

UNIVERSITÀ DEGLI STUDI DI  
MILANO-BICOCCA



SCUOLA DI SCIENZE  
CORSO DI LAUREA MAGISTRALE IN DATA SCIENCE

---

## Dispensa di Machine Learning

---

Autori:

Federico Luzzi  
Christian Uccheddu

ANNO ACCADEMICO 2019/2020

---

*Questa vuole essere una dispensa riassuntiva del corso Machine Learning del corso di laurea magistrale in Data Science dell'Università degli Studi di Milano Bicocca. La scrittura di questa dispensa è stata possibile solo grazie a tutto il materiale messo a disposizione dal Professore Fabio Stella. Il repository di questa dispensa è disponibile a questo link: [https://github.com/Chri1629/machine\\_learning\\_book.git](https://github.com/Chri1629/machine_learning_book.git)*  
*Ricorda "Il sapere si accresce solo se condiviso".*

Per qualsiasi informazione/correzione contattateci:

f.luzzi1@campus.unimib.it  
c.uccheddu@campus.unimib.it

# Indice

<b>1 Dati</b>	<b>1</b>
1.1 Data types(*) . . . . .	2
1.2 Data exploration(*) . . . . .	3
1.2.1 Definizioni . . . . .	3
1.2.2 Visualization . . . . .	4
1.3 Missing replacement(*) . . . . .	6
1.4 Data Preprocessing(*) . . . . .	7
1.4.1 Aggregation . . . . .	7
1.4.2 Sampling . . . . .	7
1.4.3 Dimensionality reduction . . . . .	8
1.4.4 Variable transformation . . . . .	11
<b>2 Classification</b>	<b>12</b>
2.1 Introduction (*) . . . . .	12
2.2 Tecniche di classificazione . . . . .	14
2.2.1 Decision Tree . . . . .	14
2.2.2 Regressione Logistica Binomiale . . . . .	16
2.2.3 Support Vector Machines . . . . .	17
2.2.4 Multi-Layer Perceptron o Artificial Neural Network . . . . .	20
2.2.5 Classificatori bayesiani . . . . .	23
2.2.6 Naive bayes . . . . .	24
2.2.7 Reti bayesiane . . . . .	25
2.2.8 Tree-augmented Naive Bayes . . . . .	25
2.2.9 Summary . . . . .	26
2.3 Performance Evaluation (*) . . . . .	26
2.3.1 Performance di un modello di classificazione . . . . .	28
2.3.2 Accuratezza . . . . .	28
2.3.3 Speed . . . . .	29
2.3.4 Robustezza . . . . .	29
2.3.5 Scalabilità . . . . .	30
2.3.6 Interpretabilità . . . . .	30
2.3.7 Holdout . . . . .	30
2.4 Comparing Classifiers (*) . . . . .	32
2.4.1 Intervallo di confidenza . . . . .	32
2.4.2 Different test set . . . . .	33
2.4.3 Same test set . . . . .	35
2.5 Class Imbalance Problem (*) . . . . .	36
2.6 Counting the cost (*) . . . . .	38
2.6.1 Matrice di costo . . . . .	38
2.6.2 Cumulative Gains . . . . .	39
2.6.3 Lift Chart . . . . .	41
2.6.4 Curva ROC . . . . .	42
2.7 Feature Selection (*) . . . . .	43
2.7.1 Filter . . . . .	44
2.7.2 Regularization . . . . .	46
2.7.3 Schema division . . . . .	47
2.8 Classificazione NON binaria . . . . .	48
2.8.1 One-Vs-All . . . . .	48

<b>3 Clustering</b>	<b>49</b>
3.1 Introduzione(*) . . . . .	49
3.1.1 Tipi di clustering . . . . .	49
3.1.2 Differenti nozioni di cluster . . . . .	50
3.1.3 Componenti di un'analisi di clustering . . . . .	51
3.2 Proximity(*) . . . . .	51
3.2.1 Introduzione . . . . .	51
3.2.2 Misure della distanza . . . . .	53
3.2.3 Altre misure di prossimità . . . . .	54
3.3 Clustering Algorithms . . . . .	56
3.3.1 Prototype Based . . . . .	56
3.3.2 Clustering Gerarchico . . . . .	62
3.3.3 Density-Based Clustering . . . . .	63
3.3.4 Graph-based Clustering Algorithm . . . . .	65
3.4 Clustering Evaluation(*) . . . . .	68
3.4.1 Esterni o supervisionati(*) . . . . .	69
3.4.2 Interni o non supervisionati(*) . . . . .	70
3.4.3 Paradigma di validità(*) . . . . .	71
3.4.4 Selezione del numero di cluster(*) . . . . .	72
<b>4 Association Analysis</b>	<b>74</b>
4.1 Introduction (*) . . . . .	74
4.2 Rule Extraction . . . . .	77
4.2.1 Algoritmo apriori . . . . .	78
4.3 Maximal/Closed Frequent Itemsets . . . . .	79
4.3.1 Rule Generation . . . . .	79
4.4 Rules Evaluation (*) . . . . .	81
4.5 Simpson's Paradox . . . . .	84

# 1 Dati

Gli ambiti più importanti nei quali vengono applicate tecniche di machine learning nella vita reale sono diversi, in particolare si usano in: finanza, sanità, agricoltura, e-commerce, social, chatbot, sensoristica (come i veicoli a guida autonoma).

Vista la grande mole di dati la necessità è quella di capire come trattare i dati. **L'obiettivo del Machine Learning è sviluppare una metodologia per dare valore ai dati in funzione di una particolare domanda che ci stiamo ponendo.**

Tipicamente le tecniche di machine learning si dividono nelle seguenti tre macro categorie:

1. *Apprendimento supervisionato o predittivo*: qualcuno ha già catalogato ad esempio delle immagini o dei dati e noi prendendo questi modelli dovremmo essere in grado di predire.
2. *Apprendimento non supervisionato o descrittivo*: ci sono delle funzioni obiettivo che vanno ottimizzate. Non usiamo etichette della singola istanza ma in qualche modo sappiamo dove arrivare.
3. *Apprendimento rinforzato*: È quello più utile in questa epoca; il suo funzionamento è basato sui premi.

In questo corso ci si concentrerà sui primi due tipi di apprendimento.

L' *apprendimento supervisionato* si divide a sua volta nelle seguenti due categorie:

- Classificazione: quando il problema consiste nel dividere in classi delle quantità discrete.
- Regressione: quando il problema consiste nel ricostruire una certa variabile date delle condizioni pregresse.

L' *apprendimento non supervisionato* si divide a sua volta in:

- Clustering: Quando il problema consiste nel ricostruire delle classi delle istanze che ci vengono consegnate, senza sapere nulla sulla storia pregressa.
- Associativa: Quando il problema consiste nel scoprire pattern che descrivono bene caratteristiche associate ad un certo fenomeno.

Per alcuni compiti la correlazione statistica va benissimo, in alcuni casi però essa è addirittura deleteria. Se infatti si provasse a vedere la correlazione tra il numero di omicidi in America e il numero di fondi investiti sulla ricerca scientifica si vedrebbe che statisticamente sono strettamente correlate. Questo è un no-sense ed è il classico esempio di *correlazione spuria*. A confermare questa visione in cui il modello per una certa trattazione dati è fondamentale è il famoso paradosso di Simpson.

**Paradosso di Simpson:** Utilizzare i dati senza un modello valido non porta alla scoperta della verità.

## 1.1 Data types(\*)

Il primo passo fondamentale è quello di prendere confidenza coi dati. E' quindi molto importante capire la natura intrinseca dei dati che abbiamo a disposizione per risolvere in modo efficace un problema, in particolare, questi sono solitamente organizzati in strutture che chiamiamo **dataset**. Per le analisi con le tecniche di Machine Learning esse non sono altro che tabelle fatte da righe e da colonne.

**Definizione 1.1.** Se definiscono **attributi** le colonne del dataset

**Definizione 1.2.** Si definiscono **istanze** le righe del dataset

Ogni attributo è caratterizzato dal fatto di avere un *tipo*, la sua conoscenza è fondamentale perché permette di sapere le proprietà che questo possiede. In particolare, gli attributi si dividono in due grandi gruppi:

- *Categorici:*
  - Nominali: ad esempio il colore degli occhi.
  - Ordinali: ad esempio possono essere i giudizi.
- *Numerici:*
  - Intervallo: ammettono operazioni di somma e sottrazione.
  - Ratio: possiamo applicare tutte le operazioni logico/matematiche.

ATTRIBUTE TYPE	DESCRIPTION	EXAMPLES	OPERATIONS
CATEGORICAL (QUALITATIVE)	The values of a nominal attribute are just different names; i.e., nominal values provide only enough information to distinguish one object from another ( $=$ , $\neq$ ).	Area Code	mode
		Churn	entropy
		State	contingency
		eye color	
		gender	
ORDINAL	The values of an ordinal attribute provide enough information to order objects ( $<$ , $>$ ).	{bad, good, excellent}	median
		grades	percentiles
		street numbers	rank correlation
			run tests
			sign tests
NUMERIC (QUANTITATIVE)	For interval attributes, the difference between values are meaningful, i.e., a unit of measurements exists (+, -).	calendar dates	mean
		temperature in Celsius or Fahrenheit	standard deviation
			Pearson's correlation
			t and F tests
RATIO	For ratio attributes, both differences and ratios are meaningful, $(*, /)$ .	Day Mins	geometric mean
		Eve Mins	harmonic mean
		monetary quantities	percentiles
		length	variation
		electrical current	

Figura 1: Esempi di attributi divisi per categoria con le relative proprietà.

Dall'alto al basso il livello gerarchico sale e le proprietà aumentano.

C'è un'ulteriore divisione che può essere fatta ed è quella in:

- Discreti (in cui la serie dei valori è finita o una infinità numerabile), che a loro volta si suddividono in:
  - Categorici
  - Numerici
  - Binari: sono i più particolari da trattare, e hanno una serie di proprietà strane.
- Continui: i cui valori sono numeri reali.

## 1.2 Data exploration(\*)

E' utile sapere come esplorare i dati in modo intelligente usando tutti gli strumenti a disposizione. Di seguito un elenco degli strumenti statistici più utili che permettono di effettuare un'esplorazione completa dei dati.

### 1.2.1 Definizioni

**Definizione 1.3.** Si definisce **quantile** di ordine  $\alpha$  il valore  $q_\alpha$  che divide la popolazione in due parti proporzionali rispettivamente ad  $\alpha$  e a  $(1-\alpha)$  e caratterizzate da valori rispettivamente minori e maggiori di  $q_\alpha$

Un quantile molto importante è il quantile di ordine  $\frac{1}{2}$  che viene definito **median**, per come costruito questo è il valore che ha esattamente minori di lui metà dei dati.

**Definizione 1.4.** Si definisce **media** il seguente valore:

$$\text{mean} = \frac{1}{n} \sum_{i=1}^n x_i$$

La media non è un buon modo di visualizzare i dati perché dice poco riguardo alla loro distribuzione. Essendo la media basata su singole osservazioni è fortemente soggetta a variazioni quando si hanno valori fortemente discostati dalla distribuzione dei dati. Tali valori si chiamano *outlier* e sono particolarmente interessanti da trattare.

Per ovviare a questo problema si è soliti usare la **media trimmed** in cui si buttano via il valore più piccolo e il valore più grande.

**Definizione 1.5.** Si definisce **media trimmed** il seguente valore:

$$\text{mean}_{\text{trimmed}} = \frac{1}{n} \sum_{i=1}^n x_i \quad x_i = x - \max x - \min x$$

Se si trova un grosso scostamento tra la media e la media trimmed probabilmente si ha la presenza di almeno un outlier.

**Definizione 1.6.** Si definisce **range** il seguente valore:

$$\text{range} = \max x - \min x$$

Questo valore serve a quantificare la dispersione dei dati, ovviamente può essere fuorviante qualora i valori siano concentrati in una stretta banda. Per ovviare a questo problema si usa la varianza.

**Definizione 1.7.** Si definisce **varianza** il seguente valore:

$$\text{var} = \sigma^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

E' preferibile però usare la deviazione standard in quanto per come è definita risulta essere della stessa unità di misura dei dati.

**Definizione 1.8.** Si definisce **deviazione standard** il seguente valore:

$$\text{std} = \sigma = \sqrt{\sigma^2}$$

Essendo queste due grandezze definite a partire dalla media esse soffrono dello stesso problema: *sono fortemente condizionate dagli outlier*; con lo stesso metodo precedente si definiscono, quindi, altre due grandezze in grado di ovviare a questo problema.

**Definizione 1.9.** Si definisce **deviazione media assoluta** il seguente valore:

$$AAD = \frac{1}{n} \sum_{i=1}^n |x_i - \text{mean}|$$

**Definizione 1.10.** Si definisce **deviazione mediana assoluta** il seguente valore:

$$MAD = \text{mediana}(x_1 - \text{mean}, x_2 - \text{mean}, \dots, x_{n0} - \text{mean})$$

E' utile anche definire il range interquartile (IQR) per le stesse ragioni precedenti.

**Definizione 1.11.** Si definisce **Range InterQuartile** il seguente valore:

$$IQR = q_{75\%} - q_{25\%}$$

Ci sono anche diverse grandezze utili da definire nei casi in cui si ha a che fare con coppie di attributi, in tal caso:

**Definizione 1.12.** Si definisce **covarianza** il seguente valore:

$$\text{cov}(X, Y) = \frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})(y_i - \bar{y})$$

Per come è costruita questa matrice è necessariamente quadrata, e il suo valore  $ij$ -esimo rappresenta la covarianza tra il valore  $i$ -esimo dell'attributo x e il valore  $j$ -esimo dell'attributo y.

Un'ulteriore misura dell'associazione tra le coppie di attributi quantitativi che non dipende dalla varianza di ciascun attributo è la seguente:

**Definizione 1.13.** Si definisce **correlazione di Pearson** il seguente valore:

$$\text{corr}(x, y) = \frac{\text{cov}(x, y)}{\sqrt{\text{var}(x)\text{var}(y)}}$$

Per come è definita si ha ovviamente che:  $\text{cov}(x, y) \in [-1, 1]$ .

### 1.2.2 Visualization

E' molto utile visualizzare i dati quando bisogna lavorarci, questo fondamentalmente per due motivi:

- Ci permette di trovare pattern tra le variabili che valori puntuali non ci permetterebbero di trovare.
- Ci permette di visualizzare i risultati di una lavorazione fatta sui dati.

Di seguito viene proposto un rapido elenco dei grafici più utili e se ne descrivono le varie caratteristiche.

I dati possono essere organizzati in istogrammi caratterizzati dalla presenza di bin(indicano la larghezza in cui i dati sono organizzati). A seconda dell'ampiezza che uso posso ottenere due disegni molto diversi come mostrato in figura.

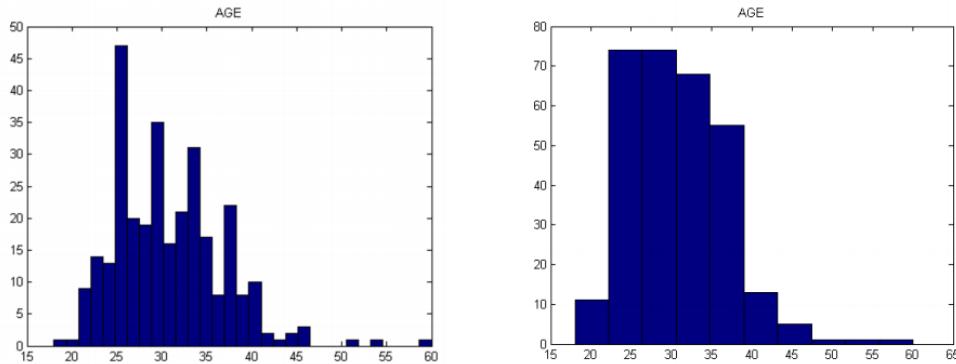


Figura 2: Differenza tra due istogrammi fatti sulla stessa distribuzione di dati ma con numero di bin diversi.

Si possono allo stesso modo creare istogrammi per dati qualitativi. La differenza rispetto agli istogrammi sui dati quantitativi è quella per cui ogni bin corrisponde ad una categoria diversa.

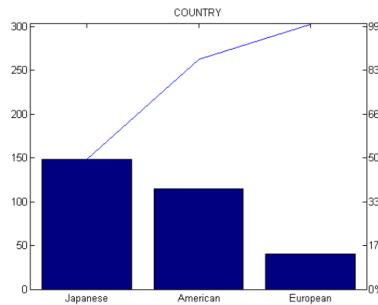


Figura 3: Istogramma fatto su dati qualitativi.

In questo caso la linea blu disegnata sopra è una linea cumulativa, ossia rappresenta dove si trova il livello sommando tutti i dati che man mano incontro, per come è costruita ovviamente dovrà finire sul valore 100%

Un altro modo di rappresentare i dati particolarmente utile è il **grafico Box and Whiskers** applicato solo ad attributi quantitativi. Rispetto agli istogrammi questo è decisamente più utilizzato perché permette di estrarre più informazioni. In particolare, proponiamo un grafico in cui sono esplicitate le informazioni ottenibili:

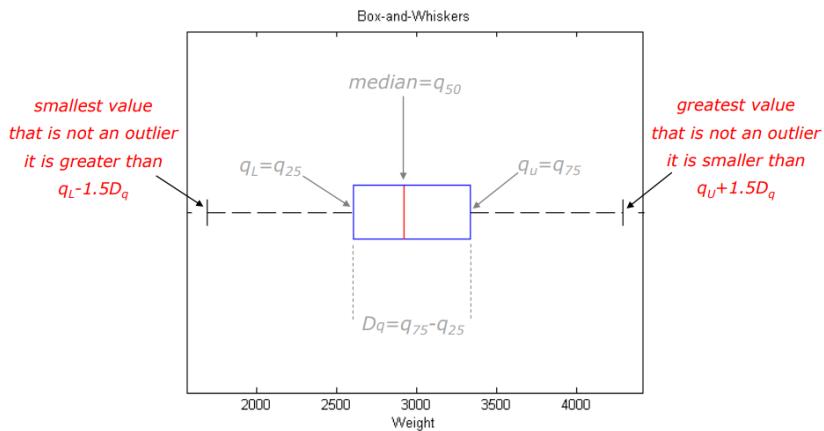


Figura 4: Informazioni ottenibili da un box plot.

### 1.3 Missing replacement(\*)

E' la parte dell'analisi dati che si occupa di studiare come sostituire i valori mancanti di certi attributi in un dataset. Essendo un problema di così larga portata si può capire bene che una trattazione esaustiva riempirebbe le ore di un intero corso, tuttavia vengono fornite le basi per essere in grado di poter effettuare una analisi dati efficace.

Le motivazioni principali del perché un database presenti delle mancanze sono le seguenti:

- Tale valore non è stato possibile misurarlo.
- Non si conosce con esattezza tale valore.
- Si è verificato qualche errore durante la presa dati.
- Quel determinato attributo fino ad un certo punto dell'analisi non è mai stato considerato importante e quindi non è mai stato registrato.

Ci sono diversi metodi che consentono di riempire quel valore mancante, di seguito viene fornita una rapida trattazione dei più usati:

- **Record removal:** è il metodo più drastico e consistere nell'eliminare tutta l'istanza che contiene quel valore. Non viene usato frequentemente perché si rischia di perdere valori che possono essere fondamentali per l'analisi dati. *In particolare se la probabilità che un attributo abbia un valore nullo è:  $p(NULL)$  ed è un record con  $m$  attributi allora la probabilità di non avere dati nulli in quel record diventa:*

$$(1 - p(NULL))^m$$

- **Global constant:** consiste nel sostituire tutti i missing values con un valore costante chiamato *place holder*. Tale metodo non è molto efficiente perché un valore costante non può essere rappresentativo di una distribuzione.
- **Manual imputation:** consiste nel sostituire manualmente i missing values tramite delle osservazioni. Lo svantaggio è che è tremendamente svantaggioso dal punto di vista computazionale.

- **Moda replacement:** consiste nel rimpiazzare tutti i missing values con la moda di quel dato attributo, valgono le stesse considerazioni fatte per il metodo della global constant. Qualora gli attributi fossero continui si effettuerebbe la stessa cosa ma sostituendo la media.
- **Conditional mean replacement:** consiste nel rimpiazzare tutti i missing values con la media a condizione del fatto che sia presente un’ulteriore condizione posta su un secondo attributo. E’ un po’ più efficace degli altri elementi ma presenta anch’esso delle criticità.
- **Most probable:** consiste nell’eseguire una regressione sul nostro dataset utilizzando un altro attributo. In questo caso è possibile usare dei modelli di regressione anche molto complessi ed è possibile lavorare anche con dati qualitativi.

E’ ora chiaro che il confine tra l’esplorazione dati e la modellizzazione degli stessi è molto sottile.

## 1.4 Data Preprocessing(\*)

Il preprocessing è una fase dell’analisi dati che consiste nel **rendere i dati più fruibili per una analisi dati**. Di seguito le tecniche più utilizzate nella data preprocessing.

### 1.4.1 Aggregation

*L’aggregazione consiste nel combinare due o più record in un unico oggetto.* Ci sono diversi vantaggi ottenibili nell’aggregare i dati, in particolare:

- *Dataset più piccoli:* durante un’analisi dati abbiamo bisogno di usare il minor quantitativo di memoria e di tempo; l’aggregazione diminuisce il tempo di esecuzione di un algoritmo.
- *Cambiamento di scopo:* ci permette di avere una visione più ampia dei dati qualora cambiassimo lo scopo della nostra analisi in corso d’opera.
- *Varianza ridotta:* gli attributi calcolati su record aggregati sono più stabili rispetto a quelli associati ai record nativi, questo per un effetto statistico.

### 1.4.2 Sampling

In molti casi avere una quantità enorme di dati può essere deleterio dal punto di vista computazionale, questo perché bisognerebbe usare algoritmi più semplificati in modo che la computazione possa essere svolta su un maggior numero di dati. Ciò che va preferito è invece usare un algoritmo migliore su un dataset ridotto. Per questo entra in gioco il concetto di **campionamento**.

Il problema si traduce nella seguente domanda: *quando un campione è rappresentativo?*

**Definizione 1.14.** Un campione si dice **rappresentativo** quando ha approssimativamente le stesse proprietà del dataset di partenza.

Si deve, quindi, trovare uno schema che ci permetta di scegliere *con grande probabilità* dei campioni rappresentativi. In questo caso il problema si riduce a trovare le appropriate:

- Dimensioni del campione.
- Tecniche di campionamento.

Esistono moltissime tecniche di campionamento e di seguito mostriamo le più basilari:

- **Simple Random Sampling:** Ogni record del dataset ha la stessa probabilità di essere incluso nel campione. Tale record può essere rimosso o meno dal dataset di partenza. Quando i campioni sono molto piccoli rispetto al dataset di partenza la rimozione o meno genera due campioni che sono molto simili. Questo metodo fallisce quando il dataset consiste di attributi qualitativi in modo che i possibili valori che possono avere hanno frequenze fortemente diverse.
- **Stratified Sampling:** Vengono presi record in modo che all'interno del campione vengano rispettate le proporzioni tra gli attributi presenti nel dataset di partenza.

Una volta scelta la tecnica di campionamento bisogna occuparsi di scegliere la grandezza del campione. Tenere un campione di grande ampiezza aumenta la probabilità che un campione sia rappresentativo, di contro elimina la maggior parte dei vantaggi computazionali di avere un campionamento. Avere un campione troppo piccolo invece manda in conto al rischio di eliminare pattern potenzialmente importanti o addirittura di mantenere pattern erronei. La giusta dimensione del campionamento va scelta in base al nostro dataset di riferimento effettuando delle prove.

#### 1.4.3 Dimensionality reduction

In diversi casi capiterà di dover analizzare dataset caratterizzati da un gran numero di attributi. Diminuirne il numero porta a diversi **vantaggi**:

- Molti algoritmi lavorano meglio se la dimensionalità è minore.
- L'interpretabilità del modello implementato aumenta perché dipende da un numero minore di attributi.
- La rappresentazione grafica dei dati è facilitata.
- L'ammontare di tempo e memoria diminuisce drasticamente.

Contrariamente a ciò che ci si aspetterebbe *avere una dimensionalità maggiore dei dati non implica un aumento delle performance*. Mostriamo questo dato con il seguente grafico.

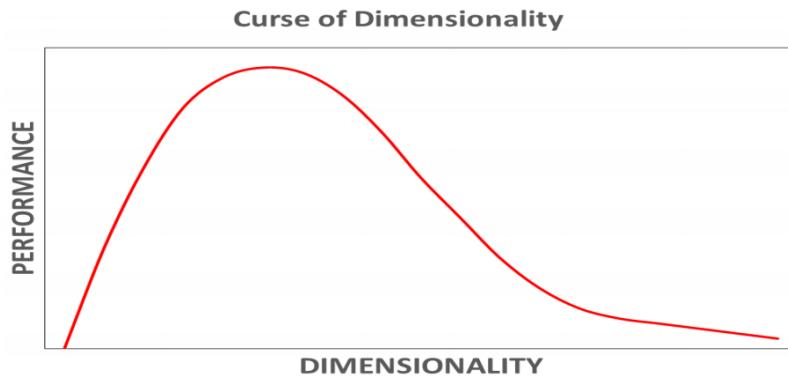


Figura 5: Andamento delle performance di un algoritmo in funzione della dimensionalità del dataset.

Molte delle tecniche usate per ridurre le dimensionalità sfruttano tecniche prese dall’algebra lineare per proiettare i dati da uno spazio dimensionale maggiore ad uno spazio dimensionale minore.

La **Principal Component Analysis (PCA)** trova nuovi attributi in modo che:

- Siano *combinazioni lineari* degli attributi di partenza.
- Siano *mutualmente ortogonali*.
- Catturino il *massimo ammontare di variabilità* nei dati.

In particolare, la **Singular Value Decomposition (SVD)** è una tecnica di algebra lineare correlata alla PCA ed è largamente usata per ridurre la dimensionalità dei dataset.

**Definizione 1.15.** Si definisce **binarizzazione** la tecnica di trasformazione di attributi continui e categorici in uno o più attributi binari.

Per binarizzare un attributo si associano  $k$  possibili valori a  $k$  valori interi nell’intervallo  $[0, k - 1]$ . Successivamente si trasformano questi  $k$  interi in un numero binario; come è noto il numero di cifre necessarie per rappresentare un numero intero sono le seguenti:

$$s = \lceil \log_2 k \rceil$$

In questo caso è necessario introdurre un attributo binario per ogni attributo categorico con cui il dato può manifestarsi. Ovviamente si è più interessati ai casi in cui si manifesta il valore 1 perché indica la presenza di tale attributo.

Quando, invece, si ha a che fare con attributi durante un’analisi di classificazione o di associazione bisogna ricorrere alla tecnica di **discretizzazione**. Come facilmente intuibile la miglior discretizzazione dipende dal tipo di algoritmo che sto utilizzando. La discretizzazione può essere:

- **Supervisionata**.
- **Non-supervisionata**.

La *discretizzazione supervisionata* utilizza ulteriori informazioni (attributi di classe) per discretizzare gli attributi. Questa tecnica divide i punti in modo che qualche misura di *purità* sia massimizzata. La misura di purità più spesso applicata è l'**entropia**.

$$e_i = - \sum_{k=1}^K p_{ki} \log_2 p_{ki}$$

Per come è costruita si ha che:

- Se contiene solo record di una data classe  $\Rightarrow e_i = 0$ , massima purità.
- Se contiene egualmente spesso tutte le classi  $\Rightarrow e_i = \max$ , minima purità.

La discretizzazione supervisionata basata sull'entropia permette di trovare i punti di divisione degli attributi *continui* tali che l'entropia totale sia *minimizzata*.

**Definizione 1.16.** Si definisce entropia totale la seguente espressione:

$$E = \sum_{i=1}^n w_i e_i$$

In cui:

$$w_i = \frac{m_i}{m}$$

E le variabili sono così definite:

- $n$ : numero di intervalli.
- $m$ : numero di record.
- $m_i$ : numero di record nell'intervallo  $i$ -esimo.

