Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Gestionale
A.a. 2021/2022
Sessione di Laurea Dicembre 2022

Anomaly Detection: Implementazione e integrazione di modelli non supervisionati su metriche di performance.

Relatore:
Tania Cerquitelli

Candidato:
Camilla Gaudiani
290630

Tutor aziendale:
Antonio Greco

Indice

1 Introduzione

In questo breve capitolo introduttivo viene fornita una descrizione del contesto nel quale si inserisce il progetto di tesi. Inoltre, viene fornita una descrizione dell'azienda in cui il progetto è stato svolto, seguita da una sezione volta ad illustrare gli obiettivi perseguiti.

1.1 Background

L'evoluzione tecnologica ha condotto a un processo di trasformazione digitale, in particolare nell'ambito del mondo produttivo. La digitalizzazione dei processi aziendali consente di raggiungere risultati migliori in termini di velocità d'esecuzione ed efficienza; tali fattori giocano un ruolo cruciale in un mercato sempre più competitivo e in continuo mutamento. L'incremento dei dispositivi in grado di automatizzare molte operazioni ha determinato la generazione di grandi quantità di dati; l'elaborazione dei cosiddetti *Big Data* richiede tecnologie e risorse che vanno al di là dei sistemi tradizionali di immagazzinamento e gestione dei dati. Tali dispositivi generano e distribuiscono dati in maniera autonoma, tramite la connessione a internet, creando una rete denominata *Internet of things.* Si sente sempre di più parlare di intelligenza artificiale; tramite tecniche di *machine learning* si rilevano strutture all'interno di basi di dati e si estraggono informazioni, con molta più efficienza di quanto farebbe l'uomo.

L'identificazione dei problemi tramite un approccio reattivo, dunque, tramite allarmi basati su soglie statistiche non risulta appropriato per l'infrastruttura *cloud* dei giorni d'oggi. La presenza di ambienti dinamici richiede un nuovo approccio che si proattivo e non più reattivo. L'analisi predittiva e le tecniche di *machine learning* consentono di identificare automaticamente le anomalie in base alle metriche specifiche di un ambiente. Tali analisi sono volte ad evitare falsi allarmi e consente di rilevare le anomalie più difficilmente individuabili.

1.2 Azienda

Il progetto di ricerca, oggetto di tesi, nasce all'interno dell'azienda Mediamente Consulting s.r.l. Il seguente paragrafo offre una breve descrizione di tale contesto aziendale e delle principali aree di specializzazione che lo caratterizzano.

Mediamente Consulting è una società di consulenza specializzata sulle tematiche e sulle tecnologie di *Business Analytics* avanzate. L'azienda nasce con l'obiettivo di valorizzare i dati delle imprese guidandole nell'innovazione e offrendo strumenti per fronteggiare le nuove sfide del mercato. La società è stata fondata nell'anno 2013 da professionisti nell'ambito IT, come *startup* innovativa, ottenendo il riconoscimento di *startup* nel 2016 da parte dell'incubatore di imprese innovative del politecnico di Torino grazie ai meriti ottenuti per crescita in termini di fatturato e occupazione. Mediamente Consulting è partecipata da Var Group che appartiene al gruppo SeSa S.p.A, quotato presso la borsa italiana. Attualmente il team è costituito da 50 persone distribuite nelle 4 sedi operative in Italia (Milano, Bologna, Empoli, Torino).

Le aree di specializzazione sono le seguenti: Infrastruttura Tecnologica, *Data Integration* e *Management*, *Corporate Performance Management*, *Advanced Analytics*, *Business Intelligence*. Inerentemente all'ambito di infrastruttura, l'azienda si occupa della gestione, dell'aggiornamento e del *tuning* di sistemi ingegnerizzati; in particolare offre attività di consulenza su diverse tecnologie Oracle. Quest'ultima rappresenta la tecnologia che l'azienda ha scelto di padroneggiare con il massimo livello di competenza. Inoltre, Mediamente Consulting è specializzata nella *governance* dei dati con l'obiettivo di valorizzarli integrando tutti i dati aziendali interni ed esterni; dunque, offre ai clienti un'infrastruttura tecnologica che consente di gestire una grande quantità di dati, trasformandoli in conoscenza tramite lo sviluppo di sistemi di analisi dei processi aziendali. In aggiunta l'azienda è specializzata nel *Corporate Performance Management* che comprende tutte le metodologie e i processi volti a definire le strategie aziendali, alla traduzione di tali strategie in piani d'azione e budget e il monitoraggio dell'esecuzione delle strategie.

1.3 Obiettivo

Lo studio proposto, oggetto di tesi, si inserisce in un progetto aziendale più ampio. L'area di Mediamente Consulting srl, specializzata in infrastruttura tecnologia, attualmente utilizza come strumento di monitoraggio *Oracle Enterprise Manager*. Quest'ultimo fornisce una soluzione completa di monitoraggio e gestione per *Oracle Database* e *Engineered Systems* distribuiti nel *Cloud* e nei *Data Center* dei clienti. *Oracle Enterprise Manager* effettua il monitoraggio di *targets* come *Databases* o i *listeners* tramite degli *agents* installati sugli *host, o targets,* oggetto di monitoraggio. Gli *agents* raccolgono informazioni sulla *performance* dei *targets* sui quali sono installati. Per effettuare il monitoraggio, per ogni metrica sono settate delle soglie. Attraverso l'ausilio di una *dashboard,* i dipendenti dell'area di infrastruttura hanno la possibilità di gestire le richieste relative ai differenti targets dei clienti. La pagina principale del pannello è caratterizzata dalla visualizzazione degli allarmi, relativi alle metriche di performance, i quali si originano dal superamento delle soglie prestabilite. Oltre ai sistemi di monitoraggio descritti, è disponibile un altro sistema di *alerting* che consiste in una comunicazione diretta tra operatore e cliente. L'operatore approccia alla problematica sulla base della sua esperienza pregressa e su una valutazione soggettiva. Tale sistema di *alerting* non si basa su valutazioni oggettive scaturite dall'analisi dei dati.

L'obiettivo di tesi nasce in risposta alla necessità di rendere più efficiente il sistema di gestione degli allarmi. Infatti, l'approccio reattivo alla rilevazione delle anomalie, adottato dall'azienda, causa spesso tempi di inattività e problemi di prestazioni. Dunque, al fine di trasformare l'approccio reattivo in un approccio proattivo, il progetto di tesi si pone l'obiettivo di fornire soluzioni per il rilevamento delle anomalie delle metriche di *Performance*, tramite l'implementazione di modelli di *Machine Learning*. I miglioramenti che l'approccio proattivo apporterebbe all'azienda sono molteplici; tra questi emerge un incremento in termini di credibilità nei confronti dei clienti e maggiore tempestività e efficienza nella gestione e nella risoluzione dei problemi con una conseguente riduzione dei costi aziendali. Inoltre, la soluzione proposta dovrebbe essere integrata con i sistemi di monitoraggio già utilizzati dall'azienda.

Infine, come obiettivo specifico di tesi, vi è lo scopo di affrontare la tematica dell'*Anomaly Detection,* attraverso la ricerca di modelli specifici e adatti per il rilevamento delle anomalie su dati generati in *real time* e caratterizzati da alte frequenze. Tali tipologie di dati caratterizzano le infrastrutture *Cloud* dei giorni d'oggi ed è proprio in relazione a tali ambiente dinamici che vi è la necessità di identificare automaticamente le anomalie tramite analisi predittive e tecniche di *Machine Learning.*

La tesi è articolata in sei capitoli. Il secondo capitolo riporta, in primo luogo, nozioni teoriche di carattere generale nell'ambito del *Machine Learning* e, successivamente, nozioni più specifiche relative alle serie storiche e ai modelli di *Anomaly Detection* implementati. Il terzo è un breve capitolo che illustra le tecnologie utilizzate per sviluppare il progetto in questione. Successivamente vi è un capitolo dedicato alla fase di preelaborazione dei dati e di esplorazione del *dataset.* In tale capitolo si illustrano principalmente le metodologie attuate per effettuare la riduzione delle *features*. Infine, la tesi si concentra sull'implementazione di modelli di *Anomaly Detection* su metriche di *performance*. Nel quinto capitolo si mostra la metodologia attuata per effettuare il rilevamento delle anomalie e si illustra il modello integrato proposto nel progetto di tesi. L'ultima sezione del capitolo è dedicata ai risultati sperimentali ottenuti dalle analisi. Il sesto capitolo è dedicato alle conclusioni e ai possibili approfondimenti futuri.

2 Stato dell'arte

In questo capitolo si riportano le principali nozioni teoriche sui concetti affrontati nel progetto di tesi. In primo luogo si affrontano tematiche generali relative al *machine learning,* per poi fornire un *overview* sulle principali tecniche che esistono in letteratura nella fase di preparazione dei dati. Infine, le ultime sezioni del capitolo sono dedicate alle serie storiche e alle metodologie esistenti in letteratura per rilevarne le anomalie.

2.1 Machine Learning

Il *Machine Learning* consente di creare sistemi che apprendono o migliorano la *performance* in base ai dati utilizzati. Esso costituisce un sottoinsieme dell'intelligenza artificiale (AI). Quest'ultima si riferisce a sistemi che imitano l'intelligenza umana per eseguire delle attività. Il *Machine Learning* è strettamente legato al riconoscimento di *patterns* ed è volto alla costruzione di algoritmi che consentano di apprendere da un insieme di dati e fare predizioni su questi. Attualmente il *Machine Learning* trova molti contesti applicativi come nel campo dell' *e-commerce*. Infatti, grazie all'utilizzo di algoritmi l'esperienza di acquisto degli utenti risulta sempre più efficiente.

Gli algoritmi rappresentano il motore che alimenta il *Machine Learning.* Attualmente i principali algoritmi utilizzati riguardano un apprendimento supervisionato, non supervisionato e per rinforzo. A metà strada tra l'apprendimento supervisionato e semi supervisionato vi è l'apprendimento semi-supervisionato. Tale classificazione è relativa alla "natura" del segnale utilizzato per l'apprendimento. Inoltre, è possibile classificare gli algoritmi in base al risultato desiderato del sistema di *Machine Learning.*

L'apprendimento supervisionato è una tecnica che mira a istruire un sistema informatico in modo da consentirgli di elaborare automaticamente previsioni sui valori di uscita di un sistema rispetto ad un input. Ciò è possibile se inizialmente sono fornite coppie di input-output. Lo scopo ultimo di tale apprendimento è di creare una funzione che sia in grado di "apprendere" dai risultati forniti durante la fase di esempio, per poi fornire risultati desiderati per gli esempi non forniti. Tra i problemi di Machine Learning supervisionati si distinguono i problemi di classificazione e di regressione.

L'apprendimento non supervisionato è una tecnica di apprendimento automatico che consiste nel fornire all'algoritmo un insieme di dati che verranno riorganizzati sulla base di caratteristiche comuni, per poter elaborare ragionamenti e previsioni su input successivi. Questi algoritmi lavorano confrontando dati, ricercando similarità o differenze. La differenza principale con gli algoritmi supervisionati risiede nell'assenza di classi note a priori. Tali classi devono essere apprese dall'algoritmo. I principali algoritmi non supervisionati sono i clustering, le regole di associazione e le tecniche di riduzione dimensionale dei dati.

L'apprendimento rinforzato addestra un agente autonomo a rispondere a un ambiente in modo da perseguire un obiettivo, utilizzando un meccanismo di prove ed errori. L'agente viene fatto interagire con un "ambiente" e ottiene una ricompensa, se le previsioni verificano determinate condizioni. Tramite la ripetizione del processo, il modello è in grado di trovare le soluzioni che massimizzano la ricompensa.

Il *Machine Learning* e il *data mining* si sovrappongono significativamente. Infatti, il *data mining* utilizza i metodi di *Machine Learning.* Tuttavia le principali differenze riguardano la modalità in cui sono valutate le prestazioni degli algoritmi. Nel primo caso le prestazioni sono valutate generalmente nella capacità di riprodurre conoscenza già acquisita, mentre il data mining ha lo scopo di estrarre nuova conoscenza dai dati e proprietà sconosciute dei dati.

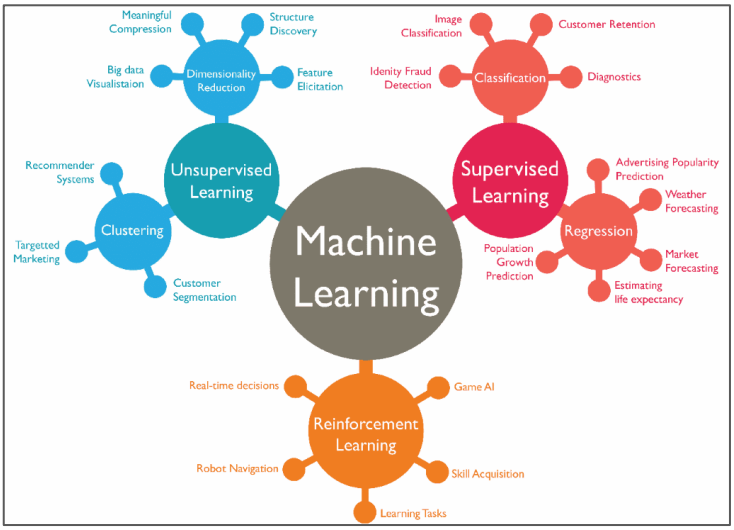

Figura 1.1: Rappresentazione schematica della classificazione dei modelli di machine learning.

2.2 Preparazione e esplorazione dei dati

La preparazione dei dati è il processo di pulizia e di trasformazione dei dati grezzi prima dell'elaborazione e dell'analisi. La fase di preparazione dei dati consente di effettuare analisi efficienti, limitando gli errori che possono verificarsi durante la fase di elaborazione. Tra gli aspetti da considerare vi è la possibilità di cambiare la granularità del dato attraverso un operazione di aggregazione. Inoltre è possibile effettuare riduzioni delle dimensioni dei dati. Tra le varie tecniche è noto il campionamento, che consente di ridurre i dati in termini di righe, e la selezione delle *features,* che consente la riduzione dei dati in termini di colonne. Nel contesto dei modelli predittivi, la presenza di *features* irrilevanti può impattare negativamente sull'accuratezza del modello. Un altro aspetto da considerare nella fase di *preprocessing* è la possibilità di creare nuove *features.* Creare nuovi attributi implica effettuare un processo di *"feature engineering",* ossia l'azione mediante la quale si trasforma un dato molto specifico in un nuovo formato. Inoltre, nell'ambito della preelaborazione dei dati si colloca la *Data Exploration*. Essa è costituita da una serie di attività che consentono di acquisire maggiore consapevolezza sui dati analizzati, mettendo in luce aspetti che consentono di estrarre una conoscenza fruibile. Tali analisi consentono di pilotare positivamente le successive fasi di analisi dei dati. In tale contesto assume un ruolo fondamentale la visualizzazione dei dati. Grazie a elementi visivi come diagrammi, grafici e *dashboard* è possibile cogliere valori anomali nei dati, tendenze e ricorrenze presenti nei dati.

Trasformazione delle *features*: Spesso si effettuano trasformazioni degli attributi; in questo contesto rientrano la normalizzazione min-max e la standardizzazione, ovvero la normalizzazione z-score. La trasformazione consente di rappresentare i domini degli attributi in domini comparabili. Per quanto riguarda la normalizzazione min-max si effettua applicando la seguente formula:

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

(2. 1)

La normalizzazione min-max scala in modo lineare ogni funzionalità all'intervallo [0,1]. La normalizzazione z-score converte tutti i valori in un punteggio z:

$$Z = \frac{x - \bar{x}}{\sigma}$$

(2. 2)

Dove $\bar{x}$ rappresenta la media di x e σ la deviazione standard.

Selezione delle *features*: Tra le tecniche di riduzione dimensionale, la *feature selection* consente di identificare un sotto-insieme di *features* che caratterizzano i dati. A tal fine è necessario individuare e, successivamente, eliminare le *features* ridondanti e irrilevanti ai fini dell'analisi. Poiché gli attributi possono essere tra di loro correlati è possibile identificare il minimo sotto-insieme di attributi non correlati. Tra le metodologie di *Feature Selection* si distingue l'approccio *Wrapper* che ricerca le combinazioni di variabili che spiegano il modello con la migliore accuratezza. L'approccio *Filter* permette di filtrare le variabili più irrilevanti. Utilizzando una misura statistica viene calcolato un punteggio per ogni variabile. La misura statistica considerata può essere il Chi – Quadro, il coefficiente di *Pearson*, il *Fisher Score* ecc.

Correlazione: è una statistica che quantifica la forza della relazione tra due variabili. Indica la tendenza che hanno due variabili a variare insieme, ovvero, a covariare. Il tipo di relazione può essere lineare o non lineare. La relazione è lineare se, rappresentata su assi cartesiani, si avvicina alla forma di una retta. Dunque, all'aumentare (o al diminuire) di una variabile, aumenta (o diminuisce) l'altra. La relazione è non lineare se, rappresentata su assi cartesiani, ha un andamento curvilineo (parabola o iperbole). È inoltre necessario considerare la forma della relazione che si caratterizza in entità e direzione. La direzione può essere positiva, se all'aumentare di una variabile aumenta anche l'altra, e negativa, se all'aumentare di una variabile, diminuisce l'altra. L'entità si riferisce alla forza della relazione esistente tra due variabili. È possibile rappresentare la correlazione tramite degli indici. Si distingue l'indice di *Pearson* che identifica correlazioni di tipo lineare e l'indice di *Spearman*. Quest'ultimo permette di stabilire quanto bene una relazione tra due variabili può essere descritta da una funzione monotona. L'indice di *Pearson* è una metrica di correlazione lineare che si basa sul concetto di covarianza e di deviazione standard. La correlazione lineare si dice perfetta quando il valore è pari a 1 o a -1. Di seguito si riporta la formula del coefficiente di *Pearson*:

$$corr(x, y) = \frac{covarianza(x, y)}{deviazione\_standard(x) * deviazione\_standard(y)}$$

(2. 3)

2.3 Serie temporali

Una serie temporale *k*-variata è una raccolta sequenziale di osservazioni relative a *k* variabili in *n* istanti di tempo. Una serie temporale univariata è una sequenza di misure, raccolte nel tempo, relative a una singola variabile.

L'analisi di una serie temporale può perseguire diversi obiettivi; tra questi è possibile utilizzare modelli per caratterizzare la serie stessa, modelli volti a spiegare come i dati del passato possono influenzare quelli del futuro e come due serie temporali "interagiscono" tra loro. Inoltre, esistono modelli volti a determinare previsioni dei valori futuri della serie temporale.

I due principali modelli utilizzati nel contesto dell'analisi delle serie temporali sono:

- modelli ARIMA (acronimo di *Auto-Regressive Integrated Moving Average*): mettono in relazione il valore attuale di una serie con valori e errori di previsione relativi a periodi del passato.
- modelli di regressione ordinari: utilizzano gli indici temporali come variabili x. Sono utilizzati per descrivere i dati e rappresentano la base di alcuni metodi di previsione.

Per poter perseguire uno studio di una serie temporale è necessario porsi alcune domande circa la presenza o meno di alcune caratteristiche. In primo luogo, bisogna valutare se, in media, le misurazioni, all'interno della sequenza, tendono ad aumentare o diminuire nel tempo e se esiste un fattore di stagionalità che implica la presenza di un *pattern* che si ripete regolarmente in alcuni periodi dell'anno come stagioni, trimestri, mesi, giorni della settimana. Inoltre, è necessario valutare la presenza di valori anomali che si discostano notevolmente dalla linea di regressione, se esiste una variazione ciclica non correlata a fattori di stagionalità e, infine, analizzare la varianza della serie temporale oggetto di analisi.

Per prevedere il valore che la serie temporale assumerà in un tempo specifico del futuro, è necessario conoscere la relazione tra ciò che è accaduto fino al tempo attuale e quello che accadrà al tempo futuro. Tale relazione, tuttavia, non è nota in quanto, per definizione, ciò che accadrà nel futuro non è stato osservato. Per risolvere tale problematica bisogna considerare un'ipotesi aggiuntiva, quale l'ipotesi di stazionarietà.

Infatti, molte serie sono non stazionarie in quanto caratterizzate dalla presenza di una componente tendenziale, ovvero presentano un *Trend* che rappresenta una tendenza persistente nel tempo e una componente stagionale. Questi fattori, che violano le condizioni di stazionarietà, riducono la validità delle previsioni basate sulle regressioni temporali.

Per analizzare una serie temporale è necessario definire alcuni concetti. Indicando con *Y* la variabile osservata al tempo t, per indicare i valori futuri e passati di *Y* si utilizza una terminologia specifica e una notazione particolare. Y_{t-1}, cioè il valore di Y nel periodo precedente, indica il primo ritardo di *Y*. Il j-esimo ritardo di Y è indicato con Y_{t-j} e rappresenta il valore di *Y, j* periodi indietro nel tempo. La differenza prima della variabile Y_t è la differenza tra il valore di Y al tempo t e il valore di Y al tempo t-1 e si indica con il "Δ". Dunque, la differenza prima di Y è indicata nel seguente modo; $\Delta Y_t = Y_t - Y_{t-1}$.

In una serie temporale il valore di *Y* in un periodo è correlato con il valore di *Y* in un periodo precedente. Pertanto, con il termine autocorrelazione si indica la correlazione di una serie temporale con i propri valori ritardati. La correlazione tra valori di *Y* in tempi adiacenti è la correlazione tra $Y_t \; e \; Y_{t-1}$ e rappresenta il primo coefficiente di autocorrelazione. In generale, il j-esimo coefficiente di autocorrelazione è dato dalla correlazione tra $Y_t \; e \; Y_{t-j}$ ed è espresso dalla seguente formula:

$$autocorrelazione_j = p_j = corr(Y_t, Y_{t-j}) = \frac{cov(Y_t, Y_{t-j})}{\sqrt{var(y_t)var(y_{t-j})}} \tag{2.4}$$

dove la covarianza tra $Y_t \; e \; Y_{t-j}$ rappresenta la j-esima autocovarianza di una serie Y_t.

2.3.1 Modelli AR

Un modello di autoregressione consente di effettuare previsioni mettendo in relazione una variabile temporale con i suoi valori passati. Un'autoregressione del primo ordine si indica con l'abbreviazione AR(1). Tale modello può essere descritto dalla seguente espressione:

$$y_t = \beta_0 + \beta_1 Y_{t-1} + u_t \tag{2.5}$$

Dove u_t rappresenta l'errore.

Tale modello utilizza Y_{t-1} per prevedere Y_t. Per incorporare l'informazione relativa a un passato più lontano bisogna aggiungere nel modello di primo ordine dei ritardi aggiuntivi; pertanto, si definisce un modello autoregressivo di ordine p (ovvero AR(p)) con p che rappresenta il numero di ritardi utilizzati dal modello. Dunque, i regressori del modello sono $Y_{t-1,}\, Y_{t-2}\, , \dots , Y_{t-p}$ e Y_t è espressa come funzione lineare di suoi ritardi. Il modello AR(p) è espresso dalla seguente espressione:

$$y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} + u_p$$

(2.6)

Dove $E(u_t | Y_{t-1}, Y_{t-2}, \dots Y_{t-P}) = 0$. Tale assunzione indica che u_t ha un valore atteso nullo condizionamente ai valori passati di *Y.* La prima implicazione dell'assunzione descritta è che la prima previsione Y_{T+1}, basata sull'intera storia, dipende solo dai p valori passati più recenti, la seconda è che gli errori u_t sono serialmente correlati. Inoltre, si definisce come errore di previsione la differenza tra il valore osservato di Y_{T+1} e la sua previsione basata su Y_T :

$$errore\ di\ previsione = Y_{T+1} - \hat{Y}_{T+1/T}$$

(2.7)

Per misurare l'entità dell'errore di previsione si utilizza l'RMSE (acronimo di *Root Mean Squared Forecast Error*), espresso dalla seguente formula:

$$RMSFE = \sqrt{E\left[(Y_{T+1} - \hat{Y}_{T+1/T})^2\right]}$$

(2.8)

Nel calcolo dell'RMSFE sono combinate due fonti d'errore. La prima dipende dal fatto che i valori di u_t non sono noti, la seconda deriva dalla stima dei coefficienti β_0 e β_1. Se la numerosità campionaria è molto elevata, può accadere che la prima fonte d'errore è molto maggiore della seconda; in questo caso l'RMSFE può essere approssimato come la deviazione standard dell'errore u_t nella regressione. Quest'ultima è stimata attraverso l'errore standard della regressione (SER, acronimo di *Standard Error of the Regression*).

Per determinare quanti ritardi includere nella regressione, in pratica, per scegliere l'ordine *p* si possono utilizzare differenti approcci. Il primo consiste nell'utilizzare test statistici come, per esempio, la statistica *t* per valutare se un determinato coefficiente è significativo. Questo metodo spesso produce modelli troppo grandi. Per arginare tale problematica è possibile effettuare la scelta dell'ordine minimizzando un "criterio d'informazione"; tra questi si distinguono il criterio d'informazione Bayesiano, o BIC (acronimo di *Bayes Information Critriation*) e il criterio d'informazione di Akaike, o AIC (acronimo di *Akaike Information Criteriation*).

Affinché un AR(p) sia stazionario, le radici del polinomio $1 - \beta_1 z - \beta_1 z^2 - \cdots \beta_1 z^p$, devono essere tutte maggiori di uno in valore assoluto. Le radici del polinomio sono i valori di z che soddisfano $1 - \beta_1 z - \beta_1 z^2 - \cdots \beta_1 z^p = 0$. Se y_t ha una radice unitaria allora presenta un trend stocastico e non è stazionaria. Per verificare la presenza di una radice unitaria si utilizza il test di Dickey – Fuller.

2.3.2 Modelli MA

Per modellare una serie storica, oltre al modello autoregressivo, è possibile utilizzare un modello a media mobile MA (acronimo di *Moving Average*). MA modella l'errore al tempo *t* come serialmente correlato, cioè come se fosse un ritardo distribuito di un altro termine d'errore inosservato. Con MA(q) si indica un modello a media mobile di *q* termini passati:

$$y_t = \sum_{i=0}^{q} \theta_i \, \epsilon_{t-i} = C(L)\epsilon_t$$

(2.9)

Un processo a media mobile (*Moving Average*) è una combinazione lineare di variabili casuali ϵ_t di tipo *white noise*. Un processo *white noise* è un processo composto da un numero infinito di variabili casuali con media uguale a zero e con varianza costante. Inoltre, tali variabili non sono correlate tra loro.

Il termine C(L) rappresenta un polinomio di ordine q nell'operatore ritardo. Se C(L) è un polinomio di grado q allora y_t è un processo *Moving Average* di ordine q.

Un processo MA ha media 0 che può essere espressa dalla seguente formula:

$$E(y_t) = E\left[\sum_{i=0}^{q} \theta_i \, \epsilon_{t-i}\right] = \sum_{i=0}^{q} \theta_i \, E(\epsilon_{t-i}) = 0$$

(2.10)

Dal momento che la media del processo è nulla, è possibile ottenere la varianza come E(y_t^2).

2.3.3 Modelli ARIMA

ARIMA (acronimo di *Auto-Regressive Integrated Moving Average*) è un modello che genera un'equazione lineare per descrivere e prevedere i dati di una serie temporale. Per generare l'equazione sono necessarie tre parti quali:

- AR (Auto-regressione)
- I (Integrazione o differenziazione)
- MA (media mobile)

il modello ARIMA(p,d,q) è caratterizzato dall'unione delle tre parti sopra descritte; dunque i parametri da fornire all'algoritmo sono:

- *p*: rappresenta l'ordine dell'autoregressione (AR)
- *d*: rappresenta l'ordine di differenziazione (I)
- *q*: rappresenta l'ordine media mobile (MA)

Il termine "AR" che sta per "*Auto Regressive*" significa che si tratta di un modello di regressione lineare che utilizza i suoi ritardi come predittori. Nei modelli di regressione lineare i predittori non devono essere correlati e devono essere indipendenti l'uno dall'altro. Il valore p (parte AR del modello ARIMA) descrive la dipendenza dei dati da quelli del passato. Dunque, il valore che assume *p* rappresenta il numero di ritardi utilizzati dal modello. Per rendere stazionaria una serie temporale si utilizza come approccio la differenziazione. Tale metodo consiste nel sottrarre il valore precedente al valore corrente. Potrebbe essere necessaria più di una differenziazione a seconda del tipo di serie considerata.

Dunque, il parametro "*d*" indica il numero minimo di differenziazioni necessario per rendere stazionaria la serie temporale.

Se la serie temporale presenta una stagionalità, è necessario aggiungere all'equazione termini stagionali e il modello prende il nome di SARIMA che sta a indicare "Seasonal ARIMA".

Per stimare il modello bisogna identificare i valori di p, d e q

Nel modello ARIMA vengono combinati i termini AR e MA e, per essere tale, deve avere un ordine di differenziazione almeno pari ad 1; se d=0, allora si tratta di un modello ARMA.

Bisogna identificare l'ordine di differenziazione d corretto. Infatti, una serie eccessivamente differenziata può andare a distorcere i parametri del modello. Il giusto ordine di differenziazione è la differenza minima richiesta per ottenere una serie quasi stazionaria che si aggira intorno a una media definita e affinché il grafico ACF (autocorrelazione) raggiunga lo zero abbastanza rapidamente. Se le autocorrelazioni sono positive per molti predittori, 10 o anche di più, allora la serie deve essere differenziata ulteriormente. Inoltre, se l'autocorrelazione del primo ritardo risulta eccessivamente negativa allora la serie è *over-differenced*. Se la scelta ricade tra due ordini di differenziazione è necessario scegliere l'ordine che determina la minor deviazione standard nella serie differenziate.

La differenziazione deve essere effettuata esclusivamente nel caso in cui la serie non è stazionaria. Per tale motivo bisogna verificare la stazionarietà della serie considerata. Un primo modo per verificarla potrebbe essere quello di osservare il grafico dell'andamento della serie nel tempo; se un processo è stazionario non presenta un andamento crescente o decrescente. Dunque, si potrebbe considerare come stazionario un processo che oscilla attorno a un valore costante. Tuttavia, tale metodo è soggettivo e difficilmente formalizzabile. In statistica esistono dei test volti a determinare se una serie è stazionaria o meno che prendono il nome di test di radice unitaria. Esistono più test di radice unitaria ma alla base di tali differenti metodologie vi è un'impostazione comune. Tra questi un test noto è l'*Augmented Dickey Fuller test.* L'ipotesi nulla del test ADF è che la serie non è stazionaria. Se il *p-value* è inferiore a 0,05 si rigetta l'ipotesi nulla e si stabilisce la stazionarietà della serie. Al contrario se il *p-value* risulta superiore al valore di 0,05 allora la serie non è stazionaria e bisogna determinare l'ordine di differenziazione. Nella scelta del parametro *d* potrebbe accadere che la serie risulti leggermente *over-differenced*; in questo

caso bisogna aggiungere termini MA all'equazione. Al contrario, nel caso in cui risultasse *under-differenced,* per aumentare la differenziazione si dovrebbe incrementare il numero di termini AR.

2.3.4 Modelli EWMA

Il modello EWMA (acronimo di *Exponentially Weigheted Moving Average*) consente di caratterizzare e descrivere una serie temporale. Il termine MA (acronimo di *Moving Average*) si riferisce al concetto di media mobile. Il modello EWMA assegna un peso inferiore alle osservazioni meno recenti; i pesi diminuiscono in modo esponenziale. Da tale principio deriva il termine EW (acronimo di *Exponentially Weigheted*).

L'EWMA è una funzione ricorsiva; tale principio determina la caduta esponenziale dei pesi. Il modello è definito dalla seguente espressione:

$$EWMA_t = \alpha x_t + (1-\alpha)\, EWMA_{t-1}$$

(2. 11)

Dove α rappresenta il peso che decresce esponenzialmente e x_t l'osservazione corrente al tempo t. L'equazione può essere riscritta in termini di pesi più vecchi:

$$EWMA_t = \alpha x_t + [(1-\alpha) * (\alpha x_{t-1} + (1-\alpha)\, EWMA_{t-2}]$$

(2. 12)

Dunque, $EWMA_t$ rappresenta la media ponderata di tutte le osservazioni precedenti.

2.3.5 Modelli di decomposizione

Una serie temporale può essere pensata come composizione di alcune componenti quali il Trend T_t, Stagionalità S_t e, infine, una componente residuale I_t. Il trend è una componente che varia lentamente nel tempo, la stagionalità indica la presenza di componenti periodiche. Alcuni modelli di composizione sono:

modello additivo: $y_t = T_t + S_t + R_t$

(2. 13)

modello moltiplicativo: $y_t = T_t S_t R_t$

(2. 14)

moltiplicativo con componente irregolare additiva: $y_t = T_t S_t + R_t$

(2. 15)

Considerando un modello di regressione classico, del tipo $y_t = (Trend)_t + (Stagionalità)_t + (Errore)_t$ con t=1,2, …,n

- Per un trend lineare, la variabile predittiva nella regressione è t
- Per un trend quadratico, le variabili predittive sono t e t^2
- Per rappresentare effetti stagionali si può definire $\vartheta_i = (Stagionalità)_i$ per i=1,…, 12 e la variabili $d_{i,t} = 1$ nel mese i-esimo, altrimenti pari a zero. Dove per $(Stagionalità)_t$ si intende, ne caso in esame, una componente periodica che si ripete ogni anno; $(Stagionalità)_{t+12} = (Stagionalità)_t$

Considerando il trend quadratico, il modello è descritto dalla seguente formula:

$$y_t = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \vartheta_1 d_{1,t} + \cdots + \vartheta_{12} d_{12,t} + (Errore)_t$$

(2. 16)

Con il vincolo lineare sui parametri $\vartheta_1 + \cdots + \vartheta_{12} = 0$. Per esprimere la componente periodica è possibile utilizzare, al posto delle variabili binarie, funzioni trigonometriche. Inoltre, è possibile esprimere interazioni tra il trend e la stagionalità, inserendo nell'equazione il prodotto tra le due variabili considerate.

Esistono alcune tecniche utili a scomporre una serie temporale nelle sue componenti elementari. In particolare, il fine di tale decomposizione è la stima della componente stagionale per andare a determinare serie destagionalizzate. Tale obiettivo viene perseguito in quanto la componente stagionale è una componente prevedibile e dunque poco interessate, che non consente di comprendere se un eventuale incremento o decremento relativo a una variabile sia reale o appunto "stagionale".

Scomposizione additiva e moltiplicativa

Nei modelli additivi e moltiplicativi si utilizza principalmente il concetto di media mobile. Quest'ultima rappresenta un metodo che consente lo smussamento della serie storica. Se la serie originale è composta esclusivamente da *trend* e dalla componente residua, allora la media mobile rimuove entrambi gli effetti che disturbano. Se nella serie storica è presente anche una stagionalità di periodo P, allora una media mobile di ampiezza P elimina anche la stagionalità. Il numero di termini coinvolti nel calcolo della media influenza il risultato. La media mobile è un metodo di adattamento locale in quanto crea una serie di valori smussati di lunghezza pari alla serie originaria, ognuno in corrispondenza del punto di osservazione t-esimo. Si parla di medie mobili centrate a k termini. Per dati mensili k=12, per dati trimestrali k=4, ecc.

