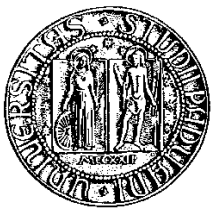

Università degli Studi di Padova

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingeneria Informatica

Tesi di Laurea Magistrale

Utilizzo di tecniche di Data Mining per la previsione di situazioni anomale in un sistema informativo aziendale

Relatore: Andrea Pietracaprina
Correlatore: Alberto Cavallarin
Correlatore: Oliviero Trivellato

Laureando: Davide Zannirato

8 Luglio 2014

He did not say that because he knew
that if you said a good thing
it might hot happen.

Hernest Hemingway - The old man and the sea

Non lo disse ad alta voce
perché sapeva che a dirle,
le cose belle non succedono.

Hernest Hemingway - Il vecchio e il mare

Autore: Davide Zannirato

Indice

Capitolo 1

Introduzione

La diffusione dei social network, l'importanza di carpire le preferenze degli utenti nei siti di e-shopping, le parole chiave utilizzate nei motori di ricerca, lo sviluppo della bioinformatica ... sono tutti esempi di enormi quantità di dati memorizzati quotidianamente, a partire dai colossi del web come Google, Facebook, Amazon e Twitter fino ad arrivare a qualsiasi sito di shopping online o istituto di ricerca. Qualcuno lo chiama web 3.0, ma è più noto come fenomeno dei Big Data. Secondo una ricerca IBM, ogni giorno vengono creati 2,5 quintilioni di byte, mentre il 90% dei dati esistenti è stato prodotto negli ultimi 2 anni. Un ritmo impressionante, che ha fatto in modo che le aziende iniziassero ad interrogarsi se fosse possibile ricavare informazioni utili celate in questi report archiviati. Queste motivazioni hanno aumentato l'importanza di discipline come il *Data Mining*, ovvero il passo analitico del processo più generale del *Knowledge Discovery and Data Mining* (KDD), che ha come scopo quello di estrarre informazioni interessanti da dataset, facendo uso di tecniche avanzate provenienti da altri settori, come ad esempio Intelligenza Artificiale, Machine Learning, Statistica e Basi di Dati. Durante il secondo anno di Ingegneria Informatica magistrale, grazie al corso "Data Mining" del Prof. Pietracaprina Andrea, mi sono appassionato a queste tematiche e ho pensato alla Tesi come opportunità per approfondire questi concetti per risolvere problematiche reali in ambito enterprise. Da queste premesse si è creata l'opportunità di uno stage, presso la sede padovana dell'azienda InfoCert S.p.a., la quale ha come core business servizi legati alla posta elettronica certificata (PEC), che comportano pesanti flussi di report giornalieri da archiviare: circa 50 GB solo per le componenti di frontend. A partire da questi dati è stata proposta un'attività di studio e analisi preliminare per la progettazione di un modello che riesca a prevedere degli allarmi di sistema, che si verificano a cadenza del tutto casuale. Il lavoro svolto è quindi iniziato con una prima fase di raccolta informazioni, durante la quale è stata studiata la struttura del sistema informatico aziendale, il flusso dei dati in input/output e le fondamenta teoriche necessarie. Successivamente sono stati analizzati i file di log per comprendere quali informazioni utili si possono estrarre con un occhio di riguardo all'obiettivo finale, mettendo a punto una strategia di attacco al problema. In base a questi dati sono stati poi progettati dei programmi atti all'estrazione e formattazione di queste informazioni, per andare in seguito a creare il dataset definitivo utile per l'attività di data mining. Infine sono stati svolti diversi test, per valutare la bontà della soluzione proposta.

1.1 Obiettivi

Dai colloqui preliminari è emerso che il sistema informativo aziendale soffre di sporadici sovraccarichi di sistema con cadenza random, che provocano un allungamento dei tempi di risposta e che vanno ad inficiare la disponibilità dei servizi offerti. Lo scopo principe del lavoro è quello di testare se attraverso i paradigmi del Data Mining applicati ai file di log del sistema è possibile prevedere con una buona accuratezza eventuali allarmi di sistema futuri, gettando le basi per lo studio, la progettazione e la realizzazione di un prototipo software adatto per tale scopo.

1.2 Azienda ospitante

InfoCert S.p.A. (`www.infocert.it`) è specializzata nella progettazione e nello sviluppo di soluzioni informatiche ad alto valore tecnologico per la dematerializzazione dei processi documentali, attraverso componenti di gestione documentale, conservazione sostitutiva, firma digitale, posta elettronica certificata ed Enterprise Content Management (ECM), il tutto con pieno valore legale. Con sedi a Roma, Milano e Padova, InfoCert è leader a livello europeo per i processi di conservazione sostitutiva dei documenti a norma di legge e per i servizi di Posta Elettronica Certificata ed è la prima Certification Authority in Italia per la firma digitale. InfoCert offre soluzioni verticali ad aziende operanti nel settore Bancario, Assicurativo, Farmaceutico, Manifatturiero, Energy, Utilities, Distribuzione Commerciale, Ambiente, Qualità, Sicurezza e Sanità, per la Pubblica Amministrazione, per le Associazioni di Categoria e per gli Ordini Professionali. Oltre il 90% dei clienti utilizza i servizi erogati in cloud dai data center InfoCert certificati ISO 9001, 27001, 20000, mentre il restante 10% si affida a soluzioni in-house o ibride. InfoCert è inoltre azionista al 25% di *Sixtema*, il partner tecnologico per il comparto artigianale e le PMI del mondo CNA. I prodotti principali di InfoCert S.p.A. sono legati essenzialmente al contesto della Posta Elettronica Certificata (PEC), della firma digitale, della gestione documentale e protocollazione, a soluzioni per il commercio elettronico e al cloud, il tutto con il massimo occhio di riguardo verso l'aspetto "legal" del prodotto, ovvero la garanzia di conformità alle normative vigenti in materia di sicurezza, integrità dei dati, disponibilità del servizio e riservatezza.

1.3 Risultati Ottenuti

Il problema del failure prediction è stato e continua ad essere oggetto di studio in vari settori, visto il notevole contributo garantito da una segnalazione proattiva dei problemi. I risultati ottenuti sono in linea con le ricerche nel settore, che sono caratterizzate da un'accuratezza discreta dei risultati (circa il 70% degli allarmi previsti) a causa dello sbilanciamento dei dati disponibili, infatti le situazioni di allarme sono molto meno numerose rispetto a quelle di non allarme. Questo significa anche che sono necessari dati riguardanti un arco temporale ampio, in modo da coprire tutte i contesti problematici, oppure modelli di apprendimento automatizzato in contesti real-time. Dall'attività

svolta sono state individuate ed estratte dai vari file di log diverse features, organizzate a slot temporali di 5 minuti. Tramite una serie di algoritmi e di ottimizzazioni discussi nella Tesi è stato poi costruito il dataset definitivo, con le informazioni necessarie per rendere proattiva la classificazione rispetto all'allarme. Lo studio preliminare e le prove effettuate hanno evidenziato come l'approccio proposto sia in grado di individuare e separare con buona accuratezza le situazioni di allarme da quelle di regime normale, tuttavia sono necessari ulteriori dati per stabilizzare i dati e per costruire un modello previsionale il più generale possibile rispetto ai vari contesti di allarme.

1.4 Roadmap Tesi

La Tesi è così strutturata: nel Capitolo 2 vengono esposte le nozioni teoriche riguardanti i principali concetti utilizzati, come il processo KDD, il task di classificazione e le support vector machines, inoltre viene analizzato lo stato dell'arte sul tema del failure prediction e log mining. Il Capitolo 3 tratta tutte le informazioni raccolte riguardanti l'azienda e il relativo sistema informativo, la descrizione delle sorgenti software che hanno generato i file di log nonché il contenuto informativo presente in questi ultimi. Vengono poi descritti gli allarmi di sistema e in che modo sono segnalati. Nel Capitolo 4 viene inizialmente messa a punto una strategia operativa, successivamente seguendo gli step del processo KDD si descrivono con maggiore precisione le trasformazioni applicate ai dati grezzi fino ad ottenere il dataset definitivo. In seguito vengono esposte le prove di mining, con i risultati ottenuti. Le conclusioni finali sono stilate nel Capitolo 5.

Capitolo 2

Nozioni Preliminari

In questo capitolo vengono presentati i fondamenti teorici sui quali poggia l'attività di tesi, a partire da nozioni riguardanti il dominio del Data Mining, il task di classificazione, le Support Vector Machines, fino ad analizzare l'ambito del failure prediction e gli studi presenti in letteratura.

2.1 Big Data e Business Intelligence

Gartner definisce Big Data come "Elevato volume, elevata velocità e/o elevata varietà delle risorse informative, che richiedono nuove forme di elaborazione per consentire migliori processi decisionali, analisi in profondità e processi ottimizzati". Detto in altre parole, il termine si riferisce ad un insieme di dataset così grandi e complessi da rendere impossibile la gestione degli stessi con strumenti tradizionali, in qualsiasi fase, dall'acquisizione fino alla visualizzazione. Questo aspetto si colloca in un contesto tecnologico e temporale nel quale il volume dei dati cresce in maniera esponenziale, nascondendo informazioni implicite che possono migliorare i profitti aziendali. Non esiste inoltre una dimensione prefissata dei dati prodotti affinchè siano considerati "Big", ma tutto è relativo all'azienda in esame. Le dimensioni che li caratterizzano sono le cosiddette 3V alle quali si è aggiunta anche una quarta:

- **Volume:** capacità di acquisizione, archiviazione e accesso ai dati.
- **Velocità:** performance con le quali è possibile effettuare analisi in real-time o quasi.
- **Varietà:** da quante fonti provengono i dati, i quali possono essere o meno strutturati.
- **Veridicità:** qualità dei dati, ovvero quanta informazione utile è possibile estrarre da essi.

Si incontrano delle limitazioni causate dai Big Data in numerosi settori, come metereologia, genomica, simulazioni fisiche, studi biologici, informazioni legate al web e alla finanza. A titolo di esempio, l'acceleratore di particelle Large Hadron Collider (LHC) situato al CERN di Ginevra è oggetto di esperimenti che coinvolgono circa 150 milioni di sensori che inviano dati 40 milioni di volte al secondo, lavorando con meno dello 0.001% di questo flusso si produrrebbero all'anno 25 petabytes di dati. Se al contrario fossero

registrate tutte le informazioni provenienti dai sensori, si produrrebbero 500 exabytes al giorno e 150 milioni di petabytes all'anno, più di 200 volte tanto rispetto a tutte le altre sorgenti di dati al mondo messe assieme.

2.2 KDD e Data Mining

Come già accennato, Data Mining rappresenta il processo di scoperta automatica di informazione interessante da grandi quantità di dati, memorizzati su database, datawarehouse o altri repository. L'informazione interessante da ricercare ha le caratteristiche di essere potenzialmente utile, poiché non conosciuta precedentemente, e di non essere esplicitamente contenuta nei dati. Data Mining rappresenta un sottoprocesso del più generale Knowledge Discovery in Databases (KDD) [1]. Nella Figura 2.1 sono illustrati i passaggi del KDD, il quale inizia con una valutazione dell'obiettivo finale e del dominio applicativo, in secondo luogo si estraggono i dati grezzi necessari o un sottoinsieme di essi, ai quali verrà successivamente applicata una fase di cleaning e preprocessing, con il fine di eliminare le informazioni non utili, ridurre il rumore, gestire i dati mancanti e formattare i dati. Il quarto step prevede la riduzione e proiezione dei dati, ovvero la definizione delle *features* o *attributi*, cioè le colonne del dataset finale, e dei *records*, le righe. Eventualmente le features possono essere selezionate o ridotte con tecniche di features selection/extraction (vedi Paragrafo 2.3.3) per semplificare la mole di dati mantenendo solo quelli più correlati all'obiettivo finale. Successivamente, in base al goal finale, viene applicato uno dei task di Data Mining, come classificazione o clustering, ai quali segue una fase di analisi esplorativa, atta all'individuazione dell'algoritmo di Data Mining e dei relativi parametri che garantiscono le performance migliori. Una volta individuate queste informazioni si può avviare la fase di Data Mining, che avrà come output un insieme di pattern (informazione utile), che verrà analizzata e interpretata con la possibilità di ritornare a qualsiasi step precedente per reiterare il processo con il fine di raffinarlo. L'ultimo passaggio riguarda l'utilizzo dell'informazione utile scoperta o del modello, ad esempio implementando un' applicazione che sfrutti queste conoscenze o semplicemente documentando i risultati ottenuti.

I cosiddetti "Task" di Data Mining possono di 2 tipi: Decrittivi e Predittivi. I primi identificano pattern e relazioni tra i dati, mentre i secondi effettuano delle predizioni sui loro valori. I task Descrittivi si dividono in: Clustering (raggruppamento dei dati secondo criteri di somiglianza), Summarizations (mappare dati in sottoinsiemi con una semplice descrizione associata), Association Rules (trovare relazioni tra attributi e oggetti in termini di causa-effetto) e Sequence Analysis (trovare relazioni tra oggetti su un periodo temporale). I task predittivi invece sono: Classification (predire l'appartenenza di un oggetto ad una classe), Regression (predire il valore di un attributo di un oggetto), Time Series Analysis (simili alle regressioni ma tengono conto del fattore temporale), Prediction (stima del valore futuro).

2.2.1 Web mining e log mining

L'applicazione di tecniche di data mining per la scoperta di pattern da sorgenti inerenti il web si chiama *Web mining* [2], una tematica di grande sviluppo e interesse, trascinata dal boom del web dell'ultimo decennio. Web mining si divide in 3 principali categorie: *Web content mining*, *Web usage mining* e *Web structure mining*. La prima riguarda l'estrazione e l'integrazione di informazioni utili dal contenuto di pagine web, come testo e immagini, specialmente per relazionare queste informazioni alle ricerce effettuate da un

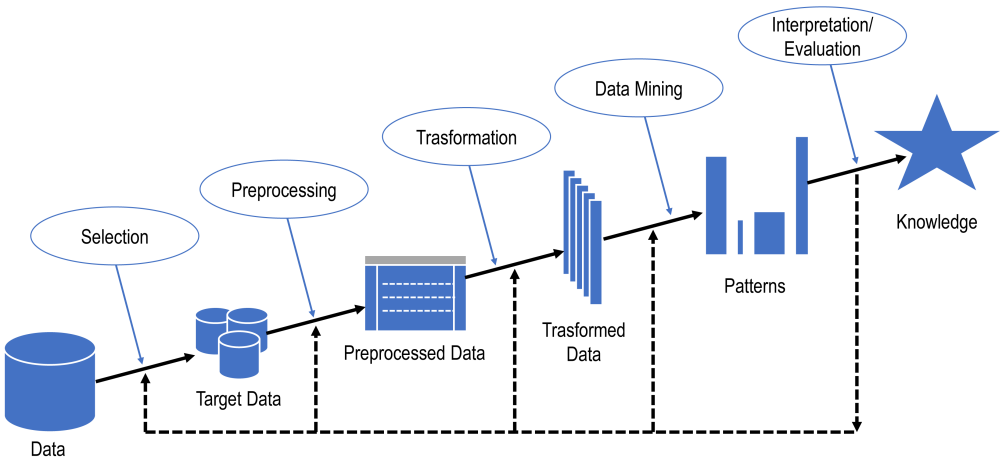

Figura 2.1: Processo KDD, Knowledge Discovery in Databases.

utente, come quanto avviene con i motori di ricerca, per garantire maggiore pertinenza dei risultati. Web usage mining mira ad estrarre informazioni utili direttamente dai file di log dei server web, per ricavare pattern riguardanti i comportamenti degli utenti, le risorse richieste, lo stato del sistema. In questa categoria rientra il lavoro svolto in questa Tesi. Infine Web structure mining tramite l'utilizzo della teoria dei grafi analizza i collegamenti tra le pagine (hyperlink) o la struttura interna delle pagine stesse per l'estrazione di informazione utile.

2.3 Classification

Classificazione è il problema di identificare a quale di un insieme di categorie (classi) appartiene un nuovo record, sulla base di un training set di dati contenenti istanze la cui classe è nota. I record sono caratterizzati da un insieme di proprietà chiamate features (attributi), che possono essere categorici (ad esempio il gruppo sanguigno) o numerici (pressione sanguigna). Il fatto che il classificatore agisca a seconda di dati già esistenti rende la classificazione un'istanza del "supervised learning". A seguire la definizione rigorosa del task:

INPUT: training set $T = t_1, ..., t_N$ di N record
m attributi $t_i(A_1), t_i(A_2), ..., t_i(A_m)$ con $t_i(D_j) \in D_j$
$t_i(C) \in \Gamma \rightarrow$ finito, con Γ dominio delle classi
OUTPUT: Classification model $f : D_1 \times D_2 \times ... \times D_m \rightarrow \Gamma$

Il risultato della classificazione è un modello, che può essere di vario tipo a seconda dell'algoritmo utilizzato, i più noti sono gli alberi di decisione, classification basata su regole, classificatori Bayesiani, reti neurali e support vector machines (SVM). Si distinguono quindi le fasi principali di *training*, durante la quale viene generato il modello tramite un set di dati con le classi note, e la fase di *testing*, dove il modello va ad agire su dati nuovi senza classe.

	Predict positive	Predict negative
Actual positive	TP	FN
Actual negative	FP	TN

Tabella 2.1: Confusion matrix

2.3.1 Performance evaluation

La valutazione delle performance di un algoritmo di apprendimento è un aspetto fondamentale nel campo del machine learning. Non solo è importante per la comparazione di diversi algoritmi, in certi casi è una parte integrante dell'algoritmo stesso. L'accuratezza di classificazione è ovviamente la metrica principale più usata, ma in alcuni casi, com'è per i problemi di class imbalance, è fondamentale l'utilizzo di metriche alternative, come AUC. L'accuratezza di classificazione è definita come la percentuale dei record di *test* correttamente classificati dal modello, mentre l'error rate è pari a $1 - accuracy$. Si fa riferimento ai record di test per la valutazione in quanto fruiscono una stima più affidabile, non essendo stati utilizzati per il training del modello. Nel caso di utilizzo di record del training set si fa riferimento al training set error come percentuale dei record non classificati correttamente. Per esporre le metriche alternative bisogna prima definire la *confusion matrix*, vedi Tabella 2.1, nella quale vengono esplicitate le quantità veri positivi, falsi positivi, falsi negativi e veri negativi.

La metrica TPR (True positive rate) indica la percentuale dei record di classe positiva correttamente classificati $TPR = \frac{TP}{TP+FN}$, viene chiamata anche *recall* (r) o sensitivity. Analagamente si definisce TNR (True negative rate) $TNR = \frac{TN}{TN+FP}$, chiamata anche specificity e FPR (False positive rate) $FPR = 1 - TNR = \frac{FP}{TN+FP}$. *Precision* è definita come $p = \frac{TP}{TP+FP}$ e combinandola con recall si ottiene $F_1 - measure = \frac{2rp}{r+p}$ che è quindi direttamente proporzionale a precision e recall. Un grafo ROC [3] (Receiver operating characteristic) è un grafo a 2 dimensioni con TPR plottato nell'asse Y e FPR nell'asse X, in questo modo vengono messi a confronto i benefici (TPR) e i costi (FPR). La classificazione perfetta è rappresentata dal punto (1,0). Informalmente un punto nello spazio ROC è migliore di un altro (a parità di FPR) se si trova più a nord rispetto ad esso. La linea diagonale a 45 gradi $y = x$ che passa per 0 e 1 viene chiamata "no-discrimination line" e rappresenta la strategia di indovinare in maniera random la classe, quindi un classificatore in quanto tale deve avere punti sopra questa linea. Molti classificatori, come gli alberi decisionali o i classificatori basati su regole, sono progettati per produrre una sola decisione di classe per ogni record e sono chiamati classificatori *discreti*. Se applicati ad un test-set producono un solo punto nello spazio ROC. Altri classificatori come le reti neurali e Naive Bayes sono chiamati classificatori *probabilistici* in quanto restituiscono per ogni record uno score di appartenenza ad una determinata classe. Questo punteggio può essere utilizzato come soglia per creare un classificatore binario, per ciascun valore di soglia si avrà un punto diverso nello spazio ROC, creando una *curva ROC*. Vista l'utilità di questa curva per la valutazione delle performance di un classificatore, esistono diversi modi per ricavare una funzione di score anche dai classificatori discreti. La metrica più importante ricavabile da questa curva è AUC (Area Under Curve), interpretabile come la probabilità che un classificatore assegni uno score più alto ad un record positivo casuale piuttosto che ad un record negativo casuale, detto in altre parole misura la capacità del classificatore di individuare e assegnare le classi. A titolo di esempio, nella Figura 2.2 sono illustrati 3 esempi di curve ROC, la curva A

è tipica di un classificatore perfetto, infatti AUC = 1, la curva B rappresenta un buon risultato, con AUC = 0.85 ed infine la linea C è la no-discrimination line con AUC = 0.5.

