UNIVERSITÀ DEGLI STUDI DI NAPOLI "FEDERICO II"

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

TESI DI LAUREA

ANALISI ED ESEMPI DI APPLICAZIONE DEI PRINCIPALI STRUMENTI DI DATA MINING PER L'ESTRAZIONE DI CONOSCENZA DA DATI ETEROGENEI

RELATORE
Ch.mo Prof. Ing.
Antonio d'Acierno

CANDIDATO
Lucia Caporaso
Matricola 41/1758

ANNO ACCADEMICO 2003-2004

A mia nonna,

la persona più forte, testarda e determinata

che abbia mai conosciuto,

a lei

che non ha mai perso la pazienza

e la forza di lottare,

che mi ha insegnato a guardare avanti,

a credere in ciò che facevo e

a non arrendermi mai.

Grazie.

Indice

INDICE ..... **I**
RINGRAZIAMENTI ..... **III**
PREFAZIONE ..... **V**

1. KNOWLEDGE DISCOVERY IN DATABASES E DATA MINING **1**
1.1 Uno sguardo ravvicinato al DM **2**
1.2 Il processo KDD **4**
1.3 Uno standard per il DM: il modello CRISP-DM **9**
1.4 Verification models e Discovery models: i due diversi aspetti del DM previsivo **11**

2. STRATEGIE DI DM SUPERVISIONATO: LINK ANALYSIS E ASSOCIATION DISCOVERY **14**
2.1 Strategie di DM **15**
2.2 Classificazione e raggruppamento: principali differenze **17**
2.3 La Link analysis **19**
2.4 Confidenza, supporto e lift **21**
2.5 L'algoritmo Apriori per la ricerca delle regole **24**
2.6 Regole interessanti e non **26**
2.7 Sequence discovery **27**
2.8 Conclusioni **28**

3. STRATEGIE DI DM SUPERVISIONATO: ALBERI DECISIONALI **29**
3.1 Gli alberi di decisione: concetti preliminari **30**
3.2 Costruzione degli alberi di decisione **30**
3.3 Criteri per la costruzione degli alberi: entropia ed information gain **32**
3.4 Algoritmi per la costruzione di alberi decisionali **35**
3.5 Algoritmo CART **36**
3.6 Algoritmo C4.5 ..... **39**
3.7 Il problema dei test **41**
3.8 Accuratezza degli alberi di decisione **42**
3.9 Come limitare l'overfitting **43**
3.10 Generazione delle regole produttive **44**
3.11 Considerazioni finali **45**

4. STRATEGIE DI DM NON SUPERVISIONATO: IL CLUSTERING **46**
4.1 Introduzione alle tecniche di clustering **47**
4.2 L'algoritmo delle k-medie **49**
4.3 Trattamento delle variabili categoriali **52**
4.4 L'algoritmo dei k-prototipi **56**
4.5 Il clustering gerarchico **59**
4.6 Il clustering basato sulla densità **63**
4.7 Gli algoritmi genetici **65**
4.8 Gli operatori genetici **67**
4.9 Algoritmi genetici e clusterizzazione **73**
4.10 Considerazioni finali **76**

5. APPLICAZIONI AVANZATE DI DM **78**
5.1 I confini del DM **79**
5.2 Il Web mining **79**
5.3 Preparazione dei dati **81**

5.4 Data mining sul web: interpretare e valutare i risultati 83
5.5 Data mining per valutare i siti web 85
5.6 Data mining per personalizzare i siti web 86
5.7 Data mining per adattare i siti web 88
5.8 Text mining 89
5.9 Applicazioni del text mining: analisi della letteratura biomedica 90
6. TECNICHE FORMALI DI VALUTAZIONE DEI MODELLI 97
6.0 Introduzione 98
6.1 Valutare i modelli di apprendimento supervisionato: le matrici di confusione 98
6.2 Cosa dovrebbe essere valutato ? 100
6.3 Parametri statistici ad un solo valore 101
6.4 Distribuzione normale e media campionaria 102
6.5 Un modello classico per testare le ipotesi 104
6.6 Calcolare gli intervalli di confidenza per i test set 107
6.7 Confronto dei modelli supervisionati 111
6.8 Tecniche di valutazione non supervisionata 113
6.9 Metodi aggiuntivi per la valutazione della clusterizzazione 115

7. IL SOFTWARE WEKA 117
7.1 Introduzione a Weka 118
7.2 Il formato ARFF 120
7.3 L'ambiente Explorer 122
7.4 Primo caso d'uso: un esempio di analisi tassonomica in botanica 127
7.5 Secondo caso d'uso: analisi del dataset Pima-diabetes 145
7.6 Validità del test diagnostico: conclusioni 165

CONCLUSIONI 167

BIBLIOGRAFIA 168

Ringraziamenti

Ringrazio il professor Antonio d'Acierno per la disponibilità, la comprensione e la cordialità che ha sempre mostrato nei miei confronti durante tutto il periodo di realizzazione di questo lavoro di tesi.

Un ringraziamento particolare è rivolto ai miei genitori, che mi hanno incoraggiato sempre nel lungo percorso di studi; un grazie sentito a mio padre per aver sempre creduto in me, per la serenità e la pace che mi ha sempre trasmesso, a mia madre per le cose belle in cui mi ha insegnato a credere.

Grazie a Nicola che mi ha sempre fatto sorridere e per la positività che mi ha regalato in ogni occasione.

Un grazie urlato a Sandra, che anche se fisicamente lontana, nell'ultimo mese mi ha particolarmente aiutato e mi è sempre stata vicina; non è cambiato niente anche se sei lontana!

Un grazie speciale a Ilaria, che è sempre stata presente nella mia vita, che è sempre riuscita a scorgere la voce dei miei silenzi, per il suo aiuto morale, per avermi sopportato ed aiutato a uscire fuori da situazioni difficili, per aver partecipato a tutti i momenti, brutti e belli, per il suo voler condividere tutto.

Grazie a Michele, l'amico che tutti vorrebbero avere e mai perdere.

Grazie a zia Lina, a zia Maria, alle mie cugine Assia e Francesca per la loro disponibilità e l'aiuto immenso che hanno sempre apportato a me e alla mia famiglia in momenti difficilmente superabili.

Ringrazio infine Mario, la persona che più amo al mondo e che, più di tutti, mi ha incoraggiato, mi è stata sempre vicina ed allo stesso tempo ha saputo mettersi da parte nell'ultimo periodo, che ha rappresentato per me un sicuro punto di riferimento e che mi ha regalato momenti indimenticabili e una felicità immensa.

Dedico questo lavoro di tesi a mia nonna, che, anche se non può festeggiare qui con me questo importante traguardo raggiunto, se fosse stata presente, sarebbe stata più felice di me: la prima ad applaudirmi!

Il vero viaggio di scoperta non consiste nel
cercare nuove terre ma nell'avere nuovi occhi.

Voltaire (1731)

Prefazione

Il concetto di DM non è ancora ben definito infatti gli sviluppi recenti di tale tecnica e gli ampi campi di applicazione, che spaziano dalle diverse tipologie di business e del marketing strategico alle applicazioni biomediche e biologiche, alla market basket analisys, alla individuazione di truffe in telefonia cellulare o in carte di credito, al web mining e text mining, fanno si che il DM risulti spesso un concetto piuttosto vago, difficile da definire e da delimitare, di cui si parla tanto senza sapere precisamente di cosa si tratti.

Le discipline scientifiche che hanno determinato, e tuttora determinano, l'evoluzione della teoria e delle pratiche del DM sono l'informatica, con particolare riferimento all'organizzazione e alla gestione delle basi di dati classiche e al data warehouse nonchè a metodologie proprie dell'intelligenza artificiale (machine learning, pattern recognition); la statistica metodologica, con particolare riferimento alla statistica computazionale; le discipline economico-aziendali, specialmente nell'ambito del marketing e dell'organizzazione aziendale.

Per definire sinteticamente il data mining, senza creare ulteriore confusione, la soluzione migliore è riportare alcune definizioni la cui combinazione può permettere di estrapolare in modo abbastanza chiaro il concetto di DM:

- *Il DM è la non banale estrazione di informazione implicita, precedentemente sconosciuta e potenzialmente utile attraverso l'utilizzo di differenti approcci tecnici (Frawley, Piatetsky-Shapiro e Matheus).*

- *Il DM è la ricerca di relazioni e modelli globali che sono presenti in grandi database, ma che sono nascosti nell'immenso ammontare di dati, come le relazioni tra i dati dei pazienti e le loro diagnosi mediche. Queste relazioni rappresentano una preziosa conoscenza del database e, se il database è uno specchio fedele, del mondo reale contenuto in esso (Holshemier e Siebes).*

- *Il DM è l'esplorazione e l'analisi, attraverso mezzi automatici e semiautomatici, di grosse quantità di dati allo scopo di scoprire modelli e regole significative (Berry, Linoff)*

- *Il DM si riferisce all'uso di una varietà di tecniche per identificare "pepite" di informazione e di conoscenza per il supporto alla decision making. L'estrazione di tale conoscenza avviene in modo che essa possa essere usata in diverse aree come supporto alle decisioni, previsioni e stime. I dati sono spesso voluminosi, così come sono, hanno un basso valore e nessun uso diretto può esserne fatto; è l'informazione nascosta nei dati che è utile (Clementine user guide)*

Infine una definizione poco formale ma molto illuminante e intuitiva:

- *Il DM è una vera e propria tortura dei dati al fine di farli confessare…….*

CAPITOLO 1

Knowledge discovery in databases e Data Mining

1.1 Uno sguardo ravvicinato al data mining

La crescente disponibilità di dati nell'attuale società dell'informazione ha evidenziato la necessità di disporre di adeguati strumenti per la loro analisi infatti la grossa abbondanza di dati costituisce un notevole potenziale informativo che, compreso e valorizzato, può permettere di intraprendere migliori decisioni ed azioni. In tale contesto il data mining si pone come processo di selezione, esplorazione e modellazione di grandi masse di dati, al fine di scoprire regolarità o relazioni non note a priori e allo scopo di ottenere un risultato chiaro ed utile al proprietario del database.

Il data mining non è una semplice analisi statistica ma si presenta come processo che impiega una o più tecniche di apprendimento computerizzate per analizzare automaticamente ed estrarre le conoscenze dai dati contenuti in un database. L'obiettivo di una sessione di data mining è identificare trend e pattern nei dati, estraendo da essi "conoscenza" che si traduca in un modello utile o in una generalizzazione dei dati disponibili.

L'insieme di queste tecniche e strumenti, che consentono di assistere " in modo intelligente e automatico" gli utenti, sono al centro di interesse del campo emergente del Knowledge Discovery in Databases (KDD).

Spesso i termini DM e KDD sono usati come sinonimi, in realtà la differenza è notevole infatti il data mining costituisce solo una fase, consistente nell'applicazione di uno specifico algoritmo per l'individuazione dei pattern, anche se la più importante,del processo di estrazione della conoscenza.

Il data mining è una strategia di apprendimento di natura induttiva che costruisce modelli per identificare pattern nascosti nei dati; il modello che si ottiene è una generalizzazione concettuale dei dati quindi i concetti, intesi come insiemi di oggetti, simboli o eventi che vengono raggruppati perché hanno in comune alcune caratteristiche, sono l'output di una sessione di data

mining. La generalizzazione ottenuta può assumere la forma di un albero, di una rete, di un'equazione o di un insieme di regole.
Si capisce quindi la profonda differenza tra DM e data query infatti il secondo può aiutarci a trovare risposte a domande che ci chiediamo sulle informazioni immagazzinate nei dati mentre il primo ci offre risposte a domande che non avremmo mai pensato di porci. I dati sono caratterizzati da un insieme di informazioni di input, organizzate in tabelle secondo il formato attributo-valore; esse sono utilizzate per creare un modello che rappresenta i dati e che consente di predire il valore di un attributo di output. Anche se il dataset è piccolo, potrebbe essere difficile per noi sviluppare una rappresentazione generale senza avere informazioni sull'importanza relativa dei singoli attributi e sulle possibili relazioni fra gli attributi. Fortunatamente appositi strumenti di DM (algoritmi di apprendimento supervisionato) riescono a svolgere questo compito al nostro posto. Il data mining è utilizzato per costruire modelli generalizzati che rappresentano dati non strutturati [1.1].
Per problemi di classificazione si potrebbe utilizzare un approccio alternativo, detto **nearest neighbor classification** (classificazione all'oggetto più vicino) che:

- Crea una tabella di classificazione contenente tutti i dati la cui classificazione è nota.
- Per ogni nuovo caso da classificare:
    1. Calcola un punteggio che rappresenta la similitudine di ogni oggetto della tabella con il nuovo caso.
    2. Assegna al nuovo caso la stessa classificazione dell'oggetto della tabella cui è più simile.
    3. Aggiunge il nuovo caso classificato alla tabella dei casi noti.

Questo approccio immagazzina i casi, anziché creare un modello generalizzato dei dati e crea almeno tre problemi. In primo luogo, i tempi di calcolo diventano lunghi quando la tabella di classificazione contiene migliaia

o milioni di record. Secondariamente, questo approccio non effettua alcuna distinzione tra attributi rilevanti e irrilevanti. In terzo luogo, non abbiamo modo di dire se alcuni degli attributi scelti siano in grado di differenziare le classi contenute nei dati.

Una variante di questo approccio, detta **K-nearest neighbor classifier**, classifica un nuovo caso nella classe più comune dei suoi k oggetti più vicini; ciò evita il rischio che un singolo caso atipico (outlier) nei dati di training porti ad una classificazione errata dei nuovi casi. Comunque, entrambi gli approcci non sono in grado di determinare gli attributi rilevanti ed è molto probabile che producano un alto tasso di classificazioni errate quando i casi sono caratterizzati da un gran numero di attributi irrilevanti.

Inoltre, contrariamente a quanto il nostro intuito ci suggerisce, un dataset con numerosi attributi indipendenti presenterà distanze quasi uguali tra due casi qualsiasi.

Il DM si basa principalmente sui dati per ottenere informazioni nuove, ecco perché qualora la qualità dei dati sia in dubbio, si preferisce affidarsi ai sistemi esperti, che utilizzano le conoscenze umane, anziché i dati, per costruire modelli di supporto al processo decisionale.

1.2 Il processo KDD

Il KDD è una procedura interattiva e iterativa [1.2], che cerca di estrarre dai dati informazioni implicite, sconosciute a priori e potenzialmente utili. Tale processo si articola in sette fasi, esse sono state identificate nel 1998 [1.3] da Usama Fayyad, Piatetsky-Shapiro e Smith (fig 1.1).

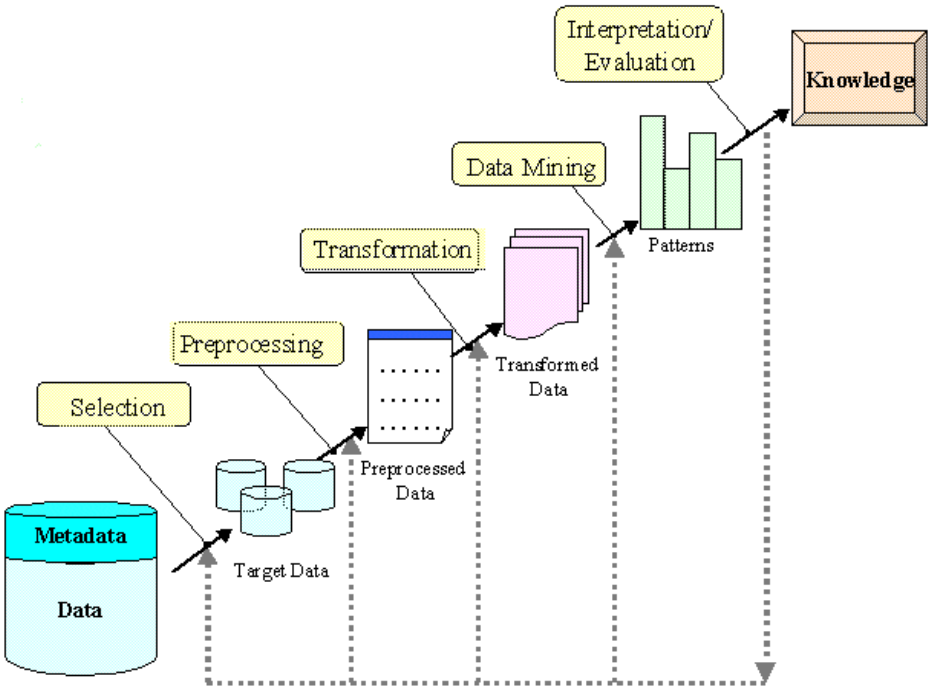

Figura 1.1: Le fasi del KDD

Andiamo ad analizzare ora le singole fasi:

- **Identificazione degli obiettivi**: l'oggetto di questa fase è l'individuazione dell'ambito di applicazione in cui deve essere considerato il KDD, inoltre bisogna individuare gli obiettivi da perseguire. Si tratta forse della fase più difficile sia in termini di allocazione risorse sia perché devono essere determinate, in modo preciso, le misure del successo e i criteri per misurare successi e fallimenti. Si può fare una lista solo parziale dei molteplici aspetti che vanno presi in considerazione, alcuni sono il costo stimato del progetto e la scelta degli strumenti di data mining da utilizzare.
- **Selezione**: In questa fase deve essere selezionato l'insieme iniziale dei dati, che devono essere sottoposti ad analisi. I dati grezzi vengono segmentati e selezionati secondo alcuni criteri al fine di pervenire ad un sottoinsieme di dati, che rappresentano il nostro target data o dati obiettivo. Infatti è abbastanza evidente che un database può contenere

diverse informazioni inutili per il caso di studio in esame. I dati target sono estratti da risorse primarie quali un data warehouse, uno o più database transazionali o uno o più flat file. Se i dati originali sono collocati in un flat file, la creazione del target risulta molto semplice. I sistemi di gestione dei database immagazzinano e manipolano dati transazionali, ciò consente ai sistemi informatici,relativi a tali sistemi, di fare aggiornamenti e di estrarre informazioni in modo rapido. Ciò è dovuto alla strutturazione dei dati tramite modelli relazionali, il cui scopo è ridurre la ridondanza dei dati, tramite la decomposizione di singole tabelle in più strutture relazionali, ed accelerare l'accesso alle informazioni. Del resto lo scopo del DM è proprio utilizzare la ridondanza dei dati per reperire "conoscenza", ecco perché è necessario ricomporre le strutture relazionali. Si intuisce quindi che è stretto il legame tra DM e Data Warehouse, il cui scopo è proprio quello di mettere insieme i dati, e non scomporli, al fine di sfruttarne la ridondanza. Spesso è anche necessario mettere insieme informazioni estratte da più fonti, cosa che può rendere complessa la fase di selezione in quanto bisogna trasformare i dati in modo da assicurare l'omogeneità in quanto, ad esempio, la codifica dei dati deve essere uguale per tutti i record dei dati target, altrimenti l'analisi risulta di scarsa utilità.

- **Preelaborazione:** Generalmente il target data disponibile non deve essere analizzato interamente ma basta estrarne un campione opportuno, eseguendo poi un'analisi su base campionaria. Inoltre i dati devono essere preprocessati, cioè "puliti", trattando in maniera opportuna i dati anomali e mancanti. Vanno individuati i valori errati delle variabili; trovare gli errori nei dati categorici diventa un problema quando si analizzano dataset molto grandi. I dati vanno anche semplificati; queste tecniche di data smoothing sono mirate alla riduzione del numero di valori per una variabile numerica. Alcuni

classificatori, come le reti neurali, utilizzano funzioni che effettuano la semplificazione durante il processo di classificazione, eseguendo così un data smoothing interno. Due semplici tecniche di semplificazione sono il calcolo e l'arrotondamento dei valori medi. Lo smoothing del valore medio è appropriato quando si usa un classificatore che non supporta i dati numerici e vogliamo avere un'informazione grossolana sui valori delle variabili numeriche; in tal caso tutti i valori delle variabili numeriche sono sostituiti con la media della classe. La presenza di dati mancanti può essere affrontata in diversi modi; in genere la mancanza di valori è indicativa di informazioni perse. Alcune tecniche di data mining sono in grado di trattare direttamente i valori mancanti tuttavia molti classificatori richiedono che tutte le variabili abbiano dei valori per cui si devono scartare tutti i record contenenti valori mancanti oppure, per valori reali, sostituire i valori che mancano con la media della classe oppure sostituirli con i valori rilevati per altre osservazione molto simili.

- **Trasformazione:** I dati, per essere utilizzati, spesso devono essere trasformati; questa fase può assumere varie forme e può essere necessaria per varie ragioni. Si possono convertire tipi di dati in altri o definirne di nuovi, ottenuti attraverso l'uso di operazioni matematiche e logiche sulle variabili, eseguire delle normalizzazioni (scalamento decimale, normalizzazione min-max o con lo z-score) o addirittura eliminare delle variabili. In genere infatti gli algoritmi di DM non lavorano in modo efficiente se i dati contengono una grande quantità di variabili che non sono in grado di prevedere la classe di appartenenza; si rende quindi utile una ricerca ed una successiva eliminazione delle variabili ridondanti e "inutili" per il problema in questione. A volte le variabili con poco potere previsivo possono essere combinate con altre per formare nuove variabili con un alto grado di capacità previsiva.

- **Data mining:** Ai dati trasformati vengono applicate una serie di tecniche in modo da poterne ricavare dell'informazione non banale o scontata. Sono gli obiettivi che si vogliono raggiungere a dare un'indicazione sul tipo di tecnica/algoritmo che deve essere applicata.
- **Interpretazione e valutazione:** Scopo di questa fase è determinare la validità del modello ottenuto con il DM; in sintesi non basta interpretare i risultati ma bisogna capire in che misura questo modello o risultato possa essere utile. Questo può essere fatto in vari modi sia attraverso un'analisi statistica che euristica o sperimentale.
- **Azione:** L'ultimo obiettivo consiste nell'utilizzare ciò che è stato appreso, creando un report o un rapporto tecnico su ciò che è stato scoperto e cercando di capire in che modo sfruttare ciò che è stato scoperto.

Si capisce bene quindi che il processo di estrazione della conoscenza è lungo e piuttosto articolato, sono fondamentali le scelte che si fanno per il trattamento di anomalie o errori nei dati e l'identificazione chiara degli obiettivi che si vogliono perseguire.

Scalamento decimale	$newVal = \frac{oldVal}{10^n}$
Normalizzazione min-max	$newVal = \frac{oldVal - oldMin(newMax - newMin) + newMin}{oldMax - oldMin}$
Normalizzazione con lo z-score	$newVal = \frac{oldVal - \mu}{\sigma}$
Normalizzazione logaritmica	$newVal = \log_b(oldval)$

Figura 1. 2: Vari tipi di trasformazione dei dati

1.3 Uno standard per il DM: il modello CRISP-DM

Sebbene il DM non ricopra ancora un ambito ben delimitato, esiste un progetto finanziato dalla Commissione europea il cui obiettivo è quello di definire un approccio standard ai progetti di data mining.

Il CRISP-DM (Cross Industry Standard Process for data mining) affronta la necessità di tutti gli utenti coinvolti nella diffusione di tecnologie di DM per la soluzione di problemi aziendali; è un prodotto neutrale sviluppato da un consorzio di numerose società il cui scopo è quello di definire uno schema di approccio indipendente dalla tipologia di business.

Secondo il CRISP, il ciclo di vita di un progetto di DM è articolato in sei fasi la cui sequenza non è rigida ma spesso ci sono dei loop all'indietro, questo dipende dalla bontà del risultato di ogni fase, che costituisce la base di partenza della fase successiva [1.4].

Le varie fasi del CRISP sono illustrate in figura 1.2, in essa sono rappresentate anche delle frecce che indicano le più frequenti dipendenze tra le fasi mentre l'ellisse che include lo schema serve a rappresentare la natura ciclica di un processo di DM infatti quanto imparato durante un processo apporta nuove informazioni per processi DM successivi.

Si capisce che, poiché il DM è la fase più importante del KDD, alcune fasi del processo CRISP sono molto simili, se non uguali, a quelle già illustrate del KDD.

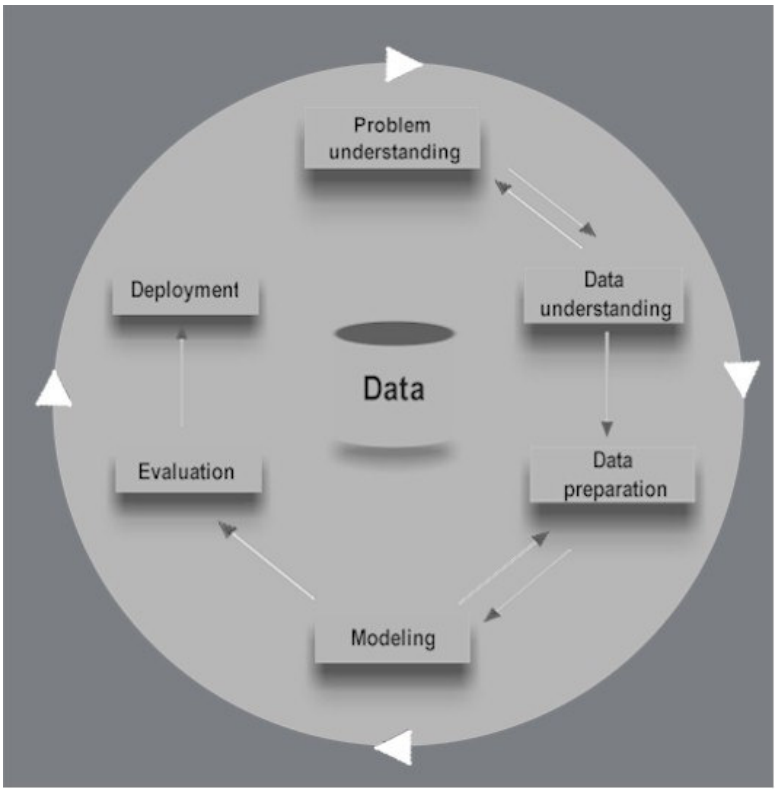

Figura 1. 3: Le fasi del Crisp-DM

Descriviamo ora nel dettaglio le fasi della figura:

- **Problem understanding:** L'attenzione è riposta principalmente sugli obiettivi e le rischieste del progetto da una prospettiva prettamente di business. Viene inoltre definito il problema di data mining.
- **Data understanding:** Individuati gli obiettivi nella fase precedente, ciò di cui disponiamo per perseguirli è rappresentato dai dati. Bisogna quindi raccogliere i dati e identificare problemi nella qualità dei dati stessi. Risulta chiaro che le prima due fasi sono collegate dato che rappresentano l'individuazione dei fini e dei mezzi di un progetto di DM.
- **Data preparation:** Essa comprende la selezione di tabelle, di record e di attributi come anche, se necessaria, la trasformazione e pulitura dei dati.

- **Modelling:** Vengono selezionate e applicate varie tecniche che consentono di ricavare modelli. Determinate tecniche, per poter essere applicate, necessitano di specifiche richieste in termini di forma dei dati, per cui è spesso opportuno tornare indietro alla fase di preparazione dei dati per modificare il dataset iniziale e adattarlo alla tecnica specifica che si vuole utilizzare.
- **Evaluation:** Tramite l'analisi dei risultati si valuta se sono stati raggiunti gli obiettivi iniziali prefissati. Prima di procedere all'impiego del modello o dei modelli costruiti, è molto importante valutare non solo essi ma anche i passi eseguiti per costruirli e capire se è stato trascurato qualche dettaglio.
- **Deployment:** Si tratta della fase finale che prevede l'utilizzo del modello/i realizzato, che consente il raggiungimento dei fini desiderati.

Dall'analisi delle singole fasi, si capisce bene che un ruolo centralissimo è svolto dalla fase di modelling, che è quella che noi andremo a vedere nel dettaglio per capire realmente cosa si intenda per DM.

1.4 Verification models e Discovery models: i due diversi aspetti del DM previsivo

Risulta complesso stabilire se il data mining sia la strategia più appropriata alla risoluzione di un particolare problema; bisogna infatti capire se siamo in grado di definire chiaramente il problema, se esistono dati significativi, se i dati contengono informazioni nascoste o hanno solo una finalità di report e infine se il costo di analisi dei dati sarà minore dell'incremento del profitto ottenibile applicando le nuove conoscenze ricavate dal progetto di DM.
Solo dopo si potrà scegliere di seguire un approccio di tipo top-down o di tipo bottom-up. Nel primo caso si tratta di utilizzare la statistica come guida per l'esplorazione dei dati, cercando di trovare conferme a fatti che l'utente

ipotizza o già conosce, o per migliorare la comprensione di fenomeni parzialmente conosciuti; tuttavia, un approccio top-down limita il compito del DM alla descrizione. Quest'ultima non può fornire quelle decisioni di supporto alle decisioni, cui si fa costantemente riferimento quando si parla di potenzialità del DM, di conseguenza si preferisce un approccio bottom-up, nel quale l'utente analizza e "scava" nei dati alla ricerca di informazioni che a priori ignora . Questo secondo approccio conduce ad un DM di tipo previsivo in cui si costruisce uno o più set di modelli, si effettuano delle inferenze sui set di dati disponibili e si tenta di prevedere il comportamento di nuovi dataset [1.5]. IBM ha identificato nel data mining previsivo due modi di operare, che possono essere usati per estrarre informazioni di interesse per l'utente: i Verification models e i Discovery models.

I verification models utilizzano delle ipotesi formulate dall'utente e verificano tali ipotesi sulla base dei dati disponibili. In questi tipi di modelli riveste un ruolo cruciale l'utente, al quale è affidato il compito di formulare delle ipotesi sui possibili comportamenti delle variabili in questione. Tuttavia risultano di gran lunga più interessanti i discovery models, che costituiscono la parte più rilevante delle tecniche di DM. In questi tipi di modelli all'utente non è affidato nessun tipo di compito specifico, è il sistema che scopre "automaticamente" importanti informazioni nascoste nei dati: si cerca di individuare pattern frequenti, regole di associazione, valori ricorrenti. Volendo esagerare si potrebbe affermare che il DM è costituito solo dai discovery models, dal momento che un'importante differenza tra i metodi tradizionali di analisi dei dati e i nuovi metodi è che i primi sono guidati da assunzioni fatte, cioè vengono formulate delle ipotesi che vengono verificate attraverso i dati, mentre i secondi sono guidati dalla scoperta, nel senso che i modelli sono automaticamente estratti dai dati [1.6]. Questa è anche la grande differenza

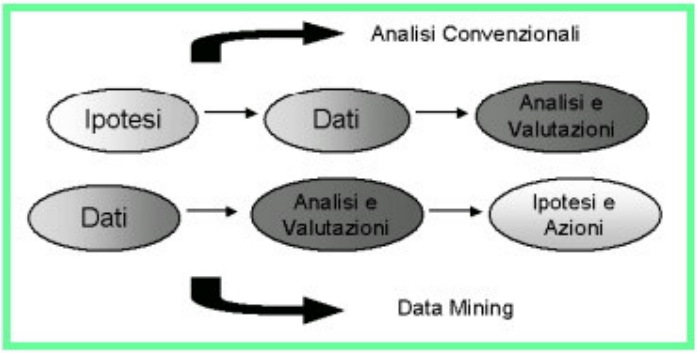

Figura 1. 4: Differenze concettuali tra analisi convenzionali e analisi di DM

tra DM e tecniche OLAP (On-line Analytical Processing), che hanno alla loro base un'analisi deduttiva infatti l'utilizzatore interroga il database per validare o meno un'ipotesi precedentemente formulata ma questo approccio diventa sempre più difficile da portare avanti al crescere del numero di variabili [1.7].

Le tecniche OLAP sono mirate ad effettuare indagini on-line sui dati di un Data Warehouse, finalizzate a supportare più processi decisionali. In realtà il punto di vista migliore è quello di guardare ai due tipi di strumenti come tecniche complementari e non alternative: il DM consente agli utenti di strumenti OLAP di andare oltre i report riassuntivi perché consente di capire perché un certo fenomeno stia succedendo, non limitandosi a dire solo cosa si sta verificando. Spesso questi due strumenti vanno utilizzati quindi contemporaneamente.

CAPITOLO 2

Strategie di DM supervisionato: Link analysis e Association discovery

2.1 Strategie di data mining

Le strategie di DM possono essere suddivise in **strategie supervisionate** e **non supervisionate**. L'apprendimento supervisionato costruisce modelli tramite l'utilizzo di attributi di input (variabili indipendenti) per predire i valori degli attributi di output (variabili dipendenti). La maggior parte degli algoritmi di tipo supervisionato ammette un solo attributo di output, solo pochi strumenti consentono di specificare più attributi di output. Quando l'apprendimento non è supervisionato, non esiste un attributo di output; di conseguenza, tutti gli attributi utilizzati per costruire i modelli sono variabili indipendenti. Le strategie di apprendimento supervisionato possono ulteriormente classificate sia in funzione del fatto che gli attributi si output siano discreti o categorici, sia in base al fatto che i modelli siano progettati per determinare una condizione corrente o per predire una situazione futura.

Figura 2. 1: Gerarchia delle strategie di DM

Le principali strategie di apprendimento supervisionato sono la classificazione, la stima e la previsione. La prima è caratterizzata da una variabile di output di tipo categorico e l'enfasi è riposta sulla costruzione di modelli in grado di assegnare nuovi casi a una di un insieme di classi ben definite. La seconda, analogamente alla classificazione, ha come scopo la determinazione di un valore per un attributo di output incognito; tuttavia gli attributi di uscita sono numerici, anziché categorici. Lo scopo di un modello di previsione, diversamente dai primi due, è determinare il comportamento futuro invece di quello attuale. Nella clusterizzazione non supervisionata invece, non esiste una variabile dipendente che guida il processo di apprendimento ma è il programma di apprendimento che costruisce una struttura di conoscenze utilizzando alcune misure della qualità dei cluster per raggruppare i casi in due o più classi. Un obiettivo primario di tale strategia è scoprire le strutture concettuali dei dati, le applicazioni più comuni della clusterizzazione non supervisionata includono:

- Determinare se i dati presentano relazioni significative nella forma dei concetti.
- Valutare le prestazioni di un modello di apprendimento supervisionato.
- Determinare l'insieme ideale di attributi di input per l'apprendimento supervisionato.
- Identificare gli outlier.

La cosa più interessante è proprio l'impiego della clusterizzazione non supervisionata per stabilire se un modello ottenuto con strategie supervisionate avrà buone o scarse prestazioni. Qualora si presenti la seconda eventualità, una soluzione potrebbe essere quella di rivedere gli attributi e i casi scelti per creare il modello supervisionato. In effetti, è possibile scegliere gli attributi migliori per un modello supervisionato applicando ripetutamente la

clusterizzazione con attributi alternativi. In questo modo, è possibile determinare quegli attributi che sono in grado di distinguere meglio le classi note presenti nei dati.

Sfortunatamente, anche quando si hanno a disposizione pochi attributi, l'applicazione di questa tecnica potrebbe risultare difficile da implementare. La clusterizzazione non supervisionata può aiutare anche a individuare alcuni casi anomali presenti nei dati, tali casi sono detti **outlier** e possono rivestire una grande importanza per cui dovrebbero sempre essere identificati. Nel data mining, infatti, gli outlier potrebbero essere proprio quei casi che stiamo cercando di identificare; l'obiettivo di un'analisi di DM può essere quello di riuscire a identificare delle situazioni anomale.

Infine all'interno delle strategie di DM è fondamentale la **market basket analysis,** il cui scopo è ricercare relazioni interessanti tra i prodotti delle vendite al dettaglio. I risultati di tale analisi aiutano i commercianti a ideare le campagne promozionali, a disporre i prodotti negli scaffali, a presentarli nei cataloghi e a sviluppare strategie di marketing incrociato.

2.1 Classificazione e raggruppamento : principali differenze

Negli algoritmi di classificazione i dati di ingresso, cioè il training set, sono organizzati in record, a ciascuno di essi è associata una etichetta di classe. Obiettivo della classificazione è quello di analizzare i dati di input e sviluppare un'accurata descrizione o un modello per ogni classe, usando le caratteristiche presenti nei dati. Gli algoritmi di classificazione hanno come obiettivo la determinazione di uno strumento che consenta di individuare la classe di appartenenza di un dato record.

Partendo dall'uso di insiemi esistenti e già classificati, gli algoritmi cercano di individuare le "regolarità", che costituiscono i tratti caratteristici di ciascuna classe. Le descrizioni delle classi vengono usate per classificare record, di cui

non si conosce la classe di appartenenza, o per sviluppare una migliore conoscenza di ogni classe nel dataset. Non a caso, alcune delle applicazioni di maggiore interesse di questa tecnica di DM includono la ricerca, da parte delle banche, di categorie di clienti al quali concedere un prestito (in tal caso ogni record è etichettato come buon creditore o cattivo creditore e gli attributi sono i dati dei clienti relativi all'età, al reddito, ecc..) o applicazioni di target marketing, con cui un'impresa può individuare, sulla base delle caratteristiche dei clienti presenti nel database, un proprio target di mercato allo scopo di rafforzare la propria posizione in un determinato settore (in tal caso etichettando ogni record del database come cliente fedele e cliente non fedele).

Esiste una sostanziale differenza, che è opportuno chiarire, tra le tecniche di classificazione e le tecniche di raggruppamento o clustering. Tramite la classificazione l'utente comunica al tool di DM la caratteristica chiave, di cui i membri del gruppo devono essere dotati, e il tool non si occupa di nessun altro attributo che i membri del gruppo possono avere in comune. Ad esempio, si può dire al tool di DM di creare dei gruppi di clienti sulla base dei diversi livelli di entrate che apportano all'azienda; utilizzando queste informazioni, l'azienda, l'azienda può investire tempo e denaro per corteggiare potenziali clienti con caratteristiche simili.

In altre parole, nella **classificazione** esiste un **attributo madre**, che è proprio la classe di appartenenza, il cui numero di modalità rappresenterà il numero dei gruppi che si verranno a formare. Due record appartenenti al medesimo gruppo in un processo di classificazione possono in realtà essere fortemente diversi tra loro. Tale diversità è legata al fatto che il tool di DM crea gruppi sulla base di tutti gli attributi presenti nel database, in modo che ogni gruppo sia caratterizzato da elementi "simili" in termini degli attributi descritti nel database e che due elementi appartenenti a gruppi diversi siano sufficientemente "distanti tra loro".

