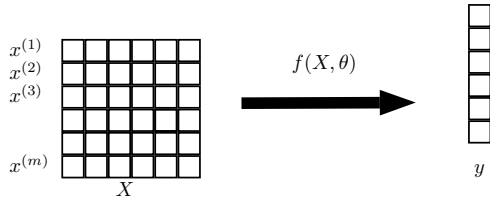
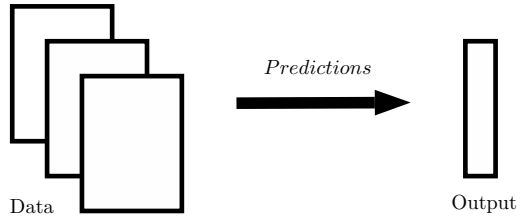


POLIMI GRADUATE SCHOOL OF MANAGEMENT

MACHINE LEARNING IN PYTHON - LAB SESSION

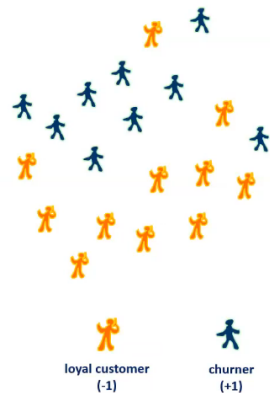
Andrea Mor - andrea.mor@polimi.it

SUPERVISED LEARNING



CLASSIFICATION PROBLEM

attributes						
customers	Area	Pothers	Pmob	...	NumSMS	Class
	2	0.14	0.59	...	18	1
	3	0.26	0.35	...	9	-1
	1	0.37	0.23	...	1	1
	⋮	⋮	⋮	⋮	⋮	⋮
	4	0.41	0.27	...	64	-1
	← past data →					
Area	Pothers	Pmob	...	NumSMS	Class	
1	0.27	0.67	...	36	?	
4	0.44	0.22	...	50	?	
4	0.31	0.47	...	14	?	
⋮	⋮	⋮	⋮	⋮	⋮	
2	0.31	0.14	...	49	?	
← future data →						

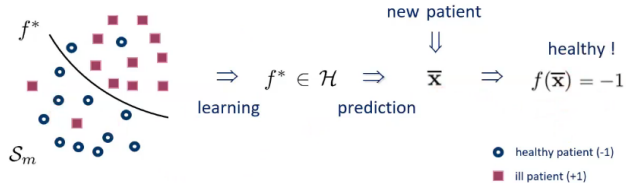


CLASSIFICATION FORMULATION

$S_m = \{(\mathbf{x}_i, y_i), i \in \mathcal{M}\}$: **training set, where** $\mathbf{x}_i \in \mathbb{R}^n$ **and** $y_i \in \mathcal{D}$

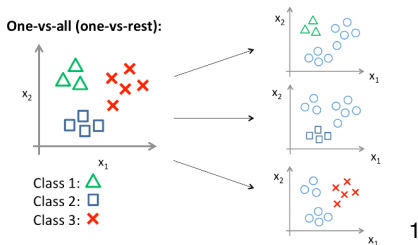
\mathcal{H} **denotes a set of functions** $f(\mathbf{x}) : \mathbb{R}^n \mapsto \mathcal{D}$

Classification problem: define a hypotheses space \mathcal{H} **and a function** $f^* \in \mathcal{H}$ **which optimally describes the relationship between** \mathbf{x}_i **and** y_i



MULTI-CLASS CLASSIFICATION

1. **One-vs-Rest** We perform $|H|$ different binary classifications: one for every class.



We decide based on a majority vote.

2. **One-vs-One** We perform $|H|(|H| - 1)/2$ binary classifications: one for every pair of classes. We decide based on a majority vote.

