POLITECNICO DI TORINO

Collegio di Ingegneria Gestionale

Corso di Laurea Magistrale in Ingegneria Gestionale

Tesi di Laurea Magistrale

Modelli di Anomaly Detection per il monitoraggio di una base di dati in Oracle

Relatore

Prof.ssa Tania Cerquitelli

Candidato

Varacalli Ilaria

A.A 2020/2021

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

Questo capitolo introduttivo, offre una breve panoramica di ciò che verrà approfondito in seguito, presentando il dominio di conoscenza e l'azienda in cui è stato svolto il lavoro.

1.1 Background

In un mondo sempre in rapido movimento, il progresso della tecnologia ha comportato un'evoluzione progressiva dei sistemi informatici: da un elevato numero di calcolatori distribuiti nel mondo ad un numero sempre più ristretto di macchine, ma con un livello di prestazioni più alto. Quest'ultima tendenza sta diventando la scelta preferita del settore IT in quanto fornisce la struttura tecnologica di base per un'ampia varietà di servizi. Oltre alle elevate prestazioni di hardware e software, si richiede flessibilità, stabilità e un tempo di attività di quasi il 100%, soprattutto per le applicazioni rivolte ai clienti. La disponibilità ininterrotta dei servizi e i volumi sempre in crescita dei dati, insieme all'implementazione di nuovi dispositivi e applicazioni di rete sono elementi caratterizzanti della moderna infrastruttura IT. La gestione di tali sistemi, sempre più complessi, ha comportato, negli ultimi anni, un innalzamento della domanda di lavoratori nel settore IT.

Queste figure professionali si trovano a gestire un sistema informatico variegato, che spazia dall'hardware al livello applicativo e si richiede loro di saper risolvere i problemi e le difficoltà in modo efficiente e veloce. Nell'ambiente sopra menzionato, affinché gli addetti ai lavori possano focalizzarsi sugli aspetti più complessi della gestione dell'IT aziendale, risulta necessaria l'ottimizzazione e l'automazione delle operazioni di manutenzione delle infrastrutture informatiche. Ne deriva, sempre per lo stesso motivo, l'interesse dimostrato in anni recenti dalle aziende per i "sistemi ingegnerizzati", ovvero soluzioni integrate, hardware e software, che consentono di eseguire carichi di lavoro cruciali dei clienti più velocemente, a costi inferiori e con maggior sicurezza, adattandosi rapidamente ai picchi di domanda in modo da ridurre i carichi di lavoro amministrativi.

Il 2020, un anno tutt'altro che tipico, ha in pochi mesi ulteriormente accelerato la trasformazione digitale, apportando dei cambiamenti rilevanti sia nel modo di vivere che di lavorare, ma ha anche evidenziato maggiormente il ruolo centrale e critico sia delle tecnologie e delle infrastrutture digitali, che dell'analisi e della predizione per il successo del business.

1.2 Azienda

Il lavoro di ricerca, oggetto di questa tesi, è stato svolto presso Mediamente Consulting, una società di consulenza operante nel settore IT, specializzata nelle tematiche e tecnologie di business analytics, ovvero nella gestione e valorizzazione dei dati. È un'azienda giovane, nata nel 2013 come start up innovativa presso l'Incubatore di imprese innovative del Politecnico di Torino, dal quale è poi uscita nel 2016. Nonostante i pochi anni di attività nel settore, ha saputo farsi strada e oggi ha 4 sedi in Italia e diversi dipendenti con vari livelli di esperienza.

I clienti principali sono medie e grandi aziende di diversi settori, dal fashion al manifatturiero, dal retail al food & beverage, ai quali offre servizi di consulenza mirati alla ricerca di soluzioni tecnologiche innovative, ma ben inseribili nella struttura e strategia aziendale.

È organizzata per business unit che corrispondono alle principali aree di specializzazione, ovvero: infrastruttura tecnologica, data integration e managment, corporate performance managment, advanced analytics e business intelligence.

Il lavoro di tesi è stato svolto presso l'area di Infrastruttura Tecnologica il cui scopo è quello di offrire ai clienti servizi di progettazione, gestione e monitoraggio dell'infrastruttura tecnologica e applicativa. Al fine di comprendere più chiaramente il contesto e le ragioni di questo progetto, è presentata una descrizione più precisa del ruolo e delle attività svolte dal team di Infrastruttura. I servizi erogati abbracciano molteplici ambiti operativi: dalla manutenzione e aggiornamento periodico dei sistemi software all'analisi delle performance e la valutazione delle prestazioni e dell'architettura. Uno dei punti di forza è sicuramente la conoscenza e la padronanza dei prodotti e delle tecnologie Oracle.

1.3 Workflow della ricerca

La ricerca è stata condotta secondo la timeline rappresentata in Figura 1.1 e si sviluppa in cinque passi successivi, partendo dallo studio del contesto aziendale e dei suoi processi per poi approfondire i software utili allo scopo, quali Swingbench e Python. La fase successive si articolano nell'individuazione, scelta e raccolta dei dati d'interesse ed utili all'applicazione dei modelli e contestualmente la sottoposizione del database a test di stress. Completata la fase di raccolta dati, questi ultimi sono stati sottoposti a preprocessing per poi essere analizzati tramite l'applicazione di modelli di classificazione e clustering. Scelti e applicati i modelli matematici considerati più

consoni rispetta all'analisi che si è scelto di fare, si è deciso di confrontare tali algoritmi sulla base di alcuni valori (Coefficiente di Silhouette, Confusion Matrix, Classification Report e ROC Curve) allo scopo di individuare il più performante ai fini della ricerca. Infine, i risultati sono stati analizzati, interpretati e presentati in forma grafica che ne consente una lettura più semplice e intuitiva anche per l'applicazione della ricerca al contesto aziendale.

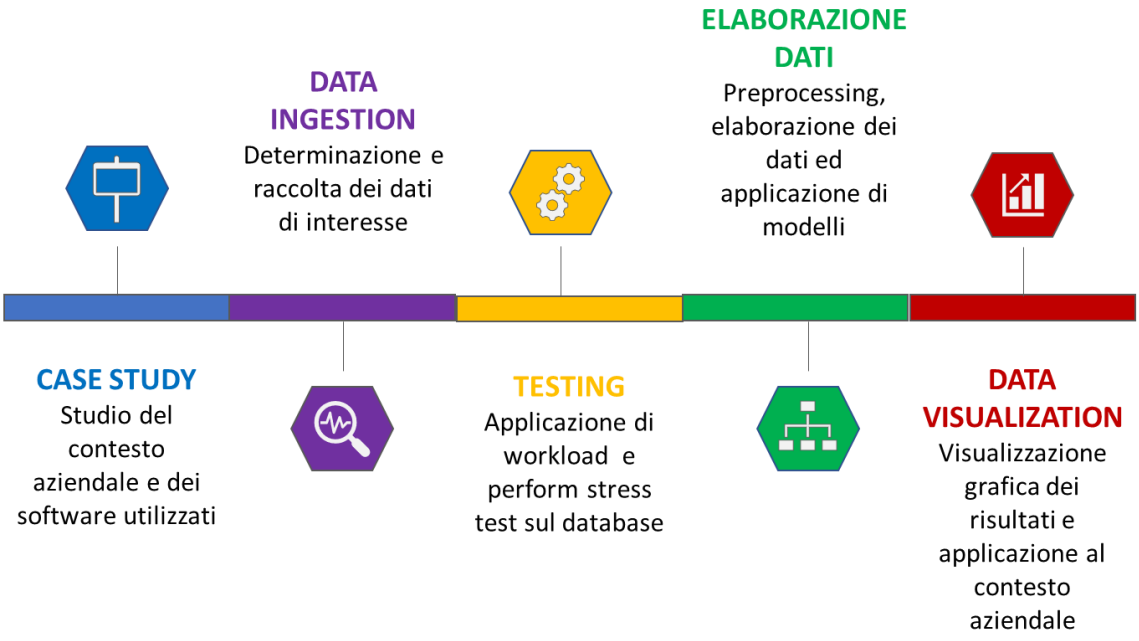


Figura 1.1: Workflow del lavoro di tesi

2 Definizione del problema

Il secondo capitolo ha lo scopo di illustrare il problema affrontato e i parametri dello specifico caso studio, insieme agli aspetti teorici, mentre gli strumenti utilizzati nello sviluppo del progetto saranno presentati nel capitolo dedicatogli.

2.1 Presentazione del problema

Il presente lavoro si inserisce nell'ottica di una sempre più accurata ottimizzazione dei processi aziendali per la gestione delle infrastrutture informatiche, in cui il monitoraggio dei sistemi informatici ne costituisce uno dei punti di valore. L'esigenza di fornire un servizio di supporto e risoluzione efficiente dei problemi richiede l'utilizzo di risorse adeguate in funzione del volume dei clienti, attuali e in prospettiva futura, e dell'eterogeneità delle problematiche che potrebbero verificarsi. Da tale necessità deriva la ricerca di soluzioni che possano migliorare l'efficienza, e di conseguenza la qualità del servizio offerto.

Per riuscire nell'intento di apportare un miglioramento al servizio, è opportuno iniziare dalla definizione di problema e da quali siano i modi e gli strumenti a disposizione per affrontarlo. In un sistema informatico, con il termine problema si intende la presenza di un malfunzionamento del sistema, corrispondente ad una variazione del normale andamento dei parametri ed una degenerazione delle prestazioni. La causa scatenante di un problema frequentemente non è unica e di semplice individuazione, ma molto più spesso concorrono un insieme di fattori e ciò ne accresce la difficoltà e le tempistiche di ricerca della soluzione. Il verificarsi di un problema può essere rilevato automaticamente dal software di monitoraggio, ad esempio nel caso in cui un parametro superi la soglia massima è inviata una segnalazione, oppure può essere comunicato dall'utente che utilizza il sistema. La fase successiva alla constatazione della presenza di un problema è un'analisi alla ricerca delle cause che lo hanno generato. Inoltre, una volta che il problema si è verificato o è stato percepito dall'utente, l'unico intervento effettuabile è a posteriori.

Il lavoro di ricerca si propone di indagare la possibilità di realizzare una prima base per uno strumento di supporto all'attività di monitoraggio in grado di garantire una valutazione tempestiva delle problematiche, fino ad un intervento proattivo, e allo

stesso tempo una visualizzazione dell'andamento delle performance utile sia in fase di analisi di un problema sia per poter anticipare il verificarsi di questo.

Durante la fase di ricerca delle cause che hanno generato il problema, l'approccio attuale è spesso basato sull'esperienza pregressa, e ciò può risultare utile per un intervento tempestivo, ma allo stesso tempo generare una certa miopia assente in un approccio data driven. Una politica di manutenzione predittiva permette di ottenere numerosi vantaggi dal punto di vista dell'efficienza, tra i quali l'identificazione di alcuni pattern che si ripropongono ripetutamente con lo scatenarsi di un evento problematico, consentendo di agire sul guasto ancora prima che esso si verifichi, andando così, per esempio, a ridurre il tempo medio di non funzionamento di un sistema. Nel dettaglio, con il presente lavoro di tesi si analizzano due aspetti paralleli. Uno è l'individuazione di gruppi di metriche, anche appartenenti a categorie differenti, che mostrano lo stesso andamento. Ciò è funzionale sia in prospettiva di un'analisi che ricerca le cause che hanno generato un problema dopo che quest'ultimo si è verificato, sia in prospettiva dell'individuazione di questo prima che si manifesti. L'altro filone della ricerca è rappresentato dalla possibilità di poter distinguere, in modo automatizzato, i momenti di stress per il sistema dai timestamp corrispondenti alla normale attività, sempre allo scopo di una manutenzione più efficacie ed efficiente.

2.2 Definizione dei parametri del caso studio

La fase preliminare di tutta l'analisi, esplicata nei capitoli successivi, è una definizione più rigorosa del problema affrontato con l'identificazione dei parametri che possono descriverne lo stato.

La definizione dello stato di un sistema non può prescindere dall'individuazione del sistema stesso, in quanto ognuno è osservato e valutato secondo i propri parametri caratteristici. Ciò significa che scelto il problema generale bisogna disporre dei dati che consentono di comprendere lo stato del sistema in tutti i suoi aspetti. Nello specifico il caso studio prende in considerazione un singolo sistema, un'istanza di una base dati relazionale (relational database management system, RDBMS) tramite metriche con informazioni sul carico del database, sulle risorse interne ed esterne utilizzate, sugli eventi di attesa ed eventuali colli di bottiglia, sull'utilizzo della CPU e le prestazioni del disco. La complessità del sistema deriva anche dalla disponibilità di un numero cospicuo di metriche, che monitorano costantemente il database in vari aspetti. Oracle fornisce una serie di viste dinamiche, ognuna contenente una moltitudine di metriche. Nonostante qualche ridondanza e la presenza di metriche che descrivono lo stesso aspetto con una frequenza di campionamento diversa o un'aggregazione per differenti intervalli, l'insieme delle metriche rimane molto ampio.

Il caso studio si compone di un ambiente creato ad hoc, costruito tramite dei test mirati. La creazione di un ambiente controllato garantisce sicuramente un maggior controllo, tuttavia ciò non significa che i risultati non siano applicabili ad un contesto

reale in quanto la scelta della tipologia dei test e il modo in cui questi sono stati eseguiti simulano in maniera piuttosto rappresentativa quello che può essere un caso reale.

Nello specifico, il caso oggetto di studio va alla ricerca di metriche, anche appartenenti a categorie diverse, con un andamento similare e indaga sulle possibilità di distinguere e classificare le metriche in base al loro andamento in momenti di stress per il database ed il normale funzionamento.

2.3 Requisiti del problema

Aspetti molto importanti da tenere in considerazione nella fase di progettazione della soluzione sono la tipologia e la qualità dei dati da analizzare e i tempi di raccolta degli stessi. Per quanto riguarda la tipologia di informazioni, che si è deciso di utilizzare, bisogna sottolineare che si tratta di soli valori numerici, corrispondenti ai valori delle metriche di database. Tali dati sono stati raccolti con una frequenza dell'ordine del minuto e memorizzati in un repository, al fine di evitare aggregazioni future automatiche di Oracle. Altro aspetto da tenere in considerazione è la quantità di informazioni da estrarre: nel caso in esame, i record raccolti corrispondono ai valori delle metriche di interesse (circa 30) per ogni istante di tempo. Esaminata la tipologia e la quantità dei dati che compongono il dataset, presentato in modo più dettagliato nel *capitolo 5 Data selection e preprocessing*, bisogna pensare anche al tempo di esecuzione dei modelli, ovvero il tempo necessario agli algoritmi per elaborare i dati in input e generare un risultato. La valutazione dei modelli si fonda sull'accuratezza e la bontà dei risultati prodotti, ma da un punto di vista più pratico, anche i tempi di esecuzione influiscono. Un modello se pur molto accurato risulta poco utile se presenta tempi di esecuzione troppo elevati. Infine, un ulteriore fattore da considerare è l'arco temporale dei dati storici da esaminare e mantenere in memoria. Tale aspetto è difficile da definire a priori in quanto può influire sia in modo positivo che negativo sulle performance del modello, ma dipende anche dal tipo di problema analizzato.

3 Stato dell'arte

In questo capitolo sono illustrate le principali tecniche nel campo del machine learning e gli algoritmi che si è deciso di utilizzare per lo studio svolto.

Il machine learning (o apprendimento automatico) è un'applicazione dell'intelligenza artificiale (AI) che si occupa della capacità di trarre dai dati determinati pattern, anche se i dati contengono degli errori (rumore), avvalendosi di algoritmi che consentono ad una macchina di acquisire conoscenza dall'analisi dei dati senza necessità di asserzioni riguardanti l'evento. I modelli di machine learning sono in grado di apprendere dai dati senza l'aiuto di un essere umano. Questa è la principale caratteristica che li differenzia dagli algoritmi classici, in cui è l'uomo a specificare il modo in cui individuare la soluzione migliore e poi l'algoritmo va alla ricerca di questa soluzione lavorando in modo più veloce ed efficace di un umano.

In letteratura esistono diversi modelli di machine learning, solitamente suddivisi in categorie in base alla natura del problema e al tipo di informazione disponibile. Una prima classificazione è sulla base della natura del dataset utilizzato in fase di addestramento, attraverso la quale si distinguono i modelli in apprendimento supervisionato, semi supervisionato e non supervisionato. Nel caso in cui ad ogni singola istanza del dataset è associata un'etichetta predefinita, allora si parla di modelli di apprendimento supervisionato, se invece non sono contenute etichette si tratta di modelli non supervisionati. Nel caso in cui una porzione di istanze presentano delle etichette, ma l'altra parte no, allora si parla di modelli di apprendimento semi supervisionati. Altro metodo di classificazione è quello basato sulla separazione delle anomalie dal dataset e distingue gli algoritmi in: metodi statistici, metodi basati sulla prossimità e metodi basati sul clustering. Di seguito, sono descritte brevemente le principali caratteristiche per ciascun modello

Il machine learning è correlato al data mining, ossia l'insieme di tecniche e metodologie rivolte all'estrazione di informazioni da grandi set di dati. Entrambi, spesso, utilizzano gli stessi metodi, ma con finalità diverse. Se il machine learning si concentra sulla previsione, il data mining va alla ricerca di proprietà sconosciute nei dati.

Apprendimento con supervisione

I modelli supervisionati sono utilizzati quando i campioni di dati, costituenti il dataset preso in esame, sono etichettati. Il processo di etichettatura, tuttavia, è piuttosto oneroso sia in termini di analisi da compiere sia in termini di tempo, in quanto è spesso effettuato a mano da un esperto del settore e di conseguenza anche piuttosto costoso. L'apprendimento con supervisione trova le associazioni esistenti fra le caratteristiche di un dataset e una variabile target. Queste associazioni consentono ai modelli di effettuare previsioni sulla base degli esempi passati, ma è anche utilizzato per ricavare le relazioni tra i dati del dataset, mostrando come e quanto le variabili si influenzano tra di loro. A partire dall'analisi di un dataset noto, l'algoritmo ne deduce una funzione attraverso la quale fa previsioni sui valori di output. Inoltre, l'algoritmo di apprendimento può anche confrontare il suo output con quello corretto e previsto e trovare errori per poi modificare di conseguenza il modello.

Apprendimento semi supervisionato

L'apprendimento semi supervisionato si colloca a metà strada tra quello supervisionato e quello non supervisionato. Esso è applicato al dataset contenenti una parte dei dati etichettati ed un'altra no, tendenzialmente la quota di dati etichettati è minore rispetto a quella non etichettati. L'idea di base risiede nello sfruttare il più possibile i dati già etichettati, vista l'onerosità sia in termini di tempo che di costi del processo di etichettatura. Inoltre, i dati senza etichetta, se usati insieme ad una piccola quantità di dati etichettati, possono produrre un notevole miglioramento nell'accuratezza dell'apprendimento. I modelli semi supervisionati si basano sull'assunzione che punti vicini hanno maggior probabilità di condividere un'etichetta e punti nello stesso cluster condividono un'etichetta.

Apprendimento non supervisionato

L'apprendimento senza supervisione accetta dataset in cui per i dati non è disponibile un'etichettatura. L'assunzione di tali modelli è che i dati seguono un certo pattern e sono in qualche modo "clusterizzabili". Il principale vantaggio, come è facilmente desumibile, è che il processo di etichettatura non è necessario, ciò significa che è molto più facile ottenere dei dati adatti ai modelli. Invece, gli svantaggi sono la difficoltà di capire se il modello si sta comportando correttamente e la perdita del potere predittivo. L'obiettivo principale dei modelli senza supervisione è trovare delle analogie e delle differenze tra i dati osservati, ricavando anche dei comportamenti analoghi tra i dati di cui un umano potrebbe non accorgersene.

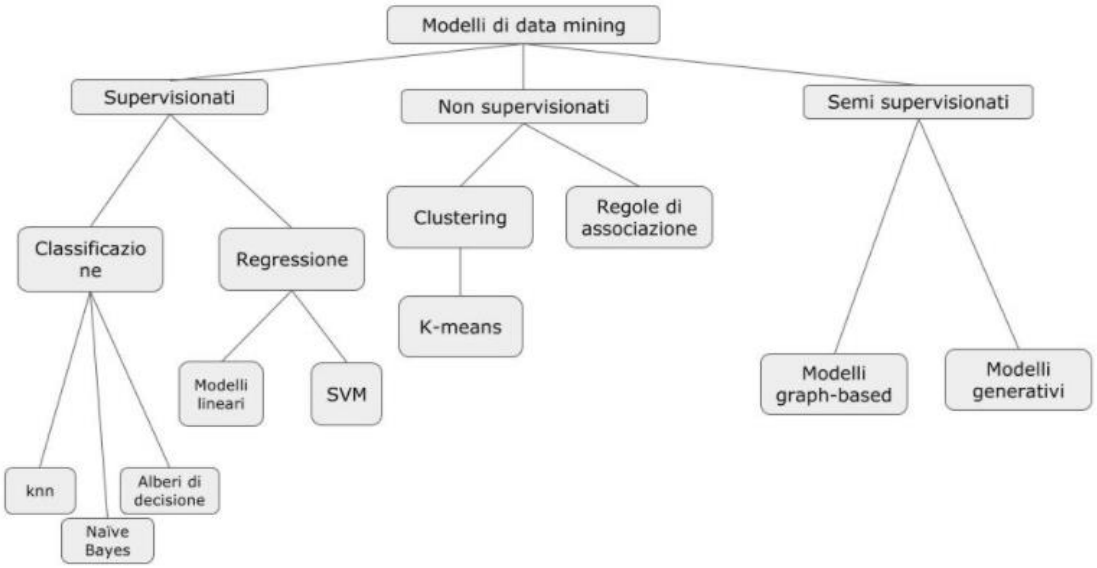


Figura 3.1: Rappresentazione schematica dei modelli di data mining [1]

Metodi statistici

I metodi statistici, o metodi basati sul modello, si fondano sull'ipotesi che i dati sono generati da processi stocastici e di conseguenza l'efficacia del modello dipende da quanto il modello statistico è in grado di adattarsi al dataset da analizzare.

Metodi basati sulla prossimità

I metodi basati sulla prossimità ipotizzano che dati da classificare come rumore sono quelli distanti dal gruppo di osservazioni più vicino. L'efficacia di questi metodi, come è facilmente intuibile, dipende dalla metrica di distanza utilizzata. Si distinguono, generalmente, in modelli basati sulle distanze, dove un'osservazione è considerata un outlier se ad una distanza superiore ad una certa soglia, e in modelli basati sulla densità, nei quali è confrontata la densità del campione con quella del vicino e se questa risulta molto inferiore allora sarà classificato come anomalia.

Metodi basati sul clustering

I metodi basati sul clustering partono dal presupposto che i dati del dataset esaminato possano essere distinti in cluster di una certa densità. L'efficacia di tali modelli, quindi, dipende molto dalla relazione che intercorre tra dati e cluster. Inoltre, molti modelli appartenenti a questa categoria hanno difficoltà nell'identificazione dei cluster nel caso di dataset con cardinalità particolarmente elevata.

