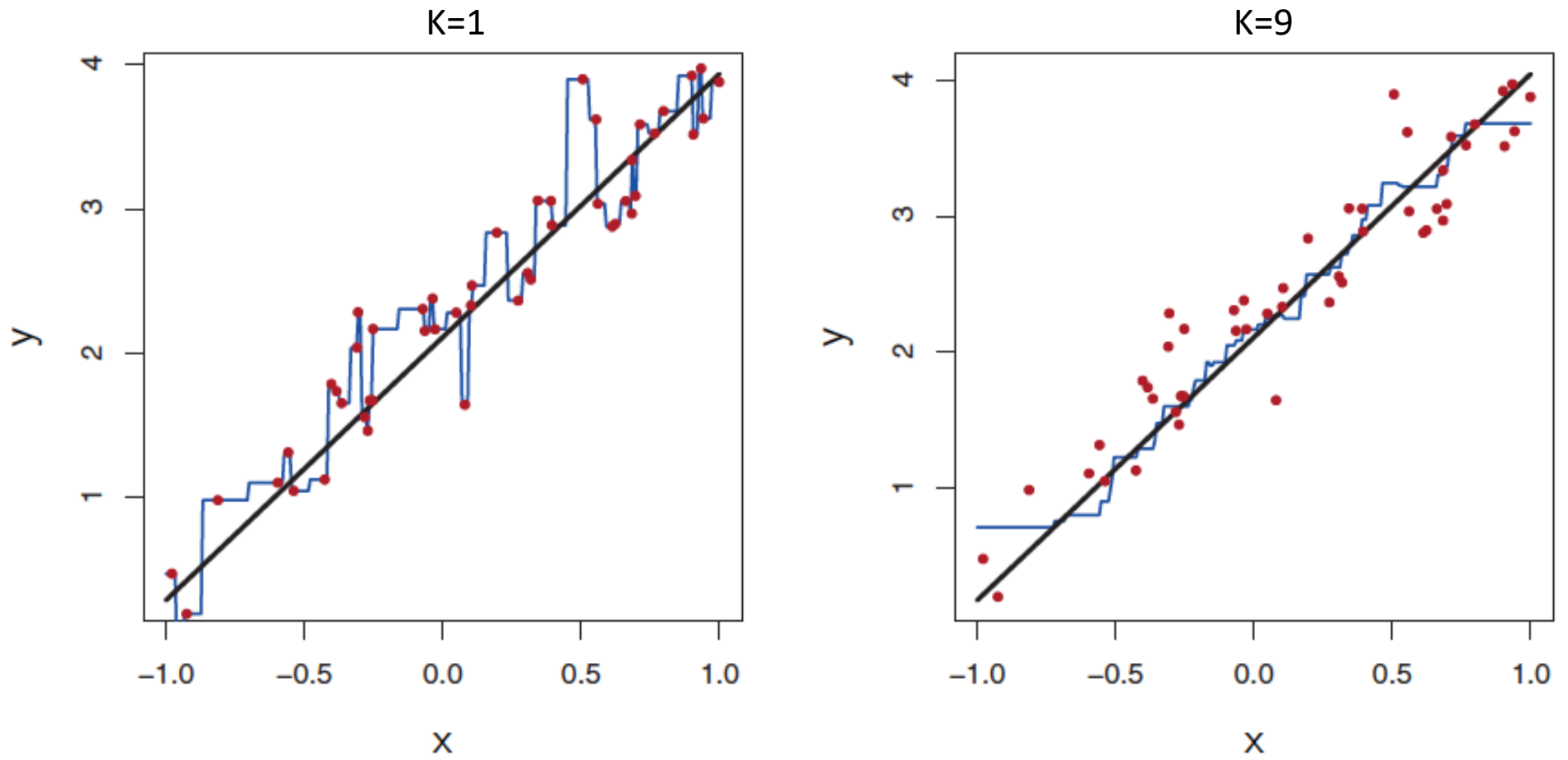
Source: ISLR

K-Nearest Neighbors

- Example 2: regression
  - For a new point x_{test} :
    - find the KNNs (x_1, x_2, …, x_K)
    - predict the output as: $y_{test}=1/K * avg(y_1, y_2, …, y_K)$
  - Changing the value of K may affect the result
    - K=3 → y_{test}=450k€
    - K=5 → y_{test}=490k€



