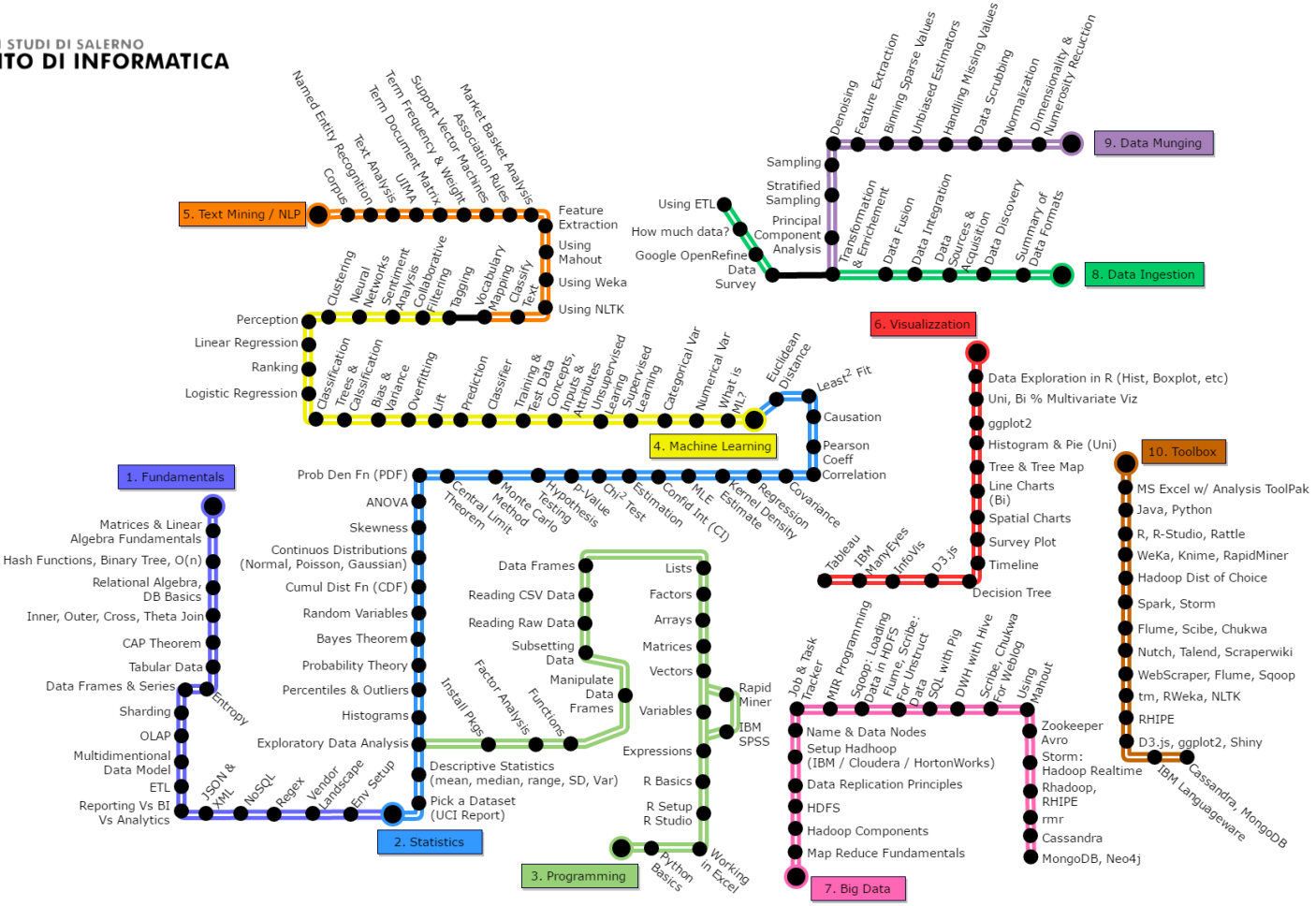
UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA



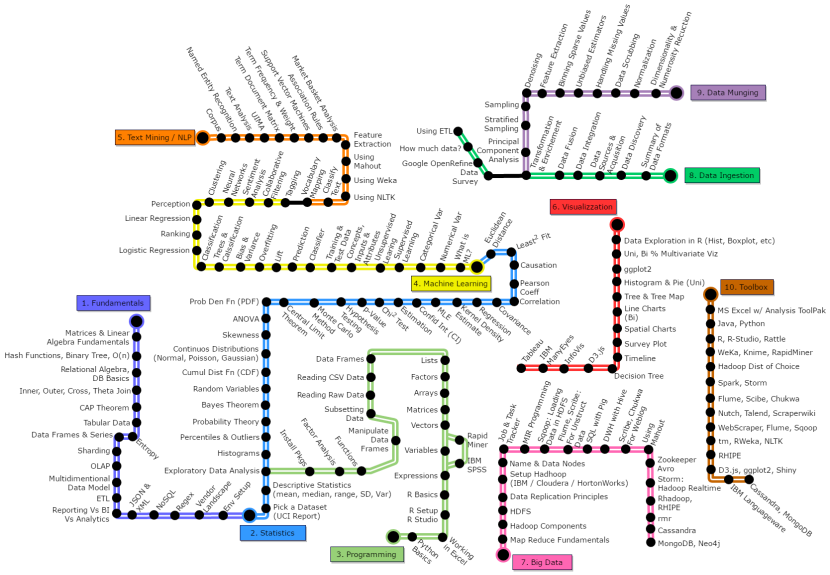
Fondamenti di Data Science e Machine Learning

Data Mining - *Modelli, metodi ed applicazioni*

Prof. Giuseppe Polese, aa 2024-25

Overview

1. **Tecniche e algoritmi di base per l'estrazione di conoscenza**
2. Regole di Associazione
3. Algoritmo Apriori



Knowledge Discovery

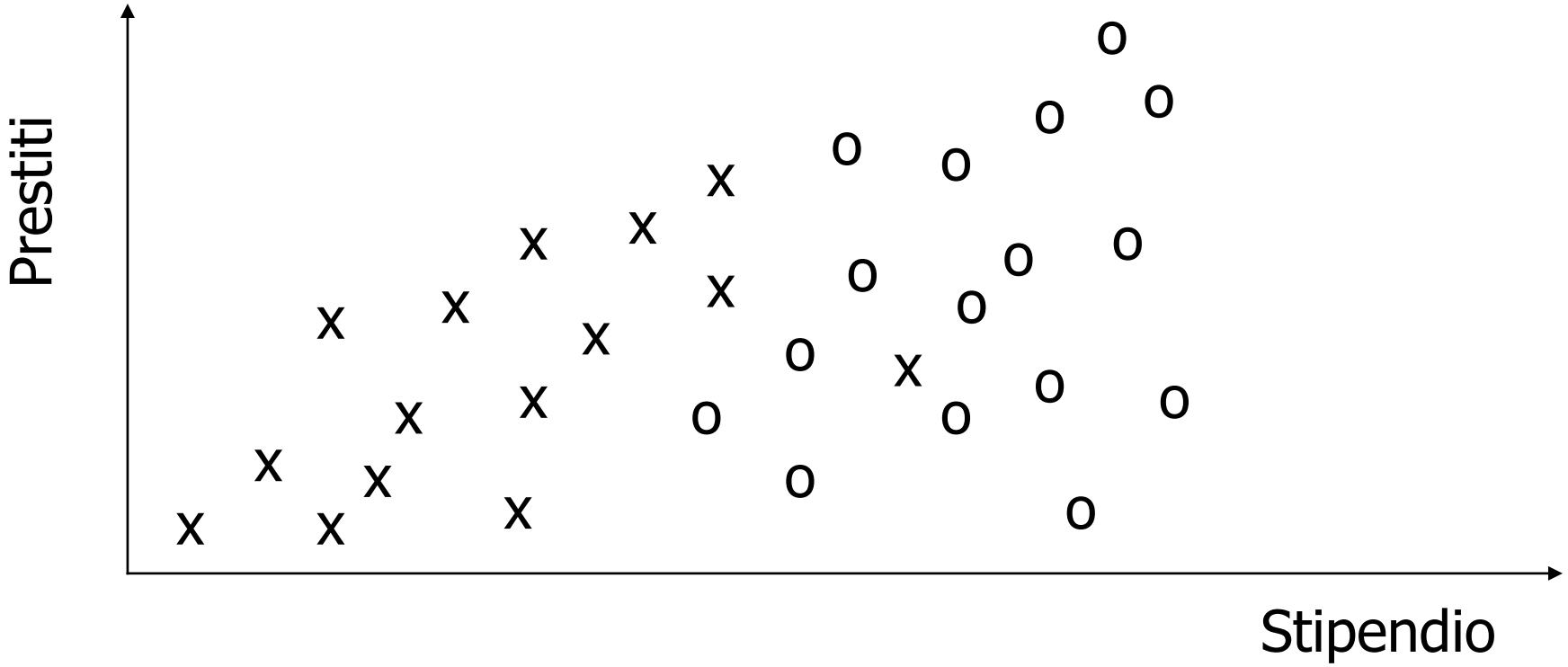
- La maggior parte delle aziende dispone di enormi basi di dati contenenti dati di tipo operativo
  - Queste basi di dati costituiscono una potenziale miniera di informazioni utili

Knowledge Discovery

- Processo di estrazione di pattern di informazioni dai dati esistenti:
  - valide
  - precedentemente sconosciute
  - potenzialmente utili
  - comprensibili

  [Fayyad, Piatesky-Shapiro, Smith 1996]

