ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Seconda Facoltà di Ingegneria
Corso di Laurea Specialistica in Ingegneria Informatica

SVILUPPO DI SISTEMI DI DATA MINING PER L'ESTRAZIONE AUTOMATICA DI MODELLI PREDITTIVI NELLA CHURN ANALYSIS

Elaborata nel corso di: Sistemi Informativi Distribuiti LS

Tesi di Laurea di:
ALBERTO PINI

Relatore:
Prof. GIANLUCA MORO

Correlatore:
Ing. ALBERTO CASTORI

ANNO ACCADEMICO 2010–2011
SESSIONE III

PAROLE CHIAVE

Data Mining
Automazione di processi
Tecniche supervisionate
Previsione di abbandono
Modelli

Dedicato a tutti coloro che mi hanno aiutato e sostenuto in tutti questi anni.

Indice

Introduzione

Il Data Mining è una recente disciplina informatica, spesso poco conosciuta, ma la cui importanza in ambito aziendale è molto rilevante ed in certi contesti addirittura fondamentale; essa consente di ricavare in maniera automatica conoscenza da un insieme di dati che ad occhio nudo spesso non riesce ad emergere. La maggior parte dei sistemi in circolazione è tuttavia costituita da Data Warehouse, ossia sistemi in grado di aggregare e visualizzare i dati secondo determinati criteri scelti dall'operatore; il motivo del predominio di questi ultimi sistemi risiede nel fatto che essi sono usati per un insieme di applicazioni generiche e sono molto meno complessi e costosi.
Automatizzare un processo di Data Mining non è un'operazione banale: quest'ultimo per definizione prevede già l'uso di algoritmi che in maniera automatica o semiautomatica estraggono informazione utile, ma la parte più complicata è automatizzare e generalizzare l'intero processo che inizia dalla preparazione dei dati e si conclude con la fase vera e propria di Data Mining con la quale verrà generato e validato un modello di previsione che potrà essere successivamente applicato a dati sconosciuti.
Tra i fenomeni più comuni e di maggiore interesse per le imprese vi è quello degli abbandoni dei clienti, tecnicamente chiamato *churn* della clientela; sapere in anticipo quali di questi ultimi è intento ad abbandonare una determinata azienda è un'informazione molto preziosa che, se sfruttata a dovere, può evitare di perdere gran parte della clientela ed avere il vantaggio così di rimanere competitivi nel mercato.

L'obiettivo di questa tesi sarà quello di analizzare i dati di una nota azienda e prevedere per quest'ultima i clienti a rischio di abbandono; tutto ciò sarà reso possibile dallo sviluppo di un sistema software che

automatizzerà l'intero procedimento.
Tutto ciò è in realtà il proseguimento di un precedente lavoro[1]: si riprenderanno quindi i contenuti di quest'ultimo e partendo da essi, lo si continuerà. Sebbene il problema affrontato sia lo stesso (il churn della clientela), questa tesi sarà incentrata tuttavia su un nuovo caso aziendale: quest'ultimo verrà dunque analizzato e si cercherà di applicare e di verificare le tecniche risolutive già consolidate; prendendo spunto da ciò, queste ultime verranno migliorate ed ampliate nel caso si ritenga che esse si dimostrino carenti o inadeguate per questo specifico caso che si andrà a trattare.
Uno tra gli scopi principali sarà quello di contribuire significativamente allo sviluppo del sistema software di base che è tuttora a disposizione, incrementandone le prestazioni ed aggiungendo funzionalità non ancora presenti; tale sistema dovrà alla fine essere in grado di automatizzare il processo nella maniera più completa ed efficiente possibile.
Concentrandosi inoltre su un caso di studio differente da quello per cui era predisposto inizialmente il sistema software, sarà possibile constatare quanto quest'ultimo possa offrire una soluzione generica adattabile a realtà aziendali eterogenee, obiettivo, quest'ultimo, molto importante.
Si percorrerà fase per fase il processo risolutivo, arrivando infine ad avere uno o più modelli ad hoc per l'impresa in questione; essi devono essere in grado di individuare in maniera ottimale i clienti a rischio di abbandono anche per dati sconosciuti; il metodo risolutivo seguito è il CRISP-DM, ampiamente utilizzato in ambito aziendale per risolvere problemi inerenti il Data Mining.

La tesi è composta da 6 capitoli:

- Nel primo capitolo viene fatta un'introduzione al Data Mining, illustrando inoltre il metodo CRISP-DM a livello teorico.

- Nel secondo capitolo si analizza il problema della previsione dei clienti a rischio di abbandono per l'azienda in questione, seguendo il procedimento CRISP-DM mostrato nel primo capitolo.

[1] Vedere riferimento bibliografico [6].

- Nel terzo capitolo si descrive il sistema software sviluppato in precedenza, usato tuttavia per risolvere un caso aziendale differente.

- Nel quarto capitolo vengono mostrate tutte quelle modifiche ed aggiunte apportate al software di base. Esse sono descritte seguendo due distinte fasi dello sviluppo del sistema: la prima riguarda gli interventi che occorrono a livello generale per renderlo più completo ed efficiente per i successivi utilizzi, la seconda, invece, si riferisce a tutti quei cambiamenti che sono necessari per predisporlo al caso aziendale che si sta affrontando. Dopo questi interventi si avrà quindi un sistema software più performante, stabile e completo.

- Nel quinto capitolo vengono spiegati i criteri con i quali sono state eseguite le simulazioni ed i test per permettere al sistema implementato di ottenere un buon modello di previsione; inoltre vengono riportati i risultati ottenuti, che ci permetteranno infine di selezionare i modelli più promettenti.

- Nel sesto capitolo si traggono le conclusioni, proponendo anche suggerimenti per futuri sviluppi e migliorie.

Capitolo 1

Data Mining

In questo capitolo si intende dare una panoramica sul Data Mining e sui processi che esso coinvolge[1].

1.1 Overview

Con il termine *Data Mining* ci si riferisce all'*estrazione di informazione utile e sconosciuta da grandi quantità di dati con l'ausilio di mezzi automatici*; da questa definizione deriva il nome di Data Mining, ossia appunto "estrazione di dati".
Il Data Mining è una disciplina nata negli anni '90 dalle conoscenze in diversi ambiti:

- Statistica
- Informatica (in particolare Database)
- Intelligenza Artificiale (Machine Learning)

Questa infatti è una scienza che si appoggia su fondate basi teoriche di Matematica e Statistica, applicando tecniche di Machine Learning a database gestiti da sistemi informatici con lo scopo di ricavare conoscenza.
La maggior parte dei sistemi attualmente in uso per il supporto alle

[1] La maggior parte del materiale presente in questo capitolo è tratto dai riferimenti bibliografici [23] è [21], con rielaborazione nella tesi specialistica [6].

decisioni è orientato prevalentemente ad un'analisi eseguita tramite strumenti OLAP[2], che si limitano ad esporre i dati in maniera ordinata, sistematica e secondo i criteri voluti da un operatore, tuttavia è poi quest'ultimo a dover trarre superficiali conclusioni ed eventualmente relazioni tra i dati, che quasi sempre per la loro mole è però impossibile individuare, perché sfuggono all'occhio umano; questo divario tra la possibilità di immagazzinare dati da parte degli elaboratori, che con il passare degli anni aumenta esponenzialmente, e la capacità umana di analizzarli, che invece rimane costante nel tempo, è definito *information gap*.
Lo scopo finale del Data Mining invece consiste nel ricavare in maniera automatica conoscenza da dati grezzi (*raw data*), sotto forma di *pattern*.
I pattern sono informazioni strutturate, degne di essere estratte ed aventi le caratteristiche di essere:

- precedentemente sconosciute
- valide in modo da essere usate su nuovi dati
- comprensibili

Un pattern serve per comprendere e prevedere un certo fenomeno; nel caso se ne conosca la struttura si ha un approccio di tipo *white box* e si possono raggiungere entrambi gli scopi, in caso contrario, si parla invece di *black box* e si può utilizzare solo per fare previsioni.

1.2 Knowledge Discovery in Databases

Il *Knowledge Discovery in Databases*, o più semplicemente *KDD*, è il processo che porta a ricavare conoscenza utile a partire dai dati grezzi. Esso coinvolge numerosi aspetti in ambito dei sistemi per il supporto alle decisioni (selezione, trasformazione, modifica, gestione ed ottimizzazione dei dati, ecc.) e per tale ragione si ricorre alle Business Intelligence[3] come strumento pratico per attuare questo lavoro. Il *KDD* si presta ad essere un processo versatile e che considereremo

[2]Vedere appendice A.3.1 - *OLAP*.
[3]Vedere appendice B - *Le Business Intelligence*.

come generale, ossia una linea guida da seguire per trasformare dati grezzi in conoscenza[4].
In questo procedimento, il Data Mining è inserito come fase finale, mentre la maggior parte del processo è dedicata alla creazione del Data Warehouse[5] o più semplicemente alla preparazione dei dati.
Il *KDD* si sviluppa generalmente in 5 fasi:

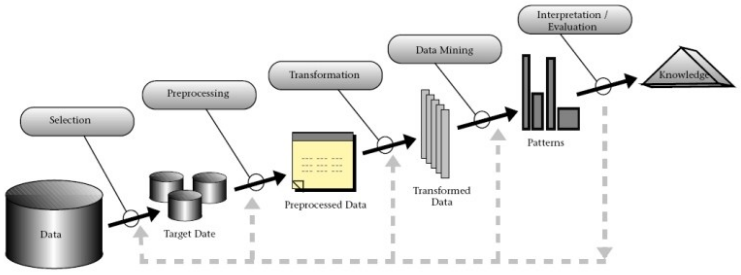

Figura 1.1: *Knowledge Discovery in Databases*

1. *Selezione*
   Vengono selezionati i dati target per la creazione di un set di dati consono allo specifico caso, scartando tutti i dati che possono risultare inutili.

2. *Preprocessing*
   Anche se si possiede un set potrebbe essere necessario prenderne in considerazione solo una parte; inoltre fa parte di questa fase anche il *data cleaning*, che consiste nel trovare la soluzione più adeguata per risolvere errori, dati inconsistenti o mancanti.

3. *Trasformazione*
   Dal momento che i dati possono provenire da fonti diverse, se necessario occorre convertirli, adattarli o derivarne di nuovi, per renderli compatibili tra loro e quindi utilizzabili.

[4]Si tratterà ampiamente nel dettaglio nel paragrafo seguente ogni singolo aspetto accennato in questa sezione, anche se sotto il punto di vista di una metodologia più specifica.

[5]Vedere appendice A - *Database operazionali e Data Warehouse.*

4. *Data Mining*
   A questo punto vengono applicati gli algoritmi di Data Mining per estrarre i modelli dai dati precedentemente selezionati ed ottimizzati.

5. *Interpretazione/Valutazione*
   I modelli vengono utilizzati con dati diversi di test per misurarne le prestazioni. A questo punto, se necessario, si procede con il ripetere il processo a partire da una fase già effettuata in precedenza.

1.2.1 Modello alternativo

Il Knowledge Discovery in Databases è più un concetto generale da seguire che una tecnica specifica vera e propria, pertanto potrebbe essere più utile prendere in considerazione uno schema del *KDD* con qualche variante appropriata rispetto a quello precedente[6], avvicinandoci, come vedremo più avanti, di più alla metodologia che si userà effettivamente come processo di Data Mining.
Le prime 3 fasi infatti appartengono ad un processo di *ETL* (*Extract, Transform and Load*), pertanto a volte risulta più efficace accorparle; inoltre una conoscenza alternativa (sebbene più superficiale) può essere data anche dagli strumenti OLAP, che quindi è possibile includere come complemento o operazione sostitutiva al Data Mining. Nel *KDD* descritto precedentemente, si dà poi per scontata la disponibilità di un'unica sorgente da cui prelevare i dati, tuttavia nella maggior parte dei casi la situazione reale è molto differente ed occorre creare innanzitutto il database consolidato.
Uno schema alternativo del *KDD* quindi è:

- **Consolidamento dei dati**
  I dati grezzi sparsi tra varie sorgenti vengono corretti e raggruppati in un unico database. In genere questa è un'operazione abbastanza complessa, lunga e laboriosa.

- **Selezione e preprocessing**
  I dati consolidati vengono selezionati, modificati o creati nel mo-

[6] Vedere riferimento bibliografico [14].

do più appropriato, a seconda del tipo di analisi che si vuole condurre e dello scopo.

- **Analisi OLAP e Data Mining**
  I dati preparati vengono sottoposti ad analisi OLAP da parte di un operatore o vengono elaborati da algoritmi di Data Mining tramite i quali si estraggono pattern e modelli di valutazione.

- **Interpretazione e valutazione**
  I modelli ricavati precedentemente vengono applicati a nuovi dati per valutarne le prestazioni e per ricavare in maniera automatica conoscenza utile alle decisioni.

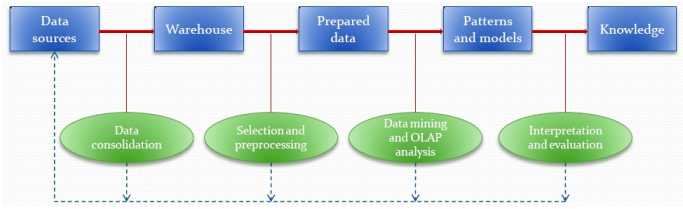

Figura 1.2: *Knowledge Discovery in Databases (variante)*

Consolidamento dati I dati che inizialmente si hanno a disposizione possono provenire da fonti eterogenee (file di differente formato o database di diversa natura) ed in generale sono disponibili in forma grezza, ossia sono strutturati in maniera non corrispondente tra loro e spesso sono anche inconsistenti ed inaffidabili. Inoltre nella maggior parte dei casi sono sparsi ed occorre recuperarli all'interno della rete per convogliarli in un unico sistema e poterli quindi gestire. È necessario perciò raccogliere le varie risorse distribuite, selezionare e tenere solo i campi che interessano, aggregare gli attributi, normalizzare i dati e correggerli, eliminando eventualmente outlier ed informazioni mancanti o che presentano errori; tale operazione, come accennato in precedenza, è detta di ETL.
Infine il risultato finale viene strutturato in un unico database che conterrà quindi i dati revisionati.

Selezione e preprocessing Anche in questo caso occorre eseguire un processo di ETL per rendere i dati idonei alle esigenze richieste. A seconda dello scopo dell'analisi che si vuole condurre e delle relazioni che intercorrono tra i dati, questi ultimi possono essere aggregati, campionati, ridotti di dimensionalità, scelti per mantenere solo un sottoinsieme, creati come attributi derivati, discretizzati, binarizzati e trasformati; inoltre, nel caso non lo si sia già fatto durante la fase di consolidazione, occorre cercare di ridurne il rumore, gestire i valori mancanti ed eliminare eventuali dati duplicati.
Per potere condurre al meglio un'analisi di tipo OLAP, in questa fase occorre definire lo schema del cubo che servirà per stabilire dimensioni, misure e gerarchie in cui sono suddivisi i dati; per un'analisi automatizzata tramite tecniche di Data Mining bisogna invece ottimizzare i dati in modo da ridurre i tempi di calcolo e possibili errori di valutazione.
Dopo tale revisione, i dati sono pronti per essere consultati manualmente con strumenti OLAP o per essere elaborati da algoritmi di Data Mining.

Analisi OLAP e Data Mining Tramite il Data Mining, come già detto, è possibile individuare automaticamente correlazioni che intercorrono tra i dati ed usarle per prevedere i valori di attributi di altri dati sconosciuti, se invece in precedenza si è realizzato un vero e proprio sistema di Data Warehouse, si possono effettuare query OLAP (nella maggior parte dei casi tramite tabelle pivot).
La fase di Data Mining produce dei modelli che potranno poi essere in seguito applicati su nuovi dati.

Interpretazione e valutazione Per ricavare conoscenza si continua la fase di Data Mining, applicando il modello prodotto precedentemente a dati sconosciuti e, se possibile, si confronta il risultato con dati di test di cui si sa il valore, per constatare la qualità e l'affidabilità della soluzione.
In caso di analisi OLAP è l'operatore che deve trarre deduzioni in base alle query effettuate al punto precedente.
Il risultato prodotto è la conoscenza del futuro andamento dei valori dei dati desiderati; è quindi possibile poi sfruttare tale conoscenza a

proprio vantaggio in ambito decisionale. Essa infine viene in genere visualizzata o conservata sotto forma di report.

1.3 Metodologia CRISP-DM

Ci si concentrerà ora invece su metodologie mirate esclusivamente al Data Mining e che puntano ad ottimizzarne il processo. Si è già visto il *KDD* come procedimento generale per estrarre conoscenza, ma nella pratica in ambito aziendale sono stati sviluppati e messi in atto altri modelli più pratici, specifici ed idonei[7].
L'approccio più ampiamente utilizzato per risolvere un problema di Data Mining è tuttora il modello di processo *CRISP-DM* (*CRoss Industry Standard Process for Data Mining*)[8] e sarà quello che si prenderà in considerazione da qui in avanti[9].
Il progetto è nato nel 1996 grazie al consorzio di alcune aziende:

- *ISL* (acquisita da *SPSS Inc.*)
- *NCR Corporation*
- *Damler-Benz*
- *OHRA*

Il processo si compone di 6 fasi:

1. **Business Understanding**
2. **Data Understanding**
3. **Data Preparation**
4. **Modeling**
5. **Evaluation**
6. **Deployment**

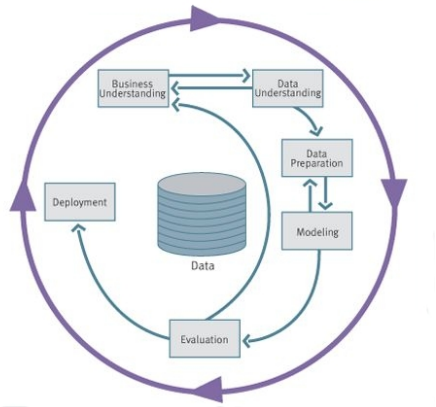

Figura 1.3: *Metodologia CRISP-DM*

Tali fasi sono eseguite prevalentemente in maniera sequenziale, ma a seconda della qualità dei risultati ottenuti, da un passo potrebbe essere necessario tornare indietro, per perfezionare e revisionare di nuovo una fase già eseguita ed ottenere così poi risultati differenti; questo procedimento (o parte di esso) viene quindi iterato fino al raggiungimento del risultato voluto. La maggior parte di queste fasi ricalcano ed inglobano quelle già presenti anche nel *KDD*, altre invece vengono aggiunte o ulteriormente sviluppate.

1.3.1 Business Understanding

In questa prima fase occorre capire e collocare il determinato progetto nella giusta categoria, in base agli obiettivi che esso deve raggiungere. Si sono individuati 4 principali *task* con cui potere affrontare uno specifico problema:

- **Classificazione**
  Si etichettano gli elementi di un insieme allo scopo di catalogarli.

[7]Vedere riferimento bibliografico [3].
[8]Vedere riferimento bibliografico [19].
[9]Esistono comunque molte altre metodologie: tra le più note si può ricordare anche *SEMMA*, sviluppata da *SAS Institute*.

I valori attribuibili sono prestabiliti, limitati e dipendenti dal particolare dominio applicativo.

- **Clustering**
  Si suddivide un insieme di elementi in sottoinsiemi omogenei; a differenza della classificazione, i raggruppamenti possibili non sono noti a priori ma devono essere scoperti.

- **Analisi delle associazioni**
  Si trovano regole associative basandosi sull'occorrenza di un elemento all'interno di una transazione in riferimento alla presenza di un altro elemento dell'insieme.

- **Predizione numerica**
  È simile alla classificazione, ma il valore da attribuire invece di essere rappresentato da una classe categorica, è un valore numerico continuo.

Classificazione e *predizione numerica*, poiché prevedono un addestramento che necessita di avere dati già precatalogati, sono approcci *supervisionati*, mentre *clustering* ed *analisi delle associazioni* al contrario sono *non supervisionati*, cioè sono adatti a problemi in cui occorre estrapolare dai dati caratteristiche e relazioni che non si conoscono a priori.
Inoltre in questa fase iniziale rientra lo studio e l'analisi globale del problema in base anche all'impatto economico ed alla sua fattibilità di risoluzione.
Il fattore cruciale nel Business Understanding rimane comunque individuare la giusta tipologia di appartenenza del problema, poiché in base ad essa poi occorrerà applicare tecniche ed algoritmi specifici.

1.3.2 Data Understanding

In questa fase si prende consapevolezza della qualità dei dati a disposizione: tramite un interfacciamento a sistemi relazionali o di Data Warehouse con SQL o MDX si cerca di creare una visione di insieme del dominio applicativo.

Dati e qualità

I dati sono un insieme di *data object*, chiamati anche record, osservazioni o istanze. Ognuno di essi è formato da una o più proprietà dette *attributi*; un data object è quindi descritto dall'insieme dei suoi attributi.
Un attributo rappresenta una caratteristica dell'osservazione che può assumere determinati valori, i quali ne stabiliscono il tipo:

nominale
: Possiede la caratteristica *distinctness*, ossia ogni valore deve essere diverso dall'altro.

ordinale
: Ha anche la proprietà di *order*, cioè i valori permettono un ordinamento.

intervallo
: Oltre alle precedenti proprietà ammette anche operazioni di *addizione* e *sottrazione*.

ratio
: Comprende tutte le precedenti proprietà con l'aggiunta di *moltiplicazione* e *divisione*.

Inoltre, a seconda della propria natura, un attributo può essere *numerico*, se può assumere come valore solo dei numeri, o *categorico* in caso contrario. È ovvio che intervallo o ratio, possono essere solo numerici, mentre gli attributi nominali ed ordinali potrebbero essere anche categorici.
Si può inoltre effettuare un'ulteriore distinzione tra attributi *discreti*, se possono solo assumere valori appartenenti ad un insieme finito o infinitamente numerabile (si ha una corrispondenza 1:1 con l'insieme dei numeri naturali), ed attributi *continui* se invece appartengono all'insieme dei numeri reali; sotto questo aspetto, gli attributi nominali ed ordinali possono essere solo discreti, mentre intervalli e ratio potrebbero essere anche continui.

Un insieme di data object si definisce *dataset* e, considerando istanze con lo stesso numero di attributi come in genere accade, potremmo

rappresentarlo tramite una matrice $m \times n$, detta *data matrix*, dove m è il numero di istanze ed n quello degli attributi. Ogni osservazione può essere quindi rappresentata come un punto in uno spazio multi-dimensionale, in cui ad ogni dimensione corrisponde un attributo.
Per valutare la qualità di un dataset occorre prendere in considerazione:

- rumore ed outlier
- valori mancanti
- dati duplicati

Rumore ed outlier Il rumore consiste in modifiche accidentali ai dati dovute ad errori di trasmissione o di memorizzazione, mentre gli outlier sono dati dalle caratteristiche apparentemente anomale perché differiscono notevolmente da tutti gli altri. L'analisi che permette di individuare gli outlier viene detta *outliers detection* e si basa sul fatto che i dati validi sono quelli in maggior quantità.

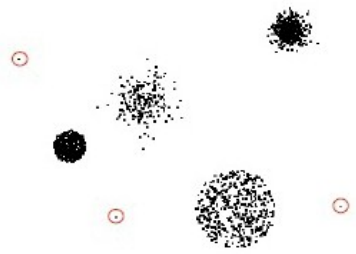

Figura 1.4: *Outliers*

Gli approcci più comuni per individuare gli outlier sono:

statistical based
: Si presuppone che all'interno di un dataset D esistano osservazioni che generalmente dovrebbero seguire una distribuzione di probabilità M, o, in caso contrario, un'altra funzione A che invece è seguita dagli outlier.
All'inizio si assume che l'intero dataset D appartenga ad M e

su di esso si calcola la funzione di probabilità logaritmica. Successivamente si toglie da M un $xt \in D$ e si calcola la nuova funzione di probabilità logaritmica: se la differenza tra questo valore e quello precedente supera una certa soglia prefissata, xt viene considerato appartenente ad A, altrimenti lo si reinserisce e si continua ad iterare il procedimento per tutti gli elementi restanti di D.
Questo approccio è applicabile solo se la funzione di distribuzione di probabilità dei dati è nota e questo è il principale inconveniente della sua possibilità di impiego.

distance based
Si calcola la distanza tra le varie istanze che vengono viste come un vettore di feature; gli outlier vengono poi stabiliti a seconda del valore ottenuto ed al caso specifico.

cluster based
I dati vengono clusterizzati: i cluster di piccole dimensioni individuati sono considerati outlier.

Valori mancanti A volte è possibile che un certo attributo non possa essere applicato a tutte le osservazioni o non possa essere fornito: si ha dunque un *missing value*.
In tal caso le alternative possibili sono:

- ignorare l'osservazione

- stimare il missing value

- ignorare il missing value

Dati duplicati I dati duplicati si hanno soprattutto quando si cerca di unire sorgenti multiple; conseguentemente questo può portare a prendere in considerazione lo stesso dato diverse volte nel caso compaia in più fonti. Anche in questa situazione, come per i valori mancanti, si dovrà seguire un tipo di approccio che dipenderà dallo specifico caso.

1.3.3 Data Preparation

Dopo un processo di raccolta dei dati e di pulizia, per ottenere il dataset finale che poi dovrà essere elaborato nella fase successiva dagli algoritmi di Data Mining, occorre effettuare una preliminare operazione di selezione e trasformazione, detta Data Preparation o Preprocessing; il dataset definitivo sarà una tabella relazionale, in genere non normalizzata.

Trasformazione dei dati

Le principali operazioni che si possono eseguire sugli attributi per renderli poi conformi alla fase successiva di Modeling sono:

- aggregazione
- campionamento
- selezione degli attributi rilevanti
- creazione di nuovi attributi
- discretizzazione
- trasformazione di attributi

Aggregazione Più attributi o istanze possono essere combinati insieme, in modo da ottenerne uno solo; grazie all'aggregazione si possono definire gerarchie, così da potere effettuare cambiamenti di scala. Inoltre questa operazione contribuisce a velocizzare i tempi di elaborazione ed ad avere meno variabilità nei valori rispetto a quelli originali.

Campionamento Se si dispone di un dataset molto grande, è spesso necessario prenderne in considerazione solo una parte, perché sarebbe troppo oneroso in termini di tempo e capacità di elaborazione usarlo per intero. Il campione tuttavia non deve essere preso a caso, ma deve essere rappresentativo dell'intero dataset originale, per preservarne le caratteristiche.

Tra le tipologie di campionamento si hanno:

simple random sampling
: È il tipo più comune: ogni istanza ha la stessa probabilità di essere selezionata, inoltre può essere utilizzata anche la variante con reimbussolamento (ogni istanza può essere scelta di nuovo anche se già selezionata precedentemente).

stratified sampling
: Suddivide il dataset in più partizioni ed ad ognuna di esse applica poi il simple random sampling.

Selezione degli attributi rilevanti Si può diminuire il numero di attributi, tralasciando quelli irrilevanti o ridondanti, in modo da ridurre la dimensionalità. Selezionandone solo un sottoinsieme e lavorando quindi poi con un numero ridotto di feature tra quelle più significative si può ottenere un netto miglioramento dei risultati prodotti dagli algoritmi di Data Mining, inoltre si evita la *"maledizione della dimensionalità"* (*the curse of dimensionality*), situazione che porta i dati ad essere molto più sparsi nello spazio in cui sono rappresentati e questo rende notevolmente difficoltoso il calcolo della densità e della distanza tra i punti.
Si può attuare un'opera di selezione in diversi modi:

brute force
: È la maniera più semplice per affrontare il problema e consiste nel provare tutte le combinazioni possibili di attributi.

wrapper
: Si utilizza l'algoritmo di Data Mining come black box per la selezione migliore.

filter
: Prima di eseguire l'algoritmo di Data Mining, viene applicato un filtraggio che tiene in considerazione solo le caratteristiche dei dati; questo significa una ricerca degli attributi che dividono il dataset nella maniera migliore. Una ricerca di tipo esaustivo,

come la brute force, è quella più semplice, ma anche la più costosa.
Nella pratica lo spazio degli attributi è percorso in modo greedy lungo due direzioni: top-down (*forward selection*) e bottom-up (*backward selection*), convergendo poi ad un ottimo globale. Nel primo caso per migliorare la soluzione ad ogni step viene aggiunto un attributo a quelli già selezionati, mentre nel secondo caso viene rimosso.

analisi delle componenti principali

L'insieme dei punti rappresentanti il nostro dataset viene mappato e riportato in un altro spazio con un numero di dimensioni inferiore, scegliendo gli assi lungo le direzioni di maggior varianza dei punti. Per eseguire tale operazione occorre:

1. trovare gli autovettori della *matrice di covarianza*[10]
2. con gli *autovettori*[11] definire il nuovo spazio

Costruita la *matrice di covarianza*, in seguito è possibile prendere solo i maggiori tra gli autovalori (riducendo così la dimensionalità) e da questi si calcolano i rispettivi autovettori.
Si ottiene infine la *matrice di rotazione*, grazie all'operazione matriciale $w = V \times x$, dove w è un autovettore e x il vettore originario. Questa matrice permette di ottenere un nuovo spazio a partire da quello originale.

[10]Per matrice di covarianza si intende la matrice quadrata di dimensioni $D \times D$ dove D è il numero di attributi di ciascuna osservazione del dataset.
Ogni elemento è definito come:

$$\sigma_{ij} = \frac{1}{n}\sum_{h=1}^{n}(x_{hi} - \mu_j)^2$$

dove n è il numero di osservazioni, μ_j la media dei valori della j-esima caratteristica, x un'osservazione e x_{hi} il valore della sua caratteristica i-esima.

[11]I valori all'interno della matrice di covarianza indicano la correlazione tra le caratteristiche i e j, mentre quelli sulla diagonale principale rappresentano la varianza delle caratteristiche e sono detti autovalori.
La correlazione vale *1* quando tendono a crescere, *-1* se tendono a diminuire, e *0* quando sono statisticamente indipendenti.

Creazione di nuovi attributi Si possono creare nuovi attributi usando quelli esistenti; a volte infatti è utile avere già a disposizione attributi che derivano da operazioni algebriche (deviazione standard, differenza, somma, media o altre funzioni particolari calcolabili) eseguite sull'attributo originale o con più attributi.

Discretizzazione Alcuni algoritmi lavorano solo con attributi discreti, pertanto occorre convertirli se invece si hanno a disposizione attributi continui. La discretizzazione può essere supervisionata nel caso si tenga conto della classe, mentre in caso contrario si definisce non supervisionata; questi algoritmi lavorano su intervalli, detti *bin*. La principale tecnica *supervisionata* è:

entropy based
: Tenta di ricavare la massima purezza delle partizioni, mettendo in ogni bin il maggior numero possibile di istanze sul totale di quelle appartenenti alla stessa classe e cercando di minimizzare l'entropia. Per evitare l'overfitting nei problemi di classificazione si applica solo al training set e si usa successivamente il risultato sul test set.

Per quanto riguarda la discretizzazione *non supervisionata* si hanno due modalità:

equal interval binning
: Il range di valori di un attributo viene diviso in bin di uguale ampiezza, senza tenere conto della distribuzione delle istanze, pertanto si potranno ottenere alla fine bin con cardinalità elevata, mentre altri che saranno vuoti.

equal frequency binning
: Si scelgono gli intervalli in base al numero di istanze in essi presenti, in modo che ognuno ne contenga lo stesso numero. Dal momento che sono entrambe tecniche non supervisionate, possono produrre discretizzazioni tali da eliminare variazioni dei dati utili per discriminare la classe di appartenenza, pertanto è sempre preferibile ricorrere a tecniche supervisionate, ma nei

casi in cui non si conosca la classe di appartenenza (per esempio con il clustering) si è comunque obbligati ad utilizzare una discretizzazione non supervisionata.

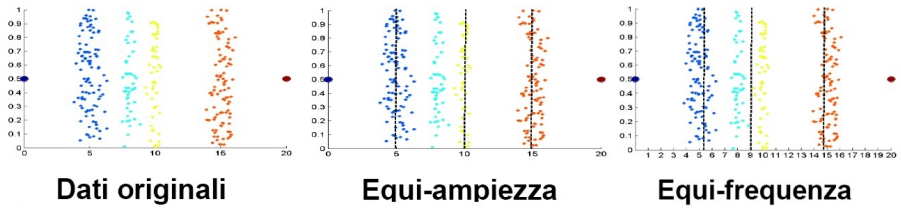

Figura 1.5: *Discretizzazione non supervisionata*

Trasformazione di attributi A volte può essere utile mappare l'insieme dei valori di un attributo su un altro insieme, in modo da mantenere comunque una corrispondenza 1:1.

