UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea magistrale in Ingegneria Elettronica

Studio e sviluppo di sistemi di manutenzione predittiva basati su industrial IoT

Study and development of predictive maintenance systems on Industrial IoT

Relatore:

Prof. **Paola Pierleoni**

Tesi di laurea di:

Giulia Temperini

Correlatore:

Ing. **Luisiana Sabbatini**

A.A. 2020/2021

ABSTRACT

Studio e sviluppo di sistemi di manutenzione predittiva basati su industrial IoT

Il progressivo sviluppo delle tecnologie di intelligenza artificiale (AI), unito alla recente proliferazione di sensori economici ma ragionevolmente precisi ha comportato, nelle ultime due decadi, un cambio di paradigma nell'ambito della manutenzione industriale. L'adozione della manutenzione predittiva (*Predictive Maintenance*, PdM) in una *smart factory* viene ad essere la soluzione vincente per sviluppare soluzioni mirate tali da ridurre notevolmente i costi di riparazione e dei fermi macchina. Tale strategia prevede il continuo monitoraggio degli asset industriali cosicché la mole di dati generata vada a divenire la base per l'identificazione *real-time* dello stato di salute dei macchinari, comportando in questo modo la possibilità di prevedere la loro vita residua utile (RUL) e la definizione di interventi manutentivi ad hoc. La presente ricerca di Tesi si pone in primo luogo l'obbiettivo di fornire una descrizione ed un confronto dei principali approcci manutentivi, sottolineando come ciascuno di essi si riveli vincente o meno a seconda dell'asset o applicazione considerata. In seguito si entra nel merito della manutenzione predittiva, approfondendo tre suoi fondamentali fattori: *a)* le tecnologie di *condition monitoring* in ambito industriale, *b)* l'architettura IoT volta alla corretta acquisizione, trasmissione ed archiviazione dei dati generati dai sensori, e *c)* i principali approcci matematici e statistici che a partire da tali dati conducono all'estrazione dello stato di salute del macchinario. In merito a quest'ultimo aspetto, la ricerca di Tesi vede come caso di studio un dataset pubblico denominato *Milling dataset* e comprendente i dati raccolti da una macchina fresatrice. Viene proposto a riguardo un framework di manutenzione predittiva per la stima della RUL del tipo *data-driven* e basato su *deep learning*. Il framework si compone di due fasi: nella fase offline in seguito al *training* in modalità non supervisionata di un *autoencoder* bidirezionale LSTM e un modello di regressione lineare, si va ad estrarre dalle letture dei sensori lo stato di salute del macchinario, definendo così una libreria di andamenti storici del processo di degrado. Nella fase online, forniti in ingresso i dati di una lavorazione di test si va ad estrarre da essi l'andamento temporale dello stato di salute. La relativa stima della RUL si basa sul confronto di tale andamento con gli andamenti di *training* memorizzati nel sistema. Tale approccio prognostico, rispetto alle più comuni tecniche di *deep learning*, si rivela maggiormente performante nei casi in cui la quantità di *failure data* sia notevolmente minore rispetto ai *health data* e quando la dimensione dell'intero *training set* sia limitata. I risultati ottenuti mostrano come l'approccio prognostico adottato sia in linea e in taluni casi superi i più moderni metodi di prognostica presenti in letteratura.

ABSTRACT

Study and development of predictive maintenance systems on Industrial IoT

The progressive development of artificial intelligence technologies (AI), combined with the recent proliferation of inexpensive but reasonably accurate sensors has led to a paradigm shift in the field of industrial maintenance in the last two decades. The advent and adoption of *Predictive Maintenance* (PdM) in a *smart factory* is the winning solution to provide for the development of targeted solutions so as to significantly reduce repair costs and downtime. This strategy provides for the continuous monitoring of industrial equipment so that the amount of data generated daily becomes the basis for real-time identification of the state of health of the machinery, thus making it possible to predict their *remaining useful life* (RUL) and the definition of *ad hoc* maintenance interventions. This thesis research aims primarily to provide a description and a comparison of the main maintenance approaches, underlining how each of them turns out to be successful or not depending on the asset or application considered. Then we enter into the merits of predictive maintenance, deepening three of its fundamental factors: a) the condition monitoring technologies integrated in the industrial field, b) the IoT architecture aimed at the correct acquisition, transmission and storage of data generated by the sensors, and c) the main mathematical and statistical approaches which, starting from such data, lead to the identification of the state of health of the machinery. With regard to the latter aspect, this Thesis research considers, as a case study, a public dataset called Milling dataset and including data collected by a milling machine. In this regard, a predictive maintenance framework for RUL estimation of the data-driven type and based on deep learning is proposed. The framework consists of two phases: in the offline phase following the training in unsupervised mode of a bidirectional LSTM autoencoder and a linear regression model, the state of health of the machinery is extracted and identified from the sensor readings, constituting a library of possible trends in the degradation process. In the online phase, once the data relating to a test process is input, the state of health and the corresponding temporal trend are extracted from them. The relative estimation of the RUL is based on the comparison of this trend with the training trends and therefore historical and stored in the system. This prognostic approach, compared to the more common deep learning techniques, is more effective in cases where the amount of failure data is significantly lower than health data and when the size of the entire training set is limited. The results obtained show how the prognostic approach adopted is in line with and in some cases outperforms the most modern prognostic methods found in the literature.

Indice

Capitolo 1

Introduzione

Nel futuro delle Cose connesse, dell'Internet of Things, i numeri sono di quelli che impressionano: miliardi di sensori per milioni di miliardi di dati. La scommessa dell'IoT, in fin dei conti, sta tutta qui: nella capacità di trasformare questi dati in informazioni, le informazioni in analisi, le analisi in azioni in grado di potenziare i processi di business.
Tra la moltitudine di ambiti toccati dalla "rivoluzione" dell'IoT, uno di particolare interesse è quello relativo alla manutenzione. Il settore industriale odierno richiede infatti così elevati standard di efficienza, produttività e qualità che un guasto ad un macchinario di produzione non può più considerarsi un'ipotesi imprevedibile. Tutto ciò rende quindi necessaria l'adozione di pratiche che consentano di ridurre al minimo l'occorrenza di guasti e la conseguente riduzione se non azzeramento dei tempi di fermo macchina. Inoltre, secondo un report del 2021 [1] il 41% delle aziende assegna oltre il 10% del proprio budget annuale alle attività di manutenzione in quanto il 32% di esse considera quest'ultima come un centro di profitto, che fornisce informazioni fruibili, risultati misurabili e un valido ritorno sull'investimento (*return on investment*, ROI) al proprio impianto. Nonostante ciò tuttavia, il 93% delle aziende descrive i propri processi di manutenzione come non molto efficienti definendo la necessità di miglioramenti. In questo scenario, l'avvento della manutenzione predittiva (*predictive maintenance*, PdM) impone un cambio di paradigma, dalla reazione alla prevenzione, con un sguardo puntato non più a ciò che è accaduto, ma a ciò che potrebbe accadere in assenza di interventi specifici e mirati. L'obiettivo della strategia predittiva è quello di raccogliere, tramite un continuo monitoraggio, informazioni passate, presenti e (previste) future sulle condizioni dei componenti, processi di produzione e macchinari per utilizzarle per rilevarne il degrado, diagnosticare eventuali anomalie, prevedere la loro evoluzione futura dello stato di salute e la loro vita utile residua (RUL, ovvero per quanto tempo il componente o l'asset può

continuare a svolgere la funzione prevista). L'idea alla base è quella secondo cui il verificarsi di un guasto, tipicamente, non avviene in maniera istantanea ma è frutto del deterioramento nel tempo delle condizioni di un determinato componente. Il monitoraggio di alcuni parametri può quindi denunciare condizioni di funzionamento anomale e segnalare la necessità di un intervento. Chiari sintomi di un funzionamento fuori standard sono ad esempio variazioni di temperatura, presenza di particelle di usura nei lubrificanti o un spettro di vibrazioni anomalo.

Idealmente, la previsione accurata dell'evoluzione futura dello stato di salute del componente consente un uso del macchinario sempre in uno stato di saluto e la possibilità di pianificare interventi di manutenzione *just-in-time* ovvero, nel momento più conveniente ed economico possibile. Pertanto, tra i benefici attesi della PdM si riscontrano l'aumento dell'affidabilità e della disponibilità dell'impianto e della durata delle apparecchiature, un minor numero di incidenti con impatto negativo sull'ambiente e la gestione ottimizzata delle parti di ricambio, massimizzando quindi i profitti di produzione e al contempo minimizzando tutti i costi e le perdite.

Storicamente, il concetto di manutenzione predittiva viene integrato nel mondo industriale già a partire dagli anni '90, anche se in realtà come riporta Control Engineering [2], *"L'inizio della manutenzione predittiva (PdM) potrebbe essere stato quando un meccanico ha accostato per la prima volta l'orecchio al manico di un cacciavite, ha toccato l'altra estremità di una macchina e ha dichiarato che sembrava che un cuscinetto stesse andando male".*

Malgrado ciò però, solo di recente e grazie alle tecnologie emergenti sempre più performanti ed economiche, tale strategia è diventata accessibile ad una vasta platea di industrie. Tenendo conto infatti, dei pilastri su cui si fonda la manutenzione predittiva, le moderne tecnologie hanno un maggior potenziale nel rilevare, isolare e identificare l'insorgenza di anomalie nel funzionamento di apparecchiature e componenti, monitorare e prevedere la progressione dei guasti e fornire un supporto decisionale o automatico per lo sviluppo di programmi di manutenzione.

Nello specifico, le tecnologie emergenti che hanno permesso la valorizzazione di questa manutenzione sono [3]:

- **IoT**: lo sviluppo della sensoristica, avendo portato alla proliferazione di sensori sempre più performanti e a basso costo, consente di raccogliere una quantità enorme e crescente di dati da sorgenti multiple (macchinari, componenti, ambiente).
- **Big Data**: l'enorme quantità di dati, generata e raccolta in una *smart factory*, ha

richiesto e condotto allo sviluppo di avanzate tecnologie di elaborazione dei big data industriali. Quest'ultimi infatti, sono stati definiti avere le caratteristiche delle 5V, *volume*, *velocità*, *varietà*, *veridicità*, *valore*, caratteristiche che hanno messa in discussione e reso obsolete le tradizionali tecniche di *signal-processing* [4]. Oggigiorno, l'elaborazione di big data industriali coinvolge dati per la riduzione della dimensionalità, l'identificazione di pattern nascosti, la valutazione e la previsione delle prestazioni.

- **Deep Learning (DL)**: negli ultimi anni sono stati inventati e maturati sempre più approcci di DL in termini di classificazione e regressione. Il maggior numero di strati e neuroni in una rete DL consente infatti l'astrazione di problemi complessi, una maggiore accuratezza nella diagnosi e prognosi dei guasti e una migliore capacità di generalizzazione lavorando con dataset sempre più grandi.

- **Deep Reinforcement Learning (DRL)** per il processo decisionale: la svolta del DRL e delle sue varianti fornisce una tecnica promettente per un controllo efficace in sistemi complicati. DRL è in grado di gestire ambienti variabili nel tempo altamente dinamici con uno spazio di stato sofisticato, che può essere sfruttato per fornire supporto decisionale per un sistema PdM.

- **Hardware**: con il rapido sviluppo della tecnologia dei semiconduttori, i potenti hardware, come l'unità di elaborazione grafica (GPU) e le unità di elaborazione del tensore (TPU), possono accelerare significativamente il processo di evoluzione e ridurre il tempo richiesto dagli algoritmi di deep learning.

1.1 Manutenzione nei processi produttivi

La manutenzione è definita, secondo la norma UNI-EN 13306 [5], come "*la combinazione di tutte le azioni tecniche, amministrative e gestionali, durante il ciclo di vita di un'entità, destinate a mantenere o a riportarla in uno stato in cui possa eseguire la funzione richiesta.*" Nel corso degli ultimi decenni sulla base di fattori come i costi, il tipo di produzione e gli impatti che i fermi macchina possono portare all'azienda si sono sviluppate differenti strategie di manutenzione che ad oggi possono essere racchiuse in tre macro categorie di seguito descritte.

1.1.1 Manutenzione reattiva

Conosciuta anche come manutenzione a guasto (*Run-to-Failure*), è certamente la più basilare tra tutti i tipi di manutenzione, in quanto non mette in atto nessuna tipologia di prevenzione ma è sempre e solo postuma all'insorgere di una problematica, prevedendo semplicemente di attendere passivamente il presentarsi della rottura di un componente o di un asset. In questi casi dunque, l'intervento viene effettuato a valle di una situazione già compromessa, con l'unico scopo di ripristinare la funzionalità del macchinario. Per questa sua caratteristica, la manutenzione reattiva risulta spesso essere la tipologia di manutenzione più costosa, dal momento che l'intervento su un macchinario compromesso può comportare tempi piuttosto lunghi per il ripristino delle funzionalità e conseguenti ingenti perdite di produzione. Tuttavia, offre il massimo utilizzo e, a sua volta, la massima resa produttiva del componente e laddove viene applicata su asset a basso costo, o tale da non compromettere il funzionamento di un intero impianto, può essere considerata una strategia vincente.

1.1.2 Manutenzione preventiva

L'approccio preventivo prevede la programmazione di interventi eseguiti a intervalli temporali regolari oppure al superamento di una certa soglia di utilizzo. Tali interventi sono volti al controllo e ad un'eventuale sostituzione di determinati componenti riducendo così, ma non eliminando del tutto, il rischio di guasti e fermi macchina improvvisi; a differenza di quella reattiva quindi, richiede un investimento in tempo e risorse per gestire e pianificare i vari interventi manutentivi.
Il processo generale di manutenzione preventiva può essere presentato in due fasi:

1. Sulla base dei dati collezionati, la prima fase prevede l'indagine statistica delle caratteristiche dei guasti del macchinario, ricorrendo a parametri come ad esempio il *meantime-to-failure (MTTF)* o la *bathtub curve* (Figura 1.1);

2. Il secondo passo consiste nello schedulare gli interventi ottimali di manutenzione massimizzando l'affidabilità/disponibilità del sistema e le prestazioni di sicurezza e al contempo minimizzando il costo di manutenzione.

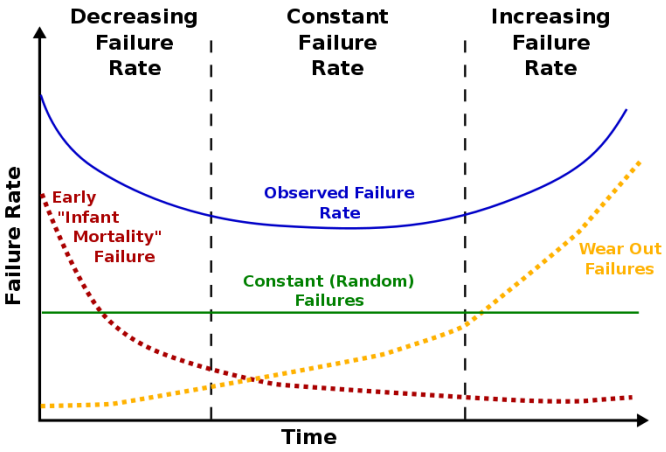


Figura 1.1: Andamento *bathtub curve*. Da `https://en.wikipedia.org/wiki/Bathtub_curve`

Sebbene la strategia preventiva sia ancora la norma nei processi industriali, conducendo ad una evidente riduzione dei costi di riparazione e i tempi di fermo macchina non pianificati, potrebbe comunque comportare riparazioni non necessarie o guasti catastrofici. Determinando infatti l'entrata di un componente nella fase di usura, basandosi sul tasso di guasto teorico invece che sulle condizioni specifiche dell'attrezzatura specifica, si può procedere alla sostituzione di un componente che non ha ancora raggiunto il suo intero ciclo di vita o può insorgere un guasto potenzialmente catastrofico poco prima dell'intervento programmato. Un esempio riguardo la limitata affidabilità dei parametri statistici è fornito da uno studio [6] svolto dagli ingegneri del gruppo SKF, un'azienda di analisi delle vibrazioni e della manutenzione predittiva. Nello studio è stato valutato e misurato il tempo di guasto di 30 cuscinetti identici, testati e sollecitati in condizione identiche. I risultati mostrati in Figura 1.2 mostrano come alcuni elementi abbiano ceduto in meno di 15 ore, e altri siano durati dieci volte più a lungo o più, arrivando ad un picco massimo di 300 ore. Nonostante quindi le identiche condizioni strutturali e di test, il tempo di guasto può variare anche di più di un ordine di grandezza.

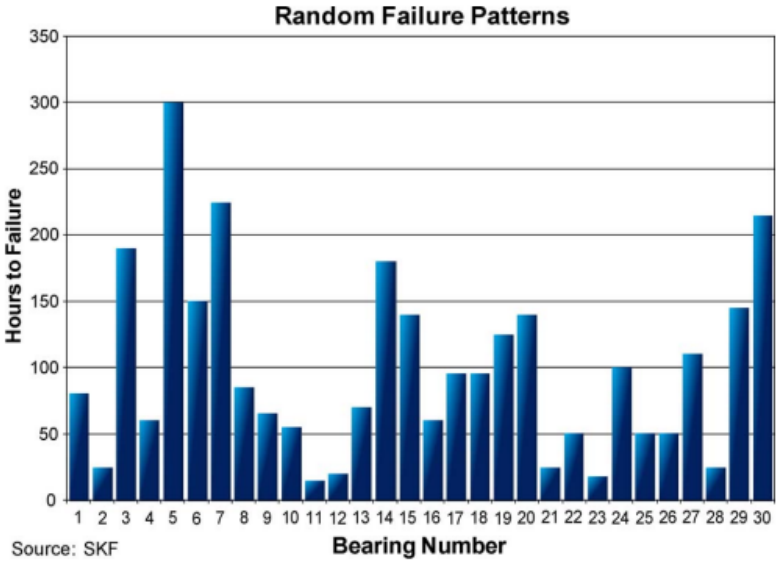


Figura 1.2: Risultati dello studio SKF. Da H. M. Hashemian and W. C. Bean, "State-of-the-Art Predictive Maintenance Tech-niques,"IEEE Transactions on Instrumentation and Measurement, vol. 60, no. 1,pp. 226–236, 2011

Un'ipotesi accreditata riguardo il limite dell'approccio preventivo è quella secondo cui tale manutenzione si fonda sul presupposto in base al quale la probabilità di guasto di un macchinario/componente aumenti con il tempo di utilizzo. A tal proposito, negli anni 1960-1978 Nowlan, Heap e Matteson definirono sei modelli di guasto, ovvero sei differenti andamenti del tasso di guasto[1] (λ) nel tempo [7]. I sei modelli, denominati A,B,C,D,E,F, possono essere divisi in due macrocategorie, *random* e *age-related*, a seconda che il tasso di guasto sia correlato o meno al tempo di utilizzo. I modelli A,B,C appartengono alla categoria *age-related* mostrando una crescente probabilità di guasto dopo una certo intervallo di tempo di utilizzo, tale crescita è spesso dovuta al progressivo invecchiamento dei componenti determinato dai processi di naturale degradazione chimico-fisica dei materiali. I pattern D,E,F al contrario, dopo una certa durata mostrano una probabilità costante. Questo non sta a significare che quel determinato componente non si degrada mai ma banalmente, il processo di degradazione è così lento che nel corso della sua normale vita operativa quel componente non presenta problemi di usura. Tuttavia, il tasso di guasto diverso da zero è dovuto alla possibilità non nulla di andare incontro a guasti casuali, dovuti ovvero a cause aleatorie.
Studi (Tabella 1.1) compilati dalla NASA e dalla Marina degli Stati Uniti e applicati ai dati relativi ai modelli di guasto mostrano che almeno il 70% dei componenti e apparecchiature presentano una probabilità di guasto costante nel tempo; per tali apparecchiature quindi non

[1]Prende il nome di tasso di guasto la funzione $\lambda(t)$ che definisce la probabilità che un componente, che al tempo t sia funzionante, si guasti in un intervallo di tempo compreso tra t e $t + dt$

risulta vantaggioso svolgere attività di manutenzione basate sul tempo o sull'utilizzo. Sulla base di ciò dunque, la manutenzione preventiva offre un vantaggio al massimo solo per il 29% delle risorse.

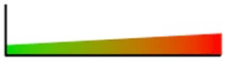
Tipologia	**Andamento temporale**	**Descrizione**	**Distribuzione**			
			1968	1973	1982	2001
Age-related	A 	**Wear-out**	4%	3%	3%	2%
	B 	**Bath curve**	2%	1%	17%	10%
	C 	**Fatigue**	5%	4%	3%	17%
Random	D 	**Initial Break-in**	7%	11%	6%	9%
	E 	**Random**	14%	15%	42%	56%
	F 	**Infant mortality**	68%	66%	29%	6%
Age-related			**11%**	**8%**	**23%**	**29%**
Random			**89%**	**92%**	**77%**	**71%**

Tabella 1.1: Modelli di guasto. Figure da `https://www.ercbv.eu/en/preventive-maintenance-covers-up-to-11-of-all-failures/`

1.1.3 Manutenzione predittiva

La premessa comune della manutenzione predittiva è che il monitoraggio regolare e continuo delle condizioni meccaniche effettive, dell'efficienza operativa e di altri indicatori delle condizioni operative dei macchinari e degli impianti di processo fornirà i dati necessari per garantire il massimo intervallo tra le riparazioni e ridurre al minimo il numero e il costo dei fermi macchina dovuti a guasti improvvisi. Più nello specifico, come riportato in [8] un programma completo di gestione della manutenzione predittiva utilizza gli strumenti più convenienti (ad es. monitoraggio delle vibrazioni, termografia, tribologia) per ottenere le condizioni operative effettive dei sistemi critici dell'impianto e, sulla base di questi dati effettivi, pianifica tutte le attività di manutenzione in base alle necessità (Figura 1.3).

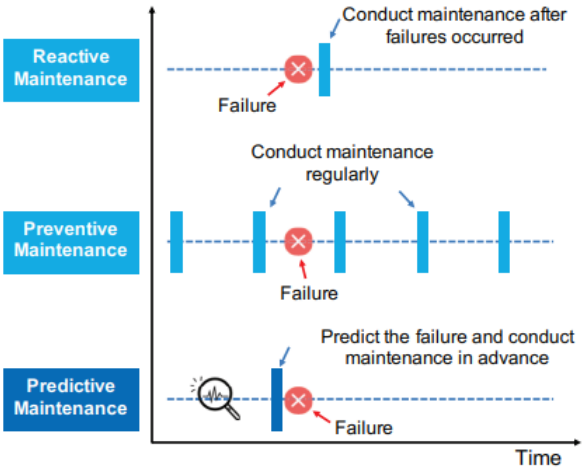


Figura 1.3: Confronto delle strategie di manutenzione RM, PM, PdM. Da Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng, "A Survey of Predictive Maintenance:Systems, Purposes and Approaches," 2019.

In [9] vengono elencate le 10 proprietà desiderabili per un sistema PdM: *rilevamento e diagnosi rapidi*, *isolabilità (distinzione tra diversi tipi di guasto)*, *robustezza*, *identificabilità di novità*, *affidabilità*, *adattabilità*, *facilità di spiegazione*, *requisiti minimi di modellazione*, *trade-off tra requisito di memoria e costo computazionale*, *identificabilità di guasti multipli.*

Una delle sfide principali dei casi d'uso industriali è la variabilità dei dati che si verifica anche in asset/macchinari che lavorano sotto le stesse condizioni operative, tolleranze meccaniche, regolazioni di montaggio e variazioni delle EOC *(environmental and operational conditions)*[2]. Questi fattori rendono difatti difficile la riutilizzabilità del modello PdM tra macchine e risorse. Altre sfide rilevanti sono la raccolta di dati di qualità, l'applicazione di un *pre-processing* adeguato e la *feature engineering* volta all'ottenimento un significativo set di features rappresentativo del processo in esame. Inoltre, essendo ogni osservazione correlata alle precedenti, occorre tener traccia dello storico del macchinario svolgendo di seguito un'analisi condizionata, aumentando però in questo modo la dimensionalità dei dati e la complessità della modellazione. Infine, la raccolta dei *run-to-failure data* non è sempre agevole e temporalmente frequente, notando come ad oggi le macchine industriali siano progettate e controllate per un corretto funzionamento e la seguente minimizzazione dei guasti.
Da un punto di vista operativo, alcuni tra i componenti chiave comunemente monitorati in PdM sono, ma non solo, cuscinetti, lame, motori, valvole, ingranaggi e utensili da taglio.

[2]Le condizioni ambientali e operative (EOC) sono le condizioni in cui un asset industriale come una macchina o un componente funziona. Le prima si riferiscono a condizioni esterne che le influenzano come la temperatura ambiente o le perturbazioni delle vibrazioni circostanti. Al contrario, le condizioni operative sono specifiche tecniche assegnate ai processi di lavoro, come la velocità, la forza o le posizioni desiderate.

Inoltre, tra i tipi di guasto più comunemente rilevati si riscontrano: fatica, usura abrasiva e da corrosione, sfregamento, shock termici, sollecitazioni esterne, assemblaggi impropri e difetti progettuali. In generale, i differenti guasti rientrano all'interno di quattro categorie principali: guasto dei componenti, impatto ambientale, errori umani e gestione delle procedure. Per quanto riguarda invece le tecniche di *condition monitoring* più comunemente utilizzate si hanno: ultrasuoni meccanici, analisi delle vibrazioni, test delle particelle di usura, termografia e analisi della corrente del segnale del motore, ma ci sono tecniche aggiuntive come emissioni acustiche, pressione o monitoraggio della temperatura.

In Tabella 1.2 viene presentato un confronto delle tre differenti strategie di manutenzione, evidenziando in particolar modo i relativi benefici e sfide, e gli scenari di utilizzo ed osservando come l'approccio manutentivo debba essere accuratamente definito in base alla tipologia di asset, costi e risorse impiegate.

	Benefici	**Sfide**	**Applicazioni adatte**	**Applicazioni non adatte**
RM	• Massimo utilizzo della risorsa di produzione • Azzeramento dei costi di prevenzione	• *Downtime* non pianificato • Elevato costo di inventario dei pezzi di ricambio • Potenziali ulteriori danni per il sistema • Costi di riparazione più elevati	• Apparecchiature ridondanti o non critiche • Riparazione dell'attrezzatura a basso costo dopo il guasto	• Il guasto dell'apparecchiatura crea un rischio per la sicurezza • È necessaria la disponibilità dell'attrezzatura 24 ore su 24, 7 giorni su 7
PM	• Basso costo di riparazione • Riduzione dei malfunzionamenti delle apparecchiature e *downtime* non programmati.	• Necessità di un inventario • Aumento del numero di *downtime* programmati • Interventi di manutenzione su apparecchiature apparentemente perfette	• Avere una probabilità di guasto che aumenta con il tempo/utilizzo dell'apparecchaitura	• Avere guasti casuali non correlati con la manutenzione
PdM	• Visione olistica della salute delle apparecchiature • Opzioni di analisi migliorate • Annullamento di sostituzione di componenti ancora funzionanti • Annullamento dei guasti	• Aumento dei costi e delle configurazioni iniziali dell'infrastruttura • Aumento della complessità del sistema	• Avere modalità di guasto che possono essere previste in modo conveniente tramite un monitoraggio regolare	• Avere modalità di guasto le cui previsioni non sarebbero convenienti in termini di costi e risorse impiegate

Tabella 1.2: Benefici, sfide e applicazioni di RM, PM, PdM

1.2 Introduzione alla Tesi

La ricerca di Tesi ha l'obiettivo di fornire in primo luogo una revisione sistematica della letteratura relativa ai moderni sistemi di manutenzione predittiva. Revisione necessaria ed antecedente all'esposizione del caso di studio in esame: analisi e disamina di un dataset pubblico denominato *Milling dataset* utilizzato per lo sviluppo ed implementazione di un framework di manutenzione predittiva.

Includendo il Capitolo introduttivo, la Tesi si compone di sei Capitoli. Il Capitolo 2 contiene lo studio e indagine dello stato dell'arte della manutenzione predittiva riservando una

maggiore attenzione a tre aspetti principali: *a)* le tecnologie di *condition monitoring* comunemente impiegate in ambito industriale, *b)* l'architettura IoT abilitante alla sua base e, *c)* gli strumenti e metodi, matematici e statistici, che a partire dai dati raccolti conducono alla stima della vita residua utile del macchinario d'interesse. Nel Capitolo 3 viene presentato il *Milling dataset* e viene fornita una revisione dei relativi e principali approcci di studio presenti in letteratura. Successivamente, nel Capitolo 4 viene illustrato e, corredato con una spiegazione teorica dei metodi componenti, l'approccio prognostico selezionato, mentre nel Capitolo 5 si va a descrivere la sperimentazione effettuata, provvedendo a riportare le considerazioni e problematiche intercorse nell'arco del suo sviluppo; infine vengono riportati i risultati ottenuti.

Il lavoro di Tesi si conclude nel Capitolo 6 dove vengono trattate le conclusioni riguardo il framework implementato e la discussione delle possibili direzioni future.

Capitolo 2

Stato dell'arte

2.1 Tecnologie per il monitoraggio delle condizioni

Come replicato più volte, la manutenzione predittiva pone le sue fondamenta nel continuo monitoraggio delle condizioni dei macchinari ed apparecchiature d'interesse. Al fine di raccogliere informazioni correlate con il processo in esame, il monitoraggio effettuato deve essere, in termini di tecnologie e sensori integrati, studiato e definito opportunamente.
A questo proposito, vengono di seguito presentate e descritte le principali tecnologie di monitoraggio ad oggi maggiormente impiegate in ambito industriale [10]:

- **Analisi delle vibrazioni**: è una tecnologia PdM matura che utilizza un sensore accelerometrico per misurare i livelli di vibrazione di una risorsa, coprendo così numerosi potenziali problemi relativi alle risorse e risultando maggiormente adatta alle apparecchiature rotanti. Questo perché, la gravità della vibrazione è soggetta a cambiamenti nel tempo e quando i livelli di vibrazione aumentano e superano i livelli di allarme preimpostati, questo spesso indica il deterioramento di asset o componenti, comportando gravi conseguenze qualora non si eseguisse un intervento mirato.
  I difetti comuni che possono essere rilevati con l'analisi delle vibrazioni includono: sbilanciamento, disallineamento, risonanza, usura dei cuscinetti, danni ai denti degli ingranaggi e cavitazione; tuttavia, l'elenco completo può essere esteso facilmente.

- **Termografia a infrarossi**: la termografia sfrutta l'elettronica avanzata della fotocamera per misurare le lunghezze d'onda nella regione degli infrarossi all'interno dello spettro elettromagnetico. É qui che si può infatti misurare il calore o l'energia termica che viene emessa da qualsiasi tipo di oggetto.
  Entrando maggiormente nel dettaglio, la termografia a infrarossi (IR) può essere defini-

ta come il processo di generazione di immagini visive che rappresentano variazioni nella radianza IR delle superfici degli oggetti. Analogamente al modo in cui oggetti, di materiali e colori diversi, assorbono e riflettono la radiazione elettromagnetica nello spettro della luce visibile (da 0,4 a 0,7 μm), qualsiasi oggetto a temperature maggiori dello zero assoluto emette energia IR (radiazione) proporzionale alla sua temperatura[1]. Utilizzando uno strumento che contiene rilevatori sensibili alla radiazione elettromagnetica IR, è possibile generare un'immagine visiva bidimensionale che riflette la radiazione IR dalla superficie di un oggetto e quindi consente di visualizzare l'oggetto in questione in base alla sua temperatura. Come tutte le tecnologie di manutenzione predittiva, l'IR cerca di rilevare la presenza di condizioni o fattori di stress che agiscono per ridurre la vita utile di un componente [11]. Per citare qualche esempio, un collegamento elettrico difettoso provoca temperature anormalmente elevate a causa dell'aumento della resistenza elettrica; prima che la connessione sia sufficientemente calda da provocare un guasto all'apparecchiatura o un possibile incendio, i modelli sono facilmente visibili attraverso una telecamera a infrarossi, e quindi la condizione viene identificata e corretta. I problemi delle apparecchiature rotanti invece, normalmente si traducono in una qualche forma di cambiamento di attrito che verrà identificato come un aumento della temperatura del componente. La perdita difettosa o completa di materiale refrattario sarà facilmente vista come un cambiamento nel profilo termico dei componenti.

In linea generale quindi, la termografia a infrarossi è in grado di rilevare una serie di problemi e modalità di guasto. Riassumendo, questi possono includere: cavi elettrici/guasti di isolamento, collegamenti elettrici e giunti allentati, fuga di gas, carichi sbilanciati, collettori di surriscaldamento, isolamenti danneggiati o mancanti, cuscinetti usurati o anche disallineamento dei giunti.

