POLITECNICO DI TORINO

Collegio di Ingegneria Informatica

Corso di Laurea Magistrale
in Ingegneria Informatica

Tesi di Laurea Magistrale

Modelli predittivi nell'Enterprise System Manager

Relatore

Prof. Paolo Garza

Candidato

Michele Giovanni Brianda

Luglio 2018

Indice

1 Introduzione

In un periodo storico in cui l'utilizzo di sistemi tecnologicamente evoluti è diventato ormai essenziale per lo sviluppo dei servizi offerti dalle società, la gestione delle infrastrutture tecnologiche a supporto di tali sistemi ha raggiunto livelli di complessità tanto elevati da rendere necessario un aumento delle assunzioni nel reparto IT tra il 60÷70% per le imprese con più di 500 dipendenti [1] e un incremento del 44% nel budget dedicato all'innovazione tecnologica da parte delle aziende nel 2018 [1].

In questo senso le nuove figure lavorative a supporto delle aziende devono essere in grado di far fronte alla gestione di un ecosistema informatico molto variegato che spazia dall'hardware al livello applicativo, come mostrato nella figura 1.1, e saper affrontare problemi e difficoltà velocemente e con efficienza.

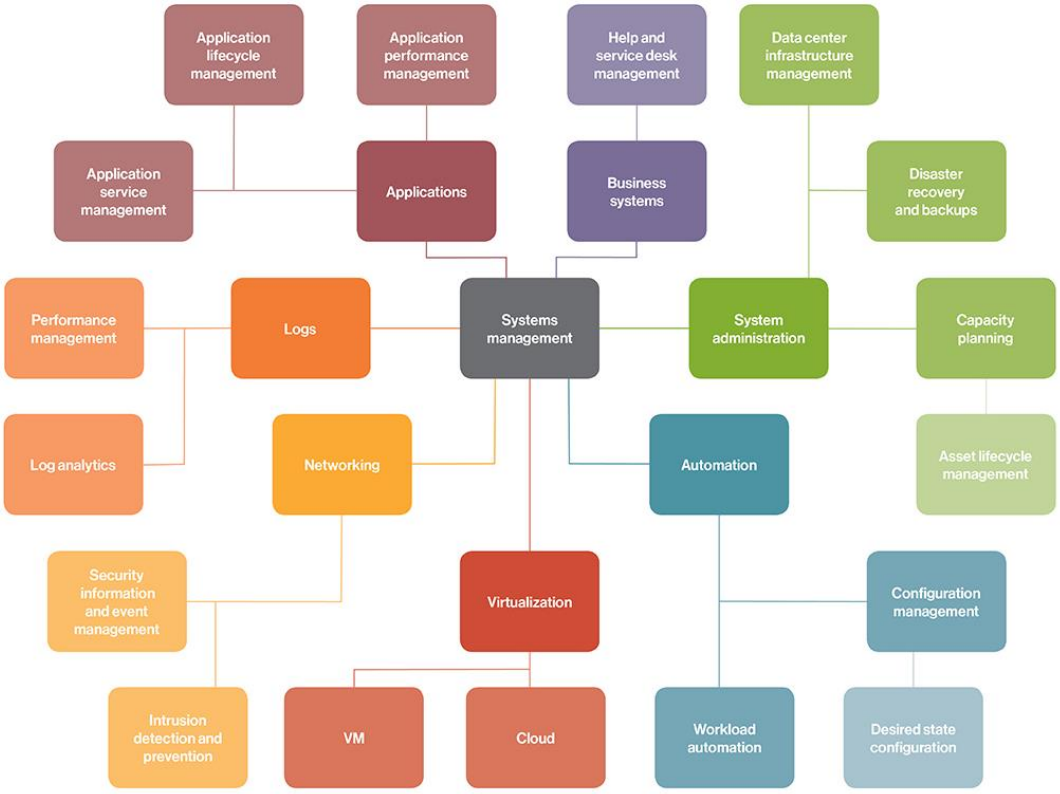

Figura 1.1: suddivisione di massima di una infrastruttura informatica nei suoi sottosistemi [2]

Nel contesto citato risulta quindi evidente la necessità di ottimizzare e automatizzare il più possibile le operazioni di manutenzione delle infrastrutture informatiche per consentire agli addetti ai lavori di focalizzarsi solo sugli aspetti più complessi della gestione dell'IT aziendale. Per questo motivo nel tempo hanno suscitato particolare interesse i cosiddetti "sistemi ingegnerizzati", installazioni che integrano hardware e software per una maggiore efficienza e semplicità del prodotto finale: ne sono un esempio i prodotti messi in commercio da Oracle che hanno registrato un incremento nelle vendite del 30% nel terzo trimestre del 2014 [3] .

Dato il chiaro indirizzo che il mercato ha ormai preso da tempo, la ricerca qui descritta vuole essere un punto di partenza per l'approfondimento delle tecniche di manutenzione predittiva nel settore IT e, più nello specifico, in ambienti che sfruttano prodotti Oracle.
In questo capitolo si illustreranno dapprima gli obiettivi del lavoro svolto, per poi descrivere i passaggi essenziali che l'hanno costituito: poiché la ricerca è stata portata avanti in ambito aziendale se ne descriverà il contesto e successivamente lo stato dell'arte relativo al concetto di manutenzione predittiva e degli algoritmi usati.

1.1 Obiettivi della ricerca

Nell'ottica di una sempre più accurata ottimizzazione dei processi aziendali per la gestione delle infrastrutture informatiche, la ricerca vuole essere un progetto pilota per l'applicazione di algoritmi di machine learning sui dati derivanti dal monitoraggio di alcuni sistemi di interesse al fine di trovare dei modelli matematici capaci di predire, con una certa accuratezza, la futura presenza di eventuali problemi considerando lo stato corrente dei sistemi analizzati.
In questo senso si è voluto prendere in considerazione i dati, relativi alle tecnologie di interesse aziendale, campionati con diversa frequenza dal software di monitoraggio "Oracle Enterprise System Manager": tale software, come descritto più in dettaglio nel capitolo *3 "Oracle Enterprise System Manager"*, consente la configurazione di alert tali per cui, superata una soglia relativa a una particolare metrica di un dato target, viene inviato un messaggio all'amministratore del sistema che dovrà considerare i provvedimenti necessari affinché le statistiche relative a tale metrica rientrino entro i limiti imposti dalla soglia citata.
Il sistema a soglie descritto fa parte della cosiddetta "manutenzione preventiva" ed in particolare della "manutenzione secondo condizione" [4]: con tale termine si vuole indicare una politica di gestione dei sistemi supervisionati che implica la reazione da parte degli

amministratori di sistema alla ricezione di eventuali allarmi dati dalla misurazione di alcune grandezze di interesse.
Obiettivo della ricerca è invece quello di approfondire la fattibilità dell'applicazione della cosiddetta "manutenzione predittiva" nell'ambito IT del contesto aziendale studiato: secondo questo tipo di politica di manutenzione, i dati relativi ai sistemi monitorati vengono processati con dei modelli matematici capaci di stimare la probabilità che vi sia un guasto analizzando alcuni parametri utili allo scopo che descrivono lo stato di salute corrente del sistema considerato [5].
Scopo ultimo del progetto è quindi quello di dare agli amministratori delle infrastrutture informatiche un supporto aggiuntivo nel monitoraggio dei sistemi gestiti per diminuire il numero di effetti negativi dati dal verificarsi di un problema imprevisto.

1.2 Workflow della ricerca

La ricerca è stata condotta secondo la timeline presente in figura 1.2 e si sviluppa quindi in sei passi successivi che partono dallo studio del contesto aziendale per arrivare all'applicazione dei risultati per il raggiungimento degli obiettivi societari.

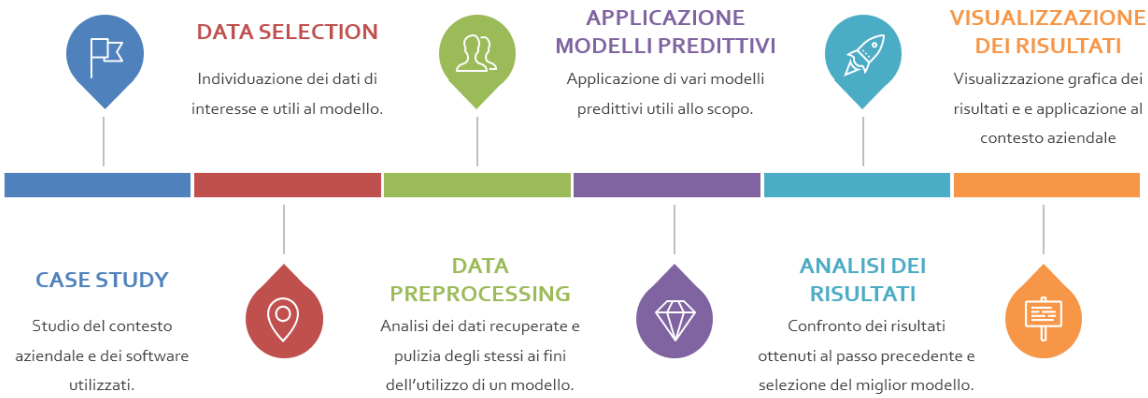

Figura 1.2: workflow della ricerca rappresentate i sei passi seguiti per lo sviluppo del lavoro

Nello svolgimento del lavoro si sono susseguite quindi varie fasi a partire dall'approfondimento dei processi aziendali che si volevano analizzare: sono stati studiati infatti i software utili al monitoraggio dei sistemi informatici, quali "Oracle Enterprise

System Manager" e "Oracle Database", e le strutture dati usate dagli stessi per la gestione dei dati campionati.
Dopo una prima parte di studio del contesto aziendale si sono quindi individuati i dati di interesse su cui focalizzare la ricerca e quelli utili per l'applicazione dei modelli matematici usati.
Dei dati prelevati dall'ambiente analizzato se ne è effettuato quindi il preprocessing, come descritto in dettaglio nel paragrafo 4.3 "*Preprocessing*", tramite l'utilizzo di varie tecniche quali ad esempio l'oversampling, la standardizzazione e la feature selection.
I passi successivi comprendono quindi la scelta e l'applicazione dei modelli matematici più consoni per l'analisi da effettuare e la loro ottimizzazione: si è deciso infatti di confrontare più modelli matematici sulla base di alcuni valori importanti (accuracy, precision, recall, confusion matrix) per trovare quello più performante ai fini della ricerca. Sono state fatte inoltre scelte di compromesso con l'obiettivo di ottimizzare alcuni aspetti critici a livello aziendale penalizzandone altri meno importanti in questo senso.
Per ultimo i risultati sono stati analizzati, interpretati e proposti in forma grafica per una migliore lettura e sono stati infine applicati gli esiti della ricerca al contesto aziendale.

1.3 Contesto aziendale

La ricerca nasce dalla necessità aziendale di migliorare l'efficienza dei processi di manutenzione e monitoraggio delle infrastrutture tecnologiche dei clienti seguiti.
In questo senso l'organizzazione di tale sistema prevede l'utilizzo di un software di monitoraggio che consente la configurazione di alcune soglie di interesse sulle tecnologie dei clienti osservate: in base alla gravità dell'evento i consulenti informatici impiegati nel monitoraggio devono infatti essere in grado di valutare i provvedimenti più utili per il rientro del problema scatenatosi.
Parallelamente al sistema sopracitato l'azienda gestisce un servizio di ticketing che permette ai clienti seguiti di comunicare eventuali problemi o malfunzionamenti in modo che questi possano essere risolti nel più breve tempo possibile.
Nel contesto aziendale studiato risulta significativo quindi il numero di alert che il software di controllo invia: a seconda della gravità dell'evento verificatosi, infatti, si possono scatenare alert di vario tipo, tra cui Warning e Critical, per ciascuna tecnologia (target) monitorata.
Il grafico in figura 1.3 riassume brevemente il numero di alert segnalati per ogni mese a partire da gennaio 2017.

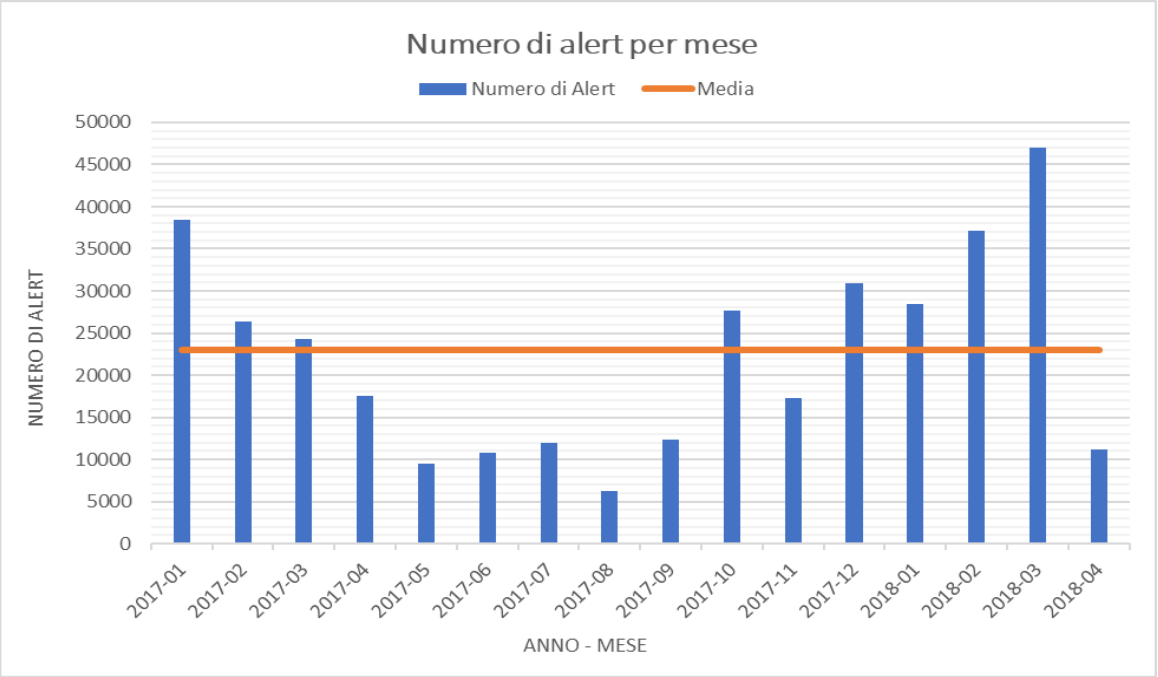

Figura 1.3: grafico rappresentante il numero di alert per mese e la relativa media totale

Dalla figura 1.3 si nota quindi come la media degli alert per mese sia superiore ai 22000, considerando i 1009 target monitorati di 51 tipi (target type) di tecnologie diverse (dati aggiornati al 15/05/2018).
Risulta quindi evidente la necessità da parte degli operatori di selezionare solo gli avvisi più importanti: in questo senso la ricerca vuole essere di supporto agli operatori del settore al fine di concentrare gli sforzi solo sulle problematiche più rilevanti.

2 Stato dell'arte

La ricerca sviluppata è nata, come spiegato nel capitolo precedente, in un contesto aziendale in cui si richiedeva un approfondimento delle tematiche della manutenzione predittiva in ambito informatico: in questo capitolo, quindi, vengono prima illustrate le caratteristiche di questo tipo di politica di manutenzione delle infrastrutture tecnologiche e come tale approccio influisca nei processi aziendali. In secondo luogo si definiscono quei modelli matematici che sono stati usati durante lo svolgimento del lavoro insieme ai parametri che ne hanno permesso una valutazione delle performance. Per ultimo si elencano gli strumenti utilizzati a partire dalla fase di raccolta dei dati fino a quella di analisi dei risultati ottenuti.

2.1 La manutenzione predittiva

L'ambito in cui la ricerca si sviluppa è quello della cosiddetta "manutenzione predittiva": con tale espressione si definisce quella politica di gestione dei sistemi informatici che cerca di prevenire eventuali problemi sfruttando le tecniche messe a disposizione dal machine learning.
Tale politica di manutenzione permette di ottenere numerosi vantaggi dal punto di vista dell'efficienza con la quale si risolvono i problemi: la capacità di identificare alcuni pattern che ripetitivamente si ripropongono prima dello scatenarsi di un evento problematico, consente infatti di agire sulle cause di un eventuale guasto prima ancora che questo si possa presentare, riducendo notevolmente quello che viene chiamato "MDT" ("Mean Down Time"), ovvero il tempo medio di non funzionamento di un sistema.

Come mostrato dalla figura 2.1, la messa in pratica di questo tipo di manutenzione richiede però un notevole sforzo da parte degli operatori nell'apprendimento di quelle che sono le problematiche essenziali dei sistemi studiati e i dati a disposizione per la loro analisi.
Dapprima è necessario infatti una fase preparatoria in cui si comprendono a fondo gli obiettivi e i sistemi utilizzati nell'ambiente in cui si vuole applicare questo tipo di manutenzione: solo successivamente, dopo aver circoscritto il dominio dei dati a disposizione, si elaborano i campioni con le tecniche di preprocessing per poi applicarli ai modelli matematici più consoni per l'obiettivo.

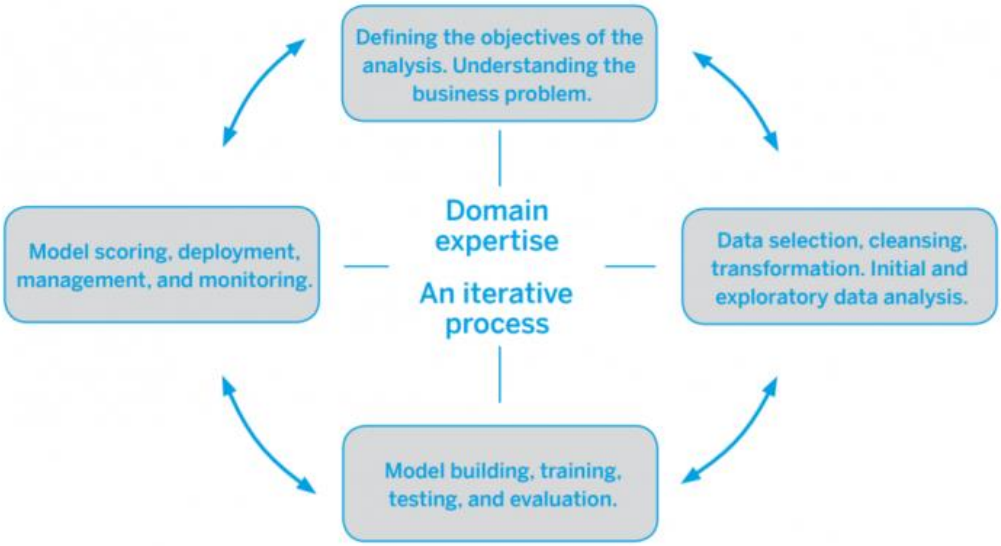

Figura 2.1: ciclo di gestione delle infrastrutture per mezzo della manutenzione predittiva

Visto l'ambito relativamente nuovo in cui vengono applicate le tecniche di machine learning, non sono molti gli studi che affrontano la questione in maniera sistematica: una gran parte degli studi, infatti, si concentrano sull'applicazione dei modelli predittivi nell'ambito della manutenzione di macchinari meccanici per mezzo del monitoraggio di caratteristiche fisiche raccolte tramite l'uso di sensori esterni. Si stima che entro 20 anni la maggior parte dei processi industriali avrà integrato un sistema capace di attuare automaticamente la manutenzione predittiva [6].

