Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

Autodiagnostica di anomalie attraverso algoritmi di machine learning

Relatore
Lamberto Ballan
Università di Padova

Laureando
Enrico Muraro

Alla mia famiglia.

Indice

Elenco delle figure

Elenco delle tabelle

1

Introduzione

Il machine learning è una branca dell'intelligenza artificiale sempre più di interesse per le aziende, anche per quelle che non riguardano direttamente il campo dell'informatica.

Le applicazioni di machine learning sono già molto numerose, e spesso vengono utilizzate senza nemmeno rendersene conto. Ad esempio viene utilizzato per il riconoscimento vocale o identificazione della scrittura manuale, dai motori di ricerca per mostrare i risultati più rilevanti, oppure nel settore della ricerca scientifica in campo medico dove gli algoritmi imparano a effettuare diagnosi di tumori o altre malattie prima che diventino un problema. Un'altra delle possibili applicazioni per il machine learning è il riconoscimento delle anomalie nel funzionamento di uno strumento o macchinario.

Avere sotto controllo lo stato di salute dei propri strumenti è di grande interesse per qualunque azienda, e con il machine learning è possibile non solo ottenere una diagnostica automatica delle anomalie, ma anche prevedere la rottura dello strumento prima che possa causare ulteriori danni.

Tutto questo è possibile solo con una grande quantità di dati sul problema che si vuole affrontare, sia per il riconoscimento vocale, sia per l'autodiagnostica di un macchinario. Meno sono i dati disponibili meno risulterà efficace ed utile l'applicazione di algoritmi di machine learning

2

Contesto aziendale

2.1 L'azienda

Lo stage si è svolto presso l'azienda Pietro Fiorentini S.p.A nella sede principale di Arcugnano. Pietro Fiorentini realizza prodotti e servizi tecnologicamente avanzati per la distribuzione e l'utilizzo del petrolio e del gas naturale a livello globale, con undici stabilimenti nel mondo. L'azienda realizza prodotti come valvole, filtri, regolatori di pressione, ma lo stage si è interessato in particolare sui Multiphase Flow Meter.

2.2 Multiphase Flow Meter

Un Multiphase Flow Meter, o MPFM in breve, è un dispositivo utilizzato per misurare le singole portate delle fasi costitutive di un determinato flusso. In questo caso il flusso è una miscela di petrolio, acqua e gas creata durante il processo di estrazione dal pozzo petrolifero. I MPFM prodotti dalla Pietro Fiorentini sono strumenti non intrusivi per effettuare misurazioni in tempo reale del flusso evitando l'uso di sistemi basati sulla separazione delle diverse fasi. Gli MPFM sono modulari e possono essere installati in diverse configurazioni, ogni modulo si occupa di effettuare una o più misurazioni sul flusso.

La misurazione dei flussi multifase è un problema complesso e richiede la considerazione di molte possibili condizioni. In particolare ci sono molti regimi di scorrimento del flusso a seconda della velocità delle parti liquide e gassose [1].

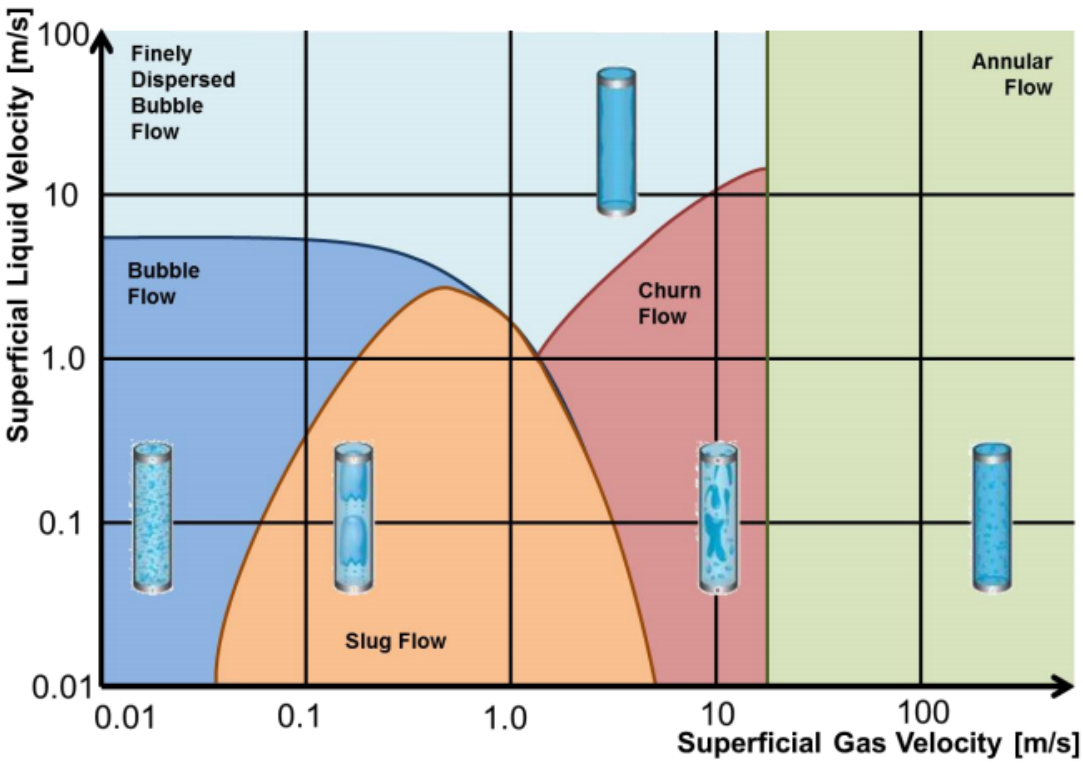

figura 2.1: I principali regimi di scorrimento del flusso multifase in base alla velocità della parte liquida e della parte gassosa [1].

I principali regimi di scorrimento sono:

- Bubble: il gas è sospeso in bollicine all'interno del liquido, accade quando la parte liquida è più veloce di quella gassosa.
- Finely dispersed bubble: il gas è sospeso in bollicine molto piccole all'interno del liquido.
- Slug: il flusso è a intermittenza prevalentemente liquido e prevalentemente gassoso, formando delle grosse bolle di gas.
- Churn: all'aumentare della velocità del gas il flusso diventa irregolare e la parte liquida sale e scende continuamente.
- Annular: ad alte velocità del gas la parte liquida forma un anello intorno alle pareti della tubatura, e il gas scorre nel centro.

L'irregolarità del flusso rende anche più difficile il riconoscimento delle anomalie nel funzionamento dello strumento. Dei valori che a prima vista sembrano anomali potrebbero semplicemente essere delle bolle più grosse di gas, oppure il passaggio improvviso a un altro regime di scorrimento. In questi casi le anomalie registrate fanno parte del funzionamento corretto dello strumento e non devono essere quindi considerati come valori anormali.

2.3 Struttura dei dati

I dati rilevanti al progetto riguardano una serie di test effettuati in vari laboratori negli ultimi dieci anni. I test consistono nell'uso di un Multiphase Flow Meter in un circuito chiuso, in modo da poter controllare accuratamente la composizione e le condizioni del flusso che viene fatto circolare.

Durante il test si registrano le letture di tutti i sensori presenti nel MPFM, questi dati vengono quindi salvati in file singoli, ognuno dei quali contiene in genere un minuto di lettura. I file sono in formato proprietario BIN o BIX e rappresentano i dati nel loro stato "raw", ad esempio un sensore che misura la pressione non salverà direttamente dei valori in atmosfere ma potrebbe salvare dei voltaggi che dovranno essere a loro volta interpretati.

Le informazioni sulla configurazione del MPFM e sulle condizioni del flusso non sono salvate nei file raw, sono salvate invece in file separati, rispettivamente un file di testo per la configurazione del MPFM e un foglio di calcolo per le condizioni del flusso. Questo foglio di calcolo, chiamato anche foglio di riferimento, rappresenta quindi le condizioni in cui sono state effettuate le letture dei sensori, in particolare descrive le seguenti proprietà del flusso per ogni file raw:

- Qgas: Il volume della componente di gas nel flusso
- Qwater: Il volume della componente d'acqua nel flusso
- Qwater: Il volume della componente di petrolio nel flusso
- WLR: Water Liquid Ratio, la percentuale di acqua sul totale del liquido. Calcolato con la formula:
$$\frac{Qwater}{Qwater + Qoil}$$
- GVF: Gas Volume Fraction, la frazione di gas sul volume totale. Calcolato con la formula:
$$\frac{Qgas}{Qwater + Qoil + Qgas}$$

- Pressure: La pressione del flusso
- Temperature: La temperatura del flusso

3
Il progetto di stage

Pietro Fiorentini sta sviluppando un sistema di auto-diagnostica per i Multiphase Flow Meter, con l'obiettivo di avere sistemi sempre più intelligenti che riescano a dare informazioni relativamente al loro stato di funzionamento e salute. I dati raccolti da vari sensori presenti nei MPFM sono analizzati attraverso algoritmi di machine learning per identificare automaticamente eventuali anomalie o malfunzionamenti. Il progetto include la creazione di adeguate infrastrutture software per la gestione dei dati e la loro analisi, per poi poter applicare e valutare gli algoritmi di machine learning.

3.1 Piano di lavoro

Lo stage segue un piano di lavoro redatto prima dell'inizio dello stage dal tutor aziendale in collaborazione con lo stagista e approvato dal tutor universitario. Il piano di lavoro definisce gli obiettivi di ogni settimana di stage.

3.1.1 Settimana 1 - Analisi dei requisiti

- Incontro con persone coinvolte nel progetto Machine Learning per discutere i requisiti e le richieste relativamente al progetto e alla qualità del software
- Verifica credenziali e strumenti di lavoro assegnati
- Presa visione dell'infrastruttura esistente

- Analisi dei requisiti, tenendo in considerazione il preesistente
- Discussione e redazione del documento del "Requirement Specification"

Risultato: Documento "Requirement Specification" che raccoglie tutti i vincoli e requisiti del progetto.

3.1.2 Settimana 2 e 3 - Gestione dei dati

- Studio dei dati e delle informazioni a corredo (quando sono stati raccolti, dove, con quale strumento)
- Studio delle infrastrutture per gestire i dati (DB, Data Warehouse, Data Lake, etc)
- Creare struttura per gestire e accedere facilmente ai dati, combinando tutte le informazioni disponibili

Risultato: Documento che spiega come implementare una infrastruttura per la gestione dei dati "Data Management Design" e l'infrastruttura software per la gestione dei dati.

3.1.3 Settimana 4 e 5 - Analisi dei dati

- Studio del formato con cui sono salvati i dati (non standard, proprietario)
- Librerie in Python per accedere al contenuto dei dati e poterli analizzare
- Creare la struttura software per estrarre le informazioni dai dati e per creare grafici e analizzarli
- Strumento o interfaccia per visualizzare e analizzare dati ricavati da fonti diverse

Risultato: Documento "Data Analytics Design" che descriva come implementare la lettura dei dati grezzi, creare grafici e analizzarli, creazione del software per interfacciarsi con i dati.

3.1.4 Settimana 6 - Feature extraction

- Studio di metodi per estrazione di caratteristiche dai dati e metriche per il confronto dei metodi, con lo scopo di rendere più semplice l'individuare anomalie
- Implementazione e confronto dei vari metodi per estrarre caratteristiche

Risultato: Documento "Feature Extraction Design & Report" che descriva i metodi di estrazione e il loro confronto e valutazione, creazione del software con i metodi implementati.

