POLITECNICO DI TORINO

Corso di Laurea Magistrale in

INGEGNERIA GESTIONALE

Tesi di Laurea Magistrale

Modelli di Anomaly Detection in Real Time su metriche di performance di un Database Oracle

Relatori

Prof.ssa Tania CERQUITELLI

Dott. Davide BURDESE

Dott. Sonia REGIS

Candidato

Giuseppe URSO

Anno accademico 2022-2023

Indice

Capitolo 1

INTRODUZIONE

1.1 INTRODUZIONE ALL'INFRASTRUTTURA INFORMATICA

L'infrastruttura informatica (componente necessaria per il funzionamento e la gestione degli ambienti IT enterprise; distribuita all'interno di un sistema di cloud computing o nelle strutture di proprietà dell'azienda [1]) (Fig. 1.1) è un sistema complesso composto da numerosi componenti, tra cui hardware e software, che assicura il corretto funzionamento del sistema informativo di un'azienda. Un significativo vantaggio competitivo, prestazioni di servizio migliori e il pieno raggiungimento degli obbiettivi aziendali sono i punti di forza che una un'infrastruttura IT aziendale conveniente riesce garantire. Senza di essa, è impossibile immaginare il lavoro di qualsiasi azienda moderna. I principali requisiti per un'infrastruttura informatica aziendale sono quelli di soddisfare le esigenze del business, garantire un flusso di lavoro stabile e continuo, assicurare l'accessibilità e la sicurezza delle informazioni interne dell'azienda. Per affrontare queste sfide, il mercato IT offre molte tecnologie diverse. Il compito principale di un manager aziendale è selezionare la soluzione

migliore che tenga conto delle specificità dell'organizzazione e che sia in grado di garantire stabilità di sviluppo e facilità d'uso.

Figura 1.1: Infrastruttura Informatica

Per la gestione di infrastrutture informatiche complesse, oggi, si usano strumenti che utilizzano dati in tempo reale per monitorare le prestazioni del sistema, dando possibilità agli operatori responsabili del monitoraggio di intervenire tempestivamente qualora si verificassero delle problematiche. Con l'avvento del Machine Learning e dei Big Data è possibile, mediante l'uso di modelli analitici e approcci predittivi, migliorare questi strumenti per fornire ulteriore supporto agli operatori in situazione critiche. Il ML ha essenzialmente bisogno di un passaggio fondamentale: il suo allenamento, un processo critico per lo sviluppo di modelli di intelligenza artificiale performanti, in grado di risolvere problemi complessi e di prendere decisioni in modo autonomo. Con la necessità, sempre più importante, di ottimizzare e automatizzare le

tecniche di monitoraggio dell'infrastruttura, il ML ha aiutato la ricerca all'individuazione automatica di eventi anomali. Con Anomaly Detection si intende la capacità di identificare elementi o eventi non conformi all'andamento previsto in uno specifico dataset; ha molte applicazioni in diversi contesti, come ad esempio la individuazione di frodi, la rivelazione di intrusioni informatiche, i sistemi di supporto alla diagnosi medica, il marketing e molte altre aree ancora. Questo tipo di analisi si concentra sulla ricerca di pattern che si discostano dai comportamenti normali e che possono indicare la presenza di anomalie o di comportamenti anomali. Grazie all'utilizzo di tecniche di Machine Learning e di algoritmi avanzati, è possibile identificare queste anomalie in modo preciso e tempestivo, aiutando le organizzazioni a prevenire problemi e a migliorare le loro attività.

1.2 COSTESTO AZIENDALE

Questo lavoro di tesi è stato svolto presso Mediamente Consulting S.r.l [2] ., società ci consulenza informatica, che vede come core business la gestione e valorizzazione dei dati aziendali. Fondata nel 2013 da un gruppo di noti esponenti nell'ambito IT e dalla partecipazione di Var Group (gruppo Sesa S.p.A quotato nella borsa italiana), la quale ha acquisito la maggioranza della proprietà nel 2022. Ha sviluppato aree di specializzazione in Corporate Performance Management, Advanced Analytics, Business Intelligence, Data Integration e Management e Infrastruttura tecnologica, mantenendo l'interesse principale verso la gestione, l'aggiornamento e il tuning di sistemi complessi e ingegnerizzati. In questo senso, Mediamente Consulting tende sempre a migliorare ed innovare le tecnologie attualmente utilizzate per il monitoraggio di infrastrutture complesse di medie e grandi aziende. La società adopera una soluzione software entreprise per il monitoraggio delle proprie infrastrutture informatiche. Tale software utilizza la configurazione di soglie critiche per segnalare eventuali problematiche successivamente prese in carico da un team di consulenti

specializzati. Per migliorare l'efficienza dei processi di monitoraggio, l'azienda intende implementare metodi alternativi basati su un approccio proattivo, puntando a prevenire le segnalazioni di problematiche da parte dei clienti tramite ticket e intervenendo prontamente in caso di problemi di performance critici per il business aziendale. Il lavoro di tesi si concentrerà quindi sull'individuazione di questi nuovi metodi di monitoraggio, in grado di garantire una maggiore rapidità e precisione nell'intervento sulle criticità.

1.3 MOTIVAZIONE DELLA RICERCA

Insieme alla società ospitante, è stato prefissato uno studio di fattibilità di una possibile soluzione che riesca a identificare in tempo reale e automaticamente delle anomalie, rispetto a comportamenti normali, riguardanti l'esecuzione di query SQL all'interno di database Oracle. L'idea alla base è quella di riuscire ad implementare il prodotto entreprise, già utilizzato in azienda, con un'intelligenza di Anomaly Detecion, in modo da poter garantire un tempestivo intervento su piccoli problemi evitando gravi danni alla infrastruttura cliente.

1.4 MOTIVAZIONE DELLA RICERCA STRUTTURA TESI

Il workflow (Figura 1.2) della ricerca che vuole essere descritta in questa tesi, bene raffigurato nella figur 2, inizia con due passaggi inerenti allo studio sia delle tecnologie usate oggi per lo studio delle anomalie, seguito da un ampliamento delle conoscenze riguardanti i servizi e i prodotti offerti da Oracle, multinazione texana leader nel settore IT e principale proprietaria dei sistemi ingegnerizzati installati ai clienti. Concluse le fasi di ricerca, ci si è subito riversati nella compilazione di codici (fase

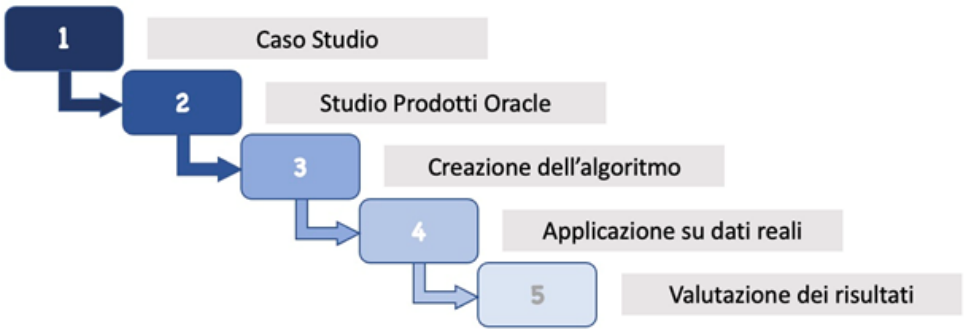

Figura 1.2: workflow

3 della figura 2) atti sia all'estrapolazione e rielaborazione degli innumerevoli dati prestazionali di un database aziendale e la successiva applicazione dei modelli di Anomaly Detection prima studiati. L'accesso a dati reali, provenienti da importanti clienti, ha creato la possibilità di allenare tre diversi algoritmi, di testarli ed individuarne le soglie e gli attributi più adatte al caso studio. Raccolti i risultati finali, la tesi si conclude cercando di delinearne una migliore soluzione tra quelle trovate.

Capitolo 2

STATO DELL'ARTE

2.1 ANOMALY DETECTION

Il processo di identificazione di elementi, eventi o osservazioni, chiamate anomalie o outlier, che presentano dei comportamenti non nomali rispetto alla consuetudine, prende il nome di Anomaly Detection [3]. In passato, le organizzazioni esaminavano manualmente i dati alla ricerca di indizi e informazioni sulle prestazioni dei loro sistemi. Tuttavia, spesso non riuscivano a individuare le cause principali dei problemi. Questo portava a una persistenza del problema e a un continuo rischio per i dati dell'organizzazione. Oggi, il rilevamento delle anomalie si basa principalmente sull'utilizzo del machine learning (ML) [4]. Il ML aiuta a identificare valori anomali che possono essere difficili da individuare manualmente, consentendo di mitigare tali anomalie e proteggere il sistema. Attraverso l'applicazione di algoritmi di apprendimento automatico ai dati, è possibile identificare i modelli e le tendenze che caratterizzano il normale funzionamento del sistema. Questi modelli vengono poi utilizzati per rilevare le anomalie che si discostano dal comportamento tipico. Il machine learning può aiutare a individuare e segnalare eventi anomali in tempo reale, consentendo alle organizzazioni di intervenire tempestivamente per risolvere

i problemi. Complessivamente, l'utilizzo del machine learning per il rilevamento delle anomalie consente alle organizzazioni di migliorare la sicurezza, la stabilità e le prestazioni dei loro sistemi, garantendo una protezione efficace dei dati. Outlier Detection, nome alternativo ad Anomaly Detection, implica varie sfide tra cui definire quali dati costituiscono un comportamento normale e quali un'anomalia:

- **Falsi Positivi e Falsi Negativi**: Il confine tra comportamento normale e anormale non è spesso definito bene, per cui i modelli potrebbero produrre molti falsi positivi e falsi negativi;
- **Aree di domini**: diversificando le aree di dominio, spesso può accadere che un'anomalia in un dominio può essere un comportamento normale in un altro e viceversa;
- **Dataset**: Spesso i set di dati non contengono l'informazione se un dato sia anomalo o meno.

Come si osserva dalla Figura 2.1, il grafico è formato da due macroaree in blu, nelle quali si concentrano la maggior parte dei punti, nella quali i dati presentano pressoché le stesse caratteristiche; si possono quindi considerare questi valori come 'normali'. Diversa è la situazione delle aree colorate in arancio (o1, o2 e o3): infatti i punti contenuti in esse sono molto pochi, e presentano dei comportamenti molto distanti dal comportamento medio, rappresentando quindi dei valori 'anomali'. Il rilevamento di anomalie è un'attività cruciale in vari settori, tra cui finanza, vendita al dettaglio e sicurezza informatica. Tuttavia, non è limitato a questi settori specifici e ogni azienda può trarre vantaggio dall'implementazione di una soluzione di rilevamento di anomalie. Ad esempio, nel settore bancario, il rilevamento di anomalie tramite sistemi di apprendimento automatico è di grande importanza, poiché consente di identificare attività fraudolente e modelli inconsueti. Nel settore della vendita al dettaglio, ad esempio, può essere utilizzato per identificare schemi di acquisto insoliti o transazioni

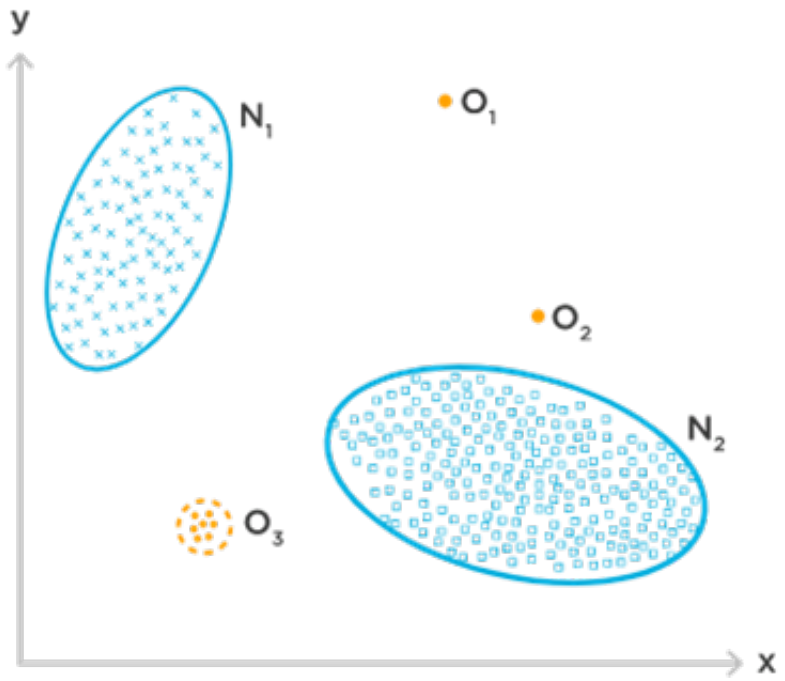

Figura 2.1: Anomaly Detection

sospette, contribuendo a prevenire frodi o perdite. Nell'ambito della sicurezza informatica, il rilevamento di anomalie può individuare attività insolite nella rete o comportamenti anomali degli utenti, consentendo di identificare potenziali attacchi o violazioni della sicurezza. In sintesi, il rilevamento di anomalie svolge un ruolo fondamentale in diversi settori, poiché consente di individuare comportamenti insoliti o eventi anomali. Le soluzioni basate su machine learning possono analizzare i dati in modo efficiente e accurato, consentendo alle aziende di identificare tempestivamente potenziali problemi o minacce, proteggendo i propri interessi e i propri clienti.

