Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

Riconoscimento e Classificazione di Anomalie in un Contesto di Industry 4.0

Relatore:
Dott.
Luca Bedogni

Presentata da:
Filippo Morselli

Correlatore:
Ing.
Simone Galasso

Sessione III
Anno Accademico 2017/2018

Indice

Capitolo 1

Introduzione

Sui costi complessivi per la produzione di beni, quelli legati alla manutenzione costituiscono dal 15% fino al 40% [1].
In particolare la fonte di perdita principale risiede in tutte quelle attività di manutenzione non prefissate, ossia quegli interventi che hanno lo scopo di risolvere problematiche relative a guasti delle macchine improvvisi e bloccanti nei confronti della produzione.
Analisi dei costi di manutenzione indicano infatti che uno stesso intervento comporta un costo circa triplicato quando viene eseguito in modo inatteso rispetto a quando invece viene programmato in anticipo [2].

Non sorprende quindi il fatto che con l'avvento dell'*Industry 4.0*, cioè quella che ci si attende essere la quarta rivoluzione industriale, uno dei settori su cui si sta investendo e ricercando maggiormente sia proprio quello della manutenzione.
L'obiettivo che ci si pone è quello di riuscire a dedurre lo stato di salute delle macchine e dei loro componenti critici, ed usare queste informazioni per pianificare quando intervenire con sostituzioni o riparazioni.

I guadagni attesi sono molteplici:

- Si evitano guasti improvvisi dovuti a componenti usurati e i costi che

derivano dalle le conseguenti manutenzioni correttive;

- Si massimizzano i tempi in cui le macchine sono in funzionamento, aumentando la produzione e l'efficienza;
- Si risparmia sui componenti, sfruttando ciascuno di essi il più possibile, sostituendolo solamente quando ormai la sua vita utile residua è prossima allo zero;
- Si riduce l'occupazione del magazzino con parti di ricambio, che possono essere ordinate quando il livello di degrado di macchine o componenti supera una certa soglia.

Le tecnologie abilitanti dell'industria 4.0 promettono di prestarsi molto bene all'implementazione di soluzioni per un tale problema.
Lo schema generale di risoluzione infatti sembra consolidato: utilizzare sensori posizionati sulle macchine per raccogliere informazioni rilevanti, comunicare in tempo reale queste informazioni a un centro di aggregazione e analisi, ed applicare modelli di machine learning per estrapolare informazioni utili riguardo lo stato di salute delle macchine, che verrà utilizzato a supporto delle decisioni e delle pianificazioni degli interventi di manutenzione.

Tuttavia sono altrettanto note alcune problematiche e difficoltà che si incontrano quando si passa all'effettiva implementazione in scenari reali di questi sistemi. Fra esse ci sono le seguenti:

- L'alta percentuale di rumore presente nei dati raccolti da sensori in un contesto come quello della produzione industriale;
- Il costo computazionale richiesto per applicazioni di machine learning su una grande mole di dati;
- La necessità di uno storico di dati con l'associazione al relativo stato di salute da cui il modello riesce ad effettuare l'apprendimento e la

forte dipendenza dal fattore umano che deve certificare la correttezza di questa etichettatura.

In questa tesi si discute il problema e si presenta una possibile soluzione.

Obiettivo e Contributo

L'obiettivo del progetto è quello di dimostrare le possibilità e i vantaggi dell'applicazione delle tecnologie di Internet of Things e machine learning in ambito industriale e presentare uno dei servizi innovativi che si rendono possibili.

Nello specifico il contributo di questa tesi è la progettazione e la realizzazione di un sistema di riconoscimento e classificazione di anomalie e guasti.
Il sistema è applicato ad un macchina funzionante, su cui è possibile intervenire facilmente in modo da portarla da condizioni di normale attività a stati di anomalia e viceversa.
La soluzione si basa su dati relativi a differenti grandezze fisiche provenienti da sensori posizionati sulla macchina per dedurne lo stato di salute. La sensoristica utilizzata è composta da componenti a basso costo e di facile reperimento.
Il modello di learning proposto è in grado di rilevare un stato di anomalia richiedendo in fase di training solamente esempi di dati raccolti durante il funzionamento in stato normale e permette inoltre ad un manutentore di associare a segnali di input una particolare tipologia di anomalia a posteriori, in modo da rendere poi possibile una classificazione di quel tipo di anomalia specifico nel caso si verifichi in futuro.
Il modello si è dimostrato anche in grado di distinguere, per uno stesso guasto, diversi livelli di stato di avanzamento, aumentando il grado e l'utilità delle informazioni utilizzabili in fase di pianificazione degli interventi di manutenzione.

Il progetto di tesi è stato svolto in collaborazione con *Bonfiglioli Consulting Srl*, una società di consulenza aziendale con sede a Casalecchio di Reno (Bo) e con la startup *Digibelt*, fondata dal gruppo Bonfiglioli, che si occupa di innovazione ed applicazioni per l'industria manifatturiera.

Struttura dell'Elaborato

Il resto dell'elaborato è strutturato nel modo seguente.

Nel secondo capitolo si fornisce una descrizione del contesto in cui si colloca il progetto di tesi, presentando quindi i concetti di Industry 4.0 e di manutenzione.

Nel terzo capitolo si riportano i lavori correlati, evidenziando i possibili approcci, le problematiche comuni e le soluzioni proposte dalla letteratura per realizzare sistemi di rilevazione, classificazione e previsione di guasti di macchinari.

Nel quarto capitolo si analizzano l'architettura complessiva e i singoli componenti della soluzione proposta.

Nel quinto capitolo si approfondiscono i modelli di machine learning utilizzati per effettuare il riconoscimento di anomalie e la classificazione di guasti e si riportano i risultati ottenuti.

Nel sesto capitolo si descrive un'applicazione che sfrutta i risultati ottenuti dalla soluzione proposta per ottimizzare le attività di manutenzione.

Infine nel settimo capitolo si presentano le conclusioni ed i possibili sviluppi futuri.

Capitolo 2

Contesto

2.1 Industry 4.0

L'industria è quel settore dell'economia che si occupa della produzione di beni materiali, in modo meccanizzato o automatizzato. Nel corso degli anni l'innovazione tecnologica ha comportato grandi e rapidi cambiamenti nell'ambito industriale, portando al verificarsi di quelle che sono state definite come rivoluzioni. L'introduzione delle macchine a vapore, la produzione di massa grazie a elettricità e catena di montaggio e la robotica e automatizzazione hanno guidato le prime tre rivoluzioni.
Con il termine *Industry 4.0* si intende la potenziale quarta rivoluzione industriale, indicando cioè la trasformazione dell'impianto di produzione in *smart factory*, dove l'intelligenza a cui si fa riferimento della definizione risiede nell'autonomia, nell'ottimizzazione e nell'interconnessione delle sue varie parti.

Le idee e i principi fondamentali del concetto di industria 4.0 sono stati inizialmente proposti da Kagermann, Lukas e Wahlster in un articolo del 2011, per poi essere formalizzati nel 2013 dal governo tedesco in un apposito manifesto, che si pone alla base di un progetto di investimenti finalizzato alla modernizzazione del sistema produttivo. Dalla Germania, il modello si è poi velocemente esteso al resto del mondo, fino ad essere seguito da numerosi

altri paesi.
Per descrivere meglio in cosa consiste l'industria 4.0 si riportano di seguito sia le tecnologie abilitanti che i principali vantaggi derivanti da esse.

2.1.1 Tecnologie Abilitanti

Cyber-Physical Systems (CPS)

I sistemi ciberfisici (o CPS) sono sistemi che permettono l'integrazione di componenti digitali con processi fisici, creando un unico processo che coinvolge elementi appartenenti a entrambi i domini. Essi si basano sull'utilizzo congiunto di capacità computazionali, capacità di interconnessione e capacità di controllo e di attuazione, ossia di intervento sul mondo fisico. La componente umana si integra nel sistema tramite apposite interfacce avanzate di comunicazione uomo-macchina, che permettono interazioni di tipo vocale, visuale o tattile.
Resi possibili in particolar modo tramite dispositivi embedded e ai principi dell'Internet of Things, sono usati in contesti Industry 4.0 con fini di supporto e miglioramento di logistica e produzione.

Industrial Internet of Things (IIoT)

L'IIoT è fondamentalmente l'applicazione delle tecnologie dell'Internet of Things in uno scenario industriale. Con il termine si definisce pertanto una rete di oggetti fisici dotati di componenti che abilitano la comunicazione e l'interazione fra essi e le persone oppure direttamente fra loro.
Congiuntamente all'uso di sensori, è possibile implementare un processo autonomo di raccolta e scambio di informazioni riguardo il mondo fisico, che rende possibile numerosi servizi innovativi, specialmente se integrati con funzioni di intelligenza artificiale e CPS.
Sensoristica e connettività su macchinari industriali significa poter abilitare il monitoraggio delle condizioni in qualunque momento e anche da remoto.

Cloud Computing

Il cloud computing è definito come un modello on demand di accesso tramite la rete a servizi e risorse, fra cui rientrano l'elaborazione, l'archiviazione e la trasmissione dati.
Una proprietà del cloud è che le risorse sono fornite da un insieme complessivo messo a disposizione dal provider, che vengono allocate e rilasciate a clienti in base alle richieste e alle necessità, tipicamente seguendo un modello di business "pay-as-you-use".
Il vantaggio principale di tale soluzione per le aziende è proprio nella possibilità di ottenere servizi e risorse necessarie per abilitare le applicazioni in contesto Industry 4.0 senza dover effettuare grossi investimenti per acquistarle fin da subito, e poter aggiustare in base all'utilizzo i volumi necessari.

Big Data

Con big data si indica la raccolta di un flusso di dati tale che in termini di volume, velocità e varietà risulta essere non gestibile da un tradizionale sistema di database relazionale.
I dati in questione possono riguardare attività, eventi, valori di sensori e stato delle macchine. Il vero vantaggio viene fuori però solamente in fase di analisi di questi dati, in cui si cercano correlazioni non note a priori fra eventi e variabili monitorate. L'analisi è quindi il momento in cui i dati vengono tradotti in informazioni, che a loro volta forniscono supporto durante le fasi di decisione.
La persistenza e l'analisi di tale mole di dati richiedono apposite soluzioni per essere implementate e tipicamente si appoggiano a servizi cloud.

Machine Learning

Il machine learning è lo studio e la realizzazione di algoritmi e modelli che permettono a sistemi di elaborazione di svolgere determinati task senza essere esplicitamente programmati per farlo, ma apprendendo tramite ricono-

scimento di pattern e inferenza.
Il vantaggio principale dell'uso di tecniche di learning risiede proprio nel fatto che possono essere utilizzati con successo per risolvere problemi che difficilmente riescono a essere approcciati con una tradizionale sequenza di istruzioni, anche con il contributo di esperti di dominio. In un contesto come l'Industry 4.0 in cui la quantità di dati potenzialmente utili è molto elevata, il machine learning diventa uno strumento fondamentale.

Tipicamente gli algoritmi di learning non usano come input direttamente il dato grezzo, ma dei valori derivati da essi, detti *feature* che fungono da prima fase di elaborazione del dato. L'estrazione delle feature serve per facilitare il compito del modello, riducendo il numero delle variabili di input ma preservandone le informazioni e le caratteristiche, ma anche per favorire l'interpretabilità.
Con apposite tecniche, anche la costruzione e la selezione delle feature può essere effettuata automaticamente.

Gli approcci di machine learning si dividono in:

- *Supervisionato*: rientrano in questa categoria gli algoritmi che costruiscono un modello a partire da un insieme di dati che per ogni elemento di input contiene il corrispettivo valore di output; il modello così realizzato potrà poi essere usato per determinare l'output di nuovi elementi di input, non noti in fase di training;

- *Non Supervisionato*: è l'insieme degli algoritmi che, a partire da un insieme di dati che contiene per ogni elemento solamente l'input, cercano di estrarre strutture e pattern comuni per raggrupparli ed identificarli in base a criteri di similarità;

- *Semi Supervisionato*: sono quelle tecniche che effettuano il training su un dataset in cui solamente a una parte dei dati è associato il loro output; permette di mostrare all'algoritmo quali sono i possibili output

senza dover etichettare tutto il dataset e comunque riporta diversi altri esempi di input anche se non classificati.

2.1.2 Prospettive e Vantaggi

L'applicazione delle tecnologie illustrate al settore industriale promette di portare i seguenti benefici.

- Il primo vantaggio è l'*integrazione* fra i processi che avvengono durante tutta la catena di produzione, permettendo la digitalizzazione e la conseguente ottimizzazione delle attività che vanno dalla logistica interna fino alla vendita.

- L'*interoperabilità* fra sistemi, anche fra compagnie differenti permette la creazione di una rete che collega e semplifica il passaggio di informazioni fra tutti i partner che collaborano alla produzione di un certo bene.

- La capacità di poter raccogliere ed archiviare dati relativi a ogni aspetto delle macchine e della produzione e la possibilità per operatori e manager di poter accedere ad essi in qualunque momento fornisce un grande supporto alle attività di *decision-making*. Metodi di analisi e intelligenza artificiale semplificano ulteriormente il compito, fornendo ulteriori informazioni estratte dai dati.

- Grazie all'automatizzazione e alla digitalizzazione si ottiene una forte capacità di riconfigurazione e di modularità. Queste caratteristiche permettono di semplificare lo sforzo necessario a soddisfare i singoli bisogni dei clienti. Si rende possibile una *produzione individualizzata* rapida e ottimizzata.

- Viene resa possibile inoltre una migliore collaborazione fra uomo e macchina, grazie ai sistemi ciberfisici e alle interfacce di interazione innovative, che facilitano e supportano gli operatori nei vari compiti che devono eseguire.

2.2 Manutenzione

La manutenzione è definita come la *combinazione di tutte le azione tecniche, amministrative e gestionali, durante il ciclo di vita di una entità, destinate a mantenerla o a riportarla in uno stato in cui possa eseguire la funzione richiesta*[1].

Il concetto e il processo di manutenzione hanno subito nel corso degli anni una forte evoluzione, passando da semplici attività di tipo tecnico eseguite singolarmente quando macchine o strumenti presentavano guasti bloccanti, fino a diventare un complesso sistema fortemente integrato con gli altri processi produttivi e con un importante ruolo strategico.

Come riportato in [1], i costi relativi alla manutenzione possono rappresentare attualmente dal 15% fino al 40% dei costi totali di produzione. Questo sta quindi fortemente spingendo le imprese a mettere a punto apposite strategie di gestione della manutenzione, finalizzata al raggiungimento dell'obiettivo ideale dello *zero downtime*, ossia il riuscire a tenere le linee di produzione sempre in attività, senza doverle fermare per la riparazione di malfunzionamenti.

Fra le attività più significative che permettono di muoversi in questa direzione ci sono l'analisi sistematica dei guasti e delle loro cause, una gestione attenta e pianificata del magazzino per l'immediata disponibilità dei ricambi, l'utilizzo di CMMS (Computerized Maintenance Management System) a supporto dei workflow di manutenzione e l'utilizzo di apposite politiche manutentive.

In particolar modo queste politiche di manutenzione sono ben note e ben definite. I possibili approcci sono riportati e descritti di seguito.

[1] Tutte le definizioni fanno riferimento allo standard UNI EN 13306:2010, ossia la norma europea che elenca e definisce tutti i termini della manutenzione

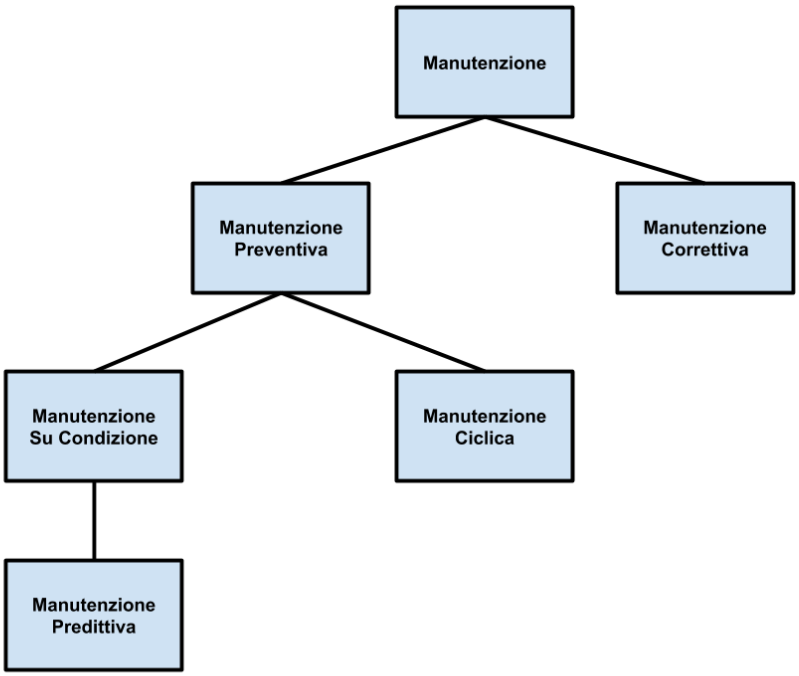


Figura 2.1: Schema degli approcci alla manutenzione

2.2.1 Manutenzione Correttiva

La manutenzione correttiva è *eseguita a seguito della rilevazione di un'avaria*, ossia segue la logica di intervenire con azioni di riparazione solamente quando si verifica un guasto dei macchinari. Per questo è anche definita come reattiva.
E' la strategia più semplice da utilizzare e quella che comporta meno costi per essere attuata, dato che non ci sono spese relative a componenti di ricambio se non quando è strettamente necessario.
Tuttavia i costi associati a questa filosofia sono quelli legati ai tempi di inattività delle macchine, che spesso sono inevitabili quando un qualche guasto si verifica. Questo comporta quindi un arresto nella produzione e un calo complessivo dell'efficienza. Inoltre si aggiunge l'impossibilità di programmare gli interventi, dato che un malfunzionamento si può verificare in un qualsiasi

momento di attività. Infine l'approccio correttivo richiede di avere sempre disponibile in magazzino tutte le parti di scorta dei componenti principali, o quantomeno di quelli critici per la produzione.

