Machine Learning

Corso AI Engineering - Lezione 2

Reti

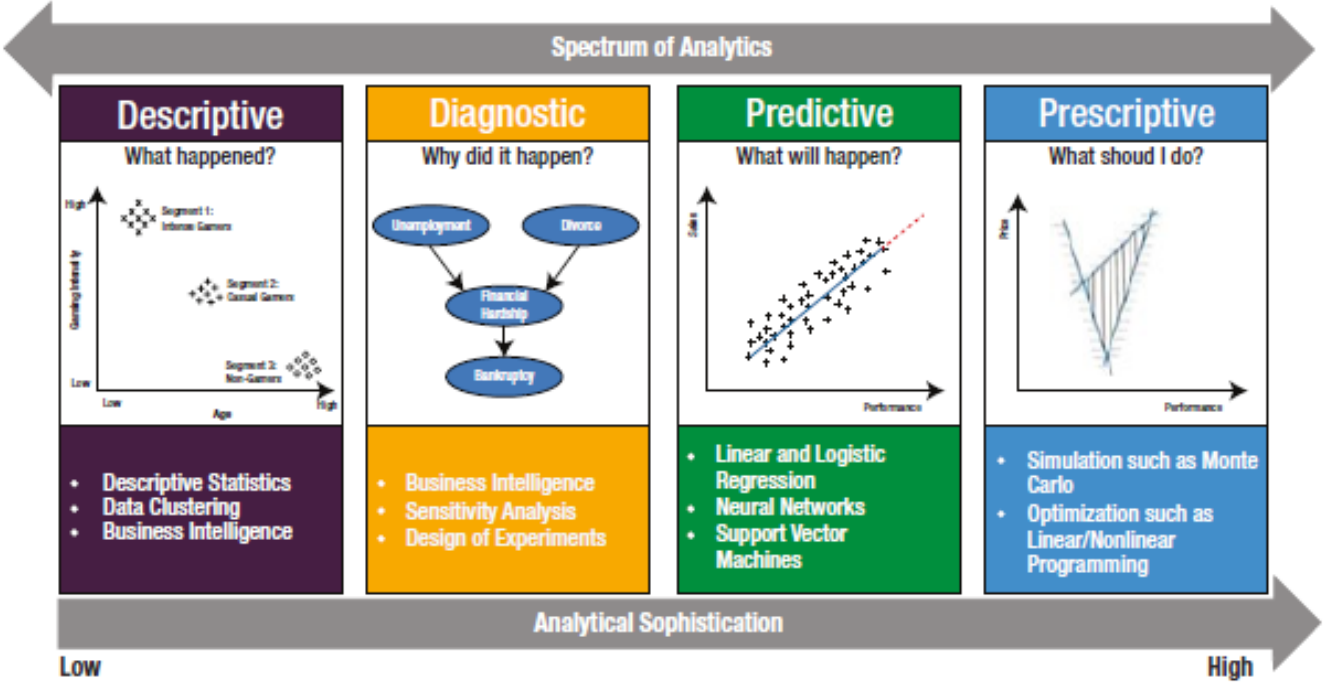
INDICE

MACHINE LEARNING

- ✓ Cosa è il Machine Learning?
- ✓ Quando usare il Machine Learning?
- ✓ Supervised Learning
- ✓ Unsupervised Learning
- ✓ Reinforcement Learning
- ✓ Tecniche di Machine Learning

COS'È IL MACHINE LEARNING?

- **Machine Learning** è una branca dell'intelligenza artificiale e basa le sue fondamenta sul principio di **apprendimento automatico a partire da dati**, identificando autonomamente modelli con un intervento umano ridotto.
- ML comprende un insieme di tecniche di definizione automatica di modelli analitici/predittivi



QUANDO USARE IL MACHiNE LEARNING?

Risorse

- Domande e obiettivi ben posti
- Dati pertinenti
- Dati accurati
- Dati coerentemente connessi
- Tanti dati (*size matters*)

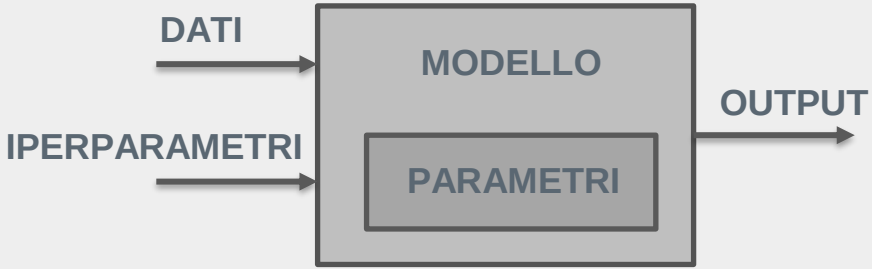
Problematiche

- **Troppo complesso**: non si riesce a formalizzare e programmare.
- **Troppo grande**: non si riesce a scalare la soluzione.
- **Troppo specifico**: la soluzione deve essere adattabile e personalizzabile.
- **Automatismo**: problemi a tenere traccia della soluzione manualmente, necessario automatismo.

Modello, Parametri e Iperparametri

Un **modello** è una "funzione" che mette in relazione i dati in input con l'output desiderato. Un modello può essere costituito da una semplice funzione matematica ma anche da algoritmi di elaborazione più complessi.

Un **parametro del modello** è un valore calcolato internamente al modello a partire dai dati.

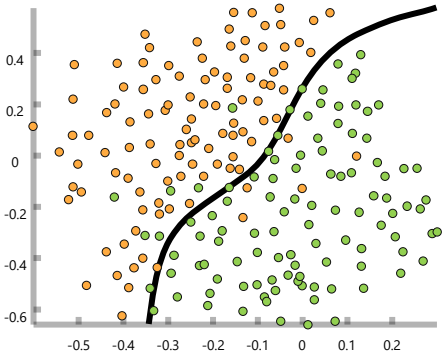


Un **iperparametro del modello** è un parametro esogeno al modello.
Modificando gli iperparametri in input al modello è possible eseguire un **tuning** del modello per migliorarne l'efficacia o migliorarne il processo di apprendimento

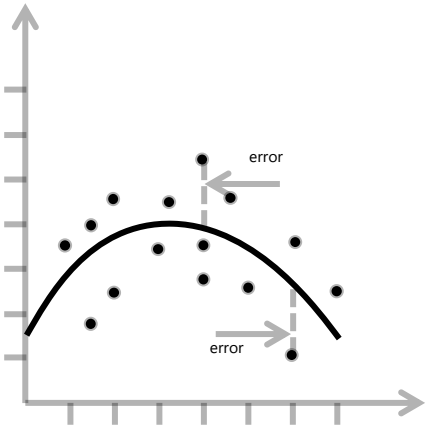
POSSIBILI CAMPI DI APPLICAZIONE

Cosa piace ai miei clienti?

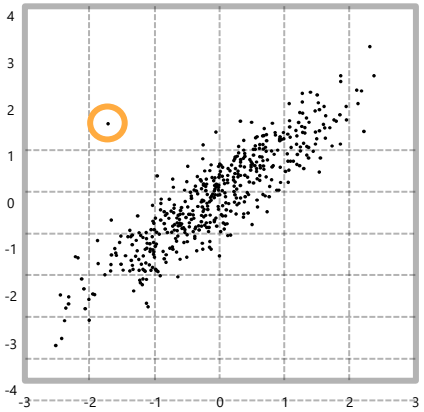
Chi è intenzionato a comprare il mio prodotto?



Quanto cresceranno gli investimenti?



C'è qualche errore?



