POLITECNICO DI TORINO

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Anomaly Detection sui Sistemi Monitorati dall'Oracle Enterprise Manager

Relatore:
prof. Paolo Garza

Candidato:
Alessandro Tedesco

Supervisore Aziendale
Mediamente Consulting s.r.l
dott. Vincenzo Scinicariello

Anno accademico 2017-2018

Sommario

In un periodo in cui sempre più aziende adottano soluzioni cloud e migrano le proprie infrastrutture IT verso servizi data center gestiti da terze parti, i guasti imprevisti degli host e il degrado delle performance delle applicazioni con relative interruzioni del business, sono le principali problematiche nella gestione di un data center. In questi contesti, malfunzionamenti e problemi di performance sono molto frequenti e quindi, una gestione efficace degli incidenti con enfasi sulla tempestiva rilevazione, diagnosi e risoluzione del problema è diventata una necessità.

Molte delle tecniche di monitoraggio esistenti si basano su regole statiche e non si adattano molto bene ai sistemi informatici complessi su larga scala. In questo senso, il lavoro di tesi proposto si pone come obiettivo quello di studiare e applicare modelli automatici di anomaly detection sui sistemi monitorati attraverso lo strumento Oracle Enterprise Manager (OEM).

Non avendo a priori conoscenze sui dati raccolti dai target monitorati dall'OEM, sono state messe a confronto quattro diverse tecniche di anomaly detection con apprendimento automatico non supervisionato: k-Means, DBSCAN, Isolation Forest e Local Outlier Factor (LOF). L'applicazione di queste tecniche, e conseguentemente il lavoro di ricerca svolto, si basano sull'assunzione che le anomalie, per definizione, sono rare e hanno caratteristiche molto diverse dal resto dei dati del dataset.

Terminata la fase di studio e definito l'insieme di dati da analizzare, attraverso un'applicazione sviluppata in Python, sono stati applicati i quattro algoritmi di anomaly detection sopra citati e messi a confronto le loro performance. Sulla base dei risultati ottenuti, il modello che ha ottenuto la migliore performance è stato il DBSCAN.

I risultati incoraggianti ottenuti da questa ricerca, hanno messo in evidenza la possibilità di applicare le tecniche di data mining sui sistemi monitorati dall'Oracle Enterprise Manager e la possibilità di implementare uno strumento di monitoraggio in grado di rilevare automaticamente anomalie sulle prestazioni dei sistemi in gestione.

Ringraziamenti

Innanzitutto desidero ringraziare il prof. *Paolo Garza* per la grande cortesia, disponibilità e professionalità dimostratemi durante lo svolgimento del lavoro di tesi.

Un particolare ringraziamento a *Mediamente Consulting s.r.l* che mi ha permesso di concludere la carriera universitaria nella più totale tranquillità ed ai colleghi, che mi hanno caricato ed incitato fino al conseguimento del tanto atteso traguardo.

Un ringraziamento doveroso al *Collegio Universitario Renato Einaudi* che per 7 lunghi anni è stata la mia seconda casa e mi ha permesso di crescere e conoscere tantissime persone speciali con cui ho condiviso moltissime avventure. In particolar modo, ringrazio *Federico Constantini* e *Salvatore Cicero* per la splendida e sincera amicizia che ci ha legato fin dal principio e per i momenti indimenticabili trascorsi insieme.

Un caloroso e speciale ringraziamento a mio *padre*, a mia *madre* e a mio *fratello* che ci sono sempre stati e faranno sempre parte della mia vita. Soprattutto li ringrazio perché hanno sempre creduto in me e con il loro incrollabile sostegno morale ed economico, mi hanno permesso di raggiungere questo stupendo traguardo.

Infine, un grazie speciale alla mia compagna *Maria*, la persona che più di tutte è stata capace di capirmi e di sostenermi nei momenti difficili. Grazie per essermi accanto ogni giorno da ben più di 5 anni e spero che questo obiettivo raggiunto, non segni solamente la fine di una lunga carriera universitaria ma anche l'inizio di nuovi progetti insieme.

Indice

Capitolo 1

Introduzione

Il traffico nei data center cloud è destinato a crescere costantemente da qui al 2021, trainato in maniera decisiva dalle applicazioni enterprise, tra le quali compaiono in primo piano i big data analytics, insieme alla pianificazione delle risorse aziendali (ERP) e altre applicazioni enterprise digitali. Queste sono le previsioni contenute nella sesta edizione del Cisco Global Cloud Index [1]. Questa notevole crescita richiederà un'innovazione dell'infrastruttura tecnologica e degli strumenti di gestione dei sistemi informatici in un data center enterprise. Una visione di insieme dei principali componenti che costituiscono la gestione di infrastrutture complesse come un data center è mostrata in Fig. 1.1.

Oggi, gli strumenti di gestione di infrastrutture informatiche complesse, sono sistemi che utilizzano dati in tempo reale per creare allarmi da mostrare agli operatori responsabili del monitoraggio. Quest'ultimi, in caso di situazioni critiche rilevate, sono in grado di reagire immediatamente per risolvere i problemi. Gli strumenti di monitoraggio, tuttavia, basano le loro analisi su soglie di allarme pre-impostate e non apprendono informazioni dai dati storici. Con l'avvento del Machine Learning e dei Big Data è possibile, mediante l'uso di modelli analitici e approcci predittivi, migliorare questi strumenti per fornire ulteriore supporto agli operatori in situazione critiche. In particolare, le informazioni memorizzate nei dati storici potrebbero essere utilizzate come fonte di conoscenza da cui apprendere i comportamenti attesi del sistema.

Nel contesto sopra descritto risulta molto evidente la necessità di ottimizzare, automatizzare e innovare il più possibile le tecniche di manutenzione e monitoraggio delle infrastrutture informatiche. In questo senso, negli ultimi anni, è cresciuto l'interesse nel monitoraggio automatico e l'individuazione di eventi anomali, cambiamenti o deviazioni nei dati raccolti, meglio definito come **Anomaly Detection**. L'anomaly detection costituisce un ramo molto importante nell'ambito delle tecniche di **Data Mining**. Esso presenta numerose applicazioni in diversi contesti, come

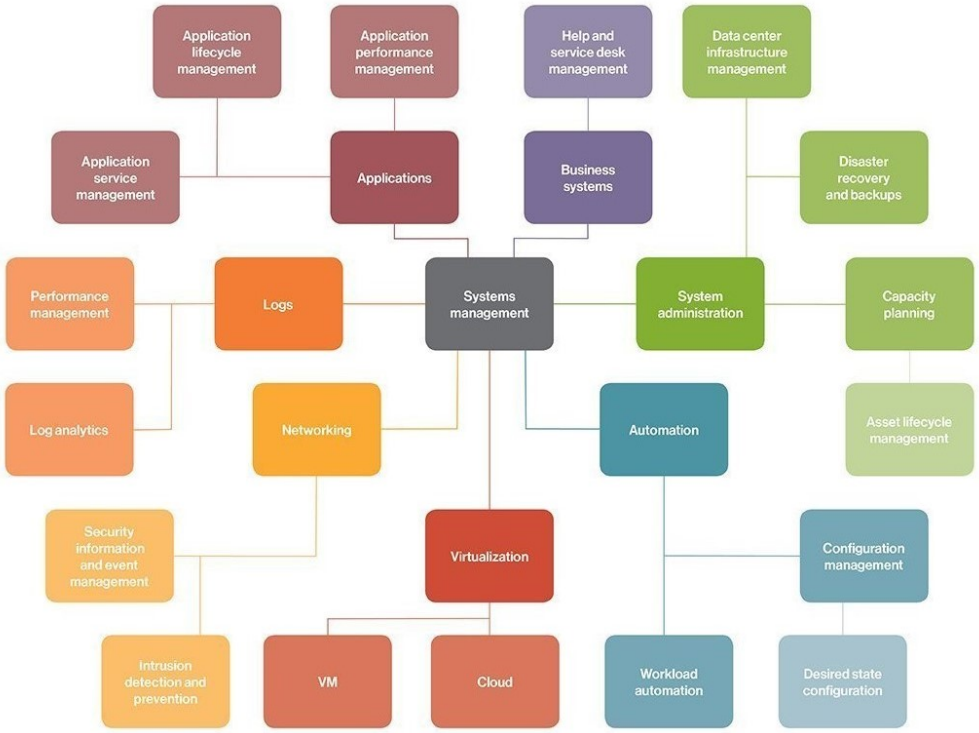

Figura 1.1. A System Management.

nell'individuazione delle frodi, nella rivelazione di intrusioni in sistemi informatici, nei sistemi di supporto alle diagnosi mediche, nel marketing e in molti altri ancora. Con anomaly detection si intende la capacità di identificare elementi o eventi non conformi all'andamento previsto in uno specifico dataset. Generalmente queste anomalie rappresentano l'effetto concreto di problematiche reali, come difetti strutturali, errori di processo, riduzioni delle prestazioni di un macchinario, ecc. Poiché le anomalie generalmente non sono note a priori, le tecniche di anomaly detection di tipo "*non supervisionato*" possono aiutare a identificare potenziali anomalie senza avere a disposizione un set di dati classificati come normali e/o anomali.

1.1 Contesto Aziendale

La tesi è stata svolta presso **Mediamente Consulting s.r.l.**, una società di consulenza, in forte crescita, specializzata nella gestione e valorizzazione dei dati. Mediamente Consulting è stata fondata nel 2013 da riconosciuti professionisti nell'ambito

IT e partecipata da Var Group (gruppo SeSa S.p.A. quotato presso la borsa italiana). In 5 anni di attività, ha sviluppato aree di specializzazione in Corporate Performance Management, Advanced Analytics, Business Intelligence, Data Integration e Management e Infrastruttura tecnologica. Uno dei principali "core business" dell'azienda riguarda la gestione, l'aggiornamento e il tuning di sistemi complessi e ingegnerizzati. In questo senso, Mediamente Consulting tende sempre a migliorare ed innovare le tecnologie attualmente utilizzate per il monitoraggio di infrastrutture complesse di medie e grandi aziende.

Ad oggi, l'azienda utilizza un software enterprise di monitoraggio, basato sulla configurazione di soglie critiche, che consente di informare, attraverso opportune procedure di alerting, i consulenti impiegati nel turno di AMS (Application Management System) in modo tale che essi possano intervenire prontamente sulle criticità rilevate. Purtroppo, non tutti gli alert ricevuti sono sintomi di criticità di un sistema. Basta pensare ad un classico caso in cui, l'utilizzo della CPU di un host supera per pochi istanti la soglia critica impostata, per poi ritornare stabilmente ai suoi valori abituali. In questo caso, il consulente impiegato al monitoraggio ricevuto l'allarme, dovrà essere in grado di valutare, con estrema reattività e precisione, la criticità dell'evento e svolgere un'analisi del problema qualora lo ritenesse opportuno. È chiaro che questo tipo di approccio è strettamente legato in primis alla taratura delle soglie critiche impostate nel software e in secondo luogo, sull'esperienza maturata e sulla confidenza dei sistemi coinvolti del consulente informatico che in quel momento è in turno nell'AMS. Inoltre, l'azienda gestisce un servizio di ticketing che permette ai clienti di segnalare, sulla base dell'esperienza di utilizzo del sistema da parte degli utenti, eventuali fault o problemi di performance critici per il business aziendale in modo tale che possano essere analizzati e risolti nel più breve tempo possibile.

Il lavoro di tesi nasce quindi dalla necessità aziendale di migliorare l'efficienza dei processi di monitoraggio puntando ad implementare metodi alternativi rispetto al sistema attualmente in uso, basato sulla configurazione di soglie critiche, con un approccio pro-attivo verso il cliente cercando di anticipare le segnalazione pervenute tramite ticket e intervenendo più rapidamente ai problemi di performance riscontrati.

1.2 Motivazioni della Ricerca

Nell'ottica di innovazione e di offrire un servizio, sempre migliore, di gestione e monitoraggio dei sistemi informatici da parte dell'azienda verso i suoi clienti, la ricerca vuole essere un progetto pilota per l'applicazione di tecniche di data mining sui dati collezionati tramite il software di monitoraggio utilizzato in azienda, l'Oracle Enterprise Manager. L'idea alla base, è quella di utilizzare modelli di anomaly detection sui dati raccolti, nel tentativo di individuare un insieme di dati nascosti che possano essere identificati come anomalie.

L'obiettivo della ricerca è quindi quello di approfondire la fattibilità dell'applicazione di modelli automatici di anomaly detection nell'ambito delle infrastrutture informatiche, gestite e monitorate nel contesto aziendale. In questo senso, la ricerca vuole fornire, al gruppo dell'AMS, un ulteriore strumento di monitoraggio che consente di scoprire anomalie nascoste e intervenire tempestivamente sul problema rilevato. Le principali fasi della ricerca possono così essere riassunte:

1. Studio delle principali tecniche di anomaly detection;
2. Identificazione dei dati di interesse per l'applicazione delle tecniche di data mining;
3. Confronto dei modelli presi in considerazione e scelta della tecnica con le migliori prestazioni.

1.3 Struttura della Tesi

Questa sezione ha lo scopo di descrivere, sinteticamente, l'organizzazione strutturale del lavoro di tesi. I prossimi capitoli sono stati organizzati come segue:

- Il capitolo successivo, descrive brevemente i punti di forza e le limitazioni delle tecniche di anomaly detection presenti in letteratura e i principali componenti e funzionalità dello strumento di monitoraggio Oracle Enterprise Manager;
- Il capitolo 3 illustra come è stato trattato l'argomento di tesi, le diverse fasi che sono state seguite durante il lavoro di ricerca e le ipotesi su cui si basano i principali modelli di anomaly detection;
- La fase di progettazione e le scelte implementative sono illustrate nel capitolo 4. In particolare, vengono descritte le varie fasi di costruzione e preparazione del dataset e i principali parametri da mettere a punto per i modelli di anomaly detection considerati;
- I risultati ottenuti durante la fase sperimentale sono presentati e argomentati nel capitolo 5;
- Infine, il capitolo 6 tratta le conclusioni e indica i possibili sviluppi futuri del lavoro.

Capitolo 2

Stato dell'Arte

2.1 Anomaly Detection

Anomaly Detection, noto anche come **Outlier Detection**, è il processo di identificazione di elementi, eventi o osservazioni con comportamenti anomali rispetto alla norma. Tali entità sono chiamate **anomalie** o **outlier**. Il processo di anomaly detection implica varie sfide tra cui definire quali dati costituiscono un comportamento normale e quali un'anomalia. Di seguito sono elencate le principali difficoltà che bisogna affrontare durante lo sviluppo di un sistema di anomaly detection:

- In alcuni casi, il confine tra comportamento normale e anormale non è molto definito, per cui modelli di anomaly detection potrebbero produrre molti falsi positivi e falsi negativi;
- Aree di domini diverse contengono diversi tipi di anomalie. Spesso accade che un'anomalia in un dominio può essere un comportamento normale in un altro e viceversa;
- Insieme di dati contrassegnati come normali o anomali non sono quasi mai disponibili.

Una semplice rappresentazione di anomaly detection, applicato su un dataset bidimensionale, è mostrata in Fig. 2.1. Come è possibile notare graficamente, il dataset è composto da due aree "normali", N1 e N2, in quanto la maggior parte dei dati si collocano in queste due regioni mentre i punti sufficientemente distanti da queste regioni, come o1, o2, e il gruppo di dati O3, sono considerate delle anomalie.

L'anomaly detection è ampiamente utilizzato in diversi domini di applicazione, dalla rilevazione di intrusioni informatiche (analizzando il traffico di rete) al monitoraggio di sistemi informatici complessi come un data center, dal rilevamento

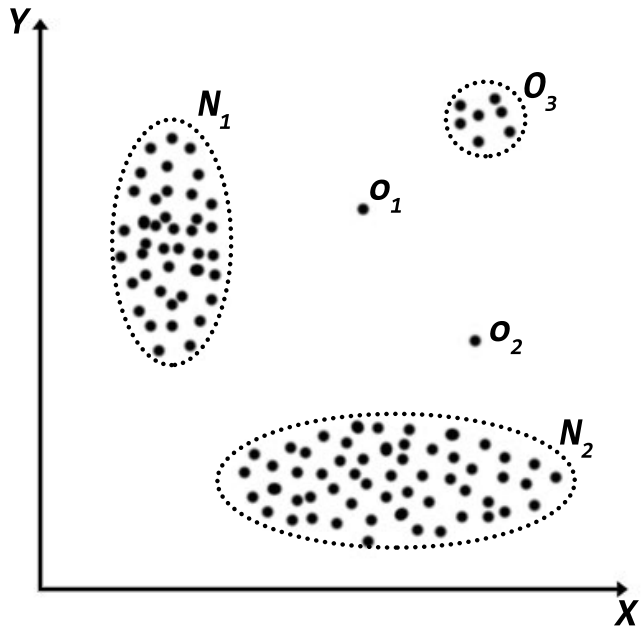

Figura 2.1. Esempio anomaly detection in dataset bi-dimensionale [3].

di frodi nelle transazioni con carte di credito al rilevamento dei guasti in processi industriali. Nell'esempio di frodi nelle transazioni con carte di credito, si parte dal presupposto che, la maggior parte delle transazioni con carta di credito siano delle transazioni normali. Nel momento in cui una carta di credito viene rubata, l'importo dell'acquisto o la posizione geografica dei successivi record potrebbero essere molto diversi dallo storico degli acquisti del proprietario autenticato e quindi identificate come anomalie. Tuttavia, non è cosi banale identificare le "vere" anomalie. Alcune anomalie sospette possono rivelarsi falsi negativi in contesti diversi e alcuni comportamenti "normali" possono rivelarsi falsi positivi se non vengono presi in considerazione determinati fattori. Infatti, riprendendo l'esempio precedente, è possibile che il proprietario della carta di credito effettui, in un dato momento, un acquisto di un prodotto costoso come un computer portatile mentre in passato ha solo acquistato prodotti economici come cibi e vestiti. In questo caso, l'acquisto non dovrebbe essere rilevato come un'anomalia.

2.1.1 Tipi di Anomalie

Prima di argomentare le principali tecniche di anomaly detection, è importante stabilire dei limiti nella definizione di un'anomalia.

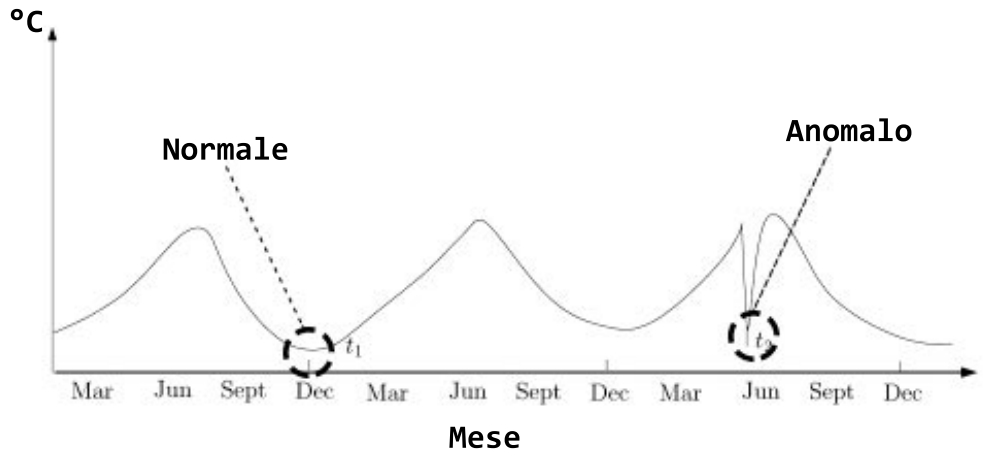

Figura 2.2. Esempio di anomalia contestuale [3].

Cosa è un'anomalia?

Un'anomalia è un'osservazione che devia molto dalla maggior parte delle altre osservazioni, può essere vista come un punto o comportamento o pattern che risulta essere statisticamente "inusuale" o "anomalo" rispetto all'intero dataset. Le anomalie possono essere classificate in:

- **Anomalia puntuale**: quando una singola istanza di dati è estremamente diversa e distante dal resto del dataset. Un tipico esempio è l'identificazione delle frodi con carte di credito basate sull'importo speso delle singole transazioni. In questo caso, l'importo di una transazione estremamente elevato rispetto agli importi storici è considerata un'anomalia puntuale. In Fig. 2.1, i punti o1 e o2 sono considerati come anomalie puntuali.