- **Analisi dell'olio lubrificante**: l'olio lubrificante viene utilizzato nelle macchine elettriche e meccaniche per ridurre l'attrito tra le superfici in movimento. L'olio di lubrificazione è un'importante fonte di informazioni per il rilevamento tempestivo dei guasti della macchina. Rispetto infatti al monitoraggio basato sull'analisi delle vibrazioni, il monitoraggio delle condizioni dell'olio di lubrificazione può fornire avvisi circa 10 volte più anticipati per malfunzionamenti e guasti della macchina [12]. Questa tipologia di analisi viene utilizzata principalmente per valutare le condizioni del fluido dell'olio lubrificante (viscosità del fluido, livello di additivo, proprietà di ossidazione e peso spe-

[1]Si ritiene generalmente che lo spettro di radiazione IR sia compreso tra 2,0 e 15 μm.

cifico), i contaminanti esterni (umidità, particelle metalliche, refrigerante e aria) e le particelle di usura dei componenti interni. Queste tre aree possono richiedere sensori separati o parzialmente integrati, evidenziando ad ogni modo differenti modalità di guasto, incluse le seguenti: cuscinetti usurati, danni ai denti degli ingranaggi, additivi lubrificanti esauriti, bassa viscosità dell'olio ed ingresso di contaminanti.

- **Analisi della firma della corrente del motore**: l'analisi della firma della corrente del motore (*Motor Current Signature Analysis*, MCSA) fornisce un metodo non intrusivo per rilevare problemi meccanici ed elettrici nelle apparecchiature rotanti motorizzate [13], [14]. La tecnologia si basa sul principio che un motore elettrico convenzionale che guida un carico meccanico funge da trasduttore. Il motore rileva infatti le variazioni di carico meccanico e le converte in variazioni di corrente elettrica che vengono trasmesse lungo i cavi di alimentazione del motore. I segnali di corrente così generati vengono registrati ed elaborati dal software per produrre una rappresentazione visiva delle frequenze esistenti rispetto all'ampiezza della corrente. L'analisi di queste variazioni può fornire un'indicazione delle condizioni della macchina, che può essere analizzata nel tempo per fornire un avviso tempestivo del deterioramento della macchina o dell'alterazione del processo. L'analisi della firma della corrente del motore è una delle tecniche predittive moderatamente complesse e costose; la complessità deriva in gran parte dalla natura relativamente soggettiva dell'interpretazione degli spettri e dal numero limitato di spettri storici o comparativi disponibili per applicazioni specifiche. In conclusione, questo tipo di analisi è in genere limitato alle applicazioni critiche con implicazioni per la vita/salute/sicurezza.

- **Emissioni acustiche e ultrasuoni**: il test delle emissioni acustiche è una tecnica di monitoraggio delle condizioni utilizzata per analizzare le onde sonore emesse e causate da difetti o discontinuità. Queste emissioni acustiche (generalmente indicate con la sigla AE) sono onde elastiche transitorie indotte da un rapido rilascio di energia di deformazione causato da piccole deformazioni, corrosione o fessurazione, che si verificano prima del cedimento della struttura. Nei macchinari, le fonti di emissioni acustiche includono urti, attrito, turbolenza, perdita di materiale e anche cavitazioni.

  In questo contesto, particolare importanza viene riservata agli ultrasuoni, onde sonore con frequenza superiore ai 20 kHz e normalmente non udibili dagli esseri umani. La maggior parte delle apparecchiature rotanti e molte condizioni del sistema fluido emetteranno modelli sonori nello spettro di frequenza ultrasonica ed i cambiamenti in queste

emissioni rifletteranno le condizioni dell'apparecchiatura. Un gas o un fluido compresso e forzato attraverso una piccola apertura crea turbolenza con forti componenti ultrasoniche sul lato a valle dell'apertura. Anche se tale perdita potrebbe non essere udibile dall'orecchio umano, la relativa rilevazione sarà comunque resa possibile con un dispositivo a scansione a ultrasuoni. In aggiunta, tale tecnologia può essere impiegata per il monitoraggio delle condizioni dei cuscinetti. Secondo una ricerca della NASA infatti, un aumento dell'ampiezza di una frequenza ultrasonica monitorata pari a 12-50 volte può fornire un'indicazione precoce del deterioramento dei cuscinetti.

In conclusione dunque, in generale il monitoraggio delle emissioni acustiche prevede una vasta gamma di applicazioni e il rilevamento di molteplici tipologie di guasto: rilevamento di perdite, valvole difettose, vibrazioni dei contatti elettrici, cuscinetti usurati, difetti dei denti degli ingranaggi e mancanza di lubrificazione.

2.2 IoT

L'Internet of Things (IoT) è uno dei pilastri principali della manutenzione predittiva, consentendo di tradurre le azioni fisiche delle macchine in segnali digitali fruibili. In tale contesto, e più in generale nel campo industriale, l'IoT è noto come Industrial IoT (IIoT) [15], [16], inteso come la definizione di una rete di nodi sensore intelligenti volta al monitoraggio delle prestazioni delle apparecchiature o addirittura di un intero processo produttivo.

Una tipica architettura IIoT si compone di tre livelli [17]: un livello di acquisizione, un livello di rete e un livello di applicazione. Il livello di acquisizione si trova alla base dell'architettura e prevede la disposizione di sensori, attuatori e dispositivi volti alla raccolta e trasmissione dei dati ai livelli superiori. Il livello di rete è al centro dell'architettura e può comprendere diverse tipologie di reti (es. reti locali (LAN), reti cellulari, Internet) e dispositivi (es. hub, router, gateway) abilitati da diverse tecnologie di comunicazione come Bluetooth, Wi-Fi e LTE. Infine, il livello applicativo è il livello IoT superiore ed è costituito da piattaforme locali o di cloud computing, offrendo servizi personalizzati agli utenti, come ad esempio archiviazione e analisi dei dati.

2.2.1 Livello di acquisizione

L'acquisizione dei dati è un processo di acquisizione e memorizzazione di diversi tipi di dati di monitoraggio da vari sensori installati sull'apparecchiatura monitorata e che sono in grado di riflettere il processo di degrado. É il primo processo di prognostica del macchinario. I dati

acquisiti vengono trasmessi a un PC o dispositivi portatili tramite un dispositivo di trasmissione dati e archiviati in una posizione di memoria per ulteriori analisi.
Da un punto di vista della tipologia dei dati raccolti, una delle principali credenze nell'industria 4.0 riguarda l'assunzione che maggiori quantità di dati collezionati e, quindi reti IoT più diffuse e performanti, si traducano sempre in prestazioni migliori. Tale assunzione si rivela in realtà fallace, sottolineando come l'obiettivo finale della digitalizzazione dovrebbe essere quello di acquisire dati intelligenti (*Smart Data*), piuttosto che *Big Data* [18]. Tra i vari fattori che determinano la qualità dei dati collezionati si trovano ad esempio: posizione dei sensori, frequenza di campionamento e tipologia dei dati memorizzati.
Il problema della posizione ottimale del sensore è stimare il numero richiesto di sensori per le migliori prestazioni e definire per essi le migliori posizioni a seconda del processo in esame [19]; mentre per quanto riguarda la tipologia dei dati memorizzati si intendono l'insieme delle tecniche di pre-processing e *features selection* adottate ed applicate ai *raw data* e definite sulla base della natura delle grandezze monitorate. Con lo scopo di spiegare come tali tecniche possano incidere sulla qualità dei dati memorizzati, viene qui riportato un esempio citato in [18]. Nell'esempio si considera un sistema di acquisizione dati da cuscinetti installati in uno stabilimento produttivo. I dati di vibrazione dei cuscinetti vengono acquisiti alla frequenza di 1,6 kHz. A causa però delle limitazioni di archiviazione dei dati, non si procede alla loro registrazione nei server ma piuttosto, si vanno ad estrarre e memorizzare solamente due *features*: l'accelerazione quadratica media (RMS) e la vibrazione picco-picco, entrambe mediate su un periodo di 0,5s. É qui che entra in gioco la necessità di un'acquisizione intelligente, poiché si dimostra come queste due *features* si rivelino efficaci nell'individuare guasti improvvisi, ma allo stesso tempo il loro contenuto informativo non è utile per sviluppare un approccio di manutenzione predittiva. La spiegazione risiede nel fatto che i segnali mediati su finestre temporali relativamente lunghe, codificano diverse condizioni di lavoro con carichi e velocità variabili, alle quali i segnali di vibrazione sono sensibili. L'analisi di questi segnali per la previsione può così portare a malintesi nella loro interpretazione.
Per quanto riguarda invece la frequenza di campionamento, essa gioca un ruolo fondamentale, in particolar modo dovrebbe essere abbastanza elevata al fine di estrarre informazioni rilevanti sulla base della grandezza in esame, ma allo stesso tempo occorre tener conto della capacità del link di trasmissione in modo da evitare eventi di congestione della rete.
Per concludere, è bene sottolineare come il rapido sviluppo dei sensori e delle tecnologie di comunicazione nelle industrie moderne, abbia condotto alla progettazione e applicazione di dispositivi di acquisizione dati sempre più avanzati. Tuttavia, è ancora difficile collezionare

dati *run-to-failure* di macchinari di alta qualità per i seguenti motivi [20]:

- I macchinari generalmente presentano un processo di degrado a lungo termine, dallo stato di salute al guasto, che può richiedere diversi mesi o addirittura molti anni. È dispendioso in termini di tempo e denaro acquisire tutti i dati *run-to-failure* durante un processo di degrado a lungo termine;
- In pratica, i macchinari non possono andare incontro a guasti poiché un guasto imprevisto può portare al guasto dell'intera macchina o addirittura a incidenti catastrofici;
- I macchinari, come i riduttori delle turbine eoliche, i riduttori per autoveicoli e i motori degli aerei, funzionano sempre in condizioni ambientali complesse. In questi casi, molte interferenze dall'ambiente esterno si combinano ai dati di monitoraggio, diminuendo così la qualità dei dati;
- Molti dati di monitoraggio vengono acquisiti durante il periodo di fuori servizio, come il tempo di inattività o il tempo di riavvio. Queste misurazioni generalmente presentano comportamenti distinti rispetto alle misurazioni acquisite durante il periodo di servizio, diminuendo così ulteriormente la qualità dei dati di monitoraggio.

2.2.2 Livello di rete

Da non confondere con il livello di rete della pila ISO-OSI, tale livello è anche noto come livello di trasmissione. Viene utilizzato per ricevere le informazioni elaborate e fornite dal livello di acquisizione e trasmettere tali informazioni all'hub IoT, ai dispositivi e alle applicazioni. Il livello di rete è il livello più importante nell'architettura IoT, poiché vari dispositivi (hub, switch, gateway, ecc.) e varie tecnologie di comunicazione sono integrati in questo livello. Tra i vari dispositivi citati, i gateway assumono una notevole rilevanza essendo i punti di connessione intermedi che connettono controller (PLC, dispositivi wireless ecc.) e sensori alla piattaforma di elaborazione, sia locale che su cloud, proteggendo inoltre anche la rete IoT e i dati trasportati.
Nella gamma di protocolli di comunicazione disponibili, uno dei requisiti principali nella scelta di quello più appropriato riguarda l'ambiente in cui deve essere eseguita la digitalizzazione, le velocità di trasmissione dei dati e la tipologia e il formato con cui essi vengono trasmessi. Infine, di fondamentale importanza è la conoscenza di base degli sviluppatori delle soluzioni e delle attuali tendenze IoT del dominio applicativo.
Di seguito vengono presentati i protocolli più comunemente impiegati suddivisi sulla base del relativo layer della pila ISO-OSI nel quale operano [21], [22], [23].

Livello data-link

I protocolli di livello data-link vengono integrati qualora l'architettura implementata non preveda un diretto interfacciamento dei nodi sensore alla rete Internet. In questo caso il collegamento avviene per mezzo di un gateway intermedio. I protocolli del livello data-link gestiscono così la comunicazione tra nodi sensore e, tra nodi sensore e gateway.
Di seguito vengono riportati i protocolli maggiormente utilizzati.

- Wi-Fi: offre comunicazioni a larghezza di banda elevata in grado di supportare protocolli di sicurezza più intensivi per i processi. Il Wi-Fi utilizza onde radio che trasmettono informazioni su frequenze specifiche, come i canali a 2,4 GHz o 5 GHz, e presenta una distanza media pari a 10-35 metri. I principali impatti sulla portata e sulla velocità di una connessione Wi-Fi sono l'ambiente circostante e la tipologia di copertura (interna o esterna). Detto questo, c'è il problema della coesistenza wireless per i protocolli più diffusi in cui le fabbriche rischiano interferenze esterne con dispositivi dipendenti dal Wi-Fi già in funzione. Tuttavia, il Wi-Fi viene spesso utilizzato per aumentare standard più specifici dell'applicazione per una connessione più fluida al cloud.

- Bluetooth 5 e Bluetooth Low Energy (BLE): offrono una portata migliorata rispetto agli standard precedenti ($<$200 m), grazie all'aumento della potenza di trasmissione massima da 10 dBm a 20 dBm. Tuttavia, la velocità dati relativamente elevata di 2 Mbps consente al dispositivo di inviare pacchetti rapidamente e di entrare più rapidamente in modalità di sospensione per migliorare la durata della batteria. Il Bluetooth ha lo stesso problema di coesistenza wireless del Wi-Fi essendo una tecnologia prolifica nella banda 2.4GHz senza licenza. Tuttavia, viene altamente implementato nell'IIoT, grazie soprattutto al suo valore per le applicazioni di tracciamento delle risorse. I beacon e i tag BLE offrono infatti il monitoraggio in tempo reale delle risorse della fabbrica.

- Basati su 802.15.4 (Zigbee, WirelessHART e ISA100.11a): tali standard soddisfano una particolare esigenza di un'applicazione a seconda dei requisiti di larghezza di banda, capacità, portata e latenza richiesti. Zigbee, WirelessHART e ISA100.11a si basano su reti personali wireless a bassa velocità IEEE 802.15.4 (LR-WPAN). Per questi standard, tutti i dispositivi nelle vicinanze devono trasmettere periodicamente dati a un gateway locale abilitato per LAN o WLAN, presentando un raggio operativo massimo di circa 200 metri e un throughput massimo di 250 kbps. Questi standard non sono specificamente progettati per un'elevata capacità con una vasta gamma e un gran numero di dispositivi collegati, in quanto hanno una potenza di trasmissione massima di

10 mW nella banda senza licenza di 2,4 GHz e sono quindi di portata limitata. Questo, combinato con la necessità di essere nelle immediate vicinanze di un gateway, rende difficile l'implementazione di decine di migliaia di nodi. Si noti che Zigbee opera anche nella banda radio industriale, scientifica e medica (ISM) a 915 MHz negli Stati Uniti, conferendogli migliori caratteristiche di propagazione. Nel complesso, questi protocolli sono ideali per applicazioni a medio raggio, velocità media e bassa latenza, rendendoli una scelta affidabile per molte applicazioni IIoT.

- LoRaWan: è un protocollo IoT per il controllo dell'accesso ai media (MAC). LoRaWAN consente ai dispositivi a bassa potenza di comunicare direttamente con le applicazioni connesse a Internet tramite una connessione wireless a lungo raggio. Inoltre, ha la capacità di essere mappato sia al 2° che al 3° livello del modello OSI. È implementato sulla modulazione LoRa o FSK per le bande radio industriali, scientifiche e mediche (ISM).

- LTE-A: Long-Term Evolution Advanced (LTE-A) è un insieme di standard progettati per adattarsi alle comunicazioni M2M e alle applicazioni IoT nelle reti cellulari. LTE-A è un protocollo scalabile e a basso costo rispetto ad altri protocolli cellulari. Utilizza OFDMA (Orthogonal Frequency Division Multiple Access) come tecnologia di accesso al livello MAC e prevede un'architettura costituita da una rete centrale (CN), una rete di accesso radio (RAN) e i nodi mobili.

Livello di rete

A differenza dei protocolli di livello data-link, le reti IoT che presentano un diretto interfacciamento dei nodi sensore alla rete Internet prevedono necessariamente la definizione di protocolli a livello di rete. In questo contesto, nonostante lo scetticismo iniziale di molti ricercatori sull'idoneità dell'architettura Internet per le reti di sensori, oggi la tendenza generale è quella di abbandonare le soluzioni proprietarie o standard chiusi per l'integrazione del protocollo IP. In effetti, i progressi nelle prestazioni dei recenti microcontrollori a 32 bit e la disponibilità di implementazioni di stack protocollari altamente ottimizzate, rendono possibile aggiungere connettività IP agli oggetti intelligenti. Dato il numero potenzialmente enorme di dispositivi connessi (Ericsson prevede più di 50 miliardi), IPv4 non può essere utilizzato a causa del suo spazio di indirizzi limitato. Una scelta migliore risulta quindi essere l'utilizzo di IPv6 il quale presenta indirizzi a 128 bit, autoconfigurazione della rete e il funzionamento senza stato. Grazie a IPv6, ogni oggetto intelligente può essere connesso agevolmente ad altre

reti basate su IP, senza la necessità di entità intermedie come gateway o proxy. Tuttavia, questo protocollo è progettato per le reti cablate; per soddisfare così le reti di sensori wireless (Wireless Sensor Network o WSN), è stato definito il protocollo 6LoWPAN. WSN è, infatti, composto da dispositivi caratterizzati da bassa potenza computazionale che spesso devono ridurre al minimo i consumi energetici. In questo modo, con l'integrazione del protocollo 6LoWPAN, i dispositivi possono affacciarsi direttamente sulla rete. L'uso ideale di 6LoWPAN è con applicazioni in cui i dispositivi embedded devono comunicare con servizi basati su Internet utilizzando standard aperti in grado di adattarsi a grandi infrastrutture di rete con mobilità. L'architettura 6LoWPAN è costituita da reti wireless a bassa potenza (LoWPAN), che sono connesse ad altre reti IP tramite router perimetrali e ciascun nodo LoWPAN è quindi identificato da un indirizzo IPv6 univoco ed è in grado di inviare e ricevere pacchetti IPv6.

Livello di sessione, presentazione e applicazione

Nell'ambito dei protocolli operanti ai livelli di sessione, presentazione e applicazione, tra i più comuni troviamo:

- *Message Queue Telemetry Transport* (MQTT): utilizzando la tecnica di pubblicazione/sottoscrizione, MQTT è un protocollo di messaggistica, utilizzato per raccogliere dati misurati su sensori/attuatori remoti e trasmettere i dati ai server. É un protocollo semplice e leggero e supporta la rete con larghezza di banda ridotta e latenza elevata.

- *Constrained Application Protocol* (CoAP): è un protocollo di messaggistica basato sull'architettura REST. Poiché la maggior parte dei dispositivi nell'IoT sono limitati in termini di risorse (ovvero, spazio di archiviazione ridotto e capacità computazionale ridotta), HTTP è difficilmente integrato nell'IoT, a causa della sua complessità. Per superare il problema, CoAP è stato proposto di modificare alcune funzioni HTTP per soddisfare i requisiti per IoT. In generale, CoAP è il protocollo a livello di applicazione nello stack di protocollo 6LoWPAN e mira a consentire ai dispositivi con risorse limitate di ottenere interazioni RESTful.

- *Data Distribution Service* (DDS): è un protocollo di pubblicazione/sottoscrizione per supportare la comunicazione da dispositivo a dispositivo ad alte prestazioni. DDS è un protocollo incentrato sui dati, in cui il multicasting può essere supportato per ottenere un elevato QoS e un'elevata affidabilità. L'architettura di pubblicazione/sottoscrizione

senza broker rende DDS adatto a comunicazioni IoT vincolate in tempo reale e da dispositivo a dispositivo.

- *Advanced Message Queuing Protocol* (AMQP): è un protocollo di accodamento messaggi utilizzato per fornire servizi di messaggistica (accodamento, routing, sicurezza, affidabilità, ecc.) nel livello applicativo. AMQP si concentra sugli ambienti orientati ai messaggi e può essere considerato come un protocollo middleware orientato ai messaggi. Utilizzando AMQP, i client possono ottenere una comunicazione stabile con i middleware dei messaggi, anche quando essi vengono prodotti da linguaggi di programmazione diversi. Inoltre, AMQP implementa vari tipi di architetture di scambio di messaggi, tra cui memorizzazione e inoltro, pubblicazione e sottoscrizione, distribuzione dei messaggi, accodamento dei messaggi, routing basato sul contesto e routing *point-to-point*.

2.2.3 Livello di applicazione

I dispositivi di sensing generano enormi quantità di dati, in modo continuo o periodico, spesso in un intervallo temporale molto breve. Secondo l'indice cloud Cisco (2013-2018) infatti, una struttura automatizzata può generare un terabyte di dati ogni ora. Anche una linea di produzione abbastanza semplice può generare enormi quantità di dati che devono di conseguenza essere acquisiti, aggregati, normalizzati e analizzati. Tuttavia, occorre notare come nelle applicazioni industriali, le azioni critiche, come allarmi ed arresti di emergenza richiedono un'azione immediata in un sito e dall'altro il trasferimento di grandi volumi di dati dalla periferia della rete a un server cloud pubblico può risultare molto costoso. In questo ambito dunque, un trend particolarmente in crescita negli ultimi anni è quello dell'*edge e fog computing*, ovvero l'implementazione di software intelligente nei nodi sensore perimetrali con lo scopo di pre-elaborare ed analizzare i dati raccolti e filtrare quelli da inviare al sistema di archiviazione centrale, risparmiando così risorse di rete e computazionali e riducendo la latenza delle risposte locali, per garantire l'affidabilità e l'agilità delle applicazioni interconnesse.
In questo contesto si introduce dunque la cosiddetta architettura a tre livelli: *cloud*, *fog* ed *edge computing* [24], [25].

- *Cloud computing*: il cloud consente alle aziende di utilizzare le risorse di elaborazione virtualmente tramite Internet senza la necessità di definire una propria infrastruttura locale. Può fornire risorse flessibili, scalabili e affidabili tra cui calcolo, archiviazione

e rete per abilitare varie applicazioni IoT. In genere, i flussi di dati in tempo reale da sensori e dispositivi distribuiti vengono trasmessi al centro cloud remoto tramite Internet, dove vengono ulteriormente integrati, elaborati e archiviati. Una caratteristica importante del *cloud computing* è che fornisce risorse di elaborazione flessibili in modalità *pay-as-you-go*, utile per i servizi IoT. Un'altra caratteristica è che può sfruttare tutti i dati dai dispositivi registrati in un'applicazione IoT, utile per addestrare modelli di *machine learning* e *deep learning* con una migliore rappresentazione e capacità di generalizzazione.

- *Fog computing*: trasferisce la capacità di archiviazione e ed elaborazione ai margini della rete, in prossimità dei dispositivi. Le strutture o infrastrutture che forniscono servizi di *fog computing* sono chiamate nodi *fog* e sono ad esempio router, switch, gateway e punti di accesso wireless. Sebbene funzioni in modo simile al *cloud computing*, il *fog computing* offre un vantaggio chiave, ovvero una bassa latenza, considerando la maggior vicinanza ai nodi sensore periferici; inoltre, può fornire continuità di servizio senza la necessità di Internet, il che è importante per applicazioni IoT specifiche con una connessione Internet instabile. L'altro vantaggio è la protezione della sicurezza e della privacy dei dati rimanendo essi confinati all'interno del perimetro della LAN.

- *Edge computing*: il termine di *edge computing* diviene in alcune pubblicazioni intercambiabile con il *fog computing*, tuttavia le due tipologie possono essere distinte in base alla loro posizione all'interno della LAN: *fog computing* si trova dal lato della rete mentre *edge computing* si riferisce al lato delle cose. In questo senso, l'*edge computing* si riferisce alla distribuzione della capacità di calcolo su dispositivi perimetrali in prossimità di sensori e attuatori. Un grande vantaggio è così la riduzione della latenza e della larghezza di banda della rete grazie all'elaborazione dei dati collezionati, direttamente sul posto e prima della trasmissione, in informazioni maggiormente compatte e strutturate. Tuttavia, a causa della capacità di calcolo ridotta dei nodi sensore, in essi è possibile integrare solamente algoritmi di pre-processing e machine learning leggeri in termini di costo computazionale.

2.3 Approcci e metodi prognostici: dai *raw data* alla stima della RUL

Collezionato un set di dati, un programma di manutenzione predittiva si compone generalmente di due processi principali: costruzione dell'indice di salute (HI) e stima della RUL (prognostica).

2.3.1 Identificazione dello stato del sistema (*health index*, HI)

Health index

Nell'ambito della manutenzione predittiva lo stato di salute di un sistema o componente viene espresso per mezzo dell'indice numerico (HI *health index HI*), un indice che varia nell'intervallo $[0, 1]$. In particolare quando $HI = 1$ il componente/sistema è perfettamente funzionante, al contrario quando $HI = 0$ il sistema è irreversibilmente rotto e necessita di essere sostituto. A seconda del meccanismo di degradazione, l'andamento temporale (Figura 2.1) di tale indice può seguire differenti modelli [26]:

- **un modello di degradazione a singolo stadio**: rappresenta un processo di degrado irreversibile e quindi monotono e continuo, come ad esempio il degrado di un inserto da taglio di una macchina utensile (Figura 2.1 curva a);
- **un modello di degradazione a due stadi**: è caratterizzato da una fase di perfetto funzionamento, senza evidenti anomalie, e una fase danneggiata, in cui il processo di degradazione inizia e prosegue fino al guasto o all'arresto della macchina. In questo caso l'algoritmo prognostico deve essere avviato all'inizio di questa seconda fase (Figura 2.1 curva b);
- **un modello di degradazione a tre stadi**: si osserva un trend di aumento-diminuzione-aumento-degrado ed è ad esempio il caso, della superficie dell'anello interno dei cuscinetti volventi: inizialmente l'impatto dei corpi volventi sulla superficie interna produce molte vibrazioni. Man mano tuttavia, il difetto viene attenuato dagli urti continui per poi aumentare nuovamente di dimensioni (Figura 2.1 curva c);
- **un modello di degradazione con multipli stadi**: il processo di degrado passa attraverso più fasi. Questo è spesso il caso di un sistema complesso affetto da guasti multipli, interattivi e concorrenti;

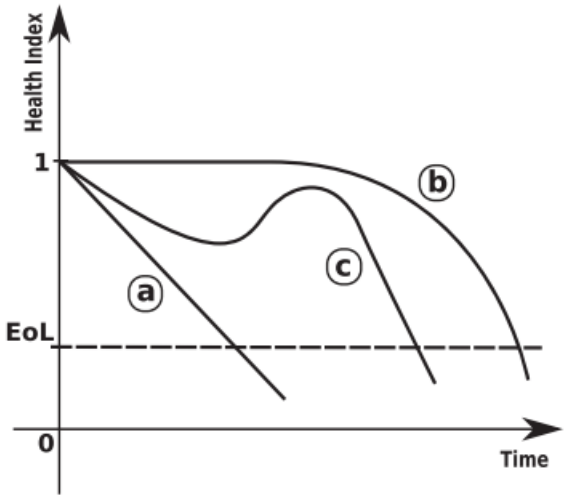


Figura 2.1: Andamento temporale *health index*. Da M. Baur, P. Albertelli, and M. Monno, "A review of prognostics and health managementof machine tools,"The International Journal of Advanced Manufacturing Technology,vol. 107, p. 2843–2863, 2020.

Bisogna inoltre considerare che, per ogni modalità di guasto che interessa il sistema, deve essere calcolato un indice di salute dedicato e nel caso di componenti multipli è possibile visionare lo stato di salute generale del sistema, ad esempio con un grafico radar (Figura 2.2). Tale grafico difatti riporta su ciascun asse l'indice HI relativo ad un singolo componente.

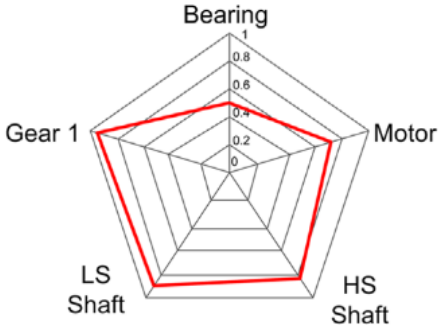


Figura 2.2: Grafico radar. Da M. Baur, P. Albertelli, and M. Monno, "A review of prognostics and health managementof machine tools,"The International Journal of Advanced Manufacturing Technology,vol. 107, p. 2843–2863, 2020.

La selezione dell'indice di salute è un momento chiave nell'impostazione di una politica di manutenzione PdM e le caratteristiche del pattern di degradazione influiscono pesantemente sulla scelta dell'algoritmo prognostico, rendendo le fasi di valutazione della salute e prognostiche strettamente connesse tra loro. In effetti, solo pochi approcci prognostici, come ad esempio Hidden Markov Models (HMM), possono trattare processi di degradazione multistadio, mentre una gamma più ampia di tecniche prognostiche è adatta per modelli di degradazione monotona a stadio singolo. A tal proposito, di seguito verranno considerati i

metodi ed approcci prognostici relativi al caso di un indice HI con un modello di degradazione a singolo stadio.

2.3.2 Prognostica

La prognostica corrisponde alla stima del tempo di guasto e del rischio per una o più modalità di guasto esistenti e future basate sull'utilizzo futuro previsto. A tal fine, la prognostica si occupa della stima dell'indice di salute del sistema e delle previsioni della vita utile residua (RUL). La *RUL*, *Remaining Useful Life*, è definita come il tempo rimanente entro il quale un componente, apparecchiatura o sistema sia in grado di eseguire le sue capacità funzionali in conformità con lo scopo previsto prima di andare incontro alla sostituzione. Formalmente e matematicamente, viene generalmente definita come "l'intervallo temporale che intercorre tra l'istante attuale e la fine della vita utile":

$$RUL_k = t_{EoL} - t_k \tag{2.1}$$

Dove t_{EoL} è l'istante di *End-of-life* del macchinario, t_k è l'istante corrente e RUL_k è la RUL corrente. Il compito principale della previsione RUL è prevedere il tempo rimasto prima che il macchinario perda la sua capacità operativa in base alle informazioni di monitoraggio delle condizioni; è quindi l'ultimo processo tecnico nonché l'obiettivo finale della prognosi dei macchinari.
Oggigiorno le tecniche e metodologie relative alla previsione della vita residua utile [27] possono essere classificate in quattro categorie principali (non tenendo in considerazioni eventuali combinazioni): *knowledge- or experience-based*, *model-based*, *statistical-based*, *data-driven*.

Knowledge- or experience-based

Negli approcci basati sulla conoscenza o sull'esperienza, l'intelligenza è inserita da esperti umani; tali approcci si basano infatti sulla creazione di database di eventi osservati precedenti e sulla deduzione dello stato di salute misurando la somiglianza tra i nuovi eventi osservati e il database. Le principali famiglie di algoritmi sono le seguenti:

- **Sistemi esperti**: risalenti agli anni '70, sono programmi software che simulano le prestazioni di esperti umani in un particolare campo. Generalmente consistono in un database di conoscenza che contiene l'esperienza accumulata da esperti in materia anche nel corso di un certo numero di anni e una base di regole, formulate come precise affermazioni *if-then*, per applicare tale conoscenza a particolari problemi noti al sistema

software (Figura 2.3). Per essere utile il database, così come le regole, devono essere più completi ed esatti possibile e aggiornati e mantenuti man mano che si acquisisce maggiore conoscenza o si modificano le configurazioni dell'impianto e del processo. Una volta sviluppati, tali sistemi vengono utilizzati per risolvere problemi normalmente risolti da specialisti umani, sia come parte di un sistema di controllo automatizzato e sia in combinazione con utenti non specializzati. Il loro vantaggio è che i risultati sono comprensibili e che è possibile stabilire un ragionamento per un particolare risultato. Sfortunatamente, i sistemi esperti sono validi solo quanto gli esperti che sviluppano la base di regole. Inoltre, non è fattibile per loro fornire una previsione variabile continua dato che gli output sono determinati da un insieme discreto di regole. Ciò limita la loro utilità nella stima diretta di RUL.