La *discretizzazione non-supervisionata* non utilizza alcuna informazione eccetto il valore dell'attributo continuo da discretizzare. Possono essere a loro volta divisi in due categorie:

- **Equal width unsupervised discretization:** gli intervalli in cui viene discretizzato hanno tutti la stessa ampiezza.
- **Equal frequency unsupervised discretization:** gli intervalli in cui viene discretizzato hanno approssimativamente la stessa frequenza.

Può succedere però che gli attributi categorici abbiano troppi valori. Se gli attributi in questione sono ordinali allora si applicano le tecniche viste in precedenza, se invece sono nominali allora bisogna trovare un altro approccio. In questo caso possiamo infatti raggruppare i valori solo se il raggruppamento si traduce in un miglioramento nelle performance di classificazione o nel raggiungimento di qualche obiettivo del trattamento dati.

#### 1.4.4 Variable transformation

**Definizione 1.17.** Si definisce **trasformazione di variabile** qualsiasi trasformazione applicata a tutti i valori di quella variabile.

Ci sono due tipi di trasformazione di variabile:

- **Funzioni semplici:** una semplice funzione matematica è applicata a tutti i valori di una variabile individualmente. Bisogna stare attenti all'ordine in cui vengono applicate e soprattutto a come si comporta la funzione per valori negativi e vicini allo 0.
- **Standardizzazione:** trasforma tutto il dataset in modo che acquisiscano una particolare proprietà.

$$Z = \frac{x - \mu}{\sigma}$$

E' molto importante perché il nuovo dataset viene trasformato in modo da avere  $\mu = 0$  e  $\sigma = 1$ . In questo modo la somma di diversi attributi continui permette ad uno o a pochi attributi di prendere grandi valori e di dominare il nuovo attributo somma. E' molto utile quindi perché fa saltare immediatamente all'occhio la presenza di outlier.

In questo caso la media è sostituita dalla mediana, e la deviazione standard è sostituita dalla AAD che ricordiamo essere definita come:

$$\sigma_x = \frac{1}{m} \sum_{i=1}^m |x_i - \mu|$$

## 2 Classification

### 2.1 Introduction (\*)

Entriamo nel dettaglio della componente di validazione nei modelli di classificazione supervisionata. La cosa più importante è **capire cosa stiamo facendo** in quanto quando si applicano algoritmi di machine learning è facile perdersi.

Nel solito dataset dei churn vogliamo determinare se un cliente abbandonerà o meno il nostro servizio; per farlo, cerchiamo di utilizzare una particolare combinazione di attributi. Bisogna innanzitutto vedere di che tipo sono le variabili presenti nel nostro database.

Lo scopo è di essere in grado sia di prevedere quando un cliente abbandonerà un determinato servizio, che di capire le **motivazioni** che l'hanno spinto a farlo. Questo tipo di compiti può essere portato a termine con l'utilizzo di un modello di **classificazione**. Un modello di classificazione è un modello che sfrutta alcuni attributi del dataset (**attributi esplicativi**) per prevedere un valore di un altro attributo (**attributo di classe**).



Figura 6: processo di classificazione

Forniamo di seguito delle definizioni utili per poter parlare più nel dettaglio dei metodi di classificazione:

**Definizione 2.1.** Si definiscono **variabili esplicative** le variabili in input.

**Definizione 2.2.** Si definiscono **variabili di classe** Le variabili in output.

**Definizione 2.3.** Si definisce **modello di classificazione** un modello in grado di risolvere un problema di classificazione; i modelli di classificazione si dividono in:

- **Modello descrittivo:** serve come strumento di spiegazione per distinguere tra oggetti di classi diverse
- **Modello predittivo:** predice la classe di un record sconosciuto, può essere visto come una scatola nera che assegna una label di una classe al record sconosciuto

**Definizione 2.4.** Si definisce **classificatore** l' approccio sistematico per costruire un modello di classificazione su un dataset.

La costruzione di un classificatore segue il seguente processo:

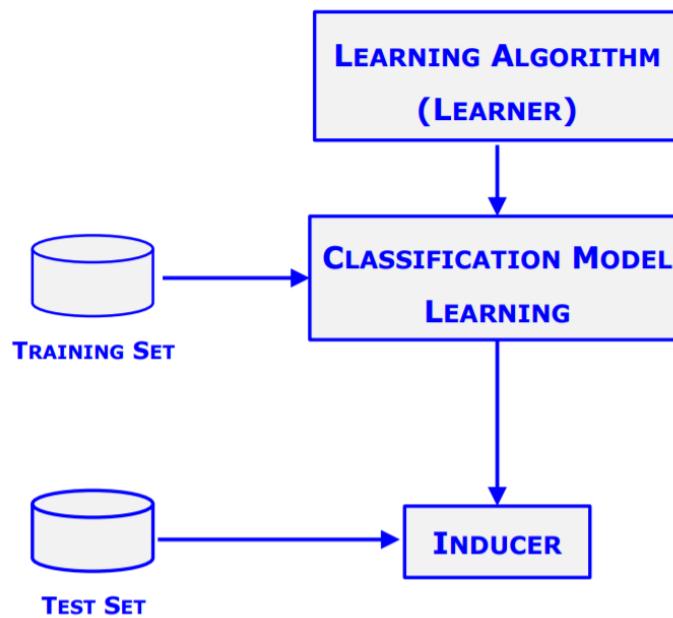


Figura 7: Processo di creazione della classificazione

**Definizione 2.5.** Si definisce **training set** l’insieme dei record in cui tutti i valori degli attributi sono noti.

**Definizione 2.6.** Si definisce **test set** l’insieme dei record i cui valori dell’attributo di classe sono sconosciuti (o presunti tali).

Si utilizza quindi un algoritmo di learning (**Learner**) sul training set, passaggio definito come **Classification Model Learning**.

**Definizione 2.7.** Si definisce **Inducer** l’output di questa operazione.

Il ruolo dell’inducer è quello di predire il valore dell’attributo di classe per il test set. La chiave di tutto è la scelta dell’algoritmo learner per l’apprendimento; la scelta degli attributi esplicativi è fondamentale, alcuni di essi possono, infatti, essere solo ridondanti nel processo e generare solo rumore. Tramite l’analisi delle performance si possono valutare queste due scelte.

Bisogna valutare le performance del modello inducer. Uno dei modi per analizzare se è efficace è la **matrice di confusione**.

Nelle righe vi sono i veri valori delle classi, nelle colonne i valori predetti dall’inducer. Il numero di righe è uguale al numero di colonne e corrispondono al numero di classi da predire.

		INDUCER PREDICTION (IP)	
		-1	+1
ACTUAL CLASS (AC)	-1	TN	FP
	+1	FN	TP

Figura 8: modello di matrice di confusione

All’interno sono identificati i risultati dei test e sono così classificati:

- **TN - vero negativo:** num. di record dove AC=-1 che sono stati correttamente predetti IP=-1
- **FN - falso negativo:** num. di record dove AC=+1 che sono stati erroneamente predetti IP=-1
- **TP - vero positivo:** num. di record dove AC=+1 che sono stati correttamente predetti IP=+1
- **FP - falso positivo:** num. di record dove AC=-1 che sono stati erroneamente predetti IP=+1

**Definizione 2.8.** La misura più utilizzata come metrica di performance è l'**accuratezza**:

$$\text{accuracy} = \frac{\sum_{i=1}^n \text{diagonal}}{\sum_{i=1}^n \text{elements}} = \frac{\#\text{CorrPrediction}}{\#\text{Prediction}} = \frac{TN + TP}{TN + TP + FN + FP}$$

La misura complementare che troveremo indicata è quella dell'errore:

$$\text{error} = \frac{FN + FP}{TN + FN + FP + TP} = 1 - \text{accuracy}$$

## 2.2 Tecniche di classificazione

Una tecnica di classificazione è un modo sistematico di aggredire un dataset e costruire i modelli di classificazione, in particolare possiamo dividerle in 4 macro categorie:

- **Euristiche:** ispezionano il suo vicinato come ad esempio: Decision Trees, Random Forest, Nearest Neighbor
- **Regression Based:** usa la probabilità condizionata parametrica, ad esempio: regressione logica
- **Separazione:** partiziona lo spazio degli attributi, fa riferimento alle Support Vector Machine e alle Artificial Neural Network
- **Probabilistici:** usano la formula di Bayes (Naive Bayes ecc...).

Forniremo una overview di tutte queste categorie senza trattarle nel dettaglio; partiamo dalla trattazione dei modelli **Euristicci**.

### 2.2.1 Decision Tree

I modelli decision tree sono modelli di tipo euristico.

Un **albero di decisione** ha una rappresentazione grafica precisa in cui gli elementi principali sono: nodi e archi. Il **nodo** rappresenta un sottoinsieme del dataset, gli **archi** sono usati, invece, per modellare gli output di modelli diversi di dataset.

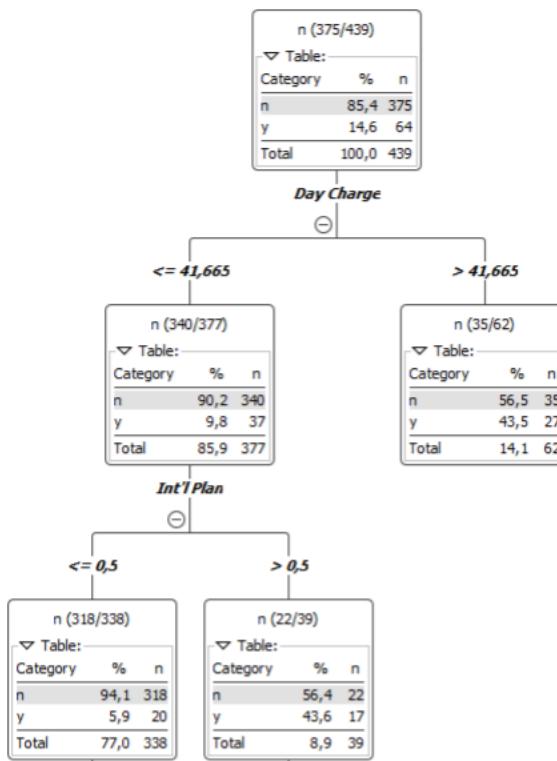


Figura 9: Modello decision tree

Gli elementi principali da cui è costituito sono:

- **Root node** (nodo radice): non ha archi in ingresso ma può averne più di due in uscita
- **Internal nodes** (nodi interni): hanno un solo arco in ingresso e almeno due in uscita
- **Leaf** (nodi foglia o terminali): sono nodi interni senza archi in uscita e con esattamente un arco in ingresso.

Ogni record viene classificato partendo **dall'alto** (radice) fino **al basso** (nodi foglia).

Leggo il primo nodo (radice) in cui è presente l'indicazione di un attributo: nell'esempio se 'day charge' sia  $\leq 41,665$  o  $>$  e divide l'albero in due nodi. Se la risposta è vera allora mi muovo in un nodo altrimenti nell'altro. All'interno del nuovo nodo ho di nuovo la valutazione di una variabile, e proseguo così finché raggiungo un nodo foglia. Quando si arriva ad una foglia devo fornire una risposta, ovvero conto la classe più frequente e rispondo al chiamante con essa (es. churn = n).

Il decision tree può essere usato per attributi nominali, ordinali così come per intervalli numerici e coefficienti.

Diverse misure possono essere usate per selezionare la politica di node splitting ottima: *entropia*, *indice di Gini* e il *classification error*. Esso comunque dipende anche dal tipo di attributo: *binario*, *nominale* e *continuo*.

E' un modello che risponde sempre *la classe più frequente ed è quindi inutile quando ho delle classi particolarmente sbilanciate*. Sono evidentemente test univariati, ciò che faccio è costruire iper parallelepipedi del nostro dataset. Vanno a definirsi delle rette per il cambio di classificazione: **decision boundary**.

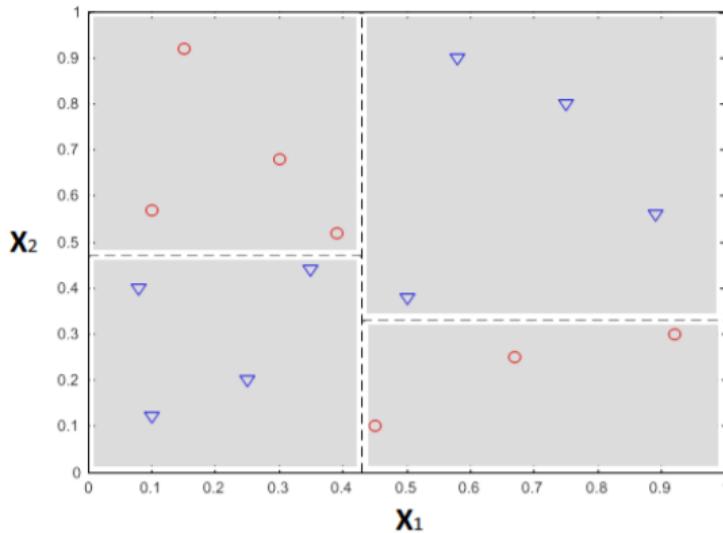


Figura 10: Decision boundary

Il risultato di queste rette fa la differenza sulla capacità di evidenziare dove siano presenti elementi di una certa classe, devo quindi imparare a posizionare questi iperpiani: **l'obiettivo è partizionare il mio spazio in iperpiani massimizzando l'accuratezza.**

Non c'è alcuna ragione per cui io non possa usare degli splitting multipli e non solo su binari, posso farlo anche su valori nominali.

Vi sono poi diversi modelli derivati dal decision tree, come il Random Forest.

**Random forest:** è un comitato di alberi di decisione. Usa degli attributi che sono in generale sottoinsiemi degli attributi, ogni albero può avere un sottoinsieme differente di attributi. Ogni albero usa attributi in modo casuale. Ogni albero apprende a modo suo e il random forest in base a dei parametri (regione dello spazio) decide a quale albero ascoltare per la decisione.

### 2.2.2 Regressione Logistica Binomiale

La regressione logistica è un metodo di classificazione basato sulla regressione.

Assumiamo che la variabile di classe  $Y$  sia compresa tra  $\{0, 1\}$ , allora la regressione logistica binaria calcola a *posteriori* la probabilità che  $Y$  dia il valore dell'input ovvero le variabili esplicative. Servono per risolvere problemi di regressione binaria a diversi livelli. È applicabile ad attributi continui e con certe accuratezze anche ad attributi nominali.

Si assume che l'attributo di classe sia tale che  $Y = \{0, 1\}$ ; il classificatore a regressione logistica binomiale calcola a posteriori la probabilità che  $Y$  assuma il valore di un input esplicativo  $\underline{X}$ . Si calcola come segue:

$$P(Y = 0 | \underline{X} = \underline{x}) = \frac{1}{1 + \exp(\underline{w} \cdot \underline{x})}$$

$$P(Y = 1 | \underline{X} = \underline{x}) = \frac{\exp(\underline{w} \cdot \underline{x})}{1 + \exp(\underline{w} \cdot \underline{x})}$$

dove  $\underline{w}$  è chiamato *vettore parametro*.

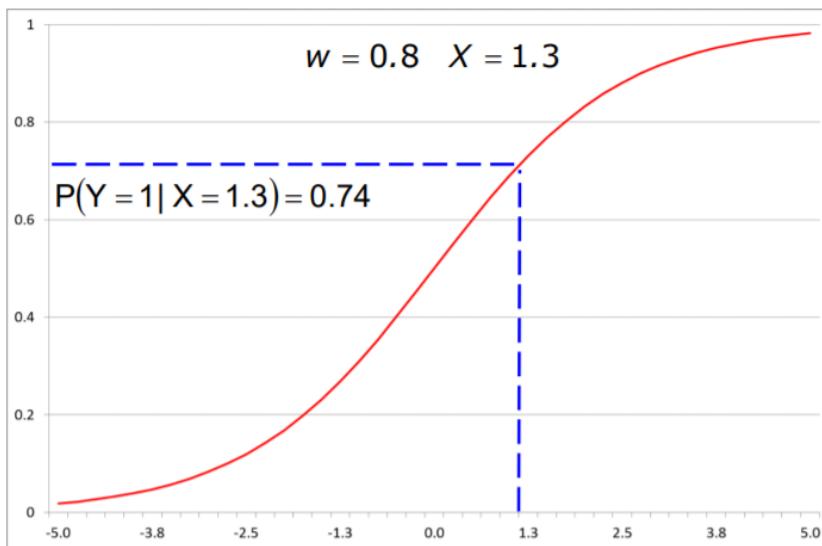


Figura 11: Esempio di regressione logistica

In questo modo vengono calcolate le probabilità di appartenere ad una certa classe.

### 2.2.3 Support Vector Machines

La tecnica Support Vector Machines è un metodo di classificazione con separazione. Lo scopo è separare o "apprendere" classi che vogliamo classificare.

Consideriamo uno spazio bidimensionale in cui sono rappresentati  $m$  record dove due attributi continui vengono misurati:  $D = \{(\underline{x}_1, y_1), \dots, (\underline{x}_m, y_m)\}$  in cui  $\underline{x}_i \in R^2$  e  $y_i \in \{-1, +1\}$

L'idea è quella di tracciare una retta (se ho due sole classi) per cui definisco l'area di appartenenza di una o dell'altra classe.

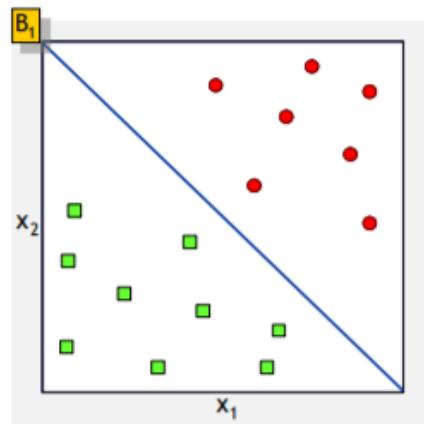


Figura 12: Esempio di una possibile retta di separazione

La retta in questione è definita dalla seguente equazione:

$$\underline{w} \cdot \underline{x} + b = w_1x_1 + w_2x_2 + b = 0$$

Per orientare la retta utilizziamo il vettore  $\underline{w} = [w_1, w_2]$  oppure  $b$ . ( $\underline{w}$  fa ruotare,  $b$  fa traslare)

se la retta esiste allora l'insieme delle istanze è *linearmente separabile*.

*Problema:* possono esserci più (anche infinite) rette utilizzabili per effettuare la separazione, quale devo scegliere? Perché non basta trovare una retta che funziona, ma la retta migliore in quanto l'algoritmo dovrà essere testato su dati nuovi. Si crea una sorta di area grigia nella quale non so dire esattamente quale delle due classi categorizzare (**Linear Decision Boundary**).

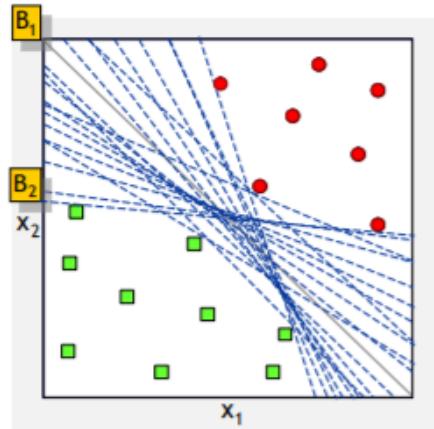


Figura 13: Diverse rette soluzioni dello stesso problema

Bisogna sostanzialmente trovare una retta che **massimizza il margine di errore**, l'*optimal linear decision boundary*.

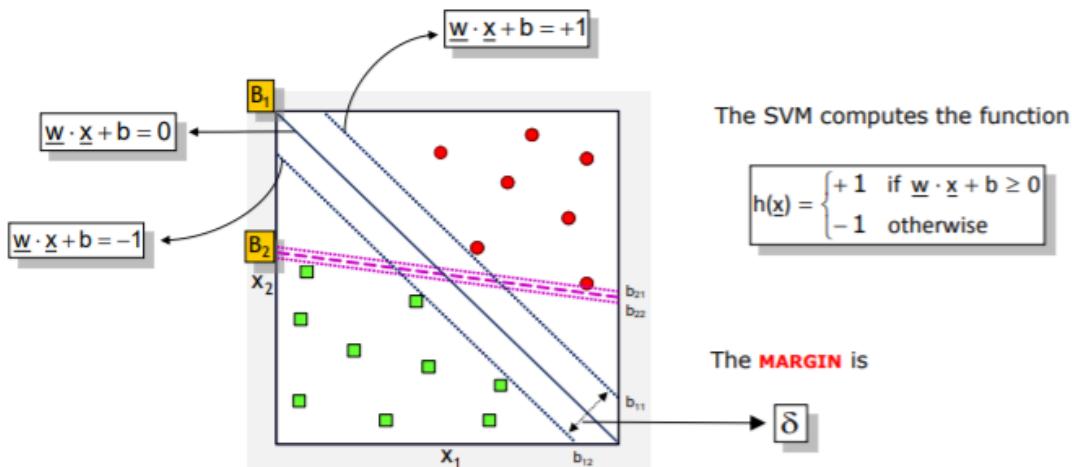


Figura 14: Boundary

La retta  $B_1$  è nettamente preferita rispetto alla retta  $B_2$  in quanto ha un margine di supporto  $\delta$  maggiore.

Naturalmente per  $n$  attributi bisogna trovare l'*iperpiano* ottimale per dividere un determinato insieme di istanze. Matematicamente parlando si cerca di massimizzare il margine  $\delta = 1/|\underline{w}|^2$ , dove la retta ha la seguente impostazione:  $\underline{w} \cdot \bar{x} + b = 0$ . Le rette ai confini del margine sono fissate a  $\underline{w} \cdot \bar{x} + b = 1$  e  $\underline{w} \cdot \bar{x} + b = -1$ .

L'argomento della retta è in 2 dimensioni, però dopo aver applicato la funzione  $h(\underline{x})$  quella retta diventa un piano che va a  $-1$  da un lato e a  $+1$  dall'altro.

Il training di una **SVM Linear Hard-margin** avviene formulando e risolvendo il seguente problema matematico:

$$\min_{\underline{w}, b} \frac{1}{2} \underline{w} \cdot \underline{w}^T$$

s.t.

$$y_i(\underline{w} \cdot \underline{x}_i + b) \geq 1 \quad \forall i = 1, \dots, m$$

È un problema di programmazione quadratica con vincoli lineari i quali devono essere risolti con tecniche numeriche speciali.

Per trovare la retta devo minimizzare l'inverso del margine  $\delta$ ; per farlo bisogna imporre dei vincoli per ogni attributo. In particolare, se  $\underline{w}$  e  $\underline{x}$  sono *concordi in segno* (positivo o negativo) allora classifica perfettamente perché il risultato è  $\geq 1$ . Questi vincoli garantiscono che tutti i casi del dataset siano classificati correttamente e tra tutti i casi che classificano correttamente scelgo quello che massimizza il margine.

Non risolveremo questa formula di ottimizzazione in modo diretto, ma la sua formulazione **duale**. In ogni caso questa formulazione funziona bene se il problema è *linearmente separabile*.

Nei casi in cui il problema **non** è linearmente separabili, non esiste **mai** una retta in grado di separare correttamente le classi. In questi casi la formulazione precedente non ammette soluzione, perché per alcuni dati i vincoli non ammettono soluzione.

Si introduce allora la **Linear Soft-margin**:

$$\min_{\underline{w}, b, \epsilon} \frac{1}{2} \underline{w} \cdot \underline{w}^T + \Delta \sum_{i=1}^m \epsilon_i$$

s.t.

$$\forall_{i=1}^m : y_i(\underline{w} \cdot \underline{x}_i + b) \geq 1 - \epsilon$$

$$\forall_{i=1}^m : \epsilon \geq 0$$

Le  $\epsilon$  devono essere non negative (variabili di slack), se utilizzo questo parametro per ammettere un errore allora esiste almeno una retta che risolve il problema di ottimizzazione. Ho sostanzialmente *rilassato* il problema di ottimizzazione, in particolare i vincoli. Graficamente parlando ciò si traduce nel traslare degli elementi di una classe diversa dalla regione di appartenenza verso la regione della classe di quell'elemento. E' molto importante notare che vettori di supporto sono quelle osservazioni che sono sul bordo del margine

**Problema:** La separazione si presta ad essere eseguita solo con una funzione **non lineare**.

**Soluzione:** Si va a cercare una trasformazione che porti dallo spazio originale in uno spazio delle features in cui posso applicare una separazione lineare. Nel nuovo spazio sono in grado di separare il nuovo dataset in due diverse classi (vedi immagine che segue).

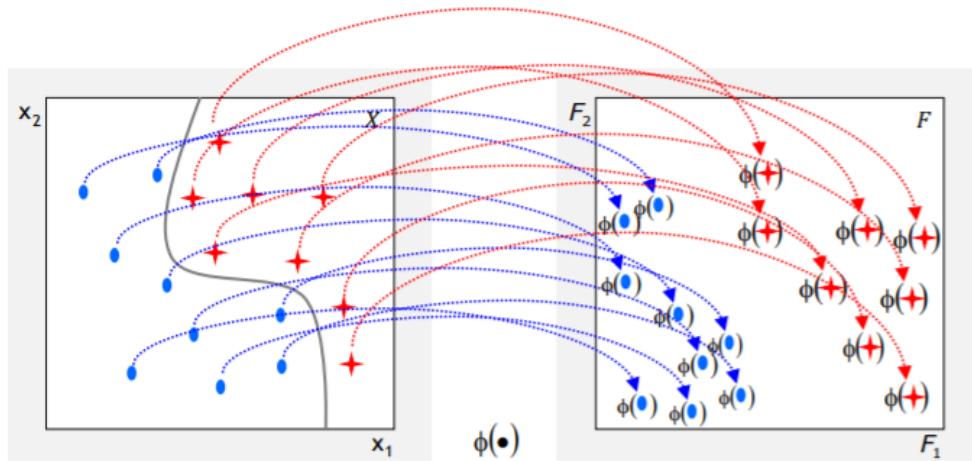


Figura 15: Applicazione di una separazione non lineare

In questo modo posso sfruttare tutta la metodologia precedente ma in uno spazio "controllato". Bisogna trovare però una trasformazione  $\phi(x)$  che mappi  $X$  in  $F$ . La nuova Linear Decision Boundary nello spazio delle feature  $F$  è definito dalla seguente equazione:

$$\underline{w} \cdot \phi(\underline{x}) + b = 0$$

Il modello diventa:

$$\min_{\underline{w}, b} \frac{1}{2} \underline{w} \cdot \underline{w}^T$$

s.t.

$$y_i(\underline{w} \cdot \phi(\underline{x}_i) + b) \geq 1 \quad \forall i = 1, \dots, m$$

Per l'algoritmo di apprendimento utilizziamo sostanzialmente delle funzioni kernel che sono del tipo:

$$K(\underline{u}, \underline{v}) = \phi(\underline{u}) \cdot \phi(\underline{v})$$

Queste funzioni kernel sono delle funzioni di similarità calcolate nello spazio attributi originale di  $x$  e sono riferite alla funzione kernel.

#### 2.2.4 Multi-Layer Perceptron o Artificial Neural Network

Il Multi-Layer Perceptron (MLP) è una tecnica di classificazione che si basa sulla separazione dello spazio degli attributi; queste tecniche sono oggi molto utilizzate per il deep learning.

**Definizione 2.9.** Si definisce **MLP Multi-Layer Perceptron** un modello che consiste in neuroni artificiali che comunicano in modo unidirezionale, dall'input  $X$  alla variabile di classe (volendo ci possono essere più neuroni di output).

