POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica (LM-32)

Tesi di Laurea Magistrale

Implementazione ed integrazione di modelli non supervisionati e predittivi per la rilevazione di anomalie nelle metriche di performance di una base dati.

Politecnico di Torino

Relatore:
Prof.ssa Tania Cerquitelli

Candidata:
Daniela De Angelis
277493

Tutor aziendali:
Dott. Antonio Greco
Dott. Graziano Fracasso

Anno Accademico 2020-2021

Sommario

In seguito alla continua crescita del settore IT in ambito economico-aziendale, la gestione dei malfunzionamenti assume un ruolo sempre più significativo e rilevante nel mercato odierno. In particolare, diverse imprese si trovano a gestire e mantenere molteplici applicazioni in un ambiente IT condiviso e composto da diverse componenti interdipendenti, rendendo ardua la manutenzione dei servizi. Di conseguenza, è necessario che l'infrastruttura tecnologica venga monitorata e mantenuta, mediante dei servizi specifici. L'obiettivo di questa tesi consiste nella realizzazione di una struttura di base per uno strumento di supporto all'attività di monitoraggio, in grado di garantire una valutazione tempestiva delle problematiche e l'individuazione e visualizzazione puntuale delle metriche di performance di una base di dati, che presentano anomalie. Tale soluzione è stata ideata per essere integrata negli strumenti di monitoraggio esistenti in una azienda, con lo scopo di prevenire gli eventi bloccanti mediante identificazione di comportamenti anomali nella base di dati, garantire una più rapida ed efficiente rilevazione e risoluzione dei problemi, diminuire i costi aziendali ed aumentare il livello di credibilità e affidabilità dell'azienda nei confronti del cliente. Il progetto proposto si focalizza sull'analisi e selezione delle metriche di performance più significative della base di dati in esame, e successiva individuazione e confronto delle più opportune metodologie data driven di rilevazione delle anomalie. In particolare, si esegue una selezione delle metriche di performance, basata sull'applicazione iterativa di algoritmi di clustering partizionale e analisi di correlazione basate sul coefficiente di correlazione di Pearson. In seguito, vengono implementati e confrontati due differenti metodi per la rilevazione delle anomalie. Il primo si basa sull'utilizzo di modelli di serie temporali, mentre il secondo sfrutta modelli di apprendimento automatico non supervisionato. Confrontando i due metodi, è emerso che non vi è un metodo nettamente più performante di un altro. Questo a causa delle caratteristiche intrinseche del segnale, che possono inficiare l'efficienza del metodo stesso. Conseguentemente, è stato proposto un modello combinato dei due metodi, che potesse sfruttare i vantaggi di entrambi. Il modello combinato ha fornito dei risultati ottimali in questo lavoro di tesi. In prospettiva futura, tale modello verrà ulteriormente migliorato e utilizzato in altri campi applicativi, per comprenderne le potenzialità e gli aspetti innovativi.

Indice

Capitolo 1

Introduzione

In questo capitolo introduttivo viene fornita una breve panoramica relativa all'importanza dei servizi di manutenzione e rilevazione delle anomalie nelle applicazioni in un contesto aziendale, viene descritta l'azienda nella quale il progetto di tesi è stato svolto, ed infine l'obiettivo della tesi e la struttura dell'elaborato.

1.1 Contesto

Recenti ricerche condotte da IDC in Italia hanno evidenziato l'importanza degli investimenti in infrastruttura IT. Per le aziende, tali investimenti risultano decisivi per l'avvio del processo di evoluzione digitale, al fine di incrementare la crescita del business e la responsività alle richieste di innovazione. Più dell'80% delle aziende in Italia registra dei problemi riconducibili ad errate scelte architetturali nell'infrastruttura tecnologica. Tale situazione, può limitare fortemente il processo di ammodernamento applicativo, con conseguente vanificazione dell'implementazione di soluzioni innovative e all'avanguardia [1].
Ne consegue, che l'infrastruttura tecnologica debba avere elevate prestazioni hardware e software, un alto grado di flessibilità e stabilità, e un tempo di attività ininterrotto. Perciò è necessario che l'infrastruttura tecnologica venga monitorata e mantenuta, mediante dei servizi specifici, che prendono il nome di *Application Maintenance Services* (AMS). I servizi di manutenzione delle applicazioni (AMS), sono essenziali per il corretto funzionamento delle applicazioni eseguite sui server. Contribuiscono al ripristino rapido delle normali funzionalità del sistema, garantendo elevati livelli di qualità e disponibilità del servizio, e riduzione dell'impatto aziendale in termini di costi e immagine. Nello specifico, l'obiettivo ultimo dell'AMS è la gestione dei malfunzionamenti avvenuti nelle applicazioni, la correzione di eventuali guasti o errori nell'infrastruttura IT e, se possibile, la previsione di eventi futuri dannosi per il sistema [2].

In seguito alla continua crescita del settore IT in ambito economico-aziendale, la gestione degli malfunzionamenti assume un ruolo sempre più significativo e rilevante nel mercato odierno. In particolare, diverse imprese si trovano a gestire e mantenere molteplici applicazioni in un ambiente IT condiviso e composto da diverse componenti interdipendenti, quali rete, hardware, software, ecc, rendendo ardua la manutenzione dei servizi. Tra i clienti che maggiormente utilizzano AMS, vi sono i fornitori di servizi (banche, compagnie aeree, ecc.) che necessitano una manutenzione dei server per l'archiviazione dei dati, gestione dei database o dei software, o le imprese del retail per la gestione di magazzini o di sistemi di e-commerce. Le cause dei malfunzionamenti sono molto spesso complicate e in alcuni casi dipendono da più fattori scatenanti. Questo richiede una diagnosi ed una indagine accurata. È necessario, quindi, l'ausilio di piattaforme analitiche opportune, in grado di valutare metriche di performance di un servizio, identificare i comportamenti anomali e, se possibile, prevedere potenziali guasti. In particolare, quando le metriche monitorate si comportano in modo anomalo, possono essere inviati dei warning ai gestori del sistema. Le anomalie nelle metriche possono essere identificate mediante diverse tecniche di rilevamento (*anomaly detection algorithms*), che permettono di individuare nei dati, pattern non conformi con il comportamento atteso.
Nella sezione 2.2, verranno chiariti gli aspetti inerenti le varie tecniche di anomaly detection, utili soprattutto per l'analisi di serie temporali.

1.2 Azienda

Il lavoro di ricerca, oggetto di questa tesi, è stato svolto presso Mediamente Consulting s.r.l, una società di consulenza specializzata nella gestione e valorizzazione dei dati. E' stata fondata nel 2013, come startup innovativa e nel 2016 l'Incubatore di Imprese Innovative del Politecnico di Torino, l'ha riconosciuta come startup dell'anno, per meriti di crescita in termini di occupazione e fatturato. Attualmente, infatti, è caratterizzata da 4 sedi operative in tutta Italia.

La mission di Mediamente Consulting è quella di *guidare l'innovazione delle imprese attraverso la Business Analytics* [3]. Pertanto, l'azienda sfrutta software e servizi per trasformare i dati in informazioni utili per prendere decisioni strategiche, per effettuare monitoraggio, valutazione delle prestazioni, ed eventuale aggiornamento dell'infrastruttura tecnologica e applicativa del cliente.
I principali clienti di Mediamente Consulting sono medie e grandi imprese di diversi settori: dal Fashion al manifatturiero, dal Retail al Food & Beverage, E-commerce ed Healthcare.

L'azienda è caratterizzata da diverse aree di specializzazione: Infrastructure, Corporate Performance Managment, Advanced Analytics, Data Integration e Managment e Business Intelligence. La business unit di Data Integration si occupa dei processi che consentono l'integrazione dei dati dei clienti, provenienti da diverse sorgenti, in una vista logica ed unificata. Infatti, uno dei processi fondamentali della Data Integration è l'ETL (Extract, Transform, Load), ossia il processo di estrazione, trasformazione, caricamento dei dati dal sistema sorgente in un Data Warehouse. L'unità di Business Intelligence, invece, combina data mining e visualizzazione dei dati per l'estrazione di informazioni utili da grandi quantità di dati. Inoltre, insieme all'area di Advance Analytics e Corporate Performance Managment cercano di generare le migliori scelte strategiche per un business.
L'area di Infrastruttura Tecnologica si occupa di gestire e mantenere l'infrastruttura tecnologica e applicativa dei clienti (database Oracle, middleware, sistema operativo). I servizi forniti spaziano dalla manutenzione e aggiornamento periodico dei sistemi software, all'analisi delle metriche performance dei sistemi.

1.3 Obiettivo

Attualmente l'area di Infrastruttura Tecnologica di Mediamente Consulting, si affida alla piattaforma Oracle Enterprise Manager, la quale offre una soluzione completa di monitoraggio e gestione per Oracle Database e Engineered Systems distribuiti sul cloud o data centers dei clienti. In aggiunta, i dipendenti dell'area IT sfruttano una dashboard da loro personalizzata per permettere la visualizzazione simultanea dei diversi sistemi dei clienti. La dashboard è caratterizzata da un insieme di box colorati. Ciascun box corrisponde ad un singolo sistema dei clienti, mentre il colore associato identifica la criticità del sistema: al verde corrisponde un livello di criticità nullo; al giallo viene associato un livello medio/basso; al rosso un livello alto ed, infine, al nero la criticità è massima. La dashboard, inoltre, consente di minimizzare il tempo di intervento, aumentare la reattività e l'organizzazione. Tuttavia, sono disponibili altri due canali di alerting in aggiunta alla dashboard: il primo prevede una comunicazione diretta tra operatore e cliente, mentre il secondo consiste in un messaggio di testo proveniente da un bot automatizzato di Telegram. Lo svantaggio di questo sistema risiede nell'approccio attualmente utilizzato, basato sull'esperienza pregressa. Nello specifico, è richiesta una notevole capacità dell'operatore per identificare il reale problema presente nel sistema ed agire tempestivamente. Dunque, il tutto si può tradurre in una valutazione puramente soggettiva da parte dell'operatore, e non su un'analisi oggettiva e approfondita dei dati. Inoltre, il tutto viene complicato dalla presenza di problemi nel sistema che non vengono segnalati. Da qui, la necessità di sviluppare un tool in grado di rilevare anomalie, in tempo reale, nelle metriche di performance del

database Oracle, al fine di effettuare un intervento proattivo. Tale soluzione deve essere integrata all'interno degli strumenti di monitoraggio esistenti.
Il seguente lavoro di tesi, quindi, si propone di realizzare una base per uno strumento di supporto all'attività di monitoraggio, in grado di garantire una valutazione tempestiva delle problematiche e l'individuazione e visualizzazione puntuale delle metriche che presentano anomalie.
I valori aggiunti di questo lavoro di tesi sono: (i) prevenzione degli eventi bloccanti mediante identificazione di comportamenti anomali del database; (ii) una più rapida ed efficiente rilevazione e risoluzione problemi; (iii) diminuzione dei costi aziendali; (iv) incremento in termini di credibilità e affidabilità dell'azienda nei confronti del cliente.

Il resto della tesi è organizzato come segue:
Nel secondo capitolo, a partire dall'analisi della letteratura allo stato attuale, vengono brevemente descritti alcuni metodi di selezione delle caratteristiche e delle metodologie per la rilevazione di anomalie su serie temporali. Nel capitolo 3 vengono presentate le tecnologie utilizzate durante il lavoro di tesi. La soluzione metodologia è illustrata nel capitolo 4. Vengono descritti in modo dettagliato il processo di individuazione e raccolta delle metriche di performance, il processo di preparazione dei dati ed analisi esplorativa, il processo di selezione delle metriche di performance e i metodi utilizzati per la rilevazione delle anomalie nelle metriche. Nello specifico, il processo di selezione delle metriche viene descritto in modo completo e sequenziale, mostrando anche i risultati ottenuti e il quantitativo di metriche utilizzate per la successiva fasi di individuazione delle anomalie. I risultati sperimentali, relativi all'individuazione delle anomalie, sono riportati nel capitolo 5. Infine, il capitolo sei è dedicato alle conclusioni e le prospettive future.

Capitolo 2

Stato dell'arte

In questo capitolo sono riportate le principali nozioni teoriche relative ai concetti di base del machine learning, processi di feature selection e metodologie per la rilevazione di anomalie. L'obiettivo è introdurre le tecniche e gli algoritmi utilizzati in questo lavoro e presentare delle soluzioni alternative provenienti dall'analisi della letteratura, allo stato attuale.

Il machine learning è una branca dell'intelligenza artificiale. Il termine Intelligenza Artificiale è nato alla fine degli anni '50, ed è stato associato a come modellare il processo della mente umana attraverso un calcolatore. Nello stesso periodo, il documento di Turing "*Computing Machinery and Intelligence*" (1950), e il suo successivo Turing Test, hanno stabilito l'obiettivo fondamentale e la visione dell'intelligenza artificiale. E' stata introdotta l'idea che le macchine possono replicare o simulare l'intelligenza umana. Ciò ha dato luogo a molte domande e dibattiti. Di fatti, nessuna definizione singolare dell'intelligenza artificiale è universalmente accettata. Solitamente, il campo dell'intelligenza artificiale viene definita da quattro diversi approcci noti come: *"thinking humanly, thinking rationally, acting humanly, acting rationally"*[4]. L'apprendimento automatico, o machine learning, studia algoritmi per la rilevazione e l'estrazione di pattern da dati grezzi e predizione di nuove informazioni alla luce di quelle apprese, con un intervento umano ridotto al minimo. Tali algoritmi, quindi, permettono la costruzione di un modello che impara automaticamente a predire nuovi dati a partire da osservazioni, differenziandosi dagli algoritmi classici, basati su istruzioni statiche. Attualmente, il machine learning è utilizzato in diversi settori, come quello medico, finanziario, industriale. In particolare, può accellarare il processo di analisi dei big data, set estremamente grandi di dati strutturati e non, con l'aiuto di algoritmi decisionali e tradurre i dati in informazioni utili per le operazioni aziendali.
Esistono differenti modalità di apprendimento, suddivise in categorie in base al loro scopo e al tipo di informazione disponibile (2.1). I principali paradigmi sono:

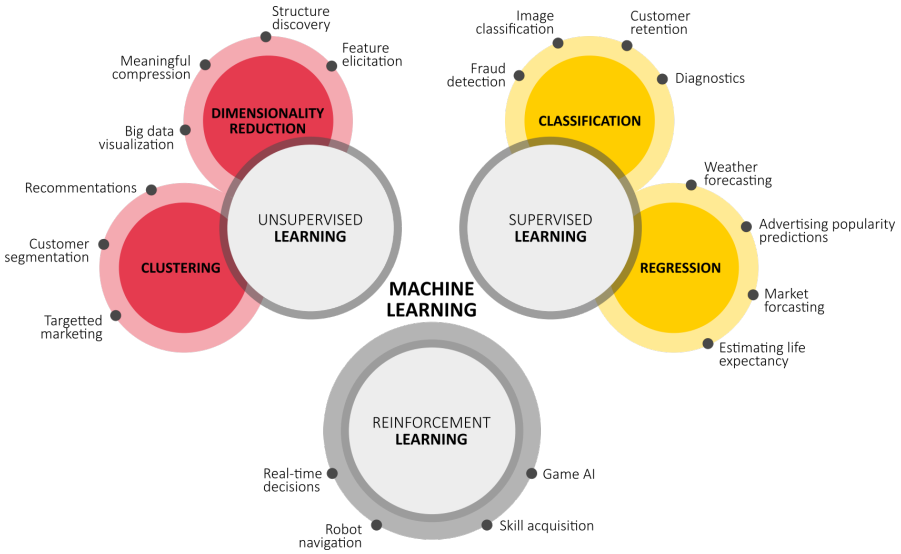


Figura 2.1. Classificazione schematica dei modelli di machine learning. Immagine da [5].

Supervised learning, Unsupervised Learning e Reinforcement Learning [5].

Apprendimento supervisionato

Gli algoritmi di apprendimento supervisionati possono essere utilizzati quando i dati di addrestramento sono associati ad una specifica etichetta, spesso noti con il termine di labeled data. Tali algoritmi cercano di modellare le relazioni e dipendenze tra le caratteristiche del dataset in input, *features*, e l'output predittivo, *variabile target*. Pertanto, si possono predire i valori di output per i nuovi dati, in base alle relazioni che ha imparato dai set di dati precedenti. Di conseguenza, a partire dall'analisi di un *training set*, ovvero un insieme di dati espressi sotto forma di coppie (variabili in input "x" e valore target corretta "y"), l'algoritmo elabora un modello predittivo. Durante la fase di apprendimento dell'algoritmo, confrontando l'output prodotto con quello corretto e previsto, l'algoritmo stesso può regolare, in modo appropriato, il proprio modello.

La classificazione e la regressione sono due esempi di modelli di tipo supervisionato. L'obiettivo del classificatore è imparare una funzione in grado di predire la classe o la categoria di appartenenza per una data osservazione. Gli input possono essere mono-dimensionali o multi-dimensionali, e la categoria di appartenenza del dato in input può essere scelta tra due o più classi. Un esempio di classificazione binaria è la suddivisione delle email, come appartenenti alla classe "spam" o "non

spam". La regressione differisce rispetto alla classificazione, in quanto predice un valore reale continuo. Ad esempio, partendo da un dataset di una lista di immobili, caratterizzati da i metri quadri, zona e prezzo, inserendo in input i metri quadri e la zona, l'algoritmo deve essere in grado di restituire in output il prezzo più probabile.

Apprendimento non supervisionato

Nell'apprendimento senza supervisione, i dati di addrestramento sono privi di etichette. I modelli unsupervised hanno come obiettivo principale trovare somiglianze e anomalie, non sempre evidenti per un essere umano, all'interno di un set di dati di tipo unlabeled, non etichettato, e categorizzarli in propri raggruppamenti . Le classi di raggruppamento, pertanto, devono essere apprese automaticamente dall'algoritmo stesso. Il principale svantaggio associato a tali algoritmi è la difficoltà di misurazione delle prestazioni, mentre la principale potenza risiede nella capacità di trovare modelli interessanti che possono superare le aspettative iniziali. Di fatti, se nell'apprendimento supervisionato la logica di classificazione viene conferita alla macchina come input, nell'apprendimento supervisionato è compito della macchina individuarne una [5], [6].

Un esempio di apprendimento senza supervisione è il clustering, volto alla selezione e raggruppamento dei dati, mediante misure relative alla similità o dissimilarità tra gli elementi, in uno spazio multidimensionale.

Apprendimento per rinforzo

L'algoritmo di apprendimento per rinforzo, o agente, è considerato il sistema di apprendimento più complesso, in quanto mira ad utilizzare le informazioni raccolte tramite l'interazione con l'ambiente circostante per intraprendere azioni che riducano al minimo il rischio o massimizzino la ricompensa. Pertanto, l'agente osserva lo stato di input, compie un'azione mediante l'utilizzo di una funzione decisionale, riceve una ricompensa o rinforzo dall'ambiente, e memorizza le informazioni relative alla coppia stato-azione e ricompensa ottenuta. La funzione di rinforzo, quindi, misura il grado di successo di un'azione o decisione, rispetto a un obiettivo predeterminato [6].
Tali algoritmi, vengono principalmente utilizzati in ambito robotica, automazione, o video gaming.

2.1 Metodi di selezione delle caratteristiche

La selezione delle caratteristiche, o feature selection, è il processo di riduzione del numero di variabili di input, mediante rimozione delle feature irrilevanti, ridondanti o rumorose [7], [8]. La selezione delle caratteristiche, di solito, può migliorare le prestazioni e l'accuratezza del modello, ridurre il costo computazionale della modellazione e migliorare l'interpretabilità del modello. Inoltre, previene l'overfitting ed evita il problema noto come "*the curse of dimensionality*", o maledizione della dimensionalità. Quest'ultima si riferisce a vari fenomeni che sorgono quando si analizzano dati in spazi ad alta dimensione, che risultano dispersi [9]. Questa dispersione è problematica, dato che ciascun elemento sembra essere dissimile rispetto agli altri, impedendo l'applicazione di tecniche di organizzazione dei dati efficienti. Dato che la ricerca esaustiva di un sottoinsieme di caratteristiche ottimali, nella maggior parte dei casi, non è praticabile, sono state proposte diverse strategie di ricerca. Di seguito, viene fornita una panoramica generale e strutturata delle diverse metodologie, presenti in letteratura, utilizzate per la selezione delle features, riportata anche schematicamente in figura 2.2.

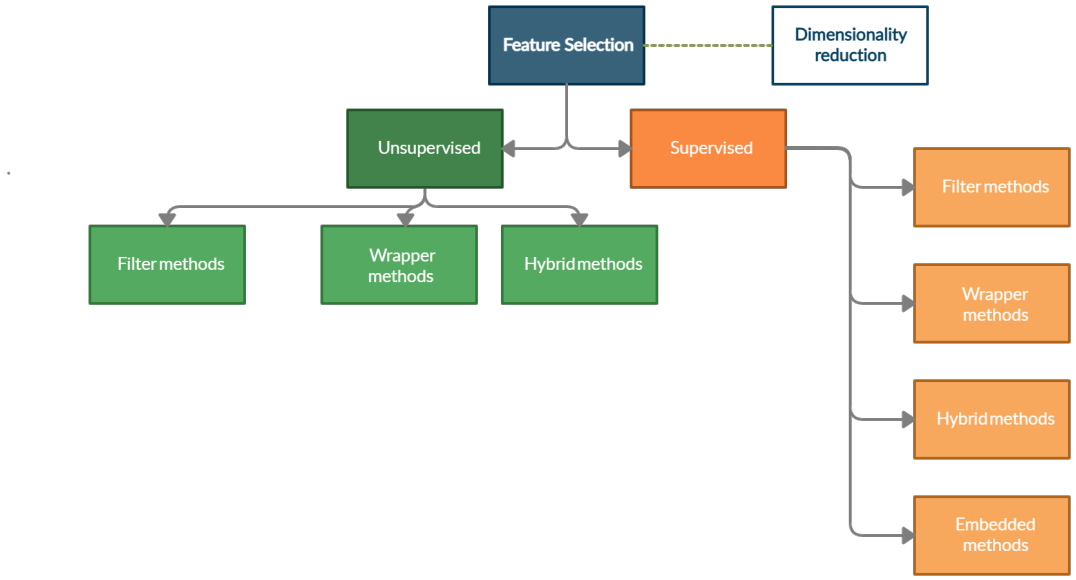


Figura 2.2. Tassonomia ad alto livello per la feature selection

Le tecniche di selezione delle caratteristiche possono essere suddivise principalmente in due categorie: metodi supervisionati e non supervisionati.

La disponibilità di dati di addestramento etichettati consente l'utilizzo di algoritmi di feature selection di tipo supervisionato. Tali algoritmi identificano le caratteristiche rilevanti ed utili per raggiungere al meglio l'obiettivo del modello supervisionato (ad es. classificazione o problema di regressione).
Le tecniche di selezione delle caratteristiche non supervisionate, invece, ignorano la variabile target, e rimuovono le features ridondanti in base ai valori di correlazione. Le metodologie per la selezione delle features, possono essere suddivise ulteriormente in *filtering methods, wrapper methods, hybrid methods*, e nel caso supervisionato anche *embedded methods* [7].
In generale, *i metodi wrapper* valutano più modelli utilizzando procedure che aggiungono e/o rimuovono i predittori per trovare la combinazione ottimale che massimizza le prestazioni del modello. Sono computazionalmente più costosi dei metodi di filtraggio, a causa delle fasi di apprendimento ripetute e della cross-validazione. Tuttavia, questi metodi sono più accurati del metodo del filtro. Alcuni esempi sono gli algoritmi di eliminazione delle features ricorsiva, o algoritmi per la selezione delle features sequenziali (recursive feature elimination e sequential feature selection algorithms)[10].
Gli hybrid methods, permettono una combinazione dei metodi filter e wrapper, in modo tale da ridurre la complessità temporale dei metodi wrapper.
Nei *metodi basati sul filtering*, invece, le caratteristiche sono selezionate in base a misure statistiche o nel caso non supervisionato, misure in grado di valutare il grado di dispersione dei dati, come l'entropia e la divergenza, in modo tale da identificare, se possibile, subset (clusters) di features. Tali metodologie sono quindi indipendenti dagli algoritmi di apprendimento e richiedono un limitato tempo computazionale [7], [8]. Uno dei metodi di feature selection, di tipo unsupervised e filtering, è l'*FSFS*, Feature Selection using Feature Similarity [11], e una delle misure statistiche maggiormente utilizzate è il coefficiente di correlazione, *Pearson correlation coefficient* [12]. Entrambi, verranno approfonditi nelle successive sezioni di tale documento (sezione 2.1.1, sezione 2.1.2). Una tecnica alternativa per la sezione delle features, che può essere utilizzata in combinazione con i filter methods, prevede l'aggiunta di algoritmi di clustering. Tali algoritmi, consentono di rimuovere i dati rumorosi o irrilevanti e di comprendere in modo ottimale la struttura delle caratteristiche. Alcuni tra i metodi di clustering più comunemente utilizzati sono stati sfruttati in questo progetto di tesi, tra cui K-means clustering (sezione 2.1.3) e DBSCAN (sezione 2.1.4). Si illustraranno, inoltre, esclusivamente le potenzialità e le caratteristiche tecniche dell'algoritmo k-Shape (sezione 2.1.5) [13]. Le proprietà del k-Shape, lo rendono uno strumento compatibile per il clustering delle time series, tuttavia non è stato preso in considerazione durante questo lavoro di tesi in quanto è ancora in fase sperimentale e non del tutto ottimizzato. Nonostante ciò, potrebbe risultare una valida alternativa per applicazioni future inerenti alla selezione delle features.

2.1.1 Coefficiente di correlazione di Pearson

Per studiare le correlazioni di coppia, pair-wise correlations, tra due variabili è utile usare il coefficiente di correlazione di Pearson, una misura della correlazione lineare tra esse. E' definito come la covarianza delle due variabili, divisa per il prodotto delle loro deviazioni standard, in modo che il risultato abbia sempre un valore compreso tra - 1 e 1 [12]. L'equazione è la seguente:

$$\rho(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y} \tag{2.1}$$

dove:

- $\rho(x, y)$= coefficiente di correlazione di Pearson;
- $cov(x, y) = E[(x - E[x])(y - E[y])]$;
- σ_x = deviazione standard di x;
- σ_y = deviazione standard di y.

Come con la covarianza stessa, la misura può riflettere esclusivamente una correlazione lineare tra i dati, e ignora molti altri tipi di relazione o correlazione. Un valore del coefficiente di Peason uguale a 1 evidenzia la presenta di una perfetta correlaizone positiva tra le due variabili in esame. Pertanto, per un aumento positivo della prima variabile, vi è un aumento positivo anche della seconda. Un valore del coefficiente pari a -1, invece, implica la presenza di una perfetta correlazione negativa tra le due variabili. Questo mostra che le stesse si muovo in direzioni opposte: per un aumento positivo in una variabile, c'è una diminuzione nella seconda. Se la correlazione tra le due variabili è 0, allora, non vi è alcuna relazione lineare tra di esse.
La matrice di correlazione è la tabella che viene utilizzata per mostrare la correlazione tra tutte le possibili coppie di variabili. Spesso, per facilitare la lettura ed interpretazione della matrice di correlazione, la stessa viene presentata mediante un grafico heatmap, dotato di una scala di colori utile ad evidenziare le correlazioni più alte e quelle più basse [14].

2.1.2 FSFS

Il metodo FSFS, Feature Selection using Feature Similarity, [11] introduce una misura statistica di similarità per ridurre la ridondanza delle features, chiamata *Maximal Information Compression Index* (MICI). Si basa sulla varianza-covarianza tra le caratteristiche e viene definita come segue:

$$2\lambda_2(x, y) = var(x) + var(y) - \sqrt{(var(x) + var(y))^2 - 4var(x)var(y)(1 - \rho(x, y)^2)} \tag{2.2}$$

Nell'equazione 2.2, $var(x)$ e $var(y)$ rappresentano rispettivamente la varianza di x, e la varianza di y, mentre $\rho(x, y)$ è il coefficiente di correlazione di Pearson (eq. 2.1). L'idea di questo metodo è di partizionare l'insieme originale di features in clusters, in modo tale che le features presenti in uno stesso cluster sono molto simili tra di loro, mentre quelle presenti in altri cluster sono diverse.
Il clustering delle features viene realizzato iterativamente in base al principio *kNN*, K-nearest neighbors, come segue: in ogni iterazione, FSFS calcola le k features più vicine di ciascuna caratteristica (usando MICI). Successivamente, la features con il sottoinsieme più compatto di k features vicine (determinato dalla distanza dalla sua features più lontana tra le k più vicine) viene selezionata, mentre le altre k features vicine vengono scartate. Questa procedura viene ripetuta per le features rimanenti fino a quando tutte loro saranno o raccolte o scartate [7], [11].

Algoritmi di clustering

Il clustering è uno dei metodi di apprendimento non supervisionati maggiormente utilizzato per l'analisi esplorativa dei dati. È il processo di creazione di gruppi di dati più piccoli, chiamati clusters, a partire dal set di dati originale, in modo tale che gli elementi appartenenti a un gruppo risultino "simili" tra loro e differenti dagli elementi negli altri gruppi (segue, alta intra-class similarity, bassa inter-class similarity) [15]. In tal modo, ciascun cluster dovrebbe rappresentare una classe. Il concetto di somiglianza è complesso e non semplice da definire. Dobbiamo pensare in termini di distanza tra variabili casuali. Gli algoritmi di clustering possono essere classificati in:

- *Partitional clustering*: richiede all'analista di definire il numero K di clusters prima di eseguire l'algoritmo. In seguito, raggruppa gli oggetti in sottoinsieme di cluster non sovrapposti, sicchè ogni oggetto appartiene esattamente a un cluster [16].

- *Hierarchical clustering*: produce un insieme di cluster annidati organizzati come un albero gerarchico [17].

Gli algoritmi di clustering possono essere ulteriormente suddivisi in clustering esclusivo o non esclusivo (i punti possono appartenere a più cluster), probabilistico, parziale vs completo, eterogeneo vs omogeneo.

Di seguito, verranno approfonditi esclusivamente gli algoritmi di clustering partizionali, utili nel mio lavoro di ricerca. In particolar modo, verranno illustriati l'algoritmo *K-means clustering*, *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN), e il *K-shape*, un algoritmo di clustering efficiente e preciso per le serie temporali.