CLASSIFICATION MODELS

- ▶ Heuristics Methods
 - Nearest Neighbours
 - Classification Trees
- ▶ Regression Methods
 - Logistic regression
- ▶ Separation Methods
 - Support vector machine
 - Perceptron
 - Neural Networks
- ▶ Probabilistic Methods
 - Bayesian Methods
- ▶ Ensemble Methods

EVALUATION DIMENSIONS

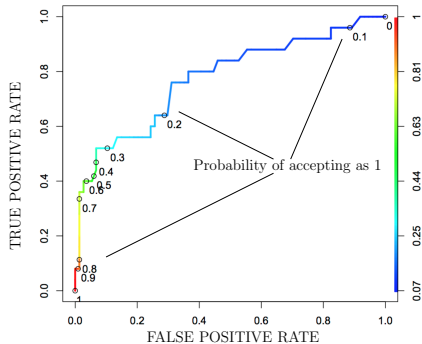
- ▶ Prediction accuracy
- ▶ Speed
- ▶ Robustness
- ▶ Scalability
- ▶ Interpretability
- ▶ Rules effectiveness

QUALITY MEASURES - CONFUSION MATRIX

		Prediction outcome	
		0	1
Actual value	0	True Negative	False Positive
	1	False Negative	True Positive

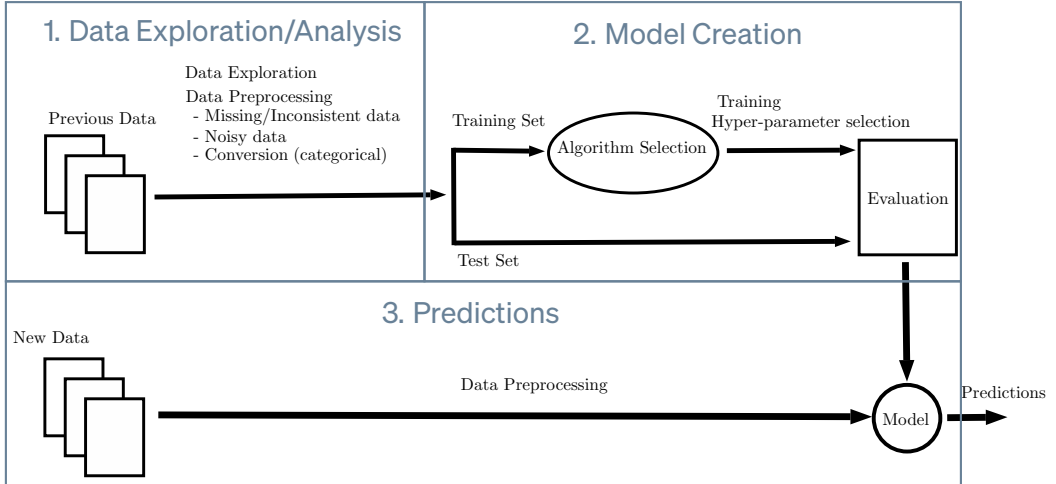
- ▶ Precision = $\frac{TP}{TP+FP}$
“proportion of true positives among **positive predictions**”
- ▶ False Positive rate = $\frac{FP}{FP+TN}$
“proportion of false positives among **actual negatives**”
- ▶ Recall (True Positive rate) = $\frac{TP}{FN+TP}$
“proportion of true positives among **actual positive**”
- ▶ Geom. mean = $\sqrt{\text{Precision} \times \text{Recall}}$
- ▶ F-score = $\frac{(\beta^2 + 1)}{\beta^2} \frac{1}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}$

QUALITY MEASURES - ROC CURVE & AUC

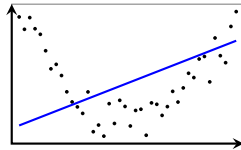
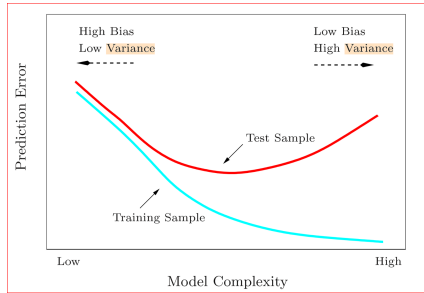


- ▶ If we accepting even with small probability then $TPR = FPR = 1$
- ▶ If we accepting just with high probability then $TPR = FPR = 0$
- ▶ The perfect classifier is the the point (0, 1)
- ▶ $AUC \in [0.5, 1]$ area under the curve is a quality measure of our algorithm.