Nel caso di questa ricerca, invece, si sono voluti applicare tali modelli matematici in un contesto prettamente informatico, sfruttando i dati messi a disposizione dall'uso di un software proprietario quale Oracle Enterprise Manager.

2.2 I modelli predittivi

Nel corso della ricerca sono stati studiati e applicati alcuni modelli matematici utilizzati per l'analisi dei dati raccolti tramite Oracle Enterprise System Manager allo scopo di predire eventuali problemi nei sistemi informatici monitorati.

Questa sezione vuole quindi illustrare le caratteristiche principali dei tre modelli di classificazione usati (Random Forest, Support Vector Machine e Neural Network) insieme ai loro iperparametri più importanti la cui ottimizzazione ha consentito un miglioramento delle prestazioni degli algoritmi.
Per poter confrontare i risultati del lavoro svolto si illustrano, inoltre, i parametri che hanno consentito una valutazione dei risultati, dando maggiore risalto a quegli aspetti maggiormente legati all'obiettivo della ricerca e del contesto aziendale in cui si è sviluppato il lavoro.

2.2.1 Random Forest

Tra gli algoritmi di classificazione supervisionati il Random Forest è uno dei più conosciuti e apprezzati tanto per la facilità di utilizzo quanto per la sua robustezza: come suggerisce il nome, l'algoritmo si basa sulla creazione di una foresta di alberi di decisione e la selezione casuale degli attributi.
Più in dettaglio, il funzionamento del Random Forest si può dividere in due fasi principali:

- creazione della foresta di alberi casuali
- elaborazione delle predizioni sulla base della foresta di alberi creata

La prima delle due fasi consiste nella selezione casuale di un numero di feature minore rispetto a quello presente nel dataset originale e nel calcolo del miglior nodo possibile (e dei suoi figli) sulla base degli stessi criteri usati dall'algoritmo Decision Tree (Gini Index e Information Gain). Dopo aver ottenuto il numero di nodi desiderato, viene ripetuta l'intera operazione allo scopo di creare un'intera foresta di alberi generati casualmente.
Nella seconda fase l'algoritmo applica le regole degli alberi della foresta creata al passo precedente ai dati di test e per ciascun dato fornisce come risultato della predizione la classe con il voto più alto, ovvero quella rilevata il maggior numero di volte nei diversi alberi di decisione: questo tipo di algoritmo di scelta della classe di appartenenza è detto "majority voting" ed è illustrato nella figura 2.2.

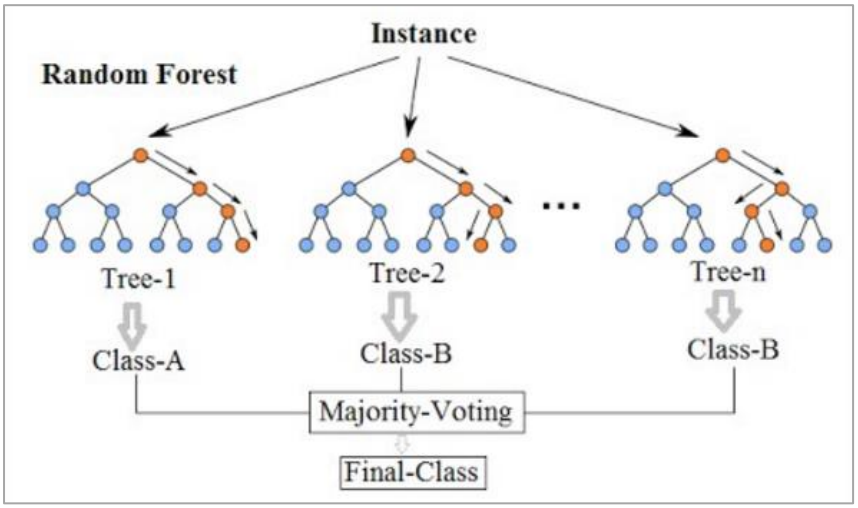

Figura 2.2: esempio di applicazione del Random Forest e relativo majority voting [7].

Un aumento del numero di alberi creati porta generalmente un miglioramento dell'accuratezza del risultato della predizione seppure renda il modello più lento dal punto di vista delle performance in quanto computazionalmente più complicato da elaborare.
Il Random Forest è apprezzato per la sua versatilità nel risolvere sia problemi di classificazione sia di regressione e la sua robustezza nei confronti di fenomeni di overfitting che non vengono tenuti in considerazione con la creazione di un numero abbastanza elevato di alberi di decisione.
Tale algoritmo inoltre, come si può leggere nel paragrafo 4.3.3 "*Feature selection*", può essere usato anche per la selezione degli attributi più importanti nel dataset a disposizione.

2.2.2 Support Vector Machine

Oltre al Random Forest, tra gli algoritmi capaci di assolvere a compiti di classificazione e regressione, vi è anche il Support Vector Machine o, più brevemente, SVM: tale algoritmo si basa sulla disposizione dei dati in uno spazio n-dimensionale (dove "n" è il numero di feature del dataset considerato) e il calcolo di un iperpiano che divida gli elementi appartenenti a classi diverse.
Come mostrato nella figura 2.3 che illustra la disposizione dei campioni per un dataset avente due feature e due classi, il calcolo dell'iperpiano si basa su due fattori principali:

margine massimo, ovvero la distanza massima tra iperpiano e i dati nello spazio n-dimensionale, e accuratezza che, brevemente, indica la precisione con cui il modello predice l'appartenenza di un dato ad una classe.

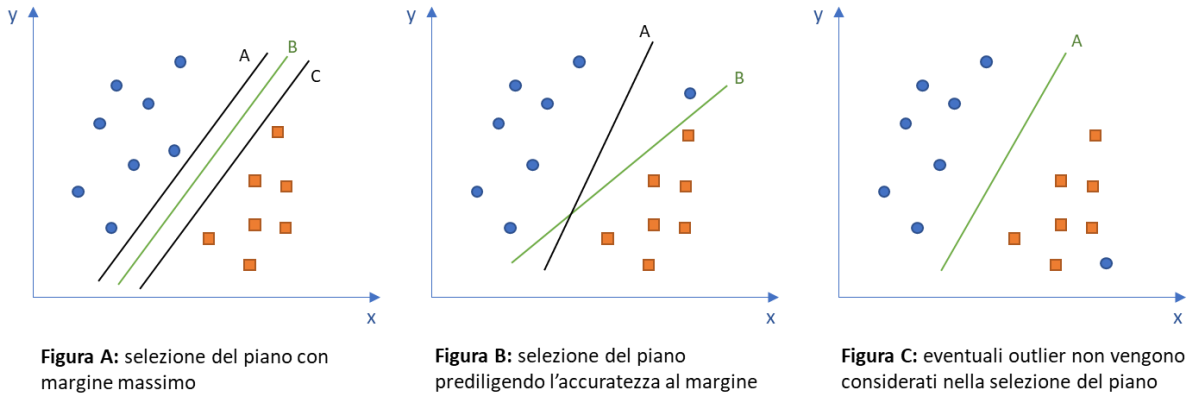

Figura A: selezione del piano con margine massimo

Figura B: selezione del piano prediligendo l'accuratezza al margine

Figura C: eventuali outlier non vengono considerati nella selezione del piano

Figura 2.3: esempio di applicazione di SVM a un generico dataset con due feature e due classi.

Dalla figura 2.3, in cui gli iperpiani scelti sono segnati in verde, si intuisce come l'algoritmo cerchi dapprima di massimizzare l'accuratezza e successivamente il margine massimo: qualora però vi siano degli outliers nel dataset, l'SVM è in grado di riconoscerli e ignorarli ai fini della predizione.
Tra le caratteristiche principali del Support Vector Machine c'è la capacità di individuare automaticamente un iperpiano che divida i dati nelle rispettive classi di appartenenza anche se queste non sono separabili linearmente: attraverso la sua funzione chiamata "*kernel trick*", l'algoritmo aggiunge un attributo fittizio (creato sulla base delle altre feature già presenti) per aumentare la dimensione dello spazio e rendere così possibile il calcolo di un iperpiano.
Un altro aspetto utile messo a disposizione dall'algoritmo è la possibilità di dare un peso alle classi di appartenenza dei dati: in molti casi infatti (come quello della ricerca qui sviluppata) la predizione su una specifica classe risulta essere più significativa rispetto ad un'altra e grazie ai diversi pesi si ha la possibilità di indirizzare i risultati del modello qualora questo non individuasse la classe in modo chiaro e netto.
Per poterne ottimizzare le performance, l'SVM è configurabile per mezzo di alcuni iperparametri tra cui il "soft-margin constant C", il tipo di "kernel" usato e il parametro "Gamma". Il primo permette di gestire il trade-off tra errori nella predizione e massimizzazione del margine: un modello con un valore di C basso tende a ignorare punti vicini all'iperpiano in quanto aumenta il margine massimo considerato per la decisione; viceversa un valore di C molto grande crea il cosiddetto "hard-margin" per cui un dato viene

sempre considerato appartenente ad una classe rispetto ad un'altra anche se molto vicino all'iperpiano calcolato, cosa che ne rende l'identificazione più dubbia.
Il parametro "kernel", invece, indica quale tipo di formula deve essere utilizzata per il calcolo dell'iperpiano e può variare tra lineare, polinomiale, sigmoidea e RBF (radial basis function): ognuna di queste ha differenti caratteristiche e complessità di elaborazione che permettono però di creare modelli affidabili per dati con distribuzioni anche molto diverse tra loro.
Per ultimo il parametro gamma, strettamente legato al kernel di tipo RBF, indica, intuitivamente, quanto lontano arriva l'influenza di un singolo punto nelle decisioni del modello: come mostrato nella figura 2.4, valori di gamma elevati tendono a creare limiti di decisione "frastagliati" in quanto molto vicini ai punti appartenenti ad una classe; al contrario valori di gamma piccoli danno maggior peso ai punti lontani dal confine di decisione che assomiglierà più ad una retta. Valori troppo elevati di gamma generano il cosiddetto problema dell'overfitting, spiegato nel paragrafo 4.3.3 "*Feature selection*".

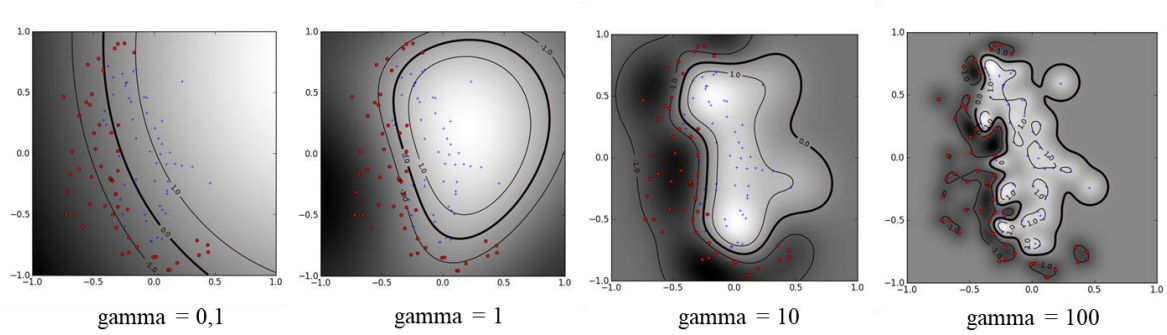

Figura 2.4: rappresentazione di un generico modello SVM al variare del parametro gamma [8]

2.2.3 Neural network

Una rete neurale, o artificial neural network, è un modello matematico vagamente ispirato alle reti neurali biologiche che compongono il cervello umano di cui cerca di imitarne i processi di apprendimento e quelli decisionali. Più in dettaglio, il modello è costituito da un numero variabile di interconnessioni tra gli elementi costitutivi fondamentali chiamati neuroni, o "percettroni", che hanno il compito di elaborare dei dati in input e trasmetterne il risultato allo strato di neuroni successivo (detto "hidden layer") fino ad ottenere il risultato finale come output del modello matematico.

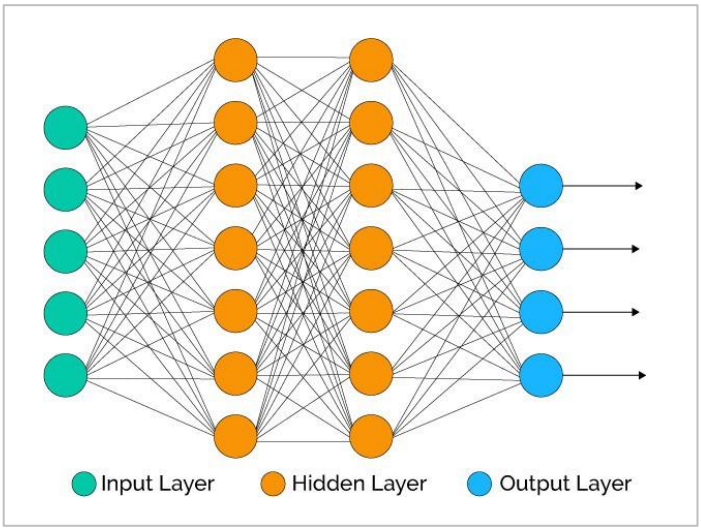

Figura 2.5: rete neurale generica con due strati di neuroni [9]

Ciascun singolo neurone ha un modo di operare molto semplice: calcola la somma pesata di tutti i suoi dati di input $\vec{x}$ usando un vettore dei pesi $\vec{w}$ (insieme con un termine aggiuntivo w_0) e applica una funzione di attivazione σ al risultato.

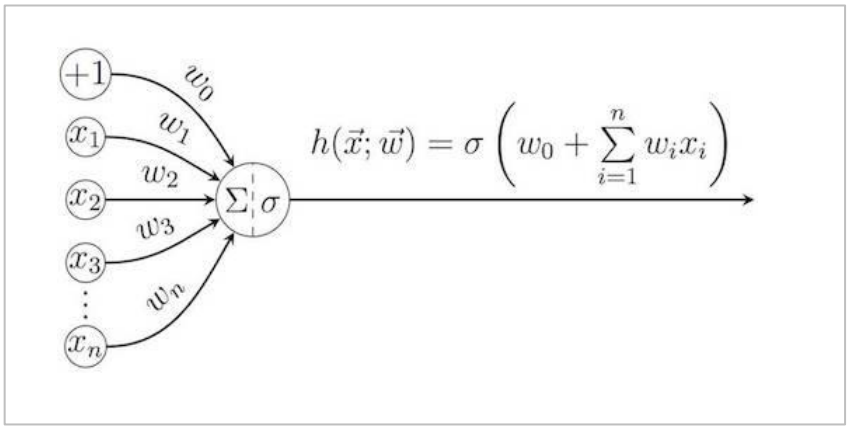

Figura 2.6: calcolo dell'output di un neurone [10]

Tra le possibili funzioni di attivazione si possono trovare le seguenti:

- identity function: $f(x) = x$
- logistic sigmoid function: $f(x) = \frac{1}{1+e^{-x}}$
- tanh: $f(x) = tanh(x)$
- rectified linear o ReLu: $f(x) = max(0, x)$

L'apprendimento del modello matematico è basato sull'aggiornamento dei pesi $\vec{w}$ ad ogni passaggio ad un nuovo hidden layer per mezzo della minimizzazione della cosiddetta "*loss function*": tra le varie disponibili vi è per esempio, la "*squared error loss function*" definita come il quadrato della differenza tra l'etichetta reale del dato di training considerato e quella calcolata per mezzo dei pesi dal neurone.
Anche il modo con cui si minimizza la loss function dipende dal tipo di algoritmo scelto: tra questi vi è per esempio il "gradient descent", che aggiorna il vettore dei pesi considerando quelli che permettono di avere la diminuzione maggiore della loss function, oppure "L-BFGS" (Limited-memory BFGS) che considera il migliore tra gli ultimi valori citati prima.
Gli algoritmi finora citati fanno parte degli iperparametri configurabili in fase di inizializzazione del modello. Altri iperparametri sono:

- *batch size*: numero di dati che vengono propaganti nella rete durante il training;
- *hidden layer size*: numero di neuroni per ciascun hidden layer;
- *max iteration*: numero di hidden layers nella rete;

2.2.4 Metodi di valutazione dei modelli predittivi

Per il confronto dei risultati ottenuti per mezzo dell'applicazione dei modelli matematici ai dati campionati tramite Oracle Enterprise System Manager si sono utilizzati parametri differenti che hanno permesso la valutazione di aspetti diversi dell'analisi svolta.
Tra questi vi sono i seguenti:

- accuracy
- precision
- recall
- f1-score
- confusion matrix

Il primo dei parametri citati indica rapporto tra il numero di predizioni corrette e il totale delle predizioni effettuate: tale parametro aiuta, nella fase di ottimizzazione dell'algoritmo, a capire se la modifica del modello porta miglioramenti alle performance o meno.
In un dataset particolarmente sbilanciato come quello disponibile per la ricerca, però, il valore dell'accuratezza tende ad essere poco significativo: in questo caso, infatti, il numero di predizioni corrette è fortemente influenzato dalla presenza dominante della classe maggioritaria, motivo per il quale il numero totale di predizioni corrette può essere alto nonostante quelle relative alla classe minoritaria siano per lo più sbagliate [11].
Nel caso si analizzi un dataset sbilanciato, quindi, si ricorre ad altri parametri di valutazione come la "precision" e la "recall": entrambi i valori vengono calcolati sui risultati di una

singola classe e permettono quindi di valutare la bontà di un modello a prescindere dall'eventuale sbilanciamento del dataset di input. Per definire formalmente questi due parametri è necessario introdurre però alcuni termini utili che descrivono i quattro possibili risultati di una predizione di un classificatore binario (le cui classi sono definite come "positiva" e "negativa") su un campione di test:

- True positive (TP): campione positivo classificato correttamente dal modello
- True negative (TN): campione negativo classificato correttamente dal modello
- False positive (FP): campione negativo erroneamente classificato come positivo
- False negative (FN): campione positivo erroneamente classificato come negativo

In questo senso la precision e la recall sono definite come:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Dal punto di vista prettamente intuitivo si può definire, quindi, la precision come il valore che indica quanti campioni, tra tutti quelli classificati dal modello come appartenenti a una certa classe, sono stati correttamente etichettati; la recall indica, invece, quanti campioni sono stati etichettati correttamente tra tutti quelli appartenenti ad una certa classe presenti nell'intero dataset.
Entrambi i parametri sono ottimali quanto più si avvicinano a 1 poiché, in questo caso, il numero di falsi positivi e falsi negativi (cioè generalmente il numero di predizioni errate) tende a 0.

L'F1-Score, o coefficiente di Dice, combina precision e recall facendone la media armonica tramite la seguente formula matematica:

$$F1 = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = 2\,\frac{precision \cdot recall}{precision + recall}$$

Può assumere tutti i valori compresi tra 0 e 1 ed è ottimale quanto più il suo valore è elevato: come si nota dalla formula, infatti, l'F1-Score è tanto più alto quanto più la recall e la precision si avvicinano a 1, valore ottimale per entrambi i parametri.

Altro parametro importante utilizzato nel corso della ricerca e strettamente legato ai concetti definiti con la precision e la recall, è la "confusion matrix" ("matrice di confusione"): come mostrato nella figura 2.7, la matrice consente la visualizzazione esplicita del numero di campioni correttamente ed erroneamente classificati dal modello. Affinché questo sia il più performante possibile il numero di falsi positivi e falsi negativi deve essere il minore possibile.