3.1 Algoritmi

Questo paragrafo approfondisce brevemente gli algoritmi presi in considerazione durante lo svolgimento del lavoro di tesi. Per ciascun modello è presentata l'idea di base, le principali caratteristiche, insieme ai punti di forza e di debolezza. Gli algoritmi utilizzati sono i seguenti: K-Means, DBSCAN, Decision Tree, Random Forest e XGBoost

3.1.1 Algoritmi di Clustering

Funzionali allo scopo di individuare dei sottoinsiemi di metriche, anche appartenenti a categorie differenti, caratterizzate da trend similari, sono gli algoritmi non supervisionati ed in particolare sono stati approfonditi i modelli KMeans e DBSCAN.

3.1.1.1 K-Means

L'algoritmo k-Means è un modello di machine learning non supervisionato che mira a suddividere i dati in k gruppi sulla base dei loro attributi e delle loro caratteristiche. Per la semplicità e la velocità che lo contraddistinguono si presta bene ad essere un buon punto di partenza nell'analisi oggetto di questa tesi.

L'idea principale dell'algoritmo è definire k centroidi, uno per cluster, partendo dal numero k di cluster dato in input. Con centroide si definisce il centro di un cluster, e può essere anche considerato il punto medio del cluster. Dopo questa prima inizializzazione, il processo si articola in un loop di altri due passaggi fino al raggiungimento di un criterio di arresto. Il primo passo è l'assegnazione di un campione di dati al centroide più vicino e il secondo consiste nella generazione di k nuovi centroidi prendendo il valore medio di ogni cluster generati precedentemente. Il ciclo continua finché non si giunge ad una condizione di arresto, di solito rappresentata da una delle seguenti opzioni:

- Nessun data points cambia cluster;
- La somma delle distanze è ridotta al minimo;
- Viene raggiunto un numero massimo di iterazioni;
- I centroidi individuati non cambiano più posizione.

Input:

Dataset contenente n campioni di dati
k: numero di cluster;

1) Scegliere k centroidi iniziali
2) Per ogni punto, assegnare tale punto al centroide più vicino
3) Ricalcolare la posizione del centroide
4) Ripetere passaggi 2 e 3 fino al raggiungimento di un criterio di arresto

Output:

Insieme di k cluster;

Punti di forza dell'algoritmo:

- Algoritmo semplice, intuibile e converge velocemente;
- Non necessita di un dataset etichettato, essendo un algoritmo di clustering non supervisionato.

Limitazioni dell'algoritmo:

- Il numero di cluster k è fissato a priori;
- Molto sensibile alla presenza di anomalie e rumore del dataset;
- La scelta casuale dei centroidi può produrre di clustering diversi su diverse sequenze dell'algoritmo, quindi manca di ripetibilità e di coerenza;
- Si ottengono ottimi risultati solamente quando i cluster son ben separati e di forma sferica.

3.1.1.2 DBSCAN

L'algoritmo di DBSCAN (Density-Based Spatial Clustering of Applications with Noise) è un modello basato sulla densità, che identifica i cluster ad alta densità separati da aree a bassa densità di punti. L'idea principale del DBSCAN è il raggruppamento di oggetti simili in cluster, ossia gruppi di punti in un'area di raggio eps contenete un numero minimo di elementi. Il DBSCAN è un algoritmo di clustering non supervisionato che si propone come alternativa agli altri algoritmi di clustering, come il KMeans e il clustering gerarchico.

L'algoritmo richiede due parametri: eps, ossia il raggio di vicinanza attorno ad un dato, e il numero minimo di punti richiesto per poter considerare una regione densa, MinPts. Maggiore è il valore di epsilon minore sarà il numero di cluster prodotti, ma di dimensioni maggiori, mentre maggiore è il valore di MinPts più robusti saranno i

cluster e vi sarà minor quantità di valori anomali. Assegnati questi due parametri, si inizia da un punto non ancora visitato e si crea un cluster se sono soddisfatte le condizioni, altrimenti è contrassegnato come rumore. Tuttavia, nel caso in cui successivamente venisse trovato un cluster all'interno del quale rientra anche il punto prima etichettato come rumore, questo è assegnato al nuovo cluster e non più noise. Il processo continua con un core point non ancora assegnato, e si crea un nuovo cluster, per poi ricorsivamente cercare tutti i punti density connencted e assegnarli al cluster individuato. L'algoritmo termina non appena tutti i punti sono stati marcati come visitati, e tutti quelli non appartenenti a cluster sono considerati outlier.

Per una comprensione migliore dell'algoritmo, è utile comprendere anche alcuni concetti chiave del DBSCAN. Innanzitutto, l'algoritmo definisce tre tipi di punti:

- **Core point**: punto del dataset che contiene un numero di dati maggiore di MinPts all'interno della regione di raggio eps;
- **Border point:** punto collocato all'interno di un'area di un qualsiasi core point, ma il numero dei suoi vicini all'interno della sua regione di raggio eps è inferiore a MinPts;
- **Outlier:** punti residui, che non sono stati considerati core point né border point.

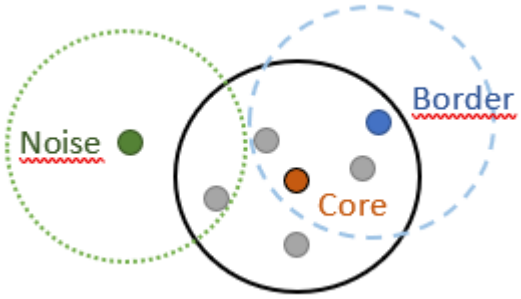


Figura 3.2: Rappresentazione di core pont, border point e outlier

Altri concetti chiave sono le seguenti definizioni:

- **Direct Density Reachable**: un punto A si può definire direct density reachable da un punto B, se A è all'interno dell'area di raggio eps di B e B è un core point;
- **Density Reachable**: un punto A è density reachable con un altro punto B se esiste un insieme di core point che vanno da B a A;
- **Density Connected:** due punti sono considerati density connected se esiste un ulteriore punto tale per cui entrambi risultano density reachable con esso.

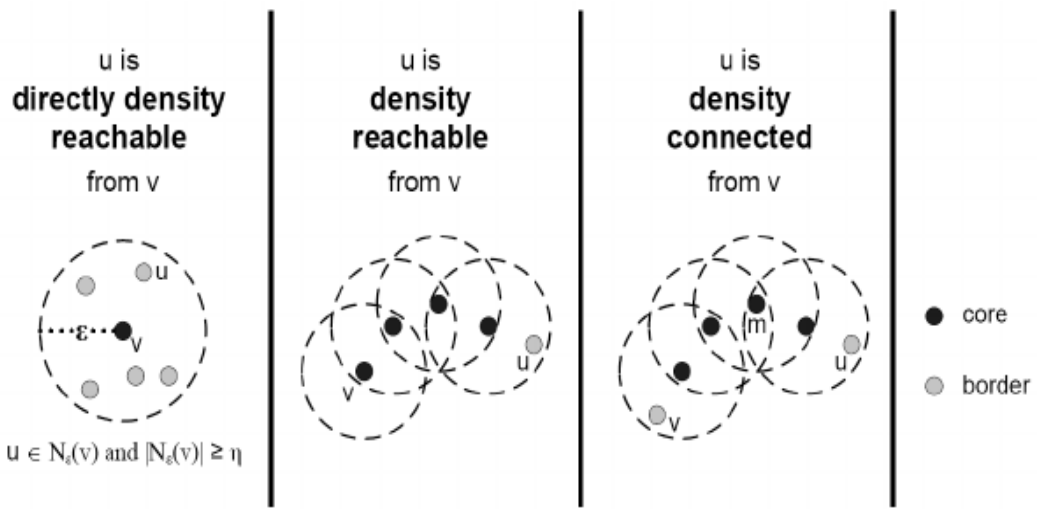


Figura 3.3: Rappresentazione dei concetti di direct density reachable, density reachable, density connected [2]

Input:

Dataset contenente n campioni di dati
eps: distanza minima tra due dati per essere considerati vicini;
MinPts: numero minimo di dati per formare un cluster

1) Calcola la distanza tra dati del dataset e marca i core point;
2) Crea un nuovo cluster al punto di core point;
3) Assegna ad ogni punto non core un cluster vicino se il cluster è un vicino (eps), altrimenti lo assegna a noise.

Output:

Insieme di n cluster;

Punti di forza dell'algoritmo:

- Non richiede di fissare a priori il numero k di cluster da generare;
- Possono essere individuati cluster di forma arbitraria;
- Identifica autonomamente gli outliers.

Limitazioni dell'algoritmo:

- L'algoritmo ha difficoltà a separare cluster di densità simili;
- I parametri eps e MinPts devono essere fissati a priori e spesso ciò non è così semplice;

- Può soffrire con dati ad alta dimensione.

3.1.2 Algoritmi di classificazione

L'altro obiettivo di questa ricerca, come già evidenziato nella definizione del problema analizzato, è la possibilità di distinguere le anomalie dal normale funzionamento del sistema, ovvero i momenti di stress del database dall'attività base. A tale scopo sono stati utilizzati algoritmi supervisionali di classificazione, ed in particolare sono stati applicati il Decision Tree, Random Forest e XGBoost.

3.1.2.1 Decision Tree

L'algoritmo di Decision Tree è uno degli algoritmi di machine learning più ampiamente utilizzato, grazie alla sua semplicità di interpretazione e alla bontà dei risultati, soprattutto con dati categorizzati. Un albero decisionale è un insieme di nodi e archi, un grafico direzionale paragonabile a un diagramma di flusso, nel quale il flusso ha inizio nei nodi radice e termina con la decisione presa nei nodi foglia. La *Figura 3.4* mostra un esempio di albero decisionale su tre livelli.

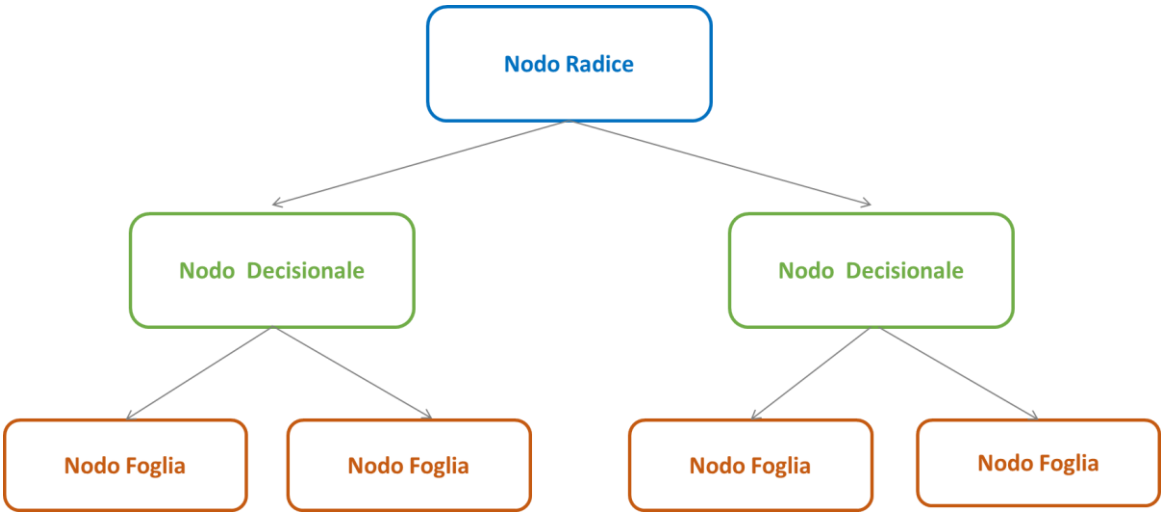


Figura 3.4: Albero decisionale

Gli alberi di decisione si differenziano in: alberi di classificazione se applicati su dati il cui risultato è di natura discreta o categorico, ad esempio la distinzione in assenza o

presenza di una caratteristica, e in alberi di regressione se utilizzati con dati di natura continua, come ad esempio un prezzo o una durata.

Decision Tree Classifier è un tipo di apprendimento automatico supervisionato e per comprendere meglio il suo funzionamento sono definite, di seguito, le sue componenti:

- **Nodi**: decisioni su un valore di un attributo
- **Rami**: una delle risposte ad un nodo, che fa anche da connessione tra nodi
- **Nodo radice**: punto di inizio dell'albero decisionale, perciò rappresenta l'intero database analizzato che poi verrà suddiviso in sottogruppi;
- **Nodo decisionale**: nodo secondario da cui partono dei rami che conducono a più sotto-nodi;
- **Nodo foglia**: nodo terminale, punto di uscita per un risultato dell'albero

Seppure ad un primo sguardo il modello Decision Tree sembra essere semplicemente un insieme di affermazioni secondo la logica if-else, in realtà, con la parte di machine learning, l'algoritmo riesce a trovare il modo migliore di costruire il modello che consente di prendere poi la decisione corretta, e predire il posizionamento di un nuovo dato.

Il funzionamento parte dall'individuazione di un nodo radice, quello con la precisione più alta possibile, attraverso il quale il dataset viene suddiviso in base a tale caratteristica. Questo processo si ripete ricorsivamente fino al raggiungimento di un criterio di stop, ovvero:

- Il nodo è puro, ossia non ha più senso dividere ulteriormente il nodo in quanto tutti i dati hanno lo stesso output e quindi si contrassegna in nodo come nodo foglia
- Raggiungimento del livello massimo, oltre il quale l'albero diventa troppo complicato e difficile da interpretare. Tale parametro è deciso arbitrariamente in base al dataset e all'esperienza

Input:

Dataset contenente n campioni di dati

1) Selezione del nodo radice;
2) Suddivisione del dataset in base alla caratteristica;
3) Individuazione di un nodo decisionale
4) Ripetere passaggi 2 e 3 fino al raggiungimento di un criterio di stop

Output:

Albero decisionale

Punti di forza dell'algoritmo:

- Semplice e poco costoso da costruire;
- Elevato potere esplicativo: output facilmente interpretabile anche da coloro che non hanno un background matematico o analitico;
- Assenza di parametri decisi a priori;
- Veloce nella classificazione dei record sconosciuti;
- Non richiede preelaborazione dei dati;
- Identifica autonomamente gli outlier.

Limitazioni dell'algoritmo:

- Instabilità: piccole modifiche nei dati di addestramento possono comportare grossi cambiamenti alla logica decisionale;
- Poco adatto a problemi complessi poiché la complessità spaziale derivante potrebbe essere proibitiva;
- Possono risultare imprecisi rispetto ad altri metodi, ma ciò è facilmente risolvibile usando Random Forest;
- Overfitting: eccesso di adattamento del modello ai dati sample, che può verificarsi soprattutto nei casi in cui il numero di parametri risulta eccessivo rispetto alla numerosità del dataset preso in esame. Questo fenomeno, però, può essere facilmente risolto imponendo dei vincoli sui parametri del modello (limitazioni sulla profondità o un taglio dell'altero);
- Gli alberi di grandi dimensioni possono essere difficili da interpretare.

3.1.2.2 Random Forest

Random Forest è un algoritmo di classificazione supervisionato composto da più alberi decisionali fatti su campioni di dati scelti casualmente. La soluzione finale è la previsione risultante migliore tra quelle degli alberi decisionali.

L'idea principale risiede nella creazione di più alberi decisionali, ciascuno basato su un campione casuale di dati estratto dal dataset in ingresso. L'utilizzo di campioni casuali indipendenti è studiato in modo tale da risolvere il problema di overfitting che spesso si genera con decision tree profondi basati su campioni molto ampi. L'insieme di questi alberi decisionali è chiamata foresta. Ogni albero è generato utilizzando criteri di selezione degli attributi, quale ad esempio l'indice di Gini (*paragrafo 3.2. Metodi di valutazione*). Nel caso di un problema di classificazione, la creazione degli alberi decisionali è seguita da una fase di votazione dalla quale poi emerge il risultato finale, ossia la classe con voto maggiore.

Maggiore è il numero di alberi che compongono la foresta migliori saranno i risultati ottenuti dal modello. Inoltre, la bassa correlazione tra gli alberi decisionali fa in modo

che si riducano gli errori individuali: anche nel caso in cui degli alberi riportino dei risultati sbagliati gli altri alberi con risultati corretti guideranno il modello nella giusta direzione. Al momento della suddivisione di un nodo, un albero della foresta sceglie tra un insieme casuale di caratteristiche, mentre un albero decisionale considera le caratteristiche che provocano la massima separazione tra le osservazioni dei nodi successivi, questa è una delle principali differenze tra i due modelli.

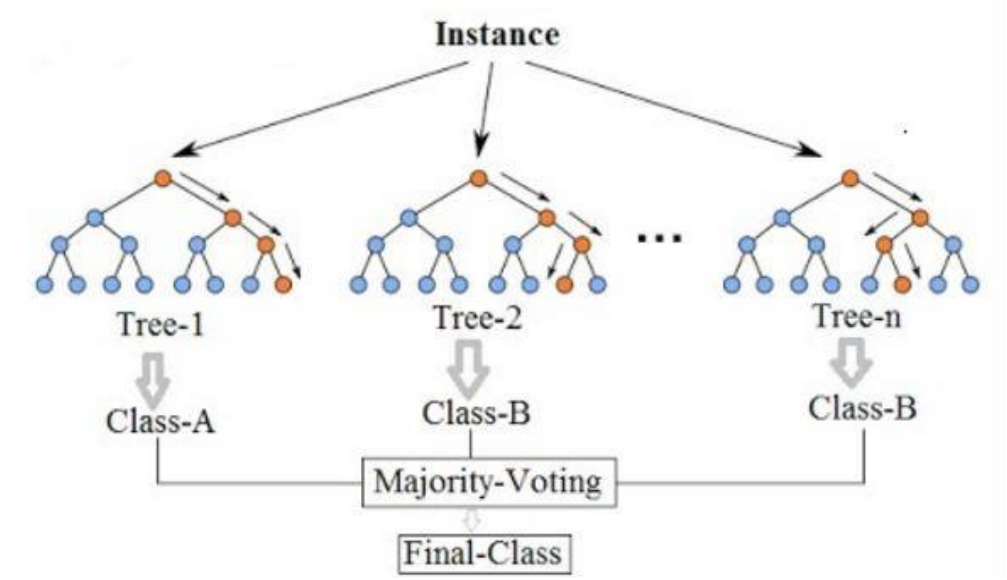


Figura 3.5: Random Forest [3]

Input:

Dataset contenente n campioni di dati

1. Seleziona campioni casuali dal dataset;
2. Costruisce un albero delle decisioni per ogni campione;
3. Esegue un voto per ogni risultato previsto;
4. Seleziona il risultato con voti maggiori.

Output:

Classificazione risultata migliore

Punti di forza dell'algoritmo:

- Accuratezza e robustezza grazie alla presenza di più alberi decisionali che partecipano al processo;

- Non soffre di overfitting, grazie all'utilizzo di campioni casuali.

Limitazioni dell'algoritmo:

- Lentezza nel generare previsioni a causa della presenza di più alberi decisionali;
- Complessità d'interpretazione: la struttura più complessa provoca maggiori difficoltà nell'interpretare il modello rispetto ai semplici alberi decisionali;
- Può richiedere un numero elevato di tentativi per migliorare il proprio apprendimento.

3.1.2.3 XGBoost

Alternativa alla creazione di un unico modello robusto potrebbe essere un approccio che genera un modello robusto a partire da un insieme di modelli più deboli. Se l'algoritmo di decision tree è già uno strumento di per sé molto potente, un insieme di alberi decisionali lo è molto di più. Inoltre, la combinazione e l'utilizzo di modelli più semplici consente una visione d'insieme maggiore.

XGBoost, acronimo di Extreme Gradient Boosting, è una tecnica di Gradient Boosting utilizzata per la costruzione di modelli predittivi sia per problemi di classificazione che di regressione, che mira a trovare il miglior modello ad albero tramite una procedura di apprendimento basata sull'aggiunta di nuovi modelli che consentono una stima sempre più accurata della variabile risposta. Tale algoritmo si è dimostrato essere particolarmente efficace se applicato con dati strutturati.

Il modello Gradient Boosting si costruisce con l'inserimento, ad ogni iterazione, di un modello debole, che aggiunto ai precedenti definisce un modello sempre più robusto. Il miglioramento delle performance generali del modello, ad ogni iterazione, è da imputare all'utilizzo dei risultati degli step precedenti per definirne i nuovi. L'aggiunta sequenziale degli alberi decisionali costituisce la principale differenza con l'algoritmo di Random Forest, che modella gli alberi in modo casuale.

La *Figura 3.6* mostra un modello di Gradient Boosting con un esempio di discriminazione di alcuni punti rispetto ad altri.

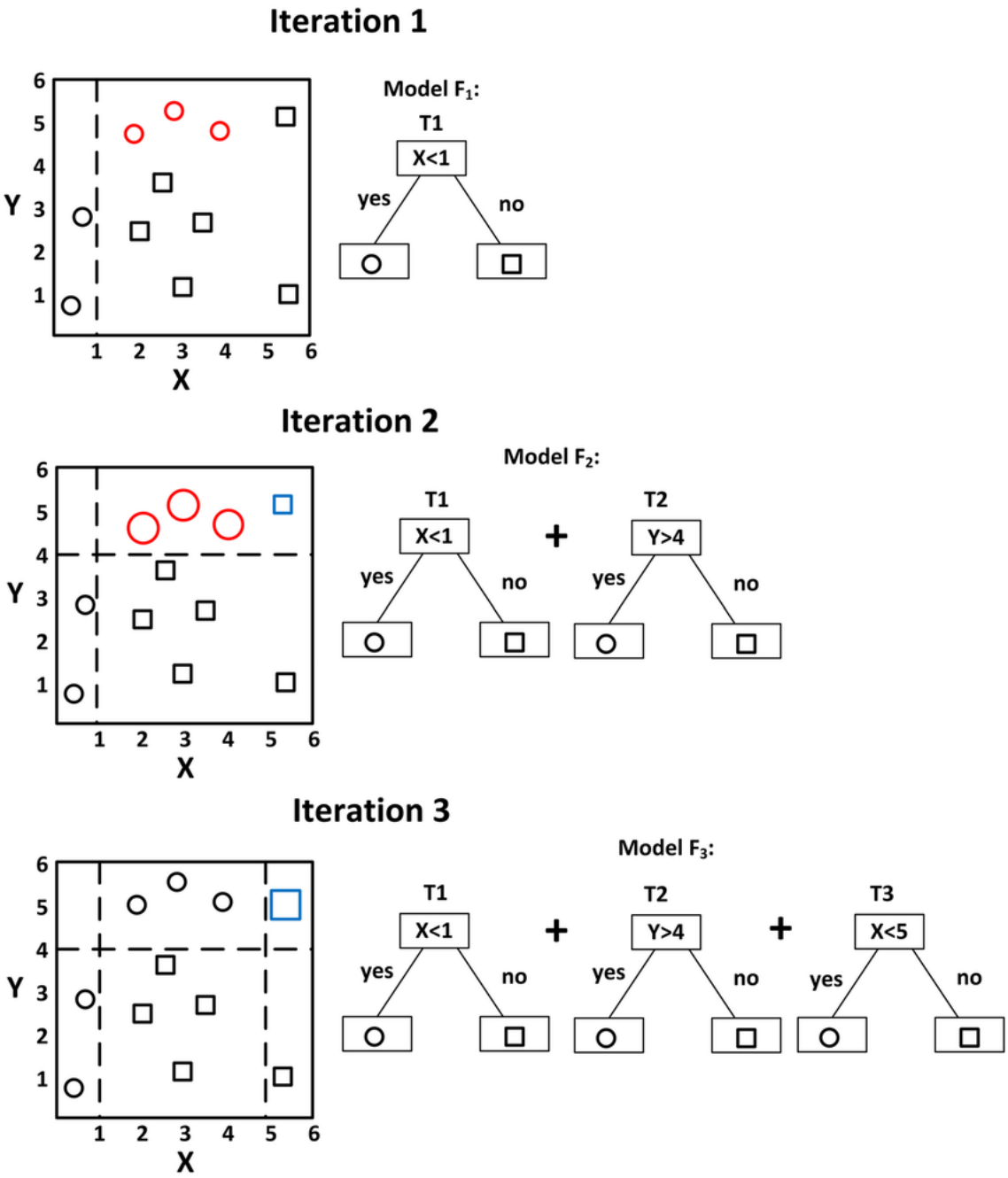


Figura 3.6: Gradient Booosting [4]

Quindi, il Boosting genera una serie di modelli consecutivamente, assegnando sempre più peso agli errori effettuati nei modelli precedenti, in fase di learning. Partendo dall'analisi di modelli più deboli, si utilizzano le conclusioni per generare modelli via via più forti. In questo modo ciascuno influisce sulla soluzione finale con un certo peso facendo sì che il modello aggregato finale abbia un'accuratezza migliore rispetto a tutti i modelli che lo costituiscono. In questo caso i modelli sono alberi decisionali, inseriti uno alla volta nell'insieme e adattati per correggere gli errori di previsione commessi dai modelli precedenti. Inoltre, si mira alla riduzione al minimo del gradiente di perdita,

il Mean Squared Error (MSE) rappresentante la differenza tra il valore predetto o classificato e il valore osservato, per questo tale tecnica è definita Gradient Boosting.