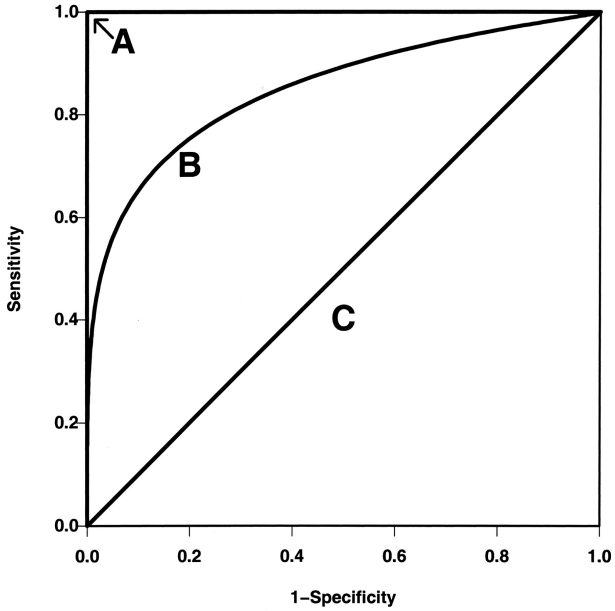

Figura 2.2: Esempi di curve ROC.

2.3.2 Overfitting e Validation

Dopo la fase di training spesso si ricorre alla fase di *validation*, utile per la selezione del modello migliore e per stimarne le performance, in questo modo è possibile impostare i parametri ottimali per migliorare l'accuratezza. La validazione è utile soprattutto per alleviare il problema di *overfitting*, che si presenta quando il modello si adatta troppo ai dati di training e non generalizza a sufficienza, causato sia da una fase di apprendimento troppo lunga, sia da pochi dati di training che non presentano nessuna relazione causale con la funzione target. In questa situazione l'accuratezza di classificazione sarà alta per i dati già utilizzati, mentre sarà pessima per i dati nuovi. Esistono diverse tecniche per la validazione [4], tra i quali ricordiamo Holdout Method, Random Subsampling, Cross-Validation e Bootstrap. La più utilizzata è la Cross Validation, chiamata anche K-fold Cross Validation, perché divide il training set in K insiemi della medesima cardinalità, ad ogni step K-1 insiemi sono utilizzati come training set e il modello risultante viene validato sul rimanente insieme, questo processo è ripetuto K volte, mediando i risultati ottenuti.

K-FOLD CROSS VALIDATION:
Training set iniziale $E \subseteq U$ universo di record
$E = E_1 \cup E_2 \cup ... \cup E_k$ con $E_i \cap E_j =$ per $i \neq j$
$|E_i| = \frac{|E|}{k}$
$M_i =$ modello costruito su $E - E_i$
Stima dell'accuracy: $acc = \frac{\sum_{i=1}^{k} |t \in E_i : M_i(t) = t(C)|}{|E|}$

Una volta trovato l'algoritmo di classificazione ottimale, i relativi parametri e aver costruito il modello, quest'ultimo sarà in grado di classificare nuovi record.

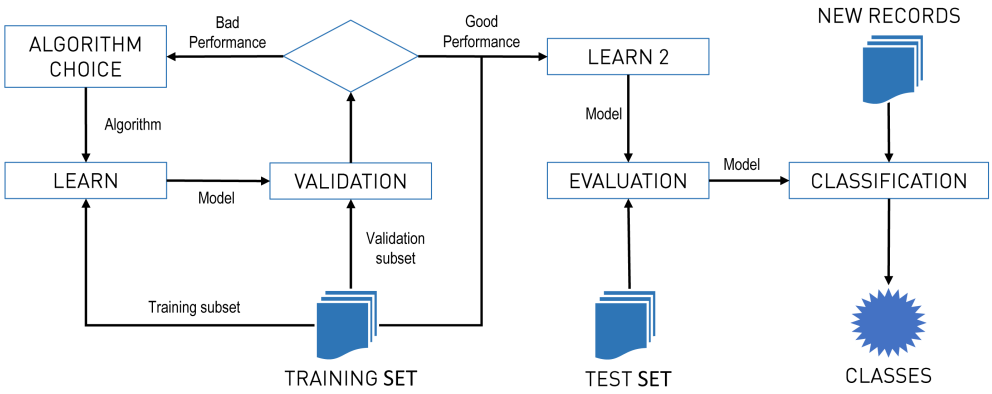

Figura 2.3: Processo di Classification.

2.3.3 Class imbalance

Può capitare che l'insieme di dati a disposizione per il training sia caratterizzato da una classe sotto-rappresentata [5], o classe rara, ovvero pochissimi record rispetto al totale sono etichettati con una certa classe. Questo fenomeno causa diversi problemi, primo tra tutti il fatto che anche se il classificatore presenta un'accuracy molto elevata (i.e. 99%) potrebbe comunque classificare male, infatti è sufficiente "predire" i record assegnando sempre la classe più numerosa per avere valori di accuratezza altissimi, il task però diventa automaticamente inutile. Inoltre la classe rara potrebbe essere quella più importante, da predire correttamente, la cui scarsità aumenta la difficoltà dell'operazione, perché il rischio di overfitting è maggiore. Alcuni scenari di queste situazioni si possono ritrovare nelle diagnosi mediche, fraud detection, credit approval e text categorizations. Le tecniche per scongiurare questo sbilanciamento si dividono in tre gruppi principali: textitsampling, cost-sensitive classification e dimensionality reduction.

	Predict positive	Predict negative
Actual positive	$C(0,0) = c_{00}$	$C(0,1) = c_{01}$
Actual negative	$C(1,0) = c_{10}$	$C(1,1) = c_{11}$

Tabella 2.2: Matrice di costo per cost-sensitive classification a 2 classi

2.3.4 Sampling

Sampling si divide in *under-sampling* e *over-sampling*. Nel primo caso si selezionano un sottoinsieme di record pari o vicino al numero di record della classe rara, mentre con il secondo approccio si replicano i record della classe rara fino a ottenere un numero uguale o vicino a quello della classe numerosa. Bisogna però porre maggiore attenzione in quanto questi approcci sono più inclini all'overfitting.

SMOTE

L'algoritmo più utilizzato per l'over-sampling è senza dubbio SMOTE (Synthetic Minority Over-sampling Technique) [6] introdotto nel 2002 da Chawla, Hall, e Kegelmeyer. Per ogni record della classe di minoranza, vengono rilevati i suoi k-vicini record di minoranza, dei quali ne vengono selezionati j in maniera random per generare dei campioni sintetici effettuando il join tra questi j vicini e il record iniziale. Ovviamente j varia a seconda di quanti nuovi dati sintetici si desiderano generare. Anche se SMOTE è sicuramente più efficiente di un qualsiasi random oversampling dal punto di vista dell'overfitting e della qualità della curva ROC, è possibile comunque incorrere in qualche problema, come ad esempio l'overgeneralization, causata dal sampling cieco, senza un occhio di riguardo alla classe maggioritaria. Infatti quando si hanno dataset con distribuzioni di classe altamente asimetriche, cioè con i record della classe rara sparsi rispetto alla classe di maggioranza, l' utilizzo di SMOTE non fa altro che accentuare questa "miscela" di classi. Sono state recentemente sviluppate delle versioni di SMOTE più efficienti, come Borderline-SMOTE, Safe-level-SMOTE e LN-SMOTE

2.3.5 Cost-sensitive classification

Per *cost-sensitive* classification/learning si intende l'assegnazione di un costo per predizioni false e per predizioni vere di ciascuna classe presente. In questo modo è possibile incentivare ad esempio la corretta classificazione dei pochi record della classe rara per dataset sbilanciati, questo si può ottenere assegnando un costo maggiore per classificazioni errate di questa classe (falsi negativi). Per definire questa pesatura viene utilizzata una matrice di costo C, nel caso di classificazione a 2 classi la matrice sarà come nella Tabella 2.2, nella quale i valori più significativi sono c_{01} ovvero il costo per i falsi negativi e c_{10} il costo dei falsi positivi.

2.3.6 Dimensionality reduction

Per *dimensionality reduction* si intende una riduzione dei dati in modo da ottenerne una rappresentazione ridotta (anche in termini di spazio su disco) e meno ridondante, ma che produce gli stessi risultati analitici. In questo modo si evita la "Curse of dimensionality", ovvero all'aumentare della dimensionalità dei dati questi diventano sempre più sparsi, richiedendo quindi più esempi per l'apprendimento. Intuitivamente, quando la dimensionalità si abbassa, lo stesso numero di esempi "riempie" maggiormente lo spazio

disponibile. In pratica superato un certo numero di attributi considerati le performance della classificazione degradano piuttosto di migliorare. Dimensionality reduction si divide in *feature selection* e *feature extraction* [7].

Feature Selection

Il processo di Feature Selection consiste nel selezionare un sottoinsieme di j features tra quelle disponibili che coincidono con il miglior sottoinsieme minimo in termini di performance di classificazione. Gli attributi selezionati saranno quelli più correlati all'obiettivo, che contengono più informazione predittiva per le classi. Vengono distinte 3 categorie principali:

Filter approccio per la rimozione di features irrilevanti prima di applicare l'algoritmo di apprendimento. Il sottoinsieme risultante verrà poi utilizzato per la fase di training. Ci sono 2 componenti fondamentali di questa tecnica, la prima è la politica di ricerca dei sottoinsiemi di features candidati, alcuni esempi sono Exhaustive Search, Best First, Simulated Annealing, Genetic Algorithm, Greedy Forward Selection, Greedy Backward Elimination, Scatter Search. Il secondo componente è la funzione di scoring utilizzata, tramite la quale assegnare un punteggio di performance ai vari candidati, alcuni esempi sono il Correlation Coefficient di Karl Pearson, Chi-square, Information Gain e Odds ratios. Alcuni approcci ritornano un ranking delle features piuttosto che un preciso migliore sottoinsieme di attributi, il punto di cut-off di questa lista viene scelto tramite cross-validation.

Wrapper metodo simile a Filter per la fase di generazione dei sottoinsiemi candidati ma che utilizza direttamente un algoritmo di apprendimento (i.e. Support Vector Machines) per la valutazione delle performance. Computazionalmente è più complesso rispetto ai metodi Filter e la probabilità di overfitting è maggiore.

Embedded approccio che applica il processo di feature selection come parte integrale dell'algoritmo di apprendimento. Esempi possono essere il metodo LASSO oppure l'algoritmo Recursive Feature Elimination applicato alle Support Vector Machines.

Correlation Coefficient

Il coefficiente di correlazione è un test statistico che misura la forza e la qualità di una relazione tra 2 variabili. Esso può variare da -1 a +1, per valori prossimi al +1 significa forte correlazione tra le variabili, 0 nessuna correlazione e -1 un'altissima non correlazione. Nei problemi di machine learning, il coefficiente di correlazione è usato per valutare quanto accuratamente una feature predice la classe target rispetto alle altre features. Questi attributi saranno successivamente ordinati a seconda del loro punteggio, in modo da poter scegliere il sottoinsieme dei più significativi. Data una qualsiasi feature X_i e la classe target Y, se la loro covarianza $cov(X_i, Y)$ e le varianze $var(X_i)$ e $var(Y)$ sono note a priori la correlazione può essere calcolata direttamente con:

$$R(i) = \frac{cov(X_i, Y)}{\sqrt{var(X_i) \cdot var(Y)}} \tag{2.1}$$

Se questi valori sono sconosciuti, una stima della correlazione può essere ottunuta tramite il coefficiente di correlazione prodotto-momento di Karl Pearson su un campio-

ne della popolazione (x_i, y), questa formula richiede soltanto di trovare la media tra ciascuna feature e il target per calcolare:

$$R(i) = \frac{\sum_{k=1}^{m}(x_{k,i} - \bar{x}_i)(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{m}(x_{k,i} - \bar{x}_i)^2 \cdot \sum_{k=1}^{m}(y_k - \bar{y}_i)^2}} \tag{2.2}$$

dove m è il numero di punti, $\bar{x}_i$ è la media della i-esima feature, e $x_{k,i}$ è il valore della i-esima feature nel k-esimo punto. Bisogna ricordare però che la correlazione evidenzia solo le relazioni lineari tra le variabili, che potrebbero quindi avere una relazione non-lineare. Un altro aspetto da considerare è come si costruisce il rank delle features, se questo viene generato in base al valore di score, prendendo solo le features con il punteggio più alto o vice-versa quelle con il punteggio più basso, si rischia di non scegliere gli attributi con la relazione più forte con la classe. Al contrario se si selgono le features con il valore assoluto più elevato non è possibile stabilire la frazione di attributi con punteggio positivo o negativo bisogna scegliere per ottenere i risultati migliori [8].

Feature Extraction

Con il processo di feature extraction invece si trasformano le features esistenti in uno spazio dimensionale minore, con il fine di estrarre l'informazione rilevante in modo da eseguire il task in questi dati piuttosto che sull'intero dataset. La tecnica per eccellenza è la Principal Component Analysis (PCA) inventata da Karl Pearson nel 1901, che tramite una trasformazione ortogonale converte i dati di possibili features correlate in un insieme di valori di variabili linearmente non correlate, chiamate principal components. Una variante di PCA è la Latent Semantic Analysis, che facendo uso di SVD (Singular Value Decomposition) analizza le relazioni tra un insieme di documenti e i termini in essi contenuti.

2.4 SVM: Support Vector Machines

Le Support Vector Machines (SVM) [9] sono modelli di apprendimento supervisionati associati ad algoritmi di apprendimento che analizzano i dati alla ricerca di pattern. Sono state inizialmente introdotte da Vladimir N. Vapnik e successivamente aggiornate con l'approccio soft-margin da Corinna Cortes e dallo stesso Vapnik nel 1993 [10]. Le SVM sono usate per la classificazione o la regressione. Dato un insieme (detto *training set*) di coppie (dato, classe) dove classe appartiene a -1, +1 nella forma più semplice, l'algoritmo di apprendimento della SVM genera un modello che assegna ogni dato in input ad una o all'altra classe. Il modello mappa i punti nello spazio in modo tale che dati di classi diverse siano il più possibile distanti tra loro. Una SVM costruisce un iperpiano o un insieme di iperpiani in uno spazio multi-dimensionale o di infinite dimensioni che viene usato poi per la classificazione o per la regressione. Un buon risultato si può ottenere in maniera semplice cercando l'iperpiano che massimizza il margine di separazione tra le classi. Questo viene chiamato *Margine Funzionale (Functional Margin).* Ci sono casi in cui l'insieme dei dati non è linearmente separabile in uno spazio. In questo caso si può mappare il training set (chiamato anche input-space) in uno spazio con un numero superiore di dimensioni (il feature-space), sperando di ottenere una separazione più semplice. La computazione nel feature-space è però in genere molto onerosa, a causa appunto dell'elevata dimensionalità. La soluzione è il cosiddetto *kernel trick*, ovvero la sostituzione di tutti i prodotti scalari con una *kernel function k(x,y)*, in modo

da evitare il calcolo esplicito delle coordinate di ciascun punto. Con questo trucco le SVM possono quindi eseguire efficacemente anche una classificazione non lineare.

2.4.1 SVM Lineari

I dati di training sono costituiti da coppie di valori di cui il primo appartiene ad un vettore reale di p dimensioni mentre il secondo è la classe a cui appartiene. L'insieme dei dati di training D può essere scritto nella seguente forma:

$$D = \{(\mathbf{x}_i, \mathbf{y}_i) | \mathbf{x}_i \in \Re^p, \mathbf{y}_i \in \{-1, 1\}\}_{i=1}^n \tag{2.3}$$

Si vuole cercare l'iperpiano con il massimo margine che divida i punti con valore 1 da quelli con valore -1. Un iperpiano è definito da tutti i punti che soddisfano la formula $\mathbf{w} \cdot \mathbf{x} - b = 0$ in cui l'operatore $\cdot$ indica il prodotto interno e $\mathbf{w}$ è un vettore normale all'iperpiano.

Qualora i dati fossero linearmente separabili si possono selezionare due iperpiani cercando di massimizzare distanza tra loro. Gli iperpiani che si individuano sono descritti dalle seguenti equazioni:

$$\mathbf{w} \cdot \mathbf{x} - b = 1 \tag{2.4}$$

e

$$\mathbf{w} \cdot \mathbf{x} - b = -1. \tag{2.5}$$

La distanza tra i due iperpiani risulta quindi:

$$\frac{2}{\|\mathbf{w}\|} \tag{2.6}$$

Essendo costante il termine al numeratore, l'obiettivo diventa la minimizzazione di $\|\mathbf{w}\|$. Per evitare che i punti cadano all'interno del margine é necessario aggiungere i seguenti vincoli: per ogni i si ha:

$$\begin{aligned} &\mathbf{w} \cdot \mathbf{x} - b \geq 1 \qquad \text{se il punto } \mathbf{x}_i \text{ appartiene alla prima classe} \\ &\mathbf{w} \cdot \mathbf{x} - b \leq -1 \qquad \text{se il punto } \mathbf{x}_i \text{ appartiene alla seconda classe} \end{aligned} \tag{2.7}$$

Così facendo ogni punto sara mappato all'interno di una classe e, al limite, potra cadere su uno dei due iperpiani selezionati per la separazione. I punti più vicini agli Optimal Separating Hyperplanes (OSH) sono quelli che servono per l'individuazione degli OSH: vengono detti perciò *support vectors*. Questo modello è il più semplice da implementare e puó essere utilizzato solo con dati lineramente separabili. La Figura 1.1 mostra un esempio di dati linearmente separabili collocati sul piano cartesiano: i dati in nero (che appartengono alla prima classe) vengono separati da quelli in bianco (che appartengono alla seconda) utilizzando un iperpiano che non passa per l'origine.

Il problema di ottimizzazione precedentemente descritto può essere riscritto nel seguente modo, detto forma primale:

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha} \geq 0} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i \left(\mathbf{w} \cdot \mathbf{x}_i - b \right) - 1 \right] \right\} \tag{2.8}$$

Vista la natura quadratica della funzione obiettivo, il problema diviene quindi un problema di programmazione quadratica. Nella formulazione duale del problema, due sono i vantaggi principali: 1) I vincoli del problema duale vengono rimpiazzati da quelli dei

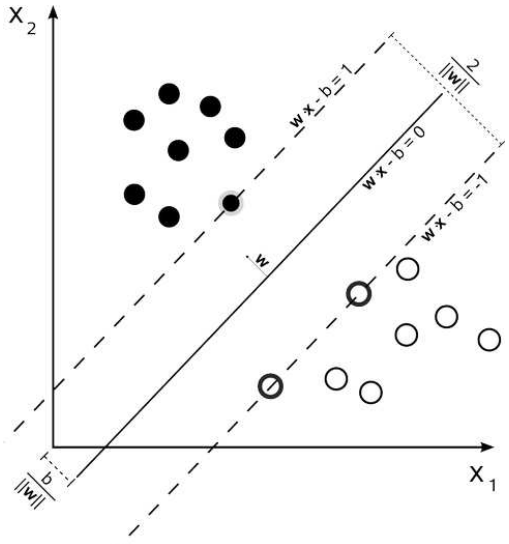

Figura 2.4: Esempio di dati linearmente separabili

moltiplicatori di Lagrange: questo li rende più semplici da utilizzare; 2) Con questa riformulazione del problema i dati di training appariranno come prodotto interno di vettori. Sfruttando questa proprietà cruciale si riesce generalizzare la procedura estendendola al caso non lineare. Nella forma duale si ha il seguente problema di massimizzazione:

$$\begin{aligned} maximize_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j = \\ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k\left(x_i, y_j\right) \\ \text{in cui:} \sum_{i=1}^{n} \alpha_i y_i = 0, \alpha_i \geq 0. \end{aligned} \tag{2.9}$$

in cui $k\left(\mathbf{x}_i, \mathbf{y}_j\right) = \mathbf{x}_i \cdot \mathbf{x}_j$ rappresenta la funzione Kernel. Riscrivendo il problema in forma duale risulta più semplice individuare l'iperpiano che porta al margine massimo. Per semplicità spesso è necessario che gli iperpiani passino per l'origine degli assi del sistema di coordinate. Questi iperpiani sono detti *unbiased*, gli altri invece sono detti biased. Per ottenere un iperpiano che passa per l'origine è sufficiente imporre $b = 0$ nell'equazione della forma primale oppure imporre il seguente vincolo nella forma duale:

$$\mathbf{w} = \sum_i \alpha_i y_i x_i \tag{2.10}$$

Fino ad ora sono state considerate SVM che classificano il training set senza commettere errori. Questo vincolo può essere rilassato tramite l'approccio chiamato *soft margin*. Così si possono classificare anche insiemi di dati non linearmente separabili, a costo di una tolleranza calcolata per l'errore sull'insieme di training. L'algoritmo di learning deve considerare quindi un compromesso tra la larghezza del margine e il numero di errori di training causati dalla scelta di tale margine. Nel caso di errori sull'insieme

di training, le equazioni, che rappresentavano i vincoli del problema di ottimizzazione discusso in precedenza, vengono violate. Questo problema viene risolto aggiungendo a tali vincoli delle variabili ξ dette *slack*:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x_i} + b \geq 1 - \xi_i \qquad & \text{se} \quad y_i = 1 \\ \mathbf{w} \cdot \mathbf{x_i} + b \leq -1 + \xi_i \qquad & \text{se} \quad y_i = -1 \end{aligned} \tag{2.11}$$

dove $\forall i : \xi_i \geq 0$.