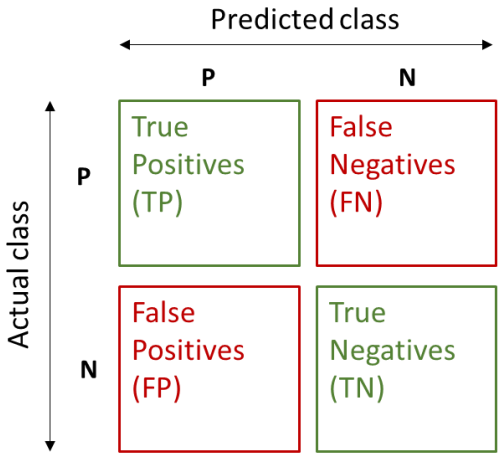

Figura 2.7: matrice di confusione.

2.3 Strumenti utilizzati

Per lo sviluppo della ricerca sono stati sfruttati vari strumenti tecnologici utili per l'acquisizione e la successiva analisi dei dati: tra questi si è scelto di usare come linguaggio di programmazione Python che, grazie alle librerie messe a disposizione dal pacchetto open source "Anaconda", consente di avere a disposizione una grande quantità di algoritmi dedicati al machine learning. Sono stati quindi usati i pacchetti "Scikit-Learn", per l'applicazione dei modelli predittivi, e "PonyORM", per la comunicazione con un database remoto.
Altre librerie sfruttate sono "MatplotLib", per la visualizzazione dei risultati tramite grafici, e "Imblearn", per l'oversampling dei dati durante la fase di preprocessing
I dati analizzati sono stati invece raccolti grazie all'utilizzo del software "Oracle Enterprise Manager" nella versione 13c: tale programma, come spiegato più in dettaglio nel capitolo 3 *"Oracle Enterprise System Manager"*, effettua il campionamento di varie grandezze di interesse dei sistemi informatici osservati e le salva all'interno di un repository interno costituito da un database Oracle.

3 Oracle Enterprise System Manager

Ai fini della raccolta dei dati utili alla ricerca è stato sfruttato uno dei prodotti commerciali usati nel contesto aziendale studiato e messo a disposizione da Oracle Corporation, ovvero Oracle Enterprise System Manager 13c: tale software sfrutta le funzionalità del database Oracle come repository interno per memorizzare i dati campionati dai sistemi monitorati.
Questo capitolo vuole quindi offrire un quadro generale delle funzionalità dei programmi citati e degli aspetti più rilevanti che hanno influito nella trattazione della ricerca.

3.1 Cosa è l'Oracle Enterprise Manager

Come cita il sito Oracle.com, "*Oracle Enterprise Manager is Oracle's on-premises management platform, providing a single pane of glass for managing all of a customer's Oracle deployments, whether in their data centers or in the Oracle Cloud. Through deep integration with Oracle's product stack, Enterprise Manager provides market-leading management and automation support for Oracle applications, databases, middleware, hardware and engineered systems."* [12]
Dalla definizione che ne da il produttore stesso si denota la capacità da parte del software di centralizzare il controllo di varie tecnologie all'interno di una stessa piattaforma per dare la possibilità agli operatori impiegati di disporre dei dati essenziali per il loro monitoraggio.

Come illustrato in figura 3.1, l'architettura di tale piattaforma è strutturata in più livelli: all'interno di ogni host su cui si vogliono effettuare operazioni di monitoraggio vi è installato un Management Agent, un particolare software che rimane in ascolto dei dati inviati dai Plugins. Questi ultimi sono moduli specifici per ogni tipo di tecnologia (*target*) gestibile da OEM e, con l'installazione di default, permettono di raccogliere i dati dai seguenti prodotti:

- Oracle Database
- Oracle Fusion Middleware
- Oracle Exadata
- Oracle Cloud Framework
- Oracle System Infrastructure

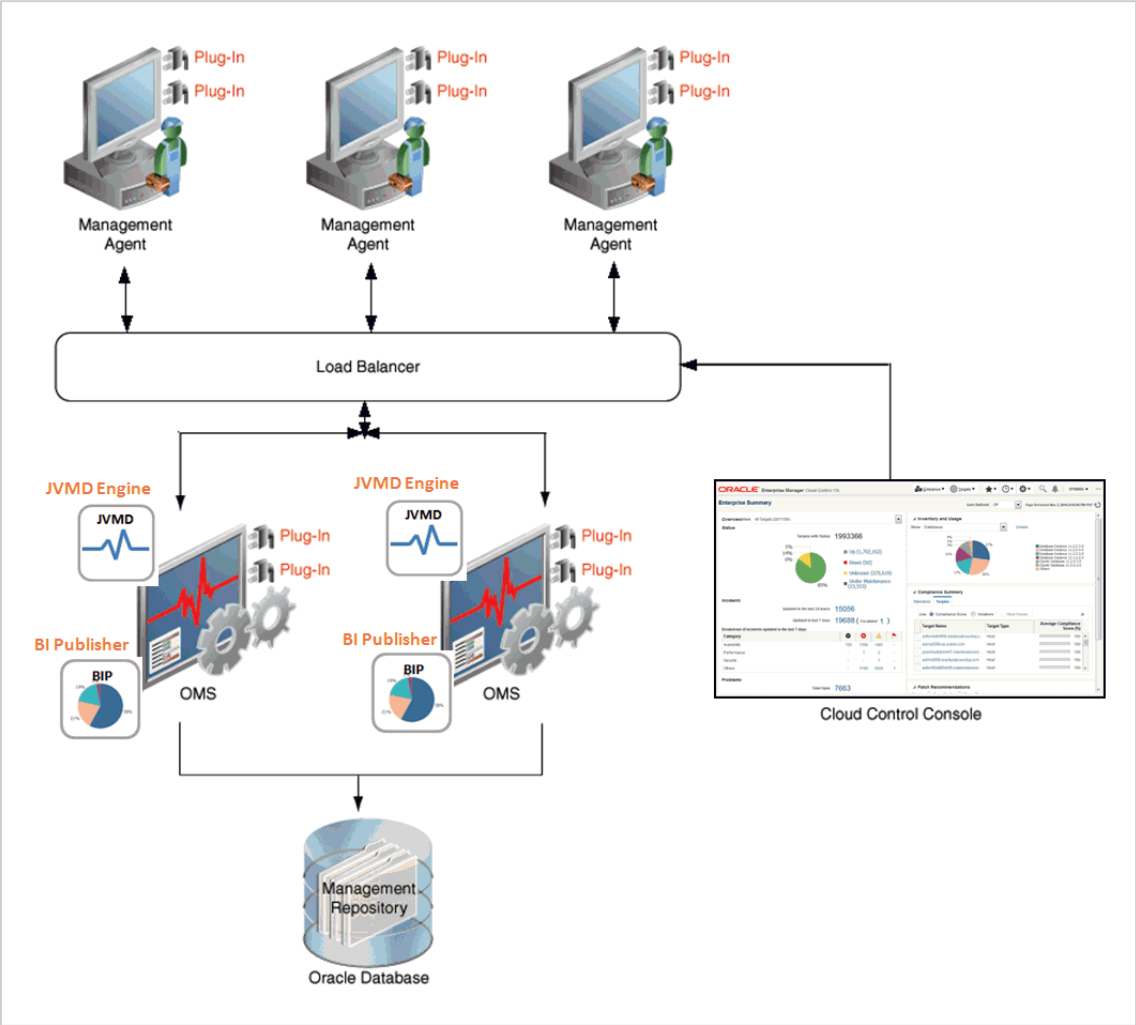

Figura 3.1: architettura dell'Oracle Enterprise Manager [13]

Più internamente la piattaforma è composta da uno o più JVDM Engine (Java Virtual Machine Diagnostics Engine), associati ad altrettanti OMS (Oracle Management Repository) messi in comunicazioni con gli Agents tramite un Load Balancer che consente una migliore gestione del traffico di rete: il JVDM Engine e l'OMS si occupano insieme di calcolare le statistiche relative ad ogni target e memorizzarle all'interno del repository, mostrato in figura 3.1 con il nome di Management Repository.
Quest'ultimo è il componente principale dell'infrastruttura citata, utilizzato per lo svolgimento della ricerca: è costituito da un database Oracle e quindi da oggetti quali jobs, packages, procedure, viste e tabelle.

Come si può leggere in dettaglio nel paragrafo successivo dedicato al repository in questione, l'OEM mette a disposizione una grande quantità di dati, relativi ai target selezionati, campionati con frequenze di aggiornamento differenti: tali dati prendono il nome di "metriche".
Di default ogni metrica viene associata ad uno o più soglie che permettono di scatenare un allarme di gravità crescente a seconda del livello di superamento della stessa e consentono quindi di intraprendere azioni correttive automatiche o guidate dall'amministratore del sistema.

3.2 Il Management Repository

Come precedentemente scritto, il database utilizzato dall'Oracle Enterprise Manager, detto anche Management Repository, è la principale fonte sfruttata durante la ricerca per l'estrazione dei dati necessari alle analisi.
Questo paragrafo vuole quindi illustrare le caratteristiche più importanti di tale componente e le funzionalità che hanno influito sullo svolgimento del lavoro.

3.2.1 I targets

I target, nella terminologia di Oracle Enterprise Manager, sono sistemi informatici, software e hardware, di cui si può effettuare il monitoraggio.
Tali entità possono essere configurate manualmente per essere gestite tramite la piattaforma o se ne può effettuare il cosiddetto processo di autodiscovery che permette di aggiungere l'entità alla lista dei "managed target".
Tra i possibili target si trovano le seguenti famiglie di sistemi (target type):

- Oracle Database tra cui Oracle RAC e Oracle ASM
- Oracle Fusion Middleware comprendente la suite di sistemi Oracle WebLogic
- Oracle Exadata
- Oracle Cloud Framework
- Oracle System Infrastructure per la gestione dei sistemi hardware

Ad ognuno dei target citati è associato un software chiamato Plug-in che permette la comunicazione dei dati campionati (metriche) al Management Agent: tale servizio viene installato sull'host (fisico o virtuale) su cui risiedono i target che si vogliono monitorare. Un eventuale problema all'Agent non permette quindi di recuperare le informazioni relative al

target e sarà etichettato dall'OEM con lo stato "Agent Down", come meglio spiegato nel paragrafo 3.2.3 "*Gli stati*".
Per la ricerca, visto il contesto aziendale studiato (vedi paragrafo 4.1 "*Individuazione dei dati di interesse*"), sono stati analizzati i dati relativi ai target delle famiglie Oracle Database, Oracle Fusion Middleware e Oracle System Infrastructure.

3.2.2 Le metriche

Al variare del tipo di target, Oracle Enterprise Manager mette a disposizione un set di informazioni, dette "metriche", ricavate dal campionamento di alcuni dati di interesse: questi vengono recuperati con frequenza variabile dall'esecuzione di alcuni comandi di sistema da parte del Management Agent sull'host in cui è installato e risultano di fondamentale importanza per la gestione degli alert. Ad ogni metrica infatti possono essere associate una o più soglie che, una volta superate, scatenano l'invio di un alert di gravità diversa (sotto forma di email, chiamata telefonica o altro) verso l'amministratore di sistema.
Vista l'ampia scelta di metriche disponibili all'interno del Management Repository, è stato scelto di utilizzare, nella maggior parte dei casi, quelle espresse in percentuale per permettere una migliore comparazione tra i sistemi analizzati che hanno parametri di configurazione differenti.
Come dettagliato nelle tabelle sottostanti, infatti, sono state considerate metriche differenti per i vari tipi di target studiati, analizzando per ognuno di questi le famiglie di dati che nell'esperienza lavorativa risultano più importanti.

Tabella 3.1. Metriche usate per il dataset relativo ai target di tipo "WebLogic". [14]

Nome metrica	Descrizione	Unità di misura	Frequenza di campionamento
connWait_perc	Connection Requests that Waited	Percentuale	15 minuti
connWaitSuccesses_perc	Connection Wait Successes	Percentuale	15 minuti
cpuUsage_perc	CPU Usage	Percentuale	15 minuti
ejbCacheHit_perc	Cache Hits	Percentuale	15 minuti
ejbPoolAccessSuccess_perc	Bean Successes	Percentuale	15 minuti
heapUsedPerc_value	Heap Usage	Percentuale	15 minuti
stmtCacheSatisfied_perc	Cached Statements Used	Percentuale	15 minuti

Come si può notare dalla tabella 3.1 e da quelle sottostanti, ogni metrica, a prescindere dal tipo di target a cui si riferisce, è soggetta ad una frequenza di campionamento diversa che, eventualmente, può essere personalizzata dall'amministratore di sistema.

Tabella 3.2. Metriche usate per il dataset relativo ai target di tipo "Database". [15]

Nome metrica	Descrizione	Unità di misura	Frequenza di campionamento
dumpUsedPercent	Dump Area Used	Percentuale	30 minuti
redosize_ps	Redo Generated	Per secondo	5 minuti
usercall_pct	User Calls	Percentuale	1 minuto
tabscanstotal_ps	Total Table Scans	Per secondo	10 minuti
bufcachehit_pct	Buffer Cache Hit	Percentuale	10 minuti
libcache_hit_pct	Library Cache Hit	Percentuale	10 minuti
dictionaryhit_pct	Data Dictionary Hit	Percentuale	10 minuti
inmem_sort_pct	Sorts in Memory	Percentuale	1 minuto
redologalloc_hit_pct	Redo Log Allocation Hit	Percentuale	5 minuti
cursorcachehit_pct	Cursor Cache Hit	Percentuale	10 minuti
cpuusage_ps	CPU Usage	Per secondo	10 minuti
cpu_time_pct	Database CPU Time	Percentuale	10 minuti
libcache_miss_pct	Library Cache Miss	Percentuale	10 minuti
pgacachehit_pct	PGA Cache Hit	Percentuale	10 minuti
dictionarymiss_pct	Row Cache Miss Ratio	Percentuale	10 minuti
shared_free_pct	Shared Pool Free	Percentuale	15 minuti
large_free_pct	Large Pool Free	Percentuale	15 minuti
pctUsed	Tablespace Space Used	Percentuale	10 minuti

Tabella 3.3. Metriche usate per il dataset relativo ai target di tipo “Host”. [16]

Nome metrica	Descrizione	Unità di misura	Frequenza di campionamento
cpuLoad	Run Queue Length (5 min avg,per core)	Scalare	5 minuti
noOfProcs	Total Processes	Scalare	5 minuti
noOfUsers	Total Users	Scalare	5 minuti
swapUtil	Swap Utilization	Percentuale	5 minuti
memUsedPct	Memory Utilization	Percentuale	5 minuti
memfreePct	Free Memory	Percentuale	5 minuti
logicMemfreePct	Free Logical Memory	Percentuale	5 minuti
cpuUtil	CPU Utilization	Percentuale	15 minuti
cpuUser	CPU in User Mode	Percentuale	5 minuti
cpuKernel	CPU in System Mode	Percentuale	5 minuti
cpuIOWait	CPU in I/O Wait	Percentuale	5 minuti
usedLogicalMemoryPct	Used Logical Memory	Percentuale	5 minuti
DiskActivitybusy	Disk Activity Busy	Percentuale	5 minuti

3.2.3 Gli stati

Al variare delle condizioni dell’infrastruttura i target possono trovarsi nei seguenti stati:

- Blackout
- Pending/Unknown
- Metric Error
- Agent Down
- Target Down
- Target Up

Ciascuno di questi descrive lo stato in cui il target monitorato si trova e permette agli amministratori del sistema di capire se vi sia un problema o meno: in questo senso, all’interno del database, ognuno degli stati sopracitati è accompagnato da un campo che indica la data di inizio del cambio di stato ed uno che ne indica la fine (campo eventualmente impostato a *null* in caso lo stato sia ancora attivo).

Di tali stati, chiamati "availability status", l'Agent Down indica un problema di comunicazione con il Management Agent, causato da un problema di rete o un problema interno al software che gestisce l'agent: in questo caso l'amministratore di sistema è chiamato a risolvere il problema in quanto non è possibile monitorare lo stato del target se l'agent non è più disponibile. Gli stati "Metric Error" e "Pending/Unknown" indicano invece rispettivamente un problema nel calcolo delle statistiche relative alla metrica scelta per il target e uno stato indefinito dovuto a svariate cause come la non corretta configurazione di un agent.
Qualora si debbano effettuare delle operazioni programmate su un target è possibile, inoltre, forzarne lo stato a "Blackout" che permette di sospendere le notifiche relative agli alert durante la fase di manutenzione.
Per lo svolgimento della ricerca sono stati considerati gli stati "Target Up" e "Target Down" per i target relativi ai Database (Oracle Database) e WebLogic (Oracle Fusion Middleware) che indicano il corretto funzionamento o meno del target considerato. Diverso è il caso dei target Host (Oracle System Infrastructure) per cui sono stati considerati gli stati "Target Up" e "Agent Down" in quanto, qualora un host non funzionasse correttamente (come per esempio in caso di spegnimento dell'host), anche il Management Agent installato sulla macchina genererebbe un errore.
Come mostrato nella figura 3.2, nel contesto aziendale studiato la distribuzione degli stati per i diversi target considerati è differente ed è legata al modo con cui essi vengono gestiti e si comportano all'interno dell'infrastruttura tecnologica.

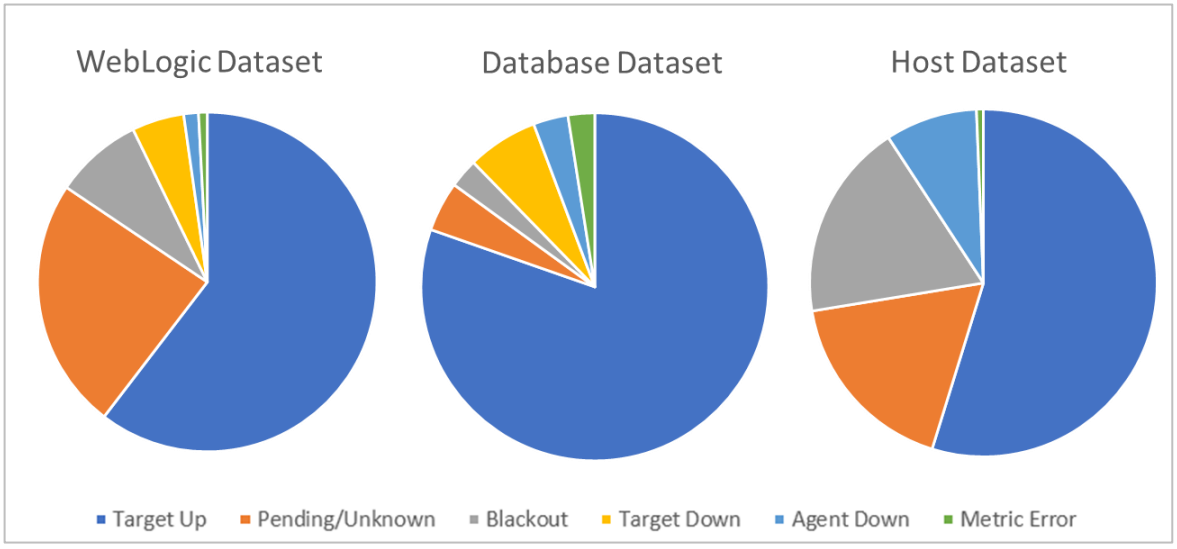

Figura 3.2: distribuzione degli stati per i diversi target considerati nella ricerca

Tale distribuzione ha influito nell'analisi dei dati, come spiegato nel paragrafo 4.3.2 "*Gestione sbilanciamento classi*": sono stati infatti adottati opportuni provvedimenti per bilanciare la suddetta distribuzione ai fini dell'ottimizzazione dei modelli matematici usati.