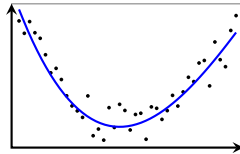
WORKFLOW



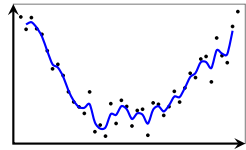
UNDER/OVER-FITTING



Underfitting

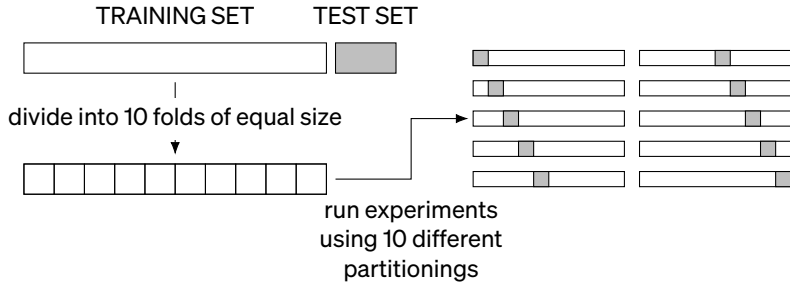


Balance



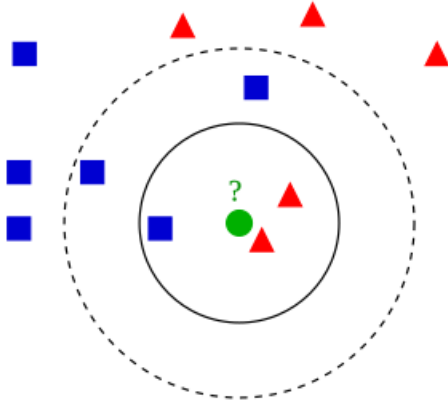
Overfitting

CROSS VALIDATION



THE ALGORITHMS

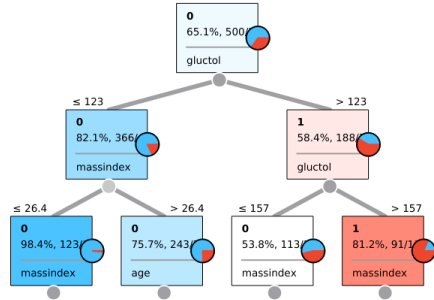
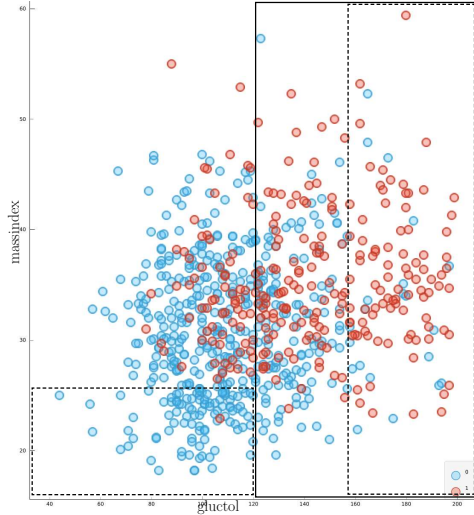
KNN K-NEAREST NEIGHBOURS



Main Parameters

- ▶ k : number of neighbours
- ▶ neighbour weights
- ▶ distances

CLASSIFICATION TREE

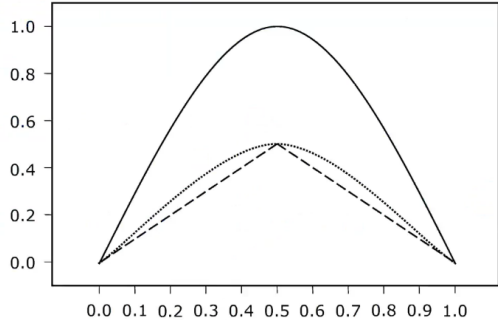
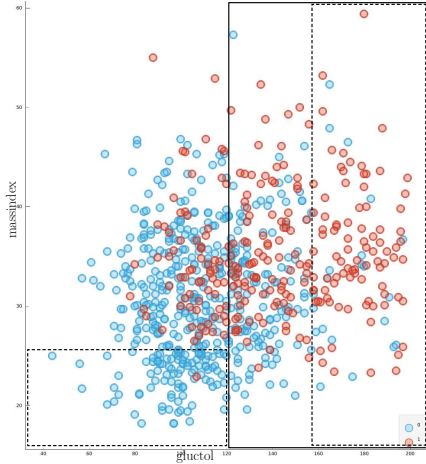