I modelli di scomposizione additivi e moltiplicativi si basano sulla stessa metodologia di scomposizione. In primo luogo si calcola un trend-ciclo di prima approssimazione, tramite la determinazione della media mobile, dove MM_t è il valore di tale media al tempo t. Si determina poi una componente SE_t. Nel modello additivo SE_t si ottiene sottraendo al valore della serie al tempo t (y_t), il valore della media MM_t (nel modello moltiplicativo, invece, SE_t si ottiene effettuando la divisione tra y_t e MM_t). Considerando dati mensili, si determina un coefficiente di stagionalità θ_m con m=1,…12 calcolando la media di $(SE)_t$. Per il calcolo della media, nel contesto di dati mensili, si considerano i termini t=m, m+12, m+24 ecc. In seguito si deriva la serie destagionalizzata $D_t = y_t - \theta_m$ (nel modello moltiplicativo si effettua il rapporto). Successivamente si stima il ciclo-trend mediante una media mobile a 3 termini sui dati D_t. Infine, si effettua una stima della componente sistematica della serie, data dalla somma della componente di trend e stagionale (nel modello moltiplicativo si effettua il prodotto), per poi sottrarre dalla serie y_t questa componente e ottenere il residuo.

Scomposizione STL

Il metodo di decomposizione STL (acronimo di *Seasonal-Trend Decomposition Procedure Based on Loess*) è stato ideato *Robert B.Cleveland, William S.Clevenad, Jean E. McRae, e Irma Terpenning*, autori della pubblicazione "*Seasonal-Trend Decomposition Procedure Based on Loess".* La scomposizione STL si basa sull'utilizzo del metodo *Loess* o *Locally Weighted Regression* .

Il metodo Loess (acronimo di *Locally Estimated Scatterplot Smoothing*), o regressione locale, è un metodo di regressione non parametrica. La regressione locale combina la semplicità della regressione lineare dei minimi quadrati con la flessibilità della regressione non lineare. Tale concetto è stato sviluppato da *William S.Cleveland* nel 1979. La regressione locale è anche conosciuta come regressione polinomiale con ponderazione locale. Per ogni punto del set di dati si determinano i coefficienti di un polinomio di basso grado, eseguendo la regressione su un sottoinsieme di dati. Dunque, non si determina una singola funzione globale che descrive il modello di regressione, ma si calcolano più funzioni locali. Per determinare i coefficienti del polinomio si utilizza il metodo dei minimi quadrati ponderati, che dà più peso ai punti vicini al punto di cui si stima la risposta e meno peso ai punti più lontani. Generalmente si utilizzano polinomi di primo o secondo grado.

Considerando una serie di misurazioni x_i e y_i, con *i = 1…n,* per stimare la variabile y tramite il calcolo della funzione di regressione $g(X)$ di tipo *Loess*, bisogna determinare un valore per il parametro $q \leq n$. Vengono selezionati q valori x_i più vicini a *x.* Ad ognuno di essi viene assegnato un peso sulla base della distanza da *x.* Per ogni *x* si definisce un *"neighborhood weight"* espresso dalla seguente formula:

$$v_i(x) = W\ \frac{|x_i - x|}{\varphi_q(x)}$$

(2. 17)

Con $\varphi_q(x)$ che rappresenta la distanza del q-esimo punto più lontano da *x. W* rappresenta la *"tricube weight function".* Successivamente si adatta un polinomio di grado d (solitamente il grado è pari a uno o due) e considerando il peso $v_i(x)$ si ottiene $g(X)$.

L'algoritmo STL effettua la decomposizione della serie storica nelle sue tre componenti quali il *Trend*, la stagionalità e il residuo attraverso due procedure ricorsive; un ciclo interno annidato e un ciclo esterno. Per quanto riguarda il ciclo esterno, sulla base della dimensione della componente residua, a ciascuna osservazione viene assegnato un peso. L'assegnazione del cosiddetto "*robustness weights*" si basa sul principio in base al quale più un *outlier* presenta un valore della componente residua alto più il peso assegnato sarà più piccolo o pari a zero. Invece, per quanto riguarda il ciclo interno, la componente stagionale e il *Trend* vengono aggiornati in modo iterativo. In primo luogo si sottrae la stima del *Trend* dalla *Raw series.* Successivamente si effettua una suddivisione della serie temporale tramite l'identificazione di *cycle-subseries.* Se i dati sono mensili, e sono relativi a più anni, si

determinano 12 cicli; ognuno di essi sarà caratterizzato dalle osservazioni di uno stesso mese, considerando tutti gli anni. Il processo di *smoothing* tramite *Loess* viene effettuato su ognuna delle 12 *cycle-subseries.* Successivamente viene effettuato un processo di destagionalizzazione, seguito da una fase di *Trend smoothing*; dalla serie destagionalizzata, tramite l'applicazione del metodo *Loess*, si ottiene la componente tendenziale. Infine si calcola la componente residuale, sottraendo la serie originale le componenti di Trend e stagionali precedentemente stimate.

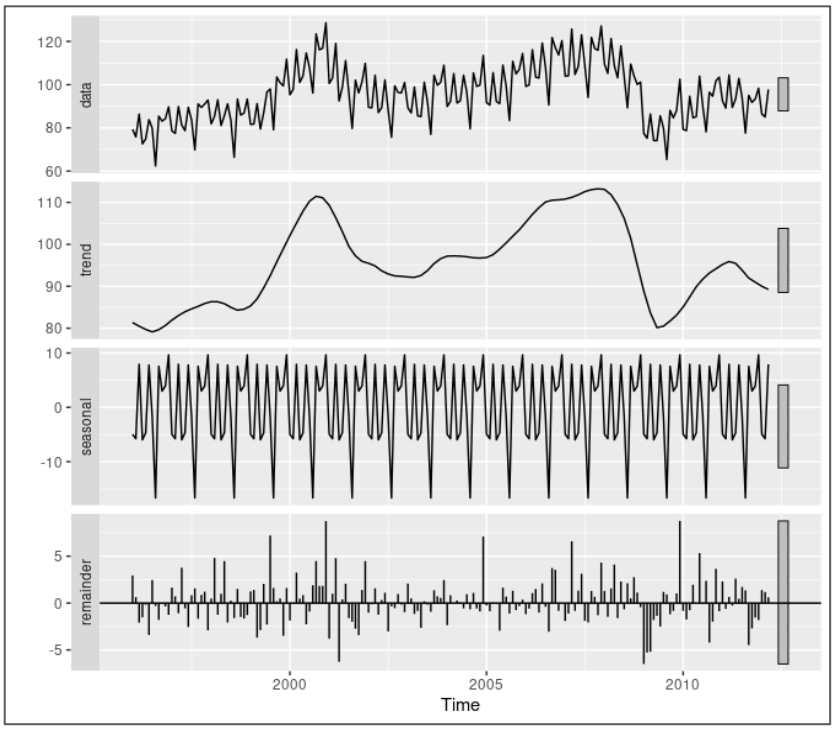

Figura 2.1: Esempio di decomposizione con metodo STL

2.4 Anomaly Detection

Le anomalie sono osservazioni rare e significativamente diverse all'interno di un set di dati. Nel dettaglio le anomalie possono essere definite come osservazioni che si discostano sufficientemente dalla maggior parte delle altre osservazioni tale da destare il sospetto che siano state generate da un processo generativo diverso dagli altri punti del dataset e, dunque, definibile come un processo non normale. Si distinguono differenti tipi di anomalie quali le anomalie puntuali, cioè singole istanze anomale in un set di dati più ampio, anomalie collettive, se l'anomalia comprende un insieme di istanze e, infine, le anomalie contestuali. In quest'ultimo caso il punto in questione risulta essere un'anomalia se inserito all'interno di un determinato contesto.

Sono molti i domini applicativi che richiedono tecniche di rilevamento delle anomalie. Gli algoritmi di *Anomaly detection* vengono utilizzati per il rilevamento delle intrusioni, delle frodi, per la sorveglianza, il monitoraggio e il controllo della qualità dei dati. Alcuni contesti richiedono che il rilevamento delle anomalie sia molto veloce, in *real time,* mentre in altre applicazioni è richiesto un maggior *focus* in termini di prestazione in quanto la mancanza del rilevamento dell'anomalia determina costi molto elevati.

Gli algoritmi di *anomaly detection* si distinguono in metodi di *machine learning* supervisionati, semi-supervisionati, e non supervisionati a seconda del tipo di dato disponibile. Inerentemente agli algoritmi supervisionati, quest'ultimi possono essere utilizzati se sono presenti dati etichettati e dunque se le anomalie sono già note. Esempi di algoritmi supervisionati sono il *Support Vector Machines* (SVM) e l'*Artificial Neural Networks* (ANN). Il rilevamento semi-supervisionato delle anomalie si basa sulla conoscenza del modello dei dati normali e dall'idea che tutto ciò che si discosta da tale modello rappresenta un'anomalia. Dunque, tale metodo utilizza etichette per i dati che sono classificati come normali.

2.4.1 Isolation Forest

L' *Isolation Forest* è un modello di *Anomaly Detection unsupervised.*. Nella pubblicazione del 2008 intitolata "*Isolation Forest*", gli autori *Fei Tony Liu, Kai Ming Ting e Zhi-Hua Zhou*

hanno illustrato le caratteristiche innovative dell'algoritmo ideato, mostrandone il funzionamento. L'algoritmo si basa sulla costruzione di più alberi decisionali similmente al *Random Forest.* L'idea di base è che le anomalie, all'interno di un *dataset*, sono "poche e diverse" rispetto agli altri punti e, dunque, per l'albero decisionale risulta più semplice isolarle dalle altre osservazioni. Pertanto, i *record* che ricadono nei rami più corti rappresentano delle anomalie; al contrario quei campioni che si trovano in profondità dell'albero decisionale non rappresentano anomalie perché hanno richiesto più tagli per poter essere isolati. Di seguito, in figura 2.2, si riporta una raffigurazione di un punto anomalo, facilmente isolabile, mentre in figura 2.3 un punto normale e difficilmente isolabile.

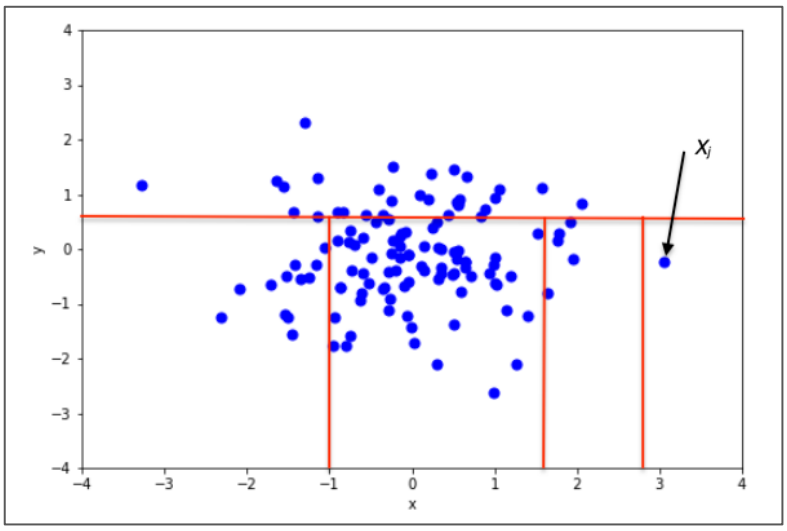

Figura 2.2: esempio di isolamento di un punto anomalo.

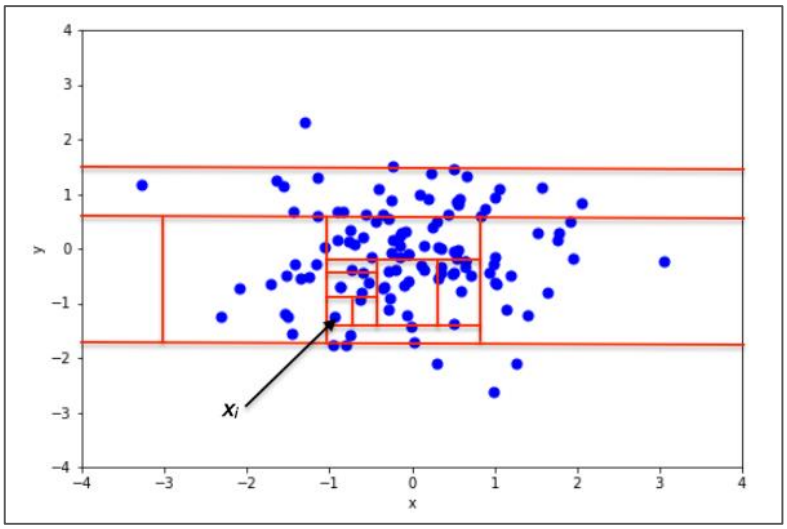

Figura 2.3: esempio di isolamento di un punto normale.

Dunque, la partizione ricorsiva dei punti del dataset, raffigurata a titolo di esempio nelle figure sovrastanti, da un punto di vista matematico, viene rappresentata tramite una struttura ad albero quale *l'Isolation Tree.* Inoltre, il numero di partizioni necessarie per isolare un punto viene interpretato come la lunghezza del percorso necessario per raggiungere un nodo foglia da un nodo radice.

Il processo di *Anomaly Detection* con *l'Isolation Forest* è composto da due fasi principali:

1. Costruzione degli alberi di isolamento tramite un set di dati di *training*
2. Assegnazione dell'"*Anomaly Score*" ai punti di un set di dati di *test,* processati dal modello precedentemente costruito, e determinazione dei punti anomali in base alla definizione di una soglia specifica del dominio applicativo.

Nel dettaglio l'algoritmo esegue differenti passaggi. In primo luogo, viene selezionato un sotto-campione casuale di dati dal *dataset.* Formalmente, dato un set di punti $X = \{x_1, \dots, x_n\}$ di d dimensioni, viene selezionato $X' \subset X$. Considerando le d *features* del *dataset,* la ramificazione dell'albero inizia selezionando in modo casuale una prima caratteristica q e la soglia in base alla quale viene eseguita la ramificazione (valore di split p) viene scelta in maniera casuale (ovvero un valore qualsiasi compreso nell'intervallo costituito dal valore minimo e il valore massimo della caratteristica considerata). Se il valore di q è inferiore alla soglia selezionata, $q < p$, esso andrà a finire nel ramo destro o sinistro dell'albero decisionale. Tale processo si ripete fino a quando ogni punto del *dataset* risulta isolato o al raggiungimento della massima profondità, se definita. Dopo aver creato un insieme di alberi

decisionali binari chiamati "*iTrees*", si determina, appunto, una foresta di alberi (*Isolation Forest*). Di seguito si riporta l'algoritmo che corrisponde alla prima fase del processo di rilevamento delle anomalie, cioè la fase di training.

Algoritmo 1.1 *Isolation Forest*

Fase 1: fase di *training.*

Inputs : dati di input *X* , numero di alberi *t,* dimensione del sotto-campione φ.

Output : set di *t* "*iTrees*".

1.1.1 Inizializzare la Foresta.

1.1.2 Settare il limite di profondità *l* = *celing*($log_2\varphi$).

1.1.3 Per *i=1* a *t* determina:

$X' =$ campione (X, φ)

Foresta = Foresta ∪ "*iTree".*

1.1.4 Termina il ciclo.

1.1.5 Restituisci la Foresta.

Un "*iTree*" è un albero binario in cui ogni nodo dell'albero è costituito esattamente da zero o due nodi figli. Dato un *dataset* $X = \{x_1, \dots, x_n\}$ costituito da *n* istanze, considerando un albero costruito in modo completo, il numero di nodi esterni è pari a *n* mentre il numero di nodi interni è pari a *n-1*. Il numero totale di nodi in un albero binario è pari $2n - 1$; per tale motivo la richiesta di memoria è limitata e cresce linearmente con *n*.

Nella seconda fase del processo a ogni punto viene assegnato un valore pari a 1 o -1. Se il *sample* ottiene il valore 1 significa che rappresenta un dato normale se, invece, ottiene il punteggio pari a -1 il dato rappresenta un valore anomalo. Un punto del dataset viene classificato come anomalo o normale in base al calcolo di un punteggio chiamato "*Anomaly Score*". Tale valore viene assegnato in funzione della profondità raggiunta dal *record* in questione considerando tutti gli alberi della foresta. Dunque, per ogni punto $x_i \in X$, si definisce con $h(x_i)$, la lunghezza del percorso (numero di rami) necessario per raggiungere il nodo foglia, cioè il punto x_i isolato dall'algoritmo, dal nodo radice. Il calcolo dell'"*Anomaly Score*", relativo al punto *x* è espresso dalla seguente formula:

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

(2. 18)

Dove *E(h(x))* rappresenta la media di *h(x)* considerando tutti gli alberi decisionali generati dall'algoritmo e *c(m)* rappresenta la media di *h(x)* dato m. Dal momento che un "*iTree*" presenta la stessa struttura di un albero di ricerca binario, BST (acronimo di *Binary Search Tree*), la stima della media di *h(x)* per un nodo foglia è pari a quella che corrisponde a una ricerca senza successo in un BST. Dunque, *c(m)* può essere utilizzato per normalizzare *h(x)*. Per il calcolo di *c(m)* si utilizza la seguente formula:

$$C(m) = \begin{cases} 2H(m-1) - \ 2(m-1)/n & per\ m > 2 \\ 1 & per\ m = 2 \\ 0 & altrimenti \end{cases}$$

(2. 19)

Dove *n* è la dimensione del dataset di *test,* mentre *m* è la dimensione del set di campioni. *H* è il numero armonico che può essere stimato dall' espressione $H(i) = ln\,(i) + \gamma$, dove γ= 0,5772156649 e rappresenta la costante di Eulero-Mascheroni.

Le anomalie vengono determinate usando i seguenti criteri:

- Se *S(x,m*) è vicino a 1 è molto probabile che *x* sia un'anomalia, in quanto implica una lunghezza del percorso piccola.
- Se *S(x,m)* è minore di 0,5 è molto probabile che *x* sia un punto normale, in quanto implica una lunghezza del percorso grande.
- Se *S(x,m)* è pari a 0,5 per tutti i punti del dataset allora non sono presenti anomalie.

Algoritmo 1.2 *Isolation Forest*

Fase 2: fase di valutazione.

Inputs : istanza x , un albero "*iTree*" *T,* lunghezza del percorso corrente *c.*

Output : lunghezza del percorso di x

1.2.1 Se x è un nodo esterno allora

c+*c(*T.dimensione) dove *c*(.) funzione descritta dall'equazione 2.15

1.2.2 Se *a* è un attributo di split

1.2.3 se $x_a <$ T.$split$.valore

Restituisci LunghezzaDelPercorso(*x*, T.sinistra, *c*+1).

1.2.4 Se $x_a \geq$ T.$split$.valore

Restituisci LunghezzaDelPercorso(*x*, T.destra, *c*+1).

L'*Isolation forest* utilizza un approccio diverso rispetto alle tecniche più comunemente impiegate per il rilevamento delle anomalie. Quest'ultime si basano su metodi *profile-based*, volte, dunque, a determinare un profilo di ciò che è considerato "normale" all'interno del *dataset*. In questi modelli le anomalie sono quelle istanze del *dataset* che non sono conformi a tale "*normal profile"*. Tra questi modelli si individuano, per esempio, i metodi statistici e i metodi basati sui *clustering.* Nella pubblicazione del 2008 gli autori *Fei Tony Liu, Kai Ming Ting e Zhi-Hua Zhou* hanno evidenziato i principali svantaggi legati ad un approccio basato sull'individuazione del cosiddetto "profilo normale". In primo luogo, il risultato de ll'*Anomaly Detection* potrebbe essere caratterizzato da molti falsi allarmi o potrebbe individuare un numero inferiore di anomalie rispetto a quelle realmente tali. Secondariamente, i modelli esistenti hanno un'alta complessità computazionale e sono vincolati all'analisi di *dataset* di piccole dimensioni. Al contrario, l'*Isolation forest* isola in modo esplicito i punti anomali nel *dataset.* I punti di forza dell'algoritmo sperimentato, secondo gli ideatori del modello, risiedono nelle caratteristiche intrinseche dei dati anomali: essi costituiscono la minoranza dei dati e presentano valori degli attributi molto differenti rispetto a quelli assunti dagli altri dati. Nella pubblicazione del 2008 gli autori elencano i principali vantaggi dell'algoritmo in relazione ai modelli *density-based* e *distance-based esistenti*:

- Possibilità di esplorare sotto-campioni di dati (*sub-sampling*): infatti, non risulta necessario, ai fini dell'*Anomaly Detection,,* la costruzione di alberi decisionali

completi che isolano tutti i valori normali. Di conseguenza, l'algoritmo lavora bene quando la dimensione del campione è mantenuta piccola.

- Riduzione dei maggiori costi computazionali legati al calcolo delle distanze, caratterizzanti i modelli *density-based* e *distance-based.*
- Tempo computazionale lineare e basso.
- Capacità di gestire *Dataset* di grandi dimensioni (*High Dimensional Data*): infatti la maggiore limitazione degli algoritmi *distance-based* è la capacità di gestire dataset di grandi dimensioni. La principale motivazione risiede nel fatto che in uno spazio dimensionale di grandi dimensioni, ogni punto risulta ugualmente sparso e, dunque, utilizzare misure di distanza per determinare la separazione è inefficiente.

Un' altra proprietà dell'*Isolation Forest* risiede nella capacità dell'algoritmo di performare correttamente anche in assenza di punti anomali nel *dataset* di training. Infine, le ultime proprietà dell'algoritmo si identificano come "*swamping*" e "*masking*". Per quanto riguarda il fenomeno "*swamping*", si verifica quando le istanze normali sono troppo vicine alle istanze anomale. Di conseguenza il numero di partizioni necessarie per isolare i punti normali cresce e risulta più difficile distinguere quelli normali da quelli anomali. Per quanto riguarda il cosiddetto "*masking*", quando il numero di anomalie è alto, è possibile che esse si aggreghino in cluster molto ampi e risulta più difficile isolare le singole anomalie.

Successivamente, in una pubblicazione del 2012, effettuata dagli stessi ideatori dell'algoritmo (*Fei Tony Liu, Kai Ming Ting* e *Zhi-Hua Zhou),* vengono riportati alcuni esperimenti volti a dimostrare che effettivamente l'*Isolation Forest* presenta i punti di forza sopradecritti.

Tuttavia, una delle principali difficoltà del modello riguardava non il modello di per sé, ma piuttosto il modo in cui veniva calcolato il punteggio per determinare un'anomalia. Nel 2018 è stata proposta un'evoluzione del modello di partenza quale *l'Extended Isolation Forest* (EIF). È stato, inoltre, pubblicato un documento che descrivi i miglioramenti apportati da tale nuovo modello. In particolare, le evoluzioni riguardano la maggiore coerenza e affidabilità nel calcolo dell'"*anomaly score*".

Un vantaggio è che attraverso questo algoritmo è possibile creare modelli molto sofisticati ma allo stesso tempo il numero crescente di *features* può impattare negativamente sulle prestazioni computazionali. In questo caso è necessario effettuare una selezione delle stesse.

2.4.2 Modelli di serie temporali

Questa sezione del capitolo fornisce una spiegazione dettagliata di alcuni modelli sviluppati per identificare le anomalie in dati temporali. Per ogni modello si riportano spiegazioni estratte dalle pubblicazione scientifiche sviluppate dagli stessi autori degli algoritmi.

Joint Estimation of Model Paramaters and Outlier Effects in Time Series

Il primo modello che si illustra è stato sviluppato da *Chung Chen* e *Lon-Mu Liu*, autori della pubblicazione "*Joint Estimation of Model Paramaters and Outlier Effects in Time Series*" . Quest'ultimi hanno proposto un metodo di *Anomaly Detection* iterativo, basato su una procedura che consente di ottenere in modo congiunto la stima dei parametri del modello e degli effetti degli *outliers*, attraverso aggiustamenti successivi. La procedura iterativa si sviluppa attraverso tre fasi principali. La prima consiste nell'identificare tutti i possibili *outliers* in base a una stima preliminare dei parametri del modello. Successivamente si effettua la stima congiunta del modello e degli effetti degli *outliers*, identificati nella fase iniziale. L'ultimo *stage* è volto a identificare nuovamente gli *outliers* e ad effettuare una seconda stima dei loro effetti in base ai parametri stabiliti nella seconda fase. *Chung Chen e Lon-Mu Liu*, per effettuare lo studio, considerano quattro tipi di *outliers: innovational outlier* (IO), *additive outlier* (AO), *level shift* (LS) e infine, *temporary change* (TC).

Considerando $\{Y_t\}$, una serie temporale descritta da un generico processo ARMA:

$$Y_t = \frac{\theta(B)}{\alpha(B)\varphi(B)} a_t$$

(2. 20)

Con *t=1, ... n.* Dove *n* rappresenta il numero di osservazioni della serie temporale, $\theta(B)$, $\alpha(B)$, $\varphi(B)$ sono polinomi di B; tutte le radici di $\theta(B)$, $\varphi(B)$ sono al di fuori del cerchio unitario mentre, quelle di $\alpha(B)$ all'interno. Dunque, quest'ultimo rappresenta un operatore non stazionario. La procedura di stima congiunta del modello, in presenza di molteplici *outliers* è espresso dalla seguente formula:

$$Y_t^* = \sum_{j=1}^{m} \omega_j \, L_j(B) I_t(t_j) + \frac{\theta(B)}{\alpha(B)\varphi(B)} a_t$$

(2. 21)

Dove *m* rappresenta il numero di eventi non ripetitivi, che si verificano negli istanti temporali $t_1, t_2 \ldots, t_m$, determinando la presenza di *outliers*. Il termine $\omega_j \, L_j(B)$ rappresenta l'entità e il *pattern* dinamico che l'*outlier* determina nella serie storica, il cosiddetto "*outlier effect*". In particolare, $L_j(B)$ assume un differente valore a seconda della tipologia di *outlier:* $L_j(B) = (\theta(B))/(\alpha(B)\varphi(B))$ per un *IO,* $L_j(B) = 1$ per un AO, $L_j(B) = \frac{1}{1-B}$ per un LS e, infine, $L_j(B) = \frac{1}{1-\delta B}$ per un TC, per $t = t_j$.

Un *outlier* di tipo AO (*Additional Outlier*) rappresenta un picco isolato, uno di tipo *Transient change* costituisce un picco transitorio che impiega poco tempo per scomparire. Un *outlier* di tipo *Level shift* determina un brusco spostamento del livello medio della serie storica. Infine, un IO (*Innovational Outlier*) impatta fortemente sulla serie storica, andando a determinare un cambiamento brusco e improvviso che non si era mai verificato precedentemente.

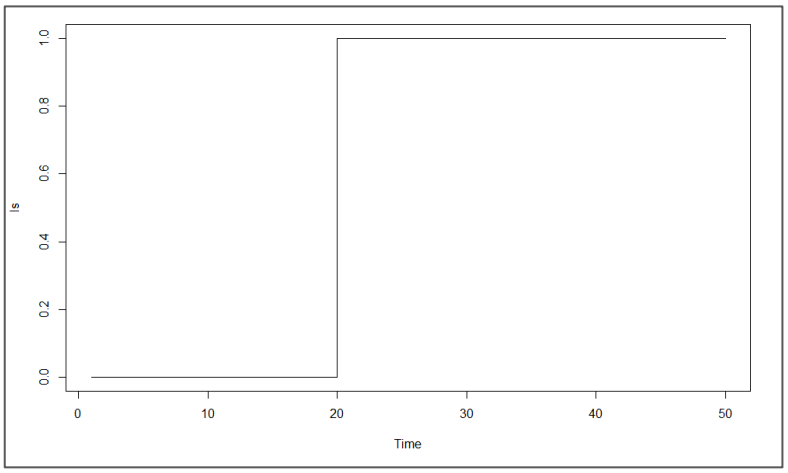

Figura 2.4: Esempio di un *outlier* di tipo *Level Shift* (LS).

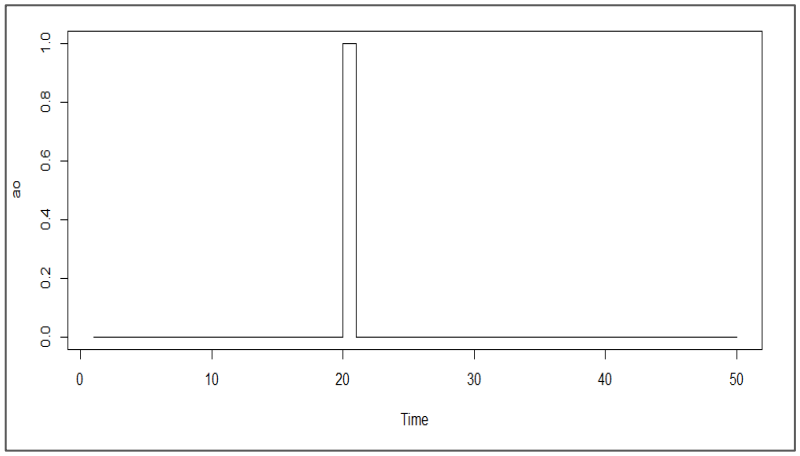

Figura 2.5: Esempio di un *outlier* di tipo *Additional* (AO).

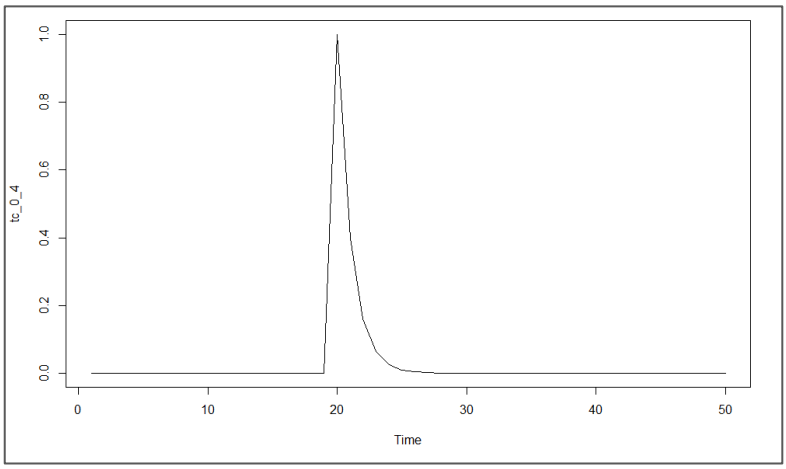

Figura 2.6**:** Esempio di un *outlier* di tipo *Transient Change* (TC).

Il residuo $\hat{e}_t$ per adattare un modello ARMA al modello Y_t^*, può essere espresso come

$$\hat{e}_t = \sum_{j=1}^{m} \omega_j \, \pi(B) L_j(B) I_t(t_j) + a_t$$

(2. 22)

Ogni tipologia di *outlier* impatta in modo differente sulla serie storica, in particolare l'*outlier "temporary change"* (TC), produce un effetto iniziale ω al tempo t_1 che decresce gradualmente con il tempo. Il valore di δ, dipende dal contesto applicativa ma *Chung Chen* e *Lon-Mu Liu* utilizzano, nelle loro analisi, $\delta = 0{,}7$. Inerentemente all' "*additive outlier"* (AO), esso impatta in modo istantaneo sulla serie determinando un effetto di tipo "*one–*

shot". L'*outlier* di tipo "*level shift"* (LS) determina una variazione brusca e permanente nella serie.

L' innovational outlier (IO) ha un effetto più complesso rispetto alle atre tipologie. Se un IO si presenta al tempo $t = t_1$, il suo effetto per $Y_{t_{1+k}}$, per $k \geq 0$, è uguale a $\omega\varphi_k$ che rappresenta il k-esimo coefficiente del polinomio $\varphi(B) = \varphi_0 + \varphi_1 B + \varphi_2 B^2 + \cdots$. Se la serie è stazionaria IO produce un effetto temporaneo in quanto il coefficiente k-esimo del polinomio decresce esponenzialmente. Se la serie non è stazionaria, l'effetto dell'*outlier* (IO) dipende dal tipo di modello Y_t. Se quest'ultimo è un modello ARIMA con parametri (0,1,1), allora determina un effetto iniziale per poi causare uno scostamento di livello, mentre in un modello ARIMA (1,1,1) tale scostamento diventa permanente. Di seguito si riporta una spiegazione dettagliata delle tre fasi dell'algoritmo ideato da *Chung Chen e Lon-Mu Liu.*

Algoritmo 2.1 *Joint estimation procedure in the presence of multiple outliers*

Fase 1: stima iniziale dei parametri del modello e *Outlier Detection*.

2.1.1 Calcolare una stima dei parametri del modello sulla base della serie originale (per la prima iterazione), mentre per le iterazioni successive si considera una serie "aggiustata" sulla base degli *outliers* identificati.

Eseguire *outlier detection, con parametri fissati del modello,* attraverso un Loop

2.1.2 Per *t=1, ... n* calcolare $\hat{\tau}_{TC}(t)$, $\hat{\tau}_{AO}(t)$, $\hat{\tau}_{LS}(t)$, $\hat{\tau}_{TC}(t)$ usando i residui ottenuti al punto 2.1.1, considerare la statistica con il massimo valore in modulo e confrontarla con un valore critico predeterminato *C. L*a statistica, oggetto di confronto, sarà relativa a uno dei quattro tipi di outliers.

2.1.3 Se non è stato trovato un *outlier*, allora si esegue lo step 2.1.4. Se, al contrario, è stato rilevato, si rimuove l'effetto dal residuo (effetto specifico del tipo di *outlier* identificato) e si riesegue lo step 2.1.2 per verificare la presenza o meno di un ulteriore *outlier.*

2.1.4 Se nella prima iterazione del loop non è stato rilevato un *outlier*, allora la serie storica non presenta *outliers* e non presenta effetti distorsivi dovuti alla loro presenza. Se sono stati rilevati outliers allora si ritorna al punto 2.1.1 per ristimare i parametri del modello. Se il numero totale di *outliers*, considerando tutti i loop, è maggiore di 0 e nel loop corrente non si identifica nessun'altro *outlier*, allora si passa alla seconda fase dell'algoritmo, in particolare allo *step* 2.2.1.

Algoritmo 2.2 *Joint estimation procedure in the presence of multiple outliers*

Fase 2: stima congiunta dei parametri del modello e degli effetti degli *Outliers.*

2.2.1 Suppore di aver individuato *m* possibili valori anomali in corrispondenza degli istanti di tempo $\boldsymbol{t_1}$,… $\boldsymbol{t_m}$.$\boldsymbol{L}$'effetto dell'*outlier* j-esimo, cioè il valore di $\boldsymbol{\omega_j}$ può essere stimato congiuntamente utilizzando la formula del residuo, dove $\hat{\boldsymbol{e}}_{\boldsymbol{t}}$ è considerato l'output, mentre $\boldsymbol{L_j(B)I_t(t_j)}$ come variabili di input.

2.2.2 Dopo aver calcolato $\boldsymbol{\omega_j}$, ricalcolare la statistica $\hat{\boldsymbol{\tau}}_{\boldsymbol{j}} = \frac{\hat{\boldsymbol{w}}_{\boldsymbol{j}}}{\boldsymbol{std}(\hat{\boldsymbol{w}}_{\boldsymbol{j}})}$. Se il modulo della statistica considerata è minore o uguale al valore critico *C*, lo stesso del punto 2.1.2, allora si elimina l'*outlier* in questione dal set dei possibili *outliers* e si ritorna al punto 2.2.1 con i rimanenti *m-1 outliers.* Altrimenti si procede allo step 2.2.3.

2.2.3 Si effettua la rimozione dell'effetto dell'*outlier* dalla serie, utilizzando il valore stimato $\boldsymbol{\omega_j}$ più recente, ottenendo una serie "aggiustata".