1.3.4 Modeling

Dopo aver individuato il task di appartenenza del problema ed aver preparato adeguatamente il dataset da usare, si possono ora applicare le tecniche di Machine Learning per estrapolare i pattern dai dati a disposizione. È determinante al fine di ottenere una soluzione di buona qualità, scegliere l'algoritmo di Data Mining più adeguato da applicare ed è altrettanto importante configurarne in maniera appropriata ed ottimale i parametri. Alcuni problemi potrebbero richiedere anche l'uso di più algoritmi, ad ognuno dei quali occorre un dataset con caratteristiche differenti, per questo potrebbe essere necessario ripetere la fase di Data Preparation per ogni applicazione di un diverso algoritmo.
Gli algoritmi da utilizzare dipendono prevalentemente dal tipo di task che richiede il problema, quindi occorrerà distinguere tra classificazione, clustering, analisi delle associazioni e predizione numerica[12].

[12]In questa sezione sono presentati a livello generale, per alcuni esempi di specifici algoritmi noti vedere l'appendice C - *Algoritmi di Data Mining*.

Classificazione

La classificazione consiste nell'estrarre dai dati un modello di classificazione, ossia una funzione che permetta di assegnare ad ogni istanza avente un determinato insieme di attributi il giusto valore dell'attributo *classe*. La classe, che è la variabile dipendente della funzione estratta, può essere solo un attributo categorico e discreto, mentre tutti gli altri attributi, che sono le variabili indipendenti, possono essere di qualsiasi altro tipo.
Genericamente un task di classificazione ha bisogno di un *training set*, ossia un dataset in cui le istanze sono state già precedentemente etichettate, ed hanno quindi l'attributo classe noto, e su questo insieme di istanze si applica un particolare algoritmo di Data Mining estraendo poi il modello. La qualità di quest'ultimo viene poi valutata successivamente grazie ad un secondo dataset, detto *test set*, che, al contrario del primo, si ipotizza avere l'attributo classe sconosciuto; si applica quindi il modello al test set per etichettarne le istanze ed infine si valutano le prestazioni in base agli errori di assegnazione.

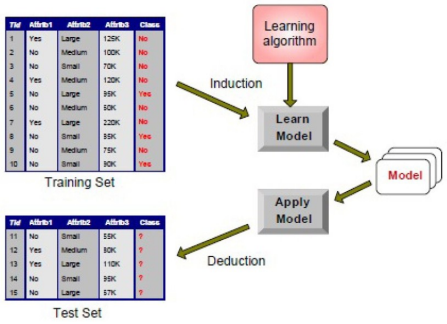

Figura 1.6: *Generazione ed applicazione di modelli di classificazione*

Una tecnica di classificazione è un approccio per la costruzione di un determinato tipo di modello o classificatore e per ognuna di esse esistono più algoritmi di Machine Learning che le implementano.
Tali classificatori si possono distinguere in:

- decision tree
- rule based

- instance based
- bayesiani
- reti neurali

Inoltre esistono anche tecniche per combinare più modelli (*aggregatori* o *classificatori misti*), sia prodotti dalla stessa tipologia, sia da tipologie differenti di classificatori.

Classificatori decision tree Procedono seguendo una struttura gerarchica ad albero, formata da nodi connessi da archi.
I nodi possono essere:

- root (nodo di partenza)
- interni
- foglie (nodi terminali)

Ad ogni foglia viene assegnato un valore dell'attributo classe; l'albero viene esplorato partendo dalla root e navigando tra i vari nodi interni ad ognuno dei quali corrisponde una determinata condizione test su un particolare attributo e questo determinerà la scelta del successivo nodo presente nella ramificazione; il processo prosegue fino a quando si giunge ad un nodo terminale, di cui si assegnerà la classe.

Decision tree induction è il procedimento che consiste nel ricavare un albero decisionale da un training set; a partire dal medesimo dataset possono essere prodotti più alberi, nella maggior parte dei casi è computazionalmente impossibile trovare quello ottimo, tuttavia in base all'algoritmo utilizzato è possibile ricavare alberi accettabilmente buoni in tempi ragionevolmente brevi.
Tutti gli algoritmi che appartengono a questa tipologia di classificazione attualmente procedono con una strategia di tipo greedy, separando i record a seconda di una condizione test presente ad ogni nodo su un attributo che ottimizza determinati criteri; si procede in questo modo aggiungendo mano a mano altri nodi finché non si hanno miglioramenti.

Nella costruzione di un albero ci sono alcuni aspetti da considerare:

- come specificare la condizione di test
- come determinare lo split migliore di volta in volta
- quando fermare la costruzione dell'albero

Tuttavia in questa sezione non ci soffermeremo nei particolare di come determinare queste caratteristiche o situazioni[13].

L'uso di classificatori decision tree, porta numerosi vantaggi:

- induzione computazionalmente efficiente
- velocità a classificare nuove istanze
- modelli facili da interpretare con alberi di piccole dimensioni

Tuttavia esistono anche diversi contro:

- alta sensibilità ai missing value (possono alterare le stime di impurità)
- data fragmentation con alberi molti grandi (molti nodi contenenti un basso numero di istanze)
- espressività limitata (il test sullo split avviene su un attributo alla volta e su più attributi sarebbe troppo oneroso, inoltre questo porta i decision boundary in cui si partiziona il dataset ad essere paralleli agli assi, rendendo difficile modellare relazioni complesse)

Una volta ottenuto un modello decision tree da un training set ed applicato ad un test set, si definisce *errore di risostituzione* quello sul primo, *errore di generalizzazione* quello sul secondo[14]; per avere un buon modello occorre che essi siano entrambi bassi.

[13]Per maggiori dettagli vedere l'appendice C.1.1 - *Classificatori: Decision Tree, costruzione di un albero.*

[14]Vedere il paragrafo seguente, sulla valutazione dei risultati.

Si possono avere comunque situazioni particolari:

model underfitting (entrambi gli errori alti).
: In genere si verifica con modelli troppo semplici o grossolani, non rappresentativi del dominio di interesse.

model overfitting (errore di risostituzione basso, errore di generalizzazione alto).
: Compare con modelli che si adattano troppo bene al training set e quindi non riescono a generalizzare sulla realtà in esame, avendo poi scarsi risultati su nuovi test set.

In caso di overfitting le cause possono essere:

- presenza di rumore (esempi sbagliati del training set portano alla costruzione di alberi inconsistenti).

- training set poco rappresentativi della popolazione in esame, e ciò richiede di rivedere i criteri di campionamento.

- complessità eccessiva (alberi con un elevato numero di foglie).

Per evitare i primi due punti occorre rivedere la fase di preprocessing; invece per risolvere il problema della complessità si possono adottare due differenti tecniche di costruzione dell'albero:

pre-pruning
: Si termina anticipatamente la costruzione dell'albero imponendo condizioni più restrittive: il processo viene fermato se il numero di istanze di un particolare nodo è inferiore ad una specifica soglia, se non si migliora la misura di impurità e se l'attributo classe ha bassa correlazione con quelli disponibili.

post-pruning
: L'albero viene costruito interamente, poi si procede con la potatura procedendo bottom-up, terminando poi quando non si hanno più miglioramenti. La potatura consiste nel sostituire un sottoalbero con una foglia la cui classe è quella di maggioranza della parte potata.

Il post-pruning risulta molto più efficace del pre-pruning, tuttavia è computazionalmente più pesante.

Classificatori rule based I classificatori rule based generano modelli sotto forma di lista di regole di classificazione.
La generica struttura di una regola è:

$$Condition \Rightarrow Label$$

Label è l'attributo classe (*RHS*), mentre *Condition* è una congiunzione di attributi (*LHS*)[15].
Una regola copre un'istanza x se gli attributi di questa soddisfano *LHS* della regola e si definisce *copertura* di una regola la frazione di record che soddisfano il suo *LHS*, mentre *accuratezza* la frazione di quelli che soddisfano sia *LHS* che *RHS*.
Se in una lista di regole di classificazione ogni record è coperto al massimo da una regola, la lista si definisce *mutuamente esclusiva*, se invece ogni suo record è coperto da almeno una regola, si dice *esaustiva*; se possiede entrambe le proprietà, è equivalente ad un albero decisionale. Queste caratteristiche tuttavia possono venire perse nel caso si decidesse di semplificare la lista a causa di eccessiva grandezza e questo costringerebbe ad adottare degli adeguati provvedimenti.
Le regole devono essere ordinate secondo un criterio di priorità, dal momento che un record può essere coperto da più regole (verrà messa in atto quella di maggiore priorità); inoltre nel caso un record non sia coperto da alcuna regola, gliene deve essere assegnata una di default.

Un insieme di regole, se estratto dai dati, viene definito costruito *direttamente*, mentre se è derivato da un altro modello si dice costruito *indirettamente*.
Nel caso diretto (detto anche *sequential covering*) sono previsti i seguenti passi:

1. costruzione di una singola regola.
2. rimozione delle istanze coperte da questa, per assicurare che la prossima regola costruita non sia identica alla corrente.
3. esecuzione del pruning se necessario (si rimuove una congiunzione dall'*LHS*, si valuta poi se l'errore si riduce, in caso contrario la regola viene lasciata inalterata).

[15]Per esempio: $(BloodType = Warm)\ AND\ (LayEggs = Yes) \Rightarrow Bird$

4. aggiunta della regola al *rule set* (*decision list* se c'è un criterio di ordinamento).

5. reiterazione dell'intero procedimento.

I classificatori rule based sono un'alternativa per rappresentare un pattern, ed anch'essi come i decision tree presentano vantaggi e svantaggi, che spesso dipendono dal caso specifico.

Classificatori instance based A differenza delle altre categorie di classificatori, non producono un modello, ma utilizzano direttamente il training set per classificare nuove istanze e per questo motivo sono detti *lazy*.
I classificatori instance based sono dispendiosi in occupazione di memoria e poco performanti nel classificare nuove istanze in termini di tempo.

Classificatori bayesiani Si basano sul *teorema di Bayes*[16], in cui la *probabilità condizionata* risulta essere:

$$P(C|A) = \frac{P(A|C)P(C)}{P(A)}$$

dove *P(C)* indica la probabilità di una classe, *P(A)* quella di una combinazione di attributi.
Si classifica un record con attributi A cercando la classe C che massimizzi la probabilità $P(C|A)$, che equivale a massimizzare il prodotto $P(C|A) \cdot P(C)$, assumendo che in A i vari attributi siano statisticamente indipendenti tra loro.
Per effettuare i calcoli bisogna tenere presente che:

$$P(C) = \frac{N_C}{N} \qquad\qquad P(A|C) = \frac{N_{AC}}{N_C}$$

dove N_C è il numero di record aventi classe C, N quello totale e N_{AC} quello delle istanze aventi valore di attributi A ed appartenenti alla classe C.

[16]Noto anche come *teorema della probabilità delle cause*.

Si usa poi la *correzione di Laplace* per evitare che l'espressione sia *0* a causa di una probabilità condizionata nulla:

$$P(A_i|C) = \frac{N_{Ci} + 1}{N_C + c}$$

con c numero delle classi.
Questo tipo di classificatore ha prestazioni ottimali perché è robusto al rumore, agli attributi irrilevanti ed ai missing value, inoltre lavora in modo incrementale, richiedendo poca occupazione di memoria durante l'elaborazione, tuttavia non sempre nella realtà vale l'assunzione di indipendenza tra gli attributi in A, perciò nella pratica non si può realizzare, ma si può solo prendere in considerazione un'approssimazione del modello ideale.

Classificatori con reti neurali Una rete neurale è caratterizzata da una funzione di attivazione, da una funzione di uscita e da n ingressi ad ognuno dei quali viene associato un peso w[17]. Nei classificatori *ANN* (*Artificial Neural Network*) la funzione di attivazione è la somma dei prodotti di ogni ingresso per il relativo peso:

$$\sum_i w_i x_i$$

La funzione di uscita è realizzata in modo tale da essere discriminante della classe sulla base di una certa soglia entro cui la sommatoria deve rientrare.

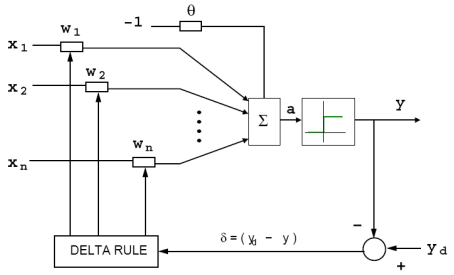

Figura 1.7: *Rete neurale*

[17]Vedere riferimento bibliografico [12].

Addestrare la rete significa trovare il giusto valore per ogni peso w_i[18]. Dal momento che:

$$w_i(t+1) = w(t)_i + \eta\delta x_i$$

dove η è il coefficiente di apprendimento e δ la differenza tra l'output ed il valore atteso.
Si procede iterando l'addestramento fintanto che:

$$w_i(t+1) = w_i(t)$$

Tale classificatore funziona se le istanze sono linearmente separabili e con due ingressi (ossia nel nostro caso due attributi).
Il modello prodotto ha la forma:

$$w_1x_1 + w_2x_2 - \vartheta > 0$$

e suddivide lo spazio in maniera lineare in 2 zone distinte.

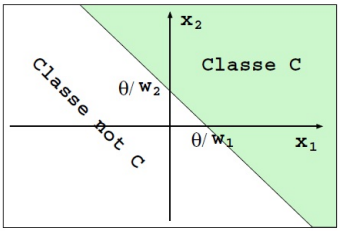

Figura 1.8: *Suddivisione lineare dello spazio di classificazione nelle reti neurali con uscita binaria*

Classificatori misti Esistono tecniche per aggregare e combinare più modelli anche generati con classificatori diversi, che permettono così di avere a disposizione più alternative e criteri di confronto; questo offre poi l'opportunità di selezionare il modello migliore o combinare le caratteristiche di modelli multipli.

[18]Le reti adatte ad un *2-class problem* sono quelle che seguono il modello del neurone binario a soglia (come il perceptron), mentre per le reti stratificate è utilizzato la *back propagation* come meccanismo di addestramento.

Tali tecniche sono[19]:

bagging
: Dato un training set di cardinalità N per ogni modello da generare campiona con reimbussolamento n istanze, genera il modello usando l'algoritmo di Machine Learning e lo salva. In fase di classificazione si sottopone l'istanza a ciascun modello e, secondo il criterio di elezione, le viene assegnata la classe più votata, attribuendo ad ogni voto egual peso.

boosting
: A differenza del bagging, i modelli vengono generati in successione ed ad ogni iterazione quello corrente è influenzato dalle prestazioni di quelli precedenti. Durante la classificazione i voti non sono uguali ma sono pesati in base alla bontà del modello votante[20].

stacking
: Viene usato per combinare modelli prodotti con tipologie diverse di classificazione; fa affidamento ad un *metalearner*, ossia un algoritmo di Machine Learning usato per scoprire come combinare al meglio l'output dei modelli a disposizione. Il metalearner genera il metamodello o *level-1 model*, mentre i *level-0 models* sono i modelli di base. Al livello 1 un'istanza ha tanti attributi quanti sono i modelli da combinare, poi in fase di classificazione essa viene data in input ai modelli di livello 0, ed infine tutte le predizioni vengono riunite in un'osservazione che diverrà l'input del metamodello, il quale restituirà come output la predizione finale. Parte del dataset viene usata come training set per addestrare i classificatori di base mentre l'altra viene riservata per il metalearner; una volta generati i modelli di livello 0, vengono testati sulla parte restante, creando in questo modo un training set per il metalearner in cui gli attributi sono le predizioni dei classificatori base e l'etichetta la reale classe.

[19] Vedere riferimento bibliografico [7].

[20] Per varianti del boosting vedere l'appendice C.1.4 - *Classificatori: AdaBoost.M1.*

Clustering

Il clustering consiste nel suddividere un dataset in gruppi, detti *cluster*, tali che tutti gli elementi che fanno parte del medesimo gruppo abbiano caratteristiche simili e siano diversi dai gruppi restanti; con il termine *clustering* ci si riferisce invece all'insieme di tutti i gruppi. Il criterio di similarità varia a seconda del dominio: uno tra i più comuni è quello della distanza.
A differenza della classificazione, i dati vengono analizzati usando solamente le informazioni contenute in essi e questo è estremamente utile quando non si conoscono ancora i criteri o la classe di raggruppamento.

Il clustering è utilizzato per diversi scopi:

understanding
: Raggruppare oggetti omogenei aiuta a comprendere un determinato ambito.

summarization
: Con domini ad elevata cardinalità è possibile ricondurre un intero cluster ad un unico elemento scegliendo il più rappresentativo.

Esistono vari tipi di clustering:

gerarchico
: I cluster sono organizzati secondo una struttura gerarchica.

partizionale
: I cluster non possono essere sovrapposti.

esclusivo
: Ogni elemento appartiene ad un solo cluster.

overlapping
: Un elemento può appartenere a più cluster.

fuzzy
: Un elemento appartiene ad ogni cluster ed ad esso è assegnato un livello di appartenenza.

complete
: Tutto l'insieme è clusterizzato.

parziale
: Solo un sottoinsieme del dataset è soggetto a clustering.

Vi sono inoltre differenze anche a livello di cluster:

well-separated
: Ogni elemento di un cluster è più simile o vicino a ciascun altro elemento del medesimo cluster, rispetto a qualsiasi altro cluster.

center-based
: Ogni elemento di un cluster è più vicino al centroide di un cluster rispetto a quelli degli altri cluster. Nel caso di insiemi numerici il centroide è la media degli elementi di un cluster, negli altri casi si parla di mediode, ossia il punto più rappresentativo.

contiguity-based
: Tutti gli elementi appartenenti ad un cluster sono collegati tra loro, ma non quelli esterni ad esso (utile per dati usati per la definizione di grafi).

density-based
: Un cluster è una regione ad alta densità circondata da altre regioni a bassa densità. Tale caratteristica permette di riconoscere anche cluster con forma convessa.

Analisi delle associazioni

Consiste nel trovare regole associative che mettono in relazione l'occorrenza di uno o più item in funzione di altri all'interno di una transazione, intesa come insieme di *item* raggruppati secondo un certo criterio; le regole associative sono dunque utili per trovare relazioni complesse nei dati.
Il problema può essere rappresentato in forma tabellare, dove ad ogni identificativo della transazione corrisponde una serie di item relativa a quest'ultima.
Un *itemset* è un insieme di item, *k-itemset* se ci si riferisce ad itemset

con k item; una transazione T contiene un itemset X se questo è un sottoinsieme dell'insieme di item associati a questa.
Si definisce *supporto di un itemset* la frazione di transazioni sul totale che la contengono, inoltre si ha un *itemset frequente* se esso ha un supporto superiore ad una certa soglia *supmin*.
Una regola associativa è un'implicazione avente forma:

$$X \Rightarrow Y$$

dove $X, Y \subseteq I$ e $X \cap Y = \emptyset$, con I insieme di tutti gli itemset.

La qualità di una regola associativa è valutata tramite:

supporto

$$support(X \Rightarrow Y) = \frac{|X \cup Y|}{|T|}$$

ossia la percentuale di transazioni che contengono sia X che Y sul totale delle transazioni esistenti.

confidenza

$$confidence(X \Rightarrow Y) = \frac{|X \cup Y|}{|X|}$$

che invece indica le transazioni che contengono sia X che Y rispetto alle transazioni che contengono almeno X.

Definire un problema di analisi delle associazioni vuol dire trovare regole associative con supporto maggiore di *minsup* e confidenza maggiore di *minconf* in un certo dataset, considerato come un insieme di transazioni.
Un approccio *brute force* è da escludersi a causa dell'eccessiva elaborazione che sarebbe richiesta; gli attuali algoritmi scompongono il problema in due sottofasi:

1. *frequent itemset generation*
   Si trovano gli itemset frequenti, cioè quelli aventi supporto maggiore di *minsup*.

2. *rule generation*
   Dagli itemset frequenti trovati precedentemente si estraggono regole associative con confidenza superiore a *minconf*.

Poiché da k item possono essere generati 2^k itemset, segue che si avrà un albero degli itemset molto grande e conseguentemente poco gestibile se non si attuano adeguate strategie di esplorazione, quindi anche in questo caso un approccio di tipo *brute force* è da accantonare.
Le tecniche principali adottate sono:

- ridurre il numero di itemset da esplorare[21]

- ridurre il numero di confronti, tramite utilizzo di opportune strutture dati

Predizione numerica

La predizione numerica differisce dalla classificazione solo nel fatto che la classe è un attributo di tipo numerico invece di essere nominale.
Poiché occorre lavorare con attributi numerici, per stabilire il valore della classe si usa la *regressione lineare*. L'idea di base è esprimere la classe x come combinazione lineare degli attributi a ciascuno dei quali viene associato un peso:

$$x = w_0 + w_1 a_1 + ... + w_k a_k$$

dove w sono i pesi ed a gli attributi.
L'obiettivo è determinare il peso degli attributi in base al training set; per ottenere buoni risultati occorre che quest'ultimo abbia un numero di elementi molto superiore rispetto a quello degli attributi.

1.3.5 Evaluation

Una volta costruito un modello, in fase di Evaluation lo si applica con lo scopo di valutarne la bontà[22]. I criteri di valutazione ovviamente dipenderanno dal tipo di modello, pertanto anche in questo caso occorrerà fare delle distinzioni.

[21] Vedere appendice C.3 - *Regole associative: Algoritmo Apriori.*
[22] Vedere riferimento bibliografico [18].

Classificazione

La prima misura per valutare la qualità di un modello di classificazione è l'*accuratezza* definita come:

$$accuratezza = \frac{[\#\ istanze\ classificate\ correttamente]}{[\#\ totale\ istanze]}$$

Questo valore sarà compreso nell'intervallo *[0,1]* oppure può essere espresso in percentuale.
Il *tasso di errore* o semplicemente *errore* è la stima complementare dell'accuratezza:

$$errore = 1 - accuratezza$$

Per avere risultati attendibili bisogna calcolare accuratezza ed errore su dati sconosciuti al modello; ciò comporta quindi di avere la disponibilità di un test set oltre al training set.
Se si ha a disposizione un unico dataset si possono utilizzare diverse tecniche:

holdout
: Si divide il dataset in parte in training set (in genere i $^2/_3$) ed in parte in test set (il restante $^1/_3$). È *stratificato* quando si fa in modo che le classi siano egualmente rappresentate.

bootstrap
: Si costruisce il training set campionando con reimbussolamento n elementi da un dataset; in questo modo alcuni elementi saranno presenti più volte nel training set, mentre altri non saranno mai considerati ed andranno a fare parte del test set.
Per tenere conto di questo, l'errore viene calcolato come:

$$err = 0.632 \cdot err_{test} + 0.368 \cdot err_{training}$$

k cross validation
: L'intero dataset viene diviso in k sottoinsiemi; una volta a turno uno di essi viene usato come test set, mentre gli altri come training set. L'operazione viene rieseguita fino a quando tutti i sottoinsiemi sono stati utilizzati come test set.

L'accuratezza globale viene calcolata come:

$$acc_{kcrossval} = \frac{[\#\ classificazioni\ corrette\ totali]}{[\#\ campioni\ iniziali]}$$

Sebbene sia vivamente sconsigliato, talvolta occorre utilizzare il training set come test set; definendo *errore di risostituzione* quello sul training set ed *errore di generalizzazione* quello sul test set, si possono adottare 2 approcci:

ottimistico

$$err_{ris} = err_{gen}$$

pessimistico

$$err_{gen} = \frac{n_e + n_l p}{N_t}$$

dove n_e è il numero di errori sul training set, n_l il numero di nodi del decision tree, p è la penalità associata a ciascun nodo (in genere *0.5*) e N_t la cardinalità del training set.

Avendo un modello M testato su un dataset di cardinalità N ed ottenendo un'accuratezza A, se si prova a testare M su altri test set di dimensioni differenti, per sapere come cambia A occorre trovare un *intervallo di confidenza*, ossia appunto il range entro cui, con una certa probabilità, varia l'accuratezza.
Si può modellare l'output di un classificatore tramite una *distribuzione binomiale*.[23] Se la dimensionalità del dataset è grande, per il *teorema del limite centrale* è possibile approssimare la distribuzione binomiale con quella normale, diventando la *distribuzione di probabilità dell'accuratezza*; in base a quest'ultima considerazione e tralasciandone la

[23] È possibile in quanto possiede le caratteristiche di un *esperimento di Newton*:
- output espresso sotto forma di successo o insuccesso
- un output corrisponde ad un esperimento e ciascuno è indipendente dagli altri
- la probabilità di successo p è la stessa per tutti gli esperimenti

dimostrazione matematica[24], si giunge alla seguente formula per il calcolo dell'*intervallo di confidenza*:

$$p = \frac{2 \cdot N \cdot acc + Z^2_{\alpha/2} \pm Z_{\alpha/2}\sqrt{Z^2_{\alpha/2} + 4 \cdot N \cdot acc - 4 \cdot N \cdot acc^2}}{2\left(N + Z^2_{\alpha/2}\right)}$$

dove p è la reale accuratezza che si vuole stimare.
Il valore di $Z_{\alpha/2}$ va dunque imposto in base alla probabilità di confidenza $1 - \alpha$ richiesta, secondo quanto riportato in tabella.

$\boldsymbol{1-\alpha}$	*0.99*	*0.98*	*0.95*	*0.9*	*0.8*	*0.7*	*0.5*
$\boldsymbol{Z_{\alpha/2}}$	2.58	2.33	1.96	1.65	1.28	1.04	0.67

L'accuratezza dà una stima generale sulla qualità di classificazione di un modello, ma non permette di valutare il comportamento di quest'ultimo su una singola classe[25].
Per questo motivo sono state introdotte altre misure: *precision*, *recall* e *f-measure*.
Innanzitutto, avendo classe con due etichette, P (positivi) e N (negativi), definiamo:

TP, true positive
: elementi di classe P classificati correttamente

FP, false positive
: elementi di classe N classificati come P

TN, true negative
: elementi di classe N classificati correttamente

FN, false negative
: elementi di classe P classificati come N

[24]Si modella il processo di classificazione come un processo di Bernoulli.

[25]Per esempio, in un *2-class problem*, avendo un dataset a cardinalità 100 con 90 elementi di classe A e 10 di B, un classificatore potrebbe classificare correttamente tutti gli A e sbagliare completamente i B, raggiungendo tuttavia un'accuratezza dello *0.9*; questo risultato darebbe una falsa idea sulle prestazioni, che in realtà sono tutt'altro che accettabili.

Specificato ciò, possiamo ridefinire l'accuratezza come:

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

Inoltre ora possiamo esprimere le nuove misure:

precision (precisione; istanze appartenenti ad una classe classificate correttamente, rapportate al totale delle istanze classificate come tali):

$$precision(P) = \frac{TP}{TP + FP} \qquad precision(N) = \frac{TN}{TN + FN}$$

recall (recupero; istanze appartenenti ad una classe classificate correttamente, rapportate al totale delle istanze appartenenti realmente a tale classe):

$$recall(P) = \frac{TP}{TP + FN} \qquad recall(N) = \frac{TN}{TN + FP}$$

f-measure (media armonica della relativa recall e precision):

$$fmeasure(P) = \frac{2 \cdot (recall(P) \cdot precision(P))}{recall(P) + precision(P)}$$

$$fmeasure(N) = \frac{2 \cdot (recall(N) \cdot precision(N))}{recall(N) + precision(N)}$$

Tutti i vari casi possono essere riassunti tramite la *matrice di confusione*:

		classe predetta	
		sì	**no**
classe reale	**sì**	TP	FN
	no	FP	TN

Da essa[26] è possibile ricavare tutte le misure descritte precedentemente; inoltre il numero di istanze classificate correttamente è dato dalla somma dei valori della diagonale principale.

Altri criteri importanti per valutare la bontà di un classificatore sono:

kappa statistic
: Stima la bontà di un classificatore mettendolo a confronto con un classificatore casuale; in quest'ultimo le classi vengono attribuite per ogni riga della matrice seguendo una proporzione prestabilita, poi ne vengono cambiati i valori, ma mantenendo la somma di righe e colonne inalterate.
La misura di bontà si determina nel seguente modo:

$$kappa_{statistic} = \frac{[miglioramento\ ottenuto]}{[miglioramento\ classificatore\ perfetto]}$$

dove

miglioramento ottenuto =
$[istanze\ corrette] - [istanze\ corrette\ classif.\ casuale]$

miglioramento classificatore perfetto =
$[ist.\ corr.\ classif.\ perfetto] - [ist.\ corr.\ classif.\ casuale]$

ROC curve
: La curva ROC (*Receiver Operating Characteristic* o *Relative Operating Characteristic*) è un grafico per classificatori binari che mostra in ascissa la percentuale di veri positivi ed in ordinata quella dei falsi positivi. Offre una visualizzazione del tasso di errore e soprattutto dell'andamento della precisione della classificazione.

Clustering

Per valutare invece la qualità di un modello di clustering vengono usati due tipi di indici:

[26] In questo caso specifico, valida per *2-class problem*.

external index
Valuta quanto le etichette dei cluster corrispondano a quelle fornite.
Ci sono diversi approcci:

classification oriented
Viene misurata la tendenza di un cluster a contenere elementi di un'unica classe, usando gli indici tipici della classificazione: *precision*, *recall* ed *entropia*.

similarity oriented
Si misura la tendenza di due oggetti ad essere nello stesso cluster e di conseguenza nella stessa classe.

indice di Rand
È il più usato, ed occorre innanzitutto calcolare prima le seguenti quantità:

- a, numero di coppie che sono nello stesso cluster e nella stessa classe
- b, numero di coppie che appartengono allo stesso cluster ma non alla stessa classe
- c, numero di coppie che appartengono alla stessa classe ma non allo stesso cluster
- d, numero di coppie in cluster e classi diversi

Quindi si calcola l'*indice di Rand* come:

$$Rand = \frac{a+d}{a+b+c+d}$$

internal index
Valutano la bontà di un clustering senza l'ausilio di informazioni supplementari. Questo indice è influenzato da alcune caratteristiche che deve avere un buon clusterizzatore:

compattezza o *coesione*; membri di uno stesso cluster devono essere i più vicini possibili tra loro.
Viene calcolata come:

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

dove x è il centroide e c_i un punto del cluster.
Altrimenti:

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

dove m_i è la dimensione del cluster.

separazione; i cluster devono essere ben distinti tra loro.
Viene usato il quadrato della media delle distanze tra ciascun centroide c_i e la media degli altri punti c:

$$SSB = \sum_{i=1}^{k} (m_i \times dist(c_i, c)^2)$$

Maggiore è il valore di *SSB* e meglio sono separati i cluster.

indice di silhouette; combina stime sia di separazione che di coesione.
Permette di valutare quanto un punto contribuisce alla qualità di un cluster.
Il calcolo viene eseguito in 3 passi:

1. viene calcolata la distanza media per l'elemento i da tutti gli altri elementi interni; tale valore viene nominato a_i.
2. per ogni altro cluster, si calcola la distanza tra a_i ed i suoi elementi; viene preso il minimo tra questi valori, denominandolo b_i.
3. il *coefficiente di silhouette* viene espresso come:
$$s_i = \frac{b_i - a_i}{max(a_i, b_i)}$$
Esso varia nell'intervallo *[-1,1]*; se risulta negativo indica la presenza di oggetti di altri cluster più vicini rispetto ad alcuni del medesimo.

Regole associative

Per valutare la bontà di una determinata regola associativa $X \Rightarrow Y$, si può innanzitutto calcolare *supporto* e *confidenza*, valutando poi quanto

essi soddisfino le attese; tuttavia utilizzare solo queste due misure può essere limitativo, in quanto esse possono stimare come buone, regole che racchiudono pattern ovvi e trascurare invece altre relazioni interessanti.
Per questo motivo sono stati sviluppati altri criteri di valutazione:

lift o *fattore di interesse*

$$LIFT = \frac{conf(A \Rightarrow B)}{supp(B)}$$

ovvero il rapporto tra confidenza di una regola ed il supporto del conseguente.
I principali limiti di questa misura derivano dal fatto che i due termini hanno ugual peso, per cui può capitare che abbiano un buon $LIFT$ anche regole con confidenza e supporto del conseguente entrambi bassi.

correlazione

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

I vari f sono i valori che compaiono dentro la seguente *tabella di contingenza* relativa a due itemset A e B.

	$\boldsymbol{B}$	$\overline{\boldsymbol{B}}$	
$\boldsymbol{A}$	f_{11}	f_{10}	f_{1+}
$\overline{\boldsymbol{A}}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

- f_{11}, numero di volte in cui A e B compaiono insieme
- f_{01}, numero di volte in cui compare B ma non A
- f_{10}, numero di volte in cui compare A ma non B
- f_{00}, numero di volte in cui non compare né A né B
- f_{1+} è il supporto di A ed analogamente per gli altri f_{x+}

La correlazione varia tra *-1* ed *1*; quando vale *0* denota variabili statisticamente indipendenti.
Dal momento che questa misura pesa allo stesso modo coassenza e copresenza di un elemento di una transazione, il suo limite risiede nel fatto che può capitare che coppie che appaiono poco insieme abbiano un buon coefficiente.

Attualmente esistono altri tipi di misure meno importanti; generalmente il consiglio da seguire è di cercare di comprendere il dominio applicativo con cui si tratta e scegliere di volta in volta quella che si adatta meglio.