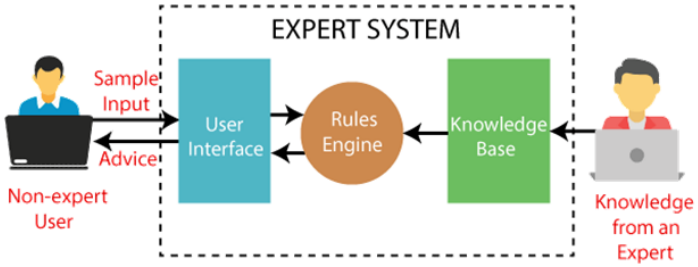


Figura 2.3: Architettura di un sistema esperto (ES). Da `https://www.javatpoint.com/expert-systems-in-artificial-intelligence`

- **Teoria fuzzy**: nella logica classica utilizzata dai sistemi esperti, un'affermazione può essere vera o falsa; i dati possono quindi essere inclusi o esclusi da un insieme. Tuttavia, non è sempre possibile definire in modo preciso insiemi e membri associati. Come nei sistemi esperti, i sistemi a logica fuzzy utilizzano semplici regole *if-then*, derivate empiricamente; tuttavia a differenza dei sistemi esperti, tali regole sono intenzionalmente imprecise, ovvero a ciascuna proposizione viene assegnato un grado di verità compreso tra 0 e 1. Una tipica istruzione di logica di processo fuzzy può ad esempio apparire come:

  if (il processo è troppo caldo) ***and*** (il processo si sta riscaldando rapidamente) ***then*** (raffreddare rapidamente il processo)

  Normalmente sono richieste solo poche regole, in quanto una regola fuzzy può sostituire un certo numero di regole convenzionali. I modelli logici fuzzy sono più efficaci in quanto possono gestire l'incertezza intrinseca alla conoscenza degli esperti umani e possono trattare variabili continue e dati contenenti alti livelli di rumore (causati ad esempio da

fluttuazioni di processo). Sono anche in grado di fornire risultati con dati incompleti o imprecisi, come si riscontra comunemente nella pratica. Inoltre, possono spiegare il loro ragionamento e, in virtù del minor numero di regole, sono più semplici da adattare rispetto ai sistemi esperti. Sfortunatamente, permane il problema della disponibilità e validità di una figura esperta nella specifica delle regole.

Model-based

Gli approcci prognostici basati su modelli tentano di impostare modelli matematici o fisici per descrivere i processi di degrado dei macchinari e aggiornare i parametri del modello utilizzando i dati misurati. I modelli comunemente usati includono il modello di processo di Markov, il modello di processo Winner, il modello *Gaussian Mixture*. I metodi basati su modelli potrebbero incorporare entrambi conoscenza esperta e informazioni in tempo reale dai macchinari. Di conseguenza, possono funzionare bene nella previsione RUL dei macchinari. Di seguito vengono riportati i vantaggi nell'utilizzo di approcci prognostici *model-based*:

- Risultano essere più efficace degli approcci privi di modello o basati sui dati quando viene costruito un modello corretto e accurato;
- Un modello basato sulla fisica può tenere conto di diverse condizioni operative, rendendo l'algoritmo prognostico più robusto e versatile;
- Possono essere necessari meno dati, rispetto a un metodo basato sui dati, per mettere a punto un algoritmo prognostico basato su modello;

Tuttavia, esistono ancora alcuni problemi nell'ambito di questi approcci:

- Un modello del sistema non è sempre disponibile, a causa della mancanza di approcci di modellazione per il sistema di interesse, o per il costo troppo elevato della modellazione dovuto alla complessità del sistema. La causa risiede nel fatto che i modelli prognostici sono a tutti gli effetti modelli di degrado, che si basano quindi sull'effettiva comprensione dei meccanismi della fisica del guasto;
- Le loro prestazioni dipendono fortemente dall'accuratezza del modello;
- Il problema della riconfigurabilità, ovvero l'impossibilità di riutilizzare il modello per un diverso tipo di macchina.

Statistical model-based

Gli approcci basati su modelli statistici, denominati anche approcci basati su modelli empirici, stimano la RUL dei macchinari stabilendo modelli statistici basati sulla conoscenza empirica e generalmente presentano il risultato della previsione RUL come una funzione di densità di probabilità condizionata alle osservazioni. Il processo di degradazione è visto quindi come un processo stocastico soggetto a diverse fonti di variabilità e incertezza e la previsione della RUL avviene adattando le osservazioni disponibili ai modelli con coefficienti casuali o modelli di processo stocastico secondo un metodo probabilistico, senza fare affidamento su fisica o principi. Generalmente, vengono introdotte varianze casuali nei parametri del modello per descrivere le incertezze causate da diversi tipi di fonti di variabilità; per questo motivo, tali approcci sono efficaci nel descrivere l'incertezza del processo di degradazione e la sua influenza sulla previsione RUL. Ordinariamente, i diversi metodi vengono classificati in due categorie:

- **Modelli stocastici basati su processi di stato direttamente osservati**: questi modelli sono basati su dati o features relativi a un *condition monitoring* diretto, cioè dati o features che possono descrivere direttamente lo stato di degrado o dualmente lo stato di salute del sistema. In questi casi, la prognosi si traduce nella previsione del momento in cui verrà raggiunta una soglia predefinita. I seguenti modelli stocastici rientrano in questa categoria:
  - I modelli basati sulla regressione modellano l'evoluzione dello stato come un processo continuo. I più usati sono i *modelli di serie temporali autoregressivi (AR)*, come i *modelli autoregressivi a media mobile (ARMA)*, che presuppongono che l'evoluzione futura della variabile prevista sia una funzione lineare sia delle osservazioni passate che normalmente rumore casuale distribuito, sotto le ipotesi di dati stazionari e di indipendenza statistica degli errori. I loro vantaggi sono che sono facili da implementare, non richiedono molte risorse computazionali per il calcolo e i risultati possono essere facilmente spiegati. D'altro canto però, le loro prestazioni sono fortemente influenzate dalla qualità delle osservazioni passate, ed inoltre di solito sono efficaci per la previsione a breve termine e irrealizzabili per la previsione a lungo termine. Infine, non funzionano bene durante la fase iniziale del guasto, essendo in questa fase il parametro *HI (Health Index)* tipicamente non monotono e rumoroso.
  - I modelli di Markov rappresentano l'evoluzione dello stato come un processo nello spazio degli stati discreto e finito e assumono che il processo di degrado del sistema

sia rappresentato da una serie di trasformazioni nello spazio degli stati; trasformazioni che obbediscono alla proprietà di Markov[2] e quindi ad un'assunzione senza memoria. I modelli markoviani sono stati ampiamente applicati alla stima della RUL e al supporto decisionale per la manutenzione per il semplice motivo che la condizione di funzionamento dell'impianto può essere suddivisa in diversi stati significativi, come "*Buono*", "*OK*", "*Solo difetti minori*", "*Manutenzione richiesta*", "*Non riparabile*", in modo che la definizione di stato sia più vicina a quella utilizzata nell'industria rispetto ad altri modelli stocastici, e quindi di facile comprensione. Tuttavia, mostrano differenti limiti come ad esempio l'assunzione senza memoria che non è sempre valida nei processi di degrado e la probabilità di transizione tra gli stati del sistema è determinata o dalla conoscenza empirica o da grandi insiemi di dati, che non sono sempre disponibili.

- **Modelli stocastici basati su processi di stato direttamente osservati**: detti anche modelli di processi di stato parzialmente osservati, poiché esiste una relazione stocastica tra i dati di monitoraggio delle condizioni osservati e lo stato di degrado non osservabile. Questi modelli stocastici si basano su dati relativi a un *condition monitoring* indiretto, cioè dati che possono indicare solo indirettamente o parzialmente lo stato di salute del sistema, come ad esempio i dati di vibrazione. I seguenti modelli stocastici rientrano in questa categoria:

  - Approcci basati sul filtraggio stocastico modellano il processo di degradazione come uno stato che deve essere stimato sulla base delle osservazioni delle variabili di output. Esempi di filtri sono il filtro di Kalman e il particle filter [28] [29].
  - Il modello di Markov nascosto (HMM) è un modello di processo stocastico e parametrico, composto da due processi stocastici, una catena di Markov nascosta, che non è osservabile e rappresenta lo stato reale del deterioramento, e un processo osservabile, che tiene conto dell'osservazione ottenuta dal monitoraggio e dai test. Di conseguenza, viene utilizzata una misura di probabilità condizionata per rappresentare la relazione tra le osservazioni e lo stato nascosto. Gli HMM sono stati adottati per questo tipo di prognosi dall'inizio del ventunesimo secolo tuttavia presentano alcuni svantaggi come la difficoltà incontrata nella stima dei parametri

[2]Proprietà secondo cui, in un processo aleatorio, la probabilità di transizione, che determina il passaggio a uno stato di sistema, dipende solo dallo stato del sistema immediatamente precedente e non da come si è giunti a questo stato.

HMM (grandi risorse di calcolo e memoria richieste) e nella stima della probabilità di transizione di stato. In aggiunta a ciò, la proprietà di Markov limita la capacità dell'HMM di modellare la struttura temporale dei problemi di previsione e con questo approccio è possibile stimare solo la media e la varianza RUL.

Data-driven

Negli ultimi anni, le reti di sensori wireless industriali (IWSN) e i sistemi cyber fisici industriali sono diventati una tecnologia emergente di acquisizione dati nell'ambito industriale, permettendo di raccogliere un'importante mole di dati in tempo reale, con un'elevata affidabilità e utilizzando varie tipologie di sensori. Pertanto, a causa del continuo miglioramento nella capacità di acquisizione dei dati, della crescita esponenziale del volume dei dati, e degli importanti progressi che gli algoritmi di intelligenza artificiale (AI) hanno compiuto negli ultimi cinque anni, i metodi *data-driven* hanno riscosso un notevole successo [30]. A differenza degli approcci di manutenzione basati su modelli, che si basano sulla previsione del degrado delle prestazioni mediante l'uso di modelli stocastici, le pratiche PdM basate sui dati non prevedono una conoscenza preliminare delle condizioni di degrado. Le prestazioni dipendono dunque dalla qualità dei dati, dalla loro relativi analisi e dall'algoritmo prognostico utilizzato. Mentre per i sistemi complessi, le soluzioni basate su modelli possono essere costose e imprecise, i metodi *data-driven* sono un'alternativa promettente al rilevamento e all'isolamento di guasti/anomalie.

Oggigiorno un sistema PdM *data-driven* prevede e si basa sull'integrazione di algoritmi di *machine learning* (ML) e *deep learning* (DL). Volendo definire la sostanziale differenza tra le due metodologie, come mostrato in Figura 2.4, gli algoritmi ML tradizionali (*regressione logistica* (LR), *support vector machine* (SVM), *decision tree* (DT) e *random forest* (RF)) richiedono generalmente la raccolta di grandi quantità di dati storici raccolti a partire dalle condizioni di perfetto funzionamento fino ad arrivare allo scenario di guasto. Successivamente, viene condotta la fase di *feature engineering* nel dominio del tempo, della frequenza o simultaneamente in entrambi e le features così estratte vengono utilizzate per l'apprendimento dello stato di degrado del sistema. D'altro canto, i modelli di *deep learning* evitano la complessa fase di *feature engineering* e si basano un metodo di apprendimento end-to-end, che viene implementato aggiungendo strati profondi tra i *raw data* e il risultato della previsione. Pertanto, i modelli profondi possono essere considerati come una vera e propria *black-box*, che a partire dall'input emette direttamente il risultato della previsione.

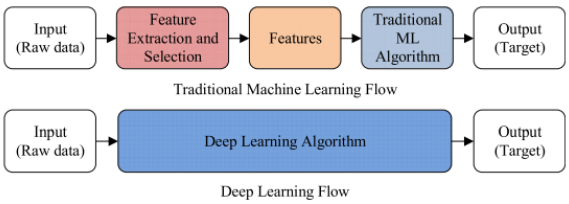


Figura 2.4: Diagramma di flusso di un sistema ML e DL. Da W. Zhang, D. Yang, and H. Wang, "Data-Driven Methods for Predictive Maintenance ofIndustrial Equipment: A Survey,"IEEE Systems Journal, vol. 13, no. 3, pp. 2213–2227,2019.

Per quanto riguarda la modalità di apprendimento in entrambi i casi (ML e DL) e sulla base della disponibilità dei dati e delle rispettive etichette, questa può essere classificata in tre differenti categorie: *a)* apprendimento supervisionato, in cui un set di dati di addestramento etichettato viene utilizzato per una mappatura dall'insieme di valori delle variabili predittive a una variabile target specificata; *b)* apprendimento semi-supervisionato, in cui l'obiettivo è apprendere da insiemi di dati che hanno il valore della variabile target solo per un sottoinsieme di esempi; e *c)* apprendimento non supervisionato, in cui la macchina apprende da un set di dati senza variabili target.
Nell'ambito specifico delle pratiche PdM, è difficile assegnare etichette al flusso di dati in tempo reale dai sensori in un impianto industriale. In primo luogo per le limitate tipologie di misurazioni e, in secondo luogo, per il costo e la fattibilità di far analizzare i dati da uno o più specialisti. Pertanto, è possibile sostenere come l'utilizzo dell'apprendimento supervisionato non sia una soluzione altamente fattibile in tale contesto.
Di seguito, per ciascuna delle due metodologie, vengono riportati i principali algoritmi.

- **Machine learning**

  Machine learning è un termine generico utilizzato per rappresentare differenti metodi computazionali che spaziano dal trovare pattern nascosti nei dati a effettuare previsioni. In ambito PdM gli algoritmi più comuni risultano essere:

  - ***Logistic Regression (LR)***: è un noto modello di classificazione in ML con la minore complessità dell'algoritmo. Appartiene all'apprendimento supervisionato; pertanto, i dati raccolti devono avere etichette corrispondenti per essere inseriti nel modello. Inoltre, il modello LR prende una combinazione lineare di *features* come input e applica una funzione non lineare per condurre la mappatura, in modo che ogni output rientri nell'intervallo $[0, 1]$ e si possa ottenere un'interpretazione probabilistica. Di conseguenza, LR risponde al caso in cui si disponga di una notevole quantità di dati etichettati e requisiti critici sulla complessità del modello.

– ***Support vector machine (SVM)***: il modello SVM viene generalmente utilizzato nell'ambito della classificazione binaria. Nella manutenzione predittiva delle apparecchiature industriali, gli SVM sono stati ampiamente applicati per identificare uno stato specifico in base al segnale acquisito. Tuttavia può essere facilmente esteso a problemi di classificazione multiclasse.

– ***Decision Tree (DT)* e *Random Forest* (RF)**: un albero di decisione (DT) è un sistema con n variabili in input e m variabili in output. Le prime, denominate anche attributi, sono derivate dall'osservazione dell'ambiente, le seconde invece, identificano la decisione/azione da intraprendere. Negli alberi decisionali molto profondi le variabili in output intermedie, in uscita dai nodi genitori, coincidono con le variabili in input dei nodi figli e condizionano il percorso verso la decisione finale. Il processo decisionale è rappresentato con un albero logico rovesciato (Figura 2.5) costituito da un nodo radice, nodi intermedi e foglie. Ogni nodo a partire da quello radice e scendendo progressivamente verso il basso, verifica una condizione di test su un determinato attributo presentando due o più diramazioni verso il basso. A seconda del valore dell'attributo, in ciascun nodo il flusso prende una direzione e dunque il processo decisionale consiste in una sequenza di test. Infine, dopo aver analizzato le varie condizioni, l'agente giunge ad un nodo foglia e quindi alla decisione finale. Sulla base di tale struttura quindi, uno dei vantaggi del loro utilizzo è l'inevitabile semplicità di esecuzione e di comprensione da parte degli utenti, potendo verificare direttamente come la macchina giunge alla decisione finale. Tuttavia, la rappresentazione ad albero è poco adatta per i problemi complessi, poiché aumentando il numero di attributi la complessità spaziale dell'algoritmo potrebbe diventare proibitiva. Inoltre, può descrivere soltanto una relazione tra una funzione di verità e le combinazioni logiche di attributi, potendo replicare solo un limitato numero di funzioni. In aggiunta a ciò, RF è un algoritmo di apprendimento di insieme composto da più classificatori DT e la categoria del suo output è determinata congiuntamente da ciascun singolo albero. RF è dotato di molti vantaggi significativi come ad esempio la possibilità di gestire dati ad alta dimensione senza una selezione delle caratteristiche; l'indipendenza tra gli alberi durante il processo di formazione e la semplicità di implementazione. Inoltre, la velocità di addestramento è generalmente elevata e allo stesso tempo la capacità di generalizzazione è notevole.

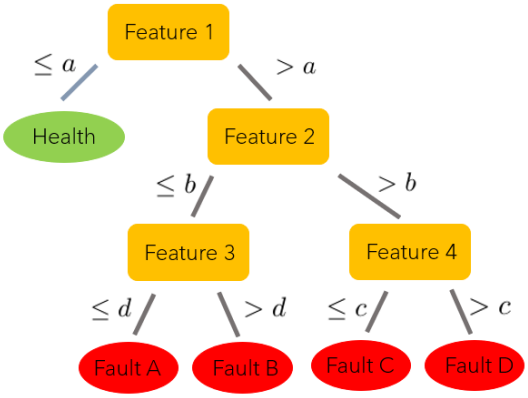


Figura 2.5: *Decision Tree* (DT). Da `https://towardsdatascience.com/data-driven-predictive-maintenance-in-a-nutshell-ccc65a13b998`

- **Deep learning**

  Gli approcci ML tradizionali mostrano prestazioni migliori per quantità inferiori di dati di input. Tuttavia, i progressi nelle tecnologie di rilevamento e l'emergere di tecnologie come l'IoT producono una grande quantità di dati e, di conseguenza, le prestazioni delle tecniche ML tradizionali non potrebbero soddisfare la scala richiesta. In questo contesto, il *deep learning* diventa una scelta obbligata grazie alla sua capacità di elaborare dati sequenziali altamente non lineari e variabili con un input umano minimo in diversi domini di conoscenza. I principali metodi e strumenti di DL utilizzati nel campo della manutenzione predittiva sono:

  – **CNN (*Convolutional Neural Network*)**

    La rete neurale convoluzionale (CNN) è uno dei modelli di deep learning più importanti grazie alla sua capacità di estrarre le caratteristiche locali dei dati di input e combinarle strato per strato per generare funzioni di alto livello. Una tipica struttura CNN si compone di quattro strati: strato di input, strato di convoluzione, strato di pooling e uno strato completamente connesso. Gli strati di convoluzione e pooling estraggono le caratteristiche dai dati di input, mentre i layer completamente connessi apprendono una rappresentazione dei dati di training per stimare o classificare le variabili target. Entrando nel merito della struttura, lo strato di input può essere presentato in forma bidimensionale come lo spettro tempo-frequenza o unidimensionale come i dati di serie temporali; nel livello di convoluzione, il kernel di convoluzione (filtro) agisce sui dati di input del livello precedente attraverso una serie di pesi e compone un output di funzionalità, generalmente chiamato mappa di funzionalità. Successivamente con l'inserimento

di uno strato di pooling si applica un campionamento, che viene utilizzato per ridurre i parametri del modello e conservare le informazioni efficaci. Allo stesso tempo, l'overfitting può essere evitato in una certa misura e la velocità di apprendimento può essere migliorata. Infine, dopo diverse forme di combinazione di layer di convoluzione e layer di pooling, seguiranno più layer completamente connessi, che possono convertire le mappe di funzionalità ottenute in una colonna o in una riga. Aggiungendo poi uno strato di classificazione o regressione si raggiungono gli obbiettivi prefissati. Generalmente, la CNN viene combinata con un determinato algoritmo di elaborazione del segnale per una migliore prestazione di diagnosi dei guasti. Infine, occorre sottolineare come sebbene, la CNN 1D richieda una quantità limitata di dati per un training efficace e disponga di un'implementazione hardware a basso costo per applicazioni in tempo reale, presenti una scarsa capacità di estrazione delle funzionalità per i dati dei sensori con formato 1D.

– **RNN (*Recurrent Neural Network*)**

  Le reti neurali ricorrenti sono un gruppo di reti neurali per la gestione di dati sequenziali. Come modello sequenziale, RNN può costruire connessioni cicliche tra le sue unità nascoste e mantenere una memoria degli input precedenti nello stato interno della rete. In questo modo, l'output finale è la rappresentazione appresa di tutti i dati sequenziali di input. Tuttavia, venendo addestrate con il metodo *Back Propagation Through Time* (BPTT), presentano il famigerato problema di scomparsa/esplosione del gradiente. Per superare questo problema, sono state introdotte due architeturre principali: *Long short term memory* (LSTM) e *Gated recurrent units* (GRU).

– **Autoencoder**

  Un autoencoder è una rete neurale unica, composta concettualmente da due parti, encoder e decoder, e con l'obiettivo di apprendere una rappresentazione che replichi fedelmente gli input come output. I livelli di input e output hanno lo stesso numero di nodi e tutti i livelli sono completamente connessi, ma la rete introduce un collo di bottiglia per forzare la rete a imparare solo le caratteristiche essenziali. Il collo di bottiglia viene introdotto scegliendo il numero di nodi, nello strato di connessione, tra la parte codificatore e la parte decodificatore, inferiore al numero nello strato di input. Il processo di addestramento è simile ad altre reti neurali, in cui vengono appresi i pesi e i bias della rete riducendo al minimo una determinata funzione di perdita. Nel processo di *training* l'encoder apprende una rappresentazione dei

dati di input e il decodificatore ricostruisce l'input dalla rappresentazione appresa. Questo processo di apprendimento e ricostruzione viene sfruttato per vari scopi tra cui il denoising e la riduzione della dimensionalità. Il denoising può aiutare a migliorare la qualità dei dati e quindi la qualità delle previsioni. La riduzione della dimensionalità può aiutare invece a migliorare l'efficienza computazionale quando i dati sono ad alta dimensionalità.

2.3.3 Conclusioni

Varie tecniche, modelli, algoritmi e i loro principi di modellazione sono stati sin qui discussi. Nello specifico, ci sono una serie di sfide e problemi pratici da studiare ulteriormente prima che modelli validi e robusti possano essere applicati a sistemi pratici:

1. É auspicabile sviluppare un modello di stima RUL basato su pochissime o nessuna situazione di dati. Questo è tipico per i sistemi di nuova messa in servizio in cui non esistono dati di guasto osservati e informazioni di monitoraggio. In questo caso, la quantità e la completezza dei dati risulterebbero insufficienti per adattarsi ai modelli statistici completi discussi in precedenza. Ciò può rendere necessario l'utilizzo di modelli basati sulla fisica con l'aiuto di conoscenze esperte soggettive provenienti dalla progettazione e dalla produzione. Tuttavia, come stabilire un modello RUL basato sulla fisica e come utilizzare al meglio le informazioni di giudizio soggettivo nella stima RUL rimangono una sfida.

2. La seconda sfida risiede nella fusione dei dati in cui devono essere trattati dati di input multidimensionali. Tipicamente nell'ambito PdM si collezionano dati multicanale, che possono essere correlati e non sulla stessa scala e unità di misura. Ciò imporrà anche una seria sfida ai modelli basati su soglie che sono stati per lo più stabiliti sotto un unico livello di soglia.

3. La terza sfida è come modellare l'influenza delle variabili ambientali esterne nella stima RUL. Questo è un problema piuttosto rilevante poiché tali variabili avranno un impatto sulle grandezze osservate che a loro volta influenzeranno la stima RUL. Se non viene eseguito correttamente, può verificarsi un adattamento eccessivo, che può ridurre la robustezza del modello di stima sviluppato.

4. La quarta sfida è lo sviluppo di un modello in grado di gestire più modalità di guasto per un singolo componente, che è ancora uno scenario comune osservato nella pratica.

Ovviamente questo complicherà ulteriormente il problema poiché per alcune tecniche, i segnali osservati sono pesantemente influenzati dalle modalità di guasto sottostanti, e quindi la RUL è dipendente dalla modalità di guasto.

Capitolo 3

Caso di studio

3.1 Dataset

Il caso di studio presentato e analizzato nel presente lavoro di Tesi si basa sul dataset pubblico *Milling* [31], un dataset collezionato da Agogino e Goebel e presentato nel *Prognostic Center of Excellence (NASA – PCoE)*. Il dataset raccoglie i dati derivanti da sperimentazioni svolte su una macchina fresatrice, in particolar modo raccoglie i valori registrati da sei sensori durante tutto il ciclo vita di 16 Casi, sperimentando un totale di 8 condizioni operative e 167 lavorazioni (definite *corse* nel seguito). Ogni condizione operativa, testata in due Casi, è caratterizzata da due parametri di lavorazione, che seguono le raccomandazioni del produttore dell'utensile, e dal tipo di materiale lavorato di dimensioni fisse (483 mm x 178 mm x 51 mm). Fissata la velocità di taglio pari a $200m/min$, i tre parametri di lavorazione, rappresentati da variabili dicotomiche, sono: la velocità di avanzamento (feed), con valori $0,25mm/s$ o a $0,5mm/s$, la profondità di taglio (DOC), con valori a $0,75mm$ o a $1,5mm$; e i materiali del pezzo di lavorazione che sono ghisa (cast iron) o acciaio (steel). In Tabella 3.1 vengono riportate le condizioni operative e le corse effettuate per ciascun Caso.

Material	**Depth of cut** (mm)	**Feed** (mm/rev)	**Case - Runs**	
			First	**Second**
Cast iron	1.5	0.5	$\#1-17$	$\#9-9$
Cast iron	0.75	0.5	$\#2-14$	$\#12-15$
Cast iron	0.75	0.25	$\#3-14$	$\#11-23$
Cast iron	1.5	0.25	$\#4-7$	$\#10-10$
Steel	1.5	0.5	$\#5-6$	$\#16-6$
Steel	1.5	0.25	$\#6-1$	$\#15-7$
Steel	0.75	0.25	$\#7-8$	$\#13-15$
Steel	0.75	0.5	$\#8-6$	$\#14-9$

Tabella 3.1: Condizioni operative di ciascun caso

Come riportato brevemente poc'anzi, i dati collezionati dal dataset sono quelli registrati da sei differenti sensori: sensori di emissioni acustiche e vibrazioni entrambi su posizionati sul mandrino e sul tavolo e sensori di corrente per la componente AC e DC della corrente del motore del mandrino. Per effettuare le lavorazioni è stata impiegata la Matsuura MC-510V, una macchina a controllo numerico verticale con una fresa da 70 mm a sei inserti. Sono stati utilizzati inserti KC710, caratterizzati da diversi rivestimenti in sequenza di carburo di titanio, carbonitruro di titanio e nitruro di titanio, che conferiscono una buona resistenza al cratering e all'usura degli spigoli cosìcche il fenomeno di usura principale è il *flank wear*. Quest'ultimo è stati infatti scelto come parametro principale per la valutazione dell'usura dell'utensile ed indicato nel dataset con la notazione VB.

3.1.1 Flank wear

Nell'ambito delle fresatrici industriali, l'usura degli utensili deve essere costantemente monitorata. Questa necessità è dovuta all'alta qualità dei prodotti, i quali prevedono finiture superficiali di alta qualità e precisione dimensionale. A questo proposito dunque, un utensile usurato e non mantenuto correttamente potrebbe provocare una deformazione della superficie lavorata e un maggiore attrito, comportando a sua volta temperature di taglio più elevate, dalle quali possono in seguito derivare effetti indesiderati.
L'usura degli utensili si presenta in diverse forme. Oltre all'intuitivo arrotondamento del tagliente, una tipologia piuttosto rilevante è l'usura sul fianco (*flank wear*) dovuta all'attrito dell'utensile sul pezzo lavorato. Tale tipologia si verifica generalmente quando la velocità di taglio è molto elevata, provocando molte perdite, come ad esempio l'aumento della rugosità della superficie del prodotto finale; tuttavia, è influenzata anche da altri parametri tra cui la profondità di taglio e la velocità di avanzamento. Nel *Milling dataset* l'usura sul fianco, indicata con VB, è stata utilizzata come parametro per valutare l'usura dell'utensile. Inoltre nella pratica, è stata misurata come la distanza dal tagliente all'estremità dell'usura abrasiva sulla faccia laterale dell'utensile (Figura 3.1) e nelle relative fasi di misurazioni è stata misurata con l'ausilio di un microscopio estraendo l'inserto dall'utensile.

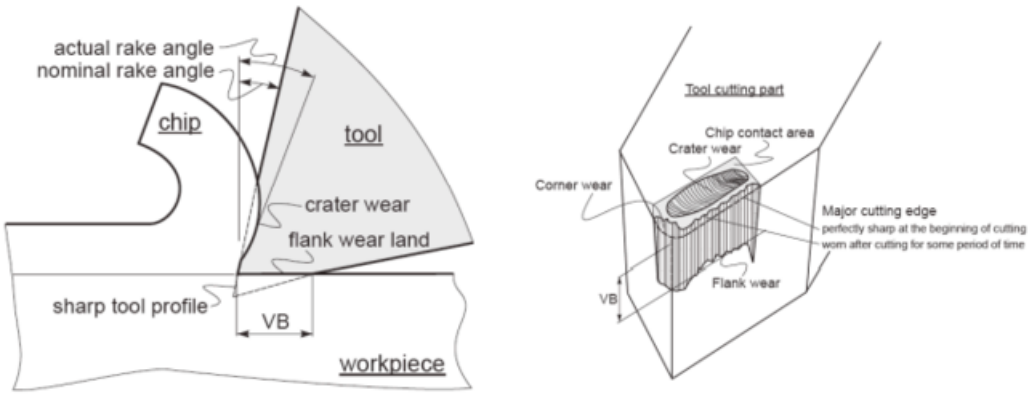


Figura 3.1: Processo di taglio e *flank wear*. Da `https://quizlet.com/gb/496469667/operation-of-production-machines-l3-flash-cards/`

3.1.2 Struttura Dataset

Il dataset viene fornito come una struttura MATLAB 13×167 i cui campi vengono riportati in Tabella 3.2. Al contempo in Figura 3.2 vengono riportati alcuni records di esempio della struttura MATLAB.

Field	**Descrizione**
case	Condizione operativa (1-16)
run	Numero di corsa relativa alla condizione operativa
VB	Tool wear
time	Durata della corsa
DOC	Profondità del taglio
feed	Velocità di avanzamento
material	Materiale
smcAC	Corrente AC del motore del mandrino
smcDC	Corrente DC del motore del mandrino
vib_table	Vibrazione del tavolo
Vib_spindle	Vibrazione del mandrino
AE_table	Emissione acustica del tavolo
AE_spindle	Emissione acustica del mandrino

Tabella 3.2: Descrizione dei campi del Milling dataset

1×167 struct with 13 fields

Fields	case	run	VB	time	DOC	feed	material	smcAC	smcDC	vib_table	vib_spindle	AE_table	AE_spindle
1	1	1	0	2	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
2	1	2	NaN	4	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
3	1	3	NaN	6	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
4	1	4	0.1100	7	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
5	1	5	NaN	11	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
6	1	6	0.2000	15	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
7	1	7	0.2400	19	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
8	1	8	0.2900	22	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
9	1	9	0.2800	26	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
10	1	10	0.2900	29	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
11	1	11	0.3800	32	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
12	1	12	0.4000	35	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
13	1	13	0.4300	38	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
14	1	14	0.4500	41	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
15	1	15	0.5000	44	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
16	1	16	NaN	46	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
17	1	17	0.4400	48	1.5000	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double
18	2	1	0.0800	3	0.7500	0.5000	1	9000×1 do…	9000×1 do…	9000×1 double	9000×1 double	9000×1 double	9000×1 double

Figura 3.2: Esempi di records della struttura MATLAB del *Milling dataset*

Mentre i sensori hanno raccolto dati in modo continuo, sia durante l'attività che durante il fermo macchina, le misurazioni del *flank wear* (VB) sono state effettuate in modo periodico. Data la diversità dei parametri di lavorazione, sono stati ottenuti cicli con tempo di attività e numero di lavorazioni differenti per ogni Caso. Inoltre, data la discontinuità delle misurazioni, non è stata rispettata una soglia massima fissa di *flank wear* ottenendo dei valori massimi piuttosto eterogenei. Infine, il parametro *flank wear* non è sempre stato misurato, e in taluni casi, nel rispettivo record del dataset, è stato inserito per VB un valore pari a NaN.