2.1.3 K-means clustering

L'algoritmo K-means clustering permette di raggruppare n elementi simili, in K clusters, e di scoprire modelli sottostanti [16]. Di solito, la 'prossimità' può essere misurata mediante la distanza euclidea, cosine similarity, correlazione, ecc. L'algoritmo funziona come segue:

Algoritmo 1 K-means algorithm.

1: Scegli k punti come centroidi iniziali
2: **ripeti**
3: Forma K cluster assegnando ciascun punto al centroide più vicino
4: Ricalcola il centroide di ciascun cluster
5: **finchè** tutti i centroidi si stabilizzano e i punti non si muovono più tra i cluster

È importante scegliere in modo appropriato il numero K di cluster, per individuare il numero di centroidi necessari nel set di dati, ed assegnare a ciascun punto un cluster in base alla vicinanza al suo centroide. Il centroide è tipicamente la media dei punti del cluster.

Per selezionare opportunamente il numero di cluster, all'interno di un dataset, è possibile utilizzare l'*Elbow Method* [18]. Il metodo consiste nel tracciare il valore della funzione di costo in relazione al numero di cluster e selezionare il numero k di cluster in corrispondenza dei punti di inflessione della curva (Figura 2.4). Possono essere usate diverse metriche, dal *distortion score*, calcolato come la somma delle distanze di ciascun punto al suo centroide assegnato al quadrato , *silhouette score*, ossia il coefficiente di silhouette medio per tutti i campioni o il *calinski harabasz score*, che calcola il rapporto di dispersione tra e all'interno dei cluster.

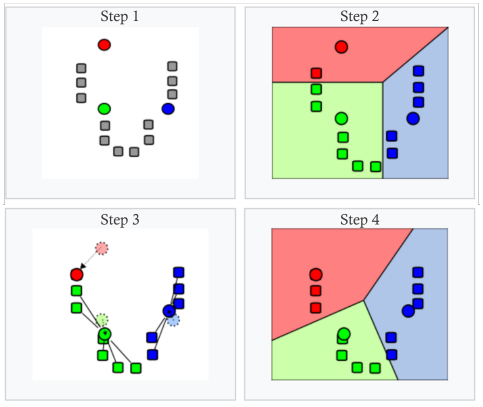


Figura 2.3. Procedimento algoritmo K–means clustering

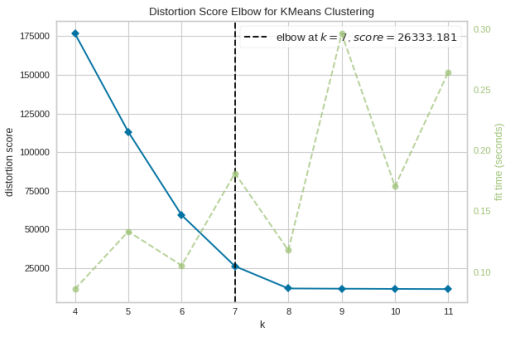


Figura 2.4. Metodo Elbow per il K–means clustering. Immagine da [18]

La complessità dell'algoritmo è O(n * K * I * d), dove n rappresenta il numero di punti, K il numero di cluster, I il numero di iterazioni, d il numero di attributi. Emerge che la complessità di questo algoritmo è all'incirca lineare rispetto al numero di cluster [19], [20]. L'algoritmo k-means converge, ed in alcuni casi a soluzioni sub-ottime. In tal caso, l'algoritmo termina quando abbiamo piccole oscillazioni intorno ad un minimo locale [21].

Punti di forza: Il k-means clustering è un algoritmo semplice, intuibile e converge velocemente. Essendo un algoritmo di clustering non supervisionato, non necessita di un dataset etichettato.
Limiti: La bontà del clustering può essere negativamente influenzata dalla presenza di outlier [22]. Il numero di cluster è fissato a priori e si ottengono buoni risultati quando i cluster sono ben separati e presentano una forma sferica [16].

2.1.4 DBSCAN

Il DBSCAN, Density-Based Spatial Clustering of Applications with Noise, è un metodo di clustering basato sulla densità, che identifica regioni di punti con densità sufficientemente alta. In particolare, l'algoritmo stima la densità attorno a ciascun punto, item, contando il numero di punti presenti in un intorno di raggio eps specificato dall'utente, ed in base al valore del parametro minPts, minimo numero di elementi, identifica gruppi di clusters [23]. L'algoritmo, di fatti, richiede esclusivamente i parametri *minPts*, numero minimo di punti affinchè una regione si possa considerare densa ed *eps*, il raggio. L'algoritmo DBSCAN, classifica i punti in tre diverse categorie, come riportato in figura 2.5:

1. *Core point*: un punto p è un core point, se in intorno di raggio ϵ vi sono almeno minPts punti. Tali punti si dicono direttamente raggiungibili da p, e insieme al punto p formano un cluster;

2. *Border point*: un punto q si definisce border point, se non è un core-point, ma è direttamente raggiungibile da esso, e non può essere utilizzato per raggiungere altri punti. Fanno quindi parte di un cluster, ma ne rappresentano il bordo;

3. *Noise point*: un punto n si definisce noise point, se non è raggiungibile da alcun punto, rappresentando un outlier.

Un cluster, quindi, soddisfa due proprietà: tutti i punti in un cluster sono sono mutuamente density-connected, e un punto raggiungibile da qualsiasi altro data point interno al cluster, risulta parte del cluster stesso.
In particolare, due punti p e q sono definiti *density-connected* se esiste un punto o tale che sia p che q siano raggiungibili da o [23].
Di seguito, viene riportato il funzionamento dell'algoritmo DBSCAN.

Algoritmo 2 Density-Based Spatial Clustering of Applications with Noise.

1: Calcola la distanza tra ciascun punto del dataset;
2: Trova gli ϵ - vicini di ciascun punto, in base al valore del raggio ϵ;
3: Identifica i core points, i punti con più di minPts punti vicini;
4: Assegna al cluster C_i i punti core che abbiano distanza $<$ di ϵ da almeno uno degli altri punti assegnato al cluster;
5: Assegna ciascun punto non-core ad un cluster vicino se il cluster è un ϵ (eps) - vicino, altrimenti lo si assegna a rumore.

La scelta dei parametri del modello, ϵ (esp) e minPts, dovrebbe avvenire in base al dominio del problema. Un valore maggiore per eps si traduce in cluster più ampi, mentre un valore minore stabilisce cluster più stretti. Il parametro minPts specifica il numero minimo di punti (cioè i membri del cluster) necessari per produrre un nuovo cluster. Un valore maggiore di minPts assicura un cluster più robusto ma può escludere alcune aree potenzialmente più piccole in quanto tenta di fonderle in un cluster più grande. D'altra parte, un valore minore estrae molti cluster, ma i cluster risultanti possono includere anche rumore.
In generale, il valore del parametro minPts viene determinato in modo empirico, basandosi sulla conoscenza del dominio e la familiarità con il set di dati. Il valore di ϵ , eps, invece può essere determinato automaticamente mediante una tecnica, che si è rivelata utile in diversi scenari, nota come *k-distance graph*.
L'idea di base è che i k-esimi nearest neighbor, per i core point, si trovino ad una distanza simile e risultino abbastanza vicini. Questa tecnica calcola la distanza media tra ogni punto e i suoi k vicini, dove k è il valore Minpts scelto in precedenza. Le distanze medie vengono poi riportate in ordine ascendente su un grafico, k-distance graph. Il valore ottimale per ϵ viene individuato nel punto di massima curvatura, dove il grafico presenta la maggiore pendenza [24].
In figura 2.6, viene riportato un esempio di k-distance graph.

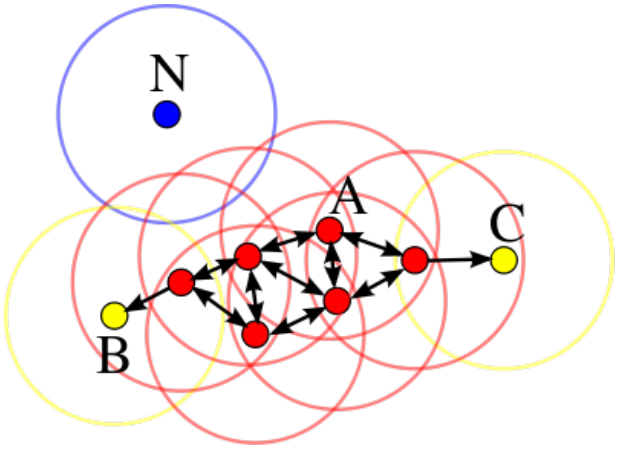


Figura 2.5. In figura, il parametro minPts assume valore 3 o 4. Il punto A e gli altri punti rossi sono core point, poiché l'area che circonda questi punti in un raggio di ampiezza ϵ contiene almeno 3 o 4 punti. I punti B e C sono border point, raggiungibili da A attraverso altri punti core, ed appartenenti, quindi, allo stesso cluster. Il punto N è un noise point, non essendo nè un core point né density-reachable. Immagine da [25]

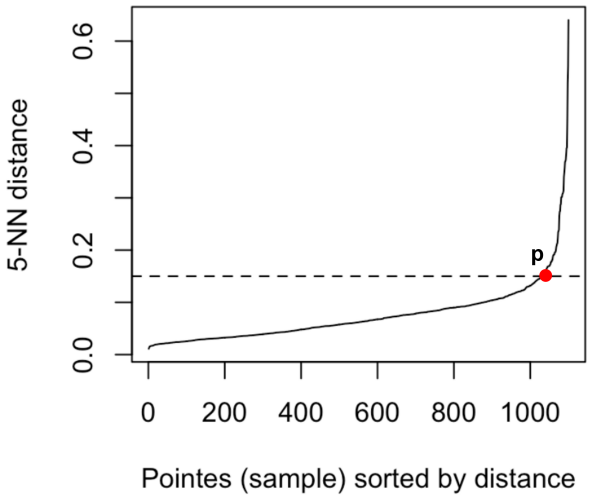


Figura 2.6. K-distance graph per la ricerca del valore ottimale del parametro epsilon del DBSCAN. In figura, il valore di ϵ, eps, è dato dall'ordinata del punto p. Immagine da [26]

Punti di forza: Resistente al rumore, può generare cluster di forma arbitraria, identifica autonomamente gli outliers [23].
Limiti: Le performance si riducono in presenza di dati con elevata dimensionalità, in quanto a causa dell'elevata dispersione degli stessi, risulta difficile definire il concetto di densità. La scelta dei parametri eps e minPts non è semplice e spesso dipende dalla conoscenza del dominio del problema.

2.1.5 K-shape clustering per le serie temporali

Uno degli approcci più avanzati al raggruppamento delle serie temporali è il k-Shape, introdotto per la prima volta nel 2015 da John Paparrizos e Luis Gravano [13]. È un algoritmo di clustering che permette di preservare la struttura delle sequenze temporali durante il confronto, mediante l'utilizzo di una nuova misura di distanza, chiamata *SBD* (*Shape-based distance*, equazione 2.4), che è invariante al ridimensionamento e al cambiamento (scale- and shift-invariant). Tale misura, si basa sulla cross-correlation measure, e permette di calcolare i centroidi dei cluster delle serie temporali [27]. L'algoritmo k-Shape, ad ogni iterazione, esegue due step: (i) aggiorna l'assegnazione delle serie temporali ai cluster, confrontando ciascuna time series con i centroidi calcolati e associando a ciascuna di esse il cluster il cui centroide è il più vicino; (ii) ricalcola i centroidi dei cluster, in modo tale da riflettere i cambiamenti, avvenuti nel passaggio precedente, all'interno di ciascun cluster. Tali operazioni vengono ripetute dal k-shape, fino a quando non si verifica alcun cambiamento nell'appartenenza al cluster o è stato raggiunto il numero massimo di iterazioni scelte per l'algoritmo. Nel primo step, il k-Shape usa la Shape-based distance (equazione 2.4), mentre nel secondo step fa riferimento al metodo per il calcolo dei centroidi. Il calcolo del centroide può essere visto come un problema di ottimizzazione dove l'obiettivo è quello di individuare il minimo della somma delle distanze al quadrato, da tutte le altre sequenze di serie temporali [28]:

$$\vec{c_j} = \arg\min_{\vec{w}} \sum_{\vec{x_i} \in p_j} dist(\vec{w}, \vec{x_i})^2, \vec{w} \in \mathbb{R}^m \tag{2.3}$$

$X = \vec{x_1}, ..., \vec{x_n}$ è il set di osservazioni di input, $P = p_1, ..., p_k$ sono i k cluster disgiunti e $\vec{c_j}$ è il centroide della partizione $\vec{p_j} \in P$. Dato che la cross-correlation individua la *similarity*, piuttosto che la *dissimilarity* delle serie temporali, l'equazione sopra riportata viene riscritta come un problema di massimizzazione.
Mediante ottimizzazioni e considerazioni, si giunge alla formulazione di un problema ben noto: massimizzazione del quoziente di Rayleigh [13].
L'algoritmo di k-Shape, viene riportato di seguito.

Algoritmo 3 k-Shape clustering for Time Series

Input:

Dataset contenente n serie temporali z-normalizzate;
Numero di cluster k da produrre.

Output:

Assegnazione delle sequenze temporali in k cluster.

1: Assegna casualmente le serie temporali ai k cluster;
2: Calcola i centroidi di ciascun cluster;
3: **ripeti**
4: Assengna ciascuna serie temporale ad un cluster usando la distanza SBD;
5: Ricalcola il centroide di ciascun cluster;
6: **finchè** tutti i centroidi si stabilizzano e le times-series non si muovono più tra i cluster o si raggiunge il massimo numero di iterazioni

La misura di distanza SBD [29], Shape-based distance, viene definita come segue:

$$SBD(\vec{x}, \vec{y}) = 1 - \max_{w} \left(\frac{CC_w(\vec{x}, \vec{y})}{\sqrt{R_o(\vec{x}, \vec{x}) \cdot R_o(\vec{y}, \vec{y})}} \right) \tag{2.4}$$

Nella formula 2.4, $CC_w(\vec{x}, \vec{y})$ rappresenta la *cross-correlation sequence*, definita come segue:

$$CC_w(\vec{x}, \vec{y}) = R_{w-m}(\vec{x}, \vec{y}), w \in 1, 2, ..., 2m - 1$$

La misura Shape-based distance, assume valori compresi tra 0 e 2, dove 0 implica una perfetta somiglianza tra le sequenze temporali. Tale distanza, così formulata, garantisce la shift-invariance, mentre per la scaling invariance, è necessario normalizzare ciascuna sequenza temporale, rendendo la sua media µ a zero e la sua deviazione standard equivalente a uno.

Il k-shape funziona, quindi, in modo simile al k-means clustering: entrambi gli algoritmi utilizzano un approccio iterativo per assegnare i dati ai gruppi in base alla distanza tra questi e al centroide del gruppo più vicino. La differenza tra i due metodi risiede proprio nella metrica che viene utilizzata per calcolare le distanze.

L'algoritmo K-Shape ha una dipendenza lineare con il numero di serie temporali, e la sua complessità, in totale, per ogni iterazione è pari a $O(\max\{n \cdot k \cdot m \cdot log(m), n \cdot m^2, k \cdot m^3\})$, dove n è il numero di serie temporali, k è il numero di cluster, m è la lunghezza delle serie temporali [29]. È evidente come la maggior parte del costo di calcolo dipende dalla lunghezza della serie temporale (m). Tuttavia, questa lunghezza è di solito molto più piccola del numero di serie temporali (m « n). Segue,

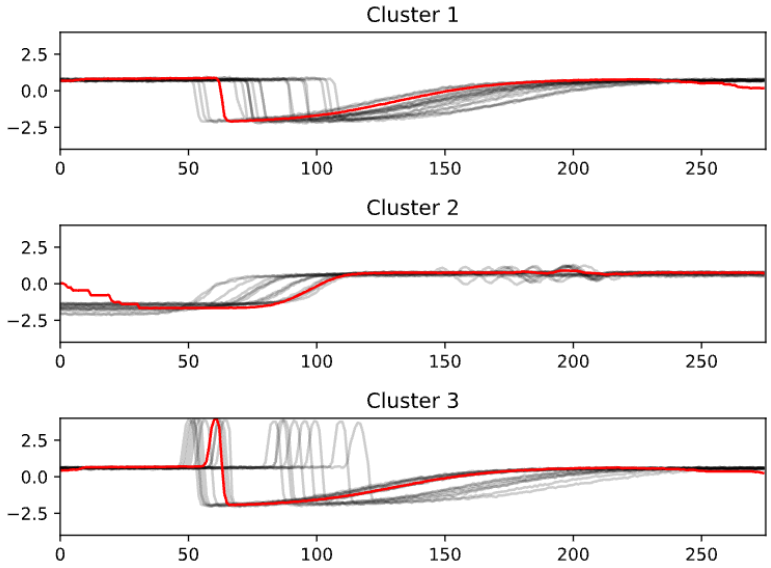


Figura 2.7. Esempio di clustering di serie temporali mediante algoritmo k-Shape. Immagine da [30].

che la dipendenza da m non rappresenta un collo di bottiglia.
Punti di forza: l'algoritmo k-shape è indipendente dal dominio, accurato e scalabile per il clustering di serie temporali, e richiede una regolazione minima dei parametri [27].
Limiti: Si tratta di una procedura abbastanza recente e poco utilizzata in diversi campi di ricerca. Conseguentemente, i limiti di tale procedura non sono stati approfonditi.

2.2 Metodi di rilevazione delle anomalie su serie temporali

L'anomaly detection, o rilevazione di anomalie, copre diversi campi applicativi, dal rilevamento di intrusioni di rete, rilevamento di frodi, nonchè in ambito medico, biologico e ambientale. Nello specifico, l'anomaly detection rappresenta il processo di identificazione di elementi o eventi imprevisti all'interno di un dataset, che risultano inattesi e differiscono dalla norma [31]. In particolare, la definizione di anomalia assume diversi significati a seconda del problema preso in considerazione. Generalmente occorre categorizzare il concetto di anomalia, al fine di semplificarne la comprensione. Di seguito, viene illustrata la categorizzazione generalmente

accettata delle rilevazione delle anomalie, che prende in considerazione la natura dei dati, la tipologia di anomalia, le etichette dei dati e l'output del processo.

1. Natura dei dati: I dati da analizzare possono essere dati sequenziali, dati spaziali, etc. In particolare, i dati delle serie temporali sono dati sequenziali, mentre i dati di rete sono "graph data". Ne consegue che le tecniche di relavamento delle anomalie da utilizzare per ciascun caso possano essere molto diverse tra loro [2].

2. Tipologia di anomalia: vi sono tre tipologie di anomalia (figura 2.8). La prima classe di anomalie rappresenta le *anomalie puntuali*, che cercano singoli elementi differenti dagli elementi considerati "normali". Spesso ci si riferisce all'anomalia puntuale con il termine *outlier*, ossia un valore che si discosta molto dalle osservazioni [32], [33]. Ad esempio, data in input una serie temporale x, un outlier è una coppia tempo-valore $< t, x_t >$, dove il valore osservato x_t è significativamente diverso dal valore atteso della serie temporale al tempo t ($E(x_t)$). Questo tipo di anomalie può essere individuata con classici metodi di rilevamento che sfruttano come metrica la distanza.
   In secondo luogo troviamo le *anomalie constestuali*, in cui il comportamento anomalo viene definito all'interno di un contesto, ossia una istanza di dati non è considerata anomala, a meno che non si verifichi all'interno di un contesto predefinito. Di conseguenza, l'istanza di dati deve possedere caratteristiche ben precise che appartengono al contesto (informazioni temporali, spaziali, etc.) [31], [34].
   Infine, vi sono le *anomalie collettive*, identificate come un gruppo di dati con comportamento anomalo rispetto all'intero set di dati. Nello specifico, non si identificano anomalie di singoli dati, bensì raccolte di dati anomale [32], [35].

3. Etichette di dati: le etichette dei dati sono delle notazioni utili ad indicare se un'istanza di dati è anomala o normale. È possibile effettuare una classificazione delle tecniche di rilevamento delle anomalie in base alla disponibilità delle etichette: (i) *supervised anomaly detection*, dove sia la classe normale che la classe anomala sono etichettate. Solitamente viene costruito un modello predittivo, che permette la classificazione di un dato come appartenente alla classe anomala o normale [36]; (ii) *semi-supervised anomaly detection*, dove i dati di addestramento presentano le etichette solo per le istanze della classe normale e non richiede etichette per la classe anomala [37]; (iii) *unsupervised anomaly detection* utilizzati quando i dati in input sono privi di etichette.

4. Output del processo: è possibile effettuare una distinzione in base al modo in cui le anomalie vengono riportate: (i) *punteggi*, se l'output del processo di rilevamento dell'anomalia è uno score relativo al grado di severità della

potenziale anomalia; (ii) *etichette*, se l'output del processo di rilevamento dell'anomalia deriva da una classificazione binaria.

In questo lavoro di tesi, il problema del rilevamento delle anomalie per lo sviluppo di uno strumento di supporto all'attività di monitoraggio, si basa sull'individuazione delle anomalie all'interno delle metriche di performance di un database. Nello specifico, i dati da analizzare sono delle sequenze temporali e non presentano etichettatura.

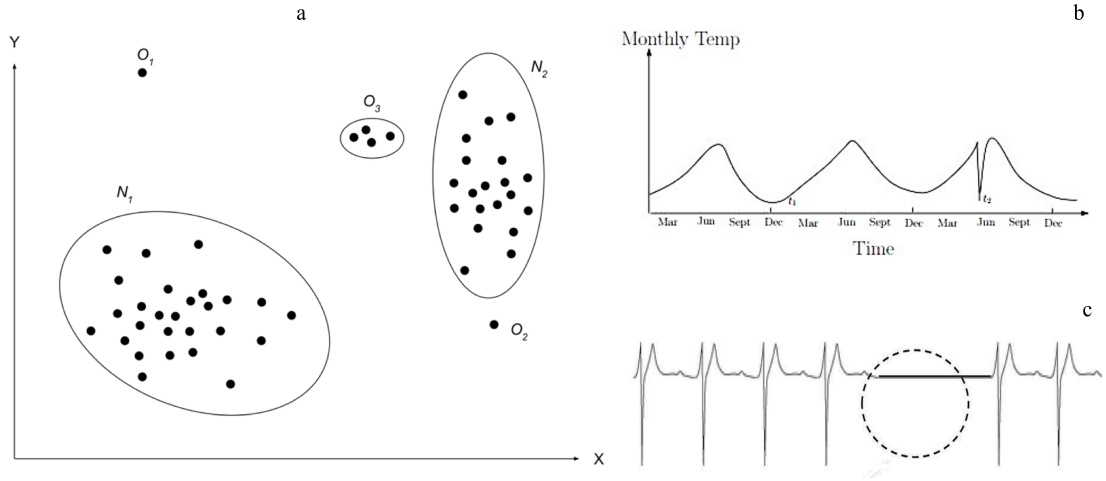


Figura 2.8. Esempi di diverse tipologie di anomalie:
a) *Anomalia puntuale*: rappresentata da O_1, O_2, O_3. Questi punti sono al di fuori dei cluster concentrati N1 e N2.
b) *Anomalia contestuale*: rappresentata dal valore più basso della temperatura registrato nel mese di giugno (June). Tale valore è anormale per il mese.
c) *Anomalia collettive*: rappresentata dal gruppo dati cerchiato in figura. Tale regione del grafico è anomala rispetto al normale pattern del modello.
Immagine da [38]

La serie temporale, o time series, è una sequenza di dati composta da osservazioni registrate in punti contigui equidistanti nel tempo [39]. I valori sono, quindi, campionati in base ad una dimensione temporale, come secondi, minuti, ore, mesi o anni. Di conseguenza, tali osservazioni sono registrate in modo ordinato e presentano correlazioni nel tempo. Esistono diverse tecniche di rilevazione di anomalie che possono essere applicate nel caso di analisi delle serie temporali [40],[41]. Di seguito, vengono illustrate le principali caratteristiche delle più comuni tecniche attualmente utilizzate dagli analisti. In particolare, si espone la scomposizione di

una serie temporale in componenti (sezione 2.2.1), il rilevamento mediante previsione, i.e forecasting (sezione 2.2.2) e la classificazione delle anomalie tramite algoritmo ad albero, Isolation Forest (sezione 2.2.3).

2.2.1 Metodi basati su scomposizione di una serie temporale

Al fine di individuare, se esiste, un *pattern* sistematico nella serie temporale, e di separarlo da eventuali oscillazioni accidentali, spesso è utile scomporre la serie temporale in componenti, nota come *Time series decomposition*. Tramite i metodi di scomposizione è possibile identificare due *pattern* : stagionalità e componente *trend-cycle*. Quest'ultima, di solito, è data dalla combinazione della componente di fondo (trend) e oscillazioni congiunturali (ciclo). Di conseguenza, è possibile immaginare una time series come composta da tre componenti: componente trend-cycle, componente di stagionalità e componente residua. Spesso questo processo permette di migliorare la comprensione delle serie temporali o, in alcuni casi, l'accurazza delle previsioni. In generale, per effettuare una scomposizione della serie temporale si possono seguire due diversi approcci, ovvero un *modello additivo* o *modello moltiplicativo* [42], [43].
Il modello additivo, o additive decomposition, viene definito come segue:

$$y_t = S_t + T_t + R_t \tag{2.5}$$

dove y_t è il dato, S_t è la componente stagionale, T_t è la componente di trend-cycle, R_t è la componente residua, ciascuna al tempo t, con $t = 1, ..., N$ data points misurati. Un modello additivo risulta appropriato quando la fluttuazione stagionale non varia con il livello della serie temporale. Quando l'ampiezza dell'oscillazione stagionale diminuisce o aumenta in modo proporzionale con la diminuzione o aumento del livello della serie, allora il modello più adeguato è quello moltiplicativo. Il modello moltiplicativo può essere descritto dalla seguente equazione:

$$y_t = S_t \times T_t \times R_t \tag{2.6}$$

dove, anche in questo caso, y_t è il dato, S_t è la componente stagionale, T_t è la componente di trend-cycle, R_t è la componente residua, ciascuna al tempo t, con $t = 1, ..., N$ data points misurati. Il modello moltiplicativo trova largo impiego in ambito economico, in quanto diverse serie economiche evidenziano fluttuazioni stagionali che crescono all'aumentare del livello della serie. Un'alternativa, al modello moltiplicativo, consiste nel trasformare i dati fino ad ottenere una variazione nella serie stabile nel tempo, e successivamente procedere con una scomposizione additiva. Tuttavia, questo processo equivale all'uso di un modello moltiplicativo se la funzione di trasformazione usata è la funzione logaritmica [42]. Di fatti, si ha:

$$y_t = S_t \times T_t \times R_t \text{ è equivalente a } \log S_t + \log T_t + \log R_t$$

Quando non è di interesse primario la variazione in una serie temporale dovuta alla stagionalità, è possibile analizzare delle serie destagionalizzate (*seasonally adjusted*). Tali serie contengono esclusivamente la componente residua e la componente trend-cycle. Di conseguenza, i valori "destagionalizzati" sono ottenuti tramite rimozione della componente stagionale dai dati originali [42], [44]. In particolare, nel caso di un modello addittivo il dato destagionalizzato D_t è ottenuto da:

$$D_t = y_t - S_t \tag{2.7}$$

Mentre, nel caso di un modello moltiplicativo, il valore destagionalizzato D_t è dato da:

$$D_t = \frac{y_t}{S_t} \tag{2.8}$$

Le serie destagionalizzate non sono regolari (*smooth*), ma presentano flessioni o ribaltamenti che possono essere fuorvianti, soprattutto nel caso in cui si vogliano individuare ed analizzare i punti di svolta, turning points, in una serie temporale. Tali punti rappresentano i minimi e i massimi relativi di una funzione, ossia le ordinate dei punti di un grafico in cui la funzione passa da descrescente a crescente e viceversa. In particolare, determinano localmente rispettivamente il più piccolo o il più grande valore di ordinata della funzione. Di conseguenza, per individuare in modo corretto e preciso tali punti, è meglio analizzare la componente trend-cycle piuttosto che i dati destagionalizzati.

STL decomposition

Uno dei metodi più robusti e versatili per la decomposizione delle serie temporali è *STL*, *Seasonal and Trend decomposition using Loess*, sviluppato da R. B. Cleveland, Cleveland, Mcrae, & Terpenning (1990) [45]. STL utilizza, quindi, il metodo Loess, Locally Estimated Scatterplot Smoothing, per stimare le componenti della serie temporale.

Loess Data una serie di misurazioni x_i, y_i, con $i = 1, ..., n$, la curva di regressione Loess, $g(x)$, fornisce una stima regolare (*smoothing*) della variabile y, per ciascun valore di x lungo l'asse delle ascisse, ovvero non solo per i valori x_i, per i quali il corrispondente y è stato misurato [46]. Questo, di fatti, rende possibile trattare i valori mancanti ed effettuare un *detrend* della componente stagionale, in modo semplice. Un detrend comporta la rimozione degli effetti della tendenza da un set di dati, al fine di evidenziare esclusivamente le differenze nei valori della tendenza, e permettere l'identificazione di modelli ciclici.
Per calcolare la funzione loess $g(x)$, viene scelto un numero intero positivo q, con

$q \leq n$. In seguito, vengono selezionati q valori x_i, che risultano più vicini a x, e ad ognuno viene assegnato un peso, *neighborhood weight*, in base alla sua distanza da x [45]. Il neighborhood weight, per ciascuna x, è definito come segue:

$$v_i(x) = W\left(\frac{|x_i - x|}{\lambda_q(x)}\right) \tag{2.9}$$

dove $\lambda_q(x)$ è la distanza del q-esimo punto x_i più lontano da x, W è la *tricube weight function*. Dall'equazione 2.9, si evince che più i valori x_i sono vicini a x, più il peso a loro assegnato sarà maggiore, e all'aumentare della distanza, il valore diminuirà fino ad arrivare a zero nel q-esimo punto più lontano x_i da x.
Nello step successivo, si adatta un polinomio di grado d, tipicamente pari a uno o due, ai dati (x_i, y_i) con peso $v_i(x)$, ottenendo $g(x)$. Talvòlta, ciascun dato può essere ulteriormente pesato, tramite dei robustness weights, ρ_i. In tal caso, i robustness weights vengono incorporati al neighborhood weight, ottenendo un peso definito da $\rho_i v_i$ [45], [46].
Nella decomposizione STL, risulta fondamentale effettuare un'attenta e corretta selezione dei parametri di smoothing q e d. Di fatti, all'aumentare della variabile q, la funzione $g(x)$ diventa più regolare (smooth).