Predizione numerica

Per la predizione numerica, come per la classificazione, è meglio avere training set e test set indipendenti.[27]
A differenza della classificazione, l'output non viene distinto tra classe corretta o sbagliata, perciò occorre usare altri indici.
Il più usato è la *Sum of Squared Error* (*SSE*):

$$SSE = \sum_{i=1}^{n}(x_i - \sum_{j=0}^{k} w_j a_{ji})^2$$

L'obiettivo è avere un valore di *SSE* il più basso possibile.

1.3.6 Deployment

L'estrazione della conoscenza rappresentata da un modello, non è la fine del processo di Data Mining; occorre che questa venga espressa e rappresentata in una maniera adeguata per l'utente finale affinché quest'ultimo riesca ad utilizzarla a proprio vantaggio.
L'obiettivo conclusivo potrà quindi consistere nella generazione di un report o nella creazione di un software che traduca, esponga e gestisca le informazioni apprese tramite l'applicazione del modello.
La scelta del prodotto è dettata dai requisiti.

[27] Valgono le stesse considerazioni fatte nella sezione dedicata all'evaluation dei modelli di classificazione.

Capitolo 2

Previsione degli abbandoni per un caso aziendale

In questo capitolo verrà analizzato un problema concreto di Data Mining: il *churn* della clientela nel caso di un'importante azienda. Si cercherà di prevedere tale fenomeno utilizzando il metodo CRISP-DM descritto nel precedente capitolo.

2.1 Introduzione al problema

In ogni azienda, indipendentemente dal settore, gli abbandoni sono una caratteristica comune sia che queste imprese producano beni e sia nel caso eroghino servizi; il procedimento di risoluzione di tale problema rientra nell'ambito del *Customer Relationship Management* ed è definito *churn analysis*[1].
L'abbandono, nella maggior parte dei casi, fa parte del processo naturale e comune di riciclo della clientela, non è quindi da prendere sempre in termini strettamente negativi, come verrebbe spontaneo pensare; l'importante è che i nuovi clienti acquisiti compensino almeno quelli che hanno deciso di abbandonare.

[1]Letteralmente *churn* significa agitare o scombussolare. In senso figurato è l'atto di un elemento di passare da un insieme ad un altro; nel nostro ambito è usato per indicare l'abbandono della clientela di un'azienda per normale ricambio o per malcontento.

Per le aziende sarebbe di vitale importanza poter sapere quanti e quali clienti sono in procinto di abbandonarle, per cercare di recuperarli tramite adeguati provvedimenti prima che essi lascino definitivamente tali aziende. Per un'impresa infatti costa molto meno trattenere un cliente rispetto a riuscire ad acquisirne di nuovi.
Parecchie aziende sono quindi molto interessate a soluzioni software che siano in grado di prevedere in maniera automatica questo fenomeno che riguarda i loro clienti abituali, per evitare ovviamente ingenti perdite economiche e per mantenere il predominio sul mercato; per questo motivo molte aziende nel settore dell'ICT si sono dedicate a produrre soluzioni software in questo campo molto complesso, dove per ora l'esigua offerta non riesce a coprire la massiccia domanda di mercato.
Ora si cercherà di affrontare il problema della previsione degli abbandoni seguendo la metodologia CRISP-DM illustrata nel precedente capitolo, procedendo fase per fase.

2.2 Business Understanding

La maggior parte dei sistemi in circolazione per condurre analisi su dati a livello manageriale è costituita da Data Warehouse; i motivi per cui ancora il Data Mining non è particolarmente diffuso sono soprattutto organizzativi ed economici:

- è una disciplina abbastanza recente e molto complessa, questo porta ad avere un alto indice di rischio.

- i sistemi di Data Mining in circolazione devono essere sviluppati ad hoc per un'azienda, ossia occorre crearli ogni volta da capo quando devono essere applicati ad altre aziende o per casi differenti da quello corrente, questo influisce sui costi di sviluppo che sono conseguentemente molto elevati.

- necessita di personale qualificato; la complessità della disciplina richiede che sia utilizzata da persone competenti, questo vuol dire per un'azienda dover procurarsi esperti assai costosi.

In sintesi, l'azienda dovrebbe investire in qualcosa di veramente costoso, per avere poi un prodotto ad alto rischio di affidabilità, tuttavia

riuscire, come nel nostro caso, ad individuare i clienti in procinto di abbandono attira molto le imprese perché eviterebbe pesanti perdite economiche.
Tra gli obiettivi che ci poniamo c'è dunque quello di studiare un procedimento che alla fine sarà concretizzato da un sistema software che:

- sia più generico possibile e riesca dunque ad applicare lo stesso procedimento ad ogni azienda che ne ha bisogno.
- sia di facile utilizzo, anche per utenti non esperti.
- produca risultati soddisfacenti e potenzialmente affidabili.

Questo renderebbe il software alla portata anche delle PMI[2] per diversi motivi: sia a livello economico, perché verrebbero abbattuti gli elevati costi che hanno contraddistinto sempre questo genere di applicazione, dovuti sia ai tempi di sviluppo, che risultano spesso lunghi a causa della reimplementazione di tutto il sistema, e sia a livello pratico, in quanto può essere usato anche da un utente medio.

Dopo questa breve introduzione, come già accennato, vogliamo ora prevedere gli abbandoni per un caso concreto: una grossa impresa attualmente attiva ed affermata nel mercato da molto tempo; su tale caso si baserà e prenderà sviluppo tutto il procedimento di risoluzione e cercheremo alla fine di ottenere un buon modello che riesca ad individuare per tale azienda i clienti a rischio di abbandono.

2.2.1 Scenario di mercato

L'azienda per cui ci accingiamo a prevedere gli abbandoni opera nel settore fashion e quindi produce e distribuisce beni di consumo.
Innanzitutto occorre precisare che per cliente non intendiamo la singola persona che è posta al termine della catena di distribuzione, ma i punti vendita che in genere effettuano acquisti in blocco per rifornire il negozio. In una catena molto affermata possono esserci sia negozi appartenenti al franchising, sia negozi che vendono merce mista;

[2]Piccole e Medie Imprese.

ovviamente questi ultimi sono quelli a maggior rischio di abbandono perché soffrono della concorrenza di prodotti di altre marche.
Risulta quindi molto gravosa la perdita anche di un solo cliente dato che ognuno di essi implica una considerevole quantità di merce acquistata periodicamente, al contrario per esempio di aziende o negozi che effettuano vendita diretta e che quindi per clienti hanno singoli utenti finali i quali ovviamente acquistano quantità di merce per un valore molto basso.

2.2.2 Task richiesto dal problema

La previsione degli abbandoni è un classico problema di classificazione: bisogna stabilire per ogni cliente se esso appartenga alla classe degli abbandoni o dei fedeli. Difatti da questo ultimo presupposto ci si accorge che la predizione numerica non è adatta poiché occorre catalogare i clienti; inoltre tali categorie le conosciamo a priori e pertanto si escludono tutte le tecniche non supervisionate (clustering e regole associative).
In conclusione, l'intero procedimento di risoluzione andrà improntato come un problema 2-class di classificazione.

2.3 Data Understanding

La prima questione che occorre affrontare è l'analisi dei dati che si hanno a disposizione; in base ad essa potremo poi stimare quali sono i parametri che ci occorrono ed i valori da prendere in considerazione. C'è stata fornita una quantità elevata di dati, questo rende l'analisi molto complessa e lunga, anche perché gli strumenti per analizzarli spesso impiegano molto tempo per eseguire le interrogazioni necessarie a comprenderne le caratteristiche. È quindi di vitale importanza interpretare il meglio possibile queste informazioni, perché sarebbe poi difficoltoso ricavare in maniera empirica i parametri che ci occorreranno nelle fasi successive per gli algoritmi che si useranno, dato che l'unico modo è provare tutte le combinazioni (o comunque anche con quest'ultimo approccio un'accurata analisi preliminare può permette di eseguire simulazioni più mirate e di escludere la maggior parte di

valori non idonei).

2.3.1 Struttura e contenuto del database

I dati inizialmente[3] si riferivano ad un periodo temporale di circa 10 anni, che andava da giugno del 2001 (la fattura più vecchia risaliva al 15-06-2001) fino a dicembre del 2011 (per la precisione 01-12-2011) comprendendo 2.342.580 fatture, alle quali si riferivano ben 27.084.300 dettagli e 1.908.350 osservazioni per i pagamenti.
Per la precisione, le principali tabelle[4] di nostro interesse sono strutturate nella maniera seguente:

clienti(cliente_fatturazione)

fatture(id_fattura, cod_fattura, anno_documento, id_agente, id_causale, data_fattura, id_cliente_fatturazione)

fatture_dettagli(id_fattura, cod_fattura, anno_documento, riga_fattura, id_prodotto, id_agente, id_causale, fatturato, costo_acquisto, sconto, mot_id, caa_id, cau_id, fsa_id)

fatture_pagamenti(id_fattura, nr_rata, data_scadenza, data_pagamento, fatturato_inattesa, fatturato_pagato, insoluto_recuperato, insoluto, insoluto_perso)

Tali tabelle sono contenute in un apposito schema *ds* dedicato ai dati sorgenti ed a cui si cercherà di accedere sempre in sola lettura per non alterare la struttura originale del database.
I clienti catalogati erano 38.718, suddivisi in 2 categorie: *Wholesale* e *Retail*. I primi sono punti di vendita di merce appartenente ad un numero considerevole di marche fra quelle in circolazione, mentre i secondi sono i rivenditori, ossia negozi di proprietà che vendono esclusivamente i prodotti di tale azienda. Ovviamente la gran parte dei negozi appartiene alla categoria Wholesale e sono quelli a cui siamo

[3] I dati sono stati successivamente revisionati (vedere più avanti).

[4] La tabella *clienti* qui riportata in realtà è stata ricostruita successivamente dalla tabella *fatture*; la tabella contenente i clienti era presente solo in una vecchia versione del database.

interessati: dai dati a nostra disposizione essi risultano 38.040 sul totale dei clienti, mentre l'esigua parte restante sono quindi clienti di tipo Retail.

Su tali dati è stata in seguito effettuata una pulizia preliminare[5] (secondo criteri semantici) per permettere una loro corretta e migliore interpretazione, poiché oltre ad essere eccessivi, facevano emergere comportamenti e valori statistici anomali difficili da catalogare. Si è scelto di tenere i soli clienti del mercato italiano ed appartenenti alla categoria dei Wholesale, escludendo quelli del franchising; inoltre sono state eliminate tutte le intercompagnie e le interaziende con le quali l'impresa è legata ed ha rapporti di vario genere. Oltre a ciò si sono tenute esclusivamente le fatture riguardanti gli acquisti, che sono le uniche che ci interessano.
Sono stati considerati alla fine solo 7.103 clienti con un totale di 367.000 fatture e con un complessivo di 4.244.940 dettagli e 633.152 pagamenti, per uno storico che va dall' 08-01-2002 al 30-09-2011.
Come si può notare, la quantità di dati è stata drasticamente diminuita e questo, oltre a permetterci di effettuare un'analisi efficace, consente anche di avere tempi di elaborazione ridotti, che altrimenti sarebbero stati insostenibili; tuttavia la cardinalità finale dei dati, seppur diminuita, rimane comunque considerevole per le massicce elaborazioni che saranno richieste.

2.3.2 Analisi dei dati

Cerchiamo innanzitutto di analizzare i dati interrogando il database, in modo da renderci conto delle caratteristiche e dell'andamento nel tempo di alcuni fenomeni utili per avere un quadro generale dei dati che abbiamo di fronte[6]:

- numero di acquisti per ogni anno
- numero di clienti in relazione al numero di acquisti effettuato

[5] Ad opera di terzi, responsabili del prelevamento e strutturazione di tali dati.
[6] Si prende spunto dal lavoro eseguito dal prof. Moro, docente presso l'Università di Bologna (vedere riferimento bibliografico [11]).

- longevità della clientela
- distanza media di acquisto
- customer retention
- clienti persi ed acquisiti ogni anno
- andamento di fatturato e margine
- numero di clienti in base al fatturato

Come si può vedere, l'analisi si baserà principalmente su due tipi di tendenza: quella relativa alla periodicità del fenomeno (come frequenza e distanza nel tempo) e quello relativo alla sua quantità (per esempio la stima del fatturato o del numero di clienti o di acquisti).

Numero di acquisti per ogni anno Analizziamo innanzitutto l'andamento delle vendite per capire l'ammontare annuo medio degli acquisti effettuati; in questa analisi, come per la maggior parte delle prossime, escludiamo il 2011, poiché non abbiamo i dati completi dell'intero anno e quindi il confronto risulterebbe inaffidabile.
Per ogni acquisto (anche cumulativo di più prodotti) è emessa una singola fattura, pertanto il numero di acquisti sarà dato dalla quantità di fatture.

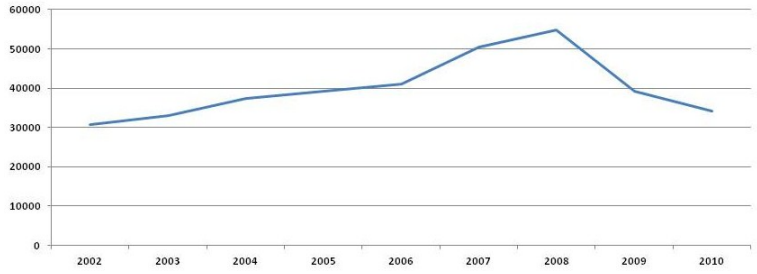

Figura 2.1: *Numero di acquisti ogni anno*

Si osserva che dal 2002 gli acquisti sono andati in crescendo, partendo

da un minimo di circa 30.000 ed arrivando ad un apice di quasi 50.000 nel 2007, da tale anno poi sono invece scesi, fino a ritornare nell'ultimo anno a quasi 34.000.
Ovviamente frequenza e volume di acquisto incidono su questo grafico e variazioni di essi potrebbero avere ripercussioni sull'andamento di questo fenomeno tali da trarre in inganno: i punti vendita nel corso del tempo (o quelli nuovi che si sono aggiunti) potrebbero infatti adottare una politica di acquisto costituita da acquisti più rari ma più consistenti.

Numero di clienti in relazione al numero di acquisti effettuato Se consideriamo il numero di clienti che ogni anno fanno un certo numero di acquisti e dividendo quest'ultima quantità per fasce possiamo dedurre quale sia più o meno la media annua di acquisti.

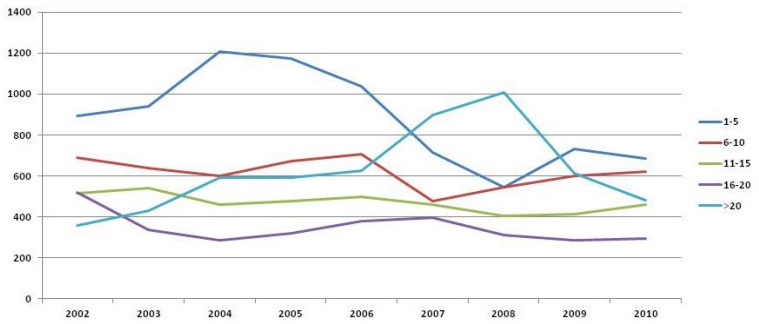

Figura 2.2: *Clienti per numero di acquisti*

Vediamo che la clientela si suddivide principalmente o tra chi fa pochi acquisti (da 1 a 5), che costituisce la maggioranza nei primi anni (poi bruscamente in calo), e chi ne fa più di 20 all'anno.

Longevità della clientela Per osservare la durata temporale media della clientela, si considera per ogni cliente il periodo tra il primo acquisto effettuato e l'ultimo: chiamiamo tale misura *longevità*.
Per misurare tale valore basta ovviamente guardare la data della fattura più vecchia di un cliente e di quella più recente.

Considerando questa misura, si ottiene una media di 1145 giorni, ossia poco più di 3 anni.

Distanza media di acquisto Un'altra osservazione importante riguarda la distanza media tra un acquisto e l'altro che ci dà l'idea della costanza con cui tali acquisti vengono effettuati.
Considerando tutti i clienti, si riscontra una periodicità media di 60 giorni (59 se si esclude il 2011), cioè 2 mesi.

Customer Retention Un'analisi significativa è riferita ai clienti che, effettuato un acquisto in un determinato anno, ne hanno effettuato uno anche nell'anno precedente; questa tendenza, detta *Customer Retention*, mette in luce la volontà di un cliente di continuare o meno ad acquistare.

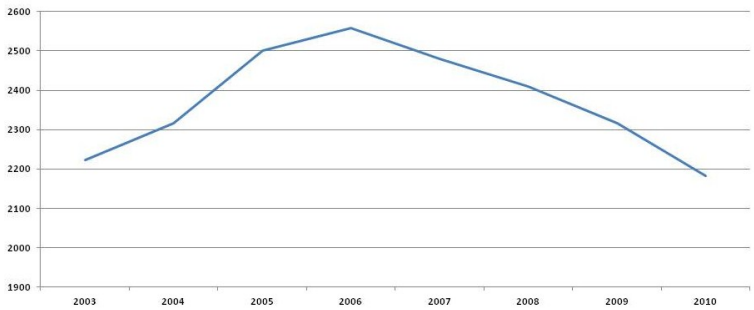

Figura 2.3: *Customer Retention*

Notiamo che questo fenomeno è aumentato nei primi anni fino al 2006, per poi diminuire lentamente.

Clienti persi ed acquisiti ogni anno Occorre poi stimare il fenomeno del riciclo della clientela, utile per il problema che dovremo affrontare.
Preso in esame un determinato anno, si considerano come acquisiti i clienti che negli anni precedenti non hanno effettuato alcun acquisto,

mentre come persi tutti quei clienti che poi non hanno più comprato negli anni successivi; si procede quindi a calcolarne il numero per ognuna delle due categorie. È ovvio che la stima sugli acquisiti sarà meno affidabile più ci si avvicina all'estremo inferiore dello storico, ossia il 2002, mentre al contrario il calcolo dei persi potrebbe essere non veritiero per gli anni prossimi all'estremo superiore rappresentato dal 2011, poiché in entrambi i casi non si può sapere cosa sia avvenuto oltre i dati conosciuti.

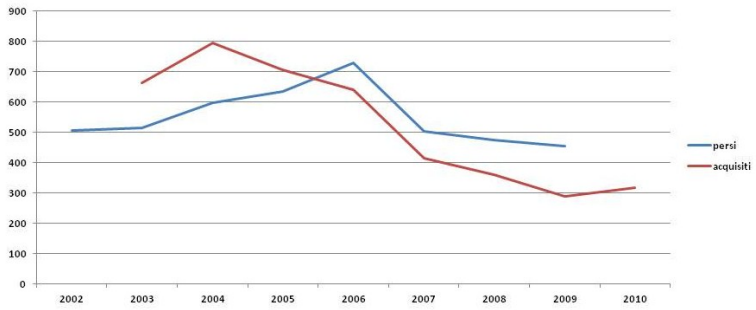

Figura 2.4: *Clienti persi ed acquisiti ogni anno*

Si può notare come nei primi anni vi sia stata una grande quantità di clienti acquisiti, mentre recentemente il fenomeno si è invertito, riscontrando più clienti persi rispetto ai primi. Per avere una reale idea del danno o del guadagno bisognerebbe capire per quanto fatturano quelli persi e per quanto quelli acquisiti.

Andamento di fatturato e margine Osservando i dati si nota un andamento del fatturato sempre in crescita, a parte l'ultimo anno.
Inoltre, se introduciamo il *margine* definendolo come la differenza tra fatturato e costo d'acquisto[7], possiamo renderci conto dell'effettivo

[7]Bisogna precisare che la formula utilizzata offre una stima quantitativa del margine, ma se si vuole avere un indice percentuale in relazione al fatturato, occorre usare il rapporto del *margine commerciale* definito come:

$$\frac{fatturato - costo\ d'acquisto}{fatturato} \cdot 100$$

guadagno dell'azienda[8].

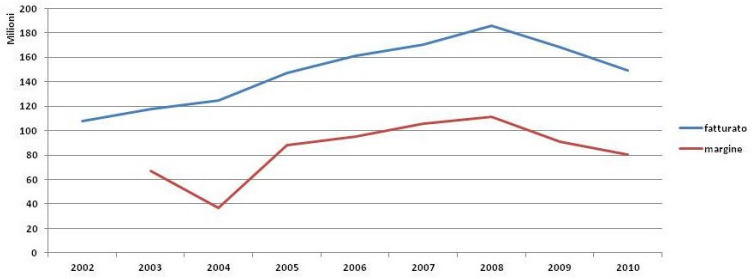

Figura 2.5: *Fatturato e margine*

Numero di clienti in base al fatturato Se guardiamo il numero di clienti in base al fatturato degli acquisti, si nota che all'inizio vi erano molti clienti ma con una quantità di acquisti bassa; questa tendenza è poi calata bruscamente mentre pian piano sono aumentati i clienti che effettuano acquisti per la fascia più alta di fatturato.

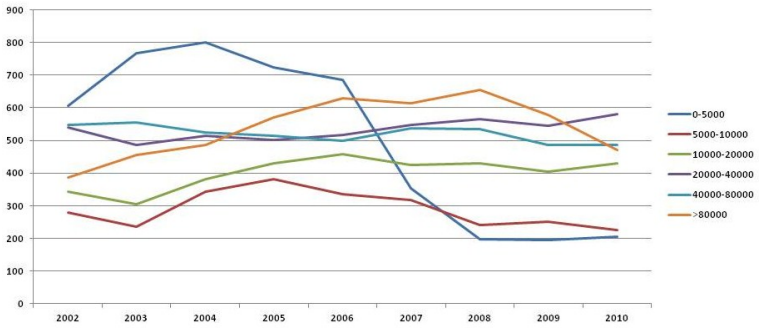

Figura 2.6: *Clienti in base al fatturato*

[8]Il margine in questo caso è stato calcolato escludendo i valori mancanti di costo d'acquisto (vedere la sezione più avanti).

2.3.3 Qualità dei dati

Sebbene sia stata effettuata una sostanziale revisione del database, sono stati comunque riscontrati alcune incongruenze nei dati e valori importanti non presenti: bisogna quindi prendere una decisione su come gestirli, in modo da porvi rimedio durante la fase di ETL iniziale. Per migliorare quindi la qualità dei dati occorre trovare una soluzione per:

- dati inconsistenti
- valori mancanti

Dati inconsistenti Per il 2011 vi è un numero di acquisti molto anomalo rispetto agli altri anni anche solo in proporzione ai 9 mesi di cui si dispongono i dati; infatti, come abbiamo visto precedentemente, il numero minimo di acquisti risale ai primi anni ed è di circa 30.000. Se si pensa che disponiamo del 75% dei dati di tutto il 2011 (ossia settembre compreso), abbiamo un numero di acquisti pari a poco più di 6.000; considerando che l'azienda è ancora in piena attività e continua a vendere regolarmente senza essere in crisi, questo fa presupporre che anche per i mesi di cui si dispone, i dati siano incompleti o comunque non aggiornati.
Avevamo già escluso il 2011 dalle analisi di mercato, ora si pensa di lasciarlo fuori anche per quanto riguarderà le fasi successive di ETL e di Data Mining.

Valori mancanti Per calcolare il margine occorre che il costo di acquisto sia specificato per ogni dettaglio di fattura, tuttavia questo valore non sempre è fornito: quando non è pervenuto è lasciato nullo. Ovviamente non si può calcolare il margine in maniera mista, ma bisogna usare un criterio adeguato, costringendoci a fare delle scelte. Le possibili soluzioni possono essere:

- escludere le istanze con costo d'acquisto nullo quando si calcola il margine
- evitare di fare calcoli in cui è previsto l'uso del margine

- cercare di sostituire i valori mancanti con un valore approssimativo

Per decidere occorre innanzitutto stimare quantitativamente l'assenza di tale valore.
Prendiamo quindi in considerazione tutto lo storico a disposizione e di ciascun anno analizziamo la percentuale di valori nulli sul totale dei dettagli di fattura appartenente a tale anno.

2002	**2003**	**2004**	**2005**	**2006**	**2007**	**2008**	**2009**	**2010**	**2011**
99,04%	3,42%	3,72%	2,92%	1,26%	2,27%	0,83%	7,98%	7,32%	3,83%

La maggior parte, come si può vedere, appartiene al 2002, tuttavia questo ci interessa ben poco, dato che probabilmente lo escluderemo, così come abbiamo già fatto per il 2011; la percentuale rimane comunque abbastanza elevata per il 2009 ed il 2010.
Se tagliamo quindi fuori il 2002 ed il 2011, le osservazioni affette da questo problema rimangono in ogni caso in una quantità piuttosto consistente; cerchiamo comunque di recuperarle in maniera approssimativa ma intuitiva, osservando le istanze dello stesso prodotto in cui il costo d'acquisto è presente.
Bisogna poi tenere conto del fatto che il costo d'acquisto riportato non è unitario, ma complessivo dell'intera quantità di merce acquistata appartenente ad una determinata tipologia di prodotto e questa quindi sarà proporzionata al relativo fatturato.
Per ognuna delle istanze dove il costo d'acquisto non è nullo ricaviamo quindi una costante moltiplicativa, data da:

$$cost_{prod} = \frac{costo^{acq}}{fatturato}$$

ed eseguiamo poi una media tra le costanti dello stesso prodotto.
Per le osservazioni in cui il costo d'acquisto manca, basandoci sul fatturato dell'osservazione e sul prodotto in questione, si va quindi a sostituire nel seguente modo:

$$costo^{acq} = cost_{prod} \cdot fatturato$$

Tuttavia, anche con questo rimedio, rimangono scoperte ancora diverse migliaia di istanze, che non hanno riscontro con prodotti in cui sia riportato il costo d'acquisto.

Occorre poi pensare che il costo di un prodotto normalmente subisce variazioni nel corso degli anni, quindi è più logico effettuare tale operazione tenendo in considerazione anche l'anno a cui è associata la costante ricavata.
Sotto quest'ultima ipotesi è stata effettuata un'analisi e si è riscontrato che se si suddivide per anno, la variabilità delle costanti ottenute è circa la metà rispetto a quella presa considerandoli nell'intero storico; questo ci porta ad utilizzarle nel primo caso. Con questa ultima decisione si ha però lo svantaggio di avere un'ulteriore restrizione del numero delle osservazioni risolvibili, poiché non per tutti gli anni è possibile avere un riscontro che porta a ricavare la costante che ci serve.
Alla fine si hanno ancora quindi circa 74.000 osservazioni che non possono essere rimediate, con però quasi la stessa cifra di recuperate. Tutte le osservazioni rimaste irrisolte verranno ignorate quando si calcolerà il margine, trattandole come se non esistessero.

2.4 Data Preparation

Dopo una prima analisi dei dati, occorre selezionare, preparare e trasformare i dati allo scopo di costruire i dataset che poi verranno utilizzati nella fase di Data Mining.
I dati originali, come detto in precedenza, sono contenuti nello schema *ds* del database che durante tutto il processo non verrà alterato; lo schema di lavoro invece sarà costituito da *dm* ed ospiterà tabelle temporanee, d'appoggio o ricostruite; in quest'ultimo saranno ospitate anche le tabelle di output con i risultati del Data Mining.

2.4.1 Riduzione del dataset iniziale

Visto l'enorme mole di dati che si deve trattare, sarà indubbiamente necessario fare un campionamento dei dati originali, per poi continuare a lavorare esclusivamente su tale campione; l'elaborazione sarebbe onerosa per quanto riguarda tempo e memoria utilizzando l'intero dataset, probabilmente lo sarà ancora anche lavorando con solo una frazione di esso.
La selezione va fatta sui clienti coinvolti e non sulle singole istanze

delle varie tabelle, perché comunque le fatture di ogni cliente vanno mantenute tutte affinché gli algoritmi di Data Mining riescano a dedurre valide relazioni, dando così risultati efficaci.

Con la prima versione del database, si era pensato quindi di prendere in maniera casuale il 20% dei clienti, che però risultassero tra le fatture (non tutti quelli inseriti nella tabella dei clienti hanno delle fatture associate); inoltre si era deciso di considerare solo quelli che erano di tipo Wholestore, dato che probabilmente sono i clienti che ci possono offrire i risultati più interessanti.

Con il database revisionato, in cui i dati sono stati ridotti di numero, tale campionamento non è stato più strettamente necessario, tuttavia si può prendere un dataset anche all'80% per avere poi tempi di estrazione più brevi e velocizzare quindi tutto il processo.
Dopo aver scelto casualmente i clienti, si ricrea la tabella delle fatture e le restanti tabelle che derivano da quest'ultima, inserendo solo le istanze che si riferiscono a clienti che abbiamo selezionato in precedenza.
Nel fare ciò effettuiamo un'ulteriore riduzione del dataset, escludendo le fatture appartenenti al 2002 ed al 2011, dato che dall'analisi è emerso che tali dati non sono affidabili; inoltre tagliando fuori le fatture molto vecchie, ossia precedenti al 2003, si evitano a priori eventuali incompatibilità di valuta[9].

2.4.2 Definizione di variabili

Possiamo ora pensare di aggiungere alle osservazioni dei clienti nuove variabili, ossia altri attributi che presumibilmente possono individuare o comunque caratterizzare l'andamento di un cliente che decide di abbandonare; il sistema cercherà successivamente di capire tramite gli algoritmi di Data Mining se esistono effettivamente delle relazioni tra essi in modo così da riuscire a classificare correttamente i vari clienti.
Le variabili che si possono sfruttare o comunque derivare dipendono dal dominio dei dati del database e possono quindi cambiare notevol-

[9]Si intende tra euro e lira, dato che quest'ultima ha finito ufficialmente il suo corso nel marzo del 2002.

mente da caso a caso.

Tra i valori che abbiamo a disposizione dal dominio si ha:

- *fatturato*
- *costo d'acquisto*
- *insoluto*
- *insoluto recuperato*
- *sconto*

Da queste ricaviamo il *margine*, già definito nella sezione precedente:

$$margine = fatturato - costo^{acq}$$

Inoltre possiamo ricavare altre informazioni utili:

- *numero di acquisti*
- *numero di resi*[10]
- *motivo*[11] *del reso*

Combinando le variabili di base ed utilizzando anche il margine, possiamo derivare quindi le nuove variabili:

fatturato su margine

$$fatturato_m = \frac{fatturato}{margine}$$

Possiamo poi anche considerare *minimo*, *massimo*, *media* e *deviazione standard* di questo valore tra le varie fatture, per un totale di 5 variabili.

[10]Con tale termine ci si riferisce alla merce restituita.

[11]La causa che ha portato a restituire la merce; il motivo è espresso tramite un identificativo.

sconto sul rapporto fatturato/margine

$$sconto_{fm} = \frac{sconto}{fatturato_m}$$

Considerando anche il *minimo*, il *massimo*, la *media* e la *deviazione standard* di tale valore si hanno a disposizione 5 diverse variabili.

rapporto fatturato/margine sul numero di acquisti

$$fatturato_{mn} = \frac{fatturato_m}{N_{acq}}$$

insoluto su fatturato

$$insoluto_f = \frac{insoluto}{fatturato}$$

Da questo valore possiamo ricavare anche *minimo*, *massimo*, *media* e *deviazione standard*, ottenendo un totale di 5 variabili.

insoluto recuperato su fatturato

$$insoluto_f^{rec} = \frac{insoluto^{rec}}{fatturato}$$

Anche da questa misura possiamo alla fine ottenere complessivamente 5 variabili, se ne aggiungiamo anche il *minimo*, il *massimo*, la *media* e la *deviazione standard*.

distanza di acquisto

Misura la distanza tra un acquisto e l'altro:

$$distanza^{acq} = data_{i+1}^{acq} - data_i^{acq} \qquad data_{i+1}^{acq} > data_i^{acq}$$

Da questa misura possiamo ricavare complessivamente 12 variabili se si prendono in considerazione *minimo*, *massimo*, *media*, *deviazione standard* sia basandosi sui *giorni*, sia sulle *settimane* e sia sui *mesi*.

ritardo di riacquisto
Distanza tra l'ultimo acquisto effettuato e l'estremo superiore dell'intervallo di tempo considerato:

$$ritardo^{riacq} = sup(T) - max(data^{acq})$$

Se si considerano i *giorni*, le *settimane* ed i *mesi* da questa misura possiamo ottenere 3 differenti variabili.

frequenza
Distanza tra primo ed ultimo acquisto sul numero di acquisti totali:

$$frequenza = \frac{max(data^{acq}) - min(data^{acq})}{N_{acq}}$$

ratio del ritardo di riacquisto
Ritardo di riacquisto sulla somma della media con la deviazione standard della distanza di acquisto:

$$ratio^{rit\_riacq} = \frac{ritardo^{riacq}}{avg(distanza^{acq}) + stddev(distanza^{acq})}$$

ritardo dell'ultimo acquisto
Distanza tra il penultimo acquisto effettuato e l'ultimo:

$$ritardo^{ult\_acq} = data_i^{acq} - data_{i-1}^{acq} \qquad data_i = max(data^{acq})$$

Come per il ritardo di riacquisto, possiamo considerare i *giorni*, le *settimane* ed i *mesi*, ottenendo 3 differenti variabili.

ratio dell'ultimo acquisto
Ritardo dell'ultimo acquisto sulla somma della media con la deviazione standard della distanza di acquisto:

$$ratio^{ult\_acq} = \frac{ritardo^{ult\_acq}}{avg(distanza^{acq}) + stddev(distanza^{acq})}$$

valore dei resi su fatturato

Una variabile importante è il valore[12] dei resi sul fatturato totale:

$$resi_f = \frac{fatturato^{resi}}{fatturato}$$

Se di questa variabile prendiamo in considerazione anche *minimo*, *massimo*, *media* e *deviazione standard*, abbiamo un totale di 5 variabili.

numero dei resi su numero di acquisti

Oltre al rapporto economico dei resi visto in precedenza, ci interessa anche la frequenza del fenomeno, pertanto si introduce anche come misura il numero di resi totali sul numero degli acquisti totali:

$$resi_{acq} = \frac{N_{resi}}{N_{acq}}$$

Inoltre si è pensato che fosse utile ricavare dalle informazioni sui resi ulteriori variabili secondo un criterio più specifico: si può infatti considerare il numero dei resi dovuti ad un determinato motivo sul numero di acquisti totali:

$$mot_i^{reso} = \frac{N_{mr_i}}{N_{acq}} \qquad i = 1, ..., 10 \quad N^c_{mr_i} > N^c_{mr_{i+1}}$$

Questo ultimo valore lo prendiamo in base ai *10 motivi di reso* più frequenti a livello globale (considerando tutti i clienti presenti nel database); per ciascun cliente viene calcolato tale valore per ognuno di questi motivi.
Poiché si ha una variabile diversa per ogni motivo di reso e considerando anche la precedente misura globale, abbiamo un totale di 11 variabili.