3.1.3 Acquisizione e pre-processing dei dati

I dati registrati dai sensori, 9000 campioni temporali per ciascuna corsa e con frequenza di campionamento pari a 250Hz, sono stati inviati al sistema di processamento attraverso una scheda di acquisizione dati ad alta velocità con frequenza di campionamento massima di 100 KHz. Per il campionamento dei dati in uscita è stata utilizzato il software LabVIEW12 per l'elaborazione del segnale generato dai sensori. Inoltre, i segnali dei sensori, eccetto quelli della corrente (sia alternata che continua) del motore del mandrino, sono stati sottoposti a un processo di pre-elaborazione come mostrato schematicamente nella Figura 3.3. Sia per i sensori di emissioni acustiche che per quelli di vibrazione, il segnale è stato amplificato per soddisfare i requisiti di soglia minima richiesti dalle apparecchiature di processamento dei segnali. Tali valori amplificati sono stati poi forniti in input ad un dispositivo RMS come riportato nell'Equazione seguente:

$$RMS = \sqrt{\frac{1}{\Delta T}\int_0^{\Delta T} f^2(t)dt} \tag{3.1}$$

L'RMS è dunque proporzionale al contenuto energetico del segnale, con ΔT un intervallo di tempo costante, fissato a $8,00ms$, e $f(t)$ è la funzione del segnale proveniente dai sensori a una data frequenza (nel caso in esame 250 Hz). Tale passaggio consente di attenuare il segnale dalle oscillazioni misurate, così da renderlo più accessibile all'elaborazione del processo.

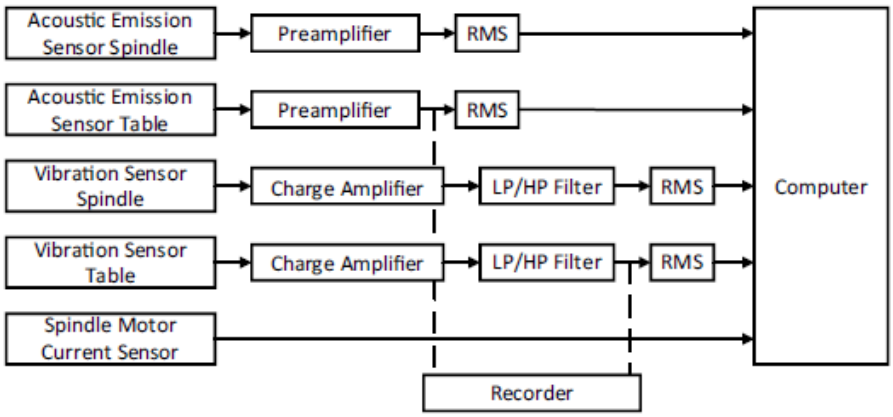


Figura 3.3: Acquisizione dati. Da A. Agogino and K. Goebel, "Milling Data Set."https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/, 2007.

3.2 Lavori correlati

Negli ultimi anni, in letteratura sono stati presentati differenti metodi e approcci prognostici, del tipo *data-driven* e basati su *machine learning* e *deep learning*, per la stima del *flank wear* e RUL e convalidati sul presente dataset. Di seguito vengono riportati alcuni lavori con le relative considerazioni implementative, suddivisi sulla base dell'approccio considerato.

3.2.1 Long short-term memory network (LSTM)

I segnali registrati dai sensori e collezionati dal dataset rappresentano sequenze temporali. Una soluzione generalmente implementata in tale contesto è quella che vede l'utilizzo delle *recurrent neural network* (RNN): reti neurali artificali le quali auto-estraggono le informazioni e dipendenze temporali contenute nelle serie storiche, mantenendo allo stesso tempo una bassa complessità computazionale. Tali reti tuttavia, mostrano delle limitazioni dovute alla bassa efficienza nell'apprendimento di dipendenze a lungo termine, presentando gradienti che tendono ad esplodere quando propagati per molti stadi successivi. Tra la gamma di soluzioni presentate per tale limitazione, si ha l'introduzione del modello *long short-term memory* (LSTM), la cui struttura permette l'apprendimento di dipendenze a lungo termine tramite l'inserimento dei cosiddetti *forget gates*.

Nel sistema definito in [32] viene utilizzata quest'architettura per l'estrazione delle features temporali dai *raw data* generati dai sensori. Il sistema implementato (Figura 4.2) è un modello d'informazione ibrido che prevede la combinazione delle features temporali, estratte tramite una rete LSTM, con le informazioni di processo (materiale, velocità di avanzamento, profondità del taglio). La combinazione genera un unico vettore, il quale va a costituire l'ingresso di un modello di regressione non lineare che a sua volta conduce alla stima del *flank wear*. Nello studio, le prestazioni del sistema sono state comparate con quelle dei modelli LR, SVR, CNN e MLP considerando i due scenari in cui vengano considerate o meno le informazioni di processo. I risultati proposti dagli autori mostrano come il sistema da loro presentato risulti maggiormente performante rispetto ai quattro modelli proposti, e poc'anzi riportati, ed inoltre la combinazione delle informazioni di processo permette di raggiungere un aumento dell'accuratezza media di un valore pari al 39%.

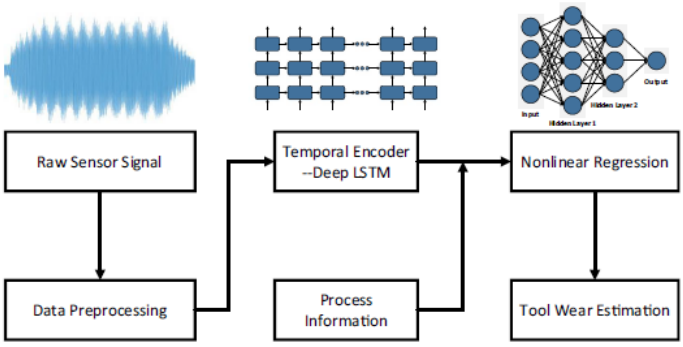


Figura 3.4: Framework presentato in [32]. Da W. Cai, W. Zhang, Y. Liu, and X. Hu, "A hybrid information model based on long short-term memory network for tool condition monitoring,"Journal of IntelligentManufacturing, vol. 31, p. 1497–1510, 2020.

3.2.2 Convolutional neural networks (CNN)

Un approccio differente è quello che prevede l'utilizzo delle 2D *convolutional neural network* (CNN) applicato alle sequenze temporali generate dai sensori e basato sul metodo della finestra scorrevole. Nel framework presentato in [33] viene utilizzato tale modello di deep learning per modellare le complesse relazioni esistenti tra le features estratte dai *raw data* dei sensori e il *flank wear*. Il framework (Figura 3.5) fa uso solamente del segnale di corrente del mandrino e ha come punto di partenza l'applicazione della trasformata Wavelet. Trasformata tempo-frequenza che permette di ottenere simultaneamente informazioni nel dominio del tempo e in frequenza. Successivamente, si applica una sottrazione spettrale in base alla quale, dallo spettro complessivo si va a rimuovere il contributo della componente normale stazionaria (intesa come componente relativa al condizione di buon funzionamento del siste-

ma), isolando così la componente additiva di guasto. Occorre sottolineare che lo spettro della componente normale stazionaria viene stimato a partire dai dati presenti nel dataset.
Infine l'estrazione delle features e il modello CNN costituiscono gli stadi finali del framework. Nello studio vengono confrontati con il modello CNN i metodi SVR, K-NN e la regressione *bayesian ridge*, concludendo infine la preminenza del modello CNN. Viene inoltre osservato come la sottrazione spettrale permetta di raggiungere un'accuratezza maggiore del 5% rispetto al caso in cui essa non venga integrata.

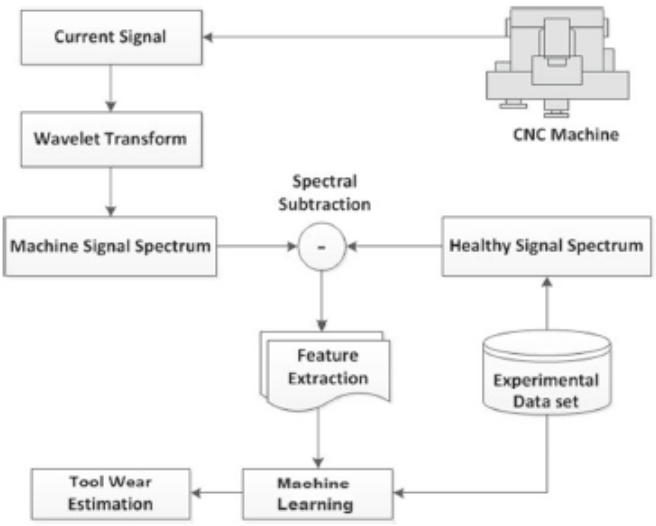


Figura 3.5: Framework presentato in [33]. Da F.Aghazadeh, A.Tahan, and M.Thomas, "Tool condition monitoring using spectral sub-traction and convolutional neural networks in milling process,"The International Journalof Advanced Manufacturing Technology, vol. 98, p. 3217–3227, 2018.

3.2.3 Temporal Convolutional Network (TCN)

Una soluzione proposta recentemente è quella che vede la combinazione, in unico sistema, del modello CNN con quello LSTM con lo scopo di lavorare con i loro singoli vantaggi, dimostratosi infatti essere complementari. Le reti CNN infatti, autoestraggono le features spaziali per mezzo di finestre temporali, mentre le reti LSTM autoestraggono le features temporali relative alle dipendenze temporali. Sulla base di ciò in [34] viene implementata tale combinazione, arrivando a definire una *temporal convolutional network* (TCN) (Figura 3.6).

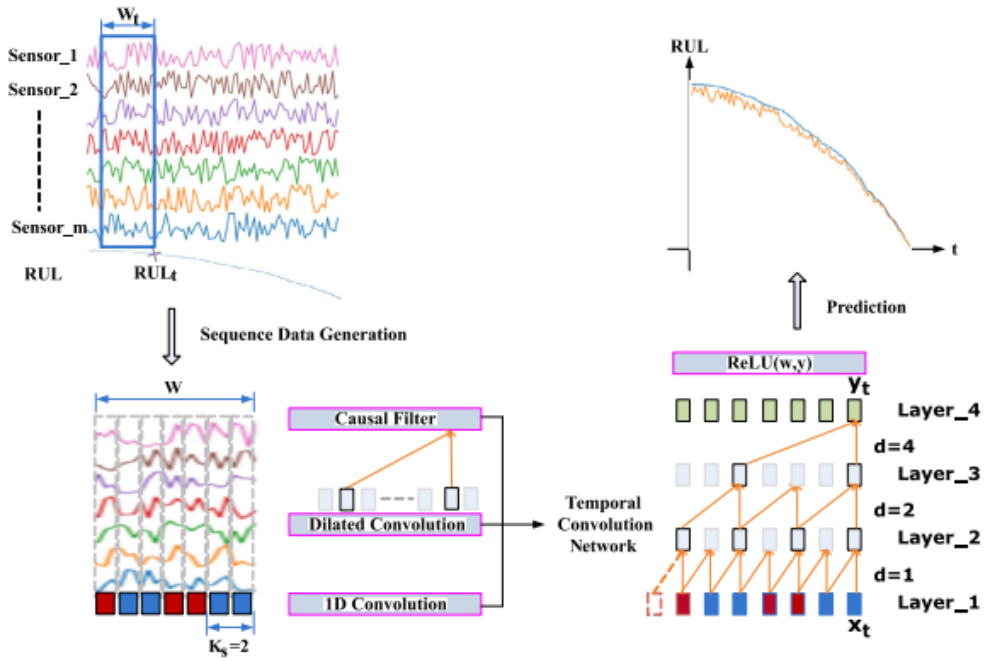


Figura 3.6: Framework presentato in [34]. Da J.Chen, D.Chen, and G. Liu, "Using temporal convolution network for remaining usefullifetime prediction,"Engineering Reports, vol. 3, no. 3, p. 1497–1510, 2021.

Questa tipologia di rete poggia le sue fondamenta sulle seguenti due componenti: una rete convoluzionale monodimensionale, così da generare una sequenza in uscita della stessa lunghezza di quella fornita in ingresso, e convoluzioni causali e dilatate. In merito a quest'ultime, la causalità comporta che in ciascun layer l'uscita all'istante t venga calcolata solamente sulla base dei valori, del layer precedente, relativi all'istante t o antecedenti ad esso. Tuttavia, il limite delle convoluzioni causali è dato dal fatto che se si volesse aumentare la memoria del modello occorrerebbe aumentare il numero di layer o utilizzare dei filtri di dimensioni maggiori. Nel caso dunque di lunghe sequenze in ingresso, si dovrebbe implementare una rete piuttosto profonda o filtri troppo grandi. Una soluzione allora, che non aumenta il carico computazionale, è l'integrazione di convoluzioni dilatate. Quest'ultime prevedono che il filtro venga applicato ad una sequenza di dimensione maggiore rispetto alla sua lunghezza saltando i valori di input con un passo costante.

Oltre alla struttura standard delle TCN in [34] viene inserito il concetto di blocco residuale delle ResNet per ovviare al problema della degradazione delle prestazioni e del *vanishing gradient* nel momento in cui si aumenta sempre più la profondità della rete.

Dai risultati, il sistema implementato non risulta performante in merito al materiale *steel* ed inoltre viene sottolineato dagli autori come esso incontri difficoltà nel caso in cui la RUL presenti intense fluttuazioni iniziali.

3.2.4 Autoencoder e similarity-based curve matching

In [35] e [36] viene proposto un approccio *data-driven* in cui la stima della RUL si basa sulla tecnica *similarity-based curve matching* applicata alle curve HI (*health index*). Tale metodo prevede che i *raw data* relativi ad una singola corsa vengano opportunamente trasformati in una curva HI unidimensionale, la quale rappresenta la degradazione dell'unità del sistema da una condizione di perfetto stato ($HI = 1$) a una condizione di guasto ($HI = 0$). Le curve HI così generate e relative a tutte le corse di training, vengono memorizzate in una libreria. Lo scopo finale è quello di stimare la RUL di una corsa di test usando quelle curve HI di training che mostrano una maggiore somiglianza con la curva HI di test.

Nei due lavori la costruzione delle curve HI, e dunque la stima dell'indice HI relativo a ciascuna corsa, si ottiene tramite un modello di regressione lineare il cui input è definito da features estratte dai *raw data*, tramite ad esempio la tecnica PCA come in [36]. Il training del modello di regressione prevede inoltre, come valori HI target, i valori estratti in precedenza da un autoencoder LSTM: una rete neurale artificiale, addestrata in modalità non supervisionata e costituita strutturalmente da un *encoder* e un *decoder*. L'*encoder* processa la sequenza di input e ne fornisce una rappresentazione compressa (*embedding vector*), dalla quale a sua volta il decoder ricostruisce la sequenza di input. Nella fase di training, l'*encoder* e il *decoder* aggiornano i rispettivi parametri al fine di definire un *embedding vector* significativo, in termini di bontà di ricostruzione dell'input.

Una volta terminata la fase di training, il decoder viene disattivato e si fa uso solamente della componente *encoder* al fine di estrarre una rappresentazione compressa per ciascun ingresso fornito.

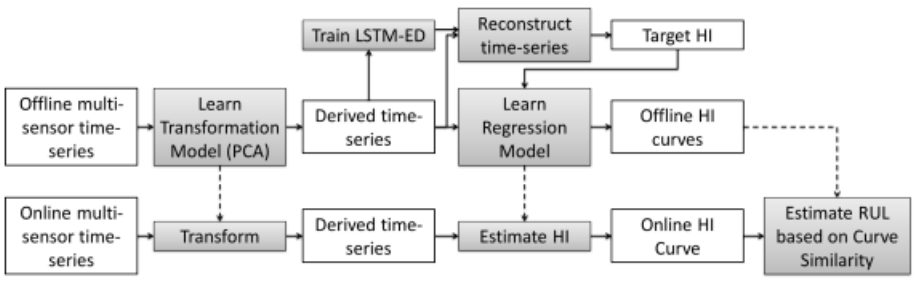


Figura 3.7: Framework presentato in [36]. Da P. Malhotra, T. Vishnu, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. M.Shroff, "Multi-sensor prognostics using an unsupervised health index based on lstmencoder-decoder,"ArXiv, vol. abs/1608.06154, 2016.

Una miglioria ai sistemi sopra proposti viene fornita in [37] e consiste nell'inserimento di un *attention layer* nell'autoencoder per distinguere, nella fase di costruzione dell'*embedding*

vector, il contributo delle informazioni relative a diversi timestamps. In tutti e tre i lavori considerati, i risultati hanno mostrato la validità dell'approccio *similarity-based curve matching*, sottolineando soprattutto come esso possa condurre a notevoli prestazioni anche laddove la dimensionalità del dataset utilizzato sia ridotta, come nel caso specifico del *Milling dataset*.

3.2.5 Domain Adversarial Transfer Learning (DATL)

Un paradigma completamente diverso, è quello seguito in [38] e il quale consiste nel *Domain Adversarial Transfer Learning* (DATL). Il *transfer learning* di solito mira a gestire il problema della mancanza di *label data* per i sistemi di destinazione. Come sappiamo infatti, gli approcci basati sull'apprendimento profondo (ad es. CNN, RNN, ecc.) richiedono una grande quantità di dati relativi sia alla condizione di buon funzionamento che a quella di guasto. Tuttavia, per un sistema di produzione, i primi vengono facilmente collezionati riuscendo così ad ottenerne una mole consistente mentre, per quanto riguarda i secondi la quantità ottenuta è fortemente ridotta e talvolta insufficiente. Occorre infatti notare come gli eventi di guasto siano rari a causa delle conseguenze insostenibili e gravi in cui si incorre nel corso e in seguito ad essi, ed occorre inoltre sottolineare il tempo richiesto dal potenziale processo di degrado prima che si verifichi il guasto desiderato. Tra la varietà di soluzioni a tale problema, un metodo consiste quindi nell'impiegare il *transfer learning*.

Il *transfer learning* consente di addestrare un modello di deep learning, su un determinato task o dominio sorgente, per poi essere applicato su un differente task o dominio target. Nel caso qui considerato il task rimane invariato, essendo la stima della RUL, ciò che varia sono i domini in input che si differenziano per le condizioni operative. Il concetto di reti neurali avversarie invece, viene introdotto per risolvere il problema del *covariate shift*, con il quale si intende la diversa distribuzione dei dati tra training set e testing set. In tale metodo è presente un classificatore di dominio il cui compito è definire da quale dominio, se sorgente o target, deriva la *feature* data in ingresso. Contemporaneamente, i layer di estrazione delle *features* ottimizzano i propri parametri al fine di rendere le *features* generate il più possibile indistinguibili tra loro.

In questo modalità avversaria, la rete è condotta alla definizione dei parametri dei layer di estrazione delle *features* in modo tale che essi mappino le features del dominio sorgente e target sulla stessa distribuzione.

Dal punto di vista strutturale l'intero sistema (Figura 3.8) prevede: una componente di estrazione delle features, il classificatore di dominio e un modello di regressione per la stima della RUL.

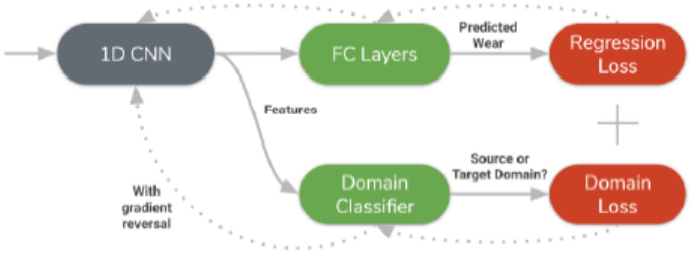


Figura 3.8: Framework presentato in [38]. Da P.Wang and M.Russell, "Domain adversarial transfer learning for generalized tool wearprediction,"Annual Conference of the PHM Society, vol. 12, no. 1, p. 8, 2020.

La prima è implementata con una rete CNN 1D al cui ingresso vengono forniti i segnali di vibrazione generati dai sensori. Come riportato infatti dagli autori, le reti CNN bidimensionali presentano buone prestazioni nel riconoscimento delle immagini mentre le reti CNN unidimensionali hanno buon successo su input sequenziali, come ad esempio i segnali di vibrazione. Le features così generate vengono date in ingresso sia al modello di regressione per la stima della RUL sia al classificatore di dominio. Quest'ultimo nel corso nel training minimizza la perdita di classificazione di dominio. Allo stesso tempo, retropropagando i gradienti dal classificatore di dominio ai layer di estrazioni di features, quest'ultimi concorrono alla massimizzazione della perdita di classificazione del dominio. Tutto ciò comporta la convergenza del sistema ad un'estrazione delle features tali da presentare la stessa distribuzione su entrambi i domini.

Nella sperimentazione sono stati utilizzati solo i casi relativi al *cast iron* scegliendo come dominio sorgente i casi con *feed*$= 0.25mm/s$ e i casi *feed*$= 0.5mm/s$ come dominio target. I risultati hanno mostrano che, in assenza di *transfer learning*, diminuendo nella fase di training la quantità di dati del dominio target, la differenza tra le prestazioni conseguite su tale dominio diventano sempre più marcate. Questo è dovuto al fatto che in tal caso, quando pochi campioni del dominio target sono disponibili, la rete è portata ad apprendere features specifiche relative al dominio sorgente. L'integrazione dunque del *transfer learning* permette di compensare la bassa disponibilità di label target.

3.2.6 Classificatore SVM

A differenza degli approcci sopra seguiti i quali prevedevano la stima del parametro RUL o *tool wear*, in [39] viene implementato un classificatore a tre classi il quale identifica la condizione del *tool wear* sulla base di tre possibili classi: (a) normale (*tool wear* $< 0.2mm$), (b) warning ($0.2mm \leq$ *tool wear* $\leq 0.4mm$), (c) failure (*tool wear* $0.4mm$). Per la classificazione

viene usata la tecnica *Support Vector Machine (SVM)* la quale in un primo momento genera differenti iperpiani per poi definire l'iperpiano ottimo che separa nel modo più efficiente le diverse classi identificate da opportune features. In tal caso, al fine di classificare i segnali non lineari vengono integrate nel metodo delle funzioni *kernel* non lineari. In particolar modo vengono confrontate le prestazioni del sistema usando le tre diverse funzioni *kernel*: lineare, polinomiale e gaussiana.
Il sistema prevede inizialmente l'estrazione di 14 differenti features dai *raw data*. In seguito, a causa dell'elevata dimensionalità del dataset delle features viene applicata la tecnica PCA (*principal component analysis*): tecnica matematica che permette di mappare un dataset multi-dimensionale in un dataset con dimensioni ridotte pur mantenendo le informazioni globali. Il sistema ha riscontrato elevati valori di accuracy (87%), tuttavia in virtù degli approcci finora presentati, la classificazione del *flank wear* e non la sua stima puntuale, porta tale modello ad essere meno performante in un tipico contesto applicativo industriale.

3.2.7 Confronto modelli di regressione e classificazione

In [40] viene presentato un framework (Figura 3.9) costituito due modelli di regressione per il monitoraggio del *flank wear* (VB) e stima della RUL, e un modello di classificazione per determinare se in un determinato istante l'utensile da taglio debba essere sostituito o meno. Il lavoro ha avuto come obbiettivo il confronto di diversi modelli di regressione e classificazione con lo studio delle relative prestazioni. Strutturalmente il punto di partenza è l'estrazione delle features dai *raw data*, seguito da una fase di *data cleaning* e *features selection*. A questo proposito, come criteri di selezione sono stati considerati il coefficiente di correlazione a la multicollinearità, seguendo l'approccio proposto in [41].
Per quanto riguarda la parte predittiva del sistema per la definizione della RUL e del VB sono stati studiati e comparati diversi modelli di regressione:

- Linear Regression (LR)
- Decision Forest regression (DF)
- Bayesian Linear Regression (BLR)
- Boosted Decision Tree regression (BDT)
- Neural Network regression (NN)

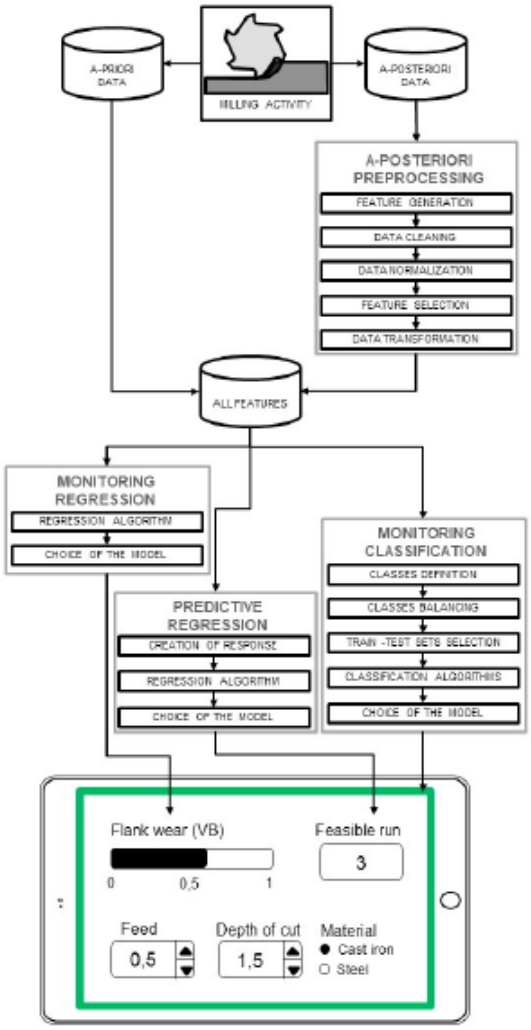


Figura 3.9: Framework presentato in [40]. Da E. Traini, G. Bruno, G. D'Antonio, and F. Lombardi, "Machine learning framework forpredictive maintenance in milling,"IFAC-PapersOnLine, vol. 52, no. 13, pp. 177–182,2019.

Mentre per la fase di classificazione è stato implementato un classificatore binario di classi *safe* e *worn* definite dal superamento o meno del parametro VB della soglia VB_{max}. Anche in questo sono stati confrontati diversi modelli:

- Logistic Regression (LogR)
- Decision Forest (DF)
- Decision Jungle (DJ)
- Boosted Decision Tree (BDT)
- Neural Network (NN)

Come metodo per il bilanciamento dei dati viene utilizzata la tecnica SMOTE (*The Synthetic Minority Oversampling Technique*). I risultati presentati individuano la regressione NN come miglior algoritmo per la definizione sia della RUL che del VB mentre il modello *Boosted Decision Tree* per il compito di classificazione.

Capitolo 4

Metodologia

L'approccio presentato per lo studio del *Milling dataset* ha previsto la definizione e implementazione di un framework appartenente alla categoria dei metodi *data-driven* basati su algoritmi di *machine learning* e *deep learning*. Ai fini di un'agevole comprensione dell'approccio seguito è utile sottolineare come la categoria dei metodi *data-driven* venga generalmente suddivisa, sulla base della formulazione della stima della RUL, in due sotto-categorie principali: *a)* gli approcci diretti, i quali applicano e definiscono un mapping diretto tra i *raw data* e RUL e *b)* gli approcci indiretti, i quali invece prevedono, a partire dai *raw data*, una fase iniziale di definizione e valutazione dell'indice HI e successivamente l'utilizzo di questo nella stima della RUL (Figura 4.1).

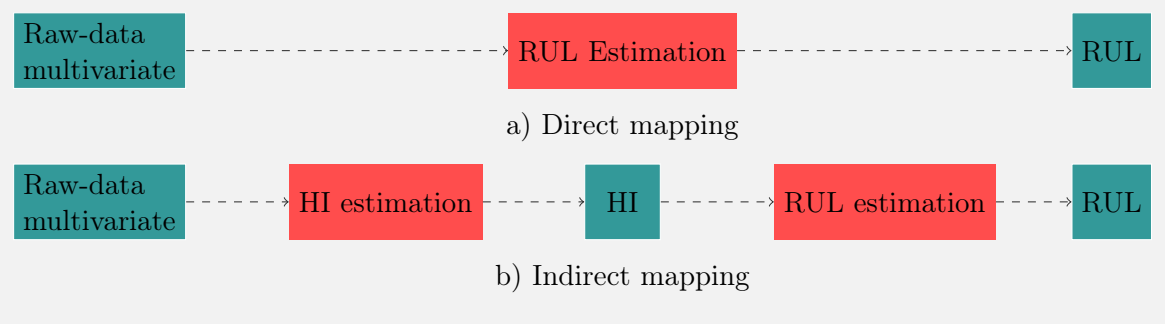


Figura 4.1: Due principali sottogruppi relativi agli approcci *data-driven*: *a)* mappatura diretta tra input e RUL, e *b)* mappatura indiretta tra input e RUL usando l'approccio *HI-based*

Il primo gruppo definisce quindi direttamente il mapping tra gli input e l'output target, ovvero RUL, tramite alcuni popolari modelli di apprendimento automatico, come ad esempio *Support Vector Regression* (SVR) e le sue varianti, *Neural Networks* (NN), combinazione di *Support Vector Machine* (SVM) e SVR e infine *Extreme Learning Machine* (ELM). Normalmente, gli input sono vettori di features accuratamente selezionate ed estratte dai *raw data* e il mapping

viene appreso in modalità supervisionata, assegnando a priori *label* RUL ai corrispondenti vettori di *features* in input. Sebbene gli approcci prognostici diretti siano stati sfruttati per la loro semplicità e facilità di implementazione, spesso mancano della capacità di monitorare il degrado del sistema e della flessibilità nell'incorporare la conoscenza del dominio pertinente. Inoltre, i confronti hanno mostrato che questi metodi possono, nella maggior parte dei casi, ottenere solo prestazioni comparabili o peggiori, in relazione alle previsioni RUL basate su HI. Gli approcci indiretti invece, utilizzano l'indice HI come indicatore intermedio dello stato di salute intrinseco di un sistema, consentendo l'integrazione di conoscenze di dominio rilevanti, prima di stimare la vita residua utile (RUL). Qui, la modellazione e l'analisi del degrado sono prerequisiti, generalmente con l'assunzione di ipotesi implicite o esplicite sulla forma del degrado. In particolar modo,in un primo momento il modello implementato estrapola l'indice HI dai *raw data* e successivamente un secondo modello definisce il mapping $HI \rightarrow RUL$.

Descritto in Figura 4.2, il framework proposto rientra nella categoria degli approcci *HI-based* e prevede:

1. Il modello *autoencoder* e regressione lineare rispettivamente per l'estrazione dell'indice HI dai *raw data* e la successiva costruzione delle curve HI

2. La tecnica *similarity-based curve matching* per la stima della RUL a partire dall'indice HI. L'idea alla base di questa tecnica è quella di confrontare l'attuale trend di degrado del sistema, rappresentato da una traiettoria dell'indice HI, con una libreria di curve HI relative alle sperimentazioni pregresse e tra queste curve utilizzare solo quelle più simili per stimare direttamente la RUL. Il vantaggio indiscutibile di tale tecnica è che essa non necessita di alcuna ipotesi per costruire il modello di degrado e al contempo può lavorare su dataset anche limitati dimensionalmente ottenendo risultati di previsione più accurati anche quando i dati mostrano tendenze distintive e fluttuazioni di grande entità.

Architetturalmente e concettualmente si compone di due parti principali: *a)* Fase offline, volta alla costruzione della libreria di curve HI relative ai Casi di training e *b)* Fase online, in cui dato un Caso di test si arriva a stimare la rispettiva RUL.

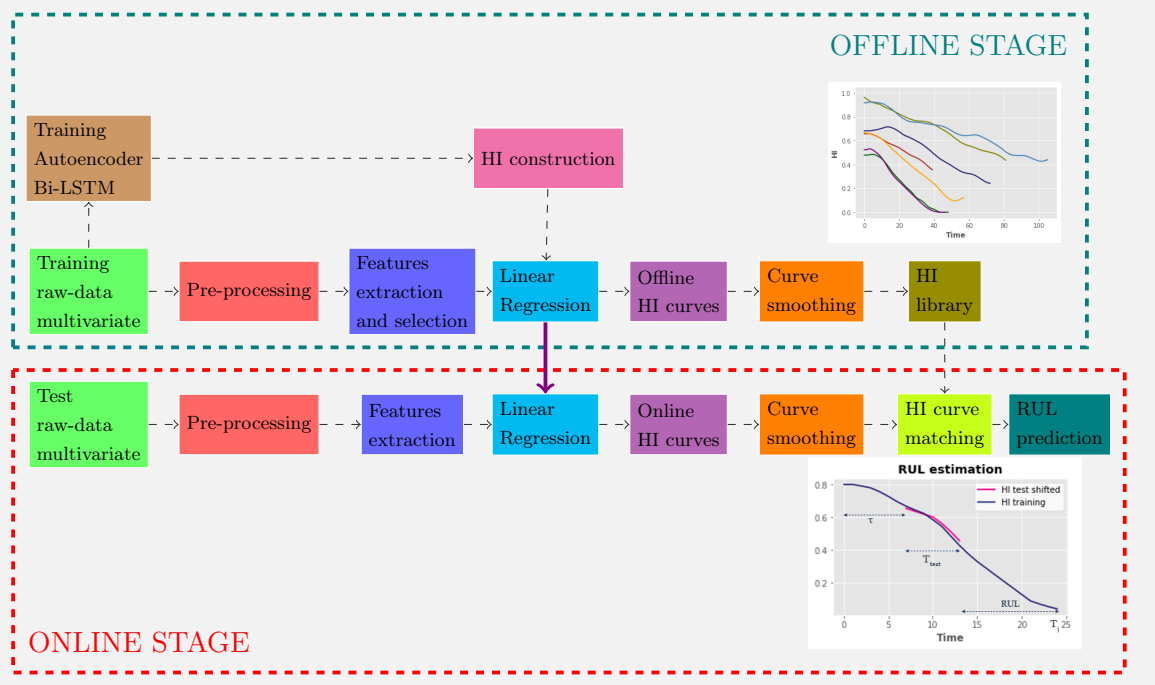


Figura 4.2: Framework proposto per il *Milling dataset*

4.1 Descrizione framework

Come descritto in precedenza nella Sezione 3, il *Milling dataset* si compone di 16 Casi. Ogni Caso rappresenta il monitoraggio del sistema, in una determinata condizione operativa, nell'arco temporale che ha inizio dalla condizione di buon funzionamento ed arriva a quella di malfunzionamento o guasto, comprendendo un numero diverso di corse (lavorazioni). Una corsa non è altro che una serie temporale multi-variata di 9000 campioni registrati consecutivamente dai sensori.