Algoritmo L'STL decomposition ha come obiettivo quello di separare una serie temporale y_t nelle sue componenti: trend-cycle, stagionalità, residuo, secondo l'equazione 2.5. Questo è reso possibile attraverso due procedure ricorsive: un ciclo interno annidato in un ciclo più esterno [45], [46]. Nel ciclo esterno, viene definito un peso (robustness weights) per ciascun punto temporale, a seconda della dimensione della componente residua. In tal modo, si riducono o vengono eliminati gli effetti dovuti alla presenza dei valori anomali. Di fatti, un outlier presenterà un valore della componente residua molto grande, e di conseguenza ad esso verrà assegnato un peso piccolo o pari a zero.
Nel ciclo interno, il trend e la componente stagionale vengono aggiornati in modo iterativo. Inizialmente, viene sottratta la stima corrente del trend dalla raw series. In seguito, si identificano delle cycle-subseries, tramite suddivisione delle serie temporali. Ad esempio, se si analizzano dei dati mensili in diversi anni, questi verranno suddivisi in 12 cycle-subseries, in modo tale che tutti i mesi di Gennaio si trovino in un'unica time series, tutti i mesi di Febbraio in una seconda, e così via. Ciascuna cycle-subseries, viene sottoposta al processo di *smoothing* tramite Loess, e successivamente attraversa un filtro passa-basso. I row data vengono privati della componente stagionale, precedentemente calcolata, mediamente processo di *deseasonalizing*. In seguito, si effettua il processo di trend smoothing, ossia la serie deseasonalizzata subisce un processo di smoothing tramite loess, ottenendo la componente trend. Di conseguenza, la componente residua viene calcolata a

partire dalla time series originale, mediante rimozione della componente stagionale e trend, precedentemente definite. Di seguito, viene riportato l'algoritmo di decomposizione STL [46].

Algoritmo 4 STL decomposition

Input:

Serie temporale Y_v;

Output:

Suddivisione della serie temporale Y_v in componente trend-cycle T_v, componente di stagionalità S_v, componente residua R_v.

1: Inizializza la componente trend-cycle come $T_v^{(0)} = 0$ e la componente residua $R_v^{(0)} = 0$;
2: **Outer loop - calcola i robustness weights. Esegui $n_{(o)}$ volte**
3: Calcola R_v;
4: Calcola i robustness weights $\rho_v = B(|R_v|/h)$, dove B è la funzione bi-square weight, ed $h = 6$ median $(|R_v|)$. Per il primo loop $\rho_v = 1$;
5: **Inner loop - calcola iterativamente componente trend e componente stagionale. Esegui $n_{(i)}$ volte**
6: *Detrend*: calcola $Y_v - T_v^{(k)}$, dove k è il numero del ciclo;
7: *Cycle-subseries smoothing*: suddividi le serie temporali prive della componente trend in $n_{(p)}$ cycle-subseries. Sottoponi ciascuna sottoserie a un processo di smoothing tramite Loess, con $q = n_{(s)}, d = 1$. I valori smoothed danno origine ad una serie temporale stagionale e temporanea $C^{(k+1)}$;
8: *Low-pass filter*: applica un filtro passa-basso su $C^{(k+1)}$, ottenendo $L^{(k+1)}$;
9: *Detrending of smoothed cycle-subseries*: calcola la (k+1)-esima stima della componente stagionale, come:$S^{(k+1)} = C^{(k+1)} - L^{(k+1)}$;
10: *Deseasonalizing*: rimuovi la stima della componente di stagionalità, quindi esegui $Y_v - S^{(k+1)}$;
11: *Trend smoothing*: sulla serie destagionalizzata ottenuta al passo precedente, effettua uno smoothing con Loess, settando i parametri q=$n_{(t)}$ e d=1, ottenendo $T^{(k+1)}$, ovvero la k+1-esima stima della componente trend.

Nel modello sono presenti diversi parametri fondamentali: (i) $n_{(p)}$ rappresenta il periodo della stagionalità; (ii) $n_{(p)}$ numero di cicli del loop interno, tipicamente pari a 1 o 2; (iii) $n_{(o)}$ numero di cicli del loop esterno. Tipicamente un numero maggiore riduce in modo più efficace l'effetto dei valori anomali; (iv) $n_{(s)}$ parametro di smoothing per il filtro stagionale; (v) $n_{(t)}$ parametro di smoothing per il comportamento del trend.

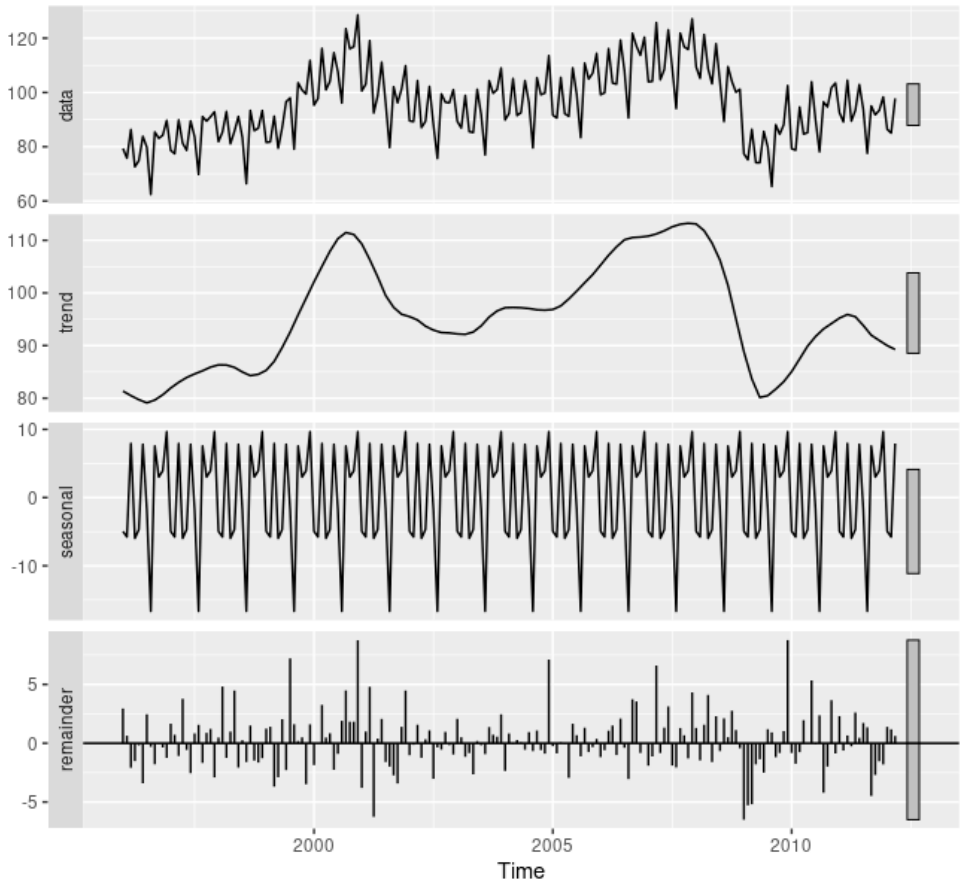


Figura 2.9. Esempio di decomposizione STL del segnale nelle sue componenti: trend, stagionalità (seasonal) e residuo (remainder). Immagine da [47].

Anomaly detection tramite STL Scomponendo la serie temporale nelle sue principali componenti, mediante STL decomposition, come in figura 2.9, analizzando la componente residua e introducendo un valore di soglia, è possibile identificare i valori anomali nella time-series. Nello specifico, se il valore assoluto della componente residua, per un punto X_t, è maggiore della soglia stabilita, si può definire X_t come un'anomalia di punto, o outlier [48], [49].
Punti di forza: Tale metodo per la rilevazione di anomalie è molto semplice, facile da realizzare in quanto i soli dati necessari per effettuare l'analisi sono i residui della decomposizione, robusto, può gestire diverse situazioni, e l'interpretazione delle anomalie è intuitiva.
Limiti: Questo metodo è caratterizzato da una forte rigidità, dettato dalle esigue

scelte che si possono operare. Nello specifico, oltre a definire correttamente il valore di soglia e l'intervallo di confidenza, non sono realizzabili ulteriori modifiche, utili a migliorare le performance. Inoltre, la scelta del parametro soglia non è semplice e richiede un'analisi puntuale.

La decomposizione delle serie temporali, inoltre, può essere un step utile nella produzione di previsioni, ed in particolar modo, nella rilevazione delle anomalie mediante l'utilizzo di algoritmi di forecasting, discussi nella sezione 2.2.2, come evidenziato in [50].

2.2.2 Metodi basati su modelli di previsione

Il processo di Forcasting può essere utilizzato come ulteriore metodo per la rilevazione di anomalie. Esso si basa sulla stima di una previsione futura di punti di dato, in relazione alla valutazione di punti nel passato. Si prevede anche la possibile presenza di un rumore bianco, o più in generale di una variabile casuale. Vi sono diverse metodologie di previsione basate su medie mobili (*Moving Averages*), approccio autoregressivo (*Autoregressive approach*), ARIMA (*Auto Regressive Integrated Moving Average*) con le sue diverse varianti, etc.
Di seguito, vengono fornite le principali nozioni relative ai modelli ARIMA, e come gli stessi possono essere utilizzati al fine di generare un modello di rilevazione di anomalie.

ARIMA

I modelli ARIMA, *Auto Regressive Integrated Moving Average*, rappresentano uno degli approcci maggiormente utilizzati per la previsione delle serie temporali. In particolare, tali modelli mirano a descrivere le autocorrelazioni nei dati.
Per poter comprendere i modelli ARIMA, è necessaria una breve introduzione riguardo due concetti fondamentali: stazionarietà (stationarity) e differenziazione delle serie temporali (differencing).

Stationarity. L'obiettivo dell'analisi delle serie temporali è l'individuazione di un opportuno processo stocastico avente traiettorie che si adattano ai dati, al fine di formulare, in seguito, previsioni. In particolare, un processo stocastico viene definito stazionario se la sua struttura probabilistica non varia nel tempo [51]. In particolare, un processo aleatorio X_t si definisce stazionario se:

1. $E(X_t) = \mu$;
2. $Var(X_t) = \sigma^2$, per ogni t;
3. $Cov(X_t, X_t + h) = Cov(X_s, X_s + h) = \gamma(h)$.

Di conseguenza, si evince che la media e la varianza sono invarianti nel tempo e le covarianze tra le due variabili del processo coinvolte sono determinate in funzione della distanza nel tempo, h. Il rumore bianco è un tipico esempio di serie temporale stazionaria. Di fatti, in qualsiasi momento essa venga osservata presenta pressochè lo stesso andamento.
La presenza o assenza di stazionarità in una serie temporale può essere verificata mediante l'utilizzo di un test statistico comune, noto come ***Augmented Dickey Fuller test*** (ADF test). Si tratta di un test statistico di significatività, ovvero inizialmente si assume una ipotesi nulla, si procede calcolando un test statistico, il cui risultato va confrontato con un valore critico tabulato in apposite tabelle. Se il risultato del test supera il valore critico, allora l'ipotesi nulla viene respinta e viene accettata l'ipotesi alternativa, in caso contrario l'ipotesi nulla viene accettata. Nello specifico, l'ipotesi nulla nell'ADF afferma la presenza di una *unit root* in un campione di serie temporali. L'ipotesi alternativa, invece, afferma la presenza di stazionarietà [52]. Una *unit root* è una tendenza stocastica di una serie temporale, che evidenza un modello sistematico non prevedibile. Di fatti, la serie temporale risulta non stazionaria. La procedura di testing per il test ADF viene applicata al modello:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \cdots + \delta_{p-1} \Delta y_{t-p+1} + \varepsilon_t, \tag{2.10}$$

dove α è una costante, β il coefficiente su una tendenza temporale, e p l'ordine di latenza del processo autoregressivo. Lo unit root test viene effettuato sotto l'ipotesi nulla $\gamma = 1$, che implica la presenza di una unit root nella serie. Se l'ipotesi nulla non viene rifiutata, la serie temporale in considerazione non è stazionaria. Di conseguenza, per poter respingere l'ipotesi nulla, ed affermare che la serie temporale è stazionaria, è necessario ottenere un *p-value* inferiore al livello di significatività definito (solitamente pari a 0.05).

Differencing. Con il termine *differencing*, o differenziazione, si indica il processo attraverso il quale una serie temporale viene resa stazionaria. In particolare, la serie temporale di partenza viene trasformata in una nuova serie, i cui valori sono calcolati come differenza tra osservazioni consecutive [53]. Tale procedura può essere applicata sequenzialmente dando luogo alle "first differences", "second differences", etc. Quando la serie temporale presenta una forte asimmetria, è possibile applicare una trasformazione logaritmica, al fine di ottenere una distribuzione normale. Di fatti, una trasformazione logaritmica consente di stabilizzare la varianza di una serie temporale e di ridurre ad effetti additivi un effetto moltiplicativo (come discusso nella sezione 2.2.1). La differenziazione ha, quindi, come obiettivo quello di associare una varianza ed una media stabili nel tempo ad una serie temporale. Tale processo può essere effettuato tramite la rimozione dei cambiamenti nel livello di una serie temporale, e, conseguentemente, l'eliminazione o riduzione

della componente trend o stagionalità. Infatti, la tendenza e la stagionalità possono determinare, in una serie temporale, rispettivamente, una media variabile nel tempo ed una varianza non costante nel tempo, rendendo la serie temporale non stazionaria.

First-order differencing. Nella first-order differencing, o lag-1 difference, la *differenced series* si ottiene mediante differenza tra l'osservazione attuale e l'osservazione precedente, nella serie originale:

$$y_t^{'} = y_t - y_{t-1} \tag{2.11}$$

Dall'equazione 2.11, si evince che la differenced series avrà esclusivamente t-1 valori, in quanto per la prima osservazione non è possibile ricavare una differenza.

Second-order differencing. Alcune serie temporali, in seguito ad un'operazione di differenziamento, risultano ancora non stazionarie. In tal caso, può essere necessario applicare nuovamente il processo di *differencing*, con lo scopo di ottenere una serie stazionaria. Questa metodologia viene definita second-order differencing, e viene definita come:

$$y_t^{''} = y_t^{'} - y_{t-1}^{'} = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) = y_t - 2y_{t-1} + y_{t-2} \tag{2.12}$$

Nell'equazione 2.12, si può notare che $y_t^{''}$ possiede t-2 valori. Inoltre, è evidente come il processo di differenziazione può essere iterabile fino a quando la serie temporale risulta invariabile nel tempo. In particolare, si definisce *order of differencing* il numero di esecuzioni del processo di differencing. Tale valore rappresenta uno dei parametri del modello ARIMA, da impostare in modo ottimale, ed è rappresentato dal termine "d".

Seasonal differencing. La seasonal differencing è definita come la differenza tra un'osservazione corrente e un'osservazione precedente, nella medesima stagione (season):

$$y_t^{'} = y_t - y_{t-m} \tag{2.13}$$

dove m è il numero di stagioni.

Il modello ARIMA è una generalizzazione di un modello autoregressivo a media mobile. In particolare, è ottenuto dalla combinazione di un modello AR (*AutoRegressive model*) ed uno MA (*Moving Average model*), ed un processo di differenziazione (*Differencing*).

In un *modello autoregressivo* la variabile di interesse viene prevista mediante combinazione linerare dei valori che la variabile ha assunto nel passato, al fine di tener

conto di una possibile dipendenza statistica tra osservazioni rilevate in istanti differenti [54]. Conseguentemente, un modello autoregressivo di ordine p, può essere scritto come:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t \tag{2.14}$$

dove i parametri ϕ_1, ϕ_2, .., ϕ_p costituiscono i coefficienti della regressione lineare della variabile y_t, rispetto ai suoi stessi valori passati, mentre ϵ_t è il termine di errore che descrive un rumore bianco, ovvero una sequenza di variabili casuali incorrelate, con media pari a zero e varianza costante. Il modello autoregressivo spesso viene indicato come AR(p), con p ordine del modello. Generalmente, l'uso di questi modelli è limitato all'analisi di dati stazionari. In tal caso si necessita l'introduzione di vincoli sui valori che possono assumere i parametri. In particolare, nei modelli AR(1) è necessario che $-1 < \phi_1 < 1$, mentre nei modelli AR(2) si deve rispettare la condizione: $-1 < \phi_2 < 1, \phi_1 + \phi_2 < 1, \phi_2 - \phi_1 < 1$.

Il *modello a media mobile*, MA, sfrutta gli errori di previsione del passato in un modello simile alla regressione, definito come:

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} \tag{2.15}$$

dove i parametri $\theta_1, \theta_2, ..., \theta_1$ sono delle costanti, mentre ϵ_t rappresenta il termine di errore usato come regressore nelle q unità temporali prese in considerazione [55]. Nell'equazione 2.15, si può notare come ciascun valore y_t può essere considerato come una media mobile di q termini passati che identificano gli ultimi errori di previsione. Il modello a media mobile, spesso, viene indicato come MA(q), con q ordine del modello e viene chiamato *invertibile*, se rispetta le seguenti condizioni:
(i) $-1 < \theta_1 < 1$, in un modello MA(1)
(ii) $-1 < \theta_2 < 1, \theta_1 + \theta_2 > -1, \theta_1 - \theta_2 < 1$, in un modello MA(2).
Un modello invertibile presenta delle ottime proprietà matematiche e consente la conversione di un modello MA a un modello AR.

Il modello ARIMA, a differenza dei modelli precedentemente descritti, può essere applicato anche in situazioni in cui le serie temporali non sono stazionarie, ovvero evidenziano una media variabile nel tempo. In particolare, durante la fase iniziale del modello, la serie temporale viene resa stazionaria mediante differenziazione, che può essere applicata ripetutamente fino alla completa eliminazione della componente di tendenza, che, per l'appunto, causa l'effetto di non stazionarità . Questo procedimento definisce, quindi, *la parte integrata del modello*, e il significato associato al termine *I*, nell'acronimo del nome del modello [56].

In conclusione, un modello *ARIMA non-seasonal*, può essere definito come:

$$y_t' = c + \phi_1 y_{t-1}' + ... + \phi_p y_{t-p}' + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} + \epsilon_t \tag{2.16}$$

dove y_t' rappresenta la serie differenziata. I predittori, nella parte destra, presentano sia i *lagged values* di y_t, sia i *lagger errors*. $\phi_1, ..., \phi_p$, rappresentano, come visto in precedenza, i parametri della parte autoregressiva del modello, mentre $\theta_1, ..., \theta_q$ sono i parametri della parte di modello a media mobile. Infine, ϵ_t è il termine di errore, o rumore bianco [57].
Nel modello ARIMA, vengono applicate le stesse condizioni relative a stazionarietà e invertibilità, viste in precedenza nei modelli autoregressivi e a media mobile. Generalmente, questo modello viene definito tramite la notazione: ARIMA(p, d, q), dove p è l'ordine della parte autoregressiva del modello, d è il grado di differenziazione, e q è l'ordine della parte di modello a media mobile.
In particolare, tra i modelli ARIMA, vi sono alcuni casi speciali ben noti, come:

- il modello *random walk*, definito come $y_t = y_{t-1} + \epsilon_t$, è equivalente ad un modello ARIMA(0,1,0);
- il modello *random walk with drift*, definito come $y_t = c + y_{t-1} + \epsilon_t$, è equivalente ad un modello ARIMA(0,1,0) con costante;
- il modello di autoregressione è equivalente ad un modello ARIMA(p,0,0);
- il modello a media mobile è equivalente ad un modello ARIMA(0,0,q);
- il rumore bianco è equivalente ad un modello ARIMA(0,0,0).

Alla luce di quanto è stato precedentemente descritto, è necessario stimare in modo corretto tutti i parametri del modello ARIMA. Solo in questo modo sarà possibile effettuare un forecasting capace di fornire delle stime quanto più attendibili e precise possibili. Attualmente, esistono delle metodologie che possono essere utilizzate per effettuare un apprendimento automatico dei migliori parametri da applicare ad un modello ARIMA, sulla base dei propri dati di partenza. Un esempio, è rappresentato dal modello *Auto ARIMA*, del package Python pmdarima.arima [58]. Tale metodo consiste nell'applicazione di vari modelli ARIMA, ciascuno dei quali caratterizzato da una diversa combinazione dei parametri. In seguito, per ciascun modello vengono calcolati i valori dei parametri BIC, Bayesian Information Criterion, e AIC, Akaike information criterion. Il modello che assume i valori più bassi di BIC e AIC, viene selezionato, determinando la lista più adeguata dei parametri del modello. In particolare, l'*Akaike information criterion* può essere scritto come:

$$AIC = -2\log(L) + 2(p + q + k + 1) \tag{2.17}$$

dove L è la probabilità dei dati (*likelihood*), p è l'ordine della parte autoregressiva, q è l'ordine della parte media mobile, e k rappresenta l'intercetta del modello ARIMA. Inoltre, se $c \neq 0$ allora $k = 1$, mentre se $c = 0$ allora $k = 0$ [59].

Tuttavia, la forma più corretta di AIC, adatta per i modelli ARIMA, viene riscritta come segue:

$$AIC_c = AIC + \frac{2(p+q+k+1)(p+q+k+2)}{T-p-q-k-2} \tag{2.18}$$

Invece, il *Bayesian Information Criterion* è definito come:

$$BIC = AIC + (\log(T) - 2)(p+q+k+1) \tag{2.19}$$

L'obiettivo è, quindi, minimizzare i valori di AIC, AIC_c o BIC, per definire un ottimo modello ARIMA. Tuttavia, questi criteri di informazione tendono ad essere una buona guida per la corretta selezione dei valori di p e di q, ma non di d, ovvero l'ordine di differenziazione di un modello. Ciò avviene in quanto il processo di *differencing* modifica i dati su cui viene calcolata la probabilità, rendendo impossibile un confronto tra i valori AIC di modelli con ordini di differenziazione diversi. Conseguentemente, si è vincolati ad utilizzare un altro approccio per scegliere correttamente il parametro d, e successivamente usare l'AIC_c per selezionare i valori dei parametri p e q [58].
Fino a questo momento, sono stati trattati esclusivamente i modelli ARIMA non stagionali. Tuttavia, esistono diverse varianti dei modelli ARIMA. In particolare, tali modelli sono in grado di adattarsi anche ad un'ampia gamma di dati stagionali.

Un modello ARIMA stagionale, noto con il termine di SARIMA (Seasonal ARIMA), include nel semplice modello ARIMA, precedentemente analizzato, dei termini stagionali aggiuntivi. Viene definito come:

$$\text{ARIMA } (p,d,q)(P,D,Q)_m$$

dove m equivale al periodo stagionale, i termini (p,d,q) rappresentano la parte non stagionale del modello, mentre i termini (P,D,Q) sono i termini aggiuntivi, che identificano la parte stagionale del modello [60]. È evidente, come i parametri delle due diverse componenti, stagionale e non, risultino molto simili tra di loro. Tuttavia, i termini introdotti con il metodo stagionale, coinvolgono i backshifts del periodo stagionale. In particolare, un'operatore di backshifts su dei dati y_t produce come effetto lo spostamento dei dati indietro di un periodo. e spesso viene definito come: $B_{y_t} = y_{t-1}$.

Forecasting with decomposition

La decomposizione delle serie temporali (discussa nella sezione 2.2.1) può essere utilizzata come processo preliminare al fine di migliorare la rilevazione delle

anomalie eseguita con modelli di forecasting. In particolare, una decomposizione additiva di una serie temporale (equazione 2.5), può essere riscritta come:

$$y_t = S_t + A_t \tag{2.20}$$

dove $A_t = T_t + R_t$ rappresenta la componente destagionalizzata. In presenza, invece, di una decomposizione moltiplicativa, l'equazione 2.6 come:

$$y_t = S_t \times A_t \tag{2.21}$$

dove $A_t = T_t \times R_t$. Al fine di effettuare un forecasting di una serie temporale decomposta, bisogna prevedere la componente stagionale S_t separatamente rispetto alla componente destagionalizzata A_t. Nello specifico, si presume che la componente stagionale sia immutabile, o che vari molto lentamente nel tempo. Di conseguenza, viene utilizzato un metodo di forecasting semplice e basilare, generalmente noto come *naïve method*. Tale metodo consiste nell'assegnare alle previsioni il valore dell'ultima osservazione, ovvero: $y_{T+h|T} = y_T$ [61]. Parallelamente, per la previsione della componente destagionalizzata, è possibile utilizzare un qualsiasi metodo di previsione non stagionale, come ad esempio i modelli non-seasonal di ARIMA, discussi nella sezione 2.2.2.

2.2.3 Isolation Forest

L'Isolation Forest [62], o Iforest, è uno degli algoritmi maggiormente utilizzati in ambito anomaly detection. Si tratta di un algoritmo non supervisionato, il cui principio di funzionamento si basa esclusivamente sull'identificazione esplicita delle anomalie. Di fatti, l'Isolation Forest differisce dai comuni metodi di rilevamento, in quanto non si focalizza sulla caratterizzazione del comportamento normale di una serie di punti di dato. L'Isolation Forest è in grado di rilevare le anomalie esclusivamente basandosi sul fatto che esse sono presenti in numero ridotto e differiscono rispetto agli altri punti di dato, ossia assumono valori molto diversi rispetto alle istanze "normali". Nello specifico, tale algoritmo, per isolare le anomalie, utilizza degli alberi di decisione binari, noti come Itree, non richiedendo, quindi, alcuna misura di distanza o di densità. La procedura di isolamento rappresenta l'elemento chiave dell'Isolation Forest. Essa consiste in una metodologia iterativa atta a definire un insieme di regole, le quali permettono una suddivisione dello spazio delle features, in un sottospazio caratterizzato da un solo elemento [63]. Questo è possibile perchè gli outlier, nello spazio delle features, corrispondono a dei punti che possono essere più facilmente isolati rispetto agli inliers, i quali identificano il comporamento "normale". In particolare, il rilevamento delle anomalie mediante Iforest, consiste in una procedura caratterizzata da due fasi principali. Durante la prima fase, i dati di training vengono utilizzati per costruire

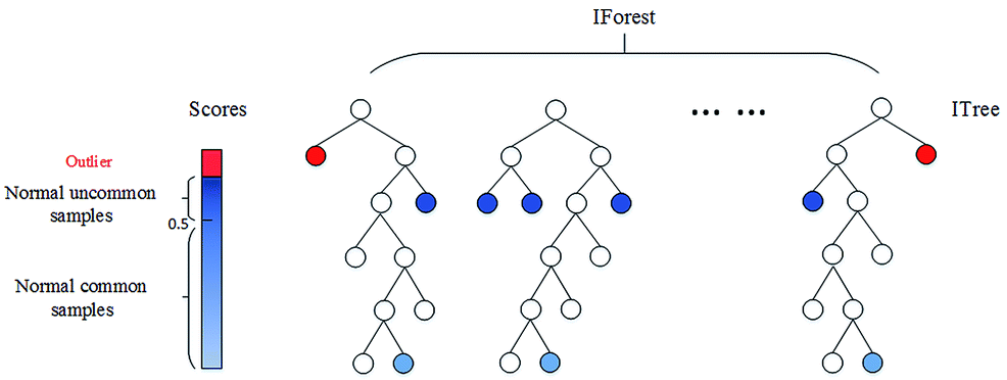


Figura 2.10. Rilevazione delle anomalie mediante iForest. In figura, i punti rossi nell'Itree rappresentano le anomalie nei dati, i punti blu rappresentano i punti normali non comuni, infine i punti azzurro chiaro, rappresentano i punti normali comuni. Tale suddivisione è stata effettuata mediante analisi dell'anomaly score delle osservazioni. Immagine da [64].

gli alberi di isolamento (Itree). Nella seconda fase, invece, ciascun dato nel set di testing passa attraverso gli alberi di isolamento, precedentemente creati, ed ad ognuno viene assegnato un punteggio di anomalia, noto come *anomaly score*. A seconda dei valori che assume l'*anomaly score*, permette di identificare, in modo esplicito, le anomalie. Se il punteggio associato ad un punto di dato è superiore ad una soglia predefinita, viene contrassegnato come punto anomalo.
In ciascun Itree, si seleziona casualmente un attributo ed un valore di divisione, o soglia, per l'attributo selezionato tra i valori minimi e massimi che esso può assumere, in modo tale da generare delle partizioni dei dati. Una struttura ad albero permette, quindi, di rappresentare un partizionamento ricorsivo. Conseguentemente, il numero di partizioni necessarie ad isolare un punto equivale alla lunghezza del percorso dal nodo radice, *root node*, al nodo terminale [63]. Generalmente, le anomalie si individuano in nodi più vicini al nodo radice dell'albero, comportando un percorso più breve, in quanto più facilmente isolabili rispetto agli altri punti. Un punto non anomalo, invece, è più difficile da isolare, richiede più divisioni, e spesso si trova all'estremità più profonda dell'albero (figura 2.10).
L'Isolation Forest è costituito da un insieme di isolation trees, precedentemente descritti, e le anomalie che individua, sono le istanze del dataset che presentano brevi lunghezze medie di percorso nell'isolation tree.
Di seguito, vengono riportati in modo preciso gli step eseguiti dall'algoritmo Isolation Forest.

Algoritmo 5 Isolation Forest

Input:

Set di dati di N caratteristiche;
M: numero di alberi da costruire;

Output:

Identificazione delle anomalie nel set di dati.