2.2 ANOMALIA

Descriviamo ora, prima di approfondire le diverse tecniche, cosa sia un'anomalia. In generale un evento è anomalo se si discosta dal comportamento atteso. Ad esempio,

nel lancio di una moneta l'esito atteso è di ottenere un risultato di testa o croce e una moneta che resta in bilico è un evento alquanto anomalo/improbabile. Il problema è a dir poco complesso perché la definizione stessa di anomalia può essere difficile da dare e può variare da un contesto a un altro [5]. Le anomalie quindi si possono classificare in:

- **Anomalie puntuali**: La forma più semplice di anomalia si presenta quando si vuole determinare se un singolo dato può essere considerato anomalo rispetto a tutti gli altri. Questo è il focus principale della maggior parte delle tecniche di individuazione delle anomalie. Nella figura sottostante (Figura 2.2), è rappresentato un set di dati a due dimensioni. In questo possiamo notare come quasi la totalità delle rilevazioni si possono distribuire lungo la diagonale del grafico (dalla coordinata -4, -3 alla coordinata +4, +3). Da questo dataset si discosta molto un solo valore, che può rappresentare una anomalia puntuale, racchiuso in un quadratino rosso.

- **Anomalie contestuali**: Come accennato in precedenza, la natura di un dato come anomalo o normale dipende dal contesto in cui viene considerato. È importante comprendere che il contesto è determinato dalla struttura dei dati e deve essere definito come parte del problema. Ogni istanza di dati è quindi definita utilizzando due insiemi di attributi:

  - **Attributi di contesto**:Questi attributi vengono utilizzati per determinare il contesto specifico di un dato. Ad esempio, in un set di dati di serie temporali, l'attributo di contesto potrebbe essere il tempo, che stabilisce la posizione di un dato all'interno dell'intera sequenza temporale;
  - **Attributi di comportamento**: Questi attributi definiscono le caratteristiche non contestuali di un dato e sono utilizzati per individuare anomalie all'interno di un contesto specifico. Ad esempio, se stiamo analizzando una

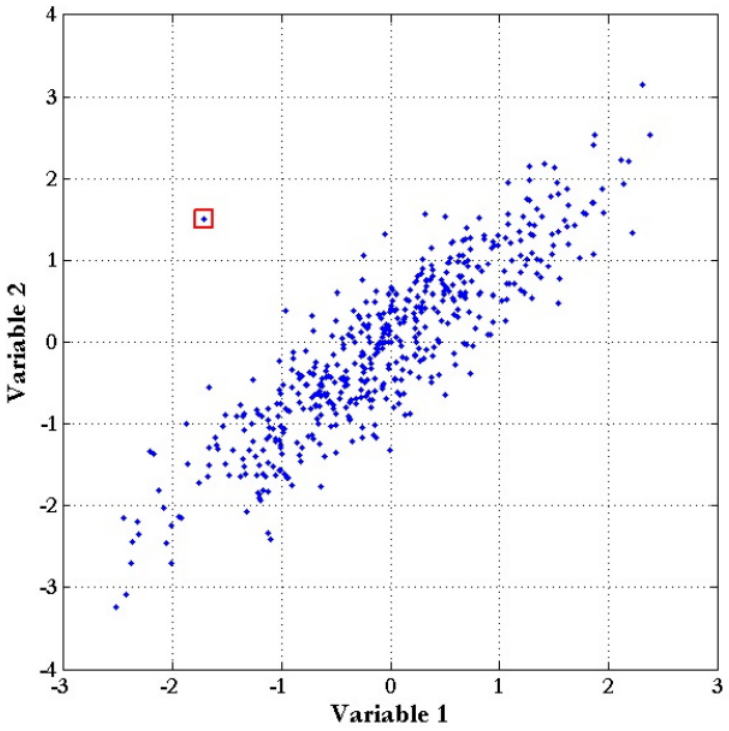

Figura 2.2: Esempio anomalia puntuale in un set a due-dimensioni

serie temporale di temperature, l'attributo di comportamento potrebbe essere la media delle temperature, che ci permette di identificare valori anomali rispetto alla normale distribuzione delle temperature.

Nella Figura 2.3 è illustrata una serie temporale che rappresenta le temperature mensili di un'area specifica nel corso di tre anni. In questo contesto, una temperatura di 35°F potrebbe essere considerata normale durante l'inverno (al tempo t1), ma la stessa temperatura durante l'estate (al tempo t2) potrebbe essere considerata un'anomalia.

- **Anomalie Collettive**: anche note come anomalie di gruppo, si verificano quando un insieme di osservazioni è considerato anomalo rispetto al comportamento generale del resto del set di dati. Queste anomalie possono essere difficili da individuare, poiché richiedono l'identificazione di pattern o strutture in tutto

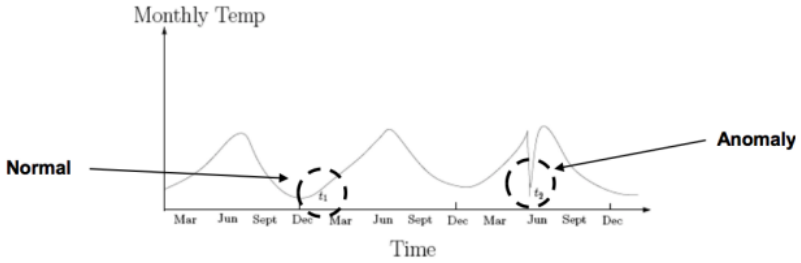

Figura 2.3: Esempio di anomalia contestuale

il set di dati. Ad esempio, in un dataset di dati finanziari, potrebbe essere presente un gruppo di transazioni sospette che, prese individualmente, sembrano normali, ma analizzate collettivamente rappresentano un'anomalia.

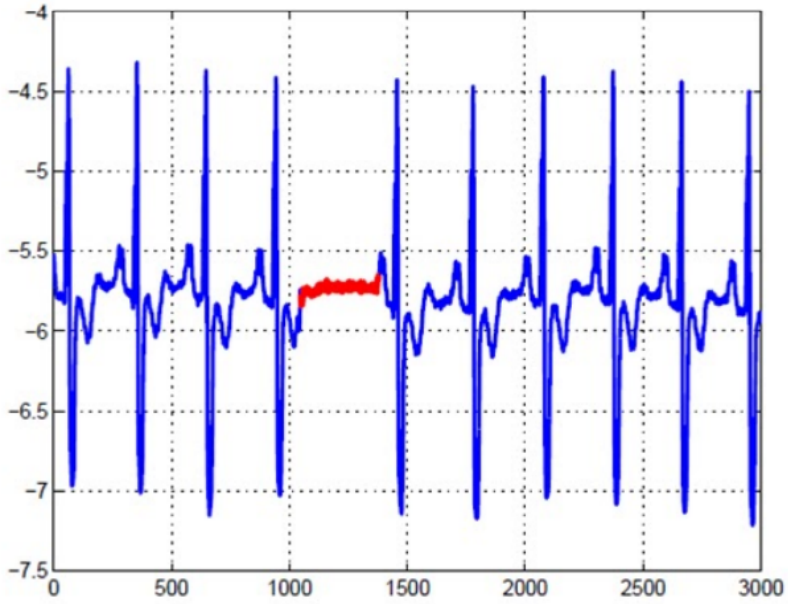

Figura 2.4: Anomalia collettiva corrispondente a una contrazione prematura atriale

Nella Figura 2.4 è presente un grafico di un elettrocardiogramma (ECG). Si può osservare una regione evidenziata in rosso che rappresenta un'anomalia nel segnale. Questa anomalia è rilevata perché il valore del segnale si mantiene costante per un periodo di tempo troppo lungo. È importante notare che il

singolo valore preso isolatamente potrebbe non essere considerato un'anomalia, ma se si verifica una prolungata durata con lo stesso valore, diventa una caratteristica anomala.

2.3 TECNICHE DI OUTLIER DETECTION

Esistono due grandi categorie che differenziano i modelli di Anomaly Detection: la prima si basa sulla natura del dataset che si utilizza; una seconda, invece, li diversifica in base ai metodi di separazione delle anomalie rispetto al resto del dataset.

2.3.1 Apprendimento supervisionato, semi supervisionato e non supervisionato

La Figura 2.5 classifica i diversi modelli di apprendimento in base alla natura del dataset che si utilizza in fare di addestramento del modello.

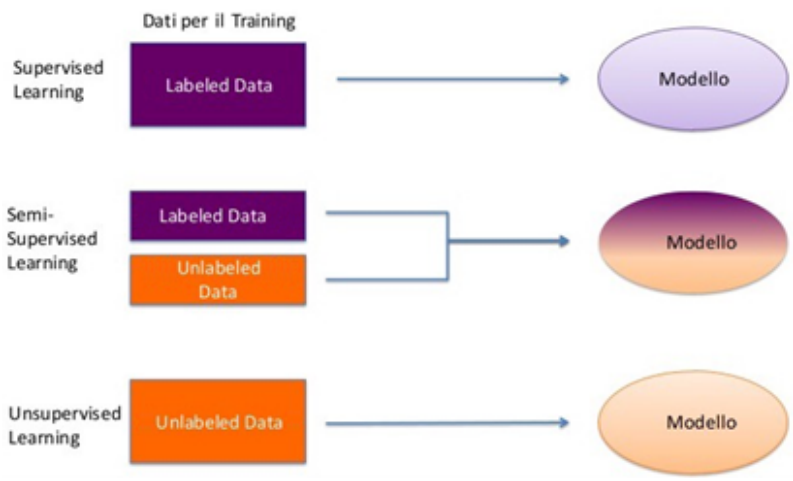

Figura 2.5: Apprendimento supervisionato, semi supervisionato e non supervisionato

- **Apprendimento Supervisionato**: L'apprendimento supervisionato risolve problemi noti e usa una serie di dati etichettati per addestrare un algoritmo ad eseguire compiti specifici. I dati di formazione dovrebbero essere etichettati correttamente se normali o anormali da esperti in materia. L'apprendimento supervisionato permette agli algoritmi di "imparare" dai dati storici/di addestramento e applicarli a input sconosciuti per ricavare l'output corretto. Le principali categorie dell'apprendimento supervisionato sono [6]:
  - **La classificazione**:La macchina viene addestrata alla classificazione dal supervisore tramite l'aggiunta di etichette sui dati in cui giudica il risultato. Ogni etichetta è una classe discreta che identifica il risultato atteso (es. spam o no spam) oppure un giudizio di valore.
  - **La regressione lineare**:Nella regressione lineare il risultato (output) è un valore continuo. Alla macchina spetta il compito di trovare una relazione tra i valori di input (valori descrittivi) e di output.
- **Apprendimento Semi-supervisionato**[**7**]:L'apprendimento semi-supervisionato è una tecnica di apprendimento automatico che colma il divario tra l'apprendimento supervisionato e l'apprendimento non supervisionato sfruttando sia i dati etichettati che quelli non etichettati per migliorare l'accuratezza del modello. In questo contesto, ci sono due approcci principali: l'autoformazione e la co-formazione. Il self-training è un tipo di apprendimento in cui un modello viene inizialmente addestrato su un set di dati etichettati, per poi essere usati per associare delle etichette ai dati ancora sprovvisti. Questo processo viene ripetuto iterativamente, consentendo al modello di apprendere dai dati etichettati e non etichettati in modo incrementale. Il co-training è un altro tipo di apprendimento in cui vengono addestrati due modelli, ciascuno su una vista diversa degli stessi dati. Le due viste possono essere create attraverso diverse rappresentazioni, estrazioni di caratteristiche o prospettive sulle stesse

istanze. In seguito, i due modelli vengono utilizzati per etichettare i dati senza etichettatura, condividendo e confrontando le etichette predette. I punti dati con etichette concordanti tra i due modelli vengono considerati affidabili e aggiunti al set di addestramento. Questo processo viene iterato per migliorare le prestazioni del modello attraverso una collaborazione tra i due modelli.