2.2.4 Calcolare i parametri del modello considerando la serie "aggiustata" ottenuta con lo step 2.2.3. Se l'errore standard del residuo, calcolato prima e dopo aver sottratto dalla serie gli effetti degli outliers, è maggiore di $\boldsymbol{\epsilon}$, si ritorna allo step 2.2.1 per effettuare ulteriori iterazioni, altrimenti si procede alla terza fase. Il valore di $\boldsymbol{\epsilon}$ è scelto dall'analista e serve a controllare l'accuratezza della stima.

Algoritmo 2.3 *Joint estimation procedure in the presence of multiple outliers*
Fase 3: Individuazione degli *Outliers* sulla base dei parametri definitivi del modello. **2.3.1** Calcolare i residui sulla base dei parametri ottenuti al punto 2.4. **2.3.2** Utilizzare i residui ottenuti nello step 2.3.1 e ri-effettuare la prima fase e la seconda con le seguenti modifiche: (a) i parametri stimati usati nel loop dello stage 1 sono fissi e uguali a quelli stimati allo step 2.2.4, (b) gli step 2.2.3 e 2.2.4 sono omessi nello step 2.2.2. La stima di $\boldsymbol{\omega_j}$ che si ottiene nell'ultima iterazione, in corrispondenza dello step 2.2.3 è la stima definitiva dell'effetto dell'*outlier* rilevato.

MSTL: A Seasonal - Trend Decomposition Algorithm For Time Series With Multiple Seasonal Patterns

Un altro modello di *Anomaly Detection* che è stato ideato per le serie temporali si basa su una scomposizione innovativa sviluppata da *Kasun Bandara, Rob J Hyndman, Christoph Bergmeir,* autori della pubblicazione "MSTL*: A Seasonal - Trend Decomposition Algorithm For Time Series With Multiple Seasonal Patterns*". Il metodo proposto si basa su una decomposizione delle serie temporali, denominata MSTL (acronimo di *Multiple Seasonal – Trend decomposition using Loess*), estensione del metodo tradizionale STL. Il nome del metodo, di per sé, è significativo; l'MSTL, infatti, consente la scomposizione di serie temporali caratterizzate dalla presenza di molteplici *patterns* stagionali. Nella maggior parte dei *dataset* reali i dati sono rilevati con un'alta frequenza (giornaliera, oraria o al minuto), da qui la presenza di proprietà più complesse come, ad esempio, cicli stagionali multipli. Secondo gli ideatori del metodo MSTL, i metodi classici, come per esempio l'STL, nonostante siano performanti in molte applicazioni del mondo reale, sono in grado di gestire esclusivamente *time-series* caratterizzate da una singola stagionalità.

Il metodo MSTL applica la procedura di decomposizione STL in modo iterativo al fine di determinare le distinte componenti stagionali presenti all'interno della *time-series.* Spesso, i molteplici *patterns* stagionali interagiscono tra loro; di conseguenza, durante la decomposizione, i cicli stagionali più piccoli possono essere "assorbiti" da quelli più grandi. Per minimizzare tale fenomeno, come secondo step, MSTL ordina in modo ascendente i cicli stagionali identificati. Successivamente, se la serie temporale presenta componenti

stagionali, MSTL applica l'algoritmo STL iterativamente ad ognuna delle frequenze stagionali identificate. Dall'ultima iterazione STL, infine, si ottiene la componente di *trend*. MSTL effettua il calcolo del residuo considerando le distinte componenti stagionali identificate e la componente di *trend* precedentemente stimata. Se al contrario la serie temporale risulta essere *non-seasonal*, MSTL stima direttamente il *trend* della *time-series* e ottiene la componente residuale sottraendo il *trend* dalla serie temporale originale.

Data un'osservazione al tempo t, X_t, MSTL applica una decomposizione additiva, similarmente al metodo STL, e può essere definita nella seguente modalità:

$$X_t = \hat{S}_t + \hat{T}_t + \hat{R}_t$$

(2. 23)

Dove $\hat{S}_t, \hat{T}_t, \hat{R}_t$ rappresentano, rispettivamente, la componente stagionale, di *trend* e il residuo. MSTL estende l'equazione sovrastante includendo *patterns* stagionali multipli come segue:

$$X_t = \hat{S}_t^1 + \cdots \hat{S}_t^n + \hat{T}_t + \hat{R}_t$$

(2. 24)

Dove *n* rappresenta il numero di stagionalità presenti in X_t.

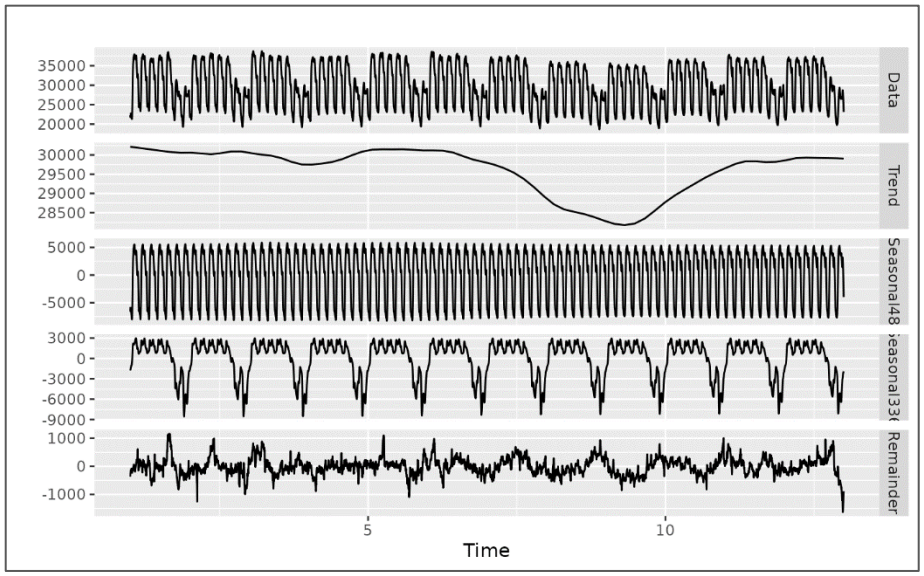

Figura 2.7 : Esempio di decomposizione di una serie storica con il metodo MSTL

Dopo avere effettuato la decomposizione, l'algoritmo prevede il calcolo di un valore che indica la "potenza" della stagionalità (F_s), all'interno della serie. Tale valore soglia è espresso dalla seguente espressione:

$$F_s = 1 - \frac{VAR(y_t - \hat{T}_t - \hat{S}_t)}{VAR(y_t - \hat{T}_t)}$$

(2. 25)

Se $F_s > 0{,}6$ e dunque $\frac{VAR(y_t - \hat{T}_t - \hat{S}_t)}{VAR(y_t - \hat{T}_t)} < 0{,}4$, significa che la serie temporale, oggetto di analisi, presenta una componente stagionale forte. In tal caso si effettua un aggiustamento della serie, sottraendo la componente stagionale e andando a determinare $y_t^* = y_t - \hat{S}_t$. Se invece la componente stagionale è debole o non esistente (nel caso di dati a bassa frequenza), cioè se $F_s \leq 0{,}6$, significa che la stagionalità è stata sovrastimata dalla decomposizione MSTL. Tale condizione potrebbe condurre a un mancato rilevamento di alcuni punti che in realtà sono anomali ma non vengono rilevati come tali perché inclusi nella componente stagionale sovrastimata. Dunque, in tal caso, il modello non prevede aggiustamenti della serie, cioè non viene sottratta la componente stagionale dalla serie temporale e $y_t^* = y_t$.

Successivamente viene effettuata un ulteriore stima della componente di *Trend.* Infine, gli *outliers* vengono rilevati nella componente residuale data da $\hat{R}_t = y_t^* - \hat{T}_t$. Per rilevare gli *outliers* viene applicato il metodo IQR (acronimo di *Interquartile Range*). Il valore assunto da IQR è la differenza tra il 75th percentile (Q_3) e il 25th percentile (Q_1). Si definisce un *lower bound* pari a $Q_1 - (1{,}5 * IQR)$ e un *upper bound* uguale a $Q_3 + (1{,}5 * IQR)$.

Algoritmo 3 *A Seasonal - Trend Decomposition Algorithm For Time Series With Multiple Seasonal Patterns, interquartile range*

Input : serie storica x_t.

Output : valori anomali.

3.1 Scomposizione della serie storica x_t tramite il metodo MSTL.

3.2 Calcolo della potenza della stagionalità F_s , equazione 2.21.

3.3 Se $F_s > 0{,}6$

restituisci $y_t^* = y_t - \hat{S}_t$.

3.3 Altrimenti

restituisci $y_t^* = y_t - \hat{S}_t$.

3.4 Stima della componente di Trend.

3.5 Stima della componente residuale $\hat{R}_t = y_t^*$ - $\hat{T}_t$.

3.6 applicazione del metodo IQR sulla componente $\hat{R}_t$.

Piecewise Median Anomaly Detection

Owen Wallis, Jordan Hochenbaum, Arun Kejarival, autori della pubblicazione *"A Novel Technique for Long-Term Anomaly Detection in the Cloud",* hanno sviluppato una tecnica statistica al fine di rilevare automaticamente le anomalie *long-term* in dati *Cloud*. Tali dati sono caratterizzati da una grande velocità di generazione, in *real-time*, e da grandi volumi. Il metodo è stato implementato considerando la metrica *Tweets Per Sec* (TPS) ma replicabile anche a metriche di sistema quale, per esempio, la CPU *Utilization.* I dati di Twitter, utilizzati per sviluppare tale metodo, presentano sia una rilevante componente di *Trend* sia una componente stagionale che impattano negativamente sull'efficacia dell'*anomaly detection*. Infatti, la presenza di tali componenti conduce, in alcuni casi, al rilevamento di anomalie che non sono realmente tali. Secondo gli ideatori del metodo, le tecniche esistenti nell'ambito dell'*Anomaly Detection* non risultavano efficaci per l'individuazione delle anomalie nelle serie temporali caratterizzate da una componente tendenziale rilevante. Il metodo statistico ESD tiene conto della media campionaria $\bar{x}$ e della deviazione standard per il rilevamento delle anomalie. La distorsione della media campionaria, dovuta alla presenza

delle anomalie nel *dataset*, aumenta con $x_t \to \infty$ (con x_t che rappresenta l'osservazione al tempo *t* all'interno della serie temporale *X*). La tecnica sviluppata, dunque, si basa sull'utilizzo di metriche statistiche quali la mediana e la MAD (*Median Absolute Deviation*). Quest'ultime sostituiscono rispettivamente la media e la deviazione standard utilizzate nel metodo statistico ESD. La MAD è definita come la mediana della deviazione dalla mediana campionaria, in valore assoluto ($MAD = median_i(|x_i - median_j(X_j)|)$.

Algoritmo 4 *Piecewise Median Anomaly Detection*

Input : serie storica x_t, finestre temporali non sovrapposte $w_x(t)$.

Output : valori anomali.

Per ogni $\boldsymbol{w_x(t)}$ determinare:

4.1 n_w = numero di osservazioni nella finestra $w_x(t)$

$$k \leq n_w \times 0{,}49$$

4.2 Estrarre la componente stagionale S_x utilizzando la decomposizione STL.

4.3 Calcolare la mediana $\tilde{\boldsymbol{x}}$.

4.5 Calcolare la componente residuale $R_x = X - S_x - \tilde{X}$.

4.6 Eseguire il test statistico ESD per calcolare il vettore contente le anomalie $x_A = ESD(R_x,K)$ considerando $\tilde{\boldsymbol{x}}$ e MAD nel calcolo del test statistico.

4.7 $v = v + x_A$.

Per verificare l'efficacia del metodo, in assenza di etichette circa la classificazione del dato come anomalo o normale, *Owen Wallis, Jordan Hochenbaum* e *Arun Kejarival* si sono confrontati con alcuni *teams* addetti al servizio Twitter al fine di identificare i falsi positivi e i falsi negativi. L'algoritmo *Piecewise Median* è stato valutato in relazione alle due metodologie standard quali il metodo STL e il metodo *Quantile Regression B spline*. Dal confronto è emerso che circa il 45 % delle anomalie rilevate con il metodo *Piecewise Median* sono state individuate anche gli altri due metodi. Inoltre, *Piecewise Median* riesce a rilevare maggiormente le anomalie presenti all'interno di una giornata ed ha molti falsi positivi in meno rispetto al metodo STL. Inoltre, tale metodo risulta più performante dal punto di vista del *run-time*; impiega, infatti, 4 minuti per analizzare dati campionati al minuto in un

orizzonte temporale di tre mesi, mentre il metodo STL e il metodo *Quantile Regression B spline* circa 13 minuti.

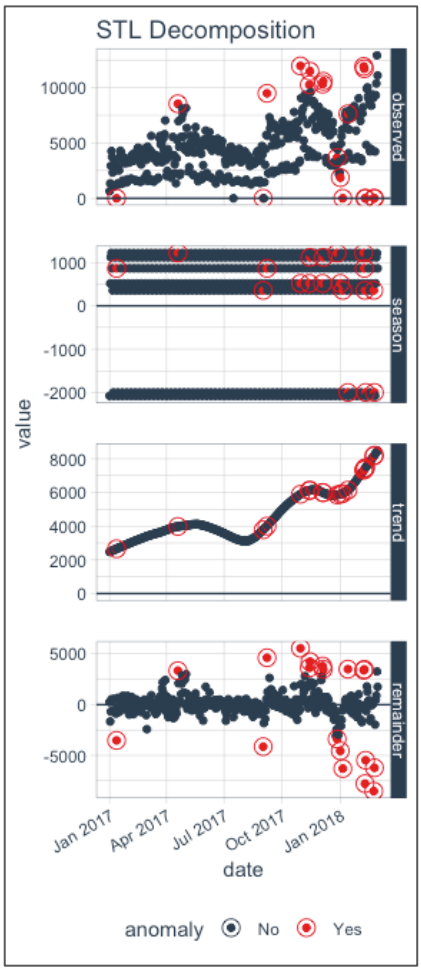

Figura 2.9: Esempio di *Anomaly Detection* con decomposizione STL

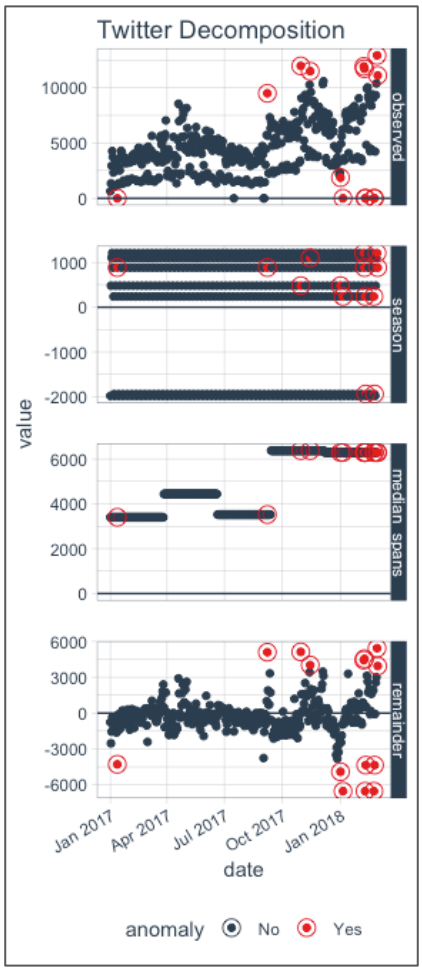

Figura 2.8: Esempio *Anomaly Detection* con decomposizione Twitter

Il metodo GESD (*Rosner* 1983): Il metodo è utilizzato per individuare le anomalie nel contesto di un'analisi univariata. L'assunzione di base è che il *dataset* presenta approssimativamente una distribuzione normale. Tale metodo supera il limite di dover specificare il numero *k* di *outliers* attesi imposto dal *Grubbs test* e dal *Tietjen-Moore test.* Il GESD *test* richiede di impostare un *upper bound r* per il numero di *outliers* attesi e dato *r*, effettua *r* distinti *tests;* un *test* per un *outlier*, uno per due *outlier*s fino a considerare le *r* anomalie.

Il GESD *test* è definito in base alle seguenti ipotesi:

H_0: Non ci sono *outliers* nel *dataset*

H_a: Ci sono fino a r *outliers* nel dataset

Dunque, il test statistico R_i è dato dalla seguente formula:

$$R_i = \frac{max_i|x_i - \bar{x}|}{s}$$

(2. 26)

Con *i*= 1,2, … *r.* Il valore di $\bar{x}$ *,* espresso nella formula sovrastante*,* rappresenta la media*,* mentre s la deviazione standard. Ad ogni iterazione viene rimossa dal *dataset* l'osservazione che massimizza $|x_i - \bar{x}|$ e successivamente, si ripete il calcolo della statistica sopradescritta con *n-1* osservazioni. Il processo si ripete finché non vengono rimosse le *r* osservazioni. Poiché il metodo GESD è iterativo risulta più oneroso del metodo IQR.

Probabilistic Exponentially Weighted Moving Average (PEWMA)

Un metodo per rilevare se un dato è anomalo nel contesto di un flusso di dati ad alto volume, i cosiddetti *data streaming,* è l'EMWA (acronimo di *Exponentially Weighted Moving Average*). Secondo *Kevin M.Carter* e *William W.Streile* in "*Probabilistic Reasoning For Streaming Anomaly Detection"*, l'EMWA risulta efficiente nel processare flussi di dati in *real-time*, ma risulta molto volatile nel caso di cambiamenti bruschi e transitori nei dati, non consentendo un adeguato rilevamento delle anomalie. Al fine di superare tali limiti, *Carter e Streilen* hanno sviluppato un approccio probabilistico al metodo EWMA, quale il metodo PEWMA (acronimo di *Probabilistic Exponentially Weighted Moving Average*)

Un sistema di *Anomaly Detection* volto al monitoraggio di segnali molto numerosi si inserisce nel campo della *streaming analysis.* Nel caso in cui i segnali sono molteplici, dell'ordine di decine di migliaia, e non presentano una classificazione come anomali o normali, è necessario un modello di *Anomaly Detection* che possa essere efficiente in tali condizioni. Secondo gli ideatori del metodo PEWMA altri metodi esistenti, quale, per esempio, il modello ARIMA non si adatta alla natura dei *real-time data streaming*. L'EWMA, invece, è un algoritmo che consente il monitoraggio dello stato di un processo con lo scopo di determinare se le variabili di *performance*, oggetto del monitoraggio, rientrano nei limiti previsti ma non si adatta alla situazione in cui un processo subisce un brusco cambiamento.

I metodi progettati per essere robusti nell'*Anomaly Detection* sono tipicamente *rule-based,* ma nel contesto della *streaming analysis* è necessario un metodo capace di adattare in modo dinamico la definizione dei parametri; dunque, il metodo *Probabilistic* EWMA (o PEWMA) non richiede una parametrizzazione *a priori,* producendo soglie dinamiche e *data-driven.* Inoltre*,* tale metodo, consente di processare un gran numero di segnali con una bassa complessità computazionale. Tale metodologia consente di avvertire la presenza di un'anomalia nel caso in cui il segnale sia caratterizzato da un cambiamento brusco e transitorio ma non risulta adatto nel rilevare la presenza di segnali caratterizzati da variazioni graduali.

L'Exponentially Weighted Moving Average o EWMA calcola la media locale μ_t di una serie temporale X_t con la seguente formula $\mu_t = \sum_{i=1}^{t} \alpha^{t-i} (1-\alpha) X_i$, con $0 \leq \alpha < 1$. Il termine *"Exponentially Weighted"* si riferisce al fatto che il peso α decresce in modo esponenziale man manco che i dati diventano meno recenti. La formula può anche essere espressa nel modo seguente $\mu_t = \alpha\mu_{t-1} + (1-\alpha)x_t$.

Questa equazione si presenta nella forma di un modello autoregressivo di primo ordine. Il modello identifica X_t come un'anomalia nel caso in cui si verifica la condizione per la quale $|X_t - \mu_t| > m\sigma_t$, dove m è una costante e σ_t è la stima della deviazione standard di X_t al tempo *t.* L'EMWA si basa sull'assunzione di base per la quale la media campionaria, che potenzialmente non è stazionaria, può variare solo gradualmente nel rispetto dell'equazione; tuttavia, un cambiamento significativo di X_t*,* come nel caso di un valore anomalo, per un valore di $0 \leq \alpha < 1$ ha un impatto rilevante sul calcolo della media campionaria e ciò viola l'ipotesi di base del modello. La conseguenza è che il modello, nel caso di un brusco cambiamento, perde la capacità di rilevamento delle anomalie.

Il modello *Probabilistic* EWMA pondera il parametro α con la probabilità P_t, con l'assunzione che il segnale abbia una distribuzione normale standardizzata. Ponderando il peso α con il fattore (1 - βP_t), se l'osservazione t-esima ha una bassa probabilità di accadimento, essa avrà un impatto inferiore sulla stima della media. La media μ_t è stimata dal modello PEWMA nel seguente modo:

$$\mu_t = \alpha(1 - \beta P_t)\mu_{t-1} + (1 - \alpha(1 - \beta P_t))x_t \quad \textbf{(2. 27)}$$

Inoltre, l'ipotesi di base è la stazionarietà di X_t; dunque, la stima della media al tempo *t* viene approssimata come l'errore quadratico medio minimo stimato al tempo *t+1*. La deviazione standard locale $\hat{\sigma}_{t+1}$ è stimata come $\sqrt{s_2 - s_1^2}$ e $\hat{X}_t = \mu_{t-1}$. La variabile X_t viene standardizzata $Z_t = \frac{(X_t - \hat{X}_t)}{\hat{\sigma}_t}$.

I parametri del modello vengono inizializzati settando $s_1 = X_1$ e $s_2 = X_1^2$ e effettuando la predizione del secondo valore $\hat{X}_2 = X_1$. Inoltre si definisce un periodo T di "*Train*" in cui i parametri del modello sono settati per calcolare la media e la deviazione standard dei dati. Nel periodo di *"train"* T il peso $\alpha_t = (1 - \frac{1}{t})$, per $t \leq T$.

Algoritmo 5 Probabilistic Exponentially Weighted Moving Average (PEWMA)
Input : X_t, $\hat{X}_t$, $\hat{\sigma}_t$, T, t.
Output : P_t, $\hat{X}_{t+1}$, $\hat{\sigma}_{t+1}$, anomalie.
5.1 $P_t = \frac{1}{\sqrt[2]{2\pi}} \exp(-\frac{z_t^2}{2})$.
5.2 $s_1 = \alpha_t s_1 + (1 - \alpha_t) x_t$.
5.3 $s_2 = \alpha_t s_2 + (1 - \alpha_t) x_t^2$.
5.4 $\hat{X}_{t+1} = s_1$.
5.5 $\hat{\sigma}_{t+1} = \sqrt{s_2 - s_1^2}$.
5.6 $\mu_t = \alpha(1 - \beta P_t)\mu_{t-1} + (1 - \alpha(1 - \beta P_t))x_t$.
5.7 Se $\lvert \hat{X}_{t+1} - \mu_t \rvert > m\hat{\sigma}_{t+1}$ $\hat{X}_{t+1}$ è anomalo.

Shift-detection basato su EWMA (SD-EWMA)

Secondo *Haider Raza, Girijesh Prasad, Yuhua Li*, autori della pubblicazione "*EWMA model based shift-detection methods for detecting covariate shifts in non-stazionary environments*", la maggior parte dei modelli di classificazione si fondano su un'assunzione di base: i dati seguono una distribuzione che soddisfa le condizioni di stazionarietà in tutte le fasi del processo. Quest'ultime comprendono la fase di generazione del modello, le fasi di *test* e, successivamente, anche quelle operative. Tuttavia, nel contesto dei *data streaming*, l'ipotesi di stazionarietà non è sempre valida. Nel caso in cui non si dispongono informazioni

a-priori circa le caratteristiche del processo di generazione dei dati sono necessari *test* d'ipotesi per verificarne la stazionarietà.

Un modello di classificazione di *time-series*, in cui il flusso dei dati è molto elevato, in condizioni di assenza di stazionarietà, deve essere efficiente dal punto di vista computazionale e, allo stesso tempo, deve essere in grado di rilevare in tempo reale variazioni del *dataset* (il cosiddetto "*dataset shift-point")*. Negli ambienti non stazionari tali spostamenti, "*shifts*", possono presentarsi in tre differenti forme quali brusco, transitorio e graduale. Identificare tali variazioni significa rilevare delle anomalie. Dunque, la *shift - detection*, può essere classificata come *retrospective detection* e *online, o real-time detection*. Per valutare la correttezza del metodo di rilevazione delle anomalie si considerano due aspetti principali: il ritardo nell'identificazione del punto anomalo e l'accuratezza.

Il metodo *online* di *shift -detection* può essere applicato in vari campi come nel caso del rilevamento delle intrusioni nella rete. In aggiunta, gli algoritmi di *shift – detection* possono essere attivi o passivi. Il metodo attivo effettua una valutazione del tempo e della severità dello spostamento, subito dal processo, e, se necessario, applica un modello per rilevare le anomalie. Un modello di classificazione che sfrutta un metodo passivo, si basa sull'assunzione dello stato di continuo mutamento caratterizzante l'ambiente. Pertanto, quest'ultimo metodo apprende dall'ambiente e aggiorna la sua conoscenza di base. Nei metodi di *shift – detection* ci sono due tipi di test: test statistici parametrici e non parametrici. Quelli parametrici sono ad esempio il *test* t di *Student* e il test di *Fisher*. Questi *test* si basano sulla valutazione della variazione della media e della varianza di un processo. Un test parametrico generalmente richiede una stima della funzione di densità di probabilità del processo di generazione dei dati precedente e successiva alla *shift-detection*. Nei test non parametrici non è necessaria una solida informazione *a-priori*.

Alcuni metodi si sono dimostrati performanti nell'effettuare *shift-detection*, tra questi si inserisce la carta di controllo *Shewart* che si basa sulla valutazione di una singola osservazione per volta. Inoltre, in questo contesto, si inserisce anche il metodo *Cumulative Sum* (CUSUM); viene effettuato il calcolo di una somma cumulativa sequenziale e quando la somma supera i limiti predefiniti allora viene rilevato lo "*shift*". Il *computanional intelligence* CUSUM (CI-CUSUM) è un metodo che è stato sviluppato per adattarsi agli ambienti *just-in –time* (JIT) e non stazionari. Tuttavia, il metodo CI-CUSUM non è accurato in quanto comporta ritardi e falsi allarmi nel rilevamento delle anomalie. L' *Intersection of*

Confidence Interval (ICI), invece, si basa su una struttura gerarchica e si è dimostrato migliore in termini di falsi allarmi rilevati ma, allo stesso tempo, non supera la problematica relativa al ritardo nell'identificazione delle anomalie. Altri approcci statistici sono basati, per esempio, sulle reti neurali e risentono delle stesse problematiche. Tuttavia, questi metodi non sono robusti in quanto dipendono da modelli parametrici predefiniti come la distribuzione di probabilità sottostante, oppure i modelli autoregressivi che tracciano alcune statistiche come la media e la varianza. La principale problematica delle metodologie sopradescritte è il ritardo nel rilevamento delle anomalie; ciò trova limitazioni nell' applicazione dei problemi riguardanti i *data streaming*. Inoltre, un'altra problematica è la generazione dei falsi allarmi che rappresenta un ostacolo nelle applicazioni reali.

L'approccio proposto, quale l'SD-EWMA, secondo gli ideatori stessi del metodo, presenta alcuni punti di forza; risulta efficiente dal punto di vista computazionale, richiede meno memoria, si basa sul rilevamento della non stazionarietà, riduce il tempo di identificazione delle anomalie e il numero di falsi allarmi. La formulazione proposta nasce per effettuare *shift detection* nell'ambito di dati univariati e multivariati. Questo metodo di *shift detection* si fonda su un modello EWMA.

In relazione ai modelli di classificazione, il cosiddetto *dataset shift* è categorizzato in tre differenti tipologie quali la *covariate shift*, *prior probability shift* e *concept shift*. Nel definire modelli di classificazione si considera un problema descritto da un *set* di *features* o *inputs x* e una variabile target *y.* Il *covariate shift* si verifica nel caso in cui la probabilità condizionale nei dati di *training* e di *test* rimane invariata ($P_{train}(Y/x) = P_{test}(Y/x)$), ma si verifica un cambiamento della distribuzione di input *P(x)* cioè $P_{train}(x) \neq P_{test}(x)$. Considerando dati di *training* caratterizzati da una distribuzione normale, se la media dei dati di *test*, anch'essi costituiti da una distribuzione normale, si discosta da quella del *dataset* di *training* allora il risultato *è il covariate shift.*

Il modello di *shift-detection* basato su EMWA (SD-EMWA) lavora in due fasi. La prima è una fase di *training*, mentre la seconda è una fase di *testing.* Nel primo step si effettua il calcolo dei parametri al fine di stabilire le condizioni di stazionarietà, definendo un'ipotesi nulla (ipotesi di stazionarietà). Nella fase di test avviene il monitoraggio del processo attraverso la carta di controllo EMWA. Quando un punto ricade al di fuori dei limiti di controllo, vuol dire che quel punto è anomalo (punto di *covariate shift*). In altri termini, se il processo supera i limiti imposti significa che non è più in condizioni di stazionarietà e

viene rigettata l'ipotesi nulla a favore dell'ipotesi alternativa che determina l'anomalia del punto considerato.

Le carte di controllo sono strumenti che consentono di determinare se il processo si trova in uno stato di controllo statistico. Generalmente sono rappresentate da tre linee plottate lungo gli assi orizzontali. La linea centrale corrisponde al valore *target* (μ), il limite di controllo superiore (UCL acronimo di *upper control limit*) è dato da $UCL = \mu + L\sigma$, mentre il limite inferiore (LCL acronimo di *lower control limit*) è espresso da $LCL = \mu + L\sigma$. Il termine $L\sigma$, dove L è una costante e σ la deviazione standard del processo, rappresenta la deviazione consentita dal valore target, affinché il processo sia in condizioni di controllo.

Il modello EWMA è definito come $z_{(i)} = \alpha x_{(i)} + (1-\alpha) z_{i-1}$. Per maggiori dettagli sul modello EWMA, riferirsi alla sezione 2.3.4. Se il dataset contiene una sequenza di osservazioni $x_{(i)}$ autocorrelate, allora il modello EMWA può essere utilizzato per effettuare una previsione, cioè determinare $\hat{x}_{i+1}$. Le carte di controllo EMWA sono utilizzate sia per i dati non correlati sia per quelli correlati, ma nelle reali applicazioni in cui i processi non sono stazionari, i dati spesso sono caratterizzati da correlazioni. L'osservazione $x_{(i)}$ può essere definita da un modello ARIMA e, dunque, dalla seguente equazione:

$$x_{(i)} = x_{(i-1)} + \varepsilon_i - \theta \varepsilon_{i-1} \tag{2. 28}$$

Dove ε_i rappresenta una sequenza di variabili casuali con media pari a zero e varianza costante. Inoltre, vale la relazione $\theta = 1 - \alpha$. Il valore di z_{i-1} rappresenta la predizione, dallo stato *i-1,* per il valore di $x_{(i)}$. Dunque, l'errore di previsione è definito come $err_i = x_{(i)} - z_{(i-1)}$. Inoltre, i limiti di controllo superiore e inferiore della carta EWMA sono definiti come:

$UCL = z_{(i-1)} + L\sigma_{err(i-1)}$

$LCL = z_{(i-1)} - L\sigma_{err(i-1)}$

Dove la varianza dell'errore può essere stimata in vari modi, come, per esempio, attraverso la MAD (*Mean Absolute deviation*). L'algoritmo sviluppato da Haider Raza, Girijesh Prasad, Yuhua Li è espresso in modo formale nella seguente modalità:

Algoritmo 6.1 Shift-detection basato su EWMA (SD-EWMA)

Fase 1: fase di *training.*

6.1.1 Assegnare i dati di training a $x_{(i)}$ per *i = 1, ... n* dove *n* è la dimensione del dataset di *training*.

6.1.2 Calcolare la media dei dati di input e assegnarla a $z_{(0)}$.

6.1.3 Per ogni $x_{(i)}$per *i = 1, ... n* e per un range di valori di α,calcolare $z_{(i)} = \alpha x_{(i)} + (1-\alpha) z_{i-1}$.

6.1.4 Calcolare l'errore di previsione, per ogni previsione *one-step-ahead*

$err_i = x_{(i)} - z_{(i-1)}$.

6.1.5 Stimare α minimizzando la somma del quadrato dell'errore di predizione nei dati di *training*.

6.1.6 Calcolare la somma dell'errore di predizione quadratico dividendolo per il numero di osservazioni e utilizzare tale valore come valore iniziale della varianza dell'errore.

Algoritmo 6.2 Shift-detection basato su EWMA (SD-EWMA)

Fase 2: fase di *test.*

6.2.1 Per ogni a $x_{(i)}$ appartenente al dataset di test,

6.2.2 Calcolare $z_{(i)} = \alpha x_{(i)} + (1-\alpha) z_{i-1}$.

6.2.3 Calcolare $err_i = x_{(i)} - z_{(i-1)}$.

6.2.4 Calcolare la stima della varianza $\sigma^2_{err(i)} = \theta err^2_{(i)} + (1-\theta)\sigma^2_{err(i-1)}$.

6.2.5 Calcolare $UCL_{(i)}$ *e* $LCL_{(i)}$.

6.2.6 $UCL_{(i)} = z_{(i-1)} + \mathrm{L}\sigma_{err(i-1)}$.

6.2.7 $LCL_{(i)} = z_{(i-1)} - \mathrm{L}\sigma_{err(i-1)}$.

6.2.8 IF $(LCL_{(i)} < x_{(i)} < UCL_{(i)})$.

THEN (continuare a processare nuovi record)

ELSE (individuazione di un punto anomalo, *covariate shift*, e intraprendere azioni correttive)

Dunque, tale modello, è caratterizzato da due fasi: *training e test. Nella fase di training* si definiscono i parametri del modello mentre la fase di *test* consente di identificare i dati anomali, sfruttando la carta di controllo EMWA; infatti, per verificare se un dato è anomalo si verifica se ricade all'interno del limite superiore e inferiore della carta considerata. Se ciò non accade è necessario intraprendere azioni correttive.

3 Tecnologie utilizzate

In questo breve capitolo si riporta una descrizione delle principali tecnologie utilizzate nel progetto di tesi. L'obiettivo del capitolo è di informare sui linguaggi di programmazione e sulle librerie adottate per effettuare l'analisi dei dati.

3.1 Python

Python è un linguaggio di programmazione ad oggetti di "alto livello" e multi-paradigma. Esso consegue tre obbiettivi principali quali dinamicità, semplicità e flessibilità. Tale linguaggio offre librerie specializzate per affrontare problemi di analisi dei dati, inoltre, per ognuna di esse, garantisce un'ampia documentazione. Per la sua vasta flessibilità, Python è stato utilizzato per effettuare le analisi di preelaborazione dei dati illustrate nel capitolo 4.

- Pandas
  Libreria open-source che consente di effettuare l'analisi e la manipolazione dei dati.

- Numpy
  Libreria open-source che consente di effettuare calcoli scientifici, lavorando in modo efficiente su strutture dati e array multidimensionali. Ampiamente utilizzata nel campo della scienza e dell'ingegneria.

- Seaborn
  Libreria che offre funzionalità per visualizzare i dati, tramite interfacce di alto livello. Nello specifico, Seaborn consente di visualizzare grafici statistici.