Tree

types

- ▶ Binary tree (zero/two descendants)
- ▶ General trees
- ▶ Uni-variate tree ($X_j < b$)
- ▶ Multi-variate tree ($\sum_{j=1}^n w_j x_j = b$)

CLASSIFICATION TREE



criteria

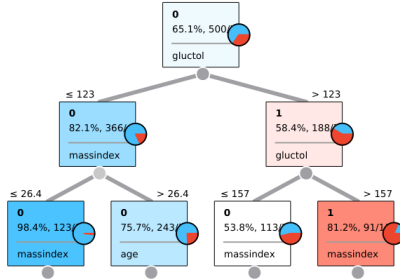
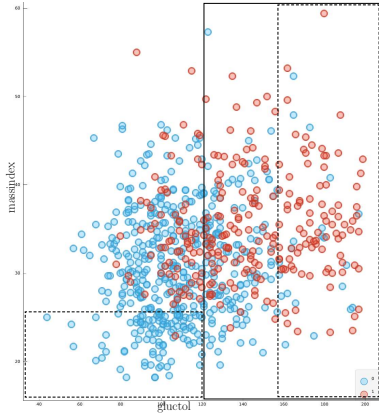
► **Gini index:**

► **Entropy index:**

► **Miss-classification index:**

$$1 - \sum_{h=1}^H f_h^2$$
$$- \sum_{h=1}^H f_h \log_2 f_h$$
$$1 - \max_h f_h$$

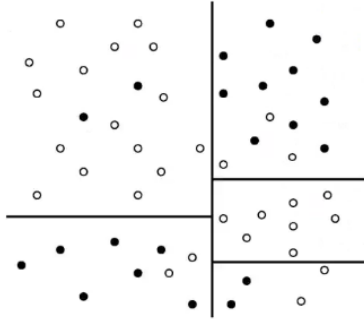
CLASSIFICATION TREE



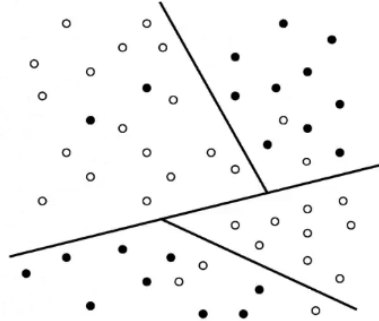
Main Parameters

- ▶ impurity measure: “gini”, “entropy”
- ▶ max_depth
- ▶ min_samples_split: minimum number of samples to split an internal node
- ▶ min_sample_leaf: minimum number of samples required to be at a leaf node

CLASSIFICATION TREE

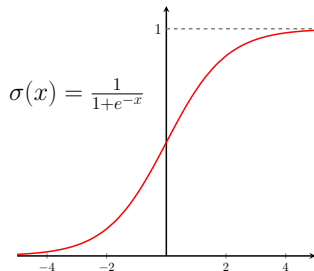


classification by an
axis parallel tree



classification by an
oblique tree

LOGISTIC REGRESSION



The model postulates that

$$\log \left[\frac{P(y = 1|x)}{P(y = 0|x)} \right] = w^T x$$

then if $p = P(y = 1|x)$:

$$\frac{p}{1-p} = e^{w^T x}$$

$$\Rightarrow p = P(y = 1|x) = \frac{e^{w^T x}}{1 + e^{w^T x}} = \frac{1}{1 + e^{-w^T x}},$$

$$P(y = 0|x) = 1 - p = \frac{1}{1 + e^{w^T x}} = \frac{e^{-w^T x}}{1 + e^{-w^T x}}$$

NAIVE BAYESIAN CLASSIFIER

- Bayes Theorem

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|y) P(y)}{\sum_{l=1}^H P(\mathbf{x}|y) P(y)}$$

- Maximum a posteriori hypothesis

$$y_{MAP} = \arg \max_{y \in \mathcal{H}} P(y|\mathbf{x}) = \arg \max \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

- Independence (Naive)

$$P(\mathbf{x}|y) = P(x_1|y) \times P(x_2|y) \times \cdots \times P(x_n|y) = \prod_{j=1}^n P(x_j|y)$$