- **Apprendimento non Supervisionato**:A differenza dell'apprendimento supervisionato, l'apprendimento senza supervisione è utilizzato quando i dati di input non sono etichettati o privi di un corrispondente valore di output e l'algoritmo deve scoprire eventuali relazioni sottostanti. Questo metodo è adatto per individuare modelli nascosti nei dati che non sono immediatamente evidenti a causa di informazioni sovrapposte o di grandi quantità di dati che richiedono l'ausilio di un'elaborazione computazionale. In assenza di dati etichettati, il modello non può essere addestrato su un insieme di dati preparati con il corretto output corrispondente, ma deve identificare autonomamente le differenze o le similitudini fra gli input, individuandone le caratteristiche principali. Grazie alla mancanza di fasi di addestramento e di validazione, la preparazione del dataset di input richiede meno sforzi.

2.3.2 Metodi Statistici, Metodi basati sulla Prossimità e Metodi basati sul Clustering

Una seconda classificazione, come discusso in precedenza, si basa sui metodi utilizzati per la separazione dei campioni anomali rispetto al resto dei dati.

- **Metodi statistici**: si ipotizza che i dati normali vengano generati da processi stocastici, cioè da un insieme di funzioni che evolvono nel tempo, ognuna delle quali è associata ad un determinato elemento dello spazio campione, così che il risultato di un esperimento casuale corrisponde di fatto all'estrazione di una di

queste funzioni. Ne consegue che le anomalie si verificheranno nelle regioni di bassa probabilità del modello.

- **Metodi basati sulla prossimità**: si assume che un dato sia un anomalo se i suoi vicini più prossimi sono lontani nello spazio delle caratteristiche, ovvero se la vicinanza del dato ai suoi vicini si discosta significativamente dalla vicinanza della maggior parte degli altri dati ai loro vicini nello stesso dataset.
- **Metodi basati sul Clustering**:in questo caso, come mostra bene la Figura 2.6, che rappresentate una distribuzione bidimensionale di un dataset, si ipotizza che i dati appartenenti a dai cluster, gruppi o insiemi di oggetti, grandi e densi e non rappresentino delle anomalie. Se invece, il dato non appartiene a nessun cluster, o appartiene a cluster piccoli e poco densi, allora è un dato anomalo.

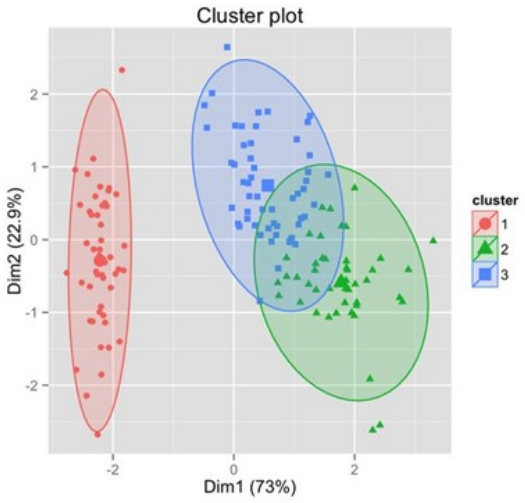

Figura 2.6: Esempio di Clustering

2.4 ALGORITMI

Si vuole ora approfondire brevemente gli algoritmi che verranno presi in considerazione in questo lavoro di tesi, valutandone l'idea di base, i pro e i contro.

- **Z-score**: Lo z-score [8] è una misura statistica che determina quanto è lontano un dato dal resto del dataset. In un termine più tecnico, avendo un dataset che segue una distribuzione Normale, dopo aver standardizzato tutti i valori (riportarli in una distribuzione normale con media centrata nell'origine degli assi e con la stessa deviazione standard), il valore z associato ad un determinato dato ne rappresenta la sua locazione all'interno della nuova distribuzione. La procedura di standardizzazione parte quindi con il calcolo della media dei valori e della deviazione standard:

$$\text{Mean} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$StandardDeviation = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \text{Mean})^2}{N}}$$

  Viene quindi applicata una formula matematica (descritta qui sotto) a tutti i dati.

$$\text{Z-score} = \frac{x - \text{Mean}}{\text{Standard deviation}}$$

  Se un dato assume un valore Z molto grande, che per comodità viene identificato se il valore supera l'1,645 che corrisponde a una percentuale del 95% (preso dalla tavola della distribuzione gaussiana [9]), potrebbe rappresentare una anomalia. Per determinare se un dato è anomalo o meno si deve, quindi, scegliere un intervallo di confidenza, monolaterale o bilaterale (in questo caso viene scelto monolaterale, in quanto, per gli obbiettivi aziendali, sono di interesse solo quei valori che consumano più risorse, o tempo, rispetto al comportamento solito), cioè un intervallo percentuale entro il quale si concentrano i dati considerati normali. Limitazione importante di questo algoritmo è rappresentata dalla sensibilità della media nei confronti di valori molto grandi, che ne possono invalidare il calcolo (del valor medio) e quindi l'intero metodo.

- **Local Outlier Factor**: Il metodo del Local Outlier Factor [10] (Figura 2.7) funziona confrontando la densità di un punto con le densità relative dei suoi

vicini. Se un punto è relativamente meno denso dei suoi vicini, è un possibile outlier. Se il rapporto tra la densità dei vicini e la densità del punto è troppo alto, si ottiene un alto valore di LOF che indica un outlier. Innanzitutto, si

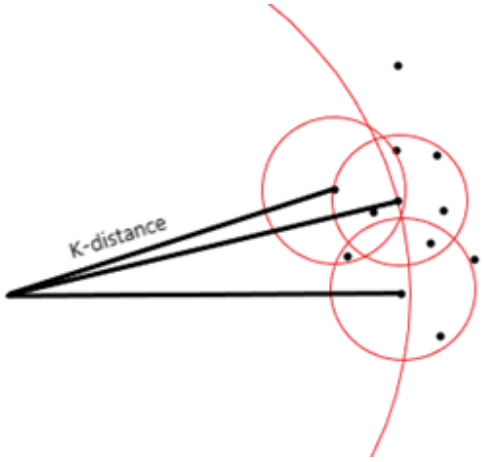

Figura 2.7: Local Outlier Factor

definisce la K-distance di un punto. Questa è intuitivamente la distanza del K-esimo punto dal sui vicino più prossimo:

$$K_{\text{distance}}(A) = \text{Dist}(A, K_{\text{th}} \text{ nearest neighbour})$$

Il K-neighbourhood è costituito dai K punti più vicini a un determinato punto.

$$N_k(A) = \{P \,|\, \text{Dist}(A, P) \leq \text{K-distance}(A)\}$$

La reachability distance definisce quanto sia facile raggiungere un punto da un altro punto. Se un punto A è nel K-vicinato di B, allora è più facile raggiungere A e la distanza di raggiungibilità è inferiore. Per un punto A che non è nel K-vicinato di B, la distanza di raggiungibilità è maggiore.

$$RD_k(A, B) = \max(N_k(B), \text{distance}(A, B))$$

La local reachability density è definita come il reciproco della distanza di raggiungibilità media dei punti intorno a A rispetto ad A:

$$lrd_k(A) = \frac{1}{\sum_{B \in N_k(A)} \frac{RD_k(A,B)}{|N_k(A)|}}$$

Ora che conosciamo la densità di un punto, il fattore di raggiungibilità locale è semplicemente il rapporto medio di densità dei punti vicini di A rispetto ad A.

$$LOF_k(A) = \frac{\sum_{B \in N_k(A)} \frac{lrd_k(B)}{lrd_k(A)}}{|N_k(A)|}$$

Si noti che, prendendo il rapporto di densità con i punti vicini e non le densità effettive stesse, il metodo LOF può gestire set di dati con regioni di diverse densità. Un'importante limitazione di questo algoritmo riguarda il fatto che il parametro K-neighbours deve essere fissato a priori, prima dell'allenamento del modello. Ciò ha creato un ostacolo nello studio, costringendo ad eseguire diverse prove, con dei valori diversi di questo parametro, modificando il modello sotto analisi e di conseguenza variando il numero di valori anomali identificati. Si è previsto, come fase finale, un confronto con gli esperti del settore, che, analizzando i risultati ottenuti, possono oggettivamente scegliere un corretto valore del parametro.

- **Isolation Forest**: L'algoritmo di anomaly detection Isolation Forest [11] (Figura 2.8) rappresenta un metodo non supervisionato che si differenzia dai metodi tradizionali basati su misure di distanza e densità. L'iForest sfrutta il concetto di isolamento come metodo efficiente ed efficace per identificare le anomalie, poiché queste sono per definizione poche e diverse rispetto ai dati normali del dataset. L'idea principale è che le anomalie sono più sensibili all'isolamento rispetto ai dati normali. L'iForest misura la deviazione di un punto, valutando il tempo necessario per isolare il punto dal resto dei dati del dataset, costruisce modelli parziali e sfrutta il sub-sampling per migliorare le prestazioni dell'algoritmo.

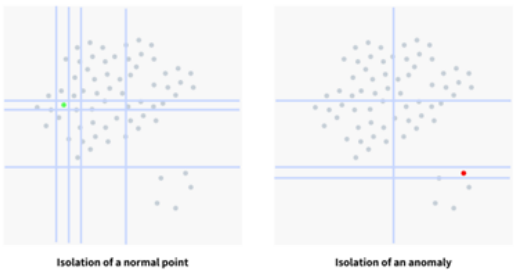

Figura 2.8: Isolation Forest

Come viene mostrato nella figura sottostante, l'algoritmo di Isolation Forest è basato sulla costruzione di alberi casuali di isolamento binari, chiamati iTree (Figura 2.9). L'obiettivo dell'algoritmo è quello di isolare i punti anomali in un

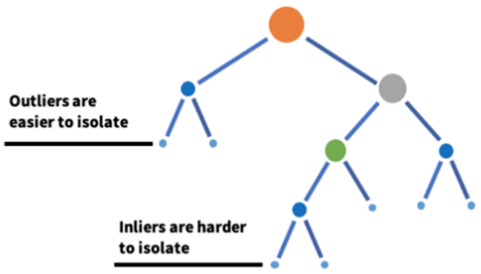

Figura 2.9: ITree

insieme di dati. Per fare ciò, vengono partizionate, in modo casuale, le istanze di dati all'interno degli alberi, selezionando un attributo e un valore a caso. Questo processo avviene in modo ricorsivo, partendo dal nodo radice e dividendo il set di dati in due sottoinsiemi in base al valore dell'attributo selezionato, e ripetuto finché si non si raggiunga un nodo foglia o si raggiunga una soglia predefinita di profondità dell'albero. Il numero di partizioni necessarie per isolare un punto corrisponde alla lunghezza del percorso tra il nodo radice e il nodo foglia. Poiché gli iTree sono costruiti in modo casuale, la lunghezza del percorso varia per ogni punto e non dipende dalla dimensione del set di dati. Per determinare se

un punto è un'anomalia, si calcola la media della lunghezza di ogni percorso : più esso risulti breve, maggiore è la probabilità che il punto sia un'anomalia. L'Isolation Forest diventa molto utilizzato in dataset di grandi dimensioni e con un'elevata cardinalità richiedendo risorse ridotte.

