Data Mining - Appunti

Federico Calò

Contents

1 KDD process

L'automazione delle attività economiche produce un incremento dello stream di dati perchè anche singole transazioni (una chiamata telefonica, il credito di una carta, un test medico) sono tipicamente registrate in un computer. Le basi di dati scientifiche e governative sono in rapida crescita. C'è un divario crescente tra la generazione di dati e la loro comprensione. Risulta quindi necessario l'utilizzo di computer per analizzare i dati, ma questo non è sufficiente.

Necessitiamo di una metodologia matura che spieghi come grandi strutture di dati possono essere analizzate. Questa metodologia è stata studiata in un'area di ricerca conosciuta come KDD. Lo scopo è investigare come tecniche di Machine Learning possono essere applicate a estratti di "conoscenza" di una grande massa di dati disponibili. All'inizio vi era una certa confusione sull'area di interesse ricoperta dal Machine learning, Data Mining e Knowledge Discovery. Ora, si è giunti alla conclusione che il KDD denota l'intero processo di estrazione della conoscenza, dalla raccolta dei dati per poi effettuare una pre-elaborazione degli stessi fino all'interpretazione dei risultati.

Il Data Mining è lo step, all'interno del processo KDD, nel quale le informazioni sono estratte dai dati applicando ad essi opportuni algoritmi. Questi algoritmi sono il più delle volte quelli di Machine Learning. In un contesto aziendale il termine Data Mining è ancora utilizzato per denotare il processo di knowledge discovery e questo causa qualche confusione. Sempre sulla terminologia: KDD è ancora utilizzato anche se la conoscenza non è rigorosamente estratta dai "database".

Si potrebbe definire il Knowledge Discovery come un processo nell seguente modo: "Il Knowledge Discovery è l'estrazione non banale di informazioni implicite, precedentemente sconosciute e potenzialmente utili dai dati."

- **Non banale**: si intende che nel processo è coinvolta qualche ricerca o inferenza statistica, quindi non è un semplice calcolo di quantità predefinite;
- **Implicita**: ci si riferisce al fatto che l'informazione è implicita nel dato e non formalmente esplicita, l'informazione esplicita è estratta attraverso altre tecniche;
- **Sconosciuta**: l'informazione deve essere nuova, la novità dipende dal quadro di riferimento assunto;
- **Utile**: l'informazione deve essere utile a raggiungere lo scopo del sistema o dell'utente. I pattern completamente estranei agli obiettivi dati sono di scarsa utilità e non costituiscono conoscenza all'interno della situazione data.

Possiamo formalmente definire il KDD nel seguente modo:

Dati:

- un insieme di fatti (data) F,
- una rappresentazione in un linguaggio L,
- una certa misura di certezza C,

possiamo definire un pattern come una dichiarazione S in L che descrive le relazioni tra un sotto insieme F_S di F con una certezza $\boldsymbol{c}$, tale che S è più semplice (in un certo senso) dell'enumerazione di tutti i fatti in F_S.

Un pattern è considerato conoscenza se è interessante e abbastanza certo (o valido). Nel KDD siamo interessati in pattern che sono espressi in un linguaggio di alto livello, come:

Se Età < 25 e Corso-di-Educazione = no

Allora Incidente = si

Con probabilità = 0.2 a 0.3

In questo modo alcuni pattern possono essere capiti e usati direttamente dalle persone o possono essere input ad altri programmatori. Definiamo ora il termine **certezza** come qul livello sufficiente di certezza senza il quale i modelli diventano ingiustificati e non riescono a diventare conoscenza. La certezza coinvolge diversi fattori, quali:

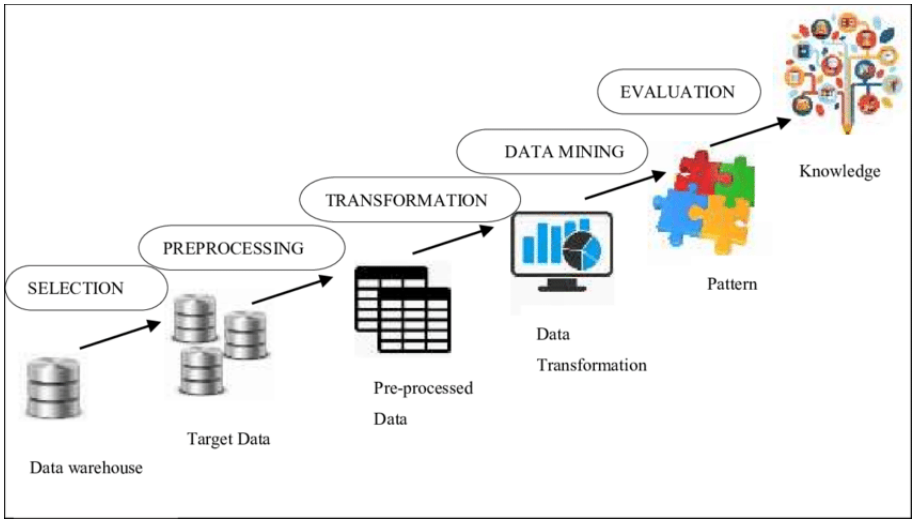
- Integrità dei dati;
- Dimensione del campione su cui è stata effettuata la scoperta;
- Il grado di supporto dalla conoscenza del dominio disponibile.

Un pattern è definito **interessante** quando è:

- Nuovo;
- Utile;
- Non banale da calcolare.

Vi sono principalmente due differenti interpretazioni dei pattern e dei modelli. La *prima interpretazione* che possiamo dare consiste nel definire un **modello** come una sintesi globale del data-set, mentre il **pattern** è una caratteristica locale del data-set, limitato a un sub-set di osservazioni e/o attributi. La *seconda interpretazione* esplica come il data mining implica l'adattamento o la determinazione di pattern da dati osservati. In questo caso il pattern è visto come una istanza del modello. I modelli adattati svolgono il ruolo di conoscenza dedotta.

Il KDD è un processo iterativo e interattivo, costituito da molti passaggi che includono molte decisione prese dall'utente.



In una prima fase si sviluppa una **comprensione del dominio** dell'applicazione, delle relative conoscenze di base e degli obiettivi dell'utente finale. Si prosegue creando **un data set target**, selezionando un dataset o ci si focalizza su un sotto

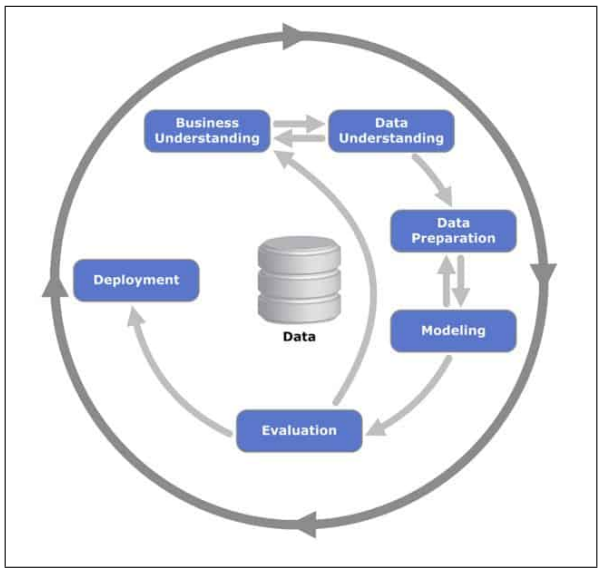
insieme di variabili o di esempi di dati sui quali deve essere eseguita la scoperta. Una volta selezionati questi dati si passa alla fase di **pulizia dei dati e preprocessing** nella quale si effettuano svariate operazioni volte a rimuovere il rumore o dei valori anomali, collezionando le informazioni necessarie per modellarle o tenere conto del rumore, decidere strategie per gestire i campi di dati mancanti o tenere conto delle informazioni sulla sequenza temporale e delle modifiche note. Successivamente si entra in una fase di **riduzione e proiezione dei dati**, nella quale si trovano funzioni utili per rappresentare i dati a seconda dell'obiettivo dell'attività. Vengono utilizzati metodi di riduzione o trasformazione multidimensionale per ridurre l'effettivo numero di variabili da considerare o per trovare rappresentazioni di dati invarianti. Conclusa questa attività, si passa alla **selezione del task di data mining**, decidendo qual è l'obiettivo del processo KDD, scegliendo tra classificazione, regressione, clustering.

Definito l'obiettivo si passa alla **scelta degli algoritmi** di data mining. Vengono selezionati i metodi che verranno utilizzati per ricercare i pattern frequenti all'interno dei dati. Vengono definiti i modelli e i parametri più appropriati e si cerca di far corrispondere particolari metodi di data mining con i criteri globali dei processi di KDD. Quindi si entra nella fase di **Data Mining**, nella quale si ricercano modelli di interesse in una particolare forma rappresentativa. L'utente può contribuire significativamente in questa fase eseguendo correttamente i passaggi precedenti. Alla fine del processo di data mining, vi è la fase di **interpretazione dei pattern minati**, nella quale si effettuano le valutazioni sui risultati ottenuti ed eventualmente si ritorna ad iterare su uno degli step precedenti. Infine si **consolida la conoscenza scoperta**, incorporando la conoscenza ottenuta nel sistema per migliorarne le prestazioni o semplicemente si documenta per essere segnalata ad altre parti di interesse. Questa fase include anche il controllo e la risoluzione di particolar conflitti con conoscenze precedentemente estratte o scoperte.