2.3 La link analysis

La link analysis è un approccio descrittivo, di DM supervisionato, di esplorazione dei dati, che può aiutare ad identificare relazioni fra gli attributi in un database, attraverso l'analisi dei valori dei dati. **L'obiettivo è riuscire a determinare una regolarità nei dati**, ciò può essere utile in vari ambiti applicativi per capire ad esempio:

- Quali farmaci vengono somministrati contemporaneamente a pazienti affetti da una certa patologia.
- Quali geni sono sensibili all'azione di un farmaco.
- Se si possono classificare automaticamente dei documenti Web

I due approcci più comuni alla link analysis sono l'association discovery e la sequence discovery (che rientra nell'ambito dei modelli sequenziali/temporali).
Diversamente dalle tradizionali regole produttive, che non sono altro che una traduzione degli alberi decisionali, **le regole associative possono avere uno o più attributi di output**. Inoltre un attributo di output per una regola può essere un attributo di input per un'altra regola [2.1].
Le regole associative sono una tecnica comune nella market basket analysis, in quanto possono essere esaminate tutte le possibili combinazioni di raggruppamenti di prodotti interessanti. Per questo motivo un numero limitato di attributi può generare centinaia di regole associative. Una regola di associazione è rappresentata dai simboli $X \Rightarrow Y$ dove X è chiamato antecedente e Y conseguente. Se $L = A_1, A_2, A_3, ...A_n$ è un insieme di attributi che rappresentano le qualità di ciascun record nel database, indicato con $P(L)$ l'insieme delle parti di L, $P(L)$ rappresenterà l'insieme di tutti i sottoinsiemi di L e dunque l'insieme di tutte le combinazioni di attributi. Un record del database potrà essere rappresentato da un opportuno $r \in P(L)$ e

potremmo indicare con $R \subseteq P(L)$ l'insieme di tutti i record del database (R può ovviamente contenere record uguali). Nella regola di associazione $X \Rightarrow Y$, individuata attraverso l'esame dell'insieme R, si ha $X \subseteq L$ e $Y \in L \mid X$ cioè X è un sottoinsieme di attributi e Y è un singolo attributo che non coincide con nessuno degli attributi dell'insieme X, ovvero, in termini insiemistica, $\{X \cap Y\} = 0$.

Esempio 1:

Supponiamo di avere a disposizione il seguente database, all'interno del quale si vogliono individuare delle regole associative:

A	B		D	E	F
A	B	C		E	F
	B	C	D	E	
A	B		D		F
A	B	C	D	E	F
A	B			E	
A	B	C		E	F
	B	C			
	B	C	D	E	F
A	B		D	E	

Figura 2.2: Tabella di un ipotetico database

In base a quanto detto: $L = \{A, B, C, D, E, F\}$ è l'insieme di attributi, $P(L) = \{\{A\},\{B\},\{C\},\{D\},\{E\},\{F\},\{A,B\},\{A,C\},....\{A,B.C,D,E,F\},0\}$ ed $R = \{r_1, r_2, r_3, ..., r_{10}\}$ infatti R non ha elementi uguali. Si possono quindi determinare le seguenti regole di associazione:

- A, B $\Rightarrow$ E con $X = \{A,B\}$ e $Y = E \in L \mid X$
- B $\Rightarrow$ E con $X = \{B\}$ e $Y = E \in L \mid X$

- A, B, C $\Rightarrow$ F con $X = \{A, B, C\}$ e $Y = F \in L \mid X$
- D, E $\Rightarrow$ F con $X = \{D, E\}$ e $Y = F \in L \mid X$

All'interno del database si possono individuare anche altre regole di associazione ma è evidente che non tutte le regole hanno rilevanza statistica, e che alcune potrebbero essere ovvie; è quindi fondamentale stabilire dei criteri di scelta delle regole.

2.4 Confidenza, supporto e lift

Le regole associative sono diverse dalle tradizionali regole di classificazione perché una variabile che si presenta come precondizione di una regola può apparire come condizione conseguente di una seconda regola. Inoltre, le regole tradizionali di classificazione, di solito, ammettono nella condizione conseguente una sola variabile. I generatori di regole associative ammettono che la condizione conseguente di una regola possa contenere una o più variabili [2.2]. Diventa però fondamentale individuare relazioni statisticamente valide nel database introducendo due parametri statistici che consentono di selezionare le regole statisticamente significative.

- Si definisce **indice di confidenza** della regola di associazione $X \Rightarrow Y$ la frazione di record del database per i quali la presenza di X implica la presenza di Y.
- Si definisce **supporto** della regola di associazione $X \Rightarrow Y$ il numero di record nel database per i quali si verifica la presenza contemporanea degli attributi che compongono la regola di associazione.

In pratica **l'indice di confidenza ci dice la probabilità condizionata che si verifichi Y dato X**; il supporto invece è semplicemente la percentuale minima di osservazioni che contengono tutti gli oggetti elencati in una particolare

regola associativa. **Il supporto di una regola di associazione da l'idea della base su cui poggia la regola stessa**: un basso valore del supporto indica che la regola non è statisticamente rilevante e quindi importante.
Nell' *esempio 1* :

- La regola A, B $\Rightarrow$E ha supporto 6 (6/10 pari al 60%) e confidenza 6/7
- La regola B$\Rightarrow$E ha supporto 8 (8/10 pari al 80%) e confidenza 8/10
- La regola A, B, C$\Rightarrow$F ha supporto 3 (3/10 pari al 30%) e confidenza 3/3
- La regola D, E$\Rightarrow$F ha supporto 3 (3/10 pari al 30%) e confidenza 3/5

Esempio 2:

Supponiamo che il seguente sia un database in cui siano registrati i farmaci somministrati a più pazienti affetti da una stessa patologia:

Patient-id	Drugs
10	A, B, C
20	A, C
30	A, D
40	B, E, F

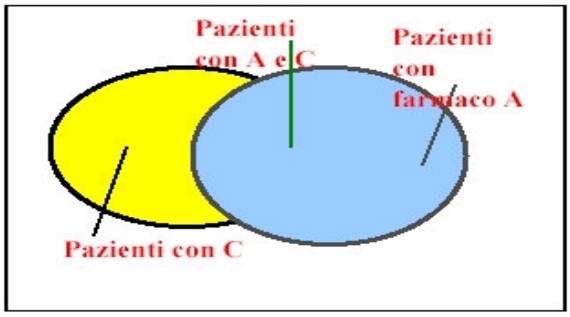

Figura 2.2: Tabella di un ipotetico database

Se assumiamo di voler trovare le regole aventi come supporto minimo il 50% e come confidenza minima il 50%, otteniamo:

- A $\Rightarrow$C con supporto del 50% e confidenza del 66,6%

- $C \Rightarrow A$ con supporto del 50% e confidenza del 100%

Si trova quindi uno stretto legame nell'utilizzo dei due farmaci A e C.
Un altro parametro che consente di misurare la potenza di un'associazione è il **lift,** che è uno strumento di misura proveniente direttamente dal marketing; esso consente di misurare l'influenza che il verificarsi dell'antecedente X ha sul verificarsi del conseguente Y. Esso ha la seguente espressione:

$$Lift(X \Rightarrow Y) = confidenza(X \Rightarrow Y) / frequenza(Y)$$

Si può anche esprimere il lift facendo ricorso alla teoria della probabilità condizionata infatti c'è un'analogia tra la formula del lift e quella classica della probabilità condizionata:

$$P(A \setminus B) = \frac{P(A \cap B)}{P(B)}$$

Il lift è importante perché non ci informa semplicemente sulla probabilità condizionata che si verifichi Y dato X ma rapporta questa probabilità alla frequenza del verificarsi di Y nel database.
La ricerca delle regole di associazione si può ricondurre alla ricerca, fra tutte le possibili regole costruibili tra gli attributi che compaiono nel database, di quelle che abbiano un minimo livello di supporto e di confidenza, in più il lift consente di scegliere la soluzione migliore.
Gli algoritmi per la ricerca di regole di associazione mirano a determinare tutte le regole $X \Rightarrow Y$ con supporto maggiore o uguale ad un dato s e con confidenza maggiore o uguale ad un dato c.
L'efficienza con cui i vari algoritmi individuano le regole che soddisfano i requisiti richiesti è uno degli elementi differenziatori fra i vari algoritmi ed in un contesto di DM, trattando con grossi database, il problema dell'efficienza è fondamentale. Quasi tutti gli algoritmi di ricerca delle regole di associazioni inclusi nei software per il DM permettono anche di trovare tutte le regole che

rispondano a requisiti ancora più specifici dei classici vincoli di supporto e confidenza. Si può, ad esempio, richiedere al software di trovare tutte le regole in cui un particolare attributo sia il conseguente e tali che le regole trovate rispettino dei vincoli di supporto e confidenza. Al fine di evitare la scelta di regole con un basso valore di supporto e confidenza, si considera un ulteriore limite inferiore sotto il quale la regola viene esclusa, anche se rientra nei limiti di supporto e confidenza minimi prescritti. Questo ulteriore parametro è detto **comunanza**.

2.5 L'algoritmo Apriori per la ricerca delle regole

Nelle applicazioni di DM il numero di righe (può essere dell'ordine di 10^6) ed il numero di colonne di un database (può essere dell'ordine di 10^3) può essere di ordine elevato per cui, a seconda dei valori soglia di supporto e confidenza, non è raro ottenere da un tale database centinaia o migliaia di regole di associazione che rientrano nei parametri richiesti.

Il valore di soglia tipico del supporto va da 10^{-2} a 10^{-4} e il valore soglia di confidenza è generalmente compreso tra 0 e 1 [2.3].

La crescita esponenziale del numero di regole di associazione è inoltre legata al fatto che il numero di attributi dell'antecedente non è fisso, né d'altro canto è fisso il conseguente; questo modo di procedere ha il vantaggio di permettere l'individuazione di regole inaspettate, che in tal modo non vengono automaticamente escluse dalla procedura di ricerca, ma ha lo svantaggio di portare ad una crescita esponenziale del numero di regole che soddisfano i valori di soglia.

L'algoritmo **Apriori** ultilizza un approccio del tipo "genera il candidato e testa". Si dice che un insieme X è **frequente** quando il supporto di X è tale che $s(X) \geq s$.

L'algoritmo si basa sul fatto che ogni sottoinsieme di un pattern frequente (itemset) deve essere frequente per cui se {A, B, C} è frequente lo è anche {A,

B} e ogni esempio contenente {A, B, C} contiene anche {A, B}. Inoltre se c'è un itemset che non è frequente, il suo superset non deve essere generato/testato.

Una volta individuati tutti gli insiemi frequenti, trovare le regole di associazione è semplice poiché per ogni insieme X frequente e per ogni B appartenente ad X si può verificare semplicemente se la regola $X \mid B$ implica B ha un valore di confidenza sufficientemente alto o al limite $\geq c$. **Il problema si riduce allora alla necessità di individuare tutti gli insiemi X di attributi frequenti** e tale problema viene risolto adattando l'algoritmo generale di una tecnica di DM al caso specifico delle regole di associazione.

L'algoritmo opera nel seguente modo: in un primo momento si trovano tutti gli insiemi di ampiezza 1, ovvero da una prima lettura del database si registra il numero di volte in cui ogni attributo A si verifica. Quindi gli insiemi di ampiezza 1 frequenti sono candidati ad essere di ampiezza 2, ovvero insiemi costituiti da coppie di attributi entrambi frequenti.

Non appena si individuano gli insiemi frequenti di ampiezza 2, essi sono candidati ad essere insiemi di ampiezza 3.

Sembra quindi chiaro il meccanismo con cui procede l'algoritmo. Inoltre, per quanto detto precedentemente e cioè che tutti i sottoinsiemi di insiemi frequenti sono anche essi frequenti, se $s(\{B, C, D\}) \geq s$ segue necessariamente che $s(\{B, C\}) > s$, $s(\{B, D\}) > s$, $s(\{C, D\}) > s$; per cui se si verifica che, ad esempio, $s(\{C, D\}) < s$ non potrà mai verificarsi che $s(\{B, C, D\}) \geq s$ e l'insieme {C, D} non è candidato a formare insiemi di grandezza 3. In generale nel momento in cui non vi sono più insiemi di n-1 candidati a formare insiemi di grandezza n il procedimento si arresta.

Il procedimento descritto può essere così schematizzato:

- C è l'insieme inizialmente costituito da tutti i valori A di R.
- F è l'insieme i cui elementi sono i sottoinsiemi frequenti trovati, inizialmente esso è vuoto.

- i=1;
- finchè C≠∅ :
    a. $\overline{F}$ è costituito dagli insiemi X di C che sono frequenti;
    b. Aggiungi $\overline{F}$ ad F;
    c. C:= insiemi Y di grandezza i+1 tali che ogni sottoinsieme W di Y di grandezza i è frequente;
    d. i:=i+1;
- End.

Questo algoritmo è costretto a leggere il database almeno k+1 volte, essendo k la grandezza del più grande insieme frequente. Nelle applicazioni pratiche k è piccolo, generalmente k = 10, per cui il numero di passi dell'algoritmo attraverso i dati è computazionalmente ragionevole.

2.6 Regole interessanti e non

Spesso capita che le regole trovate, sebbene siano statisticamente significative, non siano interessanti o siano ridondanti, ovvero che discendano logicamente da altre regole statisticamente rilevanti e interessanti. Le regole interessanti possono essere specificate attraverso l'uso di **template**.
Un template descrive un insieme di regole specificando quali attributi debbano verificarsi nell'antecedente e quali nel conseguente. L'utilizzatore quindi può richiedere al software di determinare tutte e sole le regole statisticamente rilevanti con specifici attributi sia antecedenti che conseguenti; in tal modo le regole tirate fuori saranno tutte interessanti per una determinata applicazione.
Lo svantaggio legato all'uso di template è che esso non consente di trovare regole inaspettate perché, una volta stabilite le regole su cui poggiare le regole interessanti, esse sono escluse a priori dall'output dell'algoritmo.

In generale, un template è un'espressione del tipo $A_1, A_2, A_3, ...A_k \Rightarrow A_{k+1}$ dove A_i è una classe comprendente vari attributi o un singolo attributo. L'utilizzatore specifica quali regole sono per lui importanti e i software di DM sono dotati di tool che consentono di utilizzare i template in due modi: in modo inclusivo, cioè verranno scelte le regole che soddisfano quel template, oppure in modo restrittivo, cioè verranno escluse tutte le regole descritte da quel template.

2.7 Sequence discovery

L'analisi della scoperta di sequenze è simile all'analisi delle associazioni con l'eccezione che le relazioni tra gli argomenti sono distribuiti nel tempo. In effetti, la maggior parte degli strumenti di DM trattano le sequenze semplicemente come associazioni i cui eventi sono collegati dal tempo. Mentre le regole di associazione riguardano gli attributi che si verificano contemporaneamente in un evento, quale ad esempio una transazione commerciale, le sequenze costituiscono delle regole che si manifestano, ad esempio, nel corso di due o più transazioni commerciali avvenute in tempi successivi. Gli algoritmi principalmente utilizzati per la scoperta di sequenze sono AprioriAll, AprioriSome, DynamicSome.

2.8 Conclusioni

Un quesito importante che potremmo porci è il seguente: quanto reali sono le regole di associazione trovate? In che modo è possibile sfruttare le regole trovate per fini di tipo commerciale o in generale per fini di business? Per rispondere al primo quesito diremo che le associazioni o le frequenze trovate non esistono realmente; piuttosto si tratta di descrizioni di relazioni in un particolare database. Non esiste nessuna possibilità di estendere i risultati trovati per quel database ad altri dati, in maniera tale da estendere la potenza predittiva di queste regole. Quindi una regola di associazione statisticamente rilevante ed interessante per un database di un centro commerciale, può essere statisticamente non rilevante per un altro centro o anche nell'ambito dello stesso database l'utilizzo di regole di associazioni per fini commerciali si basa sull'assunzione implicita che il comportamento passato continuerà in futuro. Premesso, inoltre, che questa ipotesi sia soddisfatta e cioè che il comportamento passato continui in futuro, una regola di associazione può, ad esempio contribuire ad un aumento del valore del "paniere medio della spesa".

Infatti scoprire che due prodotti sono fortemente associati può spingere gli addetti alla gestione del centro commerciale a posizionare i due prodotti lontani fra loro allo scopo di "rendere più difficile la vita al cliente", massimizzando in tal modo la probabilità di acquisto di prodotti, che nel caso precedente non si sarebbero potuti trovare nel percorso teorico che un cliente segue in un centro commerciale.

Le sfide maggiori per l'association-rule mining sono risolvere problemi quali il trattamento di grosse moli di dati, il conteggio molto lungo dei candidati, le ricerche e i continui accessi alle tabelle. Un miglioramento dell'algoritmo Apriori deve quindi mirare alla risoluzione di questi problemi.

CAPITOLO 3

Strategie di DM supervisionato: Alberi decisionali, algoritmi di classificazione e regole produttive

3.1 Gli alberi di decisione: concetti preliminari

Gli alberi decisionali sono strutture molto conosciute nell'apprendimento supervisionato, essi sono strumenti molto utili per la risoluzione di svariati problemi reali.
La costruzione di modelli sulla base di alberi decisionali ha lo scopo di ripartire un dataset in una serie di passi. La ripartizione avviene in base alle relazioni che legano la variabile target che si cerca di prevedere e una serie di variabili utilizzate come predittori. Il risultato del modello può essere rappresentato da un set di regole per ottenere i valori della variabile target o da una struttura ad albero.
Un albero decisionale è una struttura semplice in cui i nodi non terminali rappresentano i test di uno o più attributi e i nodi terminali riflettono le decisioni risultanti. Gli alberi decisionali offrono numerosi vantaggi: sono facili da capire, possono essere trasformati in regole e hanno dimostrato di ottenere buoni risultati.
Negli algoritmi di classificazione, e in particolare in quelli per la costruzione di alberi, i dati di input, chiamati anche **training set,** consistono in record ognuno dei quali avente attributi o caratteristiche multiple. Inoltre ogni record è etichettato con una speciale etichetta di classe.

3.2 Costruzione degli alberi di decisione

Gli alberi di decisione costituiscono il modo più semplice di classificare degli "oggetti" in un numero finito di classi. Essi vengono costruiti suddividendo ripetutamente i record in sottoinsiemi omogenei rispetto alla variabile risposta,

dove quando si parla di omogeneità ci si riferisce al fatto che le combinazioni di attributi dell'oggetto conducono alla medesime modalità di classe. La suddivisione produce una gerarchia ad albero, dove i sottoinsiemi di record sono chiamati nodi e, quelli finali, foglie. In particolare, i nodi sono etichettati con il nome degli attributi, i rami dell'albero sono etichettati con i possibili valori dell'attributo soprastante, mentre le foglie dell'albero sono etichettate con le differenti modalità dell'attributo classe che descrivono le classi di appartenenza. Un oggetto è classificato seguendo un percorso lungo l'albero che porti dalla radice ad una foglia. I percorsi sono rappresentati dai rami dell'albero che forniscono gli strumenti per la costruzione di una serie di regole. Un esempio di albero costruito a partire da un dataset costituito da cinque attributi, di cui uno di output è mostrato in figura 2.1.

La struttura del dataset è la seguente:

1. outlook { sunny, overcast, rainy }
2. temperature { hot, mild, cool }
3. humidity { high, normal }
4. windy { TRUE, FALSE }
5. play { yes, no }

L'albero viene costruito per capire in quali condizioni del tempo sia possibile o meno giocare a tennis.

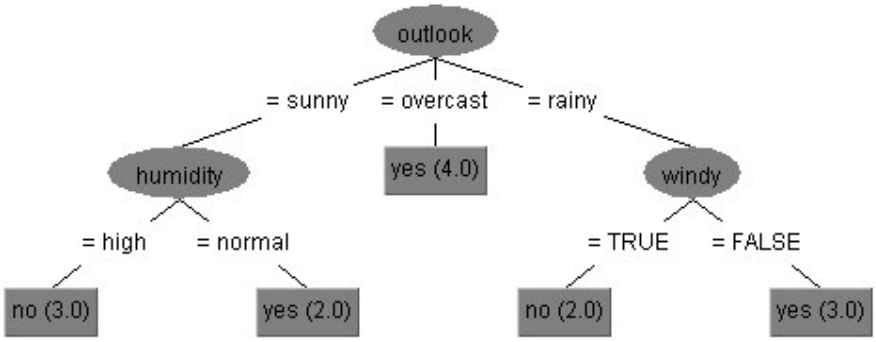

Figura 3. 1: Esempio di albero

3.3 *Criteri per la costruzione di alberi di decisione: Entropia ed Information gain*

La costruzione di alberi non è fatta in maniera casuale ma è realizzata nel rispetto di determinati criteri al fine di cercare di ottimizzare dei parametri.
Questi criteri ci consentono di capire in che modo va fatta la selezione degli attributi che portano ad una ramificazione dell'albero infatti l'obiettivo è cercare di guadagnare la maggiore informazione possibile man mano che si scende lungo l'albero [3.1].
I vari algoritmi esistenti differiscono principalmente per il tipo di funzione euristica che viene utilizzata per scegliere i campi splitter dell'albero.
Consideriamo, per semplicità, un problema di classificazione in cui siano presenti solo due classi, indicate con "+" e "-", e sia S l'insieme di record attraverso i quali dobbiamo creare un albero di decisione. Indichiamo con P_+ la percentuale di esempi classificati con "+" e con P_- la percentuale di esempi classificati con "-".
Si definisce **entropia** di S :

$$H(S) = -P_+ \cdot \log_2 P_+ - P_- \cdot \log_2 P_-$$

Se rappresentiamo su un grafico l'andamento dell'entropia al variare di una delle due percentuali, è evidente che essa assume valori tra 0 e 1 ed in particolare $H(S) = 0$ nel caso in cui $P_+ = 100\%$ o $P_- = 100\%$ ovvero nei casi in cui la totalità degli esempi è classificata in una sola delle due classi.
Del resto $H(S) = 1$ nel caso in cui $P_+ = 50\%$ e conseguentemente $P_- = 50\%$ ovvero nel caso in cui gli esempi sono egualmente suddivisi nelle due classi.

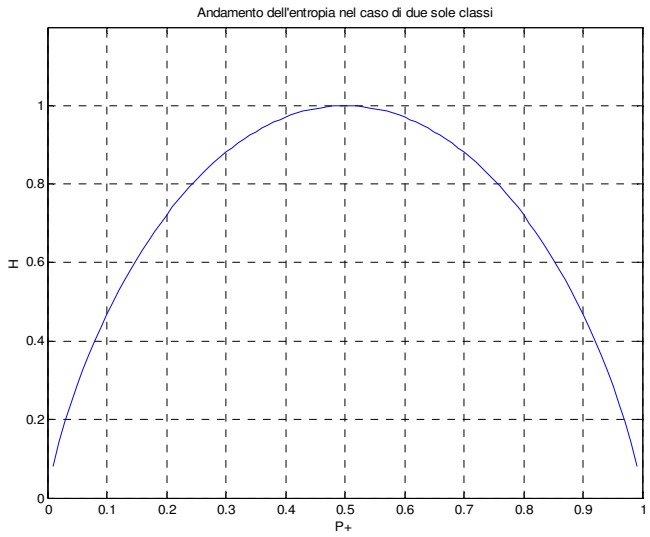

Figura 3. 2: Andamento dell'entropia

Se siamo in un caso più complesso cioè se siamo in presenza di n classi, l'espressione dell'entropia sarà la seguente:

$$H(S) = -\sum_{i=1}^{n} P_i \cdot \log_2 P_i$$

Abbiamo così definito l'entropia associata ad un training set, che non è altro che una misura del contenuto informativo dei dati di input del nostro algoritmo per la generazione dell'albero decisionale. Se l'entropia ha un valore elevato allora è presente un "disordine" elevato nello spazio dei record di ingresso per cui è piuttosto difficile classificare un record come appartenente ad una determinata classe sulla base degli attributi che la caratterizzano.

Si procede allora in questo modo: partendo da una situazione di massimo disordine in cui $H(S) = 1$ o da un valore comunque alto di entropia, si partizionano i record sulla base di un particolare attributo A, portando così ad un nuovo valore di entropia $H'(S)$ inferiore o uguale a quella di partenza.

L'entropia dopo il partizionamento si esprime come media pesata delle entropie dei singoli sottoinsiemi generati dai valori dell'attributo A.

Si può definire, a questo punto, un nuovo parametro che consente di misurare la diminuzione di entropia, e quindi di disordine, che si ottiene partizionando i dati rispetto ad un certo attributo A.
Si definisce **information gain**:

$$G(S,A) = H(S) - \sum_{v \in valori(A)} \frac{|S_v|}{|S|} \cdot H(S_v)$$

Tale parametro ha proprio il significato di informazione aggiuntiva che si ha sull'attributo classe dopo aver partizionato i record attraverso gli altri attributi del training set.
Un criterio di scelta dei nodi di un albero di classificazione può essere quello di scegliere di volta in volta l'attributo A che consente di massimizzare l'information gain e quindi di avere la massima diminuzione di entropia; molti algoritmi utilizzano questo parametro, e tra questi anche l'algoritmo **ID3,** uno dei più noti per ragioni storiche.
Sussistono però dei problemi legati all'uso dell'information gain infatti esso ha valori molto elevati in corrispondenza di attributi che sono fortemente informativi e che quindi aiutano ad identificare con buona probabilità la classe di appartenenza dei record ma spesso più gli attributi sono informativi, più perdono in generalità. Se, ad esempio, consideriamo un database di clienti di una compagnia telefonica il campo codice fiscale corrisponde ad un elevato information gain. Il codice fiscale infatti individua univocamente gli utenti e quindi consente una classificazione molto accurata dell'utenza; in altre parole, usando questo campo, lo spazio risulta più ordinato. Nonostante ciò questo campo non consente di effettuare generalizzazioni e quindi comporta un grave limite all'algoritmo di DM.
Per evitare i problemi derivanti dall'uso dell'information gain è possibile utilizzare una misura della dispersione che si ottiene utilizzando un certo attributo.

Si definisce così lo SplitInfo:

$$SplitInfo(S,A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \cdot \log_2\left(\frac{S_i}{S}\right)$$

Questo nuovo parametro consente poi di definire un nuovo criterio di split sugli alberi, detto Gain Ratio, esso non è altro che un guadagno normalizzato.

$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitInfo(S,A)}$$

Questo parametro è molto utilizzato in quanto ha tutti i vantaggi connessi all'uso dell'information gain e allo stesso tempo consente di ottenere alberi non troppo "specifici".
L'algoritmo C4.5, attualmente più utilizzato, fa uso di questo criterio per la generazione dei rami dell'albero; tale algoritmo è l'evoluzione di ID3, altro sistema del medesimo autore, J.R.Quinlan.

3.4 Algoritmi per la costruzione degli alberi decisionali

In letteratura esistono diversi algoritmi per la costruzione di alberi decisionali; i più implementati nei pacchetti commerciali sono ID3, CART, C4.5, CHAID.
L'algoritmo ID3 (Quinlan, 1986) è stato studiato a lungo ed è il precursore di C4.5.
CART (Breiman e altri, 1984) è di particolare interesse dal momento che sono state implementate diverse varianti commerciali; inoltre è stato il primo sistema a introdurre gli alberi di regressione. Essenzialmente, gli alberi di regressione assumono la forma di alberi decisionali i cui nodi foglia hanno valori numerici, anziché categorici [3.2].
CART è molto simile all'algoritmo C4.5 ma ci sono alcune differenze da mettere in evidenza.

La prima differenza importante è che CART effettua sempre divisioni binarie dei dati senza tener conto della natura numerica o categorica delle variabili.
Una seconda differenza è che CART utilizza i dati di test per effettuare lo sfoltimento e poi generalizzare l'albero binario ottenuto, mentre C4.5 usa soltanto i dati di training per creare la struttura ad albero finale [3.3].
CHAID (Kass, 1980) è un altro algoritmo di creazione di alberi decisionali che si trova in pacchetti statistici commerciali, come SAS e SPSS.
CHAID differisce da CART e C4.5 perché opera solo con variabili categoriche inoltre ha un'impostazione prettamente statistica in quanto usa il test statistico X^2 per determinare le variabili adatte alla costruzione dell'albero decisionale.

3.5 L'algoritmo CART

Ogni algoritmo per la costruzione di alberi decisionali consta generalmente di due fasi: nella prima fase si procede alla generazione dell'albero, nella seconda fase si effettua il pruning.
Nella prima fase spesso si ottengono alberi completi ma molto complessi,nel senso che sono molto profondi e quindi costituiti da un numero elevato di regole. Si tende infatti a cercare di separare il più possibile elementi di classi differenti; in questo modo però la descrizione può essere troppo particolareggiata e causare un fenomeno di **overfitting.** Per evitare tale fenomeno è necessario definire una serie di criteri per semplificare gli alberi di decisione in modo da ottenere uno strumento valido di classificazione dei record in cui è stata eliminata la ridondanza cioè i rami meno significativi in quanto l'albero costruito può essere troppo legato ai dati di training che sono stati dati in pasto all'algoritmo.
L'algoritmo CART procede in questo modo: si parte dalla totalità dei record appartenenti al training set e si comincia una divisione binaria in classi. Il criterio sulla base del quale viene effettuata questa divisione si basa

sull'information gain e quindi mira ad ottenere il massimo guadagno di informazione ad ogni successiva suddivisione.

Poiché l'algoritmo CART effettua una divisione binaria, il concetto di information gain si particolarizza nel concetto di **diversità**. Questa grandezza è una funzione della frazione di record classificati in ciascuna classe infatti, indicata con p_c la frazione di record classificati nella classe c (stiamo parlando di classificazione binaria), è possibile definire una funzione diversità $D(c_1, c_2)$. Per fissare le idee supponiamo che la funzione assume valori in $[0,1]$; poiché essa fornisce una misura di quanto i record sono equamente distribuiti nelle classi, nei due casi limite i record sono rispettivamente concentrati solo in una classe se $D = 1$ oppure divisi in parti uguali fra le due classi se $D = 0$.

Se N è il numero di record del training set, una prima divisione binaria porterebbe alla formazione di due classi ognuna delle quali possiede una determinata frazione della totalità dei record.

Poiché l'obiettivo dell'algoritmo è quello di operare una classificazione posizionando i record in classi prestabilite (si ricordi la sostanziale differenza tra classificazione e clustering dove non si ha una specificazione iniziale delle classi), esso cercherà di posizionare i record in maniera "quanto più possibile equa" tra le diverse classi prestabilite e non considererà quei campi o quei valori dei campi che non portano ad una divisione equa dei record [3.4].

In termini formali, l'algoritmo nelle varie fasi sceglierà dei **campi splitter** (campi divisori presenti nel training set) e proverà diversi valori in modo che in ogni nodo venga minimizzata la funzione diversità e quindi massimizza la funzione:

$$F = D_{beforesplit} - \left(D_{leftchild} + D_{rightchild}\right)$$

Massimizzare tale funzione coincide col trovare nella varie fasi i campi e i valori dei campi che in quella fase operano la bipartizione più equa possibile.

L'algoritmo procede nel seguente modo:

1. **Sort:** per ogni campo si ordinano tutti i possibili valori in base a quanto permettono di diminuire la funzione diversità.
2. **Max**: fra i migliori splitter di ogni campo, si prende quello che più diminuisce la funzione diversità. Tale campo è scelto come splitter per il nodo.
3. Si procede iterando il punto 1 fin quando non si riesce a minimizzare ulteriormente la funzione diversità.

L'algoritmo descritto precedentemente genera tutto l'albero sulla base di un training set di record. **Per applicare efficacemente l'albero ottenuto ad altri dataset** è necessario eliminare alcuni rami, diminuendo l'errore complessivo dell'albero.
Per determinare quale sottoalbero utilizzare, si introduce un **tasso di errore aggiustato**:

$$A_t = E_t + \alpha \cdot \lambda_t$$

dove l'indice t indica il particolare sottoalbero e λ_t è il numero di foglie nel sottoalbero t. Il tasso di errore aggiustato tiene conto del numero di foglie λ_t, nonché del peso di tale numero (peso indicato con α); esso permette inoltre di effettuare confronti tra i sottoalberi e l'albero completo.
Se t=0 si ha l'albero completo e λ_0 è il numero di foglie dell'albero, A_0 è l'errore aggiustato dell'albero completo.
Esso coincide con E_0 solo se $\alpha = 0$ ed in generale se $\alpha = 0$ l'errore aggiustato di un qualunque sottoalbero coincide con E_t.
L'algoritmo di pruning è così articolato:

1. Si fissa α e si calcola A_t per ogni sottoalbero contenente la radice.

2. Si aumenta α da 0 a 1 fino a che per qualche sottoalbero t_1 risulti $A_{t1} \leq A_0$. Si eliminano tutti i rami che non fanno parte di t_1.
3. Si procede nello stesso modo fino a che non si arriva alla radice, ottenendo in tal modo un set di sottoalberi $\{t_1, t_2, ...\}$.
4. Si calcola l'errore di tutti i sottoalberi utilizzando un insieme di dati (test set) e si sceglie il sottoalbero con l'errore minore.

Sull'algoritmo CART si possono fare varie osservazioni computazionali infatti esso consiste nell'iterazione di una procedura di ordinamento ed una di ricerca di un massimo, per ogni nodo. Trattandosi di un albero (nel nostro caso binario), ogni nodo produce due rami che crescono indipendentemente e possono essere calcolati in parallelo.
Dal momento che il volume dei record che arrivano in ogni nodo diminuisce via via che il nodo si allontana dalla radice, potrebbe risultare determinante la possibilità di variare dinamicamente il numero di processori utilizzati, il tipo di template, ecc. Anche per la strategia di pruning è possibile pensare varie strategie di parallelizzazione, in ogni caso sia per la generazione che per il pruning, la parte più costosa sembra l'accesso ai dati, non tanto la computazione in se.

3.6 L'algoritmo C4.5

Tale algoritmo è uno dei più utilizzati per la costruzione degli alberi decisionali infatti esso è implementato nella maggior parte dei pacchetti software per DM come ad esempio WEKA.
Si basa sostanzialmente sull'algoritmo CART ma si differenzia da esso per degli aspetti che vanno sottolineati:

- L'albero non è binario; da ogni nodo partono un numero di rami pari a tutti i possibili valori che possono essere assunti dal campo splitter per quel nodo.
- Il criterio utilizzato per decidere quale campo utilizzare come splitter in ogni nodo si basa sul concetto di GainRatio.
- La procedura di pruning si basa sull'attribuzione di un errore previsto, ad ogni foglia. In pratica questo errore previsto viene assegnato ipotizzando che l'errore commesso sul training set in corrispondenza di una data foglia, sia il massimo che si può ottenere. Non viene calcolato un tasso di errore aggiustato, come viene fatto nell'algoritmo CART, utilizzando il test set ma si calcola l'errore previsto basandosi sempre sul training set, ciò consente anche di contenere il problema dell'overfitting in quanto consente di dare all'albero una maggiore generalità e quindi di non essere eccessivamente legato ai dati di training ma anche di ottenere vantaggi in termini di costo computazionale.

L'algoritmo C4.5 introduce quindi delle migliorie importanti che lo rendono adatto ad un numero di applicazioni sempre maggiori. La prima, e più importante di queste modifiche, permette di lavorare su attributi qualsiasi, in particolare interi, reali e dati non supportati da altri algoritmi, ma introduce anche una serie di problemi che vengono affrontati di volta in volta in maniera differente.
In particolare esistono diverse revisioni della funzione di costo che si occupa di scegliere la miglior discretizzazione degli attributi continui e il miglior punto in cui effettuare la suddivisione dell'albero.
Poiché le ultime versioni di questo algoritmo sono state rilasciate come prodotto commerciale, non è più disponibile una descrizione approfondita delle tecniche usate al suo interno ma soltanto un elenco di miglioramenti e nuove funzionalità.

3.7 Il problema dei test

Un altro aspetto fondamentale degli alberi di decisione è capire in che modo vanno fatti i test sugli attributi; è evidente infatti che a seconda della tipologia di attributo si ricorre ad un test appropriato.
L'algoritmo C4.5 prevede tre tipi di test possibili:

- Attributo discreto: un risultato per ogni valore.
- Attributo discreto: un risultato per ogni gruppo di valori.
- Attributo continuo: due possibili risultati (test binario).

Inoltre c'è anche il vincolo che almeno due tra i test devono contenere un numero minimo di esempi.
Qualora il test su un attributo discreto venga fatto nel primo modo, esso è definito completamente invece quando si utilizza un risultato per un gruppo di valori, bisogna stabilire quanti e quali gruppi considerare indagando sul dominio.
Il test su un attributo continuo viene fatto scegliendo un valore di soglia Z pari ad uno degli m valori assunti da un attributo A; tale valore suddivide gli m valori in due gruppi. Viene fatta una valutazione della soglia da scegliere infatti una variazione della soglia comporta una variazione del partizionamento e quindi dello SpliInfo e ciò comporta una tendenza a favorire test con molti risultati.
Si nota quindi, poiché vanno fatte delle valutazioni in più, che c'è un peggioramento delle prestazioni in problemi con molti attributi continui.

3.8 Accuratezza degli alberi di decisione

Vogliamo qui introdurre i parametri che consentono di capire se l'albero decisionale costruito classifica correttamente o anche, se sbaglia, quando costa l'errore commesso.

Indichiamo con n_i^* il numero totale di record del training set che terminano nella foglia i e con n_i il numero di record classificati correttamente in i, è possibile associare ad ogni foglia i un errore ε_i così definito:

$$\varepsilon_i = \frac{n_i}{n_i^*}$$

Si può così definire l'errore associato all'intero albero; esso infatti si ottiene come somma degli errori di tutte le foglie, pesati rispetto alla probabilità che un record finisca su ciascuna foglia. Se il numero totale di record del training set è pari ad N, la probabilità che un record finisca nella foglia i è data da n_i / N per cui si ha la seguente espressione dell'**errore associato all'albero**:

$$E = \sum_i \frac{n_i}{N} \cdot \varepsilon_i$$

La condizione di ottimo si ha quando $E = 1$ infatti in tal caso l'albero non sbaglia mai nel senso che classifica correttamente tutti i record del training set perché l'errore associato ad ogni singola foglia è pari ad uno ed E risulta uguale alla probabilità che un record cada in una qualsiasi foglia cioè ad uno. Questo tipo di valutazione delle potenzialità di un albero non è sempre appropriata perché spesso è necessario valutare l'entità dell'errore; ad esempio diagnosticare un tumore come benigno quando in realtà è maligno risulta sicuramente più pericoloso del caso contrario.

