



SAPIENZA  
UNIVERSITÀ DI ROMA

# METODI DI INTELLIGENZA ARTIFICIALE E MACHINE LEARNING IN FISICA

S. Giagu - AA 2019/2020  
Lezione 3: 4.3.2020

# METRICHE PER VALUTARE LE PRESTAZIONI DI UN ALGORITMO DI ML

- La valutazione delle prestazioni di un dato algoritmo di classificazione costituisce uno degli aspetti fondamentali del machine learning
- permette di decidere:
  - quanto buono sia il classificatore progettato e come si confronti con algoritmi e tecniche competitive
  - se sia possibile o meno migliorarne la prestazione tramite tuning dei parametri (hyperparameter tuning)
- varie figure di merito utilizzate in letteratura, discuteremo solo quelle più popolari ...

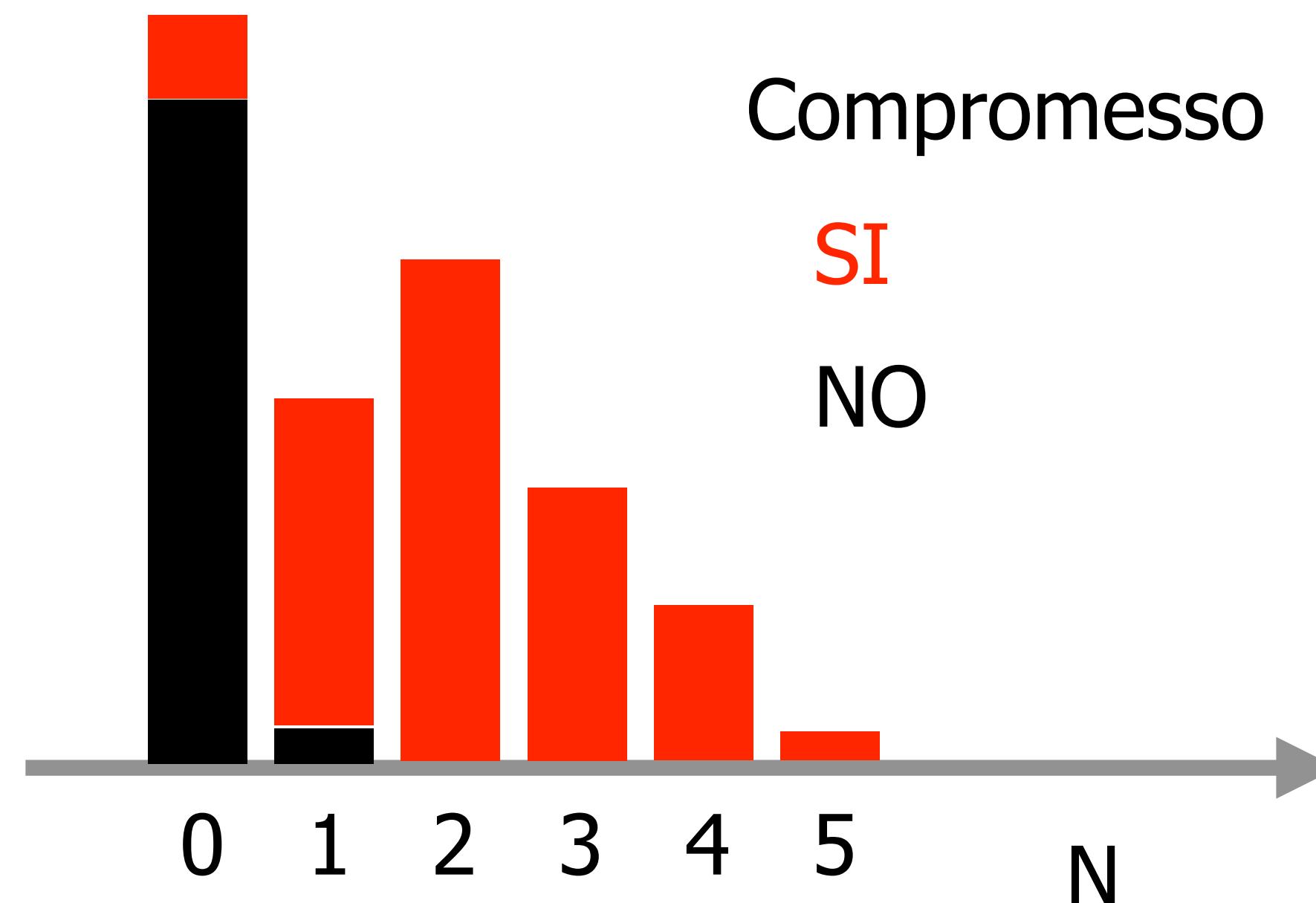


# UN ESEMPI ODI ALGORITMO DI CLASSIFICAZIONE

- un software che analizza gli account dei membri di FaceBook e decide per ogni account se è stato compromesso o meno da parte di hacker

dati a disposizione (esempio):

Un certo numero di casi di account per i quali è noto se sono stati compromessi o meno e per ognuno di tali casi il numero di accessi (N) all'account nello stesso giorno da computer in regioni geografiche diverse



Algoritmo Classificazione:

`if N>0: ⇒ 1 {violato}  
else: ⇒ 0 {non viol.}`



produce un uscita a valori  
discreti (classi di appartenenza)

# METRICHE USATE PER ALGORITMI DI CLASSIFICAZIONE: ACCURACY

- una delle metriche più utilizzate: la frazione di eventi correttamente classificata dall'algoritmo
- Esempio a due classi:
- Dataset  $T=\{\mathbf{x}_i, t_i\}$  con  $N$  eventi con label  $t_i = \{-1, +1\} = \{\text{classe 2, classe 1}\}$
- Regola di decisione:  $g(\mathbf{x}) > 0 \Rightarrow \text{classe 1, altrimenti classe 2}$

$$Acc(g, T) = \frac{1}{N} \sum_{i=1}^N \Theta[g(x_i)t_i]$$

se tutti gli eventi sono correttamente classificati:  $Acc = 1$

- LIMITI DELL'ACCURACY:
  - fallisce nel caso di campioni sbilanciati:
  - esempio:

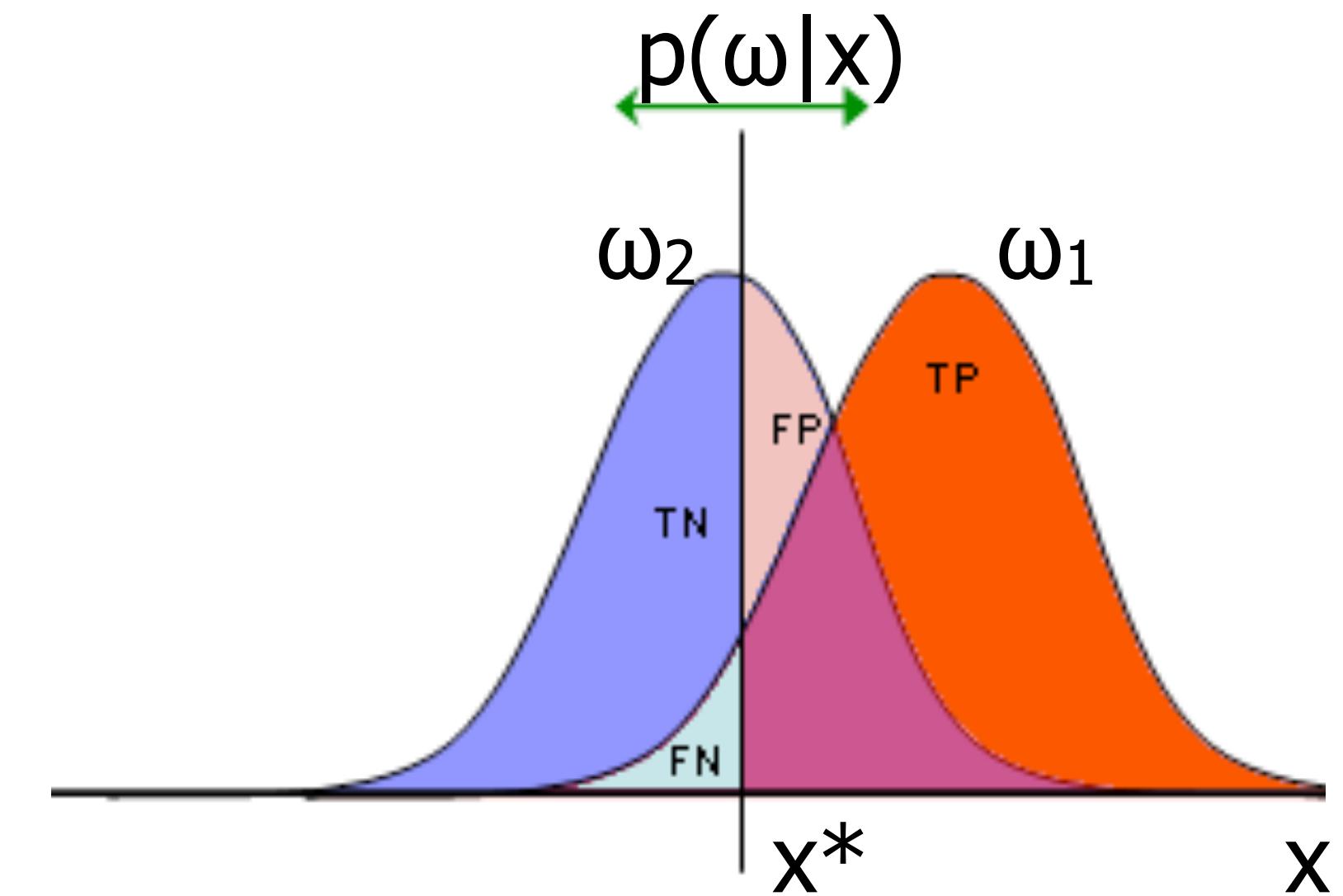
$N=1000$  con 20 eventi  $t=-1$  e 980 eventi  $t=+1$   
→ regola banale  $g(\mathbf{x}) = +1 \forall \mathbf{x}$  fornisce un accuracy del 98%!