1.1 Steps del KDD

La maggior parte del lavoro nel KDD è focalizzata sullo step del data mining. Anche se gli altri step sono considerati importanti per il successo dell'applicazione del KDD nella pratica. La necessità per una standardizzazione del processo della scoperta della conoscenza ha portato alla definizione dello standard industriale CRISP-DM. L'obiettivo di questo standard è quello di sviluppare un processo neutrale per condurre il KD, e di definire i compiti, i loro outputs, la terminologia e i problemi tipici di caratterizzazione.

Il modello di processo CRISP-DM è per la scoperta i conoscenza consiste in sei fasi.



La sequenza delle fasi non è rigida. È possibile spostarsi avanti e indietro tra le diverse fasi a seconda dell'esito di ciascuna fase. Le frecce indicano le dipendenze più importanti/frequenti. Il cerchio esterno simboleggia la natura ciclica di un processo KDD che può continuare dopo l'implementazione di una soluzione. Ogni fase contiene un numero di task che produce specifici output.

1.1.1 Business Understanding

La *prima fase* di questo modello viene definita **Business Understanding**, il cui primo step è quello di determinare gli obiettivi commerciali. I requisiti minimi sono:

- un problema o un'opportunità commerciale percepita
- un certo livello di sponsorizzazione esecutiva

Sviluppare una definizione chiara e comprensibile dei bisogni aziendali non è un compito semplice. E' richiesta la collaborazione dei business analyst e dei data analyst. Questo passaggio del processo è anche il momento in cui iniziare a definire le aspettative. Alla fine di questo step vengono prodotti:

- **Il background**, che descrive le informazioni note sulla situazione aziendale all'inizio del processo;
- **Gli obiettivi di business**, che descrivono i principali obiettivi dal punto di vista del business;
- **I criteri economici di successo**, che definiscono le misure per risultati di alta qualità del progetto dal punto di vista del business.

Dopo aver definito gli obiettivi commerciali, si passa alla **valutazione della situazione**, con lo scopo di raccogliere informazioni sulle risorse, sui vincoli e sulle assunzioni. Alla fine di questo sotto processo verrà prodotto un inventario contenente:

- le risorse: personale, dati, calcoli;

- vincoli: schede di compilazione, questioni legali, comprensibilità;
- assunzioni: disponibilità dei dati;

Inoltre non dimentichiamo che vengono prodotti anche **un glossario di termini**, che copre la terminologia di business e di data mining, e **un'analisi costo-beneficio**, cioè un documento contenente le spese del progetto dovrebbero essere confrontate con i potenziali guadagni.

Successivamente si passa alla **determinazione degli obiettivi del Data Mining**. In questa fase si trasformano gli obiettivi commerciali in obietti del processo di Data Mining e si costruiscono i relativi criteri di successo. Possiamo classificare gli obiettivi del Data Mining in:

- Classification
- Estimation (produrre una stima, Estimation)
- Prediction
- Affinity grouping
- Clustering
- Description e Profiling

I due obiettivi primari del Data Mining tendono ad essere:

- **Predizione**, che include l'uso di alcune variabili indipendenti per predire valori sconosciuti o futuri che dipendono da altre variabili;
- **Descrizione**, nella quale non si fa una distinzione tra variabili dipendenti e indipendenti e si concentra sulla ricerca di modelli interpretabili dall'uomo che descrivono i dati.

L'obiettivo della classificazione è apprendere una funzione che mappa, o classifica, un dato in una predefinita classe. Questa tecnica è molto utilizzata per i database. Invece quando si parla di regressione, si ha come obiettivo apprendere una funzione che mappa un dato in una variabile di previsione reale. Quando si cerca di creare un procedimento per trovare le associazioni tra gruppi di variabili, si utilizza il metodo Affinity Grouping. Il clustering è un'attività descrittiva in cui si cerca di identificare un insieme finito di categorie o cluster per descrivere i dati. Le categorie possono essere mutuamente esclusive ed esaustive, oppure consistere in una rappresentazione più ricca come una gerarchia o categorie sovrapposte. La clasterizzazione viene utilizzata spesso per scoprire sotto-popolazioni omogenee, identificare sotto categorie, o analisi di dati. Strettamente correlato al clustering è il compito della stima della densità di probabilità, che consiste in tecniche per stimare dai dati la funzione di densità di probabilità multivariata congiunta di tutte le variabili/campi nel database. Per **Summarization** (riassunto o descrizione o profilazione), si intendono tutti quei metodi per trovare una descrizione compatta per un sottoinsieme di dati. Invece il task **Dependency modeling** (modellare le dipendenze), consiste nel trovare un modello che descrive dipendenze significative tra variabili. Esistono due tipi di modelli dipendenti:

- il livello strutturale di specifici modelli le cui variabili sono localmente dipendenti tra loro

- il livello quantitativo il cui modello specifica quanto le variabili sono dipendenti usando una scala numerica.

Il task di rilevamento di modifiche e deviazioni si focalizza sulla scoperta delle modifiche più significanti nei dati rispetto a valori misurati o normativi in precedenza. Consiste nel trovare un modello che descrive significanti dipendenze tra variabili.

Dopo aver definito il task del Data Mining si passa a **produrre un piano progettuale** per il raggiungimento egli obiettivi di data mining e quindi il raggiungimento degli obiettivi di business. L'output di questa fase è ovviante un piano progettuale, che specifica l'insieme degli step per la restante parte del progetto, la sua durata, le risorse richieste, gli input, gli output e le dipendenze.

1.1.2 Data Understanding

Dopo aver contestualizzato il business in cui si andrà a progettare il sistema, si passa a una fase di **Data Understanding**, cioè di compresone dei dati, che inizia con una fase di raccolta iniziale dei dati. Durante questa fase si accede ai dati rilevanti nell'inventario delle risorse e si produce un report iniziale dei dati raccolti, nel quale si elenca la posizione dei dati, i metodi usati per acquisirli e i problemi incontrati. Successivamente si passa a descrivere i dati esaminando le loro proprietà e creando un report nel quale si descrive il formato dei dati, i potenziali valori, la quantità, l'identificatore dei campi e tutte le caratteristiche che vengono scoperte.

Ci sono due metodi principali per descrivere le variabili:

- **variabili categorighe**: i possibili valori finiti e i differenti tipi che una variabile può assumere. Questa categoria si può suddividere in:
  - **variabili nominali** che denominano il tipo di oggetto a cui si riferiscono, ma non esiste un ordine tra i valori possibili. (stato del materiale, genere, livello di educazione)
  - **variabili ordinali** che assumono un ordine tra i possibili valori. (valutazione del cliente)
- **variabili quantitative**: sui quali sono consentite operazioni aritmetiche, e si suddividono a loro volta in:
  - **variabili discrete**, i cui valori sono interi
  - **variabili continue**, i cui valori sono numeri reali.

Dopo aver descritto la tipologia di dati si passa alla verifica della loro qualità ispezionando e affrontando diverse caratteristiche quali:

- **Accuratezza**: conformità del valore memorizzato rispetto a quello effettivo
- **Completezza**: nessun valore mancante
- **Consistenza**: rappresentazione uniforme
- **Attualità**: i dati storicizzati non sono obsoleti.

Scarsa qualità dei dati e *scarsa integrità dei dati* sono i maggiori problemi all'interno dei progetti KDD. La maggior parte dei dati operativi non è mai stata acquisita o modellata per scopi di data mining. I dati selezionati vengono generalmente

raccolti da numerosi sistemi operativi, incoerenti e scarsamente documentati. È importante comprendere la **sensibilità temporale** dei dati. Lo specialista della gestione dei dati è responsabile della raccolta e dell'integrazione dei dati nell'ambiente informativo.

Al termine della verifica della qualità dei dati viene gerato un report di qualità dei dati che riporta i risultati della verifica e se vi sono dei problemi, sarà possibile discutere di eventuali soluzioni.

A valle della verifica della qualità dei dati, è possibile avviare la fase di **esplorazione dei dati**, alla fine della quale seguirà un relativo report dei risultati ottenuti. Per le *variabili categoriche*, le distribuzioni della frequenza dei dati sono il metodo migliore per capire il contenuto dei dati. Istogrammi e grafici a torta aiutano a identificare gli schemi della distribuzione e i valori mancanti o non validi. Mentre quando lavoriamo con *variabili quantitative*, l'analista dei dati è interessato a misure come il massimo e il minimo, la media, moda, mediana e misure statistiche. Se combinate, queste misure offrono un modo efficace per determinare la presenza di dati non validi e distorti.

