ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA

Corso di Laurea Magistrale in

Ingegneria Informatica

Tree ensemble methods for Predictive Maintenance: a case study

Tesi di laurea in

Intelligent Systems

Relatore

Prof. Michela Milano

Presentata da

De Giorgi Marcello

Sessione III

Anno accademico

2019-20

Sommario

Introduzione

Negli ultimi vent'anni, la disponibilità di una maggior potenza di calcolo e di una quantità impressionante di dati ha permesso, e reso necessario, lo sviluppo di strumenti sempre più sofisticati per la loro gestione e manipolazione; l'impiego di sistemi di calcolo distribuiti e la possibilità di utilizzare algoritmi, fino ad ora relegati alla sola teoria, hanno favorito enormemente il progresso in numerosi ambiti, dal marketing all'industria, dalla medicina alla logistica e molti altri, fino a rivoluzionare il modo in cui interagiamo e comunichiamo col mondo.

Nel lavoro descritto in questa tesi sono stati addestrati dei modelli per il miglioramento produttivo in ambito industriale, rendendo possibile una politica di manutenzione predittiva su macchine utensili. A partire dall'analisi delle variabili raccolte durante la lavorazione e dall'integrazione di queste con informazioni relative ad eventi di manutenzione (quali guasti alla componentistica, blocco di alcune parti, anomalie segnalate dal controllore numerico), sono stati applicati metodi e algoritmi di intelligenza artificiale col fine di anticipare il verificarsi di questi eventi. Lo stesso approccio è stato quindi utilizzato per identificare il momento migliore per la sostituzione dell'utensile che effettua la lavorazione, con lo scopo di aumentare il numero di pezzi lavorati che soddisfino gli standard di qualità.

Nel capitolo 1 sarà fornito uno sfondo al problema trattato, con lo scopo di familiarizzare con l'ambito industriale in questione: in particolare si parlerà brevemente di Industria 4.0 e delle tecnologie legate alla nuova rivoluzione industriale. Si procederà quindi illustrando in maniera generale le macchine utensili a controllo numerico computerizzato, il fenomeno di usura degli utensili e il concetto di *fine vita*. Dopo aver trattato della lavorazione nel caso in esame, verranno presentate diverse politiche di manutenzione attuabili e si discuterà di come, nell'Industria 4.0, la migliore politica sia quella predittiva.

Il capitolo 2 sarà focalizzato sull'acquisizione e sull'aggregazione dei dati: saranno descritte le variabili raccolte e il processo di conversione, preparazione e aggregazione dei dataset, e l'integrazione delle informazioni sui cambi utensile e sugli eventi di manutenzione.

Nel capitolo 3 saranno presentati i risultati delle analisi effettuate sui dati: si illustrerà il comportamento assunto da alcune variabili tra un cambio utensile e il successivo e della loro

correlazione. Si parlerà quindi della divisione del dataset in cicli di lavorazione e della qualità dei dati osservando alcuni esempi di cicli validi e non validi. Si procederà illustrando la differenza tra tempo di lavorazione complessivo ed effettivo, e di come quest'ultimo abbia un ruolo importante nell'algoritmo di classificazione. Infine, verrà trattata l'aggregazione dei cicli validi, con la finalità di preparare un dataset di dimensioni ridotte da essere utilizzato dagli algoritmi di machine learning.

Nel capitolo 4, si discuterà degli algoritmi scelti e dei risultati ottenuti nella predizione degli eventi in macchina: dopo aver discusso delle motivazioni legate alla scelta degli ensemble tree methods, verrà trattato l'algoritmo Random Forest, illustrandone il funzionamento generale, le varianti e i miglioramenti, ed applicandolo al problema trattato. Si procederà quindi presentando un altro algoritmo, XGBoost, descrivendone i punti fondamentali e riportando i risultati ottenuti. Verrà quindi trattato il problema dell'accuratezza con dataset non bilanciati e sarà proposta un'alternativa per la stima della bontà dei modelli. Infine, verrà effettuato un confronto tra i due.

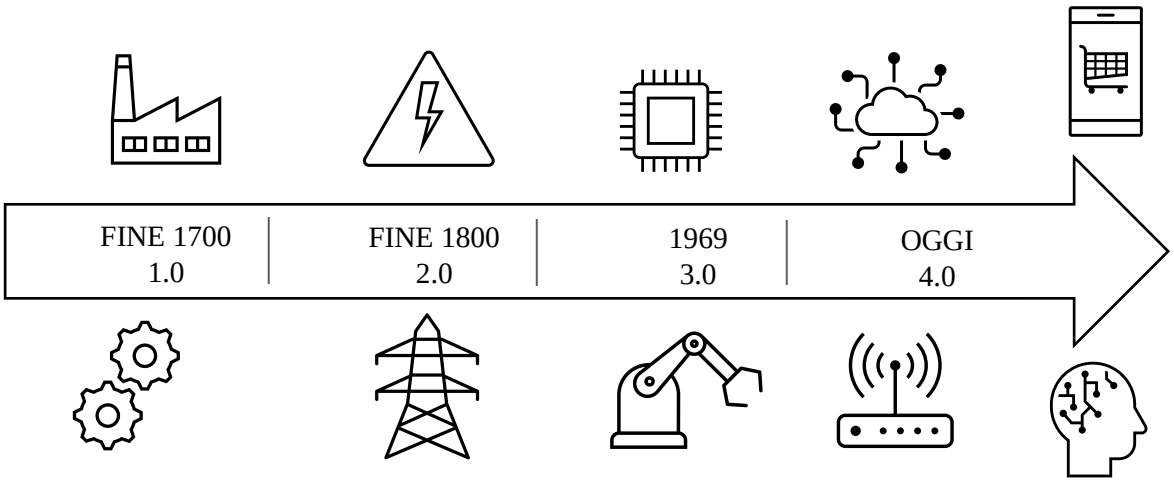
Quanto descritto in questo lavoro di tesi, la scelta degli algoritmi, le analisi e i dati utilizzati, sono frutto della mia esperienza di tirocinio presso **MindIT Srl**, azienda che mi ha permesso di vedere concretamente ed applicare concetti e metodologie viste durante il corso di studi, guidandomi con grande disponibilità e professionalità durante tutta l'esperienza.

Capitolo 1

Nel seguente capitolo verranno illustrati brevemente i concetti fondamentali dell'Industria 4.0. Saranno quindi introdotte le macchine CNC e si discuterà dell'usura degli utensili e della lavorazione della macchina in esame.

1.1 Industria 4.0

Con la disponibilità di nuove tecnologie, negli ultimi anni la lavorazione industriale e la logistica stanno procedendo verso una completa trasformazione degli approcci finora utilizzati. L'obiettivo posto da questa trasformazione è la soddisfazione diretta dei bisogni del consumatore, andando a presentare soluzioni precise e non più generalizzate. A tale scopo si ha una completa rivisitazione dell'intero ciclo produttivo, dalla progettazione iniziale alla delivery, fino al riciclo dei prodotti, in un flusso continuo di interazioni tra le parti in gioco [1]. Si parla pertanto di quarta rivoluzione industriale, ad indicare la netta separazione con il passato: dal vapore e dal carbone all'elettricità, dai macchinari industriali manovrati dagli operatori all'automatizzazione e infine, dall'automatizzazione ad un sistema vivo ed intelligente, che impara in maniera costante migliorandosi. Il passaggio non prevede soltanto l'ammodernamento delle macchine industriali con l'integrazione di sensoristica e raccolta costante di dati, ma l'impiego e l'integrazione di metodi interdisciplinari per l'ottimizzazione di ogni aspetto del sistema produttivo.



L’industria 4.0 si basa principalmente su 4 concetti: Internet of Things, Industrial Internet of Things, Cloud based manufacturing e Smart manufacturing. Questi concetti vengono sostenuti dall’applicazione di nove tecnologie, nel dettaglio:

1. **Big Data and Analytics**

   L’enorme mole di dati raccolta grazie alla sensoristica viene gestita, organizzata e infine analizzata per individuare pattern e informazioni di valore per il miglioramento complessivo degli step produttivi e per la predizione di eventi (ad esempio di guasti) il cui realizzarsi può portare all’interruzione del ciclo produttivo e ad un danno economico;

2. **Cloud**

   L’impiego di un servizio di cloud computing è fondamentale, non solo per garantire l’accesso rapido alle risorse a tutti gli attori del ciclo produttivo, ma anche per disporre di un’architettura scalabile di calcolo.

3. **Simulations**

   L’utilizzo di modelli virtuali che simulino con un buon livello di accuratezza il mondo reale permette di ridurre drasticamente i tempi di setup delle macchine e la creazione di prodotti difettosi o non conformi a causa di una non corretta configurazione iniziale dei macchinari, o verificare il consumo di energia e altri aspetti che richiederebbero l’impiego di cicli di lavorazione lunghi.

4. **Autonomous Robots**

   L’impiego di robot capaci di agire in autonomia permette l’ottenimento di risultati estremamente precisi e l’eliminazione di un controllo costante da parte di un operatore umano. Inoltre, la lavorazione può avvenire in luoghi non sicuri per l’uomo.

5. **Industrial Internet of Things**

   Provvedendo ogni attore del ciclo produttivo di sensori connessi a Internet si crea una rete dinamica capace di rispondere ai cambiamenti. L’IoT non è da limitarsi alla sola produzione industriale ma deve essere applicata ad ogni processo: ad esempio, provvedere pallet e scaffali di un magazzino di sensoristica permette di tenere un tracciamento rapido e preciso dell’inventario.

6. **3D Printing**

Con la possibilità di basarsi su modelli sempre più accurati, di salvare un inventario di modelli in cloud e di sfruttare i risultati delle analisi dei dati estrapolati durante la lavorazione, la stampa 3D risulta essere più veloce e accurata, permettendo inoltre la lavorazione di materiali compositi [2].

7. **Augmented Reality**

   La realtà aumentata permette di modificare totalmente l'approccio del lavoratore al processo produttivo. È possibile, ad esempio, ottenere informazioni in tempo reale su come effettuare una riparazione, o visionare in scala reale un componente ancora in fase di progettazione, scomponendone virtualmente le parti costitutive e favorendo le attività di progettazione e di training.

8. **System Integration**

   Un'integrazione digitale completa permette un'automazione della comunicazione tra i processi, presupposto fondamentale per un'ottimizzazione complessiva del ciclo produttivo

9. **Cyber Security**

   Un sistema altamente interconnesso richiede una sicurezza informatica estremamente efficace nel bloccare eventuali attacchi che potrebbero compromettere totalmente ogni aspetto del ciclo produttivo.

Il termine Industria 4.0 è stato usato per la prima volta da Henning Kagermann durante la fiera di Hannover del 2011, nella presentazione di un progetto per l'ammodernamento del sistema produttivo tedesco. Il progetto è stato, negli anni a seguire, preso come guida da numerosi paesi, primi tra tutti Stati Uniti, Inghilterra e Giappone. A partire dal 2016, l'Italia, con il piano Impresa 4.0, ha imposto una serie di misure atte ad incentivare gli investimenti nel settore tecnologico.

Questo lavoro di tesi affronta, nello specifico, l'applicazione di algoritmi di intelligenza artificiale a dati raccolti su macchine utensili in un contesto di lavorazione industriale, dove le macchine, in servizio da circa 20 anni, sono state aumentate con diversi sensori. È interessante notare pertanto, come i concetti dell'industria 4.0 possano essere applicati senza necessariamente rivoluzionare l'assetto fisico dei sistemi, ma con piccole ma importanti integrazioni.

1.2 Macchine utensili

Le macchine utensili sono macchinari che operano per rimozione selettiva di materiale in eccesso, al fine di produrre oggetti di forma e dimensioni specifiche. Guidate da un CNC, "computer numerical control", permettono lavorazioni di altissima precisione su una vasta tipologia di materiali: vetro, tessuto, legno, plastica, metalli leggeri, polistirolo e molti altri. Nonostante l'utensile sia uno dei componenti fondamenti di questa tipologia di macchine, non possiamo definirlo un organo vero e proprio della stessa; esso, infatti, viene costantemente sostituito vista l'usura legata ai ritmi di lavorazione cui l'utensile è sottoposto; inoltre, materiali e lavori differenti richiedono utensili specifici. Sebben controllata dal software del CNC, la lavorazione può essere interrotta da guasti ed anomalie, la cui risoluzione spetta ad un operatore esterno, non essendo questa tipologia di macchine in grado di intervenire autonomamente.

Diversi sono i criteri di classificazione delle macchine utensili. Distinguiamo tra:

- **Macchine monoscopo**, come la fresatrice, il tornio, l'alesatrice
- **Macchine multiscopo** o centri di lavorazione, in grado di effettuare differenti tipi di lavorazione in un unico ciclo di lavoro e con un solo posizionamento del pezzo

Tra le macchine monoscopo, una classificazione più granulare è legata al moto di lavorazione. Si hanno:

- Macchine a moto circolare uniforme, come il tornio o il trapano
- Macchine a moto rettilineo alternato, come la piallatrice o la limatrice
- Macchine a moto speciale, come la mola
- Macchine a moto circolare variabile

Le macchine utensili possono essere dotate di differenti assi [3]; le più moderne macchine CNC, ad esempio, arrivano fino a 5 assi, ovvero l'utensile è in grado di spostarsi lungo gli assi X, Y e Z e di ruotare di 180° intorno a due di essi: ci si riferisce a queste rotazioni rispettivamente come assi A, B e C. La possibilità di inclinare e spostare l'utensile rende possibile la realizzazione di prodotti complessi che richiedono estrema precisione. La posizione lungo gli assi può avere valori positivi o negativi: più precisamente troviamo coordinate positive nel verso di allontanamento dell'utensile dal pezzo. A titolo d'esempio, in Figura 1 si ha una fresatrice a 4 assi con **mandrino** ad asse orizzontale. Sul **basamento** è presente una

colonna, il **montante**, che può scorrere lungo la direzione dell'asse del mandrino (asse Z), e la **tavola porta pezzo**. La **testa operatrice** può effettuare una traslazione lungo l'asse Y avvicinandosi o allontanandosi (direzione positiva) dal pezzo. Infine, la tavola porta pezzo può traslare rispetto al basamento lungo l'asse X, e in tal caso la direzione positiva è ricavata con la regola della mano destra.

La macchina utensile oggetto del lavoro in questa tesi è una fresatrice a 5 assi, e sarà analizzata nel dettaglio nel paragrafo 1.5.

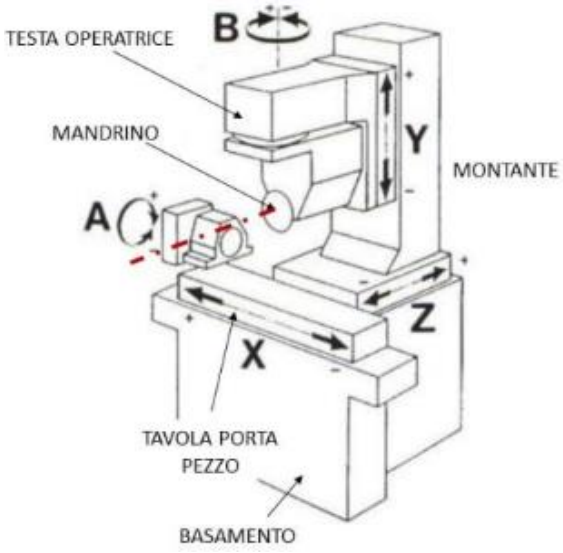
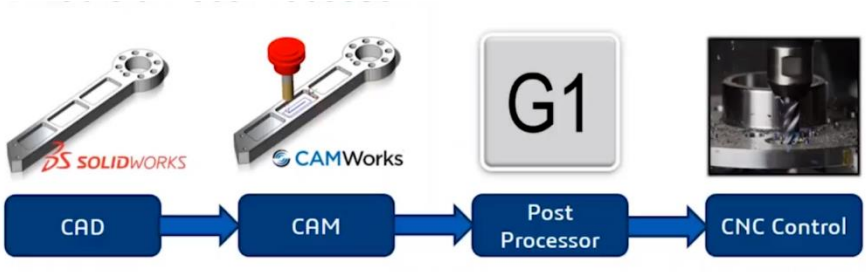


Figura 1

1.3 Controllo numerico computerizzato

Il CNC è un sistema di controllo computerizzato che permette la coordinazione dei movimenti di lavorazione della macchina utensile; grazie ad una completa automatizzazione è possibile ottenere risultati di elevatissima qualità in tempi estremamente rapidi e grande flessibilità qualora fosse necessario modificare la lavorazione. Viene quindi tolto all'operatore il controllo della macchina, al quale resta solo il compito di sorveglianza, risoluzione dei guasti e, in alcune macchine, di sostituzione dei pezzi. Generalmente le istruzioni per la generazione del prodotto finale sono fornite in formato grafico tramite software CAD: una volta realizzato il modello grafico, viene analizzato da un software CAM [4] che procede cercando eventuali errori geometrici che possano avere un impatto nel processo di lavorazione; quindi viene creato un percorso utensile ottimale, ovvero l'insieme di coordinate che l'utensile dovrà seguire durante la lavorazione; prima di essere inviate alla macchina, queste informazioni vengono tradotte grazie al post-processor [5] in un linguaggio (G-Code) interpretabile dall'unità di governo della macchina.



L'unità di governo interpreta le istruzioni tradotte dal post-processor e trasmette dei segnali ai servomeccanismi che comandano i movimenti delle tavole e della testa operatrice. Fissato quindi il punto da raggiungere, i trasduttori di posizione segnalano all'unità di governo la posizione attuale delle tavole rispetto al punto prefissato. La differenza tra la posizione attuale e quella stabilita viene elaborata per determinare la cinematica degli assi: tramite un sistema retroazionato a catena chiusa si cerca di minimizzare la differenza operando sui motori di ogni asse. In macchine utensili più semplici, come ad esempio i plotter, si preferisce utilizzare invece la catena aperta, dando fiducia alla precisione dell'elaboratore.

Sebbene i vantaggi di una macchina utensile CNC siano molteplici non sono da sottovalutare i forti costi iniziali, la licenza dei software e la necessità di disporre di figure professionali, come programmatori e operai specializzati nella manutenzione dell'elevato numero di componenti elettronici.

1.4 Usura e fine vita dell'utensile

La sostituzione dell'utensile avviene generalmente a fine vita dello stesso, ma non è rara una sostituzione anticipata legata alla rottura dello strumento o ad un'usura profonda che causa una lavorazione di scarsa qualità. L'usura è legata alle alte temperature e alla pressione sviluppate nel corso della lavorazione e può essere di diversi tipi [6]:

- l'**usura abrasiva** viene a formarsi sul dorso e sulla faccia dell'utensile a causa dello sfregamento eccessivo del truciolo rimosso che aderisce all'utensile a causa della produzione di ossidi ad elevata durezza;
- l'**usura adesiva**, tipica di materiali duttili, è invece causata da una serie di microsaldature sviluppate dal calore di frizione, che si verificano tra utensile e pezzo. Il progressivo sfregamento tra le due parti causa la frattura della parte saldata, che viene

risaldata immediatamente sul lato opposto. Ciclicamente le due parti strappano e saldano metallo in maniera reciproca e alternata, portando all'accumulo di una notevole quantità di detriti (specie se presente del lubrificante) che, per sfregamento, possono portare alla rottura dell'utensile e alla rovina del pezzo.

Con "fine vita" si intende il limite massimo che l'utensile può sopportare: può essere un limite temporale, un limite alla quantità di materiale asportato o più semplicemente il raggiungimento di una perfetta lavorazione di un certo numero fissato di pezzi. La durata di un utensile è influenzata dalla tipologia e dalla velocità di lavorazione, dal materiale lavorato, dal materiale dell'utensile e dalla sua geometria. Generalmente, il fattore che influenza maggiormente l'usura è la velocità di taglio V_t, strettamente correlata alla temperatura nella zona di taglio: fissato pertanto un limite all'usura del dorso dell'utensile, all'aumentare del tempo di taglio l'usura presenta il seguente andamento:

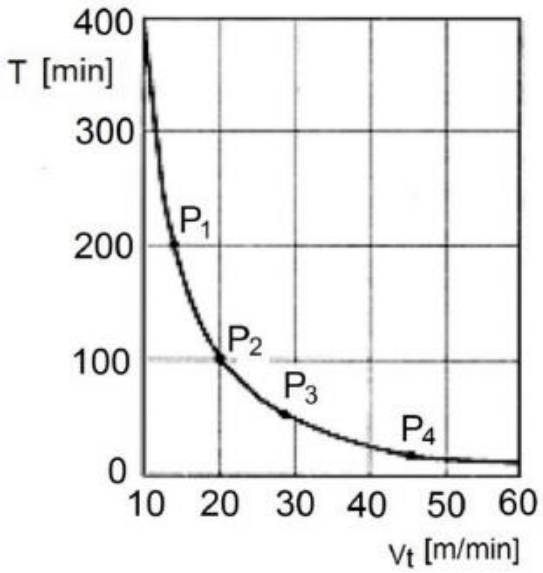


Figura 2

Come si può osservare nella Figura 2, all'aumentare della velocità di taglio la vita utile dell'utensile T diminuisce drasticamente secondo un andamento iperbolico [7].