Esempio

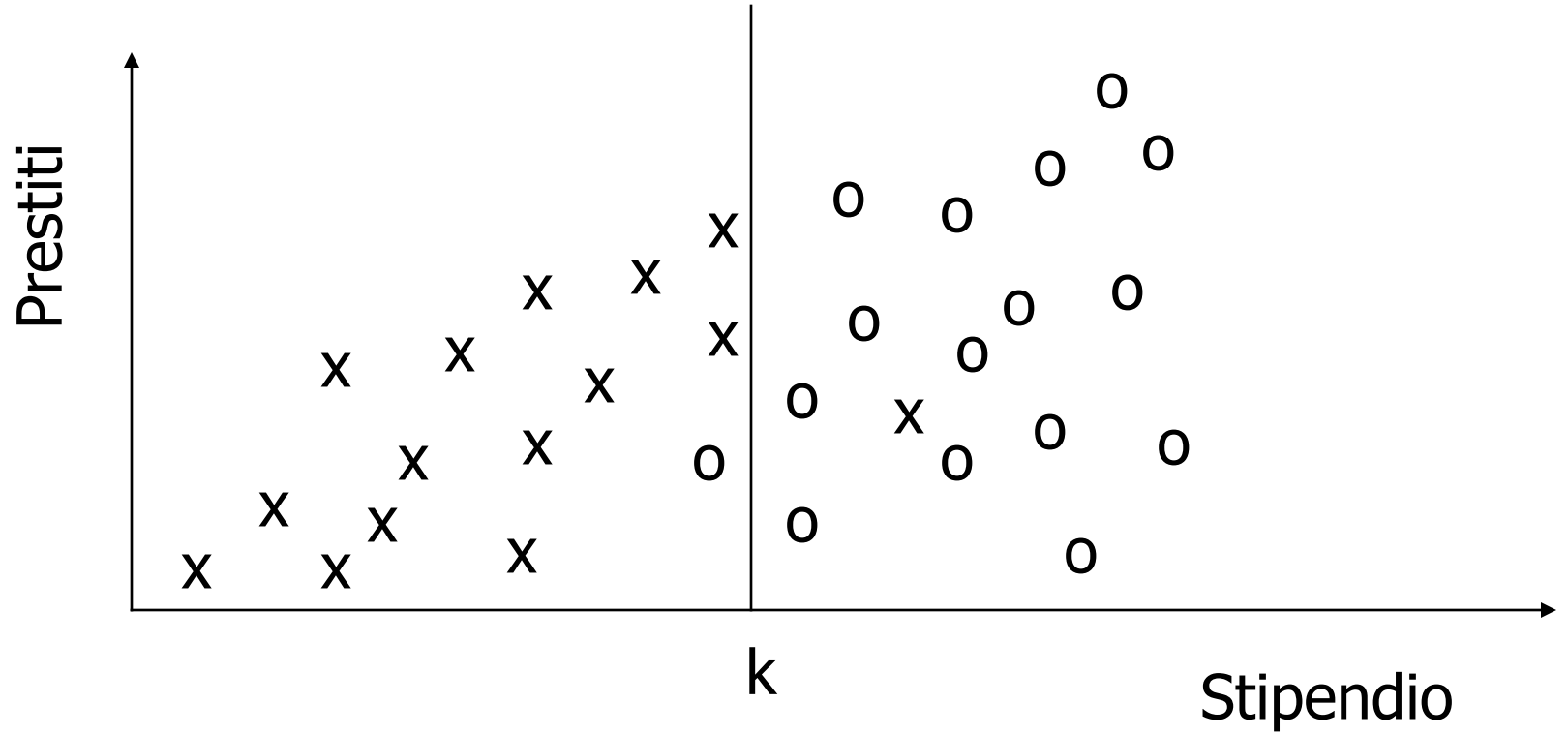


Persone che hanno ricevuto un prestito dalla banca:
x: persone che hanno mancato la restituzione di rate
o: persone che hanno rispettato le scadenze

Knowledge Discovery

- Un processo di KD si basa sui seguenti elementi:
  - *Dati*: insieme di informazioni contenute in una base di dati o data warehouse
  - *Pattern*: espressione in un linguaggio opportuno che descrive in modo succinto le informazioni estratte dai dati
    - regolarita`
    - informazione di alto livello

Esempio estrazione di informazione

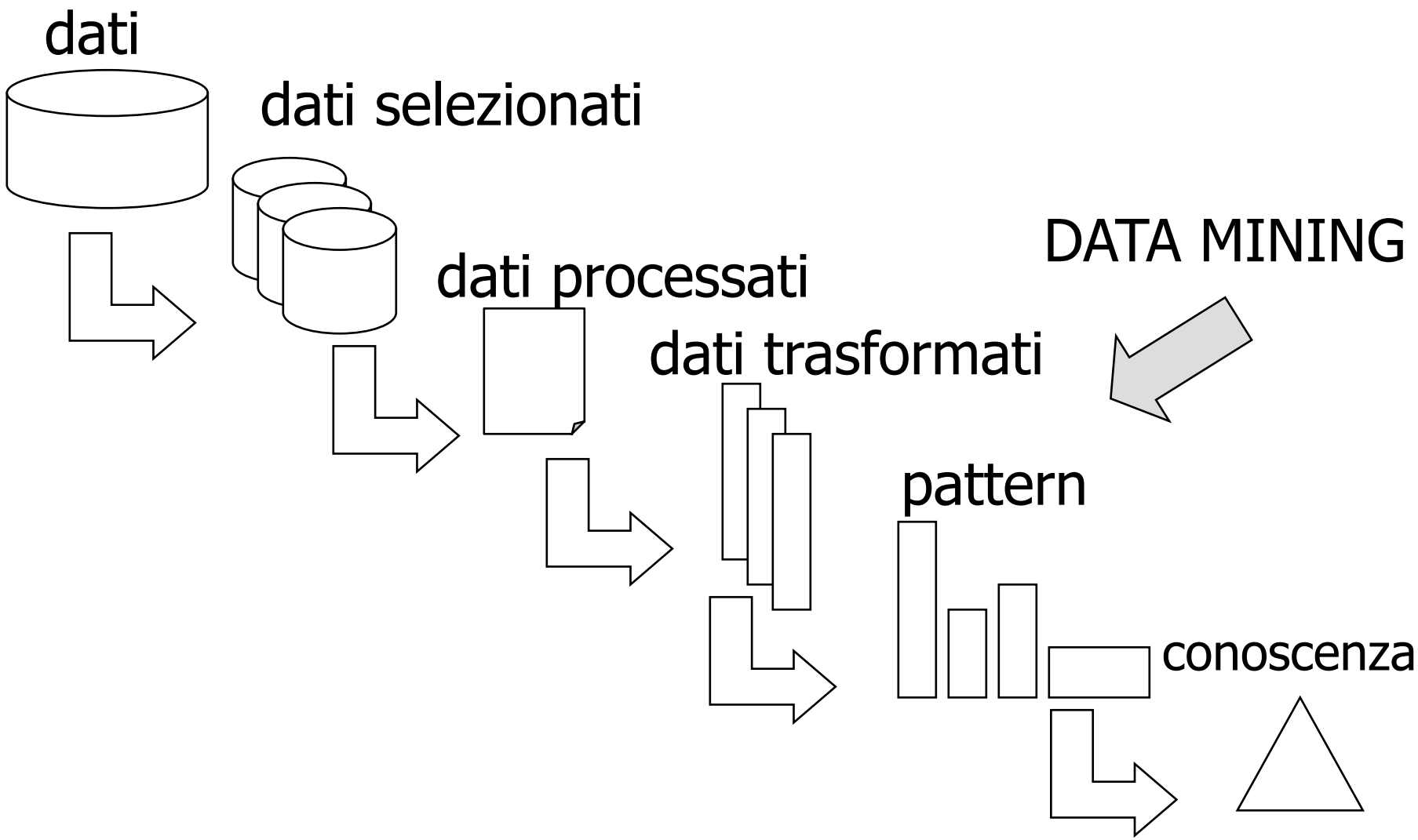


IF stipendio < k THEN mancati pagamenti

Caratteristiche dei pattern

- *Validità*: i pattern scoperti devono essere validi su nuovi dati con un certo grado di certezza
- *Novità*: misurata rispetto a variazioni dei dati o della conoscenza estratta
- *Utilità*
  - Esempio: aumento di profitto atteso dalla banca associato alla regola estratta
- *Comprensibilità*: misure di tipo
  - sintattico
  - semantico

Processo di estrazione

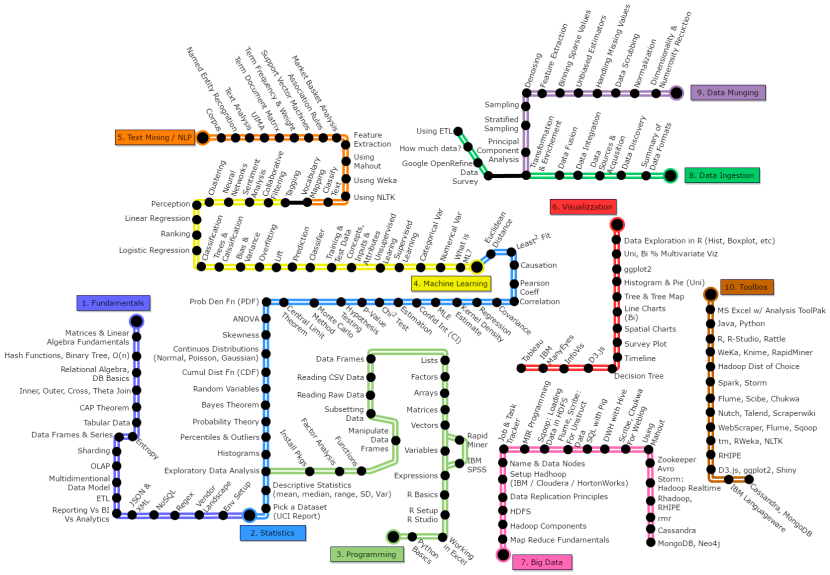


Processo di estrazione

- Il processo di estrazione in genere parte da insiemi di dati eterogenei
- deve garantire adeguata efficienza, ipotizzando che i dati risiedano su memoria secondaria
- deve essere scalabile
- deve associare misure di qualità ai pattern estratti
- deve permettere di applicare criteri di estrazione diversificati

