POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Machine Learning per l'automazione di processi di Data Quality

Relatore
Prof. Paolo Garza

Candidato
Vito Valente

Supervisore Aziendale
Marco Gatta
Data Reply Srl

2019-2020

Sommario

L'aumento esponenziale delle fonti di dati a disposizione delle organizzazioni, ha spinto la crescita di modelli decisionali con approccio data-driven. La consapevolezza del fatto che la bontà dei risultati ottenuti è direttamente proporzionale alla qualità dei dati analizzati, ha come conseguenza un incremento dell'attenzione da parte delle aziende nei processi di data quality. La difficoltà nel definire in maniera oggettiva e generica il concetto di qualità, che risulta variabile in base all'ambito in cui ci si trova, rende gli algoritmi tradizionali di analisi della qualità delle informazioni dispendiosi e poco accurati. Lo scopo di questo lavoro è dunque quello di migliorare tale approccio sfruttando meccanismi basati sul Machine Learning al fine di eliminare le soglie fisse dettate dagli esperti di settore ed inserite nelle regole di business che dominano gli algoritmi tradizionali. Verranno presentate di seguito due strategie atte al miglioramento della fase di data quality all'interno di una pipeline di raccolta ed elaborazione di informazioni provenienti da veicoli. La prima strategia prevede l'uso dell'algoritmo di Isolation Forest e si pone come obiettivo la ricerca efficace ed automatica di outliers all'interno dei dati caratterizzati da un elevato numero di dimensioni. Tale problematica risulta particolarmente complicata dal momento che le dimensioni meno rilevanti tendono ad aumentare il livello di sparsità dei dati e nascondere così gli elementi sospetti. Il secondo approccio è mirato invece alla ricerca locale di anomalie all'interno di un sottospazio definito e limitato dei dati, facendo uso del metodo di Local Outlier Factor. In questo caso ci si allontanerà dall'esplorazione globale al fine di analizzare dimensioni caratterizzate da legami particolarmente importanti. Entrambe le soluzioni permetteranno di eliminare i lunghi tempi necessari all'identificazione dei parametri di accettabilità dei dati da parte degli esperti di dominio e i possibili errori tipici dell'operazione. Le performance dei metodi implementati verranno misurate e comparate in termini di precisione e richiamo nell'ultima parte dell'elaborato, dove verranno inoltre evidenziate le principali limitazioni e i possibili sviluppi futuri.

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Importanza dei dati e della loro correttezza

Nel corso dell'ultimo decennio si è sviluppata all'interno della società la consapevolezza di poter progredire sotto molteplici aspetti sfruttando l'informazione digitale che deriva dalla raccolta e dall'analisi dei dati. Lo sviluppo di tecnologie che rientrano nell'*Internet of Things (IoT)* e di piattaforme, principalmente web, che valorizzano sempre di più la raccolta di dati degli utenti, ha portato ad un aumento esponenziale degli strumenti a disposizione delle organizzazioni per ottimizzare profitto, struttura e metodologie di lavoro.
L'origine di tali informazioni non è univoca: si parla infatti di dati generati direttamente dall'uomo quando si fa riferimento a ciò che deriva ad esempio dai social media, dai blog e da e-commerce (generalmente attraverso i famosi cookies) e di dati generati dalle macchine, in maniera automatica, se si pensa e sensori o dispositivi GPS.
L'insieme di tutti questi dati va a comporre il concetto di *Big Data* definito come: "una raccolta di dati informativi così estesa in termini di volume, velocità e varietà da richiedere tecnologie e metodi analitici specifici per l'estrazione di valore o conoscenza" [1]. Come suggerito dalla definizione, affinché si possa trarre reale vantaggio dalle informazioni raccolte, è necessario applicare tecniche di elaborazione delle stesse che portino ad un risultato utilizzabile. Tra le varie possibilità, particolare rilievo hanno il Data Mining, l'intelligenza artificiale e il Machine Learning. Data la grande varietà di applicazioni che traggono beneficio da questo tipo di processi (profilazione dei clienti, sistemi di monitoraggio, previsione e sicurezza...), il numero di aziende che fa uso di queste tecnologie all'interno del proprio processo decisionale è in continuo aumento.
Questa opportunità può essere però limitata dalla difficoltà di gestione e integrazione di molteplici fonti di dati e, soprattutto, da una scarsa qualità dei dati stessi. Da un basso livello qualitativo possono derivare decisioni errate e perdite dal punto di vista economico per l'organizzazione che decide di sfruttare le informazioni

raccolte come parte principale del proprio business. Risultano quindi necessarie e fondamentali tecniche di misurazione della qualità del dato per distinguere ciò che è utile per l'analisi da ciò che invece risulterebbe controproducente.
In questo contesto si sviluppa la nozione di *Data Quality*, ovvero l'insieme delle strategie adottate per valutare e incrementare la qualità dei dati. La prima problematica che sorge è però proprio quella di definire con certezza quando una informazione possa essere considerata di alta qualità o accettabile piuttosto che categorizzata come non valida o compromessa. Il valore del dato risulta infatti essere strettamente legato all'ambito in cui viene utilizzato ma la definizione di metriche oggettive, condivise e replicabili rappresenta un passaggio indubbiamente indispensabile per ogni organizzazione. Per far fronte a questa necessità sono nate e si sono evolute nel tempo diverse tecniche di Data Quality, dalla semplice analisi dell'esperto di dominio a metodologie sempre più automatiche e intelligenti.

1.2 Scopo del lavoro

Uno degli approcci più diffusi al problema del Data Quality risulta essere quello di creare una serie di regole di business da parte di esperti del settore, i quali stipulano un insieme di soglie che determinano se l'informazione analizzata risulta più o meno valida. Questa soluzione, pur risultando efficiente ed efficace in alcuni contesti, presenta diverse limitazioni.
In primo luogo si osserva che creare una o più regole per ogni tipo di dato analizzato può diventare ingestibile a fronte della crescita esponenziale dei dati stessi. L'uso di soglie fisse è inoltre inappropriato per tutti gli ambienti dinamici che vedono cambiare significativamente le informazioni ricevute nel tempo. Aspetto ancor più importante è legato alle correlazioni esistenti tra i dati che potrebbero influenzarne l'effettiva validità: se le dimensioni di interesse all'interno della stessa informazione sono molteplici, è possibile che osservando singolarmente ogni dimensione non si vada a violare nessuna soglia ma con una visione globale vengano a galla diverse perplessità sulla validità di ciò che si è analizzato.
Si pensi come esempio alle informazioni ricevute da un veicolo durante la marcia. Una velocità media elevata (come può essere un valore di 120 km/h) rientra senza dubbio tra i valori possibili. Tale dato può essere considerato valido se si tratta di una normale vettura in un tratto extraurbano ma diventa meno plausibile se, considerando anche il peso del veicolo, questo risulti superiore alla decina di tonnellate. Inserire vincoli statici che tengano conto di tutte le possibilità è sicuramente un compito non banale, dove la probabilità di tralasciare scenari è elevata.
Lo scopo di questo lavoro è dunque quello di sfruttare meccanismi di Machine Learning per riuscire ad eliminare qualsiasi tipo di soglia definita a priori e creare un sistema che, in maniera autonoma, definisca se un dato può essere considerato valido o meno. Nei prossimi capitoli verranno presentate, applicate e confrontate

due diverse strategie: la prima completamente automatizzata, in grado di osservare tutte le proprietà che compongono l'informazione e prendere una decisione di conseguenza, la seconda in cui si analizzano specifiche proprietà, decise dall'utente sulla base di un legame particolarmente interessante.

1.3 Struttura dell'elaborato

Lo scopo di questo paragrafo è quello di fornire una visione generale sulla struttura dell'elaborato e sugli argomenti approfonditi di seguito.
Il capitolo 2 presenta una definizione di 'Data Quality', ne specifica i punti salienti e le relative dimensioni di analisi. Viene inoltre fornita una visione dei possibili meccanismi diffusi attualmente che costituiscono lo stato dell'arte.
Il capitolo 3 analizza il concetto di 'Outlier' e le relazioni presenti tra l'operazione di identificazione delle anomalie e la qualità dei dati. Sono inoltre elencate le possibili tipologie di algoritmi esistenti per questo tipo di compito, evidenziando i vari punti di forza e di debolezza.
Nel capitolo 4 sono esposti i dettagli delle problematiche affrontate, l'insieme dei processi che costituiscono la pipeline di ingestion e trasformazione dei dati e le modifiche apportate. In questa sezione sono elencate in maniera puntuale le caratteristiche dei metodi effettivamente usati, insieme alle motivazioni che hanno portato alla scelta degli stessi. Nell'ultima parte sono poi riportati i particolari relativi all'ambiente di lavoro con i vantaggi che ne derivano.
Il capitolo 5 presenta i risultati ottenuti in termini di precisione e richiamo, un confronto tra le due tecniche e le loro principali limitazioni.
Nella parte conclusiva vengono infine esposti i miglioramenti apportati rispetto alla soluzione originale ed introdotte delle possibili tecniche per ulteriori incrementi di efficienza ed efficacia per la soluzione proposta.

Capitolo 2

Data Quality

2.1 Dimensioni del Data Quality

Con il termine 'Data Quality' si intende l'analisi dello stato del dato, che è strettamente connesso alla sua abilità di risolvere più o meno efficacemente il compito per cui viene raccolto. Questo stato può essere 'buono' o 'cattivo' in base agli standard definiti dall'organizzazione che utilizzerà i dati stessi, tenendo conto dei vari parametri (o dimensioni) che definiscono la qualità del dato, un concetto multidimensionale la cui valutazione implica la definizione di metriche soggettive, adattabili ad un particolare contesto di business. È comunque possibile tentare di definire delle metriche generalizzate che sono condivise dalla maggior parte dei domini applicativi. Ottenere un livello di bontà dell'informazione pari al 100% sotto ogni punto di vista è un'impresa ardua, sia in relazione ai costi che alle tempistiche necessarie, per questo ogni azienda tende a concentrarsi sugli aspetti che vengono indicati come essenziali nel proprio business. In alcuni casi la perfezione, oltre che costosa, potrebbe risultare non necessaria: operazione comune è quella di definire delle soglie di accettabilità per ogni proprietà di interesse per poi applicare le tecniche individuate per il raggiungimento dell'obiettivo.
Le dimensioni del Data Quality possono essere definite a livello di schema, di processo o di dato. Analizzare lo schema implica la verifica della struttura logica necessaria per la rappresentazione dell'informazione. Utilizzare ad esempio un unico campo di testo per permettere ad un utente di inserire il proprio indirizzo contenente la via, il numero civico, la città e il codice di avviamento postale è una soluzione indubbiamente meno efficace rispetto alla preparazione di campi separati, uno per ogni componente di interesse, al fine di evitare errori e incongruenze. A livello di processo si verifica le modalità attraverso le quali i dati vengono raccolti, mentre a livello di dato si analizza direttamente il valore raccolto. Bisogna evidenziare che le scelte apportate a livello più alto (schema, processo) influiscono sulla qualità anche a livello più basso (dato).
Nella figura 2.1 sono riportate le principali dimensioni a livello di dato identificate

come le grandezze più importanti da considerare quando si approccia una analisi di qualità.

Figura 2.1. Dimensioni del Data Quality - bluwaveanalytics.com

2.1.1 Accuratezza

L'accuratezza del dato è definita come la distanza tra un valore v e un valore v' considerato come la corretta rappresentazione del fenomeno reale che v intende esprimere [2]. Descrive il grado di correttezza tra ciò che è stato raccolto e quello che è presente nel 'mondo reale'. Problemi di accuratezza si verificano nella primissima fase, quella di raccolta e misurazione del dato. Si pensi ad esempio all'errore umano durante la compilazione di un form oppure al guasto di un sensore di temperatura che inizia ad inviare dati errati alla centrale. Se si conosce il contesto in cui le informazioni vengono raccolte, gli errori possono essere riconosciuti semplicemente osservando i dati, operazione proibitiva nella società attuale con la mole di informazioni che le organizzazioni ricevono.
L'accuratezza può essere valutata sotto due aspetti:

- Accuratezza semantica: basata su un confronto tra il dato osservato e il valore reale, che in questo caso deve essere noto. Non si preoccupa di approfondire quanto l'informazione ricevuta può essere vicina al valore vero ma fornisce una valutazione dicotomica: o il dato è accurato quanto il valore reale oppure non

lo è. Questa dimensione si dimostra quindi fondamentale nella definizione del concetto di correttezza del dato.

- Accuratezza sintattica: un singolo dato è considerato "sintatticamente accurato" quando esso coincide con il valore di una fonte identificata di informazioni convalidate. Si verifica la presenza di ciò che viene misurato all'interno del dominio D che contiene tutte le possibilità accettabili. Se tale elenco è considerato esaustivo, il dato viene riconosciuto come accurato se il suo valore fa parte dell'elenco e non accurato in caso contrario. Se invece si è consapevoli del fatto che il dominio conosciuto è incompleto si può affermare che il valore è valido nel caso in cui si avesse riscontro positivo circa la sua presenza nell'insieme noto ma non si può valutare con certezza la qualità del dato in caso risulti assente.
  Una delle possibili correzioni consiste nello sfruttare dati analoghi ottenuti in tempi passati oppure degli standard esistenti (se presenti) per trovare l'elemento del dominio che più si avvicina al dato raccolto e che può andare a sostituire lo stesso in fase di analisi.

2.1.2 Consistenza

Quando un dato viene memorizzato in punti diversi, tutte le repliche devono essere consistenti tra loro. Se ad esempio gli stessi dati per un utente sono stati acquisiti da fonti diverse, questi non devono presentare incongruenze: le regole semantiche definite su un insieme di dati devono essere rispettate. Facendo riferimento ad una semplice base dati di tipo relazionale possono essere identificati diversi livelli di consistenza:

- Consistenza di chiave: richiede che all'interno di una tabella non siano presenti tuple diverse con la stessa chiave primaria. Si consideri un database che contiene le informazioni anagrafiche degli studenti iscritti ad una certa università. Utilizzare come chiave primaria della tabella l'insieme degli attributi *nome, cognome, anno di nascita* risulta inadeguato in quanto il vincolo di chiave primaria richiederebbe l'assenza di due studenti con stesso nome e cognome, nati nel medesimo anno. La situazione descritta pur non essendo particolarmente comune, è certamente possibile ed il verificarsi di questa condizione metterebbe in risalto l'inadeguatezza della chiave. Una scelta sicuramente migliore sarebbe quella di utilizzare la matricola dello studente come identificativo.

- Consistenza di inclusione: richiede che un insieme di colonne di uno schema sia contenuto in un ulteriore insieme di colonne, appartenente allo stesso schema o ad un'istanza diversa. Tale vincolo è noto anche come 'integrità referenziale'. Ritornando all'emepio precedente, se oltre alla tabella anagrafica ne fosse presente una contenente le informazioni sui corsi ai quali gli studenti risultano

iscritti, non potrebbero esserci all'interno di quest'ultima delle righe contenenti matricole (chiave esterna) che non compaiono nella tabella di anagrafica (dove la matricola è chiave primaria).

- Dipendenze funzionali: permette di conoscere con certezza il valore di un attributo data la tupla che ne fa riferimento. Se si pensa al codice fiscale, questo risulta funzionalmente dipendente dalle informazioni che lo compongono (nome, cognome, data e luogo di nascita, sesso).

2.1.3 Univocità

Ogni oggetto o evento del mondo reale dovrebbe essere rappresentato all'interno del sistema una e una sola volta [3]. Avere registrati all'interno del sistema due utenti come 'Ale Rossi' e 'Alessandro Rossi' che condividono la maggior parte (o tutti) dei campi restanti dovrebbe far scattare un campanello di allarme. Questo aspetto è strettamente legato alla *consistenza di chiave* dato che parte del problema può essere mitigato con una corretta selezione dell'identificativo usato.
La dimensione dell'univocità è raggiunta quando si può affermare che nessuna entità esiste più di una volta nel dataset. Per raggiungere questo obiettivo possiamo distinguere due sistemi di monitoraggio:

- Controllo programmato: tutti i record vengono inseriti nella base dati e con una certa cadenza e regolarità vengono lanciati meccanismi per individuare e segnalare i potenziali duplicati. Questa soluzione non aggiunge carico nella fase di inserimento ma potrebbe portare a situazioni indesiderate con presenza di duplicati per alcuni lassi di tempo.

- Controllo dinamico: ogni volta che si cerca di inserire un record all'interno della tabella, vengono eseguite verifiche sulla effettiva univocità della tupla e si permette l'inserimento soltanto nel caso in cui il vincolo sia soddisfatto. Se il meccanismo di verifica è efficace, lo stato della tabella è sempre corretto ma il prezzo da pagare è il tempo necessario alla fase di inserimento, che cresce al crescere del numero di dati presenti.

2.1.4 Validità

L'analisi della validità prevede il confronto del dato raccolto con uno specifico formato, con una precisa regola di business, con il range di valori ammissibili, con un pattern particolare o semplicemente verificare che il tipo inserito sia quello atteso. Questa dimensione si applica sia ai dati veri e propri che ai metadati. Anche in questo caso è presente la possibilità di scegliere se effettuare il controllo di validità direttamente in fase di inserimento e terminare o meno l'operazione in base all'esito ottenuto, oppure memorizzare il dato ed eseguire la verifica in un secondo momento. In questo caso è spesso consigliabile prevenire direttamente l'inserimento del

dato non valido visto che i confronti messi in piedi non necessitano di particolari risorse e hanno tempistiche prevalentemente brevi. Tecnica comune per risolvere questa problematica è usare validazione mediante regular expressions e il modo più comune per la misurazione della validità è estrarre la percentuale dei dati ritenuti corretti sull'insieme raccolto.

2.1.5 Completezza

La completezza misura il grado di disponibilità dei dati richiesti rispetto all'intera popolazione di interesse e il livello di dettaglio del dato in funzione del suo scopo. Uno degli aspetti principali è valutare se l'assenza di determinate informazioni nei dati disponibili, abbia o meno introdotto un certo bias[1].
Il problema della completezza può verificarsi a due livelli:

- A livello di riga se ci si riferisce alla mancanza di interi record all'interno della base dati, che porta all'analisi di un sottoinsieme della popolazione che potrebbe fornire una visione distorta della realtà e ottenere risultati che si discostano da quelli desiderati. In alcuni casi tale aspetto è difficile da valutare in quanto potrebbero esserci delle difficoltà nell'individuare con certezza il valore esatto della popolazione considerata pur avendo chiare le caratteristiche della classe di appartenenza.

- A livello di colonna quando si misura la mancanza di specifici attributi all'interno di una tabella. Per mitigare questo problema, alcune possibilità consistono nell'eliminare tutte quelle righe che contengono dei valori mancanti oppure eliminare un'intera colonna, se quel particolare attributo risulta non presente nella maggior parte dei casi. Una alternativa non sempre applicabile è invece quella di completare automaticamente, ad esempio attraverso una stima, i campi sprovvisti di valore tenendo comunque presente che tale operazione andrà ad influire sul risultato finale.

Un'ulteriore insidia per la dimensione di completezza è la presenza di valori di default durante la raccolta dei dati che possono nascondere un tasso di reale risposta basso per uno specifico attributo. Anche in questo caso, piccoli cambiamenti nel meccanismo di acquisizione delle informazioni, possono portare a cambiamenti considerevoli dei risultati ottenuti.
Generalmente la completezza viene calcolata come rapporto del campione posseduto e la cardinalità della popolazione.

[1]Il bias o distorsione è un errore sistematico che fornisce una stima falsata sulla popolazione che ne è affetta. Un campione è distorto se la probabilità che un membro della popolazione sia incluso nel campione dipende dalle caratteristiche oggetto dell'inferenza.