3.2.4 Le viste utilizzate

L'installazione di Oracle Enterprise Manager e del relativo Management Repository permette l'interrogazione di un insieme di viste messe a disposizione da Oracle per la fruizione dei dati raccolti, indipendentemente dalle tabelle relazionali sottostanti che potrebbero subire modifiche con eventuali upgrade della piattaforma: ciò consente agli sviluppatori terzi di creare applicazioni la cui compatibilità con il database è garantita nel tempo.
Nello sviluppo della ricerca sono state utilizzate le seguenti viste (i cui schema sono illustrati in figura 3.3) su cui sono state poi effettuate delle particolari query, descritte nel paragrafo 4.2 "*Raccolta dei dati*", per recuperare solo i dati utili agli scopi del lavoro:

- MGMT$AVAILABILITY_HISTORY
- MGMT$METRIC_HOURLY

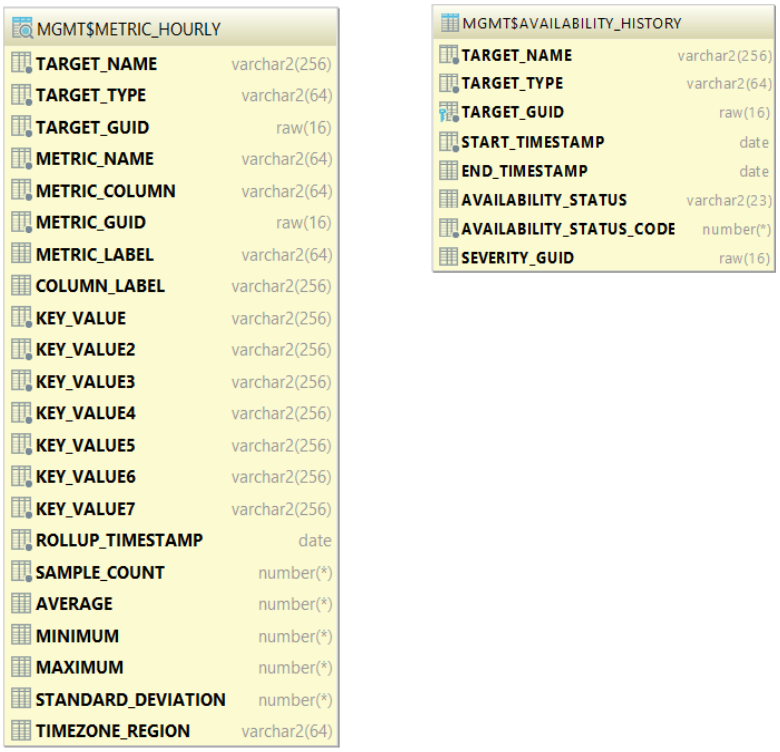

Figura 3.3: schema viste MGMT$AVAILABILITY_HISTORY e MGMT$METRIC_HOURLY

La prima contiene per ogni record della vista uno stato del target (attributo AVAILABILITY_STATUS) il quale è identificato univocamente dagli attributi TARGET_GUID, TARGET_NAME e TARGET_TYPE.
L'attributo START_TIMESTAMP riconosce invece la data di inizio dello stato (quindi quello in cui c'è stato un cambio di stato) mentre END_TIMESTAMP quello di fine: quest'ultimo può eventualmente essere non disponibile (*null*) qualora lo stato indicato sia ancora attivo.

	TARGET_NAME	TARGET_TYPE	TARGET_GUID
97	[redacted]_Service	oracle_oms	[redacted]
98	[redacted]_Service	oracle_oms	[redacted]
99	[redacted]02_1	weblogic_nodemanager	[redacted]

	START_TIMESTAMP	END_TIMESTAMP	AVAILABILITY_STATUS	AVAILABILITY_STATUS_CODE	SEVERITY_GUID
97	2018-03-26 11:40:44	2018-03-26 11:41:16	Pending/Unknown	6	684E556C84333649E053...
98	2018-03-26 11:41:16	*null*	Target Up	1	*null*
99	2017-09-12 17:09:55	2017-10-11 09:33:34	Target Up	1	*null*

Figura 3.4: esempio di tre record della vista MGMT$AVAILABILITY_HISTORY

La vista MGMT$METRIC_HOURLY invece contiene, per ogni record, i valori aggregati per ora dei campionamenti di una specifica metrica per un dato target.
Come illustrato nella figura 3.5, infatti, sono presentii campi METRIC_NAME e METRIC_COLUMN che identificano rispettivamente la metrica di riferimento e la relativa famiglia di metriche a cui appartiene; come nella vista precedentemente illustrata, i campi TARGET_NAME, TARGET_GUID e TARGET_TYPE sono relativi al target identificato univocamente e sono stati utilizzati come chiavi per le join tra le due viste citate, come spiegato nel paragrafo 4.2 "*Raccolta dei dati*"; l'attributo ROLLUP_TIMESTAMP indica invece la data e l'ora di aggregazione dei dati.

	TARGET_NAME	TARGET_TYPE	TARGET_GUID	METRIC_NAME	METRIC_COLUMN
2978	[redacted]fi	host	[redacted]	Load	cpuQLen

	METRIC_GUID	METRIC_LABEL	COLUMN_LABEL	KEY_VALUE	KEY_VALUE2
2978	8E38B49C40C71F8B6B2507457FEF2E58	Load	CPU Queue Length		

	KEY_VALUE3	KEY_VALUE4	KEY_VALUE5	KEY_VALUE6	KEY_VALUE7	ROLLUP_TIMESTAMP
2978						2018-03-20 04:00:00

	SAMPLE_COUNT	AVERAGE	MINIMUM	MAXIMUM	STANDARD_DEVIATION	TIMEZONE_REGION
2978	12	0.08333333333...	0	1	0.28867513459481288225...	Europe/Berlin

Figura 3.5: esempio di un record della vista MGMT$METRIC_HOURLY

Per ultimi, i campi AVERAGE, MINIMUM e MAXIMUM indicano rispettivamente il valore medio, minimo e massimo tra quelli campionati nell'ora di riferimento: per lo sviluppo della ricerca si è deciso di utilizzare il valore massimo come indicativo del caso peggiore qualora si fosse verificata una anomalia nel target analizzato.

3.2.5 Retention policy e aggregazione dei dati

Con lo scopo di ridurre il numero di record presenti all'interno del Management Repository e ottimizzarne le performance, Oracle Enterprise Manager adotta delle politiche di aggregazione dei dati e di retention policy che hanno influenzato in maniera attiva lo sviluppo della ricerca: la piattaforma infatti raccoglie i dati appena campionati all'interno della tabella GC_METRIC_VALUES e li rielabora periodicamente per metterli a disposizione dei programmatori esterni tramite alcune viste.
Tra queste troviamo le seguenti:

- MGMT$METRIC_CURRENT;
- MGMT$METRIC_HOURLY;
- MGMT$METRIC_DAILY:

rispettivamente permettono la visualizzazione degli ultimi dati campionati, di quelli aggregati per ora e per giorno [17].
La politica di eliminazione dei dati è affidata alla cosiddetta "retention policy" che, di default, prevede l'eliminazione dei dati "crudi" (ovvero quelli non ancora rielaborati e aggregati nelle viste) ogni 24 ore. Allo scadere di questo periodo di tempo i record vengono aggregati per ora e per giorno e sono visibili nelle ultime due viste sopracitate.
Gli aggregati orari così ottenuti sono quindi disponibili per 31 giorni dalla creazione mentre quelli giornalieri rimangono memorizzati per 24 mesi prima della loro definitiva eliminazione.
Viste le necessità di avere un numero abbastanza grande di dati per effettuare una analisi esauriente del sistema e dei valori che ne descrivessero lo stato in un intorno di tempo sufficientemente puntale relativamente al cambio di stato descritto, si è scelto quindi di sfruttare le metriche aggregate per ora che sono state esportate a partire dal 01/01/2018.
La vista MGMT$METRIC_HOURLY garantisce in questo senso la precisione sullo stato del sistema poiché permette di estrapolare il valore massimo che la metrica studiata ha assunto nell'intorno di tempo compreso tra i 30 minuti prima e 30 minuti dopo che è avvenuto un cambio di stato, come descritto più approfonditamente nel paragrafo 4.2 "*Raccolta dei dati*".

4 Data selection e preprocessing

Dato il contesto innovativo a cui lo studio si rivolge per l'applicazione dei modelli predittivi, la prima fase essenziale per lo sviluppo della ricerca riguarda la selezione dei dati utili allo scopo e la loro rielaborazione per una migliore performance dei suddetti algoritmi.
In questo senso il capitolo illustra le scelte che sono state effettuate durante il lavoro svolto a partire dall'individuazione dei dati di interesse per la ricerca e la loro raccolta tramite il Management Repository. In secondo luogo, vengono descritte le tecniche usate per la rielaborazione dei dataset così creati e quindi tutta la fase comunemente detta di "preprocessing dei dati".

4.1 Individuazione dei dati di interesse

Come visto nel paragrafo 1.1 "*Obiettivi della ricerca*", si è voluto approfondire la possibilità di dare agli operatori impiegati nel monitoraggio delle infrastrutture informatiche un supporto ulteriore per il loro lavoro: in questo senso si è analizzato il contesto aziendale con l'intento di ottimizzare alcune delle situazioni più critiche.
Tra queste risulta evidente l'elevato numero di alert che mediamente vengono scatenati in un mese su tutti i sistemi monitorati (vedi figura 1.3): si è quindi indirizzata la ricerca verso quei tipi di target la cui attività genera il maggior numero di avvisi considerando tra questi solo quelli di tipo "Critical" e "Warning", due soglie rispettivamente più e meno gravi di alert.

Il grafico nella figura 4.1 illustra il numero di alert scatenati da tutti i tipi di target negli ultimi due anni di attività dell'Oracle Enterprise System Manager.
Nonostante il numero elevato di alert scatenati, non è stato considerato il target di tipo "generic_service" in quanto poco utile ai fini della ricerca: i target appartenenti a questo tipo, infatti, vengono creati manualmente dagli amministratori della piattaforma e consentono di verificare periodicamente il funzionamento di sistemi di natura diversa tra loro. Può essere utilizzato, per esempio, per effettuare delle richieste HTTP o IMAP su un web service al fine di verificarne il suo stato.

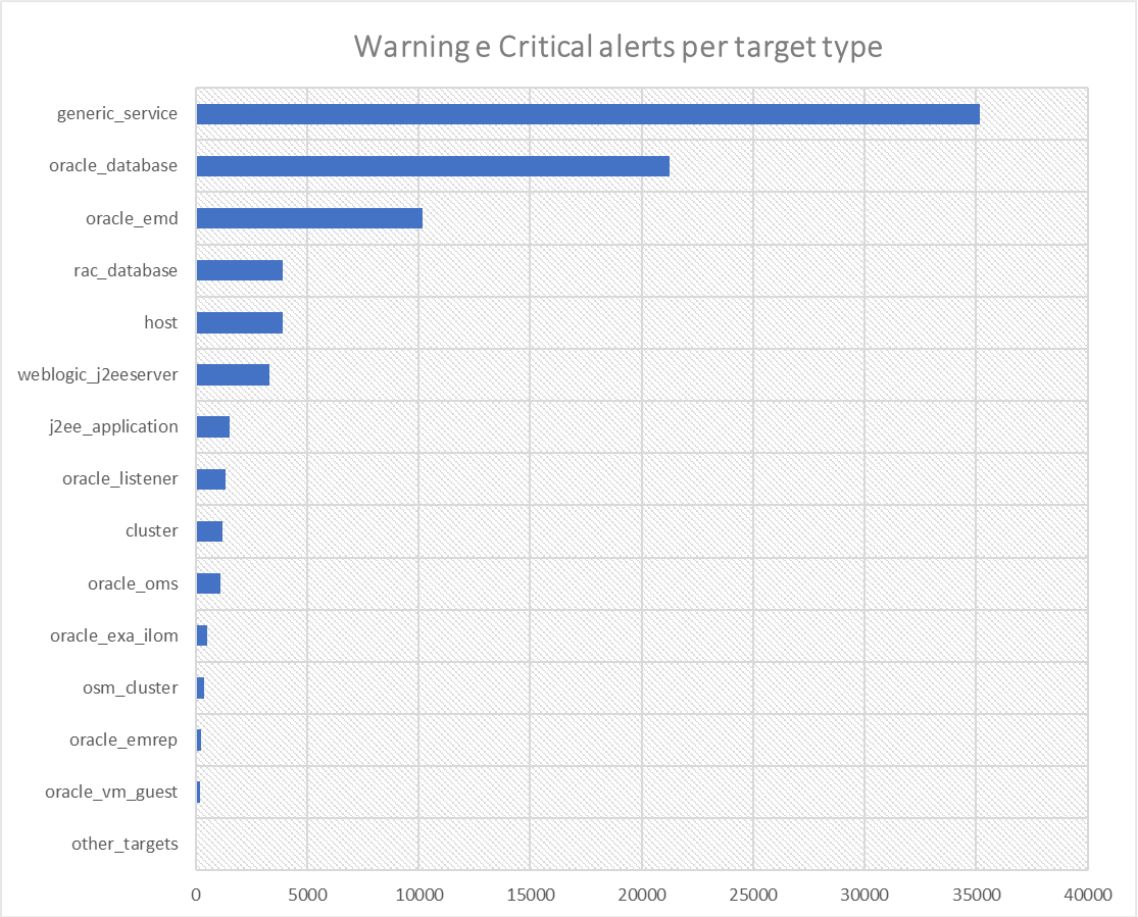

Figura 4.1: numero totale di alert di tipo Critical e Warning per tipo di target.

Come si evidenzia dalla figura 4.1 alcuni target type risultano più significativi per numero di eventi da gestire: per questo motivo la ricerca si è concentrata sulle famiglie di target esplicitate nella tabella 4.1.

Tabella 4.1. tipi di target studiati nella ricerca.

Famiglia di target	Target Type
Oracle Fusion Middleware	weblogic_j2eeserver weblogic_cluster weblogic_nodemanager
Oracle Database	oracle_database rac_database
Oracle System Infrastructure	host

Insieme allo studio degli alert scatenati è stata effettuata una analisi del numero di stati che i vari target assumono nel tempo: il grafico in figura 4.2 mostra il numero di stati totali assunti dai vari target raggruppati per tipo di stato.

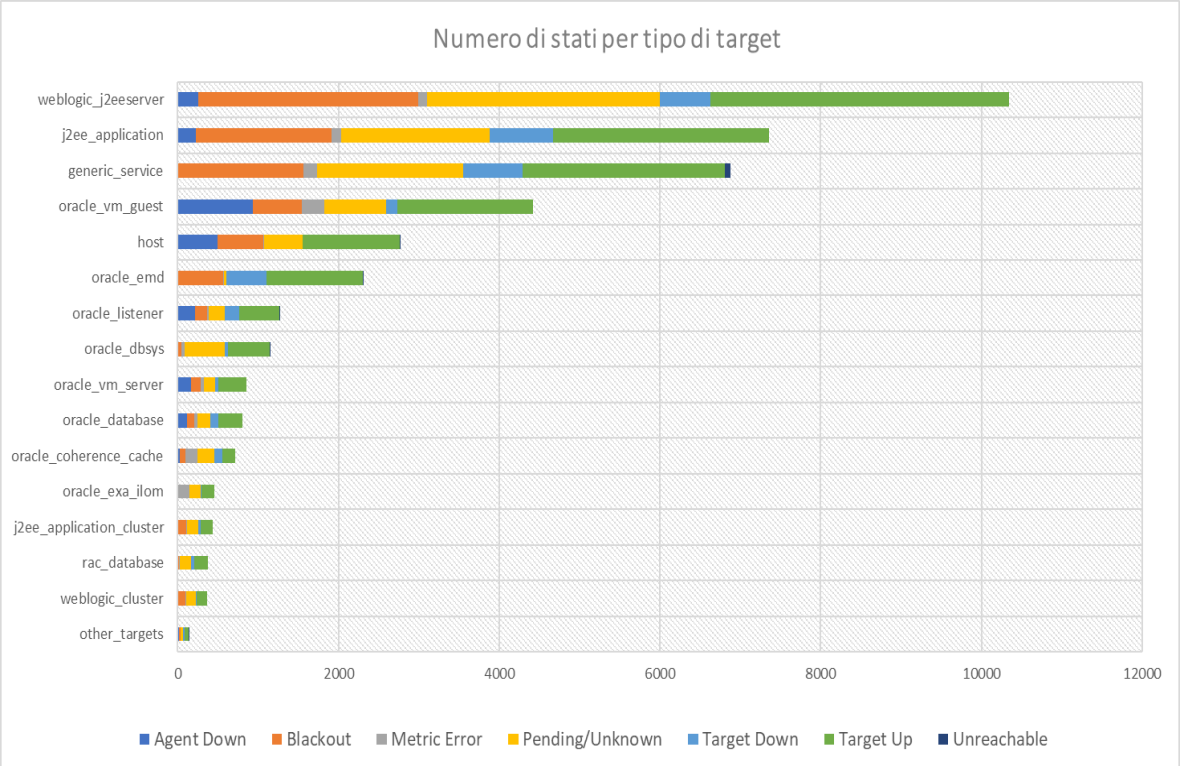

Figura 4.2: numero di stati per tipo di target. (Dati dal 01/01/2016 al 13/04/2018)

Anche dal precedente grafico, come già visto nel paragrafo 3.2.3 "*Gli stati*", si può notare come, per i tipi di target scelti, la distribuzione degli stati sia generalmente sbilanciata: tale fenomeno ha portato quindi all'utilizzo di tecniche apposite per ridurne gli effetti sull'applicazione dei modelli predittivi scelti (vedi paragrafo 4.3.2 "*Gestione sbilanciamento classi*").
Infine, poiché i dati utilizzati sono aggregati per ora, si è deciso di considerare le metriche relative ad un cambio di stato (in particolare "Target Up", "Target Down" e "Agent Down" come spiegato nel sopracitato paragrafo), campionate nei 30 minuti precedenti e 30 minuti successivi all'avvenuto cambio.

4.2 Raccolta dei dati

Per la raccolta dei dati individuati sono stati tenuti separati i tipi di target studiati con l'obiettivo di creare tre dataset differenti: il primo chiamato genericamente "Database Dataset", il secondo "WebLogic Dataset" e l'ultimo "Host Dataset". Come si può notare dal codice delle query eseguite sul Management Repository, le metriche sono state raggruppate per fascia oraria tramite l'utilizzo dell'operatore PIVOT allo scopo di ottenere per ogni record tutte le metriche raccolte dentro l'intervallo di tempo di un'ora dal cambio di stato.
Delle metriche usate si è quindi scelto di sfruttare il valore massimo dell'aggregazione effettuata da Oracle Enterprise Manager in modo da considerare il valore più significativo per il target studiato.

I tre dataset ottenuti con l'esecuzione delle query descritte nei listati sottoelencati, hanno caratteristiche diverse: in questo senso la tabella 4.2 ne dettaglia la cardinalità, elemento che ha influito nella scelta di alcune tecniche di preprocessing rispetto ad altre.
I numeri tengono conto anche degli stati non considerati nelle analisi successive:

Tabella 4.2. Cardinalità e numero di features dei dataset creati.