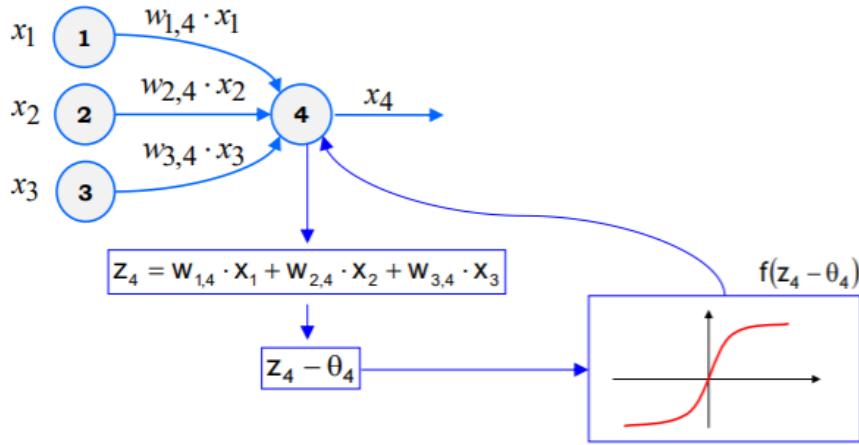


Figura 16: Esempio di MLP

In figura vi sono 3 neuroni di parametri continui a cui viene aggiunto un quarto neurone che calcola una funzione. Ad ogni input nel neurone è associato un peso  $w$ . Il neurone calcola una **combinazione lineare** tra gli input ed il peso dato ad esso, il  $j$ -esimo neurone calcola:

$$y_j = f\left(\sum_{i=1}^n w_{i,j} \cdot x_i - \theta_j\right)$$

Ad esempio il quarto neurone in figura vale:

$$z_4 = w_{1,4} \cdot x_1 + w_{2,4} \cdot x_2 + w_{3,4} \cdot x_3$$

A questo calcolo viene posto una valore di soglia pari a  $\theta$  applicato alla combinazione lineare. Successivamente, il neurone risponde con un valore pari all'applicazione di una funzione di attivazione rispetto all'input meno il threshold, nel nostro caso  $f(z_4 - \theta_4)$ .

**Definizione 2.10.** Si definisce **funzione di attivazione** la funzione applicata ad un neurone che restituisce il valore che verrà portato ad un altro strato con cui questo neurone comunica.

Storicamente le funzioni applicazione sono la tangente iperbolica e la funzione logistica (intervallo tra -1 e 1 e tra 0 e 1). Oggi vengono utilizzate delle funzioni non derivabili più complesse, come funzioni **RELU** (Rectify Linear Unit) che assumono valore 0 nel semiasse negativo e 1 nel semiasse positivo.

Vi sono 3 tipi di neuroni: di input, di output e nascosti:

- **Input** sono collegati con le variabili esplicative e con *ciascun* nodo nascosto.
- **Nascosti** (o hidden) ricevono i segnali dai neuroni di input e il segnale viene propagato a quelli di output.
- **Output** sono associati alla variabile di classe e ricevono i segnali da visualizzare.

Ogni neurone di input è connesso con ogni neurone di strato nascosto ed ogni nodo di strato nascosto comunica con il nodo di output, rete **fully-connected**.

Ogni arco ha associato un peso e ogni nodo ha un valore di soglia  $\theta_j$  e una funzione di attivazione.

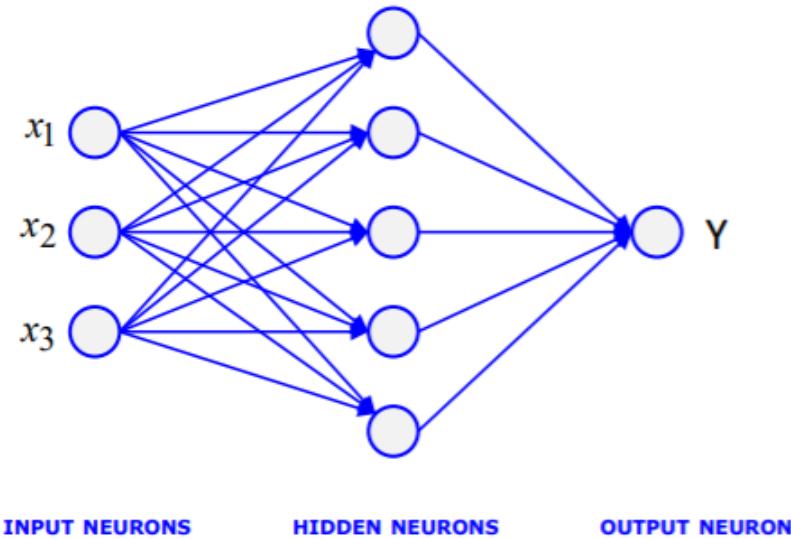


Figura 17: Modello MLP semplice

Per la disposizione dell'architettura ho diverse scelte da fare: quanti neuroni usare, quanti nello strato nascosto, quanti strati nascosti usiamo, che funzione di attivazione, ecc.

Determinare l'architettura della nostra rete è fondamentale per avere buone performance. Posso pensare di aggiungere un **livello** di neuroni nascosti. Non vi sono vincoli rispetto a comunicare saltando livelli. **Non** si può però comunicare all'indietro, infatti queste reti sono chiamate **feed-forward neural network** (possono possedere fino a centinaia di strati nascosti).

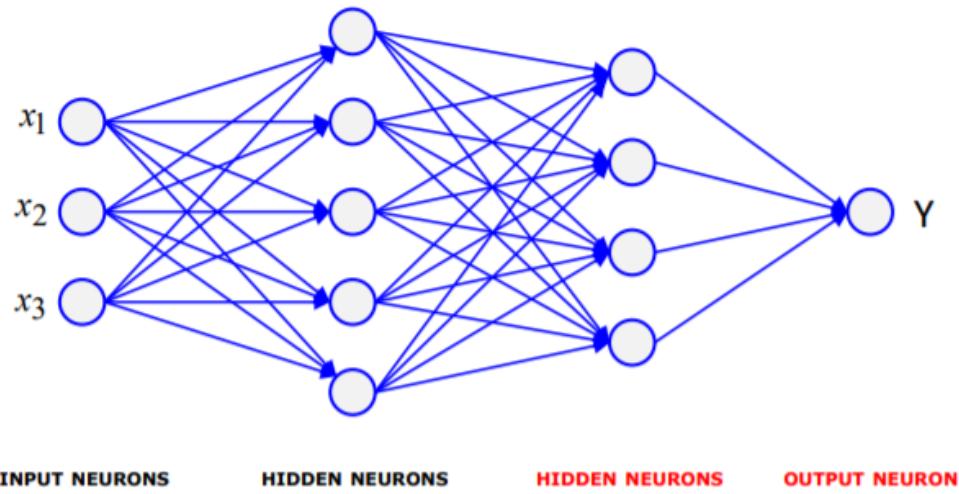


Figura 18: Esempio MLP con 2 strati nascosti

Per migliorare l'apprendimento si possono propagare le valutazioni fatte all'output per tutti i nodi risalendo fino all'input modificando i pesi degli archi. Questa cosa funziona bene e ha senso con le RELU.

Il problema MLP. non ha soluzione oggi, pertanto si procede in modo **empirico**, non è ancora possibile, infatti, arrivare ad una soluzione ottima, non si riesce a capire se si ha raggiunto il massimo/minimo globale ma si cerca quella

che dà risultati accettabili e migliori di altri. Il numero di nodi, archi o livelli nascosti è quindi scelto in base all'esperienza.

### 2.2.5 Classificatori bayesiani

I **classificatori probabilistici** calcolano la probabilità condizionata usando il **teorema di Bayes** e cercano di capire il valore della classe attributo dalle altre variabili.

**Definizione 2.11.** La **formula di Bayes** calcola la probabilità che un evento accada dati altri eventi:

$$P(Y|\underline{X}) = \frac{P(\underline{X}|Y) \cdot P(Y)}{P(\underline{X})}$$

Dove:

- $P(Y)$  è la probabilità dell'attributo di classe.
- $P(\underline{X}|Y)$  la verosimiglianza di un vettore di attributi dato l'attributo di classe.
- $P(\underline{X})$  probabilità dell'evidenza (nel senso di certezza).
- $P(Y|\underline{X})$  probabilità a posteriori dell'attributo di classe dato il vettore di attributi esplicativi.

Vediamone un esempio, assumiamo di avere:

- $Y$  variabile binaria di classe  $\{-1, +1\}$
- $\underline{X}$  variabile binaria esplicativa  $\{\text{Male}, \text{Female}\}$

Assumiamo di avere le seguenti probabilità:

$$P(Y) = (0.3, 0.7) \quad P(\underline{X}|Y = -1) = (0.2, 0.8) \quad P(\underline{X}|Y = +1) = (0.9, 0.1)$$

Vogliamo classificare  $\underline{X} = \text{Male}$  tramite la formula di Bayes:

$$P(Y = -1|\underline{X} = M) = \frac{P(\underline{X} = M|Y = -1) \cdot P(Y = -1)}{P(\underline{X} = M)} = \frac{0.06}{P(\underline{X} = M)}$$

$$P(Y = +1|\underline{X} = M) = \frac{P(\underline{X} = M|Y = +1) \cdot P(Y = +1)}{P(\underline{X} = M)} = \frac{0.63}{P(\underline{X} = M)}$$

In questo caso quindi la  $Y$  è più probabile che abbia valore  $+1$  se  $\underline{X} = \text{Male}$ .

Assumiamo ora che il numero di variabili esplicative aumenti all'aumentare  $n$  ed esse sono binarie, così come l'attributo di classe, allora avremo bisogno di conoscere i seguenti parametri:

$$\theta_{ki} = P(\underline{X} = x_k | Y = y_i) \quad k \in \{1, \dots, 2^n\}, y_i \in \{-1, +1\}$$

Quando il numero di variabili esplicative cresce, le devo binarizzare quindi avrò  $2^n$  parametri:

n	1	2	3	10	20	30
# parametri	2	4	8	1,024	1,048,576	1,073,741,824

Per risolvere il problema si usa l'**assunto della condizione di indipendenza**.

### 2.2.6 Naive bayes

**Definizione 2.12.** Dati gli attributi  $X, Y, Z$ , diremo che  $X$  è **indipendente condizionatamente** da  $Y$  dato  $Z$ , se e solo se la probabilità di  $X$  è indipendente dal valore dell'attributo  $Y$  una volta che  $Z$  sia noto, formalmente:

$$\forall i, j, k P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

Assumendo questa espressione per le variabili utilizzate si riduce enormemente il numero di parametri da calcolare; il **Naive Bayes** assume indipendenza condizionale e ci permette di calcolare la probabilità a posteriori dell'attributo classe dati gli attributi esplicativi come segue:

$$P(X_1, \dots, X_n | Y) = \prod_{i=1}^n P(X_i | Y)$$

$$2 \cdot (2^n - 1) \rightarrow 2 \cdot n$$

Il **Classificatore Naive Bayes** calcola la probabilità a posteriori dell'attributo di classe nel modo seguente:

$$P(Y = y_k | X_1, \dots, X_n) = \frac{P(Y = y_k) \cdot \prod_{i=1}^n P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \cdot \prod_{i=1}^n P(X_i | Y = y_j)}$$

Il record di variabili viene etichettato con il valore di classe che massimizza la probabilità a posteriori

$$\arg \max_{y_k} P(Y = y_k | X_1, \dots, X_n)$$

Il modello Naive Bayes fa parte dei modelli **grafico-probablistici**. Esiste un'intera famiglia di modelli di questo tipo: reti bayesiane dinamiche, ecc... non andremo nel dettaglio.

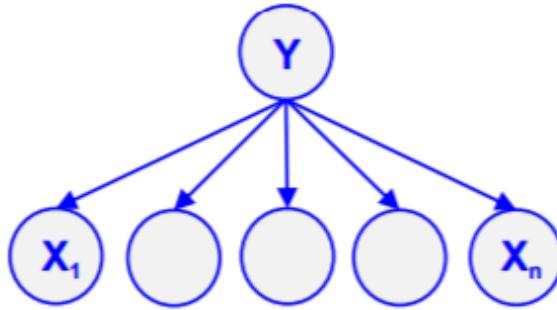


Figura 19: Rete naive bayes

A fianco del risultato output comunque è bene fornire la probabilità con il quale si è ottenuto il risultato, in modo da dare una *misura di affidabilità*.

Il classificatore Naive Bayes può essere applicato ad attributi categorici (nominali e ordinali), numerici (intervalli e tassi). Ad ogni attributo numerico è associata una densità di probabilità condizionale di classe normale:

$$P(X_i = x_i | Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu_{ik})^2}{2\sigma_{ik}^2}\right)$$

i cui parametri sono:

$$\mu_{ik} = E[X_i|Y = y_k] \quad \sigma_{ik}^2 = E[(X_i - \mu_{ik})^2|Y = y_k]$$

**NB:** solitamente si combinano attributi categorici (nominali e ordinali) con attributi numerici (intervalli e tassi).

Il modello Naive bayes si comporta generalmente bene ma richiede *una enorme premessa* per essere applicato correttamente, ovvero l' (**indipendenza condizionata**). Per ovviare a questo problema è stata creata una versione più flessibile mantenendo la stessa capacità computazionale.

### 2.2.7 Reti bayesiane

Il classificatore Naive bayes viene generalizzato dal modello di rete **Rete bayesiana** che è meno forzato dall'assunzione di indipendenza condizionata ma sfrutta il concetto di **Sparsity**.

Un esempio di rete bayesiana è il seguente:

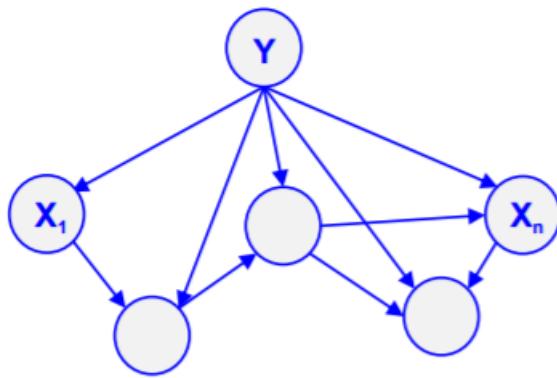


Figura 20: Esempio di rete bayesiana generalizzata

Gli attributi esplicativi sono associati tramite i nodi, ma non solo quelli con un arco in entrata dalla radice; gli attributi esplicativi possono puntare ad altri attributi esplicativi. Nel grafo **non** vi possono essere **cicli** in quanto un nodo successivo non può essere causa di un nodo precedente. Per ogni nodo bisogna specificare una tabella di probabilità condizionata rispetto al valore dei suoi genitori. In sostanza prendo in considerazione tutte le possibili configurazioni dei genitori e per ognuna do un output diverso.

In questo modello gli attributi esplicativi non sono più assunti come indipendenti dalla classe attributo.

Questo è un modello particolare di rete bayesiana che viene utilizzato per la classificazione. La cosa bella di questo modello è che anche con valori null si può comunque fare inferenza perché nativamente ha questa caratteristica.

*Nelle reti neurali non è possibile far computare un modello senza avere tutti gli input.*

### 2.2.8 Tree-augmented Naive Bayes

Vi sono diverse versioni di reti bayesiane, una di queste è il Tree-augmented Naive Bayes.

**Definizione 2.13.** Si definisce **Tree-augmented Naive Bayes** una rete che oltre ad avere il nodo genitore Y può permettersi di avere un altro nodo genitore (sempre).

Se io addestro la rete senza nodo di classe allora ho un albero, dopo inserendovi il nodo Y fa inferenza (nel caso in figura vedi nodo  $X_1$ ).

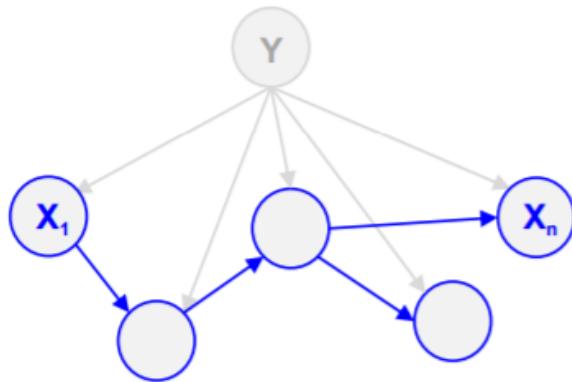


Figura 21: tree-augmented Naive bayes

È molto potente per la feature selection, ovvero per la ricerca delle feature più significative. Ci sono altre modifiche possibili al Tree-augmente naive bayes, presenti il letteratura, come AODE, HNB, ecc...

### 2.2.9 Summary

Per capire come *nativamente* si differenziano i modelli di classificazione è presente la tabella successiva.

	Explanatory Attribute				Class Attribute	
	Categorical			Numeric	Categorical	
	Nominal	Ordinal	Binary		Nominal	Binary
Decision Tree	X	X	X	X		X
Logistic Regression				X		X
SVM		X	X	X		X
MLP		X	X	X		X
NB	X	X	X	X	X	X

Figura 22: Comparazione tra i principali modelli nativi

In ogni caso, è possibile **trasformare** attributi esplicativi e di classe nominali applicando tecniche di classificazione non assegnate per attributi nominali. È da notare comunque che le tecniche di classificazione **non** sono l'opzione migliore quando bisogna predire un attributo **ordinale**.

## 2.3 Performance Evaluation (\*)

La stima dell'accuratezza non è sufficiente da sola per comprendere la confidenza di un modello, in particolare, quando bisogna applicarlo a dati mai visti. Bisogna rendersi conto del rischio di **overfitting** quindi magnificare alcuni risultati, e di **underfitting** ovvero non abbastanza accurati.

Vi sono due diversi tipi di *errori*:

- **Training error:** numero di record del training set mal classificato.
- **Generalization error:** errori su record no visti precedentemente (test set).

*Un buon classificatore non deve fittare troppo bene il training set ma deve anche classificare accuratamente i record che non ha mai visto prima.*

**Definizione 2.14.** Si parla di **Overfitting del modello** quando un modello di classificazione ottiene delle performance molto alte sul training set ma ha un generalization error molto alto.

Nella pratica: non bisogna adattare troppo il modello sui dati di training ma bisogna puntare ad un buon compromesso tra l'errore fatto sul training set e quello fatto sul test set.

Consideriamo il seguente esempio:

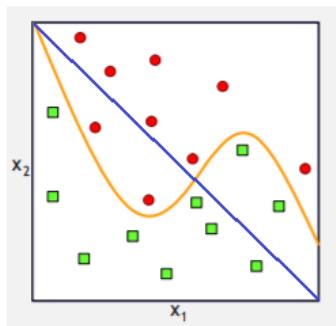


Figura 23: Esempio di overfitting

- Linea blu: modello con poco training error (non overfitting)
- Linea arancione: modello con zero training error (overfitting)

Come si può notare il modello **arancione** si adatta troppo ai dati di training quindi c'è un forte rischio che non classifichi bene i dati di test. Al contrario il modello **blu** segue meglio la tendenza della distribuzione delle due classi di elementi.

Ora vediamo come si comportano i due modelli nel test set

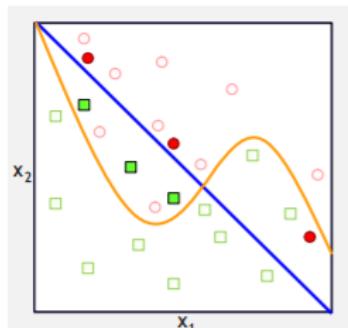


Figura 24: Esempio di overfitting test

Ora si può notare come il modello **blu**: più alto training error e più basso generalization error) classifichi **più correttamente** rispetto al modello **arancione**: più basso training error e più alto generalization error).

**Definizione 2.15.** Il fenomeno contrario è definito **Underfitting**, in cui sia il training error che il generation error sono elevati. Accade quando performance del training e test sono simili e basse.

La complessità del modello scelto non è sufficiente per i dati che sto studiando. Non abbiamo utilizzato tutta la flessibilità che potevamo usare nella definizione del modello.

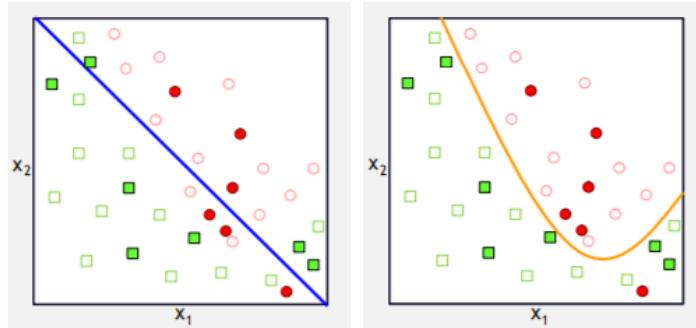


Figura 25: Esempio di underfitting (sinistra)

Guardando agli esempi in figura, il modello **blu** classifica erroneamente 6 record, per questo non è un buon modello. Invece, il modello **arancione** fitta meglio il training ed risponde meglio anche al test (1 errore), in questo caso però bisogna stare attenti a non incappare in overfitting.

La **soluzione ottimale** la si evince tenendo conto di quello che succede sia nei dati di training che nei dati di test.

*È impossibile avere una misurazione certa sugli errori del modello, si può solo avere una stima.*

### 2.3.1 Performance di un modello di classificazione

In una analisi di classificazione l'obiettivo è sviluppare un modello che dia una migliore classificazione possibile. Viene valutata in termini di:

- *Accuracy*
- *Speed*
- *Robustezza*
- *Scalabilità*
- *Interpretabilità* (tema più dibattuto)

### 2.3.2 Accuratezza

L'accuratezza è una misura fondamentale perché:

- Misura la capacità del modello nel dare classificazioni affidabili su nuovi record
- Permette di selezionare l'istanza di modello che fornisce le migliori performance sui nuovi record

Consideriamo:

$D_T$  training set con  $t$  record.

$D_{TS}$  test set con  $v$  record.

$$D = D_T \cup D_{TS}, D_T \cap D_{TS} = \emptyset, m = t + v$$

Un buon indicatore è la percentuale di record di test ( $D_{TS}$ ) che sono classificati correttamente.

Definiamo:

- $y_i$  come il valore della variabile di classe dell'istanza  $\underline{x}_i \in D_{TS}$
- $f(\underline{x}_i)$  valore della classe predetto per l'istanza  $\underline{x}_i \in D_{TS}$  dal modello di classificazione

Consideriamo quindi la seguente funzione di Loss:

$$L(y_i, f(\underline{x}_i)) = \begin{cases} 0 & \text{se } y_i = f(\underline{x}_i) \\ 1 & \text{se } y_i \neq f(\underline{x}_i) \end{cases}$$

L'**Accuratezza** viene calcolata in questo modo:

$$acc(D_{TS}) = 1 - \frac{1}{v} \sum_{i=1}^v L(y_i, f(\underline{x}_i))$$

In alcuni casi si preferisce l'**Errore** associato:

$$err(D_{TS}) = 1 - acc(D_{TS}) = \frac{1}{v} \sum_{i=1}^v L(y_i, f(\underline{x}_i))$$

### 2.3.3 Speed

Gli algoritmi di classificazione differiscono per:

- Tempo di apprendimento
- Spazio di memoria occupato

Oggi però si dà meno importanza a queste caratteristiche in quanto vi sono diverse tecnologie e tecniche per ottimizzare i tempi e lo spazio di memoria costa sempre meno (economicamente parlando).

Un algoritmo che impiega molto tempo e/o una grande quantità di memoria per l'addestramento, può essere trainato dopo un campionamento del dataset originale. In questi casi accettiamo di non sfruttare tutte le informazioni disponibili invece di trainare un tipo di modello che riteniamo assicurerà buone performance.

### 2.3.4 Robustezza

**Definizione 2.16.** Un modello/algoritmo può essere **robusto** o meno rispetto a:

- *Outliers*: possono influenzare significativamente il modello
- *Missing data*: problema centrale (i dati di base sono sporchi e soggetti a deperimento)
- *Variazione tra training set e test set*: si parte dal presupposto che i dati di training siano simili e utili per quelli di test, se questa affermazione decade non è più robusto l'algoritmo

### 2.3.5 Scalabilità

**Definizione 2.17.** Si definisce **scalabilità** la capacità di apprendere da enormi quantità di dati. Questa proprietà è intresecamente connessa alla speed.

Es. le reti bayesiane scalano molto male.

### 2.3.6 Interpretabilità

**Definizione 2.18.** Quando la classificazione è orientata dall'**Interpretabilità** di un problema per essere risolto non ci si può limitare ad assicurare alti valori di accuratezza, è importante estrarre regole semplici e chiare per l'esperto di dominio del caso di studio.

L'interpretabilità è un problema enorme, perché noi umani non ragioniamo in termini quantitativi, ma qualitativi. Una spiegazione può essere la più dettagliata e precisa possibile ma se non intercetta il linguaggio e il contesto in cui vive il soggetto che voglio raggiungere, non viene raggiunto l'obiettivo di far interpretare il lavoro.

### 2.3.7 Holdout

**Definizione 2.19.** Si definisce **Holdout** la partizione del dataset  $D$  in due sottinsiemi di training e test set attraverso un procedura di campionamento.

**Best practice** suggerisce 2/3 per il training set e 1/3 per il test set.

Se si hanno molti dati a disposizione non è necessario seguire questa proporzione si può aumentare sul training.

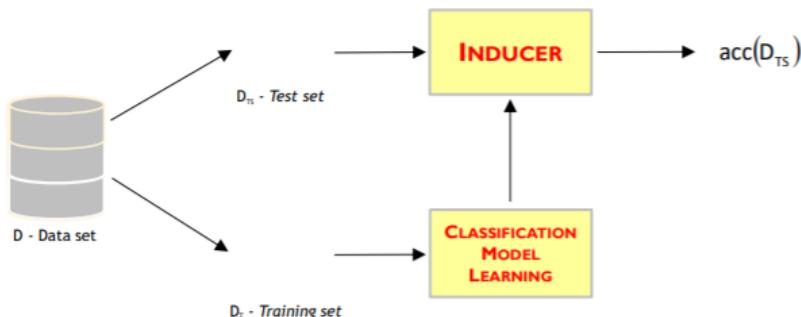


Figura 26: procedura di Holdout

*La stima dell'accuratezza dipende fortemente dalla particolare scelta di divisione tra training e test set. Vi sono alcune tecniche per rendere più efficiente la divisione.*

**Definizione 2.20.** Si definisce **Iterated Holdout** la tecnica che consiste nel ripetere iterativamente  $r$  volte il metodo di holdout per cui apprendo e stimo l'accuratezza.

Ad ogni iterazione  $r$  si estrae un campione random  $D_{Tr}$  di  $t$  record, ottenendo:

$$D_{TS_r} = D - D_{Tr}$$

Si ripete la procedura  $r$  volte e l'accuratezza del classificatore è stimata dalla media dei valori di accuratezza campionati  $acc(D_{TS_r})$  calcolati su ogni test set  $D_{TS_r}$ :

$$acc = \frac{1}{R} \sum_{r=1}^R acc(D_{TS_r})$$

Il numero di iterazioni  $r$  può essere scelto tramite specifiche tecniche statistiche.

*L'iterated Holdout è chiaramente un metodo più robusto, le sue performance hanno un bias più piccolo rispetto al normale holdout; tuttavia, non permette il controllo del numero di volte in cui un record è presente nel training e test set.* Non si ha la certezza di aver usato dati ottimi per la stima fatta. In alcuni casi si ricorre a uno schema più efficace, capace di ridurre l'impatto degli outlier.

**Definizione 2.21.** Si definisce **Cross-validation** la tecnica di holdout che garantisce che ciascun record di un dataset  $D$  sia incluso nel training set con lo stesso numero di volte ed esattamente una volta nel test set.