- **Anomalia contestuale**: quando una singola istanza è considerata un'anomalia solo in specifici contesti. In questi casi, il contesto deve essere specificato come parte della definizione del problema. Questo tipo di anomalia si presenta, solitamente, nei dati influenzati dal tempo. Un classico esempio, illustrato in Fig. 2.2, è la rilevazione di una temperatura di 5° a Torino che può essere considerata un'anomalia se rilevata durante la stagione estiva (es. nel mese di Giugno) ma nella norma se si considera la stagione invernale (es. nel mese di Dicembre).

- **Anomalie collettive**: quando un gruppo di istanze di dati assumono un comportamento anomalo rispetto al resto del dataset ma non individualmente.

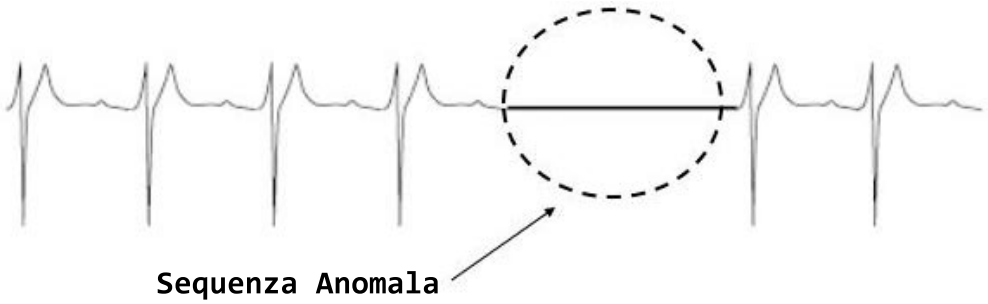

Figura 2.3. Esempio di anomalie collettive [3].

Tipico esempio, illustrato in Fig. 2.3, è la sequenza anomala registrata durante un esame di ECG.

2.1.2 Tecniche di Anomaly Detection

In letteratura esistono diversi modelli di anomaly detection e in questa sezione vengono presentati due possibili modi di categorizzare queste tecniche. Una prima classificazione si basa sulla natura del dataset che si utilizza per la costruzione di un modello di anomaly detection. A seconda se i campioni di dati di un dataset vengono forniti con delle etichette assegnate da esperti in materia, le tecniche di anomaly detection vengono classificate in modelli con apprendimento supervisionato, semi supervisionato e non supervisionato. Un altro modo di classificare questi modelli si basa sui metodi di separazione delle anomalie rispetto al resto del dataset. In questo caso possiamo classificarli in tre tipi: metodi statistici, metodi basati sulla prossimità e metodi basati sul clustering.

Modelli con Apprendimento Supervisionato, Semi Supervisionato e Non Supervisionato

La principale differenza tra le tre tipologie di apprendimento è la natura del dataset utilizzato in fase di addestramento del modello. Quando ad ogni singola istanza del dataset è associata un'etichetta predefinita, parliamo di modelli con apprendimento supervisionato, viceversa quando il dataset non contiene alcuna etichetta ci troviamo nel caso di modelli con apprendimento non supervisionato. Si parla invece di modelli semi supervisionati, quando una piccola porzione di istanze del dataset viene fornita etichettata mentre il resto dei dati senza etichette. Il processo di etichettatura di un dataset viene spesso eseguito manualmente da esperti del settore e richiede, solitamente, un notevole sforzo sia in termini di analisi che in termini di tempo.

In generale, a causa del processo oneroso di etichettatura dei campioni di dati, in scenari reali, non sempre si hanno a disposizione dataset etichettati. Di seguito, sono descritte brevemente le principali caratteristiche di ciascun modello:

Anomaly Detection Supervisionato

I modelli di anomaly detection supervisionati sono utilizzati quando i campioni di dati, che costituiscono il dataset, sono stati etichettati come "normali" o "anomali" da esperti in materia [4]. Tipico approccio in questi casi è trattare l'anomaly detection come un problema di classificazione. Sebbene in letteratura esistono molti modelli di classificazione da applicare, le principali sfide per l'anomaly detection supervisionato sono le seguenti:

- Le due classi (cioè normale vs anomala) sono tipicamente sbilanciate. Ovvero, la popolazione di dati etichettati come anomali è generalmente molto minore rispetto ai dati normali. In questi casi, metodi per la gestione dello sbilanciamento delle classi devono essere implementati. Un tipico metodo è l'oversampling delle anomalie il quale, ha il compito di aumentare la loro distribuzione nel training-set utilizzato per la costruzione del classificatore. La mancanza di campioni anomali può limitare l'efficienza dei classificatori.

- In molte applicazioni di anomaly detection, identificare il maggior numero di anomalie è molto più importante che classificare erroneamente un comportamento normale come anomalia. Di conseguenza, in fase di valutazione, quando un modello di classificazione è utilizzato come anomaly detection supervisionato bisognerebbe dare più peso al richiamo per minimizzare la perdita di identificazione delle anomalie. In ambito della classificazione, il "richiamo" detto anche "recupero", è definito come il numero dei veri positivi diviso il numero totale di elementi che appartengono alla classe su cui si sta calcolando questa metrica.

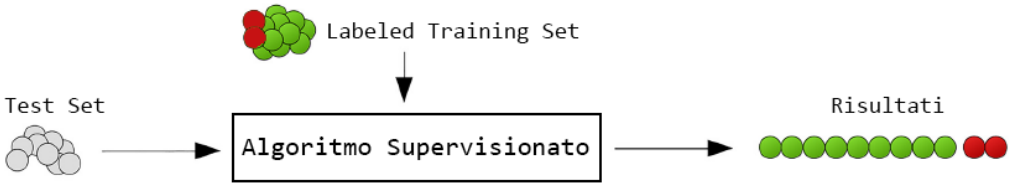

Figura 2.4. Anomaly detection supervisionato.

Anomaly Detection Semi Supervisionato

A differenza dei metodi supervisionati dove l'intero dataset è etichettato, i modelli semi supervisionati si basano su dati misti in cui una minima parte è già etichettata e la maggioranza è costituita da dati non etichettati [4]. L'idea alla base è quella di sfruttare la piccola porzione di dati etichettati per identificare tra i dati non etichettati quelli che più si somigliano. Questi metodi nascono principalmente per il fatto che il processo di etichettatura dei dati è un'operazione onerosa, sia in termini di tempo che di costi, in quanto è necessario l'intervento di un esperto o analista.

Anomaly Detection Non Supervisionato

Negli scenari in cui dati etichettati come "normali" o "outlier" non sono disponibili, vengono utilizzati i metodi di anomaly detection con apprendimento non supervisionato [4]. Questi modelli assumono implicitamente che i dati normali sono in qualche modo "clusterizzabili" o che in altre parole, i dati normali seguono un pattern molto più frequentemente delle anomalie. Il principale vantaggio di questo metodo è che il processo di etichettatura non è necessario. Invece, lo svantaggio principale è che, se l'assunzione iniziale non è soddisfatta, il modello soffre di un gran numero di falsi negativi e falsi positivi.

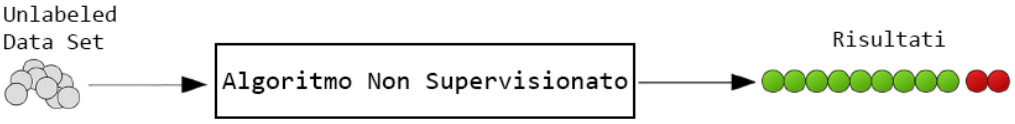

Figura 2.5. Anomaly detection non supervisionato.

Metodi Statistici, basati sulla Prossimità e basati sul Clustering

Come discusso in precedenza, sulla base dei metodi utilizzati per la separazione dei campioni anomali rispetto al resto dei dati, le tecniche di anomaly detection possono essere classificate in tre tipi: metodi statistici, metodi basati sulla prossimità e metodi basati sul clustering. Di seguito, sono descritte brevemente le principali caratteristiche di ciascun modello.

Metodi Statistici

Metodi statistici, noti anche come **metodi basati sul modello**, ipotizzano che i dati normali vengono generati da processi stocastici e che conseguentemente i dati

normali si verificano nelle regioni di alta probabilità del modello mentre i dati che si presentano nelle regioni di bassa probabilità vengono considerati "outlier" [4]. L'efficacia di questi modelli dipende fortemente da quanto il modello statistico è in grado di adattarsi al dataset da analizzare.

Metodi basati sulla Prossimità

Metodi basati sulla prossimità ipotizzano che le anomalie si collocano lontane dal gruppo di osservazione più vicino (nearest neighbors) [4]. L'efficacia di questi metodi dipende fortemente dal tipo di metrica di distanza utilizzata. Le tecniche di anomaly detection basate sulla prossimità possono essere classificate, a loro volta, in **modelli basati sulla distanze** e **modelli basati sulla densità**:

- Un modello basato sulla distanza, si basa su quanto un'osservazione è lontana rispetto ai suoi vicini. Se la distanza dai suoi vicini supera una certa soglia, sarà considerata come un'anomalia.
- Invece, un modello basato sulla densità mette a confronto la densità del campione e il suo vicinato. Se la sua densità è molto inferiore a quella dei suoi vicini, sarà trattata come un'anomalia.

Metodi basati sul Clustering

I modelli basati sul clustering ipotizzano che i dati normali appartengono a cluster grandi e densi, mentre le anomalie appartengono a cluster piccoli e sparsi oppure non appartengono a nessun cluster [4]. L'efficacia di questi metodi dipendono molto dalla relazione tra i dati e i cluster. In questo metodo, i campioni di dati vengono considerati outlier al verificarsi di una delle tre condizioni sotto elencate:

- Se un campione di dato non appartiene a nessun cluster, dovrebbe essere considerato come un'anomalia;
- Se un'osservazione è molto distante dal cluster più vicino, dovrebbe essere considerato come un outlier;
- Se un dato appartiene ad un cluster piccolo e sparso allora l'intero cluster dovrebbe essere considerato come anomalo.

Il principale svantaggio dei modelli di anomaly detection basati sul clustering è l'operazione onerosa di identificazione dei cluster che rendono queste tecniche difficilmente applicabili quando la cardinalità del dataset è molto elevata.

2.1.3 Algoritmi

Questa sezione approfondisce brevemente gli algoritmi di anomaly detection che sono stati presi in considerazione durante lo svolgimento del lavoro di testi. Gli algoritmi utilizzati sono i seguenti: k-Means, DBSCAN, Isolation Forest e Local Outlier Factor. Per ciascun modello, di seguito, sono descritti l'idea alla base e i principali passaggi eseguiti dall'algoritmo, i punti di forza e le limitazioni.

k-Means

Per la sua semplicità e velocità, l'algoritmo di clustering **k-Means** [5] è un buon punto di partenza come modello di anomaly detection non supervisionato. k-Means è un algoritmo di clustering che permette di suddividere un insieme di dati in k gruppi sulla base dei loro attributi e delle loro caratteristiche. L'algoritmo si pone come obiettivo quello di minimizzare la varianza totale all'interno del singolo cluster.

I principali passaggi dell'algoritmo sono i seguenti: fissato a priori un numero di cluster **k**, l'idea principale dell'algoritmo è definire **k** centroidi, uno per ciascun cluster. Il risultato finale dipende fortemente dalla scelta iniziale dei centroidi, solitamente la scelta migliore è quella di metterli il più lontano possibile l'uno dall'altro. Dopo una prima fase di inizializzazione, l'algoritmo consiste in un loop di altri due passaggi. Il primo passo assegna ogni campione di dati al centroide più vicino. Il secondo passo, genera k nuovi centroidi dei cluster prendendo, per ciascun cluster generato al precedente step, il valore medio di tutti i suoi campioni. Il ciclo continua fino a quando non verranno apportate ulteriori modifiche o in altre parole fino a quando i centroidi non cambiano più posizione.

Algorithm 1: Pseudocodice k-Means

Input:
D: dataset di n campioni di dati;
k: numero di cluster;

Output:
Insieme di k cluster;

```
1 Sceglie arbitrariamente k dati da D come centroidi iniziali;
2 while Il valore medio dei cluster varia do
3 |   (Ri) assegna ogni dato al centroide più vicino;
4 |   Genera i nuovi k centroidi prendendo il valore medio di ogni cluster;
5 end
```

Punti di forza dell'algoritmo

- L'algoritmo è semplice, intuibile e converge velocemente;
- Essendo un algoritmo di clustering non supervisionato, non necessita di un dataset etichettato.

Limitazioni dell'algoritmo

- Il numero di cluster k deve essere fissato a priori;
- Generalmente, si ottengo degli ottimi risultati solamente quando i cluster all'interno del dataset da analizzare sono ben separati e di forma sferica;
- Estremamente sensibile alla presenza di anomalie e rumore nel dataset.

DBSCAN

A differenza del k-means precedentemente descritto, il quale modella i cluster come insiemi di dati vicini ai loro centroidi, algoritmi basati sulla densità come il **DBSCAN** [6] (Density-Based Spatial Clustering of Applications with Noise) identificano i cluster come gruppi di dati ad alta densità separati da aree a bassa densità di punti.

L'idea principale del DBSCAN è che, ogni punto appartenente ad un cluster, deve essere circondato in un area di raggio **eps** da almeno un numero minimo di punti **MinPts**. Questi due importanti parametri, richiesti dall'algoritmo in fase di inizializzazione, definiscono rispettivamente il raggio di vicinanza attorno ad un punto x e il numero minimo di punti che devono coesistere all'interno dell'area di raggio eps per poter formare un cluster. Praticamente, questi due parametri definiscono il concetto di densità adottato dal DBSCAN.

Prima di approfondire i vari step eseguiti dall'algoritmo per l'identificazione dei cluster, occorre comprendere meglio alcuni concetti chiavi del DBSCAN. Innanzitutto, l'algoritmo definisce tre tipi di punti (illustati in Fig. 2.6):

1. Ogni punto x del dataset che contiene, all'interno della sua regione di raggio eps, un numero di dati maggiore o uguale a MinPts è detto **core point**;
2. Un dato x è considerato **border point**, se il numero dei suoi vicini all'interno della sua regione di raggio eps è inferiore a MinPts ma il dato stesso è collocato all'interno di un'area di un qualsiasi core point;

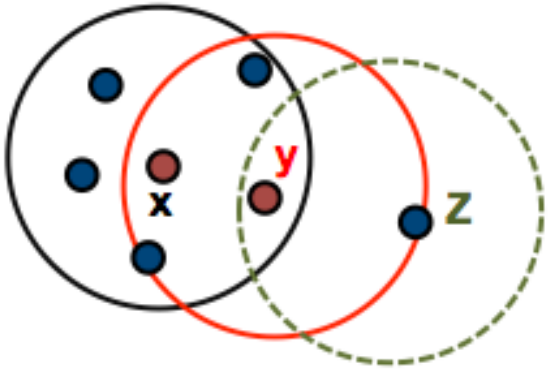

Figura 2.6. Rappresentazione di core point, border point e outlier con MinPts=6. In questo esempio, x è un core point, y è un border point e z è un outlier. (fonte: sthda.com)

3. Tutti i restanti punti che non sono né core point, né border point sono considerati **outlier**.

Altri concetti chiavi che necessitano di una definizione sono:

- **Direct Density Reachable**: un punto A è direct density reachable da un punto B, se A è all'interno dell'area di raggio eps di B e B è un core point;
- **Density Reachable**: un punto A è density reachable da B, se esistono un insieme di core point che vanno da B a A;
- **Density Connected**: i punti A e B sono density connected se esiste un core point C tale che sia A che B sono density reachable da C.

I principali passaggi dell'algoritmo sono i seguenti: per ogni punto x, viene calcolata la distanza tra x e gli altri punti del dataset. Tutti i punti che all'interno della loro area di raggio eps contengono un numero di dati maggiore o uguale a MinPts, vengono marcati come core point. Dopodiché, per ogni core point non ancora assegnato ad un cluster, crea un nuovo cluster. A questo punto, ricorsivamente, cerca tutti i punti density connected e li assegna allo stesso cluster del core point. L'algoritmo termina non appena tutti i punti sono stati marcati come visitati. Infine, tutti i

punti che non appartengono a nessun cluster sono considerati outlier.

Algorithm 2: Pseudocodice DBSCAN

Input:
D: dataset di n campioni di dati;
eps: raggio di vicinanza attorno ad un dato;
MinPts: numero minimo di dati per formare un cluster;

Output:
Insieme di n cluster;

```
1 Calcola la distanza tra i dati del dataset D e marca i core point;
2 foreach Dato d non ancora visitato nel dataset D do
3 |   Contrassegna d come visitato;
4 |   if d è un core point then
5 |   |   Crea un nuovo cluster c;
6 |   |   Cerca tutti i dati density connected, li assegna al cluster c e li
  |   |     marca come visitati;
7 |   end
8 end
9 Tutti i punti che non sono stati assegnati ad un cluster sono considerati
    anomali;
```

Punti di forza dell'algoritmo

- A differenza del k-Means, DBSCAN non richiede di specificare a priori il numero di cluster k da generare;
- Possono essere identificati cluster di forma arbitraria;
- Identifica automaticamente gli outlier.

Limitazioni dell'algoritmo

- I parametri eps e MinPts devono essere fissati a priori e spesso sono difficili da determinare;
- L'algoritmo va in difficoltà quando deve identificare e separare cluster di densità simile.

Isolation Forest

L'**Isolation Forest** [7] (iForest) è un algoritmo non supervisionato di anomaly detection fondamentalmente molto diverso dagli altri metodi presenti in letteratura. Questo modello introduce l'uso dell'**isolamento** come mezzo più efficace ed efficiente per identificare le anomalie rispetto alle misure di distanza e di densità. L'iForest punta sul fatto che, le anomalie sono per definizione poche e diverse rispetto agli altri dati del dataset. Queste caratteristiche rendono le anomalie più sensibili al concetto di isolamento rispetto ai campioni di dati "normali" ed è l'idea principale su cui si basa l'algoritmo. L'iForest misura la deviazione di un punto, determinando quanto tempo ci impiega per isolarlo dal resto dei dati del dataset.

Il modo in cui l'algoritmo costruisce l'isolamento consiste, innanzitutto, nel creare alberi casuali di isolamento binari detti **iTree** e successivamente, nel partizionare ricorsivamente le istanze di dati. Come illustrato in Fig. 2.7, è possibile osservare come un punto "normale" x_i richieda generalmente più partizioni per essere isolato rispetto ad un punto "anomalo" x_0.

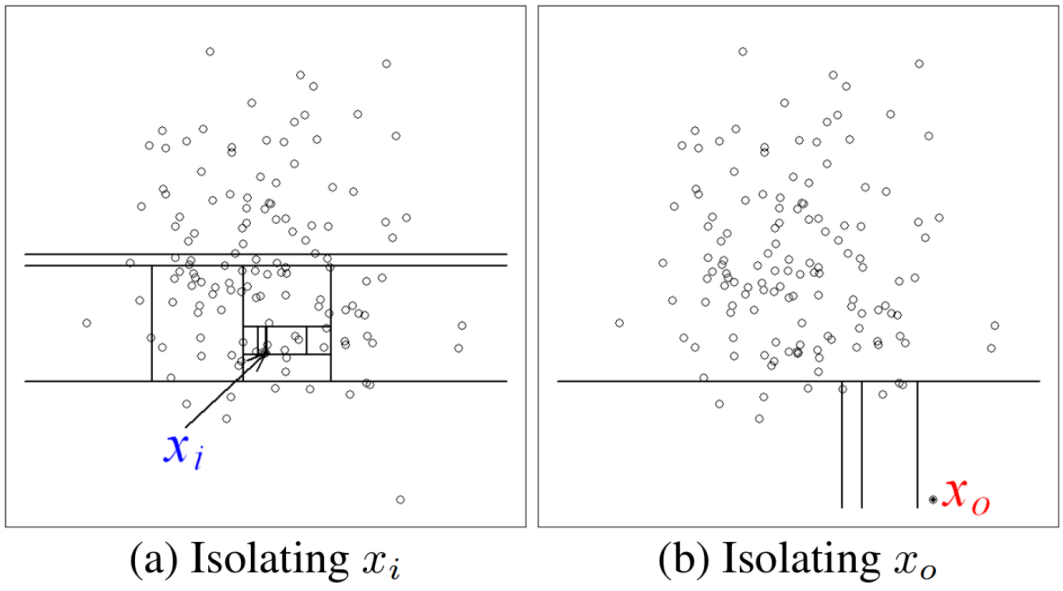

Figura 2.7. Isolamento di un punto "normale" x_i e uno "anomalo" x_0 [7].