2.2.2 Manutenzione Preventiva

Comprende tutte quelle attività di manutenzione *eseguite a intervalli predeterminati o secondo criteri, previste per ridurre le probabilità di guasto o il degrado dell'entità.*
Alla base di questo approccio risiede il fatto che un intervento programmato permette un forte risparmio rispetto al caso reattivo. Analisi sui costi di manutenzione riportano infatti che il costo è circa un terzo rispetto all'intervento correttivo [2].
Le ispezioni e le sostituzioni vengono quindi fatte basandosi su criteri temporali, con lo scopo di intervenire prima che il malfunzionamento avvenga. Gli intervalli temporali sono dedotti da dati storici, individuando la durata media della vita utile di tutti i componenti sottoposti a degrado.
Anche la manutenzione preventiva è tuttavia soggetta a criticità, ed esse sono dovute al fatto che la durata media dei componenti è puramente un indicatore statistico, ma non fornisce alcuna garanzia. Sono possibili infatti sia scenari in cui si sostituiscono componenti in buono stato e ancora funzionanti causando spreco di risorse e sia casi in cui si ricade nella manutenzione correttiva perché altri si guastano molto prima della loro durata attesa.

2.2.3 Manutenzione su Condizione

E' una specializzazione della manutenzione preventiva che *include una combinazione di attività automatiche o manuali di monitoraggio delle condizioni e la loro analisi.*
Ciò significa quindi che prima di agire effettivamente sui componenti si effettua una rilevazione del loro stato di salute e si interviene solamente nel caso in cui venga evidenziata una certa probabilità di rischio.

La rilevazione di una condizione di anomalia si ha quando un qualche parametro fisico della macchina risulta essere non conforme con il normale funzionamento. Esempi tipici sono l'aumento del rumore, delle vibrazioni o della temperatura.
Essa può essere messa in atto sia dall'uomo, ad esempio da manutentori con apposite ispezioni o da operatori esperti che si accorgono delle variazioni durante l'uso, oppure tramite appositi sensori, che servono al monitoraggio continuo dei parametri ritenuti significativi.

2.2.4 Manutenzione Predittiva

La manutenzione predittiva è una ulteriore specializzazione della manutenzione su condizione che *viene eseguita in seguito a una previsione derivata dall'analisi ripetuta o da caratteristiche note e dalla visualizzazione di parametri significativi afferenti il degrado dell'unità.*
Secondo questo approccio, i dati relativi al funzionamento e alle condizioni dei vari componenti sono registrati e salvati in uno storico, per poi essere utilizzati per costruire un trend del comportamento complessivo. Le informazioni così ottenute sono sfruttate per predire l'evoluzione del livello di degrado di un componente e quindi pianificare una relativa attività di manutenzione.
Il vantaggio principale rispetto alla manutenzione su condizione risiede proprio nella parte di analisi del trend e della costruzione di un modello di evoluzione dello stato sulla base dell'esperienza dedotta dell'analisi passata, che permette di riuscire a stimare il tempo di vita utile residua del componente dopo aver rilevato uno scostamento dal normale funzionamento quando ancora è ad una sua prima fase.
Un sistema efficace di predittiva permette di migliorare ed ottimizzare in modo notevole la disponibilità dei macchinari e il tempo passato in produzione, riducendo il numero degli interventi di manutenzione e il loro costo. Inoltre ha effetti positivi anche sulla qualità del prodotto.

2.3 Manutenzione in Contesto Industry 4.0

La manutenzione è un ambito che può beneficiare in modo molto significativo dall'avvento dell'Industry 4.0.
Come descritto nella sezione precedente, gli approcci di manutenzione su condizione (e quindi anche la predittiva), seppur possibili anche appoggiandosi ad ispezioni manuali eseguite dai manutentori, sono più facilmente implementabili attraverso appositi sistemi hardware e software.

Se si analizzano i componenti e le funzionalità necessarie per implementare un sistema di **Condition-Based Maintenance** (**CBM**), si ottengono i seguenti elementi:

- *Sensori*, installati sulle macchine per effettuare misurazioni riguardo i parametri fisici di interesse;
- *Comunicazione*, per poter trasmettere in qualche modo i dati raccolti a un centro di aggregazione;
- *Archiviazione*, per mantenere lo storico dei valori dei sensori, ed eventualmente integrarlo con una lista di eventi e attività;
- *Analisi*, per estrapolare correlazioni tra variabili e stato della macchine, e di conseguenza riconoscere e predire malfunzionamenti.

Si evince pertanto una forte corrispondenza fra essi e i servizi e le tecnologie abilitanti dell'Industry 4.0.

Per quanto riguarda i vantaggi e i benefici provenienti dall'applicazione di tali sistemi, si elencano di seguito i principali.

- Si ha possibilità, tramite la visualizzazione dei dati raccolti dai sensori in tempo reale, di effettuare attività di *monitoraggio continuo* dei macchinari di produzione, anche durante il funzionamento e senza dover ricorrere a ispezioni periodiche dei manutentori;

- La raccolta di dati storici favorisce l'utilizzo di modelli statistici o di machine learning, in modo da aumentare progressivamente le conoscenze e le capacità del sistema, fino a poter individuare la presenza di anomalie fin dallo stadio iniziale e stimare, combinando stato e storico, un trend di degrado;

- Siccome non è possibile riuscire a individuare e prevenire il 100% dei guasti bloccanti, quando essi si verificano un sistema di CBM risulta utile per individuare il componente che ha causato il fermo e il tipo di problema che lo ha interessato, semplificando il compito dei manutentori e riducendo il tempo necessario al ripristino delle attività di produzione;

Oltre a questi vantaggi elencati, caratteristici del sistema di CBM, si hanno ovviamente anche tutti i guadagni in termini di produttività, efficienza, pianificazione e qualità che conseguono dall'applicazione delle politiche di manutenzione su condizione e predittiva che questi sistemi abilitano.

Capitolo 3

Lavori Correlati

I contributi e i risultati della ricerca negli ultimi anni nell'ambito dei sistemi di Condition-Based Maintenance sono stati numerosi.
Essi, come già riportato nel capitolo precedente, sono principalmente dovuti al forte interesse del settore industriale, che vede la manutenzione su condizione e predittiva come una delle applicazioni più profittevoli dell'Industry 4.0.

Dunque, data la vastità dell'argomento in questione, in questo capitolo dell'elaborato si presentano le principali definizioni, i diversi problemi che si vogliono risolvere e le metodologie più rilevanti utilizzate nelle soluzioni proposte in letteratura.

3.1 Definizione del Problema

Sotto la famiglia della CBM rientrano diversi problemi, che però si possono raggruppare in quanto sono finalizzati all'ottenimento degli stessi obiettivi di business, ossia ridurre i costi legati alla manutenzione delle macchine ed aumentare il tempo effettivo passato in produzione.

Cercando di riassumere e di generalizzare la struttura di una applicazione

di CBM, si ha che essa è composta da tre fasi [5], comuni a tutte le differenti implementazioni specifiche. Le fasi sono:

- **Acquisizione Dati**: ossia il processo di raccolta di tutte quelle informazioni che si ritengono rilevanti per poter dedurre lo stato della macchina o dei suoi componenti;

- **Elaborazione Dati**: ossia la gestione e l'analisi dei dati raccolti per poterne fornire un'interpretazione e la loro trasformazione in conoscenza sulla macchina;

- **Decisioni di Manutenzione**: ossia la definizione di una politica di decisioni riguardo alle azioni di manutenzione da eseguire che dipende anche dalle informazioni aggiuntive ottenute tramite lo step di elaborazione.

La distinzione principale all'interno delle applicazioni di CBM è quella fra diagnostica e prognostica.
Lo scopo di un sistema di **diagnostica** è quello di rilevare ed identificare un guasto quando quest'ultimo accade. Nel caso ideale, ciò significa quindi monitorare un sistema, segnalare quando qualcosa non sta funzionando nel modo atteso, indicare quale componente è colpito dall'anomalia e specificare la tipologia di anomalia.
La **prognostica** invece ha come obiettivi quello di determinare se un guasto è prossimo al verificarsi o quello di dedurne la probabilità di accadimento.
Ovviamente, essendo la prognostica un'analisi a priori, può fornire un contributo maggiore per quanto riguarda la riduzione dei costi degli interventi, ma è un obiettivo più complesso da raggiungere.

Un'ulteriore opzione è quella di utilizzare contemporaneamente soluzioni di diagnostica e prognostica applicati allo stesso sistema. La loro combinazione fornisce infatti due preziosi vantaggi:

- La diagnostica permette di intervenire al supporto delle decisioni nei casi in cui la prognostica fallisce; questo scenario è infatti inevitabile, in quanto ci sono guasti che non seguono un modello tale da essere previsti, e anche i guasti che sono prevedibili con buona precisione non riescono a essere individuati nella totalità dei loro accadimenti;
- Le informazioni ottenute tramite applicazioni di diagnostica possono essere utilizzate come input aggiuntivo ai sistemi di previsione, permettendo quindi la realizzazione di modelli più sofisticati e precisi.

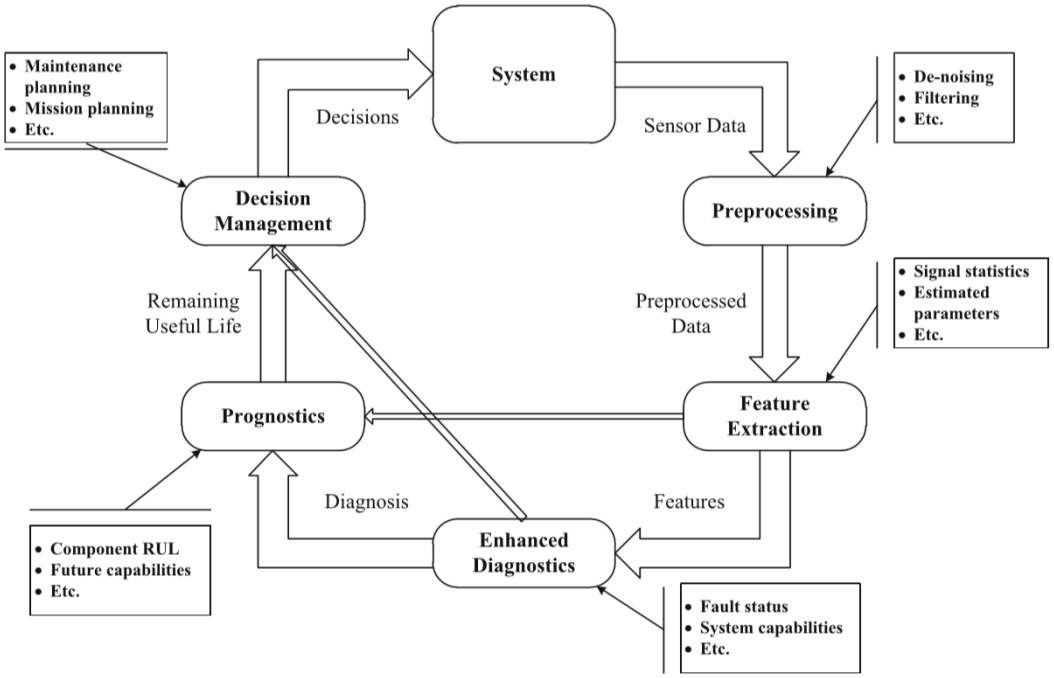


Figura 3.1: Esempio di un sistema di CBM che combina diagnostica e prognostica [7]

3.2 Metodologie di Approccio

Una delle categorizzazioni più utilizzate in letteratura riguardo i sistemi di CBM è quella riguardo la metodologia di approccio utilizzato. E' una

distinzione ad alto livello, in quanto ognuna delle classi riportate è composta a sua volta da differenti modelli specifici.
Esse si differenziano fra loro per caratteristiche quali il costo di applicazione, la complessità, la generalizzabilità, l'accuratezza attesa e la tipologia di input che necessitano per funzionare.
Si ritene utile illustrarle per fornire una visione completa delle strade che che sono state percorse dalla ricerca per risolvere i problemi di diagnostica e prognostica.

3.2.1 Modelli di Tipo Fisico

Il primo possibile approccio sono i modelli di tipo fisico (*physics-based*), che si basano sulla descrizione dell'effettivo processo di degrado dei componenti di interesse delle macchine. Questo significa nello specifico modellare in termini di leggi della Fisica come le condizioni di funzionamento influiscono sull'efficienza e sulla longevità degli asset.
Tra le variabili più rilevanti rientrano svariate grandezze termiche, meccaniche, chimiche ed elettriche. Riuscire a rappresentare come esse abbiano un impatto sulla salute dei macchinari è un compito molto complicato. Pertanto la figura che si occupa di realizzare questo tipo di soluzioni necessita di elevate conoscenze di dominio e di abilità di modellazione.
Una volta realizzato il modello è necessario avere a disposizione dei sensori che permettano di ottenere valori corrispondenti alle grandezze ritenute rilevanti in fase di analisi e modellazione, in modo da utilizzarle come input.

Il principale vantaggio di questa tipologia di approccio è che risulta essere descrittiva, ossia permette di analizzare le motivazioni di ogni output che fornisce, proprio perché si basa su una descrizione fisica del processo. Di conseguenza permette validazione e certificazione [7]. Per quanto riguarda la precisione, essa è fortemente legata alla qualità dell'analisi e della modellazione da parte degli esperti di dominio.
Gli aspetti negativi sono invece la complessità e l'elevato costo di realizzazio-

ne, insieme all'elevata specificità per il sistema, che comportano una scarsa possibilità di riutilizzazione ed estensione.

3.2.2 Modelli Basati su Conoscenza

Anche per la realizzazione di modelli basati su conoscenza (*knowledge-based*) si fa affidamento ad esperti di dominio, in quanto ciò che si vuole modellare con questo tipo di approccio sono direttamente le competenze e il comportamento degli esperti stessi.
L'obiettivo è quello di ottenere una formalizzazione della conoscenza che essi possiedono, in modo da permettere di riprodurla ed applicarla in modo automatico.
I **sistemi esperti** sono infatti programmi che utilizzano basi di conoscenza raccolte a partire da persone competenti in un determinato ambito per poi applicare su di esse meccanismi di inferenza e di ragionamento per emulare il pensiero e fornire supporto e soluzioni a problemi pratici.

Fra gli approcci più comuni per l'implementazione di questo tipo di modelli ci sono i meccanismi basati su regole e la fuzzy logic [8].
I primi hanno come pregio la semplicità di realizzazione e l'interpretabilità, ma possono risultare non sufficienti per esprimere condizioni complicate e possono soffrire di esplosione combinatoria quando il numero delle regole è molto elevato.
L'uso della fuzzy logic permette di descrivere lo stato del sistema tramite input più vaghi e imprecisi, rendendo più semplice e intuitivo il processo di formalizzazione e di descrizione del modello.
Anche per i sistemi esperti, come per i metodi di tipo fisico, i risultati sono fortemente dipendenti dalla qualità e dal livello di dettaglio raggiunto dal modello e sono altamente specifici.

3.2.3 Modelli Basati sui Dati

I modelli basati sui dati (*data-driven*) applicano tecniche di statistica o di learning ai dati raccolti relativi alle macchine, con lo scopo di poter poi riconoscere lo stato dei componenti. L'idea è quella di riuscire ad ottenere il maggior numero di informazioni riguardo lo stato dei macchinari in tempo reale, tipicamente tramite sensori e dai log delle attività di produzione e di manutenzione, e di correlarli con il livello di degrado dei singoli componenti o con le performance del sistema nel suo complesso.
Da una analisi della letteratura disponibile sulla CBM si evince che questo tipo di approccio risulta attualmente essere il più approfondito dai ricercatori e il più utilizzato in casi pratici. Le motivazioni sono le seguenti [7, 9]:

- Gli approcci data-driven, come la definizione stessa suggerisce, necessitano di grandi quantità di dati per risultare efficaci, ma con l'avvento dell'Industry 4.0 e in particolare dell'Industrial Internet of Things questa necessità non è difficile da soddisfare;

- Rispetto agli altri approcci hanno il grande vantaggio di non richiedere conoscenze approfondite specifiche per il dominio di applicazione, rendendo quindi meno determinante il contributo degli esperti sulle performance finale del modello; il contributo degli esperti può comunque essere utile per velocizzare il processo di selezione delle grandezze da utilizzare come input, ma ha un peso molto minore se confrontato con i metodi knowledge-based o physics-based; in aggiunta si ha che le tecniche di learning e data mining possono essere in grado di rilevare relazioni fra i parametri di input e lo stato del sistema che anche agli stessi esperti non sono note a priori;

- Sono disponibili numerosi tool[1] che implementano algoritmi di machine learning che possono essere utilizzati per questi scenari di CBM che

[1]Di questi tool, i più citati e utilizzati in letteratura sono TensorFlow, Scikit-Learn, Keras, PyTorch, Theano e SciPy

richiedono poche operazioni di configurazione e di ottimizzazione per il caso specifico.