Overview

1. Tecniche e algoritmi di base per l'estrazione di conoscenza
2. **Regole di Associazione**
3. Algoritmo Apriori



Regole di associazione

- Dati del problema:
  - I insieme di items
    - prodotti venduti da un supermercato
  - *transazione T*: insieme di items $T \subseteq I$
    - oggetti acquistati nella stessa transazione di cassa al supermercato
  - *base di dati D*: insieme di transazioni

Regole di associazione (2)

- Regola di associazione $X \Longrightarrow Y$, con $X,Y \subseteq I$
- Supporto (X) = # transazioni che contengono X in D
- *Supporto* ($X \Longrightarrow Y$): supporto($X \cup Y$)
  - rilevanza statistica
- *Confidenza*($X \Longrightarrow Y$): supporto($X \cup Y$)/supporto(X)
  - significatività dell'implicazione

Esempio

$$\text{Latte} \Rightarrow \text{Uova}$$

- Supporto: il 2% delle transazioni contiene entrambi gli elementi
- Confidenza: il 30% delle transazioni che contengono latte contiene anche uova

In una base di dati relazionale

- Items: valori associati ad un certo attributo in una certa relazione
- transazione: sottoinsieme di items, raggruppati rispetto al valore di un altro attributo (ad esempio un codice)

T1	111	201	01/05/1999	ink	1
	111	201	01/05/1999	milk	3
	111	201	01/05/1999	juice	6
T2	112	105	03/06/1999	pen	1
	112	105	03/06/1999	ink	1
	112	105	03/06/1999	milk	1
T3	113	106	10/05/1999	pen	1
	113	106	10/05/1999	milk	1
T4	114	201	01/06/1999	pen	2
	114	201	01/06/1999	ink	2
	114	201	01/06/1999	juice	4

Applicazioni

- Analisi market basket
  - $* \Longrightarrow$ uova
    - cosa si deve promuovere per aumentare le vendite di uova?
  - Latte $\Longrightarrow *$
    - quali altri prodotti devono essere venduti da un supermercato che vende latte?
- Dimensione del problema:
  - oggetti: 10^4, 10^5; transazioni: > 10^6
  - base di dati: 10-100 GB

Regole di associazione

- Problema:
  - determinare tutte le regole con supporto e confidenza superiori ad una soglia data

Esempio

TRANSACTION ID	OGGETTI ACQUISTATI
1	A,B,C
2	A,C
3	A,D
4	B,E,F

- Assumiamo:
  - supporto minimo 50%
  - confidenza minima 50%

Esempio (continua)

TRANSACTION ID	OGGETTI ACQUISTATI
1	**A**,B,**C**
2	**A,C**
3	**A**,D
4	B,E,F

- Regole ottenute:
  - A $\Rightarrow$ C supporto 50% confidenza 66.6
  - C $\Rightarrow$ A supporto 50% confidenza 100%

Determinazione regole di associazione

- Decomposizione problema
  - Trovare tutti gli insiemi di item (itemset) che hanno un supporto minimo (*frequent itemset*)
    - Algoritmo fondamentale: APRIORI

    [Agrawal, Srikant 1994]
  - Generazione delle regole a partire dai frequent itemset

Esempio

- Passo 1: estrazione dei frequent itemset

TRANSACTION ID	OGGETTI ACQUISTATI
1	A,B,C
2	A,C
3	A,D
4	B,E,F

- supporto minimo 50%

FREQUENT ITEMSET	SUPPORTO
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

Esempio (continua)

- Passo 2: estrazione regole
  - confidenza minima 50%
  - Esempio: regola A $\Longrightarrow$ C
    - supporto {A,C} = 50%
    - confidenza = supporto {A,C}/supporto{A} = 66.6%
  - regole estratte
    - A $\Longrightarrow$ C supporto 50%, conf. 66.6%
    - C $\Longrightarrow$ A supporto 50%, conf. 100%

Algoritmo Apriori

- Ad ogni passo:
  - costruisce un insieme di itemset candidati
  - conta il numero di occorrenze di ogni candidato (accedendo alla base di dati)
  - determina i candidati che sono frequent itemset
- Al passo k:
  - C_k: insieme di itemset candidati di dimensione k (potenzialmente frequent itemset)
  - L_k: insieme di frequent itemset di dimensione k

Algoritmo Apriori

L_1 = {singoli items frequenti}

for (k=1, $L_k \neq \{\}$, k++)

begin

C_{k+1} = nuovi candidati generati da L_k

foreach transazione t in D do

incrementa il conteggio di tutti i candidati in C_{k+1}

che sono contenuti in t

L_{k+1} = candidati in C_{k+1} con supporto minimo

end

frequent itemsets = U_k L_k

Apriori: generazione candidati

- Proprietà: ogni sottoinsieme di un frequent itemset è un frequent itemset
- Soluzione A:
  - dato L_k , C_{k+1} si genera aggiungendo un item agli itemset in L_k,
- Soluzione B (ottimizzata rispetto ad A)
  - Da C_k si possono cancellare tutti i candidati che contengono un sottoinsieme non frequent
  - In pratica:
    - calcolo il join di L_k con L_k, imponendo che almeno k-1 elementi siano uguali
    - elimino dal risultato gli itemset che contengono sottoinsiemi non frequenti

Esempio

- Base di dati D

TID	Items
100	1, 3, 4
200	2, 3, 5
300	1, 2, 3, 5
400	2, 5

- Supporto minimo 50% (cioè almeno 2 transazioni)
- nel seguito con supporto intendiamo il numero di transazioni e non la percentuale per comodità

Esempio: Soluzione A

Scansione D (1)

C_1

Itemset	Supporto (*4)
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

Itemset	Supporto
{1}	2
{2}	3
{3}	3
{5}	3

Esempio (continua)

Itemset
{1, 2}
{1, 3}
{1, 4}
{1, 5}
{2, 3}
{2, 4}
{2, 5}
{3, 4}
{3, 5}

C_2 ⟶ Scansione D (2)

C_2

Itemset	Supporto
{1, 2}	1
{1, 3}	2
{1, 4}	1
{1, 5}	1
{2, 3}	2
{2, 4}	0
{2, 5}	3
{3, 4}	1
{3, 5}	2

⟶

L_2

Itemset	Supporto
{1, 3}	2
{2, 3}	2
{2, 5}	3
{3, 5}	2

Esempio (continua)

Itemset
{1, 3 ,2}
{1, 3, 4}
{1, 3, 5}
{2, 3, 4}
{2, 3, 5}
{2, 5, 1}
{2, 5, 4}
{3, 5, 4}

C_3 ⟶ Scansione D (3)

C_3

Itemset	Supporto
{1, 3 ,2}	1
{1, 3, 4}	1
{1, 3, 5}	1
{2, 3, 4}	0
{2, 3, 5}	2
{2, 5, 1}	1
{2, 5, 4}	0
{3, 5, 4}	0

⟶

L_3

Itemset	Supporto
{2, 3, 5}	2

Esempio: Soluzione B

Scansione D (1)

C_1

Itemset	Supporto (*4)
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

Itemset	Supporto
{1}	2
{2}	3
{3}	3
{5}	3

Esempio (continua)

Itemset
{1, 2}
{1, 3}
{1, 5}
{2, 3}
{2, 5}
{3, 5}

C_2 ⟶ Scansione D (2)

C_2

Itemset	Supporto
{1, 2}	1
{1, 3}	2
{1, 5}	1
{2, 3}	2
{2, 5}	3
{3, 5}	2

⟶

L_2

Itemset	Supporto
{1, 3}	2
{2, 3}	2
{2, 5}	3
{3, 5}	2

Esempio (continua)

Itemset

{2, 3, 5} C_3 ⟶ Scansione D (3)

C_3

Itemset	Supporto
{2, 3, 5}	2

⟶ L_3

Itemset	Supporto
{2, 3, 5}	2

Esempio

- Supporto = 75% (3 transazioni su 4)
- relazione:

Transid	custid	date	item	qty
111	201	01/05/1999	pen	2
111	201	01/05/1999	ink	1
111	201	01/05/1999	milk	3
111	201	01/05/1999	juice	6
112	105	03/06/1999	pen	1
112	105	03/06/1999	ink	1
112	105	03/06/1999	milk	1
113	106	10/05/1999	pen	1
113	106	10/05/1999	milk	1
114	201	01/06/1999	pen	2
114	201	01/06/1999	ink	2
114	201	01/06/1999	juice	4

Esempio: soluzione A

- Level 1:
  - L1: {pen} 1, {ink} 3/4, {milk} 3/4
- Level 2:
  - C2: {pen, ink}, {pen,milk}, {pen, juice}, {ink, milk}, {ink, juice}, {milk, juice}
  - L2: {pen,ink} 3/4, {pen, milk} 3/4
- Level 3:
  - C3: {pen, ink, milk}, {pen, ink, juice}, {pen, milk,juice}
  - L3: nessuno

Esempio: soluzione B

- Level 1:
  - L1: {pen} 1, {ink} 3/4, {milk} 3/4
- Level 2:
  - C2: {pen, ink}, {pen,milk}, {ink, milk}
  - L2: {pen,ink} 3/4, {pen, milk} 3/4
- Level 3:
  - C3: nessuno

Generazione regole

- Il supporto dei frequent itemset è già superiore ad una certa soglia, vogliamo costruire regole in cui anche la confidenza è maggiore di una certa soglia.
- Sia X un frequent itemset, dividiamo X in due itemset LHS e RHS tali che X = LHS U RHS e consideriamo la regola LHS $\Longrightarrow$ RHS.
- La sua confidenza è supporto(X)/supporto(LHS), ma per la proprietà dei frequent itemset, essendo LHS sottoinsieme di un frequent item set è esso stesso frequent, quindi il suo supporto è stato calcolato nella prima fase dell'algoritmo Apriori. Calcolo quindi la confidenza e verifico se supera il limite posto.