Capitolo 3

ORACLE

Questa sezione approfondisce meglio il contesto nel quale è stata sviluppato il lavoro. Partendo da una idea generale sul software di gestione del database fornito da Oracle, entrando poi nel dettaglio con i vari servizi e prodotti effettivamente utilizzati.

3.1 ORACLE DATABASE

Oracle Database [12] è un sistema di gestione di basi di dati relazionale sviluppato dalla società Oracle Corporation. È uno dei database più popolari al mondo ed è utilizzato da organizzazioni di ogni dimensione e settore. Questo sistema offre una vasta gamma di funzionalità, e gode di caratteristiche quali:

- **Affidabilità**: Oracle è noto per la sua affidabilità, grazie a una combinazione di tecnologie di ridondanza e di backup avanzate. Queste funzionalità consentono al database di gestire un enorme volume di transazioni in modo efficiente e affidabile.
- **Scalabilità**: Oracle è altamente scalabile e può gestire un numero molto elevato di utenti, transazioni e dati. Ciò lo rende adatto a organizzazioni di ogni

dimensione, dalle piccole alle grandi aziende.

- **Sicurezza**: la sicurezza dei dati è una priorità per Oracle, che offre una vasta gamma di funzionalità di sicurezza, tra cui crittografia avanzata, autenticazione a più fattori e controllo degli accessi granulare.

- **Performance**: il database Oracle è noto per la sua alta performance e velocità di elaborazione dei dati. Ciò lo rende adatto a organizzazioni che richiedono tempi di risposta rapidi e che elaborano un elevato volume di dati.

- **Gestione del ciclo di vita dei dati**: Oracle offre una vasta gamma di funzionalità di gestione del ciclo di vita dei dati, tra cui archiviazione, backup e ripristino dei dati, monitoraggio delle prestazioni e gestione degli spazi di archiviazione.

Per utilizzare Oracle, è necessario acquisire familiarità con il linguaggio di programmazione SQL, che viene utilizzato per interrogare e manipolare i dati nel database. Sono messi a disposizione diversi strumenti di sviluppo come Oracle SQL Developer, che fornisce un'interfaccia grafica per la gestione del database.

3.2 ORACLE ENTERPRISE MANAGER

Oracle Enterprise Manager (OEM) è una piattaforma di gestione di database e applicazioni che consente di monitorare e gestire i sistemi Oracle e le applicazioni aziendali su una vasta gamma di ambienti. OEM fornisce una serie di funzionalità tra cui:

- **Monitoraggio e gestione di database e istanze**;

- **Gestione di gruppi di database e applicazioni**;

- **Automazione di attività di gestione, come backup e ripristino, manutenzione degli indici, gestione dello storage e altro ancora**;
- **Monitoraggio delle prestazioni dei database e delle applicazioni**;
- **Capacità di pianificare e gestire i cambiamenti**;
- **Capacità di identificare e risolvere i problemi di sistema**.

OEM offre anche funzionalità di sicurezza, come la gestione delle autorizzazioni degli utenti e l'auditing. Inoltre, è possibile integrare OEM con altri strumenti di monitoraggio e gestione del sistema per ottenere una visibilità completa dell'ambiente IT dell'organizzazione.

3.3 AUTOMATIC WORKLOAD REPOSITORY

L'AWR, il cui processo di estrazione è descritto nella seguente immagine (Figura 3.1), viene attivato in modo predefinito in Oracle Database ed è integrato con Oracle Enterprise Manager. Raccoglie i dati delle prestazioni ogni ora e li archivia per un periodo di tempo configurabile.

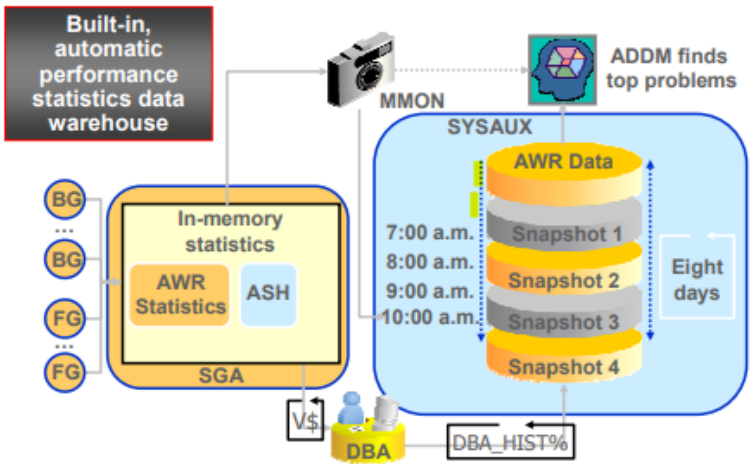

Figura 3.1: Concetto logico dietro agli AWR

Il processo di generazione del rapporto AWR prende i dati cumulativi di due snapshot e sottrae i dati cumulativi del primo snapshot dal secondo, generando un rapporto delta che mostra le statistiche e le informazioni rilevanti per il periodo di tempo richiesto. Raccoglie, elabora e archivia dati relativi alle prestazioni del database, in modo che gli amministratori del database possano analizzare e risolvere eventuali problemi di prestazioni.

Viene incluso il report Automatic Database Diagnostic Monitor (**ADDM** [13]) che analizza i dati nell'Automatic Workload Repository per identificare potenziali colli di bottiglia delle prestazioni. Per ciascuno dei problemi identificati, individua la causa principale e fornisce raccomandazioni per correggere il problema. Viene eseguita un'attività di analisi ADDM e i relativi risultati e suggerimenti vengono archiviati nel database ogni volta che viene acquisita un'istantanea AWR, a condizione che il parametro STATISTICS_LEVEL sia impostato su TYPICAL o ALL.

La limitazione maggiore degli AWR è legata al suo costo, infatti solo chi ha acquistato il pacchetto Diagnostic and Tuning di Oracle può sfruttare questa funzionalità aggiuntiva. Le metriche sulle quali vengono analizzate le performance degli statement sono di seguito descritte:

- **Elapsed Time**: L' "ELAPSED TIME" è il tempo totale impiegato dall'esecuzione di una query SQL, che comprende sia il tempo di CPU che il tempo di attesa per ottenere l'accesso a risorse come il disco o la memoria. In altre parole, è il tempo trascorso dalla prima esecuzione di una query fino al suo completamento, inclusi i tempi di attesa per le risorse necessarie per completare l'esecuzione;

- **Monitoraggio e gestione di database e istanze**:

- **CPU Time**: Il "CPU TIME" di un'operazione SQL in un database rappresenta il tempo in cui la CPU del sistema è stata impegnata nell'esecuzione di quella

specifica operazione. Di conseguenza, rappresenta il tempo in cui il processore ha dedicato risorse al calcolo richiesto dall'operazione SQL;

- **Buffer Gets**: "Buffer Gets" è una statistica utilizzata per misurare l'attività di lettura dalla buffer cache di Oracle. La buffer cache è una zona di memoria condivisa utilizzata per contenere i dati letti da un'unità di I/O del disco rigido. Quando un'istruzione SQL viene eseguita, Oracle cerca prima i dati richiesti nella buffer cache, in modo da evitare la necessità di accedere al disco per recuperare i dati. Se i dati richiesti non sono presenti nella buffer cache, Oracle dovrà accedere direttamente al disco per recuperarli, e questo comporterà un aumento delle "Buffer Gets";

- **Disk Reads**: "Disk Reads" si riferisce al numero di letture di dati effettuate dal disco rigido durante l'esecuzione di una determinata operazione o query nel database Oracle. Come delineato in precedenza, quando il motore del database non è in grado di trovare i dati richiesti nella memoria cache, è necessario recuperare i dati dal disco rigido. Questo processo richiede più tempo rispetto alla lettura dei dati dalla memoria cache, poiché più lento. Di conseguenza, un alto numero di "Disk Reads" può indicare una cattiva prestazione del database, in quanto potrebbe indicare che la maggior parte delle operazioni richiedono la lettura dei dati dal disco rigido invece che dalla memoria cache;

- **Executions**: Le esecuzioni di una query in un database Oracle rappresentano il numero di volte in cui quella specifica query è stata eseguita;

- **Parse Calls**: Parse call si riferisce al processo di analisi di un'istruzione SQL e creazione di un piano di esecuzione per essa. Quando una istruzione SQL viene eseguita per la prima volta, il database deve analizzare l'istruzione e determinare come eseguirla. Questo comporta il controllo della sintassi, la determinazione

delle tabelle e delle colonne coinvolte e la generazione di un piano di esecuzione che specifica le operazioni necessarie per recuperare o modificare i dati;

- **Shareable Memory**: La memoria condivisa si riferisce alla porzione di memoria utilizzata dall'istanza del database, che potrebbe essere condivisa da più utenti e processi. Questa memoria include vari componenti, come la buffer cache, shared pool, il buffer dei redo log e la large pool. La buffer cache del database è la porzione di memoria utilizzata per memorizzare i blocchi di dati che sono stati letti di recente dal disco o modificati in memoria. La shared pool è utilizzata per memorizzare il codice SQL e PL/SQL condiviso, le cache del dizionario dei dati e altre strutture di memoria condivisa. Il redo log buffer del è utilizzato per trattenere temporaneamente le voci di redo log prima che siano scritte su disco, mentre il large pool viene utilizzato per varie allocazioni di memoria di grandi dimensioni, come operazioni di backup e ripristino, operazioni di query parallela e così via;

- **Version Count**: "Version Count" si riferisce al numero di versioni attive di una particolare istruzione SQL nella shared pool del database. Quando viene eseguita un'istruzione SQL, Oracle ne genera un hash value univoco e lo memorizza nella shared pool. Se la stessa istruzione viene eseguita di nuovo, Oracle controlla la shared pool per l'hash value e, se già esistente, riutilizza il piano di esecuzione. Tuttavia, se viene analizzata una nuova versione dell'istruzione a causa di una modifica dell'istruzione o dei suoi oggetti sottostanti, verrà generato un nuovo hash value e piano di esecuzione e memorizzato nella shared pool. La metrica "Version Count" in AWR tiene traccia del numero di versioni attive di un'istruzione nella shared pool durante un periodo di tempo specificato, il che può essere utile per identificare le istruzioni SQL che consumano memoria eccessiva o causano problemi di prestazioni.

- **Cluster Wait Time**: "Cluster Wait Time" è una metrica che misura la quantità di tempo trascorso in attesa di risorse di cluster durante le operazioni del database. Un cluster è un gruppo di server interconnessi che lavorano insieme per fornire alta disponibilità e scalabilità per il database. Le risorse del cluster possono includere dischi condivisi, interconnettori e altri componenti hardware e software necessari per la comunicazione e il coordinamento tra i server nel cluster. Quando un'operazione del database richiede l'accesso alle risorse del cluster, potrebbe essere necessario attendere che tali risorse diventino disponibili. La metrica "Cluster Wait Time" in AWR tiene traccia del tempo trascorso in attesa di risorse di cluster, il che può essere utile per identificare i colli di bottiglia delle prestazioni e ottimizzare il database per una migliore performance in un ambiente di cluster.

3.4 LE VISTE

Le viste [14] sono un elemento utilizzato dalla maggior parte dei Database management system, cioè dal sistema software progettato per consentire la creazione, la manipolazione e l'interrogazione efficiente di database. Si tratta, come suggerisce il nome, di "modi di vedere i dati". OEM mette a disposizione diverse viste descritte in dettaglio nella documentazione ufficiale di Oracle e il loro numero supera le migliaia. Una vista è rappresentata da una query di SELECT, il cui risultato può essere utilizzato come se fosse una tabella. Gli sviluppatori possono utilizzare queste viste per creare applicazioni e sfruttare i dati del repository senza preoccuparsi di eventuali modifiche alla struttura dell'infrastruttura. Sono, ora, descritte le tre viste di maggiore interesse per il nostro lavoro.