In linea di principio si potrebbe applicare la stessa funzione obiettivo al problema di minimizzazione, ma questa non impone nessuna restrizione al numero massimo di errori che il confine scelto può commettere. Questo porterebbe quindi ad avere un margine molto ampio ma probabilmente a costo di troppi errori di train (bias). La soluzione è una funzione obiettivo che penalizza grandi valori per le variabili slack:

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} + C(\sum_{i=1}^{N} \xi_i)^k \tag{2.12}$$

dove C e k sono dei parametri da impostare sperimentalmente.

2.4.2 SVM Non Lineari

Nel paragrafo precedente si è parlato della determinazione di una separazione lineare (retta o iperpiano) degli esempi di train nelle rispettive classi. Le SVM non lineari si applicano quando tale separazione in classi non è più fattibile in maniera lineare, il classico esempio da letteratura è il dataset a doppia spirale in Fig. 2.5.

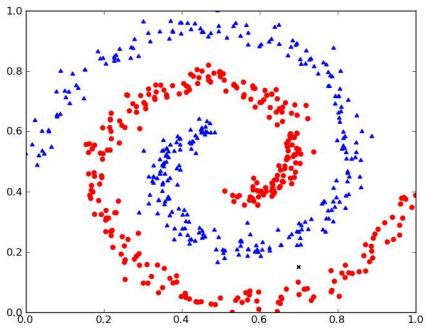

Figura 2.5: Il dataset non lineare Dual spiral.

Il trucco sta nel trasformare i dati dal loro spazio di coordinate originale in un nuovo spazio dove può essere effettuata una separazione lineare: $x \longmapsto \Phi(x)$. Dopo aver mappato i dati si possono applicare le tecniche studiate nelle sezioni precedenti (vedi esempio in Fig. 2.6).

Applicando questa mappatura dei dati possono sorgere alcuni aspetti negativi, infatti non è sempre chiaro che tipo di mappa utilizzare e la mappa può generare uno spazio con un numero molto alto di dimensioni (se non infinito), quindi potrebbe essere troppo costoso risolvere il problema di ottimizzazione dal punto di vista computazionale. Dalle considerazioni delle sezioni precedenti deriviamo che l'iperpiano di separazione ha questa

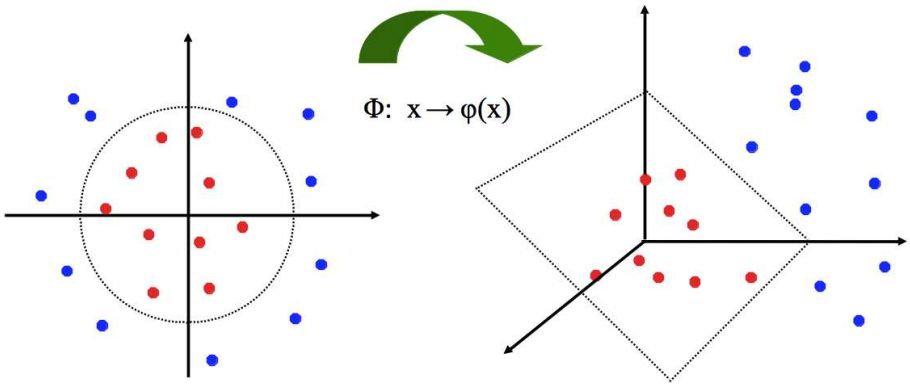

Figura 2.6: Mappa Φ applicata al dataset per rendere possibile una separazione lineare.

forma: $\mathbf{w} \cdot \Phi(\mathbf{x}) + b = 0$. E quindi una SVM non lineare può essere formalizzata come il problema di ottimizzazione:

$$\begin{aligned} &\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} \\ \text{applicato a:} \quad &y_i(\mathbf{w} \cdot \Phi(\mathbf{X}_i) + b) \geq 1, \qquad i = 1, 2, \ldots, N \end{aligned} \tag{2.13}$$

Seguendo lo stesso approccio per le SVM al caso lineare deriviamo il seguente Lagrangiano duale:

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{2.14}$$

Dopo che i λ_i sono ricavati utilizzando tecniche di programmazione quadratica, i parametri $\mathbf{w}$ e b si ottengono con le seguenti equazioni, del tutto analoghe a quelle riferite al problema lineare:

$$\mathbf{w} = \sum_i \lambda_i y_i \Phi(\mathbf{x}_i) \tag{2.15}$$

$$\lambda_i \{ y_i (\sum_j \lambda_j y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) + b) - 1 \} = 0 \tag{2.16}$$

E la funzione di test che mappa nella classe predetta dalla SVM un nuovo dato z:

$$f(z) = sign(\sum_{i=1}^{n} \lambda_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) + b) \tag{2.17}$$

Le ultime 2 Equazioni (2.16 e 2.17) utilizzano il prodotto interno tra 2 vettori nello spazio trasformato $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ e ciò può essere un problema dato che non c'è un limite alle dimensioni di tale spazio. Un metodo per superare questo ostacolo è conosciuto come *kernel trick*. Con il metodo *kernel trick* il prodotto interno tra i vettori trasformati viene sostituito da una funzione non lineare chiamata appunto *kernel*: detto V lo spazio trasformato, $K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_V$. La condizione matematica che questa funzione non lineare deve soddisfare è detta *Mercer's condition*. Questo principio assicura che la funzione kernel possa essere sempre espressa in termini del prodotto interno di 2

vettori nello spazio trasformato V e garantisce che il problema di ottimizzazione sia convesso e che la soluzione sia unica. La scelta del kernel dipende dal problema che si sta affrontando, da cosa si sta cercando di modellare, perchè è sempre possibile mappare l'input in uno spazio di dimensione maggiore del numero di punti del training set e produrre un classificatore perfetto, tuttavia questi generalizzerebbe malissimo su dati nuovi, per via dell'overfitting. Alcuni esempi di funzioni kernel possono essere le seguenti:

$$\begin{aligned} &\textbf{Kernel Lineare:} \\ &k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y} + C \\ &\textbf{Kernel Polinomiale:} \\ &k(\mathbf{x}, \mathbf{y}) = (\gamma\mathbf{x}^T\mathbf{y} + C)^d \\ &\textbf{Kernel Radial Basis (RBF):} \\ &k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right) \\ &\text{oppure: } k(\mathbf{x}, \mathbf{y}) = \exp\left(-\gamma\|\mathbf{x}-\mathbf{y}\|^2\right) \end{aligned} \tag{2.18}$$

2.5 Failure prediction

Con la continua crescita ed evoluzione dei sistemi informatici, la gestione proattiva dei problemi rappresenta un approccio efficiente per il miglioramento della disponibilità e dell'efficienza dei servizi. La chiave di volta per lo sviluppo di queste tecniche è l'*online failure prediction*, basato su un monitoraggio real-time di una serie di modelli e metodi che utilizzano lo stato corrente e passato del sistema (sia software che hardware) per generare previsioni e conseguenti alert. Si tratta logicamente di una tematica non facile, non solo a causa dell'imprevedibilità del futuro, ma anche dell'eterogeneità dei contesti e della variabilità sia delle applicazioni in esecuzione sia delle configurazioni hardware. Ciononostante è comunque possibile ottenere delle discrete prestazioni su previsioni nel breve termine. Come definito da Safner et al. [11], per online failure prediction si intende:

> "Identify *during runtime* wheather a failure will occur in the *near future* based on an assessment of the *monitored current system state*."

È chiaro che nel caso venga previsto un problema al sistema informatico nell'immediato futuro, gli addetti possono intervenire con il necessario anticipo per scongiurare questa situazione o per predisporre le necessarie contromisure. Se la predizione risulta accurata e precisa è possibile anche automatizzare tale processo in base al problema rilevato. L'output del modello predittivo può essere sia uno stato con dominio binario alarm/no-alarm, oppure una misura continua per giudicare lo stato corrente del sistema.

2.5.1 Definizioni

Secondo quanto riportato da Avižienis et al. [12], vengono definiti i concetti di *failure*, *fault*, *error* e *symptom*. Un *failure* si ha quando un servizio erogato devia rispetto lo stato di normalità, questo comportamento può essere rilevato sia da un operatore umano, sia da una sonda. Detto in altre parole, un failure accade quando si ha la percezione del problema relativamente allo stato dei servizi, infatti ci possono essere guasti all'interno del sistema, ma fino a quando il problema non si riscontra in un output non corretto non si ha failure. Per *error* si intende invece una deviazione dai valori

normali dello stato del sistema, non più dei servizi, un error è una parte del processo che porta ad un failure del servizio. Gli errors possono essere rilevati o non rilevati, ovvero il sistema non ne è necessariamente al corrente. I *faults* rappresentano le ipotetiche o certe cause dell'errore, per questo si dice che gli error sono la minifestazione dei faults. Gli errors oltre a provocare failures hanno come effetto collaterale uno sbilanciamento dei parametri dai valori normali, questi sono i *symptoms*. I symptoms sono riscontrabili dai file di log del sistema, nei quali sono quindi notificati in chiaro gli errors rilevati mentre sono "celati" quelli non rilevati, ma che rimangono comunque deducibili. Come si vedrà in seguito nel Capitolo 3, in questa Tesi il modello di predizione verrà addestrato per rilevare failure segnalate da sonde automatizzate in base ai sintomi rilevati dai file di log dei servizi.

2.5.2 Online prediction

Praticamente tutte le strategie di predizione online dei problemi poggiano gli algoritmi su schemi come quello presentato in Figura 2.7 dove t è il tempo corrente, al quale è disponibile una finestra di informazioni di dimensioni Δt_d. In base a questi valori il modello può prevedere problemi fino ad un massimo di Δt_l unità di tempo, chiamato anche lead-time. La predizione di allarme o non allarme ha una validità di Δt_p, dopodiché decade. Logicamente più è esteso questo tempo più si aumentano le probabilità di una corretta predizione. Infine Δt_w indica il tempo minimo per l'avviso proattivo di un problema, se inferiore a questa soglia qualsiasi contromisura si rivelerà vana.

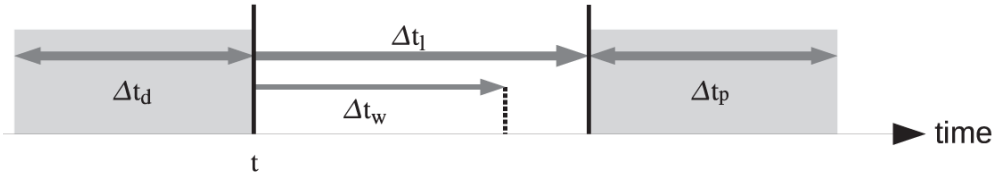

Figura 2.7: Gestione del tempo nell'online failure prediction.

2.5.3 Tassonomia dei metodi di online failure prediction

Come indicato nella Figura 2.8 si possono perseguire diverse strategie per la progettazione di un sistema di online failure prediction. A grandi linee, con *failure tracking* vengono tratte conclusioni sui failures futuri in base alle occorrenze passate, sia in termini di temporali che di tipologia di failures. *Symptom monitoring* si basa sul concetto di *system degradation*, ovvero il failure è riscontrabile attraverso un progressivo mutamento di alcune variabili di sistema, sia a livello hardware che software, memorizzate nei vari report di sistema archiviati. Nei failures considerati da questo metodo rientrano anche e soprattutto quelli non critici, più difficili da comprendere.*Detected error reporting* fa utilizzo dei file di log degli errori come input per prevedere quelli futuri, questi file di log sono composti da eventi e da feature categoriche come ID del componente o ID evento, al contrario del symptom monitoring dove le features sono soprattutto valori reali. Infine la ricerca di stati incorretti o errori inosservati in un sistema rientra nell'*undetected error auditing*, che si differenzia dalla categoria precedente perché riguarda anche i dati che non vengono utilizzati in quel preciso istante.

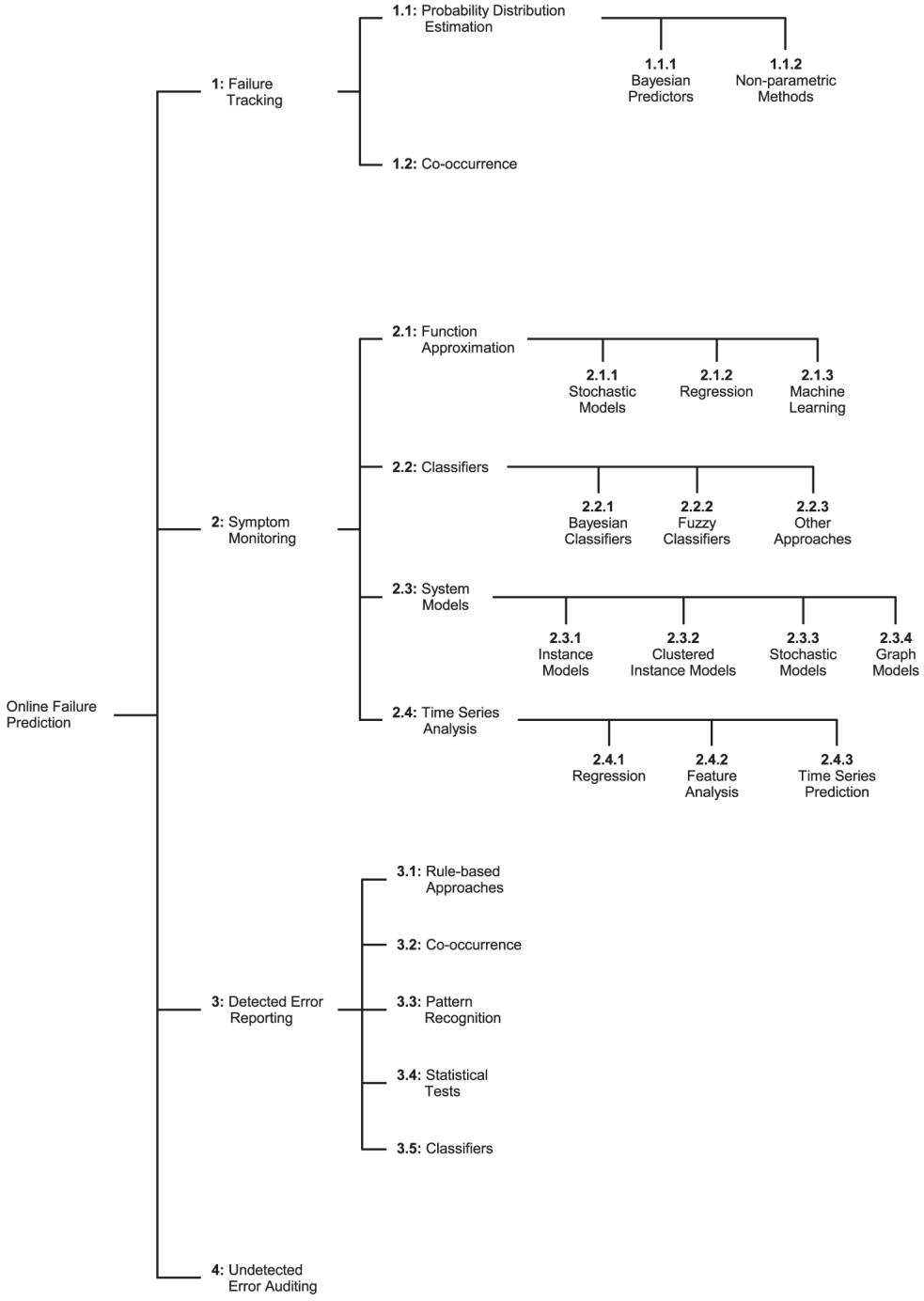

Figura 2.8: Tassonomia dei metodi di online failure prediction.

Lo studio condotto nella presente Tesi rientra in pieno nella seconda categoria pre-

sentata, *symptom monitoring*, poiché vengono estratte delle metriche di carico o stato del sistema dai file di log di diverse piattaforme. In particolare rientra nella categoria *classifiers* (Immagine 2.9), nella quale viene addestrato un modello di classificazione (vedi 2.3) sia su dati affetti da failure sia su dati riguardanti un stato normale, viene quindi stabilito un decision boundary che servirà durante l'esecuzione online per giudicare se i nuovi punti appartengono ad uno stato di failure o meno.

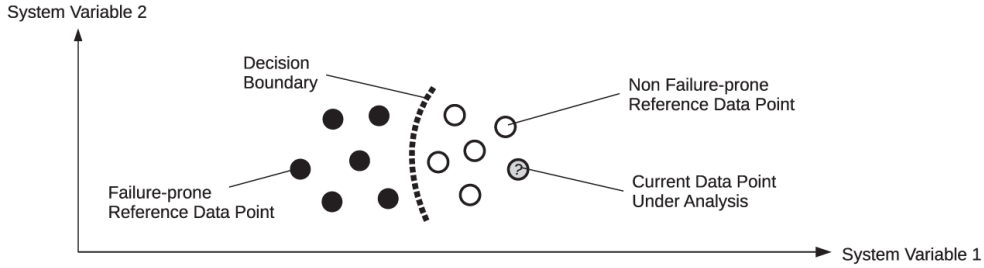

Figura 2.9: Online failure prediction tramite classificazione di osservazioni di metriche di sistema.