- Matplotlib
  Libreria che consente di creare grafici statici, animati e interattivi

- Panel
  Libreria open-source che consente di creare web-app e *dashboards* interattive e customizzate connettendo i *widgets* definiti dall'utente a grafici, immagini ecc.

- Scipy
  Libreria open – source che include moduli statistici, matematici, algebra lineare ecc.

- Statsmodels
  Libreria che fornisce funzioni per la stima di modelli, test e esplorazione di dati statistici.

3.2 Software R

Il software R è un linguaggio di programmazione ad oggetti specifico per la statistica e per la grafica computazionale. Nel progetto di testi, il software R è stato utilizzato per implementare i modelli di *Anomaly detection,* descritti nel capitolo 5. Di seguito una breve illustrazione delle principali librerie utilizzate.

- Lubridate :
  funzione per effettuare un'analisi rapida e intuitiva di dati temporali. Consente l'estrazione e l'aggiornamento di componenti di dati temporali, come per esempio anni, mesi, giorni ecc.

- Readxl :
  pacchetto progettato per recuperare dati da Excel.

- Tidyverse :
  include molteplici pacchetti per effettuare analisi dei dati. Tra questi ggplot2 che rappresenta un sistema per creare grafici e dplyr che consente di effettuare la manipolazione dei dati.
- Timetk :
  pacchetto progettato per poter lavorare con le serie temporali.

- Tsoutliers :

pacchetto che implementa la procedura basata sull'approccio descritto da *Chen e Liu* (2013), descritto nel capitolo 2 , per rilevare automaticamente le anomalie nelle serie temporali.

- Forecast :
  include strumenti per visualizzare e analizzare serie temporali univariate.

- Anomalize :
  pacchetto che consente di rilevare le anomalie nei dati, effettuando la decomposizione delle serie temporali.

- Otsad :
  consente di implementare algoritmi per il rilevamento delle anomalie. Tra i vari modelli consente di implementare il metodo PEWMA e SDEWMA, descritti nel capitolo 2.

- Isotree :
  pacchetto che consente di implementare il modello di *Anomaly Detection,* quale l' *Isolation Forest*, riportato nel capitolo 2.

4 Preelaborazione e selezione delle metriche di *performance*

La preelaborazione dei dati è una fase essenziale in un processo di *data mining*; infatti, analizzare dati che non sono stati attentamente esaminati può produrre risultati fuorvianti. Nel contesto della preelaborazione dei dati, si colloca la selezione delle *features;* la scelta può basarsi sia su considerazioni statistiche, come, per esempio, l'analisi di correlazione, sia su valutazioni circa l'obiettivo di *Business* dell'analisi. Inoltre, *l'Exploratory data analysis* (EDA) consente di esplorare e analizzare il *dataset* tramite grafici.

Il seguente capitolo mostra, in primo luogo, le principali caratteristiche del *dataset*, oggetto di analisi, seguito da un' illustrazione delle principali tecniche utilizzate per effettuare la preparazione dei dati. Infine si affronta la tematica della selezione delle *features*, applicata alle metriche di *performance* del *dataset*. Dunque, le ultime sezioni del capitolo sono dedicate alle metodologie a ai risultati ottenuti nel progetto di tesi per effettuare la *feature selection.*

4.1 Costituzione del dataset

Il *dataset*, oggetto di analisi, è costituito da metriche di *performance* appartenenti alla vista gv$sysmetric. Tali metriche di *performance* consentono il monitoraggio dei *databases* Oracle, da parte dei dipendenti dell'azienda *Mediamente Consulting*. Il controllo viene effettuato al fine di identificare i colli di bottiglia delle prestazioni. A livello generale le metriche del *dataset* consentono di monitorare alcuni aspetti. Tra questi vi è il monitoraggio di problemi di query e istruzioni Sql. Inoltre, altre metriche sono volte all'identificazione di problemi di capacità, come l'utilizzo della Cpu, i dischi o la memoria insufficiente. Inoltre, si monitorano problematiche legate ai conflitti tra gli utenti; se molti accedono contemporaneamente allo stesso *database*, i blocchi transazionali e i "deadlock" interrompono le operazioni in modo brusco. Infine, altre metriche sono volte al monitoraggio dei problemi di configurazione, come la cache dei dischi insufficiente.

Le metriche del *dataset* in totale sono 161 e sono suddivise per tipologia, nello specifico le categorie sono 7: Cpu*,* Cache*,* Debug*,* Enqueue, I/O, Rac, Sql. Di seguito si riporta una breve descrizione delle 7 categorie di metriche, oggetto di studio:

- **Cpu:** Le metriche appartenenti alla categoria Cpu, sono relative all'utilizzo della Cpu. Per esempio la metrica "*host cpu utilization %"*, rappresenta la percentuale di utilizzo della Cpu in relazione a uno specifico *host,* mentre la metrica "*Database CPU Time Ratio*" si ottiene dividendo la quantità di Cpu utilizzata in un *database* e il tempo totale speso dal *database* stesso.
- **Cache:** Le metriche appartenenti alla categoria Cache si riferiscono all'utilizzo della cache. Quest'ultima è utilizzata dai *databases* Oracle per immagazzinare blocchi di dati provenienti dai dischi. Tra le metriche appartenenti alla categoria Cache si distingue, per esempio, la metrica " *buffer cache hit ratio*". Il monitoraggio di tale metrica consente di massimizzare l'utilizzo dei *buffers.*
- **Debug:** Le metriche appartenenti a tale categoria si riferiscono alla possibilità di eseguire il debug di un *database*. Tra le metriche si identifica, per esempio, *"Current OS Load"* che rappresenta il numero di processi che sono stati eseguiti dalla Cpu. Tale metrica consente di cogliere se la Cpu è sovraccarica.
- **Enqueue:** *Enqueues* rappresentano strutture di memoria condivisa (blocchi) che serializzano l'accesso alle risorse del *database*. Per esempio la metrica *"Enqueue Deadlocks Per Sec"* rappresenta il numero di volte al secondo in cui un processo ha rilevato un potenziale "*deadlock*", durante lo scambio di due *buffer*s, generando un errore interno riavviabile.
- **I/O:** Le prestazioni di molte applicazioni software sono intrinsecamente limitate dall'input/output (I/O) del disco. Tra le metriche vi è, per esempio, "*Disk Sort Per Sec*" che rappresenta il numero di ordinamenti sul disco al secondo per il periodo di campionamento.
- **Rac:** Le metriche appartenenti a tale categoria monitorano i *databases* di tipo Rac. Quest'ultimi consentono di eseguire un unico *database* Oracle su più *server* massimizzando la scalabilità orizzontale.
- **Sql:** Tra le metriche di questa categoria si identifica, per esempio, la metrica **"***Average Active Sessions"* che rappresenta la media delle sessioni attive in un determinato momento**.** Un'altra metrica di tipo Sql è *"Library Cache Hit Ratio"* che rappresenta l'efficienza della *cache* della libreria delle richieste Sql condivise.

La maggior parte delle metriche del *dataset* appartengono alla categoria Sql con un totale di 42, seguita dalla categoria Cache con 35 metriche e, successivamente, dalla tipologia I/O con 33 metriche. Per quanto riguarda la tipologia User, il *dataset* risulta caratterizzato da 19

metriche. La tipologia Rac, invece, presenta 10 metriche di *performance*, mentre Enqueue e Debug ne presentano 8, seguite dalla tipologia Cpu con 6 metriche.

Per l'estrazione e la raccolta dei dati si è partiti dalla vista gv$sysmetric. Nello specifico i dipendenti dell'area infrastruttura hanno creato una tabella con il medesimo nome della vista , GV_SYSMETRIC. Il processo di storicizzazione è stato necessario in quanto la vista non consentiva la visualizzazione dello storico dei valori assunti dalle metriche, offrendo esclusivamente la fotografia dello stato attuale. Con la storicizzazione è stato ottenuto un *dataset* caratterizzato da 162 colonne. Tra queste, 161 corrispondono alle metriche, mentre una colonna contiene l'informazione temporale *"sample_time"*. Infine, Il *dataset* è costituito da 79199 righe. La vista è stata storicizzata con una frequenza di campionamento al minuto.

Per ogni metrica del *dataset*, i dati sono *time series* con granularità al minuto. Lo *span* temporale ricoperto da ognuna di esse è di due mesi; la prima osservazione è stata registrata il 19-01-2022 alle ore 00:00:00 mentre l'ultima il 14-03-2022 alle 23:58:00. A titolo d'esempio si riporta la rappresentazione dell'andamento nel tempo di una metrica del *dataset* cioè *"Background Cpu Usage Per Sec"*.

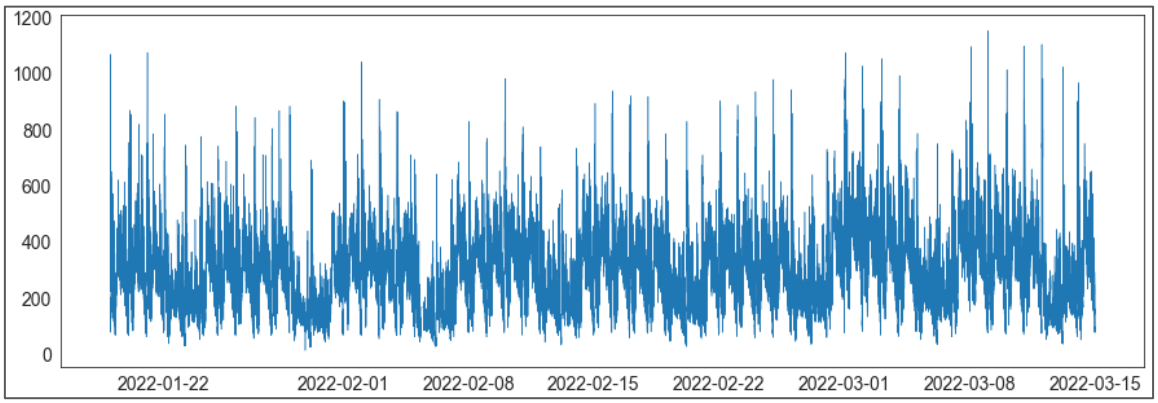

Figura 4.1: *Background CPU Usage Per Sec*. Esempio dell'andamento nel tempo di una metrica di *performance* del *dataset.*

Inoltre, in figura 4.2 si riporta l'immagine della stessa metrica con i dati aggregati a 15 minuti. Data la granularità al minuto, la lettura del *dataset* non è risultata immediata. Per tale motivazione e per avere una maggiore chiarezza dell'andamento della metrica sono stati analizzati anche i dati aggregati tramite un *reseampling* a 15 minuti. Anche se risulta utile esaminare dati maggiormente visibili e leggibili, nel contesto dell'*anomaly detection* in *real-time* si è scelto di procedere con l'analisi al minuto.

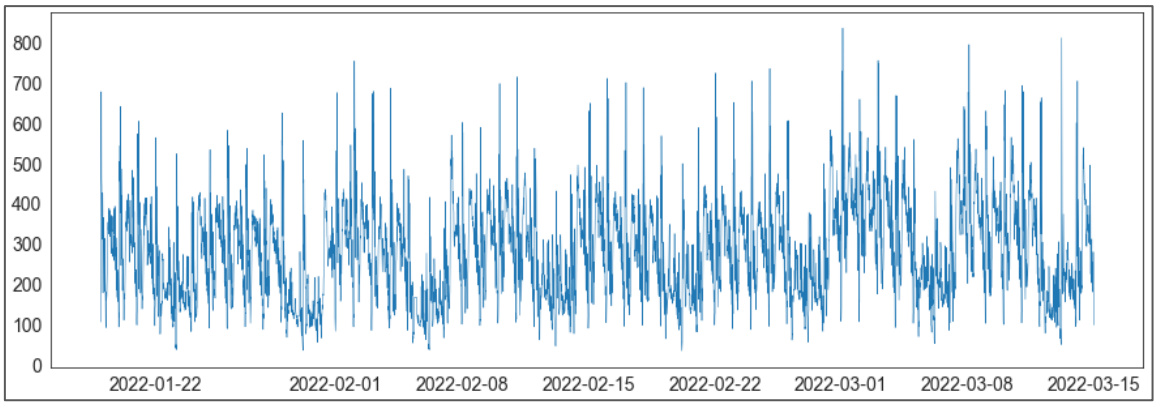

Figura 4.2 : *Background Cpu Usage Per Sec*. Andamento nel tempo della metrica di *performance* con *reseampling* dei dati a 15 minuti.

Come ulteriore analisi esplorativa, è stata effettuata anche un'aggregazione giornaliera. A titolo d'esempio, in figura 4.3, si riporta la raffigurazione della metrica *"Background Cpu Usage Per Sec",* con i dati raggruppati a 15 minuti (andamento rappresentato con il colore blu) e al giorno (andamento rappresentato con il colore arancione).

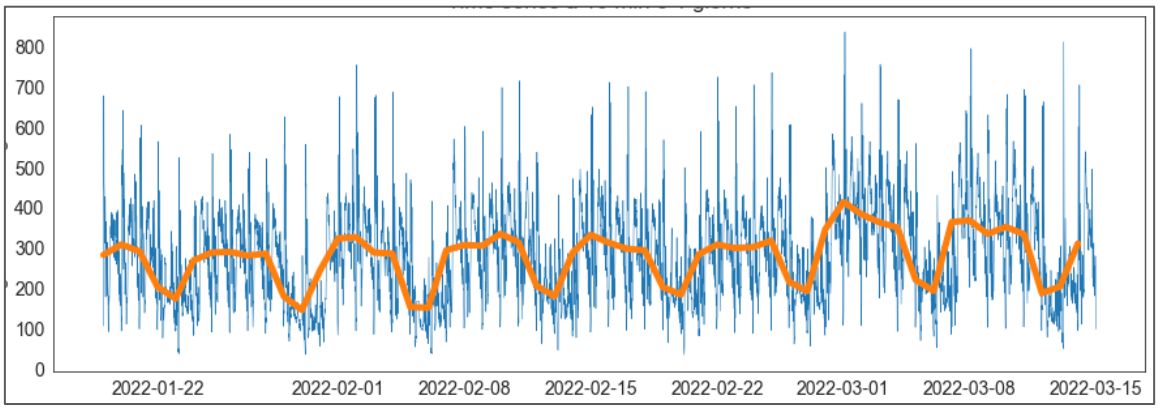

Figura 4.3: *Background Cpu Usage Per Sec*. Andamento nel tempo della metrica di *performance* con *reseampling* dei dati a 15 minuti, in blu, e giornaliero, in arancione.

4.2 Preparazione dei dati

La preparazione dei dati è il processo di pulizia e di trasformazione dei dati grezzi prima dell'elaborazione e dell'analisi. Una buona preparazione dei dati consente di ottenere un'analisi più efficiente, limitando gli errori e le imprecisioni. Nel progetto di tesi, la prima fase del processo di preparazione dei dati si è basata su un'operazione di pulizia seguita da un processo di standardizzazione.

4.2.1 Pulizia dei dati

Il processo di *data cleaning* garantisce, con una certa soglia di affidabilità, la correttezza dei dati analizzati. Effettuare una buona pulizia dei dati significa elevare di un certo livello la qualità dei dati in esame. In primo luogo sono state eliminate le metriche con valori sempre nulli. Le analisi hanno condotto all'esclusione di 6 metriche caratterizzate da valori "NULL". Le successive fasi di analisi hanno condotto all'esclusione di altre *features*; sono state eliminate metriche correlate ad altre e metriche con valori costanti. Le considerazioni e i risultati relativi a tali analisi si riportano rispettivamente nelle sezioni 4.3.1 e 4.3.2.

L'eliminazione delle metriche con valori nulli ha determinato una prima riduzione del *dataset:* da 161 metriche a 155. Le metriche che sono state eliminate sono le seguenti: *"Captured user calls*", *"Replayed user calls"* e "*Workload Capture and Replay status*" appartenenti alla categoria User e *"VM out bytes Per Sec", "Streams Pool Usage Percentage"* , "*Global Cache Blocks Corrupted*", appartenenti alla categoria Cache.

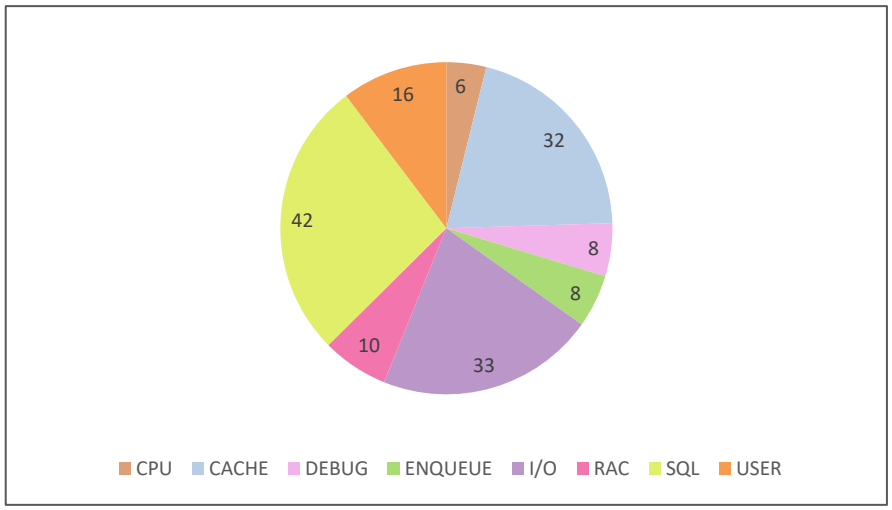

Figura 4.4 : Conteggio delle metriche per tipologia, in seguito all'eliminazione delle metriche con valori nulli.

L'immagine sovrastante raffigura il conteggio delle metriche suddivise per tipologia. Tale rappresentazione grafica consente di comprendere in modo intuitivo la suddivisione del *dataset*. In particolare, si evince una forte disomogeneità: infatti, la categoria che comprende più metriche resta Sql con un totale di 42 (26,09% del totale delle metriche). Le uniche metriche eliminate appartengono alla categoria User che si è ridotta da 19 metriche a 16 e alla categoria Cache che si è ridotta da 35 a 32 metriche.

4.2.2 Ridimensionamento dei dati

Il ridimensionamento dei dati o *feature scaling* è un processo che generalmente viene eseguito in fase di preelaborazione dei dati. Si definisce anche come *Attribute Transformation* ; trasformare un attributo significa applicare una funzione che mappa i valori assunti dallo stesso in una scala predefinita. Generalmente, la trasformazione si applica in quei modelli che sfruttano il calcolo della distanza euclidea tra due punti. Infatti, se un attributo assume un'ampia gamma di valori, la distanza sarà regolata da quello specifico attributo. Inoltre è utile applicare la trasformazione anche nel caso in cui i dati presentano valori o unità molto variabili.

Il lavoro di tesi si è basato sull'analisi di dati che possono essere definiti come *Big Data;* infatti, il *dataset* è costituito da 155 metriche di *performance*, con circa 80000 *record* per ognuna di esse. La vastità del *dataset* richiede di per sé un lavoro di esplorazione

significativo e oneroso. Il ridimensionamento del *dataset* è stato effettuato, dunque, per facilitare il confronto tra le metriche, risultando utile, essenzialmente nella fase di esplorazione dello stesso. Analisi preliminari del *dataset*, infatti, hanno rilevato come la gamma di valori assunti dalle metriche di *performance* fosse ampiamente variabile.

Si è scelto di utilizzare, nel lavoro di tesi, la normalizzazione z-score. Tale metodo di trasformazione fa sì che tutti i valori assunti da ciascuna *feature* (metriche di *performance*) del *dataset* abbiano media zero e varianza unitaria. Dunque, per ogni valore assunto da una metrica, la standardizzazione si effettua sottraendo il valore medio e dividendo il valore ottenuto per la deviazione standard. Per maggiori dettagli sulla formula utilizzata riferirsi alla sezione 2.2 della tesi.

4.3 Selezione delle metriche di *performance*

La selezione di un *subset* di *features* rappresenta una modalità di riduzione della dimensione del *dataset.* Il fine dell'analisi, proposta nella sezione in questione, è quello di identificare le metriche di *performance* ridondanti e irrilevanti. Una *feature* è ridondante se le informazioni contenute al suo interno replicano quelle presenti in altri attributi; per perseguire tale obiettivo si è scelto di effettuare un' analisi di correlazione tra le 155 metriche di *performance* del *dataset* . Per identificare le *feature* irrilevanti ai fini dell' *anomaly detection* sono state effettuate analisi univariate, al fine di individuare metriche con andamenti costanti. A tal fine è stato esplorato il *dataset* attraverso la caratterizzazione delle distribuzioni delle metriche di *performance* selezionate a valle dell'analisi di correlazione.

Entrambe le analisi sopradescritte, sono state affiancante dall'ausilio di strumenti visivi e, dunque, da una consistente esplorazione del *dataset*. Di seguito si riporta una rappresentazione schematica della metodologia proposta nel lavoro di tesi per effettuare la selezione delle metriche *di performance.*

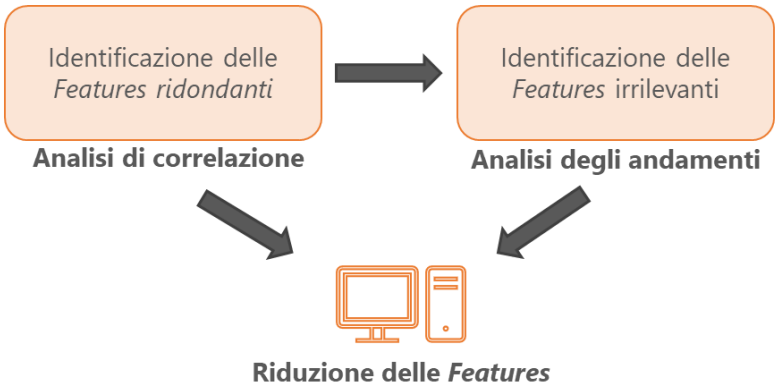

Figura 4.5: Processo di selezione delle metriche di *performance.* Identificazione delle *features* ridondanti, tramite l'analisi di correlazione e delle *features* irrilevanti, tramite l'individuazione delle metriche con andamenti costanti.

4.3.1 Analisi di correlazione

In generale, se due variabili risultano correlate, analizzare entrambe non aggiunge un contributo all'analisi del problema. Tuttavia, nel contesto del progetto in questione, la correlazione assume un'accezione specifica. L'analisi dei dati (metriche di *perfomance*) è volta al rilevamento delle anomalie e l'obiettivo è di individuarle analizzando volta per volta una singola metrica (analisi univariata). Nel contesto specifico, rilevare le correlazioni tra le metriche ed effettuare *feature reduction*, significa ritenere che ci sia una forte sovrapposizione tra le anomalie individuate in una metrica e quelle rilevate nella metrica correlata. Dunque, l'analisi di correlazione si è composta di tre fasi. In primo luogo sono state individuate le correlazioni lineari tra coppie di variabili, utilizzando il coefficiente di *Pearson*. Nella seconda fase, invece, sono state effettuate valutazioni al fine di individuare il coefficiente di correlazione adeguato per effettuare la riduzione delle metriche ridonanti nel contesto dell'obiettivo specifico di analisi quale l'*Anomaly Detection.* Infine è stata effettuata la riduzione del *dataset* di partenza, eliminando alcune metriche se correlate ad altre. Di seguito si riporta una rappresentazione schematica che evidenzia in modo chiaro le fasi dell'analisi di correlazione.

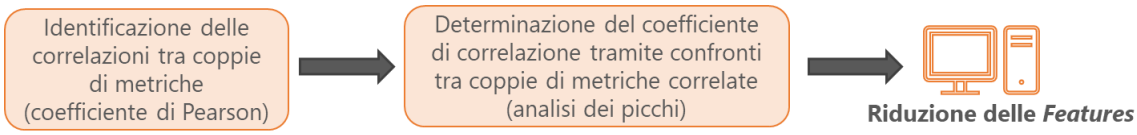

Figura 4.6 : Prima fase del processo di selezione delle metriche di *performance:* analisi di correlazione tra coppie di metriche. Identificazione delle correlazioni tramite il coefficiente di Pearson e confronti tra coppie di metriche correlate, per determinare il coefficiente di correlazione. Analisi dei picchi. Processo di identificazione delle *features* ridondanti.

La correlazione è una statistica che quantifica la forza della relazione che lega due variabili. Nel progetto di tesi è stata utilizzata la correlazione lineare tra coppie di serie temporali. La correlazione di *Pearson* utilizza il concetto di covarianza e di deviazione standard (formula 2.3). Per maggiori dettagli relativi alla correlazione, riferirsi alla sezione 2.2. La correlazione lineare è perfetta quando il valore è 1 o -1. Per effettuare l'analisi è stata determinata la matrice di correlazione tra le metriche di *performance*. Di seguito si riporta una tabella che mostra i coefficienti di correlazione tra alcune coppie di metriche. Data la numerosità delle metriche del *dataset*, si riportano esclusivamente le coppie caratterizzate da coefficienti di correlazione più rilevanti.

METRICA 1	METRICA 2	CORRELAZIONE
Process Limit %	Session Limit %	1
Physical Read Bytes Per Sec	Physical Read Total Bytes Per Sec	1
Physical Writes Per Sec	Physical Write Bytes Per Sec	1
Physical Reads Per Sec	Physical Read Total Bytes Per Sec	1
Host CPU Usage Per Sec	Host CPU Utilization (%)	1
Physical Reads Per Sec	Physical Read Bytes Per Sec	1
Average Active Sessions	Database Time Per Sec	1
Session Limit %	User Limit %	1
Session Limit %	Current Logons Count	1
Session Limit %	Session Count	1
Process Limit %	User Limit %	1
Process Limit %	Current Logons Count	1
Process Limit %	Session Count	1
I/O Megabytes per Second	Cell Physical IO Interconnect Bytes	1
User Transaction Per Sec	User Commits Per Sec	1
Physical Read Total IO Requests Per Sec	Physical Read IO Requests Per Sec	1
Session Count	User Limit %	1
Current Logons Count	User Limit %	1
Session Count	Current Logons Count	1
Consistent Read Gets Per Txn	Logical Reads Per Txn	0,99
Current OS Load	Host CPU Usage Per Sec	0,99
Logical Reads Per Sec	Consistent Read Gets Per Sec	0,99
Executions Per Txn	Open Cursors Per Txn	0,99
Current OS Load	Host CPU Utilization (%)	0,99
Physical Read Total Bytes Per Sec	Cell Physical IO Interconnect Bytes	0,98
Open Cursors Per Sec	Executions Per Sec	0,98
I/O Megabytes per Second	Physical Read Total Bytes Per Sec	0,98
Physical Writes Per Sec	Physical Writes Direct Per Sec	0,97
Physical Reads Per Sec	Cell Physical IO Interconnect Bytes	0,97
Executions Per User Call	Logical Reads Per User Call	0,97
Leaf Node Splits Per Txn	Branch Node Splits Per Txn	0,97
Physical Reads Per Sec	I/O Megabytes per Second	0,97
Consistent Read Changes Per Txn	CR Undo Records Applied Per Txn	0,97
Physical Read Bytes Per Sec	Cell Physical IO Interconnect Bytes	0,97
Current OS Load	CPU Usage Per Sec	0,97
CPU Usage Per Sec	Host CPU Usage Per Sec	0,97
CPU Usage Per Sec	Host CPU Utilization (%)	0,97
Physical Writes Direct Per Sec	Physical Write Bytes Per Sec	0,97
I/O Megabytes per Second	Physical Read Bytes Per Sec	0,97
User Commits Percentage	User Rollbacks Percentage	-1
Database Wait Time Ratio	Database CPU Time Ratio	-1
Rows Per Sort	Memory Sorts Ratio	-1

Tabella 4.1: Elenco di alcune coppie di metriche altamente correlate e i rispettivi coefficienti di correlazione lineare (indice di *Pearson)*

Inoltre, nella figura sottostante, si riporta l'*heatmap* al fine di visualizzare la matrice di correlazione tra coppie di variabili. Tale elemento visivo permette di cogliere le differenti entità di correlazione tra le metriche tramite l'utilizzo di diverse tonalità di colore. Data la numerosità del *dataset,* costituito da 155 metriche, non è stato possibile ottenere un elemento

che raffigurasse in modo leggibile le correlazioni tra tutte le coppie di *features,* dunque sono state selezionate solo alcune metriche tra quelle riportate nella tabella 4.1.

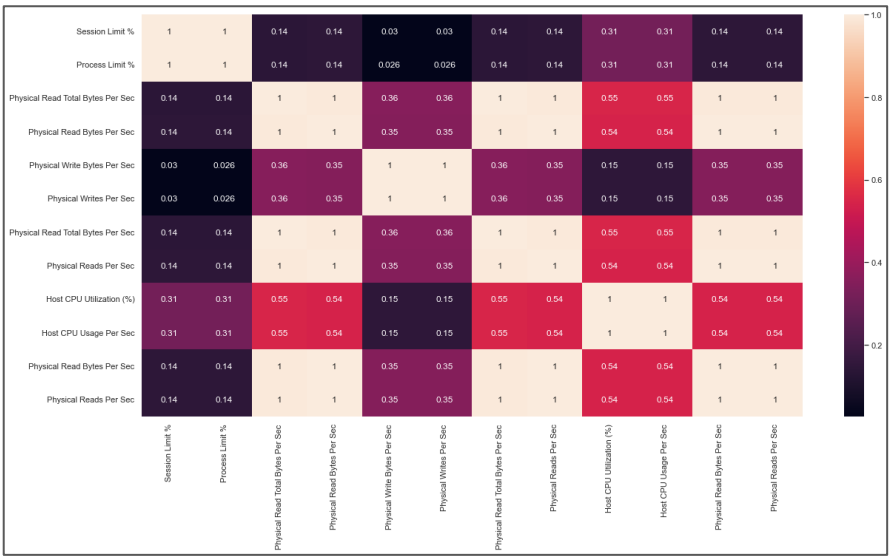

Figura 4.7: *heatmap*. Matrice di correlazione tra alcune delle metriche altamente correlate riportate in tabella 4.1.

Dopo aver determinato le coppie di metriche correlate, si è proceduto con la seconda fase dell'analisi di correlazione. Lo scopo di tale fase è stato quello di validare la possibilità di effettuare *feature reduction* di alcune metriche perché correlate ad altre, nel contesto specifico del rilevamento delle anomalie. A tal fine l'obiettivo è stato verificare l' esistenza o meno di una sovrapposizione, ed eventualmente l'entità della stessa, tra le anomalie rilevate tra coppie di metriche con alti coefficienti di correlazione. In ultima analisi è stato determinato il coefficiente di correlazione ottimale, da considerare per ridurre le *features.*

Per effettuare l'analisi si è scelto di adottare un approccio grafico, tramite l'esplorazione del *dataset*. Sono stati condotti *test* di unità tramite valutazioni a coppie di metriche. Per verificare la coincidenza delle anomalie è stato utilizzato il concetto di "picco". La presenza di un picco, in generale, può essere intesa come un elemento anomalo da identificare. Un evento anomalo, infatti, può essere definito come un evento "raro"(o atipico). Il concetto di rarità richiede di determinare quanto un evento è probabile; a tal fine si utilizzano le distribuzioni di probabilità. Un picco, infatti, è un valore che si discosta notevolmente dal "normale" *range* di oscillazione dei dati ed è, dunque, raro.

Per rilevare i picchi in maniera semi automatica, è stata utilizzata la funzione di *scipy "find_peaks*". Tramite tale funzione è possibile suggerire all'algoritmo di trovare i picchi sulla base delle proprietà della *time series* specifica. In particolare, la funzione richiede di settare il valore della prominenza. Quest'ultima è una misura che stabilisce quanto un elemento spicca rispetto agli elementi nel suo intorno. La prominenza di un picco può essere definita come il minor dislivello necessario per arrivare dalla vetta a qualsiasi terreno più alto. In altri termini, in topografia, la prominenza di un picco viene definita come la differenza tra la sua altitudine a la quota minima alla quale bisogna discendere dalla vetta del rilievo, per raggiungere un qualsiasi altro rilievo di altitudine maggiore. In generale, il concetto di picco permette di effettuare un'analisi qualitativa basata sul valore settato di prominenza, consentendo, inoltre, di cogliere esclusivamente le anomalie globali ma non contestuali.

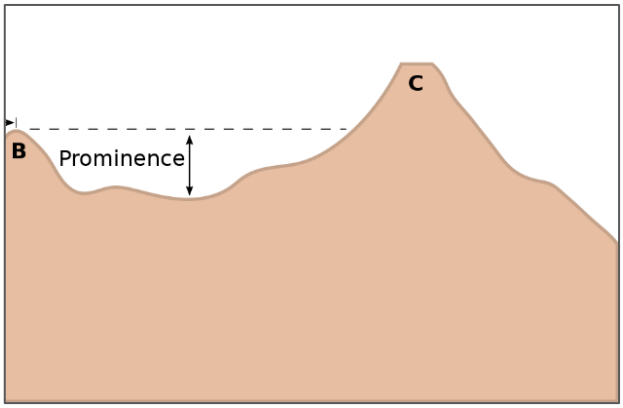

Figura 4.8: Prominenza di un picco in topografia: la differenza tra la sua altitudine a la quota minima alla quale bisogna discendere dalla vetta del rilievo, per raggiungere un qualsiasi altro rilievo di altitudine maggiore.

Il primo *test* che è stato effettuato è il confronto tra due metriche quali "*Logical Reads Per Sec*" e " *Current OS Load*" che risultano correlate con un coefficiente pari a 0,8, come si evince dalla tabella 4.1. Di seguito, in figura 4.9, si riporta una raffigurazione dell'andamento delle due metriche sovrapposte. In particolare "*Logical Reads Per Sec"* è rappresentata in blu, mentre " *Current OS Load*" in arancione.

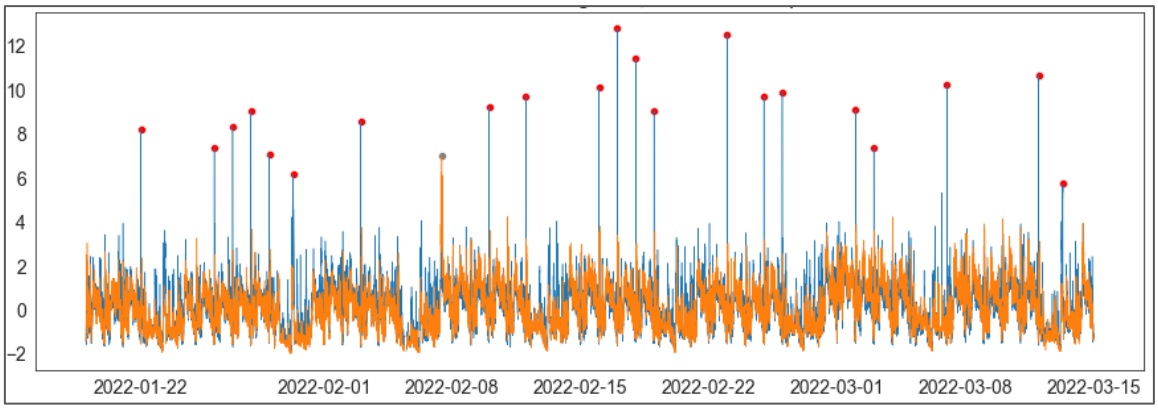

Figura 4.9: Sovrapposizione delle metriche "*Logical Reads Per Sec*", in blu, e " *Current OS Load*", in arancione, correlate con un coefficiente di correlazione di 0,8. In rosso si evidenziano i picchi della metrica "*Logical Reads Per Sec*", mentre in grigio quelli di " *Current OS Load*". La figura mostra la non sovrapposizione tra i picchi delle due metriche correlate.