2.1.6 Tempestività

La tempestività può essere misurata come il tempo che intercorre da quando si vuole utilizzare un dato al momento in cui quel dato è effettivamente presente e utilizzabile dal sistema. Sebbene questa dimensione non sia particolarmente importante per quelle applicazioni che fanno uso dello storico per prendere una decisione, dove conoscere l'ultimo dato inserito non va a modificare il comportamento atteso, è assolutamente fondamentale nel mondo dei programmi real-time che per portare a termine il proprio compito sono vincolate dal ricevere il più velocemente possibile le informazioni dall'ambiente di interesse. Si pensi ad esempio ad una applicazione per la gestione dei semafori che, attraverso una telecamera, rileva il numero di pedoni in attesa di attraversare la strada e combinando tale informazione con l'ultimo istante in cui è stato abilitato il passaggio dei pedoni, determini lo stato più appropriato per il semaforo. E' chiaro che il funzionamento di tale logica funziona tanto meglio quanto più aggiornato è il dato richiesto.
La tempestività ha una forte dipendenza dal meccanismo che viene utilizzato per la creazione dello stesso: se si invia l'informazione non appena si verifica l'evento monitorato la latenza si abbassa notevolmente; viceversa se si processano i dati in batch si nota un aumento del ritardo proporzionale alla grandezza del batch. Modifiche a questa pipeline possono cambiare radicalmente il risultato ottenuto.

2.2 Stato dell'arte

La consapevolezza del peso che la qualità dei dati ha nel processo decisionale di ogni azienda è cresciuta di pari passo con la diversificazione e l'aumento delle fonti informative a disposizione delle organizzazioni. Il tema del Data Quality è stato affrontato già a partire dal 1950, con la nascita di molteplici definizioni, ciascuna delle quali inquadra il problema da una prospettiva diversa mentre le ricerche vere e proprie a livello applicativo sono state affrontate a partire dagli anni '90. I contesti di tale ricerca scientifica sono diversi e vanno dalla statistica all'informatica. Di seguito verranno descritti i principali meccanismi che si sono sviluppati negli anni e che sono alla base delle implementazioni moderne.

2.2.1 Miglioramento della raccolta dati

Dallo studio dell'intero processo messo in atto per la raccolta dei dati, possono emergere migliorie che portano ad un innalzamento della qualità degli stessi. Come esempio si può pensare all'utilizzo di menù di selezione piuttosto che campi di input testuali all'interno dei form. Permettere all'utente di selezionare la sua data di nascita attraverso un calendario a comparsa, ad esempio, fa sì che vengano eliminate tutte le possibili incongruenze (ed i relativi controlli) sul formato della data inserito: quante cifre utilizzare per indicare giorno, mese e anno, come separare

tali informazioni e tutti i vincoli sintattici in generale.
Adottare questi accorgimenti non dà però alcuna garanzia sulla effettiva qualità dei dati e restano comunque indispensabili ulteriori meccanismi di analisi per valutare la validità delle informazioni ricevute. In altri casi, inoltre, non è possibile (o è molto complicato) operare sul processo di raccolta: quando alla sorgente dei dati troviamo una macchina, ad esempio un sensore, che si limita ad inviare le misurazioni effettuate, le operazioni possibili risultano piuttosto limitate.

2.2.2 Integrazione

L'integrazione dei dati è l'insieme dei processi tecnici utilizzati per combinare i dati provenienti da origini eterogenee in informazioni utili per il processo di business a cui fanno riferimento e per fornire una visione unificata agli utenti [4]. Tra gli obiettivi di questo metodo troviamo una riduzione della ridondanza dei dati e conseguente aumento della loro integrità, una riduzione dei costi di acquisizione e l'ottimizzazione della trasmissione delle informazioni verso i responsabili delle decisioni.
In ambito Big Data questa tecnica è essenziale per sviluppare analisi avanzate dalle quali estrarre nuova conoscenza e pattern nascosti tra le diverse fonti. Come illustrato in [5] esistono quattro approcci diversi per affrontare il processo di integrazione dei dati:

- Silos: approccio tradizionale, con dati memorizzati in repositories diversi per dipartimento aziendale in maniera isolata e quindi privi di comunicazione e efficace integrazione.

- Data Warehouse: utilizzati spesso come storico di dati strutturati rappresentabili attraverso relazioni e tabelle, raccolgono informazioni da una o più fonti esterne.

- Data Lake: luogo destinato all'archiviazione di dati nel loro formato nativo. A differenza del data warehouse, permette quindi di analizzare dati strutturati e non strutturati, eliminando qualsiasi meccanismo e vincolo di modellazione preventiva alla memorizzazione. L'elevata eterogeneità che è possibile inserire all'interno di questi sistemi di storage si unisce a metodi di conservazione su file sistem distribuiti che garantiscono uno spazio pressoché infinito con costi di espansione delle strutture azzerati.

- Modello integrato: le diverse esigenze di storage, gestione ed analisi, vengono soddisfatte attraverso la presenza contemporanea di modelli su base Data Warehouse e Data Lake.

2.2.3 Extract, Transform, Load (ETL)

Con il termine ETL si identifica un processo suddiviso in tre fasi: estrazione dei dati dalle varie sorgenti, manipolazione degli stessi e caricamento dei dati manipolati all'interno del sistema target come può essere un data warehouse. La fase di validazione e pulizia dei dati avviene nello step di trasformazione, che rappresenta una sorta di prerequisito per la valutazione finale della qualità dei dati. Integrare meccanismi di Data Quality all'inetrno della fase di ETL, va proprio ad evidenziare il gap esistente tra la qualità dei dati estratti e quella desiderata.

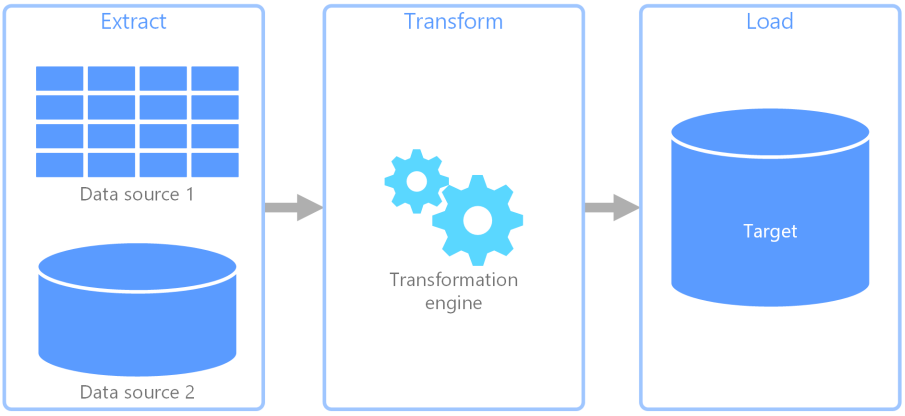

Figura 2.2. Processso di ETL - Microsoft Documentation

In [6] sono specificati gli standard di data quality che vengono comunemente affrontati dal processo di ETL:

- Standardizzazione: definizione di tipo e formato dei dati che verranno trattati.
- Deduplicazione: identificare all'interno delle varie fonti prese in esame la presenza di duplicati attraverso l'analisi sintattica e semantica in modo da unificare anche quei dati che contengono le stesse informazioni suddivise in porzioni diverse.
- Verifica: analisi dei metadati con il fine di individuare malfunzionamenti di sistemi e applicazioni nel processo di raccolta, computazione o inoltro di informazioni.
- Ordinamento: raggruppare e ordinare i dati raccolti per ottimizzare processi di elaborazione e ricerca.

2.2.4 Data matching

Consiste nell'operazione di confronto di record diversi, potenzialmente provenienti da sorgenti diverse, che fanno riferimento alla stessa entità. La situazione ideale per questa attività sarebbe quella di avere a disposizione, su tutte le sorgenti usate, una stessa chiave primaria che assume valori coerenti e che permetta di associare ogni record ai corrispondenti provenienti dalle altre fonti. Nella maggior parte dei casi reali, i record non presentano queste caratteristiche e il problema diventa più complicato e molto più dispendioso in termini computazionali. In questi casi la tecnica può ancora essere applicata utilizzando il prodotto cartesiano tra le tabelle che contengono le informazioni, identificando all'interno dello spazio ottenuto un sottoinsieme di interesse e definendo una metrica che possiamo chiamare 'distanza' per determinare il livello di correlazione tra i record.
Un ulteriore fattore di complicazione in questa situazione si ha a causa delle modifiche che i dati subiscono nel tempo e che potrebbero non essere riflesse allo stesso modo su tutte le sorgenti. Si pensi ad esempio ad un individuo che modifica il proprio indirizzo: se il cambiamento viene registrato solo da alcuni database tra quelli presi in considerazione, la probabilità di ottenere 'falsi negativi' tra i vari matching di quel record aumentano.
In caso di database di grandi dimensioni, il prodotto cartesiano necessario nella fase iniziale, potrebbe richiedere tempistiche non accettabili per l'analisi. Per mitigare questo problema, forme di indicizzazione possono essere applicate, sebbene la scelta di tale indice risulti un compito tutt'altro che banale [7].

2.2.5 Vincoli di dominio

A causa delle numerose problematiche di applicazione e dei costi computazionali di molti dei metodi descritti, è nata la necessità di avere meccanismi di ispezione e correzione basati su vincoli di dominio. La grande diffusione di tali principi ha portato alla creazione di algoritmi oltre che veri e propri strumenti che consentono di analizzare la qualità dei dati.
Una delle soluzioni che viene impiegata maggiormente in questo ambito è quella basata sulla definizione, da parte di esperti del dominio di applicazione, di una serie di regole che devono essere rispettate dai dati affinché questi possano ritenersi di qualità. Le regole di business forniscono supporto principalmente in due modi: possono automatizzare le decisioni operazionali e validare i dati secondo le politiche interne.
Una seconda soluzione che fa uso di vincoli di dominio è quella basata su dipendenze funzionali che permettono di associare il valore di un attributo al valore di un altro. Anche in questo caso le dipendenze vengono individuate dagli esperti del settore e, dal punto di vista implementativo, vengono spesso realizzati attraverso i trigger. Questi permettono di esprimere regole anche piuttosto complesse che vengono verificate in maniera automatica quando determinate condizioni risultano

vere. Il problema principale dei trigger è rappresentato dalla possibilità di essere eseguiti in cascata: ogni funzione può, nel corso delle sue operazioni, richiamare a sua volta uno o più trigger, aumentando anche notevolmente il tempo di esecuzione oppure, nella peggiore delle ipotesi, portare al verificarsi di situazioni caratterizzate da cicli infiniti in caso di cattiva implementazione.
Il terzo approccio possibile al problema della qualità dei dati, è rappresentato dalle soluzioni basate sull'apprendimento. Se nei metodi precedenti era compito dell'esperto di dominio definire le regole per la valutazione dei dati, in questo caso è il software stesso che, analizzando l'input, costruisce le metriche necessarie per la classificazione. La difficoltà più grande presente in questo tipo di algoritmi è la validazione dei risultati restituiti e la necessità, per alcune soluzioni, di dati etichettati indispensabili nella fase di training.

2.2.6 Nuove problematiche nate con i Big Data

Le caratteristiche tipiche dei Big Data hanno portato alla nascita di nuove sfide dal punto di vista del Data Quality. I punti chiave sono dati da: volumi elevatissimi di dati (dai TeraByte in su), la velocità con cui i dati vengono generati e la necessità di elaborare gli stessi in tempi prossimi al real-time, la varietà delle informazioni che vengono raccolte dove troviamo formati diversi e valori strutturati così come non strutturati e infine una scarsa densità dei dati, tendenzialmente inversamente proporzionale alla quantità.[8]

Figura 2.3. Le 4 V dei Big Data - VectorStock.com

- Volume e Varietà: l'aumento del numero di sorgenti eterogenee contenenti sia dati strutturati che non, ha portato all'aumento di conflitti e inconsistenze tra i dati. Con volumi di modeste dimensioni, possono essere applicate verifiche manuali da parte degli esperti, impraticabili per ovvi motivi quando si passa ai livelli di centinaia di GigaByte.

- Velocità: i dati di tipo non strutturato che rappresentano ormai la percentuale più grande delle informazioni disponibili, richiedono tempistiche elevate per l'elaborazione. Queste tempistiche si vanno inoltre a scontrare con le necessità di ridurre il più possibile il ritardo di memorizzazione.

- Veridicità: allarga il concetto di qualità del dato a quanto affidabile risulta la sorgente, il tipo e il processo di raccolta e analisi.

Capitolo 3

Outlier Detection

Un outlier è definito come un'osservazione che si discosta dagli altri dati al punto da sollevare sospetti sul fatto che tale valore sia stato generato dallo stesso meccanismo o da uno differente [Hawkins - 1980]. Si tratta sostanzialmente di valori estremi che sono più lontani dalla media dei dati di quanto sarebbe logico aspettarsi. Un esempio grafico è rappresentato nella Figura 3.1.
Le principali cause di questo tipo di misurazioni sono:

- Errore umano: possibilità di errore da parte dell'utente che inserisce il dato.
- Errori della strumentazione: il malfunzionamento degli strumenti utilizzati per la raccolta dati possono portare alla generazione di outliers.
- Errori di elaborazione: durante la manipolazione del dato, errori logici potrebbero causare la generazione di valori lontani da quelli reali.
- Cattiva valutazione della distribuzione dei dati.
- Errore casuale non controllabile.

Queste anomalie possono presentarsi in diverse forme: è possibile trovare outlier sotto forma di punti isolati che rappresentano singoli valori che si discostano dal resto della distribuzione, *contextual outliers* che possono essere considerati come una forma di rumore all'interno dei dati raccolti e *collective outliers* che rappresentano invece un possibile segnale per la scoperta di un nuovo fenomeno, in quanto il comportamento anomalo è esteso a più campioni che nella visione d'insieme differiscono dal risultato atteso.

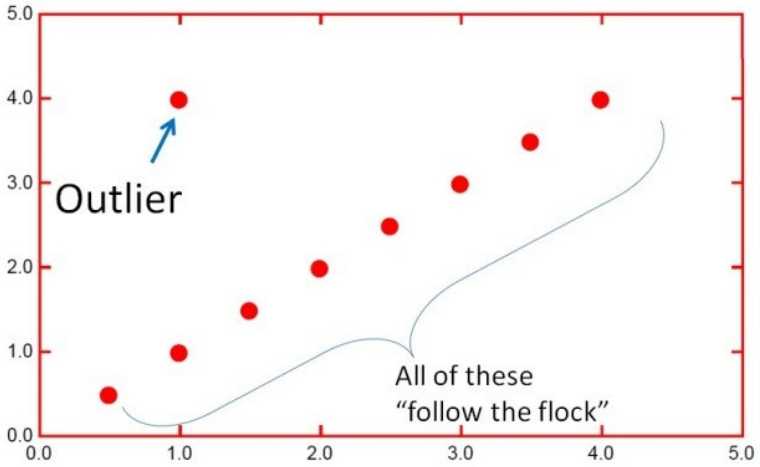

Figura 3.1. Rappresentazione di un outlier - Lead Up Collective

3.1 Anomaly detection per migliorare la qualità dei dati

L'identificazione degli outliers è un aspetto fondamentale del Data Quality dato che consente di valutare la qualità delle informazioni utilizzando i dati stessi come strumento. Le dimensioni che risultano maggiormente influenzate sono l'accuratezza e la consistenza. Sebbene le anomalie possano sembrare indicatori di errori nella teoria o all'interno della pipeline di elaborazione, la presenza di una piccola percentuale di outliers è data per scontata quando il numero di dati raccolti aumenta. Eliminare il prima possibile tali dati da quelli utilizzati per le decisioni di business è fondamentale per evitare comportamenti indesiderati causati dall'influenza degli outliers sulle statistiche generali e aumentare il valore dei KQI[1].
In [9] viene evidenziato come il riconoscimento delle caratteristiche inusuali derivanti dai dati anomali che vengono raccolti, riesca a fornire strumenti utili per le decisioni specifiche relative al dominio di appartenenza. Tra gli esempi applicativi troviamo:

- Intrusion Detection Systems: attraverso informazioni come il traffico di rete, i comandi invocati a livello di sistema operativo e il carico di lavoro di CPU e memoria, è possibile identificare attività dannose al sistema ed eventuali accessi non autorizzati.

[1]I Key Quality Indicators (Indicatori della qualità del dato) forniscono una visione generale dello stato delle informazioni secondo gli standard definiti dal contesto analizzato, pesando opportunamente le dimensioni coinvolte nella verifica della qualità.

- Uso non autorizzato di Carte di Credito: il luogo in cui la carta viene utilizzata e l'importo della spesa sono strumenti utili al riconoscimento di operazioni sospette.

- Raccolta dati da sensori: un cambiamento brusco tra valori misurati in istanti di tempo successivi e non troppo distanti, è spesso sinonimo di un malfunzionamento nel contesto di acquisizione per mezzo di sensori e macchinari.

- Studio della terra: attraverso le anomalie che vengono identificate in ambito meteorologico o terrestre forniscono preziose informazioni circa l'attività dell'uomo o l'andamento ambientale che può esserne causa.

3.2 Caratteristiche dei metodi usati

I metodi di identificazione degli outliers possono essere divisi in due categorie: metodi monodimensionali e metodi multidimensionali. Soprattutto le tecniche meno recenti per affrontare il problema di anomaly detection, si focalizzano su ogni variabile in maniera indipendente, identificando i valori estremi relativi alla distribuzione dei dati. Per esempio, se la distribuzione è di tipo normale, i dati che differiscono dalla media più di una certa soglia vengono categorizzati come outliers. Uno dei problemi che sorge è che i valori di media e deviazione standard, che sono usati per stabilire se un dato dovrebbe o meno essere considerato nell'analisi, sono a loro volta influenzati da queste anomalie e causano falsi positivi così come falsi negativi. Uno degli approcci più utilizzati per mitigare il problema è il metodo di troncamento, dove un numero prefissato di dati agli estremi della distribuzione, viene escluso dal calcolo dei parametri in questione.
Il limite più grande di questa tecnica è dato dal fatto che gli outliers rispetto ad una data relazione esistente tra due o più variabili non vengono identificati dato l'approccio monodimensionale. Per questo motivo si rendono necessari nuovi meccanismi in grado di valutare l'interazione tra diverse variabili all'interno del data set.
Diversi studi mostrano come questo ultimo tipo di implementazione possa avere un impatto decisivo nella ricerca di valori non attesi, al punto da ritenersi indispensabili ed estremamente più efficaci dell'approccio monodimensionale. Un confronto approfondito è presentato in [10] mentre in [11] viene analizzato il contributo di un meccanismo di outlier detection di tipo multidimensionale in un caso pratico.
Una ulteriore distinzione può essere fatta sulla base dell'output restituito dai metodi di rilevazione delle anomalie. La prima possibilità prevede la definizione di un punteggio tanto più elevato quanto più il dato si discosta dal valore atteso. Questo punteggio può essere usato per inserire i dati in una sorta di classifica che tiene conto di tutti gli attributi considerati ma non fornisce una visione puntale sui dati che devono essere considerati effettivamente degli outliers.

L'alternativa a questa soluzione è utilizzare una etichetta binaria come output che indica la validità del dato. In questo caso l'informazione ricevuta è meno completa del caso precedente ma fornisce il dato indispensabile per il processo decisionale. Utilizzando opportune soglie è inoltre possibile convertire un risultato basato su punteggio in una soluzione binaria.
Lo scenario più diffuso nel mondo reale per questo tipo di applicazioni è caratterizzato dall'assenza di dati di training per il modello, si parla quindi di una modalità *unsupervised*. Nel caso in cui fosse possibile allenare il modello fornendo sia dati appartenenti alla normale distribuzione che quelli considerati anomalie, si parla di modalità *supervised*. Lo stato intermedio, dove si hanno a disposizione i dati facenti parte esclusivamente ad una categoria, viene definito come scenario *semi-supervised*.