Dataset	Cardinalità	Numero di features
WebLogic	6396 records	7
Database	926 records	18
Host	2041 records	13

I seguenti listati descrivono in dettaglio le tre query utilizzate per la creazione dei tre dataset:

Listato 4.1 Query per la creazione del Database Dataset.

```
SELECT
 RES_MH.TARGET_GUID                AS TARGET_GUID,
 MAX(RES_MH.TARGET_NAME)           AS TARGET_NAME,
 MAX(RES_MH.TARGET_TYPE)           AS TARGET_TYPE,
 MAX(AH.AVAILABILITY_STATUS)       AS AVAILABILITY_STATUS,
 AH.START_TIMESTAMP                AS START_TIMESTAMP,
 MAX(AH.END_TIMESTAMP)             AS END_TIMESTAMP,
 MAX(RES_MH.ROLLUP_TIMESTAMP)      AS ROLLUP_TIMESTAMP,
 MAX(RES_MH.dumpUsedPercent)       AS dumpUsedPercent,      -- Dump Area Used
 MAX(RES_MH.redosize_ps)           AS redosize_ps,          -- Redo Generated
 MAX(RES_MH.usercall_pct)          AS usercall_pct,         -- User Calls
 MAX(RES_MH.tabscanstotal_ps)      AS tabscanstotal_ps,   -- Total Table Scans
 MAX(RES_MH.bufcachehit_pct)       AS bufcachehit_pct,    -- Buffer Cache Hit
 MAX(RES_MH.libcache_hit_pct)      AS libcache_hit_pct,   -- Library Cache Hit
 MAX(RES_MH.dictionaryhit_pct)     AS dictionaryhit_pct,  -- Data Dictionary
Hit
 MAX(RES_MH.inmem_sort_pct)        AS inmem_sort_pct,      -- Sorts in Memory
 MAX(RES_MH.redologalloc_hit_pct)  AS redologalloc_hit_pct,-- RedoLog Alloc
 MAX(RES_MH.cursorcachehit_pct)    AS cursorcachehit_pct,  -- Cursor Cache Hit
 MAX(RES_MH.cpuusage_ps)           AS cpuusage_ps,         -- CPU Usage
 MAX(RES_MH.cpu_time_pct)          AS cpu_time_pct,        -- Database CPU
Time
 MAX(RES_MH.libcache_miss_pct)     AS libcache_miss_pct,   -- Library
CacheMiss
 MAX(RES_MH.pgacachehit_pct)       AS pgacachehit_pct,     -- PGA Cache Hit
 MAX(RES_MH.dictionarymiss_pct)    AS dictionarymiss_pct,  -- Row Cache
MissRat
 MAX(RES_MH.shared_free_pct)       AS shared_free_pct,     -- Shared Pool Free
 MAX(RES_MH.large_free_pct)        AS large_free_pct,      -- Large Pool Free
 MAX(RES_MH.pctUsed)               AS pctUsed              -- Tablespace Used
FROM
 MGMT$AVAILABILITY_HISTORY AH,
 (SELECT
   TARGET_GUID, TARGET_NAME, TARGET_TYPE, METRIC_NAME, ROLLUP_TIMESTAMP,
   dumpUsedPercent, redosize_ps, softparse_pct, usercall_pct, tabscanstotal_ps,
   bufcachehit_pct, libcache_hit_pct, dictionaryhit_pct, inmem_sort_pct,
   redologalloc_hit_pct, cursorcachehit_pct, cpuusage_ps, cpu_time_pct,
   libcache_miss_pct,pgacachehit_pct, dictionarymiss_pct, shared_free_pct,
   large_free_pct, pctUsed
  FROM MGMT$METRIC_HOURLY
  PIVOT ( MAX(MAXIMUM) FOR METRIC_COLUMN IN (
   'dumpUsedPercent'          AS dumpUsedPercent,
   'redosize_ps'              AS redosize_ps,
   'softparse_pct'            AS softparse_pct,
   'usercall_pct'             AS usercall_pct,
   'tabscanstotal_ps'         AS tabscanstotal_ps,
   'bufcachehit_pct'          AS bufcachehit_pct,
   'libcache_hit_pct'         AS libcache_hit_pct,
   'dictionaryhit_pct'        AS dictionaryhit_pct,
   'inmem_sort_pct'           AS inmem_sort_pct,
   'redologalloc_hit_pct'     AS redologalloc_hit_pct,
   'cursorcachehit_pct'       AS cursorcachehit_pct,
   'cpuusage_ps'              AS cpuusage_ps,
   'cpu_time_pct'             AS cpu_time_pct,
   'libcache_miss_pct'        AS libcache_miss_pct,
```

```
    'pgacachehit_pct'          AS pgacachehit_pct,
    'dictionarymiss_pct'       AS dictionarymiss_pct,
    'shared_free_pct'          AS shared_free_pct,
    'large_free_pct'           AS large_free_pct,
    'pctUsed'                  AS pctUsed ))
   WHERE TARGET_TYPE = 'oracle_database') RES_MH
WHERE
 AH.TARGET_GUID = RES_MH.TARGET_GUID AND
(RES_MH.ROLLUP_TIMESTAMP BETWEEN AH.START_TIMESTAMP - NUMTODSINTERVAL(30,
'MINUTE')
 AND AH.START_TIMESTAMP + NUMTODSINTERVAL(30, 'MINUTE'))
 AND AH.TARGET_TYPE = 'oracle_database'
GROUP BY RES_MH.TARGET_GUID, AH.START_TIMESTAMP
ORDER BY RES_MH.TARGET_GUID, AH.START_TIMESTAMP;
```

Listato 4.2 Query per la creazione del WebLogic Dataset.

```
SELECT
 RES_MH.TARGET_GUID                      AS TARGET_GUID,
 MAX(RES_MH.TARGET_TYPE)                 AS TARGET_TYPE,
 MAX(AH.AVAILABILITY_STATUS)             AS AVAILABILITY_STATUS,
 AH.START_TIMESTAMP                      AS START_TIMESTAMP,
 MAX(AH.END_TIMESTAMP)                   AS END_TIMESTAMP,
 MAX(RES_MH.ROLLUP_TIMESTAMP)            AS ROLLUP_TIMESTAMP,
 MAX(RES_MH.cpuUsage)                    AS cpuUsage,
 MAX(RES_MH.connectionWaitSuccesses)     AS connectionWaitSuccesses,
 MAX(RES_MH.ejbPoolAccessSuccess)        AS ejbPoolAccessSuccess,
 MAX(RES_MH.oldHeapPercentFreeAfterGc)   AS oldHeapPercentFreeAfterGc,
 MAX(RES_MH.ejbTransactionCommit)        AS ejbTransactionCommit,
 MAX(RES_MH.ejbCacheHit)                 AS ejbCacheHit,
 MAX(RES_MH.jtaTransactionCommit)        AS jtaTransactionCommit,
 MAX(RES_MH.connectionWait)              AS connectionWait,
 MAX(RES_MH.stmtCacheSatisfied)          AS stmtCacheSatisfied,
 MAX(RES_MH.heapUsedPercentage)          AS heapUsedPercentage,
 MAX(RES_MH.percentTimeInGc)             AS percentTimeInGc,
 MAX(RES_MH.connectionSuccess)           AS connectionSuccess
FROM
 MGMT$AVAILABILITY_HISTORY AH,
 (SELECT
  TARGET_GUID,
  TARGET_TYPE,
  METRIC_NAME,
  ROLLUP_TIMESTAMP,
  cpuUsage, connectionWaitSuccesses, ejbPoolAccessSuccess,
  oldHeapPercentFreeAfterGc, ejbTransactionCommit, ejbCacheHit,
  jtaTransactionCommit, connectionWait, stmtCacheSatisfied,
  heapUsedPercentage, percentTimeInGc, connectionSuccess
 FROM MGMT$METRIC_HOURLY
 PIVOT ( MAX(MAXIMUM) FOR METRIC_COLUMN IN (
   'cpuUsage.percentage'                 AS cpuUsage,
   'connectionWaitSuccesses.percentage'  AS connectionWaitSuccesses,
   'ejbPoolAccessSuccess.percentage'     AS ejbPoolAccessSuccess,
   'oldHeapPercentFreeAfterGc'           AS oldHeapPercentFreeAfterGc,
   'ejbTransactionCommit.percentage'     AS ejbTransactionCommit,
   'ejbCacheHit.percentage'              AS ejbCacheHit,
```

```
   'jtaTransactionCommit.percentage'       AS jtaTransactionCommit,
   'connectionWait.percentage'             AS connectionWait,
   'stmtCacheSatisfied.percentage'         AS stmtCacheSatisfied,
   'heapUsedPercentage.value'              AS heapUsedPercentage,
   'percentTimeInGc'                       AS percentTimeInGc,
   'connectionSuccess.percentage'          AS connectionSuccess ))
 WHERE
  TARGET_TYPE IN
      ('weblogic_j2eeserver','weblogic_cluster','weblogic_nodemanager')
 ) RES_MH
WHERE
 AH.TARGET_GUID = RES_MH.TARGET_GUID AND
 (RES_MH.ROLLUP_TIMESTAMP BETWEEN
                    AH.START_TIMESTAMP - NUMTODSINTERVAL(30, 'MINUTE') AND
                    AH.START_TIMESTAMP + NUMTODSINTERVAL(30, 'MINUTE'))
GROUP BY RES_MH.TARGET_GUID, AH.START_TIMESTAMP
ORDER BY RES_MH.TARGET_GUID, AH.START_TIMESTAMP;
```

Listato 4.3 Query per la creazione del Host Dataset.

```
SELECT
 RES_MH.TARGET_GUID              AS TARGET_GUID,
 MAX(RES_MH.TARGET_NAME)         AS TARGET_NAME,
 MAX(RES_MH.TARGET_TYPE)         AS TARGET_TYPE,
 MAX(AH.AVAILABILITY_STATUS)     AS AVAILABILITY_STATUS,
 AH.START_TIMESTAMP              AS START_TIMESTAMP,
 MAX(AH.END_TIMESTAMP)           AS END_TIMESTAMP,
 MAX(RES_MH.ROLLUP_TIMESTAMP)    AS ROLLUP_TIMESTAMP,
 MAX(RES_MH.cpuLoad)             AS cpuLoad,               -- Run Queue Length
 MAX(RES_MH.noOfProcs)           AS noOfProcs,             -- Total Processes
 MAX(RES_MH.noOfUsers)           AS noOfUsers,          -- Total Users
 MAX(RES_MH.swapUtil)            AS swapUtil,           -- Swap Utilization
 MAX(RES_MH.memUsedPct)          AS memUsedPct,         -- Memory Utilization
 MAX(RES_MH.memfreePct)          AS memfreePct,         -- Free Memory (%)
 MAX(RES_MH.logicMemfreePct)     AS logicMemfreePct,    -- Free Logical Memory
 MAX(RES_MH.cpuUtil)             AS cpuUtil,            -- CPU Utilization
 MAX(RES_MH.cpuUser)             AS cpuUser,            -- CPU in User Mode
 MAX(RES_MH.cpuKernel)           AS cpuKernel,          -- CPU in System Mode
 MAX(RES_MH.cpuIOWait)           AS cpuIOWait,          -- CPU in I/O Wait
 MAX(RES_MH.usedLogicalMemoryPct)  AS usedLogicalMemoryPct,  --
UsedLogicalMemory
 MAX(RES_MH.fs_used)             AS fs_used,             -- Total Filesystem
Used
 MAX(RES_MH.DiskActivitybusy)    AS DiskActivitybusy     -- Disk Busy
FROM
 MGMT$AVAILABILITY_HISTORY AH,
 (SELECT
  TARGET_GUID, TARGET_NAME, TARGET_TYPE, METRIC_NAME, ROLLUP_TIMESTAMP,
  cpuLoad,noOfProcs,noOfUsers,swapUtil,memUsedPct,
  memfreePct,logicMemfreePct,cpuUtil,cpuUser,cpuKernel,
  cpuIOWait,usedLogicalMemoryPct,fs_used,DiskActivitybusy
 FROM
  MGMT$METRIC_HOURLY
```

```
 PIVOT ( MAX(MAXIMUM) FOR METRIC_COLUMN IN (
   'cpuLoad'               AS cpuLoad,
   'noOfProcs'             AS noOfProcs,
   'noOfUsers'             AS noOfUsers,
   'swapUtil'              AS swapUtil,
   'memUsedPct'            AS memUsedPct,
   'memfreePct'            AS memfreePct,
   'logicMemfreePct'       AS logicMemfreePct,
   'cpuUtil'               AS cpuUtil,
   'cpuUser'               AS cpuUser,
   'cpuKernel'             AS cpuKernel,
   'cpuIOWait'             AS cpuIOWait,
   'usedLogicalMemoryPct'  AS usedLogicalMemoryPct,
   'fs_used'               AS fs_used,
   'DiskActivitybusy'   AS DiskActivitybusy ))
 WHERE TARGET_TYPE = 'host') RES_MH
WHERE
 AH.TARGET_GUID = RES_MH.TARGET_GUID
 AND (RES_MH.ROLLUP_TIMESTAMP BETWEEN
                        AH.START_TIMESTAMP - NUMTODSINTERVAL(30, 'MINUTE') AND
                        AH.START_TIMESTAMP + NUMTODSINTERVAL(30, 'MINUTE'))
 AND AH.TARGET_TYPE = 'host'
GROUP BY RES_MH.TARGET_GUID, AH.START_TIMESTAMP
ORDER BY RES_MH.TARGET_GUID, AH.START_TIMESTAMP;
```

4.3 Preprocessing

Durante l'applicazione dei modelli predittivi scelti ai dataset acquisiti, si è resa subito evidente la necessità di effettuare una fase di preprocessing sui dati atta a migliorare le performance degli algoritmi sfruttatati: il campionamento delle metriche raccolte da Oracle Enterprise Manager, infatti, non risulta essere sufficientemente preciso per il tipo di analisi svolta.

In questo paragrafo vengono quindi illustrati i dettagli delle tecniche di preprocessing usate sui dataset a partire dalla prima fase di "Data cleaning", utile all'eliminazione dei record superflui, per poi discutere la gestione del bilanciamento delle classi data da una distribuzione non omogenea degli stati dei target considerati.

Infine vengono illustrate le tecniche di "feature selection", grazie alle quali è stato ridotto il numero di metriche considerate dagli algoritmi predittivi, e le scelte effettuare sui dataset per la divisione degli stessi in train e test set.

4.3.1 Data cleaning e data transformation

Con il termine "data cleaning" si fa riferimento a tutti quei processi atti a migliorare la qualità dei dati presenti all'interno di un dataset, database o data warehouse per consentirne la loro memorizzazione e successiva analisi: nel caso della ricerca sviluppata tale fase ha permesso di ottimizzare le performance dei modelli predittivi, descritti nei paragrafi successivi, e garantirne il loro corretto funzionamento.
Un primo evidente problema, legato al metodo di campionamento effettuato da Oracle Enterprise Manager, è rappresentato dall'assenza di molti valori relativi alle metriche analizzate: la gestione dei valori mancanti all'interno di un dataset può essere affrontata in due modi differenti, ovvero con l'eliminazione dei records affetti da tale problema oppure con l'immissione di valori calcolati sulla base di altri dati presenti.
Nello sviluppo della ricerca sono stati combinati entrambi i metodi sopracitati sia per far fronte alla assenza dei valori sia per evitare una eccessiva riduzione del numero di records presenti nel dataset: in questo senso si è deciso di adottare una politica diversa per ognuno dei dataset creati in base alla loro cardinalità e alle caratteristiche del problema presentato.
Il grafico in figura 4.2 mostra il numero di valori assenti per tipo di dataset: ogni record, infatti, può contenere una o più metriche non campionate da Oracle Enterprise Manager.

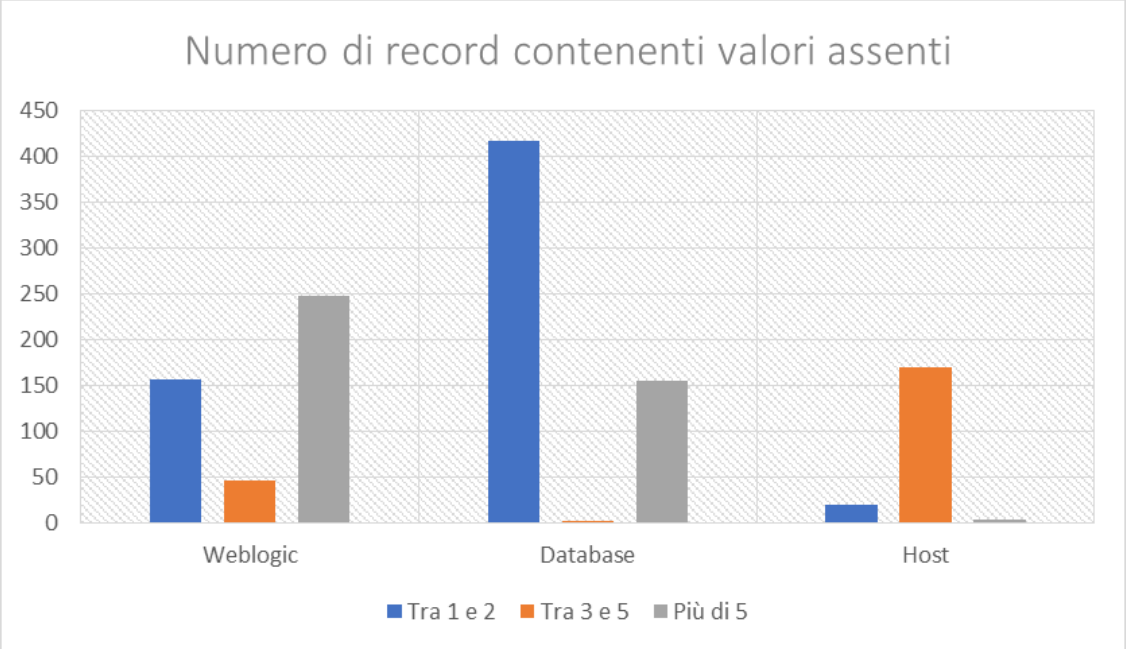

Figura 4.2: numero di record contenenti valori assenti nei tre dataset studiati.

Poiché i dataset Database e Host risultano più piccoli in termini di cardinalità (vedi 4.2 "*Raccolta dei dati*") sono stati conservati i record aventi al più due metriche non campionate e sono stati invece eliminati tutti gli altri record contenenti un numero maggiore di valori nulli.
Per il dataset Weblogic, avente invece cardinalità superiore rispetto agli altri, sono stati eliminati tutti i record aventi più di un valore nullo.

Una volta rimossi i record con una maggiore quantità di valori nulli, sono stati elaborati i restanti dati: in questo senso si è scelto di inserire al posto dei valori mancanti la media calcolata utilizzando i dati della colonna corrispondente.
I provvedimenti presi hanno di conseguenza modificato i dataset studiati e la tabella 4.3 ne riassume le cardinalità prima della fase di data cleaning, considerando solo gli stati analizzati (Target Up, Target Down e Agent Down per Host Dataset) e dopo la fase di data cleaning:

Tabella 4.3. Cardinalità dei dataset prima e dopo la fase di data cleaning.

Dataset	Cardinalità pre data cleaning	Cardinalità post data cleaning
WebLogic	4182 records	3888 records
Database	805 records	649 records
Host	1293 records	1120 records

Altro aspetto del preprocessing dei dati è quello caratterizzato dalla cosiddetta fase di "data transformation": con questo termine si indicano tutte quelle strategie atte a modificare i valori presenti in un dataset per renderli tra loro omogenei.
Ai fini dell'applicazione dei modelli predittivi scelti si è quindi effettuata la standardizzazione dei valori secondo la formula

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x}$$

Dove μ_x è la media relativa alla colonna della metrica considerata, e σ_x è la corrispondente deviazione standard.