Inoltre, nell'immagine sovrastante, figura 4.9, sono stati evidenziati i picchi, per ognuna delle due metriche, con colori differenti. In generale, nel lavoro di tesi, per il confronto a coppie delle metriche, è stato utilizzato il colore rosso per i picchi della metrica rappresentata in blu, mentre il colore grigio per quelli relativi alla metrica in arancione.

Per rilevare i picchi delle due metriche, quali "*Logical Reads Per Sec*" e " *Current OS Load*", è stato utilizzato un valore di prominenza pari a 7. Il valore di prominenza, nel lavoro di tesi, è stato settato in base alle caratteristiche specifiche della serie temporale oggetto di analisi. Dall'immagine sovrastante, risulta evidente che la metrica "*Logical Reads Per Sec*" presenta un range di oscillazione più ampio e conseguentemente la funzione "*find_peaks*" ha rilevato più picchi appartenenti a tale metrica, rappresentati in rosso, rispetto a quelli individuati per " *Current OS Load*", rappresentati in grigio. Ciò che risulta evidente è la non coincidenza dei "picchi". Da qui si evince che una correlazione che di per sé è "forte" (le due metriche presentano un coefficiente di correlazione pari a 0,8) non determina una sovrapposizione tra le possibili anomalie globali (picchi) delle due metriche confrontate. Dopo aver confrontato le altre coppie di metriche, correlate con lo steso coefficiente di correlazione, al fine di effettuare *feature reduction*, si è scartato il coefficiente 0,8, in quanto non ottimale per l'obiettivo specifico di analisi. A sostegno ulteriore di tale valutazione, si riporta, nell'immagine sottostante, l'andamento delle due metriche ottenuto considerando un numero ristretto di osservazioni, 1440 osservazioni, appartenenti a una finestra temporale di una giornata. In particolare, l'intervallo considerato comprende un periodo di tempo che parte dalle ore 18:00 del 24/02/2022 e termina il 25/02/2022 alla stessa ora.

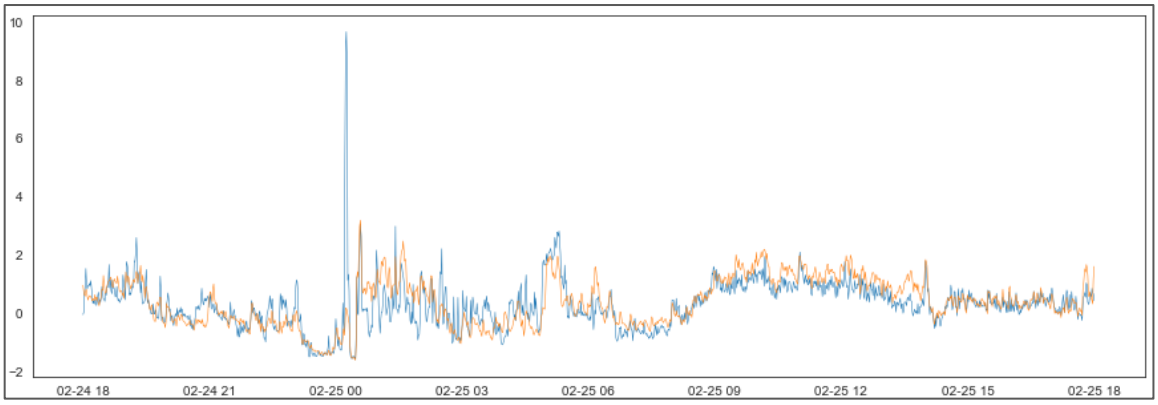

Figura 4.10: Sovrapposizione delle metriche "*Logical Reads Per Sec*", in blu, e " *Current OS Load*", in arancione, correlate con un coefficiente di correlazione di 0,8. Rappresentazione di una finestra temporale che parte dalle ore 18:00 del 24/02/2022 e termina il 25/02/2022 alla stessa ora. La figura mostra la non sovrapposizione tra i picchi delle due metriche, considerando un numero ristretto di osservazioni (1440).

L'immagine sovrastante mostra, in modo evidente, una forte sovrapposizione tra le due metriche, data la correlazione che le lega. Tuttavia, si evince chiaramente come la metrica "*Logical Reads Per Sec*" presenti un picco, in corrispondenza del quale non risulta per la metrica " *Current OS Load* ".

Come precedentemente affermato, il "picco" può essere interpretato come un valore anomalo. Tuttavia, ciò si adatta esclusivamente al concetto di anomalia "globale"; quest'ultima rappresenta un osservazione che si discosta notevolmente dal resto dei dati. Nell'ambito delle serie storiche, esiste anche un tipo di anomalia che si definisce come "contestuale". Tale osservazione anomala appartiene al normale range di oscillazione dei dati ma risulta essere un *outlier* se si considerano i fattori stagionali e di trend della *time series.* Occorre precisare che, nel contesto delle serie storiche, tenendo in considerazione fattori stagionali, un picco potrebbe, di per sé, non rappresentare un'anomalia. Tale picco potrebbe appartenere alla componente stagionale della serie storica e dunque, essere un valore "normale", che si ripete nel tempo, e non "anomalo".

Per tale motivazione, l'analisi dei picchi è stata accompagnata da scomposizioni delle serie storiche. Come metodo di scomposizione si è scelto di utilizzare l' STL (acronimo di *Seasonal Trend Decomposition using Loess*). Per maggiori dettagli sulle tecniche di scomposizione esistenti in letteratura riferirsi alla sezione 2.3.5. Nell'immagine sottostante si riporta la decomposizione della metrica "*Logical Reads Per Sec"*.

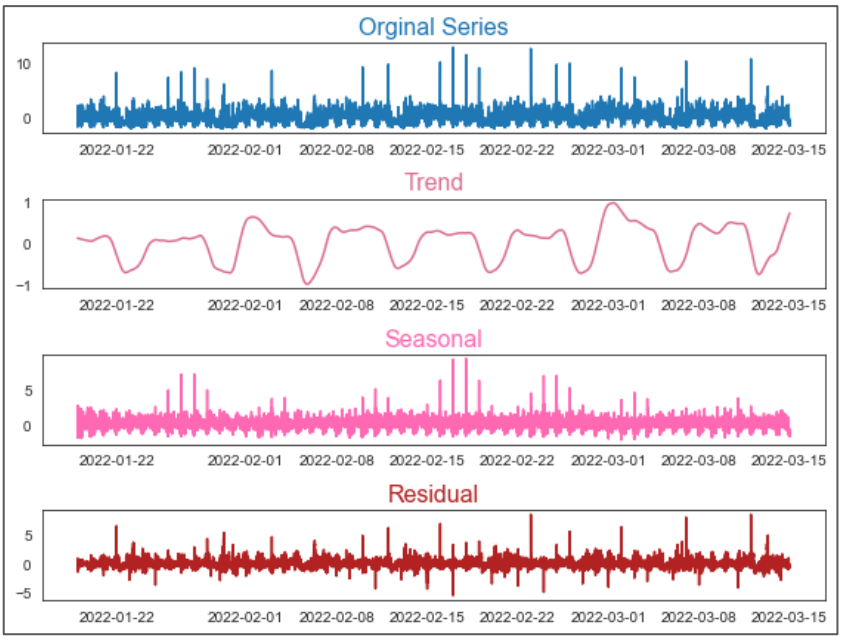

Figura 4.11: Scomposizione della metrica "*Logical Reads Per Sec", nelle componenti di Trend, Stagionale e Residuale. Utilizzo del metodo di decomposizione STL. La figura mostra che alcuni picchi appartenenti alla serie originale, rappresentano dei fattori stagionali.*

Alla base dell'*Anomaly Detection* effettuata attraverso metodi di scomposizione delle serie storiche, vi è il principio in base al quale le anomalie appartengono alla componente residuale della stessa. Il residuo, infatti, rappresenta il "rumore" e costituisce tutto ciò che non è previsto. Nell'immagine sottostante si riporta con il colore rosa la "componente prevista" data dalla somma della componente stagionale e di Trend. In blu è raffigurata la metrica "*Logical Reads Per Sec*". Tale raffigurazione mostra le aree in cui non vi è corrispondenza, il che significa che lì risiedono le anomalie.

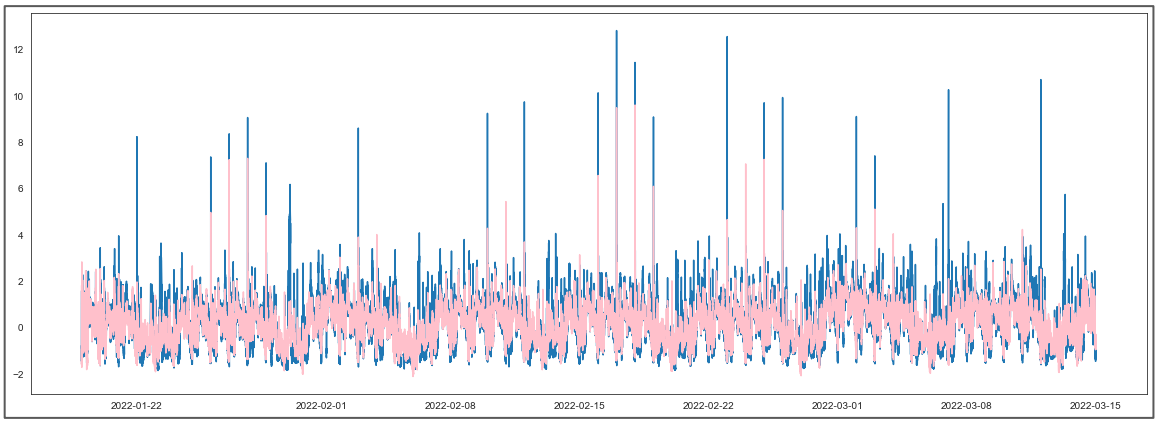

Figura 4.12: rappresentazione della metrica "*Logical Reads Per Sec",*in blu e della "componente prevista" in rosa. La "componente prevista" è data dalla somma del Trend e della stagionalità. La figura mostra, qualitativamente, le aree in cui risiedono i valori anomali.

La figura 4.11 conferma che alcuni picchi della serie non rappresentano anomalie in quanto appartenenti alla componente stagionale della stessa. Dunque, la criticità del "picco" come indice di anomalia globale, è stata dimostrata dall' analisi delle componenti della serie storica. Per rendere il metodo più robusto, le analisi di correlazione tra coppie di metriche, sono state affiancante da scomposizioni delle stesse.

Considerando nuovamente la metrica "*Logical Reads Per Sec*", si riporta nell'immagine sottostante una raffigurazione della componente residuale, in rosso, sovrapposta alla *time series* originale, in blu. La fascia in verde rappresenta l'area i cui non risiedono le anomalie; tale fascia è stata ottenuta settando un limite superiore e inferiore. Il metodo di *Anomaly detetcion*, esistente in letteratura, individua come valori anomali quelle osservazioni della serie originale in corrispondenza delle quali il valore residuale ricade al di fuori dei limiti superiori e inferiori precedentemente settati. Nello specifico, l'*upper bound* è stato impostato con un valore pari alla media (della componente residuale) sommato di 3 deviazioni standard (della componente residuale) mentre il *lower bound* è pari al valore medio sottratto di 3 volte la deviazione standard. Senza entrare nello specifico del rilevamento delle anomalie tramite l'algoritmo, inserendoci in una fase di esplorazione, la raffigurazione della fascia verde vuole mostrare come, nel caso specifico, la maggior parte dei picchi individuati per la metrica, figura 4.13, effettivamente, anche secondo l'algoritmo risulterebbero come anomalie. Tuttavia si evince come l'algoritmo rileverebbe come anomale anche delle osservazioni che non rappresentano dei picchi quali le anomalie contestuali. Infine, nonostante le criticità del "picco" , esso rappresenta un metodo intuitivo per il rilevamento

della presenza o meno delle anomalie globali, in una fase esplorativa, se accompagnato da valutazioni delle componenti delle serie analizzate.

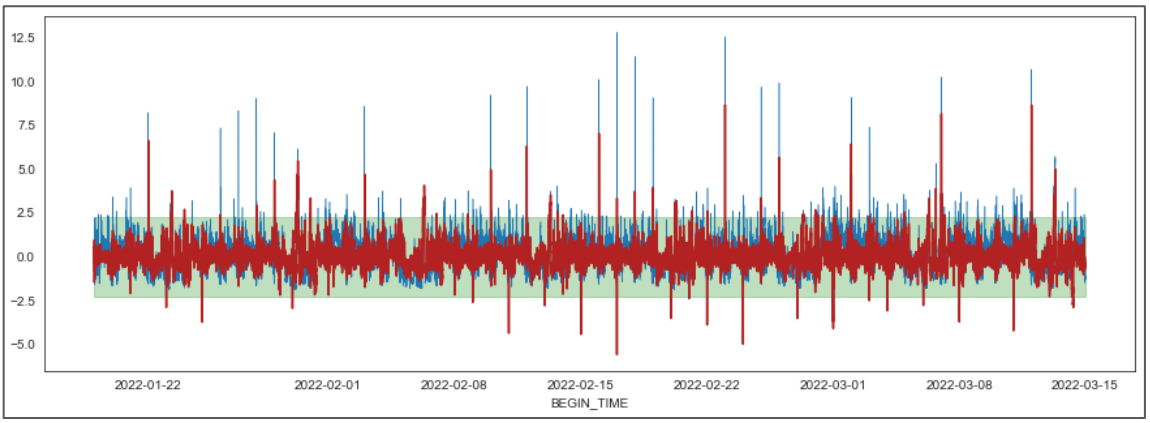

Figura 4.13: rappresentazione della metrica "*Logical Reads Per Sec",*in blu, della componente residuale, in rosso e della banda verde al di fuori della quale risiedono le anomalie. Metodo utilizzato per validare l'individuazione del picco, come indice di un valore anomalo.

Dopo aver scartato il coefficiente di correlazione pari a 0,8, ai fini della feature *selection*, sono stati effettuati ulteriori confronti aumentando di volta in volta il coefficiente considerato. In primo luogo, sono state valutate coppie di metriche correlate con un coefficiente pari a 0,9. A titolo di esempio si riportano i risultati ottenuti per due metriche quali "*Logical reads per sec*" e *"Total Table Scan Per User Call".* Nell'immagine sottostante, si riportano le due metriche a confronto raffigurate rispettivamente in blu e in arancione. Inoltre, l'immagine mostra i picchi individuati per entrambe le metriche settando un valore di prominenza pari a 7.

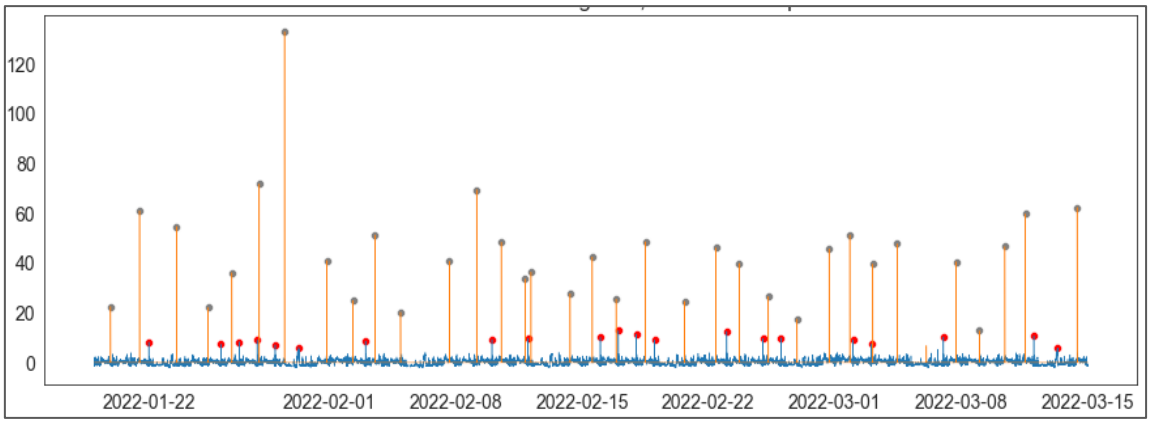

Figura 4.14 : Sovrapposizione delle metriche "*Logical reads per sec*", in blu, e *"Total Table Scan Per User Call*", in arancione, correlate con un coefficiente di correlazione di 0,9. In rosso si evidenziano i picchi della metrica "*Logical Reads Per Sec*", mentre in grigio quelli di *"Total Table Scan Per User Call"*. La figura mostra la non sovrapposizione tra i picchi delle due metriche correlate.

Dall'immagine riporta in figura 4.14 si evince come, nonostante 0,9 sia un coefficiente di correlazione molto alto, esso non determina, nel caso in esame, picchi coincidenti per le due metriche. Per maggiore chiarezza, si riporta, in figura 4.15, l'andamento delle due metriche sovrapposte, considerando una finestra temporale di una giornata.

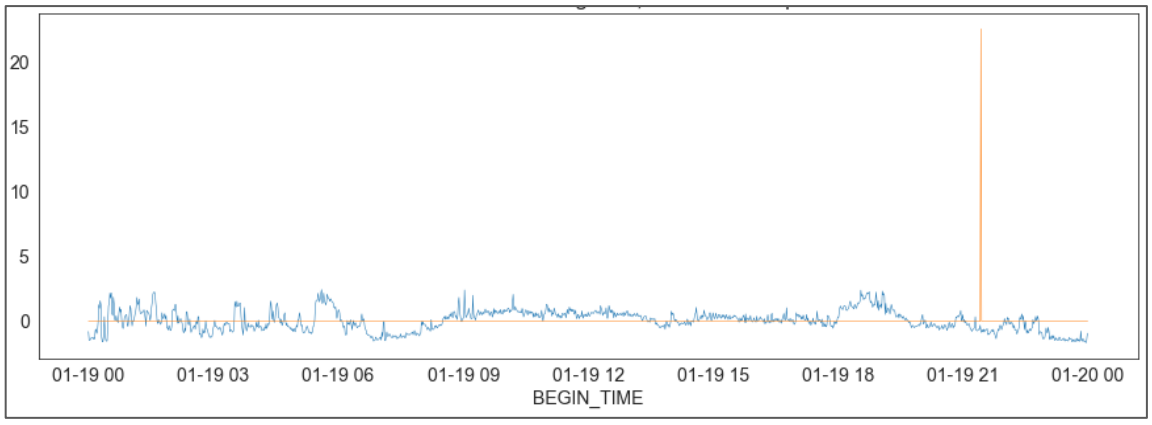

Figura 4. 15: Sovrapposizione delle metriche "*Logical reads per sec*", in blu, e *"Total Table Scan Per User Call*", in arancione, correlate con un coefficiente di correlazione di 0,9. La figura mostra la non sovrapposizione tra i picchi delle due metriche, considerando un numero ristretto di osservazioni (1440).

L'Immagine sovrastante, figura 4.15, mostra come la metrica "*Total Table Scan Per User Call"* presenti un picco, in un determinato istante temporale. Tuttavia, "*Logical reads per sec*", nonostante sia correlata con un coefficiente di correlazione pari a 0,9 con "*Total Table Scan Per User Call",* non presenta un picco in corrispondenza dello stesso istante di tempo. Il confronto tra le metriche sopracitate e le altre coppie di metriche correlate con lo stesso

coefficiente, ha condotto all'esclusione del parametro 0,9, al fine di effettuare *feature selection*.

Metriche correlate con un coefficiente pari a 1, rappresentano, invece, il caso di perfetta sovrapposizione. A titolo di esempio sono state considerando due metriche quali "*Process Lmit %"* in blu e "*Session Limit %* " in arancione. L'immagine sottostante mostra le due metriche sovrapposte. È possibile notare come, nel caso di correlazione lineare perfetta, la sovrapposizione è tale da non riuscire a distinguere i colori.

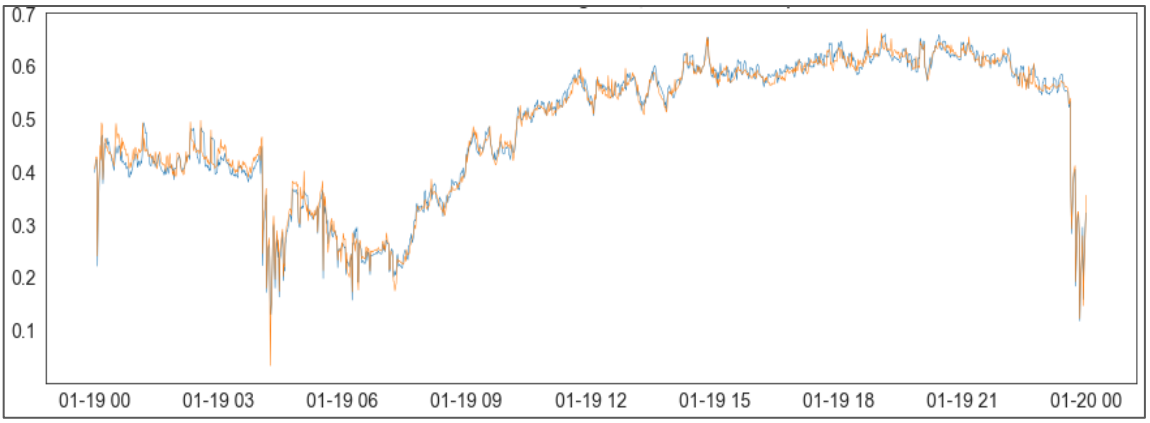

Figura 4.16: Sovrapposizione delle metriche "*Process Lmit %"*, in blu, e "*Session Limit %*", in arancione, correlate con un coefficiente di correlazione pari a 1. La figura mostra la perfetta sovrapposizione tra le due metriche.

Dalle analisi condotte, attraverso un approccio esplorativo, è emerso che il coefficiente di correlazione, nel contesto specifico dell'obiettivo di *business*, deve essere compreso tra il valore di 0,9 e 1. Per validare la scelta del parametro, sono stati effettuati confronti tra più coppie di metriche correlate con un coefficiente di correlazione compreso in tale *range*. Inoltre, sono state effettuate verifiche considerando molteplici coppie correlate con uno stesso coefficiente di correlazione, e sono stati considerati, per ognuna di esse molteplici intervalli temporali.

Infine, le evidenze grafiche hanno mostrato che un valore di correlazione pari a 0.97 può rappresentare, con una buona approssimazione, il parametro da utilizzare per effettuare *feature selection* nel contesto dell'*Anomaly Detection*. A titolo di esempio, si riporta il confronto effettuato tra due metriche cioè " *Physical Writes Per Sec*" e "*Physical Writes Direct Per Sec*". L'andamento delle due serie storiche è riportato in figura 4.17, dove le due metriche sono raffigurate rispettivamente in blu e in arancione.

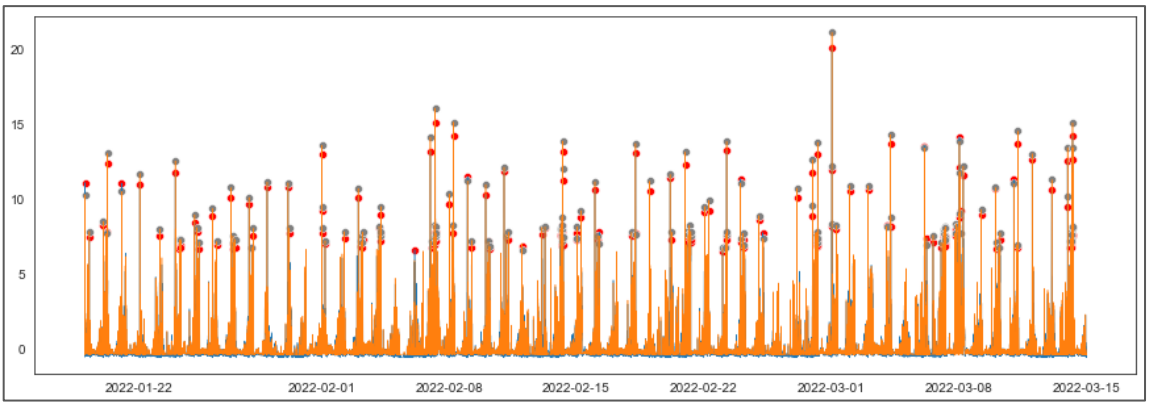

Figura 4.17: Sovrapposizione delle metriche " *Physical Writes Per Sec*", in blu, "*Physical Writes Direct Per Sec*", in arancione, correlate con un coefficiente di correlazione di 0,97. In rosso si evidenziano i picchi della metrica " *Physical Writes Per Sec"* mentre in grigio quelli di "*Physical Writes Direct Per Sec*" . La figura mostra la sovrapposizione tra i picchi delle due metriche correlate.

Dall'immagine si evince che i picchi delle due metriche coincidono per la maggior parte delle osservazioni; è possibile notare, infatti, come ci sia una forte sovrapposizione tra i pallini rossi, che rappresentano i picchi relativi alla metrica *" Physical Writes Per Sec"*, e grigi, che costituiscono quelli relativi alla metrica "*Physical Writes Direct Per Sec"*. Dato il coefficiente di correlazione considerato, cioè 0,97, quindi un coefficiente molto alto, risulta evidente come ci sia una forte sovrapposizione, quasi perfetta, tra le metriche tale da non riuscire a identificare la metriche raffigurata in blu. Risulta intuitivo che ottenere una perfetta sovrapposizione è possibile considerando esclusivamente correlazioni perfette. Tuttavia la scelta di una correlazione pari a 0,97 consente, con una buona approssimazione, di ottenere una sovrapposizione adeguata per l'obiettivo di analisi. La prima fase del processo di *feature selection*, basata sull'analisi di correlazione, figura 4.6, ha condotto all'esclusione di 26 metriche: riducendo il *dataset* da 155 a 129 metriche.

Come precedentemente esposto, la metodologia adottata presenta delle criticità; tra queste il concetto di picco consente di cogliere esclusivamente le anomalie globali, e in alcuni casi un picco potrebbe non rappresentare un'anomalia. Inoltre la sovrapposizione quasi perfetta potrebbe non essere un indice di perfetta coincidenza nel rilevamento delle anomalie. Inoltre il risultato, di coincidenza o meno tra le anomalie tra due metriche correlate, è fortemente dipendente anche dal tipo di algoritmo utilizzato. L'approccio adottato, basato su evidenze grafiche, rappresenta un punto di partenza per possibili approfondimenti futuri. Tale tematica sarà ripresa nella sezione 6.1 dedicata agli sviluppi futuri.

4.3.2 Analisi degli andamenti delle metriche di *performance*

Dopo aver effettuato l'analisi di correlazione tra le metriche, volta ad eliminare le *features* ridondanti, è stata condotta un'ulteriore fase di esplorazione del *dataset* con lo scopo di indentificare le *features* irrilevanti. Nello specifico contesto di *business*, quale l'*Anomaly Detection*, possono essere considerate come poco interessanti, ai fini dell'analisi, quelle metriche che non presentano picchi significativi. Di seguito si riporta una rappresentazione schematica della seconda fase del processo di selezione delle *features,* condotta nel progetto di tesi.

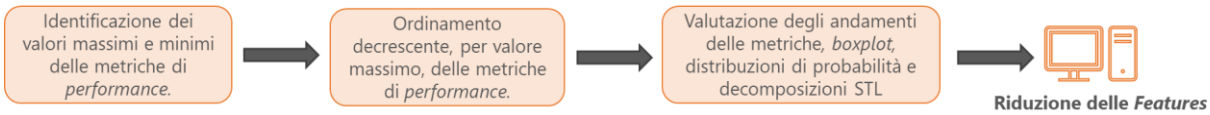

Figura 4. 18: Seconda fase del processo di selezione delle metriche di *performance:* analisi esplorativa dell'andamento delle metriche, dei *boxplot*, delle distribuzioni di probabilità e, infine, decomposizioni STL. Processo di identificazione delle *features* irrilevanti.

Nonostante la riduzione delle 26 *features*, avvenuta nella fase precedente, il *dataset* ridotto risultava caratterizzato da un numero di metriche pari a 129. Tale numerosità rendeva l'analisi onerosa e difficile da gestire. Tuttavia, se si considerano le 161 metriche, caratterizzanti il dataset iniziale, un numero di 129 metriche è già significativamente inferiore.

Data la numerosità del *dataset*, si è scelto di procedere con un ulteriore esplorazione dello stesso. Tali attività esplorative, oltre a consentire il rilevamento delle *features* irrilevanti, hanno permesso di ottenere una maggiore consapevolezza dei dati analizzati in una fase successiva. Il concetto di "*Data Exploration*", infatti, può essere inteso come un'analisi preliminare volta ad estrarre una conoscenza fruibili dai dati. Date le dimensioni del *dataset,* tale fase di analisi è risultata rilevante al fine di acquisire una maggiore consapevolezza dei dati con un conseguente risparmio di tempo nelle fasi successive di analisi.

Oltre all'analisi dei picchi, descritta nella sezione precedente, si è proceduto con la caratterizzazione delle distribuzioni degli attributi, attraverso un'analisi univariata. In particolare, per ognuna delle 129 metriche, è stato determinato l'istogramma e la stima della densità di Kernel. L'istogramma rappresenta un grafico che mostra la frequenza di ciascun valore assunto dalla metrica. La stima della densità di Kernel rappresenta una modalità di

stima della funzione di densità di probabilità della variabile considerata. Tale metodo utilizza una funzione di peso che assicura che l'area racchiusa dalla curva sia pari a 1. Per realizzare tali grafici è stata utilizzata una libreria di Python, quale "*seaborn"*, utilizzando la funzione *"histplot"*. Per determinare anche la stima della densità di *kernel,* è stato impostato il parametro "kde=True".

La scelta di esaminare le distribuzioni delle metriche è stata dettata da una motivazione ben precisa. Infatti, come il "picco" può rappresentare un indice di un'anomalia, anche la rappresentazione delle distribuzioni di probabilità consente di cogliere aspetti rilevanti a tal scopo. Attraverso la valutazione dei percentili, è possibile determinare gli *outliers;* quest'ultimi rappresentano quei punti che appartengono al primo e all'ultimo percentile. Come precedentemente ribadito, un evento anomalo può essere definito come un evento "raro". Il concetto di rarità richiede di determinare quanto un evento è probabile. Il percentile, infatti, rappresenta una misura usata in statistica per indicare il minimo valore al di sotto del quale ricade una data percentuale degli altri elementi sotto osservazione.

In generale, data una distribuzione, risulta utile determinare il valore massimo, il valore minimo, la media e la deviazione standard. A tal fine tramite "*pandas-profiling*" è stato possibile creare un report HTML interattivo che determina, in primo luogo, per ogni colonna, la presenza o meno di "*missing values*". Tuttavia, per nessuna metrica sono stati rilevati valori mancanti; tale risultato era atteso in quanto la verifica era già stata effettuata in una fase precedente. Inoltre, per ogni metrica, il report mostra alcune statistiche quali il valore massimo, il valore minimo, la mediana, il primo e il terzo percentile e, infine, il cosiddetto "*Interquartile Range"* che rappresenta la differenza tra il terzo e il primo percentile. Inoltre il *report* mostra anche le statistiche descrittive, come per esempio la media, la moda, la deviazione standard. In aggiunta, è possibile visualizzare i valori più frequenti, per ogni metrica.

Tuttavia, data la numerosità delle metriche e delle osservazioni registrate per ciascuna di esse (129 metriche con 80000 record per ognuna), non è stato possibile determinare un *report* che contenesse tutte le metriche del *dataset*. Dunque, per generarlo, sono state considerate 40 metriche alla volta. Nell'immagine sottostante, si riporta una raffigurazione di una sezione del *report* che mostra le statistiche descrittive ottenute per due metriche quali "*Cursor Cache Hit Ratio*" e "*User Rollback UndoRec Applied Per Sec".*

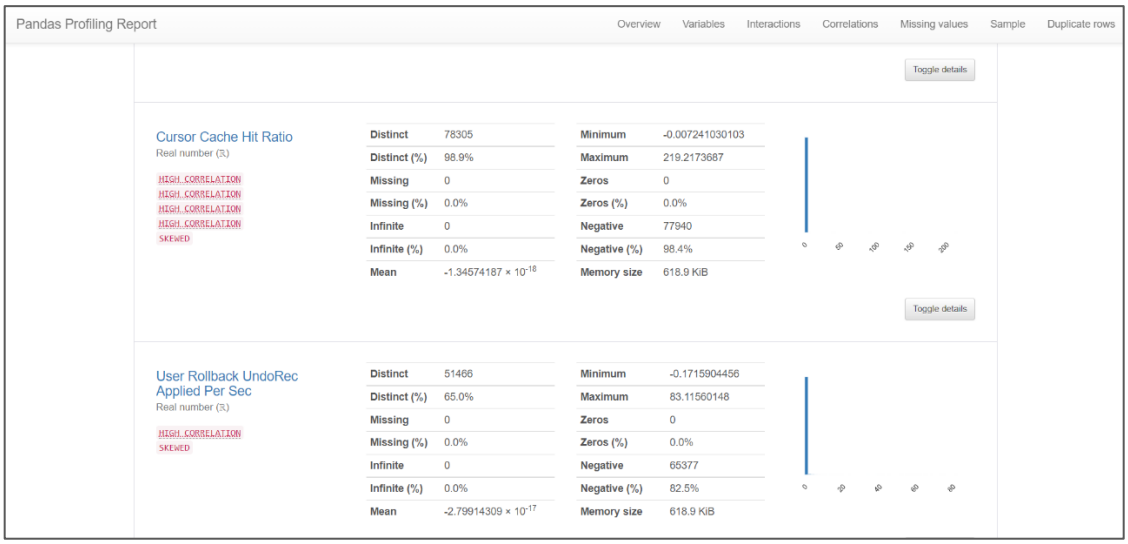

Figura 4.19: Rappresentazione delle statistiche descrittive ottenute per due metriche "*Cursor Cache Hit Ratio*" e "*User Rollback UndoRec Applied Per Sec"* estratta dal report generato attraverso *Pandas-profiling.*

Infine, per visualizzare gli *outliers,* sono stati valutati i *boxplot* per ciascuna metrica. I *boxplot* costituiscono una rappresentazione grafica che consente di descrivere la distribuzione di un campione tramite indici di dispersione e di posizione. Il *boxplot* è rappresentato tramite un rettangolo suddiviso in due parti. Il rettangolo è delimitato dal primo e dal terzo quartile ed è diviso, al suo interno, dalla mediana. I segmenti, o "baffi", sono delimitati dall' *upper bound (*uguale alla somma del terzo quartile e il *range* interquartile moltiplicato per 1,5) e il *lower bound* (uguale al primo quartile sottratto del range interquartile moltiplicato per 1,5).

Grazie all'utilizzo degli strumenti sopra descritti, è stato possibile effettuare un'analisi esplorativa. Tuttavia, per facilitare l'analisi, si è scelto di creare un pannello, utilizzando la libreria "*Panel"* di Python. Tale libreria consente di creare dei *widget* dando la possibilità di modificare i grafici da visualizzare senza la necessità di dover determinare 129 grafici statici differenti, uno per ognuna delle metriche del *dataset.* Di seguito si riporta una raffigurazione del pannello creato.