1: Seleziona ed assegna un sottocampione casuale dei dati ad un albero binario;
2: **ripeti**
3: **ripeti**
4: Seleziona in modo casuale una caratteristica (feature) del set di dati ed il valore di soglia ad essa associato (un qualsiasi valore presente nell'intervallo di valori minimi e massimi della caratteristica selezionata).
5: **se** il valore di un punto di dato è inferiore alla soglia selezionata **allora**
6: va verso il ramo sinistro dell'albero;
7: **altrimenti**
8: va verso il ramo destro dell'albero;
9: **finchè** ciascun punto di dato è completamente isolato o si raggiunge la massima profondità (se definita).
10: **finchè** M alberi binari casuali, isolation trees, sono stati costruiti, completando la fase di training del modello.
11: Durante la fase di testing, ciascun punto di dato attraversa gli alberi addestrati in precedenza.
12: Assegna un punteggio, *anomaly score*, a ciascun punto di dato in base alla profondità dell'albero necessaria per arrivare a quel punto. Questo punteggio, corrisponde ad una aggregazione della profondità ottenuta da ciascun isolation tree.
13: **se** l'*anomaly score* di un punto di dato è circa 1 **allora**
14: si tratta di un punto anomalo;
15: **altrimenti**
16: se il valore dell'*anomaly score* è inferiore a 0.5, è un punto normale;

Anomaly score. L'*anomaly score* per un punto x può essere definito come:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{2.22}$$

dove h(x) è la lunghezza del percorso dal nodo radice al nodo esterno x, E(h(x)) è la media di h(x), mentre c(n) è la media di h(x) dato il numero di nodi esterni n, e viene usata per normalizzare h(x) [65]. Mediante l'utilizzo dell'*anomaly score* s, sono possibili le seguenti valutazioni:
(i) Se il punteggio per una osservazione è vicino a 1, allora la lunghezza del percorso è molto piccola, ed il punto è facilmente isolabile. Conseguentemente, si tratta

di un punto anomalo; (ii) Un punteggio, associato ad una osservazione, inferiore a 0,5 implica, invece, una lunghezza del percorso grande. In tal caso, si tratta di un punto di dato normale; (iii) Se a tutte le osservazioni è associato un punteggio pari a 0,5, allora l'intero set di dati non presenta alcuna anomalia.

Punti di forza: L'algoritmo Isolation Forest è caratterizzato una ridotta e lineare complessità temporale e richiede un esiguo quantitativo di memoria [65].
Limiti: Un numero elevato di features può inficiare sulle prestazioni computazionali del modello. In tal caso, quindi, è opportuno applicare una selezione accurata delle features [49]. Inoltre, tra i parametri presenti in questo metodo vi sono: il numero di alberi da costruire, la dimensione del sottocampionamento e la contaminazione, ossia la proporzione di valori anomali nel set di dati. Quest'ultimo è un parametro importante e difficile da impostare, che inficia sull'efficacia del modello. Di fatti, si deve scegliere un valore opportuno di contaminazione, per evitare il non rilevamento delle reali anomalie (se, ad esempio, il valore di contaminazione impostato è troppo basso), o la segnalazione di false anomalie, che corrispondono quindi a punti normali (se, ad esempio, il valore di contaminazione impostato è troppo alto) .

Capitolo 3

Tecnologie utilizzate

In questo capitolo sono riportate le principali nozioni teoriche relative alle tecnologie utilizzate durante il progetto di tesi. L'obiettivo è spiegare le caratteristiche intriseche di ciascuna tecnologia adottata e come le stesse sono state integrate per la realizzazione del progetto finale.

3.1 Confluent Platform

Confluent Platform è una piattaforma di streaming che consente di accedere, archiviare ed organizzare flussi di dati in real-time , provenienti da fonti diverse, in un unico sistema centrale affidabile e ad elevate prestazioni [66]. Il nucleo di Confluent Platform è *Apache Kafka* (figura 3.1), piattaforma open source per il data streaming [67]. Confluent Platform espande i vantaggi della piattaforma Apache Kafka, rimuovendo l'onere della gestione dei meccanismi sottostanti, come ad esempio il processo di trasporto dati e l'integrazione tra i diversi sistemi.
Nello specifico, Confluent semplifica :

- le connessioni con i data source, mettendo a disposizione diversi connettori Kafka precostruiti;
- la creazione di real-time applications, mediante l'utilizzo di *ksqlDB*;
- il monitoraggio e gestione dell'infrastruttura Kafka, mediante tool di monitoring e managment, tra cui *Control Center*.

Confluent Control Center è un sistema che utilizza una semplice interfaccia grafica per gestire facilmente Kafka Connect, per controllare i flussi di dati dal produttore al consumatore (assicurando che ciascun messaggio venga consegnato), per ottenere una rapida panoramica dell'integrità del cluster e per sviluppare o eseguire query ksqlDB [66].

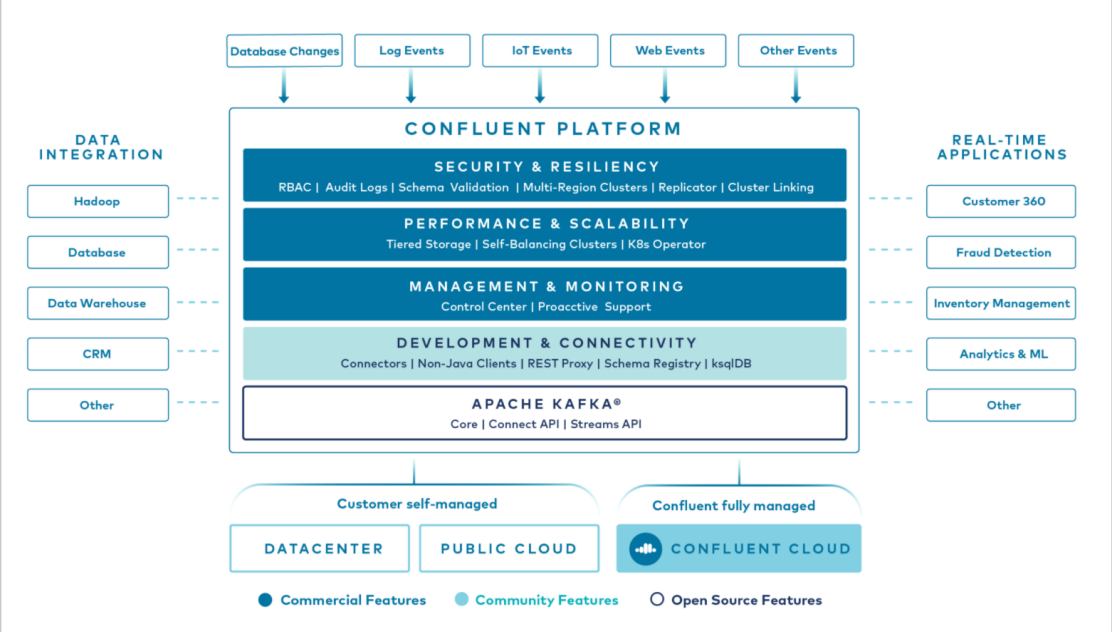


Figura 3.1. Confluent Platform Components. Immagine da [66].

Kafka Connect è uno dei componenti open source di Apache Kafka [68]. Semplifica la scrittura di connettori di alta qualità, i quali si dividono in due diverse categorie:

- *Source connector*: raccoglie i dati da uno o più data source, come database e streams tables e li inserisce all'interno di *Kafka topics*.
- *Sink connector*: preleva i dati dai *Kafka topics* e li trasmette a sistemi secondari per l'analisi offline.

Confluent Schema Registry è un ulteriore componente di Confluent Platform. Fornisce un'interfaccia RESTful per l'archiviazione e il recupero di schemi Avro e JSON. Tale componente memorizza tutte le versioni degli schemi utilizzati per ciascun Kafka topic e consente l'evoluzione degli stessi in base alle impostazioni di compatibilità definite dall'utente [69]. Di conseguenza, gli sviluppatori possono modificare in sicurezza gli schemi secondo le necessità.

3.2 InfluxDB

InfluxDB è un Time Series Database open source, ed è stato sviluppato dall'azienda InfluxData [70]. Nella prima versione, si basava su un linguaggio di interrogazione chiamato InfluxQL, simile a SQL, successivamente sostituito da Flux.
E' stato ideato per facilitare la gestione di database che archiviano serie temporali. In questi database, generalmente, le misurazioni vengono solo aggiunte, e raramente aggiornate o rimosse. All'interno di tali database, infatti, possono confluire milioni di record di dati, che creano un flusso di dati costante. Una volta confluiti nel database, questi dati devono essere elaborati in tempi relativamente rapidi [71]. Un'altra caratteristica di un TSDB (Time Series Database) è la possibilità di filtrare le misurazioni utilizzando tag. Ciascun dato, infatti, è etichettato con un tag che aggiunge informazioni di contesto, ad esempio dove è stata presa la misura [70]. InfluxDB può essere utilizzato per l'elaborazione e l'analisi di dati provenienti da Confluent Kafka. Quest'ultimo, infatti, fornisce un sink connector per trasferire i dati da un Kafka topic in tabelle InfluxDB. Inoltre, per garantire una migliore visualizzazione dei dati mediante l'utilizzo di dashboard e grafici, è possibile collegare InfluxDB a Grafana, tool per la data visualization. La connessione di queste diverse tecnologie consente la creazione di dashboard real time utili per monitorare e visualizzare statistiche, vendite, processi, etc.

3.3 Python libraries

Python è un linguaggio di programmazione orientato agli oggetti, alquanto intuitivo e di facile utilizzo. Tale caratteristica, di fatti, lo rende uno dei linguaggi di programmazione maggiormenti utilizzati dai programmatori o analisti [72]. Python è adatto allo sviluppo di applicazioni distribute e all'elaborazione di script utili per l'analisi di dati. Inoltre, è multipiattaforma, ovvero lo si esegue sui principali sistemi operativi senza alcun problema. Tuttavia, il principale vantaggio associato al linguaggio Python è relativo alla vasta disponibilità di librerie e framework, open-source, che forniscono un supporto efficace in diversi ambiti, dal machine learning, allo sviluppo di applicazioni web. In particolare, tali librerie sono risultate valide nello sviluppo del lavoro di tesi, in quanto hanno permesso lo svolgimento di attività come l'accesso ai database, operazioni di scrittura e lettura di file, manipolazione dei dati, elaborazione di report, visualizzazione grafica dei dati , l'applicazione di algoritmi di apprendimento automatico, etc.
Per poter organizzare in un unico documento lo script Python che effettua una prima analisi esplorativa dei dati, seguita da una selezione delle features e dall'esecuzione di algoritmi di machine learning per la rilevazione di anomalie, si è deciso di utilizzare Jupyter Notebook. Jupyter Notebook è un'applicazione web

open source, che permette la creazione di documenti contententi oggetti quali paragrafi, equazioni, figure e codice sorgente eseguibile. Supporta diversi linguaggi di programmazione, incluso Python [73]. Jupyter facilita l'installazione e la configurazione delle varie librerie Python, tra le quali la libreria numpy, pandas, pandas-profiling, matplotlib, plotly, scikit-learn e TensorFlow.
La libreria *numpy* permette una facile gestione di diverse strutture dati, quali array e matrici, grazie alle funzioni matematiche di alto livello messe a disposizione [74]. La libreria *pandas*, invece, rappresenta la migliore libreria per effettuare data analysis. Fornisce, infatti, strumenti efficaci per l'importazione e la scrittura di dataset, per l'esplorazione, la pulizia e l'analisi dei dati. Pandas offre due principali strutture di dati: (i) *dataframe*, ovvero strutture dati bidimensionali contenenti assi etichettati (righe e colonne); (ii) *series*, ossia array uni-dimensionali [75].
Plotly e *Matplotlib* sono delle librerie Python utili per la visualizzazione dei dati. In particolare, consentono la realizzazione di diverse tipologie di grafici: dai grafici a barre, ai grafici in 3D [76]. Una libreria che rappresenta una valida alternativa a Plotly e Matplotlib, è *Seaboarn*. Nello specifico, si preferisce Seaborn per le visualizzazioni statiche, mentre per le visualizzazioni incorporate nel web, altamente interattive si predilige la libreria Plotly. In generale, queste librerie Python, forniscono un primo approccio, molto semplice ed immediato alla visualizzazione di dati, da preferire, ad esempio, nel caso in cui la si voglia integrare in un unico script di Jupyter notebook. Invece, per raggruppare i dati in strutture più complesse e rappresentative, quali dashboard, che migliorano la visualizzazione e l'analisi dei dati è consigliato utilizzare dei tool per la data visualization, approfonditi nella sezione 3.4.
Scikit-learn e *TensorFlow* sono delle librerie utili per l'applicazione di modelli di machine learning, di tipo unsupervised o supervised, e specificamente in ambito shallow learning, e deep learning [77].
Infine, *Pandas-profiling* è una libreria utile per la generazione di report dettagliati a partire da un dataframe pandas [78]. In particolare, rileva in modo autonomo i diversi tipi di colonne all'interno di un dataframe, identifica la presenza di valori mancanti, e calcola alcune statistiche descrittive come mediana, moda, deviazione standard, etc. Inoltre, la libreria pandas-profiling, è in grado di riconoscere i valori più frequenti e le correlazioni tra le variabili, attraverso il calcolo del coefficiente di correlazione di Pearson o Spearman.

3.4 Strumenti per la visualizzazione dati

La Data Visualization ha visto una crescita considerevole in seguito all'avvento dei Big Data e dell'analisi dei dati distribuita, in cui i dati provenienti da diverse sorgenti, venivano standardizzati e presentati in un'unica interfaccia grazie ai

front-end web dei browser [79]. La convergenza di queste soluzioni garantiva un'integrazione tra Data Visualization e Business Intelligence ed una valorizzazione di entrambe. In particolare, la Data Visualization consiste nella rappresentazione grafica dei dati, in due o tre dimensioni, statica o dinamica. Nel tempo, sono stati sviluppati diversi strumenti che sfruttano elementi visivi quali mappe, grafici, barre o altro, per offrire un modo semplice e intuitivo per la visualizzazione dei dati [80]. Tali strumenti sono utili per individuare situazioni anomale, tendenze o regolarità nei dati. Si pongono quindi come elementi di base per una efficace e semplice visualizzazione, spesso interattiva, dei dati. I tool per la Data Visualization sono, quindi, fondamentali in ambito industriale in quanto forniscono un'interfaccia user friendly, vantaggiosa in fase di comunicazione tra gruppi operativi con diverse competenze e conoscenze [80], [81]. Di fatti, rappresentano un mezzo indispensabile per i manager cosicchè essi possano monitorare l'andamento della produzione o delle vendite dei prodotti. Inoltre, questi tool forniscono al Planning gli strumenti utili per poter valutare le esigenze di mercato e confrontare il proprio andamento con quello dei competitor. Tali strumenti, di conseguenza, forniscono un supporto ai processi di reporting e analytics. Il reporting consiste nell'organizzazione dei dati in un unico documento di forma sintetica, chiamato report. Quest'ultimo si presenta come una composizione di grafici e tabelle che evidenziano le misure di rilievo associate ai dati analizzati. L'analytics, invece, è il processo di esplorazione dei dati, ed estrazione delle informazioni maggiormente significative [82].
Saper associare forma e funzioni in modo equilibrato, è quindi l'obiettivo di uno strumento di Data Visualization applicato ai Big Data. I dati vengono raggruppati in strutture rappresentative e tramite colori, sfumature, forme, operazioni statistiche è possibile dare vita a dashboard dinamiche, facilmente personalizzabili e interattive.
Di seguito, vengono approfondite le applicazioni che sono state utilizzate durante l'elaborazione del progetto di tesi e che hanno permesso una semplice ed efficace rappresentazione dei dati: Grafana [83] e Microsoft Power BI [84].

3.4.1 Grafana

Grafana è un'applicazione web utile per la visualizzazione e analisi interattiva dei dati. Si tratta di un software open source, user-friendly che permette di interrogare, visualizzare e comprendere facilmente i dati [80], [83].
Mediante Grafana, è possibile visualizzare infografiche e allarmi per il web su dashboard flessibili e versatili. Infatti, attraverso funzionalità avanzate di query e trasformazione, è possibile personalizzare le dashboard secondo le proprie necessità, per creare visualizzazioni significative, efficaci ed utili [83]. Grafana offre una varietà di visualizzazioni per supportare diversi casi d'uso, quali visualizzazione di

serie temporali (predefinita), grafici a barre per i dati categorici, istogrammi, heatmap, grafici a torta, tabelle, etc. Grafana può essere utilizzato per interagire con dati provenienti da diverse sorgenti preconfigurate, quali Influxdb, Elasticsearch, Mysql, Microsoft SQL Server, etc.
Permette, inoltre, di usare variabili nella formulazione delle query e nei titoli dei pannelli, al fine di creare dashboard facilmente riutilizzabili in casi d'uso differenti. Le variabili vengono visualizzate come elenchi a discesa nella parte superiore della dashboard. Modificando il valore di una variabile, le query relative alle metriche nella dashboard vengono aggiornate per riflettere il nuovo valore. Conseguentemente, invece di codificare i nomi delle metriche, sensori o server nelle query, è possibile usare le variabili che consentono la creazione di template generici e impiegabili in molteplici situazioni. Ad esempio, se si deve gestire una dashboard per monitorare diversi server, è possibile creare una dashboard per ciascun server o una sola se si utilizzano i template di query e si imposta il nome del server come variabile. In aggiunta, Grafana mette a disposizione dei meccanismi di alerting. Vengono monitorate delle regole di alerting, generalmente delle soglie, impostate dall'utente, ed inviate delle notifiche nel momento in cui l'alert cambia stato. Possono essere configurati diversi canali attraverso il quale ricevere gli avvisi, tra cui SMS e-mail o Slack. In figura 3.2 viene mostrato un esempio di dashboard personalizzata realizzata con l'ausilio dell'applicazione Grafana.

Figura 3.2. Esempio illustrativo di una dashboard realizzata con Grafana. Immagine da [80].

3.4.2 Microsoft Power BI

Microsoft Power BI è un servizio d'analisi aziendale, che offre una vista unificata dei dati di business più importanti. È costituito da diversi elementi che interagiscono tra loro, a partire dai tre fondamentali: (i) Power BI Desktop; (ii) un servizio SaaS (Software as a Service) online; (iii) app per dispositivi mobili [84].
Power BI, tramite un'interfaccia semplice e intuitiva, fornisce agli utenti una visualizzazione interattiva dei dati e funzionalità di business intelligence. Facilita gli utenti nella creazione di dashboard e report efficaci, con i quali è possibile monitorare, simultaneamente, l'integrità del business [81], [84]. Di fatti, l'analisi di dashboard consente agli utenti di ricevere, in tempo reale, una panoramica a 360 gradi circa le metriche più importanti. La creazione di dashboard risulta particolarmente semplice, grazie alla diffusione di template pre-definiti.
Power BI abilita l'accesso ai dati anche sui dispositivi mobile, tramite l'utilizzo di app native (Figura 3.3). Questo garantisce la possibilità di controllare periodicamente i dati, in qualsiasi luogo. Power BI, concede, inoltre, l'opportunità di pubblicare i report all'interno dell'azienda in completa sicurezza e di configurare l'aggiornamento automatico dei dati, al fine di mostrare le informazioni più recenti a tutti. In aggiunta, consente l'estrazione dei dati da diverse sorgenti, semplifica la preparazione degli stessi e l'esplorazione dettagliata [85]. Tutto questo, rende tale sistema un valido strumento di supporto alla Data Visualization. Di fatti, diverse sono le grandi aziende che oggigiórno hanno deciso di affidarsi a tale sistema. Per di più, la versione di Power BI Desktop è gratuita, mentre per la versione PRO, che permette di sfruttare ai massimi livelli il sistema, viene richiesta una quota mensile. Power BI è in continua evoluzione grazie alla sua community, che fornisce quotidianamente nuove idee per perfezionarsi e migliorare le performance.

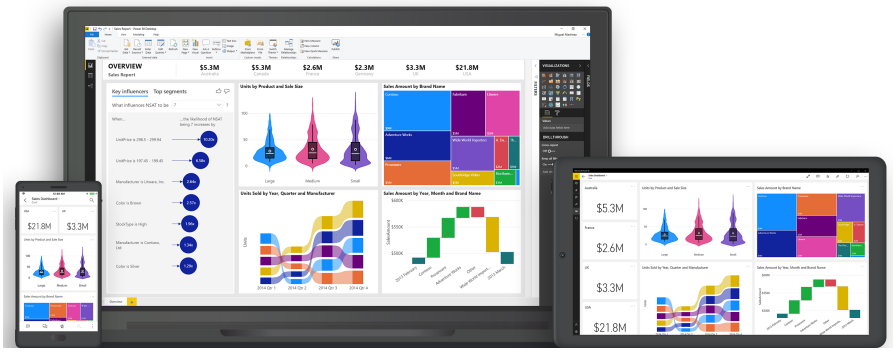


Figura 3.3. Esempio illustrativo di una dashboard realizzata con Microsoft Power BI e visualizzata su diversi dispositivi, mediante l'uso di app native. Immagine da [85].

Capitolo 4

Soluzione metodologica

In questo capitolo viene descritta, in modo dettagliato, l'architettura del progetto di tesi, riportando, precisamente, i passaggi sequenziali che sono stati compiuti. L'obiettivo è quello di mostrare le scelte che sono state attuate e le metodologie applicate in funzione dell'obiettivo proposto dalla tesi, ossia la realizzazione di uno struttura di base per uno strumento di supporto all'attività di monitoraggio, in grado di garantire una valutazione tempestiva delle problematiche e l'individuazione e visualizzazione puntuale delle metriche di performance di una base di dati, che presentano anomalie. Conseguentemente, nelle sezioni successive vengono presentate le metodologie utilizzate per l'individuazione e raccolta delle metriche di performance di una base di dati di un cliente, il processo di analisi esplorativa dei dati ottenuti e preparazione degli stessi, il processo di selezione delle caratteristiche e le metodologie applicate per la rilevazione della anomalie.

4.1 Individuazione e raccolta delle metriche di performance di una base di dati

Come riportato nella sezione 1.3, si è ritenuto necessario sviluppare una struttura di base per una soluzione da integrare all'interno degli strumenti di monitoraggio attualmente esistenti all'interno dell'azienda. Generalmente, i database dei clienti, gestiti dai dipendenti dell'area di infrastruttura tecnologica, presentano una tecnologia Oracle. Di conseguenza, si è deciso di analizzare esclusivamente le metriche di performance di una base dati Oracle. In particolare, è stata analizzata la vista *gv$sysstat*, messa a disposizione da Oracle, che fornisce una varietà e moltitudine di metriche di performance, che delineano un quadro preciso e dettagliato del comportamento di un database. Il vasto numero di tali metriche rende quasi impossibile a livello di tempo e di rapidità di reazione, l'analisi puntuale e manuale delle singole metriche per la rilevazione delle anomalie e delle corrispettive cause

di un malfunzionamento. Di fatti, gli impiegati, esperti nel settore, analizzano esclusivamente un gruppo limitato di metriche di performance, che consentono di ottenere una visione sufficientemente chiara delle attività e prestazioni di una base di dati, seppur alquanto soggettiva. In questo lavoro di tesi, invece, si è deciso di considerare ed esaminare tutte le metriche di performance della vista *gv$sysstat*, in modo tale da effettuare una successiva analisi approfondita delle metriche, e selezione delle stesse mediante l'utilizzo di metodi di apprendimento automatico. Come evidenziato nella tabella in figura 4.1, le colonne della *gv$sysstat* sono statistics#, name, class, value e stat_id, ossia, rispettivamente, il numero associato alla statistica, il nome della statistica, il numero della classe o delle classi di appartenenza, il valore che assume la statistica in un determinato istante temporale e l'identificativo univoco ad essa associato. Si mostra, quindi, come le statistiche, o metriche di performance, possano essere raggruppate in categorie, in base alla tipologia di informazione che offrono. In particolare, è possibile associare a ciascuna metrica di performance una o più classi di appartenenza specificando un numero identificativo della classe e il suo nome. Le principali classi di appartenenza delle metriche sono 9, e sono definite come: 1. User, 2. Redo, 4. Enqueue, 8. Cache, 16. OS, 32. Real application Clusters, 64. SQL, 128. Debug, 256. Instance level. I numeri delle classi, inoltre, sono additivi, ovvero una classe con numero identificativo pari a 72 è definita da 8+64, quindi fa riferimento sia alla classe SQL, che a quella di Cache. Un esempio, è la metrica *Batched IO (bound) vector coun*t, che appartiene alla classe 72. Tra le metriche associate alla classe User, invece, ci sono le metriche che definiscono la quantità di tempo di utilizzo della cpu in una sessione, *CPU used by this session* , o il numero di chiamate ricorsive generate sia a livello utente che di sistema, *recursive calls*. Nella classe di Redo, invece, rientrano le metriche che descrivono, ad esempio, il numero totale di blocchi redo scritti, *redo blocks written*, o il numero di volte in cui una redo entry è copiata in un buffer di redo log, *redo entries*. In Enqueue, invece, appartengono le metriche che identificano eventi di attesa, come ad esempio *enqueue waits*, la quale indica che una sessione è in attesa di un blocco che è tenuto da un altro utente (o sessioni). Le metriche che appartengono alla classe di Cache, sono metriche che rilevano eventi associati all'utilizzo delle cache (quali, *blocks encrypted*, *blocks decrypted*, *cell flash cache read hits*), mentre nella classe OS vengono raggruppate le metriche che forniscono informazioni sull'utilizzo e sulle prestazioni del sistema operativo (come *OS User time used, OS Page faults, OS Block input operations*) . Nella classe SQL, invece, vengono raccolte le metriche di performance che evidenziano informazioni relative alle istruzioni sql eseguite sul database (come *session cursor cache hits*, ovvero il numero di hits nella cache del cursore di sessione. Un hit implica che l'istruzione SQL non deve essere riparata). Nella classe di Debug, invece, rientrano una moltitudine di statistiche, relative, ad esempio, alle operazioni di rollback effettuate.

Column	Datatype	Description
STATISTIC#	NUMBER	Statistic number **Note:** Statistics numbers are not guaranteed to remain constant from one release to another. Therefore, you should rely on the statistics name rather than its number in your applications.
NAME	VARCHAR2(64)	Statistic name. You can get a complete listing of statistic names by querying the V$STATNAME view.
CLASS	NUMBER	A number representing one or more statistics class. The following class numbers are additive: • 1 - User • 2 - Redo • 4 - Enqueue • 8 - Cache • 16 - OS • 32 - Real Application Clusters • 64 - SQL • 128 - Debug • 256 - Instance level
VALUE	NUMBER	Statistic value
STAT_ID	NUMBER	Identifier of the statistic

Figura 4.1. Descrizione dettagliata delle colonne della vista *gv$sysstat*. Tabella da [86].

In questo progetto, le metriche di performance esaminate sono state 2011 e ciascuna è stata campionata con un intervallo temporale di un minuto. Per l'estrazione e la raccolta dei dati delle metriche di performance, si è partiti dalla vista *gv$sysstat*. Nello specifico, gli esperti del sistema, dell'area di infrastruttura, hanno creato una tabella dal medesimo nome, GV_SYSSTAT, in grado di raccogliere, ad intervalli periodici di 1 minuto, i valori assunti dalle metriche di performance. Si è deciso, inoltre, di realizzare un'architettura in grado di prelevare in real time i nuovi record inseriti all'interno di questa tabella, mediante l'utilizzo della piattaforma Confluent, e di esaminare ed analizzare in tempo reale l'andamento delle metriche di performance tramite tool per la visualizzazione dei dati. L'obiettivo della metodologia proposta è quello di eseguire una prima analisi delle metriche di performance, basata esclusivamente sulla visualizzazione grafica dell'andamento dei valori nel tempo. In prospettiva futura, tale architettura può rappresentare uno strumento utile per l'analisi e l'identificazione di anomalie nelle metriche di performance in tempo reale. Nella prima fase progettuale, ci si è focalizzati, quindi, sull'estrazione dei dati in real-time, caricamento dati all'interno di un time series database e di un database Oracle, e visualizzazione grafica delle informazioni utili mediante dashboard. La pipeline di dati in streaming è stata realizzata mediante l'utilizzo di diverse tecnologie, quali Confluent Platform, InfluxDB e Grafana, come riportato in figura 4.2. Confluent Platform è stata, quindi, la piattaforma streaming utilizzata per archiviare ed organizzare i flussi di dati.

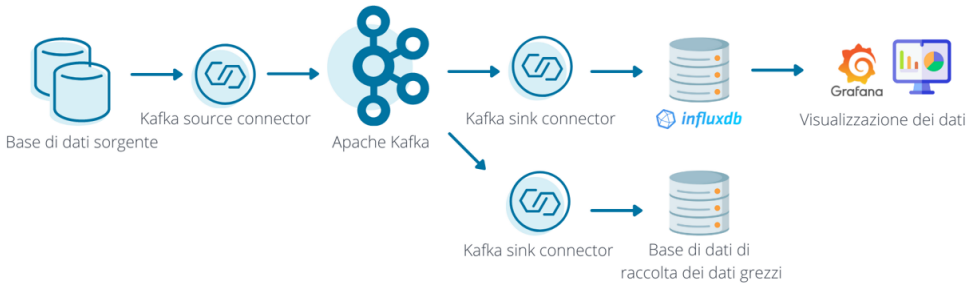


Figura 4.2. Processo di estrazione ed analisi delle metriche di performance in tempo reale mediante Confluent Platform, InfluxDB e Grafana. In figura, il flusso alternativo rappresenta il processo di storicizzazione dei dati in una basa di dati Oracle.