1.1.3 Data Preparation

Lo step successivo al Data Understang vi è lo step di **Data Preparation**, la cui prima fase consiste nel selezionare i dati, nella quale il problema maggiore consiste nel selezionare dati da tuple di database relazionali. Si possono comunque seguire alcuni principi che includono:

- la rilevanza del dato rispetto all'obiettivo principale.
- vincoli tecnici e qualitativi,
- limiti al volume dei dati o ai tipi di dati

In questa fase si produce un report di inclusione/esclusione dei dati. La selezione dei dati può essere eseguita manualmente o automaticamente (campionamento e selezione delle caratteristiche).

Il più semplice tipo di campionamento è il **Simple random sampling** (campionamento semplice casuale): ogni gruppo di oggetti della dimensione richiesta ha la stessa probabilità di essere il campione selezionato. È possibile ottenere un campione molto atipico, tuttavia le leggi della probabilità impongono che più ampio è un campione, più è probabile che sia rappresentativo della popolazione da cui proviene.

Il metodo tradizionale di scelta di un campione casuale inizia con una numerazione dei membri della popolazione target. L'ordine di numerazione è irrilevante. Una volta che ogni membro della popolazione ha un numero, il campionatore consulta una tabella di numeri casuali per selezionare gli indici dei membri da includere nel campione. Ovviamente il presupposto principale è quello di essere in grado di numerare i membri della popolazione target.

Vi sono due tipi di campionamento semplice casuale:

- con sostituzione
- senza sostituzione

Se la popolazione è grande rispetto al campione, c'è una probabilità molto piccola che qualsiasi membro venga scelto più di una volta e le due tecniche sono essenzialmente le stesse.

Quando la popolazione è divisa in strati o gruppi, è utile invece usare un **campionamento casuale stratificato**. Viene selezionato un campione casule semplice da ciascuno strato separatamente e la loro unione produce un campione stratificato. Un vantaggio di questo campionamento consiste nel fatto che l'analista può controllare il numero di osservazioni all'interno di ogni gruppo o strato e può garantire che particolari gruppi all'interno della popolazione sono adeguatamente rappresentati nel campione. Quando uno strato ha un'appartenenza molto più piccola degli altri, il semplice campionamento casuale può produrre campioni senza rappresentativi di quello strato. La dimensione del campione è solitamente proporzionale alla dimensione relativa degli strati. Tuttavia, questa non è una regola.

Se i membri all'interno degli strati sono più simili tra loro rispetto ai membri di strati diversi, le stime specifiche per strato saranno più precise di quelle dell'intero campione. Attenzione però, è importante adeguarsi alla sovra rappresentazione quando le inferenze si riferiscono alla popolazione target nel suo insieme.

Alcune regole per una buona rappresentazione di progettazione per un campionamento stratificato, sono:

- gli strati devono essere scelti per:
  - avere dei mezzi che differiscono sostanzialmente tra loro
  - minimizzare la varianza all'interno di uno strato e massimizzarla tra i vari strati
- Le dimensioni del campione devono essere proporzionali alla deviazione standard dello strato.

Un'altra tecnica di campionamento è il **cluster sampling** o campionamento clusterizzato, nel quale i membri della popolazione arrivano naturalmente da cluster, ciò rende possibile campionare i cluster. In questo caso tutti i membri di ogni cluster sono considerati. Nel contesto di grandi basi di dati, un'applicazione comune del cluster sampling è di rendere casuale la scelta di blocchi di dati, e successivamente usare tutti i dati nei blocchi. La motivazione dietro questo approccio è che per recuperare un record di database da un blocco particolare, l'intero blocco deve essere letto in memoria. Questo tipo di campionamento è anche chiamato **block scampling**. I vantaggi del cluster sampling riguardano la riduzione dei costi richiesti per accedere ai campioni, e al tempo stesso aumenta la variabilità delle stime campionarie al di sopra di quella del semplice campionamento casuale, a seconda di quanto i cluster differiscono tra loro, rispetto alla variazione all'interno del cluster.

Se i membri di un cluster sono più simili dei membri di cluster diversi, gli approcci statistici che presuppongono che i dati siano indipendenti porteranno a inferenze distorte. La modellazione gerarchica può modellare esplicitamente la struttura indotta dall'amplificazione nei dati.

Il **two-stage sampling** (campionamento a due stadi) combina due idee principali: la celta casuale dei cluster e il campionamento all'interno di ogni cluster.

Quando è possibile numerare in qualsiasi modo gli individui di una popolazione, si può effettuare il **systematic sampling** (campionamento sistematico), conosciuto anche come **every k-th sampling**. Questo tipo di campionamento si sviluppa scegliendo un membro in maniera casuale da quelli numerati tra 1 e k, successivamente include ogni k-th membro dopo il campione. Il vantaggio maggiore di questo tipo di campionamento è la sua facile implementazione,anche nel caso in cui la dimensione della popolazione è inizialmente sconosciuta o il conteggio dei membri della popolazione è computazionalmente costoso. Al contempo bisogna prestare attenzione, dato che la selezione non è casuale, campioni sistematici possono non essere rappresentativi della popolazione e quindi devono essere utilizzati con attenzione. Questo metodo è particolarmente vulnerabile alle periodicità nell'elenco dei membri della popolazione. Se la periodicità è presente e il periodo è un multiplo di k, risulterà una distorsione.

Quando vogliamo organizzare il nostro campionamento basato sul valore di una o pù variabili, ma non conosciamo l'intervallo o la distribuzione di queste variabili nella popolazione target, possiamo avvalerci del **two-phase sampling** o campionamento in due fasi. Un campione iniziale può facilitare prendere decisioni più consapevoli sulle strategie di campionamento da utilizzare. Un campione iniziale può aiutare a determinare la dimensione del campione. I calcoli delle dimensioni del campione spesso richiedono stime di determinati parametri della popolazione, come la forza della relazione tra due variabili. In assenza di conoscenze e/o dati pregressi, un campione iniziale fornirebbe stime per queste quantità e queste stime determinerebbero la dimensione del campione per il campione della seconda fase.

Una domanda che ci si pone spesso è quanto un campione deve essere grande. In statistica vi sono semplici meccanismi per stimare la dimensione del campione necessaria per avere una certa probabilità di rilevare un effetto di una dimensione pre-specificata o superiore. Questi meccanismi, che raggruppati prendono il nome di *analisi di potenza*, si basano su stime delle medie e varianze delle variabili nella popolazione da campionare. E' importante capire il sistema di cause che determinano popolazione e garantire che tutte le fonti di variazione siano prese in considerazione. Un gran numero di osservazioni non ha alcun valore se le principali fonti di variazione vengono trascurate nello studio.

Un'alternativa ad impostare una dimensione del campione predeterminata consiste nel lasciare che i dati "scelgano" la dimensione del campione. L'idea di base è continuare ad aumentare la dimensione del campione fino a quando i risultati o i riepiloghi non cambiano più molto (dove "molto" è impostato in anticipo). Le varianti di questa idea sono note come campionamento progressivo, campionamento adattivo o campionamento sequenziale.

Bisogna però far attenzione al fatto che molti degli strumenti comuni per l'inferenza statistica, inclusi i t-test, presuppongono che i dati comprendano un semplice campione casuale di una popolazione e che i singoli punti dati siano quindi statisticamente indipendenti. Molte delle tecniche di campionamento violeranno questa ipotesi. Esistono tecniche statistiche specializzate per trattare i dati generati da un numero qualsiasi di campionamento casuale da un database fornisce stime imparziali delle caratteristiche del database. Tuttavia, se il database stesso rappresenta un campione casuale o sistematicamente distorto dalla popolazione reale di interesse, nessuna tecnica statistica può salvare le inferenze risultanti.

I database in tempo reale contengono degli attributi (chiamati anche **features** o caratteristiche). Il problema della selezione delle caratteristiche sorge perchè la complessità di ricerca nello spazio delle ipotesi deve essere ridotta per ragioni pratiche, e caratteristiche ridondanti o irrilevanti possono avere effetti significativi sulla qualità dei risultati del metodo di analisi (**maledizione della dimensionalità**). L'idea alla base della maledizione della dimensionalità è che dati di grande dimensione sono difficili da processare, per una serie di motivi: il numero di esempi cresce esponenzialmente con il numero di variabili e non ci sono abbastanza osservazioni per ottenere buone stime.

La **feature selection** è un processo che scegli un subset ottimo di features seguendo alcuni criteri. Vi sono sostanzialmente 3 approcci:

- **wrapper models**
- **filter models**
- **embedded methods**

Il *wrapper model* si basa su un algoritmo di data mining per determinare se un sottoinsieme di funzionalità è valido. L'algoritmo viene utilizzato come parte della funzione di valutazione e anche per indurre i patterns o il modello finale.

L'algoritmo DM può risolvere task predittivi o task descrittivi. Se vogliamo avere un buon insieme di funzionalità per miglio-

rare l'accuratezza di un classificatore, possiamo utilizzare proprio questa misura per basare le evoluzioni del classificatore. Però sorgono diversi problemi, il principale consiste nel determinare veramente l'accuratezza predittiva evitando l'overfitting. Altri problemi consistono nel fatto che un classificatore richiede tempo per apprendere i dati, oppure i dati sono troppo grandi per eseguire un algoritmo di apprendimento, quindi è necessario ridurne la dimensionalità. Vi è quindi la necessità di definire alcuni criteri di stop per garantire che il processo di valutazione termini e che non entri in un loop infinito.