4.3.2 Gestione sbilanciamento classi

Come visto nel paragrafo 3.2.3 *"Gli stati"*, le classi "Target Down" ("Agent Down" per il dataset Host) e "Target Up", codificate rispettivamente con i numeri 0 e 1, risultano tra loro molto sbilanciate in tutti i dataset creati. Gli effetti principali di tale aspetto sono da ricercarsi nei risultati dell'applicazione dei modelli predittivi: qualora il numero di dati di una classe sia molto superiore agli altri, infatti, i modelli matematici tenderanno a dare maggior peso alla classe maggioritaria influenzando negativamente la precisione dei risultati.
I possibili metodi per la gestione di tale problema si possono dividere in due categorie: "data level methods" e "classifier level methods" [11]. Il primo opera sulla distribuzione delle classi nel training set cambiandone il livello di sbilanciamento con l'utilizzo di tecniche quali "oversampling" e "undersampling"; il secondo tipo di metodo non modifica invece i dataset usati ma piuttosto sfrutta alcune caratteristiche intrinseche dei modelli matematici per la soluzione del problema.

Nello sviluppo della ricerca sono stati combinate entrambe le metodologie in modi diversi a seconda del dataset considerato e del modello predittivo applicato: per tutti i dataset, infatti, è stata eseguita una prima fase di "oversampling" dei dati della classe minoritaria e successivamente una di "undersampling" della classe maggioritaria per consentire di dar maggior peso allo stato "Target Down" che, a livello aziendale, è considerato più critico e quindi più utile nella fase di predizione.
Tale approccio è stato portato avanti per i modelli predittivi "Random Forest" e "Neural Network" mentre per il "Support Vector Machine" è stata sfruttata la possibilità intrinseca del modello di dare maggiore peso ad una classe rispetto che ad un'altra, motivo per il quale non è stato effettuato l'undersampling della classe maggioritaria.

I seguenti grafici mostrano la diversa distribuzione degli stati dopo una prima fase di oversampling della classe "Target Down" ("Agent Down" per il dataset Host) e un'ultima di undersampling della classe "Target Up" per i tre dataset studiati:

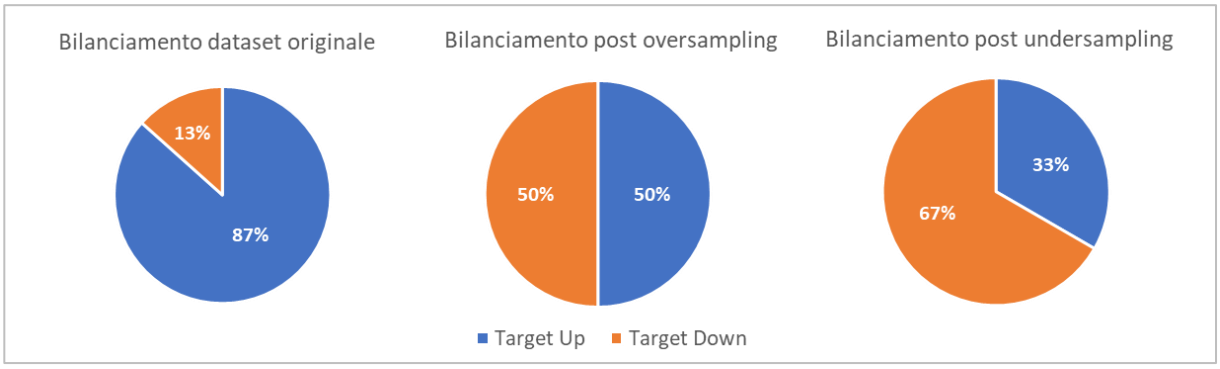

Figura 4.3: bilanciamento classi per il dataset Host.

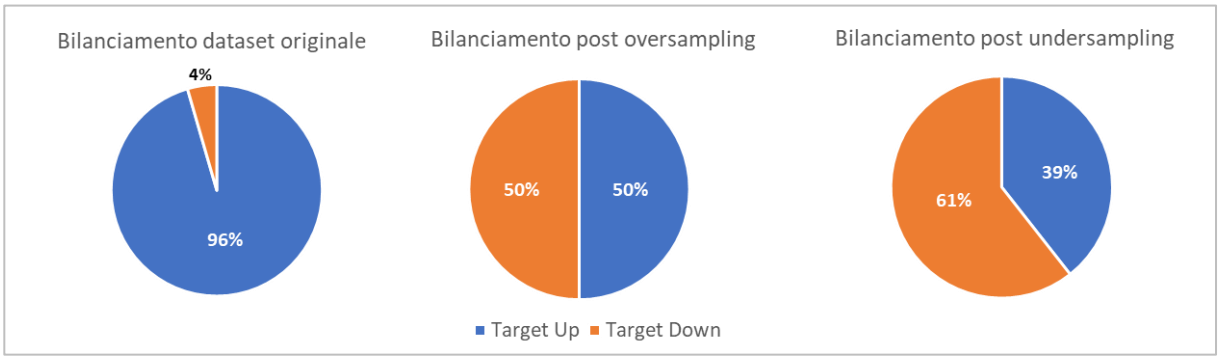

Figura 4.4: bilanciamento classi per il dataset WebLogic.

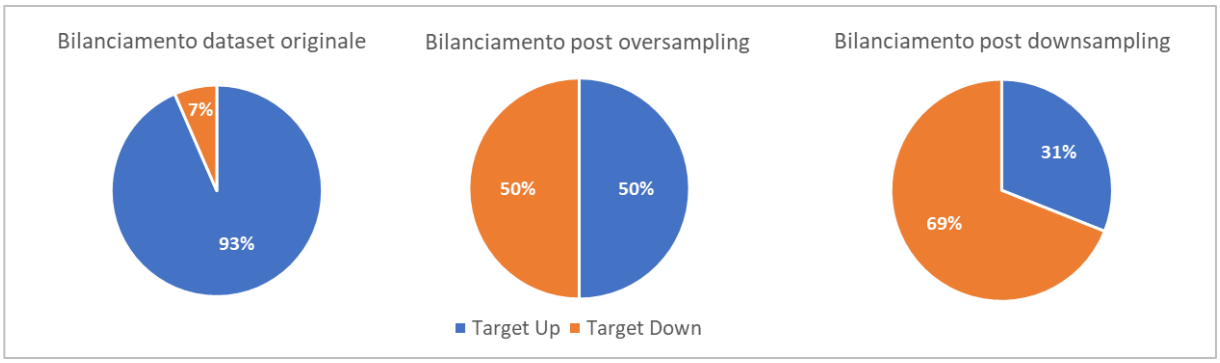

Figura 4.5: bilanciamento classi per il dataset Database.

Come si può notare dai grafici precedenti l'operazione di oversampling è stata utilizzata per pareggiare il numero di dati delle due classi mentre quella di undersampling per sbilanciare le classi in favore della classe "Target Down": il parametro principale utilizzato per valutare l'efficacia del nuovo sbilanciamento ottenuto è la cosiddetta "recall" sulla classe "Target Down", come spiegato nel paragrafo 2.2.4 "*Metodi di valutazione dei modelli*".
Nell'utilizzo del modello SVM, invece, è stato tralasciato l'undersampling della classe maggioritaria in favore dell'applicazione di un peso pari a 0.75 per la classe "Target Down" e 0.25 per la classe "Target Up", ovvero si è dato alla prima classe un peso tre volte superiore all'altra.

L'implementazione dell'oversampling dei dati è stata possibile grazie all'utilizzo dell'algoritmo SMOTE (Synthetic Minority Over-sampling TEchnique) [18]: tale tecnica consente la sovracampionatura dei dati sulla base di quelli esistenti senza però crearne dei duplicati come avviene per altre tecniche di oversampling. La classe minoritaria, infatti, è sovracampionatura introducendo esempi "sintetici" sulla linea che collega un campionamento e i suoi k campionamenti vicini appartenenti alla stessa classe.

4.3.3 Feature selection

Una delle fasi di preprocessing che hanno permesso l'ottimizzazione delle performance dei modelli predittivi applicati è la cosiddetta "feature selection" (o "selezione degli attributi"): con questa espressione si fa riferimento a quelle tecniche atte all'individuazione degli attributi più utili ai fini dell'analisi.
Questo tipo di operazione permette di ottenere alcuni vantaggi sui risultati dei modelli predittivi:

- ottimizzazione della generalizzazione del modello (riduzione dell'overfitting)
- tempi di elaborazione ridotti
- semplificazione del modello per una migliore interpretabilità dei risultati

L'utilizzo di un numero eccessivo di attributi, infatti, lede alle performance dei modelli predittivi in quanto da origine al fenomeno dell'overfitting: in questo caso il modello scelto tende a non generalizzare i risultati dell'apprendimento effettuato sul training set causando la creazione di un modello poco utile ai fini dell'analisi di dati mai visti prima.
In figura 4.6 viene illustrato un esempio di overfitting e underfitting (fenomeno inverso) per uno stesso generico modello predittivo:

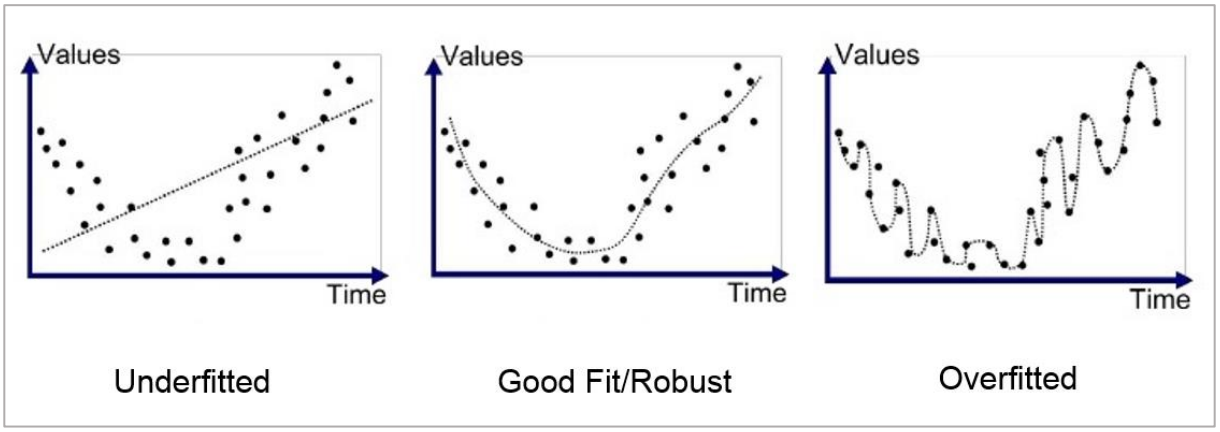

Figura 4.6: esempio di overfitting e underfitting per un generico modello matematico [19].

Come si può notare dalla figura 4.6, un modello "overfitted" segue in maniera troppo precisa i dati di training comportando, di conseguenza, la poca accuratezza dei risultati dell'applicazione dello stesso modello su dati di test.

Tra le possibili tecniche di feature selection disponibili in letteratura, è stata utilizzata quella chiamata "RFE" (Recursive Feature Elimination): l'obiettivo di tale tecnica è quello di selezionare ricorsivamente un insieme di attributi sempre più piccolo grazie all'applicazione di un modello matematico. Nel caso della ricerca sviluppata è stato scelto il Random Forest che, una volta allenato con il dataset di training, ha permesso di raccogliere le statistiche relative agli attributi più utili sulla base del valore detto "Gini Importance" o "Mean Decrease in Impurity (MDI)": tale formula calcola l'importanza di ciascun attributo come la somma di splits (su tutti gli alberi creati) che includono tale attributo, rispetto al numero totale di splits creati [20].

I grafici seguenti illustrano il risultato della feature selection per i tre dataset considerati: maggiore è il valore dell'importanza di un attributo, maggiore è l'utilità dello stesso nell'applicazione dell'algoritmo:

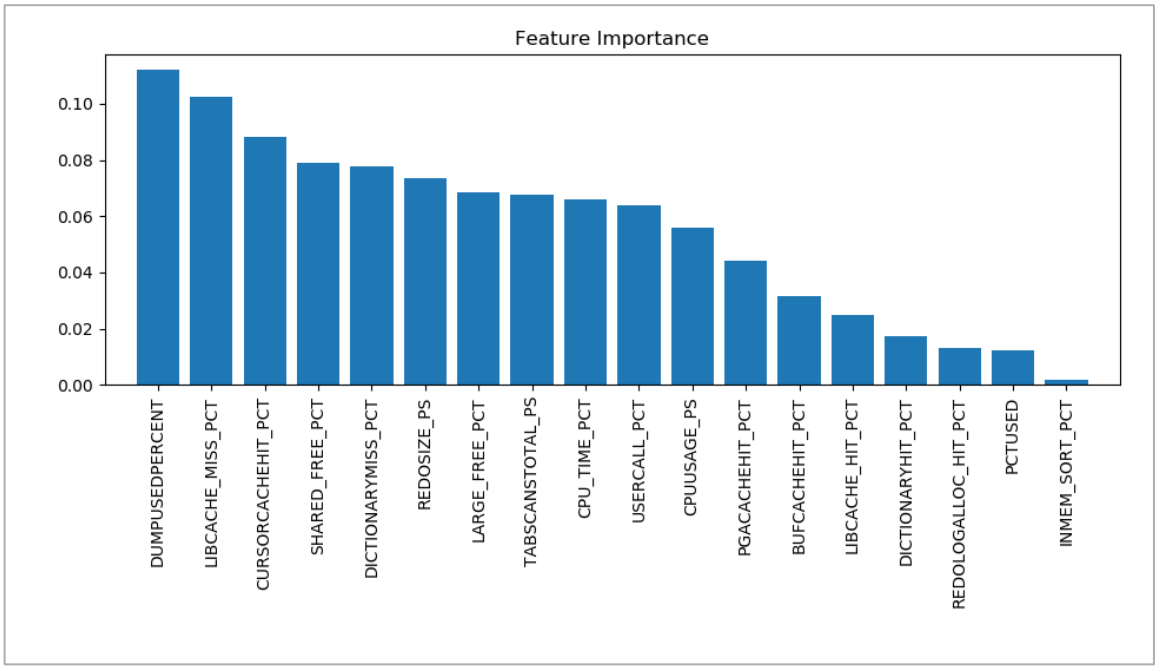

Figura 4.7: feature importance per il dataset Database.

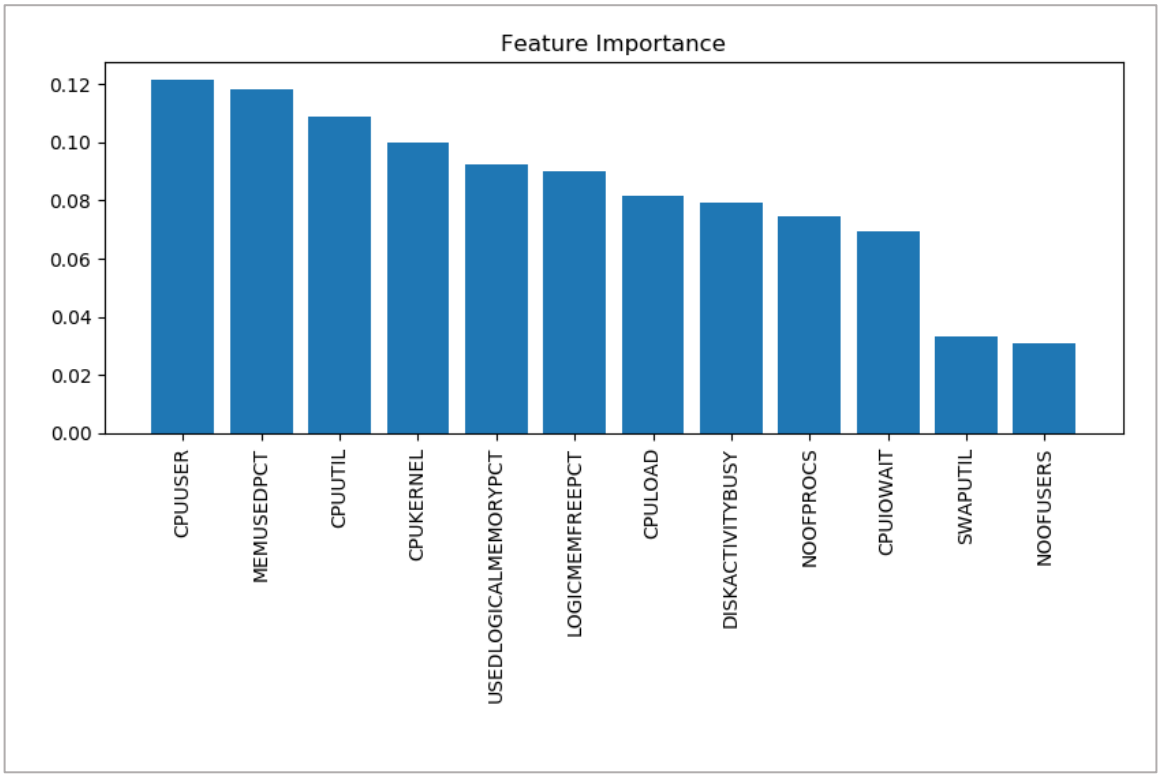

Figura 4.8: feature importance per il dataset Host.

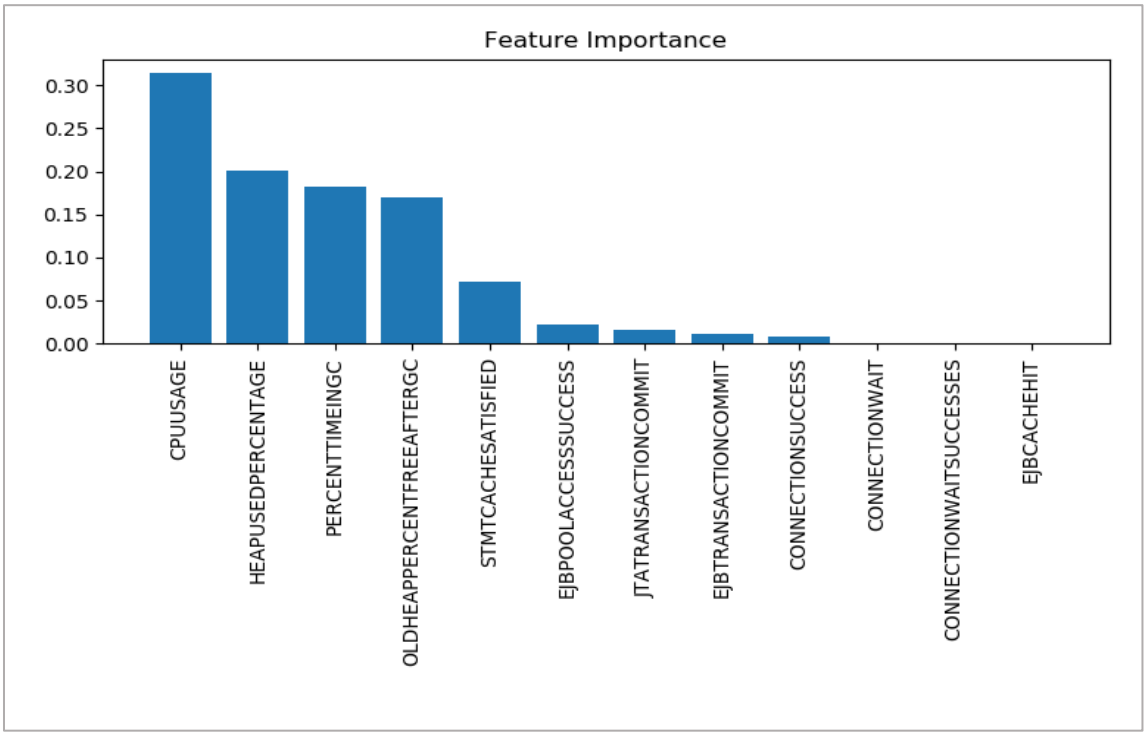

Figura 4.9: feature importance per il dataset WebLogic.