*Il dataset  $D$  viene partizionato in  $K$  sottoinsiemi disgiunti detti **fold**, esaustivi e con circa lo stesso numero di record:*

$$D_1, D_2, \dots, D_K$$

Eseguiamo  $K$  iterazioni training-test. Alla  $k$ -esima iterazione avremo:

$$D_{T_k} = \{D_1, \dots, D_{k-1}, D_{k+1}, \dots, D_K\} \quad D_{TS_k} = D_k$$

L'insieme  $D_{T_k}$  è usato per il training mentre il  $D_k$  viene usato per il test della  $k$ -esima iterazione, in pratica ho più training set e un test set.; l'accuratezza viene, quindi, stimata facendo la media delle  $K$  iterazioni:

$$acc = \frac{1}{K} \sum_{k=1}^K acc(D_k)$$

Esistono diverse selezioni per il valore di  $K$ . I valori tipici sono  $K = 3, 5, 10$  mentre un caso limite viene applicato quando i dati sono scarsi chiamato **Leave One Out Cross Validation (LOOCV)**. *LOOCV è una tecnica che assume che ogni record sia una partizione del dataset, così il valore di  $K$  equivale al numero di record del dataset  $D$ .*

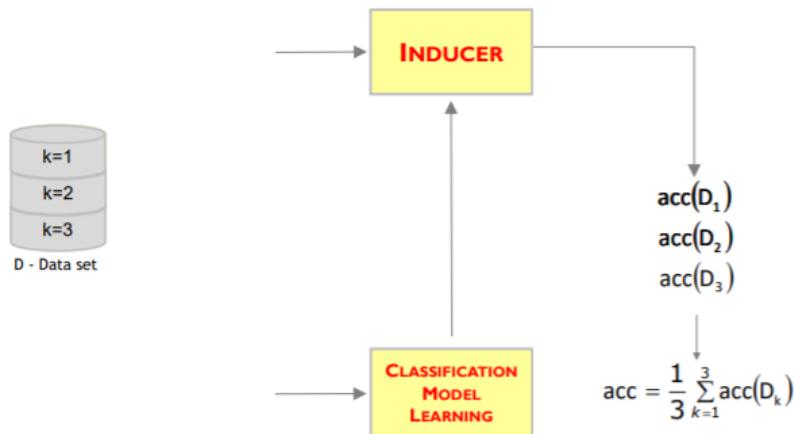


Figura 27: Procedura di cross validation

Solitamente ci si chiede se ogni partizione del dataset contenga la *stessa proporzione* di valori possibili della classe attributo. Quando le proporzioni sono fortemente sbilanciate si adottano specifiche tecniche di campionamento, es. **campionamento stratificato**. È buona norma fidarsi di più del **k-folds cross-validation** in quanto rispetto all'*iterated holdout* si ha la certezza che i record non siano ripetuti.

## 2.4 Comparing Classifiers (\*)

La comparazione tra due diversi *classificatori* non è semplice, perché dipende dai differenti modelli utilizzati e dalla distribuzione dei dati, nonché dalla loro quantità. L'accuracy ha diverso valore se ricavata da pochi record di dati. es.

- **Inducer A:** accuracy = 0.85 su un test set di 30 record
- **Inducer B:** accuracy = 0.75 su un test set di 5 000 record

Qual è il migliore?

*L'accuracy da sola non basta. Bisogna stimare l'intervallo di confidenza dell'accuracy per i due inducer e testare l'importanza statistica delle deviazioni osservate.*

### 2.4.1 Intervallo di confidenza

Consideriamo il problema di predire il valore di una classe attributo da un test record come esperimento **binomiale**.

Dato un test set  $D_N$  contenente  $N$  record, consideriamo:

- **X:** numero di record correttamente predetti dall'inducer
- **p:** vera, ma non la conosciamo, accuracy dell'inducer (probabilità di successo della distribuzione binomiale)

Modelliamo il problema con X distribuita secondo una binomiale con

- media =  $N \cdot p$
- varianza =  $N \cdot p \cdot (1 - p)$

L'oggetto del nostro studio è l'**accuratezza empirica** calcolata come:

$$acc = \frac{X}{N}$$

Che è distribuita secondo una binomiale con  $\mu = p$  e  $\sigma^2 = \frac{p \cdot (1-p)}{N}$

Sebbene la distribuzione binomiale possa essere usata per stimare l'intervallo di confidenza per l'accuratezza, se la dimensione del test set è sufficientemente grande, è buona norma approssimare la distribuzione con una **normale**. Così facendo l'intervallo di confidenza per l'**accuratezza empirica** diventa:

$$P\left(-Z_{1-\alpha/2} < \frac{acc - p}{\sqrt{p \cdot (1 - p/N)}} < Z_{1-\alpha/2}\right) = 1 - \alpha$$

Rimodellando, la diseguaglianza ci porta al seguente intervallo di confidenza con confidenza  $1 - \alpha$  per  $p$ :

$$\left[ acc + \frac{Z_{1-\frac{\alpha}{2}}^2}{2 \cdot N} - Z_{1-\frac{\alpha}{2}} \cdot \sqrt{\frac{acc}{N} - \frac{acc^2}{N} + \frac{Z_{1-\frac{\alpha}{2}}^2}{4 \cdot N^2}}, acc + \frac{Z_{1-\frac{\alpha}{2}}^2}{2 \cdot N} + Z_{1-\frac{\alpha}{2}} \cdot \sqrt{\frac{acc}{N} - \frac{acc^2}{N} + \frac{Z_{1-\frac{\alpha}{2}}^2}{4 \cdot N^2}} \right]$$

$$\left[ \frac{Z_{1-\frac{\alpha}{2}}^2}{(1 + \frac{Z_{1-\frac{\alpha}{2}}^2}{N})}, \frac{Z_{1-\frac{\alpha}{2}}^2}{(1 + \frac{Z_{1-\frac{\alpha}{2}}^2}{N})} \right]$$

**NB:** se prendessi diversi test set indipendenti l'uno dall'altro è ovvio che vengono diversi intervalli di confidenza. Però in questi casi fissato  $\alpha$  posso stabilire statisticamente quali intervalli siano più significativi di altri. Più si riduce  $\alpha$  più aumenta  $\beta$  ovvero il caso di errore in cui non viene rifiutata l'ipotesi nulla quando invece dovrei rifiutarla.

Dato un modello con  $acc=0.8$  sul test set di 100 record, fissiamo l'intervallo di confidenza per l'accuratezza a 0.95%. L'andamento dell'intervallo è il seguente:

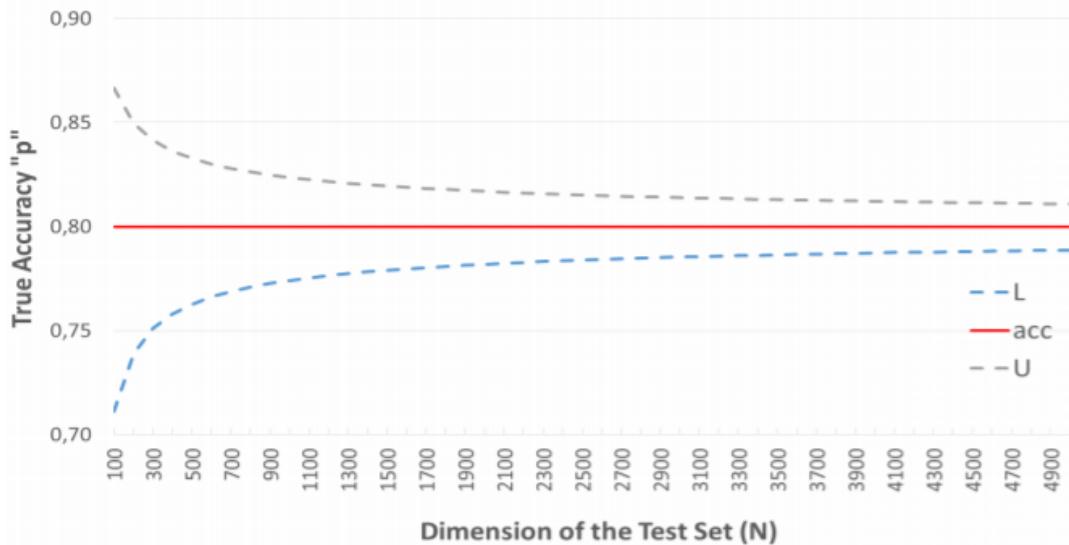


Figura 28: Andamento dell'intervallo di confidenza

#### 2.4.2 Different test set

Supponiamo il caso in cui sono presenti 2 modelli e vengono valutati su due test set diversi (è una cosa molto comune e che può accadere per numerevoli ragioni):

- $M_1$  valutato su test set  $D_1$  contenente  $n_1$  record con tasso di errore  $e_1$
- $M_2$  valutato su test set  $D_2$  contenente  $n_2$  record con tasso di errore  $e_2$

I due test set devono essere assunti *indipendenti*. Il nostro obiettivo è testare se la **differenza** tra i due errori è statisticamente significativa. Assumendo che  $n_1$  e  $n_2$  siano sufficientemente grandi, allora i tassi di errore  $e_1$  ed  $e_2$  possono essere approssimati usando distribuzioni *normali*.

La differenza osservata è denotata come:

$$d = e_1 - e_2$$

Sono distribuiti secondo una **normale** con

$$\mu = d_t \quad \sigma^2 = \sigma_d^2$$

La varianza di  $d$  può essere stimata come segue:

$$\sigma^2 \cong \hat{\sigma}_d^2 = \frac{e_1 \cdot (1 - e_1)}{n_1} + \frac{e_2 \cdot (1 - e_2)}{n_2}$$

L'intervallo di confidenza per la differenza  $d_t$  è:

$$(d - z_{1-\alpha/2} \cdot \hat{\sigma}_d, d + z_{1-\alpha/2} \cdot \hat{\sigma}_d)$$

Ci possono essere sostanzialmente 3 casi:

1.

$$0 \in (d - z_{1-\alpha/2} \cdot \hat{\sigma}_d, d + z_{1-\alpha/2} \cdot \hat{\sigma}_d)$$

In questo caso si conclude che la differenza osservata **non è statisticamente significativa** ad un livello  $\alpha$ , i modelli non sono significativamente differenti

2.

$$d + z_{1-\alpha/2} \cdot \hat{\sigma}_d < 0$$

Se il limite superiore della confidenza è negativo allora il modello  $M_1$  è **migliore del modello  $M_2$**  a un livello  $\alpha/2$

3.

$$d - z_{1-\alpha/2} \cdot \hat{\sigma}_d > 0$$

Se il limite inferiore della confidenza è positivo allora il modello  $M_2$  è **migliore del modello  $M_1$**  a un livello  $\alpha/2$

È buona norma fare il test di ipotesi **solo** se prima mi faccio la domanda se siano o meno differenti, non bisogna condurre test a casaccio o a tappeto perché altrimenti si rischia di raggiungere dei risultati assurdi. Bisogna porsi la domanda e poi eseguire tutta la procedura sulla coppia di modelli, in quanto vi è un **problema dei confronti multipli** devo adattarli tutti allo stesso livello di  $\alpha$  altrimenti le differenze non sono confrontabili.

### 2.4.3 Same test set

Nel caso in cui sia possibile utilizzare lo **stesso test set** si può svolgere un test più **potente**; ovvero, si può valutare il caso di errore di secondo tipo: si afferma che la differenza tra i due classificatori non è significativa quando in realtà lo è ( $\beta$ ).

Si confrontano  $M_1$  ed  $M_2$  usando il *k-fold cross validation*. Ho il dataset  $D$  partizionato in  $K$  subset disgiunti con circa lo stesso numero di record.

$$D_1, D_2, \dots, D_K$$

Si applicano le tecniche di classificazione per costruire i modelli  $M_1$  e  $M_2$  dalle  $k - 1$  partizioni e si testano con la partizione rimanente. Il passaggio deve essere ripetuto  $K$  volte, ogni volta usando una differente partizione per il test set. Si ottiene:

- $M_{1k}$  **inducer** per il modello  $M_1$  ottenuto alla k-esima iterazione con  $e_{1k}$  errore
- $M_{2k}$  **inducer** per il modello  $M_2$  ottenuto alla k-esima iterazione con  $e_{2k}$  errore

La differenza tra gli errori durante la k-esima iterazione è:

$$d_k = e_{1k} - e_{2k}$$

Per  $K$  sufficientemente grande, allora  $d_k$  è distribuito *normalmente* con:

$$\mu = d_t^{cv} \quad \sigma = \sigma^{cv}$$

Dove la varianza osservata è stimata utilizzando la seguente formula:

$$\hat{\sigma}_{d^{cv}}^2 = \frac{\sum_{k=1}^K (d_k - \bar{d})^2}{K \cdot (K - 1)} \quad \bar{d} = \frac{1}{K} \sum_{k=1}^K d_k$$

Usiamo una distribuzione T di Student per calcolare l'intervallo di confidenza per il valore della vera media  $d_t^{cv}$

$$\left( \hat{d} - t_{1-\frac{\alpha}{2}}^{K-1} \cdot \hat{\sigma}_{d^{cv}}, \hat{d} + t_{1-\frac{\alpha}{2}}^{K-1} \cdot \hat{\sigma}_{d^{cv}} \right)$$

Dove

$$t_{1-\frac{\alpha}{2}}^{K-1}$$

È ottenuta dalla tavola di probabilità, il quantile associato alla confidenza  $1 - \alpha$  e  $K - 1$  gradi di libertà.

Ovviamente valgono le **stesse considerazioni** fatte precedentemente sull'intervallo di confidenza per la differenza tra due classificatori: se l'intervallo di confidenza contiene il valore 0, concludiamo che la differenza osservata *non è statisticamente significativa* al livello  $\alpha$  (confidenza  $1 - \alpha$ ).

## 2.5 Class Imbalance Problem (\*)

Consideriamo di nuovo il dataset dei Churner. In particolare è noto che il 14.5% sono dei churner. Un modello di classificazione che etichetta i record come non churner avrà per forza un'alta accuracy in quanto è la classe più frequente nel dataset.

**Definizione 2.22.** Si definisce la **ZeroR Rule** il modello che risponde sempre con la classe più frequente, risultando, quindi, il modello più inutile.

La misura di *accuracy* tratta classi equamente importanti, non è adatta ad analizzare **dataset sbilanciati**, dove la classe più rara è considerata *più interessante* di quella più frequente. Per la classificazione binaria si distingue:

- La **classe positiva** ovvero la classe più rara (o meno frequente)
- La **classe negativa** ovvero la classe più frequente

Consideriamo la **matrice di confusione**.

		INDUCER PREDICTION (IP)	
		-1	+1
ACTUAL CLASS (AC)	-1	TN	FP
	+1	FN	TP

Figura 29: Matrice di confusione binaria

Dalla matrice calcoliamo i seguenti indicatori:

- **TNR, specificità** (*tasso veri negativi*): frazione di record negativi correttamente predetti dal modello

$$TNR = \frac{TN}{TN + FP}$$

- **TPR, sensitività** (*tasso veri positivi*): frazione dei record positivi correttamente predetti dal modello

$$TPR = \frac{TP}{TP + FN}$$

- **FPR, tasso falsi positivi**: frazione dei record negativi predetti come classe positiva dal modello

$$FPR = \frac{FP}{TN + FP}$$

- **FNR, tasso falsi negativi**: frazione dei record positivi predetti come classe negativa dal modello

$$TNR = \frac{FN}{TP + FN}$$

Le metriche che si calcolano nella ricerca di una classe più importante delle altre sono la **Precision** e il **Recall**.

**Definizione 2.23.** La **Precision** determina la frazione di record che effettivamente si rivela essere positiva nel gruppo che il classificatore definisce come classe positiva.

$$p = \frac{TP}{TP + FP}$$

Si prende il punto di vista del classificatore. Valuto quanti di quelli positivi ha valutato come effettivamente positivi. Più è *alta* la precisione più è *basso* il numero di falsi positivi commessi.

**Definizione 2.24.** L'indice **Recall** misura la frazione di record positivi correttamente predetti dal modello di classificazione; in particolare, è definito come:

$$r = \frac{TP}{TP + FN}$$

In questo caso si prende il punto di vista della realtà. Al denominatore si hanno quelli effettivamente positivi, al numeratore ho quelli che il classificatore ritiene positivi. Un *alto* Recall significa *pochi* record positivi *erroneamente* classificati come classe negativa. Infatti, il Recall è equivalente al *TPR*.

Vediamo un esempio sul problema del churn:

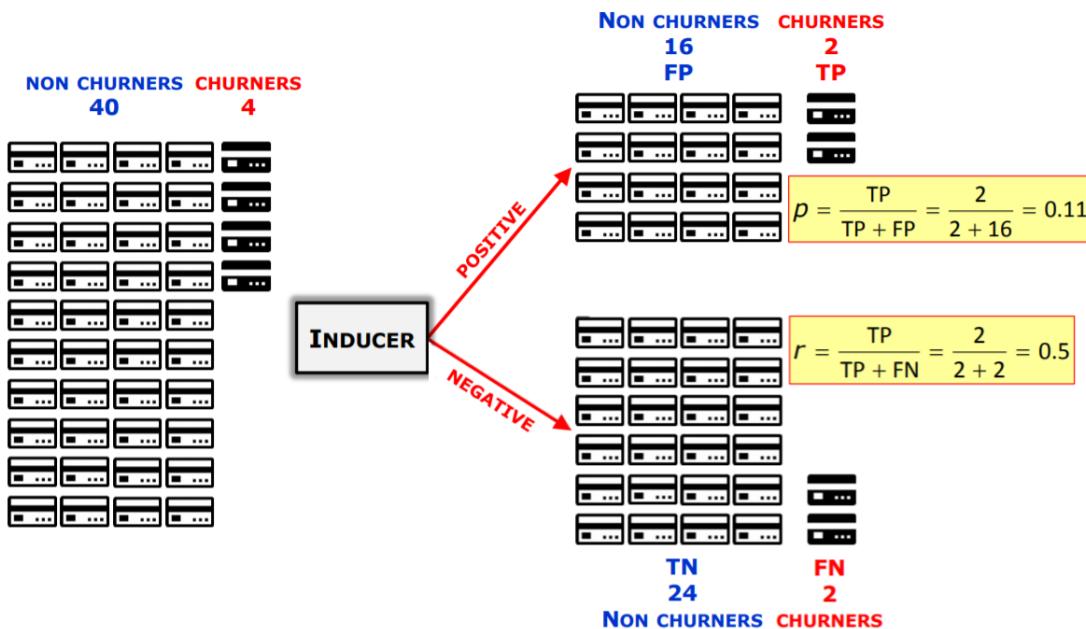


Figura 30: Esempio calcolo precision e recall

Vi è un *legame molto stretto* tra questi due indici, si può avere un recall molto alto ma allora probabilmente avrà una precision bassa. Oppure si può puntare ad avere una precisione alta ma in quei casi probabilmente si avrà una recall bassa.

Queste due misure per convenzione sono calcolate sulla classe positiva (quella meno frequente) non è però detto che non vadano utilizzate su quella negativa. Inoltre non è sempre detto che queste quantità siano sempre calcolabili (possibile divisione 0/0).

Siccome queste due misure sono legate (pensa alla problema della coperta corta). Si utilizza una misura che le riassume.

**Definizione 2.25.** Si definisce  $F_1$  **measure** la media armonica tra **Recall** e **Precision**.

$$F_1 = \frac{2 \cdot r \cdot p}{r + p}$$

Per come è definita un alto valore di  $F_1$  implica alti valori di recall e precision. Esiste una generalizzazione sotto il nome di  $F_\beta$  **measure** per esaminare il tradeoff tra Precision e Recall:

$$F_\beta = \frac{(\beta^2 + 1) \cdot r \cdot p}{r + \beta^2 \cdot p}$$

- $F_\beta$  con  $\beta = 0$  è la Precision
- $F_\beta$  con  $\beta = \infty$  è la Recall

## 2.6 Counting the cost (\*)

### 2.6.1 Matrice di costo

Se pensiamo al nostro problema del churner, l'azienda ha interesse ad evitare quello che si prevede, ovvero che il cliente se ne vada, in sostanza si cerca di prevedere chi vuole andarsene e si cerca di evitare che questa previsione sia effettiva applicando politiche di dissuasione. L'azienda può spendere un tot budget per invertire questo fenomeno, bisogna fare il meglio per identificare i possibili churner e attuare **solo** su di loro le politiche di dissuasione (se lo facessi per tutti i clienti non avrebbe senso per quanto riguarda i costi).

Bisogna convincere il proprio interlocutore che il modello sviluppato sia migliore rispetto al modello che storicamente hanno utilizzato. La misura più utilizzata per questo è l'accuracy legata alla matrice di confusione. In aggiunta, però, bisogna associarci la **matrice di costo**.

**Definizione 2.26.** La **Matrice di Costo** stabilisce in base a falsi positivi e negativi quanto costo (in soldi/lavoro) sostiene l'azienda per classificare un certo soggetto (es. come churner o non churner).

Il costo è così calcolato:

		PREDICTED CLASS	
		-1	+1
ACTUAL CLASS	-1	TN	FP
	+1	FN	TP

		PREDICTED CLASS	
		-1	+1
ACTUAL CLASS	-1	p	q
	+1	q	p

$$N = TP + FN + FP + TN$$

$$\text{ACCURACY} = (TP+TN)/N$$

$$\begin{aligned}
 \text{COST} &= p \cdot (TP+TN) + q \cdot (FN+FP) \\
 &= p \cdot (TP+TN) + q \cdot (N - TP - TN) \\
 &= q \cdot N - (q-p) \cdot (TP+TN) \\
 &= N \cdot [q - (q-p) \cdot \text{ACCURACY}]
 \end{aligned}$$

Figura 31: Legame tra matrice di confusione (sn) e matrice di costo (dx)

$$Cost = C_{--} \cdot TN + C_{-+} \cdot FP + C_{+-} \cdot FN + C_{++} \cdot TP$$

**NB** se matrice di costo è simmetrica allora il costo corrisponde all'accuratezza; per comprenderlo, vediamo il seguente esempio. Confrontiamo le performance di accuracy e costo tra lo *Standard Mailout Procedure* (**SMP**), procedura già utilizzata per dissuadere i clienti, con il modello nuovo di classificazione **M** da noi costruito:

		PREDICTED CLASS	
		-1	+1
ACTUAL CLASS	-1	830	111
	+1	45	114
<b>ACCURACY = 0.858</b>			

		PREDICTED CLASS	
		-1	+1
ACTUAL CLASS	-1	931	10
	+1	57	102
<b>ACCURACY = 0.939</b>			

Figura 32: Matrici di confusione: a sinistra SMP, a destra il modello M

Ora, però, bisogna associarci la **matrice di costo** per le rilevazioni e ne risulta:

		PREDICTED CLASS	
		-1	+1
ACTUAL CLASS	-1	0	1
	+1	100	-1
<b>COST = 4,497</b>		<b>COST = 5,608</b>	

Figura 33: Matrice di costo churner

Come si può notare, nonostante il nostro modello abbia una misura di **accuracy** migliore, ha un **costo** (in termini di soldi) superiore rispetto al modello storicamente utilizzato dall'azienda. Pertanto, in questo caso, si preferirà la vecchia procedura SMP piuttosto che la nostra soluzione. È importante notare però che ci sono situazioni nelle quali perdere un cliente è più grave che perderne un altro. Devo capire se sto utilizzando la matrice dei costi vera, oppure, se è solo rappresentativa. Se essa non è certa posso ragionare in un altro modo.

### 2.6.2 Cumulative Gains

Supponiamo che ci sia una popolazione di 1.334 clienti di cui 500 sono possibili churners. Per esperienza accumulata sappiamo che il 15% dei clienti è un chunner, quindi:

$$1334 * 0.15 = 200 \text{ chunner sul totale};$$

$$500 * 0.15 = 75 \text{ chunner sui potenziali churners}.$$

Se si considerasse positiva la classe dei churners e negativa quella dei non churners e si provasse un campionamento casuale su di essi si avrebbe questo risultato:

$$\begin{array}{ccccc} & -1 & +1 & & \\ \text{RANDOM SAMPLING} & -1 & 709 & 425 & \rightarrow 37.5\% \\ & +1 & 125 & 75 & 75/(125+75) \end{array}$$

Il modello di classificazione sviluppato, se applicato, identifica sui potenziali churner il 60% di chi effettivamente se ne va:

$200 * 0.6 = 120$  churners correttamente identificati.

CLASSIFICATION	-1	754	380	+1			60%
MODEL M	+1	80	120		→	$120/(80+120)$	

Pertanto possiamo affermare che il nostro modello sia molto meglio della *zeroRule*, ovvero il campionamento casuale.

**Definizione 2.27.** Si definisce **Lift Factor** il rapporto tra il modello  $M$  rispetto al campionamento casuale:

$$Lift = \frac{performance(M)}{performance(zeroRule)}$$

Questo coefficiente è usato per calcolare l'incremento di performance tra due modelli. Nel nostro caso:  $0.6/0.375 = 1.6$

**NB:** si considera che il nostro classificatore è bravo ad identificare i casi semplici ma rischia di sbagliare di molto in quelli complessi. Man mano che viene forzato a rispondere lui tenderà a rispondere in modo sempre più random.

Si può ora sfruttare il *Lift factor* per comprendere il livello di **profitabilità** una volta che i costi coinvolti sono noti. In questo caso bisogna avere un sottoinsieme di customers che hanno *alta proporzione* di record positivi, *maggiori* rispetto al dataset di partenza.

1. **Ordinare** gli output con probabilità di riconoscimento corretto **prob(y)** in ordine decrescente
2. **Estrarre** un primo sottoinsieme (partendo dall'alto) e calcolare il *lift factor*, per forza di cose sarà alto
3. **Considerare** allora un sottoinsieme più grande (sempre partendo dal primo valore) e calcolare il *lift factor*, ci si aspetta un valore minore rispetto al precedente
4. **Ripetere** il punto 3 finché si includono tutti i record

Nell'esempio in figura due passaggi del procedimento

Rank	RowID	Prob(y)	Churn	Rank	RowID	Prob(y)	Churn
1	134	0.95	y	1	134	0.95	y
2	221	0.88	y	2	221	0.88	y
3	18,923	0.86	n	3	18,923	0.86	n
4	90,034	0.73	n	4	90,034	0.73	n
5	874,823	0.64	n	5	874,823	0.64	n
6	67	0.47	n	6	67	0.47	n
7	98,324	0.32	n	7	98,324	0.32	n
8	2,553	0.15	y	8	2,553	0.15	y
9	289	0.10	n	9	289	0.10	n
10	2,349	0.03	n	10	2,349	0.03	n

subset consisting of 3 records  
2 records out of the 3 positive ones  
are correctly identified  
 $2/3 = 0.66$  of positive records  
 $\text{Lift} = 0.66/0.3 = \textcolor{red}{2.22}$

subset consisting of 5 records  
2 records out of the 3 positive ones  
are correctly identified  
 $2/3 = 0.66$  of positive records  
 $\text{Lift} = 0.66/0.5 = \textcolor{blue}{1.33}$

Figura 34: Calcolo del **lift factor** primo sottoinsieme (sinistra) e secondo sottoinsieme (destra)

**Definizione 2.28.** I valori di Lift factor così calcolati vanno a formare la cosiddetta curva dei **Cumulative Gains**.

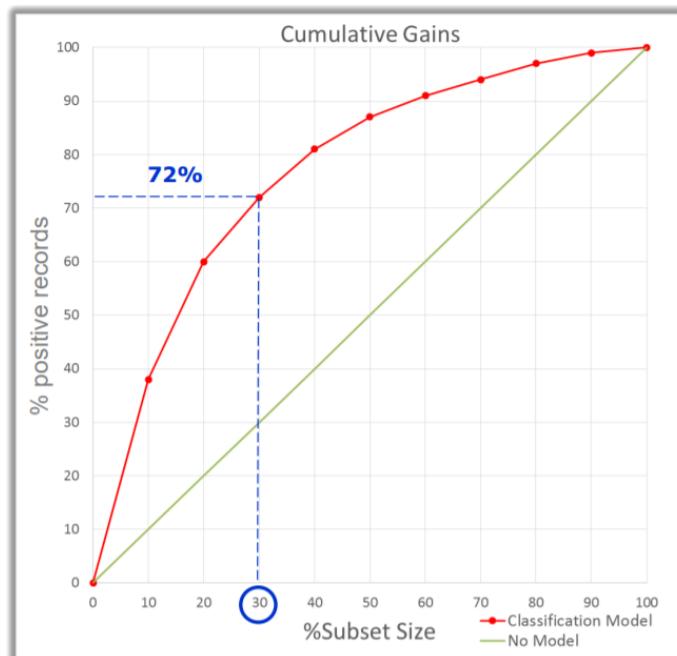


Figura 35: Cumulative gains evidenziando un punto di alto valore aggiunto

**NB:** La retta verde calcola la cumulative gain per un modello che risponde in maniera *casuale*, sull'asse x è la *recall*, i positivi riconosciuti correttamente (ripetendo il test ad ogni step scelto)

### 2.6.3 Lift Chart

La curva *Cumulative Gains* può essere mappata direttamente nella **Lift Chart**.