Il filter models è indipendente dall'algoritmo di data mining che sarà utilizzato sul subset di feature. I subset sono valutati utilizzando proprietà intrinseche dei dati quali le misure informative e di distanza. Se consideriamo l'accuratezza stimata da un classificatore come un'altra misura, possiamo unificare i modelli di filtering e wrapper in un modello generale. Ogni componente può avere diverse scelte. Le varie combinazioni di queste scelte sono alla base di molti algoritmi di selezione delle caratteristiche esistenti.

Alcune strategie per la selezione delle caratteristiche dei subset sono:

- **enumerazione** di tutte i possibili subset e selezione dei migliori
- **generazione casuale** dei subset e selezione del migliore.
- **generazione sequenziale** dei subsets.

Possiamo inoltre selezionare i subset attraverso la **foward selection**, iniziando con un subset vuoto e gradualmente aggiungere una caratteristica alla volta, oppure attraverso lla **backward selection**, nel quale si inizia con un insieme completo e si rimuove una caratteristica alla volta. Queste strategie sono basate sulla **strategia di ricerca greedy** in uno spazio di subset grande 2^N, dove N è il numero di caratteristiche.

Indipendentemente dal metodo di generazione dei sottoinsiemi di funzionalità adottato, è necessaria una misura per decidere quale funzionalità deve essere aggiunta o rimossa, oppure quale sottoinsieme deve essere mantenuto. Questo è possibile grazie alla **misura dell'informazione**, data una funzione di incertezza **U** e le probabilità delle classi precedenti $\mathbf{P}(c_i)$, l'informazione ottenuta da una caratteristica X, $\mathbf{IG(X)}$, è definita come la differenza tra la precedente incertezza e l'incertezza posteriore attesa usando X. In formula

$$IG(X) = \sum_i U(P(c_i)) - E[\sum_i U(P(c_i|X)) \tag{1}$$

Una regola di valutazione delle caratteristiche derivata dal concetto di guadagno di informazioni afferma che la caratteristica X è preferita alla caratteristica Y se $IG(X) > IG(Y)$. Cioè, una caratteristica dovrebbe essere selezionata se può ridurre più incertezza. Se U(x)=$-x * log(x)$ allora $\sum_i U(P(c_i))$ è una misura di entropia.

Misure della distanza. Se l'obiettivo è la classificazione, una misura alternativa può essere la distanza tra le funzioni di densità classe-condizione. Se *P(X|C)* è la funzione di densità condizionata dalla classe della caratteristica X, nei due casi della clase (C=c_1 o C=c_2),P($X|C$) è definita da P($X|c_1$) e P($X|c_2$). Se D(X) è la distanza tra P($X|c_1$) e P($X|c_2$), una regola di valutazione basata su P afferma che X è preferito a Y se D(X)>D(Y). Questo perchè stiamo cercando una feature che può separare due classi il più possibile. Maggiore è la distanza, più facile separare le due classi.

La divergenza Kullback-Leibler è una misura della distanza tra distribuzioni probabili, P e Q su un dominio V. Questa divergnza è definita come:

$$m_{KL}(P,Q) = \sum_{v \in V} q(v) log(\frac{q(v)}{p(v)}) \tag{2}$$

Questa divergenza misura in quale misura la distribuzione P è un'approssimazione della distribuzione Q o, più precisamente, la perdita dell'informazione se noi prendessimo P invece di Q. In sintesi si misura quanto P diverge da Q. LE proprietà di questa misura sono:

- Assimetrica, ovvero $m_{KL}(P,Q) \neq m_{KL}(Q,P)$
- non definita quando p(v)=0
- nel caso specifico di q(v)=0, q(v)log(q(v)/p(v))=0
- il range dei valori non è limitato. Quindi, possiamo utilizzare questo valore per stabilire quale delle due distribuzioni Q e Q' è una migliore approssimazione di P, ma non ci permette di determinare in termini assoluti se Q è una buona approssimazione di P osservando $m_{KL}(P,Q)$

La divergenza del χ^2 è definita come:

$$m\chi^2(P,Q) = \sum_{y \in Y} \frac{|p(y) - q(y)|^2}{p(y)} \tag{3}$$

è rigorosamente topologicamente più forte della divergenza KL data la disuguaglianza $m_{KL}(P,Q) \leq m\chi^2(P,Q)$, la convergenza nella funzione di divergenza χ^2 implica la convergenza nella divergenza KL, ma non il contrario. Inoltre è assimetrica e non definita quando p(y)=0.

Un ultimo tipo di misura della distanza eè la distanza di variazione, data dalla formula $m_1(P,Q) = \sum_{y \in Y} |p(y) - q(y)|$, detta anche distanza di Manhattan per funzioni di probabilità p(y) e q(y) e coincide con la distanza di Hamming quando tutte le features sono binari. Similarmente si può utilizzare la distanza Euclidea data da $m_2(P,Q) = \sum_{y \in Y} |p(y) - q(y)|^2$. Queste due metriche soddisfano la proprietà di simmetria e le proprietà metriche.

Dipendenza dalle misure. Si verifica con quanta forza una caratteristica è associata alla classe. Denotando attraverso R(X) una misura di dipendenza tra la feature X e la classe C, preferiamo la feature X alla feature Y se R(X)>R(Y). Un problema con le tre misure precedenti è che non possono rompere i legami tra due features ugualmente buone. Pertanto queste funzionalità non possono rilevare se una di esse è ridondante.

Inconsistenza delle misure. Si tenta di trovare un numero minimo di funzionalità che separano le classi in modo coerente come può fare l'intero set di funzionalità. In altre parole, le misure di incoerenza mirano a raggiungere $P(C|FullSet) = P(C|SubSet)$ Le regole di valutazione delle funzionalità derivate dalle misure di incoerenza affermano che è necessario selezionare il sottoinsieme minimo di funzionalità in grado di mantenere la coerenza dei dati mantenuti dall'insieme completo di funzionalità.

Per **selezionare le feature** ci sono 3 componenti necessari: un generatore di subset, un valutatore e un criterio di stop. Vi sono diversi metodi.

Approcci completi ed esaustivi: il *focus* è applicato su una misura di consistenza e valuta esaustivamente tutti i subset partendo da una feature, mentre il *branch and bound* consiste in una enumerazione sistematica di tutte le soluzioni, dove

ampi sottoinsiemi di candidati infruttuosi sono scartati in massa utilizzando dei limiti superiori e inferiori della quantità da ottimizzare. Inizia con un insieme completo di feature e valuta l'accuratezza stimata.

Approcci Euristici: ***SFS*** (sequential foward search) e ***SBS***(sequential backward search) possono essere applicati a ciascuna delle misure, ***DTM*** è la più semplice versione di una modalità di weapper - impara un classificatore una volta e usa qualsiasi caratteristica trovata nel classificatore.

Approcci non deterministici: ***LVF*** (Las Vegas Filter) e ***LVW*** (Las Vegas wrapper), generano subset di feature casualmente e li testano in maniera differente, LVF applica una misura inconsistente, LVW usa una stima accurata attraverso un classificatore; ***algoritmi generici*** e ***ricottura simulata*** sono anche usati nella selezione di feature. Il primo può produrre più sottoinsiemi, il secondo produce un singolo sottoinsieme.

Approcci basati sulle instanze: ***Relief***, molti piccoli campioni di dati sono campioni proveniente dai dati. Le funzionalità vengono ponderate in base ai loro ruoli nella differenziazione di istanze di classi diverse per un campione di dati. È possibile selezionare funzioni con pesi maggiori.

Per le attività di data mining non di classificazione (nessuna etichetta di classe disponibile), dovrebbero essere presi in considerazione metodi alternativi. Ad esempio, una misura di entropia può essere introdotta per classificare in sequenza le caratteristiche. L'idea di base è che le caratteristiche sono rilevanti se possono descrivere le istanze in termini di cluster relativamente chiaramente definiti.

La **scalabilità** è un altro problema. In LVS la parte più dispendiosa in termini di tempo di un processo di selezione delle funzionalità viene identificata e ritardata fino a quando non è necessario. LVS è un'estensione di LVF che utilizza una misura di incoerenza (IC) con una complessità di runtime di controllo O(n), dove n è il numero di istanze. Se n è enorme, è costoso calcolare IC molte volte. Si noti inoltre che il componente di generazione di sottoinsiemi di funzionalità genererà sempre più sottoinsiemi non validi che non soddisfano IC man mano che la cardinalità di un sottoinsieme valido diminuisce.