La scelta di uno specifico modello di tipologia data-driven è fortemente dipendente dall'obiettivo che si vuole ottenere dal sistema. In base all'obiettivo infatti il problema viene modellato in modo differente.
Le principali opzioni sono riportate di seguito [37]:

Classificazione Binaria

Il modo più semplice è quello di rappresentare la CBM come un problema di classificazione binaria, ossia in cui ogni singolo input rappresentante lo stato del sistema deve essere etichettato con uno fra due possibili valori mutuamente esclusivi.
In caso di problema di diagnostica, questo significa *decidere se la macchina sta funzionando correttamente o non correttamente*, facendo rientrare tutti i possibili stati in queste due classi.
Per applicare la classificazione binaria anche al caso di prognostica l'interpretazione diventa quella di *decidere se la macchina può guastarsi entro un intervallo di tempo fissato.*
La differenza fra le due accezioni è data semplicemente dalla diversa interpretazione delle etichette. Ciò significa che uno stesso modello può risolvere entrambi i problemi. Quello che sarà differenziato è il labeling del dataset utilizzato per svolgere la fase di training del modello.

Classificazione Multiclasse

La versione multiclasse è una generalizzazione della classificazione binaria, in cui viene incrementato il numero di possibili etichette fra cui scegliere. Ad ogni input deve comunque essere associata una sola etichetta.
Il caso di diagnostica estende in modo molto intuitivo il caso precedente, ossia *decidere se la macchina sta funzionando correttamente o non correttamente, e nel secondo caso in quale fra i possibili stati di anomalia.*

Mentre le applicazioni di prognostica si può vedere il problema come *decidere in quale intervallo di tempo prima del guasto si trova la macchina*, dove quindi le possibili etichette rappresentano diversi intervalli di prossimità ad un guasto.

Regressione

La regressione può essere utilizzata per modellare problemi di prognostica. Il che significa permettere di *stimare la vita utile rimanente di un componente* in termini di un numero continuo (fornito appunto dal modello di regressione) di unità di tempo prefissata.
In questo specifico caso il dataset di training deve contenere solamente dati relativi a componenti che sono stati soggetti a guasti, per poter permettere l'etichettatura degli input a ritroso a partire dall'istante di guasto.

Anomaly Detection

Un'ulteriore possibile rappresentazione per problemi di diagnostica è quella di considerarlo come problema di anomaly detection.
Ciò significa che il modello deve essere in grado di stabilire se il funzionamento della macchina rientra in uno stato normale o se si discosta da esso, rientrando cioè in un caso di anomalia.
L'interpretazione del problema è dunque molto simile a quella della classificazione binaria. Tuttavia questa metodologia si differenzia dalla classificazione dato che rientra nei casi di learning semi-supervisionato (a differenza dei casi precedenti, che sono tutti supervisionati), in quanto il modello necessita solamente di imparare da input che rappresentano stati di funzionamento corretti e deve, in seguito alla fase di training, riconoscere stati anomali non noti, ossia di cui non conosce le caratteristiche.

3.3 Dati e Dataset

Come è facile intuire, i dati giocano un ruolo fondamentale per le applicazioni di CBM, specialmente in caso di approccio data-driven. In questa sezione si vogliono pertanto descrivere la caratteristiche rilevanti dei dati che sono utilizzati in sistemi di questo tipo e si riportano inoltre i principali dataset incontrati in letteratura, descrivendone la composizione.

3.3.1 Tipologia e Fonti dei Dati

I dati si possono suddividere nelle seguenti tipologie [8, 37]:

Dati Sensoristici

Sono le misurazioni di tutte quelle grandezze fisiche che descrivono in qualche modo lo stato della macchina durante il suo funzionamento; sono ottenute tramite appositi sensori che convertono il valore fisico in valore elettrico. Esempi di questi parametri utilizzati sono il rumore, le vibrazioni, la pressione, la temperatura e l'umidità, dove la rilevanza di ciascuno di essi dipende fortemente dal sistema che si sta monitorando.
Entrando maggiormente nello specifico si possono suddividere in base al tipo di valore che i sensori forniscono [5]:

- *Valori Semplici*: ossia un singolo valore, tipicamente numerico, raccolto a un preciso istante di tempo, come ad esempio temperatura, pressione e umidità;

- *Segnali*: ossia l'andamento di una singola grandezza per un intervallo di tempo, come ad esempio un'onda sonora o un segnale vibrazionale;

- *Valori Multidimensionali*: ossia una molteplicità di valori raccolti allo stesso istante di tempo riferiti a uno stesso concetto, come ad esempio una fotografia o una termografia a infrarossi.

Dati Statici

Definiti in [37] anche *metadati*, sono i dati che mettono in correlazione a ciascun istante di tempo le condizioni statiche di funzionamento della macchina o dell'impianto, come ad esempio il tipo di pezzo prodotto, il codice dei materiali utilizzati, la velocità di produzione della macchina, identificazione e caratteristiche dell'operatore che sta utilizzando la macchina.
Le fonti di queste informazioni possono essere i PLC delle macchine o i sistemi ERP dell'impianto di produzione, oppure, nel caso in cui essi non siano disponibili, le dichiarazioni manuali degli operatori, che devono essere digitalizzate e integrate a posteriori;

Dati di Log

Sono lo storico degli eventi e delle azioni rilevanti che riguardano una macchina e i suoi componenti. In particolare risultano utili gli elenchi degli interventi di riparazione e sostituzione o la cronologia dei guasti riscontrati. Anche in questo caso essi possono essere ottenuti grazie a sistemi ERP o CMMS, oppure da apposite dichiarazioni degli operatori.

3.3.2 Dataset dalla Letteratura

NASA Turbofan e PHM 08 Challenge

Questi due dataset [39, 38] rappresentano lo stesso identico scenario applicativo e sono strutturati in maniera identica, differiscono solamente nel valore e nel quantitativo dei dati.
Essi contengono le informazioni relative a un insieme di motori turboventola per jet[2], in cui ciascun esemplare di motore viene fatto funzionare ininterrottamente fino al raggiungimento di uno stato di guasto (*Run-to-Failure*), definito a priori come il superamento di una certa soglia per un certo parametro non specificato nella documentazione.

[2]In realtà i valori non provengono da motori reali, ma sono generati da una simulazione effettuata con C-MAPSS (Commercial Modular AeroPropulsion System Simulation)

La struttura dei dati è quella di una timeseries multivariata, dove l'unità di tempo è il ciclo operativo del motore e 24 variabili rappresentano le condizioni operative (3) e il valore di sensori (21). Non è specificato né il significato delle condizioni operative, né la tipologia di sensori. La timeseries per ciascun motore termina al verificarsi del guasto.
Si presta principalmente per testare algoritmi di prognostica, dato che nel dataset sono presenti sia timeseries complete per il training, ossia che si interrompono al raggiungimento dello stato di guasto, sia timeseries di testing, che si interrompono ad un momento intermedio per le quali va stimato il numero di cicli operativi rimanenti.

NASA Bearing e FEMTO Bearing

I dataset [40, 41] riportati in questa sezione hanno in comune lo scenario applicativo e la forte somiglianza della loro struttura interna. Le informazioni che contengono riguardano per entrambi lo stato di salute di cuscinetti a sfera sottoposti ad un carico radiale.
Anche in questo caso il problema viene presentato tramite timeseries run-to-failure, ma a differenza del caso precedente ogni entry non è un valore semplice, bensì un segnale vibrazionale, che a sua volta è una timeseries.
I dataset sono stati ottenuti nel seguente modo: ogni 10 minuti (10 secondi nel caso FEMTO) si effettua un processo di campionamento del valore di accelerazione tramite un accelerometro fissato su di un cuscinetto, ad una frequenza di 20KHz (25.6KHz) per un intervallo di tempo di 1 secondo (0.1 secondo). Si ripete il procedimento fino a quando non si verifica un guasto del cuscinetto che sta venendo monitorato.
Per entrambi i dataset è fornito un file per ciascun segnale di vibrazione ottenuto.

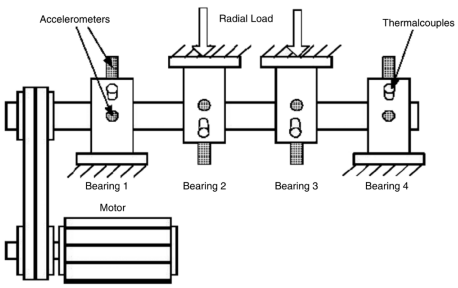


Figura 3.2: Schema del setup del dataset NASA Bearing [4]

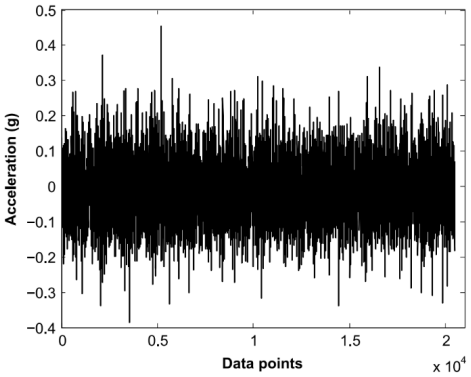


Figura 3.3: Esempio di segnale di vibrazione di NASA Bearing [4]

Azure Telemetry

E' un dataset [42] che contiene informazioni relative a 100 macchine, con dati eterogenei raccolti per un intero anno di funzionamento [3].
Il dataset si suddivide in:

- *Dati di Telemetria*: timeseries multivariata di pressione, voltaggio, rotazione e vibrazione della macchina, dove i valori di ogni variable per ciascuna entry della timeseries rappresentano l'aggregazione delle misurazioni per un intervallo di un'ora;
- *Dati degli Errori*: storico dei log riguardo gli errori non bloccanti rilevati per ciascuna macchina, specificando l'ID dell'errore e l'istante di accadimento, arrotondato all'ora per essere coerente con la telemetria;
- *Dati di Manutenzione*: storico degli interventi di manutenzione, programmata e non, dove per ogni entry è specificata la macchina e il componente su cui si è intervenuti, con timestamp arrotondato all'ora;
- *Dati delle Macchine*: elenco delle macchine e delle loro caratteristiche, in particolare il codice del modello e gli anni di funzionamento:

[3] Anche in questo caso i dati non provengono da macchinari reali, ma sono frutto di una simulazione

- *Dati dei Guasti*: storico dei guasti, dove per ciascuno di essi è indicata la macchina, il componente e l'istante di tempo, anche in questo caso arrotondato all'ora.

3.4 Soluzioni Proposte

3.4.1 Architettura

Per quanto riguarda l'architettura complessiva di un sistema di CBM in ambito Industry 4.0, la maggior parte degli articoli consultati (come ad esempio [13, 14, 10, 16]) utilizza un approccio comune, sia per quanto riguarda i componenti e le loro responsabilità, che per le interazioni che avvengono fra di essi.
Si riporta quindi in Fig. 3.4 uno schema complessivo, che generalizza le soluzioni proposte dagli articoli sopracitati e si fornisce una descrizione dei ruoli di ciascun componente.

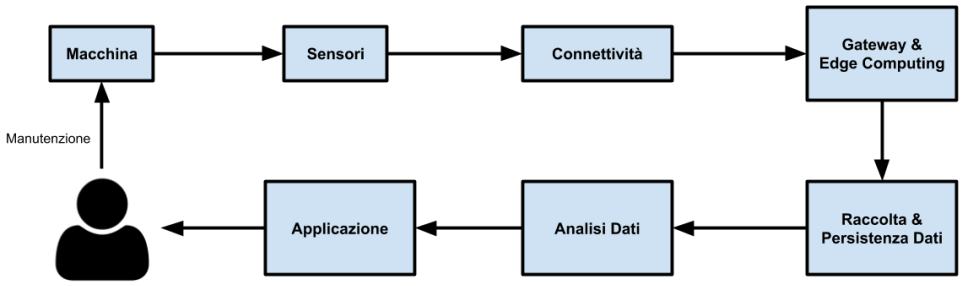


Figura 3.4: Architettura di un generico sistema di CBM

- *Sensori:*: come già descritto in 3.3.1, sono quei dispositivi che si occupano di rilevare le grandezze fisiche di interesse dalla macchina;
- *Connettività*: dispositivi che si interfacciano direttamente con i sensori per effettuare le letture dei dati da essi raccolti per poi trasmetterli tramite una qualche tecnologia di comunicazione, che può essere tramite cavo o wireless a seconda delle caratteristiche dello specifico scenario;

- *Gateway*: è un primo punto di raccolta dei dati grezzi provenienti da multipli sensori; questi dati possono essere filtrati o aggregati secondo una logica ben definita, con lo scopo di ridurre il traffico di dati sulla rete e di rilevare e scartare eventuali dati anomali o non significativi il prima possibile;

- *Raccolta e Persistenza Dati*: si occupa della raccolta delle informazioni provenienti dai gateway ed è il livello che conosce quali dati vanno mantenuti e quali invece possono essere scartati; i dati che si salvano su database saranno poi utilizzati in seguito per le analisi;

- *Analisi Dati*: è il componente che implementa il modello statistico o di apprendimento e trasforma quindi i dati in informazioni significative;

- *Applicazione*: ossia dove le informazioni dedotte dal componente precedente sono presentate all'utente finale, eventualmente anche intervenendo sulla fase di decision-making, suggerendo azioni correttive che l'utente potrà poi svolgere.

3.4.2 Modelli

Come evidenziato dal paragrafo precedente, più che nell'architettura del sistema, le maggiori differenze fra le soluzioni proposte si hanno nella realizzazione del modello vero e proprio.
Fra le tre tipologie di approccio al problema riportate ci si concentra nella descrizione di lavori che utilizzano strategia data-driven in quanto, anche se molti di essi sono pensati e testati in scenari specifici, l'idea e il principio di funzionamento sono facilmente generalizzabili ed utilizzabili anche in contesti differenti.
Si riportano quindi i modelli presenti in letteratura ritenuti più significativi, evidenziando il tipo di problema di CBM che risolvono e la tipologia di dati che utilizzano.

In [18] si utilizza **Support Vector Machine** (**SVM**) per risolvere un problema di diagnosi di guasti al rotore di motori a induzione. Il dataset utilizzato è formato da segnali di vibrazione raccolti da motori reali in diverse condizioni di salute, con una frequenza di campionamento della vibrazione di 40KHz. Sono poi proposte tre diverse tecniche di estrazione di feature, che sono utilizzate come input per il modello SVM. In particolare utilizzando come feature i coefficienti di un modello autoregressivo per modellare i segnali di vibrazione, gli autori affermano di ottenere una precisione del 100% nel riconoscimento dei guasti in fase di testing.

Anche in [19] si utilizza SVM per una applicazione di monitoraggio e diagnostica, ma nel caso di un processo chimico industriale. A differenza del caso precedente si vuole distinguere fra più di due classi, e pertanto la soluzione proposta è quella di trasformare il singolo problema multiclasse in multipli problemi di classificazione binaria, allenando quindi N (12, nel caso dell'articolo) modelli, ciascuno con il compito di distinguere una classe rispetto a tutte le altre, e ottenendo complessivamente una diagnosi corretta in 559 di 576 casi di test.

Una metodologia simile è usata da [22] per distinguere la qualità di una saldatura a partire da corrente, voltaggio e velocità dei macchinari, che utilizza però i modelli SVM in sequenza tra loro, in modo tale che il primo distingua fra la classe (1) e la classe (2 & 3), mentre il secondo distingua fra le classi (2) e (3), anche in questo caso con ottimi risultati, ossia con il 94.4% di correttezza.

In [21] viene presentata una soluzione per la prognostica, ossia la stima della *Remaining Useful Life* (*RUL*). SVM viene usata in questo caso per un problema di regressione (Support Vector Regression), e testata in due casi d'uso, la previsione di vita rimanente di turbocompressori di motori diesel e la distanza ancora percorribile di motori di automobili.

In [25], si propone un metodo per rilevare e classificare difetti di cuscinetti a sfera di motori elettrici partendo da segnali di vibrazione utilizzando una **Rete Neurale**. Il dataset che si utilizza è composto da valori di accelerazione campionati a 32KHz e si vuole risolvere un problema di classificazione a 4 classi, tra cui una di corretto funzionamento e 3 di differenti stati di guasto. Come input alla rete si utilizzano feature nel dominio del tempo estratte dai segnali, ossia varianza, scarto quadratico medio, simmetria e curtosi. I risultati riportano una percentuale di classificazione corretta che oscilla tra il 96% e il 100%.