Esempio

- Frequent itemset: {pen} 1, {ink} 3/4, {milk} 3/4, {pen,ink} 3/4, {pen, milk} 3/4
- voglio costruire regole non banali (confidenza = 1)
- considero {pen,milk}
  - supporto({pen,milk}) = 3/4
  - supporto({pen}) =1
  - supporto({milk}) =3/4
  - confidenza(pen => milk) = 3/4 non restituita
  - confidenza(milk => pen) = 1 restituita
- considero {pen,ink}
  - supporto({pen,ink} = 3/4
  - supporto({pen}) =1
  - supporto({ink}) =3/4
  - confidenza(pen => ink) = 3/4 non restituita
  - confidenza(ink => pen) = 1 restituita

Estensioni

- In molti casi gli item sono organizzati gerarchicamente

- il supporto di un itemset può solo aumentare se un item viene rimpiazzato con un suo antenato nella gerarchia

Estensioni

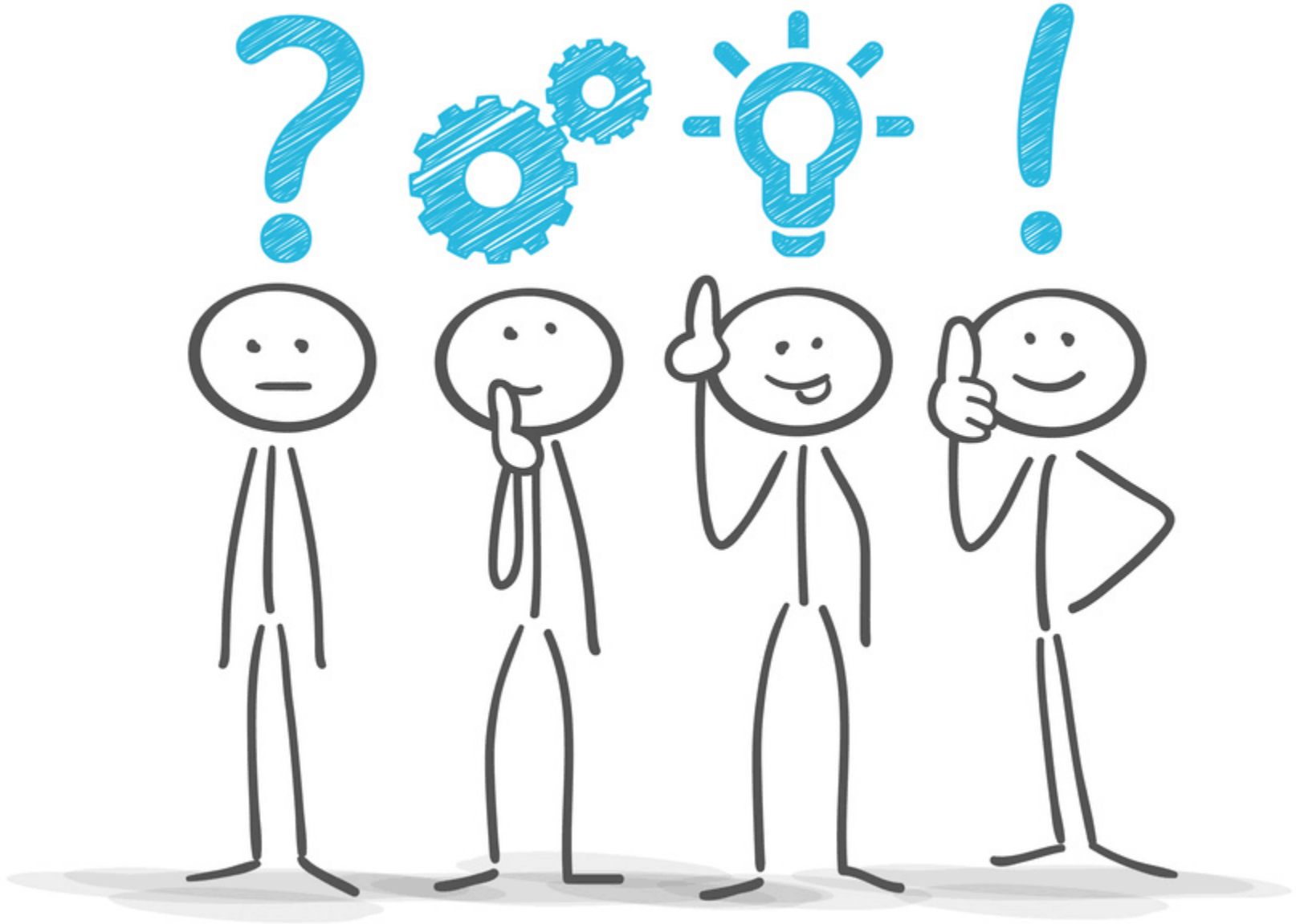
- Supponendo di avere informazioni anche per gli item generalizzati, si possono calcolare le regole nel modo usuale

Transid	custid	date	item	qty
111	201	5/1/99	stationery	3
111	201	5/1/99	beverage	9
112	105	6/3/99	stationery	2
112	105	6/3/99	beverage	1
113	106	5/10/99	stationery	1
113	106	5/10/99	beverage	1
114	201	6/1/99	stationery	4
114	201	6/1/99	beverage	4

- Per esercizio: provare a calcolare le nuove regole di associazione

Estensioni

- Determinazione regole di associazione nel contesto di sottoinsiemi di dati che soddisfano determinate condizioni
  - se una penna è acquistata da un certo cliente, allora è probabile che lo stesso cliente comprerà anche latte
    - si considerano solo gli acquisti di un certo cliente
- Pattern sequenziali
  - tutti gli item acquistati da un certo cliente in una certa data definiscono un itemset
  - gli itemset associati ad un cliente possono essere ordinati rispetto alla data, ottenendo una sequenza di itemset (pattern sequenziale)
  - il problema è determinare tutti i pattern sequenziali con un certo supporto



Regole di classificazione e regressione

- Regola di classificazione:

  $P_1(X_1) \wedge \ldots \wedge P_k(X_k) \Longrightarrow Y = c$

  Y attributo *dipendente*

  X_i attributi *predittivi*

  $P_i(X_i)$ condizione su attributo X_i

- Due tipi di attributi:
  - numerici
  - categorici (tipi enumerazione)

Regole di classificazione e regressione(2)

- Se l'attributo dipendente è categorico, si ottiene una regola di *classificazione*
- se l'attributo dipendente è numerico, si ottiene una regola di *regressione*
- se X_i è numerico, $P_i(X_i)$ in genere coincide con $l_i \leq X_i \leq g_i$, con l_i e g_i appartenenti al dominio di X_i
- X_i è categorico, $P_i(X_i)$ coincide con $X_i \in \{v_{i,\ldots,} v_n\}$

Esempio

- InfoAssicurazioni(età:int, tipo_auto:string,rischio:{alto,basso})

- Regola di classificazione

$$\text{età} > 25 \wedge \text{tipo\_auto} \in \{\text{Sportiva, utilitaria}\} \Longrightarrow \text{rischio = alto}$$

Supporto e confidenza

- Il supporto di una condizione C è la percentuale di tuple che soddisfano C
- il supporto di una regola $C_1 \Rightarrow C_2$ è il supporto della condizione $C_1 \wedge C_2$
- la confidenza di una regola $C_1 \Rightarrow C_2$ è la percentuale di tuple che soddisfano C_1 che soddisfano anche C_2

Applicazioni

- Le regole di classificazione/regressione vengono utilizzate in tutte le applicazioni che presentano problematiche di classificazione dei dati:
  - classificazione risultati scientifici, in cui gli oggetti devono essere classificati in base ai dati sperimentali rilevati
  - analisi dei rischi
  - previsioni economiche

Classificazione

- Il problema della classificazione può essere introdotto in un modo più generale
- Dati del problema:
  - insieme di classi (valori per un attributo categorico)
  - insieme di oggetti etichettati con il nome della classe di appartenenza (*training set*)
- Problema:
  - trovare il profilo descrittivo per ogni classe, utilizzando le feature dei dati contenuti nel training set, che permetta di assegnare altri oggetti, contenuti in un certo *test set,* alla classe appropriata

Settori di sviluppo

- Statistica
- machine learning
  - alberi di decisione
  - inductive logic programming
- reti neurali
- sistemi esperti
- data mining