Pertanto, ha senso separare il calcolo di IC come cardinalità per tutti i dati dalla generazione di sottoinsiemi di funzionalità. Ma abbiamo bisogno di dati per generare sottoinsiemi di funzionalità. Il compromesso è che invece di utilizzare l'intero set di dati, ne utilizziamo solo una parte per la generazione di sottoinsiemi di funzionalità. Quando un sottoinsieme viene testato come valido sulla porzione di dati, viene calcolato l'IC per l'intero dato. Successivamente se IC è stato soddisfatto, allora la selezione della feature è completata, altrimenti se IC non è stato soddisfatto, le istanze inconsistenti vengono aggiunte alla porzione di dati e viene eseguito un altro ciclo di generazione di sottoinsiemi di funzionalità sulla porzione di dati ingrandita. Questo metodo è particolarmente efficace solo quando n è sufficientemente largo a causa del sovraccarico nel LVS.

I **wrapper models** cercano di risolvere uno specifico problema, quindi il criterio può realmente essere specificato. Invece consuma molto tempo se bisogna valutare uno schema a ogni iterata. I **filter models** sono molto più veloci ma non incorporano algoritmi di data mining usati per la generazione di un modello o pattern, quindi il modello può essere subottimale.

A differenza dei precedenti, nei **metodi embedded** la parte di selezione delle caratteristiche non può essere separata dall'algoritmo di data mining. È parte integrante dell'algoritmo. Per esempio, nell'algoritmo di decision tree, la selezioni delle caratteristiche che contribuiscono alla creazione dell'albero finale è parte della costruzione dell'algoritmo di decision tree. Poiché siamo interessati a selezionare le funzionalità nella fase di trasformazione dei dati, non consideriamo gli embedded methods.

La **pulizia dei dati** aumenta la qualità dei dati al livello richiesto. Ciò comporta la selezione di sottoinsiemi di dati puliti, l'inserimento di impostazioni predefinite adeguate o tecniche più ambiziose come la stima della modellazione dei dati

mancanti. Alla fine di questa fase vi è un rapporto sulla pulizia dei dati, che descrive le decisioni e le azioni per affrontare i problemi di qualità dei dati ed elenca le trasformazioni dei dati per la pulizia e i possibili impatti sull'analisi dei risultati. I due problemi più comuni sono la mancanza di valori e dati di rumore.

I **noisy data**, ovvero tutti quei dati che generano rumore, consistono in variabili che hanno valori non conformi con quelli che ci aspettiamo dalle variabili. Le osservazioni in cui si verificano questi noisy data sono chiamate valori anomali, ovvero (*outliers*). Differenti tipi di outliers possono essere trattati in differenti modi.

Vi possono essere errori umani, i quali possono essere corretti o cancellati dall'analisi, oppure le distribuzioni simmetriche spesso indicano valori anomali. La mancanza di dati, invece, include valori che non sono presenti nei dati selezionati e necessitano di essere eliminati durante il rilevamento del rumore. Questo caso si verifica se viene commesso un errore durante l'inserimento dei dati, oppure se l'informazione non era disponibile al momento dell'inserimento oppure se i dati selezionati all'interno di risorse eterogenee hanno creato dei mismatch.

Per correggere questo tipo di errore vengono eseguite diversi tipi di azioni, l'inserimento di un valore predefinito come il termine "none" è l'ideale. Si potrebbe cancellare le righe che presentano valori mancanti, però questa tecnica, per quanto facile da implementare, può generare la perdita di dati che possono essere valutati.

Un'altra tecnica consiste nell'eliminazione della variabile dall'analisi se presnta un significativo numero di osservazioni con valori mancanti per la stessa variabile. Infine, un'ultima tecnica, consiste nel rimpiazzare il valore mancante con un altro valore, che nel caso di variabili quantitative può consistere nella media o nella mediana, mentre per le variabili categoriche può essere rappresentato dalla moda o dal valore "sconosciuto". Si potrebbe anche pensare di attuare un approccio più sofisticato è quello di predire il valore più probabile delle variabili all'interno delle osservazioni.

Successivamente si passa alla fase di **costruzione dei dati**, la quale include operazioni di preparazione dei dati, come la generazione di variabili derivate, inserimento di nuovi record o trasformazione di variabili esistenti. I dati possono essere trasformati in *una singola variabile*, per essere ulteriormente perfezionati per soddisfare i requisiti del formato di input dei particolari algoritmi di data mining da utilizzare.

Esempi sono la conversione delle variabili di tipo data dal formato US a quello Europeo, oppure il calcolo dell'età data la data di nascita, l'aggregazione dei valori all'interno di un conto corrente per gli ultimi 3,6,12 mesi. Inoltre si potrebbe effettuare un ridimensionamento o una normalizzazione dei dati. In questo caso si parla di normalizzazione dei dati quando colonne numeriche sono trasformate usando funzioni matematiche in dei range. Questo processo è importante perchè tutte le variabili all'interno di una colonna devono essere trattate in maniera uguale e non devono influenzarsi a vicenda, oppure perchè alcuni dati possono ricevere solo alcuni valori all'interno di un range.

Un altro tipo di normalizzazione è la normalizzazione min-max che performa una transformazione lineare sui dati originali. Supponendo che min_A e max_A sono i valori di minimo e di massimo di un attributo A, questa normalizzazione mappa un valore v di A in v' nel range $[newMin_A; newMax_A]$ attraverso la formula:

$$v' = \frac{v - min_A}{max_A - min_A}(newMax_A - newMin_A) + newMin_A \tag{4}$$

Le relazioni tra i valori dei dati originali vengono preservate. Se un caso di input futuro per la normalizzazione non rientra nell'intervallo di dati originale per A, si verifica un errore "out of bounds" (fuori dal limite).

Nella normalizzazione definita z-score, anche chiamata a media zero, i valori per un attributo A sono normalizzati sulla base della media o della deviazione standard di A. Un valore v di A è normalizzato in v' attraverso la formula:

$$v' = \frac{v - mean_A}{standDev_A} \tag{5}$$

dove $mean_A$ e $standDev_A$ sono la media e la deviazione standard dell'attributo A. Questo metodo di normalizzazione è utile quando il minmo e il massimo dell'attributo A sono sconosciuti, o quando ci sono gli outliers che dominano la normalizzazione min-max.

La normalizzazione attraverso il decimal scaling transforma i valori dell'attributo A in punti decimali, per assicurarsi che il range dell'intervallo i cui essi sono compresi sia {-1,+1}. Il numero di punti decimali dipende dal massimo valore assoluto di A. Un valore v di A è normalizzato in v' attraverso la funzione:

$$v' = \frac{v}{10^j} \tag{6}$$

dove j è il più piccolo intero tale che $max(|v'|) < 1$

Si parla di **discretizzazione delle variabili** quando convertiamo variabili quantitative in variabili categoriche dividendo il valore di input in intervalli. Due metodi di discretizzazione sono l'**Equale Width** e l'**Equal depth**. In entrambi questi metodi le informazioni sulla classe non vengono utilizzate nel caso in cui le osservazioni siano preclassificate, inoltre la discretizzazione viene applica a ogni attributo indipendentemente dagli altri.

Il partizionamento attraverso l'equal width divide il range in N intervalli di uguale lunghezza, se A e B sono il più piccolo e il più grande valore all'interno dell'attributo, l'intervallo di width sarà:

$$W = \frac{B - A}{N} \tag{7}$$

Questo metodo è il più diretto, però ha anche degli aspetti negativi perchè non gestisce bene i dati distorti ed è sensibile ai valori anomali.

Il partizionamento attraverso l'equal-depth divide il range in N intervalli, ognuno contenente approssimativamente lo stesso numero di esempi. Questo tipo di partizionamento ha una buona scalabilità, gestisce gli attributi categorici attraverso alcuni trucchetti, minimizza le informazioni perdute durante il processo di partizionamento.

In entrambi i casi il numero di elementi tralasciati è definito dall'utente. Nel caso di classificazione dei dati, è buona pratica che questo numero non sia minore del numero di classi che vogliamo riconoscere, oppure determinarlo attraverso la formula:

$$N_{bins} = \frac{M}{3 * C} \tag{8}$$

dove M è il numero di esempi di training e C il numero di classi.

Procedure complesse di conversione di dati hanno l'obiettivo di costruire un piccolo insieme di indici da un vasto numero di variabili, in modo tale che solo alcune informazioni vengano perse e il numero totale di funzioni venga ridotto. **L'analisi fattoriale** e **l'analisi del componente principale** sono due tecniche di analisi multivariate per la riduzione dei dati.

L'analisi fattoriale affronta il problema dell'analisi della struttura delle interrelazioni (correlazioni) tra un grande numero di variabili (ad es. punteggi dei test, elementi del test, risposte al questionario) definendo un insieme di dimensioni sottostanti

comuni, note come fattori. Questa non è una tecnica che usa dipendenze, ovvero non vi è nessuna variabile considerata come criterio o variabile dipendente da cui tutte le altre dipendono e sono le variabili predittore o indipendenti. E' una tecnica interdipendente in cui ogni variabile è considerata simultaneamente e ognuna è in relazione alle altre.