L'algoritmo Extreme Gradient Boosting si fonda sul principio base dei modelli di Gradient Boosting, ovvero il miglioramento continuo tramite l'aggiunta di modelli deboli, a cui apporta delle ulteriori migliorie sia a livello del sistema che di algoritmo:

- Ottimizzazione del sistema tramite l'implementazione parallela della costruzione sequenziale degli alberi, creando due loop paralleli, uno per l'enumerazione dei nodi foglia e l'altro per il calcolo delle caratteristiche.
- Ottimizzazione del sistema tramite il taglio degli alberi superflui, riducendo così l'onerosità computazionale, senza peggiorare le prestazioni del modello.
- Miglioramenti dell'algoritmo con la regolarizzazione, ossia la penalizzazione dei modelli più complessi, evitando anche i problemi di overfitting.
- Miglioramento dell'algoritmo con l'aggiunta di una convalida incrociata, ad ogni iterazione, per validare i risultati e la loro accettabilità.

Input:

Dataset contenente n campioni di dati

1. Esegue una previsione
2. Calcolo dei residui
3. Crea un albero xgboost
4. Definizione di una tecnica di pruning per il miglioramento delle prestazioni dell'albero
5. Calcolo dei nuovi residui
6. Ripetere gli step da 3 a 6
7. Eseguire previsione finale

Output:

Classificazione risultata migliore

Punti di forza dell'algoritmo:

- Velocità;
- L'algoritmo è parallelizzabile, sfruttando così la potenza di computer multi-core;
- Ampia varietà di parametri di ottimizzazione e convalida incrociata;
- Capacità di gestire predittori sia quantitativi che qualitativi;
- Capacità di gestire anche grandi quantità di dati.

Limitazioni dell'algoritmo:

- Necessità di conoscere e saper gestire i diversi iperparametri dell'algoritmo;
- Non può estrarre combinazioni lineari tra le variabili.

3.2. Metodi di valutazione

Per il confronto dei risultati ottenuti dall'applicazione dei modelli, illustrati nel paragrafo precedente, al dataset si è scelto di utilizzare diversi parametri, che permettono la valutazione di aspetti differenti dell'analisi svolta.

Silhouette Coefficient

Il coefficiente di Silhouette è una metrica utilizzata per valutare le prestazioni dell'algoritmo di clustering. Il valore di Silhouette per ogni osservazione si calcola nel seguente modo:

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

Con:

- a: distanza media rispetto a tutti gli altri punti del cluster
- b: distanza media rispetto a tutti gli altri punti del cluster più vicino

Il punteggio del coefficiente di Silhouette è limitato in un intervallo di valori compresi tra -1 e 1, ne consegue che sono preferibili valori prossimi a 1 poiché indicano che le tecniche di clustering hanno identificato correttamente i gruppi di dati simili tra loro. Valori prossimi a -1 sono inaccettabili, in quanto segnalano una clusterizzazione errata, mentre valori intorno allo 0 indicano la presenza di cluster sovrapposti.

Confusion Matrix

Altro parametro utilizzato nel corso della ricerca per valutare i modelli di machine learning è la confusion matrix, o matrice di confusione, che permette di comprendere le performance di un modello e se questo sia accurato ed efficace. La matrice di confusione è una tabella in cui le colonne rappresentano le previsioni e le righe lo stato effettivo. Come mostrato in *Figura 3.7*, la matrice riporta una visualizzazione esplicita del numero di campioni correttamente ed erroneamente classificati dal modello. Per definire questo parametro, e non solo, è necessario introdurre alcuni termini utili che

descrivono i quattro possibili risultati per un classificatore binario (le cui classi sono definite come “positiva” e “negativa”):

- **True positive (TP)**: campione positivo classificato correttamente dal modello
- **True negative (TN)**: campione negativo classificato correttamente dal modello
- **False positive (FP)**: campione negativo erroneamente classificato come positivo
- **False negative (FN)**: campione positivo erroneamente classificato come negativo

Più il numero di falsi positivi e falsi negativi è piccolo tanto il modello è performante.

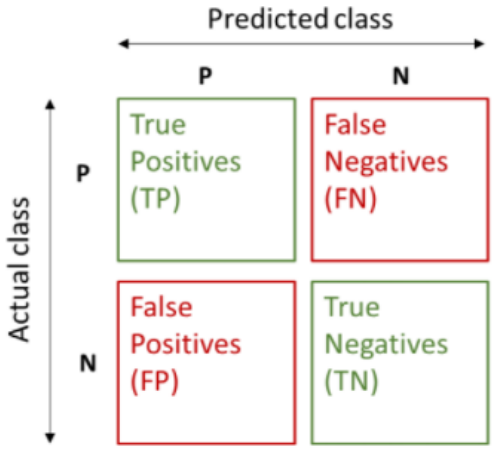


Figura 3.7: matrice di confusione [5]

Classification report

Il report di classificazione è un importante strumento per la valutazione delle prestazioni di un modello di machine learning, dopo averlo addestrato. L’output è una matrice al cui interno sono presenti i seguenti parametri:

- **Accuracy Score**
- **Precision**
- **Recall**
- **F1-score**
- **Support**

L’accuratezza di un modello rappresenta la capacità di questo di classificare correttamente sia gli aspetti positivi che negativi. Matematicamente, come riporta la formula di seguito, è il rapporto tra il numero di predizioni corrette e il totale delle predizioni effettuate. Pertanto, il valore migliore è 1 e il peggiore è 0.

$$Accuracy = \frac{TP + TN}{TN + FP + FN + TP}$$

Nel caso di analisi di dataset sbilanciati, ad esempio, il valore dell'accuratezza risulta poco significativo e si ricorre ad altri parametri, quali la precision e la recall, che consentono di calcolare la bontà di un modello a prescindere dall'eventuale sbilanciamento del dataset di input tramite l'utilizzo, nel calcolo, di una singola classe.

La precision potrebbe essere definita, in modo prettamente intuitivo, come il numero di campioni correttamente etichettati tra tutti quelli classificati dal modello come appartenenti ad una certa classe. La recall, invece, indica il valore dei campioni etichettati correttamente tra tutti quelli appartenenti ad una classe presenti nell'intero dataset.

Il punteggio di precisione del modello rappresenta la capacità del modello di prevedere correttamente gli aspetti positivi di un evento sul totale di tutte le previsioni positive effettuate, quindi matematicamente calcolato con il rapporto tra i veri positivi e la somma di veri e falsi positivi.

$$Precision = \frac{TP}{TP + FP}$$

Il punteggio di recall è una misura di sensibilità del modello, che rappresenta la capacità di questi di prevedere correttamente i positivi rispetto ai positivi effettivi. Matematicamente calcolato tramite il rapporto tra i veri positivi sulla somma dei veri positivi e i falsi negativi.

$$Recall = \frac{TP}{TP + FN}$$

Entrambi i valori sono ottimali quanto più si avvicinano a 1 poiché ciò sta a indicare che il numero di classificazioni errate (numero di falsi positivi e falsi negativi) tende a 0.

L'F1-Score è una misura utile per confrontare i modelli di classificazione puntando all'ottimizzazione delle prestazioni del modello, in quanto combina i punteggi di Precision e Recall. Matematicamente, è data dalla media ponderata delle metriche di Precision e Recall come segue:

$$F1 - Score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} = \frac{2\ x\ Recall\ x\ Precision}{Precision + Recall}$$

Sono consentiti valori compresi tra 0 e 1 e sono auspicabili valori elevati, infatti, come si può notare dalla formula, F1-Score assume valori tanto più alti quanto più sia la Recall che la Precision si avvicinano a 1, valore ottimale per entrambi i parametri.

Il Support è semplicemente il numero di occorrenze effettive della classe nel set di dati in input. Il supporto non è una misura utile per il confronto tra modelli, quanto più una valutazione interna al modello stesso poiché valori sbilanciati indicano la presenza di debolezze strutturali e la necessità di campionamenti ribilanciati o stratificati.

Sulle righe, oltre l'accuratezza, la matrice di classificazione riporta i valori di Macro avg e Weighted avg, ovvero, rispettivamente la media delle medie e la media della media pesata per etichetta di tutti i parametri sopra analizzati.

Curva ROC e AUC

La curca ROC (Receiver Operating Characteristic) è uno strumento di valutazione delle performance per gli algoritmi di classificazione binaria, attraverso una rappresentazione grafica. Essa permette di fare un'analisi qualitativa per la valutazione degli algoritmi di classificazione. Come spiegato nella matrice di confusione, nel caso di una classificazione binaria vi sono quattro possibili esiti: true positive, false positive, true negative e false negative. La curva ROC è prodotta attraverso il calcolo del tasso dei veri positivi e quello dei falsi positivi.

$$TPR\ (True\ Positive\ Rate) = \frac{TP}{TP + FN}$$

$$FPR\ (False\ Positive\ Rate) = \frac{FP}{FP + TN}$$

Il tasso dei veri positivi è una misura della probabilità con cui un elemento positivo sia classificato come tale. Il tasso dei falsi positivi rappresenta, invece, la frequenza con cui si verificano dei "falsi allarmi", ossia è la probabilità con cui un elemento negativo sia classificato come positivo.

Sebbene già la visualizzazione della curva ROC dia un aiuto per la scelta del classificatore più adatto al caso studio, spesso tale informazione è affiancata dall'AUC (Area Under the ROC Curve). Il punteggio AUC fornisce un supplemento che rende ancora più immediata l'analisi della curva ROC. Più è alto il punteggio AUC migliore è il rendimento del classificatore, più la curva è spostata verso l'alto, maggiore è il rapporto tra il tasso di True Positivi e quello di False Positive. Un valore AUC pari a 0,5 rappresenta il potere predittivo di un classificatore casuale, perciò tendenzialmente i modelli ricadono nell'intervallo tra 0,5 e 1, ma possono verificarsi casi con valori inferiori a 0,5 se le prestazioni del modello sono inferiori a quelle di un classificatore casuale.

Impurità di Gini

Il criterio di suddivisione dei nodi degli algoritmi di classificazione scelto è l'impurità di Gini. Tale coefficiente valuta la bontà del guadagno di informazioni che si ha effettuando lo split del nodo, è la misura della probabilità con cui una variabile venga erroneamente classificata. L'impurità di Gini, quindi, misura la frequenza con cui una variabile scelta casualmente non verrebbe erroneamente etichettata se avvenisse un'etichettatura casuale secondo la distribuzione delle etichette. L'impurità di Gini può essere considerata come un criterio per minimizzare la probabilità di un'errata classificazione ed è calcolato secondo la formula:

$$I_G = 1 - \sum_{j=1}^{c} p_j^2$$

L'impurità assume valore 0 quando si genera un nodo con esempi associati ad un'unica classe, quindi un nodo puro, mentre i valori superiori allo 0 sono indice di un guadagno sempre maggiore ed una migliore classificazione dei dati del dataset all'aumentare del valore dell'indice di Gini.

Altro criterio utilizzabile per la divisione dei nodi degli alberi decisionali è l'entropia, studi hanno dimostrato che per tutti gli scopi è possibile utilizzare praticamente entrambi, l'unica differenza risiede in una maggiore lentezza nel calcolo dell'entropia poiché richiede di calcolare una funzione logaritmica. Per facilitare i tempi di esecuzione si è, quindi, scelto di utilizzare Gini. [6]

K-Fold Cross Validation

L'ottimizzazione dei modelli e dei rispettivi iperparametri è una delle fasi cruciali nell'utilizzo degli algoritmi di machine learning. Per creare un buon modello è necessario assicurarsi che l'accuratezza di questo su ogni set di test sia buona quanto quella ottenuta sul set di training. La semplice suddivisione del dataset in training e test set potrebbe presentare delle difettosità nei casi in cui il modello si imbattesse in dati differenti nei due set, degradando così le sue performance. Il problema è facilmente verificabile e risolvibile con l'applicazione del K-fold Cross Validation. Tale tecnica prevede, come da consuetudine, una prima suddivisione del set di dati in training e test set, per poi procedere ad una successiva scomposizione casuale del training set in k-fold, con k scelto dall'utente. Di questi sottogruppi k-1 sono utilizzati per la fase di training del modello e il rimanente k-esimo per la fase di test. Quest'ultimo passaggio è ripetuto k volta e ad ogni replica si registra la metrica di valutazione delle performance in modo tale da poter calcolare la media e la varianza alla fine.

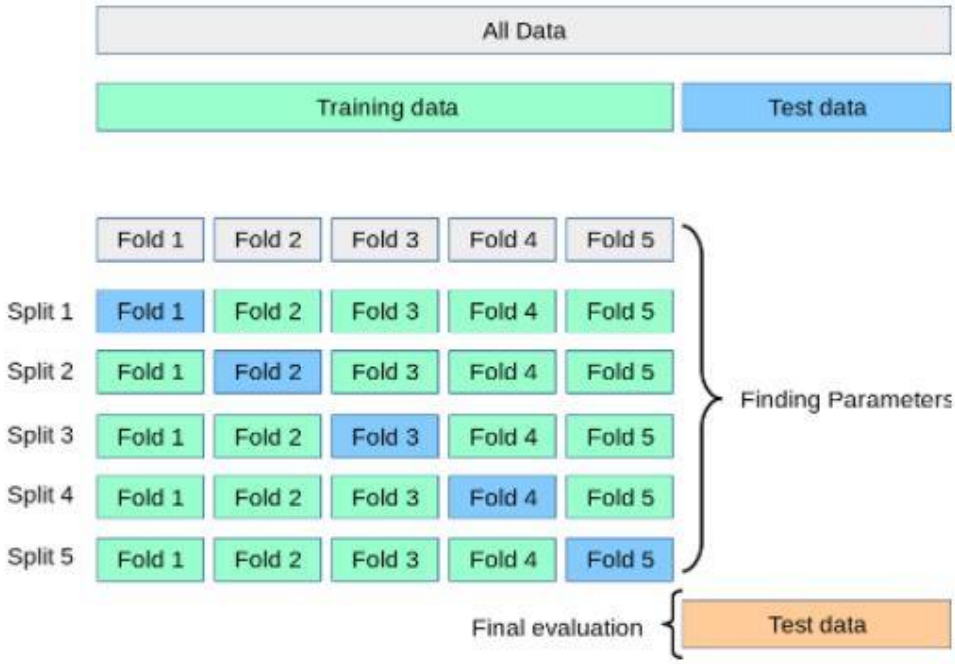


Figura 3.8: Rappresentazione del funzionamento di K-fold Cross Validation [7]

La scelta dell'utilizzo di tale tecnica di convalida incrociata deriva da un'analisi approfondita del dataset. Se ad un primo impatto si può notare la componente temporale, questa in realtà non è influente sulla generazione dei momenti di stress, ovvero il verificarsi di uno stress per il database non è dipendente dallo storico. Constatato ciò si è deciso di dare maggior importanza al valore in sé della metrica, slegato da quanto accaduto nei momenti precedenti.

4 Tecnologie utilizzate

Lo sviluppo di uno strumento software, che consente l'elaborazione e l'analisi di dati, non può prescindere da valutazioni e scelte sulle tecnologie utilizzabili. Proprio dall'adozione del linguaggio di programmazione e delle librerie da utilizzare deriva, in una certa misura, la possibilità di attuare la soluzione.

Questo breve capitolo ha lo scopo di informare sulle tecnologie utilizzate e sulle motivazioni che hanno portato al compimento di queste scelte.

4.1 Swingbench (version 2.6.0.1137)

Swingbench è un generatore di database workload e perform stress test dell'ambiente di Oracle Database. Tale strumento non è un prodotto Oracle ufficiale, ma è un'utility di benchmark creata da un dipendente, Dominic Giles. Il software consente di generare il carico tramite la creazione di uno schema e, di conseguenza utenti e tabelle, su cui eseguire delle operazioni che generano lo stress sul database.
È uno strumento basato su Java e dispone di diversi benchmark. Per il presente lavoro di tesi ne sono stati utilizzati:

- **OrderEntry** benchmark basato sullo schema "oe" fornito con Oracle e costruito per stressare le interconnessioni e la memoria tramite pesanti contention, dove diversi processi cercano di accedere contemporaneamente su un piccolo numero di tabelle

- **SalesHistory** benchmark basato sullo schema "sh" fornito con Oracle ed è progettato per testare le prestazioni di query complesse, eseguite su tabelle di grandi dimensioni.

Seppur effettuando degli stress differenti e basandosi su schemi diversi, entrambi, tramite interrogazioni sulle tabelle create con lo schema Oracle, simulano un contesto relativo alle vendite. Il primo è più improntato sull'ambito delle vendite con ricerche relative a ordini, clienti, inventari, prodotti e magazzini, mentre il secondo si concentra

sulle statistiche delle vendite dei prodotti con relativi costi e promozioni, per quali canali, a quali clienti dislocati in diversi paesi. Gli stress test consistono in una serie di interrogazioni, ognuno con una certa quota die select, insert e update.

4.1.1 Schemi Oracle

Gli schemi di database Oracle non sono altro che un insieme di strutture logiche di dati e oggetti. Nello specifico, gli schemi di esempio forniti da Oracle sono stati progettati con determinate caratteristiche, quali:

- Semplicità e facilità di utilizzo;
- Estensibilità, ovvero forniscono una base logica e fisica che si presta anche all'aggiunta di altri oggetti nell'eventualità sia necessario;
- Pertinenza, ossia sono rappresentativi degli oggetti di schema maggiormente usati per gli utenti tipici.

Tutti gli schemi si basano su una società fittizia che vende beni attraverso differenti canali ed opera in tutto il mondo. Tale società ha diverse divisioni, ognuna delle quali è rappresentata da uno schema. Come espresso precedentemente, sono stati utilizzati gli schemi OE e SH, il primo ritrae la divisione di Order Entry che tiene traccia delle vendite e delle scorte di prodotti attraverso i vari canali, mentre il secondo si riferisce alla divisione di Sales History che, invece, mantiene in memoria varie statistiche, le quali fungono da base per facilitare le decisioni aziendali.

4.1.1.1 Schema OE

Lo schema di Order Entry richiama un'azienda che vende prodotti di varie tipologie e mantiene le informazioni su questi nella relativa tabella, alcuni esempi sono la categoria, il numero identificativo ed il periodo di garanzia. Poiché i prodotti sono venduti in tutto il mondo, i nomi e le descrizioni sono conservate in diverse lingue. L'azienda gestisce più magazzini ed ognuno è registrato con nome, descrizione e ubicazione. Inoltre, sono catalogate anche le informazioni sui prodotti nei vari magazzini e la relativa quantità disponibile. Altri elementi memorizzati sono le caratteristiche dei clienti, quali nome, indirizzo e recapito. Ogni cliente ordina diversi prodotti e di ciò si tiene traccia tramite la tabella Orders con le varie indicazioni, quali lo stato dell'ordine, le modalità di spedizione e l'importo. Mentre i dati sui prodotti acquistati, le relative quantità e i prezzi unitari dei prodotti per ordine sono mantenuti nella tabella Orders_Items.

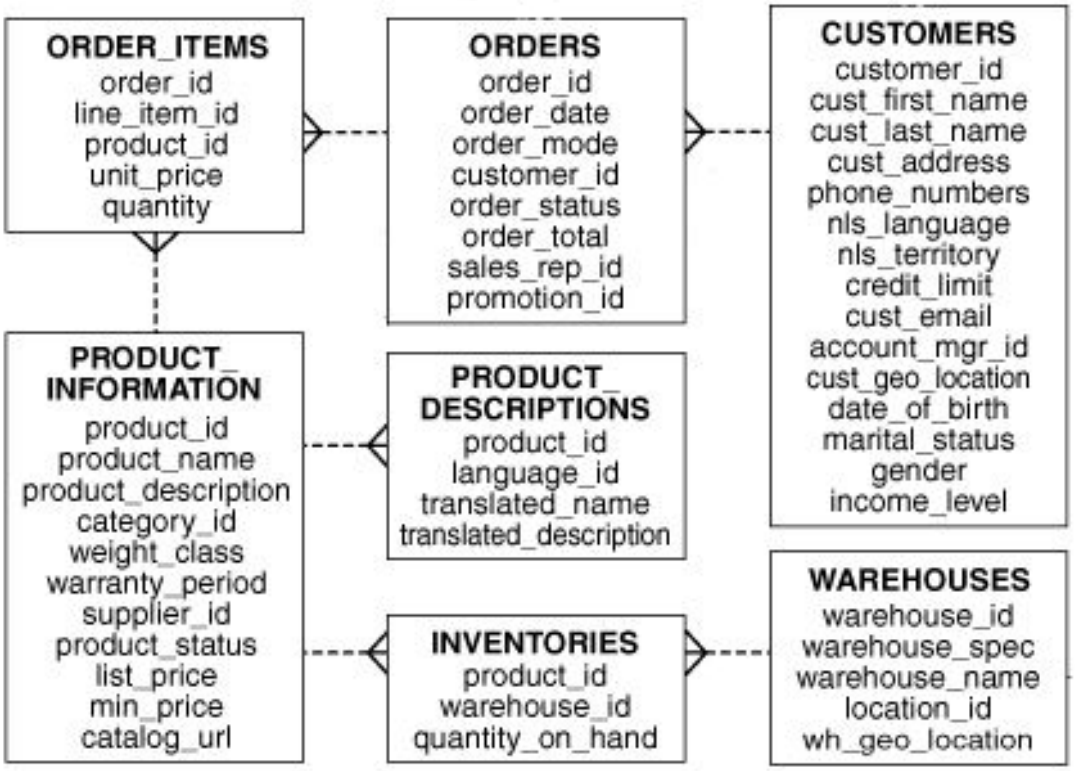


Figura 4.1: Schema OE [8]

4.1.1.2 Schema SH

Lo schema Sales History fa riferimento alla società campione che svolge un elevato volume di attività e ha necessità di statistiche aziendali che supportino il processo decisionale. Tali report statistici analizzano l'andamento e le tendenze del tempo con i volumi di vendta annuali, trimestrali, mensili e settimanali per prodotto, grazie alla presenza della tabella Times che tiene traccia dei momenti in cui sono state applicate le promozioni. Oltre le analisi temporali, è possibile osservare le vendite per area geografica ed esaminare i canali di distribuzione, attraverso le promozioni speciali sui prodotti e l'impatto che queste hanno sui volumi di vendita. Tutto ciò è realizzato tramite interrogazioni dello schema riportato di seguito, costituito da tabelle con le informazioni relative ai prodotti, alle promozioni e agli sconti, legate da relazioni in modo tale da poter ricavare lo sconto applicato per prodotto, inserito in una determinata promozione, per cliente e per canale di distribuzione. Anche le informazioni relative ai clienti e ai canali di distribuzioni e i paesi di dislocazione sono tracciate nelle relative tabelle.