Si hanno quindi a disposizione 13 variabili principali, ma considerando di derivare da esse le altre variabili menzionate si arriva ad averne 58.

[12] Il valore corrisponde al fatturato della relativa merce.

Inoltre possiamo ottenere un'altra importante misura tramite la regressione lineare: invece di prendere i valori complessivi nell'intero periodo temporale, calcoliamo tali variabili suddividendo quest'ultimo in trimestri; riportiamo poi i risultati in un grafico cartesiano in funzione dei trimestri[13]. Il coefficiente angolare della retta di interpolazione che approssima i punti trovati, ci indica la variabilità della misura in esame nel periodo considerato.
Perciò, dal momento che il coefficiente angolare può essere calcolato per ognuna delle variabili disponibili, alla fine possiamo ottenerne il doppio di quelle già definite finora, arrivando quindi ad un totale di 116.

2.4.3 Valori precalcolati

La maggior parte delle variabili per essere calcolata ha bisogno di valori presenti nei dettagli delle fatture.
Tuttavia possiamo fare alcune osservazioni:

- La tabella delle fatture ha alcuni milioni di osservazioni.
  È chiaro che eseguire frequentemente interrogazioni su questa tabella può portare ad un notevole impiego di tempo.

- Valori importanti e ricorrenti come la data ed il cliente sono presenti solo nella tabella delle fatture, non in quella dei dettagli.
  È ovvio che l'unico modo per ricavarli quando si prelevano i valori dai dettagli delle fatture è eseguire una *join* (o comunque fare un confronto con un metodo simile) tra quest'ultima tabella e quella delle fatture: un'operazione così computazionalmente costosa con una tabella molto grande come quella dei dettagli, peggiora notevolmente i tempi di ricerca dei dati già di per sè onerosi.

- L'unità di riferimento di cui abbiamo bisogno è il contenuto di un'intera fattura, non le sue singole parti.
  È inutile quindi ogni volta che ce ne è bisogno sfogliare tutta la tabella dei dettagli e calcolarne i valori totali per ogni fattura.

[13] Ossia inseriamo i valori ottenuti in ordinata ed i trimestri in ascissa.

Partendo da questi presupposti, si pensa che sia quindi meglio evitare di accedere alla tabella dei dettagli delle fatture e la maniera migliore per farlo è accorpare i valori che ci occorrono nelle uniche tabelle:

fatture(id_fattura, cod_fattura, anno_documento, id_agente, id_causale, data_fattura, id_cliente_fatturazione, fatturato_tot, margine_tot, fatturato_resi_tot, sconto_tot, insoluto_tot, insoluto_recuperato_tot, fatturato_inattesa_tot, fatturato_pagato_tot)

motivi_reso(id_cliente_fatturazione, id_fattura, mot_id)

Aggiungendo quindi colonne di valori precalcolati[14] che prima erano presenti solo tra i dettagli di fattura, si potrà utilizzare esclusivamente la tabella delle fatture con i seguenti vantaggi:

- accessi più veloci poiché la tabella è molto più piccola
- mancanza di operazioni di *join* (o simili) tra le due tabelle
- valori già calcolati una volta per tutte per ogni fattura

Come si può notare, si è dovuta aggiungere anche una tabella per contenere le informazioni sui motivi di reso: i valori presenti in essa non sono aggregati, tuttavia è molto più veloce eseguire una ricerca in questo modo rispetto a farlo sull'intera tabella dei dettagli.
Tramite l'id della fattura che viene mantenuto per ogni osservazione di questa tabella, è possibile ricavare dalla tabella delle fatture tutte le informazioni che occorrono per il calcolo delle variabili sui resi.
Con questo metodo si è riscontrata una riduzione dei tempi di elaborazione del 40%, confermando quindi un aumento considerevole delle prestazioni.

2.4.4 Estrazione ed etichettatura

Per creare training e test set è necessario selezionare la clientela ed attribuire una classe ad ogni elemento dell'insieme ottenuto; per fare ciò occorre prima definire alcuni parametri essenziali:

[14]Il nuovo attributo *margine_tot* è un valore ulteriormente derivato e consiste nella differenza tra il fatturato ed il costo d'acquisto.

Numero minimo di acquisti
: Per l'estrazione della clientela di interesse è fondamentale stabilire un numero minimo di acquisti che un cliente deve effettuare nel periodo in esame per essere considerato abituale e non occasionale; questi ultimi verranno esclusi.

Vita
: È l'arco di tempo in anni in cui considerare gli acquisti minimi.

Definizione di abbandono
: La definizione di abbandono è il periodo in mesi sulla base del quale un cliente viene decretato come abbandono o fedele. Inoltre, durante l'estrazione, non vengono selezionati i clienti che hanno fatto trascorrere tale periodo tra un determinato acquisto e quello successivo.

Data di riferimento
: La data di riferimento definisce la collocazione temporale dei parametri precedenti.

Gli intervalli temporali di interesse vengono ricavati procedendo a ritroso: il periodo di etichettatura è quello che precede la data di riferimento, della durata dei mesi della definizione di abbandono; a partire dall'inizio di questo, il periodo di vita è preso andando indietro negli anni.

Figura 2.7: *Criterio di etichettatura*

In conclusione si avrà che la fine del periodo di vita coinciderà con l'inizio del periodo di etichettatura, mentre il termine di quest'ultimo sarà fissato dalla data di riferimento.

L'etichettatura avviene secondo la formula:

$$f_e \leqslant x \cdot f_v$$

dove f_e è il fatturato medio nel periodo di etichettatura e f_v è quello nel periodo di vita; x è una costante proporzionale che regola tale funzione, ed è compreso nell'intervallo *[0,1]*.
Un cliente sarà dunque etichettato come:

- *A*, ossia abbandono, nel caso che tale condizione sia verificata

- *F*, ossia fedele, in caso contrario

Da questa relazione si intuisce che se la costante x è fissata a *0*, viene catalogato come abbandono ogni cliente che nel periodo di etichettatura non ha più effettuato alcun acquisto (ossia se fattura *0 euro*); se invece la soglia non è nulla, allora l'etichettatura avviene in proporzione al fatturato prodotto durante il periodo di vita.

2.4.5 Training e test set

I dati per costruire training e test set per generare e validare il modello saranno presi dallo stesso dataset di dati sorgenti, perché per il momento sono gli unici dati di cui disponiamo ed inoltre perché di essi conosciamo il valore. L'unica differenza sarà il periodo temporale in cui essi verranno scelti, però le caratteristiche ed i criteri di selezione, validazione ed etichettatura delle due tipologie di dataset saranno simili; ovviamente il training set dovrà essere temporalmente antecedente al relativo test set, mentre il numero degli uni e degli altri può essere variabile, a seconda degli obiettivi della simulazione.
Essi saranno composti da osservazioni aventi come attributi:

- l'id del cliente

- le variabili utilizzate

- la classe

2.5 Modeling

Una volta costruita la coppia di dataset, si passa alla fase di Data Mining, dove verrà prodotto un modello a partire dal training set generato. Il modello verrà indotto per ogni algoritmo scelto e per ogni coppia prevista di training e test set; si vedrà nella fase successiva di Evaluation quali combinazioni daranno risultati migliori.

2.5.1 Algoritmi utilizzati

Dal momento che ci si trova davanti ad un 2-class classification problem occorrerà usare degli algoritmi adatti alla circostanza.
Tra gli algoritmi di classificazione se ne sono scelti 5 tra quelli ritenuti più adatti, prendendo anche combinazioni fra essi:

- *J48*[15]
- *SimpleCart*[16]
- *RandomForest*[17]
- *AdaBoostM1*[18] *con J48*
- *AdaBoostM1 con DecisionStump*[19]

Come si può notare ci si affida principalmente ai decision tree (J48 e SimpleCART) che in genere offrono buoni risultati e sono quelli più comunemente utilizzati; poi si fa ricorso ad un aggregatore di classificatori che usa decision tree (RandomForest) ed ad un altro aggregatore (AdaBoostM1) usato prima con un decision tree (J48) e poi con un classificatore rule based (DecisionStump).
La scelta inoltre ricade in questi classificatori perché essi estraggono

[15]Implementazione open source di C4.5 (vedere appendice C.1.1 - *Classificatori: C4.5*).
[16]Variante di CART (vedere appendice C.1.1 - *Classificatori: CART*), ma produce alberi decisionali più ridotti.
[17]Vedere appendice C.1.4 - *Classificatori: RandomForest.*
[18]Vedere appendice C.1.4 - *Classificatori: AdaBoost.M1.*
[19]Implementazione di R1 (vedere appendice C.1.2 - *Classificatori: 1R*).

dati in maniera white box, permettendoci di comprendere un fenomeno oltre a prevederlo.
J48 verrà utilizzato in combinazione con un classificatore *cost-sensitive* che permetterà di essere più efficace a discriminare falsi abbandoni e falsi fedeli. Infatti i due errori di previsione pesano in maniera totalmente differente: emerge che è più dannoso per l'azienda il secondo rispetto al primo, perché i falsi fedeli sono in realtà degli abbandoni, che non sono stati riconosciuti. Inoltre training e test set risultano sbilanciati in favore dei fedeli in genere in una proporzione di 10:1, a causa di ciò si rischia che nell'addestramento del sistema, quest'ultimo si abitui a riconoscere bene i fedeli, per poi invece non individuare gli abbandoni.
Per questo motivo, a seconda del caso, si cercherà di pesare le due situazioni in maniera differente; in genere si privilegerà di fare in modo che il sistema stia più attento a non predire falsi fedeli anche a costo di predire maggiori falsi abbandoni.
Infine per tali algoritmi occorre stabilire la cardinalità minima di oggetti che possono essere presenti in un nodo dell'albero; questo evita che quest'ultimo venga splittato in maniera eccessiva adattandosi così troppo allo specifico training set da cui sarà indotto il modello, tutto ciò riduce inoltre frammentazione e dispersione nell'albero.

2.5.2 Parametri per il Data Mining

In questa fase sarà quindi fondamentale l'uso di due nuovi parametri:

Differenza di costo
: ossia il peso che influisce sulla matrice dei costi[20] sulla quale si basa l'assegnazione di una classe. Con un valore pari a *0* si decreta l'equità dei due errori, mentre un valore positivo influisce

[20]La matrice dei costi è strutturata come la matrice di confusione; i valori delle istanze corrette (TP e TN) sono mantenuti a *0*, mentre quelli corrispondenti agli errori (FP e FN) sono tarati con un peso dipendente dalla rispettiva importanza. Un algoritmo cost-sensitive viene poi configurato con tale matrice prima di essere applicato, in modo tale da classificare successivamente basandosi prevalentemente sul peso dell'errore piuttosto che sulla sua probabilità.

maggiormente sul riconoscimento dei falsi fedeli, mentre negativo su quello dei falsi abbandoni. Nel primo caso il sistema commetterà pochi errori gravi, nel secondo, invece, commetterà meno errori sulla previsione di abbandono.
La matrice dei costi sarà come quella mostrata di seguito[21].

		classe predetta	
		F	**A**
classe reale	**F**	0	$1 + \|d^-\|$
	A	$1 + \|d^+\|$	0

Numero minimo di elementi in un nodo dell'albero
nel caso gli oggetti all'interno di un nodo raggiungano il valore impostato, tale nodo sarà decretato come una foglia dell'albero ed il resto del ramo sarà potato.

Tali parametri saranno usati dagli algoritmi di Data Mining per indurre il modello e verranno applicati per ogni coppia di dataset (training e test set).

2.6 Evaluation

Per ottenere un buon modello bisogna ricorrere a migliaia di simulazioni che generino altrettanti modelli; questi ultimi vengono prodotti sulla base di una serie di combinazioni tra un insieme di possibili valori[22] dei parametri descritti nelle precedenti sezioni; analizzando le prestazioni si può decidere di volta in volta di concentrarsi su determinati valori, scartandone altri, fino ad ottenere dei modelli ottimali[23]. La bontà dei modelli è valutata attraverso il calcolo delle prestazioni su più test set; per avere maggiore stabilità dei risultati si cerca di

[21] Il valore di base per *FF* e *FA* è *1*, il valore assoluto del parametro *differenza di costo* verrà sommato ai primi nel caso il suo valore relativo sia positivo, ai secondi nel caso sia negativo.

[22] Occorre effettuare un'analisi preliminare per scegliere gli iniziali valori candidati.

[23] Questo aspetto sarà discusso nel dettaglio nel capitolo 5.

valutarne la stazionarietà, applicando ogni modello generato con un determinato training set a diversi test set costituiti da dati presi in periodi temporali differenti e valutando poi quanto i vari risultati differiscono tra loro[24]; ovviamente più la diversità è minore, tanto più il modello produce risultati stabili.

2.6.1 Criterio di valutazione

Le principali misure su cui ci si può basare per valutare la bontà dei modelli sono quelle tipiche della classificazione: *accuratezza* a livello generale mentre *precisione*, *recall* e *f-measure* per quanto riguarda separatamente abbandoni e fedeli.
Come già detto nelle precedenti sezioni, questo è un problema sbilanciato e l'errore più grave si commette nello sbagliare ad individuare i fedeli, tuttavia il nostro obiettivo è predire gli abbandoni, pertanto per avere dei risultati affidabili occorre basarsi soprattutto sulla precisione su questi ultimi.
Quindi, in definitiva, le misure di riferimento saranno:

$$precision(A) = \frac{TA}{TA + FA} \qquad recall(A) = \frac{TA}{TA + FF}$$

e la relativa *f-measure*.
Non è indicato e non ha molto senso far ricorso all'accuratezza; infatti modelli che all'apparenza possono sembrare buoni perché producono risultati in cui quest'ultima supera il 90% potrebbero in realtà trarre in inganno se il restante 10% è dato proprio dalla maggior parte degli abbandoni che non è stata individuata, ma al contrario è stata classificata come fedele.

2.7 Deployment

Questo processo dovrà essere automatizzato da un sistema software, che concretizzerà l'analisi e le soluzioni esposte finora.

[24] Una misura per tale scopo è la deviazione standard.

Esso deve essere costituito principalmente da due parti: una più complessa predisposta per la generazione dei modelli, che chiameremo *metamacchina* e che svolgerà tutto quello descritto nelle precedenti sezioni, e la parte più comune che sarà invece l'effettivo *deployment*, adibita all'applicazione dei modelli generati dalla metamacchina a nuove istanze di dati.
Se si utilizzano più modelli contemporaneamente occorre scegliere il criterio di assegnazione che in generale può essere all'unanimità o ad elezione (maggioranza).

2.7.1 Sistema software ed output

Si può pensare di fornire all'azienda solo i risultati ottenuti sotto forma di file, ma è più ovvio rilasciare l'intero deployment con i migliori modelli trovati[25] già precaricati, in modo da poterli applicare ogni volta che vi siano nuovi dati sconosciuti disponibili che richiedono di essere predetti. Per tale motivo occorre progettarlo per essere usato da un'utenza che non abbia esperienza nel campo del Data Mining, quindi bisogna implementare un deployment in grado di eseguire tutto il lavoro in maniera automatica semplicemente avviandolo; inoltre il numero di parametri necessari per configurarlo deve essere ridotto al minimo.
Quindi, dal momento che il deployment dovrà essere rivolto ad un'utenza comune, esso dovrà lanciare il processo tramite una semplice GUI, il risultato sarà poi prodotto in maniera automatica e scritto su una tabella relazionale all'interno del database, in una sezione distinta da quella usata per contenere i dati sorgenti.
L'output è costituito dalla lista dei clienti con la corrispondente classe predetta A o F, tuttavia dato che l'obiettivo finale sarà quello di prevedere gli abbandoni, basterà creare una tabella contenente solo quelli appartenenti alla prima tipologia.

[25] I modelli non devono essere comunque troppi (massimo fino a 5-6).

Capitolo 3

Il sistema software di Data Mining

In questo capitolo verrà presentata la *D2D suite*, ossia il software sviluppato ed utilizzato per generare, validare ed applicare modelli di Data Mining e che quindi automatizza l'intero CRISP-DM.

3.1 Introduzione

La *D2D* (*Data To Decisions*) *suite* è un software sviluppato dall'Ing. Castori[1]; essa comprende vari tool che permettono di eseguire analisi di Data Mining, generare modelli ed applicarli, utilizzando specifici algoritmi. Tale suite inoltre integra anche requisiti non funzionali che le conferiscono le caratteristiche di un prodotto che potrebbe essere poi messo in futuro sul mercato. Il linguaggio di programmazione utilizzato è *java*[2], che offre sia il vantaggio di essere *platform independent*, sia quello di potere usufruire di tool di supporto sviluppati in tale linguaggio che sono per la maggior parte open source e questo permette di trarre anche un beneficio economico.
Lo strumento su cui ci si è appoggiati per applicare gli algoritmi noti

[1] La suite è stata sviluppata come progetto di tesi specialistica dall'Ing. Castori presso la Seconda Facoltà di Ingegneria, Università di Bologna (vedere il riferimento bibliografico [6]); gli algoritmi ed i procedimenti su cui si basa la suite sono stati ideati invece dal prof. Moro, docente presso l'Università di Bologna.

[2] Vedere riferimento bibliografico [20].

di Data Mining è *WEKA*[3] ed è integrato come libreria nei tool principali della suite.
Il software di base inizialmente sviluppato, è predisposto per affrontare il churn della clientela, ossia lo stesso problema di previsione di cui ci stiamo occupando in questa tesi, tuttavia esso è stato implementato sulla base di un caso aziendale diverso[4], pertanto nel descrivere i parametri che occorrono per configurarlo, bisogna tenere in considerazione tale particolare. Il procedimento di elaborazione ed il modo di operare, invece, rimangono indipendenti dal caso a cui è applicato.

3.2 D2D suite

I vari tool di cui si compone la *D2D suite* sono[5]:

- **BusinessFramework**
  Applicativo per la generazione, la validazione ed il salvataggio crittografato su file di modelli, secondo alcuni algoritmi di Data Mining predisposti. Costituisce il tool principale della suite ed è rivolto ad utenti esperti.

- **DMU**
  La *Data Mining Unit* è lo strumento che utilizza i modelli cifrati generati dal BusinessFramework e li applica a nuovi dati. È il tool della suite di uso più comune ed è rivolto alla normale utenza.

- **XMLConfigurator**
  Strumento grafico per editare in maniera intuitiva il file *xml* di configurazione per il BusinessFramework e la DMU.

- **ServerLicense**
  Piccola applicazione aggiuntiva per gestire il database delle licenze.

[3] Vedere appendice B.2.1 - *WEKA*.

[4] Il sistema era predisposto per lavorare con i dati di un'azienda di prodotti per l'igiene.

[5] Tuttavia in questa tesi prenderemo in considerazione solo i primi 3 qui riportati.

- **ServerDMU**
  Strumento integrativo con il compito di verificare la validità della licenza di utilizzo della suite.

3.2.1 BusinessFramework

Il BusinessFramework è la metamacchina, ossia lo strumento dedicato a generare, validare e salvare i modelli ricavati dai dati in input tramite specifici algoritmi di Data Mining. L'interazione con il database avviene tramite query SQL utilizzando driver JDBC.
L'esecuzione di tale strumento prevede:

1. generazione del training set
2. generazione del test set
3. induzione del modello dal training set
4. validazione sul test set
5. scrittura degli indicatori di prestazione in *dm.performance*, appartenente allo stesso database
6. scrittura del decision tree su file di testo
7. memorizzazione su file del modello e del suo descrittore

Le cartelle usate per contenere gli output sono:

models
: per i modelli, aventi estensione *.model*

decriptors
: per i descrittori dei modelli, aventi estensione *.desc*

trees
: per gli alberi decisionali generati

Inoltre ad ogni avvio viene creato nella directory principale un file di log che tiene traccia dei passi di esecuzione ed eventualmente degli errori.

I modelli vengono salvati crittografati tramite *Password Based Encryption* (*PBE*); come identificativo delle varie simulazioni e del nome attribuito ai diversi file di output, viene utilizzata una stringa composta dai valori dei principali parametri usati nella specifica esecuzione. Per generare il modello vengono utilizzati 5 algoritmi differenti: dall'output a video è possibile confrontarne le prestazioni e scegliere il migliore o comunque quello ritenuto più opportuno.
Gli algoritmi utilizzati[6], già elencati in precedenza, sono:

0 *J48*

1 *SimpleCart*

2 *RandomForest*

3 *AdaBoost con J48*

4 *AdaBoost con DecisionStump*

Dopo la loro esecuzione ed ottenuti come risultato i rispettivi modelli, nella GUI di output vengono visualizzate le caratteristiche e le prestazioni derivanti dall'applicazione di questi ultimi al test set:

- cardinalità del training e del test set
- numero di attributi
- parametri dell'algoritmo
- albero decisionale generato
- matrice di costo e matrice di confusione
- accuratezza
- precisione, recall e f-measure su entrambe le classi
- kappa statistic.

[6]La numerazione riportata rappresenta l'ID associato al rispettivo algoritmo.

Parametri I parametri per l'esecuzione del BusinessFramework vengono impostati attraverso il file *conf.xml*, dal quale il sistema legge i valori necessari e li applica durante l'elaborazione.
I parametri sono suddivisi in categorie.

Parametri di *input*:

acquisti
[*Valore numerico intero (singolo o vettore di valori)*]
Numero di acquisti minimo che un individuo (o punto vendita) deve avere effettuato per essere considerato un cliente abituale e non occasionale (questi ultimi non verranno presi in considerazione).

vita
[*Valore numerico intero (singolo o vettore di valori)*]
Arco di tempo, espresso in anni, entro cui verranno considerati i dati relativi ad un determinato cliente.

abbandono
[*Valore numerico intero (singolo o vettore di valori)*]
Numero di mesi senza acquisti per considerare un cliente come abbandono. Rappresenta il periodo di etichettatura.

data
[*Data*]
Data[7] di riferimento per costruire training e test set[8].

Parametri per il *database*:

url
[*Testo*]
URL del database che contiene i dati sorgenti ed in cui dovranno essere scritti i risultati ottenuti.
Dal momento che utilizziamo *PostgreSQL*[9] come DBMS, nel nostro caso sarà del tipo *jdbc:postgresql:database*.

[7] La data è espressa nel formato *gg/mm/aaaa*.
[8] Per la DMU (vedere la sezione successiva) corrisponde invece alla data di previsione ed è inoltre la data di riferimento per estrarre i clienti coinvolti.
[9] Vedere riferimento bibliografico [15].

user
[*Testo*]
Il nome utente (o proprietario) del database.

password
[*Elenco di valori numerici interi*]
Password associata all'utente del database. Essa viene conservata nel file di configurazione crittografata secondo la password PBE, pertanto occorre ricavarla nella sua nuova forma attraverso appositi strumenti che permettano tale passaggio a partire dalla forma in chiaro.

Parametri per la scelta delle *variables*:

fatturato_margine
[*Valore booleano*]
Indica se deve essere presa in considerazione la variabile relativa al rapporto fatturato su margine durante la generazione dei dataset.

sconto_fama
[*Valore booleano*]
Indica se bisogna considerare la variabile relativa al rapporto sconto su fatturato e margine.

fama_numacq
[*Valore booleano*]
Indica se bisogna considerare la variabile relativa al rapporto fatturato e margine su numero di acquisti.

insoluto_fatturato
[*Valore booleano*]
Indica se bisogna considerare la variabile relativa al rapporto insoluto su fatturato.

insoluto_r_fatturato
[*Valore booleano*]
Indica se bisogna considerare la variabile relativa al rapporto insoluto recuperato su fatturato.

ritardo_riacquisto
[*Valore booleano*]
Indica se bisogna considerare la variabile relativa al ritardo di riacquisto.

distanza_acquisti
[*Valore booleano*]
Indica se bisogna considerare la variabile relativa alla distanza tra acquisti.

frequenza
[*Valore booleano*]
Indica se bisogna considerare la variabile relativa alla frequenza di acquisto.

ratio
[*Valore booleano*]
Indica se bisogna considerare la variabile di ritardo di riacquisto sulla somma di media e deviazione standard della distanza di acquisto.

settore
[*Valore booleano*]
Indica se bisogna considerare la variabile basata sul settore merceologico.
È una variabile non usata nel caso che stiamo trattando.

agente
[*Valore booleano*]
Indica se bisogna considerare la variabile basata sull'agente di vendita.
È una variabile non usata nel caso che stiamo trattando.

categoria_fatturato
[*Valore booleano*]
Indica se bisogna considerare la variabile per le categorie a maggior fatturato.
È una variabile non usata nel caso che stiamo trattando.

categoria_margine

[*Valore booleano*]
Indica se bisogna considerare la variabile per le categorie a maggior margine.
È una variabile non usata nel caso che stiamo trattando.

Parametri necessari per la generazione del *model*:

minnumobj

[*Valore numerico intero (singolo o vettore di valori)*]
Indica la cardinalità minima degli oggetti che possono essere presenti in un nodo dell'albero. Nel caso questa condizione non venga soddisfatta, l'albero non viene più splittato: si interrompe tale ramo e si considera l'ultimo nodo accettato come una foglia. Questo evita che il modello che deve essere generato non si adatti troppo allo specifico training set dal quale è indotto.

difference

[*Valore numerico intero positivo o negativo (singolo o vettore di valori)*]
Indica la differenza di costo[10] che deve esserci tra falsi fedeli e falsi abbandoni. In caso questo valore sia diverso da *0*, l'algoritmo viene tarato per riconoscere meglio uno dei due (valore positivo per i primi, negativo per i secondi); questo è stato introdotto perché, come già detto, nella pratica potrebbero pesare di più errori di previsione prodotti in un caso rispetto ad un altro (in genere sono i falsi fedeli quelli con rilevanza maggiore).

mode

[*Valore booleano*]
Indica la modalità di costruzione di training e test set e come devono essere interpretati i successivi valori di k e h, in riferimento a *data*:

FALSE: *multitest*
Vengono prese tante date di test quanti sono gli elementi di h, ognuna al valore indicato di mesi successivi alla data

[10] Valutata secondo la matrice dei costi.

di riferimento.
Per ogni data di test ricavata, si ottengono poi tante date di training quanti sono gli elementi di k, ognuna in corrispondenza del valore indicato di mesi precedenti alla data di test associata.

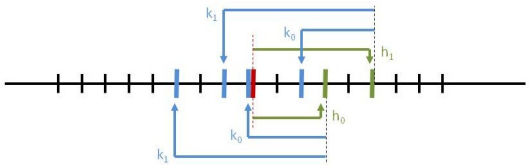

Figura 3.1: *Multitest: date di training (k) e di test (h)*

TRUE: *valutazione della stazionarietà*
Vengono prese tante date di training quanti sono gli elementi di h, ognuna al valore indicato di mesi successivi alla data di riferimento.
Per ogni data di training ricavata, si ottengono poi tante date di test quanti sono gli elementi di k, ognuna in corrispondenza del valore indicato di mesi successivi alla data di training associata.

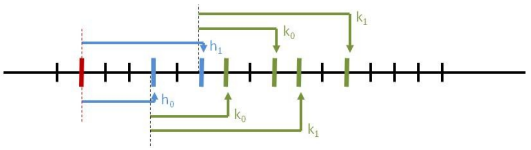

Figura 3.2: *Stazionarietà: date di training (h) e di test (k)*

k

[*Valore numerico intero (singolo o vettore di valori)*]
Offset delle date di test nella modalità di valutazione della stazionarietà, di date di training nel caso multitest.

h

[*Valore numerico intero (singolo o vettore di valori)*]
Offset delle date di test nella modalità multitest, di date di training nel caso di valutazione della stazionarietà.

trindex

[*Lista di vettori di valori numerici interi*]
Contiene le informazioni per unire i training set nella modalità *multitest sperimentale*[11]. I valori di ogni vettore inserito corrispondono agli indici[12] delle posizioni degli elementi presenti nel vettore k[13]: i training set generati in riferimento a tali valori saranno quelli che dovranno essere uniti.
Tutto ciò è stato introdotto per avere la possibilità di unire contributi di differenti set ed evitare di avere uno sbilanciamento nei risultati a favore solo delle caratteristiche di certi particolari set[14]. Questo tipo di esecuzione è ancora in una fase primordiale, perciò il suo sviluppo non è per ora stabile e non ha prodotto ancora risultati significativi.
Può lavorare anche senza necessità di unione dei test set ed in questo caso si parla di modalità *multitraining*: per ogni data di test (ognuna attribuita ad un valore di h), si costruiscono tanti training set quanti sono i vettori di *trindex*. Ogni vettore specifica gli indici degli elementi del vettore k che devono essere presi in considerazione per costruire ed unire i rispettivi training set ed ottenerne uno solo.

teindex

[*Lista di vettori di valori numerici interi*]
Contiene le informazioni per unire i test set nella modalità *multitest sperimentale*. I valori di ogni vettore inserito corrispondono

[11] Concettualmente differente da quello del parametro *mode*.

[12] Si considera come primo indice lo *0*.

[13] La costruzione e l'estrazione delle date di training e di test secondo il significato dei vettori k e h seguono in questo caso la logica della modalità standard del multitest del parametro *mode*.

[14] Come già detto più volte, poiché il problema degli abbandoni è molto sbilanciato a favore dei fedeli che costituiscono l'indiscussa maggioranza, questo porta i training set ad abituarsi a riconoscere bene i fedeli, ma poi a classificare tali anche la maggior parte degli abbandoni.

agli indici delle posizioni degli elementi presenti nel vettore h: i test set generati in riferimento a tali valori saranno quelli che dovranno essere uniti[15]. Non è preso in considerazione se il parametro *trindex* è nullo (in tal caso si procederà all'esecuzione nella modalità standard).

thr

[*Valore numerico decimale (singolo o vettore di valori)*]
Soglia di etichettatura delle istanze, usata nella formula[16]:

$$f_e \leqslant thr \cdot f_v$$

dove f_e è il fatturato nel periodo di etichettatura e f_v è quello nel periodo di vita.
Con un valore di soglia pari a *0* vengono quindi etichettati come abbandoni i clienti che non hanno effettuato alcun acquisto.

giorni

[*Valore booleano*]
Indica se il calcolo della media del fatturato deve essere eseguito sui giorni:

FALSE:

$$avg(fatturato) = \frac{[fatturato\ totale]}{[\#\ acquisti]}$$

TRUE:

$$avg(fatturato) = \frac{[fatturato\ totale]}{[\#\ giorni]}$$

distance

[*Valore booleano*]
Indica se devono essere accettate solo combinazioni di training e test set con data di test distante rispetto alla data di training almeno tanti mesi in avanti quanti sono i mesi di abbandono.

[15] Vedere il parametro *multitest2* per dettagli sul suo uso.
[16] Vedere il capitolo precedente.

multitest2

[*Valore booleano*]
In caso di *trindex* e *teindex* viene utilizzata una modalità di *multitest sperimentale* per unire i training ed i test set; tale parametro indica se utilizzare la modalità multitest sperimentale di default o quella alternativa. Come per il multitest standard[17], gli elementi del vettore k saranno attribuiti alla generazione del training set, mentre quelli del vettore h si riferiranno al test set, seguendo in linea generale lo stesso principio per la loro costruzione.