K-Nearest Neighbors
Regression



Source: ISLR

Outline

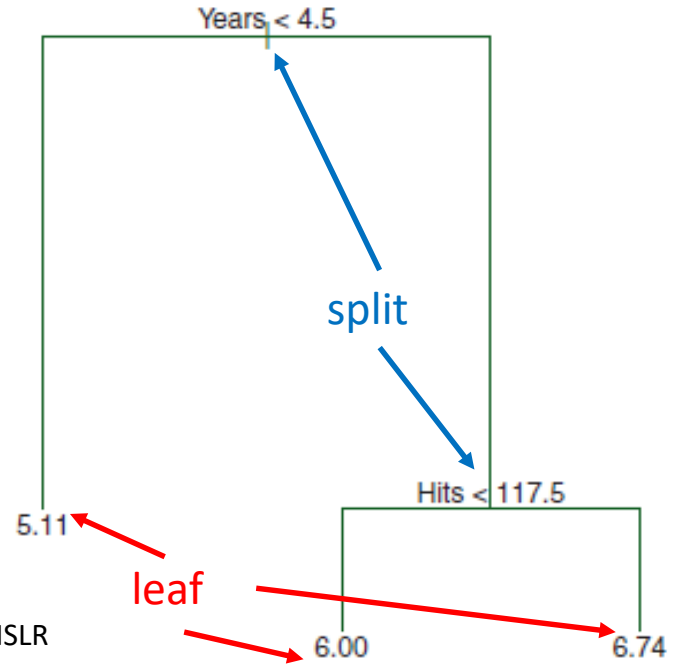
- K-Nearest Neighbors
- Tree-based methods
- Case Based Reasoning
- Anomaly detection

Tree-based methods

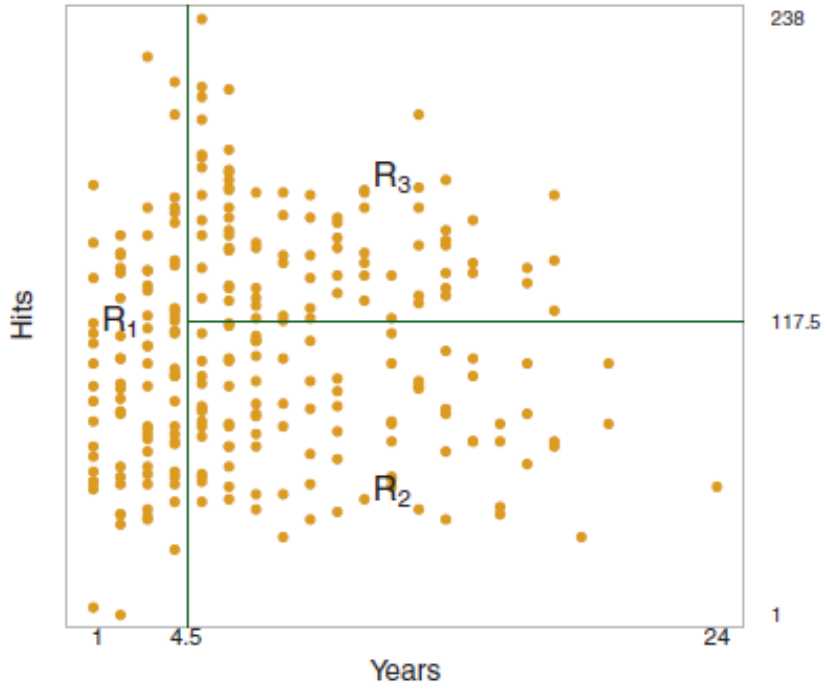
- Decision Trees (DTs) are very good in **interpretability**
  - Very close to human decision-making
  - Easy “visualization” of features-space segmentation
  - Used for both regression and classification
- However…
  - DTs suffer from prediction accuracy limitations (e.g., w.r.t. linear/logistic regression)
- To overcome accuracy limitations→ combine many DTs:
  - Bagging
  - Random forest
  - Boosting

Tree-based methods
Example of Decision Tree

- Baseball players and their (log)salary as a function of:
  - Nr of **years** in highest category
  - Nr of **hits** in the last season
- Every "leaf" (or "terminal node") of the tree corresponds to a region (R_j)



Source: ISLR



Tree-based methods
Decision Tree rationale

- Stratification of the feature space

1. **BUILD the DT**: divide the features space (i.e., the set of all possible values for $X_1, X_2, \ldots, X_n$) into J **distinct and non-overlapping regions**, $R_1, R_2, \ldots, R_J$
   - Critical choice of no. of regions J (no. of leaves in the DT)

2. **USE the DT**: prediction for a **new** observation falling in region R_j:
   - the ***mean*** of the values of responses for all other observations contained in R_j (regression)
   - the ***mode*** (the most recurrent value) of the values for all other observations contained in R_j (classification)

Tree-based methods
How do we build a DT?

