POLITECNICO DI TORINO

Collegio di Ingegneria Gestionale

Corso di Laurea Magistrale in Ingegneria Gestionale

Tesi di Laurea Magistrale

Machine learning e analisi dati per l'Anomaly detection applicata a database Oracle

mediaMENTE consulting

Relatore

Prof. Daniele Apiletti

Candidata

Ilenia Giunta

A.A. 2021/2022

Sommario

Indice delle figure

Indice delle tabelle

1. Introduzione

Questa tesi ha l'obiettivo di individuare anomalie di una base dati attraverso modelli di tipo non supervisionato partendo da un dataset di indicatori facenti riferimento alla performance di un database. Il lavoro è stato svolto all'interno dell'azienda di consulenza informatica Mediamente Consulting srl, in particolare nella business unit di Infrastruttura Tecnologica, nella quale sono create, gestite e monitorate strutture informatiche.

Al giorno d'oggi l'anomaly detection ricopre un vasto ambito di ricerca del machine learning, viste le numerose possibilità di applicazione in settori come, ad esempio, biologia, prevenzione di frodi, riconoscimento oggetti, analisi di immagini. Il progetto si basa sull'idea di utilizzare modelli di machine learning in supporto agli odierni metodi di monitoraggio di un database; il percorso, intrapreso già da altri lavori di tesi sviluppati nella stessa azienda, procede cercando di trovare dei modelli in grado di distinguere gli istanti in cui la base dati non è performante.

È stato seguito il processo del KDD (Knowledge Discovery in Databases) che, attraverso le fasi di data cleaning, data integration, data transformation, data mining, pattern evaluation e knowledge presentation, è in grado di avere dei risultati consistenti. In questo caso studio, si è partiti dalla creazione di un dataset in un ambiente di sviluppo interno mediante Swingbench, un tool di simulazione di attività impattanti su un database e successivamente analizzato utilizzando librerie e tool del linguaggio di programmazione Python.

A Swingbench - e i benchmarks utilizzati al suo interno quali Sales History, Order Entry e TPCDS – è stato affiancato in parallelo un test personalizzato allo scopo di aumentare il carico del database. Seguendo un approccio esplorativo, sono stati analizzati i valori del dataset, depurati e trasformati per essere più significativi con il fine di estrarre conoscenza e informazione dall'applicazione di modelli di tipo non supervisionato. L'approccio non supervisionato in particolare permette di trovare anche pattern non visibili senza l'intervento umano, da sfruttare successivamente per creare o migliorare degli strumenti più mirati al problema preso in esame.

2. Azienda

Il progetto di tesi si è svolto presso la società Mediamente Consulting s.r.l., fondata nel 2013 all'interno dell'I3P, Incubatore Imprese Innovative Politecnico Torino, che attualmente conta più di cinquanta dipendenti.

Mediamente Consulting è una società di consulenza informatica specializzata nelle tecnologie di business analytics avanzate. Si occupa della gestione dei dati e della loro analisi, in particolare partendo dall'infrastruttura fino a coprire all'elaborazione di soluzioni tailored per le aziende.

L'azienda è organizzata in business units, ciascuna corrispondente alle principali aree di competenza, infrastruttura tecnologica, data integration and management, Corporate performance management, advanced analytics e business intelligence.

La società conta tra i suoi clienti imprese medio grande calibro di diversi settori a partire dall'Healthcare, Manufacturing, Retail, Fashion, dall'Automotive al Food & Beverage, Banking e E-Commerce.

La tesi è stata svolta nella business unit di Infrastruttura Tecnologica, la quale fornisce ai propri clienti soluzioni di gestione e controllo dell'infrastruttura tecnologica e lato application.

2.1 Il contesto

Il presente lavoro tenta di individuare dei modelli e delle metodologie nell'ambito del machine learning che rilevino scenari in cui si verificano delle anomalie.

Il progetto di tesi è stato svolto in collaborazione e stretto contatto con la business unit di infrastruttura tecnologica che, come anticipato nella descrizione dell'azienda, si occupa non solo della ideazione dell'infrastruttura che sorregge tutta la base dati, ma anche della sua gestione e del suo monitoraggio. La sezione che si occupa di questi ultimi due step opera in un contesto di intervento rapido allor quando si presenti una qualsiasi anomalia che turbi lo status quo del sistema informatico, risolvendo il problema. Questa tesi fa parte di un più ampio progetto di anomaly detection; aggiunge un tassello a ciò che è stato già avviato in progetti di tesi precedenti.

Prima che sia definito il problema, è stato analizzato l'attuale status del processo di detection di un'anomalia nei sistemi informatici. Come già detto, ci si accorge del presentarsi di un'anomalia soltanto ex post, ma questo può accadere in diverse modalità. In primis, può essere segnalata da un cliente sia per evidenti crash del sistema, nei casi più eclatanti, che per rallentamenti o blocchi temporanei. In secondo luogo, l'anomalia può essere rilevata perché è stato piazzato un alert su alcuni indicatori i cui valori sono noti agli esperti di domino e che, superando un valore soglia minimo o massimo, segnalano un malfunzionamento.

2.2 Il problema

Questo capitolo esplica più ampiamente e dettagliatamente il caso tenuto in considerazione.

Questo progetto fa parte di uno più ampio in progress della società: realizzare uno strumento di supporto alla business unit in grado di monitorare lo status dei sistemi informatici in uso, di rilevare in tempo reale eventi anomali e/o allertare gli esperti di dominio in via preventiva.

In tale contesto ci si riferisce al problema come sinonimo intercambiabile di anomalia, definita come il presentarsi di un comportamento da parte del sistema che potrebbe comprometterne il normale funzionamento, alterando le performance e, incorrendo, con sufficiente probabilità, ad un blocco delle attività correnti/di routine. È, tuttavia, complicato risalire alle cause dell'anomalia, limitando il contributo degli esperti di dominio ex post l'avvenuto rilevamento del problema.

In questo progetto sono state operate delle semplificazioni, così da permettere un focus mirato su un aspetto particolare.

Si è scelto di sfruttare un ambiente di sviluppo interno, più controllato rispetto a un qualsiasi contesto reale, garantendo la possibilità di creare un numero svariato di test, con diversi livelli di approssimazione a casi reali. I dati raccolti fanno, quindi, riferimento alle performance di normale attività di un'istanza Oracle.

2.3 Workflow del caso studio

Il lavoro è stato condotto seguendo i passaggi mostrati nella figura sottostante iniziando dall'apprendimento e approfondimento degli strumenti, tra cui software necessari come Swingbench e Python. Successivamente sono state impostate le modalità di raccolta dei dati, il livello e il tipo di stress a cui il database doveva essere sottoposto. Portata a termine la data

collection, i dati sono stati visualizzati per poi essere trasformati al fine di permettere agli algoritmi di essere più efficaci. Una volta selezionati e applicati i modelli, è stato effettuato un confronto basato sulla quantità e qualità di datapoint ritenuti essere anomalie e, quindi, corrispondenti ad istanti di stress. Infine, i risultati sono stati presentati in forma grafica per agevolarne la comprensione e facilitare il proseguimento della ricerca.

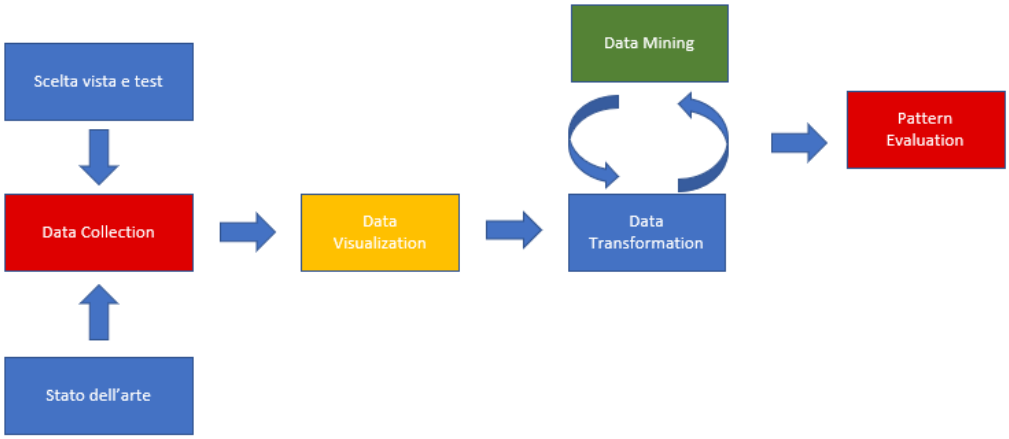

Figura 1 Workflow del processo

2. Stato dell'arte

Il Machine Learning (ML) è un campo dell'intelligenza artificiale nel quale i computer apprendono dai dati, solitamente per migliorare la propria performance su task ben definiti, senza essere stati esplicitamente programmati. Il termine Machine Learning è stato coniato nel 1959 da Arthur Samuel, ma in quel periodo il ML restava una nicchia di ricerca per gli accademici. Il machine learning è stato parte dell'evoluzione dell'Intelligenza Artificiale (AI) fino alla fine degli anni '70; successivamente si sviluppò in autonomia espandendosi dall'eCollerce fino a settori come biologia, medicina, frode bancaria, monitoraggio prestazioni.

2.1 Anomaly detection

Nell'ambito del machine learning, è sempre stato molto interessante l'individuare istanze "insolite" nei record di un dataset. Questo processo è comunemente chiamato anomaly detection o outliers detection; la prima definizione risale al 1969, probabilmente data da Grubbs come <<*An outlying observation, or* outlier*, is one that appears to deviate markedly from other members of the sample in which it occurs*>>. Sebbene questa definizione sia ancora valida, la motivazione che porta al riconoscimento degli outliers è oggi molto diversa. Inizialmente, il motivo principale per il rilevamento era la rimozione dei valori erratici dai dati dell'allenamento, poiché gli algoritmi di identificazione del modello erano molto sensibili a tali valori. Il processo precedentemente menzionato è anche denominato data cleansing. Una volta sviluppata una classificazione più robusta, l'interesse per l'individuazione delle irregolarità è notevolmente diminuito. C'è stata una svolta nel 2000, quando i ricercatori hanno iniziato a concentrarsi maggiormente sulle anomalie stesse, perché spesso sono legate a particolari circostanze interessanti o a records discutibili. In tale contesto è stata ampliata anche la definizione di Grubbs per cui è ora noto che l'anomalia presenta due caratteristiche importanti: le anomalie differiscono dallo standard per quanto riguarda le loro proprietà e sono rari nel set di dati rispetto ai casi normali.

L'anomaly detection è uno degli scopi per cui il ML è usato: identificazione degli elementi imprevisti in dataset riconosciuti come fuori norma. A differenza degli obiettivi di classificazione standard, il rilevamento di anomalie si applica spesso ai dati non etichettati, tenendo conto unicamente della struttura/composizione interna del set di dati. Questa branca è conosciuta come unsupervised anomaly detection - rilevamento non controllato di

anomalie – ed è utilizzata in numerose applicazioni pratiche, quali il rilevamento di interferenze nelle reti, l'individuazione di frodi e il rilevamento di tali anomalie, nonché nelle scienze della vita e in medicina. Sono stati proposti decine di algoritmi in questo settore con i rispettivi punti di forza e di debolezza, la capacità di rilevare errori, l'impatto dei parametri e la capacità di rilevamento di anomalie globali e locali.

In molti campi d'applicazione si stanno utilizzando gli algoritmi di anomaly detection e spesso migliorano i tradizionali sistemi di rilevazione regolati come intrusion detection, fraud detection, data leackage prevention (DLP), medical application.

L'intrusion detection è probabilmente l'applicazione più rilevante ai fini dell'anomaly detection. Questo scenario applicativo monitora il traffico di rete e le applicazioni server.

Si possono impiegare diversi approcci per il data mining, la quale è una branca della data science che si occupa dell'individuazione di caratteristiche o proprietà dei dati in un set; quindi, si possono avere apprendimenti di tipo supervisionato, semi-supervisionato o non supervisionato.

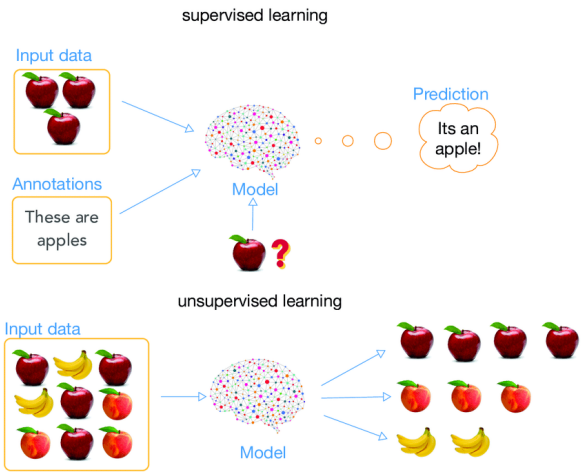

Figura 2 Supervised vs. Unsupervised Learning [1]

a. SUPERVISIONATO.

   L'apprendimento supervisionato eccelle nell'ottimizzazione delle performance in task ben definiti in cui sono presenti le etichette. Dopo aver allenato un algoritmo con i dati, si sarà in grado di misurare la propria performance (attraverso una funzione di costo) confrontando le etichette predette con quelle presenti nei file. Le etichette sono potenti, danno al data analyst una misura di errore, utile per migliorare la performance del modello in fasi successive. Di contro, l'operazione di labeling è time

consuming e i sistemi di apprendimento supervisionato hanno delle limitazioni nella generalizzazione e nell'applicare la conoscenza acquisita in dataset su cui non sono stati allenti. In sintesi, l'apprendimento supervisionato è performante su problemi locali, ma non per la risoluzione di problemi più ambiziosi o meno definiti.

b. SEMI-SUPERVISIONATO.

L'apprendimento semi supervisionato combina, durante la fase di training, una grande quantità di dati non etichettati con una piccola quantità di dati etichettati. È considerato essere, per questo motivo, un approccio intermedio tra supervisionato e non supervisionato. Se in presenza di un dataset misto, nessuno degli approcci citati precedentemente sarà in grado di performare correttamente. Confrontandolo con l'approccio supervisionato che usa solo dati etichettati, si ha una predizione più accurata in quanto si tengono in considerazione anche di dati non etichettati.

c. NON SUPERVISIONATO.

Contrariamente all'apprendimento supervisionato, il non supervisionato le labels non sono disponibili. Quindi, la performance non può essere misurata in modo chiaro e il data analyst non ha un task ben definito; tuttavia, la soluzione, se gestita, risulta più potente.

L'apprendimento non supervisionato apprende, infatti, la struttura sottostante dei dati su cui ha eseguito il training; si è in grado di identificare pattern distinti nel dataset. Rende anche problemi più risolvibili e migliora anche la generalizzazione della conoscenza. L'apprendimento non supervisionato rende problemi precedentemente non affrontabili più facili da risolvere e rende più visibili dei pattern all'interno dei dati altrimenti nascosti. Infine, con l'apprendimento non supervisionato la conoscenza in output può essere generalizzata.

Alcune applicazioni degli algoritmi di unsupervised machine learning sono: segmentazione dei dati, analisi di mercato, rilevazioni di anomalie, frodi e attacchi e duplicazione testi.

Come mostrato in figura, esistono svariate categorie di algoritmi utilizzabili per approcci non supervisionati. Nel presente lavoro di tesi ne sono stati selezionati alcuni che secondo gli studi si sono rivelati essere efficaci per l'outlier detection.

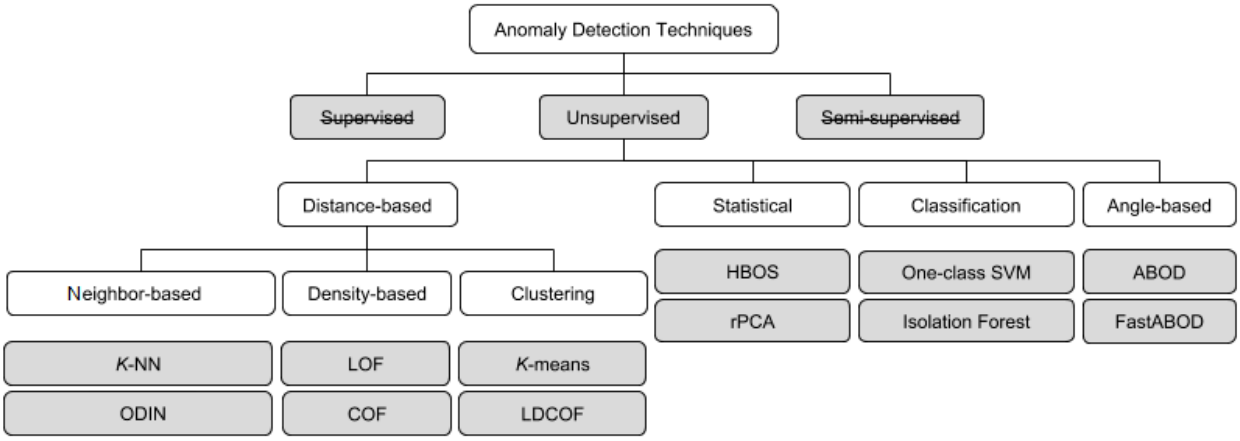

Figura 3 Tecniche di Anomaly Detection Non supervisionate

Un altro tipo di categorizzazione possibile è quella in approccio univariato, bivariato e multivariato.

- Oltre al tipo di approccio, le analisi si possono dividere in univariate, bivariate e multivariate. L'analisi di tipo univariato è la più semplice forma di analisi dato che l'informazione è contenuta in una sola variabile che assume valori diversi. Non riguarda cause ed effetti o legami con altre variabili; il principale obiettivo dell'analisi è quello di descrivere i dati e trovare dei pattern al loro interno. Le analisi univariate più comuni sono il controllo della tendenza, del range e la deviazione standard.
- Le analisi bivariate permettono di confrontare due variabili per approfondire la loro relazione; le variabili potrebbero essere dipendenti o indipendenti l'una dall'altra.
- Si parla di analisi multivariate quando coinvolgono tre o più variabili; il tipo di analisi conducibile su questo tipo di dati dipende dall'obiettivo che si vuole raggiungere. Negli algoritmi di tipo multivariato le variabili sono modellate congiuntamente, in modo da avere un risultato e una visione globale più compatibile con scenari realistici.

2.2 Gli algoritmi unsupervised

Gli algoritmi non supervisionati [12] riescono a restituire dei pattern che potrebbero rivelarsi inaspettati, a differenza dei supervisionati in cui le etichette sono state prestabilite ex ante. Gli algoritmi unsupervised si prestano anche a dataset di grandi dimensioni, in quanto

procedere al labeling risulterebbe time consuming oltre che in termini di effort. Saranno di seguito spiegati il funzionamento, gli assunti e i punti di forza di Isolation Forest.

Isolation forest

Isolation Forest, - anche detto iForest - propone un diverso tipo di metodo model-based che isola esplicitamente le anomalie anziché profilare le istanze considerate normali. [10] L'algoritmo si basa su due caratteristiche quantitative delle anomalie: sono presenti in minoranza rispetto ai datapoint del dataset e hanno valori molto differenti dalle normali istanze. Queste caratteristiche le rendono, infine, più individuabili e la struttura ad albero si rivela efficace nell'isolare le istanze anomale.
Le anomalie tendono a isolarsi nelle aree più vicine alla radice, mentre le istanze normali tendono a isolarsi nelle aree più lontane. [10]

Isolation forest è un algoritmo di tipo non supervisionato che usa gli alberi decisionali (decision trees) allo scopo di rilevare i valori anomali nel dataset in input. Un decision tree è un algoritmo di apprendimento supervisionato non parametrico, usato sia per la classificazione che per la regressione dei dati. Presenta una struttura ad albero, gerarchica che consiste in un nodo radice, dei rami, nodi interni ed esterni, come mostrato in Figura

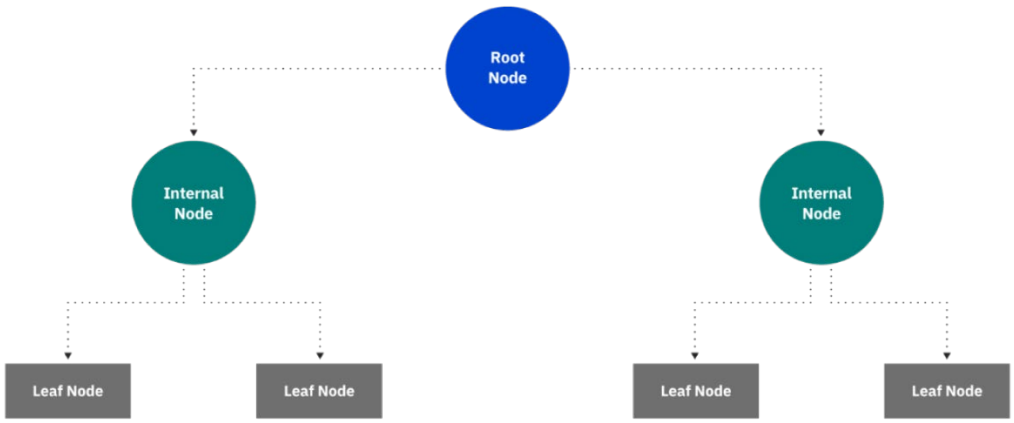

Figura 4 Schema Decision Tree

Il processo di splitting, che parte dal nodo radice (Root Node), è di tipo top-down e ricorsivo lungo tutta l'estensione dell'albero decisionale generato; dalla separazione si creano sempre

due subset di dati, data la condizione espressa alla loro radice, finché non si raggiunge uno stato di fine in cui la maggior parte dei datapoint è stata classificata sotto una specifica etichetta e i nodi foglia hanno anche superato un valore determinato o l'albero ha raggiunto la sua massima estensione. I decision trees hanno come punto di debolezza il rischio di overfitting; questo è però bypassato dall'applicazione dell'algoritmo Random Forest, il quale, tramite la suddivisione iniziale in più subset, restituisce risultati più accurati, specie in presenza di dati non correlati. La classificazione avviene in fase finale eseguendo un majority voting sui singoli trees.