Con l'obbiettivo di facilitare la comprensione della procedura seguita, viene presentata in un primo momento in Tabella 4.1 la notazione utilizzata e la definizione strutturale delle grandezze considerate, successivamente viene illustrato nel dettaglio il profilo del framework.

Notazione	Descrizione	Forma
$\mathbf{I_j}$	j-esimo Caso del dataset di durata T_j. Comprende N_j corse che nella loro successione rappresentano la storia del sistema da una condizione di perfetto funzionamento a quella di malfunzionamento, sotto una determinata condizione operativa. In questo modo il *Milling dataset*, indicato con $\mathbf{D}$, può essere definito come: $\mathbf{D} = \{\mathbf{I_1}, \mathbf{I_1}, \mathbf{I_3}, ..., \mathbf{I_{16}}\}$	$\mathbf{I_j} = \{\mathbf{X_1}, \mathbf{X_2}, ...\mathbf{X_i}, ..., \mathbf{X_{N_j}}\}$
$\mathbf{X_i}$ (o $\mathbf{X_i^{(j)}}$)	i-esima corsa all'interno del j-esimo Caso. Strutturalmente è una sequenza temporale multivariata di 9000 campioni registrati dai sensori	$\mathbf{X_i} = \{\mathbf{x_1}, \mathbf{x_2}, ..\mathbf{x_t}.., \mathbf{x_{9000}}\}$
$\mathbf{x_t}$ (o $\mathbf{x_t^{(ij)}}$)	Vettore che contiene i dati registrati dai sei sensori all'istante t	$\mathbf{x_t} = \begin{pmatrix} x_{t1} \\ x_{t2} \\ \vdots \\ x_{t6} \end{pmatrix}$

Tabella 4.1: Descrizione strutturale delle grandezze del dataset

Prendendo come esempio il j-esimo Caso, ai *raw data* relativi all'i-esima corsa $\mathbf{X_i}$ viene applicata preliminarmente, una fase di *pre-processing*, volta al *data cleaning* e con lo scopo di migliorare la qualità dei dati. Una volta gestiti quindi gli outliers, i dati invalidi e duplicati, la sequenza temporale ottenuta segue simultaneamente due vie.

In primo luogo essa viene utilizzata per il training di un *autoencoder bidirezionale* LSTM, il cui addestramento è volto alla costruzione di indici HI target $h^{(Tg)}$ in modalità non supervisionata. In questo modo per ogni corsa in ingresso $\mathbf{X_i} = \{\mathbf{x_1}, \mathbf{x_2}, ..\mathbf{x_t}.., \mathbf{x_{9000}}\}$ viene generato e costruito il relativo indice HI target $h_i^{(Tg)}$ ottenendo la coppia $(\mathbf{X_i}, h_i^{(Tg)})$.

La seconda via prevede, per ciascuna corsa, l'estrazione di un set di *features* da utilizzare successivamente come input per un modello di regressione lineare. Questo modello (Equazione 4.1) permette di stimare l'indice h_i per ogni corsa, quando questa è rappresentata da un vettore di features m-dimensionale: $\mathbf{f_i} = \{f_1, f_2, ..., f_m\}$:

$$h = \beta_0 + \boldsymbol{\beta}^{\mathbf{T}}\mathbf{f} \tag{4.1}$$

Rientrando la regressione lineare nella categoria degli algoritmi di *machine learning* supervisionati, la fase di training richiede dati etichettati. In questo caso, nella fase di training del modello di regressione lineare, per ogni vettore di *features* in ingresso $\mathbf{f_i}$, relativo alla singola corsa $(\mathbf{X_i})$, viene utilizzato come output osservato l'indice HI target $h_i^{(Tg)}$ generato in precedenza tramite l'autoencoder; in altre parole, il training del modello si base sulle coppie

di variabili $\{(\mathbf{f_i}, h_i^{(Tg)})\}$.

Per quanto concerne le *features* estratte e incluse nel modello, a partire da un set iniziale è stato selezionato un sottoinsieme di features ottimo adoperando come metodo di *features selection* l'algoritmo *stepwise regression*.

Una volta addestrato il modello di regressione lineare, questo viene utilizzato per stimare l'indice HI di ciascuna corsa. In questo modo, considerando il j-esimo Caso $\mathbf{I_j}$, costituito da N_j corse, si giunge a mappare la rispettiva sequenza temporale di corse in una sequenza temporale di indici HI:

$$\mathbf{I_j} = \{\mathbf{X_1}, \mathbf{X_2}, ...\mathbf{X_i}, ..., \mathbf{X_{N_j}}\} \Longrightarrow \mathbf{HI_j} = \{h_1, h_2, ...h_i, ..., h_{N_j}\} \tag{4.2}$$

Ottenendo così una curva HI ($\mathbf{HI_j}$).

Applicando la procedura fin qui illustrata per tutti gli M Casi di training si arrivano ad ottenere M curve HI, ciascuna delle quali descrive l'andamento nel tempo del degrado del sistema in quel particolare Caso. Da un punto qualitativo, prima che tali curve vadano a popolare la libreria offline $\mathbf{\Gamma} = \{\mathbf{HI_j}\}$, si vanno ad applicare ad esse delle tecniche di *smoothing* a causa della loro elevata rumorosità.

Nella fase online di test, si lavora sul Caso di test $\mathbf{I_{test}}$ composto di N_T corse[1]. Per ciascuna di esse, conseguentemente alla fase di *pre-processing* e *feature extraction*, si stima il rispettivo indice HI h^{test} utilizzando il modello di regressione lineare addestrato nella fase offline.

Iterando questo procedimento per tutte le N_T corse, si arriva ad ottenere la curva HI del Caso di test. A questo punto, la stima della RUL finale prevede la fase di *curve matching*, ovvero il calcolo della somiglianza, espressa da un'opportuna metrica, tra la curva di test e ciascuna curva HI presente nella libreria $\mathbf{\Gamma}$. Una volta selezionate quelle curve della libreria che presentano un livello di somiglianza maggiore, per ciascuna di esse si calcola la singola RUL. Da ultimo, la RUL finale non è altro che una media ponderata delle singole RUL così calcolate e i pesi sono espressi dal corrispondente grado di somiglianza.

Nelle Sottosezioni successive vengono presentati e descritti, da un punto di vista teorico, i principali algoritmi e metodi integrati nel framework: *autoencoder* Bi-LSTM (Sottosezione 4.2), Regressione Lineare (Sottosezione 4.3), *stepwise regression* (Sottosezione 4.4).

[1] $\mathbf{I_{test}} = \{\mathbf{X_1}, \mathbf{X_2}, ...\mathbf{X_i}, ..., \mathbf{X_{N_T}}\}$

4.2 Autoencoder Bi-LSTM

4.2.1 Autoencoder

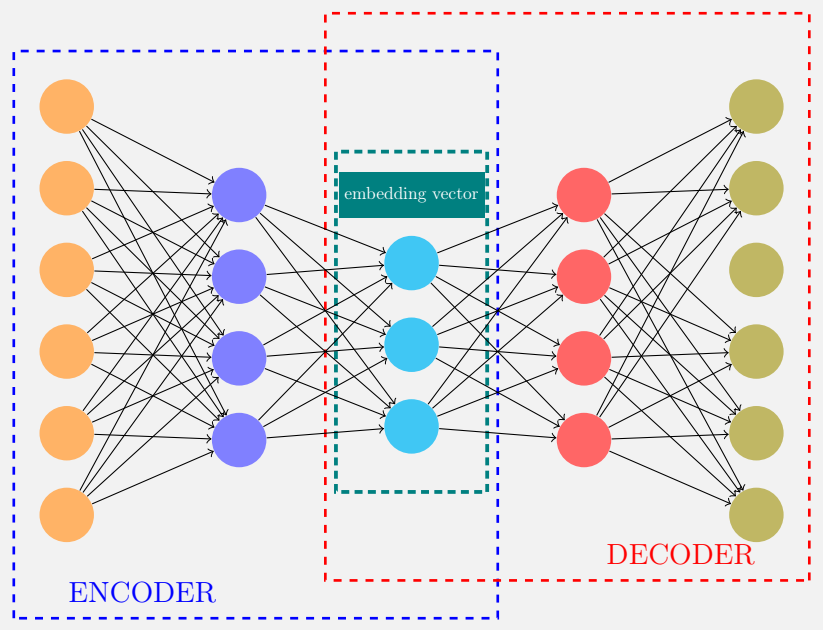


Figura 4.3: Architettura *autoencoder*

Un autoencoder è una tipologia di rete neurale artificiale, il cui principale obbiettivo è quello di apprendere la rappresentazione dei dati in input, estraendo features non lineari da questi [42]. Per raggiungere tale obbiettivo l'autoencoder consiste di tre *layer*: input, hidden e output layer; e in esso si vanno generalmente a considerare due parti: *encoder* e *decoder*.
Formalmente, dato in ingresso un input del tipo $\mathbf{x} = \{x_1, x_2, ..., x_p\} \in \mathbb{R}^{1\times p}$, l'encoder, costituito da input e hidden layer, mappa tale input in una rappresentazione compressa $\mathbf{z} = \{z_1, z_2, ..., z_n\} \in \mathbb{R}^{1\times n}$ denominata *embedding vector* secondo la seguente relazione:

$$\mathbf{z} = f_{activation}(\mathbf{W_{xz}x} + \mathbf{b_{xz}}) \tag{4.3}$$

Dove $f_{activation}(\cdot)$ è una funzione di attivazione che effettua il mapping non lineare, $\mathbf{W_{xz}x}$ e $\mathbf{b_{xz}}$ sono rispettivamente la matrice dei pesi e il vettore dei bias della rete neurale.
E' importante sottolineare come la rappresentazione $\mathbf{z}$ appartenga ad uno spazio di dimensionalità ridotta rispetto $\mathbf{x}$, poiché se così non fosse l'utilizzo dell'autoencoder non porterebbe ad alcun vantaggio.
Successivamente, il decoder, costituito da hidden e output layer, a partire dalla rappresenta-

zione $\mathbf{z}$ ricostruisce l'input originale $\mathbf{x}$, secondo il seguente processo di decodifica:

$$\mathbf{x'} = g_{activation}(\mathbf{W_{zx'}x} + \mathbf{b_{zx'}}) \tag{4.4}$$

Dove $g_{activation}(\cdot)$ è una funzione di attivazione che effettua il mapping non lineare, $\mathbf{W_{zx'}x}$ e $\mathbf{b_{zx'}}$ sono rispettivamente la matrice dei pesi e il vettore dei bias della rete neurale, mentre $\mathbf{x'} = \{x'_1, x'_2, ..., x'_p\} \in \mathbb{R}^{1\times p}$ è la ricostruzione dei dati.
Nella fase di training dell'autoencoder le due componenti lavorano simultaneamente e vengono addestrate al fine di realizzare un'*identity mapping* dove l'output $\mathbf{x'}$ approssima quanto meglio possibile l'input corrente $\mathbf{x}$. A tal proposito, in tale fase, la definizione dei parametri delle due componenti è volta alla minimizzazione di un'opportuna *loss function*, come ad esempio la quantità $\mathbf{E}=\|\mathbf{x\text{-}x'}\|$, che quantifica la differenza tra l'input originale e la conseguente ricostruzione. In altre parole, l'autoencoder è addestrato per definire una rete neurale che presenti input ed output identici e quindi per imparare a ricostruire l'input a partire da una sua rappresentazione compressa.
Per concludere, è necessario sottolineare come, una volta terminata la fase di training, l'autoencoder viene utilizzato per estrarre da ciascun input la relativa rappresentazione $\mathbf{z}$, rispondendo in questo modo al problema della riduzione della dimensionalità; inoltre, non prevedendo *target data* nella fase di training rientra nella categoria dell'apprendimento non-supervisionato.

4.2.2 Recurrent Neural Network

Una *Recurrent Neural Network* (RNN) è un tipologia di rete neurale artificiale che lavora su dati sequenziali o dati relativi a serie temporali. Tale classe di algoritmi di deep learning vengono comunemente usati per problemi ordinali o temporali, come ad esempio la traduzione linguistica, l'elaborazione del linguaggio naturale e il riconoscimento vocale; venendo così incorporati in applicazioni popolari quali Siri, ricerca vocale e Google Translate. Sulla falsariga delle reti neurali *feedforward* e convoluzionali (CNN), le reti neurali ricorrenti utilizzano i dati di addestramento per apprendere il mapping input-output; tuttavia, si distinguono da queste per la presenza di una componente di "memoria" che permette all'output corrente di essere influenzato da input processati in precedenza. É doveroso inoltre aggiungere, come anche gli eventi futuri possano essere utili nella determinazione dell'output corrente; ciò risulta possibile integrando la classe di reti RNN di tipo bidirezionale.
Entrando nel merito delle reti RNN, la componente di memoria che le contraddistingue è

definita come uno stato nascosto ricorrente (h_t) la cui attivazione all'istante t dipende da quella dell'istante precedente.

Formalmente, data in ingresso una sequenza di dati $\{x_k\}$, il neurone ricorrente all'istante t aggiorna il suo stato nascosto h_t e fornisce l'output O_t sulla base delle seguenti relazioni:

$$h_t = \tanh\left(\mathbf{U}_{\mathbf{x_t}} + \mathbf{W}_{\mathbf{h_{t-1}}} + b\right) \tag{4.5}$$

$$O_t = f(\mathbf{V}_{\mathbf{h_t}} + c) \tag{4.6}$$

dove $\mathbf{U}$, $\mathbf{W}$ e $\mathbf{V}$ sono le matrici dei pesi definite nella fase di training, rispettivamente per le connessioni *input-hidden*, *hidden-to-hidden* e *hidden-to-output*, mentre b e c sono i vettori di bias che consentono a ciascun neurone di apprendere un offset. Generalmente, la funzione di attivazione nel processo di aggiornamento dello stato nascosto corrente h_t è $\tanh(\cdot)$, al fine di mantenere lo stato nascosto nell'intervallo $[-1, 1]$, mentre la funzione di attivazione dell'uscita, $f(\cdot)$, dipende dal task del sistema in esame. Lo stato nascosto corrente h_t contiene quindi tutte le informazioni precedenti, e l'output O_t è influenzato sia dalle informazioni di input correnti che da quelle storiche. In virtù di tutta la descrizione, come mostrato in Figura 4.4, la rete RNN lavora sulla sequenza di input passo dopo passo, processando sequenzialmente ciascun istante temporale.

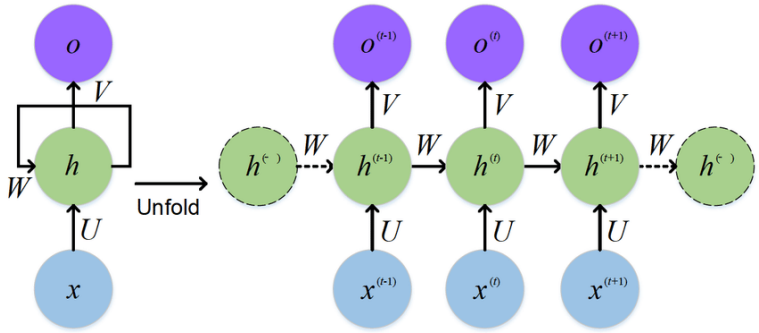


Figura 4.4: Architettura *Recurrent Neural Network*. Da W.Feng, N.Guan, Y. Li, X. Zhang and Z. Luo, (2017). Audio visual speech recognition with multimodal recurrent neural networks. 681-688. 10.1109/IJCNN.2017.7965918.

Nonostante le reti RNN siano una delle soluzioni migliori nello studio delle sequenze temporali, è stato osservato come non siano performanti nell'estrazione di lunghe dipendenze temporali. Tale limitazione è causata dalla probabilità non nulla di incorrere, nella fase di training, nel calcolo di gradienti che "esplodono" o "svaniscono", rispettivamente *exploding gradient* e *vanish gradient*, a sua volta dovuti alla modalità di *Backpropagation through time* (BPTT) e alla struttura base della rete RNN.

Il training di una rete RNN viene eseguito definendo una funzione di perdita $\mathcal{L}$, che quantifica l'errore tra l'output reale $\mathbf{O}$ e l'output calcolato $\hat{\mathbf{O}}$, ed è volto alla minimizzazione di tale quantità combinando consecutivamente gli step *forward* e *backward*:

$$\mathcal{L} = \sum_i L_i(O_i, \hat{O}_i) \tag{4.7}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}}, \frac{\partial \mathcal{L}}{\partial \mathbf{V}}, \frac{\partial \mathcal{L}}{\partial \mathbf{W}}, \frac{\partial \mathcal{L}}{\partial b}, \frac{\partial \mathcal{L}}{\partial c} \tag{4.8}$$

Nella fase *forward*, per ogni singolo step temporale k, considerando l'input corrente x_k si calcola l'output corrente $\hat{O}_k$ e di conseguenza la relativa funzione di perdita $L_i(O_i, \hat{O}_i)$. Procedendo per tutti gli istanti temporali si giunge a definire la funzione di perdita totale $\mathcal{L}$, concludendo così lo step *forward*.
La fase successiva, denominata *backward*, prevede invece il calcolo delle derivate descritte in 4.8 alla base dell'aggiornamento dei pesi della rete:

$$\mathbf{W} \longrightarrow \mathbf{W} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{W}} \tag{4.9}$$

Prendendo come esempio di riferimento la matrice dei pesi $\mathbf{W}$, a causa della dipendenza dello stato nascosto corrente h_t da tutti gli stati nascosti agli istanti precedenti il calcolo di $\frac{\partial \mathcal{L}}{\partial \mathbf{W}}$ diventa:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \sum_{t=0}^{T} \frac{\partial L_t}{\partial \mathbf{W}} \tag{4.10}$$

$$\frac{\partial L_t}{\partial \mathbf{W}} = \sum_{k=0}^{t} \frac{\partial L_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial \mathbf{W}} \tag{4.11}$$

$$= \sum_{k=0}^{t} \frac{\partial L_t}{\partial h_t} \left(\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial \mathbf{W}} \tag{4.12}$$

Si osserva come nel calcolo del gradiente della funzione di perdita calcolata all'istante t ($\frac{\partial L_t}{\partial W}$), il contributo dello stato all'istante k è vincolato dalla quantità $\left(\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \right)$.
In particolar modo considerando le due casistiche:

$$\text{Vanish gradient} \longrightarrow \left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 < 1 \tag{4.13}$$

$$\text{Exploding gradient} \longrightarrow \left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 > 1 \tag{4.14}$$

si può incorrere rispettivamente nei problemi: *exploding gradient* e *vanish gradient*.

Nel primo caso, moltiplicando numerosi termini del tipo < 1 i valori del gradiente si riducono in modo esponenziale, arrivando a svanire completamente solo dopo alcuni istanti temporali considerati. I contributi del gradiente dagli istanti "lontani" diventano zero e di conseguenza lo stato in quegli istanti non contribuisce allo stato e all'output corrente; si finisce così per non apprendere le dipendenze a lungo raggio.
Nel secondo caso, il termine va all'infinito in modo esponenziale, e il valore del gradiente diventa presto un NaN a causa del processo instabile. É facile immaginare come, a seconda delle funzioni di attivazione utilizzate e dei parametri di rete, sia possibile incorrere in uno dei due casi. Tuttavia, il problema del *vanish gradient* ha ricevuto una maggiore attenzione da parte della letteratura scientifica essendo più problematico, difficile da verificare e affrontare. Questo perché l'*exploding gradient* comportando la presenza di valori NaN nella fase di training porta all'arresto automatico del programma e quindi del training. Inoltre, esso presenta una possibile soluzione definita *gradient clipping* che impone un limite superiore ai possibili valori assunti dal gradiente. Fortunatamente, ci sono alcuni metodi per affrontare il problema del *vanish gradient*. Una corretta inizializzazione della matrice $\mathbf{W}$ può ridurne l'effetto, così come la regolarizzazione o anche l'utilizzo della funzione di attivazione $ReLu$, invece delle funzioni di attivazione $tanh$ o $sigmoide$.
Tuttavia, una soluzione ancora più diffusa in letteratura consiste nell'utilizzare l'architettura Long Short-Term Memory (LSTM).

4.2.3 Long-short term memory

Una rete *long-short term memory* (LSTM) [43] rientra nella categoria delle *recurrent neural networks* ma a differenza delle classiche RNN presenta e introduce in ciascuna cella di memoria (Figura 4.5) tre differenti *gate*: *input*, *output* e *forget gate*. Lo scopo dei *gate* è quello di valutare e definire quali informazioni ricevute conservare e quali escludere. In questo modo, nel corso del processamento della sequenza di input vengono utilizzate e trasmesse solamente le informazioni più rilevanti, risolvendo in questo modo il problema delle reti RNN dell'estrazione delle dipendenze a lungo termine.

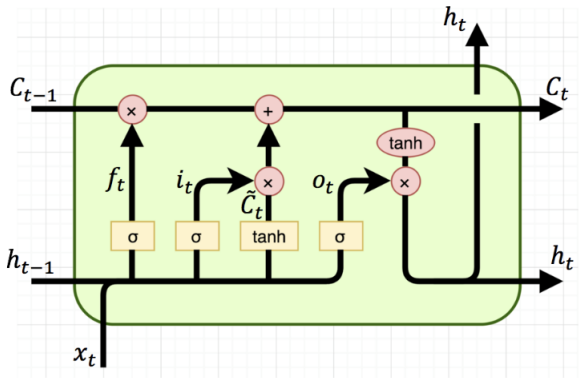


Figura 4.5: Architettura cella LSTM. Da https://laptrinhx.com/lstm-gradients-1235477976/

In particolar modo, fornita in ingresso la sequenza $\mathbf{x} = \{x_1, x_2, .., x_T\}$, sequenzialmente da $t = 1$ a $t = T$, la rete aggiorna i parametri *input gate*, *output gate* e *forget gate* sulla base dell'input corrente x_t e dello stato interno all'istante temporale precedente h_{t-1}. Infine vengono anche aggiornati lo stato interno corrente h_t e lo stato corrente della cella C_t. All'istante t la fase di aggiornamento prevede le seguenti equazioni:

$$f_t = \sigma(\mathbf{U_f} x_t + \mathbf{W_f} h_{t-1} + b_f) \tag{4.15}$$

$$i_t = \sigma(\mathbf{U_i} x_t + \mathbf{W_i} h_{t-1} + b_i) \tag{4.16}$$

$$o_t = \sigma(\mathbf{U_o} x_t + \mathbf{W_o} h_{t-1} + b_o) \tag{4.17}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t = f_t \odot C_{t-1} + i_t \odot \tanh(\mathbf{U_C} x_t + \mathbf{W_C} h_{t-1} + b_C) \tag{4.18}$$

$$h_t = O_t = o_t \odot \tanh C_t \tag{4.19}$$

Dove:

- $\mathbf{O} = \{O_1, O_2, .., O_T\}$ è la sequenza di output;
- $\mathbf{U_f}, \mathbf{U_i}, \mathbf{U_o}$ e b_f, b_i, b_o rappresentano rispettivamente le matrici dei pesi e i bias che definiscono la relazione tra i relativi gate e l'input corrente x_t
- $\mathbf{W_f}, \mathbf{W_i}, \mathbf{W_o}$ rappresentano le matrici dei pesi che descrivono la relazione tra i rispettivi gates e lo stato interno h_{t-i}, o dualmente l'output O_{t-i}, entrambi relativi all'istante precedente.
- $\sigma(\cdot)$ è una funzione di attivazione non lineare che riporta i valori dei gate all'interno un determinato range. La scelta più comune per tale funzione di attivazione risulta essere la funzione *sigmoidea* il cui codominio è l'intervallo $[0, 1]$.

- Nell'Equazione 4.18 l'aggiornamento dello stato corrente della cella C_t avviene combinando lo stato della cella all'istante precedente C_{t-1}, regolato dal forget gate, e l'output, regolato dall'input gate, della funzione di attivazione tanh(·) che tiene conto dello stato corrente e dello stato interno all'istante precedente .

- Infine, l'Equazione 4.19 rappresenta il processo di aggiornamento dello stato interno corrente h_t, qui espresso sulla base dello stato corrente della cella C_t.

4.2.4 Autoencoder Bi-LSTM

L'autoencoder implementato nel framework e mostrato in Figura 4.6 rientra nella tipologia *Bidirectional* LSTM (Bi-LSTM).

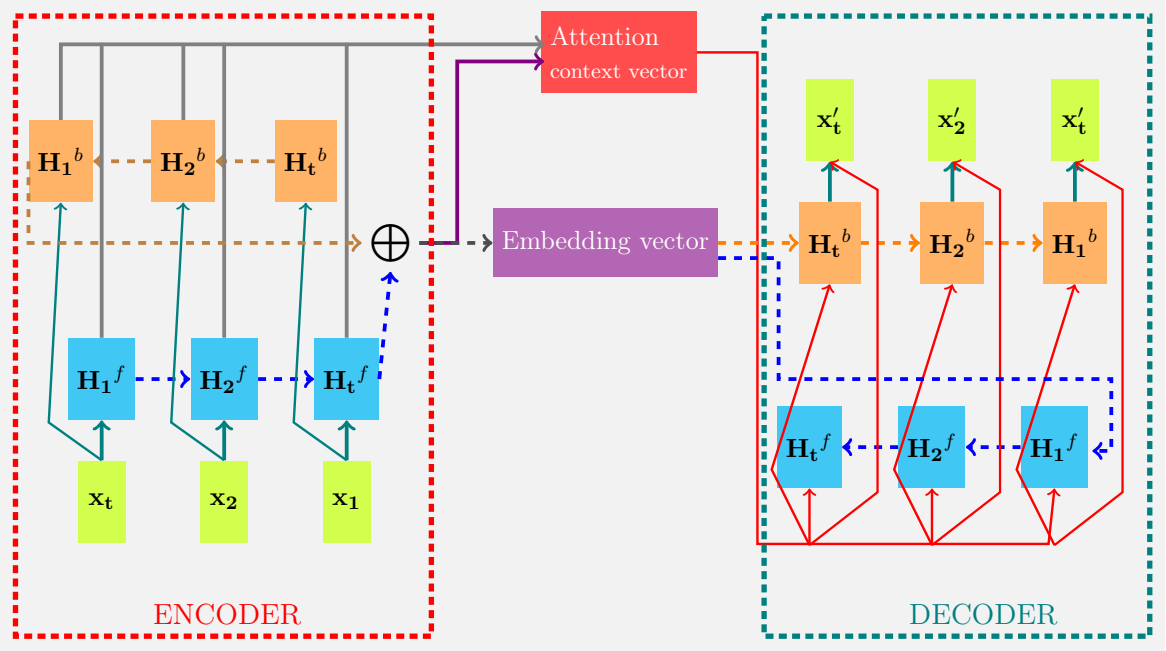


Figura 4.6: Architettura *autoencoder* Bi-LSTM

Strutturalmente, tale tipologia prevede che, sia l'*encoder* che il *decoder*, siano entrambi costituiti da una rete LSTM bidirezionale [44]: una rete neurale composta da due layer LSTM, denominati rispettivamente *forward* e *backward* a seconda della direzione in cui la sequenza temporale in ingresso viene processata. Il layer LSTM *forward* riceve e processa la sequenza in ordine positivo (da $t = 1$ a $t = T$), mentre il layer LSTM *backward* riceve la sequenza in ordine inverso (da $t = T$ a $t = 1$). Al termine del processo di codifica, l'*embedding vector* finale, che corrisponde allo stato interno finale, si ottiene concatenando il lo stato interno

finale del layer *forward* con lo stato interno finale del layer *backward*:

$$\mathbf{z} = \mathbf{H_T}^e = \mathbf{H_T^f}^e \oplus \mathbf{H_T^b} \tag{4.20}$$

Nel processo di decodifica è invece prevista l'inizializzazione del decoder, impostando $\mathbf{H_F^f}$ come stato interno iniziale per il layer *forward* e dualmente, $\mathbf{H_T^b}$ come stato interno iniziale per il layer *backward*.

Nel presente lavoro di Tesi, è stato integrato nell'*autoencoder* un meccanismo di *attention*. Come più volte riportato, nell'architettura encoder-decoder, l'intera sequenza di informazioni in ingresso deve essere catturata da una singola rappresentazione. Questo pone problemi nel trattenere le informazioni che si presentano all'inizio della sequenza e quindi nella codifica delle dipendenze a lungo raggio. A questo proposito, l'idea centrale del meccanismo di *attention* [45] è fornire, nella fase di costruzione della rappresentazione, una diversa importanza agli istanti temporali della sequenza di input, fornendo un maggior peso a quegli istanti che risultano più rilevanti al fine di una rappresentazione più significativa in termini di prestazioni. Da un punto di vista implementativo, questo si traduce nella derivazione di un *context vector* $\mathbf{c}$, ovvero un vettore che tiene conto di tutti gli istanti temporali e dei loro rispettivi contributi. Nella costruzione del *context vector* si prendono in cosiderazione tutti gli stati interni dell'encoder $\mathbf{H}_i$ e si calcola un vettore di pesi **attention**, di dimensione pari al numero di istanti temporali, confrontando lo stato interno finale $\mathbf{H}_T$ con ciascuno stato interno precedente $\mathbf{H}_s$:

$$\mathbf{attention}^{(1\times T)} = \text{align}(\mathbf{H}_s^{(1\times 2h)}, \mathbf{H}_T^{(1\times 2h)}) \tag{4.21}$$

$$= \frac{\exp(\text{score}(\mathbf{H_T}, \mathbf{H}_{s'}))}{\sum_{s'} \exp(\text{score}(\mathbf{H}_T^{(1\times 2h)}, \mathbf{H}_{s'}^{(1\times 2h)})} \tag{4.22}$$

$$\text{score}(\mathbf{H}_T^{(1\times 2h)}, \mathbf{H}_{s'}^{(1\times 2h)}) = (\mathbf{H}_T^{(1\times 2h)})^T \mathbf{H}_{s'}^{(1\times 2h)} \tag{4.23}$$

$$\tag{4.24}$$

Moltiplicando il vettore **attention** dei pesi con la sequenza degli stati nascosti si ottiene il

context vector:

$$\mathbf{c}^{(1\times 2h)} = \mathbf{a}^{(1\times T)} \times \mathbf{H}^{(T\times 2h)} \tag{4.25}$$

$$= \sum_s \mathbf{a}_s^{(1\times 1)} \mathbf{H}_s^{(1\times 2h)} \tag{4.26}$$

In altre parole, essendo il *context vector* una media pesata degli stati interni codificati, esso fornisce una rappresentazione maggiormente significativa della sequenza d'input, dando maggior peso e risalto a quegli istanti temporali maggiormente correlati. Nel processo di decodifica, il *context vector* viene ripetuto T volte e la sequenza così ottenuta viene fornita in ingresso al decoder:

$$\mathbf{H}_T^{(1\times 2h)} \xrightarrow{repeat} \mathbf{H}_T^{(T\times 2h)} \tag{4.27}$$

$$\mathbf{x}'^{(1\times T)} = f_{decoder}(\mathbf{c}^{(1\times 2h)}, \mathbf{H}_T^b{}^{(1\times 2h)}, \mathbf{H}_T^b{}^{(1\times 2h)}) \tag{4.28}$$

4.3 Regressione Lineare

L'analisi di regressione lineare è una tecnica che permette di analizzare la relazione lineare tra una variabile dipendente (o variabile di risposta) e una o più variabili indipendenti (o predittori), basandosi sull'ipotesi dell'esistenza di una relazione di tipo causa-effetto.
Lo studio di questa relazione può avere un duplice scopo:

- esplicativo: comprendere e ponderare gli effetti delle variabili indipendenti sulla variabile dipendente in funzione di un determinato modello teorico;
- predittivo: individuare una combinazione lineare di variabili indipendenti per predire in modo ottimale il valore assunto dalla variabile dipendente.