Oltre alla costruzione dei dati, vi è l'integrazione dei dati, che ha come obiettivo la combinazione di informazioni provenienti da tabelle multiple o record per creare nuovi record o valori. L'output di questa fase è un insieme di dati uniti, che si riferisce all'unione di due o più tabelle unite che contengono differenti informazioni sugli stessi oggetti, o dati aggregati, sui quali verranno effettuate delle operazioni per produrre nuovi dati. Il risultato è il modello dei dati analitici, che rappresenta una ristrutturazione consolidata, integrata e dipendente dal tempo dei dati selezionati e pre-elaborati dalle varie fonti operative ed esterne.

Una decisione importante presa in questo compito riguarda l'unità di analisi. Nella ricerca nelle scienze sociali, le unità di analisi più tipiche sono le persone individuali. Altre unità di analisi possono essere gruppi, manufatti (libri, foto, giornali), unità geografiche (città, censimento, stato), interazioni sociali (relazioni diadiche, divorzi, arresti). L'unità di analisi non deve essere confusa con le unità di osservazione, che è l'unità su cui vengono raccolti i dati, cioè vengono effettuate osservazioni sistematiche. Ad esempio, uno studio può avere un'unità di osservazione a livello individuale (ad es. alunni), ma può avere l'unità di analisi a livello di gruppo (ad es. una classe). L'unità di osservazione è la stessa dell'unità di analisi quando le generalizzazioni ricavate da un'analisi statistica sono attribuite all'unità di osservazione (cioè gli oggetti su cui i dati sono stati raccolti e organizzati per l'analisi statistica).

Un'altra procedura consiste nel formattare la sintassi di alcuni tipi, non andando ad alterare i significati, perchè questi formati sono richiesti da altri modelli.

1.1.4 Modeling

In questa fase viene scelta la tecnica di modellazione tra le tecniche e gli strumenti preselezionati, come output si hanno le tecnice e le assunzioni modellate dalle tecniche prese in input. Ci sono diversi fattori su cui basare la scelta dell'algoritmo o degli algoritmi per modellare i dati, tra cui la natura dell'obiettivo, la capacità di gestire determinati tipi di dati, la capacità di gestire multiple relazioni e generare pattern relazionali, la scalabilità o il livello di famigliarità.

Si possono identificare tre componenti primari in qualsiasi algoritmo di data mining:

- **modello di rappresentazione**: relazionale o proposizionale, qualitativo o quantitativo, adeguatezza per un utente.
- **modello di valutazione**: l'incertezza del modello si base su stime probabilistiche, test statistici ecc.
- **ricerca**

Il *modello di rappresentazione* consiste nel linguaggio L usato per descrivere patterns che possono essere scoperti. Dipende dal tipo di rappresentazione, la capacità descrittiva, l'adeguatezza per un dato utente. La conoscenza scoperta può essere categorizzata dal tipo di pattern dei dati. Una scoperta quantitativa mette in relazione valori di campi numerici usando equazioni matematiche, mentre una relazione qualitativa mette in relazione logica diversi campi. Spesso si possono trovare queste due relazioni combinate. Un altro aspetto che bisogna considerare nella creazione del modello è la sua rappresentazione che deve essere appropriata per l'utente previsto.

La maggior parte degli utenti per i quali bisogna creare una rappresentazione sono gli umani, programmi per computer e sistemi di scoperta. Se si parla di utenti umani bisogna utilizzare un linguaggio naturale, molto utile ma non conveniente per

la manipolazione da parte di algoritmi di scoperta, oppure si potrebbe utilizzare un linguaggio logico, molto più utile per la computazione e se necessario può essere anche traslato in un linguaggio normale. Le informazioni sulla forma e la densità dei gruppi di record sono un altro tipo di conoscenza che si presenta al meglio visivamente per mezzo di grafici bidimensionali o tridimesionali, quindi attraverso opportune rappresentazioni visive.

Nel caso in cui gli utenti finali siano dei programmi, la rappresentazione include linguaggi di programmazione e formalismi dichiarativi. Quando invece l'utente è un sistema il cui compito è scoprire conoscenza, le nuove scoperte vengono reinserite nel sistema come conoscenza di dominio, la conoscenza di dominio e la conoscenza scoperta devono condividere una rappresentazione comune. Se la rappresentazione è limitata, nessun addestramento o nessun tipo di esempio produrrà un modello accurato per i dati. È importante che un analista di dati comprenda appieno i presupposti rappresentativi che possono essere inerenti a un particolare metodo.

Alcuni patterns sono spesso associati a un grado di incertezza, rappresentata attraverso la probabilità, la deviazione standard, misure di credenza o fuzzy sets. In maniera visiva l'informazione incerta può essere convertita in dimensione, densità e ombreggiatura.

Quando i database sono veramente grandi, con milioni di record, un'analisi completa di tutti i dati è infattibile. Gli algoritmi di rilevamento devono quindi basarsi su una qualche forma di campionamento, per cui viene considerata una parte dei dati. Le scoperte risultanti in questi casi sono necessariamente incerte. Le tecniche statistiche possono misurare il grado di incertezza. Possono anche essere utilizzati per determinare la quantità di campionamento aggiuntivo necessaria per ottenere il livello di fiducia desiderato nei risultati.

Il modello di valutazione stima l'incertezza di un particolare pattern. Ci sono diversi metodi per valutare questa incertezza. La **cross validation** è una di queste procedure che consiste nel mescolare i dati del dataset, partizionarli in k sottoinsiemi di uguale lunghezza n, per ogni sottoinsieme chiama i-esimo subset di n oggetti come insieme test e lo mette da parte, addestra il sistema sui rimanenti sottoinsiemi e testa il sistema sull'insieme di test e memorizza le performances, successivamente pulisce la memoria dimenticando ciò che ha imparato durate l'addestramento, calcola le performance medi di ogni insieme di test.

La scelta dell'algoritmo di data mining è fondamentale per conoscere se la ricerca dovrebbe essere performata nello spazio dei parametri o nello spazio dei modelli o entrambi. Se non è possibile una soluzione, si possono utilizzare metodi iterativi greedy.

La ricerca del modello avviene come un ciclo sul metodo di ricerca dei parametri, la cui rappresentazione viene modificata in modo da considerare una famiglia di modelli. Per ogni rappresentazione specifica del modello viene istanziata la modalità di ricerca dei parametri per valutare la qualità di quel particolare modello. Le implementazioni dei metodi di ricerca del modello tendono a utilizzare tecniche di ricerca euristica poiché la dimensione dello spazio dei possibili modelli vieta la ricerca esauriente e le soluzioni in forma chiusa non sono facilmente ottenibili.

Prima di costruire i modelli, è importante generare una procedura per testare la loro qualità. Successivamente la tecnica di modellazione è applicata ai dati e alla fine della quale vengono descritte le varie motivazioni delle scelte effettuate, le impostazioni dei parametri e l'output model con accuratezza aspettata, la robustezza e le possibili carenze.

L'output dell'algoritmo di data mining può essere espresso seguendo alcuni standard industriali. Il **Predictive Model Markup Language** (PMML) è uno standard basato su XML, il cui obiettivo è quello di definire e condividere modelli predittivi usando uno standard aperto. Un complesso mosaico di applicazioni software generano e consumano conoscenza, quindi si ha la necessità di una rappresentazione indipendente dal fornitore di output di data mining.

Alcuni benefici di questo standard consistono nell'eliminare i problemi e le incompatibilità di software proprietari per lo scambio di modelli tra applicazioni, oltre alla facilitazione nello sviluppo di modelli utilizzando qualsiasi fornitore e maggiore facilità nella distribuzione dell'applicazione.

Alla fine della creazione del modello, bisogna valutarlo. I risultati sono interpretati in base ai criteri di successo dell'algoritmo di data mining. Alcuni tipici criteri sono l'accuratezza del modello, quanto esso riesce a descrivere i dati osservati, quanta confidenza può essere riposta nella predizione. Alla fine di questa fase viene effettuata una valutazione dei modelli generati e revisionati i parametri delle impostazioni.

I modelli predittivi vengono valutati in base alla loro accuratezza su dati non visti in precedenza. La valutazione del modello può avvenire a livello dell'intero modello o a livello di singole previsioni. Due modelli con la stessa accuratezza complessiva possono avere livelli di varianza abbastanza diversi tra le singole previsioni. La valutazione del modello dovrebbe basarsi su un set di test indipendente dal set di formazione.

Per i modelli di classificazione, la stima dell'accuratezza è il numero complessivo di classificazioni corrette, diviso per il numero totale di campioni nel test set. Il tasso di errata classificazione (o errore) è il complemento della stima dell'accuratezza. La precisione potrebbe non essere appropriata quando gli errori possono avere costi (e conseguenze) diversi. In tal caso, un costo di errore è una misura migliore di un errore di classificazione errata. Per i modelli di regressione o per i modelli di stima dei parametri, l'accuratezza è stimata come la somma media degli errori quadrati (differenze tra valori previsti ed effettivi) (errore quadratico medio).

1.1.5 Valutazione

Mentre la valutazione del modello si occupa di fattori quali l'accuratezza e la generalità dei modelli, questa attività valuta i modelli in base agli obiettivi di business originali e ai criteri di successo. La sfida consiste nel presentare le nuove scoperte in modo convincente e orientato al business. Questa attività può essere eseguita correttamente da un analista di dati esperto che lavora con un analista aziendale.