# CONFUSION MATRIX

- nel contesto della classificazione binaria (2 classi), la confusion matrix viene utilizzata per avere una visione più completa delle prestazioni del modello di ML
- tiene in conto le differenti probabilità associate alle diverse decisioni che il classificatore può prendere

		Predicted class	
		+	-
Actual class	+	TP True Positives	FN False Negatives Type II error
	-	FP False Positives Type I error	TN True Negatives



- Permette di costruire varie metriche globali:

Metric	Formula	Interpretation
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Overall performance of model
Precision	$\frac{TP}{TP + FP}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{TP}{TP + FN}$	Coverage of actual positive sample

uso contemporaneo di precision e recall:  
si massimizzano entrambi (media armonica)

$$F_1 = 2 \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

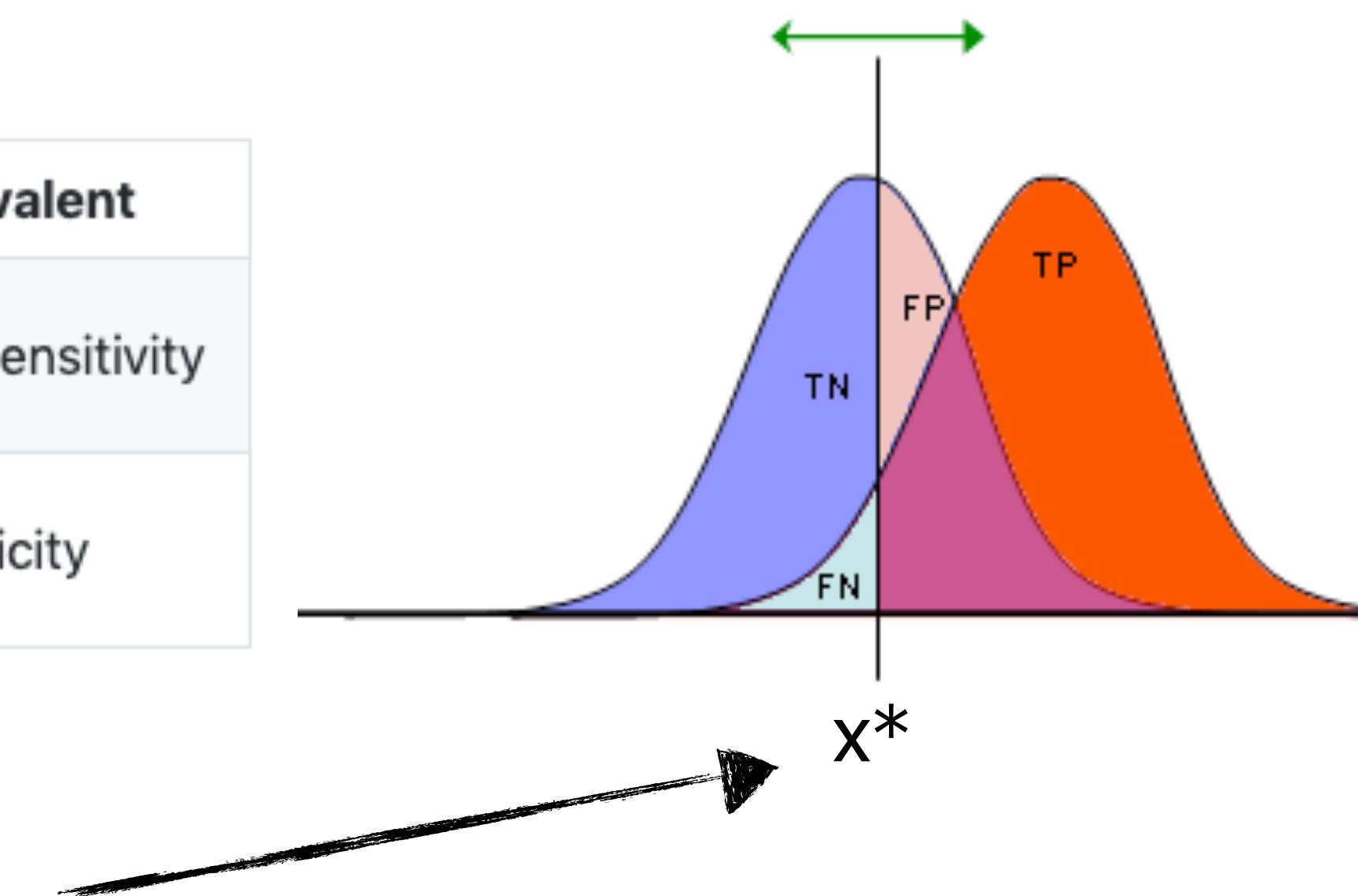
F1 score	$\frac{2TP}{2TP + FP + FN}$	Hybrid metric useful for unbalanced classes
----------	-----------------------------	---