Figura 4.20: Rappresentazione del pannello creato, tramite l'utilizzo della libreria *Panel* di *Python*. Il pannello consente di selezionare le metriche e di visualizzare per ognuna di esse, l'andamento della metrica con i picchi, la stima della densità di Kernel e il *boxplot.*

Per ogni metrica, il pannello consente di visualizzare i grafici dell'andamento della stessa, con la possibilità di settare il valore di prominenza e di individuare, conseguentemente, i picchi. Inoltre, il pannello mostra l'istogramma con la stima della densità di Kernel e il *boxplot*, che evidenzia gli *outliers*. Per creare il pannello è stato definito il selettore della colonna da analizzare e della prominenza. Tale selettore consente di selezionare la metrica che si vuole analizzare. Successivamente è stata creata una griglia per inserire i grafici e sono stati definiti i grafici da inserire. Inoltre per ogni metrica, la *dashboard* mostra le statistiche descrittive.

Dunque il pannello e il *report* sono stati realizzati al fine di agevolare la fase esplorativa. Tuttavia, tali strumenti sono stati predisposti non solo per essere utilizzati in una fase preliminare di analisi ma anche per validare i risultati ottenuti in seguito all'applicazione degli algoritmi di *Anomaly Detection*.

Per identificare le *features* irrilevanti, perché caratterizzate da andamenti costanti e *range* di oscillazioni poco variabili, si è proceduto raccogliendo, in primo luogo, i valori massimi e minimi di ogni metrica. Per ognuna di esse è stato selezionato il massimo, in valore assoluto, tra il minimo e il massimo. Successivamente le metriche sono state ordinate in ordine decrescente, sulla base del valore massimo. Le analisi di valutazione sono partite dalle

metriche con il valore massimo minore al fine di scartare quelle con i picchi meno significativi.

Tale approccio, basato sul confronto tra i valori massimi, in valore assoluto, delle metriche, è stato possibile in quanto le stesse erano state precedentemente standardizzate. Dunque, esse risultavano caratterizzate da un valore medio pari a 0. Questo tipo di trasformazione ha permesso di effettuare il confronto considerando esclusivamente il valore massimo o minimo, senza dover calcolare la differenza dal valore medio. Inoltre, per effettuare l'ordinamento, si è scelto di non considerare il *range* di oscillazione; infatti la maggior parte delle metriche presentava oscillazioni significative in una sola direzione. A titolo di esempio si può considerare la metrica "*DDL statements parallelized Per Sec*" che è caratterizzata da un valore massimo pari a 52 e da un valore minimo uguale a -0,05. In figura 4.21 si riporta una rappresentazione delle metriche e dei rispettivi valori massimi, in modulo.

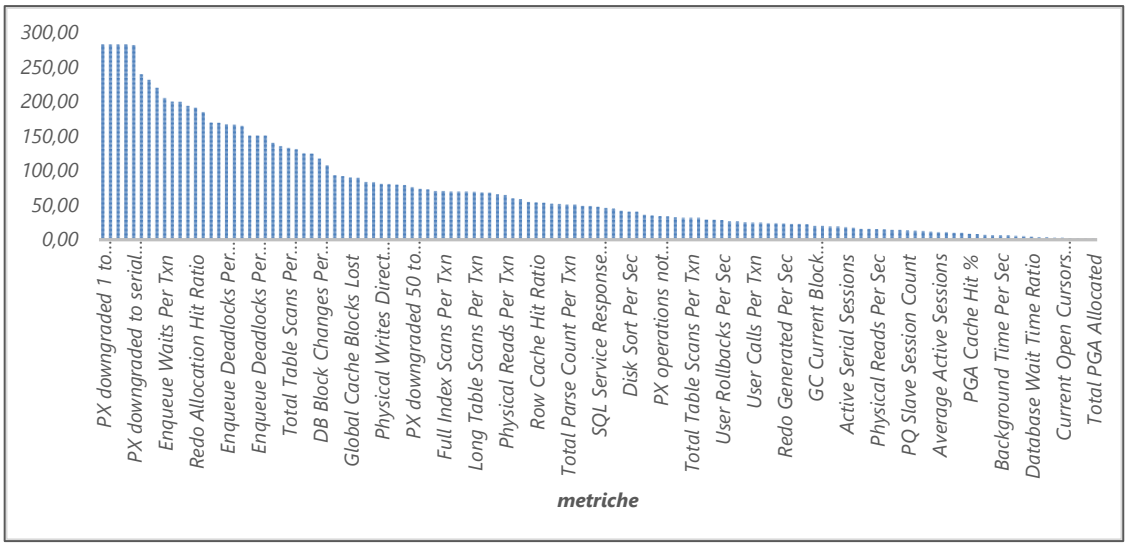

Figura 4.21: Rappresentazione dei valori massimi delle metriche, in valore assoluto. Metriche ordinate in ordine decrescente di valore massimo (in valore assoluto).

Dall'ordinamento effettuato si evince che "*PX downgraded 1 to 25% Per Sec"* è la metrica che assume il valore massimo maggiore, tra le 129 caratterizzanti il *dataset*, con un valore pari a 281,42. Tale metrica, inoltre, presenta un minimo pari a 0. La *feature* considerata rappresenta un caso estremo; è caratterizzata, infatti, da un solo picco pari a 281,42 mentre i restanti valori risultano costanti e pari a 0. La maggior parte delle metriche del *dataset* presentano, invece, molteplici picchi, più o meno significativi, o in positivo o in negativo, mentre i restanti valori che le caratterizzano oscillano intorno allo zero. Dunque, il *dataset*

presentava anche metriche con oscillazioni in negativo, come il caso di "*Redo Allocation Hit Ratio*".

La metrica con il valore massimo minore è, invece, "*Total PGA Allocated".* Quest'ultima è stata la prima metrica ad essere valutata, al fine di coglierne la rilevanza. La metrica in questione è caratterizzata da un valore massimo pari a 1,74 e un valore minimo pari a -1,71. Ciò che si evince è la presenza di un andamento notevolmente differente dalle altre del *dataset*; la maggior parte delle *features,* infatti, risulta caratterizzata da oscillazioni rilevanti, o in positivo o in negativo. "*Total PGA Allocated*", invece, è caratterizzata da un range di oscillazione piccolo e costante intorno al valore medio pari a 0. Nell'immagine 4.22 si riporta l'andamento della metrica "*Total PGA Allocated".*

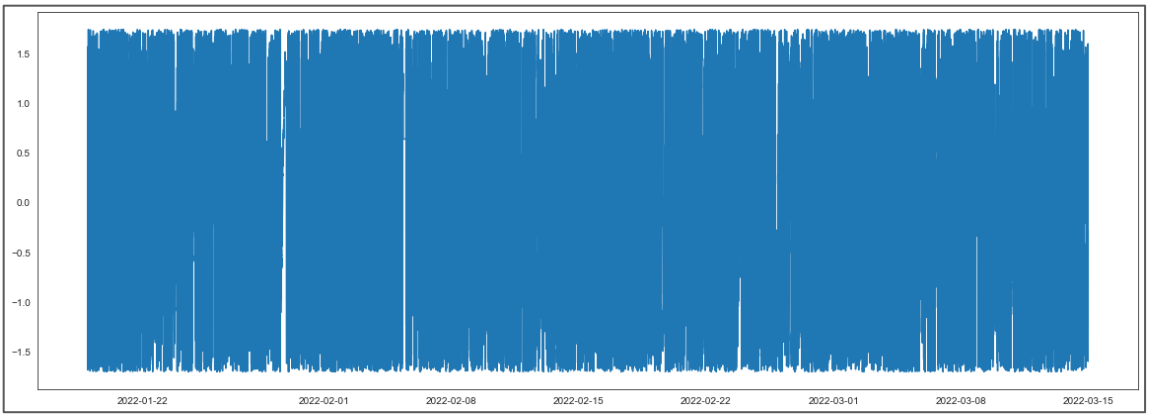

Figura 4.22: *Total PGA Allocated,* metrica con il valore massimo minore. Andamento della metrica nel tempo. La figura mostra un range di oscillazione piccolo e costante.

Dunque, tale *feature* è stata eliminata. Infatti dall'immagine si evince come non è possibile individuare, visivamente, dei picchi. Per giustificare la valutazione, inoltre è stato esaminato il *boxplot* che effettivamente mostra l'assenza di *outliers.* Il *boxplot* è riportato in figura 4.23.

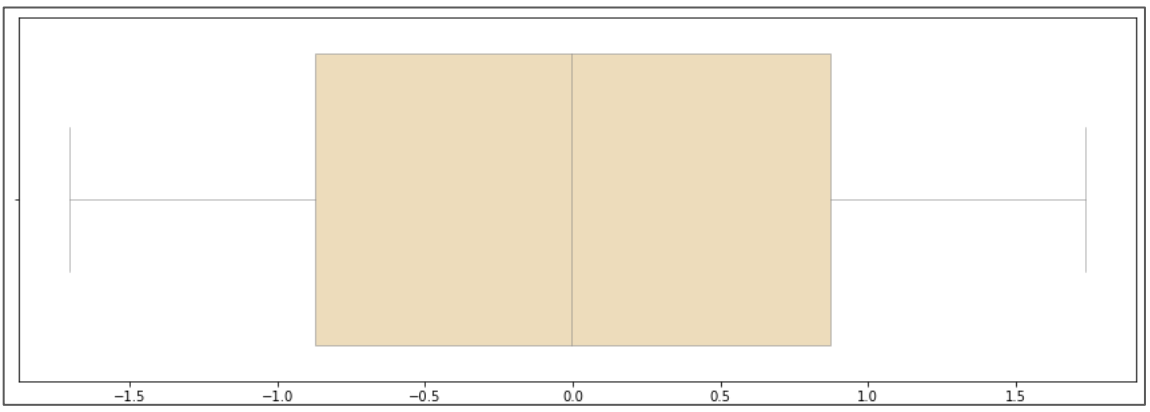

Figura 4.23: Boxplot, *Total PGA Allocated,* metrica con il valore massimo minore. La figura mostra l'assenza di *outliers.*

Per la metrica in questione, inoltre, è stata effettuata la decomposizione STL. La figura 4.25 mostra la componente residuale, raffigurata in rosso, e la fascia verde delimitata dai limiti superiori e inferiori. Con evidenza grafica si evince, come non ci siano punti che ricadono al di fuori dei limiti settati. Per determinare la fascia verde è stato considerato come *upper bound* un valore pari alla media (della componente residuale) sommato di 3 deviazioni standard (della componente residuale) mentre il *lower bound* è pari al valore medio sottratto di 3 volte la deviazione standard. Senza entrare nello specifico del rilevamento delle anomalie tramite l'algoritmo, inserendoci in una fase di esplorazione, la raffigurazione della fascia verde vuole mostrare come la metrica sia effettivamente irrilevante al fine dell'analisi.

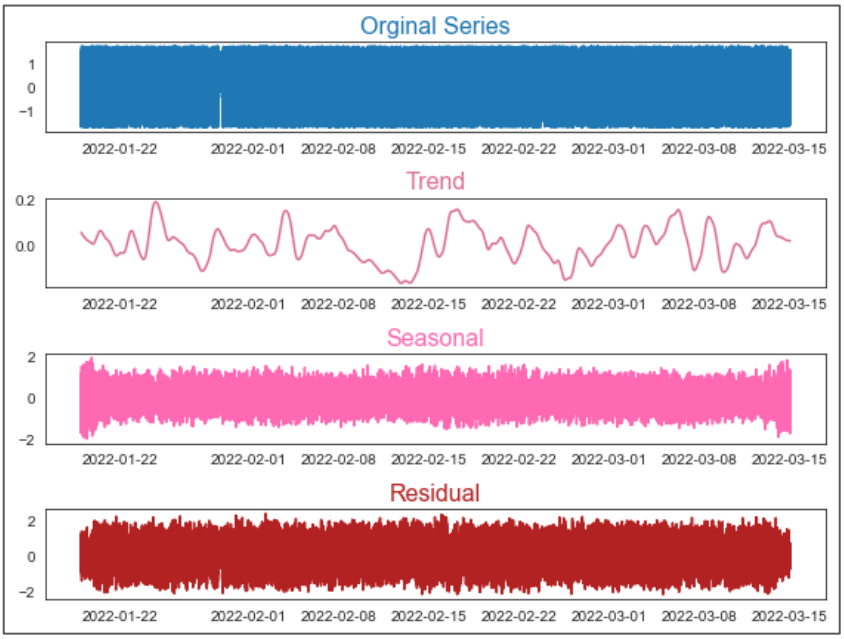

Figura 4.24: Scomposizione della metrica *Total PGA Allocated,* tramite metodo STL.

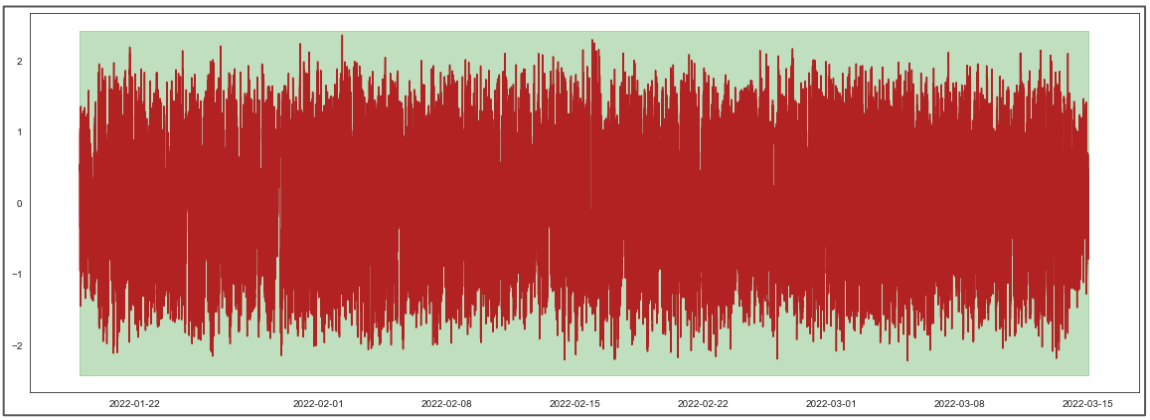

Figura 4.25: *Total PGA Allocated,* metrica con il valore massimo minore. Componente residuale e banda verde al di fuori della quale risiedono le anomalie. La figura mostra l'assenza di valori anomali.

Effettuando analoghe considerazioni, sono state eliminate altre metriche. Le metriche scartate, infatti, presentano, al pari di "*Total PGA Allocated*", *range* di oscillazioni costanti. Nella tabella 4.2 si riportano le metriche eliminate e i rispettivi valori massimi e minimi. La valutazione dei *range*, di per sé, permette di cogliere come l'oscillazione sia più o meno costante intorno al valore medio. Tuttavia, sono state effettuate valutazioni grafiche per confermarne l'ipotesi. Dunque le metriche che sono state scartate sono 7, riducendo il *dataset* finale da 129 metriche a 122.

METRICA	MAX	MIN
Process Limit %	4,17	-5,54
Total Parse Count Per Sec	3,58	-1,48
Current Open Cursors Count	2,68	-3,38
User Calls Ratio	2,77	-2,72
I/O Requests per Second	2,14	-1,04
Temp Space Used	1,90	-1,51
Total PGA Allocated	1,74	-1,71

Tabella 4.2: Elenco delle metriche eliminate, con i rispettivi valori massimi e minimi. Metriche eliminate a seguito dell'analisi degli andamenti.

Altre metriche caratterizzate da valori massimi non elevati, se rapportati ai valori assunti dalle altre, non sono state scartate in quanto caratterizzate da evidenti picchi. È il caso della metrica "*Total Table Scans Per Sec*" che presenta un valore massimo pari a 10,76. L'Andamento della metrica "*Total Table Scans Per Sec*" è riportato in figura 4.26.

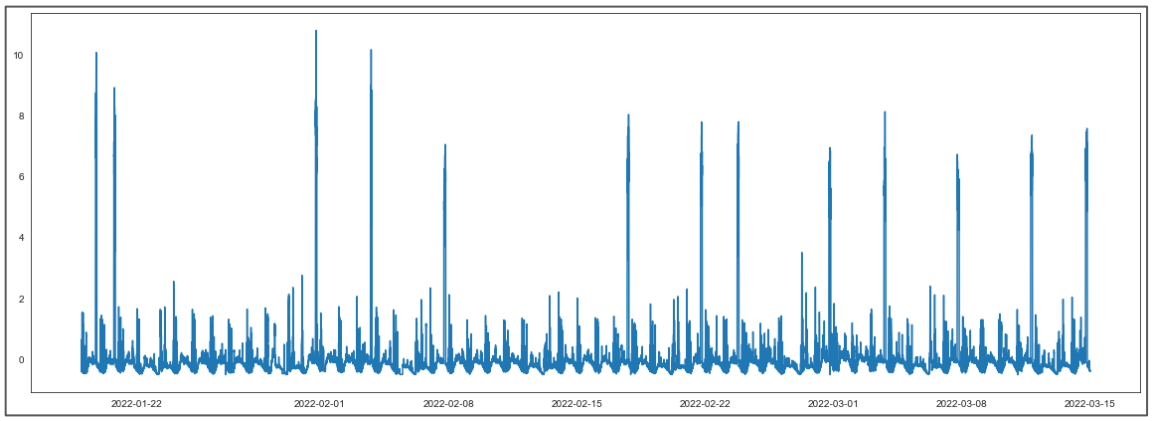

Figura 4.26: *Total Table Scans Per Sec,* metrica con un valore massimo pari a 10,76. La figura mostra la presenza di picchi e, dunque, la possibile presenza di anomalie. La metrica non è stata eliminata dal *dataset*.

5 Modelli, metodologia e risultati

Dopo aver effettuato l'esplorazione del *dataset* (*Exploratory Data Analysis*) e l'analisi di correlazione tra le metriche di *performance*, che ha condotto all'eliminazione di alcune *features*, si è proceduto con l'individuazione delle anomalie. Nel progetto di tesi si è scelto di utilizzare due differenti approcci; il primo si basa sull'applicazione di cinque modelli di serie temporali, il secondo, invece, utilizza un algoritmo di *machine learning* non supervisionato che sfrutta gli alberi decisionali, quale l'*Isolation Forest*. Di seguito si riporta una raffigurazione della metodologia perseguita nel suddetto progetto, che verrà illustrata nel capitolo in questione. Lo schema rappresentato in figura 5.1 offre una visione complessiva e sequenziale delle principali fasi di analisi.

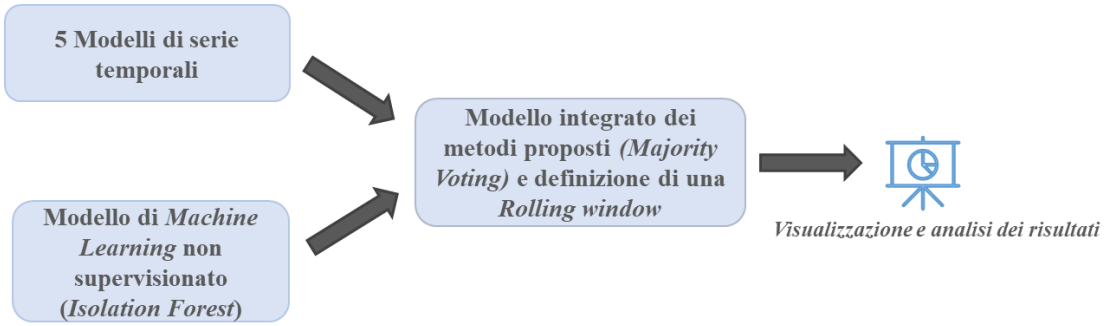

Figura 5.1: Processo di individuazione delle anomalie. Implementazione di 5 modelli di serie temporali e un modello di *Machine Learning* non supervisionato. Integrazione dei metodi tramite la definizione di un modello basato sul principio di "*Majority Voting*". Applicazione di *Rolling windows* per effettuare il rilevamento dei valori anomali.

5.1 Rilevazione delle anomalie nelle metriche di *performance*

In questa sezione del capitolo si riportano i modelli di *Anomaly Detection* che sono stati applicati alle metriche di *performance.* Per ognuno di essi si dettagliano le funzioni che sono state utilizzate e i parametri settati. Successivamente si illustra il metodo di analisi specifico che è stato sviluppato per il progetto di tesi.

5.1.1 Modelli di *Anomaly Detection*

Inerentemente all'implementazione dei modelli di serie temporali per effettuare *Anomaly Detection*, sono stati applicati alcuni algoritmi proposti in letteratura. In generale, si è scelto di implementare gli algoritmi mantenendo i valori dei parametri di *default*; quest'ultimi rappresentano i valori consigliati dagli ideatori dei modelli. Per quanto riguarda l*'Isolation Forest* si è scelto di implementare il modello aggiungendo alcune *features*, applicando un processo di *feature engineering*. Le motivazioni di tale scelta sono esposte nell'ultima sezione del paragrafo. Per effettuare le analisi è stato utilizzato il software R.

Il primo modello considerato nel progetto in questione si basa sui modelli ARIMA ed è la procedura proposta da *Chung Chen* e *Lon-Mu Liu* nella pubblicazione "*Joint Estimation of Model Paramaters and Outlier Effects in Time Series*" (sezione 2.4.2). Quest'ultimi hanno ideato un metodo di *Anomaly detection* iterativo che consiste in una procedura che consente di ottenere in modo congiunto la stima dei parametri del modello e degli effetti degli *outliers*. Inoltre, permette di individuare quattro tipologie di *outliers: innovational outlier* (IO), *additive outlier* (AO), *level shift* (LS) e infine, *temporary change* (TC).

A tal fine è stato utilizzato il pacchetto *'tsoutliers'*. Quest'ultimo fornisce una funzione, *'tso'*, che consente di implementare i tre *stages* della procedura proposta da *Chen* e *Liu*, in modo automatico. Di fatti, come riportato nella sezione 2.4.2, il metodo si compone di tre differenti fasi. Tale modello, nel progetto di tesi verrà identificato come "*tso*", dal nome della funzione che lo implementa. L'unico argomento della funzione che è stato specificato è la serie temporale, oggetto di analisi. Di *default*, il parametro '*cval'*, che rappresenta il valore critico per determinare la significatività di ogni tipo di *outlier* , dipende dalle dimensioni del campione. Il parametro '*cval'* si riferisce al parametro *C* dell'algoritmo 2.1. Se $n \leq 50$ allora 'cval' è impostato con un valore pari a 3, se $n \geq 450$ allora il parametro è uguale a 4, altrimenti il valore è settato con un valore di $3 + 0{,}0025 * (n - 50)$. Dunque, poiché ogni serie temporale del *dataset* è caratterizzata da un numero di osservazioni pari circa a 80000, di *default* il parametro assume un valore pari a 4. Inoltre, viene settato il numero massimo di "*inner loop*" (algoritmo 2.1) con un valore paria a 4.

Come precedentemente esposto, il modello in questione si basa sui modelli ARIMA. Nello specifico, la funzione "*tso*", applica come metodo per determinare i parametri del modello ARIMA la funzione "*auto.arima*". Quest'ultima determina la combinazione ottimale dei

parametri (p,d,q), calcolando un criterio di informazione per ognuna di esse. Nello specifico, come criterio di informazione utilizza il cosiddetto "BIC"(acronimo di *Bayesian Information Criterion)*. Per maggiori delucidazioni sui parametri dei modelli ARIMA e sui criteri di informazione, riferirsi alla sezione 2.3.3.

Il secondo modello, proposto nel progetto di tesi, è stato implementato utilizzando la funzione *'tsoutliers()'* del pacchetto '*Forecast'* disponibile nel software R. Tale funzione consente di indentificare le anomalie nelle serie temporali, nel contesto di un'analisi univariata. Dunque, l'unico argomento inserito nella funzione è stata la serie temporale specifica, oggetto di analisi. La funzione esegue una procedura che si compone di più fasi. In primo luogo la serie temporale viene scomposta nelle tre componenti quali il trend, la stagionalità e il residuo. Dopo aver rimosso le componenti di Trend e Stagionali, le anomalie sono rilevate nella componente residuale.

Tuttavia, la componente stagionale può contenere differenti *patterns;* tale condizione si verifica nel caso in cui la serie presenta osservazioni ad alta frequenza (come, per esempio, dati giornalieri, orari o al minuto). Dunque, la funzione *'tsoutliers()'* effettua automaticamente una decomposizione che sfrutta il metodo MSTL(acronimo di *Multiple Seasonal Trend Decomposition*). Quest'ultimo rappresenta un'estensione del tradizionale *Seasonal-Trend Decomposition using Loess* (STL). Tale scomposizione consente di cogliere le distinte stagionalità caratterizzanti la serie temporale. Tale metodo è stato ideato da Kasun Bandara, Rob J Hyndman, Christoph Bergmeir, autori della pubblicazione *"MSTL: a Seasonal – Trend Decomposition for time-series with multiple seasonal patterns".* Per maggiori dettagli, vedere la sezione 2.4.2.

Dopo avere effettuato la decomposizione, viene automaticamente calcolato un valore che indica la "potenza" della stagionalità (F_s), all'interno della serie. Tale valore soglia è espresso dalla seguente espressione $F_s = 1 - \frac{VAR(y_t - \hat{T}_t - \hat{S}_t)}{VAR(y_t - \hat{T}_t)}$. Se $F_s > 0{,}6$ e dunque $\frac{VAR(y_t - \hat{T}_t - \hat{S}_t)}{VAR(y_t - \hat{T}_t)} < 0{,}4$, significa che la serie temporale, oggetto di analisi, presenta una componente stagionale forte. In tal caso la funzione *"tsoutliers"* effettua automaticamente un aggiustamento della serie andando a determinare $y_t^* = y_t - \hat{S}_t$. Se invece la componente stagionale è debole o non esistente (nel caso di dati a bassa frequenza), cioè se $F_s \leq 0{,}6$, significa che la stagionalità è stata sovrastimata dalla decomposizione MSTL. Tale condizione potrebbe condurre a un mancato rilevamento di alcuni punti che in realtà sono

anomali ma non vengono rilevati come tali perché inclusi nella componente stagionale sovrastimata. Dunque in tal caso la funzione *"tsoutliers"* non effettua aggiustamenti, cioè non viene sottratta la componente stagionale dalla serie temporale e $y_t^* = y_t$.

Successivamente viene effettuata un' ulteriore stima della componente di *Trend* applicando la funzione "*supsum()"alla serie* y_t^*. Infine, gli *outliers* vengono rilevati nella componente residuale data da $\hat{R}_t = y_t^* - \hat{T}_t$. Per rilevare gli *outliers* viene applicato il metodo IQR (acronimo di *Interquartile Range*). Il valore assunto da IQR è la differenza tra il 75th percentile (Q_3) e il 25th percentile (Q_1). Si definisce un *lower bound* pari a $Q_1 - (1{,}5 * IQR)$ e un *upper bound* uguale a $Q_3 + (1{,}5 * IQR)$. Tale modello, nel progetto di tesi verrà identificato come "iqr", dal metodo di rilevamento degli *outliers* applicato.

Il terzo modello relativo alle serie temporali, oggetto di analisi del suddetto lavori di tesi , si basa su un metodo di decomposizione della serie ideato da *Owen Wallis, Jordan Hochenbaum* e *Arun Kejarival,* autori della pubblicazione *"A Novel Technique for Long-Term Anomaly Detection in the Cloud".* Per maggiori dettagli, vedere la sezione 2.4.2.

Tale tecnica statistica è stata sviluppata al fine di rilevare automaticamente le anomalie *long-term* in dati *Cloud*; dati caratterizzati da una grande velocità di generazione, in *real-time*, e da grandi volumi. Il metodo è stato sviluppato considerando la metrica *Tweets Per Sec* (TPS). Per implementare l'algoritmo è stato utilizzato il pacchetto "*Anomalize"* ed è stata utilizzata, in primo luogo, la funzione *"time_decompose*()". Come parametro è stato inserito, oltre alla serie temporale, il metodo di decomposizione scelto cioè "method = 'Twitter'".

Il metodo di decomposizione *Twitter* rimuove, in primo luogo, la componente stagionale e successivamente effettua un'approssimazione per segmenti della componente di *Trend*, considerando, nel calcolo, la mediana. La letteratura mostra che tale metodo è più efficiente, rispetto al metodo STL, nel caso in cui i dati presentano componenti stagionali preponderanti rispetto alle componenti di *Trend*. Difatti, la fase esplorativa del dataset ha mostrato che le serie temporali delle metriche di performance, oggetto di analisi, presentano *patterns* stagionali, mentre la componente di *Trend* non risulta particolarmente significativa. Da qui la scelta del metodo *Twitter.*

La seconda fase del metodo consiste nel rilevamento delle anomalie. È stata, a tal fine, utilizzata la funzione *"Anomalize()",* ed è stata inserita, come argomento della funzione*,* la componente residuale. Infatti, quest'ultima, rappresenta il "rumore" di una serie temporale.

Il metodo statistico scelto, per effettuare *Anomaly Detection,* è il metodo GESD (acronimo di *Generalized Extreme Studentized Deviate Test*). Tale metodo elimina progressivamente gli *outliers* ultizzando il T-test di Student, comparando tale test con un valore critico. Ogni qual volta che viene eliminato un *outlier*, viene effettuato nuovamente il test statistico. Si innesca, dunque, un *loop* finchè il test non scende al di sotto del valore critico. A quel punto tutti gli *outliers* sono considerati rimossi. Per maggiori dettagli sul metodo GESD, consultare la sezione 2.4.2. Tale modello, nel progetto di tesi verrà identificato come "gesd", dal metodo di rilevamento degli *outliers* applicato.

Infine, per l'analisi delle serie temporali, si è scelto di applicare due metodi sviluppati nell'ambito dei *data streaming*; ovvero, flussi di dati in *real-time.* In tale contesto, il primo modello è il metodo PEWMA. Tale metodo è stato ideato da *Kevin M.Carter* e *William W.Streile* autori della pubblicazione "*Probabilistic Reasoning For Streaming Anomaly Detection*"(sezione 2.4.2). Il termine PEWMA sta per *probabilistic* EWMA.

Per implementare l'algoritmo è stato utilizzato il pacchetto "otsad" e la funzione CpPewma. Inerentemente ai parametri della funzione utilizzati, sono stati considerati quelli di *default.* Nello specifico, è il parametro "n_train" assume un valore pari a 5. Quest'ultimo rappresenta il numero di osservazioni del *dataset* di "*train*" T, necessario per determinare la media e la deviazione standard dei dati. Il parametro "alpha0" è stato impostato con un valore pari a 0,8, mentre "beta" con il valore di 0,3. Infine "l" è la costante moltiplicativa che definisce lo stato di anomalia o meno di un'osservazione. I parametri "alpha0" e "beta" si riferiscono rispettivamente a α e β dell'equazione $\mu_t = \alpha(1 - \beta P_t)\mu_{t-1} + (1 - \alpha(1 - \beta P_t))x_t$. Per maggiori dettagli del modello consultare la sezione 2.4.2.

Il secondo modello è SD – EWMA, ideato da *Haider Raza, Girijesh Prasad, Yuhua Li*, autori della pubblicazione "*EWMA model based shift-detection methods for detecting covariate shifts in non-stazionary environments*". Tale algoritmo è caratterizzato da due *stage* quali le fasi di *training e* di *test.* Nella fase di *training* si definiscono i parametri del modello, al fine di stabilire le condizioni di stazionarietà, mentre la fase di *test* identifica i valori anomali, sfruttando la carta di controllo EMWA. Dunque, per verificare se un dato è anomalo si verifica se esso ricade all'interno del limite superiore e inferiore della carta considerata.

Per implementare l'algoritmo, è stato utilizzato il pacchetto *"Otsad"* del software R, tramite la funzione *"CpSdEmwa"* che svolge entrambe le fasi dell'algoritmo. Di *default* il parametro *threshold* assume un valore pari a 0,01, ed è il valore assunto dal parametro θ nella stima

della varianza $\sigma^2_{err(i)} = \theta err^2_{(i)} + (1-\theta)\sigma^2_{err(i-1)}$, mentre il parametro L che compare nel calcolo del limite di controllo superiore e inferiore della carta di controllo, di *default* , assume un valore pari a 3 ($UCL_{(i)} = z_{(i-1)} + L\sigma_{err(i-1)}$, $LCL_{(i)} = z_{(i-1)} - L\sigma_{err(i-1)}$). Per maggiori dettagli sul funzionamento dell'algoritmo vedere sezione 2.4.2.

L'ultimo modello considerato nel lavoro di tesi è *l'Isolation Forest*. Tale algoritmo sfrutta gli alberi decisionali al fine di isolare i valori anomali nel *set* di dati. Il principio sul quale si basa l'algoritmo è che le anomalie sono "diverse" e "poche" rispetto alle altre osservazioni del *dataset*, dunque facilmente identificabili. L'algoritmo genera dei partizionamenti recursivi andando a selezionare casualmente una *feature* e un valore divisivo per tale *feature* (che sia compreso tra il valore massimo e minimo assunto dalla feature in questione). Le anomalie avranno bisogno di meno partizioni casuali per poter essere isolate.

L'*Isolation Forest* non prende in considerazione il fattore tempo, in quanto non rappresenta un modello di serie temporali. Al fine di adattare l'algoritmo al tipo di dato cioè *Time Series* caratterizzate da molteplici stagionalità e, inoltre, per consentire un confronto con i modelli di serie temporali, si è scelto di procedere attraverso un processo di *feature engineering.* Al fine di perseguire l'obiettivo di analisi sopradescritto, per ogni metrica di *performance*, l'algoritmo è stato implementato aggiungendo, preliminarmente, tre *features* al *dataset:* il minuto (0\59), l'ora (0\23) e il giorno settimanale (0\6). Tuttavia, per ogni tipologia di metrica sono state effettuate analisi specifiche, in relazione alle sue caratteristiche stagionali. Le osservazioni e i risultati relativi alle singole metriche sono riportate nel paragrafo 5.2 dedicato ai risultati sperimentali.

Per implementare l'algoritmo è stata utilizzata la libreria "*Isotree*" e la funzione "*Isolation.forest*", disponibile sul software R. Come parametro è stato settato "ntrees=100". Quest'ultimo si riferisce al numero di alberi "*Itrees*" generati. Inoltre, è stato impostato "sample_size=256". Il parametro "sample_size" si riferisce al numero di campioni utilizzati per la generazione degli alberi. Tali parametri sono stati suggeriti dagli ideatori stessi del modello, nella pubblicazione del 2018 "*Isolation Forest*". Infine, è stato impostato "ndim=4" in modo da considerare come *features,* per effettuare la generazione degli alberi, anche le informazioni sul minuto, sull'ora e sul giorno della settimana della singola osservazione. Tale modello, nel progetto di tesi verrà identificato come "isoforest".

5.1.2 Metodo di analisi

Nel progetto di tesi è stato proposto un modello che integra tutti i metodi di *Anomaly Detection*. Quest'ultimo è stato ideato per rispondere all'assenza di etichettatura circa il reale stato di anomalia di un'osservazione. Infatti, in assenza di etichettatura, risulta un compito arduo determinare quale algoritmo sia in grado di effettuare *Anomaly Detection* nel modo più corretto. Tuttavia, lo scopo ultimo di analisi consisteva nel determinare un modello che fosse in grado di identificare correttamente le anomalie nel contesto di serie storiche caratterizzate da dati campionati al minuto.