Una stima sulla vita utile dell'utensile **T** in minuti è data dalla legge di Taylor generalizzata:

$$v_t \cdot T^n \cdot a^r \cdot p^s = C$$

da cui

$$T = C^{\frac{1}{n}} \cdot V_t^{-\frac{1}{n}} \cdot a^{-\frac{r}{n}} \cdot p^{-\frac{s}{n}}$$

dove:

- **V_t**: velocità di taglio [m/min],
- **n**: sensibilità alla durata della velocità di taglio
- **C**: costante pari alla V_t corrispondente alla durata dell'utensile di un minuto
- **a**: avanzamento [giri/min]
- **p**: profondità di passata [mm]
- **r**: sensibilità della durata all'avanzamento.
- **s**: sensibilità della durata alla profondità di passata

La costante C può essere ottenuta dall'intersezione della curva in Figura 2 con l'asse delle ascisse.

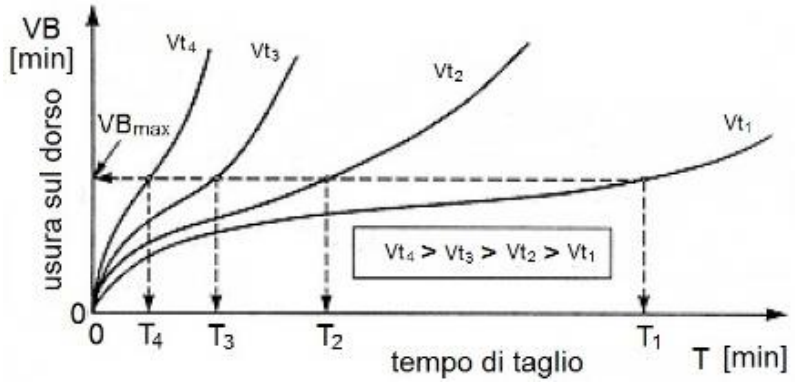


Figura 3

Come già accennato, la vita utile può anche però essere espressa in numero di pezzi lavorati o quantità di materiale asportato; nel caso in esame la vita utile è stata fissata a 300 pezzi lavorati.

1.5 Lavorazione nel caso d’uso

Come già accennato, la macchina utensile in esame è una fresatrice a 4 assi che lavora su dei pezzi di ghisa. Essa è dotata di un controllore numerico FANUC da cui vengono estratte informazioni sugli assi tra cui la corrente assorbita e la posizione, la velocità del mandrino, e di un misuratore di potenza assorbita. È dotata inoltre di un PLC, *programmable logic computer*, che registra gli eventi di automazione della macchina. Maggiori informazioni sulle variabili estratte saranno riportate nel capitolo 2.1. La macchina utensile è in grado di lavorare contemporaneamente 6 pezzi, mentre un altro pallet viene caricato in un’area di carico/scarico adiacente a quella di lavorazione, come in figura 4.

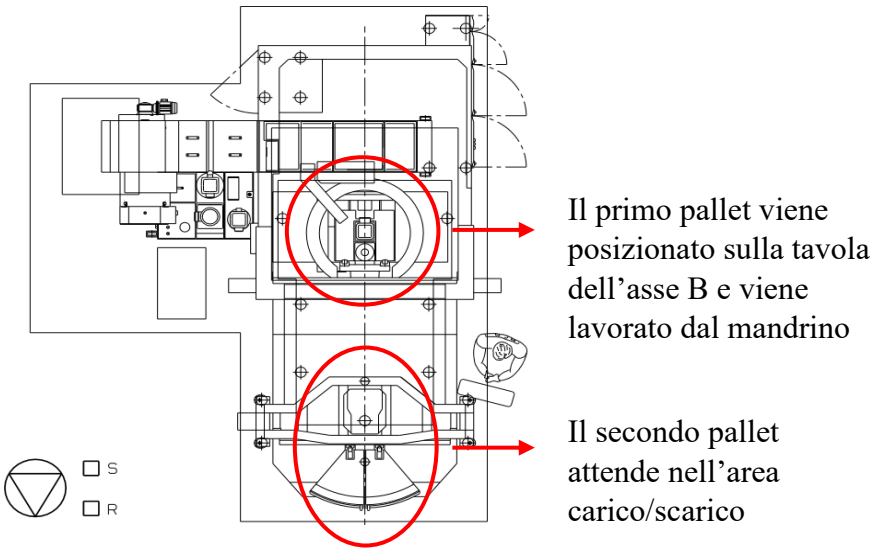


Figura 4

I pezzi nel pallet nell’area di lavorazione vengono lavorati uno ad uno asportando il sovramateriale dalla faccia anteriore; terminato l’ultimo pezzo la tavola sull’asse B ruota di 180° e la lavorazione prosegue sulla faccia posteriore. Al termine della lavorazione le due facce, inizialmente identiche, risulteranno differenti, dal momento che i programmi di lavorazione sono diversi.

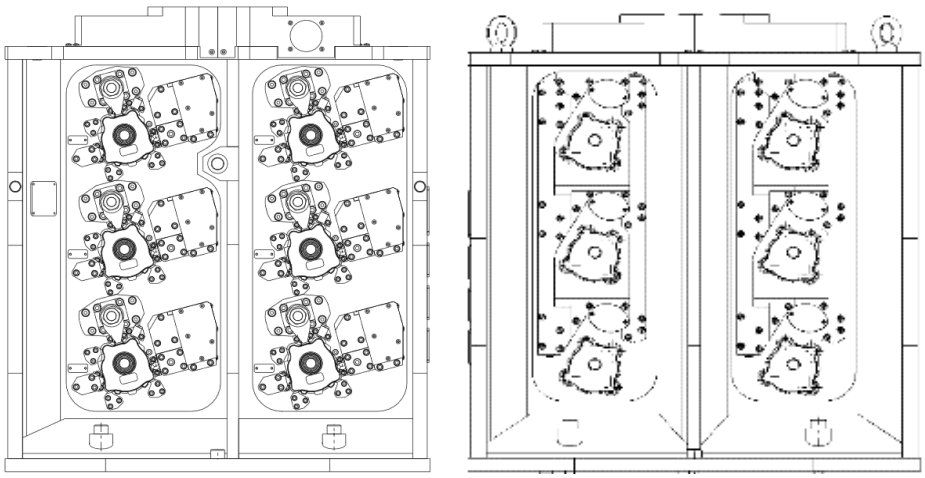


Figura 5 - Faccia anteriore (a sinistra) e posteriore del pallet

Per ogni faccia, ogni pezzo è individuato da uno slot. Una lavorazione corretta opera nel seguente ordine:

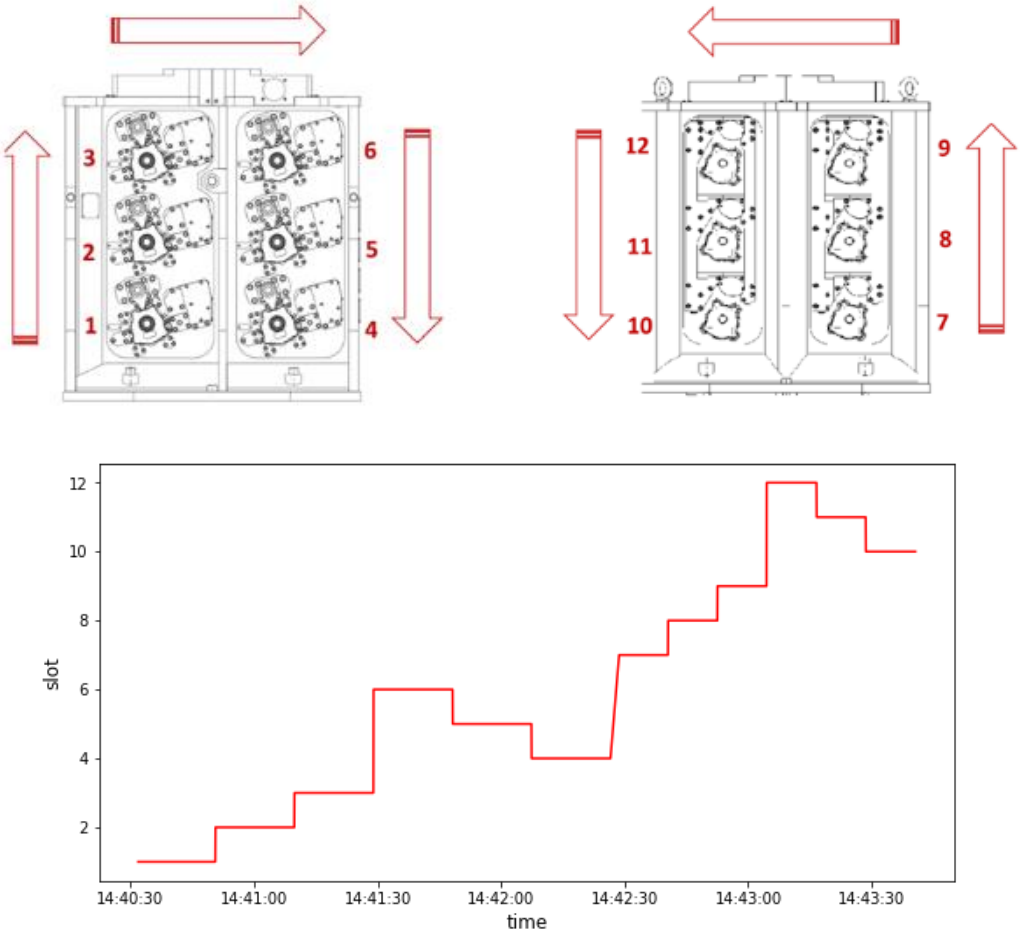


Figura 6 - variazione degli slot nel tempo

Come si vedrà nel capitolo 3.2, quest'ordine non è sempre rispettato, con la conseguenza che alcune pezzi vengono lavorati più volte sulla stessa faccia. Generalmente, ogni circa 65 pezzi si effettua un controllo qualitativo a campione sull'output dell'ultima lavorazione: se anche uno solo dei parametri tra quelli stabiliti risulta essere fuori soglia, la lavorazione viene interrotta e un controllo straordinario viene effettuato sulla macchina. Ad esempio, se sul pezzo risultano esserci delle frastagliature l'intero pallet viene etichettato come non conforme e quindi scartato; si effettua pertanto la sostituzione dell'utensile.

1.6 Maintenance Management policies

La rottura di un componente interno o più banalmente il disallineamento dei pezzi sul tavolo di lavoro portano inevitabilmente al blocco del ciclo di lavorazione e, di conseguenza, al rallentamento dell'intero ciclo produttivo e ad un costo. Una corretta gestione delle operazioni di manutenzione può pertanto limitare guasti inattesi e limitare le interruzioni ai soli controlli di routine e agli eventuali interventi. Diverse sono le politiche applicabili [8], qui elencate per ordine di efficienza e complessità crescente:

- **Run to Failure**: si stratta dell'approccio più banale (e più frequentemente adottato), nel quale l'intervento di manutenzione viene effettuato solo nel momento in cui si verifica un guasto o un'interruzione non voluta della lavorazione. Il costo e il tempo di interruzione sono più alti rispetto ad altre politiche, non avendo in anticipo alcuna informazione sul tipo di guasto cui la macchina andrà incontro.
- **Preventive Maintenance**: con questo approccio le manutenzioni vengono calendarizzate, effettuando un intervento dopo un certo tempo di lavorazione o in seguito alla lavorazione di un certo numero di pezzi. In questo modo è possibile ridurre il costo andando ad eseguire controlli approfonditi durante le interruzioni volute, ma non è possibile conoscere in anticipo quali tipi di guasti si presenteranno. Inoltre, la lavorazione viene interrotta anche nei casi cui un intervento correttivo non era necessario, con la conseguenza di uno spreco di risorse evitabile.
- **Predictive Maintenance**: sfruttando i dati raccolti dalla macchina in lavorazione è possibile predire con un certo anticipo il verificarsi di un'interruzione nella lavorazione,

ed applicare un intervento di manutenzione ad hoc. Questo metodo è il più complesso ma anche il più efficiente.

1.7 Predictive Maintenance

La manutenzione predittiva può essere realizzata seguendo differenti approcci, ad esempio mediante l'analisi di un modello fisico che descriva al meglio il comportamento del sistema in analisi; la creazione di un buon modello è un problema che richiede conoscenze profonde sul funzionamento del sistema e di come le variabili in gioco possano andare ad influenzarne il comportamento; maggiore sarà la complessità del modello maggiore sarà l'accuratezza. Se il modello è realizzato correttamente, fornendo degli input il modello fornirà degli output attendibili che potranno essere analizzati con metodi statistici; il problema di questi modelli, oltre la loro complessità, risiede nella loro specializzazione: una volta creato un modello esso non potrà essere riutilizzato per macchine differenti o con condizioni operative differenti (ad esempio l'utilizzo della stessa macchina con un utensile diverso o un materiale diverso). Vista pertanto la dinamicità di utilizzo delle macchine utensili questo metodo non risulta adatto; come già accennato, nell'Industria 4.0 l'ambiente di lavorazione è integrato ad una serie di sensori che raccolgono costantemente dati durante tutte le fasi: un approccio migliore alla manutenzione predittiva risulta quindi essere di tipo data-driven, e quindi basato sull'apprendimento. L'enorme mole di dati viene riorganizzata, pulita e quindi integrata con uno storico degli interventi di manutenzione e la causa di tali interventi; in questo modo è possibile avere i dati strutturati temporalmente e analizzare il comportamento delle variabili tra un intervento e il successivo. Individuando quindi eventuali pattern nei dati e variazioni di tali schemi si può prevedere, con un certo anticipo e con certo livello di accuratezza, quando la macchina si romperà: in tal modo si potranno effettuare interventi di manutenzione mirati allo specifico problema, con la conseguenza di interrompere la lavorazione solo per vera necessità (a differenza della politica *preventive*) con l'effetto di minimizzare i costi. Gli algoritmi di apprendimento per la manutenzione predittiva possono essere di diverso tipo:

- **Supervised Learning:** l'apprendimento supervisionato è la tipologia più utilizzata nella manutenzione predittiva, vista la disponibilità di dati storici. Questa modalità consiste nel fornire ad un modello di machine learning un insieme di dati, detto *training set*, per

i quali si conosce già il valore della variabile che si vorrebbe predire; se la variabile da predire è una label si parla di classificazione. Tramite l'applicazione di un algoritmo di machine learning, il modello viene addestrato a riconoscere la relazione tra le label e i dati. Fornendo quindi al modello generato nuovi dati non etichettati, esso sarà in grado di effettuare una classificazione, producendo in output le label adeguate. Quest'approccio sarà maggiormente analizzato nel capitolo 4.

- **Unsupervised Learning:** l'apprendimento non supervisionato non richiede che i dati forniti presentino già la variabile da predire; tramite algoritmi di machine learning il modello sarà invece in grado di generare dei cluster a partire dai dati, quindi a individuare caratteristiche comuni tra le variabili. Nella manutenzione predittiva quest'approccio risulta essere utile nell'individuare comportamenti analoghi delle variabili in prossimità di interventi di manutenzione differenti.

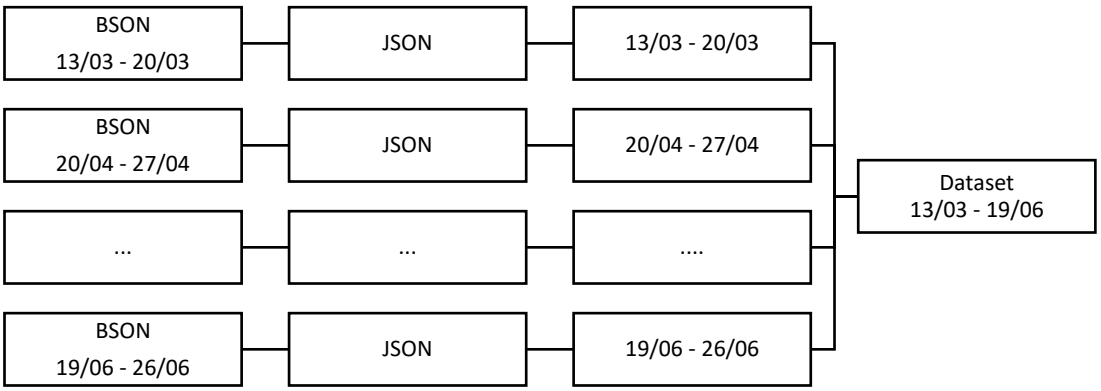
Capitolo 2

Nel seguente capitolo analizzeremo nel dettaglio l'acquisizione e la struttura dei dati e le operazioni di aggregazione per la generazione del dataset.

2.1 Acquisizione e struttura dei dati

La macchina utensile è affiancata da un'unità di elaborazione dedicata, detta Recorder, sviluppata dall'azienda. Il *recorder* è essenzialmente un dispositivo per l'edge computing: l'acquisizione e una prima elaborazione dei dati avvengono in prossimità del luogo in cui questi dati vengono generati, per ridurre notevolmente il tempo di latenza in elaborazione e il traffico sulla rete. Quindi, dopo un breve tempo di permanenza nei buffer di acquisizione, i dati grezzi vengono filtrati, pre-analizzati, strutturati ed esportati tramite protocollo MQTT e quindi salvati in Cloud. Le variabili acquisite dal CNC sono relative al mandrino e agli assi; è presente, inoltre, un accelerometro che registra le vibrazioni del mandrino lungo le tre dimensioni; il power-meter invece registra l'energia in ingresso e l'assorbimento di corrente della macchina in maniera continua. Il PLC, infine, salva gli eventi di automazione, rappresentati da vettori binari: ogni qualvolta si verifica un evento, ad esempio lo sblocco del mandrino, viene innescata la registrazione.

Per le finalità di questo lavoro di tesi, le variabili prese in considerazione sono solo quelle acquisite dal CNC e dall'accelerometro, in dettaglio nella *tabella 1*. I dati raccolti sono stati aggregati in file BSON contenenti registrazioni relative ad un'unica settimana, e salvati nel database MongoDB. Scaricati i file, è stata effettuata una conversione in formato JSON e una successiva trasformazione della loro struttura, come vedremo nel dettaglio più avanti. Infine, i dataset risultanti, relativi alle singole settimane, sono stati concatenati in un unico dataset.



I dati utilizzati in questo lavoro sono stati registrati per un periodo di 4 mesi. I file contenenti i dati su mandrino ed assi, a cui ci riferiremo in seguito col nome di *working step*, sono separati da quelli contenenti i dati sull'accelerometro, provenendo da fonti diverse. In Tabella 1 un dettaglio delle variabili.

Tabella 1 – dettaglio delle variabili

Variabili	**Unità di misura**	**Data Type**	**Sampling rate (ms)**	**Descrizione**
X_AXIS_CURR	A	double	48	Corrente assorbita dall'asse X
Y_AXIS_CURR	A	double	48	Corrente assorbita dall'asse Y
Z_AXIS_CURR	A	double	48	Corrente assorbita dall'asse Z
B_AXIS_CURR	A	double	48	Corrente assorbita dall'asse B
X_AXIS_POS	mm	double	48	Posizione dell'asse X
Y_AXIS_POS	mm	double	48	Posizione dell'asse Y
Z_AXIS_POS	mm	double	48	Posizione dell'asse Z
B_AXIS_POS	mm	double	48	Posizione dell'asse B
X_AXIS_DELTAPOS	mm	double	48	Errore d'inseguimento dell'asse X
Y_AXIS_DELTAPOS	mm	double	48	Errore d'inseguimento dell'asse Y
Z_AXIS_DELTAPOS	mm	double	48	Errore d'inseguimento dell'asse Z
B_AXIS_DELTAPOS	mm	double	48	Errore d'inseguimento dell'asse B
SP_SPEED	rpm	double	48	Velocità di rotazione del mandrino
SP_CURR	A	double	48	Corrente assorbita dal mandrino
SLOT	-	int	48	Slot in lavorazione
PALLET	-	Int	48	Pallet in lavorazione
tmp	C°	float	48	Temperatura del basamento
timeStamp_ws	ms	long	48	timeStamp assi e spindle
LINE_NUM	-	int	48	Codice relativo alle azioni comandate dal CNC alla macchina
velMOD_RMS	mm/s	double	100	Modulo della velocità di vibrazione dei cuscinetti del mandrino nelle 3 dimensioni
timeStamp_acc	ms	long	100	timeStamp accelerometro

2.2 Creazione dei dataset a partire dai file JSON

Analizzando uno dei file JSON del working step in forma tabellare possiamo osservare alcune particolarità delle registrazioni:

timeStamp	**sensor**	**unit**	**sampling**	**dataObj**	**…**
{'$date': {'$numberLong': '1592797884289'}}	slot	-	{'$numberInt': '48'}	{'n': {'$numberInt': '0'}, 'data': [{'$numberD..	…
{'$date': {'$numberLong': '1592797884289'}}	X_AXIS_CURR	A	{'$numberInt': '48'}	{'n': {'$numberInt': '0'}, 'data': [{'$numberD...	…
{'$date': {'$numberLong': '1592797884289'}	Y_AXIS_CURR	A	{'$numberInt': '48'}	{'n': {'$numberInt': '0'}, 'data': [{'$numberD...	…
…	…	…	…	…	

Il campo timeStamp contiene un valore comune a tutti i sensori; inoltre, i dati veri e propri delle misurazioni sono contenuti nel campo *dataObj*. A titolo d'esempio osserviamo il contenuto di questo campo di uno dei sensori:


```
"dataObj":{
   "n":{"$numberInt":"0"},
   "data":[
      {"$numberDouble":"190000.0"},
      {"$numberDouble":"0.0"},
      {"$numberDouble":"0.0"},
      {"$numberDouble":"0.0"},
      {"$numberDouble":"-169.0"},
      {"$numberDouble":"-1002.0"},
      {"$numberDouble":"-2381.0"},
                  ……..
   ]
}
```


Il campo *dataObj* è costituito da due sottocampi:

- **n**: indica il numero di buffer riempiti (la capienza massima è 10000 valori)

- **data**: contiene i dati delle misurazioni

Per tutti i sensori che condividono lo stesso valore di *timeStamp*, il valore **n** e la lunghezza del campo **data** sono costanti. Questo suggerisce che l'inizio e la fine delle misurazioni è uguale per tutti i sensori, e il campo *timeStamp* ne indica l'inizio. Per costruzione, questi file sono *sensor-oriented*; per un'opportuna analisi sarà necessario renderli *time-oriented* sfruttando il *timeStamp*, il sampling rate e la lunghezza del campo **data**.