Le partizioni, come mostrato in Fig. 2.8, vengono generate selezionando in modo causale un attributo e un valore tra il minimo e il massimo dell'attributo selezionato. Poiché il partizionamento può essere rappresentato da una struttura ad albero, il numero di partizioni necessarie per isolare un punto equivale alla lunghezza del percorso tra il nodo radice e il nodo foglia.

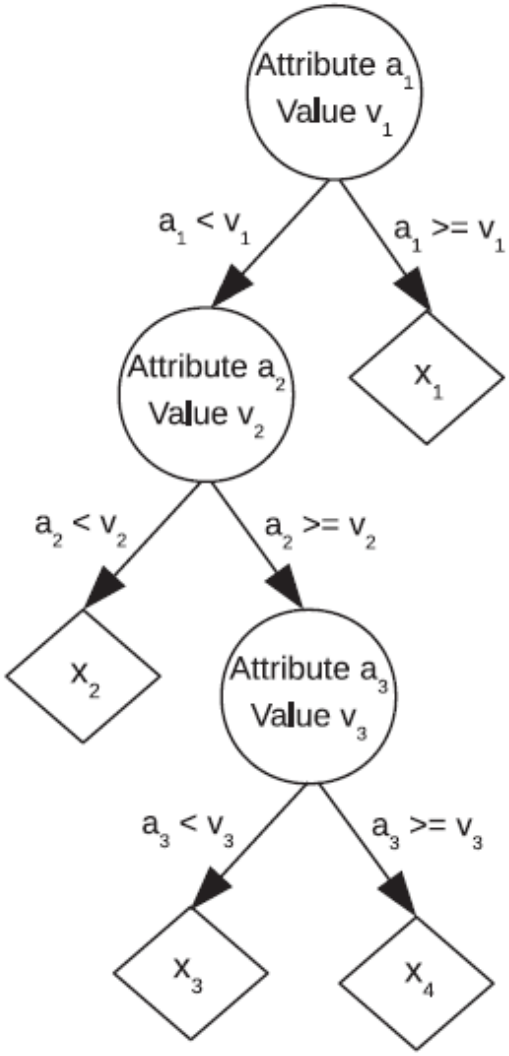

Figura 2.8. In questo esempio di iTree, i vertici dell'albero rappresentano gli attributi a e i valori divisori v. I nodi terminali sono campioni di dato. Le anomalie hanno maggiori probabilità di essere isolate più vicino alla radice dell'albero rispetto ai punti normali. In questo esempio, solo il punto dati x_1 ha un valore per l'attributo a_1 maggiore di v_1.

Per un determinato dato, la media della lunghezza del percorso ottenuta dalla foresta di iTree è la misura di normalità del campione di dato (Fig. 2.9). Più il percorso è breve, maggiore è la probabilità che quel campione di dato sia un'anomalia.

Una semplificazione della logica seguita dall'Isolation Forest, per la costruzione degli iTree, è presentata nello pseudo-codice di seguito.

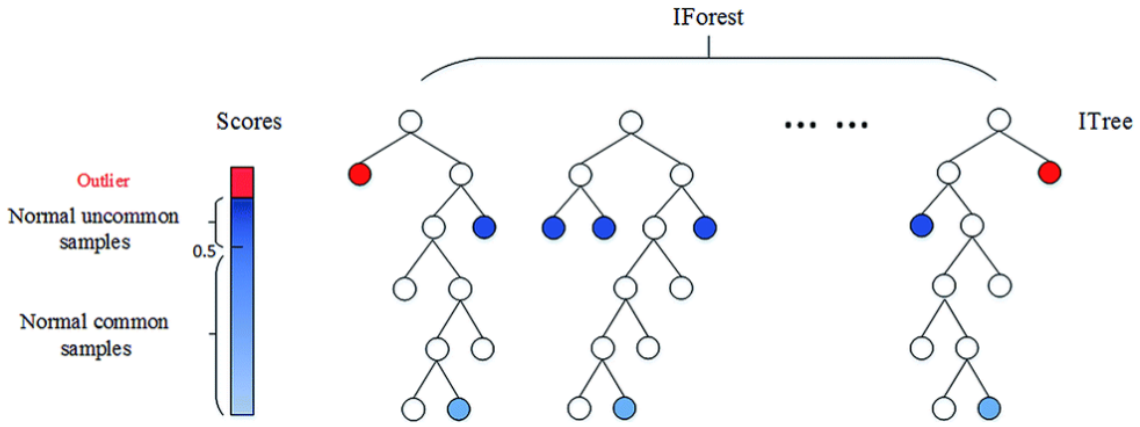

Figura 2.9. Foresta di iTree.

Algorithm 3: Pseudocodice costruzione iTree

Input:
D: subset di n campioni di dati;

Output:
iTree;

```
1 while Tutti i dati d nel subset D non sono isolati do
2 |   Considera A come la lista degli attributi in D;
3 |   Sceglie casualmente un attributo a in A;
4 |   Sceglie casualmente un valore v compreso tra il massimo e il minimo di
  |     a in D;
5 |   Genera due archi a < v e a >= v;
6 |   Ri-esegue ricorsivamente la procedura sui due nuovi subset generati
  |     dalle condizioni del punto precedente;
7 end
```

Punti di forza dell'algoritmo

- L'algoritmo scala molto bene con dataset che hanno un'elevata cardinalità e un'alta dimensionalità.

- Richiede risorse di memoria ridotte e complessità computazionale lineare.

Local Outlier Factor

Local Outlier Factor [8] (LOF) è un algoritmo non supervisionato di anomaly detection basato sulla densità. L'idea alla base dell'algoritmo LOF è di rilevare i campioni che hanno una densità sostanzialmente inferiore rispetto ai loro vicini. In altre parole, LOF si basa sul presupposto che un'osservazione normale ha una densità locale simile a quella dei vicini, mentre i dati anomali dovrebbero avere una densità locale molto più bassa. L'idea di base è graficamente descritta in Fig, 2.10.

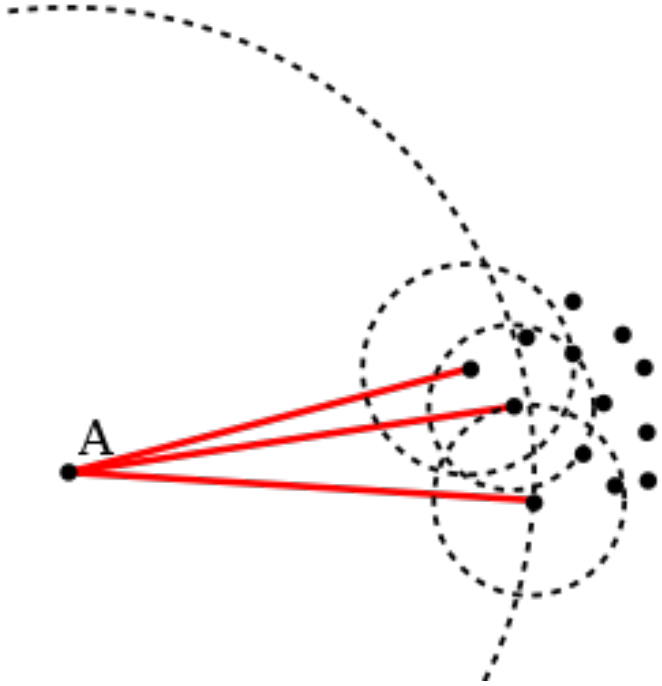

Figura 2.10. Idea di base del LOF: confronto della densità locale di un punto con le densità dei suoi vicini. A ha una densità molto inferiore rispetto ai suoi vicini (fonte: wikipedia.org).

L'algoritmo calcola un punteggio (detto "local outlier factor") che riflette il grado di anormalità delle osservazioni. Il punteggio di un'osservazione viene calcolato come il rapporto tra la densità locale media dei suoi k-nearest neighbors e la sua densità locale. Intuitivamente, i dati normali che appartengono a regioni dense hanno un punteggio di LOF prossimo a 1, mentre i dati con alti valori di LOF » 1 sono considerati anomali. Una versione semplificata dei principali passaggi dell'algoritmo LOF è illustrata di seguito:

Algorithm 4: Pseudocodice LOF

Input:
D: dataset di n campioni di dati;
k: numero di vicini da considerare;

Output:
Insieme di punti anomali e normali;

```
1  foreach Dato d nel dataset D do
2  |  Calcola k-distance e k-distance neighborhood di d;
3  |  Calcola local reachability density di d;
4  |  Calcola punteggio LOF come il rapporto tra local reachability density
   |   di d e dei suoi k-nearest neighbors;
5  |  if LOF » 1 then
6  |  |  Considera il punto come anomalia;
7  |  else
8  |  |  Considera il punto come normale;
9  |  end
10 end
```

Punti di forza dell'algoritmo

- La forza dell'algoritmo LOF è che prende in considerazione le proprietà locali e globali del dataset.

Limitazioni dell'algoritmo

- Il parametro k neighbors deve essere fissato a priori;
- Complessità quadratica temporale dell'algoritmo.

2.1.4 Metodi di Valutazione

La valutazione prestazionale di un algoritmo di clustering e di anomaly detection non è così semplice e banale come calcolare la precisione e il richiamo di un algoritmo di classificazione supervisionato. Quando le etichette di un dataset non sono note a priori, la valutazione deve essere eseguita utilizzando il modello stesso. In base alle conoscenze acquisite, in letteratura non esistono molte metriche di valutazione degli algoritmi di clustering. Uno dei metodi più noti in questo campo è il "**Coefficiente di Silhouette**".

Coefficiente di Silhouette

Il coefficiente di Silhouette è una misura di quanto un dato sia simile al cluster di appartenenza rispetto ad altri cluster. Il valore di Silhouette s per un singolo dato è calcolato come:

$$s = \frac{b - a}{max(a, b)}$$

dove:

- **a** è la distanza media tra un campione di dato e gli altri dati appartenenti allo stesso cluster.
- **b** è la distanza media tra un campione di dato e gli altri punti del cluster più vicino.

Il punteggio di Silhouette è limitato in un intervallo che va tra -1 e +1. I valori prossimi a -1 indicano una clusterizzazione errata, mentre i valori prossimi a +1 indicano che le tecniche di clustering hanno identificato correttamente i gruppi di dati più simili tra loro. I punteggi intorno allo zero indicano invece cluster sovrapposti. Generalmente, valori negativi di Silhouette indicano che un campione di dato è stato assegnato al cluster sbagliato e che quindi un altro cluster risulta essere più similare.

2.2 Oracle Enterprise Manager

L'Oracle Enterprise Manager è una piattaforma sviluppata da Oracle Corporation che negli anni, grazie a successive acquisizioni di aziende, è divenuta leader mondiale in diversi settori dell'informatica. Questo capitolo ha lo scopo di descrivere brevemente l'architettura, le funzionalità e le limitazioni del prodotto.

2.2.1 Architettura

L'OEM è una soluzione enterprise che fornisce una piattaforma basata su interfaccia web per la gestione e il monitoraggio di prodotti Oracle tra cui applicazioni, database, middleware, hardware e sistemi ingegnerizzati.

Come mostrato in Fig. 2.11, l'architettura dell'OEM include i seguenti componenti principali:

- Enterprise Manager Cloud Control Console;

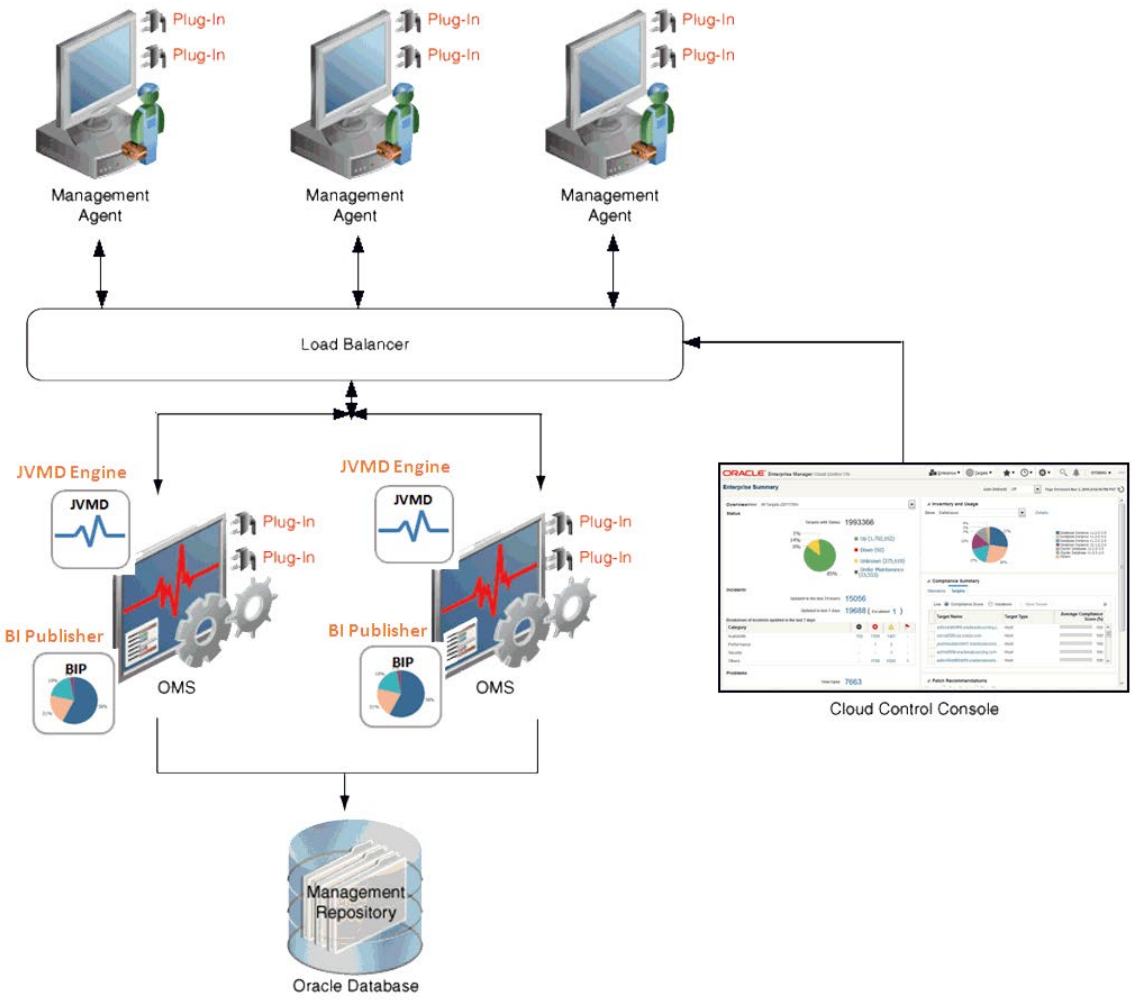

Figura 2.11. Architettura Oracle Enterprise Manager (fonte: oracle.com).

- Oracle Management Service;
- Oracle Management Agent;
- Oracle Management Repository;
- Plug-ins.

Enterprise Manager Cloud Control Console

La console è il componente che fornisce l'interfaccia web all'utilizzatore del prodotto. Essendo l'OEM un sistema centralizzato, con l'aiuto della console, è possibile

monitorare e gestire facilmente tutte le risorse di un sistema da un'unica posizione.

Oracle Management Service

L'OMS è l'anima dell'architettura dell'OEM ed è il componente che, attraverso gli agent e i plug-in, gestisce e controlla i target sotto monitoraggio ed archivia le informazioni raccolte in un repository per riferimenti e analisi future.

Oracle Management Agent

L'agent è il modulo che viene installato negli host da monitorare. Il principale compito che svolge questo componente è di comunicare costantemente con l'OMS le informazioni raccolte sulle risorse dell'host su cui è installato.

Oracle Management Repository

Il repository è il componente storage dove vengono archiviate tutte le informazioni raccolte dagli agent installati sugli host monitorati. È composto da oggetti come: database jobs, packages, procedure, viste e tablespace. Una descrizione più dettagliata di questo modulo, con particolare attenzione sulle viste, sarà fornita nella sezione successiva.

Plug-ins

I plug-in sono moduli customizzati per consentire di monitorare specifici tipi di target tramite l'OMS e gli agent. Di default, con l'installazione base dell'OMS e degli agent, vengono distribuiti i plug-in per la gestione e il monitoraggio dei seguenti prodotti Oracle:

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

2.2.2 Oracle Management Repository

L'Oracle Management Repository è il database utilizzato dall'OEM per l'archiviazione delle metriche raccolte dai target monitorati.

Per "**target**" si intende un'entità che l'OEM è in grado di amministrare e monitorare. Esempi di target sono: host, database, applicazioni, application server, listener, ecc... Un elenco più ampio dei tipi di target gestiti dall'OEM sono presentati in Tab. 4.3.

Per "**metrica**" si intende una misura quantitativa di una risorsa tenuta sotto monitoraggio dall'OEM. Esempi di metriche per i target host sono: utilizzo CPU, utilizzo memoria RAM, utilizzo memoria SWAP, ecc...

Le Metriche

L'OEM, per ciascun tipo di target, vanta una vasta gamma di metriche che complessivamente raggiungono l'ordine delle migliaia. Potenzialmente, per ogni metrica, può essere impostata una soglia di warning e di critical che, una volta superata, può innescare una procedura di "alerting" segnalando all'utente del sistema la gravità dell'incidente e il valore della metrica rilevato sul target sotto monitoraggio.

Di default, le metriche hanno frequenza di campionamento variabile e possono essere modificate manualmente dall'amministratore del sistema. Generalmente, le metriche che subiscono maggiori variazioni nel tempo (es. utilizzo della CPU) vengono campionate con una frequenza maggiore rispetto alle metriche che crescono/decrescono lentamente nel tempo (es. spazio libero nel disco).

Un'altra caratteristica di come l'OEM gestisce le metriche all'interno del repository è che a seconda delle informazioni che forniscono, esse vengono raggruppate in macro categorie. In Fig. 2.12, è mostrato un esempio di gruppo di metriche che forniscono informazioni sul carico di un "host".

Le Viste

Oltre all'interfaccia web fornita dalla console, l'OEM mette a disposizione degli sviluppatori, un insieme di viste per la fruizione dei dati raccolti ed archiviati all'interno del repository. Le viste sono ben documentate dal manuale ufficiale di Oracle [2] ed anche in questo caso la loro cardinalità raggiunge l'ordine delle migliaia.

Poichè le viste fornite sono semplici query read-only su un database e sono indipendenti dalle tabelle relazionali sottostanti che potrebbero subire modifiche con

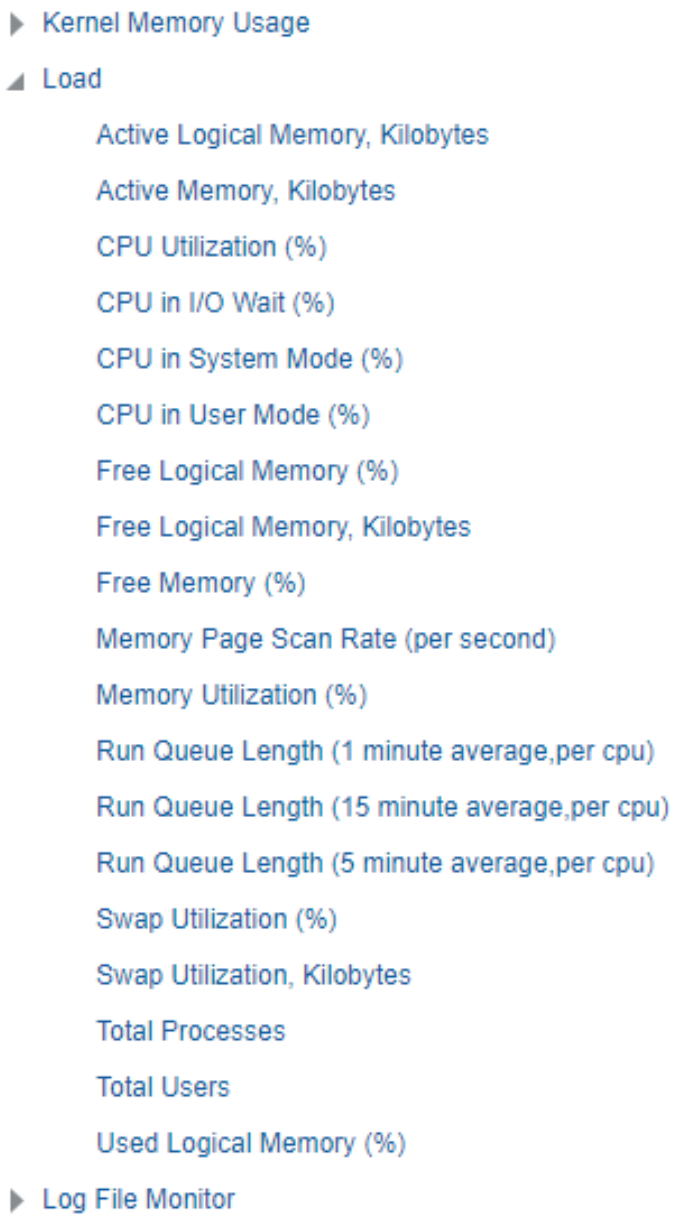

Figura 2.12. Esempio Load Metrics Group.

eventuali upgrade della piattaforma, gli sviluppatori possono creare applicazioni utilizzando queste viste, la cui compatibilità con il repository è garantita nel tempo da Oracle.