Le anomalie tendono, pertanto, a isolarsi nelle aree più vicine alla radice e che quindi hanno un path mediamente più breve, mentre le istanze normali tendono a isolarsi nelle aree più lontane, come mostrato in Figura

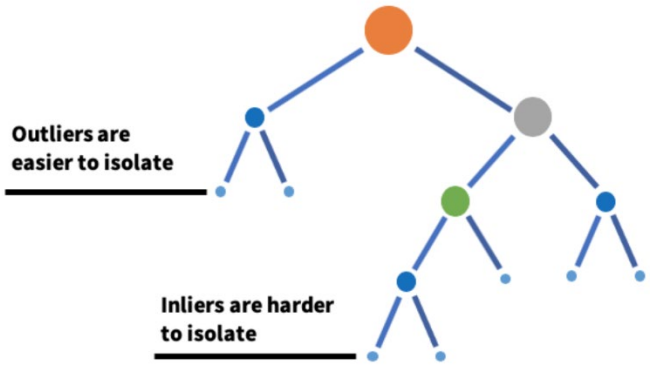

Figura 5 Outliers in Isolation Forest

Isolation forest è in grado di rilevare le anomalie più velocemente e con un minor numero di alberi, dimostrandosi efficiente anche in presenza di dataset con elevata dimensionalità. iForest si distingue dai metodi di altro tipo come model-based, distance based e density-based grazie alle seguenti caratteristiche:

- Complessità computazionale lineare (O(n))
- Capacità di scale up in presenza di dataset di alta dimensionalità con un elevato numero di attributi irrilevanti
- Non si serve di misure di densità o distanza per individuare anomalie, eliminando i maggiori costi di computazione;

- Gli Isolation Trees risolvono eventuali effetti di *swamping*, i.e. la classificazione punti normali come anomali, e il *masking*, i.e. la presenza di un eccessivo numero di punti anomali che l'algoritmo di Isolation Forest non riesce a rilevare facilmente.

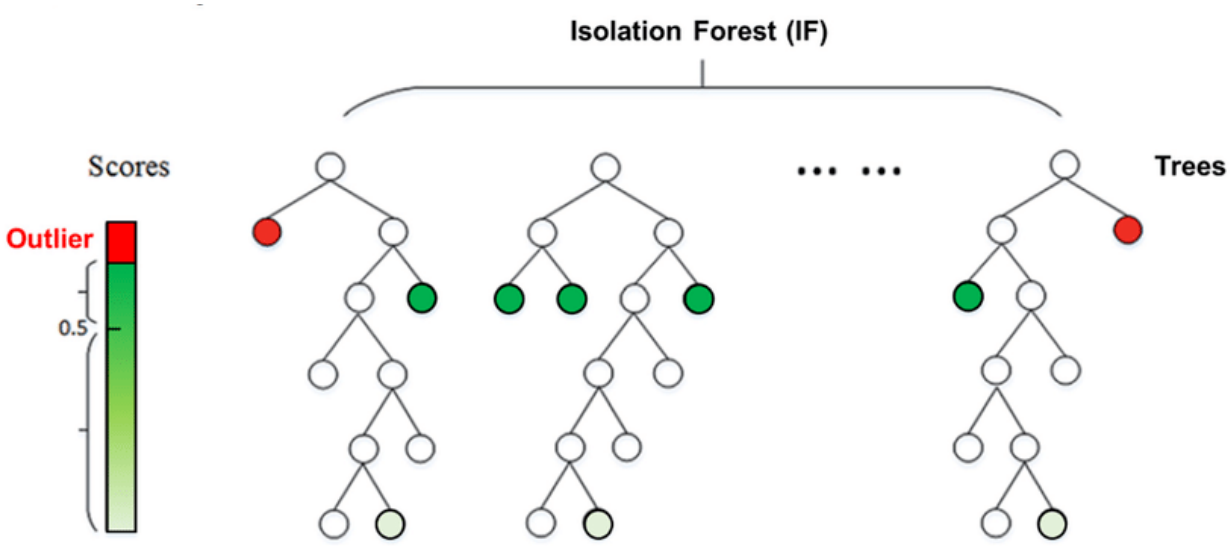

Figura 6 Selezione punti anomali con soglia Isolation Forest

Un iTree è un *proper binary tree* in cui ciascun nodo ha esattamente nessuno o due nodi foglia. Come tutti i metodi di anomaly detection è richiesto un anomaly score *s* calcolato come segue $s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$, dove

- E(h(x)) è il valore medio di h(x) di un gruppo di isolation trees
- h(x) è la lunghezza del path di un datapoint x
- c(n) è il valore medio di h(x), dato n, la cui formula è $c(n) = 2H(n-1) - (\frac{2(n-1)}{n})$, dove H(i) corrisponde al numero armonico stimabile come ln(i) + 0,5772156649 (Costante di Eulero).

Ne viene assegnato uno a ogni datapoint nell'intervallo $0 \leq s \leq 1$ in modo tale da poter classificare le anomalie facendo un ranking in ordine decrescente e scremando sulla base di una soglia k, al di sopra della quale le istanze sono considerate anomale. Altra modalità di assessment è la seguente:

- s prossimo a 1, le istanze sono anomale;

- s inferiore a 0,5 le istanze sono considerate come normali;
- s prossimo a 0,5 per tutte le istanze indica che l'intero campione non presenta anomalie.

3. Tecnologie utilizzate

3.1 Python

Python è un linguaggio di programmazione orientato a oggetti, di alto livello, molto versatile: può essere, infatti, utilizzato per sviluppare applicazioni, creare script o fare test di sistema. Una sua peculiarità è il costrutto del linguaggio, molto leggibile anche grazie alla indentazione per la sintassi delle specifiche, sostituta strategica delle parentesi. Python possiede una libreria standard molto ampia, uno dei punti di forza, questa peculiarità garantisce una vasta versatilità. Inoltre, possono essere aggiunti altri scritti in C o Python i moduli già presenti in libreria.

In particolare, in questo lavoro di tesi, è stato utilizzato per la elaborazione della fase di data preprocessing e nella fase di data visualization nella versione 3.8.8 con l'impiego delle librerie elencate di seguito:

- Pandas: è una libreria open-source creata principalmente per lavorare intuitivamente con i dati etichettati o dati relazionali. Fornisce svariate strutture dati e operazioni per manipolare serie temporali e dati numerici; altri vantaggi di Pandas sono la velocità, la produttività e performance per gli utenti, l'upload di molti formati di file e la facile manipolazione dei dataset come pivot, fusioni o reshaping [5];
- Numpy: è anch'essa una libreria open-source. Al suo interno troviamo molteplici funzioni matematiche e di algebra lineare [6];
- SciPy (inserire la versione): fornisce delle strutture dati e algoritmi ampiamente applicabili a vari domini. Alcuni di questi algoritmi servono per interpolazioni, integrazioni, equazioni differenziali e funzioni statistiche. Le sue estensioni sono degli strumenti ottimi per la manipolazione degli array, matrici e alteri multidimensionali [7];
- Sci-kit learn (inserire la versione): contiene degli strumenti altamente efficienti e semplici per l'analisi dei dati predittiva, costruito su Numpy e SciPy, è una libreria open-source [8];
- Os (inserire la versione): è un modulo di Python utile per far interagire il programma con il sistema operativo del computer e compiere diverse operazioni come, ad esempio, creare, rinominare un file o gestire i permessi di una cartella.

3.1 Swingbench (version 2.6.0.1173)

Swingbench è un generatore di carico libero (e benchmark) progettato per stressare un database Oracle nelle versioni 12c, 18c, 19c; è uno strumento non ufficiale di Oracle, ma un tool creato da Dominic Giles, un loro ex dipendente. [13] Swingbench è un tool molto versatile, può essere infatti utilizzato per dimostrare e testare tecnologie come RAC, backup e recovery online o ricostruzione di tabelle. Include di default sei benchmark: Order Entry, TPC-DS like, Sales History, JSON, Calling Circle e Stress Test [1]. Sono stati presi in considerazione solo i primi tre schema di cui segue una breve descrizione:

- Order Entry: si basa sullo schema oe, un sample schema di Oracle presente nelle tre versioni su cui Swingbench può lavorare. Si utilizza per stressare memoria e interconnessioni su tabelle di piccole dimensioni. Questo benchmark simula l'attività di un'azienda di retail di vari prodotti venduti in tutto il mondo memorizzati in una tabella, così come l'elenco dei clienti, le transazioni e lo storico degli acquisti.
- TPC-DS like: è un benchmark simile a TPC-DS. Simula un carico sia a livello di transazioni che di query, ma in file di configurazione separati. TPCDS simula un data warehouse, modella delle funzioni di supporto alle decisioni per un'attività di reati su vari canali: negozio fisico, e-commerce, catalogo. Si tiene traccia di negozi, clienti – demografica, storico degli acquisti, transazioni -, promozioni, prodotti.
- Sales History: come oe, è un sample schema di Oracle, usato per testare delle query complesse su un database contenente tabelle di grandi dimensioni. In questo caso si simula la creazione di statistiche necessarie alla società a supporto dell'iter decisionale. Nelle tabelli si trovano informazioni su prodotti, sconti o promo relazionate ai clienti e ai canali di cui hanno usufruito per beneficiarne, oltre che i dati demografici.

Questa utility dà l'opportunità di variare il tipo di test e, per aumentare la verosimiglianza a un business reale, si possono variare, ad esempio, il numero di utenti, durata del test – runtime -, tempo di attesa tra due test eseguiti consecutivamente, uguali o di tipo diverso, e, infine, le modalità di visualizzazione.

Swingbench permette anche la creazione di benchmark e test personalizzati, per assecondare tutte le esigenze di ambiente di test, che, in aggiunta ai test sopracitati, potrebbero risultare in uno scenario più realistico; possono essere utilizzati dei sample schema Oracle, crearne uno sulla base di schema in possesso degli utenti o generarne ex novo.

3.1.1 Sample Schema Oracle

I database sample schema forniscono una piattaforma comune e fanno parte di un set di database schema interconnessi: Human Resources (HR), Order Entry (OE), Online Catalog (OC), Product Media (PM), Information Exchange (IX), Sales History (SH), Customer Orders (CO) [2]. Questi schema sono stati creati seguendo i principi di:

- Semplicità e facilità d'uso: in particolare, per esempio, lo schema OE è stato progettato volutamente per essere semplice, e, inoltre, per dare un percorso graduale verso livelli di utilizzo intermedio del database;
- Rilevanza per gli user abituali: sia gli schema che le loro estensioni evidenziano le funzioni che gli utenti utilizzano maggiormente. L'intero set di schema mette a disposizione una base di partenza ampliabile con funzionalità aggiuntive;

Schema OE

Lo schema Order Entry (OE) ripropone l'attività di un'impresa che vende svariati prodotti come strumenti, musica, abbigliamento. Questo schema immagazzina informazioni sui prodotti come il codice identificativo (product_id), la categoria in cui rientra (category_id), l'inserimento dell'ordine (OE), il range di peso ai fini della spedizione (weight_class), periodo di garanzia (warranty_period), il fornitore (supplier_id), lo stato di disponibilità del prodotto(product_status), il prezzo di listino (list_price), un prezzo minimo a cui il prodotto deve essere venduto (min_price) e l'URL del catalogo in cui il prodotto è presente (catalog_url). I prodotti sono venduti a livello globale, pertanto i prodotti sono memorizzati in diverse lingue, così come anche le descrizioni.

L'azienda gestisce i magazzini dislocati in diverse località; ogni magazzino ha un numero identificativo (warehouse_id), nome (warehouse_name), il numero di identificazione della posizione (location_id) e la descrizione della struttura (warehouse_spec).

Sono monitorate le informazioni dei clienti tramite codice identificativo (customer_id) oltre che le generalità (nome. cognome, data di nascita, numero di telefono, e-mail, indirizzo, codice postale); il database memorizza anche l'area geografica di appartenenza e la lingua madre. I clienti hanno un limite massimo di credito per limitare la quantità di prodotti acquistati in un singolo ordine; è possibile effettuare un upgrade a profilo manager.

Nel momento in cui un cliente effettua un ordine, la società tiene traccia dello stesso (order_id), la data in cui è stato effettuato (order_date), dello stato (order_status), della modalità di spedizione (order_mode) dell'importo totale (order_total) e dell'agente che lo ha concluso. Inoltre, per ogni ordine si tiene traccia degli articoli contenuti e delle quantità ordinate, oltre che il prezzo.

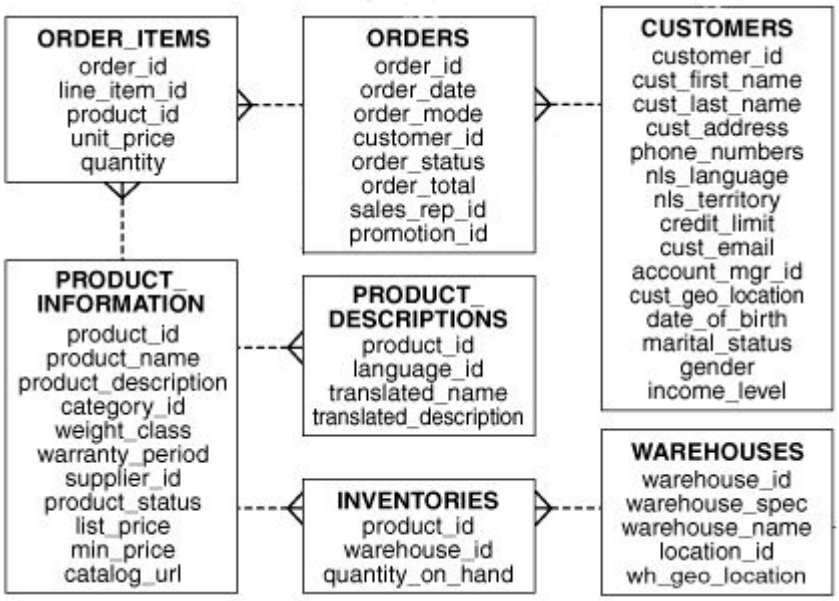

Figura 7 Schema Order Entry

Schema SH

La società, il cui schema è rappresentato da Sales History (SH), simula un alto volume di operazioni e transazioni di business e stila delle statistiche da cui sono redatti dei report per aiutare l'organizzazione nel decision making strategico. I dati analizzati sono i trend raccolti nei periodi precedenti, la società carica regolarmente i dati nei propri data warehouse per raccogliere le statistiche fondamentali per i report. Questi report contengono info e prospetti delle vendite in periodi annuali, quadrimestrali, mensili e settimanali per ogni prodotto.

La società raccoglie dati per stilare dei report sui canali di distribuzione attraverso cui le vendite sono portate a termine. Quando la società esegue promozioni sui prodotti, è analizzato anche l'impatto delle promozioni sulle vendite. Sono anche esaminate le vendite relative alle diverse aree geografiche in cui l'azienda opera.

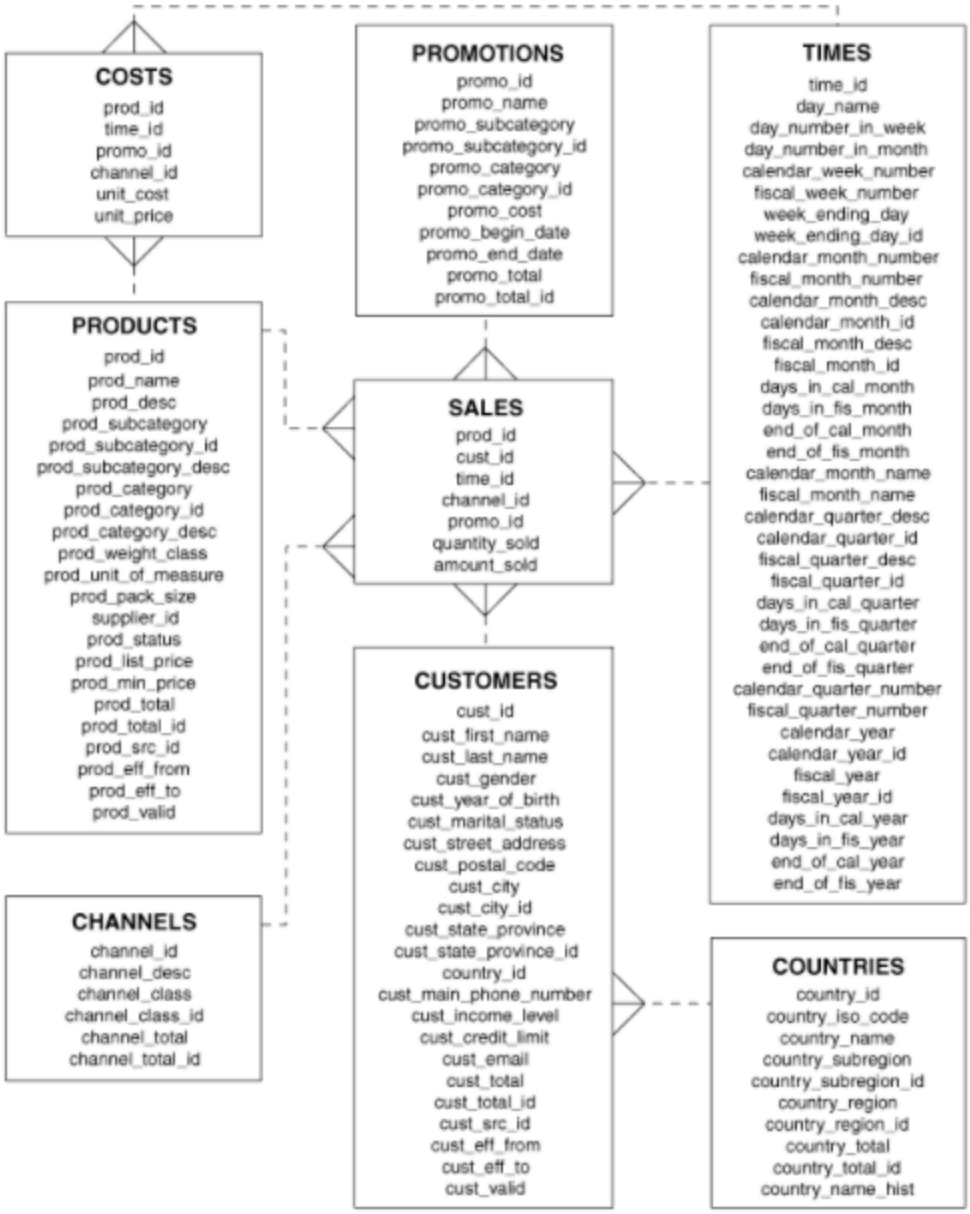

Figura 8 Schema Sales History

Schema TPCDS- like

Questo schema ricalca, come suggerisce anche il nome, ha una struttura simile al TPCDS, uno dei diversi tipi appartenenti al gruppo dei benchmark TPC. “The TPC Benchmark™DS (TPC-DS) is a decision support benchmark that models several generally applicable aspects of a decision support system, including queries and data maintenance. The benchmark provides a representative evaluation of the System Under Test’s (SUT) performance as a general purpose decision support system.” [3]; questo benchmark rientra tra quelli cosiddetti Decision Support System (DSS) e simula un carico di lavoro analitico e transazionale.

TPCDS modella un data warehouse e si focalizza su OLAP (OnLine Analytical Processing). Tale benchmark riproduce il traffico di un'attività di retail con tre canali di vendita: retail stores, cataloghi e on-line marketplace; è uno snowflake schema.