Tale piattaforma espande i vantaggi della piattaforma Apache Kafka, rimuovendo l'onere della gestione dei meccanismi sottostanti, come ad esempio il processo di trasporto dati e l'integrazione tra i diversi sistemi. Confluent, infatti, fornisce una serie di connettori precostituiti che permettono di estrarre dati dalle sorgenti e di trasportali verso strutture in output. In questa architettura è stato utilizzato un connettore JDBC per realizzare una connesione tra il database Oracle, all'interno del quale è contenuta la tabella sorgente, e il nucleo Apache Kafka della piattaforma Confluent. In seguito, impostando la proprietà *auto.create.topics.enable* a true, è stato creato in modo automatico dalla piattaforma Confluent, un Kafka topic, il quale è in grado di conservare le informazioni provenienti dalla tabella sorgente. In generale, abilitando tale impostazione, i kafka topics vengono generati automaticamente quando le applicazioni producono, consumano o recuperano metadati da un topic non ancora esistente. Le informazioni trasmesse ai Kafka topics, vengono inviate sottoforma di messaggi. I Kafka topics, quindi, sono utili per la gestione e organizzazione dei messaggi. Questi, vengono inviati o letti da topics specifici. In particolare, i produttori sono le applicazioni client che pubblicano (scrivono) dati su dei Kafka topics, mentre i consumatori sono coloro che si iscrivono ai Kafka topics e quindi sono in grado di leggere e processare i dati dai topics. In Kafka, produttori e consumatori sono completamente separati. Ad esempio, i produttori non devono mai aspettare che i consumatori abbiano terminato le loro operazioni. Di conseguenza, questo permette a Kafka di raggiungere elevati livelli di scalabilità. Ciascun topic assume un nome univoco nell'intero cluster Kafka, e può avere da zero a molteplici consumatori e/o produttori. Successivamente la creazione del Kafka topic, è stato costruito un connettore sink di tipo InfluxDB, che permette di inserire i dati provenienti da un Kafka topic, all'interno del time series database InfluxDB. Inoltre, è stato configurato un connettore sink di tipo JDBC, al fine

di inserire i dati provenienti dal Kafka topic in un database Oracle, adatto alla storicizzazione dei dati. Pur usufrendo di Kafka connector pre-costruiti, è stato necessario modificarli ad hoc per farli funzionare all'interno dell'architettura. In particolare, durante il processo di creazione del *Source Connector*, è stata configurata la connesione verso la base di dati sorgente. Sono stati definiti, quindi, il JDBC url, l'utente, la password ed il numero massimo di tentativi di riconnessione. Inoltre, è stata specificata la tabella sorgente da cui estrarre i dati, e la modalità di aggiornamento della tabella. Esistono 4 diverse modalità: (i) bulk: periodicamente viene ricaricata tutta la tabella; (ii) incrementing: usa una colonna strettamente incrementale della tabella per rilevare esclusivamente le nuove righe. Tale colonna deve essere specificata; (iii) timestamp: utilizza una colonna di tipo timestamp per rilevare esclusivamente le righe nuove e modificate; (iv) timestamp + incrementing: sfrutta una colonna di tipo timestamp per identificare le righe nuove o modificate e la colonna strettamente incrementale per definire un ID unico a livello globale. Nel lavoro di tesi, si è scelto un aggiornamento della tabella di tipo incrementale. L'ultima informazione configurata nella struttura del connettore sorgente, è stata il prefisso del topic in cui pubblicare i dati. Dopo aver definito i dettagli della struttura del connettore sorgente, la sua configurazione è stata memorizzata in un file con formato JSON, come riportato in figura 4.3.
Nella configurazione del *Sink Connector* di tipo InfluxDB (figura 4.4), invece, sono stati definiti i parametri per la connessione al database InfluxDB di destinazione, quali l'url e il nome del database. Inoltre, è stato specificato il nome del topic di Apache Kafka (dal quale il connettore preleva i record per poterli inserire nel database di destinazione), ed il nome della misura di destinazione (measurement.name.format). Si è indicato, anche, il nome del campo nel record Kafka, che identifica l'istante temporale in cui si è verificato l'evento. Tale timestamp deve essere inserito nella base di dati di destinazione. Se non impostato, il valore temporale di default scritto su Influxdb è il timestamp del record Kafka, ovvero il momento in cui è stato creato il record Kafka. L'ultimo connettore configurato è stato il *JDBS Sink Connector* (figura 4.5). Anche in questo caso, al fine di creare il connettore, sono stati definiti i parametri per la connessione al database Oracle utilizzato per la storicizzazione dei dati. Conseguentemente, sono stati definiti il JDBC url, l'utente, la password e il numero massimo di tentativi di riconnessione. Infine, si è specificato il nome del topic da cui estrarre i dati e il nome della tabella nel database Oracle di destinazione in cui inserire i dati.

I dati estratti dalla piattaforma Confluent, tramite il procedimento precedentemente descritto, giungono in un time series database InfluxDB e in una base di dati Oracle. All'interno del database di InfluxDB i dati non sono stati ulteriormente manipolati. Infatti, in questo elaborato si è deciso di sfruttare tale database esclusivamente come staging area dei dati e come punto di partenza per la visualizzazione dei dati su Grafana. Grafana, d'altronde, supporta innumerevoli sorgenti

```
{
  "name": "JdbcSourceConnectorConnector",
  "config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",
    "tasks.max": "1",
    "connection.url": "jdbc:oracle:thin:@//192.168.2.83:1521/OPA",
    "connection.user": "UC_METRICS",
    "connection.password": "**********",
    "connection.attempts": "3",
    "table.whitelist": "GV_SYSSTAT",
    "mode": "incrementing",
    "incrementing.column.name": "RECORDID",
    "table.types": "TABLE",
    "topic.prefix": "ucmetrics"
  }
}
```

Figura 4.3. File JSON del JDBC Source Connector.

```
{
  "name": "InfluxDBSinkConnector",
  "config": {
    "connector.class": "io.confluent.influxdb.InfluxDBSinkConnector",
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "topics": "ucmetricsGV_SYSSTAT",
    "influxdb.url": " http://192.168.2.81:8086",
    "influxdb.db": "ucmetrics_db",
    "measurement.name.format": "${topic}",
    "event.time.fieldname": "SAMPLE_TIME",
    "influxdb.timeunit": "MILLISECONDS"
  }
}
```

Figura 4.4. File JSON dell' InfluxDB Sink Connector.

```
{
  "name": "JdbcSinkConnectorConnector",
  "config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",
    "topics": "ucmetricsGV_SYSSTAT",
    "connection.url": "jdbc:oracle:thin:@//192.168.2.83:1521/OPA",
    "connection.user": "Daniela",
    "connection.password": "************",
    "connection.attempts": "3",
    "table.name.format": "kafka_${topic}"
  }
}
```

Figura 4.5. File JSON del JDBC Sink Connector.

di dati, incluso InfluxDB. Diversamente, il database Oracle è stato utilizzato per storicizzare i dati e come sorgente per le successive fasi di preparazione dei dati, analisi esplorativa ed individuazione delle anomalie.
L'applicazione web Grafana è stata fondamentale durante la fase iniziale del progetto di tesi, in quanto ha permesso una prima analisi e visualizzazione interattiva dei dati. Tramite semplici tool grafici presenti all'interno dell'applicazione web, è stato possibile configurare facilmente le query verso il database sorgente e modificare l'aspetto della visualizzazione dei risultati. Inoltre, al fine di creare delle dashboard facilmente riutilizzabili per le differenti metriche, è stato realizzato un template di query, sostituendo il nome della metrica con una variabile. In questo modo, è stato possibile visualizzare graficamente ed in tempo reale, l'andamento delle diverse metriche di performance della base di dati, nel tempo. Dall'analisi dei grafici, è emerso che una moltitudine di metriche di performance assumevano valori nulli o costanti. Conseguentemente, si è deciso di non considerare tali metriche di performance, in quanto prive di informazioni utili e soprattutto poco significative per la rilevazione di anomalie. In particolare, la rimozione di tali metriche rumorose è avvenuta durante il processo di preparazione dei dati, descritto nella sezione 4.2. L'analisi in tempo reale dei grafici delle metriche ha evidenziato la presenza di un errore nella visualizzazione dei dati. Nello specifico, tale errore è riconducibile ad una mal interpretazione dei valori assunti dalle metriche. Questo perchè non sempre il valore registrato di una metrica corrisponde al reale valore assunto dalla metrica in un preciso istante temporale. Di fatti, tale valore può essere definito dalla somma del valore della metrica all'istante precedente e il valore reale assunto dalla metrica in uno specifico istante temporale. Conseguentemente, durante il processo di preparazione dei dati, descritto nella sezione 4.2, è stata applicata una trasformazione sui dati in modo tale da otterene un set di dati in grado di rappresentare in modo corretto l'andamento reale delle diverse metriche di performance. Il processo di raccolta ed estrapolazione dati è durato in totale 8 settimane e 6 giorni, a partire dal 2021-06-04 00:00:00 fino al 2021-08-04 23:59:00. Sono stati memorizzati nella base di dati Oracle 179.542.080 record, in quanto è stato inserito 1 record al minuto per 62 giorni per le 2011 metriche considerate.

4.2 Preparazione dei dati

La preparazione dei dati, talvolta anche chiamata pre-elaborazione, consiste in un processo di pulizia e trasformazione dei dati grezzi usati nelle successive fasi di elaborazione ed analisi. Si tratta di un passaggio fondamentale, in cui si controllano i dati al fine di evitare la presenza di possibili errori. Nel caso in cui essi fossero presenti, è necessario procedere ad una correzione degli stessi. L'obiettivo di questa fase è, infatti, iniziare a selezionare dati di ottima qualità al fine di ottenere

ottime prestazione durante il processo di elaborazione ed analisi. La preparazione dei dati rappresenta, quindi, un prerequisito essenziale, che consente la trasformazione dei dati in informazioni rilevanti, mediante rimozione delle distorsioni, come dati ridondanti, incompleti o non corretti. Generalmente, il processo di preparazione dei dati include la standardizzazione dei formati, l'arricchimento dei dati e/o eliminazione dei valori anomali. Nel progetto di tesi il processo di preparazione dei dati è risultato essenziale, in quanto ha permesso la trasformazione dei dati grezzi in dati di qualità migliore. La prima operazione eseguita è stata una semplice trasformazione dei valori assunti dalle metriche di performance. Questo passaggio è risultato fondamentale in quanto i valori assunti dalle metriche di performance nei diversi istanti temporali, non rappresentavano sempre i reali valori assunti dalle metriche. In particolare, il valore della metrica al tempo t deriva dalla somma del valore della metrica all'istante *t-1* e del valore reale assunto dalla metrica all'istante temporale t, se il valore della metrica all'istante *t-1* è minore del valore all'istante t. Se, invece, il valore della metrica in un preciso istante temporale t risulta minore del valore all'istante *t-1*, allora non sono necessarie delle trasformazioni. Tale trasformazione è stata eseguita mediante l'esecuzione di una query SQL sul database di raccolta dei dati, mostrata in figura 4.6.

```
CREATE TABLE UC_METRICS.V_SYSTAT_TRASFORMED as
SELECT NAME,
       SAMPLE_TIME,
       VALUE,
       P_VALUE,
       CASE
           WHEN VALUE = 0 THEN VALUE
           WHEN P_VALUE IS NULL THEN VALUE
           WHEN VALUE < P_VALUE THEN VALUE
           ELSE VALUE - P_VALUE END ABSOLUTE_VALUE
FROM (SELECT NAME, SAMPLE_TIME, VALUE, LAG(VALUE) over (PARTITION BY NAME ORDER BY SAMPLE_TIME ASC) P_VALUE
      FROM UC_METRICS.V_SYSSTAT
     ) A;
```

Figura 4.6. Query SQL che ha consentito la trasformazione dei dati grezzi precedentemente raccolti. In particolare, tramite la seguente query è stato possibile ricavare il corretto valore assunto da ciascuna metrica nei diversi istanti temporali.

4.2.1 Pulizia dei dati

In seguito alle osservazioni fatte in fase di esplorazione dei dati, riportate nella sezione 4.3, si è proceduto con l'eliminazione delle metriche con valori costanti o sempre nulli. La rimozione di tale metriche ha comportando una riduzione notevole del numero totale di metriche di performance prese in considerazione. Si è passati, infatti, da 2011 metriche a 408 metriche. Nonostante ciò, si è ritenuto tale numero ancora non soddisfacente e possibilmente ulteriormente riducibile. Conseguentemente, si è deciso di effettuare degli approfondimenti aggiuntivi, riportati

sezione 4.4. Nello specifico, sono state valutate ed applicate delle metodologie per la selezione ulteriore delle metriche di performance.

4.2.2 Ridimensionamento dei dati

Il ridimensionamento dei dati, o Feature scaling, è un processo che viene generalmente eseguito durante la fase di pre-elaborazione dei dati. Si tratta di un processo fondamentale, in quanto permette di standardizzare l'intervallo di variabili indipendenti o delle caratteristiche dei dati [87]. Il ridimensionamento dei dati, è un metodo utile nel caso di applicazione di algoritmi di apprendimento automatico basati sulla distanza. Molti classificatori, ad esempio, utilizzano come misura la distanza euclidea. Conseguentemente, se una caratteristica (feature) assume una vasta gamma di valori, influsce maggiormente nel calcolo della distanza. Pertanto è necessario normalizzare i dati, in modo che ogni caratteristica contribuisca proporzionalmente alla distanza finale. Nel progetto di tesi, in seguito all'analisi esplorativa dei dati, riportata nella sezione 4.3, è emerso che la gamma di valori assunti dalle metriche di performance variava ampiamente. Di conseguenza, è stata eseguita una normalizzazione dei dati, al fine di ottenere per ciascuna metrica di performance la stessa scala. Sono state valutate due metodologie di ridimensionamento dei dati: la normalizzazione min max e la standardizzazione o normalizzazione Z-Score. In seguito ad uno studio approfondito del dominio si è deciso di utilizzare per il presente lavoro di tesi una normalizzazione min max, con lo scopo di lasciare i valori positivi come in partenza. I dati sono stati ridimensionati e scalati su un intervallo fisso [0,1]. Per ciascuna metrica, il suo valore minimo è stato trasformato in 0, il suo valore massimo in 1, e qualsiasi altro valore è stato trasformato in un decimale compreso tra 0 e 1. Questa normalizzazione è stata applicata per migliorare l'accuratezza dei modelli di machine learning attuati per la selezione delle metriche di performance, descritti nella sezione 4.4. La formula per una normalizzazione min-max è la seguente:

$$Z = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{4.1}$$

dove Z è il valore normalizzato, x il valore originale e min/max si riferiscono rispettivamente ai valori minimi e massimi del set di dati.

4.3 Analisi esplorativa dei dati

L'analisi esplorativa dei dati, o EDA (Exploratory data analysis), è il processo attraverso il quale è possibile studiare i dati ed identificare potenziali relazioni tra le variabili, mediante l'uso di di riepiloghi numerici e rappresentazioni grafiche.

In particolare, si sfruttano sia tabelle di riepilogo statistico, che evidenziano i valori delle stastiche descrittive, quali media, deviazione standard, valore minimo e massimo, sia grafici quali istogrammi, box plots, matrici di correlazione, etc. L'obiettivo dell'analisi esplorativa dei dati è comprendere la distribuzione delle singole variabili in un set di dati, individuare possibili relazioni tra variabili, verificare la presenza di outlier o punti insoliti ed identificare dei pattern nel tempo. L'analisi esplorativa consente, quindi, di formulare domande o ipotesi interessanti che possono essere verificate in seguito tramite l'applicazione di metodi statistici più rigorosi. Nel progetto di tesi, l'analisi esplorativa dei dati è stata effettuata tramite il supporto della libreria pandas-profiling. Tramite la funzione ProfileReport, è stato possibile generare automaticamente un report dettagliato contenente le informazioni relative alla più comuni statistiche descrittive e alcune rappresentazioni grafiche dei dati. Nello specifico, vista la vastità di metriche in esame, non è stato possibile utilizzare la versione estesa della libreria, ma esclusivamente una sua versione limitata. Tale versione è stata abilitata andando ad inserire all'interno della funzione ProfileReport, il parametro minimal=True. Questa configurazione disabilita i calcoli computazionalmente costosi, come l'individuazione delle correlazioni tra le variabili e il rilevamento di righe duplicate. Conseguentemente, all'interno del report realizzato, non è stato possibile includere matrici di Spearman, Pearson e Kendall, le quali permettono di identificare le correlazioni tra le variabili. Nonostante ciò, l'analisi delle correlazioni tra le variabili è stata eseguita in una fase successiva del lavoro di tesi, ed è risultata parte fondamentale del processo di selezione delle metriche di perfomance. Il report generato è stato memorizzato in formato html, in modo tale da permettere una facile visualizzazione e un'intuibile interazione. Il report è caratterizzato, infatti, da una serie di bottoni con i quali è possibile interagire per visualizzare o nascondere ulteriori informazioni. Ad esempio, come riportato in figura 4.7, premendo il bottone "Toggle details" è possibile visualizzare la finestra sottostante il bottone, che fornisce dettagli rilevanti, quali statistiche descrittive e quantili della metrica. Invece, il bottone "Common Values" consente la visualizzazione dei valori più comuni per tale metrica. Premendo nuovamente sul tasto Toggle details, è possibile nascondere la finestra sottostante. Il riquadro di informazioni nella parte superiore del report, invece, è sempre visibile e offre una panoramica delle statistiche fondamentali associate ad una metrica, quali il numero di valori distinti, mancanti, pari a zero e negativi e le loro rispettive percentuali. Inoltre, vengono mostrate la media, il valore minimo e il valore massimo. Queste informazioni vengono riportate per ciascuna metrica di performance. Il tasto Overview, in alto a destra nella figura 4.7, fornisce delle informazioni di carattere generale, relative a tutte le metriche in esame. Nello specifico, mostra il numero totale di metriche di perfomance esaminate, il numero totale di osservazioni, quanti valori nulli sono presenti tra tutte le metriche di performance e la tipologia di variabile. In questo caso, ciascuna metrica trattata

era di tipo numerico.

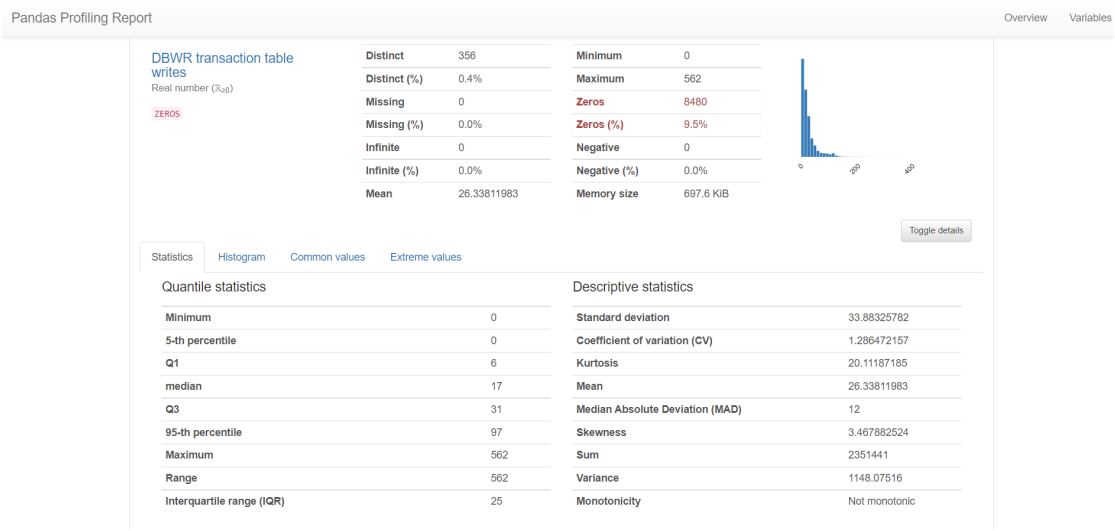
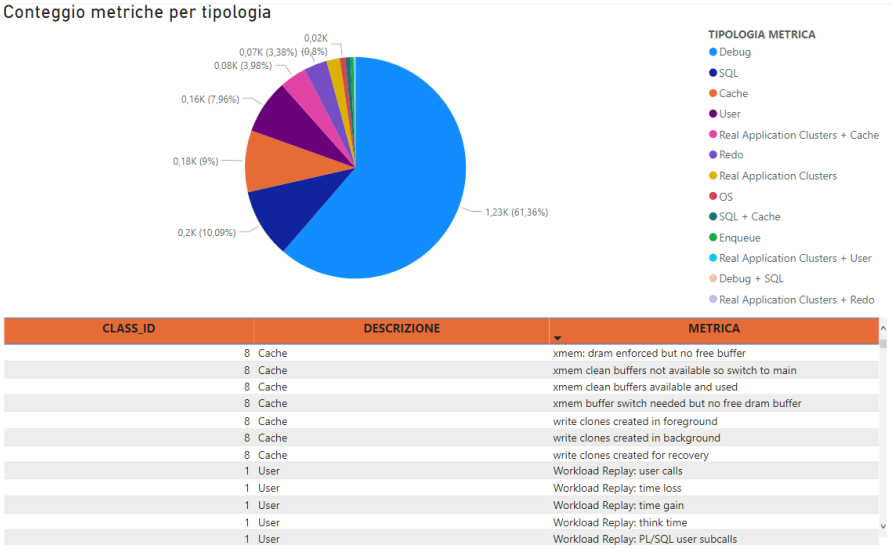


Figura 4.7. Rappresentazione delle statistiche descrittive e dei quantili della metrica DBWR transaction table writes, estratta dal report generato con il supporto della libreria pandas-profiling.

L'analisi esplorativa dei dati ha confermato la presenza di un vasto numero di metriche di performance con valori nulli o costanti. In aggiunta, si è notato come alcune metriche mostrino un andamento altamente sbilanciato (highly skewed) e caratterizzato da diversi valori prossimi allo zero e singoli picchi che si discostano dal resto delle osservazioni, o comportamenti insoliti per brevi periodi di tempo. Queste osservazioni verranno poi ulteriormente investigate nella sezione 4.5, in cui vengono riportate le diverse metodologie utilizzate per l'identificazione delle anomalie nelle metriche di performance.

Per l'analisi esplorativa dei dati si è utilizzato anche Power BI, strumento per la visualizzazione dei dati. Nello specifico, si è esaminata la distribuzione delle metriche di performance in base alla tipologia (classe di appartenenza). L'obiettivo era individuare il numero di metriche per ciascuna tipologia e quantificare la percentuale di ciascuna tipologia rispetto al totale delle metriche (figura 4.8). La classe di appartenenza più rappresentativa è la classe Debug.



CLASS_ID	DESCRIZIONE	METRICA
8	Cache	xmem: dram enforced but no free buffer
8	Cache	xmem clean buffers not available so switch to main
8	Cache	xmem clean buffers available and used
8	Cache	xmem buffer switch needed but no free dram buffer
8	Cache	write clones created in foreground
8	Cache	write clones created in background
8	Cache	write clones created for recovery
1	User	Workload Replay: user calls
1	User	Workload Replay: time loss
1	User	Workload Replay: time gain
1	User	Workload Replay: think time
1	User	Workload Replay: PL/SQL user subcalls

Figura 4.8. Rappresentazione grafica della distribuzione delle metriche di performance per tipologia.

4.4 Selezione delle metriche di performance

La selezione delle caratteristiche, o feature selection, è il processo di riduzione del numero di variabili di input, mediante rimozione delle features irrilevanti, ridondanti o rumorose (sezione 2.1). Nel lavoro di tesi, il processo di selezione delle metriche di performance è risultato indispensabile ed imprescindibile, in quanto ha comportato una riduzione notevole del numero di metriche da considerare nelle successive fasi di elaborazione dei dati. In questa fase progettuale è stato diminuito il numero totale di metriche di performance, al fine di migliorare ed agevolare l'interpretabilità e l'applicazione dei modelli univariati usati per l'identificazione delle anomalie. Nello specifico, è stata proposta una metodologia basata sull'applicazione iterativa di algoritmi di clustering ed analisi della correlazione delle metriche in ciascun cluster individuato (figura 4.9).
Il primo passaggio eseguito è stato una trasformazione della matrice dei dati. In particolare, sono stati trasposti gli indici e le colonne, riportando i valori delle metriche di performance sulle righe, anzichè sulle colonne. La trasformazione permetterà l'applicazione di modelli di clustering sulle metriche di performance.

Il primo modello di clustering applicato è stato un algoritmo di DBSCAN, Density-Based Spatial Clustering of Applications with Noise, importato dalla libreria *sklearn*. I parametri definiti sono stati *eps* e *min_sample*. Il primo rappresenta la distanza

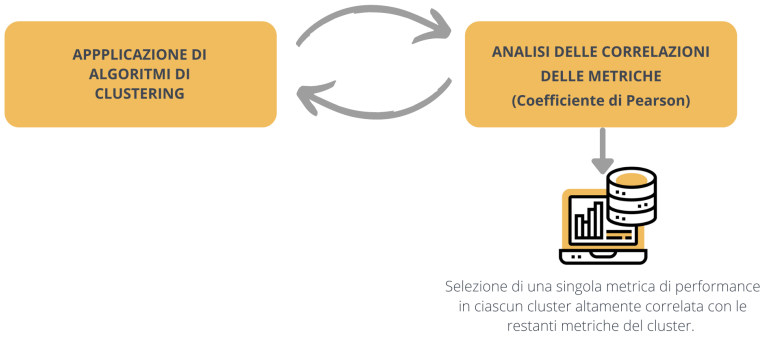


Figura 4.9. Processo di selezione delle metriche di performance tramite applicazione iterativa di algoritmi di clustering ed analisi delle correlazione delle metriche di performance in ciascun cluster. Per ogni cluster, viene selezionata, una singola metrica di performance, con un elevato coefficiente di correlazione con le altre metriche.

massima tra due campioni per essere considerati prossimi tra di loro, e di conseguenza interni ad uno stesso cluster. Tale valore però non rappresenta un limite massimo sul valore delle distanze dei punti presenti all'interno di un cluster. Il parametro eps deve essere scelto in modo appropriato rispetto al dataset da analizzare e alla funzione di distanza scelta, per ottenere risultati significativi dall'algoritmo. Il parametro *min_sample*, invece, identifica il numero minimo di campioni vicini per considerare un punto un core point, e la regione di punti come densa. Nel lavoro di tesi, la scelta dei parametri è stata eseguita in maniera empirica tramite un'analisi approfondita del dominio dei dati e con l'obiettivo di raggruppare il più possibile le metriche in cluster, seppur di dimensione piccola. In particolare, sono state confrontate le esecuzioni con le diverse combinazioni dei due valori. La stima del valore del parametro eps è stata effettuta mediante l'utilizzo dell'algoritmo Nearest Neighbors. Tale metodologia consiste nel calcolare la distanza media tra ciascun punto nel set di dati e i suoi *min_pts* punti più vicini. In seguito, si ordinano i valori della distanza in ordine crescente e si produce un grafico denominato k-distance graph. Il valore ideale del parametro eps corrisponde al punto di curvatura massima. Dopo aver definito i valori dei parametri dell'algoritmo DBSCAN, si è proceduto all'esecuzione dello stesso. Sono stati stimati 7 cluster di metriche di performance ed un cluster di punti rumore, ovvero contenete le restanti metriche non collocate in alcun cluster (tabella 4.1). Successivamente, è stata effettuata un'analisi delle correlazioni delle metriche di performance raccolte

in ciascun cluster. Nello specifico, si è esaminata la matrice di correlazione delle metriche. La matrice di correlazione è una tabella quadrata che riporta al suo interno gli indici di correlazione di Pearson tra due o più variabili. L'obiettivo è identificare delle forti correlazioni tra le metriche in uno stesso cluster, in modo tale da selezionare per ciascun cluster una singola metrica. In particolare, se le correlazioni tra le metriche sono prossime ad 1 in uno stesso cluster, le metriche di performance sono simili tra di loro e sono in grado di modellare lo stesso evento. Conseguentemente, la selezione di una singola metrica in ciascun cluster non comporta alcuna perdita di informazione. Dall'analisi delle matrici di correlazione è emerso che l'algoritmo DBSCAN è stato in grado di identificare 7 gruppi di metriche con indici di correlazione vicini ad 1. Ad esempio, i cluster con etichetta 2 e 5, contenenti rispettivamente 9 e 6 metriche di performance, presentano valori di indice di correlazione che superano 0.999 (figure 4.10, 4.11).
Queste osservazioni hanno consentito la selezione di una singola metrica di performance per i 7 cluster di metriche precedentemente individuati.
Per le restanti 366 metriche di performance contenute nel cluster con etichetta -1, l'analisi della matrice di correlazione ha evidenziato la presenza di alcune correlazioni tra piccoli gruppi di metriche. In particolare, sono stati individuati 42 sottogruppi di metriche, ciascuno contenente da 2 fino a 14 metriche di performance con indici di correlazione che superano il valore di soglia 0.94 (figura 4.12). Il valore della soglia è stato scelto dopo un'accurata analisi dei grafici delle metriche di performance tramite PowerBI, strumento per la visualizzazione dei dati. Nello specifico, sono stati valutati diversi valori di indici di correlazione, partendo dal valore massimo 1 e seguendo con la sua progressiva diminuzione. Si è deciso di fermare il valore soglia di indice di correlazione a 0.94, in quanto analizzando i grafici è emerso che molte metriche di performance, pur presentando un indice di correlazione al di poco inferiore di 0.94, mostravano un andamento molto diverso.

Etichetta cluster	Numero di metriche
-1	366
0	3
1	4
2	9
3	4
4	4
5	6
6	3

Tabella 4.1. Tabella riassuntiva dei risultati del primo algoritmo DBSCAN.

Anche in questo caso, si è deciso di selezionare una singola metrica per ciascun gruppo di metriche identificato.

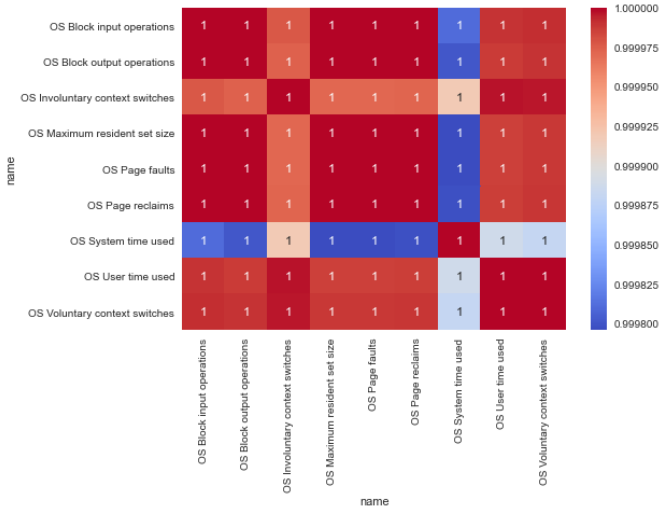


Figura 4.10. Primo algoritmo DBSCAN: matrice di correlazione delle nove metriche di performance del cluster con etichetta numero 2.

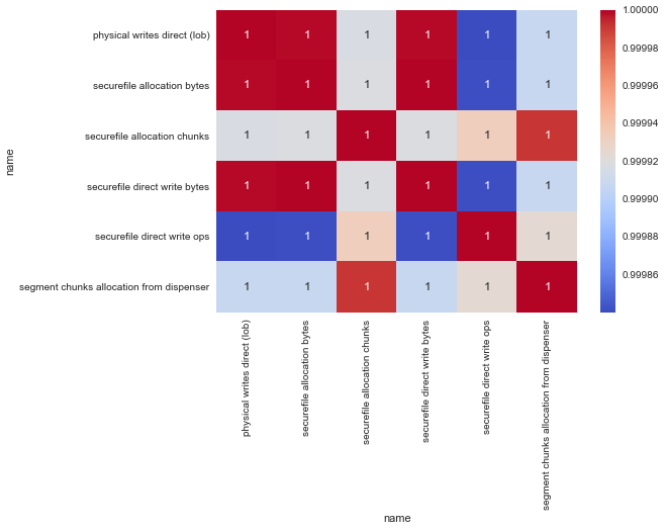


Figura 4.11. Primo algoritmo DBSCAN: matrice di correlazione delle sei metriche di performance del cluster con etichetta numero 5.