Di particolare interesse rispetto al lavoro svolto, sono le "*Monitoring Views*", presentate in Fig. 2.13, che forniscono informazioni sulle risorse monitorate dei target

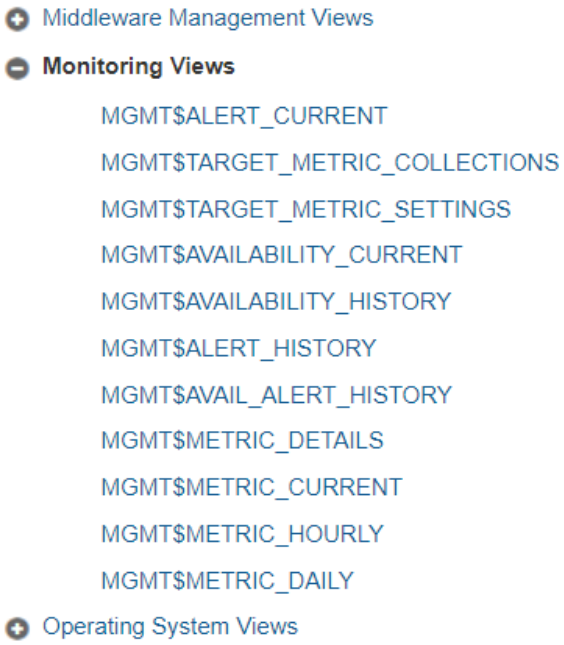

Figura 2.13. OEM Monitoring Views (fonte: oracle.com).

censiti. Alcune di esse, vengono descritte brevemente di seguito:

- MGMT$ALERT_CURRENT: contiene le informazioni correnti sugli ultimi alert archiviati nel repository;
- MGMT$AVAILABILITY_CURRENT: visualizza le informazioni sullo stato di disponibilità dei target censiti nel OEM;
- MGMT$METRIC_CURRENT: visualizza informazioni sui valori di metrica più recenti, di ciascun target, caricati nel repository;
- MGMT$METRIC_HOURLY: visualizza le informazioni statistiche sulle metriche di ciascun target, che sono state aggregate dai singoli campioni di metrica in periodi di tempo orari. Ad esempio, se una metrica viene raccolta ogni 15 minuti, il rollup di 1 ora aggrega i 4 campioni in un singolo valore orario calcolando la media dei 4 singoli campioni insieme;
- MGMT$METRIC_DAILY: contiene le statistiche delle metriche, di ciascun target, che sono state aggregate dai campioni raccolti nel precedente periodo di 24 ore.

2.2.3 Limitazioni

L'Oracle Enterprise Manager è sicuramente un'ottima soluzione enterprise di monitoraggio che permette di gestire un intero sistema in tempo reale da una postazione centralizzata, di collezionare metriche sui target monitorati per riferimenti e analisi future e di configurare un sistema di alert (es. tramite email) in modo da informare l'utente ogni qualvolta una metrica di un determinato host supera una soglia di warning/critical pre-impostata. In base all'esperienza maturata nel campo, le principali limitazioni notate durante l'utilizzo di questo prodotto sono:

- Sistema di alerting basato su soglie fisse;
- Necessita di tuning manuale e meticoloso delle soglie di warning/critical onde evitare spam di alert (operazione "time-consuming");
- Non sempre il superamento della soglia di una metrica è sintomo di anomalia. Caso classico è l'utilizzo della CPU di un host che per pochi instanti potrebbe raggiungere il 100% di utilizzo della risorsa innescando così il processo di segnalazione dell'anomalia verso l'utente.

I limiti sopra riportati, sono le principali motivazioni che ha spinto l'azienda nella ricerca di nuove soluzione da implementare puntando, attraverso tecniche di machine learning, ad aggiungere ulteriore intelligenza ad un già ottimo prodotto di monitoraggio come l'Oracle Enterprise Manager.

Capitolo 3

Approccio al Problema

Questo capitolo ha lo scopo di illustrare e definire brevemente l'obiettivo della ricerca e l'approccio che è stato seguito per il raggiungimento di una soluzione aziendale ottimale.

3.1 Obiettivo

L'obiettivo che è stato prefissato di raggiungere con questo lavoro di ricerca, è lo studio di fattibilità di una possibile soluzione che sia in grado di identificare automaticamente pattern nascosti e anomali rispetto a comportamenti normali. L'idea alla base è quella di sfruttare un prodotto enterprise di monitoraggio, già utilizzato in azienda, e di implementare su di esso dell'intelligenza di anomaly detection utilizzando tecniche di machine learning.

3.2 Fasi della ricerca

Il lavoro di tesi è stato condotto secondo il workflow mostrato in Fig. 3.1, il quale riassume le fasi principali della ricerca, iniziando dallo studio del caso di business sino all'analisi dei risultati ed applicazione al contesto aziendale.

Caso Studio

La prima fase è stata incentrata sullo studio del caso e degli strumenti di monitoraggio dei sistemi informatici messi a disposizione dall'azienda. In questa fase è stata fatta una ricerca dello stato dell'arte relativo alle tecniche di anomaly detection

Figura 3.1. Workflow della ricerca.

e sono stati studiati sia dal punto di vista dell'utilizzatore che dello sviluppatore "l'Oracle Enterprise Manager", un prodotto enterprise di monitoraggio e "l'Oracle Database" uno dei più famosi database management system (DBMS). L'Oracle database è utilizzato dall'OEM come repository dei dati raccolti. In Fig. 3.2, sono illustrati i principali componenti che devono essere presi in considerazione in un problema di anomaly detection.

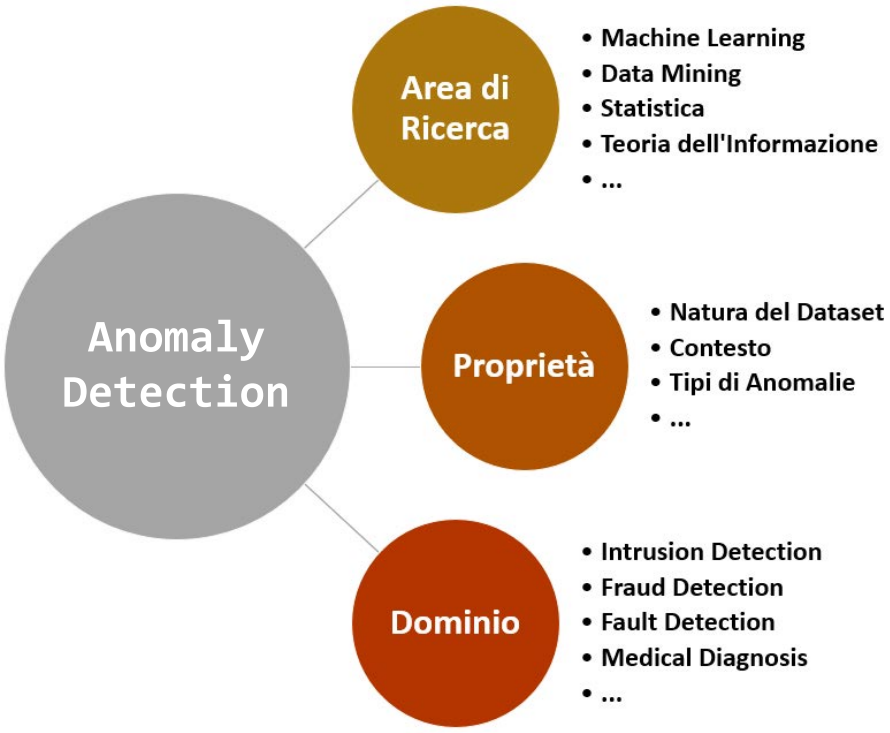

Figura 3.2. Principali componenti di un modello di anomaly detection.

Definizione del Dataset

Terminata la fase di studio del caso, lo step successivo è stato quello di analizzare i dati a disposizione. L'obiettivo di questa fase è di individuare i dati utili ai fini della ricerca e prepararli per l'utilizzo di modelli di machine learning. Questi due processi sono spesso noti come "Data Selection" e "Data Preprocessing".

Implementazione delle Soluzioni

Avendo a questo punto un dataset a disposizione da analizzare, in questa fase è stata implementata un'applicazione in Python per l'applicazione dei diversi modelli di anomaly detection presi in considerazione.

Analisi dei Risultati

Questa è la fase cruciale della ricerca, in cui sono stati messi a confronto i risultati ottenuti dal passo precedente. Per la valutazione delle performance dei modelli di anomaly detection sono stati adottati due criteri differenti: calcolo del punteggio di Silhouette e interpretazione grafica dei risultati. Al termine di questa fase, è stato selezionato il modello più performante rispetto al dataset a nostra disposizione.

Applicazione al Contesto Aziendale

Quest'ultima fase, consiste nell'attivare l'applicazione implementata su un numero ristretto di clienti e dunque mettere a disposizione dei colleghi, responsabili del monitoraggio dei sistemi, un ulteriore strumento di rilevamento delle anomalie.

3.3 Assunzioni

Le principali assunzioni su cui si basano i più diffusi modelli di anomaly detection e conseguentemente questo lavoro di ricerca sono le seguenti:

1. Le anomalie sono rare e quindi, la loro distribuzione all'interno del dataset è nettamente inferiore rispetto alla distribuzione dei dati normali;
2. Le anomalie hanno caratteristiche molto diverse dai dati normali;
3. Per le tecniche basate sul clustering, i dati normali appartengono ai cluster più grandi e densi, mentre gli outlier a nessun cluster o a cluster piccoli e poco densi

3.4 Tecnologie utilizzate

Per lo svolgimento del lavoro di tesi, sono state utilizzate le seguenti tecnologie:

- **"Oracle Enterprise Manager 13c"** come strumento di monitoraggio e **"Oracle Database Enterprise Edition Release 12.1.0.2.0"** come storage dei dati raccolti.
- **Python**, come linguaggio di programmazione scelto per l'implementazione di possibili soluzione di anomaly detection. La scelta di questo linguaggio nasce essenzialmente da due principali motivi:
  1. È un linguaggio multi-paradigma, che ha tra i principali obiettivi dinamicità, semplicità e flessibilità. Inoltre, nel contesto aziendale, Python è già stato adottato in altri progetti.
  2. Mette a disposizione una vasta scelta di librerie standard e/o open-source come: **Scikit-learn**, una libreria open-source che implementa i principali algoritmi di machine learning; **Pandas** un'altra libreria open-source focalizzata sulla manipolazione e analisi dei dati; **Matplotlib** libreria open-source per la creazione di grafici in 2 dimensioni.
- Gli approcci ai problemi di outlier detection, oggetto di questo lavoro, rientrano nella categoria delle tecniche non supervisionate. Per questo motivo, in questa ricerca, sono stati messi a confronto quattro diversi modelli di anomaly detection: **k-Means**, **DBSCAN**, **Isolation Forest** e **Local Outlier Factor**.

Capitolo 4

Progettazione e Implementazione

In questo capitolo vengono illustrate le scelte implementative adottate durante il lavoro di ricerca, partendo da una descrizione generale del flusso di dati che interessa il sistema di anomaly detection progettato fino alle scelte dei modelli utilizzati per il raggiungimento dell'obiettivo stabilito.

4.1 Panoramica del Sistema

Il sistema implementato può essere riassunto ad alto livello dalla Fig.4.1. Le tecniche di anomaly detection si collocano al top dello stack applicativo, in quanto sfruttano il lavoro di monitoraggio svolto dall'Oracle Enterprise Manager, che a sua volta utilizza come repository dei dati raccolti l'Oracle Database.

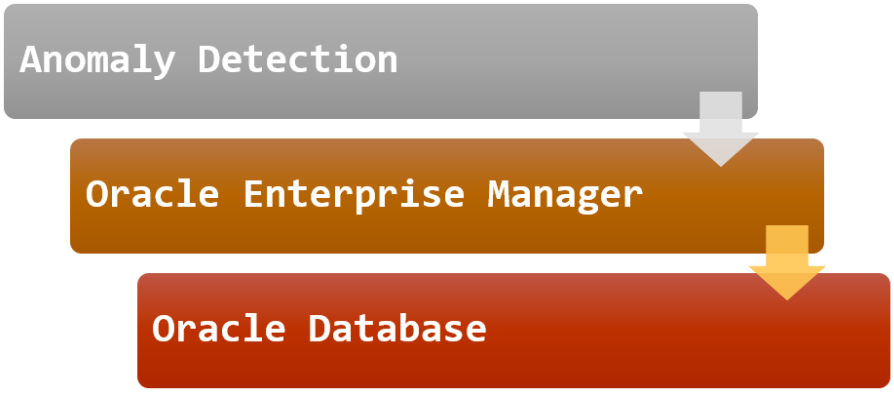

Figura 4.1. Stack applicativo ad alto livello.

In Fig. 4.2, viene invece mostrato il flusso di dati che si sviluppa all'interno del sistema considerato. Il flusso inizia dalla comunicazione delle informazioni sulle risorse monitorate tra gli host e l'OEM, il quale le archivia all'interno del suo repository. Successivamente, i dati storici delle metriche raccolte vengono utilizzate come dataset di addestramento del modulo di anomaly detection implementato. Terminata la fase di addestramento del modello, il modulo di anomaly detection interroga frequentemente gli ultimi dati raccolti dall'OEM e nel caso dovesse riscontrare un comportamento anomalo, innesca una procedura di alerting portando all'attenzione dell'utente l'alert identificato. Dal momento che il carico e l'utilizzo delle risorse di un determinato target può variare nel tempo, periodicamente si rende necessario un aggiornamento dei dati di addestramento del modello. Nell'ottica di ridurre la quantità di falsi allarmi verso l'amministratore che monitora i sistemi informatici, è possibile implementare una semplice logica applicativa di attivazione della procedura di alerting solamente nei casi in cui il modello identifica per n volte consecutive un comportamento anomalo del target.

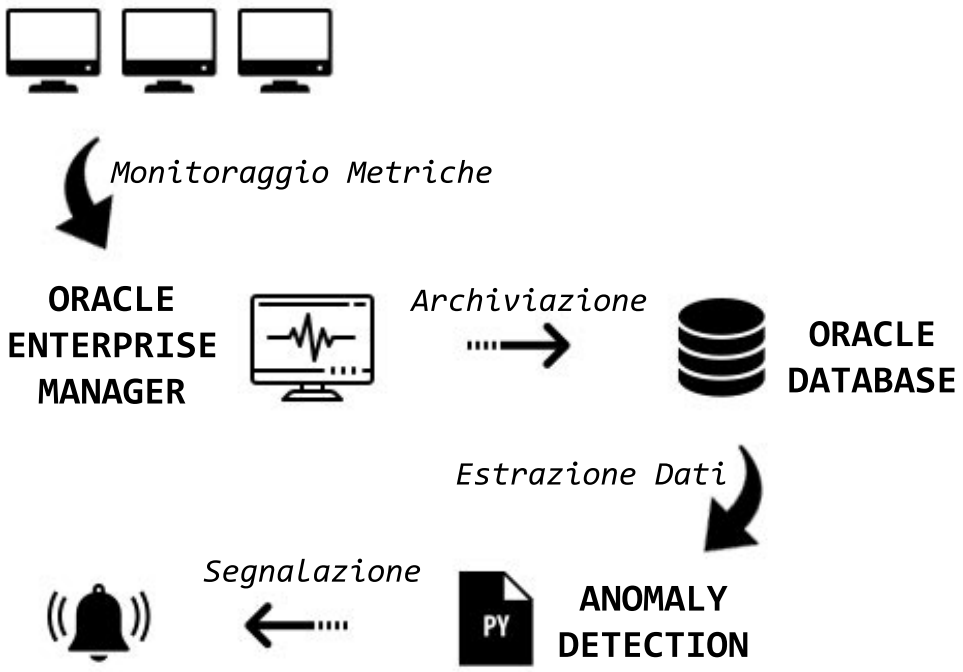

Figura 4.2. Flusso dati del sistema di anomaly detection.

4.2 Dataset

Avere un dataset affidabile e adatto all'applicazione di modelli di machine learning è uno degli step essenziali in questo tipo di ricerca. Dal momento che non era a disposizione un dataset pronto per l'applicazione delle tecniche di anomaly detection, si è resa necessaria una prima fase di analisi e individuazione dei dati utili allo scopo della ricerca.

Come primo step, sono state analizzate le viste messe a disposizione agli sviluppatori dall'Oracle Enterprise Manager, le quali sono già state descritte brevemente nel capitolo 2. Sulla base dell'obiettivo di questa ricerca, sono state analizzate e fatte delle considerazioni sulle "Monitoring Views", in particolare sulle viste "MGMT$METRIC_CURRENT", "MGMT$METRIC_HOURLY" e "MGMT$METRIC_DAILY".

- La **MGMT$METRIC_DAILY** è stata fin da subito esclusa in quanto, l'aggregazione giornaliera delle metriche, non ci permette di avere un livello di dettaglio dei dati raccolti che possa rappresentare accuratamente il comportamento nel tempo di uno specifico target; per cui, nel nostro contesto, la vista non risulta essere adatta per l'applicazione dei modelli di anomaly detection sulle performance di un target in tempo reale.
- La **MGMT$METRIC_CURRENT** è stata anch'essa esclusa fin dal principio in quanto, essa permette di interrogare i dati raccolti solamente nelle ultime 24 ore e quindi, anch'essa, non risulta essere adatta a descrivere un comportamento nel tempo di un target.
- La **MGMT$METRIC_HOURLY** invece, ai fini della nostra ricerca, è stata individuata come una possibile sorgente del dataset di addestramento per i nostri modelli di anomaly detection. Questa vista visualizza i dati aggregati ora per ora con una politica di conservazione di 31 giorni.

Analizzando nel dettaglio la MGMT$METRIC_HOURLY, si è risalito ad un'ulteriore vista che l'OEM sfrutta per la generazione degli aggregati ora per ora, ovvero la **GC_METRIC_VALUES**. Quest'ultima vista permette di interrogare i dati raccolti, di ciascun target, senza aggregazioni orarie e con una politica di conservazione di 7 giorni.

Considerando che, per ciascun target, i campioni di dati che possono essere estratti dalla MGMT$METRIC_HOURLY sono dell'ordine di 700 osservazioni mentre per la GC_METRIC_VALUES sono dell'ordine di 2000, si è arrivati alla conclusione che i comportamenti di un determinato target possono essere meglio rappresentati

Nome	Tipo
ENTITY_TYPE	VARCHAR2(64)
ENTITY_NAME	VARCHAR2(256)
ENTITY_GUID	RAW(16 BYTE)
METRIC_GROUP_NAME	VARCHAR2(64)
METRIC_COLUMN_NAME	VARCHAR2(64)
...	...
METRIC_GROUP_ID	NUMBER(38)
METRIC_GROUP_GUID	RAW(16 BYTE)
METRIC_GROUP_LABEL	VARCHAR2(64)
METRIC_GROUP_LABEL_NLSID	VARCHAR2(64)
METRIC_COLUMN_ID	NUMBER(38)
METRIC_COLUMN_GUID	RAW(16 BYTE)
METRIC_COLUMN_LABEL	VARCHAR2(64)
METRIC_COLUMN_LABEL_NLSID	VARCHAR2(64)
DESCRIPTION	VARCHAR2(1024)
SHORT_NAME	VARCHAR2(40)
UNIT	VARCHAR2(64)
IS_FOR_SUMMARY	NUMBER
IS_STATEFUL	NUMBER
IS_TRANSPOSED	NUMBER(1)
NON_THRESHOLDED_ALERTS	NUMBER
METRIC_TYPE	NUMBER(1)
USAGE_TYPE	NUMBER(1)
METRIC_KEY_ID	NUMBER(38)
NUM_KEYS	NUMBER(1)
METRIC_KEY_VALUE	VARCHAR2(256)
KEY_PART_1	VARCHAR2(256)
KEY_PART_2	VARCHAR2(256)
KEY_PART_3	VARCHAR2(256)
KEY_PART_4	VARCHAR2(256)
KEY_PART_5	VARCHAR2(256)
KEY_PART_6	VARCHAR2(256)
KEY_PART_7	VARCHAR2(256)
COLLECTION_TIME	DATE
COLLECTION_TIME_UTC	DATE
VALUE	NUMBER

Tabella 4.1. Schema ridotto della vista GC_METRIC_VALUES.

dalla vista GC_METRIC_VALUES. Una versione ridotta dello schema della vista GC_METRIC_VALUES è mostrata in Tab. 4.1.