Le reti neurali sono utilizzate anche in [24, 25] per applicazioni di prognostica. Il primo lavoro utilizza anch'esso l'analisi vibrazionale per ottenere, tramite trasformata wavelet e trasformata di Fourier, le feature dei segnali ed utilizzarle come input per una rete a N neuroni di output, uno per ogni possibile guasto individuato. Il valore dei neuroni di output è compreso fra 0 e 1 e rappresenta la percentuale di trovarsi in quello stato. Risulta quindi che la rete neurale di per sé effettua solamente la diagnostica, e per passare da questa informazione alla stima di vita rimanente si usano metodi statistici, derivati da uno storico della durata residua associata a ciascuna tipologia di guasto.
Il secondo articolo invece utilizza la rete direttamente per un problema di regressione, quindi con un unico neurone di output che fornisce la percentuale di vita rimanente stimata. Il metodo descritto ottiene un errore medio del 10.6% nella stima della RUL, che si riduce al 3.6% se si considerano solo i punti in cui il componente analizzato ha una vita residua inferiore al 10%, a dimostrazione del fatto che il metodo diventa molto più preciso in prossimità del raggiungimento di condizioni critiche.

Entrambi i lavori descritti in [26, 27] si utilizzano invece **Alberi di Decisione** per effettuare la previsione di guasti futuri, inteso come verificabili

in una finestra temporale di durata fissata, a partire dall'istante attuale.
Il primo sfrutta come dati di interesse lo storico dei log degli eventi di macchine da gioco elettroniche, inviati dalle macchine stesse a un elemento centrale di raccolta. Riesce a ottenere una accuratezza oltre il 90% e, nonostante altri metodi riescano a ottenere un pari livello di performance, gli autori preferiscono gli alberi di decisione perché interessati anche a capire ed analizzare ulteriormente le motivazioni e le cause che hanno portato il modello a segnalare una macchina come in procinto di guastarsi.
Lo stesso principio guida anche il secondo lavoro, che combina lo storico di eventi di manutenzioni e ispezioni con dati raccolti da sensori su treni per effettuare previsioni di stati anomali.

Una tecnica simile è presentata da [28], che sfrutta **Random Forest**, per individuare lo stato e la possibile presenza di guasti nel compressore di camion. L'approccio è sempre con la finestra di previsione, e non usando la regressione del *time-to-repair*, che viene però presentato come un possibile sviluppo futuro.

In [29] viene proposto un approccio per la stima di RUL di cuscinetti a sfera a partire dal relativo segnale di vibrazione. Il dataset su cui lavorano gli autori è il già citato dataset di cuscinetti dal repository della NASA (3.3.2). La tecnica presentata si basa su **Hidden Markov Model** (**HMM**) in cui ogni stato del modello rappresenta un diverso stato di salute del cuscinetto e permette di ottenere, a partire dallo stato stimato corrente, le probabilità di arrivare a uno stato di guasto. Il percorso più probabile è quindi utilizzato per calcolare la RUL. Le feature utilizzate dalla soluzione in questione sono estratte dai segnali grezzi di accelerazione tramite decomposizione in pacchetti wavelet.

Gli articoli [30, 31] propongono un metodo basato su hidden semi-Markov model per diagnostica e prognostica di pompe idrauliche. HSMM si diffe-

renzia da HMM in quanto ogni stato genera una sequenza di osservazioni, e permette di conseguenza l'integrazione nel modello del concetto di durata temporale di permanenza in uno stato, semplificando il processo di calcolo della RUL. I risultati ottenuti (in particolare in [31]) raggiungono il 91% di precisione nella diagnostica e 8.3% di RUL di errore medio nella prognostica.

Un altro modello che viene utilizzato spesso in letteratura per questo genere di applicazioni è l'**Auto Regressive Integrated Moving Average** (**ARIMA**), utilizzato per modellare il comportamento di timeseries e poterne poi predire l'andamento.
In [17] viene usato ARIMA per monitorare lo stato di salute dei sensori stessi, effettuando anomaly detection. Si utilizza lo storico dei valori raccolti per allenare il modello, che viene poi utilizzato per fornire previsioni dei prossimi punti. Si confrontano poi i valori effettivamente forniti dal sensore con quelli stimati, e se la differenza supera una soglia prefissata si segnala una anomalia.

Anche in [14] ARIMA viene utilizzato per la previsione dei valori dei sensori in instanti futuri, anche se si differenzia dal lavoro precedente in quanto non si propone un sistema a soglia fissa per rilevare le anomalie, ma si usa un sistema di learning supervisionato per effettuare classificazione di difetti di macchine tagliatrici. In questo caso la fase di training della classificazione avviene utilizzando dati reali raccolti da sensori in uno stato di guasto, mentre in fase di testing i valori sono quelli previsti da ARIMA. Si ottiene una precisione in classificazione compresa fra il 94% e il 98% a seconda del classificatore usato.
Un approccio simile a quello appena descritto viene usato anche da [32], che però effettua outlier detection e non classificazione.

In [34] viene proposto **One-Class SVM** per prevedere anomalie che riguardano convertitori di potenza di treni. Sono utilizzati dati provenienti da sistemi di log e monitoraggio continuo a bordo dei treni. Questi dati sono

poi raggruppati per viaggio e, solo quelli relativi a viaggi senza anomalie, sono utilizzati per allenare il modello. I risultati riportano una precisione del 25.5%, ma questo accade poiché l'autore ha scelto di segnalare come anomalie solo i casi dove la confidenza supera il 90%, con lo scopo di ridurre al minimo i falsi allarmi.

In [33] si utilizza one-class SVM per la diagnostica di impianti di riscaldamento. Si vogliono poter classificare 6 tipologie differenti di guasto, pertanto gli autori utilizzano un sistema composto da 6 one-class SVM, ciascuno allenato nel riconoscimento di una specifica classe, preferendo utilizzare più modelli semplici e specializzati rispetto a uno unico ma più complesso.

Infine l'ultima tipologia di approccio che si riporta sono gli **Autoencoder**. In [17] sono utilizzati per effettuare anomaly detection. Sono allenati in fase di training con dataset che hanno sia come input che come output le stesse identiche finestre temporali di segnale di dimensione fissata. L'idea è che il modello deve imparare feature nascoste e relazione fra i valori del segnale. In fase di testing si considera poi l'errore di ricostruzione del segnale. Se è superiore a una certa soglia significa che l'input ha caratteristiche differenti da quelli utilizzati in training, ed è pertanto segnato come anomalia.

In [35] gli autoencoder sono utilizzati invece per rimuovere il noise dal segnale, ossia ricostruire un segnale pulito a partire da un segnale in input con rumore. La fase di classificazione è in seguito fatta da un SVM.

3.5 Sfide e Problemi Noti

Nonostante i possibili approcci per la realizzazione di una soluzione siano numerosi e diversificati, ci sono alcune problematiche che risultano essere comuni, e che tutti i sistemi devono in qualche modo affrontare.
Essi sono elencati e descritti di seguito.

- I dati utilizzati, specialmente quando si tratta con valori raccolti da sensoristica posizionata su macchinari industriali, contengono di solito un livello significativo di rumore. Inoltre possono anche presentare variazioni dovute a diverse condizioni ambientali o ad altri fattori esterni. Gli algoritmi devono pertanto essere sufficientemente robusti da tollerare queste oscillazioni non legate allo stato di salute dei componenti.

- I modelli di learning su una quantità e una varietà molto elevata di dati richiedono elaborazioni molto intense dal punto di vista del costo computazionale. Questo aspetto risulta essere maggiormente delicato quando si vogliono realizzare sistemi per la diagnostica o la prognostica real-time, in cui la reattività gioca un ruolo fondamentale.

- Il procedimento effettuato da approcci data-driven per fornire un output è completamente indipendente dal reale processo fisico. Se non per pochi algoritmi (fra cui gli alberi di decisione, come indicato nella sezione precedente), non si riesce a interpretare in modo intuitivo il motivo per cui il modello ritiene di essere in un determinato stato.

- Si deve tenere in considerazione la gestione dell'incertezza. Ci sono diverse fonti di incertezza: la prima è introdotta dal rumore nei dati e dalle condizioni esterne, la seconda è quella del riconoscimento dello stato attuale, e infine quella della previsione degli stati futuri.

- Per quanto riguarda la prognostica si ha inoltre una difficoltà aggiuntiva dovuta al fatto che uno stesso guasto può verificarsi in seguito a percorsi di degrado differenti. Inoltre spesso si ha una forte dipendenza fra componenti della macchina, e il guasto di uno può influire sulla salute degli altri. Bisogna quindi anche considerare il caso di occorrenza di guasti multipli contemporanei.

- Molti degli approcci descritti nei lavori correlati suppongono di avere a disposizione un dataset contenente esempi di valori di sensori clas-

sificati per classe specifica di guasto. In uno scenario reale è molto difficile che esso sia presente, e se manca significa che bisognerebbe indurre appositamente sulla macchina i difetti, e ciò risulta spesso molto complicato da mettere in atto.

- Infine si ha una forte dipendenza dal fattore umano. Sono gli esperti del settore che devono effettuare la classificazione sul dataset che verrà usato. E' una fase molto delicata, in quanto errori durante l'etichettatura possono causare un training errato e di conseguenza il sistema rischia di risultare inutile se non addirittura dannoso.

Capitolo 4

Architettura

In questo capitolo si descrivono le varie parti, sia hardware che software, che compongono il sistema e le interazioni fra esse.
La struttura generale si basa su quella che risulta essere la più utilizzata in letteratura, riportata in Fig. 3.4, in cui le responsabilità di storage e di analisi sono state condensate in un unico nodo.
Di seguito si riportano il diagramma rappresentante l'architettura del sistema e la descrizione dei singoli componenti.

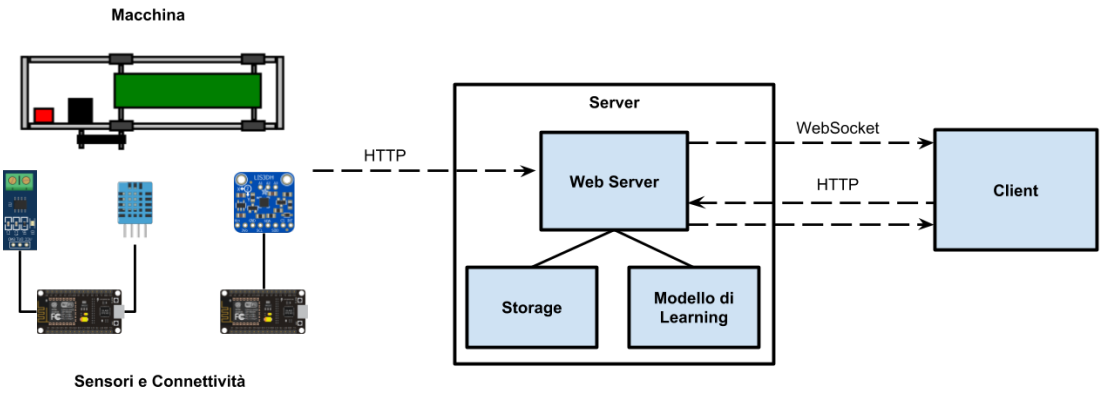


Figura 4.1: Architettura del sistema proposto

4.1 Macchina

La macchina a cui è stato applicato il sistema di rilevazione e classificazione delle anomalie è un nastro trasportatore che è stato progettato e realizzato appositamente per il progetto di tesi.

Durante la fase di analisi del progetto, si sono individuati i requisiti che la macchina avrebbe dovuto soddisfare, che sono i seguenti:

- La possibilità di applicare su di essa una serie di sensori per rilevare parametri significativi sul suo stato, senza interferire con il funzionamento;
- La disponibilità a tempo pieno della macchina per utilizzarla ai fini di test e analisi esplorative;
- La possibilità di intervenire su di essa in modo semplice per portare lo stato in una o più situazioni di anomalie e da esse ritornare allo stato di funzionamento ottimale;
- La facilità di accensione e utilizzo, in modo tale da non richiedere competenze specifiche per allenare e testare il sistema di CBM;
- Il costo contenuto e la facile reperibilità dei suoi componenti, in modo da operare su di essa con maggiore libertà;
- Somiglianza o corrispondenza con macchinari industriali, nella funzione e nei componenti.

Non essendo stato possibile trovare qualcosa che rispettasse tutti i requisiti, si è deciso di procedere alla realizzazione autonoma di tale macchina.
L'ispirazione è stata presa da [43], a cui però sono state effettuate diverse migliorie nel design e nelle funzionalità.

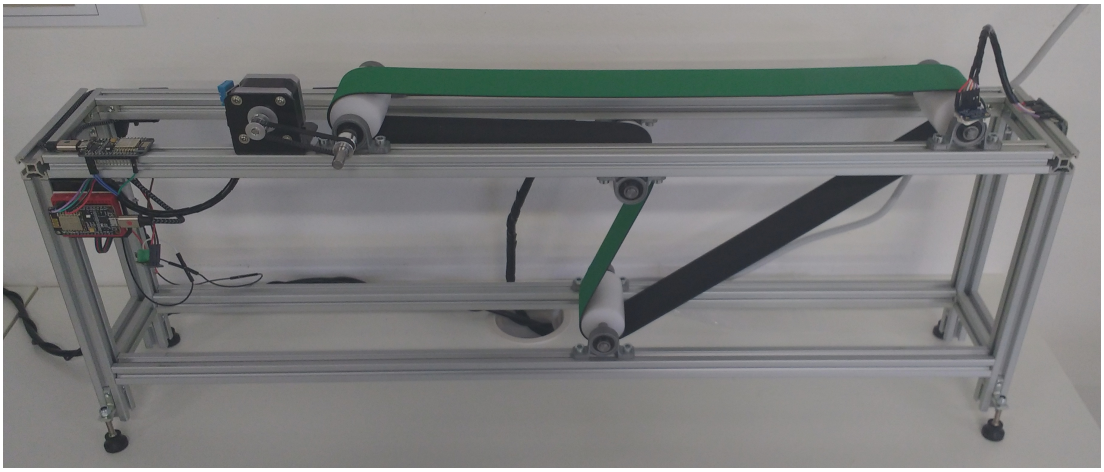


Figura 4.2: Nastro trasportatore utilizzato per la realizzazione del progetto

Il nastro trasportatore è riportato in Fig. 4.2 ed è azionato da un Nema 17, cioè un motore passo passo che viene tipicamente usato nelle stampanti 3D.
Tramite una piccola cinghia il moto viene trasmesso al nastro, che è realizzato con una cinghia industriale. Il nastro si appoggia su quattro rulli a bombè e per ciascun rullo sono presenti un albero e due cuscinetti a sfera.
La struttura della macchina è formata da profilati di alluminio e l'intera macchina poggia su quattro piedini, utilizzati per garantire più stabilità durante il funzionamento.
Il tutto è fatto funzionare con un'alimentazione da 12V e 2A.

Il nastro è azionato tramite un NodeMCU, ossia una board di sviluppo che integra un ESP8266, che è un microcontrollore dotato funzionalità WiFi. Il NodeMCU pilota il motore appoggiandosi a un driver A4988.
Il NodeMCU è programmato per funzionare da web server, ed espone una semplice pagina HTML tramite la quale l'utente è quindi in grado di azionare e bloccare il nastro, regolare il senso di rotazione del motore e impostare la velocità.
Per tutti i test eseguiti, il motore è stato fatto funzionare ad una velocità di

600 step al secondo, che corrispondono a 3 rotazioni complete al secondo, in quanto un singolo step del Nema 17 equivale a 1.8°.

4.1.1 Guasti

L'obiettivo del progetto è quello di realizzare un modello in grado di poter riconoscere e classificare gli stati di guasto quando essi si verificano sulla macchina. Nel caso del nastro trasportatore gli stati di guasto che si sono considerati sono i seguenti:

- Allentamento dei serraggi che fissano i vari cuscinetti ai profilati di alluminio che costituiscono la struttura della macchina;
- Aumento dello sforzo necessario al movimento del nastro, ottenuto ostacolando la cinghia;
- Rallentamenti della velocità di rotazione del motore;
- Combinazione dei precedenti.

Si è inoltre voluta verificare la capacità di individuare per uno stesso guasto diversi stati di avanzamento del problema. Per realizzare questo scenario si è provocato il progressivo allentamento di uno stesso cuscinetto, agendo sulla vite con una chiave dinamometrica, al fine di testare diversi valori di coppia di serraggio.

4.2 Sensori

Ai fini di determinare lo stato di funzionamento della macchina si sono individuate alcune grandezze fisiche che si ritengono significative e che sono misurabili tramite sensori sulla macchina.
Questi parametri sono le vibrazioni, la corrente e la temperatura del motore. Per ciascuna di esse si è scelto un sensore in grado di misurarne i valori, che sono letti da un NodeMCU a intervalli regolari e inviati al server di archiviazione ed elaborazione tramite richieste HTTP via WiFi.