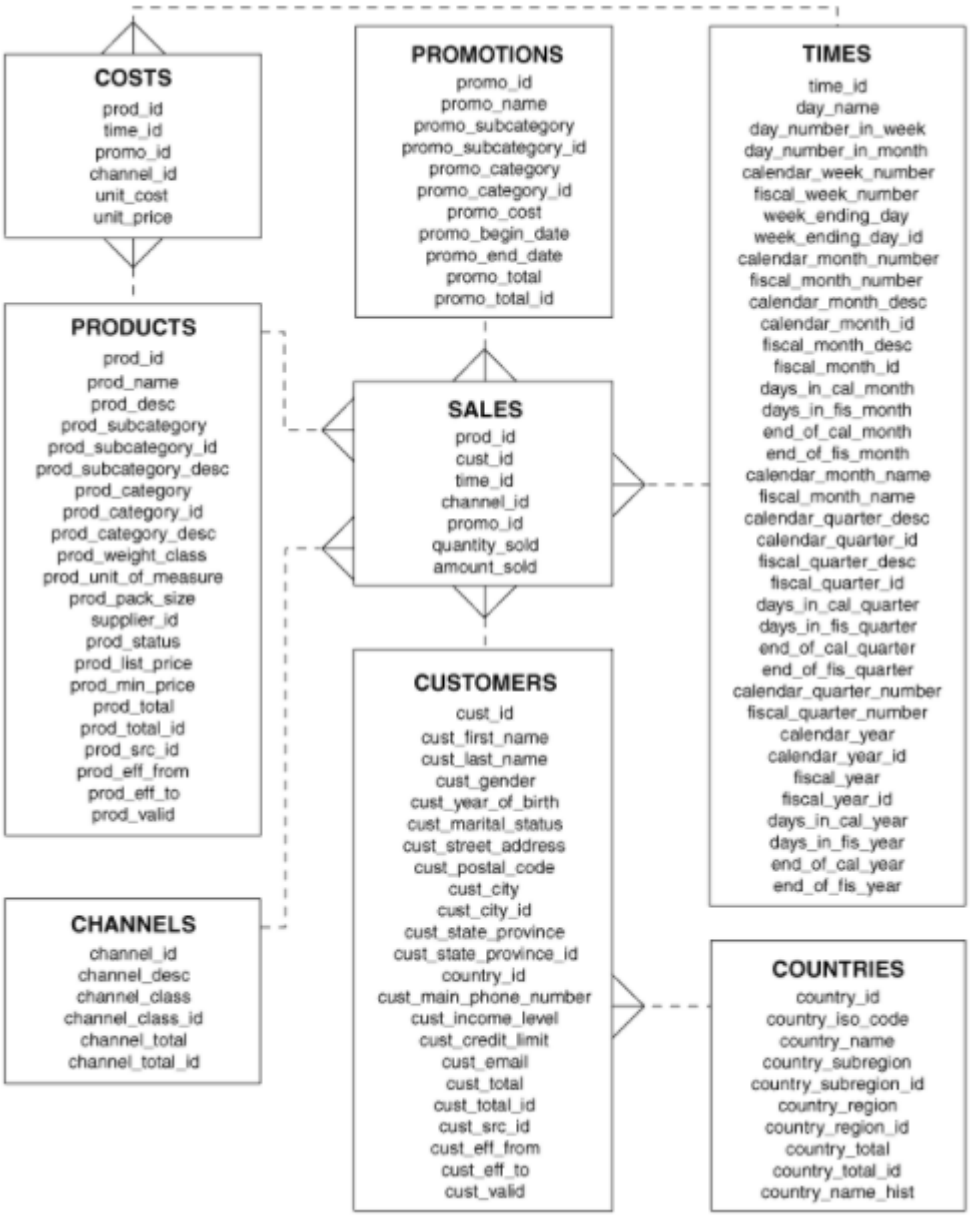


Figura 4.2 : Schema SH [8]

4.1.2 Criticità

Seppur le premesse sono molto buone, con l'utilizzo di Swingbench si sono presentate diverse criticità. In primo luogo, l'assenza di documentazione e materiale che spiegasse nel dettaglio il funzionamento e i meccanismi dei benchmark. Swingbench non è un prodotto Oracle ufficiale, pertanto non è possibile ottenere il supporto da Oracle, ma l'unica assistenza fornita è costituita dalla documentazione propria del generatore di stress test. Se sull'installazione e l'avvio dei test non è difficile trovare

del materiale a riguardo, diversa è la situazione nel caso in cui si voglia modificare e personalizzare un test.
Come espresso nei paragrafi precedenti, gli stress test dovrebbero simulare un contesto aziendale tipico con una serie di interrogazioni, costituite da query di visualizzazione, aggiornamento e inserimento dati nelle tabelle dello schema corrispondente. Proprio per rendere lo stress il più vicino possibile al contesto aziendale e simile al comportamento degli utenti, è possibile modificare diversi parametri della configurazione dello stress, quali ad esempio il numero di utenti che fanno accesso contemporaneamente ed eseguono le query, il run time del test o il tempo di attesa che intercorre tra i logon.
Dall'analisi dei risultati ottenuti da varie configurazioni su diversi intervalli temporali ci si è accorti che, nonostante le query eseguite siano di tipologie diverse, queste non risultano abbastanza di impatto da generare valori anomali che dimostrino la presenza di una situazione di forte stress per vari aspetti del database. Un esempio su tutti è l'assenza di un numero consistente di Redo Logs Switch che ci si aspetterebbe in un ambiente con molto stress in cui sono eseguite operazioni di modifica (update, insert, delete, ecc). Oracle utilizza i Redo Logs e Archive Logs per garantire che i dati del database non vengano persi, ma possano essere recuperati nel momento desiderato. Perciò, ogni operazione di modifica effettuata nel database è prima scritta nei Redo Log Buffer. Per quanto appena spiegato, in un contesto di forte stress, con molte query in esecuzione, ci si aspetta che i Redo Log si saturino velocemente e quindi vi siano diversi Redo Logs Switch.
Dalla constatazione di assenza o di un numero standard di Redo Logs Switch, nasce la necessità di modificare e personalizzare gli stress test affinché questi siano più rappresentativi di reali momenti di stress che vanno a intaccare diversi aspetti del database, visibili dall'analisi delle metriche corrispondenti.
Durante questa fase si è constatato ancora di più l'assenza di un'adeguata documentazione che spieghi nel dettaglio come agiscono i benchmark, ma anche cosa e come poter modificarli per far in modo che i test si avvicinino il più possibile alla realtà aziendale. La conseguente ricerca di risposte e autoapprendimento dei meccanismi di funzionamento del generatore di stress sono risultati tutt'altro che semplici e immediati. Il paragrafo successivo approfondisce maggiormente le motivazioni della creazione di uno stress personalizzato e come questo è stato realizzato.

4.1.3 Customer benchmark

Allo scopo di creare un ambiente di test il più vicino e rappresentativo della realtà si è deciso di utilizzate i benchmark Order Entry e Sale History forniti da Swingbench, affiancati da un costumised stress test.

Come già accennato in precedenza, Swingbench permette di modificare i test offerti, ma anche di crearne di personalizzati. La scelta di costruire un nuovo benchmark anziché aggiungere nuove query, secondo le esigenze specifiche, agli stress esistenti deriva in primis da una maggior facilità di creazione anche a causa delle difficoltà riscontrate nell'avere un'ampia visione del funzionamento del generatore di stress

test, dovute alla mancanza di documentazione, ma anche da una scelta consapevole di volerne differenziare le tipologie. Se nel presente caso studio si è deciso di creare un ambiente complesso attraverso la simulazione di tre stress differenti in contemporanea, si possono presentare anche casi diversi, semplificati e con minor variabilità rispetto a quello considerato. Quindi, in ottica aziendale, la decisione di mantenere separati le diverse tipologie di stress è funzionale ad un'applicazione futura del presente lavoro ad ambienti variegati tra di loro.

Per la costruzione del test personalizzato non è stato utilizzato uno schema di esempio di Oracle, ma è stato creato un insieme di tabelle collegate tra loro. Ognuno di esse è costituita da molteplici campi e record, sulle quali sono state costruite diverse tipologie di query particolarmente di impatto per il database.

In conclusione, lo stress test effettuato si compone di un'alternanza casuale tra i benchmark SalesHistory e OrderEntry applicati con un numero di utenti e per un intervallo di tempo diverso, in concomitanza all'esecuzione del costumised test stress anche esso settato con un certo run time e un determinato numero di utenti.

4.2 Python

Python è un linguaggio di programmazione a oggetti che coniuga molto bene semplicità d'uso ed espressività. Proprio per queste sue caratteristiche è molto utilizzato in fase di prototipazione. La preferenza nell'utilizzo di Python deriva proprio dalla possibilità di realizzare un prodotto in poco tempo e con uno sforzo relativamente basso. Inoltre, un altro fattore decisivo per la scelta di questo linguaggio è la disponibilità di un vasto numero di librerie standard e/o open-source per l'analisi dei dati e l'apprendimento automatico.
Nello specifico per il presente lavoro è stata utilizzata la versione 3.9.1 di Python e le librerie indicate di seguito insieme alla corrispondente versione:

- **Scikit-learn** (0.24.1)
  Libreria software open source nata per gli algoritmi di apprendimento automatico di Python, in particolare per l'apprendimento supervisionati e non supervisionato.

- **Pandas** (1.2.2)
  Libreria open-source che rende disponibili strutture dati particolarmente adatto per l'analisi e la manipolazione dei dati.

- **Scipy** (1.6.20)
  Libreria con numerose funzioni scientifiche utilizzare in diverse aree matematiche.

- **Matplotlib** (3.3.4)
  Libreria open-source per la creazione di grafici in due o più dimensioni.

- **Xgboost** (1.3.3)
  Libreria necessaria per l'utilizzo del modello XGBoost.

- **Numpy** (1.20.1)
  Libreria open-source ampiamente utilizzata nel campo della scienza e dell'ingegneria per lavorare efficacemente sulle strutture dati, quali matrici e array multidimensionali, grazie alle funzioni matematiche di alto livello che offre.

- **Yellowbrick** (1.3post1)
  Libreria open-source che estende l'API Scikit-learn con strumenti di analisi visiva e diagnostica, servendovi anche di matplotlib.

5 Data selection e preprocessing

La prima fase essenziale per lo sviluppo della ricerca riguarda la selezione dei dati utili allo scopo e la loro rielaborazione. Il capitolo illustra le scelte che sono state effettuate a partire dall'individuazione dei dati di interesse e la loro raccolta, ed infine le tecniche utilizzate per il preprocessing dei dati.

5.1 Individuazione dei dati di interesse

Come visto nel capitolo 2 si è voluto approfondire la possibilità di dare agli operatori impegnati nel monitoraggio delle infrastrutture un supporto ulteriore al loro lavoro, ed in particolare si è deciso di prendere in esame le metriche relative al carico di lavoro di un database.

Oracle mette a disposizione una moltitudine di metriche organizzate in viste dinamiche. Proprio la disponibilità di un elevato numero di statistiche che descrivono il database e l'andamento delle sue performance rende difficile, se non quasi impossibile a livello di tempo, l'analisi manuale di tutte le metriche per individuare le cause che hanno generato un malfunzionamento. Perciò, anche gli esperti nel settore si limitano all'esame di un sottogruppo ristretto di metriche che permettono di avere una visione abbastanza chiara, se pur limitata, della salute del database. Per la presente ricerca si è scelto di utilizzare la vista da V$SYSMETRIC_HISTORY, dalla quale sono state estratte un insieme di metriche relative a diversi aspetti del database.

Tali metriche sono serie temporali, o serie storiche, ossia sequenze di valori ordinati nel tempo e riferiti ad una variabile casuale. L'analisi di queste variabili fornisce informazioni sull'andamento nel tempo di un determinato fenomeno, sia questo il prezzo di un titolo quotato in borsa o, come in questo caso, le metriche di un sistema, la cui unità di misura dell'asse temporale dipende dalla tipologia del fenomeno. Nel caso in esame l'intervallo temporale di campionamento è di un minuto.

Come dettagliato nella tabella sottostante, sono state considerate metriche che descrivono diversi aspetti dell'attività di un database e che possono essere suddivise in tre macrocategorie come segue:

- **Database statistics**
  Metriche che forniscono informazioni sul tipo di carico del database, e sulle risorse interne ed esterne utilizzate dal database. In questa categoria rientrano gli eventi di attesa. La presenza di un'attesa potrebbe essere sintomo di un problema che influisce sulle prestazioni, con un rallentamento e in casi estremi anche con un fermo. A questa macrocategoria appartengono anche le metriche di time model, come per esempio il DB time e quelle legate alle sessioni attive.

- **Metriche del sistema operativo**
  Queste statistiche forniscono informazioni sull'utilizzo e le prestazioni dei principali componenti hardware del sistema, e di conseguenza sulle prestazioni del sistema operativo. Appartengono a questa categoria le metriche sull'utilizzo della CPU, sulle prestazioni del disco e del suo sottoinsieme di input/output (I/O), sulla memoria virtuale, principalmente per verifica che non vi sia troppa attività di paging.

- **Statistiche interpretative**
  Statistiche che aiutano a capire se vi è la presenza o meno di un problema confrontate con altri fattori o statistiche raccolte:
    - Rapporto di hit cache buffer utile a capire se esiste veramente un collo di bottiglia
    - Eventi di attesa, da confrontare con i tempi totali per capire il tempo reale impiegato per l'esecuzione di un evento e quello di attesa trascorso

E poi ulteriormente suddivise in categorie più specifiche, per una più semplice analisi e gestione, in:

- Cpu
- Sessioni
- Database
- Sql
- Logon
- I/O

Tabella 5.1: Metriche usate per il dataset

Nome metrica	Descrizione	Macrocategorie	Microcategorie
os_cpu	Host CPU Utilization (%)	sistema operativo	CPU
l_reads_s	Logical Reads Per Sec	sistema operativo	I/O
db_cpu_ratio	Database CPU Time Ratio	sistema operativo	CPU
cpu_per_s	CPU Usage Per Sec	sistema operativo	CPU
h_cpu_per_s	Host CPU Usage Per Sec	sistema operativo	CPU
read_mb_s	Physical Read Total Bytes Per Sec	sistema operativo	I/O
read_iops	hysical Read Total IO Requests Per Se	sistema operativo	I/O
read_bks	Physical Reads Per Sec	sistema operativo	I/O
read_bks_direct	Physical Reads Direct Per Sec	sistema operativo	I/O
write_mb_s	Physical Write Total Bytes Per Sec	sistema operativo	I/O
write_iops	hysical Write Total IO Requests Per Se	sistema operativo	I/O
write_bks	Physical Writes Per Sec	sistema operativo	I/O
write_bks_direct	Physical Writes Direct Per Sec	sistema operativo	I/O
db_wait_ratio	Database Wait Time Ratio	Database	DB
aas	Average Active Sessions	Database	SESSIONI
db_time	Database Time Per Sec	Database	DB
sql_res_t_cs	SQL Service Response Time	Database	SQL
bkgd_t_per_s	Background Time Per Sec	Database	DB
commits_s	User Commits Per Sec	Database	DB
redo_mb_s	Redo Generated Per Sec	Database	DB
db_block_gets_s	DB Block Gets Per Sec	Database	DB
se_sess	Active Serial Sessions	Database	SESSIONI
s_blk_r_lat	age Synchronous Single-Block Read Lat	Database	CPU
cell_io_int_mb	Cell Physical IO Interconnect Bytes	Database	I/O
logons_total	Current Logons Count	Interpretative	LOG
exec_s	Executions Per Sec	Interpretative	SQL
hard_p_s	Hard Parse Count Per Sec	Interpretative	SQL
logons_s	Logons Per Sec	Interpretative	LOGON
db_block_changes_s	DB Block Changes Per Sec	Interpretative	DB

5.1.1 Analisi trend

Il primo passo per l'analisi di serie temporali, come da consuetudine, è l'osservazione del fenomeno. Già dalla semplice osservazione dei valori assunti delle metriche è possibile intuirne l'andamento, o la presenza di pattern ricorrenti. Dunque, per tracciare l'evoluzione delle statistiche, ne vengono plottati i valori su un piano cartesiano dove l'asse delle ascisse è rappresentato dal tempo. A titolo esemplificativo è presentato nella *Figura 5.1* l'andamento della metrica Logon_s. Le figure sottostanti, *5.2 e 5.3*, invece, mostrano l'evoluzione nel tempo delle metriche appartenenti, rispettivamente, alle microcategorie Database e SQL. Sono stati riportati tali gruppi di statiche solo per una semplicità di visualizzazione essendo questi le due sottocategorie di metriche con la numerosità minore. In entrambi i casi all'interno della stessa categoria si evidenziano andamenti anche molto diversi tra di loro, sia a livello puntuale che di trend. Tuttavia, è possibile individuare due metriche che invece presentano lo stesso trend nonostante appartengano a categorie differenti (Commits_s e Exec_s).

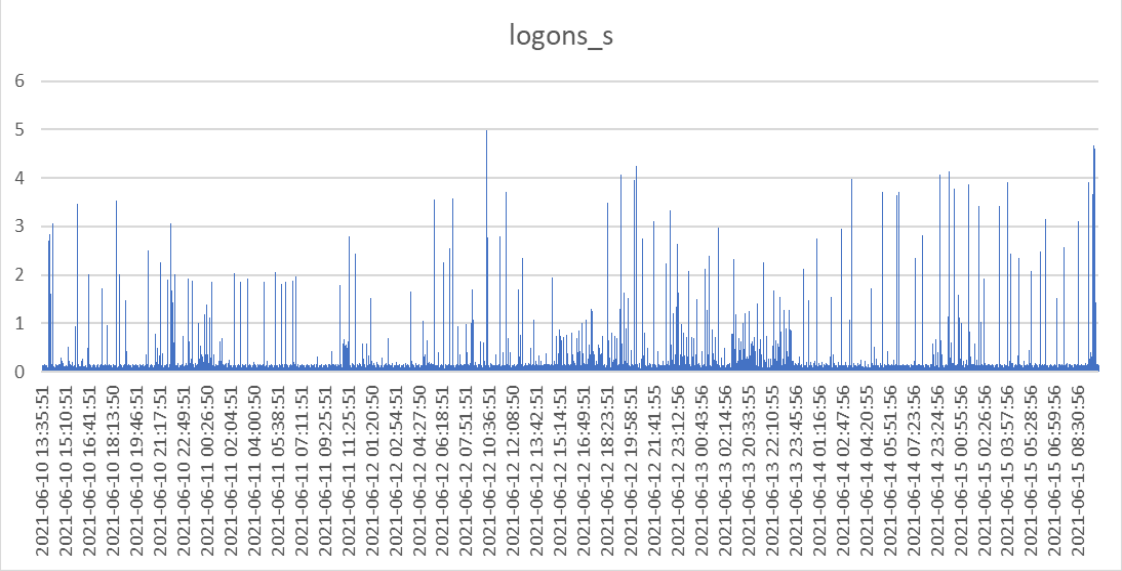


Figura 5.1: Evoluzione nel tempo della metrica Logon_s

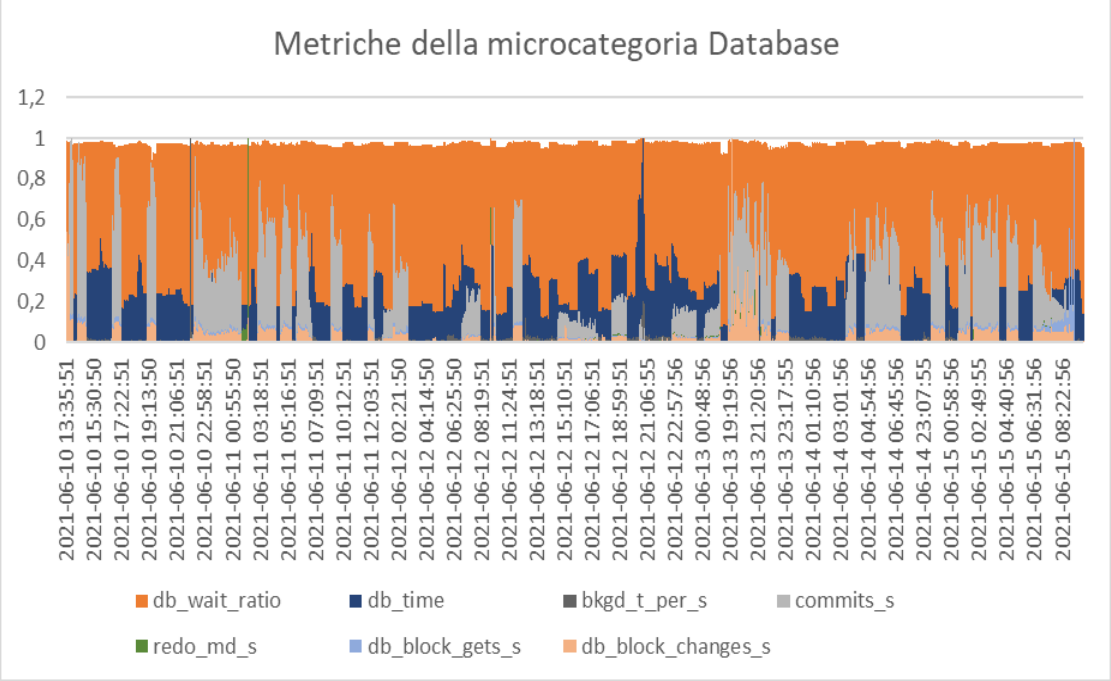


Figura 5.2: Plot dell'andamento delle metriche della sottocategoria Database

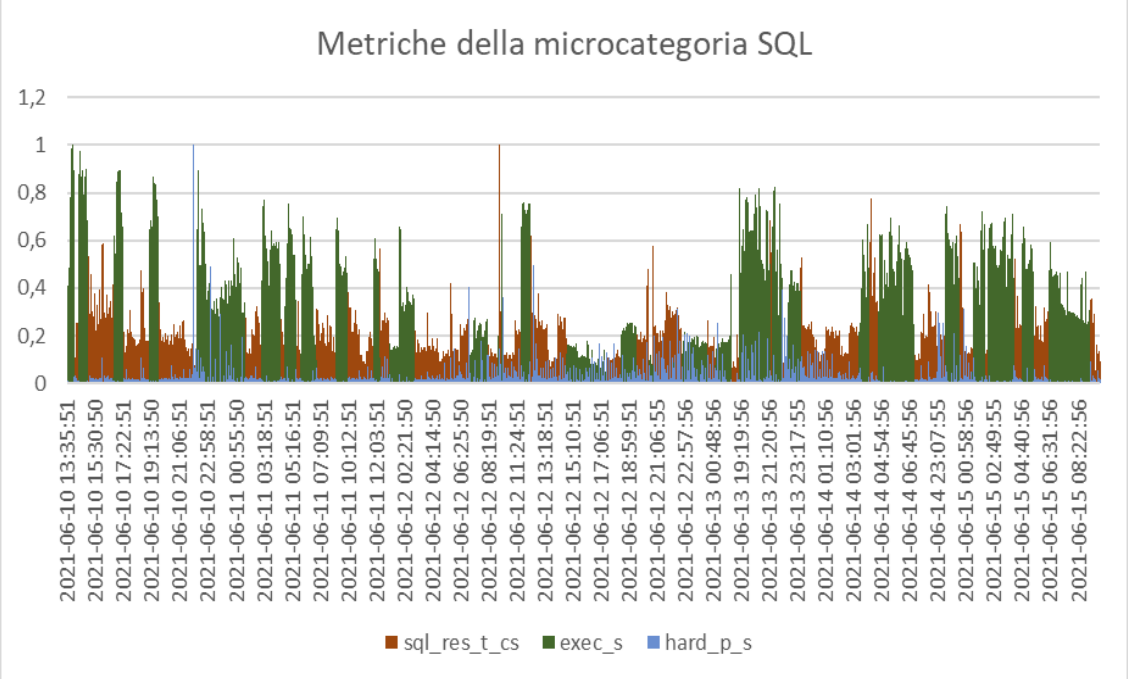


Figura 5.3: Plot dell'andamento delle metriche della sottocategoria SQL

5.2 Raccolta dati

Per la raccolta dei dati individuati è stata creata la tabella *SYSMETRIC_HISTORY_ALL* per la memorizzazione permanente e la vista *SYSMETRIC_HISTORY_LATEST*, ossia una query che determina una sorta di tabella virtuale contenente le metriche prescelte provenienti da *V$SYSMETRIC_HISTORY.* La necessità di avere una tabella che funge da repository per i dati nasce proprio dall'esigenza di utilizzare la vista *V$SYSMETRIC_HISTORY,* dynamic performance view di Oracle, così chiamata perché continuamente aggiornata mentre il database è in uso e contenete le metriche che descrivono diversi aspetti dell'attività di carico di lavoro del database. Nel dettaglio, questa vista mostra le metriche di sistema di un intervallo che spazia dagli ultimi 15 secondi alla sola ultima ora, valori poi aggregati e inseriti nella corrispondente vista dba_hist. La scelta di raccogliere i dati rilevati dalla vista dinamica deriva dalla volontà di non utilizzare dati aggregati per ora o giorno nelle statistiche, meno funzionali per un'analisi puntuale e per un eventuale applicazione in futuro di algoritmi di forecast.