- Let's consider a *regression* problem (***regression tree***)
- *Recursive binary splitting*:
  1. for every feature ***j*** and split point ***s*** define **half-planes**:

$$R_1(j,s) = \{X \mid X_j < s\} \text{ and } R_2(j,s) = \{X \mid X_j \geq s\}$$

  2. find *(j,s)* such that the following is minimized:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

$\hat{y}_{R_1}$: average value in R_1

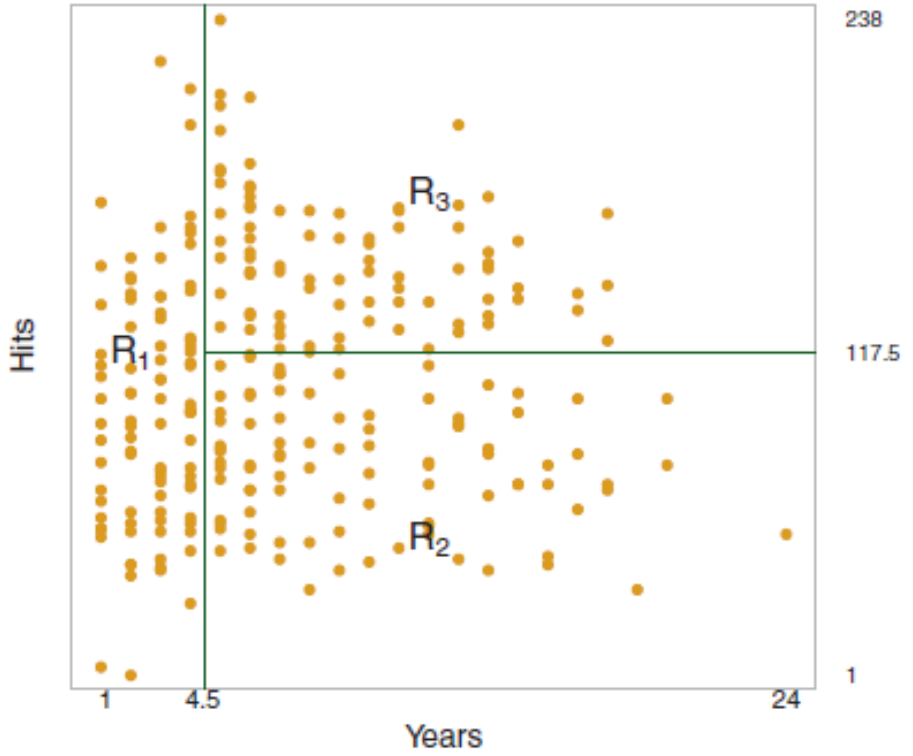
$\hat{y}_{R_2}$: average value in R_2

RSS (residual sum of squares)

  3. split the features space using *(j,s)*
  4. repeat steps 1,2,3 by splitting one of the existing regions
     - in general, we aim at minimizing the overall RSS
  5. STOP when a certain condition is met (e.g., every region contains no more that N observations)

NOTE: at each step, previously-split features can be used again!

Tree-based methods
Example of Decision Tree



Source: ISLR

Tree-based methods

Tree pruning

- A DT can highly overfit the data
  - Many splits (many regions) → high variance & low bias
  - Few splits (few regions) → low variance & high bias
- 1st alternative: stop splitting until there is a predefined amount of reduction in the RSS
  - Problem: a good split can be neglected as it might come after a bad split
- 2nd alternative: build a very large tree T_0, then **prune it**
  - For a given α, there exists a ***pruned subtree T***, s.t. the following is minimized:

Overall RSS

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

Size of the tree (nr of regions it contains)

  - α controls the bias/variance trade-off (i.e., complexity vs accuracy)

Tree-based methods
Classification trees

- RSS cannot be used as a splitting criteria
- Use other metrics, i.e., minimize one of the following metrics:
  - Classification error rate