Accelerometro

Da una analisi dei dataset e dei modelli di CBM presenti in letteratura si è notato che i segnali di vibrazione rappresentano la tipologia di input maggiormente utilizzata, e numerosi articoli riportano risultati positivi ottenuti usando la vibrazione come input.
Pertanto l'accelerometro è stato individuato come elemento necessario fin dalle prime fasi del progetto. Fra le caratteristiche dei sensori usati negli articoli consultati si è notato che la frequenza di campionamento dei valori dell'accelerazione varia dai 20KHz fino ai 40KHz. Hardware in grado di garantire prestazioni di tale livello richiede costi molto elevati, fuori dal budget previsto per il progetto.
Tuttavia fra gli accelerometri a basso costo si è utilizzata proprio la frequenza di campionamento come discriminante per la scelta del dispositivo da utilizzare. Si è optato per il LIS3DH [46], un accelerometro MEMS[1] triassiale a bassi consumi con interfaccia SPI[2] e con data rate configurabile da 1Hz fino ad un massimo di 5.3KHz.

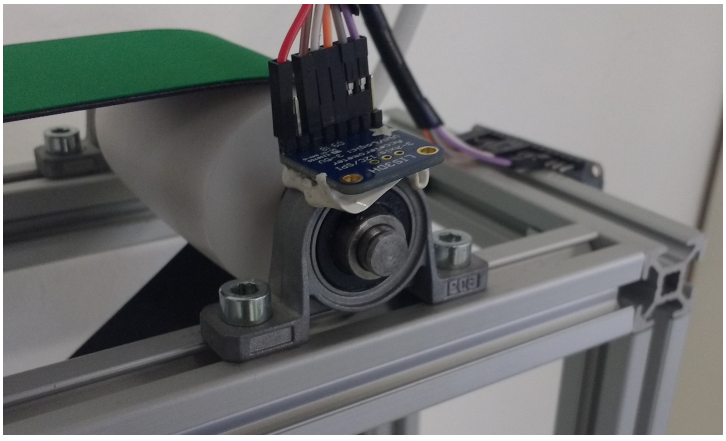


Figura 4.3: Posizionamento dell'accelerometro sul cuscinetto della macchina

[1]Micro Electro-Mechanical Systems
[2]Serial Peripheral Interface

L'accelerometro è stato fissato su un cuscinetto, come mostrato in Fig. 4.3. A differenza dei lavori in letteratura l'obiettivo non è quello di riconoscere lo stato di degrado dei componenti interni del singolo cuscinetto su cui è piazzato, ma di analizzare lo stato vibrazionale tutto il sistema composto da cuscinetti, rulli e nastro.

La tecnica di campionamento rispecchia quella utilizzata nel dataset NASA Bearing, ossia si campiona 1 secondo di vibrazioni a frequenza di 5.3KHz ogni 10 secondi.
Dopo ogni intervallo di campionamento, il NodeMCU invia i dati raccolti al server, con una serie di chiamate POST HTTP.

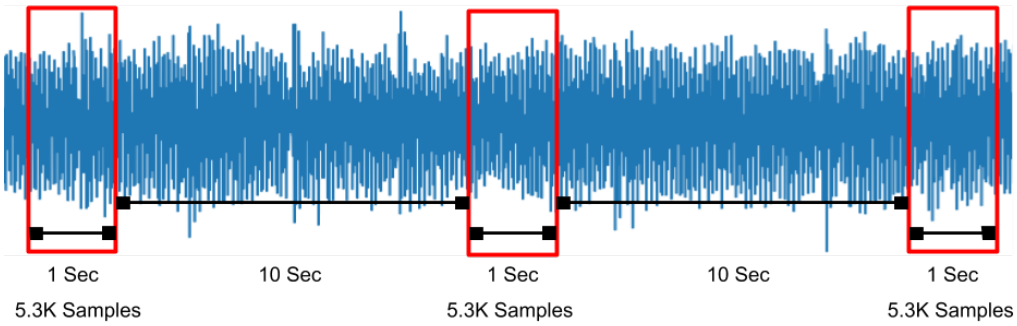


Figura 4.4: Schema del processo di campionamento dell'accelerazione

Sensore di Corrente e di Temperatura

Dopo una primo periodo in cui si ha lavorato solamente con i dati provenienti dall'accelerometro, si è deciso di integrare anche il monitoraggio di altri parametri fisici.
Il primo è la corrente utilizzata per il funzionamento del motore. Per la misurazione si è utilizzato un sensore di corrente ACS712 [45].
L'altra grandezza ritenuta interessante è la temperatura, rilevata tramite un sensore DHT11 [44] posizionato sul Nema 17.

Anche le letture provenienti da questi sensori sono effettuate a intervalli regolari di 10 secondi ma, a differenza di quanto accade con le vibrazioni, ciò che viene rilevato sono valori puntuali e non dei segnali nel tempo.
Entrambi i sensori sono collegati a uno stesso NodeMCU, che invia i valori raccolti al server tramite chiamate POST HTTP.

4.3 Raccolta ed Elaborazione Dati

Le funzionalità di raccolta e di analisi dei dati sono svolte da un apposito server. Il programma in esecuzione su questo nodo del sistema è diviso in tre moduli principali: storage, modello di learning e web server.

Il modulo di *Storage* si occupa della persistenza, dell'accesso e dell'interpretazione dei dati. La persistenza è realizzata seguendo la struttura suggerita dai dataset NASA e FEMTO Bearing, ossia si crea un file per ciascun intervallo di campionamento dei dati. Nel caso specifico quindi ogni file che viene creato contiene un valore di temperatura, uno di corrente e circa 5300 valori di accelerazione che rappresentano l'intero segnale di vibrazione corrispondente alla finestra di campionamento di un secondo.
Per maggiore chiarezza, invece di utilizzare CSV, i file sono in formato JSON. Un esempio di file è riportato di seguito.

```
{"current": 0.71,
"temperature": 24.0,
"signal": [
      {"x": 220},
      {"x": 246},
      ...
      {"x": -112}
   ]
}
```

Esempio 4.1: Struttura di un file di dati per un singolo intervallo

Il modulo di *Learning* è quello che realizza il modello di riconoscimento e classificazione dello stato di funzionamento della macchina a partire dai dati ricevuti.
L'implementazione del modello è descritta nel dettaglio nel capitolo successivo. In questo paragrafo si elencano però le funzionalità che esso mette a disposizione, tramite la descrizione dell'interfaccia esposta dal web server.

Il *Web Server* funge infatti da punto di accesso unico ai moduli di storage dei dati e di rilevazione di anomalie. Il server espone una API per l'accesso ai singoli servizi accettando richieste HTTP, con lo scopo di rendere l'interazione con il sistema indipendente da un client o da una logica applicativa specifica.
Oltre a permettere ai NodeMCU di inviare i valori raccolti dai sensori tramite POST, come già descritto nella sezione precedente, gli altri servizi esposti sono pensati per interfacciarsi con una generica applicazione utente che vuole usufruire delle capacità del sistema.
Un elenco completo di questi servizi è riportato di seguito.

Creazione o Selezione di una Sessione - Il software permette di gestire diverse sessioni di funzionamento, ciascuna con il proprio storico di dati, le proprie classi di guasto e i propri modelli. Questa funzionalità permette all'utente di inizializzare una nuova sessione o di caricarne una dall'elenco di quelle presenti. I dati ricevuti dai sensori sono sempre associati alla sessione attiva e in caso di nessuna sessione selezionata non verranno salvati in modo persistente.

Richiesta dello Storico di Dati - Permette all'utente di richiedere i dati relativi a uno o più intervalli di campionamento.

Creazione di una Classe - Una classe è un'etichetta che viene assegnata ai dati. Rappresenta uno dei possibili stati di funzionamento della macchina. Durante la creazione di una nuova classe l'utente deve specificare se lo stato rappresenta uno stato normale o di anomalia. Ciò permette

al sistema di tollerare anche condizioni diverse che però indicano stati accettabili, utile ad esempio quando la stessa macchina può lavorare in più formati o sotto condizioni differenti.

Classificazione di Dati - Consiste nell'attività di labeling, ossia l'associazione vera e propria di classi ai singoli input, dove ogni input è l'insieme dei dati raccolti in un intervallo di campionamento, ossia il singolo file JSON. La classificazione è poi usata dal modello di learning per effettuare la fase di training.

Avvio e Chiusura di Fasi di Training - Permette di avviare una fase di training per una classe specifica. Ciò significa che durante il periodo di tempo fra l'avvio e la chiusura tutti i segnali che vengono ricevuti dai sensori sono automaticamente etichettati con la classe specificata. Subito dopo la chiusura, i nuovi dati raccolti sono utilizzati per aggiornare i modelli di rilevazione e classificazione di anomalie.

Anomaly Detection - Dato un input e una classe fra quelle disponibili per la sessione in corso, viene stimato quanto il segnale di input si distanzia dal modello costruito per la classe selezionata. Questa funzionalità permette quindi di evidenziare anomalie rispetto a una condizione di base per una certa classe. Ciò risulta molto utile soprattutto quando si hanno a disposizione solamente esempi di funzionamento corretto, ma si vuole comunque riconoscere quando ci si sta discostando da essi.

Classificazione - Questa funzione integra la precedente, e permette, dato uno specifico segnale di input, di individuare a quale fra tutte le possibili classi conosciute il segnale risulta più simile. Per utilizzare questo servizio la sessione in corso deve avere definito almeno due classi distinte.

Tutte le chiamate ai servizi descritti sopra avvengono tramite richieste HTTP, pertanto le interazioni sono tutte iniziate dall'applicazione utente. Per abilitare anche comunicazioni *push-based*, il server supporta anche il protocollo

WebSocket.

La WebSocket è aperta dal client in fase iniziale, poi però viene usata dal server per funzionalità di notifica. I casi d'uso principali per cui è stata introdotta riguardano l'informazione dell'applicazione della ricezione di nuovi segnali di input da parte dei sensori oppure la notifica di essere in stato di anomalia.

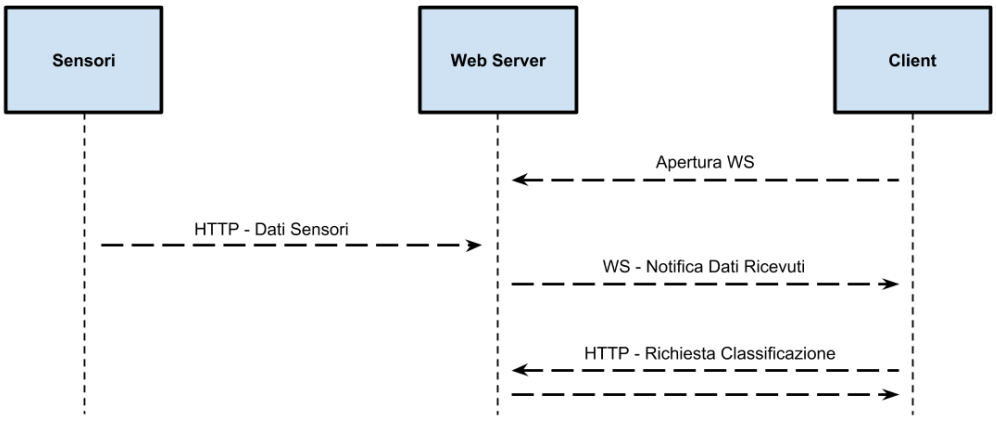


Figura 4.5: Diagramma di sequenza dell'uso di WebSocket

Capitolo 5

Rilevazione e Classificazione di Anomalie

In questo capitolo dell'elaborato si descrive come i dati raccolti dai sensori sono utilizzati per il riconoscimento di anomalie sulla macchina e quindi come sono effettivamente implementate le funzionalità esposte dal sistema descritte nel capitolo precedente.
Sono presentate le tecniche di preprocessing ed estrazione di feature dai dati e sono poi riportate le diverse tecniche sperimentate, ciascuna con i risultati ottenuti e una valutazione di aspetti positivi e negativi.
E' infine riportato il modello che si ritiene la migliore fra le soluzioni testate per il problema trattato.

5.1 Preprocessing e Feature

Per lo scenario specifico in cui si è svolto il progetto, ogni elemento di input proveniente dalla macchina che si vuole analizzare tramite i modelli è composto da un singolo valore di temperatura, un singolo valore di corrente e un segnale di vibrazione di un intervallo di un secondo, costituito da circa 5300 valori di accelerazione.
Temperatura e corrente, in quanto valori numerici, non necessitano di ul-

teriori elaborazioni, mentre il segnale di vibrazione non è adatto per essere utilizzato in modo grezzo dagli algoritmi di learning.
Su di esso infatti si effettua prima una fase di filtro e in seguito si esegue l'estrazione di feature.

Applicare un filtro ha lo scopo di rimuovere il rumore dal segnale, cercando di estrapolare la vibrazione originaria. In questo caso particolare la fase di preprocessing si ritiene utile per mitigare la poca precisione del sensore LIS3DH, fortemente soggetto a noise quando si richiede un'elevata sensibilità e frequenza di campionamento.
Durante l'implementazione del progetto sono stati testati tre filtri, riportati di seguito.

Media Mobile - Si ottiene utilizzando una finestra mobile di dimensione fissata, fatta traslare lungo tutto il segnale. Per ogni posizione della finestra si effettua una media dei valori, e il segnale filtrato è dato dalla sequenza di tutte le medie ottenute.

Filtro Mediano - Si applica tramite un procedimento analogo a quello della media mobile, ma per ogni campione si tiene la mediana dei valori nella finestra.

Filtro Passa Basso - E' un filtro che consente di rimuovere dal segnale tutte le frequenze al di sopra di una certa frequenza fissata.

Dopo la fase di filtro del segnale si procede quindi con l'estrazione delle feature, con lo scopo di ridurre la dimensionalità dell'input e di rendere maggiormente evidenti caratteristiche utili note. Le feature che si è scelto di utilizzare sono prese dalla letteratura e risultano essere fra le più utilizzate in analisi vibrazionale o analisi dei segnali in generale.
L'elenco delle feature estratte, con nome e descrizione, è il seguente:

1. **Mean** - La media di tutti i valori di accelerazione del segnale
2. **Max** - Il massimo fra i valori di accelerazione del segnale

3. **Min** - Il minimo fra i valori di accelerazione del segnale

4. **Std** - La deviazione standard dei valori di accelerazione

5. **MaxFreq** - La frequenza nel segnale con la massima ampiezza

6. **MaxPow** - Il valore dell'ampiezza massima fra le frequenze del segnale

7. **Peak** - Il picco, definito come la metà della differenza fra il massimo e il minimo valore di accelerazione

8. **RMS** - Il valore efficace, ossia la radice della media quadratica dei valori di accelerazione
$$\sqrt{\frac{1}{n}\sum_{i=0}^{n-1} x_i^2}$$

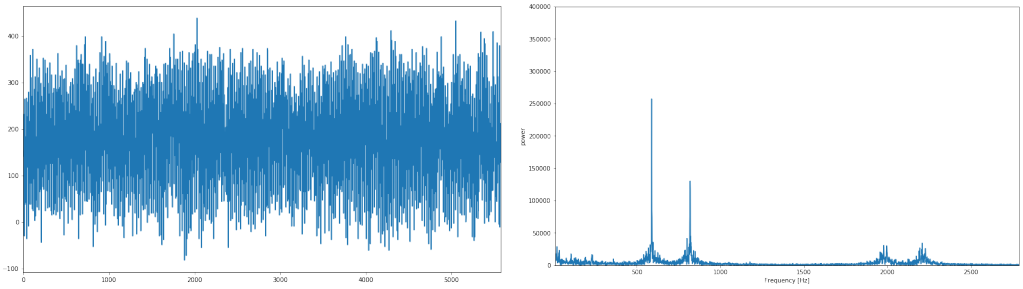
9. **Crest** - Il fattore di cresta, definito come il rapporto fra il valore di picco e RMS

10. **Kurtosis** - Coefficiente di curtosi, che fornisce indicazioni sulla forma della distribuzione, definito come:
$$\frac{1}{n}\sum_{i=0}^{n-1} \frac{(x_i - \bar{x})^4}{(Std)^4}$$

11. **Skew** - Indice di simmetria, definito come:
$$\frac{1}{n}\sum_{i=0}^{n-1} \frac{(x_i - \bar{x})^3}{(Std)^3}$$

12. **Entropy** - L'entropia della distribuzione dei valori di accelerazione del segnale:
$$\sum_{i=0}^{n-1} P(x_i) log_2 P(x_i)$$

Oltre a questi valori, che sono estratti dal segnale di vibrazione, si aggiungono anche corrente e temperatura, per un totale di 14 parametri per ogni input.



mean	max	min	std	freq1	pow1	peak	rms	crest	kurtosis	skew	entropy
177.075117	438	-82	122.331929	589.0	257194.351761	260.0	215.216205	1.208087	-1.240159	-0.064552	5.357869

Figura 5.1: Segnale di vibrazione nel dominio del tempo, delle frequenze e le feature estratte da esso

5.2 Classificazione

Il primo obiettivo che si voleva raggiungere per il progetto era quello di riuscire a distinguere fra uno stato di normale funzionamento della macchina e stati con presenza di anomalia. Si vuole quindi ottenere un servizio in grado di fornire diagnostica.
Si è scelto di affrontare il problema utilizzando algoritmi di machine learning per la risoluzione di classificazione multiclasse.
I modelli in questione sono riportati di seguito.