La primary key sul campo date begin consente di evitare duplicati in caso di ricaricamento degli stessi dati durante l'esecuzione della procedura per il caricamento dei dati nella tabella *SYSMETRIC_HISTORY_ALL* schedulata ogni 10 minuti.

Come si può notare dal codice della query riportato di seguito, le metriche sono state raggruppate per fascia oraria, ossia per il timestamp del momento a cui risale la rilevazione tramite l'utilizzo dell'operatore di pivot allo scopo di ottenere per ogni record tutte le metriche raccolte in quel dato momento.

Il dataset risultante prende in esame 29 metriche e presenta una cardinalità di 4084 records.

Listato 5.1: Query per la raccolta del dataset

```
CREATE TABLE SYSMETRIC_HISTORY_ALL
(
    "begin" DATE,
    "end" DATE,
    "os_cpu" NUMBER,
    "db_wait_ratio" NUMBER,
    "db_cpu_ratio" NUMBER,
    "cpu_per_s" NUMBER,
    "h_cpu_per_s" NUMBER,
    "aas" NUMBER,
    "db_time" NUMBER,
    "sql_res_t_cs" NUMBER,
    "bkgd_t_per_s" NUMBER,
    "logons_s" NUMBER,
    "logons_total" NUMBER,
    "exec_s" NUMBER,
    "hard_p_s" NUMBER,
```

```
    "l_reads_s" NUMBER,
    "commits_s" NUMBER,
    "read_mb_s" NUMBER,
    "read_iops" NUMBER,
    "read_bks" NUMBER,
    "read_bks_direct" NUMBER,
    "write_mb_s" NUMBER,
    "write_iops" NUMBER,
    "write_bks" NUMBER,
    "write_bks_direct" NUMBER,
    "redo_mb_s" NUMBER,
    "db_block_gets_s" NUMBER,
    "db_block_changes_s" NUMBER,
    "gc_cr_rec_s" NUMBER,
    "gc_cu_rec_s" NUMBER,
    "gc_cr_get_cs" NUMBER,
    "gc_cu_get_cs" NUMBER,
    "gc_bk_corrupted" NUMBER,
    "gc_bk_lost" NUMBER,
    "px_sess" NUMBER,
    "se_sess" NUMBER,
    "s_blk_r_lat" NUMBER,
    "cell_io_int_mb" NUMBER,
    PRIMARY KEY("begin")
);



CREATE OR REPLACE VIEW SYSMETRIC_HISTORY_LATEST  AS
select begin_time "begin", end_time "end",
   max( decode(metric_name,'Host CPU Utilization (%)',
       value, null)     )  "os_cpu",
   max( decode(metric_name,'Database Wait Time Ratio',
       value, null)     )  "db_wait_ratio",
   max( decode(metric_name,'Database CPU Time Ratio',
       value, null)     )  "db_cpu_ratio",
   max( decode(metric_name,'CPU Usage Per Sec',
       value, null)     )  "cpu_per_s",
   max( decode(metric_name,'Host CPU Usage Per Sec',
       value, null)     )  "h_cpu_per_s",
   max( decode(metric_name,'Average Active Sessions',
       value, null)     )  "aas",
   max( decode(metric_name,'Database Time Per Sec',
       value, null)     )  "db_time",
   max( decode(metric_name,'SQL Service Response Time',
       value, null)     )  "sql_res_t_cs",
   max( decode(metric_name,'Background Time Per Sec',
```

```
        value, null)    )  "bkgd_t_per_s",
    max( decode(metric_name,'Logons Per Sec',
        value, null)    )  "logons_s",
    max( decode(metric_name,'Current Logons Count',
        value, null)    )  "logons_total",
    max( decode(metric_name,'Executions Per Sec',
        value, null)    )  "exec_s",
    max( decode(metric_name,'Hard Parse Count Per Sec',
        value, null)    )  "hard_p_s",
    max( decode(metric_name,'Logical Reads Per Sec',
        value, null)    )  "l_reads_s",
    max( decode(metric_name,'User Commits Per Sec',
        value, null)    )  "commits_s",
    max( decode(metric_name,'Physical Read Total Bytes Per Sec',
        value, null)    )  "read_mb_s",
    max( decode(metric_name,'Physical Read Total IO Requests Per Sec',
        value, null)    )  "read_iops",
    max( decode(metric_name,'Physical Reads Per Sec',
        value, null)    )  "read_bks",
    max( decode(metric_name,'Physical Reads Direct Per Sec',
        value, null)    )  "read_bks_direct",
    max( decode(metric_name,'Physical Write Total Bytes Per Sec',
        value, null)    )  "write_mb_s",
    max( decode(metric_name,'Physical Write Total IO Requests Per
Sec',
        value, null)    )  "write_iops",
    max( decode(metric_name,'Physical Writes Per Sec',
        value, null)    )  "write_bks",
    max( decode(metric_name,'Physical Writes Direct Per Sec',
        value, null)    )  "write_bks_direct",
    max( decode(metric_name,'Redo Generated Per Sec',
        value, null)    )  "redo_mb_s",
    max( decode(metric_name,'DB Block Gets Per Sec',
        value, null)    )  "db_block_gets_s",
    max( decode(metric_name,'DB Block Changes Per Sec',
        value, null)    )  "db_block_changes_s",
    max( decode(metric_name,'GC CR Block Received Per Second',
        value, null)    )  "gc_cr_rec_s",
    max( decode(metric_name,'GC Current Block Received Per Second',
        value, null)    )  "gc_cu_rec_s",
    max( decode(metric_name,'Global Cache Average CR Get Time',
        value, null)    )  "gc_cr_get_cs",
    max( decode(metric_name,'Global Cache Average Current Get Time',
        value, null)    )  "gc_cu_get_cs",
    max( decode(metric_name,'Global Cache Blocks Corrupted',
        value, null)    )  "gc_bk_corrupted",
    max( decode(metric_name,'Global Cache Blocks Lost',
```

```
        value, null)    )  "gc_bk_lost",
    max( decode(metric_name,'Active Parallel Sessions',
        value, null)    )  "px_sess",
    max( decode(metric_name,'Active Serial Sessions',
        value, null)    )  "se_sess",
    max( decode(metric_name,'Average Synchronous Single-Block Read
        Latency', value, null)    )  "s_blk_r_lat",
    max( decode(metric_name,'Cell Physical IO Interconnect Bytes',
        value, null)    )  "cell_io_int_mb"
    from(
            select  value, begin_time, end_time,metric_name
            from V$SYSMETRIC_HISTORY
            where
              metric_name in ('Host CPU Utilization (%)','CPU Usage
              Per Sec','Host CPU Usage Per Sec','Average Active
              Sessions','Database Time Per Sec','Executions Per
              Sec','Hard Parse Count Per Sec','Logical Reads Per
              Sec','Logons Per Sec','Physical Read Total Bytes Per
              Sec','Physical Read Total IO Requests Per Sec','Physical
              Reads Per Sec','Physical Write Total Bytes Per
              Sec','Redo Generated Per Sec','User Commits Per
Sec','Current Logons
              Count','DB Block Gets Per Sec','DB Block Changes Per
              Sec','Database Wait Time Ratio','Database CPU Time
              Ratio','SQL Service Response Time','Background Time Per
              Sec','Physical Write Total IO Requests Per
Sec','Physical
              Writes Per Sec','Physical Writes Direct Per
              Sec','Physical Writes Direct Lobs Per Sec','Physical
              Reads Direct Per Sec','Physical Reads Direct Lobs Per
              Sec','GC CR Block Received Per Second','GC Current Block
              Received Per Second','Global Cache Average CR Get
              Time','Global Cache Average Current Get Time','Global
              Cache Blocks Corrupted','Global Cache Blocks
              Lost','Active Parallel Sessions','Active Serial
              Sessions','Average Synchronous Single-Block Read
              Latency','Cell Physical IO Interconnect Bytes'
                )
             )
       group by begin_time, end_time
       order by begin_time, end_time
WITH READ ONLY;
```

5.2.1 Costituzione del dataset

Il dataset fornito in input agli algoritmi di clustering è costituito semplicemente dai dati estratti dalla tabella *SYSMETRIC_HISTORY_ALL,* come appena spiegato, e poi sottoposti all'operatore di pivot, per trasporre le metriche e ad averle così sulle righe e non per colonne.

I modelli supervisionati coinvolgono anche una variabile target e necessitano di un processo di etichettatura. I dati componenti il dataset utilizzato per i modelli di classificazione sono gli stessi rispetto quelli usati per le tecniche di clustering, semplicemente con una struttura diversa. Innanzitutto, per tale dataset non è stato applicato il pivot, ma sono state mantenute le metriche sulle colonne. In contemporanea è stata aggiunta una nuova colonna contenente una variabile booleana, indicante la presenza o meno di un momento di stress. L'individuazione dello stato di stress per il database è avvenuta attraverso l'analisi degli strumenti di monitoraggio dei database già in uso in azienda. Nel dettaglio sono stati etichettati come Stress tutti i timestamp per i quali è stata rilevata un'anomalia, sia essa un rallentamento del funzionamento o il superamento di una soglia limite.

5.3 Pre-processing

Prima di applicare un qualsiasi modello di Data Mining, generalmente, è necessaria una fase di preprocessing dei dati raccolti in modo da trasformare il dataset in un uno idoneo all'applicazione delle tecniche scelte. Oltre il miglioramento della qualità del dataset, la fase di preprocessing ha come obiettivo la riduzione della cardinalità e della dimensione del dataset per il conseguente miglioramento delle performance e dell'accuratezza dei modelli. In questa sezione sono presentate le tecniche utilizzate sui dati estratti.

5.3.1 Data Cleaning

La prima fase di data pre-processing è il cosiddetto "data cleaning", termine con il quale si indica un insieme di processi atti a migliorare la qualità dei dati per consentirne poi un'analisi migliore. Il dataset esaminato non presenta campi con valori mancanti, né andamenti stazionari o periodici o metriche con frequenze di campionamento molto diverse tra loro, di conseguenza non è stato necessario applicare alcuna tecnica di raffinazione dei dati. L'assenza di tali problematiche, generalmente presenti, è da imputare nella scelta di raccogliere i dati da una vista v$ e non dall'aggregato nella dba_hist (*paragrafo 5.2 Raccolta dati*), e dalla creazione di un "ambiente controllato" eseguendo direttamente degli stress test se pur simulativi di quello che è il comportamento tipico.

5.3.2 Data Transformation

Altro aspetto del pre-processing dei dati è la fase di "data transformation", termine con la quale si indicano l'insieme delle tecniche atte a modificare i valori del dataset per renderli tra loro omogenei. Il ridimensionamento dei dati è una parte essenziale del machine learning e le due tecniche più utilizzate sono la normalizzazione e la standardizzazione. La normalizzazione min-max garantisce che tutte le caratteristiche abbiano la stessa identica scala, ma il suo svantaggio principale risiede nell'incapacità di gestire correttamente i valori anomali. La standardizzazione, invece, riesce a gestire i valori anomali, ma non produce dati esattamente con la stessa scala. Tramite la Normalizzazione, o Min-Max Scaling, si ridimensionano i dati in un intervallo fisso, tra 0 e 1, secondo a formula:

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

Anche la standardizzazione è un processo di ridimensionamento degli attributi, ma secondo la formula:

$$z_{std} = \frac{\mathrm{x_i} - \mu}{\sigma}$$

dove μ è la media relativa alla metrica considerata, e σ la corrispondente deviazione standard.

Accanto a queste due tecniche di feature scaling si è deciso di testarne un'altra in cui i valori dei dati sono riportati in un intervallo tra -1 e 1, secondo la formula:

$$z_{perc} = \frac{x - \mathrm{soglia}}{\mathrm{soglia}}$$

Tale tecnica esprime una sorta di percentuale di superamento della soglia di allarme sia in positivo che in negativo. Soglie individuate tramite l'osservazione dell'andamento delle singole metriche, dei valori più frequenti e prove empiriche.

La scelta della tecnica di data transformation è dipesa in primo luogo dal dataset e dai modelli di machine learning scelti da utilizzare per il presente lavoro di tesi. Di conseguenza, in vista di future applicazioni aziendali, ma anche dalla necessità di avere i dati sulla stessa scala per poterne confrontare l'andamento puntuale e non solo a livello di trend negli algoritmi di clustering, si è preferito scartare la standardizzazione in favore della normalizzazione. La successiva scelta, tra normalizzazione e la "tecnica percentuale", si è basata su un'analisi empirica delle prestazioni degli algoritmi con i due ridimensionamenti. La decisione finale è ricaduta sulla standardizzazione, sia per le prestazioni che per la volontà di limitare l'intervento umano nella scelta delle soglie, rifacendosi solo all'osservazione dell'andamento e all'esperienza pregressa.

5.3.3 Training e Test set

Altro aspetto rilevante nella fase di selezione e preprocessing dei dati è la suddivisione dei dati in training e test set. Nel machine learning si definisce training set la parte di database dedicata alla creazione del modello matematico, mentre si indica con test set la restante porzione di dataset utilizzata per la validazione e analisi del modello precedentemente generato. La scelta di come distribuire i dati influenza attivamente la ricerca, in quanto ne deriva il numero di valori che il modello dispone per l'analisi dei dati. Molti studi sulla questione dimostrano efficace la scelta di distribuire la maggior parte dei dati del dataset per il training e poi una parte più piccola per la valutazione del modello. Durante il presente lavoro di ricerca si è seguita la medesima politica di divisione dei dati, scegliendo di lasciare al training set il 70% dei valori e il restante 30% al test set. Di conseguenza partendo da un dataset di 4084 records si è ottenuto un training set costituito da 2859 records e un test test da 1225 valori.

Figura 5.4: Distribuzione dati in training e test set del dataset e delle etichette di Stress e No Stress

5.3.4 Feature selection

Altra fase di preprocessing è la cosiddetta "feature selection" (o "selezione degli attributi"). Con questa espressione si fa riferimento alle tecniche atte ad individuare gli attributi più utili ai fini dell'analisi. L'utilizzo di un numero eccessivo di attributi spesso non apporta dei miglioramenti, quanto piuttosto lede le performance dei

modelli, generando overfitting. Un modello "overfitted" segue in modo troppo preciso i dati di training set determina poca accuratezza dei risultati e peggiori prestazioni sui dati di test. La Figura 5.5 mostra un esempio di overfitting e di underfitting, il fenomeno inverso, per uno stesso generico modello.

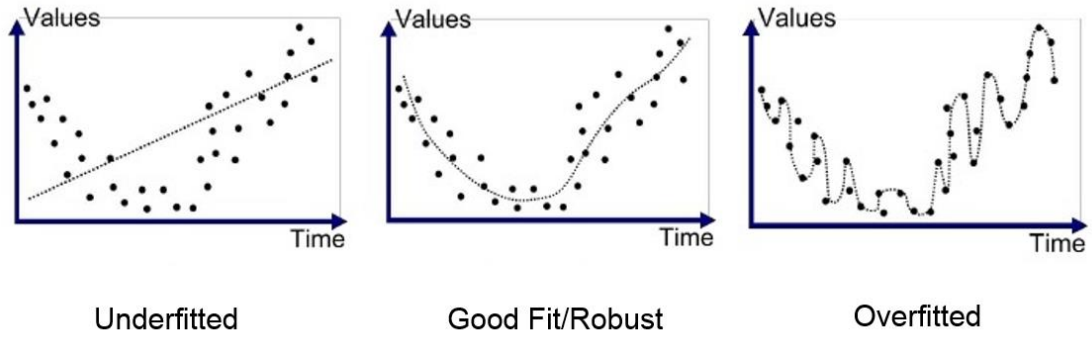


Figura 5.5: Esempio di overfitting e underfitting per un generico modello matematico [9]

Tale operazione di feature selection apporta alcuni vantaggi sui risultati dei modelli, ovvero:

- Ottimizzazione delle generalizzazioni del modello (riduzione dell'overfitting);
- Tempi di elaborazione minori;
- Semplificazione del modello e un conseguente miglioramento dell'interpretabilità dei risultati.

Tra le varie tecniche di feature selection presenti in letteratura, si è deciso di utilizzare la matrice di correlazione per l'analisi delle correlazioni tra le metriche e il calcolo delle importanze delle caratteristiche con successiva eliminazione ricorsiva secondo la tecnica "RFE" (Recursive Feature Elimination). L'obiettivo di quest'ultima è una selezione di un insieme di attributi sempre minore al fine da individuare le metriche più influenti e apportare al modello i vantaggi sopra indicati. Nel caso della ricerca sviluppata è stata applicata tale tecnica ai modelli di classificazione. La valutazione delle importanze è stata eseguita sulla base del valore di Gini Importance (*paragrafo 3.2 Metodi di valutazione*) secondo la formula che calcola l'importanza come la somma di splits che includono l'attributo, rispetto al numero totale di splits creati. Valori maggiori di importanza dell'attributo dimostrano una maggiore utilità di questo nell'applicazione del modello. I grafici seguenti illustrano il risultato della feature selection per ogni algoritmo di classificazione.

5.4 Algoritmi

Considerando il contesto aziendale e la parte di business dedicata all'offerta di un servizio di monitoraggio dei sistemi informativi e la tipologia di dataset preso in esame, gli algoritmi considerati sono: K-Means, DBSCAN, Decision Tree, Random Forest e XGBoost.

La scelta di questi modelli è dettata essenzialmente da due motivi principali:

- L'ampio utilizzo e riscontro in letteratura;
- La possibilità di sfruttare le implementazioni fornite dalle librerie di Python.

Nei problemi legati al machine learning la fase di tuning dei parametri di inizializzazione degli algoritmi è necessaria. Per ogni algoritmo i principali criteri sono:

- **K-Means:** *n_cluster* ossia il numero di cluster che l'algoritmo ha il compito di fornire;
- **DBSCAN:** *eps* e *min_samples* che indicano rispettivamente la massima distanza tra due campioni per essere considerati nello stesso cluster e il numero minimo di campioni per poter considerare una regione densa;
- **Decision Tree:** *criterion*, la scelta della funzione che misura la qualità delle suddivisioni, *max_depth*, profondità dell'albero, *min_samples_split*, numero minimo di campioni necessari per dividere un nodo interno, *min_samples_leaf*, ossia la dimensione minima delle foglie campione, *max_features,* ovvero il numero massimo di funzionalità per albero e *random_state*;
- **Random Forest:** criterion, la scelta della funzione che misura la qualità delle suddivisioni, *max_feature*, ovvero il numero massimo di funzionalità per singolo albero, *n_estimators*, il numero di alberi da costruire prima del voto massimo, *min_sample_leaf*, ossia la dimensione minima delle foglie campione e *random_state*;
- **XGBoost:** il numero di alberi (*n_estimators*) e profondità dell'albero (*max_depth*), il tasso di apprendimento (*eta/learning_rate*), il numero di campioni (*subsample*) e numero di funzionalità (*colsample_bytre*).

Per quanto riguarda gli algoritmi di clustering, il tuning dei parametri è stato eseguito cercando di massimizzare il coefficiente di Silhouette che, come descritto in precedenza, fornisce una valutazione della bontà dei cluster generati. Nel dettaglio, il numero di cluster (n_cluster) è l'iperparametro più importante per il K-Means e la ricerca del valore ottimale è stata realizzata confrontando il metodo Silhouette con il metodo Elbow. Quest'ultimo si basa sulla minimizzazione della varianza totale all'interno del cluster. Ciò è ottenibile calcolando le misure di distorsione, ovvero la media delle distanze al quadrato dai centri del cluster (within-cluster sum of square) oppure calcolando l'inerzia, ossia la somma delle distanze al quadrato dei campioni rispetto al centro più vicino (sum of square error). Dalla visualizzazione della curva in funzione del numero di cluster k si individua il valore ottimale di n_cluster nella posizione di ripida caduta (gomito della curva). Il metodo Silhouette si fonda sulla misura media della qualità di un clustering, perciò il numero di cluster ottimale si raggiunge con la massimizzazione del valore del coefficiente di Silhouette.