Il modello integrato considera lo "*score*" per determinare lo stato di anomalia o meno di un'osservazione. Lo "*score*" viene calcolato considerando il rapporto tra il numero di metodi che classificano un'osservazione come anomala sul numero totale di metodi esaminati. Tanto più lo "score" assume un valore vicino ad 1, tanto più alta è la possibilità che l'osservazione sia realmente anomala. Il principio sul quale il modello si basa è quello di sfruttare la potenza di più algoritmi per superare le criticità che un singolo modello presenta, migliorando il risultato finale. Dunque, il concetto sul quale si fonda il modello integrato è il cosiddeto "*Majority Voting*"; se la maggior parte degli algoritmi classificano un'osservazione come anomala, il modello integrato assegna lo stato di anomalia a tale osservazione, appunto per maggioranza di voti.

Inoltre, si è scelto di effettuare il rilevamento delle anomalie applicando i metodi di *Anomaly Detection* in modo recursivo, considerando delle finestre mobili.

A tal fine è stato definito un *dataset* denominato di "*training"* caratterizzato da un numero di osservazioni pari a 14 giorni, dunque a 20160 osservazioni; le restanti osservazioni, invece, caratterizzeranno il *dataset* di "*test".* Per effettuare *Anomaly Detection* delle metriche di *performance* è stata utilizzata una metodologia che sfrutta il concetto di finestra mobile: i modelli vengono applicati al *dataset* di *train* e all'osservazione del *dataset* di *test* immediatamente successiva ai dati di *training*. Il fine è quello di caratterizzare quest'ultima osservazione come anomala o meno e tale metodologia consente di effettuare l' identificazione sulla base del numero di osservazioni settate. Dunque, il *test set* rappresenta il *dataset* con le nuove informazioni, di cui si vuole rilevare l'eventuale stato di anomalia.

Il concetto di finestra mobile si riferisce al fatto che il *dataset* di *training* viene aggiornato e si sposta; infatti, non appena si applicano i modelli su un osservazione essa entrerà a far

parte del dataset di *train* e verrà eliminata la prima osservazione dello stesso, in modo tale da mantenere costante il numero di osservazioni al suo interno. Tale processo si ripete finché non vengono esaminati tutti i dati del *dataset* di *test* iniziale. La metodologia descritta consente di effettuare il rilevamento delle anomalie ogni minuto.

Per determinare il numero di osservazioni da considerare nel dataset di *train* è stata effettuata una valutazione circa la stagionalità della serie considerata. L'analisi ha mostrato la presenza di elementi stagionali giornalieri e orari. In alcune metriche risulta evidente che i valori assunti nelle giornate non lavorative, quali il sabato e la domenica sono sistematicamente inferiori a quelli settimanali e lavorativi. Tale analisi risulta necessaria per considerare un valore ottimale e adatto allo specifico contesto di *Business*. Infatti scegliere un *dataset* di *train* caratterizzato da un numero di osservazioni pari a quelle registrate in singolo giorno dunque, 1440 osservazioni (60 osservazioni/ ora * 24 ore/giorno) non consentirebbe di cogliere le stagionalità giornaliera e oraria caratterizzanti le serie temporali, oggetto di analisi. Infatti, per cogliere la stagionalità oraria è necessario considerare più di un giorno, per verificare se in uno stesso orario della giornata, in giorni diversi, si ripetono sistematicamente dei picchi. Per quanto riguarda la stagionalità giornaliera, invece, è necessario comprendere, nel *dataset* di *train*, un numero di osservazioni di almeno due settimane; tale finestra temporale consente di rilevare se in uno stesso giorno, appartenete a settimane differenti, ci sono picchi sistematici.

Dunque per cogliere i due *patterns* stagionali rilevati e per considerare nel calcolo un numero sufficiente di osservazioni, si è scelto di considerare una finestra di due settimane e quindi, un numero di osservazioni uguale a 20160. Nella sezione 5.2 del capitolo in questione si riportano analisi e risultati relativi alla scelta dell'ampiezza della finestra di calcolo, in relazione alle specifiche caratteristiche stagionali delle singole metriche.

Infine, l'utilizzo di una "*rolling window"*, appunto la finestra mobile descritta, invece che una finestra definita come "*expanding window"* è una conseguenza del numerosità del *dataset* analizzato. Dunque, per rendere l'analisi più efficiente dal punto di vista computazionale sono stati applicati i modelli su una finestra mobile.

In conclusione si riporta uno schema che rappresenta l'architettura complessiva del progetto di tesi.

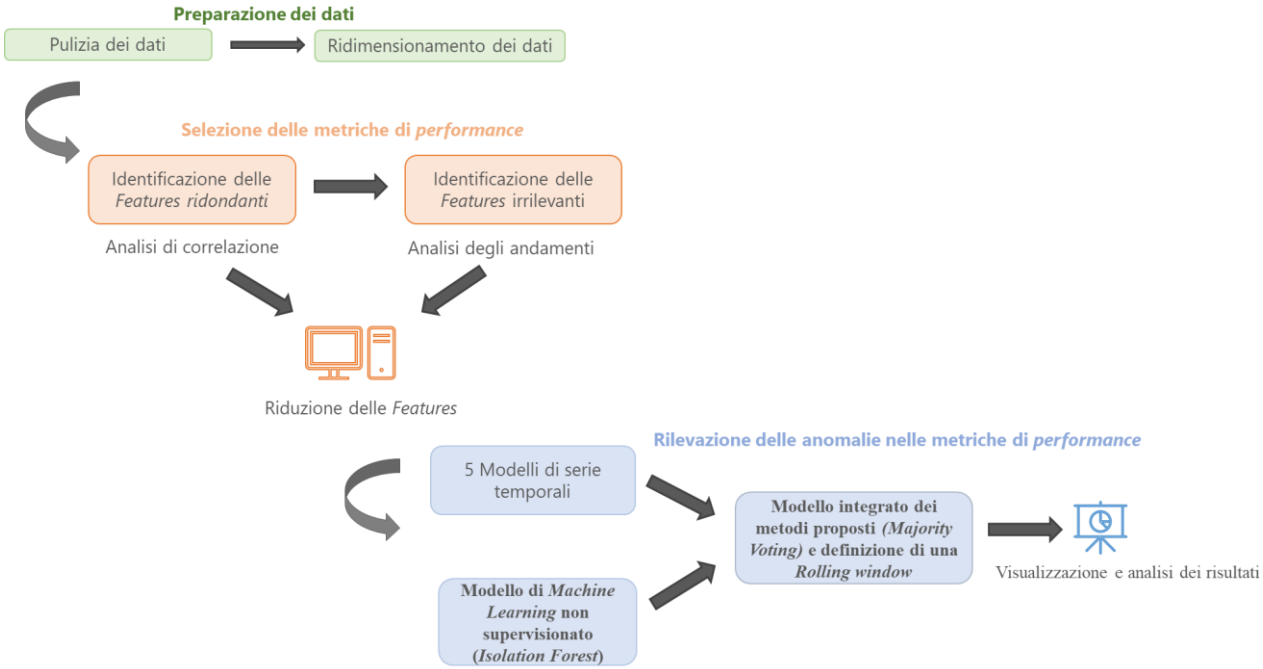

Figura 5.2 : Rappresentazione schematica dell'intero progetto di tesi.

5.2 Risultati sperimentali

In questa sezione vengono illustrati i risultati sperimentali ottenuti dall'applicazione degli algoritmi di *Anomaly Detection*. Per l'elaborazione dei risultati sono state esaminate le 122 metriche di *performance* caratterizzanti il *set* di dati. In primo luogo, si mostrano i confronti tra ciascuna metodologia di *Anomaly Detection* proposta e si riportano considerazioni generali. Tali valutazioni sono relative sia al tempo computazionale, sia a differenze percentuali, tra i vari modelli, nel rilevamento delle anomalie. Successivamente si illustrano i risultati relativi a specifiche metriche di *performance* al fine di mostrare la capacità di rilevamento delle anomalie globali, effettuando considerazioni su ciascun algoritmo proposto. Inoltre, si mostra il funzionamento del modello integrato, proposto nel progetto di tesi. In seguito, si effettuano analisi e si presentano risultati sull'impatto che l'ampiezza della finestra di calcolo determina nel rilevamento delle anomalie. Infine, si analizza *l'Isolation Forest*, effettuando considerazioni relative ai parametri settati per implementare l'algoritmo.

Nel progetto di tesi sono stati analizzati sei algoritmi di *Anomaly Detection*. Nello specifico, cinque modelli di serie temporali e un modello che sfrutta gli alberi decisionali, quale l'*Isolation Forest* . Per maggiori delucidazioni consultare la sezione 5.1.1.

Il confronto computazionale tra gli algoritmi ha condotto all'esclusione di un primo modello, quale la procedura proposta da *Chung Chen* e *Lon-Mu Liu*, basata sui modelli ARIMA. Si ricorda che tale procedura è stata nominata come *"tso"*, dalla funzione che la implementa sul *software R.* L'algoritmo è stato escluso in quanto inefficiente dal punto di vista computazionale. Per verificarne l'inefficienza è stata considerata la metrica *"Active Serial Sessions"* e una finestra mobile di calcolo di 1 giorno cioè 1440 osservazioni. Dunque, sono stati confrontati i tempi computazionali di ciascun algoritmo. In primo luogo sono stati registrati i tempi computazionali, in minuti, per il rilevamento delle anomalie su 5 osservazioni. Il numero di osservazioni su cui è stato effettuato il calcolo è stato incrementato fino a considerarne 25. In tabella 5.1 si riportano i risultati ottenuti.

	METODI					
NUMERO OSSERVAZIONI	**tso**	**iqr**	**gesd**	**pewma**	**sdewma**	**isoforest**
5 osservazioni	5 min	0 min	0 min	0 min	0 min	0 min
10 osservazioni	12 min	0 min	0 min	0 min	0 min	0 min
15 osservazioni	17 min	0 min	0 min	0 min	0 min	0 min
20 osservazioni	21 min	0 min	0 min	0 min	0 min	0 min
25 osservazioni	26 min	0 min	0 min	0 min	0 min	0 min

Tabella 5.1: Confronto computazionale tra i metodi di *Anomaly Detection*. Calcolo ottenuto considerando una Rolling Window di 1440 osservazioni (1 giorno). La tabella mostra l'inefficienza, dal punto di vista computazionale, del metodo "*tso*".

I risultati mostrano, in modo evidente, come il modello "tso" sia inefficiente dal punto di vista computazionale. Infatti, "tso" impiega alcuni minuti per rilevare le anomalie mentre gli altri metodi oggetto di analisi, quali "iqr", "gesd", "pewma", "sdewma" e "isoforest", soltanto alcuni secondi. Si evince, inoltre, come all'aumentare del numero di osservazioni, considerando un incremento di 5 osservazioni alla volta, per il metodo "tso" ci sia una crescita del tempo computazionale, al contrario delle altre metodologie esaminate. In figura 5.3 si riporta l'andamento del tempo di calcolo, per il metodo "tso", considerando una finestra mobile di 1440 osservazioni, in funzione del numero di osservazioni su cui si effettua *Anomaly Detection.*

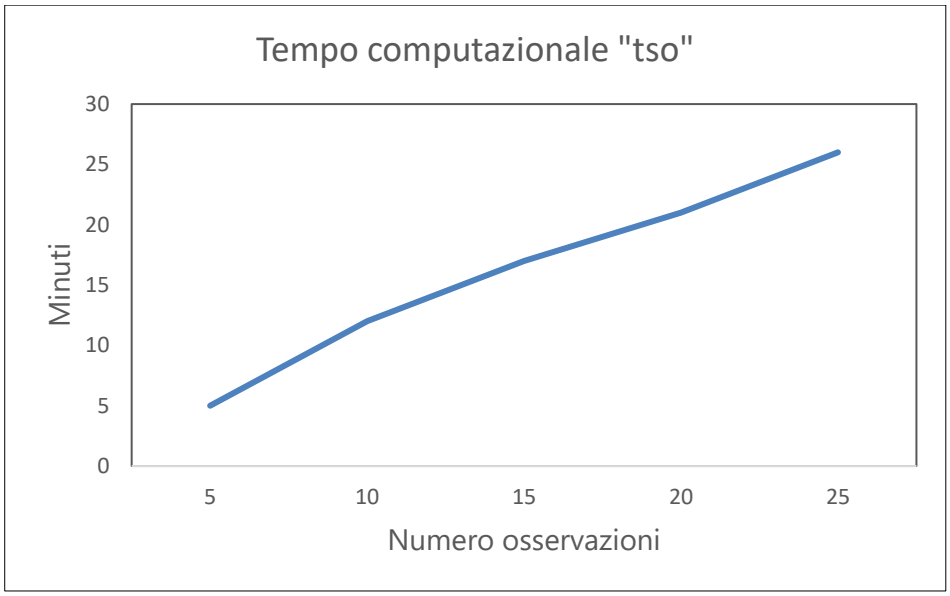

Figura 5.3**:** Andamento del tempo di calcolo, per il metodo "tso", considerando una finestra mobile di 1440 osservazioni, in funzione del numero di osservazioni su cui si effettua *Anomaly Detection.* La figura mostra l'andamento crescente del tempo computazionale.

I risultati ottenuti sono coerenti con quanto affermato in letteratura. Infatti, i modelli di serie temporale che sono stati scelti per il progetto di tesi nascono per essere implementati con dati ad alta frequenza, a differenza del metodo tradizionale proposto da *Chung Chen* e *Lon-Mu Liu*, basato sui modelli ARIMA. Come esposto in sezione 2.4.2, il metodo nominato come "iqr" sfrutta una decomposizione che è stata ideata per le serie temporali caratterizzate da proprietà più complesse come, ad esempio, cicli stagionali multipli. Tali proprietà caratterizzano la maggior parte dei *dataset* reali in cui dati sono rilevati con un'alta frequenza (giornaliera, oraria o al minuto). Il metodo "gesd", sfrutta una decomposizione che è stata implementata dagli ideatori considerando la metrica Tweets Per Sec (TPS), sviluppando ,dunque, una tecnica statistica al fine di rilevare automaticamente le anomalie *long-term* in dati *Cloud*. Tali dati sono caratterizzati da una grande velocità di generazione, in *real-time*, e da grandi volumi. Infine, i metodi "pewma" e "sdewma" sono stati ideati nel contesto dei *data streaming*, anch'essi caratterizzati da una natura *real-time.*

I risultati riportati in tabella 5.1 relativi alla metrica "*Active Serial Sessions"* sono stati ottenuti considerando una finestra di calcolo di 1 giorno cioè 1440 osservazioni ed è stato calcolato il tempo computazione per rilevare le anomalie considerando un massimo di 25 osservazioni. Se si effettuasse l'analisi utilizzando una finestra di calcolo più ampia, o se si effettuasse *l'Anomaly Detection* al minuto su un numero maggiore di osservazioni, il tempo

computazionale crescerebbe notevolmente. Dunque, se si considerano le dimensioni del *dataset*, caratterizzato da 79199 osservazioni, si evince in maniera ancora più evidente come non sia opportuno utilizzare il metodo "tso" nel contesto dei *Big data*.

Dopo aver confrontato gli algoritmi di *Anomaly Detection* dal punto di vista computazionale, sono stati effettuati dei confronti a coppie di algoritmi. Lo scopo di analisi consisteva nel verificare le percentuali di coincidenza del risultato di *Anomaly Detection* ottenuto tra i due algoritmi analizzati. Il confronto è stato effettuato su più metriche di *performance* e considerando una finestra mobile costituita da 14 giorni, dunque da 20160 osservazioni. Esaminare un numero di osservazioni ampio per effettuare il confronto ha consentito di ottenere un risultato più attendibile.

I risulati dell'analisi mostrano che le differenze tra un algoritmo e l'altro sono dipendenti dalla specifica serie temporale analizzata; non si rilevano coppie di algoritmi che sistematicamente, considerando più metriche di *performance*, presentano risultati con maggiori percentuali di coincidenza nel rilevamento delle anomalie. Inoltre, si evince che, nonostante l'*Isolation forest* sia un modello di *machine learning* che sfrutta gli alberi decisionali e non rappresenti un modello di serie temporali, esso non si discosta maggiormente dai risultati ottenuti dalle altre metodologie. A titolo di esempio si riportano alcuni risultati ottenuti per la metrica "*Redo Writes Per Txn*". Le analisi hanno mostrato che su 20160 osservazioni i metodi nominati come "iqr" e "gesd" differiscono nel rilevamento delle anomalie per un totale di 2883 osservazioni, dunque la coincidenza è dell'85,70%. Il metodo "iqr" ha una coincidenza dell'89,13% con il metodo "pewma" e del 93,47% con il metodo "isoforest".

Successivamente, partendo dalle metriche caratterizzate dai picchi più alti, classificate in ordine di valore massimo decrescente (figura 4.21 sezione 4.3.2), sono state effettuate analisi al fine di verificare la capacità degli algoritmi di identificare correttamente i picchi come valori anomali. Per effettuare le analisi sono state considerate finestre mobili di 14 giorni e, dunque, 20160 osservazioni. La prima metrica di cui si riporta il risultato ottenuto è *"Global Cache Average CR Get Time"*. Quest'ultima presenta un solo picco il cui valore standardizzato è pari a 281,41. Gli altri valori assunti dalla metrica oscillano intorno allo zero. I risultati mostrano che tutti gli algoritmi, oggetto di analisi, identificano come anomalo il picco relativo al giorno 27/01/2022, verificatosi alle ore 01:09. Inoltre, tutti gli algoritmi

rilevano come anomala anche l'osservazione immediantamente precedente al picco. In tabella 5.2 si riportano i risulatti ottenuti per le due ossservazioni sopra citate.

tempo	*valore*	*iqr*	*gesd*	*pewma*	*sdewma*	*isoforest*	*score*	*anomaly*
27/01/2022 01:08	0,046	1	1	1	1	1	1.0	1
27/01/2022 01:09	281,41	1	1	1	1	1	1.0	1

Tabella 5.2: *Global Cache Average CR Get Time:* risultati di *Anomaly Detection* dei 5 metodi, oggetto di analisi. La tabella mostra la coerenza tra i risultati ottenuti dai 5 algoritmi nell'identificare il picco come valore anomalo.

Dalla tabella 5.2 si evince che tutti i modelli classificano le due osservazioni del 27/01/2022 come anomale; infatti, in corrispondenza di ciascun metodo, è presente un valore pari a "1" che indica lo stato di anomalia. Si dimostra, dunque, l'efficacia delle metodologie proposte nel rilevare le anomalie globali. Quest'ultime rappresentano le osservazioni che si discostano dal normale *range* di oscillazione dei dati. Tuttavia, si evince che anche alcune osservazioni che non rappresentano dei picchi vengono considerate anomale, come quella relativa al 27/01/2022 alle ore 01:08 che assume un valore pari a 0,046. Queste tipologie di anomalie rappresentano le cosiddette anomalie contestuali.

Nel caso delle due osservazioni riportate in tabella 5.2 il valore dello "score" è pari a 1, in quanto cinque modelli su cinque classificano la specifica osservazione come un'anomalia. Dunque il valore assunto dalla colonna "anomaly" è pari a 1; infatti, il modello integrato, proposto nel progetto di tesi, considera come anomalie le osservazioni in cui almeno tre modelli su cinque la classificano come tale.

Tuttavia, alcune osservazioni sono state classificate come anomale soltanto da alcuni algoritmi. In figura si riporta la metrica con le anomalie rilevate dai vari metodi. Tale raffigurazione rappresenta 104 osservazioni e, dunque, una finestra temporale che parte dal 27/01/2022 alle ore 00:00 e termina il 27/01/2022 alle ore 01:44.

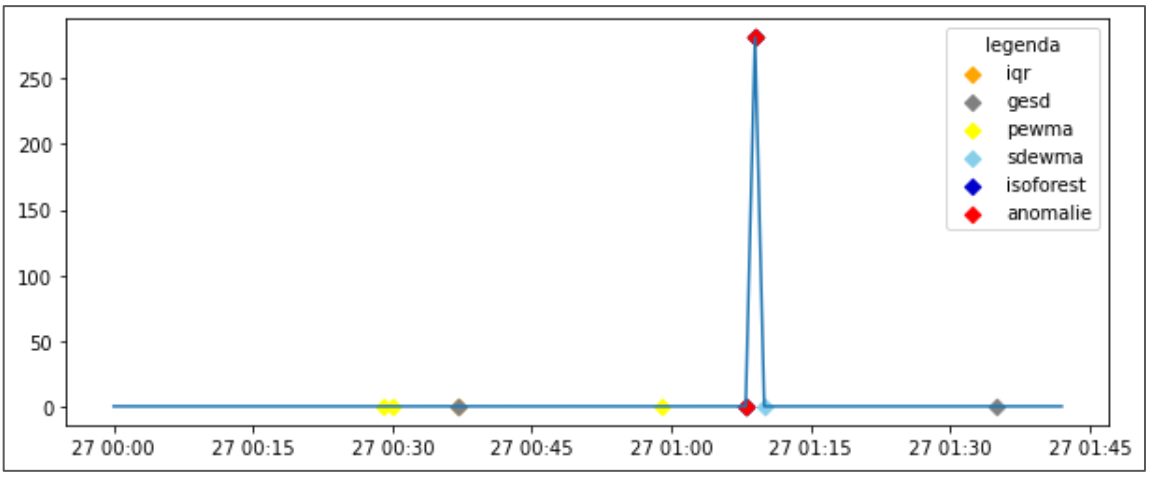

Figura 5.4 : *Global Cache Average CR Get Time:* risultati di *Anomaly Detection* dei 5 metodi, oggetto di analisi. La raffigurazione mostra l'andamento della metrica considerando 104 osservazioni e i risulati di *Anomaly Detection*. Dalla figura si evince che soltanto due osservazioni vengono classificate come anomale dal modello integrato, appunto il picco e l'osservazione immediatamente precedente. La figura mostra, nell'intervallo considerato, la presenza di risultati discordanti per altre osservazioni, non classifciate come anomale dal modello integrato.

In rosso si evidenziano le osservazioni che vengono classificate come anomalie dal modello integrato, proposto nel progetto di tesi.

Nel dettaglio, l'osservazione relativa al 27/01/2022 alle ore 00:37 è stata classificata come anomala esclusivamente dai metodi "iqr" e "gesd". Dalla figura 5.4 si osserva un "rombo" di colore grigio (metodo "gesd") che si sovrappone ad un altro di colore arancione (metodo "iqr"), come è possibile osservare dai contorni del rombo stesso. L'osservazione del 27/01/2022 delle 01:35, invece, risulta anomala soltanto secondo il metodo "gesd". Il metodo "pewma" identifica come anomale alcune osservazioni come, per esempio, quelle del 27/01/2022 alle ore 00:29 e alle ore 00:30. In questo caso di analisi, l'algoritmo "isoforest" identifica come anomale soltanto le due osservazioni riportate in tabella 5.2.

I risultati relativi alla metrica *"Global Cache Average Current Get Time"* mostrano che anche per la metrica in questione, come per "*Global Cache Average CR Get Time",* tutti gli algoritmi sono in grando di identificare l'unica anomalia globale caratterizzante la metrica stessa, quale l'osservazione relativa al 28/01/2022 alle ore 23:00. Tale anomalia assume un valore pari a 281,40. In tabella 5.3 si riportano i risultati ottenuti per ciascun metodo in corrispondenza del picco e di alcuni valori precedenti.

tempo	valore	iqr	gesd	pewma	sdewma	isoforest	score	anomaly
28/01/2022 22:56	-0,009	0	0	0	0	0	0	0
28/01/2022 22:57	-0,010	0	0	0	0	0	0	0
28/01/2022 22:58	-0,009	0	0	0	0	0	0	0
28/01/2022 22:59	-0,002	0	0	0	0	0	0	0
28/01/2022 23:00	281,40	1	1	1	1	1	1	1

Tabella 5.3: *Global Cache Average Current Get Time:* risultati di *Anomaly Detection* dei 5 metodi, oggetto di analisi. La tabella mostra la coerenza tra i risultati ottenuti dai 5 algoritmi nell'identificare il picco come valore anomalo.

Considerando la metrica "*Row Cache Hit Ratio",* anch'essa presenta un unico picco e viene individuato da tutti gli algoritmi. Tali risultati mostrano che il picco rappresenta effettivamente un'anomalia. Inoltre anche per la metrica di *performance* "*PX downgraded to serial Per Sec*" i risultati confermano la capacità degli algoritmi di identificare le anomalie globali. La metrica in questione presenta valori che oscillano intorno allo zero ed caratterizzata da due picchi, entrambi rilevati come anomali in modo coerente dai cinque modelli proposti.

Tuttavia, sono state rilevate alcune eccezioni. Per esempio la metrica *"Enqueue Waits Per Txn*" è una metrica di *performance* che presenta valori che oscillano intorno allo zero e alcune oscillazioni consecutive più rilevanti che determinano alcuni picchi. In questo caso i metodi "pewma" e "sdewma" non classificano quei picchi come valori anomali. Tuttavia il modello integrato, basato sul concetto di "score" determina lo stato di anomalia dei picchi consecutivi. In figura 5.5 si riporta una raffigurazione della metrica *"Enqueue Waits Per Txn*" e delle anomalie rilevate dai vari modelli e dal modello integrato.

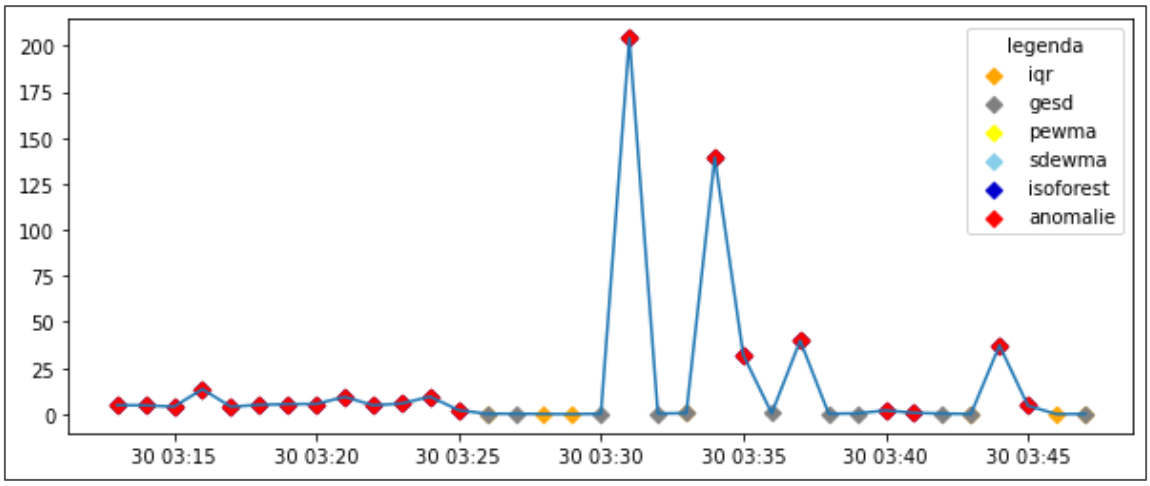

Figura 5.5 : *Enqueue Waits Per Txn*: risultati di *Anomaly Detection* dei 5 metodi, oggetto di analisi. Dalla figura si evince che il modello integrato classifica come anomali i 4 picchi e anche altre osservazioni. La figura mostra, nell'intervallo considerato, la presenza di risultati discordanti per alcune osservazioni, non classifciate come anomale dal modello integrato.

In tabella 5.4 si riportano alcune ossservazioni relative alla metrica *"Enqueue Waits Per Txn*" che il modello integrato considera come anomalie (colonna "anomaly" caratterizzata da valori pari a 1).

tempo	valore	iqr	gesd	pewma	sdewma	isoforest	score	anomaly
30/01/2022 03:13	4,83	1	1	0	0	1	0.6	1
30/01/2022 03:14	4,71	1	1	0	0	1	0.6	1
30/01/2022 03:15	3,99	1	1	0	0	1	0.6	1
30/01/2022 03:16	13,50	1	1	0	0	1	0.6	1
30/01/2022 03:17	3,96	1	1	0	0	1	0.6	1
30/01/2022 03:18	5,01	1	1	0	0	1	0.6	1
30/01/2022 03:19	5,39	1	1	0	0	1	0.6	1
30/01/2022 03:20	5,50	1	1	0	0	1	0.6	1
30/01/2022 03:21	9,35	1	1	0	0	1	0.6	1
30/01/2022 03:22	4,73	1	1	0	0	1	0.6	1
30/01/2022 03:23	5,74	1	1	0	0	1	0.6	1
30/01/2022 03:24	9,47	1	1	0	0	1	0.6	1
30/01/2022 03:25	2,11	1	1	0	0	1	0.6	1
30/01/2022 03:31	204,20	1	1	0	0	1	0.6	1
30/01/2022 03:34	139,40	1	1	0	0	1	0.6	1
30/01/2022 03:35	32,07	1	1	0	0	1	0.6	1
30/01/2022 03:37	39,90	1	1	0	0	1	0.6	1
30/01/2022 03:40	1,97	1	1	0	0	1	0.6	1
30/01/2022 03:41	0,68	1	1	0	0	1	0.6	1
30/01/2022 03:44	37,34	1	1	0	0	1	0.6	1
30/01/2022 03:45	4,53	1	1	0	0	1	0.6	1

Tabella 5.4 : *Enqueue Waits Per Txn*: risultati di *Anomaly Detection* dei 5 metodi, oggetto di analisi. La tabella mostra soltanto le osservazioni classificate come anomale dal modello integrato (colonna "anomaly"=1)

Il modello che integra i cinque algoritmi esaminati, proposto nel progetto di tesi, considera come anomalie quelle osservazioni in cui almeno tre metodi su cinque le classificano come tali. A tal proposito la tabella 5.4 mostra il funzionamento del modello integrato.

Il modello integrato considera lo "score" per determinare lo stato di anomalia o meno di un'osservazione. Tanto più lo "score" assume un valore vicino ad 1, tanto più alta è la possibilità che l'osservazione sia realmente anomala. Il modello integrato considera come anomalie quelle osservazioni che sono state classificate come anomale da più algoritmi e che hanno un valore dello score >= 0,6. Il principio sul quale il modello si basa è quello di sfruttare la potenza di più algoritmi per superare le criticità che un singolo algoritmo presenta, migliorando il risultato finale.

Analizzando i risulati relativi alla metrica di *performance "Enqueue Waits Per Txn*" si evince che i modelli "pewma" e "sdewma" non identificano come anomalie gli unici picchi caratterizzanti la metrica analizzata. Tuttavia, i risultati di "pewma" e "sdewma" non risultano altrettanto critici per altre metriche quali, per esmepio, *"Global Cache Average Current Get Time" e Global Cache Average CR Get Time"*. Il caso specifico di *"Enqueue Waits Per Txn*" mostra come il modello integrato sfrutti i risultati di più algoritmi per determinare lo stato finale di un'osservazione, superando le criticità dei singoli metodi.

Un altro esempio che si riporta, al fine di mostrare il funzionamento del modello proposto, è relativo alla metrica *"Logical Reads Per Sec"*. Essa risulta caratterizzata da molteplici picchi. Alcuni di essi vengono identificati come anomali da tutti e cinque gli algoritmi come, per esempio, l'osservazione del 27/01/2022 alle 00:13 e quella del 28/01/2022 alle 00:13. Altri, invece, sono considerati anomali solo da alcuni metodi. In tabella 5.5 si riportano le osservazioni che il modello integrato calcola come anomalie, considerando una finestra di dati tra il 26/01/2022 alle 00:00 al 29/01/2022 alle ore 00:30. In figura 5.6 si riporta una raffigurazione della metrica dove si evidenizano, con il colore rosso, le anomalie determinate dal modello integrato.

tempo	valore	iqr	gesd	pewma	sdewma	isoforest	score	anomaly
26/01/2022 00:14	7,34	1	1	0	0	1	0.6	1
27/01/2022 00:13	7,51	1	1	1	1	1	1	1
27/01/2022 00:14	8,33	1	1	0	0	1	0.6	1
28/01/2022 00:13	8,74	1	1	1	1	1	1	1
28/01/2022 00:14	9,03	1	1	0	0	1	0.6	1
29/01/2022 00:13	3,62	1	0	1	1	1	0.8	1
29/01/2022 00:14	7,08	1	1	1	1	1	1	1
29/01/2022 00:15	7	1	1	0	0	1	0.6	1

Tabella 5.5: Logical Reads Per Sec : risultati di *Anomaly Detection* dei 5 metodi, oggetto di analisi. La tabella mostra soltanto le osservazioni classificate come anomale dal modello integrato (colonna "anomaly"=1). Finestra di dati dal 26/01/2022 alle 00:00 al 29/01/2022 alle ore 00:30.

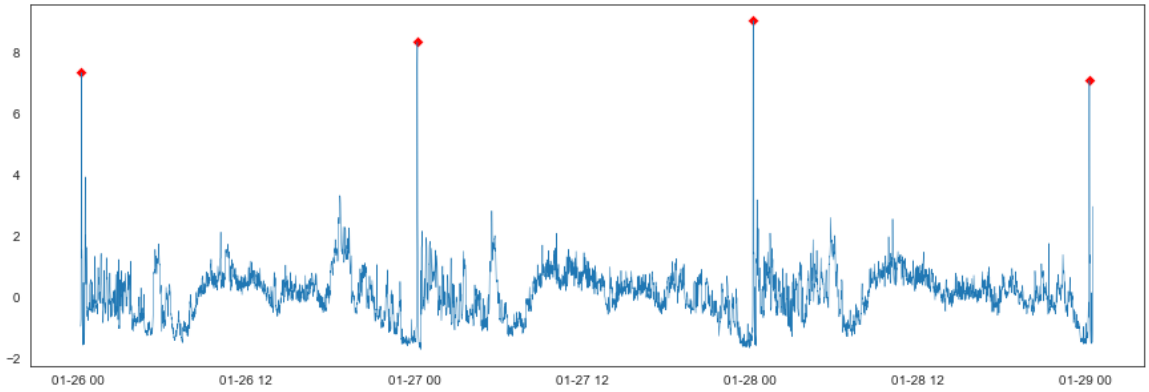

Figura 5.6 : *Logical Reads Per Sec :* risultati di *Anomaly Detection* del modello integrato. Finestra di dati dal 26/01/2022 alle 00:00 al 29/01/2022 alle ore 00:30.

Dalla figura 5.6 si evince che il modello integrato calcola come anomalie esclusivamente i picchi della finestra considerata. Alcune osservazioni, come quella del 26/01/2022 alle 00:30 risulta anomala soltanto secondo il metodo "iqr" e "isoforest", dunque essa ha una bassa probabilità di essere realmente tale.

Dopo aver verificato la capacità di individuare le anomalie globali da parte degli algoritmi, sono state effettuate analisi relative all'ampiezza della finestra mobile. Quest'ultima, come esposto in sezione 5.1.2 rappresenta il numero di osservazioni che vengono utilizzate dagli algoritmi per effettuare *Anomaly Detection*.

Al fine di esaminare l'impatto che l'ampiezza della finestra mobile determina sul risultato finale, è stata analizzata la metrica *"Phisycal write per sec"*. Quest'ultima è caratterizzata

da rilevanti *patterns* stagionali. In figura 5.7 si riporta un grafico che consente di coglierne la stagionalità.