Supervised

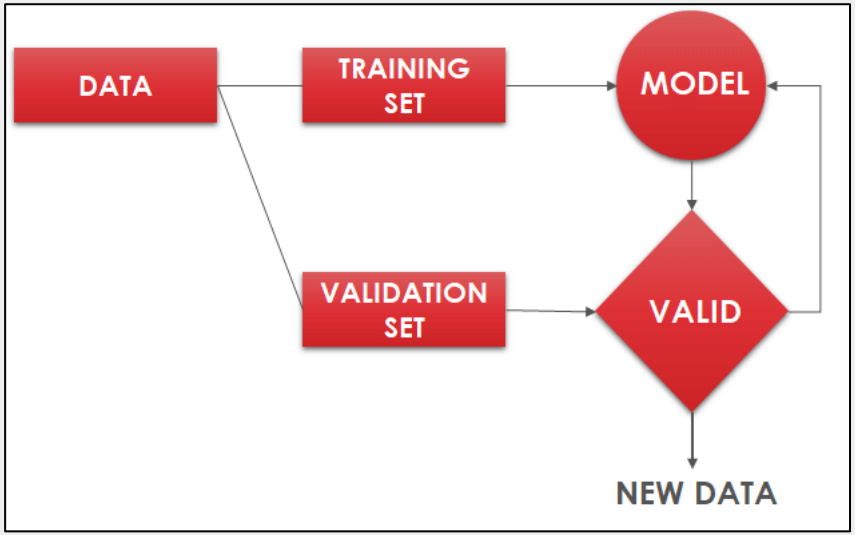
Unsupervised

Reinforcement

Supervised Learning

Nel supervised learning abbiamo degli **esempi di coppie input-output realmente misurate,** che possono essere usate come guida per l'algoritmo di apprendimento supervisionato durante la fase di training.

Dividendo in due parti il dataset originario possiamo usarne una parte per addestrare il modello e la seconda per **validare se il modello stima correttamente i valori richiesti**.



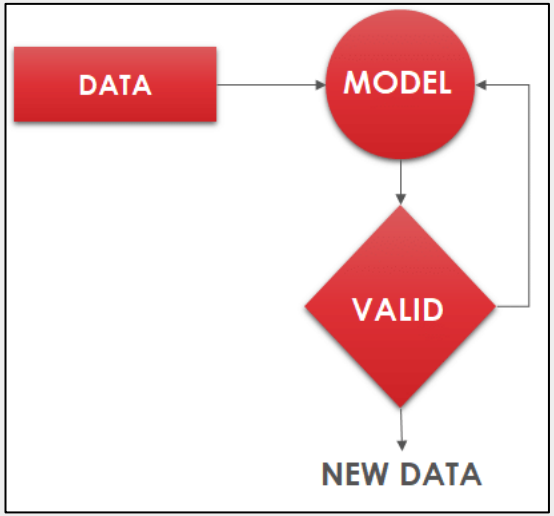
Il supervised learning si divide sostanzialmente in due gruppi:

1. **Regression:** vogliamo ottenere una stima o una prediction di una quantità continua.
2. **Classification:** vogliamo dividere le osservazioni in gruppi o categorie.

Unsupervised Learning

Si parla di **unsupervised learning** se non abbiamo dati sull'output desiderato. L'obbiettivo principale in questo caso è **trovare dei pattern** all'interno dei dati disponibili.

Per validare questi tipi di modelli è necessario usare delle **tecniche di valutazione della qualità dei pattern** riconosciuti. Per questo motivo spesso si ha bisogno di ricorrere a riparametrizzazioni del modello prima di ottenere dei risultati accettabili.



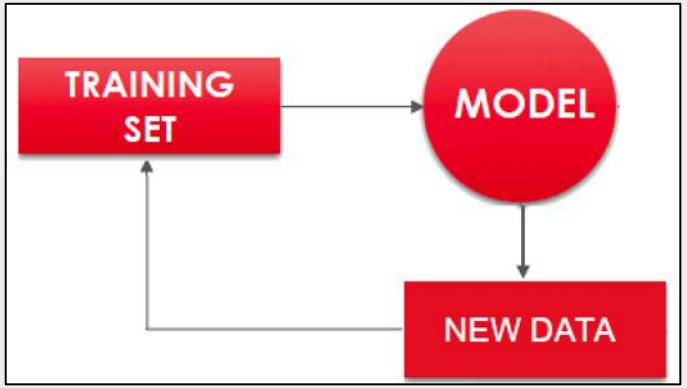
L'unsupervised learning comprende diverse tipologie di problemi, tra cui:

1. **Clustering:** vogliamo raggruppare le osservazioni in un certo numero di gruppi secondo una misura di similarità.

2. **Generative models:** vogliamo generare nuovi dati coerenti al dataset di partenza.

Reinforcement Learning

In maniera similare al unsupervised learning, nel **reinforcement learning** non forniamo all'algoritmo degli esempi da cui apprendere, ma gli forniamo **un metodo per la valutazione della qualità della soluzione ottenuta,** portandolo a massimizzare tale valore.



Il reinforcement learning viene utilizzato in particolare in questi casi:

1. **Anomaly detection:** vogliamo monitorare la presenza di anomalie tramite delle metriche.

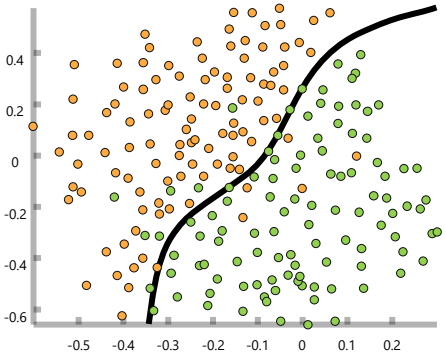
2. **Autonomous Agent:** vogliamo che un agente (software o hardware) esegua una efficientemente una azione in maniera autonoma.

CAMPI DI APPLICAZIONE

<u>Cosa</u> piace ai miei clienti?

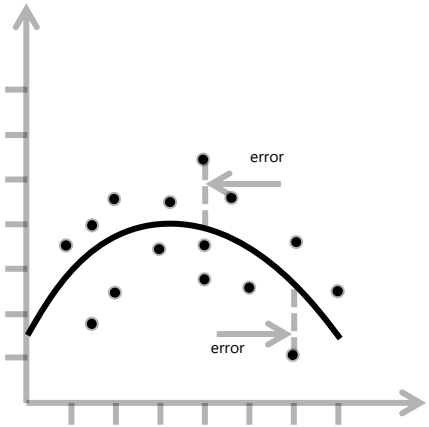
?

<u>Chi</u> è intenzionato a comprare il mio prodotto?



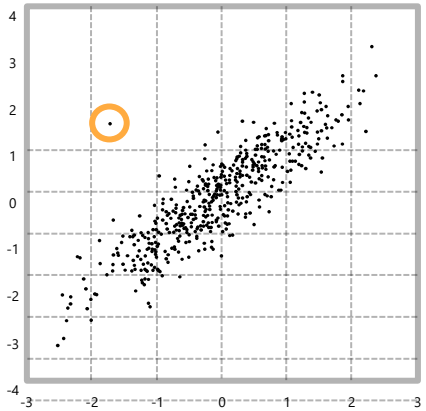
?

<u>Quanto</u> cresceranno gli investimenti?



?

C'è qualche errore?



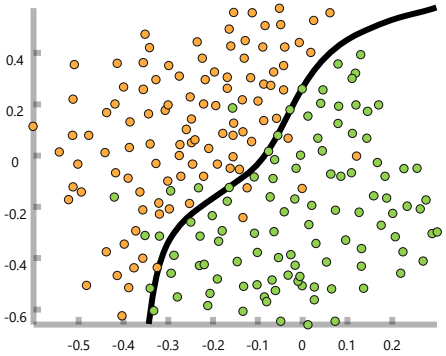
?

CAMPI DI APPLICAZIONE

<u>Cosa</u> piace ai miei clienti?