Diversi studi sono stati condotti in questa categoria, come ad esempio quello di Hamerly ed Elkan [13] del 2001, dove vengono descritti 2 approcci basati su failure prediction Bayesiana. Nel primo "NBEM" viene addestrato un modello naïve Bayes con l'algoritmo Expectation Maximization su un dataset reale di valori SMART di hard disk. Precisamente, è stato proposto un modello misto dove ogni sotto-modello naïve Bayes m è pesato da un modello precedente $P(m)$ e un algoritmo di expectation maximization viene utilizzato per tarare iterativamente sia il modello precedente sia le probabilità del sotto-modello. Nel secondo approccio invece viene addestrato un classificatore standard naïve Bayes dallo stesso dataset. Gli autori riportano che entrambi i modelli superano gli altri algoritmi esistenti in materia, specialmente NBEM, che riesce a prevedere il 30% dei failure. Successivamente lo stesso gruppo di ricerca [14] ha applicato le SMV (vedi 2.4) e un algoritmo di clustering non supervisionato.
Nel 2003 Turnbull e Alldrin [15] hanno utilizzato una rete radial basis function RBFN per classificare valori monitorati di sensori hardware come temperatura e voltaggi su schede madri. Più precisamente, gli N valori monitorati che rientrano in un data window sono stati rappresentati come vettore di feature che viene poi classificato come appartenente o meno alla categoria delle sequenze soggette a failure. Il risultato è stato ottimo (0.87 TPR e 0.1 FPR), però il dataset è equamente distribuito tra le 2 classi, non presenta sbilanciamenti.
Berenji et al. [16] hanno utilizzato un classificatore RBF basato su regole per etichettare quando un componente è guasto o meno: utilizzando regole Gaussiane, un modello diagnostico calcola un segnale basato su valori di input e output dei componenti, variabili da 0 (non guasto) a 1 (guasto). Le regole di base sono state algoritmicamente ricavate mediando i risultati ottenuti clusterizzando i dati di training, questo approccio è stato utilizzato dalla Nasa Ames Research Center in un impianto di combustione ibrida per predire failures che cambiano lentamente nel tempo.
Nello studio del 2005 di Bodík et al. [17] sono state analizzate le frequenze di click delle pagine web per identificare velocemente failures non critiche in un grande sito web commerciale. È stato utilizzato un classificatore naïve Bayes, il training set però non era etichettato, ovvero non si conoscevano inizialmente le occorrenze dei failures, quin-

di è stato usato un classificatore unsupervised, però l'output di quest'ultimo si rivela un punteggio anomalo. Kiciman e Fox [18] hanno costruito un albero decisionale (utilizzando algoritmi come C4.5 e ID3) dalle sequenze di componenti e da altre features come indirizzi IP che sono coinvolti nella gestione di una richiesta in una piattaforma di application server come J2EE, etichettando le richieste che hanno avuto buon fine e quelle che hanno fallito.
Nel 2007 Hoffmann et al. [19] hanno condotto uno studio comparativo tra diverse tecniche di creazione di modelli con l'obiettivo di predire il consumo di risorse in un server web Apache. I modelli presi in esame riguardano UBF (universal basis function), un approccio sviluppato dagli autori e derivato da RBF che utilizza una combinazione convessa pesata di 2 funzioni kernel piuttosto che un kernel singolo, lo stesso RBF, le SVM e ML (multivariate linear regression). Ne è risultato che UBF garantisce le migliori prestazioni per quanto concerne predizioni sulla quantità di memoria fisica libera, mentre i tempi di risposta del server sono stati predetti con accuratezza maggiore da SVM. Un risultato notevole ricavato dagli autori è che il problema della scelta del miglior sottoinsieme di variabili di input (Paragrafo 2.3.6) ha un'influenza molto maggiore rispetto alla scelta della tecnica di costruzione del modello.
Fu e Xu [20] hanno costruito una rete neurale per approssimare il numero di failures in un intervallo temporale, l'insieme di variabili di input è composto sia da un fattore di correlazione spazio-temporale delle failures, sia da altri attributi riguardanti ad esempio l'utilizzo di CPU o il numero di pacchetto trasmessi. Il modello risultante riesce ad approssimare il numero di failures in una finestra temporale ma non il momento delle occorrenze.

Le ricerche effettuate che più si avvicinano all'attività di Tesi svolta sono però le seguenti: Sahoo et al. [21] hanno sviluppato un sistema di controllo e predizione a partire dagli eventi dei file di log di sistema e dai report parametrici di un sistema cluster di 350 nodi su un periodo di attività pari ad un anno. È stato costruito un event-parser per la raccolta e costruzione di features sia discrete che continue, in grado di effettuare un filtraggio per i report falsi e altre operazioni manutentive. Sono state poi individuate delle features primarie per andare a comporre il dataset finale, che è risultato minore dell'1% di tutte le informazioni raccolte. Gli autori hanno scoperto che approcci basati su *Time-series analysis* si sono rivelati migliori per la stima di valori continui, come l'utilizzo della CPU e l'idle-time. Per la predizione di eventi critici rari, un classificatore basato su regole è stato in grado di predire correttamente il 70% degli eventi, basando l'apprendimento su finestre temporali di dati. Carlotta Domeniconi et al. [22] hanno sviluppato un framework chiamato SVD-SVM per la predizione di eventi in una rete di 750 hosts, a partire da un dataset formato da attributi categorici, rimarcando l'inefficienza di metodi time-series analysis per features non numeriche. Uno step iniziale di feature construction viene applicato, nel quale si costruiscono record raccogliendo informazioni su eventi attinenti finestre temporali chiamate *monitor-window* (MW), la cui dimensionalità dipende dal numero di eventi che si vuole considerare. Per ciascuna MW viene considerata una *warning-window* immediatamente successiva e anch'essa di durata definita e viene utilizzata per etichettare le MW come in allarme o meno, a seconda se sulla WW ne vengono rilevati. La fase successiva è di feature selection, attraverso la singular value decomposition (SVD) di questa matrice di attributi. Sono stati confrontati infine i risultati ottenuti con il metodo SVD-SVM con C4.5 e con SVM standard, SVD-SVM risulta il più accurato, seguito da vicino da SVM, prestazioni peggiori invece sono state rilevate con C4.5. Infine Weiss e Hirsh [23], hanno sviluppato timeweaver, un sistema di machine learning che dopo aver appreso informazioni da sequenze di eventi

(riguardanti failures di componenti di telecomunicazione) con features categoriche organizzate a finestre (come nel lavoro di Domeniconi), applicano un algoritmo genetico per identificare i patterns e per costruire delle regole per la classificazione.

Capitolo 3

Analisi delle informazioni raccolte in azienda

Questo capitolo è dedicato allo studio preliminare del contesto informativo aziendale, sia per quanto concerne l'architettura di sistema e le piattaforme, sia dal punto di vista del contenuto informativo dei report generati da ogni singolo componente. Per lo scopo della tesi, sono stati inizialmente forniti i file di log di 2 mesi di attività, del periodo Dicembre-Gennaio 2014, per un totale di circa 3 TB di file grezzi. Successivamente sono stati aggiunti anche i dati di Febbraio e Marzo 2014. Questi files, visto il contenuto ricco di informazioni personali, sono stati concessi dall'azienda InfoCert s.p.a. con l'obbligo di accesso e utilizzo interno all'azienda, almeno fino a quando non è stato effettuato un cleaning dei dati sensibili.

3.1 Posta elettronica e PEC

La posta elettronica è un metodo di scambio di messaggi digitali da un mittente a uno o più destinatari, inizialmente il sistema prevedeva che i soggetti interessati fossero necessariamente online al momento dello scambio, successivamente si è passato al modello store-and-forward implementato nei mail server, i quali hanno la funzione di inviare, ricevere e salvare i messaggi. Originariamente inoltre erano consentiti soltanto messaggi formati da testo ASCII, in seguito sono stati approvati standard, come il *MIME* (Multipurpose Internet Mail Extension), che ha consentito di allegare contenuti multimediali. Anche l'infrastruttura di rete è stata cambiata rispetto alla rete ARPANET abbinata al protocollo *FTP* (File Transfer Protocol), passando al moderno internet con *SMTP* (Simple Mail Transfer Protocol), introdotto alla fine del 1981 anche se sviluppato negli anni '70. Il meccanismo della posta elettronica, esposto in Figura 3.1, si basa su alcune tecnologie e protocolli che vengono di seguito discussi a grandi linee. Le componenti software principali sono:

- **MUA:** *Mail User Agent*, programmi utilizzati dall'utente finale per inviare e leggere i messaggi, sono spesso chiamati mail client e i più diffusi sono Mozilla Thunderbird, Microsoft Outlook, IBM Lotus Notes, Apple Mail, Eudora.

- **MTA:** *Mail Transfer Agent*, software responsabile unicamente dell'invio e ricezione dei messaggi tramite il protocollo SMTP, gli MTA più diffusi sono Exim, Postfix, Sendmail e Qmail.
- **MSA:** *Mail Submission Agent*, programma che riceve i messaggi dal MUA e coopera con MTA per la spedizione, spesso è integrato direttamente nel software MTA, in genere MSA si occupa di accettare email da utenti dello stesso dominio, mentre MTA accetta messaggi provenienti da domini esterni.
- **MDA:** *Mail Delivery Agent*, software responsabile della ricezione di un messaggio da un MTA e del suo salvataggio nella mailbox del destinatario, il quale provederà a leggerla.

Per quanto concerne i protolli principali, il più importante è senza dubbio Simple Mail Transfer Protocol (*SMTP*), il protocollo standard per la trasmissione di e-mail via internet. Durante la fase di invio della posta tramite un client, questo si collega al server SMTP del proprio provider di posta, tramite appunto il protocollo SMTP (porta 25). In ascolto sul server del provider c'è un MTA che, analizzando il dominio del destinatario, verifica a quale server il messaggio va inoltrato. Se i provider di mittente e destinatario sono gli stessi allora si passa direttamente alla fase di consegna, altrimenti l'MTA contatta il server definito nel record DNS del dominio di destinazione (record MX: *mail exchanger*), e inoltrerà il messaggio all'MTA del server responsabile della ricezione del nuovo provider, che ripeterà i passaggi precedenti. Una volta che il messaggio è stato ricevuto dall'MTA del server del destinatario, quest'ultimo lo passa all'MDA, che immagazzina il messaggio in attesa che l'utente lo rilevi. Esistono 2 protocolli principali che permetto di rilevare e fruire della posta:

- **POP3:** *Post Office Protocol*, il più anziano dei due, permette di scaricare il proprio messaggio (lasciando eventualmente una copia sul server) dopo essersi autenticati. Viene utilizzata la porta 110 e la connessione è di tipo TCP, senza cifratura.
- **IMAP:** *Internet Message Access Protocol*, permette la sincronizzazione dello stato dei messaggi (letto, cancellato, spostato) tra più client, con questo protocollo viene comunque conservata una copia dei messaggi sul server per consentire la sincronizzazione, inoltre ha più funzioni di POP3. Sfrutta la porta 143, mentre se si utilizza SSL la porta è la 993.

3.1.1 Posta Elettronica Certificata (PEC)

La Posta Elettronica Certificata (*PEC*) è un tipo speciale di posta elettronica che viene regolamentata direttamente dalla legge italiana. Il suo scopo è quello di simulare una raccomanda con ricevuta di ritorno, compreso il valore legale. Il funzionamento si basa sul rilascio di *ricevute* al mittente, sia al momento della spedizione del messaggio (ricevuta di accettazione), sia in fase di consegna o mancata consegna (ricevuta di presa in carico e di consegna). Queste ricevute hanno valore legale e sono rilasciate dai *gestori PEC*, enti abilitati per la fornitura di tale servizio. L'elenco dei gestori, la vigilanza e il coordinamento degli stessi è affidata all'Agenzia per l'Italia Digitale (AGID), ex DigitPA e CNIPA, tramite il rilascio del Codice dell'Amministrazione Digitale (CAD) [24]. Utilizzando la PEC viene quindi garantito il non-ripudio, inoltre firmando elettronicamente il contenuto del messaggio o criptandolo viene garantita anche l'autenticazione,

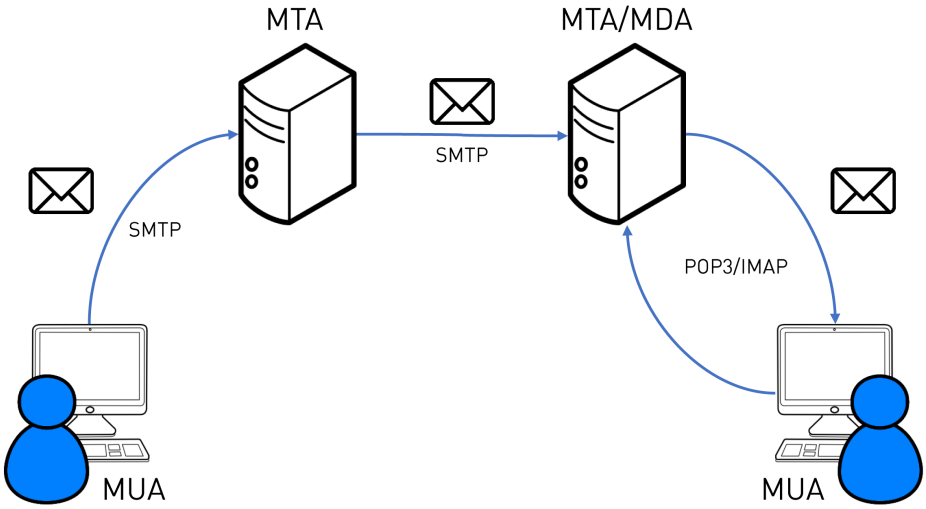

Figura 3.1: Schema base di un'infrastruttura mail.

l'integrità dei dati e la confidenzialità. Ogni gestore ha molti obblighi da rispettare, come ad esempio uno SLA superiore al 99.8% su base quadrimestrale, oppure mantenere il backup dei log dei messaggi per almeno trenta mesi. Nell'Immagine 3.2 viene esposto il funzionamento essenziale della PEC, il mittente prepara un messaggio e lo sottopone al proprio gestore, dopo essersi autenticato con username e password. Il gestore mittente verifica la correttezza formale del messaggio e in caso positivo rilascia una ricevuta di accettazione firmata digitalmente dal gestore stesso per garantire l'integrità del messaggio e degli allegati. Il gestore mittente invia il messaggio al gestore destinatario inserendolo in una busta di trasporto firmata digitalmente per garantire l'inalterabilità del contenuto. Una volta ricevuto il messaggio, il gestore destinatario rilascia una ricevuta di presa in carico al gestore mittente ed effettua dei controlli antivirus sul messaggio prima di consegnarlo alla casella di posta del destinatario. Viene infine rilasciata al mittente una ricevuta di avvenuta consegna.

3.2 Cenni sul sistema informativo aziendale

La Figura 3.3 illustra l'architettura del sistema informativo alla base del servizio PEC, basata per lo più su componenti FOSS ("Free and Open Source Software") affiancati ad altri applicativi realizzati ad-hoc. L'utente ha la possibilità di accedere alla propria casella di posta sia tramite applicazioni MUA (vedi 3.1), sia tramite browser con l'utilizzo di Webmail, il sito web dedicato alla gestione della casella personale. In quest'ultimo caso le azioni eseguite tramite l'interfaccia grafica del sito generano delle richieste che vengono prese in carico dalle istanze del server HTTP Apache e girate agli Application Server (AS) tramite Mod_JK, dopo i dovuti controlli. Sia il client MUA sia l'applicazione web si interfacciano con la piattaforma Courier-IMAP, che implementa i protocolli standard IMAP e POP3. La piattaforma MTA adibita per l'invio e la ricezione della

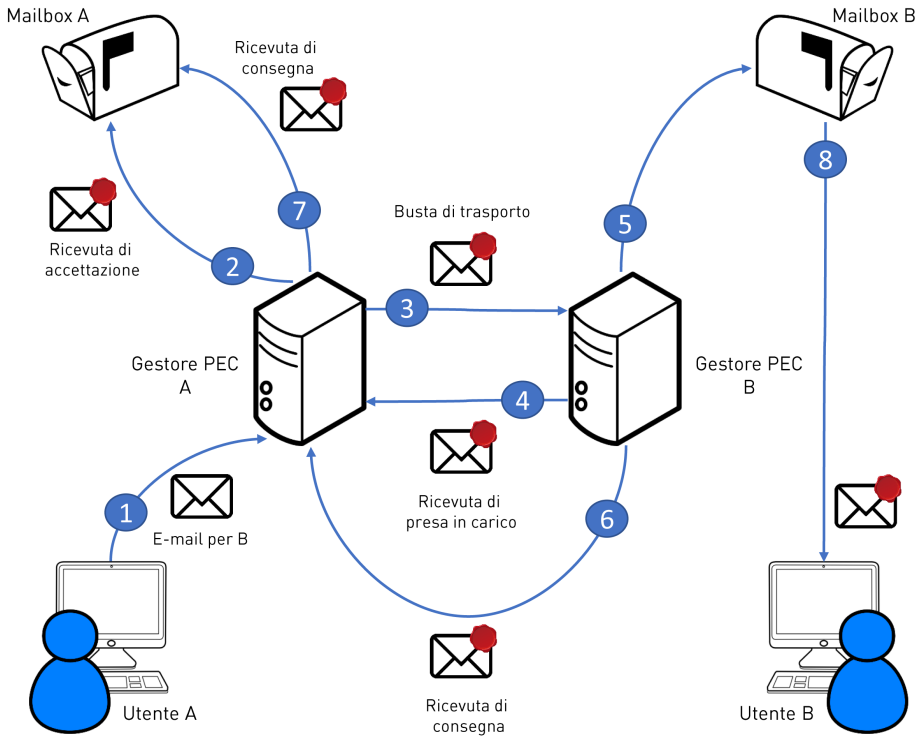

Figura 3.2: Funzionamento della PEC.

posta è Sendmail, la quale si appoggia a Maildrop come componente MDA del sistema. Per mantenere le garanzie e le certificazioni richieste dal modello PEC, ogni messaggio in entrata ed in uscita viene processato dal motore PEC, un'applicazione sviluppata ad-hoc che si occupa tra le altre cose di generare i certificati necessari e di storicizzare gli eventi nei report. Di seguito vengono descritti con più precisione questi applicativi che compongono l'architettura di sistema, in particolare vengono analizzati i formati dei file di log generati e quali informazioni sono ricavabili da essi.

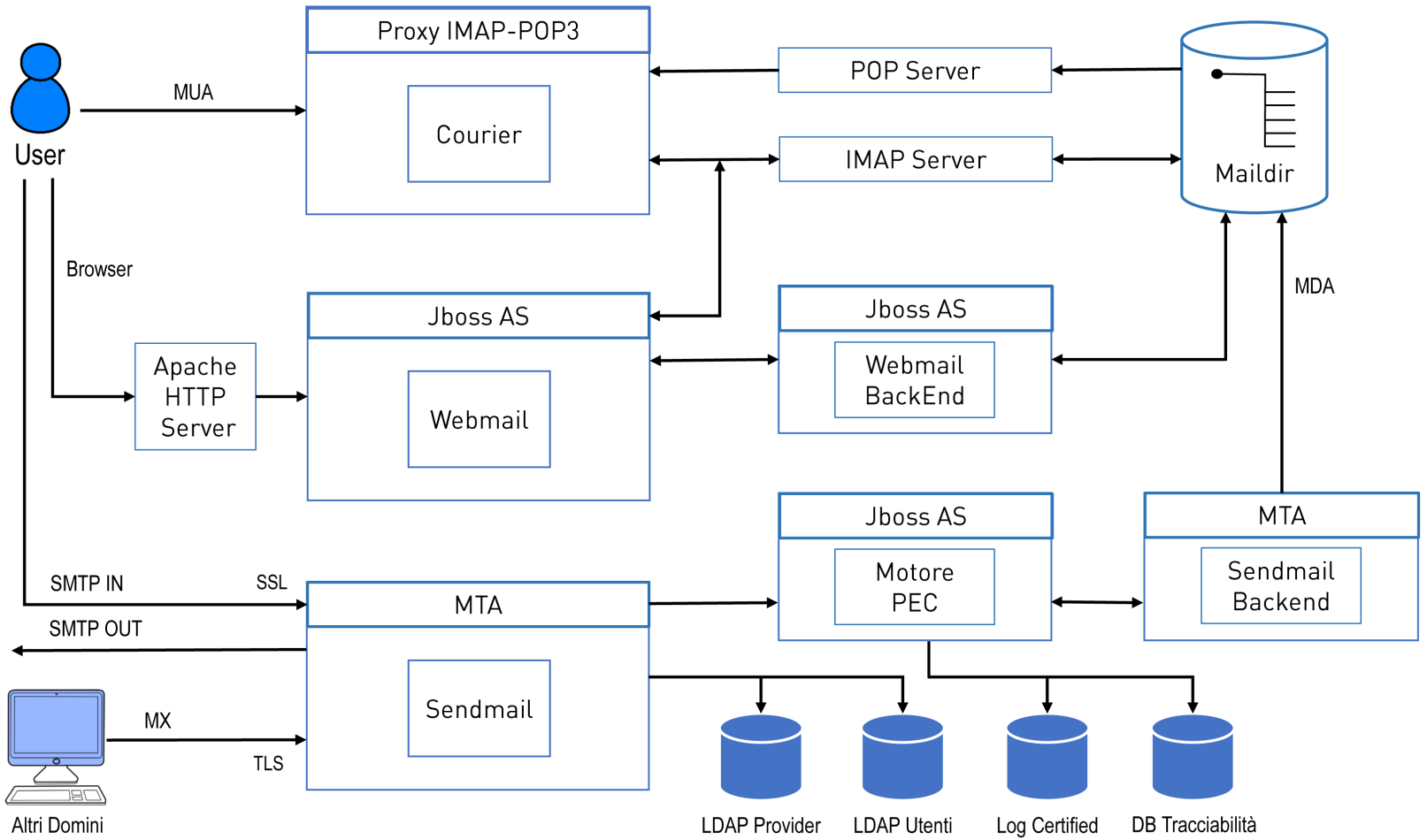

Figura 3.3: Architettura del sistema informativo aziendale.