3.1.5 Settimana 7 - Anomaly detection

- Studio di modelli e algoritmi di machine learning per la rilevazione di anomalie, studio di metriche per il confronto
- Valutazione e confronto dei modelli sui dati per identificare anomalie sulla base di metriche prestabilite

Risultato: Documento ”Anomaly Detection Design & Report” che descriva gli algoritmi di identificazione anomalie, il loro confronto e valutazione, creazione del software con gli algoritmi implementati

3.1.6 Settimana 8 - Conclusione

- Incontro con il team di Ricerca e Sviluppo per presentare e insegnare ad usare quanto prodotto
- Live-demo di tutto il lavoro di stage

Risultato: Presentazione e live-demo

3.2 Obiettivi

3.2.1 Obiettivi minimi

Documenti:

- Requirement Specification: sezione ”General Requirement”
- Data Management Design: diagramma a blocchi dell’infrastruttura
- Data Analytics Design: diagramma a blocchi dell’infrastruttura
- Feature Extraction Design & Report: descrizione di un solo metodo di feature extraction
- Anomaly Detection Design & Report: due algoritmi di anomaly detection descritti e valutati
- Presentazione finale

Software:

- Database di gestione dei dati
- Software per l'analisi dei dati (script per accedere ai dati)
- Software per feature extraction, un solo metodo implementato
- Software per anomaly detection, implementazione, confronto e valutazione di due algoritmi

3.2.2 Obiettivi desiderabili

Tutti gli obiettivi minimi con le seguenti aggiunte o modifiche.

Documenti:

- Requirement Specification completo di informazioni per verificare e testare i requisiti
- Data Management Design completo
- Data Analytics Design completo
- Feature Extraction Design & Report: descrizione di tre metodi di feature extraction, valutazione e confronto
- Anomaly Detection Design & Report: descrizione di cinque algoritmi di anomaly detection, valutazione e confronto
- Presentazione finale e dimostrazione live del software sviluppato su un set di dati

Software:

- Infrastruttura di gestione dati completamente implementata
- Software per l'analisi dei dati completamente implementato
- Software per feature extraction, confronto e valutazione di tre metodi
- Software per anomaly detection, confronto e valutazione di cinque algoritmi

3.3 Vincoli

Il progetto deve rispettare dei determinati vincoli imposti dall'azienda o dall'università

3.3.1 Vincoli temporali

L'università impone un vincolo temporale minimo di 300 ore di lavoro durante lo stage. Per questo motivo il progetto è stato diviso in 8 settimane, lavorando 40 ore a settimana per un totale massimo di 320 ore.

3.3.2 Vincoli metodologici

I documenti, il codice e i commenti scritti durante il progetto devono essere completamente in inglese.

Tutto ciò che viene prodotto durante lo stage deve essere versionato nel repository Git fornito dall'azienda.

L'azienda non ha definito altri vincoli metodologici, e ha dato libera scelta per gli strumenti di sviluppo. Come IDE è stato utilizzato PyCharm essendo uno degli IDE più completi per lo sviluppo in Python. GitKraken è stato usato come client di Git per facilitare il lavoro condiviso attraverso il repository fornito. Per la documentazione del codice è stato usato Sphinx, in modo da generare la documentazione automaticamente a partire dai commenti presenti nel codice.

3.3.3 Vincoli tecnologici

L'utilizzo di Python 3 come linguaggio di programmazione è stato fortemente consigliato dall'azienda vista la presenza di molte librerie per la visualizzazione dei dati e l'applicazione di feature extraction e anomaly detection.

La scelta del database è stata libera, ma MySql è stato preferito in quanto già installato e funzionante all'interno dell'azienda.

4

Analisi dei requisiti

Questo capitolo descrive i requisiti individuati per soddisfare tutti gli obiettivi definiti nel piano di lavoro. I requisiti descrivono in modo chiaro tutte le funzionalità del sistema finale senza entrare in dettagli implementativi o vincolare l'uso di specifiche librerie o framework.

4.1 Casi d'uso

I casi d'uso riportati sono emersi nella prima settimana di lavoro, in base al piano di lavoro stabilito precedentemente e alle informazioni fornite dal responsabile aziendale. Questi casi d'uso forniscono una panoramica di tutte le funzionalità che dovrà avere il sistema completo.

L'unico attore è l'utente o user, ovvero colui che utilizza il sistema. Nel contesto aziendale sarà uno dei ricercatori che vuole analizzare o effettuare altre operazioni sui dati.

4.1.1 Gestione dei dati

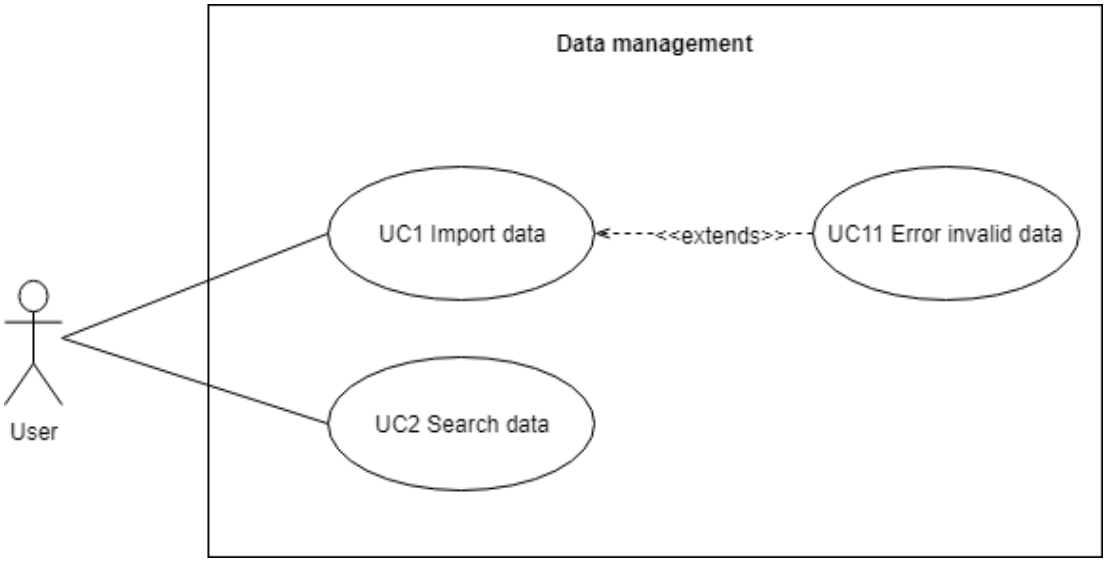

figura 4.1: Panoramica dei casi d'uso del sistema di gestione dei dati.

- UC1 Import data: l'utente può importare nuovi dati nel sistema di gestione dati
- UC2 Search data: l'utente può cercare particolari dati in base a una condizione data, il risultato viene ritornato per poter essere analizzato in seguito
- UC11 Error invalid data: l'utente viene informato se i dati importati non sono nel formato corretto

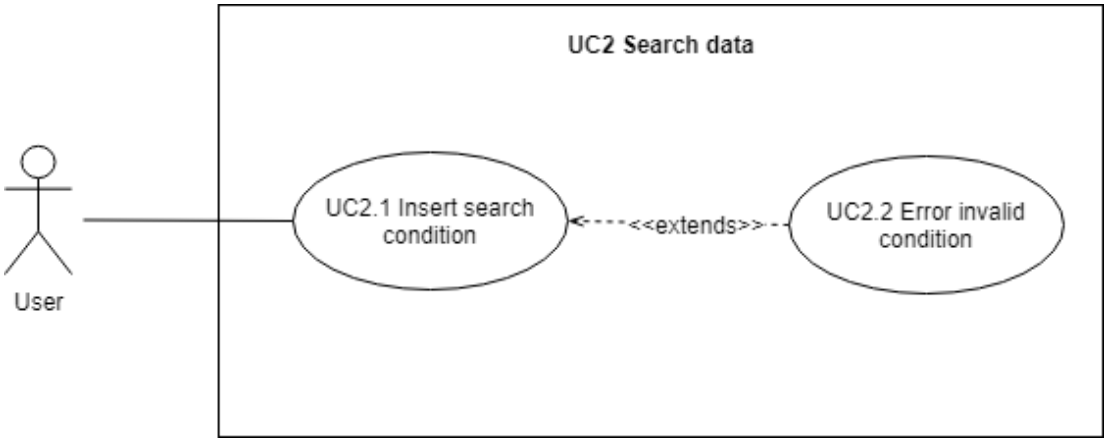

figura 4.2: Sottocasi d'uso di UC2.

- UC2.1 Insert search condition: l'utente può specificare una condizione di ricerca che include uno o più dei seguenti parametri dei dati:
  - Qgas: Il volume della componente di gas nel flusso
  - Qwater: Il volume della componente d'acqua nel flusso
  - Qwater: Il volume della componente di petrolio nel flusso
  - WLR: Water Liquid Ratio, la percentuale di acqua sul totale del liquido
  - GVF: Gas Volume Fraction, la frazione di gas sul volume totale
  - Pressure: La pressione del flusso
  - Temperature: La temperatura del flusso
  - Modules: Quale dei moduli era installato il momento della lettura dei dati
  - Time: La data e l'orario di lettura dei dati
  - Place: Il luogo in cui sono stati letti i dati
- UC2.2 Error invalid condition: l'utente viene informato se la condizione inserita non è formattata correttamente

4.1.2 Analisi dei dati

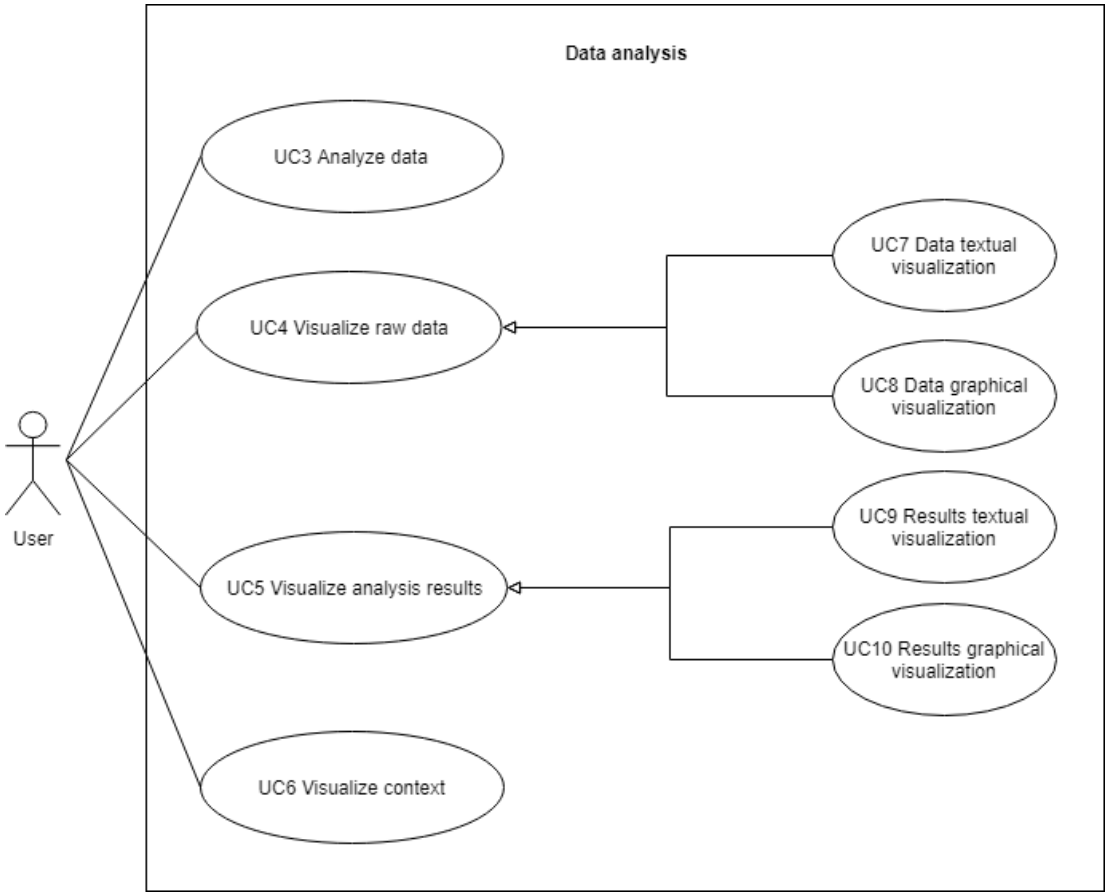

figura 4.3: Panoramica dei casi d'uso del sistema di analisi dei dati.