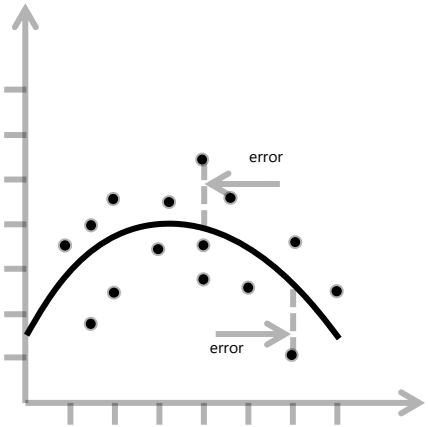
RECOMENDERS (CLUSTERING)

<u>Chi</u> è intenzionato a comprare il mio prodotto?



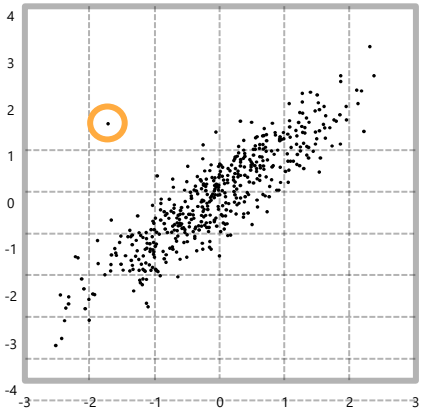
CLASSIFICAZIONE

<u>Quanto</u> cresceranno gli investimenti?



REGRESSIONE

C'è qualche errore?



ANOMALY DETECTION

TECNICHE DI MACHINE LEARNING

MODELLI

Focus on:

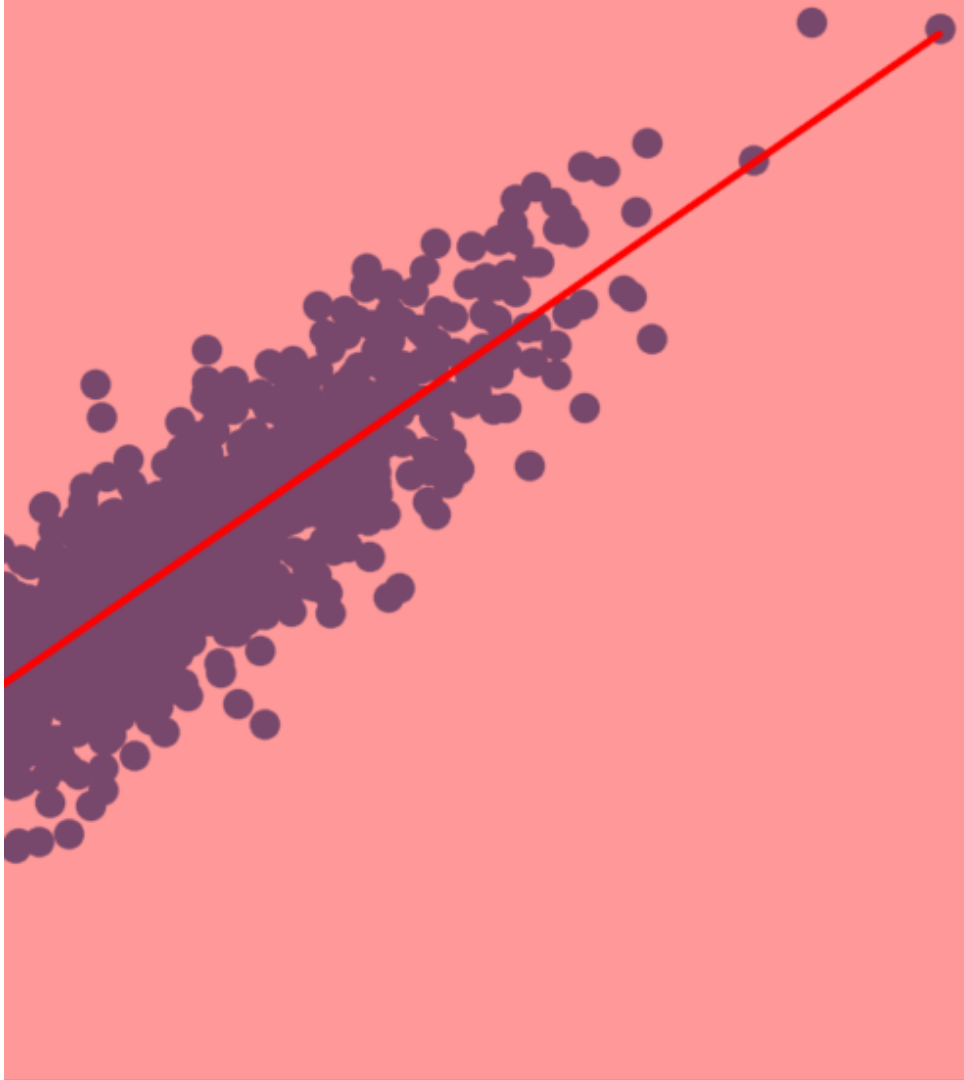
- ✓ Regressione
- ✓ Classificazione
- ✓ Alberi Decisionali e Random Forest
- ✓ Clustering

Regressione

Una regressione fornisce una relazione numerica tra due o più variabili presenti in un dataset.

Di conseguenza avremo:

- ✓ una variabile dipendente o **response variable** (y)
- ✓ una o più variabili indipendenti detti **regressori** o **predictor variables** o **features** $(x_1, x_2, x_3, \dots)$.



Regressione Lineare

Il più semplice modello di questa categoria è la regressione lineare.
In questo caso il modello è costituito da una *semplice* funzione matematica, $\mathbf{h_\theta(\cdot)}$.

- La regressione lineare produce buoni modelli quando si osserva una dipendenza lineare tra regressori (o trasformazioni di essi) e variabile dipendente.

Formulazione Matematica:

$$y = h_\theta(x_1, x_2, x_3, \dots, x_n) = \sum_{i=1}^{n} \theta_i x_i$$

ovvero

$$y = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n.$$

- $\mathbf{x} = [x_1, x_2, x_3, \dots, x_n]$ è il vettore di **regressori o feature**
- y è l'output del modello, la **response variable**
- $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3, \dots, \theta_n]$ è il vettore di **parametri del modello**
- Di conseguenza possiamo riscrivere il modello in termini *vettoriali:*

$$y = h_\theta(\boldsymbol{x}) = \boldsymbol{\theta x}^T.$$

Definizione dei parametri

La formula che determina l'output dipende dal vettore dei parametri $\boldsymbol{\theta}$. Tali valori vengono calcolati a partire dai dati in input.

Essendo un problema di supervised learning, abbiamo degli esempi di coppie $(y, \boldsymbol{x})$**:**

➔ Troviamo $\boldsymbol{\theta}$ in modo tale da **minimizzare le differenze** tra y e $\boldsymbol{\theta x}^T$.

Problema dei minimi quadrati:

Dato un training dataset $(\hat{y}, \hat{\boldsymbol{x}})$ con:

- N osservazioni
- F feature

i parametri del modello saranno determinati dalla soluzione del seguente problema di *ottimizzazione (o programmazione) quadratica:*

$$\text{minimize}_{\boldsymbol{\theta} \in \mathbb{R}} \sum_{i=1}^{N} \left(\hat{y}_i - \boldsymbol{\theta}\hat{\boldsymbol{x}}^{\boldsymbol{T}}\right)^2$$

Questi tipi di problemi vengono risolti con algoritmi iterativi (**gradient descent**) o con tecniche matematiche (risoluzione dell'**equazione normale**).