Le **matrici di costo** sono strumenti più sofisticati che consentono di individuare le differenze negli errori di classificazione, in pratica esse consentono di differenziare i falsi positivi dai falsi negativi cioè di separare gli errori che non generano grossi problemi dagli errori che possono portare a commettere gravi errori

Ogni elemento non nullo della matrice rappresenta il costo di un certo errore di classificazione, ovviamente quindi una matrice di costo avrà la diagonale principale tutta nulla infatti quegli elementi rappresentano i casi di corretta classificazione dei record.

Generalmente tali matrici sono non simmetriche perché il costo di un'errata classificazione di un elemento nella classe i piuttosto che nella vera classe j è diverso quasi sempre dal costo di errata classificazione opposto.

3.9 Come limitare l'overfitting.

Il problema dell'overfitting non va sottovalutato perché lede all'obiettivo primario legato alla costruzione degli alberi. Uno dei metodi più semplici per non incorrere in questo problema è fermare la crescita dell'albero prima che raggiunga il punto in cui classifica perfettamente i dati di training (prepruning) oppure lasciar crescere l'albero e poi potarlo successivamente [3.5].

Come determinate allora la dimensione ottimale dell'albero? I metodi possibili sono tre:

- Utilizzare un insieme di esempi separato per valutare l'albero che si sta costruendo (approccio "training and validation set")
- Utilizzare tutti i dati disponibili per il training ma applicare un test statistico per stabilire se tenere o meno un nodo
- Utilizzare una misura esplicita della complessità di codificare l'albero di decisione o gli esempi di training che sono eccezioni e fermare la

crescita quando questa misura è minimizzata (principio di minima lunghezza di descrizione)

3.10 Generazione delle regole produttive

Un qualsiasi albero decisionale può essere tradotto in un insieme di regole produttive, che hanno la seguente forma:

IF *condizioni antecedenti*
THEN *condizioni conseguenti*

Le condizioni antecedenti definiscono i valori o gli intervalli di valori di uno o più attributi di output. La tecnica per trasformare un albero decisionale in un insieme di regole produttive è semplice. Una regola è creata partendo dal nodo radice e seguendo uno dei possibili rami fino al nodo foglia.
La condizione antecedente di una regola è data dalle combinazioni dei valori degli attributi che si incontrano lungo il percorso.
La condizione conseguente della corrispondente regola è il valore del nodo foglia.
Esistono comunque diversi modi per passare dagli alberi alle regole, molte di queste varianti tendono a semplificare oppure a eliminare le regole esistenti, ciò è fatto per eliminare regole non interessanti o ridondanti.
Con riferimento all'albero della figura 3.1, si ottengono le seguenti regole produttive:

1. **IF** ((outlook=sunny) and (humidity=high)) **THEN** play=no
2. **IF** ((outlook=sunny) and (humidity=normal)) **THEN** play=yes
3. **IF** (outlook=overcast) **THEN** play=yes
4. **IF** ((outlook=rainy) and (windy=true)) **THEN** play=no
5. **IF** ((outlook=rainy) and (windy=false)) **THEN** play=yes

3.11 Considerazioni finali

Gli alberi decisionali sono molto utilizzati in vari campi di applicazione del DM in quanto comportano numerosi vantaggi. Essi sono infatti facilmente implementabili e sono facilmente trasformabili in un insieme di regole produttive, non fanno alcuna ipotesi preventiva sulla natura dei dati, sono applicabili con successo a problemi reali e infine sono in grado di costruire modelli a partire da dataset che contengono dati numerici e categorici.
Essi, inoltre, consentono di trattare dati di dimensionalità elevata.
Nonostante ciò, ci sono degli elementi di critica all'uso degli alberi di decisione nei problemi di classificazione:

- Le variabili di output devono essere categoriche, inoltre non è possibile utilizzare più di una variabile di output.
- Gli algoritmi degli alberi decisionali sono **instabili** in quanto piccole variazioni nei dati di training possono indurre a modificare le variabili scelte per ogni punto all'interno dell'albero. Gli effetti possono essere significativi perché tutta la struttura dell'albero dipende dalla scelta delle variabili.
- Gli alberi creati da dataset numerici possono essere molto complessi poiché le suddivisioni delle variabili per i dati numerici sono tipicamente binarie.
- L'attributo divisore è scelto quasi sempre da un algoritmo "ingordo", che non tiene conto dell'influenza che la scelta dell'attributo potrebbe avere sui futuri divisori. La decisione avviene ad ogni nodo dell'albero, in un preciso momento e non viene più riconsiderata in seguito. Quello che succede è che tutti i campi divisori vengono scelti sequenzialmente e ogni campo divisore è di fatto dipendente dai precedenti. Quindi tutti i campi divisori dipendono dal nodo radice dell'albero a tal punto che una modifica del campo divisore del nodo radice potrebbe portare alla costruzione di un albero completamente differente.

CAPITOLO 4

Strategie di DM non supervisionato:

il clustering

4.1 Introduzione alle tecniche di clustering

Abbiamo già parlato della distinzione tra apprendimento supervisionato e non supervisionato e abbiamo visto che nel secondo caso c'è la completa assenza di una variabile di output; di conseguenza, tutti gli attributi utilizzati per costruire i modelli sono variabili indipendenti. Nella clusterizzazione non supervisionata non esiste una variabile dipendente che guida il processo di apprendimento ma l'obiettivo è quello di scoprire le strutture concettuali dei dati, non sapendo precisamente questo a cosa porterà. Il clustering è una tecnica di analisi dei dati volta alla selezione e raggruppamento di elementi omogenei in un insieme di dati; esso è il processo di raggruppamento di un insieme di oggetti fisici o astratti in classi di oggetti simili (Han 2001), un cluster non è altro che una collezione di oggetti simili tra loro, che sono dissimili dagli oggetti degli altri cluster. Quello che si fa è quindi cercare raggruppamenti "interessanti" nei dati. La bontà del risultato dipende dal tipo di misura scelto. Definita la distanza, si ottiene una matrice delle distanze che è ovviamente simmetrica, in quanto la distanza dell'elemento i dall'elemento j è ovviamente uguale a quella di j d i, e inoltre ha la diagonale nulla. Tutte le tecniche di clustering si basano sul concetto di similarità e quindi di distanza tra due elementi per cui la bontà delle analisi ottenute dipende essenzialmente da quanto è significativa la metrica e quindi da come è stata definita la distanza infatti gli algoritmi raggruppano gli elementi a seconda della distanza e quindi l'appartenenza o meno ad un insieme dipende da quanto l'elemento preso in esame è distante dall'insieme [4.1].

Le tecniche di clustering possono procedere seguendo due filosofie molto diverse:

- Dal basso verso l'alto: si procede supponendo che inizialmente tutti gli elementi siano considerati cluster a se e poi l'algoritmo provvede a

raggruppare i cluster più vicini. L'algoritmo continua ad unire elementi al cluster fino ad ottenere un numero prefissato di cluster oppure fino a che la distanza minima fra i cluster non supera un certo valore.

- Dall'alto verso il basso: Tutti gli elementi sono inizialmente raggruppati in un unico cluster e poi l'algoritmo inizia a dividere il cluster in tanti altri di dimensioni inferiori. Il criterio che guida la divisione è sempre quello di cercare di ottenere elementi omogenei. L'algoritmo procede fin quando non raggiunge un numero prefissato di cluster.

Inoltre vi sono due tipologie di clustering: il **distance based clustering,** caratterizzato dal fatto che un gruppo di oggetti appartengono allo stesso cluster se sono vicini rispetto ad una determinata "distanza"; il **conceptual clustering,** caratterizzato dal fatto che un gruppo di oggetti appartengono ad un cluster se questo definisce un "concetto" comune ai diversi oggetti. Le applicazioni della clusterizzazzione non supervisionata sono molteplici infatti essa serve sia a determinare relazioni significative nei dati, sia a valutare le prestazioni di un modello non supervisionato, sia a determinare l'insieme ideale di attributi di input per l'apprendimento supervisionato, sia a identificare gli outlier. In biologia può essere utilizzato per derivare tassonomie di animali e piante, nel marketing per derivare e caratterizzare gruppi di consumatori, nell'analisi dei dati viene impiegato per studiare come i dati si distribuiscono nello spazio. Nonostante tutti i vantaggi di questa tecnica di DM, vi sono due motivi importanti che ne limitano l'applicazione:

- I metodi di clustering non sono efficienti quando si hanno di fronte i databases tipici delle applicazioni di DM; tali database hanno infatti dimensioni enormi, dell'ordine dei terabyte, con migliaia di record, ciascuno dei quali consta di un centinaio di attributi.
- I metodi di clustering non prevedono la presenza di variabili categoriche con il loro trattamento relativo; di conseguenza il modo tradizionale di trattare le variabili categoriali come variabili numeriche

non sempre produce risultati significativi, anche per il fatto che le modalità di una variabile categoriale non sono ordinate.

Sono state trovate comunque delle soluzioni che consentono di aggirare, almeno in parte, questi problemi.

4.2 L'algoritmo delle K-medie

L'algoritmo delle K-medie (Lloyd 1982) è una tecnica statistica semplice ma efficace per effettuare del clustering a partizioni.
Tale algoritmo suddivide un insieme di dati in cluster disgiunti ed è semplice da capire ed implementare [4.2]. I passi dell'algoritmo possono essere così schematizzati:

1. Si sceglie il numero totale K di cluster da determinare.
2. Si scelgono, in maniera casuale, k osservazioni nel dataset. Esse saranno i centri iniziali dei cluster.
3. Si utilizza una distanza per assegnare le restanti osservazioni ai centri dei cluster più vicini.
4. Si utilizzano le osservazioni contenute in ogni cluster per calcolare una nuova media di ogni cluster.
5. Se le nuove medie sono identiche a quelle calcolate all'iterazione precedente, l'algoritmo termina altrimenti le nuove medie sono utilizzate come centri dei cluster e si ripetono i passi da 3 a 5.

L'algoritmo quindi richiede una scelta iniziale del numero di cluster che riteniamo siano presenti nei dati ma anche del criterio di similitudine da utilizzare per assegnare le osservazioni ad uno dei cluster; in genere si usa la semplice distanza euclidea.

Il processo di clustering continua fin quando l'algoritmo non mostra ulteriori variazioni dei centri dei cluster.

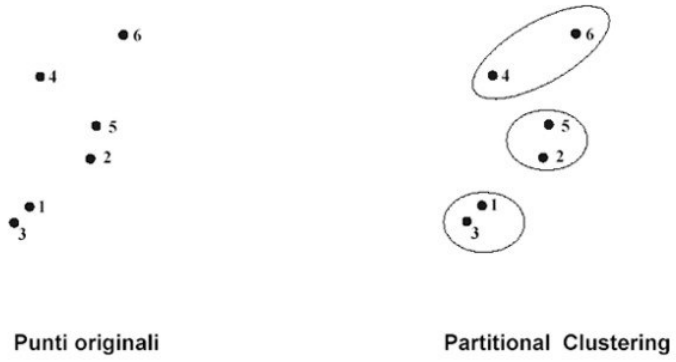

Figura 4.1: Il clustering a partizioni

Uno dei limiti di questo procedimento è che la configurazione finale dei cluster potrebbe essere diversa per ogni scelta alternativa dei centri iniziali dei cluster e quindi, anche se l'algoritmo garantisce la creazione di cluster stabili, non è detto che l'assetto finale, per quanto stabile, sia ottimale.
Ci sono delle altre problematiche connesse all'algoritmo delle K-medie:

- Non esiste un criterio per capire quali variabili siano significative e quali no nella determinazione dei cluster per cui variabili irrivelanti potrebbero portare a risultati insoddisfacenti.
- L'algoritmo può operare solo con valori reali per cui se è presente una variabile categorica nel dataset, essa deve essere eliminata oppure essere soggetta a trasformazioni in valori numerici equivalenti.
- Bisogna scegliere il numero K di cluster da determinare. Un metodo per superare questo problema consiste nell'eseguire più volte l'algoritmo con diversi valori di K in modo da farsi un'idea sul numero di cluster presenti nei dati.
- L'algoritmo funziona bene quando i cluster presenti nei dati hanno dimensioni quasi uguali. Se invece la soluzione migliore è

rappresentata da cluster di dimensioni diverse, l'algoritmo delle K medie non sempre è in grado di trovare la soluzione ideale.

- Abbiamo noi la responsabilità di dare significatività ai cluster cioè di interpretare i risultati ottenuti. Spesso si utilizzano tecniche di DM supervisionato per capire meglio la natura dei cluster generati.

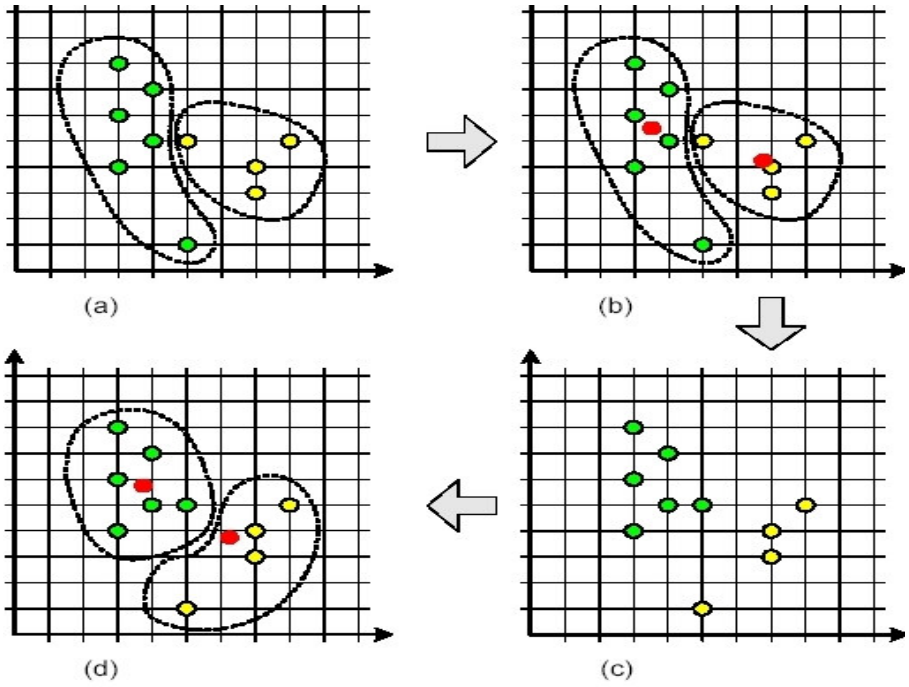

Figura 4.2: Successione di passi di un algoritmo a partizioni

Nonostante tutti questi limiti, l'algoritmo delle K-medie è molto utilizzato in quanto facilmente implementabile ed efficiente anche nel caso di grossi dataset.

Il pregio di questo algoritmo è proprio quello di essere relativamente efficiente in quanto ha una complessità dipendente sia dal numero di iterazioni, che dal numero di cluster, che dal numero di oggetti del dataset ma essendo i primi due molto più piccoli del terzo, si può dire che la complessità è

$O(n)$. L'algoritmo spesso termina in un ottimo locale; l'ottimo globale si può rincorrere utilizzando algoritmi genetici.

Tra i difetti di questo algoritmo sicuramente va evidenziato che bisogna specificare in anticipo il numero di cluster che si vogliono formare inoltre è molto sensibile alla presenza di rumore e di outlier.

Esso non è adatto per cluster con forme non convesse, non ipersferici e di dimensioni non uniformi infatti in tal caso il clustering ottenuto non è di buona qualità e soffre di vari problemi.

Il più grosso inconveniente è l'impossibilità di trattare variabili categoriali, tuttavia l'algoritmo delle K-medie è un buon punto di partenza per la costruzione di algoritmi che gestiscano anche dati di tipo categoriale, salvaguardando l'efficienza.

Un esempio è costituito dall'algoritmo dei K-prototipi, chiamato così perché gli oggetti o record sono raggruppati a valle di un confronto con k oggetti prototipi invece che con le k medie di gruppo.

4.3 Trattamento delle variabili categoriali

L'algoritmo dei K-prototipi (Kaufman e Rousseeuw 87) opera su variabili categoriche, esse però vanno opportunamente trattate [4.3],[4.4].

Indichiamo con $X = \{X_1, X_2, ..., X_n\}$ un set di n oggetti x_i, ciascuno dei quali costituito da m attributi x_{ij}.

L'obiettivo di un algoritmo di clustering è quello di raggruppare ciascuno degli n oggetti in uno dei K cluster; la soluzione più comune per perseguire questo scopo è scegliere un criterio di raggruppamento che possa guidare l'utente nella ricerca della partizione e quindi dei gruppi che da essa derivano.

Un criterio molto utilizzato è scegliere una "funzione di costo":

$$E = \sum_{l=1}^{k} \sum_{i=1}^{n} y_{il} \cdot d(X_i, Q_l)$$

dove:

- $Q_l = [q_{l1}, q_{l2}, ..., q_{lm}]$ è un "**vettore rappresentativo**" o "**prototipo**" per il cluster l
- y_{il} è il generico elemento di una matrice di partizione Y
- d è una misura di similarità, spesso definita come distanza euclidea in uno spazio m-dimensionale

La matrice Y di dimensione $[n * k]$ è la matrice di partizione, poiché ad essa è associata un'unica partizione degli n oggetti in k gruppi; essa è caratterizzata dalle seguenti proprietà:

1. $0 \leq y_{il} \leq 1$
2. $\sum_{l=1}^{k} y_{il} = 1$

Supponiamo, in particolare, di utilizzare una **hard partition** cioè $y_{il} \in \{0,1\}$ per cui, se $y_{il} = 1$ l'oggetto X_i è assegnato al cluster l.

La sommatoria interna della funzione di costo rappresenta il costo totale E_l di assegnazione degli oggetti presenti in X al cluster l cioè la dispersione totale degli oggetti nel cluster rispetto al suo prototipo.

L'espressione E_l è minimizzata se:

$$q_{lj} = \frac{1}{n_l} \cdot \sum_{i=1}^{n} y_{il} \cdot x_{ij} \qquad j = 1,2,...,m$$

dove n_l è il numero di oggetti nel cluster l e si può ottenere come somma degli elementi della colonna l della matrice di partizione.

Nel caso in cui siano presenti variabili categoriali , bisogna sostituire la misura di similarità con una misura alternativa:

$$d(X_i, Q_l) = \sum_{j=1}^{m_r} \left(x_{ij}^r - q_{lj}^r\right)^2 + \gamma_l \cdot \sum_{j=1}^{m_c} \delta\left(x_{ij}^c, q_{lj}^c\right)$$

dove:

- $\delta(p,q) = 0$ per $p = q$
- $\delta(p,q) = 1$ per $p \neq q$
- La prima sommatoria opera su valori di attributi numerici; la seconda opera su modalità di attributi categoriali per l'oggetto i ed il prototipo di cluster l.
- m_r ed m_c rappresentano rispettivamente il numero degli attributi numerici e categoriali.
- γ_l rappresenta il peso relativo all'importanza degli attributi categoriali per il cluster l.

Tramite questo nuovo tipo di misura, si ottiene una nuova espressione per E_l:

$$E_l = \sum_{i=1}^{n} y_{il} \cdot \left(\sum_{j=1}^{m_r} \left(x_{ij}^r - q_{lj}^r\right)^2 + \gamma_l \cdot \sum_{j=1}^{m_c} \delta\left(x_{ij}^c, q_{lj}^c\right) \right) \Rightarrow E_l = E_l^r + E_l^c$$

per cui E_l può esprimersi come somma di due contributi, uno relativo al costo totale su tutti gli attributi numerici di oggetti nel cluster l che viene minimizzato se q_{lj}^r, al variare di j è scelto nel modo su visto, l'altro relativo agli attributi categoriali.

Indichiamo ora con C_j l'insieme delle modalità che può assumere l'attributo categoriale j e con $p(c_j \in C_j \mid l)$ la probabilità che la modalità c_j si verifichi

nel cluster l; possiamo riscrivere il contributo relativo alle variabili categoriali nel seguente modo:

$$E_l^c = \gamma_l \cdot \sum_{j=1}^{m_c} n_l \cdot \left(1 - p\left(q_{lj}^c \in C_j \mid l\right)\right)$$

Tale quantità è minimizzata se e solo se $p\left(q_{lj}^c \in C_j \mid l\right) \geq p\left(c_j \in C_j \mid l\right)$, per $q_{lj}^c \neq c_j$, per tutte le variabili categoriali.

La funzione di costo totale può essere così scritta:

$$E = \sum_{l=1}^{k} \left(E_l^r + E_l^c\right) = E^r + E^c$$

Poiché entrambi i due contributi sono non negativi, la minimizzazione può essere ottenuta minimizzando separatamente il costo totale degli attributi numerici e categoriali rispettivamente su tutti i cluster.

Nel caso più semplice, la misura di similarità è definita tramite una distanza euclidea per le variabili numeriche e dal numero di ineguaglianze tra oggetti e prototipi di cluster per le variabili categoriali. Un ruolo fondamentale è rivestito dal peso γ_l in quanto esso consente di distribuire in modo opportuno l'importanza delle due tipologie di attributi nel cluster l. Se il peso è nullo il raggruppamento dipende solo dagli attributi numerici altrimenti, a seconda del suo valore, è più o meno orientato verso gli attributi categoriali.

Come scegliere allora γ_l in un algoritmo di clustering?

La scelta dipende dalla distribuzione degli attributi numerici; essa è fortemente legata al valore di σ_l, cioè della deviazione standard media degli attributi numerici nel cluster l, ma esso è generalmente incognito prima delle operazioni di clustering. Al suo posto si può utilizzare la deviazione standard media della totalità degli attributi numerici, indicata con σ.

Inoltre, in un algoritmo di tipo iterativo, γ_l può essere calcolato in base ad una precedente operazione di clustering.

4.4 L'algoritmo dei K-prototipi

I passi principali di questo algoritmo sono i seguenti:

1. Selezionare k prototipi iniziali dal dataset X
2. Allocare ogni oggetto in X al cluster di cui il prototipo è più vicino secondo i criteri su visti e ridefinire il prototipo del cluster dopo ogni allocazione
3. Dopo che tutti gli oggetti sono stati allocati ai rispettivi cluster, si testa nuovamente la similarità di tali oggetti con tutti i prototipi di cluster. Se un oggetto risulta più vicino al prototipo di cluster cui non appartiene, riallocare l'oggetto a questo nuovo cluster e ridefinire nuovamente i prototipi di entrambi i cluster.
4. Ripetere il passo 3 finchè nessun oggetto subisce il cambiamento di cluster.

L'algoritmo è molto simile a quello delle K-medie ed è costruito dividendo il processo in tre sottoprocessi: "selezione iniziale dei prototipi", "allocazione iniziale" e "riallocazione".
Il cuore dell'algoritmo è costituito dall'operazione di allocazione iniziale che, partendo da un set iniziale di prototipi, assegna ogni oggetto ad un cluster e ridefinisce il prototipo di cluster in seguito a ciascuna assegnazione di un oggetto al cluster.

Vediamo ora la struttura di questo sottoprocesso:

```
for  i=1  to  Numero di oggetti
     Mindistance =Distance( X[i],O_prototypes[1] )+
      gamma * Sigma( X[i],C_prototypes[1] )
    for  j=1  to  Numero di clusters
         distance =Distance( X[i],O_prototypes[j] )+
          gamma * Sigma( X[i],C_prototypes[j] )
          if (distance<Mindistance)
            Mindistance =distance
            Cluster =j
          endif
    endfor
    Clustership[i] =cluster
    ClusterCount[cluster]+1
     for  j=1  to  Numero di attributi numerici
         SumInCluster[cluster,j]+X[i,j]
         O_prototypes[cludter,j] =SumInCluster[cluster,j]/ClusterCount[cluster]
      endfor
      for  j=1  to  Numero di attributi categoriali
          FrequencyInCluster[cluster,j,X[i,j]]+1
          C_prototypes[cluster,j] =HighestFreq(FrequencyInCluster,cluster,j)
      endfor
endfor
```

La fase di riallocazione ha caratteristiche molto simili al precedente algoritmo con la sola eccezione che, dopo la riallocazione di un oggetto, i prototipi per entrambi i cluster, sia quello di appartenenza iniziale che quello finale, devono essere ridefiniti.

Sono state fatte svariate prove su differenti dataset per avere un'idea delle performance di tale algoritmo e si è notato che esso produce delle curve di convergenza aventi un comportamento più o meno standard.

Implementando l'algoritmo in Matlab, si è notato infatti che nelle prime iterazioni si riduce molto rapidamente il numero di oggetti che cambiano cluster di appartenenza; successivamente la velocità di riduzione si abbassa fino al punto in cui il numero di oggetti che cambiano i cluster di appartenenza resta costante nonostante le ulteriori iterazioni. In seguito, dopo un breve aumento, tale numero tende definitivamente a zero. In figura 4.3 è rappresentato tale andamento; sull'asse delle ascisse abbiamo il numero di iterazioni mentre su quello delle ordinate il numero di oggetti che modificano il cluster di appartenenza ad ogni iterazione.

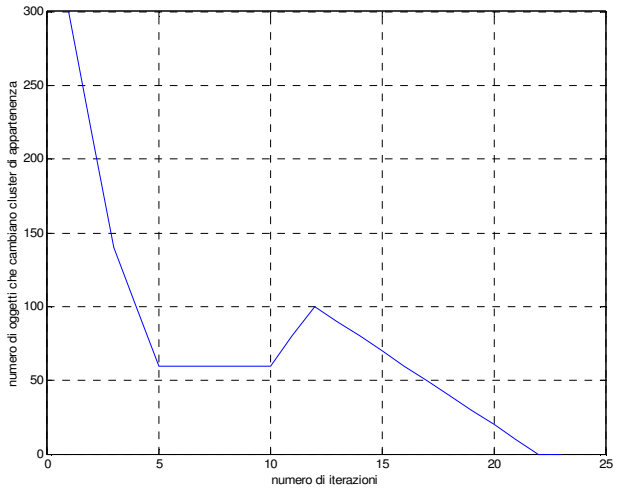

Figura 4.3: Andamento del numero di oggetti che cambiano cluster ad ogni iterazione

Ciò suggerisce che il processo di riallocazione può essere forzato a terminare prima, quando ad esempio la velocità di abbassamento del numero diventa piccola o magari è piccolo il numero di oggetti che modificano i cluster.
Questi accorgimenti possono aiutare a ridurre il tempo necessario per l'esecuzione dell'algoritmo risultando in tal modo utili nel contesto di grossi dataset.

4.5 *Il clustering gerarchico*

Quando si effettua una classificazione di tipo gerarchico i dati non vengono partizionati in un certo numero di cluster in un passo solo ma in fasi successive. L'approccio utilizzato è completamente diverso dal clustering classico; comunque il punto di partenza può essere un unico cluster contenente tutti gli individui oppure n cluster ciascuno contenente un solo individuo [4.5]. Nel primo caso l'algoritmo di clustering procede per partizioni successive e si parla di **metodi divisivi** (strategia top-down), nel secondo caso procede per fusioni successive e si parla di **metodi agglomerativi** (strategia bottom-up).

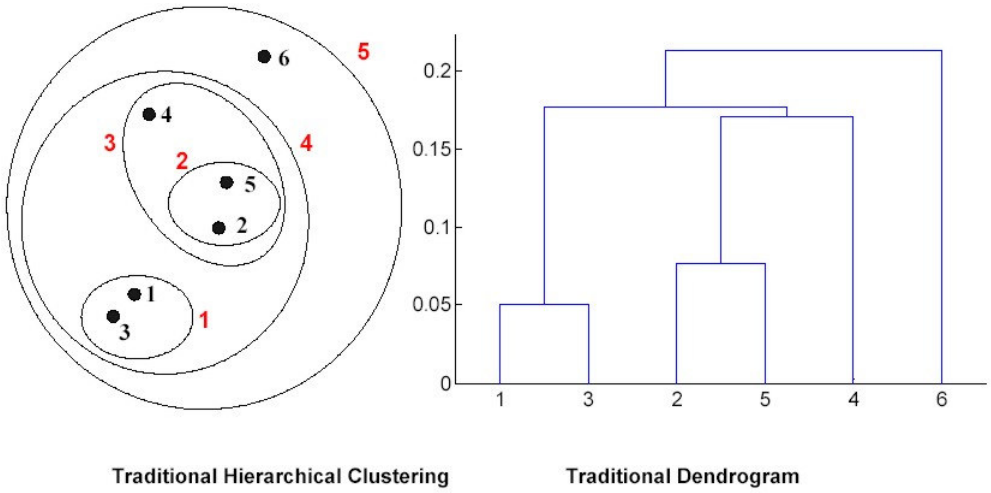

Figura 4.4: Il clustering gerarchico

Lo scopo di entrambe le varianti è il medesimo: arrivare, tramite suddivisioni o sintesi successive, ad una partizione ottimale, operando su una matrice di prossimità di qualche tipo.

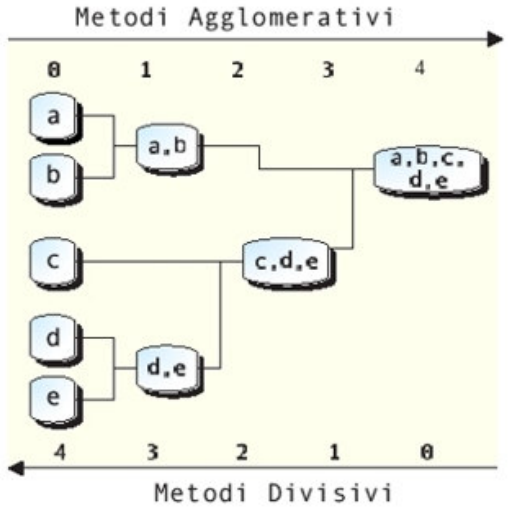

Figura 4.5: Differenza tra clustering agglomerativo e divisivo

La caratteristica principale degli algoritmi gerarchici è che le divisioni o le fusioni sono irrevocabili: i metodi agglomerativi consentono solo fusioni, quelli divisivi solo partizioni e non sono ammessi algoritmi di tipo misto quindi se due individui sono stati assegnati a cluster diversi, non potranno in seguito essere nuovamente membri di uno stesso cluster.

Ovviamente bisogna definire una nozione di distanza tra cluster simile alla distanza tra oggetti; in questo caso va però definita una distanza massima, una distanza minima ed infine una distanza media tra i vari cluster. A seconda del tipo di algoritmo verrà usata un tipo di distanza piuttosto che un'altra.

L'algoritmo **AGNES** (agglomerative nesting) e DIANA (divisive analysis), introdotti da Kaufman e Rousseeuw nel 1990 e implementati nei software statistici come SPLUS, utilizzano la distanza minima.

Le classificazioni gerarchiche possono essere rappresentate da un diagramma bidimensionale detto **dendrogramma**, che illustra le fusioni/divisioni verificatesi ad ogni stadio dell'analisi.

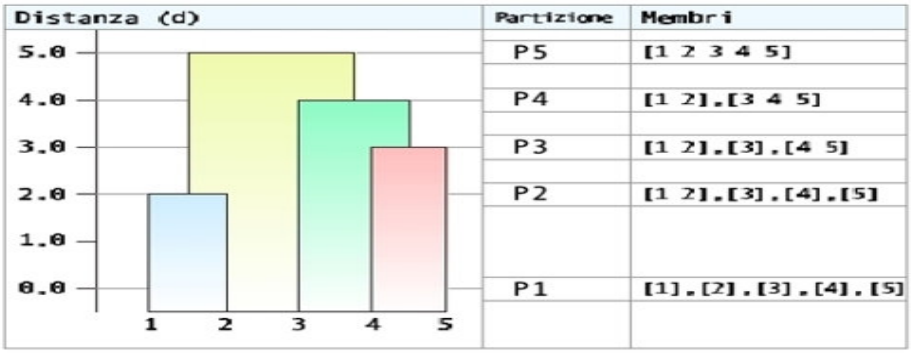

Figura 4.6: Esempio di dendrogramma

Ogni foglia del dendrogramma rappresenta un oggetto-istanza mentre ogni nodo interno rappresenta un cluster.

Il vantaggio degli algoritmi gerarchici è dovuto al fatto che essi non richiedono la specificazione del numero di cluster da formare, anche se poi bisogna stabilire un criterio per capire quando fermare l'algoritmo.

Il difetto principale è sicuramente è la non scalabilità e la complessità infatti essa è quadratica $O(n^2)$.

Inoltre la qualità dei raggruppamenti soffre del fatto che, una volta effettuata una divisione o un raggruppamento, non è possibile disfarla [4.6].

Le tecniche di classificazione gerarchiche sono molto utilizzate in ambito biologico, sociologico e in biblioteconomia e in tutti quei settori in cui è implicita una struttura gerarchica nei dati.

Sebbene il clustering gerarchico possa rivelarsi molto utile anche dove non sia presente una sottostante struttura gerarchica, allo scopo di non affrontare problematiche prestazionali è preferibile usare tali metodi solo se strettamente necessario. Poichè la scansione multipla dei dati costituisce il "collo di bottiglia" di questi algoritmi, si risolve il problema realizzando una specie di preclustering e quindi si migliorano le prestazioni, in termini di tempo o meglio

di complessità di I/O che diventa leggermente superiore ad una scansione singola, se si lavora su grandi database. I sotto-cluster generati in questa fase di preclustering possiedono un livello di accuratezza e sono in numero tale da poter essere contenuti in memoria principale.

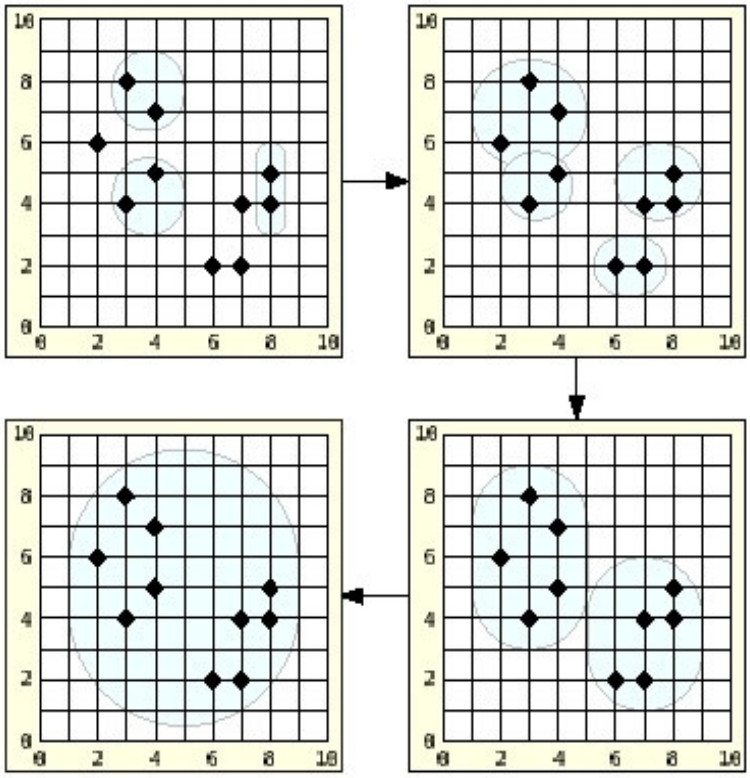

Figura 4.7: Clustering gerarchico agglomerativo

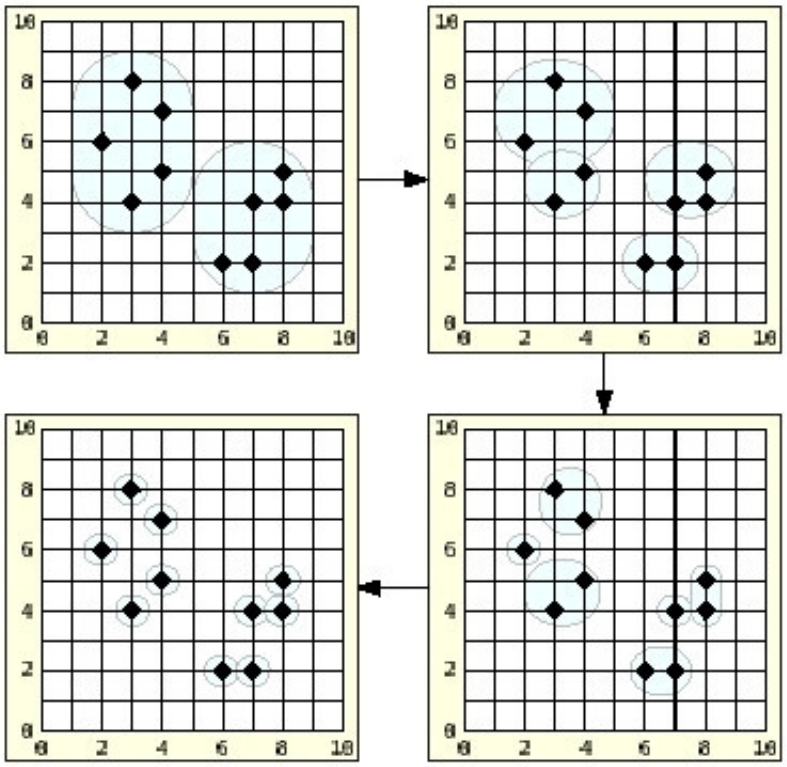

Figura 4.8: Clustering gerarchico divisivo

4.6 *Il clustering basato sulla densità*

Questo tipo di clustering non si basa sul concetto di distanza ma sul concetto di densità; il cluster viene considerato come una zona densa di istanze nello spazio dei dati, separato dagli altri cluster da zone povere di istanze [4.7].

Di solito i metodi basati sulla densità utilizzano due parametri: ε e $MinPts$.

L'ε-intorno di un oggetto è l'insieme, nello spazio dei dati, che sta in un cerchio di raggio ε centrato nell'istanza.

Se l'ε-intorno di un oggetto contiene un numero di altri oggetti maggiore di $MinPts$ (minimo numero di punti nell'intorno), l'oggetto è chiamato "core object".