La trasformazione avviene nei seguenti passi:

1. Si effettua un'operazione di raggruppamento per *timeStamp*
2. In ciascun gruppo, per ogni sensore, si accede al campo *dataObj* si itera sui valori del campo *data*. Ciascun valore viene convertito nel formato opportuno e salvato in un vettore predisposto
    a. Se sono presenti dei buffer (campo *n* > 0) si effettua un raggruppamento anche per sensore e i vettori generati vengono concatenati lungo l'asse 0
3. Si crea un vettore di timeStamp della stessa lunghezza di *data*, il cui primo valore è il valore di timeStamp del gruppo e i successivi sono dati dal valore precedente a cui è aggiunto il valore di sampling rate
4. I vettori generati si concatenano lungo l'asse 1 creando un DataFrame relativo al gruppo.
5. Infine, si concatenano i dataframe dei gruppi lungo l'asse 0

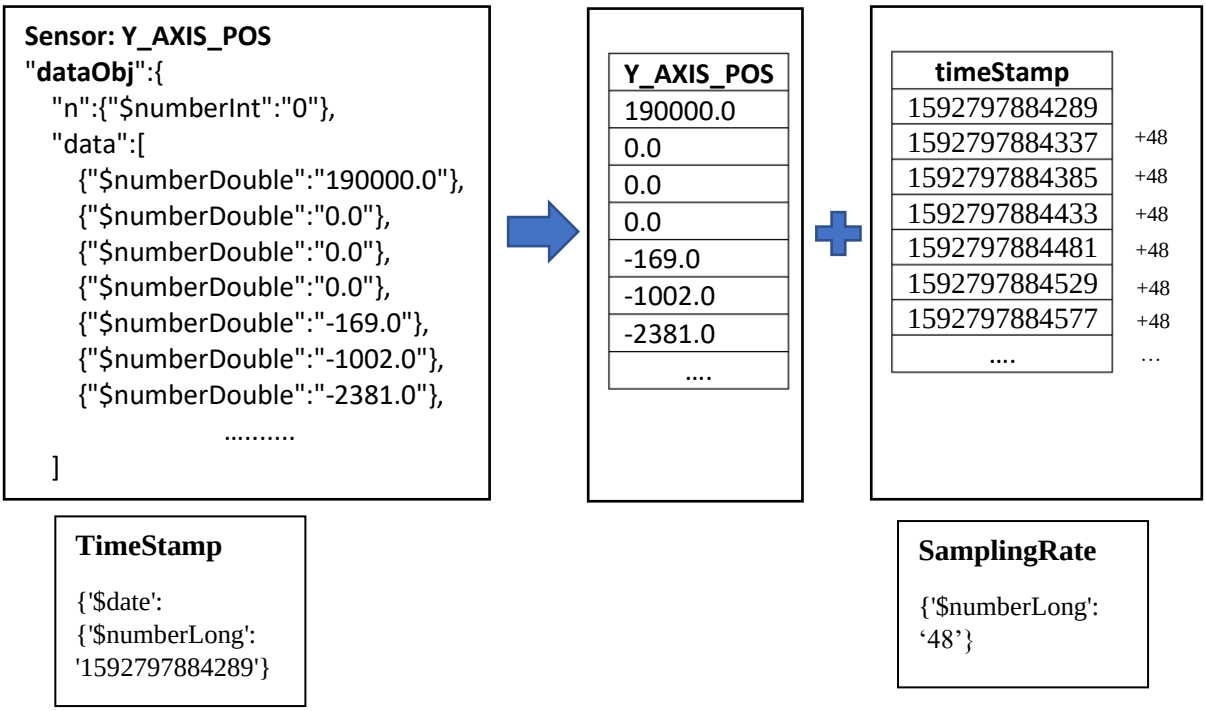


Figura 7- Creazione di un vettore di dati della variabile Y_AXIS_POS e del corrispettivo vettore di timeStamp in funzione del timeStamp iniziale

Il risultato ottenuto ha la seguente forma:

timeStamp	dateTime	X_AXIS_CURR	...	X_AXIS_POS	SP_SPEED	SP_CURR
1595217924074	2020-07-20 04:05:24.074	0.5490	...	0.0	5799.0	0.0
1595217924122	2020-07-20 04:05:24.122	0.5612	...	0.0	5799.0	0.0
1595217924170	2020-07-20 04:05:24.170	0.6467	...	0.0	5799.0	0.0
1595217924218	2020-07-20 04:05:24.218	0.5368	...	0.0	5799.0	0.0
1595217924266	2020-07-20 04:05:24.266	0.4636	...	0.0	5800.0	0.0
...	...	...	...	...	...	...
1595554830834	2020-07-24 01:40:30.834	-0.3173	...	28540.0	5800.0	0.0
1595554830882	2020-07-24 01:40:30.882	-0.2563	...	28540.0	5799.0	0.0
1595554830930	2020-07-24 01:40:30.930	-0.2563	...	28540.0	5799.0	0.0
1595554830978	2020-07-24 01:40:30.978	-0.3294	...	28540.0	5799.0	0.0
1595554831026	2020-07-24 01:40:31.026	-0.2928	...	28540.0	5800.0	0.0

Figura 8 - Esempio di dataset ottenuto dalle trasformazioni sul file JSON

Come si può osservare, il *timeStamp* di ogni riga è distanziato dalla successiva di 48 millisecondi, ovvero dal tempo di campionamento usato nel working step. La colonna *dateTime* è stata aggiunta per avere un maggiore orientamento nella consultazione del dataset.

Il dataset in figura 8 è stato generato dalla trasformazione di un unico file JSON; il numero totale di righe è di poco più di un milione. Non tutti i file JSON contengono tuttavia un numero di misurazioni così elevato, considerando che le registrazioni non sono state costanti e che durante il periodo che coprono i dati, l'intero processo di acquisizione e salvataggio era ancora in fase di rifinimento. Il dataset completo, ottenuto concatenando tutti dataset generati con questa tecnica a partire dai file JSON in un periodo di 4 mesi, ha un numero di righe pari a circa 8,5 milioni.

2.3 Inserimento degli eventi in macchina

Con eventi in macchina ci riferiamo essenzialmente agli eventi di cambio utensile e agli eventi di manutenzione in seguito ad un guasto. La registrazione di questi è inserita manualmente dagli operatori, con la conseguenza che le informazioni riportate sono spesso non attendibili, mancanti o soggette a libera interpretazione: ad esempio, l'orario di inizio o di fine di un evento può essere, a causa della dimenticanza dell'operatore, inserito in seguito alla risoluzione dell'evento, e pertanto del tutto approssimativo; le descrizioni degli eventi, a parità di eventi, possono invece essere diverse, rendendo difficile effettuare, ad esempio, un'analisi statistica sui tipi di guasti che avvengono durante la lavorazione. Per questo motivo, un algoritmo di clustering può essere utile per individuare pattern comuni di variazione delle variabili a ridosso di eventi di manutenzione apparentemente differenti.

A titolo d'esempio, nella tabella 2 sono riportate alcune colonne prese dal file in cui sono stati registrati gli eventi di manutenzione.

Tabella 2 - Esempio di tabella degli eventi di manutenzione.

Data inizio	**Ora inizio**	**Data Fine**	**Ora Fine**	**Testo Guasto**	**Codice guasto**	**costi**	**…**
01/04/2020	08:45:06	01/04/2020	08:45:06	non si abilita Pallet grezzo	0019	20,10	…
03/04/2020	21:02:51		00:00:00	anomalia precarico mandrino		20,10	…
09/04/2020	09:08:13	09/04/2020	09:38:13	all.2018 precarico mandrino	0001	40,20	…
15/04/2020	11:43:41	15/04/2020	12:40:10	anomalia traslatore utensile		40,20	…
22/04/2020	03:52:06	22/04/2020	03:54:27	limite 1 e 2 accelerometro mandrino		26,78	…
06/05/2020	15:52:42	06/05/2020	19:56:15	si blocca	0021	40,21	…
04/06/2020	00:42:04	04/06/2020	01:08:46	in zona carico scarico		60,32	…

Come si nota dalla tabella, alcune celle sono vuote causa dimenticanza degli operatori; inoltre, alcune celle di testo guasto sono del tutto approssimative, come ad esempio la dicitura "si blocca".

I guasti sono (o dovrebbero essere) più rari dei cambi utensile; come già discusso nel capitolo 1.4, la sostituzione di un utensile avviene per usura o per fine vita dello stesso. Nella lavorazione

in esame, il valore di fine vita è fissato a 300 pezzi lavorati, il che significa circa un cambio utensile al giorno. Come si osserva dalla tabella 3, come per i guasti, non è fissato uno standard per la registrazione dell'evento di sostituzione.

Tabella 3 - Esempio di tabella degli eventi di sostituzione utensile

Ora	**Data**	**Descrizione**	**Note**	**Lotto**	**Causa**	**…**
06:41:28	01.06.2020	fine vita utensile		89007074	Fine vita	…
13:42:25	01.06.2020	usura utensile	t30	89007074	usura	…
10:08:28	02.06.2020	ok continuazione produzione (MOE), sostituzione utensile	T 30	89007074	usura	…
00:46:33	03.06.2020	sostituzione utensile, usura utensile	306930	89007074	usura	…
12:36:28	03.06.2020	ok continuazione produzione (MOE), sostituzione utensile	T 30	89131421	usura	…
01:48:20	04.06.2020	sostituzione utensile, usura utensile	306998	89131421	usura	…

Le operazioni di registrazione, a causa della mole dei dati, non avvengono sempre: pertanto alcuni eventi in macchina presenti nei file possono non essere avvenuti durante le registrazioni, e quindi non utilizzabili coi dati a disposizione.

Nell'effettuare l'inserimento degli eventi nel dataset è stato utilizzato il timeStamp di ogni registrazione nel seguente modo:

1. Gli orari degli eventi sono convertiti da locale a UTC (Tempo universale coordinato)
2. Data e orario d'inizio di ogni evento vengono uniti e convertiti nel formato dei timeStamp delle registrazioni (Unix Epoch Time)
3. Ogni evento viene associato alla riga del dataset il cui timeStamp è il più vicino possibile (con vincolo di precedenza) al timeStamp dell'evento, fissando un limite temporale di un giorno.

L'imposizione di un limite temporale è necessaria per scartare tutti gli eventi che sono avvenuti al di fuori delle registrazioni.

Nell'esempio seguente (figura 9) è stata utilizzata la colonna dateTime per maggiore chiarezza. Come si può osservare l'evento del 2020-03-23 00:53:28 è stato assegnato alla riga del dataset con dateTime 2020-03-23 00:51:59.246, ovvero la più vicina all'evento. Notiamo infatti che la riga seguente è relativa ad una misurazione avvenuta alle 01:08:00.351, quindi successiva all'evento.

…	X_AXIS_CURR	SP_CURR	dateTime
…	-0.4026	0.0	2020-03-23 00:51:59.150
…	-0.3661	0.0	2020-03-23 00:51:59.198
…	-0.3032	0.0	2020-03-23 00:51:59.246
…	-0.3661	0.0	2020-03-23 01:08:00.351

…	Testo Guasto	dateTime
…	Puls.grezzo	2020-03-16 13:13:29
…	Anomalia traslatore	2020-03-17 08:29:37
…	Time over controllo integrità utensile	2020-03-22 12:53:56
…	Blocco sblocco utensile su mandrino	2020-03-23 00:53:28
…	Non si abilita il pallet grezzo	2020-04-02 06:04:34

Figura 9 - Esempio di assegnamento di un evento al dataset

Al dataset vengono quindi aggiunte le colonne dell'evento; come si può notare nella figura 10, esse avranno valore nullo in tutte quelle righe per le quali non si è verificata l'associazione sopra illustrata. Viene inoltre aggiunta una variabile booleana per ciascun evento.

dateTime	X_AXIS_CURR	guasto	...	Testo priorità	Testo guasto	Costi generali
2020-06-17 08:22:46.710	-0.3294	False	...	None	None	NaN
2020-06-17 08:22:46.758	-0.3538	False	...	None	None	NaN
2020-06-17 08:22:46.806	-0.3904	True	...	without priorities	quando porta fuori il pallet 1pos male	160.83
2020-06-17 08:38:46.220	-0.3416	False	...	None	None	NaN
2020-06-17 08:38:46.268	0.5490	False	...	None	None	NaN

Figura 10 – Esempio di arricchimento del dataset con le informazioni prese dal guasto

È infine da evidenziare che, essendo gli eventi registrati manualmente ed essendo associati al timeStamp più vicino, con una differenza di qualche minuto o nei casi peggiori di ore, la posizione temporale dell'evento è puramente indicativa, è pertanto bene tenere in considerazione un intervallo temporale precedente al timeStamp associato come periodo in cui si è, con una buona probabilità, verificato l'evento.

2.4 Inserimento dell'accelerometro

L'inserimento dei dati dell'accelerometro nel dataset viene effettuato con dinamiche analoghe a quelle viste nel paragrafo precedente. Come già discusso, l'accelerometro ha una frequenza

di campionamento di 100 ms, più bassa rispetto al working step, campionato a 48 ms: a causa di questa differenza non tutte le righe del dataset risultante avranno i dati dell'accelerometro (variabile **velMOD_RMS**), come si può osservare in figura 11.

dateTime	X_AXIS_CURR	...	SP_SPEED	velMOD_RMS
2020-06-09 07:13:54.466	0.0000	...	5816.0	0.2836
2020-06-09 07:13:54.514	0.6832	...	5810.0	NaN
2020-06-09 07:13:54.562	0.6467	...	5810.0	0.2992
2020-06-09 07:13:54.610	0.7687	...	5806.0	NaN
2020-06-09 07:13:54.658	0.6467	...	5804.0	0.3147

Figura 11 - esempio di dataset risultate dall'unione tra working step e accelerometro

Capitolo 3

In questo capitolo verrà effettuata un'analisi del comportamento di alcune variabili durante la lavorazione; verrà quindi effettuata una divisione in cicli di lavorazione e una distinzione tra cicli validi e cicli invalidi. Alla fine del capitolo verrà quindi illustrata la creazione di un dataset ridotto, il quale sarà utilizzato nell'ultimo capitolo per addestrare un modello di machine learning.

3.1 Analisi del comportamento di alcune variabili

Tra le variabili a disposizione, quelle che permettono di osservare maggiormente l'effetto dell'usura dell'utensile sono le variabili del mandrino. Analizziamo quindi il loro comportamento durante più cicli di lavorazione, tra un cambio utensile e il successivo. Come già discusso nel capitolo 1.5, la macchina lavora un pallet alla volta: ogni pallet è costituito da 6 pezzi ed ogni pezzo viene lavorato su entrambe le facce con programmi di lavorazione differenti; definiamo la lavorazione totale dei 6 pezzi *ciclo.* Come si vede in figura 12 e 13, in ogni slot la corrente assorbita dal mandrino ha lo stesso andamento. Il picco iniziale, nel primo slot, è dovuto all'avvio del mandrino.

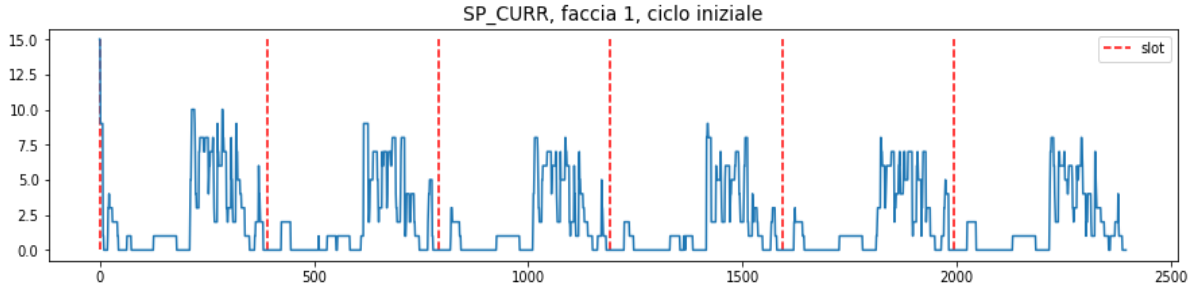


Figura 12

Nei cicli successivi, al proseguire della lavorazione, si osserva un aumento costante della corrente assorbita dal mandrino a causa dell'usura dell'utensile.

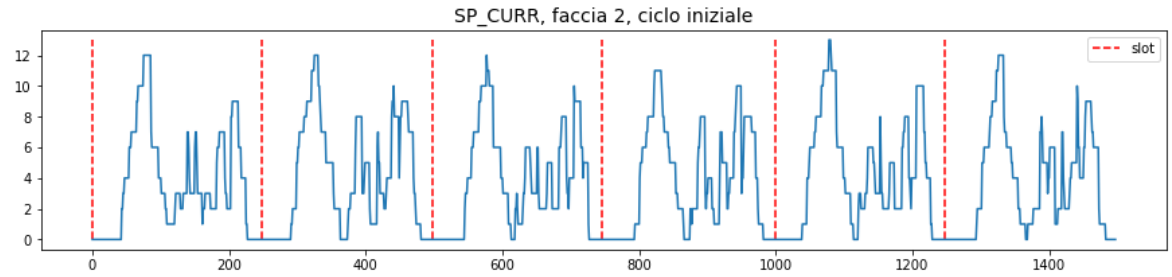


Figura 13

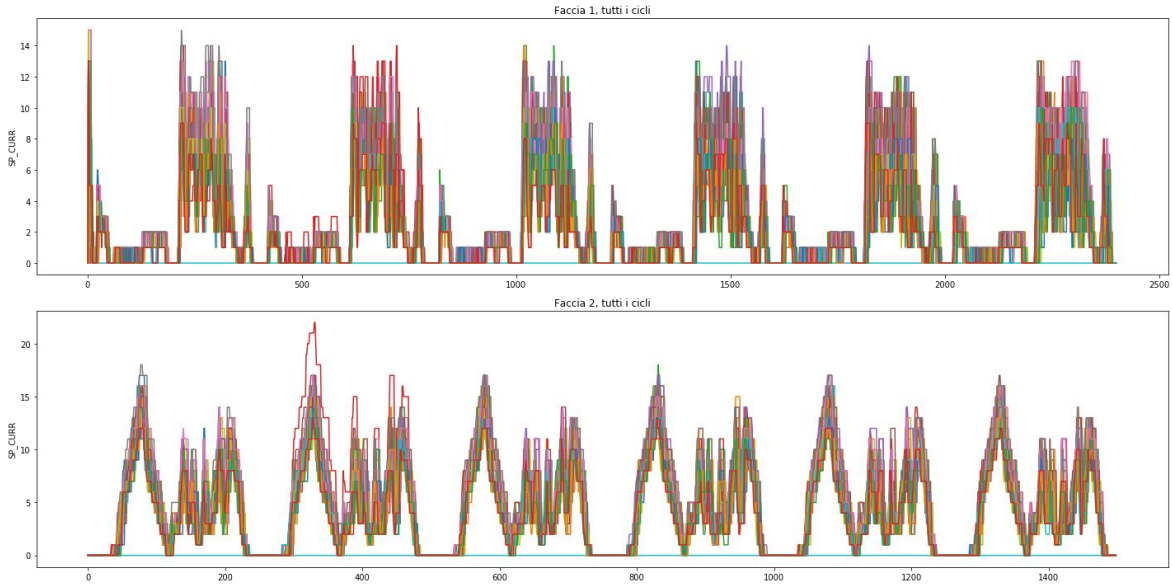


Figura 13 - Corrente assorbita dal mandrino in tutti i cicli tra due cambi utensile

Questo comportamento è maggiormente rilevabile osservando l'andamento medio nei vari cicli (figure 14 e 15).

