TDM

Sommario

Appunti del corso di Tecniche di Data Mining. A.A. 2007/2008.

Indice

1 Introduzione al Data Mining **3**

2 Il processo di estrazione della conoscenza **5**
2.1 Consolidamento dei dati 6
2.2 Selezione e Preprocessing 6
2.3 Data Mining 6
2.4 Interpretazione e valutazione 7

3 I Dati: tipi, qualità e preparazione dei dati **8**
3.1 Visualizzazione dei dati 9
3.2 Data Cleaning 10
3.3 Data Reduction 10
3.4 Data Transformation 10
3.5 Discretizzazione 12
3.6 Misure di distanza definite sui dati 13

4 Esplorazione dei dati **15**
4.1 Summary Statistics 15
4.2 Visualizzazione 15
4.3 OLAP 17

5 Clustering **18**
5.1 K-Means 19
5.2 Clustering Gerarchico 21
5.3 Clustering Density-based 24
5.4 Validazione del Clustering 24

6 Classificazione **26**
6.1 Classificatori basati su Alberi di Decisione (C4.5 ed altri) . . 26
6.1.1 Gini Index 27
6.1.2 Entropia 27
6.1.3 Errore di Classificazione 28
6.2 Problemi legati alla Classificazione 28
6.3 Classificatori Rule-based 30
6.4 Classificatori Nearest-Neighbor (K-NN) 31

7 Mining di Regole Associative **33**
7.1 Algoritmo Apriori 34
7.2 Algoritmo FP-Growth 36
7.3 Generazione di regole associative 36
7.4 Regole Associative per attributi non binari 37
7.4.1 Discretizzazione 37
7.4.2 Metodi Statistici 38
7.4.3 Metodo Min-Apriori 38
7.5 Regole Associative Multilivello 39
7.6 Analisi Associative su Dati Sequenziali 40
7.6.1 General Sequential Pattern (GSP) 41
7.6.2 Aggiunta di Vincoli Temporali 42

8 Tutela della Privacy **44**
8.1 K-Anonymity 44

1 Introduzione al Data Mining

Motivazioni del Data Mining:

- Enormi quantità di dati, provenienti da innumerevoli fonti, (spesso) a costi bassissimi.
- Possibilità di far emergere dai dati informazioni "nascoste", tramite tecniche non applicabili in maniera non automatica.

Definizione di Data Mining:

- Estrazione dai dati di informazioni non banali, precedentemente sconosciute.
- Esplorazione ed analisi di grandi quantità di dati attraverso tecniche automatiche o semi-automatiche, con il fine di individuare pattern significativi.

Il punto focale sta nella capacità di far emergere informazioni (proprietà, relazioni, ecc. ecc.) di cui non si conosce a priori l'esistenza.

Tipologie di attività di Data Mining:

- Metodi Predittivi: uso di variabili note sui dati per prevedere i valori di altre variabili il cui valore non è conosciuto.
- Metodi Descrittivi: ricerca di pattern, comprensibili dall'uomo, che descrivano relazioni esistenti fra i dati.

Branche principali del Data Mining:

- **Classificazione** (predittivo): dato un insieme di record (training set) in cui ogni record contiene un attributo che ne descrive la classe di appartenenza, si cerca un modello che permetta di "calolare" l'attributo di classe come funzione degli altri attributi. L'obiettivo finale è quello di poter usare il modello ottenuto per predire la classe di appartenenza per i record "sconosciuti".
- **Clustering** (descrittivo): dato un insieme di dati puntuali, ognuno con un insieme di attributi, ed una misura di similarità fra essi, si cerca di raggruppare i dati in cluster di individui simili.
- **Discovery di Regole Associative** (descrittivo): dato un insieme di record, ognuno dei quali contiene un certo numero di oggetti da una determinata collezione, si cerca di ottenere delle regole di dipendenza che consentano di prevedere l'occorrenza di un oggetto basandosi sull'occorrenza di altri oggetti.

- **Discovery di Pattern Sequenziali** (descrittivo): dato un insieme di oggetti, ognuno associato ad una propria timeline di eventi, si cercano regole che permettano di prevedere dipendenze sequenziali fra eventi distinti.
- **Regressione** (predittivo): predizione del valore di una data variabile continua sulla base dei valori di altre variabili, assumendo un modello di dipendenza lineare o non lineare.
- **Deviation Detection** (predittivo): individuazione di deviazioni significative rispetto ad un normale modello di "comportamento".

Attuali sfide del Data Mining:

- Scalabilità.
- Dimensionalità.
- Dati complessi ed eterogenei.
- Qualità dei dati.
- Proprietà e distribuzione dei dati.
- Tutela della privacy.
- Dati in streaming.

2 Il processo di estrazione della conoscenza

Knowledge Discovery in Databases (KDD):

> Processo di selezione e manipolazione dei dati per identificare pattern nuovi, accurati ed utili per modellare fenomeni del mondo reale

Il Data Mining è il componente principale del KDD. Come si vede dalla

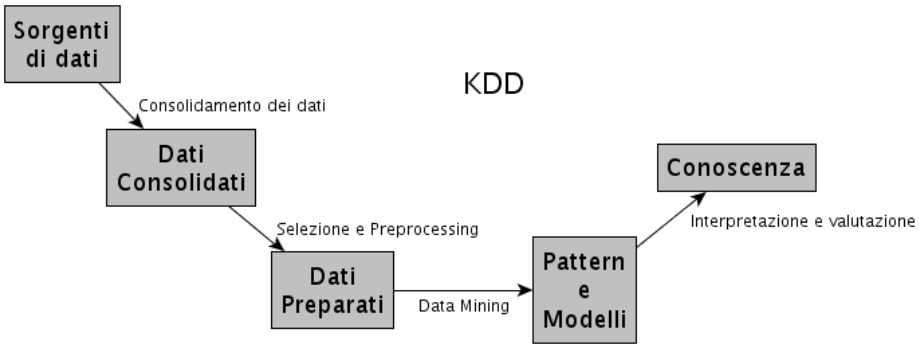

Figura 1: Knowledge Discovery in Databases

Figura 1, il KDD è un processo incrementale che prevede:

- Studio e comprensione del dominio.
- Consolidamento dei dati, ovvero creazione di un dataset obiettivo.
- Selezione e preprocessing:
  - Pulizia dei dati.
  - Riduzione e proiezione dei dati.
- Scelta delle funzioni di datamining (sommarizzazione, classificazione, clustering, ecc. ecc.).
- Scelta degli algoritmi di mining.
- Data Mining: ricerca di pattern interessanti.
- Interpretazione e valutazione, ovvero analisi dei risultati.
- Utilizzo delle conoscenze acquisite.

Nel seguito vediamo le fasi più nel dettaglio.

2.1 Consolidamento dei dati

In questa fase si cerca di migliorare la qualità dei dati, cosa che influenza direttamente la qualità dei risultati raggiungibili.
In particolare:

- Determinare la lista di attributi da usare.
- Consolidare i dati, anche provenienti da fonti diverse, in un unico "database di lavoro".
- Eliminare o stimare valori mancanti.
- Eliminare outliers[1].
- Effettuare le prime stime su ciò che ci si aspetta.

2.2 Selezione e Preprocessing

Azioni principali:

- Generare l'insieme di esempi: scelta del metodo di sampling, considerazioni sulla complessità ed il volume dei dati.
- Ridurre la dimensionalità: rimozione di attributi ridondanti, combinazione di attributi.
- Riduzione dei range di valori: raggruppamento di valori simbolici, quantificazione dei valori numerici continui.
- Trasformazione dei dati: de-correlare e normalizzare valori, modificare la rappresentazione.
- Massiccio uso di Online Analytical Processing (OLAP).