Altri esempi

La formula da minimizzare per calcolare i $\boldsymbol{\theta}$ prende il nome di **funzione obiettivo.**

Cambiando la funzione obiettivo, i valori di $\boldsymbol{\theta}$ saranno potenzialmente diversi, di conseguenza si avrà un modello di regressione differente

➔ Cambiare la funzione obiettivo porta a regressioni con proprietà peculiari.

Lasso(α):

$$\text{minimize}_{\boldsymbol{\theta}\in\mathbb{R}} \frac{1}{2N}\sum_{i=1}^{N}(\hat{y}_i - \boldsymbol{\theta}\hat{\boldsymbol{x}}^{\boldsymbol{T}})^2 + \alpha\sum_{f}^{F}|\theta_f|$$

Viene penalizzata la presenza θ_f diversi da zero.
Il modello tenderà a usare i regressori che giustificano meglio l'output, **fissando a zero i θ_f dei regressori poco significativi**.
L'iperparametro $\boldsymbol{\alpha}$ ci permette di agire sul peso della penalità.

Elastic-Net(α, β):

$$\text{minimize}_{\boldsymbol{\theta}\in\mathbb{R}} \frac{1}{2N}\sum_{i=1}^{N}(\hat{y}_i - \boldsymbol{\theta}\hat{\boldsymbol{x}}^{\boldsymbol{T}})^2 + \alpha\sum_{f}^{F}|\theta_f| + \beta\left(\sum_{f}^{F}{\theta_f}^2\right)^2$$

Ha proprietà simili al *Lasso*, ma **gestisce meglio la presenza di dipendenza o correlazione tra i regressori.**
In questo caso abbiamo due iperparametri per bilanciare il modello, $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$

Regressione Polinomiale

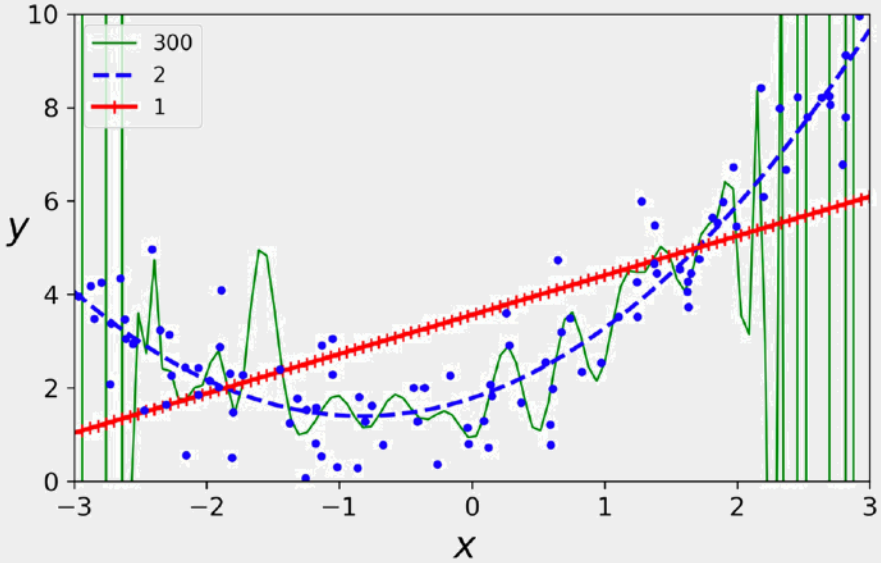
È possibile utilizzare un modello lineare per spiegare dati che presentano comportamenti non lineari.

Un modo semplice per ottenere una regressione non lineare è aggiungere al gruppo di regressori le rispettive potenze di grado indicato (k)

Questa tecnica viene chiamata **Regressione Polinomiale**

Esempio: un solo regressore x, elevato fino alla potenza k

$$y = p_\theta(x, k)$$



$p_\theta(x, 1) = \theta_1 x$

$p_\theta(x, 2) = \theta_1 x + \theta_2 x^2$

$p_\theta(x, 300) = \sum_{i=1}^{300} \theta_i x^i = \theta_1 x + \theta_2 x^2 + \cdots + \theta_{300} x^{300}$

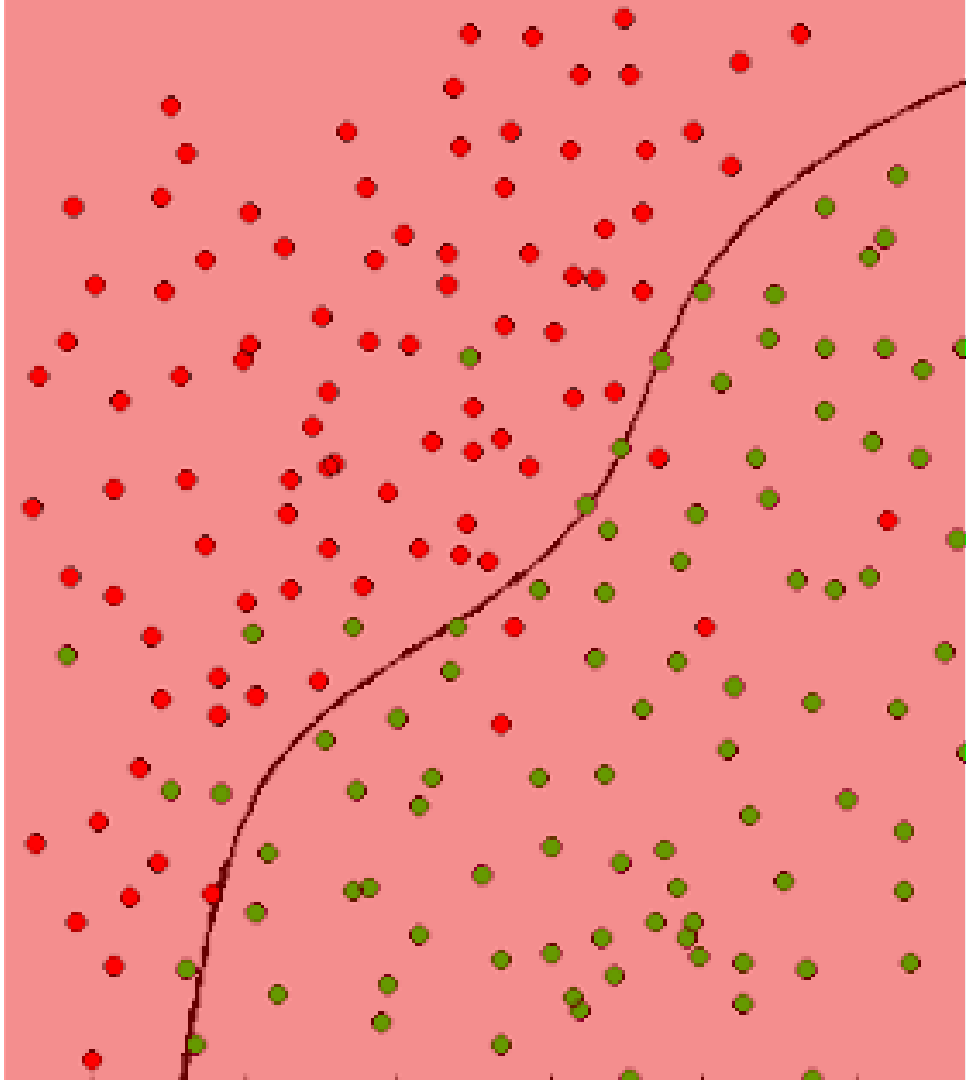
Cosa è k?