# PIANO E CURVA ROC

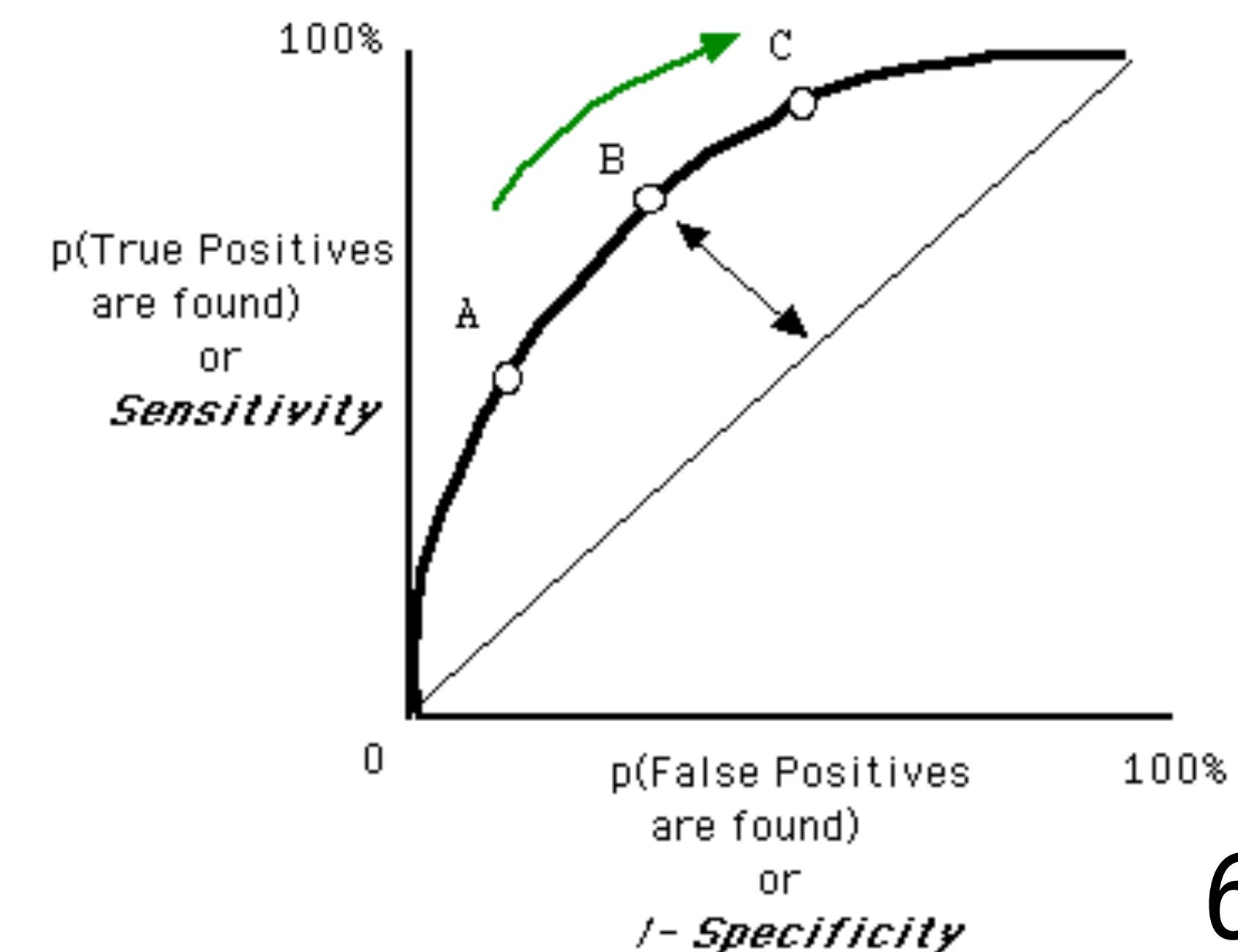
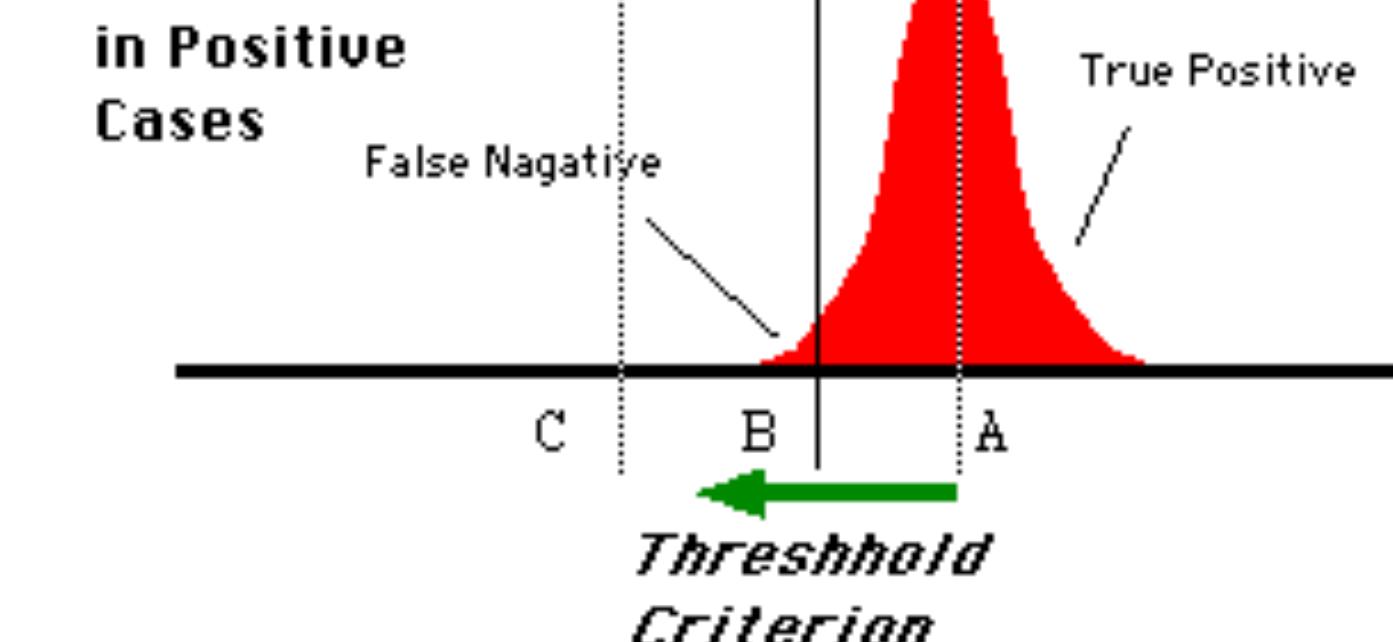
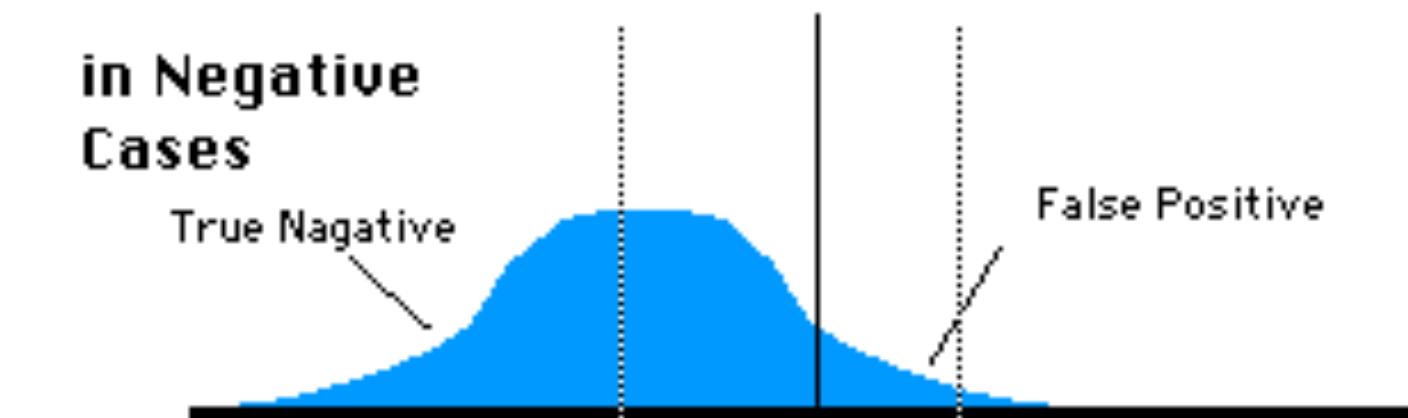
- receiver operating curve (ROC), è la curva che rappresenta TPR VS FPR al variare della soglia di decisione

Metric	Formula	Equivalent
True Positive Rate TPR	$\frac{TP}{TP + FN}$	Recall, sensitivity
False Positive Rate FPR	$\frac{FP}{TN + FP}$	1-specificity