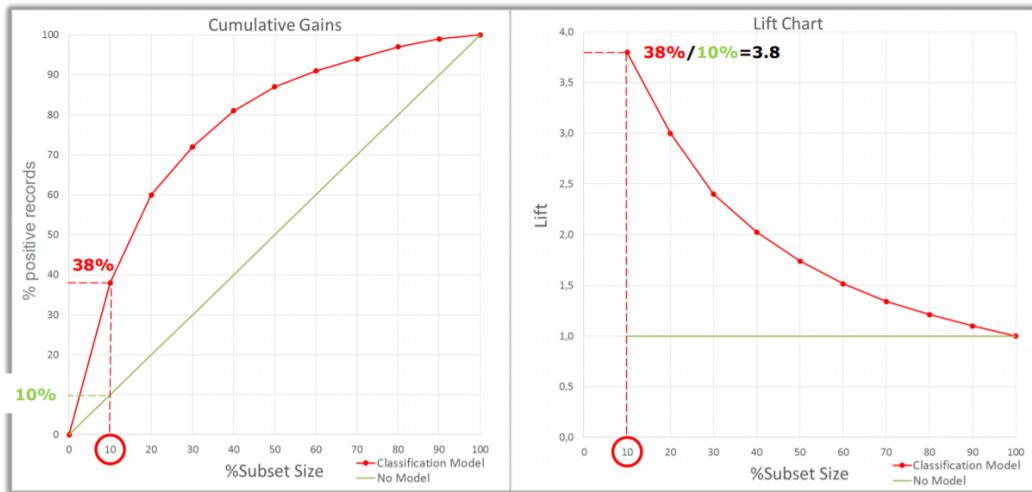


Figura 36: Da cumulative gain a lift chart

Queste valutazioni permettono di comprendere *quando* il classificatore perde efficacia. Naturalmente più il campione è piccolo meglio funziona, in quanto si usa record con **prob(y)** di riconoscimento corretto molto alta, bisogna comprendere però il punto di massima accuratezza in rapporto alla percentuale di istanze.

#### 2.6.4 Curva ROC

**Definizione 2.29.** È chiamata **ROC** e sta per Receiver Operating Characteristic curve la tecnica grafica per la valutazione di modelli di classificazione.

Assomiglia molto alla cumulative gains ma sugli assi sono presenti:

- **asse x:** la percentuale dei record falsi positivi sopportabili (**FPR**)
- **asse y:** la percentuale dei record veri positivi (**TPR**)

Entrambi i valori sono espressi in *percentuale sul totale*.

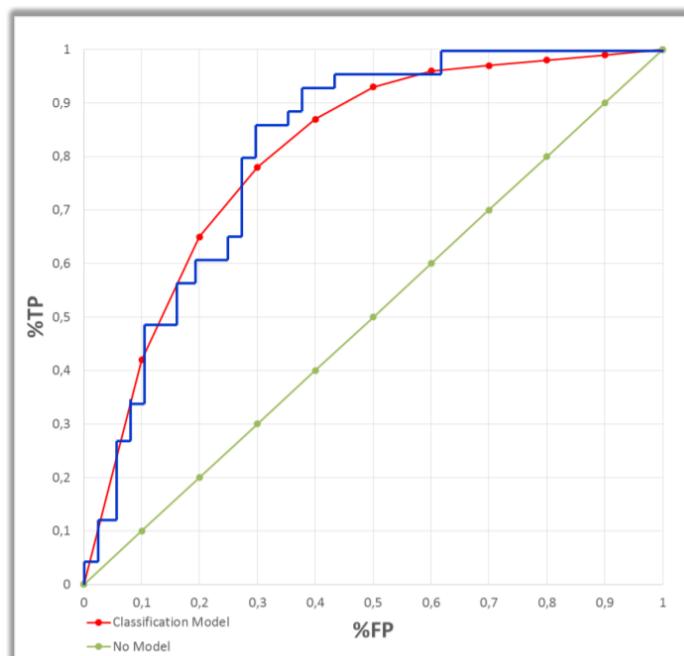


Figura 37: Curva ROC

Nell'esempio in figura vengono utilizzati particolari sottoinsiemi di dati, questa dipendenza può essere ridotta applicando la Cross-validation. La curva ROC rappresenta la performance di un classificatore senza guardare la distribuzione della classe o il costo dell'errore; serve, quindi, a confrontare diversi classificatori per cercare di comprendere dove un classificatore è più o meno efficace; un esempio è il seguente:

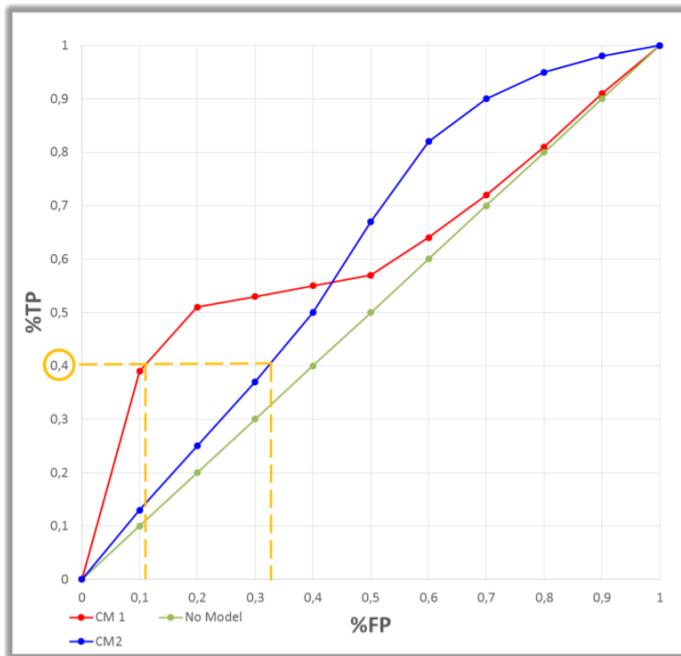


Figura 38: Confronto ROC tra due modelli

Un modello è **preferibile** ad un altro se è disposto a sopportare un certo numero di falsi positivi.

## 2.7 Feature Selection (\*)

Quasi sempre non ha senso utilizzare tutti gli attributi per generare il modello che si vuole sviluppare; Si procede, quindi, con la selezione delle feature. La **feature selection** è un compito importantissimo ed è fortemente legato alla risoluzione del problema. Analizzare le relazioni tra le variabili in modo *preliminare* potrebbe aiutare di molto nella ricerca della soluzione migliore al problema.

**Definizione 2.30.** La **feature selection** è il processo che tenta di scoprire quali attributi sono:

- *Ridondanti* ovvero quelli dei quali l'informazione è già contenuta in altri attributi, la rilevanza va sempre valutata in base ai dati a disposizione
- *Irrilevanti* contengono informazioni non utili per risolvere il problema di data mining.

Vi sono diversi **approcci** per rilevare questi attributi:

- *Brute-force*: si provano tutti i modelli per ogni combinazione di parametri usati nel modello di Classificazione. La computazione di questi casi è troppo grossa, vi sono:

$$\sum_{n=1}^{10} \binom{10}{n}$$

possibili combinazioni

- *Embedded*: gli attributi sono scelti in base alla capacità del modello della classificazione
- *Filter*: gli attributi sono selezionati prima del classificatore in base a quelli che si ritengono più rilevanti e meno rilevanti (analisi di funzione obiettivo)
- *Wrapper*: gli attributi si scelgono in base al modello di classificazione scelto, quelli che sono rilevanti per un modello non lo sono per un altro (dipende dall'ipotesi che ho fatto)

Politica Filter & Wrapper a confronto

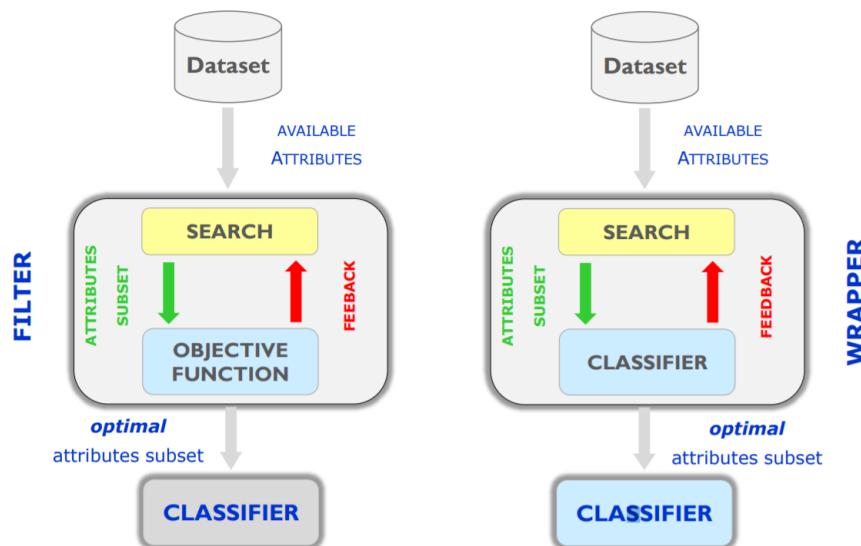


Figura 39: Filter - Wrapper, due differenti approcci per feature selection

*Non va bene fare feature selection su tutti i dati e poi trainare solo su un sottoinsieme!! altrimenti si hanno risultati non comparabili*

### 2.7.1 Filter

Nel caso di attributi **Uni-variati**:

1. Si sceglie una *misura di associazione* tra gli attributi candidato e quello di classe
2. Si *ordinano* gli attributi in base alla misura di associazione
3. Si *selezionano* le prime  $R$  migliori posizioni come attributi di input per il classificatore

Soltanamente seguendo questa procedura si identificano correttamente gli *attributi irrilevanti*; non è detto, però, che funzioni bene nella ricerca di *attributi ridondanti*. Nel caso di attributi **Multi-variati**:

- Si identificano congiuntamente *attributi rilevanti* e *irrilevanti*
- Un buon sottoinsieme di attributi deve contenere attributi fortemente associati con l'attributo di classe ma essere incorrelati tra di loro

Le tecniche più utilizzate sono:

Uni-Variate	Multi-Variate
<b>Parametric</b> t-test ANOVA Mutual Information	Correlation Feature Selection (CFS) Relief Blanket
<b>Non Parametric</b> Mann-Whitney Kruskal-Wallis Permutation test	

Figura 40: Tecniche uni-multi variate per filtrare le features

	Advantages	Disadvantages
<b>Uni-Variate</b> speed scalability independent on the classifier		ignore that attributes can be dependent ignore interactions with the classifier
<b>Multi-Variate</b> model dependency between attributes independent on the classifier computational cost compares favorably to wrapper		slower than Uni-Variate techniques less scalable than Uni-Variate Techniques ignore interactions with the classifier

Figura 41: Vantaggi e svantaggi tra tecniche uni-variate e multi-variate

I **vantaggi** per la feature selection sono:

- Riduzione del costo di collezione di dati
- Riduzione dei tempi di inferenza del classificatore relativo all'attributo di classe
- Classificatore più interpretabile
- Aumento dell'accuratezza

Le **motivazioni** per cui questo approccio viene utilizzato sono:

- Evitare l'overfitting
- Sviluppo di un miglior cost-effective classificatore
- Migliorare la comprensione del processo di generazione dati

Flowchart da seguire:

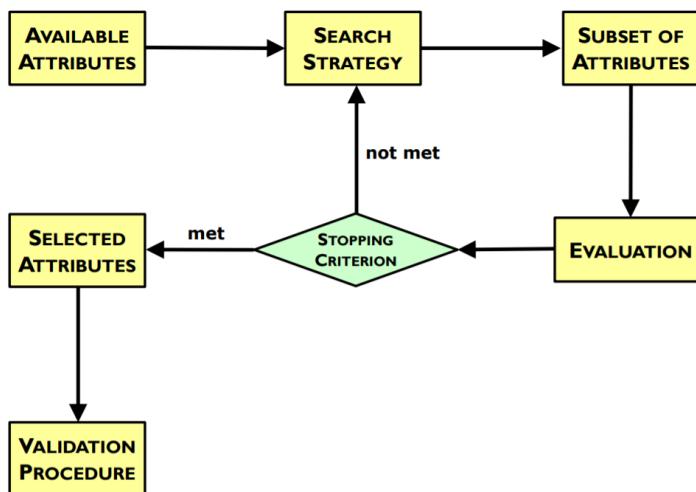


Figura 42: flowchart per la feature selection

È spesso possibile creare, dagli attributi originali, un nuovo insieme di attributi che cattura informazioni rilevanti nel dataset in modo più efficace; inoltre, il numero di nuovi attributi può essere più piccolo rispetto al numero di attributi originali.

Le metodologie per la creazione di nuovi attributi (features) sono:

- **Feature extraction:** una ampia gamma di modelli possono essere applicati in domini specifici come per classificazione di immagini/segnali
- **Mappare dati in un uovo spazio:** vedere gli attributi in modo normalizzato o secondo un'altra scala più facilmente trattabile (es. SVM) può migliorare di molto le performance
- **Feature construction:** generare feature nuove più comode per un certo modello di classificatore rispetto a quelle originali attraverso trasformazioni, es. logaritmi di somme ecc.. (non è detto che i dati originali siano i più utili per la classificazione)

### 2.7.2 Regularization

Alcuni modelli di Machine Learning ammettono iper-parametri. È dimostrato che una buona scelta dell'architettura di una rete neurale permette di approssimare qualsiasi problema. Non è però esente da overfitting, anzi, visto che non è sempre chiaro il funzionamento, è più difficile valutare quando sia presente. Si cerca di scegliere quella allocazione di parametri che riduce il più possibile l'errore quadratico del modello, quindi l'iperparametro  $\lambda$  detto di regolarizzazione:

$$E(\mathbf{w}, \lambda) = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2 + \frac{\lambda}{2} \sum_{j=1}^K w_j^2$$

$K$ : numero di parametri liberi (weights + thresholds) della rete neurale

$\lambda$ : parametro di regolamentazione

La prima parte della formula riguarda il fitting della funzione, la seconda parte è la flessibilità alla quale posso accedere.

*Non si deve usare il training set per ottimizzare il parametro di regolarizzazione  $\lambda$ . Si andrebbe ad overfittare il modello. Se viene utilizzato il test set per*

ottimizzare il parametro  $\lambda$  allora si è bruciato il dataset e devo rifare il modello, l'obiettivo è sempre quello di riconoscere dati mai visti, per questo si adottano schemi di divisione del dataset diversi.

### 2.7.3 Schema division

Lo schema di divisione del dataset ottimale potrebbe essere questo:



Figura 43: Diversi schemi di divisione del dataset

- Il train dataset viene utilizzato per trainare il modello (sui parametri  $w$ )
- Il parametro  $\lambda$  viene ottimizzato usando il validation set
- Il test set viene usato per fornire una stima delle performance senza bias
- Può essere fatto tramite holdout, iterated holdout e cross-validation

Quando viene usato l'approccio **Filter** per la feature selection solitamente si usa uno schema **Train/Test**

- Rilevanza e/o ridondanza sono stimate usando il Train set
- Dopo aver selezionato gli attributi il Train set viene usato per allenare il classificatore
- Le stime delle performance sono ottenute applicando il classificatore al Test set

Quando viene applicato l'approccio **Wrapper** per la feature selection solitamente si usa uno schema **Train/Validation/Test**

- Il Validation set viene usato per ottimizzare le performance quando vengono usati diversi attributi per il modello (i wrapper)
- Selezionare sottoinsiemi ottimali di attributi è lo stesso che selezionare il valore ottimo del parametro regolarizzato  $\lambda$
- Una volta selezionati gli attributi, il Train set e il Validation set vengono uniti e usati per addestrare il classificatore (tipicamente lo stesso tipo di classificatore usato per la feature selection)
- Le stime delle performance sono ottenute applicando il classificatore al Test set

## 2.8 Classificazione NON binaria

Nei problemi reali la classificazione avviene su più valori non solo binari.

**Definizione 2.31.** Un problema di classificazione **Non binaria** è un problema di classificazione che si divide in:

- **Multi-classe:** esattamente una classe si realizza
- **Multi-etichetta:** più di una classe può verificarsi

Vi possono essere anche problemi non di classe ma di **ranking** in cui i valori assunti dalla variabile di classe sono ordinati. Per la risoluzione di questi problemi solitamente si adotta una logica **One-Vs-All**.

### 2.8.1 One-Vs-All

L'idea è quella di trasformare un problema multi-classe in tanti attributi di classe binari. In questa modalità si verifica se il set corrente verifica o meno ciascuna delle caratteristiche da valutare.

*Bisogna ricordare che non sempre binarizzare è necessario, in particolare per quanto riguarda i naive bayes non serve, invece per un decision tree sì.*

Nel **One-vs-all** si crea un numero di *classificatori binari* diverso in base al numero di modalità dell'attributo di classe; bisogna, inoltre, normalizzare poi gli output dei classificatori. Nel modello *Multi-class* bisogna raccogliere i valori risultanti dai classificatori e fissare un threshold sopra il quale considero significativo il risultato (potenzialmente più di una classe supera il threshold).

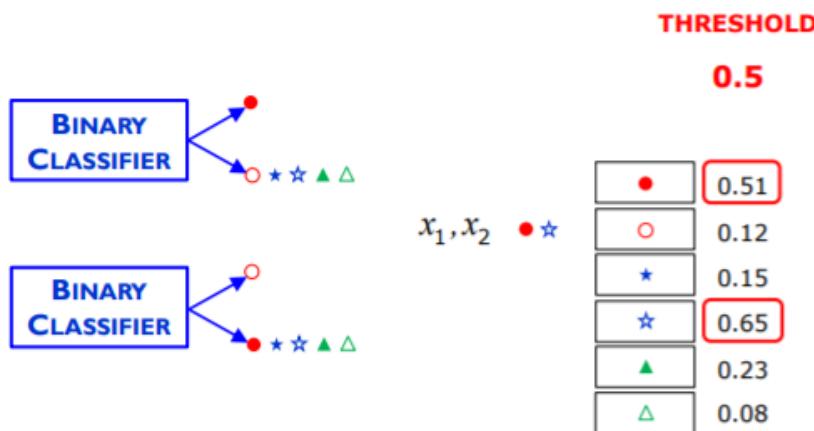


Figura 44: Esempio di classificatori binari

## 3 Clustering

### 3.1 Introduzione(\*)

L'analisi di cluster è usata per risolvere moltissimi problemi pratici; in particolare, l'analisi di cluster tratta due diversi scopi generali:

- **Comprendere:** Le classi, o gruppi di oggetti che condividono caratteristiche, giocano un ruolo importante nella comprensione del mondo. Questo succede in biologia, in informatica e in economia.
- **Utilità:** Capacità di riassumere determinate caratteristiche di un oggetto con le caratteristiche del cluster a cui appartiene. L'obiettivo è *trovare i prototipi con le proprietà più rappresentative dei cluster*.

Può essere fornita una definizione un po' più completa di analisi di clustering come segue.

**Definizione 3.1.** La **cluster analysis** è la tecnica che raggruppa i dati basandosi sulle informazioni trovate nei dati che descrivono gli oggetti e le loro relazioni.

Gli obiettivi sono semplici da ridefinire:

- Gli oggetti all'interno di un gruppo devono essere simili gli uni con gli altri, allo stesso tempo diversi (o incorrelati) con gli oggetti di altri gruppi.
- La più grande *similarità* entro un gruppo deve corrispondere ad una grande *differenza* tra i gruppi.

**Problema:** Come è possibile stabilire quando e quanto degli oggetti sono simili? Non vi è un metodo per capirlo, tendenzialmente si utilizza una soluzione intermedia rispetto alle altre.

La definizione di cluster è **intrinsecamente imprecisa**, una migliore definizione dipende infatti dalla natura dei dati e dai risultati desiderati.

#### 3.1.1 Tipi di clustering

Formare un insieme di cluster è chiamato in gergo tecnico *clustering*. Ci sono diversi tipi di analisi di clustering.

- Partitioning vs Hierarchical
- Exclusive vs Overlapping vs Fuzzy
- Complete vs Partial.

Di seguito una rapida definizione di tutte:

**Definizione 3.2.** Un clustering si dice **partizionale** se vi è una divisione del dataset in insiemi non sovrapposti tali che un elemento appartiene ad un solo insieme.

**Definizione 3.3.** Un clustering si dice **gerarchico** se ogni cluster può essere a sua volta suddiviso in sotto cluster, in questo caso il clustering è un insieme di cluster che sono organizzati come un albero.

**Definizione 3.4.** Un clustering si dice **esclusivo** se ogni oggetto è assegnato ad un singolo cluster.

**Definizione 3.5.** Un clustering si dice **sovrapponibile** se ogni oggetto può essere assegnato a più di un cluster.

**Definizione 3.6.** Un clustering si dice **fuzzy** se ogni oggetto può essere assegnato a più di un cluster contemporaneamente con un valore che tiene conto del peso che ha l'oggetto rispetto all'appartenenza ad un singolo cluster, la somma dei pesi deve essere necessariamente 1.

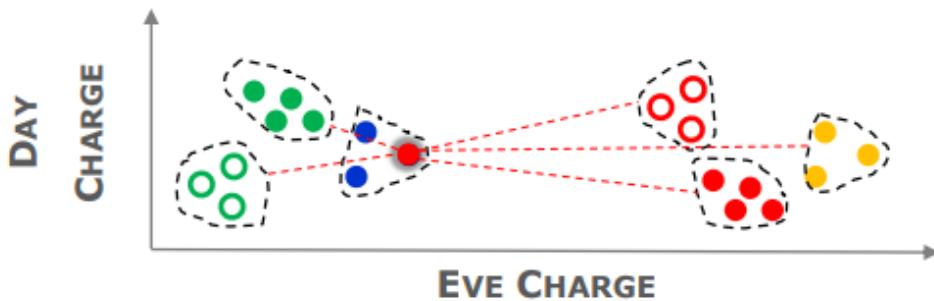


Figura 45: Clustering con modello fuzzy

**Definizione 3.7.** Un clustering si dice **completo** se ogni oggetto è assegnato ad un cluster (non ci sono oggetti liberi).

**Definizione 3.8.** Un clustering si dice **parziale** se esiste almeno un oggetto che non è assegnato a nessun cluster, si usa perché potrebbero esserci degli outlier ed inserirli all'interno di un cluster potrebbe peggiorare in modo significativo la rappresentazione di un cluster.

### 3.1.2 Differenti nozioni di cluster

I cluster naturali sono cluster che si dicono esistano per davvero anche se è molto difficile che ciò accada. Per visualizzare le differenze tra i tipi diversi di dati si sfruttano dati come punti a due dimensioni. Le categorie in cui sono divisi gli algoritmi di clustering sono le seguenti:

- **Well separated Cluster:** dato un cluster, ogni oggetto è più vicino ad ogni oggetto del cluster a cui appartiene piuttosto che a qualsiasi altro oggetto di ogni altro cluster. Un cluster così ben formato permette di avere separazioni molto nette, questo tipo di cosa succede però molto raramente in realtà.
- **Prototipe-based cluster** dato un cluster, ogni oggetto di quel cluster è più vicino al prototipo che definisce il cluster rispetto ad ogni prototipo di un altro cluster. Il prototipo è solitamente il *centroide* del cluster. Il *prototipo* di un cluster corrisponde all'individuo meglio rappresentato dal cluster (può essere anche fittizio). Cluster fatti in questo modo tendono ad essere globulari.
- **Density-based cluster** un cluster è una regione densa di oggetti che sono circostritti da una regione di bassa densità. Questi sono usati quando i cluster sono irregolari o intermittenti oppure quando c'è una grande presenza di rumore o di outlier.

- **Graph-based cluster** se i dati sono rappresentati da grafi, i nodi rappresentano oggetti e i collegamenti connettono gli oggetti. Allora ogni cluster è una componente连通的. La connessione può anche essere pesata e scelta in base ad una certa soglia. Questi cluster sono molto utilizzati in quanto c'è un sacco di ricerca già fatta.

### 3.1.3 Componenti di un'analisi di clustering

La prima fase di un'analisi di clustering consiste nella *feature selection* che assicura la trattenuta degli attributi del dataset degni di significato. Successivamente avviene la fase di *feature extraction* che serve a produrre feature che potrebbero andare meglio per scoprire la struttura dei dati, questa pratica potrebbe tuttavia generare features di difficile comprensione. Bisognerebbe usare come feature ideali quelle che permettono di distinguere i pattern degli elementi che appartengono ai diversi cluster, immuni al rumore e facili da interpretare.

Il secondo passo è quello di *determinare la misura di prossimità e costruire la funzione di merito*. Una volta che abbiamo determinato una misura di prossimità il problema di clustering si traduce in un problema di ottimizzazione di una specifica funzione.

Bisogna ricordare sempre che diversi algoritmi di clustering portano ad avere conclusioni anche totalmente diverse, questo è il motivo per cui in principio non bisogna prediligere alcun algoritmo ma confrontare i risultati ottenuti e trarne conclusioni.

## 3.2 Proximity(\*)

La proximity è uno strumento fondamentale per valutare il funzionamento di un algoritmo di clustering. Questa influenza, quindi, in modo pesante la soluzione di un problema di clustering. Bisogna fare una scelta della misura con cui si approssima la similarità degli elementi appartenenti allo stesso cluster.

### 3.2.1 Introduzione

*La analisi dei cluster affonda le sue radici nel concetto di cosa sia simile e cosa sia dissimile*, l'analisi di questo concetto in termini formali diventa abbastanza difficile. Questo avviene perché la similitudine dipende fortemente dal contesto analizzato.

*In generale la similarità è nulla se i due oggetti sono totalmente differenti sotto la caratteristica che stiamo valutando ed è uguale a 1 se sono completamente uguali, è comune però trovare misure di similitudine che hanno come valori: [0, inf].* Useremo il termine **proximity** per indicare sia la similarità che la dissimilarità.

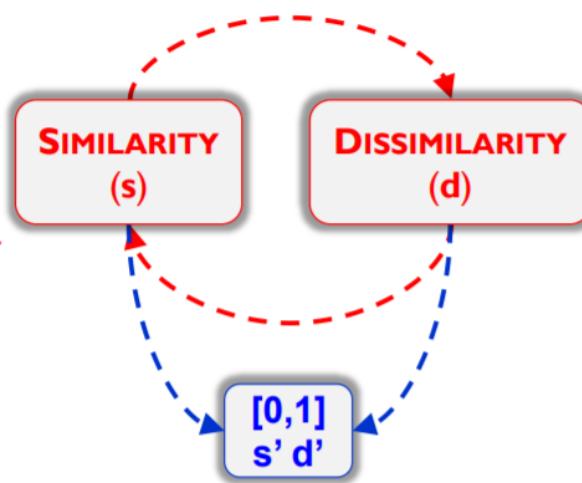


Figura 46: Relazione tra similarità e dissimilarità

Non c'è ortogonalità tra la scelta della misura e l'esito che otterrò; nasce, per questo motivo, l'esigenza di poter passare da una misura all'altra, in particolare, per trasformare una misura di similarità dall'intervallo  $[0, \infty]$  all'intervallo  $[0, 1]$  si opera la seguente trasformazione:

$$s' = \frac{s - \min s}{\max s - \min s} \quad d' = \frac{d - \min d}{\max d - \min d}$$

Ci sono diversi problemi che nascono quando viene cambiato l'intervallo in cui si trova il valore. Per farlo bisogna usare una trasformazione *non-lineare*. Possono essere usati, però, funzioni del genere:

$$d' = \frac{d}{1 + d}$$

Con questa trasformazione grandi valori della dissimilarità  $d$  vengono compressi in valori vicini a 1. Il fatto di distorcere o meno le distanze dipende dal compito che bisogna svolgere. Ci sono diversi problematiche che nascono quando si trasforma una similarità in dissimilarità e viceversa. Per farlo bisogna usare una trasformazione *non-lineare*. Si può usare, però, una cosa del genere:

$$\text{se } s, d = [0, 1] \text{ allora } s = 1 - d$$

*In generale si può usare qualsiasi funzione monotona decrescente per trasformare la similarità in dissimilarità.*

**Definizione 3.9.** Si definisce **prossimità** tra due record come la funzione di prossimità tra i corrispondenti attributi dei due record.