2.3 Data Mining

Varie fasi del Data Mining:

- Business Understanding: comprensione degli obiettivi del progetto, in modo da poterli convertire nella formulazione di un problema di mining.
- Data Understanding: caratterizzazione dei dati del problema, assestamento e verifica dei dati.

[1] Gli *outliers* sono eccezioni palesi, record i cui valori si discostano eccessivamente dagli altri.

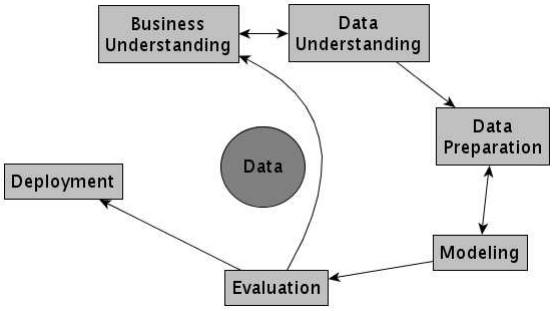

Figura 2: Data Mining

- Modeling: selezione ed applicazione di varie tecniche di modellazione fino ad ottenere, calibrando i parametri, un modello ottimale. Spesso per ottenere il modello ottimale è necessario alternare più volte le fasi di *Preparazione dei Dati* e *Modellazione*.
- Valutazione: verifica della rispondenza fra il modello ottimale raggiunto e gli obiettivi di business prefissati.
- Deployment: la conoscenza ottenuta viene organizzata e presentata in modo da essere compresa ed utilizzata dal cliente.

2.4 Interpretazione e valutazione

Valutazione:

- Validazione statistica e test di significatività.
- Esami qualitativi da parte di esperti del dominio.
- Indagini per verificare L'accuratezza del modello

Interpretazione:

- Alberi di decisione e modelli a regole possono essere letti direttamente.
- I risultati del clustering possono essere rappresentati su grafici o tabelle.
- In alcuni casi può essere generato automaticamente del codice.

3 I Dati: tipi, qualità e preparazione dei dati

La fase di Data Preprocessing affronta due problemi tipici:

- Troppi dati: dati sbagliati, "rumorosi" o irrilevanti, dati di dimensione computazionalmente intrattabile o di tipologie eccessivamente eterogenee.
- Pochi dati: attributi o valori mancanti. dimensioni insufficienti.

In generale il processo prevede l'esplorazione dei dati ed il loro arricchimento attraverso l'applicazione di tecniche specifiche, secondo direzioni che spesso dipendono direttamente dall'obiettivo.

La preparazione dei dati lavora su oggetti del dominio a cui sono associate proprietà rappresentate da variabili. E' facilmente intuibile che la fase di Data Preprocessing prevede quindi l'analisi e ma manipolazione di dati in forma di tabelle.

Tipi di dati:

- Discreti:
  - Nominali: identificatori univoci.
  - Ordinali: è definito un ordinamento fra i valori.
  - Binari: due soli valori.
- Continui:
  - Interval-based: scalabili di fattore costante.
  - Ratio-based: scalabili linearmente.

Caratteristiche dei dati:

- Sparsità: un attributo è sparso se contiene molti *null*.
- Monotonicità: crescita continua dei valori di una variabile.
- Outliers: valori singoli con una frequenza molto bassa.
- Dimensionalità delle variabili: numero di valori che una variabile può assumere.
- Dimensionalità degli oggetti: numero di attributi che un oggetto ha.
- Anacronismo: valori in una sola porzione dei dati.

Come descrivere i dati:

- Attraverso grafici:
  - Distribuzione delle frequenze.
  - Correlazione.
  - Dispersione.
- Attraverso misure:
  - Media, mediana, quartili.
  - Varianza, deviazione standard.
  - Forma, simmetria, curtosi.

Misure descrittive dei dati:

- Tendenza centrale o posizione: media aritmetica, geometrica o armonica, mediana, quartili, percentili, moda.
- Dispersione o variabilità: range, scarto medio, varianza, deviazione standard.
- Forma della distribuzione: simmetria (medie interquartili, momenti centrali, indice di Fisher) e curtosi (indice di Pearson, coefficiente di curtosi)

3.1 Visualizzazione dei dati

Visualizzazione di dati qualitativi:

- Rappresentazione delle frequenze:
  - Diagrammi a barre.
  - Ortogrammi.
  - Aerogrammi.
- Correlazione:
  - Web Diagrams
- Ciclicità:
  - Diagrammi polari.

Visualizzazione di dati quantitativi:

- Istogrammi.

- Poligoni.
- Stem and leaf.
- Dot diagrams.
- Diagrammi dei quantili (Boxplot)

3.2 Data Cleaning

La "pulizia" dei dati prevede:

- Trattamento di valori anomali: mancanti, sconosciuti (privi di significato), non validi.
- Trattamento di outliers.
- Trattamento di tipi impropri.

3.3 Data Reduction

Due tipi di riduzione:

- Verticale: riduzione del numero di tuple (sampling e clustering).
- Orizzontale: riduzione del numero di colonne, attraverso:
  - Selezione di un sotto-insieme di attributi: manuale o automatica (rispetto a qualche misura statistica di significatività).
  - Creazione di nuovi attributi per rappresentare le tuple: *Principal Componet Analysis* (trova la combinazione degli attributi nei k vettori ortonormali più significativa e proietta le vecchie tuple sui nuovi attributi).

3.4 Data Transformation

Si trasformano i dati per risolvere i problemi legati alla presenza di dati con errori o incompleti e dati mal distribuiti (forte asimmetria, molti picchi, ecc. ecc.).

Obiettivo:

Vogliamo definire una trasformazione T sull'attributo X

$$Y = T(X)$$

tale che:

- Y preserva l'informazione "rilevante" di X.

- Y elimina almeno uno dei problemi di X.
- Y è più "utile" di X.

Le linee guida per raggiungere l'obiettivo sono le seguenti:

- Scopi principali:
  - Stabilizzare la variabile.
  - Normalizzare le distribuzioni.
  - Linearizzare le relazioni tra variabili.
- Scopi secondari:
  - Semplificare l'elaborazione di dati poco "piacevoli".
  - Rappresentare i dati nella scala più adatta.

Ecco le trasformazioni utilizzate più di frequente:

- Trasformazioni esponenziali:

$$T_p \begin{cases} ax^p + b \ \ (p \neq 0) \\ c \ log \ x + d \ \ (p = 0) \end{cases}$$

  Caratteristiche:
  - Preservano l'ordine.
  - Preservano alcune statistiche di base.
  - Sono continue.
  - Ammettono derivate.
  - Sono specificate tramite funzioni semplici.
- Trasformazioni lineari: $ax + b$
- Normalizzazione min-max:

$$v' = \frac{v - min_A}{max_A - min_A}(newmax_A - newmin_A) + newmin_A$$

- Normalizzazione z-score:

$$v' = \frac{v - mean_A}{stand\_dev_A}$$

- Normalizzazione tramite decimal scaling:

$$v' = \frac{v}{10^j}$$

  dove j è il più piccolo intero tale che $Max(\mid v' \mid) < 1$

- Trasformazione logaritmica (stabilizzazione della varianza):

$$T(x) = c\ log\ x\ +\ d$$

- Trasformazione in radice (stabilizzazione della varianza):

$$T(x) = a\sqrt[c]{x}\ +\ b$$

  Utile per omogeneizzare varianze di distribuzioni particolare (ad esempio di Poisson).

- Trasformazione reciproca (stabilizzazione della varianza):

$$T(x) = \frac{a}{x^p} + b$$

3.5 Discretizzazione

La discretizzzazione consente di semplificare l'interpretazione e "normalizzare" i valori, risolvendo problemi derivanti dall'avere dati continui estremamente sparsi.