Classificazione

La **classificazione** ha come scopo assegnare una classe o una categoria tramite un insieme di dati. Le classi vengono spesso chiamate anche **target/labels.**

In questo caso abbiamo:

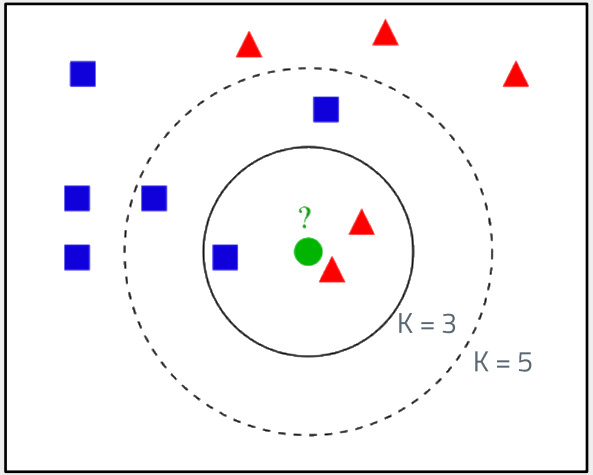
- ✓ un insieme di **varibili di input** (x)
- ✓ una **variabile *discreta o categorica* in output** (y)
- ✓ una **mapping function** $(f(\cdot))$ che mette in relazione gli input con le classi in output



K-nearest neighbors

Il **K-nearest neighbors** è un semplice modello di classificazione che consiste in:

- ✓ Prendere un punto (un nuovo dato)
- ✓ Calcolare le distanze rispetto alle altre osservazioni
- ✓ Definire il vicinato (*neighborhood*) prendendo tra le osservazioni le prime *K* vicine
- ✓ Assegnare alla nuova osservazione la classe con maggior frequenza nel vicinato



K = 3
La maggior parte del vicinato è rossa.
Il nuovo dato sarà rosso.

K = 5
La maggior parte del vicinato è blu.
Il nuovo dato sarà blu.

Modificare gli iperparametri del modello cambia il risultato finale.

Classificatore Lineare

Un **classificatore lineare** consiste nella ricerca di un *iperpiano separatore* che divida lo spazio dei dati in spazi appartenti a classi differenti.

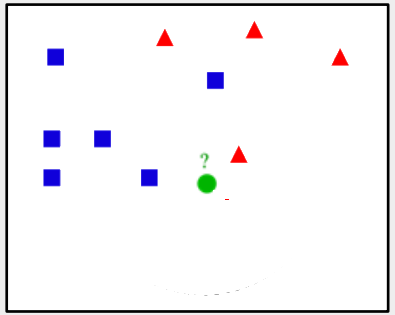
A volte sono presenti più iperpiani lineari che soddisfano questo requisito, e ciò porta **ambiguità nella scelta**.

Tuttavia, un iperpiano lineare che divida i dati in classi nette potrebbe anche **non esistere**. In tal caso sono necessarie delle strategie aggiuntive.

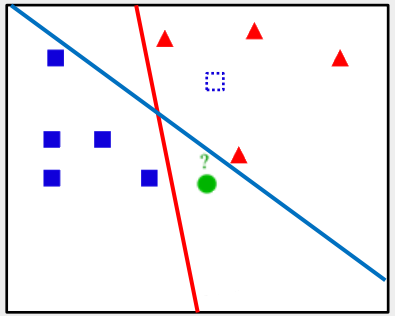
Un *iperpiano* è una figura N-1 dimensionale in uno spazio N-dimensionale.

Esempi:

- *N=1:* l'iperpiano è rappresentato da un **punto**
- *N=3:* l'iperpiano è rappresentato da un **piano**
- *N=4:* l'iperpiano è rappresentato da uno **spazio tridimensionale**



Non esistono iperpiani lineari che dividano le classi



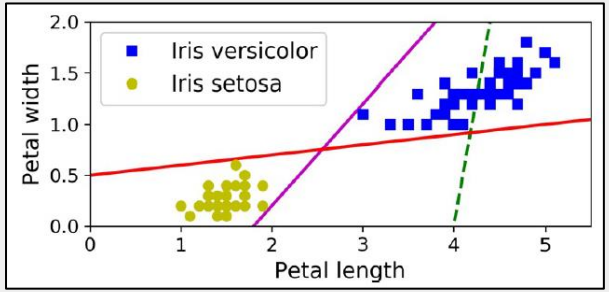
La scelta dell'iperpiano impatta fortemente la classificazione finale.

Quanti iperpiani separatori ci sono in questo esempio?

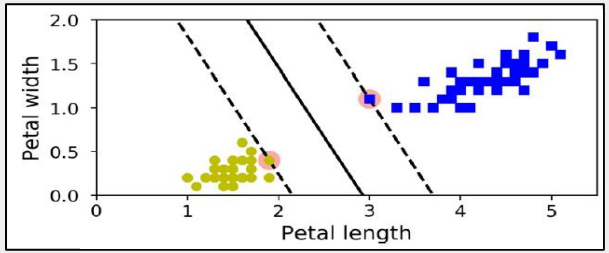
Support Vector Machine

Il classificatore **Support Vector Machine (SVM)** è un potente e versatile modello di ML, in grado di definire classificazioni lineari (e non), regressioni e anche individuare gli outlier.

Tali modelli sono particolarmente adatti per classificazioni su dataset di piccola o media dimensione.



Ho *infinite* scelte di iperpiani separatori



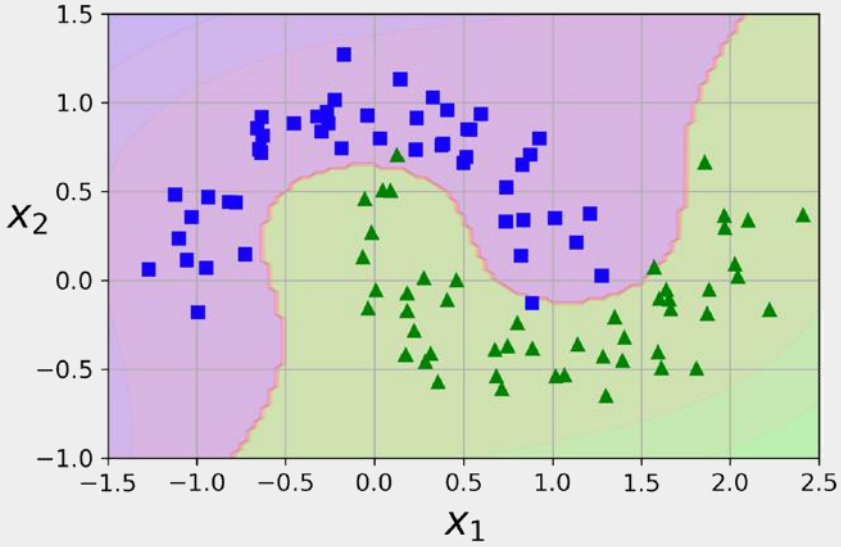
Margine: *distanza che intercorre tra l'iperpiano separatore e il suo punto più vicino (indipendentemente dalla classe)*

Il classificatore SVM seleziona l'iperpiano che che massimizza il margine.

Intuitivamente il SVM seleziona l'iperpiano che individua la *strada piu larga possible* tra le classi.