Interrogando la vista GC_METRIC_VALUES dell'Oracle Enterprise Manager attualmente utilizzato in azienda, sono state elaborate e illustrate in Tab. 4.2 alcune statistiche generali in riferimento ai sistemi di diversi clienti tenuti sotto monitoraggio all'interno del data center aziendale.

#Target Totali	**#Host**	**#Metriche Totali**	**#Metriche per Host**
2006	68	1402	84

Tabella 4.2. Alcune statistiche dell'OEM aziendale, aggiornato al 23/06/18.

Il sistema di monitoraggio gestisce attualmente 2006 target di differenti tipi tra quelli elencati in Tab. 4.3 (in grassetto i tipi di target più comuni) e colleziona informazioni per 1402 differenti metriche. Tra i 2006 target, 68 di questi sono host fisici e virtuali e per ciascun host il sistema raccoglie, mediamente, informazioni per 84 metriche.

4.2.1 Raccolta Dati

Dato il gran numero di target sotto monitoraggio, si è deciso di focalizzare l'attenzione del lavoro di tesi su un numero ristretto di target per poi estenderlo in futuro. Per questo motivo, si è scelto di iniziare questa ricerca partendo dal tipo di target più comune "**l'host**".

Con l'obiettivo di generalizzare il più possibile il nostro approccio, una prima analisi è stata quella di individuare quali tipi di metriche hanno in comune i 68 host monitorati. Questa prima analisi ha lo scopo sia di ridurre la cardinalità delle metriche da processare, sia di eliminare i campioni di dati non valorizzati per alcuni target. La differenza nel numero delle metriche raccolte per ciascun host nasce essenzialmente per due motivi:

1. Il diverso sistema operativo ospitato dagli host monitorati, cioè alcune metriche raccolte per sistemi che ospitano Windows non esistono per gli host che ospitano Linux e viceversa;

2. La possibilità di customizzare per clienti e per tipo di target quali informazioni raccogliere e conseguentemente quali metriche tenere sotto monitoraggio.

Tipi di Target	
asm_diskgroup_component	oracle_oms_console
casm_diskgroup_component	oracle_oms_pbs
cluster	oracle_pdb
generic_service	oracle_repapp
has	oracle_vm_manager
host	oracle_repserv
jee_webservice	oracle_si_filesystem_host
jrf_webservice	oracle_si_lvm_host
j2ee_application	oracle_si_network_data_link_host
j2ee_application_cluster	oracle_si_network_interface_host
metadata_repository	oracle_si_server_map
oak	oracle_si_volume_host
odba_sys	oracle_vm_cloud
oracle_apache	oracle_vm_guest
oracle_beacon	oracle_vm_server
oracle_coherence	oracle_vm_server_pool
oracle_coherence_cache	osm_cluster
oracle_database	osm_instance
oracle_emd	osm_proxy
oracle_emrep	rac_database
oracle_em_service	weblogic_cluster
oracle_exa_ilom	weblogic_domain
oracle_forms	**weblogic_j2eeserver**
oracle_listener	weblogic_nodemanager
oracle_oms	

Tabella 4.3. Tipi di target presenti nel repository, aggiornato al 23/06/18.

Questa analisi ci ha consentito di ridurre dalle 84 metriche di media raccolte per host, a 25 metriche sicuramente collezionate per tutti gli host presenti all'interno del repository. Le metriche ottenute sono elencate in Tab. 4.4.

Come è possibile notare da quest'ultima tabella, alcune metriche forniscono informazioni complementari tra loro. Un esempio sono le metriche "Free Memory (%)" e "Memory Utilization (%)" che rispettivamente indicano la quantità di memoria RAM libera e la quantità di memoria RAM occupata di un host. Considerare entrambe le metriche non risulterebbe utile ai fini della ricerca ma anzi, dovendo gestire un ulteriore attributo, aumenterebbe la complessità computazionale dell'applicazione. Sulla base di questa osservazione e da un confronto con i colleghi esperti

Metrica	#Host
Active Logical Memory, Kilobytes	68
All Network Interfaces Combined Utilization (%)	68
All Network Interfaces Read Rate (MB/sec)	68
All Network Interfaces Read Utilization (%)	68
All Network Interfaces Total I/O Rate (MB/sec)	68
All Network Interfaces Write Rate (MB/sec)	68
All Network Interfaces Write Utilization (%)	68
CPU in System Mode (%)	68
CPU in User Mode (%)	68
CPU Utilization (%)	68
Free Memory (%)	68
Max Average Disk I/O Service Time (ms) amongst all disks	68
Max Disk I/O (per sec) made by a single disk	68
Memory Utilization (%)	68
Pages Paged-in (per second)	68
Pages Paged-out (per second)	68
Swap Utilization (%)	68
Total Disk Available (%) (across all local filesystems)	68
Total Disk I/O made across all disks (per second)	68
Total Disk Size (across all local filesystems in MB)	68
Total Disk Space Available (across all local filesystems in MB)	68
Total Disk Space Utilized (across all local filesystems in MB)	68
Total Disk Utilized (%) (across all local filesystems)	68
Total Processes	68
Total Users	68

Tabella 4.4. Metriche analizzate per il target “host”.

nell’utilizzo quotidiano dell’Oracle Enterprise Manager, sono state estratte le metriche, per il tipo di target “host”, che risultano essere maggiormente critiche durante il lavoro di monitoraggio. In Tab. 4.5, sono elencate, definitivamente, le metriche con le rispettive frequenze di campionamento che sono state identificate come utili ai fini del lavoro di tesi e che sono state utilizzate per la generazione del dataset.

Per l’estrazione delle informazioni di interesse, in maniera agevole dall’Oracle Enterprise Manager, è stata definita all’interno del suo repository Oracle una nuova vista. La definizione della vista che è stata creata con l’utente amministratore dell’Oracle Enterprise Manager “SYSMAN” e chiamata “AD_HOST”, è consultabile in Fig. 4.4.

Metrica	Unità di Misura	ID	Frequenza di Campionamento
All Network Interfaces Read Rate	MB/sec	NETRD	15 min
All Network Interfaces Write Rate	MB/sec	NETWR	15 min
CPU in System Mode	%	CPUSYS	5 min
CPU in User Mode	%	CPUUSR	5 min
CPU Utilization	%	CPU	5 min
Memory Utilization	%	RAM	5 min
Swap Utilization	%	SWAP	5 min
Total Disk I/O	MB/sec	DISKIO	15 min
Total Disk Utilized	%	DISK	15 min

Tabella 4.5. Metriche prese in considerazione.

HOST	TIMESTAMP	METRICA	VALORE
host_1	2018-06-23 00:00:30	CPUSYS	1,354
host_1	2018-06-23 00:00:30	CPUUSR	5,462
host_1	2018-06-23 00:00:30	RAM	73,839
host_1	2018-06-23 00:00:30	SWAP	0

PIVOT

HOST	TIMESTAMP	CPUSYS	CPUUSR	RAM	SWAP
host_1	2018-06-23 00:00:30	1,354	5,462	73,839	0

Figura 4.3. Esempio applicazione operatore PIVOT.

Come è possibile notare, la vista fa uso dell'operatore PIVOT per trasporre le metriche da righe a colonne e successivamente le raggruppa, ad intervalli regolari di 5 minuti, in un unico record. L'aggregazione di 5 minuti è stata definita come il minimo della frequenza di campionamento delle metriche presenti in Tab. 4.5 in maniera tale da non perdere alcun dato collezionato dall'OEM.

Un esempio di utilizzo semplificato dell'operatore PIVOT è illustrato in Fig. 4.3.

```
CREATE OR REPLACE VIEW "SYSMAN"."AD_HOST" ("ENTITY_NAME",
   "COLLECTION_DATE", "HOURS", "MINUTES", "DISKIO", "CPU",
   "CPUSYS", "CPUUSR", "RAM", "SWAP", "NETRD", "NETWR", "DISK") AS
SELECT ENTITY_NAME, TO_CHAR(COLLECTION_TIME,'YYYY-MM-DD') AS
   COLLECTION_DATE, TO_CHAR(COLLECTION_TIME,'HH24') AS HOURS,
   FLOOR(TO_NUMBER(TO_CHAR(COLLECTION_TIME,'MI'))/5) * 5 AS
   MINUTES, MAX(met1) AS DISKIO, MAX(met2) AS CPU, MAX(met3) AS
   CPUSYS, MAX(met4) AS CPUUSR, MAX(met5) AS RAM, MAX(met6) AS
   SWAP, MAX(met7) AS NETRD, MAX(met8) AS NETWR, MAX(met9) AS DISK
FROM (
       SELECT ENTITY_TYPE, ENTITY_NAME,METRIC_COLUMN_LABEL_NLSID,
          COLLECTION_TIME, VALUE
       FROM GC_METRIC_VALUES
)
PIVOT (
       MAX(VALUE) FOR METRIC_COLUMN_LABEL_NLSID IN (
           'host_dasummary_totiosmade_persec' AS met1,
           'host_load_cpuIdle' AS met2,
           'host_load_cpuKernel' AS met3,
           'host_load_cpuUser' AS met4,
           'host_load_memUsedPct' AS met5,
           'host_load_swapUtil' AS met6,
           'host_networksummary_readrate' AS met7,
           'host_networksummary_writerate' AS met8,
           'host_total_disk_usage_totpercntused' AS met9
       )
)
WHERE ENTITY_TYPE='host' AND (met1 IS NOT NULL OR met2 IS NOT
   NULL OR met3 IS NOT NULL OR met4 IS NOT NULL OR met5 IS NOT
   NULL OR met6 IS NOT NULL OR met7 IS NOT NULL OR met8 IS NOT
   NULL OR met9 IS NOT NULL)
GROUP BY ENTITY_NAME, TO_CHAR(COLLECTION_TIME,'YYYY-MM-DD'),
   TO_CHAR(COLLECTION_TIME,'HH24'),
   FLOOR(TO_NUMBER(TO_CHAR(COLLECTION_TIME,'MI'))/5) * 5
WITH READ ONLY;
```

Figura 4.4. Definizione vista per generazione dataset.

4.2.2 Pre-Processing

Generalmente, prima di applicare un qualsiasi modello di Data Mining, è necessaria una fase di pre-processing dei dati raccolti in modo tale da trasformare il dataset di dati grezzi in un insieme di dati idonei all'applicazione di tecniche come l'anomaly detection.

I principali obiettivi della fase di pre-processing sono:

- Migliorare la qualità del dataset;
- Ridurre la cardinalità e la dimensionalità del dataset;
- Migliorare le performance e l'accuratezza dei modelli di Data Mining.

In questa sezione vengono quindi illustrati i dettagli delle tecniche di pre-processing che sono state utilizzate sui dati estratti dal repository dell'Oracle Enterprise Manager.

Data Cleaning

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. Come è possibile notare nel esempio 4.6, a causa di una diversa frequenza di campionamento delle metriche, definite in Tab. 4.5, durante la fase di estrazione dei dati di interesse tramite la vista implementata 4.4, risultano esserci alcuni campi non valorizzati, i quali manderebbero in crisi le diverse tecniche di anomaly detection prese in considerazione.

CPU	CPU SYS	CPU USR	RAM	SWAP	NET RD	NET WR	DISK	DISK IO
6,861	1,354	5,462	73,878	0	(null)	(null)	(null)	(null)
10,271	1,741	8,449	73,839	0	15,471	0,143	95,733	(null)
5,513	1,18	4,316	73,839	0	(null)	(null)	(null)	5129,386
4,088	0,888	3,197	73,612	0	(null)	(null)	(null)	(null)
4,142	0,921	3,208	73,543	0	1,347	0,027	95,328	(null)

Tabella 4.6. Esempio di 5 record di uno specifico target host.

Per risolvere questo problema si è scelto di ipotizzare che, per le metriche con frequenza di campionamento più alta (15 minuti nel nostro caso), il valore collezionato, in un dato momento, resta costante fino al nuovo valore campionato. In

questo senso, è stata quindi fatta una scelta di riempire i campi non valorizzati, propagando il valore campionato in avanti ed indietro fino al prossimo valore valido. Il risultato, visibile nell'esempio 4.7, è stato ottenuto in Python utilizzando il metodo **fillna** della classe **DataFrame** della libreria **Pandas**.

df.fillna(method='ffill').fillna(method='bfill')

CPU	CPU SYS	CPU USR	RAM	SWAP	NET RD	NET WR	DISK	DISK IO
6,861	1,354	5,462	73,878	0	**15,471**	**0,143**	**95,733**	**5129,386**
10,271	1,741	8,449	73,839	0	15,471	0,143	95,733	**5129,386**
5,513	1,18	4,316	73,839	0	**15,471**	**0,143**	**95,733**	5129,386
4,088	0,888	3,197	73,612	0	**15,471**	**0,143**	**95,733**	**5129,386**
4,142	0,921	3,208	73,543	0	1,347	0,027	95,328	**5129,386**

Tabella 4.7. Riempimento dei campi non valorizzati.

Standardizzazione dei valori

Altro aspetto del pre-processing 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'analisi. La standardizzazione di un dataset è solitamente un requisito comune per l'applicazione di molte tecniche di machine learning. In genere questo tipo di pre-processing viene fatto rimuovendo la media e scalando i dati alla varianza dell'unità. Tuttavia, i valori anomali possono spesso influenzare in modo negativo la media/varianza del campione di dato. In letteratura si è notato che in questi casi, rimuovere la mediana e scalare i dati in base all'intervallo quantile danno spesso risultati migliori. Per la trasformazione dei dati in Tab. 4.8, si è sfruttata la funzione **RobustScaler** fornita dalla libreria **SKlearn.preprocessing**.

Riduzione della dimensionalità

In statistica e in ambito del machine learning, la riduzione della dimensione è il processo di riduzione del numero di variabili casuali considerate, ottenendo un insieme di variabili principali. Una delle più conosciute e principale tecnica lineare per la riduzione della dimensionalità, è il "Principal Component Analysis (PCA)", il quale esegue una mappatura lineare dei dati originali in uno spazio di dimensioni

CPU	CPU SYS	CPU USR	RAM	SWAP	NET RD	NET WR	DISK	DISK IO
0.709	0.385	0.834	0.203	0	0.677	0.36	0.988	0.977
2.197	1.316	2.471	0.202	0	0.677	0.36	0.988	0.977
0.12	-0.034	0.206	0.202	0	0.677	0.36	0.988	0.977
-0.501	-0.736	-0.407	0.193	0	0.677	0.36	0.988	0.977
-0.478	-0.657	-0.401	0.191	0	-0.378	-0.228	0.92	0.977

Tabella 4.8. Standardizzazione dei dati tramite RobustScaler.

inferiori in modo tale da massimizzare la varianza dei dati analizzati. Nella libreria scikit-learn, PCA è implementato come un oggetto trasformatore in grado di proiettare un dataset con k dimensioni in uno spazio di n dimensioni definito dall'utente. In questo lavoro, è stato scelto di ridurre il dataset a 2 dimensioni in modo tale, da poter valutare e mettere a confronto graficamente i diversi algoritmi di anomaly detection.

PCA(n_components=2)

I principali obiettivi delle tecniche di riduzione della dimensione di un dataset sono i seguenti:

- Ridurre il tempo di esecuzione e lo spazio richiesto in memoria.
- Migliorare le performance dei modelli di machine learning.
- Visualizzare graficamente i dati quando questi vengono ridotti a dimensioni molto ridotte, come 2D o 3D.

Esempio di applicazione del PCA è mostrato in Tab. 4.9.

ATTRIBUTO_PCA_1	ATTRIBUTO_PCA_2
-6.344	0.927
-4.963	1.891
-6.938	0.559
-7.813	0.214
-7.727	-0.067

Tabella 4.9. Applicazione del PCA sui dati di esempio standardizzati.

4.3 Algoritmi

Considerando il contesto aziendale dove una parte di business riguarda l'offerta di un servizio di monitoraggio dei sistemi informatici di medio/grandi aziende con un elevato numero di host e configurazioni di sistemi diversi l'uno dall'altro, avere un dataset uniforme e suddiviso in classi "anomala" e "normale" diventa estremamente complesso. In questo senso, gli algoritmi di anomaly detection non supervisionati risultano maggiormente adatti al nostro contesto. Gli algoritmi presi in considerazione in questo lavoro di tesi, sono i seguenti: **K-Means**, **DBSCAN**, **Isolation Forest** e **Local Outlier Factor**. La scelta di questi algoritmi è dettata essenzialmente da due motivi principali:

1. L'ampio utilizzo riscontrato in letteratura nell'ambito dell'anomaly detection;
2. La possibilità di sfruttare le implementazioni fornite dalla libreria scikit-learn di Python.

Come spesso accade in problemi legati al machine learning, si rende necessaria una fase di tuning dei parametri di inizializzazione degli algoritmi. I principali parametri che necessitano di taratura, di ciascun algoritmo, sono i seguenti:

- **sklearn.cluster.kMeans**: *n_clusters* ovvero il numero di cluster che l'algoritmo ha il compito di formare;
- **sklearn.cluster.DBSCAN**: *eps* e *min_samples* indicano rispettivamente la massima distanza tra due campioni di dati per essere considerati dello stesso cluster e il numero minimo di campioni per considerare un punto come "core point";
- **sklearn.ensemble.IsolationForest**: *contamination* indica la proporzione dei campioni anomali all'interno del dataset;
- **sklearn.neighbors.LocalOutlierFactor**: *n_neighbors* e *contamination* rispettivamente indicano il numero di vicini utilizzato dal k-nearest neighbors e la proporzione dei campioni anomali all'interno del dataset.

Per ciascun host analizzato, il tuning dei parametri è stato eseguito in maniera empirica cercando di massimizzare il coefficiente di silhouette, che come descritto in precedenza, fornisce una valutazione sulla bontà dei cluster di dati generati. Gli intervalli di valori utilizzati nella fase di tuning sono presentati in Tab. 4.10. La fase di tuning dovrà essere eseguita ogni qualvolta che i campioni di dati all'interno del dataset verranno modificati.

L'implementazione in scikit-learn degli algoritmi considerati fornisce, per il loro utilizzo, due metodi principali:

Algoritmo	Parametro	Intervallo di Valori
k-Means	n_clusters	{2, 3, 4, ..., 10}
DBSCAN	eps	{1, 1.1, 1.2, ..., 40}
	min_samples	{3, 4, 5, ..., 12}
Isolation Forest	contamination	{0.001, 0.002, ... 0.3}
LOF	n_neighbors	{3, 4, 5, ..., 40}
	contamination	{0.001, 0.002, ... 0.3}

Tabella 4.10. Intervalli di valori utilizzati durante la fase di tuning dei principali parametri degli algoritmi di anomaly detection.

- **fit**: riceve in input il dataset non etichettato e allena il modello per l'identificazione delle anomalie;

- **predict**: riceve in input una nuova osservazione e predice il grado di anormalità del campione di dato sotto osservazione.

4.4 Esperimenti

Terminata la fase di raccolta dati, il dataset così creato degli host monitorati dall'OEM è stato elaborato ed analizzato attraverso gli algoritmi: k-Means, DBSCAN, Isolation Forest e Local Outlier Factor.

Il dataset è costituito da campioni di dati relativi a 68 host con configurazioni ed applicazioni differenti l'uno dall'altro. Con l'obiettivo di valutare quale dei quattro algoritmi fosse il più efficiente rispetto al dataset in esame, sono stati analizzati i risultati ottenuti dall'applicazione dei modelli di anomaly detection sui dati raccolti dai 68 host sotto monitoraggio.