3.3 Analisi dei software sorgenti di reportistica

Sono di seguito descritte le applicazioni che compongono l'architettura del sistema informativo aziendale presentato in precedenza, facendo particolare attenzione al formato dei file di log prodotti da ciascuno di essi. Le tipologie dei files di log analizzati sono 6: Apache access, Apache error, JBoss AS, Webmail, Sendmail e Courier, alle quali si uniscono i report degli allarmi generati da 2 sonde.

3.3.1 Apache HTTP Server

Piattaforma server web modulare più diffusa al mondo, distribuita con licenza libera Apache License da Apache Software Foundation e compatibile con i principali sistemi operativi Unix/Linux, Windows e OS X. Sviluppata a partire dal 1995, inizialmente era basata sul server HTTPd di NCSA (National Center for Supercomputing Applications), ha giocato un ruolo fondamentale per lo sviluppo a livello mondiale di Internet. La sua principale funzione è quella di soddisfare delle richieste esterne (come pagine html o contenuti di altro tipo come immagini) che sono presenti nel file system o che vengono generati in altri modi, come ad esempio tramite script. La configurazione di Apache si effettua tramite il file *Apache2.conf*, tramite il quale inoltre si può impostare tutto ciò che ruota attorno al logging. I principali file di log gestiti da Apache sono 2: *error* per gli errori e *access* per gli accessi. Con la direttiva *LogLevel* è possibile specificare il livello di verbosity dei messaggi che vengono scritti nei file di log degli errori, i valori possibili sono: debug, info, notice, warn, error, crit, alert, emerg, in ordine di importanza e verbosity crescente. Un'altra direttiva importante è *LogFormat*, tramite la quale è possibile impostare i vari formati dei file di log, i più significativi possono essere:

- **%b** Byte inviati, esclusi gli headers HTTP;
- **%f** Il Filename
- **%VARe** Il contenuto della variabile d?ambiente VAR
- **%h** L'host remoto
- **%k** Numero di richieste keepalive gestite in questa connessione
- **%a** L'indirizzo IP remoto
- **%l** Il logname remoto, se specificato
- **%p** La porta dalla quale il server esegue la richiesta
- **%P** L'ID del processo che esegue la richiesta
- **%r** La prima riga della richiesta
- **%s** Lo stato della richiesta
- **%t** L'orario delle richiesta
- **%T** Il tempo in secondi per eseguire la richiesta
- **%u** Il nome dell'utente remoto
- **%U** L'url richiesta

I file di log Apache access ed error sono separati a seconda dell'istanza Apache e del giorno, il cui inizio però non è sempre esatto alla mezzanotte. Di seguito un esempio di record Apache access, il cui LogFormat si è rivelato essere:

"%h %l %u %t %r %>s %b %{Referer}i %{User-agent}i %k"

```
192.168.1.1 - - [04/Dec/2013:23:59:27 +0100] "GET /immagini/header.png
HTTP/1.1" 302 232 "http://ciao.php" "Mozilla/5.0 (Windows NT 6.1; WOW64;
Trident/7.0; rv:11.0) like Gecko" 0
```

Mentre le righe dei file Apache error sono simili alla seguente:

```
[Thu Dec 05 07:05:22 2013] [error] [client 192.168.1.1]
File does not exist: /opt/index/immagine.png
```

3.3.2 JBoss AS (WildFly)

Gli Application Server sono i "contenitori" di applicazioni con la funzione primaria di eseguire in maniera efficace delle procedure (programmi, routines, scripts) per supportare le relative applicazioni. Ad esempio nel caso di AS Java, per le applicazioni in esecuzione il server si comporta come una Virtual Machine estesa, gestendo sia le connessioni al database, sia le connessioni al web client. In particolare, JBoss è un'application server open source con licenza GNU che implementa l'intera suite di servizi Java EE (Enterprise Edition). Originariamente è stato sviluppato dalla "JBoss inc." di Marc Fleury, la quale nell'aprile del 2006 è stata acquisita da Red Hat per 420 milioni di dollari. Il software è giunto alla versione 8.0 ed è continuamente supportato da una grande rete di sviluppatori. Per quanto riguarda il logging, i file prodotti sono molto simili a quelli di Apache, con qualche differenza nel formato. Una differenza sostanziale riguarda però il periodo temporale coperto da un singolo file, che in questo caso non è fisso. Viene infatti salvato e compresso un vecchio log e creato uno nuovo per ogni riavvio delle istanze dell' AS. Per lo scopo della tesi sono stati forniti i file access di JBoss, con il seguente formato:

"%h %l %u %t %r %>s %b"

Un esempio di riga con questo formato può essere:

```
192.168.1.1 - - [05/Dec/2013:06:05:25 +0100] "GET /immagini/img.gif
HTTP/1.1" 404 1018
```

3.3.3 Webmail Legalmail

Applicazione sviluppata ad-hoc dall'azienda, è una web application per la gestione della posta da parte degli utenti. È dotata sia di una parte di front-end, ovvero l'interfaccia fruibile online, sia di una parte di back-end. Nella reportistica prodotta da questa applicazione si possono trovare informazioni su ciascuna azione compiuta dagli utenti, come i login o logout, l'invio o la cancellazione dei messaggi, la creazione di cartelle, ecc... I log sono separati a seconda dell'istanza dell'AS e del giorno, questa volta in maniera netta, dalla mezzanotte del giorno considerato. A seguire un esempio di log webmail.

```
2013-12-05 00:01:46,515; userId = XXX; ip = 192.168.1.1;
invio messaggio: id = abc123
```

3.3.4 Sendmail

Sendmail è un *Mail Transfer Agent* (MTA) free e open source che supporta diversi metodi di trasferimento e consegna di mail, inclusi SMTP, ESMTP, Mail-11, HylaFax, QuickPage e UUCP. Deriva da *delivermail* di ARPANET sviluppato da Eric Allman, il quale ha iniziato a lavorare su Sendmail nei primi anni '80. Nel 2001 il 42% dei server mail pubblici eseguivano Sendmail, la cui diffusione ha successivamente subito

un declino, fino a raggiungere il 12,4% nel 2012. In ogni caso resta uno dei principali mail server assieme a Microsoft Exchange Server, Exim e Postfix. Il 1 Ottobre 2013 Sendmail è stato acquisito da Proofpoint Inc. per 23 milioni di dollari, è stato comunque garantito il continuo aggiornamento della versione open source.

Per quanto concerne il logging, su Sendmail è gestito dal demone *syslogd*, ogni linea del file di log è costituita da un timestamp, dal nome della macchina che lo ha generato, dalla parola "sendmail:" e da un messaggio che è una concatenazione di *nome=valore* dei diversi parametri. Ci sono 2 righe fondamentali presenti nei log, la prima è creata alla presa in carico di un messaggio per la consegna (rappresenta quindi la busta, o *envelope*) ed è formata dai seguenti campi:

- **from:** The envelope sender address
- **size:** le dimensioni del messaggio in bytes
- **class:** la classe del messaggio (precedenza)
- **pri:** la priorità iniziale (usata per l'ordinamento)
- **nrcpts:** numero dei destinatari
- **msgid:** id del messaggio
- **bodytype:** tipologi del body (7BIT or 8BITMIME)
- **proto:** protocollo usato per ricevere il messaggio (es., ESMTP o UUCP)
- **daemon:** nome del demone
- **relay:** la macchina dalla quale è stato ricevuto

Un esempio può essere:

```
Jan 15 04:16:26 istanza6 sendmail-frontend[12345]: XXXsessionIDXXX:
from=<davide@zannirato.it>, size=3871, class=0, nrcpts=0, bodytype=8BITMIME,
proto=ESMTP, daemon=MTA, relay=host.it [123.45.678.9]
```

il secondo tipo di riga viene creata ad ogni tentativo di consegna (quindi ne possono esistere molteplici per messaggio, se quest'ultimo ha più destinatari o se fallisce il tentativo), i campi che la compongono sono i seguenti:

- **to:** destinatari separati da virgola
- **ctladdr:** il nome dell'utente le cui credenziali sono utilizzate per la spedizione
- **delay:** il ritardo tra il tempo di ricezione del messaggio e il tempo del tentativo di spedizione corrente
- **xdelay:** quantità di tempo necessario nel tentativo di spedizione corrente (indicativo per la velocità della connessione)
- **mailer:** il nome del mailer usato per la consegna al destinatario
- **relay:** nome dell'host che ha accettato/rifiutato la consegna
- **dsn:** l'enhanced error code (RFC 2034) se disponibile
- **stat:** The delivery status.

Un esempio di riga di questo tipo presente nei file di log può essere:

```
Jan 15 04:16:26 istanza3 sendmail-relay[12345]: XXXsessionIDXXX:
to=davide@zannirato.it, delay=00:00:03, xdelay=00:00:03, mailer=relay,
pri=131261, relay=mail.davide.it. [123.45.67.89], dsn=2.0.0,
stat=Sent (Ok: queued as XXXXXX)
```

Questo tipo di reportistica rappresenta ovviamente la parte più corposa del materiale fornito, infatti un file giornaliero decompresso può superare anche i 22 GB di spazio su disco. Al contrario delle altre piattaforme, la collezione dei log di sendmail dalle varie istanze del programma è stata centralizzata, quindi ogni giorno (a partire però dalle 4 a.m.) viene prodotto un singolo file di report, i cui record indicano tra le varie informazioni presentate prima anche l'indicazione dell'istanza di provenienza.

3.3.5 Courier-IMAP

Courier è un server Mail Transfer Agent (MTA) sviluppato da Sam Varshavchik e con licenza GPL, che fornisce servizi di ESMTP, IMAP, POP3, SMAP, webmail, e mailing list. Esistono in realtà due versioni di Courier che possono essere utilizzate: la prima è la suite, che comprende oltre al server IMAP anche un MTA, un server POP3, MailDrop (l'MDA) e una webmail, mentre l'altra, come nel caso in esame, è il singolo Courier-IMAP, il server IMAP che offre anche supporto al protocollo POP3. Per lo storage della posta viene utilizzato il formato *maildir*, che prevede un'organizzazione dei messaggi in file separati con nomi unici e in cartelle directory e automatizza il meccanismo di lock/unlock della risorsa. Anche su Courier-IMAP i log vengono inviati tramite il demone syslogd e allo stesso modo di Sendmail la raccolta è centralizzata, mantenendo l'informazione riguardante l'istanza che ha generato il record in esame. I log contengono dati riguardanti le connessioni, gli accessi degli utenti e il protocollo utilizzato. A seguire un esempio:

```
Jan 15 04:05:33 istanza1 imapd-ssl: PROXY LOGIN, user=davide@zannirato.it,
ip=[::ffff:12.34.567.89], port=[1234]
```

3.3.6 Sonde Virtual Users

Per l'individuazione delle anomalie riscontrate nei mesi relativi ai file di log in esame, sono stati forniti 2 ulteriori report, riguardanti dei controlli sullo stato del servizio di posta effettuati da 2 sonde chiamate "Virtual Users". I test sono eseguiti con cadenza di circa 5 minuti da ciascuna sonda. Come suggerisce il nome, le sonde simulano il comportamento di utenti virtuali e delle azioni che possono compiere, ritornando un "+1" se l'azione va a buon fine o "-1" in caso contrario, a seguire un esempio di linea dei report:

```
ST LEGALMAIL - Componente PEC,28.02.2014,23:02:31,1.000000
```

La prima sonda è chiamata *VU-Webmail*, la quale effettua diverse operazioni agendo direttamente dal sito della posta, come inviare un messaggio, eliminare la posta, ecc... Se almeno una di queste operazioni non ha successo, viene indicata un'anomalia di sistema. Questa sonda è piuttosto delicata e non sempre è precisa, può presentare infatti dei falsi allarmi, dovuti ad esempio ad un errore di connessione, errore del sistema operativo e via dicendo. La seconda sonda è chiamata *VU-Pec*, la quale tramite script inoltra direttamente al motore PEC l'invio di un messaggio di posta elettronica certificata e attende l'arrivo della ricevuta, andando a testare quindi il solo invio e ricezione della posta. Al contrario della prima sonda, la operazioni sono inoltrate direttamente alla parte di back-end del sistema, in particolare al motore PEC, senza passare per la parte di front-end. Per questo motivo i valori presenti in questo report sono più attendibili rispetto agli altri, tesi sostenuta anche dal confronto del numero totale di anomalie

presenti nei 2 files, 619 per VU-Pec contro i 1356 di VU-Webmail per i primi 2 mesi di file di log forniti. Analizzando in dettaglio i report Virtual Users si nota che gli allarmi spesso sono presenti in raffiche, ovvero occorrono consecutivamente sia per brevi lassi temporali (10 - 15 minuti) sia per intere giornate. Siccome lo scopo finale del lavoro di Tesi è quello di prevedere questi allarmi, è ragionevole attribuire al primo allarme di queste raffiche un livello di importanza superiore, questa sarà la motivazione alla base di alcuni step di preprocessing dei dati, discussi successivamente.

Capitolo 4

Progettazione e sviluppo di un prototipo

In questo capitolo vengono descritte le fasi della creazione del prototipo e i risultati dei vari test effettuati. Inizialmente viene presentata a grandi linee la strategia operativa alla base della Tesi, ovvero come costruire il dataset con i record per effettuare il training di un modello di classificazione, successivamente seguendo gli step dell'approccio KDD descritto nel Capitolo 2.2 si descrivono le trasformazioni apportate alle informazioni estratte, dal formato grezzo al dataset definitivo. Vengono poi analizzate e testate diverse strategie di classificazione, confrontando i risultati per esplicitare il metodo più performante.

4.1 Strategia di attacco al problema

Nel Paragrafo 2.5.1 sono state definite le *failures* di sistema come obiettivo di previsione. Nel contesto della presente Tesi viene fatto riferimento ai failures anche come *allarmi*, essendo infatti i failures rilevati da sonde. Dopo aver analizzato le informazioni a disposizione, lo step successivo è quello di ricavare un dataset di record e features in grado di addestrare correttamente un modello di classificazione. L'idea alla base dell'attività di preprocessing è stata quella di considerare una giornata di 24 ore di attività suddivisa in intervalli regolari di tempo, chiamati *timeslot*. A ciascuno di questi intervalli si è pensato di poter assegnare una serie di informazioni e attributi ricavati adeguatamente dai file di log, riassumendo una serie di dati riguardanti ciascun intervallo, e di poter associare un'etichetta per indicare se in quel preciso slot temporale si è verificata o meno un'anomalia o failure. Il dataset risultante ha quindi la struttura presentata in Tabella 4.1.

	FEATURE 1	**FEATURE 2**	**...**	**FEATURE n**	**CLASS**
timeslot 1	$t_1(A_1)$	$t_1(A_2)$	...	$t_1(A_n)$	{+1,-1}
...	...	...	...	...	...
timeslot m	$t_m(A_1)$	$t_m(A_2)$	...	$t_m(A_n)$	{+1,-1}

Tabella 4.1: Struttura del Dataset di timeslot.

Tenendo conto che gli esiti dei controlli degli allarmi sono notificati nei report Virtual Users con una cadenza di 5 minuti, si è deciso di utilizzare come unità di misura temporale timeslot di 5 minuti. In questo modo ogni giornata viene suddivisa in 288 timeslot, ciascuno identificato univocamente dal numero dello slot, a partire da 1. I timeslot si possono quindi definire come osservazioni sull'andamento del sistema e sullo stato dei servizi aziendali riguardante un periodo di attività di 5 minuti. Se l'attributo di classe "alarm" è settato a "+1" significa che in quei 5 minuti si è effettivamente verificato un allarme sui report Virtual Users. Un esempio del dataset risultante al termine del processing dei report è presentato nella Tabella 4.2.

timeslot	**dayOfWeek**	**... altre features ...**	**alarm**
21	2	A	-1
22	2	B	-1
23	2	C	-1
24	2	D	-1
25	2	E	-1
26	2	F	-1
27	2	G	-1
28	2	H	+1
29	2	I	+1
30	2	L	-1
31	2	M	-1

Tabella 4.2: Esempio del dataset con record formati da timeslot di 5 minuti.

Il dataset risultante però manca della proprietà di predizione degli allarmi, un classificatore addestrato su questi dati infatti avrebbe la capacità di assegnare la classe di allarme soltanto quando quest'ultimo è già in atto, occorre quindi integrare in qualche modo questo aspetto. Seguendo il lavoro di Carlotta Domeniconi et al.[22], l'idea è stata quella di ri-modulare questi slot temporali in finestre chiamate *monitor-windows* di uno o più timeslot, effettuando un merge dei relativi attributi. Alle finestre viene assegnato un ruolo predittivo, etichettandole con valore +1 o -1 a seconda se entro un certo intervallo temporale, chiamato *warning-window*, si è verificato un allarme. Infatti l'obiettivo del lavoro è quello di segnalare un'anomalia prima del suo verificarsi per predisporre le necessarie contromisure, non quando è già in corso. Con il termine *monitor-windows* (MW) si intende un insieme di uno o più timeslot temporalmente successivi, con i valori delle relative features numeriche sommati. Per i valori categorici come il numero di timeslot, viene assegnato alla MW il valore del primo timeslot che compone la finestra. La *warning-window* (WW) rappresenta la finestra temporale successiva ad ogni MW e anch'essa dura uno o più timeslot. Rappresenta il tempo necessario per la predisposizione delle contromisure in vista di un allarme, maggiore è la WW più preavviso è concesso. La sua utilità è quella di assegnare un'etichetta di classe alla MW in base agli allarmi/non-allarmi futuri. Più precisamente vengono distinte due strategie di valutazione:

Partitioning-Variable-Distance (PVD): MW costruite su timeslot temporalmente successivi sommando i valori delle features numeriche. Le MW sono disgiunte tra loro, cioè non condividono le informazioni di nessun timeslot. Non appena termina il calcolo della MW, la lettura dei timeslot continua a seconda della dimensione della WW, se tra questi timeslot è presente almeno un allarme allora si

assegna un flag +1 (allarme) alla MW, altrimenti −1 e viene scritta in output. Successivamente si saltano completamente i timeslot inclusi nella MW, prendendo in considerazione i successivi. Nella Tabella 4.3 e nell'Immagine 4.1 è disponibile un esempio derivato dall'applicazione dell'algoritmo ai record della Tabella 4.2.

timeslot	dayOfWeek	... features ...	alarm
21	2	A+B+C	-1
24	2	D+E+F	+1
27	2	G+H+I	?

Tabella 4.3: Esempio di dataset con MW = WW = 3 timeslot costruite con l'algoritmo Partitioning-Variable-Distance.

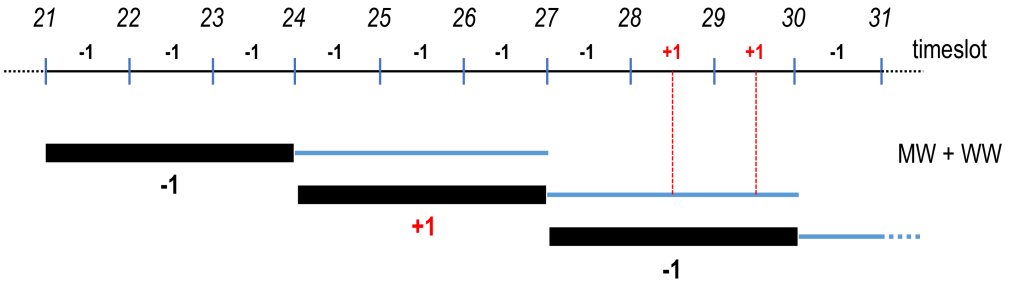

Figura 4.1: Processo di costruzione di MW = WW = 3 timeslot con l'algoritmo Partitioning-Variable-Distance.

Scrolling-Fixed-Distance (SFD): MW costruite su timeslot temporalmente successivi sommando i valori delle features numeriche. Le MW non sono disgiunte tra loro, perché costruite scorrendo il dataset di timeslot in timeslot. Al termine del calcolo di una MW viene controllato se *esattamente* dopo |WW| timeslot si ha un allarme, in questo caso viene assegnato il valore +1 alla MW, altrimenti −1 e viene scritta. Al termine l'intervallo ricoperto dalla MW salta in avanti di un timeslot, reiterando il processo. Nella Tabella 4.4 e nell'Immagine 4.2 è disponibile un esempio ottenuto applicando l'algoritmo ai record della Tabella 4.2.