Si consideri in prima analisi la misura di prossimità tra due record aventi un solo attributo ed estendiamo successivamente questa analisi a record con più di un attributo.

ATTRIBUTE TYPE CATEGORICAL (QUALITATIVE)	DISSIMILARITY		SIMILARITY
	NOMINAL	ORDINAL	
CATEGORICAL (ORDINATIVE)	$d = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{otherwise} \end{cases}$	$d = \frac{ x - y }{n - 1}$	$s = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$
		$d =  x - y $	$s = 1 - d$
NUMERIC (QUANTITATIVE)			$s = -d \quad s = \exp(-d)$
			$s = \frac{1}{1 + d}$
RATIO			$s = 1 - \frac{d - \min_d}{\max_d - \min_d}$

Figura 47: Tabella per misurare la prossimità di record con un solo attributo in funzione del tipo di attributo

Si considerino due record riferiti al medesimo attributo nominale qualitativo, tutto ciò che si può dire è se i due record abbiano lo stesso valore o meno. Per quanto riguarda gli attributi binari la dissimilarità esclude la similarità con valori 0 e 1.

Se ho attributi categorici ordinali assegno un valore intero agli stessi in base alla scala utilizzata e calcolo la dissimilarità come rapporto tra la differenza dei due record e la scala totale di valori di utilizzo. Naturalmente va notato che sto utilizzando una scala lineare, questa è ovviamente un'assunzione molto forte che però bisogna tenere in conto in quanto qualsiasi scala di valori scelta è fatta basandosi su assunzioni.

Come si nota dalla tabella è molto più facile definire la prossimità tra due attributi numerici, essa è infatti definita come *la differenza in modulo tra i due valori*.

### 3.2.2 Misure della distanza

Quando si hanno degli attributi numerici possono essere definite altre misure di distanza nel seguente modo:

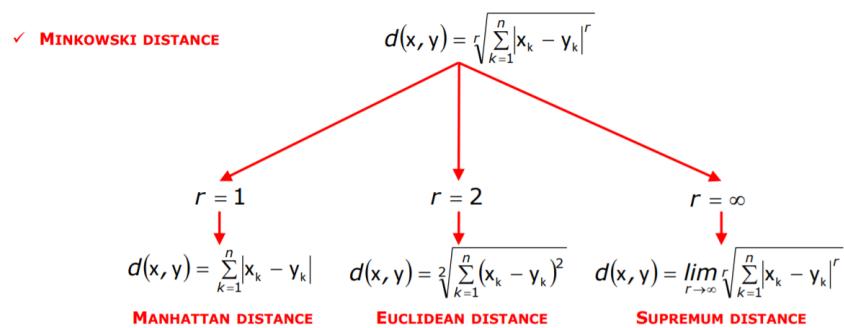


Figura 48: Misure di distanza a partire dalla distanza di Minkowski

Le proprietà che una funzione deve soddisfare per essere definita distanza sono le seguenti:

- Non negatività:  $d(x, y) \geq 0 \quad \forall x, y \quad d(x, y) = 0 \quad \text{if} \quad x = y$
- Simmetria  $d(x, y) = d(y, x) \quad \forall x, y$
- Disuguaglianza triangolare  $d(x, z) \leq d(x, y) + d(y, z) \quad \forall x, y, z$

La similarità generalmente non rispetta la disuguaglianza triangolare ma solitamente verifica le proprietà di simmetria e non negatività.

Di seguito qualche esempio di misura di prossimità:

**Definizione 3.10.** Si definisce **Simple matching coefficient** la seguente espressione:

$$SMC(x, y) = \frac{\#maching\_attributes}{\#attributes} = \frac{f_{11} + f_{00}}{f_{11} + f_{00} + f_{01} + f_{10}}$$

Questa è una misura che ha senso per attributi *simmetrici binari*, ossia per attributi che possono assumere solo i valori 0 o 1 in circa egual quantità. Questa misura è invece scomoda se non si può affermare se gli 0 siano veramente degli 0, per il valore 1 invece è chiaro. In questo caso si utilizza un'altra misura che è derivata da questa misura, ossia si usa l'assunzione per cui l'osservazione di un evento (identificata con 1), abbia peso maggiore della non osservazione di un evento (rappresentata con 0). Si definisce quindi il coefficiente di Jaccard nel seguente modo:

**Definizione 3.11.** Si definisce **Jaccard Coefficient** la seguente espressione:

$$J(x, y) = \frac{\#maching\_attributes}{\#attributes\_except00} = \frac{f_{11}}{f_{11} + f_{01} + f_{10}}$$

Questa è ovviamente una misura distorta rispetto agli 1 (infatti è un tipo di misura che va bene con attributi *asimmetrici*) che permette, però, di focalizzare la attenzione sulla presenza di questi ultimi. Di seguito la definizione di un nuovo indice:

**Definizione 3.12.** Si definisce **Extended Jaccard Coefficient** la seguente espressione:

$$EJ(x, y) = \frac{x \cdot y}{\|x\|^2 + \|y\|^2 - x \cdot y}$$

Questa misura è distorta per trattare dati sparsi, quindi tanti elementi in cui sono presenti 0 e solo poche diverse da 0, questo si usa ad esempio nell'analisi del linguaggio naturale. Si pensi ad esempio ai tweet, una parola presente in una frase ha una rilevanza maggiore rispetto ad una parola non presente nella stessa frase.

### 3.2.3 Altre misure di prossimità

Di seguito vengono proposte ulteriori misure di prossimità.

**Definizione 3.13.** Si definisce **cosine similarity**. la seguente espressione:

$$\cos(x, y) = \frac{x \cdot y}{\|x\| \cdot \|y\|}$$

Questa misura viene usata quando tutti gli attributi sono di natura numerica, e si ignorano i match di natura 00. Il vantaggio di questa misura rispetto a quella di Jaccard è che in grado di trattare anche attributi non binari. E' molto utile quindi per comparare record sparsi ed è molto usata in *Information Retrieval* dove i documenti (rappresentati come conteggi di vettori) devono essere comparati.

**Definizione 3.14.** Si definisce **Correlazione** la seguente espressione:

$$\text{corr}(x, y) = \frac{\text{cov}(x, y)}{\text{std}(x)\text{std}(y)}$$

Questa è la stessa correlazione di Pearson ma non legata alle variabili aleatorie. Di seguito diverse **problematiche** legate alle misure di prossimità:

- Come si trattano attributi su scale di ampiezza diverse e/o correlati?

Per risolvere il primo problema viene fatta la seguente cosa: si normalizzano i valori, se ciò non venisse fatto le distanze Euclidee tra i due valori risulterebbero totalmente distorte a favore del valore maggiore.

Quando gli attributi sono fortemente correlati invece il trucco sta nel fatto che la misura di similarità è molto simile al grado di correlazione tra questi attributi, in tal caso si utilizza la distanza di Mahalanobis:

$$\text{Mahal}(x, y) = (x - y)\Sigma^{-1}(x - y)^T$$

Ovviamente questa è una distanza che va usata solo se tutti gli attributi sono numerici.

- Come si calcola la prossimità tra record composti da attributi di tipo diverso?

Per risolvere questo problema si valutano tutte le misure di prossimità enumerate in precedenza stando coerenti col tipo di attributo trattato. Dopo averlo fatto si usa una variabile indicatrice  $\delta_k$  per ogni attributo k come segue:

$\delta_k$  vale:

- 0 se il k-esimo attributo è asimmetrico ed entrambi i valori hanno valore 0, o almeno uno dei record presenta un missing value
- 1 altrimenti.

Una volta definite queste allora la similarità si calcola come:

$$\text{similarity}(x, y) = \frac{\sum_{k=1}^n \delta_k s_k(x, y)}{\sum_{k=1}^n \delta_k}$$

- Come si tratta la prossimità quando gli attributi hanno diversa rilevanza, ossia quando gli attributi contribuiscono secondo pesi diversi all'analisi?

Per risolvere quest'ultimo problema si procede esattamente nel modo precedente assegnando però dei pesi, le formule risolutive diventano quindi:

$$\text{similarity}(x, y) = \frac{\sum_{k=1}^n w_k \delta_k s_k(x, y)}{\sum_{k=1}^n \delta_k}$$

Come è facilmente intuibile risulta molto complesso sostenere tutta questa specificità. Per farlo si cerca di ricondursi alla seguente scelta:

- Dati densi e continui: distanze metriche come quella euclidea sono buone rappresentazioni.
- Dati sparsi, binari asimmetrici: misure della distanza che ignorano i match 00 come cosine, Jaccard e Extended Jaccard.

### 3.3 Clustering Algorithms

Di seguito la trattazione degli algoritmi di Clustering veri e propri:

#### 3.3.1 Prototype Based

L'approccio ai *Prototype-Based Clustering* si basa sull'assunzione che ogni cluster possa essere ben rappresentato da un unico punto chiamato **prototipo**. Ogni oggetto è quindi collocato nel cluster del prototipo a cui è più vicino.

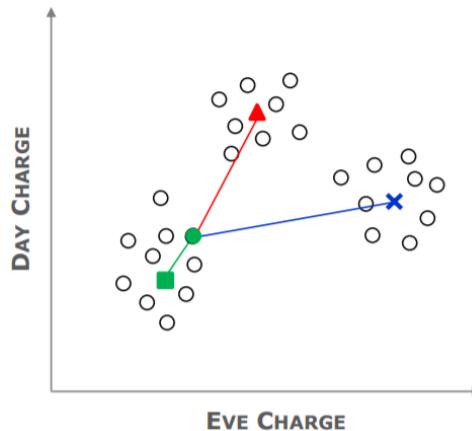


Figura 49: Esempio di prototype-based clustering

Esistono diversi tipi di algoritmi basati sul prototipo a seconda delle seguenti caratteristiche:

- Ogni oggetto deve appartenere ad un singolo cluster.
- Ogni record è nella condizione di appartenere a più di un cluster contemporaneamente.
- Il concetto di cluster è modellizzato con una distribuzione di tipo probabilistico.
- I cluster sono costretti ad avere relazioni fissate.

**K-medie** Di seguito viene proposta la descrizione di uno dei primi algoritmi basati sul prototipo. In questo caso il prototipo prende il nome di *centroide*; questo valore solitamente è identificato dal vettore media dei valori degli attributi delle osservazioni di quel determinato cluster. Non c'è vincolo a ragionare in due dimensioni, il cluster generalmente è applicato ad oggetti in uno spazio continuo n-dimensionale.

Di seguito una descrizione schematica dell'algoritmo:

- Scegliere per  $k = 0$  quali sono i centroidi facendo in modo che non si "pestino i piedi".
- **Ripetere:**
  - Formare  $k$  cluster in modo da assegnare ad ogni record il suo centroide più vicino. E' ovviamente un meccanismo esclusivo.
  - Calcolare il nuovo centroide per ogni cluster .

- Finché il centroide non cambia più.

Come si nota facilmente, bisogna esplicitare il numero di cluster prima dell'esecuzione dell'algoritmo; valori di partenza differenti possono fornire risultati molto diversi. La scelta del numero di cluster è dunque un ambito da tenere strettamente in considerazione.

L'algoritmo delle K-medie non è vincolato ad utilizzare la distanza Euclidea ma possono essere utilizzate le diverse misure di prossimità utilizzate in precedenza. In particolare:

- **Manhattan:** in questo caso si utilizzano le mediane come centroidi, l'obiettivo è quindi *minimizzare la somma delle  $L_i$  distanze dei record rispetto al centroide del cluster a cui appartiene.*
- **Squared Euclidea:** in questo caso si utilizzano le medie come centroidi, l'obiettivo è quindi *minimizzare la somma delle  $L_i$  distanze dei record rispetto al centroide del cluster a cui appartiene.*
- **Cosine:** in questo caso vengono utilizzate le medie come centroidi, l'obiettivo è quindi *massimizzare la somma delle cosine similarity dei record rispetto al centroide del cluster a cui appartiene.*

Di seguito tutta una serie di **problematiche** relative all'utilizzo dell'algoritmo delle K-medie come algoritmo di clustering.

- *Scelta dei centroidi iniziali:* è una fase fondamentale e influenza in modo pesante le performance dell'algoritmo in generale. Se si opta infatti per una scelta random dei centroidi iniziali si possono avere cluster molto diversi. Vi sono diverse alternative per ovviare a questo problema quali il clustering gerarchico.
- *Complessità spaziale e temporale:* questi sono due punti a favore del K-medie. In particolare occupa pochissimo spazio in quanto vengono salvate solo le posizioni dei centroidi; inoltre, si tratta di un algoritmo abbastanza rapido in quanto è lineare rispetto al numero di istanze considerate.
- *Cluster vuoti:* bisogna tenere in considerazione che può capitare che un cluster sia vuoto per scelta sbagliata di centroidi iniziali (magari randomica).
- *Presenza di outlier:* Questi elementi creano grossi problemi nel calcolo della media delle osservazioni di un cluster. Risulta efficiente però nella ricerca di outlier in quanto verranno identificati come cluster di singleton. Risulta spesso utile rimuoverli.

Di seguito una serie di **limiti** riferiti alla ricerca dei cluster con l'algoritmo delle k-medie:

- Difficoltà a ricercare cluster di non forma sferica.

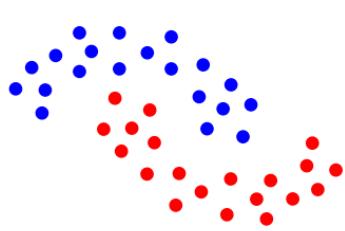


Figura 50: Cluster non sferico.

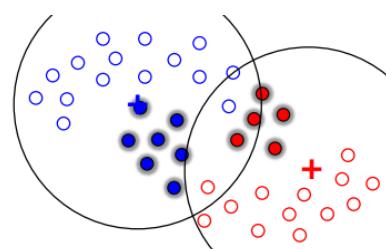


Figura 51: Cluster sferico.

- Difficoltà a trovare cluster con dimensioni diverse: questo problema nasce dal fatto che la distanza è fissata.

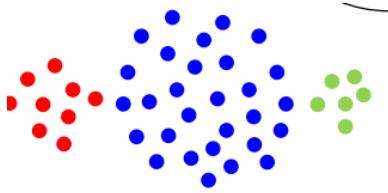


Figura 52: Distanza non fissata.

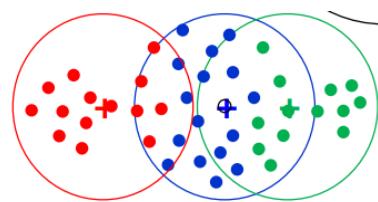


Figura 53: Distanza fissata.

- Difficoltà a indentificare cluster di diversa densità: questo prblema nasce dal fatto che i cluster possono avere dimensioni nettamente diverse.

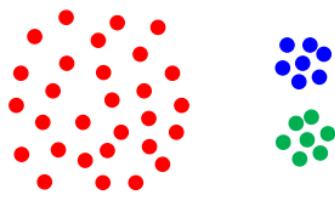


Figura 54: Densità differenti.

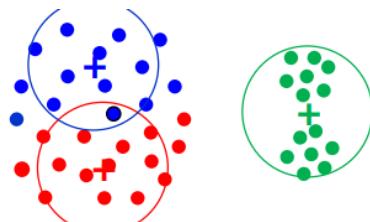


Figura 55: Densità simili.

Come si può notare queste tre problematiche portano a considerare cluster notevolmente differenti. Esiste un modo di risolvere parzialmente queste problematiche: *accettare un clustering che rompe i cluster naturali in un numero variabile di sottocluster*.

Di seguito un rapido elenco che riassume **punti di forza e debolezze** dell'algoritmo delle k-medie:

- Semplice e si applica a tanti tipi di dati.
- Abbastanza efficiente anche se vengono ripetute molte cose.
- Vi sono varianti più efficienti e meno problematiche come ad esempio l'algoritmo *bisecting k-means*.
- Non adatto a tutti i tipi di dati e non può gestire cluster non sferici, con size e densità diverse.
- Problemi con outliers.
- Strettamente legato al concetto di centroide. Vi sono delle varianti che utilizzano il *medoide* e che sono più efficienti.

**Fuzzy C-means** Se gli elementi sono distribuiti in gruppi ben separati il clustering è semplice e vengono messi in cluster disgiunti. In molti casi succede però che i record non possano essere partizionati in cluster ben separati, nasce quindi un'incertezza arbitraria nell'assegnare gli oggetti ad un particolare cluster. Questa problematica si riflette infatti sulla scelta dell'assegnazione degli elementi di frontiera al rispettivo cluster.

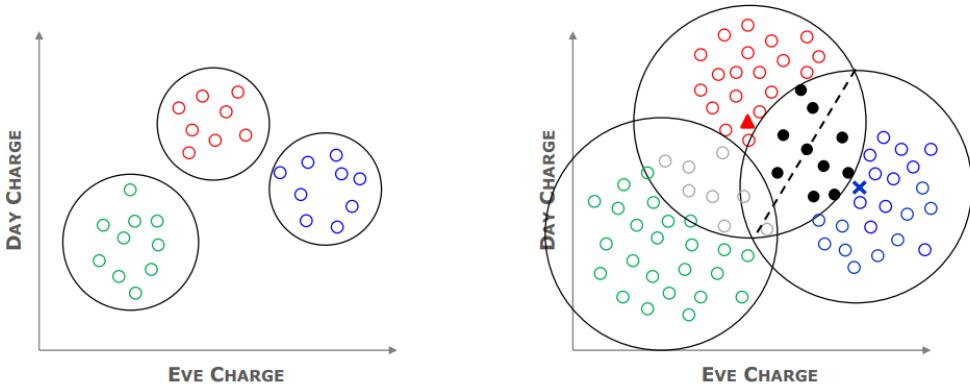


Figura 56: Differenza tra cluster ben separati e non ben separati.

Per ovviare al problema ogni osservazione viene considerata come appartenente ad ogni cluster ma con un peso diverso che indica il grado con cui un oggetto appartiene ad un ogni cluster. In particolare  $w_{ij}$  è il *peso* con cui l' $i$ -esimo oggetto appartiene al  $j$ -esimo cluster.

Si assuma di avere un dataset di  $m$  oggetti/record, dove ogni oggetto è associato ad un numero dato di attributi continui.

**Definizione 3.15.** Si definisce **Fuzzy pseudo-partition** l'assegnazione dei valori dei pesi  $w_{ij}$  con i seguenti vincoli:

$$\sum_{j=1}^K w_{ij} = 1 \quad i = 1, 2, \dots, m$$

$$0 < \sum_{i=1}^m w_{ij} < m \quad j = 1, 2, \dots, K$$

Ovviamente per come sono definiti i cluster *non vengono ammessi cluster a dimensione nulla*. Il Fuzzy C-means produce dunque un clustering che risalta l'indicazione del grado per cui un oggetto appartiene ad ogni cluster. Per come è definito ha gli stessi punti di forza e debolezze dell'algoritmo delle k-medie sebbene sia computazionalmente più pesante.

**Modelli a mistura** I modelli a mistura considerano i dati come insiemi di osservazioni da una mistura di diverse distribuzioni di probabilità. In sostanza si possono considerare le misture come distribuite secondo delle normali con uguale varianza ma medie diverse (curve di livello).

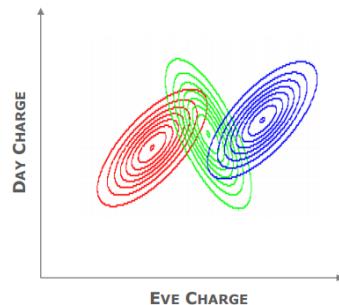


Figura 57: Curve di livello nei modelli a mistura.

Si supponga di avere uno spazio a 3 componenti, il processo di generazione delle curve di livello è il seguente:

1. Selezione di una delle 3 componenti.
2. Estrazione di un campione dalla componente selezionata.
3. Ripetere 1 e 2 m volte per ottenere il dataset.

Vengono inoltre le seguenti grandezze:

- $p(\bar{x}|\Theta) = \sum_{j=1}^K w_j p(\bar{x}|\theta_j)$ : probabilità associata all'oggetto x.
- $w_j$ : probabilità della j-esima componente.
- $p(\bar{x}|\theta_j)$ : probabilità di estrarre x dalla j-esima componente.
- $\theta_j$ : parametri associati a j.

Il processo consiste nel partire dai dati e ridurli nelle misture significative. Esiste un classe di algoritmi utilizzata per la risoluzione di questo problema che è l'**Expectation Maximization**. Si tratta di algoritmi che sono diversi in base alla loro applicazione.

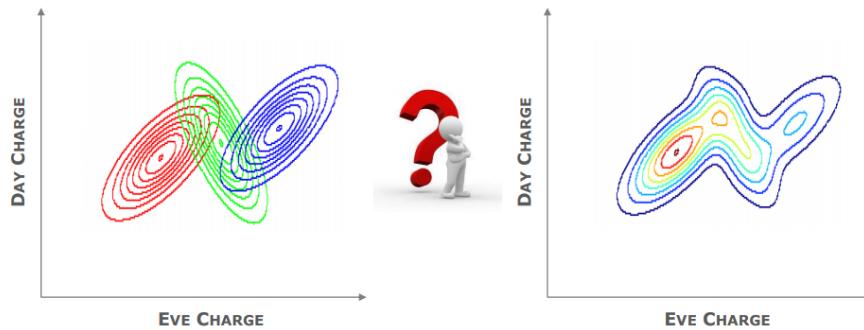


Figura 58: Risultato auspicabile.

Questa classe di algoritmi presenta però diversi **svantaggi**:

- L'apprendimento risulta essere molto lento.
- Non è pratico per i modelli con un gran numero di componenti.
- Non funziona bene quando i cluster contengono poche osservazioni.
- Non funziona bene quando gli oggetti sono co-lineari.

Questi algoritmi presentano però anche una serie di **vantaggi** non indifferenti rispetto a quelli delle k-medie:

- Sono più generali rispetto alle k-medie e alle fuzzy perché usano distribuzioni di vario tipo, possono trovare cluster di diversa grandezza e di forma ellittica.
- Disciplina il metodo di eliminazione delle complessità associate ad alcuni tipi di dati.

**Mappe di Kohonen o SOMs** E' una struttura feedforward per algoritmi di clustering; in particolare, viene imposta un'organizzazione topografica dei *neuroni* (centroidi). Durante il processo di training la SOM usa ogni oggetto per aggiornare il centroide più vicino e i centroidi che sono più vicini nell'ordinamento topografico. La differenza principale tra gli algoritmi precedenti è che i centroidi in questo caso *hanno una predeterminata relazione di ordinamento topografico*.

In particolare, si ha che i centroidi che sono più vicini l'uno all'altro nella griglia sono più strettamente correlati rispetto ai centroidi che sono lontani.

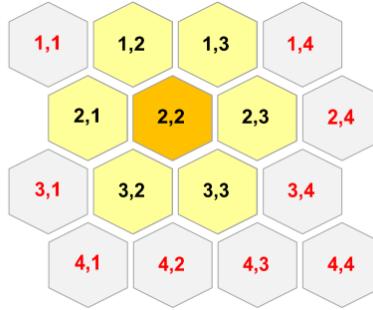


Figura 59: Giriglia di una SOM.

A causa di questo vincolo i centroidi di una griglia bidimensionale possono essere visti come giacenti su una *superficie bidimensionale che prova ad adattare i dati multidimensionali il meglio possibile*.

Si riassumono di seguito i passaggi con cui una SOM crea i cluster:

1. *Inizializzazione*: vengono selezionati i centroidi associati a ciascun neurone
2. *Competizione*: i neuroni competono tra loro per ottenere l'istanza. Ogni neurone fa un'offerta diversa per vincere quell'oggetto e la proposta è proporzionale alla distanza del neurone rispetto all'istanza considerata. Viene poi proclamato il vincitore e gli viene assegnata l'istanza (in base a una misura di performance).
3. *Collaborazione*: una volta che un neurone vince distribuisce il suo vantaggio ai vicini (processo di premi-punizioni) aggiustando i centroidi.
4. *Aggiornamento*: il centroide del vincitore e dei vicini vengono aggiornati usando l'istanza corrente.

Il punto di forza maggiore di questo algoritmo è l'*Interpretabilità*: i cluster che sono vicini sono più correlati tra loro rispetto ai cluster che non sono vicini. Questo aspetto facilita enormemente l'interpretazione e la visualizzazione dei cluster.

Questo algoritmo soffre però anche di alcune limitazioni abbastanza importanti:

- L'utente deve scegliere un gran numero di parametri quali: funzione di vicinato, tipo di griglia e numero di centroidi. Questa scelta influenza molto le performance.
- Non sempre un raggruppamento identifica un singolo cluster naturale (solitamente dopo viene applicato un k-medie sui centroidi trovati).

- Mancanza di una specifica funzione oggetto: questo rende molto difficile paragonare i risultati.
- *Non è garantita la convergenza.*

### 3.3.2 Clustering Gerarchico

Di seguito la descrizione della seconda classe di algoritmi di clustering. In particolare, un clustering gerarchico può essere:

- *Agglomerativo*: si considerano inizialmente tutti gli oggetti come cluster individuali, ad ogni step vengono unite le coppie di step più vicine. Un processo di questo tipo richiede la definizione di prossimità tra i cluster.
- *Divisivo*: parte da un cluster unico che include tutti gli oggetti e passo passo divide il cluster in cluster di singleton (oggetti individuali). C'è la necessità di decidere quale cluster splittare ad ogni passo e come splittare.

**Clustering gerarchico agglomerativo** Si concentra l'attenzione su questo tipo di clustering perché sono i più comuni. La soluzione che si ottiene è il *dendogramma* del clustering effettuato. Il dendrogramma serve a visualizzare sia le relazioni tra i cluster e i sotto-cluster sia l'ordine in cui sono stati divisi o agglomerati i vari cluster. Questo tipo di algoritmo è computazionalmente molto pesante.

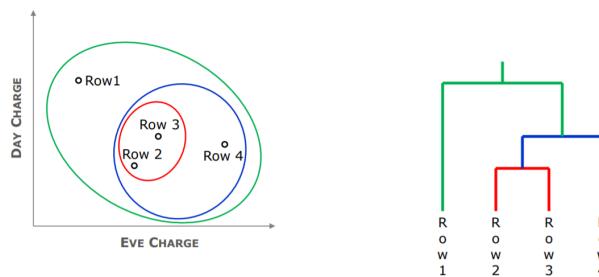


Figura 60: Clustering gerarchico sotto forma di dendrogramma

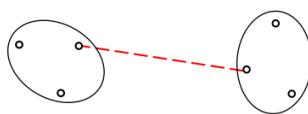
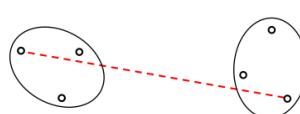
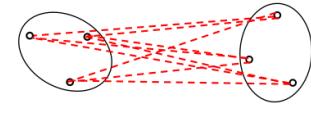
Di seguito un rapido riassunto dei passaggi dell'algoritmo:

1. Calcolare la matrice di prossimità (se necessario).
2. **Ripeti:**
  - Unione di due cluster vicini.
  - Aggiorna la matrice di prossimità (o delle distanze) che riflette la prossimità tra il nuovo cluster e il cluster originale .
3. **Finché** rimane un solo cluster.

Si pone ora il problema di calcolare la prossimità tra due cluster . Ci sono in particolare tre modi:

- *Min or Single Linkage*: minore distanza tra tutte le possibili coppie di elementi presenti nei due cluster, per come è definita è ovviamente la soluzione più ottimista.