Per tutti e tre i dataset è stato scelto quindi di selezionare solo le metriche con un valore di importanza superiore a 0.07: tale scelta ha quindi permesso di utilizzare solo le feature elencate di seguito con un conseguente miglioramento delle performance dei modelli predittivi in termini di accuratezza e recall.

Tabella 4.4. Metriche selezionate con la feature selection per i tre dataset.

Dataset Database	Dataset Host	Dataset WebLogic
dumpusedpercent	cpuuser	cpuusage
libcache_miss_pct	memusedpct	heapusedpercentage
cursorcachehit_pct	cpuutil	percenttimeingc
shared_free_pct	cpukernel	oldheappercentfreeaftergc
dictionarymiss_pct	usedlogicalmemorypct	stmtcachesatisfied
redosize_ps	logicmemfreepct	
	cpuload	
	diskactivitybusy	
	noofprocs	

4.3.4 Training e Test set

Nel machine learning si definisce "training set" quella parte del dataset studiato dedicata alla creazione di un modello matematico, mentre si indica con "test set" la restante parte del dataset utilizzata per la valutazione delle performance del modello precedentemente generato.
Tale aspetto occupa un ruolo molto importante nella fase selezione e preprocessing dei dati in quanto influenza attivamente il numero di valori di cui il modello può disporre per l'analisi dei dati: solitamente, negli studi che affrontano la questione, si sceglie di inserire nel training set la maggior parte dei dati presenti nel dataset e lasciare nel test set solo una piccola parte di valori utili alla valutazione del modello predittivo.
Durante la ricerca sviluppata è stata intrapresa questa stessa politica di divisione dei dati: in tutti e tre i dataset studiati, infatti, si è scelto di lasciare al training set il 70% dei valori mentre il restante 30% è stato dedicato al test set.
La tabella 4.5 riassume le cardinalità dei set per i tre dataset studiati dopo la fase di preprocessing e la divisione in training e test set:

Tabella 4.5. Cardinalità di training e test set dei dataset studiati dopo la fase di preprocessing.

Dataset	Training set	Test set
Database	872 records (post oversampling) 632 records (post undersampling)	202 records
Host	1564 records (post oversampling) 1173 records (post undersampling)	387 records
WebLogic	5198 records (post oversampling) 4288 records (post undersampling)	1167 records

5 Applicazione dei modelli predittivi

Una volta conclusasi la fase di raccolta dei dati e di preprocessing, ì dataset così creati sono stati elaborati per mezzo di tre modelli matematici: Random Forest, Support Vector Machine e Neural Network.
Come visto nel paragrafo 2.2 "*I modelli predittivi*", tutti e tre gli algoritmi sono in grado di agire come classificatori e possono quindi analizzare lo stato di un target grazie all'uso di un insieme di feature accuratamente selezionato: più in dettaglio i modelli etichettano ciascun campione analizzato come "Target Down" e "Target Up" a seconda della probabilità che si presenti o meno un problema nel sistema processato.
Per la valutazione dei risultati ottenuti con l'ottimizzazione dei modelli, avvenuta tramite l'utilizzo di tecniche quali GridSearch (ricerca esaustiva) degli iperparametri, si è fatto affidamento a parametri quali accuracy, precision, recall e confusion matrix descritti in maniera approfondita nel paragrafo 2.2.4 "*Metodi di valutazione dei modelli predittivi*".
Questo capitolo vuole quindi sintetizzare i risultati della applicazione dei tre algoritmi sopra citati illustrando per i dataset Host, Database e WebLogic e analizzare per ciascuno di questi quale dei tre modelli si è rivelato il migliore per l'obiettivo.

5.1 WebLogic Dataset

Nell'applicazione dei diversi modelli al dataset WebLogic si è tenuto conto del forte sbilanciamento delle classi "Target Down" (4% del totale dei dati) e "Target Up" (96% dei dati), fenomeno poi corretto nella fase di preprocessing con le tecniche descritte nel paragrafo 4.3.2 "*Gestione sbilanciamento classi*".
Per la verifica dei modelli è stato usato un test set con 1167 campioni di cui 1115 appartenenti alla classe Target Up e 52 alla classe Target Down.

Il primo tra i modelli applicati è stato il Random Forest: la sua ottimizzazione ha permesso di ottenere una accuracy pari al 86% con il seguente numero di alberi decisionali:

Tabella 5.1. Iperparametri per l'algoritmo Random Forest e il dataset WebLogic.

Iperparametri	Valore
N. Estimators	25

Con tali iperparametri si sono ottenuti i seguenti valori di valutazione sul test set:

Tabella 5.2. Parametri di valutazione per Random Forest sul dataset WebLogic.

Classe	Precision	Recall	F1 Score
Target Down	0.21	0.48	0.29
Target Up	0.97	0.92	0.94

La matrice di confusione indica invece che il 91% dei Target Up viene correttamente rilevato tale e il 48% come Target Down:

		PREDICTED CLASS	
ACTUAL CLASS		**Target Down**	**Target Up**
	Target Down	25	27
	Target Up	94	1021

Figura 5.1: matrice di confusione per l'Random Forest nel dataset WebLogic.

L'applicazione dell'algoritmo SVM per lo stesso dataset ha consentito l'utilizzo di pesi diversi per le due classi come indicato nella tabella 5.3, insieme agli iperparametri ottenuti dopo la fase di ottimizzazione del modello:

Tabella 5.3. Iperparametri per l'algoritmo SVM e il dataset WebLogic.

Iperparametri	Valore
C	1
Kernel	RBF
Gamma	100
Class Weight	Target Down: 0.75 Target Up: 0.25

Il modello così configurato ottiene un valore di accuratezza dell'81% e, nell'ottica di un aumento del numero dei Target Down correttamente rilevati, si hanno i seguenti valori di precision, recall e f1 score:

Tabella 5.4. Parametri di valutazione per SVM sul dataset WebLogic.

Classe	Precision	Recall	F1 Score
Target Down	0.17	0.81	0.28
Target Up	0.99	0.81	0.89

La matrice di confusione mette in evidenza invece che l'80% dei Target Down è correttamente rilevato come tale contro l'81% delle corrette predizioni sui Target Up

		PREDICTED CLASS	
ACTUAL CLASS		**Target Down**	**Target Up**
	Target Down	42	10
	Target Up	210	905

Figura 5.2: matrice di confusione per l'SVM nel dataset WebLogic.

Per quanto riguarda l'ultimo dei modelli provati sul dataset WebLogic, Neural Network, si è ottenuta una accuratezza pari al 78% con i seguenti iperparametri:

Tabella 5.5. Iperparametri per l'algoritmo Neural Network e il dataset WebLogic.

Iperparametri	Valore
Activation Function	ReLu
Solver	L-BFGS
Batch Size	40
Hidden Layers Size	30
Max Iteration	500

I parametri di valutazione in questo caso suggeriscono che i Target Down e Target Up vengono considerati corretti per l'83% e il 73%:

Tabella 5.6. Parametri di valutazione per Neural Network sul dataset WebLogic.

Classe	Precision	Recall	F1 Score
Target Down	0.15	0.83	0.26
Target Up	0.99	0.78	0.88

		PREDICTED CLASS	
ACTUAL CLASS		**Target Down**	**Target Up**
	Target Down	43	9
	Target Up	240	875

Figura 5.3: matrice di confusione per Neural Network nel dataset WebLogic.

5.2 Host Dataset

Nell'analisi del dataset Host, il cui sbilanciamento delle classi è meno rilevante rispetto al dataset WebLogic, si deve tenere conto che la classe di riferimento per la classificazione delle situazioni critiche è data dallo stato "Agent Down" (vedi paragrafo 3.2.3 "*Gli stati*"): per semplicità nella trattazione e simmetria con gli altri dataset tale classe, nel proseguo del capitolo, verrà comunque chiamata "Target Down", in contrapposizione alla classe "Target Up" che indica uno stato dell'host senza anomalie.
I modelli sono quindi stati testati su un dataset di test contenente 387 dati di cui 335 "Target Up" e 52 relativi al "Target Down".

Come per il dataset WebLogic, il primo degli algoritmi applicati è stato il Random Forest con il quale si è ottenuto una accuratezza del 73% utilizzando i seguenti iperparametri:

Tabella 5.7. Iperparametri per l'algoritmo Random Forest e il dataset WebLogic.

Iperparametri	Valore
N. Estimators	25

I parametri quali recall, precision, f1 score ottenuti evidenziano che il 44% dei Target Down e il 78% dei Target Up vengono valutati correttamente:

Tabella 5.8. Parametri di valutazione per Random Forest sul dataset Host.

Classe	Precision	Recall	F1 Score
Target Down	0.24	0.44	0.31
Target Up	0.90	0.78	0.84

I dati della recall e precisione sono meglio esplicitati dalla matrice di confusione rappresentata nella figura 5.4:

		PREDICTED CLASS	
ACTUAL CLASS		**Target Down**	**Target Up**
	Target Down	23	29
	Target Up	74	261

Figura 5.4: matrice di confusione per Random Forest nel dataset Host.

Diverso è il caso per il Support Vector Machine che per il dataset considerato ha portato risultati di accuratezza più bassi (58%) ma ha consentito un netto miglioramento in termini di recall sulla classe Target Down grazie alla gestione dei pesi sulle varie classi come mostrato nella tabella 5.9:

Tabella 5.9. Iperparametri per l'algoritmo SVM e il dataset Host.

Iperparametri	Valore
C	1
Kernel	RBF
Gamma	10
Class Weight	Target Down: 0.75 Target Up: 0.25

Le implicazioni di questo tipo di scelte sono discusse nel paragrafo 5.4 "*Analisi dei risultati ottenuti*" ma già dai parametri nella tabella 5.10 si nota come la recall indichi una percentuale di campioni correttamente classificati pari all' 88% per i Target Down e 52% per i Target Up:

Tabella 5.10. Parametri di valutazione per SVM sul dataset Host.

Classe	Precision	Recall	F1 Score
Target Down	0.22	0.88	0.36
Target Up	0.97	0.52	0.68

La situazione descritta è ben illustrata anche dalla relativa matrice di confusione che dimostra come 46 campioni dei 52 appartenenti alla classe Target Down sono rilevati correttamente:

		PREDICTED CLASS	
ACTUAL CLASS		**Target Down**	**Target Up**
	Target Down	46	6
	Target Up	161	174

Figura 5.5: matrice di confusione per SVM nel dataset Host.

L'utilizzo delle reti neurali ha dato esiti simili a quelli del Random Forest con una accuratezza del 74% e valori di recall poco più alti dello 0.76. In questo caso l'ottimizzazione del modello ha permesso di ottenere i seguenti valori degli iperparametri:

Tabella 5.11. Iperparametri per l'algoritmo Neural Network e il dataset Host.

Iperparametri	Valore
Activation Function	ReLu
Solver	L-BFGS
Batch Size	40
Hidden Layers Size	30
Max Iteration	500

I parametri di valutazione ottenuti sono invece esplicitati nella tabella 5.12:

Tabella 5.12. Parametri di valutazione per Neural Network sul dataset Host.

Classe	Precision	Recall	F1 Score
Target Down	0.29	0.60	0.39
Target Up	0.92	0.77	0.84

Dai valori di recall e precision sopraindicati e quelli illustrati nella matrice di confusione sottostante si deduce come il numero di Target Down rilevati correttamente sia drasticamente

più basso rispetto all'applicazione del SVM nonostante una maggiore accuratezza generale del modello descritto:

		PREDICTED CLASS	
ACTUAL CLASS		**Target Down**	**Target Up**
	Target Down	31	21
	Target Up	77	258

Figura 5.6: matrice di confusione per Neural Network nel dataset Host.

5.3 Database Dataset

Il dataset creato grazie al monitoraggio delle tecnologie Oracle Database e RAC Database è quello che ha richiesto uno sforzo maggiore relativamente all'ottimizzazione dei modelli applicati a causa del numero limitato di campioni disponibili per il training e il testing degli algoritmi: il numero totale di campioni usati per il training è infatti di 470 elementi prima della fase di preprocessing.
Per il test dei modelli è stata usato invece il 30% dell'intero dataset per un totale di 202 campioni di cui solo 15 appartenenti alla classe Target Down e 187 per la classe Target Up.

L'algoritmo Random Forest applicato ai dati del dataset Database ha permesso di raggiungere valori di accuratezza dell'80% con l'utilizzo dei seguenti iperparametri:

Tabella 5.13. Iperparametri per l'algoritmo Random Forest e il dataset Database.

Iperparametri	Valore
N. Estimators	25

Come scritto in precedenze, però, valori elevati di accuratezza non necessariamente seguiti da valori altrettanti grandi di recall e precision e la seguente tabella 5.14 ne è un esempio:

Tabella 5.14. Parametri di valutazione per Random Forest sul dataset Database.

Classe	Precision	Recall	F1 Score
Target Down	0.21	0.60	0.31
Target Up	0.96	0.82	0.88

La matrice di confusione conferma che l'82% dei Target Up è correttamente classificato contro il solo 60% dei Target Down:

		PREDICTED CLASS	
ACTUAL CLASS		**Target Down**	**Target Up**
	Target Down	9	6
	Target Up	34	153

Figura 5.7: matrice di confusione per Random Forest nel dataset Database.

Come per il dataset Host, l'applicazione del Support Vector Machine ha permesso di avere risultati di recall migliori nonostante il calo del valore di accuratezza del 73%.
Gli iperparametri ottenuti dall'ottimizzazione dell'algoritmo sono elencati nella tabella 5.15:

Tabella 5.15. Iperparametri per l'algoritmo SVM e il dataset Database.

Iperparametri	Valore
C	1
Kernel	RBF
Gamma	10
Class Weight	Target Down: 0.65 Target Up: 0.35

Con tali iperparametri si sono ottenuti i seguenti risultati:

Tabella 5.16. Parametri di valutazione per SVM sul dataset Database.

Classe	Precision	Recall	F1 Score
Target Down	0.19	0.73	0.30
Target Up	0.97	0.74	0.84

La matrice di confusione per l'algoritmo SVM conferma l'elevato numero di Target Down e Target Up correttamente identificati:

		PREDICTED CLASS	
ACTUAL CLASS		**Target Down**	**Target Up**
	Target Down	11	4
	Target Up	48	139

Figura 5.8: matrice di confusione per SVM nel dataset Database.

Come per il dataset Host, il Neural Network fornisce risultati simili al Random Forest con una accuratezza, usando i parametri seguenti, pari al 83%:

Tabella 5.17. Iperparametri per l'algoritmo Neural Network e il dataset Database.

Iperparametri	Valore
Activation Function	ReLu
Solver	L-BFGS
Batch Size	40
Hidden Layers Size	10
Max Iteration	500

I parametri di valutazione e la confusion matrix seguenti confermano il calo dei valori di recall nonostante l'aumento dell'accuratezza:

Tabella 5.18. Parametri di valutazione per Neural Network sul dataset Database.

Classe	Precision	Recall	F1 Score
Target Down	0.24	0.53	0.35
Target Up	0.96	0.86	0.91

		PREDICTED CLASS	
ACTUAL CLASS		**Target Down**	**Target Up**
	Target Down	8	7
	Target Up	26	161

Figura 5.9: matrice di confusione per Neural Network nel dataset Database.

5.4 Analisi dei risultati ottenuti

I risultati dettagliati nei paragrafi precedenti dimostrano come non esista un modello matematico migliore in assoluto per tutti i tipi di dataset in quanto questi variano molto per cardinalità e distribuzione delle classi: in questo senso, per dare dei termini di confronto, si è valutata in primo luogo l'accuratezza di ciascun algoritmo e successivamente la recall relativa alla classe Target Down. Questa, infatti, è quella più interessante dal punto di vista aziendale in quanto è preferibile avere un numero di falsi Target Down elevato piuttosto che un cospicuo numero di campioni della stessa classe erroneamente rilevati come Target Up.
Sintetizzando quest'ultimo concetto a livello pratico, per il contesto aziendale studiato è meglio avere un falso allarme in più piuttosto che un problema reale non rilevato dal sistema. Tale idea ha portato a considerare meno rilevante il valore della precision che infatti assume valori anche molto bassi nei modelli descritti: valori piccoli di questo parametro indicano, come descritto nel paragrafo 2.2.4 "*Metodi di valutazione dei modelli predittivi*", che il numero totale di rilevazioni per una classe contiene in realtà una grande quantità di campioni classificati erroneamente, motivo per il quale il modello non si può considerare preciso.
La causa che ha portato a valori di precision bassi nonostante una recall relativamente alta è da ricercarsi nella distribuzione molto sbilanciata delle classi e, soprattutto, nella gestione dei pesi che è stata effettuata nella fase di preprocessing volta a dare maggiore importanza alla classe Target Down.

Nel dataset WebLogic, il primo trattato in questo capitolo, risulta evidente che l'algoritmo Random Forest, nonostante un valore di accuratezza molto elevato, non riesca a dare buoni risultati in termini di precision e recall della classe Target Down: quasi la metà dei campioni di tale classe viene infatti rilevata erroneamente come appartenente alla classe Target Up.
Risultati migliori si ottengono dall'applicazione degli altri due modelli, SVM e Neural Network: nonostante valori di accuratezza leggermente minori di qualche punto percentuale

rispetto al Random Forest, tali modelli riescono a dare maggiore affidabilità nella rilevazione dei campioni della classe Target Down. In questo senso, infatti, si nota che i modelli sopracitati non classificano come Target Down rispettivamente il 17% e 19% dei campioni contro il 52% del Random Forest. Risulta comunque importante analizzare la quantità di falsi Target Down rilevati in quanto fanno parte dell'insieme di campioni che genererà un allarme verso un eventuale operatore che utilizza lo strumento: nel Random Forest infatti per ogni valore classificato correttamente come Target Down ve ne sono 3.76 erroneamente etichettati come tali. Tale valore tocca il picco di 5.58 campioni per l'algoritmo Neural Network, il peggiore dei tre in questo caso, contro i 5 campioni per l'SVM.
Visti i risultati ottenuti il Support Vector Machine risulta essere il miglior compromesso in termini di accuratezza e precisione del modello: l'elevato numero di falsi Target Down è mitigato dalla quantità relativamente piccola di campioni non rilevati come tali, elemento fondamentale per l'obiettivo della ricerca.

Per quanto riguarda l'analisi del dataset relativo alla tecnologia Host, il compromesso tra accuratezza e recall risulta particolarmente evidente nel confronto tra i tre diversi modelli matematici usati: tra questi, infatti, sia con il Random Forest sia con il Neural Network si ottengono valori di accuracy pari a 0.74 contro i soli 0.57 del SVM. Tale andamento è però l'opposto di quello relativo alla recall della classe Target Down: solo il Support Vector Machine, infatti, ottiene valori superiori allo 0.88 contro lo 0.6 e lo 0.44 di Neural Network e Random Forest.
Questa contrapposizione così marcata tra accuracy e recall implica che non vi sia un modello migliore in assoluto e risulta necessario piuttosto fare una scelta tra precisione nella classificazione dei campioni e affidabilità del sistema nel predire un eventuale Target Down: come scritto in precedenza, nel corso del lavoro svolto si è dato maggiore importanza a quest'ultimo aspetto motivo per il quale la scelta tra i tre modelli ricade sul Support Vector Machine.
Tale scelta è rafforzata dall'analisi delle matrici di confusione che esplicitano i valori ottenuti con la recall e la precision: in questo senso si nota come, dei 52 campioni di test appartenenti alla classe Target Down, l'SVM classifica in maniera errata l'11% di essi contro il 55% del Random Forest e il 40% di Neural Network.
Discorso a parte è quello relativo alla precision e, in particolar modo, al numero di Target Up rilevati in realtà come Target Down: in questo caso con il Support Vector Machine per ogni Target Down rilevato correttamente 3.5 campioni vengono erroneamente rilevati tali essendo in realtà Target Up. Tale parametro non migliora molto neanche per gli altri due modelli poiché il numero di corrette rilevazioni di Target Down risulta più basso: per ogni campione

di Target Down rilevato correttamente 3.2 campioni e 2.5 campioni sono erroneamente rilevati tali dagli algoritmi Neural Network e Random Forest.