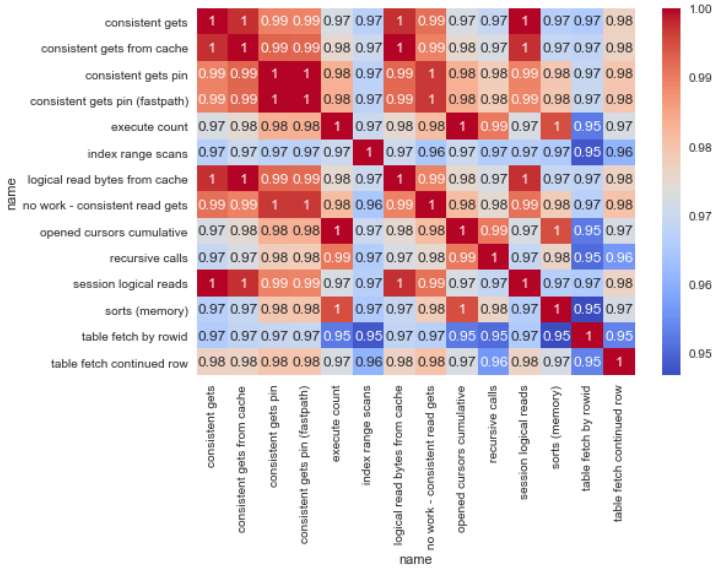


Figura 4.12. Primo algoritmo DBSCAN: matrice di correlazione di un sottogruppo di metriche del cluster con etichetta -1.

Il secondo modello di clustering utilizzato è stato nuovamente un algoritmo di DBSCAN. Tale algoritmo è stato applicato esclusivamente sulle metriche appartenenti al cluster -1, le quali sono rimaste escluse dai sottogruppi di metriche con un elevato coefficiente di correlazione. I parametri definiti sono stati *eps* e *min_sample*. La stima di tali parametri è avvenuta nello stesso modo definito nel primo modello di DBSCAN eseguito. In particolare, sono state confrontate le esecuzioni con le diverse combinazioni dei due valori. La stima del valore del parametro eps è stata effettuta mediante l'utilizzo dell'algoritmo Nearest Neighbors. L'esecuzione del secondo algoritmo di DBSCAN, ha comportato la creazione di ulteriori 6 cluster di metriche di performance, ed un cluster di punti rumore, ovvero contenete le restanti metriche non collocate in al alcun cluster (tabella 4.2).

Successivamente, è stata effettuata un'analisi delle correlazioni delle metriche di performance raccolte in ciascun cluster. Nello specifico, si è esaminata la matrice di correlazione delle metriche. L'obiettivo è l'identificazione delle forti correlazioni tra le metriche in uno stesso cluster, in modo tale da selezionare per ciascun cluster una singola metrica. Dall'analisi delle matrici di correlazione è emerso che l'algoritmo DBSCAN è stato in grado di identificare 6 gruppi di metriche, caratterizzati ciascuno da 2 metriche fortemente correlate tra di loro, in quanto gli indici di correlazione erano prossimi ad 1. Ad esempio, il cluster con etichetta 0, che raggruppava la metrica *leaf node splits* e la metrica *ASSM gsp:get free index block*,

Etichetta cluster	Numero di metriche
-1	222
0	2
1	2
2	2
3	2
4	2
5	2

Tabella 4.2. Tabella riassuntiva dei risultati del secondo algoritmo DBSCAN.

presentava una correlazione pari a 0.999973 (figura 4.13). Le due metriche risultavano altamente correlate tra di loro, ed è stata possibile una selezione casuale di una della due metriche. Questo processo di selezione è stato ripetuto per ciascuno dei 6 cluster di metriche precedentemente individuati.

Per le restanti 222 metriche di performance contenute nel cluster con etichetta -1, l'analisi della matrice di correlazione ha evidenziato la presenza di molte metriche non fortemente correlate tra di loro. Nello specifico, gli indici di correlazione generalmente non superavano la soglia impostata a 0.94. Questo valore è stato scelto all'inizio del processo di selezione delle metriche di performance. Di conseguenza, si è preferito considerare l'intero set di 222 metriche per l'applicazione del terzo modello di clustering. L'obiettivo è stato verificare ed osservare la tipologia automatica di suddivisione e raggruppamento messa in atto dal terzo modello di clustering.

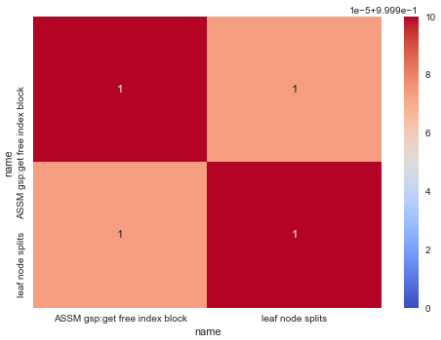


Figura 4.13. Secondo algoritmo DBSCAN: matrice di correlazione delle due metriche di performance del cluster con etichetta numero 0.

Il terzo modello di clustering utilizzato è stato l'algoritmo di clustering partizionale K-means. Tale algoritmo permette di raggruppare n elementi simili, in K clusters (sezione 2.1.3). L'algoritmo K-means utilizzato nella fase di progettazione, è stato importato dalla libreria *sklearn*. Il parametro definito è stato *n_clusters*, che rappresenta il numero di cluster da formare ed il numero di centroidi da generare. In questo lavoro di tesi, la scelta del parametro del modello è stata effettuata mediante l'utilizzo del metodo Elbow. Tale metodo fornisce un supporto per la selezione del numero ottimale di clusters. Per l'implementazione del metodo Elbow si è usufruito del visualizzatore *KElbowVisualizer*, della libreria Yellowbrick. Il metodo Elbow consiste nell'addestrare un modello con un intervallo di valori di k. In seguito, in base alla metrica stabilita si calcola un punteggio. In questo caso, la metrica scelta è stata la distorsione. Tale metrica calcola la somma delle distanze al quadrato di ciascun punto dal proprio centro assegnato. Il risultato è un grafico che rappresenta il punteggio di distorsione al variare del numero K di clusters. Il punto di flesso sulla curva, indicato spesso con il termine "gomito", esprime il punto in cui selezionare il numero k di cluster. Il *Kelbowvisualizer* permette, inoltre, di visualizzare anche la quantità di tempo per necessaria per allenare il modello di clustering al variare del parametro K. La curva viene rappresenta nel grafico con una linea di colore verde tratteggiata. Come mostrato in figura 4.14, nel progetto di tesi il parametro *n_clusters* dell'algoritmo K-means, è stato impostato al valore 3.

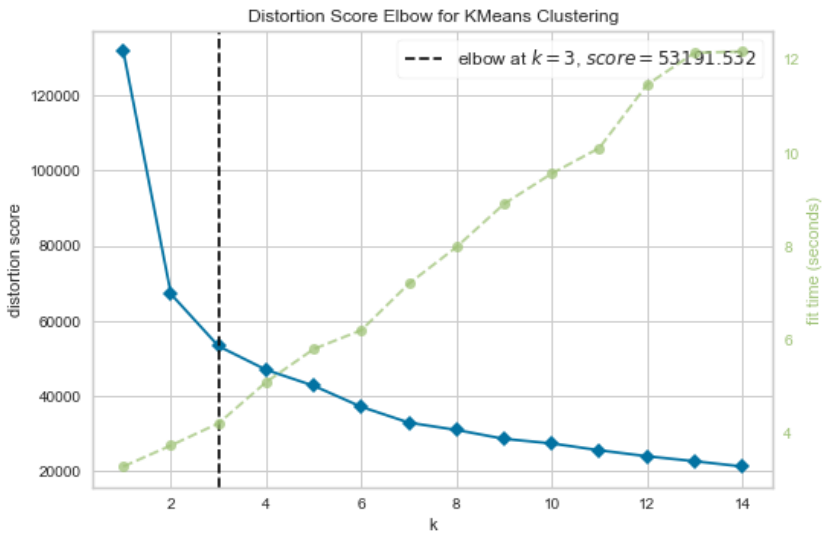


Figura 4.14. Metodo Elbow usato per la stima del parametro *n_clusters* dell'algoritmo K-means.

Etichetta cluster	Numero di metriche
-1	215
0	4
1	3

Tabella 4.3. Tabella riassuntiva dei risultati dell'algoritmo di clustering K-means.

Il passaggio successivo è stato l'esecuzione dell'algoritmo K-means. Sono stati stimati 2 cluster di metriche di performance ed un cluster di punti rumore (tabella 4.3). A seguire, è stata effettuta l'analisi della matrice di correlazione delle metriche di performance raccolte in ciascun cluster. Dall'analisi è emerso che l'algoritmo K-means è stato in grado di individuare un cluster di tre metriche di performance con elevate correlazioni (figura 4.16), ed un cluster di 4 metriche di performance con indici di correlazione non particolarmente elevati ed al di sotto della soglia impostata (figura 4.15). Questo comportamento era in parte atteso in quanto, a seguito delle analisi effettuate in precedenza, era emersa la difficoltà di individuare ulteriori cluster di metriche di performance con correlazioni elevate.

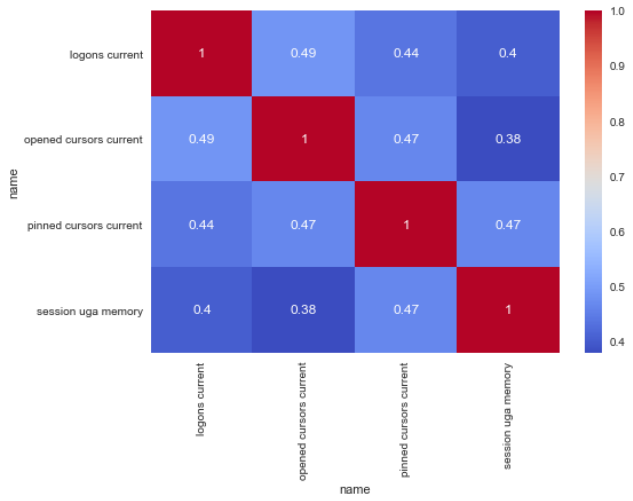


Figura 4.15. Algoritmo K-means: matrice di correlazione delle quattro metriche di performance del cluster con etichetta numero 0.

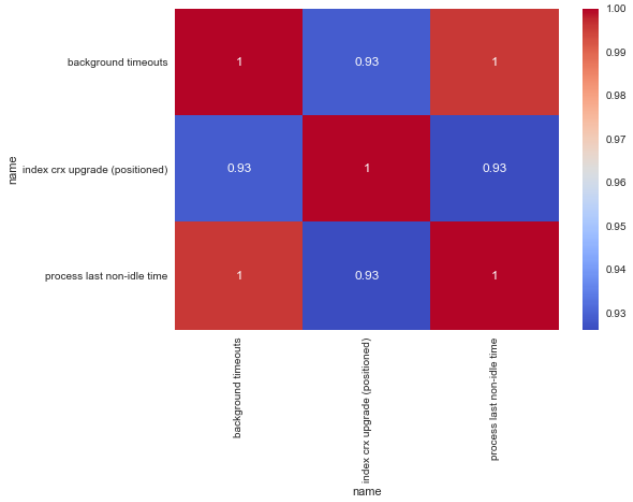


Figura 4.16. Algoritmo K-means: matrice di correlazione delle tre metriche di performance del cluster con etichetta numero 1.

Nel cluster con etichetta pari 1, la selezione è risultata molto semplice, dato che gli indici di correlazione tra le metriche di performance erano molto alti e ciascuno superava l'indice di soglia di 0.94. Di conseguenza, è stata selezionata la metrica di performance maggiormente correlata con le restanti due. Nel cluster con etichetta 0, invece, la selezione della metrica più significativa ha richiesto ulteriori analisi ed approfondimenti. Si è osservato l'andamento di ciascuna metrica di performance interna al cluster tramite Power BI, strumento utile per la visualizzazione dei dati (figure 4.17). L'analisi dei grafici delle metriche di performance ha evidenziato un andamento simile nel tempo delle metriche *opened cursors current* e *pinned cursors current*, pur presentando un indice di correlazione di 0.47. In seguito, è stata effettuata un'ulteriore indagine sulle metriche di performance, al fine di comprendere a pieno il significato associato a ciascuna metrica di performance. Esaminando l'elenco delle metriche e descrizioni della *gv$sysstat* [86], si è potuto appurare che la metrica *session uga memory* non assume alcun significato nella vista *gv$sysstat*. Conseguentemente, tale metrica è stata scartata, optando per una delle tre metriche restanti all'interno del cluster. Nello specifico, è selezionata la metrica con una maggiore correlazione con le altre metriche nel cluster.

Per le restanti 215 metriche di performance contenute nel cluster con etichetta -1, l'analisi della matrice di correlazione ha evidenziato nuovamente la presenza di un vasto numero di metriche con indici di correlazione che non superavano la soglia impostata. Ulteriori analisi di questo genere sui dati rimasti non avrebbero

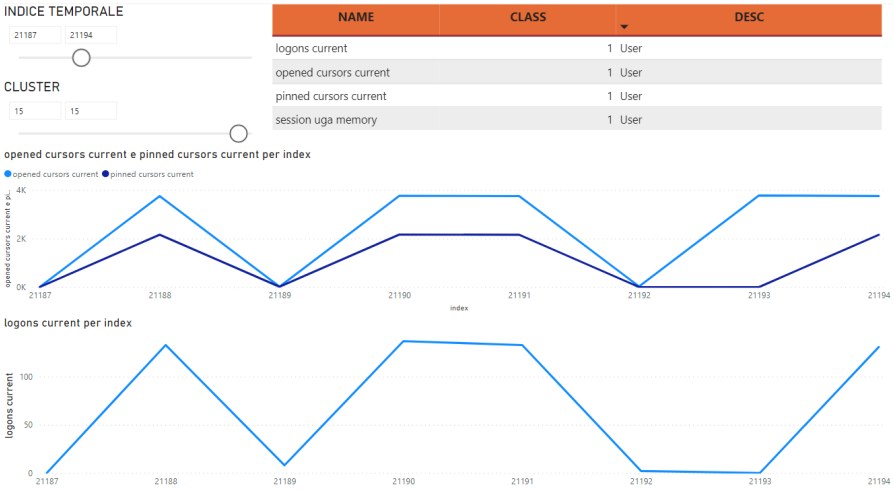
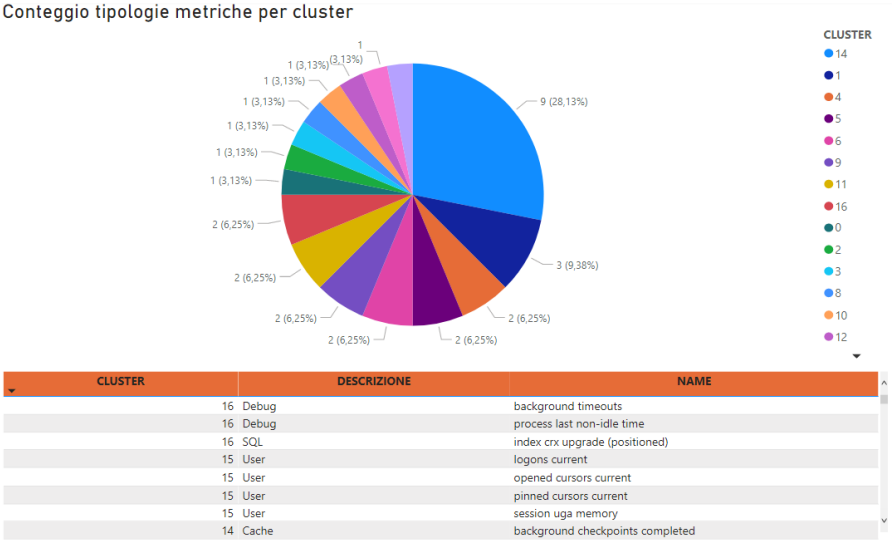


Figura 4.17. Rappresentazione dell'andamento delle metriche di performance appartenenti al cluster individuato dall'algoritmo K-means con etichetta 0. Schermata della dashboard creata su Power BI.

portato a risultati soddisfacenti. Si è proceduto all'eliminazione delle metriche di performance prive di significato per la vista *gv$sysstat*, indicati nel documento Oracle ufficiale [86]. Si è deciso, quindi, di fermare il processo di selezione ulteriore delle metriche di performance.
L'ultima analisi effettuata, propone una panoramica dei cluster ottenuti durante il processo di selezione ed in particolare lo studio della distribuzione delle diverse tipologie delle metriche in ciascun cluster. L'obiettivo è comprendere se le correlezioni delle metriche rinvenute corrispondono a correlazioni di metriche appartenenti a tipologie differenti, o ad una stessa tipologia. L'analisi è stata effettuata mediante Power BI (figura 4.18). Al fine di poter analizzare tutti i cluster ottenuti nelle diverse fasi del processo di selezione delle metriche, si è deciso di assegnare al primo cluster individuato (cluster del primo DBSCAN con etichetta 0), il valore 0 per poi incrementare il valore di ciascuna etichetta di 1. Pertanto, le etichette dei primi 7 cluster individuati dal primo algoritmo di DBSCAN applicato sono (0, 1, 2, 3, 4, 5, 6), quelle del secondo algoritmo DBSCAN sono rispettivamente (8, 9, 10, 11, 12, 13), mentre le etichette dell'algoritmo K-means sono (14, 15, 16). Il cluster con etichetta numero 14 identifica il cluster dell'algoritmo K-means dei punti rumore, ovvero contenete le restanti metriche non collocate in al alcun cluster. E' evidente che, complessivamente, ciascun cluster di metriche di performance presenti una o due tipologie di metrica. Ad esempio, nel cluster 9 le due metriche individuate appartengono a due tipologie diverse: *Debug* e *User*. Entrambe le metriche definiscono il tempo di utilizzo della cpu. Nello specifico, il



CLUSTER	DESCRIZIONE	NAME
16	Debug	background timeouts
16	Debug	process last non-idle time
16	SQL	index crx upgrade (positioned)
15	User	logons current
15	User	opened cursors current
15	User	pinned cursors current
15	User	session uga memory
14	Cache	background checkpoints completed

Figura 4.18. Rappresentazione della distribuzione delle tipologie delle metriche in ciascun cluster individuato durante il processo di selezione delle metriche di performance.

tempo di utilizzo della CPU all'avvio di una chiamata ed il tempo totale della CPU utilizzato dalle chiamate non utente. Gli algoritmi utilizzati per la selezione delle metriche di performance sono stati in grado di identificare patter simili, nonostante le due metriche appartengano a due tipologie differenti e modellino il tempo di utilizzo della CPU sotto due punti di vista diversi. Peraltro, l'analisi della matrice di correlazione, mostrava un indice di correlazione di 0,989. Osservazioni analoghe valgono per i restanti cluster di metriche di performance, escluso il cluster 14, in quanto raggruppa le metriche non collocate in alcun cluster.
In sintesi, le 2011 metriche di performance iniziali sono state ridotte a 408 mediante il processo di pulizia dei dati. Infine, attraverso il processo di selezione delle metriche si è arrivati a considerare esclusivamente 268 metriche.

4.5 Rilevazione di anomalie nelle metriche di performance

Il progetto proposto, successivamente alla fase di selezione delle metriche di performance, si è focalizzato sull'individuazione e confronto delle più opportune metodologie data driven di rilevazione delle anomalie. In particolare, sono stati implementati e confrontati due differenti metodi per la rilevazione delle anomalie (figura 4.19). Il primo si basa sull'utilizzo di modelli di serie temporali, mentre

il secondo sfrutta modelli di apprendimento automatico non supervisionato. Di seguito, viene definita la metodologia sfruttata nel progetto di tesi, seguendo una descrizione precisa e sequenziale.

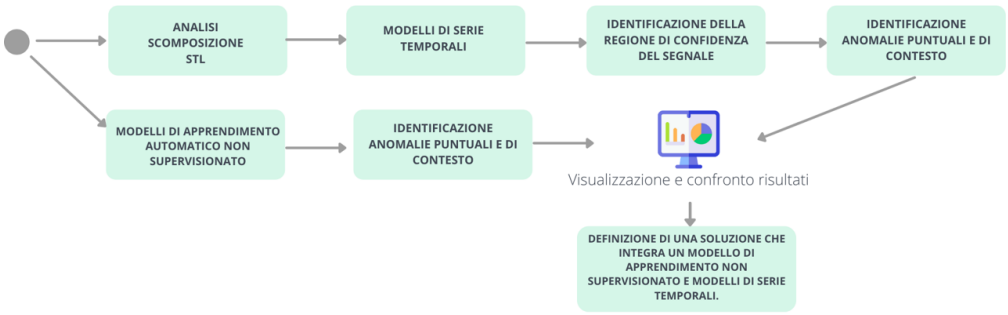


Figura 4.19. Processo di identificazione di anomalie nelle metriche di performance.

In questo lavoro di tesi, i dati analizzati erano delle serie temporali e non presentavano etichettatura. La prima metodologia proposta si basa, per l'appunto, su modelli di serie temporali. La serie temporale è una sequenza di dati composta da osservazioni registrate in punti contigui equidistanti nel tempo [39]. In questa fase progettuale, si è analizzato l'andamento di ciascuna metrica di performance, al fine di comprendere le caratteristiche della stessa. Si è proceduto ad una scomposizione delle serie temporali, mediante il metodo STL. Il metodo STL, Seasonal and Trend decomposition using Loess (sottosezione 2.2.1), risulta uno dei metodi più efficaci e versatili per effettuare la decomposizione delle serie temporali. L'obiettivo è quello di separare una serie temporale nelle sue componenti: trend, stagionalità e residuo. La scomposizione della serie temporale è avvenuta mediante l'utilizzo della funzione *seasonal_decompose*, importata dalla libreria *statsmodels*. I parametri definiti nella funzione *seasonal_decompose* sono stati: la time series, il modello ed il periodo. Il modello stabilisce la tipologia di scomposizione che si vuole effettuare, mentre il periodo, come il nome stesso riferisce, il periodo della serie. Si è deciso di effettuare una scomposizione di tipo additivo, ed il periodo è stato impostato a 1440. Il valore del periodo corrisponde ad 1 giorno. Le metriche erano state campionate una volta al minuto ($1 \cdot 60 \cdot 24 = 1440$). Ciascuna serie temporale è stata scomposta e le sue componenti sono state analizzate. Le metriche presentavano una stagionalità ben definita, come emerso dall'analisi delle componenti delle serie temporali (figura 4.23).

Queste analisi hanno consentito l'utilizzo di metodi di serie temporali applicati a

Figura 4.20. Decomposizione STL della metrica Messages sent.

segnali stagionali. Sono stati utilizzati i modelli ARIMA, Auto Regressive Integrated Moving Average. Tali modelli, rappresentano uno degli approcci maggiormente utilizzati per la previsione di serie temporali. I processi di previsione si basano

sulla stima di una previsione futura di punti di dato, in relazione alla valutazione di punti nel passato. Il modello ARIMA è una generalizzazione di un modello autoregressivo a media mobile. In particolare, è ottenuto dalla combinazione di un modello AR (AutoRe-gressive model) ed uno MA (Moving Average model), ed un processo di differenziazione (sezione 2.2.2).
Un modello ARIMA è caratterizzato da 3 termini: p, d, q. p è l'ordine del termine AR, q è l'ordine del termine MA, mentre d è il numero di differenze richiesto per rendere stazionarie le serie temporali. Ulteriori parametri si introducono nel caso in cui la serie temporale presenti modelli stagionali. Il modello ARIMA, in tal caso, diventa SARIMA, Seasonal ARIMA.
Il primo passo per costruire un modello ARIMA è rendere la serie temporale stazionaria. La presenza o assenza di stazionarità in una serie temporale è stata verificata mediante l'utilizzo di un test statistico, noto come Augmented Dickey Fuller test (ADF test, sezione 2.2.2). Si tratta di un test statistico di significatività. L'ipotesi nulla nell'ADF afferma la presenza di una tendenza stocastica in un campione di serie temporali. L'ipotesi alternativa, invece, afferma la presenza di stazionarietà [52]. Per implementare il test ADF, nel progetto di tesi, si è usufruito del pacchetto *statsmodel*. In *statsmodels.tsa.stattools*, la funzione *adfuller()*, fornisce un'implementazione affidabile del test ADF. Tale funzione restituisce in output: il *p-value*, il valore della statistica del test, il numero di ritardi (lag) considerati nella prova, e i *critical values*. Nel caso in cui la statistica del test risulti inferiore rispetto al valore critico mostrato, l'ipotesi nulla è rifiutata. In tal caso, la serie temporale viene stimata come serie temporale stazionaria. Nella funzione *adfuller()* può essere inserito un parametro opzionale. Si può definire il numero di *lag* da considerare durante l'esecuzione della regressione OLS (Ordinary Least Squares). Il suo valore di default è $12 * (Nobs/100)^{1/4}$, dove Nobs è il numero di osservazioni nella serie. Nel lavoro di tesi, si è deciso di permettere all'algoritmo di calcolare il numero ottimale di *lag* in modo iterativo. Si è impostato, quindi, l'Autolag='AIC'. In questo modo la funzione *adfuller()* sceglierà il numero di lag che produce l'AIC più basso. Il criterio informativo di Akaike (AIC - Akaike information criterion) stima l'errore di previsione e la qualità di ciascun modello statistico. Dara una serie di modelli per un insieme di dati, l'AIC è in grado di stimare la qualità di ciascun modello in relazione agli altri modelli considerati. Pertanto, l'AIC rappresenta un mezzo per la selezione di modelli. La qualità di un modello, per l'AIC, viene stimata in base alla quantità di informazioni che si perdono utilizzando il modello per rappresentare i dati. Ne consegue che al diminuire delle informazioni perse aumenta la qualità del modello considerato [88].
Dai test di Augmented Dickey Fuller è emerso che la maggioranza delle serie temporali analizzate mostrava *p-value* inferiori al livello di significatività di 0.05, e un valore della statistica ADF inferiore a qualsiasi valore critico. Si è respinta l'ipotesi nulla e si sono considerate le serie temporali come stazionarie. Sono state

esaminate anche le *rolling statistics*, o statistiche mobili. Questo metodo permette di definire la stazionarità di una serie temporale in base ad una rappresentazione grafica dei dati e delle statistiche mobili. Sono state calcolate la media mobile e la deviazione standard mobile. Si è considerata una finestra temporale scorrevole di dimensione k. Per ciascun finestra temporale si è calcolato il valore della media e della deviazione standard, sfruttando la funzione dataframe.rolling() di Pandas. Tale funzione consente di creare una finestra di scorrimento di dimensione k e di eseguire alcune operazioni matematiche su di esse. E' stata usata la funzione dataframe.rolling(window=k).mean() per calcolare la media mobile e dataframe.rolling(window=k).std() per il calcolo della deviazione standard mobile, con k dimensione della finestra.
Alcuni grafici evidenziano un andamento delle *rolling statistics* pressochè costante (4.21), confermando una corrispondenza con i risultati ottenuti dal test ADF. Dai

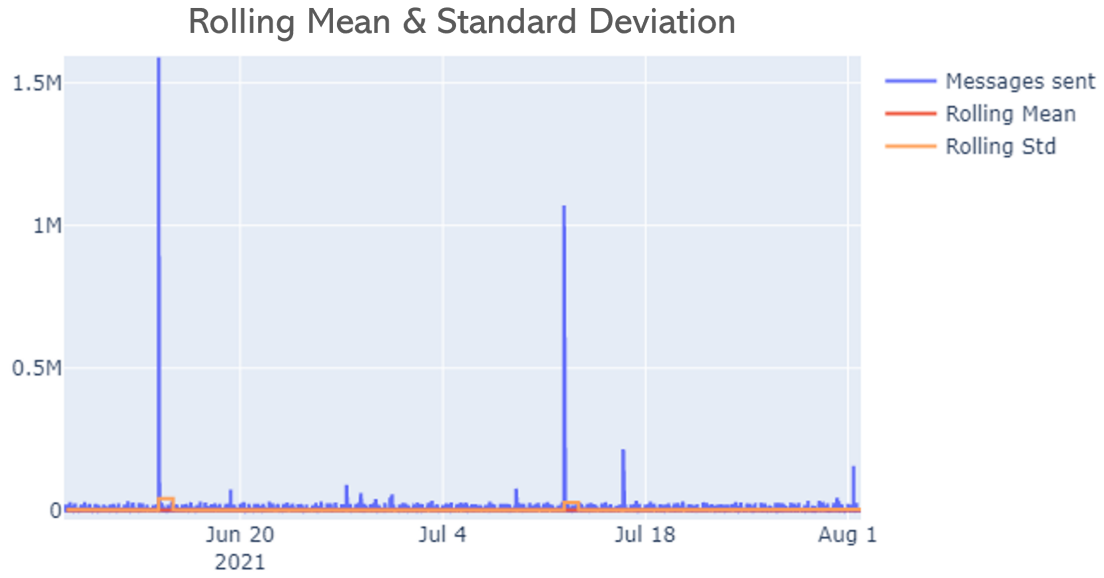


Figura 4.21. Rappresentazione grafica della *rolling mean e standard deviation* della metrica messages sent.

test eseguiti sulle metriche di performance sono emerse alcune serie temporali prive di stazionarità. L'approccio più comune per rendere una serie temporale stazionaria è eseguire un processo di differenziazione (sezione 2.2.2). La serie temporale di partenza viene trasformata in una nuova serie, i cui valori sono calcolati come differenza tra osservazioni consecutive . La differenziazione ha come obiettivo, quello di associare una varianza ed una media stabili nel tempo ad una serie temporale. A volte, a seconda della complessità della serie, può essere necessaria più di una

differenza. Il parametro d di ARIMA, è il numero minimo di differenze necessarie per rendere stazionaria la serie temporale. Se la serie temporale è già stazionaria allora d sarà impostato a 0. Al fine di poter individuare e selezionare il numero idoneo di differenze da applicare ad una serie temporale, e gli ulteriori termini del modello ARIMA, si è deciso di applicare il modello Auto ARIMA per ciascuna metrica di performance. Da questo punto in poi, l'analisi e applicazione di modelli di serie temporali è stata effettuata per ciascuna metrica di performance. L' Auto ARIMA è stato importato da *pmdarima.arima*. Il modello Auto Arima consente l'applicazione di vari modelli ARIMA, ciascuno dei quali caratterizzato da una diversa combinazione dei parametri. In seguito, per ciascun modello viene calcolato il valore dei parametro AIC, Akaike information criterion. Il modello che assume il valore più basso di AIC, viene selezionato, determinando la lista più adeguata dei parametri del modello. In seguito, data la stagionalità nei segnali, è stato applicato un modello SARIMAX, impostando i parametri precedentemente individuati. Il modello è stato addestrato e successivamente si sono ottenute le previsioni e gli intervalli di confidenza delle previsioni. In particolare, al risultato ottenuto dalla funzione *get_prediction* è stata applicata la funzione *predicted_mean*() per individuare i valori di previsione, mentre la funzione *conf_int*() è stata usata per individuare gli intervalli di confidenza. In seguito, è stato impostato il valore massimo della regione di confidenza come valore di soglia. Tale valore di soglia è stato utile per la definizione dei punti anomali.
In questo lavoro di tesi, sono stati considerati punti anomali tutti quei valori al di fuori della regione di confidenza e che superano il valore di soglia impostato. Si è ipotizzato, infatti, che i valori anomali si discostino molto dai valori che la serie temporale generalmente assume e che sono raccolti all'interno della regione di confidenza.