Discretizzazione supervisionata:

- Non etichetta le istanze.
- Il numero di classi è noto.
- Varie tecniche:
  - Natural binning: intervalli di uguale ampiezza.
  - Equal frequency binning: intervalli di identica frequenza.
  - Statistical binning: uso di informazioni statistiche (media, varianza, quartili).

Discretizzazione non supervisionata:

- La discretizzazione ha un obiettivo quantificabile.
- Il numero di classi non è noto a priori.
- Varie tecniche:
  - ChiMerge: iterativamente si uniscono valori simili.
  - Basata su entropia.
  - Basata su percentili.

3.6 Misure di distanza definite sui dati

Abbiamo visto che in diversi casi ricorre il concetto di distanza o similarità fra dati.

Definiamo:

- **Distanza euclidea**:

$$dist = \sqrt{\sum_{k=1}^{n}(p_k - q_k)^2}$$

- **Distanza di Minkowski**:

$$dist = (\sum_{k=1}^{n} \mid p_k - q_k \mid^r)^{\frac{1}{r}}$$

  Per cui:

  - $r = 1$: Manhattan distance.
  - $r = 2$: distanza euclidea.
  - $r = 3$: "supremum" distance.

Proprietà della distanza $d(p, q)$:

- Definita positiva: $d(p, q) \geqslant 0$ e $d(p, q) = 0$ se e solo se $p = q$.
- Simmetrica: $d(p, q) = d(q, p)$
- Disuguaglianza triangolare: $d(p, r) \leqslant d(p, q) + d(q, r)$

Una distanza che soddisfi queste proprietà è anche detta **metrica**.

Osservazione:

> All'aumentare della dimensionalità i dati diventano più sparsi ed i concetti di densità e distanza perdono di significato.

Proprietà delle misure di similarità:

- $s(p, q) = 1$ se e solo se $p = q$.
- $s(p, q) = s(q, p)$

Misure comuni di similarità per vettori binari:

- SMC: numero di match / numero di attributi

$$(M_{11} + M_{00}/M_{11} + M_{10} + M_{01} + M_{00})$$

- Jaccard: numero di match 11 / numero di "non entrambi zero"

$$(M_{11} + M_{00}/M_{11} + M_{10} + M_{01})$$

Per calcolare la similarità fra due oggetti con attributi non binari possiamo usare le seguenti misure di similarità:

- Coseno:

$$\frac{p \bullet q}{\mid p \mid\mid q \mid}$$

  dove $p \bullet q$ indica il prodotto vettore-vettore.

- Correlazione:

$$p' \bullet q'$$

  dove

$$r'_k = \frac{r_k - mean(r)}{std(r)}$$

4 Esplorazione dei dati

L'esplorazione dei dati è una fase preliminare che ha lo scopo di far comprendere meglio le caratteristiche dei dati, aiutando nella scelta degli strumenti per il preprocessing e l'analisi, attraverso l'uso della capacità innata nell'uomo di riconoscere pattern.

Ci concentriamo su tre tecniche ed approcci principali per l'esplorazione dei dati:

- Statistiche riassuntive (Summary Statistics).
- Visualizzazione.
- Online Analytical Processing (OLAP).

4.1 Summary Statistics

Sono numeri che riassumono le proprietà dei dati.
Le proprietà descritte includono: frequenza, posizione, diffusione.

Molte delle statistiche riassuntive possono essere calcolate con un'unica passata sui dati.

Misure usate comunemente:

- Frequenza.
- Moda.
- Percentili.
- Media.
- Mediana.
- Range.
- Varianza: $s_x^2 = \frac{1}{m-1} \sum_{i=1}^{m} (x_i - \overline{x})^2$
- Altre misure, simili alla varianza ma meno influenzate dagli outliers.

4.2 Visualizzazione

Consiste nella conversione dei dati in un formato grafico o tabellare in modo che le caratteristiche dei dati e le relazioni fra essi possano facilmente essere individuate o analizzate.

In questo caso si sfrutta la capacità umana di riconoscere pattern o individuare outliers.

Attività che caratterizzano la fase di visualizzazione:

- Rappresentazione: gli oggetti, i loro attributi e le relazioni fra essi vengono tradotti in elementi grafici (punti, linee, forme, colori, ecc. ecc.).
- Riordinamento (Arrangement): si determina e si modifica il posizionamento degli elementi nella loro visualizzazione.
- Selezione: eliminazione o de-enfatizzazione di alcuni oggetti o attributi, in modo da poter concentrare l'attenzione su sottoinsiemi più facilmente esaminabili.

Varie tecniche di visualizzazione:

- Istogrammi: mostrano la distribuzione di una variabile suddividendo lo spazio (dei valori) in intervalli.
- Istogrammi bidimensionali: mostrano la distribuzione congiunta di due attributi.
- Box Plots: mostrano la distribuzione rispetto ai quartili ed evidenziano la presenza di eventuali outliers.
- Scatter Plots: bidimensionali (tridimensionali) mostrano coppie (triple) di attributi come punti nello spazio. Altri attributi aggiuntivi possono essere visualizzati associandoli a forma, dimensione, colore dell'oggetto usato per la rappresentazione.
- Contour Plot: utili per rappresentare attributi continui misurati su una griglia spaziale (es. mappa dell'elevazione, delle temperature, ecc. ecc.)
- Matrix Plot: visualizzano matrici di dati, associando un colore ai vari valori (che spesso devono essere normalizzati)
- Coordinate Parallele: ogni oggetto è rappresentato da una linea spezzata, i cui vertici corrispondono ai vari attributi.
- Start Plot: simile alle coordinate parallele, ma con gli assi che partono da un centro comune in modo da formare poligoni e non linee.
- Chernoff Faces: si disegnano faccine in cui ogni "tratto somatico" è legato al valore di un attributo.

4.3 OLAP

Poiché i database relazionali mettono i dati in tabelle, OLAP si basa sull'utilizzo di array multidimensionali per la visualizzazione.

I passi fondamentali:

- Identificare quali attributi devono essere le dimensioni (attributi necessariamente discreti) e quali invece saranno attributi-obiettivo, i cui valori appariranno come entry dell'array multidimensionale.
- Trovare il valore di ogni entri dell'array multidimensionale, sommando i valori dell'attributo-obiettivo o contando il numero di oggetti corrispondenti a quell'entry.

Uno dei vantaggi dell'array multidimensionale consiste nella possibilità che si ha di sezionarlo, ottenendo "fette", rappresentabili come tabelle, che mettono in relazione coppie di attributi.
Secondo lo stesso principio, l'array può essere decomposto in array di dimensioni inferiori, il che corrisponde ad una selezione dei dati in base al valore degli attributi.

Altre due operazioni molto importanti sono il **roll-up** e **roll-down**, che corrispondono rispettivamente all'aggregazione e disaggregazione di attributi con un'intrinseca struttura gerarchica (ad esempio le date espresse come anno-mese-giorno).

5 Clustering

Il clustering affronta il problema di raggruppare gli oggetti in modo tale che gli oggetti di uno stesso cluster siano simili (o in relazione fra loro), mentre siano dissimili (o non in relazione) rispetto a quelli di altri cluster.

Tipi di clustering:

- Partizionale: i dati sono divisi in insiemi non sovrapposti di modo che ogni oggetto appartenga ad un unico cluster.
- Gerarchico: si ottiene un insieme di cluster annidati, organizzati secondo un albero gerarchico.