La differenza tra le due metodologie è chiara, con il metodo Partitioning-Variable-Distance le MW non condividono dati, i timeslot vengono partizionati e il dataset risulta più leggero. Il verificarsi di un allarme però viene garantito entro un range, la dimensione della WW, ovvero quest'ultima identifica il periodo massimo entro il quale si verificherà un allarme. Con l'algoritmo Scrolling-Fixed-Distance il numero di MW è maggiore, poiché catturano in maniera più dettagliata l'andamento delle features. Inoltre se viene segnalato un allarme si ha la certezza della sua occorrenza dopo |WW| timeslot, poiché la WW rappresenta il periodo esatto dopo il quale ci sarà allarme.
In questo modo, variando i parametri dimensionali di MW e di WW e scegliendo una strategia di generazione delle finestre si possono ottenere dataset pronti per un task

timeslot	dayOfWeek	... features ...	alarm
21	2	A+B+C	-1
22	2	B+C+D	+1
23	2	C+D+E	+1
24	2	D+E+F	-1
25	2	E+F+G	-1
26	2	F+G+H	-1
27	2	G+H+I	-1
28	2	H+I+L	-1
29	2	I+L+M	-1
30	2	L+M+N	?

Tabella 4.4: Esempio di dataset con MW = WW = 3 costruite con l'algoritmo Scrolling-Fixed-Distance.

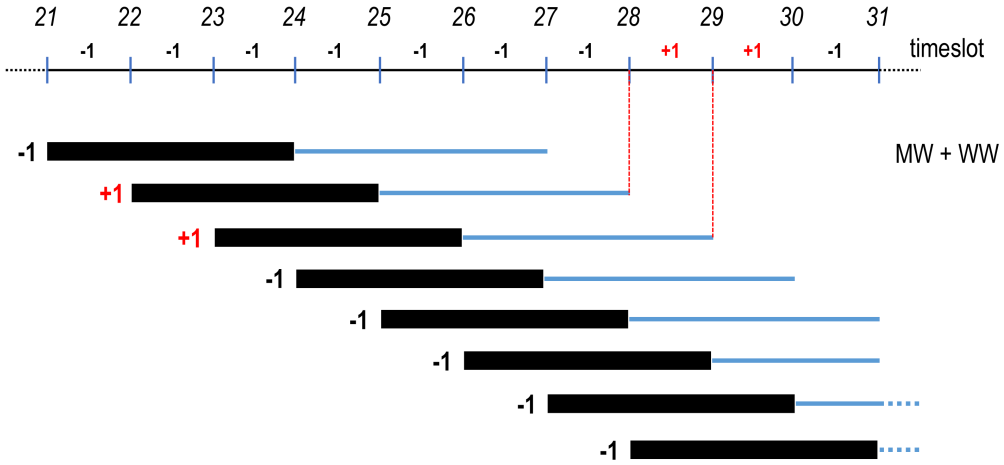

Figura 4.2: Processo di costruzione di MW = WW = 3 timeslot con l'algoritmo Scrolling-Fixed-Distance.

di classificazione su etichette allarme/non-allarme, che saranno in grado di allertare l'approssimarsi di un'anomalia ogniqualvolta sarà assegnata una classe positiva ad una monitor-window. I test futuri serviranno per stabilire la strategia migliore e i parametri ottimali per garantire le migliori prestazioni e accuratezza. Prima però occorre identificare quali dati da ciascun file di log sono utili per questo scopo, in modo da definire con precisione gli attributi dei timeslot che andranno a comporre il dataset e sviluppare i tools necessari per l'estrazione ed elaborazione automatica di questi valori.

4.2 Preprocessing dei log

Di seguito sono delineate le fasi di preprocessing applicate agli archivi dei file di log grezzi, fino ad ottenere il dataset formattato pronto per il task di classificazione. Tutte le classi e le applicazioni sono state sviluppate in Java, organizzando il preprocessing nel modo più modulare possibile, per consentire modifiche veloci in corso d'opera o future. Una visione sommaria delle elaborazioni alle quali sono stati sottoposti i files è visibile dall'Immagine 4.3. La fase di *Selection/Extraction* si occupa dell'individuazione ed estrazione dal file system degli archivi di log necessari, a seconda dell'applicazione che li ha generati e del periodo temporale richiesto. Nella fase di *Cleaning* i dati sono stati ripuliti dalle informazioni non necessarie, compresi i dati sensibili come username degli utenti o indirizzi IP, ri-formattando ciascun report secondo regole prestabilite e uniformi, con campi separati dallo stesso placeholder e con lo stesso formato del timestamp. Infine su *Transformation and data reduction* sono state specificate le features ricavabili dai dati, in modo da riassumere e ridurre queste informazioni per andare a formare record con features riassuntive di 5 minuti di attività del sistema, senza distinzioni di istanze. Ogni record/timeslot avrà inoltre un'etichetta per indicare se nei 5 minuti ai quali fa riferimento ci sono stati allarmi segnalati dalle sonde. Viene poi descritto l'algoritmo per la creazione delle finestre monitor-windows con informazione predittiva a partire dai timeslot e dai parametri in input, con relative politiche di cleaning.

4.2.1 Selection and Extraction

I files di log messi a disposizione sono delle 6 tipologie presentate nel Capitolo 3.3, ciascuno con informazioni diverse, più o meno relazionati tra loro e sono: Apache access, Apache error, JBoss AS, Webmail, Sendmail e Courier, oltre a 2 aggiuntivi report riguardanti gli allarmi rilevati dalle sonde Virtual-Users. Gli archivi compressi dei report sono stati resi disponibili in un hard disk, con le directory strutturate a seconda della macchina e dell'istanza sulla quale è presente la relativa componente, tranne per i log Sendmail e Courier che come già detto sono raccolti in modo centralizzato, senza frammentazioni dovute all'istanza. Ogni archivio log presenta una denominazione con informazioni relative al nome della componente, all'istanza e al giorno riguardante il log. L'estrazione degli archivi di interesse si è quindi basata su un'analisi del relativo nome e della path del file, predisponendo un programma in Java atto a questo scopo, che riceve come parametro di input la data di inizio del periodo temporale desiderato, in modo da poter ricavare solo i report necessari. È stata necessaria una particolare estrazione della reportistica riguardante le varie istanze dell'application server, infatti i file non sono archiviati giornalmente, ma ad ogni suo riavvio. Sono mantenuti gli ultimi 15 log per ogni istanza dell'AS, presentando denominazioni generiche da 1 a 15. Quando si presenta un riavvio, viene eliminato il report più datato e creato il nuovo file per la scrittura dei log. Per lo scopo della tesi sono state effettuati dei backup dei 15 files di ogni istanza a cadenza temporale fissa, rendendo possibile la presenza di eventuali doppioni di eventi notificati nei log. Per esplicitare quindi i file di log netti di ogni istanza dell'AS è stato dunque effettuato un preprocessing ulteriore. Questa elaborazione consiste in un controllo dei timestamp del primo e dell'ultimo evento presente nei 15 archivi di ciascuna istanza in ogni backup presente.

4.2.2 Cleaning

In questa fase sono state definite 3 classi distinte, la prima `FieldsExtractor` atta all'estrazione dei campi disponibili in ogni entry di ciascun tipo di file di log, che viene sfruttata dalla seconda classe `Cleaner` che riscrive ciascun report mantenendo soltanto le informazioni utili strutturate in un formato condiviso e infine `DailyReorganiser` che smista le entry ripulite di ogni files in files relativi a ciascuna data presente, in modo da facilitare la successiva separazione in timeslot.

FieldsExtractor

Questa classe riceve come parametri una path di file di log e un intero per indicarne la tipologia e consente tramite il metodo getNext() di ricevere un array di stringhe, ciascuna corrispondente ad un determinato campo della riga in esame. Invocando via via il metodo è possibile effettuare il parsing di ciascuna riga dell'intero report. Da un'analisi della struttura delle entry dei vari tipi di file, è risultato che gli unici file di log con i campi divisi da un separatore preciso sono quelli relativi a Webmail, i quali sono quindi facilmente processabili tramite la nota classe `StringTokenizer` di Java. Per i restanti log, si è reso necessario l'utilizzo delle espressioni regolari, effettuando diversi test per stabilire il corretto funzionamento delle 5 regex costruite, apportando le dovute correzioni. Le classi di `java.util.regex`, in particolare `Pattern` e `Matcher`, sono state impiegate per questo scopo. I primi test sui report più pesanti, ovvero quelli di Sendmail che superano anche i 20 GB quando decompressi, hanno però evidenziato tempi di elaborazione molto elevati, vicini ai 45 minuti. Una successiva ottimizzazione dei sorgenti (specie invocando il metodo *reset* di ciascun oggetto Matcher piuttosto che crearne uno nuovo) e sopprattutto delle regex (tramite l'utilizzo dei *Possessive Quantifiers* [25]) hanno fatto abbassare i tempi a circa 10 minuti nel caso peggiore. A seguire un esempio di Pattern di regex per l'estrazione dei campi di un log Apache error:

```
"\\[(\\w{3}+\\s\\w{3}+\\s\\w{2}+\\s[\\d:]++\\s\\d{4}+)\\]
              \\[(\\w++)\\] ([^§]++)$"
```

E di un log JBoss AS:

```
"^([\d.]++) (\\S++) (\\S++) \\[([\\w:/]++\\s[+\\-]\\d{4}+)\\]
            \"(.+?)\" (\\d{3}+) ([-\\d]+)\$"
```

Cleaner

Come `FieldsExtractor`, anche la classe `Cleaner` riceve in input una path di file di log e un intero e a seconda di questi parametri viene effettuata una pulizia del report, creando un nuovo file molto meno verboso e ben strutturato. Innanzitutto è stato stabilito un formato univoco per i timestamp, ovvero "yyyy-MM-dd HH:mm:ss" seguendo gli standard di denominazioni usati nella classe Java `SimpleDateFormat`, e un separatore di campo ",". A seconda della tipologia del file di log in input, viene effettuata una scrematura dei campi presenti, mantenendo le sole informazioni utili e formattate secondo gli standard prestabiliti. Oltre che all'eliminazione dell'informazione non utile, in questa fase avviene anche la pulizia di tutte le informazioni sensibili presenti, come indirizzi IP, username, oggetti dei messaggi, id dei messaggi, percorsi nel filesystem aziendale delle risorse richieste ecc ... . Ovviamente qualsiasi riga di qualsiasi file di log è dotata di timestamp, oltre a questa per quanto riguarda i log Apache access si sono mantenute informazioni sul tipo di richiesta (GET, POST, DELETE, ...), il relativo HTTP status

code, i bytes dell'oggetto richiesto e il numero di keepAlive. Per Apache error è stata conservata la sola descrizione dell'errore mentre per i log JBoss le informazioni sono le stesse di Apache access tranne per il numero di keepAlive. Inoltre, essendo i log JBoss salvati ad ogni reboot della piattaforma, è stata aggiunta una riga per indicare l'avvenuto riavvio, ritenendo quest'informazione interessante. I report Webmail sono quelli con più informazioni sensibili, è stata mantenuta la sola azione effettuata dall'utente, come ad esempio l'eliminazione di un messaggio o la creazione di una cartella. Sui log Sendmail è stata apportata la pulizia più massiccia, infatti sono state conservate solo 4 tipologie di righe presenti. La prima è l'envelope per la spedizione di un messaggio, distinta dalla parola chiave "from=", di questa linea vengono salvati sia le dimensioni in bytes sia la componente di Sendmail che ha generato il log (frontend, relay, mx). La seconda tipologia attiene ai tentativi di invio ed è dotata del campo "to=". In queste righe vengono mantenute le informazioni sul tempo di ritardo del tentativo di invio rispetto alla ricezione della richiesta, lo stato dell'invio e come prima la componente di Sendmail. Un altro tipo di linea ritenuta significativa è quella contrassegnata dal messaggio "done", che indica il corretto invio del messaggio, da questo log sono salvate le informazioni sul delay totale, il numero di tentativi e la componente Sendmail. Infine la quarta tipologia di righe salvate riguarda i log dell'antivirus di sistema, dalle quali vengono mantenute informazioni sui virus trovati. Per quanto concerne i report Courier, viene salvato il tipo di protocollo e il tipo di azione o errore segnalato nella riga. Anche i report Virtual Users degli allarmi sono stati ripuliti dalle informazioni non necessarie, riducendo ciascuna riga in un timestamp e in un valore binario di allarme o non allarme. Un trattamento particolare concordato con i responsabili aziendali è stato riservato ai report allarmi delle sonde VU-Webmail, siccome ritenuti meno veritieri sono stati eliminati i singoli allarmi isolati, ovvero adiacenti a periodi temporali di corretto funzionamento.

DailyReorganiser

Una tipologia di file di log, Webmail, risulta già struttuta in file giornalieri, che iniziano alla mezzanotte e raccolgono informazioni fino alle 23:59:59. Tutti gli altri file non seguono queste regole, ad esempio i report Apache, Sendmail e Courier sono creati giornalmente ma ad orari diversi, ricoprendo quindi porzioni temporali di due giorni. Altri come JBoss AS vengono addirittura creati ad ogni riavvio dell'application server. Per uniformare questi file e facilitare le elaborazioni e le prove future, si è ritenuto ragionevole ri-strutturare i log, in modo che un singolo file contenga lo storico degli eventi di una singola giornata intera, a partire dalle 00:00:00 fino alle 23:59:59. Per questo proposito è stata sviluppata una classe `DailyReorganiser` che, presi in input un array di file di log della medesima sorgente software e un intero per indicarne la tipologia, effettua uno smistamento delle entry in nuovi file giornalieri. Visto che l'elaborazione coinvolge i timestamp delle righe, sono stati invocati i metodi della classe Java `Calendar` e `Date`.

4.2.3 Transformation and data reduction

A questo punto i file di log risultano molto alleggeriti poiché sono state eliminate le informazioni non necessarie e strutturati in file giornalieri. Sono comunque ancora separati a seconda della sorgente software di provenienza e dell'istanza. In questa fase l'obiettivo è quello di definire le features da ricavare da ciascun report e di elaborare

questi file di log ancora frammentati fino ad ottenere un dataset unico, strutturato in timeslot della durata di 5 minuti, ciascuno con i valori delle features delle varie categorie di log adiacenti. L'ultimo step è quello di sviluppare un applicativo che a partire da questo dataset di timeslot generi un nuovo dataset di monitor-windows e warning-windows, applicando alcune politiche di cernita dei record ritenute opportune.

Definizione Features

Analizzando i dati disponibili estratti dal cleaning dei log e ragionando con i sistemisti aziendali, sono stati delineati alcuni scenari che possono aver causato i sovraccarichi o le anomalie di sistema. Sulla base di questi ragionamenti e non solo sono state definite 65 features basate sui dati estratti dai log, delle quali 3 categoriche e il resto numeriche. I 3 attributi categorici sono l'attributo classe binaria $+1/-1$ che indica o meno la presenza di allarme nel relativo timeslot, il numero di timeslot da 1 a 288 e il numero del giorno della settimana da 1 a 7 (per l'elenco completo delle 65 features consultare la Tabella A.1).

Merger

`Merger` è la classe atta alla costruzione dei timeslot e dei relativi attributi a partire dai log giornalieri. A seconda del tipo di log, vengono processate tutte le righe del file riassumendole, ovvero incrementando contatori relativi agli attributi numerici di ciascuna tipologia di file di log. In questa fase decade infatti la separazione tra le varie istanze. Non appena terminano i 5 minuti di uno slot, viene generata una riga di features con i valori dei contatori e salvata in un nuovo file, preceduta da un timestamp fittizio, cioè l'inizio del timeslot, che servirà successivamente per effettuare il match di timestamp tra i diversi file di log per unirli correttamente. Vista la presenza di 2 diverse tipologie di report per gli allarmi, VU-PEC e VU-Webmail, viene effettuato anche il merge di questi file, in modo da avere un singolo report, con i timeslot settati ad allarme se si ha un riscontro in questo senso da almeno una delle due sonde. Al termine di questa fase saranno presenti 7 files, i primi sei riguardanti timeslot e features delle sei tipologie di log a disposizione e l'ultimo relativo agli allarmi.

Matcher

Lo step successivo riguarda l'unione di questi 7 files per creare un dataset univoco, formato da timestamp con attributi ricavati da tutti i log e dal relativo flag di allarme. Nel dataset risultante vengono però scritti i soli record che risultano completi di tutte le features, infatti può succedere che, per motivi come manutenzione al sistema o salvataggio/backup dei log, siano mancanti informazioni su certi timeslot, che però causerebbero dei sbilanciamenti sui valori. La classe `Matcher` si occupa di quanto appena descritto, ricevendo in input un array dei sei file di timeslot riguardanti i log applicativi e il file degli allarmi, utilizza quest'ultimo come "master", prendendo i timestamp dei timeslot in esso contenuti e andando ad effettuare il match con i restanti sei file. Se anche un solo match fallisce il timeslot viene tralasciato e si passa al successivo. È interessante notare come il dataset risultante su quattro mesi di attività, occupi uno spazio di poco più di 6 MB, esattamente lo 0.0001% dello spazio occupato dall'insieme di tutti i log grezzi decompressi.

WindowsGeneration

Per la definizione delle monitor-windows (MW) e warning-windows (WW) è stata sviluppata la classe `WindowsGeneration`, che riceve in input la dimensione espressa in timeslot di MW e WW, il dataset dei timeslot e un file per la scrittura in output del nuovo dataset. Sono stati predisposti due diversi metodi a seconda della strategia di creazione delle monitor-windows e dell'interpretazione della warning-window per l'assegnazione dell'etichetta di allarme, ovvero i metodi Partitioning-Variable-Distance ed Scrolling-Fixed-Distance discussi nel Paragrafo 4.1.

Ai fini del task di classificazione ed in particolare di training del modello, sono stati inoltre predisposti dei metodi atti alla selezione e cleaning dei record/MW presenti in ciascun dataset generato in base ai parametri dimensionali di MW e WW. Il primo metodo `discardWindows` riceve tra i vari parametri di input una classe di riferimento, una classe da scartare, un pre-intervallo e un post-intervallo. Lo scopo del metodo è quello di eliminare i record etichettati con la classe da scartare, che risiedono entro un pre-intervallo o un post-intervallo rispetto ad un record dotato di classe di riferimento. Il metodo quindi è utile nel caso si desideri ad esempio eliminare i record di non-allarme che sono temporalmente adiacenti ad un allarme. Il secondo metodo `discardSubsequentWindows` riceve come parametro di input un valore di classe e utilizza questa informazione per effettuare l'eliminazione di tutti i record etichettati con questa classe che occorrono successivamente nel tempo, mantenendo solo il primo della serie. Questo può tornare utile quando ad esempio si desiderano eliminare le raffiche di record di allarme, mantenendo soltanto il primo nonché più significativo.

4.3 Data Mining

In questo paragrafo viene tracciato il piano dei test da eseguire per la valutazione della combinazione migliore di parametri/algoritmo di costruzione del dataset e parametri del classificatore. Inoltre viene descritta la piattaforma e gli strumenti utilizzati per questo scopo.

4.3.1 Piano parametrico dei test

Per ciascuna delle due strategie di costruzione del dataset finale discusse in precedenza, i primi parametri da stabilire per i test sono sicuramente le dimensioni di monitor-window e di warning-window. Si ritengono scelte ragionevoli monitor-windows di dimensioni 1, 3 e 6 timeslot, che sono pari rispettivamente a 5, 15 e 30 minuti di osservazioni ricavate dai file di log. Per quanto riguarda le warning-windows, è logico che più elevato è il loro valore più tempo di preavviso sarà disponibile agli addetti per allestire le contromisure in vista di un allarme. Ragionando con i sistemisti aziendali è emerso che un tempo inferiore al quarto d'ora risulta insufficiente per questo scopo. A partire da questo vincolo sono state prese in considerazione warning-windows delle dimensioni di 3, 6, e 9 timeslot, ovvero 15, 30 e 45 minuti. L'aspettativa è che le prestazioni tendano a decadere a mano a mano che si aumenta la dimensione della warning-window.