Un oggetto p è **direttamente raggiungibile** da q se p sta nell'ε-intorno di q e se q è un core object; lo stesso oggetto p si dirà invece **raggiungibile** da q se c'è una catena di punti $p_1, p_{2,}...., p_n$ con $p_1 = p$ e $p_n = q$ e tali che p_{i+1} è direttamente raggiungibile da p_i.

Due oggetti p e q sono **connessi** se c'è un punto o tale che p e q sono raggiungibili da o.

L'algoritmo, che utilizza questi concetti, più usato è il DBSCAN (KDD 1999). Esso definisce un cluster come un insieme di oggetti connesso massimale cioè tale che non esista un insieme connesso più grande.

L'algoritmo, partendo da un punto arbitrario:

- Genera un cluster per ogni punto p che è un core object.
- Iterativamente, per ogni cluster C, considera i punti che sono direttamente raggiungibili da C. Inserisce poi questi punti nel cluster C ed eventualmente fonde insieme due cluster.

Il vantaggio di questo algoritmo, rispetto a quelli gerarchici è avere una complessità $O(n \cdot \log n)$.

L'altro vantaggio è che DBSCAN consente di identificare cluster di forma qualunque, non solo convessa; nonostante ciò esso presenta i seguenti svantaggi:

1. Sensibilità elevata rispetto ai due parametri che utilizza; essi inoltre sono difficili da determinare.
2. Introduce pesanti costi di elaborazione di I/O in quanto non effettua nessuna forma di preclustering, lavorando pertanto sull'intero database. Questo non lo rende adatto ad elaborazioni su grandi volumi di dati.
3. Può fondere erroneamente due cluster che siano congiunti da una stretta ma densa linea di punti (fenomeno del **chaining**); in questi casi diventa fondamentale la scelta dei parametri.

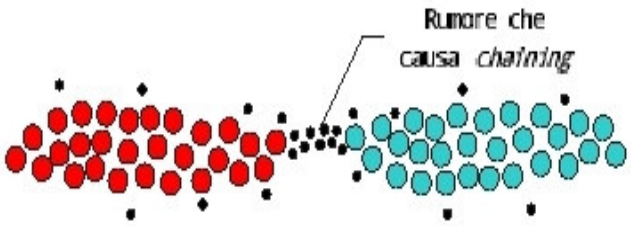

Figura 4.9: Fenomeno del chaining.

4.7 Gli algoritmi genetici

Abbiamo già visto che uno dei limiti di alcuni algoritmi di clustering, come quello delle K-medie, è il loro terminare in un ottimo locale e non globale; l'ottimo globale si può rincorrere utilizzando gli algoritmi genetici.

Gli algoritmi evolutivi, che includono algoritmi genetici, programmazione genetica, strategie evolutive e programmazione evolutiva, sono strategie euristiche che imitano i processi di evoluzione naturale per risolvere problemi anche con scarsa conoscenza del dominio.

Gli algoritmi evolutivi sono strategie euristiche che si ispirano all'evoluzione naturale, teorizzata da Darwin nel suo libro sull'evoluzione della specie, per risolvere problemi di ricerca globale. Questo tipo di algoritmi si basa sul concetto darwiniano che gli elementi più "adatti" all'ambiente hanno maggiore possibilità di sopravvivere e di trasmettere le loro caratteristiche ai successori; in pratica, si ha una popolazione di individui che evolvono di generazione in generazione attraverso meccanismi simili alla riproduzione sessuale e alla mutazione dei geni. Questo meccanismo conduce ad una ricerca euristica che privilegia le zone dello spazio di ricerca dove è maggiormente possibile trovare soluzioni migliori, non trascurando altre zone a

più bassa probabilità di successo in cui saranno però impiegate un minor numero di risorse.

All'interno degli algoritmi evolutivi si è soliti distinguere bene tra gli algoritmi genetici e la programmazione genetica; spesso infatti le due cose sono confuse pur essendo completamente differenti.

Un **algoritmo genetico** (John Holland 1986) è costituito da una popolazione finita di individui di dimensione M, da una funzione di adattamento detta fitness, che fornisce una misura della capacità dell'individuo (solitamente codificato con una stringa di bit) di adattarsi all'ambiente, cioè costituisce una stima della bontà della soluzione e un'indicazione sugli individui più adatti a riprodursi, da una serie di operatori che trasformano l'attuale popolazione nella successiva e da un criterio di terminazione, che stabilisce quando l'algoritmo si deve formare.

La **programmazione genetica**, introdotta da John R. Koza [4.8], è una tecnica di risoluzione di problemi per mezzo della quale dei programmi per computer si evolvono combinandosi, riproducendosi o mutando per dar luogo ad altri programmi che meglio si adattano a risolvere un determinato problema. Essa rappresenta un'estensione degli algoritmi genetici dove un elemento della popolazione rappresenta un programma e non una stringa di bit. Un programma è rappresentato da un albero in cui i nodi interni sono funzioni e le foglie costituiscono i simboli terminali del programma. Lo spazio di ricerca è costituito da tutti i programmi composti dai terminali e dalle funzioni definite per uno specifico problema. La popolazione dei programmi evolve utilizzando una funzione di fitness, che definisce la bontà di un programma a risolvere un determinato problema. Si capisce bene che in questo ambito quello che a noi interessa sono gli algoritmi genetici.

Il limite di entrambe le tecniche è che in genere hanno bisogno di grandi popolazioni per ottenere una buona convergenza; questo aspetto, unito al fatto che calcolare la funzione di fitness di un certo individuo può essere

oneroso, porta ad uno sforzo computazionale notevole. Per ovviare a questi problemi si sono sviluppate tecniche di implementazione parallela.
Gli algoritmi genetici sono estremamente sensibili ai parametri che sono spesso di fondamentale importanza per la convergenza dell'algoritmo per cui la fase di tuning dei parametri è delicata e non va trascurata.
Una tecnica comune per rappresentare la conoscenza genetica consiste nel trasformare gli elementi in stringhe binarie.
Gli algoritmi genetici possono essere sviluppati sia per il data mining supervisionato che per quello non supervisionato; la struttura dell'algoritmo è comunque sempre la stessa:

1. Si inizializza, in modo casuale, una popolazione P di n elementi, comunemente detti cromosomi, come una soluzione potenziale.
2. Finchè non è soddisfatta una specifica condizione di conclusione:

    - Si utilizza una funzione di fitness per valutare ogni elemento della soluzione corrente. Se un elemento supera i criteri di fitness, rimane nella popolazione P.
    - La popolazione adesso contiene m elementi (ovviamente $m \leq n$). Si utilizzano gli operatori genetici per creare $(n-m)$ nuovi elementi. Si aggiungono i nuovi elementi alla popolazione.

4.8 Gli operatori genetici

Tutti gli algoritmi genetici si basano sulla teoria di Darwin sull'evoluzione della specie, tale teoria è infatti ricca di spunti ed è possibile ricavare interessanti analogie tra i componenti del processo naturale di evoluzione e i vari componenti di un problema di calcolo.

Con il termine **genotipo** si intende il codice genetico, ovvero l'insieme dei **geni**, che definiscono le caratteristiche di un particolare individuo nella specie considerata.

La manifestazione dei caratteri codificati in tale codice è detta **fenotipo**.

Ogni gene può assumere alleli differenti, a seconda del particolare allele assunto dal gene si hanno genotipi differenti.

I geni sono contenuti in **cromosomi**. Il cromosoma di un individuo contiene un certo numero di geni che, a seconda degli alleli assunti, ne determinano le caratteristiche.

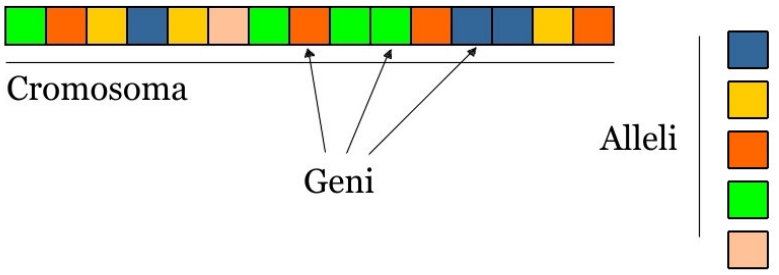

Figura 4.10: Cromosomi, geni e alleli.

Le caratteristiche di un fenotipo possono essere determinate non solo in funzione degli alleli presenti nel genotipo, ma anche in funzione di particolari combinazioni di alleli; in pratica può accadere che un allele determini una certa caratteristica invece di un'altra a seconda degli alleli assunti da altri geni. Tale fenomeno di interazione tra geni prende il nome di **epistasi**.

Un insieme di individui che condividono uno stesso genotipo costituisce una **popolazione**. Una popolazione di individui vive in un ambiente avente particolari caratteristiche a cui tali organismi si sono adattati o cercano di adattarsi per continuare a sopravvivere.

Tuttavia non tutti gli individui si adattano con la stessa efficacia o altrettanto rapidamente all'ambiente in cui vivono; per poter esprimere tale grado di idoneità si usa il concetto di fitness, ovvero un valore numerico indice di quanto un individuo è adatto all'ambiente circostante.

L'evoluzione è il processo che altera, di generazione in generazione, le caratteristiche genetiche degli organismi in modo da renderli sempre più adatti al mutevole ambiente in cui vivono.

In ogni popolazione si verificano delle mutazioni. Sono proprio le mutazioni che possono generare individui che meglio si adattano all'ambiente. Questi nuovi individui sopravviveranno quindi più a lungo generando più figli (*fase di riproduzione*).

Durante la fase riproduttiva gli individui si scambiano i geni ricombinandoli per formare il patrimonio genetico dei loro discendenti.

In realtà negli algoritmi genetici tale processo è leggermente modificato e potenziato: i figli preservano in parte i caratteri genetici dei genitori mantenendo identica a quella di uno o entrambi i genitori una parte dei cromosomi, mentre la restante parte è ottenuta rimescolando le catene di cromosomi dei genitori o mutandole geneticamente.

Questa fase è svolta tramite degli operatori genetici il cui principale è chiamato **crossover**. Tale operatore, presi due individui, taglia loro una catena cromosomica a partire da una posizione scelta a caso e successivamente scambia i cromosomi tra gli individui, generando così due nuovi individui contenenti i geni dei loro genitori (*fig. 4.12*).

Un altro operatore molto utilizzato è la **mutazione**; esso opera su un singolo individuo alterando a caso un gene del proprio cromosoma (*fig. 4.11*).

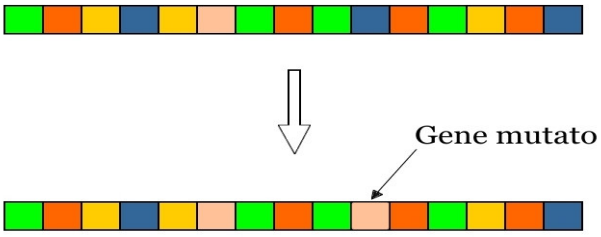

Figura 4.11: Mutazione.

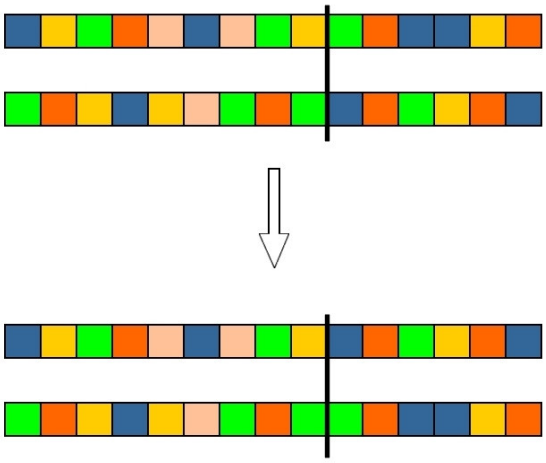

Figura 4.12: Crossover.

Gli operatori genetici mirano a facilitare la creazione di nuove "stirpi" completamente diverse dalle precedenti. Probabilmente, infatti, geni provenienti da individui aventi una buona fitness non solo saranno maggiormente diffusi nelle successive generazioni rispetto a quelli aventi una scarsa fitness ma, ricombinati, daranno luogo a nuovi individui generalmente migliori. A completamento del processo evolutivo vi è una legge di **selezione naturale** che si basa sul principio secondo cui, tra i nuovi individui generati, hanno maggiore probabilità di sopravvivere quelli che possiedono un fitness migliore.

In altre parole vengono eliminati tutti quegli individui la cui capacità di adattamento all'ambiente è giudicata insufficiente.

Computazionalmente l'implementazione di tale principio può essere realizzata anche in maniera assolutamente banale: viene permesso di sopravvivere solo a quegli individui che superano una soglia costituita da un certo valore di fitness determinato a priori e spesso dinamico, cioè che aumenta nel passaggio da una generazione alla successiva.

Tutti gli altri individui che non superano tale soglia prefissata vengono giudicati non idonei ed estinti dalla popolazione attuale. La Selezione naturale è l'operazione mediante la quale gli individui (in realtà il loro genotipo,

rappresentato dai cromosomi) sono selezionati per la riproduzione ed avviene solamente sulla base dei valori di fitness.

La selezione può essere svolta con più metodi che riflettono proprio diversi criteri di selezione adottati. E' chiaro che l'errata implementazione dell'operazione di selezione può compromettere in modo irrimediabile la riuscita dell'intero progetto. Uno fra i metodi più utilizzati è quello basato sulla selezione proporzionale alla fitness.

Va detto che è il metodo di selezione più comune, probabilmente perché è molto semplice: ad ogni individuo è assegnata una probabilità di essere selezionato. Tale probabilità è indissolubilmente legata al valore della sua fitness secondo la relazione

$$p_i = \frac{f_i}{\sum_j f_j}$$

Supponiamo di avere 4 individui con fitness

$$f_1=f_2=10 \qquad f_3=15 \qquad f_4=25$$

Allora si avrà una probabilità di selezione pari a

$$p_1=p_2=1/6 \qquad p_3=1/4 \qquad p_4=5/12$$

In base alle varie probabilità di selezione viene creata una funzione aleatoria con pesi opportuni in modo da assegnare maggiori possibilità di estrazione agli individui con fitness elevato.

In questo esempio è evidente che il 4° individuo ha probabilità molto maggiori di essere selezionato rispetto agli altri, quindi, per agevolare la sua selezione si dovrà creare un meccanismo che renda merito alla sua capacità di ottenere un valore di fitness relativamente al problema in esame.

Purtroppo questo metodo per quanto semplice ed intuitivo non è privo di controindicazioni; un problema frequente di questo approccio è la ***convergenza prematura***: se l'i-esimo individuo ha fitness molto maggiore della media della popolazione ma molto minore del massimo fitness possibile, tenderà a essere sempre selezionato e quindi a generare una popolazione "mediocre".
Altro inconveniente piuttosto frequente è la ***stagnazione***: dopo un certo numero di generazioni, tutti gli individui hanno una buona fitness e quindi tendono ad avere la stessa probabilità di essere selezionati.
Un'altra metodologia è la **rank-selection,** essa prevede che tutti gli individui siano ordinati per fitness decrescente.
Ad ordinamento terminato si impone una distribuzione di probabilità decrescente con la posizione occupata, indipendentemente dai valori di fitness.
Così facendo si aggirano i due principali svantaggi esaminati nel caso precedente:

- Non si ha convergenza prematura: poiché nessun individuo ha probabilità molto maggiore degli altri di essere selezionato
- Non c'è stagnazione: la distribuzione probabilità non varia.

Di contro come unico svantaggio, non irrilevante, si ha che l'implementazione è computazionalmente molto onerosa.
Va inoltre notato che come metodologia non è biologicamente plausibile. Un ulteriore metodo è la **tournament-selection**, che prevede l'aggregamento degli individui in gruppi; una volta selezionato un gruppo di individui, viene clonato solo il migliore. I vantaggi sono gli stessi della selezione per rango ma non c'è necessità di ordinamento, quindi in qualche modo la computazione è più snella.
Tuttavia la dimensione dei gruppi va scelta opportunamente poiché le prestazioni solo molto sensibili a tale parametro. Infine **l'elitist-selection**

consiste in questo: almeno una copia dell'individuo migliore viene replicata nella generazione successiva. Così facendo c'è il grosso vantaggio di non perdere soluzioni buone nei processi "casuali" di selezione.
Tuttavia è possibile bloccarsi in qualche massimo locale se i caratteri dell'individuo migliore diventano "dominanti".

4.9 *Algoritmi genetici e clusterizzazione*

L'apprendimento genetico è una potente tecnica di clusterizzazione non supervisionata; nel DM possiamo paragonare gli elementi alle osservazioni definite da variabili e valori.L'apprendimento genetico non supervisionato può essere utilizzato per sviluppare un insieme di cluster per dati numerici in uno spazio n-dimensionale.
Supponiamo di avere a disposizione P osservazioni all'interno dello spazio a n dimensioni; per formulare il problema, all'algoritmo viene fornito il numero di cluster m. Il modello genera k possibili soluzioni; ogni soluzione contiene m punti n-dimensionali, dove ogni punto è il miglior elemento rappresentativo per uno degli m cluster.

Osservazione	x	y
1	1.0	1.5
2	1.0	4.5
3	2.0	1.5
4	2.0	3.5
5	3.0	2.5
6	5.0	6.0

Figura 4.13: Valori di input per l'algoritmo genetico.

La figura 4.14 illustra questo concetto nel caso $m = 2$; S_i rappresenta una delle k possibili soluzioni e contiene due elementi E_{i1} e E_{i2}.

	S1	S2	S3
Elementi della soluzione (popolazione iniziale)	(1.0, 1.0) (5.0, 5.0)	(3.0, 2.0) (3.0, 5.0)	(4.0, 3.0) (5.0, 1.0)
Punteggio di fitness	11.31	9.78	15.55
Elementi della soluzione (seconda generazione)	(5.0, 1.0) (5.0, 5.0)	(3.0, 2.0) (3.0, 5.0)	(4.0, 3.0) (1.0, 1.0)
Punteggio di fitness	17.96	9.78	11.34
Elementi della soluzione (terza generazione)	(5.0, 5.0) (1.0, 5.0)	(3.0, 2.0) (3.0, 5.0)	(4.0, 3.0) (1.0, 1.0)
Punteggio di fitness	13.64	9.78	11.34

Figura 4.14: Evoluzione della popolazione per la clusterizzazione non supervisionata

Viene effettuata un'operazione di crossover spostando gli elementi (punti n-dimensionali) dalla soluzione S_i alla soluzione S_j; ci sono vari modi per implementare le operazioni di mutazione, un modo è quello di scambiare uno o più coordinate degli elementi all'interno di S_i.

Una funzione di fitness applicabile alla soluzione S_j è la distanza media euclidea fra le P osservazioni nello spazio a n dimensioni e i corrispondenti elemanti più vicini all'interno di S_j.

In altre parole, per calcolare il valore di fitness per la soluzione S_j, prendiamo ogni osservazione I in P e calcoliamo la distanza euclidea da I a ciascuno degli m elementi di S_j. La distanza più vicina viene salvata e aggiunta alla somma totale dei valori più vicini.

La media della somma delle distanze è il punteggio di fitness per S_j. Chiaramente, i valori più bassi corrispondono a punteggi di fitness migliori. Una volta che l'apprendimento genetico è terminato, viene scelta come soluzione finale la migliore delle k possibili soluzioni. Ogni osservazione nello spazio a n dimensioni viene poi assegnata al cluster associato all'elemento più vicino nella soluzione finale.

Vediamo ora come la tecnica possa essere applicata alle sei osservazioni in tabella (figura 4.13). Supponiamo che i dati contengano due cluster e indichiamo all'algoritmo di iniziare con un insieme di soluzioni formate da tre possibili soluzioni ($K = 3$).

Per $m = 2, P = 6, K = 3$, l'algoritmo genera l'insieme iniziale delle soluzioni in figura 4.14. Un elemento dello spazio delle soluzioni contiene un singolo dato rappresentativo per ogni cluster, per esempio, i punti della soluzione S_1 sono (1.0,1.0) e (5.0,5.0).

Per calcolare il punteggio di fitness 11.31 per la soluzione S_1, viene sommata la distanza euclidea tra ogni ossevazione della tabella 4.13 e i punti più vicini in S_1. La tabella 4.14 mostra S_2 come la migliore soluzione della prima generazione.

La seconda generazione è ottenuta con un'operazione di crossover fra le soluzioni S_1 e S_3, scambiando i posti dell'elemento (1.0,1.0) di S_1 con l'elemento (5.0,1.0) di S_3; si può notare che il risultato del crossover migliora (riduce) il punteggio di fitness di S_3 ma peggiora quello di S_1.

La terza generazione è ottenuta tramite la mutazione di S_1; tale operatore scambia la coordinata y del primo elemento di S_1 con la coordinata x del secondo elemento.

Il risultato della mutazione è un miglioramento del punteggio di fitness per S_1; la mutazione e il crossover continuano fin quando non sarà soddisfatta la condizione di conclusione.

Questa condizione può essere un numero massimo di iterazioni dell'algoritmo o un punteggio di fitness minimo per una o più soluzioni.

Se la terza generazione è considerata come ultima generazione, allora la soluzione finale sarà S_2. Otteniamo così un cluster contenente le osservazioni 1,3 e 5 ed un cluster contenente le osservazioni 2 e 6.

Poiché l'osservazione 4 è equidistante da entrambi gli elementi in S_2, può essere inserita in entrambi i cluster.

4.10 Considerazioni finali

Nonostante pochi pacchetti commerciali di data mining contengano uno strumento di apprendimento genetico, gli algoritmi genetici continuano a guadagnare consensi.

Vanno fatte comunque una serie di considerazioni sull'impiego delle tecniche di risoluzione dei problemi che si basano sull'apprendimento genetico.

Gli algoritmi genetici sono stati ideati per trovare soluzioni globalmente ottimizzate tuttavia non ci sono garanzie che una data soluzione sia il risultato di un'ottimizzazione locale, anziché globale.

La funzione di fitness determina la complessità di calcolo di un algoritmo genetico, essa infatti può essere molto costosa in termini di tempi di elaborazione e quindi va scelta in maniera oculata anche perché gli algoritmi genetici spiegano il loro risultato nell'ambito della funzione di fitness.

Un ulteriore inconveniente è che la trasformazione dei dati in un formato adatto agli algoritmi genetici può risultare un'impresa ardua.

Nei GAs il meccanismo evolutivo è completamente generale per cui estendere l'applicazione ad un nuovo problema comporta come unico lavoro la definizione di una nuova fitness function.

È importante simulare una popolazione sufficientemente grande ($\sim 10^2$ - 10^3), e permettere all'evoluzione di procedere per abbastanza generazioni ($\sim$ 50 - 500), comunque il successo di un algoritmo genetico dipende principalmente da come sono realizzati gli operatori di riproduzione e da quanto è ben definita la fitness function.

Un buon operatore di riproduzione produce individui nuovi con probabilità ragionevole di essere dotati di un nuovo valore di fitness significativamente più alto di quello dei suoi genitori.

Inoltre l'andamento della fitness function deve essere quello di una funzione monotona crescente, in modo da avvicinarsi sempre più alla soluzione ottima

del problema in esame. La combinazione di queste due proprietà, quando presenti, costituisce la garanzia per la convergenza verso soluzioni soddisfacenti.

Molto del lavoro teorico nel campo dei GAs è costituito dal trovare operatori di riproduzione migliori e più adatti ai vari problemi.

I risultati ottenibili con i GAs sono limitati dal "significato" dei parametri iniziali, dalle rappresentazioni sintetiche degli individui e di conseguenza dalla valutazione poco accurata che essi ricevono dalla fitness function. Nella Programmazione Genetica invece la rappresentazione di ogni individuo è un programma vero e proprio, spesso realizzato in un vero linguaggio di programmazione, e va quindi eseguito per determinare l'appropriatezza dell'individuo. In questo caso la fitness function contiene un interprete il quale deve decodificare le istruzioni di ciascuno individuo e determinare come quell'individuo risolve il problema in esame.

I risultati sono ovviamente molto più accurati ma il prezzo da pagare è una complessità molto più elevata.

Oltre a essere utilizzate nell'apprendimento supervisionato e nella clusterizzazione non supervisionata, le tecniche genetiche possono essere utilizzate con altre tecniche di apprendimento. Per esempio, gli algoritmi genetici possono essere applicati per addesrtare le reti neurali feed-forward. Ogni cromosoma o elemento di una popolazione rappresenta una possibilità per i pesi delle connessioni presenti nella rete.

L'apprendimento genetico può essere addirittura impiegato per semplificare la scelta delle variabili di input migliori per l'apprendimento supervisionato.

CAPITOLO 5

Applicazioni avanzate di DM

5.1 I confini del data mining

Nei precedenti capitoli si sono illustrate le principali tecniche di DM, che trovano un largo ambito applicativo; ovviamente ciascuna di queste tecniche può essere più o meno adatta all'analisi di un particolare insieme di dati. Gli strumenti di DM, comunque, non si limitano a quelli illustrati ma sono molteplici. Sono, ad esempio, molto impiegate le reti neurali che lavorano bene con dataset contenenti grandi quantità di disturbo infatti le funzioni di valutazione delle reti effettuano in modo naturale una regolazione della variabilità nei dati causata da outlier ed errori casuali. Esse, inoltre, possono essere applicate sia per l'apprendimento supervisionato che non supervisionato. Il più grande difetto delle reti sta nell'incapacità di spiegare il comportamento delle reti neurali, il secondo difetto è che non c'è garanzia di convergenza a una soluzione ottimale. Il DM sconfina addirittura nella fuzzy logic, tecnica euristica di successo crescente, che viene utilizzata nei sistemi rule-based per la gestione dell'incertezza. Risulta difficile, quindi, individuare i limiti del data mining; è più interessante analizzare alcune delle sue applicazioni più importanti.

5.2 Il Web mining

Le società che si occupano di e-commerce per la maggior parte dei loro affari hanno in comune un obiettivo: ottimizzare la struttura dei loro siti Web in modo da massimizzare le vendite. Per esempio, un sito Web che vende prodotti trae il massimo beneficio se presenta contemporaneamente quei prodotti che i clienti acquistano insieme con maggiore probabilità. D'altra parte, un sito Web che richiede ai potenziali clienti di spendere una quantità ingiustificata di tempo per scegliere i prodotti è molto probabile che fallirà. Il successo di un sito Web dipende in ultima analisi da come viene visto dalla

comunità degli utenti. Tre fattori principali aiutano a determinare come gli utenti percepiscono un sito: i prodotti o i servizi offerti dal sito, il design della singola pagina, il design di tutto il sito. Il design del sito è ciò che più direttamente si riferisce alla struttura di un sito Web. Il data mining può essere utilizzato per :

- Valutare un sito Web per determinare se le intenzioni del progettista del sito corrispondono i desideri dell'utente (Spiliopoulou, 2000)
- Personalizzare i prodotti e le pagine presentate a un particolare utente o a un gruppo di utenti (Mobasher, Cooley e Srivastava, 2000)
- Adattare automaticamente la struttura di un sito Web per soddisfare meglio le mutevoli esigenze degli utenti (Perkowitz, 2000)

Il data mining sul Web presenta un nuovo insieme di sfide rispetto alle altre applicazioni di DM. Tali problematiche vanno esaminate attraverso la prospettiva del KDD.

La prima fase del processo di KDD richiede **l'identificazione degli obiettivi**, cioè la determinazione di uno o più obiettivi. Sicuramente nell'ambito Web bisognerà mirare a:

- Diminuire il numero medio di pagine visitate da un cliente prima di un'operazione di acquisto
- Aumentare il numero medio di pagine visitate da un utente in ogni singola sessione
- Aumentare l'efficienza del server
- Aumentare i tassi medi di mantenimento dei visitatori
- Diminuire il numero totale di resi dei prodotti acquistati
- Personalizzare le pagine Web per i clienti
- Determinare quei prodotti che tendono ad essere acquistati o visti insieme

Identificati gli obiettivi, più di 80% del tempo speso in un progetto di data mining del Web è assorbito dalle fasi 2, 3 e 4 (fasi di preparazione dei dati) del processo KDD.

5.3 Preparazione dei dati

I dati disponibili come risultato di una o più sessioni di un navigatore Web sono immagazzinati in **log** file. Un tipico log file contiene informazioni che descrivono le sequenze clickstream seguite da un utente; i log file molto spesso offrono informazioni nella forma nota come **common log file format**. I campi associati a tale formato sono:

1. Indirizzo host
2. Data/ora
3. Richiesta
4. Stato
5. Byte
6. Pagina di riferimento
7. Tipo di browser

Noi siamo interessati ai campi del log file che ci permettono di identificare la sequenza di clickstream seguita da un utente mentre naviga in un particolare sito. Il lavoro di preparazione dei dati consiste proprio nell'estrazione dei dati rilevanti dai log e nel creare i file adatti al processo di data mining. La figura 5.1 mostra che il file creato dal processo di preparazione dei dati prende il nome di **file delle sessioni**; esso può contenere fino a migliaia di record, ciascuno dei quali rappresenta un'osservazione di una sessione di un utente.

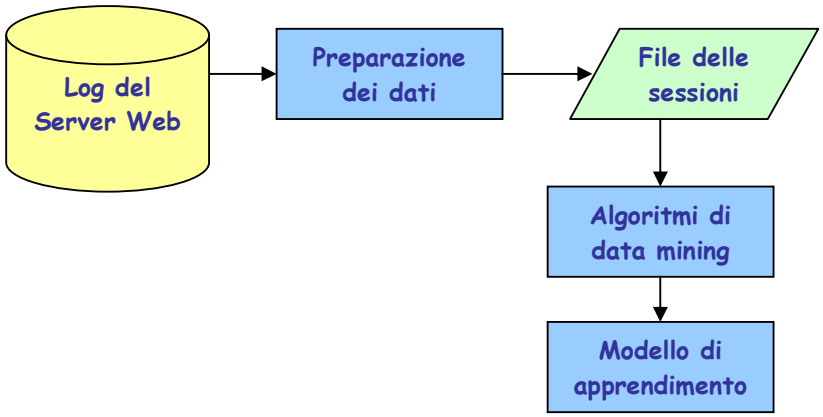

Figura 5. 1 : Un generico modello di utilizzo del Web

Una **sessione utente** è semplicemente un insieme di pagine viste che sono richieste da un singolo utente da un singolo Web server; ciascuna di queste pagine è composta da uno o più file di pagina ciascuno dei quali forma una singola finestra in un browser. Ogni pagina vista è contrassegnata da un identificatore, detto **URI**, per finalità di riconoscimento durante il processo di data mining. Creare un file delle sessioni è un compito difficile per vari motivi. Prima di tutto, per creare le singole sessioni di server dobbiamo essere in grado di identificare ciascun utente tra i tanti elencati in un log file. Gli indirizzi host sono di poco aiuto perché più utenti possono accedere a un sito dello stesso host. Quando l'indirizzo host è combinato con la pagina di riferimento, possiamo distinguere con maggiore facilità un utente da un altro. Tuttavia, la differenziazione è migliore se i siti possono utilizzare i cookie cioè i file (registrati nei computer degli utenti) che contengono tutte le informazioni della sessione utente ma ovviamente non si può fare affidamento sulle informazioni potenziali fornite dai cookie perché spesso si inibisce il permesso della loro registrazione. I problemi relativi alla creazione dei file delle sessioni non finiscono con la difficoltà di distinguere gli utenti: un secondo problema è che la richiesta di una singola pagina spesso genera più tipi di log file da vari tipi di server. Noi ovviamente non siamo interessati ai record creati dai server delle applicazioni e quindi abbiamo bisogno di una tecnica per identificare i

log indesiderati in modo che non facciano parte del file delle sessioni. Infine, possono essere necessarie una o più trasformazioni dei dati, per esempio l'aggiunta di nuove variabili ai record della sessione. Se sono disponibili ripetuti record su clienti, possono risultare utili informazioni relative alla quantità di soldi spesi precedentemente, all'ammontare degli acquisti, alla quantità di tempo speso nella transazione più recente.

5.4 Data mining sul Web: interpretare e valutare i risultati

Una volta che è stato creato il file delle sessioni, i dati vengono sottoposti alla vera e propria analisi di data mining. Gli algoritmi applicati ai dati possono essere tecniche tradizionali come la ricerca di regole associative o i metodi di clusterizzazione tuttavia, sulla base degli obiettivi di progetto, la scelta migliore potrebbe essere costituita dagli algoritmi appositamente ideati per determinare le sequenze delle pagine viste. Le osservazioni di un file delle sessioni rappresentano il comportamento sulle pagine di un singolo utente durante la sessione. Consideriamo ora un esempio che mostra come interpretare i risultati di un'ipotetica sessione di mining sul Web utilizzando la **tecnica delle regole associative**. Prendiamo in esame quattro osservazioni ($I_1 - I_4$) della sessione ipotetica, dove ogni P_i rappresenta una pagina vista:

- I_1: $P_5 \rightarrow P_4 \rightarrow P_{10} \rightarrow P_3 \rightarrow P_{15} \rightarrow P_2 \rightarrow P_1$
- I_2: $P_2 \rightarrow P_4 \rightarrow P_{10} \rightarrow P_8 \rightarrow P_{15} \rightarrow P_4 \rightarrow P_{15} \rightarrow P_1$
- I_3: $P_4 \rightarrow P_3 \rightarrow P_7 \rightarrow P_{11} \rightarrow P_{14} \rightarrow P_8 \rightarrow P_2 \rightarrow P_{10}$
- I_4: $P_1 \rightarrow P_3 \rightarrow P_{10} \rightarrow P_{11} \rightarrow P_{15} \rightarrow P_4 \rightarrow P_{15} \rightarrow P_9$

Supponiamo che il seguente sia l'output del generatore di regole sulla base delle quattro osservazioni precedenti:

$$\textbf{IF} \rightarrow P_4 \,\&\, P_{10} \;\; \textbf{THEN} \;\; P_{15}\{3/4\}$$

La regola ci dice che ci sono tre osservazioni dove P_4, P_{10}, P_{15} appaiono insieme in un singolo record di sessione; siamo sicuri al 75% che ogni volta che un utente fa riferimento a P_4 e P_{10}, fa riferimento anche a P_{15}. Poiché con esistono collegamenti diretti diretti tra le tre pagine viste, tale risultato potrebbe suggerire la modifica della struttura del sito tramite l'inserimento di uno o più link tra le suddette pagine. Come seconda possibilità, se la precondizione della regola combacia con l'attività dell'utente attuale, P_{15} può essere aggiunta alla lista delle pagine raccomandate che vengono automaticamente presentate all'utente. In questo modo, le pagine viste dall'utente possono essere personalizzate sulla base dei suoi interessi specifici. Una soluzione diversa potrebbe essere quella di impiegare la **clusterizzazione non supervisionata** per creare i cluster delle osservazioni simili del file delle sessioni. Possono essere utilizzati vari algoritmi di clusterizzazione e misure di similarità. Una strategia plausibile consiste nell'utilizzare la **clusterizzazione agglomerativa,** dove la similarità tra le osservazioni è calcolata come rapporto fra il numero totale di pagine viste, che ogni coppia di osservazioni ha in comune, e il numero totale di pagine viste contenute nelle osservazioni. Per illustrare questo metodo consideriamo le osservazioni I_1 e I_2, esse contengono cinque delle otto pagine viste per cui la loro similarità è pari a 0.625. L'uso di queste informazioni può essere fondamentale infatti si possono utilizzare i cluster formati da sessioni simili per personalizzare le pagine viste dagli utenti. Inoltre, per scoprire pattern nei dati Web, è utile ottenere statistiche riassuntive sulle attività svolte nel sito Web. Sono utilizzabili diversi pacchetti commerciali scaricabili dalla rete, che analizzano i log file con vari indici statistici; l'output consiste spesso nell'aggregare i log file raccolti che vengolo mostrati in strutture di tipo grafico. Si capisce bene che il Web mining è fondamentale in quanto solo a valle di esso si può:

- Implementare una strategia basata sui profili degli utenti per personalizzare le pagine Web visitate dagli utenti
- Adattare la struttura di un sito per riflettere meglio i percorsi seguiti dagli utenti tipici
- Creare promozioni pubblicitarie per i clienti di un sito Web
- Modificare il contenuto di un sito Web raggruppando i probabili prodotti che verranno acquistati insieme, eliminando i prodotti di scarso interesse e aumentando le offerte dei prodotti con forte richiesta

5.5 Data mining per valutare i siti Web

La valutazione dei siti Web consiste nel determinare se l'utilizzo effettivo di un sito corrisponde alle intenzioni del suo progettista. Se i percorsi seguiti dalla maggior parte dei visitatori non coincidono con quelli attesi, è probabile che sia difficile navigare all'interno del sito in questione. In tal caso, il progettista del sito deve valutare un cambiamento dell'architettura del sito per soddisfare meglio le esigenze degli utenti. Il data mining può essere utile per valutare i siti Web tramite la determinazione dei pattern frequenti e dei percorsi seguiti dalla popolazione degli utenti. Tuttavia, diversamente da molte applicazioni di Web mining, la valutazione dei siti non si concentra unicamente sulla clusterizzazione delle pagine visitate ma anche sull'ordine sequenziale delle pagine viste, cosa che consente al progettista di determinare se la struttura del percorso sia davvero la rappresentazione migliore delle abitudini di navigazione dei suoi utenti. Per **l'identificazione delle sequenze** viene utilizzata una classe speciale di algoritmi di data mining: **sequence miner**. Tali algoritmi sono in grado di scoprire le pagine visitate frequentemente che si presentano nello stesso ordine. Spliliopoulou (2000) descrive il WUM [5.1], un log miner generalizzato che può effettuare analisi di sequenze ma che permette anche ai creatori di siti di scrivere query in formato SQL, al fine di determinare i pattern

esistenti tra le singole pagine. WUM (WebLog Utilization Miner) è il risultato di un progetto sviluppato dall'università Humbdoldt di Berlino; si tratta di un tool integrato con strumenti di Log-preparation, querying e visualization, i cui punti di forza sono proprio la capacità di identificare le pagine più frequentemente visitate dai navigatori e la presenza di strumenti di postprocessing inoltre è dotata di un'interfaccia di semplice utilizzo.