SVM Non-Lineare

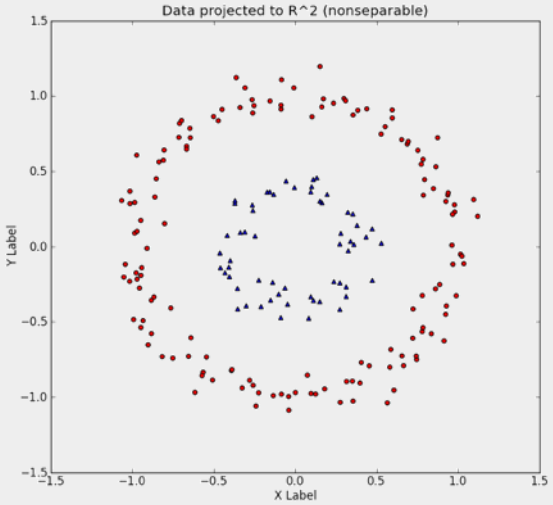
Un SVM può anche gestire una classificazione non lineare **applicando delle trasformazioni alle feature** (in maniera similare a quanto visto per la regressione polinomiale)

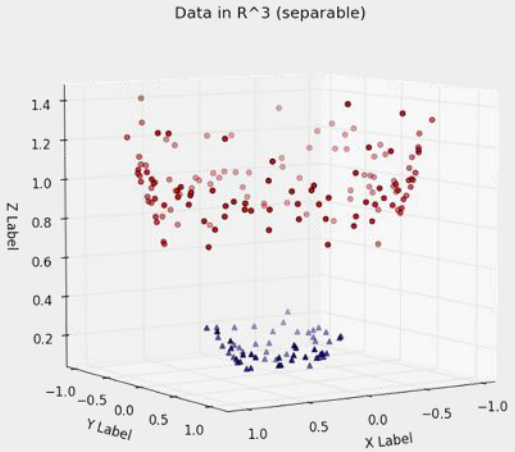


Kernel-trick

Il Kernel-trick consiste nell'utilizzo di particolari trasformazioni spaziali, detti Kernel, che aumentano la dimensione dello spazio dei dati.

L'Idea è che se i dati non sono linearmente separabili un uno spazio N-dimensionale,lo potrebbero essere in uno spazio N+1-dimensionale.
Si procede alla trasformazione spaziale, si effettua la classificazione e infine si torna nello spazio di partenza con le categorie assegnate

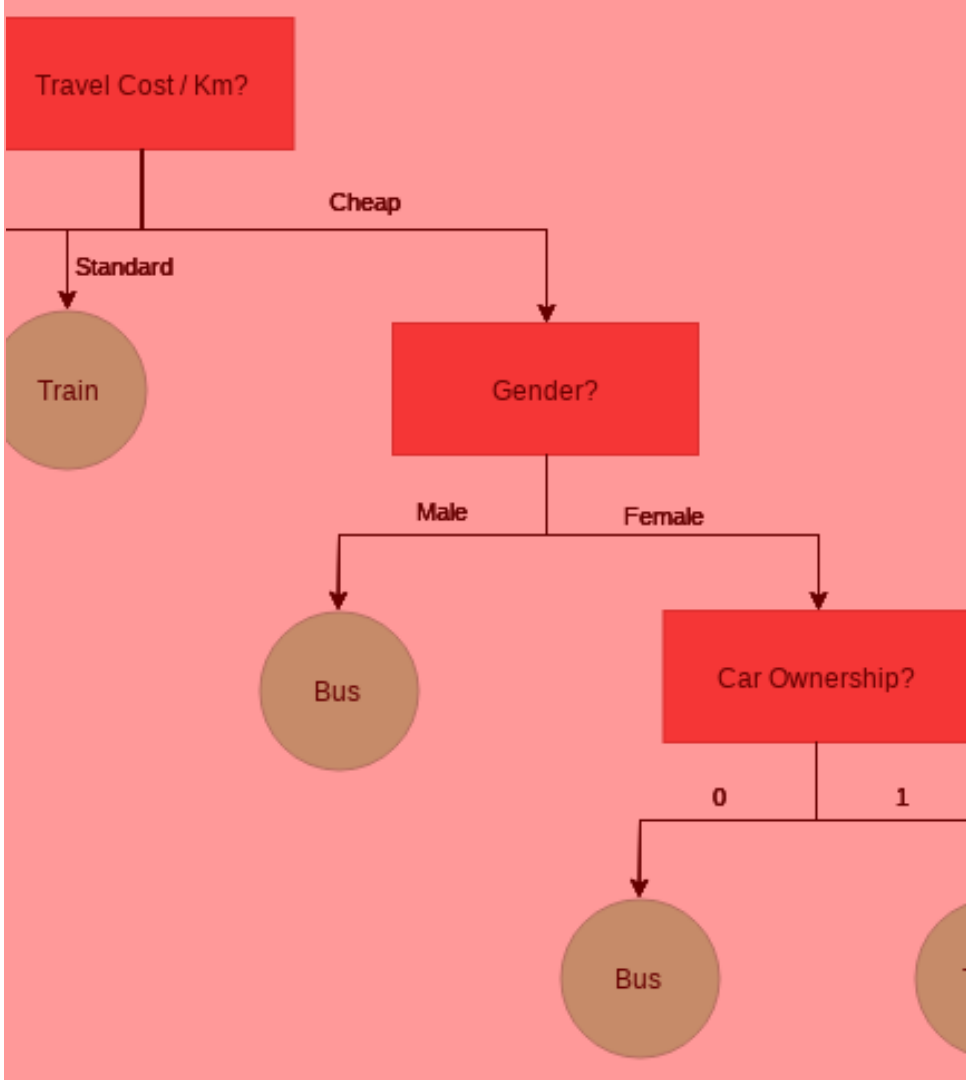




Alberi Decisionali

Un albero decisionale è una struttura a flowchart in cui:

- ✓ ogni nodo ***interno*** rappresenta una **condizione su un attributo**
- ✓ ogni **ramo** rappresenta la **risposta alla condizione** del nodo
- ✓ ogni nodo ***foglia*** rappresenta un **assegnamento di classe**.

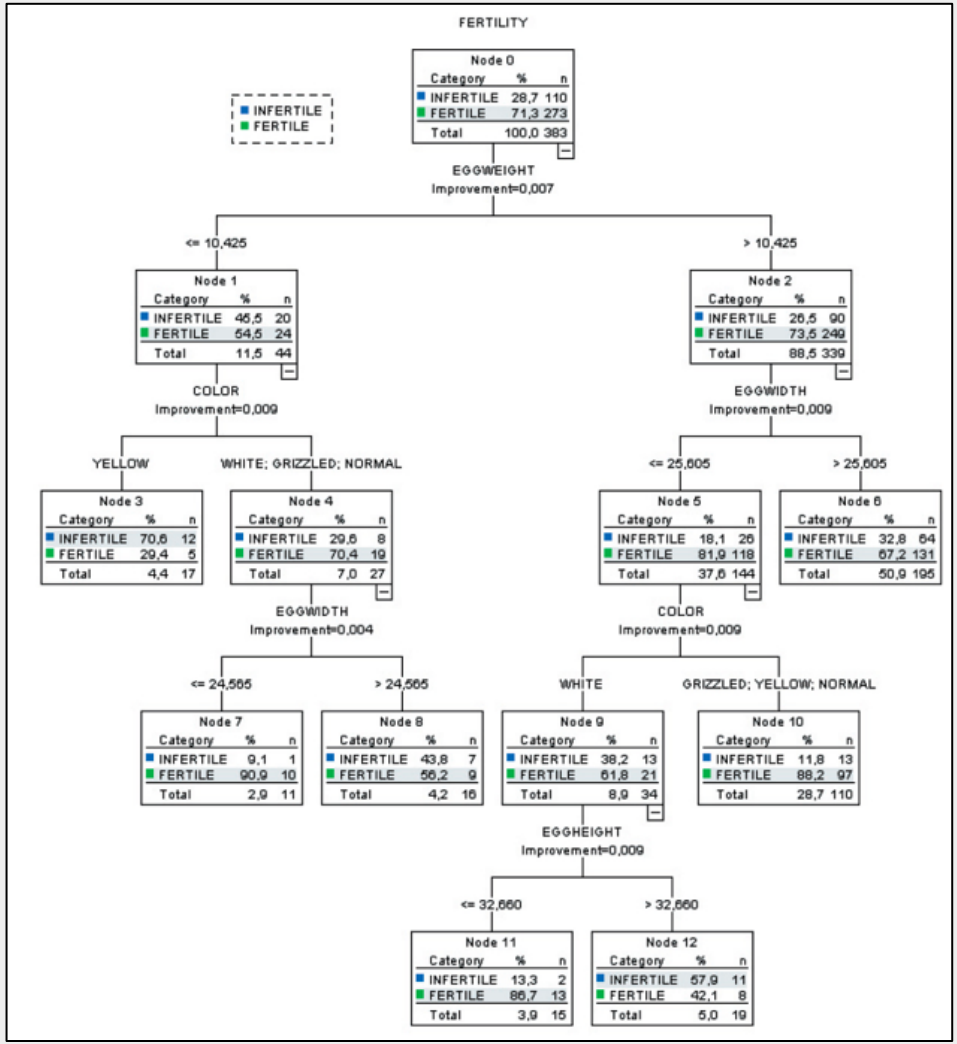


Algoritmo CART

CART = Classification And Regression Tree

L'algoritmo è composto dalle seguenti fasi:

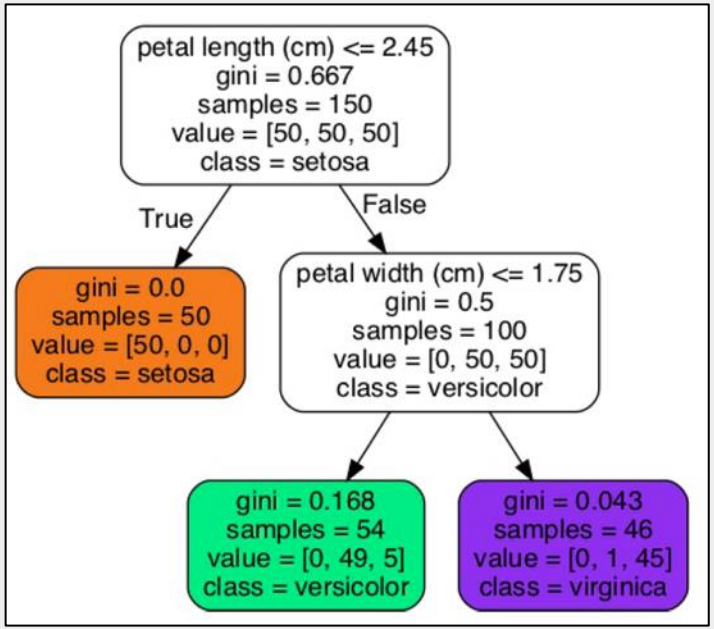
- ✓ Per ogni feature k, **individuo un threshold t** che meglio separa le classi
- ✓ Seleziono la coppia (k, t) che ha causato la miglior separazione
- ✓ **Divido il dataset in due gruppi** a dipendenza di dove si trovano le osservazioni rispetto al threshold t sulla feature k
- ✓ Ripeto il procedimento per tutti i gruppi di dataset considerando le altre feature



Classificare con alberi

A seguito della creazione dei nodi interni e delle foglie dell'albero la classificazione di nuove osservazioni avviene con questi passaggi:

- ✓ Partendo dalla radice (primo nodo interno) osservando il valore della condizione del nodo, si sceglie coerentemente il percorso da seguire
- ✓ Si reitera questo procedimento per ogni nodo interno
- ✓ Quando si raggiunge una foglia, abbiamo ottenuto la classificazione



Ensable Learning

Aggregando l'output di modelli differenti, a volte si ottengono risultati migliori rispetto all'utilizzo di un singolo modello addestrato su tutto il dataset.

Un insieme di modelli che eseguono la stessa stima viene chiamato **ensemble**.

La tecnica che usa differenti alberi decisionali per ottenere un migliore output viene chiamata **Random Forest**.

Random Forest

Per sua natura, l'albero decisionale spesso pecca di *overfitting*.

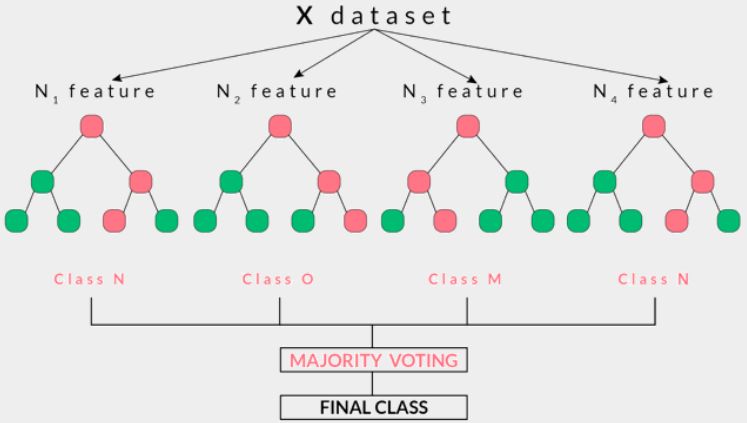
Per ovviare a questo inconveniente si crea una **random forest:**

- ✓ Si creano diversi alberi decisionali con un **numero ristretto di feature casuali**.
- ✓ Per una nuova osservazione, ogni singolo albero produce una propria classificazione.
- ✓ Si raccolgono i vari risultati, e **la classe con maggior frequenza** viene scelta come classe definitiva

L'*overfitting* è **l'eccessiva specializzazione del modello sul training set**. Quando si ha overfitting, il modello produce scarsi risultati per i dati non presenti nel training set.
Esistono tecniche e indicatori che individuano i casi di overfitting.

Esempi (un po' estremi):

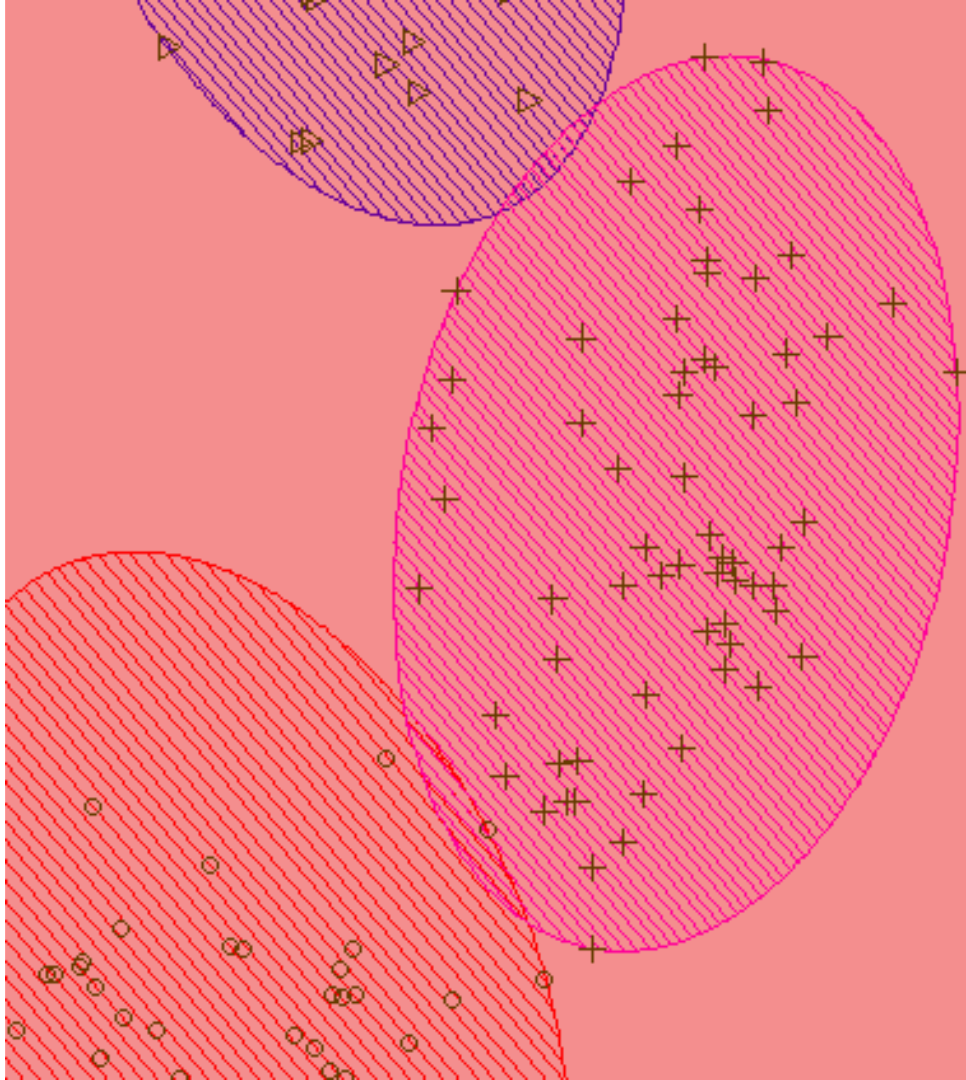
- Un albero con 200 nodi addestrato su 200 osservazioni
- Una regressione polinomiale di grado 300 con 300 osservazioni