Nello studio di Carlotta Domeniconi et al. [22] dopo la costruzione delle monitor-windows è stata inoltre effettuata una cernita euristica dei record, eliminando i non-allarmi che si trovano temporalmente adiacenti ad un record in allarme, questo per cercare di minimizzare la sovrapposizione tra le classi. È stato ritenuto opportuno applicare questa selezione scegliendo come intorno temporale ottimale 5 ore. In questo

modo tutti i record negativi rilevati nelle 5 ore precedenti e successive rispetto ad un record in allarme sono stati eliminati. Come accennato in fase di analisi delle informazioni a disposizione, i report degli allarmi Virtual Users spesso notificano le anomalie come raffiche di allarmi uno di seguito all'altro. In sede di studio proattivo di questi eventi, è triviale attribuire ai primi allarmi di queste raffiche un'importanza maggiore, essendo questi i veri obiettivi del classificatore per predire a tutti gli effetti un allarme. Tuttavia è altresì vero che applicando una pulizia in questo senso, ovvero mantenendo solo il primo record positivo di queste raffiche per il dataset di training, il numero dei record di classe positiva a disposizione scenderebbe drasticamente, inficiando la capacità di apprendimento del modello. Per questo motivo è stato deciso di testare sia i dataset con le raffiche di anomalia mantenute, sia quelli con il solo primo record di allarme.

Un altro aspetto da considerare è la dimensionalità del dataset a livello di features. Come discusso nel Capitolo 2.3.6, applicare qualche strategia di feature-selection, come la selezione in base al coefficiente di correlazione degli attributi, può portare a dei miglioramenti in termini di accuratezza. Oltretutto in un contesto online avere un dataset con meno attributi significa anche snellire il processo di raccolta dati dai report, vale quindi la pena esaminarne l'apporto prestazionale. Abbinato al coefficiente di correlazione implementanto nelle librerie Weka con la classe `CorrelationAttributeEval`, l'algoritmo `Ranker` di Weka crea una vera e propria classifica degli attributi in base ad uno score calcolato appunto sulla correlazione. Da alcuni test eseguiti si è notato come solitamente le features più valide siano tra le prime 15 posizioni del ranking, quindi si è scelto di mantenere questo quantitativo di attributi da selezionare in fase di feature selection.

Come accennato, nei problemi di failure prediction spesso i dataset a disposizione sono molto sbilanciati, in quanto i record riguardanti le anomalie sono numericamente molto inferiori rispetto ai record di periodi di normale attività del sistema. Per questo motivo addestrare un classificatore con ulteriori record campionati della classe di minoranza, ad esempio con SMOTE, può portare a dei miglioramenti sostanziali. Come tasso di over-sampling si è scelto il 200%, in modo da triplicare i record con la classe di allarme positiva.

Riepilogando, le variabili parametriche da valutare per la creazione del dataset per il train e test del modello sono le seguenti:

- Algoritmo di creazione MW/WW = {Partitioning-Variable-Distance, Scrolling-Fixed-Distance}.
- Dimensioni della monitor-window = {1, 3, 6} timeslot.
- Dimensioni della warning-window = {3, 6, 9} timeslot.
- Applicare o meno un processo di eliminazione delle raffiche di allarmi.
- Applicare o meno un processo di features-selection tramite correlation coefficient e Ranking-Top-15.
- Applicare o meno un processo di over-sampling del 200% della classe di minoranza tramite SMOTE.

Da precisare che viene tralasciata la combinazione di finestre MW = 6 e WW = 3, in quanto con l'algoritmo Partitioning-Variable-Distance ci sarebbe una perdita di dati. Infatti partizionare il dataset ogni sei timeslot e assegnare l'allarme controllando i successivi tre porta a tralasciare il controllo su tre timeslot, nei quali può occorrere

un allarme. Inoltre qualsiasi dataset per le operazioni di validazione dei parametri o di classificazione vera e propria è stato normalizzato, questo per agevolare i calcoli e le elaborazioni intensive sostenute dalle Support Vector Machines.

Svolgimento dei test

Al variare delle 8 combinazioni di costruzione dei dataset per le 8 combinazioni dimensionali delle finestre, si ottengono 64 dataset con algoritmo Partitioning-Variable-Distance e 64 dataset con algoritmo Scrolling-Fixed-Distance. Come si vede dalla Tabella 4.5 con il primo algoritmo si ottengono dataset di circa 10000 record considerando MW formate da un solo timeslot, che si abbassano a circa 2000 con MW formate da sei timeslot. Eliminando le raffiche rimangono solo circa 80 record con classe positiva, ovvero con allarme e costuiscono circa lo 0.7% del totale dei record nel caso peggiore (MW piccole) e il 4% nel caso migliore. Mantenendo le raffiche invece lo sbilanciamento delle classi si ridimensiona, con percentuali di allarmi comprese tra il 13% e il 28% del totale. Vista la logica a scorrimento dell'algoritmo Scrolling-Fixed-Distance, tutti i dataset ottenuti sono costituiti a grandi linee dallo stesso quantitativo di record, circa 12000 (vedi Tabella 4.6). Anche la quantità di allarmi non si discosta molto tra i dataset, circa 1000 mantenendo le raffiche (pari a circa il 10% del totale) e 85 senza raffiche (pari allo 0.7%).

Per ciascun dataset è stato inizialmente effettuato il tuning dei parametri SVM tramite 10 Fold-Cross-Validation. Sono stati così individuati i valori ottimali di gamma, C (Cost) ed i parametri del cost-sensitive classification. Quest'ultimo indica il costo attribuito ai falsi positivi e ai falsi negativi impostato durante la fase di training del modello tramite le librerie LibSVM (vedi Paragrafo 4.3.2). I parametri individuati garantiscono le migliori performance valutate su accuracy, AUC e TPR.

Questi settings sono stati poi sfruttati per l'addestramento dei classificatori per ciascun dataset. Sono state eseguite due prove, che si distinguono per la politica di estrazione dei record costituenti il training-set e quelli che formano il test-set.

4.3.2 Piattaforma di test

I test sono stati tutti eseguiti su un portatile dotato di processore dual-core Intel Core i5-3317U e 6 GB di RAM con sistema operativo Ubuntu 12.04. Per il task di data mining sono state utilizzate le librerie open-source del software Weka 3.7.1, affiancate alla libreria LibSVM per quanto concerne l'implementazione delle Support Vector Machines.

Weka 3.7.1

WEKA (Waikato Environment for Knowledge Analysis) è un software di machine learning open source, con licenza GNU e sviluppato in Java dall'Università di Waikato a partire dall'anno 1993. Nel 2005 ha ricevuto il premio SIGKDD Data Mining and Knowledge Discovery Service Award e nel 2006 è stata acquisita una licenza speciale dalla nota azienda di business intelligence Pentaho Corporation. Weka implementa molti tool e algoritmi di data visualization, data analysis e predictive modeling, accessibili tramite una comoda interfaccia grafica. Tramite questo software è possibile applicare ad un set di

MW	WW	BURSTS	RECORDS	ALARMS	NOT-ALARMS
1	3	YES	13467	1800	11667
		NO	11751	84	
1	6	YES	14201	2770	11431
		NO	11514	83	
1	9	YES	14674	3474	11200
		NO	11280	80	
3	3	YES	4691	812	3879
		NO	3963	84	
3	6	YES	4986	1187	3799
		NO	3881	82	
3	9	YES	5158	1436	3722
		NO	3802	80	
6	6	YES	2527	640	1877
		NO	1966	79	
6	9	YES	2598	755	1843
		NO	1922	79	

Tabella 4.5: Dimensioni dataset ottenuti con algoritmo Partitioning-Variable-Distance.

MW	WW	BURSTS	RECORDS	ALARMS	NOT-ALARMS
1	3	YES	12716	828	11888
		NO	11973	85	
1	6	YES	12738	884	11854
		NO	11939	85	
1	9	YES	12836	937	11899
		NO	11983	84	
3	3	YES	12969	1164	11805
		NO	11890	85	
3	6	YES	13040	1188	11852
		NO	11936	84	
3	9	YES	13051	1208	11843
		NO	11927	84	
6	6	YES	13160	1333	11827
		NO	11911	84	
6	9	YES	13159	1342	11817
		NO	11901	84	

Tabella 4.6: Dimensioni dataset ottenuti con algoritmo Scrolling-Fixed-Distance.

dati numerosi task di data mining, come classificazione, regressione, clustering, feature selection e preprocessing. L'ambiente principale per queste attività è *Explorer*, ma sono comunque accessibili o tramite linea di comando, oppure attraverso l'ambiente *Knowledge Flow* basato su interconnessioni di blocchi funzionali che garantiscono maggiore personalizzazione dell'attività da svolgere. L'utilizzo più comodo e rapido, specie in vista di molti test, risulta tuttavia essere l'implementazione delle librerie di Weka in codice sorgente Java.

LibSVM 3.17

Libreria per l'integrazione delle Support Vector Machines che consente l'applicazione di task come la classificazione (eventualmente anche multi-classe) e la regressione. È stata sviluppata da Chih-Chung Chang and Chih-Jen Lin presso la National Taiwan University in c++, ma sono disponibili molte estensioni e interfacce ad esempio per MATLAB, R, Python, Perl, Ruby, Weka, PHP, CUDA. Da sottolineare la presenza di diverse tipologie di kernel per le SVM, con i parametri personalizzabili a piacere.

4.4 Risultati ottenuti

Di seguito sono commentati i risultati dei test eseguiti con i due algoritmi di costruzione delle finestre proposti e discussi in precedenza. Sono state eseguite due prove, la prima con 10-Fold-Cross-Validation per il tuning dei parametri e per il test delle performance degli algoritmi progettati. La seconda invece concerne la separazione del training-set e del test-set in base a periodi temporali, in particolare utilizzando i dati dei primi tre mesi per il training e l'ultimo mese per la fase di test.

4.4.1 1° Test: 10-Fold-Cross-Validation

Come già accennato, l'applicazione di Cross Validation è stata effettuata soprattutto per il tuning dei parametri degli algoritmi e del classificatore. Come atteso, all'aumentare della dimensione di warning-window le prestazioni tendono a decadere. Questo calo comunque si è rivelato molto lieve, tanto da preferire comunque la dimensione maggiore provata di WW, ovvero 9 timeslot, per via dei benefici garantiti da un maggiore periodo di preavviso rispetto l'allarme.

Partitioning-Variable-Distance

Mantenendo le raffiche di allarmi, il dataset con il quale sono state riscontrate le prestazioni migliori è quello con la sola applicazione di features-selection, senza oversampling. Con questa configurazione abbinata ad una dimensione delle monitor-windows di tre timeslot, è possibile prevedere correttamente il 79% degli allarmi (TPR) entro un tempo di 45 minuti, con un'accuracy totale pari a circa 88% e AUC 0.85. In generale comunque, applicare un algoritmo di selezione delle features migliora le prestazioni solo nei casi di dimensioni di monitor-windows medio-alti. Al contrario SMOTE migliora il valore di AUC del 10% quando applicato a monitor-windows formate da un solo timeslot. I valori migliori per ciascuna configurazione sono confrontabili nella Tabella 4.7.

Senza raffiche di record positivi i risultati sono stati nulli quando non è stato applicato un algoritmo di oversampling. In questi dataset infatti lo sbilanciamento delle classi è molto forte, rendendo necessaria l'applicazione di SMOTE. La selezione delle features

in base al coefficiente di correlazione fa decadere il TPR del 20%, pur mantenendo valori di accuracy simili. Il risultato migliore si è quindi ottenuto con il solo oversampling, con monitor-windows di 30 minuti e warning-windows di 45 minuti. L'accuracy in questo caso è molto elevata (92.12%), con TPR del 77% e AUC 0.85 come il caso precedente con raffiche.

In generale, con raffiche o senza, si è notato come all'aumentare della dimensione delle monitor-windows l'accuracy tende a diminuire di circa il 5% ad ogni step. Questo calo è causato da un aumento del valore di False-Negative-Rate (FNR), infatti il valore degli allarmi correttamente predetti non subisce grandi variazioni. Come previsto, a parità di dimensione di monitor-window i valori di AUC rilevati tendono a diminuire con l'aumentare della dimensione della warning-window. In tutte le prove effettuate si nota inoltre una certa instabilità dei valori ottenuti, i trend segnalati infatti presentano a volte delle eccezioni. Questo è quasi certamente causato dal contributo variabile della warning-window, infatti in fase di training un record classificato positivamente indica un certo allarme entro la warning-window. In fase di testing la stessa tipologia di allarme può ricadere in attimi diversi della warning-window e la relativa monitor-window avrà inevitabilmente caratteristiche diverse, ricoprendo timeslot diversi antecedenti all'allarme.

MW	WW	BURSTS	F.S.	O.S.	TPR aav.	ACC av.	AUC av.
1	9	√	*X*	*X*	0,83	79,04	0,8
3	**9**	√	√	*X*	**0,79**	**88,06**	**0,85**
1	9	√	*X*	√	0,9	84,76	0,85
6	9	√	√	√	0,84	87,93	0,87
6	**9**	*X*	*X*	√	**0,77**	**92,12**	**0,85**
3	9	*X*	√	√	0,52	89,47	0,72

Tabella 4.7: Media dei risultati ottenuti con Partitioning-Variable-Distance con Cross-Validation.

Scrolling-Fixed-Distance

Anche con questo algoritmo, nei dataset con raffiche di allarmi mantenute, l'applicazione di features-selection migliora le performance nei casi di monitor-windows ampie. SMOTE invece tende a migliorare i risultati AUC in qualsiasi istanza, in media del 5%. Il risultato notevole è stato quindi ottenuto con eseguita sia la selezione degli attributi in base al correlation coefficient, sia con SMOTE. Per il dataset formato da monitor-windows ampie 6 timeslot sono stati predetti con anticipo di 45 minuti il 91% degli allarmi (TPR), con un'accuratezza generale del 97.1% e AUC 0.95. Gli altri risultati migliori per categoria sono consultabili nella Tabella 4.8.

Anche in questo caso per via dell'accentuato sbilanciamento tra classi si è reso imprescindibile l'utilizzo dell'oversampling per dataset senza raffiche di record positivi. In aggiunta, la features-selection migliora leggermente l'accuracy generale, ma peggiora in media del 20% il TPR. Il risultato migliore è stato conseguito con il solo oversampling, su dataset con monitor-windows di 6 timeslot. Con anticipo di 45 minuti è stato predetto il 68% degli allarmi, con un'accuracy del 98.7% e AUC 0.84.

Al contrario dell'algoritmo partitioning-variable-distance l'accuratezza rimane stabile anche con l'aumentare della dimensione delle finestre. In generale comunque aumentando le dimensioni delle monitor-windows le prestazioni in termini di AUC e TPR migliorano. Con l'algoritmo scrolling-fixed-distance, oltre ad un generale miglioramento delle prestazioni rispetto a partitioning-variable-distance, i valori acquisiti risultano molto più stabili, senza gli sbalzi anomali rilevati in precedenza. Questo è dovuto alla maggiore robustezza dell'algoritmo rispetto alla warning-window, essa rappresenta infatti l'intervallo temporale esatto che separa la monitor-window all'occorrenza dell'allarme. In questo modo i record con etichetta di allarme positiva tendono ad essere molto più correlati tra loro. Questo si riflette soprattutto in un'accuracy molto elevata (quasi sempre maggiore del 90%) e anche su una maggiore evidenza dei trend, ad esempio del lieve calo prestazionale all'aumentare della dimensione della warning-window.

MW	WW	BURSTS	F.S.	O.S.	TPR aav.	ACC av.	AUC av.
6	9	√	*X*	*X*	0,85	91,49	0,89
6	9	√	√	*X*	0,85	97,23	0,92
6	9	√	*X*	√	0,91	94,72	0,93
6	**9**	√	√	√	**0,91**	**97,11**	**0,95**
6	**9**	*X*	*X*	√	**0,68**	**98,72**	**0,84**
6	9	*X*	√	√	0,46	98,78	0,72

Tabella 4.8: Media dei risultati ottenuti con Scrolling-Fixed-Distance con Cross-Validation.

4.4.2 2° Test: Separazione temporale

In questa prova è stato considerato un training-set formato da tutti i record positivi o negativi riguardanti i mesi di Dicembre, Gennaio e Febbraio. I record del mese di Marzo sono stati usati invece come test-set per la valutazione del classificatore. Questo per provare il comportamento del modello su dati temporalmente diversi. Come controprova degli esiti ottenuti sono stati anche variati i mesi componenti tali dataset, affidando ad esempio il training ai mesi di Dicembre, Gennaio e Marzo e il test a Febbraio.
I risultati ottenuti con questa modalità sono indipendenti dai mesi utilizzati per il test e deludenti rispetto alla precedente prova con qualsiasi algoritmo di costruzione del dataset. I risultati migliori sono stati conseguiti con algoritmo Scrolling-Fixed-Distance, in particolare senza applicare oversampling o features-selection. Mantenendo le raffiche è possibile infatti prevedere con 45 minuti di anticipo il 69.4% degli allarmi con un'accuratezza molto bassa (62.9%) e AUC 0.657. La dimensione preferibile delle monitor-windows risulta essere pari ad un timeslot. Se utilizzati dataset senza raffiche di allarmi e algoritmo Scrolling-Fixed-Distance con 45 minuti di anticipo si riesce a predire il 26% degli allarmi, con accuratezza elevata 94.7% e AUC 0.608, anche questa volta senza features-extraction o oversampling, ma con dimensione monitor-window pari a 6 timeslot (vedi Tabella 4.9).

Le cause di questo calo di prestazioni sono da imputare soprattutto al cosiddetto *Concept Drift* [26], un concetto fondamentale per il mining online. Con Concept Drift

si intende la variazione dei valori delle features con il passare del tempo, detto in altre parole la stagionalità dei dati. Infatti è triviale decretare che trend di dati analizzati in mesi come Dicembre, Gennaio e Febbraio possono essere diversi rispetto all'andamento riscontrato in Marzo. Non solo i valori degli attributi possono variare, ma anche le cause alla base degli allarmi possono essere distinte. Anomalie diverse sfociano in report diversi e se non modellate in precedenza la classificazione non può che fallire. Questa prova sottolinea ancora di più la necessità di una classificazione online, con un modello che si auto-aggiorna con il tempo per seguire il concept drift.

MW	WW	BURSTS	F.S.	O.S.	TPR av.	ACC av.	AUC av.
1	**9**	√	*X*	*X*	**0,694**	**62,9**	**0,657**
1	9	√	√	*X*	0,63	59,39	0,609
1	9	√	*X*	√	0,481	68,23	0,597
1	9	√	√	√	0,272	67,38	0,561
6	**9**	*X*	*X*	*X*	**0,261**	**94,69**	**0,608**
6	9	*X*	√	*X*	0,26	90,6	0,587
6	9	*X*	*X*	√	0,174	93,64	0,559
6	9	*X*	√	√	0,217	92,95	0,577

Tabella 4.9: Risultati di test su Febbraio ottenuti con Scrolling-Fixed-Distance.

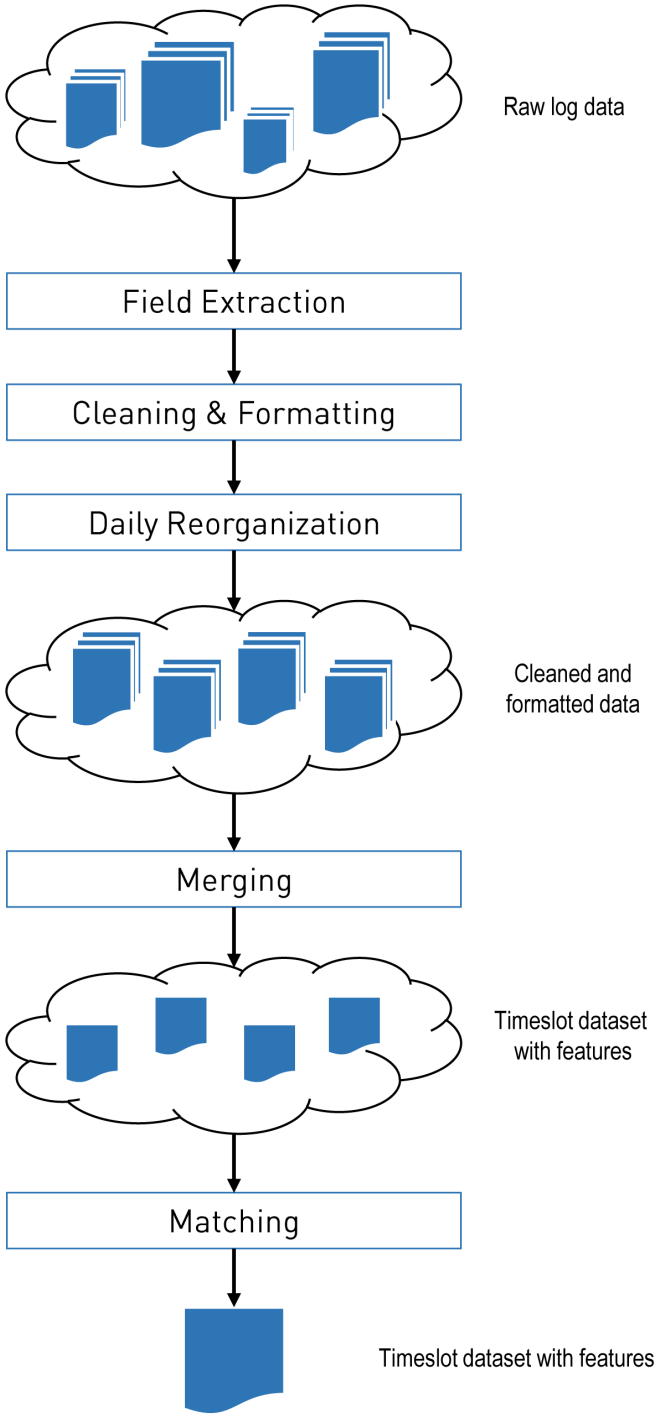

Figura 4.3: Preprocessing dei file di log: Cleaning, Transformation e Data Reduction.