Random Forest - la foresta è composta da 10 alberi di decisione, costruiti con criterio Gini e senza una profondità massima.

Reti Neurali - si è usata una rete neurale feed forward, con 14 neuroni di input corrispondenti alle 14 feature, tre layer nascosti rispettivamente da 12, 16 e 8 neuroni e un layer di output con tanti neuroni quante le classi nel dataset utilizzato.

Linear Discriminant Analysis - classificatore lineare basato su metodi statistici e regola di Bayes.

Gradient Boosting - classificatore ensemble che combina 100 alberi di decisione che vengono utilizzati in modo sequenziale, in modo che ogni classificatore della catena sia allenato sugli errori residui del modello precendente.

Bagging Classifier - classificatore ensemble che combina 10 alberi di decisione indipendenti allenati su sottoporzioni differenti del dataset, e fornisce in output un risultato che è la media dei singoli classificatori; si differenzia dai random forest in quanto per ogni nodo tutte le feature sono considerate so non solo un sottoinsieme.

Voting Classifier - classificatore ensemble con votazione di tipo hard, che combina alcuni dei classificatori precedenti, ossia: random forest, linear discriminant analysis, gradient boosting e bagging classifier.

Per analizzare lo scenario è stato quindi creato un dataset contenente 6 differenti classi, una di funzionamento corretto e le restanti rappresentanti ciascuna una singola anomalia, differente dalle altre. Per ciascuna classe sono stati raccolti 200 segnali di input.
I guasti causati sono:

1. Allentamento di una delle due viti che fissa il cuscinetto su cui è posizionato l'accelerometro;

2. Allentamento di una delle due viti che fissa il cuscinetto in posizione opposta a quello con l'accelerometro, cioè all'altro estremo dell'albero;

3. Allentamento di una delle due viti che fissa il cuscinetto in prossimità del motore, cioè il cui albero prende il moto dalla cinghia;

4. Creazione di attrito sul nastro, ostacolandone il moto;

5. Rallentamento della velocità del motore a 580 step al secondo, quando la velocità corretta si suppone 600.

Per ogni modello di classificazione si è utilizzato il 50% del dataset per il training e il rimanente per il testing.
I risultati ottenuti sono riportati per ciascun algoritmo, evidenziando le performance in termini di *accuracy*, *precision*, *recall*, *F1 score*. I valori di precisione e di recall complessivi sono ottenuti calcolando i valori singoli per ogni classe e facendone una media. Questo procedimento è possibile in quanto le classi del dataset sono bilanciate.
In più si evidenziano i valori specifici per la classe di normale funzionamento. Avere una alta precisione sulla classe di *Ok* significa infatti essere confidenti che quando il sistema classifica lo stato attuale come corretto la macchina stia funzionando davvero correttamente, ed evita guasti improvvisi.
Mentre una alta recall per la classe *Ok* significa ridurre i casi in cui si evidenzia un guasto che in realtà non è presente sulla macchina causando interventi di manutenzione non necessari.
I risultati elencati nella seguente tabella sono una media dei valori ottenuti in tre esecuzioni per ogni algoritmo.

Modello	Accuracy	Precision	Recall	F1	PrecOk	RecOk
Random Forest	0.993	0.993	0.993	0.993	0.990	0.990
Neural Network	0.986	0.986	0.986	0.986	0.968	0.989
Lin. Disc. An.	0.945	0.950	0.950	0.944	1.000	0.760
Gradient Boosting	0.981	0.982	0.982	0.981	0.950	0.971
Bagging Classifier	0.980	0.980	0.979	0.979	0.967	0.927
Voting Classifier	0.988	0.989	0.988	0.988	1.000	0.969

Come si deduce dai dati di valutazione ottenuti, i risultati sono stati molto positivi, per tutti i modelli che sono stati utilizzati.

A questo punto si è deciso di estendere l'obiettivo combinando i singoli guasti per ottenere nuovi stati in cui sono presenti multiple anomalie. Gli stati totali nel nuovo dataset sono pertanto 11, di cui solo uno di normale funzio-

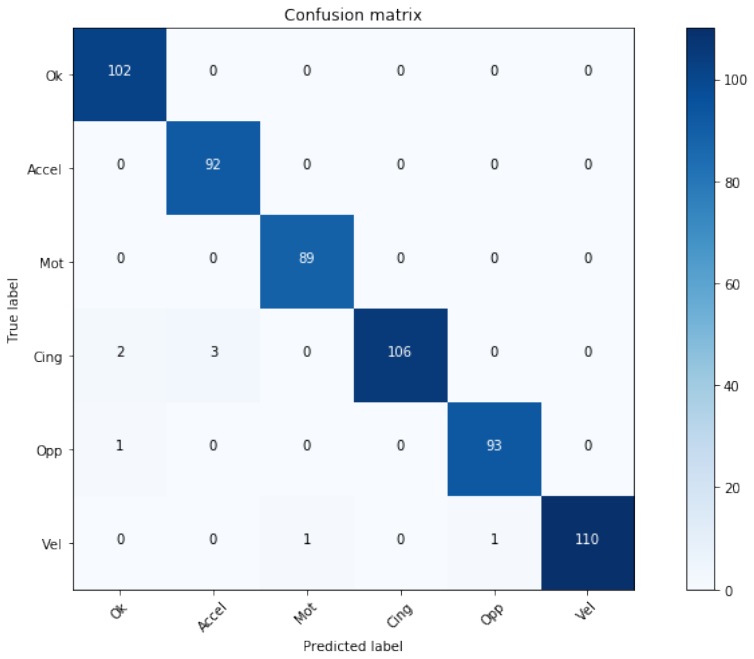


Figura 5.2: Matrice di confusione ottenuta utilizzando il Voting Classifier con il dataset a sei classi

namento. L'approccio è rimasto uguale al caso precedente. Si riportano di seguito i risultati di questo secondo caso.

Modello	Accuracy	Precision	Recall	F1	PrecOk	RecOk
Random Forest	0.955	0.956	0.955	0.955	0.988	0.960
Neural Network	0.930	0.934	0.931	0.931	0.949	0.742
Lin. Disc. An.	0.903	0.909	0.910	0.907	1.000	0.750
Gradient Boosting	0.964	0.966	0.964	0.965	0.989	0.916
Bagging Classifier	0.956	0.951	0.956	0.955	0.896	0.970
Voting Classifier	0.963	0.966	0.963	0.964	0.989	0.970

Rispetto al caso precedente i valori delle performance sono calati solamente di qualche centesimo, dimostrando quindi che è possibile individuare con buona precisione anche la presenza di guasti multipli.
Come risulta evidente dalla matrice di confusione di Fig 5.3, a parte qualche

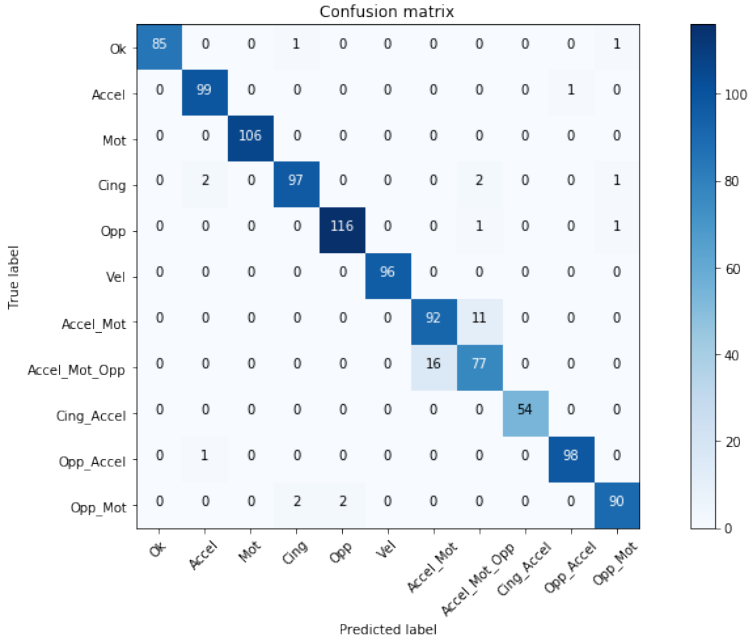


Figura 5.3: Matrice di confusione ottenuta utilizzando il Voting Classifier con il dataset a undici classi

errore sporadico, la maggior parte delle classificazioni incorrette avviene confondendo due classi molto simili. La prima è quella in cui si allentano le viti del cuscinetto con l'accelerometro e di quello vicino al motore e la seconda è quella in cui, oltre ai due dello stato precedente, si allentano le viti anche del cuscinetto opposto a quello con l'accelerometro.

Il terzo ed ultimo scenario che si è preso in considerazione per i problemi di classificazione riguarda un primo approccio ad abilitare servizi di prognostica. Si vuole infatti riuscire a classificare diversi stati di avanzamento di uno stesso guasto.
Il guasto preso in considerazione è l'allentamento della vite che fissa il cuscinetto con l'accelerometro. Sono stati eseguiti campionamenti tramite i sensori causando un allentamento della vite. L'allentamento è stato effettuato e monitorato con l'ausilio di una chiave dinamometrica, applicando una coppia di serraggio da 3N·m fino 0N·m, creando quindi un dataset suddviso

in 8 differenti classi, rappresentanti il progressivo allentamento. Anche in questo caso per ogni classe sono presenti 200 input e si sono utilizzati gli stessi modelli.
I risultati ottenuti sono riportati di seguito.

Modello	Accuracy	Precision	Recall	F1
Random Forest	0.906	0.906	0.906	0.905
Neural Network	0.910	0.911	0.910	0.910
Lin. Disc. An.	0.867	0.871	0.868	0.865
Gradient Boosting	0.920	0.920	0.919	0.919
Bagging Classifier	0.912	0.913	0.911	0.912
Voting Classifier	0.926	0.926	0.926	0.926

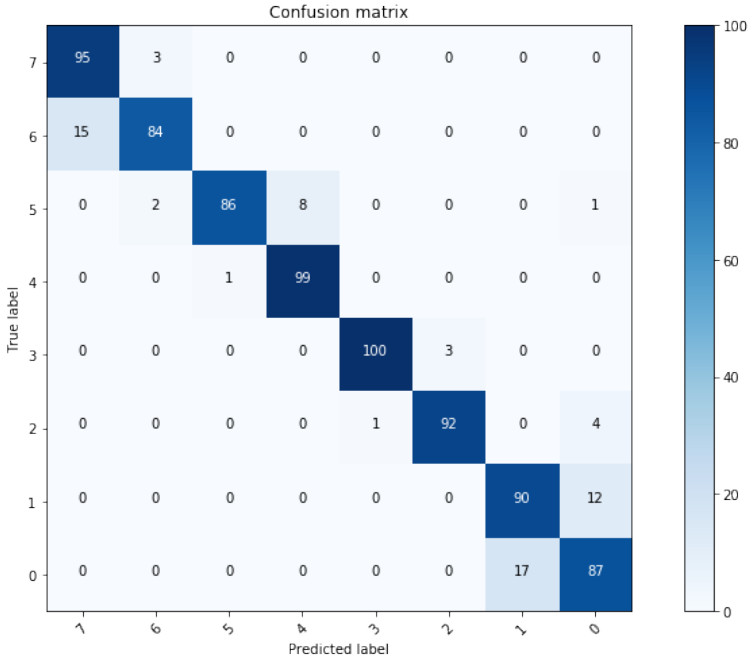


Figura 5.4: Matrice di confusione ottenuta utilizzando il Voting Classifier con il dataset di guasto progessivo a otto classi

Da una analisi dei risultati si nota che la precisione di classificazione cala leggermente rispetto ai due scenari precedenti, ma rimane comunque ad una accuratezza di circa 92% per il migliore tra i classificatori.

Inoltre osservando la matrice di confusione di Fig. 5.4, risulta evidente che la maggior parte delle classificazioni errate avviene comunque fra classi vicine, ossia fra livelli di guasto adiacenti. Questo è positivo in termini di analisi per future applicazioni di prognostica.

5.2.1 Osservazioni

Grazie ai test descritti in questa sezione è stato possibile dimostrare che modelli di machine learning sono in grado di distinguere differenti stati di funzionamento della macchina tramite i valori raccolti dai sensori scelti.
In particolare si sono ottenuti risultati positivi anche per quanto riguarda il riconoscimento di guasti multipli e la determinazione dello stato di avanzamento di uno stesso guasto, utile in particolare come base per la realizzazione di servizi di prognostica.

Tuttavia si sono individuate due pesanti criticità.

I modelli di classificazione infatti richiedono un dataset di training in cui devono essere presenti degli esempi relativi a tutte le classi che si vogliono poi riconoscere in fase di testing. Ciò significa che la macchina va effettivamente portata in stato di guasto per la creazione del dataset. E se nello scenario costruito appositamente per il progetto di tesi è stato facile ottenerlo, lo stesso non vale per casi d'uso reali in cui si vuole applicare il sistema a macchine già in produzione. E' un procedimento troppo lungo e costoso per essere effettuato in linee produttive reali.

Inoltre i modelli di classificazione in fase di testing sono limitati ad etichettare gli input che ricevono con una delle classi presenti nel dataset. Ciò significa che se si presenta una nuova tipologia di anomalia sulla macchina, essa sarà per forza fatta rientrare in una delle casistiche note anche se i pattern del particolare input si differenziano da ciascuna di esse.

Si ritene pertanto non sufficiente un modello di classificazione, ma si procede ad analizzare modelli per problemi di anomaly detection.

5.3 Anomaly Detection

Una *anomalia* è definita come qualcosa che differisce da ciò che è normale o regolare. L'idea alla base di questo approccio è proprio quello di riuscire a definire le caratteristiche o i pattern di elementi ritenuti normali per poi evidenziare quando qualcosa si distanzia da questa modellazione.

Applicandolo al caso di studio, si vuole realizzare un modello allenato esclusivamente su segnali raccolti durante il normale funzionamento della macchina, per poi riuscire ad individuare una generica anomalia. Ovviamente si perde la possibilità di distinguere i differenti guasti, ma permette di risolvere le criticità evidenziate per i modelli di classificazione, ossia la necessità di provocare fisicamente i danni sulla macchina e la restrizione a casi di guasto noti.

L'algoritmo utilizzato per questo approccio è **One-Class SVM**, un particolare tipo di support vector machine specializzato per compiti di *novelty detection*, ossia di riconoscimento di eventi rari o mai incontrati. Supponendo di usare segnali di buono stato come training, il modello avrà l'obiettivo di individuare come novità i segnali raccolti in stato di guasto.
Nello specifico si è usato un one-class SVM con kernel polinomiale, con parametri `nu` a 0.55 e `gamma` a 0.001, che è la configurazione che ha mostrato i migliori risultati.

Per verificare la capacità di distinguere le situazioni di guasto con anomaly detection si è utilizzato lo stesso dataset usato per i test della classificazione, tuttavia si è allenato il modello utilizzando esclusivamente i 200 segnali raccolti con la macchina in buono stato.
Per la fase di testing si sono utilizzati i dati raccolti in situazioni con gua-

sti come input per one-class SVM, e si è ottenuto in output la distanza dal margine costruito dal modello durante il training.

I risultati ottenuti sono riportati nella tabella sottostante, che mostra per ciascuna delle classi di guasto le distribuzioni delle distanze ottenute. Ciascuna è composta da 200 esempi.
Per facilitare l'interpretazione dei valori di output, l'unità di misura della distanza nella tabelle è la deviazione standard rispetto alla distribuzione delle distanze degli esempi di training, ossia dello stato normale.

Guasto	Avg Distance	Max Distance	Min Distance	Std Distance
Accel	44.31	46.37	42.07	0.76
Motor	10.77	13.33	7.30	1.11
Cing	1.72	41.81	0.08	4.80
Opp	3.31	4.66	1.37	0.62
Vel	7.26	9.99	4.87	1.09

Dall'analisi dei risultati si evince che la rilevazione dello stato di anomalia è possibile ma è fortemente legato alla tipologia di anomalia stessa. Ovviamente la più facile da riconoscere e quella che si distanzia di più è l'allentamento del cuscinetto su cui è posizionato l'accelerometro, in quanto è il guasto che ha i maggiori effetti sui dati raccolti dalla sensoristica. Il guasto che risulta più difficile da individuare con questo metodo è invece l'attrito sulla cinghia.

Per poter introdurre la capacità di distinguere fra le differenti anomalie utilizzando metodi di anomaly detection è necessario utilizzare multiple istanze di one-class SVM.
Ciascun modello deve essere allenato su una specifica classe in modo tale da associare a ogni input che riceve una distanza, che rappresenta un valore di similarità rispetto agli esempi di training. Per ottenere una classificazione di un nuovo segnale lo si utilizzerà come input per tutti i modelli, in modo da ottenere un valore di distanza rispetto a tutte le etichette di guasto note in

quell'istante e gli si assegna l'etichetta a cui corrisponde distanza minore.