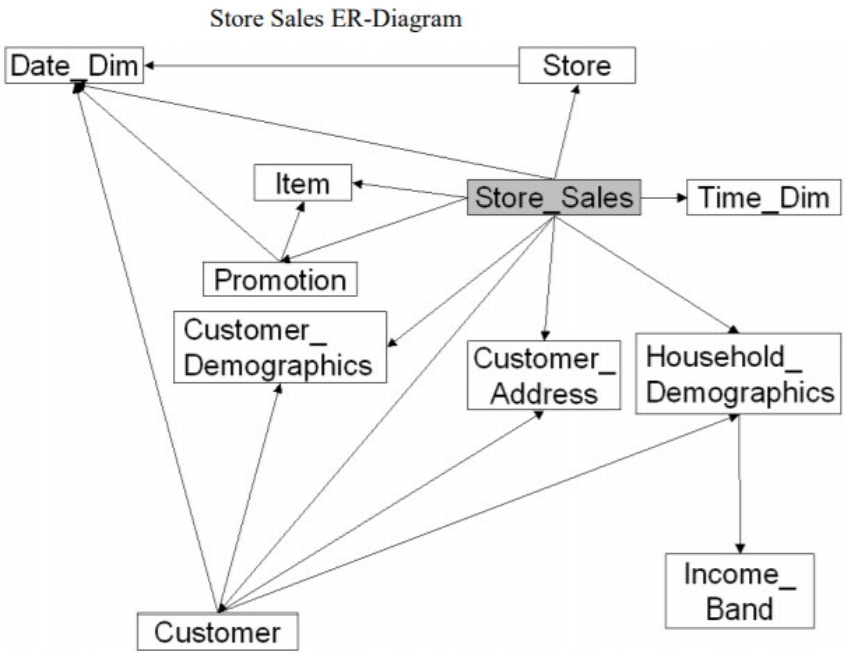

Figura 9 Schema ER di TPCDS : Store Sales

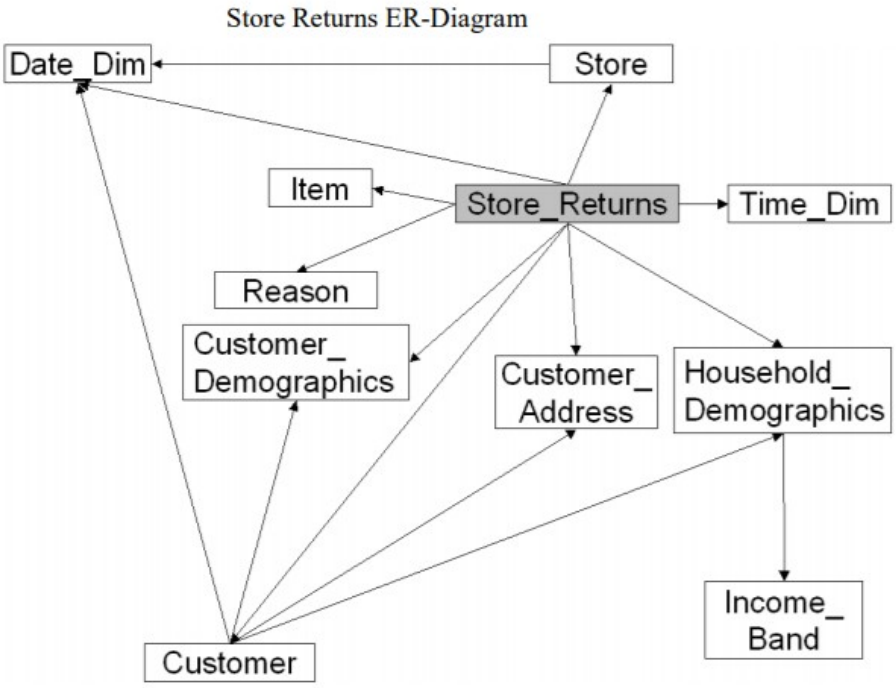

Figura 10 Schema ER di TPCDS, Store Returns

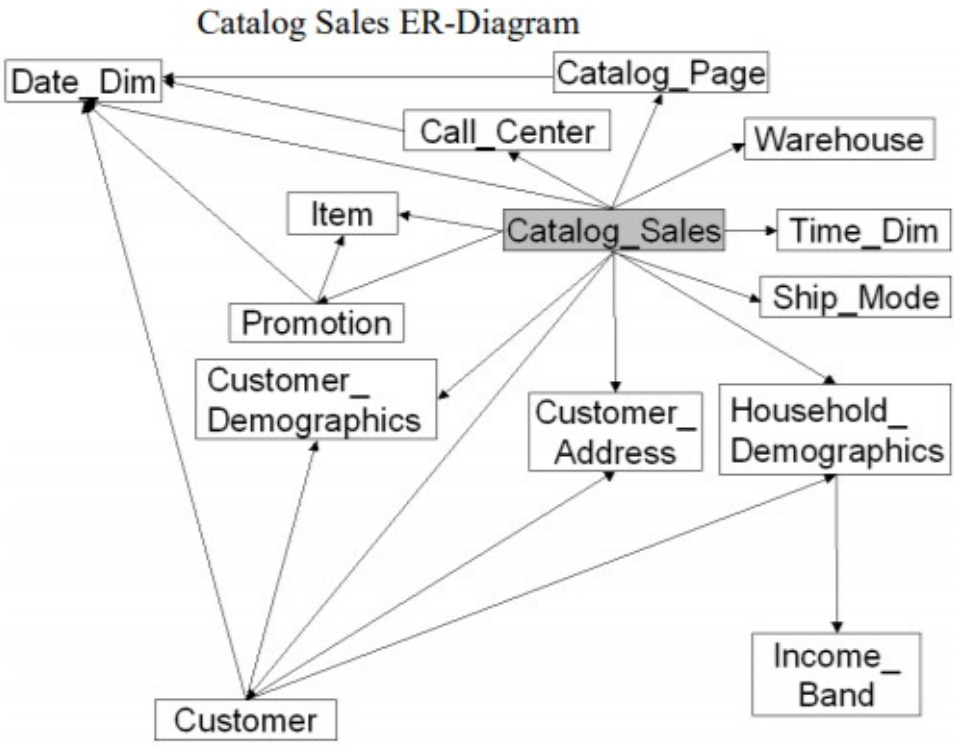

Figura 11 Schema ER di TPCDS : Catalog Sales

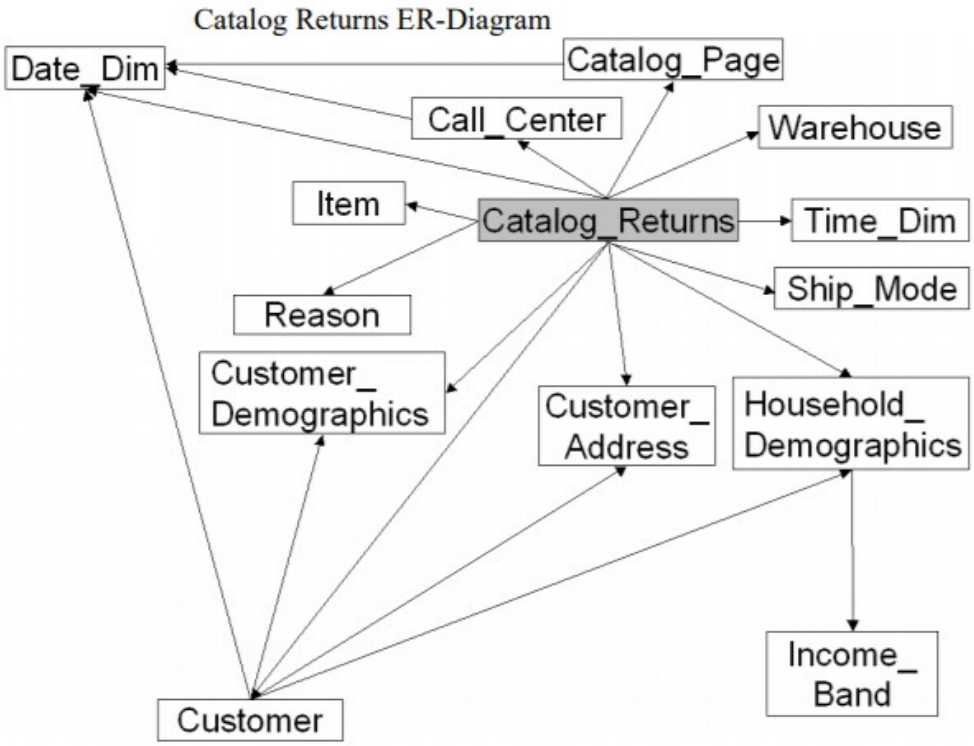

Figura 12 Schema ER di TPCDS : Catalog Returns

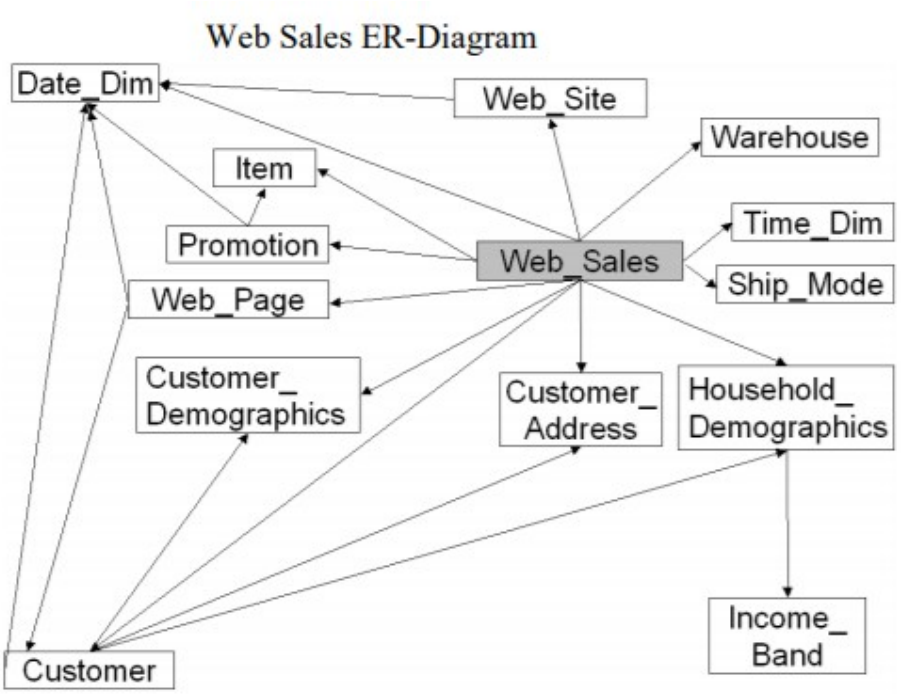

Figura 13 Schema ER di TPCDS : Web Sales

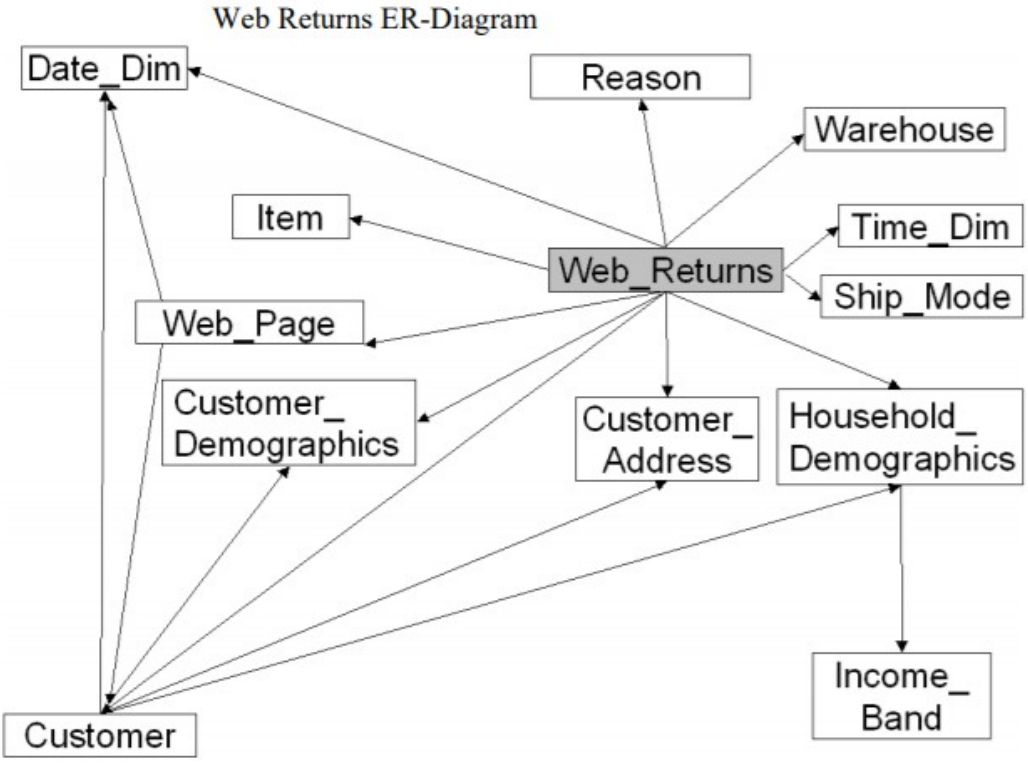

Figura 14 Schema ER di TPCDS : Web Returns

Il data workload è affidato a due componenti: carico delle query e mantenimento di dati. TPC-DS specifica un set di 99 quey distinte progettate per coprire l'intero set di dati. Lo schema è creato in modo tale da avere una parte di reporting (canale di vendita del catalogo, 40% dell'intero set di dati) e una parte ad-hoc (canali di vendita online e retail store). Le classi di query sono le seguenti: query di reporting puro, interrogazioni ad-hoc pure, query OLAP iterativa e query di estrazione o data mining.

3.2 Criticità

Swinchbench può essere, in generale, lanciato da interfaccia o da prompt dei comandi, in particolare il comando -charbench che consente agli utilizzatori il lancio dei test di carico; tuttavia, se lanciato da linea di comando, non è possibile mettere in pausa o interrompere il test. Si crea un processo in background che resta indipendente, qualsiasi sia il benchmark scelto (è da sottolineare che sono stati testati solamente i benchmark sopra citati).

I test su oe e sh da soli non hanno un impatto significativo sul carico di lavoro del database, si è optato per lanciarli in parallelo, in seguito scartato in quanto il sistema andava in eccessivo sovraccarico. La soluzione adottata è stata quella di affiancare altri test in modo da avere un impatto evidente sul database di test utilizzato, ma che fossero customizzabili. È stato, inoltre, aggiunto un nuovo benchmark su cui lanciare degli stress di carico in serie insieme a oe e sh, TPC-DSlike, descritto nel paragrafo precedente.

Altra criticità è l'impossibilità di tenere traccia delle query che Swingbench esegue, limitando l'analisi ai plot di metriche ignorando quale interrogazione sul database sia efficace nell'ipotesi in cui si voglia monitorare la relazione stress-query.

Il livello di impatto dei test è stato amplificato grazie all'opzione di Swingbench che consente la customizzazione dei test creandone alcuni che, combinati in parallelo a quelli predefiniti del tool, si sono rivelati efficaci ai fini dello stress del database. La scelta della creazione di un nuovo benchmark personalizzato ex novo, e non introdotto tra quelli presenti nei sample schema di Oracle, è stata ottimale per la maggiore elasticità nel compensare le carenze o gli "eccessi" dei benchmark base di Swingbench. Rispetto ai precedenti progetti lo stress complessivo generato è stato accresciuto dalla presenza del benchmark TCP-DSlike, che, quindi, combinato random in serie ai benchmark oe e sh, usati per creare un rumore di fondo, simula la normale attività di un database.

Infine, lo stress test complessivo è il risultato di un'alternanza randomica dei benchmark Swingbench elencati precedentemente i quali provvedono a simulare la normale attività di un database e, in parallelo, il benchmark personalizzato che aggiunge carico al traffico del database. Entrambi gli stress test hanno una durata, dei numeri di utenti e dei tempi di attesa randomizzati.

4. Data selection

4.1 Data collection

In linea con le finalità del progetto di tesi, sono state vagliate diverse opzioni in merito alla fonte di dati più adatta ad evidenziare dei comportamenti anomali dell'attività di un database Oracle. Oracle possiede una grande quantità di indicatori, raccolte in numerosissime viste di tipo dinamico in grado di descrivere al meglio le prestazioni del database nel tempo. Questo rende complessa qualsiasi analisi preliminare, rendendo necessaria una selezione di metriche che riescano a rappresentare lo status del database nel tempo. Ogni metrica è una serie ordinata nel tempo di valori che restituisce il trend di un fenomeno, la granularità è variabile, nel caso della vista dinamica presa in considerazione la frequenza di campionamento è di 60 secondi.

La creazione del dataset, nella forma finale, pronta per il pre-processing, è stata il risultato di diverse fasi, dalla raccolta, passando per l'analisi dei trend tramite plot e la selezione delle metriche utili ai fini dei passaggi successivi. Si è scelto di costruire un dataset raccogliendo delle metriche della vista di Oracle V$SYSMETRIC. La V$SYSMETRIC è una vista dinamica che cattura metriche di sistema sia a lunga durata – 60 secondi – che di breve durata – 15 secondi. possiede gli attributi elencati nella seguente immagine:

Column	Datatype	Description
BEGIN_TIME	DATE	Begin time of the interval
END_TIME	DATE	End time of the interval
INTSIZE_CSEC	NUMBER	Interval size (in hundredths of a second)
GROUP_ID	NUMBER	Metric group ID
METRIC_ID	NUMBER	Metric ID
METRIC_NAME	VARCHAR2(64)	Metric name
VALUE	NUMBER	Metric value
METRIC_UNIT	VARCHAR2(64)	Metric unit description
CON_ID	NUMBER	The ID of the container to which the data pertains. Possible values include: • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. • *n*: Where *n* is the applicable container ID for the rows containing data

Figura 15 Vista Oracle Sysmetric [9]

Raccolta dati

Per la data collection è stato creato un utente apposito in cui poter concentrare tutti i dati necessari, È stata creata una tabella, GV_SYSMETRIC, che ha gli stessi campi della vista precedentemente nominata, mostrati nella tabella con l'aggiunta del campo SAMPLE_TIME utile ai fini della raccolta dati e delle fasi successive.

Come evidenziato in precedenza, la vista V$SYSMETRIC ha un periodo di aggiornamento di 60 secondi, è stata, conseguentemente, creata una procedura di storicizzazione che raccogliesse le metriche e le inserisse nella tabella sopracitata con un periodo di campionamento equivalente a quello con cui la vista si aggiorna.

Era necessario che si avesse lo stesso intervallo di tempo affinché i dati puntuali non andassero perduti; a tale scopo è stato creato un job di tipo scheduler, come segue:

```
    BEGIN
  DBMS_SCHEDULER.ENABLE/DISABLE('TEST_GATHER_GV_SYSMETRIC)';
END;
```

abilitato durante il periodo di data collection e disattivato altrimenti.

La raccolta dei dati ha coinvolto tutte le metriche della vista, per un totale di 160 e risultante in una cardinalità di 8521 records.

Costruzione dataset

Come anticipato nei paragrafi precedenti, i dati non sono stati raccolti da un cliente, ma si riferiscono a un database utilizzato dall'azienda come ambiente di sviluppo interno.

I dati sono stati raccolti nel periodo compreso tra il 17 maggio e il 23 maggio 2022.

Per la raccolta dei dati è stata creata una tabella GV_SYSMETRIC per la memorizzazione contenente i campi mostrati di seguito

```
Create table GV_SYSMETRIC
(
        INST_ID NUMBER,
        BEGIN_TIME DATE,
        END_TIME DATE,
        INTSIZE_CSEC NUMBER,
        GROUP_IDNUMBER,
        METRIC_ID  NUMBER,
```

```
        METRIC_NAME VARCHAR2(64),
        VALU ENUMBER,
        METRIC_UNIT VARCHAR2(64),
        CON_ID NUMBER,
        DBNAME VARCHAR2(9),
        SAMPLE_TIME TIMESTAMP (6)
)
/
```

Pur inglobando tutti gli attributi della vista scelta, solo alcuni di essi ricoprono una funzione informativa necessaria all'analisi quali:

- METRIC_NAME
- VALUE
- SAMPLE_TIME

La raccolta dati ha necessitato anche di una procedura che storicizzasse le metriche e popolasse la tabella:

```
CREATE OR REPLACE PROCEDURE test.gather_gv_sysmetric AS
BEGIN
        INSERT                                                          INTO
        test.gv_sysstat(inst_id,statistic#,name,class,value,stat_id,con_id,dbname,sample_time)
        select a.*,b.name, sysdate from gv$sysstata, v$databaseb;
        INSERT INTO test.gv_sysstat (absolute_value)
        SELECT CASE
                WHEN VALUE=0  THEN  VALUE
                WHEN P_VALUE IS NULL THEN VALUE
                WHEN VALUE<P_VALUE THEN VALUE
                ELSE VALUE-P_VALUE
                END ABSOLUTE_VALUE
                FROM            (SELECTVALUE,           LAG(VALUE)          over
                PARTITIONBYNAMEORDERBYSAMPLE_TIMEASC) P_VALUE
                        FROMTEST.GV_SYSSTAT
                        WHERE DBNAME='OPA') A
;
        COMMIT;
END gather_gv_sysmetric;
/
```

La frequenza di raccolta è al minuto; si è scelto di mantenere tutti i campi della vista elencati al punto /// senza alterarne il contenuto informativo. Tuttavia, il formato della tabella rendeva l'analisi molto più complessa, sia per la mole di dati che per la disposizione degli stessi all'interno della tabella. A tal proposito, si è proceduto a creare una vista operando il PIVOT, che permette di aggregare i dati ancora grezzi tramite rotazione della tabella, mostrando i nomi delle metriche come attributi e i TIMESTAMP sulle righe, lasciando all'interno di ogni cella il VALUE della metrica (l'operazione è stata preceduta dalla creazione di una vista – filtro che seleziona gli attributi METRIC_NAME, SAMPLE_TIME, VALUE). Il codice SQL è mostrato di seguito:

```
Create view GV_SYSMETRIC_PIVOT as
Select
"SAMPLE_TIME","BackgroundCPUUsagePerSec","BackgroundTimePerSec","TempSpaceUsed"
,"UserTransactionPerSec","PhysicalReadsPerSec","PhysicalWritesPerSec","PhysicalWritesDir
ectPerSec","TotalParseCountPerSec","HardParseCountPerSec","CursorCacheHitRatio","UserR
ollbackUndoRecAppliedPerSec","PXdowngraded1to25%PerSec","PXdowngraded50to75%Pe
rSec","GlobalCacheAverageCurrentGetTime","ProcessLimit%","SessionLimit%","StreamsPool
UsagePercentage","I/ORequestsperSecond","ActiveParallelSessions","Capturedusercalls","Wo
rkloadCaptureandReplaystatus","TotalPGAAllocated","TotalPGAUsedbySQLWorkareas","Red
oAllocationHitRatio","PhysicalReadsDirectPerTxn","RedoGeneratedPerTxn","UserRollbacksP
erSec","LogicalReadsPerSec","RedoWritesPerTxn","LongTableScansPerTxn","FullIndexScans
PerTxn","UserCallsRatio","EnqueueTimeoutsPerSec","EnqueueWaitsPerTxn","ConsistentRea
dChangesPerTxn","CPUUsagePerTxn","CRBlocksCreatedPerTxn","LeafNodeSplitsPerSec","PX
downgraded25to50%PerSec","PXdowngradedtoserialPerSec","GCCRBlockReceivedPerSecon
d","GCCRBlockReceivedPerTxn","CurrentOpenCursorsCount","SQLServiceResponseTime","Da
tabaseWaitTimeRatio","SharedPoolFree%","ExecutionsPerUserCall","CurrentOSLoad","I/OM
egabytaesperSecond","PhysicalReadsDirectPerSec","OpenCursorsPerSec","DBWRCheckpoint
sPerSec","TotalTableScansPerSec","RowsPerSort","NetworkTrafficVolumePerSec","Enqueue
DeadlocksPerTxn","PXdowngraded75to99%PerSec","GCCurrentBlockReceivedPerTxn","Row
CacheHitRatio","PhysicalReadBytesPerSec","VMoutbytesPerSec","BackgroundCheckpointsPe
rSec","HardParseCountPerTxn","ParseFailureCountPerSec","ParseFailureCountPerTxn","Disk
SortPerSec","ExecuteWithoutParseRatio","EnqueueRequestsPerSec","DBBlockChangesPerTx
n","UserRollbackUndoRecordsAppliedPerTxn","LeafNodeSplitsPerTxn","GlobalCacheAverage
CRGetTime","GlobalCacheBlocksCorrupted","DatabaseCPUTimeRatio","ExecutionsPerTxn","L
ogicalReadsPerUserCall","TotalSortsPerUserCall","TotalTableScansPerUserCall","DMLstatem
entsparallelizedPerSec","PXoperationsnotdowngradedPerSec","VMinbytesPerSec","LogonsPe
```

```
rTxn","UserCommitsPerSec","RecursiveCallsPerTxn","TotalIndexScansPerSec","SoftParseRati
o","EnqueueWaitsPerSec","DBBlockGetsPerSec","ConsistentReadGetsPerTxn","ConsistentRea
dChangesPerSec","BranchNodeSplitsPerSec","PhysicalReadTotalBytesPerSec","GCCurrentBlo
ckReceivedPerSecond","PGACacheHit%","ExecutionsPerSec","TxnsPerLogon","PhysicalWrite
BytesPerSec","DBBlockChangesPerUserCall","PQQCSessionCount","DDLstatementsparallelize
dPerSec","SessionCount","MemorySortsRatio","RedoGeneratedPerSec","OpenCursorsPerTxn"
,"UserCommitsPercentage","RecursiveCallsPerSec","TotalParseCountPerTxn","EnqueueTime
outsPerTxn","DBBlockChangesPerSec","CPUUsagePerSec","CRBlocksCreatedPerSec","Physica
lWriteTotalIORequestsPerSec","GlobalCacheBlocksLost","ResponseTimePerTxn","RowCache
MissRatio","LibraryCacheMissRatio","DBBlockGetsPerUserCall","PQSlaveSessionCount","Aver
ageSynchronousSingle-
BlockReadLatency","AverageActiveSessions","Replayedusercalls","HostCPUUsagePerSec","Ce
llPhysicalIOInterconnectBytes","PhysicalReadsPerTxn","PhysicalReadsDirectLobsPerSec","Ph
ysicalWritesDirectLobsPerTxn","UserCallsPerTxn","LogicalReadsPerTxn","TotalIndexScansP
erTxn","DiskSortPerTxn","EnqueueRequestsPerTxn","ConsistentReadGetsPerSec","BranchNo
deSplitsPerTxn","PhysicalReadTotalIORequestsPerSec","LibraryCacheHitRatio","PhysicalWrit
eTotalBytesPerSec","PhysicalWriteIORequestsPerSec","ActiveSerialSessions","RunQueuePerS
ec","BufferCacheHitRatio","PhysicalWritesPerTxn","PhysicalWritesDirectPerTxn","PhysicalRe
adsDirectLobsPerTxn","PhysicalWritesDirectLobsPerSec","LogonsPerSec","UserRollbacksPer
centage","UserCallsPerSec","RedoWritesPerSec","LongTableScansPerSec","TotalTableScansP
erTxn","FullIndexScansPerSec","HostCPUUtilization(%)","EnqueueDeadlocksPerSec","DBBloc
kGetsPerTxn","CRUndoRecordsAppliedPerSec","CRUndoRecordsAppliedPerTxn","CurrentLog
onsCount","UserLimit%","DatabaseTimePerSec","PhysicalReadIORequestsPerSec","Queriesp
arallelizedPerSec"
From (SELECT SAMPLE_TIME, METRIC_ NAME, VALUE from "TEST"."GV_SYSMETRIC")
Pivot (
min (VALUE)
for METRIC_NAME in (
'BackgroundCPUUsagePerSec' as "BackgroundCPUUsagePerSec",
'BackgroundTimePerSec' as "BackgroundTimePerSec",
'TempSpaceUsed'as"TempSpaceUsed",
'UserTransactionPerSec' as "UserTransactionPerSec",
'PhysicalReadsPerSec'as"PhysicalReadsPerSec",
'PhysicalWritesPerSec'as"PhysicalWritesPerSec",
'PhysicalWritesDirectPerSec'as"PhysicalWritesDirectPerSec",
'TotalParseCountPerSec'as"TotalParseCountPerSec",
```