3.3 Scelta del modello

La scelta del modello da utilizzare deve tenere in considerazione il tipo dei dati che si vogliono analizzare, la disponibilità di campioni rappresentativi della classe delle anomalie, la dimensione della base dati in esame e la necessità di avere un risultato interpretabile. Risulta infatti preferibile ottenere un modello in grado di motivare il perché un determinato dato viene categorizzato come anomalia in modo da ampliare la visione sugli elementi da esaminare all'interno del contesto specifico per la ricerca degli outliers.
Di seguito viene riportata una panoramica dei possibili metodi di rilevazione degli outliers cercando di fornire anche un contesto applicativo appropriato per il relativo utilizzo.

3.3.1 Metodi statistici e probabilistici

Nei metodi statistici il modello viene costruito sulla base di una precisa distribuzione statistica dei dati considerati. La scelta della distribuzione rappresenta quindi l'assunzione chiave di questi metodi e i relativi parametri, utili per la definizione degli outliers, vengono calcolati considerando i dati come appartenenti a tale distribuzione. Le anomalie sono individuate selezionando i punti che hanno la più bassa probabilità di essere stati generati dalla distribuzione presa in considerazione (ad esempio i dati che distano più di tre volte la devizione standard rispetto alla media). Questo tipo di tecniche possono essere applicate sia per verifiche monodimensionali che multidimensionali.
Il principale vantaggio dei metodi statistici è la possibilità di essere applicati potenzialmente a qualsiasi tipo di dato, a patto che sia disponibile un appropriato modello generativo. Gli svantaggi si hanno in termini di robustezza, dato che tutti i valori, compresi gli outliers, vengono utilizzati per il calcolo dei parametri: media e deviazione standard risultano particolarmente sensibili a valori anomali. Considerare i dati come generati da un particolare tipo di distribuzione, può essere inoltre

inappropriato in determinate applicazioni.
Una delle tecniche basilari di identificazione degli outliers prende il nome di *Extreme-value Analysis* e si concentra su un particolare tipo di anomalie: i valori agli estremi della distribuzione vengono considerati come outliers. Il punto chiave sta nell'identificare le code della distribuzione. L'esempio più semplice si ha nel caso di distribuzione normale utilizzando lo *Z-score* come metrica che rappresenta la distanza tra un determinato valore e la media, in termini di multipli di deviazione standard [12]. Una volta che i dati risultano centrati e scalati, i valori alle estremità vengono considerati degli outliers.
Lo Z-score viene calcolato attraverso la formula 3.1 dove μ rappresenta la media e σ la deviazione standard mentre x identifica il valore considerato.

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

Come mostrato in figura 3.2, una volta calcolato il valore di z si sceglie una soglia oltre la quale considerare i dati come anomalie. Una buona regola di base consiste nello scegliere z in un range che va da 2.5 a 3.5 in valore assoluto. Uno degli

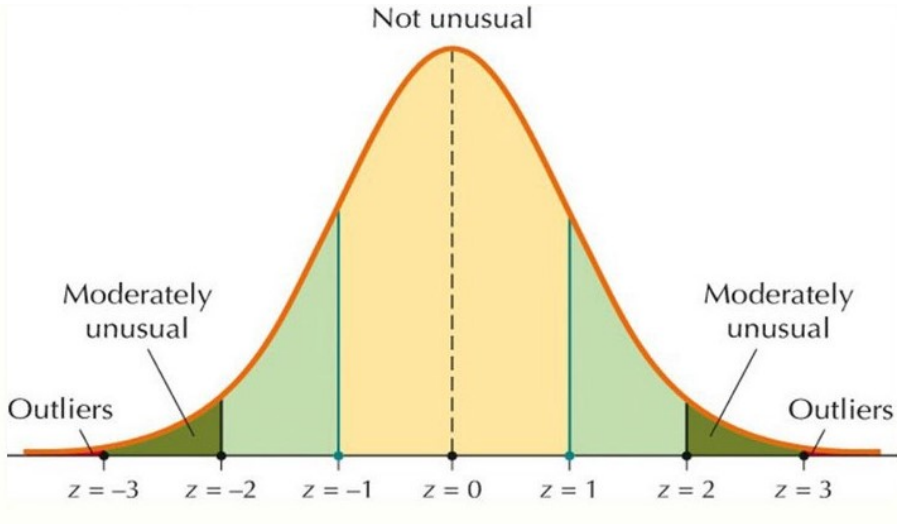

Figura 3.2. Utlizzo dello Z-score per individuare gli outliers - KDNuggets

aspetti da tenere presente quando si studia un metodo di analisi dei valori agli estremi è dato dal fatto che la concezione di outlier si allontana dalla definizione fornita all'inizio del capitolo per accostarsi al concetto di valore alla coda della distribuzione. La confusione tra outlier e dato all'estremità non è affatto rara, soprattutto nei contesti di analisi di tipo multidimensionale.
Questi metodi trovano tuttavia largo uso all'interno degli algoritmi di identificazione degli outliers, in modo particolare nelle fasi finali: molti meccanismi di anomaly detection forniscono come risultato un valore numerico, una sorta di punteggio

che trova nei valori estremi il posizionamento dei dati considerati lontani dalla normalità.

3.3.2 Metodi basati sulla distanza e prossimità

L'idea generale condivisa dei metodi che fanno uso del concetto di prossimità è basata sull'assunzione secondo cui i punti definiti outliers sono posizionati in zone scarsamente popolate rispetto ai restanti valori. Questi meccanismi sono tra i più utilizzati nel mondo dell'identificazione delle anomalie. La prossimità può essere definita in diversi modi, tra i più diffusi troviamo:

- Basata su cluster: per determinare il livello di anomalia di un dato si considerano metriche tipiche dei cluster come la dimensione del cluster più vicino, la distanza del punto considerato dagli altri cluster, la non appartenenza del punto a nessuno dei cluster trovati o una combinazione di questi fattori.
- Basata sulla distanza: si analizza la distanza di ogni punto dai k punti più vicini e si definiscono outliers i valori con più alto punteggio di distanza.
- Basata sulla densità: il numero di punti all'interno di una specifica regione di spazio viene utilizzato per definire la densità locale. Tale valore può essere poi convertito in un punteggio di anomalia utile per la classificazione finale.

Sebbene tutte queste tecniche siano tra loro legate, sono presenti differenze sostanziali che portano ad una serie di vantaggi e svantaggi propri di ogni metodologia. La differenza principale tra clustering e metodi basati sulla densità è rappresentata dalla segmentazione: con il clustering si vanno a segmentare i punti mentre con tecniche basate sulla densità si segmenta lo spazio dei dati. Ulteriore disuguaglianza è data dal livello di granularità usato per l'analisi: nel caso di clustering e algoritmi basati sulla densità, i dati vengono aggregati prima del partizionamento dei punti o dello spazio necessario per la ricerca degli outliers. Nei metodi basati sulla distanza, invece, questa viene calcolata a partire dai dati originali, andando quindi ad operare su un livello di dettaglio maggiore.
Il fattore di granularità ha un impatto significativo sul costo computazionale dei vari algoritmi. Quando si utilizza la distanza come metrica, ci si prepara ad affrontare tipicamente un algoritmo dal costo quadratico, dato che è necessario calcolare la distanza di un punto da tutti gli altri. Quando l'output richiesto è di tipo binario, forme di indici e pruning[2] possono velocizzare il meccanismo. Lo svantaggio è che queste tecniche possono essere adoperate solo in casi particolari.
Gli algoritmi di prossimità sono tra i più utilizzati vista la loro semplicità e grande interpretabilità. Questi metodi riescono ad evidenziare in maniera naturale sia

[2]Criterio di arresto che ha lo scopo bloccare le ulteriori fasi della computazione una volta ottenuto un risultato significativo e sufficiente, cercando quindi di ridurre i tempi necessari

forme di rumore che outliers in base alla metrica che viene utilizzata per definire la prossimità: la non appartenenza a nessun cluster è particolarmente utile nell'identificare il rumore, mentre l'utilizzo di metriche di densità e distanza favorisce la scoperta di anomalie.
Uno tra gli algoritmi basati sulla densità tra i più utilizzati è il *DbScan*. Ogni cluster viene definito attraverso il concetto di '*Density reachability*': "Il punto q è density-reachable da p se c'è una sequenza $p_1,\ldots,p_n$ di punti con $p_1 = p$ e $p_n = q$ dove ogni p_{i+1} è density-reachable direttamente da p_i" [13]. La distanza è definita dal valore ϵ, primo parametro dell'algoritmo. Il secondo parametro, *MinPts*, influisce sulla suddivisione in categorie dei punti che si vanno ad analizzare:

- Core point: un punto viene categorizzato come 'Core' se nel suo vicinato, definito da ϵ, troviamo un numero di punti maggiore o uguale al valore di *MinPts*.
- Boredr point: un punto viene definito di bordo se è posizionato all'interno di un cluster ma nel suo vicinato non contiene ulteriori 'Core points'.
- Outlier: è un punto che risiede da solo all'interno del proprio cluster: non è 'density reachable' da nessun altro punto.

Nella Figura 3.3 è possibile osservare un esempio di classificazione dove con A sono rappresentati i core points, con le lettere B e C troviamo i border points e gli outlier sono identificati dalla lettera N.
Se A è un 'Core point' esso forma un cluster con tutti i punti raggiungibili nel suo

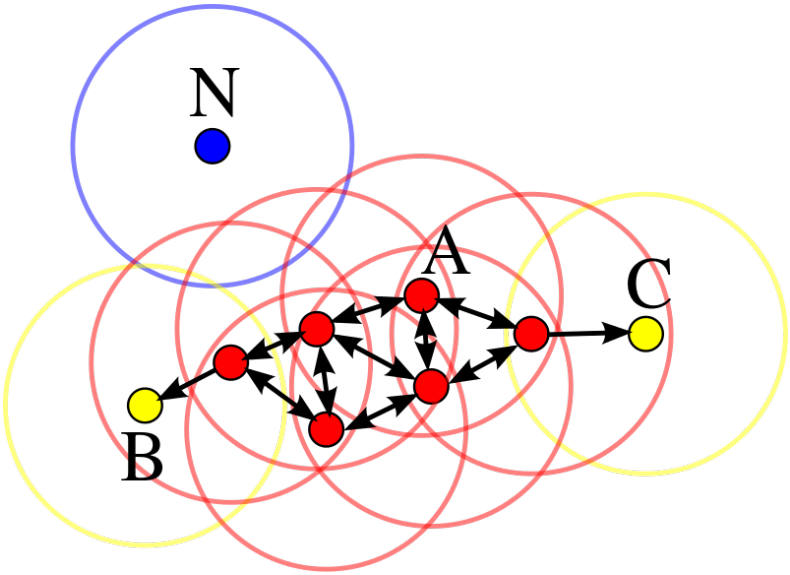

Figura 3.3. DBScan e classificazione dei punti - Wikipedia.org

vicinato. La relazione di raggiungibilità è di tipo non simmetrico, per questo due punti p e q vengono detti '*density-connected*' se esiste un punto intermedio *o* che risulti density-reachable sia con p che con q [14].
Una delle maggiori limitazioni degli algoritmi basati sulla prossimità si riscontra nel caso di dati con un elevato numero di dimensioni. Le dimensioni aggiuntive che vengono considerate non sempre sono rilevanti ai fini della identificazione delle anomalie, tuttavia il rumore presente nelle informazioni appartenenti a tali dimensioni impatta pesantemente sui parametri di prossimità e distanza e maschera gli outlier presenti. All'aumentare del numero di dimensioni, ogni punto tende sempre di più ad essere equidistante rispetto ad altri: in queste situazioni il problema della qualità degli outlier trovati prevale anche più della complessità dell'algoritmo e del suo costo computazionale.

3.3.3 Identificazione degli outlier con dimensionalità elevata

I dati con un elevato numero di dimensioni da tenere in considerazione quando si applicano meccanismi di identificazione delle anomalie, portano ai casi più complicati all'interno di questo settore. Il punto principale è che molte dimensioni potrebbero avere un grado di correlazione basso e non essere particolarmente rilevanti ai fini dell'analisi degli outlier. Oltre a non portare informazioni utili, le dimensioni in eccesso incrementano il livello di sparsità dei dati e indeboliscono il concetto di distanza, portando a punteggi di anomalia per i dati meno distinguibili: dato che gli outlier sono definiti come quei dati che risiedono in zone scarsamente popolate, l'aumento della sparsità porta ad una inefficace discriminazione dei comportamenti non ordinari.
In queste situazioni, gli outlier vengono identificati in maniera più decisa ricercando all'interno di sotto spazi locali delle caratteristiche rilevanti con numero minore di dimensioni. Questo approccio è noto come '*subspace outlier detection*' la cui assunzione è data dal fatto che spesso le anomalie sono nascoste in sotto spazi di ridotte dimensioni e tale comportamento irregolare viene nascosto dagli altri attributi osservando dallo spazio completo.
La scelta del giusto sotto spazio è probabilmente la sfida più impegnativa di questi algoritmi, alla quale si affianca il grande costo computazionale della ricerca di tutte le possibili combinazioni di interesse e l'analisi all'interno di ognuna di esse dei pattern locali, in grado di fornire informazioni utili alla ricerca delle anomalie.
La definizione di spazi di dimensione ridotta è un problema che non nasce con il data quality e la ricerca degli outlier ma è diffuso anche tra i meccanismi di clustering. Bisogna però osservare come la complessità nel settore di ricerca delle anomalie sia decisamente più elevata. Problemi come il clustering sono basati su dati aggregati, dove l'effetto di dati non attesi viene smorzato. L'uso di queste forme aggregate fornisce suggerimenti piuttosto deboli nel campo di esplorazione dei sotto spazi e quando questo si traduce in una omissione di dimensioni particolarmente rilevanti,

l'effetto può rivelarsi più dannoso rispetto all'inclusione di dimensioni non particolarmente utili.
In generale selezionare un singolo sotto spazio di interesse per ogni punto può portare a risultati impredicibili. Diventa quindi indispensabile combinare più risultati derivanti da diversi spazi di dimensioni ridotte e affrontare il problema con un meccanismo di tipo '*ensemble*'[3].

[3]Nell'apprendimento automatico indica una serie di metodi che cercano di migliorare le performance di predizione attraverso l'uso combinato di modelli multipli.

Capitolo 4

Soluzione proposta

Lo scopo di questo capitolo è fornire una panoramica della situazione attuale e descrivere le metodologie e gli algoritmi utilizzati per migliorare il processo originale. Verranno inoltre definiti gli obiettivi preposti e analizzate le piattaforme su cui la soluzione si basa.
Il fine ultimo del lavoro è quello di rendere più efficace l'attuale meccanismo di raccolta e analisi di informazioni provenienti da veicoli, tra cui la velocità media durante la missione, il consumo medio di carburante, il peso rilevato e così via. In particolare ci si concentrerà sullo sviluppo di algoritmi in grado di riconoscere in maniera automatica il livello di validità dei dati con l'intento di scartare gli outlier dal resoconto finale che viene fornito ai responsabili per le opportune verifiche. La struttura dei dati, con i dettagli sugli attributi inclusi, è riportata nel paragrafo 5.1.

4.1 Situazione attuale e obiettivi

Nella figura 4.1 è rappresentato uno schema di alto livello della soluzione aziendale di raccolta, trasformazione e salvataggio dei dati per il problema affrontato.

Le informazioni generate dai veicoli vengono inviate all'interno di un *Data Lake* che rappresenta il punto di contatto tra il cliente e l'azienda. Questo servizio semplifica la fase di inserimento e archiviazione dei dati, mettendo a disposizione la possibilità di effettuare analisi sia di tipo batch che di tipo streaming. Il passaggio successivo consiste nella manipolazione dei dati attraverso una funzione di aggregazione che applica logiche diverse in base al tipo di dati approdati sul Data Lake: molte delle dimensioni esaminate forniscono informazioni considerando valori massimi, minimi e medi. Durante questa fase si definiscono dunque i valori degli attributi, distintamente per ogni veicolo attivo, associando i dati attenendosi alla missione di appartenenza dal punto di vista temporale e formando quindi un risultato di tipo strutturato. L'output della funzione di aggregazione viene riportato

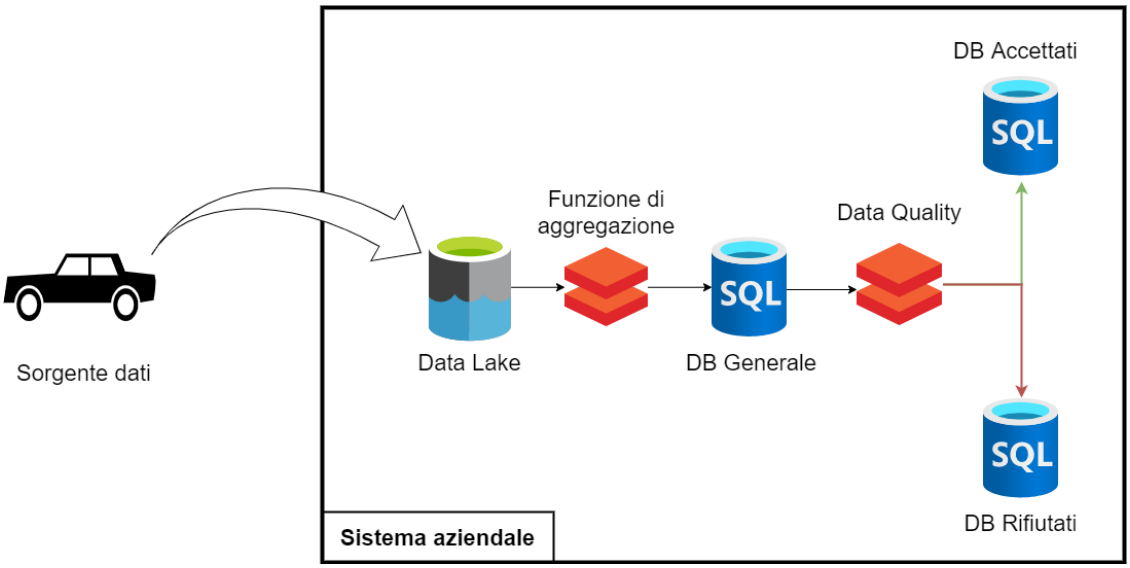

Figura 4.1. Sistema di memorizzazione dati aziendale

nel primo livello di storage aziendale che ha lo scopo di accumulare i dati a livello giornaliero prima che questi possano essere esaminati dalla funzione di data quality che ne determina la validità.
La funzione di Data Quality agisce secondo regole di business statiche, analizzando una per volta le dimensioni di interesse: prima si verificano i valori di latitudine e longitudine, poi quelli relativi alla velocità, successivamente si passa al tipo di carburante e al relativo consumo e così via. L'ultimo step della pipeline consiste nell'inserire i dati strutturati all'interno dello storico. A seconda del risultato restituito dalla funzione di data quality, le informazioni vengono inserite in due database diversi: se i dati sono stati categorizzati come validi, vengono memorizzati all'interno dello storico dei dati accettati che viene poi effettivamente usato per la generazione dei resoconti richiesti; se invece sono state riscontrate anomalie dalla fase di validazione, i dati vengono salvati in una base dati differente che contiene tutti i record rifiutati, associati alla motivazione del rigetto.
L'obiettivo del lavoro svolto è quello di migliorare la fase di certificazione della qualità dei dati sotto diversi punti di vista. In primo luogo si vuole eliminare qualsiasi tipo di soglia fissa all'interno delle regole utilizzate e rendere il meccanismo più flessibile adattando il valore limite alla categoria di appartenenza della sorgente dati. L'unica forma di adattamento presente nel modello originale consiste nel differenziare la discriminazione in base al risultato desiderato (analisi in termini di valori massimi, minimi o medi), che in molti casi risulta insufficiente per una corretta categorizzazione. Prendendo come esempio l'attributo relativo alla velocità, la soluzione attuale prevede una divisione in range per i valori che vanno fino ai 150

km/h mentre tutti quelli superiori a tale soglia vengono considerati outlier. Questa suddivisione può essere appropriata per una normale vettura ma potrebbe risultare eccessiva se il tipo di sorgente fosse diversa (ad esempio un autocarro). Ulteriore aspetto da tenere presente è l'elevato numero di dimensioni delle informazioni. Come già evidenziato nel corso dei capitoli precedenti, prendere in considerazione un solo attributo per volta va a nascondere tutti quei comportamenti anomali che risulterebbero lampanti attraverso un'analisi su più fronti.
Per sopperire queste problematiche sono state implementate due diverse strategie, la prima con lo scopo di identificare in maniera automatica i dati non validi prendendo in considerazione tutti gli attributi (o buona parte degli stessi, considerati quelli di maggiore interesse), mentre la seconda strategia si concentra su specifiche coppie di attributi strettamente legati tra loro e utili nella definizione di report puntuali piuttosto che comprendenti tutte le dimensioni presenti.
Gli algoritmi adoperati e il relativo funzionamento sono illustrati nel paragrafo 4.3.
Valutare l'efficacia effettiva di un algoritmo di outlier detection non è banale, soprattutto quando si fa riferimento ad algoritmi non supervisionati dove il valore vero della categoria non è disponibile a priori. L'output fornito dalla maggior parte dei meccanismi di identificazione delle anomalie consiste in un punteggio di anomalia che attraverso una soglia viene convertito un valore binario che rappresenta la categoria di appartenenza: outlier o inlier. Se il valore di soglia risulta essere troppo restrittivo, con l'obiettivo di minimizzare il numero di dati etichettati come outliers, il numero di falsi negativi aumenta di conseguenza. Se viceversa l'algoritmo classifica troppi punti come non validi, si verifica un incremento considerevole dei falsi positivi. Per avere un idea più chiara sulle performance e trovare il giusto compromesso, possono essere utilizzate le grandezze di *precisione* e *recupero (o richiamo)*.
La precisione (*Precision in Figura 4.2*) definita in 4.1, rappresenta il numero di oggetti etichettati correttamente come appartenenti alla classe rispetto al totale delle volte che quella classe compare nella classificazione [15]. Questa grandezza non tiene conto dei falsi negativi, che potrebbero essere in numero eccessivo anche in caso di modello sufficientemente preciso.