Si è testato questo approccio con il dataset contenente solo i singoli guasti, lo stesso usato anche nel caso precedente. Si sono dunque istanziati 6 differenti modelli, uno per classe, ciascuno allenato con 100 esempi. I rimanenti 100 input per ogni classe sono stati utilizzati come test set.
I risultati ottenuti sono riportati di seguito.

Filtro	Accuracy	Precision	Recall	F1	PrecOk	RecOk
No Filtri	0.746	0.809	0.746	0.752	0.580	0.290
Mov. Avg. (w=3)	0.480	0.627	0.480	0.456	1.000	0.693
Med. Filt. (w=5)	0.648	0.628	0.640	0.630	0.641	0.535
LowPass (1400Hz)	0.805	0.809	0.805	0.803	0.680	0.521

In tabella è riportato anche il metodo di preprocessing utilizzato sul segnale di vibrazione in quanto si è notato che ha una forte influenza sui risultati, e si evidenzia in particolare il trade off fra i risultati di classificazione complessivi e i valori di precisione e recall sulla classe *Ok*.
In generale ciò che si nota è che le prestazioni sono calate rispetto all'utilizzo dei metodi di classificazione descritti nella sezione precedente.
Dalle matrici di confusione si nota inoltre che la classe *Ok* viene confusa spesso con la classe *Cinghia*, come era lecito aspettarsi visti i risultati del test precedente.

Estendendo il test introducendo anche gli stati a guasti multipli si evidenzia un ulteriore peggioramento delle performance, con l'accuratezza che anche nei casi migliori rimane poco sotto il 60%.

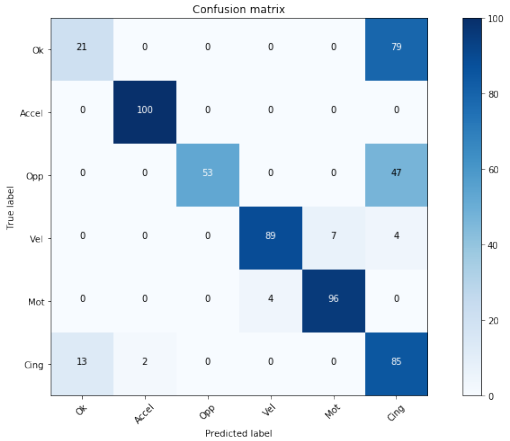


Figura 5.5: Senza Filtri

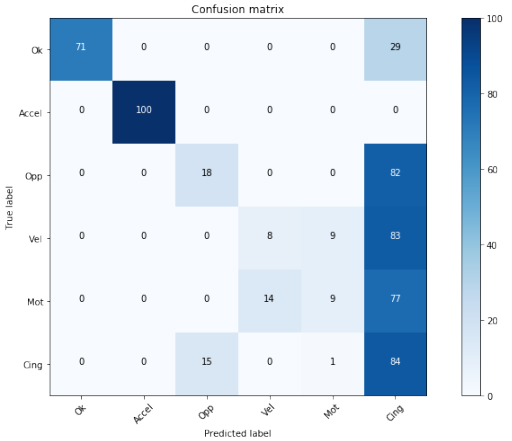


Figura 5.6: Media Mobile

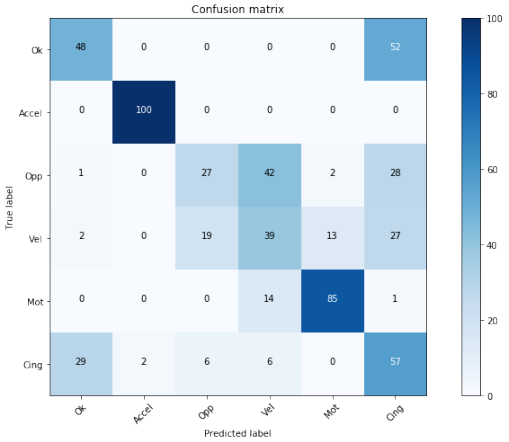


Figura 5.7: Filtro Mediano

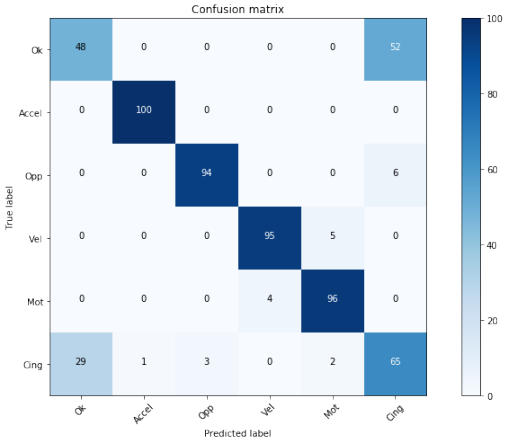


Figura 5.8: Filtro Passa Basso

Filtro	Accuracy	Precision	Recall	F1	PrecOk	RecOk
No Filtri	0.561	0.670	0.599	0.544	0.666	0.220
Mov. Avg. (w=3)	0.382	0.570	0.436	0.374	1.000	0.800
Med. Filt. (w=5)	0.398	0.509	0.407	0.395	0.531	0.340
LowPass (1400Hz)	0.576	0.672	0.591	0.561	0.657	0.250

5.3.1 Osservazioni

I test effettuati riguardanti l'anomaly detection hanno dimostrato che è possibile riconoscere con buona precisione quando si entra in uno generico stato di anomalia utilizzando in fase di training solamente esempi raccolti durante il normale funzionamento.
Lo svantaggio di questo approccio è però quello di perdere la distinzione e la classificazione fra differenti guasti.

Per superare questa limitazione si è provato un metodo che combina differenti modelli, ciascuno allenato su una specifica classe di guasto.
Il pregio principale di questa tecnica è quella di poter effettuare training in modo separato fra i modelli, e combinare i loro risultati solo in un secondo momento. Ciò significa che per introdurre una nuova classe di anomalia è sufficiente creare un nuovo one-class SVM, allenarlo sugli esempi disponibili riguardanti solamente l'anomalia in questione senza dover modificare gli altri modelli. Durante la fase di verifica delle distanze si dovrà poi solamente consultare un modello in più.

Un'ulteriore proprietà derivante da questo approccio è quella di poter identificare quando un certo input non appartiene a nessuna delle classi note. Questo è realizzabile controllando i valori delle distanze restituiti dai modelli. Se il segnale da classificare risulta essere lontano da tutti i modelli, allora con molta probabilità si tratta di una classe ancora sconosciuta.

La limitazione più grande risultano essere le prestazioni inferiori nel com-

pito di classificazione, come dimostra il confronto delle metriche utilizzate rispetto a quelle ottenute nella sezione precedente. Esse si ritengono infatti non sufficienti per l'utilizzo in scenari di CBM.

5.4 Combinazione di Classificazione e Anomaly Detection

Con lo scopo di mantenere la buona precisione dei modelli di classificazione senza perdere i vantaggi introdotti dall'anomaly detection, si propone una soluzione per problemi di CBM che combina i due tipi di modelli.

La soluzione utilizza il voting classifier presentato nella Sez. 5.2, in quanto modello di classificazione con la media di risultati migliori, e un one-class SVM per ogni stato che si vuole classificare.
La caratteristica principale della tecnica proposta è che non è necessario disporre fin dal principio di tutti i dati per il training per ogni classe, ma essi possono essere aggiunti in un secondo momento.
Il funzionamento è descritto di seguito.

- Si suppone di iniziare ad utilizzare il sistema senza aver già raccolto ed etichettato un dataset.
- Si effettua il training dello stato di normale funzionamento della macchina, raccogliendo dati dai sensori ed utilizzandoli per allenare un modello di anomaly detection implementato da un one-class SVM.
- A questo punto si può già passare in fase di testing, sfruttando il modello di anomaly detection per rilevare eventuali scostamenti dal caso normale, che verranno quindi segnalati come generica anomalia.
- In caso di anomalie riscontrate un utente del sistema può effettuare etichettatura sui singoli segnali, classificandoli con una precisa classe di guasto. In tal caso viene creato un nuovo one-class SVM, allenato

solamente sul nuovo insieme di dati etichettati. Il procedimento può essere ripetuto più volte per ogni nuovo guasto incontrato nel tempo.

- Quando sono disponibili più di un singolo modello di anomaly detection, oltre alla generica rilevazione di anomalia, è anche possibile effettuare classificazione fra gli stati noti, utilizzando il metodo descritto in Sez. 5.3, anche se il principale scopo è quello di evidenziare quando un segnale di input risulta distante da tutti i modelli, segnalando quindi la presenza in uno stato non conosciuto.

- In parallelo a quanto descritto finora, dal momento in cui si conoscono due o più classi, si utilizza anche un voting classifier, che deve essere riallenato ogni volta che l'utente classifica nuovi input. Il classificatore è utilizzato quando gli altri modelli indicano che non si tratta di una nuova classe, per distinguere il tipo specifico di anomalia.

5.4.1 Osservazioni

La soluzione riportata permette di rilevare e classificare anomalie che si presentano sulla macchina.
Elimina la necessità di avere un dataset già in partenza permettendo di partire effettuando solamente anomaly detection ed introducendo la classificazione man mano che i guasti si presentano realmente sulla macchina, utilizzandoli per riuscire a distinguerli le volte successive.

Le principali limitazioni riscontrate sono il tempo necessario ad allenare il voting classifier in fasi avanzate di test, in cui si hanno dataset corposi in termini di classi e di input. Il classificatore infatti deve essere riallenato sulla totalità del dataset ogni volta che esso viene incrementato o modificato.
Si suppone quindi che in scenari reali sia una operazione che va effettuata soltanto periodicamente, in background o in momenti in cui le macchine non sono in produzione, limitandosi ad usare i modelli di anomaly detection come

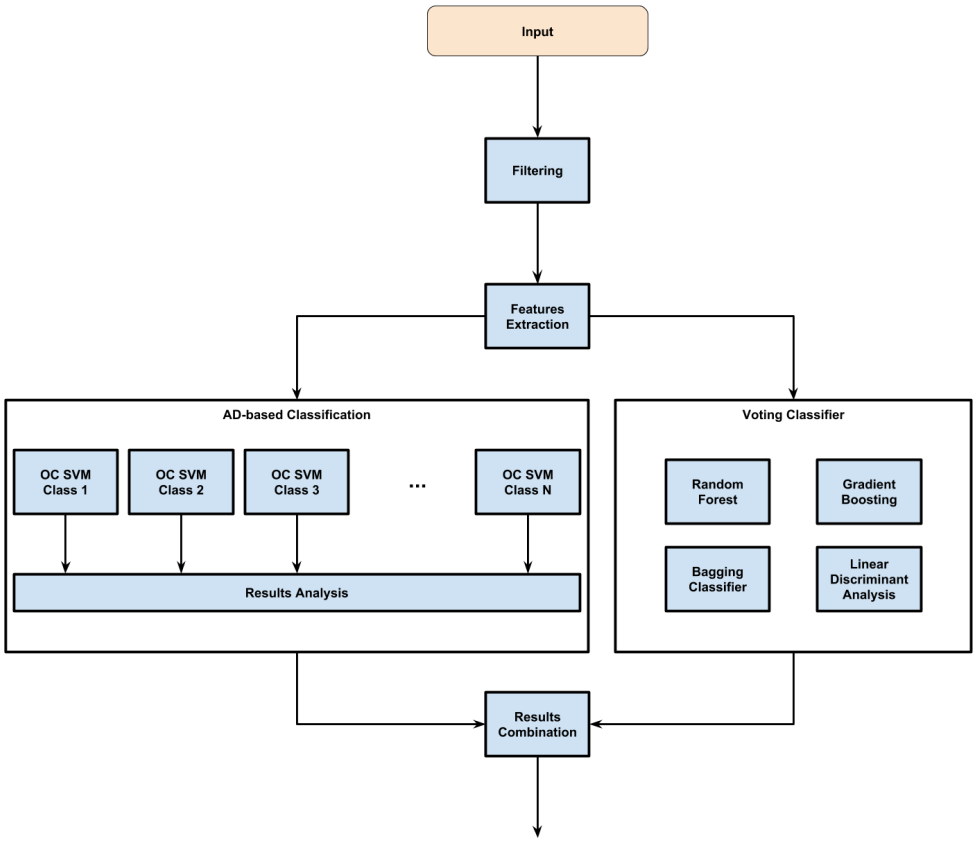


Figura 5.9: Schema complessivo del modello usato

classificatori durante il processo.

Inoltre rimane comunque la forte dipendenza dal fattore umano. Sono infatti gli utenti che effettuano le classificazioni. In particolare il problema è importante per il training della classe di *Ok*. E' fondamentale ispezionare al meglio la macchina prima di iniziare il periodo di training del normale funzionamento, per evitare di assumere come normale uno stato che in realtà non lo è.

5.5 Dettagli Implementativi

Tutto il codice necessario all'implementazione del sistema è stato scritto in Python 3.6.
In particolare si sono utilizzati i notebook di Jupyter per tutta la fase di analisi esplorativa dei dati e di sperimentazione iniziale.
Per la gestione e la manipolazione dei dataset si è utilizzata la libreria Pandas e Matplotlib per la visualizzazione grafica dei segnali di vibrazione nel dominio del tempo e delle frequenze.
Per tutti i modelli di machine learning si è sfruttata l'implementazione fornita da Scikit-Learn, ad eccezione delle reti neurali per cui si è utilizzato Keras.
Scikit-Learn è tornato utile anche per le elaborazioni delle metriche di prestazione dei vari modelli e per la creazione delle matrici di confusione.
Numpy e Scipy sono stati utilizzati per le funzionalità di estrazione delle feature e preprocessing dei segnali di vibrazione.
Infine il web server tramite il quale si può accedere alle funzionalità del sistema è realizzato con Flask.

Capitolo 6

Caso d'Uso: Manutenzione Autonoma

In questo capitolo si mostra come il sistema descritto nei capitoli precedenti possa essere utilizzato in uno scenario pratico legato alla manutenzione autonoma.

La manutenzione autonoma è un insieme di attività di manutenzione che viene effettuato sulle macchine dagli operatori e non dai manutentori. Queste attività sono tipicamente pulizie, lubrificazioni, controlli e piccole riparazioni. Ad ogni attività è associata una frequenza di esecuzione, che può quindi essere effettuata ogni turno, giorno, settimana o mese.
Le attività sono quindi suddivise in ispezioni di manutenzione autonoma, che sono sequenze di attività che vanno eseguite in uno specifico ordine.
Lo scopo ultimo della manutenzione autonoma è quello di mantenere le macchine in buono stato, prevenendo i guasti agendo fin da subito sulle piccole anomalie prima che causino malfunzionamenti di maggior entità.

L'obiettivo che ci si è posti è stato quello di realizzare un software in grado di fornire un supporto alle attività di manutenzione autonoma.

Il primo passo è stato realizzare un supporto digitale alle ispezioni, tale da fornire indicazioni su ciascuna attività da svolgere, come una descrizione testuale, immagini dei componenti coinvolti e la posizione dell'intervento sulla mappa della macchina.
La macchina utilizzata è la stessa descritta nella Sez. 4.1, e le attività inserite come elementi delle ispezioni di manutenzione sono le contromisure ai guasti descritti nella Sez. 4.1.1, ossia quelli causati per la creazione dei dataset.

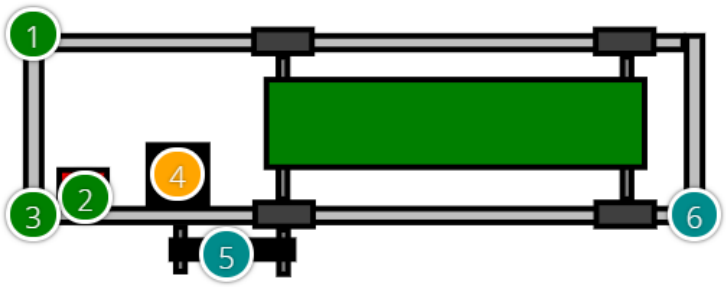


Figura 6.1: Attività di manutenzione sulla mappa della macchina, dall'interfaccia dell'applicazione per l'operatore

A questo punto si è voluto integrare il sistema di CBM alla gestione della manutenzione autonoma.
L'idea è quella di migliorare le logiche che guidano la creazione e l'esecuzione delle ispezioni. Oltre a stabilire a priori la frequenza di ciascuna attività, si vuole associare a qualcuna di esse anche un'etichetta di classificazione. Si utilizza quindi l'output fornito dai modelli di riconoscimento e classificazione di anomalie per rendere dinamica la creazione della sequenza di attività da svolgere per la prossima ispezione. Oltre a quelle da eseguire per principi di cadenza temporale, l'ispezione può contenere attività di manutenzione basate sulla condizione effettiva della macchina.