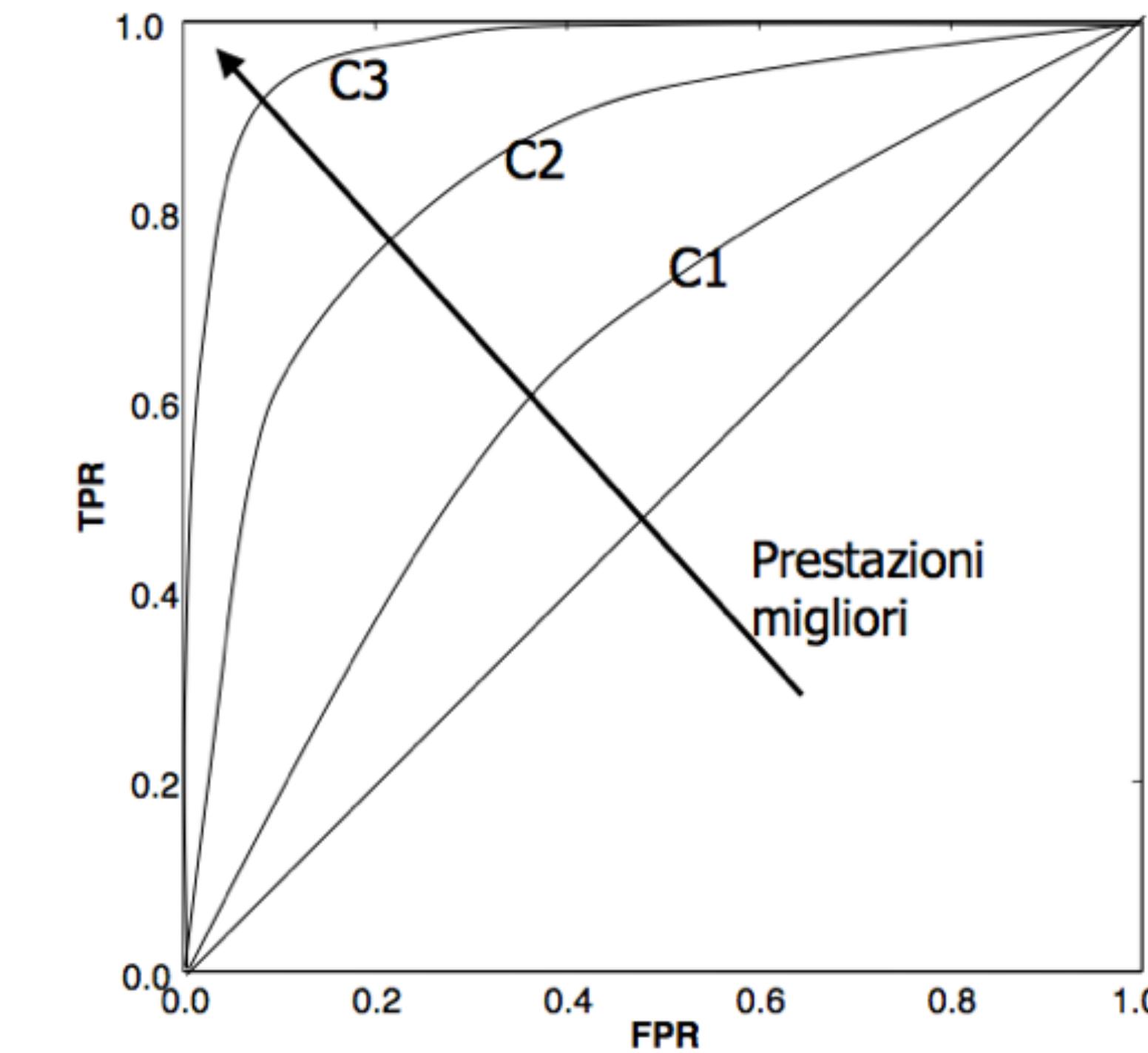
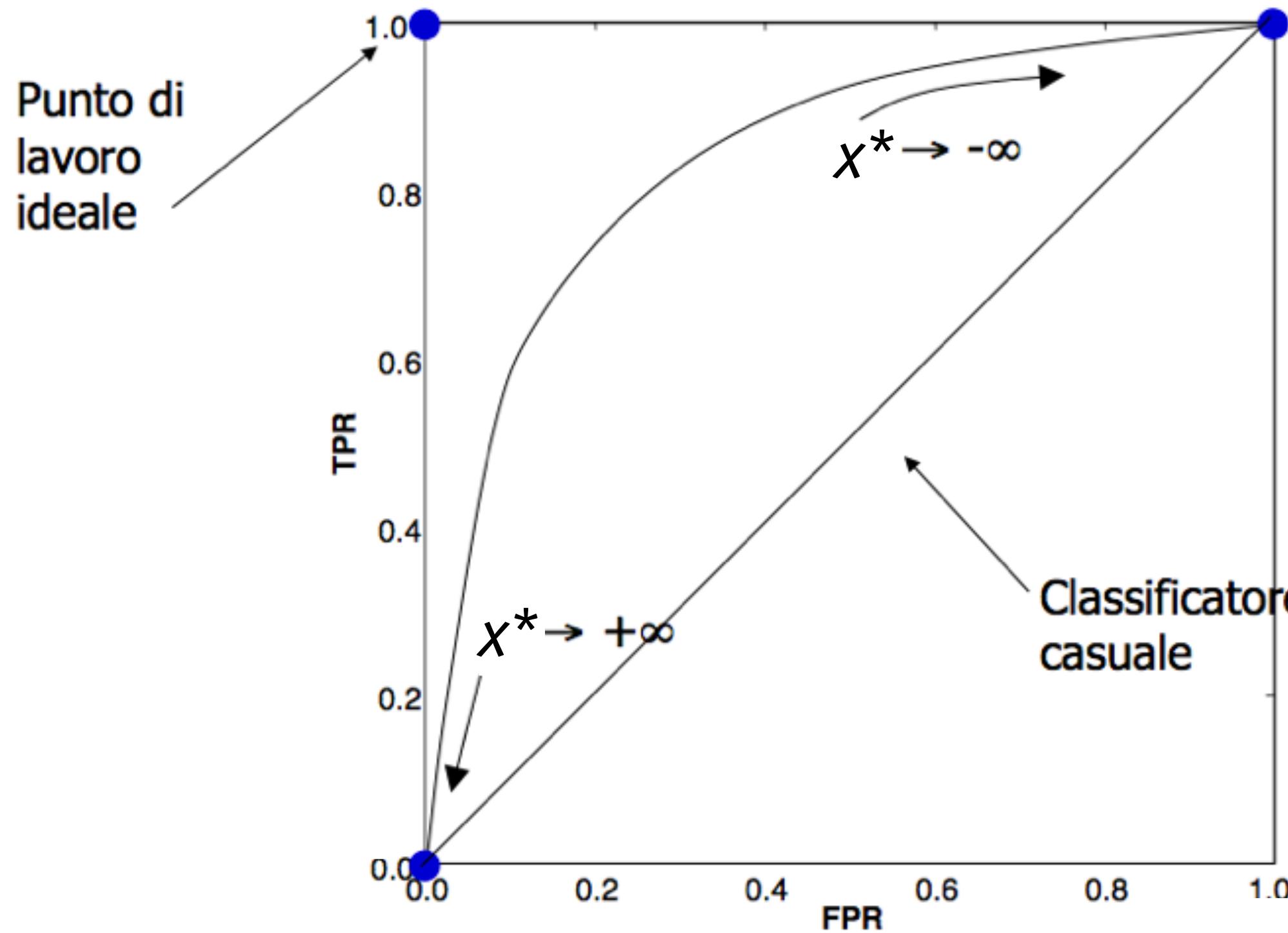
- $x^*$ : valore di soglia per la regola di classificazione

Distributions of the Observed signal strength



# CURVA ROC

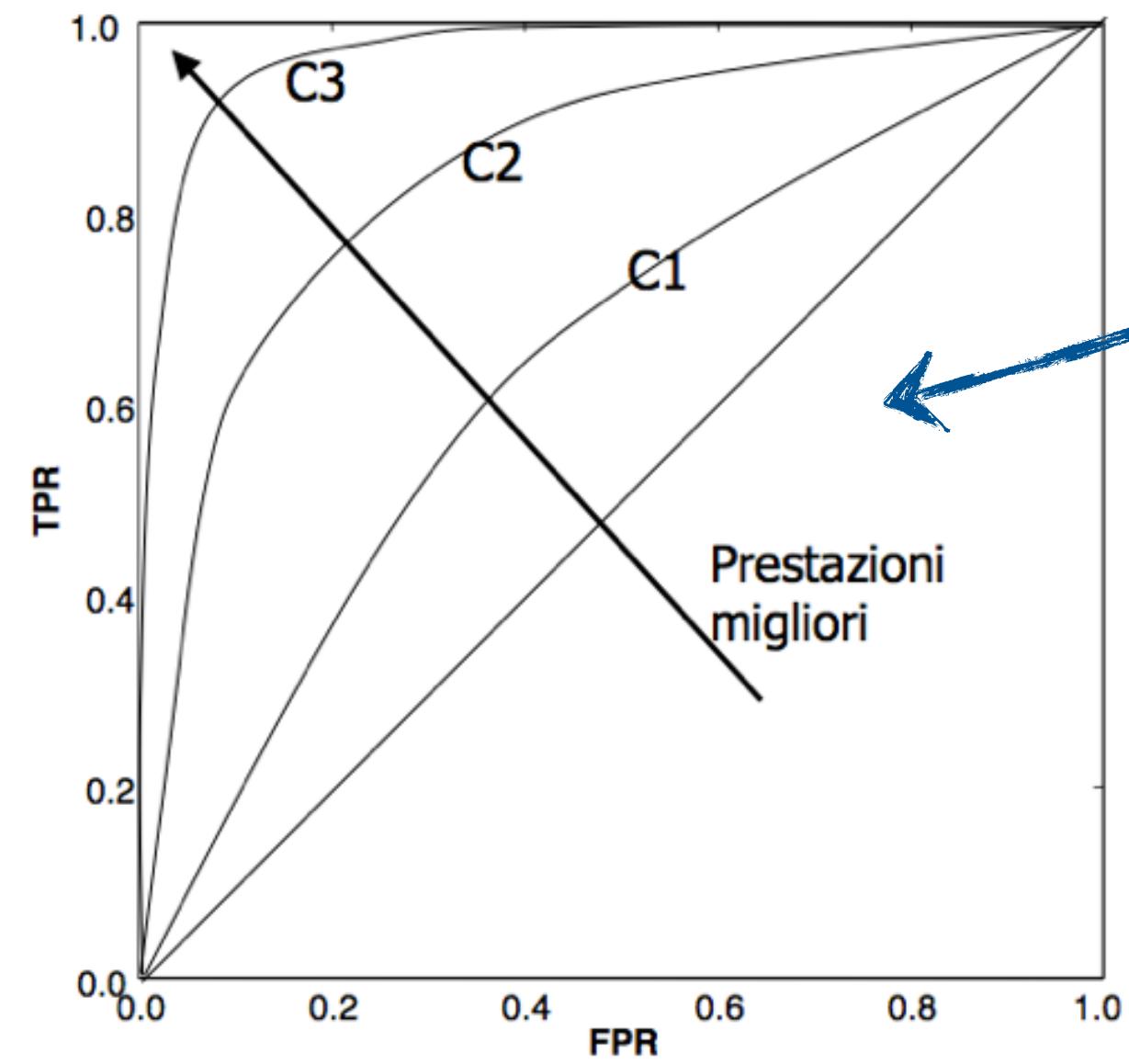
- la curva ROC ha concavità rivolta verso il basso
- è sempre al di sopra della retta  $TPR=FPR$  (classificatore casuale)



più la curva è spostata verso l'angolo in alto a sinistra,  
migliori sono le prestazioni del classificatore