Ipotesi di funzionamento

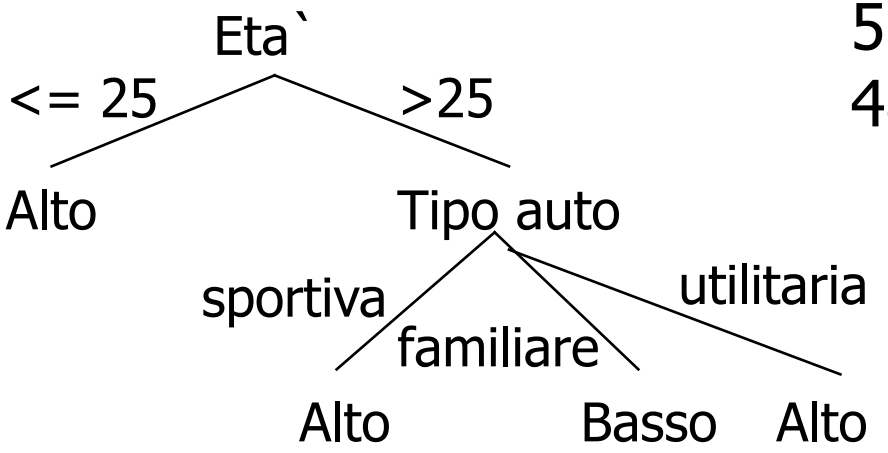
- Training set contenuto nella memoria principale del sistema
- Nei DB attuali possono essere disponibili anche Mbyte di training set
  - dimensioni significative del training set possono migliorare l'accuratezza della classificazione

Alberi di decisione

- Gli alberi di decisione rappresentano un approccio alla classificazione molto utilizzato
- permettono di rappresentare con un albero un insieme di regole di classificazione
- Caratteristiche:
  - Veloci rispetto agli altri metodi
    - Facili da interpretare tramite regole di classificazione (una per ogni cammino dell'albero)
    - Possono essere facilmente convertiti in interrogazioni SQL per interrogare la base di dati

Esempio

ETA`	TIPO AUTO	RISCHIO
40	familiare	basso
65	sportiva	alto
20	utilitaria	alto
25	sportiva	alto
50	Familiare	basso
48	utilitaria	alto



Costruzione albero

- Due fasi:
  - *fase di build*: si costruisce l'albero iniziale, partizionando ripetutamente il training set sul valore di un attributo, fino a quando tutti gli esempi in ogni partizione appartengono ad una sola classe
  - *fase di pruning*: si pota l'albero, eliminando rami dovuti a rumore o fluttuazioni statistiche
    - Esempio: si estraggono campioni multipli dal training set e si costruiscono alberi indipendenti
    - non approfondiamo questo aspetto

Fase di build

Builtree(training set T)

{Partition(T)}

Partition(Data set S)

{

if (tutti i punti in S sono nella stessa classe) then

return

foreach attributo A do

valuta gli splits su A (usando un algoritmo di split)

usa split ”migliore” per partizionare S in S1 e S2

Partition(S1)

Partition(S2)

}

Algoritmi di split

- Dato un attributo A, determinano il miglior predicato di split per un attributo:
- due problemi:
  - scelta predicato
  - determinazione bontà
- scelta predicato:
  - dipende dal tipo dell'attributo
- ottimalità:
  - un buon predicato permette di ottenere:
    - meno regole (meno split, nodi con fan-out più basso)
    - supporto e confidenza alte

Predicati di Split

- Gli split possono essere
  - binari
  - multipli
- per attributi numerici
  - split binario: $A <= v$, $A > v$
  - split multiplo: $A <= v1$, $v1 < A <= v2$, …, $vn-1 < A <= vn$
- per attributi categorici, se il dominio di A è S:
  - split binario: $A \in S'$, $A \in S - S'$ con $S' \subset S$
  - split multiplo: $A \in S1$, …, $A \in Sn$

$$\text{con } S1 \cup \ldots \cup Sn = S,\ Si \cap Sj = \{\}$$

Indici di splitting

- Valutano la bontà di split alternativi per un attributo
- Diverse tipologie di indice
- Indice Gini:
  - dataset T con esempi di n classi
  - $gini(T) = 1 - \sum_i p_i^2$
  - con p_i frequenza class i in T
- Se T viene suddiviso in T1 con esempi appartenenti a n1 classi e T2 con esempi appartenenti a n2 classi:
  - $gini_{split}(T) = (n1/n)\ gini(T1) + (n2/n)\ gini(T2)$
- split con indici più bassi sono giudicati migliori

Esempio

	ETA`	TIPO AUTO	CLASSE RISCHIO
t1	40	familiare	basso
t2	65	sportiva	alto
t3	20	utilitaria	alto
t4	25	sportiva	alto
t5	50	familiare	basso

- due classi, n = 2
- Si consideri lo split rispetto a Età <= 25
- T1= {t3, t4}, T2={t1, t2, t5}
- gini_split(T) = 1/2 (1 - 1) + 2/2 (1 - (1/9 +4/9)) = 4/9

Estensioni

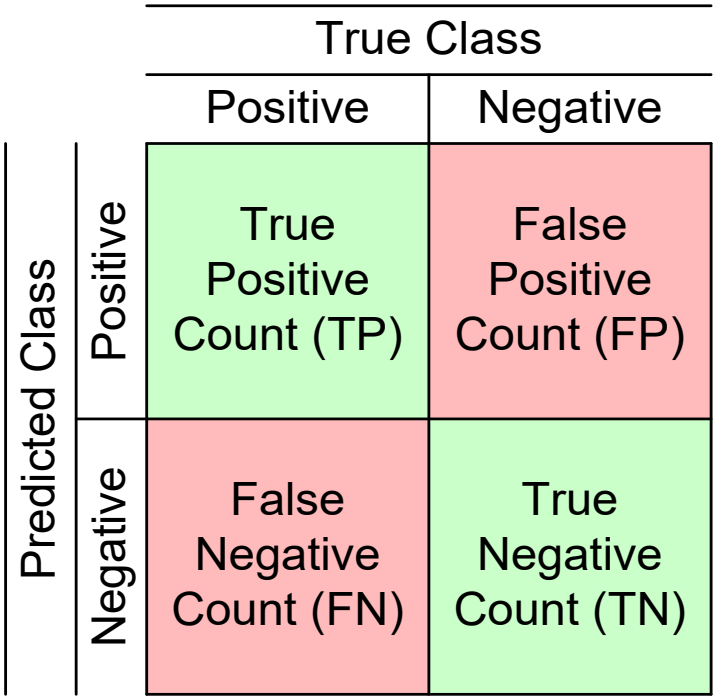
- Split su attributi multipli
- gestione training set in memoria secondaria

Metriche per la Valutazione di Classificatori

- Accuratezza della Classificazione
  - Hit rate
- Velocità di classificazione
  - Costruzione del modello di classificazione;
  - Classificazione di nuovi oggetti o casi
- Robustezza:
  - capacità del modello predittivo di effettuare classificazioni abbastanza accurate anche in presenza di valori mancanti o errori nei dati
- Scalabilità
  - capacità di costruire un modello predittivo in modo efficiente a partire da dati molto voluminosi
- Facilità di interpretazione dei risultati
  - Trasparenza
  - Facilità di spiegazione

Accuratezza di Modelli di Classificazione

- Nei problemi di classificazione, la fonte primaria per la stima dell'accuratezza è la confusion matrix



$$Accuratezza = \frac{TP + TN}{TP + TN + FP + FN}$$

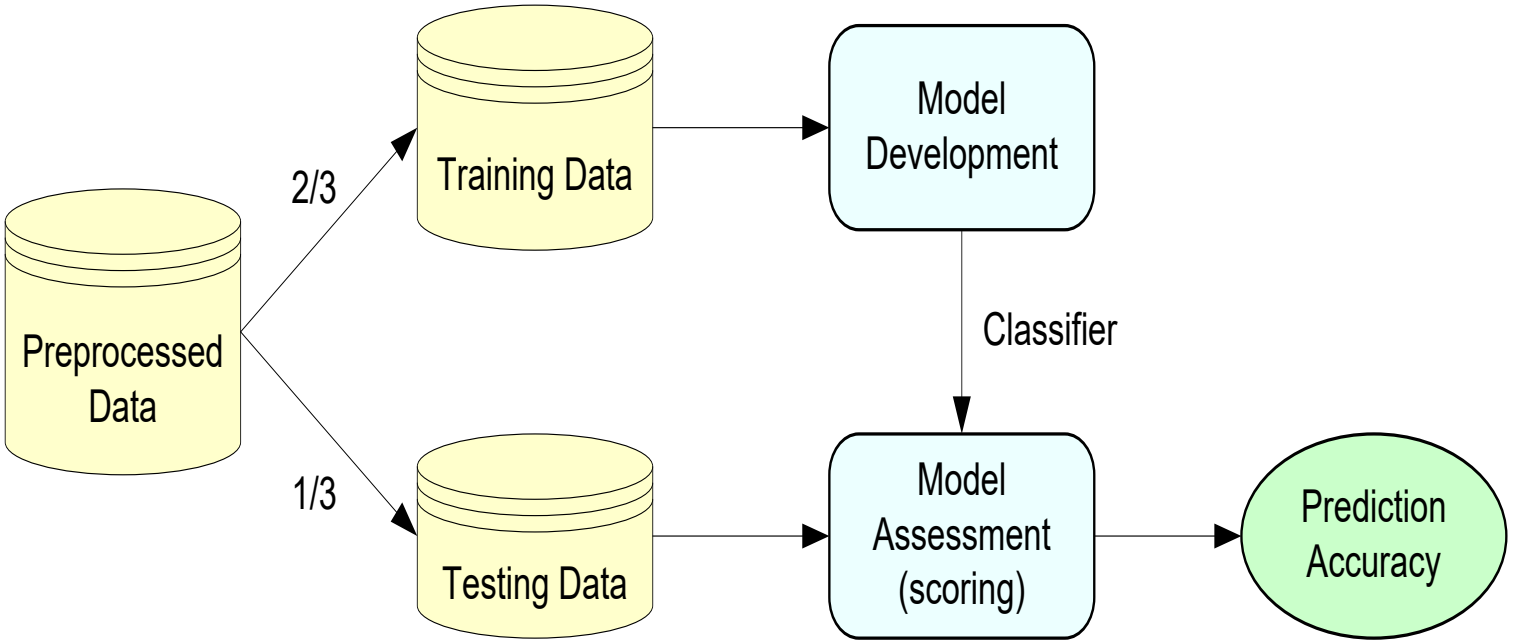
$$Tasso\ True\ Positive = \frac{TP}{TP + FN}$$

$$Tasso\ True\ Negative = \frac{TN}{TN + FP}$$

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

Split di Training e Test Set

- Split Semplice
  - Suddividere i dati in 2 due insiemi mutuamente esclusivi, di cui ~70% training e ~30% testing