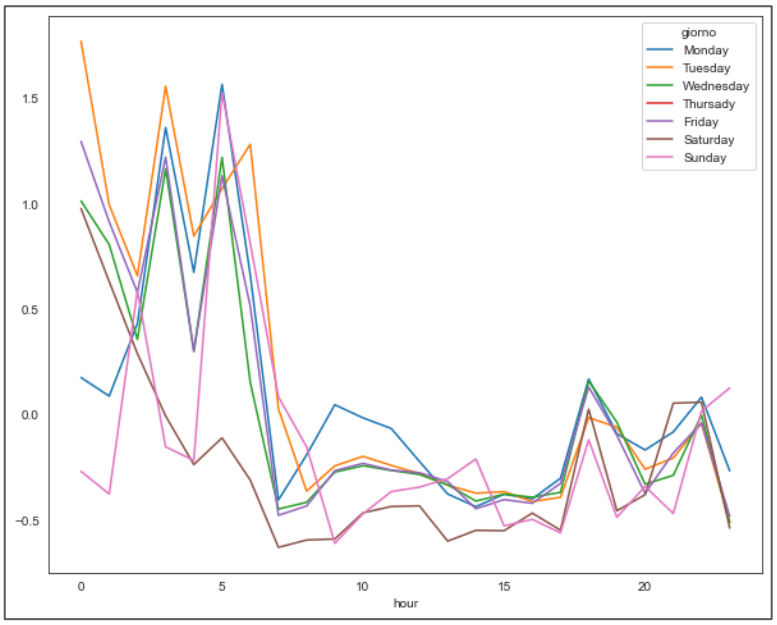

Figura 5.7 : *Phisycal write per sec,* metrica con *patterns* stagionali significativi. La figura mostra i *patterns* giornalieri e orari. La legenda mostra i colori utilizzati per rappresentare i valori assunti dalla metrica nei diversi giorni della settimana. L'asse delle ascisse rappresenta le ore.

Il grafico mostrato in figura 5.7 riporta i valori assunti dalla metrica nei distinti giorni della settimana. Per evidenziare i *patterns* giornalieri è stato utilizzato un colore differente per ogni giorno della settimana, come si evince dalla legenda in figura 5.7. L'asse delle ascisse rappresenta le ore. Dunque, dal grafico è possibile cogliere anche la presenza di stagionalità orarie.

Nel caso in esame si evince la presenza di una stagionalità giornaliera, in quanto i valori assunti dalla metrica il sabato (colore marrone in figura 5.7) sono inferiori rispetto ai valori assunti negli altri giorni della settimana. Inoltre, la stagionalità oraria si coglie verificando che i grafici della metrica, nei differenti giorni della settimana, in uno steso orario, seguono lo stesso andamento.

Sono stati calcolati i risultati di *Anomaly Detection* , relativi alla metrica *"Phisycal write per sec",* nel caso di una finestra mobile di 1 giorno e di una finestra mobile di 14 giorni. In tabella 5.6 si riportano i risultati di alcune osservazioni ottenuti considerando una finestra di calcolo di 1440 osservazioni (1 giorno), mentre in tabella 5.7 i risultati sono relativi a un calcolo effettuato su 20160 osservazioni (14 giorni).

tempo	valore	iqr	gesd	pewma	sdewma	isoforest	score	anomaly
20/01/2022 00:13	1,85	1	1	0	0	1	0.6	1
20/01/2022 00:14	4,09	1	1	0	1	1	0.8	1
20/01/2022 00:15	6,31	1	1	0	1	1	0.8	1
20/01/2022 00:16	4,02	1	1	0	0	1	0.6	1
20/01/2022 00:17	1,61	1	1	0	1	1	0.8	1

Tabella 5.6 : *Phisycal write per sec :* risultati di *Anomaly Detection* dei 5 metodi, oggetto di analisi e del modello integrato. La tabella mostra i risultati di alcune osservazioni ottenuti considerando una finestra di calcolo di 1440 osservazioni (1 giorno).

tempo	valore	iqr	gesd	pewma	sdewma	isoforest	score	anomaly
20/01/2022 00:13	1,85	0	1	0	0	1	0.4	0
20/01/2022 00:14	4,09	1	0	0	1	1	0.6	1
20/01/2022 00:15	6,31	1	1	0	1	1	0.8	1
20/01/2022 00:16	4,02	1	1	0	0	1	0.6	1
20/01/2022 00:17	1,61	0	0	0	1	1	0.4	0

Tabella 5.7 : *Phisycal write per sec :* risultati di *Anomaly Detection* dei 5 metodi, oggetto di analisi e del modello integrato. La tabella mostra i risultati di alcune osservazioni ottenuti considerando una finestra di calcolo di 20160 osservazioni (14 giorni).

Dal confronto delle tabelle 5.6 e 5.7, si evince che i risultati di *Anomaly Detection* sono dipendenti dall'ampiezza della finestra di calcolo. Nel caso in esame, si rilevano differenze per i due algoritmi "iqr" e "gesd". Infatti, in tabella 5.6 si osserva che il dato relativo al giorno 20/01/2022 alle 00:17, che assume il valore di 1,61, viene classificato come anomalo da tutti i metodi ad esclusione di "pewma". Di conseguenza, anche il modello integrato che sfrutta il concetto di "score" classifica l'osservazione come anomala ("anomaly"=1). Tuttavia, considerando una finestra di calcolo di 14 giorni, tabella 5.7, la stessa osservazione non rappresenta un'anomalia per i metodi "iqr" "gesd" e "pewma". Ciò determina che anche il modello integrato avrà un risultato differente rispetto a quello ottenuto considerando una finestra di 1 giorno.

I risultati mostrano le criticità derivanti dalla scelta dell'ampiezza della finestra di calcolo e di come ciò sia impattante sul risultate finale. La scelta di effettuare *Anomaly Detection* su finestre mobili deriva dalla numerosità del *dataset* e dalla conseguente necessità di avere algoritmi efficienti dal punto di vista computazionale. Le analisi condotte hanno evidenziato la presenza di stagionalità nella maggior parte delle metriche, ad esclusione di metriche, come *"Global Cache Average CR Get Time",* rappresentata in figura 5.4, caratterizzate da

picchi singoli. Per gestire le stagionalità giornaliere e orarie si ritiene che i risultati possano essere più accurati se si considerano finestra temporali di almeno 14 giorni.

Nel caso in esame, tabelle 5.6 e 5.7, è interessante notare che le principali differenze sono relative ai due metodi "iqr" e "gesd" che sfruttano il concetto di decomposizione delle serie temporali. Si nota che molti valori assunti dalla metrica oscillano intorno allo zero, di conseguenza i metodi sopracitati, determinano anomala un osservazione che assume un valore di 1,61, considerando 1440 osservazioni. Tramite l'utilizzo di finestre di più giorni, invece, gli algoritmi, che sfruttano la decomposizione, determinano la stagionalità dell'osservazione considerata e la conseguente non anomalia della stessa. Infine, si evince come, nel caso in esame, il modello integrato mostri più anomalie se la finestra di analisi è di 1 giorno e di come la scelta dell'ampiezza della finestra sia impattante non solo sul risultato dei singoli algoritmi ma anche sul modello che li integra.

Dopo aver analizzato una metrica caratterizzata da *patterns* stagionali rilevanti, cioè *"Phisycal write per sec",* si è scelto di focalizzare l'attenzione sulla metrica di *performance "Consistent Read Gets Per Txn"* che presenta caratteristiche stagionali meno evidenti. A tal proposito si riporta il grafico che mostra i *patterns* stagionali della stessa.

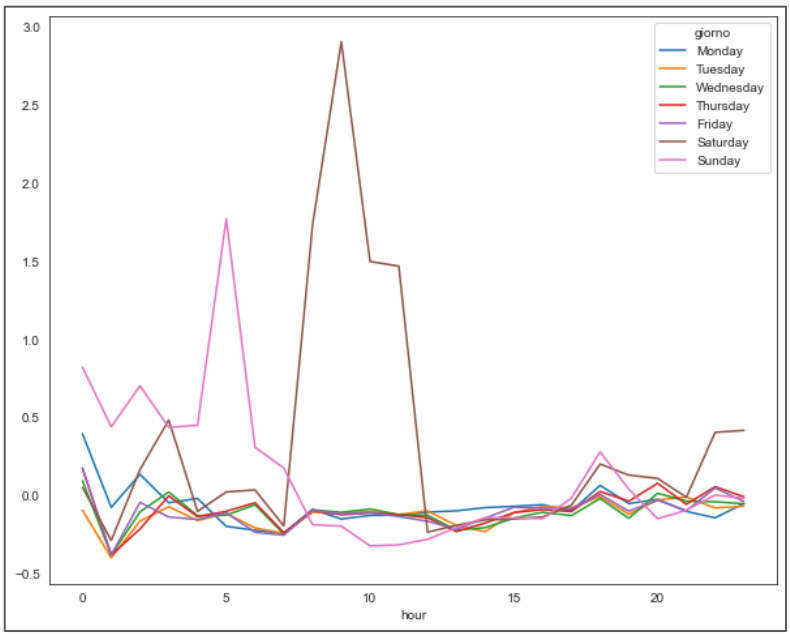

Figura 5.8 : *Consistent Read Gets Per Txn,* metrica con *patterns* stagionali non particolarmente significativi. La figura mostra i *patterns* giornalieri e orari. La legenda mostra i colori utilizzati per rappresentare i valori assunti dalla metrica nei diversi giorni della settimana. L'asse delle ascisse rappresenta le ore.

Per analizzare i risultati è stata considerata, in primo luogo , una finestra temporale di cinque giorni, dal lunedì al venerdì. Nello specifico è stata effettuata *Anomaly detection* su finestre di 7200 osservazioni. In tabella 5.8 si riportano alcuni risultati ottenuti dagli algoritmi. In seguito, è stata effettuata l'analisi considerando una finestra di sei giorni comprendente anche la domenica. Infatti, dalla figura 5.8 si evince che la domenica i valori assunti dalla metrica sono maggiori rispetto agli altri giorni della settimana. I risultati ottenuti sono riportati in tabella 5.9 .

Lo scopo di questa analisi è mostrare come modificando l'estensione della finestra di calcolo di un solo giorno, ciò può essere determinante sul risultato finale di *Anomaly detection*. È stato scelto di esaminare la metrica *"Consistent Read Gets Per Txn"* in quanto si evince, in modo evidente, come in tutti i giorni della settimana i valori assunti dalla metrica siano più o meno omogenei; solo la domenica essi si discostano notevolmente dal *range* di oscillazione dei dati. Dunque, è stato effettuato il confronto tra una finestra temporale di cinque giorni e di sei giorni aggiungendo l'informazione sulla domenica.

tempo	valore	iqr	gesd	pewma	sdewma	isoforest	score	anomaly
24/01/2022 00:06	-0,547897692	1	1	0	0	1	0.6	1
24/01/2022 00:07	-0,558663383	1	1	0	0	1	0.6	1
24/01/2022 00:08	-0,557382825	1	1	0	0	1	0.6	1
24/01/2022 00:09	-0,549223285	1	1	0	0	1	0.6	1

Tabella 5.8 : *Consistent Read Gets Per Txn* : risultati di *Anomaly Detection* dei 5 metodi, oggetto di analisi e del modello integrato. La tabella mostra i risultati di alcune osservazioni ottenuti considerando una finestra di calcolo di 5 giorni, dal lunedì al venerdi.

tempo	valore	iqr	gesd	pewma	sdewma	isoforest	score	anomaly
24/01/2022 00:06	-0,547897692	0	0	1	0	1	0.4	0
24/01/2022 00:07	-0,558663383	0	1	0	0	1	0.4	0
24/01/2022 00:08	-0,557382825	0	0	0	0	1	0.2	0
24/01/2022 00:09	-0,549223285	0	0	0	0	1	0.2	0

Tabella 5.9 : *Consistent Read Gets Per Txn* : risultati di *Anomaly Detection* dei 5 metodi, oggetto di analisi e del modello integrato. La tabella mostra i risultati di alcune osservazioni ottenuti considerando una finestra di calcolo di 6 giorni, aggiungendo l'informazione sulla domenica; la domenica i valori assunti dalla metrica sono maggiori rispetto agli altri giorni della settimana.

I risultati mostrati nelle tabelle evidenziano che aggiungendo nella finestra di calcolo più osservazioni, nello specifico una giornata, ciò determina variazioni sul risultato. Anche nel caso in esame, come per la metrica *"Phisycal write per sec",* le principali differenze sono

relative ai metodi “iqr” e “gesd” che sfruttano il concetto di decomposizione per effettuare il rilevamento delle anomalie. Di conseguenza, l’ impatto è determinante anche sul risultato finale del modello integrato.

Coerentemente con quanto atteso, in tabella 5.8 si evince che valori intorno allo zero vengono classificati come anomali da tre algoritmi su cinque. Gli stessi valori, non risultano anomali se gli algoritmi processano anche dati caratterizzati da oscillazioni più ampie, come si evince dalla tabella 5.9 .

Infine, è stato analizzato l’algoritmo *Isolation Forest*, nominato come “isoforest”, nel progetto di tesi. Le analisi hanno mostrato che i risultati dell’algoritmo sono fortemente dipendenti dalla soglia di anomalia che viene impostata. Infatti, come esposto in sezione 2.4.1, il processo di *Anomaly Detection* con *l’Isolation Forest* è composto da due fasi principali:

1. Costruzione degli alberi di isolamento tramite un set di dati di *training.*
2. Assegnazione dell’“*Anomaly Score*” ai punti di un set di dati di *test,* processati dal modello precedentemente costruito, e determinazione dei punti anomali in base alla definizione di una soglia specifica del dominio applicativo.

Tanto più l’“*Anomaly Score*” è vicino a 1, tanto maggiore è la probabilità che l’osservazione sia un’anomalia perché facilmente isolabile. Considerando la metrica *“Global Cache Avarage Current Get Time*”, essa è caratterizzata da un unico picco e da valori oscillanti intorno allo zero. Il risultato ottenuto dall’Isolation Forest dipende dalla soglia di anomalia che viene settata. Infatti, impostando una soglia pari a 0,75, l’algoritmo “isoforest” non identifica il picco come un’anomalia, tuttavia abbassando la soglia e considerando un valore di 0,6, esso individua il picco. In tabella 5.10 si riportano i valori ottenuti dai quattro modelli di serie temporali e *l’Isolation forest* con una soglia di 0,6, mentre in tabella 5.11 si riportano i risultati ottenuti considerando una soglia pari a 0,75.

tempo	valore	iqr	gesd	pewma	sdewma	isoforest	score	anomaly
28/01/2022 22:59	-0,002	0	0	0	0	0	0	0
28/01/2022 23:00	281,40	1	1	1	1	1	1	1

Tabella 5.10 : *Global Cache Avarage Current Get Time :* risultati di *Anomaly Detection* dei 5 metodi, oggetto di analisi e del modello integrato. La tabella mostra i risultati di due osservazioni, il picco e il valore precedente, settando una soglia di 0,6 per l’”*Anomaly Score*” dell’*Isolation Forest*.

tempo	valore	iqr	gesd	pewma	sdewma	isoforest	score	anomaly
28/01/2022 22:59	-0,002	0	0	0	0	0	0	0
28/01/2022 23:00	281,40	1	1	1	1	0	0.8	1

Tabella 5. 11 : *Global Cache Avarage Current Get Time :* risultati di *Anomaly Detection* dei 5 metodi, oggetto di analisi e del modello integrato. La tabella mostra i risultati di due osservazioni, il picco e il valore precedente, settando una soglia di 0,75 per l'"*Anomaly Score*" dell'*Isolation Forest*.

Anche per la metrica *"PX downgraded 1 to 25% Per Sec*" il picco che assume un valore di 281,42 viene rilevato come un'anomalia settando una soglia di 0,6 e non con una più alta pari a 0,75.

Oltre alla criticità dell' *Isolation Forest* relativa alla soglia settata per determinare lo stato di anomalia o meno di un'osservazione, ne emerge un'altra. Infatti, le metriche di *performance* analizzate costituiscono delle serie temporali. Nel progetto di tesi si è scelto di effettuare *Anomaly Detection* considerando la tipologia di dato e dando, dunque, rilevanza ai modelli specifici di serie temporali. Tuttavia, si è scelto di affiancare tali metodi con un modello di *Machine Learning* che di per sé non sfrutta l'informazione sul tempo.

Per ovviare a tale problematica si è scelto di implementare un'analisi multivariata per *l'Isolation Forest*, come spiegato nella sezione 5.1.1. Dunque, è stata aggiunta l'informazione temporale considerando come *features* anche i minuti, le ore e il giorno della settimana. In questa sezione si mostrano alcune analisi effettuate per evidenziare le differenze tra un' *Isolation Forest* univariato e multivariato.

A tal proposito è stata analizzata la metrica "*PQ QC Session Count*" . Considerando un totale di 20180 osservazioni, dalle analisi è risultato che la non coincidenza, tra il metodo multivariato e univariato, è relativa esclusivamente a 466 osservazioni. Si nota, inoltre, che sulle 466 osservazioni non coincidenti, 310 sono state rilevate come anomale dal metodo multivariato e non anomale dall'univariato. Tuttavia, non si evince una differenza sostanziale tra i risultati ottenuti dai due metodi, in quanto coincidono al 97,69%.

In generale, non sono state rilevate differenze particolarmente significative dall'utilizzo di un approccio univariato e multivariato. Tuttavia, si ritiene necessario inserire l'informazione temporale, data la tipologia dei dati esaminati.

Infine, in tabella 5.12, si riportano alcuni risultati ottenuti per la metrica *"PGA Cache Hit %".* Anche per questa metrica, come per "*PQ QC Session Count*", si evince che il metodo

multivariato coglie più anomalie rispetto all'univariato. La tabella 5.12 mostra i risultati ottenuti mettendo a confronto il metodo dell'*Isolation forest* univariato e multivariato. Inoltre, nella colonna "raggruppamento giorno e ora" si riportano i valori della metrica raggruppati per il giorno e l'ora.

tempo	valore	isoforest	isoforest multivariato	raggruppamento giorno e ora
30/01/2022 17:00	-2,53	0	1	-0,026694566
30/01/2022 17:01	-2,53	0	1	-0,026694566
30/01/2022 18:00	-2,28	0	1	0,009860411
30/01/2022 18:01	-2,27	0	1	0,009860411
30/01/2022 19:00	-2,11	0	1	0,043969231
30/01/2022 19:01	-2,1	0	1	0,043969231
30/01/2022 20:00	-1,98	0	1	-0,105955814
30/01/2022 21:00	-1,79	0	1	-0,205345249

Tabella 5.12 : *PGA Cache Hit % :* risultati di *Anomaly Detection* di alcune osservazioni, mettendo a confronto il metodo dell'*Isolation forest* univariato e multivariato. Inoltre, nella colonna "raggruppamento giorno e ora" si riportano i valori della metrica raggruppati per il giorno e l'ora.

Le maggiore differenze sono relative a quelle osservazioni che appartengo al normale *range* di oscillazione dei dati. L'aggiunta dell'informazione temporale determina come anomalie anche quelle osservazioni, che di per sé non si discostano dagli altri dati ma che, in relazione ai valori assunti dalla metrica nell'orario e nel giorno della settimana specifico, risultano discordanti.

6 Conclusioni

Questo capitolo è dedicato alle conclusioni della tesi. Nello specifico, si riportano i risultati più rilevanti, scaturiti dalla ricerca. In aggiunta, si effettuano valutazioni critiche dei risultati ottenuti, in relazione agli obiettivi di tesi. Infine, nella sezione dedicata agli sviluppi futuri, si riportano raccomandazioni e suggerimenti per possibili ricerche future.

Lo studio proposto nella tesi si è inserito all'interno di un progetto aziendale. L'area di Mediamente Consulting srl, specializzata in infrastruttura tecnologia, effettua il monitoraggio di *targets* come *Databases* o i *listeners,* tramite la raccolta di informazioni sulle loro *performances.* Il monitoraggio viene effettuato attraverso l'ausilio di strumenti come *Oracle Enterprise Manager.* Il progetto di tesi si è inserito in tale contesto; infatti, il *dataset* da analizzare comprendeva 161 metriche di *performance*. Quest'ultime sono utilizzate per il monitoraggio dei *targets* Oracle, da parte dei dipendenti dell'azienda *Mediamente Consulting*.

L'obiettivo di tesi è nato in risposta alla necessità di rendere più efficiente il sistema di gestione degli allarmi. Dunque, la tesi si è posta l'obiettivo di fornire soluzioni per il rilevamento delle anomalie delle metriche di *Performance*, tramite l'implementazione di modelli di *Machine Learning*. Infatti, la presenza di ambienti dinamici richiede un nuovo approccio che sia proattivo e non più reattivo. In questo contesto, l'analisi predittiva e le tecniche di *machine learning* consentono di identificare automaticamente le anomalie, apportando numerosi vantaggi. Nel progetto di tesi si è scelto di utilizzare due differenti approcci; il primo si è basato sull'applicazione di cinque modelli di serie temporali, il secondo, invece, sull'implementazione di un algoritmo di *machine learning* non supervisionato che sfrutta gli alberi decisionali, quale l'*Isolation Forest.*

Inoltre, come obiettivo di tesi, vi era lo scopo di affrontare la tematica dell'*Anomaly Detection,* attraverso la ricerca di modelli specifici per dati caratterizzati da alte frequenze e generati in *real-time*. Infatti, per ogni metrica del *dataset*, i dati costituivano *time series* con granularità al minuto e lo *span* temporale ricoperto da ognuna di esse era di due mesi, con un totale di 79199 osservazioni registrate per ogni metrica. I risultati sperimentali, infatti, hanno condotto all'esclusione del metodo tradizionale proposto da *Chung Chen* e *Lon-Mu Liu*, basato sui modelli ARIMA, perché inefficiente dal punto di vista computazionale. Gli

altri modelli di serie temporale, ideati appositamente per essere implementati con dati ad alta frequenza, risultavano più efficienti dal punto di vista computazionale, coerentemente con quanto atteso.

Per condurre un'analisi più efficiente è stato effettuato un processo di preparazione dei dati. Nel progetto di tesi, la prima fase del processo di preparazione dei dati si è basata su un'operazione di pulizia, che ha condotto all'esclusione di 6 metriche perché caratterizzate da valori nulli, riducendo il *dataset* da 161 a 155 metriche. Successivamente è stato eseguito un processo di trasformazione dei dati (z-score). Il ridimensionamento del *dataset* è stato effettuato per agevolare il confronto tra le metriche, risultando utile, essenzialmente nella fase di esplorazione dello stesso.

Il lavoro di tesi si è basato sull'analisi di dati che possono essere definiti come *Big Data;* infatti, il *dataset* risultava caratterizzato da 155 metriche di *performance*, con 79199 *record* per ognuna di esse. La vastità del *dataset* ha richiesto un lavoro di esplorazione significativo e oneroso. Per facilitare l'analisi, si è scelto di creare un pannello. Quest'ultimo ha consentito di visualizzare, per ogni metrica, i grafici dell'andamento della stessa, con la possibilità di settare il valore di prominenza e di individuare, conseguentemente, i picchi. Inoltre, il pannello mostrava, per ogni metrica, l'istogramma con la stima della densità di Kernel e il *boxplot*, con gli *outliers*. Come ulteriore analisi esplorativa, è stato creato un *report* contenente le statistiche descrittive delle metriche di *performance.* L'analisi esplorativa ha costituito una fase essenziale nel lavoro di tesi: gli strumenti grafici sono stati utilizzati sia nella fase di selezione delle *features* sia come metodo di validazione dei risultati degli algoritmi di *Anomaly Detection*. In seguito al lavoro svolto, si ritiene che, nel progetto di tesi, la fase esplorativa abbia rappresentato uno strumento utile, facilitando notevolmente il lavoro di analisi.

Come fase di preelaborazione dei dati, il lavori di tesi si è incentrato sulla selezione di un *subset* di *features,* tramite l'identificazione delle metriche di *performance* ridondanti e irrilevanti. Per identificare le *features* ridondanti è stata effettuata un' analisi di correlazione tra le 155 metriche di *performance* del *dataset*. Nel contesto specifico, rilevare le correlazioni tra le metriche ed effettuare *feature reduction*, significa ritenere che ci sia una forte sovrapposizione tra le anomalie individuate in una metrica e quelle rilevate nella metrica correlata. Per effettuare l'analisi si è scelto di adottare un approccio grafico; nello specifico, per determinare il coefficiente di correlazione ottimale, da considerare per ridurre

le *features,* sono stati condotti *test* di unità tramite valutazioni a coppie di metriche. Infine le evidenze grafiche hanno condotto alla scelta di un coefficiente di correlazione pari a 0,97, riducendo il *dataset* di 26 metriche (da 155 a 129). Per verificare la coincidenza delle anomalie è stato utilizzato il concetto di "picco". In seguito alle analisi effettuate si ritiene che la metodologia adottata presenta delle criticità legate al concetto di picco; tra queste il picco consente di cogliere esclusivamente le anomalie globali, e in alcuni casi un picco potrebbe non rappresentare un'anomalia. Inoltre la sovrapposizione quasi perfetta potrebbe non essere un indice di perfetta coincidenza nel rilevamento delle anomalie. Infine, il risultato, di coincidenza o meno tra le anomalie tra due metriche correlate, è fortemente dipendente anche dal tipo di algoritmo utilizzato. Per identificare le *features* irrilevanti ai fini dell' *Anomaly Detection* sono state effettuate analisi univariate, al fine di individuare metriche con andamenti costanti. A tale scopo è stato esplorato il *dataset* attraverso la caratterizzazione degli andamenti e delle distribuzioni delle metriche di *performance*. Inoltre, sono stati utilizzati anche altri strumenti come i *boxplot* e decomposizioni STL. Per agevolare il lavoro, data la numerosità del *dataset*, le metriche sono state ordinate in ordine decrescente di valore massimo. Tale metodologia ha facilitato il lavoro di selezione, consentendo di individuare facilmente le metriche con *range* di oscillazione più piccoli. Tale fase di analisi ha condotto all'esclusione di altre 7 metriche (da 129 a 122).

Nel progetto di tesi è stato proposto un modello che ha integrato tutti i metodi di *Anomaly Detection* implementati. Il concetto sul quale si è fondato il modello integrato è il cosiddeto "*Majority Voting*"; se la maggior parte degli algoritmi classificano un'osservazione come anomala, il modello integrato assegna lo stato di anomalia a tale osservazione, appunto per maggioranza di voti. Il metodo ha assunto un ruolo determinante laddove si è rilevata una discordanza dei risultati ottenuti dai diversi algoritmi. Le analisi hanno mostrato che, nel caso di metriche caratterizzate da andamenti costanti e da un numero di picchi limitato, nella maggior parte dei casi, tutti gli algoritmi erano in grado di identificare correttamente come anomali quei valori al di fuori del normale *range* di oscillazione dei dati (anomalie globali). Le principali criticità sono emerse sia nelle metriche caratterizzate da più oscillazioni, in questo caso alcuni picchi venivano rilevati come anomali solo da alcuni metodi, sia nel caso delle anomalie contestuali. Data l'assenza di etichettatura, circa il reale stato di anomalia di un'osservazione, il metodo proposto si è posto l'obiettivo di migliorare il risultato finale, sfruttando la potenza di più algoritmi. Le analisi dei picchi hanno condotto alla validazione del metodo proposto, tramite una valutazione grafica. Tuttavia, specilamente per la

rilevazione delle anomalie contestuali, la validazione grafica risulta più complessa. La principale criticità del metodo risiede nel fatto che se un algoritmo fosse più performante, in relazione a specifiche caratteristiche intrinseche di una metrica, il suo risultato potrebbe essere annullato dal metodo basato sulla maggioranza di voti.

Al fine di rendere le analisi più efficienti, per effettuare il rilevamento delle anomalie, sono state utilizzate delle "*rolling windows"* invece che finestre definite come "*expanding windows"*. Tale scelta è stata dettata dalla numerosità del *dataset* analizzato. Le analisi hanno mostrato che l'ampiezza della finestra di calcolo impatta nel rilevamento delle anomalie. Le considerazioni effettuate circa i fattori stagionali caratterizzanti le metriche del *dataset* hanno condotto alla scelta d finestre mobili costituite da almeno due settimane di dati (20160 osservazioni).

Infine si è focalizzata l'attenzione sull'*Isolation Forest*. Quest'ultimo non prende in considerazione il fattore tempo, in quanto non rappresenta un modello di serie temporali. Si è scelto di procedere attraverso un processo di *feature engineering.* Per ogni tipologia di metrica sono state valutate le caratteristiche stagionali e sono state aggiunte tre *features* al *dataset:* il minuto (0\59), l'ora (0\23) e il giorno settimanale (0\6). I risultati non hanno mostrato una differenza sostanziale tra il metodo univariato e multivariato. Inoltre ,nonostante il metodo non sia un modello di serie temporali, esso non si discosta maggiormente dai risultati ottenuti dalle altre metodologie. Infine i risultati *dell'Isolation Forest* sono fortemente dipendenti dalla *threshold* settata per "l'*anomaly score".*

6.1 Sviluppi futuri

Le potenzialità di approfondimento sono molteplici. Inerentemente all'analisi effettuata nell'ambito della *Feature Selection*, nel progetto di tesi, è stato adottato un approccio basato sul concetto di correlazione. In una prospettiva futura, si ritiene che l'analisi di correlazione possa essere esaminata ulteriormente. Nello specifico, potrebbe essere approfondito il concetto di sovrapposizione delle anomalie tra metriche correlate. Nel progetto di tesi è stato utilizzato un approccio grafico, mediamente la sovrapposizione di coppie di metriche e la valutazione dei picchi. In un'ottica futura si potrebbe valutare la coincidenza delle anomalie tramite l'applicazione di modelli sulle metriche correlate. Tale analisi potrebbe essere molto utile per agevolare il lavoro dei dipendenti dell'area infrastruttura dell'azienda Mediamente

Consulting. Quest'ultimi utilizzano un pannello che consente di visualizzare gli allarmi relativi alle metriche di *performance.* L'analisi effettuata sulle correlazioni nella tesi e, eventualmente, le analisi più approfondite sopradescritte, potrebbero essere integrate nel pannello, tramite l'aggiunta delle informazioni sulle metriche correlate. Il dipendente visualizzando un allarme relativo a una metrica, potrebbe aver bisogno delle informazioni relative alle metriche correlate per poter agire tempestivamente. Inoltre, il processo di selezione delle *features*, potrebbe essere esaminato ulteriormente, sperimentando nuove metodologie di selezione delle stesse. Il modello integrato, proposto nel progetto di tesi, potrebbe essere utilizzato come metodo per determinare delle etichette, circa lo stato di anomalia o meno di un'osservazione. Tale processo di etichettatura consentirebbe di poter sfruttare modelli supervisionati che possono essere confrontati più facilmente utilizzando, per esempio, il concetto di accuratezza. Infine, potrebbero essere sperimentati nuovi modelli per il rilevamento delle anomalie e nuovi metodi integrati per combinare i differenti modelli.

Elenco delle figure

.

Elenco delle tabelle

Bibliografia

Bandara k., Hyndman R., Bergmeir C., MSTL: A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns, Luglio 2021, [Online] https://arxiv.org/abs/2107.13462.

Carter K., Streilein W., Probabilistic Reasoning for Streaming Anomaly Detection, Agosto 2012 [Online] https://ieeexplore.ieee.org/document/6319708.

CFI, Exponentially Weighted Moving Average (EWMA), Ottobre 2018 [Online] https://corporatefinanceinstitute.com/resources/equities/exponentially-weighted-moving-average-ewma/.

Chen C., Liu L., Joint Estimation of Model Parameters and Outlier Effects in Time Series, Marzo 1993 [Online] https://www.istat.it/it/files/2014/06/Joint-Estimation-of-Model-Parameters-and-Outlier-Effects-in-Time-Series.pdf.

Cleveland R., Cleveland W., McRae J., Terpenning I., STL: A seasonal- Trend Decomposition Procedure Based on Loess, Marzo 1990, [Online] https://www.wessa.net/download/stl.pdf.

Computer World, Il Machine Learning spiegato bene: cos'è, come funziona e quali strumenti si usano, Dicembre 2019, [Online] https://www.cwi.it/tecnologie-emergenti/intelligenza-artificiale/machine-learning-124626.

Garg S., Algorithm selection for Anomaly Detection, Giugno 2020 [Online] https://medium.com/analytics-vidhya/algorithm-selection-for-anomaly-detection-ef193fd0d6d1.

Garziano G., Outliers Detection and Intervention Analysis, Dicembre 2017 [Online] https://www.r-bloggers.com/2017/12/outliers-detection-and-intervention-analysis/.

Hyndman R., Multiple Seasonal Decomposition [Online] https://pkg.robjhyndman.com/forecast/reference/mstl.html.

Liu F., Ting K., Isolation Forest, Dicembre 2008, [Online] https://ieeexplore.ieee.org/document/4781136.

Masarotto G., Analisi delle serie temporali (lucidi delle lezioni), Gennaio 2003 [Online] http://sirio.stat.unipd.it/files/ts03-04/ts03-04.pdf.

Mediamente Consulting, Gestione e analisi dei dati per le decisioni strategiche di business [Online] https://www.mediamenteconsulting.it/

Motadata, Oracle Database Monitoring [Online] https://www.motadata.com/it/oracle-database-monitoring/.

Oracle, Oracle Enterprise Manager [Online] https://www.oracle.com/enterprise-manager/.

PennState, Overview of Time Series Characteristics, [Online] https://online.stat.psu.edu/stat510/lesson/1/1.1.

Prabhakaran S., ARIMA Model – Complete Guide to Time Series Forecasting in Python, Agosto 2021, [Online] https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/#:~:text=An%20ARIMA%20model%20is%20one%20where%20the%20time,Combination%20of%20Lagged%20forecast%20errors%20%28upto%20q%20lags%29.

Raza H., Prasad G., Li Y., EWMA model based shift-detection methods for detecting covariate shifts in non-stationary environments, Dicembre 2013 [Online] https://www.researchgate.net/publication/266684854_EWMA_model_based_shift-detection_methods_for_detecting_covariate_shifts_in_non-stationary_environments.

Rezzani A., Big Data Architettura, tecnologie e metodi per l'utilizzo di grandi basi di dati, Ottobre 2013 [Online] https://books.google.it/books?hl=it&lr=&id=PYoFAgAAQBAJ&oi=fnd&pg=PR11&dq=big+data+articoli&ots=xFEUdPSMRf&sig=EIVd4Fv6F9g_NWECdGW5n18hryI#v=onepage&q=big%20data%20articoli&f=false.

R-Project, Anomalize Quick Start Guide, Ottobre 2020 [Online] https://cran.r-project.org/web/packages/anomalize/vignettes/anomalize_quick_start_guide.html.

R-Project, Anomalize: Tidy Anomaly Detection, [Online] https://cran.r-project.org/web/packages/anomalize/index.html

R-Project, Otsad: Online Time Series Anomaly Detectors https://cran.r-project.org/web/packages/otsad/index.html.

R-Project, Package "Tsoutliers", Ottobre 2014 [Online], https://cran.r-project.org/web/packages/tsoutliers/tsoutliers.pdf.

Stock J., Watson M., Introduzione All'econometria, Settembre 2016.

Talend, Che cosa significa preparazione dei dati?, [Online] https://www.talend.com/it/resources/what-is-data-preparation/.

Tremolada L, Machine learning, deep learning e reti neurali. Ecco di cosa parliamo, Gennaio 2019 [Online] https://www.ilsole24ore.com/art/machine-learning-deep-learning-e-reti-neurali-ecco-cosa-parliamo--AEaToEBH?refresh_ce=1.

Vallis O., Hochenbaum J., Keyariwal A., A Novel Technique for Long-Term Anomaly Detection in the Cloud, Giugno 2014, [Online] https://www.usenix.org/system/files/conference/hotcloud14/hotcloud14-vallis.pdf.