I 68 casi studio sono stati condotti, sotto forma di pseudo-codice, seguendo la logica applicativa descritta di seguito:

```
1 foreach ENTITY_NAME presente nel dataset do
2 |   estrarre l'insieme dei dati relativi a ENTITY_NAME, indicati come D_e;
3 |   eseguire le operazioni di pre-processing sul D_e;
4 |   eseguire il tuning dei parametri di configurazione di ciascun algoritmo;
5 |   addestrare i modelli di anomaly detection con D_e;
6 |   per gli algoritmi di clustering, considerare i cluster con cardinalità
  |     inferiore al 10% dell'intero dataset come cluster anomali;
7 |   valutare gli algoritmi sulla base del punteggio di Silhouette e della
  |     rappresentazione grafica;
8 end
```

La presentazione dei risultati ottenuti dagli esperimenti con relativa discussione saranno oggetto del capitolo successivo. Per ovvie ragioni, non potendo descrivere nel dettaglio tutti e 68 i casi analizzati, saranno riassunti in tabelle i migliori parametri ottenuti dalla fase di tuning e i punteggi di Silhouette ottenuti dai quattro algoritmi di anomaly detection; mentre, saranno discussi e approfonditi nel dettaglio i tre casi più rilevanti per la scelta dell'algoritmo che ha ottenuto i risultati migliori.

Il primo esperimento si basa sull'applicazione dei modelli di anomaly detection su un dataset di dati reali su cui è noto a priori la presenza di anomalie. In questo dataset, le anomalie sono note in quanto durante la raccolte dei dati di interesse, si è verificato un problema di performance e di blocco dell'host sotto monitoraggio, il quale ha richiesto l'intervento di un nostro collega per l'analisi e la risoluzione del problema.

Grazie alle conoscenze acquisite su una porzione di dati del dataset, con questo esperimento è stato possibile valutare con quale accuratezza i diversi algoritmi hanno identificato i campioni di dati anomali conosciuti all'interno del dataset.

Il secondo e il terzo esperimento si basano invece sull'applicazione dei modelli di anomaly detection su due dataset di due host differenti, di cui non si conoscono a priori informazioni sulla natura dei dati raccolti.

Capitolo 5

Analisi dei Risultati

In questo capitolo, sono presentati e analizzati i risultati degli esperimenti eseguiti durante il lavoro di tesi. La prima sezione è focalizzata sulla presentazione dei risultati ottenuti, mentre nella seconda sezione sono stati discussi prima singolarmente e dopo complessivamente i risultati ottenuti dagli algoritmi di anomaly detection.

5.1 Descrizione e Presentazione dei Risultati

Il lavoro di tesi nasce con l'obiettivo di identificare automaticamente le anomalie di performance dei sistemi informatici monitorati tramite l'Oracle Enterprise Manager.

Durante lo svolgimento della ricerca, sono state raccolte metriche su 68 diversi host tenuti sotto monitoraggio dall'OEM aziendale. Per ciascun host, le metriche prese in considerazione riguardano essenzialmente: il consumo di CPU, di memoria RAM, dei dischi fisici e l'utilizzo delle interfacce di rete. Il dataset, generato durante la fase di raccolta dei dati, comprende complessivamente 160.542 record relativi ai 68 host monitorati e quindi, in media 2.360 campioni di dati per ciascun host.

Cardinalità	**#Host**	**#Record medio per Host**
160.542	68	2.360

Tabella 5.1. Caratteristiche dataset target "host".

Come già indicato in precedenza, per ogni target host presente nel dataset, sono stati messi a confronto quattro diversi algoritmi di anomaly detection: k-Means, DBSCAN, Isolation Forest e Local Outlier Factor.

I criteri di valutazione, adottati per stabilire quale dei quattro algoritmi ha conseguito i migliori risultati sull'insieme dei dati raccolti, sono:

1. Calcolo del punteggio di Silhouette;
2. Interpretazione grafica dei risultati in uno spazio bi-dimensionale.

In Tab. 5.11 e 5.12 sono rispettivamente rappresentati, per ognuno dei 68 host presenti nel dataset, i valori dei principali parametri utilizzati da ciascun algoritmo e il punteggio di Silhouette ottenuto con quei parametri. Nella Tab. 5.12, i punteggi di Silhouette migliori sono stati evidenziati in grassetto.

Considerato l'elevato numero di host analizzati durante il lavoro di tesi svolto, di seguito sono presentati i tre casi studio ritenuti più significativi per la valutazione del miglior modello di anomaly detection.

5.1.1 Caso Studio n° 1

Il primo test, in riferimento alle Tab. 5.11 e 5.12, è stato eseguito sul target "**host_1**". Rispetto agli altri due esperimenti presentati in seguito, nel primo caso studio si ha il grosso vantaggio di conoscere a priori la presenza di campioni di dati anomali nel dataset analizzato in quanto, durante il lavoro di monitoraggio e di collezione dei dati, sono stati riscontrati problemi di performance sulla macchina.

Il dataset in esame è composto complessivamente da 2447 campioni di dati. Sulla base dell'analisi svolta dal collega che ha trattato questo problema di performance, sono state identificate manualmente 93 record anomali, il cui timestamp ricadeva nella finestra temporale in cui si è verificato il problema. Sui restanti 2354 dati del dataset invece, non si conoscono a priori informazioni sulla classe di appartenenza.

Algoritmo	**Parametri**	**Silhouette Score**
k-Means	k=2	**0.988**
Isolation Forest	contamination=0.050	0.955
DBSCAN	eps=26 min_samples=6	**0.988**
LOF	contamination=0.001 n_neighbors=5	0.866

Tabella 5.2. Tuning dei parametri e punteggi di Silhouette ottenuti durante il primo test.

Nella Tab. 5.2, sono stati riportati i valori dei principali parametri utilizzati dagli algoritmi di anomaly detection e i punteggi di Silhouette ottenuti in questo test.

Per quanto riguarda la rappresentazione grafica dei risultati ottenuti nel primo esperimento, le Fig. 5.1 e 5.2 mostrano come i diversi algoritmi di anomaly detection identificano il comportamento normale e anormale del target "host_1". I risultati sono poi stati riassunti, ad alto livello, in Tab. 5.3.

Algoritmo	**#Dati Normali**	**#Dati Anomali**
k-Means	2354	93
Isolation Forest	2324	123
DBSCAN	2354	93
LOF	2444	3

Tabella 5.3. Riassunto dei risultati ottenuti dagli algoritmi di anomaly detection sul dataset dell'host_1 costituito da un totale di 2447 campioni di dati.

Come è possibile notare in Fig. 5.2, i risultati ottenuti dall'algoritmo "LOF" non sono stati molto soddisfacenti. Per questo motivo, è stato deciso di indagare più a fondo tarando manualmente i parametri di inizializzazione dell'algoritmo. Il miglior risultato ottenuto è visibile in Fig. 5.3.

Come ulteriore termine di paragone, sono stati identificati e riportati in Tab. 5.4 i valori di tre punti rappresentativi delle aree di maggior interesse P_1, P_2 e P_3. È stata inoltre riportata per ciascuna metrica la media calcolata sull'intero dataset.

Punto	**CPU**	**RAM**	**SWAP**	**NET RD**	**NET WR**	**DISK**	**DISK IO**
P_1	6,861	73,878	0,000	15,471	0,143	95,733	5.129,386
P_2	5,244	67,425	0,018	14,096	0,232	89,734	43.229,085
P_3	100,000	70,796	0,018	0,000	0,000	89,726	163.405,000
MEDIA	9,408	63,737	0,007	8,283	0,099	91,799	3.221,091

Tabella 5.4. Valori delle metriche registrate nei punti P_1, P_2 e P_3 e media delle metriche calcolate sull'intero dataset dell'host_1.

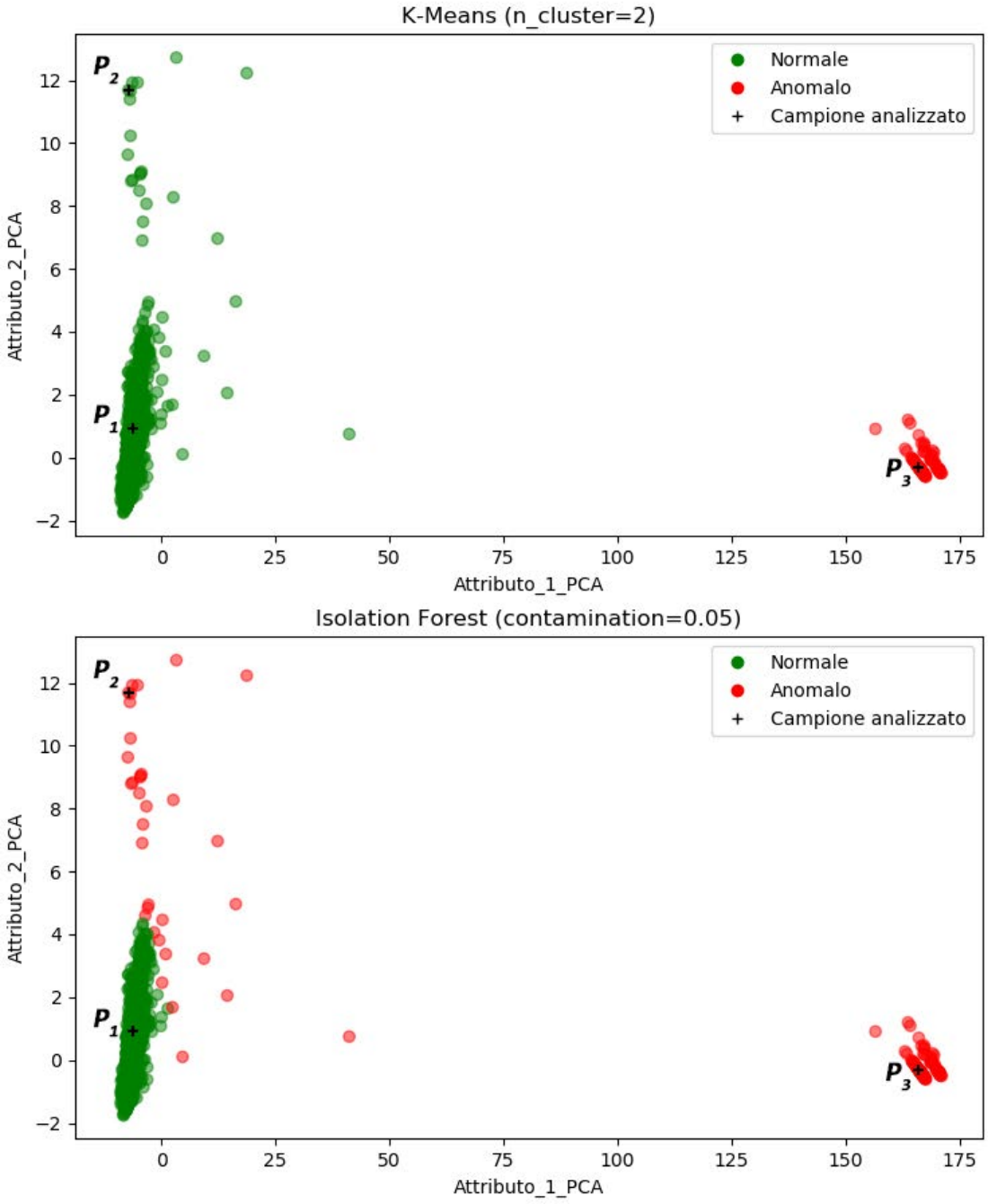

Figura 5.1. Risultati ottenuti dal k-Means e Isolation Forest sul dataset dell'host_1.

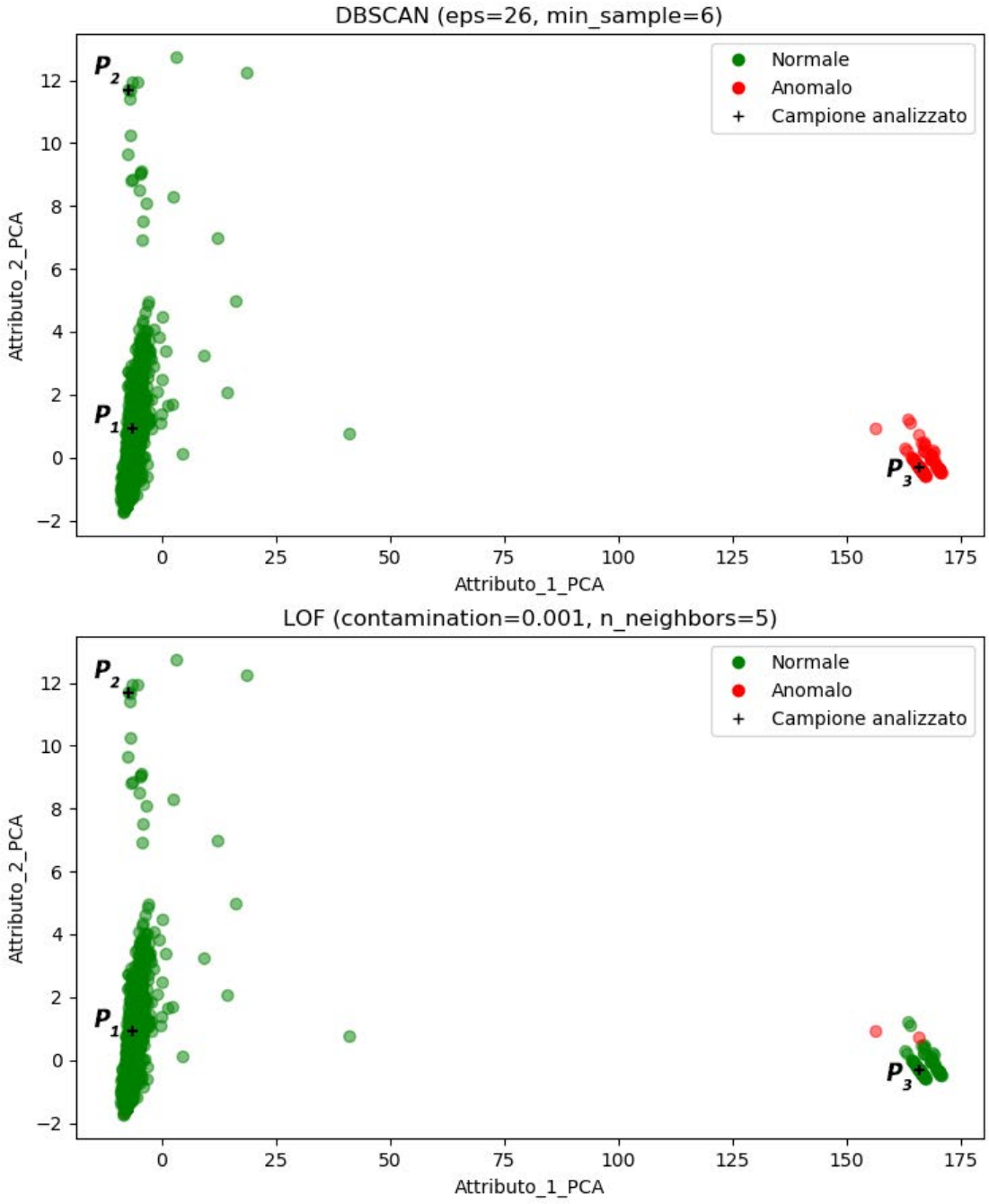

Figura 5.2. Risultati ottenuti dal DBSCAN e LOF sul dataset dell'host_1.

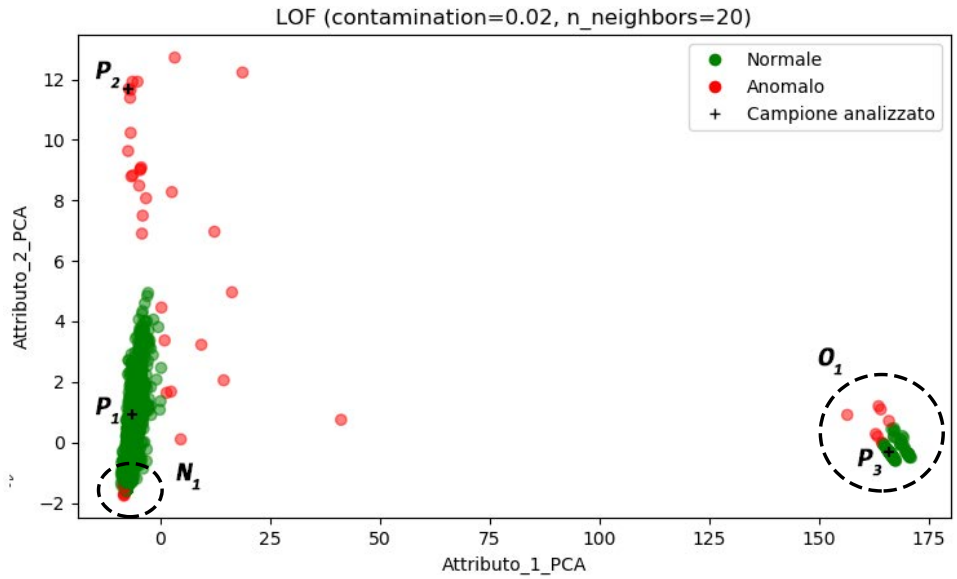

Figura 5.3. Miglior risultato ottenuto dal LOF sul dataset dell'host_1, tarando manualmente i parametri di configurazione.

5.1.2 Caso Studio n° 2

Il secondo test, in riferimento alle Tab. 5.11 e 5.12, è stato eseguito sul target "**host_54**". In questo caso studio, a differenza del primo esperimento proposto, non si hanno conoscenze a priori sulla natura dei campioni di dati del dataset da analizzare.

Algoritmo	Parametri	Silhouette Score
k-Means	k=4	**0.959**
Isolation Forest	contamination=0.002	**0.959**
DBSCAN	eps=28.6 min_samples=7	**0.959**
LOF	contamination=0.003 n_neighbors=28	0.956

Tabella 5.5. Tuning dei parametri e punteggi di Silhouette ottenuti durante il secondo test.

Il dataset in esame è composto complessivamente da 2291 campioni di dati. Durante l'attività di monitoraggio e di raccolta dei dati per questo dataset non sono stati riscontrati particolari problemi di performance per cui, non è possibile sapere a priori eventuali presenze di anomalie all'interno del dataset.

Nella Tab. 5.5, sono stati riportati i valori dei principali parametri utilizzati dagli algoritmi di anomaly detection e i punteggi di Silhouette ottenuti in questo test.

Anche per il secondo esperimento, le Fig. 5.4 e 5.5 rappresentano graficamente come i diversi algoritmi di anomaly detection identificano il comportamento normale e le anomalie del target "host_54". I risultati sono poi stati riassunti, ad alto livello, in Tab. 5.6.

Algoritmo	**#Dati Normali**	**#Dati Anomali**
k-Means	2114	177
Isolation Forest	2286	5
DBSCAN	2114	177
LOF	2284	7

Tabella 5.6. Riassunto dei risultati ottenuti dagli algoritmi di anomaly detection sul dataset dell'host_54 costituito da un totale di 2291 campioni di dati.

Anche in questo caso, come è possibile notare graficamente in Fig. 5.4 e 5.5, la configurazione automatica dei principali parametri dell'iForest e LOF non ha prodotto risultati soddisfacenti. Per questo motivo, i parametri di inizializzazione degli algoritmi sono stati tarati manualmente e i migliori risultati ottenuti sono visibili in Fig. 5.6.

Come ulteriore termine di paragone, sono stati identificati e riportati in Tab. 5.7 i valori di tre punti rappresentativi delle aree di maggior interesse P_1, P_2 e P_3. Oltre ai valori dei tre punti indicati, è stata riportata per ciascuna metrica la media calcolata sull'intero dataset in oggetto.

Punto	**CPU**	**RAM**	**SWAP**	**NET RD**	**NET WR**	**DISK**	**DISK IO**
P_1	3,386	44,249	0,000	0,002	0,001	32,670	39,943
P_2	55,256	47,351	0,000	0,002	0,002	32,680	42,899
P_3	1,273	32,114	0,000	0,002	0,002	32,458	9,951
MEDIA	3,514	43,726	0,000	0,002	0,001	32,680	40,459

Tabella 5.7. Valori delle metriche registrate nei punti P_1, P_2 e P_3 e media delle metriche calcolate sull'intero dataset dell'host_54.