- *Max or Complete Linkage*: maggiore distanza tra tutte le possibili coppie di elementi presenti nei due cluster, è una soluzione più robusta o pessimista.
- *Group Average or Avarage Linkage*: media distanza tra tutte le possibili coppie di elementi presenti nei due cluster, è una soluzione abbastanza usata in quanto media le distanze.
- *metodo di Ward*: assume che i cluster siano rappresentati dai centroidi. Misura la prossimità tra due cluster come l'incremento della somma di scarti quadratici che risulta dalla fusione di due cluster.

Figura 61: *Single*Figura 62: *Complete*Figura 63: *Average*

Le **caratteristiche** chiave di questi algoritmi sono le seguenti:

- *Mancanza di una funzione globale di oggetto*: per questo motivo non può risolvere problemi di ottimizzazione globale.
- *Abilità di gestire cluster di diverse dimensioni*: può gestire cluster con dimensioni differenti, i cluster possono anche essere pesati o meno.
- *Le decisioni di unione sono definitive*: non si può tornare indietro rispetto alla decisione (approccio greedy).

Le **problematiche** più importanti sono invece:

- Computazionalmente molto pesante e richiede molto spazio.
- Decisioni di unione di cluster sono sub-ottimali quindi creano rumore soprattutto per dati documentali.

### 3.3.3 Density-Based Clustering

Di seguito la descrizione della classe di algoritmi basati sulla densità. In particolare le tecniche di clustering basate sulla densità si occupano di *trovare regioni con un'alta densità di oggetti circondati da regioni con una bassa densità di oggetti*. Passiamo ora a parlare di un algoritmo specifico, il *DBSCAN*.

**DBSCAN** E' un semplice ma efficace algoritmo density-based che mostra un gran numero di concetti fondamentali tipici dell'approccio density-based. Ci sono diversi metodi per definire la densità, in particolare noi descriviamo il metodo center-based.

**Definizione 3.16.** Si definisce **densità** il numero di oggetti presenti all'interno di un raggio fissato ( $\epsilon$ ) rispetto ad un oggetto.

Come facilmente intuibile la densità può variare fortemente al variare del raggio.

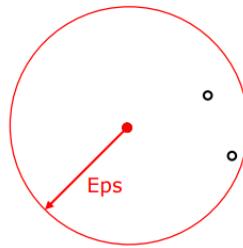


Figura 64: Densità rappresentata graficamente

Definita la densità in questo modo, i punti vengono classificati in tre tipologie diverse:

- *Core point*: stanno all'interno della regione ad elevata densità. In particolare, un punto viene definito così se il numero di punti nel suo vicinato eccede un certo valore di soglia (*MinPts*) pre-impostato dall'utente.
- *Border point* : un punto che include nel suo vicinato un core point, quindi lui si considera un vicino di core point.
- *Noise point*: non è né un core point né un border point.

Come negli altri casi di seguito i passaggi dell'algoritmo in maniera schematica:

1. Etichettare tutti i core, border, noise point.
2. Eliminare i noise point.
3. Collegare con un ponte tutti i core point che sono all'interno dei rispettivi *Eps*.
4. Trasformare ogni gruppo di core point connessi in cluster separati.
5. Assegnare ciascun border point al cluster associato al core point più prossimo.

Come detto in precedenza la scelta dei parametri *MinPts* e *Eps* è fondamentale in quanto impatta fortemente sull'esecuzione di tutto l'algoritmo. Di seguito, invece, quali sono le **differenze** principali tra l'algoritmo DBSCAN e l'algoritmo K-means:

- DBSCAN e K-Means assegnano gli oggetti ad un singolo cluster, l'algoritmo delle k-medie però assegna tutti gli oggetti mentre il DBSCAN può omettere il rumore.
- DBSCAN può trattare cluster di diverse dimensioni e forme e non è affatto da rumori o outliers. K-medie hanno difficoltà con i cluster non globulari oppure di forme diverse. Entrambi gli algoritmi performano male quando i cluster hanno significative differenze di densità.

- L'algoritmo delle K-medie può essere usato per dati che hanno centroidi ben definiti come la media o la mediana. L'algoritmo DBSCAN invece richiede una definizione di densità che è basata sulla nozione tradizionale di distanza Euclidea.
- L'algoritmo delle K-medie può essere applicato a dati sparsi e a multi-dimensional come ad esempio i dati documentali. Il DBSCAN in questi casi invece non performa per niente bene in quanto la definizione di densità Euclidea non funziona bene per dati multi-dimensionali.
- DBSCAN non fa assunzioni riguardo la distribuzione dei dati. L'algoritmo delle K-medie invece è equivalente ad approccio statistico di clustering che assume che tutti i cluster vengono da una distribuzione Gaussiana sferica con diverse medie ma con la stessa matrice di covarianza.
- Entrambi gli algoritmi cercano di costruire cluster utilizzando tutti gli attributi, ciò implica che non cercano cluster che riguardano solo un sottoinsieme degli attributi.
- L'algoritmo delle K-medie può trovare cluster che non sono ben separati, l'algoritmo di cluster invece mette insieme quelli sovrapposti.
- L'algoritmo delle K-medie ha complessità  $O(m)$  mentre il DBSCAN ha complessità  $O(m^2)$  eccetto per i dati a poche dimensioni.
- DBSCAN produce lo stesso insieme di cluster dalla prima riproduzione mentre l'algoritmo delle k-medie tipicamente non lo fa quando i centroidi sono inizializzati casualmente.
- DBSCAN determina automaticamente il numero di cluster mentre nell'algoritmo delle k-medie esso deve essere esplicitato in principio come parametro.

Vi sono altri algoritmi di density-based che non verranno trattati in questo momento quali: *grid-based clustering*, *subspace clustering*, *kernel density function*.

### 3.3.4 Graph-based Clustering Algorithm

Gli algoritmi di clustering gerarchico usano una visione dei dati basata sui *grafi*, in cui i dati sono rappresentati come *nodi* e la prossimità tra due istanze è rappresentata dal peso del *ponte* che c'è tra i rispettivi nodi. Sono stati esplorati algoritmi di clustering graph-based che esplorano diverse caratteristiche e proprietà dei grafi. Gli approcci chiave sono i seguenti:

- *Sparsificare il grafo di prossimità*: tagliare, rimuovere certi archi che non superano la verifica di una certa condizione. (Es. tutti gli archi che non superano una certa soglia, oppure un concetto di vicinato).
- *Definire una misura di similarità*: tra due oggetti basati sul numero di oggetti vicini che condividono. Questo approccio aiuta a diminuire i problemi che nascono con cluster di dimensioni diverse e con densità diverse.
- *Definire oggetti core e costruire cluster attorno ad essi*: Necessita l'introduzione di grafo di prossimità density-based.

- Utilizzare l'informazione nel grafo di prossimità per fornire una maggiore valutazione per cui due cluster dovrebbero essere uniti.

	Row 1	Row 2	Row 3	Row 4	Row 5	Row 6	Row 7	Row 8	Row 9	Row 10
Row 1	0.3	0.8	0.2	0.1	0.2	0.9	0.2	0.1	0.1	0.1
Row 2		0.2	0.4	0.1	0.9	0.4	0.4	0.6	0.2	
Row 3			0.1	0.2	0.1	0.8	0.1	0.1	0.1	0.1
Row 4				0.7	0.5	0.2	0.9	0.5	0.9	
Row 5					0.1	0.1	0.8	0.3	0.9	
Row 6						0.2	0.2	0.3	0.1	
Row 7							0.1	0.1	0.1	
Row 8								0.3	0.8	
Row 9									0.4	
Row 10										

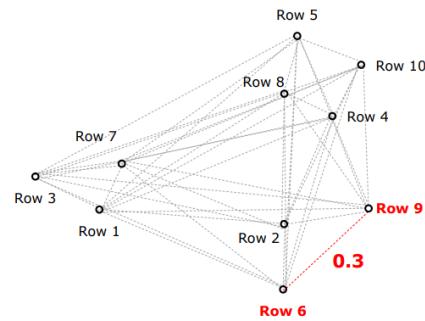


Figura 65: Correlazione tra matrice di prossimità e grafo di prossimità.

Come si può notare, ogni oggetto a un certo livello di similarità con *ogni* altro oggetto, per la maggior parte del dataset gli oggetti sono molto più simili ad un basso numero di oggetti e molto poco simili ad un alto numero di oggetti. Un'operazione utile è quella di sparsificare il grafo/matrice di prossimità ossia di settare alcuni dei valori di prossimità molto bassi a 0. Il "molto bassi" viene identificato da un valore soglia. Questo serve a *rompere tutti i nodi che hanno una prossimità sotto una data soglia o a tenere solo i collegamenti ai k-nearest neighbours per ogni punto*

**Definizione 3.17.** Definiamo quindi il **K-Nearest Neighbour** il numero di valori di archi che prendo come vicinato per ogni riga della matrice.

Come facilmente intuibile ci sono diversi vantaggi nello sparsificare la matrice. In primo luogo *la riduzione della dimensione dei dati*, settare dei valori a 0 infatti implica la non considerazione di quei collegamenti nell'algoritmo. In secondo luogo *l'algoritmo di clustering potrebbe lavorare meglio*, questo perché la sparsificazione potrebbe eliminare degli outlier o quegli elementi che costituiscono soltanto del rumore. Un ultimo motivo altrettanto importante è che *si possono trovare anche partizioni interessanti di quel grafo*.

La sparsificazione deve essere vista come lo step preliminare prima dell'uso dell'algoritmo di clustering. *Raramente succede però che la sparsificazione splitti la matrice in componenti connesse ognuna corrispondente ad un dato cluster*. L'idea è quella di ripetere ciclicamente la sparsificazione fino ad ottenere dei grafi partizionati ben definiti. Sono stati sviluppati in letteratura diversi tipi di algoritmi di cluster graph-based e che rispondono a queste necessità.

**Minimum spanning tree (MST)** L'algoritmo inizia con l'albero di supporto o a costo minimo (minimum spanning tree) del grafo di prossimità e può essere visto come un'applicazione della sparsificazione per trovare i cluster. In particolare:

**Definizione 3.18.** Si definisce **albero di supporto a costo minimo** di un grafo un sottografo che rispetta le seguenti caratteristiche:

- Non ha cicli, o equivalentemente è un albero.
- Contiene tutti i nodi del grafo.

- Ha il minimo costo totale dei pesi associati ai suoi alberi di tutti i possibili spanning tree.

Quando si parla di albero di supporto si assume che funzioni solo con le distanze o le dissimilarità.

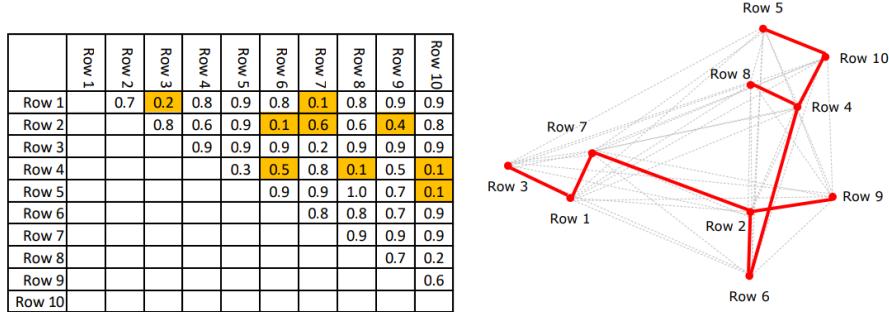


Figura 66: Albero di supporto con relativa matrice.

L'algoritmo di costruzione è il seguente:

1. Calcolare il MST per grafo di dissimilarità.
2. **Ripetere**

- Creare un nuovo cluster per rompere il collegamento corrispondente alla più grande dissimilarità.
- **Finché** rimangono solo cluster singoli.

**Opossum** è un algoritmo disegnato appositamente per clusterizzare dati multidimensionali e sparsi come i dati documentali o tipo il cestino della spesa. Si basa sullo stesso principio, ossia sulla sparsificazione del grafo. L'algoritmo nello specifico è il seguente:

1. Calcola il grafo di similarità sparsificato.
2. Partiziona il grafo in k componenti distinte (cluster) usando METIS.

Come notiamo, la misura di similarità deve essere appropriata rispetto al tipo di dati che stiamo trattando, come ad esempio la misura di Jaccard o quella del coseno. È un algoritmo molto veloce e semplice e tende a partizionare i dati in cluster di egual dimensione.

**Chamaleon** È un algoritmo di clustering agglomerativo che combina *un iniziale partizionamento dei dati*, con un ulteriore schema di clustering gerarchico che usa la nozione di vicinanza e interconnettività. L'idea chiave è che *due cluster devono essere uniti solo se il cluster risultante è simile ai cluster di partenza*. L'algoritmo si articola nei seguenti passaggi:

1. Costruire un grafo k-nearest neighbour.
2. Partizionare il grafo usando un algoritmo di partizionamento a multi livelli.
3. **Ripetere**

- Unire i cluster che preservano maggiormente la somiglianza rispetto alla relativa vicinanza e interconnettività..
- **Finché** non possono essere uniti ulteriori cluster.

Questo algoritmo riesce a clusterizzare dati spaziali considerando anche il rumore e gli outlier, riesce, inoltre, a gestire anche cluster di diversa forma e di diversa densità. Questo algoritmo ha però problemi *quando il processo di partizionamento non subisce la creazione di sotto cluster.*

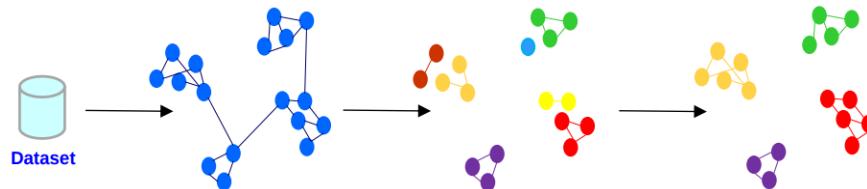


Figura 67: Schema di funzionamento dell'algoritmo Chamaleon.

### 3.4 Clustering Evaluation(\*)

La trattazione è sempre la solita, è presente un problema di clustering da risolvere. Quando si risolve un problema di questo tipo l'efficienza di un algoritmo di clustering viene valutata tramite:

- Similarità.
- Esclusività dei cluster o misure fuzzy di sovrapposizione.
- Se i cluster sono completi o parziali.

Si decide di impostare un piano sperimentale, si selezionano successivamente un certo numero di algoritmi di clustering e ai testano per vedere quali funzionano meglio. L'obiettivo è avere quindi gli stessi metodi dei classificatori per capire quando un algoritmo di clustering possa essere considerato migliore di un altro. Purtroppo il concetto di clustering è molto più complesso visto che non si conoscono esattamente il risultato da ottenere. Le **problematiche** più grosse nella valutazione sono:

- Determinare la tendenza dei cluster (che non siano fatti a caso).
- Determinare il numero corretto di cluster (qualunque cosa significhi).

Si utilizzano tre diversi tipi di indici che servono a validare i nostri cluster:

- *Esterini o supervisionati*: misurano l'estensione per cui la struttura di clustering trovata combaci con qualche struttura esterna (presupponendo di avere una vaga idea di come i dati dovrebbero organizzarsi).
- *Interni o non supervisionati*: misurano la bontà di una struttura di clustering senza tener conto di informazioni esterne. In particolare, essi possono essere:
  - Misure di coesione: determinano la connessione tra elementi di un cluster
  - misure di separazione: quanto sono ben distinti i cluster.
- *Relativi*: comparano differenti algoritmi di clustering.

### 3.4.1 Esterni o supervisionati(\*)

Si supponga di avere un partizionamento  $P = \{P_1, \dots, P_R\}$  di  $R$  insiemi disgiunti con  $m$  elementi.

Sia ora:  $C = \{C_1, \dots, C_K\}$ , la partizione ottenuta con un algoritmo di clustering in  $K$  cluster. Si compari il partizionamento  $P$  con  $C$  per vedere quali siano i casi che si realizzano. Ci sono ovviamente quattro casi che possono verificarsi:

- a.  $x$  e  $y$  appartengono allo stesso cluster di  $C$  e alla stessa categoria di  $P$
- b.  $x$  e  $y$  appartengono allo stesso cluster di  $C$  e a *diverse* categorie di  $P$
- c.  $x$  e  $y$  appartengono a *diversi* cluster di  $C$  e alla stessa categoria di  $P$
- d.  $x$  e  $y$  appartengono a *diversi* cluster di  $C$  e a *diverse* categorie di  $P$

Ovviamente le coppie che possiamo formare sono:

$$M = \frac{m(m-1)}{2} = a + b + c + d$$

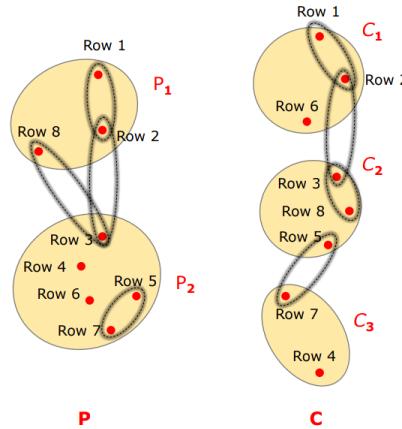


Figura 68: Partizionamenti  $C$  e  $P$  e formazione delle coppie.

Si definiscono di seguito diverse misure di similarità:

**Definizione 3.19.** Si definisce **Rand** il seguente indice:

$$R = \frac{a + d}{M} \quad R \in [0, 1]$$

**Definizione 3.20.** Si definisce **Jaccard** il seguente indice:

$$J = \frac{a}{a + b + c} \quad J \in [0, 1]$$

**Definizione 3.21.** Si definisce **Fowlkes and Mallows** il seguente indice:

$$FM = \sqrt{\frac{a}{a+b} \times \frac{a}{a+c}} \quad FM \in [0, 1]$$

**Definizione 3.22.** Si definisce  **$\Gamma$  statistics** il seguente indice:

$$\Gamma = \frac{M \times a - (a + b) \times (a + c)}{\sqrt{(a + b) \times (a + c) \times (M - a - b) \times (M - a - c)}} \quad \Gamma \in [-1, 1]$$

Per come sono definiti, più è grande il valore che assumono questi indici più sono simili  $C$  e  $P$ .

### 3.4.2 Interni o non supervisionati(\*)

Diversi indici di interni di validità del partizionamento dei cluster si basano sulla *coesione* e sulla *separazione*. L'idea è quella di ottenere un valore di coesione alto e un valore di separazione basso. In generale, *si valuta la validità totale di un cluster fatto da un insieme di K cluster come la media pesata delle validità di ogni singolo cluster.*

$$\text{overall\_validity} = \sum_{i=1}^K w_i \cdot \text{validity}(C_i)$$

In cui la validità può essere la coesione, la separazione o una combinazione lineare di queste; i pesi invece variano in base alla misura di validità del clustering. Supponendo di avere dei cluster basati su grafi si può definire la coesione e la separazione in questo modo:

**Definizione 3.23.** Si definisce **coesione** la somma dei pesi dei ponti nel grafo di prossimità che connettono i punti all'interno del cluster.

$$\text{cohesion}(C_i) = \sum_{x,y \in C_i} \text{proximity}(x, y) = \sum_{xy \in C_i} \text{similarity}(x, y)$$

Per questo motivo la coesione e la similarità sono massimizzate quando vengono minimizzate la dissimilarità/distanza.

**Definizione 3.24.** Si definisce **separazione** la somma dei pesi dei ponti nel grafo di prossimità che connettono i punti di un cluster ai punti di un altro cluster.

$$\text{separation}(C_i, C_j) = \sum_{x \in C_i, y \in C_j} \text{proximity}(x, y) = \sum_{x \in C_i, y \in C_j} \text{similarity}(x, y)$$

Per gli algoritmi di clustering basati sui prototipi invece le definizioni cambiano un pochino:

**Definizione 3.25.** Si definisce **coesione** la somma delle prossimità rispetto al prototipo all'interno del cluster.

$$\text{cohesion}(C_i) = \sum_{x \in C_i} \text{proximity}(x, c_i) = \sum_{x \in C_i} \text{similarity}(x, c_i)$$

**Definizione 3.26.** Si definisce **separazione** la prossimità tra due prototipi di due cluster diversi.

$$\text{separation}(C_i, C_j) = \text{proximity}(c_i, c_j) = \text{similarity}(c_i, c_j)$$

E' molto difficile dare una definizione sempre uguale dei pesi da fornire ai vari cluster nel momento in cui si vuole operare una misura di validità, per semplificare il problema bisogna fare riferimento alla seguente tabella.

CLUSTER MEASURE	CLUSTER WEIGHT	TYPE
$\text{cohesion}(C_i) = \sum_{x,y \in C_i} \text{proximity}(x, y)$	$\frac{1}{m_i}$	Graph-Based cohesion
$\text{cohesion}(C_i) = \sum_{x \in C_i} \text{proximity}(x, c_i)$	1	Prototype-Based cohesion
$\text{separation}(C_i) = \text{proximity}(c_i, c_j)$	$m_i$	Prototype-Based separation

Figura 69: Pesi da usare a seconda del tipo di algoritmo di clustering.

Durante questa trattazione ci si è concentrati sulle misure di coesione e di separazione come valutazione totale di un gruppo di cluster; diverse di queste misure possono essere anche applicate per valutare i singoli cluster. L'obiettivo è *ordinare i cluster secondo il loro specifico valore di validità*; in particolare, un cluster con un alto valore di coesione deve essere considerato migliore rispetto ad un cluster con un basso valore di coesione. Se sono presenti cluster non molto coesi potrebbe risultare utile separarli in dei cluster più fortemente coesi. Allo stesso modo se due cluster sono relativamente coesi ma non troppo ben separati potrebbe essere utile provare ad unirli.

*Risulta quindi utile valutare un oggetto all'interno del cluster secondo la misura in cui quell'oggetto contribuisce alla coesione o alla separazione.* Graficamente si otterebbe che gli oggetti che si trovano più "all'interno" del cluster dovrebbero contribuire in modo maggiore alla coesione, mentre quelli più esterni avranno un contributo minore. Proprio per questo motivo si definisce l'indice di Silhouette che combina coesione e separazione per l'i-esimo oggetto all'interno del cluster.

**Definizione 3.27.** Si definisce **coefficiente di Silhouette** per l'i-esimo oggetto all'interno di un cluster la seguente espressione:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \in [-1, +1]$$

Dove:

- $a_i$  è la distanza media dell'oggetto i-esimo rispetto a tutti gli oggetti del cluster a cui appartiene.
- $b_i$  è il minimo delle distanze medie dell'oggetto i-esimo da tutti gli oggetti degli altri cluster.

*Valori negativi* di questo coefficiente significano che la distanza media dei punti nello stesso cluster è maggiore della distanza media minima di quei punti rispetto ai punti di un altro cluster. Per come è definito sono preferibili *valori positivi* poiché il coefficiente assume il valore 1 quando  $a_i = 0$ .

Si possono, quindi, calcolare il *Coefficiente medio di silhouette* semplicemente calcolando la media dei coefficienti dei punti di un dato cluster. Una misura totale della bontà di un clustering può quindi essere il coefficiente medio di silhouette calcolato su tutti i punti. Per quanto riguarda il clustering gerarchico, non si usa il coefficiente di silhouette ma il **Cophenetic Correlation Coefficient**.

Questo coefficiente misura il grado di similarità tra la matrice di prossimità (P) e la matrice Cophenetic (Q) i cui elementi registrano il livello di prossimità tra coppie di osservazioni raggruppate all'interno dello stesso cluster. Il valore varia da  $[-1, 1]$ . Più il valore è vicino a 1 meglio l'algoritmo gerarchico si adatta ai nostri dati.

### 3.4.3 Paradigma di validità(\*)

Sia gli indici interni che quelli esterni sono strettamente correlati ai metodi statistici, in particolare ai test di ipotesi. Il paradigma di validità è un processo schematico che ci porta a dire se ci sia struttura o meno nei nostri dati e si articola nel seguente modo:

**Identificare la struttura e il tipo di validazione** La struttura della ricerca è basata sulla seguente ipotesi nulla: **Non c'è struttura nel dataset.**

**Determinare un indice di validazione** Cercare quale sia l'indice di validazione più appropriato per lo schema di clustering.

**Definire un'ipotesi nulla di non struttura** Il tipo di ipotesi che si va a fare dipende dal tipo di problema, può essere ad esempio :

- *Random Position* (i records si distribuiscono casualmente nello spazio n-dimensionale dei record). Viene usata per i dati razionali.
- *Random Graph* non interessa la posizione ma che ci sia una certa struttura di similarità. Viene usata per prossimità ordinali tra coppie di dati.
- *Random Label*: etichettare le osservazioni in modi differenti non cambia il tipo di coerenza che ottengo nel mio esperimento. Viene usata per tutti i tipi di dati.

**Stabilire le fondamenta della distribuzione sotto l'ipotesi nulla** Può essere operata un'analisi Monte Carlo e BootStrapping.

**Calcolare gli indici** Si calcolano tutti gli indici visti in precedenza.

**Testare l'ipotesi nulla** Si vuole arrivare ovviamente a rigettare l'ipotesi nulla. Questo rigetto non implica che ci sia necessariamente struttura nei dati ma implica che sicuramente non c'è non-struttura.

#### 3.4.4 Selezione del numero di cluster(\*)

Ha senso porsi il problema di scegliere il numero di cluster solo dopo che l'analisi precedente ha permesso di dedurre che è presente struttura adatta per un'analisi di clustering. In particolare, si utilizzano i *criteri relativi* che sono criteri che si concentrano sul confronto dei diversi risultati generati dai diversi algoritmi di clustering oppure sui risultati diversi prodotti dallo stesso algoritmo di clustering ma con parametri di input differenti. Gli indici principali per calcolare il numero di cluster sono:

- *Calinski and Harabasz*: Il valore di K corrispondente al massimo è preso per essere l'ottimale numero di cluster.
- *Dunn*: Il valore di K corrispondente al massimo è preso per essere l'ottimale numero di cluster.
- *Devies-Bouldin*: Il valore di K corrispondente al minimo è preso per essere l'ottimale numero di cluster.

Gli indici per gli algoritmi di cluster basati sulle misture probabilistiche sono invece:

- *Akaike Information Criterion (AIC)*: Il valore di K corrispondente al minimo è preso per essere l'ottimale numero di cluster.
- *Minimum Description Length (MDL)*: Il valore di K corrispondente al minimo è preso per essere l'ottimale numero di cluster.

- *Bayesian Information Criterion (BIC)*: Il valore di  $K$  corrispondente al minimo è preso per essere l'ottimale numero di cluster.

Per un algoritmo di clustering che richiede come input un numero di cluster inserito dall'utente si possono ottenere una sequenza di struttura di clustering iterando l'algoritmo di clustering  $r$  volte dove  $K$  spazia da  $K_{min}$  a  $K_{max}$ . Di seguito ora l'algoritmo che genera la sequenza delle strutture di clustering:

- Scegliere un algoritmo di clustering e un indice di validazione.
- **FOR**  $K_{min}$  **to**  $K_{max}$ :
  - **FOR**  $i = 1$  **to**  $r$ :
    - \* Lanciare l'algoritmo di clustering con  $k$  cluster e cambiare i parametri rispetto al lancio precedente.
    - \* Calcolare il valore  $q$  dell'indice di validità e impostare  $q(i) = q$
  - Scegliere il miglior valore  $q^*$  tra  $q_1, q_2, \dots, q_r$ .
  - Imporre  $Q(K) = q^*$ .

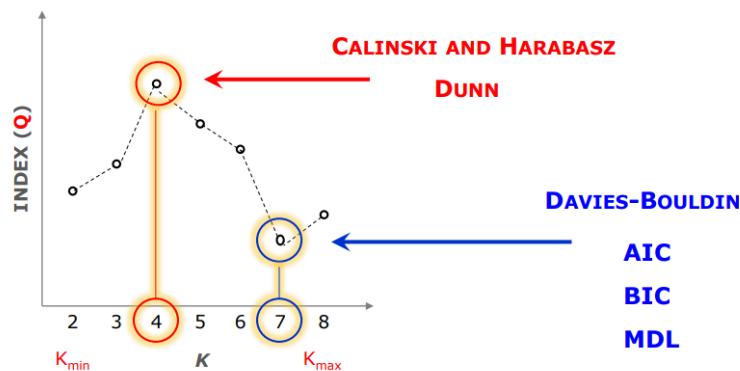


Figura 70: Scelta del numero di cluster a seconda dell'indice.