Come output di questa operazione abbiamo una valutazione complessiva del data mining rispetto ai criteri di successo aziendale, inclusa una dichiarazione finale sul fatto che il progetto soddisfi già gli obiettivi aziendali iniziali. Questo passaggio rientra nel dominio dell'analista aziendale ed è focalizzato sullo sponsor esecutivo. L'analista aziendale esperto sarà in grado di formulare i risultati in un modo che si collega direttamente agli obiettivi aziendali che erano stati fissati per il progetto all'inizio. Inoltre, un analista di dati con una vasta conoscenza del settore e/o del settore può essere una grande risorsa in questa fase.

Il più grande vantaggio del progetto potrebbe essere un riconoscimento o una rinnovata attenzione all'importanza del patrimonio di dati aziendali e alla potenza delle soluzioni basate sui dati. Questo a sua volta può generare una cultura all'interno dell'organizzazione in grado di promuovere iniziative più ampie, a lungo termine e orientate ai dati, come il data warehousing e il data marting.

Oltre alla valutazione del modello, viene effettuato un processo di revisione nel quale viene considerato l'intero processo al fine di determinare se vi è qualche fattore o compito importante che viene trascurato o per identificare una procedura generica per generare modelli simili in futuro.

1.1.6 Deployment

Per distribuire i risultati del data mining nell'azienda, questa attività sviluppa una strategia per il deployment.In questa fase si pianifica il piano per il deployment, considerando anche l'infrastruttura tecnica che deve essere impiegata per il corretto funzionamento dell'applicativo sviluppato. Successivamente a questa fase viene creato un piano per il monitoraggio e il mantenimento dell'applicazione. Viene redatto un report del prodotto finale e in fase di revisione si valut cosa è andato bene, cosa è andato male e cosa bisogna migliorare.

2 Learning sets of rules

2.1 Classificatore basato su regole

Un classificatore è espresso in uno stile logico costituito da un insieme di regole di decisione (if-then), che rappresentano la conoscenza nel miglior modo possibile in termini di espressione e comprensione. Vi sono due tipologie di regole: le **regole per l'apprendimento di attributi** e le **regole per la descrizione relazionale**.

Nel **concept learning**, apprendimento dei concetti, si hanno due punti di vista opposti: *estensionale*, quando si ha un insieme di oggetti fisici o astratti, o *intenzionale*, quando si hanno un insieme di condizioni sufficienti e necessarie. In entrambi i casi abbiamo un insieme di condizioni sufficienti da un insieme di esempi di concetti positivi e negativi.

Un ipotesi h è una congiunzione di vincoli sugli attributi di istanza. Ogni costrutto può essere uno specifico valore, senza un valore attribuito o non importante. Una definizione formale di ipotesi può essere:

Data un'istanza X, un insieme di esempi D $< x_i, c(x_i) >$, un concetto target c, delle ipotesi H espresse come congiunzione di costrutti sugli attributi, determina un ipotesi h in H come $h(x) = c(x)$ per ogni x in X. Il task di apprendimento è volto a determinare un ipotesi h identica al concetto di target c sull'intero insieme di istanze X.

Ogni ipotesi cerca di approssimare la funzione target su un insieme sufficientemente largo d esempi di training che sarà a sua volta l'approssimazione della funzione target sugli esempi non osservati. L'apprendimento di concetti può essere visto con un task i ricerca su un largo spazio di ipotesi H. L'obiettivo della ricerca è di trovare l'ipotesi che misura l'insieme di training. Lo spazio di ipotesi è implicitamente definito dalla rappresentazione dell'ipotesi.

Date due ipotesi h_k e h_j, h_j è più generale o uguale a h_k quando $h_j \geq h_k$ se e solo se ogni istanza che soddisfa h_k soddisfa anche h_j. Inoltre possiamo definire che h_j è strettamente più generale rispetto a h_k quando $h_j > h_k$ e se e solo se $h_j \geq h_k$ e $h_k \not\geq h_j$.

Un'ipotesi h **copre** un esempio positivo se è correttamente classifica l'esempio come positivo. Qundi si avrà la relazione $h(x) = c(x) = 1$. Un esempio $< x, c(x) >$ **soddisfa** le ipotesi h quando $h(x) = 1$ indipendentemente dal fatto che x sia un esempio negativo o positivo del concetto di destinazione. Mentre definiamo che un'ipotesi h è **consistente** con un esempio $< x, c(x) >$ quando h(x)=c(x). Infine definiamo un ipotesi h **consistente con un insieme di dati** D se è consistente con ogni esempio $x \in D$.

Per sfruttare *l'ordinamento dal generale allo specifico*, si potrebbe iniziare con l'ipotesi più specifica in H, per poi generalizzare questa ipotesi ogni volta che essa non riesce a coprire un esempio di addestramento positivo osservato (**bottom-up search**).

Per aiutarci in questo arduo compito descriviamo il procedimento dell'algoritmo **FIND-S**. Inizializziamo h all'ipotesi più specifica in H. Successivamente definiamo due cicli, quello più esterno per ogni istanza positiva x dell'insieme di traning esegue il ciclo su ogni attributo della condizione a_i in h. Se il vincolo a_i è soddisfatto per x, allora non si fa nulla, altrimenti si rimpiazza a_i in h con il prossimo vincolo più generale soddisfatto da x. Alla fine dell'algoritmo viene restituita l'ipotesi h.

In questo algoritmo non vi è nessuna revisione in caso di esempio negativo perchè si assume che il target concept c si trova in H e non vi sono errori nei dati di training. L'ipotesi h è l'ipotesi più specifica in H tale per cui $c \geq_g h$, ma c non sarà mai sodisfatta da un esempio negativo.

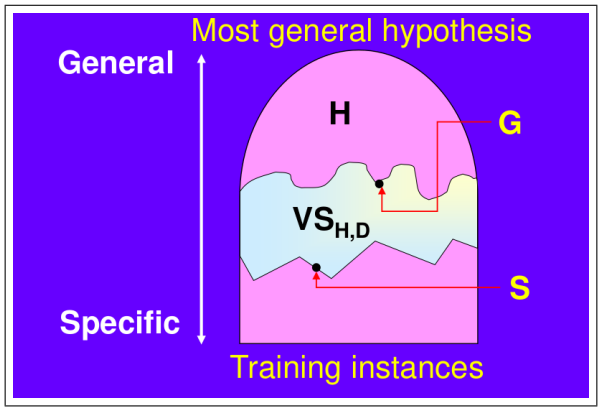
L'algoritmo Find-S ha anche degli aspetti negativi. Non ci dice se l'apprendimento ha coperto il corretto target concept, se c'è solo un'ipotesi consistente in H con i dati o ci sono più di un'ipotesi. Inoltre non possiamo rilevare quando i dati di training sono inconsistenti, in quanto l'incoerenza negli esempi di addestramento può fuorviare Find-S, poiché ignora gli esempi negativi. Potrebbero esserci diverse ipotesi coerenti massimamente specifiche. L'algoritmo dovrebbe fare marcia indietro sulle sue scelte per esplorare un ramo diverso dell'ordinamento parziale rispetto al ramo che ha selezionato.

Da qui nasce l'idea del **version space**. Si decide di restituire uno spazio delle versioni invece di una singola ipotesi. Definiamo formalmente questo version space $VD_{H,D}$ come uno spazio contenente le rispettive ipotesi dello spazio H a dei dati di esempio D. In questo mod esso risulta essere un sotto insieme di H consistente con gli esempi di traning in D. Ovvero: $VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$

Dalla definizione di questo spazio nasce l'algoritmo **List-Then-Eliminate**, nel quale in un primo passo si assume che $VS_{H,D}$ è una lista contenente ogni singola ipotesi in H. Successivamente per ogni esempio $< x, c(x) >$ rimuove da $VS_{H,D}$ ogni ipotesi h per la quale h(x)≠c(x). Infine si restituisce lo spazio rimanente.

Anche questo algoritmo ha i suoi pro e i suoi contro. Certamente garantisce come output tutte le ipotesi consistenti con i dati di training e può trovare l'inconsistenza all'interno dei dati. Si potrebbe anche effettuare una enumerazione di tutte le ipotesi possibili sono per spazi finiti di H o per grandi spazi irrealistici di H.

Il version space può essere rappresentato dai suoi membri più generali e meno generali come in figura.



Il confine generale, o **general boundary** G, rispetto allo spazio delle ipotesi H e ai dati di addestramento D, è l'insieme dei membri massimamente generali di H consistente con D. $G \equiv \{g \in H | Consistent(g, D) \wedge (\neg \exists g' \in H[(g' >_g g) \wedge Consistent(g', D)])\}$

Il confine specifico, o **specific boundary**, S, rispetto allo spazio delle ipotesi H e all'insieme dei dati D, è l'insieme dei membri generali di H consistenti in D. $S \equiv \{s \in H | Consistent(s, D) \wedge (\neg \exists s' \in H[(s >_g s') \wedge Consistent(s', D)])\}$

Una volta ottenuti questi due insiemi, possiamo definire un algoritmo per eliminare le ipotesi che non sono generali. Questo algoritmo prende il nome di **Candidate-Elimination**, nel quale in un primo passo si assegna a una varibile G le ipotesi generali in H e in una variabile S si assegnano le ipotesi specifiche in H. Successivamente per ogni esempio di traning d, **(controllo1)** si controlla se quest'ultimo è un esempio positivo e nel caso si rimuove da G ogni ipotesi inconsistente con d, per poi avviare una funzione UPDATE-S.