5.6 Data mining per personalizzare i siti Web

Scopo della personalizzazione di un sito Web è offrire ad un utente ciò che gli interessa, senza richiederglielo in modo esplicito. La personalizzazione può essere implementata manualmente o automaticamente con l'aiuto dei metodi di data mining. Le tecniche manuali richiedono all'utente la registrazione a un sito Web, di rispondere a qualche domanda e di riempire alcune caselle; tali tecniche non sono in grado di offrire una raffigurazione reale delle azioni attese degli utenti. Utilizzando il data mining per rendere automatica la personalizzazione, la soggettività delle risposte degli utenti è sostituita dal comportamento reale dell'utente. La figura 5.2 mostra le fasi generali della personalizzazione automatica. Il processo richiede la creazione del profilo di utilizzo a partire dai dati immagazzinati delle sessioni. La parte più difficile della procedura consiste nel passaggio dai cluster formati dal processo di data mining ai profili di utilizzo.

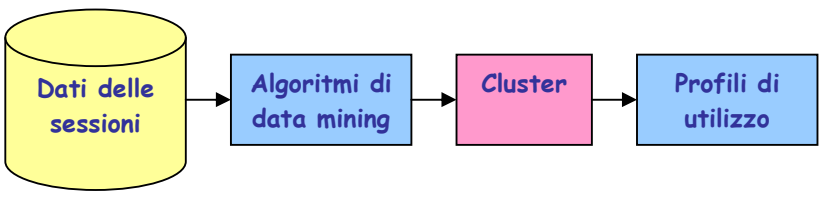

Figura 5. 2 : Procedura di creazione dei profili di utilizzo

Mobasher (2000) descrive la tecnica utilizzata dal WebPersonalizer per creare i profili di utilizzo.

Il WebPersonalizer impiega due tecniche di profiling; una tecnica impiega le regole associative per creare direttamente i profili di utilizzo, un secondo metodo crea i profili tramite la generalizzazione dei cluster.

Per ciascun cluster, solo quelle pagine che soddisfano un punteggio minimo entrano a far parte del profilo. Il punteggio è un rapporto che viene calcolato dividendo il numero totale di volte che una pagina vista appare in un cluster durante un'intera sessione per il numero totale di sessioni in un cluster. Le pagine viste che incontrano il criterio minimo entrano a far parte del profilo di utilizzo che rappresenta il cluster.

Inoltre, a ciascuna pagina vista è assegnato un peso che riflette la relativa frequenza di apparizione in tutte le sessioni. Questo peso viene utilizzato per calcolare un punteggio di raccomandazione della pagina per i singoli utenti. Per ciascun utente, solo quelle pagine viste che hanno tale punteggio al di sopra di una soglia minima sono candidate per la presentazione.

Figura 5. 3: Generazione dei link consigliati a partire dai dei profili di utilizzo

Infine, la figura 9.3 mostra che il motore che determina le raccomandazioni (Recommendation engine) per il WebPersonalizer usa come input l'insieme dei profili di utilizzo creati e combina l'attività corrente di navigazione degli utenti con i profili immagazzinati. Per le applicazioni di e-commerce, la maggior parte delle pagine consigliate saranno link a prodotti e pubblicità.

5.7 Data mining per adattare i siti Web

Le scelte iniziali del web master relativamente all'organizzazione di un sito Web sono fortemente influenzate dalle loro convinzioni su come gli utenti accederanno al nuovo sito. Anche se una struttura iniziale è adeguata per molti utenti, le rispettive necessità mutano nel corso del tempo. Quando i link iniziali non vengono più utilizzati, emergono nuovi profili di utilizzo perciò affinchè un sito Web rimanga competitivo, è necessario che la sua struttura cambi continuamente nel tempo. Il data mining consente di rendere automatico il procedimento di trasformazione. I siti Web, che sono in grado di migliorare in modo semiautomatico le loro strutture interne e i relativi metodi di presentazione grazie all'apprendimento dei pattern di accesso dei visitatori, sono noti come **siti Web adattativi**. La struttura interna di un sito ha la sua parte più importante nell'insieme delle pagine indice, ciascuna di esse non è altro che una pagina Web che contiene link con un insieme di pagine che trattano argomenti particolari. Il problema del miglioramento automatico della struttura interna di un sito si riconduce quindi alla sintesi delle pagine indice. Perkowitz e Etzioni (2000) hanno dato la seguente definizione formale del **problema della sintesi delle pagine indice** [5.3]:

> *Dato un sito web ed un log di accessi dei visitatori, creare nuove pagineindice che contengono insiemi di link con pagine che sono in relazione, ma correntemente non collegate.*

IndexFinder è un sistema automatico di sintesi di pagine che utilizza un algoritmo di clusterizzazione per candidare le pagine indice per un sito Web esistente; per ogni pagina candidata viene creata anche una descrizione rule-based. Il web master deve poi decidere se accettare o meno la pagina candidata. Se viene accettata una nuova pagina indice, IndexFinder crea una forma di pagina finale e aggiunge automaticamente la pagina al sito.

5.8 Text mining

Il text mining consiste nell'estrazione di pattern utili da testi liberi. Poiché i dati testuali non sono strutturati, un'analisi iniziale appare abbastanza difficile tuttavia il data mining testuale è possibile perché si cerca unicamente di classificare dati testuali, senza cercare di capire completamente il contenuto. Un problema comune che può essere risolto con il text mining consiste nella determinazione dell'appartenenza o meno di un determinato documento a uno specifico argomento. Questo problema può essere considerato una classificazione binaria, per esempio potremo volere esaminare quotidianamente gli articoli, contenuti in vari quotidiani, che riguardano la borsa. Un algoritmo di text mining che può essere utilizzato per la risoluzione di questo problema può essere così strutturato:

- Addestrare: Creare un dizionario di vocaboli che rappresentano le parole, tratte dagli articoli, che hanno legami con argomenti di borsa.
- Filtrare: Rimuovere le parole comuni che, notoriamente, non sono utili per differenziare i vari tipi di articoli.
- Classificare: Controllare ogni nuovo documento da classificare sulla base della presenza e della frequenza delle variabili scelte. Se un determinato documento contiene un numero minimo predeterminato di riferimenti alle variabili scelte, si classifica il documento come un elemento della classe degli articoli che riguardano il mercato azionario.

5.9 *Applicazioni del text mining: analisi della letteratura biomedica*

Tecniche di data mining sono già utilizzate, in ambito biologico, per la previsione della struttura di proteine, ricerca di omologie, analisi di sequenze genomiche, individuazione e mappatura di geni. Vogliamo ora invece approfondire l'apporto che tecniche di text mining possono dare all'analisi della letteratura biomedica.

Esistono principalmente due tipi di approccio all'analisi dei testi: l'**information extraction**, che ha come obiettivo quello di produrre una rappresentazione strutturata dell'informazione testuale, e il **text mining**, che ha come obiettivo l'individuazione automatica di pattern di parole che consentano la creazione di gruppi tematici.

Il primo richiede che entità e relazioni da ricercare nei testi siano predefinite (ad esempio le entità possono essere proteine e farmaci e le relazioni attivazione, inibizione, …), il secondo non richiede alcuna definizione a priori e consente di individuare nuovi concetti e nuove relazioni.

Il primo comprende tecniche che utilizzano sia l'analisi sintattica che l'analisi semantica, il secondo opera solamente a livello sintattico.

Questi due approcci non sono completamente separati: l'information extraction genera database che possono essere analizzati utilizzando le tecniche di data mining, mentre il text mining può trarre vantaggio dalla presenza di informazione specifica di dominio estratta con tecniche di I.E., come è mostrato nell'esempio che segue.

La fonte da cui sono stati estratti i documenti è Pubmed [5.5], da cui sono state selezionate tutte le pubblicazioni riguardanti il ciclo cellulare (cell cycle OR cell proliferation OR cell death OR oncogenes OR tumor suppressor OR apoptosis OR PARP OR caspase OR CDK OR PCNA OR Fas ligand OR cytochrome C), per un totale di circa 400 mila testi.

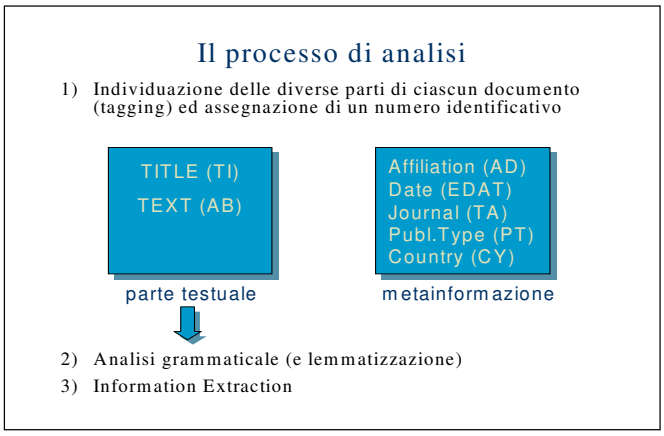

Figura 5. 4: Generazione dei link consigliati a partire dai profili di utilizzo

La preparazione dei documenti, su cui poi agisce uno strumento di DM, è stata effettuata attraverso tre fasi:

1. Separazione dell'informazione testuale dalla meta-informazione (organismo di appartenenza, data e rivista di pubblicazione, tipo di pubblicazione, paese, ecc…).
2. Analisi grammaticale della parte testuale.
3. Applicazione di tecniche di information extraction.

L'**analisi grammaticale** consente di individuare la funzione di ciascun termine e di ricondurlo alla propria radice, o forma canonica, o lemma.

Poiché, all'interno di un discorso, i sostantivi sono generalmente i termini con maggiore contenuto semantico, dalla lista dei termini lemmatizzati che si ottiene, per ogni documento, come risultato di questa fase, si selezionano tutti i sostantivi. Questi vengono a costituire l'insieme di parole chiave che caratterizza e descrive ciascun documento.

È sulla base di queste parole chiave che verrà giudicato il grado di somiglianza tra i documenti.

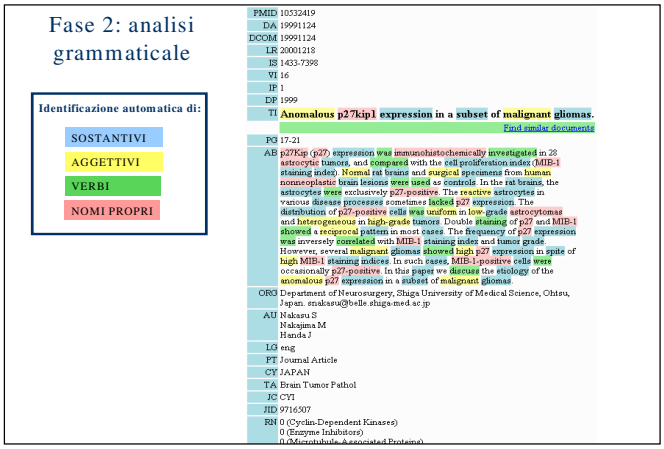

Figura 5. 5: Analisi grammaticale

Tutte le altre parti del discorso (aggettivi, verbi e nomi propri) vengono mantenute come informazioni aggiuntive, così come la meta-informazione che era stata estratta durante la prima fase.

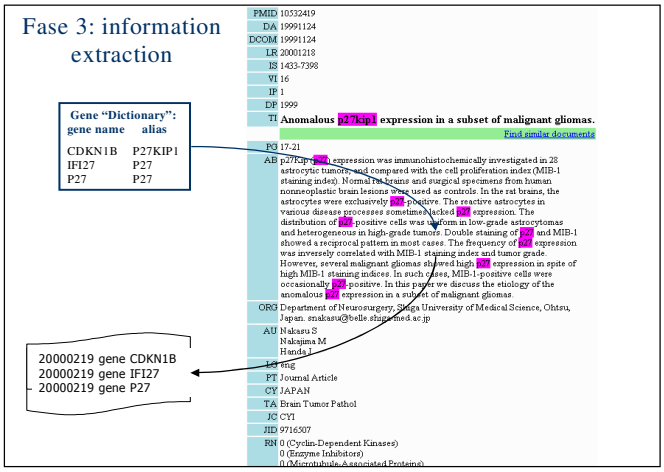

Figura 5. 6: Information extraction

Poiché nell'ambito della ricerca biologica è molto importante poter identificare i nomi dei geni, e poiché l'analisi grammaticale li classifica, genericamente, come nomi propri, è necessaria un'ulteriore fase di analisi del testo per estrarre questo tipo d'informazione (figura 5.3).

Questa fase, detta di **information extraction**, ha l'obiettivo di estrarre termini specifici (e predefiniti) dal testo ed è stata effettuata utilizzando un dizionario contenente nomi ufficiali di geni ed alias. Occorre precisare che non si tratta di I.E. vera e propria, in quanto, al momento, nessuna analisi semantica opera sul testo, per l'estrazione dei termini.

Il "dizionario", derivato dalla banca dati Locus Link, in quanto risulta essere la fonte più stabile e completa di informazioni sui geni finora individuati (Locus Link deriva da UniGene ed associa a ciascun gene un numero identificativo univoco.

Il processo si compone di tre parti: filtro, indicizzazione e generazione della meta-informazione.

Il **filtro** consente di estrarre dai record di Locus Link solo le informazioni di interesse e cioè simboli ufficiali e alias, selezionare i termini di almeno tre caratteri e creare le coppie GENE/ALIAS per la successiva indicizzazione dei documenti.

L'**indicizzazione** viene fatta ricercando ciascun termine che compare nella colonna "alias" nel testo dei documenti e registrando il termine ufficiale corrispondente (che compare nella colonna "gene") più il numero identificativo del documento che contiene l'alias.

Questo elenco viene successivamente pulito dalle ridondanze e viene ordinato per identificativo del documento. Per ogni documento si ottiene così l'elenco dei nomi ufficiali di geni che vi compaiono, sia come tali, sia come alias. Questo consente di generare la **meta-informazione** nel formato

precedentemente visto e di integrarla con le altre informazioni estratte dal documento originale.

Al termine di questa fase, per ogni documento si è ottenuta della meta-informazione aggiuntiva, data dalla lista dei nomi di geni (solo nomi ufficiali) che erano presenti nel testo (o come tali, o sotto forma di alias).

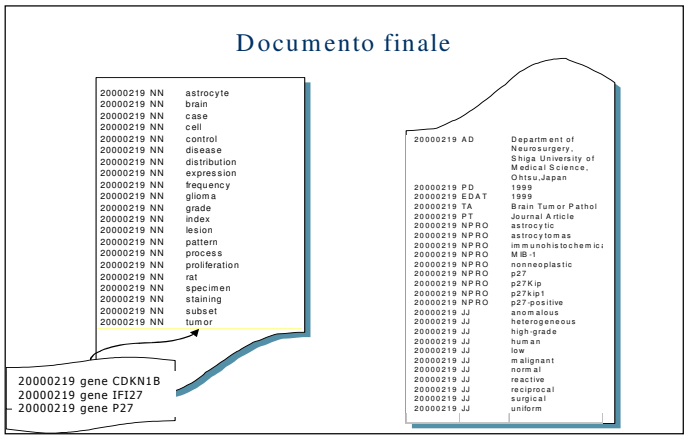

Figura 5. 6: Output della fase di Information extraction

Il documento finale, al termine della fase di preparazione, è formato da un elenco composto da tre elementi: l'identificativo del documento, il nome della variabile (nell'esempio NN indica sostantivo, cioè parola chiave, JJ aggettivo, VV verbo, NPRO nome proprio, AD "affiliation", PD data di pubblicazione, … GENE nome di gene) e il valore, o contenuto, della variabile, anche detto attributo del documento.

Questo formato consente di rappresentare l'informazione in **una matrice binaria che contiene, sulle righe ciascun documento, sulle colonne ciascun sostantivo, e all'interno i valori 1 o 0 ad indicare la presenza di quella determinata parola chiave in quel determinato documento oppure la sua assenza** (analogamente sono rappresentati gli altri attributi).

La fase di **data mining** si concretizza nell'applicazione di un algoritmo di clustering. Poiché l'informazione disponibile è tutta di tipo qualitativo, si è scelto un algoritmo partitivo basato sull'analisi relazionale.

I gruppi ottenuti vengono presentati in ordine di dimensione e sono descritti dalle parole chiave che maggiormente li caratterizzano. La "cluster map" fornisce una visione d'insieme dei *cluster* e dei loro legami (nella partizione finale dei documenti possono rimanere dei residui di somiglianza tra documenti che sono stati assegnati a gruppi diversi, questi residui sono rappresentati da linee). La mappa mette in evidenza un gruppo di *cluster* (in alto a sinistra, cerchiato di rosso) che sono collegati tra di loro, ma separati da tutti gli altri *cluster*. L'analisi del loro contenuto mostra la condivisione di alcune parole chiave (woman, age, risk,..) anche se le pubblicazioni trattano argomenti diversi inoltre emerge anche l'assenza in questi documenti del nome RET.

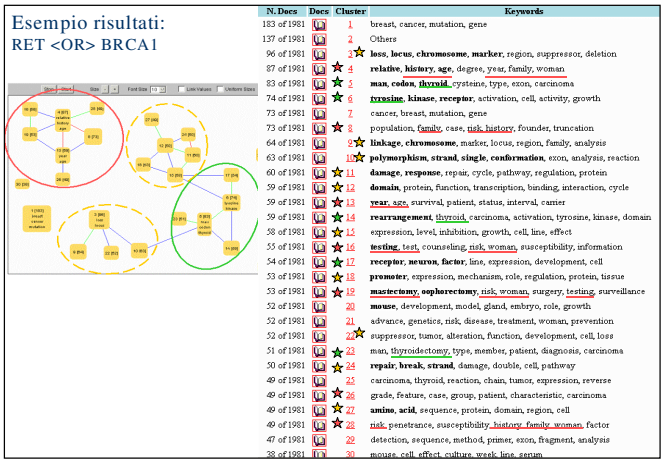

Figura 5. 7: Risultato del clustering

La meta-informazione può essere rappresentata in modo più diretto tramite un istogramma, dove ogni barra rappresenta un *cluster* (tranne la prima che indica la distribuzione dei geni su tutti i documenti selezionati) e l'estensione di

ciascun colore indica la percentuale di documenti che, all'interno del gruppo, contiene un particolare nome di gene (indicato dalla legenda).

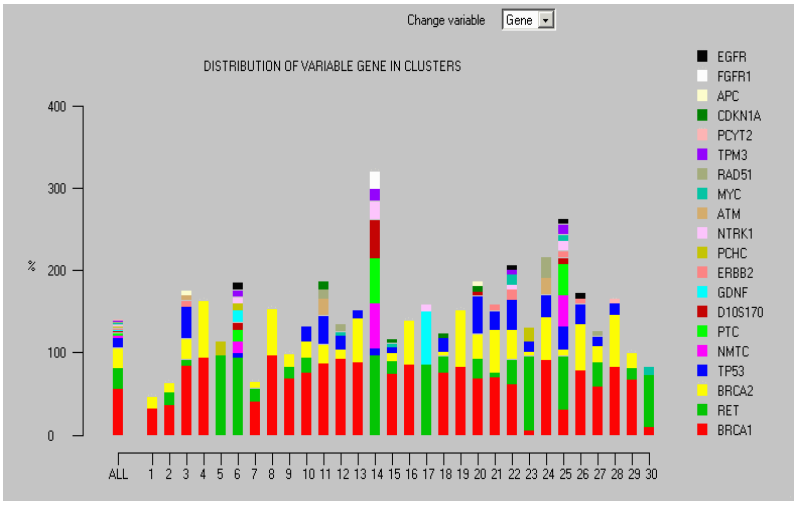

Figura 5. 8 : Istogramma relativo alla distribuzione dei geni in cluster

Dalla figura 5.7 si nota che documenti selezionati tendono a trattare dei due geni separatamente, come era prevedibile, dato che un gene è implicato nel tumore al polmone e l'altro nel tumore al seno e che le interrelazioni dovrebbero essere minime. Viceversa si può notare che, quando compare BRCA1 (colore rosso), spesso compare anche BRCA2 (colore giallo), quindi tra questi due geni deve esserci una interrelazione.

Tornando al gruppo di cluster individuato sulla mappa, possiamo notare che si tratta proprio di quei cluster che fanno riferimento solamente a BRCA1 (il colore verde, che rappresenta RET, non compare nemmeno in piccola percentuale).

I gruppi di documenti che, al contrario, trattano unicamente del gene RET (barre dove il colore rosso è assente) sono anch'essi evidenziabili sulla mappa come un insieme riconoscibile (nella mappa è stato cerchiato di verde) anche se non totalmente separato dagli altri. In questo caso sono le parole chiave "man", "thyroid", "thyrosine", … a comparire più frequentemente.

CAPITOLO 6

Tecniche formali di valutazione dei modelli

6.0 Introduzione

La valutazione della prestazioni è probabilmente il passo più critico del processo di DM infatti i problemi da affrontare e le domande a cui rispondere sono molteplici. Bisogna chiedersi se i benefici che si ricevono da un progetto di data mining sono maggiori dei costi sostenuti, bisogna capire come interpretare i risultati di una sessione di data mining e se è possibile fidarsi dei risultati di tale sessione. Ciascuno di questi punti è abbastanza articolato, più di tutti il primo perché mette in gioco svariati fattori e consente di capire se vale la pena o meno ricorrere al DM.

6.1 Valutare i modelli di apprendimento supervisionato: le matrici di confusione

I modelli di apprendimento supervisionato sono ideati per classificare, stimare e/o prevedere comportamenti futuri. Per alcune applicazioni lo scopo è quello di costruire modelli con un alto livello di accuratezza delle previsioni ad esempio un modello che accetta o rifiuta le richieste di una carta di credito così come un modello per decidere se effettuare o meno una trivellazione per trovare il petrolio, richiedono un'elevata accuratezza. La correttezza della classificazione può essere valutata meglio presentando i dati incogniti sotto forma di un set di prova al modello che deve essere valutato. L'accuratezza del modello dei dati di prova può essere riassunta in una tabella detta **matrice di confusione**, che è una matrice avente le seguenti caratteristiche:

- I valori della diagonale principale rappresentano le classificazioni corrette. L'elemento C_{ii} rappresenta il numero totale di casi della classe C_i che sono stati correttamente classificati dal modello

- I valori della riga i rappresentano quei casi che appartengono alla classe C_i
- I valori della colonna i indicano i casi classificati come membri della classe C_i (anche quelli erroneamente classificati)

I dati di una matrice di confusione possono essere utilizzati per valutare l'accuratezza del modello; per farlo basta sommare i valori contenuti nella diagonale principale e dividere il risultato per il numero totale di casi impiegati come dati di prova.

Per esempio, se applichiamo il modello a 100 casi di prova e la somma dei valori lungo la diagonale principale della matrice di confusione è pari a 70, l'accuratezza del modello è pari a 70% o 0.70. Da questo valore spesso viene ricavato il **tasso di errore** come complemento a 1 dell'accuratezza.

L'uso delle matrici di confusione è fondamentale perché ci fanno capire non solo che il classificatore sbaglia ma anche quanto sbaglia e dove sbaglia; tali matrici mostrano sia in che modo il modello predice bene, sia presentano i dettagli necessari per indicare con precisione dove le cose vanno male.

Una matrice di confusione non è molto utile per valutare i risultati numerici dei modelli di apprendimento; inoltre il concetto di correttezza della classificazione assume un significato diverso con i **modelli con output numerico** perché le osservazioni non possono essere direttamente suddivise in una delle varie classi di output. Per questi tipi di modelli si utilizzano altri parametri più comuni quali **l'errore assoluto medio**, che, per un insieme di dati di prova, è definito come media delle differenze assolute fra i valori calcolati e quelli reali, **l'errore quadratico medio**, che è dato dalla media del quadrato delle differenze assolute fra i valori calcolati e quelli reali e infine **l'errore rms** (root mean squared error), che è semplicemente la radice quadrata dell'errore quadratico medio.

6.2 Cosa dovrebbe essere valutato?

I componenti fondamentali per creare e testare un modello di apprendimento supervisionato sono illustrati in figura 5.1.

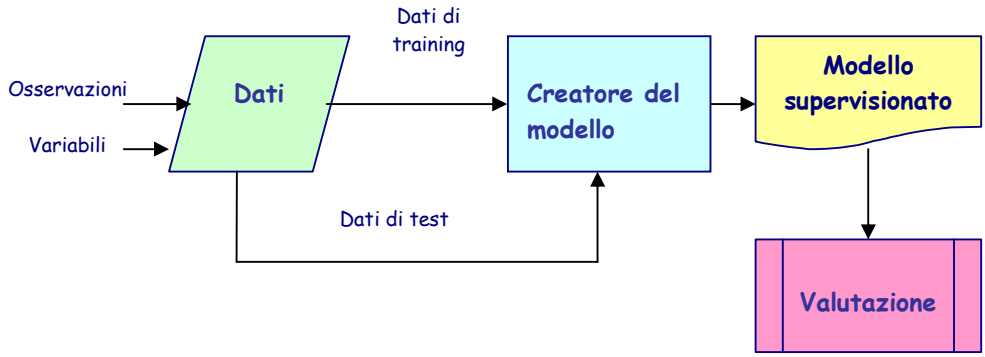

Figura 6. 1: Componenti dell'apprendimento supervisionato

Tutti gli elementi presenti nello schema contribuiscono alle performance del modello creato per cui quando un modello non soddisfa le richieste, una soluzione può essere valutare l'effetto di ogni elemento sulle prestazioni complessive. I modelli supervisionati generalmente vengono valutati sulla base dei dati di test. Il fine del test set consiste nell'offrire una misura delle performance future del modello. Dovrebbe essere data particolare attenzione alle differenti tipologie di errori di errata classificazione; bisognerà quindi introdurre strumenti di valutazione aggiuntivi come gli intervalli di confidenza per gli errori del test set relativamente a modelli supervisionati con output categorico.

Se un modello supervisionato mostra un'accuratezza di test set bassa, una parte del problema può risiedere nei dati di training. I modelli costruiti sulla base dei dati di training che non rappresentano tutte le tipologie di osservazioni o contengono una grande quantità di osservazioni atipiche, probabilmente avranno performance soddisfacenti.

La migliore misura rappresentativa consiste nella scelta casuale dei dati di training, in modo che le classi in essi contenute siano distribuite come nella popolazione. La procedura che assicura una distribuzione appropriata dei dati è nota come **stratificazione**; una soluzione alternativa consiste nell'esaminare il livello di rappresentatività dei dati. Si presenta spesso anche un problema di valutazione delle variabili per capire se esse siano in grado di definire le tipologie di osservazioni. Inoltre, anche se capita che modelli di apprendimento supervisionato,costruiti con tecniche alternative, diano luogo a tassi di errore di classificazione comparabili, bisogna capire perché si preferisce una tecnica ad un'altra; per esempio, le reti neurali tendono a non essere efficienti, rispetto ad altre tecniche, quando i dati di training contengono molte osservazioni mancanti o errate.

Molti modelli di DM supportano uno o più parametri di apprendimento specificati dall'utente; le impostazioni di tali parametri può avere un effetto marcato sulle performance del modello. Le tecniche utilizzate per comparare modelli costruiti con tecniche di data mining differenti possono essere usate anche per comparare modelli costruiti con gli stessi metodi di DM ma con parametri di apprendimento alternativi.

La figura 5.1 è riferita anche alla clusterizzazione non supervisionata, con la differenza che non si hanno dati di test che contengono osservazioni con classificazione nota.

6.3 Parametri statistici a un solo valore

Una popolazione di dati numerici è definita in modo univoco dalla media, dalla deviazione standard e dalla distribuzione di frequenza o di probabilità dei valori presenti nei dati. La **media** μ viene calcolata sommando le osservazioni e dividendo il risultato per il numero di osservazioni. Mentre la media designa un valore medio, la **varianza** σ^2 rappresenta la dispersione attorno a tale media e si calcola sottraendo ciascun valore dalla media, elevando al

quadrato e sommando tutto e infine dividendo tutto per il numero totale di osservazioni. La **deviazione standard** è semplicemente la radice quadrata della varianza. Quando calcoliamo la media, la varianza o la deviazione standard per un campione, i simboli cambiano. La media di un campione si indica con $\overline{X}$ mentre la varianza si indica con *V*.

La media e la varianza sono parametri statistici utili per riassumere i dati. Tuttavia, due popolazioni, pur avendo media e varianza molto simili possono mostrare differenze molto marcate nelle singole osservazioni. Perciò, per permettere una comprensione completa dei dati, **è necessario conoscere la distribuzione delle osservazioni all'interno della popolazione**, cosa che è semplice quando i dati sono pochi ma diventa difficile da determinare nelle grandi popolazioni.

6.4 Distribuzione normale e media campionaria

Una distribuzione fondamentale largamente studiata è la distribuzione normale, nota anche come **curva gaussiana**; infatti sono state sviluppate svariate statistiche molto utili per le popolazioni con distribuzione normale. Dal momento che molti dati non si distribuiscono lungo una gaussiana, qual è l'importanza di questo discorso nell'ambito del DM? In realtà, anche se una variabile si distribuisce normalmente, dobbiamo trattare osservazioni che contengono svariati valori numerici, molti dei quali non si distribuiscono normalmente. Il nostro obiettivo è quello di utilizzare le proprietà della distribuzione normale per valutare meglio le performance dei modelli di data mining.

Molte popolazioni di una certa rilevanza sono piuttosto grandi e ciò rende estremamente difficile l'analisi sperimentale, per questo motivo gli esperimenti sono effettuati su sottinsiemi di dati scelti casualmente dalla popolazione totale. Quando effettuiamo un campionamento da una popolazione, non

possiamo essere sicuri che la relativa distribuzione dei valori sia normale; questo vale anche quando la popolazione ha una distribuzione normale. Tuttavia, esiste una situazione particolare che garantisce la normalità, ovvero:

> *Per una data popolazione, la distribuzione delle medie prese da insiemi casuali di campioni indipendenti di uguale dimensione è normale.*

In sintesi ciò significa che ognuna delle medie campionarie ottenute costituisce una stima non distorta della media della popolazione inoltre la media delle medie campionarie prese tra tutti i possibili campioni di uguale dimensione è esattamente pari alla media della popolazione. Non abbiamo ancora, però, un valore che ci indica la bontà del valore medio calcolato. Calcoliamo quindi la varianza della popolazione che è data da V/n, dove V è la varianza campionaria e n è il numero delle osservazioni campionarie. Dobbiamo poi calcolare **l'errore standard**, che è semplicemente la radice quadrata della varianza stimata della popolazione:

$$SE = \sqrt{V/n}$$

Dal momento che la popolazione delle medie campionarie è distribuita normalmente e l'errore standard è una stima della varianza della popolazione, possiamo affermare che:

> *Ogni media campionaria varierà meno di più o meno due errori standard dalla media della popolazione nel 95% dei casi.*

Questa tecnica, oppure il modello di verifica delle ipotesi illustrato nel prossimo paragrafo, può essere utilizzato per semplificare il calcolo degli intervalli di confidenza per i tassi di errore dei test set, per confrontare i tassi di

errata classificazione di due o più modelli e per determinare le variabili numeriche che differenziano meglio le classi o i cluster.

6.5 Un modello classico per testare le ipotesi

Un'ipotesi non è altro che una supposizione mirata sul risultato di qualche evento. Testare le ipotesi è una pratica comune nelle applicazioni della scienza e della medicina e in generale in ogni campo.
Si procede innanzitutto con una selezione, da una popolazione, di due gruppi: per effettuare l'esperimento selezioniamo un gruppo come gruppo sperimentale mentre il secondo gruppo viene definito gruppo di controllo.
Una procedura classica consiste nel definire un'ipotesi da testare sui risultati dell'esperimento. Tipicamente il risultato è definito nella forma di **ipotesi nulla**.
L'ipotesi nulla assume un punto di vista negativo nel momento in cui asserisce che ogni relazione rilevata come risultato dell'esperimento sia dovuta unicamente al caso.
Una volta che l'esperimento viene effettuato e i dati che mostrano i risultati dell'esperimento sono stati raccolti, verifichiamo se il risultato presenta una differenza significativa tra i due gruppi.
Un modello classico per testare la differenza significativa tra le medie dei parametri misurati è dato dalla seguente formula:

$$P = \frac{\left|\overline{X_1} - \overline{X_2}\right|}{\sqrt{v_1/n_1 + v_2/n_2}}$$

dove:

- P è il livello di importanza
- $\overline{X_1}$ e $\overline{X_2}$ sono le medie campionarie per i campioni indipendenti
- v_1 e v_2 sono le varianze delle rispettive medie
- n_1 e n_2 sono le dimensioni dei rispettivi campioni

Il termine a numeratore è la differenza assoluta tra le due medie campionarie. Per testare una differenza significativa tra le medie, la differenza viene divisa per l'errore standard della distribuzione delle differenze medie. Dal momento che i due campioni sono indipendenti, l'errore standard è semplicemente la radice quadrata della somma delle due varianze associate alle medie campionarie.

Per essere sicuri al 95% che la differenza tra le due medie non sia dovuta al caso, il valore P nell'equazione deve essere maggiore o uguale a 2. Il modello è valido perché, come la distribuzione delle medie prese da insiemi di campioni indipendenti di uguale dimensione, anche la distribuzione delle differenze tra le medie campionarie è normale. Un livello di confidenza del 95% lascia ancora spazio all'errore e la verifica d'ipotesi considera due categorie generali di errore.

L'**errore di primo tipo** si verifica quando viene rifiutata un'ipotesi nulla vera.

L'**errore di secondo tipo** si osserva quando viene accettata un'ipotesi nulla falsa.

Queste due tipologie di errore sono mostrate nella seguente matrice di confusione:

	Accettate (calcolate)	*Rifiutate (calcolate)*
Ipotesi nulla accettata	Accettata correttamente	Errore di primo tipo
Ipotesi nulla rifiutata	Errore di secondo tipo	Rifiutata correttamente

Figura 6. 2: Matrice di confusione per l'ipotesi nulla

Una condizione per questa metodologia richiede che ogni media sia calcolata da un dataset indipendente. Con il data mining, il caso tipico consiste nel confrontare modelli con l'aiuto di test set singoli dei dati. Fortunatamente, possiamo apportare una piccola modifica al modello classico per adattarlo alla situazione in cui abbiamo un solo dataset di test.

Soffermiamoci su una struttura sperimentale standard che utilizza un esempio ipotetico nell'ambito medico.

Supponiamo di voler determinare gli effetti di un nuovo farmaco, il trattamento X, sviluppato per trattare i sintomi associati a un'allergia alla polvere domestica.

Per l'esperimento, scegliamo casualmente due gruppi da una popolazione selezionata di soggetti che soffrono di allergia. Il trattamento X è applicato al gruppo sperimentale mentre al gruppo di controllo viene somministrato un placebo nella forma di una pillola da zucchero; si fa in modo che ogni paziente non sappia a quale gruppo appartenga.

L'aumento o la diminuzione media del numero totale di reazioni allergiche giornaliere viene registrata per ogni paziente in entrambi i gruppi. Dopo un certo periodo di tempo, utilizziamo un test statistico per determinare se si rileva una differenza significativa nei parametri misurati nei due gruppi. Nel nostro esempio, l'ipotesi nulla afferma che non ci saranno differenze significative tra i due gruppi di persone.

Un'ipotesi nulla plausibile per il nostro esperimento è la seguente: non ci sono differenze significative nell'aumento o nella diminuzione media delle reazioni allergiche totali giornaliere tra i pazienti del gruppo che ricevono il trattamento X e i pazienti del gruppo che ricevono il placebo. Si noti che l'ipotesi nulla sostiene che non ci sono differenze significative, senza far riferimento agli eventuali miglioramenti; la ragione sta nel fatto che non abbiamo garanzie a priori che il trattamento X non causerà effetti collaterali e peggiorerà i sintomi associati all'allergia.

Nel nostro esperimento, un errore del primo tipo ci lascerebbe capire che il trattamento X ha un impatto significativo sul numero medio di reazioni allergiche alla polvere quando non è vero. Allo stesso modo, un errore di secondo tipo affermerebbe che il trattamento X non influisce sul numero medio di reazioni allergiche quando invece è vero il contrario.

6.6 Calcolare intervalli di confidenza per i test set

I dati di training e di prova possono essere supportati dai **dati di convalida** (validation data). Uno scopo dei dati di convalida consiste nell'aiutare a scegliere uno fra i tanti modelli costruiti a partire dallo stesso dataset di training. Una volta provato, ogni modello è presentato ai dati di convalida; viene scelto come modello finale quello che mostra la migliore correttezza di classificazione. I dati di convalida possono essere utilizzati anche per ottimizzare le impostazioni dei parametri di un modello supervisionato.
Una volta convalidato, il modello supervisionato viene valutato applicando i dati di test.
La misura più generale della performance di un modello è il tasso di errore del classificatore (classifier error rate), ovvero:

$$\textit{Tasso di errore del classificatore} = \frac{Numero \quad di \quad errori \quad del \quad test \quad set}{Numero \quad di \quad osservazioni \quad del \quad test \quad set}$$

Lo scopo di questo parametro consiste nel fornire un'indicazione sulla futura probabile performance del modello ma come facciamo ad essere sicuri che il tasso di errore sia una misura valida della performance attuale del modello? Per rispondere a questa domanda, possiamo utilizzare l'errore standard statistico per calcolare un intervallo di confidenza del tasso di errore per la performance del modello. Per applicare l'errore standard, consideriamo il tasso di errore del classificatore come una media campionaria. Nonostante il tasso di errore in realtà sia una proporzione, se il numero di osservazioni del test set è sufficientemente grande ($n > 100$), il tasso di errore può rappresentare un valore medio.
Per calcolare un intervallo di confidenza per il tasso di errore calcolato, prima calcoliamo l'errore standard associato al tasso d'errore del classificatore.