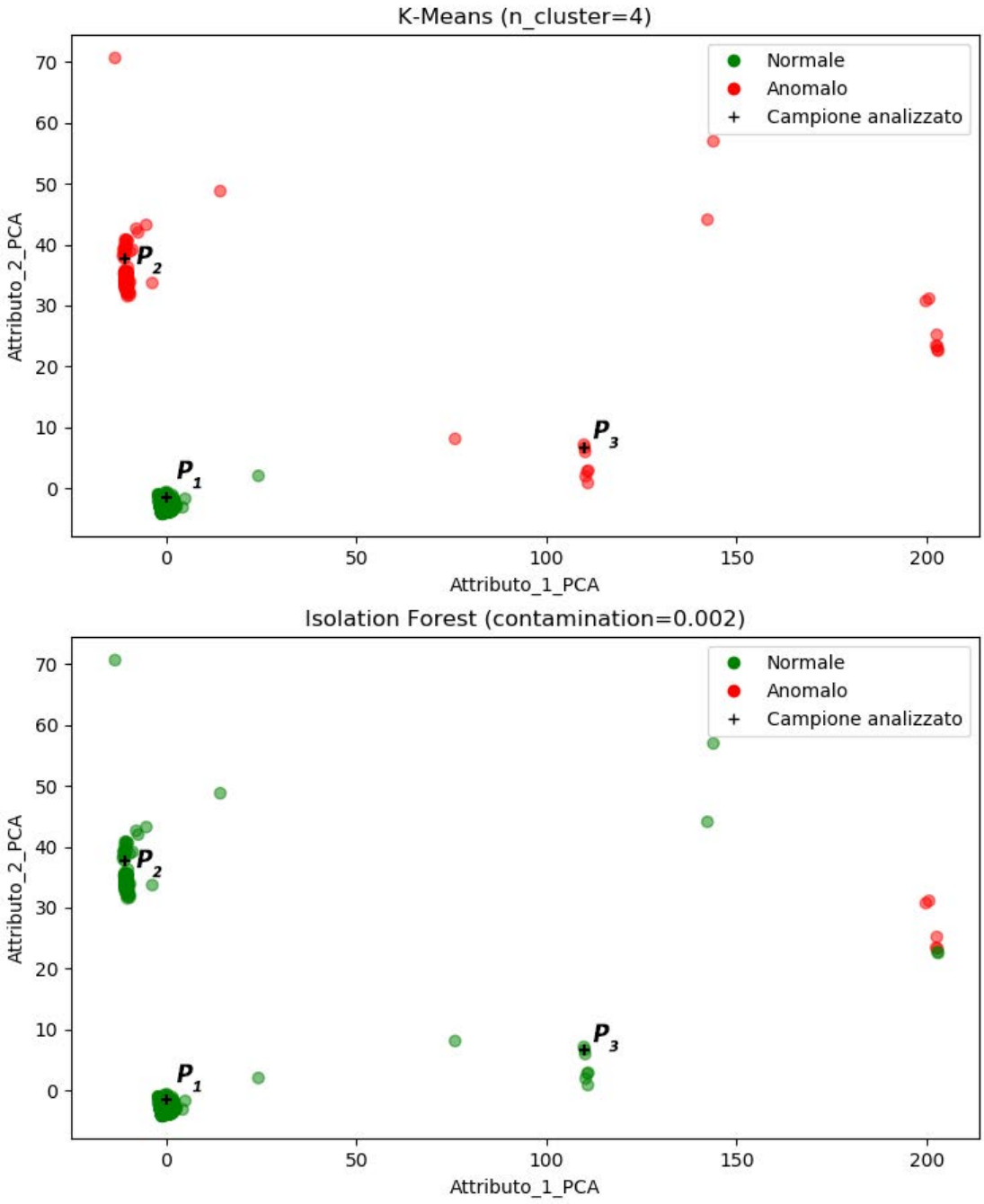

Figura 5.4. Risultati ottenuti dal k-Means e Isolation Forest sul dataset dell'host_54. Nota: il grafico del k-Means mostra solamente due cluster poiché i cluster di piccole dimensioni sono stati raggruppati in un unico cluster che identifica le anomalie.

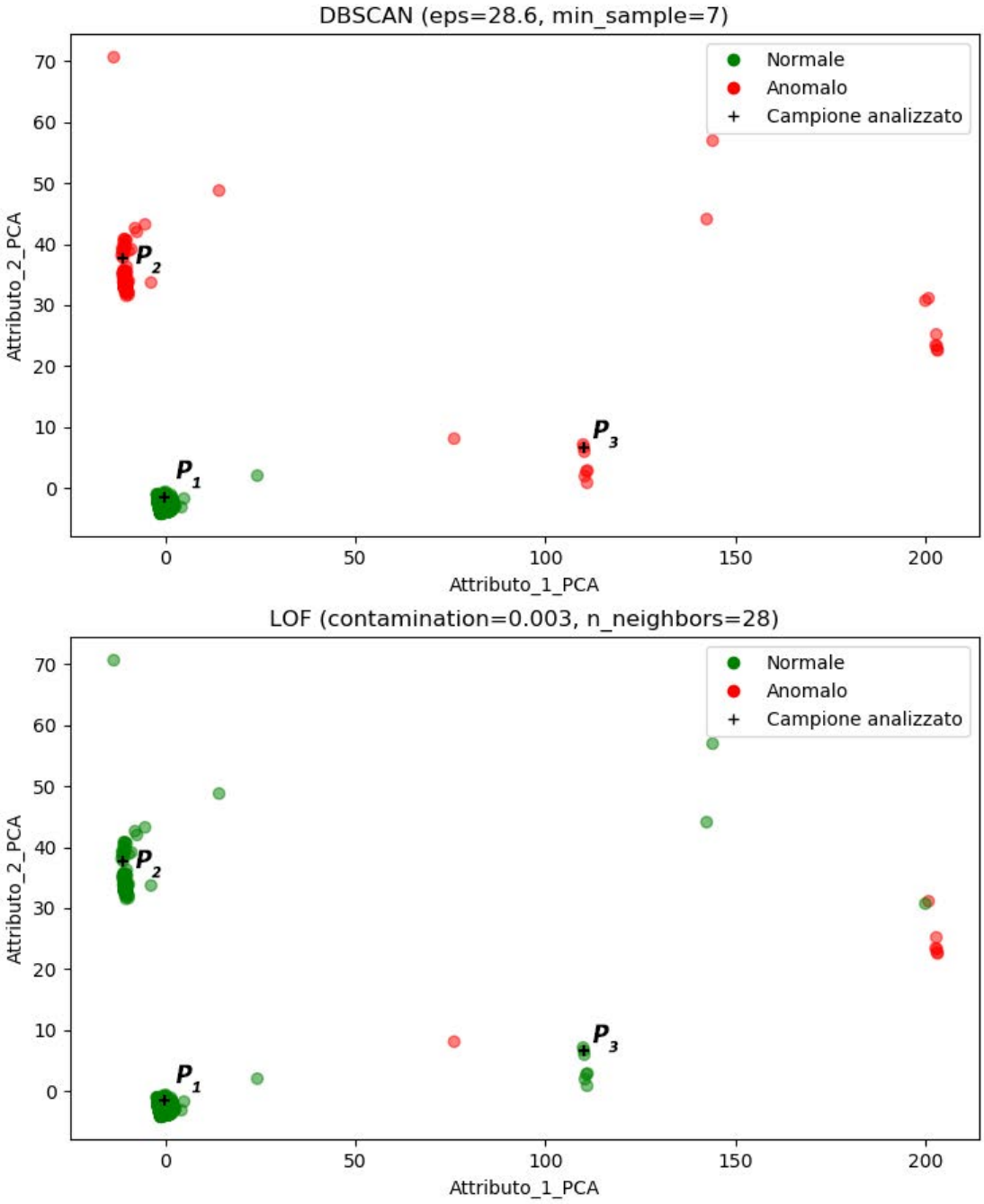

Figura 5.5. Risultati ottenuti dal DBSCAN e LOF sul dataset dell'host_54.

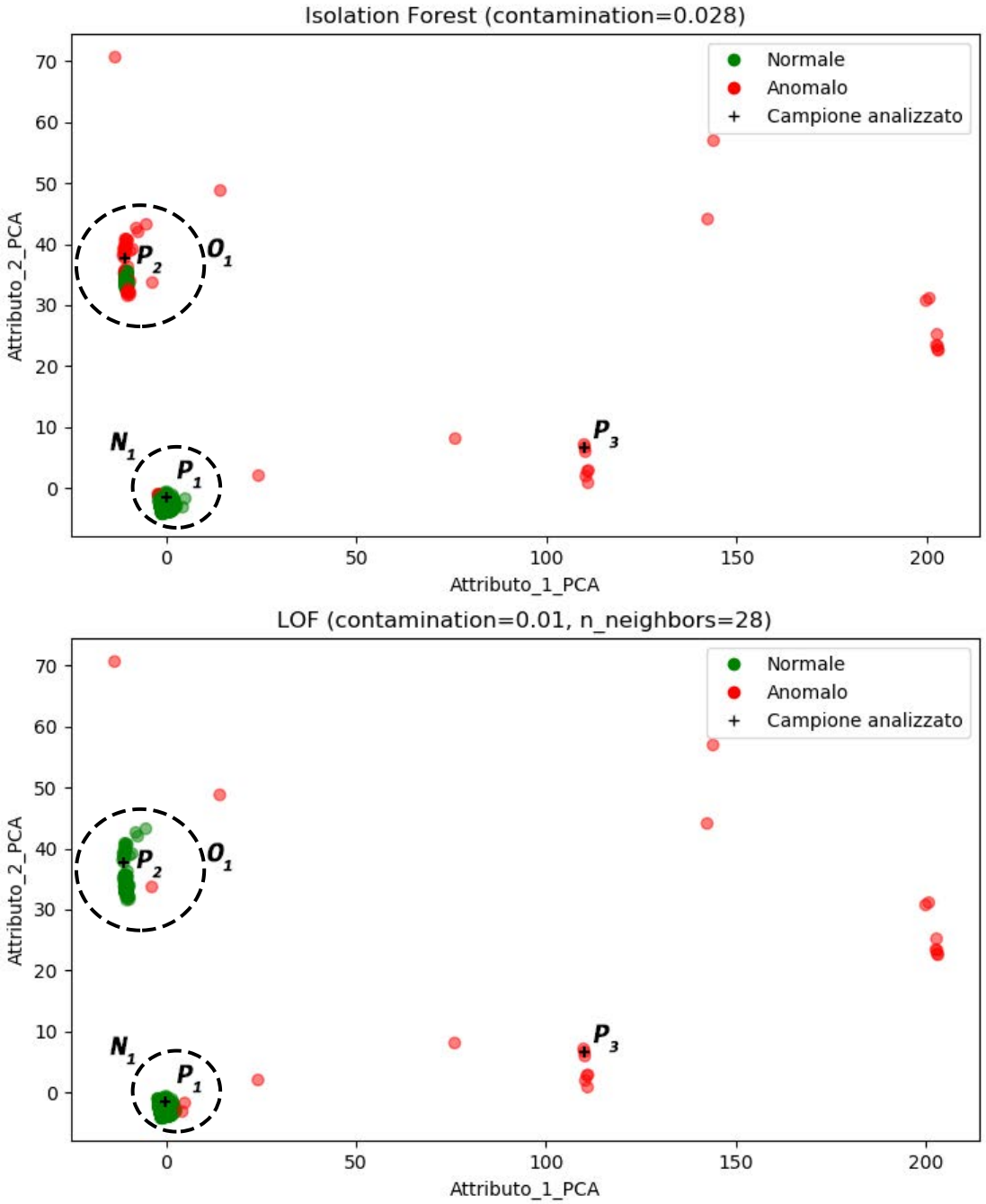

Figura 5.6. Miglior risultato ottenuto da iForest e LOF sul dataset dell'host_54, tarando manualmente i parametri di configurazione.

5.1.3 Caso Studio n° 3

Il terzo ed ultimo esperimento proposto, in riferimento alle Tab. 5.11 e 5.12, è stato eseguito sul target "**host_28**". In questo caso studio, come nel precedente esperimento, non si hanno conoscenze a priori sulla natura dei campioni di dati che costituiscono il dataset dell'host_28 da analizzare.

Il dataset in esame è composto complessivamente da 2304 campioni di dati. Durante l'attività di monitoraggio e di raccolta dei dati anche per questo dataset non sono stati riscontrati particolari problemi di performance per cui, non è possibile sapere a priori eventuali presenze di anomalie all'interno del dataset.

Nella Tab. 5.8, sono stati riportati i valori dei principali parametri utilizzati dagli algoritmi di anomaly detection e i punteggi di Silhouette ottenuti in questo test.

Algoritmo	**Parametri**	**Silhouette Score**
k-Means	k=2	0.872
Isolation Forest	contamination=0.001	**0.962**
DBSCAN	eps=10.4 min_samples=5	**0.962**
LOF	contamination=0.001 n_neighbors=17	**0.962**

Tabella 5.8. Tuning dei parametri e punteggi di Silhouette ottenuti durante il terzo test.

Come nei precedenti esperimenti, le Fig. 5.7 e 5.8 rappresentano graficamente come i diversi algoritmi di anomaly detection identificano il comportamento normale e le anomalie del target "host_28" mentre i risultati sono stati riassunti, ad alto livello, in Tab. 5.9.

Algoritmo	**#Dati Normali**	**#Dati Anomali**
k-Means	2125	179
Isolation Forest	2301	3
DBSCAN	2301	3
LOF	2301	3

Tabella 5.9. Riassunto dei risultati ottenuti dagli algoritmi di anomaly detection sul dataset dell'host_54 costituito da un totale di 2304 campioni di dati.

Come ulteriore termine di paragone, sono stati identificati e riportati in Tab. 5.10 i valori di due punti rappresentativi delle aree di maggior interesse P_1 e P_2. Oltre

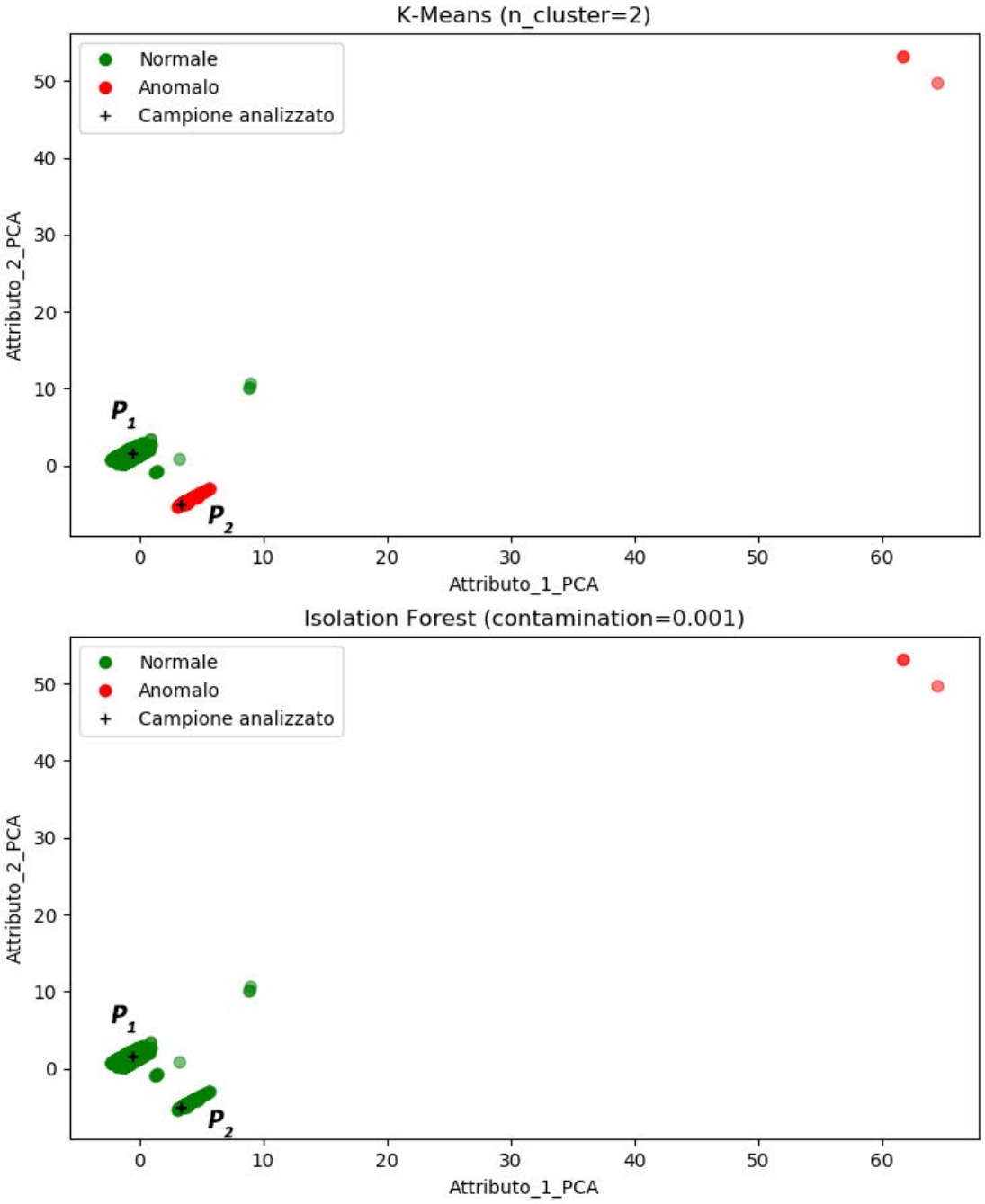

Figura 5.7. Risultati ottenuti dal K-Means e Isolation Forest sul dataset dell'host_28.

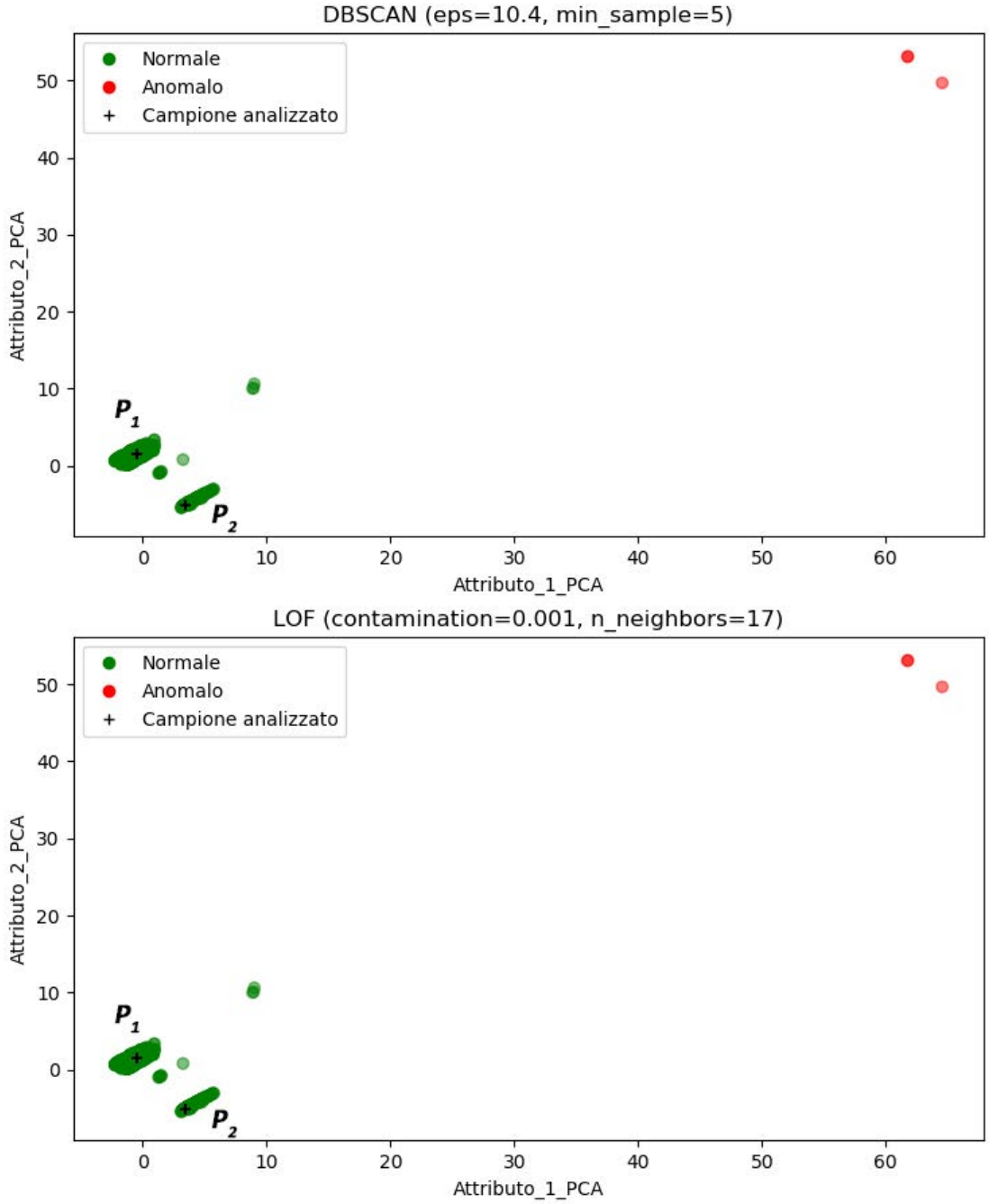

Figura 5.8. Risultati ottenuti dal DBSCAN e LOF sul dataset dell'host_28.

ai valori dei due punti indicati, è stata riportata per ciascuna metrica la media calcolata sull'intero dataset in oggetto.