'HardParseCountPerSec'as"HardParseCountPerSec",
'CursorCacheHitRatio' as "CursorCacheHitRatio",
'UserRollbackUndoRecAppliedPerSec' as "UserRollbackUndoRecAppliedPerSec",
'PXdowngraded1to25%PerSec' as "PXdowngraded1to25%PerSec",
'PXdowngraded50to75%PerSec' as "PXdowngraded50to75%PerSec",
'GlobalCacheAverageCurrentGetTime' as "GlobalCacheAverageCurrentGetTime",
'ProcessLimit%' as "ProcessLimit%",
'SessionLimit%' as "SessionLimit%",
'StreamsPoolUsagePercentage' as "StreamsPoolUsagePercentage",
'I/ORequestsperSecond' as "I/ORequestsperSecond",
'ActiveParallelSessions' as "ActiveParallelSessions",
'Capturedusercalls' as "Capturedusercalls",
'WorkloadCaptureandReplaystatus' as "WorkloadCaptureandReplaystatus",
'TotalPGAAllocated' as "TotalPGAAllocated",
'TotalPGAUsedbySQLWorkareas' as "TotalPGAUsedbySQLWorkareas",
'RedoAllocationHitRatio' as "RedoAllocationHitRatio",
'PhysicalReadsDirectPerTxn' as "PhysicalReadsDirectPerTxn",
'RedoGeneratedPerTxn'as"RedoGeneratedPerTxn",
'UserRollbacksPerSec'as"UserRollbacksPerSec",
'LogicalReadsPerSec'as"LogicalReadsPerSec",
'RedoWritesPerTxn'as"RedoWritesPerTxn",
'LongTableScansPerTxn'as"LongTableScansPerTxn",
'FullIndexScansPerTxn'as"FullIndexScansPerTxn",
'UserCallsRatio'as"UserCallsRatio",
'EnqueueTimeoutsPerSec'as"EnqueueTimeoutsPerSec",
'EnqueueWaitsPerTxn'as"EnqueueWaitsPerTxn",
'ConsistentReadChangesPerTxn'as"ConsistentReadChangesPerTxn",
'CPUUsagePerTxn'as"CPUUsagePerTxn",
'CRBlocksCreatedPerTxn'as"CRBlocksCreatedPerTxn",
'LeafNodeSplitsPerSec'as"LeafNodeSplitsPerSec",
'PXdowngraded25to50%PerSec'as"PXdowngraded25to50%PerSec",
'PXdowngradedtoserialPerSec'as"PXdowngradedtoserialPerSec",
'GCCRBlockReceivedPerSecond'as"GCCRBlockReceivedPerSecond",
'GCCRBlockReceivedPerTxn'as"GCCRBlockReceivedPerTxn",
'CurrentOpenCursorsCount'as"CurrentOpenCursorsCount",
'SQLServiceResponseTime'as"SQLServiceResponseTime",

'DatabaseWaitTimeRatio'as"DatabaseWaitTimeRatio",
'SharedPoolFree%'as"SharedPoolFree%",
'ExecutionsPerUserCall'as"ExecutionsPerUserCall",
'CurrentOSLoad'as"CurrentOSLoad",
'I/OMegabytesperSecond'as"I/OMegabytesperSecond",
'PhysicalReadsDirectPerSec'as"PhysicalReadsDirectPerSec",
'OpenCursorsPerSec'as"OpenCursorsPerSec",
'DBWRCheckpointsPerSec'as"DBWRCheckpointsPerSec",
'TotalTableScansPerSec'as"TotalTableScansPerSec",
'RowsPerSort'as"RowsPerSort",
'NetworkTrafficVolumePerSec'as"NetworkTrafficVolumePerSec",
'EnqueueDeadlocksPerTxn'as"EnqueueDeadlocksPerTxn",
'PXdowngraded75to99%PerSec'as"PXdowngraded75to99%PerSec",
'GCCurrentBlockReceivedPerTxn'as"GCCurrentBlockReceivedPerTxn",
'RowCacheHitRatio'as"RowCacheHitRatio",
'PhysicalReadBytesPerSec'as"PhysicalReadBytesPerSec",
'VMoutbytesPerSec'as"VMoutbytesPerSec",
'BackgroundCheckpointsPerSec'as"BackgroundCheckpointsPerSec",
'HardParseCountPerTxn'as"HardParseCountPerTxn",
'ParseFailureCountPerSec'as"ParseFailureCountPerSec",
'ParseFailureCountPerTxn'as"ParseFailureCountPerTxn",
'DiskSortPerSec'as"DiskSortPerSec",
'ExecuteWithoutParseRatio'as"ExecuteWithoutParseRatio",
'EnqueueRequestsPerSec'as"EnqueueRequestsPerSec",
'DBBlockChangesPerTxn'as"DBBlockChangesPerTxn",
'UserRollbackUndoRecordsAppliedPerTxn'as"UserRollbackUndoRecordsAppliedPerTxn",
'LeafNodeSplitsPerTxn'as"LeafNodeSplitsPerTxn",
'GlobalCacheAverageCRGetTime'as"GlobalCacheAverageCRGetTime",
'GlobalCacheBlocksCorrupted'as"GlobalCacheBlocksCorrupted",
'DatabaseCPUTimeRatio'as"DatabaseCPUTimeRatio",
'ExecutionsPerTxn'as"ExecutionsPerTxn",
'LogicalReadsPerUserCall'as"LogicalReadsPerUserCall",
'TotalSortsPerUserCall'as"TotalSortsPerUserCall",
'TotalTableScansPerUserCall'as"TotalTableScansPerUserCall",
'DMLstatementsparallelizedPerSec'as"DMLstatementsparallelizedPerSec",
'PXoperationsnotdowngradedPerSec'as"PXoperationsnotdowngradedPerSec",

'VMinbytesPerSec'as"VMinbytesPerSec",
'LogonsPerTxn'as"LogonsPerTxn",
'UserCommitsPerSec'as"UserCommitsPerSec",
'RecursiveCallsPerTxn'as"RecursiveCallsPerTxn",
'TotalIndexScansPerSec'as"TotalIndexScansPerSec",
'SoftParseRatio'as"SoftParseRatio",
'EnqueueWaitsPerSec'as"EnqueueWaitsPerSec",
'DBBlockGetsPerSec'as"DBBlockGetsPerSec",
'ConsistentReadGetsPerTxn'as"ConsistentReadGetsPerTxn",
'ConsistentReadChangesPerSec'as"ConsistentReadChangesPerSec",
'BranchNodeSplitsPerSec'as"BranchNodeSplitsPerSec",
'PhysicalReadTotalBytesPerSec'as"PhysicalReadTotalBytesPerSec",
'GCCurrentBlockReceivedPerSecond'as"GCCurrentBlockReceivedPerSecond",
'PGACacheHit%'as"PGACacheHit%",
'ExecutionsPerSec'as"ExecutionsPerSec",
'TxnsPerLogon'as"TxnsPerLogon",
'PhysicalWriteBytesPerSec'as"PhysicalWriteBytesPerSec",
'DBBlockChangesPerUserCall'as"DBBlockChangesPerUserCall",
'PQQCSessionCount'as"PQQCSessionCount",
'DDLstatementsparallelizedPerSec'as"DDLstatementsparallelizedPerSec",
'SessionCount'as"SessionCount",
'MemorySortsRatio'as"MemorySortsRatio",
'RedoGeneratedPerSec'as"RedoGeneratedPerSec",
'OpenCursorsPerTxn'as"OpenCursorsPerTxn",
'UserCommitsPercentage'as"UserCommitsPercentage",
'RecursiveCallsPerSec'as"RecursiveCallsPerSec",
'TotalParseCountPerTxn'as"TotalParseCountPerTxn",
'EnqueueTimeoutsPerTxn'as"EnqueueTimeoutsPerTxn",
'DBBlockChangesPerSec'as"DBBlockChangesPerSec",
'CPUUsagePerSec'as"CPUUsagePerSec",
'CRBlocksCreatedPerSec'as"CRBlocksCreatedPerSec",
'PhysicalWriteTotalIORequestsPerSec'as"PhysicalWriteTotalIORequestsPerSec",
'GlobalCacheBlocksLost'as"GlobalCacheBlocksLost",
'ResponseTimePerTxn'as"ResponseTimePerTxn",
'RowCacheMissRatio'as"RowCacheMissRatio",
'LibraryCacheMissRatio'as"LibraryCacheMissRatio",

'DBBlockGetsPerUserCall'as"DBBlockGetsPerUserCall",
'PQSlaveSessionCount'as"PQSlaveSessionCount",
'AverageSynchronousSingle-BlockReadLatency'as"AverageSynchronousSingle-BlockReadLatency",
'AverageActiveSessions'as"AverageActiveSessions",
'Replayedusercalls'as"Replayedusercalls",
'HostCPUUsagePerSec'as"HostCPUUsagePerSec",
'CellPhysicalIOInterconnectBytes'as"CellPhysicalIOInterconnectBytes",
'PhysicalReadsPerTxn'as"PhysicalReadsPerTxn",
'PhysicalReadsDirectLobsPerSec'as"PhysicalReadsDirectLobsPerSec",
'PhysicalWritesDirectLobsPerTxn'as"PhysicalWritesDirectLobsPerTxn",
'UserCallsPerTxn'as"UserCallsPerTxn",
'LogicalReadsPerTxn'as"LogicalReadsPerTxn",
'TotalIndexScansPerTxn'as"TotalIndexScansPerTxn",
'DiskSortPerTxn'as"DiskSortPerTxn",
'EnqueueRequestsPerTxn'as"EnqueueRequestsPerTxn",
'ConsistentReadGetsPerSec'as"ConsistentReadGetsPerSec",
'BranchNodeSplitsPerTxn'as"BranchNodeSplitsPerTxn",
'PhysicalReadTotalIORequestsPerSec'as"PhysicalReadTotalIORequestsPerSec",
'LibraryCacheHitRatio'as"LibraryCacheHitRatio",
'PhysicalWriteTotalBytesPerSec'as"PhysicalWriteTotalBytesPerSec",
'PhysicalWriteIORequestsPerSec'as"PhysicalWriteIORequestsPerSec",
'ActiveSerialSessions'as"ActiveSerialSessions",
'RunQueuePerSec'as"RunQueuePerSec",
'BufferCacheHitRatio'as"BufferCacheHitRatio",
'PhysicalWritesPerTxn'as"PhysicalWritesPerTxn",
'PhysicalWritesDirectPerTxn'as"PhysicalWritesDirectPerTxn",
'PhysicalReadsDirectLobsPerTxn'as"PhysicalReadsDirectLobsPerTxn",
'PhysicalWritesDirectLobsPerSec'as"PhysicalWritesDirectLobsPerSec",
'LogonsPerSec'as"LogonsPerSec",
'UserRollbacksPercentage'as"UserRollbacksPercentage",
'UserCallsPerSec'as"UserCallsPerSec",
'RedoWritesPerSec'as"RedoWritesPerSec",
'LongTableScansPerSec'as"LongTableScansPerSec",
'TotalTableScansPerTxn'as"TotalTableScansPerTxn",
'FullIndexScansPerSec'as"FullIndexScansPerSec",

```
'HostCPUUtilization(%)'as"HostCPUUtilization(%)",
'EnqueueDeadlocksPerSec'as"EnqueueDeadlocksPerSec",
'DBBlockGetsPerTxn'as"DBBlockGetsPerTxn",
'CRUndoRecordsAppliedPerSec'as"CRUndoRecordsAppliedPerSec",
'CRUndoRecordsAppliedPerTxn'as"CRUndoRecordsAppliedPerTxn",
'CurrentLogonsCount'as"CurrentLogonsCount",
'UserLimit%'as"UserLimit%",
'DatabaseTimePerSec'as"DatabaseTimePerSec",
'PhysicalReadIORequestsPerSec'as"PhysicalReadIORequestsPerSec",
'QueriesparallelizedPerSec'as"QueriesparallelizedPerSec")
)

Order by SAMPLE_TIME
/
```

La vista GV_SYSMETRIC_PIVOT ha una struttura tabellare come segue

Tabella 1 Pivot della Tabella GV_SYSMETRIC

SAMPLE_TIME	Active Serial Sessions	Average Active Sessions	Background CPU Usage Per Sec	Background Time Per Sec
17/05/2022 15:50	1	0,000381005	0,886032962	0,013107258
17/05/2022 15:51	1	0,000946929	1,017696021	0,013288996
17/05/2022 15:52	1	8,21345E-05	0,820549451	0,012819447
17/05/2022 15:53	1	0,00073436	1,024726153	0,015286815
17/05/2022 15:54	1	0,005162644	0,897331447	0,013454786
17/05/2022 15:55	1	0,000225404	0,835685034	0,013741701

La vista non prevede delle classi di raggruppamento, ma gli esperti di dominio hanno suddiviso le metriche nelle seguenti categorie d'ora in avanti citate come LABEL:

- CACHE
- CPU
- DEBUG
- ENQUEUE

- I/O
- RAC
- SQL
- USER

Data visualization e analisi trend

Il passo successivo alla raccolta è la visualizzazione dei dati in modo da verificare che l'andamento delle metriche sia variabile e variegato. In virtù del fatto che la vista SYSMETRIC contiene un alto numero di metriche, è stato inizialmente effettuato un plot di ciascuna delle metriche e una successiva analisi qualitativa di tipo esplorativo.

Sono di seguito mostrate alcune delle 160 metriche presenti nei dataset con lo scopo di evidenziare la diversità di andamento e di dominio che saranno oggetto di successivi studi all'interno del progetto di tesi:

- Active Serial Session: rappresenta il numero di sessioni attive;
- Execute Without Parse Ratio: rappresenta la percentuale di esecuzioni che non richiedono un parse corrispondente. Un sistema perfetto farebbe il parsing di tutti gli statement una vola e poi eseguire lo statement che ha subito il parsing senza aver bisogno che l'operazione sia ripetuta (valori alti sono ideali);
- Memory sorts ratio: è la percentuale di sort (da clausole ORDER BY o creazione di indici) che sono fatti nel disco vs. quelli effettuati in memoria. I sort del disco sono eseguiti nel Temp Space, centinaia di volte più lento di un sort in RAM;
- PGA Cache HIT %: la Program Global Area (PGA) è una regione di memoria di un Database che contiene dati e controlla le informazioni per un singolo processo (server o background). Di conseguenza, l'acronimo PGA può anche corrispondere a Process Global Area; contiene aree dati e Stack;
- Total Parse Count Per Sec: conta tutti i parsing, sia soft che hard, per secondo;
- Total PGA Allocated: mostra i Bytes che la PGA alloca in quell'istante;
- User Commits Percentage: la percentuale di commits operati in un determinato istante di tempo.