FALSE: *multitest sperimentale di default*
Per ogni data di test si uniscono i training set ricavati da essa secondo gli indici del vettore *trindex*, seguendo la modalità del *multitraining*; successivamente i training set precedentemente ottenuti vengono fusi in uno unico.
I test set calcolati vengono poi uniti secondo gli indici di *teindex*.

TRUE: *multitest sperimentale alternativo*
Le date vengono calcolate per ogni combinazione tra i vettori all'interno di *trindex* e quelli di *teindex*. Per il vettore di *trindex* vengono calcolate tante date di training quanti sono i suoi elementi: ognuna di essa è ottenuta sottraendo il valore dell'elemento ai mesi della data di riferimento.
Per il vettore di *teindex* vengono calcolate tante date di test quanti sono i suoi elementi: ognuna di essa è ottenuta aggiungendo il valore dell'elemento ai mesi della data di riferimento.
Infine tutti i training set vengono uniti in uno unico, la stessa cosa viene fatta per i test set, ottenendone uno solo.

Parametri per la *security*:

pbe

[*Elenco di valori numerici interi*]
Password PBE necessaria per crittografare i modelli ed altre

[17] Relativo al parametro *mode*.

password utilizzate dal sistema. È tenuta nel file di configurazione anch'essa crittografata, quindi occorre conoscere in anticipo la sua forma non in chiaro.

3.2.2 DMU

La Data Mining Unit rappresenta il deployment del nostro sistema; il suo compito è quello di usare i modelli precedentemente realizzati tramite il BusinessFramework ed applicarli a nuove istanze di dati sconosciuti, al fine di effettuare la reale previsione.
Il processo di previsione consiste in:

1. generazione del dataset
2. applicazione di uno o più modelli compatibili
3. scrittura dell'output su tabella relazionale

Un modello può essere caricato solo se è compatibile con i criteri di estrazione[18], che sono contenuti nel file descrittore del modello.
La DMU ha anche la funzionalità di avvisare tramite e-mail (ad un indirizzo specificato precedentemente) nel caso vi sia un eccessivo numero di errori di previsione, ossia quando più del 25% dei clienti classificati come abbandoni ha invece effettuato acquisti anche quando non avrebbe dovuto.
Il risultato finale sarà contenuto nello stesso database dei dati sorgenti nella tabella *dm.clienti*, dove ad ogni cliente corrisponderà una F in caso di fedeltà o A in caso di previsione di abbandono.

Parametri La DMU, a differenza del BusinessFramework, utilizza solo i parametri relativi ad *input*, *database* e *security*, mentre la categoria *variables* non è considerata[19].
Per quanto riguarda la sezione *model*, i campi sono differenti e vengono utilizzati esclusivamente due nuovi parametri:

[18] In genere ci si riferisce agli stessi *acquisti*, *vita* e *abbandono*, dato che servono per estrarre il medesimo campione di clienti.

[19] Occorre tuttavia che la DMU faccia ricorso alle medesime variabili usate dal modello che deve essere applicato.

soloabbandoni
[*Valore booleano*]
Indica se nella tabella di output devono essere inseriti solo i clienti previsti come abbandoni.

unionlista
[*Valore booleano*]
Serve per stabilire il criterio di previsione nel caso di utilizzo di più modelli.

FALSE: viene decretato l'abbandono esclusivamente all'unanimità di tutti i modelli utilizzati.

TRUE: il valore della classe che dovrà essere assegnato sarà quello di maggioranza risultante tra tutti i modelli caricati (criterio ad elezione)[20].

Pur utilizzando distinti parametri o solo un sottoinsieme di essi, il file xml di configurazione di BusinessFramework e DMU presenta la stessa struttura per entrambi i tool: valori inseriti non necessari vengono ignorati.

3.2.3 XMLConfigurator

L'XMLConfigurator permette di editare in maniera agevole il file di configurazione *conf.xml* necessario a BusinessFramework e DMU, riducendo anche la probabilità di commettere errori di impostazione.
È composto da una serie di caselle di testo per i valori testuali o numerici, da checkbox per i valori booleani e da radiobutton per le scelte esclusive.
Grazie a questo tool è possibile inserire la password per il database nella sua forma in chiaro (anche se verrà visualizzata in maniera celata durante la sua scrittura), penserà poi l'XMLConfigurator a crittografarla automaticamente al momento della creazione dell'xml.
Il file prodotto dovrà poi andare a sostituire quello già presente nei vari tool.

[20] In questo caso, per non avere un pareggio, è bene utilizzare un numero dispari di modelli.

3.2.4 ServerLicense

Applicazione non destinata all'utenza, per la gestione del database delle licenze.
Quest'ultimo contiene un'unica tabella in cui sono raccolti:

id
: Indirizzo IP di una macchina che utilizza BusinessFramework o DMU.

meta
: Flag booleano: impostato a true si riferisce a BusinessFramework, false a DMU.

init time
: Data di inizio della licenza.

license
: Stringa alfanumerica di 16 caratteri che costituisce il codice della licenza.

L'operazione di aggiunta di licenze avviene tramite GUI; successivamente viene generato in automatico il codice corrispondente e viene inserita la data corrente come quella di inizio di utilizzo della licenza.

3.2.5 ServerDMU

Il ServerDMU è adibito a controllore della licenza. Utilizzando un particolare protocollo, esso viene interrogato dal BusinessFramework o dalla DMU al momento del loro avvio; lato server si risale all'identità del client attraverso il suo indirizzo IP ed al valore del flag inviato.
Se la licenza è valida e non è scaduta, l'avvio viene convalidato e l'applicazione può proseguire normalmente la propria esecuzione, altrimenti viene avviata una procedura particolare per il rinnovo della licenza.

3.3 Interazione tra i tool

Dalle descrizioni dei singoli tool, emerge che essi lavorano in maniera indipendente; l'unico mezzo di interazione è costituito dai risultati che

essi stessi producono, ed è quindi una forma statica di interoperabilità. Se consideriamo solamente i tool fondamentali (BusinessFramework, DMU e XMLConfigurator) l'architettura complessiva è quella mostrata in figura[21].

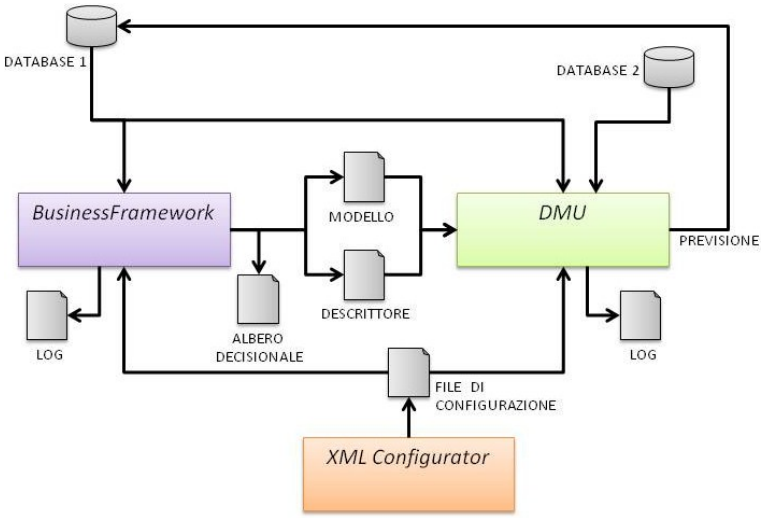

Figura 3.3: *Architettura della suite*

Il modello ed il suo descrittore[22] prodotti dal BusinessFramework vengono utilizzati dalla DMU per attuare la previsione su dati sconosciuti; quest'ultima viene scritta nel database. Questi due tool tengono aggiornato un file di log ad ogni operazione eseguita; per funzionare, entrambi hanno bisogno di un proprio file di configurazione creato separatamente dall'XMLConfigurator. Il decision tree prodotto dal BusinessFramework per ogni modello è solo di aiuto allo sviluppatore per comprendere meglio le caratteristiche del modello, ma non serve ad alcun altro tool.

[21] Come illustrato in figura, i dati possono venire da database differenti (per esempio, uno utilizzato per addestrare e valutare il sistema, l'altro invece contenente i nuovi dati da predire).

[22] Si è generalizzato, ma possono essere presi anche più modelli.

Capitolo 4

Sviluppo del sistema

In questo capitolo verranno illustrate le principali modifiche effettuate nel corso di questa tesi alla suite descritta nel precedente capitolo.
Lo sviluppo di tale software si è articolato fondamentalmente in due fasi: inizialmente si è cercato di attuare dei cambiamenti alla suite originale per migliorarla e per renderla più completa, successivamente, quando si è poi affrontato il caso aziendale di cui ci stiamo occupando, ci è accorti che al sistema mancavano alcuni requisiti e parti necessarie per risolvere in maniera adeguata tale problema e riuscire così alla fine ad ottenere un buon modello che attuasse la previsione voluta.
Pertanto in una prima fase si è cercato di migliorarlo sotto il profilo generale mentre, nella fase seguente, si sono apportati cambiamenti suggeriti dall'analisi del caso aziendale che stiamo trattando.
Tutto questo ha portato ad avere una soluzione software stabile e molto più completa, adatta a produrre un'adeguata previsione dei clienti a rischio di abbandono che vedremo nel prossimo capitolo.

4.1 Miglioramenti ed aggiunte alla suite di base

Un'ingente parte del lavoro di tesi è consistito nel miglioramento e nel conseguente sviluppo dei tool della suite prototipale; infatti, pur essendo essa già in fase avanzata, potevano essere apportate ancora molte altre migliorie, inoltre questo sistema era predisposto solo per

alcune funzioni di base. L'obiettivo finale è ottenere una suite che si adatti al meglio alla maggior parte dei problemi di Data Mining più comuni e sia capace di affrontarli nella maniera più generica e modulare possibile, continuando comunque ad offrire soluzioni di elevate prestazioni. Analizzare di volta in volta problemi differenti offre un valido spunto per espanderla e renderla sempre più completa e versatile possibile.

Nel corso di questa tesi si sono principalmente portate a termine le seguenti modifiche alla suite:

- **Revisione ed ingegnerizzazione del codice**
- **Sviluppo dei tool di base**
- **Integrazione ed interoperabilità tra i tool**
- **Incremento delle prestazioni del BusinessFramework**
- **Automazione della DMU in modalità test**
- **Implementazione della DMU in un altro linguaggio**

Tutti questi miglioramenti fanno parte solo di una prima fase dell'intero lavoro di sviluppo del software che si è attuato; più avanti verranno anche illustrate tutte quelle modifiche che si sono rese necessarie dopo l'analisi[1] del caso aziendale.

4.1.1 Revisione ed ingegnerizzazione del codice

La suite di base[2] è strutturata in maniera modulare, tuttavia tempi stretti, frequenti scadenze di presentazione di risultati ed uno sviluppo in modo incrementale, dovute a continue aggiunte stabilite all'occorrenza, hanno portato a far sì che essa presenti un codice ridondante, non ottimizzato e spesso di difficile comprensione.
Perciò, prima di qualsiasi altra modifica, è stato necessario rivedere l'intero codice per cercare di reingegnerizzazione al meglio tutto il

[1] Ci si riferisce all'analisi effettuata nel capitolo 2.

[2] Il sistema software da cui si è poi iniziato ad attuare tutto il lavoro riguardante questa tesi.

lavoro fatto fino a quel momento e risolvere tutte le sue ridondanze accumulatosi nel tempo.
Questo lavoro di revisione non mira nè ad introdurre nuove funzionalità nè ad un aumentarne le prestazioni: l'unico a trarne beneficio è esclusivamente il programmatore perché in questo modo la suite viene meglio predisposta alle future modifiche, agevolandone lo sviluppo.

Tra le operazioni più significative apportate vi sono:

Riduzione del codice
: Parti non necessarie vengono soppresse, quelle ripetute inutilmente vengono usate un'unica volta.

Accorpamento di parti comuni
: Parti comuni a più classi vengono accorpate in uniche classi da cui si possono fare derivare tutte le restanti[3].

Programmazione ad un livello più semplice
: Si è cercato di incapsulare le parti più complesse per poi astrarre, derivare e configurare ad un livello superiore solo gli aspetti più semplici o specifici.

Miglioramento della leggibilità del codice
: A livello sintattico sono stati utilizzati meccanismi di implementazione equivalenti ma più sintetici; per aumentare la visibilità delle istruzioni si sono eliminate specificazioni inutili e si sono rinominati metodi per accorciarne i nomi, oltre a separare in maniera adeguata le varie parti delle quali sono costituite le istruzioni. A volte sono stati assegnati anche nomi più consoni ad alcune variabili o metodi, per permettere di comprendere meglio che cosa questi rappresentino o la loro funzione.

Meccanismi di sviluppo più veloci
: Parti molto comuni ripetute più volte vengono implementate separatamente una volta soltanto per essere richiamate in modo veloce all'occorrenza, conferendo compattezza al codice. Inoltre

[3]Si sfrutta principalmente l'ereditarietà di java e dei linguaggi object-oriented in generale.

sono stati aggiunti semplici flag per attivare o disattivare intere parti o funzionalità di codice a scopo di debugging.

Riorganizzazione
I moduli che scandiscono le varie parti dell'applicazione sono stati accorpati ed organizzati in maniera differente secondo una logica più adeguata, in modo tale anche da ridurre il numero di quelli principali ed avere meno dispersione derivante dalle varie suddivisioni.

Gestione immediata di tag e path delle risorse
Sono stati raggruppati tutti i riferimenti di tag dell'xml e percorsi delle cartelle contenenti input ed output per permettere di gestirne in maniera rapida ed immediata l'inserimento o la modifica: venendo usati per riferimento e non più direttamente con il loro nome, si consente di variarli una volta soltanto da un unico punto, tramite costanti.

Tutto questo ha portato ad ottenere un codice con un volume ridotto di 1/4 rispetto a quello di partenza. Inoltre l'aver introdotto meccanismi più veloci di implementazione ed aver reso la visualizzazione del codice più immediata e facile da comprendere va a vantaggio degli sviluppi futuri, soprattutto se essi devono essere apportati da nuovi programmatori che non conoscono ancora la logica di funzionamento della suite.

4.1.2 Sviluppo dei tool di base

I tool sono stati migliorati rispetto alla versione di base, soprattutto per quanto riguarda l'XMLConfigurator, anche se è uno strumento secondario ed accessorio rispetto agli altri.
Sono state aggiunte alcune caratteristiche semplici ma estremamente utili, come l'estrazione automatica dei dati presenti in un file di configurazione già esistente e la successiva reimpostazione dei parametri e delle opzioni di scelta della schermata in base ad essi: questo evita di compiere ogni volta l'irritante e lunga operazione di ricompilare tutti i campi anche a fronte di piccole variazioni nei parametri di input.
Un altro aspetto modificato in tale strumento è il fatto di avere diversificato, in base al tool target, la visualizzazione della GUI e soprattutto

la struttura dell'xml prodotto: in questo modo si avrà la possibilità di impostare solo i parametri strettamente necessari a seconda che si tratti del BusinessFramework o della DMU, in maniera da non confondere l'utente (soprattutto per quanto riguarda quest'ultimo tool, dato che ha molte meno voci che devono essere configurate).

4.1.3 Integrazione ed interoperabilità tra i tool

I vari strumenti lavorano efficientemente, tuttavia in maniera totalmente indipendente tra di loro: il loro unico punto in comune è dato spesso da uno o più file prodotti come output che saranno poi usati come input di un altro tool. Un caso particolare è l'XMLConfigurator, usato sia per il BusinessFramework, sia per la DMU. Esso deve essere avviato come applicazione indipendente e produce poi come output un file, che deve infine essere spostato manualmente e sostituito con quello già presente nei vari tool che lo richiedono; questa operazione va ripetuta ogni volta che si vuole fare un cambiamento nei parametri di input.
L'idea è di non pensare più all'XMLConfigurator come ad un'applicazione a se stante, ma predisporlo per essere visto come un componente che può essere integrato nel BusinessFramework e nella DMU: occorrerà quindi incapsularlo in una libreria.
In questo modo sia hanno diversi vantaggi:

- I parametri possono essere impostati prima di ogni avvio della simulazione direttamente dall'interno dello specifico tool, permettendo una veloce configurazione.

- Parti di codice comune e replicato in tutti i tool che servivano per operazioni di estrazione di dati da file xml di particolari tag vengono eliminati: esse vengono gestite unicamente dalla libreria *xmlconfigurator.jar* e non si è quindi più costretti ad inserire inutilmente lo stesso codice nei vari tool.

- Agisce direttamente sul file di configurazione nella posizione in cui verrà utilizzato dal tool a cui è destinato, senza richiedere altre operazioni manuali di spostamento e sostituzione.

4.1.4 Incremento delle prestazioni del Business-Framework

La parte principale relativa all'esecuzione è stata sviluppata seguendo una logica molto basilare, che ha semplificato la progettazione, però questo ha portato ad avere un sistema che fa un uso inefficiente del tempo e dello spazio necessari per l'elaborazione.
Nel BusinessFramework, a partire dai parametri k e h, viene generata una serie di training e test set e su questi poi si applicano gli algoritmi di Data Mining: se l'elaborazione è basata su singoli valori per entrambi i parametri, il sistema funziona semplicemente usando un unico task che si basa su questi 2 valori per creare i corrispondenti dataset, tuttavia se si vogliono usare valori multipli, l'engine poi deve incrociare questi ultimi, effettuando tutte le combinazioni possibili tra i valori del vettore h ed i valori del vettore k (l'associazione di training e test set a questi sarà stabilita poi dalla modalità di generazione).
In quest'ultima situazione, per ogni combinazione è associato un nuovo task, quindi l'engine ricrea training e test set: questo è stato un modo veloce di progettare il sistema, ma ci si rende conto che quest'ultimo compie del lavoro in più che risulta essere inutile. Infatti anche se si hanno test set differenti, questi vengono generati in riferimento allo stesso training set, pertanto la rigenerazione di quest'ultimo porta ad un uso non ottimale del tempo (l'elaborazione è molto più lunga perché è ripetuta) e della memoria fisica (nel database vengono generate tabelle temporanee per i dataset creati da ogni task che però alla fine risultano essere per la maggior parte uguali tra loro).
L'engine è stato quindi modificato in modo tale da riconoscere training set già elaborati: nel caso si verificasse quest'ultima situazione, può riutilizzarli, saltando quindi la loro rigenerazione (le tabelle risultanti ad ogni elaborazione rimangono comunque nel database); grazie a questo, sul database viene lasciata un'unica tabella di training set per ogni test set multiplo ed inoltre vengono eliminati i tempi di rielaborazione per training set già calcolati precedentemente.

Per esempio, consideriamo un vettore k composto da 3 valori, ed un vettore h da 4.

Nello specifico prendiamo:

k: 18, 12, 6

h: 18, 15, 12, 6

Con questi valori si ottengono 12 combinazioni possibili tra i due vettori e quindi altrettanti task, ognuno dei quali dovrà calcolare 2 dataset (uno di training e uno di test).
Considerando tutti i vari abbinamenti, il numero totale di dataset da elaborare risulterebbe quindi:

$$O(sistema) = 2(N_K \cdot N_H)$$

dove N_K è la cardinalità del vettore k, N_H quella del vettore h.
Nel nostro caso specifico i dataset sarebbero quindi 24.
Ottimizzando invece l'elaborazione ai soli training set, la nuova complessità sarà:

$$O(multitest) = (N_K \cdot N_H) + N_{NK} \quad N_K \leqslant N_{NK} \leqslant N_K N_H$$
$$O(stazionarieta) = N_H(N_K + 1)$$

dove N_{NK} è il numero di date di training generate dai valori del vettore k che non coincidono tra loro.
Riportato al nostro esempio, avremo una quantità di dataset compresa tra 15 e 24 nel caso di multitest e 16 nel caso di valutazione della stazionarietà.
Questo mostra una notevole riduzione della complessità, tuttavia questi calcoli poi dipenderanno da diversi fattori, ed i tempi per ogni singola elaborazione possono essere molto eterogenei tra di loro.
Successivamente è stata ottimizzata anche la generazione dei test set, adottando lo stesso criterio.
Sia per quanto riguarda la valutazione della stazionarietà, sia per il multitest, la complessità è stata così ridotta a:

$$O(ottimizzazione) = N_H + N_{NK} \quad N_K \leqslant N_{NK} \leqslant N_K N_H$$

Ovviamente nel caso del multitest, h si riferirà al test set e k al training, viceversa per la valutazione della stazionarietà.
Riferendoci sempre ai valori di esempio riportati precedentemente, il

sistema dovrà calcolare da un minimo di 7 (caso altamente improbabile) ad un massimo di 16 dataset in entrambe le modalità.

Nei vari casi, il numero dei dataset che il sistema ha dovuto calcolare è stato:

	Multitest			**Stazionarietà**		
OTTIMIZZAZ.	*nessuna*	*training*	*training&test*	*nessuna*	*training*	*training&test*
N training	12	9	9	12	4	4
N test	12	12	4	12	12	9
TOTALE	24	21	13	24	16	13

Per quanto riguarda la velocità di elaborazione, eseguendo tre prove con i valori di esempio, si sono ottenuti i seguenti risultati:

	Multitest			**Stazionarietà**		
OTTIMIZZAZ.	*nessuna*	*training*	*training&test*	*nessuna*	*training*	*training&test*
Test 1	790	688	422	844	575	488
Test 2	796	674	432	848	539	482
Test 3	713	658	413	811	572	458
MEDIA	766,3 s	673,3 s	422,3 s	834,3 s	562 s	476 s

Si può notare come, già con l'ottimizzazione per il solo training set, le prestazioni del sistema siano state migliorate per quanto riguarda la modalità multitest, e come si sia ottenuto poi un risultato ancora più vantaggioso per la valutazione della stazionarietà.

Questi risultati ovviamente vogliono mettere in luce solo l'ottimizzazione ottenuta: le differenze tra i vari valori non dipendono nello specifico da multitest/stazionarietà (in altri contesti il vantaggio portato potrebbe essere invertito tra i due), ma dai dati sorgenti, dalla data di riferimento, dai valori e dalla cardinalità dei vettori k e h; questo si evince sopratutto nell'ottimizzazione parziale del solo training set, mentre nel caso dell'ottimizzazione su entrambi i tipi di set queste differenze sono molto meno marcate.
L'aleatorietà dei fattori che contribuiscono al tempo di elaborazione la si può notare nell'ottimizzazione per tutti i dataset, se si comparano i risultati tra multitest e stazionarietà: in quest'ultima modalità, nel caso riportato, il sistema impiega un tempo leggermente superiore nonostante venga calcolato un dataset in meno rispetto all'altra modalità, tuttavia si può vedere in conclusione come i tempi di elaborazione siano stati quasi dimezzati in entrambi i casi rispetto al sistema di partenza.

4.1.5 Automazione della DMU in modalità test

Il BusinessFramework ha la capacità di generare molti modelli nel corso di una singola esecuzione e ciò avviene combinando, calcolando ed utilizzando più date di training e test; questo è fondamentale per mettere a confronto le prestazioni dei modelli e scegliere poi i migliori. La DMU, invece, è predisposta per applicare uno o più di questi ultimi solo per una singola data di riferimento, tuttavia la si potrebbe utilizzare anche per testare maggiormente e concretamente le prestazioni di un determinato modello ancora nella fase preliminare di valutazione. Questo si potrebbe realizzare verificando gli errori commessi dal modello ad intervalli regolari per tutto lo storico disponibile; per esempio lo possiamo applicare dal 2005 al 2010 per ogni bimestre ed osservare quindi se questo è realmente affidabile.
Per effettuare ciò non si è voluto modificare il controller della DMU, perché la funzione basilare di quest'ultima è quella di applicare il classificatore ai dati sconosciuti per una sola data di riferimento, dovendo dare un'unica previsione; inoltre non si vuole aggiungere complessità all'engine, poiché può essere possibile che esso debba essere modificato successivamente secondo richieste dell'azienda che ne dovrà fare uso come deployment finale. Questa funzionalità deve essere quindi solo di ausilio e di supporto al BusinessFramework, per comprendere meglio le prestazioni dei modelli generati, svincolandosi perciò dallo scopo per cui essa è stato realizzata che è quello di attuare la reale e definitiva previsione per nuovi dati sconosciuti.

L'applicazione viene avviata da una classe main secondaria, che, una volta lanciata, esegue i seguenti passi:

1. Preleva la data di previsione dal file di configurazione e la memorizza.

2. Controlla che la data impostata sia inferiore alla data massima; quest'ultima deve essere cablata nel codice come costante.

3. Il file di configurazione viene aggiornato[4] con la nuova data.

[4]Si intende il contenuto del tag *data*.

4. L'applicazione java primaria avvia un'altra istanza di DMU che verrà eseguita in maniera indipendente; tale istanza viene lanciata da file jar esterno includendo l'opzione *multistart*[5] e specificando inoltre il caso di studio.

5. Il sistema principale si mette in attesa della fine dell'esecuzione del processo.

6. L'istanza di DMU viene eseguita immediatamente, saltando il frame iniziale. Effettuata la predizione, si chiude automaticamente da sola, facendo terminare il processo.

7. L'applicazione principale riprende l'esecuzione; aggiunge un determinato gap di mesi impostato nel codice (nel nostro caso *2*) alla data usata e tiene quest'ultima in memoria.

8. Il processo viene ripetuto dal punto 2, fin quando verrà superata la data massima impostata; in quest'ultimo caso l'intera applicazione termina.

Questo processo permette di verificare un modello ad intervalli regolari e fitti per tutta la durata dello storico, senza intaccare la struttura principale della restante DMU, grazie, come si è visto, all'aggiunta solo di un gestore di singole istanze di tale tool, le quali vengono lanciate una per volta sequenzialmente. Ad ogni iterazione la data nel file di configurazione viene aggiornata, pertanto la DMU correntemente in esecuzione baserà la sua previsione su una data diversa rispetto a quella dell'istanza che è precedentemente terminata.

4.1.6 Implementazione della DMU in un altro linguaggio

Per questioni tecniche potrebbe tornare utile avere una versione della DMU scritta in un linguaggio differente. Infatti, mentre il Business-Framework è uno strumento diretto ad un'utenza esperta (ci sono numerosi parametri da impostare, che richiedono conoscenze nel campo del Data Mining), la DMU è un tool che:

[5] Occorre specificarla come argomento.

- deve avere più ampia diffusione
- deve essere usato più frequentemente
- è rivolto ad un utente medio

La DMU non deve fare altro che applicare in maniera automatica un modello ogni qualvolta si hanno dati nuovi da analizzare. Questo spesso porta a dover usare tale tool in realtà differenti, su sistemi operativi e nelle condizioni più diverse, considerando poi che l'utente finale potrebbe avere particolari esigenze che potrebbero richiedere qualche modifica ad hoc di tale strumento.
Utilizzare java come linguaggio di programmazione porta indubbiamente numerosi vantaggi, tra cui la portabilità su quasi tutti i sistemi operativi in circolazione, nonostante ciò, avere a disposizione una DMU implementata in un linguaggio diverso può essere comunque veramente utile ed in determinati casi potrebbe essere meglio prendere in considerazione di utilizzare questa versione alternativa rispetto a quella originale.

Per tale operazione è stato scelto *C#*, un linguaggio molto simile a java, facente parte dell'ambiente di sviluppo del *.NET Framework* della Microsoft[6].
L'utilità di avere a disposizione una DMU implementata in entrambi i linguaggi, risiede nel fatto che java e C# utilizzano spesso tecnologie diverse per interfacciarsi o interagire con le varie risorse, nel nostro caso poi è di vitale importanza l'interazione con il database. Quindi in base alle condizioni ed alla realtà in cui dovrà operare la DMU, si potrà scegliere la versione che meglio si adatta all'occasione: potrebbe infatti essere necessario fare modifiche personalizzate che con un linguaggio risulterebbe problematico attuare mentre potrebbe essere più semplice attuare con l'altro; non è da escludere inoltre il fatto di dover usare particolari tipi di driver a discapito di altri per collegarsi al database, variando poi anche in base anche al DBMS che lo gestisce. Con java è necessario poi utilizzare driver JDBC, però a volte potrebbe essere più efficace usare driver ODBC o nativi per .NET e questa scelta influisce notevolmente poi sulle prestazioni in termini di tempo

[6]Vedere riferimento bibliografico [10].

impiegato per interagire con il database. Occorre considerare inoltre che a volte potrebbe essere più adeguato usare strumenti in C# nel caso la DMU debba essere eseguita in un ambiente Microsoft in cui le risorse e le connessioni ai DB vengono gestite con determinati criteri.

Detto ciò, analizziamo innanzitutto cosa occorre per replicare nella nuova versione gli stessi servizi e funzioni.
Uno dei punti chiave è che tutto il sistema utilizza *WEKA*, necessario per applicare gli algoritmi di Data Mining utilizzati: il tutto è racchiuso all'interno della libreria *weka.jar*. Essa, come si può intuire dall'estensione, è scritta in java, pertanto sorge un problema di base, dato dalla probabile incompatibilità con linguaggi differenti.
Per risolvere questo problema ci si è affidati a *IKVM.NET*, un'applicazione capace di incapsulare e convertire programmi o librerie java in linguaggi .NET, a partire dal bytecode di programmi java compilati[7].
Per attuare questa operazione con la libreria di WEKA, è sufficiente usare il comando:

```
ikvmc -target:library weka.jar
```

e si otterrà in output il file *weka.dll*, utilizzabile in ambiente .NET. A questo punto sarà sufficiente importare tale file nel proprio progetto all'interno dell'IDE di sviluppo .NET, inserendolo come riferimento; oltre a questo dovranno essere aggiunte anche tutte le *dll* di IKVM per permettere l'esecuzione e l'interpretazione delle librerie derivate da java; la libreria principale che permette ciò è *IKVM.Runtime.dll*.
Anche per la classe *d2d.dm.model.ModelDescriptor* occorre applicare lo stesso procedimento; infatti i descrittori serializzati su file come output del BusinessFramework dei quali poi si serve la DMU per estrarre informazioni sui modelli che utilizzerà, devono fare riferimento alla medesima classe (deve essere identica, compreso il package in cui essa è inserita), altrimenti non è permesso attuare il cast quando si estrae l'oggetto dal file in cui esso è salvato.
Pertanto occorre realizzare una libreria java dal nome *d2d.dm.model.jar*, in cui mettiamo solo tale classe e poi la trasformiamo in dll per permettere che venga utilizza dalla DMU in C#:

[7] Vedere riferimento bibliografico [8].

```
ikvmc -target:library d2d.dm.model.jar
```

e si otterrà *d2d.dm.model.dll* che dovrà essere aggiunta ai riferimenti dell'applicazione C#.

Analisi dell'ambiente di sviluppo e del software necessario

La versione di .NET che preferibilmente si vuole sfruttare è l'ultima, ossia la 4, poiché, a differenza delle precedenti, possiede maggiori meccanismi di sincronizzazione e di gestione della concorrenza (liste concorrenti e barriere) dei quali si è già fatto ricorso nell'implementazione della versione della DMU in java. L'unico inconveniente che potrebbe incorrere nell'utilizzare l'ultima versione di .NET risiede nella probabile incompatibilità di sistemi non ancora aggiornati (o che per altri motivi non la supportano) nei quali dovrà essere installata la DMU.
L'IDE di sviluppo per linguaggi appartenenti al .NET Framework è *Visual Studio*: se prendiamo come riferimento C# 4, allora occorre la versione 2010, poiché quella precedente, cioè la 2008, supporta al massimo .NET 3.5.

Un obiettivo che si vuole raggiungere è utilizzare questa nuova DMU anche su *Ubuntu*, una particolare versione gratuita di Linux basata su Debian, ampiamente utilizzato soprattutto per le caratteristiche di sicurezza, stabilità e per essere gratuito[8]. A tal scopo si sfrutta *Mono*, un'implementazione open source del .NET Framework, presente su Ubuntu[9]. Tuttavia, mentre Mono è in continua evoluzione ed è arrivato a supportare le versioni più recenti di .NET come la 4, molte versioni di Ubuntu non arrivano ad integrare Mono fino alla release più recente.
Per questo motivo, per potere sfruttare a pieno le potenzialità di .NET 4 con Mono, occorre avere installato almeno la versione 11.10 di Ubuntu: la versione precedente, cioè la 11.04, permette di aggiornare Mono in maniera automatica tramite repository fino alla versione 2.6.7, che offre il supporto solo fino a .NET 3.5.
Per rendere poi possibile l'utilizzo delle Windows Forms, ossia la parte grafica dei sistemi .NET, in Ubuntu occorre installarne l'apposita

[8]Vedere riferimento bibliografico [5].
[9]Vedere riferimento bibliografico [1].

versione per Mono.