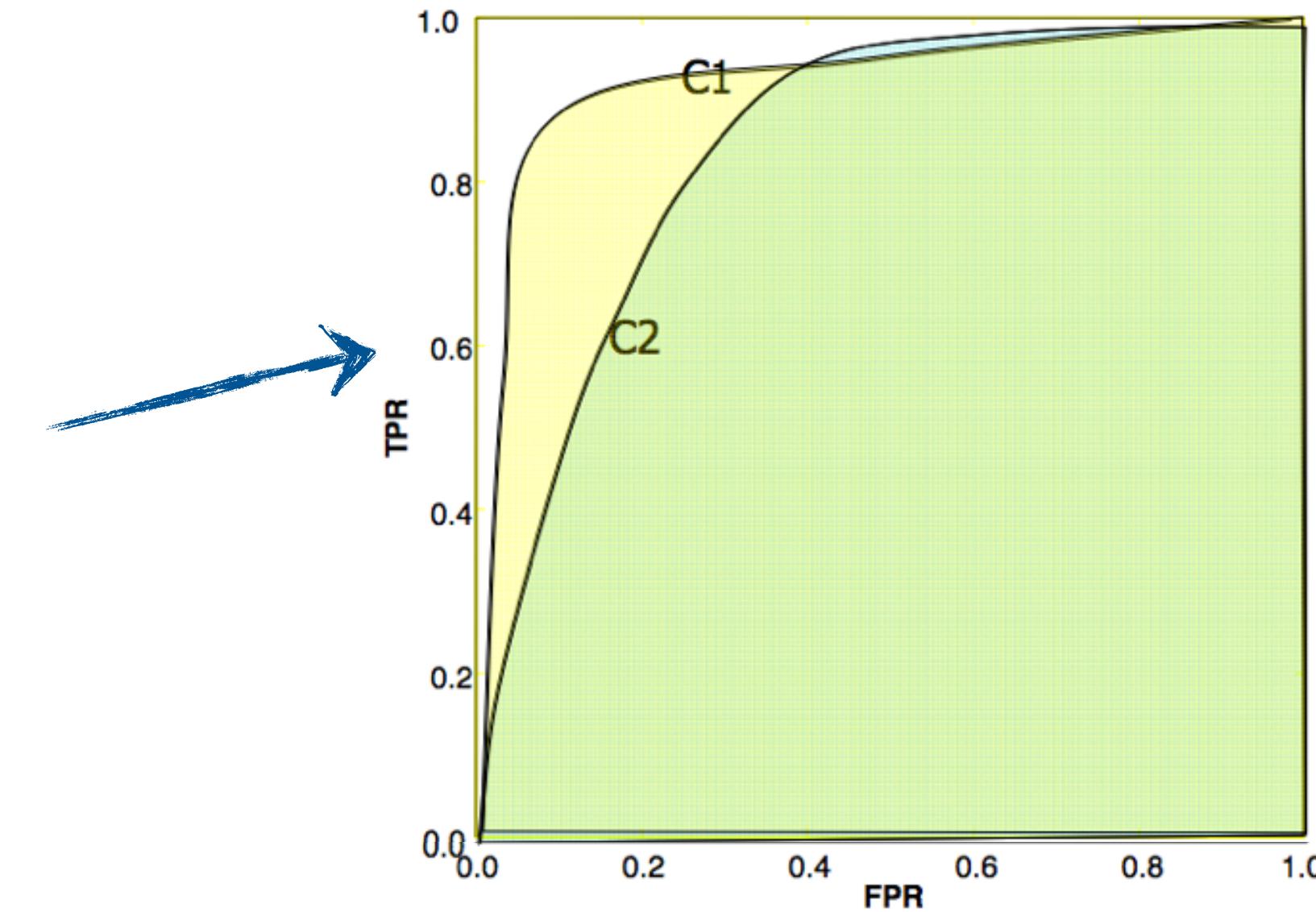
# AREA SOTTO LA CURVA ROC: AUC

- La curva ROC permette di stabilire facilmente quale sia il migliore algoritmo solo se è presente una chiara dominanza nell'intero spazio FPR



C3 migliore di C2 migliore  
di C1

in questo caso il  
confronto è invece  
ambiguo ...



in questi casi si confrontano gli algoritmi in termini dell'area sottesa alla curva ROC (AUC)

- AUC è compreso tra 0.5 (random classifier) e 1.0 (ideal/perfect classifier)
- si può dimostrare che AUC fornisce la probabilità di corretto ordinamento (i.e. la probabilità che il risultato del classificatore applicato ad un evento random del campione della classe 1 (ipotesi positiva) sia maggiore di quella ottenuta applicando il classificatore ad un evento random appartenente alla classe 2:  $P(TP) > P(FP)$ )

# LOGARITMIC LOSS

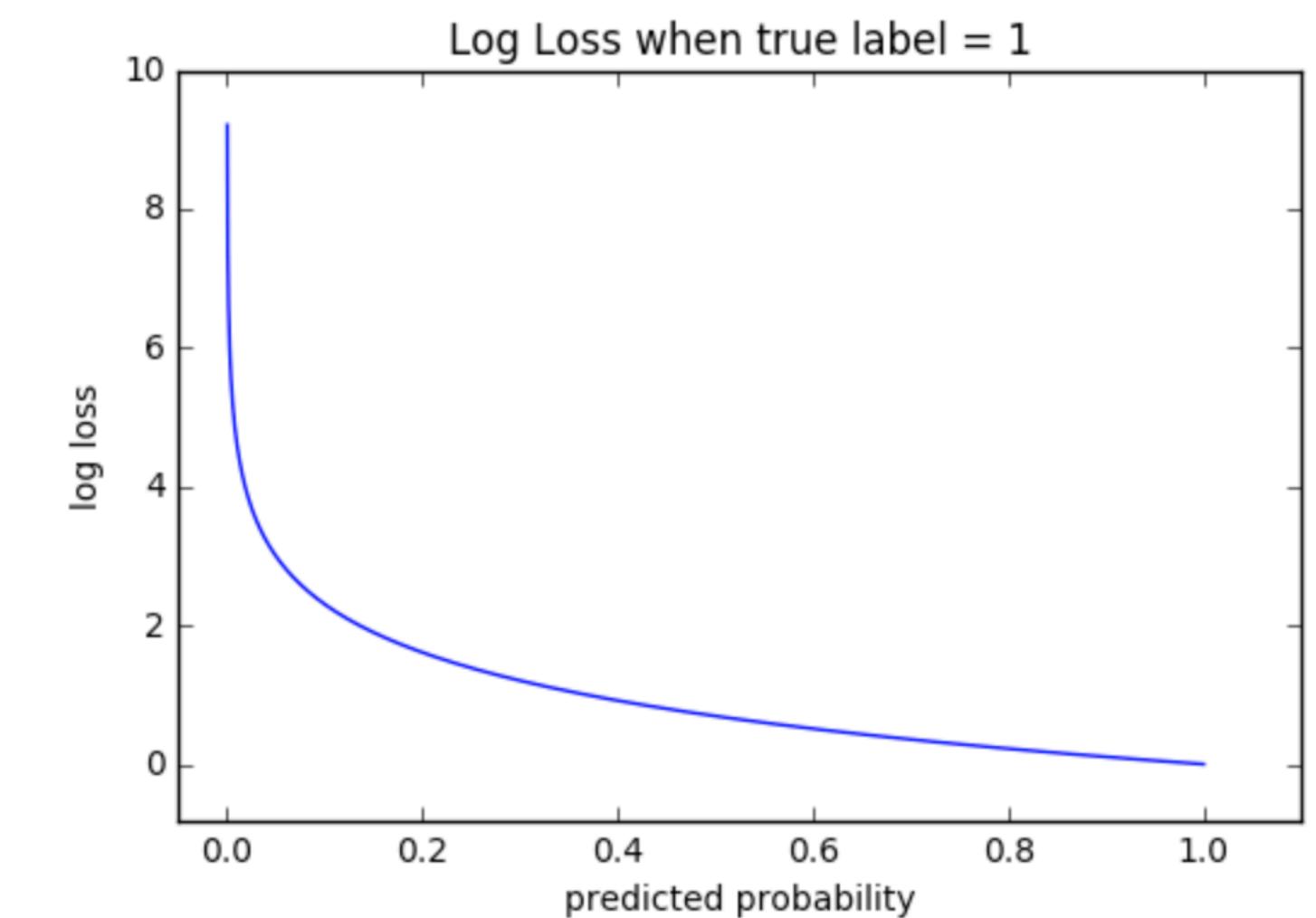
- AUC tiene in conto il corretto ordering delle probabilità ma non la capacità del modello di predire probabilità più grandi per campioni che abbiano maggiore probabilità di appartenere alla classe positiva
- il problema è risolto dalla logloss: una delle metriche basate sulle probabilità più utilizzate

- $y_i$ : label dell'evento i: 1 se l'evento i appartiene alla classe 1, 0 classe 2
- $p_i$ : probabilità che l'evento i appartenga alla classe 1 ← predizione dell'algoritmo di classificazione
- $(1-p_i)$ : probabilità che l'evento i appartenga alla classe 2

$$LogLoss = -\frac{1}{N} \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log((1 - p_i))]$$

• penalizza le classificazioni sbagliate:

$[0, \infty)$  vicino a zero → grande accuracy



legata alla cross-entropy (Kullback–Leibler divergence), una grandezza usata in teoria dell'informazione per valutare quanto simili sono due distribuzioni di probabilità