NAIVE BAYESIAN CLASSIFIER

- Categorical/discrete attributes

$$P(x_j|y) = P(x_j = r_{jk}|y = v_h)$$

→ empirical frequency of the observed value on the class v_h

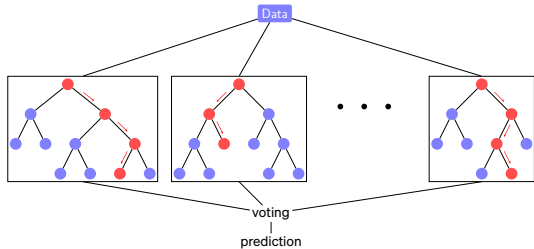
- Numerical attribute

$$P(x_j|y) \sim N(\mu_{jh}, \sigma_{jh})$$

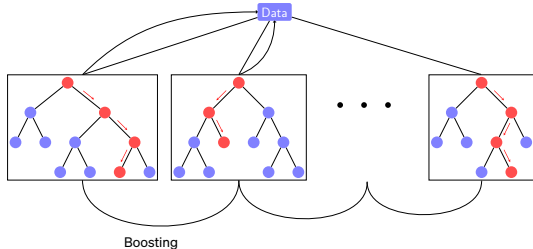
→ assuming Gaussian density with empirical parameters

ENSEMBLE METHODS

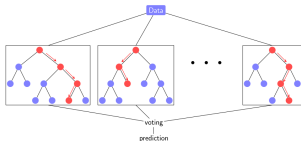
Bagging



Boosting



RANDOM FOREST

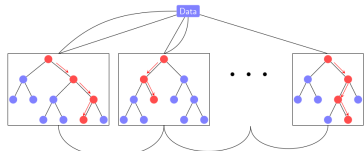


1. Create different (simple) tree models (stumps)
2. Each model is created with a subset of observation/features ($\sim 2m/3$)
3. We combine the prediction of all trees

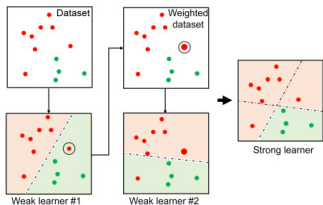
Main Parameters

- ▶ `n_estimators`: Number of trees
- ▶ `max_features`: Number of features selected for the split
- ▶ `bootstrap=False`: Use all samples
- ▶ Tree parameters

ADABOOST



Boosting



1. Assign equal weights to observations $w_i^{(0)} = 1/m$
2. For $k = 1, \dots, K$
 - Select a sample of observations based on the weights.
 - Create the k -th weak learner and compute predictions $x^{(k)}$
 - Compute the model **weighted** error and assign its coefficient:

$$\alpha^{(k)} = \lambda \times \log((1 - \text{error})/\text{error})$$

- Update sample weights:

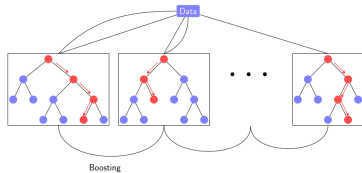
$$w_i^{(k+1)} \propto w_i^{(k)} \times \exp(-\alpha^{(k)} y_i \hat{x}_i^{(k)})$$

3. Final weighted prediction

Main Parameters

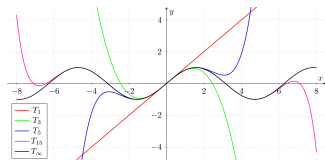
- n_estimators: Number of estimators (K)
- base_estimator: Weak estimator type
- learning_rate: weights of estimator in final decision (λ)

GRADIENT BOOSTING



1. Train a weak learner F_0 and compute predictions $\hat{y}_0 = F_0(x)$
2. For $k = 1, \dots, K$
 - Compute the difference between the target (y) and the prediction of the current learner (\hat{y}_{k-1})
 - Train a weak learner that minimizes the loss function (error):

$$P_k(x) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \dots + \frac{f^{(k)}(a)}{k!}(x - a)^k$$



$$f_k = \arg \min_f L_m = \arg \min_f \sum_{i=1}^n l(y_i, F_{k-1}(x_1) + f(x_i))$$

- $F_k = F_{k-1} + \lambda f_k$

Main Parameters

- **n_estimators:** Number of estimators (K)
- **base_estimator:** Weak estimator type
- **learning_rate:** weights of estimator in final decision (λ)

THANK YOU