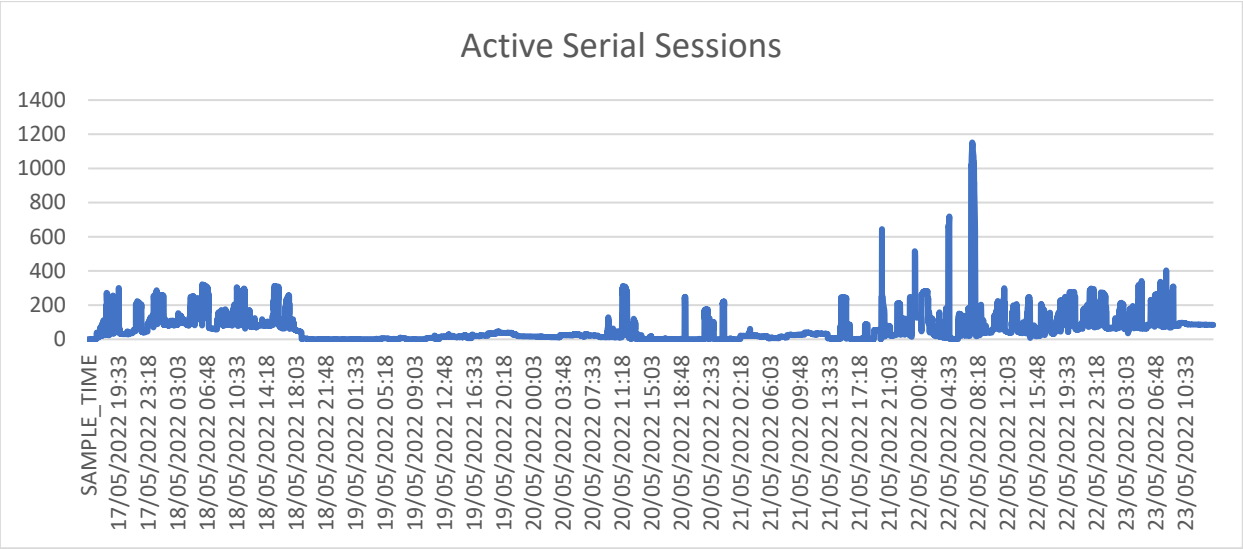

Figura 16 : Active Serial Sessions

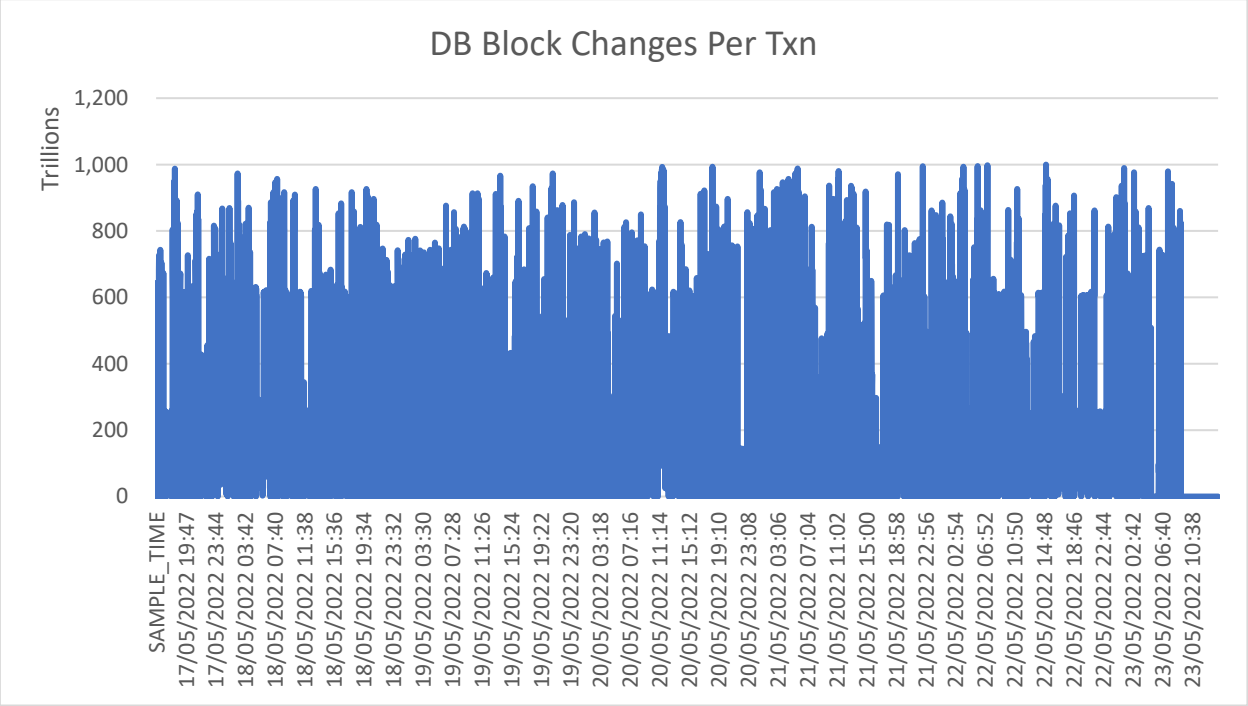

Figura 17 DB Block Changes Per Txn

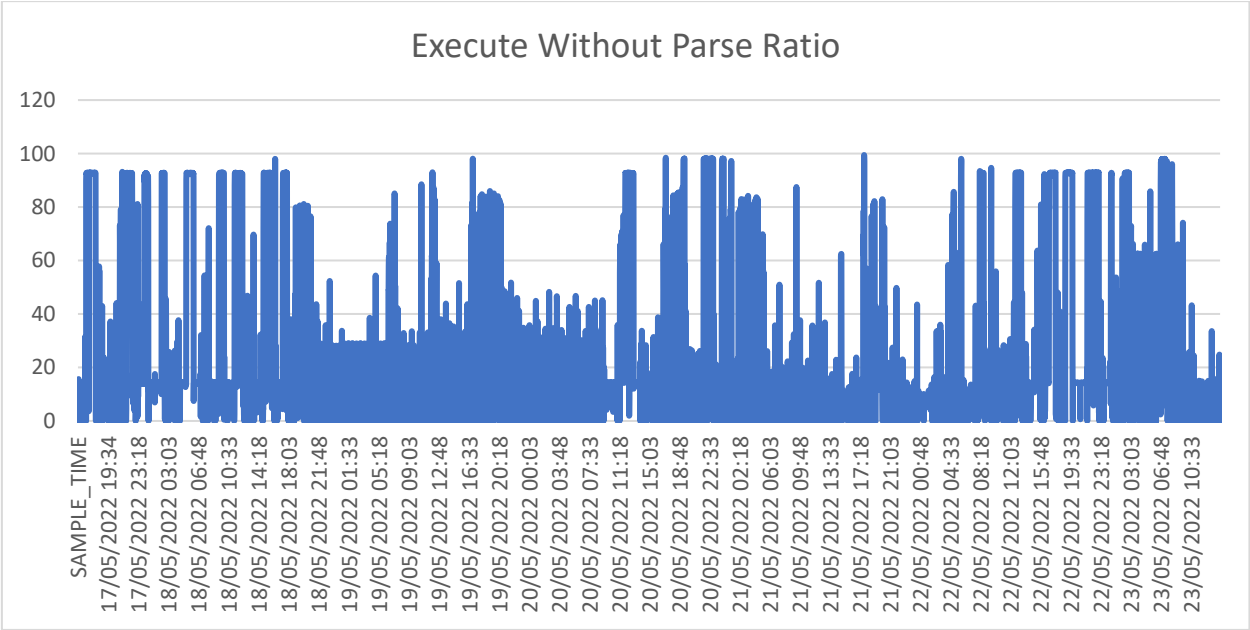

Figura 18 Execute Without Parse Ratio

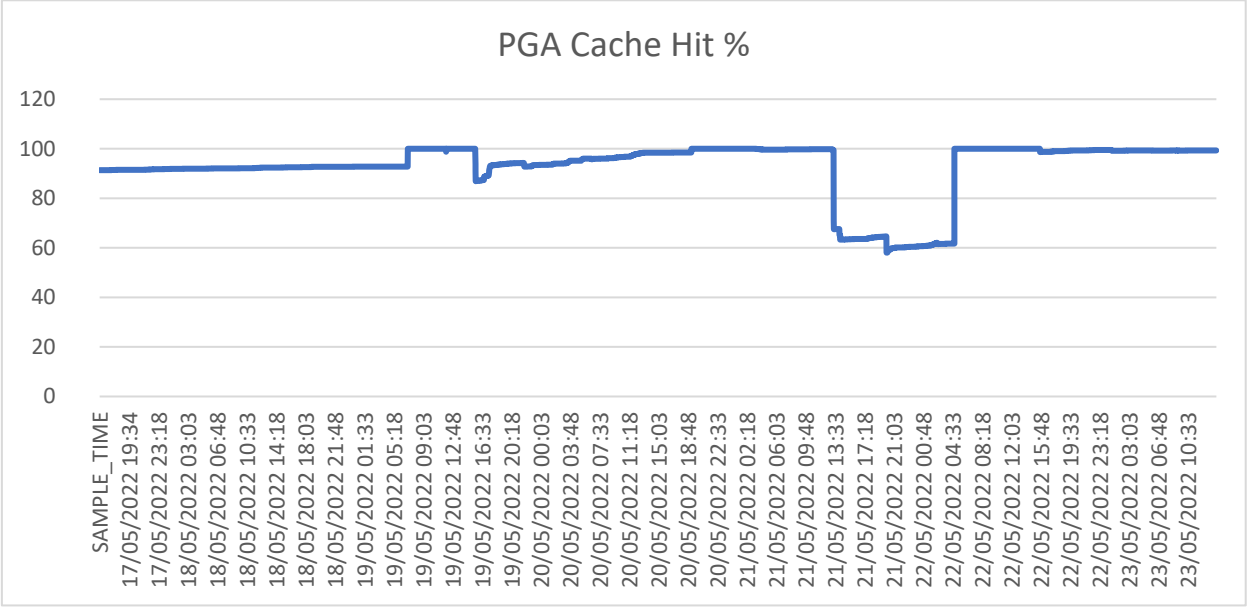

Figura 19 PGA Cache Hit %

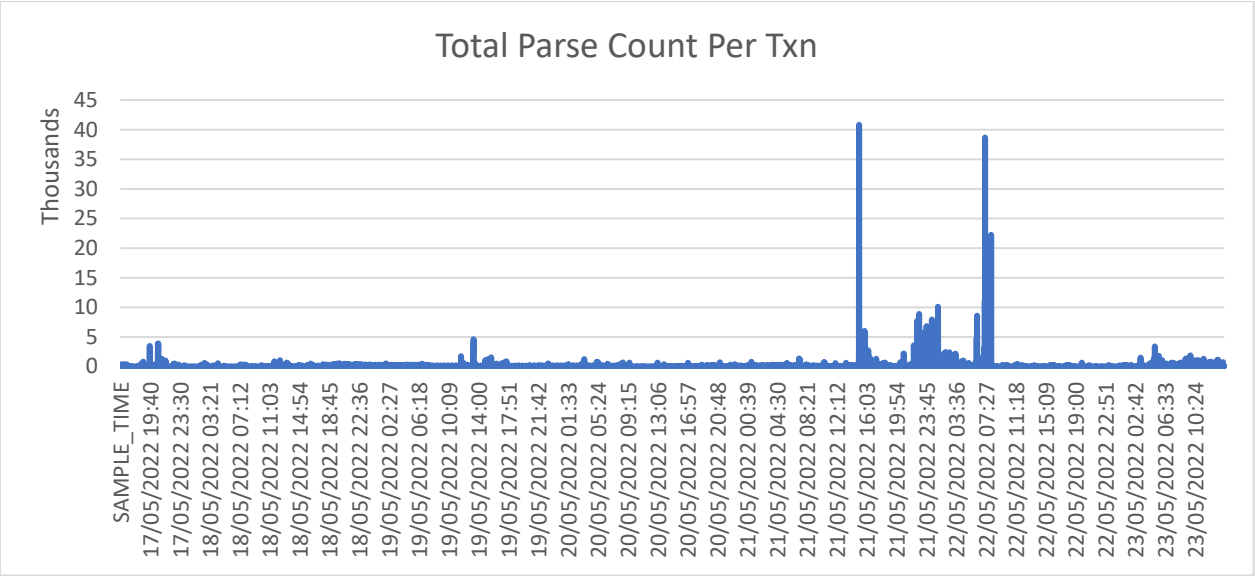

Figura 20 Total Parse Count Per Txn

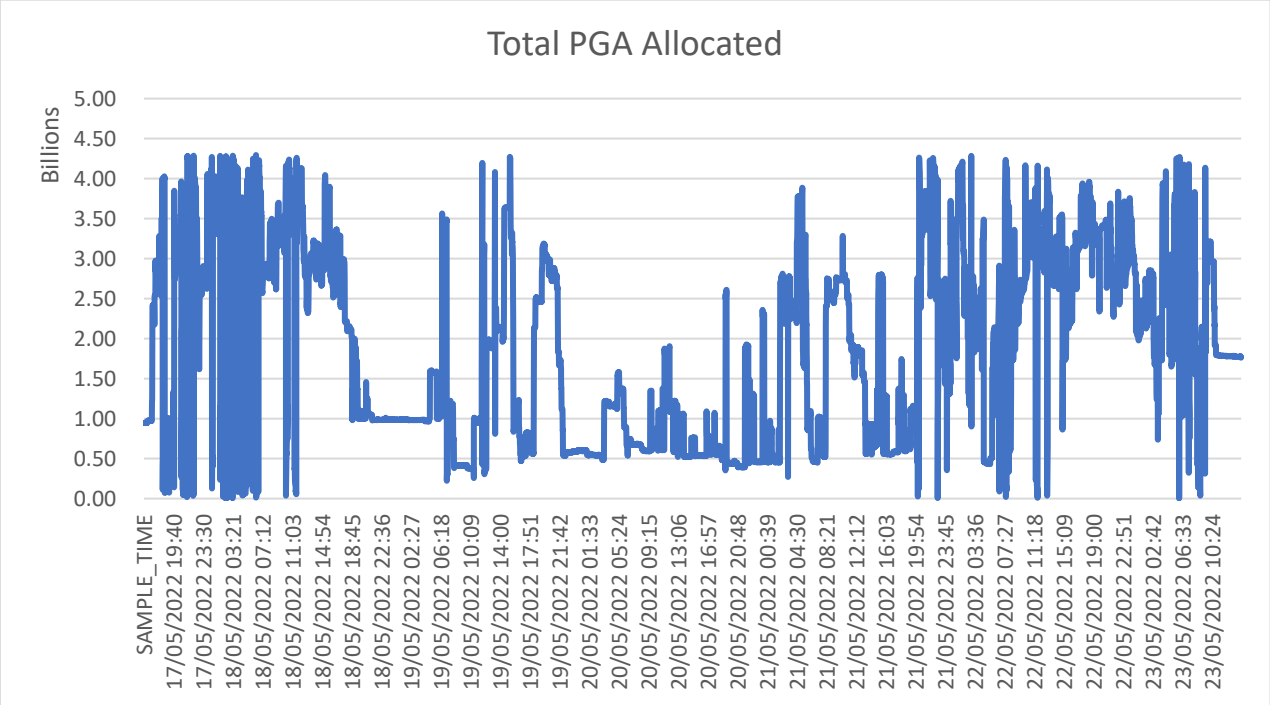

Figura 21 Total PGA Allocated

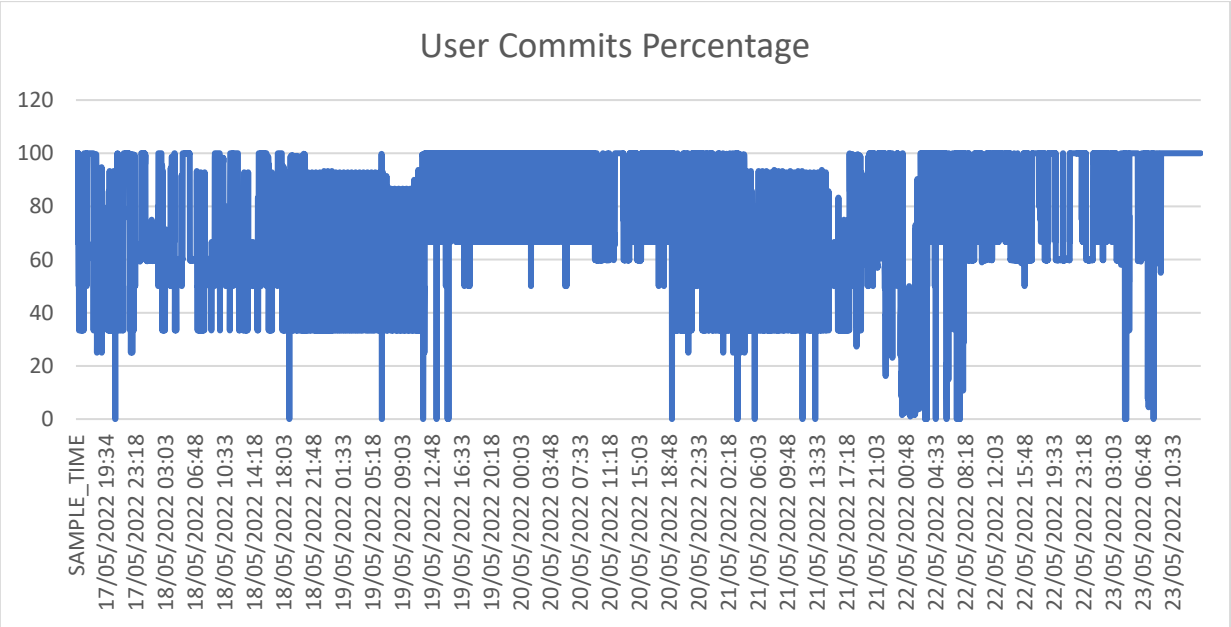

Figura 22 User commits Percentage

Dall'analisi grafica è possibile affermare che il dataset presenta un'elevata varietà in termini sia di andamento sia di informazione riguardante diversi aspetti del sistema informatico; si può altresì affermare che la grande varietà di andamenti potrebbe tradursi nella presenza di rumore e ridondanza.

5. Data Preprocessing

Per Data Preprocessing si intende l'insieme di tecniche di preparazione dei dati per migliorarne la qualità, identificare e rimuovere correttamente i dati considerati superflui o poco significanti. I metodi esistenti in letteratura sono tanti in termini di tipologie, al cui interno si trovano, inoltre, diverse strategie di approccio. Nel presente lavoro di tesi ne sono stati impiegate diverse delle suddette metodologie.

Il preprocessing è successivo alla fase di esplorazione dei dati, descritta precedentemente, ed entra in gioco per preparare il dataset al modello. Se questo step fosse trascurato, tutto il lavoro successivo sarebbe condizionato dalla qualità scadente dei dati, poco accurati.

In teoria, avere tante feature dovrebbe tradursi in un potere più discriminante; tuttavia, l'esperienza con gli algoritmi di machine learning ha mostrato come non è sempre così. Le ricerche hanno mostrato che i comuni algoritmi di machine learning possono essere influenzati negativamente da informazioni irrilevanti o ridondanti.

La fase di Preprocessing si prevede essere come la più difficoltosa e time-consuming della data-science, ma anche una delle più importanti per gli step successivi del progetto; se i dati non fossero stati necessariamente analizzati potrebbero compromettere gli output i modelli.

Le tecniche di Preprocessing servono a rendere i dati raccolti più utilizzabili, facilitando l'uso degli algoritmi di machine learning, riducono la complessità per prevenire l'overfitting, per risultare infine in un miglior modello.

Nel presente lavoro di tesi le tecniche di preprocessing utilizzate sono le seguenti: data cleaning, dimensionality reduction, data transformation di cui segue una descrizione.

5.1 Data cleaning

Uno degli scopi più influenti del data preprocessing è individuare e risolvere osservazioni inaccurate dal dataset per migliorarne la qualità. In particolare, riguarda l'individuazione di valori incompleti, inaccurati, duplicati, irrilevanti o nulli nei dati; una volta individuati, si decide se modificarli o eliminarli. La scelta migliore è dipendente dal dominio del problema e dall'obiettivo del progetto. Il data cleaning costituisce, in generale, il primo step del preprocessing, in cui ci si assicura che la qualità del dato sia consistente o che lo diventi.

Nel dataset preso in esame, non erano presenti né dati rumorosi né dati mancanti o che presentassero errori strutturali o NaN. Tuttavia, i dati raccolti per alcune metriche si sono rivelati nulli o di valore costante, sono stati pertanto eliminati dal dataset, risultato così più consistente.

5.2 Data Transformation

Costituisce uno degli step più critici nella fase di Preprocessing, in quanto converte i dati da un formato a un altro e, al contempo, mantiene invariato il pool di informazioni contenuto nei dati ricevuti in input. Alcune delle tecniche usate per risolvere il problema sono: trasformazione per variabili categoriche, normalizzazione min/max, standard scaler (standardizzazione). Va comunque tenuto in conto che la scelta va compiuta guardando con attenzione al dataset.

La normalizzazione, tra cui è presente il min-max scaler, è una delle più comuni e scalarizza i dati entro un range (di solito tra 0 e 1). Il contro principale con questa tecnica è che è sensitiva agli outliers, ma è utile se i dati non seguono la distribuzione normale.

La normalizzazione standard è un'altra tecnica ampiamente usata, anche conosciuta come z-score normalizzazione standardizzazione.

La Data Transformation racchiude delle tecniche che hanno il fine di modificare il dataset pur mantenendone invariate le caratteristiche o per rendere i dati presenti più conformi. L'Attribute Transformation è, in particolare, una funzione che riceve in input i dati di un attributo e restituisce in output un set di dati che rimpiazzano i precedenti e che li rappresentano ancora, anche a livello di caratteristiche; le tecniche più diffuse sono la normalizzazione e la standardizzazione. La normalizzazione aggiusta le differenze tra gli attributi in termini di frequenza, range o media e opera un ridimensionamento dei dati in un intervallo che tipicamente è [-1; +1] o [0; +1]. Tra le svariate tecniche troviamo la normalizzazione min-max, la scalarizzazione decimale e la normalizzazione z-score. Nel presente lavoro di tesi è stata impiegata quest'ultima, può avere sia valori positivi che negativi; la formula della normalizzazione z-score è la seguente:

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

con l'obiettivo di ridimensionare tutte le metriche con un dominio troppo ampio rispetto alle altre ed evitare la presenza di eventuali valori NaN nelle fasi successive del processo. La normalizzazione z-score è, infine, molto più sensibile alle variazioni di valori all'interno del dominio di una metrica rispetto alla normalizzazione min/max che, trasformando i valori in un range limitato, non riesce a catturare la variabilità in presenza di dati molto eterogenei tra loro.

5.3 Feature selection

Per feature selection si intende il processo di selezione di un sottogruppo di features secondo dei criteri, tra cui la rilevanza e la rappresentabilità del dataset da cui sono state selezionate per la costruzione di un modello. La feature selection, detta anche attribute selection o variable subset selection, ha come obiettivo la maggiore facilità di analisi del dataset, per migliorare la compatibilità dei dati, ridurre la dimensione del dataset, ottenere una semplificazione del set di dati, ridurre la cardinalità e i tempi di training dei modelli.

La Feature subset selection può anche essere considerata un caso delle dimensionality reduction; in questo caso le feature sono gli attributi. È importante identificare le features rilevanti che catturano la giusta variabilità dei dati sotto osservazione, scegliere le giuste tecniche con l'obiettivo di valutarne l'impatto sulla pipeline del KDD descritto in precedenza. Sono escluse le features ridondanti perché legate da informazioni comuni, correlate e anche, dati che contengono informazioni irrilevanti ai fini del data mining. Il subset ottimale dovrebbe contenere features non correlate in modo tale da ottenere una buona modellazione del problema.

Il problema in cui si incorre nel caso in cui sarebbero presenti troppe features o un subset non rappresentativo del dataset è l'overfitting, in cui il modello segue in modo quasi perfetto l'andamento dei dati, risultando poco coerente con dei dati diversi dal dataset di partenza.

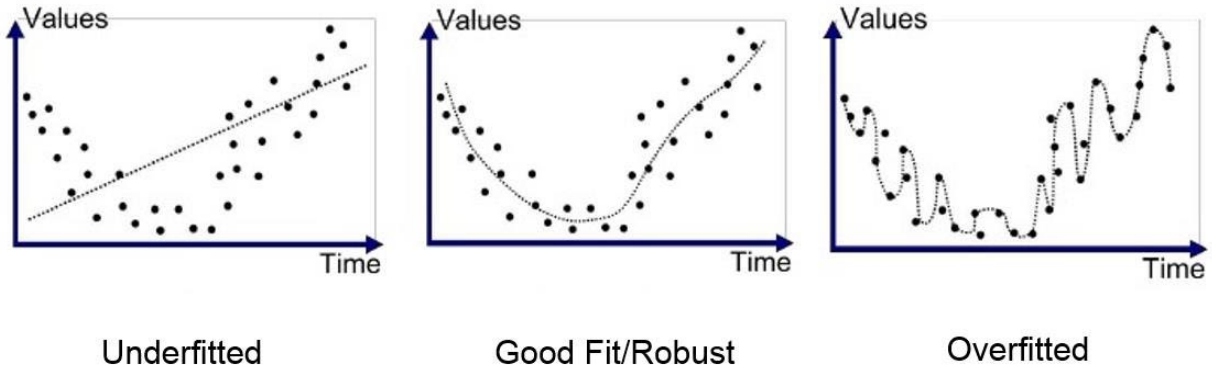

Figura 23 Diversi fitting dei dati a cui è applicato un modello matematico

5.3.1 Dimensionality reduction

Come intuibile dal termine stesso, si tratta della riduzione del numero di features in input del dataset. L'obiettivo è sì la riduzione, ma è da tenere in considerazione anche la preservazione della variabilità e della complessità delle informazioni racchiuse all'interno dei dati. Alcuni dei benefici che questo passaggio apporta al progetto sono: riduzione delle risorse computazionali; evitare l'overfitting (quando, infatti, il modello diventa troppo complicato e memorizza i dati di training anziché imparare da essi avendo come risultato una riduzione evidente della performance); rende la performance del modello migliore nel suo complesso; evita la multicollinearità (alta correlazione di una o più variabili indipendenti).

Nel presente progetto è stata presa in considerazione la Feature Selection: questo termine si riferisce al processo di selezione delle variabili ritenute più importanti e che si ritiene possano apportare un contributo al modello. Alcuni metodi applicabili automaticamente o manualmente sono: test statistici; eliminazione ricorsiva delle feature, correlazione tra le feature e soglia di varianza; sono, inoltre, presenti alcuni modelli applicabili automaticamente durante la fase di training dei dati.

La scelta è ricaduta nella correlazione tra le feature. L'indice di correlazione di Pearson – anche detto coefficiente di correlazione lineare tra due variabili esprime la possibile relazione lineare tra le stesse. L'espressione dell'indice di correlazione è la seguente $\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$, risultante dal rapporto tra la covarianza delle variabili X e Y e il prodotto delle due deviazioni standard al denominatore; può assumere valori compresi nell'intervallo [-1; +1], in particolare:

- +1 si ha perfetta correlazione positiva
- 0 assenza di correlazione lineare, potrebbe essere di altro tipo
- -1 perfetta correlazione lineare negativa

Per rendere l'analisi visiva più comprensibile è stata creata una heatmap; questa è stata programmata per far sì che le metriche altamente correlate tra loro fossero raggruppate in cluster nella stessa area della Heatmap, di cui è mostrata una sezione.