3.4.1 V$ACTIVE_SESSION_HISTORY

La vista V$ACTIVE_SESSION_HISTORY mostra l'attività delle sessioni di database campionate. Essa contiene istantanee delle sessioni attive del database prese una volta al secondo. Una sessione di database è considerata attiva se era sulla CPU o era in attesa di un evento che non apparteneva alla classe di attesa Idle. Questa vista contiene una riga per ogni sessione attiva, per campione, e restituisce per prima le righe degli ultimi campioni delle sessioni.

3.4.2 V$SQL

V$SQL elenca le statistiche sulle aree SQL condivise senza la clausola GROUP BY e contiene una riga per ogni child staccato dalla istruzione originale inserita. Le statistiche visualizzate in V$SQL vengono normalmente aggiornate alla fine dell'esecuzione della query. Inoltre vengono anche elencate le informazioni sulla sintassi SQL, i dati di Runtime e i dati di stato per ogni istruzione condivisa nella cache di condivisione, componente di memoria ad alta velocità che memorizza i dati frequentemente utilizzati per ridurre i tempi di accesso al database. (La cache è utilizzata per migliorare le prestazioni del sistema e minimizzare la necessità di accedere al disco rigido o ad altre risorse di storage più lente). La vista include informazioni sulla durata dell'istruzione, il tempo di CPU, il numero di volte in cui l'istruzione è stata eseguita e il tipo di piano utilizzato per eseguire l'istruzione. Ciò rende questa vista potenzialmente utile per identificare le query lente o inefficienti e per monitorare le prestazioni del database. In ambiente aziendale, viene spesso interrogata questa vista ogni qual volta che un cliente, presentando dei rallentamenti e sovraccarichi sul database. Viene riportata, come esempio, una selezione di alcune statistiche riguardanti un preciso 'SQL ID' fatta da un consulente di Mediamente, subito dopo aver ricevuto una segnalazione.

3.4.3 V$SQL_MONITOR

V$SQL_MONITOR visualizza le istruzioni SQL la cui esecuzione è stata o sta venendo monitorata dall'OEM. Questa vista contiene informazioni globali ad alto livello sulle operazioni semplici e composite del database, a patto che si verifichi una delle seguenti condizioni:

- le operazioni vengano eseguite in parallelo;
- le operazioni abbiano consumato almeno 5 secondi di tempo CPU o di I/O in una singola esecuzione;
- il monitoraggio per le operazioni sia forzato dall'uso dell'istruzione MONITOR;

Una voce in V$SQL_MONITOR è dedicata a una singola esecuzione di un'istruzione SQL. Se il database monitora due esecuzioni della stessa istruzione SQL, ogni esecuzione ha una voce separata. V$SQL_MONITOR presenta una voce per il processo parallel execution coordinator e una voce per ogni processo di esecuzione parallela. Poiché i processi allocati per l'esecuzione parallela di un'istruzione SQL cooperano per la stessa esecuzione, queste voci condividono la stessa chiave di esecuzione (la combinazione di SQL_ID, SQL_EXEC_START e SQL_EXEC_ID). Dopo un'attenta analisi, insieme alla collaborazione dei massimi esponenti della società si è deciso di optare di analizzare solo la vista v$sql_monitor, scartando quindi le altre. La vista v$sql, come viene descritto precedentemente, raccoglie i dati in modo cumulato, senza tenere traccia di sezione in esecuzione in parallelo (l'esecuzione simultanea di una query su più processori o core del sistema). Per quanto riguarda la vista V$active_session_history, si è scelto di scartarla in quanto le informazioni in essa salvate riguardano sezioni attive, per le quali non si è ancora conclusa la loro completa esecuzione. Si fa anche molta attenzione alle statistiche a utili a questo lavoro, alcune delle quali, come il CPU time e la quantità di memoria alla quale si

vuole richiedere l'accesso. Invece, la vista analizzata in questo sotto capitolo, riesce a raccogliere tutti i dati significativi per il nostro lavoro, tenendo anche traccia di esecuzioni in parallelo. Inoltre, registrando solo esecuzioni più lunghe di 5 secondi, riuscirebbe a segnalare tutti i record per noi anomali; come discusso con tecnici di azienda, la probabilità che il lancio di una query possa creare dei gravi rallentamenti al database durando più di 5 secondi è molto alta.

Capitolo 4

PROGETTAZIONE

In questo capitolo vengono illustrate le scelte implementative adottate durante il lavoro di ricerca, impostando prima un discorso sulle tecnologie e sui linguaggi usati per la creazione dell'algoritmo finale, arrivando alla vera e propria applicazione dei risultati su dati concreti e reali.

4.1 TECNOLOGIE USATE

Nel capitolo precedente viene brevemente descritta questa parte, parlando brevemente dei linguaggi di programmazione usati.

4.1.1 SQL Developer

Il database è una struttura software piuttosto complessa, in grado di ottimizzare l'accesso e la modifica di grandi quantità di dati archiviati su un generico dispositivo di archiviazione. Negli anni, lo sviluppo dei database ha visto una crescita notevole, e si è affermata quasi ovunque la classe dei database relazionali. Per interagire con tale classe di database vi è, oggi, un linguaggio considerato in tutto e per tutto uno standard, e denominato SQL (acronimo che sta per Structured Query Language)

[15]. L'invocazione interattiva coinvolge l'utilizzo di un programma, spesso di tipo testuale, che consente all'utente di inserire le istruzioni SQL, inviarle al DBMS e visualizzare i risultati. Un esempio di programma che fornisce questa funzionalità è "SQL Developer". (Fig. 4.1)

4.1.2 Python

Come linguaggio di programmazione scelto per l'implementazione di possibili soluzioni di anomaly detection è stato scelto Python [16]. Le motivazioni principali per l'adozione di Python in questo contesto aziendale sono le seguenti:

1. **Multi-paradigma e flessibilità**: E' un linguaggio multi-paradigma che consente di scrivere codice in diversi stili, come ad esempio programmazione procedurale, orientata agli oggetti e funzionale. Questa caratteristica offre flessibilità nello sviluppo e permette di adattarsi alle diverse esigenze del progetto. Inoltre, Python è noto per la sua sintassi chiara e leggibile, che facilita la comprensione e la manutenzione del codice.

2. **Ampia scelta di librerie**: Offre una vasta gamma di librerie standard e open-source che coprono molte aree, inclusa l'analisi dei dati e il machine learning. Ad esempio, la libreria open-source Scikit-learn fornisce una vasta collezione di algoritmi di machine learning pronti all'uso. Pandas è una libreria focalizzata sulla manipolazione e analisi dei dati, mentre Matplotlib consente la creazione di grafici in due dimensioni. L'ampia disponibilità di queste librerie semplifica lo sviluppo di soluzioni basate sull'analisi dei dati e sul machine learning, consentendo di risparmiare tempo nella scrittura di codice da zero e sfruttando soluzioni collaudate dalla comunità.

3. **Adozione in altri progetti aziendali**: Adottato con successo in altri progetti aziendali, il che suggerisce che il linguaggio ha dimostrato la sua affidabilità

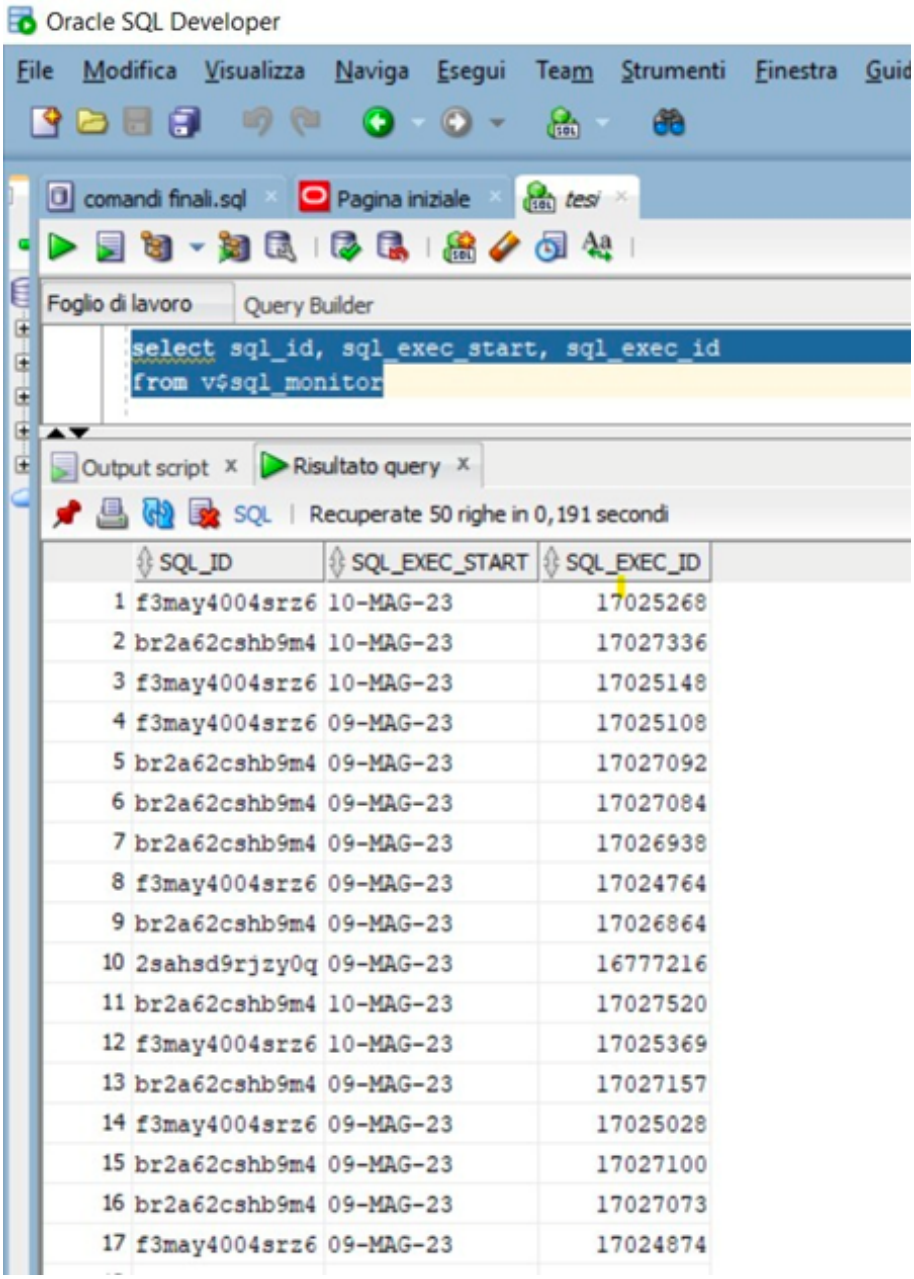

Figura 4.1: Interfaccia SQL Developer

e idoneità all'interno dell'organizzazione. Questo può facilitare l'integrazione del nuovo progetto basato su Python con l'ecosistema esistente e sfruttare l'esperienza e le competenze già presenti nel team.

4.2 CREAZIONE DELL'ALGORITMO

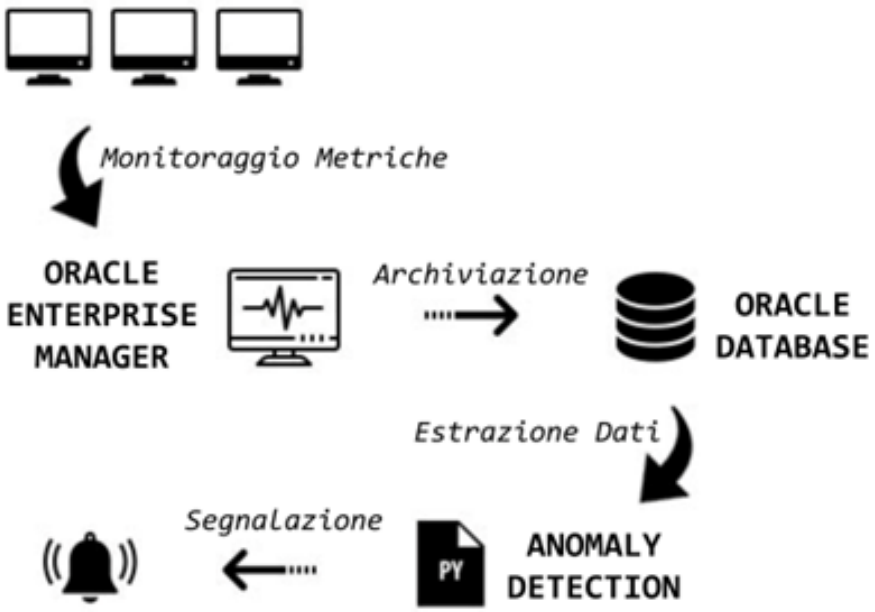

Figura 4.2: Flusso di dati all'interno del sistema considerato

Il processo, Fig.4.2, inizia con la comunicazione delle informazioni sulle risorse monitorate tra gli host e l'OEM, che archivia tali informazioni nel proprio repository. Successivamente, i dati storici delle metriche raccolte vengono utilizzati come dataset di addestramento per il modulo di anomaly detection implementato.