La gestione delle corrispondenze fra una classe e le attività da eseguire è con-

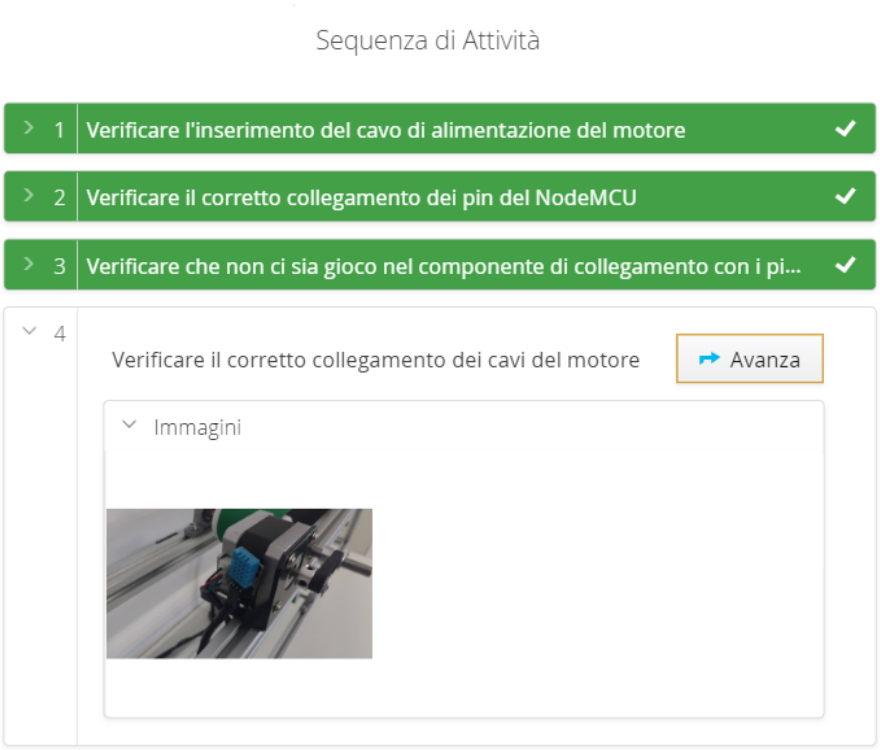


Figura 6.2: Elenco delle attività di manutenzione da svolgere, dall'interfaccia dell'applicazione per l'operatore

tenuta interamente nell'applicazione di manutenzione autonoma, ed è quindi esterna al sistema di CBM, che si limita e fornire una valutazione dello stato corrente.
Il software di manutenzione autonoma si interfaccia con i servizi di CBM attraverso il web server, richiedendo i servizi descritti nella Sez. 4.3. All'avvio dell'applicazione essa si registra presso il server del sistema, aprendo una WebSocket. La WebSocket è utilizzata come canale in modalità push, tramite cui il server notifica l'applicazione quando si rileva una anomalia nello stato della macchina.

E' stata inoltre realizzata un'ulteriore applicazione, ossia quella destinata al ruolo del manutentore. Il manutentore è infatti in grado di visualizzare nel dettaglio lo stato della macchina per come individuato dagli algoritmi

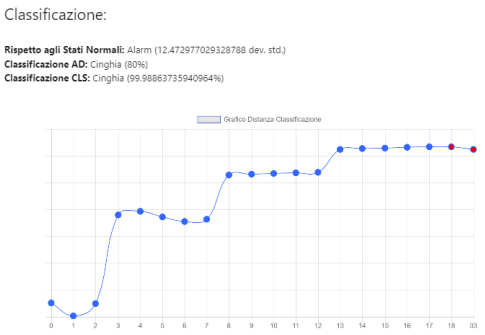


Figura 6.3: Interfaccia di valutazione dello stato

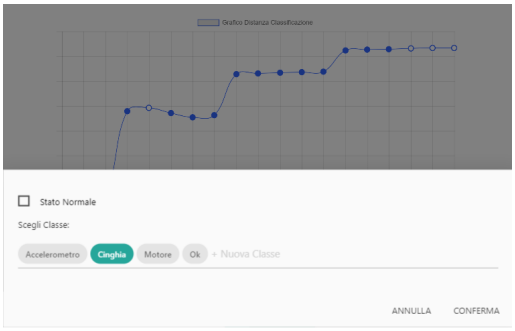


Figura 6.4: Interfaccia di classificazione

di anomaly detection e classificazione. E' inoltre in grado di avviare fasi di training, creare nuove classi ed associare una precisa etichetta a segnali di input in modo tale da permettere l'allenamento dei modelli.

Si è mostrato quindi come nel concreto le tecnologie abilitanti dell'Industry 4.0 possano supportare e migliorare i vari processi, nello specifico un particolare ambito della manutenzione, cioè quella autonoma.

Capitolo 7

Conclusioni e Sviluppi Futuri

Il presente elaborato ha descritto il progetto di tesi svolto, finalizzato alla progettazione e realizzazione di un sistema in grado di abilitare e supportare attività di manutenzione su condizione e predittiva in contesti Industry 4.0.

Il sistema sviluppato si è dimostrato essere in grado di rilevare e distinguere anomalie, abilitando quindi funzionalità di diagnostica. E' stato inoltre possibile individuare con buona precisione anche il grado di avanzamento di uno stesso guasto e ciò rappresenta una base per possibili funzionalità anche di prognostica.
Tutti i dati che sono stati utilizzati sono provenienti da una macchina reale e non da una simulazione. I sensori posizionati sulla macchina hanno raccolto grandezze fisiche durante il funzionamento della macchina, che è stata effettivamente portata in stati di guasto differenti.
I sensori che si è scelto di utilizzare sono dispositivi a basso costo e facilmente reperibili, a dimostrazione del fatto che è possibile ottenere risultati positivi anche senza strumentazione sofisticata.
Si è inoltre presentata l'architettura complessiva del sistema, che ha coperto il flusso dei dati dalla sensoristica fino all'applicazione utente. Il punto più importante dell'architettura utilizzata è il nodo che effettua la persistenza dei dati e che espone i servizi di analisi sui dati stessi tramite web API.

Questa soluzione permette di rendere indipendenti i servizi da una specifica applicazione, facilitandone il riuso e l'estensione.

Per quanto riguarda le tecniche di analisi dei dati e riconoscimento delle anomalie è stato scelto un approccio data-driven, basato su algoritmi di machine learning.
Il modello presentato integra e combina modelli di classificazione e di anomaly detection, con lo scopo di ottenere i vantaggi di entrambi. I classificatori hanno infatti mostrato ottima accuratezza nella distinzione tra stati diversi, ma necessitano di fare training su un dataset completo di tutti i guasti che si vogliono individuare. L'utilizzo in parallelo di modelli di anomaly detection permette di superare questa limitazione, in quanto sono in grado di rilevare quando un nuovo input non rientra in nessuna delle classi di guasto note.
La soluzione proposta si basa proprio su questo concetto di apprendimento progressivo, che inizia dal riconoscimento di una generica anomalia conoscendo solamente lo stato di normale funzionamento e che permette ai manutentori di incrementare l'insieme dei guasti conosciuti man mano che si presentano sulle macchine.

Infine si è presentato uno scenario reale di utilizzo del sistema, in cui l'identificazione dello stato di salute della macchina si applica a supporto delle attività di manutenzione autonoma.

Sviluppi Futuri

Si elencano di seguito possibili lavori futuri riguardo il sistema realizzato.

- Estendere i servizi offerti dal sistema in modo tale da supportare anche applicazioni di prognostica, cioè che siano in grado associare a ogni input anche una stima della vita utile rimanente, ossia del tempo rimanente prima di entrare in uno stato di guasto. Per fare ciò ser-

ve effettuare la fase di apprendimento su sequenze temporali di tipo *run-to-failure* di ciascun componente critico.

- Nella maggior parte degli scenari reali le macchine sono di grandi dimensioni. Nella fase iniziale di utilizzo del sistema quando il modello segnala una generica anomalia può essere comunque difficile per i manutentori andare a individuare il guasto. Pertanto si suppone di introdurre una divisione per zone, ciascuna con i propri sensori e i propri modelli, in esecuzione in parallelo per fornire già nelle fasi iniziali una indicazione più precisa sulle anomalie in corso.

- Rendere l'architettura più scalabile al fine di supportare la raccolta e l'analisi di dati in tempo reale anche per scenari con più macchine attive contemporaneamente.

- Testare l'utilizzo di altri modelli di anomaly detection in alternativa o in sostituzione di one-class SVM, per aumentare la precisione in classificazione ed eventualmente rendere meno determinante l'uso parallelo di un classificatore allenato sulla totalità del dataset.

Bibliografia

[1] E. Sezer, D. Romero, F. Guedea, M. Macchi, and C. Emmanouilidis, "An industry 4.0-enabled low cost predictive maintenance approach for smes," in *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pp. 1–8, IEEE, 2018.

[2] R. K. Mobley, *An introduction to predictive maintenance*. Elsevier, 2002.

[3] K. Medjaher, D. A. Tobon-Mejia, and N. Zerhouni, "Remaining useful life estimation of critical components with application to bearings," *IEEE Transactions on Reliability*, vol. 61, no. 2, pp. 292–302, 2012.

[4] A. K. Mahamad, S. Saon, and T. Hiyama, "Predicting remaining useful life of rotating machinery based artificial neural network," *Computers & Mathematics with Applications*, vol. 60, no. 4, pp. 1078–1087, 2010.

[5] A. K. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical systems and signal processing*, vol. 20, no. 7, pp. 1483–1510, 2006.

[6] G. W. Vogl, B. A. Weiss, and M. Helu, "A review of diagnostic and prognostic capabilities and best practices for manufacturing," *Journal of Intelligent Manufacturing*, pp. 1–17, 2016.

[7] H. M. Elattar, H. K. Elminir, and A. Riad, "Prognostics: a literature review," *Complex & Intelligent Systems*, vol. 2, no. 2, pp. 125–154, 2016.

[8] P. Jahnke, "Machine learning approaches for failure type detection and predictive maintenance," *Technische Universität Darmstadt*, vol. 19, 2015.

[9] A. Lahmadi, L. Terrissa, and N. Zerhouni, "A data-driven method for estimating the remaining useful life of a composite drill pipe," in *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, pp. 192–195, IEEE, 2018.

[10] M. Canizo, E. Onieva, A. Conde, S. Charramendieta, and S. Trujillo, "Real-time predictive maintenance for wind turbines using big data frameworks," *arXiv preprint arXiv:1709.07250*, 2017.

[11] A. Mosallam, K. Medjaher, and N. Zerhouni, "Data-driven prognostic method based on bayesian approaches for direct remaining useful life prediction," *Journal of Intelligent Manufacturing*, vol. 27, no. 5, pp. 1037–1048, 2016.

[12] L. Liu, D. Liu, Y. Zhang, and Y. Peng, "Effective sensor selection and data anomaly detection for condition monitoring of aircraft engines," *Sensors*, vol. 16, no. 5, p. 623, 2016.

[13] D. Jung, Z. Zhang, and M. Winslett, "Vibration analysis for iot enabled predictive maintenance," in *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*, pp. 1271–1282, IEEE, 2017.

[14] A. Kanawaday and A. Sane, "Machine learning for predictive maintenance of industrial machines using iot sensor data," in *Software Engineering and Service Science (ICSESS), 2017 8th IEEE International Conference on*, pp. 87–90, IEEE, 2017.

[15] J. Yan, Y. Meng, L. Lu, and L. Li, "Industrial big data in an industry 4.0 environment: Challenges, schemes, and applications for predictive maintenance," *IEEE Access*, vol. 5, pp. 23484–23491, 2017.

[16] S. Langarica, C. Rüffelmacher, and F. Núñez, "An industrial internet platform for real-time fault detection in industrial motors," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, pp. 351–356, IEEE, 2018.

[17] T. Kuzin and T. Borovicka, "Early failure detection for predictive maintenance of sensor parts.," in *ITAT*, pp. 123–130, 2016.

[18] S. Poyhonen, P. Jover, and H. Hyotyniemi, "Signal processing of vibrations for condition monitoring of an induction motor," in *First International Symposium on Control, Communications and Signal Processing, 2004.*, pp. 499–502, IEEE, 2004.

[19] M. Guo, L. Xie, S.-Q. Wang, and J.-M. Zhang, "Research on an integrated ica-svm based framework for fault diagnosis," in *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483)*, vol. 3, pp. 2710–2715, IEEE, 2003.

[20] K.-Y. Chen, L.-S. Chen, M.-C. Chen, and C.-L. Lee, "Using svm based method for equipment fault detection in a thermal power plant," *Computers in industry*, vol. 62, no. 1, pp. 42–50, 2011.

[21] M. das Chagas Moura, E. Zio, I. D. Lins, and E. Droguett, "Failure and reliability prediction by support vector machines regression of time series data," *Reliability Engineering & System Safety*, vol. 96, no. 11, pp. 1527–1534, 2011.

[22] K. He and X. Li, "A quantitative estimation technique for welding quality using local mean decomposition and support vector machine," *Journal of Intelligent Manufacturing*, vol. 27, no. 3, pp. 525–533, 2016.

[23] Z. Tian, "An artificial neural network approach for remaining useful life prediction of equipments subject to condition monitoring," in *2009 8th International Conference on Reliability, Maintainability and Safety*, pp. 143–148, IEEE, 2009.

[24] Z. Zhang, Y. Wang, and K. Wang, "Fault diagnosis and prognosis using wavelet packet decomposition, fourier transform and artificial neural network," *Journal of Intelligent Manufacturing*, vol. 24, no. 6, pp. 1213–1227, 2013.

[25] J. Zarei, M. A. Tajeddini, and H. R. Karimi, "Vibration analysis for bearing fault detection and classification using an intelligent filter," *Mechatronics*, vol. 24, no. 2, pp. 151–157, 2014.

[26] M. Butler and V. Kešelj, "Data mining techniques for proactive fault diagnostics of electronic gaming machines," in *Canadian Conference on Artificial Intelligence*, pp. 366–369, Springer, 2010.

[27] H. Li, D. Parikh, Q. He, B. Qian, Z. Li, D. Fang, and A. Hampapur, "Improving rail network velocity: A machine learning approach to predictive maintenance," *Transportation Research Part C: Emerging Technologies*, vol. 45, pp. 17–26, 2014.

[28] R. Prytz, S. Nowaczyk, T. Rögnvaldsson, and S. Byttner, "Analysis of truck compressor failures based on logged vehicle data," in *9th International Conference on Data Mining, Las Vegas, Nevada, USA, July 22–25, 2013*, CSREA Press, 2013.

[29] D. A. Tobon-Mejia, K. Medjaher, N. Zerhouni, and G. Tripot, "A data-driven failure prognostics method based on mixture of gaussians hidden markov models," *IEEE Transactions on reliability*, vol. 61, no. 2, pp. 491–503, 2012.

[30] Q. Liu and M. Dong, "Online health management for complex nonlinear systems based on hidden semi-markov model using sequential monte carlo methods," *Mathematical Problems in Engineering*, vol. 2012, 2012.

[31] M. Dong and D. He, "Hidden semi-markov model-based methodology for multi-sensor equipment health diagnosis and prognosis," *European Journal of Operational Research*, vol. 178, no. 3, pp. 858–878, 2007.

[32] M. Sanayha and P. Vateekul, "Fault detection for circulating water pump using time series forecasting and outlier detection," in *2017 9th International Conference on Knowledge and Smart Technology (KST)*, pp. 193–198, IEEE, 2017.

[33] M. Samanazari, A. Ramezani, S. Rajabi, and A. Chaibakhsh, "Using one-class support vector machine for the fault diagnosis of an industrial once-through benson boiler," *The Modares Journal of Electrical Engineering*, vol. 12, no. 3, pp. 39–45, 2015.

[34] S. Nair, "Predictive maintenance in a railway scenario using one-class support vector machines," Master's thesis, TU Darmstadt, Oct. 2016.

[35] Z. Gao, C. Ma, and Y. Luo, "Rul prediction for ima based on deep regression method," in *2017 IEEE 10th International Workshop on Computational Intelligence and Applications (IWCIA)*, pp. 25–31, IEEE, 2017.

[36] "Scikit-Learn." `https://scikit-learn.org/stable/index.html`.

[37] "Azure AI Guide for Predictive Maintenance Solutions." `https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/cortana-analytics-playbook-predictive-maintenance`.

[38] "PHM08 Challenge Dataset." `https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#phm08_challenge`.

[39] "NASA Turbofan Dataset." `https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#turbofan`.

[40] "NASA Bearing Dataset."
https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#bearing.

[41] "NASA FEMTO Bearing Dataset."
https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#femto.

[42] "Microsoft Azure Telemetry Dataset."
https://gallery.azure.ai/Notebook/Predictive-Maintenance-Modelling-Guide-R-Notebook-1.

[43] "Modular and Portable Conveyor Belt. Speed Control by Arduino."
https://www.instructables.com/id/MODULAR-PORTABLE-CONVEYOR-BELT-SPEED-CONTROL-BY-AR/.

[44] "DHT11 Datasheet."
https://www.mouser.com/ds/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf.

[45] "ACS712 Datasheet."
https://www.allegromicro.com/~/media/files/datasheets/acs712-datasheet.ashx.

[46] "LIS3DH Datasheet."
https://www.st.com/resource/en/datasheet/cd00274221.pdf.