Clustering

- Dati del problema:
  - base di dati di oggetti
- Problema:
  - trovare una suddivisione degli oggetti in gruppi (cluster) in modo che:
    - gli oggetti in un gruppo siano molto simili tra di loro
    - oggetti in gruppi diversi siano molto diversi
  - i gruppi possono essere anche sovrapposti o organizzati gerarchicamente

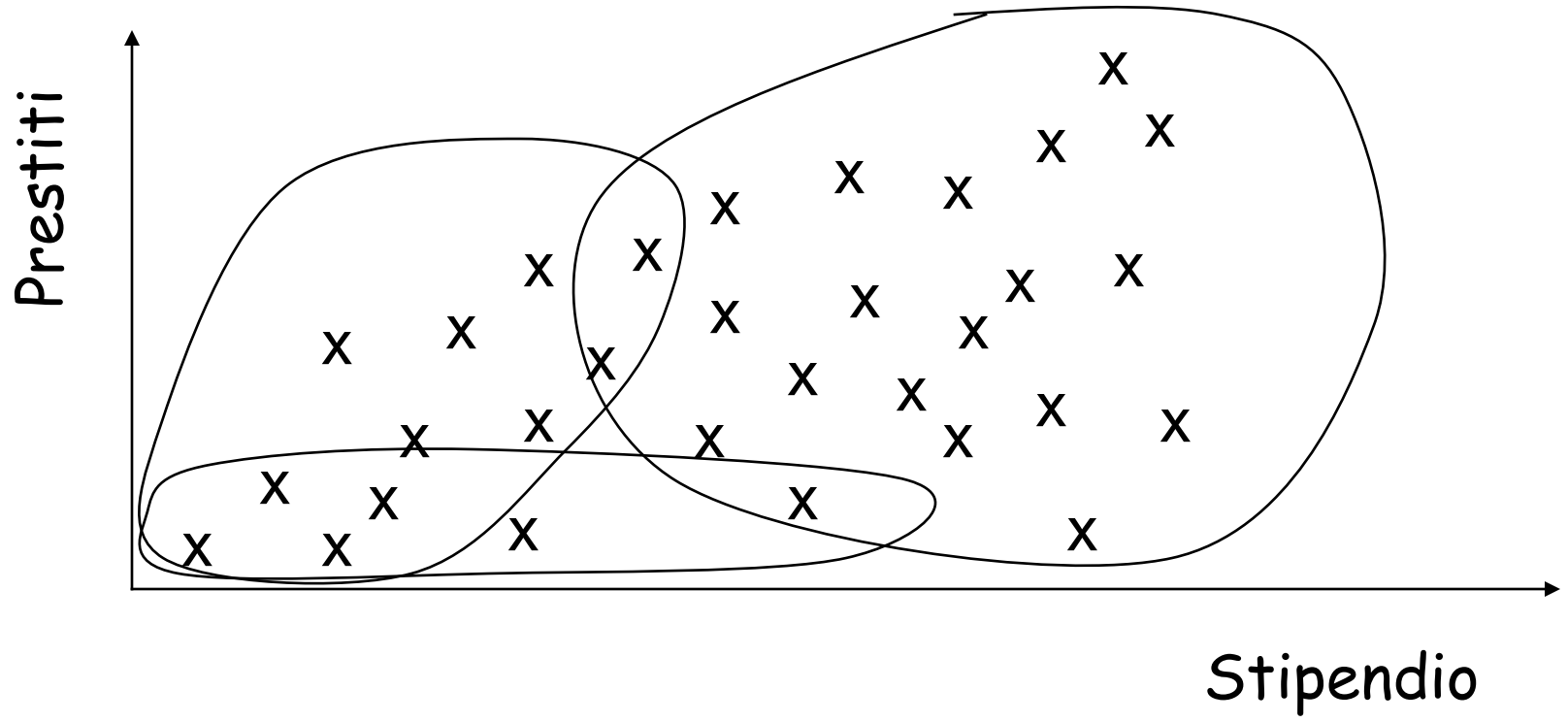
Applicazioni

- Identificazione di popolazioni omogenee di clienti in basi di dati di marketing
- valutazione dei risultati di esperimenti clinici
- monitoraggio dell'attivita` di aziende concorrenti

Settori di sviluppo

- Statistica
- machine learning
- database spaziali
- data mining
- molti algoritmi assumono che i dati risiedano in memoria
- poco scalabili

Esempio



Cluster Analysis per il Data Mining

- Metodi di Analisi
  - Metodi Statistici, quali il *k*-means, il *k*-modes, etc
  - Reti Neurali
  - Logica Fuzzy (e.g., fuzzy c-means algorithm)
  - Algoritmi Genetici

Approccio

- La similitudine tra due oggetti si ottiene applicando una funzione di distanza
- la nozione di distanza dipende dagli oggetti considerati e dal tipo di applicazione
- Due tipi di algoritmi:
  - algoritmi di partizionamento:
    - dato il numero di cluster k, partizionare i dati in k cluster in modo che certi criteri di ottimalità siano verificati
  - algoritmi gerarchici:
    - si parte da una partizione in cui ogni cluster è composto da un singolo oggetto
    - le partizioni vengono combinate, in modo da mettere insieme gli oggetti più simili
    - alternativamente, si parte da un singolo cluster contenente tutti i dati e lo si partiziona fino a raggiungere il numero k di cluster

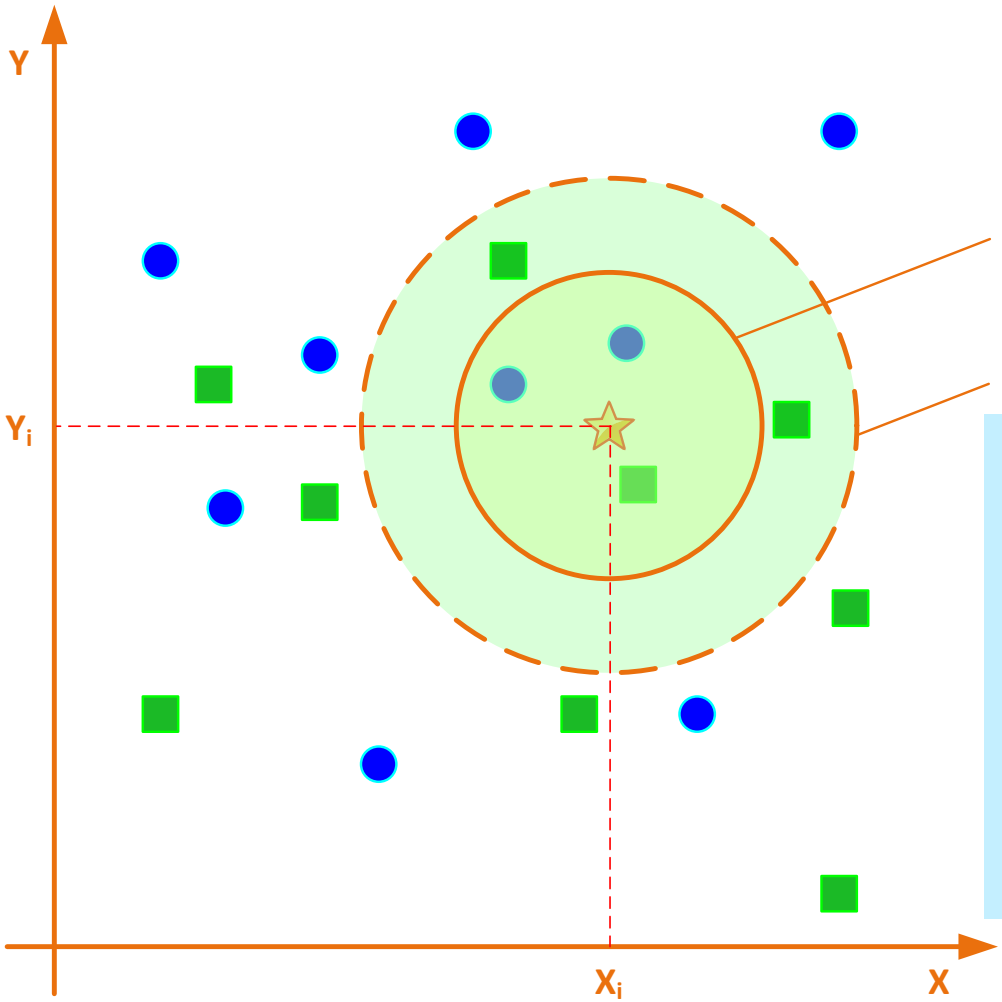
Metodo k-Nearest Neighbor (k-NN)