Per il dataset relativo alle tecnologie Oracle Database e RAC Database l'analisi dei dati è stata particolarmente influenzata dal numero esiguo di campioni su cui è stato possibile effettuare la fase di training dei modelli e, successivamente, quella di test. Come visto nel paragrafo 4.3.1 "*Data cleaning e data transformation*", il dataset, considerando solo gli stati Target Up e Target Down, è costituito da 649 campioni di cui il 70% dedicato al training set e il restante al test set.
Dall'analisi dei risultati ottenuti si può notare come il migliore in termini di accuratezza sia il Neural Network con un valore pari all'83%, a seguire vi sono quindi il Random Forest e l'SVM con l'80% e il 74%.
L'ottimizzazione del Support Vector Machine, a prescindere dal valore di accuratezza ottenuto, ha permesso di raggiungere i migliori risultati relativamente alla recall del modello sulla classe Target Down: solo tale algoritmo infatti supera il valore di 0.73 dove gli altri modelli si fermano in realtà sotto il valore di 0.60.
Come per il dataset Host, il Support Vector Machine resta da preferirsi rispetto al Neural Network e il Random Forest mantenendo presente, comunque, che per ogni campione di Target Down ve ne sono più di 4 che sono in realtà Target Up, con il rischio di creare un gran numero di falsi allarmi.

6 Applicazione della ricerca nel contesto aziendale

La ricerca effettuata, nonostante fosse un progetto pilota atto ad esplorare la possibilità di applicare algoritmi di machine learning sui dati ricavati dall'utilizzo del software Oracle Enterprise System Manager, ha consentito la creazione di un pacchetto software scritto in Python capace di sfruttare i modelli matematici fin qui analizzati.
Una volta deciso che per i dataset studiati il Support Vector Machine fornisce i risultati migliori in termini di corretta rilevazione degli eventuali futuri problemi (come spiegato nel dettaglio nel paragrafo 5.4 "*Analisi dei risultati ottenuti*") si è serializzato il modello così allenato per poterlo utilizzare in un secondo momento con i dati provenienti in tempo reale dal Management Repository.
La serializzazione del modello consente, infatti, di utilizzare l'algoritmo senza la necessità di doverne fare nuovamente il training ogni qual volta vi fosse un nuovo dato da classificare.
Nell'immagine sottostante è illustrata la struttura dei moduli creati:

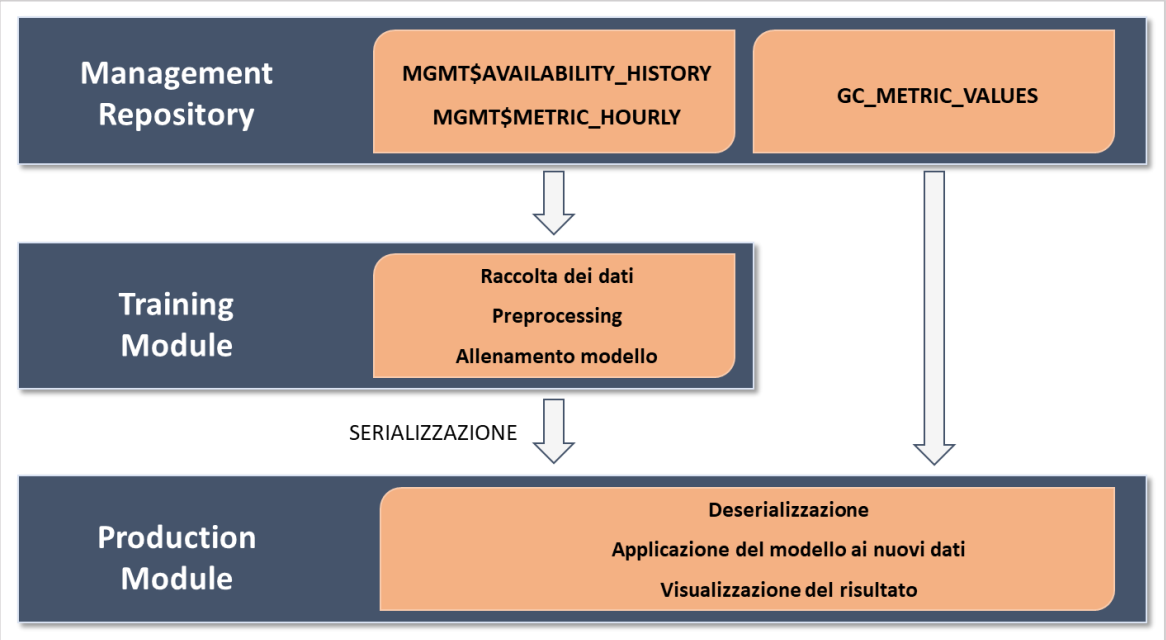

Figura 6.1: struttura del software creato in Python.

La struttura illustrata mostra come i dati vengano raccolti in maniera automatica dalle viste descritte nel paragrafo 3.2.4 "*Le viste utilizzate*" (MGMT$AVAILABILITY_HISTORY e MGMT$METRIC_HORLY) e siano elaborati dal modulo dedicato al training del modello studiato.
Una volta ottenuto il modello allenato questo viene serializzato (tramite l'utilizzo della libreria Python "Pickle") e messo a disposizione del modulo dedicato alla classificazione di nuovi dati.
Il Management Repository, infatti, raccoglie nella tabella GC_METRIC_VALUES tutti gli ultimi campionamenti effettuati sulle tecnologie monitorate prima che vengano aggregati e messi a disposizione nelle viste descritte.
Tale modulo è quindi progettato per deserializzare l'algoritmo allenato ed applicare gli ultimi dati ricevuti dal Management Repository: il risultato così ottenuto è quindi messo a disposizione degli operatori che possono considerarlo durante le proprie attività di monitoraggio.

7 Validazione sperimentale

La ricerca descritta è stata ideata con lo scopo di classificare, per mezzo di diversi modelli matematici, lo stato di salute di alcuni sistemi informatici monitorati nel contesto aziendale. Sono stati quindi raccolti i dati relativi a tre tecnologie di interesse aziendale diverse (Oracle Database, Oracle WebLogic e Oracle System Infrastructure) e, per mezzo delle metriche campionate con frequenza diversa dal software di monitoraggio Oracle Enterprise System Manager, sono stati creati tre dataset diversi relativi ciascuno ad un sistema differente.
I dataset citati (chiamati Database Dataset, WebLogic Dataset e Host Dataset) sono stati elaborati tramite varie tecniche di preprocessing per gestire alcune criticità come la presenza di valori nulli e lo sbilanciamento delle due classi: tra queste vi sono l'oversampling e l'undersampling dei dati, tecniche utili anche per dar maggior peso alla classe minoritaria (Target Down) relativa alla presenza di problematiche nel sistema analizzato.
La cardinalità dei tre dataset dopo la fase di preprocessing è riassunta nella tabella 7.1:

Tabella 7.1. Cardinalità di training e test set dei tre dataset studiati dopo la fase di preprocessing.

Dataset	Training set	Test set
Database	872 records (post oversampling) 632 records (post undersampling)	202 records
Host	1564 records (post oversampling) 1173 records (post undersampling)	387 records
WebLogic	5198 records (post oversampling) 4288 records (post undersampling)	1167 records

Sono stati quindi scelti tre modelli matematici con diverse caratteristiche (Random Forest, Support Vector Machine e Neural Network) e sono state messe a confronto le loro performance per trovare il miglioro per gli scopi della ricerca.
Poiché il principale obiettivo del lavoro è quello di classificare in maniera corretta la possibile presenza di una situazione critica in uno dei sistemi studiati si è dato maggior peso alla classe Target Down in modo che ogni campione fosse considerato tale se non chiaramente appartenente alla classe opposta (Target Up). In questo senso si è notato che il miglior algoritmo in termini di rilevazione corretta di tale stato è il Support Vector Machine per tutti e tre i dataset studiati: tale algoritmo, dato lo sbilanciamento dei pesi sulle classi, procura

però un notevole numero di falsi Target Down che devono quindi essere gestiti con l'intervento umano in un secondo momento.
In termini più prettamente formali, l'SVM ha ottenuto i seguenti risultati di accuratezza per i tre dataset:

- WebLogic dataset: 86%
- Host dataset: 73%
- Database dataset: 80%

E i seguenti valori di precision, recall e f1-score:

Tabella 7.2. Parametri di valutazione per SVM sul dataset WebLogic.

Classe	Precision	Recall	F1 Score
Target Down	0.17	0.81	0.28
Target Up	0.99	0.81	0.89

Tabella 7.3. Parametri di valutazione per SVM sul dataset Host.

Classe	Precision	Recall	F1 Score
Target Down	0.22	0.88	0.36
Target Up	0.97	0.52	0.68

Tabella 7.4. Parametri di valutazione per SVM sul dataset Database.

Classe	Precision	Recall	F1 Score
Target Down	0.19	0.73	0.30
Target Up	0.97	0.74	0.84

I modelli matematici di cui si è fatto il training sono stati quindi serializzati per essere esportati e utilizzati da un modulo secondario del software che permette la deserializzazione degli algoritmi e l'immissione di nuovi dati recuperati in tempo reale dal repository del software di monitoraggio.

8 Conclusioni

Le necessità pratiche dell'ambiente lavorativo, in cui il monitoraggio di sistemi informatici particolarmente complessi è essenziale per il corretto funzionamento di tutte le infrastrutture tecnologiche, hanno portato all'ideazione e allo sviluppo della ricerca descritta. In questo senso il lavoro svolto è nato con l'obiettivo di esplorare le potenzialità del machine learning applicate a sistemi monitorati tramite il software Oracle Enterprise System Manager, le cui funzionalità hanno permesso la raccolta di una grande quantità di dati campionati.
L'applicazione e la successiva ottimizzazione di modelli matematici diversi hanno quindi consentito di analizzare ciascuna famiglia di sistemi (Oracle Database, Oracle System Infrastructure e Oracle WebLogic) separatamente, individuando per ognuno le eventuali criticità presenti: è il caso, per esempio, della diversa distribuzione delle classi che ha portato ad ottenere dataset più sbilanciati di altri e a correggere il problema in maniera diversa a seconda del modello e del dataset preso in considerazione.
Il lavoro si è quindi orientato verso la ricerca del modello preferibile per ciascun tipo di dataset e, riassumendo quanto descritto nel dettaglio nel capitolo 5 "*Applicazione dei modelli predittivi*", si è constatato che il Support Vector Machine offre il miglior compromesso in termini di accuratezza e recall sulla classe Target Down per tutte e tre le tecnologie studiate. La superiorità di questo algoritmo rispetto al Random Forest e al Neural Network è da ricercarsi nella possibilità di associare un peso ad ogni classe analizzata e rendere quindi l'algoritmo più propenso a classificare un campione come Target Down rispetto a Target Up: tale fattore è particolarmente rilevante nel contesto aziendale in quando è preferibile avere un falso allarme piuttosto che un errore non rilevato.
Lo sbilanciamento delle classi tramite oversampling e successivo undersampling, con cui si è cercato di imitare questa peculiarità del SVM, non si è rilevato sufficiente a permettere agli altri algoritmi di raggiungere le stesse performance e, anzi, si è notato che un eccessivo sbilanciamento influisce negativamente e in maniera eccessiva nell'accuratezza del modello. Rimane comunque importante l'aspetto della precisione degli algoritmi studiati e, in particolare, del Support Vector Machine: è da prendere in considerazione, infatti, il numero di falsi Target Down rilevati che diventa particolarmente importante nei dataset WebLogic e Database. Tale fattore rende comunque necessaria la verifica a posteriori delle predizioni effettuata tramite l'intervento umano.
A prescindere dai risultati ottenuti, lo svolgimento della ricerca ha permesso di mettere in luce alcuni limiti del software di monitoraggio utilizzato derivanti soprattutto dalla scarsa frequenza di campionamento dei dati sui sistemi analizzati: con la configurazione standard,

infatti, le metriche prese in considerazione hanno una frequenza di campionamento non superiore ai 5 minuti e tale fattore ha reso necessarie alcune assunzioni, come l'utilizzo di dati aggregati per ora, che hanno reso meno performanti i modelli matematici descritti.
Il lavoro svolto ha quindi permesso di individuare le potenzialità del machine learning applicate nell'ambito del monitoraggio dei sistemi informatici: se da un lato i risultati ottenuti mostrano ampi margini di miglioramento in termini di precisione, raggiungibili con l'utilizzo di dati prelevati in altri modi, dall'altro dimostrano la possibilità concreta di sfruttare tali tecniche illustrate per attuare politiche di manutenzione predittiva e permettono di indirizzare le ricerche future verso lo studio specifico di alcuni modelli matematici rispetto ad altri.

8.1 Futuri sviluppi

Al fine di migliorare le prestazioni dei modelli matematici studiati e permettere quindi una classificazione dei campioni più affidabile si possono attuare alcuni accorgimenti implementabili in future versioni del software creato.

Poiché il principale problema si è infatti rilevato essere la bassa frequenza di campionamento dei dati da parte di Oracle Enterprise Manager, è utile implementare un software in grado di prelevare tali statistiche direttamente sulle macchine in cui sono installate le tecnologie di interesse: tale software deve poter campionare con una frequenza simile tutte le statistiche utili all'analisi e inviarle ad un sistema centralizzato in grado di raccoglierle ed elaborarle mantenendo aggiornati i modelli matematici.

Ulteriore miglioramento si potrebbe ottenere dall'analisi dei log creati dalle singole tecnologie: il parsing e la corretta interpretazione di questi file, aventi strutture e informazioni diverse a seconda del tipo di sistema a cui fanno riferimento, permette di ottenere ulteriori dati che sarebbe possibile incrociare con quelli più prettamente numerici ottenuti dal repository di Oracle Enterprise Manager.

9 Riferimenti

1. **SpiceWorks.** State of IT. *SpiceWorks.* [Online] 28 03 2018. https://www.spiceworks.com/marketing/state-of-it/report/.
2. **Rouse, Margaret.** IT Systems Management. *TechTarget.* [Online] 10 2016. http://searchitoperations.techtarget.com/definition/systems-management.
3. **Rossi, Claudia.** Oracle Italia, seconda al mondo per vendita di sistemi ingegnerizzati. *Tom's Hardware Italia.* [Online] 03 Giugno 2014. [Riportato: 28 Marzo 2018.] https://www.tomshw.it/oracle-italia-seconda-mondo-vendita-sistemi-ingegnerizzati-58498.
4. **Shin, Jong-Ho e Jun, Hong-Bae.** On condition based maintenance policy. [Online] https://doi.org/10.1016/j.jcde.2014.12.006.
5. **Amruthnath, Nagdev & Gupta, Tarun.** A Research Study on Unsupervised Machine Learning Algorithms for Fault Detection in Predictive Maintenance. [Online] https://dx.doi.org/10.13140/RG.2.2.28822.24648.
6. **Hashemian, H. M. e Bean, Wendell C.** State-of-the-Art Predictive Maintenance Techniques. [Online] https://doi.org/10.1109/TIM.2009.2036347.
7. **Koehrsen, William.** Random Forest Simple Explanation. *Medium.com.* [Online] https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d.
8. **Notebook.** The Effects of Hyperparameters in SVM. *Notebook.* [Online] https://yunhaocsblog.wordpress.com/2014/07/27/the-effects-of-hyperparameters-in-svm/.
9. **McDonald, Conor.** Machine learning fundamentals (II): Neural networks. *Towardsdatascience.com.* [Online] https://towardsdatascience.com/machine-learning-fundamentals-ii-neural-networks-f1e7b2cb3eef.
10. **Veličković, Petar.** Deep learning for complete beginners: recognising handwritten digits. *Cambridge Spark.* [Online] https://cambridgespark.com/content/tutorials/deep-learning-for-complete-beginners-recognising-handwritten-digits/index.html.
11. **Mateusz Buda, Atsuto Maki, Maciej A. Mazurowski.** A systematic study of the class imbalance problem in convolutional neural networks. [Online] https://arxiv.org/abs/1710.05381.
12. **Oracle Corporation.** Oracle Enterprise System Manager. *Oracle.com.* [Online] http://www.oracle.com/technetwork/oem/enterprise-manager/overview/index.html.
13. —. Overview of Oracle Enterprise Manager Cloud Control 13c. *Oracle.com.* [Online] https://docs.oracle.com/cd/E73210_01/EMCON/GUID-DC86504F-9F4E-4D5F-92AC-49E898384FB9.htm#EMCON110.
14. —. Enterprise Manager Middleware Plug-in Metric Reference Manual. [Online] https://docs.oracle.com/cd/E73210_01/EMASM/toc.htm.
15. —. Enterprise Manager Oracle Database Plug-in Metric Reference Manual. [Online] https://docs.oracle.com/cd/E24628_01/em.121/e25160/toc.htm.
16. —. Enterprise Manager Framework, Host, and Services Metric Reference Manual. *Oracle.com.* [Online] https://docs.oracle.com/cd/E24628_01/em.121/e25162/toc.htm.

17. —. Management Repository Data Retention Policies. *Oracle.com.* [Online] https://docs.oracle.com/cd/E73210_01/EMADM/GUID-A3CDCD1E-294D-4472-82A7-F2CD30D8D9BA.htm#EMADM12781.
18. **Nitesh V. Chawla, et al.** SMOTE: Synthetic Minority Over-sampling Technique. [Online] https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume16/chawla02a-html/chawla2002.html.
19. **Medium Corporation.** What is underfitting and overfitting in machine learning and how to deal with it. *Medium.com.* [Online] https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76.
20. —. Feature Importance Measures for Tree Models—Part I. *Medium.com.* [Online] https://medium.com/the-artificial-impostor/feature-importance-measures-for-tree-models-part-i-47f187c1a2c3.

10 Ringraziamenti

Con poche e semplici righe desidero ringraziare tutte le persone che mi hanno supportato nella stesura del lavoro e hanno arricchito lo studio con suggerimenti e critiche costruttive.

Ringrazio innanzitutto il Prof. Garza la cui competenza e disponibilità mi hanno permesso di portare a termine il lavoro contando su preziosi consigli tecnici utili per affrontare i problemi che si sono presentati.

Un ringraziamento particolare va poi a tutti i componenti dell'azienda Mediamente Consulting con cui ho svolto il tirocinio: la loro capacità di mettermi a mio agio nella mia prima esperienza lavorativa è stata essenziale per lavorare con serenità, consapevole di poter far affidamento su persone disponibili e preparate.

Desidero quindi ringraziare i miei genitori e mio fratello che in questi anni di università mi hanno supportato nei momenti difficili e hanno festeggiato con me quelli più felici; i parenti e gli amici (vicini e lontani), con cui ho condiviso esperienze che mi hanno fatto crescere e maturare, e infine tutte le persone care che hanno iniziato con me questo viaggio nel mondo universitario ma che adesso purtroppo non possono leggere queste poche righe.