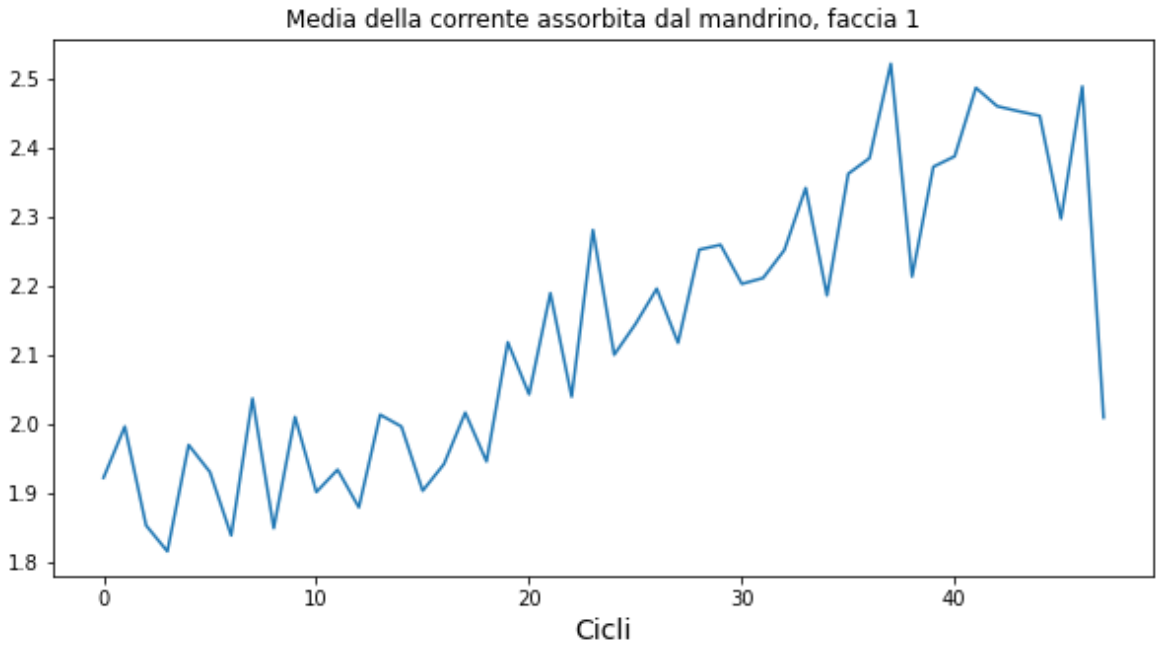


Figura 14

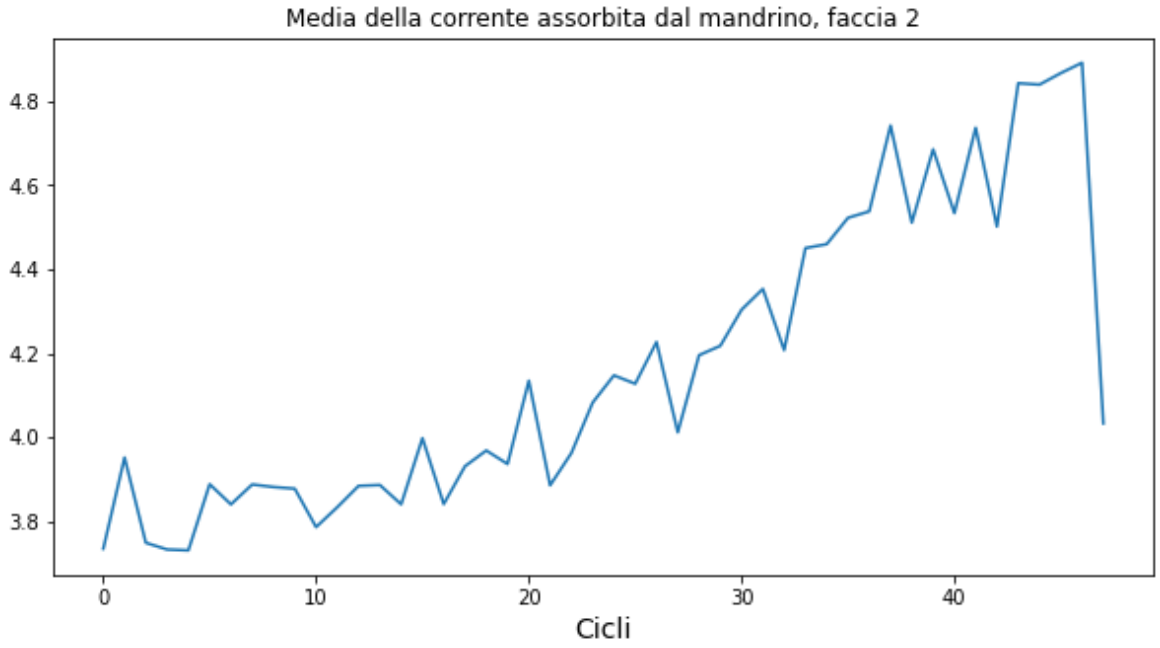
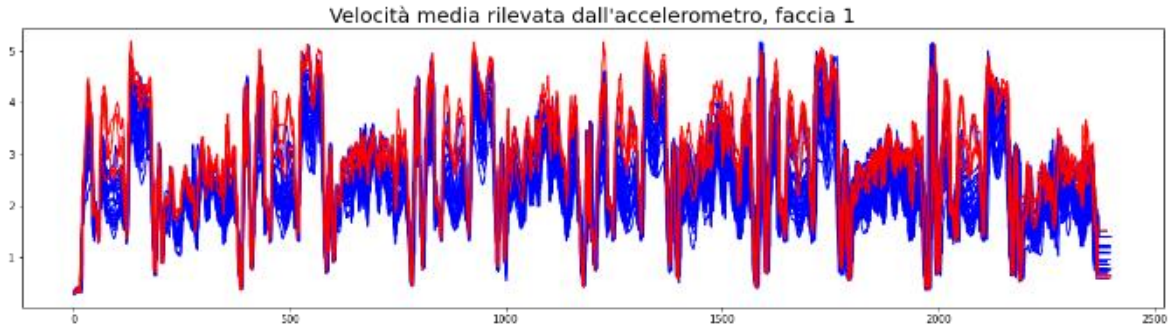
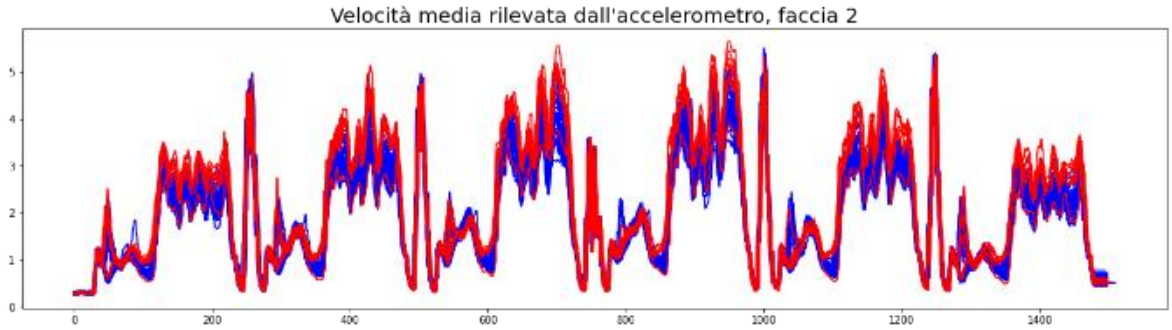


Figura 15

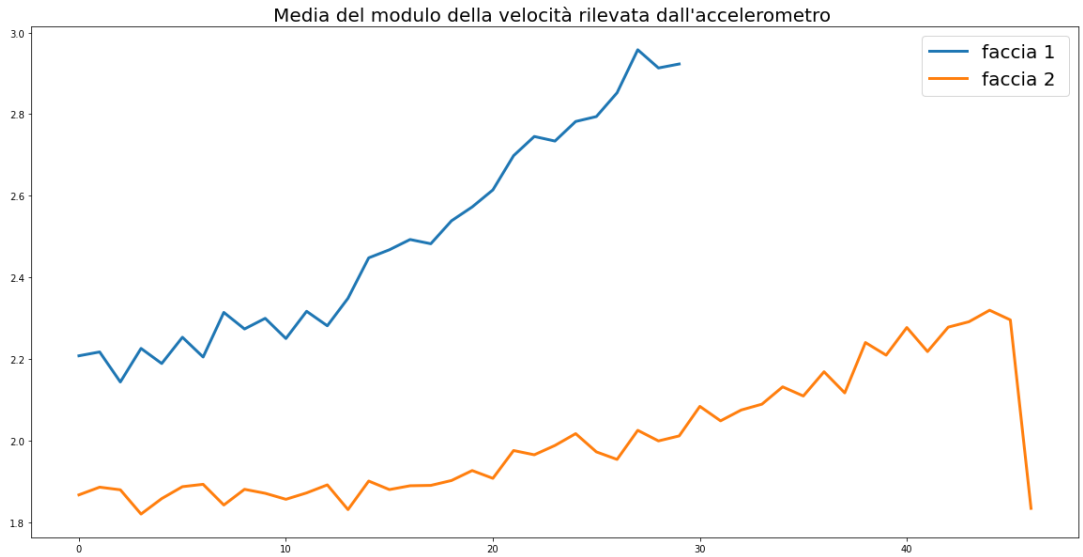
Come si può notare, la media in entrambe le facce aumenta progressivamente fino ad una drastica diminuzione: la causa di tale diminuzione è da ricercarsi in una probabile sostituzione dell'utensile, avvenuta prima della segnalazione riportata manualmente nel file dei cambi utensile.

La velocità media rilevata dall'accelerometro ha un comportamento analogo. Nell'immagine seguente sono colorati in rosso gli ultimi 5 cicli.

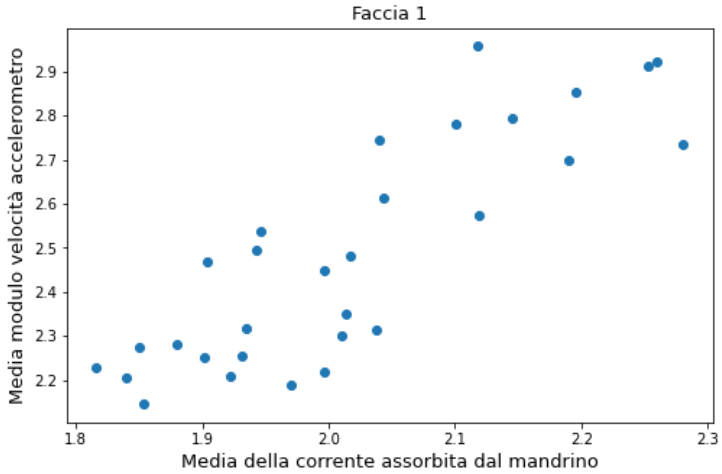




Come già visto per la corrente assorbita dal mandrino, anche la media dei valori dell'accelerometro tende ad aumentare al procedere dei cicli di lavorazione.

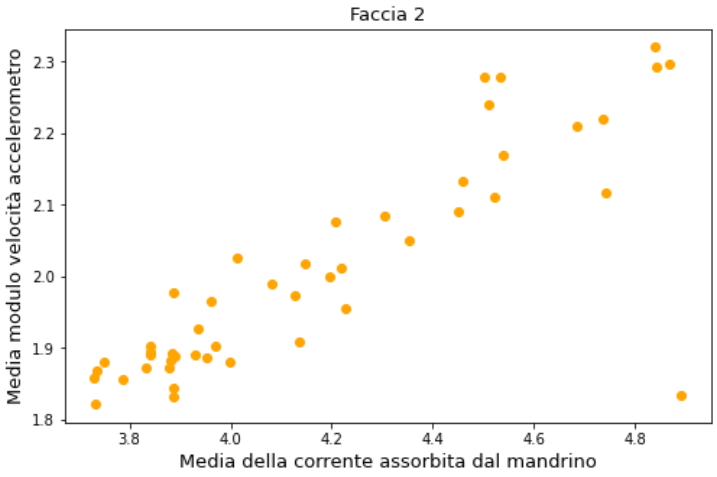


In particolare, mettendo insieme la media del modulo della velocità dell'accelerometro e la media della corrente assorbita al progredire dei cicli, si osserva una forte correlazione. I coefficienti di Pearson di entrambe le facce sono riportati di seguito:

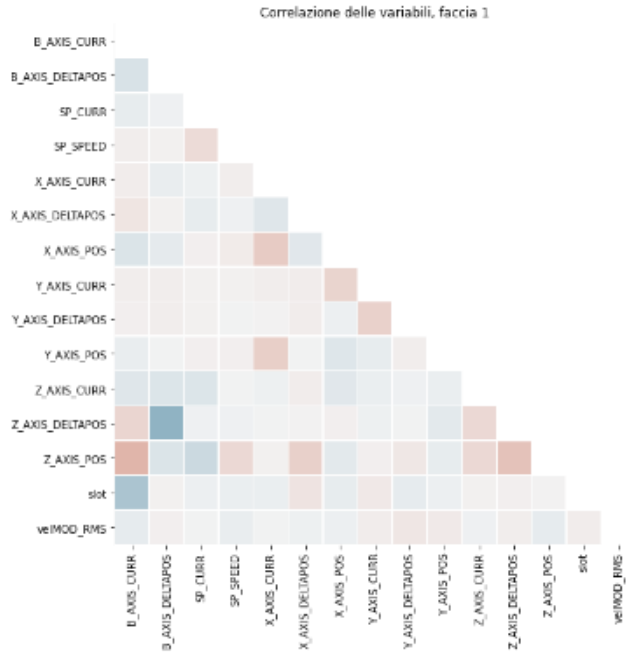


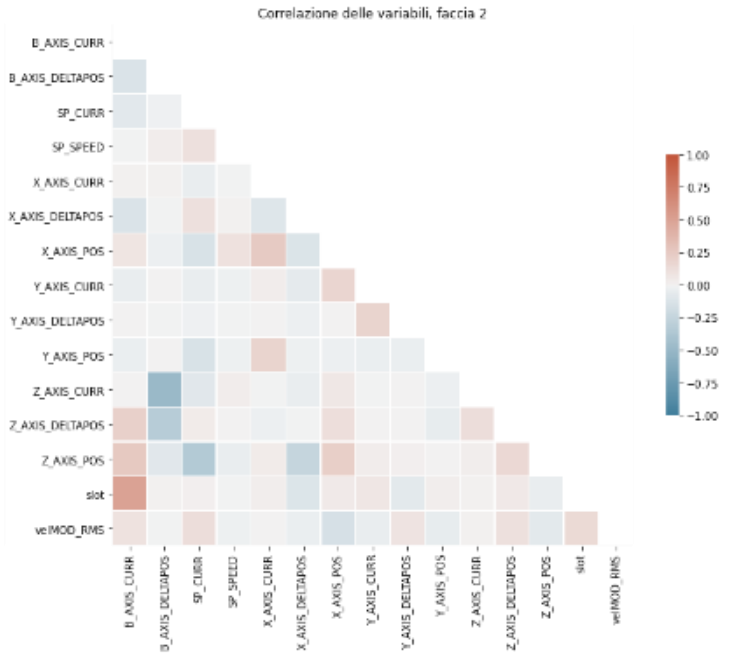
Coefficiente di correlazione faccia 1: 0.8457

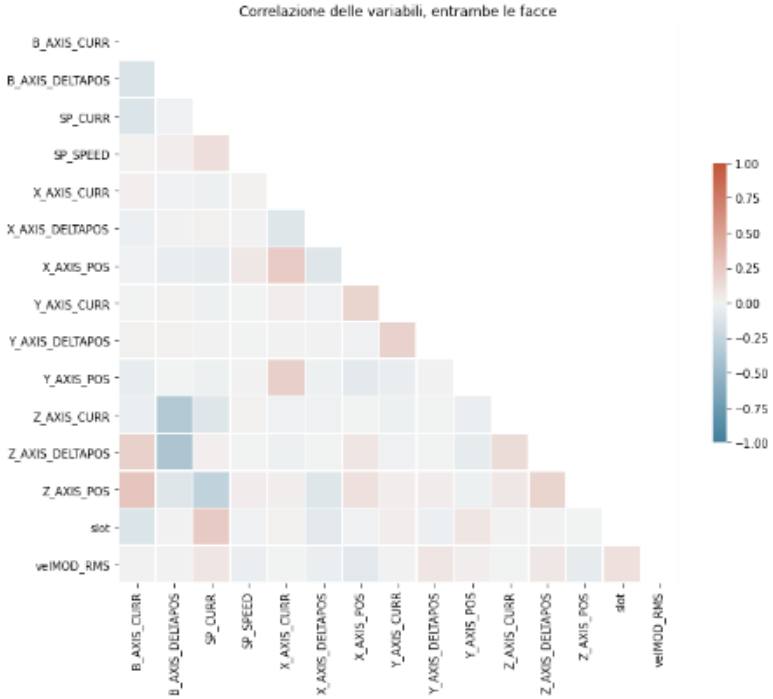
Coefficiente di correlazione faccia 2: 0.8353



Il comportamento rilevato non è invece stato riscontrato per le variabili degli assi. In generale, le variabili del *working-step* risultano essere debolmente correlate. Nelle figure seguenti si può osservare la correlazione relativamente alla faccia 1 e alla faccia 2 in tutti i cicli presi in esame, e la correlazione durante la lavorazione complessiva (faccia 1 + faccia 2). Dal calcolo è stata esclusa la variabile B_AXIS_POS perché costante: questa variabile rappresenta infatti la rotazione dell'asse della tavola su cui sono posti i pezzi, per poter effettuare la lavorazione su entrambe le facce.

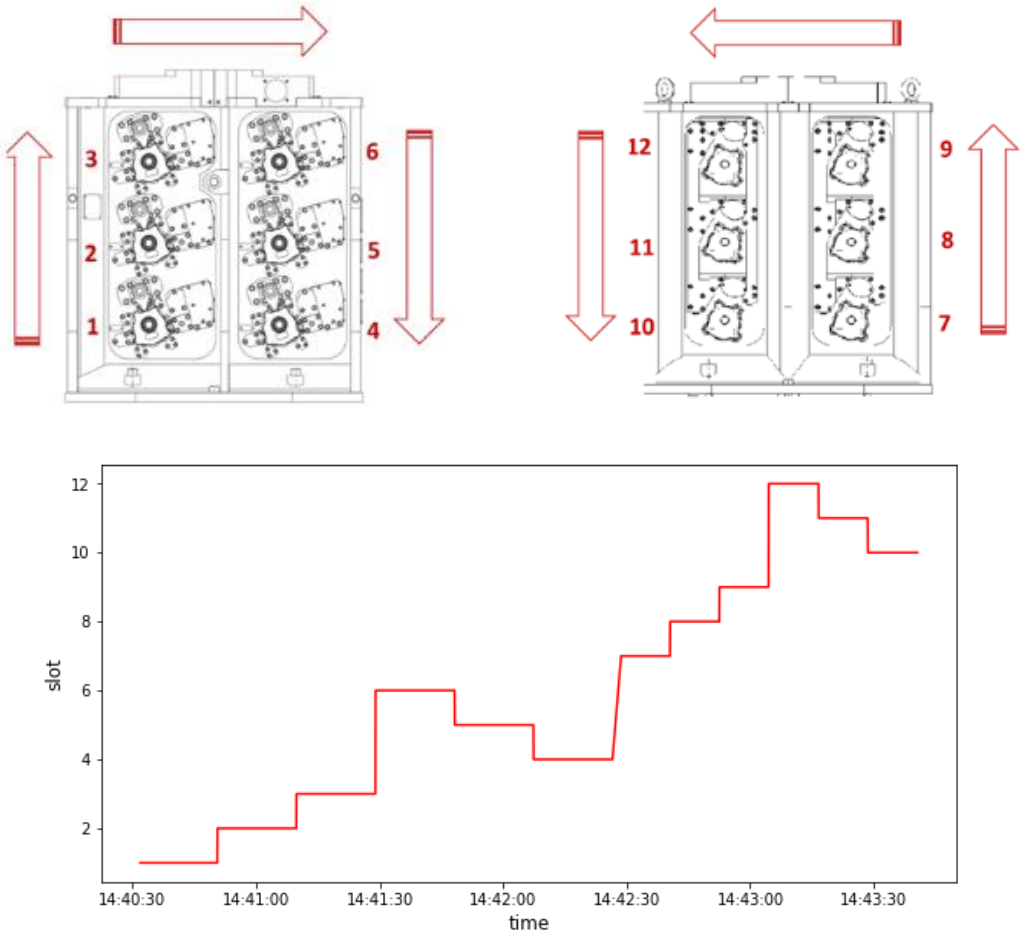

Correlazione delle variabili, faccia 1
B_AXIS_CURR
B_AXIS_DELTAPOS
SP_CURR
SP_SPEED
X_AXIS_CURR
X_AXIS_DELTAPOS
X_AXIS_POS
Y_AXIS_CURR
Y_AXIS_DELTAPOS
Y_AXIS_POS
Z_AXIS_CURR
Z_AXIS_DELTAPOS
Z_AXIS_POS
slot
velMOD_RMS


Correlazione delle variabili, faccia 2
B_AXIS_CURR
B_AXIS_DELTAPOS
SP_CURR
SP_SPEED
X_AXIS_CURR
X_AXIS_DELTAPOS
X_AXIS_POS
Y_AXIS_CURR
Y_AXIS_DELTAPOS
Y_AXIS_POS
Z_AXIS_CURR
Z_AXIS_DELTAPOS
Z_AXIS_POS
slot
velMOD_RMS
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Correlazione delle variabili, entrambe le facce
B_AXIS_CURR
B_AXIS_DELTAPOS
SP_CURR
SP_SPEED
X_AXIS_CURR
X_AXIS_DELTAPOS
X_AXIS_POS
Y_AXIS_CURR
Y_AXIS_DELTAPOS
Y_AXIS_POS
Z_AXIS_CURR
Z_AXIS_DELTAPOS
Z_AXIS_POS
slot
velMOD_RMS
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

3.2 Divisione per cicli

Un ciclo di lavorazione corretto è tale se rispetta l'ordine di lavorazione dei pezzi nelle due facce.



Il non rispetto di una sequenza corretta non è necessariamente da imputare ad un guasto alla macchina o al CNC; l'intero progetto di ammodernamento della macchina utensile in questione è ancora oggetto di miglioramenti e revisioni e pertanto, un errore di questo tipo, può essere causato da un bug nell'acquisizione dei dati da parte dei sensori, o in eventuali algoritmi di pre-processing eseguiti nel *recorder*. I cicli sono stati quindi etichettati, con un id crescente positivo se validi, negativo se non validi. La distinzione tra le due tipologie è stata fatta considerando i valori assunti dalla variabile **slot**. Per evitare quindi l'addestramento di un modello con dati di lavorazione non corretti, si è deciso di considerare solo i cicli validi, ma di mantenere l'informazione su eventuali cambi utensile o guasti presenti anche durante i cicli non validi.