Distinzioni fra tipi di clustering:

- Esclusivo vs. Non Esclusivo
- Fuzzy vs. Non-Fuzzy
- Parziale vs. Completo
- Eterogeneo vs. Omogeneo

Tipi di cluster:

- Ben separati: ogni punto (oggetto) di un cluster è più vicino ad ogni altro punto dello stesso cluster di quanto non sia rispetto ad un qualsiasi punto di un altro cluster.
- Center-based: ogni punto è più vicino al centro (o centroide) del proprio cluster rispetto a quanto non lo sia rispetto agli altri centri.
- Contiguity-based: ogni punto è più vicino ad uno o più punti del proprio cluster di quanto non lo sia rispetto ad un qualsiasi altro punto fuori dal cluster.
- Densisty-based: un cluster è una regione densa di punti, separata dagli altri cluster da regioni a bassa densità.
- Cluster concettuali: i punti in un cluster condividono una proprietà o il loro insieme rappresenta un particolare concetto.
- Descritti da una funzione obiettivo: si trovano cluster che minimizzano o massimizzano una funzione obiettivo. La funzione obiettivo può essere locale (tipicamente clustering gerarchico) o globale (clustering partizionale). In pratica in questo caso si mappa il problema di

clustering su un dominio differente sul quale si risolve il problema di massimizzazione/minimizzazione.
Esempio:

> *La matrice di prossimità definisce un grafo pesato. Il clustering corrisponde al problema di spezzare il grafo in componenti connesse tali che sia minimo il peso degli archi interni al cluster e massimo quello degli archi fra cluster diversi.*

Caratteristiche dei dati in input che maggiormente influenzano il clustering:

- Tipo di misura di sparsità o prossimità.
- Sparsità (Sparseness).
- Tipo degli attributi.
- Tipo di dati.
- Dimensionalità.
- Rumore ed outliers.
- Tipo di distribuzione.

5.1 K-Means

K-Means è un algoritmo partizionale.
Caratteristiche:

- Ogni cluster è associato ad un proprio *centroide*.
- Ogni punto è assegnato al cluster con il centroide più vicino.
- Ad ogni iterazione vengono ricalcolati i cluster ed i rispettivi centroidi.
- Si termina quando non ci sono più variazioni nella composizione dei cluster.
- Il numero di cluster k è definito apriori.

Dettagli:

- I cluster iniziali sono spesso scelti casualmente (anche se configurazioni iniziali differenti possono portare a risultati diversi).
- Il centroide è (tipicamente) la media fra i punti del cluster.

- La *vicinanza* può essere misurata da: distanza euclidea, similarità del coseno, correlazione, ecc.. Per queste metriche l'algoritmo converge.
- Buona parte della convergenza si ha nelle iterazioni iniziali.
- Complessità: $O(nkId)$
  dove:
  - n : numero di punti
  - k : numero di cluster
  - I : numero di iterazioni
  - d : numero di attributi

La misura che più comunemente si usa per valutare la qualità dei cluster è la Somma degli Errori Quadrati (**SSE**):

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} dist^2(m_i, x)$$

dove x rappresenta i punti del cluster ed m il centroide.

Osservazioni:

> Date due clusterizzazioni, si considera migliore quella con SSE più piccola.
>
> Aumentando k si riduce la SSE.
>
> La SSE può esser utile per scegliere dove posizionare un centroide quando, ad esempio, alcuni cluster rimangono vuoti.

Abbiamo già detto che la scelta dei centroidi iniziali gioca un ruolo importante per lo sviluppo dell'algoritmo. In particolare, poiché k-means evolve minimizzando la SSE, è possibile che l'algoritmo si arresti su un minimo locale che non rappresenta la clusterizzazione ottimale.
In generale sarebbe utile che i centrodi iniziali si trovassero ognuno all'interno di un cluster "reale" distinto.
Anche supponendo di conoscere il numero di cluster k, la probabilità di scegliere casualmente una configurazione iniziale in cui ogni centroide cade ad un cluster reale diverso è $\frac{k!}{k^k}$ (con $k = 10$ la probabilità è $36 \cdot 10^{-5}$).
Esistono diverse soluzioni proposte per risolvere, o almeno ridurre, il problema dei centroidi iniziali:

- Far girare più volte k-means e scegliere il risultato migliore (funziona ma non è geniale).

- Fare sampling ed usare un clustering gerarchico per determinare i centroidi iniziali.
- Usare più dei k centroidi previsti e poi selezionare i k centroidi in base al risultato.
- Fare postprocessing.
- Usare la variante Bisecting K-Means, meno dipendente dalla configurazione iniziale.

Delle varie possibilità la più interessante è rappresentata dalla variante **Bisecting K-Means**. L'algoritmo produce un clusterig gerarchico e funziona nel seguente modo:

- Si parte con un unico cluster che contiene tutti i punti.
- Ad ogni iterazione:
  - Si sceglie un cluster.
  - Si esegue n volte k-means dividendo il cluster in due ($k = 2$).
  - Si sceglie la partizione ottenuta con la minore SSE e si aggiungono i due cluster alla lista.
- Ci si ferma quando la lista contiene k cluster.

In quali casi K-Means è in difficoltà:

- Cluster di diverse dimensioni.
- Cluster di diverse densità.
- Cluster di forma non globulare.
- Presenza di outliers.

In molti casi i problemi di k-means possono essere aggirati aumentando il numero k, per poi fondere insieme alcuni dei cluster trovati, in fase di postprocessing.

5.2 Clustering Gerarchico

Il clustering gerarchico produce una serie di cluster annidati, organizzati come un albero gerarchico, che possono essere visualizzati come un dendogramma.

I punti di forza del clustering gerarchico consistono nel fatto che non è

necessario specificare un numero particolare di cluster da ricercare, e che il risultato può corrispondere a tassonomie significative.

Tipi principali di clustering gerarchico:

- Agglomerativo: all'inizio ogni punto è un cluster distinto. Ad ogni passo si uniscono coppie di cluster vicini fino ad ottenere il numero desiderato di cluster.
- Divisivo: si parte con un unico cluster che contiene tutti i punti. Ad ogni passo i cluster esistenti vengono "spezzati" finché ogni cluster contiene un solo punto.

Gli algoritmi tradizionali per il clustering gerarchico utilizzano una **matrice delle distanze** (o di similarità) ed uniscono/dividono un unico cluster alla volta.

I problemi principali legati al clustering gerarchico sono:

- Le decisioni prese non possono essere annullate.
- Non esiste una funzione obiettivo ma minimizzare.
- A seconda della tipologia di algoritmo usato si possono incontrare problemi legati a: rumore o outliers, cluster di dimensioni differenti o forme convesse, cluster grandi che vengono spezzati.

Vediamo ora più nel dettaglio i vari tipi di cluster gerarchico.

Clustering Agglomerativo
Il funzionamento, in generale, è il seguente:

- Inizialmente ogni punto è un cluster distinto.
- Ad ogni passo:
  - Si uniscono i due cluster più *vicini*.
  - Si aggiorna la matrice di prossimità.
- Si termina quando si ha un unico cluster.

E' abbastanza chiaro che il criterio usato per determinare la similarità (o vicinanza) fra due cluster gioca un ruolo fondamentale.
Ecco i criteri più comuni:

- MIN: minima distanza fra punti dei due cluster.

- MAX: massima distanza fra punti dei due cluster.
- Media di Gruppo (Group Average): distanza media fra tutti i punti dei due cluster.
- Distanza fra i centroidi.
- Metodi guidati da una funzione obiettivo (es. metodo do Ward).

Vediamo le caratteristiche dell'algoritmo in base al criterio di similarità utilizzato:

- MIN (Single-link):
  - Può gestire cluster non globulari.
  - E' fortemente condizionato da rumore ed outliers.
- MAX (Complete linkage):
  - Poco suscettibile a rumore ed outliers.
  - Tende a rompere cluster grandi.
  - Tende a creare cluster globulari.
- Media di Gruppo:
  - Buon compromesso fra MIN e MAX.
  - Poco suscettibile a rumore ed outliers.
  - Tende comunque a creare cluster globulari.
- Metodo di Ward:
  - La similarità fra due cluster è basata sull'incremento dell'SSE qualora fossero uniti.
  - Poco suscettibile a rumore ed outliers.
  - Tende a creare cluster globulari.
  - E' il corrispettivo gerarchico di K-Means.