- UC3 Analyze data: l'utente può applicare un metodo di analisi dei dati, il risultato è adatto alla visualizzazione grafica e testuale
- UC4 Visualize raw data: l'utente può visualizzare i dati originali
- UC5 Visualize analysis results: l'utente può visualizzare il risultato dell'analisi
- UC6 Visualize context: l'utente può accedere a tutti i documenti relativi ai dati, ad esempio file di setup, file excel di riferimento, note del laboratorio
- UC7 Data textual visualization: l'utente può visualizzare i dati in un formato testuale
- UC8 Data graphical visualization: l'utente può visualizzare i dati in un formato grafico

- UC9 Results textual visualization: l'utente può visualizzare i risultati dell'analisi in un formato testuale
- UC10 Results graphical visualization: l'utente può visualizzare i risultati dell'analisi in un formato grafico

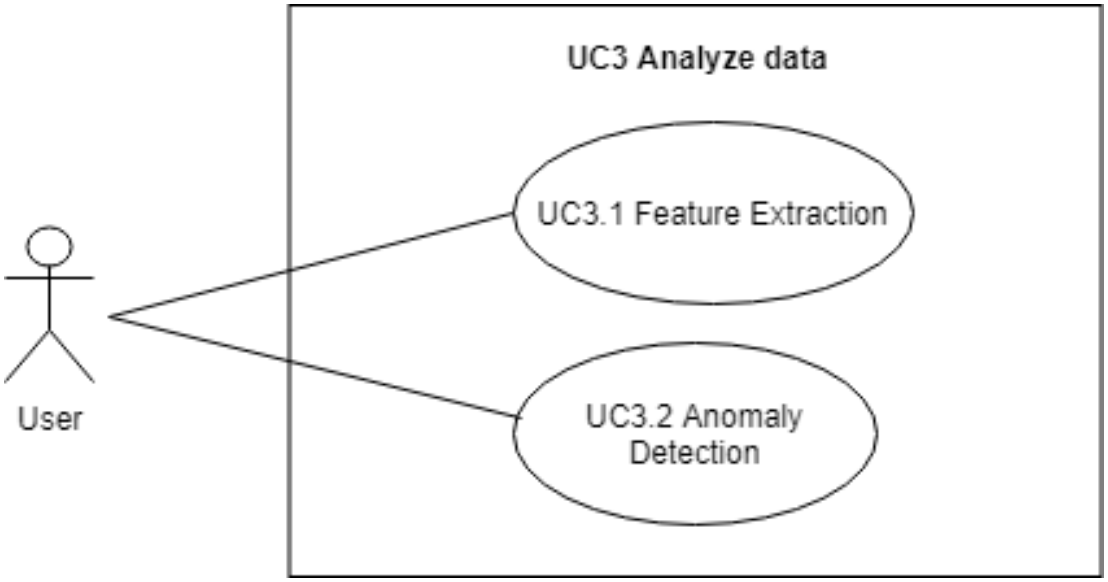

figura 4.4: Sottocasi d'uso di UC3.

- UC3.1 Feature extraction: l'utente può applicare il metodo di "Feature extraction" desiderato
- UC3.2 Anomaly detection: l'utente può applicare il metodo di "Anomaly detection" desiderato

4.2 Requisiti

4.2.1 Classificazione dei requisiti

Tutti i requisiti sono classificati in base alla tipologia, alla priorità e alla fonte. Ogni requisito è inoltre associato a un codice identificativo univoco.

Le possibili tipologie sono:

- GR: General Requirements, per i requisiti generali sul funzionamento

- SR: System Requirements, per i requisiti che riguardano le caratteristiche del sistema
- QR: Quality Requirements, per i requisiti di qualità
- DOC: Documentation, per i requisiti che riguardano la documentazione

Le possibili priorità sono:

- Obbligatorio
- Desiderabile
- Opzionale

Le possibili fonti sono:

- PF: Pietro Fiorentini
- Interno

4.2.2 Requisiti generali

ID	Priorità	Requisito	Fonte
GR-001	Obbligatorio	L'utente può importare nuovi dati nel sistema	PF
GR-002	Obbligatorio	L'utente può cercare particolari dati in base a una condizione data	PF
GR-003	Desiderabile	L'utente può applicare un qualunque metodo di analisi al risultato della ricerca	PF
GR-004	Obbligatorio	L'utente può applicare il metodo di feature extraction desiderato	PF
GR-005	Obbligatorio	L'utente può applicare il metodo di anomaly detection desiderato	PF
GR-006	Obbligatorio	L'utente può visualizzare i dati in un formato testuale	PF
GR-007	Desiderabile	L'utente può visualizzare i dati in un formato grafico	PF
GR-008	Obbligatorio	L'utente può visualizzare i dati risultanti da un'analisi in un formato testuale	PF
GR-009	Desiderabile	L'utente può visualizzare i dati risultanti da un'analisi in un formato grafico	PF
GR-010	Desiderabile	L'utente può accedere a tutti i documenti contestuali relativi ai dati ricercati	PF
GR-011	Obbligatorio	Il sistema deve offrire un'interfaccia per cercare e ritornare i file richiesti	PF
GR-012	Obbligatorio	Il sistema deve offrire un'interfaccia per accedere al contenuto dei file richiesti	PF
GR-013	Obbligatorio	Il sistema è in grado di leggere e utilizzare il contenuto dei file BIN e BIX	PF
GR-014	Opzionale	Il sistema è in grado di leggere e utilizzare il contenuto dei file txt e xls	PF

Table 4.1: Tabella dei requisiti generali

4.2.3 Requisiti di sistema

ID	Priorità	Requisito	Fonte
SR-001	Desiderabile	Il sistema è sviluppato in Python	PF
SR-002	Desiderabile	Il sistema funziona in un sistema operativo Linux	PF

Table 4.2: Tabella dei requisiti di sistema

4.2.4 Requisiti di qualità

ID	Priorità	Requisito	Fonte
QR-001	Obbligatorio	Il codice segue lo stile guida definito in PEP8	Interno

Table 4.3: Tabella dei requisiti di qualità

4.2.5 Documentazione

ID	Priorità	Requisito	Fonte
DOC-001	Obbligatorio	Stesura del documento ”Requirements Specification”	PF
DOC-002	Obbligatorio	Stesura del documento ”Data Management Design”	PF
DOC-003	Obbligatorio	Stesura del documento ”Data Analytics Design”	PF
DOC-004	Obbligatorio	Stesura del documento ”Feature Extraction Design & Report”	PF
DOC-005	Obbligatorio	Stesura del documento ”Anomaly Detection Design & Report”	PF

Table 4.4: Tabella dei requisiti sulla documentazione

5
Gestione dei dati

Prima di analizzare i dati è importante poter trovarli in modo efficiente ed efficace. Questo capitolo descrive i sistemi software implementati per la gestione dei dati, incluse operazioni come la ricerca rapida di specifici file raw in base a una o più condizioni, ma non affronta la lettura dei file stessi.

I file raw sono associati ai metadati disponibili nel foglio di calcolo di riferimento e nel file di configurazione del MPFM. Ogni file inoltre contiene nel nome la data in cui è stato creato nel formato `"YYYYmmdd hhmm"`

5.1 Architettura generale

L'architettura del sistema di gestione dei dati può essere divisa in due parti principali. La prima parte si occupa di trovare e leggere i file di riferimento e configurazione per associare i metadati ai corretti file raw. La seconda parte si occupa di caricare tutte le informazioni raccolte in un database, per poter effettuare ricerche più complesse e veloci. Navigare il sistema operativo esaustivamente per trovare tutti i file interessati è un'operazione temporalmente costosa, nell'ordine di una decina di minuti per le cartelle più grandi presenti nel server. Per migliorare il tempo di esecuzione vengono ricercati contemporaneamente tutti i tipi di file necessari, sia raw che di riferimento, invece di dover ripetere l'esplorazione più volte. La ricerca deve essere sempre completa di tutte le sottocartelle in quanto i dati arrivano da molti laboratori diversi e la posizione dei file non rispetta uno standard uniforme.

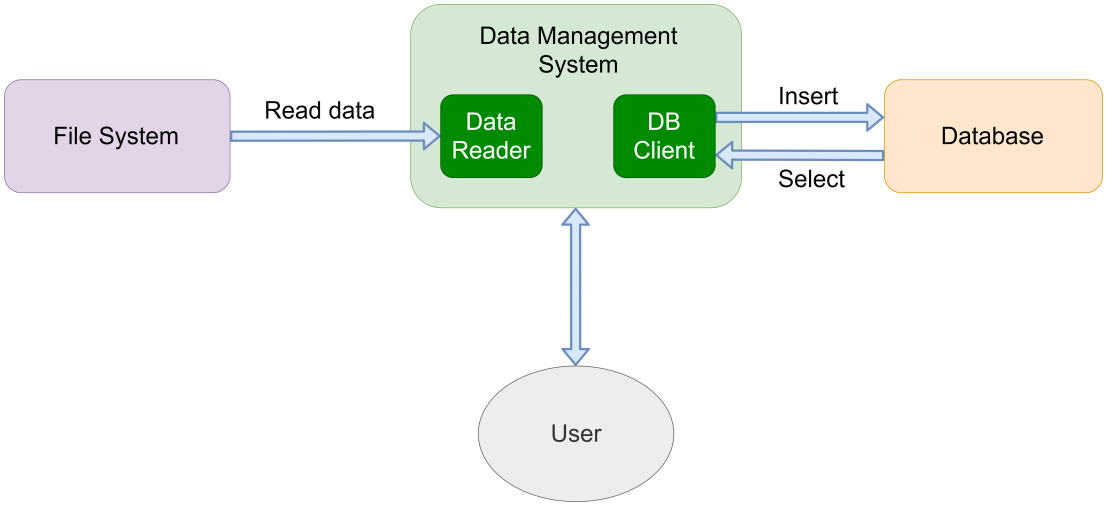

figura 5.1: Architettura generale del sistema di gestione dei dati e le interazioni verso le componenti esterne al sistema.

5.2 Diagramma delle classi

DataReader è la classe principale del sistema di gestione dati e offre tutte le funzionalità necessarie per trovare i file e associarli con i relativi metadati.