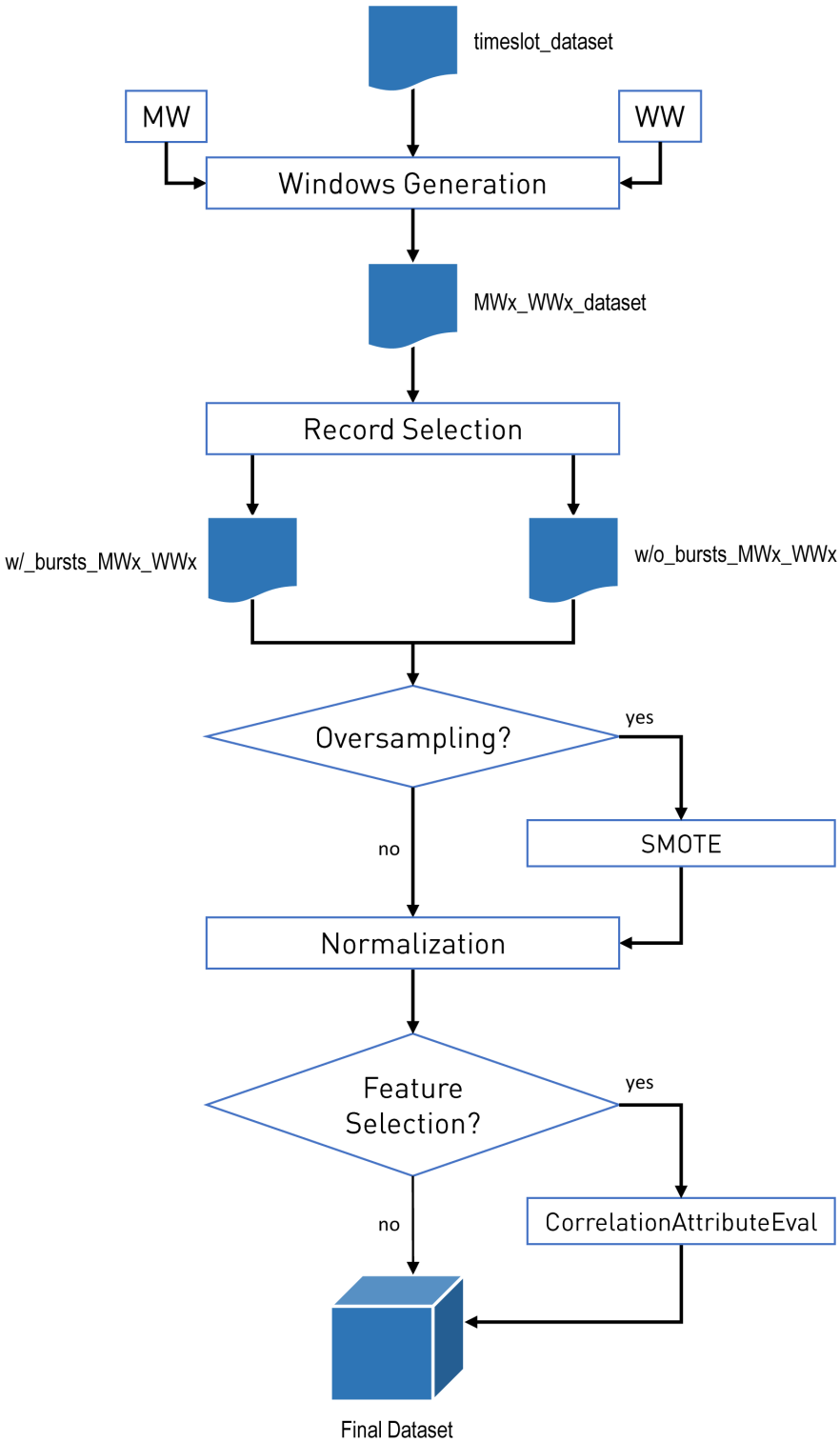

Figura 4.4: Fasi di costruzione dei dataset per i test.

Capitolo 5

Conclusioni

Nella Tesi è stato effettuato uno studio preliminare per un sistema di notifica proattiva degli allarmi in un sistema informativo aziendale. Sono stati analizzati i report prodotti, delineate le features estraibili e sviluppati i programmi atti all'estrazione e al preprocessing dei dati grezzi. Successivamente sono stati progettati due diversi algoritmi per la costruzione di un dataset adatto ad un task di classificazione, ovvero Partitioning-Variable-Distance e Scrolling-Fixed-Distance, aventi caratteristiche diverse. I dataset prodotti sono serviti per addestrare un classificatore Support Vector Machines con il fine di individuare le caratteristiche e le configurazioni che garantiscono le prestazioni migliori. In base a questi risultati ottenuti con test offline, è possibile gettare le basi di partenza per lo sviluppo di un sistema online di monitoring con modello auto-aggiornante.

In particolare sono da segnalare le prestazioni ottime conseguite con l'algoritmo Scrolling-Fixed-Distance, che in caso di allarme garantisce in maniera esatta il tempo utile rimanente per la predisposizione delle contromisure. La dimensione delle monitor-windows che ha avuto in generale risultati migliori è stata di 6 timeslot, ovvero features riguardanti 30 minuti di attività. Sicuramente il caso d'uso più interessante del modello in caso di futuri sviluppi è quello di riuscire a prevedere con buon anticipo il primo degli allarmi, piuttosto che allarmi che occorrono in successione ad altri. Per questo particolare quadro è stato provato che con l'algoritmo Scrolling-Fixed-Distance è possibile garantire una corretta predizione del 68% delle anomalie al sistema. Inoltre l'accuracy generale molto elevata, circa il 98.7%, assicura una riduzione sostanziale dei falsi allarmi. Infatti eliminando le raffiche si ottengono record di classe positiva più "puri" per la fase di training. I risultati ottenuti sono comunque in linea con i migliori valori ottenuti da altri ricercatori in ambiti equivalenti.

Sono state testate anche tecniche di oversampling tramite SMOTE e di features-selection in base al coefficiente di correlazione, che a seconda della dimensione delle monitor-windows impostata e dell'algoritmo utilizzato garantiscono o meno dei miglioramenti. L'oversampling si rivela irrinunciabile in dataset senza raffiche di allarmi, ovvero mantenendo solo il primo di più record con classe positiva in sequenza, a causa del troppo sbilanciamento tra le classi che si viene a creare. SMOTE però è ovviamente controproducente quando utilizzato su un modello di classificazione addestrato su un periodo temporale differente rispetto ai record di test. Features-selection quando applicata ha dimostrato come gli attributi più importanti selezionati siano più o meno sempre gli stessi, cioè quelli inizialmente già imputati dai sistemisti aziendali come

specchio di salute del sistema.

Oltre alle prove effettuate applicando 10-Fold-Cross-Validation, è stata condotta una seconda prova, che si differenzia per la politica di costruzione dei dataset di training e di test. Nello specifico sono stati utilizzati i dati relativi ai primi tre mesi di attività per il dataset di training e l'ultimo mese per il test del modello. I risultati di performance ottenuti però appaiono in prima analisi deludenti rispetto agli ottimi valori della Cross-Validation. Prima di spiegare il motivo alla base, occorre precisare alcune premesse. I report virtual-users dai quali sono stati decretati gli istanti temporali di failure derivano da segnalazioni di sonde che testano la disponibilità dei servizi. L'allarme infatti può derivare ad esempio da problematiche di natura hardware o software oppure da sovraccarico del sistema o altro. Questi diversi failure hanno conseguenze sulla reportistica molto differenti e le anomalie presenti nei vari mesi a disposizione non sono sempre le stesse. Il mescolamento dei record infatti consente la costruzione di un modello "conscio" delle tipologie di allarmi che si possono presentare e dei relativi contesti, ovvero i valori delle features in quelle particolari finestre temporali. Mesi di log diversi non solo non condividono le stesse tipologie di allarmi ma presentano anche contesti di utilizzo dei servizi diversi, ad esempio sulla quantità e sul tipo di richieste effettuate. Purtroppo la reportistica attuale non consente di trarre conclusioni sulle tipologie di problemi riscontrati. Inoltre un training-set ricavato da report inerenti 3 mesi di attività non è sufficiente per coprire tutti i casi possibili di failure. Nonostante questo aspetto, dalle prove effettuate si percepisce chiaramente che anche con un anticipo di 45 minuti è possibile individuare con una buona accuratezza l'approssimarsi di un failure, modellando una buona separazione tra le classi di allarme e non-allarme.

Il lavoro svolto può essere ottimizzato se, oltre all'occorrenza dell'anomalia e quindi al relativo timestamp, viene notificato dalla sonda o dagli addetti anche il tipo di problema riscontrato. In questo modo, possono essere delineate le tipologie dei problemi che sono occorse fino a quel momento e la costruzione del training-set risulterebbe sicuramente più oculata. Inoltre come base di partenza è ritenuta ragionevole una copertura di report di almeno 8 mesi, per ridurre al minimo i falsi positivi. Il modello risultante può in questo modo fungere da base di partenza per una messa in opera online sul sistema aziendale, predisponendo degli aggiornamenti del modello nel corso del tempo sia ad intervalli regolari, sia ad esempio al verificarsi di una nuova tipologia di anomalia o in caso di mancata previsione. In alternativa è comunque possibile iniziare l'attività di classificazione con un modello addestrato su pochi mesi, accettando inizialmente prestazioni non ottimali.

Nell'ambito del failure prediction online è imprescindibile tenere conto di due fattori: la mole di dati necessari al modello e il continuo cambiamento del contesto di utilizzo dei servizi, il cosiddetto *Concept Drift*. Soprattutto se questi servizi sono legati alla posta elettronica, i carichi di utilizzo sono molto variabili a seconda del numero di utenti, del periodo della giornata e dell'anno. L'importanza di quest'aspetto è stato provato anche in corso d'opera, dalle prestazioni deludenti ottenute con il training-set e il test set costruiti su mesi differenti. Interessanti sviluppi futuri della tesi possono riguardare implementazioni delle recenti tecniche di *Data Stream Mining* [27], che riescono a tener conto del Concept Drift. Il data stream mining è il processo inerente all'estrazione di informazione interessante da flussi di dati continui, come ad esempio i file di log. La libreria più illustre per questo campo è MOA (Massive Online Analysis) [28]. Alcuni recenti algoritmi come i Very Fast Decision Tree (VFDT) [29] e le successive evoluzioni CVFDT [30], VFDTc [31] oppure On-line Information Network (OLIN) [32] e IOLIN [33] rappresentano degli alberi decisionali incrementali che tengono conto del concept

drift. Essendo incrementali inoltre si aggiornano utilizzando soltanto le nuove istanze di dati, senza il bisogno di ri-processare l'intero dataset. In questo modo viene risolto sia il problema della mole di dati, sia la stagionalità dei dati.

Appendice A

Elenco features

CLASSE	1	ALERT	+1, -1	+1 se nella warning-window successiva un allarme
	2	TIMESLOT	1-288	Timeslot della giornata
	3	DAYOFWEEK	1-7	Giorno della settimana
APACHE ACCESS	4	GET2XX	num	# richieste GET 2xx
	5	GET3XX	num	# richieste GET 3xx
	6	GET4XX	num	# richieste GET 4xx
	7	GET5XX	num	# richieste GET 5xx
	8	POST2XX	num	# richieste POST 2xx
	9	POST3XX	num	# richieste POST 3xx
	10	POST4XX	num	# richieste POST 4xx
	11	POST5XX	num	# richieste POST 5xx
	12	DELETE2XX	num	# richieste DELETE 2XX
	13	DELETE3XX	num	# richieste DELETE 3XX
	14	DELETE4XX	num	# richieste DELETE 4XX
	15	DELETE5XX	num	# richieste DELETE 5XX
	16	BIGGERTHAN1MB	num	# richieste di dimensioni superiori ad 1 MB
APACHE ERROR	17	FILENOTEXIST	num	# file non trovato
	18	HANDSHAKEFAIL	num	# handshake falliti
	19	REQUESTFAIL	num	# richieste fallite
	20	TIMEOUTEXPIRED	num	# timeout scaduti
	21	SSLNEGOTIATIONFAIL	num	# connessioni ssl fallite
	22	EOFFOUND	num	# eof
	23	SSLLIBRARYERROR	num	# errori ssl
	24	CONNECTIONRESET	num	# connessioni resettate
	25	PERMISSIONDENIED	num	# permesso negato
	26	DIFFERENTHOSTNAMES	num	# hostnames che non coincidono

JBOSS ACCESS	27	GET2XX	num	# richieste GET 2xx
	28	GET3XX	num	# richieste GET 3xx
	29	GET4XX	num	# richieste GET 4xx
	30	GET5XX	num	# richieste GET 5xx
	31	POST2XX	num	# richieste POST 2xx
	32	POST3XX	num	# richieste POST 3xx
	33	POST4XX	num	# richieste POST 4xx
	34	POST5XX	num	# richieste POST 5xx
	35	DELETE2XX	num	# richieste DELETE 2XX
	36	DELETE3XX	num	# richieste DELETE 3XX
	37	DELETE4XX	num	# richieste DELETE 4XX
	38	DELETE5XX	num	# richieste DELETE 5XX
	39	BIGGERTHAN1MB	num	# richieste di dimensioni superiori ad 1 MB
	40	RESTARTS	num	# riavvii dell' AS
WEBMAIL ACCESS	41	LOGIN	num	# login
	42	LOGOUT	num	# logout
	43	DELMESS	num	# messaggi eliminati
	44	SENDMESS	num	# messaggi inviati
	45	COPYMESS	num	# messaggi inviati
	46	CREATEFOLDER	num	# cartelle create
	47	RENAMEFOLDER	num	# cartelle rinominate
	48	DELETEFOLDER	num	# cartelle con flag eliminato
	49	EXPUNGEFOLDER	num	# cartelle eliminate
	50	LOSTSESSION	num	# sessioni perse
	51	FLAGDELMESSFOLDER	num	# cancellazioni multiple da cartella
	52	MASSIVESENDING	num	# invii massivi
SENDMAIL	53	ENVELOPE	num	# richieste di invio effettive transitate sul sistema
	54	DELIVERYATTEMPTS	num	# tentativi di invio transitati sul sistema
	55	SENDINGCOMPLETE	num	# invii effettivamente completati
	56	SENDINGDEFERRED	num	# invii posticipati
	57	SENDINGERROR	num	# errori negli invii
	58	ENVELOPEBYTES1MB	num	# richieste di invio di messaggi superiori ad 1 MB
	59	DELAYSENDINGCOMPLETE1MIN	num	# invii completati con ritardo superiore ad 1 Min
	60	VIRUSFOUND	num	# virus trovati

COURIER	61	CONNECTION	num	# connessioni
	62	DISCONNECTION	num	# disconnessioni
	63	PROXYLOGIN	num	# login
	64	LOGOUT	num	# logout
	65	ERROR	num	# errori

Tabella A.1: Lista delle features

Bibliografia

[1] U. Fayyad, G. Piatetsky-shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI Magazine*, vol. 17, pp. 37–54, 1996.

[2] R. Kosala and H. Blockeel, "Web mining research: A survey," *SIGKDD Explor. Newsl.*, vol. 2, pp. 1–15, June 2000.

[3] T. Fawcett, "Roc graphs: Notes and practical considerations for researchers," tech. rep., 2004.

[4] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *IJCAI*, vol. 14, pp. 1137–1145, 1995.

[5] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[7] H. Liu and H. Motoda, *Feature extraction, construction and selection: A data mining perspective*. Springer, 1998.

[8] Z. Zheng, "Feature selection for text categorization on imbalanced data," *ACM SIGKDD Explorations Newsletter*, vol. 6, p. 2004, 2004.

[9] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[10] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[11] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Comput. Surv.*, vol. 42, pp. 10:1–10:42, Mar. 2010.

[12] A. Avizienis, J. claude Laprie, B. R, C. L, and S. Member, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 11–33, 2004.

[13] G. Hamerly and C. Elkan, "Bayesian approaches to failure prediction for disk drives," in *In Proceedings of the eighteenth international conference on machine learning*, pp. 202–209, Morgan Kaufmann, 2001.

[14] J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado, "Hard drive failure prediction using non-parametric statistical methods,"

[15] D. Turnbull and N. Alldrin, "Failure prediction in hardware systems,"

[16] H. R. Berenji, J. Ametha, and D. Vengerov, "Inductive learning for fault diagnosis," in *Fuzzy Systems, 2003. FUZZ'03. The 12th IEEE International Conference on*, vol. 1, pp. 726–731, IEEE, 2003.

[17] P. Bodik, G. Friedman, L. Biewald, H. Levine, G. Candea, K. Patel, G. Tolle, J. Hui, A. Fox, M. Jordan, and D. Patterson, "Combining visualization and statistical analysis to improve operator confidence and efficiency for failure detection and localization," in *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on*, pp. 89–100, June 2005.

[18] E. Kiciman and A. Fox, "Detecting application-level failures in component-based internet services," *Neural Networks, IEEE Transactions on*, vol. 16, pp. 1027–1041, Sept 2005.

[19] G. A. Hoffmann, K. S. Trivedi, and M. Malek, "A best practice guide to resource forecasting for computing systems.," *IEEE Transactions on Reliability*, vol. 56, no. 4, pp. 615–628, 2007.

[20] S. Fu and C.-Z. Xu, "Quantifying temporal and spatial correlation of failure events for proactive management," in *Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems*, SRDS '07, (Washington, DC, USA), pp. 175–184, IEEE Computer Society, 2007.

[21] R. K. Sahoo, A. J. Oliner, I. Rish, M. Gupta, J. E. Moreira, S. Ma, R. Vilalta, and A. Sivasubramaniam, "Critical event prediction for proactive management in large-scale computer clusters," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 426–435, ACM, 2003.

[22] C. Domeniconi, C.-S. Perng, R. Vilalta, and S. Ma, "A classification approach for prediction of target events in temporal sequences.," in *PKDD* (T. Elomaa, H. Mannila, and H. Toivonen, eds.), vol. 2431 of *Lecture Notes in Computer Science*, pp. 125–137, Springer, 2002.

[23] G. M. Weiss and H. Hirsh, "Learning to predict rare events in event sequences," in *In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 359–363, AAAI Press, 1998.

[24] AGID, "Codice dell'amministrazione digitale." `http://www.agid.gov.it/agenda-digitale/codice-amministrazione-digitale`.

[25] J. Goyvaerts, "Possessive quantifiers." `http://www.regular-expressions.info/possessive.html`.

[26] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, (New York, NY, USA), pp. 226–235, ACM, 2003.

[27] A. Bifet and R. Kirkby, "Data stream mining a practical approach."

[28] "Moa: Massive online analysis."
`http://moa.cms.waikato.ac.nz/`.

[29] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, (New York, NY, USA), pp. 71–80, ACM, 2000.

[30] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, (New York, NY, USA), pp. 97–106, ACM, 2001.

[31] J. a. Gama, R. Fernandes, and R. Rocha, "Decision trees for mining data streams," *Intell. Data Anal.*, vol. 10, pp. 23–45, Jan. 2006.

[32] M. Last, "Online classification of nonstationary data streams," *Intell. Data Anal.*, vol. 6, pp. 129–147, Apr. 2002.

[33] L. Cohen, G. Avrahami, M. Last, and A. Kandel, "Info-fuzzy algorithms for mining dynamic data streams," *Appl. Soft Comput.*, vol. 8, pp. 1283–1294, Sept. 2008.

Elenco delle tabelle

Elenco delle figure