L'errore standard insieme al tasso di errore viene utilizzato per calcolare l'intervallo di confidenza.
La procedura è la seguente:

1. Sia dato un campione di test set S di dimensione n e un tasso d'errore E.
2. Si calcola la varianza campionaria: $Varianza(E) = E \cdot (1 - E)$
3. Si calcola l'errore standard che è pari alla $Varianza(E)$ divisa per n
4. Si calcola un limite superiore per un intervallo di confidenza al 95% come $E + 2E(SE)$
5. Si calcola un limite superiore per un intervallo di confidenza al 95% come $E - 2E(SE)$

Esaminiamo un esempio per capire meglio come calcolare un intervallo di confidenza. Supponiamo che un classificatore mostri un tasso d'errore del 10% quando viene applicato ad un campione casuale di 100 osservazioni dal test set. Fissiamo $E = 0.10$ e calcoliamo la varianza campionaria come:

$$Varianza(0.10) = 0.10 \cdot (1 - 0.10) = 0.09$$

Con un test set di 100 osservazioni, il calcolo dell'errore standard è:

$$SE = \sqrt{0.09 \big/ 100} = 0.03$$

Possiamo essere sicuri al 95% che il tasso attuale di errore del test set si trovi tra due volte l'errore standard sotto 0.10 e due volte sopra l'errore standard sopra 0.10; questo ci dice che il tasso di errore cade tra 0.04 e 0.16, cui corrisponde un'accuratezza del test set tra l'84% e il 96%.

Se aumentiamo il numero di osservazioni del test set, siamo in grado di diminuire l'ampiezza del range di confidenza; se, ad esempio, consideriamo 1000 osservazioni l'errore standard diventa pari a 0.0095 e facendo gli stessi calcoli di prima viene fuori che l'accuratezza adesso è compresa tra l'88% ed il 92%.

Questo esempio chiarisce bene che l'ampiezza del test set ha un effetto marcato sul range dell'intervallo di confidenza per cui se la misura del test set diventa infinitamente grande, l'errore standard tende a zero.

Bisogna fare però tre osservazioni importanti:

1. L'intervallo di confidenza è valido solo se i dati di test sono stati scelti casualmente dall'insieme di tutte le possibili osservazioni.
2. I dati di prova, training e convalida devono rappresentare insiemi disgiunti.
3. Se possibile, le osservazioni in ciascuna classe dovrebbero essere distribuite nei dati di training, convalida e prova allo stesso modo in cui appaiono nell'intero dataset.
4. Se non sono disponibili ampi dati di test, possiamo applicare una tecnica nota come **cross validation**. Con questo metodo, tutti i dati disponibili vengono ripartiti in n unità di dimensione costante; $n-1$ unità sono utilizzate per training, mentre l'n-esima unità diventa il test set. Questo processo viene ripetuto finchè ogni unità non viene utilizzata come test set.

La correttezza del modello di prova è calcolata come l'accuratezza media ottenuta dagli n tentativi di training-test.

I risultati sperimentali hanno mostrato un valore pari a $10 \cdot n$ come massimo valore in molte situazioni. Numerose applicazioni di cross validation ai dati possono aiutare ad assicurare una distribuzione uguale delle classi all'interno dei dataset di training e di test.

Il **bootstrapping** è un'alternativa alla cross validation. Con esso, si permette al processo di selezione del training set di scegliere le stesse osservazioni di training svariate volte. Questo accade riposizionando ciascuna osservazione di training nell'insieme dei dati, dopo che è stata selezionata per il training. Si può mostrare matematicamente che se un dataset contenente n osservazioni viene campionato n volte utilizzando la tecnica del bootstrapping, il training set conterrà approssimativamente 2/3 delle n osservazioni. Di conseguenza rimane un terzo delle osservazioni per il test.

Abbiamo già visto che il tasso di errore del test set rappresenta solo una delle svariate considerazioni nella determinazione del valore di un modello supervisionato. Facciamo riferimento ad un'applicazione che richiede una buona correttezza di classificazione: supponiamo di sviluppare un modello che accetta o rifiuta le richieste di una carta di credito. Supponiamo che in media lo 0.5 di tutti i pagamenti con carta di credito sia dovuto ad operazioni fraudolente. Un modello, creato per identificare casi di frode con la carta di credito, che afferma che ogni acquisto con la carta è sempre valido avrà un'accuratezza dell99.5%. Tuttavia un modello del genere è inutile perché non è in grado di svolgere il compito per cui è stato progettato; al contrario, un modello che individua correttamente tutti i casi di frode con le carte di credito, a costo di mostrare un alto tasso di classificazione di falsi positivi, ha un valore maggiore.

Un modo per affrontare questo aspetto consiste nell'assegnare dei pesi alle classificazioni errate. Nell'esempio delle carte di credito, possiamo assegnare un peso grande alle classificazioni errate che non permettono di scoprire una carta fraudolenta e un peso più piccolo agli errori di errata identificazione di una carta fraudolenta. In questo modo, il modello vedrà aumentare il suo tasso di errore di classificazione se mostra una distorsione tale che l'utilizzo improprio di una carta di credito non venga scoperto.

6.7 Confronto dei modelli supervisionati

Il confronto di due modelli supervisionati costruiti con gli stessi dati di training può essere fatto applicando il paradigma classico della verifica di ipotesi. La misura delle performance del modello ancora una volta è il tasso di errore del test set.
Formuliamo il problema nella forma di un'ipotesi nulla:

> *Non c'è differenza significativa nel tasso di errore del test set di due modelli supervisionati* M_1 *e* M_2*, costruiti con gli stessi dati di training.*

Ci sono tre possibili scenari di test set:

1. L'accuratezza del modello è confrontata tramite l'utilizzo di due test set indipendenti selezionati casualmente da un insieme di dati campionati.
2. Gli stessi dati di test set sono impiegati per confrontare i modelli. Il confronto è basato su un calcolo a coppia, osservazione per osservazione.
3. Gli stessi dati di test sono utilizzati per confrontare la correttezza di classificazione totale del modello.

Dal punto di vista statistico l'approccio più chiaro è il primo perché in tal caso si può applicare il modello di verifica delle ipotesi. Tale approccio è appropriato solo se c'è un'ampia disponibilità di dati di test. Con grandi dataset, la possibilità di estrarre dati di test indipendenti è reale tuttavia, con dataset piccoli, si può ottenere un solo test set.
Quando viene applicato lo stesso test set ai dati, l'opzione consiste nell'effettuare l'accoppiamento, osservazione per osservazione, dei risultati del test set. Descriveremo una tecnica semplificata che confronta la correttezza totale dei due modelli. Il metodo può essere applicato allo scenario 1 ed allo

scenario 3. La forma più generale per il confronto di due modelli di classificazione M_1 e M_2 è la seguente:

$$P = \frac{|E_1 - E_2|}{\sqrt{q \cdot (1-q) \cdot \left(1/n_1 + 1/n_2 \right)}}$$

Dove:

E_1 = Tasso di errore per il modello M_1

E_2 = Tasso di errore per il modello M_2

$q = \frac{E_1 + E_2}{2}$

n_1 = Numero di osservazioni nel test set A

n_2 = Numero di osservazioni nel test set B

Il prodotto $q \cdot (1-q)$ è una varianza calcolata utilizzando la media di due tassi d'errore.

Con un singolo test set di ampiezza , la formula si semplifica così:

$$P = \frac{|E_1 - E_2|}{\sqrt{q \cdot (1-q) \cdot (2/n)}}$$

In entrambe le equazioni se il valore di $P \geq 2$, possiamo essere sicuri al 95% che la differenza nella performance del test set di M_1 e M_2 è significativa.

Esempio:

Supponiamo di voler confrontare la performance di test set di due modelli di apprendimento M_1 e M_2. Testiamo M_1 sul test set A e M_2 sul test set B. Ciascun test set contiene 100 osservazioni; supponiamo che M_1 raggiunga un'accuratezza di classificazione dell'80% con l'insieme A ed M_2 del 70% con

l'insieme B. Vorremmo sapere se la performance del modello M_1 sia significativamente migliore del modello M_2.

Ovviamente $E_1 = 0.20$, $E_2 = 0.30$ per cui $q = 0.25$ e facendo i calcoli $P = 1.633$ quindi essendo minore di due la differenza nella performance del modello non è significativa. Possiamo aumentare la nostra fiducia nel risultato scambiando i due test set e ripetendo l'esperimento. Questo è particolarmente importante se la differenza significativa è vista con la selezione del test iniziale. La media dei due valori per P viene in seguito utilizzata per il test di significatività.

6.8 Tecniche di valutazione non supervisionata

L'apprendimento supervisionato e la clusterizzazione non supervisionata si possono considerare complementari nel senso che ciascun approccio può essere usato per valutare la strategia opposta.

Quando la clusterizzazione non supervisionata viene utilizzata per la valutazione supervisionata, le osservazioni selezionate per il training supervisionato sono utilizzate dalla tecnica di clusterizzazione. Se le osservazioni si raggruppano nelle classi predefinite contenute nei dati di training, il modello supervisionato costruito con i dati di training è valido. Se le osservazioni non si raggruppano nelle classi note, il modello supervisionato non è efficiente.

Nonostante questo approccio sia chiaro, potrebbero occorrere parecchie iterazioni prima che possa essere fatta una valutazione. Supponiamo, per esempio, di utilizzare l'algoritmo delle k-medie. I cluster formati dall'applicazione di tale algoritmo sono fortemente condizionati dalla scelta iniziale delle media dei cluster. Per una singola iterazione, l'algoritmo potrebbe ottenere una convergenza non ottimale e quindi un singolo cluster potrebbe contenere un insieme di osservazioni appartenenti a classi differenti.

Dal momento che la valutazione non supervisionata è basata solamente sui dati di training, la qualità di clusterizzazione non garantisce una performance accettabile basata sul test set. Per questa ragione, la tecnica completa gli altri metodi di valutazione ed è molto utile per identificare la ragione del fallimento del modello supervisionato.

Vediamo ora in che modo l'apprendimento supervisionato possa aiutare a spiegare e a valutare i risultati della clusterizzazione.

In generale, la valutazione di un processo di data mining non supervisionato è un compito più difficile della valutazione di un modello di apprendimento supervisionato. Ciò è vero in quanto gli obiettivi di una sessione di data mining non supervisionato spesso non sono così chiari come quelli dell'apprendimento supervisionato.

Tutte le tecniche di clusterizzazione non supervisionata misurano in qualche modo la qualità dei cluster; una tecnica comune consiste nel calcolare la somma del quadrato delle differenze degli errori tra i casi inseriti in ogni cluster e il valore centrale del cluster corrispondente. **Quanto più piccoli sono i valori delle somme dei quadrati delle differenze degli errori tanto maggiore è la qualità dei cluster.**

Tuttavia per valutare dettagliatamente la clusterizzazione non supervisionata, occorre utilizzare l'apprendimento supervisionato e si procede nel seguente modo:

1. Viene effettuata una clusterizzazione non supervisionata e si definisce ogni cluster come una classe e si assegna ad ogni cluster un nome arbitrario.
2. Si sceglie un campione casuale di casi da ciascuna delle classi generate dal processo di clusterizzazione. Ogni classe dovrebbe essere rappresentata nel campione casuale con lo stesso rapporto con cui è rappresentata nell'intero dataset. La percentuale dei casi totali da

campionare può variare; tuttavia, per iniziare, si prova a campionare i due terzi di tutti i casi.

3. Si costruisce un modello di apprendimento supervisionato con il nome della classe come attributo di output, utilizzando i dati campionati a caso come dati di training. Il restante dei casi viene utilizzato per provare la correttezza di classificazione del modello supervisionato cioè si testa il modello supervisionato con le osservazioni rimanenti.

Questo metodo di valutazione ha almeno due vantaggi infatti la clusterizzazione non supervisionata può essere vista come una struttura supportata da un modello di apprendimento supervisionato. Per esempio, i risultati di una clusterizzazione creata da un algoritmo non supervisionato possono essere visti come un albero decisionale o una struttura basata su regole. Un secondo vantaggio è che il punteggio di correttezza della classificazione dei dati di prova può essere un ulteriore indice di valutazione della qualità dei cluster formati.

Dal momento che le tecniche non supervisionate mancano di uno strumento di spiegazione, la classificazione supervisionata aiuta ad analizzare i cluster formati.

Una piccola variazione di questa tecnica consiste nel costruire un modello supervisionato con le prime n osservazioni da ciascun cluster; in questo modo si possono esaminare le performance del modello dal punto di vista delle osservazioni che meglio definiscono ciascun cluster.

6.9 Metodi aggiuntivi per la valutazione della clusterizzazione non supervisionata

La valutazione non supervisionata spesso è un processo in due fasi. Una prima valutazione è interna e coinvolge una ripetizione del processo finchè non sono

raggiunti livelli ottimali prefissati dei parametri. La seconda valutazione è esterna e indipendente dal modello che crea i cluster.

Un metodo aggiuntivo per la valutazione esterna è utilizzare un apprcccio supervisionato e designare tutte le osservazioni come dati di training. Questa variante è appropriata se è di primaria importanza una spiegazione completa delle differenze tra i cluster. Un altro metodo è creare una propria misura della qualità dei cluster, si parla in tal caso di clusterizzazione agglomerativa, che è una tecnica che unisce le osservazioni a ogni passo del processo di clusterizzazione finchè non è soddisfatta una condizione finale.

Infine si potrebbe effettuare un paragone tra i valori delle variabili dei cluster e se essi differiscono in modo significativo, si può concludere che anche le osservazioni differiscono in modo significativo.

CAPITOLO 7

Il software Weka

7.1 Introduzione a Weka

Weka, sviluppato dall'università di Waikato in Nuova Zelenda, è una suite di strumenti di data mining, si tratta di una libreria Java Open Source, distribuita sotto GNU (General public license), per il preprocessing e il DM, dotata di una buona interfaccia grafica. Weka è l'acronimo di Waikato Environment for Knowledge Analysis ma è anche il nome di un uccello della Nuova Zelanda dalla natura particolarmente curiosa e inquisitrice, natura che quindi ben si sposa con l'obiettivo primario del software che è quello di scoprire la conoscenza "nascosta" nei dati [7.1]. Esistono diverse versioni di Weka; noi ci riferiamo alla versione 3.4.1, che è caratterizzata da una serie di miglioramenti e di aggiunte rispetto alle versioni precedenti. Essendo scritto in Java, esso può essere utilizzato su qualunque sistema operativo dotato di un ambiente di esecuzione Java. Se il JRE (Java Runtime Environment) è già installato, basta scaricare il solo programma di installazione Weka ed eseguirlo altrimenti la cosa più conveniente è scaricare il programma di installazione Weka+JRE che installa entrambi gli ambienti in una volta sola.

Una volta lanciato Weka, possiamo scegliere tra quattro diversi ambienti operativi:

- Simple CLI: è un ambiente a linea di comando, da usare per invocare direttamente le varie classi Java di cui Weka è composto. Tutto quello che si può fare dalla SimpleCLI è possibile farlo anche da un ambiente a linea di comando come il prompt del DOS di Windows o la shell di Unix. L'ambiente SimpleCLI è fornito soltanto per quei sistemi, come il MacOS 9, che non sono dotati di un ambiente a linea di comando.
- Explorer: è un ambiente utilissimo infatti consente di caricare degli insiemi di dati, di visualizzare in modo grafico la disposizione degli attributi, effettuare una serie di operazioni preliminari di preparazione

ed eseguire algoritmi di classificazione, clustering, selezione di attributi e determinazione delle regole associative.

- **Experimenter:** è una versione batch dell'Explorer. Consente di impostare una serie di analisi, su vari insiemi di dati e con vari algoritmi, ed eseguirle alla fine tutte insieme. In questo modo è possibile confrontare i risultati di algoritmi differenti e determinare quale è il più adatto a uno specifico insieme di dati.
- **Knowledge Flow:** è una variante dell'Explorer, in cui le operazioni da eseguire si esprimono in un ambiente grafico, disegnando un diagramma che esprime il "flusso della conoscenza". L'ambiente consente di selezionare varie componenti come sorgenti di dati, filtri, algoritmi di classificazione e collegarli tra loro in un diagramma tipicamente detto "**data-flow**" (in quanto mostra il flusso di dati attraverso varie componenti di un sistema).

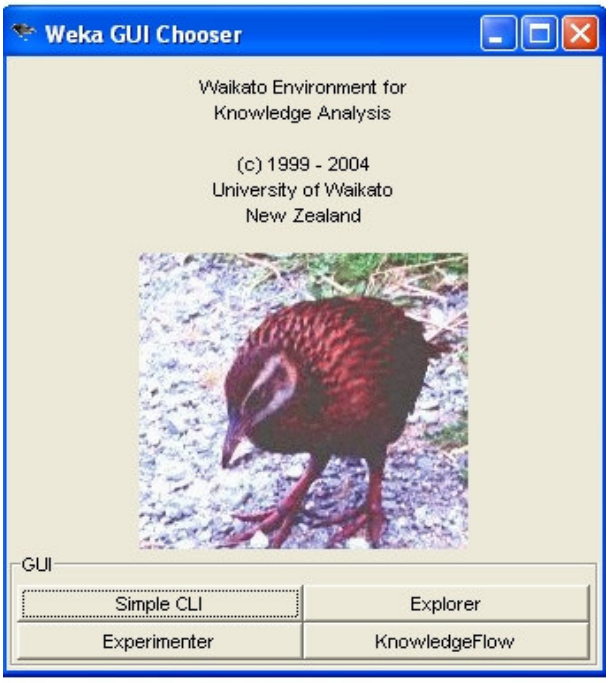

Figura 7. 1: Finestra iniziale di Weka

7.2 Il formato ARFF

Weka utilizza il formato ARFF (Attribute Relation File Format) per i file di ingresso; il file consta di due parti, una di intestazione e una sezione dati. Di seguito è riportato un esempio di file ARFF che descrive un dataset di 14 casi: ogni caso rappresenta una giornata ed è caratterizzato da attributi relativi alle condizioni climatiche e all'attributo di classe *play* (con due valori possibili *yes* e *no,* corrispondenti, rispettivamente, alla classe delle giornate in cui si può giocare e alla classe delle giornate in cui non è possibile).

```
@relation  weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy  {TRUE, FALSE }
@attribute play  {yes, no }

@data
sunny, 85, 85, FALSE, no
sunny, 80, 90, TRUE, no
overcast, 83, 90, TRUE, no
rainy, 70, 96, FALSE, yes
rainy, 68, 80, FALSE, yes
rainy, 65,70, TRUE, no
overcast, 64, 65, TRUE, yes
sunny, 72, 95, FALSE, no
rainy, 75, 80, FALSE, yes
```

Figura 7. 2: Esempio di file in formato ARFF.

La prima riga indica il nome del dataset preceduto dal tag **@relation**. Seguono le descrizioni dei vari attributi, riportate una per riga e precedute da **@attribute**: per ogni attributo è specificato il nome ed il tipo, che può essere numerico (real) o categorico. Nel caso di attributi categorici è necessario specificare i valori che l'attributo può assumere. Per default si assume che

l'ultimo attributo rappresenti la classe e, pertanto, deve essere di tipo categorico. Il tag **@data** introduce, infine, l'elenco dei casi del dataset: ogni caso (tupla) corrisponde ad una riga contenente i valori degli attributi del caso, separati da virgole. L'ordine dei valori corrisponde all'ordine secondo il quale sono stati definiti precedentemente gli attributi. I tipi di dati possibili per gli attributi sono: real, integer o un semplice tipo enumerativo-nominale (per variabili categoriche). Altri formati importanti perché più adatti a problemi specifici sono **string** e **date**. Gli attributi di tipo stringa consentono di avere attributi contenenti arbitrari valori testuali; ciò risulta particolarmente utile in applicazioni di **text mining** perché possono essere creati dataset con attributi stringa che possono poi essere manipolati attraverso filtri Weka (anche scritti dall'utente). Gli attributi stringa sono trattati come attributi nominali con molti valori possibili. Gli attributi date devono essere utilizzati nella forma:

@attribute <name> **date** [<date format>]

- Dove <date format> è una stringa opzionale che specifica come i valori data debbano essere interpretati e stampati.

dati possono comunque essere importati in Weka in vari formati oltre ad ARFF: CSV, C4.5, binary.

7.3 L'ambiente Explorer

L'ambiente Explorer è molto utile in quanto mette a disposizione molteplici funzionalità che consentono all'utente di far emergere nuova informazione dai dati di ingresso. La caratteristica fondamentale è la possibilità che da l'ambiente di preprocessare i dati, prima di essere dati in pasto ad uno degli algoritmi di classificazione o di clustering. Infatti, una volta aperto il file, è possibile applicare dei filtri ai dati; il filtraggio può andare dalla discretizzazione alla traslazione temporale per le serie,alla normalizzazione, alla conversione di tipo. Questa fase è importante e non deve essere sottovalutata, tra l'altro già abbiamo sottolineato il suo ruolo centrale quando abbiamo parlato del KDD. I dati, per essere utilizzati, spesso devono essere trasformati; questa fase può assumere varie forme e può essere necessaria per varie ragioni. Si possono convertire tipi di dati in altri o definirne di nuovi, ottenuti attraverso l'uso di operazioni matematiche e logiche sulle variabili, eseguire delle normalizzazioni (scalamento decimale, normalizzazione min-max o con lo z-score) o addirittura eliminare delle variabili. In genere infatti gli algoritmi di DM non lavorano in modo efficiente se i dati contengono una grande quantità di variabili che non sono in grado di prevedere la classe di appartenenza; si rende quindi utile una ricerca ed una successiva eliminazione delle variabili ridondanti e "inutili" per il problema in questione.

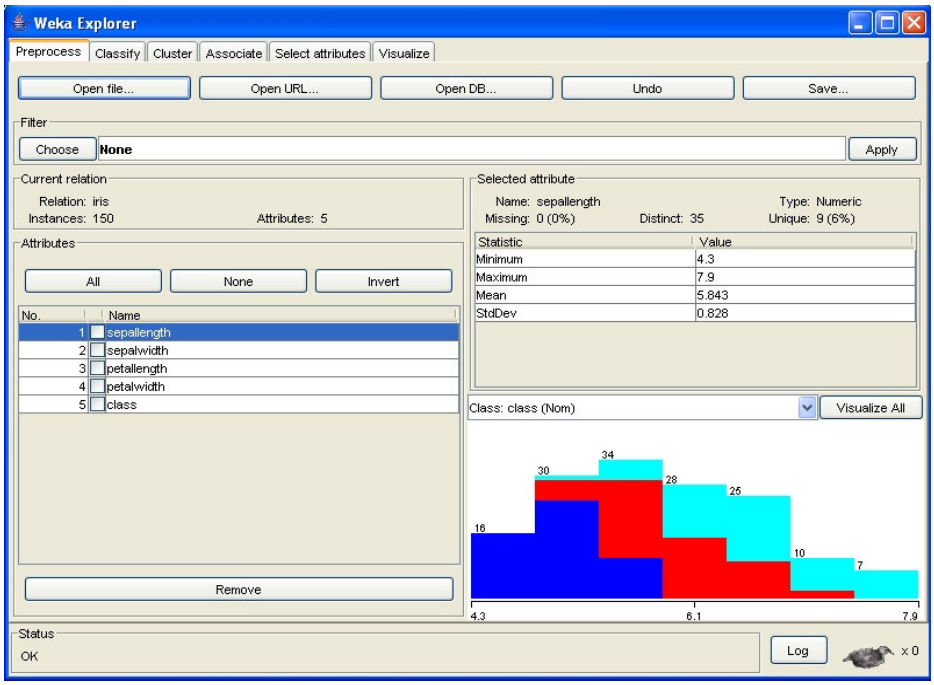

Figura 7. 3: Finestra di apertura di Weka Explorer.

In fase di preprocessing è possibile anche visualizzare la distribuzione dei dati rispetto ai singoli attributi, questa opzione di Weka è di fondamentale importanza in quanto consente di analizzare attentamente i dati per poter capire se alcuni attributi debbano essere esclusi, cioè non presi in considerazione dall'algoritmo di DM che si deciderà di applicare, e quindi consente di eliminare un'eventuale ridondanza o "inutilità" dei dati.

Un' ulteriore funzionalità di Explorer è quella di consentire la visualizzazione di alcuni parametri statistici relativi all'attributo selezionato quali il valore massimo, il valore minimo, il valor medio e la deviazione standard.

Completata la fase iniziale, è possibile scegliere se applicare ai dati un algoritmo di classificazione come un albero di regressione, una rete neurale, un classificatore bayesiano, un albero decisionale. A seconda della scelta fatta è possibile selezionare il tipo di algoritmo che si vuole utilizzare, questa scelta dipende anche dalla tipologia dei dati che si sta trattando, ad esempio se

vogliamo determinare un albero decisionale si potrà scegliere tra l'algoritmo ID3, J48, RandomTree, M5P, etc.

Scelte analoghe andranno fatte se si decide di fare un clustering o anche di ricercare regole di associazioni nei dati, infatti andrà fatta una scelta a monte sul tipo di clustering da effettuare , se gerarchico o a partizioni, e a seconda dei casi andranno scelti i parametri tipici per ogni algoritmo, come ad esempio il numero di cluster che ci si aspetta di trovare in un certo dataset.

Si potrà poi facilmente verificare la correttezza o meno della scelta in quanto Weka consente di visualizzare la distribuzione dei dati nei vari cluster e quindi di capire se effettivamente i dati sono stati "ben partizionati" dall'algoritmo e quindi se questa suddivisione consente realmente di "trovare" qualcosa di nuovo e interessante nei dati a nostra disposizione o se invece bisogna cambiare qualcosa nelle nostre scelte; spesso è necessario eseguire più volte un clustering, magari anche con algoritmi diversi, per capire quale ci consente di raggiungere risultati migliori in relazione ai dati su cui si opera. Tra i vari parametri che bisogna assegnare c'è anche il numero massimo di iterazioni che si vuole consentire di fare in modo da evitare che l'algoritmo venga ripetuto troppe volte inutilmente. Nella figura 7.4 è messo in evidenza lo stretto legame tra la selezione degli attributi, il filtraggio ed il preprocessing; tali fasi sono fortemente correlate in quanto tramite esse si vuole perseguire un medesimo obiettivo che è quello di ottenere un set di dati significativi su cui uno degli algoritmi possa portare a buoni risultati.

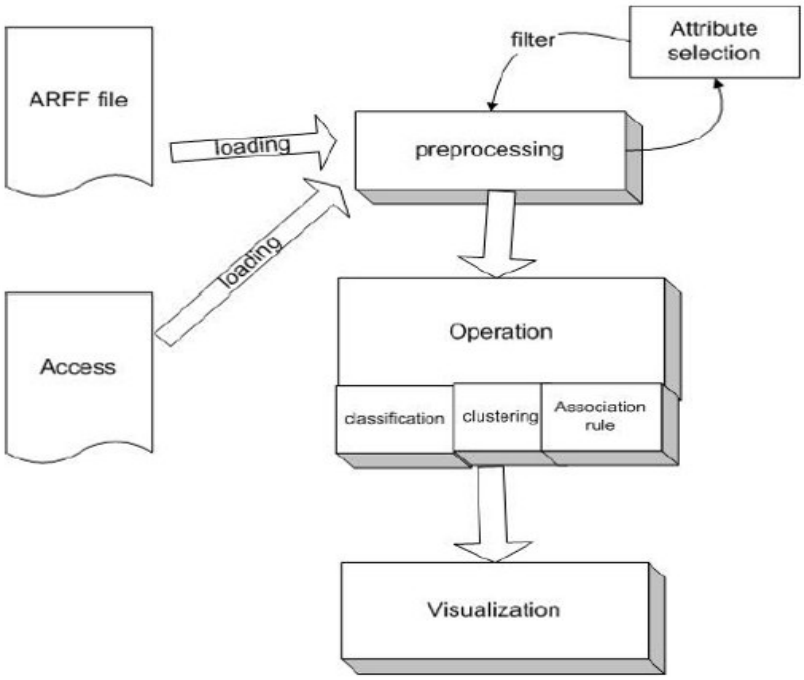

Figura 7. 4: Ciclo di utilizzo di Weka Explorer.

Esistono infine delle opzioni che consentono di testare la validità del modello ottenuto, esse sono mostrate in una finestra come **Test options** e consentono di provare la validità del classificatore sulla base dei soli dati di training o sulla base di un altro file di test set o tramite controlli incrociati o anche suddividendo i dati di ingresso in dati di training e dati di test, specificando la percentuale di dati su cui si vuole testare il modello ottenuto. Una tipologia di testing particolarmente interessante è la cross validation; essa infatti consente di validare un modello, costruito tramite addestramento a partire da un certo training set, e di capire quanto bene funzionerà su un nuovo test set; inoltre consente anche di individuare e prevenire problemi di overfitting, cioè legati a modelli eccessivamente particolareggiati (abbiamo visto questo a proposito degli alberi di decisione infatti è sempre possibile costruire un albero che classifichi perfettamente i dati di training, separando quanto più è possibile le istanze in classi diverse ma si ottiene un modello che è troppo legato

all'addestramento, rendendo difficili le generalizzazioni. La **cross validation** funziona in questo modo:

- Il dataset viene suddiviso in K subset e viene ripetuto K volte il seguente procedimento
- Uno dei K subset è usato come test set e gli altri K-1 sono fusi insieme per formare un training set.

Infine viene calcolato l'errore medio sulle K prove che rappresenta una migliore stima dell'accuratezza del modello. L'unico svantaggio di questo metodo è legato al fatto che l'algoritmo di training deve essere ripetuto K volte. La cross validation viene solitamente impiegata quando l'ammontare di dati è modesto per cui è assurdo pensare ad una ulteriore suddivisione in dati test e di training.

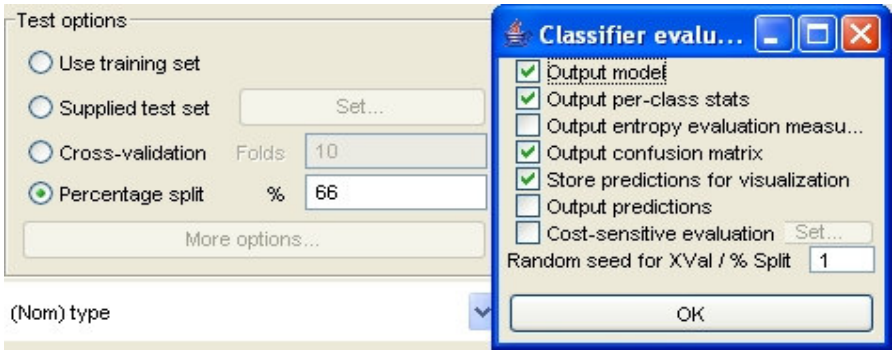

Figura 7 . 5: Test options

7.4 1° caso d'uso: un esempio di analisi tassonomica in botanica.

Vogliamo ora utilizzare Weka per risolvere un semplice problema di classificazione; nel 1936 Fischer descrisse una procedura di identificazione di tre tipi differenti di pianta appartenente alla famiglia Iris sulla base di precise caratteristiche dei petali ma anche dei sepali, cioè di ciascuna foglia formante il calice del fiore dell'Iris.

Figura 7 . 6: Iris setosa, Iris versicolor, Iris virginica.

Come si nota dalle foto in figura 7.6, i tre fiori sono molto simili, sia per quanto riguarda il colore, sia per quanto riguarda la forma dei petali, aventi caratteristiche molto simili pur appartenendo a piante diverse.

Fisher individuò però dei parametri caratterizzanti che consentono di distinguere un fiore da un altro; esse sono la lunghezza e la larghezza dei petali e la lunghezza e la larghezza dei sepali cioè di ciascuna delle foglie verdi formanti l'involucro esterno del fiore.

Il dataset di riferimento è costituito da 150 istanze, ciascuna costituita da 5 attributi, quattro di input ed uno di output (come mostrato in figura 7.7);

utilizzando questi dati abbiamo eseguito vari algoritmi di classificazione che hanno portato a risultati differenti.

```
@relation Iris Plants Database
@attribute sepalLength real [4.3, 7.9]
@attribute sepalWidth real [2.0, 4.4]
@attribute petalLength real [1.0, 6.9]
@attribute petalWidth real [0.1, 2.5]
@attribute class {Iris-setosa, Iris-versicolor, Iris-virginica}
@inputs sepalLength, sepalWidth, petalLength, petalWidth
@outputs class
@data
5.1, 3.5, 1.4, 0.2, Iris-setosa
4.9, 3.0, 1.4, 0.2, Iris-setosa
4.7, 3.2, 1.3, 0.2, Iris-setosa
.....
.....
.....
```

Figura 7 . 7: Struttura del file Iris.arff

La distribuzione delle classi sulla base di ogni singolo attributo si ottiene facilmente dalla scheda **Preprocess**, utilizzando gli istogrammi presenti nella parte bassa della finestra.

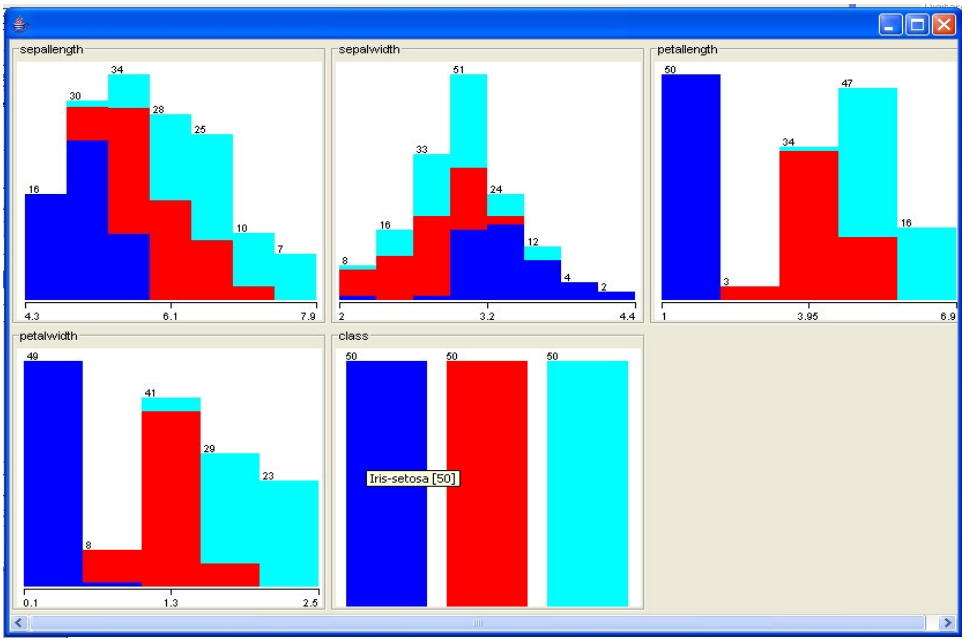

Figura 7 . 8: Distribuzione dei dati del dataset Iris

Guardando attentamente gli istogrammi, si può notare che alcuni attributi sono più discriminanti di altri che, come vedremo, non hanno un ruolo significativo nell'ambito degli algoritmi di classificazione; ad esempio, è piuttosto evidente che l'attributo petal lenght è fondamentale in quanto, a seconda dei valori che assume, i dati si distribuiscono in una classe piuttosto che in un'altra, un comportamento completamente diverso invece si ha per l'attributo sepal width. Se invece si vuole visualizzare la distribuzione delle classi in base a coppie attributi, bisogna ricorrere alla scheda **Visualize** (figura 7.9); questa scheda è importante in quanto consente di far ancora maggior chiarezza sulla distribuzione dei dati.

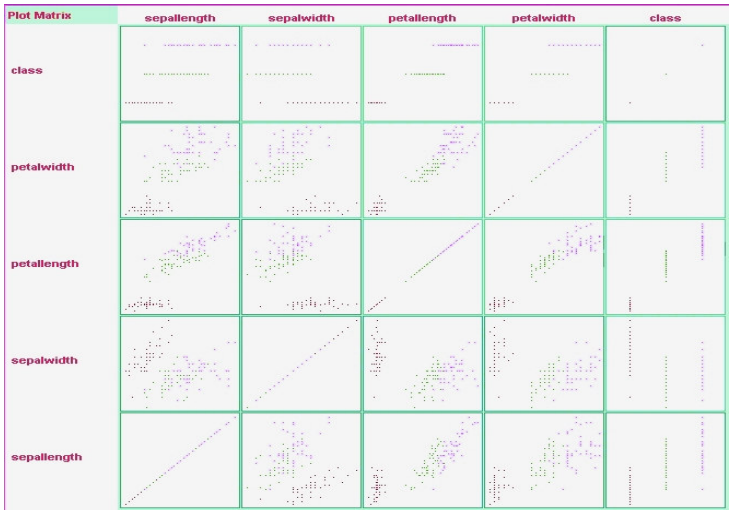

Figura 7 . 9: Scheda Visualize

All'interno della scheda Visualize è possibile selezionare quali coppie di attributi vedere con lo zoom per poter fare un'analisi più attenta; è abbastanza evidente che la riga più interessante è la prima perché consente di vedere, a seconda dei valori di un singolo attributo, come i dati sono distribuiti nelle classi.

In questo modo, ad esempio, si può intuire facilmente che l'attributo petal length è significativo perché consente una netta distribuzione dei dati mentre

l'attributo sepal length (come si vede in figura 7.10) non è significativo per la suddivisione dei dati in classi.