Il secondo metodo per l'identificazione di punti anomali, utilizzato nel seguente lavoro di tesi, è stato un metodo di tipo non supervisionato. I dati esaminati, infatti, non presentano etichettatura. Si è utilizzato l'algoritmo Isolation Forest, che si focalizza esclusivamente sull'identificazione esplicita delle anomalie, e non sulla caratterizzazione del comportamento normale di una serie temporale. L'Isolation Forest utilizza degli alberi di decisione binari, noti come Itree, per isolare le anomalie. In particolare, il rilevamento delle anomalie mediante Iforest, consiste in una procedura caratterizzata da due fasi principali. Durante la prima fase, i dati di training vengono utilizzati per costruire gli alberi di isolamento (Itree). Nella seconda fase, invece, ciascun dato nel set di testing passa attraverso gli alberi di isolamento, precedentemente creati, ed ad ognuno viene assegnato un punteggio di anomalia, noto come *anomaly score*. A seconda dei valori che assume l'anomaly score, permette di identificare, in modo esplicito, le anomalie. Se il punteggio

associato ad un punto di dato è superiore ad una soglia predefinita, viene contrassegnato come punto anomalo (sezione 2.2.3).
Il modello dell'Isolation Forest è stato importato dalla libreria *sklearn*. I parametri definiti sono stati *n_estimators*, *max_samples*, *contamination*, *n_jobs* e *random_state*. Il *n_estimator* definisce il numero di stimatori di base, ovvero di alberi da utilizzare, mentre il *max_samples* rappresenta il numero di campioni da estrarre da X per addestrare ciascun stimatore di base. Il parametro *contamination*, invece, definisce la quantità di contaminazione del set di dati, ovvero la proporzione di valori anomali nel set di dati. Tale parametro può assumere il valore "auto", impostando la soglia come nel paper originale dell'Isolation Forest, oppure un valore incluso in un intervallo da (0, 0.5]. Quest'ultimo è un parametro importante e difficile da impostare, che inficia sull'efficacia del modello. Di fatti, si deve scegliere un valore opportuno di contaminazione, per evitare il non rilevamento delle reali anomalie (se, ad esempio, il valore di contaminazione impostato è troppo basso), o la segnalazione di false anomalie, che corrispondono quindi a punti normali (se, ad esempio, il valore di contaminazione impostato è troppo alto). Nel lavoro di tesi, il valore del parametro di contaminazione è stato stabilito in modo empirico. Non è stato possibile definire un unico valore di contaminazione adatto per tutte le metriche di performance. Per ciascuna serie temporale sono stati valutati diversi valori di contaminazione e con il supporto dei dipendenti aziendali dell'area di Infrastruttura è stato possibile determinare un valore unico per ciascuna metrica. Il parametro *n_jobs* è stato impostato a -1, in modo tale da poter utilizzare tutti i processori a disposizione ed effettuare velocemente le operazioni di fit e predict per le diverse metriche di performance. L'ultimo parametro definito è stato il *random_state*. Il suo valore controlla la pseudo-casualità della selezione della caratteristica e dei valori di divisione per ogni fase di ramificazione di ciascun albero.
L'Isolation Tree è stato allenato e per ciascuna metrica di performance sono state individuate le etichette. Tali etichette definiscono ciascun punto della serie temporale come punto anomalo o non anomalo. Nello specifico, viene assegnata l'etichetta con valore 1 ai punti non anomali, mentre il valore -1 ai punti anomali. In seguito, è stato calcolato ed analizzato il punteggio di anomalia (*anomaly score*) medio di ciascun punto ottenuto dai diversi alberi di base. A tal proposito si è utilizzata la funzione *decision_function()*. In generale, più il punteggio di anomalia, associato ad un'osservazione è basso, più è l'osservazione risulta anomala. Punteggi negativi rappresentano valori anomali, mentre punteggi positivi rappresentano punti normali. La misura della normalità di un'osservazione, per un singolo albero, dipende dalla profondita in cui è collocato il nodo foglio dell'albero che contiene l'osservazione, ovvero è equivalente al numero di divisioni necessarie per isolare un punto. Generalmente, le anomalie si individuano in nodi più vicini al nodo radice dell'albero, comportando un percorso più breve, in quanto più facilmente isolabili

rispetto agli altri punti. Un punto non anomalo, invece, è più difficile da isolare, richiede più divisioni, e spesso si trova all'estremità più profonda dell'albero.

In entrambe le metodologie proposte, è stato analizzato il grafico delle diverse metriche di performance mettendo in evidenza i vari punti anomali identificati. Dall'analisi dei risultati è emerso che entrambe le metodologie di rilevamento delle anomalie presentavano dei limiti, discussi nel capitolo 5. È emerso, inoltre, che non vi è un metodo nettamente più performante di un altro. Questo a causa delle caratteristiche intrinseche del segnale, che possono inficiare l'efficienza del metodo stesso. Conseguentemente, è stato proposto un modello combinato dei due metodi, che potesse sfruttare i vantaggi di entrambi. Tale modello consiste nell'integrazione e combinazione dei due metodi precedentemente descritti, in modo tale da trarre vantaggio da ciascuno dei due e cercare di rimediare ai limiti identificati in ciascuna soluzione proposta. Per alcune metriche di performance, invece, si è preferito considerare esclusivamente l'algoritmo Isolation Forest perchè le anomalie identificate dalla prima metodologia proposta, basata su modelli di serie temporali, erano molteplici e incongruenti.

In una fase successiva, l'analisi dei risultati sperimentali è stata affiancata da degli esperti dell'area di Infrastruttura dell'azienda. Dalle considerazione esposte, è risultato che in ambito aziendale le anomalie puntuali non sempre sono significative. In alcuni casi, si presentano dei valori anomali isolati, ovvero preceduti e seguiti da valori normali. Tale punto non è particolarmente rilevante per i dipendenti aziendali dell'area di Infrastruttura. Questo perchè se in un istante temporale si registra un valore anomalo che può indicare una situazione critica, e negli istanti successivi si registrano valori normali, non si è più in una situazione critica (figura 4.22). In questo caso il problema viene risolto senza l'intervento del personale.
Si è deciso di individuare ed esaminare delle anomalie di contesto, dipendenti dal tempo, in ciascuna metrica di performance. Sono state identificate delle sequenze di punti anomali in un intervallo di tempo definito, al fine di rilevare situazioni critiche che durano nel tempo e che potrebbero richiedere un intervento da parte del personale qualificato.
Il metodo utilizzato nel progetto di tesi consiste nel definire una finestra temporale scorrevole di 5 minuti e nell'identificare le anomalie di una metrica presenti nella finestra. Si sfruttano le due metodologie precedentemente descritte per il rilevamento delle anomalie. In seguito alla registrazione di un valore anomalo, si verifica se nella finestra temporale di 5 minuti viene registrata un'ulteriore anomalia. In tal caso, le due o più anomalie registrate definiscono un'anomalia di contesto.

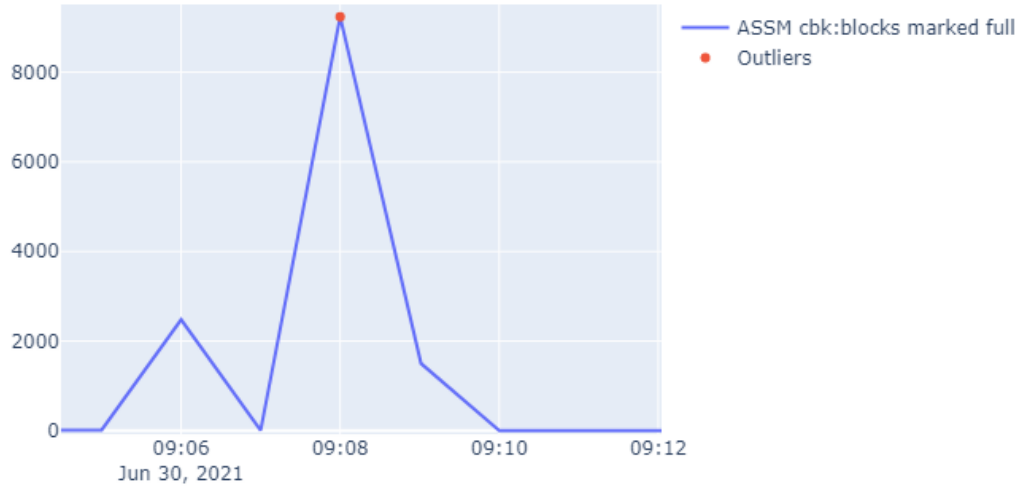


Figura 4.22. Rappresentazione grafica di un'anomalia puntuale nella metrica di performance ASSM cbk:blocks marked full.

La dimensione della finestra temporale è stata scelta in seguito ad una analisi approfondita del contesto e delle esigenze aziendali con gli esperti di dominio. Attualmente l'area di Infrastruttura Tecnologica di Mediamente Consulting, si affida alla piattaforma Oracle Enterprise Manager, la quale offre una soluzione completa di monitoraggio e gestione per Oracle Database e Engineered Systems distribuiti sul cloud o data centers dei clienti. In aggiunta, i dipendenti dell'area IT sfruttano una dashboard da loro personalizzata per permettere la visualizzazione simultanea dei diversi sistemi dei clienti. L'attuale campionamento delle metriche di performance e visualizzazione delle stesse avviene ogni 15 minuti. In questo lavoro di tesi, le metriche di performance del database sono state campionate una volta al minuto, e tramite un intervallo temporale di cinque minuti è possibile identificare dei valori anomali che durano nel tempo e che potenzialmente indicano una situazione critica per il sistema. L'intervallo di tempo stabilito è stato stimato al fine di minimizzare il più possibile il tempo di intervento ed aumentare la reattività.

In conclusione, l'architettura del progetto di tesi può essere riassunta tramite il seguente schema:

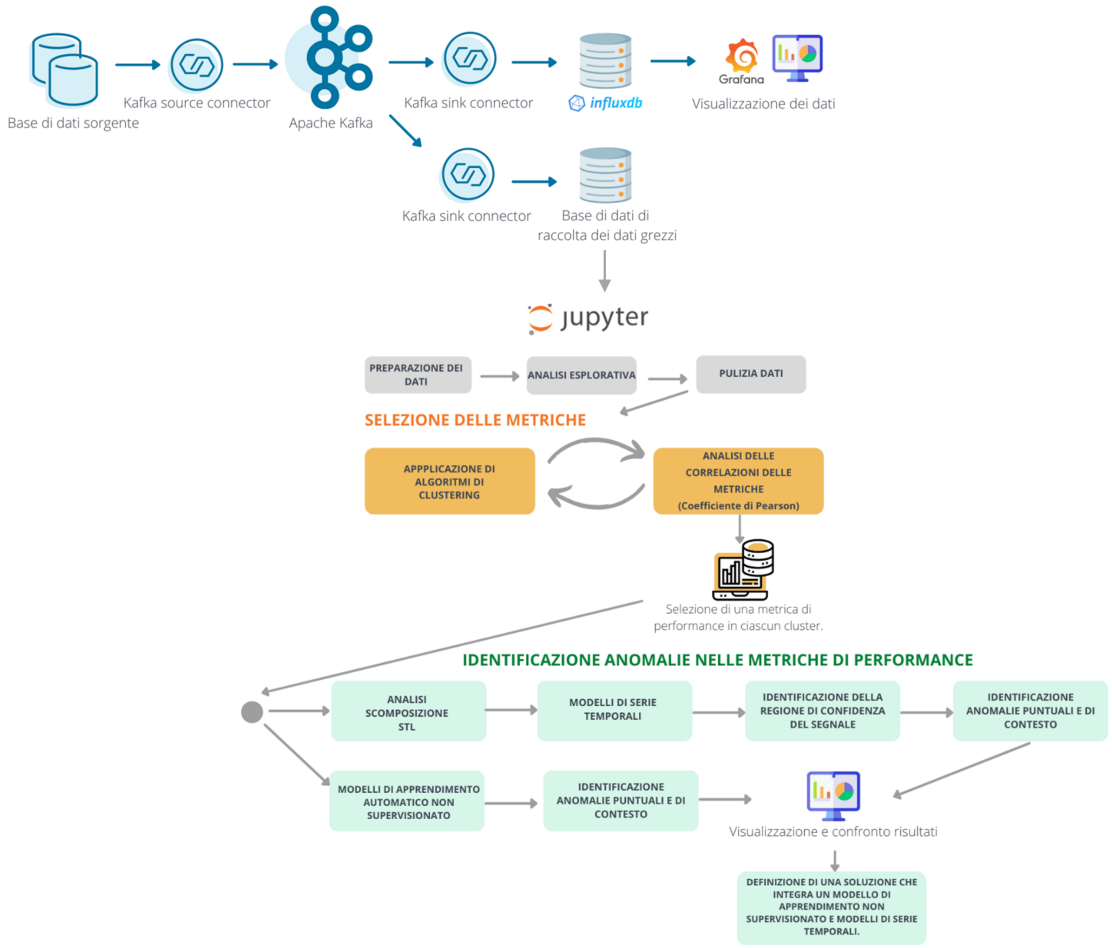


Figura 4.23. Architettura completa del progetto di tesi.

Capitolo 5

Risultati sperimentali

In questo capitolo vengono presentati i risultati sperimentali del processo di identificazione di anomalie nelle metriche di performance. Si mostrano e si confrontano i risultati ottenuti da ciascuna metodologia proposta. Si riportano le motivazioni che hanno determinato l'applicazione della terza metodologia e viene fornita una discussione critica e dettagliata dei risultati ottenuti. Le osservazioni relative ai risultati del processo di selezione delle metriche di performance sono state riportate nella sezione 4.4.

Nel progetto di tesi proposto sono stati implementati due differenti metodi per la rilevazione delle anomalie. Il primo si basa sull'utilizzo di modelli di serie temporali, mentre il secondo sfrutta modelli di apprendimento automatico non supervisionato (sezione 4.5).
Per il primo metodo implementato sono stati usati modelli ARIMA. Le anomalie sono state identificate sfruttando un valore soglia ottenuto come valore massimo della regione di confidenza della previsione della serie temporale in esame. I risultati ottenuti, per alcune metriche di performance sono stati soddisfacenti, in quanto la metodologia utilizzata è stata in grado di rilevare correttamente i valori che si discostano molto dai restanti valori nell'insieme dei dati. Il grafico 5.2, evidenzia l'andamento della metrica di performance *ASSM bg:slave fix state* in blu, la regione di confidenza della previsione della serie temporale in verde, e le anomalie identificate in rosso. In totale sono stati individuati 7 valori anomali. È chiaro come le anomalie individuate, tramite la prima metodologia proposta, risultino dei valori che si discostano molto dai restanti valori assunti dalla metrica *ASSM bg:slave fix state* (tabella 5.2). Le stesse osservazioni valgono per la metrica *data blocks consistent reads - undo records applied* (grafico 5.1, tabella 5.1), in cui sono stati individuati esclusivamente due valori anomali. Nonostante ciò, sono state individuate delle metriche di performance in cui la metodologia utilizzata

non ha garantito risultati rispondenti alle aspettative, in quanto ha rilevato anomalie molteplici e ripetitive. Il confronto con gli esperti dell'area di Infrastruttura dell'azienda, ha confermato la poca attendibilità dei risultati per quelle particolari metriche di performance. Il problema risiede nel fatto che tale metodologia è basata su modelli di previsione ed intervalli di confidenza di tale previsione. Conseguentemente, se i modelli ARIMA presentano una difficoltà nella previsione della serie temporale, tale difficoltà viene estesa anche alla fase successiva di identificazione delle anomalia, in quanto il valore di soglia stabilito non sarà corretto. A titolo d'esempio vengono mostrati i grafici le metriche *ASSM rsv:clear reserve* e *Cached Commit SCN referenced* nella quale la metodologia basata su ARIMA non risulta efficace e performante (grafico 5.3 e grafico 5.4).

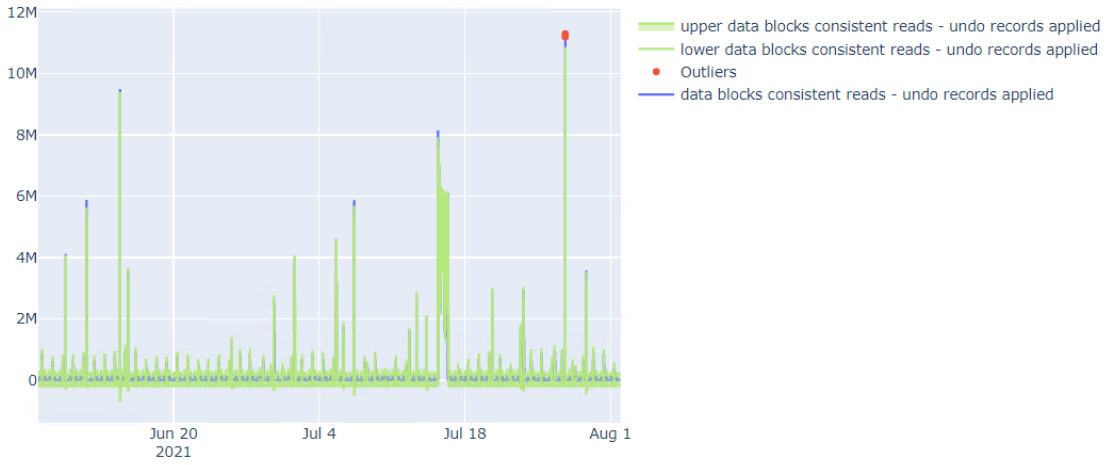


Figura 5.1. Anomalie identificate dalla prima metodologia, basata su modelli di serie temporali, per la metrica data blocks consistent reads - undo records applied.

Sample_time	**Data blocks consistent reads - undo records applied**
2021-07-27 16:03:01	11294171
2021-07-27 16:04:00	11197013

Tabella 5.1. Tabella riassuntiva delle anomalie identificate dalla prima metodologia, basata su modelli di serie temporali, per la metrica data blocks consistent reads - undo records applied.

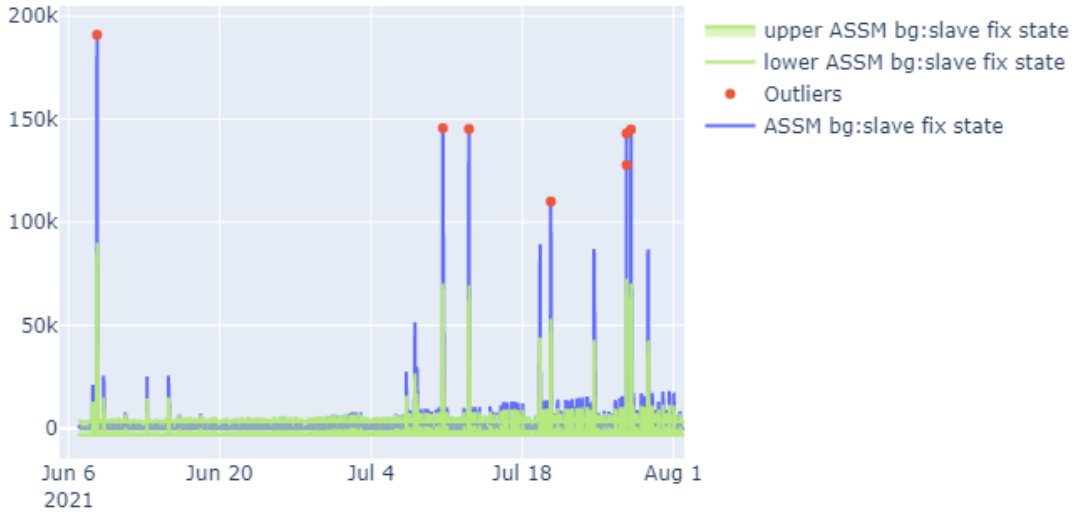


Figura 5.2. Anomalie identificate dalla prima metodologia, basata su modelli di serie temporali, per la metrica ASSM bg:slave fix state.

Sample_time	**ASSM bg:slave fix state**
2021-06-08 15:19:00	190973
2021-07-10 15:37:01	145677
2021-07-13 01:30:00	145246
2021-07-20 15:31:00	110066
2021-07-27 15:51:00	127811
2021-07-27 15:52:00	143151
2021-07-28 01:32:00	145006

Tabella 5.2. Tabella riassuntiva delle anomalie identificate dalla prima metodologia, basata su modelli di serie temporali, per la metrica ASSM bg:slave fix state.

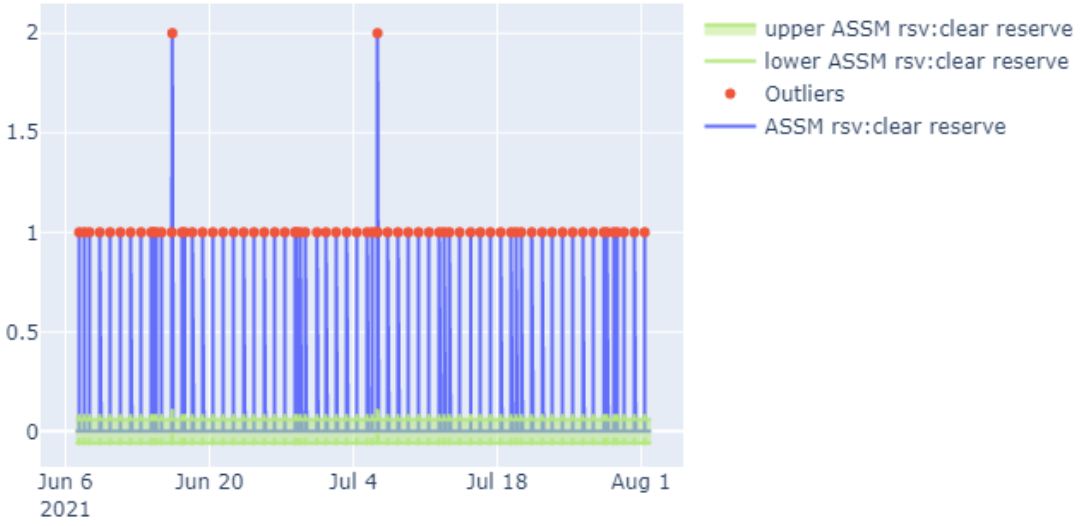


Figura 5.3. Anomalie identificate dalla prima metodologia, basata su modelli di serie temporali, per la metrica ASSM rsv:clear reserve.

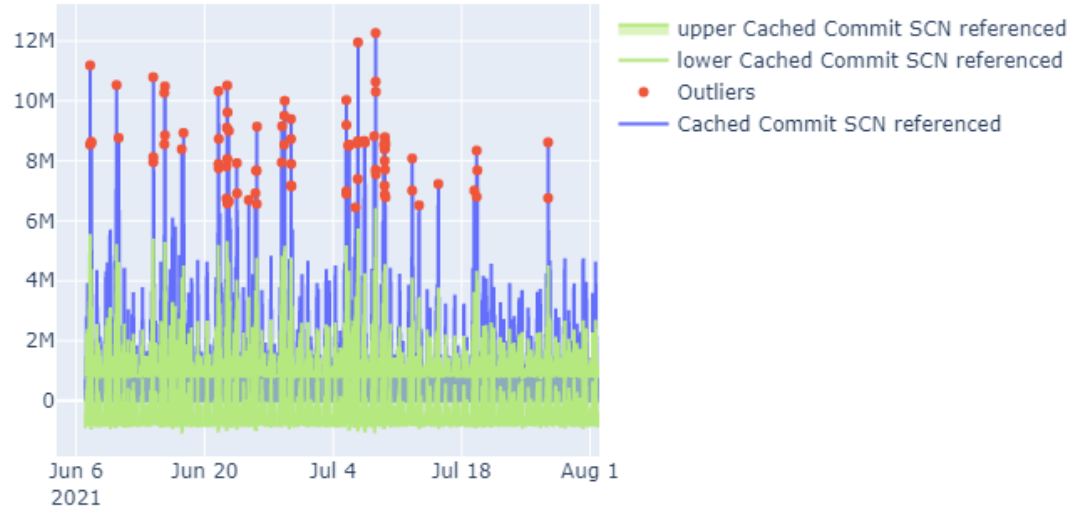


Figura 5.4. Anomalie identificate dalla prima metodologia, basata su modelli di serie temporali, per la metrica Cached Commit SCN referenced.

Il secondo metodo per l'identificazione di punti anomali, utilizzato nel seguente lavoro di tesi, è stato l'algoritmo Isolation Forest (sezione 4.5). I risultati ottenuti sono stati alquanto soddisfacenti, in quanto, anche in questo caso, la metodologia utilizzata è stata in grado di rilevare correttamente i valori che si discostano molto dai restanti valori nell'insieme dei dati. Il grafico 5.5, evidenzia l'andamento della metrica di performance *ASSM rsv:clear reserve* in blu, e le anomalie identificate in rosso. È evidente come le uniche due anomalie individuate dall'algoritmo risultino isolate rispetto al resto dei dati (tabella 5.3). I risultati sono coerenti e conformi con l'ipotesi. Le stesse osservazioni valgono per la metrica *Cached Commit SCN referenced* (grafico 5.6, tabella 5.4), in cui sono stati individuati esclusivamente quattro valori anomali.

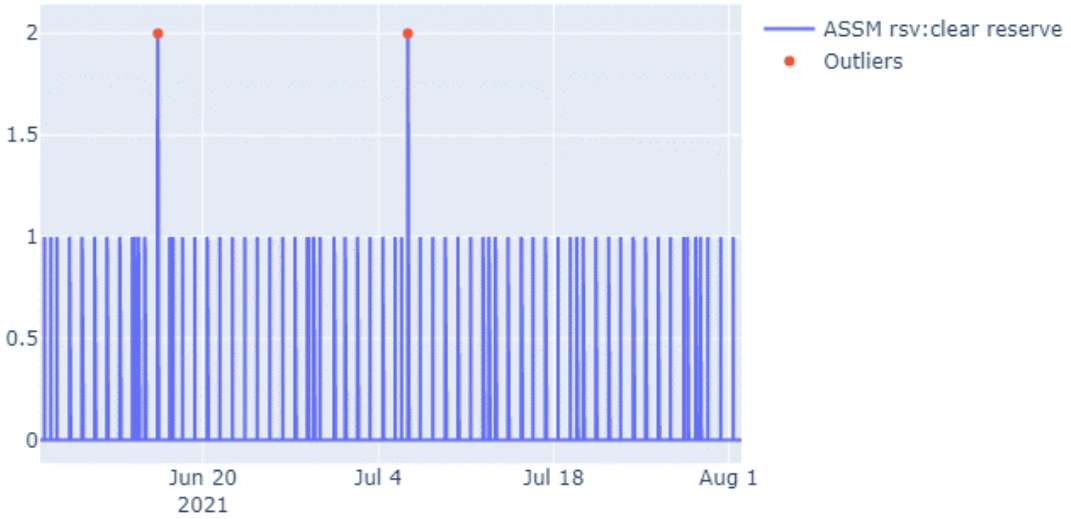


Figura 5.5. Anomalie identificate dalla seconda metodologia, basata sull'algoritmo Isolation Forest, per la metrica ASSM rsv:clear reserve.

Sample_time	ASSM rsv:clear reserve	Scores	Anomaly_label
2021-06-16 09:23:00	2	-0,000385388	-1
2021-07-06 08:27:01	2	-0,000385388	-1

Tabella 5.3. Tabella riassuntiva delle anomalie identificate dalla seconda metodologia, basata sull'algoritmo Isolation Forest, per la metrica ASSM bg:slave fix state.

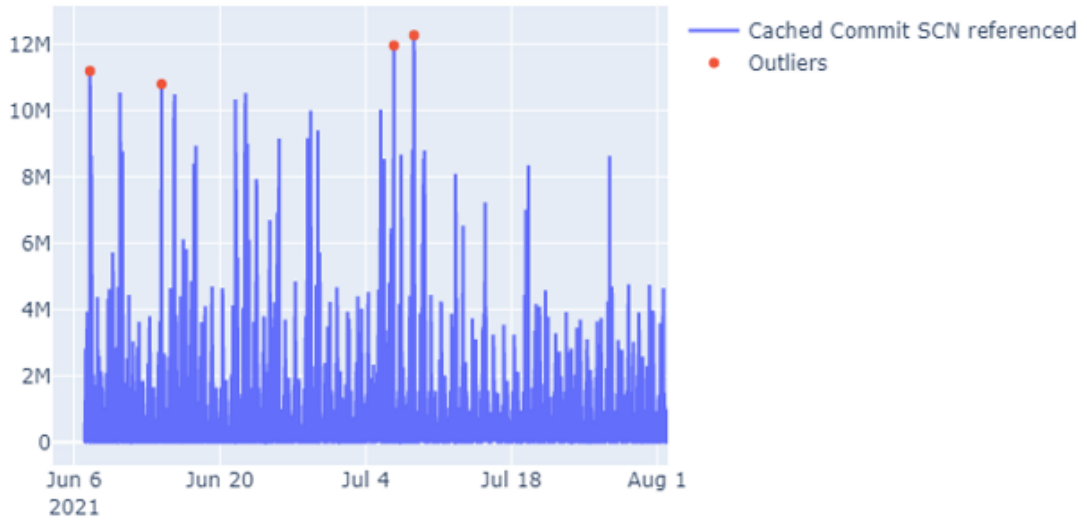


Figura 5.6. Anomalie identificate dalla seconda metodologia, basata sull'algoritmo Isolation Forest, per la metrica Cached Commit SCN referenced.