Per maggiore chiarezza, analizziamo alcuni esempi di cicli non validi.

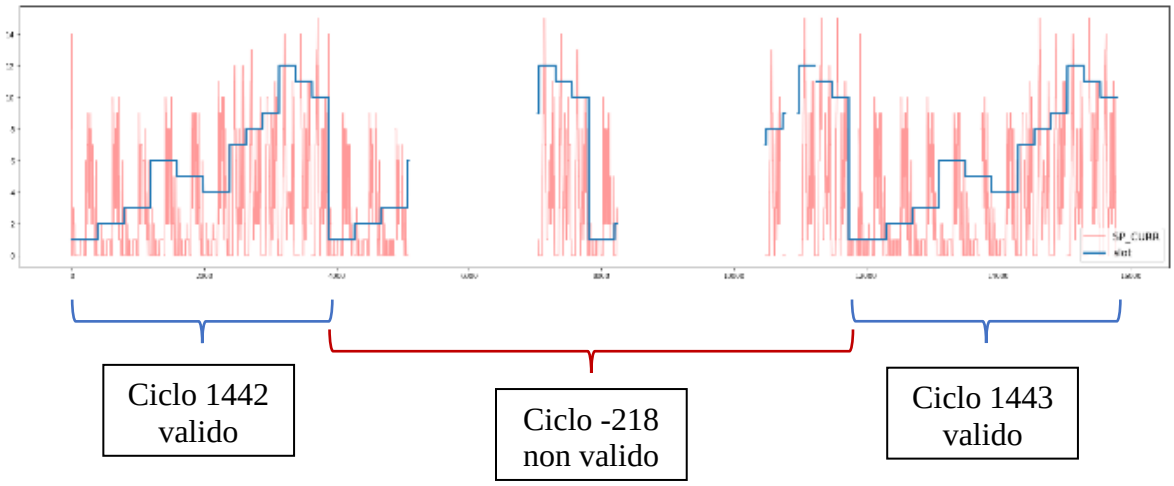


Figura 16

Nell'esempio in figura 16, il ciclo -218 non è valido. Come si può osservare, i valori assunti dalla variabile **slot** non sono conformi alla definizione di ciclo proposta, a causa di errori d'acquisizione dei dati. Questi errori si traducono in valori nulli per tutte le variabili contemporaneamente, pertanto se ne deduce che l'errore non è relativo al malfunzionamento di un singolo sensore ma al seguente salvataggio dei dati acquisiti.

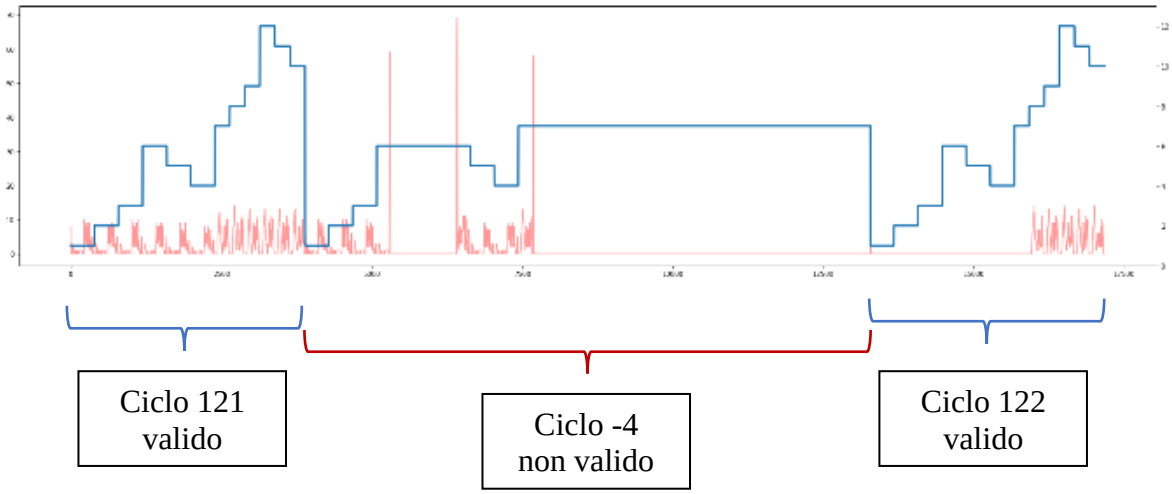


Figura 17

Nell'esempio in figura 17, il ciclo non valido è dovuto ad un errore del mandrino: come si può osservare dalla corrente assorbita, la lavorazione inizia correttamente sulla prima faccia; si ha quindi un picco di corrente seguito da un blocco; un altro picco fa ripartire la lavorazione degli ultimi 3 slot della prima faccia. Inizia quindi la lavorazione sulla seconda faccia ma si

interrompe subito, come nel caso precedente. La corrente del mandrino continua a rimanere a 0 durante gli slot della prima faccia nel nuovo ciclo fino all'inizio della seconda faccia, riprendendo la lavorazione da dove era stata interrotta. Le motivazioni legate a questo comportamento non sono chiare e non è stato registrato alcun guasto in questo periodo.

L'esempio in figura 18 ha un comportamento analogo a quello visto in figura 17. In particolare,

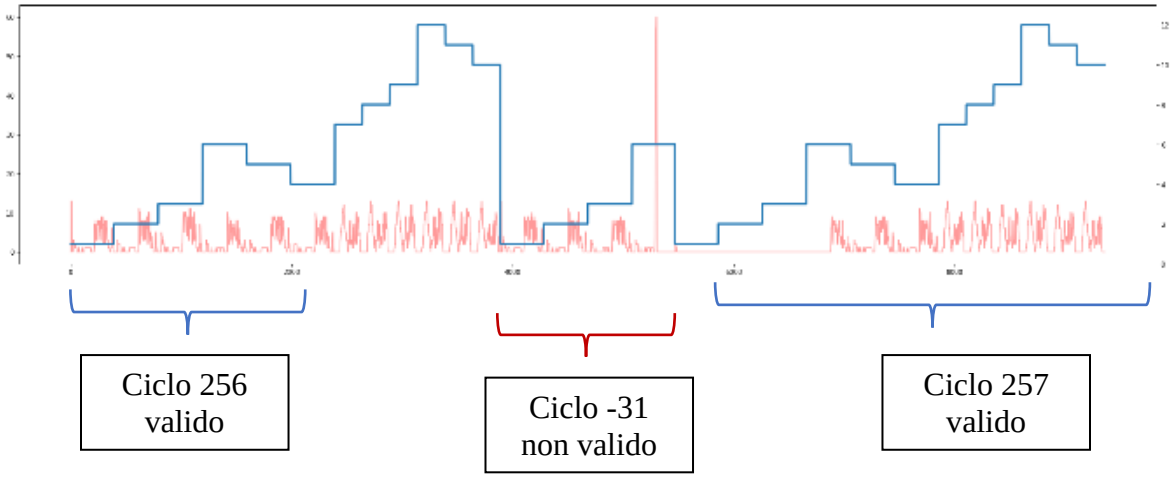
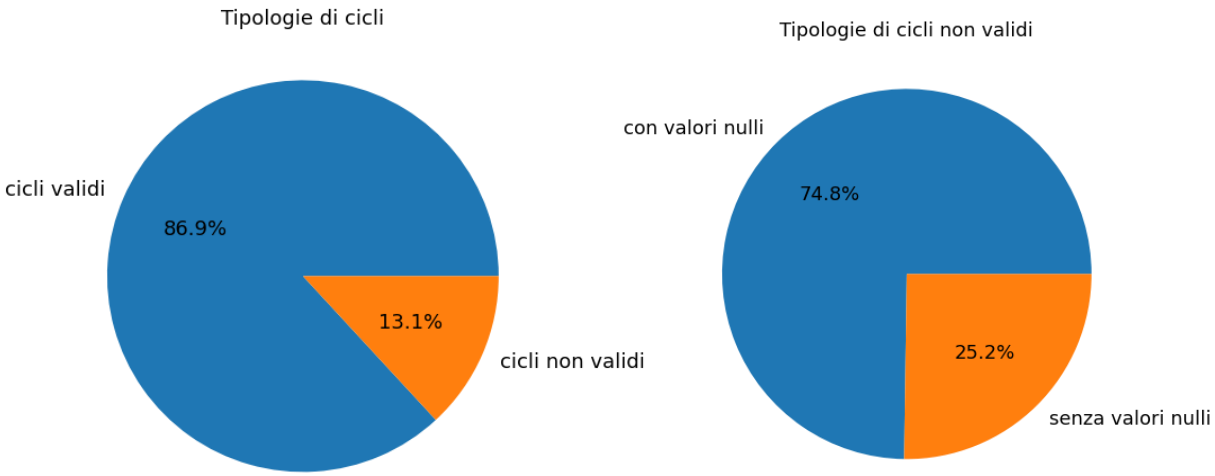
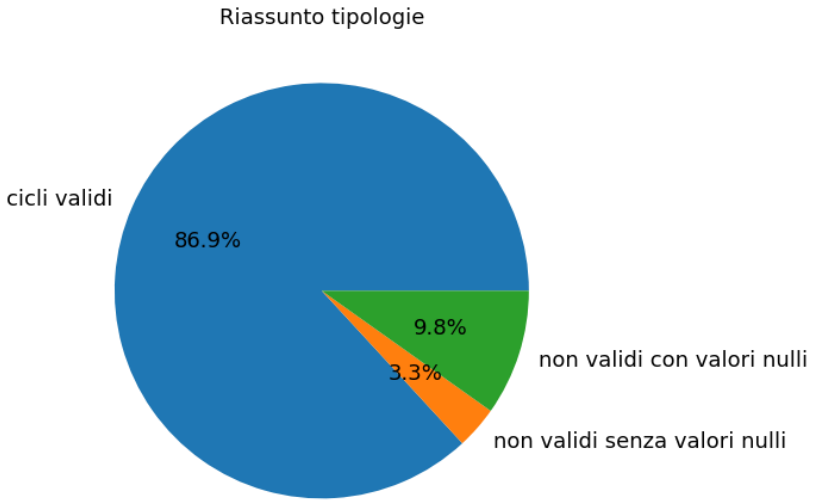


Figura 18

nel ciclo 257 la corrente assorbita dal mandrino è pari a 0 nei primi 3 slot, a differenza dell'esempio precedente in cui per tutta la prima faccia il mandrino non lavora. Entrambi i cicli (122 e 257) sono stati comunque etichettati come validi per limitare il numero di dati scartati, considerando anche il fatto che questa tipologia di cicli è molto ridotta. Si hanno infatti le seguenti statistiche:





3.3 Tempo di lavorazione complessivo ed effettivo

Un'analisi sui tempi di lavorazione può fornire informazioni di valore per migliorare l'efficienza del processo produttivo. Analizzando i tempi di lavorazione complessivi di un pallet nei differenti cicli, si osserva che essi hanno durate piuttosto differenti; in particolare, nei primi mesi di acquisizione dati, il tempo di lavorazione ha una durata media di 3 minuti. Gli ultimi due mesi sono invece caratterizzati da tempi di lavorazione molto più lunghi e meno stabili, con una media di 16 minuti e un valore massimo di 39 minuti. Il tempo di lavorazione complessivo non è calcolato in funzione di quanto la macchina effettivamente lavori, ma semplicemente con una differenza tra l'inizio della lavorazione sulla prima faccia e la fine della lavorazione sulla seconda; pertanto, in questo calcolo è riportato anche il tempo di fermo che intercorre tra le due facce. Le motivazioni dietro ad una così elevata differenza fra i tempi di lavorazione complessivi non sono ancora chiare all'azienda. Considerando invece il tempo di lavorazione effettivo, si osserva che durante tutto il periodo in analisi, la durata effettiva della lavorazione media è di circa 2 minuti, e pertanto un tempo di lavorazione complessivo così elevato è da ricercarsi unicamente nei fermi tra le due facce. Il tempo di lavorazione effettivo è calcolato nel seguente modo:

$$T_{eff} = ns_{SPCURR_{>0}} * sr$$

Dove **ns** è il numero di sample con corrente assorbita dal mandrino positiva e **sr** è il sampling rate (in millisecondi).

In particolare, distinguendo tra cicli validi e cicli non validi, si hanno le seguenti distribuzioni:

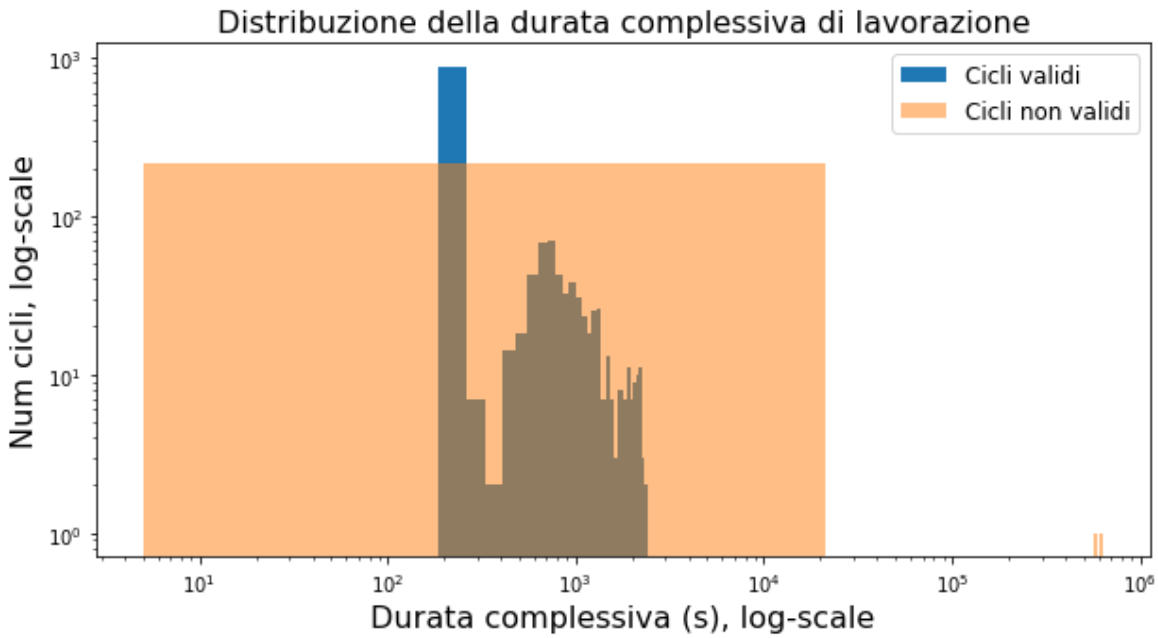


Figura 19

Come si può osservare dalla figura 19, i cicli non validi hanno un intervallo di durata complessiva maggiore; si ha una distribuzione piuttosto uniforme di cicli non validi di durata compresa tra pochi secondi e alcune ore, con degli outlier che durano alcuni giorni. Questi ultimi valori piuttosto stremi sono legati ad errori nell'acquisizione dei dati, con presenze ripetute di valori nulli che non permettono una corretta individuazione dei cicli validi, come già osservato.

In figura 20, analizzando i cicli validi, si osserva che la maggior parte dei cicli ha una durata effettiva di lavorazione di circa 120 secondi; meno di 30 cicli hanno invece una durata effettiva minore, e corrispondono a quei cicli che seguono un ciclo non valido che non presenta valori nulli, come mostrato nel paragrafo precedente. La distribuzione della durata effettiva dei cicli non validi è invece uniforme, in un intervallo che va dai pochi secondi a circa mezz'ora. Anche in questo caso è presente un outlier, la cui natura è legata a quella degli outlier della figura 19.

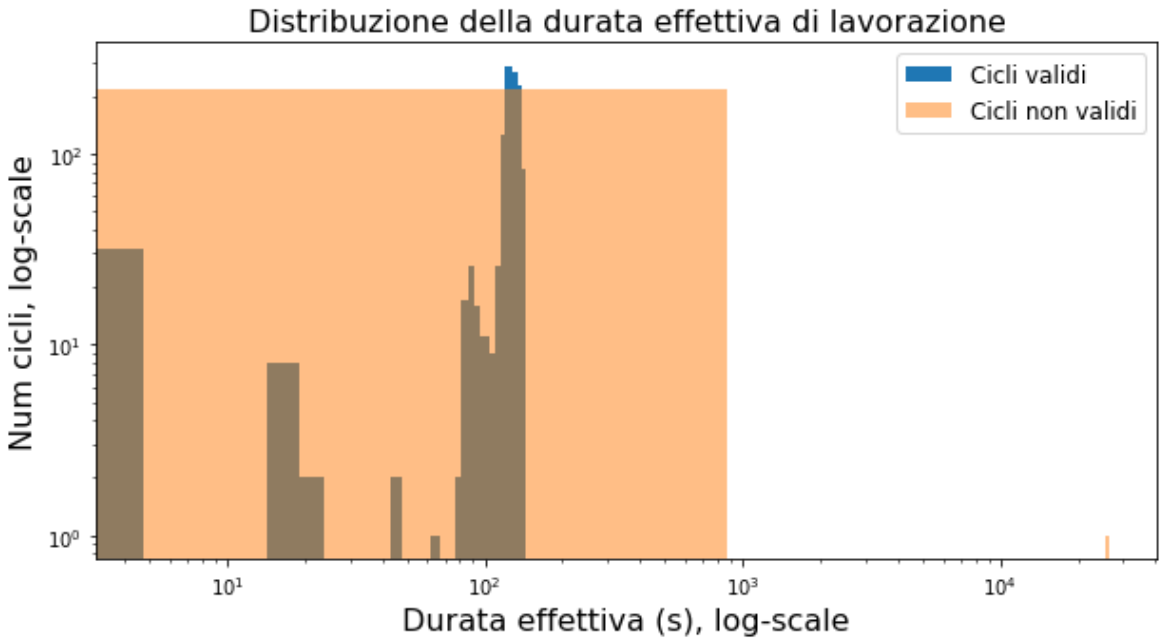


Figura 20

3.4 Remaining Working Sample Time

L'approccio di classificazione utilizzato è di tipo supervisionato, pertanto è necessario fornire al modello informazioni sui periodi antecedenti agli interventi di manutenzione e ai cambi utensile. A tal fine è utile inserire nel dataset l'informazione riguardo alla distanza tra il sample attuale e il prossimo evento in macchina. Da quanto visto nel paragrafo precedente, il tempo di lavorazione complessivo al prossimo evento non risulta un'ottima scelta, vista la variabilità di questo valore tra i cicli e la presenza del tempo di fermo tra le due facce. Il tempo di lavorazione effettivo, d'altra parte, riesce a dare un'idea di quanto ancora la macchina lavorerà, considerando l'entrata in funzione effettiva del mandrino. Con un approccio analogo a quello utilizzato per il calcolo di questo valore è possibile calcolare il tempo di lavorazione al prossimo evento; nel dettaglio:

- Si isolano i sample tra un evento in macchina e il successivo
- Si contano solo i sample con variabile SP_CURR positiva
- Si assegna, a partire dal primo sample e proseguendo verso l'evento, un valore decrescente iniziando dal numero di sample contati. Questa variabile indica i *remaining working samples (rms).*
    - Se la variabile SP_CURR di un sample è pari a 0, viene assegnato l'*rms* precedente

- Si moltiplica ogni *rms* per il sampling rate, ottenendo il *remaining working sample time (rmst)*.

	guasto	SP_CURR	rws_guasto	rwst_guasto
1582415	False	7.0	10.0	480.0
1582416	False	5.0	9.0	432.0
1582417	False	5.0	8.0	384.0
1582418	False	5.0	7.0	336.0
1582419	False	5.0	6.0	288.0
1582420	False	5.0	5.0	240.0
1582421	False	5.0	4.0	192.0
1582422	False	5.0	3.0	144.0
1582423	False	5.0	2.0	96.0
1582424	False	5.0	1.0	48.0
1582425	False	5.0	0.0	0.0
1582426	False	0.0	0.0	0.0
...				
1582442	False	0.0	0.0	0.0
1582443	False	0.0	0.0	0.0
1582444	False	0.0	0.0	0.0
1582445	True	0.0	0.0	0.0

Figura 21- Procedendo verso l'evento, rws ed rwst diminuiscono. L'ultimo sample con SP_CURR > 0 è indicato con una freccia

Si noti che arrivati al sample 1582425, *rws* ed *rwst* hanno valore 0: si tratta infatti dell'ultimo sample di effettiva lavorazione (SP_CURR > 0) prima dell'evento; infatti, tutti i 20 sample successivi prima dell'evento avranno SP_CURR = 0.