$$Precisione = \frac{Veri\ Positivi}{Veri\ Positivi + Falsi\ Positivi} \tag{4.1}$$

Il richiamo (*Recall in Figura 4.2*) invece misura la sensibilità del modello. "E' il rapporto tra le previsioni corrette per una classe sul totale dei casi in cui si verifica effettivamente" [16]. La dimensione tralasciata da tale misura è quella dei falsi positivi. La formula 4.2 esprime quanto detto in maniera più formale.

$$Richiamo = \frac{Veri\ Positivi}{Veri\ Positivi + Falsi\ Negativi} \tag{4.2}$$

Nella figura 4.2 può essere apprezzata una visualizzazione grafica delle misure appena descritte. Sul lato sinistro della figura troviamo i punti di interesse (relevant

items) mentre all'interno della circonferenza i valori selezionati dall'algoritmo (selected items).
Una possibile grandezza che mette in relazione Precisione e Richiamo è la *Precision Recall curve* che mostra il comportamento dell'algoritmo secondo queste misure considerando diverse soglie. Un'area estesa sotto la curva rappresenta allo stesso tempo alta precisione ed elevato richiamo [17].
Dal punto di vista aziendale, si è deciso di prestare maggiore attenzione alla misu-

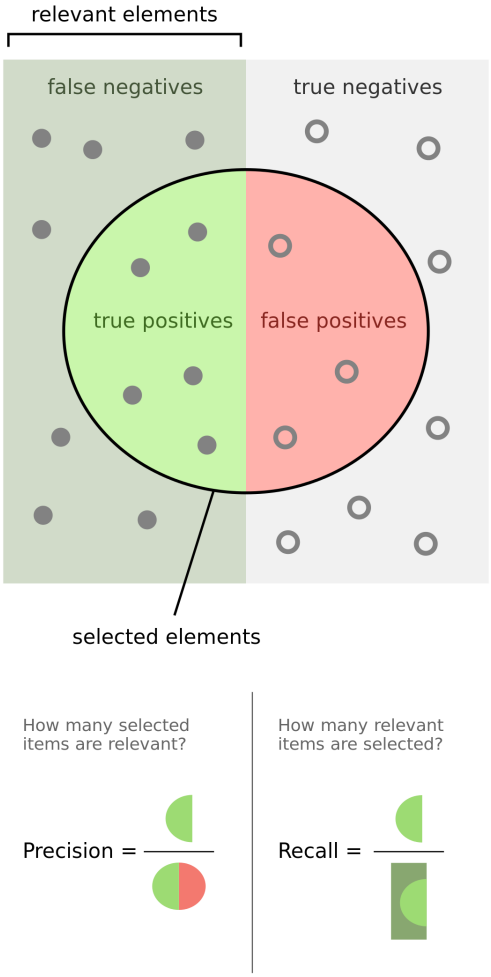

Figura 4.2. Precisione e Richiamo - Wikipedia.org

ra del richiamo in quanto si è verificato che la presenza di falsi negativi influisce in maniera considerevole sui risultati finali, mentre un valore leggermente più elevato di falsi positivi non impatta in modo particolarmente significativo.

4.2 Basi teoriche

La qualità dei dati rappresenta una delle più grandi preoccupazioni per tutte quelle società che fanno del *data-driven processing* la loro arma principale per la risoluzione dei problemi. I processi tradizionali di Data Quality si sono evoluti nel corso del tempo ma non abbastanza da soddisfare i requisiti di velocità ed efficienza richiesti dalle organizzazioni al punto da essere tutt'ora rilevati come le operazioni che fanno da collo di bottiglia nei processi aziendali.
Con l'entrata in campo dei Big Data, Il Machine Learning è diventato uno degli approcci dominanti in tema Data Quality, rinnovando implementazioni e metodologie. Una delle transizioni più importanti sta nel passaggio da regole decisionali statiche ad un approccio sempre più dinamico ed adattivo. L'apprendimento automatico ha il potenziale di riuscire a valutare la qualità delle informazioni, rilevare dati mancanti così come ridurre costo e complessità del lavoro degli specialisti dell'ambito. Attraverso la segnalazione di record incorretti o sospetti basandosi su forme di probabilità, il Machine Learning può inoltre sostituire parzialmente il ruolo dei *Data Stewards*.

4.2.1 Machine Learning

Una delle definizioni che meglio rispecchia il concetto di apprendimento automatico è quella fornita dall'americano Tom Michael Mitchell: "si dice che un programma apprende dall'esperienza E con riferimento a alcune classi di compiti T e con misurazione della performance P, se le sue performance nel compito T, come misurato da P, migliorano con l'esperienza E". Questa branca dell'informatica si allontana dal concetto più tradizionale del settore, ovvero scrivere il codice necessario all'algoritmo per prendere decisioni e fornire un output, per sfruttare direttamente i dati in input per estrarre le informazioni necessarie e fornire le risposte basandosi su metodi matematici. L'apprendimento si ottiene quando queste risposte ottenute diventano migliori con l'aumentare dei dati in ingresso analizzati. Spesso questo comportamento si raggiunge con l'uso di funzioni di "premio", in vista di un output corretto e di "penalizzazioni" quando si riscontra un errore.
Il Machine Learning può essere realizzato seguendo approcci diversi in base alla quantità di informazioni che si forniscono all'algoritmo:

- Apprendimento supervisionato (Supervised Learning): si sfruttano le conoscenze sui dati che si hanno a priori per definire in maniera esatta quale dovrebbe essere l'output dell'algoritmo. Possiamo definire quindi il compito principale di questo tipo di algoritmi come quello di trovare una funzione che approssima nel miglior modo possibile la relazione tra i dati in ingresso e il risultato in uscita una volta forniti i campioni e il risultato atteso. Il campo di applicazione più diffuso per questo tipo di meccanismi è quello della classificazione. L'analisi principale da effettuare con questa strategia è quella

sulla complessità del modello creato: modelli con elevata complessità portano spesso a situazioni di overfitting[1] su un numero ristretto di campioni.

- Apprendimento non supervisionato (Unsupervised Learning): nella strategia non supervisionata, non si hanno a disposizione i valori delle etichette di output per i dati forniti e l'obiettivo è quello di scovare una sorta di pattern naturale all'interno dell'input fornito. Compito tipico portato avanti con tecniche di unsupervised learning è il clustering. Dato che i dati sono sprovvisti delle etichette corrette, non è semplice valutare le performance di questi algoritmi. Risultano indispensabili quando è impossibile (o estremamente complesso) per un umano intercettare strutture significative nei dati analizzati: attraverso questi algoritmi possono essere trovate strade iniziali che sono poi verificate e migliorate dagli esperti [18].

- Apprendimento semi-supervisionato (Semi-supervised Learning): si tratta di un modello ibrido nel quale solo una piccola porzione di dati sono forniti delle etichette corrette e questi vengono usati dall'algoritmo come strumento in più, rispetto al modello unsupervised, per la definizione delle regole.

La soluzione al problema affrontato fa uso di algoritmi di tipo non supervisionato sia per quanto riguarda l'analisi dei dati in tutte le loro dimensioni, sia per lo studio di outliers nel caso di dimensionalità ridotte.

4.2.2 Apache Spark

La maggior parte delle operazioni descritte nella pipeline in Figura 4.1 vengono eseguite sfruttando Apache Spark.
Apache Spark è un Framework di elaborazione parallela open source che supporta l'elaborazione in memoria per migliorare le prestazioni delle applicazioni che analizzano Big Data rispetto alle alternative basate su disco [19]. Gli obiettivi di Spark all'interno del contesto Big Data sono la generalità, la bassa latenza, la tolleranza ai guasti e la semplicità. Grazie a queste caratteristiche si riescono a raggiungere velocità 100 volte maggiori rispetto ad Hadoop [20]. La nascita di questo framework è dovuta al costo delle letture e scritture da disco tipiche dei job iterativi basati sul meccanismo di MapReduce. Il crollo dei costi della memoria principale è stata l'opportunità di sviluppo perfetta per un sistema che si basa su una lettura singola da HDFS e il mantenimento dei dati letti nella memoria principale. Questo tipo di implementazione è possibile grazie alla prima astrazione introdotta da Spark: la

[1]Il fenomeno dell'overfitting si verifica quando il numero di parametri del modello considerato è eccessivo rispetto al numero di campioni. In questa situazione il modello tende a considerare caratteristiche estremamente specifiche dei dati di training che non si riflettono poi all'interno dell'insieme di test, portando quindi ad un peggioramento delle performance.

rappresentazione dei dati sotto forma di *Resilient Distributed Data sets (RDDs)*. Un RDD può contenere qualsiasi tipo di oggetto e modella una collezione di dati suddivisa in partizioni logiche distribuite sui vari nodi che compongono il cluster agevolando quindi la computazione parallela. Un RDD può essere creato a partire da un oggetto esterno (come un file) o parallelizzando una collezione interna e, in caso di problemi, vengono automaticamente ricreati garantendo la robustezza del sistema.
La flessibilità di Spark è dovuta alla possibilità di scrivere applicazioni sfruttando diversi linguaggi di programmazione tra cui Scala, Java e Python. Un programma Spark si compone di una serie di operazioni effettuate sugli RDDs. Le operazioni possibili sono di due tipi: le trasformazioni che possono essere considerate operazioni intermedie come filtri o join e le azioni che sono operazioni finali posizionate al termine della pipeline di elaborazione come count oppure save. Le operazioni in Apache Spark vengono eseguite in modalità *Lazy evaluation*: viene costruito un grafo delle attività da svolgere ma nessuna di queste viene effettivamente lanciata fino a quando non si incontra una operazione di tipo *Action*. Grazie a questo accorgimento le operazioni possono essere riordinate e ottimizzate in maniera automatica.
Apache Spark è composto da cinque elementi fondamentali, rappresentati in Figura 4.3.

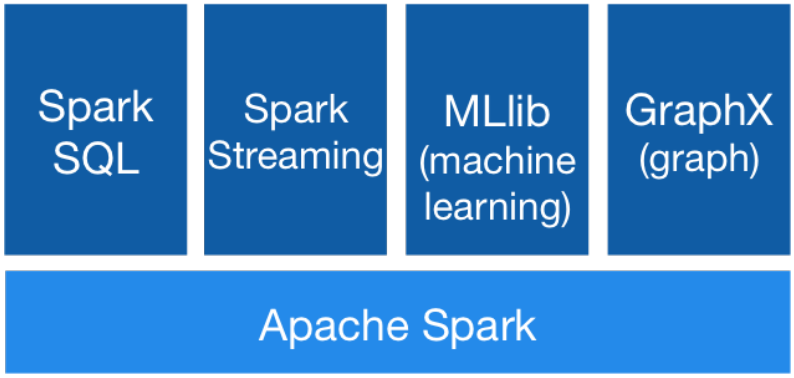

Figura 4.3. Componenti di Apache Spark - Spark documentation

- Spark Core: è il motore di base per l'elaborazione. Si occupa di pianificare le attività, distribuire e monitorare i job all'interno del cluster, fornire le API per la creazione degli RDDs e gestire la memoria. Questo componente viene sfruttato da tutti gli altri componenti di elaborazione di più alto livello.

- Spark SQL: permette a Spark di operare su dati strutturati attraverso interrogazioni in Hive Query Language (HQL), una variante del SQL tradizionale.

Attraverso questo componente è quindi semplice integrare interrogazioni standard a manipolazioni complesse sfruttando le operazioni sugli RDDs messe a disposizione dal framework.

- Spark Streaming: consente l'elaborazione di flussi in tempo reale attraverso API molto vicine a quelle usate per le operazioni standard di tipo batch semplificando il lavoro dei programmatori dal punto di vista dell'apprendimento.
- MLib: libreria integrata per parallelizzare alcuni algoritmi di machine learning e data mining.
- Graphx: fornisce molti algoritmi per la manipolazione di grafici in modalità parallela.

Sebbene sia possibile eseguire una applicazione Spark in locale, utilizzando i thread per parallelizzare le operazioni, la configurazione tipica è di tipo distribuito all'interno di un cluster. Nella Figura 4.4 è rappresentata la struttura tipica e i componenti necessari per una corretta esecuzione. L'architettura di base è di tipo Master-slave, dove il master è rappresentato dal *Driver* che contiene il metodo principale che definisce il flusso di operazioni e accede a Spark attraverso l'oggetto SparkContext, responsabile del coordinamento degli *Executors* che sono ospitati dai nodi Worker e si occupano di eseguire le operazioni sugli RDDs. Lo Spark Context si connette ad un *Cluster Manager* che alloca le risorse necessarie e successivamente definisce gli esecutori sui nodi del cluster ai quali invia prima il codice dell'applicazione e dopo le singole attività da eseguire. Queste attività risultano essere indipendenti rispetto al tipo di cluster manager scelto fin tanto che sia possibile acquisire gli esecutori e mettere in piedi il meccanismo di comunicazione.

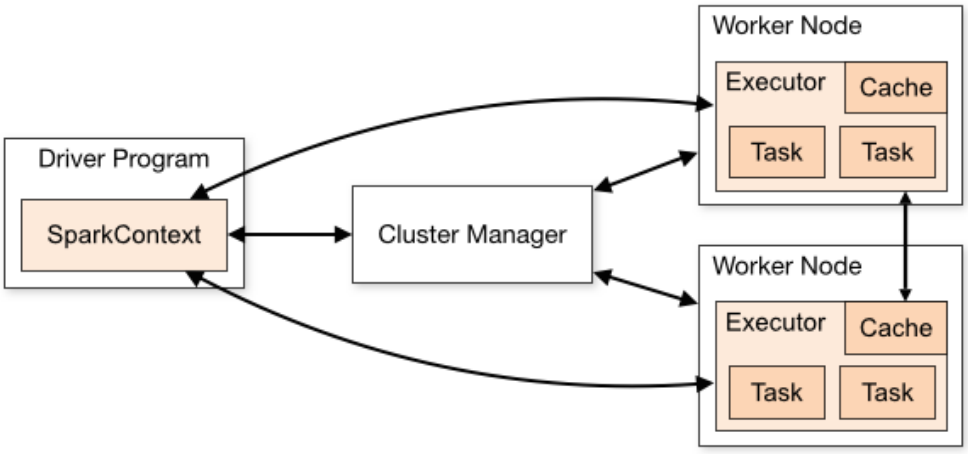

Figura 4.4. Struttura distribuita di Apache Spark - Spark documentation

4.3 Algoritmi utilizzati

Per migliorare il processo di Data Quality presente all'interno della soluzione aziendale sono state sviluppate due diverse metodologie che sfruttano algoritmi diversi: *Isolation Forest (iForest)* è stato utilizzato per la ricerca delle anomalie sui dati con elevato numero di dimensioni, mentre il *Local Outlier Factor (LOF)* supporta la strategia di outlier detection per l'analisi mirata su particolari dimensioni di spiccato interesse. Di seguito verranno presentati nel dettaglio tali algoritmi, il relativo funzionamento e i parametri caratterizzanti.

4.3.1 Isolation Forest

La caratteristica principale di questo algoritmo è data dal fatto che si pone come obiettivo quello di trovare esplicitamente le anomalie all'interno dei dati a differenza della maggior parte degli altri metodi di outlier detection, che invece basano il loro funzionamento sulla ricerca della distribuzione dei punti validi. L' *iForest* è un metodo di ensemble basato sulla combinazione di un gruppo di *isolation trees*.
In ogni albero, i dati vengono partizionati lungo l'asse di un attributo scelto casualmente, utilizzando un punto di partizione deciso in maniera altrettanto casuale. L'obiettivo è quello di dividere i dati in nodi che contengono ad ogni passo un numero sempre minore di punti, fino ad arrivare ad una situazione di completo isolamento, dove ogni nodo contiene un solo punto. Considerando che gli outlier si vanno a posizionare in regioni scarsamente popolate, questi possono essere riconosciuti analizzando la profondità di ogni ramo che si va a formare: punti particolarmente distanti dalla normale distribuzione verranno isolati facilmente e saranno quindi posizionati in nodi molto vicini alla radice. Un esempio dei partizionamenti necessari per isolare un punto normale e un punto anomalo è rappresentato in Figura 4.5.

Il numero di divisioni necessarie per ottenere l'isolamento viene usato per determinare il punteggio di anomalia. Questo processo viene ripetuto fino ad ottenere un livello di isolamento per ogni punto e, una volta terminato, viene restituita la probabilità di ogni punto di essere non valido.
Ogni albero è formato da due tipi di nodi: interni ed esterni. I nodi interni sono quelli intermedi generati ogni volta che viene scelto un valore di split. Sono necessariamente genitori di due sotto-alberi, quello a sinistra che contiene tutti i valori più piccoli del peso di partizionamento scelto e quello a destra che contiene invece tutti i valori maggiori, creando quindi un albero binario valido. I nodi esterni sono invece i nodi foglia che non possono essere ulteriormente suddivisi, fondamentali nel calcolo del punteggio di anomalia [21].
Il primo passo per creare un isolation tree a partire da un set di N punti è quello di creare il nodo radice che contiene tutti i punti. In questa fase, il nodo radice è anche l'unico nodo presente nella lista di nodi candidati C per il partizionamento.