Tabella 2 Sezione Heatmap originata dalla correlazione tra le metriche del dataset

	DB Block Gets Per Sec	Network Traffic Volume Per Sec	User Commits Per Sec	User Transaction Per Sec	DB Block Changes Per Sec	Recursive Calls Per Sec	Open Cursors Per Sec	Total Parse Count Per Sec	Txns Per Logon	User Rollbacks Per Sec	User Calls Per Sec	Executions Per Sec	Enqueue Requests Per Sec	Redo Writes Per Sec
DB Block Gets Per Sec	1,00	0,23	0,24	0,23	0,97	0,90	0,28	0,28	0,21	0,21	0,24	0,47	0,70	0,23
Network Traffic Volume Per Sec	0,23	1,00	1,00	1,00	0,25	0,55	0,82	0,97	0,90	0,98	0,99	0,67	0,71	0,73
User Commits Per Sec	0,24	1,00	1,00	1,00	0,26	0,56	0,86	0,96	0,90	0,97	1,00	0,71	0,71	0,74
User Transaction Per Sec	0,23	1,00	1,00	1,00	0,25	0,54	0,80	0,97	0,90	0,99	0,99	0,65	0,71	0,73
DB Block Changes Per Sec	0,97	0,25	0,26	0,25	1,00	0,85	0,30	0,27	0,23	0,22	0,27	0,49	0,61	0,24
Recursive Calls Per Sec	0,90	0,55	0,56	0,54	0,85	1,00	0,61	0,60	0,49	0,50	0,56	0,71	0,88	0,47
Open Cursors Per Sec	0,28	0,82	0,86	0,80	0,30	0,61	1,00	0,78	0,72	0,71	0,88	0,95	0,58	0,70
Total Parse Count Per Sec	0,28	0,97	0,96	0,97	0,27	0,60	0,78	1,00	0,88	0,97	0,96	0,62	0,77	0,71
Txns Per Logon	0,21	0,90	0,90	0,90	0,23	0,49	0,72	0,88	1,00	0,89	0,89	0,58	0,65	0,66

User Rollbacks Per Sec	0,21	0,98	0,97	0,99	0,22	0,50	0,71	0,97	0,89	1,00	0,96	0,54	0,71	0,70
User Calls Per Sec	0,24	0,99	1,00	0,99	0,27	0,56	0,88	0,96	0,89	0,96	1,00	0,74	0,70	0,75
Executions Per Sec	0,47	0,67	0,71	0,65	0,49	0,71	0,95	0,62	0,58	0,54	0,74	1,00	0,57	0,62
Enqueue Requests Per Sec	0,70	0,71	0,71	0,71	0,61	0,88	0,58	0,77	0,65	0,71	0,70	0,57	1,00	0,54
Redo Writes Per Sec	0,23	0,73	0,74	0,73	0,24	0,47	0,70	0,71	0,66	0,70	0,75	0,62	0,54	1,00

Questo procedimento ha accelerato l'iter esplorativo, permettendo di focalizzarsi in un primo tempo sui cluster formatisi per poi proseguire con le metriche correlate a poche altre e, infine, metriche aventi una correlazione alta solo con un'altra del dataset.

È stato necessario stabilire un valore ρ_{MAX} massimo indice di correlazione di una metrica per essere considerata indipendente e significativa; le metriche aventi indici superiori sono state sottoposte all'analisi qualitativa esplorando i valori delle metriche tramite la visualizzazione grafica. Il valore è stato stabilito $\rho_0 = 0{,}75$.

Il procedimento ha visto i seguenti passaggi:

- Trovare tutte le celle della heatmap aventi $\rho = \rho_0$
- Selezionare una cella e la coppia di metriche coinvolte
- Confrontare i grafici della coppia correlata ponendo l'attenzione su:
    - Andamento generale
    - Presenza di punti 'anomali' rispetto all'andamento negli stessi TIME_STAMP
- Si ripete per tutte le celle
- Si valuta la similarità a livello di coppia, se i grafici sono piuttosto simili, la soglia $\rho_{MAX} = \rho$ può essere considerata quella definitiva. Altrimenti, valore della soglia è incrementato come segue $\rho = \rho_0 + 0{,}01$

In questa fase sono state confrontate iterativamente più di 20 correlazioni tra le metriche, fino a individuare in $\rho_{MAX} = 0{,}79$ il valore soglia di correlazione come mostrato, a scopo

esemplificativo, dalla coppia di plot, rispettivamente Recursive Calls Per Txn e Enqueue Waits Per Txn, appartenenti a rispettivamente a CPU e ENQUEUE.

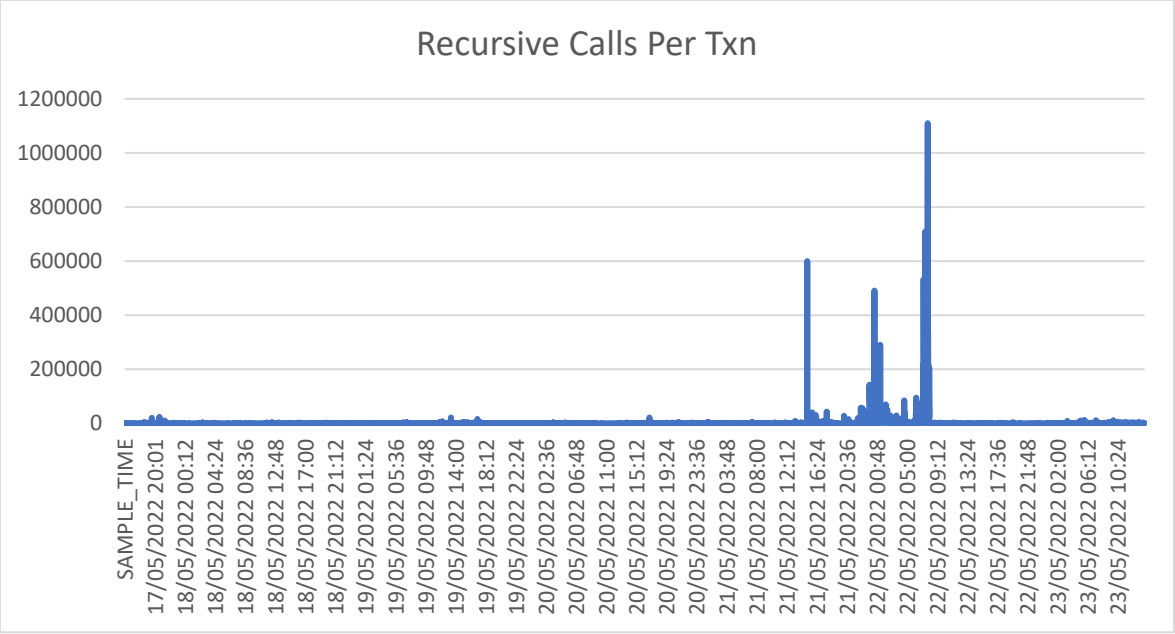

Figura 24 Metrica Recursive calls Per Tnx

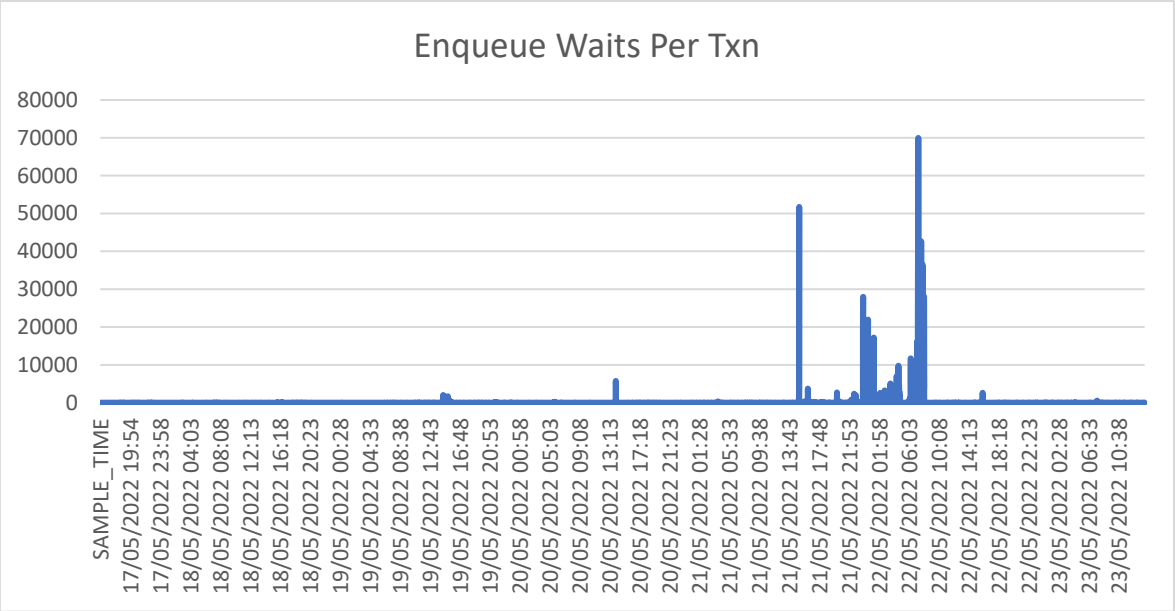

Figura 25 Enqueue Waits Per Txn

Le metriche del dataset sono state filtrate con valore ρ_{MAX} per passare allo step successivo della Feature Selection risultando in un totale di 80 metriche da esaminare, suddivise in miniscluster composti da un massimo di 10 metriche a un unico indice di correlazione

legante due metriche. Partendo dai cluster più consistenti, sono state ricercate le informazioni riguardanti le metriche che ne fanno parte. Sulla base delle informazioni raccolte e la forza della correlazione lineare della metrica con le altre, si è scelta quella più rappresentativa.

Tabella 3 Cluster correlazione Physical Read Total Bytes Per Sec

Physical Read Total Bytes Per Sec	**Physical Reads Per Sec**	0,999999
	Physical Reads Direct Per Txn	0,814096
	Long Table Scans Per Txn	0,811787
	I/O Megabytes per Second	0,999981
	Physical Reads Direct Per Sec	0,996419
	Physical Read Bytes Per Sec	0,999999
	Cell Physical IO Interconnect Bytes	0,999966
	Physical Reads Per Txn	0,816744
	Physical Read IO Requests Per Sec	0,894082

Nel caso più semplice la scelta è stata piuttosto immediata; tuttavia, per alcuni cluster si sono riscontrate alcune correlazioni concatenate. Un esempio può essere il seguente.

Tabella 4 Cluster correlazione Recursive Calls Per Txn

Recursive Calls Per Txn	**Enqueue Waits Per Txn**	0,7969
	Enqueue Timeouts Per Txn	0,9833
	Enqueue Requests Per Txn	0,9882

Tabella 5 Cluster correlazione Enqueue Request Per Tnx

Enqueue Requests Per Txn	**Recursive Calls Per Txn**	0,9882
	Enqueue Timeouts Per Txn	0,9960

Tabella 6 Cluster correlazione Enqueue Timeout Per Txn

Enqueue Timeouts Per Txn	**Recursive Calls Per Txn**	0,9833
	Enqueue Requests Per Txn	0,9960

Tabella 7 Cluster correlazione Waits Per Txn

Enqueue Waits Per Txn	**Recursive Calls Per Txn**	0,7969

La metrica Recursive Calls Per Txn è correlata singolarmente a Enqueue Waits Per Txn e contemporaneamente legata ad altre metriche in un cluster – Enqueue Rquest Per Txn e

Enqueue Timeouts Per Txn. Sebbene il valore 0,7969 sia inferiore all'indice di correlazione delle altre metriche si è scelto di selezionare Recursive Calls Per Txn e tenere comunque in considerazione l'informazione.

5.3.2 Varianza

La dimensionality reduction può anche essere usata per la riduzione dei rumori, data visualization, analisi cluster, o come step intermedio per facilitare analisi di altro tipo.

La prima parte della feature selection si conclude con un dataset composto di 80 metriche, la metà rispetto al dataset di partenza. Pur essendo inferiori in numero, l'approccio multivariato dei modelli ha evidenziato quanto i diversi andamenti delle metriche abbiano distorto in termini performativi gli output. Si è provveduto ad aggiungere un ulteriore step di Dimensionaly Reduction "vero e proprio".

La varianza è spesso considerata uno dei metodi più semplici e rapidi per ridurre la dimensionalità di un dataset. Mentre nella fase precedente sono state scartate le metriche nulle e a valori costanti, adesso sono prese in considerazione le feature i cui valori non cambiano in modo evidente da osservazione a osservazione, apportando poca informazione allo studio. La varianza dipende molto dai domini, per questo motivo i dati vanno normalizzati: questo passaggio è già stato effettuato nel punto 5.2.

Tra i punti di forza dell'applicazione della varianza è l'assunto che la bassa varianza sia sintomo del fatto che quelle feature non cambino né aggiungano informazione; in ogni caso il confronto con gli esperti di dominio è un passaggio non trascurabile.

L'analisi della varianza ha interessato le 80 metriche restanti ed è stata prevalentemente effettuata tramite visualizzazione dell'andamento dei segnali, tenendo in considerazione, oltre al valore della varianza, anche l'estensione del dominio.

Le metriche presentano tre andamenti principali mostrati nei grafici sottostanti:

- aventi l'asse delle ascisse sullo 0
- aventi l'asse delle ascisse (di riferimento) su un numero positivo, coincidente con il valore massimo

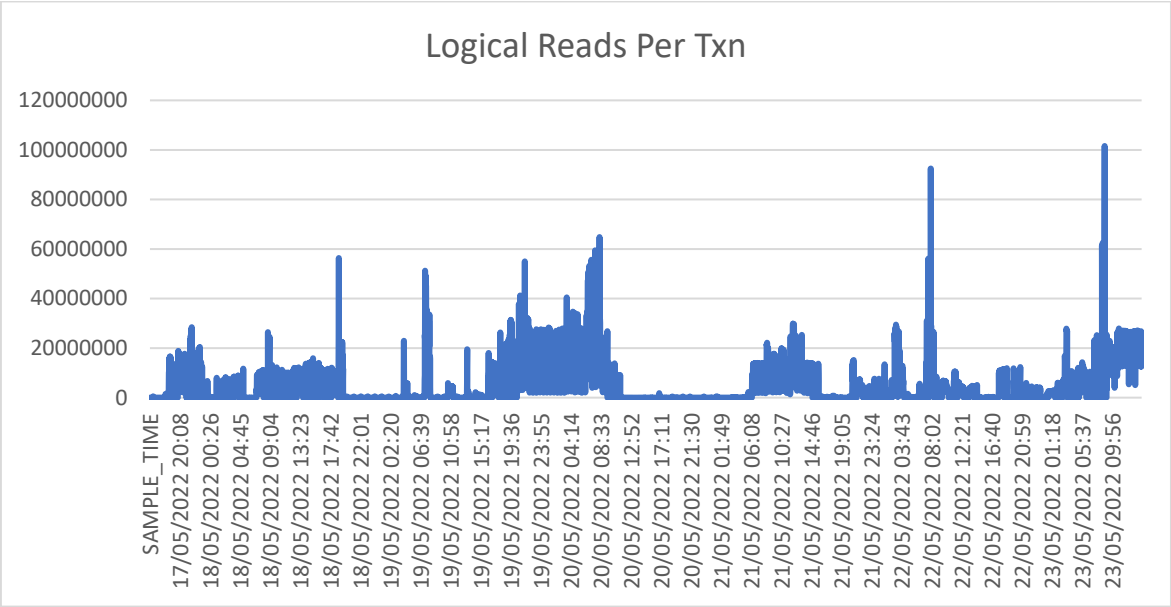

Figura 26 Metrica Logical Reads Per Txn

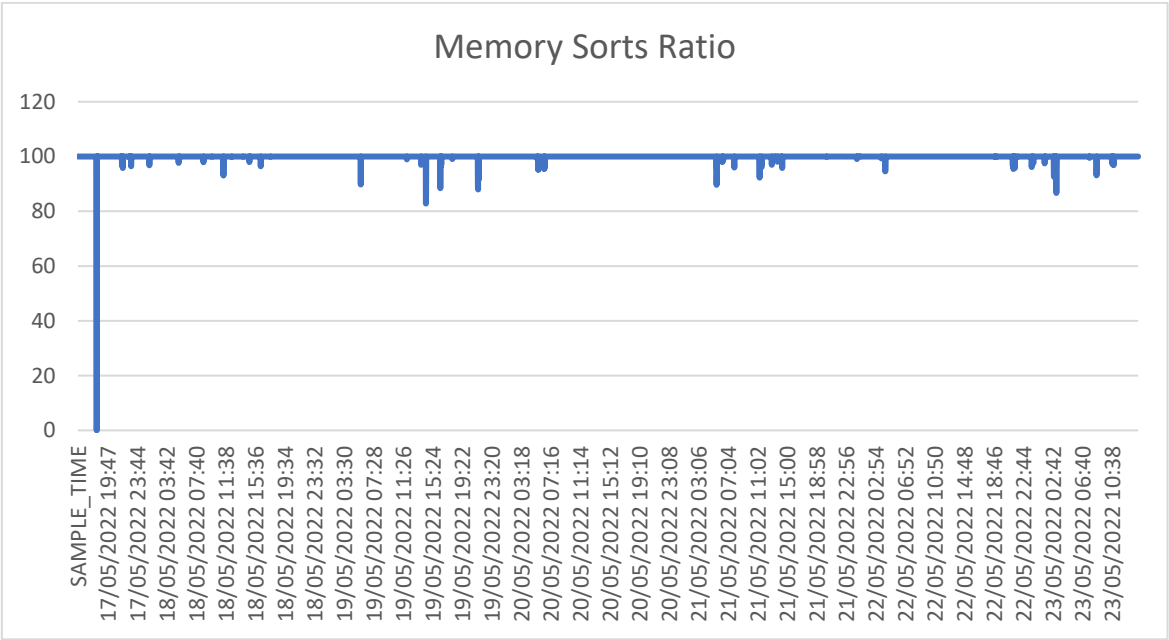

Figura 27 Metrica Sorts Ratio

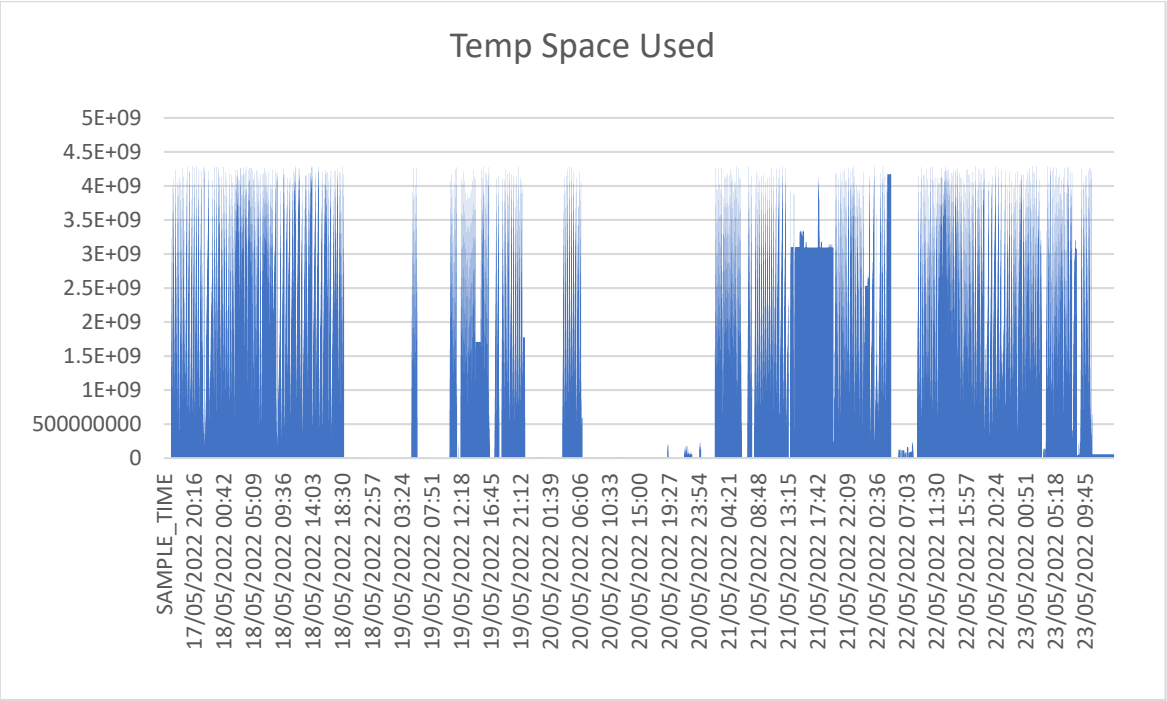

Figura 28 Metrica Temp Space Used

Il dominio è stato tenuto in considerazione come primo parametro per ridurre ulteriormente la dimensione del dataset; non sono state coinvolte le label. Il dataset è stato suddiviso in tre intervalli come si evince dalla tabella, numerose metriche presentano un dominio contenuto e, nonostante presentino una variabilità interna, non sono state reputate rilevanti ai fini dello studio.

Le features sono anche state classificate evidenziando la varianza e confrontandola con il valore massimo che la varianza poteva assumere.