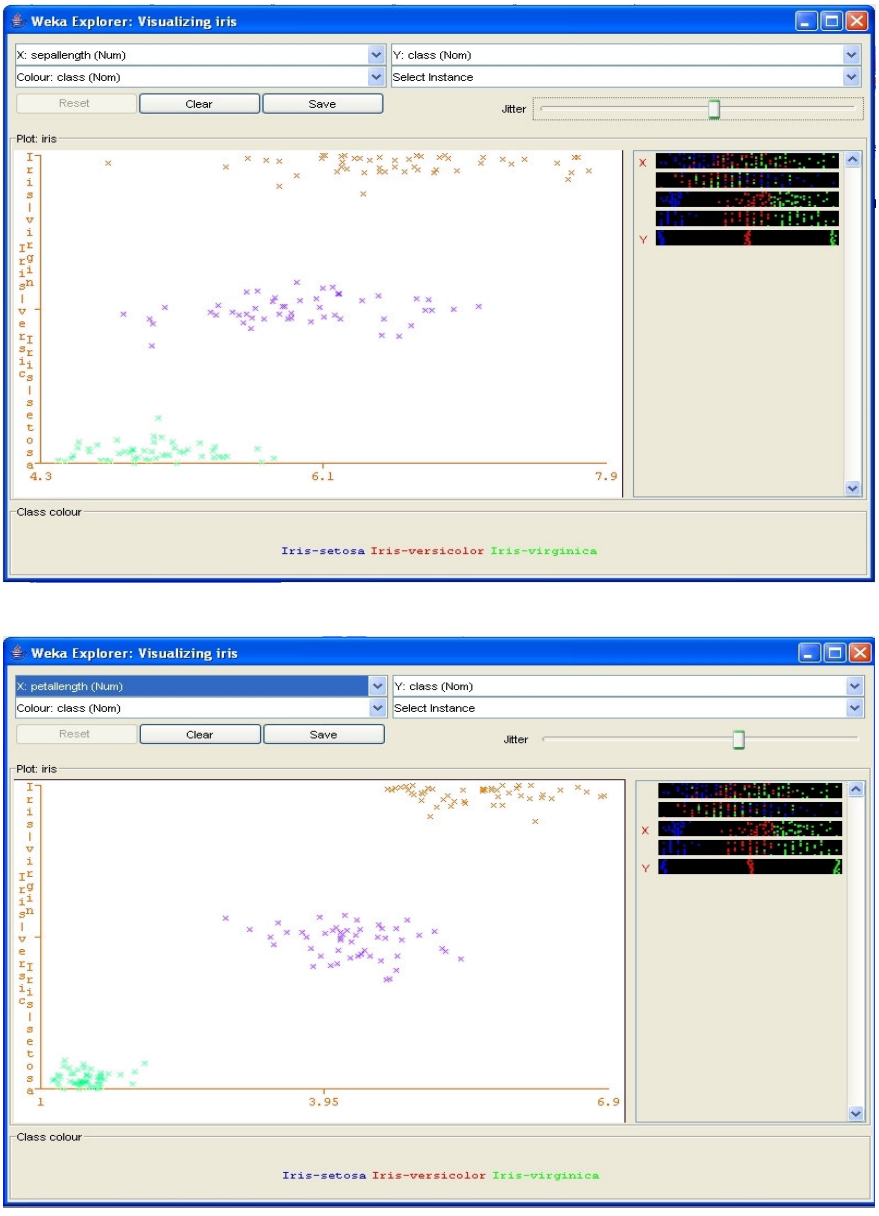

Figura 7. 10: Scheda Visualize; zoom per gli attributi sepal length e petal length.

L'insieme di questi strumenti, messi a disposizione dall'ambiente Explorer, porta quindi alla conclusione che alcuni degli attributi presenti nei nostri dati non sono caratterizzanti e quindi potrebbero anche essere eliminati senza

comportare problemi al processo di classificazione. Vedremo infatti che, se si utilizza la scheda **Select Attribute** si ottengono gli attributi "migliori", che possono da soli guidare il processo di classificazione o di clustering, consentendo di raggiungere ottimi risultati. Incominciamo con l'applicare ai dati un semplice algoritmo di classificazione, l'algoritmo **ZeroR**; esso genera un'unica regola che associa, ad ogni input, sempre la stessa classe, quella che risulta più frequente nell'insieme dei dati di addestramento. L'output di Explorer è mostrato in figura 7.11 ed è evidente che la scelta di questo algoritmo per il nostro dataset non è appropriata infatti ben il 67% delle classificazioni è sbagliata.

```
=== Run information ===

Scheme:       weka.classifiers.rules.ZeroR -D
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    evaluate on training data

=== Classifier model (full training set) ===

ZeroR predicts class value: Iris-setosa

Time taken to build model: 0 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances          50               33.3333 %
Incorrectly Classified Instances       100               66.6667 %
Kappa statistic                          0
Mean absolute error                      0.4444
Root mean squared error                  0.4714
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure   Class
```

```
 1         1        0.333    1       0.5       Iris-setosa
 0         0        0        0       0         Iris-versicolor
 0         0        0        0       0         Iris-virginica

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
 50  0  0 |  b = Iris-versicolor
 50  0  0 |  c = Iris-virginica
```

Figura 7. 11: Output dell'algoritmo ZeroR , testing sul training set

La situazione peggiora ancora se si utilizza come test option la suddivisione del dataset in dati per il training(80%) e dati per il test(20%) (percentage split) infatti si ottengono i seguenti risultati:

```
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances           9               30      %
Incorrectly Classified Instances        21               70      %
=== Confusion Matrix ===

  a  b  c   <-- classified as
  0  0 11 |  a = Iris-setosa
  0  0 10 |  b = Iris-versicolor
  0  0  9 |  c = Iris-virginica
```

Figura 7. 12: Output dell'algoritmo ZeroR , testing sul 20% del dataset

Nell'output sono presenti anche dei parametri importanti come:

- **True positive (TP) rate**: numero di esempi appartenenti ad una classe, correttamente classificati, rispetto al numero totale di esempi appartenenti a quella classe.
- **False positive (FP) rate**: numero di esempi classificati erroneamente come appartenenti ad una classe rispetto al numero totale di esempi appartenenti a quella classe.

- **Precision:** numero di esempi di classe c classificati come c, rispetto al numero totale di esempi classificati come c.
- **Recall:** identica alla TP rate.
- **F-measure:** essa è una misura di efficienza che utilizza sia precision che recall.

I valori di questi parametri e la matrice di confusione ci indicano non solo che sono presenti errori ma anche dove l'algoritmo sbaglia. In figura 7.12 si vede bene che tutte le istanze appartenenti alle classi Iris-versicolor e Iris-setosa sono erroneamente classificate.

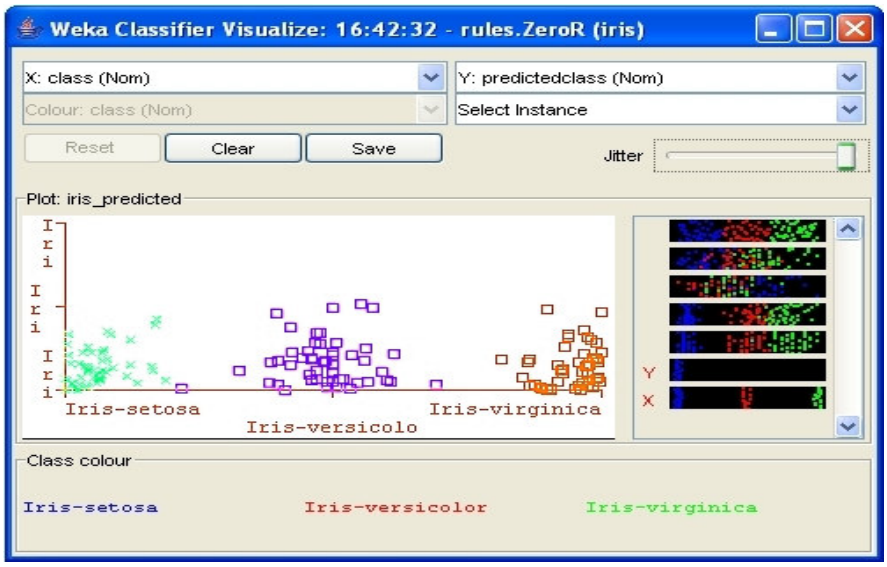

Figura 7. 13: Classifier Visualize per l'algoritmo ZeroR

La seconda prova è stata fatta utilizzando l'algoritmo di classificazione **OneR,** esso genera un albero di decisione di profondità uno, sulla base dell'attributo che minimizza il tasso d'errore (vengono inoltre discretizzati gli attributi numerici); questo algoritmo ha dato buoni risultati infatti la maggior parte degli esempi vengono ben classificati.

```
=== Run information ===
Scheme:       weka.classifiers.rules.OneR -B 6
Relation:     iris
```

```
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    evaluate on training data

=== Classifier model (full training set) ===

petallength:
      < 2.45      -> Iris-setosa
      < 4.75      -> Iris-versicolor
      >= 4.75     -> Iris-virginica
(143/150 instances correct)


Time taken to build model: 0 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances         143               95.3333 %
Incorrectly Classified Instances         7                4.6667 %
Kappa statistic                          0.93
Mean absolute error                      0.0311
Root mean squared error                  0.1764
Relative absolute error                  7      %
Root relative squared error             37.4166 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===
TP Rate   FP Rate   Precision   Recall  F-Measure   Class
1         0          1         1         1        Iris-setosa
0.88      0.01       0.978     0.88      0.926    Iris-versicolor
0.98      0.06       0.891     0.98      0.933    Iris-virginica

=== Confusion Matrix ===
  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 44  6 |  b = Iris-versicolor
  0  1 49 |  c = Iris-virginica
```

Figura 7. 14: Output dell'algoritmo OneR

L'algoritmo trova che l'attributo che minimizza il tasso d'errore è petal lenght infatti esso consente di ottenere il 95% delle classificazioni esatte e anche facendo una cross validation i risultati sono ottimi infatti si ottiene un tasso d'errore del 6%, quindi di un solo punto percentuale più grande rispetto al caso precedente anche se aumentano alcuni parametri statistici come la

perdita quadratica media. La figura 7.14 mette bene in evidenza il miglioramento rispetto all'algoritmo ZeroR infatti è bassissimo il numero di quadratini (che rappresentano le classificazioni sbagliate).

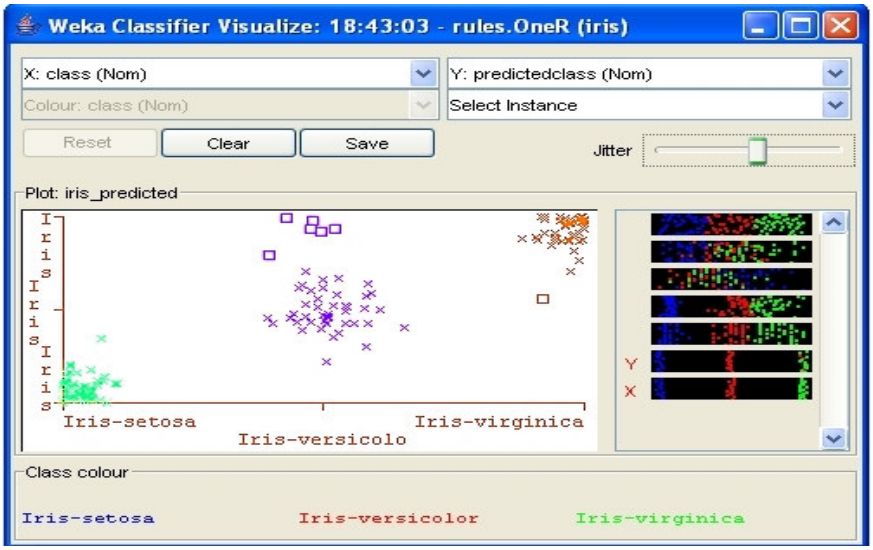

Figura 7. 15: Classifier Visualize per l'algoritmo OneR

Un ottimo risultato si ottiene anche utilizzando come insieme di dati di training, solo il 66% del dataset, cioè specificando percentage split al 66% come test option; con questa scelta il 34% del dataset viene utilizzato per il testing mentre la parte rimanente è stata precedentemente utilizzata per addestrare il classificatore.

```
Test mode:    split 66% train, remainder test
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances          49               96.0784 %
Incorrectly Classified Instances         2                3.9216 %
=== Confusion Matrix ===

  a  b  c   <-- classified as
 15  0  0 |  a = Iris-setosa
  0 19  0 |  b = Iris-versicolor
 0  2 15 |  c = Iris-virginica
```

Figura 7. 16: Output dell'algoritmo OneR con percentage split

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         141
94  %
Incorrectly Classified Instances         9
6%
Kappa statistic                          0.91
Mean absolute error                      0.04
Root mean squared error                  0.2
Relative absolute error                  9      %
Root relative squared error             42.4264 %
Total Number of Instances              150
=== Confusion Matrix ===
  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 45  5 |  b = Iris-versicolor
  0  4 46 |  c = Iris-virginica
```

Figura 7. 17: Output dell'algoritmo OneR con cross validation (K=10)

La terza prova è stata fatta utilizzando un algoritmo per la costruzione di alberi decisionali, simile a **ID3,** che non può essere impiegato nel nostro caso in quanto esso lavora solo con attributi nominali. L'algoritmo impiegato è **J48**, esso **consente di generare un albero di tipo C4.5** sia con pruning che senza pruning, e come si vede nella seguenti figure mostra risultati leggermente migliori a quelli ottenuti con l'algoritmo OneR. Prima di lanciare l'algoritmo bisogna impostare correttamente alcuni parametri come ad esempio il numero minimo di istanze che devono appertenere a ciascuna foglia dell'albero, il fattore di confidenza usato per il pruning (valori bassi comportano un pruning maggiore dei rami dell'albero) o anche se si vuole ridurre l'errore di pruning. L'output dell'algoritmo e l'albero che viene generato sono mostrati nelle figure che seguono.

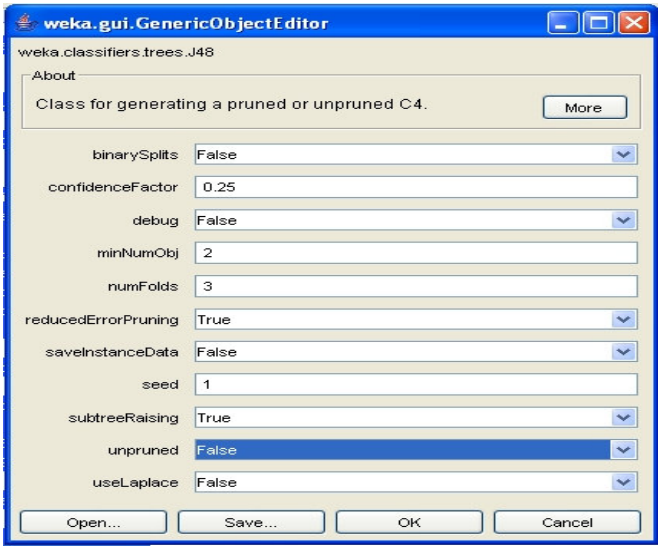

Figura 7. 18: Finestra di impostazione dell'algoritmo J48

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -R -N 3 -Q 1 -M 2
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    evaluate on training data

=== Classifier model (full training set) ===
J48 pruned tree
------------------

petalwidth <= 0.6: Iris-setosa (34.0)
petalwidth > 0.6
|   petalwidth <= 1.5: Iris-versicolor (32.0/1.0)
|   petalwidth > 1.5
|   |   petallength <= 4.8
|   |   |   sepalwidth <= 3.1: Iris-virginica (3.0)
|   |   |   sepalwidth > 3.1: Iris-versicolor (2.0)
|   |   petallength > 4.8: Iris-virginica (29.0)

Number of Leaves  :     5

Size of the tree :      9


Time taken to build model: 0 seconds
=== Evaluation on training set ===
=== Summary ===
```

```
Correctly Classified Instances    145    96.6667 %

Incorrectly Classified Instances  5       3.3333 %

Kappa statistic                          0.95
Mean absolute error                      0.0281
Root mean squared error                  0.147
Relative absolute error                  6.3125 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  1         0          1         1         1        Iris-setosa
  0.96      0.03       0.941     0.96      0.95     Iris-versicolor
  0.94      0.02       0.959     0.94      0.949    Iris-virginica

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 48  2 |  b = Iris-versicolor
  0  3 47 |  c = Iris-virginica
```

Figura 7. 19: Output dell'algoritmo J48, valutazione sul training set

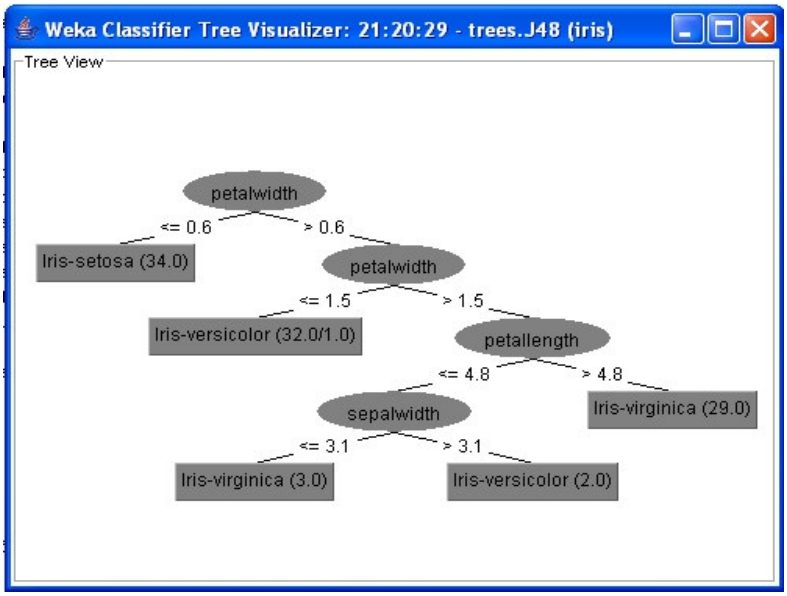

Figura 7. 20: Albero generato dall'algoritmo J48

Anche la validazione dei risultati fatta con l'opzione percentage split (70% dei dati come training) ha mostrato ottimi risultati infatti il tasso di errore è pari al 4%.

```
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances       43        95.5556 %
Incorrectly Classified Instances      2         4.4444 %
=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision      Class
  1         0          1         Iris-setosa
  1         0.069      0.889     Iris-versicolor
  0.867     0          1
=== Confusion Matrix ===

  a  b  c   <-- classified as
 14  0  0 |  a = Iris-setosa
  0 16  0 |  b = Iris-versicolor
  0  2 13 |  c = Iris-virginica
```

Figura 7. 21: Output dell'algoritmo J48 con percentage split

La figura 7.20 mostra chiaramente, in linea anche con i risultati degli altri algoritmi, che gli attributi che sono determinanti per il processo di classificazione sono due: petal length e petal width. La scheda Select Attribute consente di scegliere tra vari tipi di criteri di ricerca, come quella esaustiva, best-first o genetica ed altri ancora, e tra i vari metodi di selezione degli attributi quali il guadagno di informazione, il gainratio, le componenti principali etc. Utilizzando sia una ricerca esaustiva che il metodo best-first si ottiene lo stesso output; si giunge cioè alla stessa conclusione che identifica gli attributi relativi al petalo come fondamentali.

```
=== Attribute Selection on all input data ===

Search Method:
      Exhaustive Search.
      Start set: no attributes
      Number of evaluations: 32
      Merit of best subset found:    0.887
```

```
Attribute Subset Evaluator (supervised, Class (nominal): 5 class):
      CFS Subset Evaluator

Selected attributes: 3,4 : 2
                     petallength
                   petalwidth
```

Figura 7. 22: Output di Select Attribute

L'ultima tecnica di data mining che è stata provata sul dataset iris.arff è di tipo **non supervisionato**, infatti si è realizzato un clustering a partizioni, per realizzarlo, ovviamente, si è dovuto rimuovere l'attributo di output che specifica la classe di appartenenza supponendo di non conoscerla e cercando di capire in che modo si distribuiscono i dati.

```
== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -N 2 -S 10
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
Ignored:
              class
Test mode:    Classes to clusters evaluation on training data

=== Clustering model (full training set) ===
kMeans
======

Number of iterations: 2
Within cluster sum of squared errors: 12.143688281579719

Cluster centroids:

Cluster 0
      Mean/Mode:  6.262  2.872  4.906  1.676
      Std Devs:   0.6628 0.3328 0.8256 0.4248
Cluster 1
      Mean/Mode:  5.006  3.418  1.464  0.244
      Std Devs:   0.3525 0.381  0.1735 0.1072


=== Evaluation on training set ===
kMeans
======

Number of iterations: 2
```

```
Within cluster sum of squared errors: 24.287376563159434

Cluster centroids:

Cluster 0
      Mean/Mode:  6.262  2.872  4.906  1.676
      Std Devs:   0.6628 0.3328 0.8256 0.4248
Cluster 1
      Mean/Mode:  5.006  3.418  1.464  0.244
      Std Devs:   0.3525 0.381  0.1735 0.1072

Clustered Instances

0      100 ( 67%)
1       50 ( 33%)

Class attribute: class
Classes to Clusters:

  0  1  <-- assigned to cluster
  0 50 | Iris-setosa
 50  0 | Iris-versicolor
 50  0 | Iris-virginica

Cluster 0 <-- Iris-versicolor
Cluster 1 <-- Iris-setosa

Incorrectly clustered instances :   50.0   33.3333 %
```

Figura 7. 23: Output di K-means nel caso K=2

L'algoritmo impiegato è quello delle K-medie; sappiamo bene che questo tipo di algoritmo richiede in input il numero di cluster che noi ci aspettiamo di trovare nei dati e infatti se si inserisce, supponendo di non conoscere la distribuzione dei dati, due invece di tre l'algoritmo sbaglia totalmente e quindi può portare a conclusioni errate infatti presenta un tasso di errore elevato, sbaglia infatti nel 33% dei casi, come si vede anche in figura 7.23.

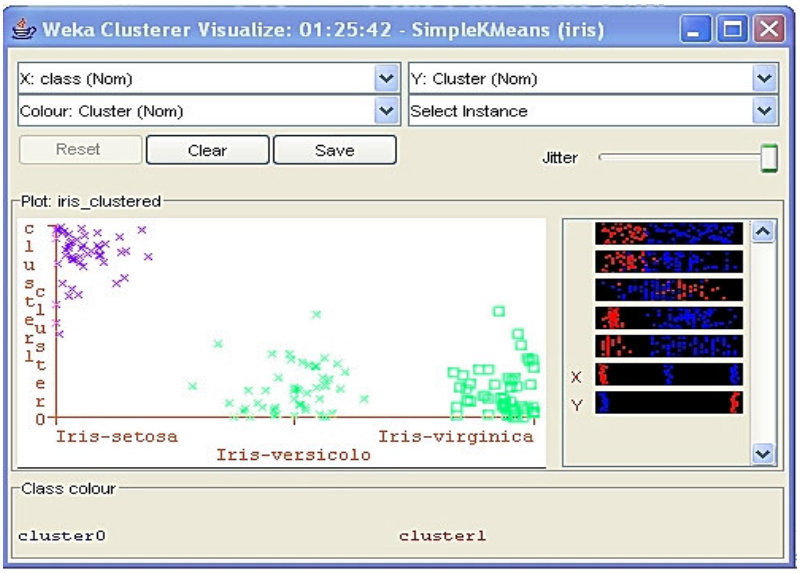

Figura 7. 24: Classifier Visualize per l'algoritmo K-means nel caso K=2

Ripetendo invece l'algoritmo, impostando a tre il numero di cluster, si ottengono ottimi risultati infatti il tasso di errore diventa12% e diminuisce l'errore quadratico anche se aumenta il numero di iterazioni necessarie affinché l'algoritmo converga.

```
=== Run information ===
Scheme:       weka.clusterers.SimpleKMeans -N 3 -S 10
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
Ignored:
              class
Test mode:    Classes to clusters evaluation on training data

=== Clustering model (full training set) ===
kMeans

Number of iterations: 10
Within cluster sum of squared errors: 7.138647703985378

Cluster centroids:

Cluster 0
      Mean/Mode:  6.7038 3.0635 5.4673 1.9769
      Std Devs:   0.5217 0.273  0.6164 0.3135
```

```
Cluster 1
      Mean/Mode:  5.006  3.418  1.464  0.244
      Std Devs:   0.3525 0.381  0.1735 0.1072
Cluster 2
      Mean/Mode:  5.7833 2.6646 4.2979 1.35
      Std Devs:   0.4244 0.2605 0.5452 0.2543

=== Evaluation on training set ===
kMeans

Number of iterations: 10
Within cluster sum of squared errors: 14.277295407970755
Cluster centroids:

Cluster 0
      Mean/Mode:  6.7038 3.0635 5.4673 1.9769
      Std Devs:   0.5217 0.273  0.6164 0.3135
Cluster 1
      Mean/Mode:  5.006  3.418  1.464  0.244
      Std Devs:   0.3525 0.381  0.1735 0.1072
Cluster 2
      Mean/Mode:  5.7833 2.6646 4.2979 1.35
      Std Devs:   0.4244 0.2605 0.5452 0.2543

Clustered Instances

0       52 ( 35%)
1       50 ( 33%)
2       48 ( 32%)

Class attribute: class
Classes to Clusters:

  0  1  2  ← assigned to cluster
  0 50  0 | Iris-setosa
 10  0 40 | Iris-versicolor
 42  0  8 | Iris-virginica

Cluster 0 ← Iris-virginica
Cluster 1 ← Iris-setosa
Cluster 2 ← Iris-versicolor

Incorrectly clustered instances :  18.0   12      %
```

Figura 7. 25: Output di K-means nel caso K=3

Il numero di iterazioni da due è salito a dieci ma in cambio i cluster che si formano rispecchiano effettivamente la distribuzione dei dati nelle tre classi anche se i risultati ottenuti con strumenti di data mining supervisionato sono migliori. Si può anche osservare che l'algoritmo non sbaglia mai con riferimento alla classe Iris-setosa, questo risultato è in linea anche con l'output degli algoritmi di DM supervisionato.

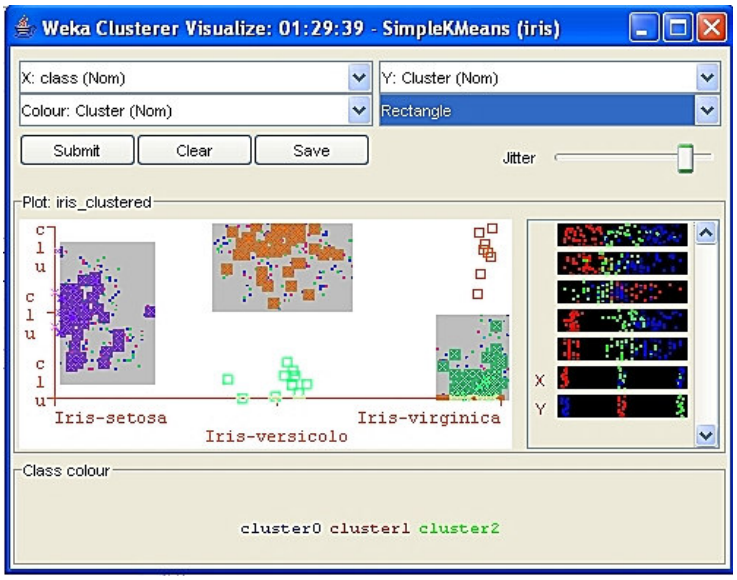

Figura 7. 26: Classifier Visualize per l'algoritmo K-means nel caso K=3

Non solo una valutazione per difetto ma anche una per eccesso del valore di K può portare a risultati completamente sbagliati infatti, mandando in esecuzione l'algoritmo con K=5, la clusterizzazione che si ottiene è visibilmente errata infatti ben il 45% delle istanze non sono assegnate correttamente.

```
Clustered Instances
0       30 ( 20%)
1       27 ( 18%)
2       27 ( 18%)
3       43 ( 29%)
4       23 ( 15%)
Class attribute: class
Classes to Clusters:
  0  1  2  3  4  ← assigned to cluster
  0 27  0  0 23 | Iris-setosa
  0  0 26 24  0 | Iris-versicolor
 30  0  1 19  0 | Iris-virginica

Cluster 0 ← Iris-virginica
Cluster 1 ← Iris-setosa
Cluster 2 ← Iris-versicolor
Cluster 3 ← No class
Cluster 4 ← No class

Incorrectly clustered instances :   67.0   44.6667 %
```

Figura 7. 27: Output per l'algoritmo K-means nel caso K=5

7.5 2° caso d'uso: analisi del dataset Pima-diabetes

Il secondo dataset su cui abbiamo lavorato è di natura medica, si tratta infatti di una raccolta di dati relativi ad alcuni pazienti diabetici e non; l'obiettivo di questo studio è cercare di creare un modello, uno strumento di classificazione che consenta, sulla base di determinati parametri (che poi sono alcuni degli attributi che compaiono nella relazione diabetes) di capire se un paziente sia affetto o meno da diabete. Il nostro interesse non si limita a questo infatti si è anche cercato di individuare quali sono le cause principali di diabete e in che modo si combinano tra loro. Il diabete è una malattia metabolica cronica in cui l'organismo, non essendo in grado di produrre o utilizzare completamente l'insulina, causa un difetto del metabolismo dei carboidrati e, in misura minore, delle proteine e dei grassi, per cui i reni, per liberare l'organismo dall'eccesso di zuccheri (glicemia), cominciano a eliminarli attraverso l'urina. L'aumento della glicemia comporta danni in tutte le strutture dell'organismo, aggravando il rischio di complicanze, quali disturbi circolatori, insufficienza renale, ictus e problemi neurologici. Le cause del diabete sono molteplici e non del tutto chiarite, vi sono i fattori ereditari, genetici, ambientali, immunologici, interferenze ormonali e fenomeni acquisiti. I dati presi in esami sono stati raccolti a Phoenix in Arizona (Stati Uniti), sono relativi a sole donne, e su di essi sono state fatte delle diagnosi in base ai criteri dell'Organizzazione mondiale della Sanità[7.2]. Il dataset utilizzato è stato ridotto ai minimi termini nel senso che si sono trascurati attributi secondari e si sono considerate solo le cause principali del diabete; esso è costituito da 768 istanze, ciascuna composta da otto attributi più un attributo di output (500 istanze relative a pazienti diabetici e 268 relative a non diabetici).

```
@relation pima_diabetes
@attribute 'preg' real
@attribute 'plas' real
@attribute 'pres' real
@attribute 'skin' real
@attribute 'insu' real
@attribute 'mass' real
@attribute 'pedi' real
@attribute 'age'  real
@attribute 'class' { tested_negative, tested_positive}
```

Figura 7. 28: Intestazione del file diabetes.arff

Il significato di ciascun attributo è:

1. Number of times pregnant: numero di mese di gravidanza
2. Plasma glucose concentration a 2 hours in an oral glucose test: concentrazione di glucosio nel sangue dopo due ore dall'ingestione di zuccheri
3. Diastolic pressure blood (mm Hg): pressione sanguigna
4. Triceps skin fold thickness (mm): spessore della pelle del tricipite
5. 2-Hour serum insuline (mu U/ml): livello di insulina dopo due ore dall'assunzione di zuccheri
6. Body mass index (weight inKg/(height in m)^2): indice di massa corporea
7. Diabetes pedigree function: funzione genealogica del diabete
8. Age (years): età
9. Class variable (0 o 1), assume il significato di test positivo se pari a 1° di test negativo altrimenti

La concentrazione di glucosio nel sangue viene misurata dopo due ore per verificare se il pancreas sta svolgendo correttamente la sua funzione infatti,dopo carico orale di glucosio, nei soggetti normali l'insulina ha un picco dopo 60 minuti mentre in caso di diabete latente e/o obesità il picco si ha dopo due ore. Inoltre si è notato che si registrano variazioni anomale del

livello di insulina nel caso di donne in gravidanza e che sicuramente c'è un forte fattore ereditario tra le cause del diabete. Tra gli attributi sicuramente è fondamentale la pressione sanguigna infatti è noto che i malati di diabete soffrono anche di ipertensione; sia il diabete che la pressione alta sono ambedue conseguenze del peso(che viene tenuto in conto grazie all'indice di massa corporeo) e dell'insulino-resistenza. Un altro attributo molto importante ai fini della classificazione è anche l'età, che, a seconda della tipologia di diabete e dei valori di altri attributi può essere considerato come uno dei fattori determinanti per la diagnosi diabetica. L'analisi ha evidenziato che, sebbene precedentemente siano già stati eliminati degli attributi dal dataset, alcuni dei fattori presi in considerazione possono essere ulteriormente trascurati nel processo di classificazione.

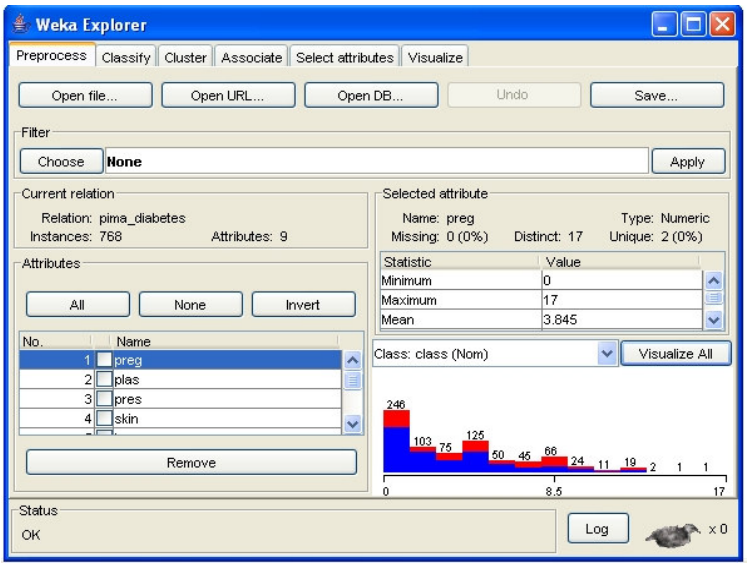

Figura 7. 29: Loading del file diabetes.arff

Sebbene possa sembrare inutile, la selezione ed il filtraggio degli attributi è un aspetto importante in quanto al crescere della mole di dati, l'elaborazione diventa più pesante (anche se il software è ottimizzato e lavora bene anche su dataset molto grandi) e quindi può essere utile eliminare qualche attributo ai fini anche della computazione. La distribuzione dei dati nelle due classi in

funzione dei valori assunti dai singoli attributi è mostrata in figura 7.27 e in questo caso non ci aiuta a capire quali attributi siano più discriminanti.