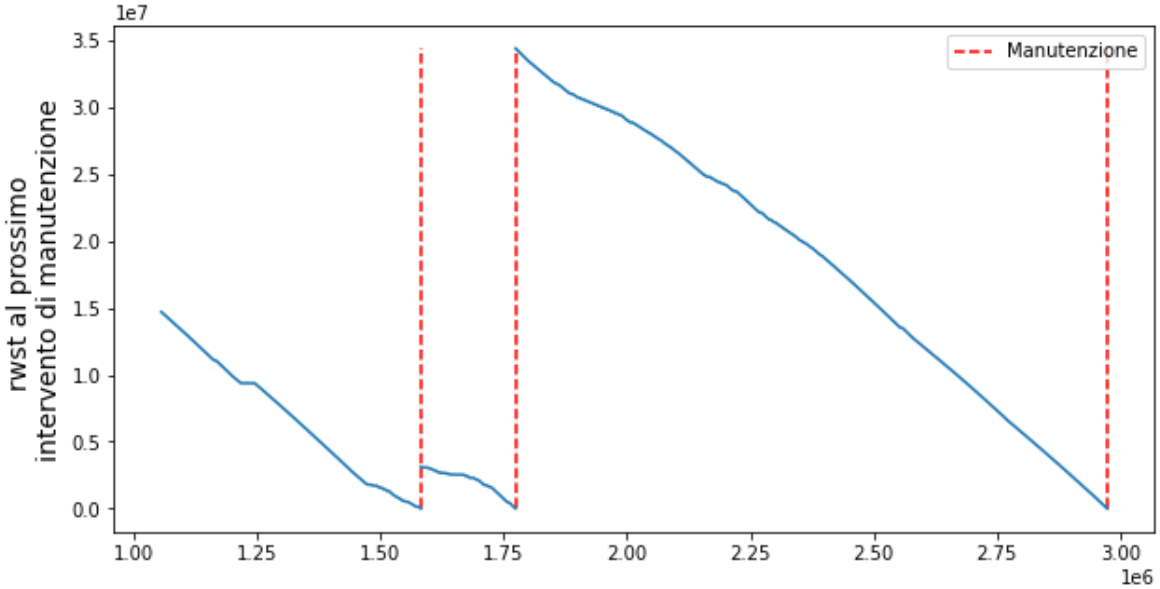


Figura 22 - Esempio di rwst all'approssimarsi degli eventi di manutenzione

Il calcolo dell'*rwst* è effettuato considerando sia i cicli validi che quelli non validi. Come vedremo nel prossimo paragrafo, la generazione del dataset da dare in input al modello supervisionato prende solo in considerazione i cicli validi; l'rwst così calcolato permette di non perdere l'informazione sugli eventi di manutenzione che avvengono nei cicli negativi.

3.5 Dataset aggregato

Il dataset attuale ha una dimensione di circa 8.5 milioni di sample; per poter allenare un modello di machine learning per la classificazione è necessario ridurne notevolmente le dimensioni, mantenendo comunque sufficiente informazione circa la variazione delle variabili all'avvicinarsi degli eventi in macchina. Per questo motivo, i cicli non validi sono stati scartati e sono state selezionate solo le variabili relative agli assi, al mandrino e all'accelerometro. Il dataset è stato quindi raggruppato per cicli e ciascun ciclo diviso tra prima e seconda faccia; sono state quindi applicate, per ciascuna faccia, le seguenti funzioni di aggregazione: media, mediana, deviazione standard, kurtosis, skewness, min e max.

Il dataset risultate è così costituito da tante righe quanti i cicli validi, e da un numero di colonne pari alle funzioni di aggregazione applicate al numero di variabili su entrambe le facce di ogni ciclo, nel dettaglio 1443 righe e 210 colonne. Vengono infine aggiunte delle colonne relative al

minimo *rwst* al prossimo evento in macchina, e alla presenza di uno specifico evento nel ciclo. In figura 23 un estratto del dataset ottenuto:

cycle_id	cambio	min_rwst_cambio	...	X_AXIS_CURR_std_1	X_AXIS_CURR_median_1	X_AXIS_CURR_kurtosis_1
1.0	False	8704320.0	...	3.792161	0.1099	76.987117
2.0	False	8567808.0	...	3.818011	0.0976	77.078304
3.0	False	8433120.0	...	3.816517	0.0732	76.887961
4.0	False	8292672.0	...	3.818156	0.1342	77.278631
5.0	False	8157792.0	...	3.837462	0.0611	74.695746

Figura 23

Capitolo 4

Nella manutenzione predittiva, diversi sono gli algoritmi utilizzabili per la classificazione: la scelta dipende essenzialmente dalla quantità dei dati a disposizione e dalla necessità o meno di avere informazioni riguardo le feature più informative. Per affrontare il problema si è scelto di utilizzare degli ensemble tree methods, ovvero degli algoritmi che utilizzano un insieme di alberi decisionali per effettuare le predizioni. Quest'approccio non richiede un preprocessing aggiuntivo dei dati (nessuna necessità di scaling delle variabili o di One Hot Encoding[1]) e permette di ottenere i risultati in tempo piuttosto rapido, fornendo informazioni sul grado d'importanza delle variabili nella costruzione del modello.

In questo capitolo, dopo aver effettuato un'operazione di labelling sul dataset aggregato, verranno illustrati due algoritmi per la classificazione: Random Forest e XGBoost. Verranno quindi applicati al problema trattato e se ne discuteranno i risultati mettendo a confronto i due modelli.

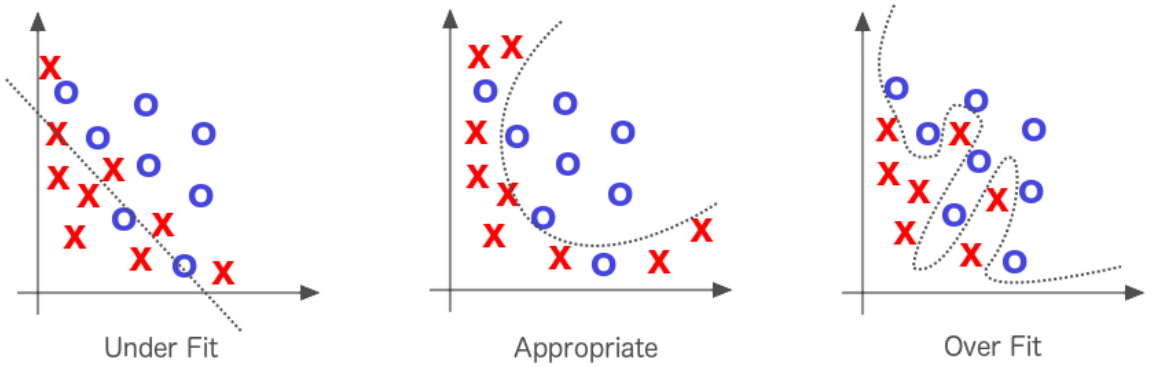
Gli algoritmi utilizzati per l'addestramento dei modelli e per il calcolo delle performance sono quelli forniti dalla libreria Scikit-Learn, ad eccezione di XGBoost.

4.1 Alcuni concetti base

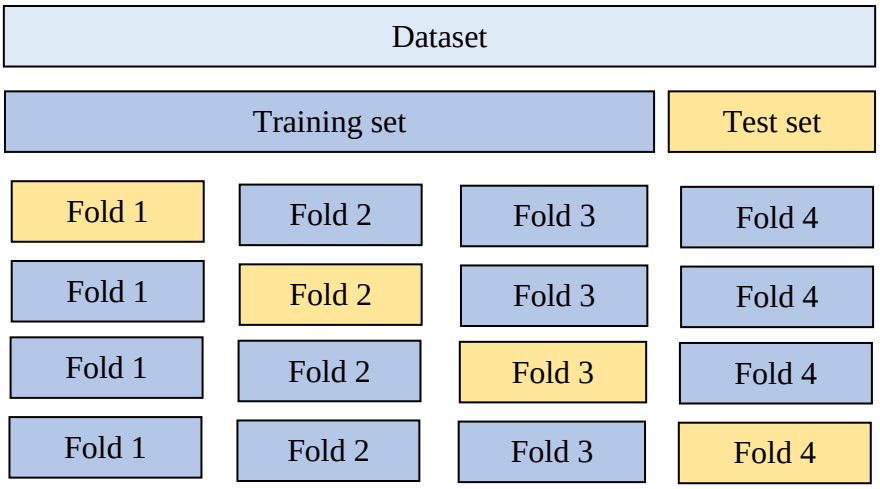
Qualunque modello allenato con un algoritmo supervisionato deve essere in grado di generalizzare: ricevuto in input un nuovo set di dati, con valori anche diversi da quelli utilizzati per l'allenamento, il modello dovrà fornire una predizione accurata. Pertanto, solo con buone capacità di generalizzazione, esso potrà avere utilità pratica. Per l'allenamento del modello, il dataset viene diviso in training, validation e test set con proporzioni differenti; l'allenamento viene effettuato con l'algoritmo scelto utilizzando il training set e la stima delle performance viene calcolata sul validation set, testando diverse configurazioni di parametri che il modello può assumere, detti *iperparametri*; individuati i migliori iperparametri, la capacità di generalizzazione del modello finale viene calcolata sul test set. In funzione della dimensione

[1] Tecnica utilizzata per mappare una variabile categorica con n valori, in n variabili booleane: ogni sample avrà una sola tra queste variabili con valore 1 e le altre a 0.

del training set, dei parametri utilizzati e di altri fattori, l'allenamento può causare *overfitting*, ovvero la creazione di un modello troppo complesso ed estremamente legato ai dati del training set, incapace quindi di generalizzare. Nel caso contrario, quando il modello è troppo semplice e produce scarse performance sia in training che in test set, si parla di *underfitting*.



Per poter ottenere un modello che abbia buone performance è quindi necessario individuare un insieme di parametri ottimali, ovvero effettuare un'operazione di tuning; a tal fine si può ricorrere al *Grid Search*, una tecnica che, forniti in input i valori da testare per ogni iperparametro, allena una serie di modelli utilizzando tutte le possibili combinazioni di iperparametri, restituendo infine il modello che ha dato il miglior risultato. Questa tecnica è spesso eseguita insieme alla *cross-validation*, procedura che permette di slegare l'allenamento e il test delle performance dei modelli da una particolare configurazione di training e validation set. Con quest'approccio, il training set viene diviso in k sottoinsiemi più piccoli, detti *fold*; il modello viene quindi addestrato su k-1 fold e la validazione è effettuata sul fold rimasto: questa procedura viene ripetuta cambiando di volta in volta il fold di validation e le performance complessive al termine del ciclo sono calcolate come media delle singole. Questa procedura viene applicata per ogni combinazione di iperparametri e può essere computazionalmente onerosa. Di seguito un'immagine riassuntiva della cross validation.



4.2 Labelling

Per poter allenare il modello è necessario definire una variabile target; una volta appresa la relazione tra i dati e il valore del target (label), il modello sarà in grado di classificare ogni nuovo dato in input, privo della variabile target, assegnandone la label opportuna. Nel caso in esame, per poter generare il target è stata utilizzata la variabile di minimo *rwst* all'evento: il fine della classificazione applicata è quello di individuare in anticipo il verificarsi di un certo evento in macchina, e pertanto sarà necessario etichettare i sample immediatamente precedenti ad un cambio utensile o ad una manutenzione, per distinguerli da quelli in cui la lavorazione non è influenzata dal loro approssimarsi. Fissata quindi una soglia temporale *th*, si aggiunge al dataset una colonna di label con i seguenti valori:

$$\begin{cases} 1 & rwst_{ev} \leq th \\ 0 & altrimenti \end{cases}$$

Nell'esempio in figura 24 è stata fissata una soglia temporale di 30 minuti di lavorazione effettiva prima del prossimo guasto.

	label	min_rwst_guasto (millis)	min_rwst_guasto (minuti)
5	0	2366352.0	39.0
6	0	2228304.0	37.0
7	0	2089296.0	35.0
8	0	1950576.0	33.0
9	1	1815072.0	30.0
10	1	1689216.0	28.0
11	1	1504512.0	25.0
12	1	1386768.0	23.0

Figura 24

Al variare della soglia, la percentuale di sample etichettati a 1 varia. La scelta della soglia dipende essenzialmente dal tipo di evento che si ha intenzione di predire; nel contesto applicativo, per gli eventi di manutenzione risulterebbe utile se la soluzione proposta avvisasse gli operatori uno o due giorni prima la necessità di un intervento. Questo tempo scende a circa un'ora o poco meno per i cambi utensile. Considerando che la variabile di soglia utilizzata è il tempo di lavorazione effettivo rimanente (*rwst*) e considerate le distribuzioni di valori per ciclo

visti nel capitolo 3.3, la soluzione proposta prevede una soglia di 1 ora per gli interventi di manutenzione e di 15 minuti per i cambi utensile.

4.3 Random Forest

Gli alberi decisionali non permettono di rappresentare modelli complessi con un buon livello di generalizzazione: all'aumentare della complessità, infatti, si ha l'overfitting del modello. Per ovviare a questo problema, si può ricorrere ad una Random Forest [9]. Una Random Forest è generata a partire da un insieme di alberi decisionali costruiti, in maniera indipendente, in un sottospazio randomico delle variabili. In questo modo, ogni albero sarà in grado di generalizzare la classificazione su una parte del dataset, esplorando configurazioni differenti di variabili. La classificazione finale è ottenuta tramite una moda matematica: la classe individuata da più alberi risulterà essere il risultato complessivo della predizione. In figura 25 uno schema dell'algoritmo.

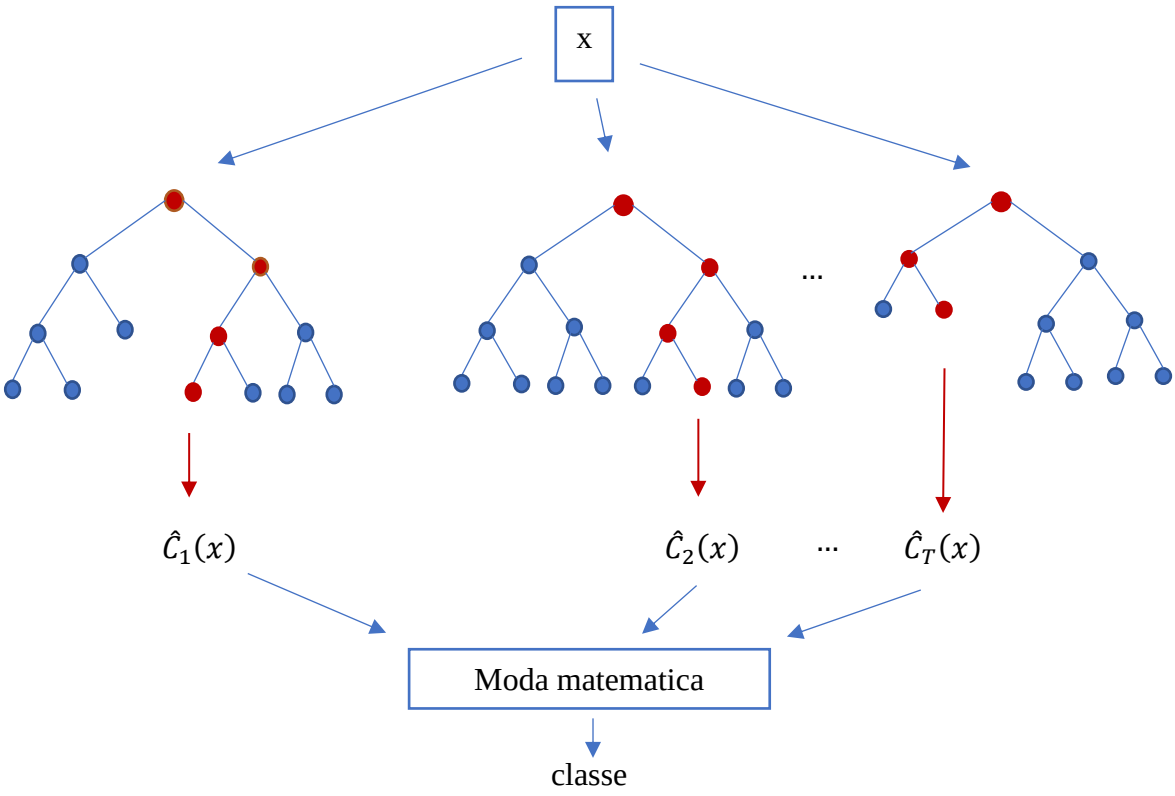


Figura 25

Nel corso degli anni, sono state proposte variazioni ed estensioni dell'algoritmo introdotto da Ho in [9] nel 1995, come ad esempio l'utilizzo di un metodo di bootstrap aggregation (bagging)

che permette di ridurre la varianza del modello, riducendo la correlazione tra gli alberi [10]. Analizziamo nel dettaglio quest'ultima versione dell'algoritmo proposta da Breiman nel 2001.

Gli alberi decisionali, cresciuti ad una certa profondità, risultano essere modelli rumorosi ma con un basso bias e pertanto sono ottimi candidati per il bagging, la cui idea di base è quella di ridurre la varianza dei modelli con queste caratteristiche. In particolare, il bagging consiste nella generazione, a partire dal training set D, di B training set di dimensione $|B_n| \leq |D|$; ogni training set viene generato effettuando un campionamento uniforme casuale con reinserimento di sample da D, con la conseguenza che all'interno del nuovo training set saranno presenti dei duplicati. Grazie al bagging, ciascun albero sarà quindi creato su un training set unico e quindi identicamente distribuito; per tale motivo, il bias del modello sarà uguale a quello del singolo albero, e quindi l'unico miglioramento che può essere fatto è sulla varianza. Se si presuppone che le variabili siano indipendenti e identicamente distribuite, ciascuna con varianza σ^2, la varianza della media su T alberi sarà $\frac{1}{T}\sigma^2$. Se però viene meno l'indipendenza, sarà necessario tener conto della correlazione a coppie delle variabili, ρ. In tal caso la varianza della media sarà [11]:

$$\rho\sigma^2 + \frac{1-\rho}{T}\sigma^2$$

All'aumentare del numero di alberi T, il secondo termine scompare, e la varianza media del modello sarà legata solo alla correlazione ρ. Per ridurre la correlazione, Ho propone [9] di generare gli alberi a partire da un sottospazio delle variabili, la cui selezione è casuale. Quindi, prima di ogni split, la variabile candidata per il test viene scelta da un sottoinsieme $m \leq p$, dove p sono le variabili del training set originale. In particolare, per la classificazione, il valore consigliato per m è $\lfloor\sqrt{p}\rfloor$ e la dimensione minima di ogni nodo è 1.

L'algoritmo complessivo è riportato di seguito:

1. Per b = 1, …, T:
    a) Si crea un bootstrap training dataset Z* di dimensione N a partire dal training set originale.
    b) Si crea un albero T_b a partire da Z* ripetendo i seguenti step fino a quando la dimensione minima dei nodi non è stata raggiunta:
        i. Si selezionano casualmente m variabili dalle p variabili del training set originale

ii. Si seleziona la variabile più adatta per lo split tramite un opportuno criterio
iii. Si divide il nodo in due nodi figli

2. Si generano gli output degli alberi $\{T_b\}_1^T$
3. Per classificare un nuovo sample ogni albero predice una label e alla fine si effettua una votazione: detta $\widehat{C_b}(x)$ la label predetta dal b-esimo albero, la predizione complessiva sarà:

$$\hat{C}^T{}_{rf}(x) = moda\left\{\hat{C}_b(x)\right\}_1^T$$

4.3.2 Random Forest: interventi di manutenzione

Per la predizione degli interventi di manutenzione è stata applicata una soglia di 1 ora: in questo modo il numero di sample con label 0 è 1210, quelli con label 1 sono 233. Per trovare il miglior modello, il tuning degli iperparametri è stato effettuato con un Grid Search esaustivo: ogni modello è stato addestrato usando la cross validation e le performance sono state valutate su un validation set. I parametri più importanti per il tuning del modello, e i rispettivi valori trovati, sono riportati nella seguente tabella.

Iperparametri	**Valore**	**Significato**
max_depth	10	Profondità massima degli alberi
n_estimators	100	Numero di alberi
criterion	entropy	Criterio per lo split dei nodi
max_features	$\sqrt{m}$	Numero massimo di features da scegliere casualmente
min_samples_split	5	Numero minimo di sample in un nodo interno per eseguire uno split
Min_samples_leaf	3	Numero minimo di sample per creare una foglia

In figura 26 sono riportati i risultati della predizione sul test set in funzione degli interventi di manutenzione (linee tratteggiate rosse). In verde si hanno le label: la soglia utilizzata è di 1 ora di lavorazione effettiva; i valori predetti sono visualizzati con degli asterischi blu.