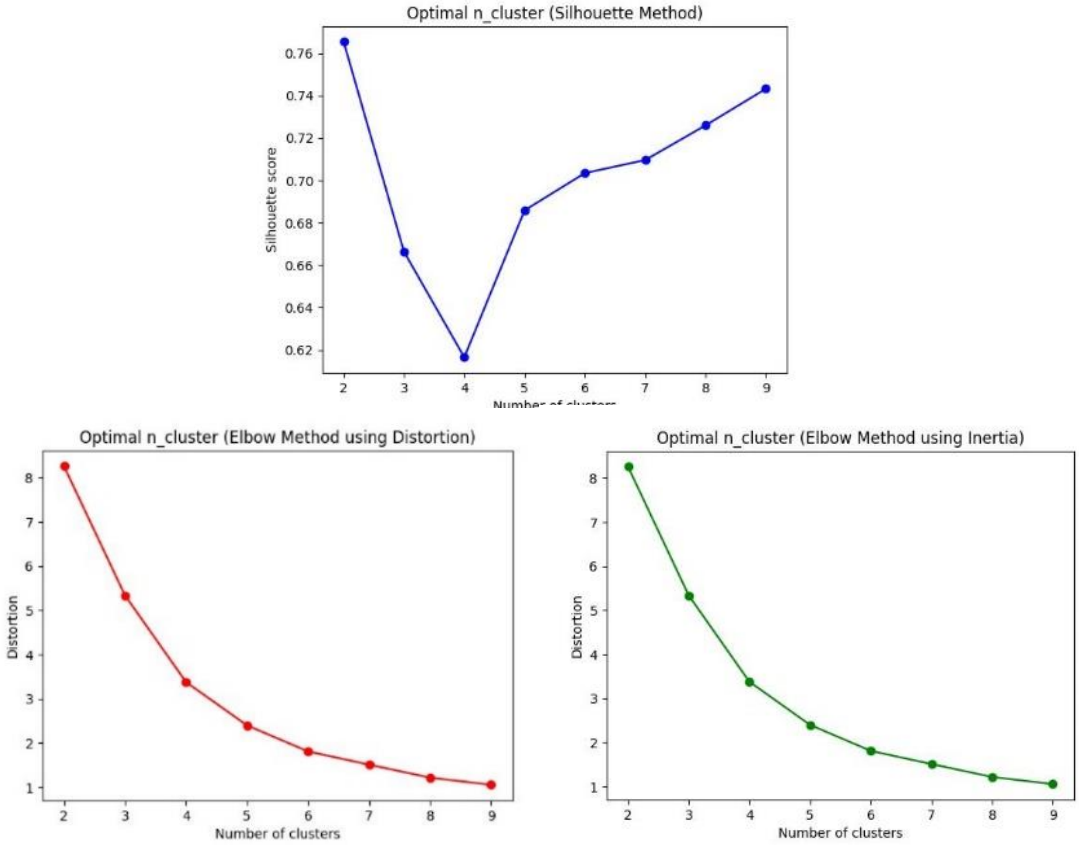


Figura 5.6: Curve utilizzate per la scelta del k durante la prima applicazione dell'algoritmo KMeans

Per quanto riguarda l'algoritmo DBSCAN, la scelta dei parametri è stata eseguita in maniera empirica sempre con l'obiettivo di massimizzare il coefficiente di Silhouette. Sono state confrontate le esecuzioni di intervalli di valori. La stima del valore ottimale di *eps* è stata individuata utilizzando l'algoritmo Nearest Neighbor Distances. Tale tecnica calcola la distanza media tra ogni punto e i relativi k vicini più vicini, con k il valore di *min_samples* selezionato, ed in base a ciò determina i cluster. Le distanze k medie sono poi tracciate in ordine crescente su un grafico, in cui il valore ottimale è nel punto di curvatura massima.

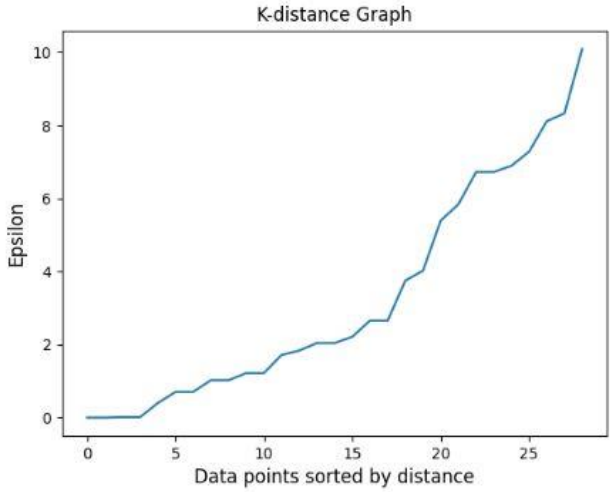


Figura 5.7: K-distance Graph utile per la scelta del valore di eps da usare nel DBSCAN

Anche il tuning dei parametri degli algoritmi di classificazione è stato eseguito in maniera empirica, ma cercando di massimizzare l'accuracy ossia, come esplicitato nel *paragrafo 3.2 Metodi di valutazione*, la capacità del modello di classificare correttamente i diversi aspetti. Per ciascun modello sono stati presi in considerazione diversi iperparametri, valutando se lasciare i valori di default o meno.

La fase di tuning dovrà essere eseguita ogni qualvolta i campioni di dati immessi in input subiscono delle modifiche. L'implementazione in scikit-learn degli algoritmi considerati fornisce due metodi principali:

- fit: riceve in input il dataset e allena il modello;
- predict: riceve in input una nuova osservazione e ne effettua una stima.

6 Applicazione dei modelli e analisi dei risultati

Conclusa la fase di raccolta dei dati e di preprocessing, il dataset è stato elaborato per mezzo dei modelli: K-means, DBSCAN, Decision Tree, Random Forest e XGBoost. In questo capitolo sono presentati e analizzati i risultati degli esperimenti eseguiti. La prima sezione si focalizza sulla presentazione dei risultati ottenuti, mentre la seconda offre una discussione sia singolarmente sia complessivamente su risultati degli algoritmi utilizzati. Come visto nel *paragrafo 3.1 Algoritmi* gli algoritmi K-Means e DBSCAN sono in grado di individuare dei cluster, mentre gli altri algoritmi supervisionati agiscono da classificatori e possono analizzare lo stato del dataset (momento di stress o meno). Per la valutazione dei risultati ottenuti si è fatto affidamento ai parametri esplicitati e approfonditi nel *paragrafo 3.2 Metodi di valutazione*. Questo capitolo vuole quindi sintetizzare i risultati delle applicazioni dei vari algoritmi sopra citati, esaminando per ciascuna categoria quale sia quello migliore per l'obiettivo.

6.1 Descrizione e presentazione dei risultati

Il lavoro di tesi nasce con l'obiettivo di individuare dei cluster di metriche con lo stesso andamento tramite i modelli di clustering e l'individuazione dei momenti di stress tramite gli algoritmi di classificazione scelti.

Durante lo svolgimento della ricerca sono state raccolte 29 metriche generando un dataset comprensivo di 4084 records. Data la non eccessiva cardinalità delle metriche si è deciso di non servirsi di tecniche di riduzione della dimensionalità, quali, ad esempio, la Principal Analysis Component. Dopo aver eseguito tutte le tecniche di preprocessing riportate nel paragrafo dedicato, a tale dataset sono stati applicati gli algoritmi di clustering, K-Means e DBSCAN, e di classificazione, Decision Tree, Random Forest e XGBoost. L'esecuzione dei modelli è seguita da una fase di valutazione secondo i criteri esplicitati nel *paragrafo 3.2 Metodi di valutazione*, ovvero il punteggio di Silhouette, la Confusion Matrix e il report di classificazione con gli indici di

accuratezza, precision e recall, la curca ROC ed infine l'interpretazione grafica dei risultati.

Una prima analisi si fonda sulle correlazioni tra le metriche attraverso il coefficiente di correlazione di Pearson, la cui matrice è riportata in *Figura 5.1*. Si evidenzia una bassa correlazione tra le metriche ad esclusione di piccoli gruppi di metriche fortemente correlate tra loro. Tendenzialmente, tali insiemi si compongono di statistiche che monitorano lo stesso elemento del database, esempi sono alcune metriche relative alla CPU o alle attività di scrittura e lettura sul database.

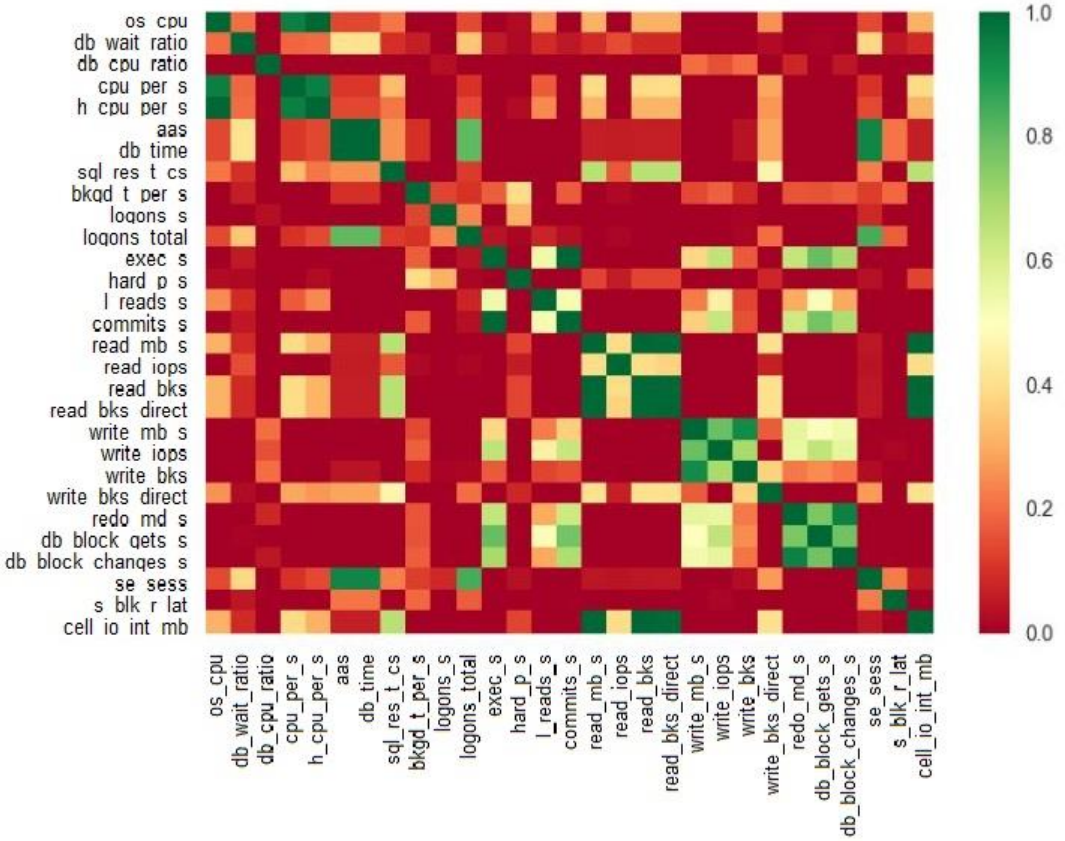


Figura 6.1 : Matrice di correlazione di Pearson tra le metriche

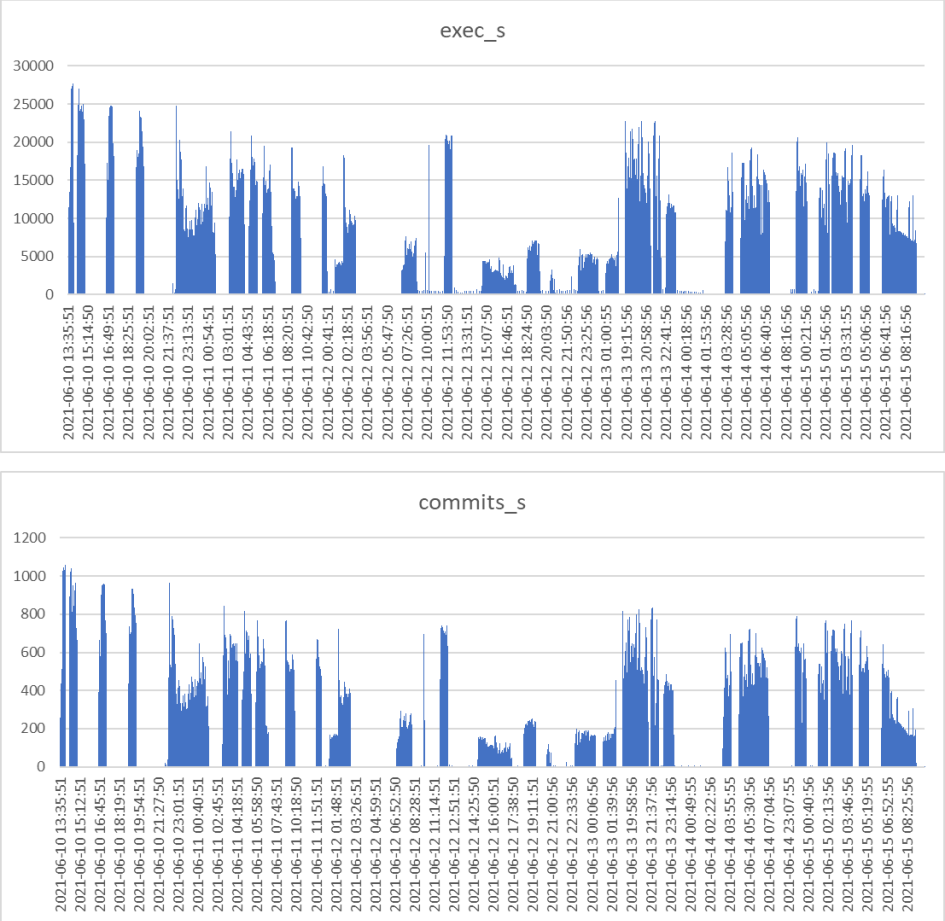
6.1.1 Clustering

La prima analisi effettuata è il clustering, attraverso gli algoritmi di K-Means e DBSCAN. Il passo iniziale è sicuramente la creazione del dataset adeguato a cui applicare poi le tecniche di preprocessing, ed in particolare la data trasformation. Lo step successivo all'esecuzione dei modelli è il tuning dei parametri e la relativa scelta, eseguita secondo i metodi specificati nel *paragrafo 5.4 Algoritmi.*

Tabella 6.1: Tuning e scelta dei parametri per Kmeans e DBSCAN e punteggi di Silhouette.

	Parametro	Intervalli	Valori	Coefficiente di Silhouette
Kmeans	K	{3, 4, 5, 6}	**5**	0,592
DBSCAN	min_samples	{2, 3, … , 8}	**3**	0,512
	eps	{2, … ,10}	**8,29**	

Nonostante gli algoritmi abbiano portato a valori diversi del coefficiente di Silhouette è interessante notare la coerenza con cui entrambi hanno classificato le metriche, creando anche cluster non solo simili, ma identici, ossia costituiti dalle stesse metriche. Dall'analisi dell'andamento delle statistiche nel tempo, suddivise per cluster, questa coerenza è giustificabile con la presenza degli stessi trend. Sono riportate, a titolo esemplificativo, l'andamento delle metriche appartenenti al cluster 4 per il KMeans e incluse negli outliers per il DBSCAN.



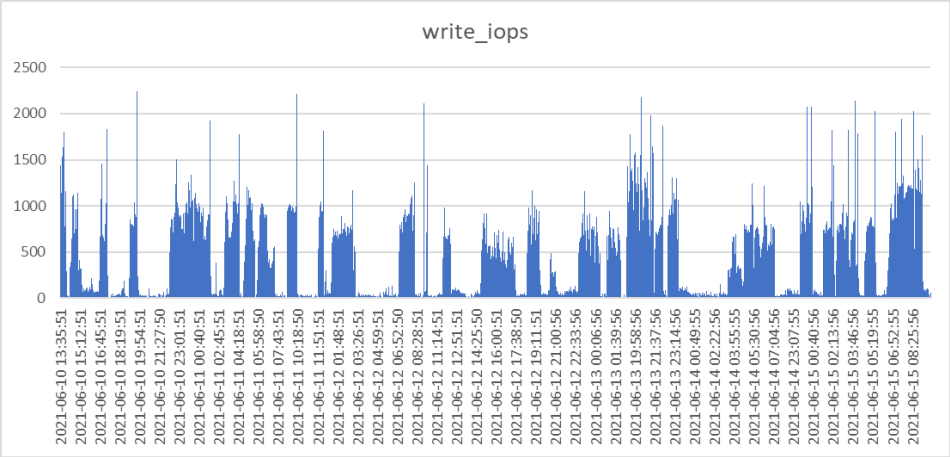


Figura 6.2: Andamento delle metriche appartenenti al cluster 4 del Kmeans e considerate come outliers dal DBSCAN insieme a l_read_s

Come è facilmente intuibile, già ad un primo sguardo queste metriche presentano dei picchi in corrispondenza degli stessi orari, ma condividono anche lo stesso trend.

La *Tabella 6.2* e la *Figura 6.3* riportano il risultato dell'applicazione dei due modelli non supervisionati e cercano di mostrare, in modo intuitivo, la coerenza prime menzionata, riconducibile a dei pattern similari presenti tra gruppi di metriche.

Tabella 6.2: Distribuzione delle metriche nei cluster individuati dai modelli di clustering

Metrics	Cluster DBSCAN	Cluster Kmeans
os_cpu	0	1
db_wait_ratio	0	1
db_cpu_ratio	1	2
cpu_per_s	0	1
h_cpu_per_s	0	1
aas	1	0
db_time	1	0
sql_res_t_cs	1	2
bkgd_t_per_s	1	2
logons_s	1	2
logons_total	1	0
exec_s	-1	4
hard_p_s	1	2
l_reads_s	-1	0
commits_s	-1	4
read_mb_s	2	3
read_iops	1	2
read_bks	2	3
read_bks_direct	2	3
write_mb_s	1	2
write_iops	-1	4
write_bks	1	2
write_bks_direct	1	2
redo_md_s	1	2
db_block_gets_s	1	2
db_block_changes_s	1	2
se_sess	1	0
s_blk_r_lat	1	2
cell_io_int_mb	2	3

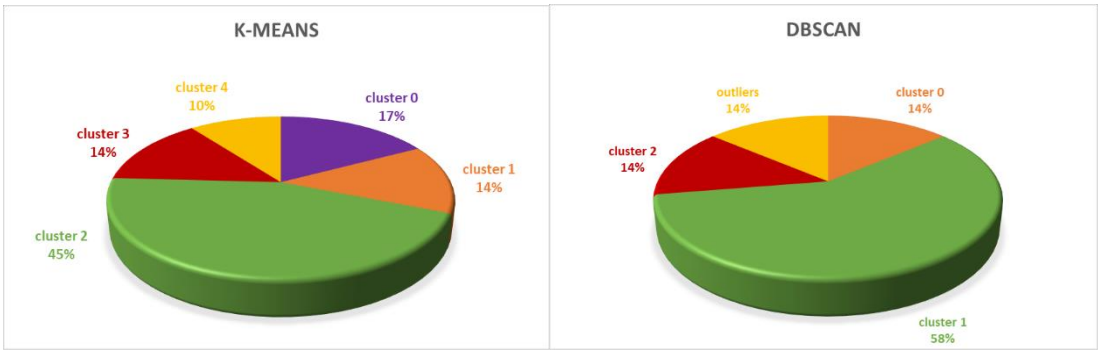


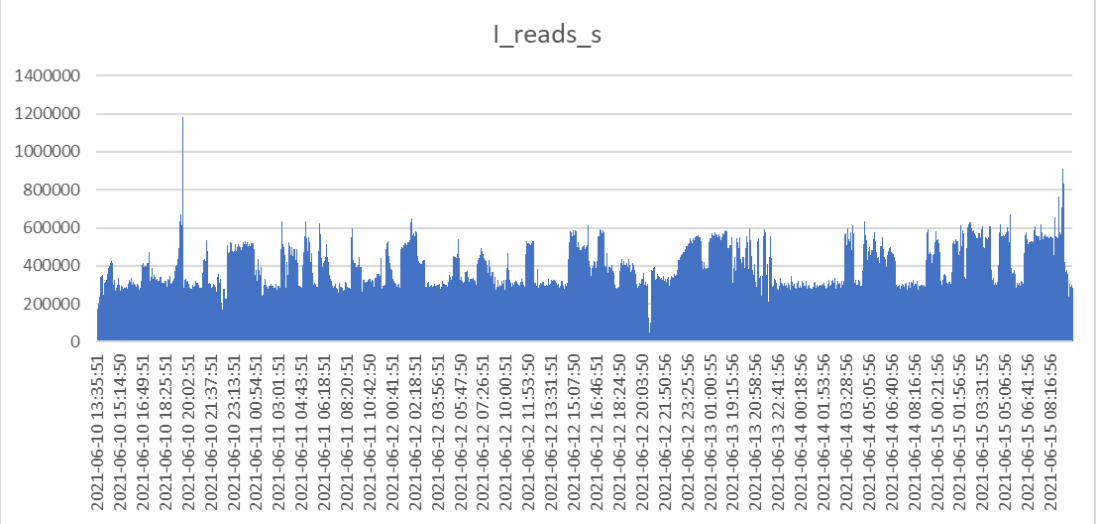
Figure 6.3: Rappresentazione grafica della distribuzione delle metriche nei cluster individuati dagli algoritmi di clustering

La presenza di trend condivisi tra le metriche appartenenti al medesimo cluster, prima menzionata, è riscontrabile in tutti i cluster ad esclusione del cluster 2 dell'algoritmo KMeans, raggruppamento contenente anche la numerosità più elevata di metriche. Si è deciso, quindi, di affinare il clustering riapplicando entrambi i modelli al sottoinsieme del dataset costituito dalle sole metriche appartenenti al cluster 2, allo scopo di verificare maggiormente le prestazioni delle tecniche prescelte, ma anche di individuare cluster di dimensioni confrontabili. Il processo di tuning e scelta dei parametri è stato il medesimo, e la *Tabella 6.3* permette di visualizzarne i risultati.

Tabella 6.3: Secondo tuning e relativa scelta dei parametri per Kmeans e DBSCAN e punteggi di Silhouette.

	Parametro	Intervalli	Valori	Coefficiente di Silhouette
Kmeans	K	{3, 4, 5}	**4**	0,487
DBSCAN	min_samples	{2, 3, 4}	**2**	0,486
	eps	{4, … ,9}	**5,89**	

Nuovamente si manifesta una congruenza nella suddivisione delle metriche in cluster tra KMeans e DBSCAN e in questo caso anche per i coefficienti di Silhouette, che differiscono solo del 0,02%.