Sample_time	Cached Commit SCN referenced	Scores	Anomaly_label
2021-06-07 12:33:01	11185517	-0.001773	-1
2021-06-14 09:37:00	10791113	-0.001773	-1
2021-07-06 17:05:00	11952607	-0.001773	-1
2021-07-08 15:13:00	12263704	-0.001773	-1

Tabella 5.4.

Durante il processo di analisi dei risultati primari ottenuti dall'Isolation Forest, è emerso che l'efficienza del modello dipenda strettamente dal parametro di contaminazione che viene selezionato per la metrica di performance. Di fatti, non è stato possibile individuare un parametro di contaminazione univico per tutte le metriche di perfomance in considerazione, ma è stata richiesta un'analisi accurata e specifica per ciascuna metrica di performance. In particolare, per ciascuna metrica di performance sono stati sperimentati valori di contaminazione in un intervallo da (0, 0.5], al fine di individuare un numero idoneo e coerente di anomalie. Si è richiesto, quindi, un dispendio di tempo per la ricerca della stima più corretta del valore di contaminazione, indispensabile per esiti finali.
Confrontando i risultati e le coppie di grafici (5.5 e 5.3) e (5.6 e 5.4) rispettivamente delle metriche ASSM rsv:clear reserve e Cached Commit SCN referenced, risulta

evidente come la metodologia basata su modelli di serie temporali presenti delle performance inferiori rispetto al modello basato sull'algoritmo Isolation Forest. Questo accade ogni qualvolta la metodologia basata sui modelli ARIMA non è in grado di effettuare una buona previsione del segnale. Conseguentemente, a causa di alcune caratteristiche intriseche della serie temporale, per alcune metriche di performance, l'algoritmo Isolation Forest riesce a individuare correttamente quei punti anomali che risultano maggiormente isolati rispetto agli altri valori nella serie temporale. Per le restanti metriche, invece, l'analisi dei risultati sottolinea che non vi è un metodo nettamente più performante di un altro. Questo proprio a causa delle caratteristiche intrinseche del segnale, che possono inficiare l'efficienza del metodo stesso. Conseguentemente, è stato proposto un modello combinato dei due metodi. Tale modello consiste nell'integrazione e combinazione dei due metodi precedentemente descritti, in modo tale da trarre vantaggio da ciascuna soluzione. In particolare, sono stati uniti i risultati provenienti dalle due metodologie proposte. Per alcune metriche di performance i risultati ottenuti dalle due metodologie coincidevano. Per altre metriche di performance, invece, i risultati ottenuti dalle due soluzioni si discostavano di poco. Conseguemente, si è deciso di considerare tutte le anomalie individuate, in modo tale da effettuare ulteriori approfondimenti in futuro (esempio in figura 5.7).

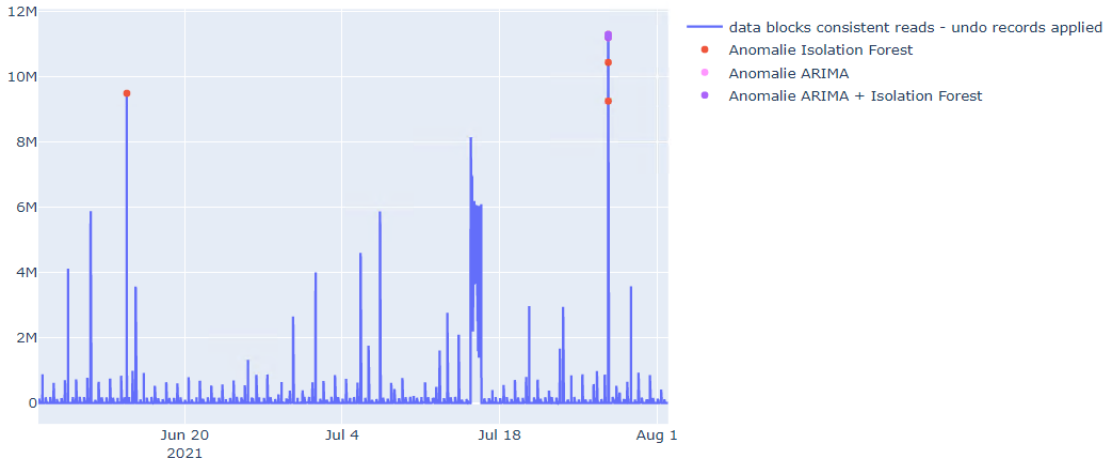


Figura 5.7. Anomalie identificate dalla terza metodologia, basata sulla combinazione di modelli di serie temporali e Isolation Forest, per la metrica data blocks consistent reads - undo records applied.

Nel grafico 5.7, vengono mostrate le anomalie individuate nella metrica di performance data blocks consistent reads - undo records applied. I due valori anomali rilevati dalla prima metodologia basata su ARIMA, sono stati individuati anche

dalla seconda metodologia basata sull'algoritmo Isolation Forest. In aggiunta, l'Isolation Forest identifica ulteriori tre anomalie. Il parametro di contaminazione scelto per la specifica metrica di performance è stato 0.00005, ed in questo caso corrisponde al valore minimo assegnabile per individuare dei punti anomali nella serie temporale. Conseguentemente, i valori anomali rilevati dall'Isolation Forest corrispondono ai minimi valori che tale algoritmo è in grado di rilevare.

Infine, l'analisi dei risultati sperimentali è stata affiancata da degli esperti dell'area di Infrastruttura dell'azienda. Dalle considerazione esposte, è risultato che in ambito aziendale le anomalie puntuali non sempre sono significative (sezione 4.5). Nel progetto di tesi, sono state individuate ed esaminate delle anomalie di contesto, dipendenti dal tempo, in ciascuna metrica di performance (figura 5.8 e figura 5.9). Sono state identificate delle sequenze di punti anomali in un intervallo di tempo di cinque minuti, al fine di rilevare situazioni critiche che durano nel tempo e che potrebbero richiedere un intervento da parte del personale qualificato. Dall'analisi dei grafici, nell'intervallo temporale stabilito, i valori anomali individuati tendono ad aumentare negli istanti successivi o a mantenersi costanti, riflettendo una potenziale situazione critica.

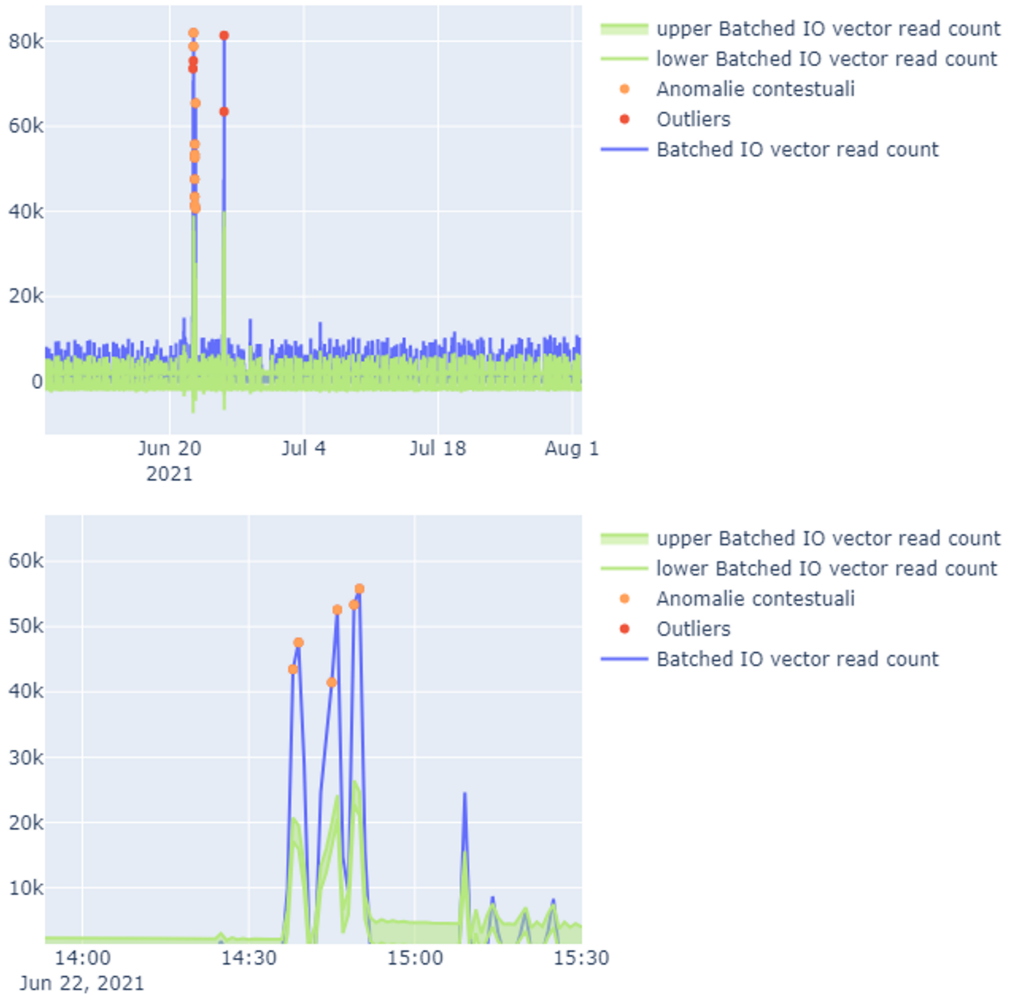


Figura 5.8. La prima figura in alto riporta le anomalie identificate dalla prima metodologia, basata su modelli di serie temporali, per la metrica Batched IO vector read count. La seconda figura è un ingrandimento della serie temporale, che evidenzia le anomalie contestuali.

Figura 5.9. La prima figura in alto riporta le anomalie identificate dalla seconda metodologia, basata sull'algoritmo Isolation Forest, per la metrica data blocks consistent reads - undo records applied. La seconda figura è un ingrandimento della serie temporale, che evidenzia le anomalie contestuali.

Capitolo 6

Conclusioni

In questo capitolo vengono presentate le conclusioni della tesi scaturite dalla ricerca progettuale. Vengono riportati i risultati chiave più rilevanti ed una valutazione critica degli stessi, in funzione dell'obiettivo della tesi. In secondo luogo, vengono presentati dei suggerimenti e raccomandazioni per possibili sviluppi futuri di questa ricerca.

La gestione dei malfunzionamenti ha assunto un ruolo sempre più significativo e rilevante nel mercato odierno. Diverse imprese si trovano a gestire e mantenere molteplici applicazioni in un ambiente IT condiviso e composto da diverse componenti interdipendenti, rendendo ardua la manutenzione dei servizi. Di conseguenza, è necessario che l'infrastruttura tecnologica venga monitorata e mantenuta, mediante dei servizi specifici. L'obiettivo di questa tesi consisteva nella realizzazione di una struttura di base per uno strumento di supporto all'attività di monitoraggio, in grado di garantire una valutazione tempestiva delle problematiche e l'individuazione e visualizzazione puntuale delle metriche di performance di una base di dati, che presentano anomalie. Tale soluzione è stata ideata per essere integrata negli strumenti di monitoraggio esistenti in una azienda, con lo scopo di prevenire gli eventi bloccanti mediante identificazione di comportamenti anomali nella base di dati, garantire una più rapida ed efficiente rilevazione e risoluzione dei problemi, diminuire i costi aziendali ed aumentare il livello di credibilità e affidabilità dell'azienda nei confronti del cliente. Il progetto proposto si è focalizzato sull'analisi e selezione delle metriche di performance più significative della base di dati in esame, e successiva individuazione e confronto delle più opportune metodologie data driven di rilevazione delle anomalie. Le 2011 metriche di performance iniziali sono state ridotte a 408 mediante il processo di pulizia dei dati. Infine, attraverso il processo di selezione delle metriche si è arrivati a considerare esclusivamente 268 metriche. La selezione delle metriche di performance è avvenuta mediante applicazione iterativa di algoritmi di clustering (DBSCAN e K-means)

ed analisi delle correlazioni delle metriche di performance basate sul coefficiente di correlazione di Pearson. Per la rilevazione delle anomalie sono stati implementati e confrontati due differenti metodi. Il primo si basava sull'utilizzo di modelli di serie temporali, mentre il secondo sfruttava modelli di apprendimento automatico non supervisionato. Per il primo metodo implementato sono stati usati modelli ARIMA. Le anomalie sono state identificate sfruttando un valore soglia ottenuto come valore massimo della regione di confidenza della previsione della serie temporale in esame. L'analisi ha fornito dei risultati significativi per alcune metriche di performance, in quanto la metodologia utilizzata è stata in grado di rilevare correttamente i valori che si discostavano molto dai restanti valori nell'insieme dei dati. Tuttavia, sono state individuate delle metriche di performance in cui la metodologia utilizzata non ha garantito risultati rispondenti alle aspettative, in quanto ha rilevato anomalie molteplici e ripetitive. Il problema risiede nel fatto che tale metodologia è basata su modelli di previsione ed intervalli di confidenza di tale previsione. Conseguentemente, se i modelli ARIMA presentano una difficoltà nella previsione della serie temporale, tale difficoltà viene estesa anche alla fase successiva di identificazione delle anomalia, in quanto il valore di soglia stabilito non sarà corretto. Il secondo metodo per l'identificazione di punti anomali, utilizzato nel seguente lavoro di tesi, è stato l'algoritmo Isolation Forest. È emerso che l'efficienza del modello dipenda strettamente dal parametro di contaminazione che viene selezionato per la singola metrica di performance. Tuttavia, Isolation Forest ha permesso di ottenere dei risultati conformi con l'ipotesi di questo lavoro di tesi. Confrontando i due metodi, è emerso che non vi è un metodo nettamente più performante di un altro. Questo a causa delle caratteristiche intrinseche del segnale, che possono inficiare l'efficienza del modello stesso. Conseguentemente, è stato proposto un modello combinato dei due metodi,che potesse sfruttare i vantaggi di entrambi. Il modello combinato ha fornito i risultati attesi in questo lavoro di tesi. Tuttavia, per le metriche di perfomance dove la metodologia basata su ARIMA, non garantiva risultati soddisfacenti, si è preferito usare esclusivamente l'algoritmo Isolation Forest. In un'ultima fase, in seguito a delle esigenze aziendali esposti dagli esperti dell'area di Infrastruttura, sono state identificate delle sequenze di punti anomali in un intervallo di tempo definito. L'obiettivo raggiunto è stato rilevare situazioni critiche che durano nel tempo e che potrebbero richiedere un intervento da parte del personale qualificato.
I risultati incoraggianti di questa ricerca hanno messo in luce la possibilità di utilizzare i modelli di rilevazione di anomalie costruiti per assegnare alle potenziali anomalie identificate delle etichette, utili per una futura implementazione di modelli di anomaly detection supervisionati.

6.1 Sviluppi futuri

In prospettiva futura, tale modello potrà essere ulteriormente migliorato ed eventualmente utilizzato come supporto per un processo di etichettatura dei dati. Nello specifico, si potrà procedere all'assegnazione delle etichette alle potenziali anomalie identificate mediante la soluzione progettuale. In seguito, potranno essere implementati e confrontati modelli di anomaly detection supervisionati. Il vantaggio dei modelli supervisionati risiede nella possibilità di valutare in modo automatico l'accuratezza confrontando il suo output con delle etichette prestabilite. Il passo successivo, potrebbe essere l'identificazione di anomalie in tempo reale, al fine di rilevare situazioni anomale o critiche in anticipo e dare la possibilità ai dipendenti aziendali dell'area di Infrastruttura di intervenire prima che il problema risulti bloccante. A tal proposito, si potrebbe sfruttare per l'estrazione ed analisi delle metriche di performance in tempo reale, il processo sviluppato nel lavoro di tesi mediante Confluent Platform, InfluxDB e Grafana, operando opportune modifiche. Successivamente, si potrebbe procedere allo sviluppo di un tool integrato all'interno degli strumenti di monitoraggio esistenti, ed in grado di rilevare e segnalare tempestivamente la presenza di un'anomalia relativa alle metriche del database. Ulteriori analisi ed approfondimenti potranno essere rivolte verso nuove metodologie per la selezione delle metriche di performance, al fine di ridurre ulteriormente il numero delle metriche di performance da considerare. Inoltre, si potranno ricercare e sperimentare ulteriori metodologie di rilevamento delle anomalie di performance a partire dalle osservazioni ed analisi condotte in questo lavoro di tesi. Gli aspetti da indagare sono innumerevoli e le potenzialità applicative del modello costituiscono un punto di partenza per investigazioni future.

Elenco delle figure

Elenco degli algoritmi

Bibliografia

[1] R. D. M. Online, "Il ruolo dell'infrastruttura it nella trasformazione digitale," 2015. [Online]. Available: https://www.datamanager.it/2015/09/il-ruolo-dellinfrastruttura-it-nella-trasformazione-digitale/

[2] F. Li, Q. Li, L. Mei, S. Li, L. Rong, W. Chen, and F. Wang, "Second order-based real-time anomaly detection for application maintenance services," vol. 2016-February. Institute of Electrical and Electronics Engineers Inc., 2 2016, pp. 37–44.

[3] M. Consulting, "Mediamente consulting: tecnologie di business avanzate." [Online]. Available: https://www.mediamenteconsulting.it/

[4] "Introduction to artificial intelligence." [Online]. Available: https://cs.lmu.edu/~ray/notes/introai/

[5] F. Bellaiche, "On quantum computing and artificial intelligence," 7 2018. [Online]. Available: https://www.quantum-bits.org/?p=2336

[6] F. Corea, "Springer briefs in applied sciences and technology computational intelligence artificial intelligence and exponential technologies: Business models evolution and new investment opportunities." [Online]. Available: http://www.springer.com/series/10618

[7] D. Butvinik, "Feature selection — exhaustive overview." [Online]. Available: https://medium.com/analytics-vidhya/feature-selection-extended-overview-b58f1d524c1c

[8] D. A. A. G. Singh, S. A. A. Balamurugan, and E. J. Leavline, "An unsupervised feature selection algorithm with feature ranking for maximizing performance of the classifiers," *International Journal of Automation and Computing*, vol. 12, pp. 511–517, 10 2015.

[9] M. Köppen, "The curse of dimensionality."

[10] J. Brownlee, "How to choose a feature selection method for machine learning," 11 2019. [Online]. Available: https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data

[11] P. Mitra, C. Murthy, and S. Pal, "Unsupervised feature selection using feature similarity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 301–312, 2002.

[12] W. Kirch, Ed., *Pearson's Correlation Coefficient*. Dordrecht: Springer Netherlands, 2008, pp. 1090–1091. [Online]. Available: https://doi.org/10.1007/978-1-4020-5614-7_2569

[13] J. Paparrizos and L. Gravano, "K-shape: Efficient and accurate clustering of time series," vol. 2015-May. Association for Computing Machinery, 5 2015, pp. 1855–1870.

[14] M. Yi, "A complete guide to heatmaps," 11 2019. [Online]. Available: https://chartio.com/learn/charts/heatmap-complete-guide/

[15] chelladurai, "Clustering new paper." [Online]. Available: http://sites.google.com/site/ijcsis/

[16] C.-E. B. N'Cir, G. Cleuziou, and N. Essoussi, *Overview of Overlapping Partitional Clustering Methods*. Cham: Springer International Publishing, 2015, pp. 245–275. [Online]. Available: https://doi.org/10.1007/978-3-319-09259-1_8

[17] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: An overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, pp. 86–97, 1 2012.

[18] Yellowbrick, "Elbow method." [Online]. Available: https://www.scikit-yb.org/en/latest/api/cluster/elbow.html

[19] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, pp. 100–108, 1979. [Online]. Available: http://www.jstor.org/stable/2346830

[20] C. D. M. R. P. S. Hinrich., *Introduction to information retrieval*. Cambridge University Press, 2008.

[21] L. B. Neuristique and Y. Bengio, "Convergence properties of the k-means algorithms."

[22] X. Jin and J. Han, *K-Medoids Clustering*. Boston, MA: Springer US, 2010, pp. 564–565. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_426

[23] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," 1996. [Online]. Available: www.aaai.org

[24] T. Mullin, "Dbscan parameter estimation using python," 7 2020. [Online]. Available: https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd

[25] Chire, "Dbscan," 10 2011. [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=17045963

[26] B. Rahdari and T. Arabghalizi, "Event-based user profiling in social media using data mining approaches," Ph.D. dissertation, 04 2017.

[27] J. Paparrizos and L. Gravano, "Fast and accurate time-series clustering," *ACM Trans. Database Syst.*, vol. 42, no. 2, Jun. 2017. [Online]. Available: https://doi.org/10.1145/3044711

[28] F. Petitjean and P. Gançarski, "Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment," *Theoretical Computer Science*, vol. 414, no. 1, pp. 76–91, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S030439751100822X

[29] J. Yang, C. Ning, C. Deb, F. Zhang, D. Cheong, S. E. Lee, C. Sekhar, and K. W. Tham, "k-shape clustering algorithm for building energy usage patterns analysis and forecasting model accuracy improvement," *Energy and Buildings*, vol. 146, pp. 27–37, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778817305352

[30] tslearn, "Kshape." [Online]. Available: https://tslearn.readthedocs.io/en/stable/auto_examples/clustering/plot_kshape.html

[31] O. Ibidunmoye, F. Hernández-Rodriguez, and E. Elmroth, "Performance anomaly detection and bottleneck identification," 7 2015.

[32] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[33] M. Xie, S. Han, B. Tian, and S. Parvin, "Anomaly detection in wireless sensor networks: A survey," *Journal of Network and Computer*

Applications, vol. 34, no. 4, pp. 1302–1325, 2011, advanced Topics in Cloud Computing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804511000580

[34] S. W. A.-H. Baddar, A. Merlo, M. Migliardi, and S. A.-H. Baddar, "Anomaly detection in computer networks: A state-of-the-art review designing sorting networks: A new paradigm view project usable captcha for smartphones view project anomaly detection in computer networks: A state-of-the-art review," 2014. [Online]. Available: https://www.researchgate.net/publication/270274504

[35] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys and Tutorials*, vol. 16, pp. 303–336, 3 2014.

[36] N. G. NICOGOERNITZ, K. R. KONRADRIECK, and U. Brefeld, "Toward supervised anomaly detection marius kloft," pp. 235–262, 2013.

[37] G. Blanchard, B.-B. De, G. Lee, and C. Scott, "Semi-supervised novelty detection *," pp. 2973–3009, 2010.

[38] M. A. M. Capretz and D. G. Bitsuamlak, "Collective contextual anomaly detection for building energy consumption," 2016.

[39] M. Peixeiro, "The complete guide to time series analysis and forecasting," 8 2019. [Online]. Available: https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775

[40] V. Chandola, D. Cheboli, and V. Kumar, "Detecting anomalies in a time series database detecting anomalies in a time series database detecting anomalies in a time series database," 2009.

[41] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250–2267, 2014.

[42] "3.2 time series components | forecasting: Principles and practice (3rd ed)." [Online]. Available: https://otexts.com/fpp3/components.html

[43] "How to decompose time series data into trend and seasonality." [Online]. Available: https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/

[44] "Seasonal adjustment - office for national statistics." [Online]. Available: https://www.ons.gov.uk/methodology/methodologytopicsandstatisticalconcepts/seasonaladjustment

[45] Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning, "Stl: A seasonal-trend decomposition procedure based on loess," *Journal of Official Statistics*, vol. 6, no. 1, pp. 3–73, 1990.

[46] "Stl algorithm explained: Stl part ii – dillon r. gardner – data scientist and consultant." [Online]. Available: http://www.gardner.fyi/blog/STL-Part-II/

[47] "3.6 stl decomposition | forecasting: Principles and practice (3rd ed)." [Online]. Available: https://otexts.com/fpp3/stl.html

[48] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," vol. 2015-August. Association for Computing Machinery, 8 2015, pp. 1939–1947.

[49] "Anomaly detection in time series: 2021 - neptune.ai." [Online]. Available: https://neptune.ai/blog/anomaly-detection-in-time-series

[50] Z. Ouyang, P. Ravier, and M. Jabloun, "Stl decomposition of time series can benefit forecasting done by statistical methods but not by machine learning ones," *Engineering Proceedings*, vol. 5, p. 42, 7 2021.

[51] "stazionarietà statistica in "dizionario di economia e finanza"." [Online]. Available: https://www.treccani.it/enciclopedia/stazionarieta-statistica_%28Dizionario-di-Economia-e-Finanza%29/

[52] S. Prabhakaran, "Augmented dickey-fuller (adf) test - must read guide - ml+," 11 2019. [Online]. Available: https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/

[53] "9.1 stationarity and differencing | forecasting: Principles and practice (3rd ed)." [Online]. Available: https://otexts.com/fpp3/stationarity.html

[54] "9.3 autoregressive models | forecasting: Principles and practice (3rd ed)." [Online]. Available: https://otexts.com/fpp3/AR.html

[55] "9.4 moving average models | forecasting: Principles and practice (3rd ed)." [Online]. Available: https://otexts.com/fpp3/MA.html

[56] "Autoregressive integrated moving average - wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

[57] "9.5 non-seasonal arima models | forecasting: Principles and practice (3rd ed)." [Online]. Available: https://otexts.com/fpp3/non-seasonal-arima.html

[58] "Time series forecasting using auto arima in python | by sushmitha pulagam | towards data science." [Online]. Available: https://towardsdatascience.com/time-series-forecasting-using-auto-arima-in-python-bb83e49210cd

[59] "9.6 estimation and order selection | forecasting: Principles and practice (3rd ed)." [Online]. Available: https://otexts.com/fpp3/arima-estimation.html

[60] "9.9 seasonal arima models | forecasting: Principles and practice (3rd ed)." [Online]. Available: https://otexts.com/fpp3/seasonal-arima.html

[61] "5.7 forecasting with decomposition | forecasting: Principles and practice (3rd ed)." [Online]. Available: https://otexts.com/fpp3/forecasting-decomposition.html

[62] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest."

[63] "Anomaly detection with isolation forest | by eugenia anello | better programming." [Online]. Available: https://betterprogramming.pub/anomaly-detection-with-isolation-forest-e41f1f55cc6

[64] W. R. Chen, Y. H. Yun, M. Wen, H. M. Lu, Z. M. Zhang, and Y. Z. Liang, "Representative subset selection and outlier detection: Via isolation forest," *Analytical Methods*, vol. 8, pp. 7225–7231, 10 2016.

[65] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, 07 2009.

[66] "Confluent platform | confluent documentation." [Online]. Available: https://docs.confluent.io/platform/current/overview.html

[67] "Apache kafka: Scopriamo le potenzialità del data streaming - seacom." [Online]. Available: https://www.seacom.it/apache-kafka-data-streaming/

[68] "Connectors to kafka | confluent documentation." [Online]. Available: https://docs.confluent.io/home/connect/overview.html

[69] "Schema registry overview | confluent documentation." [Online]. Available: https://docs.confluent.io/platform/current/schema-registry/

[70] "Influxdb: Purpose-built open source time series database | influxdata." [Online]. Available: https://www.influxdata.com/

[71] "Influxdb time series database: cogli nuove opportunità in tempo reale." [Online]. Available: https://www.extrasys.it/it/red/prodotti/influxdb

[72] "Linguaggio di programmazione python." [Online]. Available: https://www.python.it/

[73] "Project jupyter | home." [Online]. Available: https://jupyter.org/

[74] "Numpy." [Online]. Available: https://numpy.org/

[75] "User guide — pandas 1.3.4 documentation." [Online]. Available: https://pandas.pydata.org/docs/user_guide/index.html#user-guide

[76] "Plotting data in python: matplotlib vs plotly - activestate." [Online]. Available: https://www.activestate.com/blog/plotting-data-in-python-matplotlib-vs-plotly/

[77] "Tensorflow vs. scikit-learn: How do they compare?" [Online]. Available: https://www.springboard.com/library/machine-learning-engineering/scikit-learn-vs-tensorflow/

[78] "Github - pandas-profiling/pandas-profiling: Create html profiling reports from pandas dataframe objects." [Online]. Available: https://github.com/pandas-profiling/pandas-profiling

[79] R. Microlearning, "Strumenti di data visualization." [Online]. Available: https://ready4.fondirigenti.it/courses/data-analytics-2-new/lessons/6019

[80] C. Chapman, "A complete overview of the best data visualization tools." [Online]. Available: https://www.toptal.com/designers/data-visualization/data-visualization-tools

[81] "Data visualization software: cosa sono, trend, applicazioni e le 5 soluzioni top - sceglifornitore." [Online]. Available: https://sceglifornitore.it/blog/data-visualization-software-cosa-sono-trend-applicazioni-e-le-5-soluzioni-top/

[82] "Qual e la differenza tra reporting e analisi?" [Online]. Available: https://zipreporting.com/it/analytical-report/.html

[83] "Introduction to grafana | grafana labs." [Online]. Available: https://grafana.com/docs/grafana/latest/introduction/

[84] "Che cos'è power bi? - power bi | microsoft docs." [Online]. Available: https://docs.microsoft.com/it-it/power-bi/fundamentals/power-bi-overview

[85] "Power bi desktop - report interattivi | microsoft power bi." [Online]. Available: https://powerbi.microsoft.com/it-it/desktop/

[86] "Statistics descriptions." [Online]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/21/refrn/statistics-descriptions-2.html#GUID-2FBC1B7E-9123-41DD-8178-96176260A639

[87] "Feature scaling." [Online]. Available: https://en.wikipedia.org/wiki/Feature_scaling

[88] Wikipedia, "Akaike information criterion." [Online]. Available: https://en.wikipedia.org/wiki/Akaike_information_criterion

Ringraziamenti

Per la realizzazione di questo mio lavoro di tesi vorrei ringraziare la professoressa Tania Cerquitelli, e i miei tutor aziendali Graziano Fracasso e Antonio Greco per avermi concesso l'opportunità di crescere accademicamente e professionalmente, e per avermi fornito un supporto costante in questo percorso.
Ci tengo a ringraziare tutti i componenti dell'azienda Mediamente Consulting, per avermi accolto in un gruppo solido e in una grande famiglia. Grazie a loro mi sono sentita sempre a mio agio, in un ambiente sicuro, felice e sereno.
Un ringraziamento speciale va alla mia famiglia e ai miei nonni per il supporto e l'amore che ogni giorno sono in grado di donarmi. È a loro che dedico questo mio lavoro di tesi.