- E' uno dei più semplici algoritmi di machine learning
- Un'alternativa alle reti neurali ed all'algoritmo di machine leaning Support Vector Machine (SVM), computazionalmente troppo complessi
- *k*-NN è un metodo predittivo utilizzabile sia per la classificazione che per la regressione
- E' un approccio di learning di tipo instance-based (o lazy learning) – gran parte del lavoro è svolto a tempo di classificazione anziché di modellazione
- *k* : il numero di vicini usati per la classificazione

Funzionamento *di* k-NN

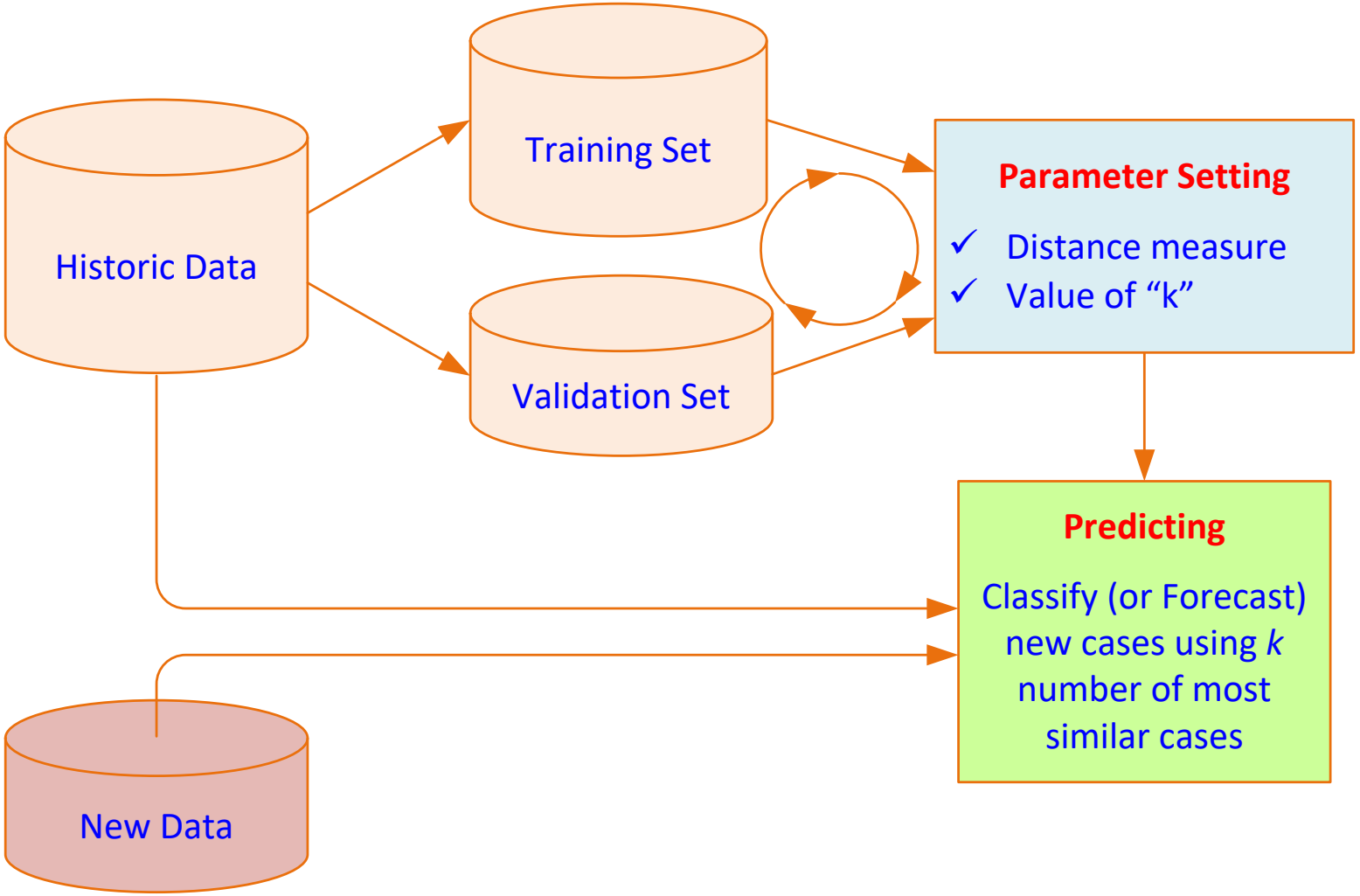
- Ogni nuovo oggetto viene classificato usando la maggioranza dei suoi *k* oggetti più vicini
- L'oggetto viene quindi assegnato alla classe più ricorrente tra i *k* oggetti più vicini
- Se k=1 l'oggetto viene semplicemente inserito nella classe dell'oggetto più vicino
- Oltre al parametro *k* occorre selezionare una funzione di distanza

Esempio di applicazione di *k-NN*



- Occorre classificare i punti in *tondi* e *quadrati*
- La stella rappresenta un nuovo punto da classificare
- La risposta dipende dal valore di *k:* se è 3 il punto è classificato come *tondo,* come *quadrato* se è 5

Il Processo del Metodo *k*-NN



Funzioni di Distanza o Similarità

Minkowski distance

$$d(i,j) = \sqrt[q]{(x_{i1} - x_{j1})^q + (x_{i1} - x_{j1})^q + \cdots + (x_{ip} - x_{jp})^q}$$

If $q=1$, then d is called Manhattan distance

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}|$$

If $q=2$, then d is called Euclidean distance

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i1} - x_{j1})^2 + \cdots + (x_{ip} - x_{jp})^2}$$

- Dove *i* e *j* sono due oggetti nello spazio *n*-dimensionale
- Funzionano per attributi numerici non per i categorici

Distanza di Jaccard

- Il coefficiente di Jaccard misura la similarità tra insiemi campionari, ed è definito come la dimensione della intersezione divisa per la dimensione dell'unione degli insiemi campionari:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

- La distanza di Jaccard, che misura la dissimilarità tra insiemi campionari, è complementare al coefficiente di Jaccard e si ottiene sottraendo il coefficiente di Jaccard da 1.

Scelta del Parametro *k*

- Il miglior valore dipende dai dati in input
- Valori più grandi di *k* riducono gli effetti distorsivi (errori, valori mancanti) nei dati, ma rendono anche meno marcati i confini tra le classi
- Un valore ottimale viene spesso trovato tramite euristiche
- La Cross Validation è una delle tecniche più frequentemente utilizzate per trovare il valore ottimale per k e per la misura di distanza

Cross Validation

- Tecnica sperimentale per inferire valori ottimali per parametri di modelli predittivi
- Ad esempio, per trovare il valore ottimale di *k* nel *k*-NN o valutare la bontà di un valore di *k*
  - si sceglie un numero *v* in modo casuale e, per ogni valore potenziale di *k,* si classifica, a turno, uno dei *v* insiemi usando gli altri *v-1* come training set
  - per ognuno di tali *v* esperimenti si calcola l'errore quadratico metrico, aggregando tali valori
  - si sceglie il valore di *k che* minimizza l'errore totale

Algoritmo di Clustering *k-Means*

- k : numero pre-determinato di clusters
- Occorre anche qui una funzione di distanza
- Algoritmo (Step 0: determina valore di k)

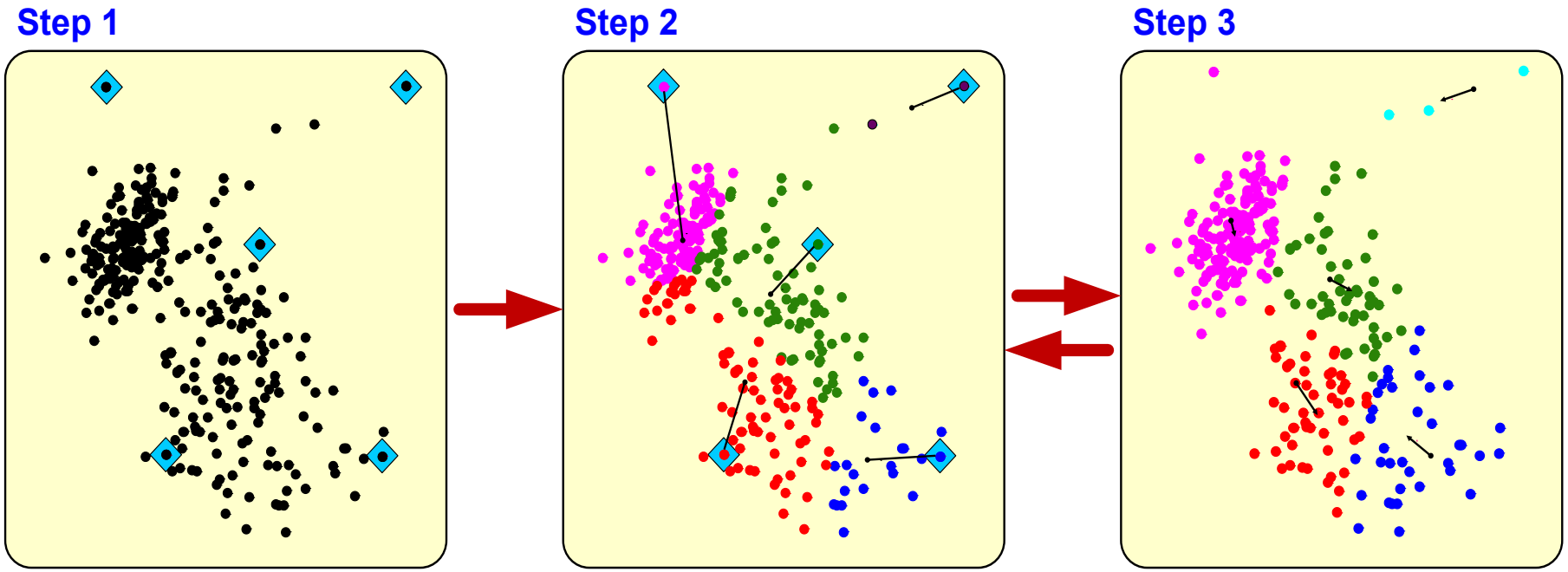
  Step 1: Seleziona in modo casuale k punti come centri iniziali dei cluster.