$$E = 1 - \max_k(\hat{p}_{mk})$$

  - Gini index

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

  - Cross-entropy

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$$

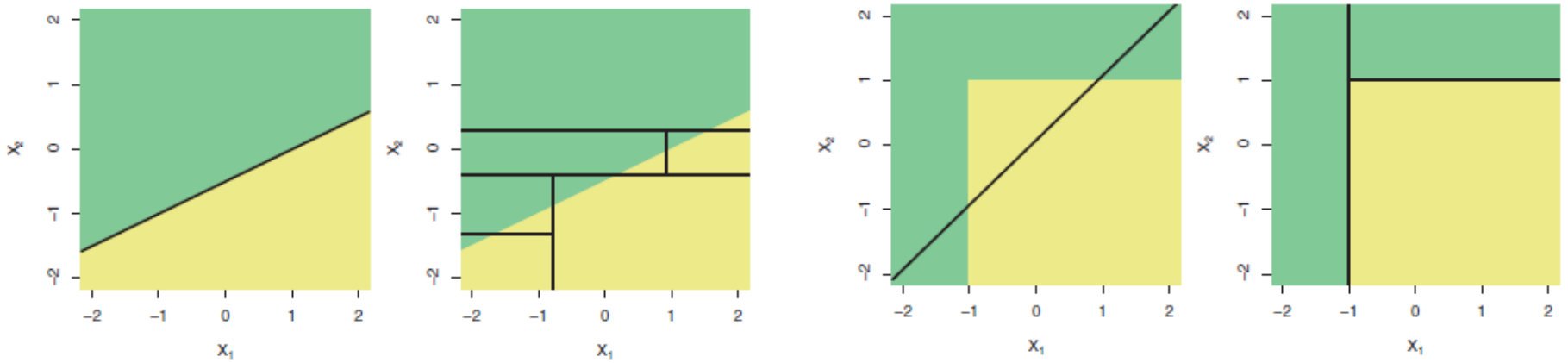
$\hat{p}_{mk}$ fraction of training observations in the *m*-th region belonging to the *k*-th class

improve *node purity*

What if we have qualitative features?
A split on one of these variables means assigning some of the qualitative values to one branch of the tree and ALL THE OTHER VALUES to the other branch (one vs all)

Tree-based methods
Tree vs linear models

- Which one is better?



- It depends on our data!

Source: ISLR

Tree-based methods

Advantages and disadvantages of DTs

- PROs
  - Very easy to explain (even easier than linear regression)
  - Close to human decision-making
  - Can be displayed graphically, and are easily interpreted even by a non-expert
  - Can easily handle qualitative features without dummy variables
- CONs
  - Can be very non-robust. A small change in the data can cause a large change in the final tree
  - Generally do not have the same level of predictive accuracy
    - To improve accuracy: *Bagging*, *Random Forest*, *Boosting*. Make use of different trees and produce “averaged” results

Outline

- K-Nearest Neighbors
- Tree-based methods
- Case Based Reasoning
- Anomaly detection

Case Based Reasoning

- Close to human reasoning
- No need for training phase
- Makes use of a Knowledge Base (KB) containing history of (case, action) pairs
  - meaning of (*x*,*y*) in the KB: when situation "*x*" occurred, I made decision "*y*"
  - KB can be updated when doing new predictions
    - Adding new (*x*,*y*)
    - Removing old (*x*,*y*): "forgetting" algorithms needed
- Need to define a *similarity function* $sim(x_1, x_2)$
- To make prediction for a new element x_{new}:
  - select (x^*,y^*) in the KB s.t. $sim(x_{new}, x^*)$ is maximum
  - Predict $y_{new} = y^*$
- KNN is a special case of CBR

Outline

- K-Nearest Neighbors
- Tree-based methods
- Case Based Reasoning
- Anomaly detection

Anomaly detection

- Given a training set ($x^{(1)}$, $x^{(2)}$, …, $x^{(m)}$)
  - $x^{(i)}$ is a *n*-featured vector
- We want to know if a new example x_{test} is anomalous
- Approach:
  - Define a model $p(x)$ representing the probability that example x is NOT anomalous
  - One new example x_{test} is anomalous if $p(x_{test})<\varepsilon$
- Assumption: examples in the data set are *normally-distributed*

$$p(x;\mu;\sigma^2)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\mu=\frac{1}{m}\sum_{i=1}^{m}x^{(i)}$$

$$\sigma^2=\frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)^2$$

Note: $x^{(i)}$, μ and σ^2 are vectors (we have *n* features)!!

Anomaly detection
Algorithm

- Given a training set ($x^{(1)}$, $x^{(2)}$, …, $x^{(m)}$) with features *1,2,…j,…,n*
- For each feature *j*, evaluate:

$$p(x_j;\mu_j;\sigma_j^2)=\frac{1}{\sigma_j\sqrt{2\pi}}e^{-\frac{(x_j-\mu_j)^2}{2\sigma_j^2}}$$

$$\mu_j=\frac{1}{m}\sum_{i=1}^{m}x_j^{(i)}$$

$$\sigma_j^2=\frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)}-\mu_j)^2$$

- For a new example x_{test} compute

$$p(x_{test})=\prod_{j=1}^{n}p(x_j;\mu_j;\sigma_j^2)$$

Independence assumption
(can be substituted by multivariate Gaussian distribution by computing covariance matrix instead of σ^2)

- x_{test} is anomalous if $p(x_{test})<\varepsilon$

ε is selected w/ cross-validation