Date p variabili indipendenti $(X_1, X_2, X_3, ..., X_p)$ e Y la variabile dipendente, e assumendo la relazione $Y = f(X_1, X_2, ..., X_p)$, l'obbiettivo dell'analisi è la definizione di un modello di regressione lineare $(\hat{Y})$, descritto nella Relazione:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + ... + \beta_p X_1 p + \epsilon = \hat{Y} + \epsilon \tag{4.29}$$

Dove:

- β_0 è l'intercetta e rappresenta il valore atteso della $\hat{Y}$ quando tutte le variabili esplicative sono pari a 0;

- $\beta_1, \beta_2, \beta_3, .., \beta_p$ sono i coefficienti di regressione. Il coefficiente di regressione incognito β_j, con $1 \leq j \leq p$, rappresenta il legame di Y rispetto ad X_j, tenendo costanti tutte le altre variabili. In altre parole l'inclinazione β_j spiega come varia $\hat{Y}$ in corrispondenza di una variazione unitaria della variabile X_j. La determinazione del modello si basa sulla determinazione di tali coefficienti;
- ϵ è l'errore che si commette nella spiegazione della variabile Y tramite una funzione lineare delle variabili indipendenti.

Fornito un campione di n osservazioni della variabile Y e delle variabili esplicative X_j si ha:

- $\mathbf{y}$: il vettore colonna di n elementi relativo alle osservazioni sulla variabile dipendente;
- $\boldsymbol{\beta}$: il vettore colonna di $p+1$ elementi relativo ai coefficenti di regressione;
- $\boldsymbol{\epsilon}$: il vettore colonna di n elementi dei termini di errore;
- $\mathbf{X}$: la matrice di dimensioni $n \times (p+1)$, la cui prima colonna è un un vettore colonna di elementi pari ad 1 (corrispondente all'intercetta), mentre le restanti p colonne sono altrettanti vettori colonna, ciascuno relativo alle n osservazioni sulla corrispondente variabile esplicativa, con $n > p+1$;

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix} \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_i \\ \vdots \\ \beta_p \end{pmatrix} \boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_i \\ \vdots \\ \epsilon_n \end{pmatrix}$$
$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1i} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2i} & \dots & x_{2p} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_{i1} & x_{i2} & \dots & x_{ii} & \dots & x_{ip} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n1} & x_{n2} & \dots & x_{ni} & \dots & x_{np} \end{pmatrix} \tag{4.30}$$

Sulle base delle n osservazioni, la relazione 4.29 può essere espressa in termini più compatti ricorrendo alla notazione matriciale:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon \tag{4.31}$$

che (a parte la presenza di un termine di errore) rappresenta la forma matriciale di un sistema di equazioni lineari con n equazioni e p incognite in cui l'i-esima equazione può essere scritta come:

$$\mathbf{y_i} = \mathbf{x_i'}\boldsymbol{\beta} + \boldsymbol{\epsilon_i} \tag{4.32}$$

Dove $\mathbf{x_i'} = (1, x_{i1}, x_{i2}, ..., x_{ip})$ rappresenta l'i-esima riga di $\mathbf{X}$, mentre $\boldsymbol{\epsilon_i}$ è determinabile come $\boldsymbol{\epsilon_i} = y_i - \mathbf{x_i'}\boldsymbol{\beta}$ se e solo se si conoscono i coefficienti di regressione. Una volta calcolate le stime $\hat{\beta}_j$ dei coefficienti β_j, la singola osservazione y_i può essere scritta come:

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + ... + \hat{\beta}_p x_{ip} + \hat{e}_i = \hat{y}_i + \hat{e}_i \tag{4.33}$$

Dove i residui $\hat{e}_i$ sono ottenuti come differenza tra il valore osservato y_i e il valore stimato $\hat{y}_i$.

La stima dei coefficienti di regressione $\hat{\beta}_j$ si basa sul *Metodo dei minimi quadrati*, il quale prevede la minimizzazione dello scarto quadrato tra le risposte osservate e quelle teoriche stimate. Formalmente, date le n osservazioni, il *Metodo dei minimi quadrati* prevede la minimizzazione della seguente quantità:

$$RSS = \sum_{i=1}^{n} (\hat{e}_i)^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{4.34}$$

Infine, geometricamente l'Equazione:

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + ... + \hat{\beta}_p X_p \tag{4.35}$$

con $\hat{\beta}_j$ i valori stimati dei coefficienti di regressione, definisce l'equazione di un iperpiano nello spazio a $p+1$ dimensioni, il quale tra gli infiniti piani, è quello che rende minima la somma dei quadrati delle lunghezze dei segmenti congiungenti i punti osservati al piano stesso, ovvero rende minima la somma delle distanze verticali tra i punti osservati e l'iperpiano stesso.
L'iperpiano si riduce a un piano di regressione se le variabili indipendenti sono pari a due X_1, X_2, e a una retta di regressione quando si ha un'unica variabile esplicativa X_1. In quest'ultimo caso si parla di analisi di regressione semplice.

4.4 Selezione features - Stepwise regression

Nel framework presentato, il modello di regressione lineare opera su vettori di *features*, estratte dai dati temporali multi-sensore, al fine di prevedere e determinare il rispettivo indice HI. A questo proposito, le *features* estratte hanno un'importante peso e ricaduta sull'affidabilità dei risultati del modello. In generale infatti, nell'implementazione di un modello di *machine learning*, dopo la definizione preliminare di un set di *features*, si applica ad esso un meccanismo di selezione al fine di ottenere un sottoinsieme ottimo.
L'integrazione di tale meccanismo è dovuta a differenti motivi:

1. Riducendo il numero di *features* considerate, si riduce la dimensionalità dei dati in ingresso, ottenendo così una riduzione dei tempi di training e del costo computazionale in termini di CPU e memoria;

2. Un eccessivo numero di *features* può influire negativamente sulle prestazioni del modello, ad esempio in termini di *overfitting*, essendo alcune di esse ridondanti o non correlate con il processo;

3. Selezionando solamente le *features* più significative si agevola la semplicità del modello e la relativa comprensione da parte di utenti e ricercatori.

Esistono una moltitudine di strategie nell'ambito della *features selection* relativa a un modello di regressione lineare. Nel caso in esame è stato utilizzato il metodo *stepwise regression*: un metodo di selezione semi-automatico e risultato della combinazione di due tecniche di selezione elementari, *forward selection* e *backward selection*. Esso prevede inoltre l'utilizzo del test d'ipotesi statistico e del *p-value* per il calcolo della significatività delle *features*. A questo proposito, precedentemente all'illustrazione del metodo, viene fornita una spiegazione delle procedure componenti evidenziandone gli aspetti teorici e implementativi.

4.4.1 Test d'ipotesi e *p-value*

Il test delle ipotesi consente di verificare se, e in quale misura, una determinata ipotesi (di carattere sociale, biologico, medico, economico, ecc.) è supportata dall'evidenza empirica. Il fenomeno studiato deve essere rappresentabile mediante una distribuzione di probabilità e l'ipotesi sulle caratteristiche del fenomeno studiato è tradotta in ipotesi su uno o più parametri della distribuzione.
In fase preliminare, il test prevede la formulazione due ipotesi:

- **Ipotesi nulla $\mathbf{H_0}$**: afferma che non esiste alcuna relazione tra il fenomeno misurato (la variabile dipendente) e la variabile indipendente, per cui qualsiasi variazione osservata nel fenomeno sarebbe dovuta a errori di campionamento (possibilità casuale) o errori sperimentali;

- **Ipotesi alternativa $\mathbf{H_1}$**: una qualunque ipotesi opposta all'ipotesi nulla, che quindi elimina il concetto di casualità prevedendo una relazione tra il fenomeno misurato e le variabili indipendenti (esplicative).

Successivamente si definisce un'opportuna statistica-test t: una quantità che calcolata sui dati osservati permette di sintetizzare l'informazione portata dal campione; e sulla base di questa si determina la relativa la distribuzione campionaria ottenuta immaginando di ripetere il test infinite volte. La statistica test assumerà infatti valori diversi descrivendo una propria distribuzione. In aggiunta si prefissa anche un livello di significatività del test.
Il livello di significatività è una soglia che determina se un risultato di uno studio, a fronte dei test statistici svolti, possa essere considerato statisticamente significativo o meno, e rappresenta la probabilità di rigettare l'ipotesi nulla quando questa è vera. Per citare un esempio: un livello di significatività pari allo $0,05$ indica un rischio del 5% nel concludere che esiste una qualche relazione tra variabile dipendente e indipendente quando in realtà non vi è alcuna relazione.

Una volta determinate le quantità sopra citate e a partire da x, un campione di osservazioni, si calcola il valore osservato $t_0 = t(x)$ ovvero il valore assunto dalla statistica-test in corrispondenza del campione. A questo punto, sulla base di t_0 l'accettazione o meno dell'ipotesi H_0 è determinata dal parametro *p-value*.
Conoscendo $f(t|H0)$, ossia la distribuzione campionaria di t sotto l'ipotesi H_0, il *p-value* [46] è una probabilità condizionata che rappresenta la probabilità di ottenere un risultato uguale o più estremo rispetto a quello osservato empiricamente.
Geometricamente (Figura 4.7) è dunque l'area sottostante la curva della funzione di densità condizionata $f(t|H0)$ compresa tra $t = t_0$ e $t = \infty$:

$$p\text{-}value = Prob(t > t_0|H_0) = \int_{t_0}^{\infty} f(t|H_0)dt \tag{4.36}$$

Tale parametro non è altro che una misura dell'evidenza contraria all'ipotesi nulla e tanto minore esso è, tanto maggiore è il livello di confidenza nel rigettare l'ipotesi nulla e quindi nel test. In altre parole, supponendo H_0 vera, la distribuzione campionaria relativa alla statistica-

test mostra quali valori ci si aspetterebbe di ottenere da quest'ultima. Se t_0, il valore osservato sul campione, cade in una delle code della distribuzione campionaria è dunque lontano da quanto atteso sotto H_0, per cui se H_0 fosse vera un simile valore sarebbe poco usuale.

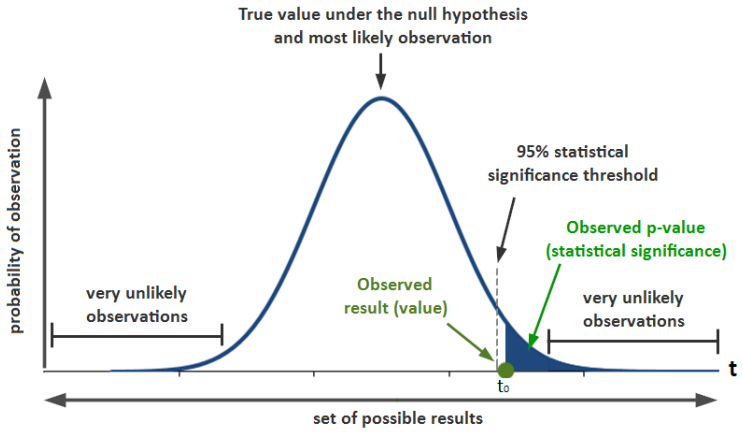


Figura 4.7: Esempio di distribuzione campionaria della statistica-test. Da `https://www.simplypsychology.org/p-value.html`

Infine, calcolato il *p-value* e fissato il livello di significatività α, l'accettazione o meno dell'ipotesi nulla H_0 è dettata dalle seguenti disuguaglianze:

- Se $p < \alpha$ concludiamo che il test ha un risultato significativo e l'ipotesi nulla H_0 viene rigettata.
- Se $p > \alpha$ l'ipotesi nulla H_0 viene accettata.

Come verrà descritto in seguito e maggiormente nel dettaglio, nella procedura *stepwise* per ciascun predittore X_j si calcola il rispettivo *p-value*. In questo calcolo, l'ipotesi nulla H_0 coincide con l'ipotesi secondo cui non esista alcuna relazione tra la variabile dipendente Y e il predittore considerato X_j, ovvero quest'ultimo è scorrelato con il processo in esame. A questo proposito una volta applicato il test e calcolato *p-value*, un basso valore di quest'ultimo comporta una forte confidenza nel rifiuto dell'ipotesi nulla e quindi una forte confidenza nel considerare significativo il predittore X_j nella stima di Y.

4.4.2 Forward e Backward Selection

Entrambi i metodi, *forward* e *backward*, prevedono in via preliminare la definizione di un livello di significatività α_{in} nel caso *forward*, (α_{out} nel caso *backward*) inteso come il massimo *p-value* (minimo *p-value*) di un predittore affinché esso possa essere aggiunto (mantenuto) nel modello di regressione. Definito un set iniziale di n features $X = \{X_1, X_2, ..., X_n\}$, il metodo

forward inizia con un modello "nullo", ovvero contenente solo l'intercetta e nessuna variabile selezionata tra i predittori. Nel primo step, relativamente al modello nullo, si calcola il *p-value* di ciascun predittore escluso e viene aggiunto al modello il predittore con l'associazione maggiormente significativa sul piano statistico (minimo *p-value*), tuttavia, se e solo se si verifica $p\text{-}value_{min} < \alpha_{in}$. Successivamente, ad ogni step, e sulla base del modello corrente, vengono ogni volta calcolati i *p-value* relativi ai predittori esclusi e tra questi si aggiunge quello con la maggiore associazione statisticamente significativa. Il processo prosegue sino a quando non vi è più alcuna variabile esclusa che presenti $p\text{-}value < \alpha_{in}$; in altre parole i predittori non integrati nel modello non risultano essere significativi sulla base della soglia fissata. Il metodo backward, procede invece in maniera speculare. In primo luogo viene anche qui fissato un livello α_{out}, coincidente con il valore massimo che il *p-value* di un predittore può assumere affinché quest'ultimo rimanga incluso nel modello, e l'inizializzazione con un modello iniziale che include tutte le variabili indipendenti a disposizione. Ad ogni step, si procede al calcolo del *p-value* per ciascun predittore inserito nel modello e si procede alla selezione del predittore con l'associazione statistica meno significativa (massimo *p-value*). Se il relativo *p-value* supera la soglia α_{out}, si procede alla rimozione del predittore dal modello, senza la possibilità futura di essere reinserito. La rimozione è dovuto al fatto che essendo $p\text{-}value > \alpha_{out}$ il predittore non risulta significativo. Il procedimento si arresta nel momento in cui non ci sono più predittori che presentano $p\text{-}value > \alpha_{out}$, ovvero tutti i predittori inseriti nel modello sono significativi sul piano statistico.

4.4.3 Stepwise regression

Il processo *stepwise* opera avanti e indietro tra i due processi, aggiungendo e rimuovendo le variabili che, nei vari aggiustamenti del modello (con aggiunta o re-inserimento di una variabile) guadagnano o perdono in termini di significatività. In questo modo una variabile introdotta precedentemente può risultare ridondante in virtù dell'entrata di nuovi predittori, infatti ad ogni step tutte le variabili esplicative considerate precedentemente nel modello sono testate di nuovo. In Figura 4.8 viene riportato il diagramma di flusso dell'algoritmo *stepwise*. Si osserva come l'algoritmo preveda l'inizializzazione con un modello nullo, ed ad ogni step vengano in una prima fase testati i predittori esclusi (*forward selection*) e in una seconda fase vengono testati i predittori interni al modello. L'algoritmo termina quando in un determinato step nessun predittore tra quelli esclusi viene aggiunto al modello, e nessuno tra quelli inclusi viene rimosso. In altre parole, tutti i predittori inclusi sono significativi sul piano statistico e tutti quelli esclusi non lo sono.

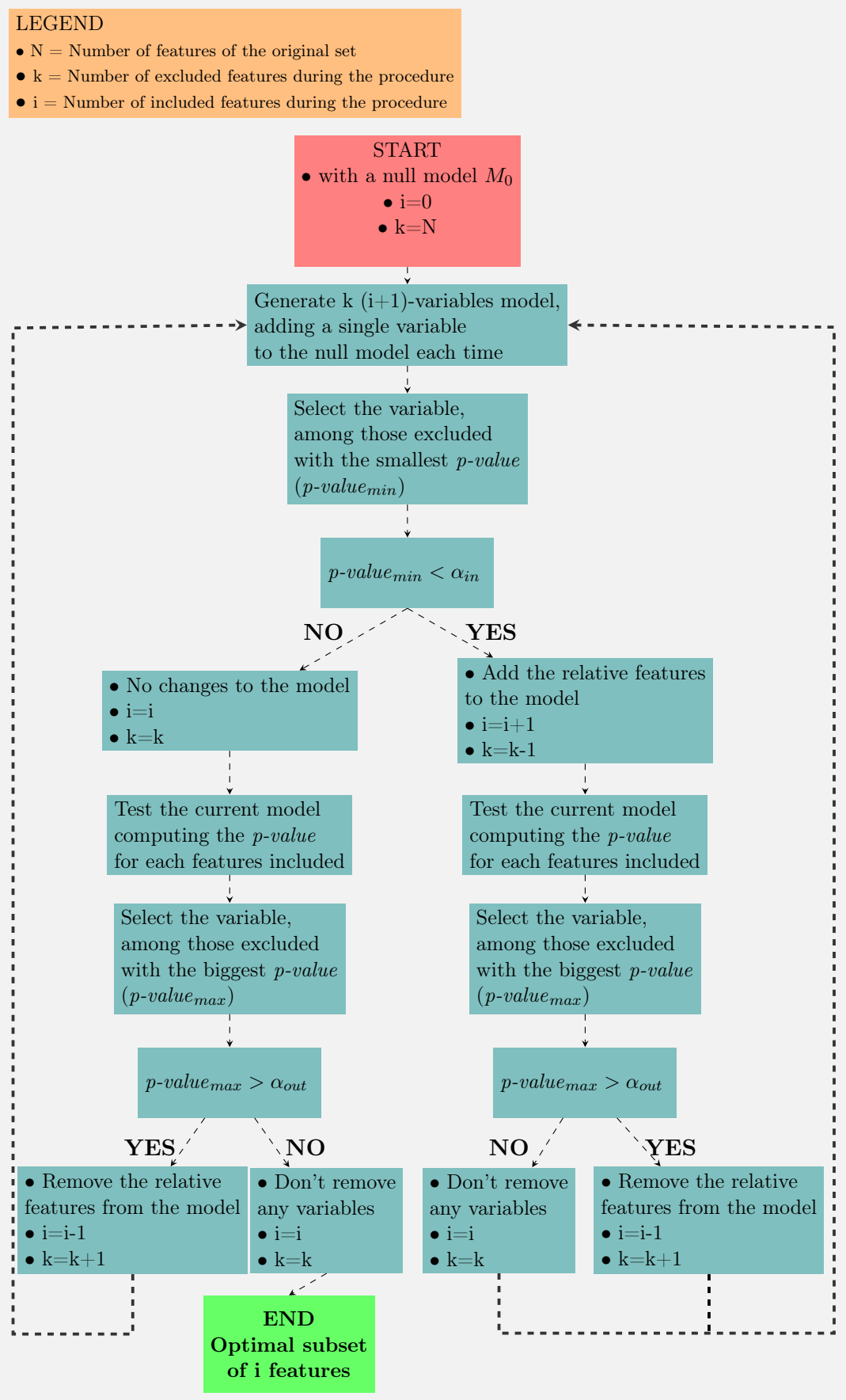


Figura 4.8: Algoritmo *stepwise regression*

Capitolo 5

Sperimentazione

Sono stati implementati due modelli prognostici distinti per i due materiali, entrambi basati sulla metodologia sopra illustrata. La giustificazione alla base di tale distinzione è dettata dal fatto che la durata media del ciclo-vita per i due materiali, sia in termini di numero di corse che in termini di durata temporale, differisce consistentemente. Inoltre a causa del numero ridotto di corse presenti nel dataset complessivo, pari a 166[1], è stata utilizzata la tecnica *leave-one-out cross validation* sia per la fase di training che di test. In questo modo, nella definizione del sistema basato su *cast iron (steel)*, su un totale di 8 (7) Casi, iterativamente è stato selezionato un Caso per il test e i restanti 7 (6) sono stati impiegati per il training del modello di predizione.
Per quanto riguarda invece le metriche utilizzate, in accordo con la letteratura in merito, le prestazioni dei modelli implementati sono state valutate in termini di RMSE (*Root Mean Squared Error*) e MAPE (*Mean Absolute Percentage Error*). Considerando per il Caso di test, l'insieme delle N coppie $\{(\hat{RUL}_i, RUL_i)\}$ con $\hat{RUL}_i$ i-esima RUL stimata e RUL_i relativa RUL target, le metriche vengono calcolate come segue:

- L'errore quadratico medio (RMSE) è la media dei quadrati degli errori di previsione (differenza tra valore effettivo e valore stimato) ed è quindi una misura di errore assoluta. Elevando infatti le singole deviazioni al quadrato, si evita che valori positivi e negativi possano annullarsi l'uno con l'altro. RMSE è dunque una misura dello scostamento medio tra valori stimati ed effettivi, concludendo quindi come minore sia il suo valore, minore è l'errore medio di previsione e quindi maggiore sono le prestazioni del sistema in esame.

[1] Il Caso 6 non è stato considerato nella sperimentazione comprendendo un'unica corsa.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{RUL}_i - RUL_i)^2} \tag{5.1}$$

- L'errore percentuale medio assoluto (MAPE) è la media degli errori percentuali assoluti dei valori stimati. Ciascun singolo errore percentuale quantifica l'entità dell'errore di previsione rapportato al valore effettivo. Essendo inoltre tali errori, calcolati in termini assoluti, come nel caso RMSE, si viene ad evitare il problema degli errori positivi e negativi che potrebbero annullarsi a vicenda. In virtù della sua definizione, minore è il MAPE e migliori sono le prestazioni del sistema in esame.

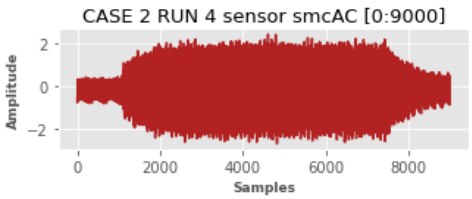
$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\frac{|\hat{RUL}_i - RUL_i|}{RUL_i} \tag{5.2}$$

La sperimentazione è stata svolta nell'ambiente Python utilizzando la piattaforma Google Colab. Ad oggi infatti, nell'ambito del *deep learning* e intelligenza artificiale, Python si rivela uno dei linguaggi di programmazione più performanti e al tempo stesso di facile comprensione, presentando una vasta gamma di librerie dedicate come ad esempio TensorFlow, Keras, Scikit-learn e Numpy.

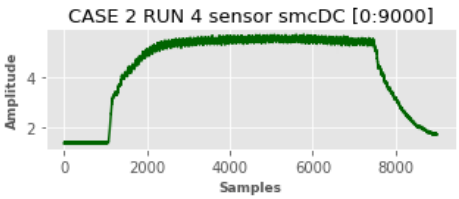
Nelle Sottosezioni successive viene presentata e descritta la fase di sperimentazione, fornendo particolare attenzione alle scelte implementative perseguite e sulla base di determinate considerazioni svolte in merito al dataset e agli algoritmi integrati.

5.1 Analisi dei dati e pre-processing

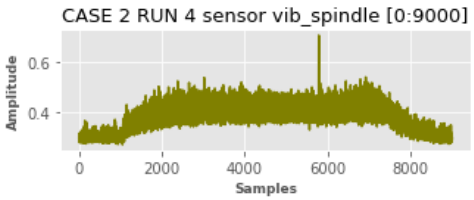
Il primo step della procedura ha comportato un'analisi preliminare dei dati volta allo studio di essi e alla conseguente valutazione delle possibili tecniche di pre-processing applicabili.
Come affermato in precedenza, ciascuna corsa $\mathbf{X_j}$ è una serie temporale multi-variata di 9000 campioni ciascuno dei quali è un vettore colonna $\mathbf{x_t} \in \mathbb{R}^{1\times 6}$. Con in mente l'obbiettivo di fornire un'idea della tipologia dei dati raccolti dal dataset, in Figura 5.1 e 5.2 vengono mostrati gli andamenti di due corse ($\mathbf{X_4}^{(2)}$, corsa 4 relativa al Caso 2 e $\mathbf{X_6}^{(8)}$, corsa 6 relativa al Caso 8) riportando in particolar modo le serie uni-variate, ciascuna corrispondente ad un determinato sensore.



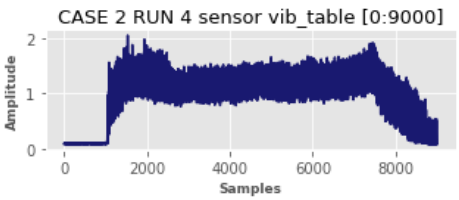
(a) Corrente alternata del motore



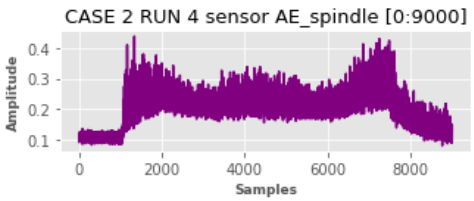
(b) Corrente continua del motore



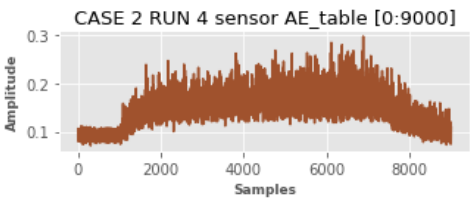
(c) Vibrazioni del mandrino



(d) Vibrazioni del tavolo

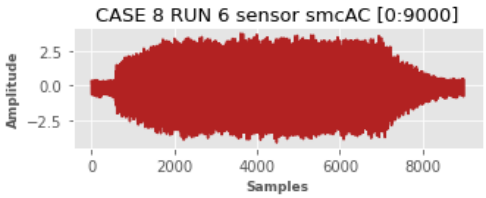


(e) Emissioni acustiche del mandrino

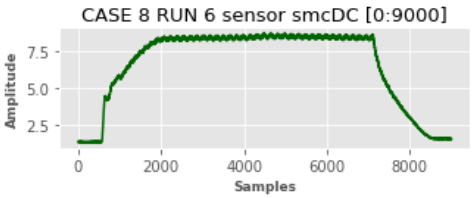


(f) Emissioni acustiche del tavolo

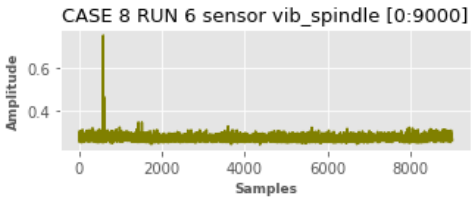
Figura 5.1: Andamento temporale dei dati raccolti dai sensori per la corsa 4 del Caso 2



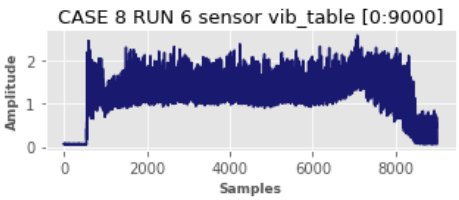
(a) Corrente alternata del motore



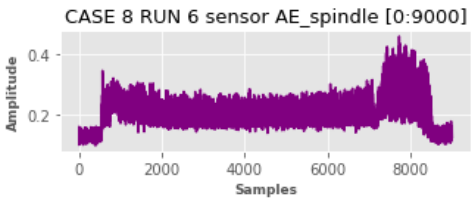
(b) Corrente continua del motore



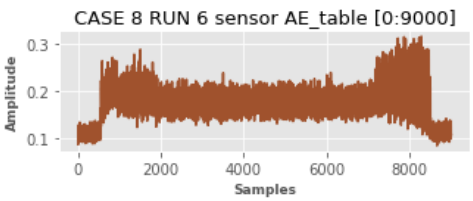
(c) Emissioni acustiche del mandrino



(d) Emissioni acustiche del tavolo



(e) Emissioni acustiche del mandrino



(f) Emissioni acustiche del tavolo

Figura 5.2: Andamento temporale dei dati raccolti dai sensori per la corsa 6 del Caso 8

Osservando le Figure 5.1 e 5.2 e tenendo conto della letteratura in merito al dataset, è possibile

notare come l'andamento temporale di ciascuna corsa sia suddivisibile in tre principali stati disgiunti e continui:

- Stato iniziale che comprende i campioni $[0, 3000]$ e in cui è presente tendenzialmente un andamento crescente dovuto all'inizio della lavorazione;
- Stato stabile e centrale relativo ai campioni $[3000, 6000]$ in cui il sistema è in lavorazione;
- Stato finale comprendente i campioni $[6000, 9000]$ e corrispondente al termine della lavorazione, si osserva infatti un trend temporale decrescente;

In virtù di tali considerazioni, e in conformità con altri lavori proposti, per ciascuna corsa sono stati considerati solamente i 3000 campioni relativi alla regione centrale di lavoro.

Data-cleaning

La prima operazione svolta nell'ambito del *pre-processing* è stata il *data cleaning*, ovvero il processo di identificazione e correzione, o rimozione, di dati incompleti, impropri e imprecisi. L'obiettivo è infatti quello di occuparsi e risolvere i cosiddetti problemi di qualità dei dati che influiscono negativamente sulle prestazioni del modello finale e compromettono il processo di analisi dei risultati. Esistono differenti tipologie di problemi legati alla qualità dei dati come ad esempio valori mancanti, dati duplicati, dati invalidi e dati incoerenti, spesso definiti *outliers*. Le tecniche di *data-cleaning* più comunemente utilizzate prevedono la rimozione di record di dati con valori mancanti, l'unione di record duplicati, la generazione di una stima migliore o al massimo ragionevole per i valori non validi. Dopo aver ristretto la dimensionalità di ciascuna corsa a 3000 campioni, e tenendo conto della struttura del dataset, specifiche tecniche di *data-cleaning* sono state applicate rispettivamente a: 1) records raccolti dai sensori e 2) parametro VB.