Clustering Divisivo: Minimum Spanning Tree (MST)
Funzionamento generale:

- Si parte con un albero con un punto qualsiasi.
- Ad ogni passo:
  - Si sceglie la più vicina coppia di punti (p, q) tale che p appartiene all'albero e q no.

  - Si aggiunge all'albero q e l'arco da p a q.

- Si termina quando tutti i nodi sono nell'albero.

Usando MST è possibile ottenere una gerarchia di cluster semplicemente rimuovendo gli archi, a partire da quelli più lunghi (nel senso che rappresentano la distanza maggiore)

5.3 Clustering Density-based

Per il clustering density-based prendiamo in considerazione l'algoritmo DBSCAN.
L'algoritmo si basa sulla suddivisione dei punti in:

- Punti Core: se hanno più di un numero prefissato (*MinPts*) di punti in un loro intorno di raggio fisso (*Eps*).
- Punti Bordo (o Frontiera): se sono nell'intorno di un punto core, ma non hanno sufficienti vicini per essere anch'essi punti core.
- Punti Rumore: tutti quei punti che non sono né core né bordo.

Limiti di DBSCAN:

- Densità diverse in cluster diversi (le soglie sono globali).
- Alta dimensionalità dei dati (perde di significato il concetto di densità).

5.4 Validazione del Clustering

Motivazioni:

- Determinare se emergono o sono verificate strutture non casuali esistenti nei dati.
- Confrontare i risultati del clustering con risultati esterni noti.
- Valutare quanto la suddivisione in cluster sia adatta ai dati a cui è applicata.
- Confrontare diverse suddivisioni in cluster per determinare quale sia la migliore.
- Determinare il "corretto" numero di cluster.

Misure usate per verificare la validità del clustering:

- Indice Esterno: misura quanto l'assegnamento delle etichette secondo il cluster sia rispondente rispetto ad informazioni esterne di classificazione degli oggetti (Entropia).
- Indice Interno: misura la bontà di una struttura di clustering senza usare informazioni esterne (SSE).
- Indice Relativo: usato per confrontare due diverse clusterizzazioni o cluster (Entropia e/o SSE)

Un altro modo per verificare la bontà di un cluster consiste nel calcolare la correlazione tra la matrice di prossimità e la matrice di incidenza, costruita come

> $a_{ij} = 1$ se gli oggetti i e j sono nello stesso cluster, $a_{ij} = 0$ altrimenti

Una correlazione alta indicherà che i punti che appartengono allo stesso cluster sono molto vicini. Ovviamente, la correlazione non è una buona stima per cluster basati su contiguità o su densità.

Un ulteriore approccio, visuale, per valutare la validità di un cluster consiste nell'ordinare la matrice di similarità rispetto alle etichette assegnate e poi visualizzarla attraverso un Martix-Plot.

6 Classificazione

Dato un insieme di record (training-set), in cui ogni record contiene fra i vari attributi uno specifico che ne definisce la classe, ci si propone di trovare un modello per determinare l'attributo di classe in funzione degli altri attributi, col fine di poter assegnare record non visti precedentemente alla classe corretta, con la massima accuratezza possibile.

Per valutare l'accuratezza del modello si usa un test-set, per cui generalmente i dati a disposizione vengono partizionati in **test-set** e **training-set**.

6.1 Classificatori basati su Alberi di Decisione (C4.5 ed altri)

Uno degli approcci più comuni per la classificazione è la costruzione di un albero di decisione, in cui:

- Gli archi sono etichettati con il valore (o il range di valori) di uno degli attributi, di modo che scandendo l'albero si possa partizionare l'insieme dei record in base al valore degli attributi.
- Le foglie indicano la classe di appartenenza dei record individuati, indicando eventualmente una percentuale di confidenza che si ha sull'assegnamento della classe al record.

Procedura generale per la costruzione dell'albero di decisione:

- Sia D_t l'insieme dei record (del training-set) che si ottengono raggiungendo il nodo t.
- Abbiamo tre casi:
  - Se i record in D_t appartengono tutti alla stessa classe y_t, allora t è una foglia etichettata con y_t.
  - Se D_t è vuoto, allora t è una foglia etichettata con l'etichetta di default.
  - Se D_t contiene record di classi differenti, allora si usa un attributo per suddividere D_t in sottoinsiemi e si procede ricorsivamente ad esaminare i nuovi nodi così ottenuti.

E chiaro che la questione principale riguarda la scelta del miglior split applicabile per un nodo. Considerando di mantenere un approccio greedy per cui il miglior split è quello che porta ad una maggiore omogeneità nella distribuzione degli oggetti rispetto alla classe di appartenenza, abbiamo bisogno di una misura che possa quantificare l'impurità di un nodo.
Le principali misure adottate per questo scopo sono:

- Gini Index.
- Entropia.
- Errore di classificazione.

6.1.1 Gini Index

Il Gini Index viene calcolato come:

$$GINI(t) = 1 - \sum_j [P(j \mid t)]^2$$

Caratteristiche:

- Valore massimo: $1 - \frac{1}{n_c}$ con n_c numero di classi.
- Valore minimo: 0 (quando la distribuzione è completamente omogenea).

Per calcolare la qualità di uno split su un nodo p, che genera k figli si usa:

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

dove N_i è il numero di record nel nodo i ed n è il numero di record in p.

Per calcolare efficientemente il Gini Index per uno split binario su attributi a valore continuo si procede nel seguente modo:

- Ordinare gli attributi in base ai valori.
- Fare una scansione lineare dei valori, aggiornando ad ogni passo la matrice dei contatori della classe di appartenenza ed il Gini Index.
- Scegliere per lo split la posizione che ha il più basso Gini Index.

6.1.2 Entropia

L'entropia è calcolata come:

$$Entropy(t) = -\sum_j P(j \mid t)\ log\ P(j \mid t)$$

Caratteristiche:

- Valore massimo: $(log\ n_c)$
- Valore minimo: 0

Rispetto al Gini Index, l'entropia tende a preferire split che siano nel complesso più omogenei, senza necessariamente privilegiare la forte omogeneità su un solo (o una parte) nodo generato dallo split.

Esistono varie misure basate sull'entropia:

- Info Gain: calcola la variazione di entropia nello split (a differenza di Entropia e gini-index viene massimizzata).
- Gain Ratio: versione di Info Gain che evita di privilegiare la formazione di molte partizioni con pochi elementi (problema tipico di Info Gain)

6.1.3 Errore di Classificazione

L'Errore di Classificazione (o Misclassificazione) è:

$$Error(t) = 1 - max(P(i \mid t))$$

Caratteristiche:

- Valore massimo: $(1 - \frac{1}{n_c})$
- Valore minimo: 0

6.2 Problemi legati alla Classificazione

I problemi principali sono:

- Underfitting ed Overfitting.
- Valori mancanti.
- Costo della classificazione.

Underfitting e Overfitting:

> Si ha **underfitting** quando il modello è troppo semplice rispetto alla natura dei dati.
>
> Si ha **overfitting** quando l'albero di decisione è più complesso del necessario. In questo caso il training-set non fornisce una buona stima di come il modello possa funzionare se applicato a dati precedentemente non visti.