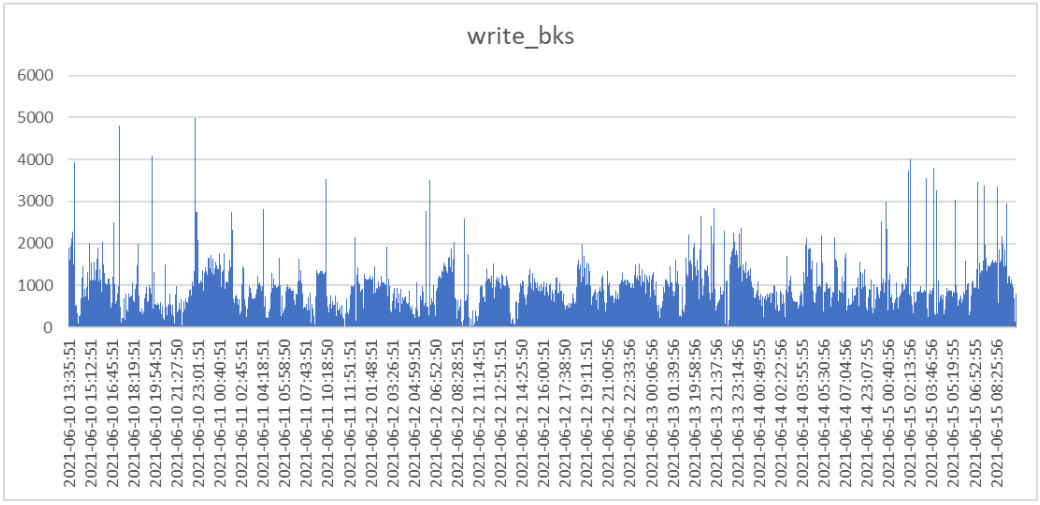


Figura 6.4: Andamento delle metriche del cluster 0 del Kmeans e cluster 1 per il DBSCAN

Tabella 6.4: Distribuzione delle metriche nei cluster individuati dai modelli di clustering

Metrics	Cluster Kmeans	Cluster DBSCAN
db_cpu_ratio	1	0
sql_res_t_cs	2	-1
bkgd_t_per_s	1	0
logons_s	1	0
hard_p_s	1	0
read_iops	3	-1
write_mb_s	0	1
write_bks	0	1
write_bks_direct	2	-1
redo_md_s	1	0
db_block_gets_s	1	0
db_block_changes_s	1	0
s_blk_r_lat	1	0

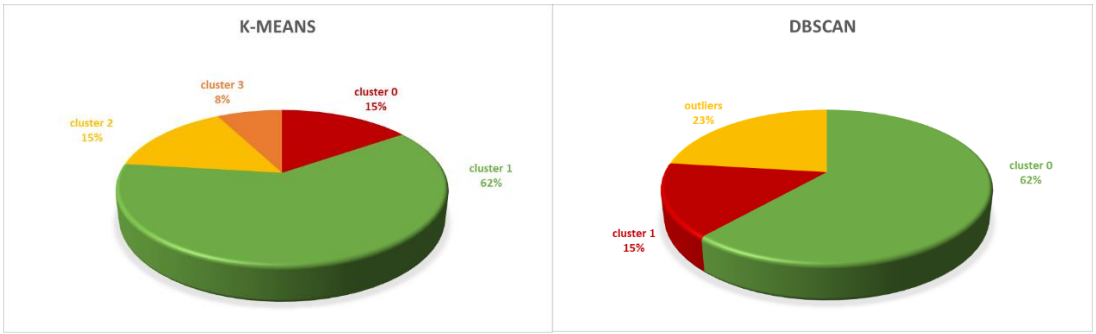


Figure 6.5: Rappresentazione grafica della distribuzione delle metriche nei cluster individuati dagli algoritmi di clustering

Altra caratteristica da sottolineare è che le metriche facenti parte di uno stesso cluster non appartengono alla stessa macro o microcategoria, anzi più spesso queste risultano essere molto distribuite. Tale constatazione rafforza l'idea iniziale, secondo la quale, per l'indagine delle cause che hanno generato un problema nei sistemi informativi o sul database, un approccio verticale su quella che l'esperienza suggerisce essere una possibile causa scatenante non è sempre efficiente ed efficace.

6.1.2 Classificazione

Per quanto riguarda i modelli supervisionati il primo è stato il Decision Tree la cui ottimizzazione dei parametri riportata nella *Tabella 6.1*, ed eseguita secondo le modalità specificate nel *paragrafo 5.4 Algoritmi*, ha permesso di ottenere un accuracy del 72%.

Tabella 6.5: Tuning degli iperparametri dell'algoritmo di Decision Tree

	Parametro	Intervalli	Valori
Decision Tree	criterion	Gini/entropy	**Gini**
	max_depth	{3, 4, 5, … ,10}	**10**
	max_features	{2, 3, 5, … ,20}	**15**
	min_samples_leaf	{2, 3, 4}	**3**
	min_samples_slit	{20, 50, 100}	**50**

Anche per gli algoritmi di classificazioni l'applicazione dei modelli è preceduta ovviamente dalla costituzione del dataset, ma anche dalla sua etichettatura e dalla successiva fase di preprocessing, ad esclusione della normalizzazione dei valori non necessaria per questa tipologia di tecniche di data mining.

Le seguenti tabelle riportano i valori delle altre variabili, oltre l'accuratezza, prese in considerazione per la valutazione del modello.

Tabella 6.6: Classification Report per il Decision Tree

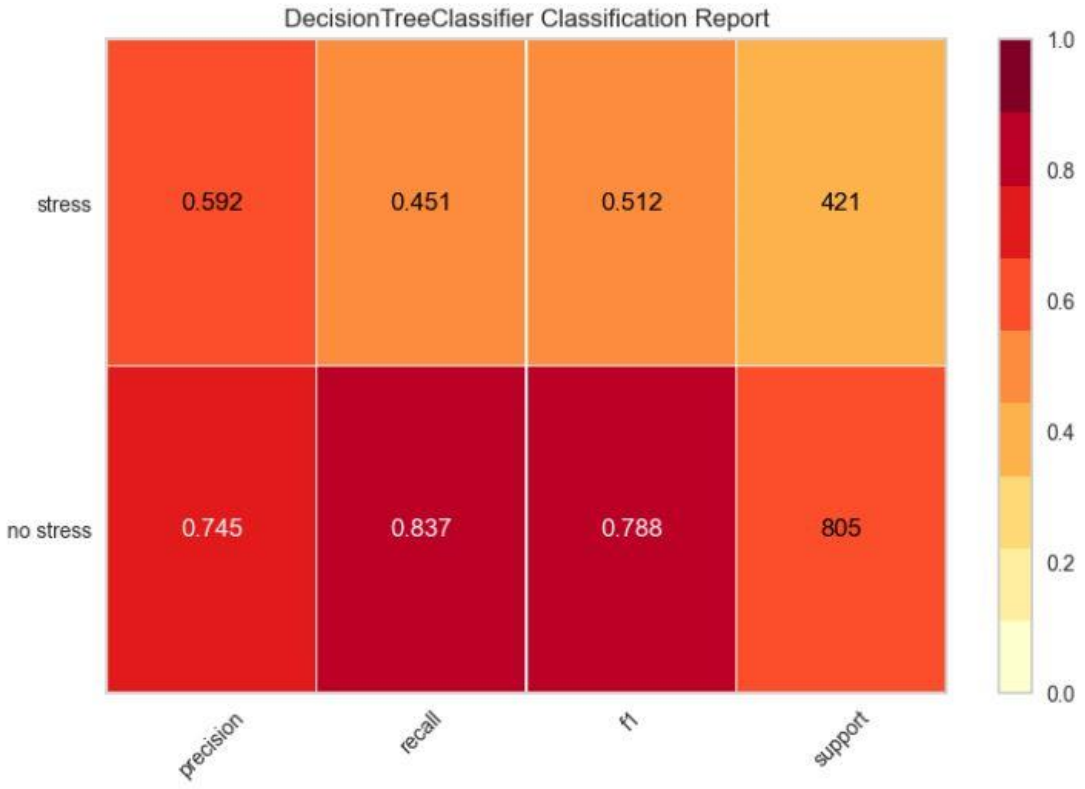


Tabella 6.7: Matrice di Confusione Per il Decision Tree

		Predicted Class	
Actual Class		**No Stress**	**Stress**
	No Stress	683	141
	Stress	197	205

L'applicazione dell'algoritmo di Random Forest allo stesso dataset è stata effettuata secondo gli iperparametri, riportati di seguito, giungendo alla scelta di quelli evidenziati in grassetto.

Tabella 6.8: Tuning degli iperparametri dell'algoritmo di Random Forest

	Parametro	Intervalli	Valori
Random Forest	criterion	Gini/entropy	**Gini**
	max_depth	{3, 4, 5, … ,10}	**10**
	min_samples_split	{4, 8, 10}	**10**
	min_samples_leaf	{2, 3, 4}	**2**
	max_features	{2, 3, 5, … , 20}	**10**
	n_estimators	{20, 50, 100}	**100**

Il modello così configurato ottiene un valore di accuracy pari al 77% e i seguenti valori di precision, recall e f1 score:

Tabella 6.9: Classification Report per il Random Forest

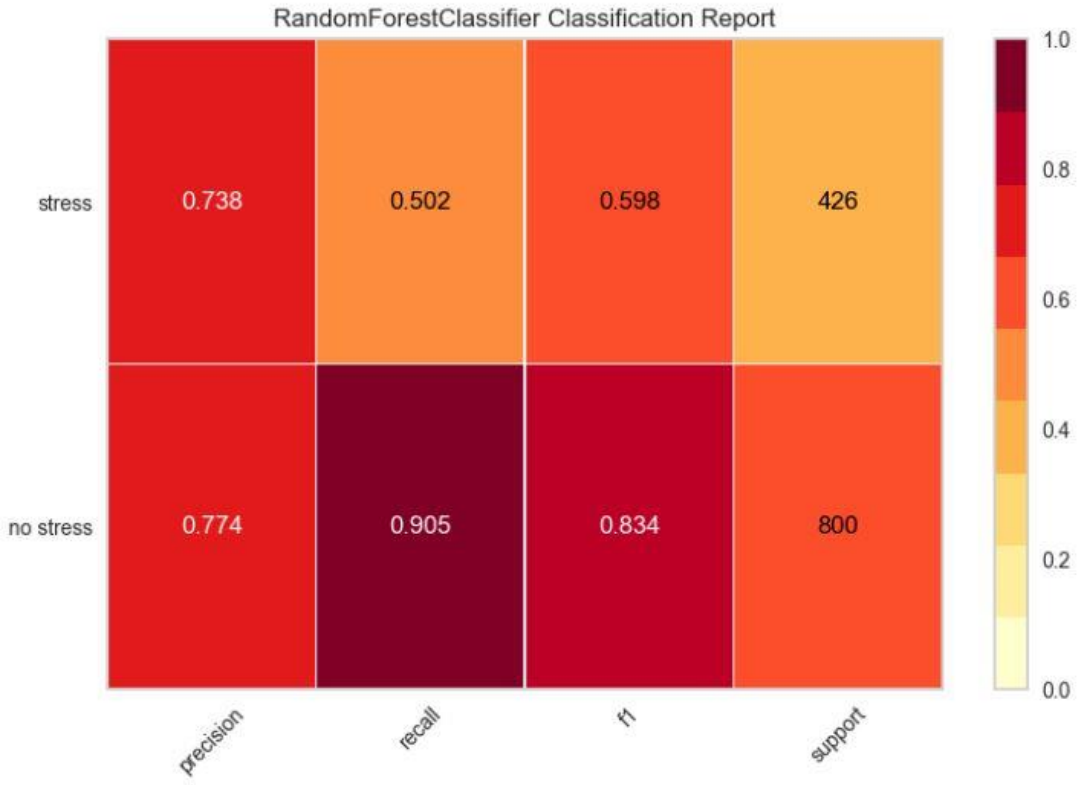


Tabella 6.10: Matrice di Confusione Per il Random Forest

		Predicted Class	
Actual Class		No Stress	Stress
	No Stress	724	76
	Stress	212	214

Per quanto riguarda l'ultimo dei modelli provati sul dataset, XGBoost, si è ottenuta un'accuratezza del 77% dopo esser stato sottoposto al tuning dei seguenti parametri caratterizzanti:

Tabella 6.11: Tuning degli iperparametri dell'algoritmo di XGBoost

	Parametri	Intervalli	Valori
XGBoost	n_estimators	{1, 3, 5, … ,50}	**40**
	max_depth	{1, 2, 3, … ,10}	**7**
	eta	{0.0001, 0.001, … ,1}	**0.1**
	subsamples	{0.1, 0.2, … ,1}	**1**
	colsample_bytre	{0.1, 0.2, … ,1}	**0.7**

L'ottimizzazione di questi parametri ha permesso di avere un'accuratezza del 77% ed i seguenti valori per la matrice di confusione e il report di classificazione:

Tabella 6.12: Classification Report per il XGBoost

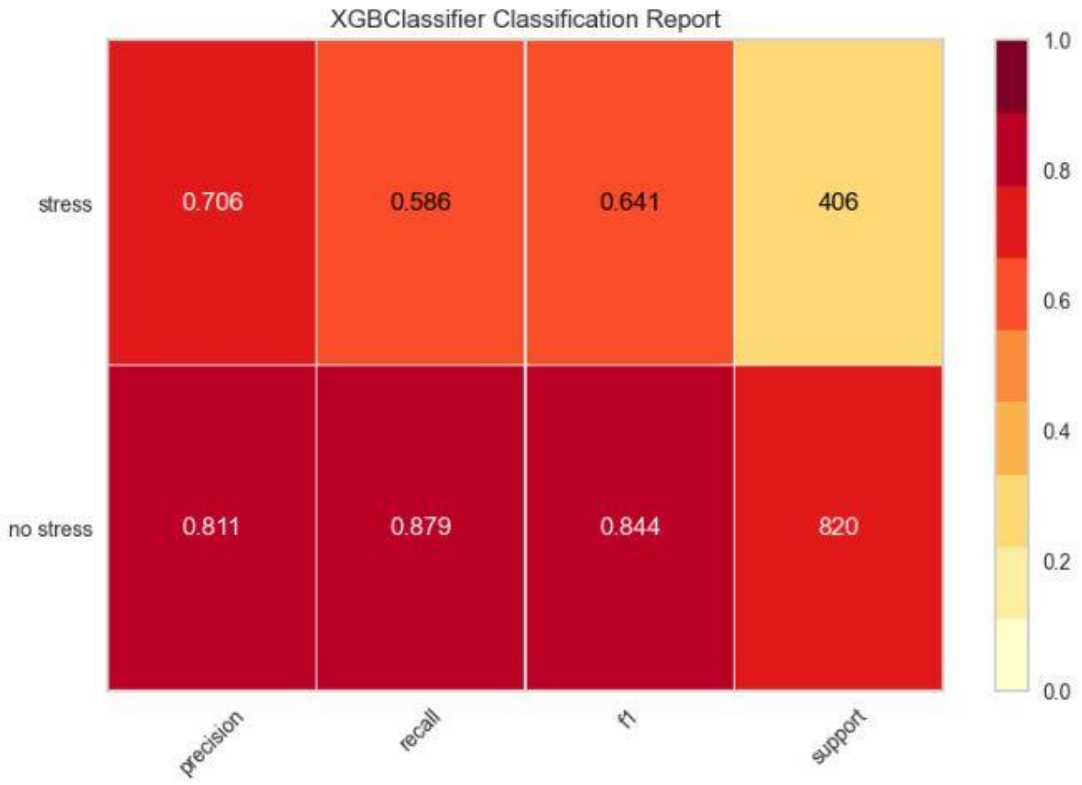


Tabella 6.13: Matrice di Confusione Per il XGBoost

		Predicted Class	
		No Stress	**Stress**
Actual Class	**No Stress**	728	91
	Stress	194	213

Oltre la valutazione dell'accuratezza, insieme ai valori di precision, recall, f1-score e la matrice di confusione è stata visualizzata e analizzata anche la curva ROC di tutti i modelli, insieme ai rispettivi valori di AUC, ossi l'area sottostante alla curva.

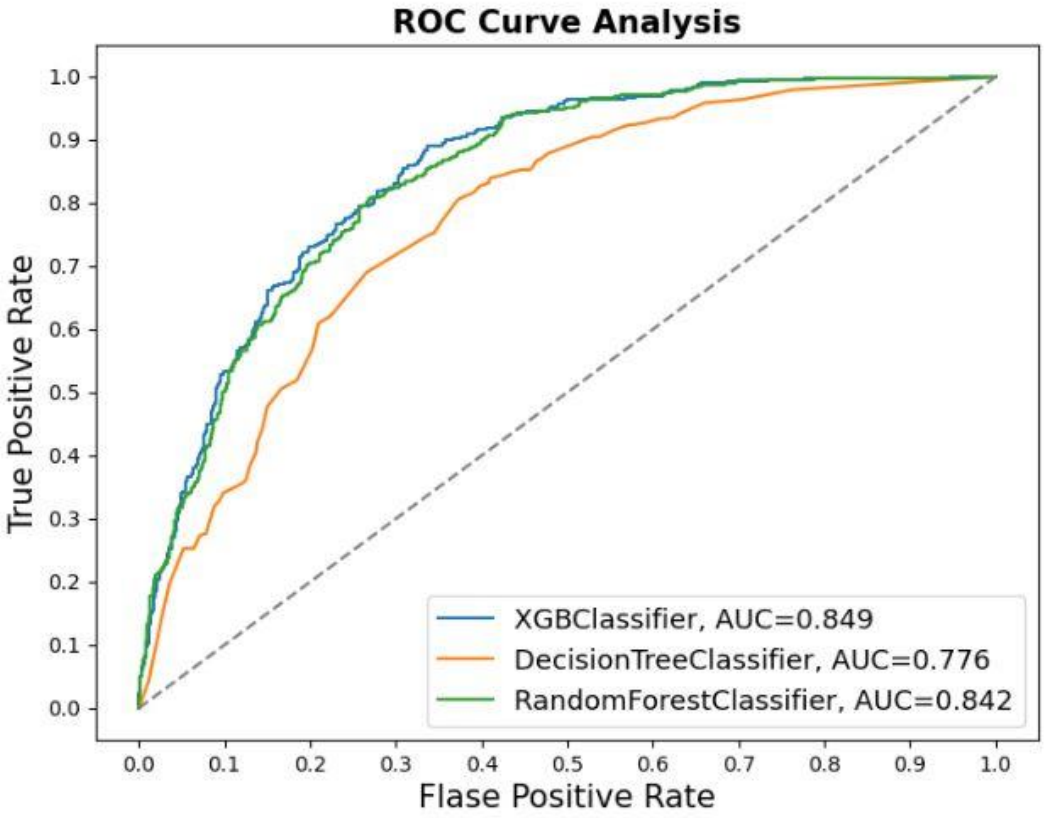


Figura 6.6: ROC Curve e AUC

Inoltre, l'ottimizzazione dei modelli e dei rispettivi iperparametri sono stati testati attraverso la K-fold Cross Validation considerando diversi valori di K. Le tabelle sottostanti mostrano i risultati con K=10 e K=100. Come è visibile, per tutti i modelli, i valori dell'accuracy sono vicini alla media calcolata con la validazione incrociata e quindi compresa nell'intervallo min-max.

Tabelle 6.14 e 6.15: Risultati K-fold Cross Validation con k=10 e k=100

		k=10	
modello	**Accuracy**	**Media**	**Std**
Decision Tree	0,73	0,7424	0,02645
Random Forest	0,77	0,778	0,021
XGBoost	0,77	0,7805	0,01969

		k=100	
modello	**accuracy**	**Media**	**Std**
Decision Tree	0,73	0,74	0,0723
Random Forest	0,77	0,7779	0,06427
XGBoost	0,78	0,77765	0,0652

Valutati i modelli sull'intero dataset, più ampiamente analizzati nel paragrafo successivo, è stata eseguita la feature selection allo scopo di allenare meglio gli algoritmi e individuare le metriche più influenti per le prestazioni dei modelli. Le figure sottostanti (*6.2, 6.3 e 6.4*) mostrano le importanze relativi di tutte le statistiche considerate, per modello utilizzato.

Al fine di ottimizzare i modelli è stata eseguita la tecnica "RFE" (Recursive Feature Elimination) fino alla riduzione del 25% della cardinalità delle metriche in input, quindi i risultati, in *Tabella 6.12*, mostrano le metriche considerate più influenti per ogni modello.