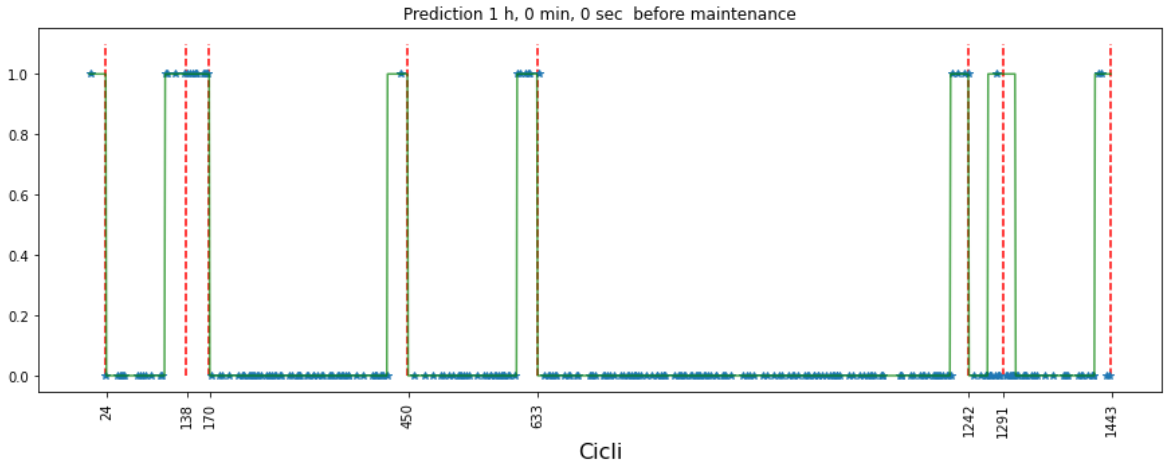


Figura 26

Come si può osservare il modello è riuscito a predire con una certa accuratezza gli interventi di manutenzione; in particolare, gli interventi ai cicli 138, 170, 633 e 1242 sono stati correttamente classificati con una buona continuità, l'intervento al ciclo 1291 presenta invece molti falsi negativi. Le capacità predittive del modello sono state valutate tramite accuracy score, calcolato nel seguente modo:

$$accuracy(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(\hat{y}_i = y_i)$$

L'esperimento in figura 26 ha riportato le seguenti accuratezze:

- Training test: 98.44 %
- Test set: 93.07 %

Nel caso della manutenzione predittiva, maggiore importanza riveste il numero di falsi positivi. Pensando infatti all'applicazione pratica della soluzione trovata, nel caso in cui si dovesse presentare un falso negativo, si ricadrebbe nel caso precedente all'applicazione della soluzione, cioè alla politica di *run to failure.* Un falso positivo, al contrario, implica il blocco della lavorazione e un'operazione di ispezione non dovuta, con un ovvio costo economico. Per avere una chiara idea della capacità predittiva del modello è possibile ricorrere ad una matrice di confusione, ovvero una tabella le cui righe rappresentano i valori veri e le colonne i valori predetti (o viceversa). In figura 27 è riportata la matrice di confusione del modello trovato:

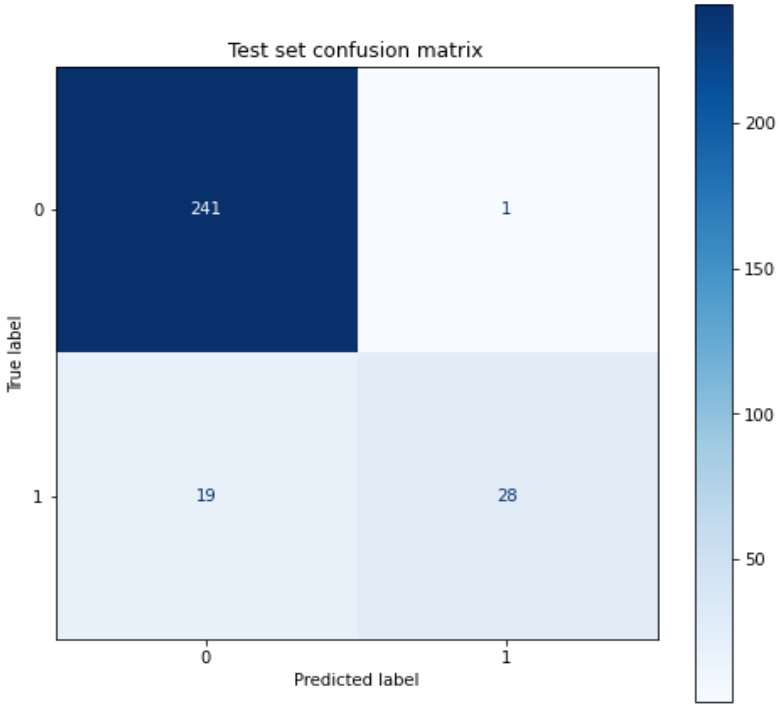


Figura 27

In funzione dei valori riportati nella matrice di confusione è possibile avere maggiori informazioni sulla bontà del modello. In particolare è possibile calcolare i valori di *precision*, *recall* e *specificity*. La *precision* indica la proporzione di veri positivi predetti tra tutti i sample individuati come positivi:

$$precision = \left(\frac{TP}{TP + FP}\right)$$

Quest'indice è molto rilevante nel momento in cui il costo dei falsi positivi è elevato, come nel caso in esame.

La *recall* invece indica la proporzione di veri positivi predetti, sul totale dei sample positivi:

$$recall = \left(\frac{TP}{TP + FN}\right)$$

Diversamente dalla *precision*, quest'indice è molto rilevante nel momento in cui il costo dei falsi negativi è elevato. Infine, la *specificity* può essere vista come il corrispettivo della *recall* per le label negative: essa indica infatti la proporzione di sample predetti negativi sul totale di tutti i sample con label negativa

$$specificity = \left(\frac{TN}{TN + FP}\right)$$

Come già sottolineato, nel problema trattato sono importanti valori di *precision* elevati; al contrario, una *recall* media può essere tollerata. Si ha:

- Precision: 96.55%
- Recall: 59.57%
- Specificity: 99.58%

La divisione del dataset è effettuata casualmente, con l'80% dei sample in training e il 20% in test set; all'interno di ogni set è stata mantenuto lo stesso rapporto di sample con label 0 e con label 1. Effettuando 1000 iterazioni di split del dataset, apprendimento e predizione su test set, è stata ottenuta un'accuratezza media di (95 ± 1.2) %.

4.3.3 Random Forest: cambi utensile

Per la predizione dei cambi utensile la soglia è stata posta a 15 minuti; in questo modo il numero di sample con label 0 è 1089, contro 354 sample con label 1. I migliori parametri trovati con il tuning sono riportati di seguito:

Iperparametri	**Valore**	**Significato**
max_depth	6	Profondità massima degli alberi
n_estimators	200	Numero di alberi
criterion	entropy	Criterio per lo split dei nodi

max_features	$\sqrt{m}$	Numero massimo di features da scegliere casualmente
min_samples_split	5	Numero minimo di sample in un nodo interno per eseguire uno split
Min_samples_leaf	1	Numero minimo di sample per creare una foglia

In figura 28 è riportata la predizione del miglior modello trovato: in rosso tratteggiato i cambi utensile per fine vita, in blu tratteggiato quelli per usura, mentre in verde il ground truth. Le predizioni sono rappresentate da asterischi. Come si può osservare, molti cambi non sono stati predetti dal modello: si tratta nella maggior parte dei casi di cambi dovuti a fine vita; questo comportamento sottolinea quanto l'usura degli utensili abbia un pattern comune nella variazione delle variabili, che il modello riesce ad apprendere con facilità.

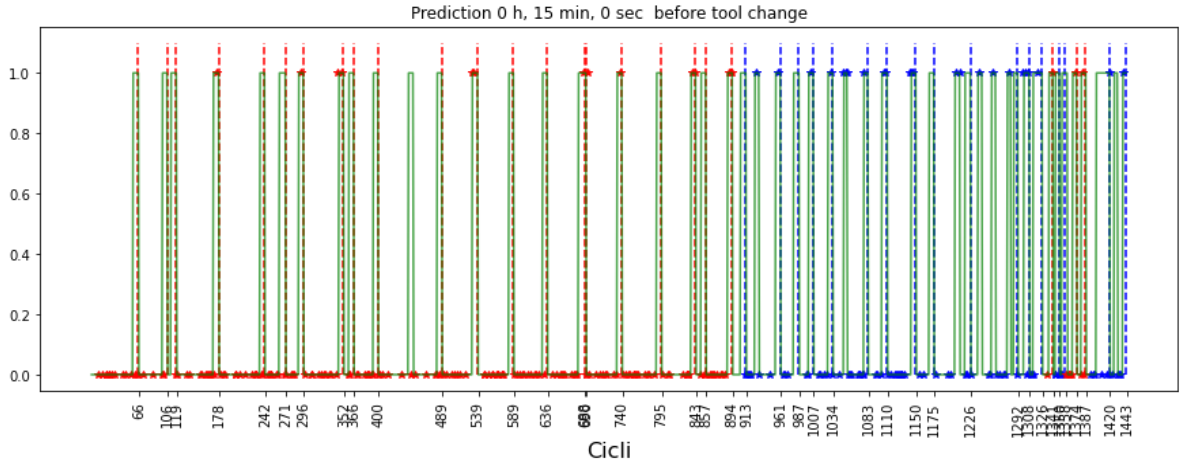


Figura 28

L'esperimento ha dato i seguenti valori di accuratezza:

- Training set: 90.98 %
- Test set: 85.46 %

In figura 29 la matrice di confusione dell'esperimento.

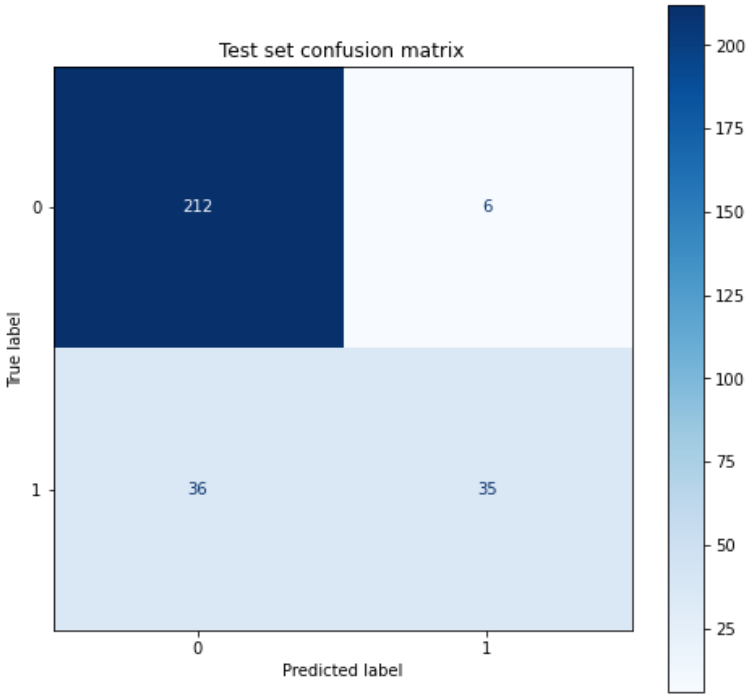


Figura 29 – Matrice d confusione dell'esperimento

Dalla confusion matrix si ha:

- Precision: 85.36%
- Recall 49.29%
- Specificity: 97.24%

Come per gli interventi di manutenzione, è stato effettuato il fit del modello su diverse istanze di training e test set, ed è stata calcolata un'accuratezza media, pari a $(83.48 \pm 1.65)\%$.

4.4 XGBoost

XGBoost è un algoritmo di tree boosting altamente scalabile che permette di ottenere un'elevata accuratezza in una grande varietà di problemi, come ad esempio la classificazione di eventi fisici ad alta energia, il comportamento dei consumatori, la percentuale di abbandono dei corsi online, e altro ancora. Al giorno d'oggi rappresenta lo stato dell'arte degli ensemble tree methods [12] grazie alla rapidità d'esecuzione su macchine singole e all'estrema scalabilità che

gli permette di operare su miliardi di sample in sistemi distribuiti. Il nome XGBoost sta per "Extreme Gradient Boosting"; esso rientra infatti nella categoria di algoritmi di tree boosting tramite discesa del gradiente ed impiego di weak learners. Come vedremo tra poco, a differenza dell'algoritmo di Random Forest, gli alberi decisionali non sono creati in parallelo in maniera indipendente, ma in modo sequenziale, andando di volta in volta a migliorare lo scarto tra i valori osservati e i valori predetti dagli alberi precedenti. Inoltre, a differenza dell'algoritmo di Random Forest, dove il risultato finale è ottenuto per moda matematica (classificazione) o per media (regressione) degli output, in XGBoost esso è ottenuto per somma pesata degli output dei singoli alberi. Analizziamo nel dettaglio l'algoritmo:

Sia dato $D = \{(x_i, y_i)\}$ $(|D| = n, x_i \in \mathbb{R}^m, y_i \in \mathbb{R})$, dataset con n esempi ed m features. Come accennato, la predizione dell'output sarà data da K funzioni additive, ovvero:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

Dove $\mathcal{F} = \{f(x) = w_{q(x)}\}$ $(q: \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ rappresenta lo spazio di regressione dell'albero. Nella definizione di tale spazio q rappresenta la struttura di ogni albero, T è il numero delle foglie di ciascuno di essi e w_q il valore in output dato da una certa struttura q ad un certo input. Affinché il modello impari le funzioni in $\mathcal{F}$ è necessario minimizzare la seguente funzione obiettivo regolarizzata:

$$\mathcal{L}(\phi) = \sum_i l(y_{i,}\hat{y}_i) + \sum_k \Omega(f_k)$$

$$con\ \Omega(f) = \gamma T + \frac{1}{2}\lambda||w||^2$$

Dove l è una loss function differenziabile convessa che permette di misurare la differenza tra il valore predetto ed il valore osservato e $\Omega(f)$ penalizza un modello troppo complesso: in particolare il primo termine incoraggia il pruning dell'albero, il secondo ne riduce l'overfitting tramite un parametro di regolarizzazione. Dato che la predizione procede per somma degli output degli alberi, la predizione $\hat{y}_i^{(t)}$ della i-esima istanza del dataset del t-esimo albero, sarà data dalla somma tra la predizione dell'albero precedente (t-1) e la f_t. Si ha pertanto:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_{i,}\hat{y}_i^{(t-1)} + f_{t(x_i)}\right) + \ \Omega(f_t)$$

La f_t aggiunta è scelta con una strategia greedy, cercando di minimizzare la funzione obiettivo $\mathcal{L}^{(t)}$. Per maggiore semplicità è possibile approssimare la $\mathcal{L}^{(t)}$ tramite uno sviluppo in serie di Taylor del secondo ordine:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} [l\left(y_i, \hat{y}^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

$$\text{dove } g_i = \frac{\partial}{\partial \hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right) \ e \ h_i = \frac{\partial^2}{\partial \hat{y}^{(t-1)2}} l\left(y_i, \hat{y}^{(t-1)}\right)$$

sono il gradiente e l'hessiana della loss function dell'albero precedente.

Dato che il primo termine della funzione obiettivo non dipende dal valore di output $f_t(x_i)$ è possibile rimuoverlo. Esplicitando $\Omega(f_t)$, si ha:

$$\tilde{\mathcal{L}}^{(t)} \ = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \ \gamma T + \frac{1}{2} \lambda \left|\left|w\right|\right|^2$$

$$\tilde{\mathcal{L}}^{(t)} \ = \sum_{j=1}^{T} [G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2] + \gamma T$$

Dove con G_j e H_j sono indicate le somme di tutti i g_i ed h_i della j-esima foglia del t-esimo albero. Per una fissata struttura q è possibile calcolare il valore del peso ottimale w_j minimizzando $\tilde{\mathcal{L}}$. Si ha:

$$w_j^* = \ -\frac{G_j}{H_j + \lambda}$$

Sostituendo questo valore nella funzione obiettivo approssimata si ottiene:

$$min\tilde{\mathcal{L}}^{(t)}(q) = \ -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

ovvero la minima funzione obiettivo. Quanto ottenuto può essere utilizzato come scoring function (*similarity score*) per misurare la qualità della struttura q dell'albero. Essendo impossibile l'enumerazione di tutte le strutture generabili, si applica una strategia greedy che mira ad individuare, a partire da una singola foglia, lo split ottimale, ovvero quello che fornisce un gain maggiore rispetto al nodo padre. Il gain sarà calcolato come differenza tra la somma dei similarity score delle due foglie e il similarity score del padre, ovvero:

$$Gain_{split} = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_r+\lambda} - \frac{G_{R\cup L}^2}{H_{R\cup L}+\lambda}\right] - \gamma$$

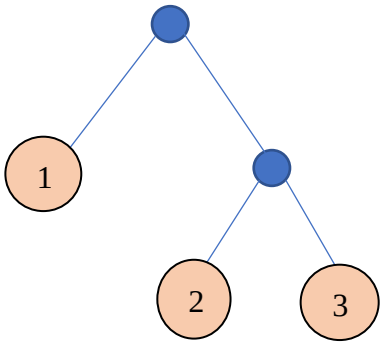
Per maggiore chiarezza consideriamo il seguente albero.

La sua minima funzione obiettivo è:

$$\tilde{\mathcal{L}}_{before} = -\frac{1}{2}\left[\frac{G_1^2}{H_1+\lambda} + \frac{G_2^2}{H_2+\lambda} + \frac{G_3^2}{H_3+\lambda}\right] + 3\gamma$$

Effettuiamo ora uno split del nodo 1 e ricalcoliamo la funzione obiettivo:

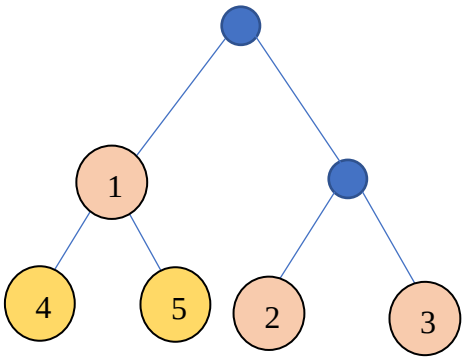
$$\tilde{\mathcal{L}}_{after} = -\frac{1}{2}\left[\frac{G_4^2}{H_4+\lambda} + \frac{G_5^2}{H_5+\lambda} + \frac{G_2^2}{H_2+\lambda} + \frac{G_3^2}{H_3+\lambda}\right] + 4\gamma$$



La differenza tra $\tilde{\mathcal{L}}_{before}$ ed $\tilde{\mathcal{L}}_{after}$ restituisce

il $Gain_{split,}$ ovvero:

$$Gain_{split} = \frac{1}{2}\left[\frac{G_4^2}{H_4+\lambda} + \frac{G_5^2}{H_5+\lambda} - \frac{G_1^2}{H_1+\lambda}\right] - \gamma$$



Di seguito un riassunto dell'algoritmo:

1. Per t = 1, …, K:
    a. Si inseriscono tutti i residui nella prima foglia calcolati come differenza tra i valori osservati e i valori predetti precedentemente

b. Si genera l'albero con strategia greedy
    i. Si sfrutta il similarity score dei nodi interessati per trovare il migliore nodo su cui effettuare lo split
    ii. Si procede nella generazione dell'albero fino a quando i nodi non raggiungono la dimensione minima. L'albero così generato minimizza la funzione obiettivo.
c. Si calcola il peso ottimale in ogni foglia
d. Per ogni sample x_i, si procede sommando i pesi di tutti gli alberi relativi alle foglie in cui x_i è stato assegnato. Questa somma è moltiplicata per un learning factor ε per diminuire la possibilità di overfitting.
e. Ottenuta la predizione, si calcolano i nuovi residui

2. Raggiunto il numero massimo K di alberi, il modello è stato allenato e nuovi sample possono essere predetti.

Generalmente, la prima predizione $\hat{y}_0$ è posta a 0.5, sia per la regressione che per la classificazione.

4.4.2 XGBoost: interventi di manutenzione

Come per la Random Forest, anche in XGBoost il tuning degli iperparametri è stato effettuato applicando un grid search esaustivo e testando i risultati su un validation set. Gli iperparametri del miglior modello sono riportati nella tabella di seguito:

Iperparametri	**Valore**	**Significato**
max_depth	3	Profondità massima degli alberi
n_estimators	200	Numero di alberi
reg_lambda	1	Valore λ di regolarizzazione
eta	0.1	Learning factor ε
gamma	0	Parametro γ che favorisce il pruning
min_child_weight	3	Numero minimo di pesi in una foglia
colsample_bytree	1	Frazione di features da essere scelte casualmente nello lo split

In figura 30 sono riportati i risultati ottenuti con il miglior modello trovato, rappresentati con asterischi blu. Come si può osservare, il modello è riuscito ad individuare con una certa continuità tutti gli interventi di manutenzione. Sono presenti, tuttavia, un cospicuo numero di falsi positivi.

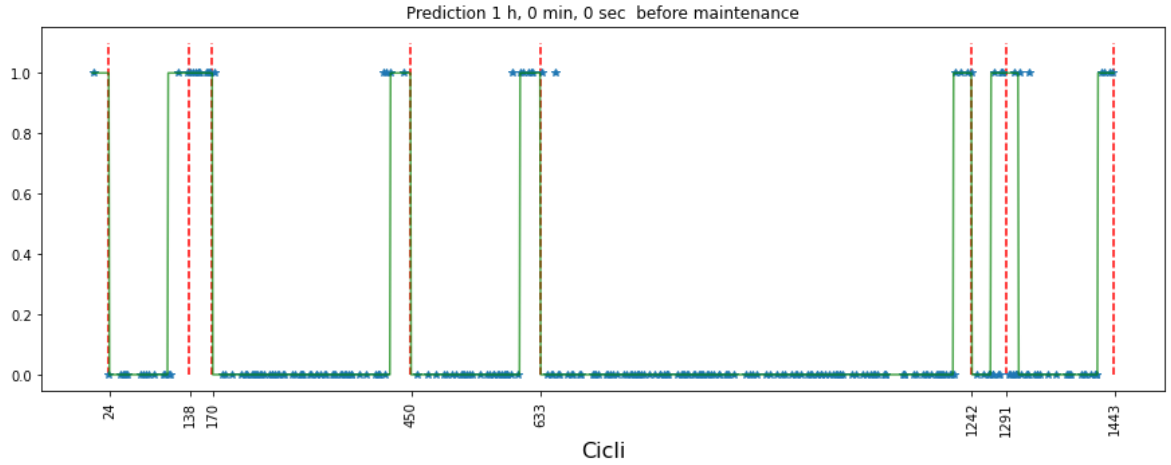


Figura 30

I valori di accuratezza calcolati per questo esperimento sono i seguenti:

- Training set: 97.31%
- Test set: 92.38%

In figura 31 la matrice di confusione.