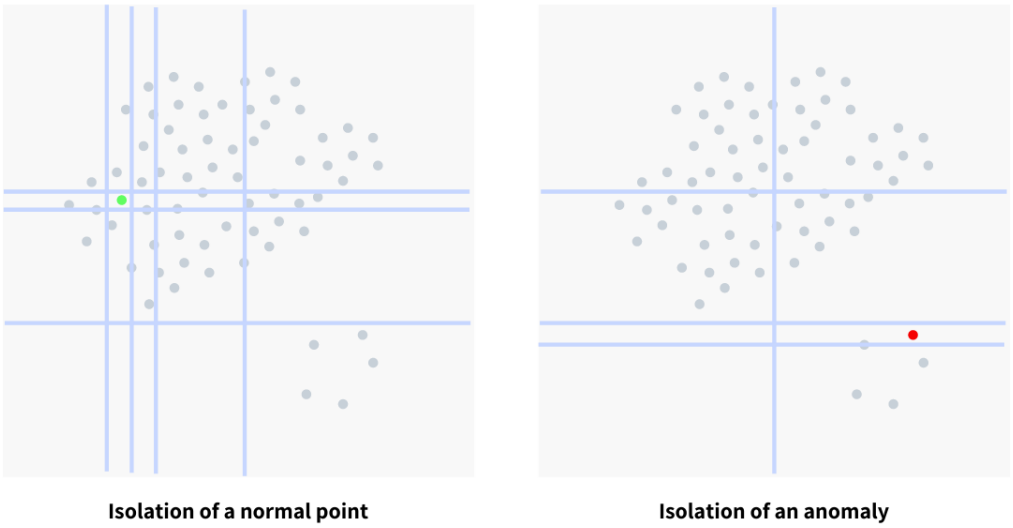

Figura 4.5. Esempio di isolamento di un punto normale (a sinistra) e di un outlier (nella parte destra) - medium.com

A questo punto l'isolation tree T viene costruito ripetendo un insieme di operazioni fino a quando la lista dei candidati risulti vuota.
La prima operazione consiste nello scegliere in maniera casuale un nodo N dalla lista dei candidati e rimuoverlo dalla stessa. I dati contenuti in R vengono divisi in due partizioni, R_1 ed R_2, sulla base di un attributo random e scegliendo un valore casuale (v) tra il minimo e il massimo. La suddivisione dei dati avviene nel seguente modo: se il punto x considerato risulta minore o uguale a v, x viene inserito in R_1; se x invece ha valore maggiore di v, entra a far parte dell'insieme R_2. Costruiti gli insiemi R_i, si procede al loro inserimento nella lista dei candidati C se contengono più di un punto o viceversa vengono etichettati come nodi foglia.
L'albero binario che viene generato è tipicamente non bilanciato dove il percorso dalla radice ai nodi foglia rappresenta il sotto spazio di isolamento con i relativi attributi utilizzati. I punti anomali nella maggior parte dei casi possono essere isolati in spazi con numero di dimensioni molto ridotto rispetto ai dati normali.
Definiti gli isolation trees, l'ultimo passo è combinare il risultato ottenuto da tali alberi attraverso la media della lunghezza dei percorsi in ogni albero per ogni punto. La Figura 4.7 presenta un possibile risultato per la costruzione di una foresta.
Il metodo di Isolation Forest è strettamente legato alle tecniche di outlier detection negli spazi sottodimensionati. Ogni ramo che viene costruito rappresenta un sotto spazio locale nell'insieme dei dati che dipende da quali attributi vengono usati per il partizionamento. Percorsi brevi corrispondono ad un numero minore di dimensioni e, di conseguenza, ad un punteggio di anomalia superiore.

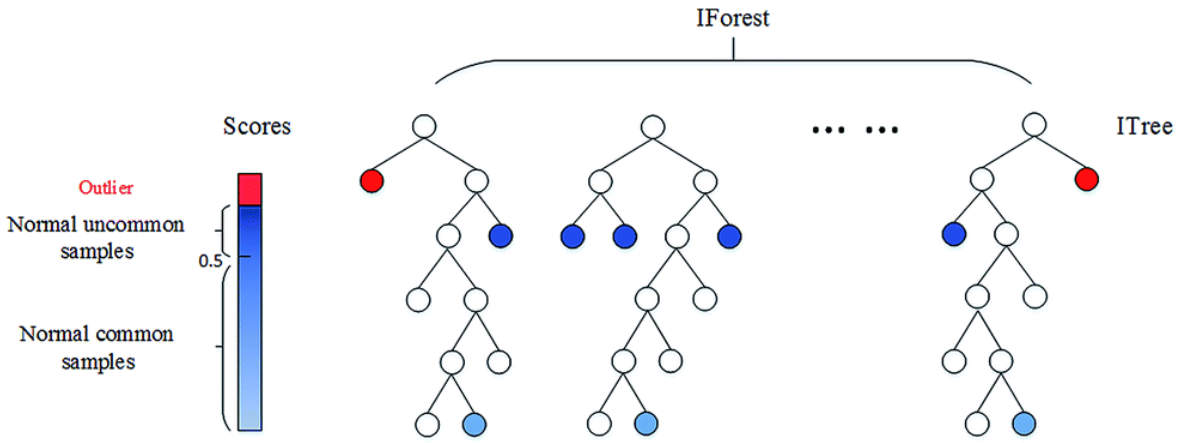

Figura 4.6. Isolation Forest - RSC Publishing

Come in ogni meccanismo di identificazione degli outlier, l'output ottenuto consiste in un punteggio che quantifica, per ogni punto, la probabilità che questo sia o meno un dato anomalo. Tale valore deve essere limitato e confrontabile. Nel caso dell'isolation forest il punteggio si ottiene come funzione della lunghezza del percorso in cui si trova il dato che viene definito come il numero di archi attraversati a partire dalla radice fino ad arrivare al nodo terminale e verrà identificato di seguito con h(x). Il punteggio di anomalia per un dato x è definito dalla formula 4.3

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{4.3}$$

dove n è il numero di nodi esterni, h(x) è come già detto la lunghezza del percorso per il dato considerato e c(n) rappresenta la lunghezza di percorso medio per ricerche fallimentari in un Binary Search Tree, espresso in forma estesa nella formula 4.4 in cui H(i) è il numero armonico.

$$c(n) = 2H_{n-1} - \frac{2(n-1)}{n} \tag{4.4}$$

La versione base dell'algoritmo di isolation tree, quando non si applica nessun meccanismo di terminazione anticipata, è priva di parametri e mantiene bassa complessità. Questo rappresenta un vantaggio non indifferente per problemi tipicamente non supervisionati come quello della rilevazione delle anomalie. L'ensemble che costruisce l' isolation forest giova di questi vantaggi e risulta particolarmente efficiente con buone garanzie di ottenere risultati soddisfacenti. In [22] viene approfondita la differenza in termini prestazionali tra l'algoritmo di isolation forest e le altre tecniche più comuni per risolvere il problema di outlier detection, sia nel caso di numero limitato di dimensioni sia all'aumentare delle stesse.
Tra i principali parametri dell'algoritmo dal punto di vista implementativo, troviamo [23]:

- Il numero di stimatori da usare per l'ensemble
- Il numero di campioni da estrarre per la fase di train di ogni stimatore
- Il grado di contaminazione che rappresenta la percentuale di anomalie all'interno dell'insieme dei dati
- Il numero massimo di attributi da considerare durante la fase di training.

La principale motivazione che ha portato all'utilizzo di questo algoritmo è la capacità di gestire in maniera abbastanza agevole ed efficiente dati con elevato numero di dimensioni. L'obiettivo è verificare se esiste la possibilità di ottenere risultati soddisfacenti senza applicare particolari tecniche di pre-processing sui dati prima della fase di Data Quality risparmiando quindi le tempistiche necessarie ad esse e gli eventuali problemi nella selezione delle informazioni rilevanti. Il comportamento unico dell'algoritmo che si pone l'obiettivo di isolare le anomalie senza definire la distribuzione normale dei dati forniti, rappresenta un ulteriore spinta per il suo utilizzo dato che dimostra come il funzionamento dello stesso risulti indipendente rispetto alle possibili ripartizioni delle informazioni.

4.3.2 Outlier Detection con Local Outlier Factor

Il Local outlier factor quantifica il punteggio di anomalia di un dato misurando la differenza di densità tra il dato stesso e i suoi vicini, analizzando dunque un'area locale piuttosto che globale. Il concetto di località è espresso attraverso il confronto con i suoi *k-neighbor*, ovvero i k valori più vicini al dato considerato. Più un punto risulta isolato dai suoi vicini, maggiore sarà il punteggio di anomalia che gli verrà attribuito.
Dato un punto x, la prima misura che deve essere introdotta per chiarire il funzionamento di questo metodo è la *k-distance* che rappresenta la distanza tra x e il suo k-esimo vicino: fissato k = 5, ad esempio, la k-distance è la distanza tra x e il quinto punto per ordine di vicinanza da esso. Questa misura viene successivamente utilizzata per il calcolo di un ulteriore valore, fondamentale per l'algoritmo: la *reachability-distance* definita nella formula 4.5.

$$reach\_dist(x, y) = max(k\_distance(y), dist(x, y)) \tag{4.5}$$

In sostanza questa grandezza rappresenta il massimo tra la distanza di due punti e la k-distance del secondo punto. In questo modo quando la distanza tra i due punti X e Y è considerevole, la reachability distance tra i due punti non è altro che la reale distanza tra gli stessi. Quando invece X si trova nel vicinato di Y, la grandezza presa in esame è la k-distance del punto Y. Bisogna notare che questa definizione di distanza non è simmetrica. All'aumentare del valore di k, punti diversi si ritroveranno una reachability distance simile.

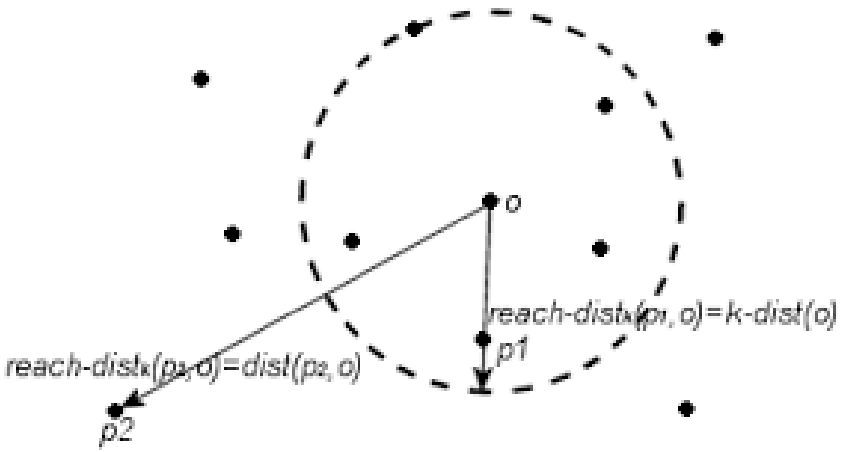

Figura 4.7. Reachability distance - citeseerx.edu

La formula 4.6 esprime la reachability distance media per il punto x, definita come la media delle reachability distances di tutti i punti nel vicinato di x $V_k(x)$.

$$ARD_k(x) = AVG_{y \in V_k(x)}\, reach\_dist(x, y) \tag{4.6}$$

Infine può essere determinato il valore del Local Outlier Factor per il punto x attraverso la formula 4.7.

$$LOF_k(x) = AVG_{y \in V_k(x)}\, \frac{ARD_k(x)}{ARD_k(y)} \tag{4.7}$$

Questa grandezza può essere vista anche sotto un ulteriore punto di vista considerando la reachability distance normalizzata per ogni punto, dove il fattore di normalizzazione è dato dalla media armonica (HMEAN nella 4.8) delle distanze del vicinato del punto considerato. Tale visione alternativa è espressa dalla formula 4.8.

$$LOF_k(x) = \frac{ARD_k(x)}{HMEAN_{y \in V_k(x)}\, ARD_k(y)} \tag{4.8}$$

L'uso di questa forma fa emergere potenziali problemi riscontrati dall'algoritmo: la media armonica potrebbe risultare pari a 0 per determinati insiemi di dati, in particolare in presenza di punti duplicati o estremamente vicini all'interno di una zona ad elevata densità. In questo caso tutti i punti nel vicinato dei dati duplicati rischiano di vedersi assegnato un punteggio di anomalia pari a ∞. Il problema può essere mitigato utilizzando medie di diverso tipo come ad esempio la media aritmetica che, tra l'altro, non va a cambiare in modo radicale l'interpretazione della grandezza.
Ulteriore possibilità è quella di eliminare i valori duplicati ma questa strategia

può essere adoperata solo se tali duplicati rappresentino del rumore e non punti significativi della distribuzione dei dati. Uno degli approcci più utilizzato è quello di aggiungere all'interno dell'equazione 4.8 un parametro di regolarizzazione α, sufficientemente piccolo e maggiore di 0, introdotto sia al numeratore che al denominatore ottenendo la 4.9.

$$LOF_k(x) = \frac{\alpha + ARD_k(x)}{\alpha + HMEAN_{y \in V_k(x)}\, ARD_k(y)} \tag{4.9}$$

Il metodo di Local outlier factor è stato presentato come un metodo basato sulla densità ma può essere compreso anche attraverso un approccio basato sulle distanze relative. La connessione tra questo meccanismo è la densità è data dalla capacità del LOF di adattare il proprio comportamento in base alla regione in cui viene analizzato sfruttando proprio il concetto delle distanze relative.
I parametri più significativi [24] comprendono:

- Dimensione del vicinato: rappresenta il valore di k che deve essere usato dall'algoritmo.
- Metrica: definisce la modalità con cui calcolare la distanza tra i punti.
- Il grado di contaminazione che rappresenta la percentuale di anomalie all'interno dell'insieme dei dati

Il punto di maggior interesse e che ha grande impatto sulla qualità del risultato ottenuto è il numero di vicini da considerare per ogni punto. Tale valore dovrebbe essere impostato in modo da risultare maggiore del numero di campioni appartenenti al cluster più piccolo in modo da esprimere al meglio il concetto di località, ma allo stesso tempo minore del massimo numero di punti vicini che rappresentano anomalie a livello locale. La complicazione è data dal fatto che tali informazioni non sono note nella maggior parte dei casi e il parametro k diventa dunque soggetto a tutti i meccanismi di ricerca del valore ottimale.
Nella pratica si verifica che la scelta di un valore di k troppo piccolo porta l'algoritmo ad essere estremamente localizzato e risulta particolarmente sensibile al rumore. Viceversa, con un valore di k troppo elevato, il LOF perde il suo concetto di località e outliers locali potrebbero non essere individuati.
Ulteriore fattore che influisce tra gli svantaggi del metodo è la difficoltà di interpretazione: l'identificazione di un outlier risulta particolarmente dipendente dal problema che si sta affrontando e il valore corretto da utilizzare come soglia per separare le anomalie dagli altri valori è estremamente variabile.
Nonostante questi svantaggi l'algoritmo è stato scelto per la risoluzione del problema di data quality affrontato per la sua capacità di adattarsi bene alla presenza di cluster diversi e con densità variabile all'interno dei dati, situazione che si è presentata ripetute volte nella fase di analisi e visualizzazione dei dati. Questo principio di località ha il vantaggio di poter valutare correttamente informazioni appartenenti alla stessa area di interesse ma aventi proprietà con diversi range di distribuzione.

4.4 Ambiente di sviluppo

Tutti i servizi utilizzati nell'architettura in figura 4.1 sono basati sulla tecnologia messa a disposizione da *Microsoft Azure* [25]. Microsoft Azure rappresenta la piattaforma di *cloud computing* pubblica di Microsoft ed offre più di 200 prodotti e servizi utili per la realizzazione di soluzioni innovative. I servizi di cloud computing pubblico sono nati nel 2006 grazie ad Amazon e ad oggi sono uno dei settori di maggior rilievo nel mondo aziendale: sono sempre di più le organizzazioni che migrano su questo tipo di tecnologia per giovare di tutti i vantaggi che ne derivano. Nel cloud pubblico le infrastrutture che vengono messe a disposizione dei vari utilizzatori sono di proprietà di un fornitore esterno (come appunto Amazon e nel nostro caso Microsoft) e sono dislocate in diversi punti a livello mondiale. L' utilizzo di queste risorse è condiviso da vari utenti nel rispetto dei massimi livelli di sicurezza e isolamento, a seguito di una sottoscrizione economica. Le principali motivazioni che hanno permesso la crescita esponenziale di questo tipo di architettura comprendono:

- Elasticità: la possibilità di scalare in modalità orizzontale piuttosto che verticale è uno dei principali vantaggi del mondo cloud. Quando si registra una certa sofferenza nella fruizione del servizio, si procede con l'aggiunta di ulteriori istanze dello stesso, eventualmente su nuova macchina che va ad affiancare quelle già presenti, invece di aggiornare le risorse proprie del dispositivo a disposizione (ad esempio aggiungere fisicamente un banco di memoria RAM). Questa soluzione permette di adattarsi in maniera dinamica al carico di lavoro con la possibilità di ottenere una velocità di risposta molto maggiore rispetto ad effettuare le operazioni necessarie per scalare verticalmente, che spesso richiedono dei tempi di sospensione del servizio. Ulteriore risparmio si ha nella fase iniziale di analisi dove si dovrebbero stimare le risorse da dedicare al processo, operazione mai banale e piuttosto critica nel caso di errori.

- Servizi sempre attivi: il provider del servizio cloud è incaricato di mantenere le applicazioni costantemente attive e raggiungibili in qualsiasi momento. Spesso quando si crea un processo distribuito in questo contesto, con numerose istanze dell'applicazione in esecuzione contemporaneamente, è il fornitore stesso a garantire un meccanismo di *load balancing* per evitare carichi eccessivi sulla singola macchina.

- Separazione delle responsabilità: la gestione dell'infrastruttura viene lasciata nelle mani del fornitore scelto che ha a disposizione tutto il livello di esperienza necessario per gestire nel modo migliore tutte le complicazioni che ne derivano, come il raffreddamento delle stanze, predisporre sistemi di backup di energia elettrica e connessione ad internet e così via. Questo consente alle singole organizzazioni di non preoccuparsi di problematiche che esulano dal proprio

dominio e dove spesso non si hanno sufficienti competenze, con conseguente risparmi di tempo e denaro.

- Risorse illimitate: è praticamente impossibile terminare le risorse a disposizione del provider e le richieste di incremento verranno sempre soddisfatte.

- Aggiornamento automatico: tutte le strutture che si trovano sotto il controllo del fornitore dei servizi cloud, che variano a seconda del modello che si adotta, vengono aggiornate in maniera automatica eliminando l'effort e le tempistiche necessarie.

- Servizi geo-localizzati: i datacenters a disposizione sono dislocati in molte aree geografiche permettendo di avvicinare il servizio offerto all'utente e migliorare le prestazioni dal punto di vista della latenza ed efficienza di rete.

- Semplifica la collaborazione: attraverso i servizi cloud la collaborazione di team situati in località remote lontane tra loro risulta più efficiente e sicura.

- Aumento dell'affidabilità dei dati: i sistemi di ridondanza dei dati sono sotto la responsabilità del fornitore dei servizi cloud.