Per le Windows Forms 2 occorre usare il comando:

```
apt-get install libmono-winforms2.0-cil
```

Invece per la versione 4 il comando è:

```
apt-get install libmono-windows-forms4.0-cil
```

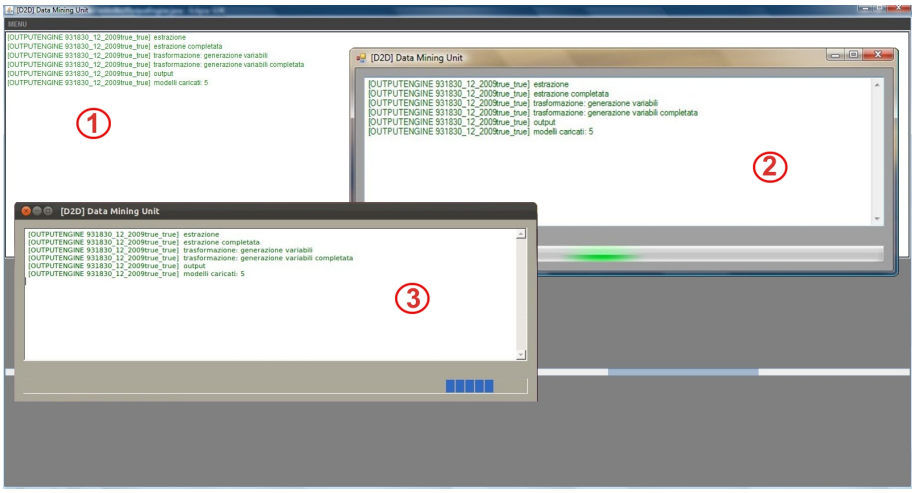

Figura 4.1: *DMU: [1] java [2] C# .NET (Windows) [3] C# Mono (Ubuntu)*

Come DBMS si utilizzerà PostgreSQL; stabilito questo, possiamo inizialmente considerare di utilizzare i driver nativi *Npgsql*, che sono specifici per PostgreSQL e si usano sotto forma di libreria .NET[10].
Si sono riscontrati alcuni problemi nel richiamare driver ODBC (nel nostro caso *pgODBC* per PostgreSQL): per essere sfruttati, essi sono vincolati ad essere utilizzati in un ambiente Microsoft, per cui la scelta migliore è sembrata quella dei driver Npgsql. Un inconveniente di questa decisione è l'impossibilità di collegamento con altri tipi di database, poiché non è una soluzione generale: l'implementazione viene impostata in questo modo sull'uso di tali driver che sono specifici

[10]Vedere riferimento bibliografico [13].

per PostgreSQL. Si è pensato quindi di sviluppare un'apposita classe per la gestione della connessione con il database e per le relative operazioni su di esso, che facesse da wrapper, cioè che incapsulasse ogni richiamo alla specifica libreria Npgsql senza farne trasparire il suo uso, ma semplicemente fornendo metodi generici. In questo modo se in futuro si vorrà utilizzare qualche altra tecnologia basterà semplicemente cambiare l'implementazione dei metodi di interrogazione e collegamento di questa classe wrapper, per il resto, i metodi utilizzati all'interno dell'applicazione principale, rimarranno gli stessi. Per potere utilizzare Npgsql occorre tuttavia una versione di Mono abbastanza recente (non è compatibile con le versioni di Mono supportate da Ubuntu inferiore a 11.10).

Incompatibilità tra linguaggi

Durante lo sviluppo di tale versione sono sorti diversi problemi dovuti all'incompatibilità tra l'ambiente java e .NET, e ci si è resi conto che non tutto può essere trasportato in maniera semplice e diretta dall'uno all'altro.
Il punto critico rimane la connessione con il database: WEKA in genere può prelevare il dataset che utilizzerà da file (in genere del formato *arff*) ma permette anche di estrarre in maniera automatica e semplice il dataset di riferimento dal database, soltanto a partire dalla query di selezione.
Ovviamente ci interessa esclusivamente il secondo caso; la procedura è abbastanza standard e prevede di aprire una piccola e momentanea connessione tramite WEKA:

```
InstanceQuery query = new InstanceQuery();

query.setDatabaseURL(url);
query.setUsername(user);
query.setPassword(password);
query.setQuery(sql);

Instances set = query.retrieveInstances();
```

In cui *Instances* rappresenta un dataset, cioè l'oggetto contenente tutte le istanze a cui WEKA dovrà applicare il modello.
Bisogna considerare che anticipatamente occorre anche caricare a runtime il driver JDBC di PostgreSQL:

```
java.lang.Class.forName(''org.postgresql.Driver'');
```

Si è pensato dunque di realizzare tale parte esclusivamente in java per permettere di utilizzare i driver JDBC. Quindi si è provveduto inizialmente a rendere tale driver compatibile per l'ambiente .NET:

```
ikvmc -target:library postgresql-9.0-801.jdbc4.jar
```

Successivamente è stata realizzata una libreria a parte che contenesse un *DataLoader* che eseguisse le istruzioni viste precedentemente.
Tale libreria è stata poi convertita in dll, aggiungendo i vari riferimenti sia alla dll di WEKA, sia a quella del driver, necessari per la sua esecuzione:

```
ikvmc -target:library -reference:weka.dll
      -reference:postgresql-9.0-801.jdbc4.dll
      dataloader.jar
```

producendo in output *dataloader.dll*.

Sebbene tale libreria a runtime carichi con successo il driver ed imposti i vari parametri per la connessione, poi non riesce ad eseguire la query (ossia l'ultima istruzione), generando un'eccezione; questo fa presupporre che l'uso in maniera diretta di tale driver sia comunque incompatibile.
Bisognerebbe invece usare un *JDBC-ODBC bridge* che fa parte della *Sun JRE/SDK*, e poi creare nell'ambiente Microsoft un user ed una risorsa DSN relativa al database, però in questo modo si rimane molto legati a tale sistema operativo, escludendo la possibilità di applicarlo in Ubuntu con Mono.
La soluzione adottata è ricreare l'oggetto Instances in maniera "manuale", ossia prelevando i dati con una semplice query come in tutte

le altre occasioni, per poi creare per ogni osservazione ottenuta un'istanza del tipo richiesto da WEKA da inserire alla fine nell'oggetto Instances.

Prestazioni

Mettendo a confronto le due versioni della DMU, si può notare che le performance sono abbastanza differenti.
Sono stati eseguiti alcuni test, consistenti in 3 simulazioni con valori di *soloabbandoni* ed *unionlista* a *true*, nelle quali si sono applicati ai dati in entrambi i casi 5 modelli precedentemente realizzati con il BusinessFramework.
I tempi di esecuzione relativi alle due differenti versioni messe a confronto sono mostrati nella seguente tabella.

	Java	**.NET**
Test 1	237	279
Test 2	235	273
Test 3	240	274
MEDIA	237,3 s	275,3 s

Si ha una media di quasi 4 minuti a simulazione per la versione in java e circa mezzo minuto in più per la versione in C#.
Questo mette in evidenza che la versione in java rimane comunque la più veloce, abbattendo quindi la speranza che la nuova versione potesse portare prestazioni migliori, considerando poi che i tempi relativi alla versione in C# sono presi avviando l'applicazione al di fuori dell'ambiente di sviluppo, altrimenti essi sarebbero maggiori di 5-6 volte.

Bisogna comunque osservare che la versione .NET è stata implementata come trasposizione di quella in java[11], utilizzando gli stessi meccanismi; se invece si sfruttassero pienamente le potenzialità della libreria Npgsql, i tempi potrebbero ridursi notevolmente (per esempio si potrebbero utilizzare i metodi per inserire righe intere, invece di eseguire semplici query SQL).
Inoltre in alcuni punti del codice sono state utilizzate classi java per

[11] Ci si riferisce alla versione della DMU con il codice revisionato, ma senza l'integrazione dell'XMLConfigurator.

semplificare l'implementazione, tuttavia questo costringe ogni volta il sistema a dover caricare le librerie IKVM incaricate di interpretare tali richiami alle librerie java: questo porta ad un rallentamento dell'esecuzione. Cercando di fare meno ricorso a tali librerie quando non è strettamente necessario, si potrebbero avere dunque prestazioni più elevate.

Questo rimane comunque un primo prototipo, ossia un'applicazione base perfettamente funzionante in cui le prestazioni non interessano più di tanto poiché lo scopo principale era quello di vedere se era attuabile una trasposizione della DMU in un linguaggio differente: si potrà poi partire da questa prima versione base per effettuare di volta in volta modifiche ad hoc per renderla sempre più efficiente.

4.2 Modifiche alla suite per il caso aziendale trattato

Nella precedente sezione si sono illustrati i principali interventi e modifiche che occorreva apportare al sistema di base, indipendentemente dallo specifico problema di Data Mining che la suite doveva risolvere; era fondamentale eseguire tali miglioramenti strutturali prima di affrontare un nuovo caso di studio.
Con l'avvento poi del nuovo caso aziendale da risolvere sono emersi altri problemi, che non ci si era posti fino a tale momento o che non erano comunque richiesti da quello precedente.

Ora verranno quindi descritti i cambiamenti e le aggiunte effettuati in questa seconda fase di sviluppo della suite:

- **Gestione di più casi di studio**
- **Modulo per il campionamento**
- **Esecuzione automatica delle trasformazioni**
- **Ottimizzazione dell'accesso alle tabelle**
- **Nuove variabili**

▶ **Struttura del file di configurazione**

4.2.1 Gestione di più casi di studio

Per permettere di gestire il nuovo caso aziendale, invece di ricreare i tool in maniera ad hoc, cioè replicando tali applicazioni e successivamente sostituendo solo le parti specifiche del vecchio caso con quelle adatte al nuovo, è stato progettato ed implementato un meccanismo che permette di raggiungere al meglio l'obiettivo di generalizzazione che ci si è sempre imposti: il sistema permette di eseguire uno switch da un caso all'altro semplicemente impostando un valore identificativo nell'apposito campo all'interno del file di configurazione; è possibile eseguire ciò anche direttamente da interfaccia grafica tramite l'XML-Configurator.
Internamente al sistema viene cambiato un flag ed in questo modo il controller capisce quali variabili deve usare e quali sono le specifiche query che deve eseguire.
Per permettere questo è stato necessario un preventivo ed accurato lavoro di separazione logica dei vari moduli del sistema: bisognava infatti distinguere le parti comuni a tutti da quelle che invece si differenziano; nel secondo caso si è cercato di uniformare e generalizzare l'uso di tali parti attribuendo loro una struttura simile stabilita da una stessa interfaccia.
Il controller non fa altro che caricare il particolare gestore di un caso di studio che poi provvederà ad usare esclusivamente le parti specifiche del suo modulo per eseguire tutte le operazioni richieste.
In questo modo è possibile gestire ed aggiungere in maniera agevole un numero illimitato di casi per problemi di Data Mining simili.

4.2.2 Modulo per il campionamento

Dall'analisi sulla prima versione dei dati, ci si era accorti che il problema richiedeva un iniziale campionamento, pertanto è stato necessario aggiungere un modulo che attuasse tale fase, elaborando i dati prima che il sistema eseguisse ogni altra operazione di ETL.

Nella pratica, il procedimento per campionare si sviluppa nel seguente modo:

1. Si guarda quanti differenti clienti sono realmente presenti nelle fatture. Per eseguire ciò basta fare un conteggio dei clienti che compaiono in maniera distinta nella tabella *fatture*.

2. Si moltiplica il valore ottenuto per una costante compresa tra *0* e *1*, che costituisce il fattore di campionamento; nel nostro caso, poiché si era deciso di campionare al 20%, tale valore era *0.2*. Successivamente vengono quindi selezionati in maniera casuale un numero di clienti corrispondente al valore calcolato precedentemente. In tale scelta si escludono eventualmente clienti non desiderati, come per esempio quelli appartenenti ai negozi *Retail*.

3. I clienti selezionati vanno a far parte di una nuova tabella, denominata *clienti_subset*.

4. A partire da questa tabella viene ricostruita la tabella delle fatture, includendo in essa quelle appartenenti ai soli clienti selezionati. Eventualmente si può approfittare del fatto che dobbiamo ricostruire tale tabella, per migliorarne la qualità: mentre la creiamo di nuovo, si possono escludere istanze di fatture troppo vecchie o di anni incompleti[12].
   La tabella risultante la chiamiamo *fatture_subset*.

5. Dalla tabella delle fatture si devono ricostruire anche le tabelle che dipendono da essa: vengono create *fatture_dettagli_subset* e *fatture_pagamenti_subset*, in base alle sole fatture rimaste, incluse in *fatture_subset*.

L'intero sistema lavorerà quindi con il nuovo campione di dataset invece che con i dati originali, per tutta la durata della simulazione.
Al lancio del BusinessFramework, questo meccanismo prevede il ricampionamento esclusivamente se manca almeno una delle quattro tabelle menzionate precedentemente, in caso contrario si procede riutilizzando

[12]Tuttavia è prevista successivamente una fase di trasformazione dedicata a queste operazioni (vedere più avanti).

il campione vecchio. Questo criterio evita di ripetere il campionamento ogni volta, dal momento che è un'operazione molto onerosa, inoltre nella stessa simulazione i vari training e test set devono basarsi sullo stesso campione.

Con la seconda versione del database, in cui i dati sono stati ulteriormente revisionati e ridotti, campionare non è più strettamente necessario, tuttavia l'elaborazione rimane molto lenta, perciò è comunque consigliabile (ovviamente prendendo una percentuale di dati molto più alta rispetto al 20%); inoltre, durante la sua esecuzione, il sistema fa riferimento alle tabelle del campionamento, pertanto, nel caso non si voglia usufruire di tale tecnica, occorre comunque effettuare un campionamento al 100%[13], in modo che il BusinessFramework lavori su una copia completa dei dati sorgenti.

Anche se questo procedimento può sembrare inutile, in realtà porta diversi vantaggi:

- si possono effettuare preliminarmente in maniera automatica operazioni di selezione per migliorare o ridurre i dati.
- si possono apportare altre modifiche anche manuali ai dati senza preoccuparsi di cancellare o alterare i dati originali, perché si lavora su una copia.
- se necessario, i dati possono essere ripristinati facilmente.

4.2.3 Esecuzione automatica delle trasformazioni

I dati in nostro possesso hanno bisogno di un'iniziale revisione prima che su di essi venga effettuato qualsiasi tipo di operazione.
Dopo il campionamento che ha permesso di avere un numero di dati più ridotto, occorre migliorare la qualità di questi ultimi; innanzitutto è necessario porre riparo ai costi d'acquisto nulli presenti nella tabella dei dettagli delle fatture. Per fare ciò, come descritto nel capitolo 2, c'è bisogno di creare la tabella *prodotti_anno* che contenga la costante moltiplicativa calcolata per ogni prodotto e per ogni anno, poi in

[13]Si esegue impostando il fattore di campionamento a *1*.

base a questo occorre eseguire un update sulla tabella del campionamento dei dettagli delle fatture, dove il campo del costo d'acquisto è nullo. Inoltre, se non lo si è già fatto in fase di campionamento, possiamo anche eliminare le fatture relative agli anni 2002 e 2011, che non si useranno in fase di generazione dei modelli: questo permetterà di velocizzare qualsiasi accesso alla tabella, dato che vi saranno meno osservazioni da scansionare.
In ultimo, se si vogliono ridurre notevolmente i tempi di ricerca, occorre modificare le tabelle in modo da utilizzare solo la tabella delle fatture con valori precalcolati aggiunti ed evitare così ogni tipo di accesso a quella dei dettagli, dal momento che quest'ultima è molto grande.
Si è pensato dunque di inserire, dopo il campionamento, un meccanismo dedicato a qualsiasi genere di trasformazione, in modo che sul dataset appena creato possano essere eseguite in maniera automatica tutte le operazioni di ETL preliminari necessarie.

Riassumendo, per ora tale modulo esegue dunque in sequenza:

1. Creazione della tabella *dm.prodotti_anno*.

2. *UPDATE* della tabella *dm.fatture_dettagli_subset*, moltiplicando nelle osservazioni in cui manca il costo d'acquisto il valore della costante relativa ad un determinato prodotto (presente in *dm.prodotti_anno*) per il fatturato.

3. *DELETE* delle righe relative agli anni 2002 e 2011 dalle tabelle *dm.fatture_dettagli_subset* e *dm.fatture_subset*.

4. Creazione della tabella *dm.motivi_reso*.

5. Aggiunta a *dm.fatture_subset* delle colonne:

   fatturato_tot

   margine_tot

   fatturato_resi_tot

   sconto_tot

   insoluto_tot

insoluto_recuperato_tot

fatturato_inattesa_tot

fatturato_pagato_tot

6. *UPDATE* delle colonne appena aggiunte con la somma dei relativi valori presenti in *dm.fatture_dettagli_subset*, ma anche in *dm.fatture_pagamenti_subset*.

Questo modulo è comunque molto generale poiché in esso è possibile inserire qualsiasi tipo di altra operazione[14]; tutte quelle presenti verranno eseguite in sequenza.

4.2.4 Ottimizzazione dell'accesso alle tabelle

Il sistema deve interagire con il database interrogandolo costantemente, tuttavia quest'ultimo, essendo composto da una grossa quantità di dati, rende difficoltosi qualsiasi confronto e ricerca; i lunghi tempi di risposta dovuti all'eccessiva elaborazione risultano spesso inaccettabili.
Si è quindi pensato di introdurre nel sistema un meccanismo di ottimizzazione che consiste nel creare automaticamente degli indici sulle tabelle del campionamento in base ai campi di confronto[15] utilizzati nelle query SQL.
Vengono utilizzati indici di *Hash* per le corrispondenze esatte, mentre per i campi coinvolti in confronti su intervalli di valori vengono usati gli indici basati su *B-Tree*.
Poiché creare degli indici è un'operazione comunque onerosa ed a volte per certe interrogazioni[16] potrebbe addirittura portare ad un peggioramento delle prestazioni, tale meccanismo di ottimizzazione è comunque facilmente escludibile a livello di codice nel caso lo si ritenesse opportuno.

[14] Si tratta di editare una query SQL composta da più query.
[15] Ci si riferisce soprattutto a quelli presenti nelle clausole *WHERE* e *GROUP BY*.
[16] Principalmente per quelle di scrittura.

4.2.5 Nuove variabili

Rispetto al sistema di partenza è stato necessario creare e gestire le nuove variabili[17], che prima non esistevano:

- ritardo dell'ultimo acquisto
- ratio dell'ultimo acquisto
- valore dei resi sul fatturato
- numero dei resi sul numero di acquisti

Tutte queste variabili, come quelle già disponibili, prevedono il calcolo della variabilità nei trimestri[18] tramite regressione lineare.
L'ultima di queste, ossia quella relativa al rapporto tra numero di resi e numero di acquisti, è più articolata e mediamente impiega molto più tempo per essere elaborata perché ognuna delle 11 variabili che comprende deve essere calcolata indipendentemente dalle altre.

Inoltre, rispetto al caso su cui era stato testato il sistema di partenza, si sono dovute invece escludere:

- agente
- settore
- categorie fatturato
- categorie margine

poiché nel database non ci sono le informazioni necessarie per ricavarle e gestirle.

4.2.6 Struttura del file di configurazione

In base alle modifiche illustrate precedentemente, la struttura del file xml di configurazione del BusinessFramework è stata ampliata.

[17] In realtà si parla di gruppi di variabili.
[18] Si intende il coefficiente angolare.

Parametro del nuovo tipo *system* per l'impostazione del sistema:

case
: [*Valore numerico intero*]
Indica il numero identificativo del caso per il quale il sistema dovrà impostare il proprio funzionamento (attualmente i valori assumibili sono *1* o *2*).

Aggiunta ai parametri di tipo *variables*:

ritardo_ultimo_acquisto
: [*Valore booleano*]
Indica se bisogna considerare la variabile relativa al ritardo dell'ultimo acquisto.

ratio_ultimo_acquisto
: [*Valore booleano*]
Indica se bisogna considerare la variabile di ritardo dell'ultimo acquisto sulla somma di media e deviazione standard della distanza di acquisto.

resi_fatturato
: [*Valore booleano*]
Indica se bisogna considerare la variabile relativa al rapporto fatturato resi su fatturato.

resi_acquisti
: [*Valore booleano*]
Indica se bisogna considerare la variabile relativa al rapporto numero di resi su numero di acquisti.

La nuova struttura complessiva del file di configurazione finale, nel caso del BusinessFramework, si presenterà in questo modo[19]:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configuration>
  <system>
    <case>2</case>
  </system>
```

[19]Il contenuto dei vari tag qui riportato è solo esemplificativo.

```
<input>
  <acquisti>5,10,15</acquisti>
  <vita>1,2,3</vita>
  <abbandono>10,15,20</abbandono>
  <data>01/01/2006</data>
</input>
<database>
  <url>jdbc:postgresql:database</url>
  <user>user</user>
  <password>0,0,0,0,0,0,0,0,0,0,0,0</password>
</database>
<variables>
  <fatturato_margine>true</fatturato_margine>
  <sconto_fama>true</sconto_fama>
  <fama_numacq>true</fama_numacq>
  <insoluto_fatturato>true</insoluto_fatturato>
  <insoluto_r_fatturato>true</insoluto_r_fatturato>
  <ritardo_riacquisto>true</ritardo_riacquisto>
  <distanza_acquisti>true</distanza_acquisti>
  <frequenza>true</frequenza>
  <ratio>true</ratio>
  <settore/>
  <agente/>
  <categoria_fatturato/>
  <categoria_margine/>
  <ritardo_ultimo_acquisto>true</ritardo_ultimo_acquisto/>
  <ratio_ultimo_acquisto>true</ratio_ultimo_acquisto>
  <resi_fatturato>true</resi_fatturato>
  <resi_acquisti>true</resi_acquisti>
</variables>
<model>
  <minnumobj>10,30,60</minnumobj>
  <difference>0,1</difference>
  <mode>true</mode>
  <k>3,6,9,12,24</k>
  <h>0,3,6,9,12,24</h>
  <trindex/>
  <teindex/>
  <thr>0.0,0.3,0.5</thr>
```

```
    <giorni>false</giorni>
    <distance>false</distance>
    <multitest2>false</multitest2>
  </model>
  <security>
    <pbe>0,0,0,0,0,0,0,0</pbe>
  </security>
</configuration>
```

Per la DMU, invece, il file di configurazione è più compatto:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configuration>
  <input>
    <acquisti>10</acquisti>
    <vita>2</vita>
    <abbandono>15</abbandono>
    <data>01/01/2010</data>
  </input>
  <database>
    <url>jdbc:postgresql:database</url>
    <user>user</user>
    <password>0,0,0,0,0,0,0,0,0,0,0,0,0</password>
  </database>
  <model>
    <soloabbandoni>true</soloabbandoni>
    <unionlista>false</unionlista>
  </model>
  <security>
    <pbe>0,0,0,0,0,0,0,0</pbe>
  </security>
</configuration>
```

Si fa notare che nella DMU innanzitutto occorre specificare un unico valore dei parametri di input, perché questi ultimi devono corrispondere esattamente ai parametri usati nel particolare modello caricato (nel BusinessFramework, al contrario, possono essere inseriti più valori per permettere di generare un vasto numero di modelli per tutte le combinazioni possibili). Non compare inoltre il campo per lo switch del caso di studio: essendo un tool dedicato all'utenza finale e che

quindi può essere usato esclusivamente per un'unica azienda, si è pensato che questa informazione fosse meglio cablarla nel codice, in modo da non poterla modificare. La struttura della DMU rimane comune la medesima in ogni caso e mantiene tutti i moduli in maniera completa, viene modificato solamente un flag interno per l'impostazione del caso da utilizzare.
L'unico cambiamento rispetto al file di configurazione della suite di partenza è la sua sinteticità: sono riportati solamente i campi strettamente necessari, mentre sono tagliati fuori tutti quelli che servono esclusivamente al BusinessFramework.

4.3 Comportamento complessivo dei tool dopo l'estensione

Vediamo ora il comportamento complessivo di BusinessFramework e DMU, dopo aver subito le variazioni precedentemente illustrate.

BusinessFramework

Dalle modifiche descritte precedentemente possiamo dire che il BusinessFramework è notevolmente cambiato, o più correttamente si può dire che è stato ampliato.

Riassumendo tutto quello fin qua detto, la fase di ETL per ogni dataset da generare prevede:

1. inizio generazione del dataset solo se non presente
2. campionamento se richiesto e mancante
3. trasformazione se richiesta
4. ottimizzazione se richiesta
5. estrazione della clientela
6. etichettatura della clientela
7. generazione delle variabili

Il comportamento restante che riguarda la fase di Data Mining, ossia la generazione e la validazione del modello, rimangono invariati, così come la creazione dei file risultanti.

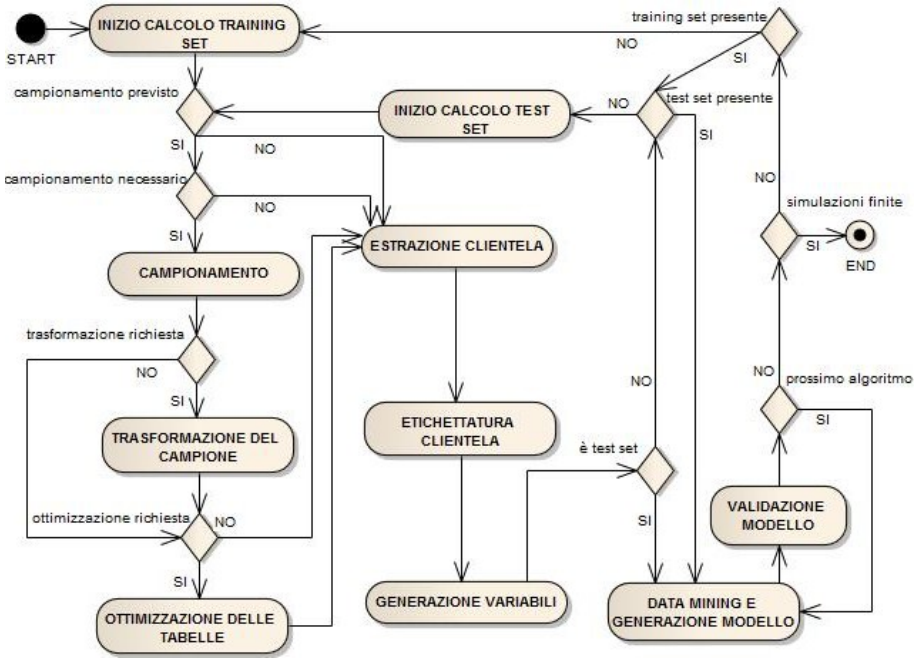

Figura 4.2: *Comportamento del BusinessFramework*

DMU

Per quanto riguarda invece la DMU, il comportamento per la singola esecuzione è rimasto lo stesso, tuttavia è possibile utilizzare un avvio in cui vengono lanciate più istanze sequenzialmente per testare la previsione per più date a distanze regolari di tempo.
I due processi contemporaneamente in esecuzione avranno un proprio flusso di controllo separato, tuttavia l'applicazione principale rimarrà in attesa che l'altra sia terminata prima di continuare e successivamente lanciare l'esecuzione di un'altra DMU.

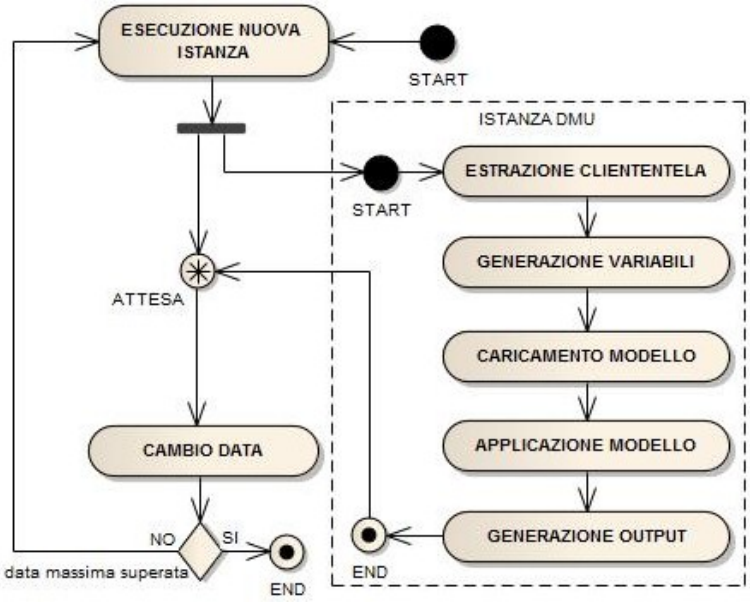

Figura 4.3: *Comportamento della DMU*

Capitolo 5

Simulazioni e test

In questo capitolo verranno illustrati i risultati ottenuti con la soluzione software descritta nei precedenti capitoli, implementata per automatizzare il processo di previsione degli abbandoni per il caso aziendale trattato. Il risultato consiste in uno o più modelli generati a partire da una combinazione ottimale di valori dei parametri coinvolti, ottenuti tramite simulazioni; i modelli definitivi vengono scelti sulla base di successivi test.

5.1 Processo di generazione e verifica dei modelli

Per ottenere un modello secondo l'analisi e le caratteristiche mostrate nel capitolo 2, i passi da eseguire sono i seguenti:

1. Generazione di molteplici modelli tramite massicce simulazioni in cui vengono provate più combinazioni di parametri di input

2. Analisi e selezione dei modelli migliori

3. Test di previsione dei modelli scelti per periodi temporali ravvicinati

4. Analisi dei risultati e selezione dei modelli che hanno generato il minor numero di errori

Alla fine di questo lungo procedimento otterremo uno o più modelli adatti a predire gli abbandoni per l'azienda in questione; nel caso questo non si verificasse, o se comunque si volessero avere risultati migliori, occorre ripeterlo cambiando i parametri usati per le simulazioni. Nelle sezioni successive verranno discusse le tappe principali che porteranno ad avere un buon modello di previsione per il caso aziendale che ci interessa.

5.2 Simulazioni

Per trovare i valori ottimali dei parametri necessari per generare i modelli relativi al nostro caso aziendale occorre eseguire numerose simulazioni.
Come primo approccio si deve partire da valori molto approssimativi o che in passato hanno già portato a buoni risultati per altri casi, per poi raffinarli e concentrarsi solo su quelli che promettono meglio.
Si parte quindi con usare il BusinessFramework configurandolo inizialmente con una serie di valori per ogni parametro disponibile e si lascia elaborare fino a quando non sono stati prodotti tutti i modelli per ciascuna combinazione.
Per riuscire a fare ciò è stato necessario utilizzare il sistema software in una macchina Ubuntu a 12 core[1] di cui 9 dedicati a tale computazione (gli altri 3 sono stati lasciati alle operazioni del sistema operativo) e con 16 GB di RAM.
Le simulazioni sono state pianificate in modo tale che le combinazioni totali da testare fossero divise fra 9 istanze di BusinessFramework (una per ogni core disponibile, quindi fino a saturazione della capacità di elaborazione), poi eseguite in parallelo.

5.2.1 Stima dei tempi di calcolo

Per eseguire così tante simulazioni (nell'ordine delle migliaia), il tempo diventa un aspetto molto critico. Anche con una buona potenza di calcolo e dividendo il lavoro tra diverse istanze di BusinessFramework, occorrono diversi giorni e questo dipende poi anche dal numero

[1] Si tratta di processori *AMD Opteron 6128*.

di valori impostato: più quest'ultimo sarà alto per ciascun parametro, più combinazioni dovranno essere calcolate.
È di vitale importanza stimare in anticipo i tempi di calcolo per evitare che l'elaborazione si protragga poi per un numero di giorni insostenibile (anche qualche mese) ed in caso di tempo eccessivo occorre effettuare delle scelte sulla quantità di valori di ogni parametro, cercando di ridurla il più possibile (si iniziano a togliere dai campi meno importanti).
Bisogna infatti tenere conto che il numero di combinazioni di coppie di training e test set da calcolare è dato da:

$$N_{comb} = N_{acquisti} \cdot N_{vita} \cdot N_{abbandono} \cdot N_{thr} \cdot N_h \cdot N_k$$

e poi per ognuna di esse la fase di Data Mining viene applicata un numero di volte pari a:

$$N_{c\_dm} = N_{minumobj} \cdot N_{difference}$$

per produrre i modelli.
Nel primo caso bisogna inoltre considerare che il tempo effettivo impiegato sarà in realtà il doppio, dato che ogni combinazione prevede il calcolo di 2 dataset:

$$t_{ETL} = t(N_{dataset}) = t(2N_{comb})$$

mentre per quanto riguarda la fase di Data Mining occorre moltiplicare il risultato per il numero di algoritmi utilizzati (quindi in generale 5) e considerare che ogni calcolo si ripete per ciascuna coppia di training e di test set:

$$t_{DM} = t(N_{comb} \cdot N_{dm}) = t(N_{comb} \cdot N_{c\_dm} \cdot N_{alg})$$

Si avrà quindi un tempo complessivo pari a:

$$t_{tot} = t_{ETL} + t_{DM} = N_{comb} \cdot [2t_{dataset} + (t_{dm} \cdot N_{c\_dm} \cdot N_{alg})]$$

dove $t_{dataset}$ è il tempo medio per calcolare un dataset (in genere 10 minuti) e t_{dm} quello per applicare ad ogni coppia un algoritmo di Data Mining (in genere 20 secondi). In realtà, nella costruzione di un dataset, la fase che in genere richiede maggior tempo è la generazione delle

variabili, pertanto il tempo complessivo richiesto per calcolare un singolo dataset dipende soprattutto da quante e quali variabili si è deciso di includere; un numero elevato di queste ultime potrebbe allungare in maniera considerevole il tempo di elaborazione, soprattutto poi se esse sono molto complesse.
È ovvio che la fase di ETL dedicata alla costruzione dei dataset è molto più dispendiosa in termini di calcolo rispetto a quella di Data Mining, perciò bisogna porgere particolare attenzione a non esagerare nella cardinalità totale di valori da elaborare, soprattutto per quelli riferiti alla generazione dei training e test set[2]; aggiungere anche un solo valore a volte potrebbe corrispondere a diverse ore, se non addirittura giorni, di calcolo in più. In genere la cardinalità di h e k deve tuttavia rimanere elevata per permette di coprire al meglio tutto il periodo temporale di interesse.