ExcelManager, RawManager e SetupManager sono tre classi che gestiscono rispettivamente i file excel, i file raw, e file di testo di setup. Ognuno di questi Manager implementa due metodi fondamentali:

- il metodo `match()` che dato il percorso e il nome di un file è in grado di riconoscere se è un file di interesse per quel manager, ad esempio ExcelManager ritornerà `True` solo se il file dato è l'excel di riferimento corretto.
- il metodo `get_data()` che data la lista di file riconosciuti in precedenza per ognuno estrae le informazioni rilevanti, ad esempio SetupManager ricaverà i moduli installati in base al contenuto del file di testo. RawManager invece si occupa solo di estrarre la data e l'ora dal nome del file e non di leggere il contenuto del raw.

La sezione 2.3 affronta in più dettaglio la struttura dei file e il loro contenuto.

FileManager è una classe astratta che impone l'implementazione di questi due metodi a tutti i Manager che ereditano da essa e permette a DataReader di usarle polimorficamente senza sapere quali e quanti Manager sono presenti. Questa scelta implementativa è stata fatta per rendere facilmente estendibile il sistema, in particolare nell'aggiunta di nuovi Manager per altri tipi di file che si potrebbero voler analizzare in futuro.

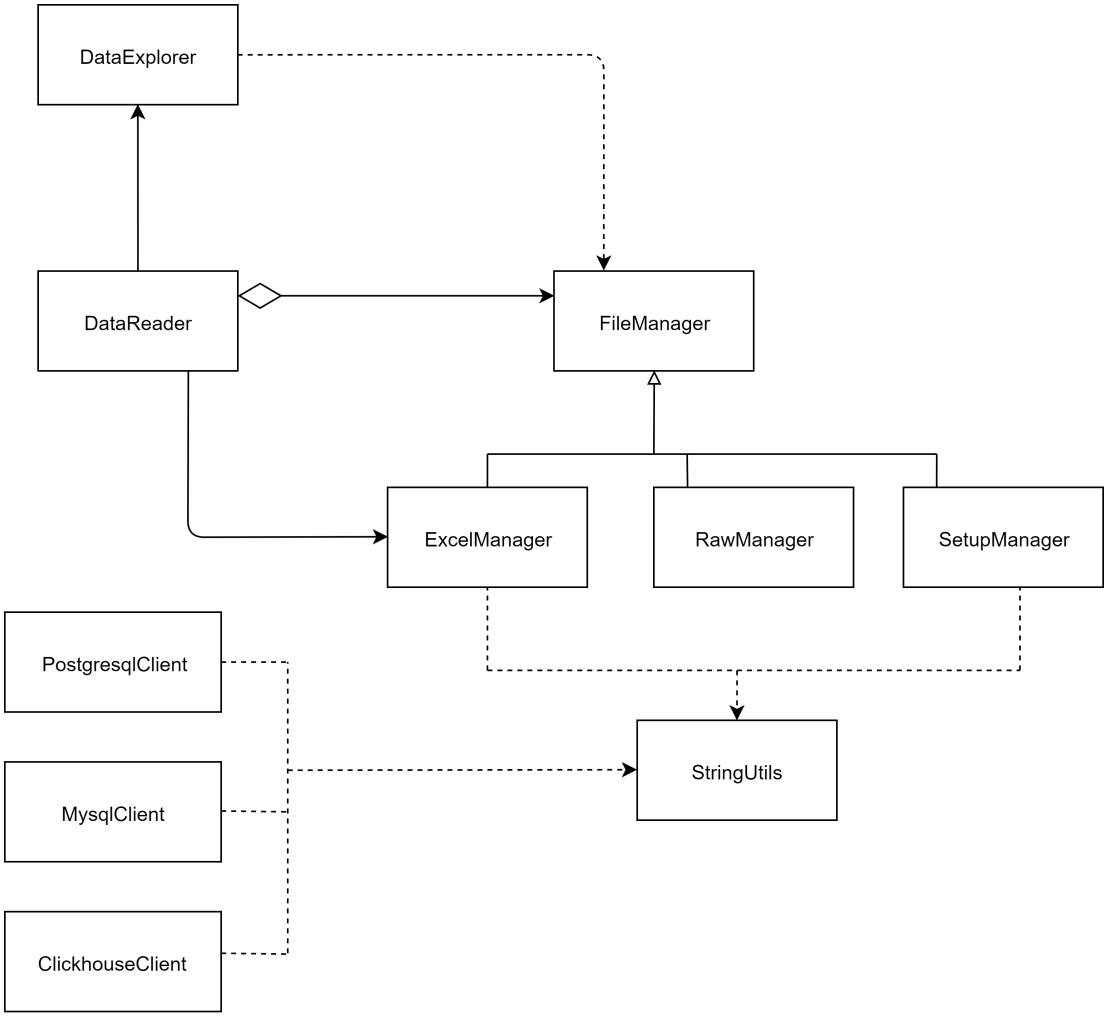

figura 5.2: Diagramma delle classi generale del sistema di gestione dei dati, rappresenta le dipendenze tra le classi senza entrare nei dettagli implementativi.

DataExplorer è la classe usata per navigare il file system e sfrutta il metodo `match()` dei FileManager per costruire una lista di percorsi dei file rilevanti.

StringUtils è una semplice classe di utilità che contiene delle funzionalità comuni per manipolare ed elaborare stringhe.

PostgresqlClient, MysqlClient e ClickhouseClient sono delle classi per la comunicazione con il database. MySQL[2] è il database scelto dall'azienda quindi per ora MysqlClient è l'unico utilizzato, le altre due classi hanno la stessa interfaccia e possono essere intercambiate liberamente se si volesse utilizzare un database diverso. Tutti e tre i client offrono la possibilità di eseguire query SQL oppure creare tabelle e popolarle automaticamente attraverso il risultato ritornato da DataReader.

ClickHouse è un DBMS column-oriented open-source, i dati sono quindi salvati in colonne invece che in righe, questo permette al database di accedere ai dati di una colonna senza dover scorrere tutte le righe e scartare i campi superflui. ClickHouse[3] performa meglio con poche tabelle formate da molte colonne ed è in grado di processare da centinaia di milioni a più di un miliardo di righe al secondo per ogni server. Le limitazioni più importanti sono la mancanza di transazioni, l'alto uso di RAM nelle query che sfruttano l'aggregazione e una debole implementazione delle operazioni UPDATE e DELETE.

PostgreSQL[4] è un DBMS relazionale ad oggetti open-source, con una forte enfasi sull'affidabilità, robustezza e performance. A differenza di ClickHouse supporta le transazioni complete delle proprietà ACID.

6 Analisi dei dati

Ora che è presente un database popolato di tutti i metadati necessari per ricercare i file raw di interesse, il prossimo passo è leggere ed estrarre correttamente i dati dai file.

Il sistema utilizza inoltre le librerie esterne matplotlib[5] e plotly[6] per produrre grafici, grafici di dispersione 2D e 3D, e istogrammi, allo scopo di visualizzare e studiare i dati estratti.

6.1 Architettura generale

Il sistema di analisi dei dati può essere diviso in due parti principali:

- la lettura dei dati dai file raw e dal database. Per questo scopo la classe BinaryReader si occupa della lettura corretta dei file raw (chiamati anche file binari), mentre la classe DataFetcher utilizza BinaryReader e uno dei client di comunicazione con il database visti nella sezione 5.2 per raccogliere i dati di interesse e prepararli alla visualizzazione.
- la visualizzazione dei dati attraverso grafici con la classe DataPlotter che comprende la creazione dei grafici più significativi e l'interazione con i grafici stessi

6.2 Lettura e contenuto dei file raw

I file sono creati attraverso LabVIEW[7], un programma della National Instruments che si occupa di gestire la lettura dei sensori, e non sono pensati per essere aperti con programmi

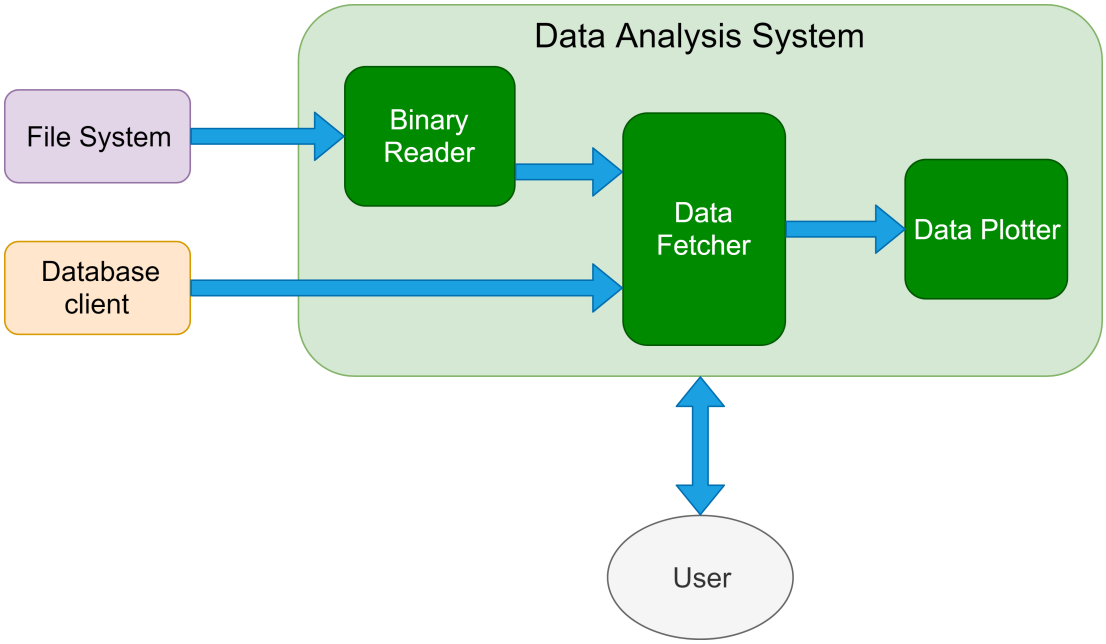

figura 6.1: Architettura generale del sistema di analisi dei dati e le interazioni verso le componenti esterne al sistema stesso.

esterni. Per questo motivo la lettura dei file raw diventa complessa attraverso solo l'uso di Python.

I file raw sono dei file binari che posso essere in uno dei formati proprietari BIN o BIX. Il formato influisce la struttura del file e come sono stati salvati i dati al suo interno.

Il contenuto dei file cambia inoltre in base alla configurazione del Multiphase Flow Meter, di conseguenza ogni BIN e BIX possono essere di tipi diversi, distinti dal campo "BIN type" o "BIX type" all'interno del file.

Ogni file raw contiene un insieme di variabili determinato dal formato e dal tipo del file stesso. Ogni variabile è caratterizzata da:

- `type`: rappresenta il tipo della variabile, ad esempio Int32, UInt32, Float, Boolean.
- `batch size`: rappresenta il numero di letture che vengono effettuate al secondo per quella variabile, può variare da 1 a 2500.
- `data`: un array che contiene le letture effettive del sensore

I dati contenuti in queste variabili corrispondo alle letture dei sensori presenti nel Multiphase Flow Meter al momento del test.