- Vantaggi economici: rappresenta probabilmente il fattore principale che porta le aziende a scegliere le soluzioni cloud. Questi vantaggi comprendono diversi aspetti tra cui particolare rilievo ha il risparmio sull'energia elettrica necessaria per le infrastrutture che rappresenta un costo sorprendentemente alto per le organizzazioni e la possibilità di pagare soltanto le risorse che si stanno effettivamente utilizzando (pay-per-use). Ulteriore taglio si evidenzia nei costi capitali necessari per l'acquisto dell'hardware necessario e per i costi di installazione e relativo training dei dipendenti.

Microsoft Azure mette a disposizione diverse categorie di servizi e in base alla modalità con cui vengono erogati si parla di *Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS)* e *Software-as-a-Service (SaaS).*
Il modello IaaS prevede la massima flessibilità dato che il cloud provider è in possesso dell'infrastruttura che comprende la rete, l'archiviazione e le risorse computazionali, messa a disposizione dell'utente attraverso un'astrazione che garantisce la creazione ed installazione di qualsiasi servizio. L'utente può quindi agire come se avesse pieno controllo sulle risorse ma al contempo viene esonerato da tutte le attività di installazione e manutenzione dal punto di vista dell'infrastruttura fisica. D'altro canto, le responsabilità che rimangono sotto il profilo dell'utente sono elevate: un esempio si può avere pensando alla gestione del traffico interno e alla possibile necessità di bilanciare il traffico su dispositivi diversi.
Nel PaaS vengono messe a disposizione dell'utente una serie di API e servizi base per la realizzazione dell'applicazione specifica che si vuole portare sul mercato. In

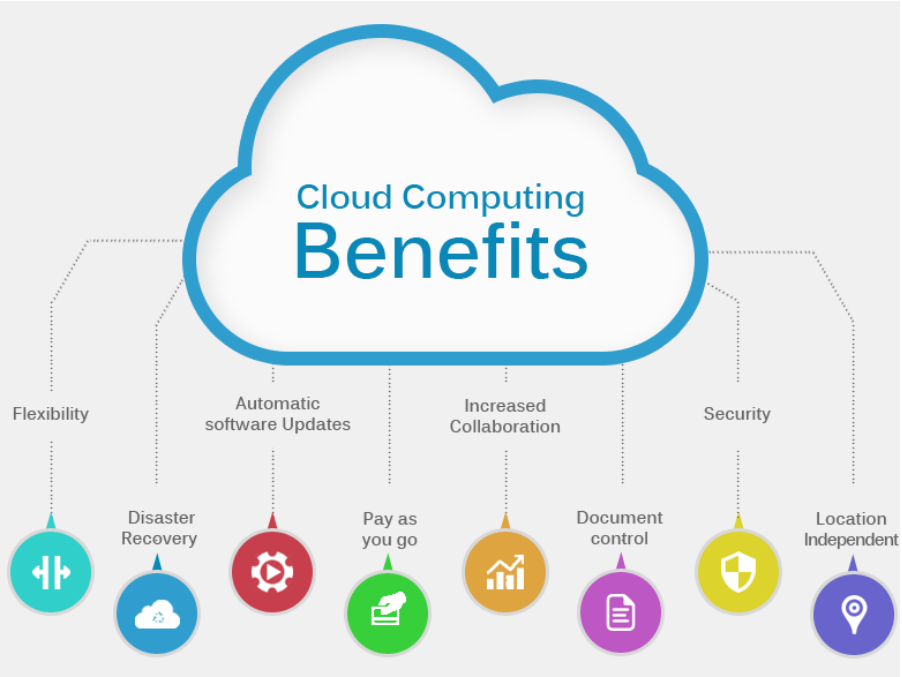

Figura 4.8. Vantaggi del cloud computing - blog.sarv.com

questo caso molte delle responsabilità che nel modello di PaaS erano sotto il controllo dell'utente vengono gestite dal fornitore dei servizi che nasconde i dettagli di risorse come il sistema operativo, l'infrastruttura di rete ed il relativo meccanismo dedicato alla sicurezza.
Infine, il livello di SaaS può essere considerato come la modalità di fruizione di più alto livello dove il provider mette a disposizione dell'utente l'intera applicazione con tutte le funzionalità sviluppate e l'utente può utilizzarla senza dover scrivere neanche una linea di codice. La manutenzione del software è completamente a carico del fornitore dei servizi cloud e il compito dell'utente è ridotto al semplice utilizzo.
Per quanto riguarda gli svantaggi delle soluzioni cloud, il primo fattore a cui si pensa è la necessità di avere una connessione di rete stabile con banda adeguata per poter interagire in maniera efficace con il sistema e accedere a dati o servizi. Utilizzare un servizio remoto può essere problematico per le applicazioni real-time che necessitano di una latenza particolarmente bassa per il corretto funzionamento. Ulteriore settore non completamente ottimizzato nel mondo cloud riguarda l'insieme di applicazioni che generano enorme quantità di dati in quanto risulta sconveniente il trasferimento di tali quantità attraverso la rete.
Il servizio di cloud pubblico introduce inoltre ulteriore riflessione sul corretto utilizzo dei dati. Nella maggioranza dei casi non è dato sapere cosa viene fatto effettivamente con i dati memorizzati da parte del provider: non è possibile avere la certezza

che le informazioni non vengano lette dal provider stesso o da terza parti. Tale problematica potrebbe portare organizzazioni che fanno largo uso di dati particolarmente sensibili ad allontanarsi da questa tecnologia.
Oltre agli svantaggi propri del mondo cloud, bisogna valutare gli inconvenienti tipici del modello che si va ad utilizzare. Nella soluzione di Software-as-a-Service, ad esempio, l'utente non ha nessuna possibilità di personalizzazione del servizio che si va ad utilizzare e funzionalità indispensabili non fornite, potrebbero portare ad una rivalutazione del meccanismo usato. Nel Platform-as-a-Service invece si è fatto riferimento all'utilizzo di particolari API per accedere alle risorse e realizzare le funzionalità richieste. Questa astrazione lega indissolubilmente l'applicazione all'ambiente in cui viene realizzata in quanto le API variano da un provider all'altro e impediscono di creare applicazioni facilmente portabili su infrastrutture diverse.
Di seguito sono elencati i principali servizi messi a disposizione da Microsoft Azure che sono stati utilizzati durante il lavoro.

4.4.1 Azure Data Factory

La piattaforma Azure Data Factory è il servizio di estrazione, trasformazione e caricamento di dati attraverso cui è possibile creare delle pipeline di lavoro per il trasferimento e la manipolazione di informazioni su larga scala [26]. Si integra con servizi come *Databricks* per la trasformazione dei dati e servizi di analytics per l'archiviazione e l'utilizzo di business intelligence.
Data Factory semplifica il processo di raccolta ed integrazione di tutte le sorgenti dei dati e consente di spostare le informazioni in maniera centralizzata per favorire eventualmente le operazioni di manipolazione successive. L'attività di copia implementata nel servizio permette di evitare la creazione di sistemi personalizzati per il trasferimento dei dati che risultano spesso costosi e complessi.
All'interno della pipeline sono inoltre presenti dei componenti di base che permettono di eseguire trasformazioni standard sui dati attraverso l'uso di Apache Spark senza però andare a modificare il codice. Se necessario tali servizi possono essere integrati attraverso funzioni personalizzate.
Il supporto alle attività di Continuous Integration e Continuous Delivery e la standardizzazione delle tecniche di monitoraggio semplificano ulteriormente lo sviluppo e la distribuzione dei processi creati.
Il concetto base all'interno di questo servizio di Microsoft Azure è quello di pipeline. Una pipeline è un insieme logico di attività per il completamento di una operazione. Attraverso questa visione si ha il vantaggio di gestire le attività come insieme piuttosto che singolarmente. Le singole attività rappresentano uno step di elaborazione all'interno della pipeline che può essere di tre tipi: attività di trasferimento, di trasformazione e di controllo.
L'attività di integrazione di diverse fonti di dati viene eseguita grazie alla definizione dei servizi collegati. Questi possono essere interpretati come stringhe di

connessione e permettono a Data Factory di accedere a risorse esterne.
Al fine di rendere maggiormente dinamica l'esecuzione di una pipeline, vengono utilizzati i parametri, ovvero delle coppie chiave-valore che permettono di aggiungere informazioni di configurazione o specificare come input di un'attività un valore ottenuto dall'esecuzione di una ulteriore attività terminata. I parametri a livello di pipeline così come la ramificazione della stessa, sono gestite dal *flusso di controllo* che include anche il passaggio di stati personalizzati e l'esecuzione di cicli iterativi.

4.4.2 Azure Databricks

"Azure Databricks è un servizio di analisi veloce, semplice e collaborativo basato su Apache Spark" [27]. All'interno della pipeline di lavoro, Databricks viene usato per prelevare i dati dalle varie origini messe a disposizione da Azure ed applicare trasformazioni per ricavare informazioni significative. Questo servizio è stato usato per realizzare le operazioni di aggregazione e di Data Quality rappresentate nella figura 4.1.
All'interno di Databricks sono inclusi tutti i principali componenti di Apache Spark presentati nella figura 4.3 ed è possibile gestire a costo zero un intero cluster Spark sicuro e affidabile insieme ad un'area interattiva per la visualizzazione dei dati e creazione di prototipi.
L'accesso alle risorse è garantito dall'area di lavoro di Databricks che organizza

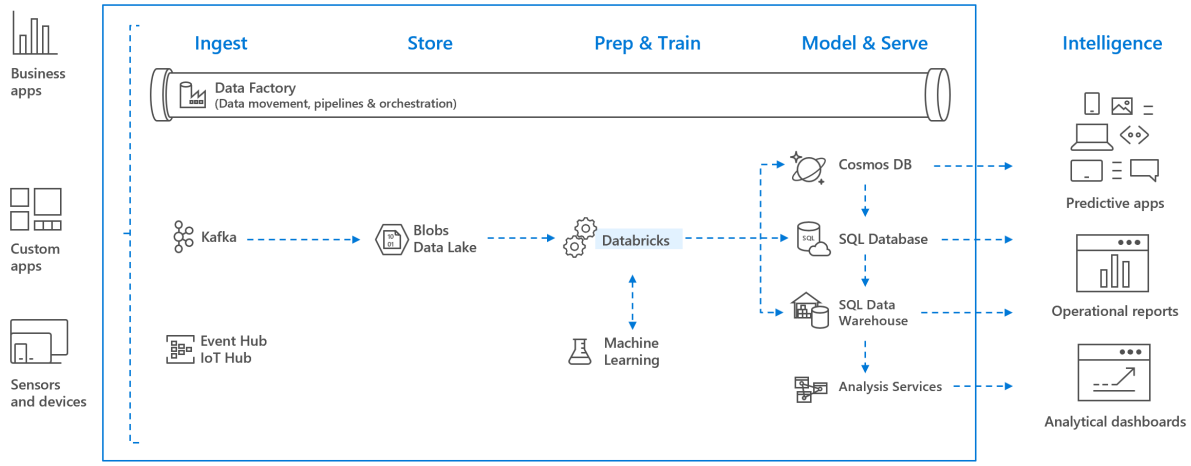

Figura 4.9. Esempio di pipeline di lavoro di Azure con rappresentazione dei servizi nelle varie fasi - Microsoft Documentation

i contenuti in cartelle. Gli elementi principali che compongono le risorse sono i notebook, che contengono i comandi eseguibili e le dashboard utili per la visualizzazione. I linguaggi supportati dai notebook sono molteplici (tra cui Scala, Python

e SQL) e possono anche essere usati in maniera combinata all'interno dello stesso foglio di lavoro.
All'interno del servizio ci sono diversi oggetti che contengono i dati su cui eseguire le analisi:

- File System di Databricks (DBFS): rappresenta una astrazione del file system tradizionale mantenuto in un archivio BLOB.
- Database
- Tabella
- Metastore: contiene le informazioni su struttura e partizione di tabelle e data warehouse così come i serializzatori di cui si ha bisogno per leggere e scrivere i dati.

A seconda dei componenti che risultano indispensabili nella realizzazione del processo che si vuole costruire, è possibile utilizzare una *Runtime di databricks* specifica e appositamente creata per ottimizzare e semplificare la gestione delle risorse necessarie.

4.4.3 Azure Data Lake

Azure Data Lake rappresenta un repository che permette di mantenere in un'unica posizione tutti i dati aziendali, a prescindere dalla loro dimensione, dal tipo o dalla velocità di inserimento [28]. Questo servizio fornisce una gestione sicura, affidabile ed ottimizzata per ottenere buone performance nella fase di analisi.
In termini di storage non è presente alcuna limitazione per quanto riguarda la dimensione degli account, dei file o la quantità degli stessi. L'affidabilità è garantita attraverso la creazione di più repliche per le quali non esiste alcun limite di durata del periodo di archiviazione. I dati possono essere memorizzati in forma nativa senza l'obbligo di effettuare trasformazioni preliminari sugli stessi o definire uno schema a priori prima del caricamento effettivo. L'interpretazione delle informazioni ricevute è lasciata completamente al framework di elaborazione.
L'efficienza del processo di analisi è garantita dal sistema di propagazione dei file su un certo numero di singoli server di archiviazione permettendo una elevata velocità di lettura nelle operazioni parallele.

4.4.4 Azure Delta Lake

Delta Lake è uno strato di archiviazione posizionato sopra il Data Lake esistente che supporta le API Spark e permette di introdurre transazioni ACID e una manipolazione scalabile dei metadati [29]. Le ottimizzazioni implementate fanno in modo che le operazioni di ETL e le query personalizzate possano essere eseguite in

maniera performante. I dati possono essere memorizzati in tabelle delta definendo uno schema ben preciso e gestendo in maniera automatica i tentativi di inserimento di dati non validi. La possibilità di applicare un sistema di controllo di versione permette di effettuare, ove necessario, operazioni di rollback, che unite al supporto delle operazioni di merge, upsert e delete semplificano la gestione di casi di utilizzo particolarmente complessi.

4.4.5 Azure Blob Strorage

L'archiviazione BLOB permette di memorizzare grandissime quantità di dati non strutturati [30]. Questo servizio è basato principalmente su tre concetti fondamentali: un account di archiviazione, un contenitore all'interno dell'account e l'oggetto di tipo BLOB mantenuto nel contenitore.
L'account di archiviazione crea uno spazio univoco per i nomi all'interno di Azure. Questo viene usato per definire l'endpoint che identifica l'archivio BLOB. All'interno di un account sono presenti uno o più contenitori. Questi possono essere paragonati a delle directory di un normale file system. Ogni contenitore può contenere un numero infinito di oggetti BLOB. Azure supporta tre diversi tipi di BLOB:

- BLOB in blocchi: specializzati per i dati di tipo testuale e binario. Ogni blocco può essere manipolato in maniera individuale.
- BLOB di aggiunta: anch'essi costituiti da blocchi ma specializzati per le operazioni di aggiunta.
- BLOB di pagine: permettono la memorizzazione di dati con modalità di accesso casuale.

A differenza di Azure Data Lake che è ottimizzato per analisi Big Data, il servizio di BLOB storage è pensato per una vasta gamma di scenari tra cui c'è anche l'analisi di Big Data. Ulteriore distinzione tra i due servizi è presente nel contesto delle limitazioni: per mantenere un livello di prestazioni accettabile, Azure BLOB storage fissa dei limiti sul numero di account di archiviazione e sulle rispettive capacità massime di memorizzazione, mentre Azure Data Lake non pone nessun vincolo in questi termini.
Per la gestione di dati e risorse, invece, i servizi di Blob e Data Lake risultano parecchio simili sia per quanto riguarda la sicurezza dei dati con i meccanismi di crittografia, autorizzazione e controllo degli accessi, sia per l'affidabilità e le API di accesso.

Capitolo 5

Risultati sperimentali

In questo capitolo si fornirà una descrizione dettagliata dei dati utilizzati, esplicitando il tipo e l'informazione aggiunta da ogni dimensione. Verranno inoltre presentati e analizzati i risultati ottenuti con gli algoritmi descritti in 4.3 per il miglioramento del processo di data quality richiesto, evidenziando eventuali punti di forza e debolezza e commentando possibili sviluppi futuri per incrementare ulteriormente l'efficacia dei metodi presentati.

5.1 Struttura dei dati analizzati

I dati che vengono ingeriti ed analizzati dal meccanismo di data quality implementato, contengono informazioni relative alle statistiche di veicoli, appartenenti a categorie diverse, durante la loro marcia. La parte più significativa degli attributi che vengono trasmessi è presentata nella tabella 5.1. Tali attributi sono ulteriormente arricchiti con informazioni di utilità generale ma che non rappresentano statistiche di rilievo e vengono escluse dall'analisi successiva. Tra queste troviamo la targa del veicolo per quanto riguarda la parte di identificazione, alcune informazioni temporali dove in particolare vengono memorizzati il tempo di inizio campionamento, il tempo di fine campionamento e il tempo di caricamento all'interno della base dati ed infine informazioni che permettono di identificare la corsa attraverso un identificativo di sessione, un codice di viaggio e un identificativo di missione.
Per alcune delle grandezze riportate in tabella 5.1 i valori raccolti si vanno a dividere ulteriormente in rilevazione minima, massima e media. Questo è il caso delle grandezze che rivestono un ruolo particolarmente importante ai fini delle statistiche e in generale dove questa suddivisione risulta sensata. Un esempio concreto si ha considerando la velocità di marcia e il consumo di carburante nel corso della missione esaminata.

Nome	**Unità di misura**	**Descrizione**
OdoStartValue	metri	Valore dell'odometro all'inizio della missione
OdoEndValue	metri	Valore dell'odometro alla fine della missione
BatteryCapacity	ampere-ora	Capacità residua della batteria
BrakePedalTime	secondi	Tempo di pressione del freno nella missione
TotalStartFuel	litri	Quantità di carburante totale all'inizio della missione
EngineOilLevel	litri	Livello di olio motore
VehicleWeight	chilogrammi	Peso del veicolo (comprende eventuale rimorchio)
CruiseControlTime	secondi	Tempo di utilizzo del cruise control
CruiseControlDistance	metri	Distanza percorsa con cruise control attivo
VehicleSpeed	chilometri-orari	Velocità di marcia del veicolo
FuelConsumption	litri/ora	Consumo di carburante misurato
Latitude	gradi	Latitudine del veicolo
Longitude	gradi	Longitudine del veicolo
EngineOilTemperature	gradi centigradi	Temperatura dell'olio rilevata
BatteryTemperature	gradi centigradi	Temperatura della batteria
ClutchTime	secondi	Tempo di pressione della frizione nella missione
AcceleratorTime	secondi	Tempo di pressione dell'acceleratore nella missione

Tabella 5.1. Dimensioni dei dati

5.2 Ricerca degli outlier nel caso di elevata dimensionalità

Il primo problema che si è affrontato riguarda il miglioramento dei processi di data quality sfruttando tutte (o buona parte) delle dimensioni caratterizzanti dei dati. In questo contesto si è utilizzato l'algoritmo di isolation forest descritto nel dettaglio in 4.3.1. Lo scopo finale è quello di analizzare l'impatto delle dimensioni 'secondarie' sulla classificazione dei punti e valutare la correttezza del metodo usato eliminando del tutto la fase di assegnazione di pesi ai vari attributi.
Al fine di ottenere un minimo riscontro visivo dei dati di input, la figura 5.1 presenta un sottoinsieme degli stessi all'interno di un sotto-spazio tridimensionale che lega il consumo minimo, la distanza percorsa e la durata della sessione di riferimento.