Tabella 8 Legenda Tabella 9

Dominio	
	0-10
	11-999

Varianza	
	0-1
	2-100
	101-1000

Tabella 9 Elenco metriche Filtrate dalla Correlazione

Metrica	**Dominio**	**Varianza**
Active Serial Sessions	1152	8992,26
Average Active Sessions	4776,29961	15532,19
Average Synchronous Single-Block Read Latency	5420	8174,95

Background Checkpoints Per Sec	0,0665779	0,00
Background CPU Usage Per Sec	325,828017	202,97
Background Time Per Sec	33,1719534	2,96
Branch Node Splits Per Sec	0,30467163	0,00
Branch Node Splits Per Txn	1,28571429	0,00
Buffer Cache Hit Ratio	107,632677	44,63
Consistent Read Gets Per Sec	1875890,58	7997,23
CPU Usage Per Txn	1263100	368918715,46
CR Undo Records Applied Per Sec	454027,264	3746040679,48
Current OS Load	221,439453	700,75
Cursor Cache Hit Ratio	27548,1812	35468,75
Database CPU Time Ratio	346,607501	1875,81
DB Block Changes Per Txn	2527305	7540831242,31
DB Block Gets Per User Call	103837,947	16214762,52
DBWR Checkpoints Per Sec	194,123523	11,12
DDL statements parallelized Per Sec	0,15233582	0,00
Disk Sort Per Sec	2,77915633	0,00
Disk Sort Per Txn	2,5	0,01
Enqueue Timeouts Per Sec	22114,1553	1939282,01
Enqueue Waits Per Sec	2940,54503	38520,61
Execute Without Parse Ratio	99,4484211	866,25
Executions Per User Call	4402,79032	10533,05
Full Index Scans Per Sec	26,26498	0,15
Full Index Scans Per Txn	36	0,43
Hard Parse Count Per Sec	82,689747	8,38
Hard Parse Count Per Txn	792	297,73
Host CPU Usage Per Sec	464,85745	27248,29
I/O Requests per Second	146256,652	11177979,92
Leaf Node Splits Per Sec	55,0778605	9,37
Leaf Node Splits Per Txn	232,428571	8,96
Library Cache Hit Ratio	185108700	401090877357648,00
Library Cache Miss Ratio	100	4,63
Logical Reads Per Txn	101582892	68560191405473,00
Logons Per Sec	15,2629827	0,52
Logons Per Txn	390	162,50
Long Table Scans Per Sec	46,7376831	5,35
Memory Sorts Ratio	100	1,42
PGA Cache Hit %	100	124,08
Physical Read Total Bytes Per Sec	3866980209	1516831262109670000,00
Physical Read Total IO Requests Per Sec	146253,876	11552712,18
Physical Reads Direct Lobs Per Sec	594,092195	173,41
Physical Reads Direct Lobs Per Txn	2951	1262,24

Physical Write IO Requests Per Sec	2590,52929	114277,61
Physical Write Total IO Requests Per Sec	3941,6209	327594,62
Physical Writes Direct Lobs Per Txn	38,5	0,34
Physical Writes Direct Lobs Per Sec	8,3677686	0,02
Physical Writes Direct Per Sec	8005,7433	301558,91
Physical Writes Per Txn	428332	406184359,50
PQ QC Session Count	1	0,00
PX operations not downgraded Per Sec	0,15233582	0,00
Recursive Calls Per Sec	50104,4563	37552927,00
Recursive Calls Per Txn	1110014	642934355,40
Redo Allocation Hit Ratio	100	3,55
Redo Writes Per Sec	907,871526	19054,60
Response Time Per Txn	15058775,1	144113494506,52
Row Cache Hit Ratio	100	2,45
Rows Per Sort	133793279	4108735691867,97
Session Limit %	66,6666667	125,59
Shared Pool Free %	56,1686611	97,38
Soft Parse Ratio	100	34,11
SQL Service Response Time	10610,3782	46697,31
Temp Space Used	4293918720	21900206770557400000,00
Total Index Scans Per Txn	1076124,5	2784713469444790,00
Total Parse Count Per Txn	40811	591716,26
Total PGA Allocated	4294958080	1413103792057450000,00
Total PGA Used by SQL Workareas	2893726720	297813030336617000,00
Total Sorts Per User Call	1137,02941	350,73
Total Table Scans Per Sec	1427,51745	19404,59
Total Table Scans Per Txn	20166,3333	548823,02
Total Table Scans Per User Call	3186	3905,91
User Calls Per Sec	7140,12972	1462025,86
User Calls Per Txn	5836	167905,01
User Calls Ratio	66,1449275	185,22
User Commits Percentage	100	617,98
User Rollback UndoRec Applied Per Sec	21817,2431	273569,96
VM in bytes Per Sec	24103132,9	194487354719,55
VM out bytes Per Sec	6205923,46	48137015209,55

Dalla tabella si può notare come in quasi tutti i casi in cui è presente un dominio limitato, sia presente anche una varianza contenuta. L'ulteriore esplorazione dei dati tramite plot ha evidenziato come alcune metriche, pur avendo ampi dominio e variabilità, possedevano un andamento che rendeva complicata rilevare l'eventuale presenza di anomalie, presentando valori elevati piuttosto frequenti. Si è pertanto deciso, dopo un confronto con gli esperti di dominio, di scartare tali metriche per non alterare il contenuto informativo delle feature restanti, le quali ammontano a un totale di 38.

Tabella 10 Metriche Filtrate Dalla Varianza

Metrica	LABEL
Consistent Read Gets Per Sec	CACHE
CR Undo Records Applied Per Sec	CACHE
Logical Reads Per Txn	CACHE
Total PGA Used by SQL Workareas	CACHE
VM in bytes Per Sec	CACHE
VM out bytes Per Sec	CACHE
CPU Usage Per Txn	CPU
Recursive Calls Per Sec	DEBUG
Recursive Calls Per Txn	DEBUG
Enqueue Timeouts Per Sec	ENQUEUE
Enqueue Waits Per Sec	ENQUEUE
Average Synchronous Single-Block Read Latency	I/O
DB Block Changes Per Txn	I/O
DB Block Gets Per User Call	I/O
I/O Requests per Second	I/O
Physical Read Total Bytes Per Sec	I/O
Physical Read Total IO Requests Per Sec	I/O
Physical Reads Direct Lobs Per Txn	I/O
Physical Write IO Requests Per Sec	I/O
Physical Write Total IO Requests Per Sec	I/O
Physical Writes Direct Per Sec	I/O
Physical Writes Per Txn	I/O
Active Serial Sessions	SQL
Average Active Sessions	SQL
Cursor Cache Hit Ratio	SQL
Executions Per User Call	SQL
Library Cache Hit Ratio	SQL
Response Time Per Txn	SQL
Rows Per Sort	SQL
SQL Service Response Time	SQL
Total Index Scans Per Txn	SQL
Total Parse Count Per Txn	SQL
Total Table Scans Per Sec	SQL
Total Table Scans Per Txn	SQL
Total Table Scans Per User Call	SQL
User Calls Per Sec	USER
User Calls Per Txn	USER
User Rollback UndoRec Applied Per Sec	USER

Considerato il numero ridotto di metriche presenti nel dataset, si è reso necessario un altro step di data visualization : le feature sono state raggruppate per LABEL, precedentemente

tralasciate, e visualizzate in blocchi evidenziando come alcune di esse, pur presentando un valore di correlazione complessivo inferiore alla soglia di correlazione individuata al punto 5.3 mostravano però una forte correlazione negli istanti in cui i picchi sono presenti. Ne sono un esempio le metriche Enqueue Timeouts Per Sec e Enqueue Waits Per Sec (coefficiente di correlazione pari a 0,73) aventi la LABEL ENQUEUE.

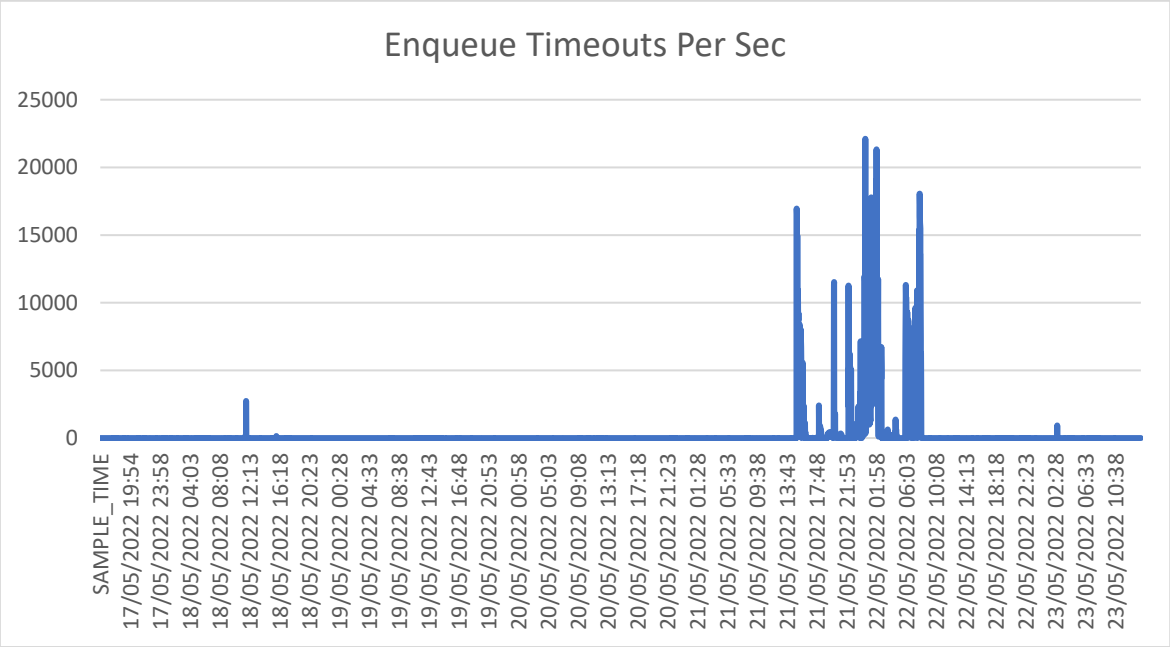

Figura 29 Metrica Enqueue Timeouts Per Sec

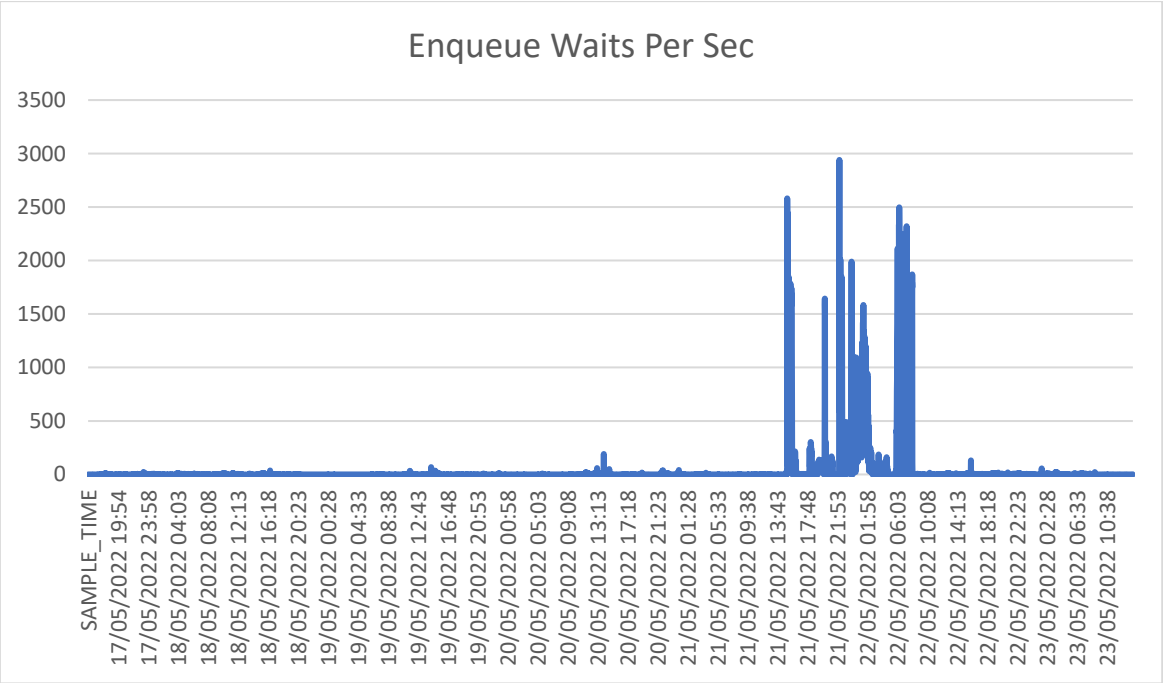

Figura 30 Metrica Enqueue Waits Per Sec

Dopo un confronto con gli esperti di dominio, la metrica Enqueue Waits Per Sec è stata stata scelta per una maggiore ampiezza di dominio. Seguendo questo iter, sono state scartate VM in Bytes Per Sec e DB Block Cangges Per Txn.

6. Analisi dati

L'analisi dei dati e la loro interpretazione sono gli ultimi passaggi del KDD; includono, nel complesso dei procedimenti e metodologie in grado di estrarre dai raw data conoscenza che sarà in seguito impiegata per la maggiore comprensione del problema e l'elaborazione di soluzioni quanto più mirate e adattabili al contesto preso in esame.

Analisi dei dati

Nel capitolo 3 (stato dell'arte) sono state evidenziate le differenze tra approccio supervisionato e non supervisionato. Un'altra distinzione possibile nel machine learning è tra approccio univariato e multivariato. Un approccio univariato avrebbe consentito una maggiore precisione a livello di identificazione delle anomalie pur tuttavia non garantendo una semplicità a livello computazionale poiché il modello avrebbe dovuto essere applicato a ogni feature singolarmente. Scegliendo un approccio multivariato si ottimizza il tempo di esecuzione, si è in grado di individuare dei pattern nei dati e, infine, una migliore e più approfondita comprensione dello scenario presente.

Conclusesi le fasi di data collection e preprocessing, il dataset è stato utilizzato ed elaborato tramite i modelli di Isolation Forest e z-Score for anomaly detection. In questo capitolo sono mostrati e analizzati i risultati in output dei singoli algoritmi e il loro successivo confronto.

Descrizione dei risultati/output

La prima analisi è stata una classificazione operata per mezzo dell'algoritmo di Isolation Forest. Come menzionato precedentemente, i dati sono stati raccolti, trasformati per essere ulteriormente resi adattabili alla fase di analisi tramite la fase di preprocessing che ha eliminato ulteriori metriche ridondanti o correlate tra loro.

Isolation forest non necessita di parametri in input, non è necessario quindi il tuning degli stessi, né di una loro selezione.

L'analisi eseguita si è focalizzata su due modalità di input del dataset: in primis, come un unicum e successivamente, sono state considerate le label delle metriche.

Nelle pagine a seguire saranno mostrati i plot delle metriche e le considerazioni derivate dalla visualizzazione dei risultati ottenuti. È stata eseguita pertanto/inoltre una nuova analisi

tenendo in considerazione dei sottoinsiemi del dataset in subset seguendo la suddivisione in LABEL.

Il procedimento seguito in questo modello prevede pochi passaggi, alla luce della precedente fase di preprocessing.

1. Suddivisione delle metriche sottoinsiemi sulla base delle LABEL rimanenti
2. Selezione della LABEL i-esama
3. Filtraggio del dataset sulla base della LABEL selezionata ottenendo un subset
4. Isolation Forrest sul subset di metriche
5. Visualizzazione del risultato per ogni metrica
6. Il prodimento è stato iterato per ogni LABEL

È da ricordare che Isolation Forest trascura la dimensione temporale delle metriche, individua solamente i picchi dei valori che le feature assumono; iForest rileva le anomalie sulla semplice condizione di isolamento dei singoli outlier mediante le condizioni poste ai nodi degli alberi.

Altro elemento da evidenziare è il fatto che il diverso numero di outliers in output ad ogni iterazione dell'algoritmo permette di fare una stima della media di anomalie, ma è visivamente complicato visualizzare i risultati; le immagini mostrate rappresentano una delle iterazioni.

Come visto nella Tabella 10 sono presenti delle LABEL – CPU ed ENQUEUE – aventi come rappresentative solo una metrica (Enqueue Waits Per Sec è stata scartatat nel passaggio precedente); queste metriche sono state comunque tenute in considerazione per completezza ai fini dell'analisi.

I grafici presentati di seguito sono stati selezionati a scopo illustrativo cosicché fosse possibile effettuare un confronto qualitativo degli output corrispondenti alle due diverse modalità di input dei dati nell'algoritmo.

In particolare, il:

- BLU: andamento metrica

- VERDE: outlier input dataset
- ROSSO: outlier input subset LABEL

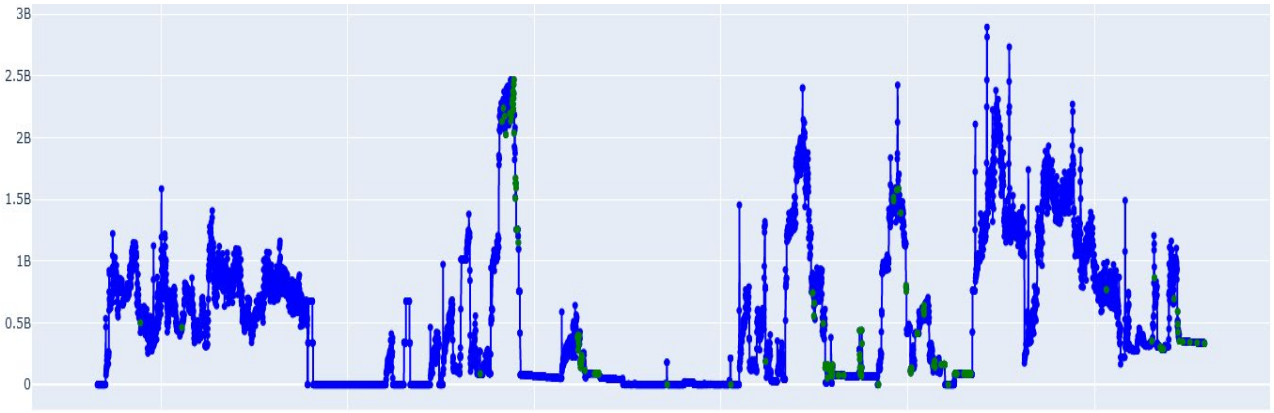

Figura 31 Isolation Forest sulla metrica Total PGA Used by SQL Workareas(CACHE)

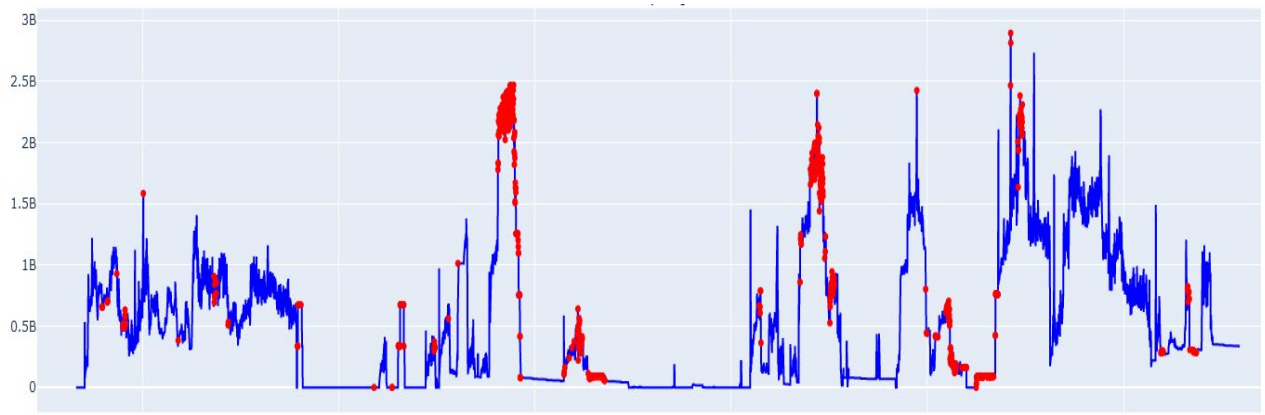

Figura 32 Isolation Forest sulla metrica Total PGA Used by SQL Workareas(CACHE)

La metrica Total PGA Used by SQL Workareas presenta un andamento scostante rendendo più complicato per l'algoritmo rilevare i punti anomali. Nella Figura 31 si nota come l'algoritmo abbia evidenziato un numero di ouliers molto inferiore rispetto alla Figura 32 in cui sono più numerosi, ma non sono inclusi tutti i picchi.

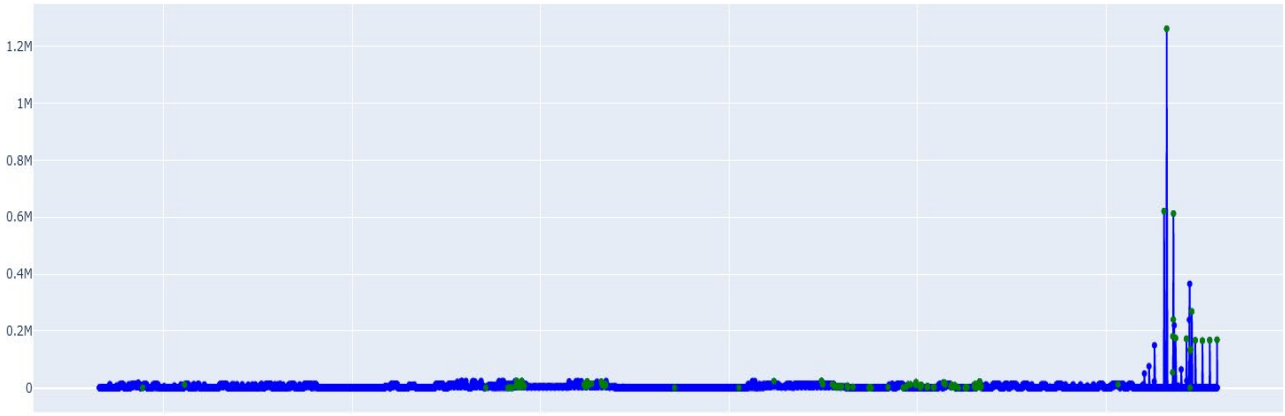

Figura 33 Isolation Forest sulla metrica CPU Usage Per Txn (CPU)

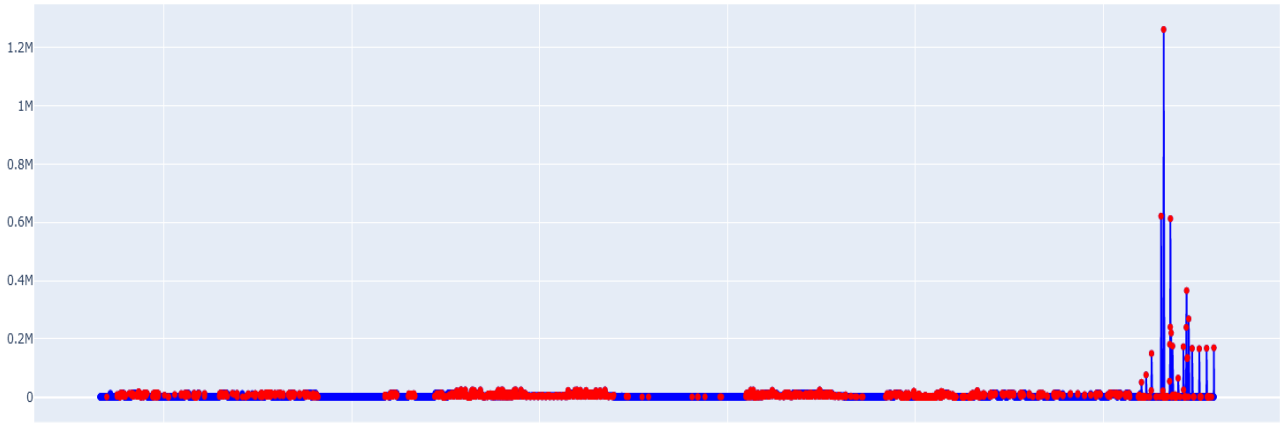

Figura 34 Isolation Forest sulla metrica CPU Usage Per Txn (CPU)

La metrica CPU Usage Per Txn a differenza della precedente presenta nella Figura 33 solo alcuni picchi evidenti mentre, in figura il modello individua come anomalie anche dei punti che si discostano dall'asse delle ascisse, ma non corrispondono a degli outliers, limitatamente a questo grafico.