  Step 2: Assegna ciascun nuovo punto al cluster il cui centro è più vicino.

  Step 3: Ricalcola i centri dei cluster aggiornati

  Step di ripetizione: Ripetere gli step 2 e 3 finchè si raggiungerà un dato criterio di convergenza (sostanzialmente, l'assegnamento di punti che rende stabili le classi).

Esempio di Clustering con *k*-Means



Similarity search

- Dati del problema:
  - base di dati di sequenze temporali
- Problema:determinare
  - sequenze simili ad una sequenza data
  - tutte le coppie di sequenze simili

Applicazioni

- Identificazione delle società con comportamento simile di crescita
- determinazione di prodotti con profilo simile di vendita
- identificazione di azioni con andamento simile
- individuazione di porzioni di onde sismiche non simili per determinare irregolarità geologiche

Settori di sviluppo

- Database temporali
- speech recognition techniques
- database spaziali

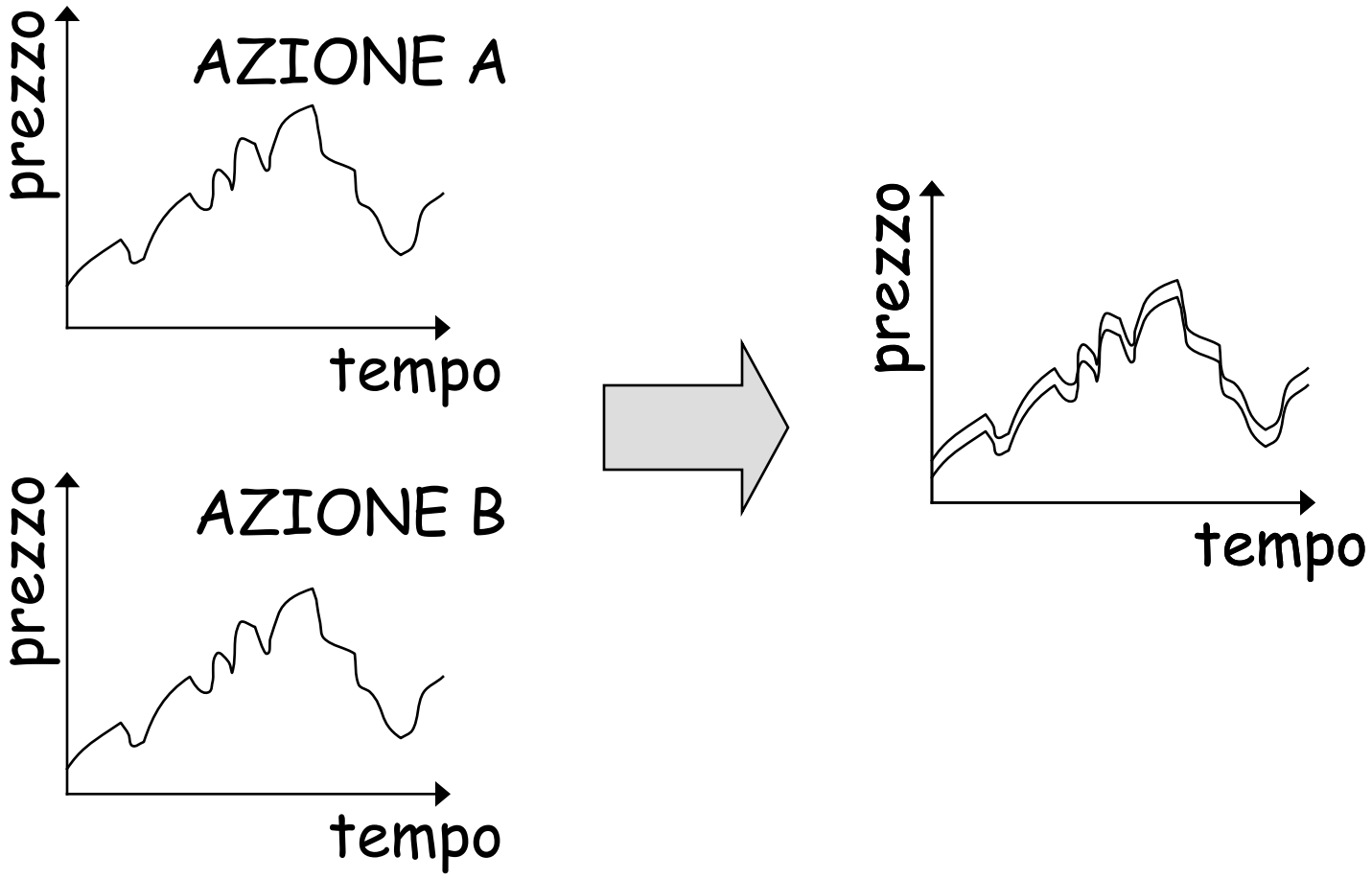
Tecniche

- Due tipi di interrogazione
  - match completo: la sequenza cercata e le sequenze della base di dati hanno la stessa lunghezza
  - match parziale: la sequenza cercata puo`essere sottosequenza di quelle recuperate dalla base di dati
- Possibilita`di traslazioni, variazioni di scala
- Diverse misure con cui confrontare le sequenze
  - Esempio: misura euclidea

    X = <x1,…,xn>

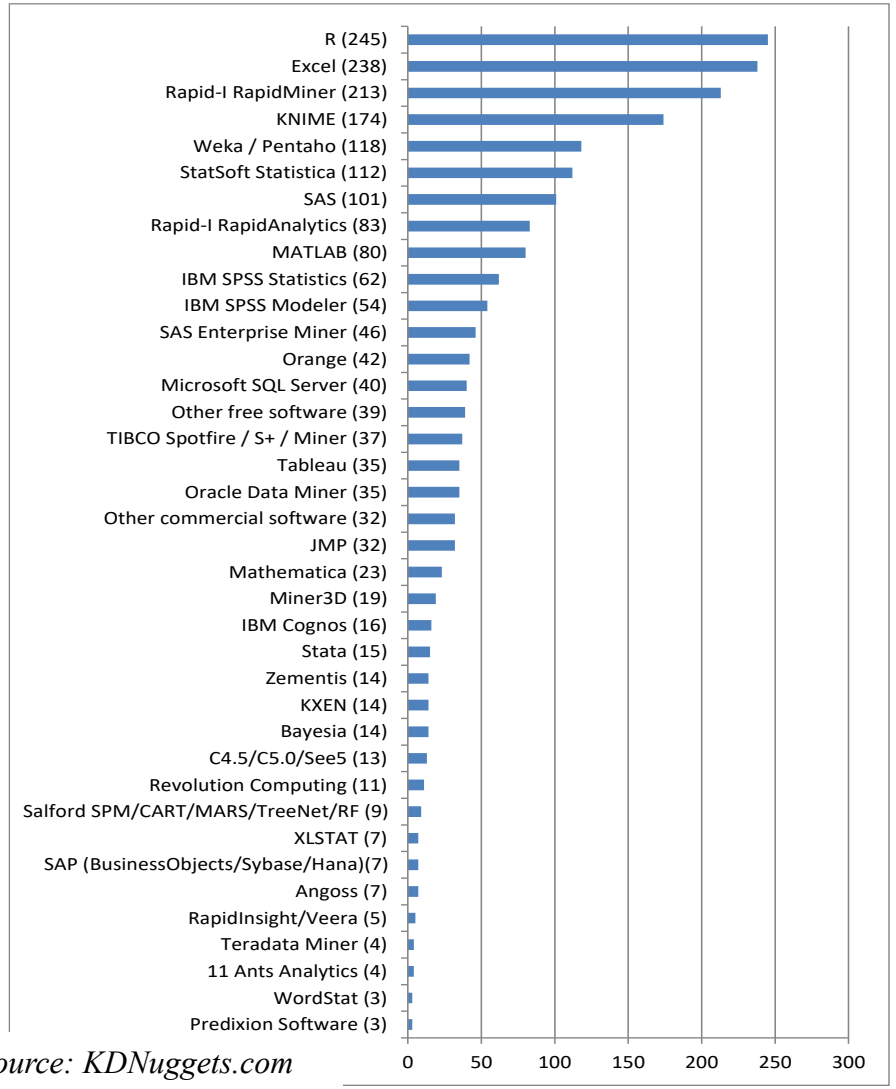
    Y = <y1,…,yn> $|| X - Y || = \sum (xi - yi)^2$

Esempio



Software per il Data Mining

- Commerciali
  - IBM SPSS Modeler (formerly Clementine)
  - SAS - Enterprise Miner
  - IBM - Intelligent Miner
  - StatSoft – Statistica Data Miner
  - … e molti altri
- Free e/o Open Source
  - R
  - RapidMiner
  - Weka…



Source: KDNuggets.com

Piattaforme Software per Big Data