Una volta completata la fase di addestramento del modello, il modulo di anomaly detection analizza regolarmente gli ultimi dati raccolti dall'OEM. Nel caso in cui venga rilevato un comportamento anomalo, viene innescata una procedura di alerting che notifica all'utente l'anomalia identificata. Tuttavia, poiché il carico e l'utilizzo delle risorse di un determinato target possono variare nel tempo, periodicamente è necessario aggiornare i dati di addestramento del modello.

4.3 DATASET

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

Avere un dataset affidabile e adatto all'applicazione di modelli di machine learning è uno degli step essenziali in questo tipo di ricerca. Come detto nei capitoli precedenti, per il raggiungimento degli obiettivi aziendali, il lavoro inizia con la storicizzazione dei dati presenti nella vista v$sql_monitor. E' necessario fornire quanti più dati possibili per allenare il modello finale; infatti la vista presa in esame è caratterizzata da una retention molto bassa, che varia da poco più di un giorno, in database poco usati, fino ad arrivare alla durata di pochi minuti per database molto impegnati, con alta transazionalità, come possono essere tutti i sistemi ingegneristici usati per il monitoraggio della produzione di grandi società.

```
INSERT INTO CONI35.HIST_SQL_MONITOR
SELECT *
FROM GV$SQL_MONITOR a
WHERE NOT EXISTS (
    SELECT *
    FROM CONI35.HIST_SQL_MONITOR b
    WHERE a.inst_id=b.inst_id
      AND a.sql_id=b.sql_id
      AND a.sql_exec_start=b.sql_exec_start
      AND a.sql_exec_id=b.sql_exec_id
      AND a.sid=b.sid
      AND a.session_serial#=b.session_serial#
)
AND a.sql_exec_start IS NOT NULL;
```

Chiave univoca: "SQL_ID", "SQL_EXEC_START", "SQL_EXEC_ID", "INST_ID", "SID", "SESSION_SERIAL#".

Grazie a questo comando, lanciato all'interno di un job schedulato ogni 5 minuti, viene creata una tabella di storicizzazione, popolata da tutti i record che vengono registrati nella vista v$sql_monitor, ma di cui si perderà traccia. Possiamo ora, quindi, allenare il nostro modello con dati che sono stati raccolalti nell'arco di un mese. Il processo di drop della tabella HIST_SQL_MONITOR avviene tramite un secondo job, schedulato questa volta ogni 30 giorni, che elimina quindi tutti i dati molto datati.

4.4 PRE-PROCESSING

La fase di pre-processing dei dati è un'importante fase preliminare nell'applicazione di modelli di Data Mining, incluso l'anomaly detection. Questa fase mira a trasformare il dataset di dati grezzi in un insieme di dati che sia adatto per l'applicazione delle tecniche di analisi.

I principali obiettivi della fase di pre-processing sono i seguenti:

- **Migliorare la qualità del dataset**: Durante la raccolta dei dati, è possibile che si verifichino errori, dati mancanti o inconsistenti. La fase di pre-processing si occupa di identificare e correggere tali problemi per garantire che il dataset sia affidabile e di alta qualità. Ciò può comportare l'eliminazione dei dati duplicati, la gestione dei valori mancanti o errati, o la risoluzione di eventuali inconsistenze.

- **Ridurre la cardinalità e la dimensionalità del dataset**: Gli insimi di dati possono essere molto grandi e complessi, con un elevato numero di variabili o attributi. Questa complessità può influire sulle prestazioni dei modelli di Data Mining e può portare a una maggiore complessità computazionale. La fase di pre-processing può includere tecniche di riduzione della dimensionalità, come la selezione delle caratteristiche o l'estrazione delle caratteristiche, al fine di ridurre il numero di attributi senza compromettere le informazioni importanti per l'anomaly detection.

- **Migliorare le performance e l'accuratezza dei modelli di Data Mining**: La qualità dei dati e la riduzione della dimensionalità possono influire sulle prestazioni e sull'accuratezza dei modelli di Data Mining. E' possibile migliorare quindi tali aspetti ottimizzando la collezione di dati per l'applicazione specifica. Ad esempio, è possibile standardizzare o normalizzare i dati per garantire

che siano comparabili tra loro, oppure bilanciare le classi in caso di dataset sbilanciati.

Nel contesto specifico dell'estrazione dei dati dal repository dell'Oracle Enterprise Manager, nella sezione successiva saranno illustrati i dettagli delle tecniche di pre-processing utilizzate. Questo processo di pre-processing sarà finalizzato a migliorare la qualità dei dati estratti, ridurre la complessità del dataset e migliorare le prestazioni e l'accuratezza dei modelli di anomaly detection applicati successivamente.

Un primo lavoro di pre-processing avviene pulendo i dati (*Data Cleaning*), applicando uno dei tanti processi atti a migliorare la qualità dei dati presenti all'interno del dataset per consentirne la loro memorizzazione. Come si può vedere dall'illustrazione delle statistiche registrate all'interno della vista v$sql_monitor, compare una voce relativa allo STATUS dell'esecuzione:

- **QUEUED** - SQL statement is queued
- **EXECUTING** - SQL statement is still executing
- **DONE (ERROR)** - Execution terminated with an error
- **DONE (FIRST N ROWS)** - Execution terminated by the application before all rows were fetched
- **DONE (ALL ROWS)** - Execution terminated and all rows were fetched
- **DONE** - Execution terminated (parallel execution)

La pulizia dei dati è avvenuta, insieme alla consulenza di esperti, andando a selezionare solo quei record che riportavano *DONE (ALL ROWS)* nel campo *status*. Prendendo come esempio una istruzione che consumi più di 5 secondi di CPU; se al tempo t_0 la non è ancora terminata la sua esecuzione e quindi viene registrata con il valore *EXECUTING*. Quando, al tempo t_1, dopo 5 minuti, viene lanciato

di nuovo lo script che storicizza la vista, la stessa esecuzione verrebbe registrata di nuovo, ma questa volta con il valore *DONE (ALL ROWS)*. Ciò porterebbe ad un dimensionamento eccessivo delle tabelle di storicizzazione, e rallenterebbe l'allenamento del modello.

4.5 MODELLO

Considerando il contesto aziendale in cui si fornisce un servizio di monitoraggio dei sistemi informatici di medio/grandi aziende con molteplici host e configurazioni diverse, la creazione di un dataset uniforme suddiviso in classi "anomale" e "normale" può risultare estremamente complesso [17]. Per questo motivo gli algoritmi non supervisionati, quindi che non hanno bisogno di etichettature sui dati, si pongono meglio al problema. La scelta dei modelli z-score, LOF e iForest, descritti nel capitolo precedente, è stata spinta anche dalla presenza della libreria scikit-learn su Python, che offre degli algoritmi specifici per ognuno di essi. Vediamo nel dettaglio quali siano i principali parametri che necessitano di taratura per ogni algoritmo:

- **sklearn.ensemble.IsolationForest**: *contamination* indica la proporzione dei campioni anomali all'interno del dataset.

- **sklearn.neighbors.LocalOutlierFactor**: *n_neighbors* e *contamination* rispettivamente indicano il numero di vicini utilizzato dal k-nearest neighbors e la proporzione dei campioni anomali all'interno del dataset.

- Per il modello normale standardizzato (**z-score**), abbiamo bisogno del limite superiore che delimita l'area con maggiore probabilità, dove i dati sono considerati normali.

  Scikit-learn, nell'implementazione degli algoritmi considerati, fornisce due metodi principali per l'utilizzo dei modelli di anomaly detection:

1. **Metodo "fit"**: Questo metodo viene utilizzato per addestrare il modello sull'identificazione delle anomalie. Prende in input il dataset non etichettato e lo utilizza per apprendere i pattern e le caratteristiche dei dati normali. Durante questa fase di addestramento, il modello impara a riconoscere i comportamenti normali e a stabilire una linea di base per valutare la presenza di anomalie. Il metodo "fit" regola i parametri del modello in base ai dati forniti, in modo da ottimizzare la capacità del modello di identificare le anomalie durante la fase successiva di predizione.
2. **Metodo "predict"**: Dopo che il modello è stato addestrato utilizzando il metodo "fit", viene utilizzato il metodo "predict" per valutare il grado di anormalità di nuove osservazioni o campioni di dati sotto osservazione. Questo metodo prende in input una nuova osservazione e restituisce una misura o una stima del grado di anomalia del campione. Il valore di output può essere un punteggio di anomalia, una probabilità o una classificazione binaria (anomalo/normale), a seconda dell'algoritmo di anomaly detection utilizzato e delle sue specifiche implementative.

L'utilizzo combinato dei metodi "fit" e "predict" consente di addestrare il modello sui dati normali e successivamente di applicarlo per identificare le anomalie nelle nuove osservazioni. Questo processo di addestramento e predizione può essere iterato periodicamente per adattare il modello ai cambiamenti nel comportamento dei dati e mantenere inalterata la sua capacità di rilevare anomalie nel tempo.

Capitolo 5

SVILUPPO DEL LAVORO

Questo capitolo racchiude tutti i vari passi di cui è composto questo lavoro, andandolo ad analizzare in dettaglio e descrivendo tutte le osservazioni e considerazioni prese per ognuno di esso.

5.1 ESTRAZIONE DEI DATI

Come detto nei capitoli precedenti, si ha bisogno, prima di tutto, di accumulare abbastanza dati per poter allenare il modello. La storicizzazione dei record registrati in `v$sql_monitor` inizia con la creazione di una nuova tabella, per comodità nominata in questo lavoro "storicizzazione_sql_monitor", copia esatta della vista. È stato pensata, in questo momento, la schedulazione di un job che, ogni 30 minuti, faccia uno screenshot della situazione in tempo reale, e salvi tutti i nuovi dati ancora non presenti nella nuova tabella creata. Questo tempo è stato scelto, dopo varie analisi su database clienti, in quanto corrisponde come la più piccola retantion della vista trovata. Come si può immaginare, per evitare di accumulare troppi dati, non resta che creare un secondo job, schedularne la

sua esecuzione ogni T giorni, che vada a sfoltire i dati presenti nella tabella dello storico, eliminando tutti i record che, da quel momento, possano avere una età maggiore di T. Durante i vari supporti specialistici avuti, si è scelto un tempo T corrispondente a 30 giorni. Questa durata garantisce di poter raccogliere i dati avendo un sistema sempre nelle stesse condizioni, e che i dati possano non essere influenzati da periodicità qualsiasi. Tutto ciò perché, durante il normale esercizio di produzioni, una qualsiasi azienda può aver bisogno di modificare la struttura del suo database, come ampliarne la memoria o aggiornarne i software. Tutto ciò può contribuire a modificare il tempo di esecuzione di una query, perché potrebbero modificarsi sia la quantità di dati alla quale si cerca di fare accesso, ma anche la struttura dei dati stessi.