La funzione UPDATE-S ha il compito di controllare ogni ipotesi s in S che non è consistente con d, eliminando s da S, per poi aggiungere a S tutte le generalizzazioni minimali h di s tali che h è consistente con d e qualche membro di G è più generale

di h. Infine rimuove da S ogni ipotesi che è più generale di un'altra ipotesi in S.

Si ritorna al **controllo1** e se questo risultasse negativo, si rimuove da S ogni ipotesi inconsistente con d e si avvia una procedura UPDATE-G per aggiornare l'insieme G. Quest'ultima funzione controlla ogni ipotesi g in G che non è consistente con d e in questo caso rimuove g da G, aggiunge a G tutte le specializzazioni minimali h di g tali che h è consistente con d e qualche mebro di S è più specifico di h. Infine rimuove da G ogni ipotesi che è mmeno generale di altre ipotesi in G.

L'algoritmo *candidate-elimination* convergerà verso il target concept a condizione che non ci sono errori negli esempi di training ed è presente qualche ipotesi in HS che descrive correttamente il target concept. Infatti il target concept è appreso quando il confine di S e G converge a un'ipotesi singola e identica.

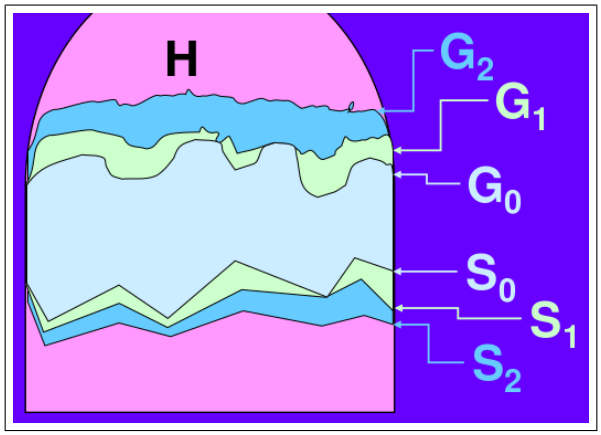
L'algoritmo genera un un version space vuoto quando l'insieme dei dati di traning contiene errori e il target concept non può essere descritto attraverso ipotesi rappresentative (*hypothesis representation*). Inoltre possiamo dire che l'algoritmo di Candidate-Elimination performa un ricerca bidirezionale, in quanto G e S possono crescere esponenzialmente nel numero di esempi di training.

Quando ci troviamo difronte a concetti appresi parzialmente, abbiamo una minore confidenza nella classificazione in casi ambigui, ma potremmo attuare un approccio basato sul voto maggiore. In questo approccio assumiamo che tutte le ipotesi in H sono uguali a priori, successivamente potremmo decidere in base:

- al maggior numero di voti, i quali forniscono la classificazione più probabile per la nuova istanza,
- alla percentuale di ipotesi che votano positivamente, interpretando questo come la probabilità che l'istanza sia positiva dato il dato di training.

In questo ci viene in aiuto un algoritmo di *interactive learning* che sceglie la nuova istanza (**query**) e restiutisce la corretta classificazione mediante un oracolo esterno. Se l'algoritmo sceglie sempre una query che è soddisfatta solo da metà delle ipotesi in VS, allora il target concept può essere trovato in $log_2|VS|$ steps.

Un problema da affrontare in questi tipi di problemi consiste nella gestione del rumore nelle istanze. Per poterlo gestirre si potrebbero rendere più blande le condizioni che descrivono la coerenza dei concetti con tutte le istanze di addestramento. Inoltre se si hanno un numero limitato e predeterminato di esempi classificati in modo errato, si potrebbero mantenere diversi insiemi G e S di consistenza variabile.



L'insieme S_i è coerente con tutti tranne gli i esempi di allenamento positivi. L'insieme G_i è coerente con tutti tranne gli i

esempi di allenamento negativi. Quando G_0 incrocia S_0 l'algoritmo può concluedere che nessun concept nello spazio delle regole è coerente con tutte le istanze di assestramento. L'algoritmo può recuperare e provare a trovare un concetto coerente con tutti gli esempi di addestramento tranne uno.

Una fondamentale propietà dell'inferenza deduttiva afferma che: "un algoritmo di apprendimento che non crea nessuna assunzione a priori riguardante l'identità del target concept, non ha basi relazionali per qualsiasi istanza invisibile". Questa proprietà prende il nome di **inductive bias** che può essere definita formalmente come segue.

Definizione formale di inductive bias. Consideriamo:

- un algoritmo L di concept learning
- un insieme X di tutte le istanze
- un target concept c
- degli esempi di training $D_c = \{< x, c(x) >\}$
- denotiamo con $L(x_i, D_c)$ la classificazione assegnata all'istanza x_i da L dopo l'allenamento sui dati D_c

L'inductive bias di L è qualsiasi insieme minimo di asserzioni B tale che per qualsiasi target concept c e corrispondenti esempi di addestrammento D_c: $(\forall x_i \in X)[(B \wedge D_c \wedge x_i) \rightarrow L(x_i, D_c)]$. Alcuni sistemi induttivi di modellazione possono essere espressi mediante sistemi deduttivi equivalenti. Il cosidetto inductive bias è un input esplicito begli algoritmi dei sistemi deduttivi.

Analizziamo l'introduzione dell'inductive bias all'interno dell'algoritmo del candidate-elimination. Il target concept c è contenuto nello spazio delle ipotesi H. Da $c \in H$ segue deduttivamente $c \in VS_{H,D}$. In questo algoritmo L restituisce la classificaziione $L(x_i, D_c)$ se e solo se ogni ipotesi in $VS_{H,D}$ produce anche questa classificazione, includendo anche le ipotesi $c \in VS_{H,D}$ (inductive bias). Quindi $c(x_i) = L(x_i, D_c)$.

Il bias induttivo è un mezzo non procedurale per caratterizzare la politica degli algoritmi di apprendimento per generalizzare oltre i dati osservati. Confrontiamo diversi algoritmi di apprendimento in relazione all'inductive bias.

Nell'algoritmo **Rote learning**, si memorizzano gli esempi, viene classificato x se corrisponde a un esempio osservato in precedenza. In questo tipo di algoritmo non c'è nessun inductive bias. Per quanto riguarda l'algoritmo **candidate elimination**, l'inductive bias consiste nel target concept rappresentato dal suo spazio di ipotesi. Mentre per quanto riguarda l'algoritmo **Find-S**, l'inductive bias è il target concept rappresentato dal suo spazio di ipotesi e da tutte le istanze negative a meno che il contrario non sia implicato dalla sua altra conoscenza (una sorta di ragionamento predefinito o non monotono).

Per apprendere concetti disgiunti si può utilizzare l'algoritmmo Candidate-Elimination, il quale è un aloritmo con il minimo impegno, che generalizza solo quando è forzato a farlo. Però la disgiunzione fornisce un modo per evitare qualsiasi generalizzazione, per uci l'algoritmo non è mai costretto a farlo. Per apprendere i concetti disgiuntivi, bisogna trovare un metodo per controllare l'introduzione di disgiunzioni, in modo da prevenire disgiunzioni banali. La copertura sequenziale, o **sequential covering**, è un approccio diffuso all'apprendimento di serie di regole.

Vi sono inoltre particolari algoritmi di copertura sequenziale che performano ripetutamente l'algoritmo di candidate-elimintaion per la ricerca di diverse descrizioni congiuntive che insieme coprono tutte le istanze di training. Ad ogni iterata, si trova un concetto congiuntivo che è coerente con alcuni degli esempi di training positivi e tutti gli esempi di training negativi. I casi

positivi che sono stati presi in considerazione vengono rimossi da ulteriori considerazioni e il processo viene ripetuto finché tutti gli esempi positivi non sono stati coperti.

Uno pseudo-codice dell'algoritmo di copertura sequenziale è il seguente:

Sequential-Covering(Target-attribute,Attributes,Examples,Threshold)
Learned-rules ← ∅
Rule ← LEARN-ONE-RULE (Target-attribute,Attributes,Examples)
while PERFORMANCE(Rule, Examples) > Threshold do:
ciao

All'interno dell'algoritmo di copertura sequenziale viene utilizzato un particolare algoritmo chiamato **LEARN-ONE-RULE**. Questo particolare algoritmo accetta un insieme di esempi di traning sia positivi che negativi e restituisce una singola regola che copre tutti gli esempi. Questo algoritmo ha il pregio di essere molto accurato, ma non ha un'alta copertura. Generalmente