Clustering

Nella **Cluster analysis** (o **clustering**) l'obbiettivo è raggruppare un insieme di oggetti in modo tale che gli oggetti in un gruppo (detto **cluster**) siano **più simili tra di loro** rispetto a quelli assegnata a gruppi differenti.

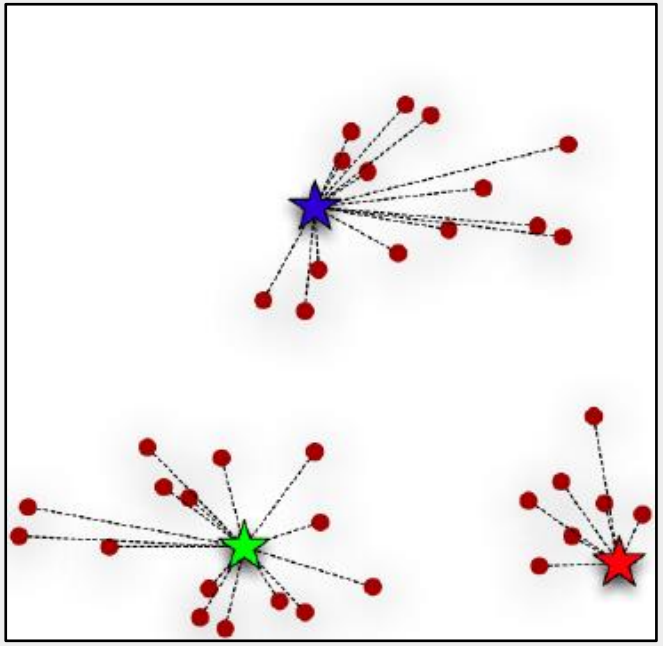
Come nella classificazione, l'obbiettivo è assegnare ad un oggetto una etichetta. Tuttavia, a differenza della classificazione, nel clustering non abbiamo risposte corrette da cui apprendere.
È un algoritmo non supervisionato



K-Means

Il **K-Means** è un algoritmo iteritavo che individua un **numero di gruppi predefinito k** nello spazio dei dati. L'algoritmo consiste nei seguenti passaggi:

1. Si generano k punti casuali nello spazio, detti **centroidi,** oppure si scelgono k punti casuali del dataset.
2. Ogni dato viene assegnato al centroide più vicino
3. Per ogni gruppo definito generato viene calcolato il suo **punto medio.** I punti trovati **divengono i nuovi centroidi.**
4. Ripetere il processo fino a che non si osservano più cambiamenti nei cluster

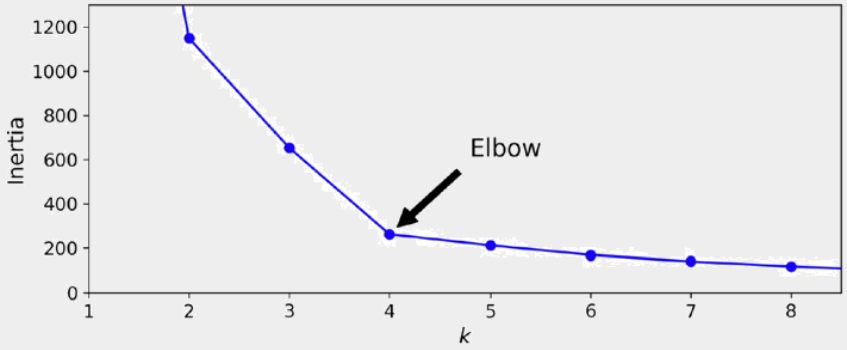


- ✓ L'algoritmo è molto sensibile alla variazione dei centroidi iniziali.
- ✓ Spesso si adotta un approccio *multi-start*
- ✓ Il numero di gruppi è un iperparametro molto importante

Quanti cluster?

Il **metodo del gomito (o elbow method)** consiste nei seguenti passaggi:

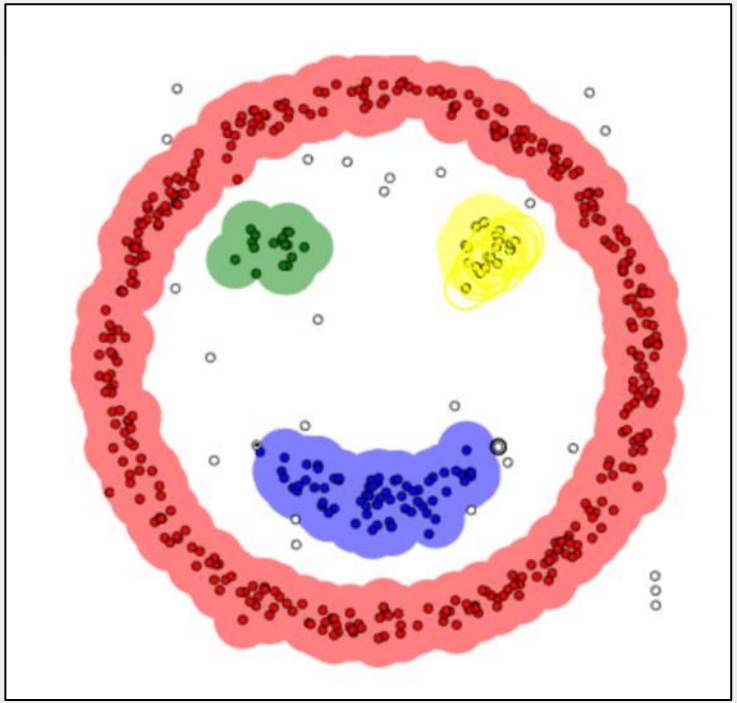
1. Utilizzare più volte l'algoritmo K-Means con un numero crescente di cluster
2. Per ogni clusterizzazione calcolare il valore di **inerzia** *(la somma degli scarti quadratici tra punti e rispettivi centroidi)*
3. Creare un grafico che mostri il valore di inerzia al crescere dei numeri di gruppi
4. Il numero di cluster *corretto* è quello in corrispondenza del *gomito della curva, (*ovvero dove la curva ha una maggiore variazione di pendenza)



DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) è un algoritmo che definisce cluster di regioni continue ad alta densità.

1. Si estrae un punto a caso nel dataset e si calcola il vicinato **entro un raggio predefinito**.
2. Se il vicinato così definite è non-vuoto, si unicono i cluster individuati e il punto.
3. Continuare fino a che tutti I punti non vengono visitati.



- GRAZIE

Via Dante, 6, 21052 Busto Arsizio VA
Tel.: +39.0331.357.400
Fax: +39.0331.622.869