6.3 Grafici e analisi dei dati

L'analisi dei dati comincia dalla scelta una delle cartelle dei test presenti nel server aziendale, si ricorda che ogni test rappresenta una serie di letture effettuate in ambiente controllato in un laboratorio. Scelto uno specifico test è possibile creare un grafico di dispersione in base alle proprietà GVF e WLR (descritte nella sezione 2.3) di tutti i file raw presenti in quella cartella di test. Questa operazione sfrutta le informazioni salvate nel database creato in precedenza in modo da non dover esplorare di nuovo tutta la cartella. Si ottiene quindi un grafico simile a quello in figura 6.2 dove ogni punto rappresenta un file raw ed è posizionato in base alle proprietà GVF e WLR che aveva il flusso nel momento in cui è stata effettuata la lettura. Si può notare che i punti sono per la maggior parte allineati in una struttura a matrice, questo perché le proprietà del flusso sono variate in modo controllato per coprire più casi possibili.

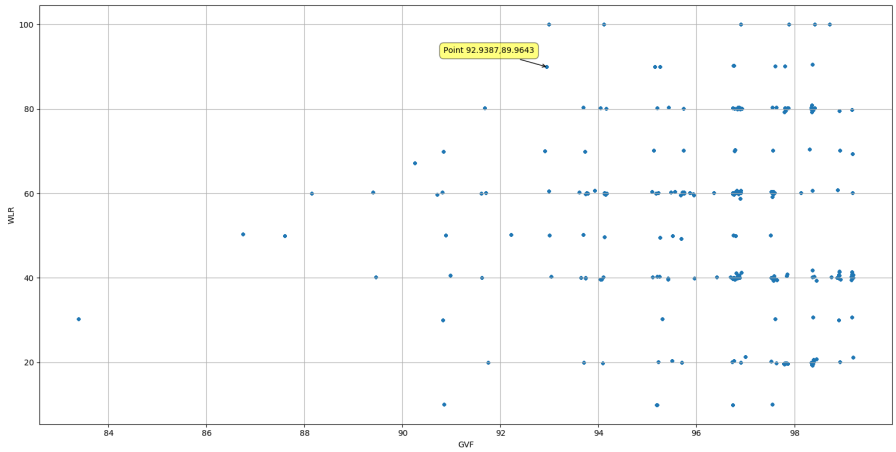

figura 6.2: Grafico di dispersione tra le variabili GVF e WLR, ogni punto rappresenta un file raw ed è posizionato in base alle proprietà GVF e WLR che aveva il flusso nel momento in cui è stata effettuata la lettura.

Cliccando un punto all'interno di questo grafico è possibile vedere graficamente i valori di tutte le variabili del file raw associato a quel punto (figura 6.3). Per motivi di velocità e dimensione i grafici delle variabili sono approssimati e contengono solo il massimo in verde, la media in arancione e il minimo in blu di ogni secondo di lettura.

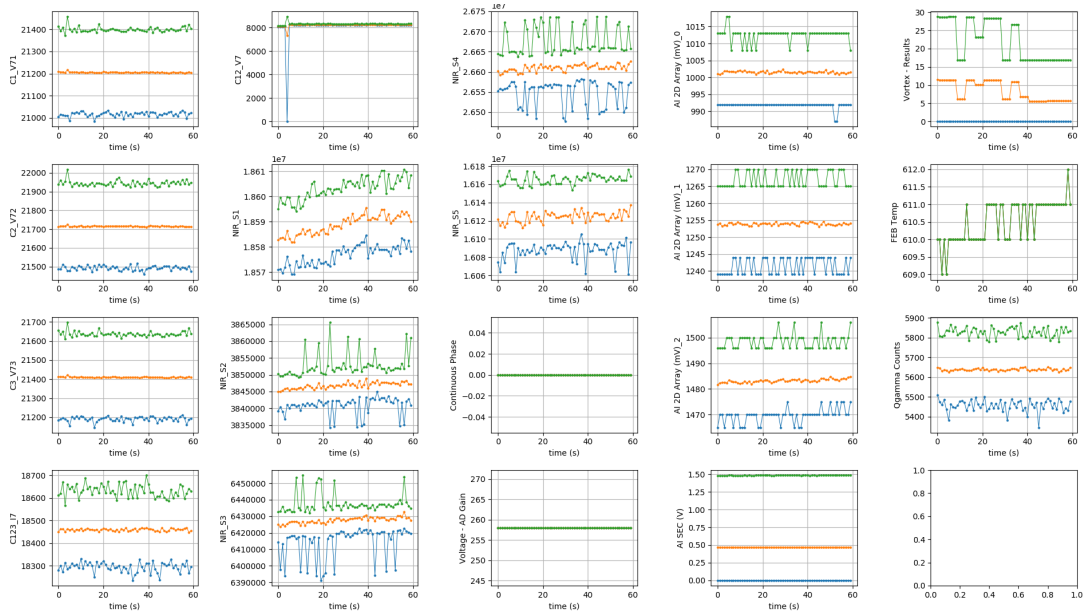

figura 6.3: Grafici delle variabili di un file raw. La linea in verde indica i massimi di ogni secondo di lettura, in arancione le medie, in blu i minimi.

I grafici delle variabili sono selezionabili tenendo premuto il tasto shift o ctrl della tastiera e cliccando da uno a tre grafici. A seconda del numero di grafici selezionati viene mostrato una tipologia di grafico diversa.

Selezionando un solo grafico si visualizza un nuovo grafico contenente tutti i punti di quella specifica variabile, e non solo un'approssimazione. Allineato al grafico dei punti è presente un istogramma orizzontale che mostra più chiaramente quali sono i valori più frequenti per quella variabile.

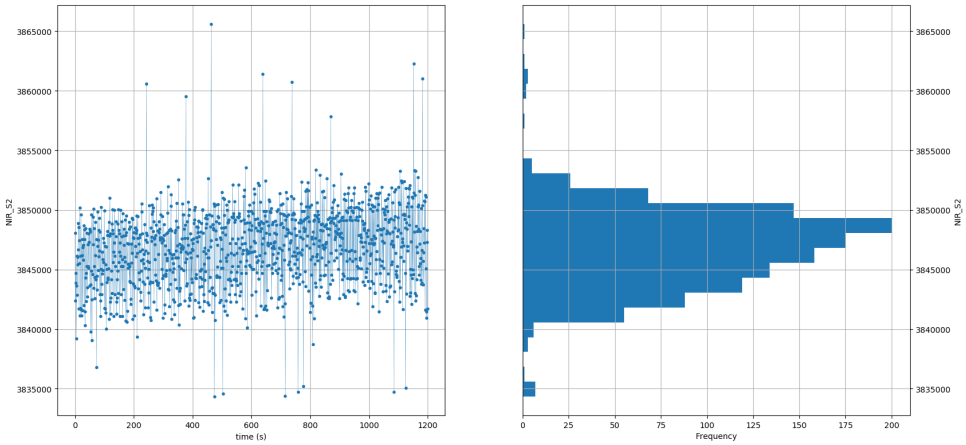

figura 6.4: Grafico e istogramma di una variabile. L'istogramma rappresenta i valori più frequenti.

Selezionando due grafici si visualizza un grafico di dispersione delle due rispettive variabili. Questo grafico può essere utile per controllare se due variabili sono correlate, punti fuori dalla distribuzione potrebbero indicare delle anomalie. Due variabili correlate si riconoscono dalla posizione dei punti nel grafico stesso, se si dispongono lungo una o più linee diagonali sono correlate, se si dispongono in uno o più gruppi sparsi non sono correlate.

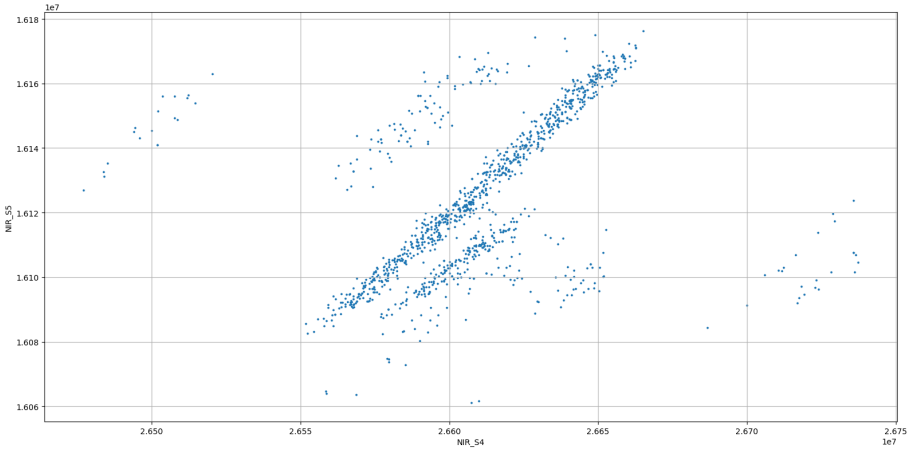

figura 6.5: Grafico di dispersione di due variabili. In questo caso sono si può notare che sono correlate.

Selezionando tre grafici si visualizza un grafico di dispersione in 3D delle tre variabili se-

lezionate, che si può spostare e ruotare a piacimento. Questo grafico è particolarmente utile quando si utilizzano tre variabili correlate tra di loro.

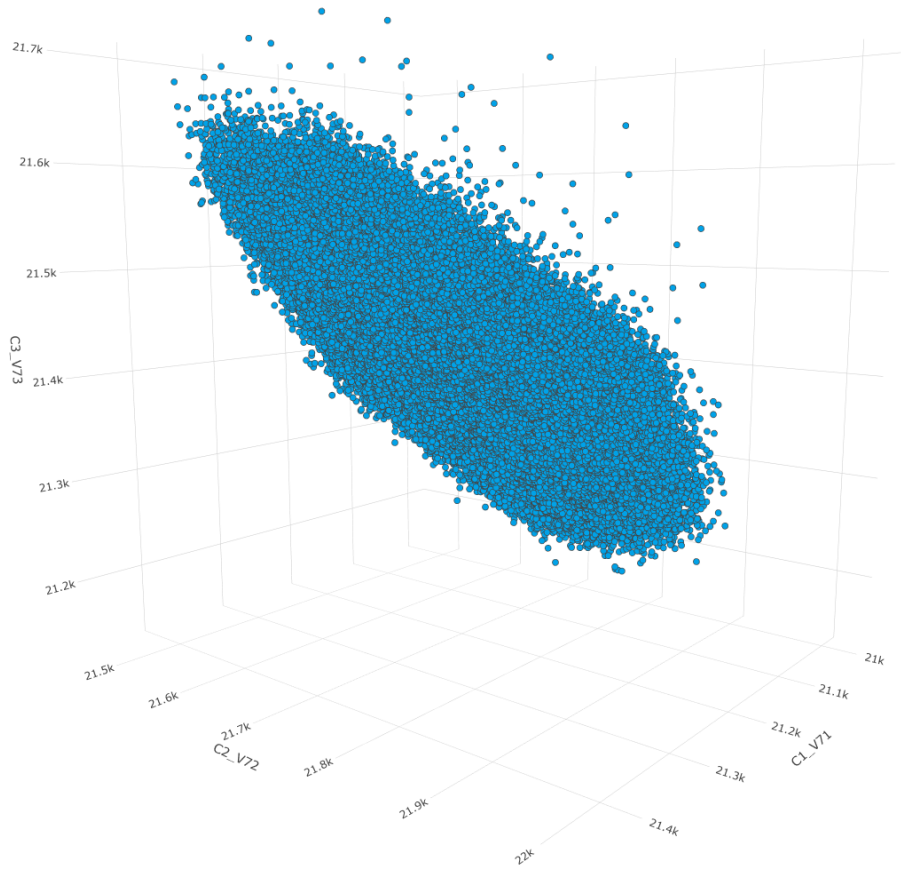

figura 6.6: Grafico di dispersione di tre variabili. In questo caso i punti sono disposti a forma di disco perché le variabili C1_V71 e C3_V73 sono correlate tra loro ma non a C2_V72.