Avere ottimizzato la fase di creazione dei dataset, ha permesso di ridurre notevolmente il tempo globale di calcolo che sarebbe stato necessario nel caso si fossero dovuti generare tutti quanti i training ed i test set, però questo non ci permette di stimare con precisione il numero di dataset che realmente saranno prodotti, perché ovviamente occorre mettere in gioco anche fattori casuali causati da sovrapposizioni di date già elaborate da precedenti simulazioni.

Una misura che possiamo adottare deriva da un metodo empirico:

$$t_{tot} = t(acq_i, vita_j, abb_p, thr_q) \cdot N_{acquisti} \cdot N_{vita} \cdot N_{abbandono} \cdot N_{thr}$$

Si basa sull'osservazione del tempo di calcolo impiegato per una maxi combinazione (si intende presa considerando solo i parametri principali, inglobando invece in essa tutti le differenti combinazioni prodotte da h e k) e moltiplicarlo per il loro numero totale. Quindi ovviamente il tempo di riferimento non sarà uguale per tutte le elaborazioni, ma cambierà di volta in volta in base alla cardinalità di h e k.

Altrimenti ci si può affidare ad un upper bound, ossia al calcolo del numero dei dataset prodotti nel caso peggiore.

[2]Ci si riferisce a quelli coinvolti nel calcolo di N_{comb}.

Se consideriamo come modalità di elaborazione la valutazione della stazionarietà avremo un numero totale di dataset di:

$$N_{ds\_ott} = N_{acquisti} \cdot N_{vita} \cdot N_{abbandono} \cdot N_{thr} \cdot N_h(1 + N_k)$$

Per un tempo di calcolo complessivo quindi di:

$$t_{tot} = t_{ETL} + t_{DM} = (t_{dataset} \cdot N_{ds\_ott}) + (t_{dm} \cdot N_{comb} \cdot N_{dm})$$

Questo ovviamente nel caso peggiore, ma nella pratica capita spesso che, oltre alla costruzione del training set, venga saltata anche la generazione di molti test set (dipende poi dai valori e combinazioni degli elementi di h e k), riducendo ulteriormente il tempo totale.

Avendo poi la possibilità di suddividere il carico di lavoro fra più istanze simultanee di BusinessFramework con una macchina a più core, avremo:

$$t_{multi\_tot} = \frac{t_{tot}}{N_{BF}} \qquad N_{BF} = N_{core}$$

Tuttavia, anche se la capacità di elaborazione è ben distribuita tra più processori, in realtà i tempi si dilungano per ogni istanza di BusinessFramework in esecuzione; questo presumibilmente è causato da un collo di bottiglia che si crea nell'accesso al database, pertanto si può supporre che il DBMS[3] non sia in grado di gestire in maniera efficiente la lettura e la scrittura concorrente sulla stessa risorsa.
Per evitare questo occorrerebbe replicare il database su più istanze di server del DBMS, o comunque fare accedere ogni BusinessFramework da porte diverse[4].
Nel caso di accesso unico per tutte le istanze di BusinessFramework, un compromesso tra velocità e quantità è dato da 6 esecuzioni contemporanee; con questa soluzione i tempi di elaborazione sono al limite dell'accettabile (in media circa 15-20 minuti per generare un dataset), se si considera che solo per la prima batteria di simulazioni ogni istanza deve in media calcolare circa 350 dataset.

[3]Ci si riferisce a PostgreSQL.
[4]La porta di default di PostgreSQL è la *5432*.

5.2.2 Valori di prova

Uno dei valori fondamentali che in genere differisce maggiormente da caso a caso, è il numero minimo di acquisti per estrarre i clienti, pertanto bisogna stimarne un iniziale valore: si prendono la media e la deviazione standard degli acquisti per cliente per ogni anno e successivamente se ne calcola ancora media e deviazione standard però questa volta rispetto all'intero periodo temporale.
L'intervallo approssimativo in cui il valore deve rientrare è dato da:

$$[avg - stddev;\ avg + stddev]$$

Applicando tutto questo al nostro caso, la media e la deviazione standard risultano:

$$avg = 12,81 \qquad stddev = 4,28$$

ottenendo quindi l'intervallo:

$$[8,53;\ 17,09]$$

che sarà perciò riferito al numero medio di acquisti effettuati nell'arco di un anno.
Dopo aver eseguito anche un confronto con risultati precedenti ed aver rapportato il tutto a casi trattati in passato su cui è già stato testato il sistema, inizialmente si è pensato di fissare il valore a *18* acquisti nell'arco dei *3* anni, poi considerando anche di fare elaborare per i *2* ed i *4* anni e tenendo conto che la deviazione standard è molto alta, si sono presi anche valori a distanza di un intervallo di 5 acquisti dal numero fissato, perciò *13* e *23*.
Inoltre, da esperienze passate, si è notato che per la definizione di abbandono, *18* è un valore che spesso dà buoni risultati, tuttavia, vedendo che la distanza di acquisto media dei clienti è di soli 2 mesi (vengono eseguiti acquisti periodicamente), si è pensato di testare il sistema anche per valori più bassi: sono stati aggiunti quindi anche *13* e *15*.
La soglia di etichettatura è stata fissata inizialmente come di consueto a *0* e si è provato a mettere anche un valore leggermente superiore, ossia *0,3*.

5.2.3 Analisi ed interpretazione dei risultati

L'operazione che si attua dopo l'esecuzione di ogni batteria di simulazioni è definita *Data Mining su Data Mining*, ossia una volta che si hanno a disposizione come risultato i vari modelli generati, sono questi ultimi ad essere sottoposti a loro volta ad analisi ed a confronto.
I modelli prodotti dalle simulazioni vengono raggruppati a seconda dei parametri utilizzati e delle date di training: si uniscono così le performance di quelli generati dagli stessi training set, mostrando le differenze tra i risultati dei vari test set associati.
Si guardano poi in linea di massima principalmente le seguenti caratteristiche:

- *numero di abbandoni (soglia minima).*
  Empiricamente è provato che un modello che predice pochi abbandoni non è veritiero, pertanto occorre stabilire un limite minimo a partire dal quale selezionare i modelli migliori.

- *media di precisione e f-measure su abbandoni (soglia minima).*
  La precisione e la f-measure sugli abbandoni sono le misure di valutazione che ci interessano maggiormente e sono indici di un buon modello per il caso che stiamo trattando, dato sono questi che alla fine dovremo predire. Per prendere in considerazione i modelli migliori bisogna provare ad aumentare il più possibile la soglia di selezione di queste misure.

- *deviazione standard di precisione e f-measure su abbandoni (soglia massima).*
  La variazione delle prestazioni precedenti ottenute sui diversi test set presi nel tempo, non deve essere elevata se si vuole un modello che offra risultati stabili e non sia adatto solo per determinati test set. Per ottenere modelli apparentemente più adatti, occorre cercare di abbassare il più possibile questa soglia.

- *periodo tra il primo e l'ultimo test set con buoni risultati (valore minimo).*
  Una caratteristica fondamentale che si cerca in un buon modello è la capacità di adattarsi alla maggior parte dei casi, ossia non

deve essere specifico solo per determinati training set; più il periodo in cui si ottengono buone prestazioni su test set differenti è lungo e più il risultato sarà stazionario.

Infine si analizzano individualmente i modelli che presentano gli aspetti migliori secondo i criteri sopra citati: osservando l'albero decisionale di ognuno di essi si cerca di capire quali sono le caratteristiche (tra cui le variabili utilizzate) che accomunano modelli che utilizzano parametri anche differenti, e quali sono i valori, invece, che sono ininfluenti al fine di una buona predizione.
Inoltre l'albero di un buon modello in genere presenta:

- un minimo di profondità
- uso di più variabili
- prestazioni bilanciate tra i vari rami

Le prime due caratteristiche si riferiscono alla complessità dell'albero: in genere il modello non è molto significativo se ha per esempio un solo split determinato da un'unica soglia o se si basa sul valore di una sola variabile assoluta; si vuole evitare che l'albero sia troppo semplice e quindi produca risultati troppo approssimativi o scontati.
La terza caratteristica evita di avere delle anomalie nel predire fedeli o abbandoni; ci interessano solo i secondi, ma nella realtà gravano più gli errori sui primi, pertanto non vogliamo che comunque abbia pessime prestazioni nel riconoscerli.

Mettendo insieme tutte queste informazioni si cerca poi di aggiustare adeguatamente i parametri di ogni simulazione per predisporre la successiva, fino ad ottenere risultati che siano i più stabili ed affidabili possibili.

5.3 Test

La bontà dei modelli più promettenti precedentemente selezionati, può essere poi verificata tramite DMU nella modalità di test descritta nei

precedenti capitoli.
Per poter carpire il possibile comportamento del modello, i test vengono eseguiti a date di previsione bimestrali, quindi a distanza molto ravvicinata; tutto questo lo si effettua a partire dal 2005 per tutti gli anni successivi.
I test produrranno la lista degli abbandoni, dato che il nostro risultato finale dovrà essere la previsione di quest'ultimi e non quella sui fedeli.

5.3.1 Tempi di elaborazione

I test sono mirati, cioè vengono eseguiti solo su un gruppo molto ristretto di modelli, tuttavia se si considera che ognuno di questi ultimi dovrà essere verificato un numero elevato di volte, occorre effettuare una stima dei tempi.
Il numero di test da eseguire sarà dato ovviamente da:

$$N_{test} = \left\lfloor \frac{data_{fine} - data_{inizio}}{frequenza_{test} \cdot 30} \right\rfloor$$

dove $frequenza_{test}$ nel caso di test bimestrali corrisponde a *2*.
Il tempo totale sarà quindi:

$$t_{tot} = t_{test} \cdot N_{test}$$

dove in genere il tempo per eseguire un test è di circa 20 minuti.
Da questa formula si può intuire che alla fine per eseguire tutti i test su un singolo modello occorreranno parecchie ore, tuttavia i tempi di elaborazione non raggiungeranno mai quelli richiesti dalle iniziali simulazioni.
Ogni DMU è in grado di eseguire i test per un solo modello alla volta, tuttavia è possibile eseguire più DMU in parallelo, verificando così fino a 6-7 modelli contemporaneamente[5].

[5]Vale la stessa osservazione già effettuata per le simulazioni: per ogni istanza aggiunta i tempi di calcolo totali si dilungano per una probabile inefficienza di gestione della concorrenza del DBMS.

5.3.2 Prestazioni dei test di previsione

Osservando gli errori prodotti dal modello per intervalli regolari dello storico, si può vedere quanto esso sia affidabile; il classificatore commette un errore quando prevede un abbandono anche se il cliente in questione ha invece successivamente acquistato nel periodo seguente[6] alla data di previsione; un modello accettabile non dovrebbe superare il 30% di errori sul totale degli abbandoni; nel caso la soglia di etichettatura fosse diversa da *0*, si deve tenere conto anche del fatturato nel periodo di vita ed in quello successivo alla data di previsione, secondo la formula vista nei capitoli precedenti. Questo metodo può aiutare a selezionare meglio i modelli più promettenti e completa quindi il lavoro compiuto dal BusinessFramework.

5.4 Risultati finali

Ora vedremo nel concreto i risultati ottenuti nelle varie simulazioni e nei test. Si menzioneranno innanzitutto i modelli migliori prodotti dalle simulazioni e poi vedremo, in base alle performance dei test, quali di essi sono stati scelti in maniera definitiva.

Simulazioni

É stato necessario lanciare diverse batterie di simulazioni, ognuna delle quali presentava accorgimenti, variabili e parametri leggermente differenti dalle precedenti.
L'ultima, che si è presa come definitiva poiché mostrava buoni modelli, ha generato circa 56.000 classificatori.
Su di essi è stata attuata una ristretta selezione[7] in base alle prestazioni sugli abbandoni e sulla stazionarietà dei risultati; se si raggruppano poi in insiemi con stesso albero decisionale e quindi anche con identiche caratteristiche, i migliori si possono ricondurre a soli 21 gruppi di

[6] Il periodo considerato è dato dal numero di mesi della definizione di abbandono.

[7] Le soglie di selezione vengono abbassate o alzate a seconda dei casi, in modo che ne rimanga solo un esiguo numero.

modelli (con le medesime date di training, ma quelle di test a distanze diverse nel tempo).

Questi modelli presentano:

- *precisione su abbandoni* $\geqslant 75\%$
- *periodo considerato* $\geqslant 2$ *anni*
- *numero di abbandoni previsto* $\geqslant 100$

Per ogni insieme si prende in considerazione sempre un modello rappresentativo (nella pratica sono uguali) e se ne guarda l'albero; se consideriamo i criteri per determinare un buon decision tree descritti in precedenza, possiamo isolare solamente 6 modelli e saranno quelli a cui ci accingeremo ad eseguire i primi dettagliati test.
I modelli in questione sono quelli generati con i parametri mostrati in tabella.

	acquisti	vita	abbandono	thr	minnumobj	diff	giorni	alg	training
M1	13	3	18	0	30	0	false	1	30-9-2007
M2	13	3	18	0	60	0	false	0	30-9-2007
M3	13	3	18	0,3	60	0	false	0	30-9-2007
M4	18	3	18	0	30	0	false	1	30-9-2007
M5	13	2	18	0,3	10	0	false	1	28-2-2007
M6	13	3	13	0,3	10	0	false	1	30-4-2008

Bisogna tuttavia considerare che rimane un'incognita per modelli generati con algoritmi di aggregazione[8], infatti per essi o non è possibile visualizzare l'albero o è troppo vasto e complicato per comprenderlo, dato che ne combinano più insieme. Quindi poi cercheremo comunque di verificare le prestazioni anche dei 15 restanti modelli selezionati in partenza, ma si daranno priorità e importanza maggiori ai risultati di quei 6 stabiliti precedentemente.

Test

Se eseguiamo un test ogni due mesi, partendo da fine febbraio del 2005[9] fino a dicembre 2010, avremo un totale di 36 previsioni per ogni

[8] RandomForest e AdaBoost.M1

[9] I modelli hanno massimo vita a 3, perciò l'estrazione dei clienti partirà dal 2002.

modello.
In questi test, nel complesso i 6 modelli hanno mostrato le seguenti performance:

	Abbandoni		**Perc. errore**		**Perc. max errore**	
	media	*stddev*	*media*	*stddev*	*media*	*stddev*
Modello 1	264,388	47,8571	0,19	0,065	0,207	0,073
Modello 2	291,277	62,508	0,207	0,063	0,226	0,068
Modello 3	291,555	62,428	0,132	0,058	-	-
Modello 4	234,25	57,701	0,224	0,085	0,243	0,089
Modello 5	190,916	63,055	0,202	0,12	-	-
Modello 6	259,611	87,813	0,369	0,169	0,395	0,167

Nella tabella è stato riportato sia l'errore calcolato nei successivi mesi di abbandono dalla data di previsione e sia quello commesso prendendo in considerazione anche tutto il restante arco temporale; logicamente il secondo sarà sempre maggiore o uguale al primo.
Nei modelli con soglia di etichettatura diversa da *0*, per semplicità di calcolo, l'errore massimo non è riportato. Comunque quest'ultimo non è fondamentale ed anche successivamente prenderemo in considerazione solo il primo.

Osservando i risultati dei test, possiamo già scartare l'ultimo modello, che supera abbondantemente la soglia del 30% di errore; il classificatore che risulta essere in assoluto il migliore è il terzo, il quale si è sempre mantenuto sotto al limite di errore in ciascuno dei singoli test. I restanti hanno valori assai simili ed il loro errore inizia ad essere abbastanza consistente, poiché si aggira sul 20%; tuttavia se approfondiamo i risultati dei singoli test (che qui non abbiamo riportato) notiamo che il quarto ed il quinto superano diverse volte la soglia del 30%, infatti si può notare come essi abbiano una deviazione standard leggermente superiore rispetto agli altri due, questo è indice che il valore dell'errore per vari test può discostarsi abbondantemente rispetto a quello dichiarato dalla media.
Si può inoltre vedere come la maggior parte di quelli che risultano essere i peggiori abbia un *minnumobj* pari a *10* e quindi molto basso, questo può aver sviato l'interpretazione dei risultati inizialmente effettuata. Infatti quando tale valore non è molto elevato il modello

rischia di adattarsi troppo allo specifico training set da cui è indotto e di conseguenza, nel caso si volessero effettuare previsioni in date diverse da quelle in cui si sono presi i test set, i risultati potrebbero deludere completamente le aspettative.
In conclusione si può dire che i modelli che possiamo giudicare come i migliori sono i primi tre, ponendo particolare attenzione al terzo.

Dopo aver eseguito i test che completano l'analisi anche per i 15 restanti modelli individuati in partenza, sono emersi altri classificatori molto buoni, che vengono presentati in tabella.

	acquisti	vita	abbandono	thr	minnumobj	diff	giorni	alg	training
M7	13	3	18	0	30	0	false	2	30-1-2008
M8	13	3	18	0,3	30	0	false	2	30-1-2008
M9	18	3	18	0,3	30	0	false	2	30-1-2008
M10	18	3	18	0,3	30	0	false	1	30-9-2007
M11	13	3	15	0	30	0	false	4	31-5-2007
M12	13	3	15	0	30	0	false	2	30-4-2008
M13	13	3	15	0,3	30	0	false	2	30-4-2008
M14	13	3	13	0,3	30	0	false	4	30-9-2007

Le performance riscontrare sono riportate successivamente.

	Abbandoni		**Perc. errore**		**N criticità**
	media	*stddev*	*media*	*stddev*	
Modello 7	282,388	56,168	0,205	0,062	2
Modello 8	311,638	63,906	0,164	0,065	0
Modello 9	236,5	53,469	0,172	0,055	2
Modello 10	234,25	57,701	0,127	0,082	3
Modello 11	211,055	43,719	0,187	0,059	2
Modello 12	185,416	42,025	0,199	0,061	3
Modello 13	215,694	46,665	0,135	0,05	0
Modello 14	169,305	37,279	0,109	0,039	0

Per numero criticità si intende quante volte il tasso di errore ha superato il 30% sui 36 test totali eseguiti per ogni modello.
Si può notare che ben 3 classificatori (l'*8*, il *13* ed il *14*) hanno raggiunto risultati ottimi.
Alcuni modelli, come il *10*, hanno una percentuale di errore molto bassa, tuttavia presentano una deviazione standard più alta della norma, infatti in rari ed isolati test singoli i risultati si discostano dagli altri avendo un errore invece molto alto.

Si è comunque osservato un andamento generale tra i 21 modelli analizzati del verificarsi delle criticità per date che cadono ad agosto per tutti gli anni presi in esame; questo fa presupporre di avere o dei dati spuri per quanto riguarda quel periodo, o molto più probabilmente un andamento anomalo della produzione o delle vendite (è anche immaginabile, dato che è in piena estate). Quindi si può dire che un errore leggermente più alto in questo periodo non è da considerarsi molto grave.
Inoltre prendendo in considerazione ogni singolo test, si è notato un andamento generale della distribuzione dell'errore concentrarsi prevalentemente nei primi mesi di previsione: approssimando i risultati di tutti i test riferiti ad ogni modello otteniamo una curva monotona decrescente.

5.4.1 Modelli definitivi

In questa sezione riassumeremo caratteristiche e prestazioni dei modelli risultati migliori dalle precedenti analisi; inoltre si forniranno altri dettagli non ancora menzionati. I modelli più adatti che alla fine sono stati scelti e che andremo a descrivere, sono 5.

Caratteristiche principali dei modelli

Il modello migliore sembra essere l'ultimo riportato nella precedente sezione.
Esso ha parametri:

- *numero minimo di acquisti*: 13
- *vita*: 3
- *definizione di abbandono*: 13

Andando a vedere l'albero e dunque le relazioni trovate dal modello, le caratteristiche intrinseche di quest'ultimo sono:

modello 14
soglia di etichettatura: *0,3*
algoritmo: *AdaBoost con DecisionStump*
variabili: non pervenute

Bisogna ricordare che per gli aggregatori di classificatori, non possiamo visualizzare (o comunque comprendere adeguatamente) l'albero, poiché è multiplo e quindi non si possono sapere a quali variabili fa ricorso.

Tuttavia la maggior parte dei modelli migliori alla fine sono quelli prodotti dalla serie di parametri:

- *numero minimo di acquisti*: 13
- *vita*: 3
- *definizione di abbandono*: 18

Le caratteristiche di questi modelli sono:

modello 3
soglia di etichettatura: *0,3*
algoritmo: *J48*
variabili: $ritardo^{riacq}$ in giorni, $distanza^{acq}$ massima

modello 8
soglia di etichettatura: *0,3*
algoritmo: *RandomForest*
variabili: non pervenute

Anche in questo caso abbiamo un aggregatore di cui non possiamo visualizzare l'albero.

Poi vi è un buon modello con parametri:

- *numero minimo di acquisti*: 13
- *vita*: 3
- *definizione di abbandono*: 15

Il modello presenta le seguenti caratteristiche:

modello 13
soglia di etichettatura: *0,3*
algoritmo: *RandomForest*
variabili: non pervenute

Come i precedenti, questo modello ha avuto un errore molto basso e nessuna criticità.

Come ultimo possiamo ricordare il *10* con parametri:

- *numero minimo di acquisti*: 18
- *vita*: 3
- *definizione di abbandono*: 18

modello 10
soglia di etichettatura: *0,3*
algoritmo: *SimpleCart*
variabili: $ratio^{rit\_riacq}$, $ritardo^{riacq}$ in giorni, media di $resi_f$, $ratio^{ult\_acq}$

Questo modello lo menzioniamo perché, come già discusso precedentemente, ha ottenuto un tasso di errore molto basso, tuttavia per alcuni periodi di test presenta risultati abbastanza variabili.

Come si può notare, i valori dei parametri usati si discostano leggermente tra un modello e l'altro; ciò che li accomuna è il parametro di *vita* a *3* e *thr* a *0,3* (quindi non nullo). Il parametro *acquisti* più ricorrente è di *13*, mentre il valore più frequente di *abbandono* è *18*. Gli algoritmi che in genere offrono prestazioni migliori risultano gli aggregatori, in particolare il *RandomForest*.

Prestazioni

Per mettere a confronto i 5 modelli sopraelencati possiamo innanzitutto riportare nel dettaglio le prestazioni derivanti dall'applicazione dei classificatori sui 7 test set usati subito dopo la loro generazione, ossia nel corso delle simulazioni.

	A rilevati		**precision(A)**			**recall(A)**			**f-measure(A)**		
	media	*min*	*media*	*min*	*max*	*media*	*min*	*max*	*media*	*min*	*max*
M14	149	104	0,89	0,82	0,94	0,40	0,35	0,43	0,546	0,498	0,574
M3	223	142	0,87	0,80	0,95	0,52	0,41	0,60	0,65	0,564	0,733
M8	242	177	0,85	0,78	0,89	0,57	0,51	0,68	0,676	0,632	0,772
M13	188	117	0,88	0,85	0,93	0,45	0,34	0,56	0,591	0,493	0,676
M10	180	103	0,89	0,83	0,97	0,53	0,39	0,62	0,661	0,541	0,759

Le performance riscontrate invece nel corso dei test di previsione e che hanno poi portato alla scelta definitiva dei modelli migliori, sono leggermente differenti: possiamo confrontarle riassumendole nella seguente tabella.

	A rilevati		**Perc. errore**		**Confidenza**		**N criticità**
	media	*stddev*	*media*	*stddev*	*min*	*max*	
M14	169,305	37,279	0,109	0,039	0,737	0,98	0
M3	291,555	62,428	0,132	0,058	0,702	0,984	0
M8	311,638	63,906	0,164	0,065	0,662	0,983	0
M13	215,694	46,665	0,135	0,05	0,669	0,959	0
M10	234,25	57,701	0,127	0,082	0,609	0,994	3

Per valutare in definitiva le performance dei modelli scelti, come si può notare dalla tabella, si è fatto ricorso anche all'intervallo di confidenza calcolato per tutti i 36 test eseguiti; per stabilire quello globale, si prendono come estremi il valore più basso e quello più alto ottenuti tra tutti i test; un buon modello dovrà avere dei valori alti e compresi in un intervallo il più piccolo possibile.
La probabilità di confidenza impostata per calcolare il rispettivo intervallo ed ottenere i risultati sopra riportati è del 95%[10].

Previsione di abbandono

Volendo ora simulare una reale previsione utilizzando questi modelli, prenderemo la più recente data che ci è permessa: infatti bisogna considerare che possiamo arrivare fino alla fine del 2010 per essere sicuri di lavorare con dati affidabili ed occorre inoltre riservare un numero di mesi pari alla definizione di abbandono per poi potere eseguire il controllo dell'errore.
I risultati sono mostrati in tabella.

	Data previsione	**Def. abbandono**	**Abbandoni**	**Errori**	**Perc. errore**
M14	30-11-2009	13	267	45	0,168
M3	30-6-2009	18	346	54	0,156
M8	30-6-2009	18	310	29	0,093
M13	30-9-2009	15	284	44	0,154
M10	30-6-2009	18	261	36	0,137

Si può notare come i modelli si comportino bene, rimanendo molto al di sotto del 20% di falsi abbandoni previsti. Il migliore in questa

[10]Quindi $Z_{\alpha/2} = 1,96$ (vedere capitolo 1).

circostanza sembra il terzo, mentre il primo, su cui si puntava maggiormente, è quello che ha ottenuto la percentuale di errore più alta; tuttavia il periodo di campionamento e la data di previsione possono avere influito significativamente su questo aspetto.

Capitolo 6

Conclusioni

Il lavoro svolto in questa tesi ha consolidato i metodi già sperimentati in precedenza e ha verificato la genericità delle soluzioni adottate anche trattando casi specifici differenti. La principale difficoltà è risieduta nel cercare di capire i metodi migliori per potere attuare tutto ciò, continuando a gestire in maniera trasparente più casi diversi anche se riguardanti il medesimo problema di Data Mining.
Nonostante il problema trattato fosse lo stesso ci si è accorti che spesso particolari che per un caso possono essere scontati o trascurabili, in un altro possono risultare molto rilevanti e rendere il lavoro assai complesso; in questa tesi uno degli aspetti più critici, infatti, è stato quello di dover trattare una grossa quantità di dati, in tempi relativamente brevi ma continuando ad ottenere buoni risultati; tutto ciò infatti ha costretto a pianificare ed introdurre meccanismi per automatizzare ed ottimizzare al massimo tutto il procedimento ed ottenere così risultati comunque soddisfacenti.
Gran parte del lavoro è consistito nello sviluppo del sistema software che ora risulta più completo e stabile, inoltre presenta prestazioni notevolmente migliori, soprattutto per quanto riguarda i tempi di elaborazione, tuttavia si è abbastanza lontani dall'avere un'applicazione che riesca a risolvere la maggior parte dei problemi più frequenti in ambito aziendale; questo è dovuto alla vastità della materia, ma ogni caso aziendale che si tratterà in futuro darà lo spunto per integrare e rendere sempre più completo il sistema, però per quanto riguarda il churn della clientela, l'intero processo che segue il CRISP-DM, ha

raggiunto un'eccellente automazione.
I modelli ottenuti in seguito alle simulazioni sono molto buoni, tuttavia non è stato possibile sperimentarne altri perché questa operazione richiederebbe un tempo eccessivo.

Dopo aver fatto il punto della situazione ed il resoconto del lavoro fin qui svolto, successivamente verranno proposti alcuni spunti per completare e migliorare il tutto, sia a livello di procedimento e studio, sia sotto il profilo dello sviluppo del software.

6.1 Possibili revisioni ed aggiunte al processo

Occorrerà in futuro sperimentare il processo di Data Mining applicato finora, sia a nuovi casi, sia a problemi ed obiettivi differenti rispetto a quelli trattati in questa tesi, in modo che si riesca a sviluppare un procedimento risolutivo sempre più a carattere generale e che faccia fronte a problemi e realtà eterogenei.

Miglioramento dei risultati

Le simulazioni richiedono molto tempo, pertanto in questa tesi si è cercato un compromesso tra la qualità e la rapidità di avere risultati utili e sfruttabili.
I risultati ottenuti sono comunque molto buoni, ma si può cercare di perfezionarli, questo significa eseguire molti più test e, incrociando i risultati, trovare altre combinazioni di parametri che offrano risultati ancora più idonei.

Calcolo degli insoluti

In questa tesi si è affrontato il problema della previsione degli abbandoni, consolidandone l'automazione del suo processo risolutivo, delle tecniche e degli strumenti operativi già utilizzati in passato per un caso simile.
Un nuovo problema molto interessante che vale la pena di affrontare

è il calcolo degli insoluti, ossia prevedere quali clienti genereranno insoluti e per quale fascia di importo (cioè chi non pagherà, o lo farà in ritardo).
Questo genere di problema a prima analisi appare come un task di predizione numerica poiché occorre innanzitutto stimare l'importo (ossia il valore numerico) dell'insoluto a cui poi deve essere successivamente applicata una semplice classificazione necessaria a stabilire la fascia di appartenenza; ogni fascia, infatti, può essere vista come una classe.
Aggiungere un problema di tipologia differente richiede di rivoluzionare notevolmente il sistema, poiché occorre rendere ancora più generico il funzionamento del controller, separando anche quest'ultimo in ulteriori moduli come è già avvenuto per la parte relativa ai dati da utilizzare.
Inoltre occorrerebbe ovviamente approfondire lo studio su questo determinato problema per potere poi scegliere gli algoritmi migliori con cui sostituire quelli attualmente usati, che per ora sono adatti solo per la classificazione.
Tutto questo richiederebbe quindi ulteriori mesi di lavoro.

Nuovi casi di studio

La suite dovrà essere testata con altri casi aziendali in modo da consolidare il procedimento automatico di risoluzione tuttora adottato per i casi già visti. Accettare nuovi casi di studio darà l'opportunità di espandere il lavoro ed il software già a disposizione, offrendo nuove idee per rendere così la suite sempre più completa, grazie alla graduale integrazione di maggiori funzionalità.

6.2 Sviluppi futuri

Dal punto di vista software, lo sviluppo della suite è stato massiccio ed intensivo, ma come ci si può aspettare da un sistema del genere, che ha intrinseca tutta la complessità e la vastità di una disciplina come il Data Mining, esso può essere sviluppato sotto molti fronti. In questa sezione elencheremo alcuni possibili interventi che in futuro possono essere effettuati sul sistema.

Componenti aggiuntivi

Il sistema è ancora in pieno sviluppo: molte parti sono state estese, ma molte altre possono essere aggiunte, considerando anche il fatto che per ora riesce a risolvere solo uno specifico (anche se molto ricorrente) problema di Data Mining.

Introduzione di nuovi moduli per i task di Data Mining mancanti

Attualmente la suite è capace di affrontare bene problemi di classificazione, ma non è predisposta per automatizzare e risolvere tipi diversi di problema (clustering, predizione numerica e regole associative). Andando avanti ad accettare casi di studio differenti, sicuramente si dovranno aggiungere anche i moduli predisposti per tali task; come già detto precedentemente, occorrerà quindi differenziare il controller, a seconda del modulo che dovrà essere caricato.

Automazione dell'output relativo all'analisi dei test di previsione

Tuttora bisogna trascrivere manualmente in file *xls*[1] o *csv* i risultati dell'analisi dei test di previsione prendendoli dalle tabelle relazionali presenti nel database; questi file serviranno poi per essere confrontati e per permettere poi di produrre i grafici che occorrono in fase di analisi. Questa lunga e tediosa operazione potrebbe essere automatizzata se si realizzasse un unico tool capace di mettere insieme da solo i risultati sull'errore di previsione e sulla sua distribuzione, incaricandosi esso stesso di produrli poi sotto forma di file.
Inoltre bisogna precisare che le tabelle di output sono già generate utilizzando piccole parziali applicazioni (altrimenti manualmente ci sarebbe voluto ancora molto più tempo) che si incaricano di iterare le query necessarie alla verifica dei risultati; ora si tratta di creare un unico strumento, organizzato in maniera efficiente e completa rispetto a ciò che si ha attualmente e che sia infine quindi capace di esportare i risultati anche al di fuori del database.