Partendo dai dati raccolti dai sensori, il problema di qualità relativo ad essi è dovuto alla presenza di *spikes* i quali rientrano della categoria degli outliers, ovvero valori che si discostano fortemente dagli altri punti del campione e tali scostamenti non sono spiegati dai normali processi del sistema; la loro rimozione dal campione è pertanto necessaria per evitare un effetto di distorsione nelle successive analisi. Una tecnica comunemente impiegata nell'identificazione degli outliers è la tecnica "valore medio più o meno tre deviazioni standard" ($mean \pm 3\sigma$) la quale prevede come definizione di outliers i valori esterni a tali intervallo. Assumendo infatti una distribuzione normale dei campioni, nell'intervallo $[mean - 3\sigma, mean + 3\sigma]$ ricade il 99.87% dei campioni. In questo modo gli outliers vanno a rappresentare lo 0.13% della

totalità dei campioni. Tale percentuale, in taluni casi può rappresentare un'identificazione degli outliers non troppo conservativa, per cui sono stati proposti intervalli maggiormente ristretti come ad esempio $[mean - 2.5\sigma, mean + 2.5\sigma]$, oppure $[mean - 2\sigma, mean + 2\sigma]$. Ad ogni modo la scelta dell'intervallo dipende dalla specifica situazione considerata e anche dalla tipologia dei dati.
Nonostante però tale tecnica venga comunemente implementata, in [47] si osserva come l'uso del valore medio e della deviazione standard come indicatori può condurre a tre problemi principali:

1. Si assume una distribuzione normale dei dati che quindi spesso differisce dalla distribuzione reale;

2. Il valore medio e la deviazione standard sono fortemente influenzati dal valore degli outliers;

3. Incapacità nell'identificazione degli outliers nel caso di dataset con dimensioni limitate.

Dunque entrambi gli indicatori risultano problematici poiché essi vengono utilizzati per definire l'identificazione degli outliers ma allo stesso tempo essi stessi sono influenzati dalla presenza degli outliers. Sempre in [47] viene quindi proposta una tecnica alternativa, ed integrata nel framework presente, la quale prevede la mediana e la deviazione mediana assoluta come indicatori per la definizione degli outliers. La mediana, come il valore medio, è una misura della tendenza centrale ma a differenza di esso è insensibile alla presenza di outliers. Ad esempio quando una sequenza di osservazioni presenta una singola osservazione con un valore pari ad infinito, il valore medio delle osservazioni diviene infinito mentre la mediana risulta invariata. E' facile dimostrare che la mediana diventi infinito solo laddove la percentuale di osservazioni con valori pari ad infinito raggiunge o supera il 50%.
Il calcolo della mediana $M(\cdot)$ di una sequenza di osservazioni $\mathbf{x} = \{x_1, x_2, ..., x_N\}$ prevede l'ordinamento in ordine crescente della sequenza $\mathbf{x}$ e la mediana coincide con il valore centrale, in caso di lunghezza della sequenza dispari, o media dei due valori centrali in caso di sequenza pari. La deviazione mediana assoluta (MAD) è invece definita come:

$$MAD = M(|x_i - M(\mathbf{x})|) \tag{5.3}$$

Ovvero il MAD è la mediana delle deviazioni assolute delle osservazioni dalla loro rispettiva mediana.
Definiti i due indicatori, la tecnica di identificazione degli outliers prevede che un campione

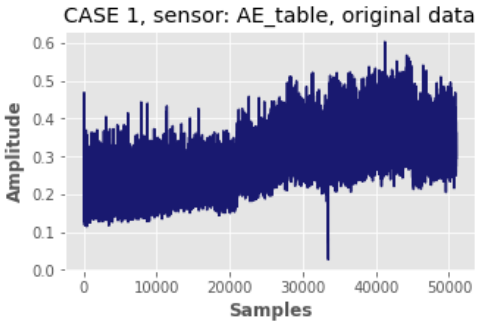
x_i è definito outlier nel momento in cui non ricade nel range $[M - 3MAD, M + 3MAD]$ o dualmente nel momento in cui è verificata la seguente condizione:

$$\frac{x_i - M}{MAD} \geq |\pm 3| \tag{5.4}$$

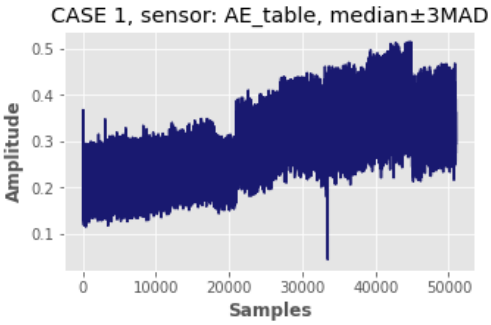
Integrando tale tecnica nel framework ed applicandola a ciascuna corsa sono stati identificati: a) 654 outliers per il materiale *cast iron*, pari al 0.033% dei campioni totali, e b) 348 outliers per il materiale *steel*, pari al 0.034% dei campioni totali. Una volta identificati si è proceduto alla loro sostituzione tramite interpolazione. In particolar modo, supponendo x_j un outlier all'instante temporale j, il valore sostituito $x_j^{(new)}$ è stato calcolato come:

$$x_j^{(new)} = \frac{x_{j-1} + x_{j+1}}{2} \tag{5.5}$$

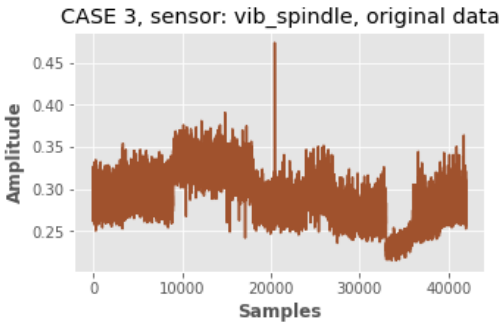
Ovvero come interpolazione dei due valori adiacenti. Questo avviene anche nel caso in cui entrambi o solamente uno dei valori adiacenti x_{j-1} risultino outliers. La giustificazione alla base di tale scelta, è dettata dal fatto che se lo scostamento anomalo dall'andamento della corsa si protrae per multipli istanti continui verosimilmente esso è dovuto al processo del sistema in esame e non ad errori di acquisizione e campionamento. In Figura 5.3 vengono riportati alcuni esempi di risultati ottenuti applicando la tecnica ($M \pm 3MAD$).



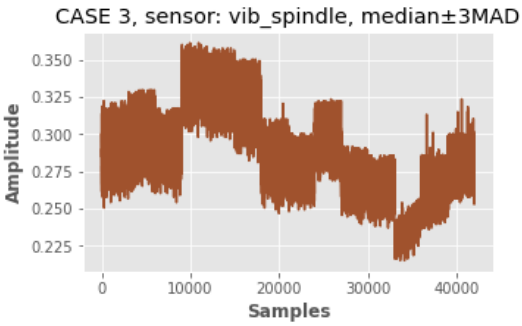
(a) Dati originari Caso 1



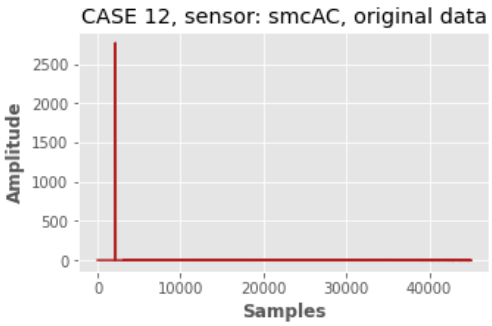
(b) Risultati *data cleaning* Caso 1



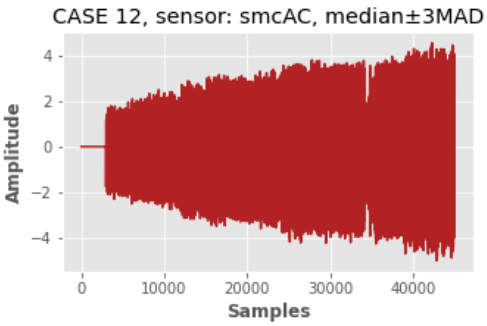
(c) Dati originari Caso 3



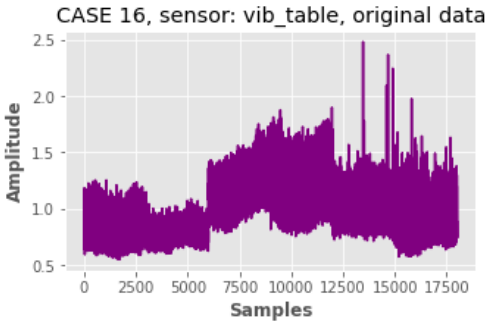
(d) Risultati *data cleaning* Caso 3



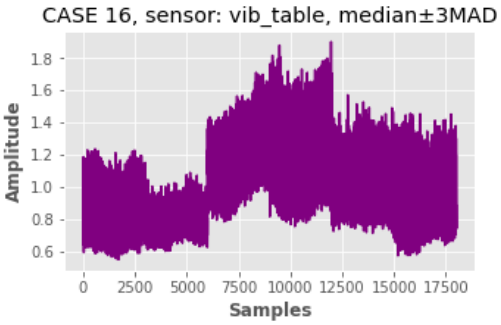
(e) Dati originari Caso 12



(f) Risultati *data cleaning* Caso 12



(g) Dati originari Caso 16



(h) Risultati *data cleaning* Caso 16

Figura 5.3: Esempi di applicazione della tecnica di *data-cleaning* "mediana±3MAD" considerando in tutte le Figure la successione delle Corse relative al Caso e sensore specificato e riportando a destra le sequenze temporali originali, e a sinistra le sequenze temporali ottenute in seguito all'applicazione della tecnica.

Nel Caso 3 e 16, si osserva come i singoli *spikes* siano stati ridotti in ampiezza, mentre nel Caso 1 il picco presente permane anche in seguito al *data cleaning*; portando a concludere come esso interessi multipli istanti temporali continui. Nel Caso 12 invece, nei dati originali si osserva un picco iniziale con un ampiezza > 2500 dovuto esplicitamente ad errori di acquisizione e memorizzazione, e limitato ad un singolo istante temporale. Infatti, in seguito al *data-cleaning* il picco viene eliminato del tutto.
Come descritto precedentemente nella Sezione 3, il parametro *flank wear* relativo alla corsa in esame è stato misurato al termine di essa, ma non necessariamente dopo ciascuna corsa. In questo caso quindi, nel rispettivo record del dataset è presente un valore mancante indicato con NaN. Si è proceduto quindi alla sostituzione di tali valori mancanti, usando la tecnica dell'interpolazione lineare, ottenendo così per ogni corsa il rispetto valore VB.
Infine, per ciascun Caso la RUL è stata settata pari a 0 quando il parametro VB ha raggiunto per la prima volta il valore $0.45mm$, in accordo con la letteratura in merito.

5.2 Training Autoencoder

Come descritto nella Sezione 4 l'autoencoder è una rete neurale artificiale addestrata in modalità non supervisionata con lo scopo di fornire, nella fase di utilizzo, una rappresentazione compressa e significativa, e denominata *embedding vector*, dei dati d'input.
Nella procedura qui considerata, l'autoencoder lavora sulle singole corse, sequenze temporali multi-variate, e una volta addestrato, estrae per ciascuna di esse il rispettivo *embedding vector*.
Da un punto di vista implementativo, prima di essere fornite in input all'*autoencoder*, alle singole corse $\mathbf{X_i} = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{3000}}\}$ viene applicato un sotto-campionamento tramite decimazione di un fattore 30, ottenendo rispettivamente le sequenze $\mathbf{X'_i} = \{\mathbf{x'_1}, \mathbf{x'_2}, ..., \mathbf{x'_{100}}\}$ ridotte a 100 campioni con $\mathbf{x'_j} = \mathbf{x_{30j}}$.

Nella fase di training, fornita in ingresso ciascuna singola sequenza $\mathbf{X'_i}$, *encoder* e *decoder* lavorano simultaneamente al fine di minimizzare la seguente funzione errore:

$$E = \frac{1}{2}\sum_{i=1}^{t}(\|\mathbf{e_i}\|_1)^2 \tag{5.6}$$

$$\mathbf{e_i} = \mathbf{x'_i} - \mathbf{y_i} \tag{5.7}$$

Dove la sequenza $\mathbf{Y_i} = \{\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_{100}}\}$ è la sequenza ricostruita dal decoder a partire dall'*embedding vector* $\mathbf{z_i}$, a sua volta a partire dall'input $\mathbf{X'_i} = \{\mathbf{x'_1}, \mathbf{x'_2}, ..., \mathbf{x'_{100}}\}$.
Da un punto di vista implementativo, ciascuna cella LSTM ha previsto un numero di *hidden nodes* pari a 100; mentre il training della rete è stato svolto usando un batch size pari a 16 e un massimo di 200 epoche. Quest'ultimo limite superiore è dovuto al fatto che è stata integrata la modalità *Early Stopping* con una patience di 10 epoche al fine di ridurre l'overtraining e l'overfitting. Tale modalità ha previsto quindi l'arresto del training qualora la metrica d'errore non avesse presentato miglioramenti per 10 epoche successive. In aggiunta, è stato definito un meccanismo di *model selection* volto alla memorizzazione del modello che nella fase di training presentava il valore di metrica errore minore.

5.3 Costruzione HI target

Una volta addestrato, l'autoencoder lavora sulle singole corse, sequenze temporali multivariate, estrae per ciascuna di esse il rispettivo *embedding vector*. In seguito, secondo un opportuno algoritmo di costruzione, quest'ultimo viene utilizzato per generare il valore HI target corrispondente alla corsa considerata.

$$Corsa\ \mathbf{X_j} \xrightarrow{Encoder} \mathbf{z_j} \xrightarrow{HIconstruction} h_j \tag{5.8}$$

Considerando il j-esimo Caso $\mathbf{I_j} = \{\mathbf{X_1}, \mathbf{X_2}, ..., \mathbf{X_{N_j}}\}$ composto da N_j corse, fornendo in ingresso all'*autoencoder* ciascuna di esse ed estraendone il relativo *embedding vector* si arriva ad ottenere una sequenza temporale univariata $\mathbf{Z_j} = \{\mathbf{z_1}, \mathbf{z_2}, ..., \mathbf{z_{N_j}}\}$ che rappresenta l'andamento del degrado del sistema per quanto riguarda l'istanza $\mathbf{I_j}$ e sulla quale si fonda l'algoritmo di generazione dell'indice HI target.

$$\mathbf{I_j} = \{\mathbf{X_1}, .., \mathbf{X_{N_j}}\} \xrightarrow{Encoder} \mathbf{Z_j} = \{\mathbf{z_1}, .., \mathbf{z_{N_j}}\} \xrightarrow{HIconstruction} \mathbf{HI_j}^{(Tg)} = \{h_1, .., h_{N_j}\} \tag{5.9}$$

Sulla base del sistema considerato, è lecito presupporre che gli *embedding vectors* relativi alle corse iniziali, $\{\mathbf{z_1}, \mathbf{z_2}, \mathbf{z_3}\}$, forniscano una rappresentazione del sistema in una condizione di buon funzionamento, essendo essi correlati alla fase iniziale di utilizzo del sistema. A questo proposito, tali vettori vanno a costituire un set di *normal embedding vectors* indicato con $\mathbf{Z_{norm}}$. Prendendo inoltre atto del fatto che, procedendo con l'attività operativa lo stato di degrado del sistema aumenta e che, gli *embedding vectors* forniscono una rappresentazione dello stato del sistema, è doveroso concludere che lo stato di degrado del sistema in un

determinato istante può essere valutato e calcolato come la deviazione del relativo *embedding vector* dai *normal embedding vectors*.
Tale deviazione può essere stimata come segue:

$$d_i = \frac{1}{N} \sum_{\mathbf{z} \in \mathbf{Z_{norm}}} \|\mathbf{z_i} - \mathbf{z}\|_2 \tag{5.10}$$

Con N numero di *normal embedding vectors* che compongono $\mathbf{Z_{norm}}$.

Come riportato poc'anzi, tale deviazione riflette lo stato di degrado, o dualmente di salute, del sistema ed è quindi concettualmente correlata con l'indice HI. Essendo quest'ultimo un indice che per definizione assume valori nell'intervallo continuo $[0, 1]$, il mapping biunivoco tra le due grandezze può essere ottenuto dalla seguente relazione:

$$h_i = \frac{d_{max} - d_i}{d_{max} - d_{min}} \tag{5.11}$$

Dove d_{max} e d_{min} rappresentano rispettivamente il valore massimo e minimo di deviazione d_i per il j-esimo Caso.

Nel presente caso, tenendo conto della natura del *Milling dataset* ed in particolar modo del numero ridotto di corse che compongono le singole istanze, come *normal embedding vector* per ogni istanza è stato considerato solamente il primo $\{\mathbf{z_1}\}$, ovvero l'*embedding vector* relativo alla prima corsa sperimentata. Applicando la Relazione 5.11 a tutte le N_j corse che compongono l'istanza $\mathbf{I_j}$ si arriva ad ottenere la sequenza temporale univariata $\mathbf{HI_j}^{(Tg)} = \{h_1, h_2, ..., h_{N_j}\}$ ovvero è stato ottenuto in modalità non-supervisionata l'andamento del degrado del sistema relativamente al j-esimo Caso.

5.4 Regressione Lineare

Features extraction

Il set di features considerato, nella fase di definizione e training del modello di regressione lineare, ha previsto l'estrazione di cinque features, per ciascuno dei sei sensori, ottenendo un set iniziale di 30 features. Considerando la j-esima corsa $\mathbf{X_j}$, la relativa sequenza temporale uni-variata di durata N corrispondente all'i-esimo sensore è stata definita come $\mathbf{x}^{(i)} = \{x_1^i, x_2^{(i)}.., x_t^{(i)}, .., x_N^{(i)}\}$.
Sulla base di tale notazione, le cinque features estratte sono state le seguenti:

- Valore medio nel dominio del tempo:

$$\mu_T = \frac{1}{N}\sum_{k=1}^{N} x_k^{(i)} \tag{5.12}$$

- Deviazione standard nel dominio del tempo:

$$\sigma_T = \frac{\sqrt{\sum_{k=1}^{N}(x_k^{(i)} - \mu_T)^2}}{N} \tag{5.13}$$

- Mediana nel dominio del tempo:

$$M_T = \begin{cases} {x^{sorted}_{(N+1)/2}}^{(i)} & \text{se N è dispari} \\ \\ \frac{x^{(i),sorted}_{N/2} + x^{(i),sorted}_{(N+1)/2}}{2} & \text{se N è pari} \end{cases} \tag{5.14}$$

- Valore medio nel dominio della frequenza:

$$\mu_F = \frac{1}{N_F}\sum_{k=1}^{N_F} X_k^{(i)} \tag{5.15}$$

- Deviazione standard nel dominio della frequenza:

$$\sigma_F = \frac{\sqrt{\sum_{k=1}^{N_F}(X_k^{(i)} - \mu_F)^2}}{N_F} \tag{5.16}$$

Applicando l'algoritmo FFT alla sequenza $\mathbf{x^{(i)}}$ e indicando così con $\mathbf{X^{(i)}}$ la corrispondente trasformata, si ha che: X_k è il k-esimo campione del sequenza $\mathbf{X^{(i)}}$ e N_F è il numero di punti per il calcolo della FFT, nel caso in esame N_F è stato fissato pari a 8192. Infine $\mathbf{x^{sorted(i)}}$ corrisponde alla sequenza $\mathbf{x^{(i)}}$ ordinata in modo crescente.

Features selection

La totalità di *features* ottenute nella fase precedente sono state testate preliminarmente, tuttavia questo ha comportato il raggiungimento di prestazioni insoddisfacenti, con valori di RMSE e MAPE relativamente elevati, supportando difatti la necessità dell'integrazione di un meccanismo di *features selection*. Tale meccanismo ha infatti lo scopo di ridurre il numero

ancora elevato di attributi, rimuovendo le *features* ridondanti o scorrelate con il processo in esame. A questo proposito, la procedura qui perseguita ha previsto due fasi principali e consecutive, con una progressiva selezione e conseguente riduzione del numero di features: fase iniziale di testing dei singoli sensori e fase secondaria di l'applicazione dell'algoritmo *stepwise regression.*

Procedendo nel dettaglio, nella prima fase sono state valutate le prestazioni dei singoli sensori al fine di discriminare quali tra questi presentassero records in linea e correlati con il processo di degrado del sistema e quali invece fossero altamente scorrelati. A seguito quindi di differenti test e prove, e supportati da lavori presenti in letteratura [35], [37], si è ottenuto che per quanto concerne il materiale steel i sensori utili e associati al processo di lavorazione risultano essere: *smcAC* (corrente alternata del motore del mandrino), *AE_spindle* (emissioni acustiche del tavolo da taglio) e *AE_table* (emissioni acustiche del mandrino); per quanto riguarda invece il materiale *cast iron* è stato identificato un solo sensore utile e corrispondente con *smcAC* (corrente alternata del motore del mandrino). Questa prima fase ha quindi comportato una riduzione del numero di *features* selezionando: cinque features per *steel* e 15 *features* per *cast iron.* Nella seconda fase è stato invece applicato l'algoritmo semi-automatico *stepwise regression* ai due set di features ottendendo il sottoinsieme di *features* ottimo $\mathcal{F}^{(sub)}$. In Tabella 5.1 vengono riportare le *features* ottenute in ciascuna fase.

	Steel			**Cast iron**		
	Features	**Sensori**	**N**	**Features**	**Sensori**	**N**
Set features iniziale	☑ μ_T ☑ σ_T ☑ M_T ☑ μ_F ☑ σ_F	☑ smcAC ☑ smcDC ☑ AE_spindle ☑ AE_table ☑ vib_spindle ☑ vib_table	30	☑ μ_T ☑ σ_T ☑ M_T ☑ μ_F ☑ σ_F	☑ smcAC ☑ smcDC ☑ AE_spindle ☑ AE_table ☑ vib_spindle ☑ vib_table	30
Testing sensori	☑ μ_T ☑ σ_T ☑ M_T ☑ μ_F ☑ σ_F	☑ smcAC ✖ smcDC ☑ AE_spindle ☑ AE_table ✖ vib_spindle ✖ vib_table	15	☑ μ_T ☑ σ_T ☑ M_T ☑ μ_F ☑ σ_F	☑ smcAC ✖ smcDC ✖ AE_spindle ✖ AE_table ✖ vib_spindle ✖ vib_table	5
	Features finali		**N**	**Features finali**		**N**
Stepwise regression	– μ_T - *AE_spindle* – v_T - *AE_table* – M_T - *AE_spindle* – v_T - *AE_spindle*		4	– v - *smcAC* – μ_F - *smcAC* – v_F - *smcAC*		3

Tabella 5.1: Features selection

Training modello di regressione lineare

Definito il sottoinsieme di features ottimo $\mathcal{F}^{(sub)}$ per entrambi i materiali, per ciascuna corsa $\mathbf{X_i}$ viene estratto il corrispondente vettore di features $\mathbf{f_i} = \{f_1^{(i)}, f_2^{(i)}, ..., f_m^{(i)}\}$, con $f_j \in \mathcal{F}^{(sub)}$; recuperando inoltre il corrispondente valore HI target $h_i^{(Tg)}$ generato in precedenza.
Il training del modello di regressione lineare:

$$h = \beta_0 + \boldsymbol{\beta}^{\mathbf{T}}\mathbf{f} \tag{5.17}$$

si basa sull'insieme di coppie:

$$\{(\mathbf{f_i^{(j)}}, h_i^{(Tg)(j)})\} \tag{5.18}$$

Con

- $\mathbf{f_i^{(j)}}$: vettore di features relativo all'i-esima corsa appartenente al j-esimo Caso di training.
- $h_i^{(Tg)(j)}$: indice HI target corrispondente alla corsa considerata e generato in precedenza sulla base dell'*autoencoder*.

Il training del modello è volto alla stima dei parametri $\hat{\beta}_0$ e $\hat{\boldsymbol{\beta}}$, definiti minimizzando:

$$E = \sum_j \sum_i (h_i^{(i)} - h_i^{(Tg)(j)})^2 \tag{5.19}$$

5.5 Costruzione della libreria offline

Le stime $\hat{\beta}_0$ e $\hat{\boldsymbol{\beta}}$ ottenute impiegando i valori HI target vengono ora utilizzate per definire e costruire le curve HI relative ai singoli Casi. Considerando infatti il j-esimo Caso $\mathbf{I_j}$, costituito da N_j corse, tramite il modello di regressione lineare si calcola l'indice h_i per ciascuna corsa $\mathbf{X_i}$, rappresentata dal vettore di features $\mathbf{f_i}$:

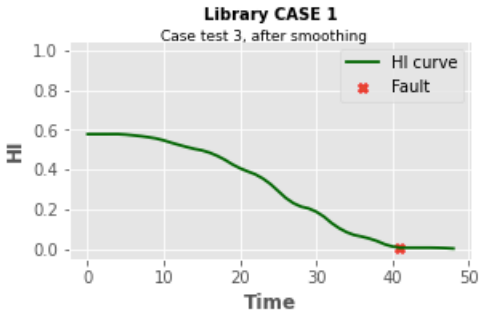
$$h_i = \hat{\beta}_0 + \hat{\boldsymbol{\beta}}^{\mathbf{T}}\boldsymbol{f_i} \tag{5.20}$$

In questo modo si è ottenuta la sequenza temporale $\mathbf{HI_j} = \{h_1, h_2, ...h_i, ..., h_{N_j}\}$ ed applicando il modello a tutti i Casi di training di arriva a definire l'insieme di curve $\mathbf{HI} = \{\mathbf{HI_j}\}$.
Si dimostra che le singole curve HI così ottenute risultano non monotone a causa del rumore intrinseco alle acquisizioni dei sensori. Al fine di ridurre tale rumore, viene integrata la tecnica *moving average* come tecnica di smoothing, fissando per entrambi i materiali la *window*

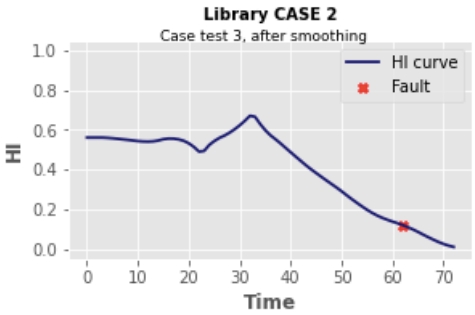
size pari a 10. In seguito a ciò le curve HI finali vanno a popolare la libreria mantenuta in memoria.

La costruzione della libreria conclude la fase offline.

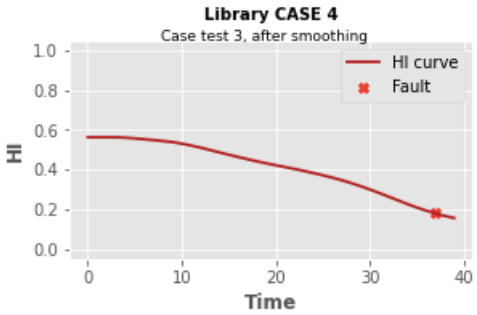
In Figura 5.4 viene mostrata la libreria considerando il Caso 3 come Caso di test e evidenziando l'istante temporale di guasto, ovvero l'istante in cui per la prima volta il parametro VB supera la soglia $0.45mm$. Si osserva che le curve HI offline così memorizzate presentano valori HI all'istante iniziale piuttosto differenti e differenti sono anche i range di variazione dell'indice HI.



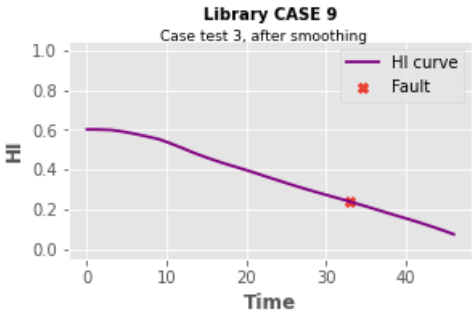
(a) Curva HI relativa al Caso 1



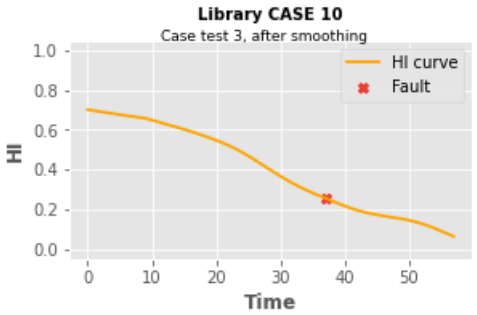
(b) Curva HI relativa al Caso 2



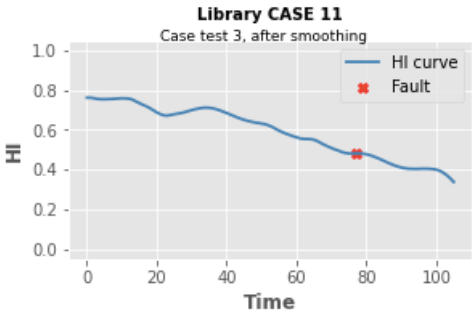
(c) Curva HI relativa al Caso 4



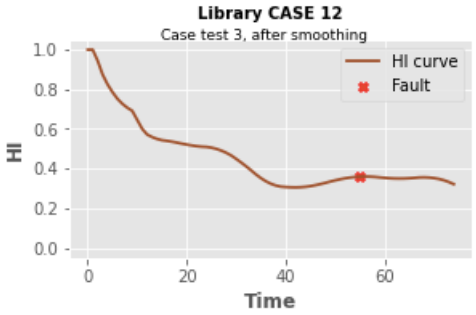
(d) Curva HI relativa al Caso 9



(e) Curva HI relativa al Caso 10



(f) Curva HI relativa al Caso 11



(g) Curva HI relativa al Caso 12

Figura 5.4: Libreria offline relativa al Caso test 3

5.6 Online test

La fase online del framework prevede la stima della RUL di un Caso di test, utilizzando le curve HI appartenenti alla libreria popolata nella fase di training (fase offline). Fornito

il Caso di test $\mathbf{I_{test}}$, composto da N_T corse $\mathbf{I_{test}}$=$\{\mathbf{X_1}, \mathbf{X_2}, ..., \mathbf{X_T}\}$, in primo luogo i *raw-data* relativi a ciascuna di esse seguono la fase di pre-processing descritta nella Sottosezione 5.1, che riassumendo prevede la selezione dei soli campioni relativi alla fase di lavoro e la tecnica MAD per l'eliminazione degli outliers. Da questo segue, per ciascuna corsa, si procede all'estrazione delle features appartenenti al set $\mathcal{F}^{(sub)}$ (in particolar modo $\mathcal{F}^{(sub)}_{steel}$ o $\mathcal{F}^{(sub)}_{castiron}$ a seconda del materiale del Caso di test).

Integrando successivamente il modello di regressione lineare, definito nella fase offline, e fornendo in input ad esso il vettore di features $f_i^{(test)}$ estratto dalla corsa $\mathbf{X_i}$ si arriva a stimare il relativo indice h_i. Procedendo per tutte le corse si costruisce la curva HI relativa al Caso di test. Restando valide le precedenti assunzioni sulla non-monotonicità e rumorosità delle curve HI, costruite seguendo tale procedura, si va ad applicare allo stesso modo la tecnica *moving average* con lo scopo di ridurre tali limitazioni, ottenendo così la curva finale di test. Integrando la tecnica *similarity curve-based matching*, tale curva finale di test $\mathbf{HI}_{test}$ viene confrontata con ciascuna curva HI appartenente alla libreria, usando un'opportuna metrica di somiglianza. Nel fase di *curve-matching*, poiché le due curve in esame possono presentare un diverso valore iniziale HI, la curva di test viene traslata di un *time-lag* pari a $\tau_{optimal}$ tale da minimizzare la distanza euclidea tra le due curve (Equazione 5.21 e 5.22).

$$\tau_{optimal} = \underset{\tau \in [0, T_j - T_{test}]}{\operatorname{argmin}} d(\mathbf{HI}^{(test)}, \mathbf{HI}^{(j)}, \tau) \tag{5.21}$$

$$d(\mathbf{HI}^{(test)}, \mathbf{HI}^{(j)}, \tau) = \frac{1}{T_{test}} \sum_{i=0}^{T_{test}} (h_i^{(test)} - h_{i+\tau}^{(j)})^2 \tag{5.22}$$

Con T_{test} lunghezza della curva $HI^{(test)}$ e T_j lunghezza della curva HI di training $HI^{(j)}$. Il calcolo della metrica di somiglianza, tra la j-esima curva HI memorizzata nella libreria (relativa al Caso j) e la curva HI di test $\mathbf{HI}_{test}$ traslata di una quantità pari a τ viene definito come segue:

$$Sim(j, \tau) = \exp\left(\frac{-d(\mathbf{HI}_{test}, \mathbf{HI}^{(j)}, \tau)}{\lambda}\right) \tag{5.23}$$

Dove il parametro λ è un fattore di rilassamento che controlla il grado di somiglianza, qui fissati pari a 0.001. La stima del valore di RUL del Caso di test sulla base del Caso di training j è pari a:

$$RUL(j, \tau_{optimal}) = T_j - T_{test} - \tau_{optimal} \tag{5.24}$$

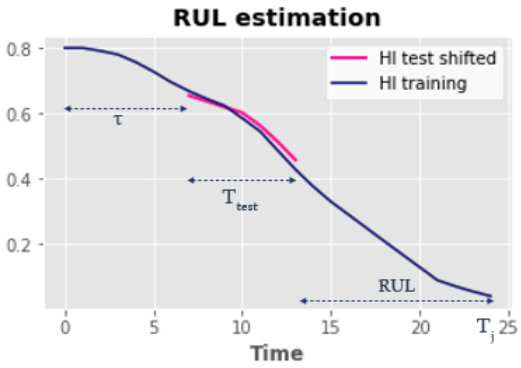
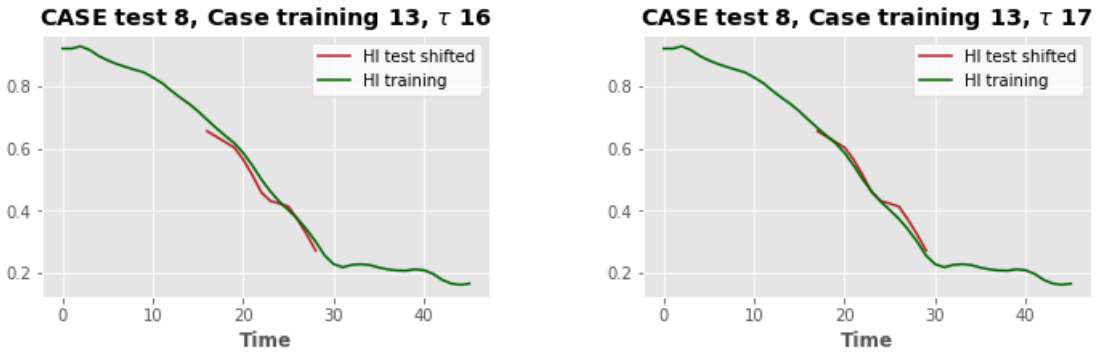


Figura 5.5: Esempi di HI curve-matching con un *time lag* τ

Considerando il significato fisico della RUL, è necessario sempre avere $RUL \geq 0$ e dunque il parametro $\tau_{optimal}$ è limitato nell'intervallo $\tau \in [0, T_j - T_{test}]$.
Nonostante tutto questo, nell'implementazione del framework, nel confronto tra la curva di test e ciascuna curva di training $\mathbf{HI}^{(j)}$ non è stato calcolato un singolo valore di RUL_j corrispondente al *time-lag* ottimo $\tau_{optimal}$, ma è stata calcolata la RUL e il grado di somiglianza per ciascun $\tau \in [0, T_j - T_{test}]$. La giustificazione alla base di tale scelta è dettata dal fatto che nella fase di *curve-matching* è possibile avere multipli valori di τ con elevati gradi di somiglianza e quindi multipli valori di RUL con gradi di somiglianza comparabili (Figura 5.6).