Ogni tipologia di grafico può essere visualizzata usando la libreria matplotlib[5] oppure plotly[6] in base a se si sono selezionai i grafici con shift o con ctrl. Si è lasciata la scelta perché entrambe le librerie hanno dei vantaggi e degli svantaggi a seconda del grafico che si vuole ottenere.

matplotlib è molto veloce nella creazione del grafico ma con molti punti, intorno alle decine di migliaia, rallenta quando si prova a ingrandire, ruotare e selezionare una parte specifica del grafico. plotly impiega un paio di minuti per la creazione del grafico perché genera una pagina HTML indipendente, ma la manipolazione del grafico risulta veloce anche

nell'ordine delle centinaia di migliaia di punti. Inoltre, essendo i grafici file HTML veri e propri, possono essere salvati e visualizzati offline, mentre matplotlib permette solo il salvataggio di un'immagine statica del grafico perdendo l'interattività.

7 Autodiagnostica di anomalie

Questo capitolo tratta l'addestramento di vari algoritmi di machine learning e la valutazione di essi, tratta inoltre il processo di feature extraction applicato ai dati prima di essere utilizzati.

7.1 Feature extraction

Nell'ambito del machine learning con feature extraction si intende l'estrazione di caratteristiche dai dati di ingresso per semplificare in seguito l'apprendimento. L'estrazione di feature è particolarmente utile quando il numero di dati in ingresso all'algoritmo di machine learning è molto alto, e renderebbe l'apprendimento troppo lungo e complesso [8].

Nel caso del Multiphase Flow Meter alcuni dei sensori effettuano fino a 2500 letture al secondo, e in un solo minuto di registrazione si ottengono già 150000 valori per uno solo delle 20 o 30 variabili lette dai sensori presenti nel MPFM. Per questo motivo prima di allenare un algoritmo di machine learning bisogna estrarre alcune feature da questi dati.

7.1.1 tsfresh

tsfresh è la libreria utilizzata per l'estrazione di feature in quanto specifica per le serie temporali, ovvero serie dove i dati sono legati all'istante di tempo in qui sono stati osservati, come per le letture dei sensori del MPFM [9].

tsfresh permette quindi di calcolare automaticamente un certo numero di feature su una serie temporale qualsiasi. Sono presenti più di 100 feature estraibili con tsfresh ma estrarle

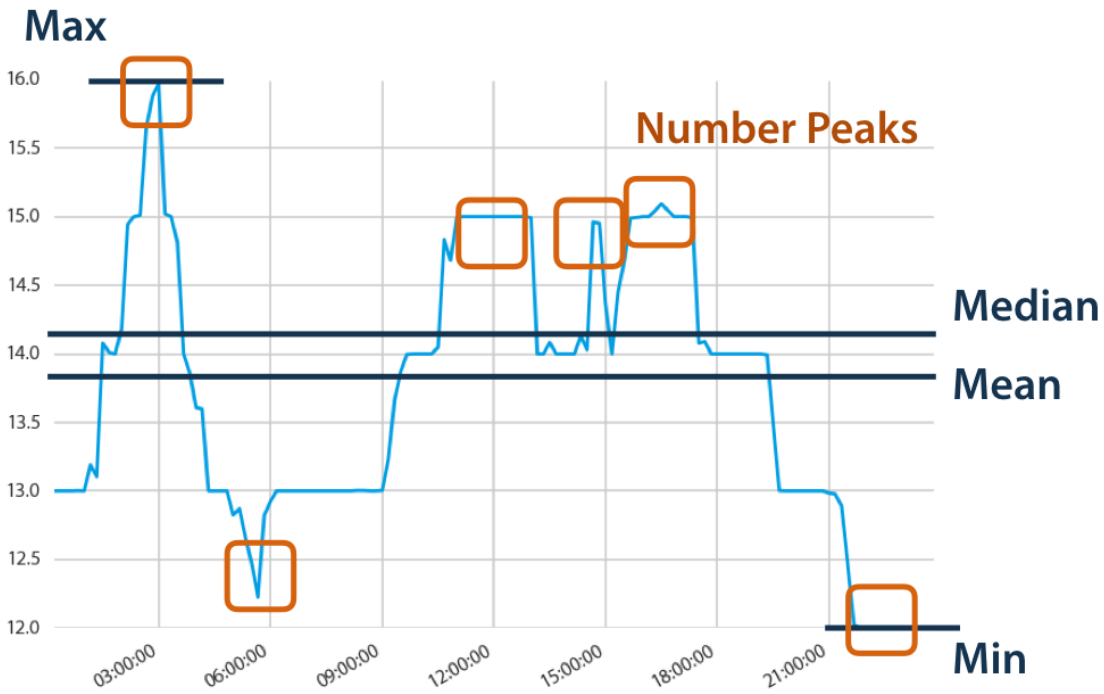

figura 7.1: Esempio di alcune features estraibili da una serie temporale attraverso la libreria tsfresh

tutte richiederebbe troppo tempo e spesso risultano ridondanti in questo contesto.

Si è scelto quindi un set ridotto di feature che comprende solamente il minimo, massimo, media, mediana, deviazione standard e varianza di ogni minuto di dati per ogni sensore. In questo modo il tempo di estrazione si riduce drasticamente, ma rimane comunque notevole quando lo si deve applicare a più file raw per avere un insieme di dati sufficiente per l'apprendimento.

Per non ripetere l'estrazione di feature ogni volta che si vuole applicare un algoritmo di machine learning, è stata effettuata l'estrazione delle feature di tutte le variabili di tutti i file raw una volta sola e il risultato è stato salvato in una tabella apposita nel database. Questo processo ha richiesto circa 8 ore per il completamento, ma una volta presenti nel database le feature si possono ottenere istantaneamente.

7.2 Anomaly detection

L'anomaly detection consiste nella identificazione di osservazioni o eventi che differiscono sostanzialmente dalla maggior parte degli altri dati. Nel contesto del Multiphase Flow meter queste anomalie nei dati possono corrispondere a dei problemi o difetti nello strumento o

nei sensori stessi. Per questo motivo individuare le anomalie attraverso algoritmi di machine learning può essere utile per diagnosticare lo stato dello strumento, e in alcuni casi prevedere quando si sta per rompere prima che accada.

Esistono due tecniche principali di anomaly detection. L'anomaly detection supervisionato richiede che i dati siano classificati come "normali" oppure "anormali" e sfrutta questa proprietà per classificare i nuovi dati dopo l'apprendimento. L'anomaly detection non supervisionato non richiede nessuna classificazione dei dati e assume che la maggior parte dei dati siano normali e cerca i punti meno simili agli altri [10].

Le letture effettuate dei sensori durante i test non sono classificate quindi è necessario usare un approccio non supervisionato.

7.2.1 scikit-learn e PyOD

scikit-learn è una libreria che offre una vasta quantità di strumenti e funzioni per machine learning in Python, tra i quali implementazioni degli algoritmi più comuni, metodi per estrarre feature e normalizzare i dati e molto altro [11]. PyOD invece è una libreria più semplice, specifica per lavorare nel campo dell'anomaly detection e spesso sfrutta alcune delle funzionalità offerte da scikit-learn [12].

Da queste librerie sono stati scelti 5 degli algoritmi più utilizzati per l'anomaly detection e sono stati valutati e confrontati tra loro.

7.2.2 Histogram-Based Outlier Detection (HBOS)

In questo algoritmo viene costruito un istogramma per ogni variabile e la combinazione di tutti gli istogrammi viene utilizzata per determinare quali punti sono "outlier" ovvero anomalie. Questo algoritmo assume che i data normali siano concentrati in una particolare zona e non performa altrettanto bene se i dati sono divisi in diversi cluster. In compenso è uno degli algoritmi più efficienti e classifica i dati in tempo lineare O(n) oppure O(n*log(n)) a seconda se il "bin size" è statico o dinamico [13].

7.2.3 Isolation Forest

Isolation Forest è un algoritmo che si concentra su isolare le anomalie che sono definite come i punti più rari e isolati. In breve una foresta di alberi binari di ricerca è costruita seguendo un approccio che divide casualmente in due lo spazio dei punti. Ad ogni iterazione ogni parte è divisa a sua volta in due fino a quando contiene un solo punto. I punti più isolati

saranno separati dagli altri in un minor numero di tagli e saranno quindi più vicini alla radice dell'albero binario. A tutti i punti viene quindi assegnato un punteggio in base alla loro altezza nell'albero, dove le anomalie avranno punteggio più alto in quanto sono più vicine alla radice e i punti normali un punteggio più basso in quanto più distanti dalla radice [14].

7.2.4 K-Nearest Neighbors (KNN)

Questo algoritmo si basa sulla distanza di ogni punto dai suoi k punti più vicini per determinare gli "inlier" e gli "outlier" ovvero i punti normali e le anomalie. La distanza di un punto dai suoi vicini è usata quindi per dare un punteggio di quanto anomalo è il punto stesso [15].

7.2.5 Local Outlier Factor (LOF)

Local Outlier Factor è un algoritmo basato sul confronto tra la densità di un punto e la densità dei suoi punti più vicini. Considera quindi come outlier i punti che hanno minor densità rispetto ai loro punti più vicini. La località dell'algoritmo permette di riconoscere anomalie che non sarebbero considerate da altri algoritmi, ma questo approccio spesso è applicabile solo a spazi di dati con poche dimensioni [16].

7.2.6 Principal Component Analysis (PCA)

A differenza dell'Isolation Forest questo metodo cerca di classificare gli "inlier" e considera i punti rimanenti come anomalie. Spesso è utilizzato quando si lavora con spazi di dati a molte dimensioni, oppure quando il numero di anomalie è basso e classificarle risulta più difficile di classificare i punti normali. PCA determina quale caratterista ha il maggior impatto sulla varianza dei dati, ovvero determina le componenti principali [17].

7.2.7 Addestramento e confronto

Per l'addestramento degli algoritmi sono state utilizzate le feature estratte in precedenza e caricate nel database. In particolare queste feature sono: massimo, minimo, media, mediana, varianza, deviazione standard di un minuto di dati. Ogni confronto è stato ripetuto per 3 variabili differenti, in particolare sono state utilizzate C1_V71_0 che misura la capacità del fluido, C1_V71_1 che misura la resistenza del fluido, e NIR_S1 che è uno dei sensori che misura il WLR. Per la variabile C1_V71_0 sono stati utilizzati 5369 minuti di dati, questo numero corrisponde quindi al numero di sample usato nell'apprendimento. Per la variabile C1_V71_1

invece sono stati utilizzati 1122 sample, e infine per NIR_S1 7267 sample. La dimensione dei dati dipende dal numero di feature utilizzate, in questo caso è 6 per tutte e tre le variabili.

Non conoscendo quali dei punti sono veramente anomalie, o se i dati non contengono nessuna anomalia, si è deciso di aggiungere del rumore casuale entro il minimo e il massimo dei punti. I punti che formano il rumore possono essere quindi considerati come le anomalie da trovare.

Di conseguenza ad ogni variabile è stato aggiunto un numero di punti casuale pari al 10% del numero di sample iniziale. La valutazione di un algoritmo dipende dal numero di punti aggiunti casualmente classificati correttamente come outlier, e dal numero di punti originali classificati correttamente come inlier.