[1] Formato MS-Office Excel.

Miglioramento del software

Nonostante i numerosi interventi apportati al software durante questa tesi, è possibile ancora sviluppare e migliorare diversi aspetti tra cui:

- DMU in C#
- Revisione del codice

DMU in C# La DMU è stata reimplementata in un altro linguaggio anche per questioni di maggiore portabilità in ambienti Microsoft, tuttavia uno degli scopi era anche ottenere maggiori prestazioni in tali sistemi; dai primi test di confronto con la versione java, è però apparso che essa elabora in tempi più lunghi. Anche se, come già detto, una delle idee di fondo era quella di avere una versione con prestazioni migliori, l'obiettivo in questa tesi era solamente creare una versione di base della DMU, poiché tale operazione richiede da sola l'impiego di molto tempo e di un sostanziale lavoro. È lasciato poi per i futuri sviluppi cercare di migliorare sempre di più tale versione, sperimentando anche combinazioni con tecnologie o driver diversi (per esempio ODBC), fino a ottenere i risultati attesi.

Revisione del codice È stata eseguita una consistente revisione di tutto il codice, tuttavia si può ancora migliorare la parte relativa alle query di interazione con il database; è infatti stato riscontrato che è possibile ridurle di numero (alcune possono essere inglobate velocemente in altre) e creare meccanismi per generalizzarle meglio (molte risultano uguali tra loro nella struttura, a meno solo di alcune costanti). Sarebbe poi consigliabile migliorare il codice SQL; questo intervento è utile anche per aumentare le prestazioni, poiché il modo di formulare una query influisce notevolmente sui tempi della sua esecuzione.
Inoltre si potrebbe fare in modo di utilizzare nomi di schemi e di tabelle diversi da quelli predefiniti: occorrerebbe introdurre nel sistema alcune costanti adatte a questo scopo, come lo si è già fatto anche per altri aspetti.
Tutto questo completerebbe il lavoro di revisione del codice già eseguito fino a questo punto.

Appendice A

Database operazionali e Data Warehouse

A.1 Tipologie di Database

I sistemi informatici su cui si appoggiano gli strumenti di Data Mining sono costituiti dai database.
Esistono principalmente due tipi di database: *database operazionali* e *Data Warehouse*[1].

A.2 Database operazionali

Ad essi ci si riferisce anche con il termine di sistemi transazionali; sono i database più semplici e sono ideati principalmente per contenere informazioni correnti che vengono aggiornate continuamente. Essi non sono quindi indicati per analisi complesse, ma sono usati principalmente in ambito amministrativo o in altri campi in cui interessa solo la versione più aggiornata dei dati.
Le query che possono essere eseguite su di essi sono rivolte alle transazioni, ossia sono di tipo *OLTP* (*On-Line Transaction Processing*).

[1] Vedere riferimento bibliografico [14].

A.3 Data Warehouse

Al contrario dei database operazionali, i Data Warehouse sono ideati per condurre analisi a scopo decisionale, mirate quindi ad offrire una consultazione di tipo statistico/analitico.
Difatti un Data Warehouse viene definito come una raccolta di dati:

integrata
: In esso devono confluire dati da fonti eterogenee, deve adottare quindi meccanismi di standardizzazione e separazione logica.

orientata al soggetto
: I dati non sono organizzati in base al particolare applicativo del dominio o settore aziendale, ma vengono archiviati per essere facilmente consultati dall'utente.

variabile nel tempo
: Ogni istanza deve essere datata e conservata, mantenendo quindi lo storico dei dati.

non volatile
: Tutte le istanze già presenti non possono essere più eliminate, modificate o aggiornate: è possibile solo inserirne di nuove tramite caricamento da fonti esterne, permettendo poi di accedere ai dati del DW solo in lettura.

di supporto ai processi decisionali
: Le sue caratteristiche permettono di utilizzare i dati per scopi analitici e statistici.

È possibile poi interagire con essi tramite interrogazioni di tipo *OLAP* (*On-Line Analytical Processing*), molto più complesse ed in grado di navigare i dati su più dimensioni e livelli.

A.3.1 OLAP

I Data Warehouse sono ottimizzati per query di tipo OLAP: un'analisi di questo tipo permette di consultare e visionare i dati da parte di un operatore, in genere anche solo tramite una tabella pivot; successivamente l'utente potrà eseguire varie operazioni direttamente su

quest'ultima, aggregando, filtrando e selezionando i dati nel modo che desidera.

Le tipiche operazioni che possono essere eseguite sono:

roll-up
: Aumenta il livello di aggregazione andando più sul generale (le dimensioni della gerarchia vengono raggruppate al livello precedente).

drill-down
: Diminuisce il livello di aggregazione scendendo più nei dettagli (le dimensioni della gerarchia vengono espanse al livello successivo).

slice
: Seleziona in base ad una determinata dimensione ottenendo una "fetta" del cubo originale.

dice
: Seleziona in base a due o più dimensioni ottenendo un sottocubo di quello originale.

pivot
: Ruota gli assi in un cuboide, facendo perno su una dimensione.

Tramite le query OLAP è possibile dunque navigare attraverso il cubo definito da uno schema che ne stabilisce le misure, le dimensioni e le gerarchie, nelle quali sono suddivise le informazioni.

A.4 Confronto tra i vari tipi di database

Le principali differenze tra database operazionali e Data Warehouse si possono riassumere nella seguente tabella:

	DB OPERAZIONALE	DATA WAREHOUSE
UTENTI	Impiegati.	Dirigenti.
NUMERO DI UTENTI	Numerosi.	Poco numerosi.
SCOPO	Operazionale: memorizzazione di tutte le attività quotidiane. Lo scopo finale dipende dall'applicazione.	Analitico/statistico: supporto alle decisioni strategiche.
TIPO DI QUERY	OLTP. Numerose transazioni brevi, semplici e predefinite di tipo read/write.	OLAP. Query complesse realizzate ad hoc prevalentemente solo read.
MODELLO	Orientato alle applicazioni. Dati normalizzati. Schema ER.	Orientato ai soggetti. Dati denormalizzati. Schema a stella o a fiocco di neve. Multidimensionalità.
OTTIMIZZAZIONE	Su una piccola parte del DB. Ottimizzazione per OLTP.	Su una vasta parte del DB. Ottimizzazione per OLAP.
FONTE DEI DATI	Direttamente dalle transazioni e/o materiale non elettronico aggiuntivo.	DB Operazionali, file ed altre fonti esterne. Sono integrati tramite strumenti di ETL.
INTEGRAZIONE DEI DATI	Dati integrati per applicazione, attività o processo.	Dati aggregati per soggetto.
TIPO DI DATI	Elementari. Sia numerici che alfanumerici.	Di sintesi. Prevalentemente numerici.
QUANTITÀ DEI DATI	Non molto grande.	Grande mole di dati.
QUALITÀ DEI DATI	In termini di integrità.	In termini di consistenza.
COPERTURA TEMPORALE	Dati correnti.	Dati correnti e storici.
AGGIORNAMENTO DATI	Continuo. Possibilità di sovrascrivere o eliminare transazioni.	Periodico. Possibilità di aggiunta di nuove istanze, ma non di modifica.

Per attuare operazioni di Data Mining occorre interfacciarsi a sistemi di Data Warehouse, poiché solo questi ultimi possiedono le caratteristiche di avere una grande quantità di dati da analizzare e di avere uno storico (quindi dati simili che variano nel tempo), necessarie agli algoritmi di Data Mining.

Appendice B

Le Business Intelligence

Le *Business Intelligence* sono delle applicazioni software prevalentemente sotto forma di suite di prodotti singoli, incentrati sui processi per la creazione e la gestione di database decisionali, allo scopo di trasformare dati ed informazioni in conoscenza. I sistemi di Business Intelligence sono dunque la tecnologia per il supporto alle decisioni.

B.1 Caratteristiche

Le BI, come già detto, sono strutturate in più componenti, ognuno dei quali si occupa di un particolare processo nella gestione dei dati. Esse sono ottimizzate ed adatte soprattutto per eseguire un lavoro di Data Warehouse, ossia sono in grado di prelevare i dati grezzi da diverse sorgenti, trasformarli ed organizzarli creando un unico database strutturato di grandi dimensioni, che è poi possibile consultare tramite strumenti di analisi OLAP, eseguendo le query direttamente su tabelle pivot. Infine dispongono in genere di tool di reportistica per documentare i risultati delle interrogazioni.

B.2 Pentaho

Una BI molto utilizzata è *Pentaho*, distribuita in 2 versioni: l'Enterprise Edition e la Community Edition[1]. L'unica sostanziale differenza

[1] Vedere riferimento bibliografico [2].

tra le due è data prevalentemente dall'assistenza tecnica presente nella EE, ma a livello software non vi sono evidenti dissimilarità. La CE è totalmente gratuita e questo la rende molto usata, mentre la EE è a pagamento.
I principali strumenti che compongono la suite sono:

- *BI Server*
  È il cuore della BI e contiene i principali strumenti per analizzare i dati di un database e produrre veloci report. Si occupa inoltre di rendere raggiungibili e disponibili a tutti i componenti i dati, le risorse condivise e gli aspetti generali configurati.

- *Data Integration (a.k.a. Kettle)*
  È lo strumento di ETL.

- *Metadata Editor*
  Utile per il disaccoppiamento logico dei dati sorgenti, ossia la creazione dei metadati.

- *Pentaho Analysis Services (a.k.a. Mondrian) / JPivot*
  Strumenti OLAP per l'analisi dei dati. Mondrian è il motore che elabora le informazioni e le query, mentre a JPivot è affidata la visualizzazione grafica.

- *Schema Workbench*
  Serve per definire gli schemi dei cubi per la navigazione OLAP.

- *Report Designer*
  Permette di creare report personalizzati.

- *Data Mining (a.k.a. WEKA - Waikato Environment for Knowledge Analysis)*
  Strumento capace di trovare pattern attraverso algoritmi di Data Mining e poi applicarli a dati ancora sconosciuti[2].

- *PAT (Pentaho Analysis Tool) / Saiku*
  Altri tool per l'analisi OLAP tramite tabella pivot.

[2]Per approfondimenti vedere la sezione successiva.

- *Design Studio*
  Serve per automatizzare sequenze di azioni e processi da eseguire all'interno della BI.

Alcuni di questi componenti sono sviluppati come progetti indipendenti rispetto alla realizzazione della suite principale di Pentaho e possono essere successivamente integrati ad esso.

B.2.1 WEKA

WEKA (*Waikato Environment for Knowledge Analysis*) è il componente di Pentaho utilizzato per la parte di Data Mining, tuttavia esso fa parte di un progetto indipendente sviluppato all'Università di Waikato, in Nuova Zelanda; è gratuito ed è concesso sotto licenza GNU[3].
WEKA è un tool scritto in java e comprende una raccolta di algoritmi di Machine Learning per le operazioni di Data Mining; gli algoritmi possono essere applicati direttamente ad un dataset o richiamati all'interno del proprio codice java.
Contiene strumenti per il data pre-processing, la classificazione, la regressione, il clustering, le regole associative e per la loro visualizzazione.
È progettato anche per potere dare la possibilità al programmatore di sviluppare dei nuovi algoritmi di Machine Learning, partendo da schemi predefiniti.
Può essere utilizzato come una semplice libreria, all'interno della propria applicazione java, richiamando nel codice di quest'ultima le relative API che sono messe a disposizione[4], tuttavia WEKA può essere utilizzato anche tramite altri diversi ambienti di sviluppo:

The Explorer
: È il modulo che consente di eseguire la fase di pre-processing su un dataset e poi di applicare su di esso gli algoritmi di Machine Learning desiderati, impostando il tutto tramite GUI; il dataset può essere caricato da diverse fonti. Per la fase di pre-processing è possibile applicare numerosi filtri per la selezione degli attributi.

[3]Vedere riferimento bibliografico [17].
[4]Vedere riferimento bibliografico [22].

Knowledge Flow

Il KF permette di preimpostare un flusso di task che devono essere eseguiti in sequenza per produrre il risultato di Data Mining voluto. Questo è permesso in maniera grafica trascinando i task generici messi a disposizione (suddivisi per categoria e compiti) nell'area di lavoro e collegandoli successivamente in cascata tra loro. Infine per ogni task inserito devono poi essere impostati i parametri specifici che consentono di personalizzare l'esecuzione.

The Experimenter

Presenta le stesse caratteristiche dell'Explorer, ma è più completo. Esso, a differenza dei precedenti tool, permette di eseguire esperimenti multipli, inoltre ha una parte dedicata all'analisi dei risultati molto più avanzata.

Simple CLI

La Simple Command Line Interface permette di eseguire i task desiderati tramite shell dei comandi; questo componente ovviamente viene usato quando si vuole eseguire qualcosa di molto semplice.

Appendice C

Algoritmi di Data Mining

In questa sezione vogliamo riportare alcuni tra i principali algoritmi di Machine Learning. Inoltre si vuole entrare più nel dettaglio in merito ad altri algoritmi già esposti, riportandone meglio il funzionamento.

C.1 Classificatori

Per questo genere di algoritmi occorre fare distinzione tra le varie tipologie:

Classificatori Decision Tree[1]:

- **Algoritmo di Hunt**
- **C4.5**
- **CART**

Classificatori Rule Based:

- **CN2**
- **Ripper**
- **1R**
- **C4.5rules**

[1]Per i decision tree bisogna prima di tutto soffermarsi anche sulle tecniche di costruzione dell'albero.

Classificatori Instance Based:

- **Rote-learner**
- **K-NN**

Classificatori misti (aggregatori):

- **AdaBoost.M1**
- **RandomForest**

C.1.1 Decision Tree

Costruzione di un albero

In questa sezione non verrà riportato un particolare algoritmo per estrapolare un modello di classificazione, ma linee guida per la realizzazione di un albero decisionale, riportando tecniche particolari per raggiungere tale scopo.

Specificare la condizione di test Si ha un *binary split* nel caso un nodo venga espanso in due figli, *multiway split* nel caso invece ve ne fossero di più. La forma della condizione dipende dalla natura dell'attributo: con attributi nominali ed ordinali, nella situazione di multiway split si hanno tante partizioni quanti sono i valori dell'attributo, altrimenti si ha una divisione in soli due insiemi (tuttavia nel caso di attributi ordinali non è detto che si riesca a preservare la caratteristica di order); con attributi continui occorre una conversione in attributi discreti o utilizzare condizioni test di confronto (<,>,=).

Determinare lo split migliore Nello split si tende a trovare quello che porta ad avere nodi figli più puri possibili, ossia contenenti istanze tutte della medesima classe.
A tal scopo occorre una misura di impurità calcolabile tramite:

GINI index:

$$gini(t) = 1 - \sum_j p\left(\frac{j}{t}\right)^2$$

entropy based:

$$entropy(t) = -\sum_j p\left(\frac{j}{t}\right) log\left[p\left(\frac{j}{t}\right)\right]$$

misclassification error:

$$error(t) = 1 - max_i p\left(\frac{i}{t}\right)$$

Nei primi due casi j indica un valore dell'attributo classe e $p(\frac{j}{t})$ il relativo numero di istanze nel nodo, mentre nel terzo caso $max\ p(\frac{i}{t})$ rappresenta il numero di istanze appartenenti alla classe di maggioranza del nodo.
La misura è minima quando il nodo contiene istanze appartenenti ad una sola classe, massima quando per ogni classe si ha un numero di istanze uguali.
Una volta scelta la misura di impurità[2], la si calcola per il nodo padre e si sceglie lo split che massimizza l'*information gap*, cioè la differenza tra l'impurità del padre e quella dei suoi figli:

$$GAIN = impurity(p) - \sum_i \frac{n_i}{n} impurity(i)$$

dove p è il nodo padre ed i uno dei suoi figli, n il numero di istanze nel nodo padre e n_i quelle nell'i-esimo figlio.

[2]Nel caso si scelga la misura di impurità basata su entropia, si tende a prediligere partizionamenti con molti nodi figli, puri ma ognuno contenente una quantità non elevata di istanze.
Per risolvere ciò, si cerca di correggere l'*information gain* massimizzando il *GAINRATIO*, dato dal rapporto:

$$GAINRATIO = \frac{GAIN}{SplitInfo}$$

dove lo *split info* introdotto nel calcolo vale:

$$SplitInfo = -\sum_i \frac{n_i}{n} log \frac{n_i}{n}$$

ed in cui i indica la partizione i-esima. Lo *split info* è tanto maggiore quanto maggiore è il numero di partizioni con bassa cardinalità.

Fermare la costruzione dell'albero La costruzione di un albero viene terminata generalmente:

- se un nodo contiene istanze tutte di una stessa classe (nella realtà non sempre avviene)

- se un nodo possiede tutte istanze con valori simili di attributi.

Algoritmo di Hunt

L'*algoritmo di Hunt* è uno dei primi algoritmi ed anche il più semplice per costruire un decision tree.
Considerato *Dt* l'insieme dei record che raggiungono un nodo:

1. se hanno la stessa classe, nodo foglia con tale classe

2. se *Dt* vuoto, nodo foglia con classe di default

3. altrimenti splitta il nodo su un attributo e procedi ricorsivamente ripetendo il procedimento

È molto banale ma permette di comprendere il funzionamento generale di un algoritmo di questo tipo.

C4.5

Il *C4.5* è uno degli algoritmi di classificazione decision tree più noti, ed è abbastanza complesso[3].
Le sue caratteristiche principali sono:

- espansione dei nodi in maniera *depth-first*

- uso di *information gain* per lo split ed *entropia* come misura di impurità.

- ordinamento degli attributi continui ad ogni nodo (rende più efficiente trovare la condizione di test)

- richiede di avere l'intero dataset in memoria centrale

[3]Vedere riferimento bibliografico [16].

CART

Classification And Regression Tree (*CART*) è un algoritmo usato anche per task di regressione o predizione numerica.
Idealmente è molto simile a *C4.5*, ma usa l'*information gain* con *GINI* come misura di impurità.

C.1.2 Rule Based

CN2

Con l'algoritmo *CN2*, nella costruzione di una regola si parte da un *LHS* vuoto, si aggiunge una congiunzione basandosi su entropia e si espande fino a quando si hanno miglioramenti di *information gain*; *RHS* è la classe di maggioranza delle istanze coperte dalla regola.

Ripper

L'algoritmo *Ripper* parte anch'esso dal generare una regola da un *LHS* vuoto.
Nell'aggiungere congiunzioni, la stima dell'*information gain* viene calcolata con la formula:

$$Gain(R0, R1) = t \left[log \frac{p_1}{p_1 + n_1} - log \frac{p_0}{p_0 + n_0} \right]$$

dove *R0* è la regola di partenza, *R1* quella estesa, p_0 e p_1 sono il numero di istanze della classe positiva (in un *2-class problem*) coperte da *R0* e *R1*; n_0 e n_1 sono quelle della classe negativa.
In un *2-class problem*, *Ripper* genera regole per la classe scelta come positiva; la negativa sarà quindi quella di default. In un *multi-class problem*, ordina le classi per cardinalità decrescente e parte a generare regole per quelle meno numerose.

1R

L'*1R* o *1-rule* genera set di regole equivalenti ad un albero decisionale di profondità *1*; benché possa apparire rudimentale, empiricamente si è riscontrata un'efficacia talvolta superiore a quella di tecniche più

complesse[4].
Per ogni attributo e per ogni suo valore viene generata una regola, poi viene contato quanto ricorre ciascuna regola e si assegna successivamente a *RHS* quella più frequente; infine viene calcolato l'*error rate*, scegliendo poi per tutti gli attributi il rule set che ha il valore più basso di quest'ultima misura.

C4.5rules

C4.5rules è un metodo indiretto che estrae rule set da un albero decisionale unpruned generato con *C4.5*, applica poi a ciascuna regola il pruning secondo il criterio visto per i metodi diretti.

C.1.3 Instance Based

Rote-learner

Il *Rote-learner* classifica una nuova istanza solo se gli attributi di quest'ultima hanno ugual valore di quelli di un'istanza di training.

K-NN

Il *k Nearest Neighbour* (*k-NN*) assegna la classe di maggioranza delle k istanze di training più vicine.
Risulta ovvio che in questo algoritmo è fondamentale la scelta della metrica con cui calcolare la distanza, che dipenderà dal tipo di attributi e dal dominio applicativo, e del valore di k, che dovrà essere indubbiamente dispari per evitare pareggi. Inoltre quest'ultimo parametro dovrà essere tarato opportunamente, poiché se fosse troppo piccolo renderebbe la classificazione sensibile al rumore, mentre se fosse troppo grande potrebbe considerare vicine anche istanze che non lo sono.

[4]Vedere riferimento bibliografico [9].

C.1.4 Misti

AdaBoost.M1

AdaBoost.M1 è una variante di *boosting*: lavora su training set pesati, ossia set in cui ad ogni istanza è associato un peso.
A questo punto l'errore verrà calcolato come:

$$errore = \frac{[\sum \ pesi \ dei \ misclassified]}{[\sum \ totale \ dei \ pesi]}$$

L'algoritmo risulta:

```
MODEL

  1. Assign equal weight to each training instance

  2. For each of t iterations:

     2.a. Apply learning algorithm to weighted dataset and
          store resulting model

     2.b. Compute error e of model on weighted dataset and
          store error

     2.c. If e equal to zero, or e greater or equal to
          0.5:  Terminate model generation

  3. For each instance in dataset:

     3.a. If instance classified correctly by model:

      (a) Multiply weight of instance by e / (1-e)

      (b) Normalize weight of all instances

CLASSIFICATION

  1. Assign weight of zero to all classes

  2. For each of the t (or less) models:

     2.a. Add -log(e / (1-e)) to weight of class predicted
          by model

     2.b. Return class with highest weight
```

Normalizzare i pesi significa fare in modo che ad ogni modifica la somma totale rimanga sempre la stessa.

RandomForest

RandomForest è un aggregatore di classificatori che si adatta bene a dataset sbilanciati; combina il *bagging* con la selezione casuale di feature[5].
Ciascun albero è costruito nel seguente modo:

1. sia N la cardinalità del training set e M il numero di feature o variabili

2. sia m il numero di variabili da considerare in un nodo, con m molto minore di N

3. per ciasun albero si costruisce un training set campionando n volte con reimbussolamento dal principale

4. in ciascun nodo si selezionano m variabili casuali, tenendo il gruppo che produce il miglior split

5. ogni albero è costruito per intero e non sottoposto a pruning

C.2 Clusterizzatori

I principali algoritmi di clustering utilizzati sono:

- **K-means**
  Produce un clustering partizionale con cluster center-based.

- **Agglomerative Hierarchical Clustering**
  Produce un clustering gerarchico.

- **DBSCAN**
  Produce un clustering partizionale non completo con cluster density-based.

[5]Vedere riferimento bibliografico [4].

K-means

L'algoritmo *K-means* si presenta in questo modo:

1. `Select K points as initial centroids`
2. `REPEAT`
   - 2.a. `Form K clusters, assign each point to its closest centroid`
   - 2.b. `Recompute the centroid of each cluster`
3. `UNTIL Centroids do not change`

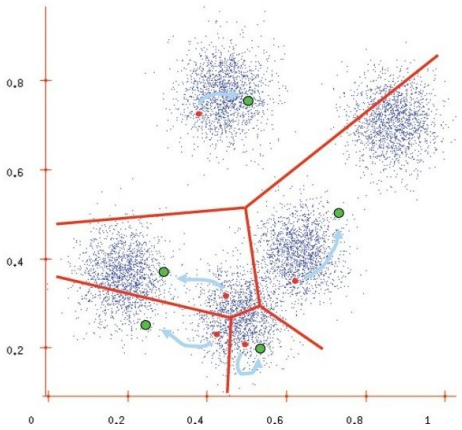

Figura C.1: *K-means*

L'input che bisogna fornire è K, ossia il numero di cluster che si vogliono; questo algoritmo converge ad una soluzione nelle prime iterazioni. È molto importante la scelta iniziale della posizione dei centroidi: essi possono essere presi in maniera casuale o si può utilizzare una tecnica più complessa:

1. scegli il primo centroide in maniera casuale o prendilo come centroide di tutti i punti.

2. per i successivi, prendi i punti più distanti dai centroidi correntemente fissati.

Se si dovesse incorrere in cluster vuoti, come a volte può capitare, si può ricorrere a due soluzioni differenti:

- si può rimpiazzare il centroide del cluster vuoto con un altro che sia il più distante dagli altri centroidi.

- si può scegliere come sostituto un punto del cluster a maggior *SSE* (*Sum of Squared Error*), valutata come:

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

  dove K è il numero di cluster, C_i un cluster, x un suo elemento e c_i il suo rispettivo centroide; *dist* è una misura di distanza.
  Più *SSE* è bassa e migliore è il clustering.
  Si possono adottare anche tecniche per migliorare *SSE* sia dopo il clustering finale (*postprocessing*), sia dopo ogni iterazione (*incremental updating*).
  Nel primo caso, ossia il *preprocessing*, si può aumentare K, ma questo non sempre è consentito, quindi in alternativa si può adottare un procedimento che alterna fasi in cui si splittano cluster, con altre in cui si uniscono:

  1. Per splittare si sceglie il cluster a maggior *SSE* o quello con la maggior deviazione standard su un attributo; si prende poi come centroide il punto più lontano dagli altri centroidi.
  2. Per unire si possono mettere insieme i due cluster che hanno i centroidi più vicini, oppure quelli che comportano il minor incremento della *SSE* complessiva.

K-means ha complessità spazio-temporale lineare, ma è molto sensibile ad outlier e fa fatica a trovare cluster con forme non sferiche; il suo più grosso limite rimane tuttavia il fatto di dover conoscere a priori il valore di K, ossia il numero di cluster finali.

Agglomerative Hierarchical Clustering

Con *Agglomerative Hierarchical Clustering* più che ad uno specifico algoritmo ci si riferisce ad un insieme di tecniche per produrre un clustering gerarchico secondo due approcci:

agglomerativo
: È il più usato tra i due; parte da singoli elementi ed ad ogni passo unisce le coppie di cluster più vicine.

divisorio
: Opposto all'agglomerativo, parte da un cluster contenente tutti i punti ed ad ogni passo decide come splittare, fintanto che non si arriva a cluster formati da un unico elemento.

L'*Agglomerative Hierarchical Clustering* si può sintetizzare nella seguente maniera:

```
1. Compute the proximity matrix if necessary
2. REPEAT
   2.a. Merge the closest two clusters
   2.b. Update the proximity matrix
3. UNTIL Only one cluster remains
```

Queste tecniche hanno complessità spazio-temporale esponenziale.

DBSCAN

DBSCAN individua cluster come regioni ad alta densità, distinguendole dalle altre a bassa densità.
Il suo comportamento è il seguente:

1. `Label all points as core, border or noise`
2. `Eliminate noise points`
3. `Put an edge between all points that are within Eps of each other`
4. `Make each group of connected core points into a separate cluster`
5. `Assign each border point to one of the clusters of its associated core`

La densità di un punto è il numero di punti attorno ad esso (compreso esso stesso) in un raggio *Eps*; dimensionare quest'ultimo parametro è fondamentale per un buon risultato dell'algoritmo.
Stimata la densità per ogni punto, si possono avere:

core points
: punti che hanno una densità maggiore di una soglia *MinPts* (da dimensionare come *Eps*).

border points
: punti con densità inferiore a *MinPts*, ma che sono vicini ad un core point.

noise o *background*
: punti che non sono né core né border point.

Questo algoritmo ha complessità spaziale lineare, ma esponenziale in tempo.
Il maggior vantaggio di questo genere di algoritmi è la capacità di individuare cluster con forme anche convesse.

C.3 Regole associative

Tra gli algoritmi utilizzati per le regole associative si menzionerà solo l'**algoritmo Apriori**.

Algoritmo Apriori

Per ridurre il numero di itemset da esplorare all'interno dell'albero generato da k item ci si affida al principio *Apriori*, che si basa sul fatto che se un itemset è frequente, lo sono anche tutti i suoi subset. Il metodo Apriori usa il supporto come primo filtro per individuare gli itemset frequenti; se un itemset è non-frequente, lo sono anche i suoi superset e questo permette di potare buona parte dell'albero.
Tutto ciò è possibile grazie alla *proprietà di antimonotonocità del supporto* che si può esprimere come:

$$X \subseteq Y \Rightarrow support(Y) \leqslant support(X) \qquad \forall X, Y$$

dove X e Y sono itemset.
L'algoritmo Apriori, sfruttando tale proprietà, si sviluppa nel seguente modo:

1. `Let k=1`
2. `Generate frequent itemsets of length 1`
3. `Repeat until no new frequent itemset are identified`
4. `Generate length k+1 candidate itemsets from length k frequent itemsets`
5. `Prune candidate itemsets containing sets of length k+1 that are infrequent`
6. `Count the support of each candidate by scanning the DB`
7. `Eliminate candidates that are infrequent`

Per ogni iterazione in cui vengono generati i candidati, due k-itemset vengono uniti se e solo se hanno i *k-1* item uguali, e questo prevede un ordinamento alfabetico o numerico degli item.
Invece di controllare se ogni itemset è contenuto in ciascuna transazione, questo algoritmo prevede dei meccanismi efficienti per l'aggiornamento del supporto: per ogni transazione vengono enumerati gli itemset contenuti in essa, per poi essere usati per aggiornare una struttura

dati hash tree che contiene per ogni itemset il relativo supporto. I principali fattori che influenzano la complessità dell'algoritmo Apriori nel trovare gli itemset frequenti sono il dimensionamento di *minsup*, il numero di transazioni ed il numero di item.
Una volta trovati gli item frequenti, si estraggono le regole associative che hanno confidenza maggiore di *confmin*; la quantità di regole ricavabili da un k-itemset sono 2^k; ognuna consiste nel partizionare in due subset X e Y ad intersezione nulla.
Sfruttando la proprietà che regole generate da uno stesso itemset hanno confidenza antimonotona, preso $\{A, B, C, D\}$ si ha che:

$$conf(ABC \Rightarrow D) \geqslant conf(AB \Rightarrow CD) \geqslant conf(A \Rightarrow BCD)$$

Nell'algoritmo Apriori una regola viene generata unendone due che hanno lo stesso prefisso nel conseguente[6], si esegue poi il *pruning* se non si rispettano i requisiti su *confmin*.

[6]Per esempio unendo $CD \Rightarrow AB$ e $BD \Rightarrow AC$ si ottiene $D \Rightarrow ABC$

Elenco delle figure

Bibliografia

[1] *Mono. Cross platform, open source .NET development framework.* http://www.mono-project.com.

[2] *Pentaho. Business analytics and business intelligence leaders.* http://www.pentaho.com.

[3] A. Azevedo and M. F. Santos. *KDD, SEMMA and CRISP-DM: a parallel overview.* In *IADIS European Conf. Data Mining '08*, pages 182–185, 2008.

[4] L. Breiman. *Random Forests. Machine Learning*, 2001.

[5] Canonical Ltd. *Ubuntu*, 2011. http://www.ubuntu.com.

[6] A. Castori. *Sviluppo di sistemi per l'automazione di processi di Data Mining con applicazione al churn della clientela.* Master's thesis, Università di Bologna, 2011.

[7] Y. Freund and R. E. Schapire. *A Short Introduction to Boosting. Journal of Japanese Society for Artificial Intelligence*, 1999.

[8] J. Frijters. *IKVM.NET*, 2002-2011. http://www.ikvm.net.

[9] W. Iba and P. Langley. *Induction of One-Level Decision Trees.* Ninth International Workshop on Machine Learning, Morgan Kaufmann, 1999.

[10] Microsoft. *.NET Framework 4*, 2011. http://msdn.microsoft.com.

[11] G. Moro. *Analisi e previsione dei clienti a rischio di abbandono*, 2010.

[12] P. Norvig and S. J. Russell. *Intelligenza Artificiale: Un approccio moderno.* Pearson, 2005.

[13] The Npgsql Development Team. *Npgsql: User's Manual*, 2.0 edition, 2002-2007. Npgsql: .Net Data Provider for Postgresql.

[14] A. Pini. *Processo di estrazione della conoscenza dai dati a scopo decisionale: il Knowledge Discovery in Databases con Pentaho.* Progetto presso l'Università di Bologna, 2011.

[15] The PostgreSQL Global Development Group. *PostgreSQL 9.1.2 Documentation*, 1996-2011.

[16] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
[17] R. R.Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, and D. Scuse. *WEKA Manual for Version 3-6-4.* The University of Waikato, 2002-2010.
[18] C. Sartori. *Valutazione dei risultati del Data Mining.*
[19] C. Shearer. *The CRISP-DM model: the new blueprint for Data Mining. Journal of Data Warehousing*, 2000.
[20] Sun Microsystems Inc. *JDK 6 Documentation*, 1995-2010.
[21] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining.* Addison-Wesley, 2006.
[22] University of Waikato. *Weka API 3.6.*
[23] I. H. Witten, E. Frank, and M. A.Hall. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, third edition, 2011.