(a) $\tau = 16$ (b) $\tau = 17$

Figura 5.6: Esempio di multipli τ con elevata somiglianza

Applicando infine tale procedura a tutte le curve HI di training $\mathbf{HI}^{(j)}$ presenti nella libreria, la RUL finale è definita come una media ponderata delle RUL relative alle singole curve di

training più simili ed in cui i pesi equivalgono al rispettivo grado di somiglianza:

$$RUL(j,\tau) = \frac{\sum_j \sum_\tau Sim(j,\tau) * RUL(j,\tau)}{\sum_j \sum_\tau Sim(j,\tau)} \tag{5.25}$$

$$\text{Con } Sim(j,\tau) \geq \beta * Sim(j,\tau)_{max} \tag{5.26}$$

$Sim(j,\tau)_{max}$ indica il massimo livello di somiglianza ottenuto, mentre il parametro β discrimina l'insieme di RUL che vanno a comporre la media ponderata.

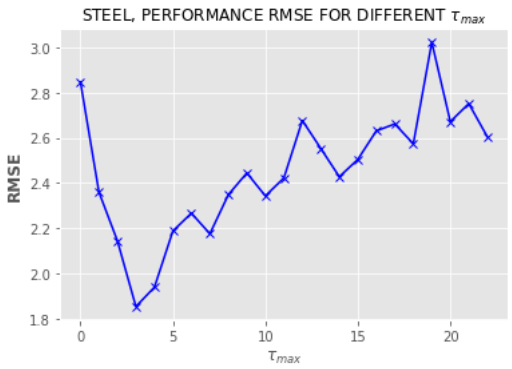
É stato qui fissato pari a 0.95.

Riprendendo l'idea applicata in [36], è stato definito ed aggiunto un ulteriore limite superiore al *time lag*[2] indicato con τ_{max}, e il quale rappresenta il massimo shift temporale della curve HI di test in relazione alla singola curva di training.

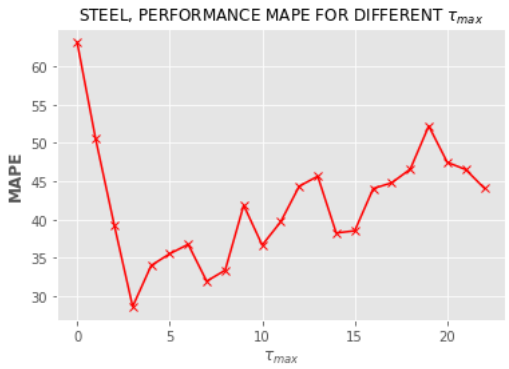
Avendo voluto definire il valore τ_{max} ottimo, sono state valutate le prestazioni del framework, in termini di MAPE ed RMSE, al variare di τ_{max} nell'intervallo $[0, (T_j - T_{test})]$.

In virtù dei risultati ottenuti e riportati in Figura 5.7, τ_{max} è stato fissato pari a 4 sia per il materiale *cast iron* che per il materiale *steel*.

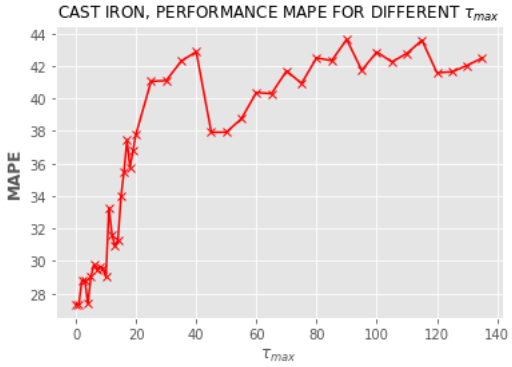
[2]Si ricordi come nella definizione iniziale si ha $\tau \in [0, T_j - T_{test}]$, e quindi necessariamente $\tau_{max} \geq (T_j - T_{test})$.



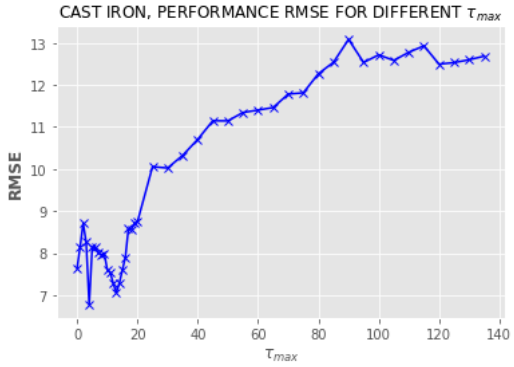
(a) Andamento RMSE, materiale *steel*



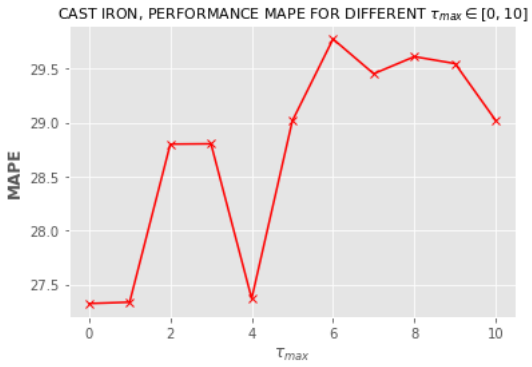
(b) Andamento MAPE, materiale *steel*



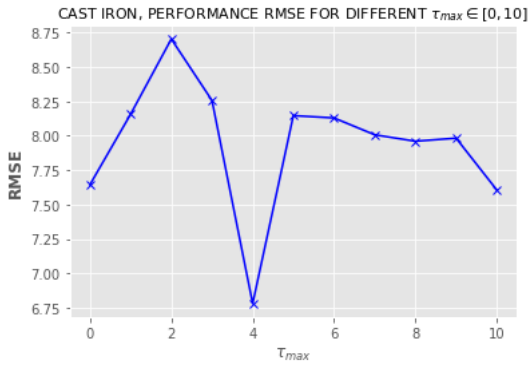
(c) Andamento MAPE, materiale *cast iron*



(d) Andamento RMSE, materiale *cast iron*



(e) Andamento MAPE nell'intervallo $\tau_{max} \in [0, 10]$, materiale *cast iron*



(f) Andamento RMSE nell'intervallo $\tau_{max} \in [0, 10]$, materiale *cast iron*

Figura 5.7: Risultati in termini di RMSE e MAPE per il materiale *cast iron* al variare del parametro τ_{max}

5.7 Risultati

5.7.1 Materiale *steel*

In Tabella 5.2 vengono riportati, per il materiale *steel*, i risultati ottenuti, in termini di MAPE e RMSE, del framework proposto (BiLSTM-ED-attention) e di diversi ulteriori approcci prognostici, allo stato dell'arte, presentati in letteratura. Si osserva come il framework im-

plementato ottenga prestazioni comparabili ed addirittura migliori dei più moderni approcci prognostici proposti.

	RMSE	MAPE
BiLSTM-ED-attention	1.86	28.04%
Temporal convolution network [34]	2.37	52%
LSTM [32]	2.45	-
LSTM-ED [36]	2.66	31.7%
BiLSTM-ED [35]	2.36	38%
BiGRU-AS [37]	2.45	32%

Tabella 5.2: Risultati per il materiale *steel* in termini di RMSE e MAPE

In Figura 5.8 vengono riportati i risultati in termini di valori di RUL, stimati e target, avendo considerato la totalità dei Casi di test e ordinando i valori RUL target in ordine crescente al fine di facilitare la visualizzazione dei risultati. In questo modo è infatti possibile osservare l'entità dell'addensamento della nuvola dei valori stimati attorno ai valori target.

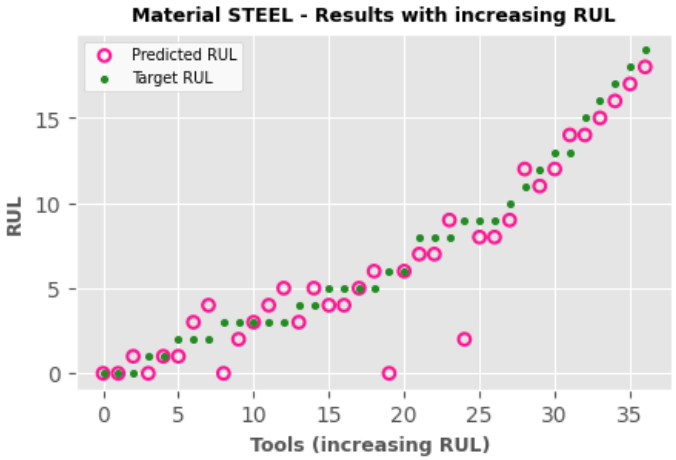
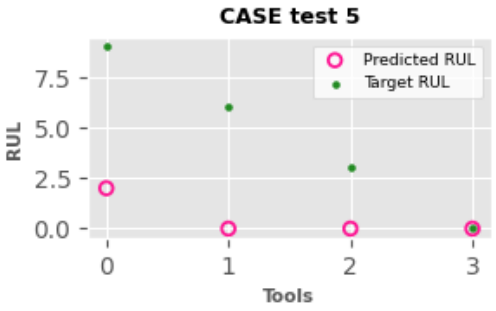
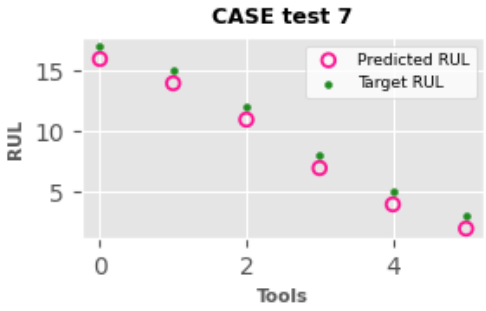


Figura 5.8: Risultati finali per il materiale *steel*

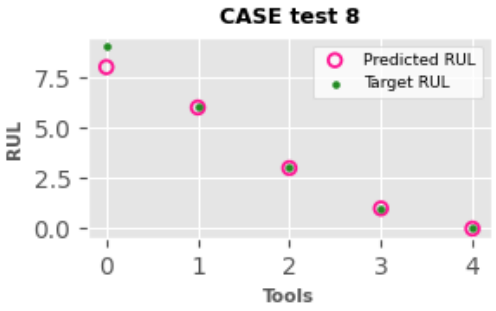
In Figura 5.9 vengono invece presentati i risultati in termini di valori di RUL, suddivisi in base al Caso di test.



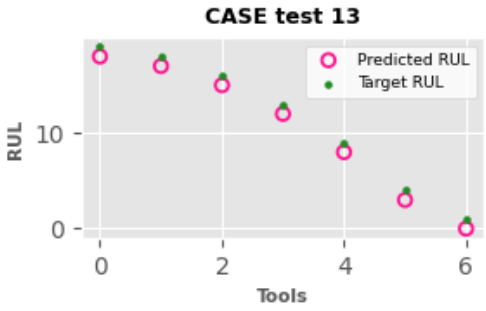
(a) Risultati Caso 5



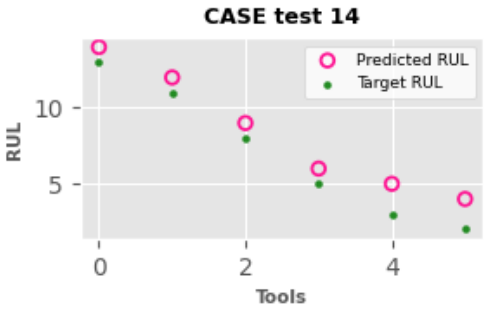
(b) Risultati Caso 7



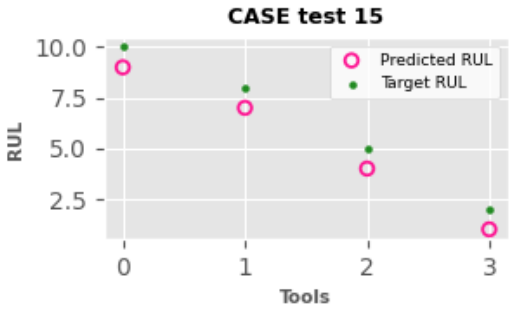
(c) Risultati Caso 8



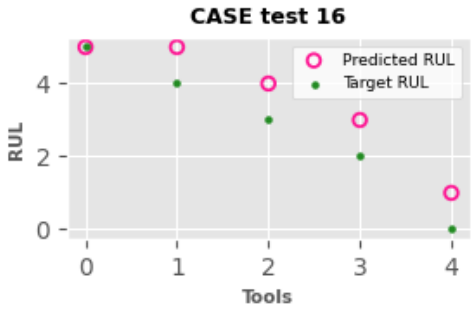
(d) Risultati Caso 13



(e) Risultati Caso 14



(f) Risultati Caso 15



(g) Risultati Caso 16

Figura 5.9: Risultati finali per il materiale *steel*, divisi per ciascun Caso

5.7.2 Materiale *cast iron*

Sulla falsa riga di cui sopra, in Tabella 5.3 vengono riportati, per il materiale *cast iron*, i risultati ottenuti (BiLSTM-ED-attention), in termini di MAPE e RMSE, mantenendo sempre

il confronto con gli approcci prima presentati. In questo caso si osserva come, in termini di RMSE, il sistema ottenga prestazioni comparabili con i restanti approcci, mentre in termini di MAPE, la soluzione qui proposta si rivela meno performante.

	RMSE	MAPE
BiLSTM-ED-attention	6.78	27.37%
Temporal convolution network [34]	5.86	46%
LSTM [32]	6.88	-
LSTM-ED [36]	8.44	26%
BiLSTM-ED [35]	7.14	24%
BiGRU-AS [37]	6.88	22%

Tabella 5.3: Risultati per il materiale *cast iron* in termini di RMSE e MAPE

In Figura 5.10 vengono riportati i risultati in termini di valori di RUL, stimati e target, avendo considerato la totalità delle fasi di test e ordinando i valori RUL target in ordine crescente al fine di facilitare la visualizzazione dei risultati.

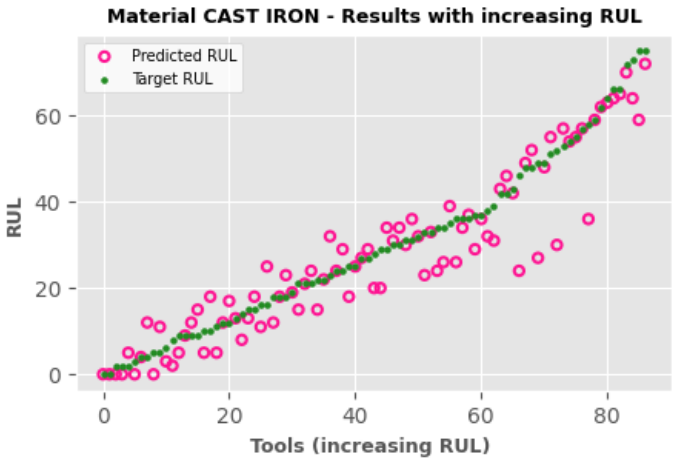
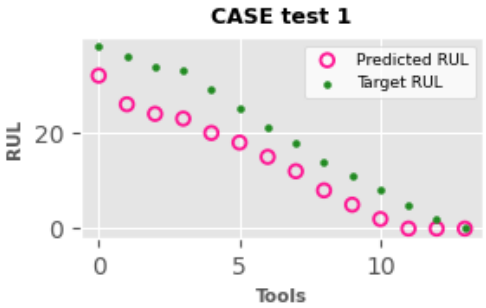
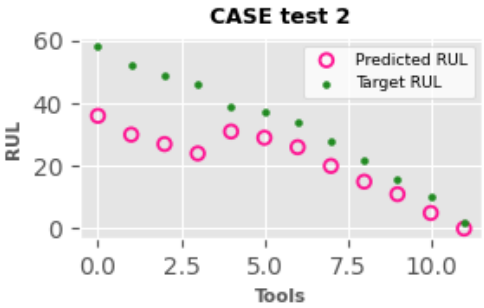


Figura 5.10: Risultati finali per il materiale *cast iron*

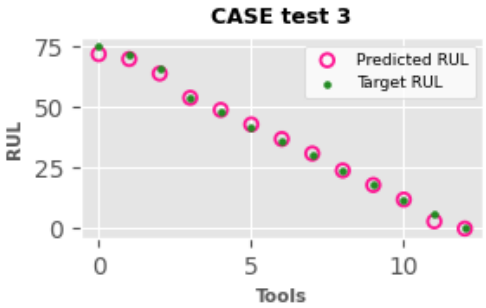
In Figura 5.11 vengono invece presentati i risultati in termini di valori di RUL, suddivisi in base al Caso di test. Si osserva in questo caso, come il sistema incontri difficoltà nel predire la vita residua utile in una situazione di buon funzionamento, ovvero nel corso delle corse iniziali. Questo comportamento inoltre ha maggiore rilievo laddove tali valori di RUL sono più elevati, ovvero nella situazione di Casi di durata temporale maggiore.



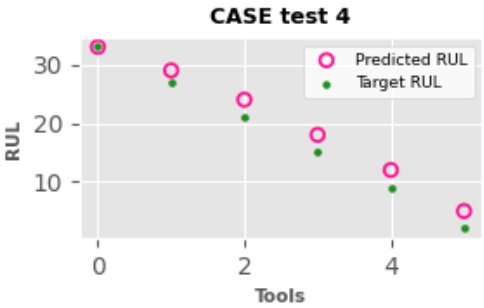
(a) Risultati Caso 1



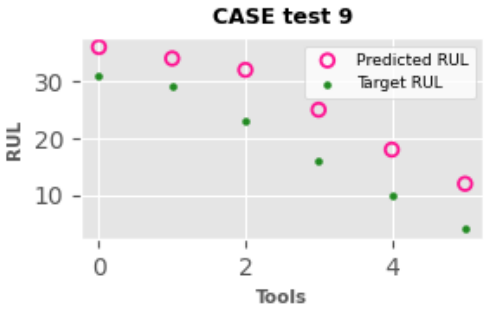
(b) Risultati Caso 2



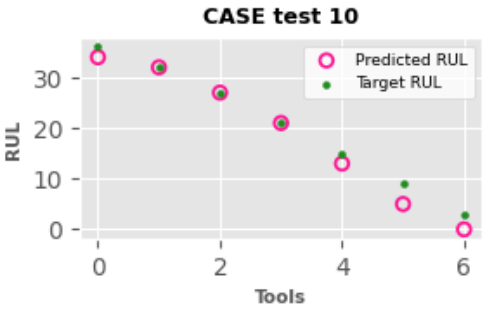
(c) Risultati Caso 3



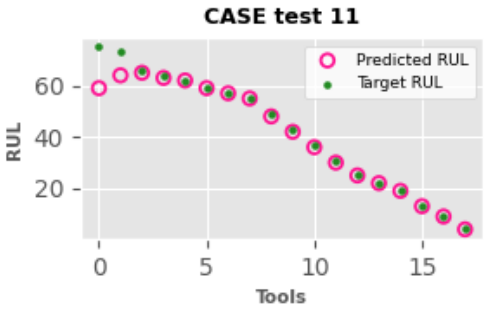
(d) Risultati Caso 4



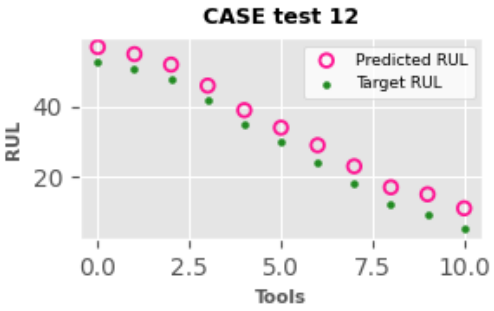
(e) Risultati Caso 9



(f) Risultati Caso 10



(g) Risultati Caso 11



(h) Risultati Caso 12

Figura 5.11: Risultati finali per il materiale *cast iron*, divisi per ciascun Caso

Capitolo 6

Conclusioni

La ricerca di Tesi ha comportato lo studio e l'analisi dei principali strumenti e metodi di manutenzione predittiva con particolare interesse a tre dei suoi aspetti costituenti: tecnologie industriali di *condition monitoring*, *Industrial Internet of Things* (IIoT) e approcci matematici e statistici volti alla previsione della RUL. Come caso studio è stato considerato il *Milling dataset*, un dataset pubblico relativo a sperimentazioni su una macchina fresatrice. Dopo una revisione dei principali approcci prognostici applicati/validati su esso e presentati in letteratura, è stato definito e implementato un proprio framework di manutenzione del tipo *data driven*. Il framework prevede in particolar modo tra i vari componenti: un *autoencoder* per l'estrazione in modalità non supervisionata dello stato di salute del macchinario e la tecnica *similarity-based curve matching* per la stima della RUL. Tale metodologia si rivela vincente per due motivi principali:

- L'addestramento dell'*autoencoder* in modalità non-supervisionata è il linea con lo scenario della manutenzione predittiva, in particolar modo con i *run-to-failure data* che nella maggior parte dei casi risultano essere difficili da etichettare. Inoltre, essendo lo stato di degrado del processo definito come scostamento dallo stato di buon funzionamento del macchinario, anche nel caso di un numero limitato di *failure data* nel *training set* è possibile ottenere una stima della RUL affidabile. A differenza quindi dei più comuni approcci di *deep learning* supervisionati (es. DNN, CNN, RNN) nei quali uno sbilanciamento tra i dati di salute e i dati di guasto nel *training set* comporta un notevole impatto nelle prestazioni dell'intero sistema.

- La tecnica *similarity-based curve matching* prevede il confronto di un Caso di test con ciascun Caso di training memorizzato nel sistema; la previsione della RUL si basa quindi

sullo storico delle osservazioni effettuate. In questo modo è possibile ottenere risultati soddisfacenti anche laddove il *training set* sia di dimensioni ridotte;

- L'architettura del framework, composta di fase online ed offline, si presta agevolmente al continuo aggiornamento della libreria memorizzata, aggiungendo ad essa il Caso di test una volta effettuata la previsione. Seppur, tale aggiornamento richieda una nuova fase di *training* sia dell'*autoencoder* che del modello di regressione lineare, è stato osservato come i relativi tempi di *training* siano considerevolmente ridotti rispetto alle più comuni tecniche di *deep learning*.

La sperimentazione effettuata ha fornito risultati comparabili e in taluni casi migliori dei più moderni approcci prognostici presenti in letteratura, dimostrando in questo modo la validità delle considerazione e scelte implementative effettuate.
Una possibile direzione futura volta al miglioramento del framework è quella che prevede le seguenti integrazioni/sostituzioni:

- Una tecnica di *data cleaning* maggiormente performante nell'identificazione degli *outliers*;

- Sostituzione del modello di regressione lineare nella definizione delle curve HI. Questo perché la regressione lineare descrive appunto con un modello lineare la relazione tra *features* estratte dai sensori e *health index*. Tuttavia, la relazione non sempre risulta lineare e in taluni casi quindi, il modello di regressione, incapace di descrivere funzioni non lineari, non si rivela efficiente. Perciò una possibile scelta è quella dell'integrazione di un modello di regressione non lineare come ad esempio un modello basato su reti neurali;

- La definizione di modelli multipli di stima della RUL che presentino la medesima architettura del framework ma ciascuno definito e addestrato sulla base di un unico sensore. Calcolando al termine il valore di RUL come un'opportuna media ponderata delle stime dei singoli modelli. In questo modo si andrebbe a differenziare il contributo dei singoli sensori nella stima della RUL, a differenza del framework qui implementato nel quale una singola curva HI rappresenta il processo di degrado del sistema considerando la totalità dei sensori.

Bibliografia

[1] Plant Engineering, "Industrial Maintenance Report," March 2021.

[2] R. L.Dunn, "Predictive Maintenance Technologies." `https://www.plantengineering.com/articles/predictive-maintenance-technologies/`, 15 June, 2002. [Online; accessed 12-09-2021].

[3] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng, "A Survey of Predictive Maintenance: Systems, Purposes and Approaches," 2019.

[4] 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.

[5] "Plastics – Determination of fracture toughness – Linear elastic fracture mechanics (LEFM) approach," standard, International Organization for Standardization, Geneva, CH, 2018.

[6] H. M. Hashemian and W. C. Bean, "State-of-the-Art Predictive Maintenance Techniques," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 1, pp. 226–236, 2011.

[7] J. M. Leod, P. A. Baziuk, R. Calvo, and S. S. Rivera, "Failure Profiles for Maintenance in Industrial Facilities," in *Proceedings of the World Congress on Engineering*, vol. II, (London, UK), July 2015.

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

[9] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part I: Quantitative model-based methods," *Computers & Chemical Engineering*, vol. 27, no. 3, pp. 293–311, 2003.

[10] *Operations & Maintenance Best Practices - A Guide to Achieving Operational Efficiency Release 3.0.* 8 2010.

[11] I. Ullah, F. Yang, R. Khan, L. Liu, H. Yang, B. Gao, and K. Sun, "Predictive Maintenance of Power Substation Equipment by Infrared Thermography Using a Machine-Learning Approach," *Energies*, vol. 10, no. 12, 2017.

[12] J. Zhu, J. Yoon, D. He, Y. Qu, and E. Bechhoefer, "Lubrication Oil Condition Monitoring and Remaining Useful Life Prediction with Particle Filtering," *International Journal of Prognostics and Health Management*, vol. 4, no. 3, 2013.

[13] B. Corne, B. Vervisch, C. Debruyne, J. Knockaert, and J. Desmet, "Comparing MCSA with vibration analysis in order to detect bearing faults - A case study," in *2015 IEEE International Electric Machines Drives Conference (IEMDC)*, pp. 1366–1372, 2015.

[14] A. Bonci, S. Longhi, G. Nabissi, and F. Verdini, "Predictive Maintenance System using motor current signal analysis for Industrial Robot," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1453–1456, 2019.

[15] L. D. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[16] S. F. Tan and A. Samsudin, "Recent Technologies, Security Countermeasure and Ongoing Challenges of Industrial Internet of Things (IIoT): A Survey," *Sensors*, vol. 21, no. 19, 2021.

[17] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[18] M. Compare, P. Baraldi, and E. Zio, "Challenges to IoT-Enabled Predictive Maintenance for Industry 4.0," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4585–4597, 2020.

[19] D. Borissova, I. Mustakerov, and L. Doukovska, "Predictive Maintenance Sensors Placement by Combinatorial Optimization," *International Journal of Electronics and Telecommunications*, vol. 58, no. 2, 2012.

[20] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, "Machinery health prognostics: A systematic review from data acquisition to RUL prediction," *Mechanical Systems and Signal Processing*, vol. 104, pp. 799–834, 2018.

[21] M. Lombardi, F.Pascale, and D. Santaniello, "Internet of Things: A General Overview between Architectures, Protocols and Applications," *Information*, vol. 12, no. 2, 2021.

[22] L. Mainetti, L. Patrono, and A. Vilei, "Evolution of wireless sensor networks towards the Internet of Things: A survey," in *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*, pp. 1–6, 2011.

[23] T. Salman and R. Jain, "Networking Protocols and Standards for Internet of Things," *IoT Handbook*, 2016.

[24] J. Zhang and D. Tao, "Empowering Things With Intelligence: A Survey of the Progress, Challenges, and Opportunities in Artificial Intelligence of Things," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 7789–7817, 2021.

[25] I. Ungurean and N. C. Gaitan, "A Software Architecture for the Industrial Internet of Things—A Conceptual Model," *Sensors*, vol. 20, no. 19, 2020.

[26] M. Baur, P. Albertelli, and M. Monno, "A review of prognostics and health management of machine tools," *The International Journal of Advanced Manufacturing Technology*, vol. 107, p. 2843–2863, 2020.

[27] J. Lee, F. Wu, W. Zhao, M.Ghaffari, L.Liao, and D. Siegel, "Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications," *Mechanical Systems and Signal Processing*, vol. 42, no. 1, pp. 314–334, 2014.

[28] L. Zhang, Z. Mu, and C. Sun, "Remaining Useful Life Prediction for Lithium-Ion Batteries Based on Exponential Model and Particle Filter, year=2018," *IEEE Access*, vol. 6, pp. 17729–17740.

[29] Y. Lei, N. Li, S. Gontarz, J. Lin, S. Radkowski, and J. Dybala, "A Model-Based Method for Remaining Useful Life Prediction of Machinery," *IEEE Transactions on Reliability*, vol. 65, no. 3, pp. 1314–1326, 2016.

[30] W. Zhang, D. Yang, and H. Wang, "Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2213–2227, 2019.

[31] A. Agogino and K. Goebel, "Milling Data Set." `https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/`, 2007.

[32] W. Cai, W. Zhang, Y. Liu, and X. Hu, "A hybrid information model based on long short-term memory network for tool condition monitoring," *Journal of Intelligent Manufacturing*, vol. 31, p. 1497–1510, 2020.

[33] F. Aghazadeh, A.Tahan, and M.Thomas, "Tool condition monitoring using spectral subtraction and convolutional neural networks in milling process," *The International Journal of Advanced Manufacturing Technology*, vol. 98, p. 3217–3227, 2018.

[34] J.Chen, D. Chen, and G. Liu, "Using temporal convolution network for remaining useful lifetime prediction," *Engineering Reports*, vol. 3, no. 3, p. 1497–1510, 2021.

[35] W.Yu, I.Y.Kim, and C.Mechefske, "Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme," *Mechanical Systems and Signal Processing*, vol. 129, pp. 764–780, 2019.

[36] P. Malhotra, T. Vishnu, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. M. Shroff, "Multi-Sensor Prognostics using an Unsupervised Health Index based on LSTM Encoder-Decoder," *ArXiv*, vol. abs/1608.06154, 2016.

[37] Y. Duan, H. Li, M. He, and D. Zhao, "A BiGRU Autoencoder Remaining Useful Life Prediction Scheme With Attention Mechanism and Skip Connection," *IEEE Sensors Journal*, vol. 21, no. 9, pp. 10905–10914, 2021.

[38] P. Wang and M. Russell, "Domain Adversarial Transfer Learning for Generalized Tool Wear Prediction," *Annual Conference of the PHM Society*, vol. 12, no. 1, p. 8, 2020.

[39] W. J. Lee, H. Wu, H. Yun, H. Kim, M. B. Jun, and J. W. Sutherland, "Predictive Maintenance of Machine Tool Systems Using Artificial Intelligence Techniques Applied to Machine Condition Data," *Procedia CIRP*, vol. 80, no. 3, pp. 506–511, 2019.

[40] E. Traini, G. Bruno, G. D'Antonio, and F. Lombardi, "Machine Learning Framework for Predictive Maintenance in Milling," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 177–182, 2019.

[41] A. Senawi, H.L.Wei, and S.Billings, "A New Maximum Relevance-Minimum Multicollinearity (MRmMC) Method for Feature Selection and Ranking," *Pattern Recognition*, vol. 67, pp. 47–61, 2017.

[42] M. A. Kramer, "Nonlinear Principal Component Analysis Using Autoassociative Neural Networks," *AIChE Journal*, vol. 37, no. 2, p. 233–243, 1991.

[43] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[44] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[45] T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (Lisbon, Portugal), pp. 1412–1421, Association for Computational Linguistics, Sept. 2015.

[46] M. Pastore, "I limiti dell'approccio NHST e l'alternativa Bayesiana," *GIORNALE ITALIANO DI PSICOLOGIA*, vol. 36, pp. 1735–80, 2009.

[47] C. Leys, O. Klein, P.Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, 2013.