Variable	#Samples	Classifier	Data			Noise		
			Inliers	Outliers	Inliers %	Inliers	Outliers	Outliers %
C1_V71_0	5369	***HBOS***	***5037***	***332***	***93.82***	***3***	***533***	***99.44***
		IForest	4987	382	92.89	1	535	99.81
		KNN	***4977***	***392***	***92.70***	***0***	***536***	***100.00***
		LOF	4519	850	84.17	527	9	1.68
		PCA	4975	394	92.66	2	534	99.63
C1_V71_1	1122	HBOS	1056	66	94.12	11	101	90.18
		IForest	***1069***	***53***	***95.28***	***0***	***112***	***100.00***
		KNN	1067	55	95.10	1	111	99.11
		LOF	987	135	87.97	97	15	13.39
		PCA	1077	45	95.99	0	112	100.00
NIR_S1	7267	***HBOS***	***6854***	***413***	***94.32***	***0***	***726***	***100.00***
		IForest	6839	428	94.11	0	726	100.00
		KNN	6875	392	94.61	5	721	99.31
		LOF	6307	960	86.79	714	12	1.65
		PCA	6854	413	94.32	0	726	100.00

Dai risultati si può vedere come tutti gli algoritmi siano riusciti a classificare correttamente come outlier praticamente tutto il rumore, tranne per LOF che non ha riconosciuto il rumore come anomalia in nessuna delle tre variabili.

Infine l'addestramento è stato ripetuto con l'aggiunta di rumore non più omogeneamente, ma raggruppato in 5 cluster casuali.

Variable	#Samples	Classifier	Data			Noise		
			Inliers	Outliers	Inliers %	Inliers	Outliers	Outliers %
C1_V71_0	5369	***HBOS***	***4809***	***560***	***89.57***	***0***	***535***	***100.00***
		IForest	4790	579	89.22	0	535	100.00
		KNN	4803	566	89.46	0	535	100.00
		LOF	4359	1010	81.19	528	7	1.31
		PCA	4779	590	89.01	0	535	100.00
C1_V71_1	1122	HBOS	1000	122	89.13	0	110	100.00
		IForest	***1024***	***98***	***91.27***	***0***	***110***	***100.00***
		KNN	976	146	86.99	0	110	100.00
		LOF	950	172	84.67	105	5	4.55
		PCA	1004	118	89.48	0	110	100.00
NIR_S1	7267	***HBOS***	***6847***	***420***	***94.22***	***0***	***725***	***100.00***
		IForest	6793	474	93.48	0	725	100.00
		KNN	6844	423	94.18	0	725	100.00
		LOF	6181	1086	85.06	708	17	2.34
		PCA	6818	449	93.82	0	725	100.00

In questo caso tutti gli algoritmi hanno performato addirittura meglio, probabilmente perché il cluster erano troppo distanti dalla maggior parte dei dati e troppo poco densi per confondere gli algoritmi basati sulla densità. LOF non riesce ancora a distinguere correttamente il rumore, forse per una configurazione sbagliata nel momento dell'apprendimento.

7.3 Visualizzazione delle anomalie

Per verificare il funzionamento degli algoritmi, e avere un'idea sulla motivazione delle scelte effettuate dagli algoritmi nel momento della classificazione, sono stati creati dei grafici che mostrano i dati iniziali e la loro classificazione dopo l'addestramento.

Nei seguenti grafici è stato utilizzato l'algoritmo HBOS addestrato sulla variabile C1_V71_0. L'addestramento comprende tutte e 6 le feature utilizzate in precedenza, ma per poter creare un grafico in due dimensioni sono state utilizzate la deviazione standard nelle ascisse e la media nelle ordinate.

7.3.1 Visualizzazione degli outlier

Nel grafico in figura 7.2 i punti in rosso sono stati classificati come outlier, mentre quelli in verde come inlier. Anche se a prima vista il numero di outlier sembra molto maggiore degli inlier, dall'istogramma si può vedere che è esattamente il contrario. La maggior parte dei

punti è altamente concentrata in una piccola zona, i punti lontani da questa concentrazione sono stati quindi classificati come anomalie.

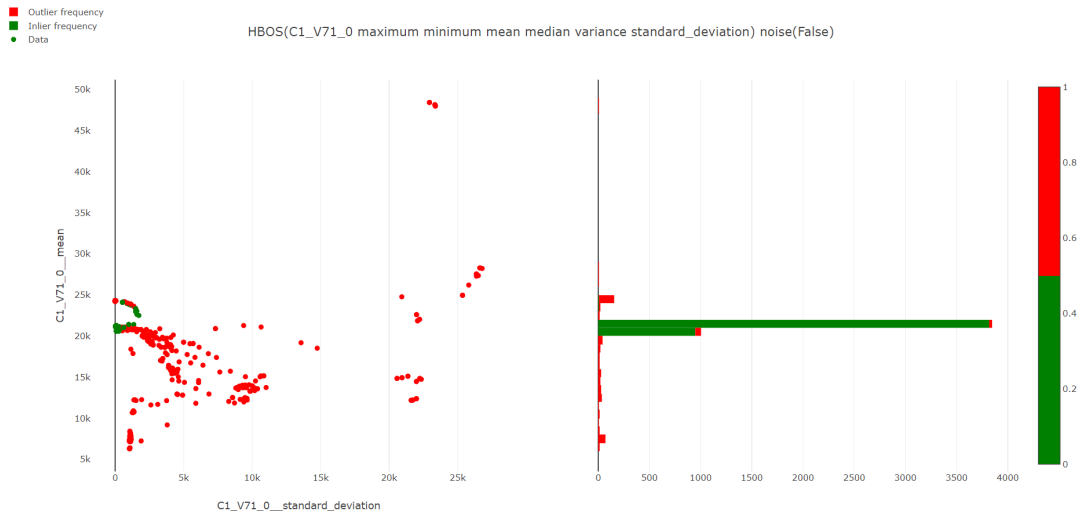

figura 7.2: Grafico dell'anomaly detection nella variabile C1_V71_0 con HBOS. In rosso gli outlier, in verde gli inlier.

7.3.2 Visualizzazione dello score dei punti

A ogni punto viene assegnato uno punteggio o score in base dall'algoritmo utilizzato, questo punteggio è un valore da 0 a 1 che determina quanto sicuro è l'algoritmo nella classificazione del punto. In figura 7.3 si può vedere lo score di ogni punto in base al colore, i punti vicino al giallo sono più probabilmente outlier mentre i punti vicino al blu scuro sono più probabilmente inlier. In questo caso il colore dell'istogramma è irrilevante.

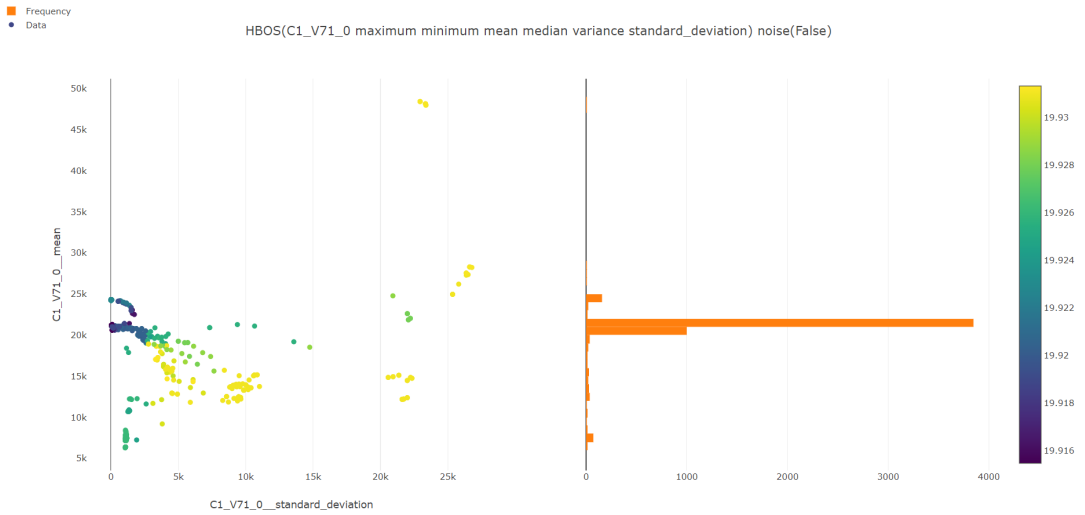

figura 7.3: Grafico dello score dei punti usando HBOS e la variabile C1_V71_0. I punti verso il giallo sono più probabilmente outlier, mentre i punti verso il blu scuro sono più probabilmente inlier.

7.3.3 Visualizzazione dello score con rumore omogeneo

Nel grafico in figura 7.4 è stato aggiunto casualmente del rumore pari al 10% dei punti iniziali. I dati iniziali sono rappresentati con dei pallini mentre il rumore con dei triangoli, tutti i punti sono colorati in base allo score assegnato dall'algoritmo.

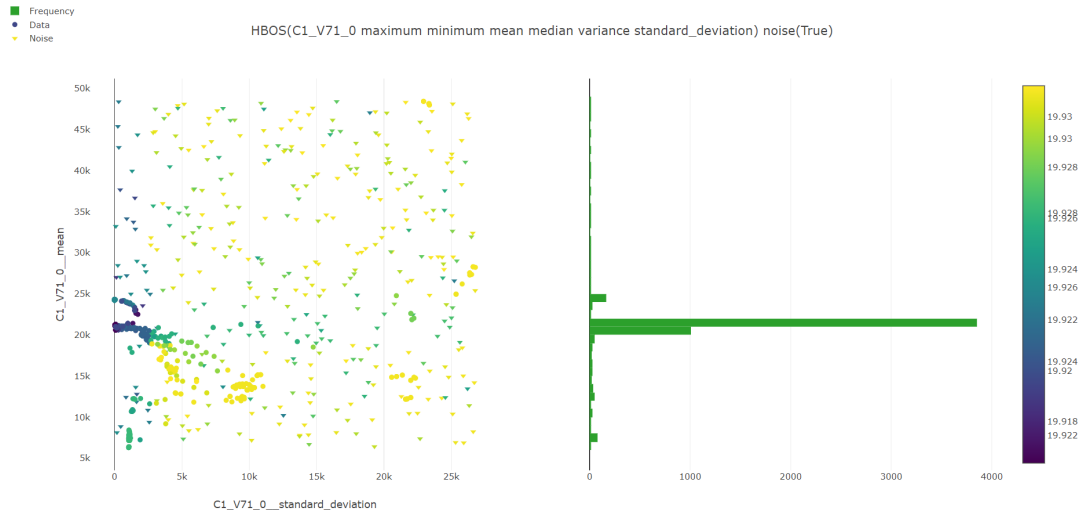

figura 7.4: Grafico dello score dei punti e del rumore usando HBOS e la variabile C1_V71_0. I punti verso il giallo sono più probabilmente outlier, mentre i punti verso il blu scuro sono più probabilmente inlier.

7.3.4 Visualizzazione degli outlier con rumore a cluster

Nella figura 7.5 il grafico mostra la classificazione degli outlier con l'aggiunta del rumore in 5 cluster casuali. Questa immagine conferma le conclusioni tratte alla fine della sezione 7.2.7, ovvero i cluster sono troppo distanti dai dati originali e troppo poco densi per poter confondere gli algoritmi di apprendimento.