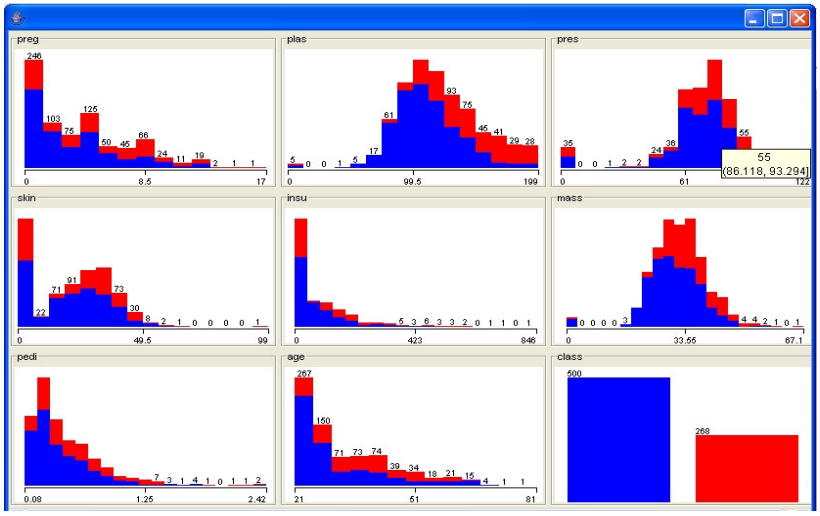

Figura 7. 30: Distribuzione dei dati del □perati diabetes

La difficoltà è anche legata all'aumento del numero di attributi, rispetto all'esempio precedente, per cui diventa più complicato fare un'analisi a monte, senza utilizzare uno degli algoritmi di selezione degli attributi. Il primo tentativo di classificazione è stato fatto impiegando l'algoritmo OneR.

```
=== Run information ===
Scheme:       weka.classifiers.rules.OneR -B 6
Relation :     pima_diabetes
Instances :    768
Attributes :   9
               preg
               plas
               pres
               skin
               insu
               mass
               pedi
               age
               class
Test mode:    evaluate on training data

=== Classifier model (full training set) ===
plas:
      < 114.5     -> tested_negative
      < 115.5     -> tested_positive
      < 133.5     -> tested_negative
```

```
      < 135.5     -> tested_positive
      < 144.5     -> tested_negative
      < 152.5     -> tested_positive
      < 154.5     -> tested_negative
      >= 154.5    -> tested_positive
(585/768 instances correct)

Time taken to build model: 0.02 seconds

=== Evaluation on training set ===
== Summary ===
Correctly Classified Instances         585         76.1719 %
Incorrectly Classified Instances       183         23.8281 %
Kappa statistic                          0.4329
Mean absolute error                      0.2383
Root mean squared error                  0.4881
Total Number of Instances              768

=== Detailed Accuracy By Class ===
TP Rate   FP Rate   Precision   Class
 0.904     0.504      0.77      tested_negative
 0.496     0.096      0.735     tested_positive
=== Confusion Matrix ===

   a   b   ← classified as
 452  48 |   a = tested_negative
 135 133 |   b = tested_positive
```

Figura 7. 31: Output dell'algoritmo OneR per diabetes.arff

Come si vede dall'output dell'algoritmo, il tasso di errore valutato sull'intero dataset di training è abbastanza alto infatti OneR sbaglia nel 24% dei casi per cui la precisione dell'algoritmo non è sufficientemente alta. Le righe della matrice di confusione corrispondono alla classe corretta mentre le colonne alla classe predetta. Sull'intero training set 183 istanze vengono erroneamente classificate, di queste 48 sono falsi positivi mentre l'aspetto peggiore è che ben 135 istanze vengono erroneamente classificate negative al test e quindi si ha un numero troppo grande di falsi negativi che in ambito medico, dove è in gioco la salute delle persone, non sono tollerabili. .Una doppia prova con test percentage split impostato prima al 70% e poi 80% ha mostrato rispettivamente un tasso di errore del 25% e del 27%. La situazione peggiora se si prova a fare una cross validation raggiungendo un tasso d'errore del 29%.

```
=== Stratified cross-validation ===
Correctly Classified Instances         555     72.2656 %
Incorrectly Classified Instances       213     27.7344 %
Total Number of Instances              768
=== Confusion Matrix ===

   a   b   ← classified as
 434  66 |   a = tested_negative
 147 121 |   b = tested_positive

Test mode:    split 70% train, remainder test
== Evaluation on test split ===

Correctly Classified Instances         172     74.4589 %
Incorrectly Classified Instances        59     25.5411 %
Total Number of Instances              231
=== Confusion Matrix ===

   a   b   ← classified as
 136  22 |   a = tested_negative
  37  36 |   b = tested_positive

Test mode:    split 80% train, remainder test
=== Evaluation on test split ===

Correctly Classified Instances         113     73.3766 %
Incorrectly Classified Instances        41     26.6234 %
=== Confusion Matrix ===

  a  b   ← classified as
 92 13 |  a = tested_negative
 28 21 |  b = tested_positive

Test mode:    split 90% train, remainder test
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances          57     74.026  %
Incorrectly Classified Instances        20     25.974  %
=== Confusion Matrix ===

  a  b   ← classified as
 45  7 |  a = tested_negative
 13 12 |  b = tested_positive
```

Figura 7. 32: Output dell'algoritmo OneR con test cross validation e percentage split

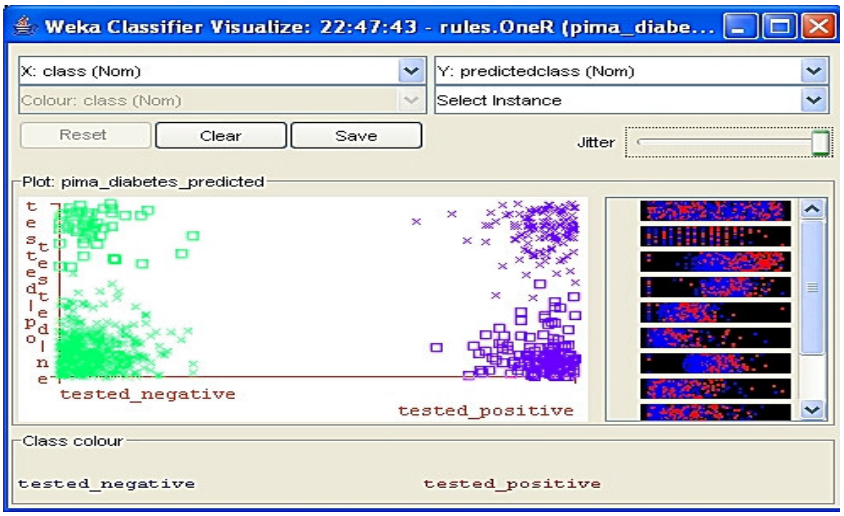

Figura 7. 33: Classifier Visualize per l'algoritmo OneR

L'idea allora è quella di sforzarsi di trovare qualcosa che consenta di ridurre al minimo i falsi negativi che sono molto pericolosi per l'esempio che stiamo considerando. Si è pensato di ricorrere agli alberi decisionali e il primo algoritmo che è stato provato è J48 con cui i risultati sono stati i seguenti:

```
== Run information ===
Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation :     pima_diabetes
Instances :    768
Attributes :   9
               preg
               plas
               pres
               skin
               insu
               mass
               pedi
               age
               class
Test mode:    evaluate on training data
=== Classifier model (full training set) ===
J48 pruned tree


plas <= 127
|   mass <= 26.4: tested_negative (132.0/3.0)
|   mass > 26.4
|   |   age <= 28: tested_negative (180.0/22.0)
|   |   age > 28
|   |   |   plas <= 99: tested_negative (55.0/10.0)
|   |   |   plas > 99
```

```
|   |   |   |   pedi <= 0.561: tested_negative (84.0/34.0)
|   |   |   |   pedi > 0.561
|   |   |   |   |   preg <= 6
|   |   |   |   |   |   age <= 30: tested_positive (4.0)
|   |   |   |   |   |   age > 30
|   |   |   |   |   |   |   age <= 34: tested_negative (7.0/1.0)
|   |   |   |   |   |   |   age > 34
|   |   |   |   |   |   |   |   mass <= 33.1: tested_positive
(6.0)
|   |   |   |   |   |   |   |   mass > 33.1: tested_negative
(4.0/1.0)
|   |   |   |   |   preg > 6: tested_positive (13.0)
plas > 127
|   mass <= 29.9
|   |   plas <= 145: tested_negative (41.0/6.0)
|   |   plas > 145
|   |   |   age <= 25: tested_negative (4.0)
|   |   |   age > 25
|   |   |   |   age <= 61
|   |   |   |   |   mass <= 27.1: tested_positive (12.0/1.0)
|   |   |   |   |   mass > 27.1
|   |   |   |   |   |   pres <= 82
|   |   |   |   |   |   |   pedi <= 0.396: tested_positive
(8.0/1.0)
|   |   |   |   |   |   |   pedi > 0.396: tested_negative (3.0)
|   |   |   |   |   |   pres > 82: tested_negative (4.0)
|   |   |   |   age > 61: tested_negative (4.0)
|   mass > 29.9
|   |   plas <= 157
|   |   |   pres <= 61: tested_positive (15.0/1.0)
|   |   |   pres > 61
|   |   |   |   age <= 30: tested_negative (40.0/13.0)
|   |   |   |   age > 30: tested_positive (60.0/17.0)
|   |   plas > 157: tested_positive (92.0/12.0)

Number of Leaves  :     20
Size of the tree :      39

Time taken to build model: 0.08 seconds
=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances         646               84.1146 %
Incorrectly Classified Instances       122               15.8854 %
Kappa statistic                          0.6319
Mean absolute error                      0.2383
Root mean squared error                  0.3452
Relative absolute error                 52.4339 %
Root relative squared error             72.4207 %
Total Number of Instances              768

=== Detailed Accuracy By Class ===
TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.936     0.336      0.839     0.936     0.885
tested_negative
  0.664     0.064      0.848     0.664     0.745
tested_positive
```

```
=== Confusion Matrix ===

   a   b   ← classified as
 468  32 |   a = tested_negative
  90 178 |   b = tested_positive
```

Figura 7. 34: Output dell'algoritmo J48 per diabetes.arff

L'output del programma mostra che la situazione è migliorata infatti il tasso d'errore è sceso al 16%, l'albero generato non è molto profondo ed è costituito da ben venti foglie e consente facilmente di generare regole produttive. Utilizzando l'opzione percentage split la situazione peggiora un poiché in quanto il tasso di errore sul test set (30% del dataset) diventa 23% ma migliora rispetto all'algoritmo precedente. Le eventuali oscillazioni del tasso d'errore possono essere indicative di dati rumorosi.

```
Scheme:       weka.classifiers.trees.J48 -S -C 0.25 -M 2

Test mode:    split 70% train, remainder test

=== Evaluation on test split ===
=== Summary ===
Correctly Classified Instances         177     76.6234 %
Incorrectly Classified Instances        54     23.3766 %

=== Confusion Matrix ===

   a   b   ← classified as
 122  36 |   a = tested_negative
  18  55 |   b = tested_positive

Test mode:    split 90% train, remainder test

=== Evaluation on test split ===
=== Summary ===
Correctly Classified Instances          58      75.3247 %
Incorrectly Classified Instances        19      24.6753 %

=== Confusion Matrix ===

  a  b   ← classified as
 43  9 |  a = tested_negative
 10 15 |  b = tested_positive

=== Stratified cross-validation K=10 ===

=== Summary ===
Correctly Classified Instances         567      73.8281 %
Incorrectly Classified Instances       201      26.1719 %
```

```
=== Confusion Matrix ===

   a   b   ← classified as
 408  92 |   a = tested_negative
 109 159 |   b = tested_positive
```

Figura 7. 35: Output dell'algoritmo J48, opzione percentage split e cross validation, per diabetes.arff

L'algoritmo J48 è stato lanciato specificando di non effettuare il pruning dell'albero, se invece si vogliono tagliare dei rami basta eseguire di nuovo l'algoritmo specificandolo; in questo modo si otterrà un albero che sbaglia di più in relazione ai dati di training (figura 7.37) ma che consente generalizzazioni maggiori senza essere troppo legato ai dati di addestramento. L'albero che si ottiene in questo caso è sicuramente meno profondo, perché alcuni rami sono stati "potati" e le regole produttive sono più semplici; questo se da un lato aumenta la leggibilità , comporta una minore accuratezza delle regole, nel senso che esse non prenderanno in considerazione alcuni fattori come quelli ereditari che sappiamo essere uno della cause principali di diabete (le statistiche dicono che ben il 25% dei malati di diabete risentono di una componente ereditaria), e i criteri di test risultanti saranno troppo semplicistici.

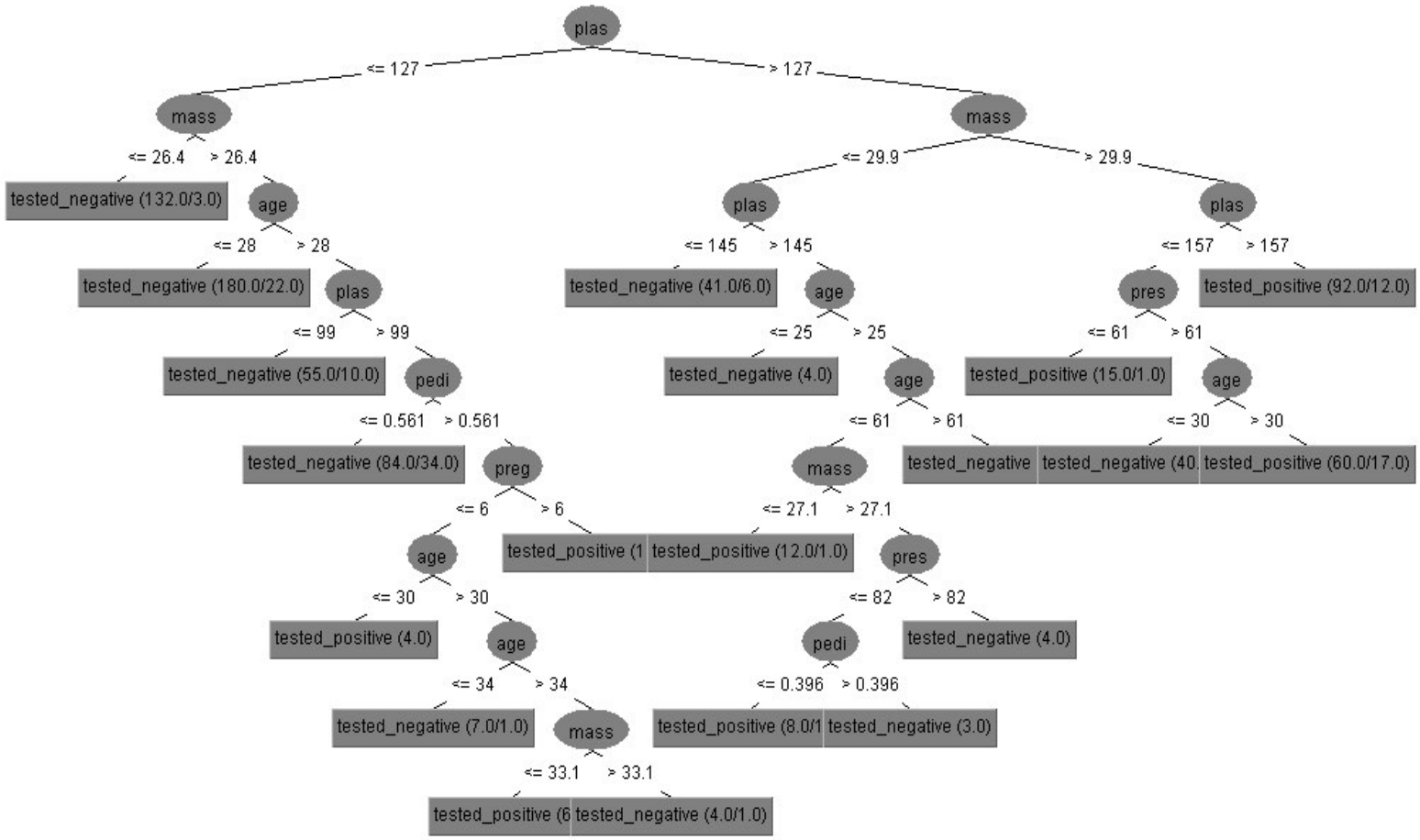

Figura 7. 36: Albero generato dall'algoritmo J48, senza pruning

Un esempio di regola produttiva che può essere generata dall'albero in figura 7.36 è la seguente:

If *((99<plas≤127) and (mass>26) and (age>28) and (pedi>0,56) and (preg>6)*

Then *tested-positive*

L'albero che invece si ottiene imponendo il pruning è mostrato in figura 7.38 e si vede subito che è molto più semplice; i risultati ottenuti mostrano un leggero peggioramento sul training set ma non si registrano grandi variazioni sia su un test percentage split che cross validation.

```
=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances         597      77.7344 %
Incorrectly Classified Instances       171      22.2656 %
=== Confusion Matrix ===

   a   b   ← classified as
 390 110 |   a = tested_negative
  61 207 |   b = tested_positive
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         170      73.5931 %
Incorrectly Classified Instances        61      26.4069 %
=== Confusion Matrix ===

   a   b   ← classified as
 127  31 |   a = tested_negative
  30  43 |   b = tested_positive
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         576      75 %
Incorrectly Classified Instances       192      25 %

=== Confusion Matrix ===

   a   b   ← classified as
 426  74 |   a = tested_negative
 118 150 |   b = tested_positive
```

Figura 7. 37: Output di J48 con pruning sul training set, sul test set e cross validation

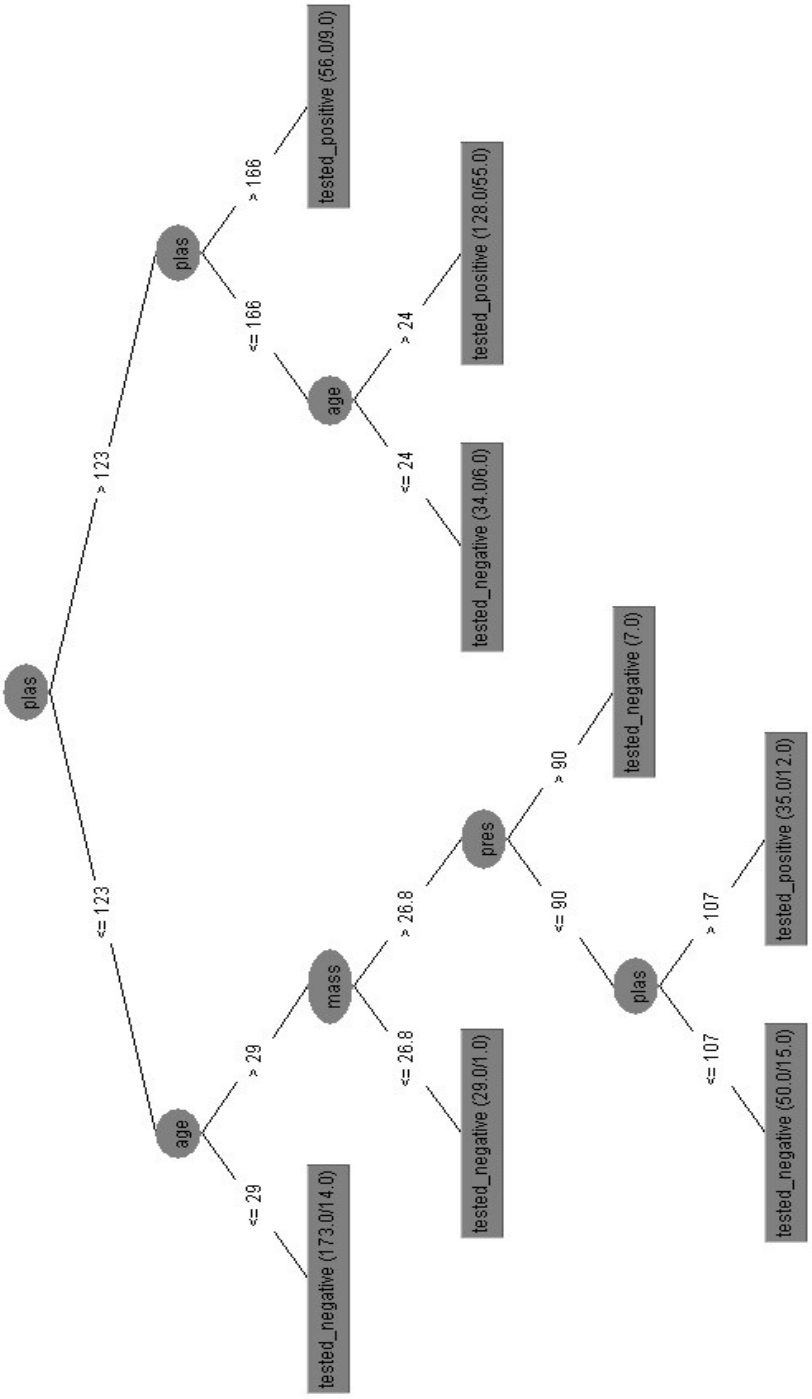

Figura 7. 38: Albero generato dall'algoritmo J48, con pruning

Sul dataset è possibile anche testare un algoritmo ID3 ma poiché esso lavora solo con attributi nominali è necessario applicare un filtro ai dati che consenta la conversione cioè la trasformazione degli attributi da numerici a nominali; abbiamo imposto una suddivisione in tre parti degli intervalli numerici.

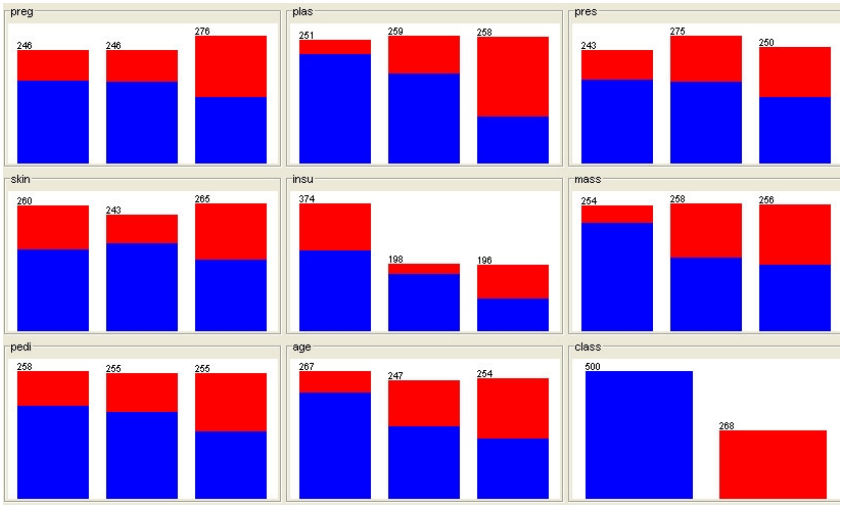

Figura 7. 39: Effetto della discretizzazione sui dati

A questo punto è stato possibile mandare in esecuzione l'algoritmo ID3 che ha presentato risultati molto simili a quelli ottenuti conJ48.

```
Scheme:       weka.classifiers.trees.Id3
Relation:     pima_diabetes-weka.filters.unsupervised.attribute.
Discretize-F-B3-M-1.0-Rfirst-1,2,3,4,5,6,7,8,last
=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances         727         94.6615 %
Incorrectly Classified Instances        41          5.3385 %
Total Number of Instances              768

=== Confusion Matrix ===

   a   b   ← classified as
 491   9 |   a = tested_negative
  32 236 |   b = tested_positive
Test mode:    split 70% train, remainder test
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         148          64.0693 %
Incorrectly Classified Instances        60          25.974  %
```

```
=== Confusion Matrix ===
   a   b   ← classified as
 111  34 |   a = tested_negative
  26  37 |   b = tested_positive
Test mode:    split 80% train, remainder test
=== Evaluation on test split ===
=== Summary ===
Correctly Classified Instances         102          66.2338 %
Incorrectly Classified Instances        46          29.8701 %
=== Confusion Matrix ===
  a  b   ← classified as
 77 23 |  a = tested_negative
 23 25 |  b = tested_positive
Test mode:    10-fold cross-validation
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances         504          65.625  %
Incorrectly Classified Instances       215          27.9948 %
=== Confusion Matrix ===
   a   b   ← classified as
 367  99 |   a = tested_negative
 116 137 |   b = tested_positive
```

Figura 7. 40: Output dell'algoritmo Id3 per diabetes.arff

Un altro algoritmo che è stato testato è il RandomTree, esso costruisce un albero considerando K attributi scelti casualmente ad ogni nodo (abbiamo impostato K=3) e sul training set ha mostrato ottimi risultati infatti il tasso di errore si è ridotto al 2% ma l'albero generato è eccessivamente profondo e quindi non consente facili generalizzazioni o comunque di generare regole di utilità pratica.

```
=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances         754      98.1771 %
Incorrectly Classified Instances        14       1.8229 %
=== Detailed Accuracy By Class ===
TP Rate   FP Rate   Precision   Class
  0.992     0.037      0.98      tested_negative
  0.963     0.008      0.985     tested_positive
=== Confusion Matrix ===
   a   b   ← classified as
 496   4 |   a = tested_negative
  10 258 |   b = tested_positive
```

Figura 7. 41: Output dell'algoritmo Randomtree per diabetes.arff

Utilizzando inoltre la cross validation o l'opzione percentage split cresce molto il tasso di errore che nel primo caso raggiunge addirittura il 36% infatti come illustrato in figura 7.34 l'algoritmo sbaglia molto e per entrambe le classi.

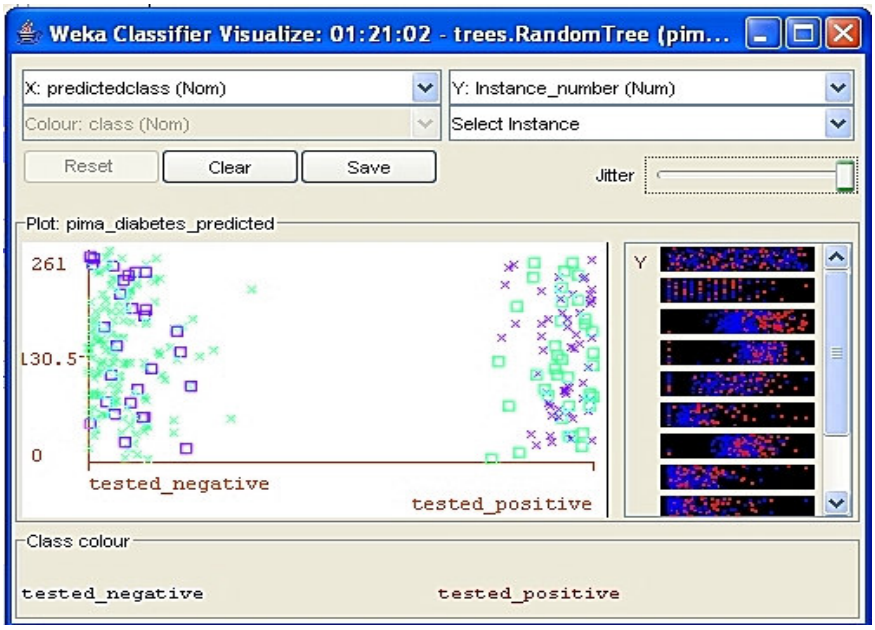

Figura 7. 42: Visualizzazione degli errori di Random Tree (cross validation)

Abbiamo poi utilizzato la scheda Select Attribute al fine di individuare gli attributi più discriminanti all'interno di quelli forniti nel dataset a disposizione e per poi mandare in esecuzione nuovamente uno degli algoritmi già provati ma su un dataset con un numero ridotto di attributi. Lo scopo è quello di paragonare i risultati e vedere se sono molto differenti tra loro. La tipologia di valutazione scelta per gli attributi è CfsSubsetEval; essa valuta la capacità di un sottoinsieme di attributi considerando "l'abilità" predittiva di ogni attributo unitamente al loro grado di ridondanza (ad esempio consente di capire se l'aggiunta di un altro attributo non aggiunge informazione). I sottoinsiemi di attributi aventi un'elevata correlazione con la classe di appartenenza e una bassa intercorrelazione sono quelli che saranno selezionati. Per quanto riguardai criteri di ricerca si è utilizzato un metodo esaustivo e un metodo di ricerca random (realizza una ricerca random tra i sottinsiemi possibili di

attributi partendo dallo start set che viene specificato in ingresso e ricercando un sottoinsieme migliore di quello di input, costituito da un eguale o minor numero di attributi; tra le specifiche di ingresso c'è anche il search space che rappresenta la percentuale dello spazio di ricerca in cui si vuole che l'algoritmo indaghi). I risultati ottenuti sono i seguenti:

```
 === Run information ===

Evaluator: weka.attributeSelection.CfsSubsetEval
Search:    weka.attributeSelection.ExhaustiveSearch
Search Method:
      Exhaustive Search.
      Start set: no attributes
      Number of evaluations: 11776

Attribute Subset Evaluator(supervised,Class(nominal): 9 class):
      CFS Subset Evaluator

Selected attributes: 2,6,8 : 3
                     plas
                     mass
                     age

=== Run information ===

Evaluator:weka.attributeSelection.CfsSubsetEval
Search:   weka.attributeSelection.RandomSearch-P 1,2,3,4,5,6,7,8-
F1.0
=== Attribute Selection on all input data ===

Search Method:
  Random search.
  Start set: 1,2,3,4,5,6,7,8
  Number of iterations: 256 (100.0% of the search space)

  Attribute Subset Evaluator(supervised,Class(nominal): 9 class):
      CFS Subset Evaluator

Selected attributes: 2,5,6,8 : 4
                     plas
                     insu
                     mass
                     age
```

Figura 7. 43: Output degli algoritmi di selezione degli attributi

L'output degli algoritmi ha evidenziato che sono tre o al massimo quattro gli attributi più discriminanti. Proviamo ora a vedere se rimuovendo gli altri attributi, gli algoritmi precedentemente testati danno gli stessi risultati che hanno dato sull'intera relazione.

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     pima_diabetes-weka.filters.unsupervised.attribute.
Remove-R1,3-4,7
Instances:    768
Attributes:   5
              plas
              insu
              mass
              age
              class
Test mode:    10-fold cross-validation
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         571           74.349  %
Incorrectly Classified Instances       197           25.651  %
Test mode:    split 70% train, remainder test
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         178           77.0563 %
Incorrectly Classified Instances        53           22.9437 %
Test mode:    split 80% train, remainder test

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         121           78.5714 %
Incorrectly Classified Instances        33           21.4286 %
Test mode:    split 90% train, remainder test

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances          59           76.6234 %
Incorrectly Classified Instances        18           23.3766 %
```

Figura 7. 44: Output di J48 dopo rimozione degli attributi 1,3,4,7

I risultati ottenuti sono molto simili, talvolta migliori, per cui è conveniente utilizzare la relazione ridotta.

Infine, anche su dataset, è stato fatto un clustering utilizzando l'algoritmo delle K-medie, impostando K=2, i risultati ottenuti sono buoni ma non come quelli ottenuti con alcuni strumenti di DM supervisionato ad esempio il tasso d'errore, rispetto a quello ottenuto con J48, è aumentato di quasi dieci punti percentuali come mostra l'output che segue.

```
=== Run information ===
Scheme:       weka.clusterers.SimpleKMeans -N 2 -S 10
Relation:     pima_diabetes
Instances:    768
Ignored:      class
Test mode:    Classes to clusters evaluation on training data

=== Clustering model (full training set) ===

kMeans

Number of iterations: 11
Within cluster sum of squared errors: 121.2579017999101

Cluster centroids:

Cluster 0
Mean/Mode:2.08 115.32 65.99 21.81  85.01 31.77 0.47 26.77
Std Devs: 1.75  30.65 19.78 14.98 110.87  8.24 0.34  5.48
Cluster 1
Mean/Mode:7.43 132.22 75.44 17.92  69.17 32.43 0.47 46.40
Std Devs: 2.99  31.66 16.77 17.50 123.21  7.08 0.30 10.00

=== Evaluation on training set ===

kMeans

Number of iterations: 11
Within cluster sum of squared errors: 242.515803599819
Cluster centroids:
Cluster 0
Mean/Mode:2.08 115.32 65.99 21.81 85.01 31.77 0.47 26.23
Std Devs: 1.75  30.65 19.78 14.98 110.87 8.24 0.34  5.48
Cluster 1
Mean/Mode:7.43 132.22 75.44 17.92  69.17 32.43 0.47 46.40
Std Devs: 2.99  31.66 16.77 17.50 123.21  7.08 0.30 10.00

Clustered Instances

0      515 ( 67%)
1      253 ( 33%)

Class attribute: class
Classes to Clusters:

   0   1  ← assigned to cluster
```

```
 380 120 | tested_negative
 135 133 | tested_positive

Cluster 0 ← tested_negative
Cluster 1 ← tested_positive
Incorrectly clustered instances :   255.0   33.2031 %
```

Figura 7. 45: Output dell'algoritmo Kmeans con K=2

Il tasso di errore più alto è anche più tollerabile in questo caso in quanto l'idea che sta alla base del clustering è completamente diversa rispetto a quella delle tecniche supervisionate; l'obiettivo non è tanto classificare quanto fungere da strumento di ausilio per capire se i dati sono suddivisi in gruppi aventi caratteristiche comuni, ciò può essere molto utile in ambito medico dove talvolta bisogna individuare delle anomalie (ad esempio se si ha a disposizione il tracciato di un elettrocardiogramma è possibile individuare i cluster fisiologici, cioè quelli rappresentanti i battiti fisiologici, dalle morfologie patologiche, che in genere formano cluster più piccoli con forme strane).

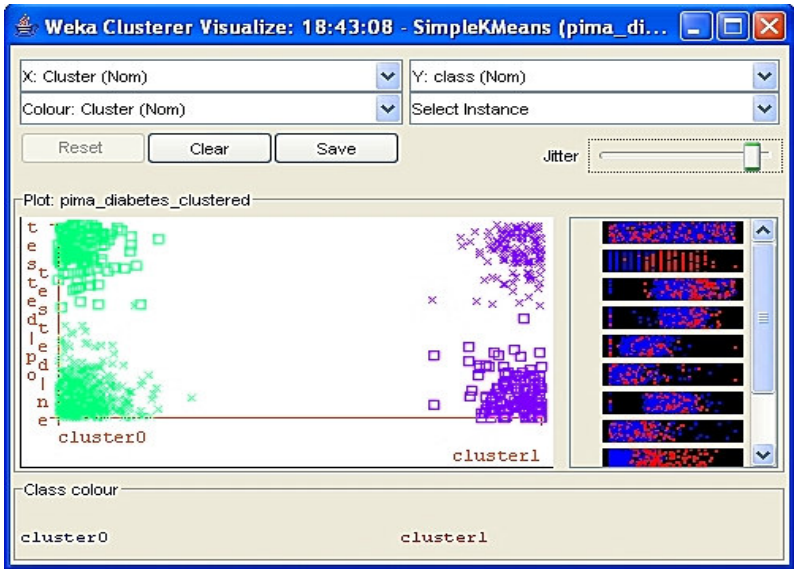

Figura 7. 46: Visualizzazione degli errori di K-means K=2

Se invece si imposta K ad un valore più alto, poiché si è scelto un valore che non rispecchia la distribuzione dei dati si ottiene, come è logico aspettarsi, un tasso di errore più alto raggiungendo il 43%.

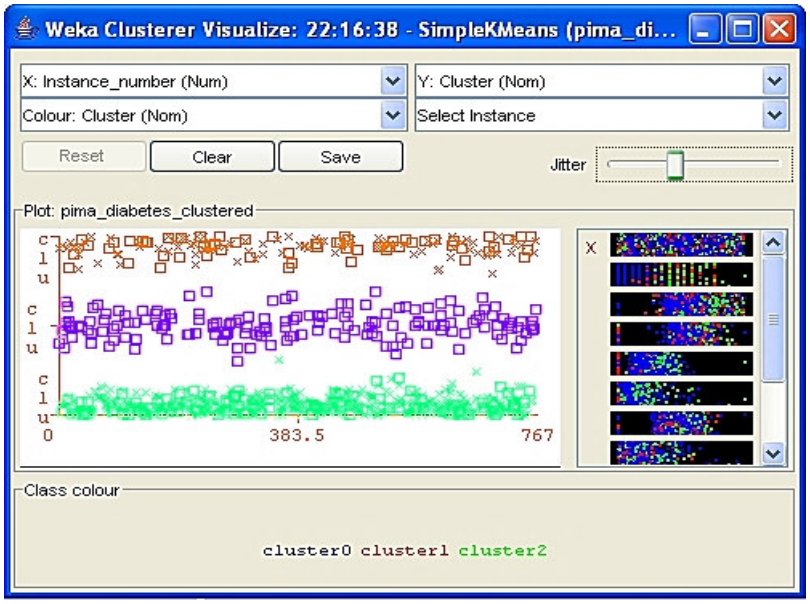

Figura 7. 47: Visualizzazione degli errori di K-means K=3

7.6 *Validità del test diagnostico: conclusioni*

Nell'attività diagnostica di routine, i test rappresentano elementi fondamentali e spesso determinante nel processo decisionale volto a confermare o a escludere la presenza di una determinata malattia già sospettata in base ai dati clinici. Fondamentali per la valutazioni di un test, sono le curve ROC (Receiver Operating Characteristic); esse sono state utilizzate per la prima volta durante la 2° guerra mondiale per l'analisi delle immagini radar e lo studio del rapporto segnale/disturbo. L'analisi ROC [7.3] viene effettuata attraverso lo studio della funzione che lega la probabilità di ottenere un test positivo nella classe dei malati veri alla probabilità di ottenere un risultato falso-positivo nella classe dei non malati. In altre parole vengono studiati i rapporti tra allarmi veri e allarmi falsi. La capacità discriminante di un test, ossia la sua attitudine a separare propriamente la popolazione in "malati"e "sani" è proporzionale all'area sottesa (AUC, Area Under Curve) dalla curva ROC ed equivale alla probabilità che il risultato su un individuo estratto a caso dal gruppo dei malati sia superiore a quella di uno estratto a caso dal gruppo dei sani. Per quanto

riguarda l'interpretazione del valore di AUC, si può tenere presente la classificazione della capacità discriminante di un test, proposta da Swets (1998); essa è basata sullo schema seguente:

- AUC=0.5: Test non informativo (La curva ROC è la bisettrice del primo quadrante)
- 0.5<AUC≤0.7: Test poco accurato
- 0.7<AUC≤0.9: Test moderatamente accurato
- 0.9<AUC≤1: Test altamente accurato
- AUC=1: Test perfetto

Con riferimento al nostro esempio, abbiamo trovato che il test che da risultati migliori è quello relativo a J48 infatti in tal caso AUC=0.75 (moderatamente accurato). I test ottenuti con RandomTree, OnerR hanno un valore di AUC circa uguale a 0.6 per cui sono poco accurati.

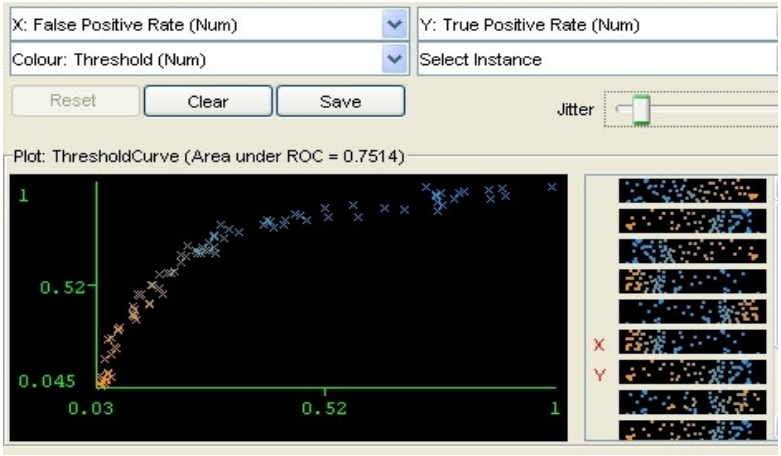

Figura 7. 48: Curva ROC per J48 (cross validation) per la classe test positivi

Una curva praticamente uguale, con la stessa AUC, si otterrà per la classe dei test negativi.

Bibliografia

[1.1] Kusiak A, Kem JA, Kernstine KH, Autonomous decision making: a data mining approach. IEEE Trans Inf Technol Biomed vol. 4 (2000): 274-84.

[1.2] htpp:// www.cineca.it

[1.3] U.Fayyad, G.Piatetsky-Shapiro, P.Smyth, R.Uthurusamy, Advances in knowledge discovery and data mining (2002)

[1.4] htpp://www.crisp-dm.org

[1.5] Cabena (1997), Discovering Data Mining: from concept to implementation, Prentice-Hall, Englewood Cliffs, NJ.

[1.6] Boley D. (2000), Principal direction divisive partioning: data mining and knowledge discovery.

[1.7] Groth R. (2000), Data mining: building competitive advantage. Prentice-Hall.

[2.1] htpp://www.kdnuggets.com

[2.2] Principles of DM Course, David Hand, Heikki Mannila, Padhraic MIT Press, 2001

[2.3] Roiger R. J., M. W. Geats, Introduzione al Data Mining, McGrawHill

[3.1] Kantardzic M., (2003), Data Mining: concepts, models, methods and algorithms, IEEE Press, Piscataway NJ.

[3.2] Breiman, Friedman, Olshen, Stone , (1984), Classification and Regression Trees, Chapman & Hall, New York-London.

[3.3] Kem, Kernstine (2000), Data Mining and decision trees.

[3.4] Antonio Manno, (2002), CART: Classificatio and Regression Trees.

[3.5] Safavian R.S, Landgreve D., (1991) A survey of decision tree classifier methodology. IEEE Trans, System Man Cybernet

[4.1] P. Arabie, L.J. Hubert end G. de Soete, (1996) Clustering and Classification, World scientific.

[4.2] htpp:// www.dsi.unifi.it

[4.3] A.K. Jain, R.C. Dubes, Algorithms for clustering data, Printice-Hall, (1988).

[4.4] D.H. Fisher, P. Langley, M.J. Pazzani, Concept formation: Knowledge and experience in unsupervised learning, Morgan Kaufmann, 1991

[4.5] D.H. Fisher, Optimization and simplification of hierarchical clusterings, Journal of Artificial Intelligence Research,1996

[4.6] www.elet.polimi.it

[4.7] M. Ester, Kriegel and X. Xu, A density-based algorithm for discovery clusters in large databases. KDD'96.

[4.8] Genetic Programming: On The Programming of Computers by Means of Natural Selection, J. Koza. MIT Press, 1992.

[5.1] Spiliopulou M. (2000), Web Usage Mining for Web Site Evaluation. Communication of the ACM, 43, 8.

[5.2] B. Mobasher, H. Dai, T. Luo, I. Sun, J. Zhu. Combining Web usage and content mining for more effective personalizzation. Conference on E-commerce and Web-Technologies (2000).

[5.3] Perkowitz, etzioni (1998), Adaptative Web Sites: Automatically Synthesizing Web Sites.

[5.4] http://medmole.test.cineca.it/

[7.1] www.cs.waikato.ac.nz/~ml/weka/

[7.2] www.lans.ece.utexas.edu/course/ee380l/04sp/

[7.3] Bottarelli, Parodi, Un test per la valutazione della validità dei test diagnostici: le curve ROC. Ann. Fac. Med. Parma (2003)