5.2 MANIPOLAZIONE DEI DATI E PRE-PROCESSING

Si vuole descrivere, nel paragrafo sottostante, l'applicativo codificato in Python, che esegue la procedura di Anomaly Detection. La lavoro di tesi che si vuole portare a termine ha riversato tanto interesse verso il mondo Python, grazie al quale si è semplificato, e non poco, il tutto. Verranno ora descritte le librerie utilizzate, importate nel sistema come segue:

```
1 import numpy as np
2 import pandas as pd
3 import cx_Oracle
4 import sklearn
5 import scipy
```

Numpy [18] è una libreria Python molto popolare che aggiunge il supporto per l'elaborazione di grandi matrici e array multidimensionali. Offre un'ampia collezione di funzioni matematiche di alto livello che possono essere applicate in modo efficiente su queste strutture dati.

D'altra parte, **Pandas** [19] è una libreria Python focalizzata sulla manipolazione di dati tabulari, come tabelle numeriche e serie temporali. Pandas fornisce strutture dati potenti come i DataFrame, che sono essenzialmente tabelle con righe e colonne.

Numpy offre le basi per la gestione efficiente di grandi array multidimensionali, mentre Pandas fornisce un'interfaccia più user-friendly per la manipolazione dei dati tabulari. Entrambe le librerie sono molto potenti e ampiamente utilizzate nella comunità Python per l'elaborazione e l'analisi dei dati.

Granzia alla libreria **Cx_oracle** viene garantito un collegamento al database Oracle, alla lettura dei comandi SQL e nell'estrazione dei dati.

Scikit-learn [20], come viene descritto nei capitoli precedenti, è una libreria open source di apprendimento automatico per il linguaggio di programmazione Python. Contiene algoritmi di classificazione, regressione e clustering (raggruppamento) e macchine a vettori di supporto, regressione logistica, classificatore bayesiano, k-mean e DBSCAN, ed è progettato per operare con le librerie NumPy e SciPy. Da qui l'importazione anche della libreria **Scipy** [18] che contiene moduli per l'ottimizzazione per l'algebra lineare, l'integrazione e tante altre funzioni speciali. Finito di installare le librerie e, con maggior attenzione, i pacchetti che andremo ad utilizzare, si è susseguita una seconda attenta selezione di dati. Facendo sempre attenzione a prendere solo le informazioni riguardanti esecuzioni terminate, escludendo quindi quelle in esecuzione, è stato creato un dataframe vuoto e lo si è popolato con ciò che si è raccolto.

Un sorge un primo problema sul metodo di salvataggio dei primi dati estratti: il salvaggio di questi ultimi su un semplice dataframe non semplifica il lavoro, visto che non garantisce una rapida lettura e complica il loro raggruppamento. In soccorso arriva il **dizionario** [21], struttura dati mutabile e non ordinata che memorizza coppie di elementi, noti come **chiavi** e **valori**.

Viene, quindi, creato un primo dizionario che raggruppa i dati per SQL_ID. Un secondo, annidato al primo, raggruppa ricorsivamente i dati per metriche, ritrovando una struttura finale così formata:

SQL_ID	[**ELAPSED TIME**]	(lista di tutti i valori raccolti)
	[**CPU TIME**]	(lista di tutti i valori raccolti)
	[**USER I/O TIME**]	(lista di tutti i valori raccolti)
	⋮	⋮

Questo processo alleggerisce il carico lavoro richiesto dal nostro applicativo, andando ad allenare un modello per ogni metrica per ogni sql_id, solo quando viene richiesto.

5.3 FORMULAZIONE DELLE FUNZIONI DEL MODELLO

Come più volte detto, sono stati sviluppati tre modelli diversi di Anomaly Detection. In questa sezione si vuole dare una visualizzazione del codice di ogni modello, riportando solo una loro versione, corrispondente alla configurazione di default. Con l'avanzare dei test, si sono modificati più volte i parametri caratteristici, per far calzare alla perfezione il risultato finale con il caso studio in esame.

```
def compute_isolation_forest(dfa, dfb, metric):
    model = IsolationForest(contamination=0.1)
    model.fit(dfa.set_index('SQL_ID'))
    anomaly_scores = model.decision_function(dfb.set_index('SQL_ID'))
    anomaly_predictions = model.predict(dfb.set_index('SQL_ID'))
    valori_presenti = False
    for i in range(dfb.shape[0]):
        score = anomaly_scores[i]
        prediction = anomaly_predictions[i]
        if prediction == -1:
            sql_id = dfb['SQL_ID'].iloc[i]
            metric_value = dfb[metric].iloc[i]
            others = df_test[(df_test['SQL_ID']==sql_id) & (df_test[metric]==
    metric_value)][['SQL_ID', 'SQL_EXEC_START', 'SQL_EXEC_ID']].iloc[0].
    to_dict()
            print(f"SQL_ID: {sql_id}, SQL_EXEC_START: {others['SQL_EXEC_START
    ']}, SQL_EXEC_ID: {others['SQL_EXEC_ID']}, METRIC: {metric}, METRIC_VAULE:
     {metric_value}, LOF: {score}, Prediction: {prediction}")
            valori_presenti = True
    return
```

Questa è la funzione definita per l'algoritmo Isolation Forest. (Parametro della contaminazione impostato a 0,1). Nell' **iForest**, la contamination corrisponde alla proporzione di valori anomali nel set di dati; è un parametro importante per la fase di fitting per definire la soglia sui punteggi dei campioni.

Avendo, ora, definita la funzione, estratto entrambe le categorie di dati utili, si può applicare il modello. Tutto inizia analizzando i dati salvati sul DataFrame: viene selezionato una prima terzina, formata dalle chiavi primarie della tabella *v$sql_monitor*, e si cercano corrispondenze all'interno del dizionario. Se presente, si può iniziare ad addestrare il modello, per ogni metrica supervisionata. Qui nasce un'ulteriore limitazione del nostro programma. Può capitare che, per un determinata esecuzione, non si abbiano abbastanza dati storici sui quali allenare. Per soccombere a questo problema si è scelto di inserire un ulteriore vincolo:

vengono salvati, dentro il dizionario, solo quei sql_id di cui sono state registrate almeno **N** esecuzioni. Per lo stesso concetto della variabile T prima nominata, anche N è stata scelta insieme a esperti del settore, mettendo in gioco la loro esperienza, arrivando a identificarne un valore minimo corrispondente a 50.

Superato questo ostacolo, l'algoritmo continua il suo lavoro, portando a completamento il processo di fit, al termine del quale parte la vera e propria Anomaly Detection. Si confronta il valore della metrica campionata con il modello fittato e se risulta anomala, vengono salvate tutte le sue informazioni all'interno di una tabella temporanea, composta dai campi: *SQL_ID*, *SQL_EXEC_START*, *SQL_EXEC_ID*, *METRICA ANOMALA*, *VALORE DELLA METRICA* (viene riportato un esamepio nella Fig.fig:Tabella dati

SQL_ID	SQL_EXEC_START	SQL_EXEC_ID	METRIC	METRIC_VALUE
124g2h47md3a1	09/05/2023 00:46	33554432	BUFFER_GETS	130416
124g2h47md3a1	09/05/2023 00:46	33554432	DISK_READS	16323
b0qr5c1t5cjbb	09/05/2023 01:28	16787883	USER_IO_TIME	0,003082
dc39kcxuf9xt5	09/05/2023 01:26	16987513	ELAPSED	0,07058
dc39kcxuf9xt5	09/05/2023 01:13	16787883	ELAPSED	0,050208
dc39kcxuf9xt5	09/05/2023 02:13	33990328	ELAPSED	0,051915
dc39kcxuf9xt5	09/05/2023 00:39	16987481	USER_IO_TIME	0,013103
dc39kcxuf9xt5	09/05/2023 01:13	16987513	USER_IO_TIME	0,033239
dc39kcxuf9xt5	09/05/2023 02:13	33990328	USER_IO_TIME	0,042108
dc39kcxuf9xt5	09/05/2023 01:45	16987541	USER_IO_TIME	0,019083
dc39kcxuf9xt5	09/05/2023 01:26	16987531	BUFFER_GETS	336
dc39kcxuf9xt5	09/05/2023 01:45	16987541	DISK_READS	16
dc39kcxuf9xt5	09/05/2023 01:45	16987541	IO_INTERCONNECT_BYTES	131072
dc39kcxuf9xt5	09/05/2023 01:13	16987513	PHYSICAL_READ_BYTES	139264

Figura 5.1: storicizzazione_sql_monitor

Processo del tutto simile si applica anche per l'algoritmo di **LOF**, andando prima a definire una funzione che poi viene richiamata ogni volta che ha bisogno di effettuare l'analisi.

```
def compute_localoutlierfactor(dfa, dfb, metric):
    model = localoutlierfactor(n_neighbors=20, novelty=False)
    model.fit(dfa.set_index('SQL_ID'))
    anomaly_scores = model.decision_function(dfb.set_index('SQL_ID'))
    anomaly_predictions = model.predict(dfb.set_index('SQL_ID'))
    valori_presenti = False
    for i in range(dfb.shape[0]):
        score = anomaly_scores[i]
        prediction = anomaly_predictions[i]
        if prediction == -1:
            sql_id = dfb['SQL_ID'].iloc[i]
            metric_value = dfb[metric].iloc[i]
            others = df_test[(df_test['SQL_ID']==sql_id) & (df_test[metric]==
    metric_value)][['SQL_ID', 'SQL_EXEC_START', 'SQL_EXEC_ID']].iloc[0].
    to_dict()
            print(f"SQL_ID: {sql_id}, SQL_EXEC_START: {others['SQL_EXEC_START
    ']}, SQL_EXEC_ID: {others['SQL_EXEC_ID']}, METRIC: {metric}, METRIC_VAULE:
     {metric_value}")
            valori_presenti = True
    return
```

Tutto molto diverso avviene nel calcolo dello **z-score**. Esso, infatti, gode di una richiesta infima di risorse del sistema per essere processato.

```
def find_anomalies(values, values2, threshold=1.97):
    std = np.std(values)
    mean = np.mean(values)
    zscore = np.array([(i-mean)/std for i in values2])
    return np.array(zscore > threshold  , dtype=int)
from tabulate import tabulate
results = []
for sql_id, metric_list in d.items():
    if sql_id in d2:
        for metric, value_list in metric_list.items():
            value_list_2 = d2[sql_id][metric]
            anomalies = find_anomalies(value_list, value_list_2)
            for i, value in enumerate(anomalies):
                if value:
                    metric_value = value_list_2[i]
                    others = df_test[(df_test['SQL_ID'] == sql_id) & (df_test[
    metric] == metric_value)][['SQL_ID', 'SQL_EXEC_START', 'SQL_EXEC_ID']].
    iloc[0].to_dict()
                    result = {
                        'SQL_ID': sql_id,
                        'SQL_EXEC_START': others['SQL_EXEC_START'],
                        'SQL_EXEC_ID': others['SQL_EXEC_ID'],
                        'METRIC': metric,
                        'METRIC_VALUE': metric_value
                    }
                    results.append(result)
print(tabulate(results, headers='keys', tablefmt='psql'))
```

A differenza dei due casi precedenti, per applicare questo modello si ha bisogno di una nuova tabella, chiamata per comodità **ZScore**, caratterizzata dai campi: *SQL_ID*, *METRICA*, *MEDIA* e *DEVIAZIONE STANDARD*. Pescando dal dizionario utilizzato precedentemente, si calcola la la media e deviazione standard di ogni lista di valori e per ogni metrica, andando a popolare ZScore con le informazioni trovate.

Per ogni *SQL_ID* esaminato, viene applicata una standardizzazione delle sue

metriche usando i valori caratteristici della normale, media e varianza. Tali risultati sono confrontanti con i valori della threshold: se supera tale soglia, siamo in presenza di un valore anomalo.

5.4 RISULTATI

In questo sottocapitolo, si vogliono presentare e analizzare i vari risultati ottenuti dalle diverse sperimentazioni condotte, utilizzando anche grafici per facilitare l'identificazione visiva delle anomalie presenti nei dati. Durante le nostre ricerche, sono stati eseguiti numerosi test su ciascun modello, utilizzando lo stesso dataset di base, ma allenandolo ripetutamente con una varietà di parametri differenti. Questo approccio ha permesso di esaminare l'influenza di ciascun parametro sui risultati ottenuti.