## 4 Association Analysis

### 4.1 Introduction (\*)

L'analisi di associazione aiuta a comprendere il concetto di causalità di variabili: dati certi valori di attributi cosa posso dire del valore di un altro attributo?

Le regole associative permettono di prendere delle scelte molto operative in diversi ambiti, in particolare in quello che viene chiamato il problema del *market basket analysis*. Il problema del carrello tratta il posizionamento di prodotti in un negozio. È noto che all'acquisto di un certo prodotto si tende ad acquistare altri prodotti connessi, quindi si cercano queste associazioni basandosi sul carrello della spesa dei clienti (appunto market basket) per poi capire come impostare la disposizione sugli scaffali.

**Obiettivo:** identificare quali siano gli **item associati** per poter prendere delle decisioni. In sostanza si generano **Regole associative** formate da coppie di insiemi *antecedente* e *conseguente*.

es.  $\{\text{Beer}\} \rightarrow \{\text{swiss cheese}\}$

$\{\text{antecedente}\} \rightarrow \{\text{conseguente}\}$

*È molto importante ricordare che questo tipo di analisi serve a trovare associazioni non causalità.*

L'analisi si basa sullo studio di due diversi dataset:

- *Product set:* contiene informazioni legate ai prodotti come il nome e il prezzo
- *transaction set:* contiene informazioni legate agli acquisti dei clienti, ogni record corrisponde ai prodotti presenti nel carrello del cliente

Si organizza il dataset delle transazioni in formato binario, ovvero ogni colonna indica se in una data transazione un certo prodotto sia o meno presente.

Transaction ID	swiss cheese	cherry coke	bio coke	Peppers	scrambled egg	Pomegranate	strawberries
0	1	0	0	0	1	0	0
1	0	1	0	0	0	0	1
2	0	0	0	1	1	0	0
3	0	0	0	0	0	1	0
4	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

Figura 71: Esempio: transaction set binarizzato

Di seguito viene proposto un elenco delle grandezze fondamentali per la trattazione di un problema di associazione; in particolare, si considerino:

$$I = \{i_1, i_2, \dots, i_d\}$$

Il set di tutti gli item nel market basket

$$T = \{t_1, t_2, \dots, t_N\}$$

L'insieme di tutte le transazioni

**Definizione 4.1.** Si definisce **Itemset** una collezione di zero o più item.

**Definizione 4.2.** Si definisce **K-Itemset** un itemset che contiene k item .

**Definizione 4.3.** Si definisce **empty set** l'itemset che non contiene alcun elemento.

**Definizione 4.4.** Si definisce **transaction width** il numero di item presenti in una transazione

Una transazione  $t_j$  contiene l'itemset  $X$ , se  $X$  è un sottoinsieme di  $t_j$ .

**Definizione 4.5.** Si definisce **Support count** il numero di transazioni che contentono uno specifico itemset.

$$\sigma(X) = |\{t_i | X \subseteq t_i, t_i \in T\}|$$

**Definizione 4.6.** Una **regola di associazione** è definita come:

$$X \rightarrow Y$$

dove  $X$  e  $Y$  sono itemset disgiunti ( $X \cap Y = \emptyset$ ).

Una regola viene valutata in termini di *supporto* e di *confidenza*.

**Definizione 4.7.** Si definisce **Support** la seguente grandezza:

$$s\{x \rightarrow Y\} = \frac{\sigma(X \cup Y)}{N}$$

Come si può notare il supporto serve a determinare quanto spesso una regola è applicabile dato un data set.

**Definizione 4.8.** Si definisce **Confidence** la seguente espressione:

$$c\{x \rightarrow Y\} = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

È evidente che determina quanto frequente sia  $Y$  presente in una transazione che contiene  $X$ .

Di seguito un **esempio** che fa comprendere come vengano utilizzate queste grandezze.

$$X = \{\text{swisscheese, cheddar}\} \quad Y = \{\text{dietcoke}\}$$

Si assume che:

- $\sigma(x) = 8$  (support count)
- $N = 20$  (numero di transazioni)
- $\sigma(x, y) = 6$

Allora il support e la confidence della regola  $X \rightarrow Y$  sono:

$$s\{X \rightarrow Y\} = \frac{\sigma(X \cup Y)}{N} = \frac{6}{20} = 0.3$$

$$c\{x \rightarrow Y\} = \frac{\sigma(X \cup Y)}{\sigma(X)} = \frac{6}{8} = 0.75$$

Le **motivazioni** che spingono ad utilizzare una grandezza come il supporto sono le seguenti:

- Se troppo basso potrebbe esserci un'associazione casuale
- Potrebbe non valere la pena seguire associazioni che si applicano in modo poco significativo dal punto di vista dei profitti

*Il concetto di supporto è utilizzato per eliminare regole non desiderate e condivide interessanti proprietà che possono essere sfruttate per la ricerca di regole associative efficaci. La confidenza è molto importante perché misura l'affidabilità dell'inferenza e:*

- Un'alta confidenza significa che Y sarà molto presente in transazioni con X.
- Si stima la probabilità condizionata di Y dato X.

**Definizione 4.9.** Si definisce **Association Rule Mining Problem** il problema che è formalmente definito come: dato un set di transazioni T, trovare tutte le regole con  $support \geq minsup$  e  $confidence \geq minconf$ , dove  $minsup$  e  $minconf$  sono i threshold corrispondenti alle due misure.

Ci sono diversi metodi per risolvere il problema, il primo che viene considerato è il seguente:

*Approccio a forza bruta* di association rule non è molto praticabile in quanto i tempi di computazione aumentano in modo esponenziale:

$$R = 3^d - 2^{d+1} + 1$$



Figura 72: Numero di regole calcolate in base al numero di itemset

Una strategia comunemente adottata in molti algoritmi è quella di decomporre il problema in 2 grandi supertask:

- *Generazione dei frequenti itemset*: vengono specificate tutte e sole quelle regole per cui il supporto è maggiore del  $minsup$ , gli itemset generati sono chiamati **Frequent Itemset**
- *Generazione delle regole*: vengono estratte tutte le regole con alta confidenza (maggiore di  $minconf$ ) dai Frequent Itemset trovati precedentemente, queste regole vengono chiamate **Strong Rules**

La complessità maggiore è richiesta dalla generazione dei Frequent Itemset.

## 4.2 Rule Extraction

Per comprendere l'inefficienza della generazione con l'approccio forza bruta si faccia riferimento al seguente esempio:

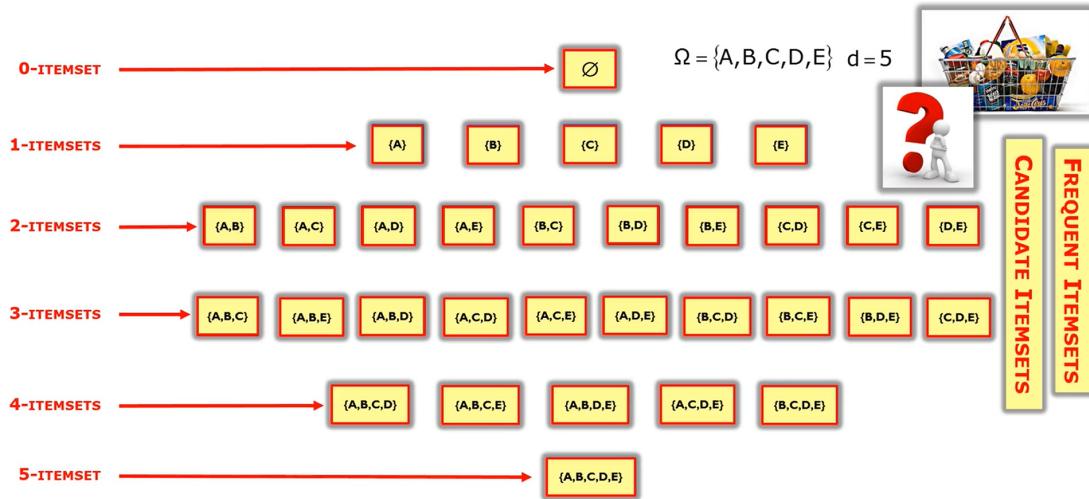


Figura 73: k-itemset brute-force

**Definizione 4.10.** Si definisce **Candidate Itemset** l'insieme di tutti gli itemset che possiamo formare.

Per come sono definiti questi insiemi potranno contenere un numero diverso di itemset; nel nostro caso il numero di itemset candidato è:

$$M = 2^d - 1 = 2^5 - 1 = 31$$

Come si può notare è presente un sistema a doppio cono tipico delle distribuzioni binomiali. Una volta che sono stati considerati tutti si è interessati solo ai più frequenti.

Se si usasse la forza bruta bisognerebbe calcolare per ogni itemset candidato il suo support count, e vedere se il suo supporto lo configura come un itemset frequente (molto dispendioso).

I confronti da effettuare sono nell'ordine di  $O(NMw)$ , dove:

- $N$  = numero di transazioni
- $M$  = numero di itemset candidato
- $w$  = massima lunghezza delle transazioni

È decisamente troppo come numero di confronti contando che molti dei quali sono inutili o poco significativi.

Vi sono due approcci per ridurre il costo computazionale della generazione di itemset frequenti:

- Ridurre il numero di candidati itemset ( $M$ ). Il principio Apriori è un metodo per eliminare alcuni candidati itemset senza contare il support count.
- Ridurre il numero di confronti anziché controllare tutte le possibili combinazioni, lo si fa con strutture dati avanzate.

**Principio Apriori** Se un itemset è frequente, allora tutti i suoi sottoinsiemi sono frequenti.

Quindi se una regola ha una frequenza bassa allora tutte le regole che prevedono come sottoinsieme la stessa non supereranno quella frequenza pertanto è inutile considerarle. Si procede attraverso il **pruning** dell'albero delle sequenze per queste soluzioni, viene chiamato **support-based pruning** (vedi immagine).

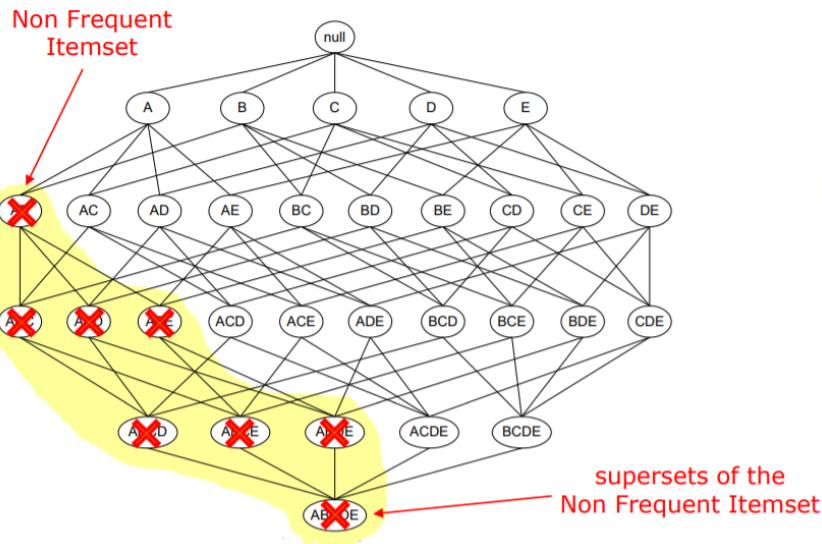


Figura 74: Esempio di support-based pruning

#### 4.2.1 Algoritmo apriori

Questo algoritmo genera due operazioni:

- **Candidate Generation:** genera nuovi candidati  $k$ -itemset basati su  $(k - 1)$ -itemset frequenti calcolati nella precedente iterazione.
- **Candidate Pruning:** questa operazione elimina alcuni candidati  $k$ -itemset usando la strategia del support-based pruning.

La complessità computazionale soffre di 4 **limiti**:

- *Support threshold*: se la soglia di supporto è troppo bassa l'albero non viene tagliato molto; questa soglia, però, non deve essere nemmeno troppo elevata altrimenti non vengono considerate potenziali associazioni rilevanti.
- *Numero di item (dimensionalità)*: se il numero di item cresce, c'è bisogno di più spazio in memoria per registrare il support count degli item; inoltre, bisogna considerare anche il costo dell'I/O per passare i dati.
- *Numero di transazioni*: l'algoritmo scorre più volte tutta la lista di transazioni, pertanto un numero alto di transazioni inficia sui tempi.
- *Average transaction width*: per dataset densi la lunghezza media delle transazioni tende ad essere grande. La massima lunghezza degli itemset frequenti tende ad aumentare; quindi, più sequenze candidate devono essere esaminate durante la generazione e support counting. In aggiunta, aumenta il numero di archi traversi nell'albero durante il support counting.

## 4.3 Maximal/Closed Frequent Itemsets

### 4.3.1 Rule Generation

Per ogni  $k$ -itemset frequente, Y può generare al limite  $2^k - 2$  regole di associazione.

Una regola di associazione può essere estratta partizionando l'itemset Y in 2 sottoinsiemi non vuoti  $\{X\}$  e  $\{Y - X\}$ , tale che  $X \rightarrow Y - X$  soddisfa il threshold di confidenza.

È molto importante notare che tutte le regole generate da itemset frequenti sono esse stesse frequenti.

In pratica, il numero di itemset frequenti prodotti da transazioni possono essere molto grandi. È utile identificare itemset rappresentativi e piccoli con i quali derivare gli itemset grandi; per questo motivo si ragiona in due rappresentazioni:

1. Maximal Frequent Itemset
2. Closed Frequent Itemset

**Definizione 4.11.** Si definisce **Maximal Frequent Itemset** un Itemset Frequente per il quale nessuno dei suoi soprainsiemi immediati sono frequenti.

Nel maximal frequent itemset per ogni nodo si verifica il vincolo della frequenza e si definisce la frontiera dove un nodo non gode più di questa proprietà; corrisponde alla massima frontiera in cui mi posso spingere.

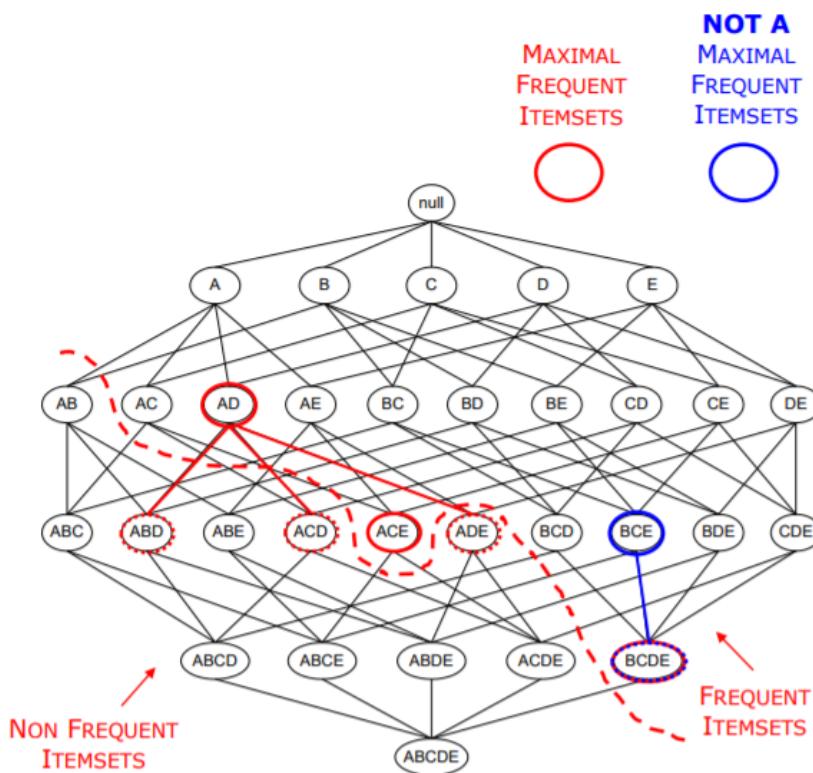


Figura 75: Esempio di frontiera di maximal frequent itemset

**Un nodo è un Maximal Frequent Itemset se è frequente e se tutte le sue estensioni non sono frequenti.**

- Fornisce una rappresentazione compatta dell'insieme di itemset che si sta cercando, la più piccola espressione in cui gli itemset sono derivabili

- Calcolato dal più piccolo insieme di itemset dal quale tutti gli itemset frequenti possono essere derivati
- È praticabile solo l'algoritmo efficiente usato esplicita la ricerca dei maximal frequent itemset senza numerare tutti i suoi sottoinsiemi

Per costruzione *però* non dice la dimensione del supporto rispetto ai suoi sottoinsiemi. In alcuni casi potrebbe servire avere una minima rappresentazione degli itemset frequenti che preservano l'informazione sul supporto.

**Definizione 4.12.** Si dice che un itemset X è **Closed Frequent Itemset** se nessun immediato super-insieme ha esattamente lo stesso support count di X. In ogni caso il suo supporto deve essere  $\geq \text{minsup}$ .

È una grandezza importante quando vi sono gruppi di prodotti venduti a blocco ignorando gli altri.

Un esempio è il seguente:

TID	a1	a2	a3	a4	a5	b1	b2	b3	b4	b5	c1	c2	c3	c4	c5
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
5	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
6	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Figura 76: Esempio di gruppi closed frequent itemset

Come si può notare in figura, ogni gruppo di variabili (Gruppo A, B e C) è perfettamente associato e non hanno bisogno di mostrare item collegati ad altri gruppi. Assumendo  $\text{minsup} = 20\%$ , il numero totale di itemset frequenti è:  $3 \cdot (2^5 - 1) = 93$ .

In ogni caso vi sono solo 3 closed frequent itemset:

$$\{a1, a2, a3, a4, a5\}$$

$$\{b1, b2, b3, b4, b5\}$$

$$\{c1, c2, c3, c4, c5\}$$

Questo tipo di itemset sono utili per rimuovere **regole di associazione ridondanti**.

**Definizione 4.13.** Si dice che una regola di associazione  $X \rightarrow Y$  è **ridondante** se esiste un'altra regola  $X' \rightarrow Y'$  che rispetta le seguenti proprietà:

- $X \subseteq X'$
- $Y \subseteq Y'$
- $s(X \rightarrow Y) = s(X' \rightarrow Y')$
- $c(X \rightarrow Y) = c(X' \rightarrow Y')$

Di seguito la gerarchia dei frequent itemset:

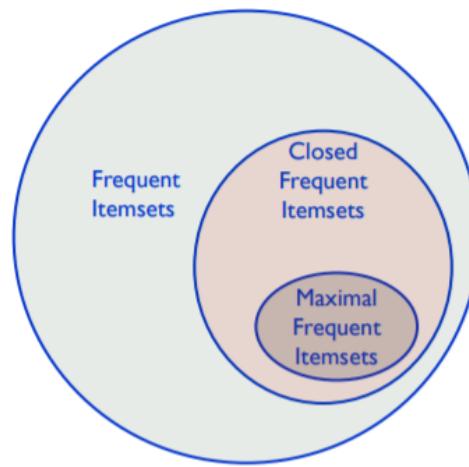


Figura 77: Gerarchia dei frequent itemset

Come si può notare i Maximal Frequent Itemset sono inclusi nei Closed Frequent Itemset perché nessun maximal può avere lo stesso support count del suo immediato superset.

## 4.4 Rules Evaluation (\*)

Generati gli insiemi di pattern potenzialmente utili, bisogna ordinarli in base al loro livello di attrattivit per il dominio di applicazione. La valutazione della qualit di una regola associativa viene effettuata secondo due criteri:

- **Statistical arguments:** per patterns di items indipendenti e coperti da poche transazioni. Il problema  che *possono catturare relazioni spurie* nei dati, per evitare:
  - *Objective Interestingness Measure*: usare statistiche derivate dai dati per determinare quali pattern sono interessanti
  - *Supporto, confidenza e correlazione*
- **Subjective arguments:** un pattern  considerato non interessante a meno che riveli informazioni inaspettate riguardo ai dati e alla conoscenza:  
*Esempio:* forte associazione tra acquisto di pannolini e birra.  difficile fare questo tipo di valutazioni, richiede una considerevole quantit di informazioni pregresse dagli esperti di domino.

*È quindi meglio cercare di applicare un approccio più oggettivo alla valutazione: data-driven, indipendente dal dominio in cui si richiede un minimo input dagli utenti (solo dei threshold) e calcolato basandosi sulle frequenze calcolate in una Contingency Table.*

	B	not B	
A	$f_{11}$	$f_{10}$	$f_{1+}$
not A	$f_{01}$	$f_{00}$	$f_{0+}$
	$f_{+1}$	$f_{+0}$	N

Figura 78: Contingency table

Dove:

- $f_{1+}$  = support count di A
- $f_{+1}$  = support count di B
- $N$  = numero di transazioni totali

Si assuma che un direttore di un minimarket voglia analizzare la relazione tra le persone che bevono tè e persone che bevono caffè. La seguente contingency table è ottenuta considerando le transazioni disponibili:

	coffee	not coffee	
tea	150	50	<b>200</b>
not tea	650	150	<b>800</b>
	<b>800</b>	<b>200</b>	1,000

Figura 79: Contingency table

Si valuti l'associazione: tea  $\rightarrow$  coffee, si ottiene:

$$support = \frac{f_{11}}{N} = \frac{150}{1000} = 15\%$$

$$confidence = \frac{f_{11}}{f_{1+}} = \frac{150}{200} = 75\%$$

Ad una prima occhiata sembrerebbe che le persone che bevono tè tendano a bere anche caffè (vedi confidenza). Si nota, però, che le persone che bevono caffè, a prescindere dal tè sono l'80% ( $\frac{800}{1000}$ ), mentre la frazione dei bevitori di tè che bevono caffè è solo il 75%.

Da questo ragionamento si può concludere che il fatto di bere tè non influisca sulle persone che bevono caffè. Infatti, nonostante l'associazione abbia un alto livello di confidenza (75%), non si può ignorare il supporto dell'itemset conseguente (80%). Pertanto, vengono definiti altri indici:

**Definizione 4.14.** Si definisce l'indice **Lift** il tasso di confidenza rispetto al supporto del conseguente

$$Lift = \frac{c(A \rightarrow B)}{s(B)}$$

**Definizione 4.15.** Si definisce l'indice **Interest Factor**, equivalente al Lift ma per attributi binari, in questo modo:

$$I(A, B) = \frac{s(A, B)}{s(A)s(B)} = \frac{Nf_{11}}{f_{1+}f_{+1}}$$

I valori sono così classificati:

- = 1 se A e B sono indipendenti
- > 1 se A e B sono positivamente associati
- < 1 se A e B sono negativamente associati

È molto importante effettuare un' **analisi di correlazione** (per attributi binari simmetrici): analizza la relazione tra coppie di attributi. Gli attributi continui possono essere analizzati con la correlazione di Pearson; la correlazione per attributi binari è misurata, invece, usando il  $\phi$ -coefficient:

**Definizione 4.16.** Si definisce il  **$\phi$ - coefficient** l'espressione che segue:

$$\phi = \frac{f_{11}f_{00} - f_{01}f_{10}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}}$$

Il valore di questo coefficiente varia da  $[-1, +1]$ . Se gli attributi sono statisticamente indipendenti il valore è 0.

**Definizione 4.17.** Si definisce l'indice **IS Measure** (per attributi binari asimmetrici)

$$IS(A, B) = \sqrt{I(A, B)s(A, B)} = \frac{s(A, B)}{\sqrt{s(A)s(B)}}$$

Se A e B sono indipendenti è invece definito come segue:

$$IS(A, B) = \sqrt{s(A)s(B)}$$

Questo indice soffre dello stesso problema della correlazione, il valore può essere grande anche per associazioni incorrelate o negativamente correlate. Vi sono altri tipi di misure per l'analisi delle relazioni tra coppie di variabili binarie.

**Definizione 4.18.** Una misura **M** si dice **Simmetrica** se:  $M(A \rightarrow B) = M(B \rightarrow A)$

*Per come è definito si può quindi dire che l' Interest Factor è una misura simmetrica.*

**Definizione 4.19.** Una misura **M** si dice **Asimmetrica** se:  $M(A \rightarrow B) \neq M(B \rightarrow A)$

*Per come è definita si può dire che la confidenza è una misura asimmetrica.*  
Di seguito la tabella relativa agli indici più utilizzati:

Simmetrico	Asimmetrico
Correlazione ( $\phi$ )	Gini index
Odds ratio	Mutual information
Kappa	Certainty factor
Interest (I)	Added value
Cosine (IS)	J-measure
Jaccard	Goodman-Kruskal
Collective strength	

È importante capire che ciascuna misura è adatta per analizzare un certo tipo di associazioni come basket market analysis o document analysis. In base al caso bisogna utilizzare gli indici migliori.

*In questa analisi sono stati presentati indici relativi a valutazioni per coppie di attributi binari, ma è possibile estendere l'analisi a più di due attributi usando tabelle delle frequenze in una Contingency Table multi-dimensionale. Gli indici come il Supporto Interest Factor e IS si prestano a questa trattazione.*

## 4.5 Simpson's Paradox

Si consideri la seguente situazione:

Un direttore di un mini-market racconta di una curiosa scoperta, legata agli item birra e hot dogs. Una società di consulenza da lui pagata per analizzare i suoi dati di vendita ha scoperto che i clienti che comprano birra sono meno tentati di acquistare hot dogs rispetto a quelli che non acquistano la birra. Il direttore, però, è convinto del contrario, lo sa per esperienza professionale che chi acquista birra tende ad acquistare hot dogs.

Come si può risolvere il paradosso?

Si consideri la seguente tabella:

	hot dogs	NOT hot dogs	Total
beer	70	98	168
NOT beer	45	55	100
	115	153	

Figura 80: Esempio: birra - hot dog

Si considerino le seguenti regole con le relative confidenze:

- $\{\text{beer}\} \rightarrow \{\text{hot dogs}\}$  - confidence = 42%
- $\{\text{NOT beer}\} \rightarrow \{\text{hot dogs}\}$  - confidence = 45%

Da questi numeri si può dedurre che i clienti che acquistano birra sono meno inclini (42%) ad acquistare hot dog rispetto a quelli che non acquistano birra (45%).

Si analizzino gli stessi dati ma categorizzando se il cliente è single o no:

		hot dogs	NOT hot dogs	Total
single	beer	30	10	40
	NOT beer	5	40	45
NOT single	beer	40	88	128
	NOT beer	40	15	55

Figura 81: Esempio: cliente single: birra - hot dog

Pertanto si possono derivare queste due coppie di regole:

- Single
  - $\{\text{beer}\} \rightarrow \{\text{hot dogs}\}$  - confidence = 75%

- {NOT beer} → {hot dogs} - confidence = 11%
- NOT Single
  - {beer} → {hot dogs} - confidence = 31%
  - {NOT beer} → {hot dogs} - confidence = 73%

Da questi dati si può affermare che: i single che acquistano birra sono più inclini (75%) ad acquistare anche hot dog rispetto a quelli che non acquistano birra (11%). Quindi sia l'azienda di consulenza che il direttore del mini-market avevano ragione soltanto che per comprenderlo bisognava categorizzare i clienti.

**Paradosso di Simpson** (Yule-Simpson effect): è un paradosso della probabilità e statistica, in cui una tendenza appare in diversi gruppi di dati ma sparisce o si inverte quando questi gruppi sono combinati.

Bisogna applicare una appropriata stratificazione dei dati per evitare la generazione di associazioni spurie risultati dal paradosso di Simpson.

Esempio: per i dati del market-basket, la catena di supermercati dovrebbe stratificarli secondo la location del negozio, mentre i dati sanitari da vari pazienti dovrebbero essere stratificati secondo fattori quali l'età o il genere.