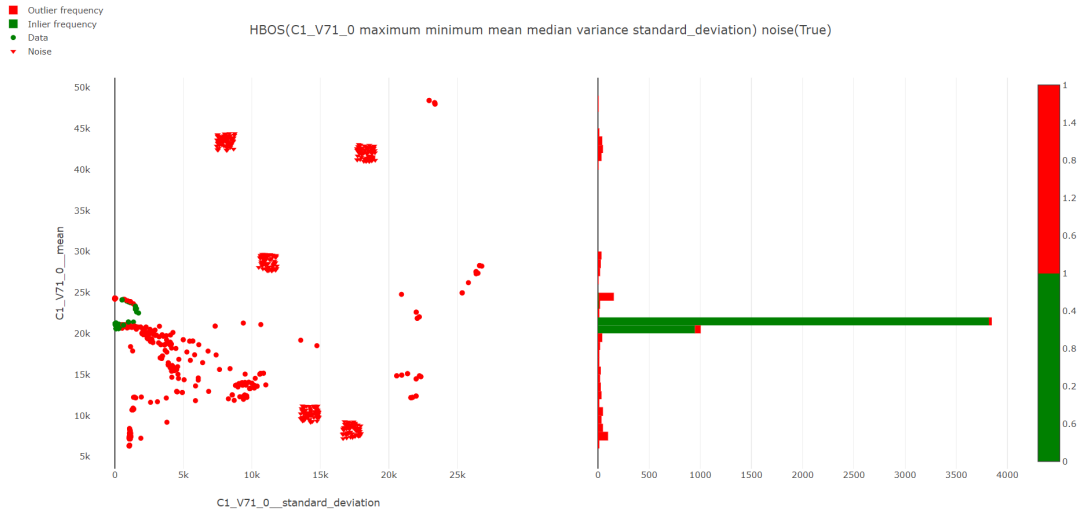

figura 7.5: Grafico degli outlier con aggiunta di rumore in 5 cluster casuali usando HBOS e la variabile C1_V71_0. In rosso gli outlier, in verde gli inlier.

7.3.5 Visualizzazione della decision surface

La decision surface divide la superficie in più zone, ognuna delle quali rappresenta la probabilità che un punto in quella zona sia un'anomalia. Come si vede in figura 7.6 le zone rosse indicano che un punto in quelle zone è molto probabilmente un'anomalia, punti nelle zone blu sono comunque classificati come outlier ma è probabile che non lo siano.

La linea tratteggiata rossa rappresenta la "decision boundary" ovvero il perimetro all'interno del quale i punti sono classificati come inlier. Per visualizzare meglio la decision surface, in figura 7.6 viene mostrata solo una parte del set di dati usato.

Inoltre si possono notare le differenze tra i cinque algoritmi in base alle decision surface che generano, ad esempio PCA estrae le due componenti principali e forma quindi un ovale, oppure HBOS genera una superficie squadrata perché considera gli istogrammi di frequenza dei due assi.

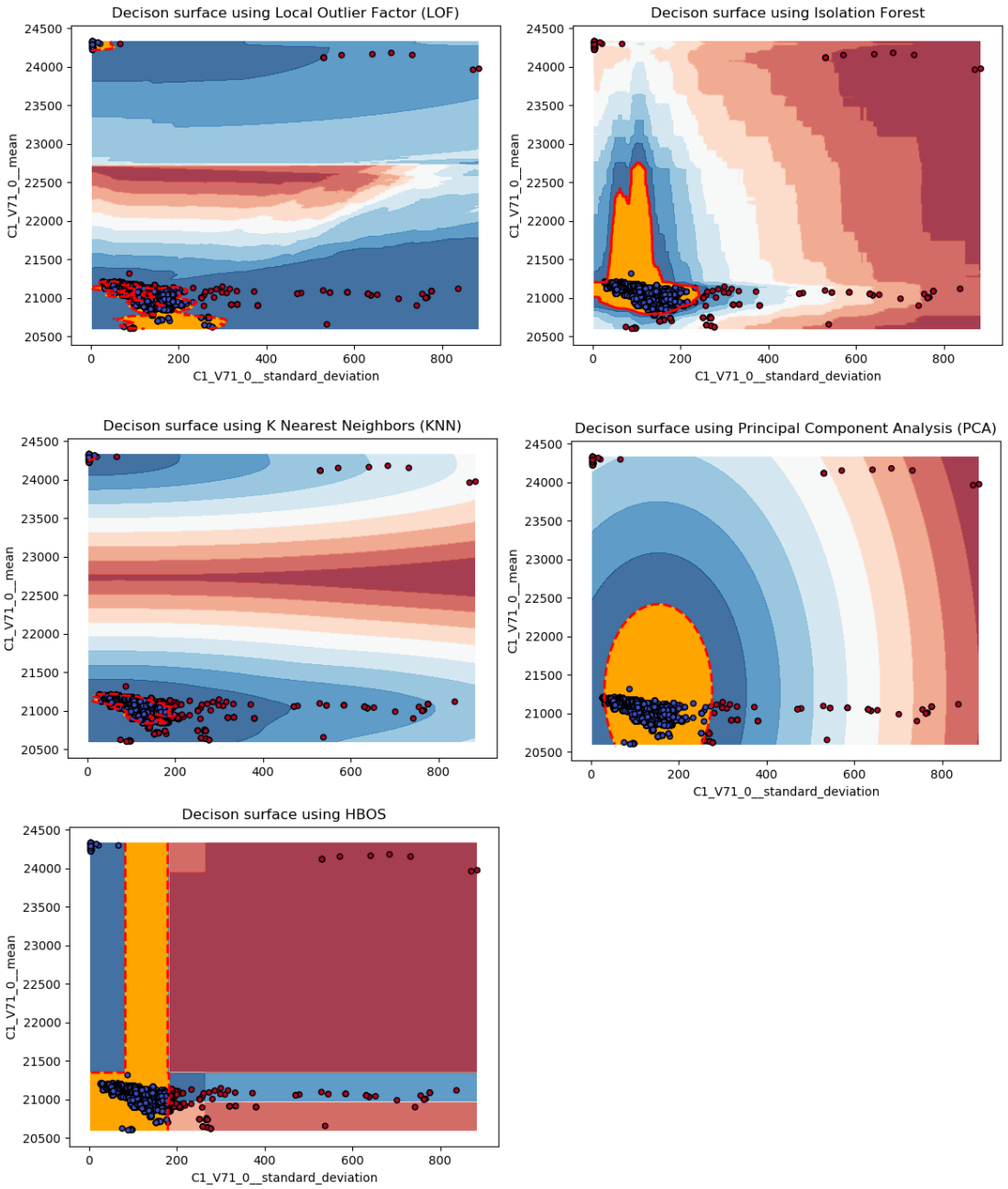

figura 7.6: Decision surface dei cinque algoritmi, le zone rosse indicano che un punto in quelle zone è molto probabilmente un'anomalia, punti nelle zone blu sono comunque classificati come outlier ma è meno probabile che lo siano. I punti all'interno della linea tratteggiata rossa sono inlier.

8 Conclusioni

Lo stage si è svolto seguendo ogni settimana il piano di lavoro descritto nella sezione 3.1. Il lavoro è stato ben pianificato dal tutor aziendale e in ogni parte del progetto non ci sono stati ritardi nelle consegne del software o dei documenti. Il tempo, infatti, è stato sufficiente anche per studiare le tecnologie utilizzate e le possibili alternative, per poterle confrontare ed effettuare la scelta migliore. Il codice segue i vincoli di stile scelti ed è stato progettato per essere il più possibile scalabile ed estendibile. Ogni classe e metodo è stato commentato, dai commenti è stata generata automaticamente la documentazione attraverso Sphinx. Grazie ai documenti prodotti e alla documentazione del codice il progetto è facilmente utilizzabile ed estendibile dai ricercatori dell'azienda. Tutti i requisiti individuati sono stati soddisfatti.

Le ultime due settimane, come da piano di lavoro, sono servite come introduzione all'intelligenza artificiale ed è stato possibile studiare vari algoritmi di feature extraction e di anomaly detection. Per addestrare un algoritmo di machine learning in grado di effettuare anomaly detection efficacemente su un problema ampio e complesso come la misurazione dei flussi multifase[1], sarebbero stati necessari mesi di studio e di lavoro aggiuntivo, chiaramente fuori dai tempi dello stage. I risultati ottenuti sono comunque un ottimo studio di fattibilità sul problema, anche se sono state utilizzate solo alcune delle variabili lette dai sensori e non sono stati ancora tenuti in considerazione i possibili regimi di scorrimento del flusso descritti in sezione 2.2.

Un approccio diverso al problema potrebbe essere effettuare dei test in cui viene manomesso o modificato intenzionalmente il Multiphase Flow Meter per simulare un'anomalia. In questo

modo i dati possono essere classificati come ”normali” quando sono letti con il MPFM funzionante e come ”anormali” quando sono letti con il MPFM modificato. Avere i dati già classificati permette di utilizzare algoritmi di machine learning basati sull'apprendimento supervisionato e ottenere un risultato probabilmente più preciso.

Bibliografia

[1] N. M. System. (2019) An introduction to multiphase flow measurement. [Online]. Available: https://www.tuv-sud.co.uk/uploads/images/1546619713906923071361/an-introduction-to-multiphase-flow-measurement.pdf

[2] Oracle. (2019) Mysql. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/

[3] Yandex. (2019) Clickhouse. [Online]. Available: https://clickhouse.yandex

[4] PGDG. (2019) Postgresql. [Online]. Available: https://www.postgresql.org/

[5] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[6] P. T. Inc. (2015) Collaborative data science. Montreal, QC. [Online]. Available: https://plot.ly

[7] N. Instruments. (2019) Labview. [Online]. Available: https://www.ni.com/it-it/shop/labview.html

[8] S. Khalid, T. Khalil, and S. Nasreen. (2014) A survey of feature selection and feature extraction techniques in machine learning. [Online]. Available: https://www.researchgate.net/publication/265727419_A_Survey_Of_Feature_Selection_And_Feature_Extraction_Techniques_In_Machine_LearningSAI2014l

[9] M. Christ, N. Braun, J. Neuffer, and A. W.Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218304843

[10] D. Shipmon, J. Gurevitch, P. Piselli, and S. Edwards, "Time series anomaly detection; detection of anomalous drops with limited features and sparse examples in noisy highly periodic data," 08 2017.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019. [Online]. Available: http://jmlr.org/papers/v20/19-011.html

[13] M. Goldstein and A. Dengel. (2012) Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. [Online]. Available: https://pdfs.semanticscholar.org/5cf8/81d1db19834f123fcfc79ad32097aeafe17f.pdf

[14] F. T. Liu, K. M. Ting, and Z.-H. Zhou. (2008) Isolation forest. [Online]. Available: https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf

[15] K. D. Borne. (2010) Effective outlier detection using k-nearest neighbor data distributions. [Online]. Available: https://pdfs.semanticscholar.org/f3eb/4573d3164345063351979c9409014ec33d4d.pdf

[16] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. (2000) Identifying density-based local outliers. [Online]. Available: https://www.dbs.ifi.lmu.de/Publikationen/Papers/LOF.pdf

[17] A. S. Deepthi and K. Rao, "Anomaly detection using principal component analysis," *IJCST*, vol. 5, no. 4, pp. 124–126, 2014. [Online]. Available: http://www.ijcst.com/vol54/1/28-Adathakula-Sree-Deepthi.pdf