Nella figura 5.2 si possono osservare gli stessi dati dopo l'applicazione dell'al-

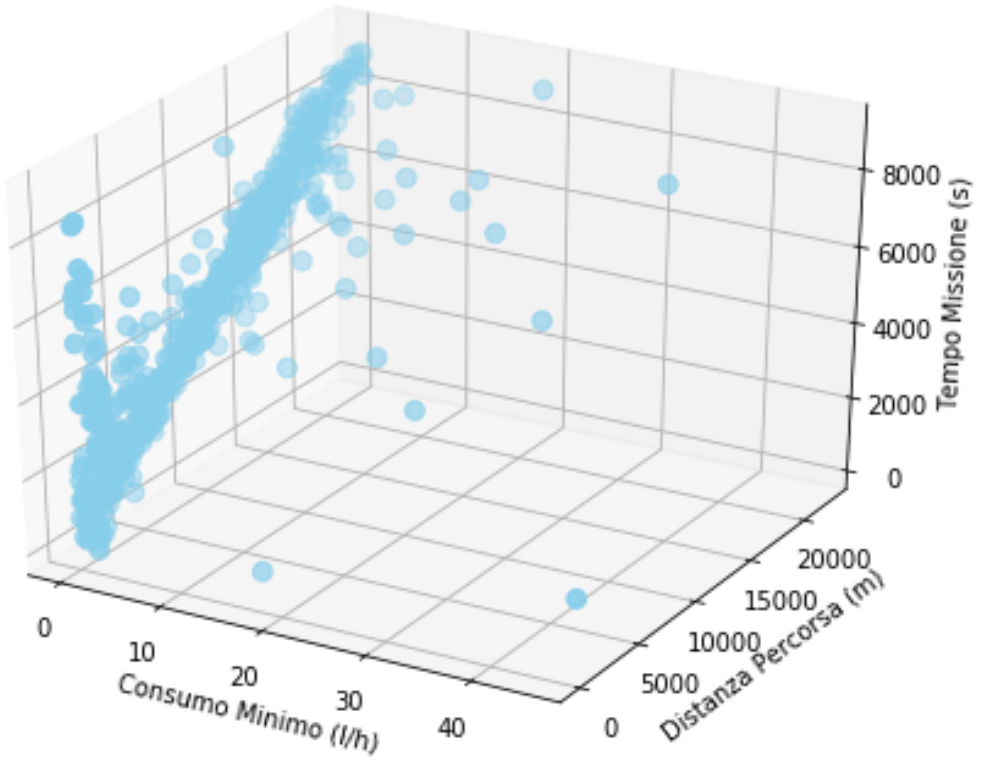

Figura 5.1. Visualizzazione dei dati in un sottospazio tridimensionale

goritmo. Tutti i punti che sono stati etichettati come outliers sono rappresentati in rosso. Concentrandosi sugli aspetti più pratici, è stato fornito all'algoritmo il valore di contaminazione come parametro. Il valore specificato deriva soprattutto

dall'esperienza che è stata maturata nel tempo circa il problema che si sta affrontando, supportato da ulteriori analisi che tengono in considerazione diversi fattori tra cui la precisione nominale dei sensori di misurazione. Alcuni dettagli sull'implementazione usata posso essere trovati in [23]. Uno dei vantaggi di questo metodo nella sua versione completa e senza limitazioni sul numero di dimensioni analizzate e sulla profondità degli alberi costruiti è dato dall'assenza di ulteriori parametri che influenzano la qualità del risultato. I restanti valori che è possibile specificare nell'implementazione utilizzata riguardano infatti esclusivamente la velocità di costruzione delle strutture necessarie e per i quali si è deciso di conservare le configurazioni automatiche o di default. Per quanto riguarda l'output è stato scelto di utilizzare la versione dell'algoritmo che restituisce direttamente la categoria finale piuttosto che il punteggio di anomalia che viene comunque salvato in apposite strutture in modo da ottenere un riscontro durante la validazione specifica dei risultati.

L'unico accorgimento che è stato preso come forma di pre-processing nella fase

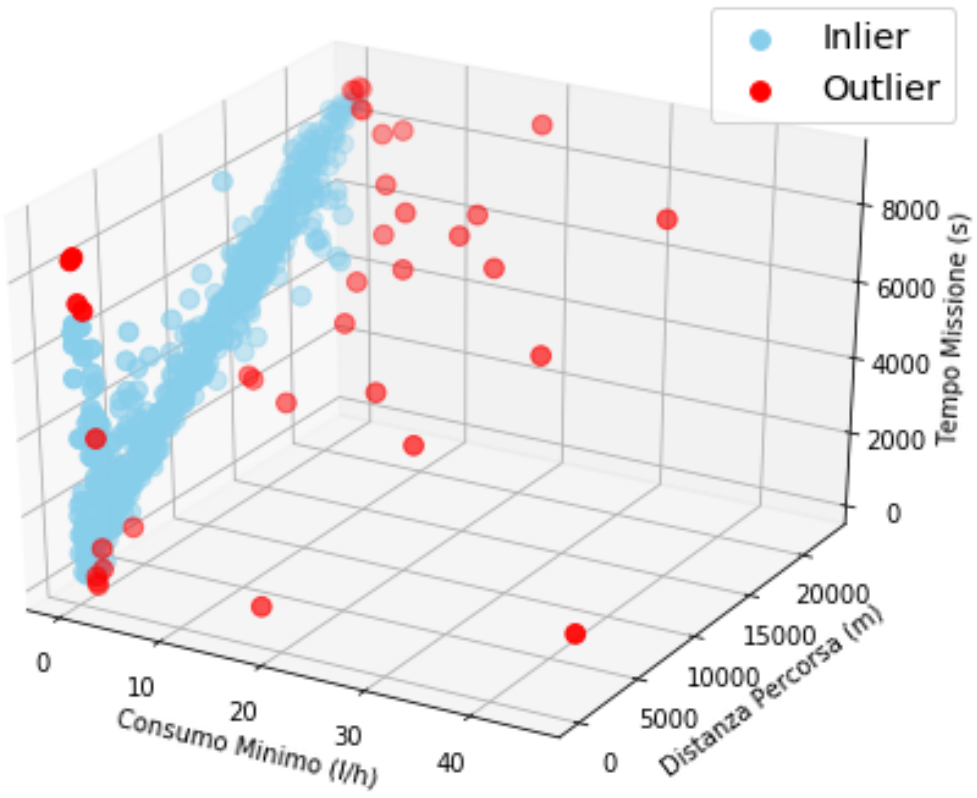

Figura 5.2. Visualizzazione dopo l'esecuzione dell'algoritmo di Isolation Forest

preliminare dell'algoritmo, è stato filtrare tutti quei valori impostati erroneamente a 0 in alcuni casi durante la fase di campionamento, che rappresentano chiaramente

dei difetti nel processo di trasmissione. Come esempio concreto può essere considerato il valore registrato per il peso del veicolo che, a prescindere dalle condizioni in cui ci si trova, non può assumere tale valore. Queste imprecisioni nei dati rientrano nel problema di Data Quality e possono essere particolarmente dannose per la soluzione adottata in quanto occorrenze ripetute di valori inammissibili potrebbero portare alla creazione di zone ad elevata densità, rendendo impossibile per l'algoritmo riconoscere tali dati come anomalie. I possibili effetti di una povera qualità dei dati su algoritmi di machine learning sono approfonditi in [31].

In figura 5.3 è rappresentata la matrice di confusione che sintetizza i risultati

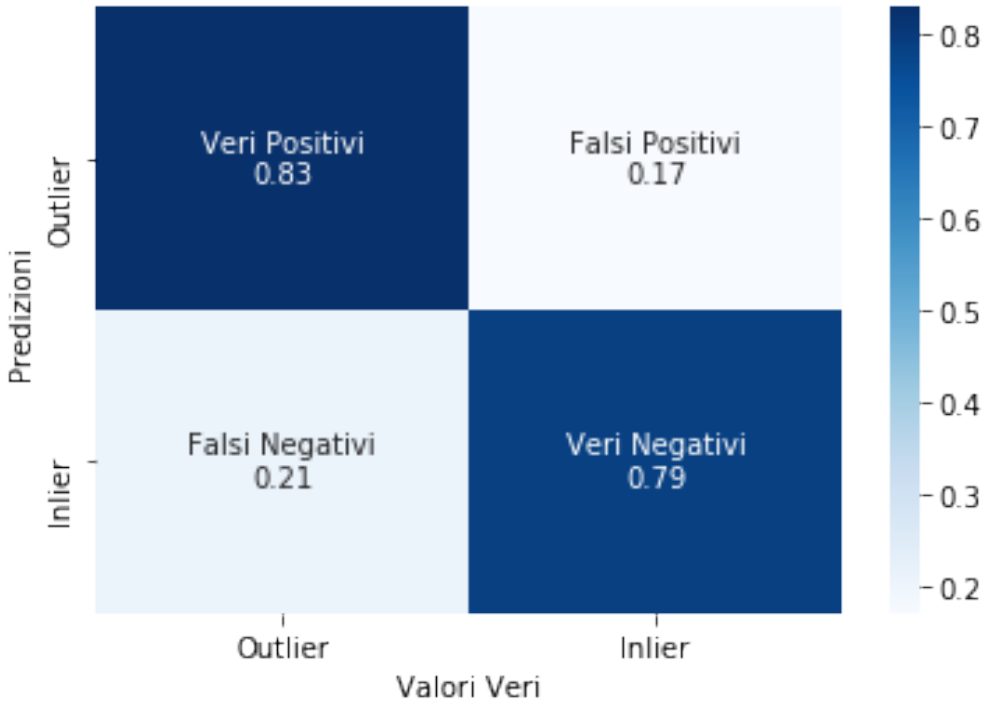

Figura 5.3. Matrice di confusione normalizzata dopo l'applicazione del metodo di Isolation Forest

ottenuti e permette di analizzare gli stessi in maniera oggettiva. Il principale valore da considerare è quello del richiamo (recall) che misura la capacità del modello nel trovare i valori di interesse all'interno del set di dati considerato. Tale valore può essere facilmente calcolato attraverso la formula 4.2 utilizzando i valori riportati in figura. Nel caso analizzato, si ottiene un valore di recall pari a 0.80 che, considerando il notevole numero di dimensioni che caratterizzano i dati, evidenzia le capacità dell'algoritmo di Isolation Forest in un contesto simile e ne giustifica la scelta. L'impatto dell'elevata dimensionalità è comunque visibile nella figura 5.3 osservando il valore dei falsi negativi, non particolarmente basso, che mette

in mostra come il rumore generato dalle dimensioni che non portano informazione particolarmente utile per determinare la natura del dato, vada a nascondere una parte delle anomalie.
Un ulteriore punto vincente del metodo implementato si ottiene confrontando il risultato a disposizione con il lavoro svolto dal meccanismo tradizionale sullo stesso set di dati. Rispetto all'applicazione sequenziale delle regole di business sulle singole dimensioni a disposizione, l'analisi multidimensionale svolta attraverso l'algoritmo di Isolation Forest ha permesso di incrementare il numero di anomalie effettive trovate di circa il 43%.
La principale limitazione legata all'utilizzo dell'algoritmo di isolation forest secondo la struttura illustrata, è la presenza all'interno del dataset di categorie diverse di veicoli. Questa varietà impedisce all'algoritmo di trovare in maniera esaustiva gli outlier presenti in quanto la validità del dato analizzato potrebbe cambiare in base al tipo di veicolo che lo ha generato. Reperire l'informazione della categoria di appartenenza non è sempre possibile e in generale contribuisce in maniera non trascurabile ai tempi di esecuzione necessari e alla complessità globale.
I risultati rappresentati nella figura 5.2, sono stati ottenuti filtrando, dove possibile, la categoria di appartenenza.
Dal punto di vista di possibili sviluppi futuri per incrementare l'efficacia del metodo studiato, particolarmente interessante risulta la versione alternativa dell'algoritmo di isolation forest conosciuta come *Extended Isolation Forest*. Come analizzato nel dettaglio in [32], l'assegnazione dei punteggi di anomalia attraverso l'algoritmo classico, subisce l'influenza negativa delle partizioni che vengono usate per la generazione degli alberi binari. Le soluzioni per mitigare il problema sono diverse: una di queste consiste nel modificare il meccanismo aggiungendo una serie di trasformazioni sui dati prima della generazione di tutti gli alberi di decisione con l'obiettivo di ottenere una situazione priva di bias. Ulteriore tecnica, probabilmente ancora più efficace, viene realizzata introducendo una certa pendenza dei piani utilizzati per il partizionamento dei dati.
L'algoritmo che si ottiene mantiene il funzionamento dell'iForest base e con esso tutti i punti a suo vantaggio. Le modifiche apportate possono contribuire ad un miglioramento dei risultati generati a fronte di un aumento delle tempistiche di esecuzione necessarie che pur non essendo sostanziale, va comunque tenuta in considerazione.

5.3 Ricerca degli outlier per dati con relazioni specifiche

In alcuni casi, oltre che ad un'analisi generale sui dati, che coinvolge tutte le dimensioni presenti, risulta necessaria un'osservazione puntuale su alcuni precisi attributi

che sono considerati particolarmente importanti all'interno del processo decisionale e della valutazioni finali. In questo frangente si è scelto dunque di utilizzare l'algoritmo di Local Outlier Factor presentato nel paragrafo 4.3.2, per la ricerca puntuale di anomalie all'interno di un sottospazio limitato di due dimensioni, che vengono scelte a priori e che sono caratterizzate da un legame stretto in grado di fornire importanti informazioni nella definizione degli outliers. Nella figura 5.4 è stato rappresentato un sottoinsieme di dati prendendo in considerazione il legame tra il peso del veicolo (sull'asse delle ascisse) e il relativo consumo medio (sull'asse delle ordinate).

Il vantaggio principale nell'uso di questo tipo di meccanismo consiste nella pos-

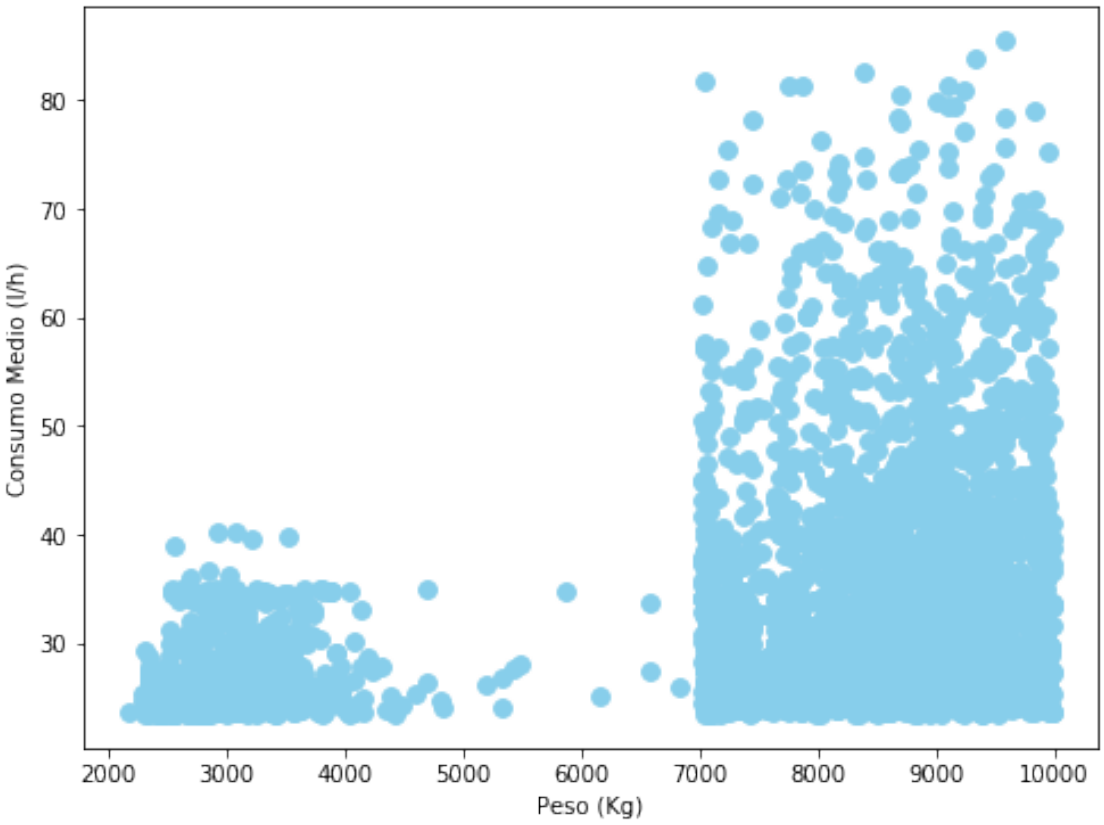

Figura 5.4. Visualizzazione dei dati

sibilità di confrontare diversi tipi e categorie di dati in maniera puntuale, senza che i valori agli estremi del dominio vadano ad influire in maniera significativa sul risultato generale. Come evidenziato dalla figura 5.4 l'intervallo di peso considerato è piuttosto esteso e permette di suddividere abbastanza agevolmente i veicoli più 'leggeri' da quelli invece più 'pesanti'.

Dopo l'applicazione dell'algoritmo, le anomalie sono state segnalate con il colore

rosso all'interno della figura 5.5. Come nel caso basato su isolation forest studiato in precedenza, l'unica operazione preliminare effettuata sui dati è stata quella di rimuovere i valori nulli che, nel caso studiato, rappresentano evidentemente degli errori nel meccanismo di misurazione in quanto è insensato parlare di veicoli a peso nullo o con consumo medio pari a zero. Sebbene in questo caso tali grandezze non dovrebbero influenzare la qualità del risultato vista la natura puntuale dell'algoritmo, eliminare tali valori dall'elaborazione influisce in maniera positiva sulla complessità totale e sul costo computazionale dell'algoritmo.