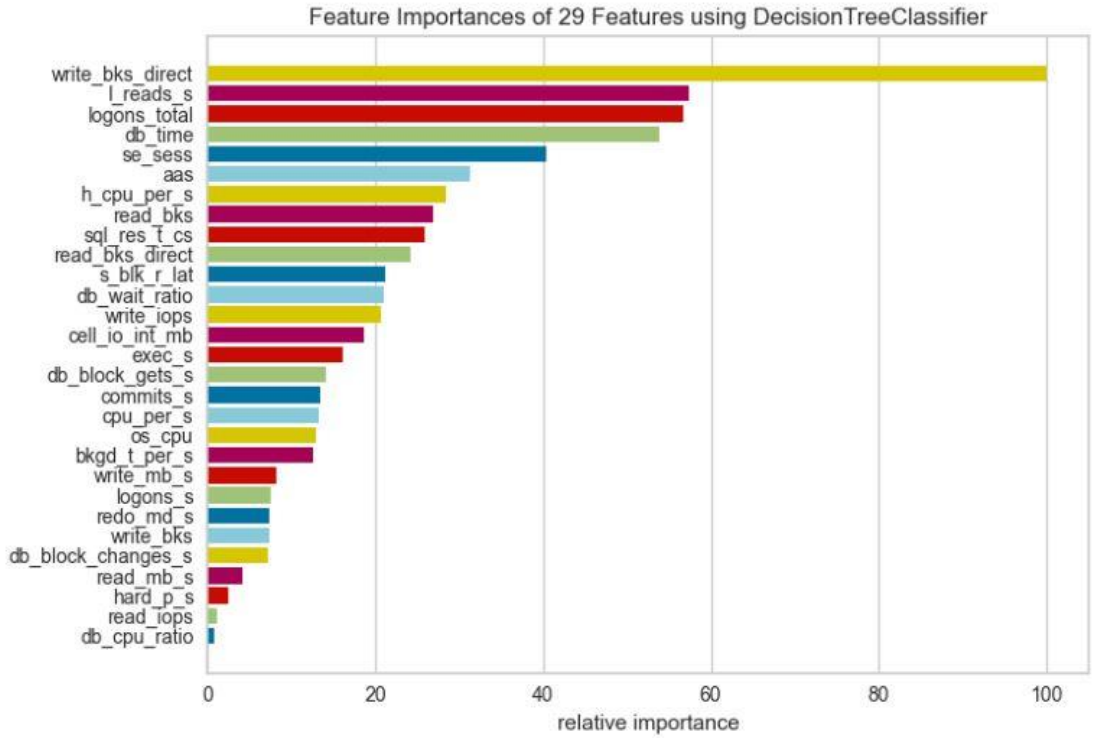


Figura 6.7: Feature Selection del modello Decision Tree

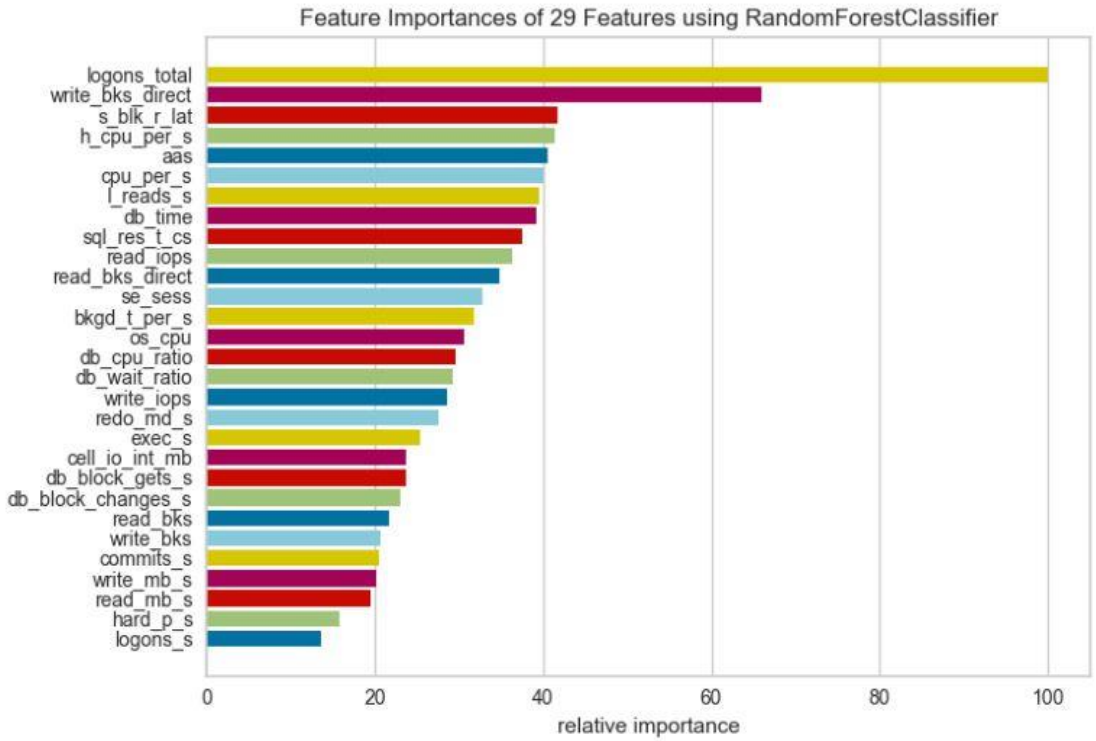


Figura 6.8: Feature Selection del modello random Forest

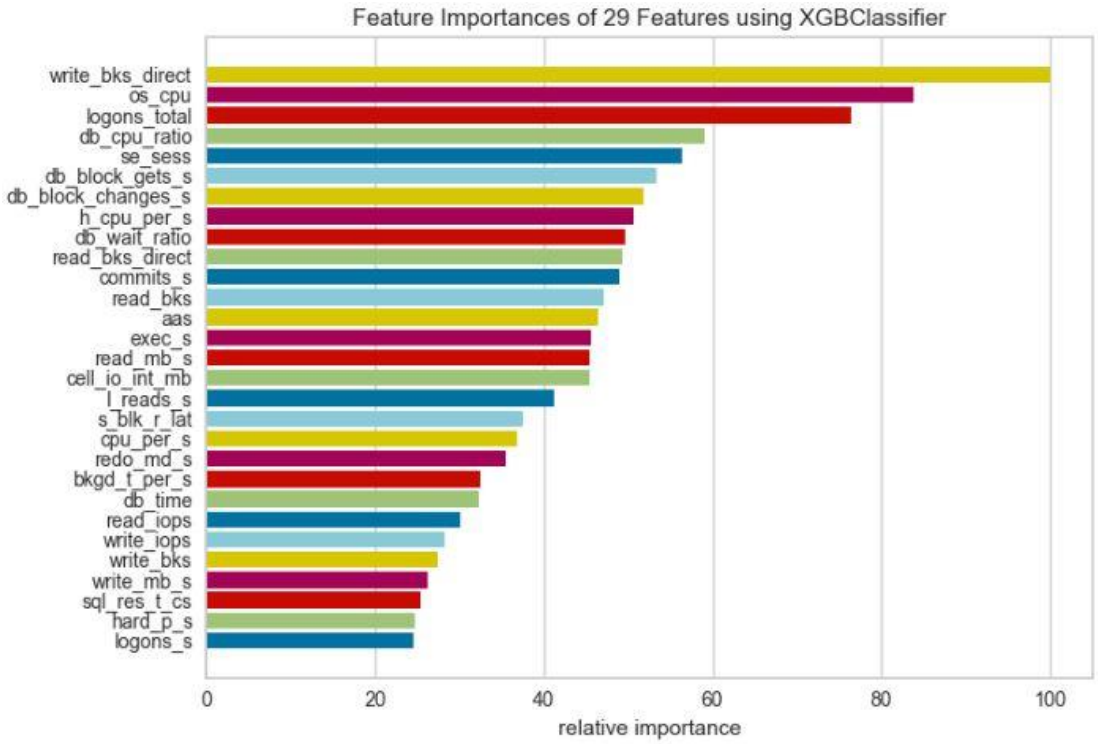


Figura 6.9: Feature Selection del modello XGBoost

Tabella 6.16: Metriche con importance più elevata per modello

Metriche più importanti		
Decision Tree	**Random Forest**	**XGBoost**
write_bks_direct	logons_total	write_bks_direct
l_reads_s	write_bks_direct	os_cpu
logons_total	s_blk_r_lat	logons_total
db_time	h_cpu_per_s	db_cpu_ratio
se_sess	aas	se_sess
aas	cpu_per_s	db_block_gets_s
h_cpu_per_s	l_reads_s	db_block_changes_s
read_bks	db_time	h_cpu_per_s
sql_res_t_cs	sql_res_t_cs	db_wait_ratio
read_bks_direct	read_iops	read_bks_direct
s_blk_r_lat	read_bks_direct	commits_s
db_wait_ratio	se_sess	read_bks
write_iops	bkgd_t_per_s	aas
cell_io_int_mb	os_cpu	exec_s
exec_s	db_cpu_ratio	read_mb_s
db_block_gets_s	db_wait_ratio	cell_io_int_mb
commits_s	write_iops	l_reads_s
cpu_per_s	redo_md_s	s_blk_r_lat
os_cpu	exec_s	cpu_per_s
bkgd_t_per_s	cell_io_int_mb	redo_md_s
write_mb_s	db_block_gets_s	bkgd_t_per_s
logons_s	db_block_changes_s	db_time
Metriche meno importanti		
redo_md_s	read_bks	read_iops
write_bks	write_bks	write_iops
db_block_changes_s	commits_s	write_bks
read_mb_s	write_mb_s	write_mb_s
hard_p_s	read_mb_s	sql_res_t_cs
read_iops	hard_p_s	hard_p_s
db_cpu_ratio	logons_s	logons_s

Infine, sono stati riapplicati i modelli al dataset costituito dalle metriche contrassegnate da un'importanza maggiore per il rispettivo algoritmo. Tale operazione ha permesso di affinare e addestrare meglio tutte le tecniche, che mostrano migliori valori di recall e accuracy. Le principali variabili utilizzate per valutare e confrontare i modelli tra loro e con la prima applicazione al dataset iniziale sono mostrate nella

Tabella 6.13. Inoltre, l'applicazione della feature selection ha permesso di individuare delle metriche poco discriminanti per tutti i modelli, quali Write_bks e Hard_s

Tabella 6.17: Tabella sintetizzante i valori delle variabili utilizzate per valutare i modelli e confrontarli con la precedente applicazione

	Accuracy		**Precision**		**Recall**	
	prima	*dopo*	*prima*	*dopo*	*prima*	*dopo*
Decision Tree	0,73	0,73	0,592	0,57	0,451	0,66
Random Forest	0,77	0,78	0,738	0,7	0,502	0,55
XGBoost	0,77	0,77	0,706	0,68	0,586	0,6

6.2 Discussione dei risultati

I risultati dettagliati nei paragrafi precedenti dimostrano che, nonostante alcuni modelli siano più performanti di altri, non esiste un modello matematico migliore in assoluto. La scelta degli algoritmi è fortemente legata alla tipologia di dataset analizzato, ma anche dalla sua cardinalità e distribuzione delle classi.

Come anche già evidenziato nella presentazione dei risultati delle applicazioni degli algoritmi di clustering, i modelli Kmeans e DBSCAN presentano una relativa coerenza nella suddivisione delle metriche in gruppi omogenei, nonostante le differenze alla base delle due tecniche. Tale congruenza è facilmente giustificabile da un'analisi approfondita sull'andamento nel tempo delle metriche con relativi trend e anomalie. Tuttavia, il KMeans presenta per entrambe le applicazioni, un coefficiente di Silhouette migliore e quindi una migliore capacità di individuare gruppi di metriche con andamenti similari. Inoltre, la preferenza verso questo algoritmo è anche dovuta all'identificazione di cluster di densità simili.

Per quanto riguarda i modelli di classificazione, invece, il confronto è stato eseguito dapprima sulla base dei valori di accuracy per poi considerare la precision e la recall della classe stress. Quest'ultima è la classe più interessante dal punto di vista aziendale, in quanto è preferibile un numero elevato di momenti non di stress classificati come momenti di stress piuttosto che un elevato numero di campioni della classe stress erroneamente considerati non di stress. Dunque, per il contesto aziendale, è preferibile avere un falso allarme che un problema reale non rilevato. La comparazione tra gli algoritmi si è servita, inoltre, della confusion matric e della curva ROC con relativi valori di AUC.

Tra i tre modelli di classificazione, il Decision Tree risulta essere quello meno performante presentando non solo un accuracy inferiore a quella degli altri modelli, ma anche i valori di precision e recall sono peggiori. Infine, la curva ROC relativa a tale

modello, è quella più spostata verso il basso, e di conseguenza la rispettiva area sotto la curva ha il valore minore.

Risultati migliori e molto similari tra loro si ottengono con i modelli di Random Forest e XGBoost. La superiorità di tali tecniche è da ricercarsi nel caso del Random Forest nella possibilità di associare un peso ad ogni classe analizzata rendendo così il modello più propenso a classificare meglio una classe anziché un'altra. Diversa è la situazione per XGBoost dove la presenza di performace migliori è da attribuire alla generazione di un modello robusto a partire da modelli più deboli ognuno dei quali contribuisce alla soluzione finale con un certo peso, compresi gli errori.

Con entrambi gli algoritmi si ottiene un'accuratezza pari a 0,77, e anche i rispettivi valori di AUC sono molto simili tra loro, distanziandosi di appena 0,006. Determinante nella scelta del modello migliore per il dataset preso in esame è, quindi, il report di classificazione, contenente precision e recall. Per i motivi sottolineati in precedenza la classe Stress è più interessante e importante per il contesto aziendale, pertanto si preferiscono valori migliori di precisione recall di tale classe. Se l'algoritmo XGBoost presenta un valore di recall più elevato, pari a 0,586 contro lo 0,506, il Random Forest supera di circa 3 punti percentuali la precision del precedente modello (73,8% contro il 70,6%).

Come approfondito *nel paragrafo 3.2 Metodi di valutazione*, il punteggio di precisione del modello rappresenta la sua capacità di prevedere correttamente gli aspetti positivi di un evento sul totale di tutte le previsioni positive effettuate. Il punteggio di recall, invece, è una misura di sensibilità del modello, che rappresenta l'abilità di questo di prevedere correttamente i positivi rispetto ai positivi effettivi. Pertanto, data la necessità di distinguere i momenti di stress e che questi non siano erroneamente classificati come Non Stress, si predilige un valore maggiore di recall. Ne consegue che tra i tre algoritmi di classificazione considerati nel presenta caso studio, il migliore per il dataset scelto è XGBoost.

7 Conclusioni

Questo capitolo conclusivo vuole essere una valutazione critica dei risultati ottenuti, ponendo una certa attenzione sulle ragioni che hanno prodotto il presente lavoro di tesi. In secondo luogo, viene dato un accenno a quelli che potrebbero essere i futuri sviluppi di questa ricerca.

7.1 Considerazioni di carattere generale

L'innovazione tecnologica e l'utilizzo sempre più massivo del machine learning hanno indotto l'azienda a valutare l'adozione di tecniche di data mining nell'ambito del monitoraggio dei sistemi complessi e ingegnerizzati. Lo sviluppo di questa ricerca nasce dalla necessità pratiche dell'ambiente lavorativo, in cui il monitoraggio dei sistemi informatici è essenziale per il corretto funzionamento di tutte le infrastrutture tecnologiche.

L'idea di partenza di questa ricerca è l'integrazione ai tradizionali sistemi di monitoraggio di uno strumento di supporto che semplifichi tale attività. In questo senso il lavoro svolto è nato con l'obiettivo di proseguire l'indagine, già iniziata in azienda attraverso altre tesi precedenti, sulle potenzialità e sulle applicazioni del data mining ai sistemi di monitoraggio dell'infrastruttura.

Il lavoro si è orientato verso la ricerca di un modello da preferire nell'ambito della classificazione e del clustering. Secondo gli esperimenti condotti e la generazione del dataset effettuati durante il lavoro di ricerca, le tecniche di data mining migliori risultano essere il Kmeans e XGBoost.

Attraverso la ricerca svolto si è dimostrato che è possibile individuare tra le metriche set di queste con lo stesso andamento, rendendo così più semplice e meno dispendiosa a livello di tempo, l'analisi dell'andamento delle metriche alla ricerca di anomalie allo scopo di individuare la causa che ha determinato il verificarsi di un problema. Altro elemento che la ricerca ha messo in risalto è l'opportunità di

distinguere momenti di stress dai momenti di normale funzionamento. L'identificazione della presenza di un'anomalia nel regolare funzionamento di un database non è un'operazione semplice, in quanto la definizione stessa di anomalia non è univoca e condivisa all'interno dello stesso sistema. Il lavoro di ricerca svolto si è basato sull'assunzione che le anomalie sono quei momenti in cui il database mostra avere caratteristiche molto diverse dal resto dei dati e, quindi, dal normale funzionamento.

I risultati incoraggianti di questa ricerca hanno messo in luce la possibilità di applicare i modelli sui sistemi di monitoraggio utilizzati in azienda fino all'implementazione futura di uno strumento di monitoraggio, in grado di rilevare gruppi di metriche con lo stesso andamento da poter analizzare in caso di malfunzionamenti, ma anche di distinguere momenti di stress da quelli non di stress fino alla previsione di questi ultimi. Inoltre, lo svolgimento della ricerca ha anche permesso di individuare alcuni limiti, quali l'utilizzo di un ambiente controllato se pur simulativo di un contesto reale e soprattutto la necessaria creazione di un generatore di stress completo per il database.

7.2 Sviluppi futuri

Come più volte evidenziato, l'analisi svolta ha tenuto conto di un sottoinsieme di metriche rappresentative dello stato del database. Al fine di avere una descrizione più accurata del sistema ed un'analisi e previsione più precisa, questi dati potrebbero essere amplianti e integrati con altri tipi di informazioni inerenti al sistema analizzato. Il primo sviluppo futuro potrebbe, quindi, essere la realizzazione di uno strumento che integra il presente lavoro di tesi con altre ricerche condotte o ancora in corso nella stessa azienda allo scopo di creare un unico strumento di supporto facilmente utilizzabile. Con l'arricchimento del dataset con un numero sempre maggiore di metriche con altri tipi di informazioni, fino anche all'utilizzo dell'insieme completo di tutte le metriche dell'Oracle Enterprise Manager, si potrebbe pensare alla riproduzione su larga scala dei modelli studiati supportati e combinati con altri algoritmi di classificazione, clustering e previsione. Sia le metriche che gli algoritmi scelti sono un sottoinsieme di quelle disponibili, perciò con l'ampliamento del dataset di metriche si potrebbe pensare anche ad analisi differenti da quelle portate in questo progetto, un esempio su tutti potrebbe essere l'utilizzo di modelli predittivi.

Infine, si potrebbe realizzare un tool in grado di prevedere e segnalare la presenza di anomalie all'interno del sistema, in cui la presente ricerca è un punto di partenza. Inoltre, essendo il presente lavoro di tesi un progetto pilota, utilizza un ambiente controllato a cui applica degli stress test, una prima implementazione futura, forse la più immediata, è sicuramente l'applicazione di quanto studiato e analizzato ad un contesto reale aziendale.

Come si può facilmente intuire, gli aspetti da indagare sono innumerevoli e la possibilità di applicare nuove metodologie di analisi costituisce uno degli aspetti interessanti della soluzione proposta, insieme, chiaramente, già a questo già affrontato con questa tesi.

Bibliografia

Arvai K., *K-Means Clustering in Python: A pratical Guide*, Luglio 2020 [Online] https://realpython.com/k-means-clustering-python/#choosing-the-appropriate-number-of-clusters

Brownlee J., *Extreme Gradient Boosting (XGBoost) Ensemple in Python*, Aprile 2021 [Online] https://machinelearningmastery.com/extreme-gradient-boosting-ensemble-in-python/

Galarnyk M., *PCA* Dicembre 2017 [Online] https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60

Govoni L., *Cos'è l'algoritmo XGboost?*, -- [Online] https://www.lorenzogovoni.com/cose-lalgoritmo-xgboost/

Giles Dominic. *Swingbench Reference and User Guide* [Online] http://www.dominicgiles.com/swingbench/swingbench22.pdf

Govoni L., *L'importanza del ridimensionamento dei dati nei problemi di machine learning*, -- [Online] https://www.lorenzogovoni.com/ridimensionamento-dei-dati/

Han J., Kamber M., Pei J., *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers Inc., 2005

Kumar A. *Accuracy, Precision, Recall & F1-Score,* Agosto 2020 [Online] https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/

Mithrakumar M*., How to tune a Decision Tree*, Novembre 2019 [Online] https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680

Mysiak K*., Exploring DBSCAN Clustering*, Luglio 2020 [Online] https://towardsdatascience.com/explaining-dbscan-clustering-18eaf5c83b31

Navlani A., *Understanding Random Forest Classifiers in Python* ,Maggio 2018 [Online] https://www.datacamp.com/community/tutorials/random-forests-classifier-python

Nelson D., *Overview of Classification Methods in Python with Scikit_Learn*, Maggio 2019 [Online] https://stackabuse.com/overview-of-classification-methods-in-python-with-scikit-learn

Oracle. *Evaluating and Comparing Oracle database Appliance* Performance Oracle, Gennaio 2020 [Online] https://www.oracle.com/technetwork/database/database-appliance/learnmore/oda-x82ha-perf-wp-5972834.pdf

Oracle. *Introduction to Sample Schemas* --. [Online] https://docs.oracle.com/en/database/oracle/oracle-database/21/comsc/introduction-to-sample-schemas.html#GUID-844E92D8-A4C8-4522-8AF5-761D4BE99200

Ozdemir Sinan. *Data Science*. *Guida ai principi e alle tecniche base della scienza dei dati* Apogeo, 2017

Robinson S., *Decsion Trees in Python with Scikit_learn*, Febbraio 2018 [Online] https://stackabuse.com/decision-trees-in-python-with-scikit-learn

Scikit-learn. Machile Learning in Python [Online] https://scikit-learn.org/stable/index.html

Swingbench [Online] https://dominic.giles.com (dominicgiles.com)/

XGBoost Documentation [Online] https://xgboost.readthedocs.io/en/latest/

Yellowbrick: Machine Learning Visualization [Online] https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/

Yiu T., *Understanding Random Forest*, Giugno 2019 [Online] https://towardsdatascience.com/understanding-random-forest-58381e0602d2

—. *Feature Importance Measures for Tree Models* — Part I. Medium.com. [Online] https://medium.com/the-artificial-impostor/feature-importance-measures-for-tree-models-part-i-47f187c1a2c3.

--. *Intelligenza artificiale* – [Online] http://www.intelligenzaartificiale.it/machine-learning/

Riferimenti

[1] --. Tipi di algoritmi per il machine learning --. datawiring.me [Online] https://www.datawiring.me/tipi-di-algoritmi-per-il-machine-learning/

[2] Schlitter N., Falkowski T., Laessing J.. DenGraph-HO: Density-based Hierarchical Community Detection for Explorative Visual Network Analysis Dicembre 2011 [Online] https://www.researchgate.net/figure/The-concepts-directly-density-reachability-density-reachability-and-density_fig1_259461372

[3] Koehrsen W. Random Forest Simple Explanation Dicembre 2017 [Online] https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d

[4] Zhang Z., Mayer G., Dauvilliers Y. *Exploring the clinical features of narcolepsy type 1 versus narcolepsy type 2 from European Narcolepsy Network database with machine learning* Luglio 2018 [Online] https://www.researchgate.net/figure/A-simple-example-of-visualizing-gradient-boosting_fig5_326379229

[5] Govoni L. *Matrice di confusione: cos'è e come funziona?* --. [Online] https://www.lorenzogovoni.com/matrice-di-confusione/

[6] Raileanu L.E. e Stoffel K., *Confronto teorico tra l'indice di Gini ei criteri di guadagno dell'informazione*. Annali di matematica e intelligenza artificiale 41: 77-93, 2004

[7] --, Cross-validation: evaluating estimator performance [Online] https://scikit-learn.org/stable/modules/cross_validation.html

[8] Oracle Corporation. Database Sample Schemas --. [Online] https://docs.oracle.com/en/database/oracle/oracle-database/19/comsc/schema-diagrams.html#GUID-01BFEF14-C6BD-42CD-8558-DDD214DB3AE0

[9] Bhande A. *What is underfitting and overfitting in machine learning and how to deal with it.* Marzo 2018 [Online] https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

Ringraziamenti

A conclusione di questo ciclo di studi ci tengo a ringraziare tutti coloro che hanno partecipato, ognuno a proprio modo, al mio percorso di crescita.

Il primo Grazie va ai miei genitori, a mia madre e mio padre, per avermi lasciato la libertà di intraprendere questo lungo percorso di studi, e per avermi supportato sempre. La loro presenza costante, nei momenti di gioia e in quelli difficili, è inestimabile. Descrivere il loro ruolo, insieme a quello di mio fratello, sarebbe riduttivo in poche righe. Sarò loro grata per sempre per tutto ciò che mi hanno insegnato e fatto per me.

Un Grazie va alla Professoressa Cerquitelli, relatrice della mia tesi, e a tutti i componenti dell'azienda Mediamente Consulting, per la loro rara disponibilità e simpatia, per aver avermi accolta e fatta sentire a mio agio sin dal primo giorno. Un grazie speciale va, però, a coloro che mi hanno seguito durante la realizzazione di questo lavoro per la pazienza e il supporto costante.

Grazie alle mie compagne di banco con le quali è stato un piacere condividere le ore di lezione e non solo, insieme ci siamo fatte forza per superare ogni ostacolo e condiviso le gioie reciproche: a Zaira, alla sua determinazione e costanza che mi è stata sempre di ispirazione; a Silvia, alla sua tenacia, fedelissima compagna anche di concertini, mercatini e aule studio; a Serena, e la sua capacità di centrare l'obiettivo nonostante l'apparente disordine intorno a lei.

Grazie anche agli amici lontani ed in particolare a Serena, che nonostante i km di distanza, mi è stata sempre vicina, supportandomi e credendo sempre in me, spesso più di quanto facessi io stessa.

Grazie ai miei fantastici coinquilini inquieti, Luigi e Giorgio, con cui ho condiviso gioie e dolori di questi ultimi anni, per le risate, i discorsi sconnessi e quelli più seri, sempre accompagnati da litri di tisane e di barbera.

Un Grazie speciale va a Simone, Ilenia e Tommaso e alle nostre "seratine tranquille" condite di discorsi infiniti, che sono stati spesso tappe fondamentali durante questo viaggio. Ma un Grazie speciale va al mio fedelissimo Sleepover Club, che resta sempre un punto di riferimento, con cui è sempre un piacere scambiare un libro, una cena, una birra, un'opinione, un pensiero o un interrogativo.

Un altro Grazie lo voglio dire ai miei compagni di viaggi estivi, con i quali è sempre bello affrontare avventure pazze in giro per l'Italia con una naturalezza disarmante sin dal primo giorno.

Infine, un Grazie va alla mia famiglia, zii e cugini, anche quelli inaspettati, grandi e piccini, che hanno sempre creduto e tifato per me, in ogni istante, e perché so di poter sempre contare su di loro.

Grazie a tutti.

Ilaria