Stimare gli errori di generalizzazione:

- Errori di Re-Sostituzione: sul training-set $(\sum e(t))$

- Errori di Generalizzazione: sul test-set ($\sum e'(t)$)
- Metodi per stimare gli errori:
  - Approccio Ottimistico: $e'(t) = e(t)$
  - Approccio Pessimistico:
    - per ogni nodo: $e'(t) = (e(t) + 0.5)$
    - errore totale: $e'(T) = e(T) + N \cdot 0.5$, con N numero di nodi
  - Reduced Error Pruning (REP): usa un dataset di validazione per stimare l'errore.

Come risolvere l'overfitting:

- Pre-pruning:
  - Si ferma l'algoritmo prima che possa generare un albero troppo esteso e quindi troppo specializzato.
- Post-pruning:
  - Si genera l'albero completo.
  - Dal basso si verifica se eventuali potature migliorerebbero l'errore di generalizzazione.
  - Dove serve si pota.

La misura più diffusa per valutare la qualità del modello è la **matrice di confusione**

	Yes	*No*
Yes	TP	FN
No	FP	TN

Dove:

- Le colonne rappresentano le classi previste.
- Le righe rappresentano le classi attuali.
- TP, TN, FP, FN: (rispettivamente *veri positivi e negativi* e *falsi positivi e negativi*) contano il numero di assegnamenti corretti o sbagliati fatti applicando il modello al test-set.

La matrice di confusione fornisce informazioni sull'**accuratezza** del modello:

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

E' possibile anche calcolare il cosiddetto **costo della classificazione**, associando alla matrice di confusione una *matrice dei costi* che assegna un peso ai vari tipi di risultati della classificazione (TP, TN, FP, FN). In questo modo è possibile preferire, ad esempio, un modello che eviti i falsi positivi piuttosto che i falsi negativi ecc.

6.3 Classificatori Rule-based

Questo tipo di classificatori non usa alberi di decisione, ma regole del tipo

$$(Condizione) \rightarrow classe$$

intuitivamente corrispondenti ad uno statement *if... then.*

Definiamo:

- Copertura di una regola: la frazione di record che soddisfano la testa di una regola.
- Accuratezza di una regola: la frazione di record che soddisfano sia la testa che la coda di una regola.

Il passaggio da un albero di decisione ad un sistema a regole è abbastanza intuitivo. Una volta trasformato il modello è poi possibile semplificare le regole ottenute (anche se si rischia di perdere alcune proprietà del modello ottenuto).

Proprietà dei classificatori Rue-based:

- Regole mutuamente esclusive:
  - Se le regole sono indipendenti fra loro.
  - Ogni record è coperto al più da una regola.
  - Si può ottenere definendo un ordinamento sulle regole.
- Regole esaustive:
  - Se il classificatore copre ogni possibile combinazione dei valori degli attributi.
  - Ogni record è coperto da almeno un regola.
  - Si può ottenere definendo una classe di default.

Costruzione delle regole di classificazione:

- Metodo diretto: si estraggono le regole direttamente dai dati (Ripper, CN2, ecc.).
- Metodo indiretto: si estraggono le regole a partire da un altro modello di classificazione (C4.5-rules).

Vantaggi dei classificatori basati su regole:

- Espressivi quanto gli alberi di decisione.
- Facili da interpretare.
- Facili da generare.
- Possono classificare rapidamente nuove istanze.
- Performance comparabili con quelle degli alberi di decisione.

6.4 Classificatori Nearest-Neighbor (K-NN)

L'idea alla base di questi classificatori è che un record sconosciuto possa probabilmente avere le stessa classe del record conosciuto ad esso più vicino (rispetto ad una qualche misura di similarità).

Come funziona:

- Cosa serve:
  - L'insieme dei record memorizzati.
  - Una metrica per misurare la distanza.
  - Un intero prefissato k che indica in numero di vicini da prendere in considerazione.
- Per classificare un nuovo record:
  - Si calcola la distanza dai training records.
  - Si selezionano i k vicini più prossimi.
  - Si usano le etichette di classe dei vicini selezionati per determinare la classe del nuovo record, prendendo la classe più ricorrente e tenendo in considerazione dei pesi inversamente proporzionali alla distanza.

Caratteristiche dei classificatori K-NN:

- Non costruiscono il modello esplicitamente (a differenza di quelli basati su albero di decisione o regole).
- La classificazione di nuovi individui è piuttosto costosa.

7 Mining di Regole Associative

Dato un insieme di **transazioni** si vogliono trovare delle regole che consentano di prevedere la presenza di un oggetto basandosi sull'occorrenza di altri oggetti nella transazione.

Un po' di definizioni:

- Itemset: una collezione di uno o più oggetti.
- K-itemset: un itemset con esattamente k item.
- Support Count (σ): frequenza dell'occorrenza di un itemset (più precisamente: il numero di volte che l'itemset compare).
- Supporto: frazione di transazioni che contengono un itemset rispetto al totale delle transazioni.
- Regola Associativa: un'espressione di implicazione nella forma

  $$X \rightarrow Y$$

  con X e Y itemset.

Metriche per la valutazione delle regole:

- **Supporto** (s): rapporto delle transazioni che contengono sia X che Y.
- **Confidenza** (c): misura di quanto spesso Y compare in transazioni che contengono X.

Formulazione del problema:

> Dato un insieme di transazioni T, l'obiettivo è trovare regole associative che abbiano:
>
> - Supporto: $s \geq minsup$
> - Confidenza: $c \geq minconf$

Approccio generale in due passi:

- Generazione degli itemset frequenti.
- Generazione di regole con sufficiente confidenza.

La generazione degli itemset frequenti ha un costo computazionale molto alto. Se adottassimo un approccio "brutale" il costo computazionale sarebbe

$$O(NMW)$$

dove:

- N: numero di transazioni.
- M: numero di itemset candidati (s^d).
- W: numero di item per transazione.

Occorrono quindi strategie specifiche per la generazione di itemset frequenti, con i seguenti obiettivi:

- Ridurre il numero di candidati (M) con tecniche di pruning.
- Ridurre il numero di transazioni (N) all'aumentare delle dimensioni dell'itemset.
- Ridurre il numero di confronti (NM) utilizzando apposite strutture dati ed evitando confronti non strettamente necessari.

7.1 Algoritmo Apriori

Apriori Principle

> Se un itemset è frequente allora lo sono anche i suoi sottoinsiemi.

Il principio si basa sulla proprietà di **anti-monotonia** del supporto:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

L'algoritmo Apriori utilizza l'Apriori Principle per ridurre il numero di itemset candidati. Ecco come funziona:

- Sia k=1
- Generare gli itemset frequenti di lunghezza 1.
- Ad ogni passo:
  - Generare i candidati di lunghezza k+1 a partire dagli itemset frequenti di lunghezza k.
  - Potare i candidati che contengono itemset di lunghezza k infrequenti.
  - Calcolare il supporto dei candidati rimasti.

  - Eliminare i candidati risultati infrequenti.

- Terminazione quando non si trovano nuovi itemset frequenti.

Per ridurre il numero di confronti necessari a calcolare il supporto degli itemset candidati è possibile evitare di dover fare ogni volta una scansione completa del dataset. Per far questo è sufficiente memorizzare i candidati all'interno di una tabella hash, in modo da limitare i confronti ai soli candidati contenuti negli hash-bucket.

Fattori che influenzano la complessità:

- Scelta della soglia di supporto minimo.
- Dimensionalità.
- Dimensione del database.
- Lunghezza media delle transazioni.

Categorie di itemset frequenti:

- Massimale: nessuno dei suoi superinsiemi immediati è un itemset frequente.
- Chiuso: nessuno dei suoi superinsiemi immediati ha il suo stesso supporto.