Per quanto riguarda i parametri dell'algoritmo, è stato usato lo stesso fattore di

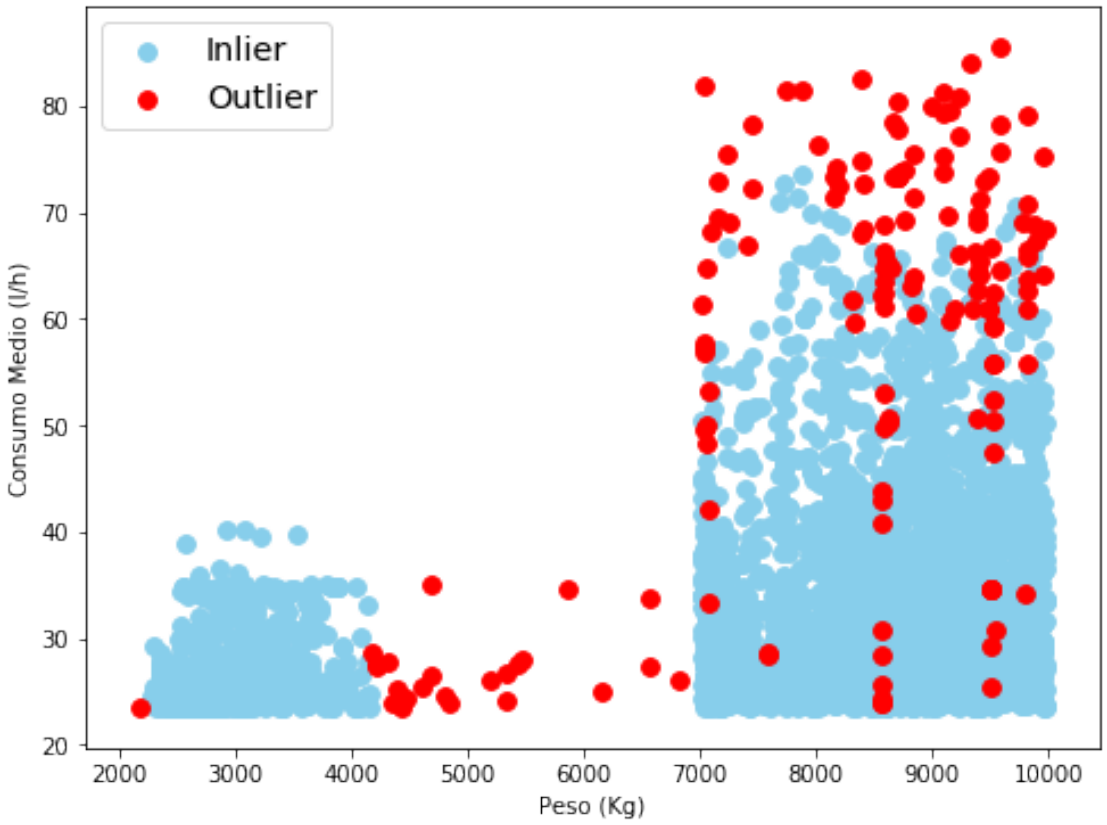

Figura 5.5. Visualizzazione dopo l'esecuzione dell'algoritmo di LOF

contaminazione utilizzato nel caso di isolation forest per il quale valgono le medesime considerazioni. In questo caso, tuttavia, il fattore che maggiormente ha impatto sulla qualità del risultato è il parametro *k (o n-neighbors)* che rappresenta il numero di vicini che devono essere considerati nella definizione di località. Il valore ottimale è il risultato di un processo iterativo che ha portato all'esecuzione dell'algoritmo applicando dieci diversi valori di k all'interno del range definito tenendo conto che valori troppo bassi di tale parametro rendono l'algoritmo particolarmente sensibile

al rumore (come spiegato in maniera più dettagliata nel paragrafo 4.3.2) mentre, nel caso opposto, si va a perdere il concetto di località desiderato, tipico dell'approccio scelto. Un ulteriore parametro che è stato considerato è relativo al metodo di calcolo della distanza e il processo iterativo descritto per il valore di n-neighbors è stato portato avanti per ognuno dei meccanismi utilizzati. Sotto questo punto di vista, le metodologie che sono state usate sono la distanza di Manhattan, la similitudine di coseno e la distanza euclidea. Ulteriori informazioni relative all'implementazione che è stata utilizzata sono presentate in [24].
Non avendo a disposizione un processo automatico per la verifica e la validazione dei risultati ottenuti, l'analisi della qualità è stata affrontata con approccio 'manuale', sfruttando i risultati delle regole di business implementate per la soluzione tradizionale e attraverso l'osservazione puntuale di alcuni insiemi di dati appartenenti a diverse porzioni di spazio. Il modello ottimale scelto alla fine del processo descritto, utilizza la distanza di Manhattan come metrica e n-neighbors = 33 per la definizione di località.
La matrice di confusione relativa al risultato è presentata nella figura 5.6. Il valore

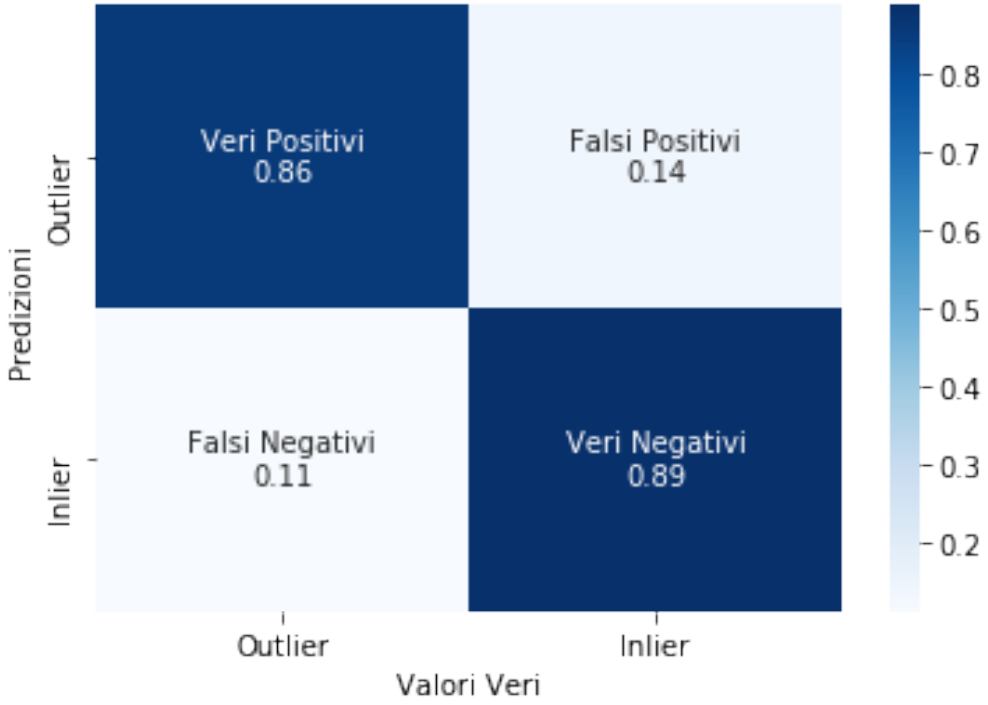

Figura 5.6. Matrice di confusione normalizzata dopo l'applicazione del metodo di Local Outlier Factor

di recall ottenuto in questo caso sale a 0.88, significativamente più alto rispetto al dato ottenuto con il metodo di isolation Forest. Da un confronto diretto tra i due

meccanismi risalta come il valore dei falsi positivi nell'ultimo caso sia notevolmente migliorato, dimostrando come attraverso la limitazione del numero di dimensioni analizzate e il concentrarsi su uno specifico sotto-spazio dei dati, sia possibile giungere a risultati più accurati.
In maniera simile a quanto successo per la soluzione basata su Isolation Forest, anche in questo caso il numero di anomalie correttamente rilevate rispetto al meccanismo tradizionale implementato attraverso le business rules è aumentato. In media infatti sono stati correttamente categorizzati come outliers il 46% di punti in più rispetto alla soluzione originale.
Dal punto di vista degli svantaggi si evidenzia che, algoritmi che concentrano la loro ricerca in porzioni limitate dello spazio possono tralasciare piccoli cluster locali. A questa osservazione si va ad aggiungere il costo computazionale tipicamente elevato, necessario per il calcolo delle distanza tra tutti i punti considerati. Questo fattore, unito al bisogno di trovare il valore ottimale per i parametri cruciali (come n-neighbor) all'interno del relativo dominio, complica ulteriormente l'utilizzo del meccanismo descritto come strumento di analisi.
Aspetto forse ancora più impattante sulle limitazioni rispetto ai tempi di elaborazione, è rivestito dalla necessità di scegliere con molta attenzione le dimensioni su cui concentrare il proprio studio e soprattutto mantenere il numero di dimensioni il più basso possibile per ottenere risultati soddisfacenti dal punto di vista qualitativo. L'algoritmo descritto è infatti condizionato da quella che viene definita come 'curse of dimensionality': all'aumentare del numero di dimensioni, i punti tendono a diventare equidistanti tra loro e gli outliers vengono nascosti dal rumore che si viene a creare.
Una valida alternativa al meccanismo proposto che potrebbe portare alcune migliorie è costituita dal metodo di *LOCI (Local Correlation Integral)*. A differenza del Local Outlier Factor che pur facendo parte dei metodi basati sulla densità fa largo uso della distanza per la definizione della stessa, il LOCI va a calcolare la densità vera e propria di un punto attraverso l'identificazione del numero di ulteriori punti all'interno di una circonferenza di raggio specificato a priori. La classificazione degli outlier avviene grazie alla definizione della media di tutte le densità calcolate all'interno del vicinato considerato. Durante l'analisi è possibile utilizzare diversi valori da impostare al raggio della circonferenza per ottenere una risposta a diversi livelli di granularità. Un punto viene considerato un outlier se il suo punteggio di anomalia, determinato attraverso trasformazioni del valore di densità media, risulta particolarmente più grande (tipicamente 3 volte il valor medio) per ogni granularità considerata.
Attraverso alcuni accorgimenti è possibile fornire un'approssimazione particolarmente accurata della grandezza usata per il calcolo del punteggio di anomalia, velocizzando significativamente le operazioni necessarie e rendendo l'algoritmo piuttosto efficiente. Un altro punto di forza rilevante dell'algoritmo si ha in termini di espressività: è possibile comprimere un alto livello di informazioni all'interno di

un grafico bidimensionale, caratteristico per ogni punto, che aiuta a comprendere le relazioni presenti per i punti all'estremità della distribuzione ai diversi livelli di granularità e il motivo per cui è stato assegnato il relativo valore di anomalia. L'uso intrinseco di diversi livelli di granularità permette di evitare uno dei problemi critici presenti nel LOF, ovvero individuare il corretto valore di n-neighbor. Sebbene l'uso dei grafici del LOCI sia uno strumento molto utile per la classificazione, di contro ogni grafico è specifico per il singolo punto e tale meccanismo risulta veramente efficace se utilizzato in maniera limitata su un numero di punti di particolare interesse, identificati dall'algoritmo stesso.

Capitolo 6

Conclusioni e sviluppi futuri

Dai risultati ottenuti e analizzati nel capitolo precedente, si è dimostrato come sia possibile automatizzare il processo di Data Quality inglobato nella pipeline di elaborazione, sfruttando gli algoritmi studiati. Le soglie fisse presenti nel processo originale sono state eliminate e l'intervento manuale necessario si limita alla selezione delle dimensioni di interesse nel caso descritto nel paragrafo 5.3.
I valori estratti nella fase di valutazione per entrambi gli algoritmi mostrano i molteplici punti di forza degli stessi rispetto alla soluzione tradizionale. I meccanismi presentati risultano senza dubbio nettamente migliori dal punto di vista della scalabilità. Il lungo e difficile compito richiesto nel modello originale per la definizione delle regole di identificazione delle anomalie, non è più necessario e viene soppiantato da una soluzione adattiva che modifica le proprie scelte in maniera coerente in base al tipo di dati che si vanno ad analizzare. La possibilità di includere nel meccanismo di individuazione degli outliers le relazioni presenti tra le varie dimensioni che compongono l'informazione finale, ha inoltre incrementato di oltre il 40% il numero di punti anomali scoperti sia per quanto riguarda il metodo di Isolation Forest, sia per quello di Local Outlier Factor.
I valori di precisione e richiamo non inferiori a 0.80 nel caso dell'algoritmo di iForest, garantiscono prestazioni accettabili già nel caso completamente automatico, che tiene conto di tutte le dimensioni e che non necessita di alcuna ricerca e ottimizzazione dei parametri, pur tenendo presente le limitazioni di applicabilità espresse nel paragrafo 5.2. Questo metodo risulta particolarmente utile quando si vogliono analizzare i dati dal punto di vista generale e fornire un'idea piuttosto precisa sui possibili punti lontani da ciò che ci si aspetta. Per analisi puntuali, risulta invece più appropriato il metodo basato su Local Outlier Factor che permette di scavare più a fondo all'interno di un sotto spazio limitato di particolare interesse. Oltre ad una maggiore accuratezza, viene anche eliminato il vincolo di applicabilità su categorie di dati diverse, visto l'approccio locale con cui viene affrontato il problema. Queste migliorie vengono tuttavia pagate attraverso l'impossibilità di estendere l'analisi ad un numero di dimensioni elevato e alla indispensabile ricerca dei parametri ottimali

usati dall'algoritmo.
Come discusso nella sezione dedicata ai risultati sperimentali, le soluzioni implementate presentano anche degli svantaggi che vanno dal costo computazionale non indifferente a limiti di applicabilità al verificarsi di determinate condizioni. In questo senso possono essere presentati i possibili sviluppi futuri per aumentare ulteriormente l'efficacia e l'efficienza di entrambi i metodi descritti: inglobare nella fase di data quality un meccanismo di selezione dei sottospazi più rilevanti, favorirebbe il metodo basato su isolation forest eliminando un certo numero di dimensioni considerate irrilevanti con considerevole miglioramento dei risultati ottenuti e delle tempistiche richieste e potrebbe fornire all'algoritmo fondato sul Local Outlier Factor una sorta di suggerimento sugli attributi da analizzare, sebbene la presenza di un legame significativo tra queste dimensioni necessiti ancora di una forma di verifica e ulteriore selezione. Tecniche di pre-processing per la *high-contrast subspace selection* possono risultare particolarmente utili per approfondire questa tematica. In questa tecnica i sotto-spazi sono ricavati tramite statistiche di aggregazione che suggeriscono le dimensioni da selezionare basandosi sull'assunzione che la probabilità di trovare comportamenti rari sia più elevata negli spazi non uniformi.
Un ulteriore interessante sviluppo è quello di analizzare gli outlier ponendo il fattore temporale come punto cruciale dell'analisi. Nei metodi affrontati i dati vengono analizzati a livello di 'missione', in maniera indipendente. Osservare l'andamento dei dati nel tempo, in missioni successive, potrebbe portare alla scoperta di ulteriori anomalie non individuabili attraverso l'approccio a singola missione.
Quando si analizzano delle serie temporali, il principale fattore di identificazione delle anomalie è dato dalla presenza di cambiamenti improvvisi e particolarmente rilevanti tra dati appartenenti a rilevazioni successive. Questo metodo risulta dunque strettamente legato al problema di rilevazione di eventi anomali, che possono essere contestuali o collettivi rispetto ad una serie temporale.
La principale limitazione per questo approccio consiste nell'incapacità di considerare outliers quei valori che cambiano lentamente nel tempo, risultando dunque sufficientemente vicini al valore dell'istante di tempo precedente, ma che in determinati momenti raggiungono valori che si discostano in maniera eccessiva da quelli attesi.
Largo spazio in questo contesto trova l'approccio basato su controlli in tempo reale, non appena nuovi dati sono ingeriti dal sistema. Tipicamente i controlli in tempo reale sono particolarmente utili nell'analisi del cambiamento dei dati rispetto al tempo mentre i meccanismi *offline* permettono di esplorare ulteriori aspetti e comportamenti.
Nel caso multidimensionale, l'osservazione dei flussi di dati potrebbe evidenziare una variazione nella distribuzione aggregata degli stessi ma, nel contempo, una variazione ridotta dal punto di vista individuale per ogni attributo, o viceversa. Lo studio della qualità di dati multidimensionali resta tuttavia uno dei settori più

difficili da affrontare, dove la capacità computazionale gioca un ruolo fondamentale per determinare gli spazi esplorabili. La chiave del successo è probabilmente rappresentata dal giusto equilibrio tra un'analisi a livello globale e una di livello locale.

Bibliografia

[1] Wikipedia l'enciclopedia libera, cur. *Big data*. 17 Set. 2020. URL: https://it.wikipedia.org/wiki/Big_data.

[2] Cesarini M. et al. *Data Quality: Un Approccio Metodologico ed Applicativo*. Documentazione. CRISP, 10 dic. 2014.

[3] Richard Farnworth. «The Six Dimensions of Data Quality — and how to deal with them». In: (28 giu. 2020). A cura di Towards Data Science. URL: https://towardsdatascience.com/the-six-dimensions-of-data-quality-and-how-to-deal-with-them-bdcf9a3dba71.

[4] IBM. *Integrazione dei dati*. 5 Apr. 2019. URL: https://www.ibm.com/it-it/analytics/data-integration.

[5] Irene Di Deo. *Data Integration: cosa significa, come farla e perché farla*. 23 Gen. 2019. URL: https://blog.osservatori.net/it_it/data-integration-cosa-significa-come-farla.

[6] Talend. *Data Integration: cosa significa, come farla e perché farla*. 18 Lug. 2019. URL: https://www.talend.com/it/resources/what-is-etl/.

[7] Medium. *What is data Matching?* 1 Lug. 2018. URL: https://medium.com/neuronio/what-is-data-matching/.

[8] Li Cai e Yangyong Zhu. «The Challenges of Data Quality and Data Quality Assessment in the Big Data Era». In: *Data Science Journal* (22 mag. 2015). DOI: 10.5334/dsj-2015-002.

[9] Charu C. Aggarwal. *Outlier Analysis*. A cura di Springer. 25 Nov. 2016.

[10] Irad Ben-gal. «Outlier Detection». In: (10 gen. 2005). DOI: 10.1007/0-387-25465-X-7.

[11] Sunderland et al. «The utility of multivariate outlier detection techniques for data quality evaluation in large studies: an application within the ONDRI project.» In: *BMC Medical Research Methodology* (15 mag. 2019). DOI: 10.1186/s12874-019-0737-5.

[12] Swetha Lakshmanan. «Outlier Detection and Treatment: A Beginner's Guide». In: (8 mag. 2019). A cura di Medium. URL: https://medium.com/@swethalakshmanan14/outlier-detection-and-treatment-a-beginners-guide-c44af0699754.

[13] Wikipedia - l'enciclopedia libera, cur. *Dbscan*. 14 Nov. 2019. URL: https://it.wikipedia.org/wiki/Dbscan.

[14] Sergio Santoyo. «A Brief Overview of Outlier Detection Techniques». In: (12 set. 2017). A cura di Towards Data Science. URL: https://towardsdatascience.com/a-brief-overview-of-outlier-detection-techniques-1e0b2c19e561.

[15] Wikipedia - l'enciclopedia libera, cur. *Precisione e recupero*. 5 Mag. 2020. URL: https://it.wikipedia.org/wiki/Precisione_e_recupero.

[16] *La differenza tra precision e recall*. 25 Lug. 2019. URL: http://www.andreaminini.com/ai/machine-learning/la-differenza-tra-precision-e-recall.

[17] *Precision-Recall*. 15 Gen. 2020. URL: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html.

[18] Devin Soni. «Supervised vs. Unsupervised Learning». In: (22 mar. 2018). A cura di Towards Data Science. URL: https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d.

[19] Microsoft Documentation. *Che cos'è Apache Spark?* 15 Ott. 2019. URL: https://docs.microsoft.com/it-it/dotnet/spark/what-is-spark.

[20] *Apache Spark*. URL: https://spark.apache.org/.

[21] Arpit. «Isolation Forest algorithm for anomaly detection». In: (18 gen. 2020). A cura di Medium. URL: https://medium.com/@often_weird/isolation-forest-algorithm-for-anomaly-detection-f88af2d5518d.

[22] Fei Tony Liu et al. *Isolation Forest*. Monash University, Victoria, Australia, 22 nov. 2018.

[23] Scikit Learn. *Isolation Forest*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html.

[24] Scikit Learn. *Local Outlier factor*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html.

[25] Microsoft Documentation. *Microsoft Azure*. URL: https://azure.microsoft.com/it-it/overview/what-is-azure/.

[26] Microsoft Documentation. *Che cos'è Azure Data Factory?* URL: https://docs.microsoft.com/it-it/azure/data-factory/introduction.

[27] Microsoft Documentation. *Che cos'è Azure databricks*. URL: https://docs.microsoft.com/it-it/azure/databricks/scenarios/what-is-azure-databricks.

[28] Microsoft Documentation. *Che cos'è Azure Data Lake Storage*. URL: `https://docs.microsoft.com/it-it/azure/data-lake-store/data-lake-store-overview`.

[29] Microsoft Documentation. *Introduzione - Azure Delta Lake*. URL: `https://docs.microsoft.com/it-it/azure/databricks/delta/delta-intro`.

[30] Microsoft Documentation. *Introduzione all'archiviazione BLOB di Azure*. URL: `https://docs.microsoft.com/it-it/azure/storage/blobs/storage-blobs-introduction`.

[31] Sessions Valerie e Valtorta Marco. *The effects of data quality on machine learning algorithms*. University of South California USA.

[32] Robert J. Brunner et al. *Extended Isolation Forest*. 8 Lug. 2020. URL: `https://arxiv.org/pdf/1811.02141.pdf`.