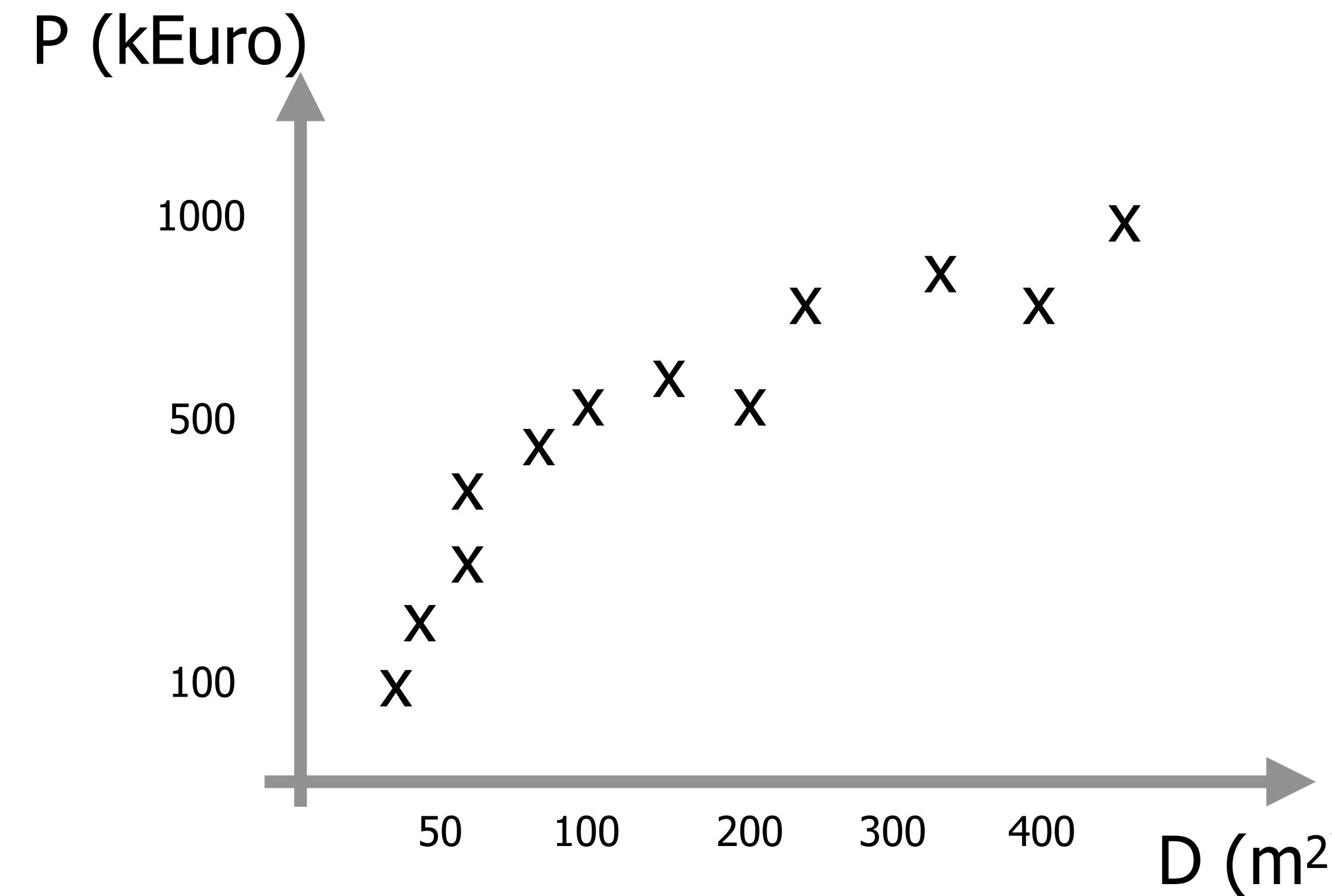


# UN ESEMPIO DI ALGORITMO DI REGRESSIONE

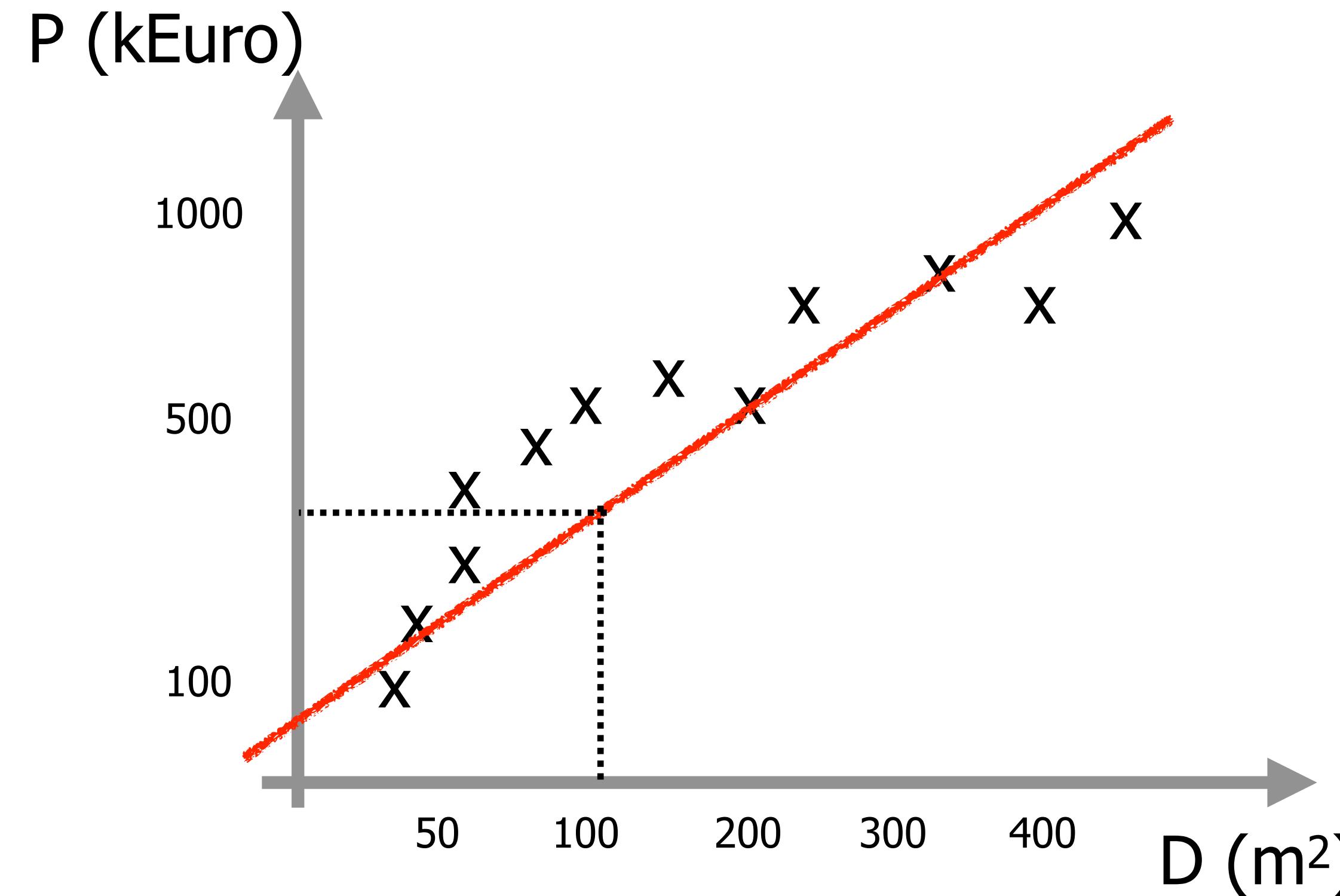
- un software che predice il prezzo di vendita di un'appartamento in un quartiere di Roma a partire dalle dimensioni di questo

dati a disposizione (esempio):

Un certo numero di casi di appartamenti venduti per i quali sia noto prezzo di vendita (P) e dimensione (D) in  $\text{m}^2$



# UN ESEMPIO DI ALGORITMO DI REGRESSIONE



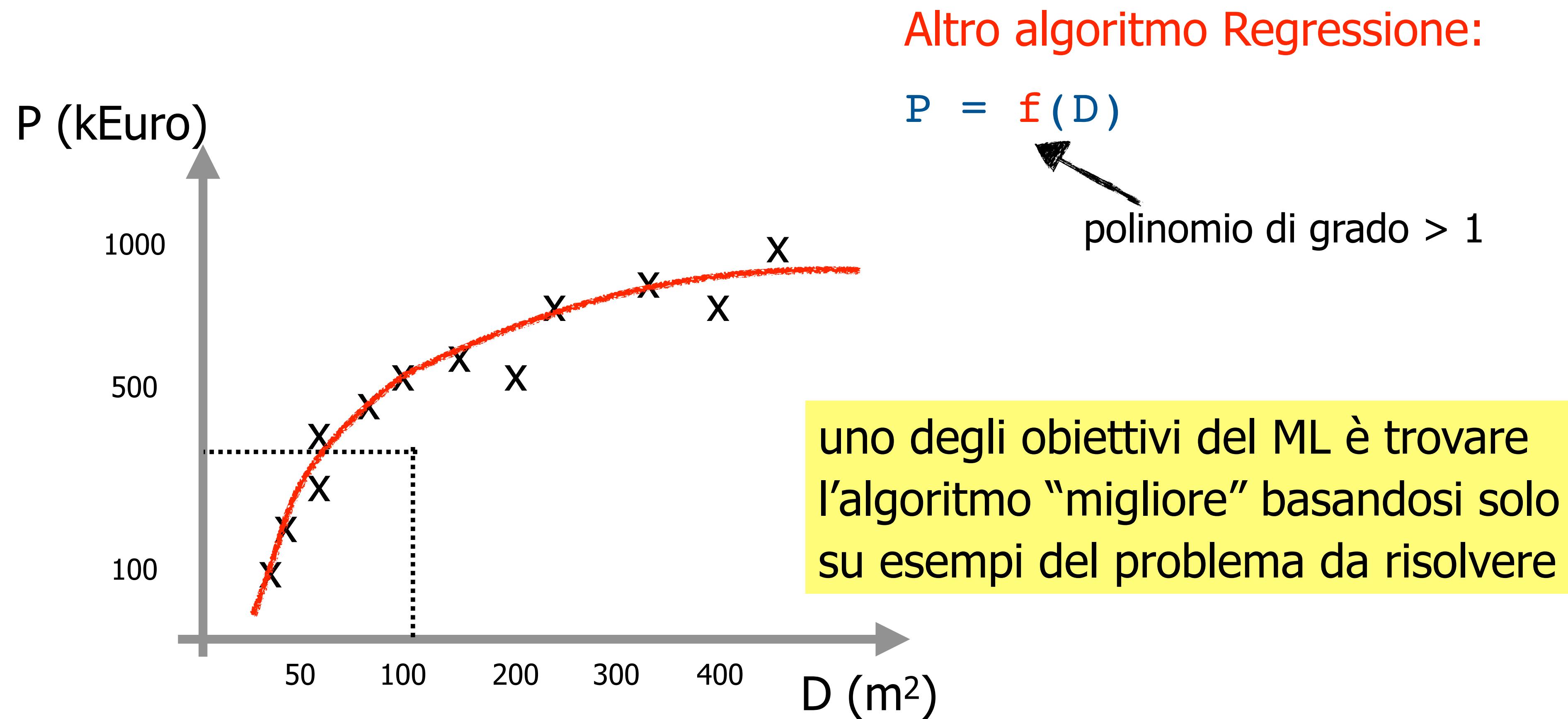
Algoritmo Regressione:

$$P = a_0 + a_1 * D$$

produce un uscita a  
valori continui

# UN ESEMPIO DI ALGORITMO DI REGRESSIONE

- si può pensare di fare meglio ...



# METRICHE USATE IN ALGORITMI DI REGRESSIONE

Le più utilizzate: MSE e MAE

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$

vantaggio: meno sensibile agli outlier  
svantaggio: non differenziabile in zero

interpretazione in termini di massima verosimiglianza (MLE) nel caso di classificatori lineari:  
corrispondono all'estimatore MLE su I parametri del modello assumendo prior Gaussiani (MSE) o di Laplace (MAE) sulle incertezze del modello



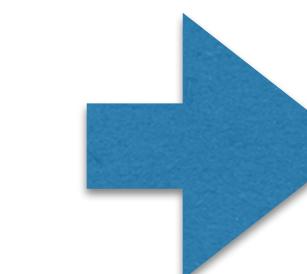
## • ALTRE METRICHE USATE IN ALGORITMI DI REGRESSIONE:

<b>Total sum of squares</b>
$SS_{tot} = \sum_{i=1}^m (y_i - \bar{y})^2$

media VS valore  
vero

<b>Residual sum of squares</b>
$SS_{res} = \sum_{i=1}^m (y_i - f(x_i))^2$

predizione  
mofrllo VS valore  
vero



$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

**Coefficiente di determinazione**

fornisce una misura di quanto bene il modello funziona rispetto alla baseline

0 = baseline

>> 0 meglio della baseline

Mallow's Cp	AIC	BIC	Adjusted $R^2$
$\frac{SS_{res} + 2(n+1)\hat{\sigma}^2}{m}$	$2[(n+2) - \log(L)]$	$\log(m)(n+2) - 2\log(L)$	$1 - \frac{(1-R^2)(m-1)}{m-n-1}$

con  $m$  dimensione del campione,  $n$  dimensione del vettore di feature,  $L$  likelihood, e  $\sigma^2$  stima della varianza associata ad ogni predizione dell'algoritmo



# CONFRONTO TRA FEATURE: SEPARATION POWER

- campione di eventi da n differenti categorie (classi)
- ogni categoria contribuisce al totale con una frazione  $f_i$ :  $\sum f_i = 1$
- $x$ : feature vector con densità di probabilità:  $p_{tot}(x|f) = \sum_{i=1,n} f_i p_i(x)$   $p_i(x)$  distribuzione di probabilità di  $x$  per eventi della classe  $i$

Possiamo usare l’“incertezza” sulle  $f_i$  come “misura” del potere di separazione della feature  $x$  ...

$$\begin{aligned} [cov(f_i, f_j)]^{-1} &\geq -E \left[ \frac{\partial^2 \log L}{\partial f_i \partial f_j} \right] = \text{minimum variance bound} \\ &= \int \cdots \int -\frac{\partial^2}{\partial f_i \partial f_j} \left( \sum_{k=1}^N \log p(x_k|f) \right) \prod_{l=1}^N p(x_l|f) dx_l = \\ &= N \int -p(x|f) \frac{\partial^2}{\partial f_i \partial f_j} \log p(x|f) dx = \frac{1}{N} \left[ \int \frac{[p_j(x) - p_n(x)] \cdot [p_i(x) - p_n(x)]}{p(x|f)} dx \right]_{ij}^{-1} \end{aligned}$$

$$\sigma^2(f) = \frac{1}{N} \left[ \int \frac{[p_S(x) - p_B(x)]^2}{fp_S(x) + (1-f)p_B(x)} dx \right]^{-1}$$

per  $n=2$  classi c’è  
un’unica frazione  $f$

$$SepPower = \frac{\sigma_{best}(f)}{\sigma(f)} = \sqrt{f(1-f) \int \frac{(p_S(x) - p_B(x))^2}{p(x|f)} dx}$$



# METODI DI OTTIMIZZAZIONE

- L'apprendimento nella maggior parte degli algoritmi di ML comprende come passo fondamentale la minimizzazione (o massimizzazione) di una **funzione obiettivo (loss function)** dipendente dai parametri interni del modello e che vengono utilizzati per predire il target ( $y$ ) dall'insieme dei predittori (vettori di feature  $x$  del campione di esempi) utilizzati dal modello
- differenti strategie di ottimizzazione adottate per addestrare in modo efficace e efficiente i diversi modelli ed ottenere predizioni accurate
- Due classi di algoritmi di ottimizzazione principalmente negli algoritmi moderni di ML:
  - **ottimizzatori non-basati sul metodo della discesa lungo il gradiente:** metodi Monte Carlo, algoritmi genetici, annealing simulato ...
  - **ottimizzatori basati sul metodo della discesa lungo il gradiente:**
    - first-order algorithms: massimizzano o minimizzano la loss function utilizzando i valori del gradiente rispetto ai parametri del modello
    - second-order algorithms: utilizzano le derivate del secondo ordine (Hessiano) nella minimizzazione della loss function. Le derivate del secondo ordine forniscono una approssimazione quadratica sulla curvatura della funzione dei loss, e quindi prestazioni migliori al costo di maggiore complessità computazionale



# METODI NON GRADIENT-BASED

- Minimizzazione analitica:
  - esatta, raramente si può fare se non in alcuni casi semplici
- Campionamento Monte Carlo:
  - campionamento random dell'intero spazio delle variabili di input (priors: uniformi, oppure gaussiani)
  - converge e non è distorto (trova sempre il minimo giusto), inefficiente (specie al crescere del numero di dimensioni (curse of dimensionality))
- Algoritmi genetici:
  - trovano soluzioni approssimate a problemi di ottimizzazione, modellando il problema con un set (popolazione) di rappresentazioni astratte (genomi) delle possibili soluzioni, ed applicando schemi evolutivi tipici delle teorie genetiche: ereditarietà, mutazioni, crossover
  - solo le popolazioni più “forti” sopravvivono (fitness function), algoritmi molto veloci ed efficienti che possono essere applicati a problemi di ottimizzazione sia non-vincolati che vincolati, e in cui le loss function possono essere discontinue, non-differenziabili, stocastiche o altamente non-lineari

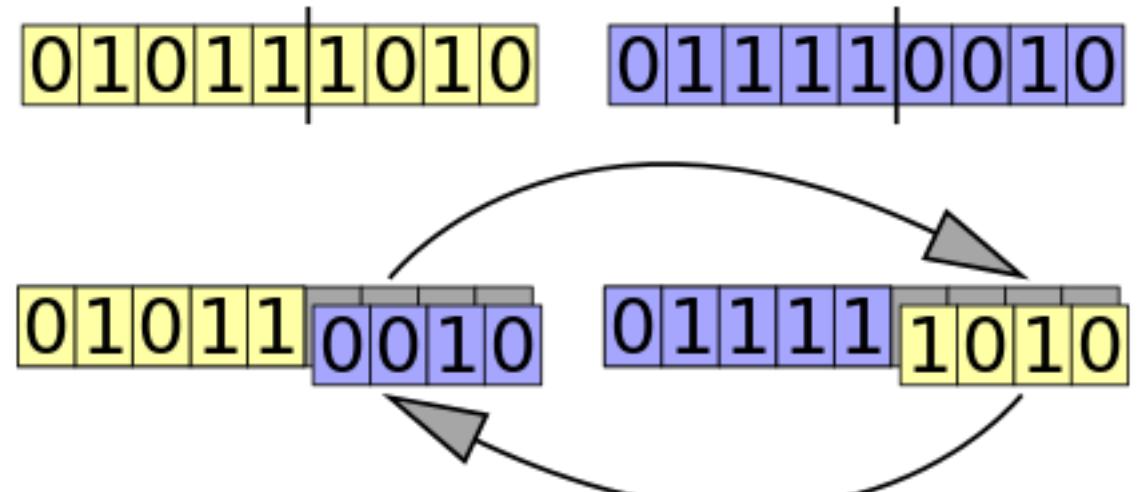


# ALGORITMI GENETICI

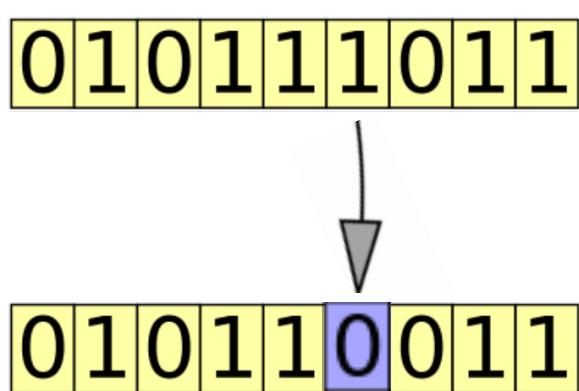
Un tipico **algoritmo** genetico, nel corso della sua esecuzione, provvede a fare evolvere delle soluzioni secondo il seguente schema di base:

1. Generazione casuale della prima popolazione di soluzioni (cromosomi).
2. Applicazione della **funzione** di *fitness* alle soluzioni (cromosomi) appartenenti all'attuale popolazione.
3. Selezione delle soluzioni considerate migliori in base al risultato della funzione di fitness e della logica di selezione scelta.
4. Procedimento di crossover per generare delle soluzioni ibride a partire dalle soluzioni scelte al punto 3.
5. Creazione di una nuova popolazione a partire dalle soluzioni identificate al punto 4.
6. Riesecuzione della procedura a partire dal punto 2 ed utilizzando la nuova popolazione creata al punto 5.

esempio: crossover ad un punto

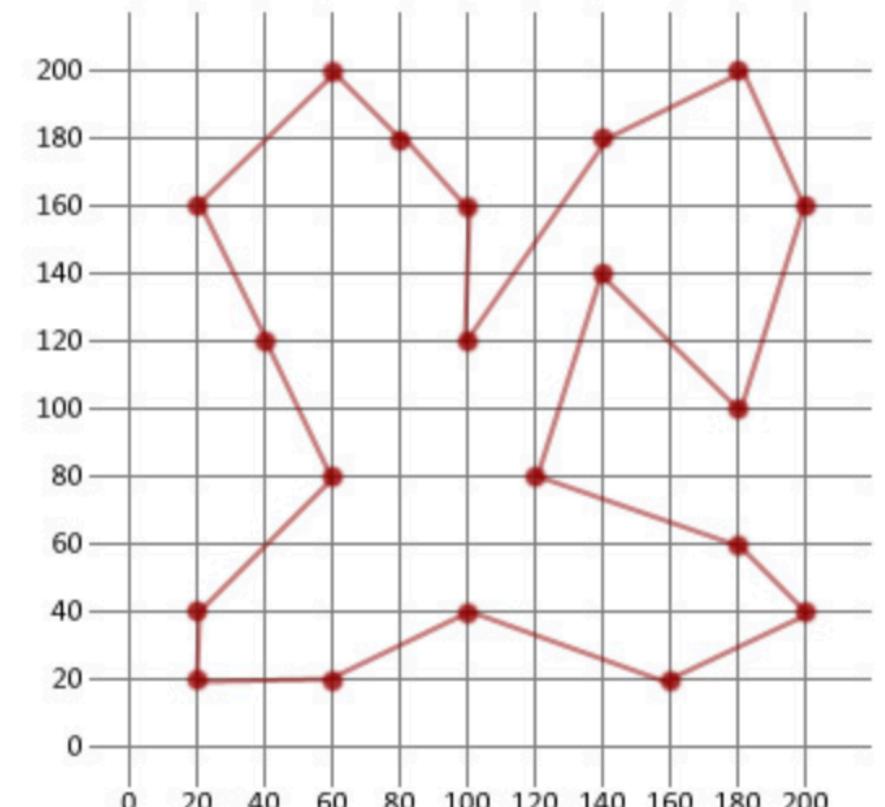
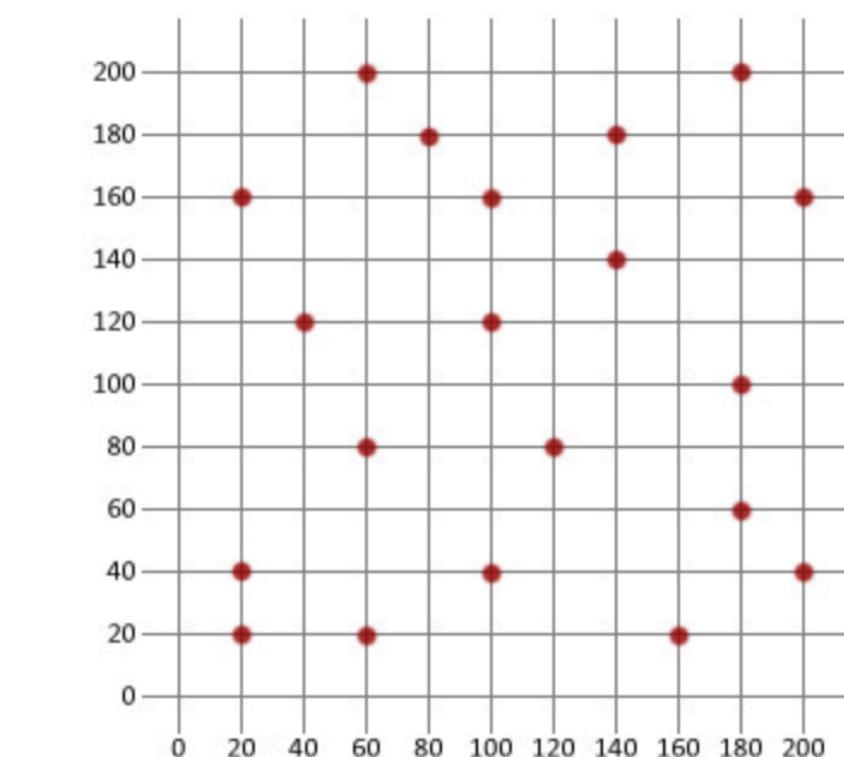


esempio: mutazione ad un bit



esercizio x casa:

provate a risolvere il problema del **traveling salesman** con un algoritmo genetico



# METODI NON GRADIENT-BASED

- Simulated Annealing (ricottura):
  - l'algoritmo sfrutta l'analogia tra il modo in cui un metallo si raffredda ottenendo un minimo energetico una volta raggiunta la cristallizzazione, ed il problema della ricerca di un minimo assoluto in un sistema generico
  - il sistema viene riscaldato, introducendo una randomizzazione ad ogni componente. Come risultato ogni variabile può temporaneamente assumere anche valori energeticamente sfavorevoli, con probabilità che diminuiscono mano a mano che il calcolo procede (i.e. la temperatura decresce) → inizialmente variazioni random in tutte le direzioni, alla fine solo verso il minimo
    - vantaggio principale rispetto ad altri metodi: abile nell'evitare di essere intrappolato in minimi locali
    - funziona male in configurazioni patologiche “golf course”: regione di minima energia piccola e circondata da configurazioni energeticamente sfavorevoli



# ANNEALING SIMULATO

## Algorithm 1 Simulated Annealing Pseudocode

---

```

1: procedure SIMANNEAL( $m, iter_{max}, T$ )
2:    $x_{curr} \leftarrow InitialConfig(m)$                                  $\triangleright$  Select some  $x \in S$ 
3:    $x_{best} \leftarrow x_{curr}$ 
4:   for  $i = 1$  to  $iter_{max}$  do
5:      $x_{prop} \leftarrow NeighbourConfig(x_{curr})$                        $\triangleright$  Propose some neighbour configuration
6:      $temp_{curr} \leftarrow CalcTemp(i, T)$                                  $\triangleright$  Anneal system
7:     if  $Cost(x_{prop}) \leq Cost(x_{current})$  then
8:        $x_{curr} \leftarrow x_{prop}$ 
9:       if  $Cost(x_{prop}) \leq Cost(x_{best})$  then
10:         $x_{best} \leftarrow x_{prop}$ 
11:      end if
12:      else if  $\exp\left\{\frac{Cost(x_{curr}) - Cost(x_{prop})}{temp_{curr}}\right\} > Random(0, 1)$  then     $\triangleright$  Accept upward step?
13:         $x_{curr} \leftarrow x_{prop}$ 
14:      end if
15:    end for
16:    return  $x_{best}, Cost(x_{best})$                                  $\triangleright$  Best configuration and associated cost
17: end procedure

```

---

- Si sceglie la  $T$  iniziale e si crea una soluzione iniziale random
- Si fa un loop fino al raggiungimento di una condizione di stop prefissata
- Si seleziona  $S^*$  vicino a  $S$
- Si decide se accettare la soluzione
- Si diminuisce la temperatura e si ripete

soluzioni cattive sono accettate con prob.

$$e^{\frac{E(s) - E(s^*)}{T}}$$

più facile ad alta  $T$