Punto	**CPU**	**RAM**	**SWAP**	**NET RD**	**NET WR**	**DISK**	**DISK IO**
P_1	3,863	59,194	0,000	0,003	0,000	41,441	4,969
P_2	3,444	60,920	0,000	0,002	0,000	39,511	5,514
MEDIA	3,720	60,470	0,000	0,002	0,000	39,630	5,248

Tabella 5.10. Valori delle metriche registrate nei punti P_1 e P_2 e media delle metriche calcolate sull'intero dataset dell'host_28.

Host	**k-Means**	**iForest**	**DBSCAN**		**LOF**	
	k	contamination	eps	min_samples	contamination	n_neighbors
host_1	2	0.050	26	6	0.001	5
host_2	6	0.001	30.0	5	0.003	5
host_3	2	0.001	2.6	9	0.001	27
host_4	2	0.019	9.0	7	0.001	18
host_5	2	0.002	14.2	9	0.003	8
host_6	2	0.002	13.0	4	0.002	16
host_7	2	0.011	4.6	4	0.001	5
host_8	2	0.001	19.4	3	0.001	11
host_9	2	0.005	39.8	11	0.004	25
host_10	2	0.008	18.6	11	0.007	27
host_11	7	0.003	31.4	3	0.001	4
host_12	3	0.001	11.2	11	0.001	22
host_13	6	0.002	13.6	6	0.001	8
host_14	2	0.002	4.0	11	0.001	23
host_15	2	0.001	3.0	11	0.001	23
host_16	2	0.001	15.0	10	0.001	16
host_17	3	0.057	13.2	3	0.073	5
host_18	2	0.001	15.2	10	0.008	20
host_19	2	0.001	9.0	11	0.001	17
host_20	3	0.006	18.0	9	0.001	16
host_21	2	0.002	16.8	4	0.006	28
host_22	2	0.001	3.6	3	0.001	16
host_23	2	0.002	35.8	10	0.002	17
host_24	2	0.010	9.2	11	0.003	20
host_25	2	0.007	4.8	5	0.002	26
host_26	2	0.001	2.8	11	0.001	20

Host	**k-Means**	**iForest**	**DBSCAN**		**LOF**	
	k	contamination	eps	min_samples	contamination	n_neighbors
host_27	2	0.003	11.0	11	0.004	28
host_28	2	0.001	10.4	5	0.001	17
host_29	2	0.004	23.4	11	0.004	26
host_30	2	0.003	29.8	5	0.001	20
host_31	2	0.003	1.4	6	0.003	17
host_32	2	0.005	36.0	10	0.001	16
host_33	3	0.003	12.2	7	0.001	16
host_34	2	0.004	20.0	11	0.009	29
host_35	2	0.004	37.2	4	0.003	26
host_36	2	0.003	32.6	8	0.003	24
host_37	2	0.001	13.4	7	0.001	4
host_38	2	0.020	27.6	11	0.019	5
host_39	2	0.002	23.2	9	0.002	28
host_40	2	0.004	8.8	11	0.004	19
host_41	2	0.114	1.6	9	0.001	6
host_42	2	0.001	2.0	11	0.001	15
host_43	2	0.057	3.8	11	0.004	13
host_44	2	0.006	2.6	5	0.006	24
host_45	2	0.015	31.6	11	0.001	21
host_46	2	0.002	6.6	11	0.001	17
host_47	2	0.001	14.0	9	0.001	24
host_48	2	0.001	38.6	11	0.003	26
host_49	2	0.001	29.4	8	0.001	4
host_50	2	0.004	33.2	3	0.003	29
host_51	3	0.001	4.0	4	0.001	28
host_52	3	0.001	2.8	6	0.001	18

Host	**k-Means**	**iForest**	**DBSCAN**		**LOF**	
	k	contamination	eps	min_samples	contamination	n_neighbors
host_53	2	0.007	32.6	11	0.007	21
host_54	4	0.002	28.6	7	0.003	28
host_55	2	0.002	26.4	5	0.002	6
host_56	2	0.003	28.0	8	0.003	24
host_57	2	0.003	9.6	11	0.003	29
host_58	3	0.001	35.0	8	0.001	29
host_59	2	0.009	33.8	4	0.016	29
host_60	2	0.001	28.4	7	0.001	20
host_61	2	0.007	18.2	8	0.002	24
host_62	3	0.011	16.6	3	0.077	5
host_63	5	0.005	37.8	9	0.002	3
host_64	2	0.060	1.4	11	0.001	17
host_65	2	0.004	33.6	7	0.010	29
host_66	2	0.004	4.6	8	0.003	23
host_67	2	0.001	31.8	11	0.002	18
host_68	2	0.003	34.2	10	0.003	20

Tabella 5.11: Tuning, per ciascun target host, dei principali parametri degli algoritmi di anomaly detection.

Host	Coefficiente di Silhouette			
	k-Means	**Isolation Forest**	**DBSCAN**	**LOF**
host_1	**0.988**	0.955	**0.988**	0.866
host_2	0.975	**0.977**	0.968	0.954
host_3	0.698	0.798	**0.815**	0.794
host_4	0.920	0.922	**0.924**	0.848
host_5	0.898	0.905	**0.917**	0.847
host_6	0.906	0.947	**0.954**	0.927
host_7	0.904	**0.905**	0.901	0.825
host_8	0.892	0.935	**0.940**	**0.940**
host_9	0.982	0.979	**0.983**	0.982
host_10	**0.994**	0.991	0.992	0.993
host_11	**0.970**	0.915	0.968	0.883
host_12	0.853	**0.910**	**0.910**	0.909
host_13	0.848	0.885	**0.905**	**0.905**
host_14	0.785	0.867	**0.870**	0.819
host_15	0.690	**0.777**	**0.777**	**0.777**
host_16	**0.984**	0.979	**0.984**	0.979
host_17	**0.973**	0.932	0.959	0.477
host_18	0.850	**0.919**	0.916	0.893
host_19	**0.994**	**0.994**	**0.994**	**0.994**
host_20	0.980	0.979	0.980	**0.983**
host_21	0.922	**0.944**	0.943	0.883
host_22	**0.979**	0.970	**0.979**	0.970
host_23	**0.986**	**0.986**	0.962	**0.986**
host_24	**0.952**	0.949	**0.952**	0.813
host_25	0.856	0.883	**0.885**	0.740
host_26	0.750	**0.756**	**0.756**	0.718
host_27	**0.986**	**0.986**	0.966	0.985
host_28	0.872	**0.962**	**0.962**	**0.962**
host_29	**0.996**	**0.996**	0.994	**0.996**
host_30	**0.993**	0.992	0.988	**0.993**
host_31	**0.997**	0.993	**0.997**	0.995
host_32	**0.998**	0.987	**0.998**	**0.998**
host_33	**0.995**	0.991	0.989	0.993
host_34	0.966	0.964	**0.968**	0.961
host_35	0.973	0.973	**0.976**	0.974
host_36	**0.984**	**0.984**	**0.984**	**0.984**
host_37	0.964	**0.965**	0.964	0.956
host_38	**0.968**	**0.968**	0.964	0.575

Host	Coefficiente di Silhouette			
	k-Means	**Isolation Forest**	**DBSCAN**	**LOF**
host_39	**0.991**	**0.991**	0.986	0.989
host_40	**0.989**	**0.989**	**0.989**	**0.989**
host_41	**0.827**	0.813	0.802	0.671
host_42	0.682	0.785	**0.805**	0.754
host_43	**0.896**	0.871	0.886	0.734
host_44	0.822	**0.864**	**0.864**	0.863
host_45	0.953	**0.955**	0.954	0.947
host_46	0.776	0.855	**0.863**	**0.863**
host_47	0.956	**0.963**	0.957	0.945
host_48	0.951	**0.965**	0.943	0.925
host_49	0.981	**0.984**	0.975	**0.984**
host_50	**0.996**	0.994	0.989	0.995
host_51	0.550	0.815	**0.833**	0.769
host_52	0.682	0.929	**0.970**	0.918
host_53	**0.954**	0.945	0.943	0.951
host_54	**0.959**	**0.959**	**0.959**	0.956
host_55	**0.993**	0.992	**0.993**	**0.993**
host_56	**0.964**	0.963	**0.964**	**0.964**
host_57	**0.963**	0.961	**0.963**	0.962
host_58	0.940	0.958	**0.962**	0.953
host_59	0.956	**0.958**	**0.958**	0.932
host_60	**0.985**	0.983	0.960	**0.985**
host_61	**0.966**	0.962	**0.966**	0.880
host_62	**0.972**	0.968	0.945	0.527
host_63	**0.973**	0.972	0.965	0.947
host_64	**0.874**	0.857	0.852	0.794
host_65	**0.990**	**0.990**	0.986	0.982
host_66	**0.992**	0.990	0.990	0.991
host_67	**0.985**	0.981	**0.985**	0.984
host_68	**0.993**	0.992	0.987	0.992
MEDIA	0.923	0.940	**0.942**	0.900

Tabella 5.12: Per ciascun target host, il massimo punteggio di Silhouette ottenuto dagli algoritmi di anomaly detection.

5.2 Discussione dei Risultati

Come già anticipato durante la presentazione dei risultati, a causa di un elevato numero di host analizzati durante il lavoro di tesi svolto, sono stati proposti i tre casi studio ritenuti più significativi per la valutazione degli algoritmi di anomaly detection, che meglio identificano le anomalie di un sistema informatico monitorato attraverso l'Oracle Enterprise Manager.

Entrando nei particolari di ciascun caso studio, il primo esperimento è stato eseguito su un dataset di cui si conosceva a priori un'importante informazione sulla natura di un sottoinsieme di dati del dataset; ovvero, 93 su 2447 campioni di dati sono da considerare come "anomalie". La conferma di quanto appena detto, è visibile in Tab. 5.4, dove il cluster rappresentato dal punto P_3 ha valori di utilizzo della "CPU" e "DISK IO" estremamente elevati rispetto sia alla media dell'intero dataset che del punto P_1 rappresentante del cluster "normale".

Come è possibile notare, sia dai punteggi di Silhouette ottenuti dagli algoritmi in Tab. 5.2 che dai risultati grafici in Fig. 5.1 e 5.2, *LOF* è stato l'algoritmo meno performante sul dataset in esame, mentre k-Means, DBSCAN e iForest hanno ottenuto ottimi risultati identificando correttamente le 93 anomalie conosciute.

Non avendo ricevuto risultati soddisfacenti dal "LOF", si è deciso di tentare di ottenere risultati migliori, tarando manualmente i parametri di inizializzazione dell'algoritmo. Il miglior risultato, mostrato in Fig. 5.3, non raggiunge ugualmente l'efficienza degli altri algoritmi. Infatti, non solo non è stato in grado di identificare correttamente il cluster anomalo O_1, ma ha erroneamente identificato come anomalie alcuni punti all'interno dell'area N_1, la quale si colloca estremamente vicina alla zona di massima concentrazione dei punti "normali".

Un'ulteriore considerazione va fatta sui 30 campioni di dati anomali in più, identificati dall'Isolation Forest rispetto al k-Means e DBSCAN. Seppur non è da considerare un chiaro errore di classificazione, è possibile notare dalla Tab. 5.4 come, a parte un lievissimo aumento dell'utilizzo di SWAP e un aumento della metrica "DISK IO", il punto P_2 sia molto simile al punto "normale" P_1. Inoltre, è stato verificato manualmente che i 30 punti anomali in più, non sono temporalmente vicini l'uno rispetto all'altro e che quindi possono essere considerati come piccole variazioni sporadiche.

Nel secondo test, come è possibile notare in Fig. 5.4 e 5.5, la densità più alta di dati risulta concentrata graficamente intorno alle coordinate (0,0); infatti, su un totale di 2291 campioni di dati, 2114 sono collocati nella regione circostante. Anche in questo test, il DBSCAN e il k-Means hanno ottenuto le migliori prestazioni sia in termini di coefficiente di Silhouette che in termini di rappresentazione grafica. Non si può dire la stessa cosa dell'Isolation Forest che, nonostante abbia ottenuto

lo stesso punteggio di Silhouette del k-Means e DBSCAN mostrati in Tab. 5.5, graficamente non è stato in grado di identificare il resto delle anomalie. Stesso discorso dell'esperimento precedente vale per l'algoritmo LOF che, anche in questo test, ha ottenuto la peggiore prestazione risultando poco idoneo ai tipi di dataset analizzati nel lavoro di tesi svolto.

I cluster rappresentati dai punti P_2 e P_3 sono da considerare "anomali" in quanto, come è possibile notare dalla Tab. 5.7, i punti circostanti a P_2 hanno subito un incremento del 50% di utilizzo della CPU rispetto all'andamento normale dell'host, mentre il punto P_3 ha avuto un decremento dell'utilizzo della memoria RAM del 15% e una drastica riduzione della metrica "DISK IO". Da un punto di vista del monitoraggio, quest'ultima anomalia potrebbe essere associata ad un crash di uno o più processi in esecuzione nell'host sotto analisi.

Anche in questo secondo test, per gli algoritmi meno performanti (iForest e LOF), si è tentato di tarare manualmente e meticolosamente i loro parametri di inizializzazione per ottenere dei risultati migliori. Come è possibile vedere in Fig. 5.6, i migliori risultati di entrambi gli algoritmi, non garantiscono la corretta identificazione delle anomalie O_1 e dei dati normali N_1.

Infine, nel terzo esperimento è graficamente evidente, in Fig. 5.7 e 5.8, come il k-Means identifichi erroneamente l'insieme dei punti rappresentati da P_2 come anomalie piuttosto che dati normali. Come è possibile notare dalla Tab. 5.10, il punto P_2 ha valori molto simili sia alla media delle metriche ottenute sull'intero dataset, sia al punto P_1 considerato da tutti gli algoritmi come un dato normale.

In conclusione, come si evince dal coefficiente di Silhouette ottenuto dalla media dei 68 casi analizzati in Tab. 5.12 e dai tre casi studio presentati in questo lavoro di tesi, le migliori prestazioni rispetto ai dataset considerati sono state raggiunte dall'algoritmo **DBSCAN**.

Capitolo 6

Conclusioni

L'era del Machine Learning e dell'innovazione tecnologica, ha indotto l'azienda a valutare la possibilità di applicare tecniche di data mining, ed in particolare di anomaly detection, nell'ambito del monitoraggio di sistemi complessi e ingegnerizzati.

L'idea alla base di questa ricerca è quella di integrare ai diffusi sistemi tradizionali di monitoraggio, basati principalmente sulla configurazione statica di soglie critiche, una soluzione di anomaly detection per l'identificazione automatica di pattern anomali, i quali sono spesso rari e ben nascosti all'amministratore del sistema. In questo lavoro di tesi è stato quindi proposto un sistema di anomaly detection basato sulla piattaforma di monitoraggio Oracle Enterprise Manager.

Non avendo a priori conoscenze sui dati raccolti dai target monitorati dall'OEM, sono state messe a confronto quattro diverse tecniche di anomaly detection con apprendimento automatico non supervisionato: k-Means, DBSCAN, Isolation Forest e Local Outlier Factor (LOF). L'applicazione di queste tecniche, e conseguentemente il lavoro di ricerca svolto, si è basata sull'assunzione che le anomalie, per definizione, sono rare e hanno caratteristiche molto diverse dal resto dei dati del dataset.

La ricerca si è focalizzata solamente sui tipi di target "host". Sono stati analizzati 68 host differenti, raccogliendo per ciascuno di essi, metriche relative allo stato della CPU, della memoria, dei dischi fisici e interfacce di rete.

In base ai risultati ottenuti dagli esperimenti condotti e alla natura del dataset generato durante il lavoro di ricerca, le tecniche di anomaly detection basati sul clustering risultano essere i modelli con le migliori prestazioni. Tra gli algoritmi valutati nella fase sperimentale, il DBSCAN è stata la tecnica di anomaly detection che ha ottenuto i risultati migliori.

Attraverso gli esperimenti svolti è stato dimostrato che con le tecniche di anomaly detection è possibile rilevare anomalie che tendenzialmente con i sistemi tradizionali sono difficilmente identificabili. Un esempio pratico è il caso studio n°2 in cui, grazie

al sistema di anomaly detection, è possibile rilevare automaticamente l'incremento del 50% di utilizzo della CPU di un host, anomalia difficilmente identificabile da un sistema tradizionale la cui soglia di warning e/o critical è staticamente impostata intorno al 90%.

I risultati incoraggianti ottenuti da questa ricerca, hanno messo in evidenza la possibilità di applicare le tecniche di anomaly detection sui sistemi monitorati dall'Oracle Enterprise Manager e di implementare un ulteriore strumento di monitoraggio, in grado di rilevare automaticamente anomalie sulle prestazioni dei sistemi in gestione.

6.1 Punti Aperti

Il lavoro di tesi svolto è un primo tentativo di applicazione delle tecniche di anomaly detection sui dati raccolti dai sistemi informatici monitorati tramite il software Oracle Enterprise Manager. I risultati ottenuti dai primi test svolti sono molto promettenti anche se consideriamo che l'implementazione di una soluzione affidabile, per la rilevazione automatica di potenziali anomalie, richieda ancora molta ricerca. Per continuare questo lavoro iniziale, di seguito sono elencati possibili sviluppi futuri della ricerca:

- Prendere in considerazione altri algoritmi di anomaly detection per ottenere prestazioni migliori;
- Analizzare ed applicare i modelli di anomaly detection sulle metriche raccolte per i tipi di target differenti dall'host (ad esempio "oracle_database", "weblogic_j233server", "j2ee_application", ecc...);
- Analizzare i requisiti hardware richiesti per l'implementazione del sistema proposto.

Bibliografia

[1] Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[2] Enterprise Manager Cloud Control Management Repository Views Reference, https://docs.oracle.com/cd/E73210_01/EMVWS

[3] A.Sari, "A Review of Anomaly Detection Systems in Cloud Networks and Survey of Cloud Security Measures in Cloud Storage Applications", Journal of Information Security, Vol. 6, No. 2, April 2015, pp. 142-154, DOI 10.4236/jis.2015.62015

[4] J.Han, M.Kamber, J.Pei, "Data Mining: Concepts and Techniques", Morgan Kaufmann Publishers Inc., 2005, ISBN: 1558609016

[5] P.K.Agarwal, N.H.Mustafa, "k-means projective clustering", PODS '04 Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, Parigi (Francia), Jun 14-16, 2004, pp. 155-165, DOI 10.1145/1055558.1055581

[6] M.Ester, H-P Kriegel, J. Sander, X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise", KDD'96 Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Oregon, Portland (USA), Aug 02-04, 1996, pp. 226-231,

[7] F.T.Liu, K.M.Ting, Z-H.Zhou, "Isolation Forest", 2008 Eighth IEEE International Conference on Data Mining, Pisa (Italia), Dec 15-19, 2008, pp. 413-422, DOI 10.1109/ICDM.2008.17

[8] M.M.Breunig, H-P.Kriegel, R.T.Ng, J. Sander, "LOF: identifying density-based local outliers', SIGMOD '00 Proceedings of the 2000 ACM SIGMOD international conference on Management of data, Dallas, Texas (USA), May 15-18, 2000, pp. 93-104, DOI 10.1145/342009.335388