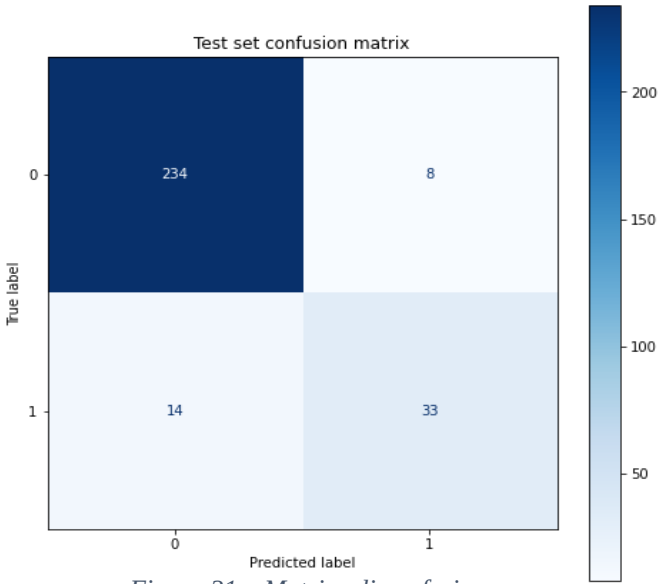


Figura 31 – Matrice di confusione dell'esperimento

Dalla matrice di confusione si ha:

- Precision: 80.48%
- Recall: 70.21%

- Specificity: 96.69%

In generale, il modello riesce a predire correttamente i risultati, con un'accuratezza media, su 1000 iterazioni, pari a $(93.06 \pm 1.35)\%$.

Nel paragrafo 4.6 verranno analizzati i risultati confrontando le performance tra i due modelli nella predizione degli eventi di manutenzione.

4.4.3 XGBoost: cambi utensile

Per la predizione dei cambi utensile è stata utilizzata una soglia di 15 minuti, ottenendo 354 sample con label 1 e 1089 con label 0. I migliori parametri trovati con il tuning sono riportati nella tabella seguente:

Iperparametri	**Valore**	**Significato**
max_depth	3	Profondità massima degli alberi
n_estimators	200	Numero di alberi
reg_lambda	3	Valore λ di regolarizzazione
eta	0.5	Learning factor ε
gamma	3	Parametro γ che favorisce il pruning
min_child_weight	2	Numero minimo di pesi in una foglia
colsample_bytree	0.3	Frazione di features da essere scelte casualmente nello lo split

In figura 32 è riportato il risultato delle predizioni del miglior modello trovato; i cambi per fine vita sono tratteggiati in rosso, quelli per usura in blu. Come si può osservare, a differenza di quanto visto con Random Forest, questo modello è riuscito ad individuare un numero maggiore di cambi utensile per fine vita; la predizione dei cambi per usura risultano essere altrettanto buoni al modello precedente.

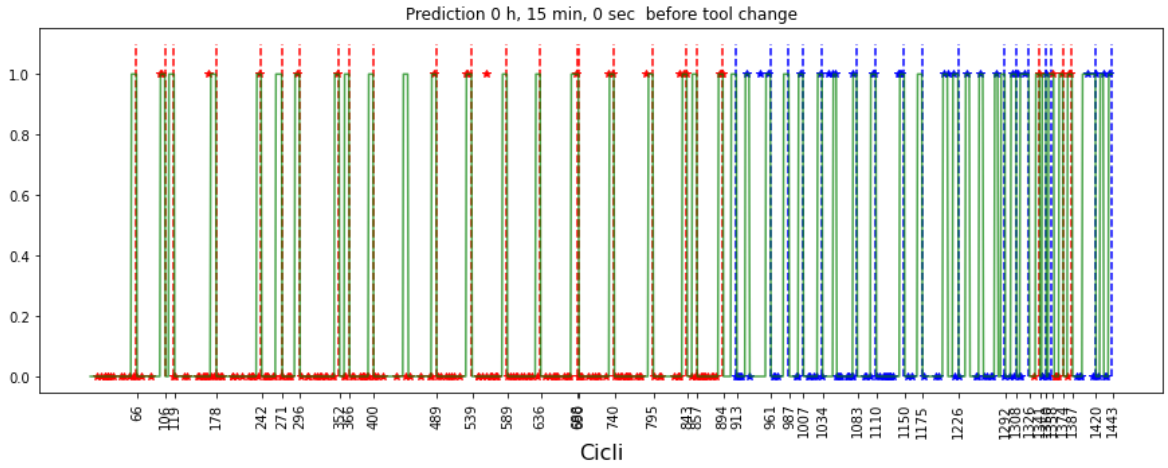


Figura 32

Nell'esperimento riportato si hanno i seguenti valori di accuratezza:

- Training set: 90.38%
- Test set: 87.19%

In figura 33 è riportata la matrice di confusione. Si hanno i seguenti valori:

- Precision: 81.48%
- Recall: 61.97%
- Specificity: 95.41%

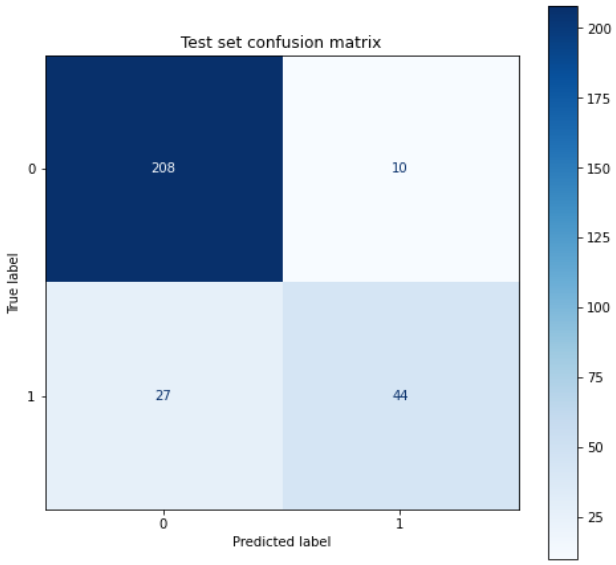


Figura 33 – Matrice di confusione dell'esperimento

Su 1000 iterazioni, l'accuratezza calcolata sul test set è di $(82.65 \pm 1.97)\%$.

Nel paragrafo 4.6 verranno analizzati i risultati confrontando le performance tra i due modelli nella predizione degli eventi di manutenzione.

4.5 Il problema dell'accuratezza

Come osservato dai risultati, l'accuratezza risulta particolarmente alta indipendentemente dal numero di veri positivi individuati. Dalla formula riportata di seguito, un dataset non bilanciato (come nel caso in esame) tenderà ad annullare quasi totalmente il contributo di una delle due classi, portando comunque ad un'accuratezza elevata.

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

Nel dataset utilizzato, come visto nel paragrafo 4.1, il numero di classi positive e negative è influenzato dalla soglia temporale applicata: più è grande l'intervallo temporale considerato maggiore sarà il numero di sample positivi, come osservabile nelle tabelle

Inteventi di manutenzione		
$rwst_{th}$	**#pos**	**#neg**
3h	533	910
2h	391	1052
1h 30'	315	1128
1h	233	1210
30'	121	1322
15'	64	1379

Cambi utensile		
$rwst_{th}$	**#pos**	**#neg**
1h	1078	365
30'	655	788
20'	466	977
15'	354	1089
10'	251	1192
5'	144	1299

È importante precisare che la scelta della soglia deve essere effettuata tenendo a mente il fine applicativo e ricordando che il tempo di lavorazione effettivo (*rwst*) non coincide con il tempo reale, ma dipende da quanto il mandrino lavora.

Nel caso in esame l'accuratezza non sembra il miglior strumento per valutare la bontà dei modelli; nella stima sarà necessario inserire dei pesi per premiare o per penalizzare i risultati.

Si può ricorrere ad una funzione di costo atteso, utilizzando dei costi medi di manutenzione e d'ispezione.

Si ha:

$$Costo_{atteso} = p(TP) \times costo_{manutenzione} + p(FP) \times costo_{ispezione} + p(TN) \times 0$$

$$+\alpha \cdot p(FN) \times \left(costo_{ispezione} + costo_{manutenzione}\right)$$

$$con\ \alpha \in]0,1]$$

Il parametro alfa permette di attenuare la penalità dei falsi negativi: un falso negativo corrisponde ad una politica *run-to-failure* e pertanto, indipendentemente dalla segnalazione ricevuta dal modello, la macchina è destinata a fermarsi; al contrario, nel caso di un falso positivo l'interruzione è del tutto superflua e deleteria, e dovrebbe ricevere una penalizzazione maggiore.

Per quanto riguarda i cambi utensile possiamo utilizzare una funzione analoga, con la sola differenza che i falsi negativi riceveranno una penalizzazione maggiore dovuta alla non segnalazione di un cambio utensile dovuto ad usura: in tal caso, infatti, la macchina continuerebbe a lavorare con un utensile difettoso, producendo dei pezzi di qualità inferiore allo standard. Si ha:

$$costo_{atteso} = p(TP) \times costo_{sostituzione} + p(FP) \times costo_{ispezione} + p(TN) \times 0$$

$$+ p(FN) \times \left(costo_{sostituzione} + \beta \cdot costo_{prodotti}\right)$$

$$con\ \beta = \frac{cambi_{usura}}{cambi_{fine\ vita} + cambi_{usura}}$$

Dove con $costo_{prodotti}$ ci si riferisce al costo legato alla lavorazione e allo scarto di prodotti non conformi agli standard, a causa di un utensile troppo usurato.

Se il numero di cambi per usura è ridotto, i falsi negativi non riceveranno una grossa penalizzazione e l'utensile verrà sostituito per fine vita. A differenza degli interventi di

manutenzione, infatti, la soluzione proposta non va a sostituire la politica finora utilizzata ma ne integra il funzionamento con l'individuazione e la segnalazione della necessità di un cambio utensile per usura o per fine vita effettiva; pertanto la soglia massima di pezzi per la sostituzione rimane fissa a 300 unità lavorate, e qualora questa soglia dovesse essere raggiunta senza alcuna segnalazione (come nel caso dei falsi negativi) l'utensile verrebbe sostituito.

4.6 Confronto tra i modelli

Entrambi i modelli hanno riportato ottimi risultati nella classificazione degli eventi in macchina, nonostante la ridotta dimensione del dataset. Per stimarne le performance è possibile ricorrere alle funzioni di costo atteso introdotte precedentemente, e alle curve ROC (*receiver operating characteristics*) per indagare quanto i modelli siano in grado di distinguere tra lavorazione normale e lavorazione a ridosso di un evento.

Non avendo a disposizione informazioni riguardo ai singoli costi, è possibile effettuare una stima, ipotizzando che:

$$costi_{prodotto} \geq costi_{manutenzione} \geq costi_{sostituzione} \geq costi_{ispezione}$$

In virtù di ciò, fissando i valori dei costi in una scala da 3 a 10 e ponendo rispettivamente $\alpha = 0.5$ e $\beta = 0.44$ (si hanno 20 cambi utensile per usura e 25 per fine vita) e calcolando la media dei costi attesi in 500 esperimenti si ha:

Intervento di manutenzione	
Modello	**Costo atteso**
Random Forest	0.79 ± 0.01
XGBoost	0.85 ± 0.05

Sostituzione utensile	
Modello	**Costo atteso**
XGBoost	1.78 ± 0.05
Random Forest	1.84 ± 0.05

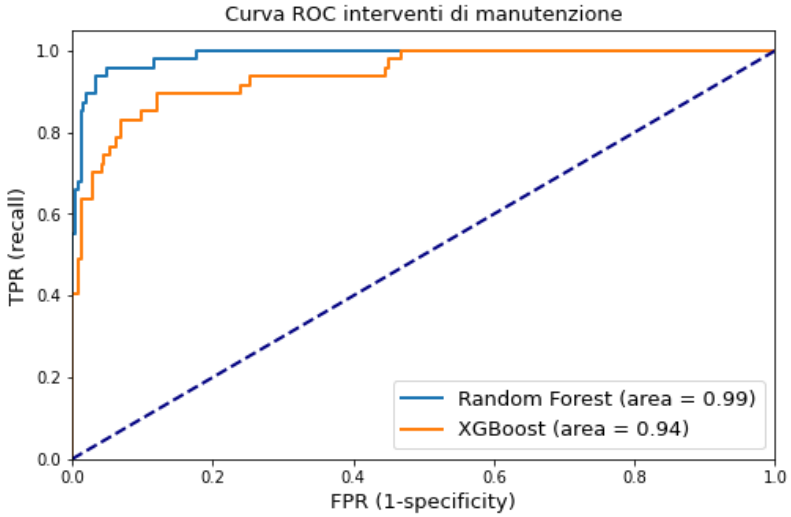
Come si può osservare, l'algoritmo Random Forest risulta leggermente migliore nella classificazione degli interventi di manutenzione rispetto ad XGBoost; la situazione inversa si verifica invece con l'individuazione anticipata dei cambi utensile. Osservando le formule dei costi attesi risulta che più è alta la capacità di separazione dei modelli, minore sarà il costo, dal momento che, con una separazione perfetta, l'unica componente sarà quella dei veri positivi. Ha pertanto senso disegnare la curva ROC e calcolarne l'area sottesa. La curva ROC, *receiver*

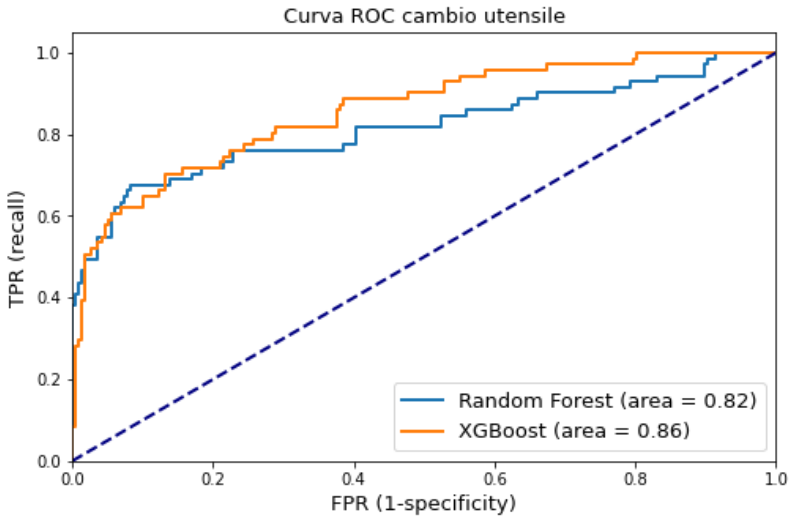
operating curve, è una curva di probabilità disegnata in funzione dei valori di TPR (*true positive rate*) e di FPR (*false positive rate*) assunti al variare della soglia di predizione. In particolare si ha:

$$TPR = \frac{TP}{TP + FN}, \qquad FPR = \frac{FP}{TN + FP}$$

Il TPR è già stato introdotto nel paragrafo 4.2.2 con il termine di *Recall*, mentre $FPR = 1 - Specificity$. L'area sottesa la curva, AUC (*Area Under The Curve*) è indice di quanto il modello sia in grado di classificare correttamente le classi: essendo la ROC una curva di probabilità, l'AUC assume valori compresi tra 0 e 1; un'AUC di 1 è ottenuta da un modello ideale, perfettamente in grado di predire con una precisione del 100% la classe corretta; un'AUC di 0, al contrario, è ottenuta da un modello che riesce sempre a predire la classe opposta, non individuando mai TP e TN.

Le curve ROC dei due modelli per la predizione degli interventi di manutenzione e dei cambi utensile sono riportate nelle figure seguenti:





Come già osservato con i costi attesi, anche se la differenza è piuttosto ridotta, Random Forest risulta un algoritmo migliore per l'attuale istanza del problema relativamente agli interventi di manutenzione, ottenendo un'AUC di 0.99. XGBoost, al contrario, è più performante con i cambi utensile, con un'AUC di 0.86 contro lo 0.82 dell'altro modello. In conclusione, entrambi gli algoritmi utilizzati sono risultati incredibilmente validi nonostante la ridotta dimensione del dataset; disponendo di un numero di dati maggiore, raccolti in anni di lavorazione piuttosto che in mesi, si potranno ottenere modelli sempre più in grado di individuare un comportamento anomalo nelle variabili, riuscendo ad anticipare con grande precisione gli eventi in macchina.

Conclusione

In questo lavoro di tesi è stato presentato un approccio reale alla manutenzione predittiva su macchine utensili in un contesto industriale. A partire dalla raccolta, dalla pulizia e dalla creazione di un dataset unico è stata condotta un'esplorazione dei dati che ha portato all'individuazione di alcuni comportamenti delle variabili durante la lavorazione ordinaria della macchina. Soffermandoci quindi sui tempi di lavorazione, l'individuazione di una variabile, il *remaining working sample time,* è stata necessaria per effettuare il labelling e procedere con la generazione di un dataset aggregato, le cui dimensioni fossero compatibili con l'addestramento di un modello di machine learning. Dopo aver individuato negli ensemble tree methods degli algoritmi di classificazione ideali al caso in esame, è stato effettuato il tuning degli iperparametri ed infine sono stati condotti differenti esperimenti ricampionando di volta in volta nuovi sample per training e test set. In questo modo sono stati realizzati dei modelli capaci di predire, con ottime performance, la necessità di un intervento di manutenzione sulla macchina o di una sostituzione dell'utensile.

La soluzione proposta è aperta ad eventuali modifiche da parte degli esperti del contesto applicativo, come ad esempio la scelta di una soglia temporale diversa da quelle proposte per i due eventi, strettamente legata all'organizzazione e alla preparazione delle squadre di manutenzione. Altro parametro di customizzazione è la soglia di predizione, che può essere modificata per ridurre il numero di falsi positivi aumentando i falsi negativi, col fine di ridurre le interruzioni non necessarie nel processo di lavorazione.

L'approccio utilizzato è solo uno dei tanti con cui è possibile affrontare un problema di manutenzione predittiva; ad esempio, per il problema in esame, alternativamente all'utilizzo della variabile *rwst*, sarebbe stato altrettanto valido, un approccio di anomaly detection: utilizzando un autoencoder [13] addestrato a ricostruire i dati acquisiti durante la lavorazione normale della macchina, sarebbero stati etichettati come positivi tutti quei sample individuati come anomalie, ovvero quei sample che l'autoencoder non sarebbe riuscito a replicare. Un'altra possibile metodologia, ad esempio, prevede l'utilizzo di un sistema di classificatori multipli come illustrato in [8], addestrati con diverse soglie temporali (orizzonti), con lo scopo di minimizzare un'opportuna funzione obiettivo. In generale, la scelta degli algoritmi, delle modalità e delle funzioni di aggregazione del dataset, nonché dell'individuazione di una variabile per il labelling, dipendono esclusivamente dal contesto applicativo e dalla quantità di

dati a disposizione. Con il caso studio proposto si è voluto inoltre sottolineare come, indipendentemente da ciò, l'analisi dei dati e l'applicazione di algoritmi di intelligenza artificiale, con un'architettura che li supporti, abbiano grande impatto nel processo di lavorazione in ambiente industriale, fornendo conoscenza aggiunta sulle performance e sui possibili miglioramenti tecnici e organizzativi che l'azienda può adottare per ottimizzare ogni step.

Bibliografia

[1] S. Vaidya, A. Prashant e B. Santosh , «Industy 4.0 - A Glimpse,» *Procedia Manufacturing*, p. 233–238, 2018.

[2] B. Fox e A. Subic, «An Industry 4.0 Approach to the 3D Printing of Composite Materials,» *Engineering*, n. 5, pp. 621-623, 2019.

[3] N. Grammatico, «Macchine CNC: Cosa sono e Come funzionano,» Ronchini Massimo srl, [Online]. Available: https://www.ronchinimassimo.com/it/macchine-cnc-cosa-sono-e-come-funzionano/.

[4] M. Deans, «What is Computer Aided Manufacturing (CAM)?,» Autodesk, [Online]. Available: https://www.autodesk.com/products/fusion-360/blog/computer-aided-manufacturing-beginners/.

[5] «CNC Post Processor,» CAMplete Solutions inc, [Online]. Available: https://camplete.com/cnc-post-processor/.

[6] «Usura degli utensili da taglio: come controllarla,» Tecnoutensili Decca, 29 11 2018. [Online]. Available: https://www.tecnoutensilidecca.it/utensileria/informativa/usura-utensili-da-taglio/.

[7] «Meccanica Tecnica,» 4 12 2018. [Online]. Available: https://meccanicatecnica.altervista.org/durata-e-curva-di-usura-del-utensile/.

[8] G. Susto, A. Schirru, A. Pampuri, A. McLoone e A. Beghi, «Machine Learning for Predictive Maintenance: A Multiple classifiers approach,» *IEEE Transactions on Industrial Informatics*, vol. 3, n. 11, pp. 812-820, 2015.

[9] T. K. Ho, «Random Decision Forest,» in *Proceedings of 3rd international conference on document analysis and recognition*, Murray Hill, NJ 07974, USA, 1995.

[10] L. Breiman, «Random Forest,» *Machine Learning*, n. 45, pp. 5-32, 2001.

[11] J. H. Friedman, R. Tibshirani e T. Hastie, «Random Forest,» in *The Elements of Statistical Learning*, Springer, 2001, pp. 587-589.

[12] T. Chen e C. Guestrin, «XGBoost: A Scalable Tree Boosting System,» in *KDD '16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco , 2016.

[13] P. Baldi, I. Guyon, G.Dror, V.Lemaire e G.Taylor, «Autoencoders, unsupervised learning, and deep architectures,» in *JMLR: Workshop and Conference Proceedings*, 2012.