Osservazione:

Si ha che:

$$ISFrequenti \supseteq ISFrequentiChiusi \supseteq ISFrequentiMassimali$$

Vari approcci per la generazione di itemset frequenti:

- General-to-Specific
- Specific-to-General
- Bidirectional
- Prefix Tree
- Suffix Tree
- Breadth First
- Depth First

7.2 Algoritmo FP-Growth

L'algoritmo usa una rappresentazione compatta del database, attraverso un albero dei prefissi (FP-tree) ed applica un approccio ricorsivo divide-et-impera per estrarre gli itemset frequenti.

Costruzione dell'albero FP-tree:

- Viene costruito come albero dei prefissi.
- Ad ogni nodo è associato un contatore che indica quante volte l'item compare in quella posizione all'interno delle transazioni.
- Si costruisce anche una *Tabella degli Header* che mantiene, per ogni item, dei puntatori ai punti in cui gli item compaiono sull'albero.

L'algoritmo FP-Growth (linee guida):

> Per generare gli itemset frequenti l'algoritmo adotta un approccio bottom-up sull'FP-tree. Usando la tabella degli header ed i contatori, l'algoritmo trova gli itemset frequenti che terminano con uno specifico item. Dopo aver esaminato in questo modo tutti i possibili item terminali, l'algoritmo procede ricorsivamente a trovare itemset con suffissi sempre più grandi, agendo soltanto sulle porzioni di albero che gli interessano per un determinato suffisso.

Caratteristiche di FP-Growth:

- E' più complesso rispetto ad Apriori.
- E' più efficiente.
- Se la rappresentazione del dataset tramite FP-tree è sufficientemente compressa, permette di mantenere l'intero dataset in memoria.

7.3 Generazione di regole associative

Abbiamo visto alcuni approcci per la generazione degli itemset frequenti. Rimane ancora da vedere il passo successivo, ovvero la generazione delle regole associative.

Dato un itemset frequente L il problema è quello di trovare tutti i sottoinsiemi non vuoti ($f \subset L$) per cui la regola ($f \rightarrow L - f$) abbia la confidenza richiesta.
Se $\mid L \mid = k$ ci sono $2^k - 2$ regole associative candidate. Per evitare di dover testare tutte le regole candidate ancora una volta si ricorre alla proprietà di

anti-monotonia, questa volta della confidenza.

Anti-monotonia della confidenza:

> La confidenza di regole generate da uno stesso itemset verifica la proprietà di antimonotonia.
> Es. L=ABCD $c(ABC \rightarrow D) \geq (AB \rightarrow CD) \geq (A \rightarrow BCD)$

7.4 Regole Associative per attributi non binari

Fin'ora abbiamo considerato solo il caso di attributi con valori binari asimmetrici[2]. Per poter gestire attributi con caratteristiche diverse da quelle appena descritte è necessario operare trasformazioni dei dati, dipendenti dal tipo di dato.

Come gestire i vari tipo di attributo:

- Attributi Categorici: vengono trasformati in variabili booleane asimmetriche, introducendo un nuovo "item" per ogni possibile possibile coppia attributo-valore.
  Potenziali problemi:
  - L'attributo ha molti valori per cui si possono creare molti item con supporto molto basso.
  - La distribuzione dei valori è fortemente sbilanciata e quindi avremo scarsa confidenza.
- Attributi Continui: esistono in questo caso molti metodi diversi per trasformare gli attributi in variabili booleane asimmetriche. Le varie tecniche sono basate su:
  - Discretizzazione.
  - Statistica.
  - Non-discretizzazione (Min-Apriori).

Vediamo le varie tecniche più nel dettaglio.

7.4.1 Discretizzazione

Due tipi:

- Non supervisionata:
  - Equal-width binning.

[2]I due valori non hanno la stessa importanza, perché a noi interessa solo la presenza di un item e non la sua mancanza.

  - Equal-depth binning.
  - Clustering.

- Supervisionata: eseguita "manualmente" osservando i valori e la natura dei dati.

In generale la dimensione degli intervalli influenza il supporto e la confidenza:

- Intervalli troppo piccoli: supporto insufficiente.
- Intervalli troppo grandi: confidenza insufficiente.

7.4.2 Metodi Statistici

I metodi basati sulla statistica si applicano quando la parte destra della regola è un attributo quantitativo, caratterizzato dalla propria statistica.
Ad esempio:

$$Browser = Mozzilla \ \wedge \ Buy = Yes \rightarrow Age : \mu = 23$$

In questi casi per determinare se una regola è interessante o meno, confrontiamo la statistica per il segmento di popolazione coperto dalla regola con il segmento di popolazione non coperto

$$A \rightarrow B : \mu \quad vs. \quad \overline{A} \rightarrow B : \mu'$$

7.4.3 Metodo Min-Apriori

In questo caso abbiamo a che fare unicamente con attributi continui dello stesso "tipo" (ad esempio il count della presenza di parole in un documento).

Come funziona:

- Si normalizzano i valori di modo che ogni "item" abbia supporto 1 (somma per colonna).
  Esempio:

	w1	w2	w3	w4	w5
d1	2	2	0	0	1
d2	0	0	1	2	2
d3	2	3	0	0	0
d4	0	0	1	0	1
d5	1	1	1	0	2

  diventa

	w1	w2	w3	w4	w5
d1	0.4	0.33	0	0	0.17
d2	0	0	0.33	1	0.33
d3	0.4	0.5	0	0	0
d4	0	0	0.33	0	0.17
d5	0.2	0.17	0.33	0	0.33

- Si modifica il concetto si supporto

$$sup(C) = \sum_{i \in T} min_{j \in C}\ D(i,j)$$

  Esempio: $sup(w1, w2, w3) = 0 + 0 + 0 + 0 + 0.17 = 0.17$

Poiché è conservata la proprietà di anti-monotonia del supporto, possiamo dunque applicare l'algoritmo Apriori.

7.5 Regole Associative Multilivello

L'introduzione del concetto di gerarchia per la generazione di regole associative risponde al fatto che spesso le regole al livello più basso non hanno sufficiente supporto per comparire in un itemset frequente.
Questa caratteristica, dovuta ad una eccessiva specificità dell'item, comporta una perdita di informazione, recuperabile attraverso l'introduzione di una gerarchia.

Come variano supporto e confidenza introducendo il concetto di gerarchia:

- Se X è genitore di X_1 e X_2 allora:

$$\sigma(X) \leq \sigma(X_1) + \sigma(X_2)$$

- Se $\sigma(X_1 \cup Y_1) \geq minsup$ e X è genitore di X_1, e Y è genitore di Y_1, allora:

$$\sigma(X \cup Y_1) \geq minsup,\ \sigma(X_1 \cup Y) \geq minsup,\ \sigma(X \cup Y) \geq minsup$$

- Se $conf(X_1 \rightarrow Y_1) \geq minconf$ allora

$$conf(X_1 \rightarrow Y) \geq minconf$$

Come generare regole associative multilivello:

- Si estendono le regole associative aggiungendo ad ogni transazione gli items dei livelli superiori.
  Problemi:
  - Gli item dei livelli superiori avranno supporto molto alto e compariranno in moltissimi itemset frequenti.
  - Aumenta la dimensionalità dei dati.
- Si generano prima i pattern frequenti dei livelli più alti della gerarchia.
  Problemi:
  - Aumenta il costo delle operazioni di I/O perché sono necessarie più passate sui dati.
  - Si può non individuare pattern interessanti che riguardano livelli diversi della gerarchia.

7.6 Analisi Associative su Dati Sequenziali

In questo caso abbiamo a che fare con dataset che hanno associate ai record (eventi) anche informazioni temporali.

Abbiamo quindi:

- Una tabella di oggetti (non più *items* ma *objects*).
- Per ogni oggetto:
  - Un'informazione temporale.
  - Una serie di eventi.