I risultati di queste sperimentazioni sono stati attentamente valutati in collaborazione con un team specializzato della Business Unit di Infrastruttura. Questa squadra ha fornito preziose competenze tecniche e pratiche, contribuendo a convalidare solo quei parametri che delineano un caso di studio il più realistico possibile. Grazie a questo processo di validazione, si è potuto garantire che le conclusioni tratte fossero applicabili a scenari reali, migliorando così l'affidabilità e la robustezza dei modelli sviluppati.

Ora, si desidera illustrare, con un esempio concreto, come il modello di Isolation Forest possa essere utilizzato per rilevare anomalie in un dataset. A tale scopo, verranno mostrati i risultati ottenuti variando i valori dei parametri del modello e osservando come queste variazioni influenzano il numero di anomalie identificate all'interno del dataset.

Per semplificare l'analisi, ci concentreremo su dati relativi al tempo di esecuzione (ELAPSED TIME) di un particolare *SQL_ID*, scelto per la sua frequente

presenza nei dati storici. Questo offre un ricco insieme di dati su cui testare il modello, permettendo di osservare come diverse configurazioni di parametri possano alterare la capacità del modello di identificare anomalie.

Utlizzando un campione composto da 10000 record salvati, si è giunti ai seguenti risultati:

Modello	**Parametro**	**Numero di anomalie**
iForest	0,01	43
	0,1	271
	0,5	1018
LOF	5	180
	10	129
	20	108
z-score	1,97	355
	3,5	162
	4,5	138

Come possiamo osservare, sono stati rilevati valori molto discordanti tra loro, sia considerando i tre modelli diversi, sia analizzando i risultati di ogni singolo modello. Questa variabilità nei risultati evidenzia la complessità intrinseca nel processo di rilevazione delle anomalie e l'importanza di scegliere con attenzione i parametri adeguati per ogni modello.

Durante le nostre analisi, gli esperti del settore hanno esaminato a fondo il dataset utilizzato, applicando diverse etichette e valutando attentamente i risultati prodotti da ciascun modello. Dopo un'analisi approfondita e discussioni collaborative, è stato raggiunto un consenso sui parametri ottimali da utilizzare per ciascun modello. In particolare, per il modello Isolation Forest (iForest) è stato deciso di utilizzare un parametro di 0,1; per il modello Local Outlier

Factor (LOF) è stato scelto il parametro 5; e per il modello z-score, il parametro selezionato è stato 3,5. Questi parametri sono stati ritenuti i più idonei per garantire un equilibrio tra accuratezza e praticità nell'identificazione delle anomalie.

Dati i diversi scenari emersi dalle nostre analisi, l'attenzione si è ora focalizzata su quale modello sia il più efficiente in termini di tempi di esecuzione. Valutare questa tempistica è fondamentale perché, in un contesto operativo reale, l'efficienza temporale di un algoritmo può determinare la sua utilità pratica. Un modello che rileva anomalie con alta precisione ma richiede un tempo di esecuzione eccessivo potrebbe non essere pratico per applicazioni in tempo reale o su larga scala.

Modello	**Parametro**	**Tempi di esecuzione (s)**
iForest	0,01	9,6
	0,1	11,2
	0,5	15,1
LOF	5	3,8105
	10	3,8244
	20	3,9009
z-score	1,97	1,1006
	3,5	1,0896
	4,5	1,0596

Questo report presenta una realtà che sorprende e stravolge tutti i risultati ottenuti finora. Attraverso un'analisi dettagliata dei tempi di esecuzione dei vari modelli, emerge chiaramente che il modello più veloce è lo z-score. Questo risultato non solo conferma le aspettative, ma le supera, rivelando che lo z-score, tra i tre algoritmi considerati, è quello che richiede meno risorse di sistema.

Questa caratteristica lo rende non solo il più rapido ma anche il più efficiente dal punto di vista computazionale.

D'altra parte, l'analisi ha rivelato che il modello che presenta i tempi di esecuzione più elevati è l'Isolation Forest. Questo algoritmo, sebbene efficace nella rilevazione delle anomalie, richiede un tempo di calcolo considerevolmente maggiore rispetto agli altri modelli testati. L'Isolation Forest, con la sua complessità intrinseca e il maggiore utilizzo di risorse computazionali, riporta unità di misura dei tempi di esecuzione significativamente superiori rispetto al modello più prestazionale, lo z-score.

Capitolo 6

CONCLUSIONI

Questo segmento finale mira a delineare i risultati conseguiti seguendo gli obiettivi prefissati all'inizio di questa ricerca, che hanno motivato l'intero sviluppo di questo lavoro di tesi.

La seconda sezione ha posto la sua attenzione sulla raccolta di informazioni per ciò che concerne il concetto di Anomalia, punto di partenza per successivo lavoro svolto.

Il capitolo successivo ha analizzato il prodotto che viene più usato attualmente per il monitoraggio delle infrastrutture dati azindali, quale Oracle Database.

Le sezioni successive hanno approfondito l'analisi delle dinamiche legate alle query, in particolare osservando le circostanze in cui si generano nuovi piani di esecuzione e come tali eventi possano indicare potenziali anomalie, valutando l'impatto sulle prestazioni in termini di tempi di esecuzione e identificando le cause sottostanti.

Si va a commentare ora come il lavoro è stato sviluppato, descrivendo tutti i vari passagi e analizzando a pieno i diversi modelli utilizzati per la ricerca dei valori anomali all'inerno del dataset fornito.

È essenziale ribadire l'importanza del Real Time Monitoring, che rappresenta il cardine e il vero valore aggiunto dell'intero progetto. Mentre molte aziende si orientano ancora verso soluzioni di Near Real Time, la transizione verso il Real Time potrebbe significare un miglioramento sostanziale dell'efficienza operativa per molte realtà aziendali.

Il contributo di questa tesi si estende su diversi fronti. Primo tra tutti, i dipendenti di Mediamente Consulting, che interagendo con gli strumenti sviluppati, hanno espresso la necessità di modelli che agevolino e velocizzino il loro lavoro di consulenza, supportandoli nei processi decisionali. L'adozione di un approccio proattivo, piuttosto che reattivo, permetterebbe una pianificazione anticipata e più efficace delle azioni da intraprendere.

In maniera indiretta, anche i clienti di Mediamente trarrebbero vantaggio, interfacciandosi con servizi più rapidi ed efficienti, il che potrebbe accrescere significativamente la loro soddisfazione nei confronti dell'assistenza ricevuta.

In sintesi, gli obiettivi prefissati sono stati pienamente raggiunti, e il sistema di Anomaly Detection in tempo reale ha prodotto una procedura di alto valore, sia dal punto di vista aziendale che per il cliente. La capacità di monitorare continuamente le prestazioni di un database e di identificare tempestivamente eventuali criticità rappresenta un miglioramento sostanziale rispetto alle pratiche precedenti.

Guardando al futuro, il progetto non si conclude con questa tesi, ma vi è la chiara intenzione di continuare a svilupparlo e adattarlo entro il framework esistente, per facilitare le analisi quotidiane. Si prevedono anche ulteriori sviluppi, come l'espansione delle analisi ad altre viste di sistema che Oracle offre, e la possibile integrazione di altri modelli di Anomaly Detection che potrebbero risultare più adatti o performanti. Per esempio, l'evoluzione del deep learning apre nuove prospettive entusiasmanti per l'anomaly detection, specialmente

attraverso l'uso di architetture neurali avanzate come Autoencoder Variazionali e Generative Adversarial Networks, che possono catturare rappresentazioni latenti dei dati per identificare anomalie e, inoltre, l'adozione di tecniche di apprendimento semi-supervisionato e non supervisionato potrebbe affrontare la sfida dei dataset non etichettati, comuni in questo campo. L'adattabilità, la scalabilità e l'integrazione con tecnologie emergenti come IoT e edge computing promettono di portare l'anomaly detection verso applicazioni real-time più efficaci e efficienti, rendendo i sistemi in grado di reagire rapidamente e con precisione alle anomalie direttamente dove i dati vengono generati.

Le analisi condotte sono estremamente preziose e continueranno a essere sviluppate nel futuro prossimo.

Bibliografia

[1] *Cos'è un'infrastruttura IT?* [Pubblicato 17 giugno 2019]. 2019. URL: https://redhat.com/it/topics/cloud-computing/what-is-it-infrastructure (cit. a p. 1).

[2] *About us.* [Sito ufficiale dela società Mediamente consulting S.r.l.] 2024. URL: https://www.mediamenteconsulting.it/About-us (cit. a p. 3).

[3] Varun Chandola, Arindam Banerjee e Vipin Kumar. «Anomaly Detection: A Survey». In: *ACM Computing Surveys* (2009) (cit. a p. 6).

[4] *Medium Articles on Machine Learning.* 2022. URL: https://medium.com/topic/machine-learning (cit. a p. 6).

[5] *Individuazione delle Anomalie: da problema a opportunità.* URL: https://polarisengineeringspa.com/individuazione-delle-anomalie-da-problema-a-opportunita/ (cit. a p. 9).

[6] Andrea Minini. *L'apprendimento supervisionato.* URL: https://www.andreaminini.com/ai/machine-learning/apprendimento-supervisionato (cit. a p. 13).

[7] Adore. *Una guida completa sull'apprendimento semi-supervisionato.* 2023. URL: https://comeaprire.com/definizioni/una-guida-completa-sullapprendimento-semi-supervisionato/ (cit. a p. 13).

[8] Mahbub Alam. *Z-score for anomaly detection*. [Published in Towards Data Science]. 2020. URL: https://comeaprire.com/definizioni/una-guida-completa-sullapprendimento-semi-supervisionato/ (cit. a p. 16).

[9] *Tavola della distribuzione normale standard*. Tavola della distribuzione normale standard. URL: https://www.webtutordimatematica.it/materie/statistica-e-probabilita/tavola-normale (cit. a p. 16).

[10] MLNerds. *Local Outlier Factor for Anomaly Detection*. 2022. URL: https://machinelearninginterview.com/topics/machine-learning/local-outlier-factor-lof/ (cit. a p. 16).

[11] Zhi-Hua Zhou Fei Tony Liu Kai Ming Ting. *Isolation Forest*. URL: https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf (cit. a p. 18).

[12] *Oracle Blogs | Oracle Database Blog*. 2022. URL: https://blogs.oracle.com/database/ (cit. a p. 21).

[13] *Automatic Database Diagnostic Monitor (ADDM) in Oracle Database 10g*. [Updated: 2019-07-31]. 2005. URL: https://oracle-base.com/articles/10g/automatic-database-diagnostic-monitor-10g (cit. a p. 24).

[14] *Vista (basi di dati)*. [ultima volta il 10 set 2021 alle 17:48.] 2021. URL: https://it.wikipedia.org/wiki/Vista_(basi_di_dati)#:~:text=Le%20viste%20sono%20un%20elemento,diversi%20modi%20di%20fare%20questo. (cit. a p. 27).

[15] Vito Gentile. *Introduzione a SQL*. [Guida linguaggio SQL]. 2015. URL: https://www.html.it/pag/55229/introduzione-20/ (cit. a p. 32).

[16] Philip Wilkinson. *Univariate Outlier Detection in Python*. [Online; in data 12 Agosto 2021]. 2021. URL: `https://towardsdatascience.com/univariate-outlier-detection-in-python-40b621295bc5` (cit. a p. 32).

[17] Alessandro Tedesco. *Anomaly Detection sui Sistemi Monitorati dall'Oracle Enterprise Manager*. [Tesi di Laurea Magistrale Politecnico di Torino]. 2018. URL: `https://webthesis.biblio.polito.it/secure/8158/1/tesi.pdf` (cit. a p. 39).

[18] *SciPy documentation*. [Version: 1.13.1]. 2024. URL: `https://docs.scipy.org/doc/scipy/` (cit. a p. 43).

[19] *pandas documentation*. [Version: 2.2.2]. 2024. URL: `https://pandas.pydata.org/docs/` (cit. a p. 43).

[20] Scikit-learn developers. *Scikit-learn: Machine Learning in Python*. 2022. URL: `https://scikit-learn.org` (cit. a p. 43).

[21] Ezio Melotti. *Dizionari*. [Guida Python]. 2017. URL: `https://www.html.it/pag/15614/dizionari/` (cit. a p. 44).