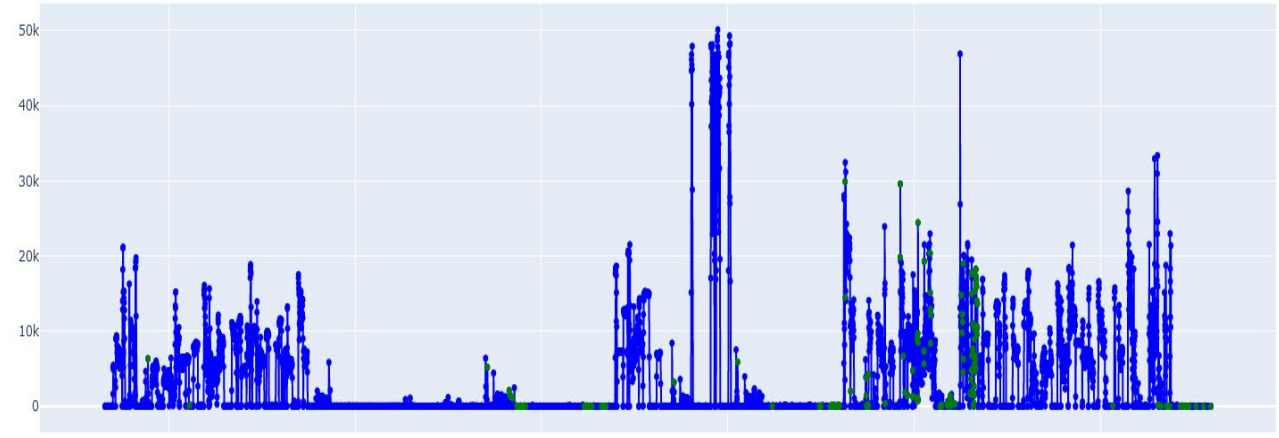

Figura 35 Isolation Forest sulla metrica Recursive Calls Per Sec (DEBUG)

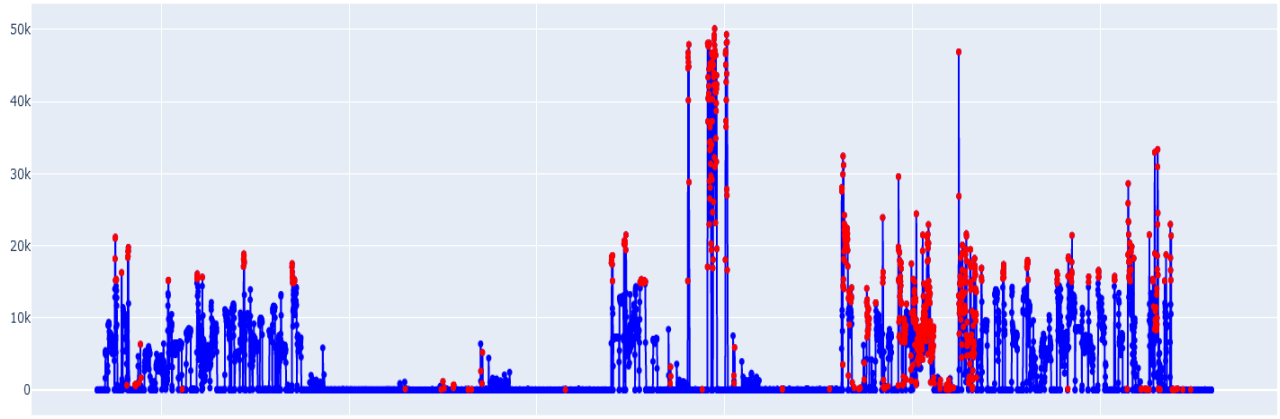

Figura 36 Isolation Forest sulla metrica Recursive Calls Per Sec (DEBUG)

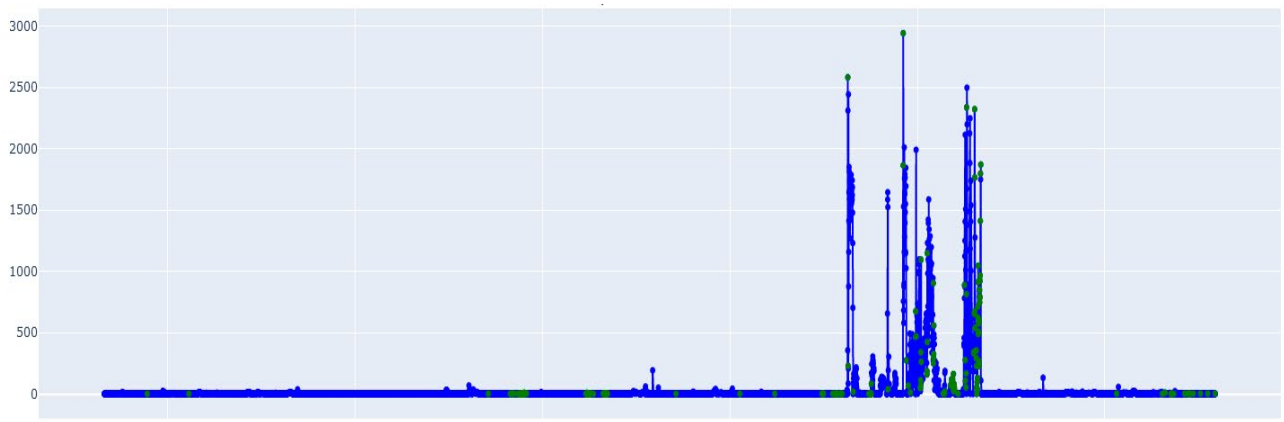

Figura 37 Isolation Forest sulla metrica Enqueue Waits Per Sec (ENQUEUE)

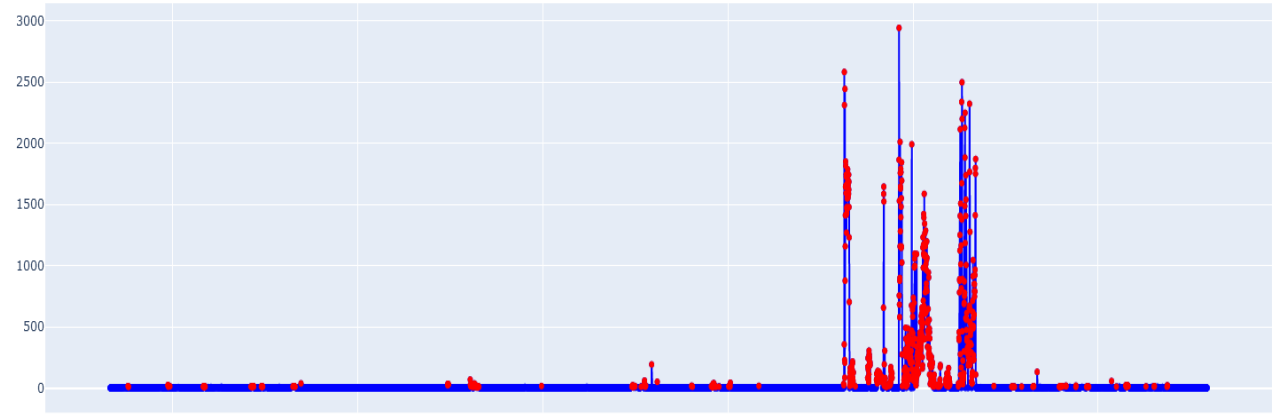

Figura 38 Isolation Forest sulla metrica Enqueue Waits Per Sec (ENQUEUE)

In questo confronto è evidente che, nonostante questa LABEL abbia questa metrica, il modello sia comunque in grado di riconoscere come anomali più punti nel secondo caso.

Tuttavia, è da attenzionare il fatto che sono considerati outliers dei datapoint con un valore non alto.

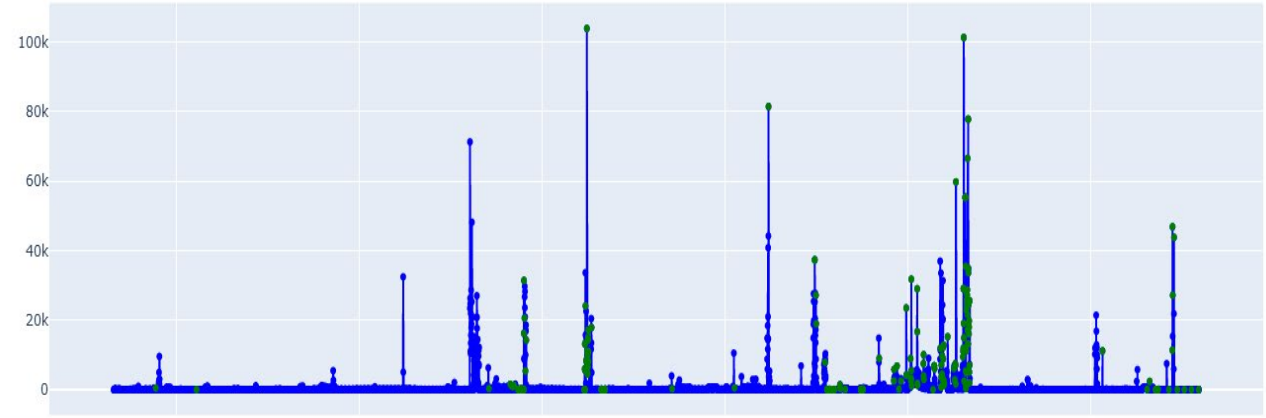

Figura 39 Isolation Forest sulla metrica DB Block Gets Per User Call (I/O)

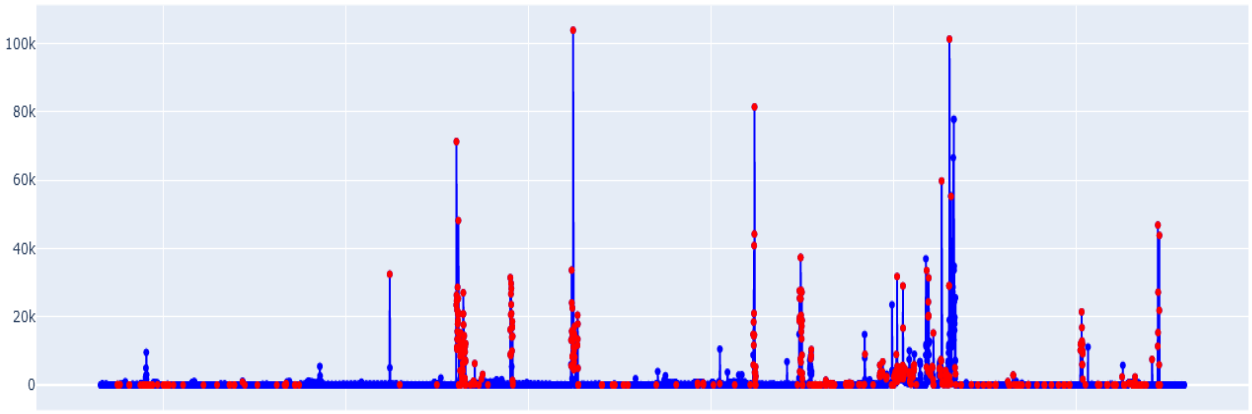

Figura 40 Isolation Forest sulla metrica DB Block Gets Per User Call (I/O)

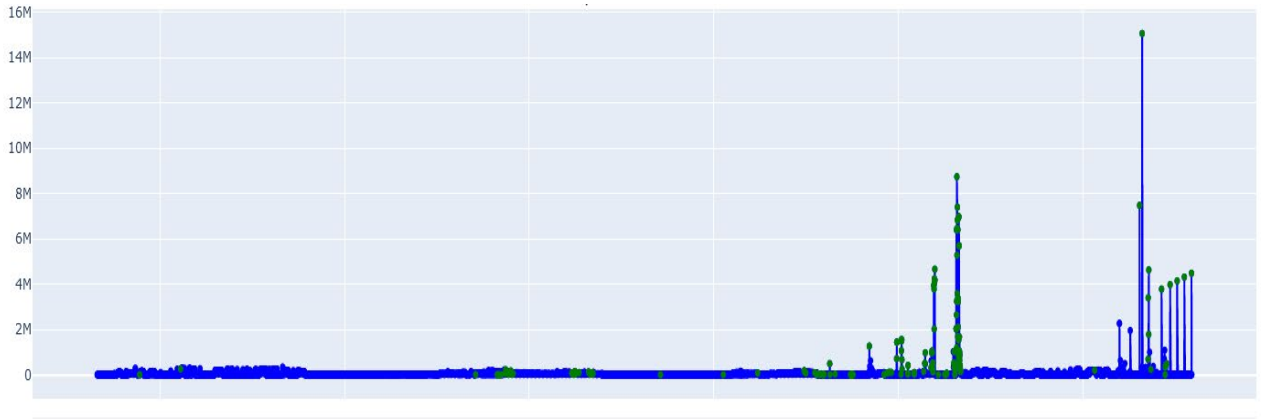

Figura 41 Isolation Forest sulla metrica Response Time Per Txn (SQL)

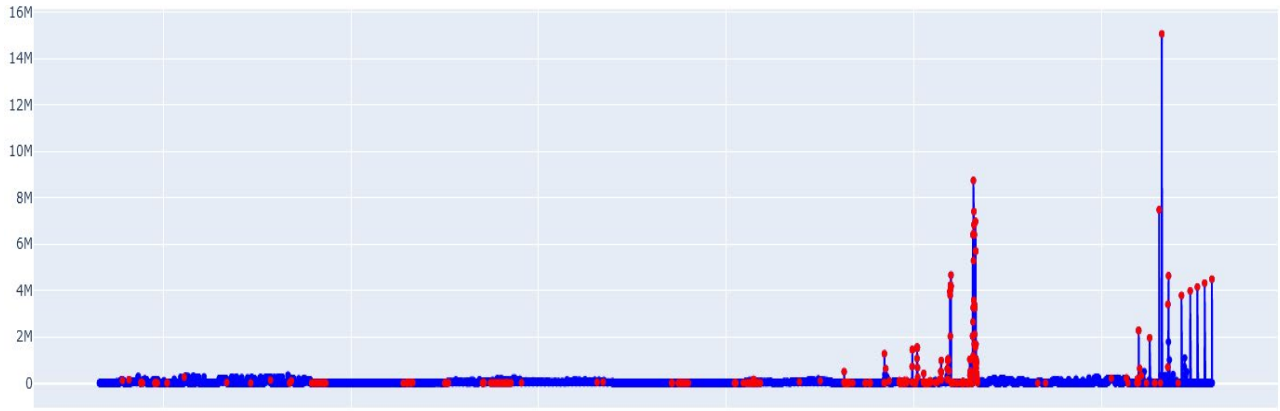

Figura 42 Isolation Forest sulla metrica Response Time Per Txn (SQL)

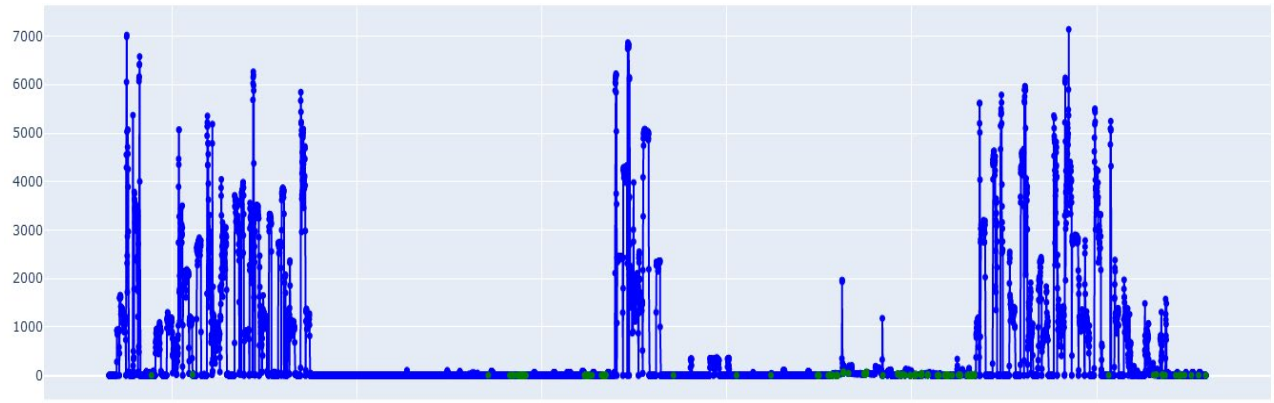

Figura 43 Isolation Forest sulla metrica User Calls Per Sec (USER)

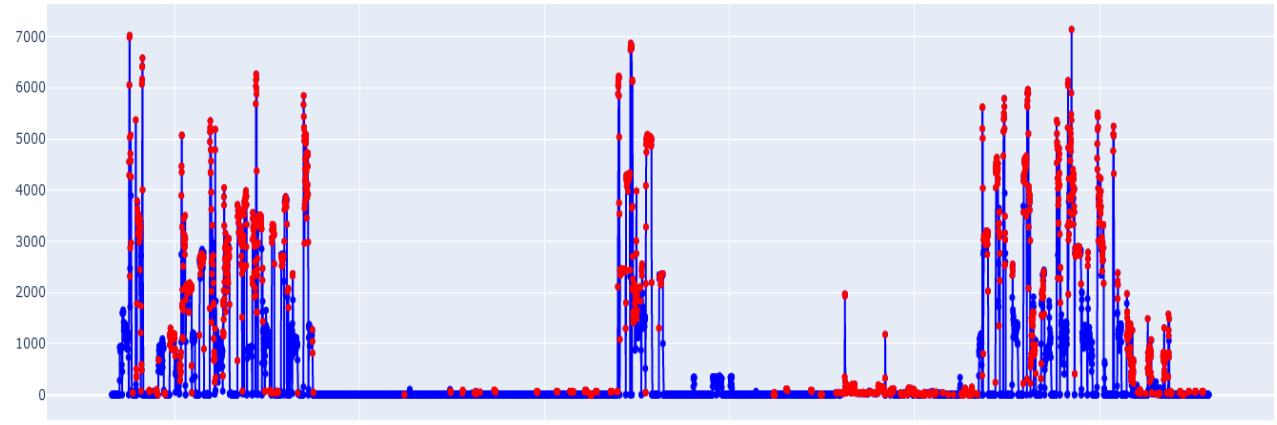

Figura 44 Isolation Forest sulla metrica User Calls per Sec (USER)

Dopo aver concluso l'analisi qualitativa delle features presenti si può affermare che la prima modalità di input garantisce il riconoscimento di punti anomali se questi sono presenti in

percentuale limitata rispetto al totale delle istanze del dataset. In presenza di metriche cin andamenti più variabili, il secondo metodo di input rileva più outliers; il punto di debolezza di quest'ultimo si evidenzia nel caso in cui il subset o dataset abbia poche feature.

7. Conclusioni

In quest'ultimo capitolo sarà eseguita, considerata la conclusione del progetto, un assessment dei risultati ottenuti.

Questo progetto è nato in seno ad uno di più ampia estensione, intrapreso dall'azienda anche con il contributo di progetti di tesi preliminari che hanno permesso di esplorare vari aspetti del problema. L'obiettivo nel lungo termine sarà quello di sviluppare uno strumento in grado di individuare die punti anomali nelle metriche di viste Oracle prese in considerazione con lo scopo aiutare nella gestione e previsione di anomalie che alterano le performance del database.

L'ambiente di sviluppo interno ha giocato un ruolo non indifferente nella tipologia di dati successivamente elaborati e anche nella ricerca di un modello che fosse in grado di evidenziare datapoint anomali anche in andamenti non costanti o, più in generale, regolari.

Attraverso la ricerca svolta è stato evidenziato che è necessario effettuare un corposo lavoro di preprocessing, anche a più step, alternato a una fase di testing sui modelli; questi due passaggi iterati fanno sì che il dataset abbia una elevata variabilità e un alto contenuto informativo.

Ulteriore complessità è stata aggiunta dall'approccio multivariato che, tenendo in considerazione tutte le metriche del dataset in input, evidenziava dei risultati molto differenti da metrica a metrica. È stata quindi tenuta on considerazione la possibilità di suddividere il dataset in subset di metriche che afferissero alla stessa LABE, ponendo in evidenza dei datapoint considerati riferibili a istanti di stress in corrispondenza dei valori di picco nelle feature.

È da sottolineare che, in generale, l'identificazione delle anomalie risulta piuttosto complicata anche in virtù del fatto che non è ad oggi stata ancora data una definizione univoca del termine anomalia. In questa specifica casistica sono stati considerati come anomali dei datapoint corrispondenti a dei picchi nei grafici e, quindi, dei valori eccessivamente al di sopra del consueto andamento generale.

L’algoritmo di classificazione Isolation Forest, utilizzato in un contesto multivariato, ha mostrato come sia possibile avere un numero diverso di outliers nel caso in cui il dataset sia dato in input interamente che nel caso in cui siano stati creati dei modelli a partire dai subset del dataset.

Bibliografia e sitografia

[1] Background Augmentation Generative Adversarial Networks (BAGANs): Effective Data Generation Based on GAN-Augmented 3D Synthesizing - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Supervised-learning-and-unsupervised-learning-Supervised-learning-uses-annotation_fig1_329533120 [accessed 4 Oct, 2022]

[2] https://docs.oracle.com/en/database/oracle/oracle-database/19/comsc/introduction-to-sample-schemas.html#GUID-65A9CECE-E50B-430B-8A52-4393B45209B1

[3] https://www.tpc.org/tpc_documents_current_versions/pdf/tpc-ds_v3.2.0.pdf

Immagine di tpcds - https://medium.com/hyrise/a-summary-of-tpc-ds-9fb5e7339a35

[4] https://en.wikipedia.org/wiki/Feature_selection

[5] https://www.geeksforgeeks.org/introduction-to-pandas-in-python/

[6] https://numpy.org/

[7] https://scipy.org/

[8] https://scikit-learn.org/stable/index.html

[9]https://docs.oracle.com/en/database/oracle/oracle-database/12.2/refrn/VSYSMETRIC.html#GUID-623748C3-F765-4149-8378-F5CDAD59909A

[10] Liu, Fei Tony, Ting, Kai Ming and Zhou, Zhi-Hua. “Isolation forest.” Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on.

[11] Goldstein M, Uchida S (2016) A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. PLoS ONE 11(4): e0152173.

[12] Goldstein, Markus & Uchida, Seiichi. (2016). A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. PloS one. 11. e0152173. 10.1371/journal.pone.0152173. https://doi.org/10.1371/journal.pone.0152173

[13] https://www.dominicgiles.com/index.html