Da un simile dataset quello che vogliamo ottenere sono regole che consentano di prevedere (letteralmente) che ad una sequenza di eventi succederà nel tempo un evento particolare.

Definizioni:

- **Sequenza**:
  - Una sequenza è una lista ordinata di elementi $s =< e_1 e_2 e_3 ... >$ tale che:
    - ogni elemento contiene una collezione di eventi: $e_i = \{i_1, i_2, ..., i_k\}$
    - ogni elemento è associato ad uno specifico tempo.
- **Sottosequenza**:

– una sequenza $< a_1 a_2 ... a_n >$ è contenuta in un'altra sequenza $< b_1 b_2 ... b_m >$ $(m \geq n)$ se esistono degli interi $i_1 < i_2 < ... < i_n$ tali che:

$$a_{i_1} \subseteq b_{i_1},\ a_{i_2} \subseteq b_{i_2},\ ...,\ a_{i_n} \subseteq b_{i_n}$$

Esempi:

$$< \{2,4\}\{3,5,6\}\{8\} > \supseteq < \{2\}\{3,5\} >$$
$$< \{1,2\}\{3,4\} > \not\supseteq < \{1\}\{2\} >$$
$$< \{2,4\}\{2,4\}\{2,5\} > \supseteq < \{2\}\{4\} >$$

Definizione di **Supporto**:

> Il supporto di una sottosequenza w è definito come la frazione di sequenze di dati che contengono w.

Definizione di **Pattern Sequenziale**:

> Un pattern sequenziale è una sottosequenza frequente (con supporto $s \geq minsup$.

Formulazione del problema:

> Dato un database di sequenze, ed una soglia *minsup*, vogliamo trovare tutte le sottosequenze con supporto $s \leq minsup$.

Osservazione:

> Data una sequenza con n eventi (n-sequenza), il numero di k-sottosequenze che possiamo trovare è: $\begin{pmatrix} n \\ k \end{pmatrix}$

7.6.1 General Sequential Pattern (GSP)

Funzionamento dell'algoritmo:

- Si inizia scandendo il database e raccogliendo tutte le 1-sequenze frequenti.
- Ad ogni iterazione:
  - Si generano i candidati: si fa il merge dei candidati trovati al passo (k-1)-esimo per ottenere i candidati con k items.
  - Potatura dei candidati: si eliminano i candidati che contengono (k-1)-sequenze infrequenti (Apriori Principle).
  - Si calcola il supporto eseguendo una nuova passata sul database.

  - Si eliminano i candidati risultati infrequenti.

- Si termina quando non si ottengono più nuove sottosequenze frequenti.

Come si esegue il merge di sottosequenze:

> Una (k-1)-sottoseqenza candidata w_1 viene unita con un'altra (k-1)-sottosequenza candidata w_2 per produrre una nuova k-sequenza candidata, se e solo se si ottiene la stessa sottosequenza rimuovendo da w_1 il primo evento e da w_2 l'ultimo evento.
> In questo caso la k-sequenza risultante sarà l'estensione di w_1 con l'ultimo evento di w_2:
>
> - Se gli ultimi 2 eventi in w_2 appartengono allo stesso elemento, allora l'ultimo evento di w_2 diventa parte dell'ultimo elemento in w_1.
> - Altrimenti l'ultimo evento di w_2 viene "appeso" a w_1 come elemento separato.
>
> Esempi:
>
> $$< \{1\}\{2,3\}\{4\} > \ merge \ < \{2,3\}\{4,5\} > = < \{1\}\{2,3\}\{4,5\} >$$
> $$< \{1\}\{2,3\}\{4\} > \ merge \ < \{2,3\}\{4\}\{5\} > = < \{1\}\{2,3\}\{4\}\{5\} >$$

7.6.2 Aggiunta di Vincoli Temporali

Possiamo fare in modo che le sottosequenze rispettino determinati vincoli temporali.
In particolare:

- **max-gap**: impone che non trascorra più di un certo tempo fra l'inizio di un evento e la fine del successivo.
- **min-gap**: impone che ci sia un tempo minimo fra un evento e il successivo.
- **maximum-span**: impone che la durata dell'intera sequenza non superi una determinata soglia.

L'aggiunta di vincoli temporali può essere affrontata in due modi possibili:

- Generare i pattern sequenziali normalmente e poi postprocessare i pattern scoperti, eliminando quelli che violano i vincoli.

- Modificare GSP in modo che consideri anche i vincoli nella fase di pruning. Questa soluzione richiede accorgimenti particolare, perché in presenza di *max-gap* non è più valido l'Apriori Principle.

Definizione di **sottosequenza contigua**:

> Una sottosequenza s è contigua a $w =< e_1 >< e_2 > ... < e_k >$ se almeno una delle tre condizioni seguenti è verificata:
>
> - s si ottiene da w eliminando e_1 o e_k.
> - s si ottiene da w eliminando un item da ogni elemento e_i che contiene più di un item.
> - s è sottosequenza contigua di s' e s' è sottosequenza contigua di w

Sfruttando la nozione di sottosequenza contigua è possibile modificare il passo di pruning dei candidati in GSP, in questo modo:

- Senza vincolo *max-gap*: un candidato è potato se almeno una delle sue (k-1)-sottosequenze è infrequente.
- Con vincolo *max-gap*: un candidato è potato se almeno una delle sue (k-1)-sottosequenze contigue è infrequente.

Un altro vincolo comunemente usato è il **window size** che impone che ogni evento non contenga più di un numero prefissato di eventi.

8 Tutela della Privacy

Le direttive per la tutela della privacy impongono che i dati vengano cancellati o anonimizzati quando non sono più strettamente necessari. Le ricerche di data mining devono quindi essere eseguite su dati anonimi, ma si presenta il problema di definire quando un dato possa considerarsi completamente anonimo.

La semplice eliminazione di attributi identificativi (che per altro sono generalmente inutili ai fini del mining dei dati) non risulta sempre una soluzione soddisfacente.
In questi casi infatti si può ricorrere a cosiddetti **quasi-identificatori** (ad esempio la coppia *data di nascita-ZipCode*) che consentono di identificare record integrando informazioni provenienti da diverse fonti (anche pubbliche).

Anche eliminando i quasi-identificatori esistono situazioni in cui i dati forniscono informazioni sull'identità di un record, indicandone il comportamento, gli interessi, gli spostamenti (ad esempio la lista delle ricerche effettuate su un motore di ricerca, o il listato degli spostamenti frequenti).

Un terzo approccio possibile consiste nell'aggregare i dati: pubblicare informazioni aggregate e mantenere segreti i sorgenti.
Anche questo approccio però risulta inadeguato in alcuni casi. Se abbiamo una regola associativa con altissimo supporto, ad esempio, possiamo usarla per individuare informazioni specifiche sui pochi record che non la soddisfano.

La soluzione definitiva (nei limiti del possibile) è quindi quella di perturbare i dati in modo che non sia possibile l'identificazione dei record sul database originale, ma in modo anche di poter estrarre conoscenza valida dai dati.

8.1 K-Anonymity

L'algoritmo si basa sulla generalizzazione dei quasi-identificatori.
L'idea generale è che, fissato un intero k, si modificano i dati in modo che ogni quasi-identificatore compaia almeno k volte nella tabella.
In questo modo:

- Ogni riga della tabella è "nascosta" in almeno k righe.
- Ogni persona coinvolta è indistinguibile da almeno altre k persone.

Osservazione:

Ovviamente l'assegnazione dei quasi-identificatori deve essere fatta con criterio: se tutti e k gli individui con uno stesso quasi-identificatore si ritrovassero ad avere lo stesso valore per un determinato attributo, allora K-Anonymity avrebbe completamente mancato il suo obiettivo.